NETWORK ARCHITECTURE FOR LARGE-SCALE DISTRIBUTED VIRTUAL ENVIRONMENTS

by

Tatiana Vladimir Balikhina

A thesis submitted in partial fulfillment of the requirements of Oxford Brookes University for the degree of
Doctor of Philosophy

Oxford Brookes University
Department of Computing
School of Technology
Wheatley Campus
Oxford, OX33 1HX
UK

March, 2005

Abstract

Distributed Virtual Environments (DVEs) provide 3D graphical computer generated environments with stereo sound, supporting real-time collaboration between potentially large numbers of users distributed around the world. Early DVEs has been used over local area networks (LANs). Recently with the Internet's development into the most common embedding for DVEs these distributed applications have been moved towards an exploiting IP networks.

This has brought the scalability challenges into the DVEs evolution. The network bandwidth resource is the more limited resource of the DVE system and to improve the DVE's scalability it is necessary to manage carefully this resource. To achieve the saving in the network bandwidth the different types of the network traffic that is produced by the DVEs have to be considered.

DVE applications demand exchange of the data that forms different types of traffic such as a computer data type, video and audio, and a 3D data type to keep the consistency of the application's state. The problem is that the meeting of the QoS requirements of both control and continuous media traffic already have been covered by the existing research. But QoS for transfer of the 3D information has not really been considered. The 3D DVE geometry traffic is very bursty in nature and places a high demands on the network for short intervals of time due to the quite large size of the 3D models and the DVE application requirements to transmit a 3D data as quick as possible.

The main motivation in carrying out the work presented in this thesis is to find a solution to improve the scalability of the DVE applications by a consideration the QoS requirements of the 3D DVE geometrical data type.

In this work we are investigating the possibility to decrease the network bandwidth utilization by the 3D DVE traffic using the level of detail (LOD) concept and the active networking approach.

The background work of the thesis surveys the DVE applications and the scalability requirements of the DVE systems. It also discusses the active networks and multiresolution representation and progressive transmission of the 3D data.

The new active networking approach to the transmission of the 3D geometry data within the DVE systems is proposed in this thesis. This approach enhances the currently applied peer-to-peer DVE architecture by adding to the peer-to-peer multicast network layer filtering of the 3D flows an application level filtering on the active intermediate nodes. The active router keeps the application level information about the placements of users. This information is used by active routers to prune more detailed 3D data flows (higher LODs) in the multicast tree arches that are linked to the distance DVE participants.

The exploration of possible benefits of exploiting the proposed active approach through the comparison with the non-active approach is carried out using the simulation-based performance modelling approach. Complex interactions between participants in DVE application and a large number of analyzed variables indicate that flexible simulation is more appropriate than mathematical modelling. To build a test bed will not be feasible.

Results from the evaluation demonstrate that the proposed active approach shows potential benefits to the improvement of the DVE's scalability but the degree of improvement depends on the users' movement pattern. Therefore, other active networking methods to support the 3D DVE geometry transmission may also be required.

Acknowledgments

I am extremely grateful to my director of study, Professor David Duce, for his excellent guidance and valuable advice. I sincerely wish to thank my supervisor Doctor Frank Ball for his continuous guidance and support throughout the period of my research. I have been fortunate that they have given me a lot of their time during which I gained much from their great knowledge.

This work was funded and supported by a scholarship from University of Petra, my thanks to all academic and administrative staff for making it possible for me to embark upon this.

I would like to thank Dr Nuha AlKhalili for her advice in relation to computer graphics and Dr Ali Maqousi for his advice in relation to networking and simulations.

I would like to thank all members, past and present, of the School of Technology, Oxford Brookes University, In particular, academic and administrative staff John Nealon, Sue Greenwood, Sue Flint, Elizabeth Maynard, Peter Ells, Chris Flux, and all members of Multi-service Systems Group, Head of the group Professor David Duce, and my colleagues, Kashinath Basu, Musbah Sagar, Xianglin Li, Zaineb Ben Fredj, Ralph Targett.

Finally, I would like to dedicate the thesis to my family, my husband Ali, my son Younis and my daughter Katia: without their love, support, and patience, I would never have finished it.

Table of contents

Abstract

Acknowledgments

Table of Contents I

List of Figures V

List of Tables X

Glossary XII

1- Introduction 1

1.1 Thesis Aims and Objectives 6

1.2 Thesis Outline 6

2- DVE Applications and Scalability Requirements of DVE Systems 11

2.1 DVE Applications – an Overview 11

2.1.1 Distributed Virtual Environments. Perspectives and Challenges 11

2.1.2 Current Architectures 12

2.1.3 Summary 20

2.2 Key Scalability Aspects for Large-Scale DVEs 20

2.2.1 Scalability and Resource Management 20

2.2.2 Data Flow Restriction 22

2.2.3 Exploiting Level-of-Detail Perception 26

2.2.4 Architecture Design 28

2.2.4.1 Multicast Communications 28

2.3 Summary 32

3- Active Networks 34

3.1 What is an Active Network? 35

3.2 Challenges of Active Network Design 38

3.3 Active Network Architectures 38

3.3.1 Active Capsules: ANTS 39
3.3.2 SwitchWare 41
3.4 Active Networks for Supporting Multicast Communication 42
3.4.1 Active Reliable Multicast 43
3.4.2 Aggregated Hierarchical Multicast 45
3.4.3 Active Video Transcoding 45
3.4.4 Router-assisted Control for Layered Multicast 48
3.4.5 Active Interest Filtering for Distributed Simulation 49
3.5 Multi-service in Packed Switched Networks 53
3.6.1 Integrated Services 54
3.6.2 Differentiated Services 54
3.6.3 Hierarchical Approach 56
3.6 Summary 56
4- Multiresolution Representation and Progressive Transmission of 3D Data 59
4.1 Multiresolution Techniques 60
4.1.1 Discrete Multiresolution Modelling 61
4.1.2 Continuous Multiresolution Modelling 62
4.2 Progressive Transmission 65
4.3 Compressed Progressive Meshes 68
4.4 Summary 71
5- Methodology and Framework for Evaluation 73
5.1 Methodology 75
5.2 Framework for Evaluation 75
5.3 Summary 82
6- Active Networking Architecture Model 84
6.1 Architecture model 84
6.2 Active Node Architecture Model 85
6.2.1 Active Node Architecture Model 86
6.2.2 Packet Formats and Filtering Mechanism 89
6.3 Active Host Architecture Model 93
6.3.1 3D Data Transmission within the Post Initialization 93

Stage of DVE Application
6.3.2 Host Architecture Model 97
6.4 Summary 98
7- Flow and Application Models 99
7.1 Flow Model 99
7.1.1 Mesh Simplification Algorithm 101
7.1.2 Simplification Results Example 106
7.1.3 Progressive 3D Data used in the Evaluation Process 107
7.1.4 Summary 110
7.2 Application Model 111
7.2.1 Users' Movement Generation Algorithm 111
7.2.2 Application Models used in the Evaluation Process 114
7.2.3 Summary 117
7.3 Chapter Summary 118
8- Evaluation of the Active Approach within the Initialization Stage of the Application 119
8.1 Simulation Model 119
8.2 Experiment Description 120
8.3 Results from Simulation 121
8.3.1 Non-active Approach Results 121
8.3.2 Source Smoothing Scenario Results 124
8.3.3 Active Approach Results 126
8.3.3.1 Results for 3 LOD Scenario 126
8.3.3.2 Results for 9 LOD Scenario 132
8.4 Discussion of Results 135
9- Evaluation of the Active Approach within the Post Initialization Stage of the Application 136
9.1 Comparison between Active and Non-active Approaches' Performance for Basic Scenario Experiment 136
9.1.1 Simulation Model 137
9.1.2 Experiment Description 138

9.1.3 Results from Simulation 140
9.2 Evaluation of Active Approach for Different Application Models with Changes to the Experimental Parameters (Duration, Bandwidth, Number of Users) 147
9.2.1 Set 1 of Experiments 148
9.2.2 Set 2 of Experiments 162
9.2.3 Set 3 of Experiments 165
9.3 Discussion of Results 172
10- Conclusion 176
10.1 Research Contributions and General Findings 180
10.2 Further Work 180

References 183
Appendix I Mesh Simplification Results 191
Appendix II Movement Scripts 204
Appendix III OPNET Simulation Networks and Nodes 209
Appendix IV SVG Visualization 213

List of Figures

Figure 2-1 Client-server architecture 14

Figure 2-2 Peer-to-peer architecture 16

Figure 2-3 Server clusters architecture 18

Figure 2-4 Aura-nimbus model 23

Figure 2-5 Rectilinear and hexagonal grids 26

Figure 3-1 Active network structure 37

Figure 3-2 Capsule format 40

Figure 3-3 Active video scaling 47

Figure 3-4 Example of active filtering 52

Figure 3-5 Network architecture 56

Figure 4-1 PM simplification and reconstruction for triangle mesh 64

Figure 4-2 Error/time evolution 67

Figure 4-3 Progressive transmission solutions 68

Figure 5-1 Proposed active network architecture for DVE application 74

Figure 6-1 Active node architecture model 86

Figure 6-2 Active processing on a node 87

Figure 6-3 Geometrical data packet format 89

Figure 6-4 LOD table 90

Figure 6-5 Users' placements in two-dimensional grid 90

Figure 6-6 Control packet format 91

Figure 6-7 Filtering within one multicast group 92

Figure 6-8 User A moves in the virtual world 94

Figure 6-9 Active host architecture model 97

Figure 7-1 Invalid edge collapses 101

Figure 7-2 The edge collapse transformation 102

Figure 7-3 One step of mesh simplification algorithm 105

Figure 7-4 Simplification and restoration results 106

Figure 7-5 Virtual world partition 112

Figure 7-6 Algorithm to generate sequence of movements for one user 113

Figure 7-7a *Random* scenario 115
Figure 7-7b *Far* scenario 116
Figure 7-7c *Centre* scenario 117
Figure 7-7d *Join* scenario 117
Figure 8-1 Network configuration that is used in experiments 121
Figure 8-2 Queue occupancy for routers R1 and R3 within the non-active scenario 124
Figure 8-3 Queue occupancy for router R1 with increased mean of inter-generation time 125
Figure 8-4 Queue occupancy for router R3 with increased mean of inter-generation time 125
Figure 8-5 Initial placement of users in the grid 3x3 for the initialization stage experiments 127
Figure 8-6 Queue occupancy for non-active and active 3 LOD experiments for router R3 129
Figure 8-7 User B moves in the virtual world 129
Figure 8-8 Comparison of queue's length for static and dynamic scenarios for 3 LOD case 130
Figure 8-9 Changes in LOD lookup table on routers 131
Figure 8-10 Initial placements of users in the 9x9 grid those are consistent to the initial spatial relationship of users in 3 LOD experiments 132
Figure 8-11 PM data packets generated by users within 9 LODs experiment scenario 134
Figure 8-12 Queue occupancy for 3 LOD and 9 LOD experiments (router R3) 134
Figure 9-1-1 Initial placements of users for basic scenario experiments 139
Figure 9-1-2 Script example for user C 140
Figure 9-1-3 Queue occupancy results for router R1 (random scenario, 20 minutes) 141
Figure 9-1-4 Queue occupancy results for router R3 (random scenario, 142

20 minutes)
Figure 9-1-5 Queue occupancy results for router R4 (random scenario, 20 minutes) 143
Figure 9-1-6 Example of PM data sent and received within active 3 LOD scenario 146
Figure 9-1-7 Example of PM data sent and received within active 9 LOD scenario 146
Figure 9-2-1 Movements of users for all application models (left – 20 minutes and right – one hour) 150
Figure 9-2-2 Queue occupancy results for 20 minutes and one hour (*random* scenario) 152
Figure 9-2-3 Queue occupancy results for 20 minutes and one hour (*far* scenario) 153
Figure 9-2-4 Queue occupancy results for 20 minutes and one hour (*centre* scenario) 154
Figure 9-2-5 Queue occupancy results for 20 minutes and one hour (*join* scenario) 155
Figure 9-2-6 Memory requirements for 3D data on host B (20 minutes) 159
Figure 9-2-7 Memory requirements for 3D data on host B (one hour) 160
Figure 9-2-8 Maximum queue length results for increasing bandwidth 164
Figure 9-2-9 Network configuration that was used in experiments with different number of users 165
Figure 9-2-10 Initial placements of users that was used in experiments with different number of users 166
Figure 9-2-11 Maximum queue length for *random* scenario (from 4 to 8 users, 20minutes and one hour) 167
Figure 9-2-12 Maximum queue length for *far* scenario (from 4 to 8 users, 20minutes and one hour) 168
Figure 9-2-13 Maximum queue length for *centre* scenario (from 4 to 8 users, 20minutes and one hour) 169
Figure 9-2-14 Maximum queue length for *join* scenario 170

(from 4 to 8 users, 20minutes and one hour)
Figure 9-2-15 Average initial reception time on host for increasing number of users (user A) 172
Figure I-1 Crude model (LOD=0, 96 vertices) of the *baseball bat* object progressive representation 196
Figure I-2 Refinement batch 1 (LOD=1) of the *baseball bat* object progressive representation 199
Figure I-3 Refinement batch 2 (LOD=2) of the *baseball bat* object progressive representation 203
Figure II-1 Fragments of the movement scripts. User C, *random* scenario, 4 users group (a) 3 LOD, b) 9 LOD) 205
Figure II-2 Fragment of the movement script (User D, *far* scenario, 4 users group, 9LOD) 205
Figure II-3 Fragment of the movement script (User A, *centre* scenario, 4 users group, 9LOD) 206
Figure II-4 Fragment of the movement script (User B, *join* scenario, 4 users group, 9LOD) 207
Figure II-5 Fragment of the movement script (User C, *centre* scenario, 6 users group, 9LOD) 207
Figure II-6 Fragment of the movement script (User D, *join* scenario, 8 users group, 9LOD) 208
Figure III-1 Network model (non-active, 4 users) 209
Figure III-2 Router model (non-active) 209
Figure III-3 Host model (non-active) 210
Figure III-4 Network model (active) 210
Figure III-5 Router model (active) 211
Figure III-6 Host model (3 LOD active) 211
Figure III-7 Host model (9 LOD active) 212
Figure III-8 Network model (active, 8 users) 212
Figure IV-1 SVG visualization (7 users, *random*, 20 minutes) 213

Figure IV-2 SVG visualization (8 users, *random*, 20 minutes) 213
Figure IV-3 SVG visualization (6 users, *centre*, 20 minutes) 213
Figure IV-4 SVG visualization (6 users, *far*, 20 minutes) 214
Figure IV-5 SVG visualization (8 users, *join*, 20 minutes) 214
Figure IV-6 SVG visualization (8 users, *centre*, 20 minutes) 214

List of Tables

Table 6-1	Movement of user A	95
Table 6-2	Transmitted LODs	95
Table 7-1	3 LOD flow model	109
Table 7-2	9 LOD flow model	110
Table 8-3-1	Initial reception time (non-active, four senders)	122
Table 8-3-2	Initial reception time (non-active, one sender)	122
Table 8-3-3	Numbers of processed packets (non-active initial transmission, router R1)	123
Table 8-3-4	Reception time for different inter-generation means	126
Table 8-3-5	Reception time for users in the group (initial stage, active 3 LOD, static)	127
Table 8-3-6	Reception time for users in the group (initial stage, active 3 LOD, dynamic)	130
Table 8-3-7	Reception time for users in the group (initial stage, active 9 LOD, static)	133
Table 9-1-1	Maximum queue length (basic scenario)	144
Table 9-1-1	Utilization of the queue (basic scenario)	144
Table 9-1-3	Example of numbers of transmitted packets for non-active and 3 LOD active approach	145
Table 9-1-4	Example of numbers of transmitted packets for the 3 LOD and 9 LOD active approaches	147
Table 9-2-1	Maximum queue length (Set 1 of experiments)	149
Table 9-2-2	Utilization of the queue (Set 1 of experiments)	156
Table 9-2-3	Memory requirements on host (20 minutes)	161
Table 9-2-4	Memory requirements on host (one hour)	162
Table 9-2-5	Maximum queue length (Set 2 of experiments, *random* scenario)	163
Table 9-2-6	Maximum queue length (Set 2 of experiments, *far* scenario)	163
Table 9-2-7	Maximum queue length (Set 2 of experiments, *centre* scenario)	163
Table 9-2-8	Maximum queue length (Set 2 of experiments, *join* scenario)	163
Table 9-2-9a	Maximum queue length (Set 3, *random*, 20 minutes)	167

Table 9-2-9b Maximum queue length (Set 3, *random*, one hour) 167
Table 9-2-10a Maximum queue length (Set 3, *far*, 20 minutes) 168
Table 9-2-10b Maximum queue length (Set 3, *far*, one hour) 168
Table 9-2-11a Maximum queue length (Set 3, *centre*, 20 minutes) 169
Table 9-2-11b Maximum queue length (Set 3, *centre*, one hour) 169
Table 9-2-12a Maximum queue length (Set 3, *join*, 20 minutes) 170
Table 9-2-12b Maximum queue length (Set 3, *join*, one hour) 170
Table 9-2-13 Average initial reception time on host (Set 3 of experiments) 171

Glossary

ABone	Active network Backbone
AFE	Active Forwarding Engine
AN	Active Network
ANEP	Active Networking Encapsulation Protocol
ANTS	Active Network Transfer System
ARM	Active Reliable Multicast
CPM	Compressed Progressive Meshes
DAN	Distributed code caching for Active Networks
DARPA	Defence Advanced Research Projects Agency
DIS	Distributed Interactive Simulation
DSL	Digital Subscriber Line
DVE	Distributed Virtual Environment
DVMRP	Distance Vector Multicast Routing Protocol
EE	Execution Environment
EF	Expedited Forwarding
FE	Forwarding Engine
IGMP	Internet Group Management Protocol
IP	Internet Protocol
LAN	Local Area Network
LOD	Level of Detail
MBone	Multicast Backbone
MMPORG	Massively Multiplayer Online Role-Playing Games
NACK	Negative Acknowledgment
NLM	Network-based Layered Multicast
PFS	Progressive Forest Split
PHB	Per Hob Behaviour
PIM	Protocol Independent Multicast
PLAN	Programming Language for Active Networks
PM	Progressive Meshes
QoS	Quality of Service
RSVP	Resource ReSerVation Protocol
SVG	Scalable Vector Graphics
SLA	Service Level Agreement
SRM	Scalable Reliable Multicast
TCP	Transport Control Protocol
TOS	Type of Service
TTL	Time-to-Live
VE	Virtual Environment
VRML	Virtual Reality Modelling Language
VRTP	Virtual Reality Transfer Protocol
VW	Virtual World

Chapter 1

Introduction

Distributed Virtual Environments (DVEs) provide 3D graphical computer generated environments with stereo sound, supporting real-time collaboration between potentially large numbers of users distributed around the world. A DVE system consists of four basic elements: graphics engines and displays, communication and control devices, processing systems, and a data network [Singhal, 99]. These elements work together to ensure the sense of immersion among users at different sites. Network bandwidth is a cornerstone of a DVE system and its availability defines the quality and the scale of the DVE application. However, network capacity is a limited resource. Originally, DVE systems could only be used at universities or large military or industrial institutions for an academic research and battle ground simulations using fast local area networks (LAN). The scale of these early DVEs was less than a dozen users due to the limited capacity of earlier LANs and the lack of network resource management.

Over the last decade the world of networking has changed radically. LAN capacities have increased dramatically with deploying 1 Gbits/s Ethernet. Research has resulted in modern DVEs that manage the available bandwidth more carefully and can support several thousand simultaneous participants using LAN communications [Singhal, 99]. Furthermore, the Internet develops into the most common embedding for DVEs, as standard Web browsers become an execution engine for DVE applications. Recently large-scale DVEs are attracting increasing attention and the todays DVE systems evolve to the distributed systems that employ wide area data networks.

The recent widespread household adoption of broadband Internet access services such as digital subscriber line (DSL) and cable modem services, many service providers are searching for attractive content that can take advantage of these high-speed, always-on connections. Massively multiplayer online role-playing games (MMORPGs) that can be regarded as a derivative of DVEs, are emerging as popular uses of such Internet connections[Kawahara, 2004]. Consequently, these applications

will be available for significantly increased numbers of users and it might be expected that persistent, large-scale, distributed virtual environments inhabited by millions of entities will eventually emerge.

To achieve this vision for DVE applications the networking scalability challenges have to be considered. DVE applications demand exchange of the data that forms different types of traffic such as a control data type, video and audio, and a 3D data type to keep the consistency of the application's state.

The computer data type usually requires reliable delivery of data but do not impose stringent requirements for the timeliness of delivery (e.g. not real time E-mail application). However, within the DVE application that is an interactive application the data traffic comprises the state update control messages, the signalling and management, and the text messages and has to be delivered reliably but within real time. In this case methods such as Scalable Reliable Multicast (SRM) [Floyd, 97] could be applied.

The interactive video and audio traffic is also produced by the DVE application and is sent continuously at some general rate. For these traditional continuous data types within the DVE application transcoding could be applied to minimize the size of transmitted data and therefore decrease the time to receive the information.

The 3D data traffic includes the 3D graphical representations of the users and the virtual world. In comparison, for example, with light-weight state update messages this kind of traffic is bulky and has to be delivered reliably and within real time.

The problem is that meeting the QoS requirements of both control and continuous media traffic has already been covered by existing research [Floyd, 97]. But QoS for transfer of the 3D information has not really been considered.

The 3D objects are described by the geometrical data or triangulated meshes that approximate the object's surface. The size of these meshes, which now can be measured by millions, or even hundreds of millions of triangles, imposes limits on applications for which complex 3D models must be accessed remotely. However, only very simple models can be accessed over today's common communication links

for immediate viewing. Generally, the amount of geometrical information required to represent an object in full detail can be quite large, and if updates within the DVE are to be sufficiently responsive then the transmission of geometrical information should be carried out as quickly as possible. Also given that the transfer of geometrical information will not generally be a continuous process this implies that the resulting traffic will be very bursty in nature and will place high demands on the network for short intervals of time. Whilst such bursts could be smoothed out, this will result in longer transfer times, which in turn will reduce responsiveness. Furthermore, this type of traffic could be sensitive to packet loss, as the need for retransmission would also degrade responsiveness.

The transcoding paradigm that is applied for the video and image to provide variations of the same object using different modalities can be naturally extended to 3D data multimedia type. To achieve transcoding of 3D data a multiresolution representation of the object with model simplification and level of detail (LOD) management could be used. The human eye does not always distinguish the fine details of a mesh, especially when the mesh is displayed far back in the view frustum. Therefore a set of object approximations referred to as Level-of-Detail (LOD) models, where each one represents the original object with a different level of detail could be deployed to describe a 3D geometry. The closer the viewer gets to the object, the more details are rendered. Multiple representations with decreasing resolution (LODs) are used to reduce rendering cost for distant or otherwise less important objects. As only one level of detail of a given object can be displayed at any time instead of transmission of geometry data for a scene as a whole, it is possible to transmit exactly packets with certain LOD to the receivers who need to view the scene with this certain level of fidelity. Therefore the filtering of unnecessary 3D geometry flows that is based on the spatial proximity relation or LOD concept could be carried out for the DVEs' 3D data media type. This approach seems to improve the 3D geometry traffic profile by reducing the burst sizes and might be a solution to the scalability of DVEs.

The growth of numbers of applications that require complex network services (e.g. multimedia applications) recently led to the idea about network-embedded

functionality which has been classified as active networking [Schmid, 2002]. This is a new networking paradigm for packet-switched data networks whereby the focus changes from the pure packet forwarding to the dynamic programming of network nodes. Active network techniques enhance the multi-service Internet approaches (Differentiated and Integrated Services approaches) by adding more flexible, dynamically installed, and programmable services. The performance of intermediate active nodes inside the network is customized by users' traffic processing (e.g. flow control, payload processing, caching and others) which supports the needs of applications. An elegant realization of this idea is seen in the work of Keller *et al* [Keller, 2000] in their Active Router architecture for multicast video distribution. The authors employ an active wavelet-based video encoding scheme to provide video scaling. This Active Router approach allows the decision on whether or not to forward a video packet to be made in the router. To a certain extent we can see an analogy between the active video scaling and the spatial proximity interest filtering of 3D content. In the case of the video scaling, the quality of received video will be defined by the receiver's ability to obtain a video stream (e.g. through congested line). In case of the DVE's interest filtering the distance between objects in the virtual world will identify the fidelity of the object [Balikhina, 2002a].

In this thesis we explore how active networking ideas can be applied to the distribution of 3D data in a DVE. We propose the novel active networking architecture that exploits the LOD concept and aims to improve the currently applied decentralized peer-to-peer DVE architecture. The peer-to-peer multicast is more scalable than the current alternative centralized client-server architecture where a single server could be a bottleneck and a single point of failure. Within the peer-to-peer architecture participant distributes state update information to all other DVE participants. In this case some users would receive irrelevant information (e.g. remote users or users that are less interested in this data). To save bandwidth and processing power on hosts that would be used to transmit and process this irrelevant information, participants of DVE are usually grouped into multicast groups. The common grouping technique is a spatial partitioning when groups are created using the geographical areas of the virtual world. Therefore the filtering of unnecessary data

flows is implemented on the network layer by the multicast routing. However, with increasing the application's scale the size of the multicast group has to be considered. The large groups might limit the scale of the DVE application by a huge bandwidth utilization that could be needed to transmit data to all members of the one large multicast group. Otherwise, the large number of small sized multicast groups might cause a multicast address space limitation and frequent time-consuming joining and leaving processes.

The active networking approach to peer-to-peer multicast proposed in this work adds the application layer LOD filtering to the network layer multicast filtering and improves data flow restriction within the large size multicast group. The active processing on the routers supports more precise restriction of unnecessary flows of 3D data which is controlled by the application information. A spatial partitioning method is applied to divide the virtual world into subspaces [Macedonia, 95]. The spatial positions of users within the grid of the virtual world define the required quality of the representation of objects (if user is placed closer to the viewed object more detailed representation is needed to be rendered). This application layer information is used by active routers to prune more detailed 3D data flows in the multicast tree arches that are linked to the distance DVE participants. To accomplish this task the 3D data in the proposed approach is structured as a multiresolution representation of the object with decreasing fidelity approximations or levels of detail (LOD). However, the continuous multiresolution progressive representation that incorporates the LOD concept [Hoppe, 96] [Pajarola, 2000] is used. This representation allows transmission of 3D data progressively to minimize the waiting time to view the object and avoids duplications that exist within the transmission of the several discrete LOD representations employed by the current DVEs (e.g. DIVE system [Frecon, 98]).

This thesis describes the development process of a novel active networking architecture for large-scale DVEs. Firstly, the DVE applications, active networking and multiresolution representation of 3D models are considered. Secondly, the methodology and the evaluation framework are defined. The performance modelling approach is applied in this work. Therefore the architecture model is developed.

Then, the flow and application models that are used by the evaluation process are developed. Finally, the proposed active approach is modelled through the simulation and simulation experiments are used to evaluate the performance of the active approach in comparison with the conventional non-active approach.

1.1 Thesis Aims and Objectives

The general aim of this work is to explore the possible benefits of exploiting active networks and the LOD concept for the transmission of the 3D DVE data traffic and to investigate how this active approach could help to improve the scalability issues for large-scale DVE applications. A comparison is made with the non-active approach.

To achieve this it is necessary to carry out the following:

1) To develop a novel active networking architectural model that could improve the 3D DVE traffic profile by restriction unnecessary flows of the 3D data within one multicast group.
2) To develop the 3D DVE data flow and application models which complement the architecture model. These models are essential parts of the evaluation framework.
3) To evaluate the proposed active networking architecture performance through simulation experiments by comparison with the conventional non-active transmission of the 3D DVE data.

1.2 Thesis Outline

The thesis is structured into ten chapters including this introduction.

In chapter 2 distributed virtual environments (DVEs) and scalability requirements of DVE systems that are important for the design of the large-scale DVE applications are introduced. The current architectures that are used for the DVE applications are illustrated by three major approaches client-server, peer-to-peer, and hybrid architectures. The key scalability aspects for the large-scale DVE applications

are discussed. The resource management to improve the scalability in the DVE is studied. Three resource management techniques to achieve the scalability of the DVE applications that are used in this thesis are discussed. Data flow restrictions that exploit a spatial partitioning of virtual worlds, limited user perception or the level of detail (LOD) concept, and system's architecture modifications all provide the saving of such major resources as the network bandwidth and the processing on hosts and therefore increase the scalability of DVE systems. Also this chapter outlines the novel architecture for large-scale DVEs that exploits active networking to improve the peer-to-peer architecture by applying a data flow restriction which is based on spatial partitioning and the LOD resource management techniques combined with the progressive transmission of 3D streams.

Chapter 3 discusses active networking. The definition, active networking challenges and the current active network architectures are presented. The discussion is focused on the recent research for active networking approaches to improve the performance of multicast applications. The survey of the active networking techniques to improve the applications based on the multicast transmission is provided. As the main examples an active reliable multicast [Lehman, 98], active video transcoding [Keller, 2000], router-assisted traffic control for layered multicast [Son, 2003], and an interest filtering for the distributed simulation [Zabele, 2000] are used. In addition to discussion of active networks chapter 3 also discusses the general solutions which add multi-service capability to packet-switched networks. The Integrated and Differentiated services, as well as the hierarchical approach that combines both of these services, are introduced. The active networking aims to add to these general approaches more flexible, dynamically installed, and programmable services.

Chapter 4 introduces the multiresolution representation and progressive transmission of 3D data. The LOD concept that is used in this work to restrict unnecessary 3D flows within the multicast group incorporates the progressive transmission of the multiple approximations of the 3D object with increasing fidelity.

The conventional 3D representation is outlined together with the storage and transmission problems caused by the recent enormous increase in the size of the 3D objects. As the 3D objects become more realistic they need huge amounts of data to describe them. In this context multiresolution techniques are shown to be a promising solution to the 3D storage and transmission problems. The discrete and the continuous multiresolution models are defined. The continuous models are classified as more desirable as they allow progressive transmission of the 3D data. Then one of the multiresolution continuous models; Hoppe's progressive meshes (PM) are described. Chapter 4 includes the description of the Compressed Progressive Meshes (CPM) [Pajarola, 2000] model that uses grouping of the refinement operations and improves the fine-grained PM approach. The CPM representation which integrates LOD concept is used in this work as a basis for the 3D content flow model.

Chapter 5 describes the methodology applied in this work and the framework for the evaluation of the proposed active network architecture for large-scale DVE applications. The main features of the active networking architecture proposed in this work are outlined. The methodology is presented as a performance modelling approach and simulation based evaluation. The evaluation framework includes the architecture modelling, flow and application modelling, defining performance metrics, design of simulation experiments and simulation-based evaluation of effectiveness of proposed approach. All parts of the evaluation framework are described in chapter 5.

Chapter 6 discusses in detail the model active networking architecture that is proposed in this work. This architecture uses active networking techniques to improve the multicast of the 3D DVE data by restriction of unnecessary flows of 3D packets and is an example of the discrete active approach or programmable switch. Three layers of the programmable switch approach that are used in our architecture are presented which are the active node, active extension and active packets. Two components of the active architecture are introduced – the active router and active host. The active node architecture is presented in context with router architecture.

The active processing in a router is shown in the form of an execution environment or active extension. The active packet formats and filtering mechanism are explained. The DVE host architecture model is also provided.

In chapter 7 the description of the framework for the evaluation is continued. The flow and application models parts of the evaluation framework are introduced here. The flow model in this work represents the progressive 3D geometrical data which is transmitted by DVE users. The simplification algorithm that has been developed to convert the conventional single-resolution 3D meshes into progressive meshes which incorporate LOD aspect is described in this chapter. The results of the simplification and restoration for a typical small mesh are illustrated. Progressive representations of a typical large 3D model with different numbers of LODs that are used as the flow models in the evaluation process are included. The important component of the evaluation process is an application model. The application model is defined by the interactions of DVE's participants and consists of movement pattern descriptions for every participant. These scripts drive the exchange of the 3D data streams between the users within the post initialization stage of the application. Chapter 7 presents the algorithm that has been developed to generate the users' movement scripts. The application models obtained reflect four different user patterns of movement. These four different scenarios are used in the evaluation process.

Chapter 8 presents the simulation modelling of the proposed architecture and demonstrates the comparative performance of the non-active and active approaches within the initialization stage of the DVE session. This first part of the evaluation experiment shows the comparison between the active and non-active routers' and hosts' performance within the initial transmission of the 3D representations of users and demonstrates that the simulation models work accurately and mirror the active architecture model. Alternative to the proposed active approach for smoothing the data source approach experiments are also presented in this chapter.

Chapter 9 includes the second part of the evaluation experiments and presents the evaluation of the complete active architecture model that models the entire runtime or the persistent stage of a DVE application. The simulation of the DVE host component is added to the simulation model to support the post initialization stage of the application. The comparative performance of the active and non-active approaches is shown for different application models. These application models are tested by changing the duration of the experiment, the available bandwidth for the 3D geometry traffic, and the number of users.

Chapter 10 concludes with a summary of the work and identifies the major contributions that have been made. It also provides a discussion of possible contributions that the proposed active networking approach for the large-scale DVEs could make in the future work.

Chapter 2

DVE Applications and Scalability Requirements of DVE Systems

2.1 DVE Applications – an Overview

2.1.1 Distributed Virtual Environments. Perspectives and Challenges

A virtual environment (VE) is a real-time computer-generated simulation of a real or imaginary world where users navigate and interact with 3D objects within it. VEs can be multi-user, supporting multiple interacting users, and distributed, running on several computers connected by a network. It is common to refer to a VE with both these additional properties as a DVE (Distributed Virtual Environment).

DVEs provide 3D graphical computer generated environments with stereo sound, supporting real-time collaboration between potentially large numbers of users distributed around the world. They seek to provide a sense of presence and engagement akin to that experienced in real-world collaboration [Schroeder, 2001]. DVEs are being used for education and training, for engineering and design, for commerce, and for entertainment.

Recently large-scale DVEs are attracting increasing attention, not least from the simulation community (for example battleground simulation). There are also strong similarities between DVEs and distributed multiuser games.

It was at first only possible to use such environments where local area networks were available, though the push to higher performance public data networks is now changing that picture. For a long time DVEs could only be used for academic research or large military or industrial institutions with fast local area networks. But over the past few years the world of networking has changed dramatically (e.g. emergence of 1 Gb/s Ethernet and recent DVEs that manage available network resources more carefully). This has enabled DVEs to escape from the realm of private networks into use of public data networks. Also with standard Web browsers as an execution engine for DVE, the Internet will increasingly become the most common embedding for DVEs. Eventually there

will be persistent, large-scale, distributed virtual environments inhabited by millions of entities.

Furthermore, with the recent widespread household adoption of broadband Internet access services such as digital subscriber line (DSL) and cable modem services, many service providers are searching for attractive content that can take advantage of these high-speed, always-on connections. Massively multiplayer online role-playing games (MMORPGs) that can be regarded as a derivative of DVEs, are emerging as popular uses of such Internet connections. In these games users can take part in a persistent online gaming world at any time and individual titles can attract hundreds of thousands of registered users for online gaming. There is an expectation that the MMORPG market will grow significantly in the future as more households become permanently connected to the Internet [Kawahara, 2004].

Which qualities of online games or other DVE applications distinguish them from existing applications and attract the users? The presence of multiple independent users differentiates DVEs from standard virtual reality or gaming systems. The ability to share objects differentiates them from traditional chat rooms, and the real-time interactivity distinguishes DVEs from traditional Web browsing or electronic mail. DVEs are most appropriate for applications that demand the creation of telepresence, the illusion that other users are visible from remote locations.

These environments aim to provide users with a sense of realism and create an immersive experience. This experience has the following features: shared sense of space, shared sense of presence, shared sense of time, way to communicate, and way to share. In a DVE users demand a sense of realism approaching that which otherwise can only be achieved by face-to-face contact. This is very important for the future of this technology. DVEs will not be used, for example, if they distort "normal" social relationships, independently of the technology issues [Tromp, 98].

To provide this sense of realism or immersion among users at different sites all components of a DVE system work together.

A DVE system has four basic components [Singhal, 99].

- Graphics engines and displays. The display provides the user with a three-dimensional window into the virtual world, and the engine generates the images for display.
- Communication and control devices help users to move about, pick up and manipulate objects, and communicate with other participants in the virtual environment.
- Processing systems. Decreasing processor prices have driven much of the growth behind DVE deployment, as DVEs demand a considerable amount of processing capacity.
- A data network. DVEs rely on the data network to exchange information about the current state of the virtual environment.

Availability of add-on high-speed graphics processors, decrease in processor prices, and emergence of more precise control devices are encouraging the widespread deployment of DVE applications nowadays.

Despite the recent advances in communication technology network capacity remains a limited resource especially for large-scale environments and this fact creates the following challenges for DVE designers:

- How to provide network support for the mixed traffic types demanded by DVEs, typically voice, video, 3D content, and data, with their very different characteristics with regard to network properties such as delay, jitter and loss rate?
- How to manage group participation in a DVE? Not all data are relevant to all participants in a DVE.
- How to manage persistent state within a DVE and manage state updates?
- How to keep these applications scalable?

As the network latency can distort the sense of immersion among the DVE users, the networking challenges in the design of DVE systems motivates researchers to explore and develop new networking solutions for large-scale DVEs. And one of the main problems is the question of what architecture to adopt in order to support such environments so that the same architecture will support growth to large numbers of users.

2.1.2 Current Architectures

Currently a wide range of communication architectures is used in DVEs. However, there are two extremes of a spectrum of possible architectures – client-server (centralized) and peer-to-peer (distributed or decentralized) approaches [Diehl, 2001]. Because the main goal of a DVE is to provide users with the illusion that they are all seeing the same things and interacting with each other in the virtual space, we have to consider different architectural approaches with respect to managing this consistent view or dynamic shared state.

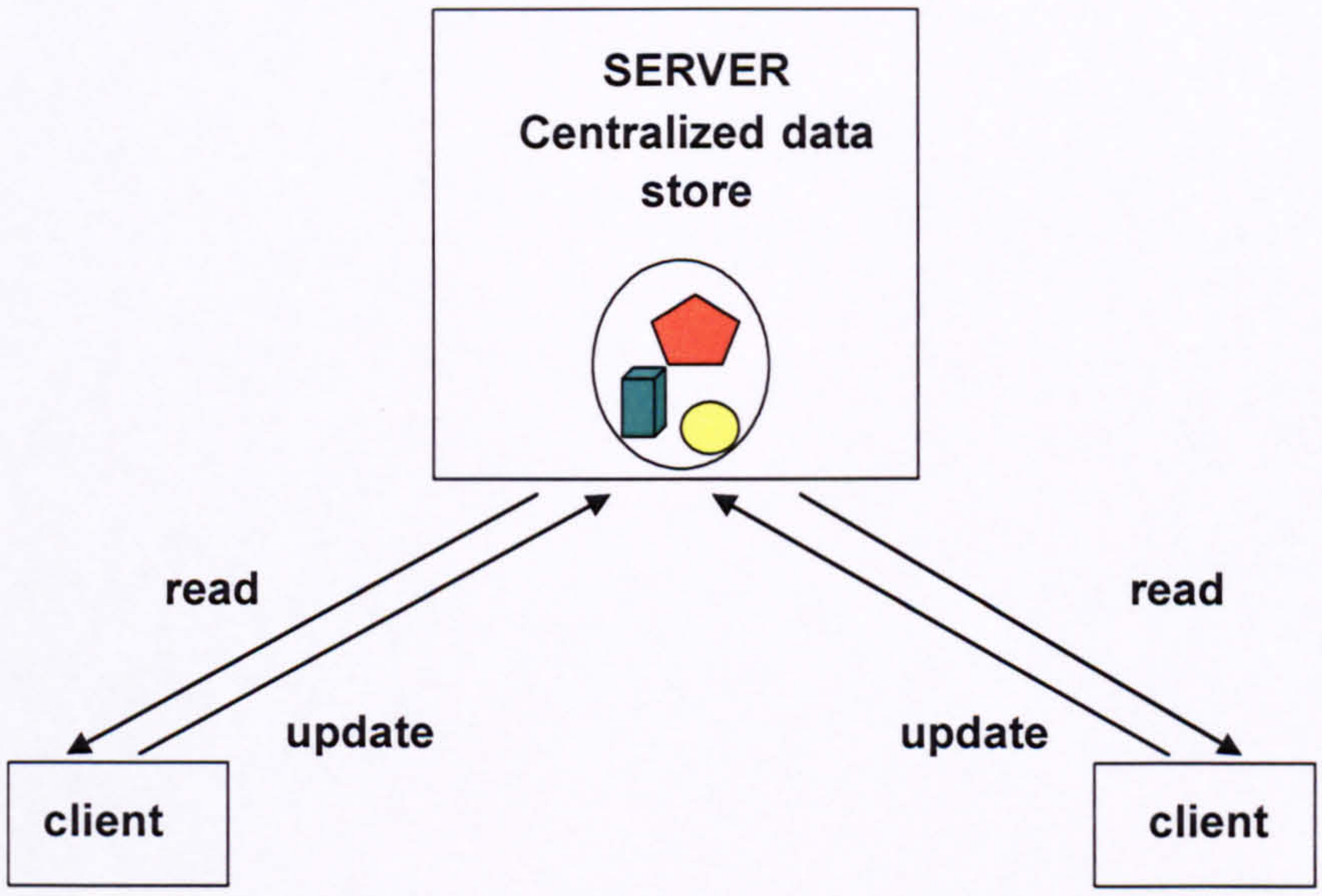


Figure 2-1 Client-server architecture

Presently most DVEs use the client-server architecture (Figure 2-1). The server manages the current state of the virtual environment. A client sends an update to the server which propagates it to all other clients. As all communication is via the server, the server becomes a bottleneck. The other problem is the possible faillure of the system when the server collapses (single-point-of-failure). An additional problem occurs when the client caches the data. In this case complex cache-consistency protocols have to be used too. However these systems are more reliable, because of using TCP/IP connections and are relatively easy to implement. These factors have led to the popularity of this architecture for supporting small to medium-sized DVEs. As an example we can take Internet game servers like Kali and Mplayer systems [Kali, 2003] [Mplayer, 2004].

Some of the client-server DVE systems are presented below.

BrickNet [Singh, 94] is primarily aimed at collaborative design environments, where a complete design task is distributed over multiple client workstations. The client-server approach is deployed by this system, as each node in a BrickNet collaboration is responsible for its part of design and does not require to have a complete copy of the virtual environment database.

NetEffect [Das, 97] uses a client-server architecture. In each world, there is a master server that is connected to many peer servers through point-to-point connections. The virtual world is broken into many "communities". Clients connect to one peer server at a time, but can migrate to another server at any time.

MAVERIK [Hubbold, 96] tries to support a persistent virtual environment that does not ever need to be rebooted. In an attempt to make the system scalable, each server task can be distributed among many processors.

The client-server systems try to increase the efficiency by applying multiple servers or deploying a single server that uses multiple processors to manage the state of the DVE application.

In the peer-to-peer architecture every client has a partial copy of the state. In other words, the DVE state is kept not in a single location (single disk or single application process), but distributed among all clients. If something changes, a client has to send this change to all other clients. If a client crashes or simply leaves the environment, all its data are lost, but remaining clients can continue without this client. Peer-to-peer communication as illustrated in Figure 2-2 allows us to design a scalable large DVE within the available limits of computing resources.

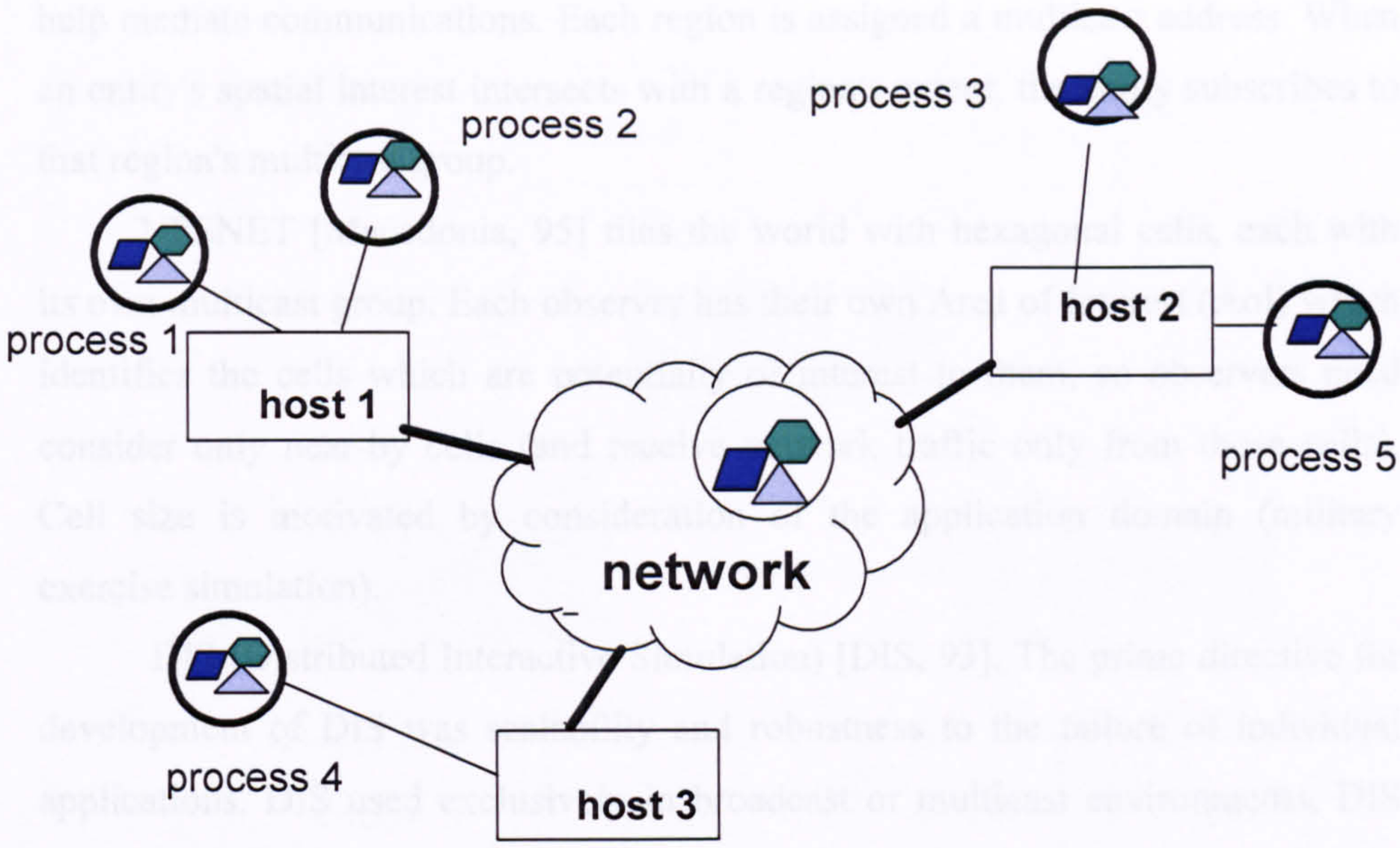


Figure 2-2 Peer-to-peer architecture

Use of multicast [Tanenbaum, 2003] is a principle idea in this case to minimize the number of message duplications between the sender and the receivers. Multicast enables a sender to send data to all hosts in a multicast group, avoiding unnecessary duplication of transmission by sending only one copy along each arc of the routing tree between sender and receivers.

A number of existing systems that use the peer-to-peer architecture is outlined below.

DIVE (Distributed Interactive Virtual Environment) system [Frecon, 98], the distribution model in DIVE is based on peer-to-peer distribution, multicast protocols, and coarse-grained partitioning of data. Peers connected to the same multicast group interact by making concurrent access to replicated data (entities) and by sending messages. DIVE can be thought of as a distributed database. The DVEs' concept of an aura of an object which is the 3D volume in which the object can interact with other objects is used in this system. The entities' "aura managers" are used to signal to the peers which parts of the database need to be replicated. Each replicated part is associated with a different multicast group.

MASSIVE-2 [Greenhalgh, 98], like DIVE, uses the concept of auras to represent the extent to which interaction with other entities is possible. However, MASSIVE-2 uses arbitrarily shaped, mobile regions called "third-party objects" to

help mediate communications. Each region is assigned a multicast address. When an entity's spatial interest intersects with a region's extent, the entity subscribes to that region's multicast group.

NPSNET [Macedonia, 95] tiles the world with hexagonal cells, each with its own multicast group. Each observer has their own Area of Interest (AoI) which identifies the cells which are potentially of interest to them, so observers need consider only near-by cells (and receive network traffic only from those cells). Cell size is motivated by consideration of the application domain (military exercise simulation).

DIS (Distributed Interactive Simulation) [DIS, 93]. The prime directive for development of DIS was scalability and robustness to the failure of individual applications. DIS used exclusively in broadcast or multicast environments. DIS manages the shared state of all entities in a distribution fashion, using an entity-ghost paradigm with dead-reckoning. However, the DIS protocol is limited by its origin (battlefield simulation). Also DIS implies that state information is small and can be regularly broadcast to all participants.

Alongside with two basic structures, client-server and peer-to-peer architecture there are also a large number of hybrid structures.

First, the basic client-server architecture may include a federation, or cluster, of servers that communicate in a peer-to-peer manner. As an example of this approach we can take the RING system [Funkhouser, 95] where clients connect to one of many servers. As the number of entities in the virtual world increases, more servers can be added. Servers are connected to each other via high-speed networks, while clients can use lower speed connections to communicate to their server. Having multiple servers reduces workload on each server because the clients are divided among them. However the introduction of multiple servers does incur some cost. So, this approach is more suitable where the servers are connected by high-speed LANs. Alternatively, these servers may themselves behave as clients in a hierarchical client-server relationship.

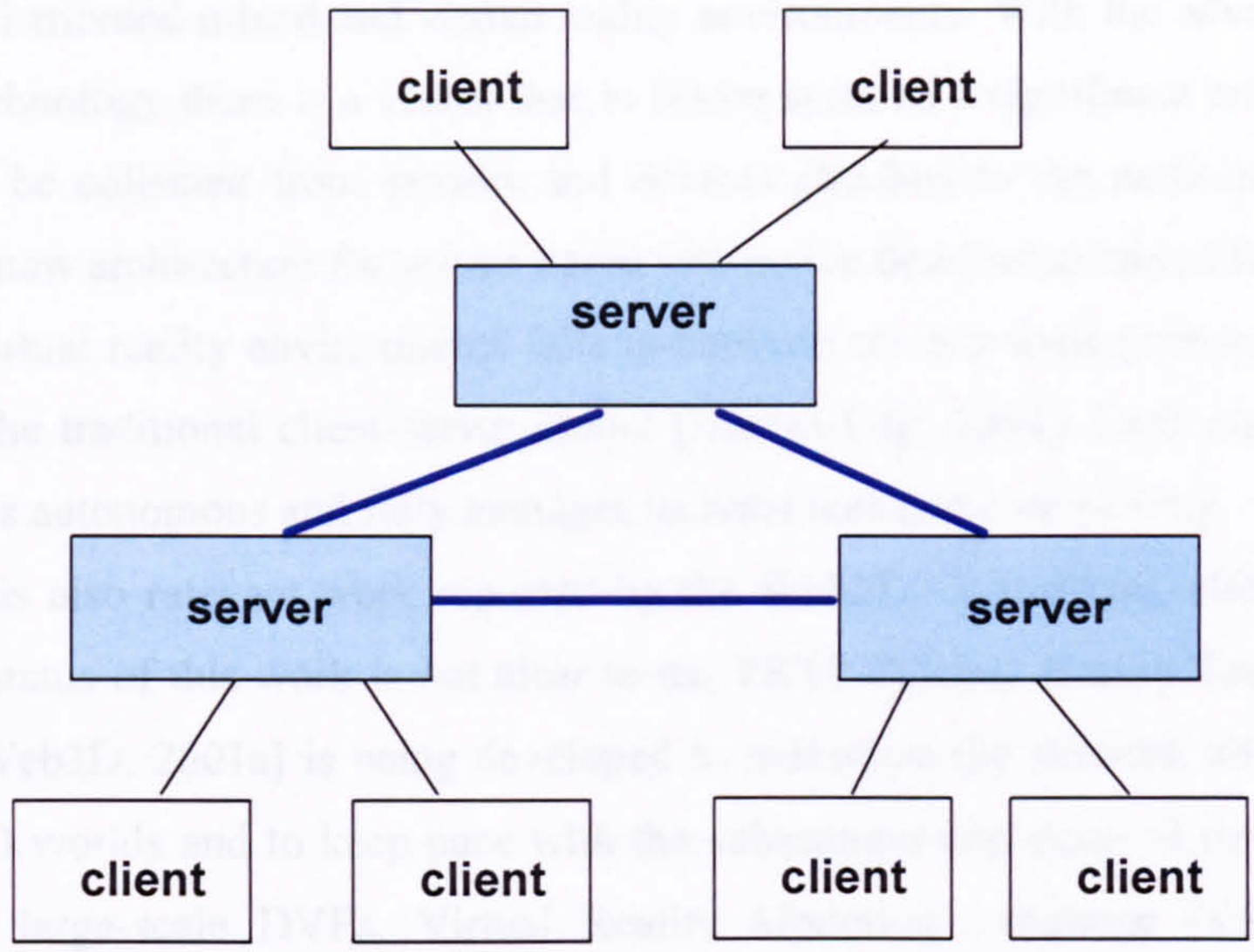


Figure 2- 3 Server clusters architecture

Second, the client-server and peer-to-peer structures may be merged into a so-called peer-server architecture, in which packets travel to some destinations in a peer-to-peer manner and to other destinations via a server.

We can observe this in the Distributed Worlds Transfer and communication Protocol (DWTP) [Broll, 98]. There are two roles: participants and demons. A demon can be simultaneously responsible for several worlds. Scalability is achieved by the use of multicasting in combination with demons. There are demons for different tasks and demons can run on different computers.

ATLAS [Lee, 2002] also uses the peer-server model as its primary communication architecture. A server joins several multicast addresses assigned to regions and maintains the membership of users and the states of a virtual world. Each participant multicasts its update messages directly to other participants in its region and its neighbouring regions as well.

VELVET [Oliveira, 2003] is an adaptive hybrid architecture that allows a greater number of users to interact through a DVE. This is accomplished through an adaptive host filtering scheme based on multicasting that supports heterogeneity amongst the various participants. VELVET allows real-time adaptation, according to the local client needs and network resources availability. Such a feature allows systems with different processing power and networking

Hybrid Nodes with Sensors [Hamza-Lup, 2004] use an architecture for interactive distributed mixed and virtual reality environments. With the advances in sensor technology there is a vision that in future systems a significant amount of data will be collected from sensors and devices attached to the participating nodes. This new architecture for sensor based interactive distributed mixed-reality (MR) and virtual reality environments falls in-between the atomistic peer-to-peer model and the traditional client-server model [Hamza-Lup, 2004]. Each node in this system is autonomous and fully manages its resources and connectivity.

There is also relevant work reported by the Web3D Consortium, although the current status of this work is not clear to us. VRTP (Virtual Reality Transfer Protocol) [Web3D, 2001a] is being developed to maximize the network abilities of shared 3D worlds and to keep pace with the subsequent explosion of network demand by large-scale DVEs. Virtual Reality Modeling Language (VRML) [Web3D, 2004a] and now its enhanced successor X3D [Web3D, 2004a] permit the construction of large-scale virtual environments using the Internet and the Web. Additional capabilities for many-to-many peer-to-peer communications plus network monitoring need to be combined with client-server capabilities of HTTP. VRTP is designed to support interlinked VRML worlds in the same manner as HTTP was designed to support interlinked HTML pages. The aim of the architecture is to include different components for providing the full spectrum of functionality that exists between client-server and peer-to-peer approaches. For example servers would be responsible for shared 3D worlds and peer-to-peer components would be used to provide scalable behaviour interactions between numerous interacting worlds.

Another Web3D working group DIS-Java-VRML has been created to establish initial networking conventions for building multicast-capable large-scale virtual environments. DIS, Java and VRML can provide all of the pertinent capabilities needed to implement such environments. DIS is essentially a behaviour protocol tuned for physics-based (i.e. "real world") many-to-many interactions. Java is the programming language used to implement the DIS protocol, perform mathematical calculations, communicate with the network and communicate with the VRML scene. VRML 3D graphics are used to model and render both local and remote entities [Web3D, 2001b]. This presents possibilities for implementing virtual worlds on both high-cost and low-cost hardware. The

hope of the working group is that the construction of large physics-based virtual worlds becomes inexpensive and scalable.

The overview of the earliest and latest DVE architectures shows that rather than defining radically new architectures, recent research has explored ways of augmenting or combining two basic structures to support greater scalability or performance.

2.1.3 Summary

In the two previous sections the nature of DVE applications, networking design challenges, and current architectural approaches for these applications have been discussed. In general, distributed approaches (peer-to-peer) are more efficient and scalable than centralized (client-server) approaches.

At the same time, to be able to create the robust, scalable peer-to-peer systems, developers of DVE applications have to take into consideration various scalability design aspects. The key scalability requirements that we take into consideration to define our new architectural solution proposed in this work are discussed in the next section.

2.2 Key Scalability Aspects for Large-Scale DVEs

2.2.1 Scalability and Resource Management

Scalability refers to the effects of increasing the scale of a problem or system. So a scalable system is one for which the cost of increases in scale is regarded as "small" or "acceptable" [Greenhalgh, 98]. The size or scale of a DVE is measured by the number of entities that may simultaneously participate in the system. A system is scalable if it can handle a large number of entities before its overall performance degrades beyond an acceptable quality. With growing numbers of users several factors of a DVE will also scale up. Greenhalgh defines these factors as follows:

- Geographical distance between participants, co-located or remote.
- Network distance between participants.

- Total user population, i.e. all those who use the system, whether they are currently doing so or not.
- Number of simultaneous participants.
- Scope of participant awareness, i.e. the fraction of the total environment to which a single participant has access at any moment.
- Complexity of the virtual environment.
- Richness of communication, e.g. number of media, level of detail or fidelity.
- Variability of delivery platforms.

Scaling up of any of these parameters of an application will lead to significant growth in the total resources required to support the DVE system. Improved resource management can generally improve the interactive performance of the large-scale DVEs. For example, reduced network load can lower the transmission latencies caused by network congestion and can therefore ensure that shared state information and updates are exchanged more quickly.

Aside from scene rendering and graphics processing, network bandwidth and state maintenance processing are the two most significant bottlenecks in a DVE. Therefore it is necessary to minimize the demands on these resources to achieve improved scalability and performance.

Network bandwidth provides the cornerstone of a DVE. The network is used to exchange information so that participants maintain a consistent view of the virtual environment's state and so that participants can engage in real-time conversation [Singhal, 99]. Singhal explains that network bandwidth requirements increase with the number of users in a DVE because of three reasons.

1. Each of the additional participants must receive the initial DVE state; as well as updates that the other participants are already receiving. Unless the new users are on the same local area network (LAN) as the existing users, additional network resources are required to deliver that information.
2. Each additional participant potentially introduces new updates to the existing shared state of the DVE and new interactions with existing users. Network bandwidth is required to disseminate these additional updates.
3. Each new user introduces additional shared state to the DVE. At a minimum, each user has a position, orientation, and graphical

representation. More network bandwidth is required to share this new state information and updates to it.

These additional interactions and state updates must be exchanged over the network. This exchange consumes network bandwidth, and those packets must be generated, sent, received, and processed, and rendered by the host. Therefore there is an increase not only in the network bandwidth required to maintain data, but also an additional burden on the host processors which must render the new users as well as manage additional state updates and interactions.

Singhal refers to the relationship between networking and processing in a DVE as the DVE information principle:

> *"The resource utilization of a DVE is directly related to the amount of information that must be sent and received by each host and how quickly that information must be delivered by the network".*

Resource management allows for improving the scalability and performance of DVEs and it is an active area of research. Each DVE uses a different combination of techniques, and a single commonly accepted suite of techniques has not yet emerged.

In the following sections we discuss three resource management categories that are key scalability aspects. They are: data flow restriction, leveraging of limited user perception, and system architecture modifications. All of these three approaches are used in our proposed architecture that aims to increase the DVE's scalability (see Chapter 6).

2.2.2 Data Flow Restriction

One of the approaches to keep DVE applications scalable is reducing the number of hosts that receive data. The underlying assumption behind data flow restriction is that DVEs contain enormous amounts of information, yet an individual user only needs to know a small portion of the total available information. Hence, by sending information to only those hosts that really need it the aggregate bandwidth requirements of a DVE could be reduced.

The aura-nimbus information model [Greenhalgh, 97] shown in Figure 2-4 illustrates the data flow optimization within a DVE.

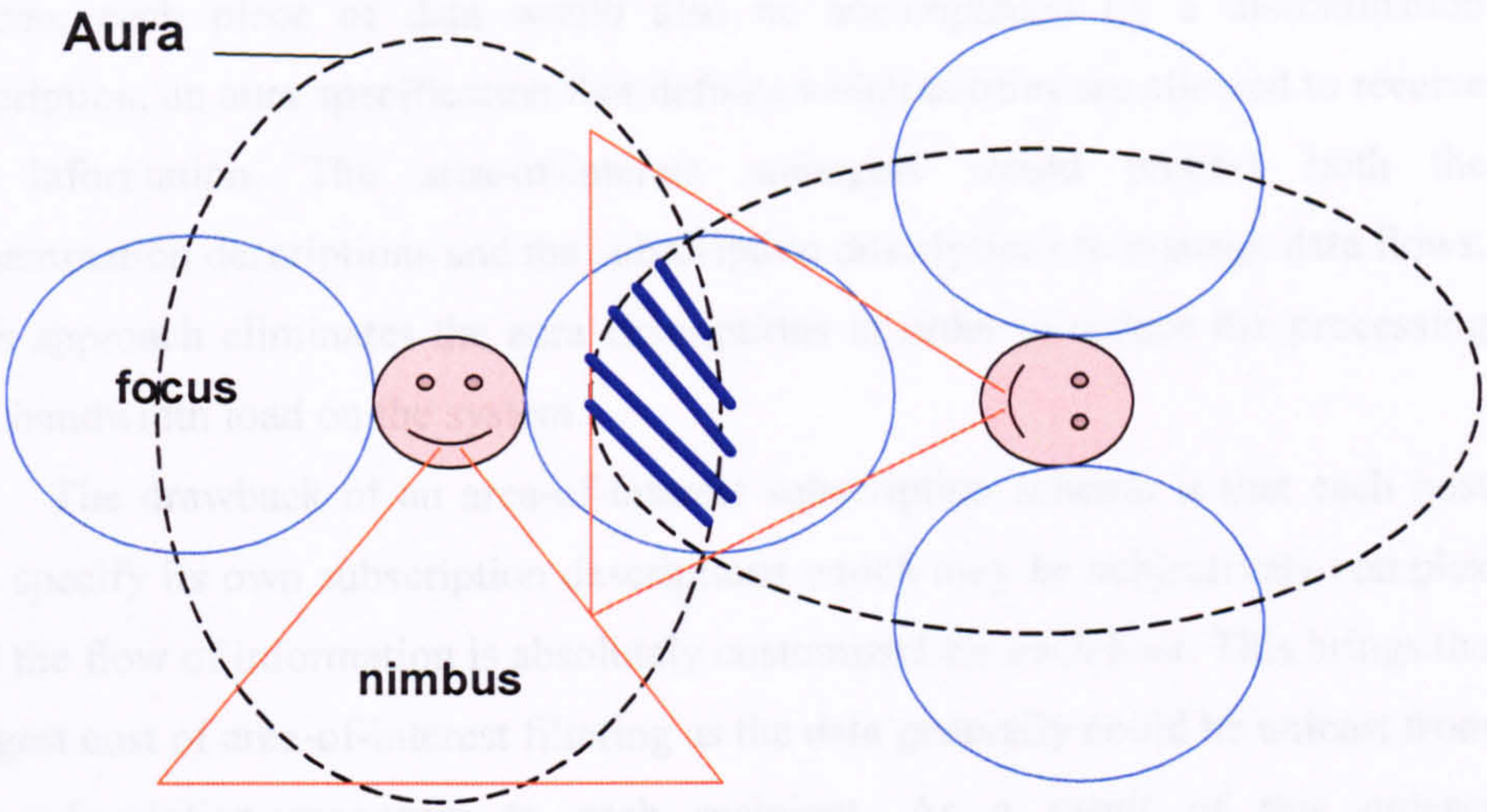


Figure 2-4 Aura-nimbus model

This approach takes into consideration the facts that humans can only see things in their field of view whereas they can hear acoustic events that are behind them.

For DVEs the aura of an object is the 3D volume in which the object can interact with other objects. There is then the concept of focus, the 3D volume in which an object can perceive events. The nimbus is the spatial region in which events caused by this object can be perceived by other objects.

The motivation behind these concepts is that information has only to be delivered to entities whose nimbi intersect with the source entity's aura. The nimbus of an object can be different for each medium as a consequence of the differing properties of human perception for different media.

To make the aura-nimbus model scalable two main techniques are deployed by data flow management: area-of-interest filters and multicasting.

The first of them is area-of-interest filtering, where hosts transmit information to a set of subscription managers, alternatively referred to as "area-of-interest managers" or "filtering servers" [Singhal, 99]. These servers try to link the available information with hosts that are interested in that information. Area-of-interest filtering is simply a restricted form of the "pure aura-nimbus" model [Greenhalgh, 97] applied in both DIVE and MASSIVE. The subscription

descriptions really specify the subscribing entity's nimbus. In a pure aura-nimbus system, each piece of data would also be accompanied by a dissemination description, an aura specification that defines which entities are allowed to receive the information. The area-of-interest managers would process both the dissemination descriptions and the subscription descriptions to manage data flows. This approach eliminates the aura descriptions in order to reduce the processing and bandwidth load on the system.

The drawback of an area-of-interest subscription scheme is that each host can specify its own subscription descriptions which may be subjectively complex and the flow of information is absolutely customized for each host. This brings the biggest cost of area-of-interest filtering as the data generally could be unicast from the subscription managers to each recipient. As a result of this unicast communication the duplications in transmission, and therefore inefficiency, means that, scalability problems will occur. Network multicasting could eliminate this duplication, but it cannot be exploited well with area-of-interest filtering.

Area-of-interest filters provide what can be called *intrinsic filtering* because the filter must inspect the application content within each packet to determine whether it should be delivered to a particular destination host [Morse, 2000]. The set of fields that must be examined is determined by the area-of-interest filter provided to the subscription manager. By being aware of the packet contents, intrinsic filtering can dynamically partition data based on fine-grained entity interests. An alternative approach, *extrinsic filtering*, does not consider application level information but instead filters packets based on network properties such as a packet's destination address. Extrinsic filtering can be implemented more efficiently than intrinsic filtering and may even be performed within the network itself. At the same time, the level of filtering cannot be as sophisticated as that provided by intrinsic filtering. An alternative hybrid approach that combines the extrinsic filtering with some elements of intrinsic filtering might offer better solutions. In this research we explore the ability of active networks to achieve this goal by adding a flexible application layer filtering to the network layer multicasting.

Multicasting helps to design scalable systems and is commonly used to implement extrinsic filtering within DVE applications. Multicasting is a network protocol technique whereby the application transmits each packet to a multicast

group by supplying a special multicast address (for more details on multicasting see section 2.2.4). The packet is only delivered to those hosts who have subscribed to the multicast group. When a packet is transmitted, a single copy of the packet travels along each network link in the multicast tree. As a result of this distribution method, the packet only travels to parts of the network that have interested subscribers. If the data is partitioned in some way, each partition can be sent on a separate network connection. The key challenge for implementing multicast-based solutions is determining how to partition the available data among a set of multicast groups and therefore partition traffic in the network.

One approach to solve the multicast mapping problem is group-per-entity allocation. This approach assigns a different multicast address to each entity in the DVE [Abrams, 98] and allows each host to receive information about all hosts that lie within its nimbus. However, unlike the area-of-interest approach, information subscriptions can only be made on a per-entity basis, rather than a per-packet basis. To discover the nearby entities hosts learn about the entities that are located in close proximity in the virtual world and the multicast addresses used by those entities. Such an entity location service is provided by *beacon servers* in the Diamond Park DVE developed at Mitsubishi Electric Research Laboratories (MERL) [Barrus, 96].

This approach is potentially costly in large DVEs because it consumes a large number of multicast addresses (see section 2.2.4 on limitations of multicast). Also it creates an overhead on network routers which must process the corresponding large number of group membership join and leave requests. Finally, most network cards can only support a limited number of simultaneous multicast addresses.

An alternative approach to create multicast groups is a group-per-region allocation. In the group-per-region approach instead of assigning a multicast address to each entity we can partition the virtual world into regions and assign each region to one or more multicast groups. Each entity transmits its data to groups corresponding to regions that cover its current location. The group-per-region approach effectively manages communication overhead by limiting aura and nimbus specifications by only spatial categories. Each entity changes its target group as it travels throughout the virtual world and upon entering the new region learns the multicast addresses associated with that region.

PAGE/PAGES
EXCLUDED
UNDER
INSTRUCTION
FROM
UNIVERSITY

closely the fidelity requirements of the users at the receiving hosts. Nearby viewers expect to see the entity rendered with full graphical detail and with accurate structure, position, and orientation. On the contrary, distant viewers can tolerate rendering the entity with less graphical detail. Many inaccuracies cannot even be detected on a finite-resolution graphical display. Thus, transmitting high resolution to these distant viewers imposes unnecessary bandwidth burdens on the network and processing burdens on the receiving hosts.

In graphics systems Level-of-Detail (LOD) processing reduces the rendering time. In DVE systems the LOD representation is a method to save bandwidth. Methods for spatial partitioning divide a three-dimensional space into subspaces and compute a visibility relation or use a predefined one. By means of the visibility-relation the bandwidth as well as the computing time for a scene can be reduced, because only visible subspaces must be loaded over the net and rendered [Diehl, 2001]. The closer the viewer gets to the object, the more details are rendered. To render a scene graph of the object the browser chooses between several representations of the object that have different levels of fidelity depending on the distance to the object from the viewer. Multiple representations of one object with decreasing resolution (LODs) are used to reduce rendering cost for distant or otherwise less important objects. As only one level of detail of a given object can be displayed at any time instead of transmission of geometry data for a scene as a whole, it is possible to transmit exactly packets with certain LOD to certain receivers. There is the method when each entity can transmit multiple independent channels that provide information at a different level of detail. However, the number of channels for each entity is constrained due to the additional cost of supporting large numbers of multicast addresses within DVEs.

Recently the complexity of 3D models of objects has increased radically along with rising numbers of DVE participants, so transmission of 3D content develops into a major scalability issue in DVEs. As the 3D data sets have turned into huge geometrical databases, the earliest approaches with using a small set of discrete LODs have evolved into new methods using 3D data compression, multiresolution techniques and progressive transmission (for more details see Chapter 4). For large-scale DVEs it has become impractical to store full copies of the environment that must contain all levels of detail of all objects on every computer because the whole scene must fit into memory. This prohibits the

exploration of large, continuous data spaces and would destroy the sense of immersion. Other techniques explore the idea that LODs with progressively higher detail will be transmitted as a participant is approaching an object [Schmalstieg, 96]. Such on demand 3D data transmission would gain significant savings in network bandwidth and local memory requirements, allowing handling of more complex, more interesting data sets between ever-growing numbers of participants.

2.2.4 Architecture Design

This final resource management category involves changing the network architecture of the DVE system to enable more efficient information dissemination. By optimizing information delivery, these techniques can save network bandwidth, reduce the number of hosts that receive each packet, and reduce the timeliness requirements on packet delivery. To offset these advantages, these techniques demand additional computational resources among the DVE hosts and supporting infrastructure to determine the optimal message routing and delivery [Singhal, 99].

Current architecture modifications merge traditional peer-to-peer and client-server techniques (see Section 2.1.2) to achieve scalability. These hybrid systems mainly consider two ways in which the network architecture can be changed: deploying server clusters and peer-server architecture. These methods assume that conventional network connections are used and all additional computations are performed only on DVE hosts. Recent research, for example the work presented by Zabele *et al* on active networking for distributed simulation [Zabele, 2002], indicates that there are also new technologies such as an active networking that might benefit large-scale DVEs (for details on active networks see Chapter 3).

2.2.4.1 Multicast Communications

A lot of past and current DVE systems use IP multicast as an architectural approach to improve scalability and performance.

IP multicast is a many-to-many network delivery mechanism based on the UDP datagram. Senders of IP multicast packets do not send to a particular Internet address, but instead send the packet to a group address or an IP multicast address. IP version 4 (IPv4) reserves Class D addresses from 224.0.0.0 through 239.255.255 as multicast group addresses [Wittmann, 2001]. An advantage of multicast is that because the packets are addressed to a group rather than to a specific receiver, the sender never needs to know the set of receivers, so it may be arbitrary in number. Another advantage is that only one copy of the packet needs to be transmitted by the sender no matter how many receivers are listening. Receivers must "join" the multicast groups that they interested in. Joining is achieved by sending an Internet Group Management Protocol (IGMP) packet to a well-known multicast address. This "join" packet informs routers listening on this address that the sender would like packets from a certain multicast group forwarded on to the local network segment. As a result senders transmit on many different channels, and receivers tune to the channels they are interested in. Multicast routing protocols e.g., Distance Vector Multicast Routing Protocol (DVMRP), Protocol Independent Multicast (PIM) are used to route multicast packets. Although there is nothing inherently limiting in the IP multicast architecture, there are some problems and limitations with current implementation on the Internet.

a) Address space limitations

One limitation of multicast is the amount of address space available, and how it is allocated. Under the IPv4 mapping to the Ethernet, the address space for multicast is limited to just over 8 million addresses [Stevens, 98]. Taking into consideration that the possible numbers of large-scale DVE participants could reach millions (see section 2.1.1) this resource may become insufficient. However, this limitation is not a fundamental as it would be improved by IP version 6.

Already IP version 6 (IPv6) is being used on the Internet in the form of the 6Bone [Durand, 99], an IPv6 network tunneled over the regular Internet. Under the current IPv6 mapping to Ethernet, the address space used for multicast routing is over 32 bits, but has allocated space for over 112 bits. IPv6 is designed to run side by side with its predecessor IPv4, allowing a slow but easy migration to a

native IPv6 Internet. Given the massive investment in IPv4 routers currently deployed, the conversion process will probably take a decade [Tanenbaum, 2003].

b) Router configuration problems

Even if a software application has a set of addresses reserved for its use, there are still some of routers on the net that are not configured for IP multicast. The early-introduced method to get multicast traffic across the vast Internet is via the MBone (Multicast Backbone) [Eriksson, 94]; a tunnelled network similar in concept to the 6Bone. Currently, aside with Mbone, manufactures like CISCO company, make multicast a standard feature of the router's functionality. Increasingly, multicasting and Quality of Service (QoS) are coming together, as discussed in [Striegel, 2002].

c) Unreliable nature

Multicast delivery systems were first used for the delivery of time critical data such as video and audio streams, where it is pointless to retransmit lost packets. The design of protocols to support scalable multicast for data types that demand reliable delivery, for example state update messages or 3D content within DVE system, is a non-trivial problem.

The Scalable Reliable Multicast framework (SRM) [Floyd, 97] provides basic functionality for scalable application level reliability in multicast environments. SRM is fully decentralized (no ring or central controller) and handles arbitrarily large groups of participants. To achieve the main goal, which is reliability, receivers learn that they are missing data either from gaps in sequence space or from someone else's report and multicast a "repair request" to ask for missing data. It is possible for transmitters to be swamped by such NAK (Negative AcKnowlegement) requests (known as the NAK implosion problem). SRM addresses this problem by using a scalable NAK algorithm to randomly select one or more participants who should reply to the retransmission request (i.e., any participant can help with the repair of the lost packets, not just the original source of data). SRM methods are used in DIVE to reduce the amount of message passing and increase reliability and scalability [Hasgand, 96].

The future development on the Internet that will help with the reliability problem with multicast is the concept of QoS. QoS allows a router to guarantee network qualities such as latency, jitter, and bandwidth. Although there are several competing systems for handling QoS [Wang, 2001], the leading research

seems to be focused on "differentiated services". The Differentiated Services approach divides the traffic into a small number of classes and allocates resources on a per-class basis. The highest quality of service offered by "differentiated services" is called "Premium" or "Virtual Leased Line". This "Premium" service guarantees, among other attributes, an amount of bandwidth. As long as a stream stays within its bandwidth limits, its packets would never get dropped due to router congestion. This would prevent multicast packets (e.g. the data or 3D packets) from being dropped due to congestion, making multicast much more reliable without the need for supporting reliability protocols. But, this approach is less efficient for some traffic patterns. The reserved bandwidth resources might be wasted for example for bursty traffic where the high bandwidth actually is needed only for short periods of time. Also the slowing down the traffic to fulfil Differentiated Services bandwidth limits might increase delays.

d) Time to join problem

The other difficulty with using multicast is the considerable delays caused by joining and leaving processes. When a host requests to join a multicast group it has to wait an uncertain amount of time to receive data from the multicast group. This time is required to complete the following tasks. When a computer joins a multicast group, it sends an IGMP message to the nearest router. The router then subscribes to the multicast group by sending messages to other neighbouring routers. This process continues along this new branch of the multicast group's distribution tree until it is joined onto the main tree. Time to join the group is therefore the time to propagate the subscription request up to the nearest existing branch of the distribution tree, so that multicast packets can be routed to the new member of the group. This time is considerable (e.g., hundreds of milliseconds to join a group and tens of seconds to leave a group) and is not easy to improve [Wittmann, 2001]. To be able to minimize the impact of these delays, an application can decrease multicast subscriptions.

Although the current multicast implementation on the Internet has its limitations, research is well on its way toward supporting large-scale multicast applications with greater numbers of multicast addresses and wide-spread employment of multicast routers. But it is worth to say that even with network-level multicast and peer-to-peer communication employed, a system might fail to

scale. To support the millions of future DVEs participants there is the need to improve the current multicasting architectural design.

2.3 Summary

This chapter introduced distributed virtual environments and scalability issues that are important for the design of large-scale DVE applications.

In the beginning of this chapter DVE applications have been defined and the future perspectives of emerging new types of large-scale DVE applications and present challenges have been introduced.

Then the current architectures that are used for DVE applications have been discussed. The existing DVE systems that illustrate the three major approaches client-server and peer-to-peer architecture and hybrid approaches have been presented.

The second part of this chapter contains a discussion about the key scalability aspects for large-scale DVE applications. Firstly, scalability and resource management in DVEs have been introduced. Then, three approaches to achieve the scalability of DVE applications that are used in the novel architecture proposed in this thesis have been discussed. Data flow restrictions provide the saving of major resources such as the network bandwidth and processing on hosts and therefore increase the scalability of DVE systems.

To reduce the impact of multicast limitations on scalability and performance of DVE, the multicast groups must be carefully allocated. There are two main approaches for group management for multicast applications: small groups and large groups. Both of them have their disadvantages. The small groups approach needs a large number of multicast group addresses, consumes host and router resources to support a lot of small groups, and imposes considerable delays due to frequent joining and leaving processes. Large groups cause the need of the delivery of the huge amount of data to every member of the one large multicast group. The advantage of the large groups approach is that it avoids the need for large numbers of multicast addresses and decreases delays related to the joining and leaving processes. So, it might be possible to enhance the large group approach by applying additional filtering within one multicast group and therefore reducing the total amount of exchanged information. This work explores the

suitability of this approach that combines extrinsic filtering (see Section 2.2.2) with some elements of intrinsic filtering. Augmenting intrinsic filtering might bring the reflection of the application level information into the simple network multicasting filtering and could allow more fine-grained filtering of flows of information within large multicast groups.

Our architecture proposed in this work facilitates the coarse-grained (large groups) approach to peer-to-peer multicast by adding the intrinsic application level filtering based on two concepts:

1.Using active network techniques.

2.Level of details concept and progressive transmission are used for 3D graphics and exploited by active network filtering.

This approach is aimed to improve 3D content transmission between participants within large-scale DVEs.

Before presenting the more detailed explanation of our approach in chapter 5, the background knowledge about active networking and 3D graphics progressive transmission methods that are exploited in our work is required to be discussed. Hence, the next two chapters are concentrated on these two areas of computer science and the current research in these areas that has given the inspiration for our work.

Chapter 3

Active Networks

In recent years the Internet has changed dramatically. It is rapidly evolving from a set of cables and routers that carry packets to a complex infrastructure that delivers a set of sophisticated services to end users. Services can range from bit transport to distributed value-added services like video teleconferencing, virtual private networking, data mining, and distributed interactive simulations.

The emergence of the World Wide Web and the beginning of the commercialisation of the Internet have caused this remarkable growth of the Internet since the mid 1990s.

The main reason of this growth is most likely the simplicity and cheap deployment cost of Internet technology. The simplicity was based on employing in packet switched networks simple store-and-forwarding router devices with minimum processing that aimed only to support forwarding of packets. But, the requirements of the transmission of packets in packet switched networks have changed recently. These changes can be outlined as follows:

1) The Internet has evolved from an academic research platform to a wide public commercial network. This has stressed network security, efficiency, and scalability requirements.
2) Multimedia applications, such as continuous or interactive audio and video have created new delivery requirements that differ from a simple best-effort packet forwarding which is provided by the original packet switched networks. Therefore, the Internet now presents mostly a multimedia data network rather than a pure data network.
3) Introducing recent technologies, such as wireless or satellite links, that had not been considered at the early Internet stages.
4) The increasing performance of general computers has approached the stage where it has become practical and economically feasible to use more powerful computers in the network instead of simple store-and-forwarding routing devices.

As the requirement for the new complex network services is continuously increasing, a recent tendency has been seen in taking advantage of placing

processing capabilities inside the network. An increasing amount of user or application specific processing is accomplished on the edges of the network at the intermediate nodes where it is most efficient. The most important examples of this tendency are network caching, deployment of security firewalls, Quality of Service (QoS) mechanisms, application specific gateways and proxies, as well as application-layer network support.

Still, the development and deployment of new network services, for example services that work on the IP layer, is too slow to standardize and implement. It cannot catch the rapid growth of requirements in various applications. Therefore, there is a need to improve the current Internet architecture in order to allow a more rapid introduction and programmability of new services.

Recently the idea about network-embedded functionality has grown. Changing the network, from a static entity that only forwards packets from router to router into a dynamically programmable system, offers new possibilities, for example customized data transmissions and application specific network support.

The devices that execute processing inside the network nowadays are: the routers which are closed boxes that execute a restricted set of vendor software; computing and storage servers which are typically dedicated to supporting one type of service. An alternate architecture is to have an open infrastructure in which specific services can be installed and instantiated on demand; much as it is done on a PC today [Gao, 2000]. Active networking approaches may help to obtain such programmability of the network. The key idea in Active Networks is that by placing computational resources directly within the network specifically to support end-user processing requirements, higher performance will be achievable than by conventional means, service deployment will be improved and new classes of service will become possible.

3.1 What is an Active Network?

Active networking is a recent approach whereby intermediate nodes inside the network can be involved in the customized processing of control and data traffic [Schmid, 2002].

The concept of active networking emerged from discussions within the broad DARPA research community in 1994 and 1995 on the future directions of

networking systems. As a result of the initial DARPA-funded research program for active networking [DARPA, 98], an informal working group has been formed to define the scope and fundamental tenets of active networking along with an architectural framework [ANW, 98].

According to DARPA, an *active network* consists of a set of *active nodes* that are connected by variety of network technologies. Each active node runs an *operating system*, responsible for managing the node's resources, and one or more *execution environments*. Each execution environment implements a virtual machine that processes the delivered active packets. An execution environment may provide general computational services or simply a forwarding engine whose computation is controlled by packet data.

This is a new networking paradigm for data networks whereby the focus changes from pure packet forwarding to dynamic programming of network nodes. In this case programmability means that users (for example, network administrators, privileged users, or even end users) are able to control dynamically the network to process packets in a required way, rather in a static manner offered by vendor. Active networks normally implement code distribution and code execution mechanisms to facilitate this, so that injecting code programmed by users can enforce control of the networks [FAIN, 2000].

For example, a user of an active network could send a "trace" program to each router and arrange for the program to be executed when their packets are processed. Figure 3-1 illustrates how the routers of the network could be augmented to perform such customized processing on the packets flowing through them. These active routers could also interoperate with legacy routers which transparently forward packets in the traditional manner [Tennenhouse, 97].

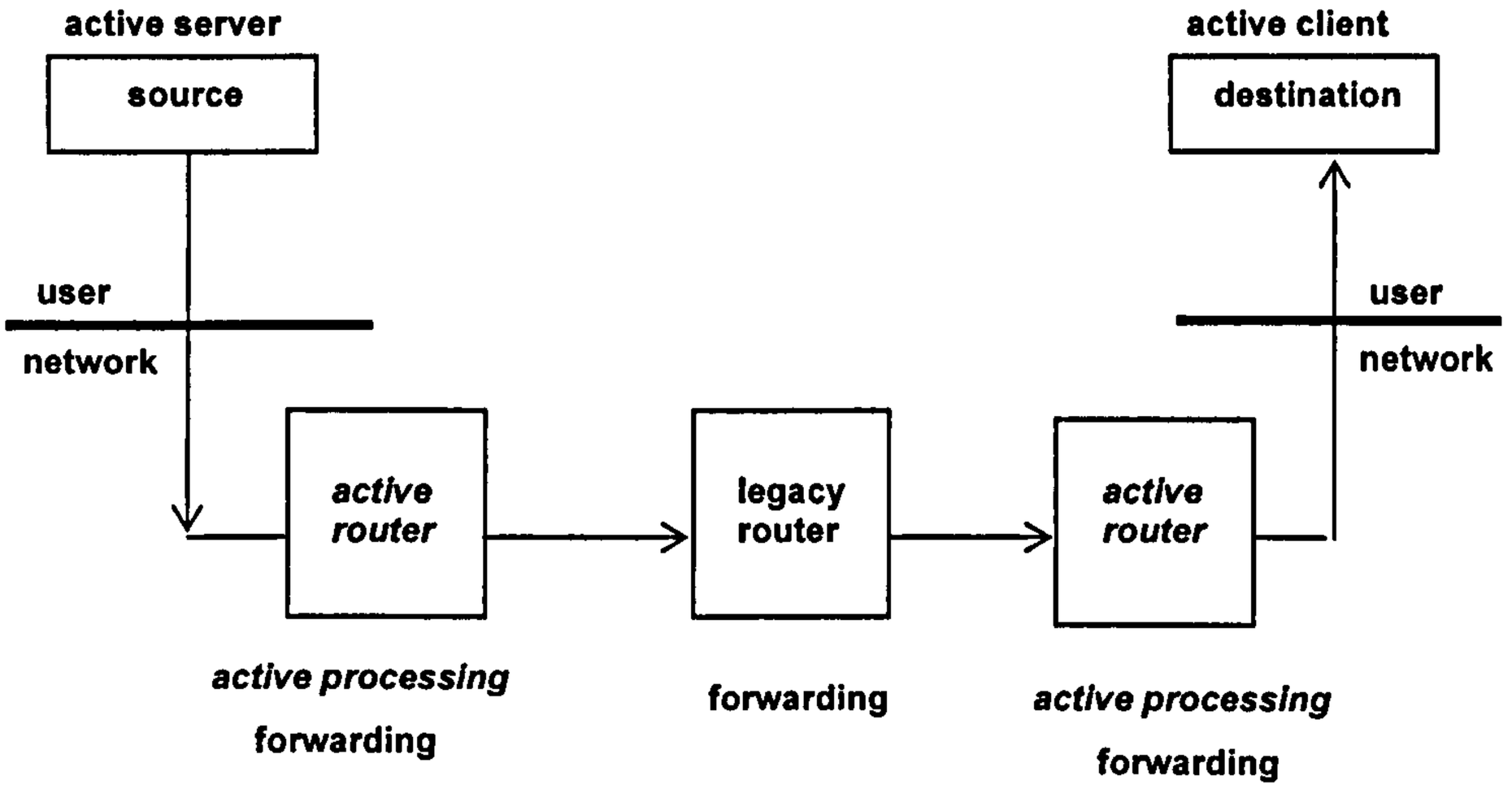


Figure 3-1 Active network structure

The active networking idea of messages carrying procedures and data is a significant step in the evolution of packet-switched networks. Combined with execution environments within network nodes, this program-based approach provides a foundation for expressing networking systems as the composition of many smaller components with specific properties where services can be distributed and configured to meet the needs of applications.

There are two extremes in the realization of active networks. The *programmable switch (or active extension)* approach provides a discrete mechanism that supports the downloading of programs. Within this discrete approach the processing of packets is separated from injecting programs into the node. In contrast the integrated approach - the *capsule (or active packet)* approach is one in which every message is a program. Every message, or capsule, that passes between nodes contains an executable program, consisting of code (for example Java code) and data. When a capsule arrives at an active node, its contents are evaluated. Therefore a program encapsulated in an active capsule defines the processing of the capsule's data by router.

3.2 Challenges of Active Network Design

Computational power inside the network and the extended flexibility create a number of challenges for the design of active networks. The major ones are the efficiency and security of the network. The required bandwidth is constantly increasing; therefore the routers need to process more and more packets at line speeds to keep their internal buffers from overflowing. In case of active networks, executing a program contained in a packet would typically take longer than processing a fixed format packet header [Scott, 98]. This will unsurprisingly limit the efficiency of an active router. Also, with programs being injected into network nodes on the fly, another concern arises - the safety and security of the network. A program can cause inconsistent consumption of resources at a router, or can disrupt/crash a router thus affecting the core network function. Active network architectures try to address all of these concerns. Typical active network architectures, thus, represent a point in the design space defined by three dimensions: flexibility, performance and security [Moore, 2000].

The DARPA-funded research community has agreed upon several fundamental tenets and objectives for active networks to follow the requirements of flexibility, performance and security [ANW, 98]. These objectives could be outlined as follows:

- Network API that enables programmability.
- Communication dominates over computation.
- Network packets are the data units for computation.
- Minimal assumptions on underlying technology.
- Independent node administration.
- Scaling to global active networks.
- Ensuring security and robustness of active nodes.

3.3 Active Network Architectures

Active network architectures show a significant diversity in their programming models and interfaces. However, since all of these architectures have been proposed for packet switched networks, they still have the same

components as the traditional packet switched networks (e.g., networks nodes and packets). Active network architectures provide programmability at one or more of these components to support flexible network service design. At network nodes, programmability is provided in terms of dynamically loadable functionality. In network packets, allowing the packets to carry programs along with data provides programmability. The spectrum of potential programming models ranges from simple configuration scripts to sophisticated software extensions and from conservative to highly dynamic and flexible levels of programmability [Schmid, 2002].

Two directions in active network research that investigate integrated and discrete approaches have been classified by the early active network projects, namely the ANTS project at MIT [Wetherall, 98] and the SwitchWare project at the University of Pennsylvania [Alexander, 98].

3.3.1 Active Capsules: ANTS

The Active Network Transfer System (ANTS) [Wetherall, 98], developed at MIT, represents an integrated approach and is based on active code distribution with using the data packets. It enables the flexible deployment of multiple communication protocols in active networks providing a set of core services including support for the transportation of mobile code, loading of code on demand and caching techniques. ANTS provides a network programming environment for building new capsule-based programmable network architectures. Examples of such programmed network services include enhanced multicast services, mobile IP routing and application-level filtering. Incorporated features include node access, capsule manipulation, control operations and soft-state storage services on IP routers. Active nodes execute capsules and forwarding routines, maintain local state and support code distribution services for automating the deployment of new services.

The three key components used in the ANTS architecture are:

1) The packets found in traditional networks are replaced by *capsules* that carry the processing instructions to be executed by the active routers along the transmission path. Capsule types that share information within the network are grouped into *protocols*; a

protocol provides a service and is the unit of network customisation and protection.

2) Selected routers within the Internet and at participating end nodes are replaced by *active nodes* that execute the capsules of a protocol and maintain protocol state. Unlike ordinary routers, active nodes provide an API for capsule processing routines, and execute those routines safely by using operating system and language techniques.
3) A *code distribution* mechanism ensures that capsule processing routines are automatically and dynamically transferred to the active nodes where they are needed. This component does not exist in traditional networks, and is handled by the system, not the service programmer.

The development of new services with this model requires the service developer to define the capsule types and their processing routines as a protocol structure. To use a new service, an application only needs to supply the protocol definition to the local node and start sending and receiving capsules of the appropriate types.

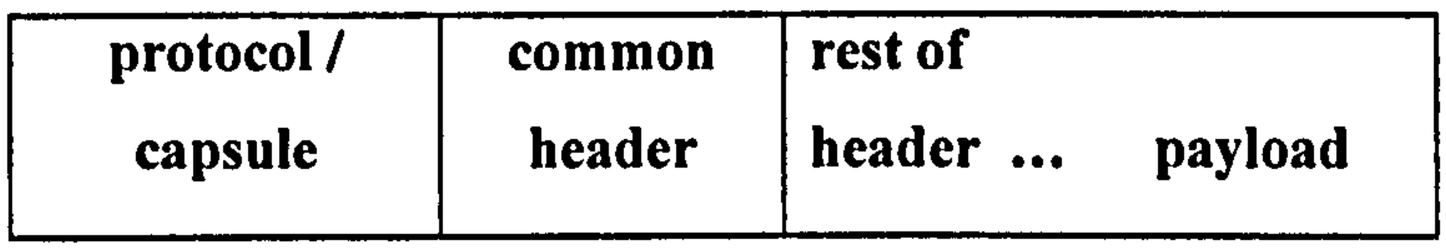


Figure 3-2 Capsule format

ANTS tackles the performance or security challenges as well as offering flexibility. To achieve this goal this approach guarantees some level of protection, allocation and performance of shared resources in an untrusted environment. Protection is based on the inability to specify processing for another user's packets and the encapsulation of protocol state by the associated capsule processing routines. Allocation is based on the limited resources that will be granted to each packet by nodes.

The functionality of ANTS has been demonstrated through the development and deployment of several protocols and extensions, including a high performance reliable multicast extension [Lehman, 98] and a TCP SYN-

flooding defence protocol [Van, 97]. Furthermore, ANTS has been incorporated at different sites of the ABone [ABONE], which is the active networks test bed project in US, as one of several active network technologies.

3.3.2 SwitchWare

The SwitchWare active network architecture [Alexander, 98] was developed at the University of Pennsylvania and is based on the concept of active extensions or discrete approach where programs and data are distributed separately.

In comparison with active packet based programming which is inherently restricted to lightweight programming (only small programs can be included in every packet), the "programmable switch" approach has the potential for more fundamental extensions.

The SwitchWare network solution is based on three distinct layers: a *secure active router, active extensions*, and *active packets*.

At the bottom layer of this architecture or at the operating system level, an active IP-router component is responsible for providing a secure foundation that guarantees system integrity.

The middle layer of programmability is formed by active extensions. These active extensions are called *switchlets* and are basically the libraries. Active extensions, like active packets, can be dynamically loaded into secure active routers through the set of security mechanisms that include encryption, authentication and program verification. While switchlets do not go through the network it is not required for them to be lightweight and portable. For that reason active extensions can be written in general-purpose programming languages. Also the correct behaviour of active extensions can be verified off-line by applying "heavyweight" methods, since the deployment of such extensions is done over slow time scales. When executed on a node, active extensions provide extended services to the data streams passing through the node. They generally have a long-term impact that exceeds the lifetime of a packet by far. As active extensions are not attached to a particular data stream (i.e., they are loaded out-off-band), they can potentially be applied to multiple or all data streams.

The top layer of the SwitchWare architecture is the active packet layer. Active packets can roam and customize the network in a similar way as capsules do. Active packets are written in functional languages (e.g., Caml and PLAN [Hicks, 98]) and carry lightweight programs that invoke node-resident service routines supported by active extensions. PLAN (Programming Language for Active Networks) has been specifically designed to meet the requirements of active packet programming in environments where low-level functionality can be extended by means of active extensions. In the SwitchWare the active router infrastructure is a secure environment that supports the execution of the upper layers – switchlets, and PLAN packets. Basic services such as routing, address resolution, forwarding, and even PLAN interpretation are implemented as switchlets.

SwitchWare applies heavyweight security checks on active extensions, which may represent major releases of switch code, and more lightweight security checks on active packets. This approach allows the network architect to balance security concerns against performance requirements.

3.4 Active Networks for Supporting Multicast Communication

The usage spectrum of active networks is broad and, of course also includes applications that are not based on group communication. Considering group or multicast communication, there could be an advantage to using active networks to improve performance seen by applications. In this section we provide examples of active networking to improve performance of multicast applications.

Active networking could be useful to facilitate the following aspects of multicasting [Wittmann, 2001]:

- *Heterogeneous quality of service*. RSVP (Resource ReSerVation Protocol) can be used to signal heterogeneous quality of service but does not provide it. Active networks could be used for this purpose. The functions required in adapting a data stream could be loaded and activated in the intermediate systems through signalling. The data units flowing through such a node can then use the service provided by these functions. For example, the data rate of a video could be reduced before entering wireless transmission

links. This way the video stream can be forwarded to and decoded by a wireless attached system.

- *Dedicated error control.* In the heterogeneous systems routers with a wireless outbound link could check their forward-error correction on a data stream forwarded across this link. The probability that data will be received correctly increases with forward-error correction. Active networks can also be used for the individual operation of wireless links through other error control and recovery algorithms.
- *Local group management.* Functions for error control and recovery can also be used to implement local group managers which will dramatically reduce the acknowledgment and feedback implosion at the sender.
- *Subcasting.* This mechanism enables group administrators to contact subgroups over group addresses.
- *Flow and congestion control.* This solution particularly applies to adaptive algorithms for response to or avoidance of congestion.

Since active networks are the subject of very current research, no comprehensive concepts are yet available. However, it is easy to see that multicast communication in particular could gain considerable benefit from exploiting active networks.

The following examples of exploiting active networking within multicast communication illustrate advantages of the active networks architectural approach.

3.4.1 Active Reliable Multicast

Multicast protocols provide a point-to-group communication facility; a multicast protocol is reliable if it continues to try to deliver information until it is received by all members of the group [Wittmann, 2001].

Providing an efficient and scalable reliable multicast service over a wide-area network is a difficult problem; it has been a topic of considerable interest in the network community (e.g., Scalable Reliable Multicast (SRM) [Floyd, 97]). The key challenges in providing multicast reliability are managing bandwidth utilization of bottleneck links, not overloading the sender with retransmission

requests, and keeping latency of retransmissions low. In practice these challenges turn into finding mechanisms for preventing NACK (negative acknowledgment) implosion, distributing responsibility for sending retransmissions, and limiting the delivery scope of retransmitted packets.

In existing end-to-end approaches, considerable effort is made to control NACK implosion and distribute responsibility for repair, but at the cost of increased retransmission latency and/or aggregate bandwidth utilization. These approaches do not offer attractive scoping techniques and are not fault-tolerant and robust to topology changes.

Lehman [Lehman, 98] presents a loss recovery scheme, Active Reliable Multicast (ARM), for large-scale reliable multicast. ARM utilizes intermediate routers to protect the sender and network bandwidth from unnecessary feedback and repair traffic. ARM routers employ three error recovery strategies. First, they suppress duplicate NACKs to control the implosion problem. By reducing the number of NACKs, ARM minimizes the amount of feedback traffic that needs to cross the bottleneck links. Second, a router-based local recovery scheme is used to reduce the end-to-end wide-area latency and to distribute the load of retransmissions. More specifically, routers at strategic locations perform "best-effort" caching of multicast data for possible retransmission; as NACKs traverse upstream towards the original source of the data packet, any router can perform retransmission if the request packet is in its cache. Finally, to reduce network bandwidth consumption, routers use partial multicasting to limit scope of retransmitted packets so they are delivered only to receivers that previously requested them. To accomplish the loss recovery task the routers perform three different actions which are triggered by the receipt of one of three kinds of ARM packets: multicast data packets, NACK packets, and retransmitted packets [Lehman, 98].

The authors report that to build the prototype of ARM they used ANTS. Thus code for ARM can be downloaded into active node as needed.

Whereas the conventional networks approach (e.g., SRM) trades recovery latency off against repair bandwidth, active networks technique (ARM) utilizes network based processing and storage to reduce both.

3.4.2 Aggregated Hierarchical Multicast

This approach considers application scenarios where a large number of sensors and computing devices are connected to the network. The authors in [Wolf, 2003] propose using programmable networks to improve many-to-many communication within large-scale applications which employ sensor devices.

These devices might constantly send status information via multicast to a number of applications or users. One big challenge in this environment is the amount of data generated by such sensors, which depends on the size of the data, the transmission frequency, and the number of senders and receivers. However, for certain applications it is sufficient to receive less accurate, aggregated data from a group of sources. This leads to the possibility of using programmable or active routers to perform such data aggregation inside the network. [Wolf, 2003] addresses how a large number of sources can be organized in a hierarchical structure to allow multiple users to get a view of all sensors at different levels of aggregation. With the control information that is provided by an aggregating node, the user can traverse the aggregation tree by joining different multicast sessions that transmit different levels of detail. This provides a novel communication paradigm that reduces the network overhead from continuously transmitting sources and organizes data in a way that is useful to receivers.

As a demonstration of this active networking approach to multicasting the authors provide two detailed example scenarios: a battlefield simulation system, which aggregates geographic location data of units, and a conferencing application, which aggregates audio data.

3.4.3 Active Video Transcoding

Other suggestions include provision of transcoding services within a network, to address issues of bandwidth heterogeneity amongst destinations in a multicast group. The generalization here is that the application knows what transcoding processing to apply, the network knows where best to apply it.

An elegant realization of this idea is seen in the work of Keller *et al.* [Keller, 2000] in their Active Router architecture for multicast video distribution. Conventional adaptive approaches to video distribution rely on feedback from

receivers to adapt the video output rate to network load. This is not suitable for interactive applications primarily because of receiver latency. When used with a set of receivers or links with heterogeneous bandwidth, the effect is to adapt to the slowest receiver which can severely penalize the faster receivers. Another approach for multipoint distribution is to transmit the video as different layers (requiring different bandwidths) over different multicast addresses. Receivers subscribe to the most appropriate layer for them, depending on the bandwidth available and hence the quality they wish to receive. Receivers will make "join" experiments to decide which layer is appropriate and this can take several seconds. Joining multicast groups is an expensive operation, and the approach does not perform well under frequent changes in link load that are common on the Internet.

Keller *et al.* employ an active wavelet-based video encoding scheme to provide video scaling. This approach allows the decision on whether or not to forward a video packet to be made in the router. This decision is based on observation of the router's link load, without requiring any feedback from the receiver. Adaptation of multimedia traffic is called *media scaling*, and the router implementing video scaling is called a *media gateway*. The number of proposed video encoding and decoding schemes used for media scaling is already large and it is growing constantly, also modifications to these schemes are frequent. For these reasons it is crucial that a scaling algorithm can be deployed automatically and on-demand to routers implementing video gateway functionality. The router must be on-the-fly extensible to perform the scaling without service interruption. Application-specific active packet processing at a programmable router is supported by the Active Router Plugins software architecture. *Router plugins*, as Keller *et al.* describe, are code modules which implement specific datagram processing functionality like encryption, congestion control, reliable multicast or, for example, Wave-Video scaling. Within Wave-Video scaling a wavelet transform is applied to the image. This process splits an image into a low-frequency and three high-frequency subbands and is repeated on the low-frequency subband at each level of the transformation. In this architecture the regular IP forwarding loop is extended to look for special headers between the IP header and the UDP and TCP header, which reference plugins. If such a header is found, the packet is passed to the referenced plugin prior to being forwarded. If

the referenced plugin is not present on the system, it is downloaded from a so-called *code server* the network and automatically installed in the kernel. This on-demand downloading of plugins is called *Distributed Code Caching for Active Networks* (DAN) [Decasper, 99]. The special active headers within WaveVideo datagrams are represented by Active Networking Encapsulation Protocol Version 2 (ANEP) [ANEP] header and DAN header. The ANEP header is the active networking community's standard header to precede the active networking payload. The DAN header is specific to Active Router Plugins environment. The DAN header essentially identifies the plugin to perform the video scaling. It carries a WaveVideo-specific tag which is used by the scaling algorithm to decide whether or not to forward the packet based on the current load situation of the outgoing link.

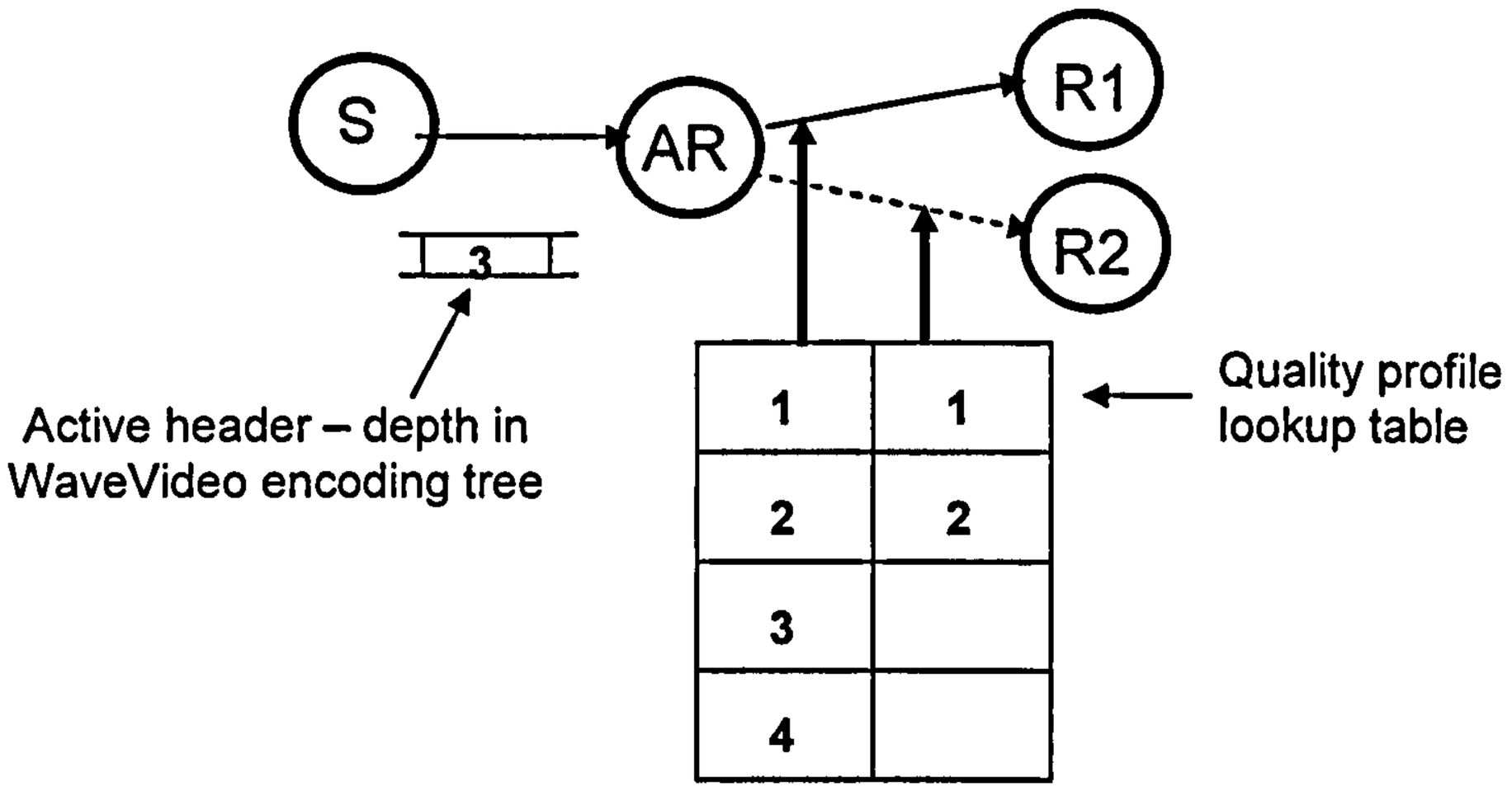


Figure 3-3 Active video scaling

The active tag describes the frame's content such as the corresponding frequency subband, colour channel, and depth in the wavelet tree. The tag allows for reduction of the stream's bandwidth by selectively dropping packets belonging to a particular quality set. It serves to prevent the randomly dropping packets to ensure that datagrams containing low-frequency coefficients (which define the overall image) reach the video sink even under congestion. The scheme suggested by Keller *et al.* continuously monitors the forwarded video stream and periodically adapts the quality profile to fit the available bandwidth. Thus, if the forwarded bandwidth is higher then that which is available the quality profile will be modified so that more high-frequency parts are eliminated from the video stream.

Figure 3-3 shows a multicast communication group that contains one sender (S) and two receivers (R1 and R2). The quality profile for the link connected to receiver R2 is set to forward only packets with a depth tag lower or equal to level 2. Therefore incoming active WaveVideo packets with an active tag marked, for example, by level 3 will not be forwarded to R2.

The active multicast video distribution architecture, presented by Keller *et al.* allows adaptation of the video stream to fit the momentary network load. This video adaptation scheme ensures that low-frequency wavelet coefficients (which are crucial for the general definition of the image) are always forwarded but drops high-frequency parts (that describe image details) if bandwidth becomes scarce. It dramatically improves the video quality on the receivers even with transmission through congested lines.

3.4.4 Router-assisted Control for Layered Multicast

In the layered multicast, video data are encoded into multiple layers – a *base layer* and *enhancement layers*. The sender transmits the encoded data over separate multicast groups, and each receiver determines how many layers to subscribe depending on its capability or desired level of quality of video. Network-based layered multicast approach proposed in [Son, 2003] exploits a TCP-friendly fine-grained traffic control scheme with routers assistance for layered multicast. The proposed scheme is based on a Network-based Layered Multicast (NLM) [Kang, 2001], which is a layered video delivery scheme using network-wide traffic information. To support the fine-grained traffic control, the proposed scheme includes the session-based traffic adaptation algorithm using network parameters: total number of outgoing sessions and the queue occupancy ratio of video traffic.

As the number of sessions increases, the number of video sessions that could be affected by the control scheme increases. This may cause rapid traffic change and video reception quality instability at the receivers. To avoid this problem, the traffic control scheme moderates the sensitivity of layer add/drop criteria. Bandwidth occupancy ratio of video traffic is correlated with level of the layer, as in layered encoding with increase of level of layer the data rate increases exponentially.

NLM [Kang, 2001] assumes that the source assigns a proper Time-to-Live (TTL) value and Type-of-Service (TOS) bits to each packet of layered video packets. The TTL value is used as a tag to show which layer the flow belongs to and the TOS bit is used to differentiate packets under NLM schemes. A receiver subscribes to as many layers as its link bandwidth permits and the initial membership may last until the end of the flow. The quality of received video is determined by the number of data layers received.

NLM employs a traffic controller including a *filter* and *measurer* in a router. The *filter* resides before the queue and controls the amount of output packets to a corresponding link. It checks if the packet is qualified to be forwarded. The *measurer* measures the average queue length, which is used as the traffic metric. Based on this information, NLM categorizes the degree of current link traffic status into three levels – *unloaded, loaded, and congested.* The decision how the TTL threshold would be changed is based on the predicted status level. A traffic controller requests the neighbouring controller for a given link to change the TTL threshold. The neighbouring controller modifies the corresponding TTL as requested. These steps are repeated periodically. By changing the value of the TTL threshold the traffic controller reduces the traffic by dropping the data of less significant layers.

In the suggested algorithm [Son, 2003], the router collects the bandwidth occupancy ratio of the video traffic over the total traffic and the number of sessions, adding to the traffic state information about average queue length. Using this information the TTL threshold is determined adaptively, thus allowing more fine-grained traffic control.

This approach is aimed to prevent congestion within heterogeneous TCP-connections and improve video layered multicast by increasing stability in reception and filtering less important layers. This goal is achieved by exploiting processing on network nodes where adaptive traffic control is accomplished via router assistance.

3.4.5 Active Interest Filtering for Distributed Simulation

Distributed simulation applications are DVE applications that are attracting increasing attention not least from the simulation community (for

example battleground simulation). There are also strong similarities between these applications and distributed multiuser games.

While distributed simulation infrastructures have evolved dramatically over the past several years to provide ever-increasing levels of flexibility, abstraction, and interoperability, the scalability and performance of the simulation infrastructure continues to be a critical limiting factor. In particular, it is now becoming apparent that the limitations of the supporting networking technologies are a significant obstacle to achieving the necessary levels of scalability and performance.

Active networks can be used to increase the scalability and performance of distributed simulations in numerous ways. For example, by applying the active networking implementations of reliable multicast transport protocols and the interest filtering within the given multicast group. Zabele and Stanzione describe the use of Active Network techniques for distributed simulation [Zabele, 2000]. They describe a project which is developing an active interest filtering capability. They illustrate their approach with a simulation scenario of a fast flying aircraft at low altitude with limited sensor range. If ground vehicles are sparsely distributed over a large region, the aircraft simulation is only interested in small numbers of vehicles at any one time, but large numbers over a period of time. Communication grouping approaches have to balance the disadvantages of a small number of groups (likely to include too many receivers) against a large number of small groups (requiring more frequent maintenance of group membership). They contrast these approaches with dynamic interest filters in the form of agents injected into network routers on behalf of the aircraft simulation. The idea here is that these filters can examine position information in state update messages from ground vehicles and only forward those within a given distance of the aircraft's current position.

Within IP/Multicast communications the only manner in which an entity can control the data it receives is through the joining and leaving of multicast groups. Technically, this is accomplished through the Internet Group Management Protocol (IGMP) and network multicast routing protocols which provide only coarse control over data delivery at relatively long time scales (e.g., 100s of milliseconds to join a group, and tens of seconds to leave a group). Here, active interest filtering offers a means to introduce fine-time-scale, flexible

(programmable) per-entity information filters into the routers' forwarding function that can react to changes in receiver interests nearly instantaneously and selectively prune irrelevant data from the multicast streams. Therefore, even though all subscribers are in fact members of the single multicast group, only those subscribers needing the data actually receive the message.

The approach described by [Zabele, 2000] represents the following key aspects:

- *Coarse-grain multicast grouping* – The entities are divided into a number of groups that can be reasonably supported by the network infrastructure.
- *Embedded interest headers* – Information relevant for interest management (e.g., latitude, longitude, altitude) is already embedded within the state update messages. In this situation, the active interest header is really a virtual entity that imposes no additional processing or message overhead.
- *Multicast group subscription* – During the course of the simulation exercise, subscribers and publishers join and/or leave the coarse-grain multicast groups.
- *Interest filter specification* – During the session, subscribers submit, modify and/or retract fine-grained interest filters to active nodes.
- *Interest filter distribution* – The active routers merge interest filters from downstream entities (subscribers or other active routers) before sending filter specifications upstream.
- *Multicast routing* – Routers that are not also active routers provide standard support for setting up and maintaining multicast routes in response to join and leave requests form subscribers and publishers.
- *Active filtering* – Active routers decide to forward state update messages or not with respect to both an admissible multicast route and the active interest filtering decision.

The Figure 3-4 shows the active filtering example. Upon the receipt of the message, each active router compares the information in the interest header with the interest filter associated with each outbound interface. The message is replicated only onto those interfaces with interest filters that will admit the

message. As such, no irrelevant data is received, and the system behaves as if a larger number of more precisely specified groups are in use.

The authors name the proposed active filtering architecture SANDS [Zabele, 2002]. This architecture is based on the ANTS architecture.

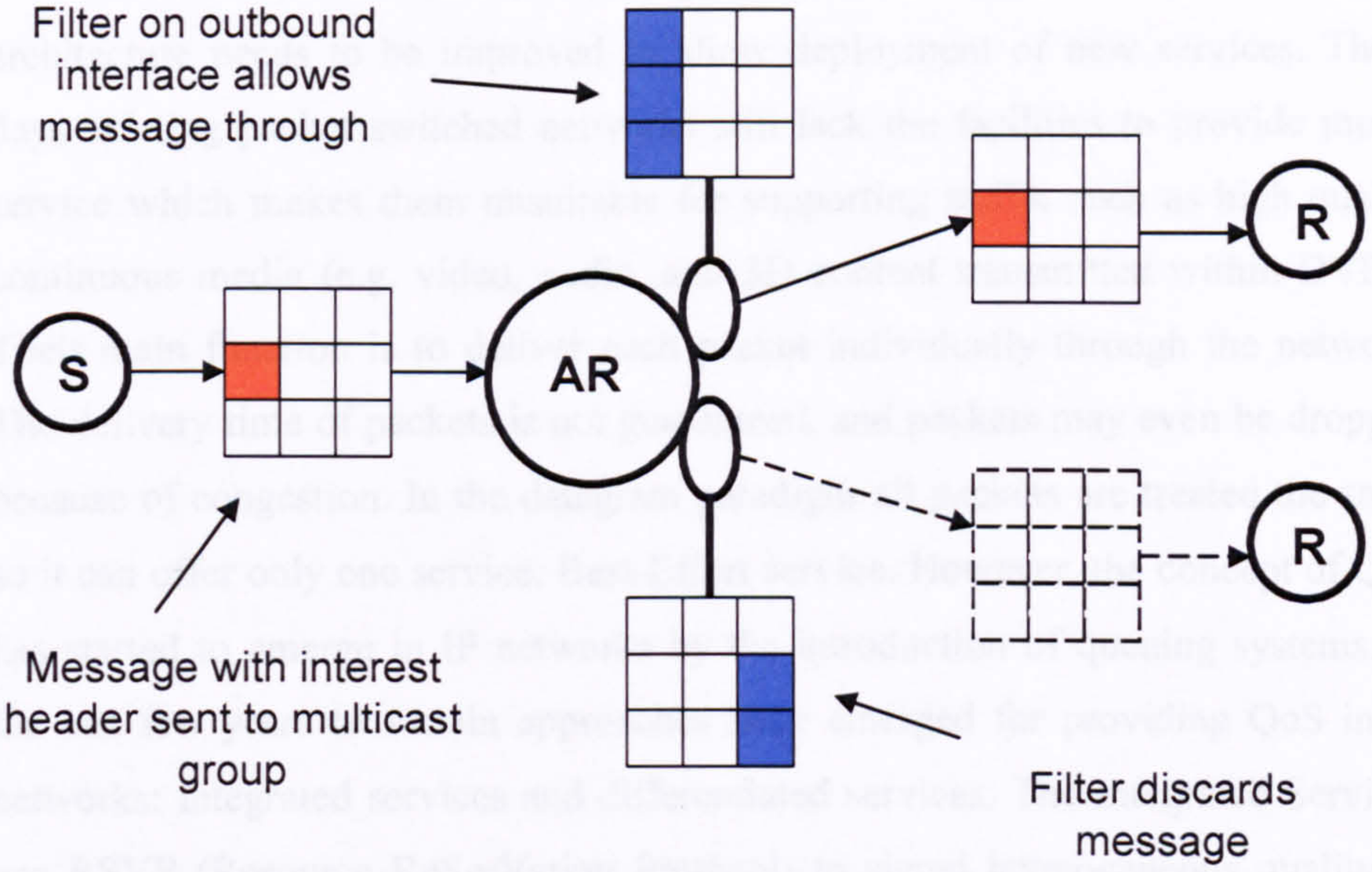


Figure 3-4 Example of active filtering

Use of active networks to provide interest management services offers important benefits to large-scale simulations [Zabele, 2002]:

- An entity can rapidly and dynamically select which data router will forward. This selection can be done at a sub-millisecond timescale, as compared to the multiple-second timescale imposed by the existing multicast group join/leave mechanisms.
- Because each entity can install its own filters, information filtering is accomplished in a "receiver-driven" manner, allowing each entity to customize its filters according to its own need. This decentralized approach allows active filtering to scale well as the number of entities grows large.
- Because active filtering is performed at a routing point, filtering can also be dependent on state (e.g., congestion level) at that router. In particular,

this allows both entities and network routers to determine which data should be discarded in times of congestion.

3.5 Multi-service in Packet Switched Networks

As has been discussed in the beginning of this chapter, the current Internet architecture needs to be improved to allow deployment of new services. These days existing packet switched networks still lack the facilities to provide multi-service which makes them unsuitable for supporting traffic such as high quality continuous media (e.g. video, audio, and 3D content transmitted within DVEs). Their main function is to deliver each packet individually through the network. The delivery time of packets is not guaranteed, and packets may even be dropped because of congestion. In the datagram paradigm all packets are treated the same so it can offer only one service: Best-Effort service. However, the concept of QoS has started to emerge in IP networks by the introduction of queuing systems. In the last few years two main approaches have emerged for providing QoS in IP networks: integrated services and differentiated services. The Integrated Services use RSVP (Resource ReSerVation Protocol) to signal heterogeneous quality of service and maintain QoS. The RSVP protocol is used to establish a resource reservation between a sender and receiver. In order to request a QoS level from the network on behalf of an application data stream a host uses the RSVP protocol. Differentiated Services provide resource assurance through provisioning combined with prioritization. The main issues in these two approaches are resource allocation and performance optimization. The principles of Integrated Services and Differentiated Services are discussed next.

3.5.1 Integrated Services

Integrated Services (IntServ) proposed a new architecture for resource allocation in order to meet the QoS requirements of real-time applications [Wang, 2001]. To receive performance assurance from the network, an application must set up the resource reservation along the path before it can transmit packets. This requires flow-specific state in the routers which represents an important and fundamental change to the current Internet model. Integrated services extend the

service model used by the traditional Internet (Best-Effort service) to include two more services targeted towards real-time traffic: guaranteed service; and predictive (controlled-load) service. The RSVP protocol is used to establish a resource reservation between a sender and a receiver. It is receiver-initiated and it provides a general mechanism for creating and maintaining a reservation state over a multicast or a unicast path. RSVP is a signalling protocol and it runs over IP, both IPv4 and IPv6. RSVP does not perform its own routing, instead it operates with unicast and multicast routing protocols to determine where it should carry reservation requests. Although the Integrated Services architecture proposes new service models, its complex implementation imposes extra load to the network which results in it not being scalable for large networks. To provide a guaranteed service there should be a Resource Reservation Protocol combined with flow control at each node. Furthermore, in order to integrate different classes of traffic, bandwidth-partitioning mechanisms will also be required.

3.5.2 Differentiated Services

The complication of implementing IntServ and the urgent need of an enhanced service model motivated the IETF to propose a simpler and more scalable approach to offer multi-service [Wang, 2001]. The IETF called this new proposal Differentiated Services (DiffServ) [RFC 2474, 98]. The Differentiated Services approach divides the traffic into a small number of classes and allocates resources on a per-class basis. Classes represent aggregated traffic. The DiffServ architecture standardizes a set of per hop behaviours (PHB) as the basic building blocks. End-to-end services can be constructed from these PHBs combined with admission control mechanisms. There are significant differences between the IntServ approach and the DiffServ approach: in DiffServ state information is held for each class of traffic; rather than for individual flows as is the case with IntServ. DiffServ provides resource assurance through provisioning combined with prioritization, rather than per flow reservation. DiffServ defines the way that packets should be treated rather than offering different end-to-end services. In the DiffServ architecture, services are defined in the form of a Service Level Agreement (SLA) between a customer and its service provider. The SLA includes the service parameters for traffic profiles and policing actions such as token

bucket parameters for the class, performance metrics (throughput, delay, and drop priorities) and actions for non-conforming packets. Applications can specify a desired service by marking the Differentiated Services (DS) field of each packet in the IP header with a specific value. The type of service (TOS) field is a single byte present in the IPv4 or IPv6 packet header. Due to the TOS field present in each packet header, each node provides differentiated services on a per hop basis. The value encoded in TOS specifies per hop behaviour (PHB) that defines how the traffic should be treated. One of the proposals to implement DiffServ in the Internet [Nicols, 97] offers three different classes of service: premium, assured, and best-effort. Premium traffic has higher priority than assured traffic. Only non-conforming packets are dropped or delayed until they are within the service profile. Assured traffic has lower drop priority than ordinary best-effort service. The best-effort delivery service is defined by the traditional IP protocol transmission of packets from point to point without any guarantee of specific bandwidth or minimum time delay, but with promises not to drop packets unnecessarily and to deliver them as quickly as possible.

In the DiffServ architecture, the tasks performed by edge routers and core routers are different (Figure 3-5). The edge router tasks include classification of packets, policing, and shaping. They are mapping packets to different classes, checking whether the traffic flows conforming to the SLA, and finally dropping non-conforming packets. On the other hand, a core routers' responsibility is only forwarding packets based on their class encoded in the packet header to determine the treatment of the packets. This division of responsibilities will reduce the complexity of implementing the DiffServ architecture and make them scale over large networks and give them an advantage over IntServ which requires all nodes to perform packet classification to identify packets from reserved flows and schedule them with per-flow queuing.

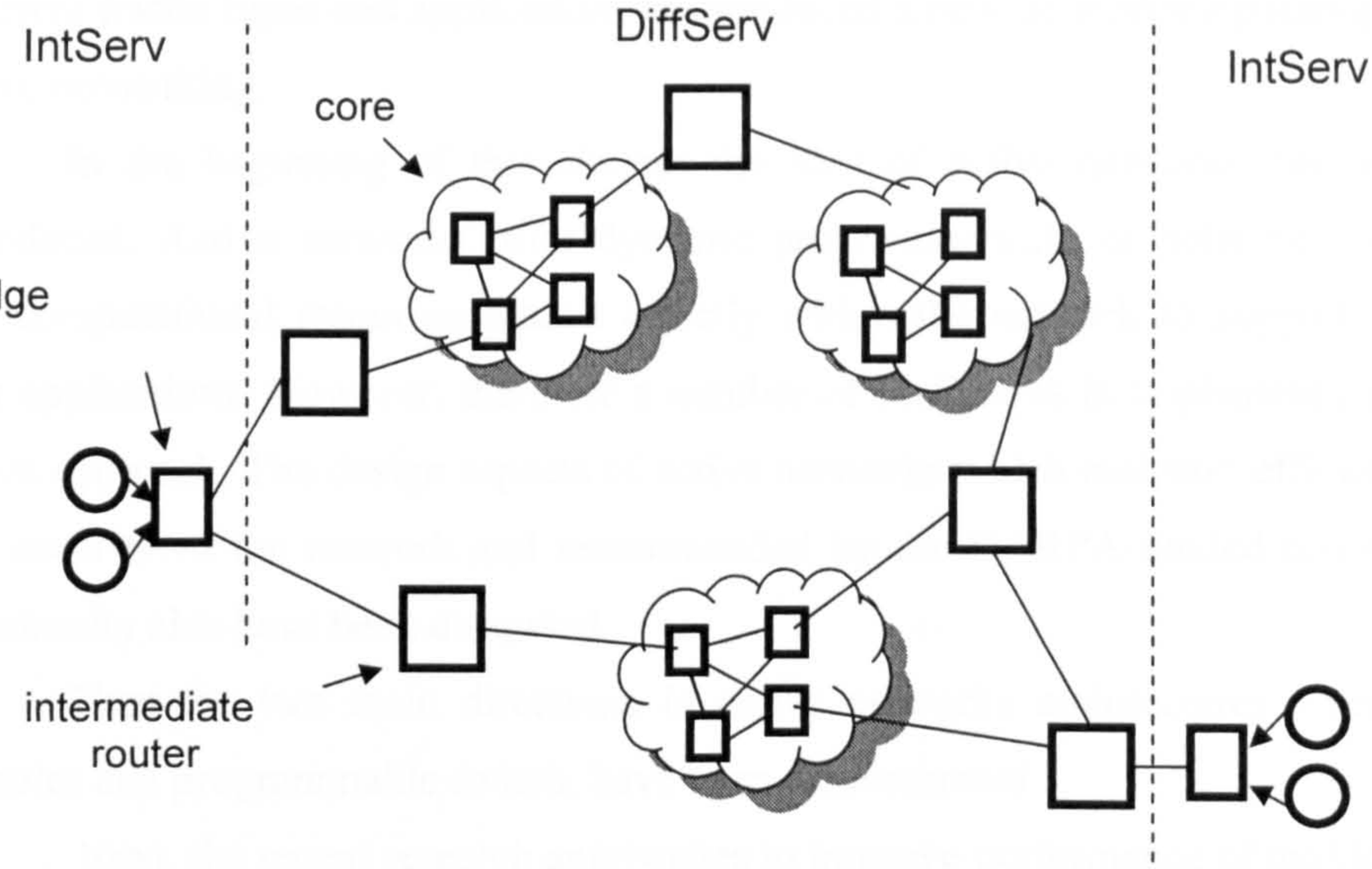


Figure 3-5 Network architecture

3.5.3 Hierarchical Approach

There are also new proposals that incorporate both these approaches into one combined network architecture [Rajan, 98] [Eichler, 2000] [Mahfooz, 2001]. This approach uses a concept of Integrated Services, or similar approach, at the edge of the network, and Differentiated Services in the core of network. It is proposed to provide guaranteed service in multi-service networks using this approach, but with specific modifications. A modified Integrated Services with admission control test architecture is implemented at the edge routers, and Differentiated Services offering the Expedited Forwarding (EF) PHB to Guaranteed Services in the core of the network on intermediate nodes. In other words, different types of traffic will need to be treated individually at the edge of network where the link capacity is limited to medium speeds i.e. 100 Mbps, whilst they can be grouped in one real-time class such as Expedited Forwarding class in the core of network where they are being served by a very high speed link.

3.6 Summary

The recent changes in networked applications have led to the emergence of new networking technologies. The Internet's need for multi-service for

different traffic types and applications has produced a new networking paradigm - active networking.

In the beginning of this chapter the idea of active networks has been introduced. Active networks offer dynamic programmability of network nodes and computational resources placed directly within the network to support end user applications. However, there are a number of challenges in implementing of active approach. The design aspects of active networks which maintain efficiency and security of the network and recommended by the DARPA-funded research community also have been discussed.

Then the two main directions in active networks architectures - active capsules and programmable switch, have been demonstrated.

Next, the recent research approaches to improve performance of multicast applications have been presented. For example, an active reliable multicast (ARM) [Lehman, 98] utilizes intermediate routers to improve scalable reliable multicast (SRM) approach. The heterogeneous quality of service of video multicasting is achieved by active video transcoding [Keller, 2000] and router-assisted traffic control for layered multicast [Son, 2003]. An interest filtering for distributed simulation [Zabele, 2000] exploits active routers to filter unneeded data within one large multicast group.

All these approaches illustrate the advantages of employment of active networking techniques to improve multicasting, as they provide programmable, fine-grained services for end user applications.

The last part of this chapter demonstrates the general solutions which add multi-service to packet-switched networks and provide QoS. The Integrated and Differentiated services, as well as the hierarchical approach that combines both of these services, have been introduced. The active networking aims to add to these general approaches more flexible, dynamically installed, and programmable services. As it has been defined in Section 3.2, active networking is a recent approach whereby intermediate nodes outside the core of the network can be involved in the customized processing of control and data traffic. Therefore the model of classes of traffic that is proposed by Differentiated services might be used to distinguish active data packets on intermediate active nodes. Therefore, the part of the bandwidth resource on routers would be allocated to the traffic

class that contain certain active traffic packets. For example, 3D content data transmitted within DVE applications could be assigned to one of these classes.

The current research examples presented in this chapter suggest to us that there is a potentially useful synergy between the notions of aura and nimbus, the notion of levels of detail in geometric modeling, with the Active Network approaches of Keller *et al.* on video scaling and Zabele and Stanzione on Active Interest filtering.

In the next chapter we discuss the techniques of progressive transmission of 3D data. This discussion completes the background information part of the thesis.

Chapter 4

Multiresolution Representation and Progressive Transmission of 3D Data

The number and complexity of 3D models that must be accessed through the Internet is growing rapidly. Recent technological advances have provided extensive data bases of highly detailed objects. For example, the acquisition of data via satellite, laser scanners or medical image devices are methods that, thanks to their high precision, allow the creation of large three-dimensional (3D) data sets. In turn, there are many applications that take advantage of the availability of this type of object. Fields such as cartography, computer-aided design, computer graphics, computer vision, analysis of finite elements and scientific visualization, apply them to terrain representation, flight simulators, distributed virtual environments, networked games, etc. Normally, these objects are represented by means of polygonal surfaces or meshes. Polygonal meshes are the primary representation for visualizing 3D data and they are central to Internet and broadcasting multimedia standards such as VRML and MPEG-4 [MPEG-4, 99].

The data that describes each polygon is stored into tables that are to be used in the subsequent processing, display, and manipulation of the objects. A convenient organization for storing geometric data is to create three lists: a vertex table, an edge table, and a polygon table. Coordinate values for each vertex in the object are stored in the vertex table. The edge table contains pointers back into the vertex table to identify the vertices for each polygon edge. And the polygon table contains pointers back into the edge table to identify the edges for each polygon. An alternative arrangement is to use just two tables: a vertex table and a polygon table [Hearn, 97]. Nowadays this type of representation is the most frequently used in the field of interactive computer graphics. In VRML and MPEG-4 standards a polygonal mesh is defined by the position of its vertices (geometry); by the association between each face and it's sustaining vertices (connectivity); and optional colours, normals and texture coordinates (properties) [Taubin, 99]. The polygonal meshes are popular due to various reasons such as the high availability of polygonal surface data, the fact that they are used by most visualization software, and that graphic systems accelerate the visualization of polygons. However, the costs of visualization, storage or transmission of these

surfaces increase considerably when they are formed by tens of millions of polygons [Ribelles, 2002]. Common representations of triangulated meshes usually store the triangles as an indexed face list, where the coordinates of each vertex are three floating-point numbers and the incidence relation between triangles and vertices uses three integer vertex references per triangle. Therefore, each triangle requires 12 bytes for the indices and every vertex 12 bytes for the coordinates which adds up to 18 bytes per triangle (there are roughly twice as many triangles as vertices in typical model), not counting colour and texture information. Even when using short-integer incidence indices, an indexed face list for triangles still requires 9 bytes per triangle [Pajarola, 2000]. Therefore, the size of triangle meshes, which now can be measured by millions, or even hundreds of millions of triangles, imposes limits on applications for which complex 3D models must be accessed remotely. However, only very simple models can be accessed over today's common communication links for immediate viewing. This situation is not likely to improve, since the need for more accurate 3D models will outpace the improvements in transmission bandwidth to the end users. For traditional multimedia types (e.g. text, images, audio and video) transcoding is used to minimize the size of transmitted data and therefore decrease the time to receive the information. Image compression and video conversion to different rates and encodings, for example, provide variations of the same object using different modalities. The transcoding paradigm can be naturally extended to 3D data multimedia type. To achieve transcoding of 3D data one could use the technologies such as multiresolution representation with model simplification and level of detail management for improved graphics performance, 3D model compression for reduced storage requirements and faster delivery, streaming technologies for progressive downloading of 3D data.

4.1 Multiresolution Techniques

Multiresolution 3D modelling technologies which were developed over the past decade play an important role in addressing the bandwidth bottleneck within 3D content delivery. Multiresolution modelling [Heckbert, 94] [Turk, 92] has been successfully presented as a solution to the problem of efficient manipulation of highly detailed polygonal surfaces. It consists of representing an object by

means of multiple approximation or levels of detail (LoDs) where each approximation or LoD represents the original object using a different number of polygons. The development of the multiresolution modelling was probably stimulated in the main by the need to achieve the interactive visualization of these objects.

The first multiresolution schemes managed relatively small numbers of LODs and they were developed with the main objective of accelerating the visualization of the scene. Later, multiresolution schemes that manage a continuous range of approximations appeared. They allow the level of detail polygonal surface to be adapted to the application requirements with a high degree of accuracy. The first schemes are called *discrete resolution models* and the second, *continuous multiresolution models*.

Independent of the multiresolution modelling type (discrete or continuous), whenever we need to build a multiresolution representation, a simplification method must be used. A simplification method automatically converts a polygonal surface into another surface formed by a smaller number of polygons. Multiresolution modelling needs simplification methods in order to generate the different approximations by means of their repeated application.

4.1.1 Discrete Multiresolution Modelling

The simplest way to create a multiresolution representation is to generate a set of independent approximations referred to as Level-of-Detail (LOD) models, where each one represents the original object with a different level of detail. The human eye does not always distinguish the fine details of a mesh, especially when the mesh is displayed far back in the view frustum. This fact has been utilised to reduce the amount of computing power required to render a mesh. If the mesh is far back in the view frustum, a lower resolution approximation can be substituted, since the detail of the mesh will not be apparent that far back. It is very easy to implement because complex data structures and algorithms are not needed to store and retrieve the LODs. In this approach a fixed number of simpler meshes approximating the original mesh is precalculated and each mesh is stored independently in a list. Each mesh in this list contains fewer vertices than the one

preceding it. Each mesh is also given a "*range of detail*" value which is used as a filter by the renderer in order to select the appropriate LOD model in the list. Therefore, discrete multiresolution representations are computationally cheap and they allow visualization of larger data sets without overloading the CPU. This technique is used in VRML.

Unfortunately, LOD models have several drawbacks. Since each LOD model is stored independently, the number of models in the list is restricted by the amount of memory that is available. Although theoretically the maximum number of LODs could be high, the cost of storing each LOD independently makes this number small in practice (between 5 and 10 normally). Because only a small number of LODs is stored, the algorithm that selects the most appropriate LOD for each frame is often forced to choose an approximation of more detail or one of less detail than the one required. This causes the problem in the transition between different approximations that is called "*popping*". When one approximation is being visualized and the selection criterion decides to visualize another one with a different level of detail, a jump in quality between the two representations can be perceived.

The other drawback of discrete multiresolutuion is the need to transmit all LODs. This creates the data duplications in transmitted 3D geometrical sets and therefore increases delays to view and interact with objects. As in today's desktop computers the resource of transmission bandwidth is more restricted than CPU power more complicated geometric representations, that utilize more processing power have emerged.

4.1.2 Continuous Multiresolution Modelling

Continuous multiresolution modelling has come about to solve the problems presented by discrete multiresolution modelling. A continuous multiresolution provides a wide range of different approximations that represent the original object. The iterative application of a simplification method, starting from the original data set, M^{n-1}, produces a sequence of approximations M^{n-1},... ,M^2 ,M^1, M^o, where M^o is the approximation with the least detail. A continuous multiresolution representation encodes the original data set. One scheme of

continuous multiresolution modelling schemes is the incremental representation. Incremental representations were mainly developed for application to progressive transmission, compression and interactive visualization using approximations of uniform level of detail. Most of the schemes have been proposed in recent years. These schemes encode the set of approximations by using the coarsest level of detail or the base mesh, M^o, and a linear sequence of refinement operators. These operators reconstruct each approximation when applied in the order of the sequence. Continuous multiresolution representations are acceptable as today's CPUs are able to process incoming data faster than it is received over the Internet. The receiver can display the base mesh as soon as it is received. The refinements to the base mesh can be applied and displayed as they arrive. A continuous multiresolution model of the object can provide a high number of meshes (almost infinite) representing a single mesh at different resolutions. This representation of objects shows many advantages:

- It supports efficient query processing in terms of being able to extract a mesh from the model at a given resolution in real time.
- The size of the model is not significantly larger than the size of the mesh at the highest resolution.
- The transition between each LOD is very fine therefore reducing popping considerably.
- It allows progressive transmission of the mesh.

One of these schemes is a progressive mesh representation described by Hoppe [Hope, 96]. Hoppe introduces the progressive mesh (PM) representation, a scheme for storing and transmitting arbitrary triangle meshes. This representation addresses several practical problems in graphics: smooth geomorphing of level-of-detail approximations, progressive transmission, mesh compression and selective refinement.

Within PM representation the arbitrary mesh M is stored as a much coarser mesh M^o together with a sequence of n detail records that indicate how to incrementally refine M^o exactly back into the original mesh $\hat{M} = M^n$. Each of these records stores the information associated with a *vertex split*, an elementary mesh transformation that adds the additional vertex to the mesh. The PM representation

of $\hat{M}$ thus defines a continuous sequence of meshes M^0, M^1, ..., M^n of increasing accuracy, from which LOD approximations of any desired complexity can be efficiently retrieved.

The simplification procedure for constructing a PM representation from a given mesh $\hat{M}$ seeks to preserve not just the geometry of the mesh surface, but more importantly its overall appearance.

Almost all of the polygonal meshes in computer graphics are triangle meshes. Geometrically, a triangle mesh is a piecewise linear surface consisting of triangular faces attached together along the edges. The mesh geometry can be denoted by a tuple (K,V),where K is a *simplicial complex* specifying the connectivity of the mesh simplices (the adjacency of vertices, edges, and faces), and $V=\{\mathbf{v}_1, \ldots, \mathbf{v}_m\}$ is the set of vertex positions defining the shape of the mesh in 3D space [Hoppe, 96].

The transformation operation that allows to simplify meshes in PM is edge collapse, which unifies two adjacent vertices. One of these vertices and two adjacent faces vanish in this process. Thus an initial mesh $\hat{M}=M^n$ can be simplified into a coarser mesh M^0 by applying a sequence of n successive edge collapse transformations:

$$(\hat{M}=M^n) \xrightarrow{ecol_{n-1}} \ldots \xrightarrow{ecol_1} M^1 \xrightarrow{ecol_0} M^0$$

The particular sequence of edge collapse transformations must be chosen carefully, since it determines the quality of approximating meshes.

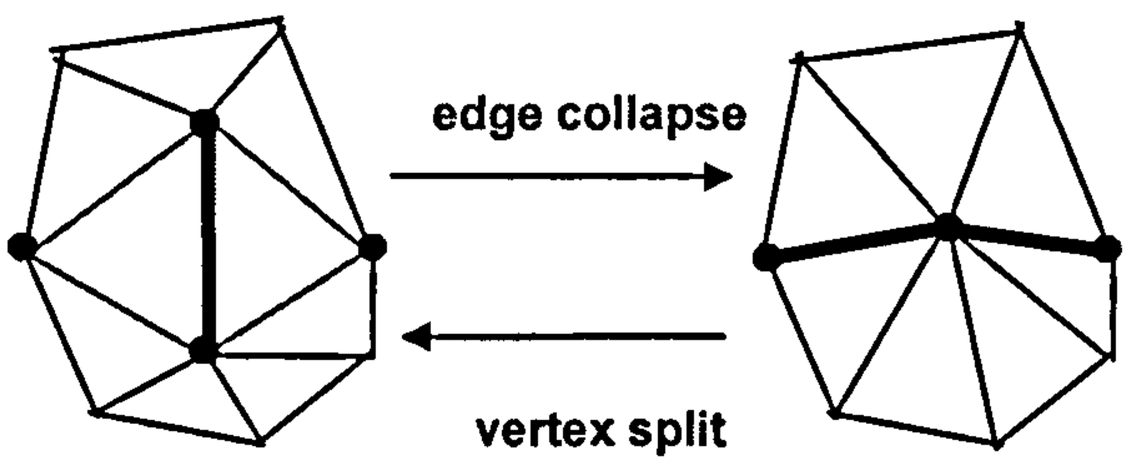


Figure 4-1 PM simplification and reconstruction for triangle mesh

The edge collapse transformation has an inverse operation – a *vertex split*. A vertex split adds a new vertex and two new faces and updates the attributes of the mesh in the neighbourhood of the transformation. It is possible to represent an

arbitrary triangle mesh $\hat{M}$ as a simple mesh M^o with a sequence of *n vsplit* records:

$$M^o \xrightarrow{vsplit_0} M^1 \xrightarrow{vsplit_1} \ldots \xrightarrow{vsplit_{n-1}} (M^n = \hat{M})$$

Each vertex split is specified for the current level of detail by identifying two edges and a shared vertex. The mesh is refined by cutting it through the pair of edges, splitting the common vertex into two vertices and creating a quadrilateral hole which is filled with two triangles sharing the edge connecting the two new triangles.

The coarse model M^o and a sequence of vertex split records (M^o, {$vsplit_0$, ..., $vsplit_{n-1}$}) define a PM representation of the original mesh M^n. Progressive meshes are a natural representation for progressive transmission.

4.2 Progressive Transmission

The progressive transmission of polygonal meshes allows the decoder process to make the low resolution levels of detail of a multiresolution mesh available to the rendering system before the whole bitstream is fully received and decoded [Taubin, 99]. The PM scheme introduced by Hoppe was the first method to address progressive transmission. In this scheme the compact mesh M^o is transmitted first (using a conventional uni-resolution format), followed by the stream of the *vsplit* records.The receiving process incrementally rebuilds $\hat{M}$ as the records arrive. The original mesh $\hat{M}$ is recovered exactly after all *n* records are received, since PM is a lossless representation.

By this technique, the amount of data is not reduced but the ordering of the information parts is optimised such that crucial shape information is sent first and less important detail information is then transmitted in the order of decreasing relevance. Since the user immediately can see and interact with approximation of the final 3D object, the waiting time is not perceived disturbing. Longer waiting time yields better visual quality and if the geometric details are not required, the transmission can be terminated at any time [Bischoff, 2002].

To decrease the amount of transmitted 3D data and therefore the waiting time it is desirable to combine progressive transmission with geometry compression.

Geometry compression aims to minimize the number of bits needed to store one triangle [Deering, 95]. Compression within progressive transmission could be achieved by means that the stored information in each record that add detail to base mesh, affects a very localized region of the surface, so that affected elements can be encoded more efficiently. But progressive transmission of multiresolutuion polygonal mesh data with many levels and high compression ratios are difficult to achieve at the same time. There are methods to progressively transmit meshes with as many levels of detail as vertices [Hoppe, 96] with low compression efficiency. Also these fine-grain compression techniques that refine the model by inserting one vertex at a time, may offer increasingly better accuracy early on, but require significantly longer to reproduce the full resolution model and, thus, tie up the communication channel for longer then needed.

An alternative compression methods are the methods that use a vertex clustering algorithm with compression ratios comparable to the best single resolution schemes [Taubin, 99], but not progressively. To achieve better compression ratios within progressive transmission the algorithms that improve Hoppe's fine granularity approach by adding grouping refinement records have emerged. In the Progressive Forest Split (PFS) scheme introduced by Taubin et al [Taubin, 98] the *forest split* refinement operation can be seen as a grouping of several consecutive edge split operations into a set instead of a sequence and provides a trade-off between compression ratios and granularity. Furthermore, such kinds of algorithms construct limited number of mesh approximations that naturally incorporate the level of detail notion. When rendering an object it is not necessary to have the fine-grained control of an object's triangle count. If the total triangle size is in the millions, small changes in triangle numbers are completely unrecognisable. When we are building levels of detail, the number of triangles we want to discard is usually proportional to the number of triangles in the mesh. If it is required to create LOD for a 5,000-triangle object, it makes sense to build a reduced mesh at 2,500 triangles, but building a mesh at 4,900 triangles would be imperceptible by the user [Blow, 2002]. This feature is exploited, for example, by the Enbaya company [Enbaya, 2002].

All these developed techniques to support progressive transmission of 3D objects can be compared by using approximation error as a function of time

PAGE/PAGES
EXCLUDED
UNDER
INSTRUCTION
FROM
UNIVERSITY

progressive technique (blue curve) immediately starts improving the crude model, but it takes much longer (*d* bits) than a nonprogressive technique to download the entire model. The red line illustrates batched updates which take slightly longer to download the full model, in time *c*, than a nonprogressive approach, but much less time than the fine-grain PM methods [Pajarola, 2000].

The goal of the CPM method is to compress progressive representation and therefore decrease the volume of transmitted data and the time of transmission. It is possible to say that the CPM approach divides the family of PM meshes into several levels of detail. After applying all refinement operations from one batch to the mesh a more detailed LOD mesh will be reconstructed.

Although the accuracy remains constant while the next batch of upgrades is received and decoded, the overall waiting time and bandwidth utilization are reduced because batched upgrades may be compressed more efficiently than individual upgrades exploiting the economy of scale. Instead of encoding the ids of all the edges that will have to be split, the authors encode one bit for each vertex of the current mesh. A zero means leave it unchanged. A one means split it. The authors suggest to traverse the mesh and to specify split-vertices by marking vertices with one bit instead of indexing them explicitly, as is done in PM.

In order to reduce the total cost of marking the split-vertices, the authors seek to maximize the ratio of split-vertices at each batch. At the same time, there must be a clear separation of the different cut-edges.

The CPM format orders the information parts of mutiresolution model in the following way. The crude model, M_0, is stored using a single resolution compressed format. M_0 usually contains 5 to 10 percent of the number of triangles of the entire model. The other part of the CPM format contains the sequence of refinement batches. These may be downloaded systematically, or only on request, and create the sequence of increasingly precise approximations $M_1, M_2, \ldots, M_{max}$. For instance, if the model's screen representation remains small, only M_0 may be needed.

Each batch $M_{l+1} \longrightarrow M_l$ includes:

- split-vertex marking bits in M_l;

- cut-edges encoding for every split-vertex;
- the entropy encoded prediction correction vectors of the split-vertex.

The simplification stops at a crude model M_0, when a given error threshold or mesh complexity is reached.

In each simplification batch $M_{l+1} \longrightarrow M_l$, the number of triangles $\mid M_{l+1} \mid$ is decreased by $3 \cdot \tau_e \mid M_{l+1} \mid$. The ratio τ_e denotes the fraction of edges of M_{l+1} that can be collapsed in the same batch (typically 11 percent).

To optimize coding, CPM attempts to maximize the selection ratio τ_e of collapsed edges for a simplification batch $M_{l+1} \longrightarrow M_l$. However, there are restrictions for collapsing edges in M_{l+1}, which must be fulfilled to avoid singularities in the simplified mesh M_l.

1. At most two vertices may be collapsed into one.
2. Two edges that forming a quadrilateral cannot be collapsed in the same batch.

To achieve a good approximation at each stage, the CPM method uses an error metric to evaluate the error introduced by every single edge collapse.

The CPM method selects a subset of the less expensive edges that do not violate restrictions for collapsing edges.

To reduce the number of coordinates that define a vertex-split, all selected edges are always collapsed to their midpoint.

The connectivity coding is needed to restore M_{l+1} from M_l. The connectivity coding is achieved by construction and traversing a vertex-spanning tree of M_l and marking split-vertices (i.e., the results of an edge collapse in M_{l+1}). Then for every marked split-vertex v, its cut-edges are encoded as follows:

1. Compute the indices of the two cut-edges in the sorted list of the incident edges of v, clockwise, starting with the edge from v to its parent in the vertex spanning tree.
2. Given the degree d of the split-vertex in mesh M_l, the two edge numbers being identified as one possible choice out of $\binom{d}{2}$ for selecting the cut-edges.

To complete the compression, Pajarola *et al* encode the geometry coordinates of the original vertices of the collapsed edges. The basic idea is to predict an unknown vector from known vertices and to encode the prediction error.

The decompression process uses the same prediction and reconstructs the correct vector by adding the decoded correction.

The CPM representation combines progressive transmission of compressed 3D data with more coarse-grain partition of refinement operations which offers several LODs. These features make the CPM scheme useful for the large 3D models transmission within large-scale DVE applications. While the LOD structure would allow progressively higher detail to be transmitted as a participant is approaching the object, compressed representation of 3D data saves bandwidth and permits use of more complex and realistic representations within DVE applications.

In this dissertation the CPM method is used as a prototype for the traffic flow model which will be described in chapter 7.

4.4 Summary

This chapter discusses multiresolution representation and progressive transmission of 3D data.

Firstly the conventional 3D representation has been outlined. Then the storage and transmission problems caused by the recent enormous increase in the size of the 3D objects have been shown and the multiresolution techniques have been introduced as the promising solution to these problems.

The discrete and the continuous multiresolution models have been defined. The continuous models have been found more desirable as they allow progressive transmission of the 3D data. One of the multiresolution continuous models, Hoppe's progressive meshes (PM) has been described.

After that progressive transmission has been considered. The different approaches to progressive transmission have been shown including fine-grain (e.g. PM) approach and the schemes that use grouping of refinements (PFS). Approximation error as a function of time [Rossignac, 2000] has been included to compare different progressive transmission schemes.

The last part of this chapter describes Compressed Progressive Meshes model that uses grouping of the refinement operations and improves fine-grained PM approach. The CPM representation is used in this work as a prototype of 3D content traffic model.

Chapter 5

Methodology and Framework for Evaluation

The aim of this work is an investigation into how Active Network (AN) ideas might benefit large-scale Distributed-Virtual-Environments (DVEs). The main suggestion that is proposed in this research is using an active network for peer-to-peer architecture to improve scalability issues of the DVE application. The peer-to-peer approach currently used by several DVE systems (e.g. DIVE) seems more promising for future large-scale systems. But, along with advantages of this current approach (e.g. there is no such problems as a bottle neck and single point of failure that are represented by server within an alternative client-server architecture), some scalability limitations would arise. These restrictions would be caused by such multicast limitations as an address space limitation and frequent time-consuming joining and leaving processes (see section 2.2.4.1) within the small-size groups or a fine-grained grouping. At the same time, the large groups within a coarse-grained grouping approach might limit the scale of a DVE application by a huge bandwidth utilization that is needed to deliver data flows of numerous types (video, audio, 3D streams and text) to all members of the one large multicast group. However, not all of data flows that are generated by all users of a large multicast group are fully required by the members of the group. Spatial partitioning techniques (see section 2.2.2) allow filtering of unneeded data flows within one multicast group. In this work 3D content data flows generated by DVE users are considered as a special traffic class that is serviced on active routers. The approach, proposed in this dissertation, adds more precise flow control by incorporating an application level filtering to the network multicast which is based on spatial partitioning of the virtual world into subspaces (grid partitioning), exploiting the level of detail (LOD) concept (see section 2.2.3), and progressive transmission of 3D data (see section 4.3). This method aims to gain from the active networks' ability to take away from the application the need to understand the network topology and delegate the execution of certain actions, for

example intelligent message pruning, to the network itself. Therefore, the filtering of unneeded 3D flows is proposed to be accomplished by the intermediate active nodes.

Figure 5-1 illustrates a proposed active network architecture that aims to improve scalability issues of the large-scale DVE application by restriction of unnecessary 3D data flows. This architecture includes two parts: an active host component and an active node component. The active host keeps application level information which reflects placements of the participants within subspaces of the virtual world and is used to update the similar active network (AN) node's application level information. On the active host AN application level information is used to control the generation of the 3D progressive streams. On the active router this information helps to perform active application level filtering (shown in Figure 5-1 as Application filter) in addition to the network level multicast filtering (Multicast filter) of the 3D DVE data flows.

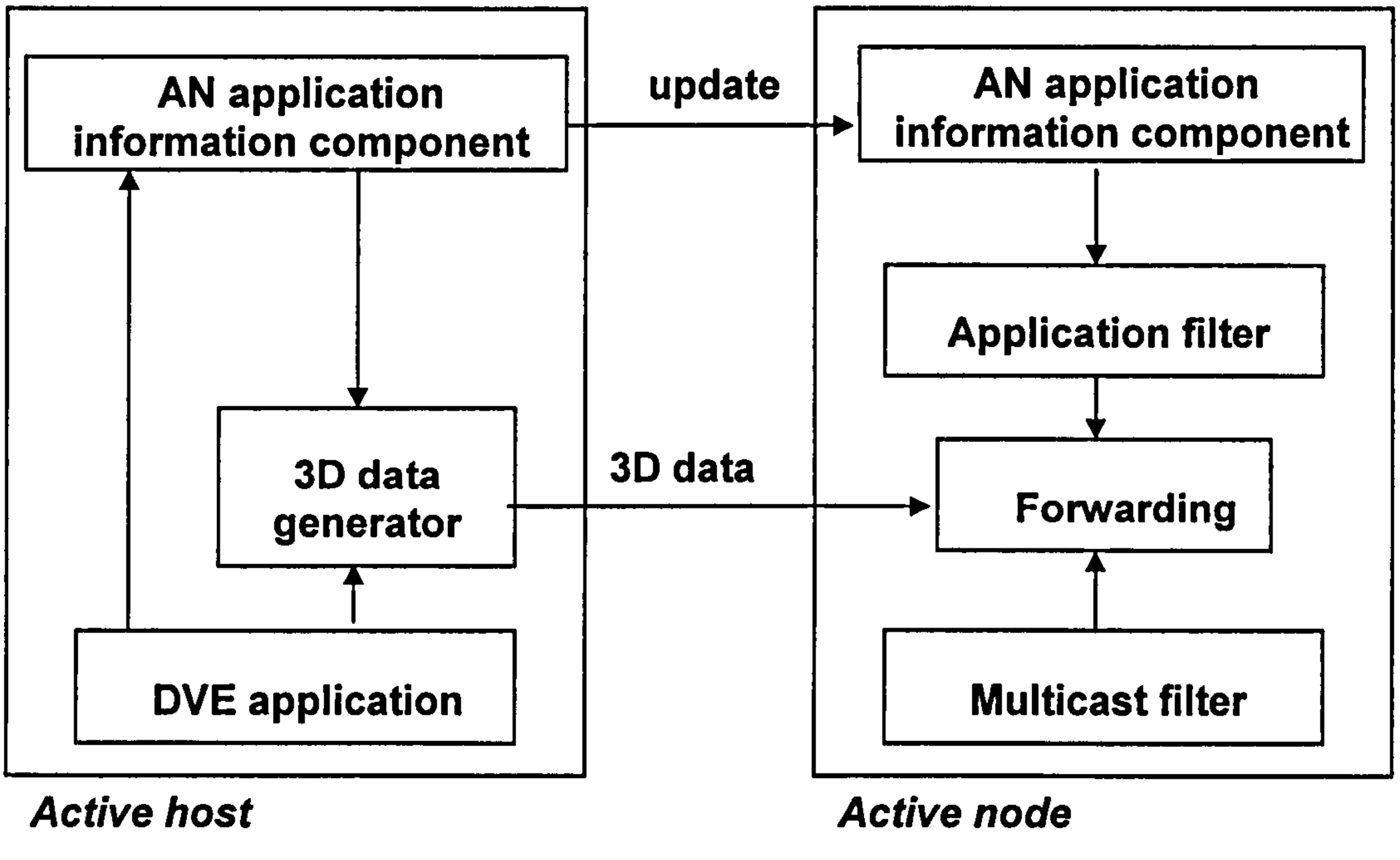


Figure 5-1 Proposed active network architecture for DVE applications

Next the methodology that is used in his work to explore and to evaluate an active network's filtering of 3D content flows within large-scale DVE applications is discussed.

5.1 Methodology

The aim of this work is to evaluate the active networking architecture to support 3D data transmission within DVE applications. The methodology applied in this dissertation introduces the performance modelling approach. It includes modelling of an active network's and host's architecture and modelling of a DVE application. Simulation is used to model DVE users' interactions and active processing on the routers and the hosts. To evaluate the performance of the proposed active approach a series of simulation experiments has to be performed.

Complex interactions between participants in DVE application and a large number of analyzed variables indicate that flexible simulation is more appropriate for proposed architecture's evaluation than mathematical modelling. To build a test bed will not be feasible due to its complexity and long implementation time. The reason for choosing a simulation is that a simulation is generally more flexible than analytical methods. In our work we need to look at the DVE participants' interaction within large number of scenarios. To carry out the evaluation of the proposed approach the OPNET simulation package has been chosen. The OPNET simulation provides a comprehensive development environment supporting the modelling of communication networks and distributed systems. Both the behaviour and performance of modelled systems can be analyzed by performing discrete event simulations [OPNET].

5.2 Framework for Evaluation

The framework for evaluation consists of the architecture modelling, flow and application modelling, defining performance metrics, design of simulation

experiments and simulation-based evaluation of effectiveness of proposed approach. These parts of the evaluation framework are described below.

1) *Architecture modelling.*

The modelling of the novel proposed active networking architecture for supporting 3D transmission within a large-scale DVE application includes two aspects: modelling of active router's filtering mechanisms and modelling of active host processes that support active nodes filtering. These two models present not a real architecture but arguments of the possible architectural design.

The active host controls the generation of the 3D progressive streams that is driven by the application information about the changes of users' placements within the partitioned virtual world. Also the active host provides an active node with the similar application level information about the spatial placements of the users.

On the active router this information is used to perform active application level filtering in addition to the network level multicast filtering of the 3D progressive flows. The active router models an active network node that also has multiservice capability. In this way DVE traffic can be allocated its own class and assigned its own portion of the bandwidth.

2) Flow model.

The modelling of the 3D content flow includes the construction of progressive mesh representations of the 3D object that present one flow of 3D data traffic generated by the DVE application. To construct the progressive mesh model the mesh simplification algorithm which uses the CPM algorithm [Pajarola, 2000] as a prototype needed to be developed.

Within an active approach the flow intensity depends on spatial relations of users in the virtual world (less 3D data would be delivered to the receivers that are placed far from the sender in the virtual world). Each user in the multicast group has its multicast tree to distribute information to other members. For conventional multicast the load on the certain network link depends on overlapping of these multicast trees. For the active approach the spatial placements of users would

control the intensity of flows along with conventional multicast routing trees and produce the aggregation of traffic flows.

3) Application model.

To define an application model which would be used in the evaluation process the generic type of the DVE application has to be considered. In the DVE application users move around a 3D world. The 3D objects in the DVE application are the users' representations and the virtual world representation. The participants in this application interact with each other and with the virtual world. These interactions create the patterns of users' movement. The intensity of users' interaction within the DVE application identifies the intensity of data that is required to be transmitted between users' hosts over the application's runtime. In order to evaluate the effectiveness of the active networking approach several application models that model a range of user interaction and the resulting 3D information traffic have been constructed.

In these application models the only objects that exist in the virtual environment are the representations of the users and the virtual world is subdivided by the two-dimensional grid for simplicity. We adopt the rectilinear grid for the group-per-region allocation of the multicast groups (see section 2.2.2) therefore the part of the virtual world that is allocated to one multicast group is represented by a square. Hence, the grid that subdivides this square is also divided by the rectilinear grid. We choose this rectilinear method for the first stage of our research; however other methods (e.g. hexagonal grid) could be used in the further work. Users can be placed either randomly or selectively, on squares within this grid and then at discrete intervals in time they either remain in position or move to any adjacent square according to some rules and random factors, the parameters of which can be adjusted to simulate various categories of user interaction. The set of initial placements together with the set of movements that take place over a predefined interval of time will generate a particular scenario that can be used to drive the generation of the 3D data flows and updating the spatial positioning application level information on the active routers.

The movements of users across subspaces of the grid would achieve the changes of distances between users and therefore of the required level of fidelity for representations of other users or LODs. These movement events invoke an additional transmission of more detailed 3D data flow in the case when a sender is approaching a receiver. Therefore to create the application model it is needed to model a sequence of movements of users in the virtual world.

Large-scale DVEs, that are based on geographical partitioning assume that users are somewhat uniformly dispersed in the virtual world. This idea leads to the assumption that by reducing the area that is "seen" by each participant the number of users in one multicast group and therefore the amount of received and processed information at each host would be reduced. If, however, most (or all) users are packed together in a small area of a DVE, the number of objects a given user must deal with may still be too large. This diversity of the users' movement patterns in the virtual world is caused by changes of their areas of interest (AoI) over time. To be able to examine a range of 3D traffic patterns within DVE applications in our work we model several scenarios of users' movements in the grid that divides one region of the virtual world into subspaces. These different patterns of movement define several application models that were used in the evaluation process. In our application's model all users are members of one multicast group and "see" all representations of the others, but the level of detail differs depending on the distance to the viewed user. Transmission of different batches of progressive 3D representations that correspond to different levels of detail would generate a diverse load of the network over time within different scenarios of movement.

A methodology that includes the study of different DVE application scenarios can be found, for example, in the Coven Project [Tromp, 98]. This project investigates group behaviour issues, sense of presence and social interactions between participants within different application scenarios. Another piece of work that uses different movement scenarios [Morillo, 2003] describes evaluation of an adaptive balancing technique that improves the partitioning problem for DVEs with multiple servers' architecture. The evaluation of the

proposed techniques is carried out using different movement patterns of avatars and measurement of CPU utilization on servers.

In this work several application models that are defined by different movement patterns of users are used for both non-active and active approaches to be able to compare routers' and hosts' performance in the active and non-active cases. The measurement of network node and host characteristics is used in the evaluation process.

4) Simulation and performance metrics.

In order to investigate the feasibility and performance of the proposed active approach simulation modelling was used. Intuitive principles of investigation were used in simulation. Simulation was employed as a random process and the analysis of different scenarios or application models was carried out. Non-active and active approaches were simulated for all scenarios. For the active approach simulation models for different progressive 3D representations (3LOD and 9LOD representations) were built. The simulation model consists of two basic elements, hosts and routers. The host element simulates a host architectural model. The router element simulates a node architectural model. The network models that simulate the DVE applications of various scale and configuration were constructed with using these two basic elements. To build simulation models the simulation package OPNET was used. The comparative performance of active and non-active approaches was evaluated by studying the simulation results.

The performance of the DVE application is defined by the resource utilization of a DVE which is directly related to the amount of information that must be sent and received by each host and how quickly that information must be delivered by the network [Singhal, 99]. The proposed active filtering mechanisms decrease the amount of information that is sent and received by each DVE host by restricting unnecessary highly detailed flows of 3D data. To evaluate the consequence of decreasing the amount of 3D traffic a resource trade-off between the processing, storage, and bandwidth was considered, as the decrease of utilization of any of these resources reduces the cost of the DVE system.

The main goal of applying the active filtering approach is improving DVE scalability. This goal would be achieved by resource management that reduces the cost of increases in the scale of application (see section 2.2.1). As a scalable system is one for which the cost of increases in scale is regarded as "small" or "acceptable" [Greenhalgh, 98], to achieve the scalability the cost of increases in scale should be decreased.

Node performance was analyzed as a trade-off between the available bandwidth for 3D DVE data traffic class, the router storage space that is required for queuing this traffic class's packets, and processing on the router that executes active filtering mechanism. As the 3D DVE traffic is allocated its own class and assigned its own portion of the link bandwidth, it is assumed that the router bandwidth or service rate for 3D DVE data traffic class is constant. Hence the major analysis was made for queue occupancy results or storage requirements. The methodology of this research also assumes that the additional processing resource on the router is utilized by the active filtering and this active processing is not measured.

The evaluation of the proposed active approach integrates several performance metrics or Quality of Service (QoS) parameters that characterize queue occupancy on a router:

1) Storage requirements on the active router:
    - Maximum length of the queue.
2) Utilization parameters:
    - Total time within the experiment time when the queue is empty.
    - Percentage of time when the queue is empty.

To explore more precisely the effect of applying the active filtering approach on DVE application performance it is also required considering an active host's performance.

The host performance was analyzed in several aspects. First, the waiting time to receive the 3D representations of other participants was examined. The reception waiting time defines the time to see and start to interact with a 3D object

and it is a crucial quality metric of a DVE application. Minimizing this parameter gains the better sense of presence for the user. Otherwise, longer waiting times are disturbing and might lead the user to leave the environment.

Second, a trade-off between the storage, processing and bandwidth was analyzed for the host. The storage resource on the host is a memory requirement to store representations of other users during the runtime of the application. A decrease in the host memory requirements to store 3D representations improves the scalability of DVE applications, as it allows the user to store simultaneously the representations of large numbers of users. It is assumed that the user's link bandwidth is constant. The methodology of this research also assumes that the active processing on the host includes two parts, where the first part is the processing that implements active host component of the active networks architecture and the second part is the processing that is required to reconstruct the mesh to the higher LOD (to add data from the one batch). It is assumed that transmission bandwidth and storage are more restricted resources within remote access to the highly detailed 3D objects (see Section 4.1) so processing on the host is not measured.

Therefore the host's performance metrics or Quality of Service (QoS) parameters are defined as follows:

1) Initial reception time of other users' representations (waiting time to start interaction).
2) Memory requirements to store representations of other users during the runtime of the application.

5) Design of experiments.

The main objective of the experiments is to evaluate the novel active networking architecture approach through comparison to the currently used non-active approach. Hence, all sets of experiments were designed for two methods – active and non-active architectures. For the non-active approach as a flow model the original representation of 3D objects was used. For the active approach a progressive mesh representation of the 3D object would modell the flow of 3D data.

Two stages of the application runtime were considered. The first stage is the initialization stage when initial transmission of the 3D representations is accomplished. The objectives of the initialization stage tests are: 1) providing confidence in our modelling methodology; 2) analysis of host's initial reception time performance metric; 3) observation of preliminary queue occupancy results for routers.

The second stage of the application is the post initialization or dynamic stage that reflects interactions of users. The objective of this stage tests the evaluation of router's and host's performance for the active approach within the life time of the DVE application.

To evaluate the active approach's performance in different scenarios, experiments were designed with changing experimental parameters. By changing one of the parameters a new series of tests was carried out. Every set of experiments was designed for both the active and non-active approaches. The list of experimental parameters is presented below.

- The number of levels of detail (LODs) that is used in progressive meshes representation of the 3D object (1 LOD for non-active approach).
- Users' movement patterns in the virtual world.
- Duration of the experiment.
- The available bandwidth on routers.
- Number of users.

5.3 Summary

This chapter outlines proposed in this work an active networking approach for supporting 3D transmission within a large-scale DVE application and describe the frame work for the evaluation process of the proposed approach.

First, the idea of exploiting active filtering for 3D transmission within a DVE application is explained and the architecture presented by two components - an active host and an active node is illustrated.

Next, we discuss the methodology used in this work.

Finally, the framework that is used in this work to evaluate the proposed architecture is described. The evaluation process consists of several elements: the architecture modelling, flow and application modelling, defining performance metrics, design of simulation experiments and simulation-based evaluation of effectiveness of proposed approach. All these parts of the evaluation framework were introduced in this chapter.

The next chapters contain a detailed description of the evaluation framework elements. Chapter 6 presents the architecture model. Chapter 7 provides the flow and application models' construction. The simulation modelling and the simulation experiments' results are included in chapter 8 and chapter 9.

Chapter 6

Active Networking Architecture Model

An active networking architecture proposed in this work aims to improve scalability issues of the large-scale DVE application by restriction of unnecessary 3D data flows. In this architecture the novel idea of improving the QoS for the 3D DVE geometrical data type by exploiting the level of detail (LOD) concept and active networks is suggested. This architecture includes two parts: an active host component and an active node component.

6.1 Architecture Model

The novel active networking architecture for large-scale DVEs follows the discrete approach or programmable switch (see Section 3.4) and is based on three layers: an active router, active extensions, and active packets. Active processing on an active node is executed like the active extensions in the SwitchWare architecture [Alexander, 98]. Active extensions or execution environments (by DARPA classification) are loaded into the active nodes and provide extended services to the data streams passing through the node. On active programmable routers the processing of messages may be architecturally separated from injecting programs into the node. Users would first inject their custom processing routines into the required routers. Then they would send their packets through such "programmable" nodes. When a packet arrives at a node, its header is examined and the appropriate program is dispatched to operate to its content [Tennenhouse, 97]. Network programs, which are loaded into the router, contain the "built-in" primitives available at each node. The services or common primitives built-in to each node might include several categories of operations: primitives that allow the packet itself to be manipulated, e.g., by changing its header, payload, length, etc.; primitives that provide access to the node's environment, e.g., the node address; and primitives for controlling packet flow, such as forwarding, copying, discarding [Tennenhouse, 97]. The proposed active architecture deploys active extensions that control packets' flow by forwarding, copying, and discarding 3D DVE data class packets. These 3D content packets are active packets and include active headers. Active headers are set by active hosts and carry the information

that is used by programmable routers to control the flows of active 3D DVE data packets. Active hosts in the proposed active networking architecture along with tagging 3D packets with active headers also generate the signalling control active packets that update application level information on active routers that is used by active filtering extensions.

The models of the active node and active host architectures are described below. Section 6.2 introduces active router architecture model with a brief description of the active extensions. In Section 6.3 the active host architecture model is described.

6.2 Active Node Architecture Model

An active network consists of a set of active nodes that are connected by a variety of network technologies [DARPA, 98]. Each active node runs an operating system, responsible for managing the node's resources, and one or more execution environments. Each execution environment implements a virtual machine that processes the delivered active packets. An execution environment may provide general computational services or simply a forwarding engine whose computation is controlled by the packet data.

The conventional network communication is usually expressed in terms of the layered abstractions of the OSI Reference Model [ISO, 84]. The network layer (layer 3) provides support for transmission of packet data between end systems. According to the OSI model, ordinary network routers are explicitly constrained to layer 3 processing as a transport-layer (or higher layers) provides end-to-end services which are by definition the domain of end systems. Unfortunately, the implications for active networks would be that active computations inside the network are restricted to data concerning layer 3 (i.e., the IP header) and below [Schmid, 2002].

This idea led to the assumption that network nodes need to be very simple and very fast. However, these presumptions have changed due to the continuous increase of computational power and the advent of active and programmable network technologies (for example, specialised network processors). Now there are many examples where the applications use network functionality that show

that active computation of transport layer or even payload information "inside the network" can significantly improve data communication between end-systems. As an example of this category of applications the DVE application would be considered.

The current focus of our investigation into active networks' abilities to facilitate large scale DVEs is to address the possibility of active filtering using the level of detail (LOD) concept. This restriction of unnecessary 3D data flows aims to improve the scalability of DVE application. Progressive mesh (PM) 3D streams are considered as sender-receiver flows of packets. Every stream contains several batches of packets. These batches correspond to different levels of detail in the progressive 3D representation of the objects. The active nodes that we propose in our work control these 3D content packet flows by forwarding, copying, and discarding packets. The application level data such as spatial positions of users in the virtual world is used to invoke active filtering processes on an active router.

6.2.1 Active Node Architecture Model

The active router architecture model is shown in Figure 6-1. There are two main parts: execution environments (EE) and active node OS. While the active node OS manages and controls access to node local resources and supports forwarding of packets, the execution environments execute active code instructions. As it was shown in chapter 5 (Figure 5-1) the node component of the proposed architecture includes the AN application information element and the Application filter, that are both incorporated in the Execution Environment shown in Figure 6-1. The Multicast filter element (Figure 5-1) is represented by the Routing table (Figure 6-1). The Forwarding element in Figure 5-1 is represented by the Active Forwarding Engine (AFE) in Figure 6-1. The several AFE are shown to explain the possibility of adding some other active processing on the router except the proposed active filtering of the 3D DVE data.

Upon receipt of a data packet, a packet *classifier* determines whether or not packets passing the node need to be processed by a certain active execution environment. Active filtering of 3D DVE data packets is an active process carried out on the active node by one of the execution environments.

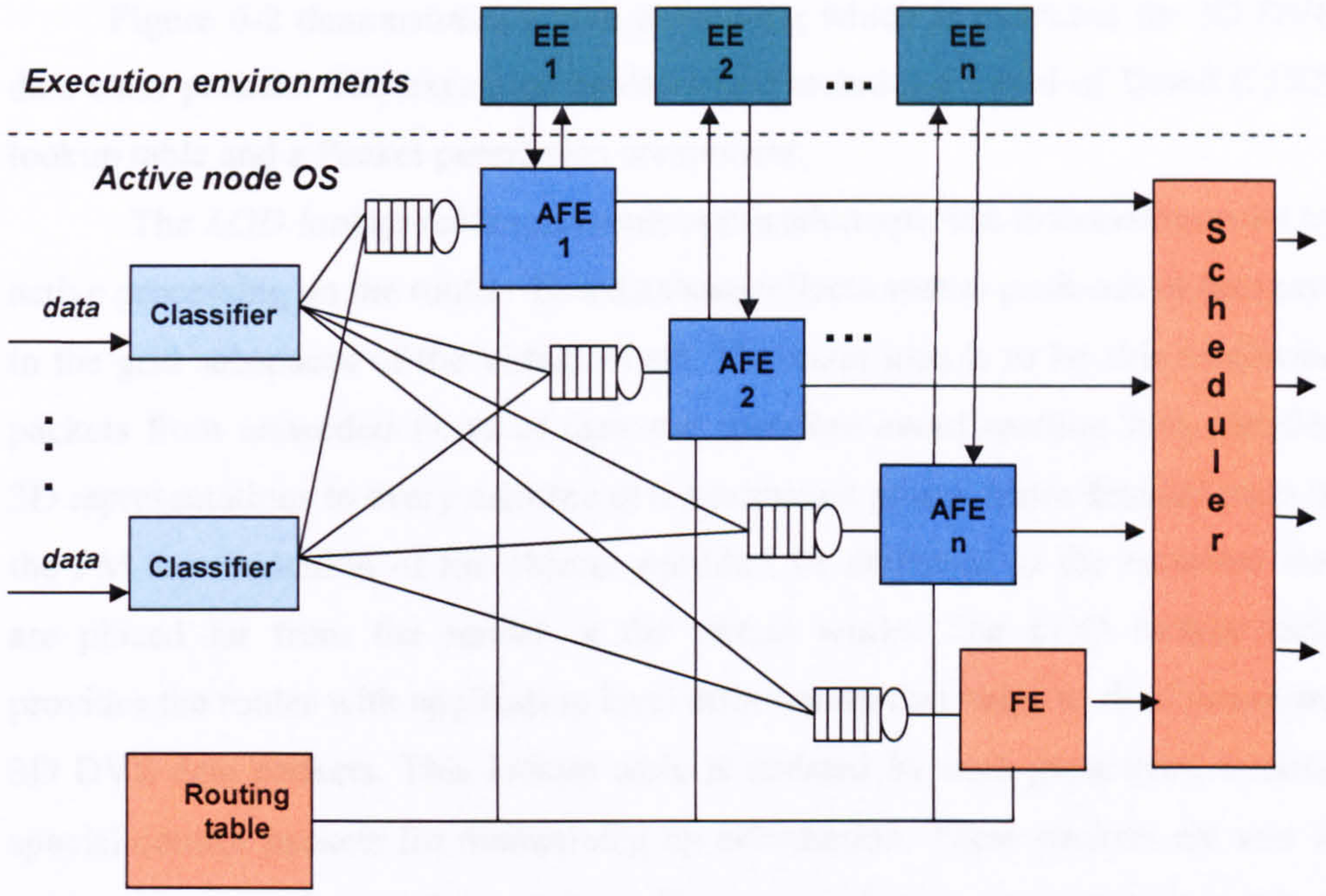


Figure 6-1 Active node architecture model

We use one queue for all input links because we expect very bursty traffic and we look to this kind of architecture to have a large storage space for this traffic. Later in further work we plan to separate queues and carry on with analyzing an individual output links' queues in more detail.

After the completion of active processing the active forwarding engine (AFE) forwards or drops the packet. Non-active packets are forwarded by the conventional forwarding engine (FE) using only routing table information.

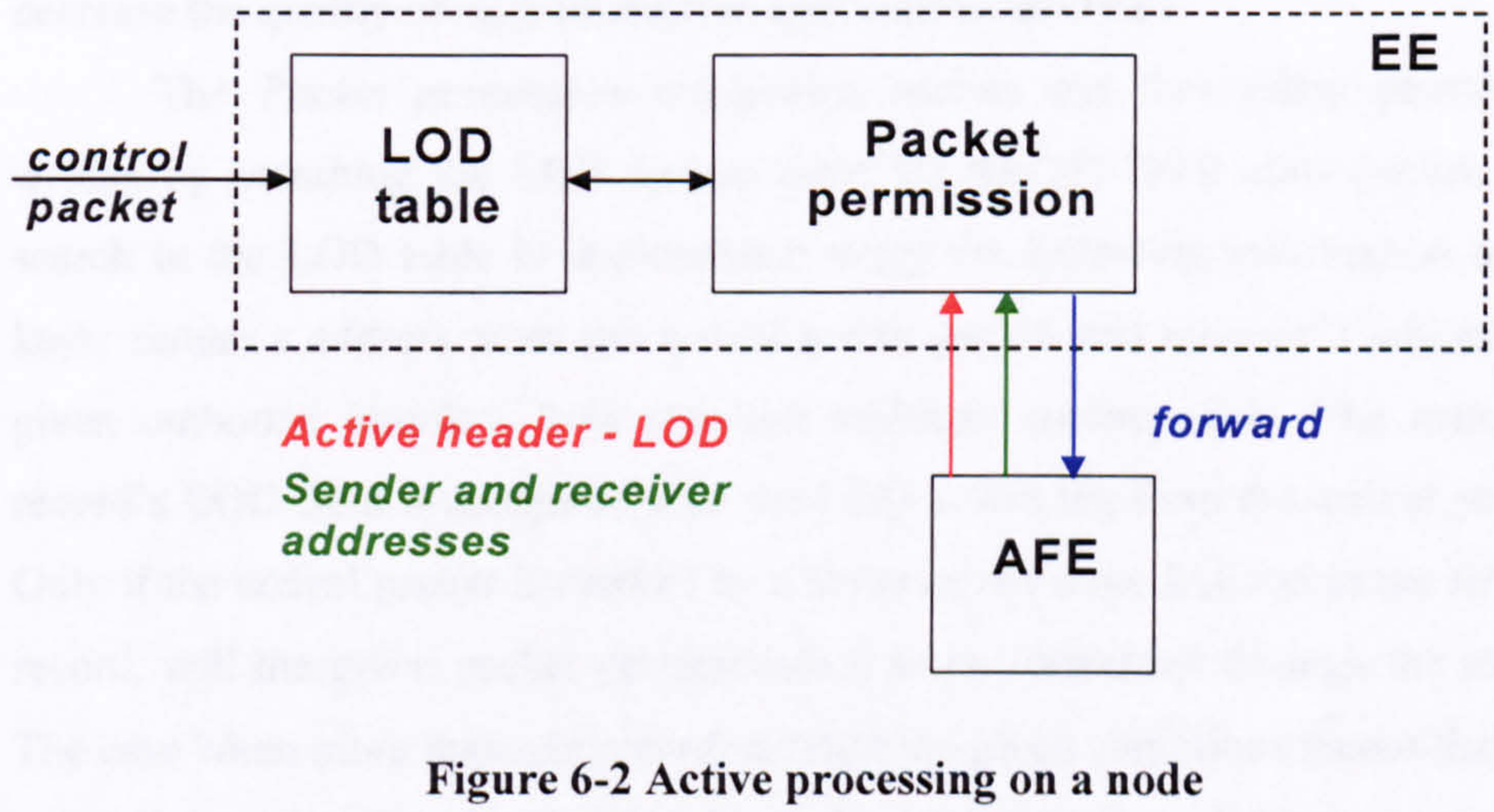


Figure 6-2 Active processing on a node

Figure 6-2 demonstrates active processing which is executed for 3D DVE data class packets. The execution environment includes a Level of Detail (LOD) lookup table and a Packet permission component.

The *LOD lookup table* represents application-specific information used by active processing on the router. This database reflects spatial positions of the users in the grid subspaces of the virtual world. The main idea is to be able to discard packets from unneeded flows of data and therefore avoid sending fully detailed 3D representations to every member of the multicast group. More detailed parts of the PM representation of the objects wouldn't be delivered to the receivers that are placed far from the sender in the virtual world. The LOD lookup table provides the router with application level information that helps to filter unneeded 3D DVE data packets. This lookup table is updated by each participant sending special control packets for maintaining its information. These packets are sent to routers in case of a user's movement from one subspace or part of the virtual world to another. Each control packet contains the new mutual distance between the user (receiver) and one of the other users (senders). A user's movement event causes the user to send several control packets to reflect changes in the distances to all members of the multicast group. In comparison with state update messages that reflect every slight change in the object's coordinates, this information would be sent relatively rarely and wouldn't overload the network. Also the interpreting of every state update message for all users would possibly cause delays and decrease the quality of such interactive applications as DVEs.

The *Packet permission* component carries out forwarding permission checks by searching the LOD lookup table for the 3D DVE class packet. The search in the LOD table is implemented using the following information as the keys: sender's address from the arrival active packet and receiver's address for given outbound interface from standard multicast routing table. The matching record's LOD field is compared with the LOD active tag from the arrival packet. Only if the arrival packet is marked by a lower or the same LOD as in the fetched record, will the given packet get permission to be forwarded through the router. The case when more than one record matches the given conditions means that this router is not the last in the branch of the routing tree and there are several receivers within the given multicast group linked to this outbound interface. To reflect this case a list of the required LOD of the sender for all receivers of this

branch will be extracted from the lookup table. If the packet is marked by a lower or the same level of detail as the highest level of detail in the extracted LOD list it receives forwarding permission.

The forwarding path for data packets is supported by the active node OS. After the packet identification by the classifier the active packets that belong to progressive representation flows are forwarded by the active forwarding engine (AFE). The AFE takes into consideration the packet permission condition (*forward* condition) (Figure 6-2) and the multicast routing decision for this packet.

The packet *scheduler* manages the forwarding of different packet streams using a set of queues. The packets are forwarded to different output lines. The packet scheduler must be implemented at the point where packets are queued; this is the output driver of a typical operating system, and corresponds to the link layer protocol [RFC, 94].

According to the DARPA active node architecture [DARPA, 98] it is possible to have multiple execution environments or active data processes in one node. The active router architecture described here reflects active processing for filtering 3D graphics progressive streams and includes components that are used to complete this one special task.

6.2.2 Packet Formats and Filtering Mechanism

3D DVE data class packets are presented as active packets. These packets encapsulate the progressive mesh (PM) representation of the object. This representation consists of several batches of geometrical 3D data (see Section 4.4). Each batch represents data that is required to render the object with a certain level of detail. These multiresolution representations of one object with decreasing resolution (LODs) are used to reduce rendering cost for distant or otherwise less important objects. As only one level of detail of a given object can be displayed at any one time instead of transmission of geometry data for an object as a whole, it is possible to transmit exactly packets with certain LOD to certain receivers. The PM partition of the 3D stream into batches is used to tag the packets of every batch with a special active header that would invoke active filtering of 3D DVE data class packets on an active router. The packet format of an active geometrical packet is shown in Figure 6-3.

Sender Address	Multicast Group Address	LOD	Geometrical Payload

Figure 6-3 Geometrical data packet format

The LOD header defines the level of detail or batch number in the sequence of batches of PM representation to which the packet belongs. This LOD header would be analysed by the router as an active header. The geometrical packet format used in the proposed architecture model also contains sender address, multicast group address (as several multicast groups might exist for one DVE application), and geometrical payload that includes 3D PM data. The active extension on the router (or executive environment EE) uses information carried by the active LOD header to filter flows of 3D data. The main part of the EE is the LOD lookup table that reflects the spatial placements of users in the VE. Each record of the LOD table contains: receiver and sender addresses, LOD for each sender from the point of view for every receiver (the mutual distance between two participants measured in the required level of detail), and multicast group address. Figure 6-4 shows some records as an example of the lookup table.

Sender Address	Receiver Address	LOD	Multicast Group Address
A	B	1	M1
A	C	1	M1
A	D	0	M1
B	A	1	M1
B	C	2	M1

Figure 6-4 LOD Table

The LOD table example shown in Figure 6-4 illustrates spatial placements of users within a two-dimensional grid. The virtual world is divided into a 3x3 grid (See Figure 6-5), therefore the number of possible levels of detail is three. The highest LOD=2 means that users are placed in the same subspace of the grid and require fully detailed representations of each other to be received. LOD=0 is only the crude model of the object (See Section 4.4) and needs to be received if users are placed within the farthest subspaces of the grid (for example users A and D).

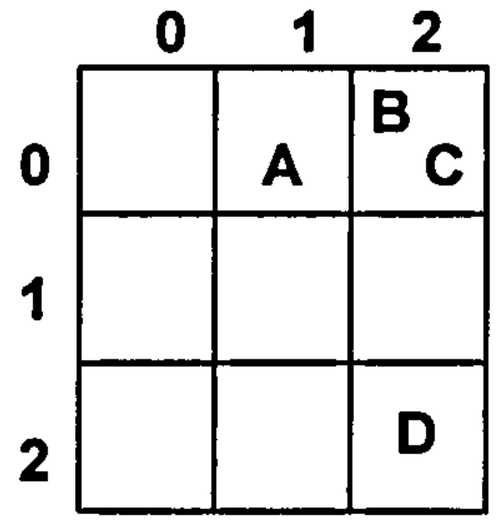


Figure 6-5 Users' placements in two-dimensional grid

The size of this lookup table will depend on the number of members for one of the multicast groups served by the router for a particular application for a given moment and the number of such multicast groups.

In the generic DVE application the 3D objects are the virtual world description and graphical representations of users. With applying the LOD ideas to transmission of the virtual world description, each part of the virtual world will be considered as a graphic scene with a similar level of detail requirement and all objects which form this part of the world will gain the similar LOD. Consequently every part of the VW would have its own address as a source of data. So, in our architecture model there would be no distinction between transmission of users' representations and parts of the VW in terms of construction of our LOD table. Therefore all packets that carry information about the group of objects which represents one part of the VW will be handled by the filtering mechanism similar to packets with geometrical representations of the users.

Creation of the LOD lookup table will be performed on the router at an initialization stage of the application. The initial placements of users are reflected in LOD table at this time. Within the post initialization stage of the application users dynamically change their positions in the virtual world and these events update the LOD table. To accomplish this task users send special signalling control packets. These control packets are also active packets as they have an active header - control tag (see Figure 6-6). The control packet contains information that is used to update the LOD table which is: sender and receiver addresses, and new required LOD.

Multicast Group Address	Control Tag	Sender Address	Receiver Address	LOD

Figure 6-6 Control packet format

The arrival of an active control packet will invoke the LOD table maintenance operation. To maintain the LOD lookup table the following procedures are applied:

1. Search in the table for an entry matching (M,S,R), where M – multicast group address, S – sender address, R – receiver address;
2. If found, replace the old LOD with the new one from the control packet;
3. When a new user joins the multicast group all records for this new user will be added to the LOD table, which will define distances to all members for this group;
4. When a user leaves the group all related records will be deleted.

The packet permission component of the execution environment performs an active filtering mechanism. This mechanism accomplishes the spatial interest filtering of 3D DVE data class packets and in addition to the standard multicast routing decisions determines whether a packet should be forwarded to a given outbound interface (Figure 6-2). Therefore the active filtering performed on active routers enhances the control of the intensity of flows that is achieved by multicast routing.

To demonstrate the control of the intensity of flows that is carried out by active routers, the example of one multicast group is considered. The multicast group which contains receivers (R1-R5), active routers (AR1-AR5) and a sender (S) is shown in Figure 6-7. Active routers will apply filtering to the packets containing geometrical data. For example, the active router AR1 will not forward geometrical packets that contain data with the finest level of detail (LOD=2) to the link connected to the active router AR2 and therefore to the receivers R1 and R3 because they require only representations with LOD=0 for receiver R3 and LOD=1 for receiver R1. However, the packets with full representation would be forwarded to another interface of the active router AR1which is connected to the active router AR3, as one of the receivers (R4) linked to this multicast routing tree branch requires a precise image (LOD=2) of the sender.

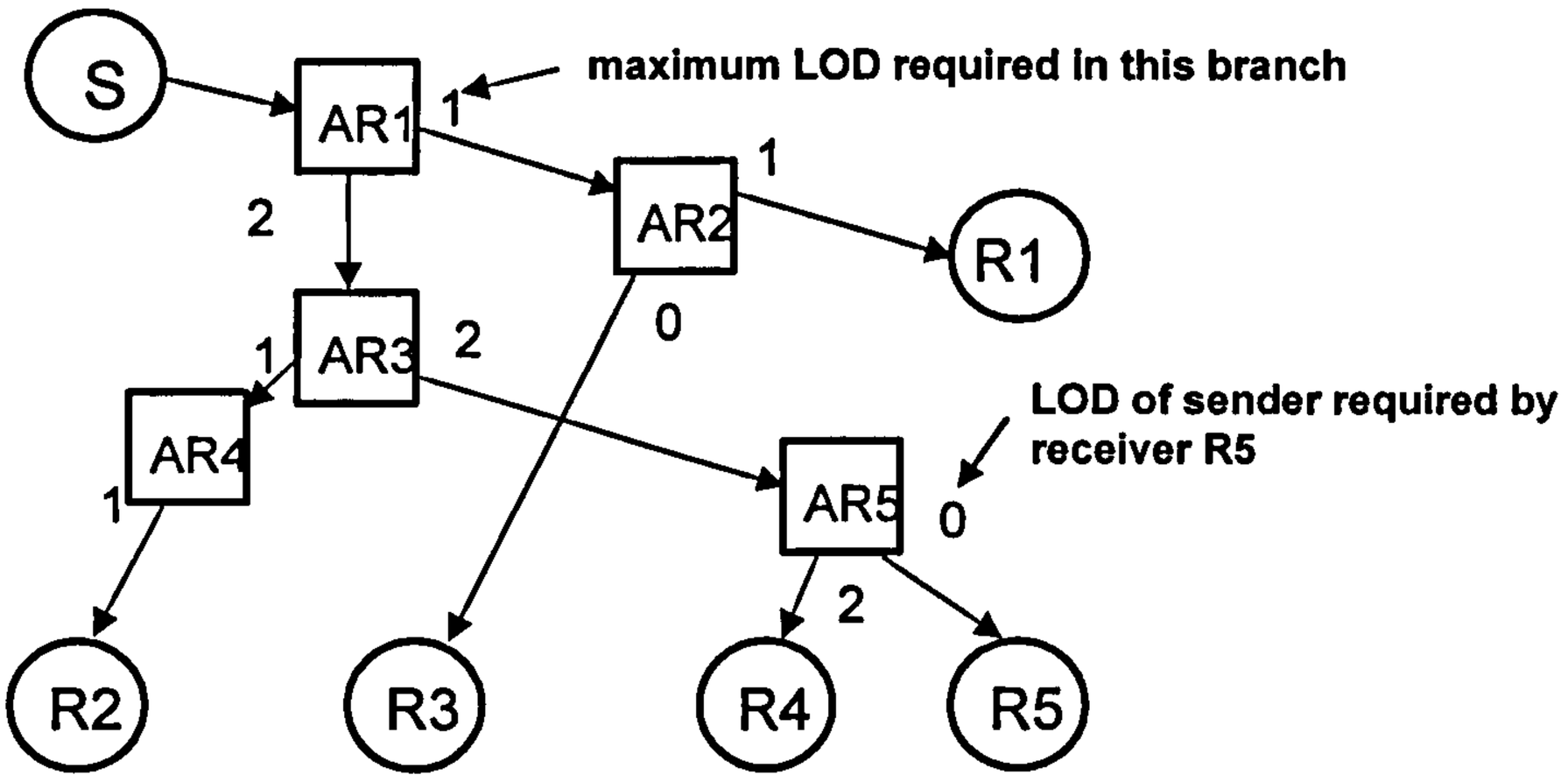


Figure 6-7 Filtering within one multicast group

As has been shown active filtering of 3D DVE packets will control the intensity of flows along with conventional multicast routing trees and produces the aggregation of traffic flows within multicast transmission of 3D DVE data.

6.3Active Host Architecture Model

DVE applications will be persistent applications that are always on. It will attract more users to participate in DVEs as they will be able to join the environment at any moment. Especially for large-scale DVEs there will be a large number of users that exist within the application over time. Our active architecture manages active filtering of the 3D data between users of one multicast group for whole time when the application is on. This task will be achieved by deploying active processing on DVE hosts. In this section the model of the host's active architecture is described.

6.3.1 3D Data Transmission within the Post Initialization Stage of DVE Application

The movements of users change the distances between them over time. As distances change the fidelity requirements to view other users also change. How will these events of movements be reflected by the application? We assume that for large-scale applications there is the need to relinquish part of the 3D data not

actually needed to be able to store representations of large number of participants and the description of the larger virtual world. The storage space defines an ability to design large continuous places with more realistic graphics that users will explore without distorting the sense of immersion. In current DVEs the sense of immersion could be distorted by downloading new parts of the virtual world. So, saving of storage resources on hosts seems necessary as there is a tendency that along with a growth in the number of users, the users' and VW representations become more realistic and require more memory to store.

As a user goes farther from an object (e.g. the representation of another user) the more detailed LODs of the object's representation are relinquished from the user's storage space, hence as it approaches it will need to receive them again. As there are large numbers of users in the environment it is more likely that not only one but many of the users can move toward the sender and therefore enter the subspace that is nearer to the sender within one time slot. It is better to use multicast to deliver the required LODs to the subgroup of the users that are getting closer to the sender subspaces. In the virtual world both the sender and receiver have to see changes in the representations of each other with changes in the distance between them. To take this fact into consideration it is necessary that also one user that moves closer to the subgroup of users has to transmit its highest LOD to all of them. The use of multicast in this case will save bandwidth. But the multicast could lead to redundancy in transmission, as not all the members of the multicast group need this more detailed 3D data. The example of user's movement in the virtual world that is shown in Figure 6-8 and the changes in mutual users' distances for two successive time slots that are presented in Table 6-1 provide an example of users' interaction within a DVE application. As shown in Table 6-1, the distance between user A and user B is not changed after movement of user A. Applying the active filters on the routers will allow selection from all members of the multicast group just the subgroup of users who become closer to the sender after its movement.

Table 6-1 Movement of user A

Time 1		Time 2	
Users	**LOD**	**Users**	**LOD**
A-B	0	**A-B**	0
A-C	0	**A-C**	1
A-D	0	**A-D**	1

Table 6-2 Transmitted LODs

user	sent	received
A	LOD1	LOD1(C) LOD1(D)
B	none	none
C	LOD1	LOD1(A)
D	LOD1	LOD1(A)

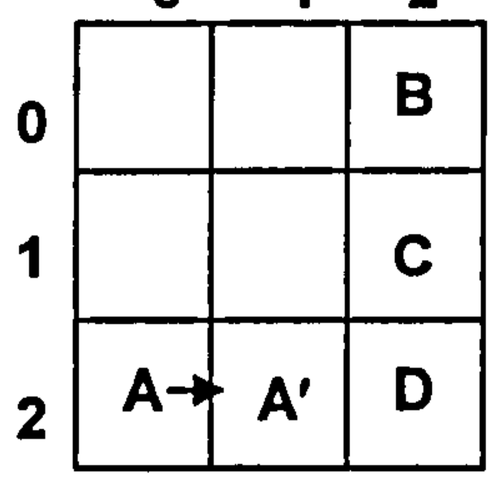


Figure 6-8 User A moves in the virtual world

With changing the position of user A in our example user A transmits LOD1 of its representation. Table 6-2 shows all sent and received LODs by users within the transmission caused by the movement of user A. Why does A send LOD1 only? There is no need to multicast the whole sequence of batches starting from LOD0 as it has been done within initial transmission of the 3D representation. As users move farther from each other, we relinquish one (highest of presently stored) LOD for each subspace while the distance between them increases. Therefore if one user moves nearer to the another user by one subspace it is only needed to receive the next highest batch again (or batch that follows the highest presently stored). As shown in Table 6-2, only users C and D receive LOD1 of user A. User B doesn't receive LOD1 of the user A as the distance between A and B doesn't decrease. User D multicasts LOD1 of its representation in response to change in the distance D-A. User B doesn't send any information as there is nothing changed in the mutual distances between user B and others. So, a user sends more detailed data if some of the mutual distances between it and the receivers decreases (cases when others getting closer or sender moves towards them).

The application of every user chooses which LOD needs to be multicast at every time slot. Therefore every application has to keep the same LOD lookup table as the routers do. This host's LOD lookup table would have the same

structure as the routers' and be updated by active control packets. The decrease in mutual distances will be reflected in the application's LOD table. The application periodically checks the LOD table. If some decrease in mutual distances is detected it invokes multicast transmission of higher LODs to the multicast group. In cases when new users join the multicast group every host transmits the sequence of LODs which includes a crude model and several higher LODs up to the highest required by the group. This transmission would allow newcomers to render representations of users in the group. The application multicasts the highest required by the group LOD, but not all members of the multicast group are approaching the sender within the given time slot. Therefore the active routers will filter these higher LODs and restrict the reception of this data only to the subgroup of receivers that approach the sender. To complete this task within the post initialization or runtime stage of the application the routers' LOD lookup tables are used.

How frequently would the check of the application LOD table be performed? It could be chosen by taking into consideration the application's specific characteristics. If the users are walking within very large subspaces (kilometres) (slow movement), application checks for the need to retransmit 3D data relatively rarely. If the application consists of users with high velocities (planes) the check could be carried out more frequently. For environments where the velocity of users differs radically the application could for example use the difference between the required LOD for two successive check time slots and multicast the sequence of LODs (from lower to higher) as users moving at high speed could cross several subspaces within the period of time between two verifications of the LOD table.

The speed of users' movement defines the intensity of the 3D DVE traffic. For example, in the case of high speed movement more frequent changes of the grid's subspaces require the check of users' placements and therefore the retransmission of the higher LODs more frequently. However, the size of the grid's subspaces is also dependent on the speed of users' movement and for applications with fast moving objects the size of the subspaces could be increased. It will increase the time that the user spends to cross one subspace and therefore will allow to keep the traffic intensity at the level where it does not overload a network. Therefore the granularity of the time intervals of the checks of the LOD

table as well as the size of the grid's subspace has to be chosen to reflect the speed of the participants' movement and the size of the virtual world.

6.3.2 Host Architecture Model

The active host architecture model is presented in Figure 6-9. This model incorporates previously discussed application events. This model includes the LOD lookup table that is similar to the router's lookup table and mirrors the same application information. This table is updated by the arrival of control packets from all hosts. The control packets generator sends control packets to the active routers that support this application and to the active participants' hosts to update the LOD lookup tables when a user moves. Information about the user's movement is supplied by the host's DVE application. The next element of the host's architecture model is the geometrical active packets generator that consists of two parts: a single LOD generator and a sequence LOD generator. Single batches would be generated in cases where other users approach this one or its own movement toward any other user. The sequence of batches would be sent when the group is joined by a newcomer. Therefore these two 3D data generators will create a 3D DVE traffic pattern that reflects the changes in the required fidelity of users' representations within one multicast group and consists of two types of streams: 1) single batch streams that is required by the members of the group; 2) sequences of batches streams that is necessary to deliver to the newcomers.

The filtering mechanisms on the active routers will filter the sequence of batch streams using the information about the changes in the multicast group in addition to the LOD lookup table information and forward the complete sequence of batches exactly to any newly users.

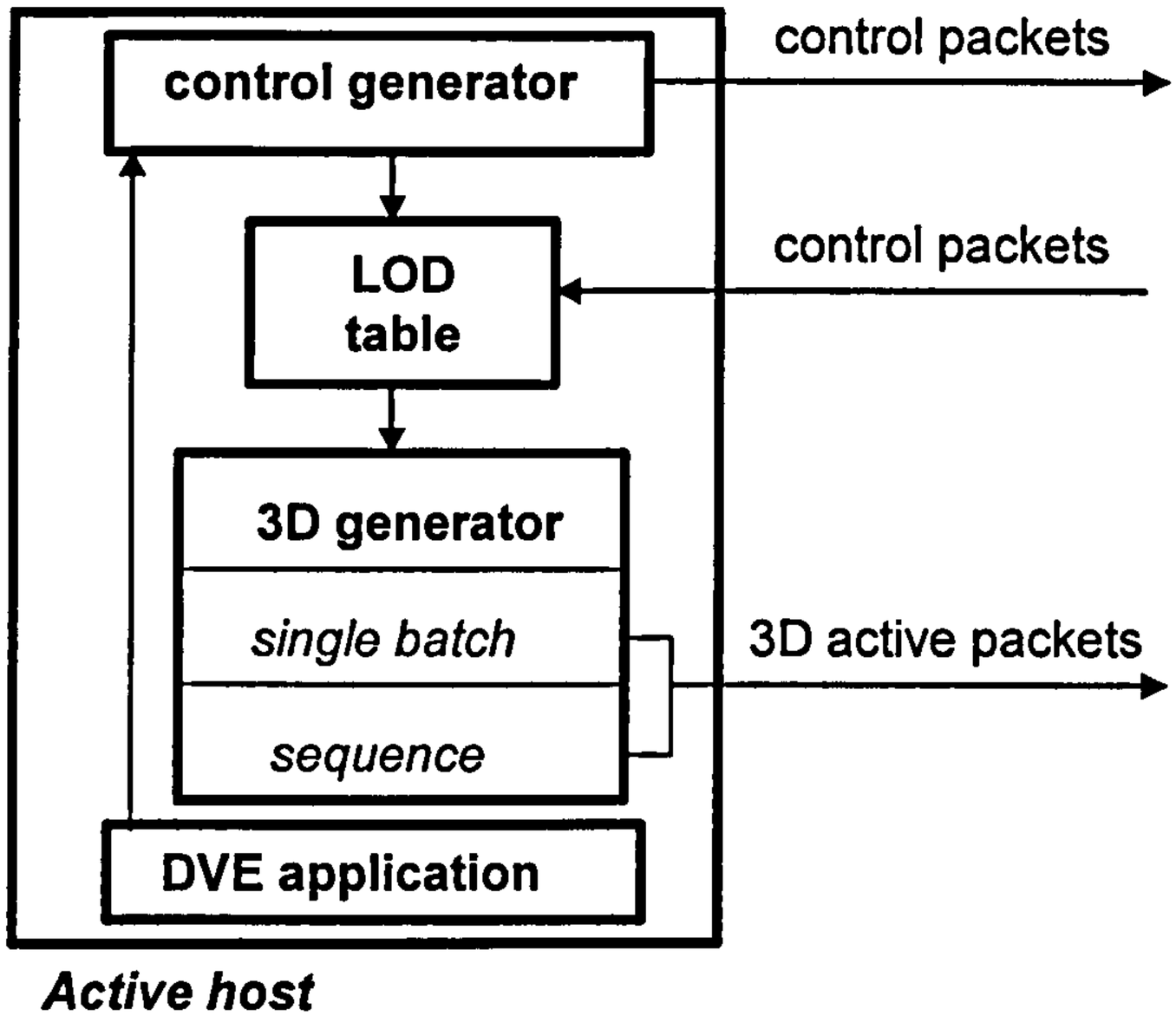


Figure 6-9 Active host architecture model

6.4 Summary

This chapter discusses an active networking architecture model that is proposed in this work. This architecture employs active networking techniques to improve multicast of 3D DVE data by restriction of unnecessary flows of 3D packets. At the beginning of the chapter the discrete active approach or programmable switch has been named as the basis for the proposed architecture. Then the three layers of the programmable switch approach that are used in our architecture have been described which are the active node, active extension and active packets.

After that it was discussed that the proposed architecture includes two components – the active router and active host.

Firstly the model of an active node has been presented. Here the active node architecture in terms of the router architecture has been discussed. Then the execution environment that performs active processing has been described. Next the active packets formats and filtering mechanism have been explained.

Secondly the active host architecture model has been considered. In the context of the host architecture 3D data transmission within a DVE application

runtime has been studied. This study led to the design of the active host architecture model.

PAGE NUMBERING AS ORIGINAL

Chapter 7

Flow and Application Models

In this chapter the part of the experimental framework that consists of the flow model and application model is described. These two models are necessary for running simulation testing experiments and evaluation of the proposed active architecture for DVE applications. The flow model represents the 3D geometrical data which is transmitted by DVE users. The application model controls the pattern of transmission of these 3D streams that is driven by the sequence of movement events of every user.

7.1 Flow Model

The proposed active networking architecture considers transmission of progressive representations of 3D objects. Therefore, it was necessary to construct a 3D representation that incorporates the level of detail concept and allows progressive transmission to be able to test our architecture. As a prototype for our approach Progressive Meshes [Hoppe, 96] was chosen, as well as the idea of grouping all vertex-split operations into batches introduced in Compressed Progressive Meshes (CPM) by Pajarola et al [Pajarola, 2000] (see Chapter 4). However, the flow model for evaluating the active networking architecture does not include all the compression aspects described by the authors of the CPM approach. To evaluate bandwidth saving that is achieved by an active filtering approach it is important to know the relative sizes of batches in the progressive mesh representation. However, the use of compressed data will gain additional saving of the bandwidth but this will be achieved by 3D data compression methods and not by our approach. The flow model benefits from CPM's coarse-grain partition of refinement operations which offers several LODs and reflects the relative sizes of parts of this representation. The simplification algorithm that transforms the original single-resolution mesh into a multiresolution progressive representation has been implemented. As original meshes VRML polygonal models have been used. The VRML polygonal model is simply a list of faces with each face described by a sequence of vertex references [Taubin, 99b]. Each vertex reference points to a separately stored triple of floating point numbers

representing the coordinates in three-dimensional space. A polygonal mesh with V vertices and F faces is represented in VRML by an `IndexedFaceSet` node with a `coord` array, and `coordIndex` array as a minimum. Three floating-point numbers describe the position of each vertex in the `coord` array. Each face of the polygonal mesh with $n \geq 3$ corners is represented in `coordIndex` by n different indices into the `coord` array. The resulting representation consists of two parts. First, the crude model, M_0 that usually contains 5 to 10 per cent of the number of triangles of the entire original model and must be transmitted to the user. Second, the sequence of batches of refinement operations (see Section 4.4). These may be downloaded systematically, or only on user's request, and create the sequence of increasingly precise mesh approximations $M_1, M_2, \ldots, M_{\max}$. Construction of the new representation proceeds in several steps. First, as the simplification algorithm applies a sequence of edge-collapse transformations to the original mesh the creation of the first batch content $M_{\max} \longrightarrow M_{\max-1}$ is started. After this first stage we obtain the simplified mesh $M_{\max-1}$ and the batch $M_{\max} \longrightarrow M_{\max-1}$. Next, we'll apply the same calculations and we'll get the more crude mesh $M_{\max-2}$ and the second batch $M_{\max-1} \longrightarrow M_{\max-2}$. The simplification stops at a crude model M_0, when a given mesh complexity is reached. The mesh complexity of the crude model is defined by the visual image of the simplified object. The CPM authors report that when the simplified mesh reaches a size from 5 to 10 per cent of the number of triangles of the entire original model, further simplification steps make the object unrecognizable.

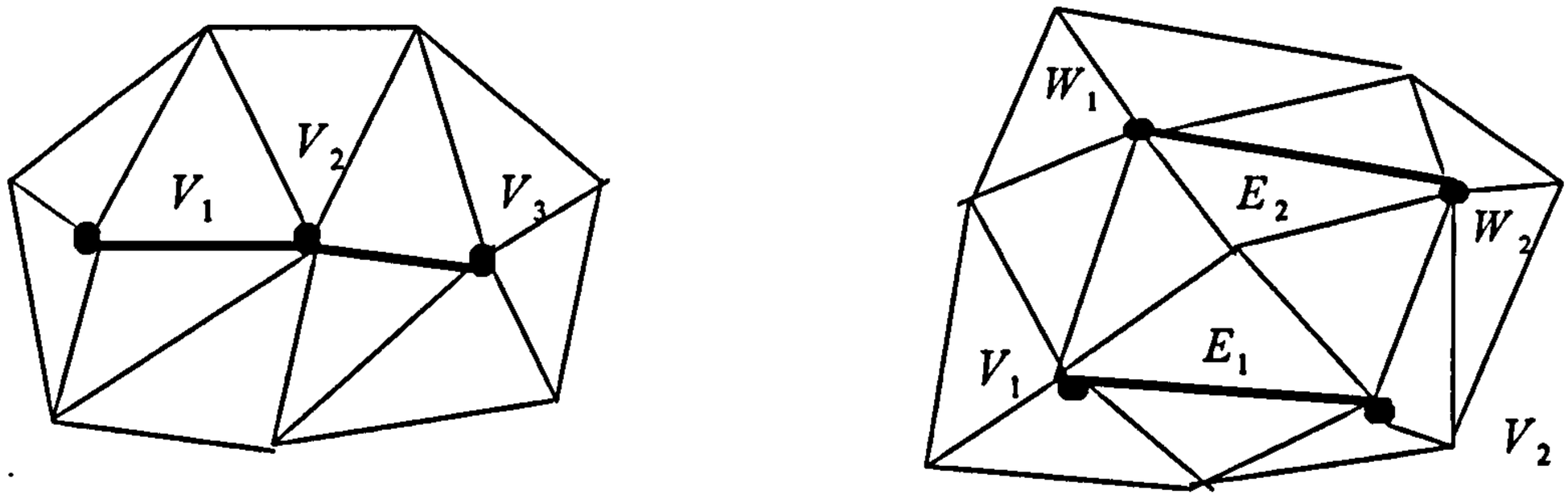
Therefore the multiresolution representation that includes several LODs of the object is obtained. The coarse model M_0 will be transmitted first and all geometrical packets, which contain data of M_0 will gain 0 level of detail for LOD in the active header in our flow model. After that the sequence of refinement batches, beginning from coarser batches, will be transmitted. Each batch in this sequence contains the geometrical packets with corresponding LOD active headers. For example, packets from batch $M_0 \longrightarrow M_1$ will have LOD active header equal to 1.

Next, our simplification algorithm along with the restoration process will be discussed. Then the progressive representations obtained for VRML files with different sizes will be presented.

7.1.1 Mesh Simplification Algorithm

At every step of simplification the algorithm decreases the number of triangles that define the approximation of the object's surface. To obtain the approximation with less detail the number of edges of triangles is marked as collapsible and then removed from the mesh. To increase the number of triangles discarded in each simplification step and therefore to reduce the number of levels of detail, it might be reasonable to include the maximum edge collapses for construction of each simplification batch $M_{l+1} \longrightarrow M_l$. However, we follow some restrictions for collapsing edges in M_{l+1} described in the CPM paper, which must be fulfilled to avoid singularities in the simplified mesh M_l and to keep degradation of the object's image acceptable for the viewer.

1. At most two vertices may be collapsed into one. The three vertices v_1, v_2, and v_3 may not be collapsed into one (Figure 7-1a).
2. For each edge $e_1=(v_1, v_2)$ that will be collapsed and any edge $e_2=(w_1,w_2)$ forming a quadrilateral (v_1,v_2,w_1,w_2) with e_1; e_1 and e_2 cannot be collapsed in the same batch (Figure 7-1b).



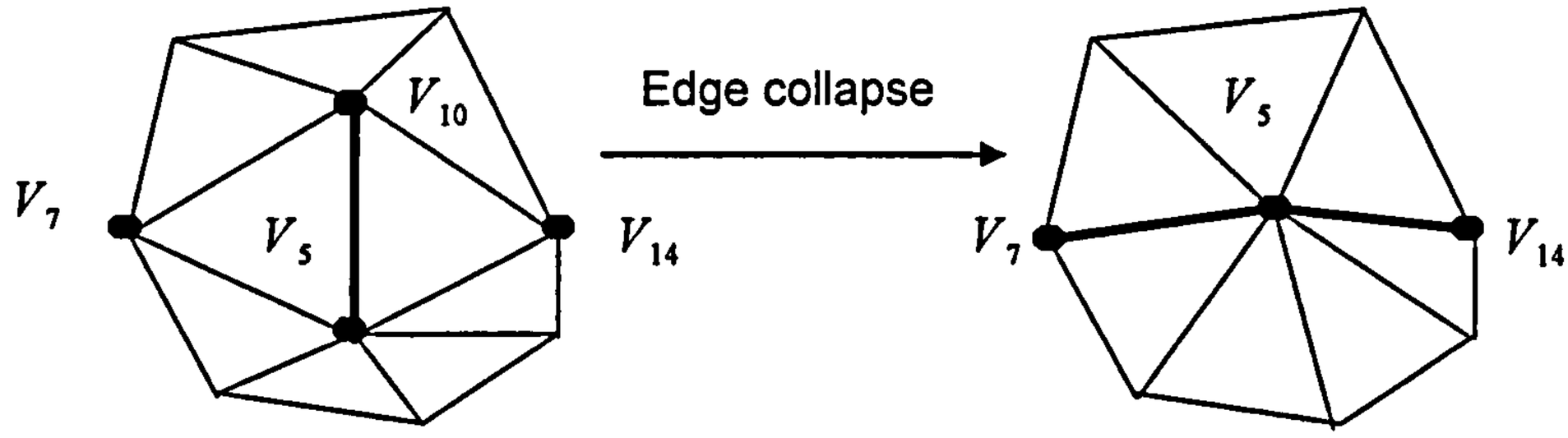
a) b)

Figure 7-1 Invalid edge collapses

These restrictions to choose the collapsible edges in our task of construction the flow model help to divide all refinements into the batches that

reflect several LODs of the object. The size of each batch will decrease from the highest detail batch to the lowest detail batch. The simplification algorithm will apply the same fulfilment connectivity restrictions to every approximation mesh and because the size of the mesh diminishes the number of collapsible edges and therefore the size of the batch will decrease for every successive less detailed batch. For simplicity we won't consider approximation errors and instead of working with a selection of collapsible edges in order of increasing approximation error, the algorithm will check only fulfilment of connectivity restrictions.

To be able to work with edges the new list `EdgeTable` has to be created from `coordIndex` array of triangles in our given VRML data structure. As a result of traversing the `EdgeTable` the subset of edges, which will be collapsed in the next step will be created. Collapse of edges marked in the M_{i+1} approximation mesh leads to a new simplified approximation M_i by building the new `coord` and `coordIndex` arrays. Figure 7-2 illustrates the edge collapse transformation.



a) M_{i+1} mesh b) M_i mesh

Figure 7-2 The edge collapse transformation

For example the edge $V_5 V_{10}$ is marked for collapse. The collapse of this edge will lead to obtaining the new split-vertex in the `coord` array of M_i. The coordinates of this vertex will be calculated as: $\mathbf{V}_5 = (\mathbf{V}_5 + V_{10})/2$. As it does not make any difference to the traffic flow model, we assume that all edges collapse to their midpoint instead of choosing a better approximation point as applied in the PM approach.

Also we remove two faces from the `coordIndex` table of M_{i+1} to obtain M_i. In our example it is $V_5 V_7 V_{10}$ and $V_5 V_{10} V_{14}$. From the `coord` array vertex V_{10} vanishes and vertex V_5 will be marked as split-vertex in M_i. After collapse of all marked edges we construct a vertex-spanning tree of M_i and produce split-vertex marking bits for batch $M_{i+1} \longrightarrow M_i$ by traversing this tree and mark every split-vertex by the bit equal to 1. We include the connectivity coding (cut-edges) and the prediction correction vector of the split-vertex into the batch content as well as the index of the split-vertex in the mesh M_{i+1} and its coordinates, to be able to reconstruct the mesh M_{i+1} from the mesh M_i. However, this information is not compressed as the compression issues are out of the scope of our work. The algorithm for one step of the construction of the progressive representation is shown in Figure 7-3. The simplification program repeats this procedure until the resulting mesh reaches the required minimum level of detail.

As a first step in the implementation of the algorithm which constructs the PM representation of objects is the parser program which extracts geometrical information from `IndexedFaceSet` geometrical node of the scene graph described by a VRML file has been developed. As a result the vertex (`coord`) and polygon (`face`) tables are obtained. Then our algorithm creates the third required table – an edge table (`EdgeTable`, as it shown in Figure 7-3). Because the construction of the new simplified mesh can be achieved by applying a series of the edge collapse operations, it is necessary to build a subset of the collapsible edges `colEdge` of the each simplification step. The subset of the collapsible edges has the same structure as the edge table and it is a list of two vertex references. To follow Figure 7-3 we can find the next structure – an array s_vert. This array contains all split vertices of the given simplification step and has the same structure as a `coord` table. The results of applying each step of our algorithm will be the new `coord` and `face` tables which describe the new simplified mesh and serve as input data structures for the next simplification step and the batch which includes the split-vertex marking bits for batch $M_{i+1} \longrightarrow M_i$, the connectivity coding (cut-edges), the prediction correction vectors of the split-vertices, the indices of the split-vertices in the mesh M_{i+1} and their

coordinates. These results are organized into the following data structures: `coord_new`, `face_new`, and batch structures. To obtain the split-vertex marking bits for the batch we traverse the `coord_new` table and search for the vertices' coordinates that are equal to the coordinates of vertices in the subset of the split-vertices `s_vert`. If the vertex from the `coord_new` table exists in the `s_vert` table it will be marked as the split-vertex and the bit will gain value 1. Otherwise the bit will be marked, as 0 and a restoration process will not split this vertex.

A restoration program also has been implemented. The aim of showing restoration of the progressive meshes was to ensure that information included into refinement batches is sufficient to reconstruct the series of approximation meshes with increasing levels of detail. The next section demonstrates simplification and restoration results for VRML representations.

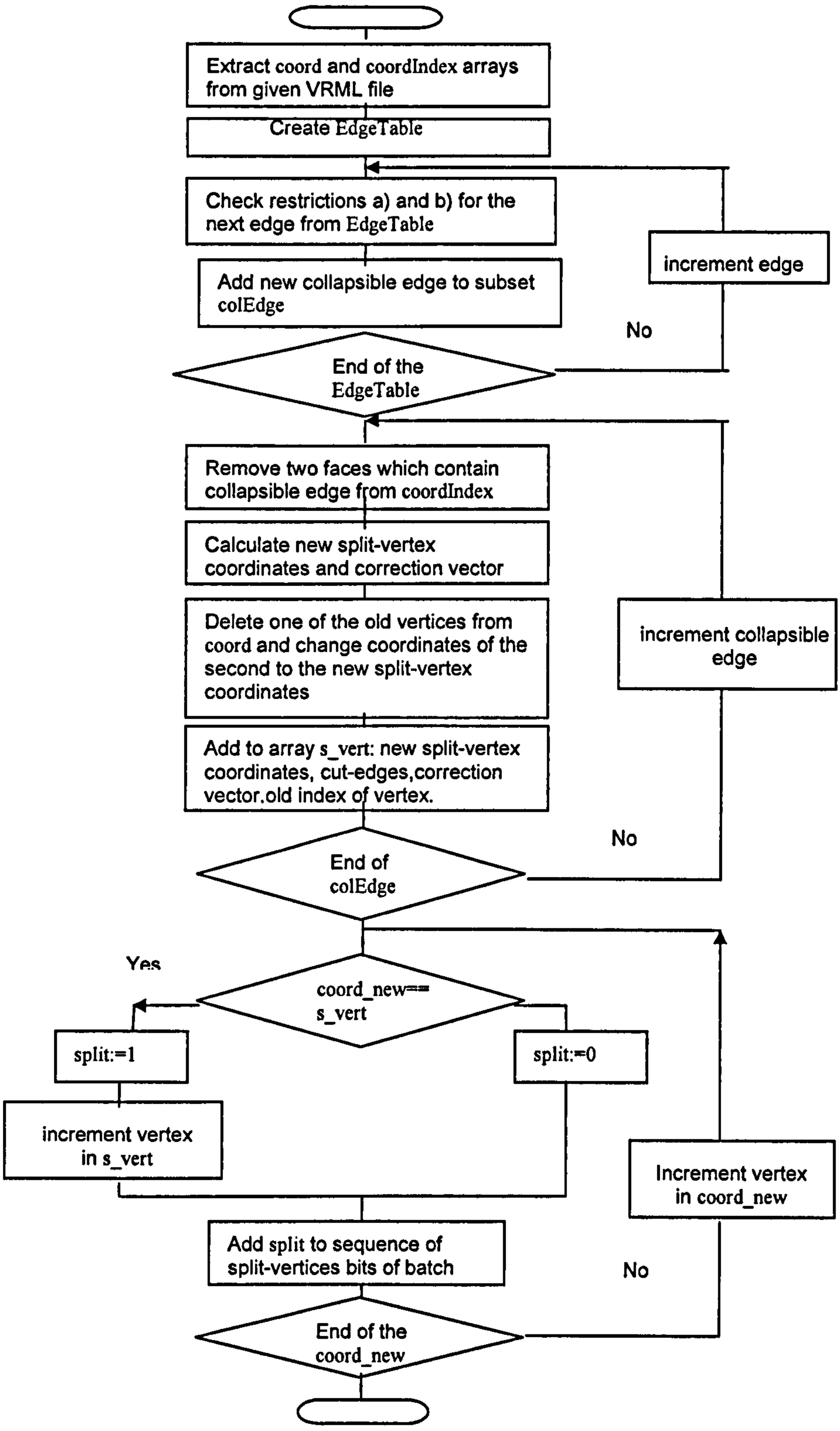


Figure 7-3 One step of mesh simplification algorithm

7.1.2 Simplification Results Example

The simplification and restoration programs have been implemented using the C language. The simplification algorithm has been tested for small size 3D objects. The VRML file *baseballbat.wrl* that contains a 3D geometrical set with 196 vertices and 360 faces has been used as an original single-resolution representation. This object's VRML file has been downloaded from the site: www.web3d.org/x3d/vrml/objects .

The simplification of this file has been performed for two steps. The restoration algorithm has also been applied. The results for *baseballbat.wrl* VRML file are presented in Figure 7-4.

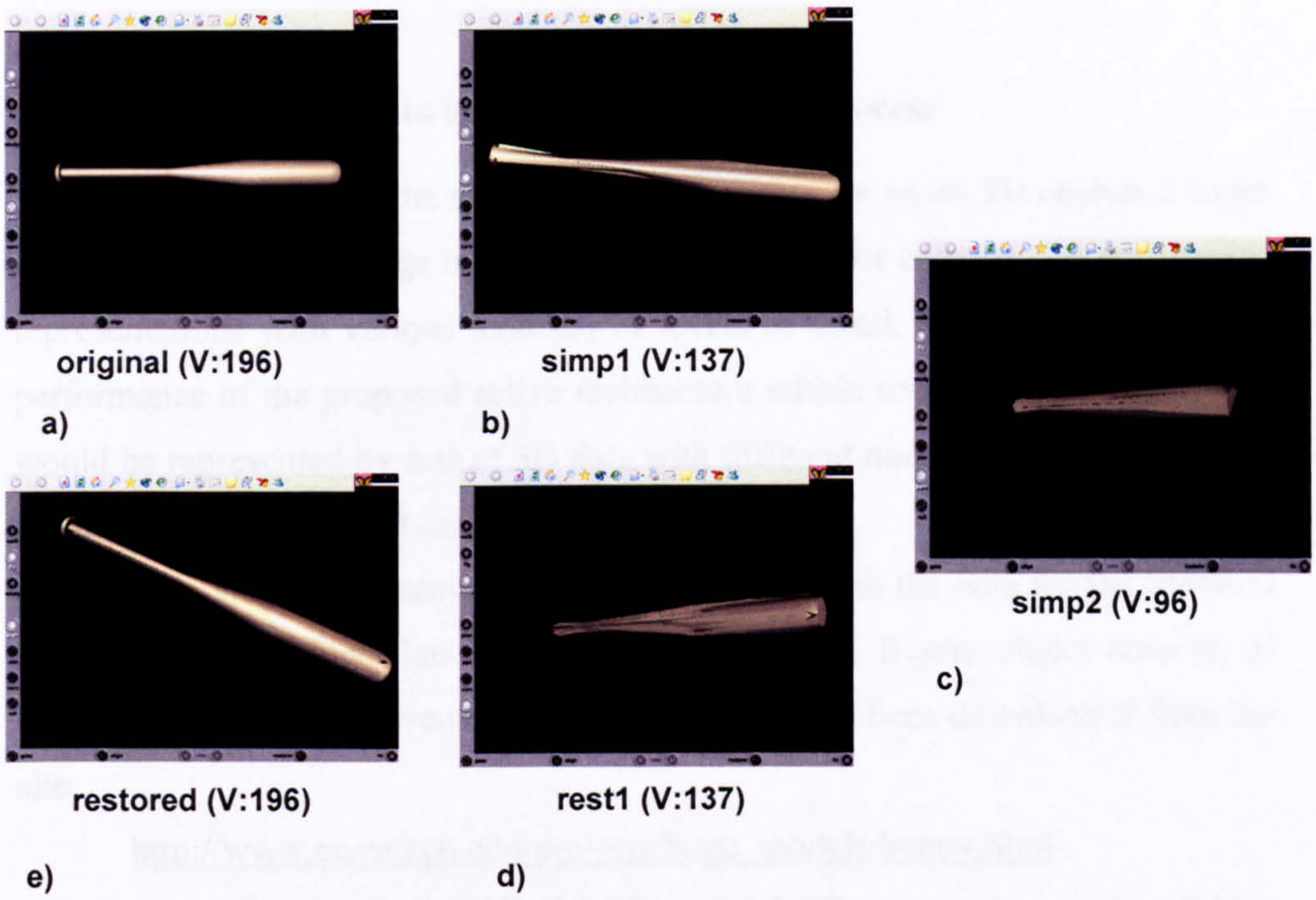


Figure 7-4 Simplification and restoration results

Figure 7-4a shows the original object. The object that has been obtained after the first step of simplification and contains 137 vertices is shown in Figure 7-4b. The Figure 7-4c presents the simplified *baseballbat* object after the second step of simplification which contains 96 vertices. Two restoration steps results are presented by Figures 7-4d – 7-4e.

Therefore a progressive representation with 3 levels of detail has been constructed for the original *baseballbat.wrl* VRML file. This representation contains a crude model (*simp2.wrl*) that would form the 3D geometrical packets with level of detail 0 for the LOD active header in our flow model. The geometrical data that allows reconstruction of the *simp2* mesh to the *rest1.wrl* file (Figure 7-4d) would be included into refinement *batch1* that would form the 3D geometrical packets with level of detail 1 for the LOD active header in our flow model. The geometrical data that allows reconstruction of the *rest1* mesh to the *restored.wrl* file (Figure 7-4e) would be included into refinement *batch2* that would form the 3D geometrical packets with level of detail 2 for the LOD active header in our flow model. The content of the crude model and the two batches is presented in Appendix I.

7.1.3 Progressive 3D Data used in the Evaluation Process

After the testing the simplification algorithm for small 3D objects a larger mesh has been used. Large meshes are more suitable for construction progressive representations with various numbers of levels of detail. As we aim to evaluate performance of the proposed active architecture within scenarios when the object would be represented by sets of 3D data with different number of LODs, the flow models for these different cases have been obtained.

As a single-representation non-progressive mesh the data for the Stanford Bunny large 3D object has been used. The Stanford Bunny object consists of 69,451 faces and 35,947 vertices. This object's file has been downloaded from the site:

http://www.cc.gatech.edu/projects/large_models/bunny.html .

Flow models with 1 LOD, 3 LOD, and 9 LOD progressive representations were used for evaluation.

1) 1 LOD flow model.

The flow model for the non-active scenario consists of the original single-resolution representation which is a 1 LOD representation.

Common representations of triangulated meshes usually store the triangles as an indexed face list, where the coordinates of each vertex are three floating-point numbers. Therefore, each triangle requires 12 bytes for the

indices and every vertex, 12 bytes for the coordinates. To transmit the original model using the uncompressed representation would require: (number of faces + number of vertices)× 12bytes. Therefore the size of the 1 LOD model used in the non-active scenario is equal to **1,264,776 bytes.** We assume that the payload of geometrical packets has constant size equal to 1Kbyte. Hence the non-progressive 1 LOD flow model would require **1265 packets.**

2) 3 LOD flow model.

The 3 LOD progressive representation of the original file which includes two refinement batches and the crude model was obtained after applying two steps of the simplification algorithm. After the first step of simplification the number of faces and vertices has decreased to 46,219 faces and 24,291 vertices. The simplified mesh after the second step of the simplification of the crude model contained 31,713 faces and 17,008 vertices.

Next the calculation of the crude model and batches sizes is shown.

Crude model size (LOD0): (31,713+17,008)× 12 bytes=**584,652bytes** or **585 packets.**

First batch (LOD1) consists of:

Sequence of split-vertex marking bits: 17,008× 1bit=17,008 bits.

Indices of simplified mesh vertices in the more detailed previous representation: 17,008× 4bytes=68,032 bytes.

Data for every split-vertex (correction vector - 3 floating-point numbers, two cut edges – 4 integer numbers, and index of deleted vertex in more detailed previous representation – 1integer number):7,261× 32bytes=232,352 bytes, where 7,261 is number of split vertices and it is equal to the difference between the number of vertices in the crude model and the next more detailed mesh (24,291-17,008=7,261).

17,008bits+68,032bytes+232,352bytes=**302,510bytes** or **303 packets – batch1 size.**

Second batch (LOD2) consists of:

Sequence of split-vertex marking bits: 24,291× 1bit=24,291 bits.

Indices of simplified mesh vertices in the previous more detailed representation: 24,291× 4bytes=97,164 bytes.

Data for every split-vertex (correction vector - 3 floating-point numbers, two cut edges – 4 integer numbers, and index of deleted vertex in the previous more detailed representation – 1integer number):
11,616×32bytes=371,712 bytes.
24,291bits+97,164bytes+371,712bytes=**471,912bytes** or **472 packets–batch2 size.**

Table 7-1 summarizes the sizes for single refinement batches and the sequence of batches. The sequence of batches is sent within an initial transmission and on events of joining the group by a new user. These flows would be generated on the active host by a single batch or a sequence of batches components of the 3D packets generator (see Section 6.2).

Table 7-1 3 LOD flow model

Number of packets	Level of detail		
	0	1	2
Single batch	585	303	472
Batches 0 - n	585	888	1360

3) 9 LOD flow model.

To obtain 9 LOD progressive representations eight simplification steps have to be applied. To construct the 9 LOD model the two highest batches from the 3 LOD model have been used as highest batches for the 9 LOD model. It was found that the general pattern of decreasing the number of vertices of the mesh is the similar to that described by authors of the CPM approach [Pajarola, 2000] and shows a decrease after every simplification step by about 32 %. The number of vertices discarded by every simplification step depends on the smoothness of the object as it is reported by the authors of CPM method and for the objects with smooth surfaces the decrease of the size of the mesh tends to be the same for all following steps of the simplification. The Stanford Bunny large model shows a very smooth surface and is generally used as an example of a smooth object. Taking into consideration these factors the following less detailed meshes sizes have been estimated.

The size of the crude model for 9 LOD model decreases from 69,451 faces and 35,947 vertices for the original representation to 5,036 faces (7.69% from original number of faces) and 3,066 vertices (8.53% from original number of

vertices). In the CPM algorithm [Pajarola, 2000] the authors show that the crude model M_0 usually contains 5 to 10 percent of the number of triangles of the entire model. Therefore, the nine levels of detail representation are acceptable, as the size of the crude model has reached the minimum size. Hence, the estimated sizes of the refinement batches and the crude model were used for the 9 LOD flow model. Table 7-2 summarizes the sizes for single refinement batches and sequence of batches for the 9 LOD flow model.

Table 7-2 9 LOD flow model

Number of packets	Level of detail								
	0	1	2	3	4	5	6	7	8
Single batch	116	40	53	71	98	139	203	303	472
Batches 0 - n	116	156	209	280	378	517	720	1023	1495

7.1.4 Summary

Because not all the compression issues are considered in our simplification algorithm (uncompressed data is used for every split-vertex to restore the mesh) the total size of the multiresolution representation is larger than the original representation (original – 1265 packets, and progressive – 1360 and 1495 packets for 3 LOD and 9 LOD respectively). However, the use of compressed progressive meshes (e.g. CPM representation) would further improve the bandwidth saving within 3D transmission, especially for the 9 LOD representation, as within the 9 LOD CPM representation much more geometry data would be encoded as the compressed refinement operations data. Nevertheless, our flow models reflect the relative sizes of parts of the progressive representation and allow us to explore the comparative performance of active and non-active approaches, since within a non-active approach we also use an uncompressed explicit representation of the 3D geometry (see Section 7.1.3).

However, using flow models with larger numbers of LODs the virtual world would be divided into more subspaces where the probability that users are placed in the same subspace decreases and, therefore, the restriction of unnecessary 3D data flows within one multicast group would be implemented more effective. This goal would be achieved by dividing the VW into more

subspaces and therefore filtering the highest more detailed batches (see Table 7-2) for the receivers that are placed far from the sender in the virtual world.

7.2 Application Model

The next part of the evaluation framework is the application model which reflects the interactions between participants in the DVE application. In this work we investigate an exchange of 3D data between DVE users; therefore the model of the application would include events that define the transmission of 3D data flows within the runtime or post initialization stage of the application. Users' movement events invoke an additional transmission of more detailed 3D data flow when a sender is approaching a receiver. Because, when users move from the one subspace of the two-dimensional grid to another the distances between users and therefore the required level of fidelity (or LODs) for representations of other users will be changed. Hence, the application model defines the transmission pattern of the 3D streams that is driven by the sequence of the movement events of every user.

7.2.1 Users' Movement Generation Algorithm

A program that generates scripts of users' movements has been developed to obtain the application model. The two-dimensional virtual world has been divided into two grids (3x3 grid and 9x9 grid) that will allow generation of two movements script for every user (Figure 7-5). The virtual world model incorporates two different progressive representations. Why have 3 LOD and 9 LOD representations been chosen? The 9 LOD representation has been shown to be acceptable in terms of the size of the crude model. As the limit of reduction is shown in the CPM method from 5 to 10 percent so it in general can be 10 LODs representation. However, for better modelling of the virtual world we choose the 9 LOD representation as it allows the use of the same grid for the 3 LOD and 9 LOD models. To be able to evaluate an active architecture for 3D representations with less LODs the 3 LOD model has been used as the coarser 3x3 grid could be easily divided into smaller subspaces to form the 9x9 grid. Two generated scripts keep consistency for 3 LOD and 9 LOD scenarios as they would capture the same

trajectory of user's movement through the virtual world. The difference there is that for 3 LOD case the events of crossing by user 3x3 grid subspaces boundaries would be described in the 3 LOD script, as for 9 LOD scenario the events of crossing by user 9x9 grid subspaces boundaries would be reflected in the correspondent script.

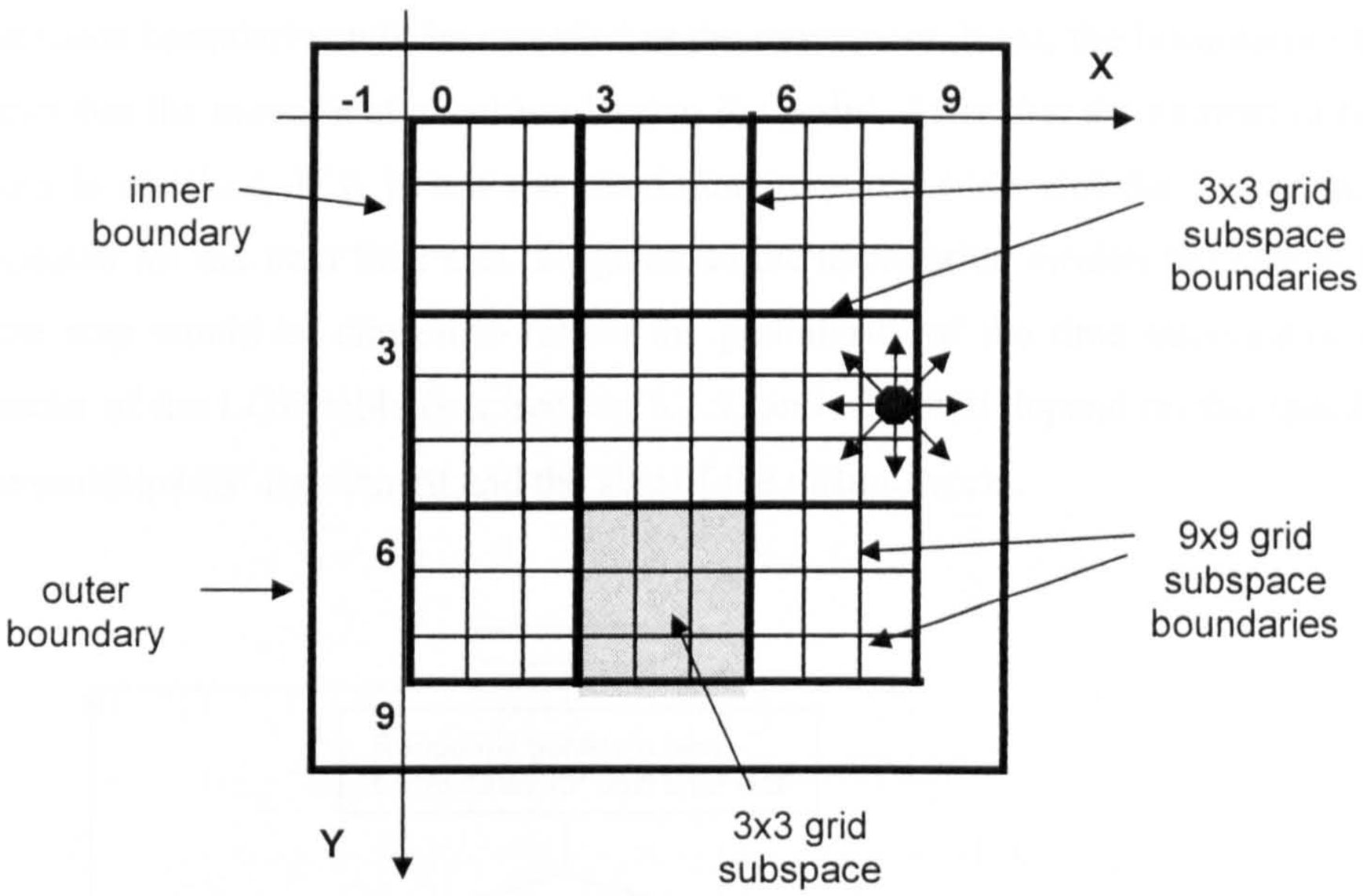


Figure 7-5 Virtual world partition

To model events of leaving and joining the group by the users the VW has outer boundaries. A user is allowed to cross the inner boundary, but is unable to cross the outer boundary. This restriction will let users leave and rejoin a group. The events of leaving and joining the group form the movements script for the non-active scenario and these events also keep consistency with two active scenario scripts. The movement event is modelled using random numbers generated according to a normal distribution. The user can move to the adjacent subspaces in eight directions or stay in the same subspace (see Figure 7-5). The probabilities of user's movement directions are defined by randomly generated numbers. The user moves in a certain direction if the generated random number exists within the predefined range, so for example, by increasing this range the probability to move in this direction will be increased. By changing these probabilities this method allows creating diverse movement patterns.

The algorithm that is used to generate a sequence of movements for one user is outlined in Figure 7-6. First, the new coordinates (coordinates of the square in the grid) of the user are generated randomly. Then, if there are any changes in coordinates in comparison with the previous time slot movement they are recorded. Then, the calculation of the new mutual LOD distances to the all other users is performed. However, for the 3 LOD case only events of crossing 3x3 grid subspace boundaries will be recorded as the movement. Next, the information that describes the movement event is added to the script. After that the number of time slots is checked, if it is not the maximum required time slot the procedure is repeated for the next time slot. To generate the application models the size of the time step would be chosen to reflect the granularity of the time intervals of the checks of the LOD table (see Section 6.2.1) and also will depend on the speed of the participants' movement and the size of the virtual world.

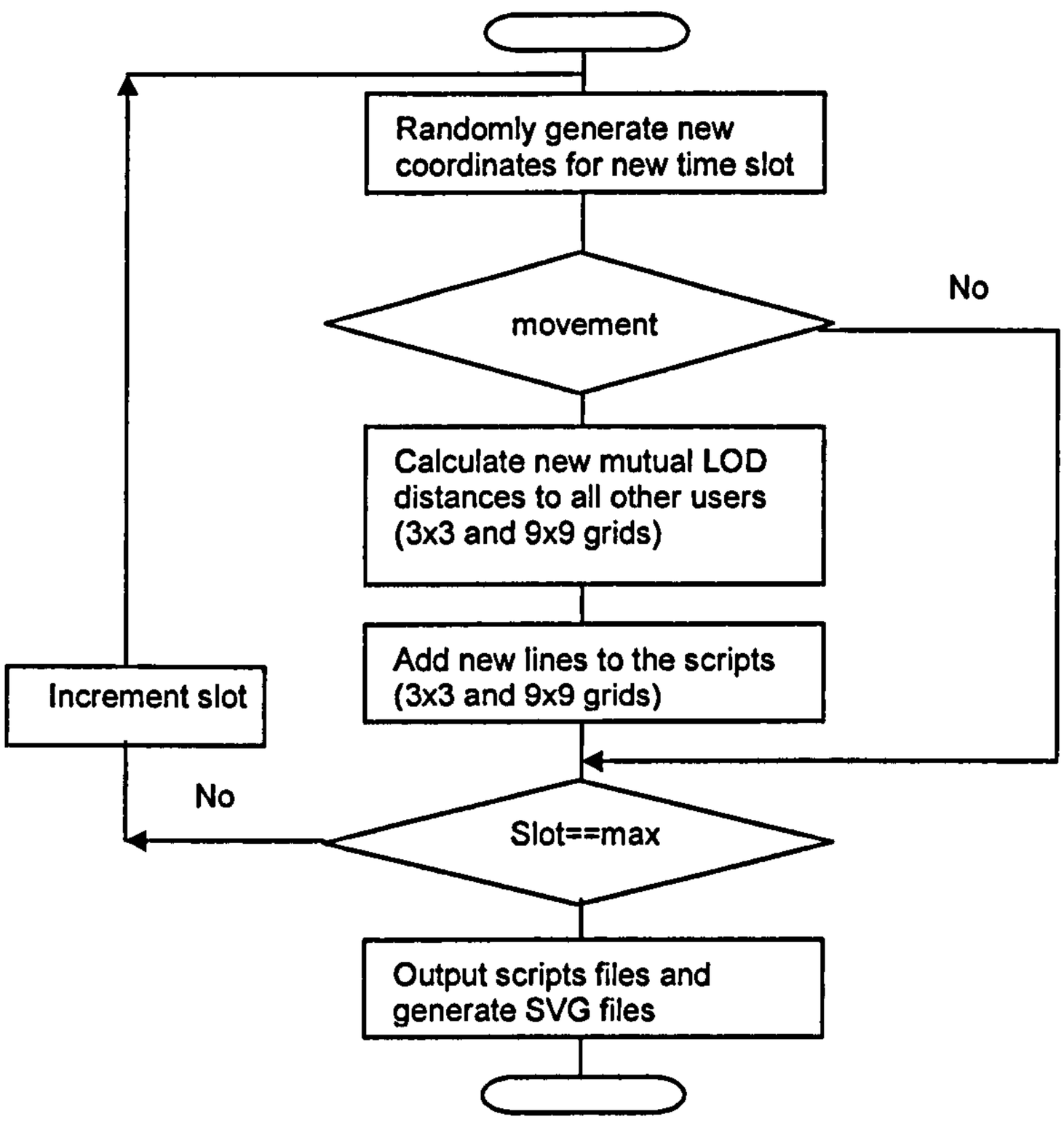


Figure 7-6 Algorithm to generate sequence of movements for one user

When construction of the 3 LOD and 9 LOD movement scripts is finished this data is presented as text files to be used by the simulation program. The host uses these files to invoke the generation of control packets that update the LOD lookup tables on active routers and hosts.

Also we visualize the users' movement by generation SVG scripts for every user. The SVG code is generated by our C program that implements the script generation algorithm. The SVG visualization of the users' movement clearly shows the pattern of movements of all users and the changes of their mutual distances over time. This visualization is essential for the observation of the different movement scenarios and the development process of the algorithms that produce these different movement scenarios. The SVG programs create the graphical representation of the application model which is generated in form of the script text files to be used by simulation.

The script contains information that describes events of the user's movement. With every move the user sends a number of control packets that is equal to the number of other users in the group (n). These control packets reset the LOD distances to other users in the environment. Every movement is represented in the script by n rows, where every row consists of: 1) sender address; 2) receiver address; 3) required LOD (or distance between sender and receiver, where LOD equal to -1 means that the user has left the virtual world); 4) time in seconds when the control packet will be sent by the user. Sample scripts examples for 3 LOD and 9 LOD scenarios are presented in Appendix II.

7.2.2 Application Models used in the Evaluation Process

Several application models that have different movement patterns have been obtained using the script generation program. To obtain different movement patterns of participants the probabilities of users' movements in particular directions has been changed along with adding some restrictions on crossing the boundaries.

To be able to examine a range of traffic patterns several scenario scripts of users' movement have been generated. For convenience of discussion these scenarios have been given the names – *random, far, centre, and join.* In these application models we approximate the actual movement patterns of the DVE application users: 1) users that randomly move in the virtual world; 2) users that

keep to stay far away from each other; 3) clusters of users that are all interested in one place of the virtual world (e.g. centre of the virtual world); 4) users who frequently leave and rejoin environment. Therefore the names of the scenarios reflect the features of the pattern of movement.

Initially the number of users was four. The maximum number of time slots is 60 (or one-hour duration as we use the granularity of time equal to one minute). For every scenario the movements of users have been visualized. The full coverage of users' movements for 20 minutes is shown in Figures 7-7a – 7-7d. Every user is represented by coloured rectangle. The faint rectangles of a certain colour trace the movements of one user. When a rectangle gets more opaque it indicates that this subspace has been visited by the user more than one time within the duration of the application run. The examples of generated movement script text files for the different movement patterns are presented in Appendix II.

Next, the descriptions of the movement scenarios along with the SVG visualizations are presented.

1) *Random* scenario.

Users move randomly within the grid. Every user explores different parts of the virtual world. The probabilities of a users' movement in a particular direction are similar for all users. As there is no restriction on crossing the inner boundaries, some of the users (e.g. user C that is represented by the green rectangle) leave and join the group.

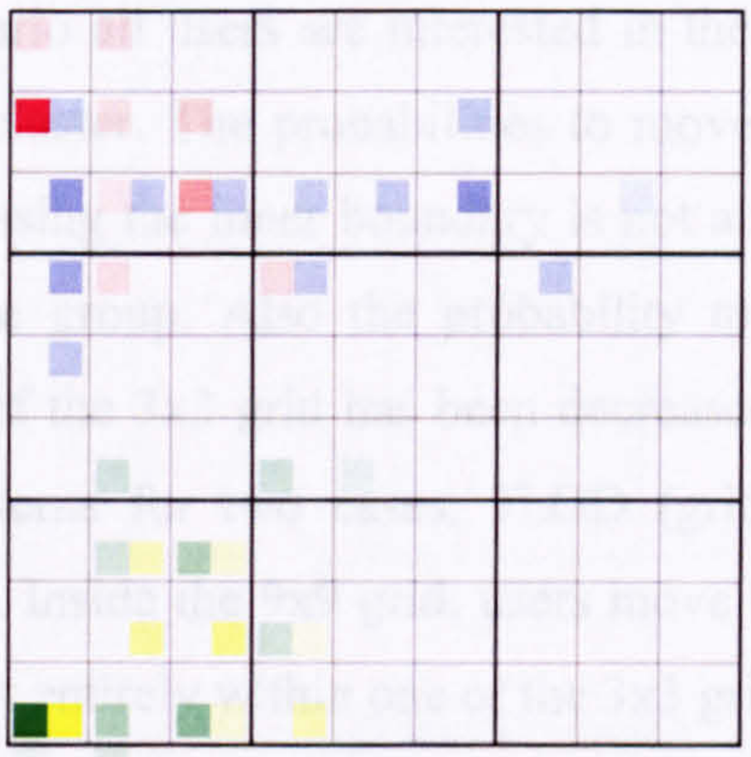


Figure 7-7a *Random* scenario

2) *Far* scenario.

Users are allocated far from each other in the corners of the grid. Every user moves in the corner that represents the user's area of interest in this scenario. This is achieved by increasing the probability of movement towards the corners for every user. Users never leave and join the group in the time of the experiment as they are not allowed to cross the inner boundary of the virtual world model (see Figure 7-5). Some of the users (user A – red rectangle, and user B – blue rectangle) remain entirely within one of the 3x3grid subspaces.

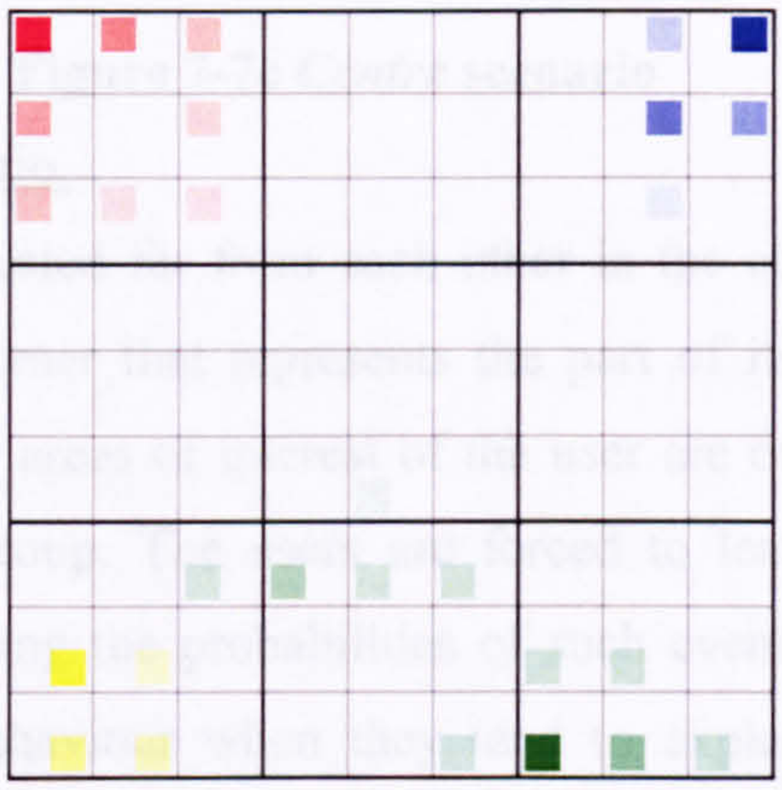


Figure 7-7b *Far* scenario

3) *Centre* scenario.

Users tend to stay in the middle of the grid with movements around the central area. In this scenario all users are interested in the one part of the virtual world and they form a cluster. The probabilities to move toward the centre have been increased and crossing the inner boundary is not allowed. Therefore, users never leave and join the group. Also the probability to cross the boundaries of the central subspace of the 3x3 grid has been decreased significantly for every user. The different patterns for two cases, 3LOD (grid3x3) and 9LOD (grid 9x9), should be noticed. Inside the 9x9 grid, users move between subspaces. However, they remain almost entirely within one of the 3x3 grid subspaces.

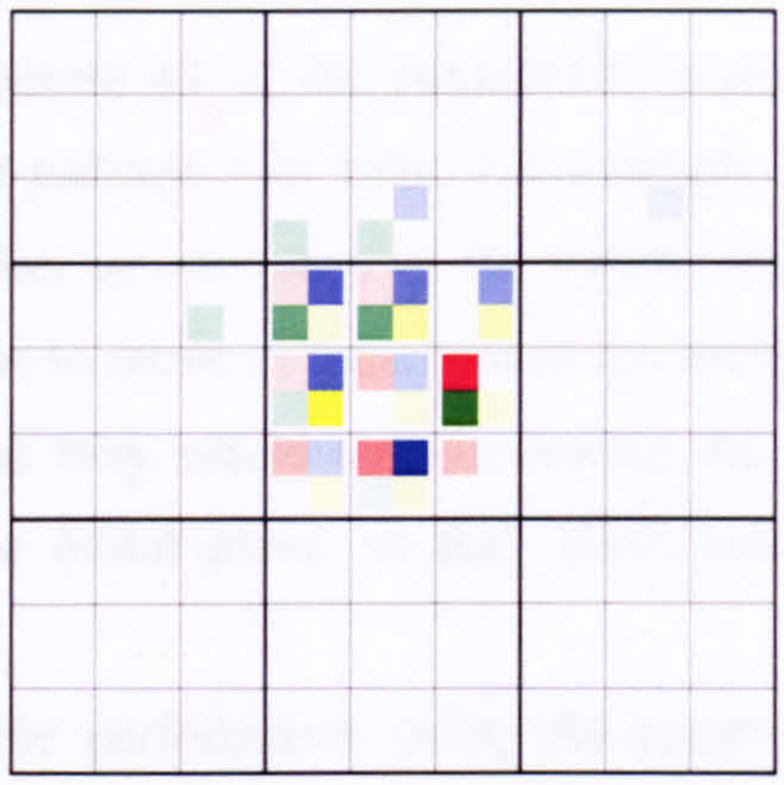


Figure 7-7c *Centre* scenario

4) *Join* scenario.

Users are allocated far from each other in the corners of the grid. Every user moves in the corner that represents the part of its area of interest in this scenario, where other areas of interest of the user are outside the grid. All users leave and join the group. The users are forced to leave and rejoin the group frequently by increasing the probabilities of such events. This kind of scenario models the users' behaviour when they tend to explore adjacent parts of the virtual world and more probably leave and join the group.

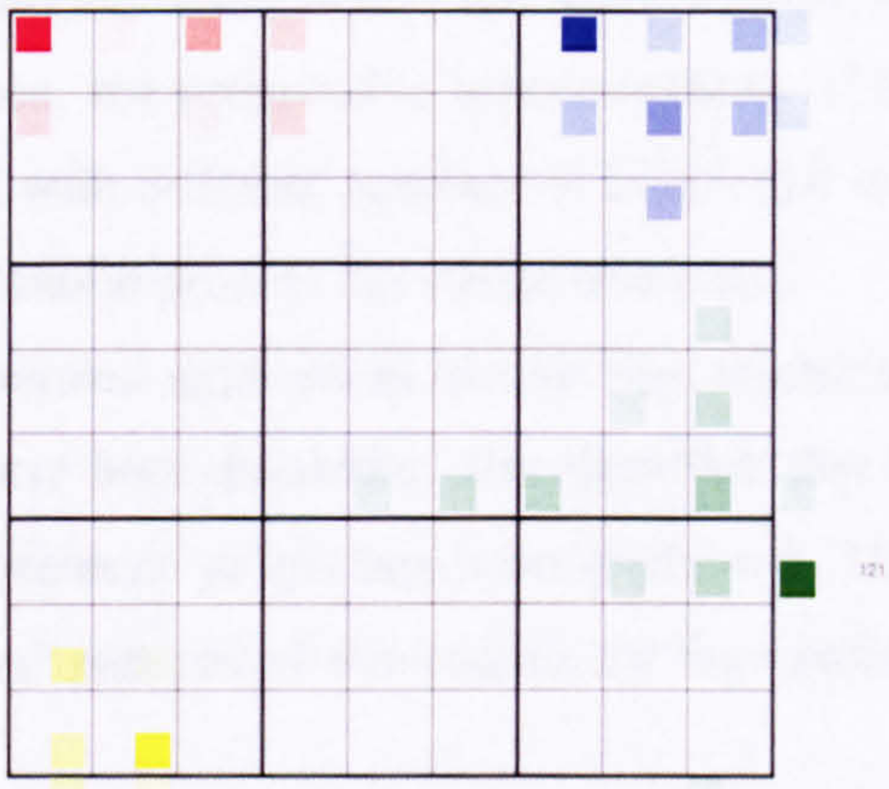


Figure 7-7d *Join* scenario

7.2.3 Summary

These four application models were used in the evaluation experiments. The different scenarios of movements allow comparison of the performance of the

proposed active approach for different DVE application models. The real DVE applications can incorporate all of the considered scenarios as the users could change their movement patterns over time. For example, at the beginning of the DVE session users gather in one place of the virtual world (that reflects *centre* scenario), then they start to move to the edges of the world to explore new places (*far* scenario), after that they would try to examine the adjacent worlds and to compare them with the initial place, so they leave and rejoin the group (join scenario).

Evaluation of the performance using the generated application models aims to discover the scenarios where the active approach is more efficient.

7.3 Chapter Summary

In this chapter the flow and application model parts of the evaluation framework have been presented.

First, the flow model that represents the 3D geometrical data which is transmitted by DVE users has been described. The simplification algorithm that has been implemented to convert conventional single-resolution 3D meshes into progressive meshes which incorporate LOD has been shown. The results of simplification and restoration of the small mesh (the baseball bat object) also have been demonstrated. Then, the progressive representations of the large mesh (the Stanford Bunny object) with different numbers of LODs that would be used as the flow models in the evaluation process have been described.

Second, the generated application models that represent different patterns of users' movements have been discussed. The algorithm that has been applied to generate the users' movement scripts has been explained. Then, the application models that reflect users' patterns of movements for four different scenarios have been demonstrated.

The work presented in this chapter aimed to model the flows of 3D data within DVE applications and the pattern of the exchange of such data among the members of one multicast group within the active and non-active approaches. These models will allow simulation experiments to be run in order to evaluate the proposed active approach (including 3 LOD and 9 LOD representations) in comparison to the non-active case for these different scenarios.

Chapter 8

Evaluation of the Active Approach within the Initialization Stage of the Application

The initialization stage of the DVE application is characterized by the initial users' transmission of their 3D representations at the beginning of the multicast session. Therefore, the application model deployed in these tests includes only transmission of a 3D stream at the beginning of the experiment with no following additional transmissions. The simulation models applied in these tests aim to evaluate the impact of using active node filtering on the host and router performance and to demonstrate that the simulation model of the active node part of the active network architecture reflects the active processing on routers and works properly. These simulation models were then extended by adding the active host component to be able to investigate the complete architecture model within the DVE application's runtime (see Chapter 9).

8.1 Simulation Model

The simulation model includes two basic elements, hosts and routers. The host and router elements have been implemented for both active and non-active approaches to be able to simulate application behaviour for both cases.

For this initialization stage simulation the host elements produce control packets and geometrical-information packets at the beginning of experiment. The geometry-information packets streams form the flows of 3D DVE data class (that could be named as the Geo_Info class or geometry information class). The control packet generation aims to test the correctness of updating the LOD lookup tables on the routers, therefore the control packets are sent within the time interval before the end of the initial transmission of 3D streams. The destination process on the host simulates the reception of the geometrical packets. The numbers of received packets and the time of reception are monitored by the destination process. The host model

for the non-active approach includes the geometrical packet generator that generates the sequence of packets that represent the original single-resolution non-progressive 3D model and the destination process.

The router element models an active network node that also has a multiservice capability. In this way DVE 3D data traffic can be allocated its own class (Geo_Info class), assigned its own portion of the bandwidth and therefore will be isolated from other classes of traffic. For simplicity the router is internally non-blocking, so queues will only build up in the output stages. The router can distinguish between active and non-active packets, which allows DVE packets to be sent using either active or non-active mode. In this way it is possible to compare the performance advantages gained from the active approach, by running the same scenario in both active and non-active mode.

To simulate non-active and active router's processing the simulation node models for both approaches have been developed. Non-active node presents the ordinary router functionality and includes only the multicast routing of the geometrical packets. The active node includes the processing of active Geo_Info class packets and active control packets. The LOD lookup table is kept by the active node and is updated by users' control packets. The active processing of geometrical packets is implemented as described in Section 6.1.2.

The OPNET simulation package has been used in this work. The OPNET (Optimum Networks) simulation provides a comprehensive development environment supporting the modelling of communication networks and distributed systems. Both the behaviour and performance of modelled systems can be analyzed by performing discrete event simulations [OPNET].

Examples of the OPNET simulation models for networks and nodes and used to evaluate the active approach within the initialization stage of application are included in Appendix III.

8.2 Experiment Description

For the initialization stage experiments we have used the simulation elements described in the previous section to build a network simulation model. It contains four hosts and five routers, as shown in Figure 8-1. The router nodes are named as routers

R1, R2, R3, R4, R5 and host nodes are named users A, B, C, and D. This particular network topology has been chosen by taking into consideration that we need to keep a reasonably small size of the network to start our research direction. The links between active routers could be considered as tunnels across the core networks.

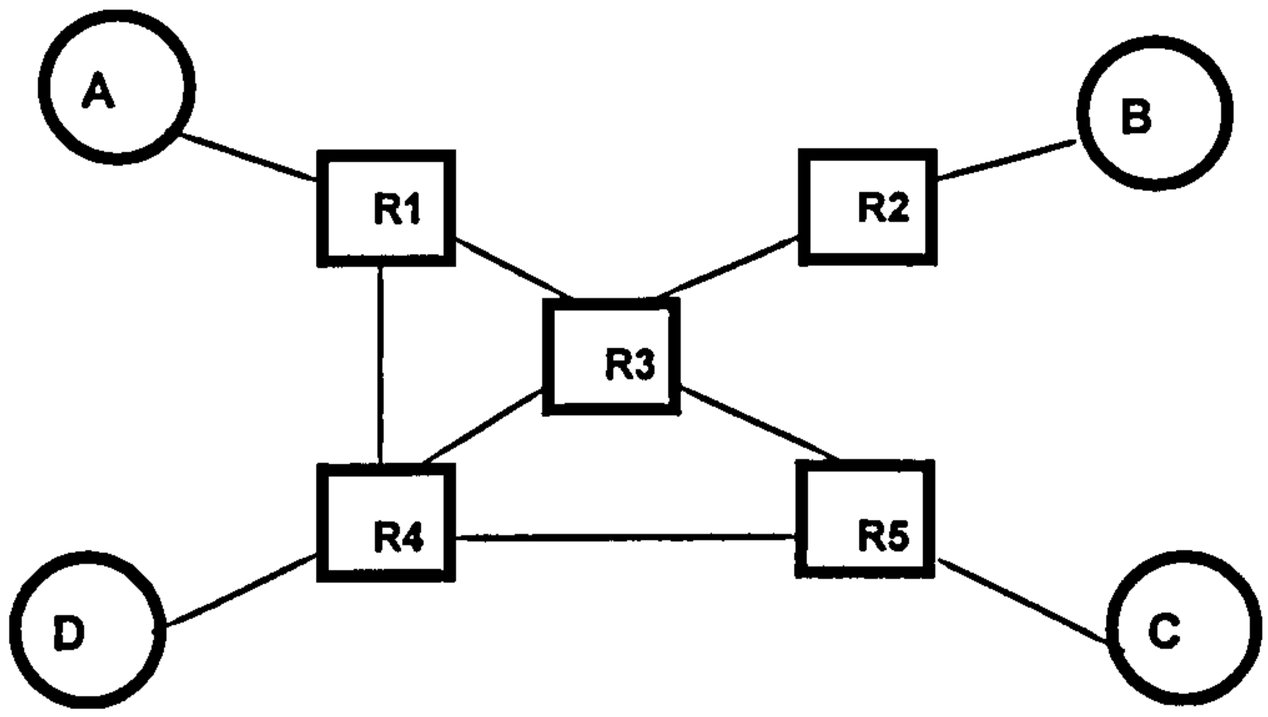


Figure 8-1 Network configuration that is used in basic experiments

The link rate between the hosts and the routers is set at 2Mbit/s, the links between the routers are set at 100Mbit/s, and the bandwidth assigned to the 3D data DVE traffic or Geo_Info class is 2Mbit/s.

All packets that convey geometrical information, whether that information be progressive or otherwise, are a fixed size of 1 kbyte. To simulate the processing delays associated with accessing the geometrical information and preparing the packets for transmission, geometrical-information packets are generated according to an exponential distribution. The use of the exponential distribution is an arbitrary choice to put variability into a packetizing process. We could try any kind of the distribution as it will not make any difference to the overall traffic pattern, because the links between the hosts and routers will smooth out the generated traffic.

8.3 Results from Simulation

8.3.1 Non-active Approach Results

This experiment has been carried out to obtain the non-active approach results that later will be compared with the active approach for our architecture model. The application model is represented by four users in the multicast group that transmit their representations to the group only at the beginning of the multicast session. The

flow model which is used in this experiment is the single-resolution original representation of the Stanford Bunny model which includes 1265 packets (see Section 7.1.3).

To test the correctness of the simulated multicast routing and to examine the end-to-end delay in reception time of the entire 3D representation the reception times on hosts have been measured. Within the DVE application this characteristic is important as it defines the initial waiting time to view the objects in the virtual environment.

Table 8-3-1 shows the reception time of the original representation from other multicast group members for user A. The required time to generate all packets by user A was **12.218** seconds.

Table 8-3-1 Initial reception time (non-active, four senders)

User	Number of hops	Reception time (s)
B	3	20.1819
C	3	20.8149
D	2	19.4498

The delay time (the difference between the time when the last packet in the representation was generated by sender user A (12.218 s) and reception time of this packet on the receivers (e.g. for the receiver user B it is equal to 20.1819 s) have been compared to the delays obtained in the experiment when only one user (user A) sends the data has been carried out and where the required time to generate all packets was **12.3251** seconds.

Table 8-3-2 shows the reception times for the experiment when only user A sends the data.

Table 8-3-2 Initial reception time (non-active, one sender)

User	Number of hops	Reception time (s)
B	3	12.3458
C	3	12.3458
D	2	12.3416

Tables 8-3-1 shows that the reception time increases in comparison with file transmission by only one user (Table 8-3-2). This is due to the fact that the queues are

building up on the routers as more flows of packets are generated and forwarded over the network and therefore the bandwidth requirements are increasing.

When several flows are aggregated, the resulting arrival rate on the router is the sum of the mean arrival rates of all flows. The aggregated inter-arrival time shows how frequently the packets from all aggregated flows arrive at the router. The aggregated inter-arrival time has to be greater than the service time for the packet on the router to avoid queuing of packets.

For example, we calculate the aggregated inter-arrival time for the central router R3 (Figure 8-1). Four flows of packets arrive on the router. Every flow has the same arrival rate $\lambda=1/\bar{a}$, where $\bar{a}$ is the mean of the inter-arrival time that is equal to the generation time 0.01(s).

As an example, the aggregated arrival rate on router R3 is calculated as follows:

$$\lambda_{agg} = \lambda_1 + \lambda_2 + \lambda_3 + \lambda_4 = 100 + 100 + 100 + 100 = 400 \text{ (packets/s)}$$

Then the aggregated inter-arrival time can be calculated as:

$$\bar{a}_{agg} = 1/\lambda_{agg} = 1/400 = 0.0025 \text{ (s)}$$

Service time is the time required to complete packet service at the node and is calculated as the packet *length/service rate* or 1Kbyte/2Mbit = 0.004 (s).

As the aggregated inter-arrival time (0.0025s) is less than the service time (0.004s) the packets will be queued (delayed) on the router.

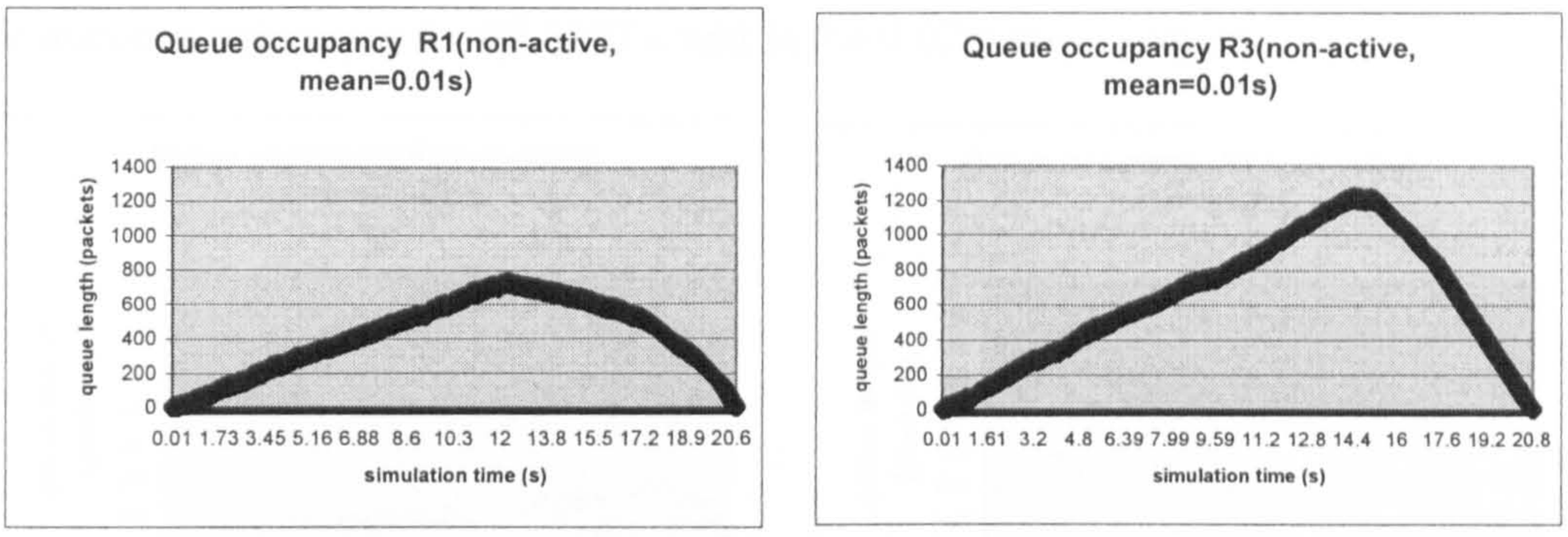
Table 8-3-3 shows the number of packets that are processed by router R1 within initial non-active transmission.

Table 8-3-3 Numbers of processed packets (non-active initial transmission, router R1)

Flow	Number of packets
from A to the group	1265
from B to A	1265
from C to A	1265
from D to A	1265

The total number of processed packets for R1 was 5060 packets and total number of processed packets for all routers in the network was 25300 packets. Later these results will be compared with results for the active approach.

Figure 8-2 illustrates the length of the queue on the router for the non-active approach. It shows that the router experiences a very high congestion as the aggregated inter-arrival time for packets is less than the router's service rate. The duration of congestion is 20.8108 s for router R1 and 20.8155 s for router R3.



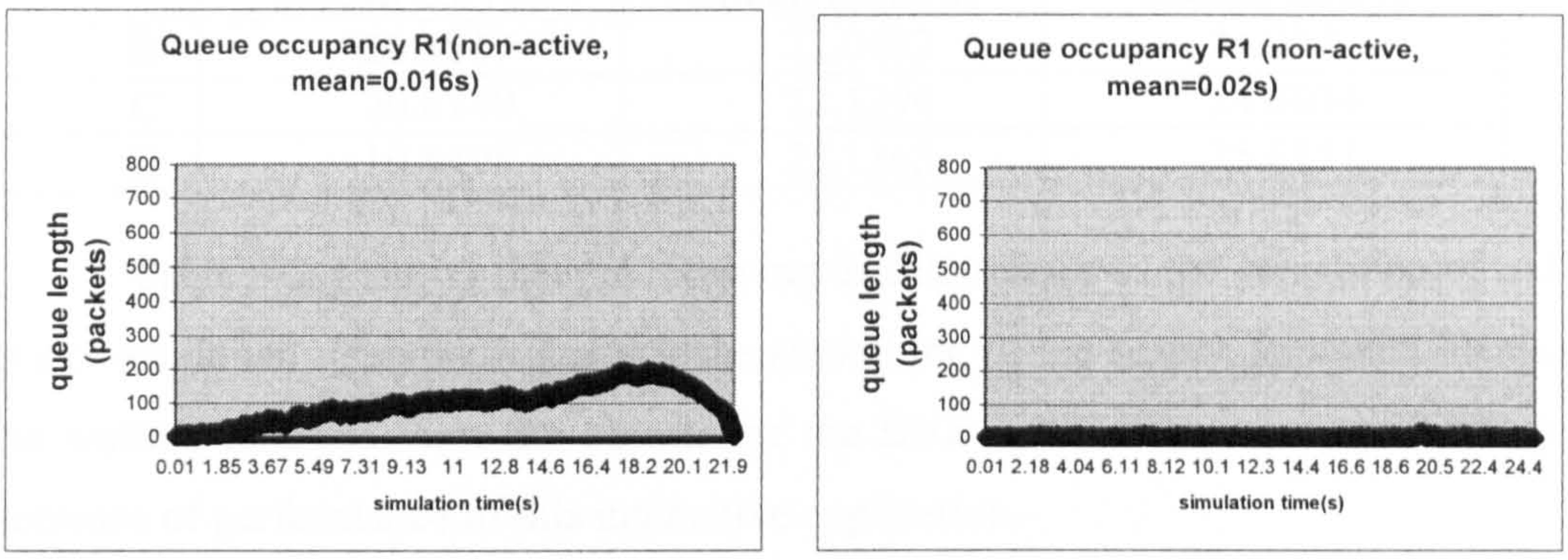
a) **b)**

Figure 8-2 Queue occupancy for routers R1 and R3 within the non-active scenario

8.3.2 Source Smoothing Scenario Results

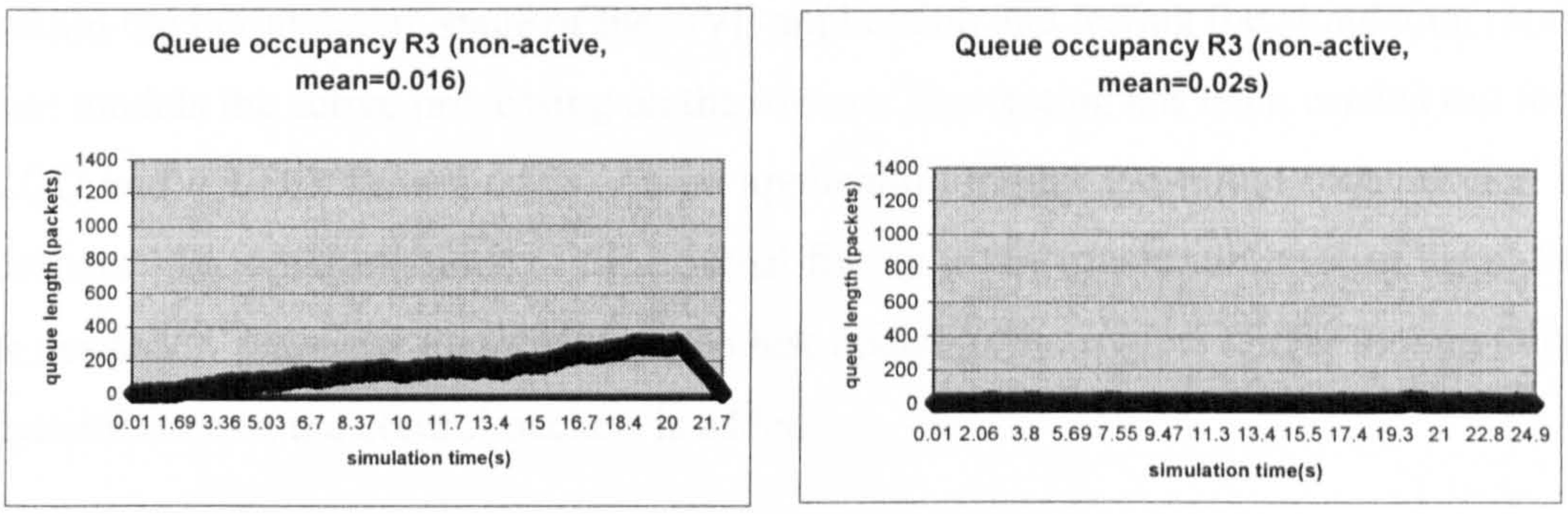
The results of the transmission of 3D data within a non-active scenario, which are presented in Section 8.3.1, illustrate that this traffic has a very bursty character. In these experiments an infinite size for the queue on the router has been used. However, real routers offer significantly less buffers for queuing packets. It is possible to decrease the size of the queue on routers and therefore to avoid congestion and discarding 3D geometry packets by limiting the source's output link bandwidth. In this case the source traffic has to be smoothed or slowed down. Within this scenario experiments the mean time between generation of packets has been increased and therefore the generation of packets by senders has been slowed down.

Figures 8-3a – 8-3b show the queue occupancy for router R1 with means equal to 0.016s and 0.02s that are compared with the queue occupancy for router R1 with mean equal to 0.01 s (Figure 8-2a). Figures 8-4a – 8-4b show the queue occupancy for router R3 with means equal 0.016s and 0.02s that are compared with the queue occupancy for router R3 with mean equal to 0.01 s (Figure 8-2b). These results show a considerable decrease in the length of the queue with increasing inter-generation time mean. However, the duration of congestion is increased, for example router R3 experiences congestion for 22.1207 s within the 0.016 s mean scenario and the queue is not empty for 25.6107 s within the 0.02 s mean scenario.



a) **b)**

Figure 8-3 Queue occupancy for router R1 with increased mean of inter-generation time



a) **b)**

Figure 8-4 Queue occupancy for router R3 with increased mean of inter-generation time

However, the transmission time and therefore the end-to-end delays are increased by applying the smoothing of the source approach. The example of times are those required to generate packets of the original 3D representation and receive these packets for all members of the multicast group are presented below in Table 8-3-4 for the host user A. The time to generate all packets by the user also increased with delaying the generation (e.g. 19.5488 s for the mean equal to 0.016 s and 24.436 s for the mean equal to 0.02 s).

Table 8-3-4 Reception time for different inter-generation means

User	Reception time(s) (mean 0.01 s)	Reception time(s) (mean 0.016 s)	Reception time (s) (mean 0.02 s)
B	20.1819	22.0682	24.937
C	20.8149	22.1294	24.6034
D	19.4498	22.1163	25.6151

Table 8-3-4 shows that the receiving time increases as the generation of traffic is slowed down. This indicates that the smoothing of the source approach increases the waiting times to view the objects for the DVE application users and leads to a decrease of performance of this interactive application.

8.3.3 Active Approach Results

The aim of these experiments is observation of the routers' queue occupancy results and hosts' reception times using the active approach for transmission 3D data within the initialization stage of the DVE application and testing the simulation model that models the active processing on the routers. The testing has been carried out for 3 LOD and 9 LOD flow models. As an application model the initial transmission by users of PM representations of the original file up to the maximum level of detail that is needed to represent them is used. To test updating the routers' LOD lookup tables transmission of the control packets is added.

8.3.3.1 Results for 3 LOD Scenario

For these series of experiments the 3 LOD flow model has been used. The experiments have been accomplished for the static case and the dynamic case (users move within the time of the initial transmission).

I. Static case results

Figure 8-5 shows initial placements of users that remain unchanged during the experiment. These initial placements define the mutual distances between users and therefore the level of detail required by each user. These requirements are reflected in the LOD lookup table. As users don't change their positions there is no need to update the content of the LOD lookup table.

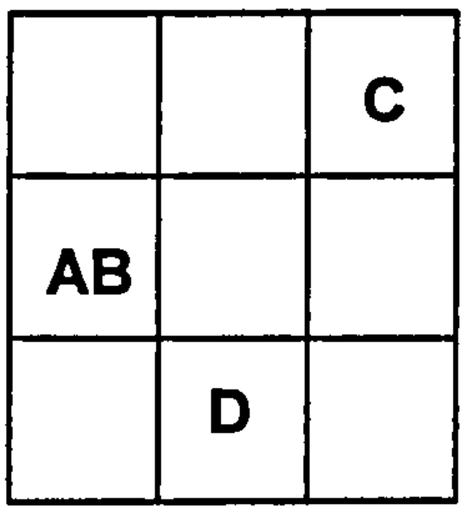


Figure 8-5 Initial placement of users in the 3x3 grid for the initialization stage experiments

The Table 8-3-5 summarizes reception times for all users in the group. For example for user A the required time to generate the packets that contain geometrical data up to LOD=2 was **12.6814**(s). User A sends data up to highest level of detail, as user B is placed in the same subspace as user A. Other users receive different amount of 3D data which depends on the required level of detail.

Table 8-3-5 Reception time for users in the group (initial stage, active 3 LOD, static)

Sender	Receiver	Number of received packets	Required LOD	Reception time (s)
A	B	1360	2	17.251
A	C	585	0	9.938
A	D	888	1	13.187
B	A	1360	2	17.249
B	C	585	0	8.281
B	D	888	1	13.368
C	A	585	0	10.715
C	B	585	0	8.236
C	D	585	0	8.149
D	A	888	1	15.107
D	B	888	1	14.988
D	C	585	0	8.511

In comparison with the non-active approach the time to receive the required 3D representation decreases for the receivers that are placed far from the sender in the virtual world. For example *user C* receives all the 3D data that is required in order to view other users' representations within only 8–9(s) instead of the 19–20(s) that was needed to receive the non-progressive representation. The decrease in the reception time is achieved by the active filtering of unnecessary batches for higher LODs.

Also if users are placed in the same subspace of the grid (e.g., user A and user B) and the full representation (all batches) is needed to be transmitted between them, the simulation results show that the time to receive all required packets decreases in comparison with the non-active approach. The time to receive three batches of the 3 LOD flow model (1360 packets) from user A to user B for the active approach is **17.251** (s), but the time to receive non-progressive representation (1265 packets) from user A to user B for the non-active approach is **20.1819** (s) (see Table 8-3-1).

This could be explained by the decrease in the total number of packets that are needed to be transmitted between members of the multicast group (*users A, B, C,* and *D*). In this case the routers' aggregated inter-arrival time increases and therefore, the packets could be processed with less queuing delay.

As the filtering mechanism on each router doesn't replicate unneeded batches of packets to the next node the number of packets that are transmitted between multicast group members is reduced and therefore the saving of bandwidth is achieved.

The total number of packets that have been processed on all routers is **18168** packets. In comparison with the non-active approach we achieve a decrease in the total number of transmitted packets (e.g. **25300** packets for non-active approach and **18168** packets for active 3 LOD approach).

The router's performance metric, queue occupancy, has been measured within these experiments. By smoothing out the peak in the queue occupancy (or the queue length) we decrease the maximum length of queue that built up due to transmission of the geometry packets. The results in Figure 8-6 show that we achieve considerable decrease in the maximum queue length for the busier router R3 (from more than **1200**

packets for the non-active to about **800** for the active 3 LOD approach). In our experiments we use infinite queues but in the real scenario when queue length is limited, increasing the number of packets in the queue leads to discarding of packets by the router and therefore high losses in transmitted data. Hence, the capability to reduce the queue's length improves the reliability of transmission of 3D content within the DVE application.

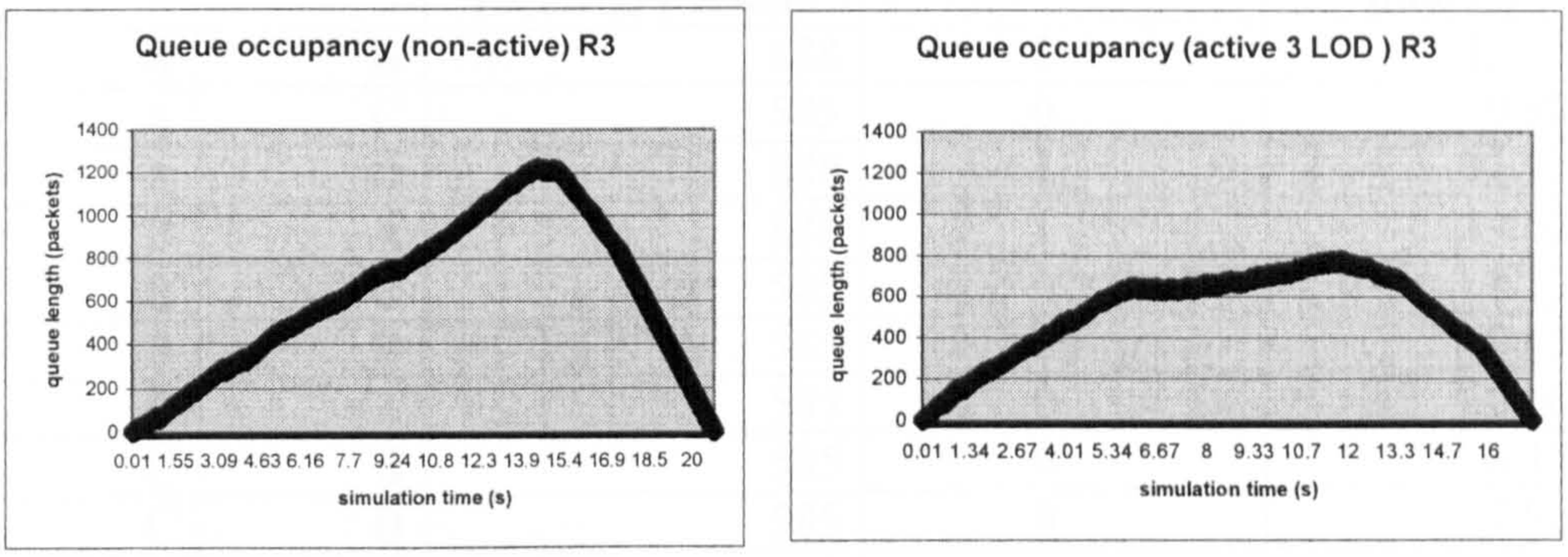


Figure 8-6 Queue occupancy for non-active and active 3 LOD experiments for router R3

The congestion time is decreased within the 3 LOD scenario from 20.8155 s to 17.2458 s for router R3.

II. Dynamic case results

The aim of this experiment is test of the accuracy of the LOD lookup table updating process.

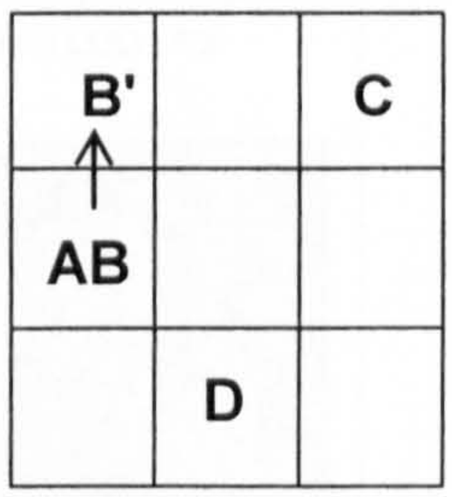


Figure 8-7 User B moves in the virtual world

In this experiment it is assumed that user B moves before the initial transmission is completed. It causes a change in the mutual distances between user B

and other users in the environment. After the movement user B needs LOD1 of user A's representation and LOD0 of user D's representation.

Table 8-3-6 summarizes reception times for all users in the group for the dynamic case.

Table 8-3-6 Reception time for users in the group (initial stage, active 3 LOD, dynamic)

Sender	Receiver	Number of received packets	Required LOD	Reception time (s)
A	**B**	888	1	12.018
A	**C**	585	0	9.905
A	**D**	888	1	11.926
B	**A**	888	1	14.628
B	**C**	585	0	8.223
B	**D**	585	0	10.656
C	**A**	585	0	10.668
C	**B**	585	0	8.178
C	**D**	585	0	7.962
D	**A**	888	1	14.649
D	**B**	585	0	11.007
D	**C**	585	0	8.333

Within the dynamic scenario the movement of user B farther from other users leads to the additional decline in the number of transmitted packets. Figure 8-8 illustrates how the maximum length of the queue on router R3 decreases due to the decline in the number of transmitted packets (from about **800** packets for static experiments to less than **700** for dynamic experiments).

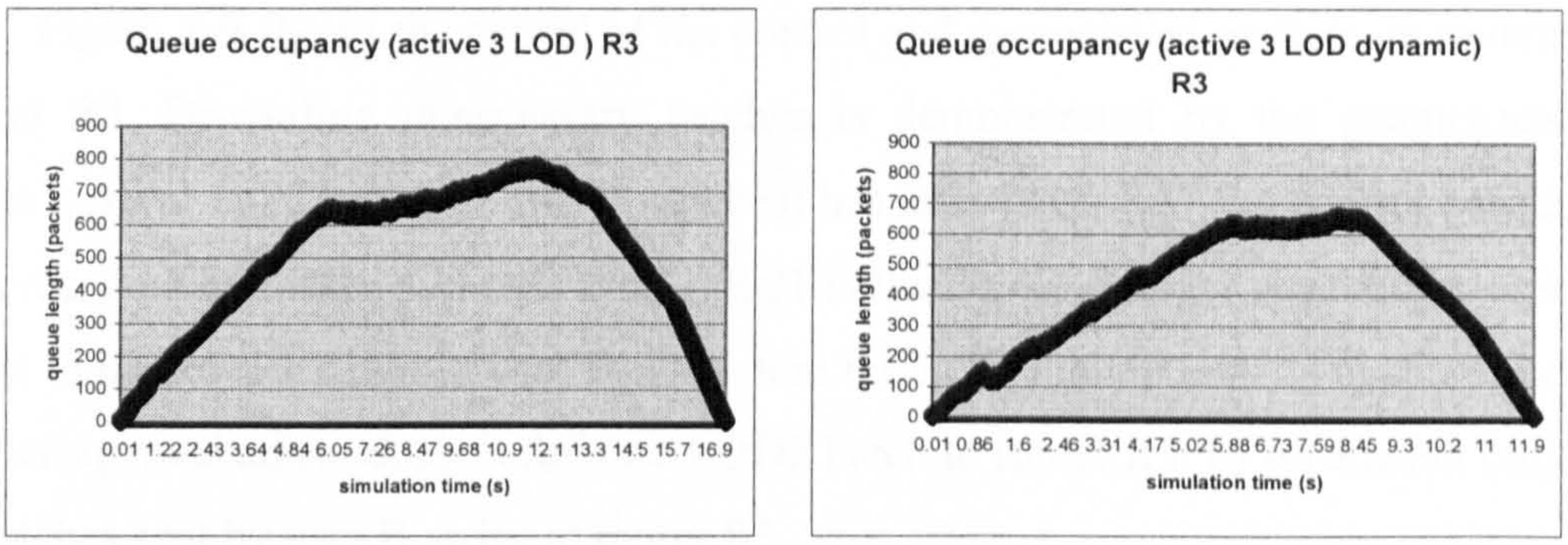
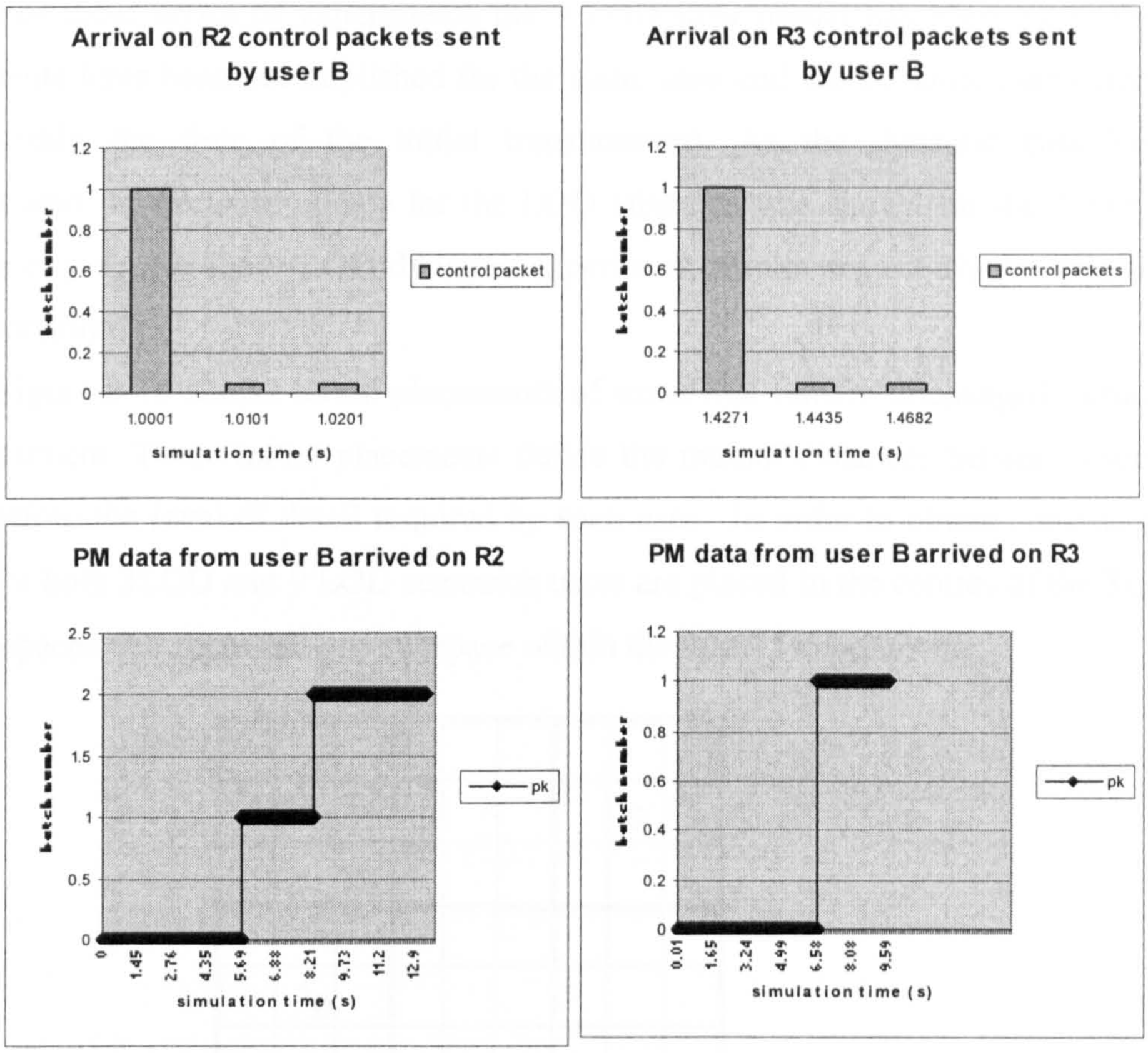


Figure 8-8 Comparison of queue's length for static and dynamic scenarios for 3 LOD case

The next graphs demonstrate changes in the LOD lookup table on the router caused by movement of the user (Figure 8-9).



a) **b)**

Figure 8-9. Changes in LOD lookup table on routers

Figure 8-9 shows the arrival of the control and geometrical packets on routers R2 and R3. Discarding unnecessary batches is demonstrated by the geometrical packets' arrival. a) The whole file (3 batches) has arrived on R2. The control packet that is received by router R2 at the time 1.0001 indicates the changes required by user A level in the level 1 detail of user B. Therefore the LOD lookup table is updated and the filtering mechanism doesn't forward LOD2 batch to router R3. b) As a result only two batches sent by user B arrive at router R3.

PM data received by members of the group shows that every router dynamically updates its LOD table in response to the movement of user B.

8.3.3.2 Results for 9 LOD Scenario

For these series of experiments the 9 LOD flow model has been used. The experiments have been accomplished for the static case and the dynamic case (users move within the time of the initial transmission). As the dynamic case has demonstrated the accurate update for the LOD tables as was shown for the 3 LOD dynamic experiment the 9 LOD dynamic experiment results are not shown here to avoid repetition.

Figure 8-10 shows initial placements of users that remain unchanged during the experiment. These initial placements define the mutual distances between users and therefore the level of detail required by each user. In order to obtain consistent results for both 3LOD and 9 LOD scenarios users are placed in the centres of the 3x3 grid subspaces that represent one subspace within the 9 LOD experiments.

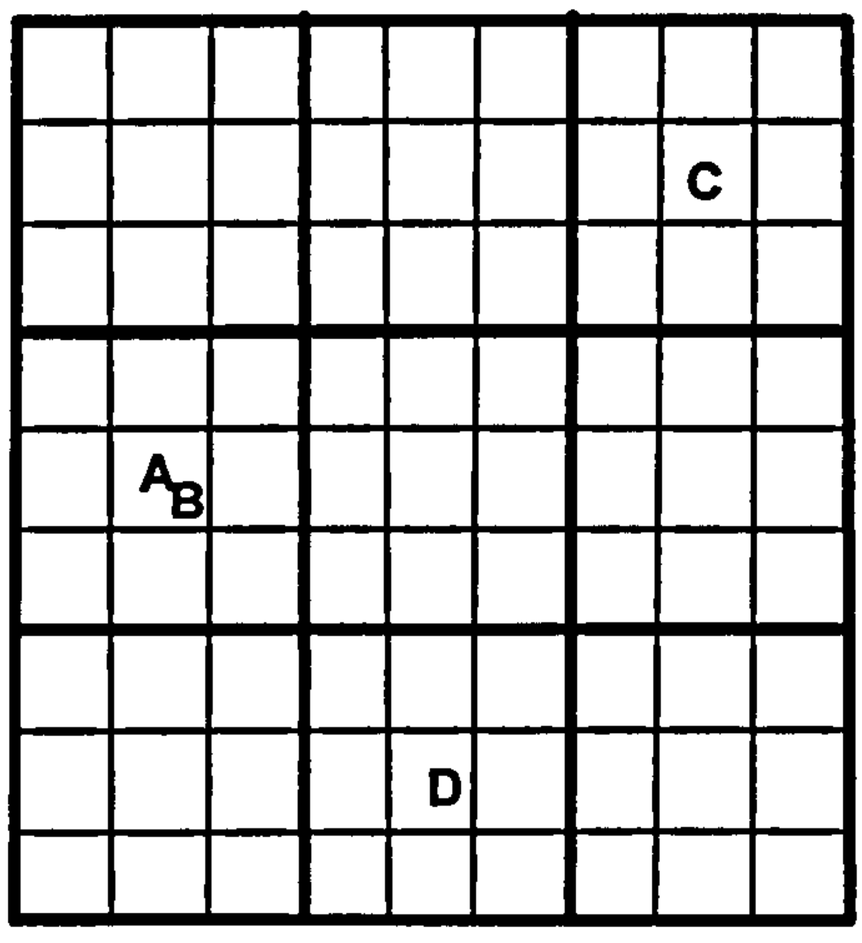


Figure 8-10 Initial placements of users in the 9x9 grid those are consistent to the initial spatial relationship of users in 3 LOD experiments

Table 8-3-7 summarizes reception times for all users in the group for the static variant of the 9 LOD case.

Table 8-3-7 Reception time for users in the group (initial stage, active 9 LOD, static)

Sender	Receiver	Number of received packets	Required LOD	Reception time (s)
A	**B**	1495	8	15.292
A	**C**	209	2	3.692
A	**D**	517	5	7.163
B	**A**	1495	8	15.280
B	**C**	209	2	2.989
B	**D**	517	5	7.245
C	**A**	209	2	3.503
C	**B**	209	2	2.677
C	**D**	209	2	2.664
D	**A**	517	5	7.816
D	**B**	517	5	7.858
D	**C**	209	2	2.870

Results presented in Table 8-3-7 show that in comparison with the 3 LOD scenario the end-to-end delay or waiting time to receive the 3D data up to the required level of detail decreases for remote users. For example, user C's time to receive all data needed from user A within the 3 LOD scenario was **9.938** (s) (see Table 8-3-5), but for the 9 LOD scenario this number is **3.692** (s) (Table 8-3-7).

In comparison with the active approach where less levels of detail are used (3 LODs) more saving of bandwidth is achieved. The entire number of transmitted 3D content packets in this experiment is **13744** packets that is less than it was transmitted in the 3 LODs scenario (**18168** packets). It is possible to say that the batches' sizes of 9 LODs PM representation could cause the decrease in the number of transmitted packets. Figure 8-11 shows the flow model of objects that consists of 9 batches. The higher (more detailed) batches contain more packets than the lower batches. It is connected to the simplification algorithm which chooses less and less edges to be collapsed as it performs simplification for more crude models (meshes with fewer numbers of the vertices). Using the 9x9 grid allows us to describe the mutual distances between users more precisely and avoid transmission of higher batches that contain significantly more packets than lower batches.

Despite the larger size of the complete PM representation (1495 packets) the results of the 9 LODs scenario show that increases in the LOD lead to additional

saving of bandwidth within the initialization stage of the application and, therefore, decreases in waiting times to receive 3D content for DVE participants.

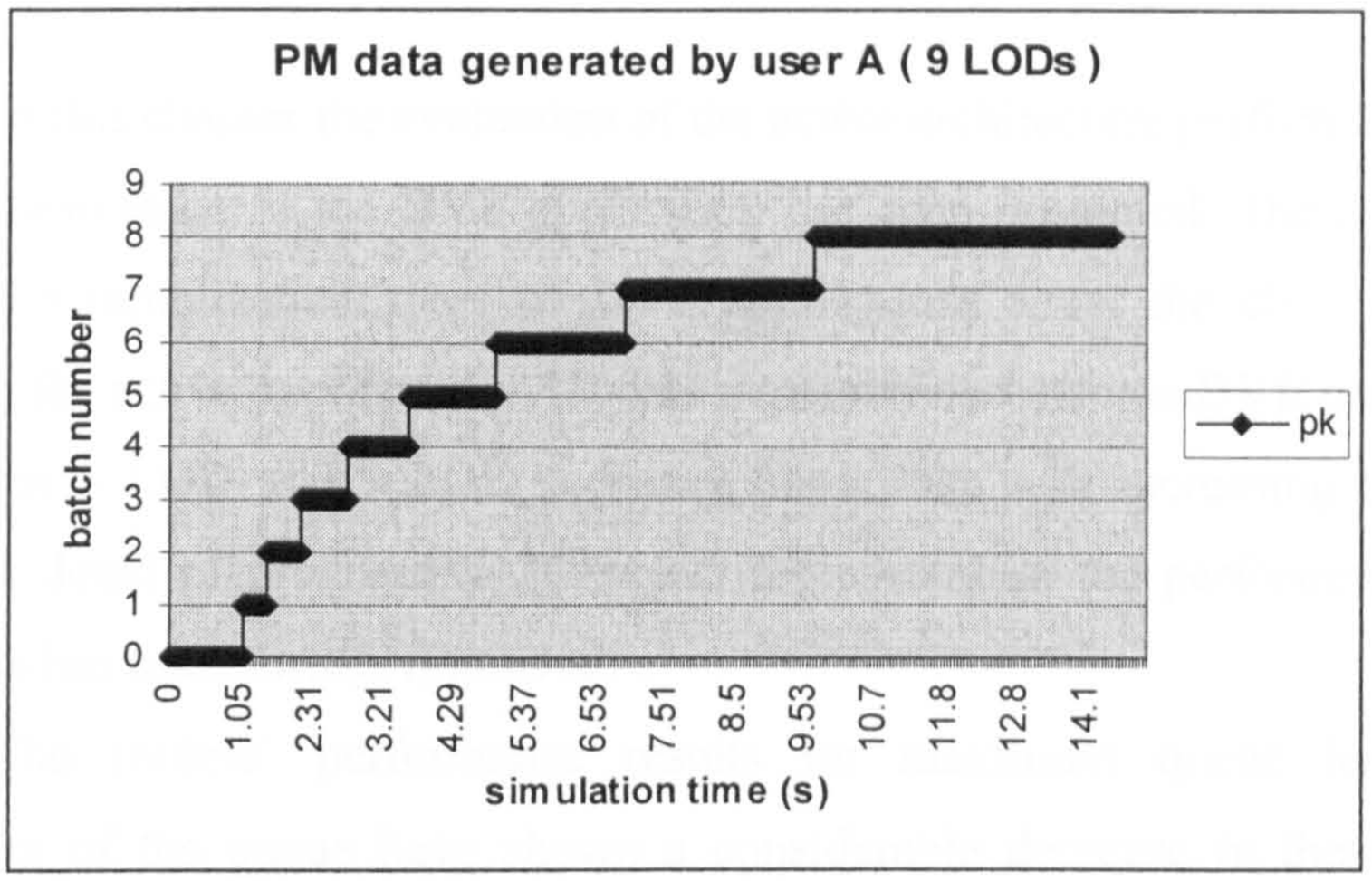


Figure 8-11 PM data packets generated by users within 9 LODs experiment scenario

The queue occupancy results have been obtained for the 9 LOD initialization stage. Results show (Figure 8-12) that we achieve a considerable decrease in the maximum number of packets in the queue in comparison with the 3 LOD scenario (from about **800** packets for 3 LOD approach to about **350** for active 9 LOD approach for router R3).

The congestion time is decreased from 17.2458 s for 3 LOD case to 15.2885 s for 9 LOD case.

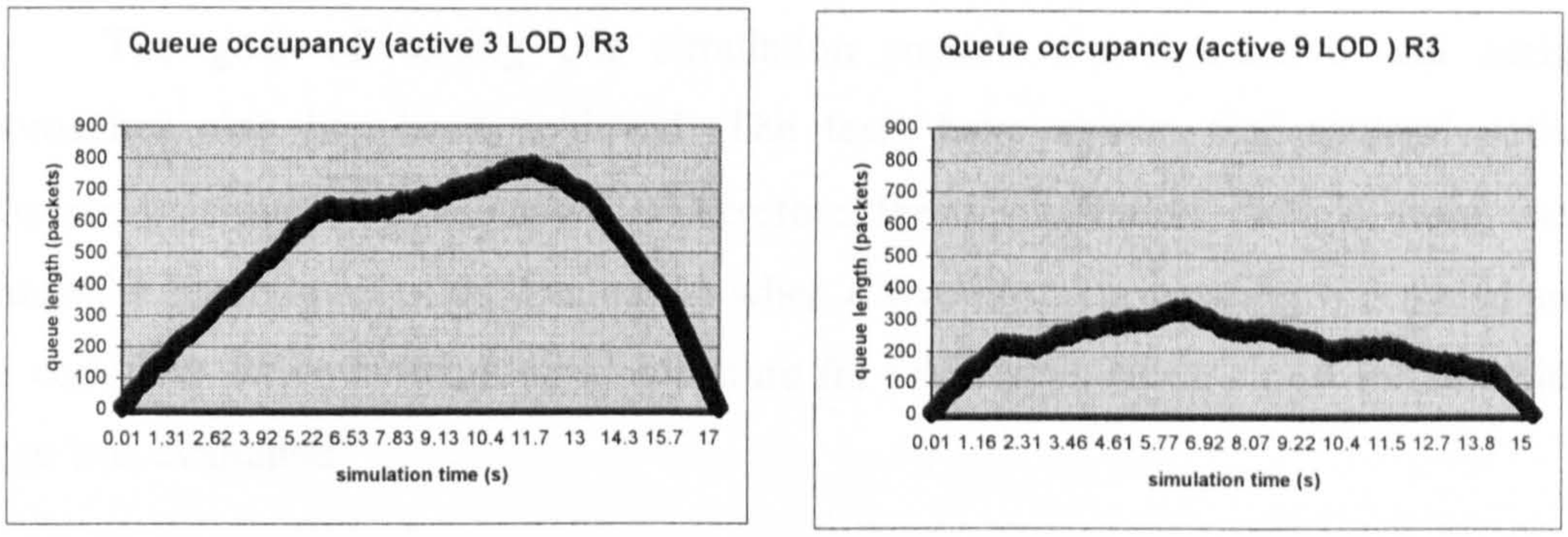


Figure 8-12 Queue occupancy for 3 LOD and 9 LOD experiments (router R3)

8.4 Discussion of Results

In this chapter the evaluation of the active architecture performance within the initialization stage of the DVE application has been presented. The results obtained within the initialization stage of DVE applications show the clear advantages of applying the active approach for 3D data transmission between DVE participants. The results for 3 LOD and 9 LOD scenarios show that with increasing the number of levels of detail of progressive 3D object representation the performance on routers and hosts has been further improved.

The routers' performance results for maximum queue length and the utilization of the queue have shown a considerable decrease in these metrics that allows congestion on the router to reduce.

The hosts' performance results for reception time of 3D data have also decreased with the active approach. The reduced load on the network that has been achieved by the active approach allows decreasing reception times and therefore times to view the 3D objects by DVE participants. The alternative approach that slows down the transmission of 3D data packets by the hosts (the source smoothing) has also been examined. It was found that with increase in the mean inter-generation time the queue length declines dramatically, but the congestion time or the time when the queue is not empty (e.g. 0.02 s mean results) increases. Therefore, the reception times increase and the responsiveness of the application decreases.

The goal of testing our simulation models for non-active and active approaches also has been achieved. The tests have shown that routers' active processing is working accurately. Therefore these simulation models were used within the second part of the evaluation when active hosts' processing was added and the complete active networking architecture for DVE application's post initialization stage was evaluated.

Chapter 9

Evaluation of the Active Approach within the Post Initialization Stage of the Application

This chapter presents the evaluation of the active networking architecture within the post initialization of the DVE application. After filtering active processing on routers had been tested and comparative results for active and non-active approaches had been obtained for the initialization stage of the application the complete architecture model that includes an active router's and an active host's components has been examined. In these experiments the application model of users' movements has been used to construct the transmission pattern of 3D streams that is driven by the sequence of movement events of every user.

First, the performance evaluation for the random scenario of movements is discussed. Within this basic scenario the comparison between non-active and active approaches is carried out for the post initialization stage of the application.

Second, the evaluation of the proposed architecture has been continued with changing the experiments' parameters. Comparison between routers' performances for different application models that present different patterns of users' movements has been accomplished. As experimental parameters the available bandwidth for the 3D DVE active packets class on the router and the duration of the experiment have been used. Then increasing the number of users has been analysed for all application models.

9.1 Comparison between Active and Non-active Approaches' Performance for the Basic Scenario Experiment

The objectives of these experiments were to provide confidence that the complete model of the active networking architecture was working properly and to compare the non-active and active approaches within the post initialization stage of the application.

9.1.1 Simulation model

The simulation model comprises two basic elements, hosts and routers.

The host and router elements have been implemented for both active and non-active approaches to be able to simulate application behaviour for both cases.

The active host elements produce control packets and geometrical-information packets, the generation of both being driven by the simulated interaction of the user within the virtual environment that is provided by the application model. For the basic scenario user interaction is simulated through events and actions that are generated by a simple random process. Within this scenario users move randomly. Every user explores different parts of the virtual world. Some of the users leave and join the group during the experiment.

Every host includes elements that simulate the following processes: 1) control packet generator; 2) geometrical packet generator including single batch generators and sequences of batches generators; 3) LOD lookup table search and update; 4) destination process. The control packet generator uses an application model's scripts of movements to send the control packets that update the LOD tables on routers and hosts. The host performs periodical checks of its LOD lookup table. If changes in the mutual distances between users occur within the given time slot and the LOD lookup table check will show the decrease in such distances the corresponding geometrical generators will be invoked. As there is the possibility that several users would move closer to the user and the distances to each of these approaching users might differ, several batch generators will be invoked in one time slot. When the event of joining the group by a new user is detected one of the sequence batch generators (maximum required by the group LOD) will be invoked. On arrival of control packets from all users in the group (including the given host) the LOD table will be updated. The destination process simulates the reception of the geometrical packets. The numbers of packets received, time of reception and memory requirements on the host are monitored by the destination process. The host models for 3 LOD and 9 LOD active approaches differ in the geometrical packet generators. The 3 LOD host model

includes only three single batch generators and three sequence batch generators whereas the 9 LOD model contains nine single batch generators and nine sequence batch generators. Also the number of generated packets is defined by the flow models for the 3 LOD and 9 LOD case.

The host model for the non-active approach includes the geometrical packet generator that generates the sequence of packets that represent the original single-resolution non-progressive 3D model and the destination process. In addition to the initialization stage the process that simulates leaving and joining events of multicast group members is also included. Within the post initialization stage for the non-active case retransmission of the original mesh will be performed on join events. When a given user leaves the group its host will not be able to transmit data to the group when a new user appears.

The router model used within the initialization stage is enhanced by the ability to filter the sequence batch streams using the information about the changes in the multicast group and forward the sequence of batches exactly to the just joined user.

To simulate non-active and active router's processing the simulation node models for both approaches have been developed. Non-active node presents the ordinary router functionality and includes only the multicast routing of the geometrical packets. However, the process that simulates the leaving and joining events of the multicast group members is also implemented to adjust the multicast trees. For both the active and non-active approach static trees routing is used.

Examples of the OPNET simulation models for networks, nodes and processes used to evaluate active approach within application's runtime are included in Appendix III.

9.1.2 Experiment Description

For this basic experiment the network configuration and the experimental parameters used for the initialization stage experiments have been used (see Section 8.2).

Figure 9-1-1shows the initial placements of users for this series of the experiments. The placements are presented in relation to the 3x3 grid (thick boundary lines) and 9x9 grid (thin boundary lines).

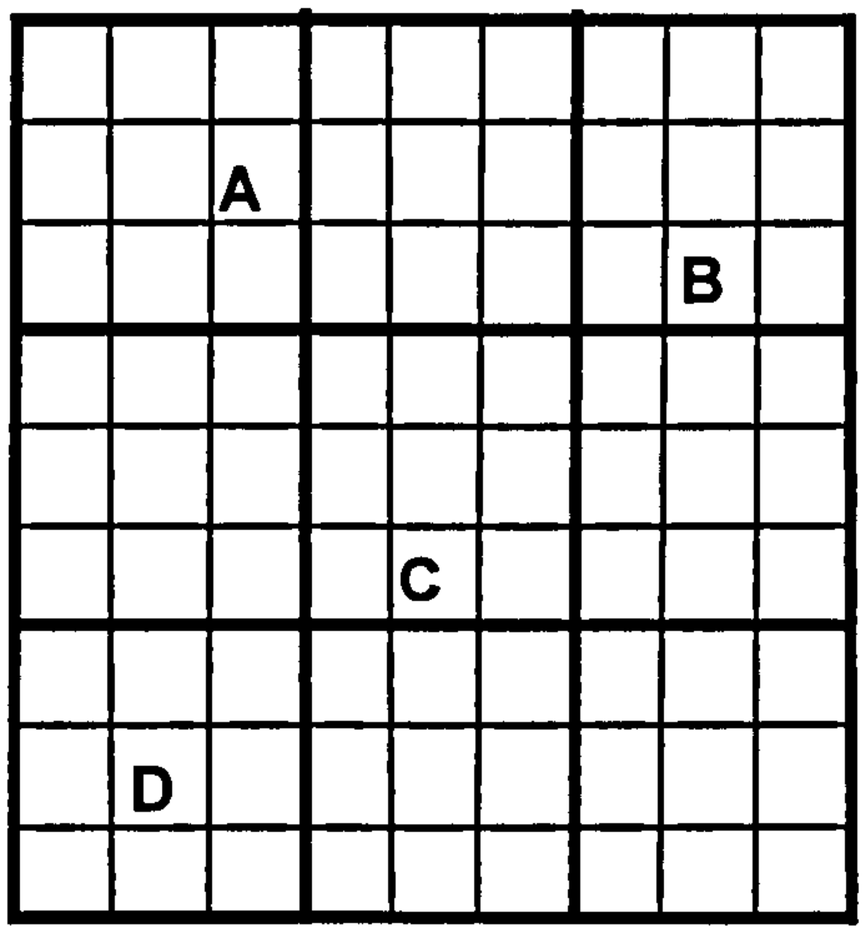


Figure 9-1-1 Initial placements of users for basic scenario experiments

Computer generated scripts of movements that represent the *random* application model have been used for every participant. The set of initial placements together with the set of movements that take place over a predefined interval of time (20 minutes) will define an application scenario. Figure 9-1-2 shows the fragment of user C's movement script within 3x3 grid. The addresses of users are: 1 for user A, 2 for user B, 3 for user C, and 4 for user D. The example shows the events of leaving (LOD=-1) and joining the group beside with events of movement.

Movements of user C within the 3x3 grid Number of time slots: 20			
Sender	**Receiver**	**LOD**	**Time (s)**
3	1	0	61
3	2	0	62
3	4	2	63
3	1	2	317
3	2	1	318
3	4	1	319
3	1	0	393
3	2	0	394
3	4	1	395
3	1	1	425

3	2	0	426
3	4	1	427
3	1	0	564
3	2	0	565
3	4	2	566
3	1	-1	687
3	2	-1	688
3	4	-1	689
3	1	0	727
3	2	1	728
3	4	2	729
3	1	-1	867
3	2	-1	868
3	4	-1	869
3	1	0	969

Figure 9-1-2 Script example for user C

The SVG graphical representation of the *random* application model for 20 minutes is presented in Section 7.2.2.

9.1.3 Results from Simulation.

The experiments have been carried out for 20 minutes for non-active, 3LOD and 9LOD active approaches. Evaluation of these approaches' performance for different movement scenarios has been based on testing several QoS parameters for nodes (routers) and hosts.

I. Testing of router's QoS parameters.

The queue occupancy results are presented for routers R1, R3, and R4.

Figures 9-1-3 – 9-1-5 show the plotted queue occupancy results and present all three approaches (non-active, 3LOD and 9LOD active).

1) Maximum queue length.

Table 9-1-1 summarises the maximum values of the queue length that have been admitted on routers and presented by the queue occupancy graphs in Figures 9-1-3 – 9-1-5.

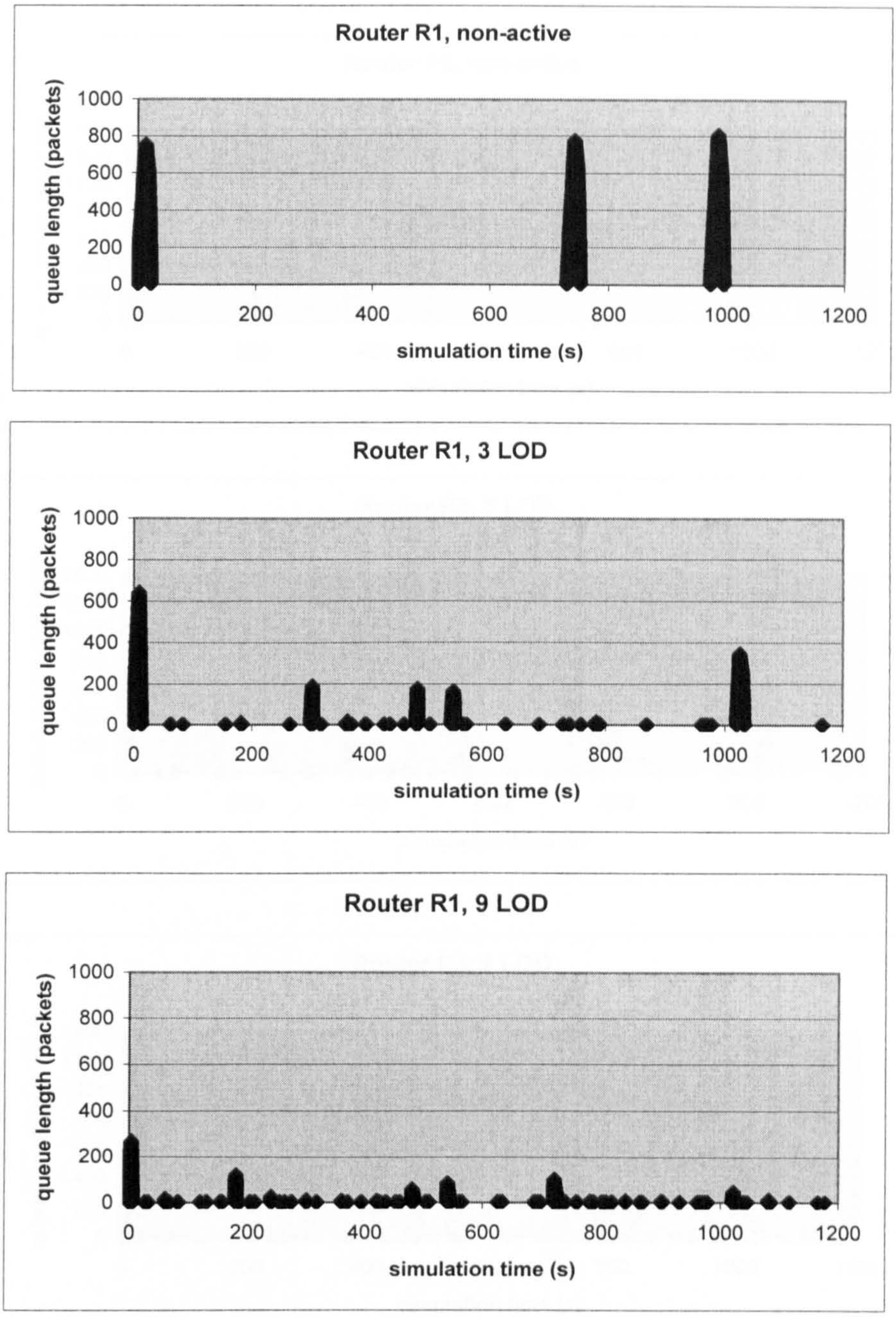


Figure 9-1-3 Queue occupancy results for router R1 (random scenario, 20 minutes)

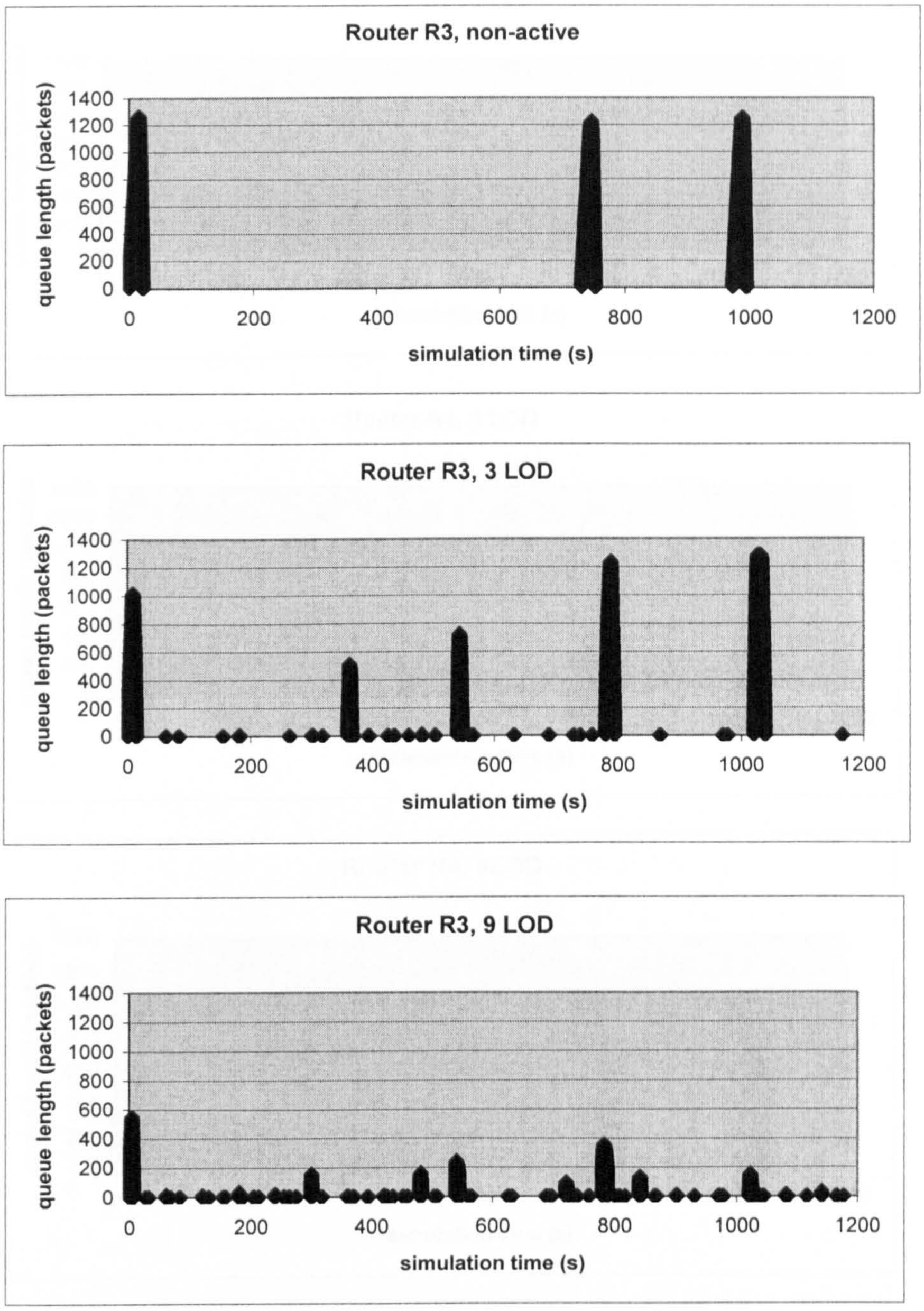


Figure 9-1-4 Queue occupancy results for router R3 (random scenario, 20 minutes)

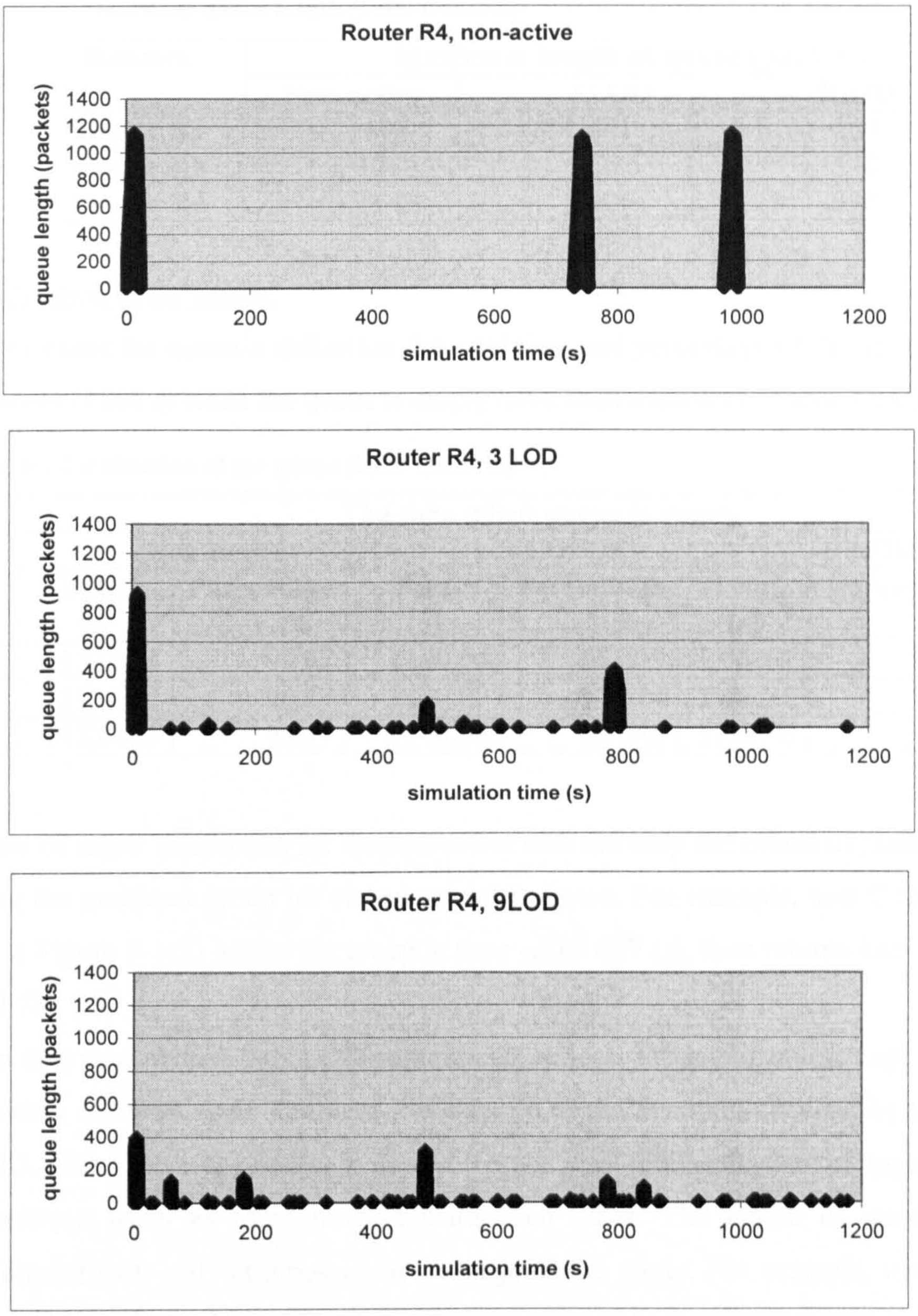


Figure 9-1-5 Queue occupancy results for router R4 (*random* scenario, 20 minutes)

Table 9-1-1 Maximum queue length (basic scenario)

Routers	Maximum length of queue (packets)		
	Non-active	3LOD	9LOD
Router R1	807	641	269
Router R3	1266	1285	548
Router R4	1149	926	398

2)Utilization of the queue.

To evaluate the queue's utilization the total time and percentage of the time of the experiment (1200 s) when the queue is empty have been measured (Table 9-1-2).

Table 9-1-2 Utilization of the queue (basic scenario)

Routers	The time when queue is empty					
	Non-active		3 LOD		9 LOD	
	Total (s)	Percentage %	Total (s)	Percentage %	Total (s)	Percentage %
R1	1133.689	94.47	1140.06	95.005	1155.61	96.30
R3	1133.685	94.47	1130.09	94.174	1146.26	95.52
R4	1133.689	94.47	1140.56	95.046	1152.96	96.08

Events of users' interaction for the non-active case are only the events of joining and leaving the multicast group (or virtual world) by users. For example, user C (see the script in Figure 9-1-2) leaves the group at time equal **687** (s), then returns back at time equal **727** (s).

The first peak within the 3LOD experiments reflects the initialization stage of the application when all users multicast PM data up to the maximum needed by the group LOD. Then several smaller peaks correspond to retransmission of higher batches between users as users move toward each other. The length of queues increases dramatically only at times when a user joins the group. For example, when user C joins the group at times **727** (s) and **969** (s) router R3 experiences the longest queues (1285 packets). In comparison with the non-active results the initialization stage gives lower peaks as users multicast not whole representations but only the sequence of batches up to the maximum needed by the group LOD. The post initialization stage for the active scenario demonstrates lower peaks in the queue length on routers R1 and R4, however, router R3 still experiences high congestions in the 3 LOD active approach.

The overall savings in bandwidth achieved by the 3 LOD active approach could be examined by studying the number of sent and received packets during the experiment. The number of sent and received packets within the group for 3 LOD is lower in comparison with the non-active approach.

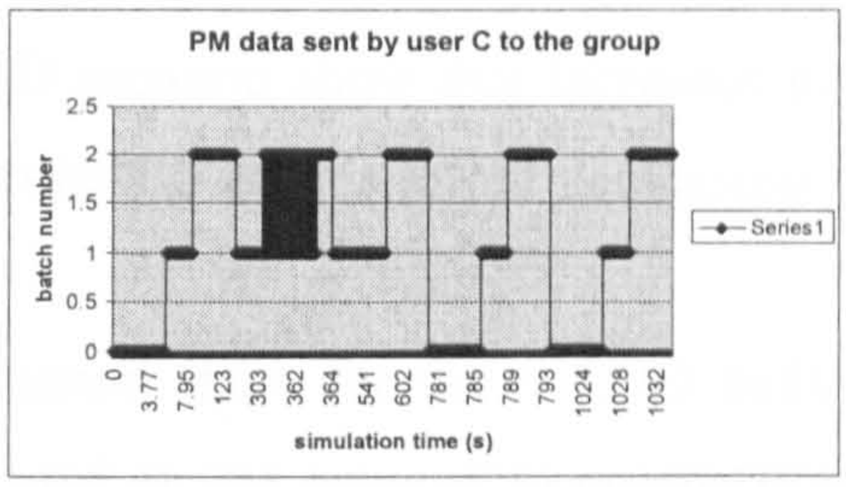
Table 9-1-3 Example of numbers of transmitted packets for non-active and 3 LOD active approach

Packets	Non-active	3LOD active
PM data sent by user C	21505	6236
PM data received from user C by user A	21505	3439
PM data received from user C by user D	21505	5461

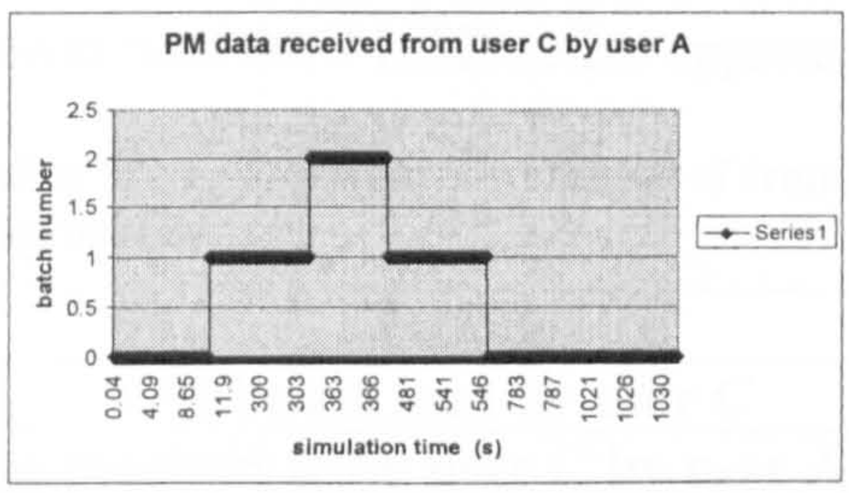
Figure 9-1-6 shows the example of sent and received packets for the 3 LOD case. At the times when user C joins the group it sends the sequence of batches up to the maximum required by the group. As user A and user D need different LODs of user C's representation at these times active filters on routers forward only the necessary batches from the sequence sent by C to the different receivers. When user C moves in the virtual world (e.g. from time equal to 60 (s) to time equal 687 (s)) only single batches are sent.

For the 9LOD experiment results the first peak also reflects the initialization stage of the application when all users multicast PM data up to the maximum needed by the group LOD. In comparison with the 3LOD results (See Figures 9-1-3 – 9-1-5) this peak is smoothed out and just about half of the same peak for the 3LOD scenario (for example on router R1 it is only about **270** packets for 9LOD instead of **600** packets for 3LOD scenario). Also all the following peaks that correspond to retransmission of 3D data within the runtime of the application are smoothed out.

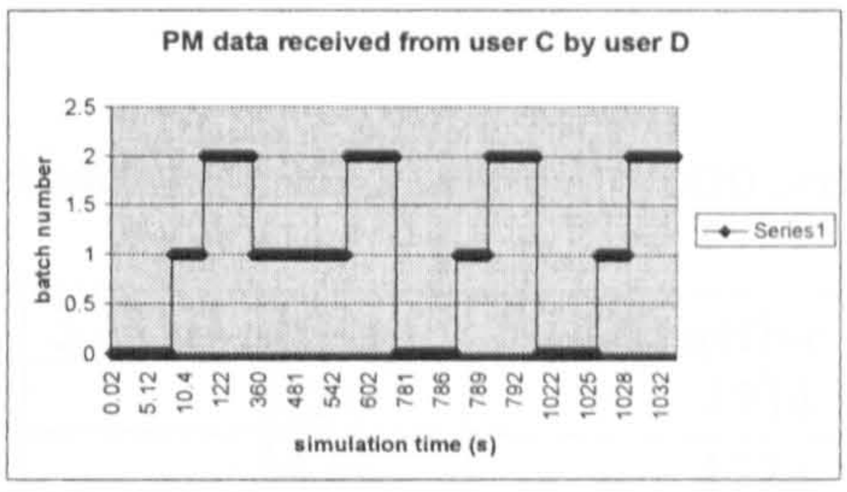
It is possible to say that the batch sizes of the 9 LOD PM representation could cause the decrease in the number of transmitted packets. The higher (more detailed) batches contain more packets than lower batches. It is connected to the simplification algorithm that chooses less and less edges to be collapsed as it performs simplification for more crude models (meshes with fewer numbers of vertices). Using the 9x9 grid allows us to describe the mutual distances between users more precisely and avoid transmission of higher batches that contain significantly more packets than lower batches.



a)

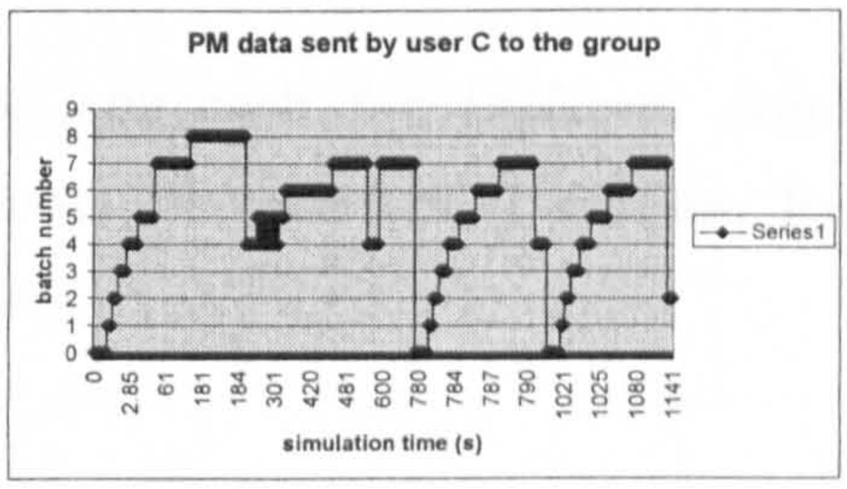


b)

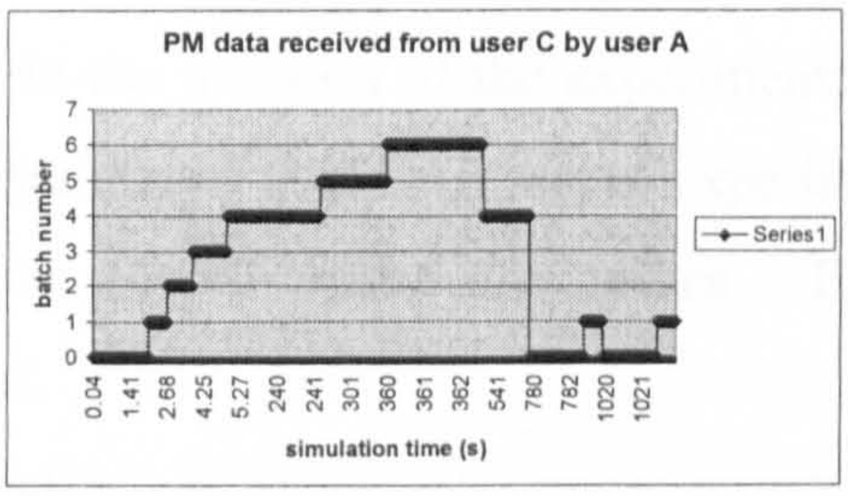


c)

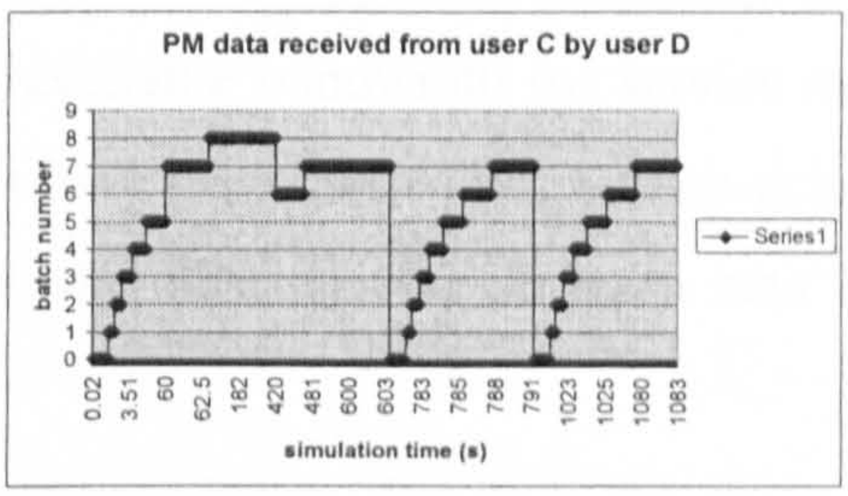
Figure 9-1-6 Example of PM data sent and received within active 3LOD scenario



a)



b)



c)

Figure 9-1-7 Example of PM data sent and received within active 9LOD scenario

Despite the larger size of the complete PM representation (1495 packets) the results of the 9 LOD scenario show that increases in the LOD number leads to additional saving of bandwidth and therefore decreasing the length of the queue on routers.

The post initialization stage for the 9 LOD active scenario demonstrates a further decrease in the load on the network in comparison with the 3 LOD scenario almost all the time of the experiment.

The number of sent and received packets within the group is reduced in comparison with the 3 LOD active approach.

Table 9-1-4 Example of numbers of transmitted packets for the 3 LOD and 9 LOD active approaches

Packets	3 LOD active	9 LOD active
PM data sent by user C	6236	4934
PM data received from user C by user A	3439	1228
PM data received from user C by user D	5461	4147

Figure 9-1-7 demonstrates PM data that are sent and received within the active 9LOD scenario experiment.

9.2 Evaluation of Active Approach for Different Application Models with Changes to the Experimental Parameters (Duration, Bandwidth, Number of Users)

To evaluate the proposed active approach three sets of the experiments have been carried out. These experiments evaluate the active approach with respect to changes in the duration of the experiment, the available bandwidth (or service rate) and the number of users. All sets of experiments have been carried out for non-active, 3LOD and 9LOD active approaches. Below the descriptions of these sets are presented.

Set 1.

Number of users: four.

The application models: *random, far, centre* and *join*.

Duration: 20 minute and one hour.

Bandwidth: 2 Mbit/s.

In this set we observe: a maximum queue occupancy, queue utilization, host memory usage.

Set 2.

Repeat the set 1 but for different service rates: 2.5, 3, 3.5, 4 Mbit/s.

In this set we observe maximum queue occupancy.

Set 3.

Repeat for 2 Mbit/s case for 5, 6, 7, 8 users.

In this set we observe maximum queue occupancy and initial reception time on host.

9.2.1 Set 1 of Experiments

The aim of these experiments was to find the changes in performance of non-active and active approaches that would be caused by diverse movement patterns of participants of DVE within different scenarios of movement and the longer simulation run of the experiment. The experiments have been carried out for four application models: *random, far, centre,* and *join* (see Chapter 7).

The full coverage of users' movements within *random, far, centre,* and *join* scenarios for 20 minutes and one hour is shown in Figure 9-2-1a - 9-2-1d.

Comparison of one-hour movement patterns with 20 minute movement patterns demonstrates the continuation of similar behaviour of users for the longer duration of the experiment. The less transparent colour of rectangles for 1-hour visualisation is defined by the fact that users visit the same subspaces of the grid more times within one-hour scenario than within the 20 minute scenario (see Chapter 7). For the *centre* scenario users are almost restricted from crossing the central subspace in the 3x3 grid, but they are moving randomly within this subspace. For the *far* scenario users tend all the time to move farther from each other (to occupy the 9x9 grid corner subspaces). SVG visualization also shows a similar movement pattern for the one-hour scenario

for the *join* and *random* models. The initial placements of users which have been used for the basic random scenario (see Figure 9-1-2) are also used for all other scenarios.

Evaluation has been based on examining several QoS parameters for nodes (routers) and hosts.

I. Testing of router's QoS parameters.

The queue occupancy results are presented for one of the nodes – router R3.

Figures 9-2-2 – 9-2-5 show the plotted queue occupancy results of four different movement scenarios and present all three tested approaches (non-active, 3LOD and 9LOD active) for every scenario for 20 minutes (a) and for one hour (b) experiments. The spikes have been made thick.

1) Maximum queue length.

Table 9-2-1 summarises the maximum values of the queue length that have been admitted on router R3 and presented on the queue occupancy graphs in Figures 9-2-2 – 9-2-5.

Table 9-2-1 Maximum queue length (Set 1 of experiments)

Move-ments' pattern	Maximum length of queue (packets)					
	Non-active		3 LOD		9 LOD	
	20 min	1 hour	20 min	1 hour	20 min	1 hour
random	1266	1266	1285	1285	548	1015
far	1221	1221	1019	1019	548	548
centre	1221	1221	1019	1019	864	905
join	2251	2251	1041	1041	560	560

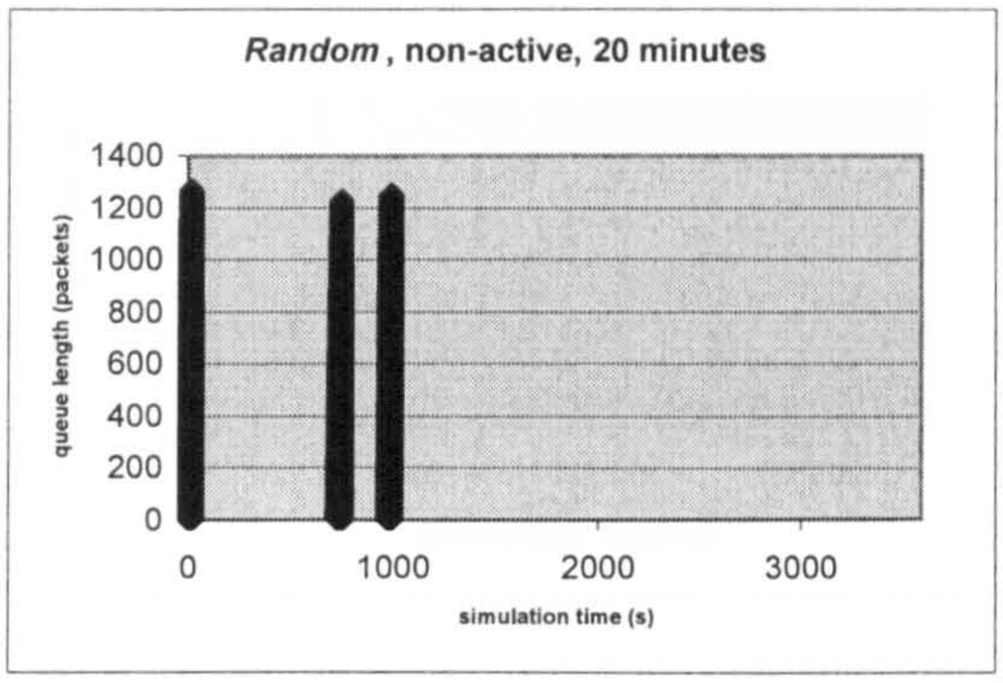
We start the examining of the results from the 20 minute experiments to find out the differences in performance that have been caused by using different movement patterns. After that we will observe the changes in performance caused by using the longer duration of the experiment.

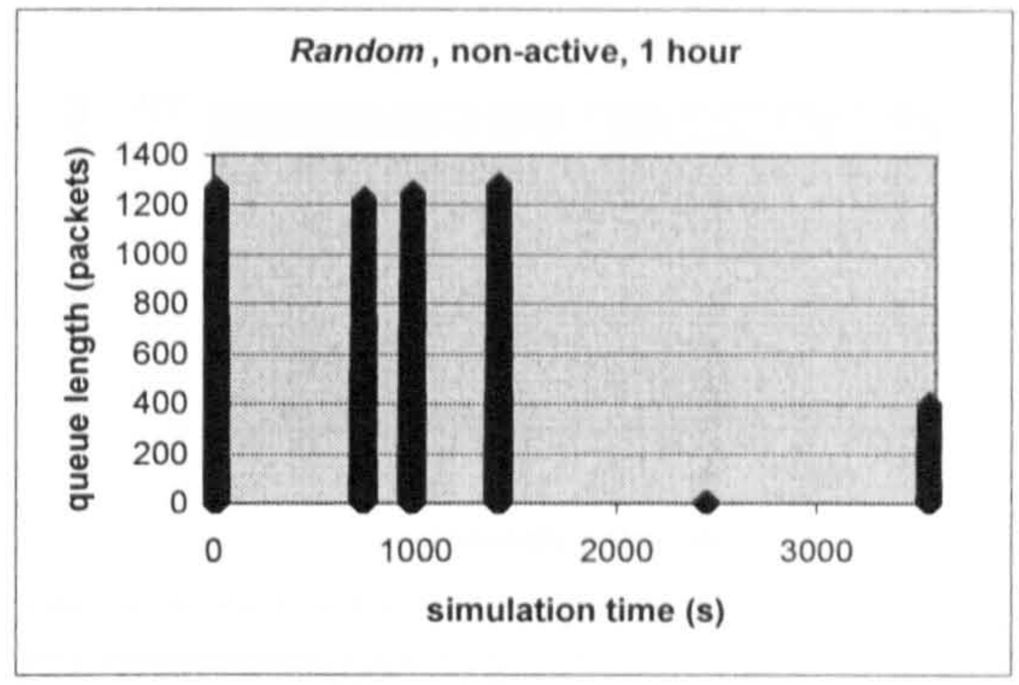
Figure 9-2-1 Movements of users for all application models (left - 20 minutes and right - 1 hour)

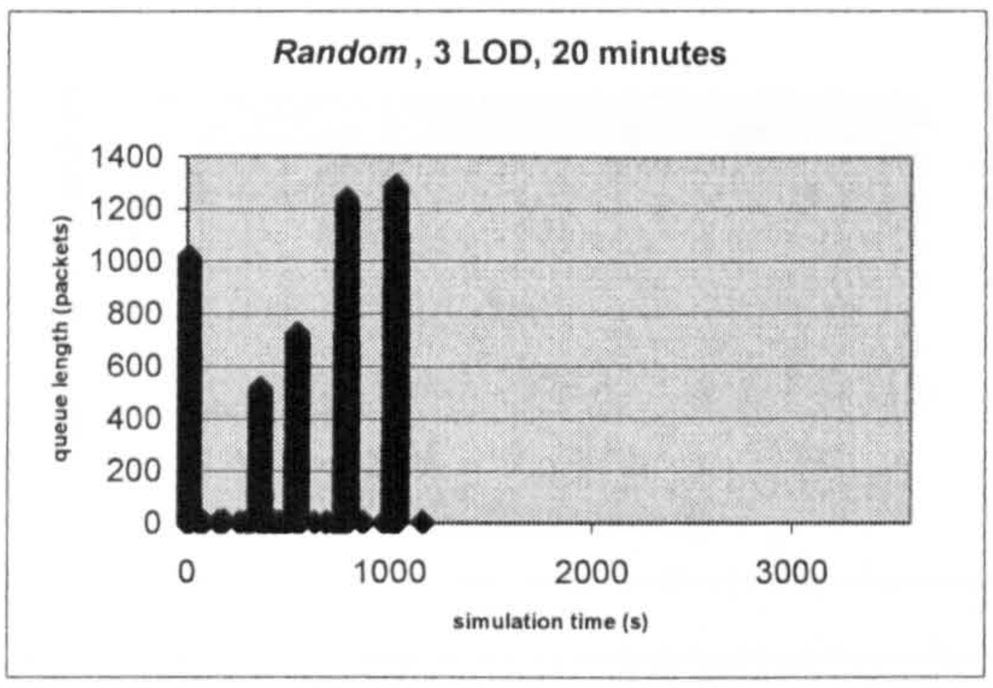
The results in Table 9-2-1 show that the maximum queue length differs depending on the movement scenario for all approaches. For example, the non-active approach shows the increase in queue length from 1266 packets for the *random* scenario to 2251 packets when users frequently leave and join the multicast group (*join* scenario). This significant increase of the queue length for the non-active *join* scenario can be explained by the fact that events of joining the multicast group by the users invoke retransmissions of 3D data close to each other in time scale and lead to higher congestion. The active approach excludes this situation by retransmission of 3D data only at certain time slots that are fixed by checking the LOD look up table at the host. For example, the 3 LOD active approach decreases the maximum length of the queue that has been obtained with exploiting the non-active approach for the *join* scenario more than twice. However, the 9 LOD active approach shows more than four times the decrease.

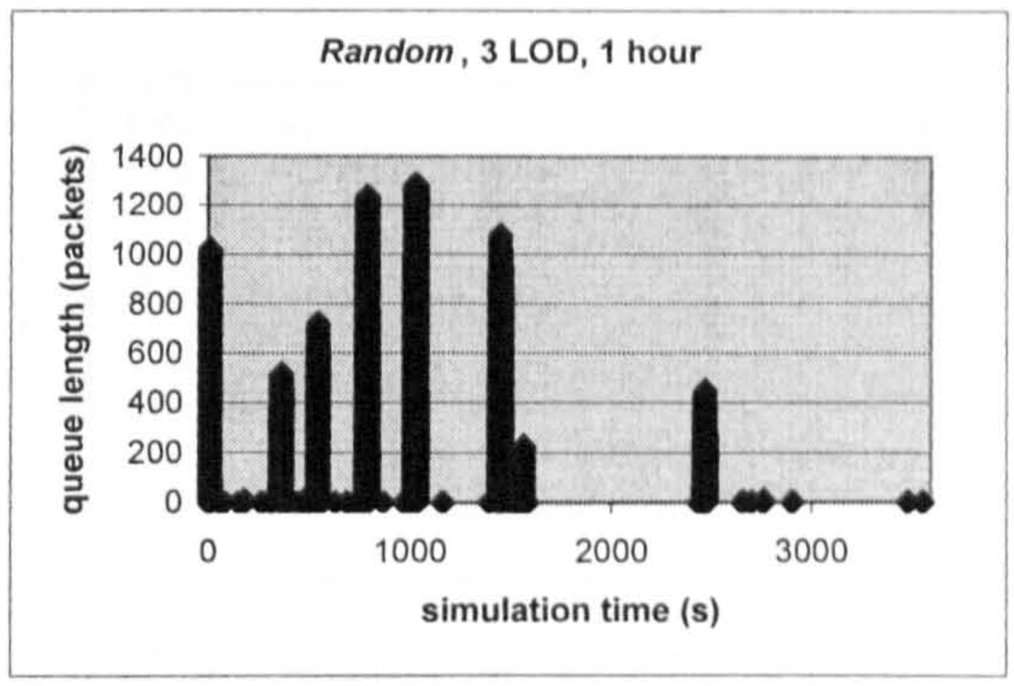
The *centre* scenario experiment gives the increase in the queue length for 9 LOD active approach from 548 packets for the *random* scenario to 864 packets. Within this scenario users retransmit highest batches of their progressive 3D representations that contain a lot of packets (e.g. LOD7=303 packets, LOD8=472 packets). Also, as the users are packed in the central area, there is a high possibility that not only one of the users would approach the sender but several. Therefore every user has to generate several different batches at one certain slot time. This will lead to invocation of several geometrical generators simultaneously on one host and considerably increase the aggregated multicast flow and the load to the network. The 3 LOD active *centre* scenario experiment does not show an increase of the queue length, as the users remain almost entirely within the central subspace of the 3x3 grid (See Section 7.2.2). For the *centre* scenario test the 3 LOD active approach decreases the maximum length of the queue that has been obtained with exploiting the non-active approach by 17% and the 9 LOD active approach gives a decrease by 29%.

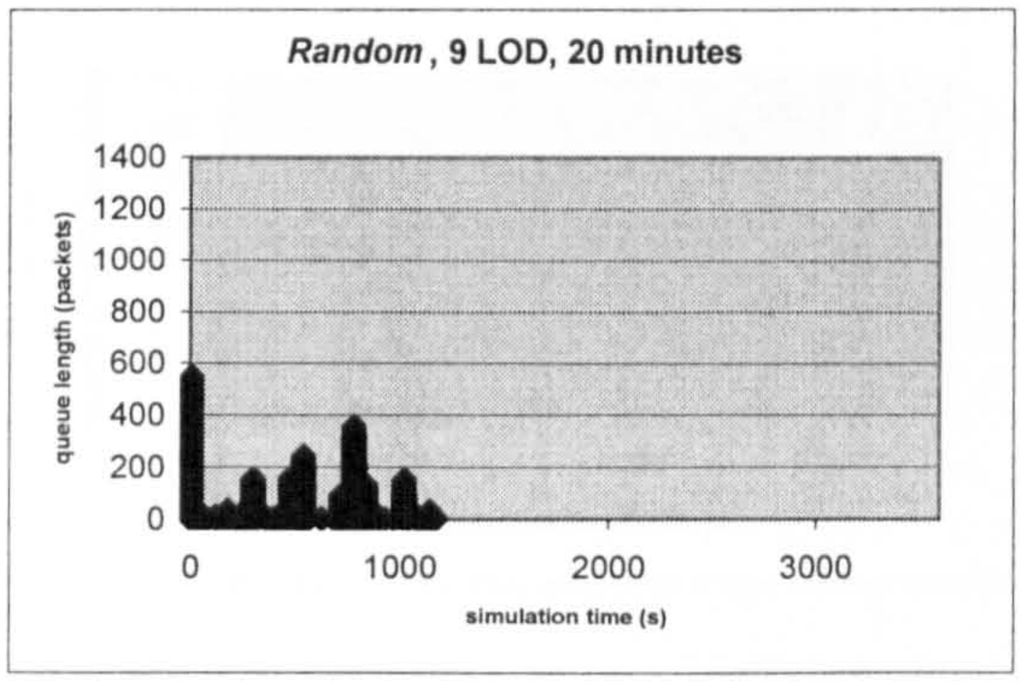
The one-hour scenario results show that there are no changes in this performance metric for the increased duration of the experiments for *far* and *join* scenarios. However, the *centre* scenario demonstrates a slight increase in the queue length. The *random* scenario shows a considerable increase in the queue length (from

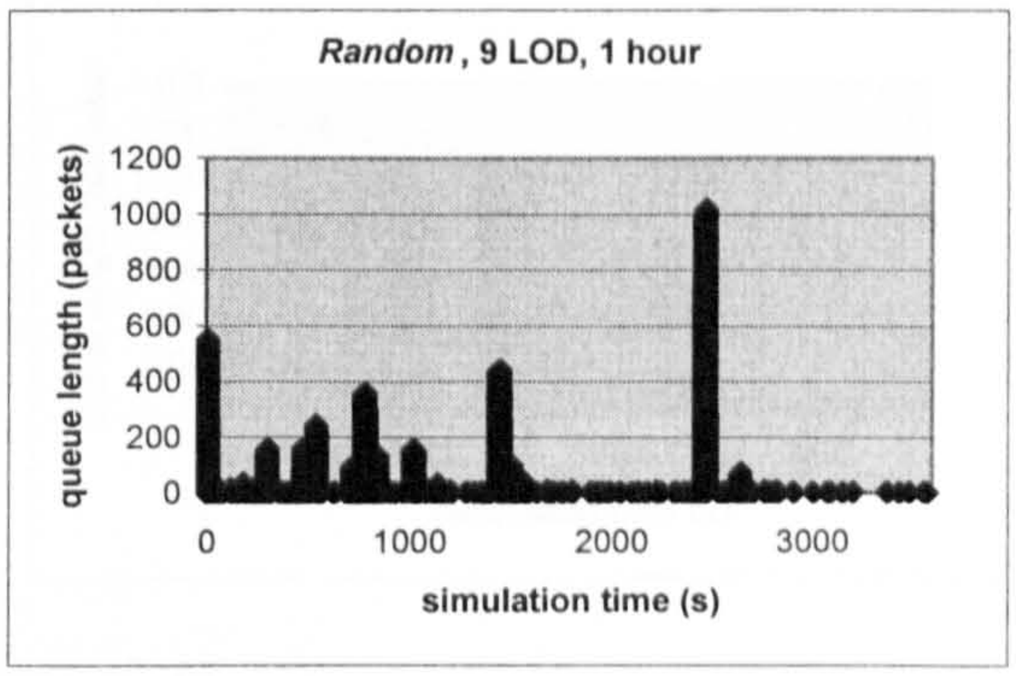






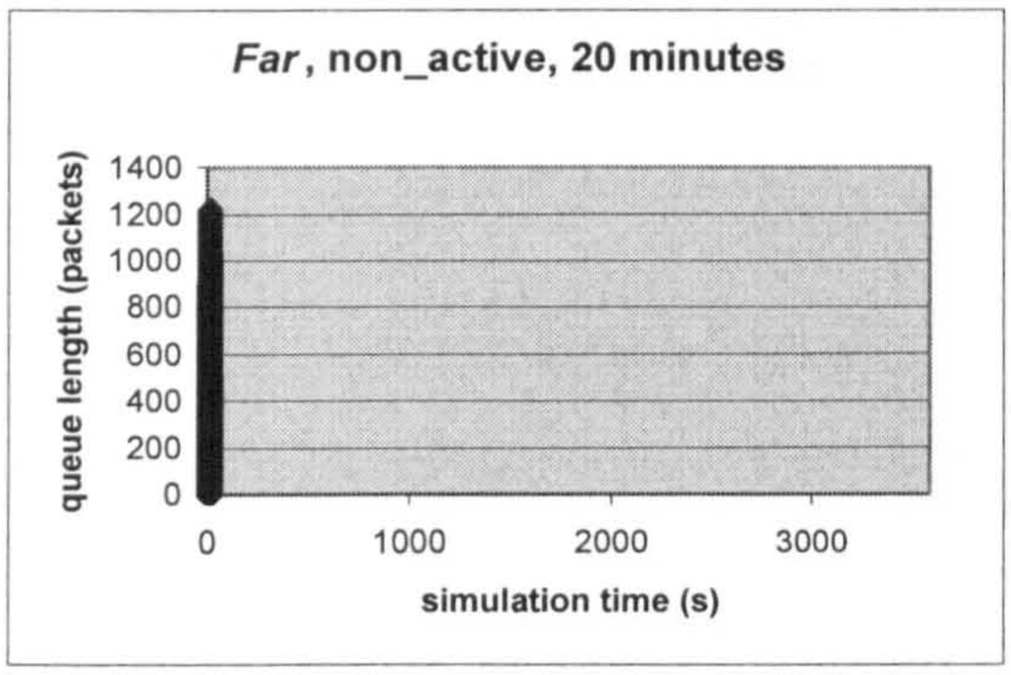


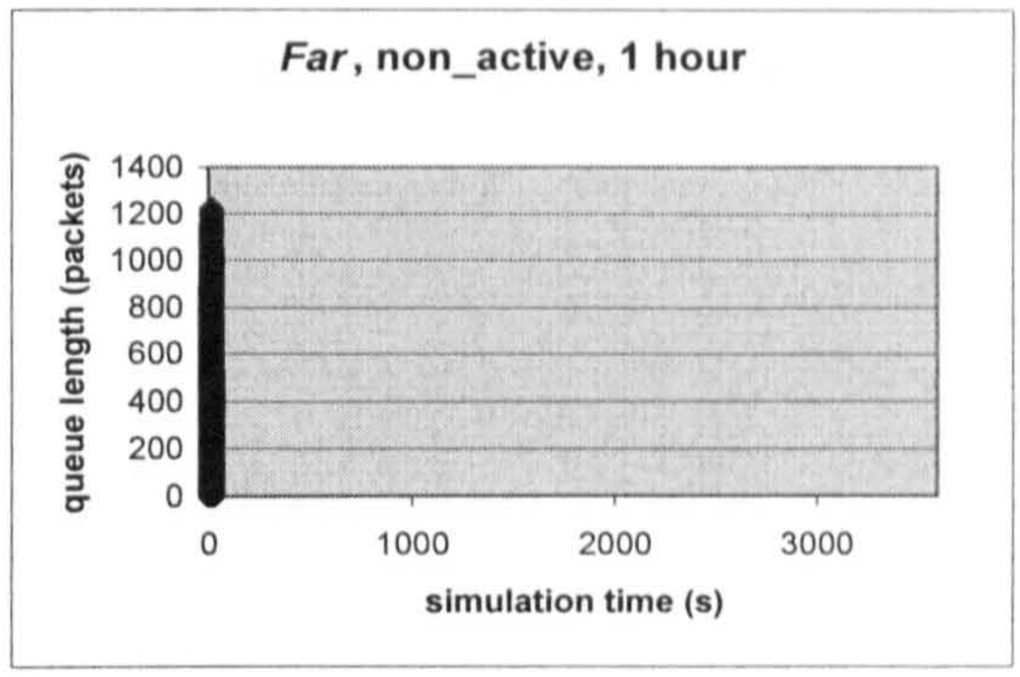


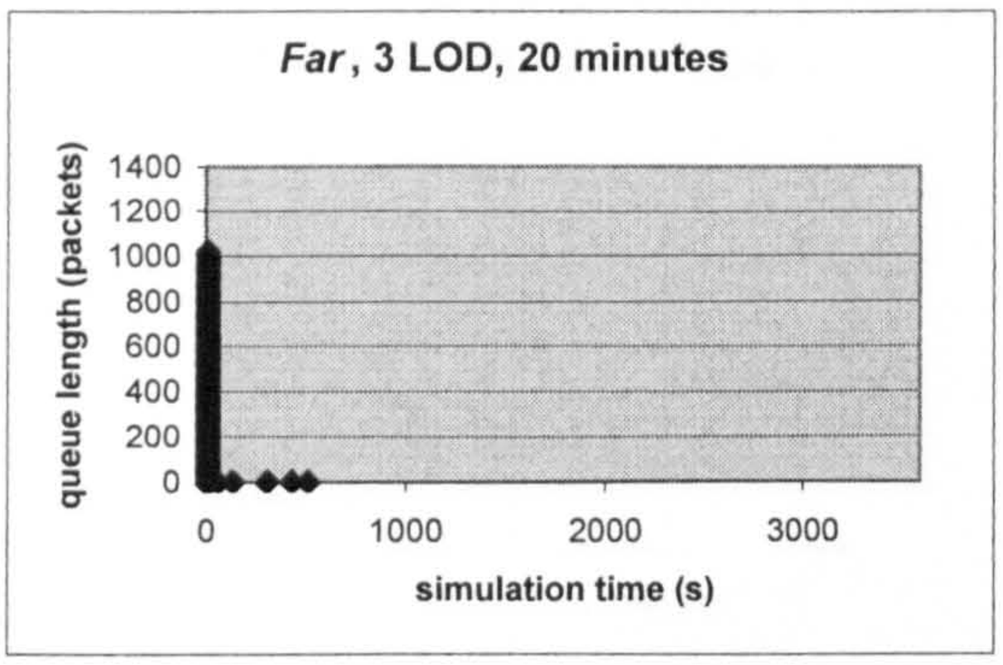


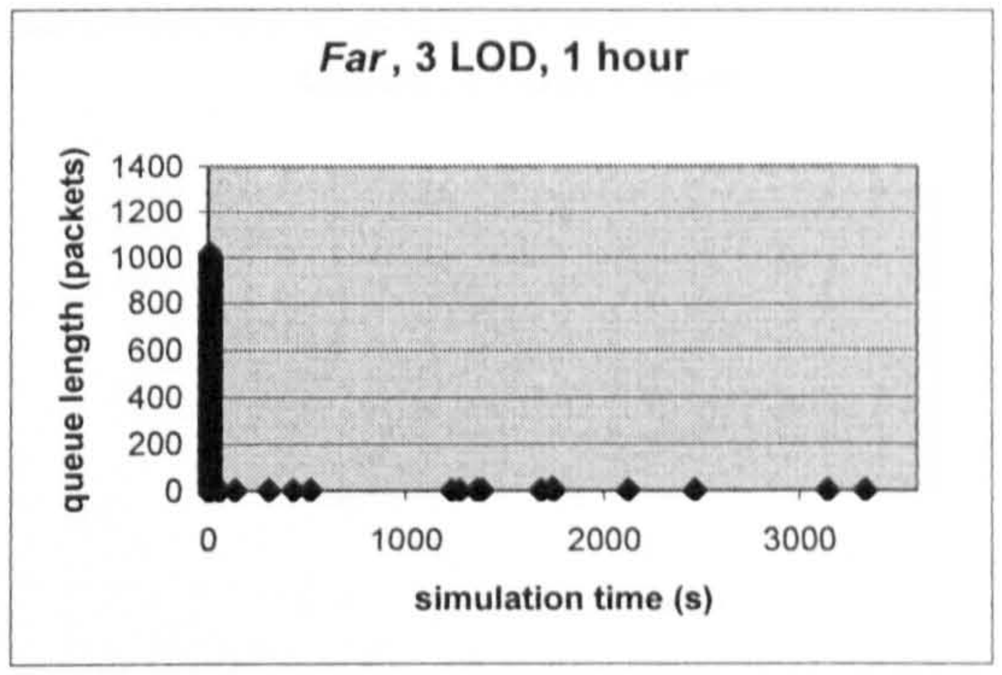
a) **b)**

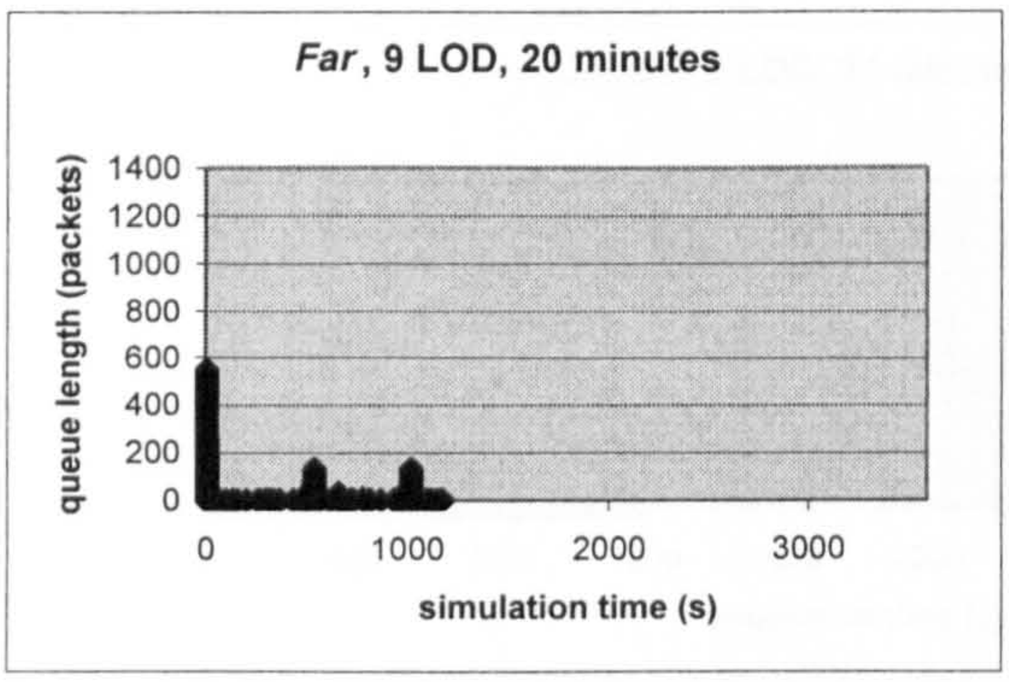
Figure 9-2-2 Queue occupancy results for 20 minutes and one hour (*random* scenario)

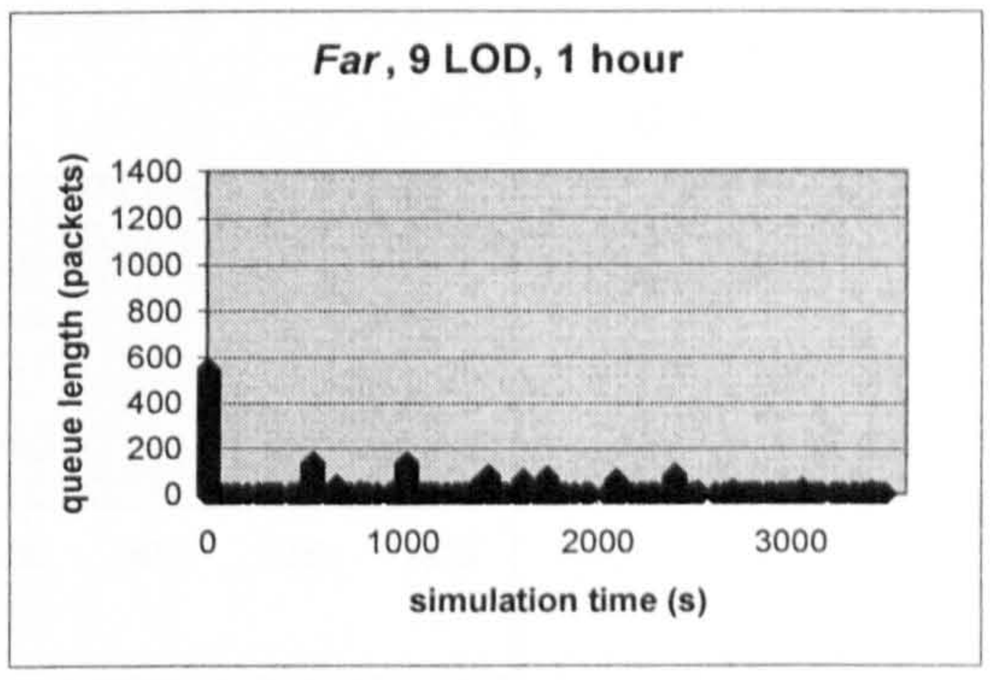






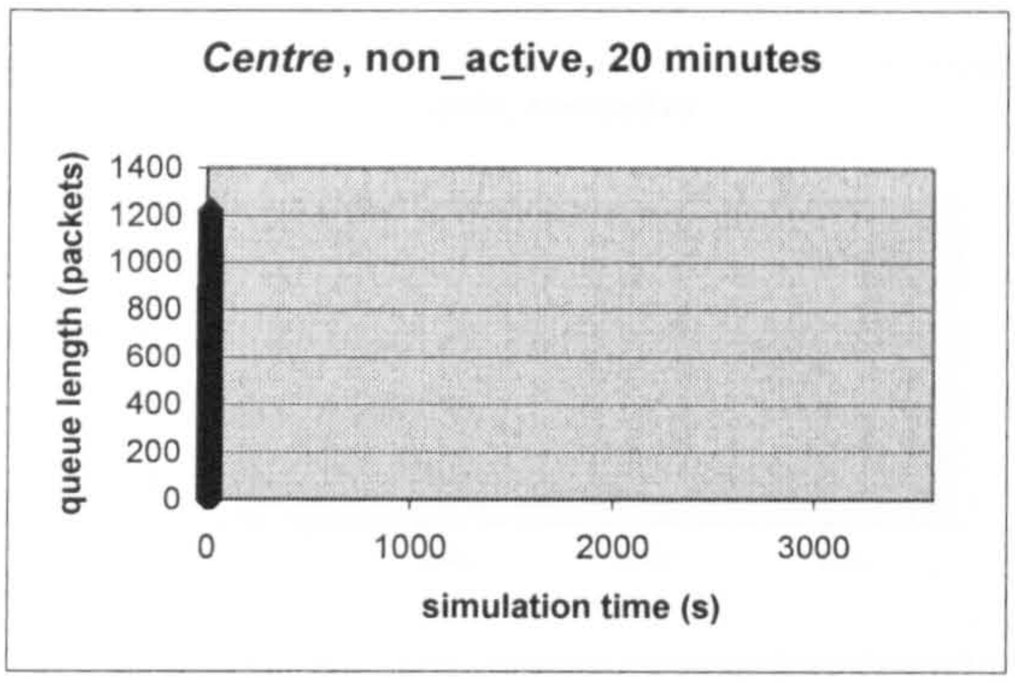


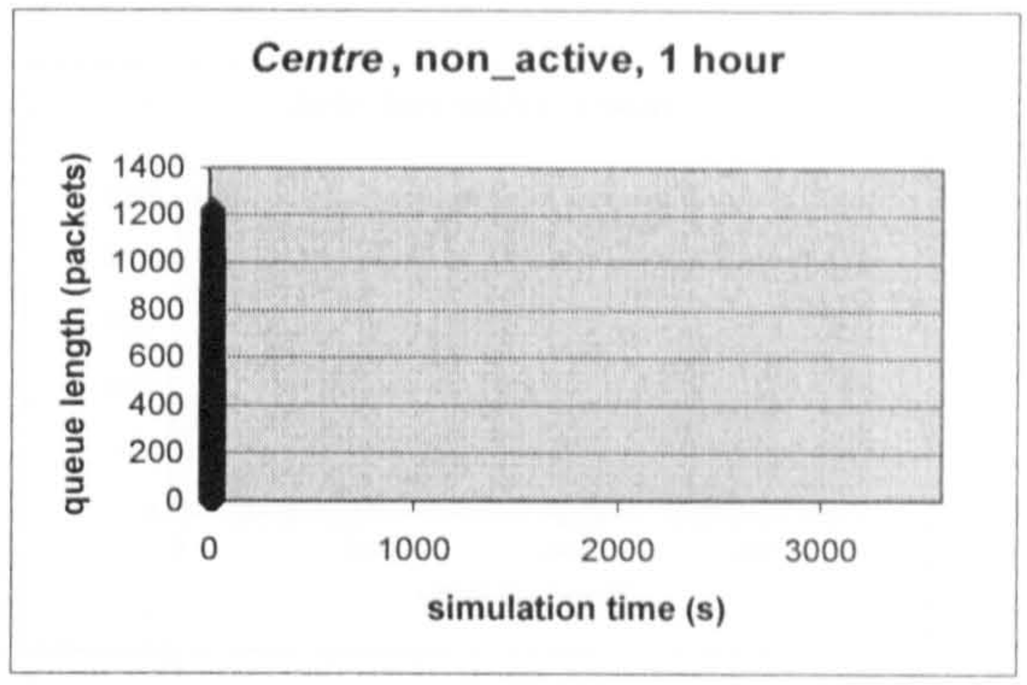


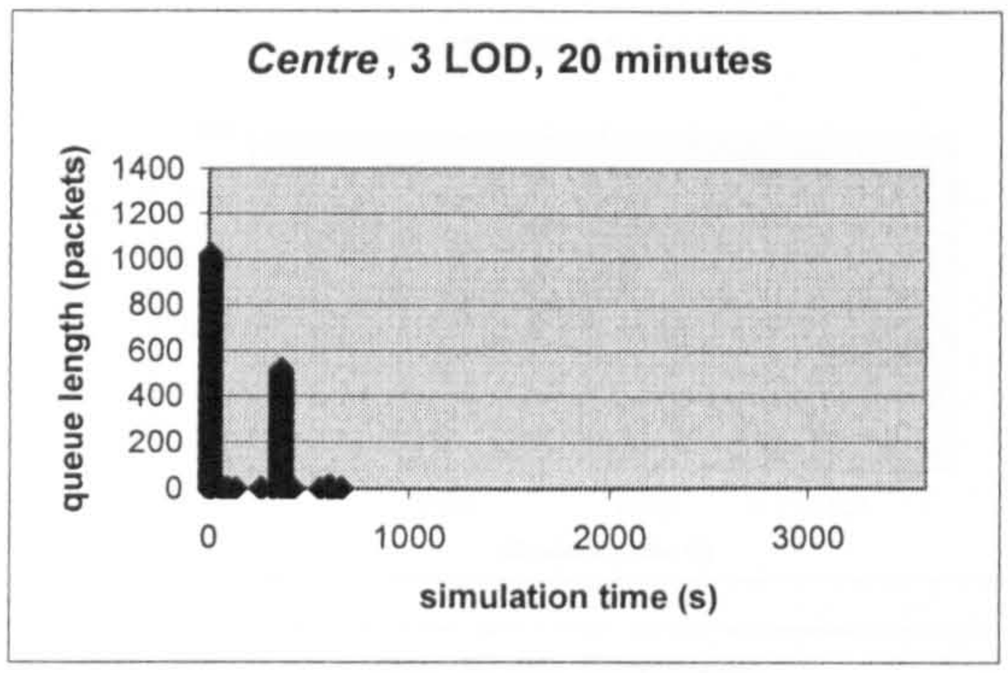


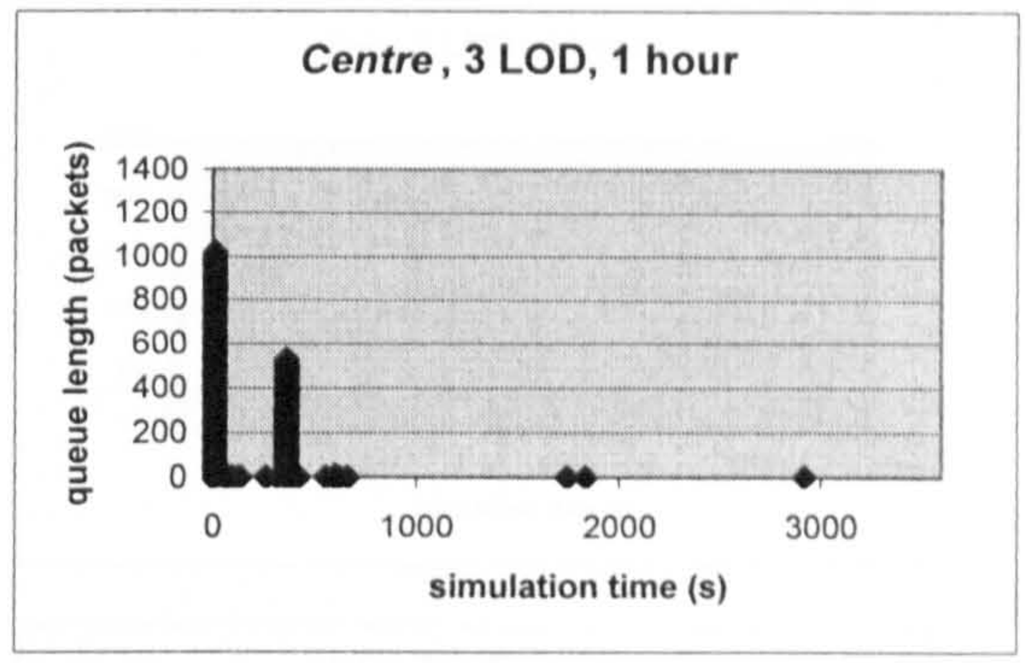
a) **b)**

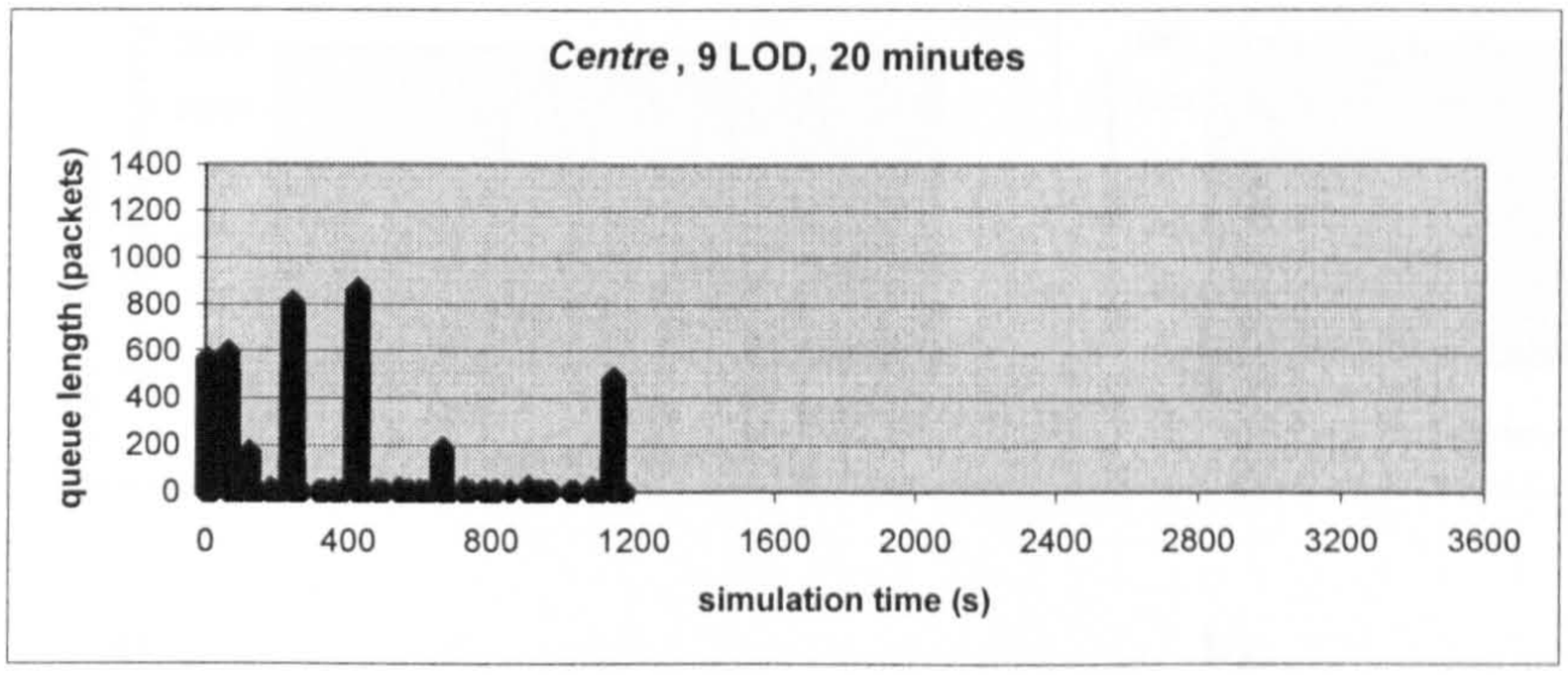
Figure 9-2-3 Queue occupancy results for 20 minutes and one hour (*far* scenario)

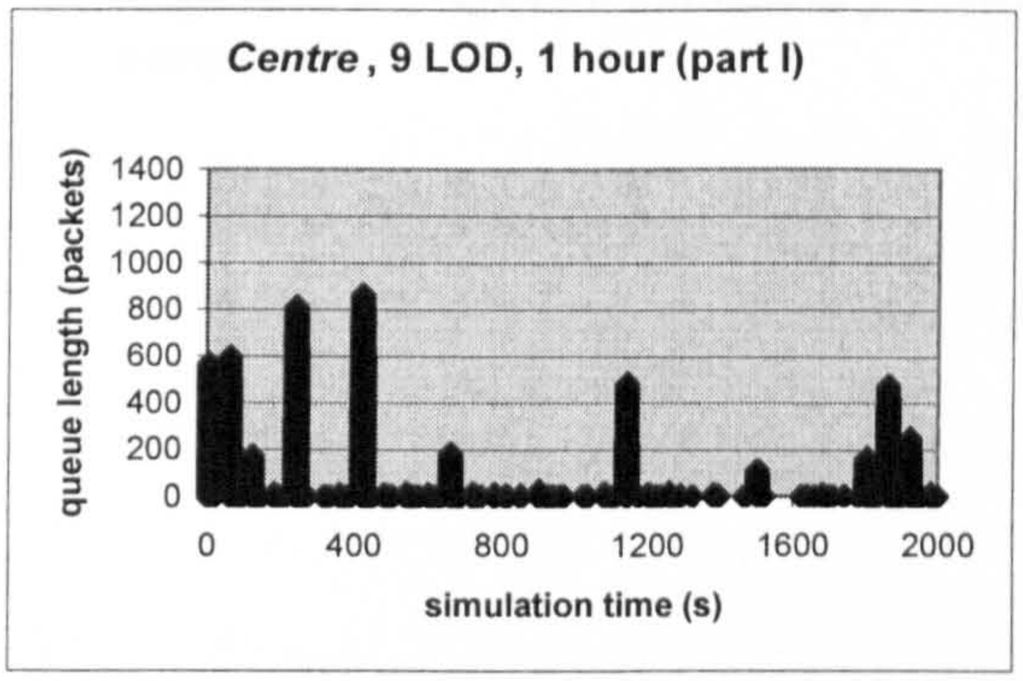












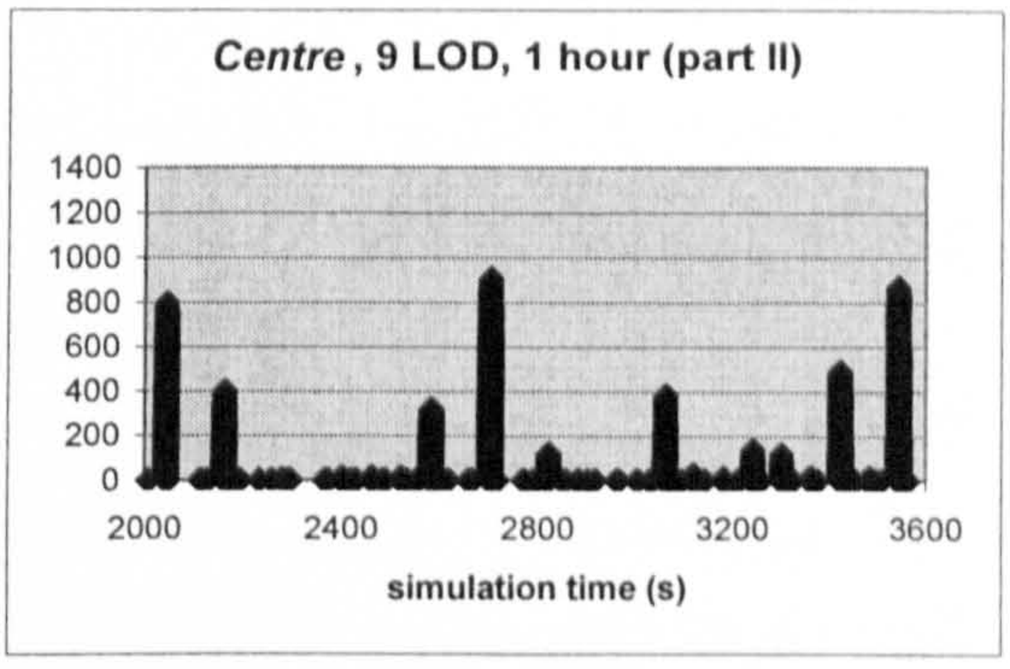
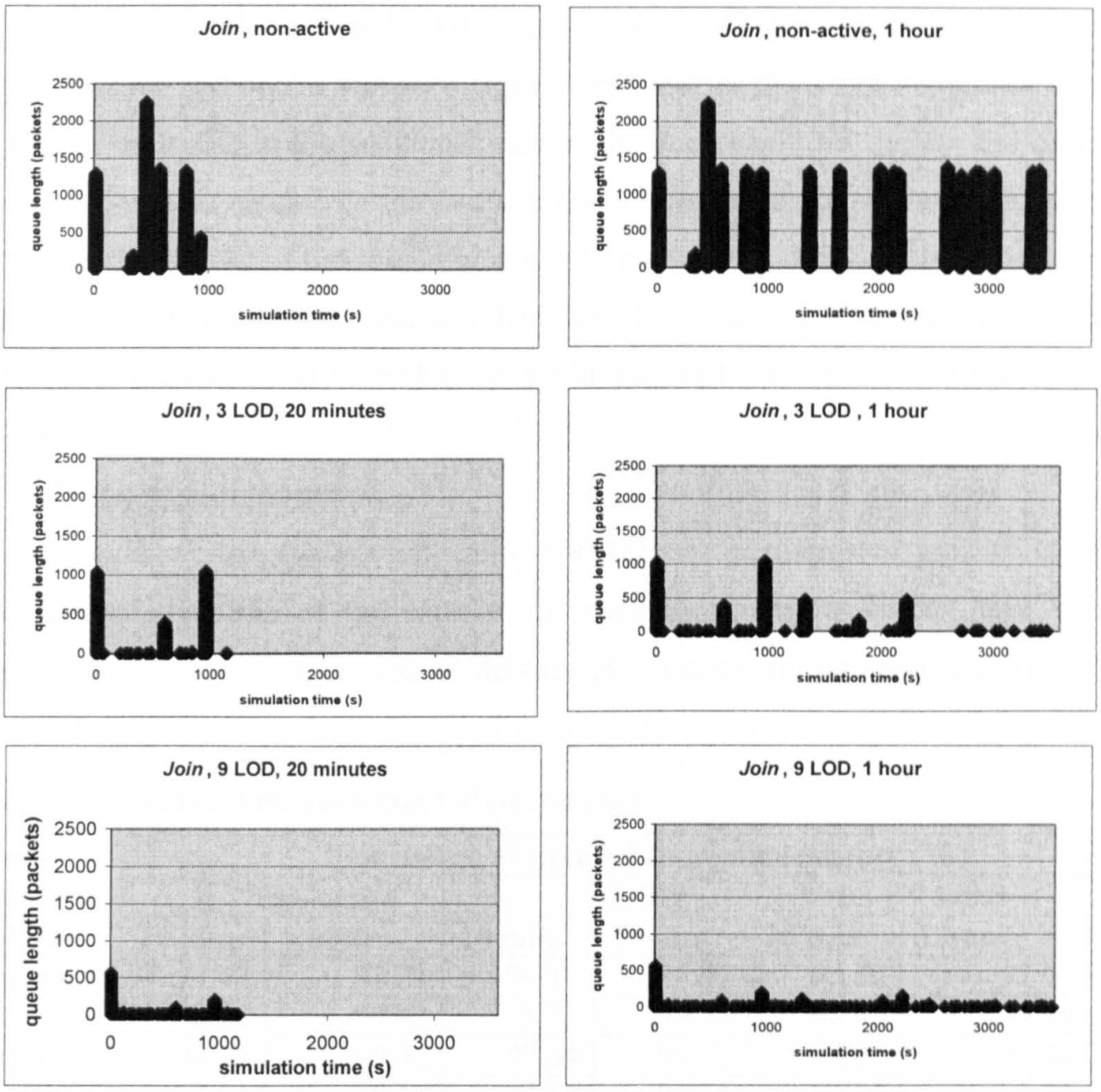


Figure 9-2-4 Queue occupancy results for 20 minutes and one hour (*centre* scenario)



a) **b)**

Figure 9-2-5 Queue occupancy results for 20 minutes and one hour (*join* scenario)

548 for 20 minutes to 1015 for one hour experiment). The graphs presented in Figures 9-2-2 – 9-2-5 show that for the 9 LOD approach the traffic patterns remain the same for one-hour *far* and *centre* scenarios experiments and for the 3 LOD approach there was almost no traffic in the additional time of the experiment for the *far* and *centre* scenarios. This was caused by the fact that users have been almost static within the 3x3 grid in this period of time (see Figures 9-2-1b - 9-2-1c). The *random* case shows high peaks within a one-hour scenario that could be caused by invocation of several single batch generators as several users could approach the sender simultaneously (see Section 9.2.1).

2) Utilization of the queue.

To evaluate the queue's utilization for different scenarios of movement the percentage of the time of the experiment when the queue is empty have been compared for different application models for 20 minutes and one-hour experiments (Table 9-2-2).

Table 9-2-2 Utilization of the queue (Set 1 of experiments)

Move-ments' pattern	Percentage of time when queue is empty (%)					
	Non-active		3 LOD		9 LOD	
	20 min	1 hour	20 min	1 hour	20 min	1 hour
random	94.47	95.21	94.17	96.92	95.52	97.07
far	98.26	99.42	98.82	99.53	98.99	99.39
centre	98.26	99.42	97.64	99.22	94.06	94.74
join	89.54	88.11	94.64	97.51	97.93	98.81

The 20 minute scenario results in Table 9-2-2 show that despite the generally very low utilization of the queue within 3D data retransmissions by the DVE application for all scenarios (the queue is empty about 89% - 99% of experiment's time), the *join* scenario demonstrates a better decrease in queue utilization by raising the empty time percentage from 89.53% for the non-active approach to 97.93% for the 9 LOD active approach. *Far* scenario experiments, in contrast, show a very insignificant decrease in the queue utilization (raising the empty time percentage from 98.26% for the non-active approach to 98.99% for the 9 LOD active approach). However, it indicates that the amount of 3D traffic (batches of 9 LOD progressive representation) caused by user's movements within the 9x9 grid within the *far*

scenario creates less load on the network than only initial transmission of the full original representations of users within the non-active *far* scenario. The *centre* scenario experiments show increasing the load on the network within the 9 LOD approach experiment and it is reflected in a decrease of the empty time percentage from 98.26% (that corresponds only to initial transmission) for the non-active centre test to 94.06% for the 9 LOD active test. The 3 LOD scenario does not give significant changes in queue utilization for *random*, *far*, and *centre* scenarios in comparison with the non-active approach, but the *join* scenario shows the decrease in the queue utilization by raising the empty time percentage from 89.53% for the non-active approach to 94.64% for the 3 LOD active approach.

The one-hour scenario results in Table 9-2-2 demonstrate that the 9 LOD experiments show almost the same percentage of empty time for 20 minutes and one-hour tests (98.99% and 99.39% for *far* scenario, and 94.06% and 94.74% for *centre* scenario). The 3 LOD tests show a slight increase of percentage of empty time for 20 minutes and one- hour tests (98.82% and 99.53% for *far* scenario, and 97.64% and 99.22% for *centre* scenario). There is no significant difference in the utilization of allocated bandwidth between the active and non-active cases except for the *join* case. However, even in this worse case the utilization is still very low for both 20 minute and one-hour scenarios.

The node performance results show that the traffic generated by transfer of geometric information is very bursty for all application models and results in the need for large queue sizes, but a low overall utilization of the allocated bandwidth. The results indicate that the active approach, particularly when using nine levels of detail, can generally offer a significant improvement over the non-active approach in respect to queue size requirements, however, the degree of improvement is dependent on the movement pattern of users. Even with the improvements offered by the active approach this type of traffic still required quite large queue sizes.

II. Testing of host's QoS parameters.

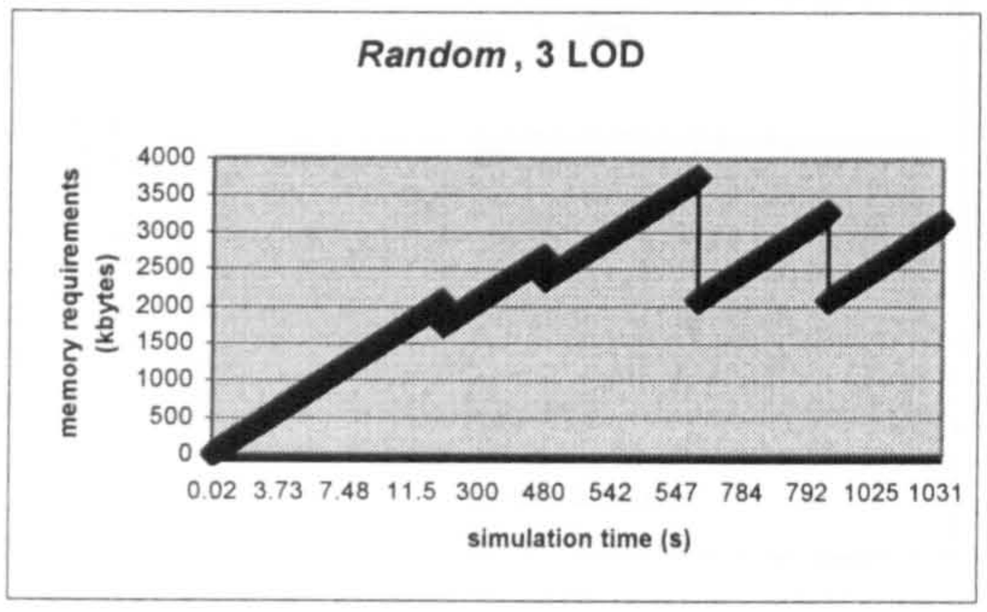
Not only the network QoS parameters that reflect congestion and delays but also the QoS parameters on a host would be improved by applying the active

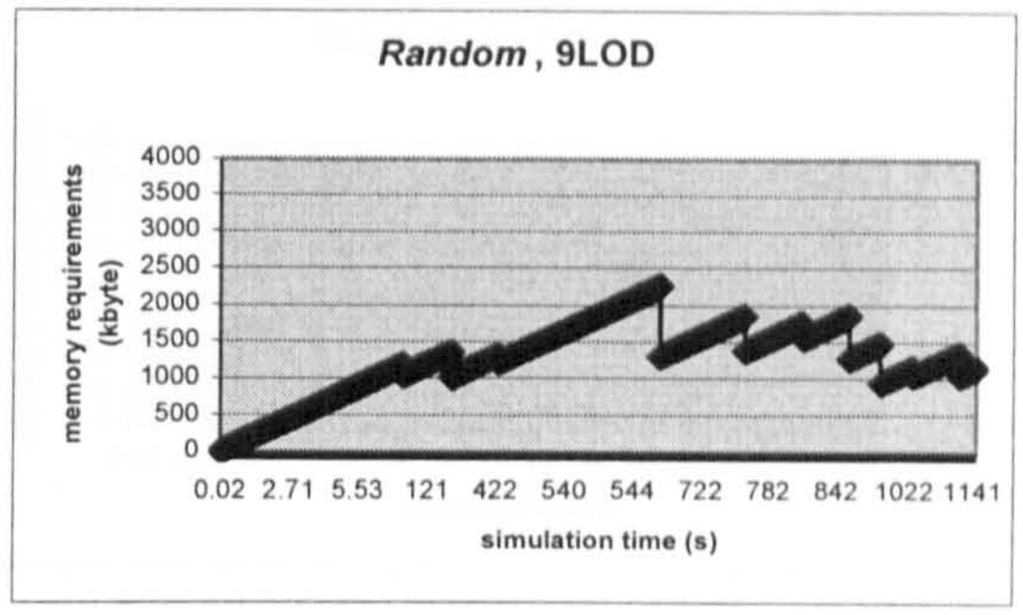
approaches of 3D progressive data transmission. As both the network node's and host's functionalities have an effect on performance of DVE applications it was important in the testing experiments to monitor changes of host QoS parameters within different movement scenarios. The change in memory requirements on hosts for storing 3D representations of other members of the multicast group has been monitored. A decrease in the host memory requirements to store 3D representations for the DVE application improves the scalability of the application as it allows the user to store simultaneously the representations of larger numbers of users.

The memory requirements results are presented for one of the hosts – user B. Figure 9-2-6a – 9-2-6d shows the plotted memory requirements results of four different movement scenarios and presents 3LOD and 9LOD active approaches for every scenario. The required memory has been measured in the times when memory requirements have been changing (e.g. reception of the new more detailed batch or relinquishing the unnecessary batch). For example, within the *far* 3 LOD scenario (see Figure 9-2-6b) user B receives the representations of the others only at the initialization stage of the application. Since the distance from user B to others remains unchanged in the grid 3x3 until the end of the experiment no additional 3D data will be received and no further changes in memory requirements will occur (see Section 7.2.2). Therefore user B in this experiment needs a constant amount of memory after the completion of the initial reception of 3D data to the end of the experiment.

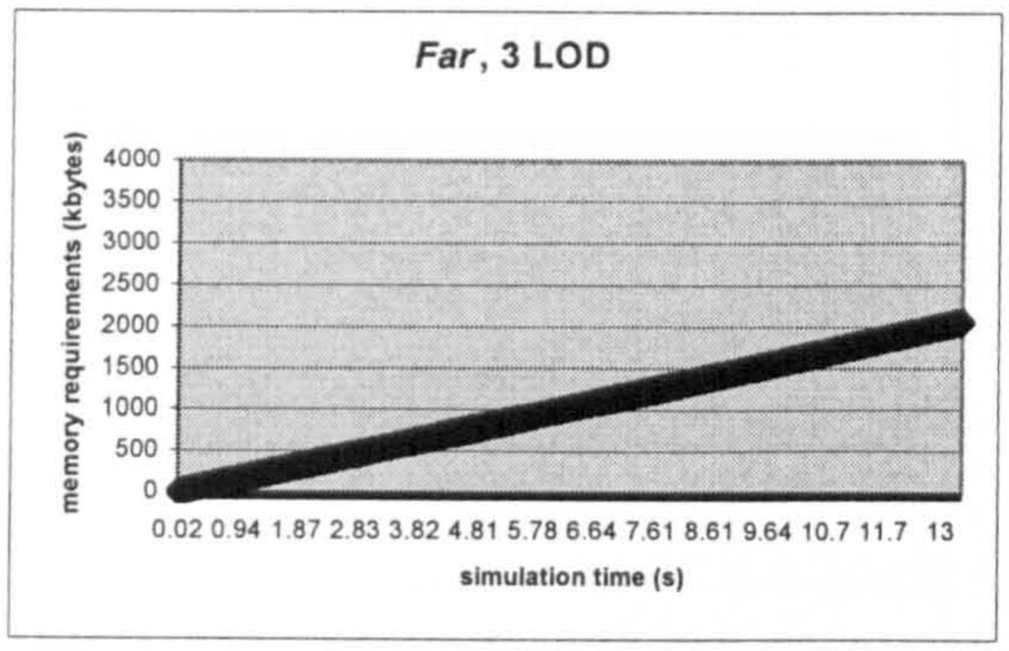
Comparison between non-active and active approaches has been based on monitoring the maximum value of the memory required to store representations of the other users. As there are four users in the group, every host needs to store representations of three other viewed participants. The full detailed representations differ in size for non-active, 3 LOD and 9 LOD approaches (non-active – 1265 kbytes, 3 LOD – 1360 kbytes, and 9 LOD – 1495 kbytes). Therefore the maximum possible memory requirements for the user are calculated as follows:

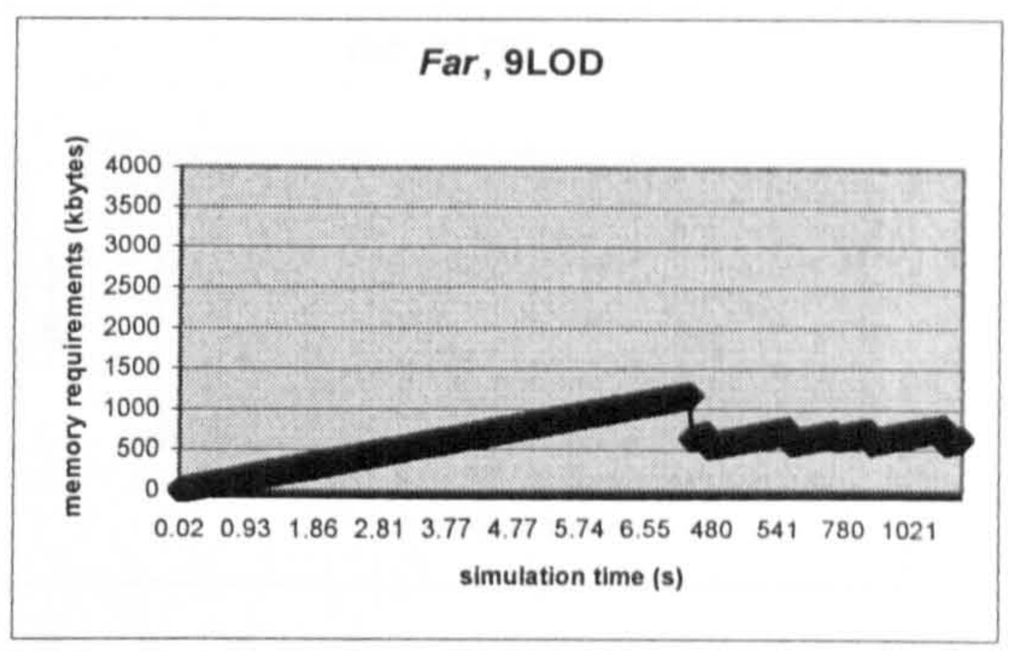
- Non-active approach – 1265*3=3795 kbytes;
- 3 LOD active approach – 1360*3=4080 kbytes;
- 9 LOD active approach – 1495*3=4485 kbytes;



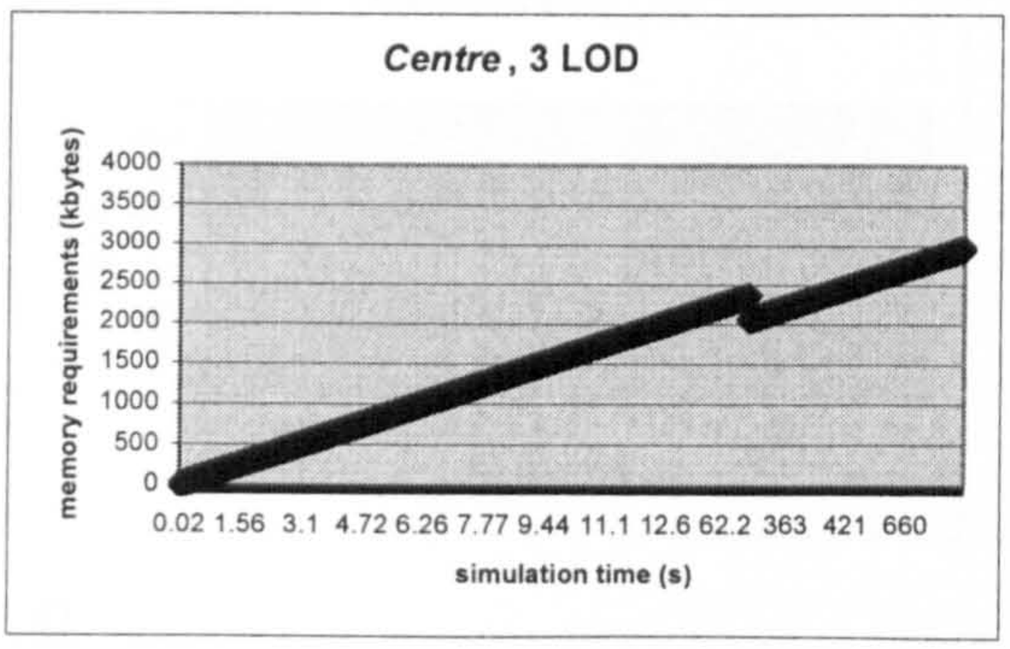


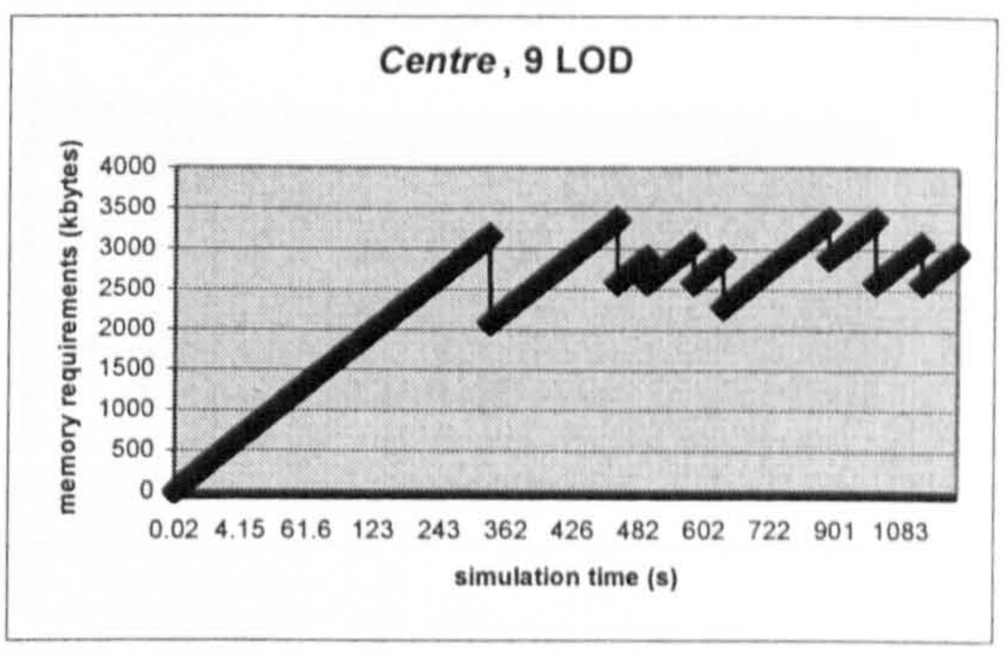
a)



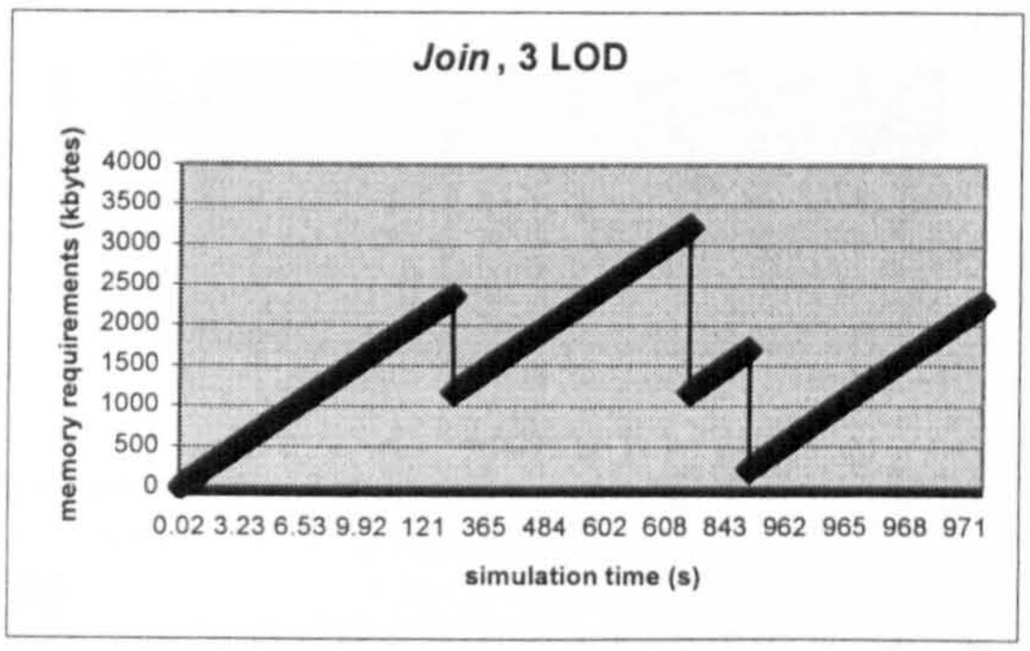


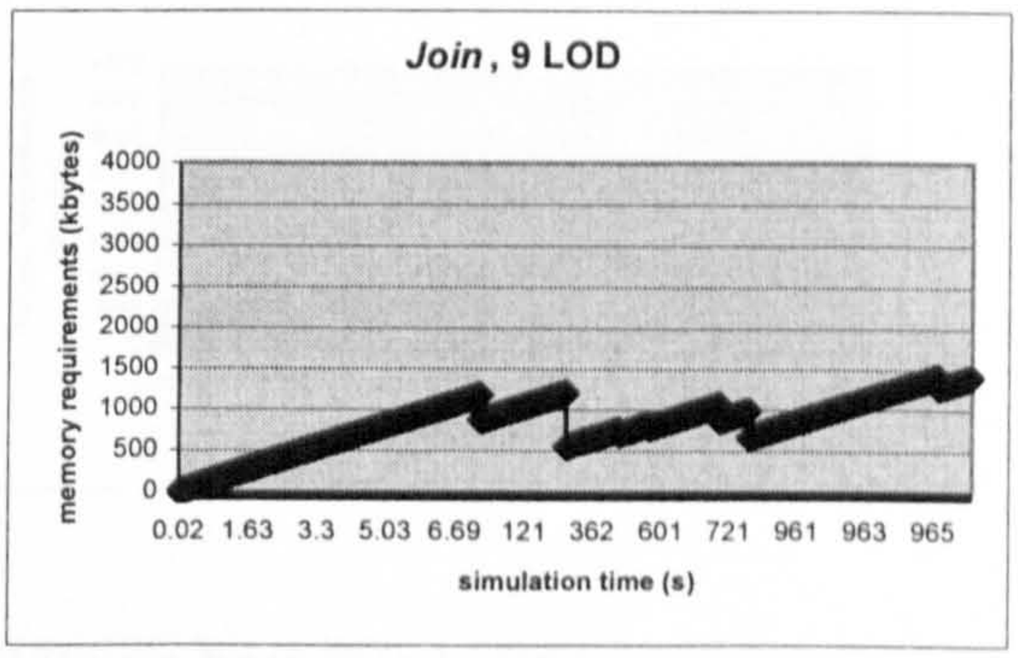
b)





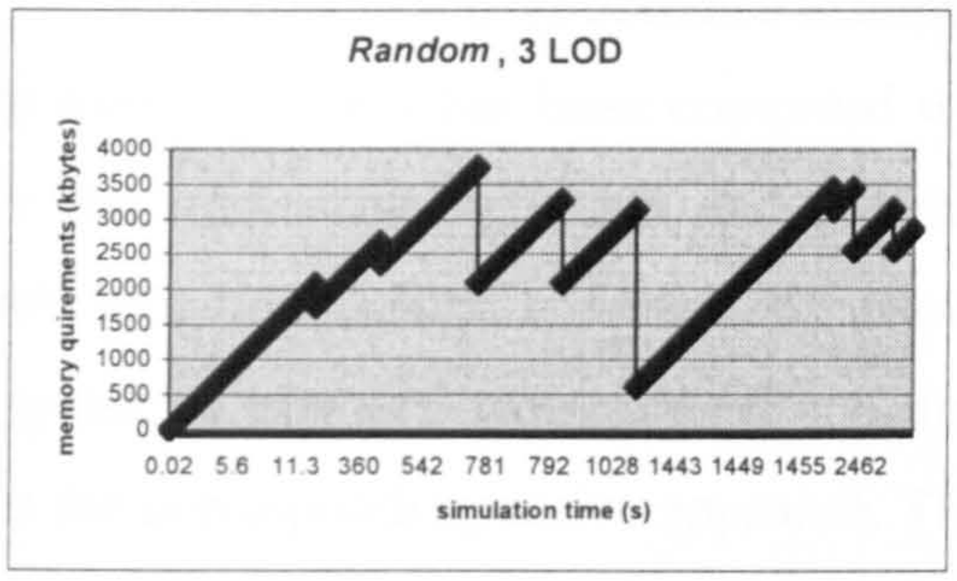
c)

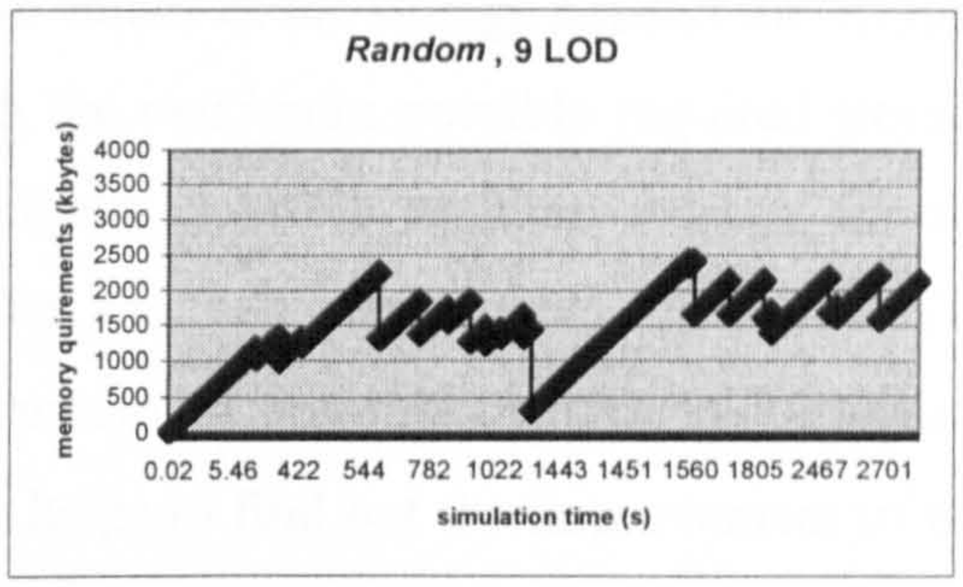




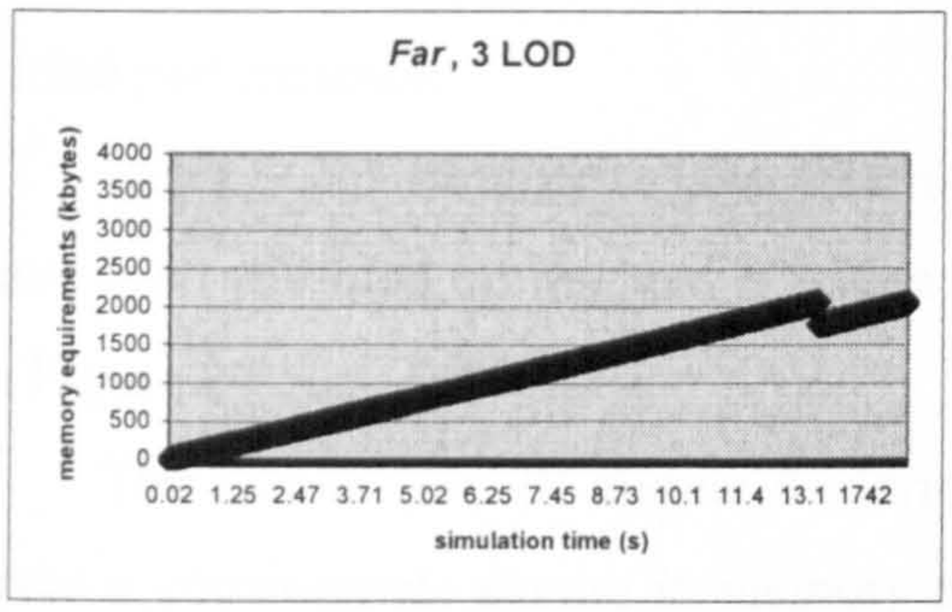
d)

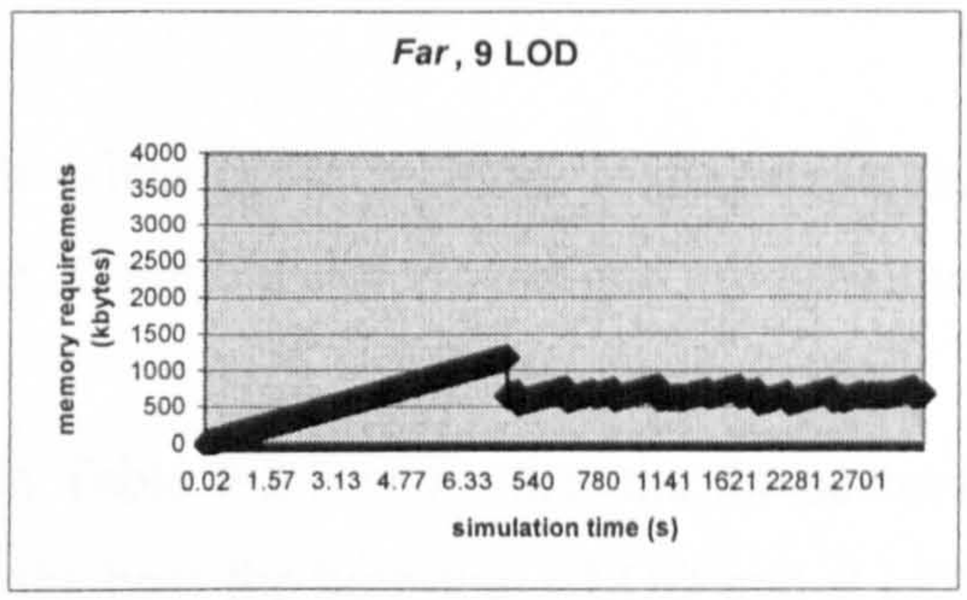
Figure 9-2-6 Memory requirements for 3D data on host B (20 minutes)



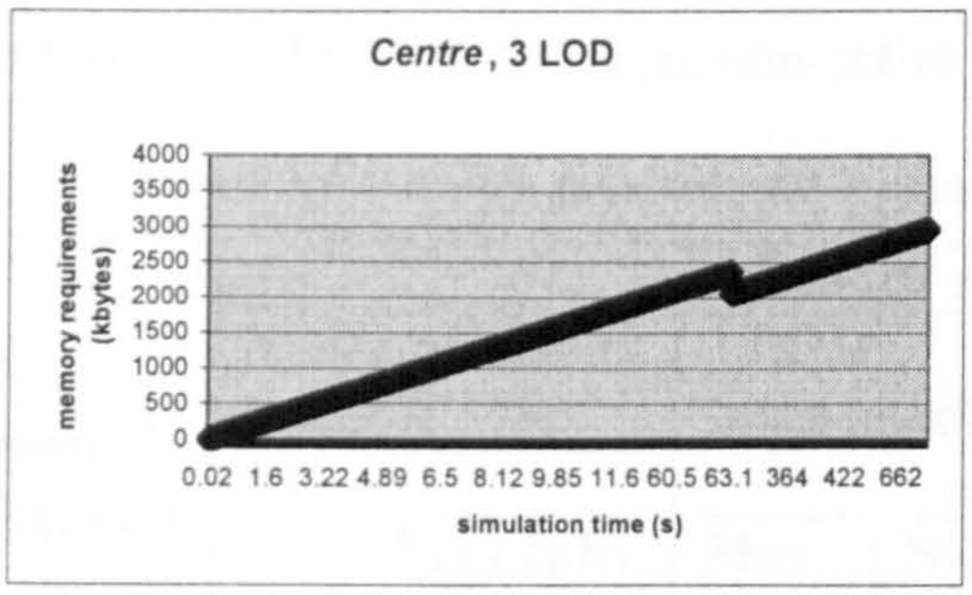


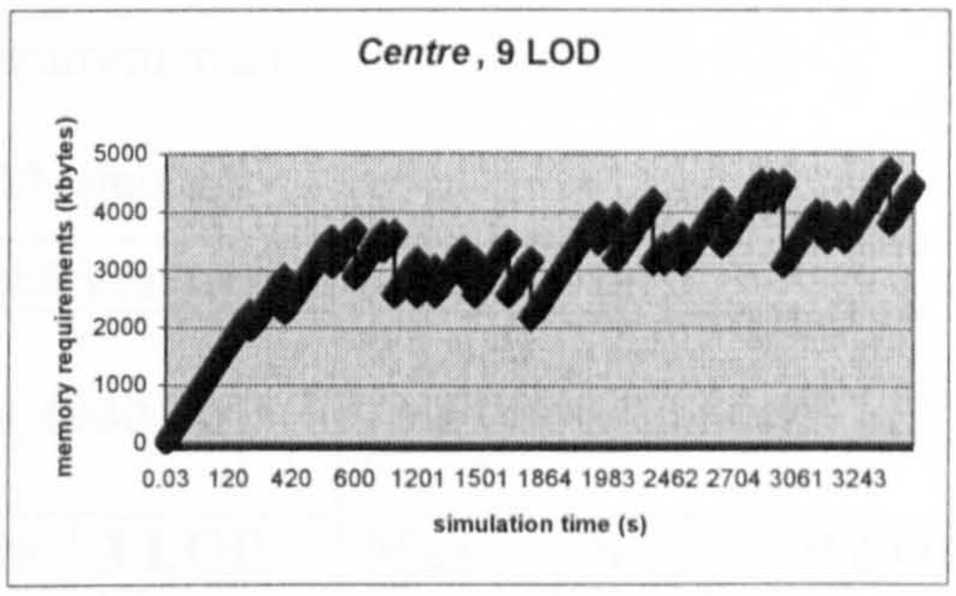
a)



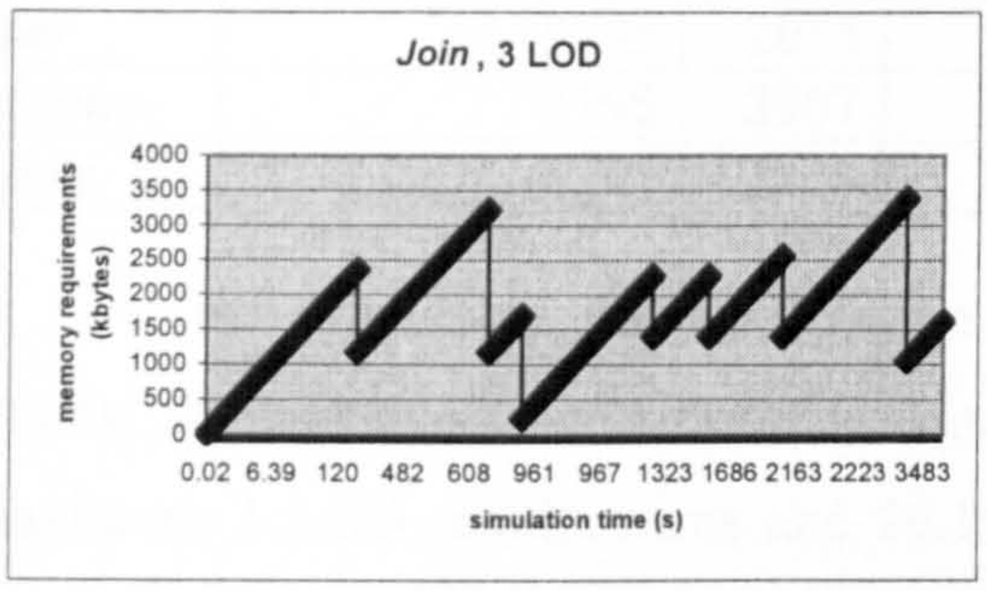


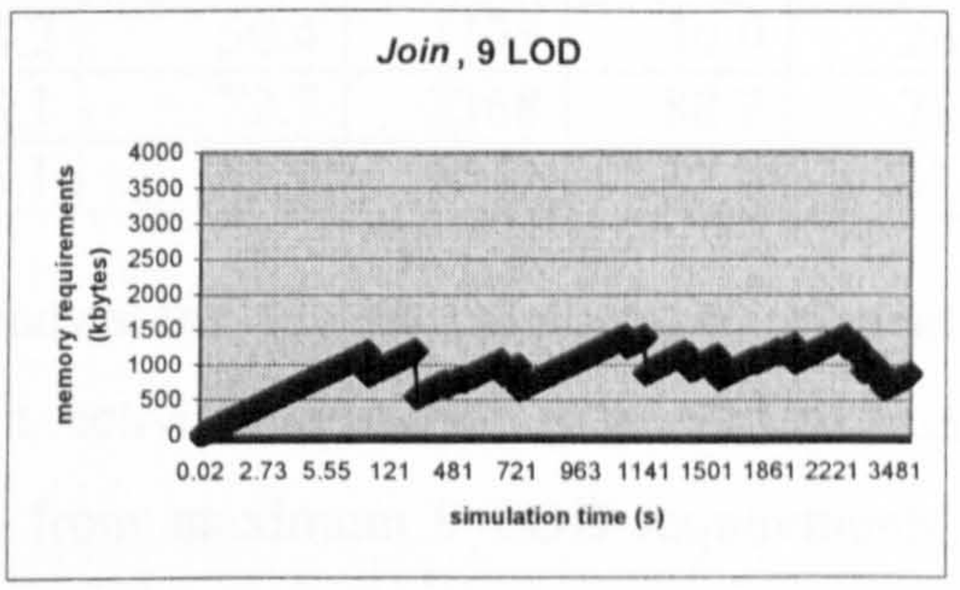
b)





c)





d)

Figure 9-2-7 Memory requirements for 3D data on host B (one hour)

The experimentally obtained maximum values of the storage needed for 3D data for every scenario has been compared with the maximum possible required storage for the non-active approach and the maximum possible required storage for the corresponding active approach by showing the percentage from the non-active approach's memory requirements and the maximum possible memory requirements for the corresponding active approach. This helps to find out the improvement of the storage management that the active approaches gain in comparison with the non-active and detect the movement scenarios within certain active approaches that give varied performance.

Table 9-2-3 summarises the maximum values of the memory requirements that have been admitted on the host B within the 20 minute experiment and are presented in Figure 9-2-6a – 9-2-6d.

The results for 20 minute scenario in Table 9-2-3 show that the *far* scenario gives a considerable saving in memory on the host for both the 3 LOD and 9 LOD approaches in comparison with the non-active approach (e.g. for the 3 LOD 54.2% and for the 9 LOD 30.9% from non-active requirements).

Table 9-2-3 Memory requirements on host (20 minutes)

Move-ments' pattern	Memory requirements on host (kbyte)						
	Non-active(max possible 3795 kb)	3 LOD (max possible 4080 kb)			9 LOD (max possible 4485 kb)		
	Max (kb)	Max (kb)	N-A%	3 LOD %	Max (kb)	N-A %	9 LOD %
random	3795	3742	98.6	91.7	2254	59.4	50.2
far	3795	2058	54.2	50.4	1175	30.9	26.1
centre	3795	2967	78.1	72.7	3368	88.7	75.1
join	3795	3228	85.1	79.1	1412	37.2	31.4

At the same time the *far* scenario shows the lowest memory requirements among all movement scenarios for certain active approaches (e.g. 50.4% from maximum 3 LOD requirements and 26.1% from maximum 9 LOD requirements). The highest memory requirements for the active approach have been recorded for the *random* 3 LOD experiment – 3742 kbytes, that is 98.6% from maximum non-active

requirements, and for the *centre* 9 LOD experiment – 3368 kbytes, that is 88.7% from the maximum non-active requirements.

Table 9-2-4 Memory requirements on host (one hour)

Move-ments' pattern	Memory requirements on host (kbyte)						
	Non-active(max possible 3795 kb)	3 LOD (max possible 4080 kb)			9 LOD (max possible 4485 kb)		
	Max (kb)	Max (kb)	N-A%	3 LOD %	Max (kb)	N-A %	9 LOD %
random	3795	3742	98.6	91.7	2417	63.68	53.89
far	3795	2058	54.2	50.4	1175	30.9	26.1
centre	3795	2967	78.1	72.7	4485	118	100
join	3795	3384	89.17	82.94	1412	37.2	31.48

Table 9-2-4 summarises the maximum values of the memory requirements that have been recorded on the host B within the one-hour experiment and are presented in Figure 9-2-7a – 9-2-7d. The *centre* scenario shows the highest memory requirements (100% from maximum 9 LOD requirements). The one-hour experiment's graphs show that memory requirements for all scenarios have the similar patterns for the one-hour case in comparison with the 20 minute case.

9.2.2 Set 2 of Experiments

In this set of experiments we are going to repeat the above for different service rate allocation or the bandwidth available on the router for Geo_Info class packets.

The utilization has been found quite low (about 89% - 99% of time the queue is empty), so we are not looking at it in the Set 2 and Set 3.

Tables 9-2-5 – 9-2-8 show the maximum queue lengths for all movement scenarios for 20 minute and one-hour experiments. The plotted results for the maximum queue length for all scenarios are shown in Figure 9-2-8.

The results show that the 9 LOD *centre* case presents the worst case, as with increasing bandwidth the queue length remains high (e.g. 441 packet for 4 Mbit/s). The *join* scenario shows that the active approach shows considerable improvement in comparison with the non-active case (e.g. the queue length is 18 packets for the 9

LOD 4Mbit/s experiment in comparison with 832 for the non-active approach with the same service rate).

Table 9-2-5 Maximum queue length (Set 2 of experiments, *random* scenario)

Service rate (Mbit/s)	Maximum length of queue (packets)					
	Non-active		3 LOD		9 LOD	
	20 min	1 hour	20 min	1 hour	20 min	1 hour
2	1266	1266	1285	1285	548	1015
2.5	885	888	696	696	357	865
3	471	471	426	840	202	675
3.5	92	100	593	593	111	516
4	14	14	334	334	55	339

Table 9-2-6 Maximum queue length (Set 2 of experiments, *far* scenario)

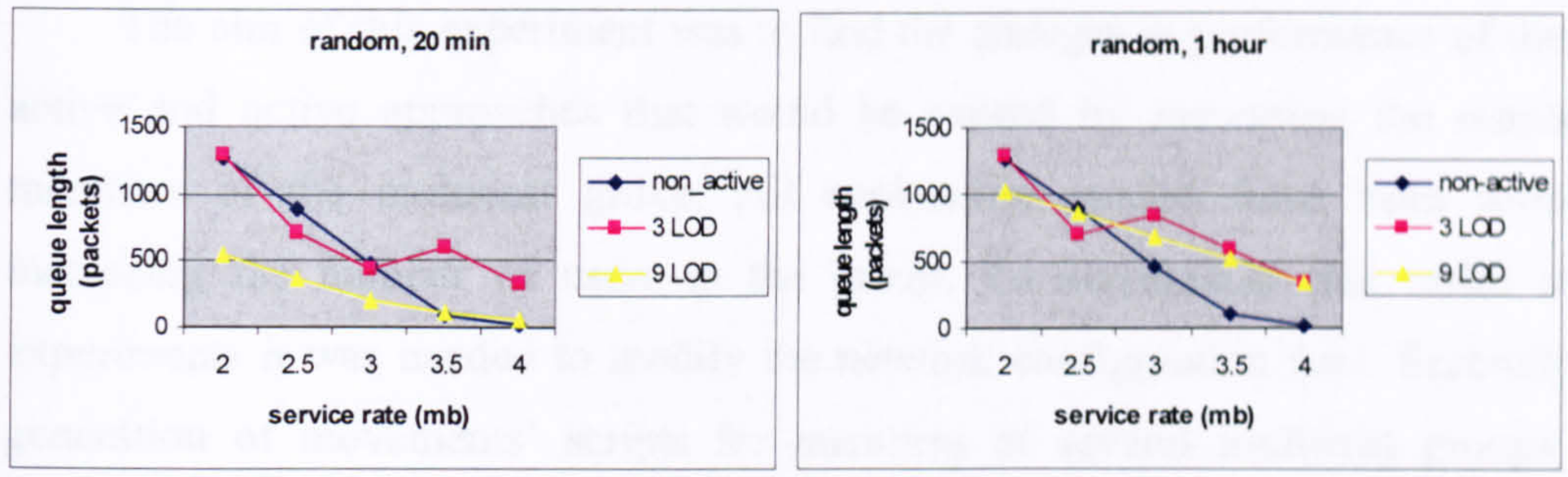
Service rate (Mbit/s)	Maximum length of queue (packets)					
	Non-active		3 LOD		9 LOD	
	20 min	1 hour	20 min	1 hour	20 min	1 hour
2	1221	1221	1019	1019	548	548
2.5	885	885	696	696	363	363
3	454	454	384	384	202	202
3.5	92	92	82	82	82	92
4	14	14	18	18	60	63

Table 9-2-7 Maximum queue length (Set 2 of experiments, *centre* scenario)

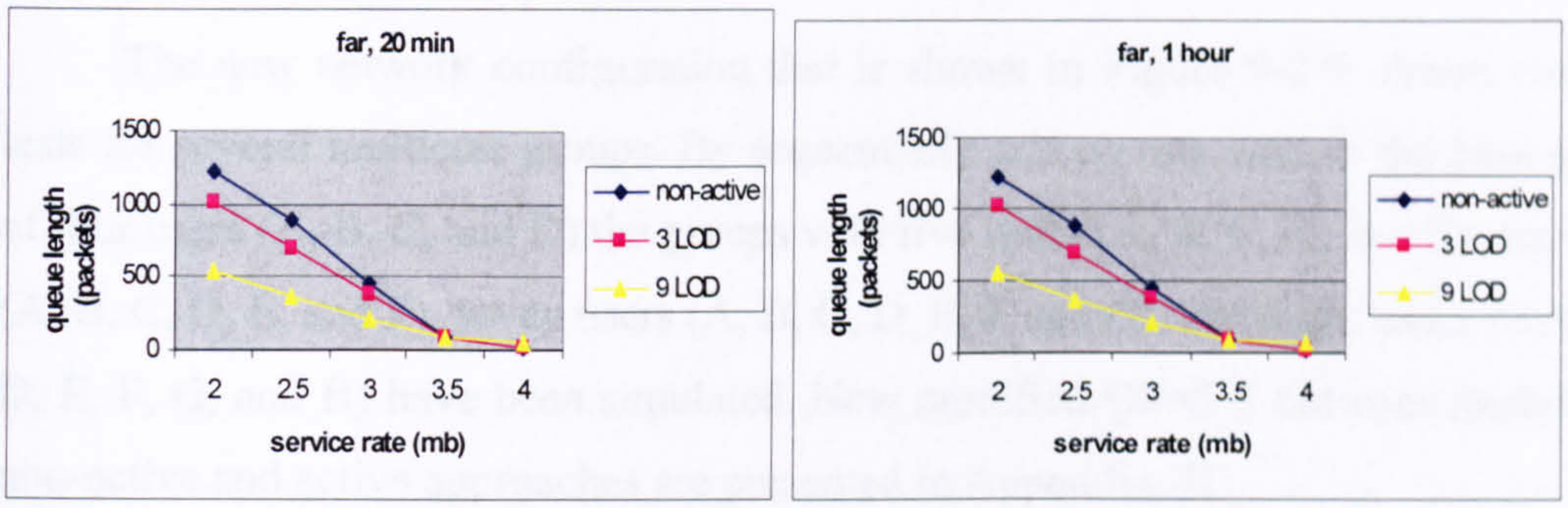
Service rate (Mbit/s)	Maximum length of queue (packets)					
	Non-active		3 LOD		9 LOD	
	20 min	1 hour	20 min	1 hour	20 min	1 hour
2	1221	1221	1019	1019	864	905
2.5	885	885	708	708	747	792
3	454	454	384	384	630	670
3.5	92	92	92	92	518	558
4	14	14	18	18	384	441

Table 9-2-8 Maximum queue length (Set 2 of experiments, *join* scenario)

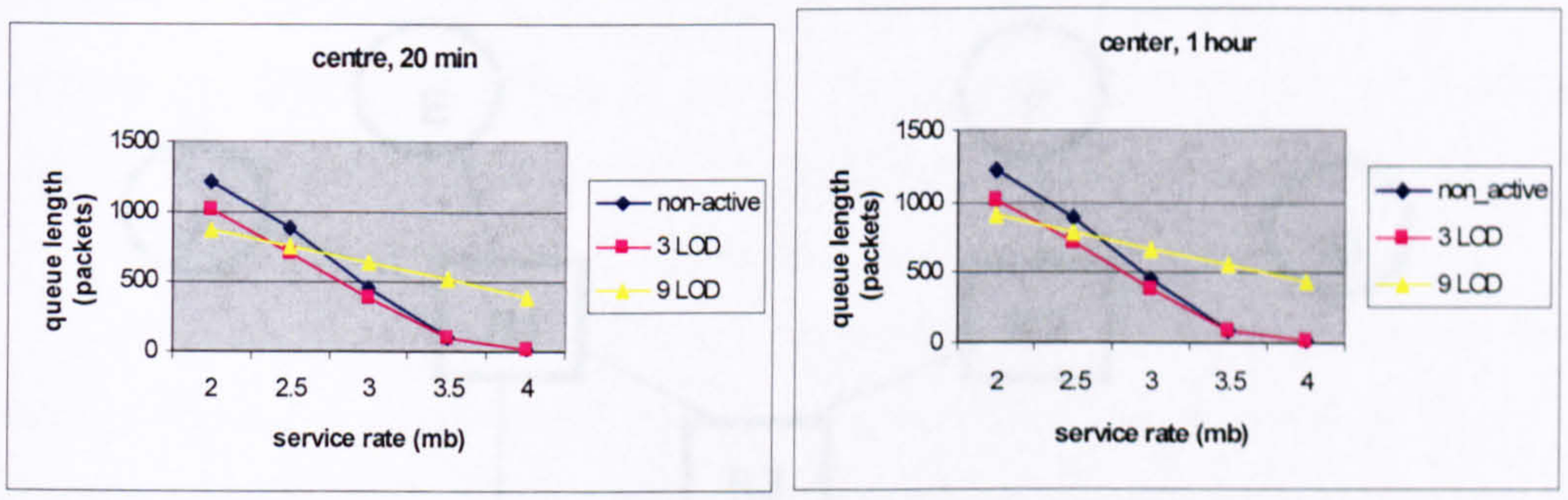
Service rate (Mbit/s)	Maximum length of queue (packets)					
	Non-active		3 LOD		9 LOD	
	20 min	1 hour	20 min	1 hour	20 min	1 hour
2	2251	2251	1041	1041	560	560
2.5	1772	1772	708	708	357	357
3	1316	1316	384	384	202	202
3.5	1060	1060	123	123	75	75
4	832	832	27	27	18	18



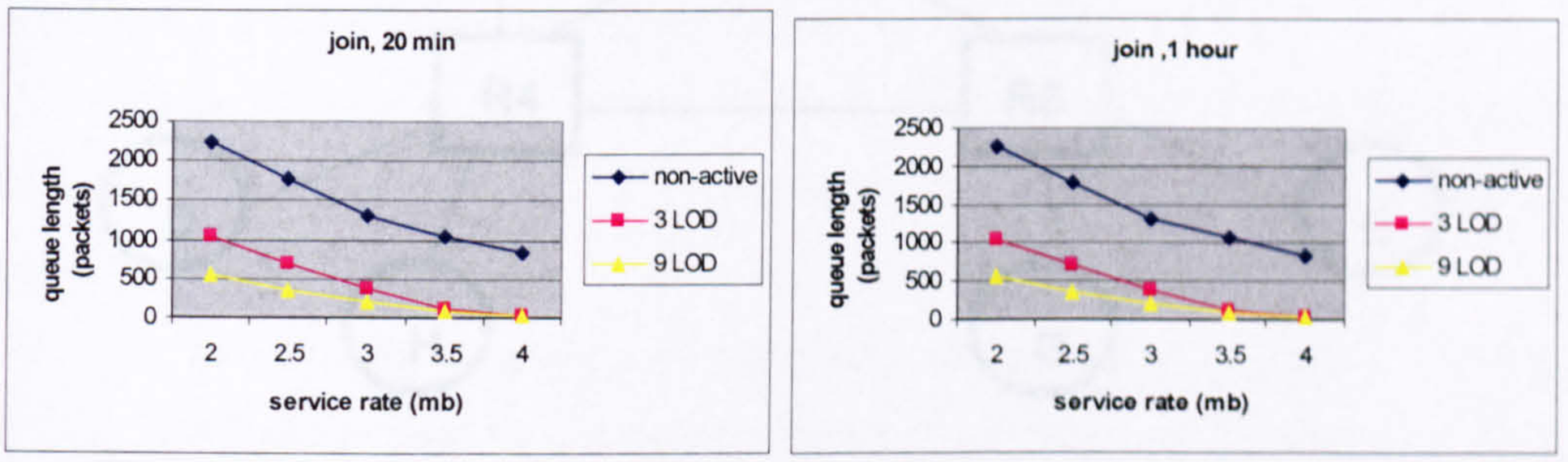
a)



b)



c)



d)

Figure 9-2-8 Maximum queue length results for increasing bandwidth

9.2.3 Set 3 of Experiments

The aim of this experiment was to find the changes in performance of the non-active and active approaches that would be caused by increasing the number of members of the multicast group. All application models have been tested on increasing the number of users in the group. To accomplish this series of the experiments it was needed to modify the network configuration first. Secondly, the generation of movements' scripts for members of several multicast groups with different sizes has been required.

The new network configuration that is shown in Figure 9-2-9 allows running tests for several multicast groups. By sequentially adding one user to the base group of four users (A, B, C, and D) the groups with five users (A, B, C, D, and E), six users (A, B, C, D, E, and F), seven users (A, B, C, D, E, F and G), and eight users (A, B, C, D, E, F, G, and H) have been simulated. New modified OPNET network models for non-active and active approaches are presented in Appendix III.

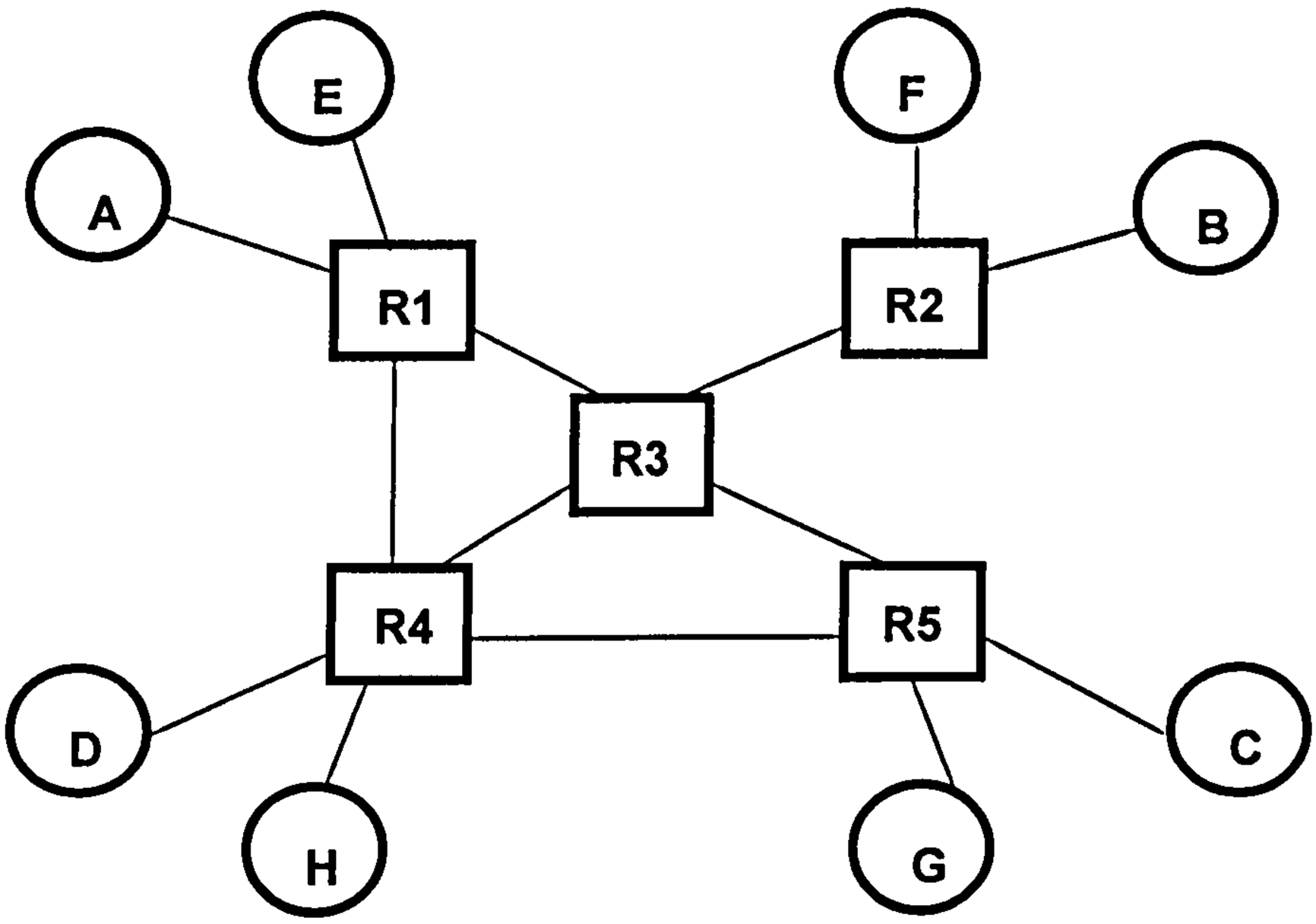


Figure 9-2-9 Network configuration that was used in experiments with different number of users

Figure 9-2-10 shows the initial placements of users for the eight users' series of experiments.

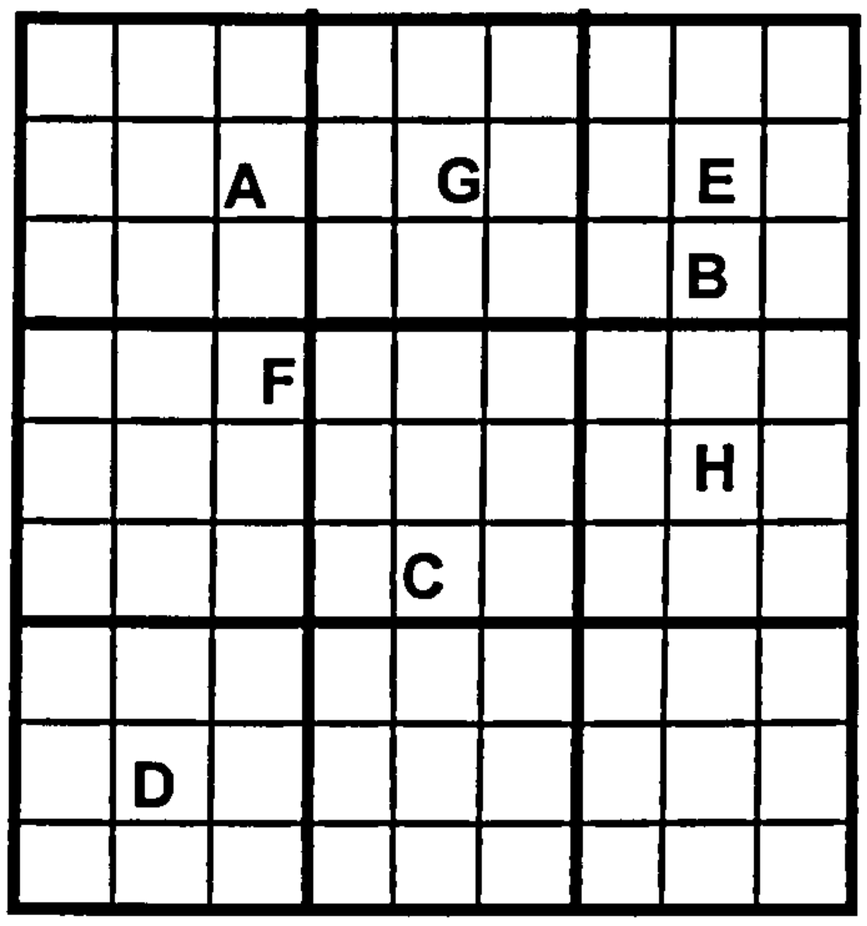


Figure 9-2-10 Initial placements of users that was used in experiments with different number of users

Movement scripts have been generated separately for these new multicast groups; because the number of members defines the number of control packets (to update the LOD look up tables on nodes and hosts) that a user sends to the group if it changes position in the grid. The scripts have been generated for all application models for different numbers of users. The examples of scripts are presented in Appendix II, the SVG visualization examples are presented in Appendix IV.

I. Testing of router's QoS parameters.

In this series of experiments the maximum queue length (queue occupancy) has been evaluated for increasing size of the group. The queue occupancy results for multicast groups with different numbers of users are presented for the node – router R3. Duration of the experiments was 20 minutes and one hour.

Tables 9-2-9 – 9-2-12 summarise the maximum values of the queue length that have been admitted on router R3 for groups with different sizes for all application models. Figures 9-2-11 – 9-2-14 plot these maximum queue results.

For some multicast groups we were not able to complete one-hour experiments for all application models due to the hardware limitations. However, we use the complete simulation runs' results to build graphs for the one-hour experiments. Incomplete runs' results are shown by '****' and the time when our simulation has reached the limit. Therefore, the one-hour maximum queue length results in Tables 9-2-9b – 9-2-12b show the stage where we can not go further.

Table 9-2-9a Maximum queue length (Set 3, *random*, 20 minutes)

Size of multicast group	Maximum length of queue (packets)		
	Non-active	3 LOD	9 LOD
4 users	1266	1285	548
5 users	1886	1504	832
6 users	2890	2141	1370
7 users	3857	2051	1349
8 users	4472	2341	1998

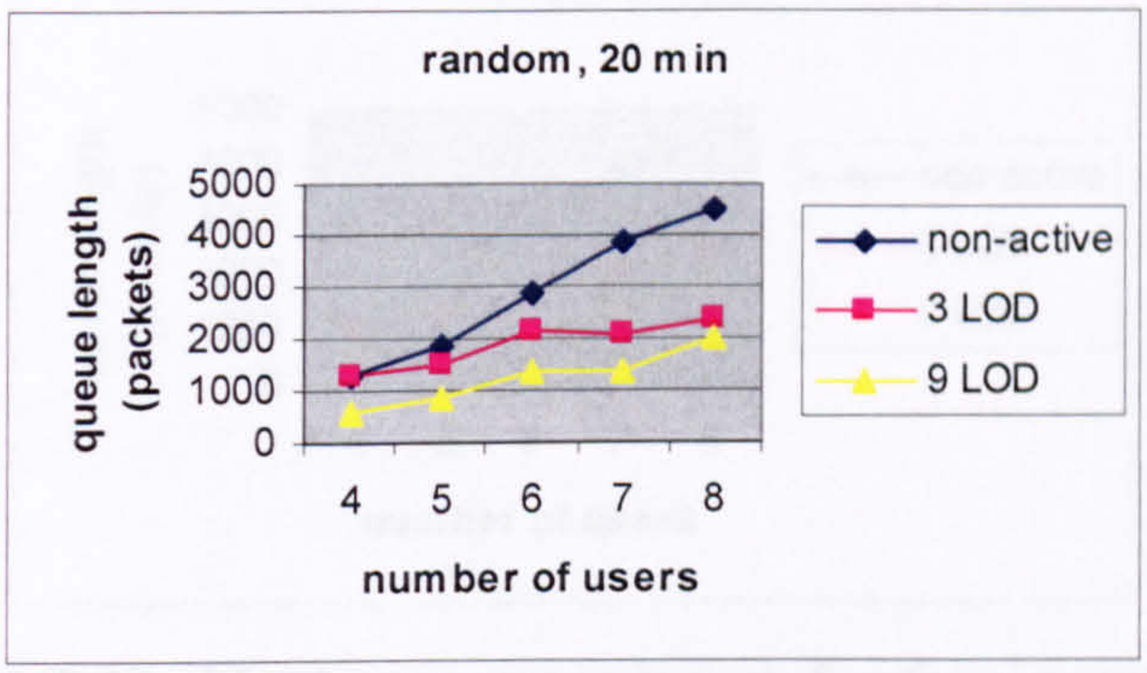


Figure 9-2-11a Maximum queue length for *random* scenario (from 4 to 8 users, 20 minutes)

Table 9-2-9b Maximum queue length (Set 3, *random*, one hour)

Size of multicast group	Maximum length of queue (packets)		
	Non-active	3 LOD	9 LOD
4 users	1266	1285	1015
5 users	2272	2086	846
6 users	2888	****	1370
7 users	4687	3226	1349
8 users	4969	****	****

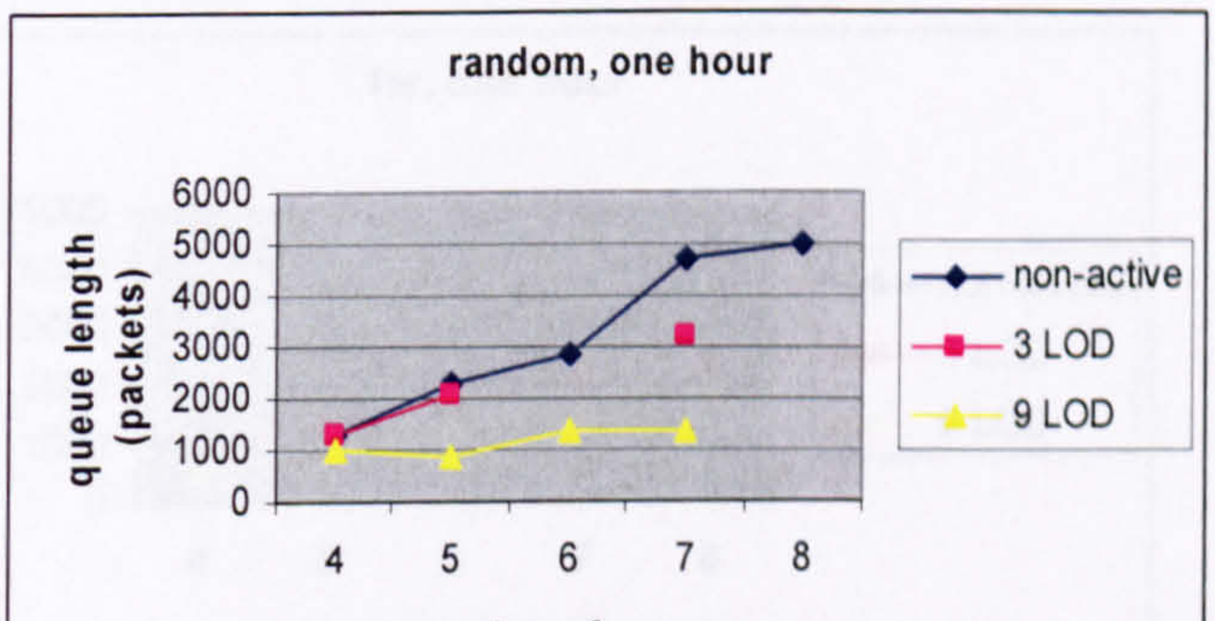


Figure 9-2-11b Maximum queue length for *random* scenario (from 4 to 8 users, one hour)

Table 9-2-10a Maximum queue length (Set 3, *far*, 20 minutes)

Size of multicast group	Maximum length of queue (packets)		
	Non-active	3 LOD	9 LOD
4 users	1221	1019	548
5 users	1864	1285	799
6 users	2890	1504	848
7 users	3846	2050	1347
8 users	4467	2341	1894

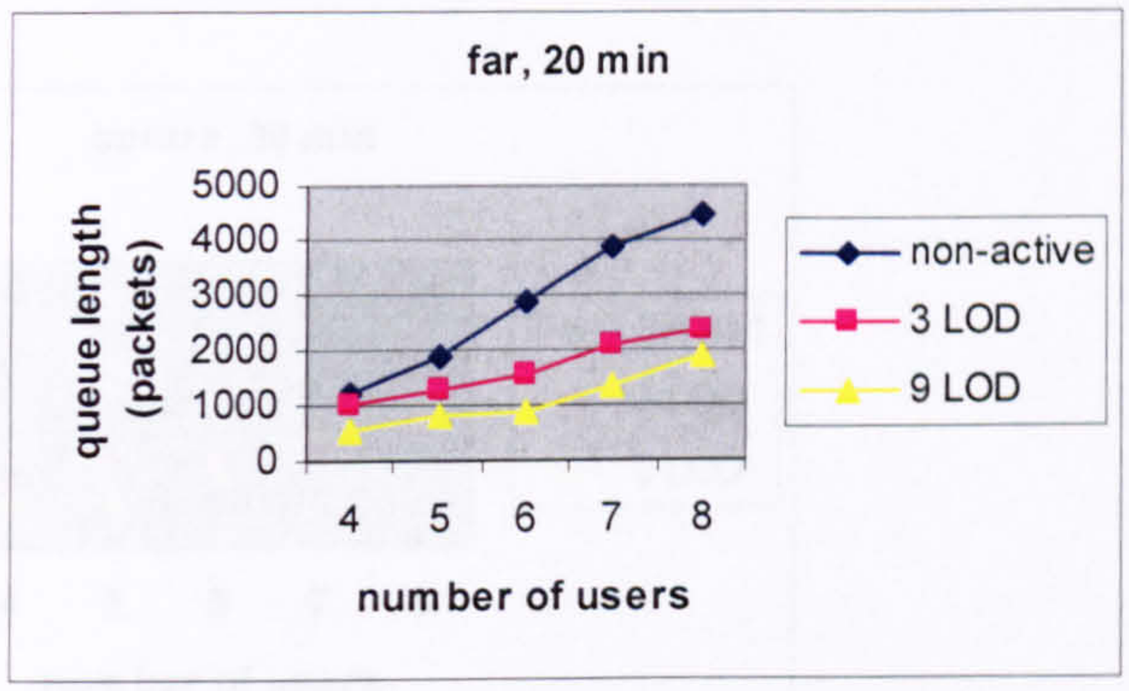


Figure 9-2-12a Maximum queue length for *far* scenario (from 4 to 8 users, 20 minutes)

Table 9-2-10b Maximum queue length (Set 3, *far*, one hour)

Size of multicast group	Maximum length of queue (packets)		
	Non-active	3 LOD	9 LOD
4 users	1221	1019	548
5 users	1864	1284	800
6 users	2890	1677	1286
7 users	3846	2050	1347
8 users	4467	2341	1894

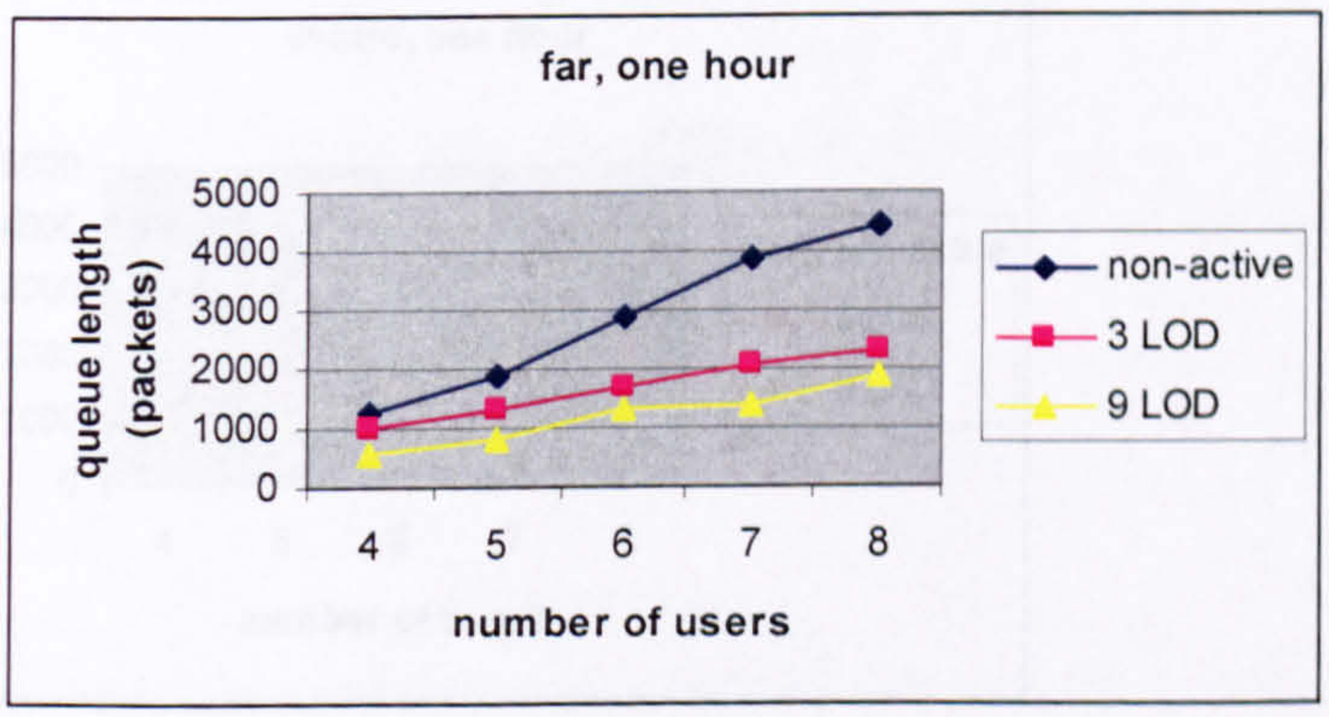


Figure 9-2-12b Maximum queue length for *far* scenario (from 4 to 8 users, one hour)

Table 9-2-11a Maximum queue length (Set 3, *centre*, 20 minutes)

Size of multicast group	Maximum length of queue (packets)		
	Non-active	3 LOD	9 LOD
4 users	1221	1019	864
5 users	1864	1527	833
6 users	2890	1415	1274
7 users	3846	1706	2014
8 users	4467	2074	2293

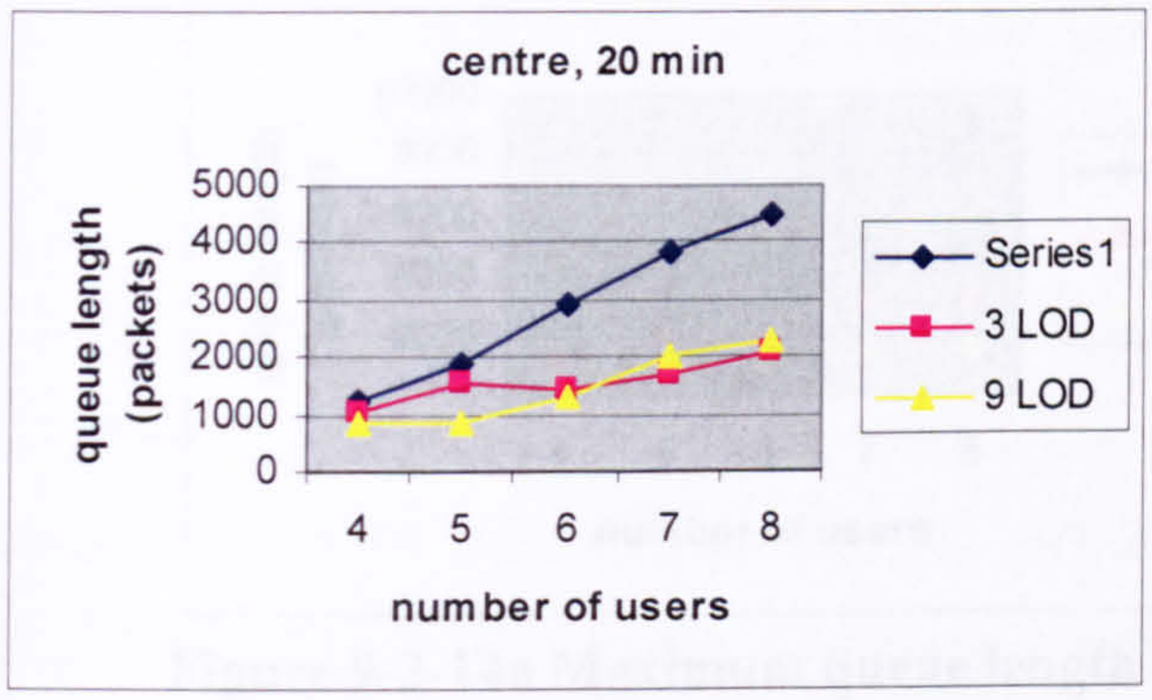


Figure 9-2-13a Maximum queue length for *centre* scenario (from 4 to 8 users, 20minutes)

Table 9-2-11b Maximum queue length (Set 3, *centre*, one hour)

Size of multicast group	Maximum length of queue (packets)		
	Non-active	3 LOD	9 LOD
4 users	1221	1019	905
5 users	1864	1526	916
6 users	2890	1415	1369
7 users	3846	1706	2014
8 users	4467	2074	****

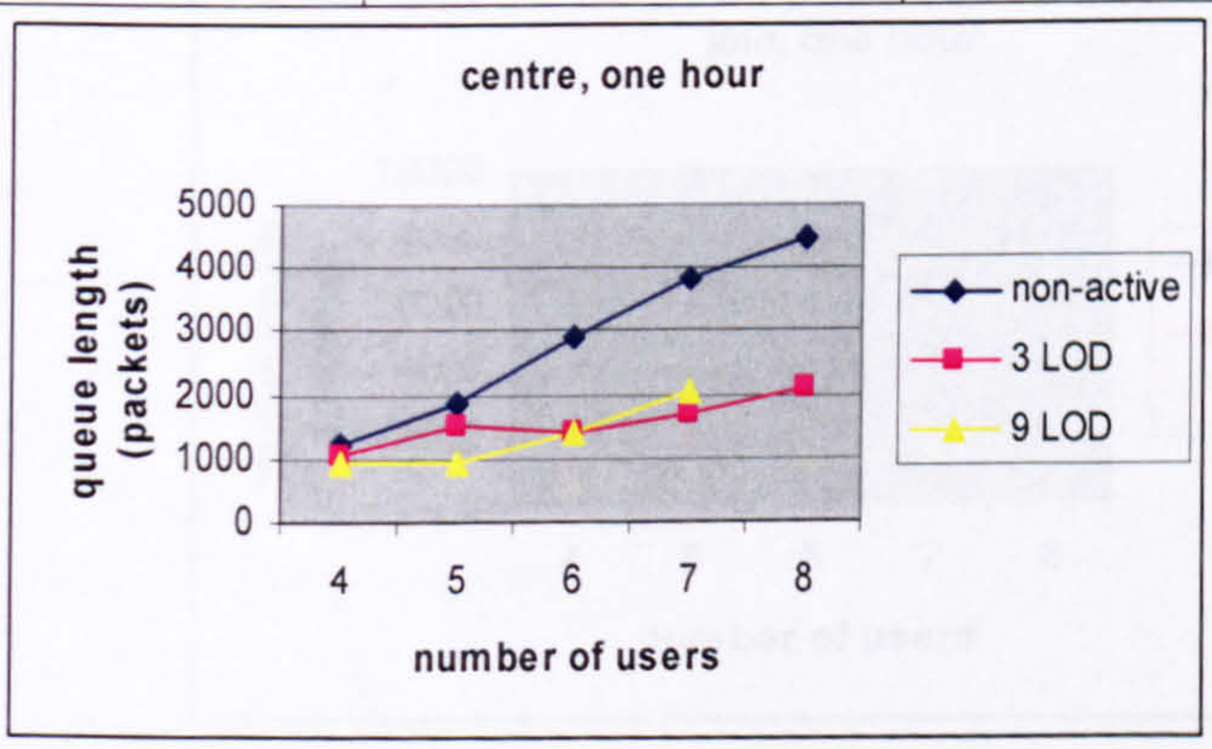


Figure 9-2-13b Maximum queue length for *centre* scenario (from 4 to 8 users, one hour)

Table 9-2-12 Maximum queue length (Set 3, *join*, 20 minutes)

Size of multicast group	Maximum length of queue (packets)		
	Non-active	3 LOD	9 LOD
4 users	2251	1041	560
5 users	2275	1894	833
6 users	3143	1679	1268
7 users	5061	2973	1569
8 users	8937	3011	1620

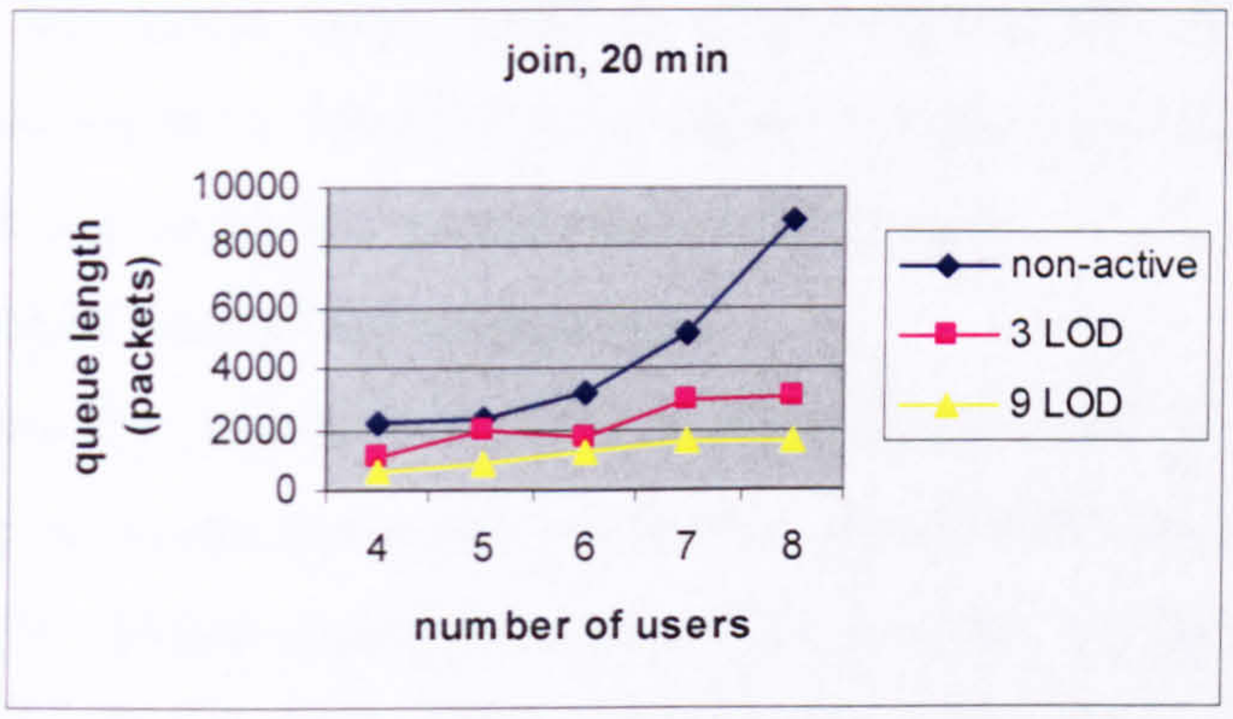


Figure 9-2-14a Maximum queue length for *join* scenario (from 4 to 8 users, 20 minutes)

Table 9-2-12b Maximum queue length (Set 3, *join*, one hour)

Size of multicast group	Maximum length of queue (packets)		
	Non-active	3 LOD	9 LOD
4 users	2251	1041	560
5 users	3003	1893	831
6 users	3241	****	1267
7 users	6493	****	1569
8 users	8937	****	****

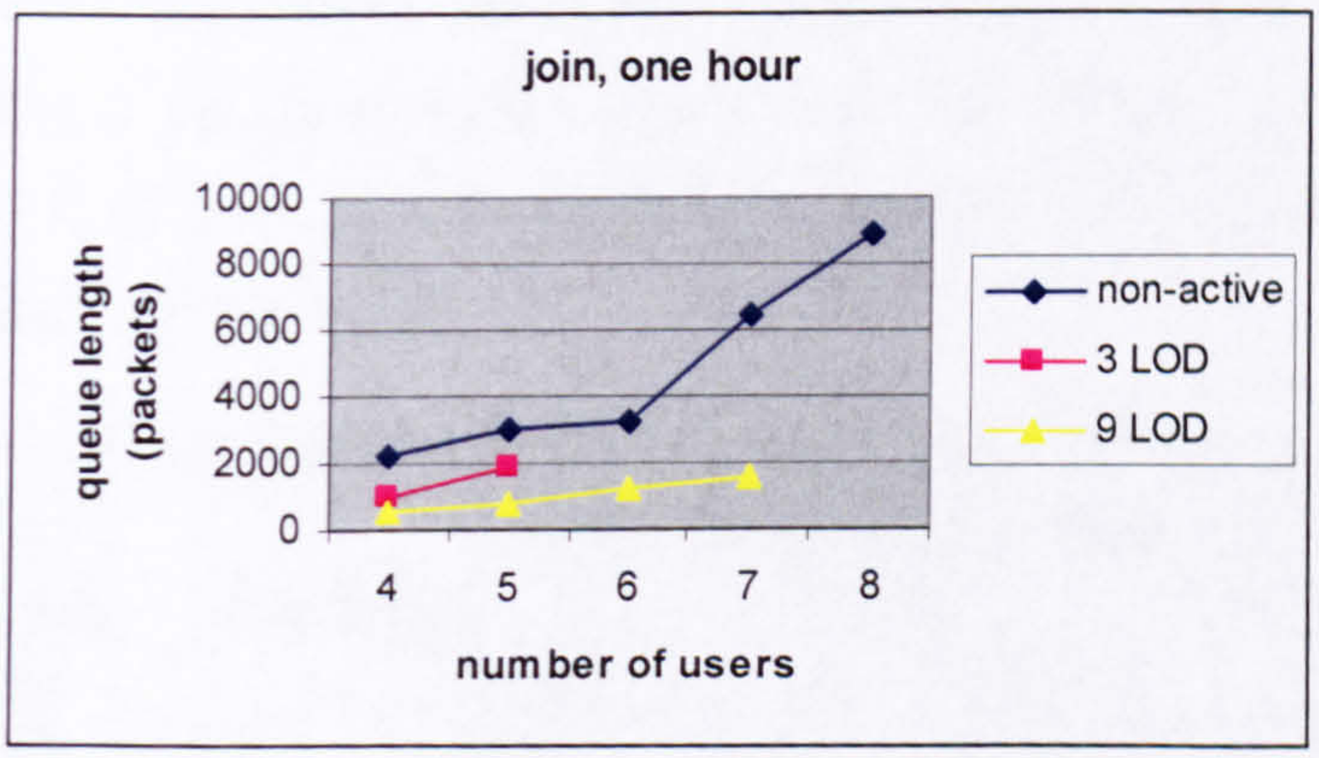


Figure 9-2-14b Maximum queue length for *join* scenario (from 4 to 8 users, one hour)

For this case all application scenarios show a better performance for the active approach than the non-active case with increasing the scale of the application. The results for 20 minute experiments have been obtained for the multicast groups with number of users up to eight. The *random*, *far*, and *centre* 9 LOD active cases for eight users' group show the growth of the queue length by about 50% less than the corresponding non-active experiments do. Furthermore, the *join* scenario for the similar group shows more than five times decrease in the queue length for the 9 LOD active approach in comparison with the non-active result.

II. Testing of host's QoS parameters.

To evaluate the performance of the active approach with increasing numbers of multicast group members the host's parameters along with nodes performance have been tested. The initial reception time which defines the waiting time to start interaction with other users' representations has been analysed. The results are presented for one of the hosts - user A.

In the base *4 users* group each user receives representations of the three other users. To keep consistency of the results for all other multicast groups the average reception time is also calculated for the same number of received representations. For example, the average initial reception time for user A has been calculated as the average of the reception times of the representations of users B, C, and D for all experiments.

Table 9-2-13 summarises the average initial reception times that have been recorded on host A for groups with different sizes for the three approaches tested (non-active, 3LOD and 9LOD active). Figure 9-2-15 shows the reception time results that are presented in Table 9-2-13.

Table 9-2-13 Average initial reception time on host (Set 3 of experiments)

Size of multicast group	Average initial reception time on host (s)		
	Non-active	3 LOD	9 LOD
4 users	21.629	12.294	5.976
5 users	27.574	14.213	7.013
6 users	31.824	17.736	8.816
7 users	36.816	21.549	12.298
8 users	42.741	24.862	13.262

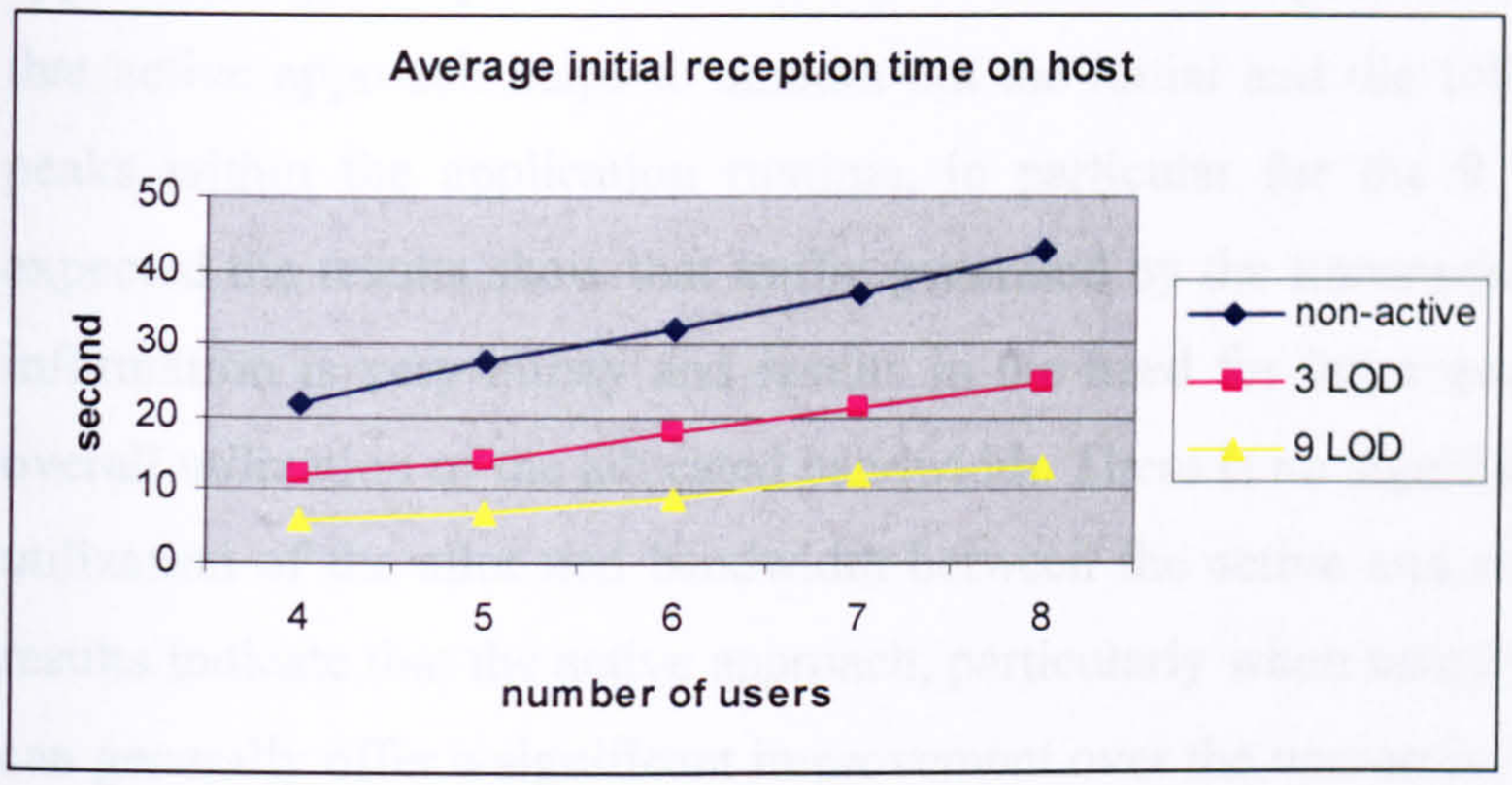


Figure 9-2-15 Average initial reception times on host for increasing number of users (user A)

9.3 Discussion of results

In this chapter the evaluation of the active architecture performance within the post initialization stage of the application has been presented. The evaluation process of this stage of the application has been divided into two steps.

First step includes the evaluation of the complete active networking architecture using the *random* application model that is used as a basic application scenario to test the accuracy of the complete architecture simulation model. The second step includes examining of the different application models using changes to the duration of experiments, allocated bandwidth and the number of users.

The results obtained in the basic scenario experiments show that simulation models work properly and it is generally demonstrated by the changing behaviour of the queue occupancy on routers R1, R3, and R4. The peaks in the queue occupancy and the quiet periods coincide with general patterns of the traffic load as generated by the application. Also to examine a correctness of the complete architecture model more precisely the sent and received by the users flows of packets have been monitored and the obtained results show that they fulfil the movement pattern described by the *random* application model. The comparison between the non-active and active approaches for the post initialization stage of the application show

considerable decrease of the amount of overall transmitted packets within the active approach, especially for the 9 LOD case. The routers' queue occupancy results show that active approach helps to smooth out the initial and the following queue length peaks within the application runtime, in particular for the 9 LOD case. As was expected the results show that traffic generated by the transmission of the geometric information is very bursty and results in the need for large queue sizes, but a low overall utilization of the allocated bandwidth. There is no significant difference in the utilization of the allocated bandwidth between the active and non-active cases. The results indicate that the active approach, particularly when using nine levels of detail, can generally offer a significant improvement over the non-active approach in respect to queue size requirements. However, even with the improvements offered by the active approach this type of traffic still required quite large queue sizes.

The second step of the evaluation of the active approach includes an examination of the different application models using changes to the duration of experiments, the bandwidth allocated for Geo_Info class packets on routers, and the number of users. This step consists of three sets of the experiments.

The first set of experiments (Set 1) objective was the comparison between the different application models' performances using changed duration of experiment. Within these experiments the performance on routers and hosts has been observed.

The memory requirements to store 3D representations of other users on hosts have been evaluated. The results show the certain memory usage patterns for every application model that were unchanged within the longer duration experiments. The maximum memory usage for the non-active and active approaches can be predicted, so the difference between the maximum memory requirement and the memory usage required within the certain movement scenario application model could be reallocated to store representations of the larger number of users. Therefore, these memory usage results show the possibility of the active approach to improve a scalability of DVE applications. However, the memory requirements results are preliminary and need more detailed investigation in the further research.

The routers' queue occupancy results have been obtained. The QoS parameters that have been observed on routers are: the maximum queue length and

the utilization of the queue. The maximum queue length and the utilization of the queue (illustrated by results for the router R3) have shown that a degree of the improvement achieved by the active approach is dependent on the movement pattern of users. It was found that the *join* scenario shows the largest decrease of the maximum queue length in comparison with the non-active case. However, the *centre* scenario demonstrates the smallest decrease of the maximum queue length in comparison with the non-active case. The *centre* scenario that could be classified as the worst case shows also the increase in the queue utilization. In contrast the *join* scenario demonstrates the decrease in the queue utilization and could be classified as the best case.

Comparison of the one-hour queue occupancy results with 20 minute results demonstrates the continuation of the similar traffic patterns almost for all application scenarios for the longer duration of the experiment. Except the *random* case which shows a radical increase in the maximum queue length metric for the 9 LOD experiment. This high peak could be caused by the invocation of the several single batch in addition to the sequence of batches generator as several users could approach the sender and a new users join the group within one time slot. This finding reveals a drawback of the active approach that has to be eliminated within the further stages of the active networking architecture development.

In Set 2 of the experiments all experiments from the first set have been repeated with increasing the available for Geo_Info class packets bandwidth. The Set 2 results show that the *join* scenario is the best case, as the maximum queue length decreases better for the active approach than for the non-active case with increasing of the available bandwidth. However, the *random* (one-hour) and *centre* (20 minutes and one-hour) scenarios experiments are worst cases, because the maximum queue length decreases for the non-active approach better for these two application models with increasing of the available bandwidth. With increasing of the available bandwidth the benefits of the active approach disappear for some movement scenarios. However, for some cases (*join* scenario) the active approach still shows benefits. The reason for this performance is that there is a very complex interactivity relationship between the DVE users that needs a further study.

Set 3 of experiments has been carried out to evaluate the proposed active approach with increasing the numbers of users. Within these experiments the maximum queue length on the router and the initial reception time on the host have been observed. All application scenarios show a better performance for the active approach than the non-active case with increasing the scale of the application. The *random*, *far*, and *centre* 9 LOD active cases for the multicast group that consists of eight users show down the growth of the maximum queue length in comparison with the multicast group with four users by about 50% less than the corresponding non-active experiments do. Furthermore, the *join* scenario for the eight users group shows more than five times decrease in the queue length for the 9 LOD active approach in comparison with the non-active result. The growth of the initial reception time on the host with increasing the scale of the application is also slowed down within the active approach. For example, the average initial reception time for the eight users group within the 9 LOD experiment is less than the non-active result by more than three times.

The experiments for the increased numbers of users also have been carried out for the one-our duration experiments. However, we could not complete the simulation runs for all application models up to the maximum used number of users due to the system resources available to OPNET limitations.

Chapter 10

Conclusion

The work in this thesis has been motivated by the need to find a solution to improve the scalability of the DVE applications by a consideration the QoS requirements of the 3D DVE data type.

Recently the Internet has developed into the most common embedding source for DVEs with the standard Web browsers used as an execution engines for the DVE application, so the large-scale DVEs are attracting increasing attention from the networking research community.

The emergence of the new Internet applications such as large-scale online games that can be regarded as a derivative of DVEs and the availability of the broadband Internet DSL high-speed, always-on connection for the growing number of the end users let the large-scale DVE applications be available for significantly increased numbers of users. There is an expectation that persistent, large-scale, distributed virtual environments inhabited by millions of entities will emerge.

This vision for DVE applications would be achieved by improving the scalability issues using the numerous resource management techniques.

The network bandwidth resource is the more limited resource and its shortage can cause the scalability problems for the DVEs. To achieve the saving in the network bandwidth and therefore to increase the scalability of the DVE the different types of the network traffic that is produced by the DVEs have to be considered. The DVEs generate the traditional media traffic as the computer data (e.g. state update messages) and continuous types (e.g. interactive audio and video), that have already been considered by existing research. However, the 3D data generated by DVEs has not really been covered.

In this work the research that aims to handle this particular kind of the DVE traffic has been carried out.

The size of the geometrical representations of the 3D objects that now can be measured by millions of triangles defines very bulky character of the 3D data. Furthermore, the interactive DVE applications require that this geometrical

information has to be delivered as fast as it possible. The resulting 3D geometry traffic will be very bursty in nature and will place a high demands on the network for short intervals of time.

In this work we are investigating the possibility to decrease the network bandwidth utilization by the 3D DVE traffic using the LOD concept and the active networking approach.

A set of the object's approximations, where each approximation represents the original object with a different level of detail (LOD), is used to describe a 3D geometry. As only one level of detail of a given object can be displayed at any time instead of transmission of geometry data for an object as a whole, it is possible to transmit exactly packets with certain LOD to the receivers who needs to view the object with this certain level of fidelity. Therefore, the restriction of the unnecessary 3D flows would be achieved.

We propose to exploit the active networks to accomplish the filtering of the unnecessary 3D flows which is based on the spatial placements of the users in the virtual world on the intermediate routers where it is more efficient.

The new active networking approach to the transmission of the 3D geometry data within the DVE systems has been proposed in this thesis. This approach enhances the currently applied peer-to-peer DVE architecture by adding to the peer-to-peer multicast network layer filtering of the 3D flows an application level filtering on the active intermediate nodes. The active router keeps the application level information about the placements of users. This information is used by active routers to prune more detailed 3D data flows (higher LODs) in the multicast tree arches that are linked to the distance DVE participants.

To evaluate the proposed active approach through the comparison with the non-active approach the performance modelling evaluation approach which is represented by the framework for evaluation has been applied. The developed framework for evaluation consists of the architecture modelling, flow and application modelling, defining performance metrics, design of simulation experiments and simulation-based evaluation of effectiveness of proposed approach.

The evaluation has been carried out in two stages. First, the proposed approach has been evaluated within the initialization stage of the DVE application. Within this stage the active approach has shown that it performs better than the non-active approach. The maximum queue size on the router and the initial reception time on the host have been decreased especially using the 9 LOD scenario.

Second, the proposed approach has been evaluated within the post initialization stage of the DVE application. In this stage the complete model of the active networking architecture has been evaluated. Initially the basic experiments using the *random* application model have been accomplished. The active approach has shown the better performance by smoothing out the initial an all following peaks in the queue occupancy on the router apparently for the 9 LOD scenario.

As the proposed active approach relies on the spatial placements of users, the investigation has been continued using the different patterns of users' movements that have been represented as the application models. It was found that the *centre* scenario (or users' cluster) can be classified as the worst case, because the active approach benefits are minimal among all other scenarios in comparison with the non-active approach. In contrast the *join* scenario (users tend to leave and rejoin the multicast group frequently) has been defined as the best case where using the active approach is more efficient. However, the repetition of the experiments using the longer duration has uncovered the radical increase of the maximum queue length for the *random* scenario, when this metric almost reaches the non-active approach value. This could be caused by the invocation of the several single batch in addition to the sequence of batches generator as several users could approach the sender and a new users join the group within one time slot. This finding has to be considered within the further stages of the active networking approach development.

The active approach has shown the diverse performance for different application models within the experiments with increasing the available for Geo_Info class packets bandwidth. The *join* scenario has been found the best case, as the maximum queue length decreases better for the active approach than for the non-active case with increasing of the available bandwidth. The *far* scenario has been shown to be the similar to the non-active case performance. However, the *random*

(one-hour) and *centre* (20 minutes and one-hour) scenarios experiments are worst cases, because the non-active approach performance has been found to be better than the active approach performance for these application models. The reason for this performance is that there is a very complex interactivity relationship between the DVE users that needs a further study.

The last part of the evaluation process aimed to examine the proposed active approach by increasing the scale of the DVE application. All application scenarios have shown better performance for the active approach than the non-active case with increasing the scale of the application. The growth of the maximum queue length on the router and the initial reception time on the host with increasing the scale of the application have been shown slowed down within the active approach for all application models in comparison with the non-active approach. Therefore the proposed active approach brings the improvement to the DVE's scalability by reducing of the cost of increases in the scale that is achieved by saving the network bandwidth resource.

10.1 Research Contributions and General Findings

The main contribution of this research is the evaluation of possible benefits of exploiting an active networks and a LOD concept within the transmission of the 3D DVE data traffic and the investigation to how this active approach could help to improve the scalability issues of the large-scale DVE applications through the comparison with the non-active approach.

Comparison between the results for the active and non-active approaches for the DVE applications has led to conclusions about the viability of the proposed active approach.

The active approach proposed in this work assumes the per-class resource allocation with the multi-service possibility where the 3D DVE geometry data has its own Geo_Info class. The active approach adds a flexible, application-aware, programmable service for 3D DVE data Geo_Info class.

The proposed active approach makes savings in the bandwidth resource in more cases than it does not. The degree of the benefits shown by the active approach depends on the application model which is defined by the users' movement pattern. Therefore, the proposed active approach can not be defined as a fully sufficient by its own. We did not set out to prove by our research is it better method or not. However, we have found that the proposed active approach is generally beneficial. So, this research lets us to look forward to apply other active networking methods to improve the 3D DVE data transmission. It could be possible, for example, to prevent the worst case performance (e.g. users' clusters movement pattern performance) by adding a prediction of user's movement and caching of the 3D DVE geometry on active routers.

This research has also led to the following individual contributions:

- Development of the plausible architecture which combines the progressive 3D streaming and the active networking approach in order to improve 3D geometrical data type transmission within the large-scale DVE applications.
- Development of the approach to modelling behaviour of the users' movement in the virtual world.
- Development of a novel algorithm to generating different scenarios of users' movement in DVE.
- Development of the SVG visualization tool to observe these scenarios.

10.2 Further Work

The work carried out in this thesis has shown that some potential benefits of the proposed active networking approach to large-scale DVEs have been identified and it is worthwhile to continue with this approach. However, as it was found for some of movement scenarios (e.g. *centre* scenario or clusters of users) that the nature of the 3D DVE geometrical type of traffic is still bursty and needs large queue sizes even using the active approach. Therefore, other methods to support the active 3D geometry transmission within the DVE may also be required.

As a further work an improvement of the performance of the proposed active networking could be investigated. This further stage of the work would include two directions.

First, the work done could be improved by the improvement of the research methodology including applying better simulation methods and the better deployed hardware. The OPNET limitation on the number of the generated and transmitted packets in the DVE application simulation has been detected within the one-hour experiments for the increased number of users. To overcome this limitation in the future work a different simulation platform would be used. Furthermore, the possible direction to continue this research to write our own customized simulation tool. This could allow us to enlarge the scale of the simulation and observe the performance of the proposed approach for larger numbers of the DVE users.

Second, the exploiting other solutions of the active network processing for the 3D DVE geometrical data type could be investigated. The predicting of the users' behaviour could support the caching of the 3D data on active routers. The caching mechanisms on active routers could help to retransmit highly detailed 3D batches not by the sender but by the nearest to the receivers active router. The compressed progressive meshes (CPM) that aim to minimize the size of the 3D geometry and are used in our work as the flow model prototype are appropriate for caching. Developed in this work active filtering mechanisms could help routers to invoke the transmission of the cached 3D data. As all the application layer information about the spatial placements of users is kept and updated on routers in the form of the LOD lookup table and the higher batches of the users' representations could be cached on this node, the active router could retransmit additional required more detailed 3D flows acting as a sender host which is placed in the network more closer to the receivers. This technique would additionally reduce the network bandwidth utilization, because the active routers would use a partial multicasting to limit the scope of the retransmitted packets only to the receivers that are closer to the active router in terms of the network. The cached data would be defined by the needs of the subgroup of hosts that are placed close to the given active router in the network and updated in respect to the changes in the application data of users' placements in the virtual

the virtual world. Therefore, by applying caching the application data of users' placements in the virtual world could be mapped more precisely on the physical network topology.

The reliability of 3D DVE transmission could also be improved using caching of the 3D compressed data on the active routers. The CPM 3D geometry streams are very sensitive to packet losses. The reconstruction of more detailed mesh by the host requires the reception of all packets from the refinement batch. At the same time, if the retransmission of lost packets will take a long time, the waiting time to view the object up to the required LOD will be increased. Caching and retransmission of the 3D lost packets by the active routers could help to decrease the delays of the repair retransmission as it is proposed by the active reliable multicast (ARM) [Lehman, 98].

References:

[ABONE] ABONE. Active network Backbone. http://www.csl.sri.com/ancors/abone

[Abrams, 98] H. Abrams, K. Watsen, M. Zyda. Three tiered interest management for large-scale virtual environments. In Proceedings of Virtual Reality Systems and Technology (VRST) 1998, ACM, Taipei, 1998.

[Alexander, 98] D.S. Alexander, W.A. Abraugh, M.A. Hicks, P. Kakkar, A. Keromytis, J.T. Moore, S.M. Nettles, J.M. Smith. The SwitchWare Active Network Architecture. IEEE Network Special Issue on Active and Controllable Networks, vol.12 no.3, 1998.

[ANEP] Active Network Encapsulation Protocol. RFC, http://www.cis.up-enn.edu/~switchware/ANEP/docs/ANEP.txt, July 1997.

[ANW, 98] Active Networks Working Group. K.L. Calvert (Ed.). Architectural Framework for Active Networks. Draft, August 1998.

[Balikhina, 2002a] T. Balikhina, F. Ball, D. Duce. Distributed Virtual Environments – An Active Future? In Proceeding of EUROGRAPHCS UK 2002, Leister June 2002.

[Balikhina, 2002b] T. Balikhina, F. Ball, D. Duce. Active Network Support for distributed Virtual Environments. In Proceedings of the 3rd annual Postgraduate Symposium on Convergence of Telecommunications, Networking and Broadcasting, Liverpool John Moores University (17th –18th June 2002) UK.

[Balikhina, 2004] T. Balikhina, D.Duce, A. Maqousi and F. Ball. Level of Detail Filtering in Active Networks to Support Distributed Virtual Environments. 20th UK Performance Engineering Workshop, University of Bradford, July 2004.

[Barrus, 96] J. Barrus, R. Waters, D. Anderson. Locales and beacons: Efficient and precise support for large multi-user virtual environments. In Proceedings of the 1996 Virtual Reality Annual International Symposium (VRAIS), IEEE Neural Networks Council, Santa Clara, 1996.

[Broll, 98] W. Broll. SmallTool – a Toolkit for Realising Shared Virtual Environments on the Internet. Distributed Systems Engineering Journal, Special Issue on Distributed Virtual Environments, Vol.5, 1998.

[Campbell, 99] A. Campbell, et al. A Survey of Programmable Networks. ACM SIGCOMM Computer Communication Review, vol.29, no.2, Apr.1999.

[DARPA, 98] DARPA Active Network Research Program. Active Networks CBD Reference BAA #98-03), 1998.

[Das, 97] T.K. Das, G. Singh et al. Developing Social Virtual Worlds using NetEffect 6th IEEE Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises, MIT, Cambridge Massachusetts, 1997.

[Decasper, 99] D. Decasper. A Software Architecture for Next Generation Routers. Ph.D. Thesis, ETH Zurich, May 1999.

[Deering, 95] M.Deering. Geometry Compression" (SIGGRAPH'95 Proceedings), 1995, p13-20.

[Diehl, 2001] S.Diehl. Distributed Virtual Worlds. Springer 2001.

[DIS, 93] Standard for information technology, protocols for distributed interactive simulation. DIS-ANSI IEEE Standard 1278-1993, American National Standards Institute, 1993.

[Durand, 99] A. Durand, B. Buclin. RFC2546, 6Bone Routing Practice. October 2002 http://ieft.org/rfc/rfc2546.txt

[Eichler, 2000] G. Eichler, H. Hussmann, G. Mamais, C. Prehofer, and S. Salsano. Implementing Integrated and Differentiated Services for the Internet with ATM Networks: A Practical Approach. IEEE Communications Magazine, pp. 132-141, 2000.

[Enbaya, 2002] Enbaya Inc. Technology Overview, January 2002 www.enbaya.com

[Eriksson, 94] H. Eriksson. MBONE: The Multicast Backbone. Communications of the ACM, 37(8): 54-60, 1994.

[FAIN] FAIN project WWW server – May 2000. October 2003 https://face.ee.ucl.ac.uk/fain/

[Floyd, 97] S. Floyd, V. Jacobson, Ching-Gung Lin, S.McCanne and Lixia Zhang. A Reliable Multicast Framework for Light-weight Sessions and Application Level Framing. IEEE/ACM Transactions on Networking, December 1997.

[Frecon, 98] E. Frecon, M. Stenius. DIVE:A Scalable network architecture for distributed virtual environments. Distributed Systems Engineering Journal, Special Issue on Distributed Virtual Environments, Vol.5,No 3,Sept.1998.

[Funkhouser, 95] T.A. Funkhouser. RING: A Client-Server System for Multi-user Virtual Environments. Symposium on Interactive 3D Graphics, Monterey, CA USA, 1995.

[Gao, 2000] J. Gao, P. Steenkiste, E. Takahashi, and A. Fisher. A Programmable Router Architecture Supporting Control Plane Extensibility. IEEE Communications Magazine, 38(3):152-159, March 2000.

[Greenhalgh, 97] C. Greenhalgh and S. Bendford. Boundaries, awareness, and interaction in collaborative virtual environments. Proceedings of the Sixth IEEE Workshop on Enabling Technologies: Infrastructure for Collaborative Enterprises (WETICE), 193-198. IEEE Computer Society, Cambridge, MA. June 1997.

[Greenhalgh, 98] C. Greenhalgh. Large Scale Collaborative Virtual Environments. Springer, 1998.

[Grosso, 98] R.Grosso, T.Ertl. Progressive Iso-Surface Extraction from Hierarchical 3D Meshes, Eurographics'98, Vol.17, No 3.

[Guskov, 99] I.Guskov, W.Sweldens, P.Schröder. Multiresolution Signal Processing for Meshes. (SIGGRAPH'99 Proceedings), 1999, p325-334.

[Hamza-Lup, 2004] F. G. Hamza-Lup and J. P. Rolland. Adaptive Scene Synchronization for Virtual and Mixed Reality Environments. In IEEE Virtual Reality 2004, Chicago, IL, March 2004.

[Hagsand, 96] O. Hagsand. Interactive Multiuser VEs in the DIVE System. IEEE Multimedia Magazine, Vol.3, No.1, 1996.

[Hearn, 97] D.Hearn, M.P. Baker. Computer Graphics. Second edition, Prentice Hall, 1997.

[Heckbert, 94] P. S. Heckbert, M. Garland. Multiresolution modelling for fast rendering. In Proceedings of Graphics Interface'94, pp43-50, 1994.

[Hicks, 98] M. Hicks, et al. PLAN: a Programming Language for Active Networks. In Proceedings of ICFP'98, 1998.

[Hoppe, 93] H.Hoppe, T.DeRose, T.Duchamp, J.McDonald, W.Stnezle. Mesh Optimization. Computer Graphics (SIGGRAPH'93 Proceedings), 1993, p19-26.

[Hoppe, 96] H.Hoppe. Progressive Meshes. Computer Graphics (SIGGRAPH'96 Proceedings), 1996, p99-108.

[Hubbold, 96] R. Hubbold, X. Dongbo, et al. MAVERIK – the Manchester Virtual Environment Interface Kernel. 3rd Eurographics Workshop on Virtual Environments, Monte-Carlo, 1996.

[ISO, 84] ISO. Information Processing Systems: Open System Interconnection – Basic Reference Model. Technical report, ISO/IEC, 1984.

[Kali, 2003] Kali server. November 2004http://www.kali.net

[Kang, 2001] K. Kang, D. Lee, H. Young, K. Chon. NLM: Network-based Multicast for Traffic Control of Heterogeneous Network. Computer Communications, Volume 24, 2001.

[Kawahara, 2004] Y. Kawahara, H. Morikawa. A Peer-to-Peer Message exchange Scheme for Large-Scale Networked Virtual Environments. Telecommunication Systems, 25:3, 4, 353-370, 2004.

[Keller, 2000] R. Keller, S. Choi, M. Dasen, D. Decasper, G. Fankhauser, B. Plattner. An Active Router Architecture for Multicast Video Distribution. *INFOCOM 2000*, March 2000.

[Kobbelt, 2002] L.P.Kobbelt. Multiresolution Techniques. September 2003 http://www-i8.informatik.rwth-aachen.de/publications/publications_2002.html

[Lee, 2002] D. Lee, et al. ATLAS – A Scalable Network Framework for Distributed Virtual Environments. Proceedings of ACM Collaborative Virtual Environments (CVE 2002), pp47-54, Germany, 2002.

[Lehman, 98] L.H. Lehman, S.J. Garland, D.L. Tennenhouse. Active reliable multicast. In Proceedings of IEEE INFOCOM 98, pages 581-589, 1998.

[Macedonia, 95] M. Macedonia, M. Zyda, D. Pratt, and P. Barham. Exploiting reality with multicast groups: A network architecture for large-scale virtual environments. Proceedings of the 1995 Virtual Reality Annual International Symposium (VRAIS), March 1995.

[Mahfooz, 2001] S. Mahfooz. Mechanisms for Differentiated Services in the Access networks. PhD Thesis, Liverpool John Moores University, 2001.

[Morillo, 2003] P. Morillo, et al. An Adaptive Load Balancing Technique for Distributed Virtual Environment Systems. In Proceedings of the IASTED International Conference Parallel and Distributed Computing and Systems, November 2003, USA.

[Morse, 2000] K. L. Morse, L. Bic, et al. Interest Management in Large-Scale Virtual Environments. Presence 9(1), 2000.

[Moore, 2000] Jonathan T. Moore and Scott M. Nettles. Towards Practical Programmable Packets. Technical Report MS-CIS-00-12, University of Pennsylvania, May 2000.

[Mplayer, 2004] Mplayer server. October 2004 http://www.mplayer.com

[MPEG-4, 99] MPEG-4 overview. ISO/IEC JTC1/SC29/WG11. Document No. W2725, March 1999.

[Nichols, 97] K. Nichols, V. Jacobson, and L. Zhang. A Two-bit Differentiated Services Architecture for the Internet. draft-nichols-diff-svc-arch-00.txt, INTERNET DRAFT, December, 1997.

[Oliveira, 2003] J. C. Oliveira, N.D. Georganas. VELVET: An Adaptive Hybrid Architecture for Very Large Virtual Environments. Presence, Vol.12, No.6, December, pp 555-580, 2003.

[OPNET] OPNET. Modeler 7.0. Online Documentation. http://www.opnet.com/

[Pajarola, 2000] R.Pajarola, J.Rossignac. Compressed Progressive Meshes. IEEE Transactions on Visualization and Computer Graphics, Vol.6, No1, January-March 2000.

[Rajan. 98] R. Rajan, J. Martin, and R. Chaudhury. Schema for Differentiated Services and Integrated Services in networks. Internet draft: Location ftp://ftp.ietf.org/internet-drafts/draft-rajan-policy-qosschema-00.txt, pp. 29, October 1998.

[RFC, 94] RFC1633. Integrated Services in the Internet Architecture: an Overview. R. Braden, D. Clark, S. Shenker, 1994. June 2003http://www.faqs.org/ftp/rfc/rfc1633.pdf

[RFC 2474, 98] K. Nichols, S. Blake, F. Baker, D. Black. Definition of the Differentiated Services Field (DS Field) in the IPv4 and IPv6 Headers. December 1998.

[Ribelles, 2002] J. Ribelles, A. Lopez, O. Belmonte, I. Remolar, M. Chover. Multiresolution modelling of arbitrary polygonal surfaces: a characterization. Computers & Graphics 26, pp449-462, 2002.

[Rossignac, 2000] J.Rossignac, "Dealing with Shape Complexity for Internet Access and Graphic Applications", Eurographics'2000 Proceedings, 2000. June 2002 http://www.eg.org/DL/

[Schmalstieg, 96] D.Schmalstieg, M.Gervautz. Demand-Driven Geometry Transmission for Distributed Virtual Environments. Computer Graphics Forum, Volume 15/3, 1996.

[Schmid, 2002] S. Schmid. A Component-based Active router Architecture. PhD Thesis, 2002. December 2003http://www.mobileipv6.net/~sschmid/publications/PhD-Thesis.pdf

[Schroeder, 2001] R. Schroeder et al. Collaborating in networked immersive spaces: as good as being there together? Computers & Graphics, 25(5), pp. 781-788, 2001.

[Scott, 98] D. Scott Alexander. ALIEN: A Generalized Computing Model of Active Networks. Ph.D. Thesis, University of Pennsylvania, 1998.

[Singh, 94] G. Singh, L. Serra, et al. BrickNet: A Software Toolkit For Network-Based Virtual Worlds. Presence: Teleoperators and Virtual Environments 3(1), 1994.

[Singhal, 99] S. Singhal and M. Zyda. Networked Virtual Environments – Design and Implementation. Addison-Wesley, New York, 1999.

[Smith, 95] J. Smith, K. Russo, et al. Prototype Multicast IP Implementation in ModSAF. 12th Workshop on Standards for the Interoperability of Defence Simulations, Florida, 1995.

[Son, 2003] J. Son et al. Router-assisted TCP-Friendly Traffic Control for Layered Multicast. Proceedings of ICOIN 2003, pp 121-133, Korea, February 2003.

[Stevens, 98] W. Stevens. Networking APIs: Sockets and XTI. Prentice Hall, 1998.

[Striegel, 2002] A. Striegel, G. Manimaran. A Survey of QoS Multicasting Issues. IEEE Communications Magazine, vol.40, 82-87, June, 2002.

[Tanenbaum, 2003] A. S. Tanenbaum. Computer Networks (Fourth Edition). Pearson Education, 2003.

[Taubin, 98a] G.Taubin, A. Gueziec, W.Horn, F.Lazarus. Progressive forest split compression. In Proceedings Siggraph'98 Conference, pp 123-132, July 1998.

[Taubin, 98b] G.Taubin, W.Horn, F.Lazarus, J.Rossignac. Geometry Coding and VRML. Proceedings of IEEE, 86(6):1228-1243, June 1998.

[Taubin, 99] G.Taubin. 3D Geometry Compression and Progressive Transmission. Eurographics'99, 1999.

[Taubin, 2000] G.Taubin. Geometric Signal Processing of Polygonal Meshes. Eurographics'2000 Proceedings, 2000.

[Tennenhouse, 97] D. Tennenhouse, J. Smith, W. Sincoskie, D. Whetherall, and G. Minden. A Survey of Active Network Research. IEEE Communications Magazine, Vol.35, No, Jan 1997.

[Tromp, 98] J. Tromp, A. Bullock, A. Steed, A. Sadagic, M. Slater, and E. Frecon. Small Group Behaviour Experiments in the Coven Project. IEEE Computer Graphics and Applications, Nov/Dec, 1998.

[Turk, 92] G.Turk. Re-Tiling polygonal surfaces. Computer Graphics (SIGGRAPH'92 Proceedings), 1992, p55-64.

[Van, 97] V.C. Van. A Defence against Address Spoofing using Active Networks. M. eng thesis, Massachusetts Institute of Technology, June 1997.

[Walsh, 2002] A.Walsh, D.Gehringer. Java 3D API Jump-Start. Prentice Hall PTR, 2002.

[Wang, 2001] Zheng Wang. Internet QoS: Architectures and Mechanisms for Quality of Service. Morgan Kaufmann Publishers, 2001.

[Watt, 98] A.Watt, F.Policarpo. The Computer Image. Addison-Wesley, 1998.

[Web3D, 2001a] Web3D Consortium Web site. VRTP Working Group. December 2001http://www.web3d.org/WorkingGroups/vrtp

[Web3D, 2001b] Web3D Consortium Web Site. DIS-JAVA-VRML Working Group. December 2001http://www.web3d.org/WorkingGroups/vrtp/dis-java-vrml/

[Web3D, 2004a] Web3D Consortium Web Site. X3D VRML archives. September 2004 http://www.web3d.org/x3d/vrml

[Web3D, 2004b] Web3D Consortium Web Site. X3D. September 2004 http://www.web3d.org/x3d

[Wetherall, 98] D. Wetherall, J. Guttag, D. Tennenhouse. ANTS: A Toolkit for Building and Dynamically Deploying Network Protocols. Proc. IEEE OPENARCH'98, San Francisco, CA, April 1998.

[Wittmann, 2001] R. Wittmann, M. Zitterbart. Multicast Communication. Protocols and Applications. Morgan Kaufmann Publishers, 2001.

[Wolf, 2003] T. Wolf, S. Y. Choi. Aggregated hierarchical multicast – a many-to-many communication paradigm using programmable networks. Systems, Man and Cybernetics, Part C, IEEE Transactions on, Volume: 33(3), August, 2003.

[Zabele, 2000] S. Zabele, T. Stanzione. Interest Management Using an Active Networks Approach. 00S-SIW-030, Spring 2000 Simulation Interoperability Workshop, 2000.

[Zabele, 2002] S. Zabele, M. Dorsch et al. SANDS: Specialized Active Networking for Distributed Simulation. IEEE DARPA Active networks Conference and Exposition (DANCE), USA, May 2002.

```
#VRML V1.0 ascii

Separator {
   ShapeHints {
         vertexOrdering COUNTERCLOCKWISE
         faceType      CONVEX
         creaseAngle    1
   }
   DEF bat Separator {
         Coordinate3 {
         point   [-0.0295812   0.0663223   0.0295812,
          -0.021492   -0.0684449   0.021492,
          -0.0295812   -0.0684449   0.011299,
          -0.0347748   0.0663223   0.011299,
          -0.0347748   -0.0684449   0,
          -0.0365644   0.0663223   0,
          -0.0365644   -0.0684449   -0.011299,
          -0.0347748   0.0663223   -0.011299,
          -0.0347748   -0.0684449   -0.021492,
          -0.0295812   0.0663223   -0.021492,
          -0.0295812   -0.0684449   -0.0295812,
          -0.021492   0.0663223   -0.0295812,
          -0.021492   -0.0684449   0.0295812,
          -0.021492   0.0663223   -0.0133749,
          0.00217289   0.127384   -0.0312293,
          -0.00630595   0.132596   -0.0329258,
          0.00521495   0.111747   -0.0347748,
          0.0056495   -0.0185615   -0.0301345,
          0.0056495   -0.247904   -0.012016,
          0.00202096   -0.50576   -0.01534,
          0.00593216   -0.654157   -0.0253138,
          0   -0.654157   -0.0267499,
          0.00869155   -0.659006   -0.0246943,
          0.0039112   -0.664431   0,
          0.0210787   -0.013349   -0.0192025,
          0.00913505   -0.357363   -0.0102215,
          0.0116969   -0.65537   -0.0210482,
          0.0148791   -0.664431   -0.0196435,
          0.0265404   0.122171   -0.0159822,
          0.0290124   -0.013349   -0.0144592,
          0.0144196   -0.654157   -0.0121986,
          0.0210482   -0.660506   -0.0128717,
          0.0347748   -0.138445   -0.00390424,
          0.0125999   -0.357363   0.000979515,
          0.0267498   -0.659006   -0.00782242,
          0.0120375   -0.664431   -0.0163946,
          0.0312981   0.122171   0,
          0.0127599   -0.654157   0,
          0.0267202   -0.656582   0,
          0.0253138   -0.664431   0.0108644,
```

0.0350008 -0.013349 0.00662555,
0.0347748 -0.138445 0.00195211,
0.0254124 -0.656582 0.00825694,
0.0240749 -0.664431 0.0126119,
0.0290124 -0.013349 0.0144591,
0.0105819 -0.50576 0.0148791,
0.0210482 -0.657938 0.0220746,
0.0210787 -0.013349 0.0199013,
0.00768819 -0.50576 0.0204793,
0.011568 -0.65537 0.0279401,
0.0157057 -0.661718 0.0247523,
0.0186821 0.132596 0.0263826,
0.0104299 0.122172 0.0233954,
0.00665998 -0.0680785 0.0309884,
0 -0.0185615 0.0244448,
0.00202096 -0.50576 0.0126344,
0 -0.654157 0.0257073,
0.0019556 -0.663074 0.0263759,
-0.0110817 -0.013349 0.0233954,
-0.00404192 -0.50576 0.0240749,
-0.0102172 -0.656582 0.0244136,
-0.0210787 -0.013349 0.0199013,
-0.00768819 -0.50576 0.0204793,
-0.0148791 -0.654157 0.0204793,
-0.0113507 -0.664431 0.0240435,
-0.0119946 0.132596 0.0270514,
-0.0303108 0.0465927 0.00936265,
-0.021617 -0.656582 0.013481,
-0.0336289 0.116959 0.00819775,
-0.0226783 -0.322103 0.00939166,
-0.0124398 -0.50576 0.00782236,
-0.0240749 -0.654157 0.0117853,
-0.0236146 -0.661718 0.00782236,
-0.0240749 -0.664431 0.00434576,
-0.0191968 -0.579958 0,
-0.0267202 -0.661718 -0.0104299,
-0.0320998 0.122172 -0.0056495,
-0.0356696 -0.0185615 -0.011299,
-0.0236073 -0.247904 -0.00586333,
-0.0254124 -0.661718 -0.0102172,
-0.0260952 0.132596 -0.0103327,
-0.0241412 0.116959 -0.0156009,
-0.0115109 -0.50576 -0.00742632,
-0.0173284 -0.654157 -0.0121986,
-0.021617 -0.656582 -0.0273058,
-0.0255366 -0.0185615 -0.0295812,
-0.0145901 -0.247904 -0.0106701,
-0.00586506 -0.654157 -0.0204793,
-0.0166925 -0.657938 -0.0196481,
-0.0148791 -0.664431 -0.0247523,

```
 -0.0104299  0.122172  -0.0347748,
 -0.011299  -0.0185615  -0.012016,
 -0.00404192  -0.357363  -0.0240749,
 -0.0078224  -0.654157  -0.0274382,
 -0.00434578  -0.659006  -0.0240749]
}
DEF batcolor Material {
   ambientColor   0.788235 0.662745 0.529412
   diffuseColor   0.788235 0.662745 0.529412
   specularColor  0.788235 0.662745 0.529412
   transparency   0
}
IndexedFaceSet {
   coordIndex   [ 20, 21, 22, -1,
                  28, 16, 24, -1,
                  24, 19, 25, -1,
                  25, 20, 26, -1,
                  26, 23, 27, -1,
                  27, 23, 35, -1,
                  28, 24, 29, -1,
                  25, 26, 30, -1,
                  30, 26, 31, -1,
                  31, 27, 35, -1,
                  36, 14, 28, -1,
                  28, 29, 32, -1,
                  32, 25, 33, -1,
                  30, 31, 34, -1,
                  34, 31, 35, -1,
                  36, 32, 40, -1,
                  33, 30, 37, -1,
                  37, 30, 38, -1,
                  38, 35, 39, -1,
                  51, 14, 36, -1,
                  51, 36, 40, -1,
                  41, 40, 33, -1,
                  37, 38, 42, -1,
                  42, 38, 43, -1,
                  43, 39, 35, -1,
                  51, 40, 44, -1,
                  44, 41, 45, -1,
                  45, 42, 46, -1,
                  46, 43, 35, -1,
                  51, 44, 47, -1,
                  47, 44, 48, -1,
                  48, 46, 49, -1,
                  49, 46, 50, -1,
                  50, 46, 35, -1,
                  52, 47, 53, -1,
                  53, 48, 55, -1,
                  49, 50, 57, -1,
```

57, 50, 35, -1,
65, 14, 51, -1,
65, 51, 53, -1,
54, 53, 55, -1,
56, 49, 57, -1,
65, 53, 58, -1,
58, 55, 59, -1,
59, 56, 60, -1,
60, 57, 64, -1,
64, 57, 35, -1,
65, 58, 61, -1,
61, 59, 62, -1,
62, 59, 63, -1,
65, 61, 66, -1,
66, 62, 69, -1,
69, 62, 67, -1,
67, 60, 72, -1,
72, 64, 35, -1,
80, 14, 65, -1,
80, 65, 68, -1,
70, 69, 71, -1,
71, 67, 72, -1,
73, 72, 35, -1,
68, 69, 77, -1,
77, 70, 74, -1,
74, 71, 75, -1,
75, 73, 35, -1,
76, 68, 77, -1,
78, 74, 82, -1,
82, 74, 83, -1,
83, 75, 79, -1,
79, 75, 35, -1,
80, 76, 81, -1,
81, 78, 85, -1,
85, 78, 82, -1,
84, 79, 88, -1,
88, 79, 35, -1,
15, 14, 80, -1,
86, 83, 87, -1,
87, 84, 88, -1,
89, 88, 35, -1,
90, 85, 91, -1,
91, 86, 92, -1,
87, 88, 93, -1,
93, 88, 94, -1,
94, 88, 0, -1,
0, 89, 35, -1,
14, 91, 16, -1,
16, 91, 17, -1,
17, 91, 18, -1,

18, 87, 19, -1,
19, 93, 21, -1,
23, 0, 35, -1,
20, 19, 21, -1,
22, 21, 94, -1,
23, 22, 94, -1,
16, 28, 14, -1,
17, 24, 16, -1,
18, 24, 17, -1,
19, 24, 18, -1,
20, 25, 19, -1,
26, 20, 22, -1,
23, 26, 22, -1,
25, 29, 24, -1,
31, 26, 27, -1,
25, 32, 29, -1,
30, 33, 25, -1,
32, 36, 28, -1,
33, 40, 32, -1,
38, 30, 34, -1,
35, 38, 34, -1,
43, 38, 39, -1,
41, 44, 40, -1,
45, 41, 33, -1,
37, 45, 33, -1,
42, 45, 37, -1,
46, 42, 43, -1,
48, 44, 45, -1,
46, 48, 45, -1,
52, 51, 47, -1,
53, 47, 48, -1,
49, 55, 48, -1,
53, 51, 52, -1,
49, 56, 55, -1,
54, 58, 53, -1,
55, 58, 54, -1,
56, 59, 55, -1,
57, 60, 56, -1,
59, 61, 58, -1,
63, 59, 60, -1,
62, 66, 61, -1,
67, 62, 63, -1,
60, 67, 63, -1,
72, 60, 64, -1,
68, 65, 66, -1,
71, 69, 67, -1,
69, 68, 66, -1,
70, 77, 69, -1,
74, 70, 71, -1,
75, 71, 72, -1,

```
73, 75, 72, -1,
76, 80, 68, -1,
78, 77, 74, -1,
75, 83, 74, -1,
81, 76, 77, -1,
78, 81, 77, -1,
84, 83, 79, -1,
81, 15, 80, -1,
86, 85, 82, -1,
83, 86, 82, -1,
84, 87, 83, -1,
90, 15, 81, -1,
85, 90, 81, -1,
86, 91, 85, -1,
87, 92, 86, -1,
0, 88, 89, -1,
14, 15, 90, -1,
91, 14, 90, -1,
18, 91, 92, -1,
87, 18, 92, -1,
93, 19, 87, -1,
94, 21, 93, -1,
23, 94, 0, -1]
        }
    }
}
```

Figure I-1 Crude model (LOD=0, 96 vertices) of the *baseball bat* object progressive representation

Batch 1 content.

0	0
0	1
0	2
0	3
0	4
0	5
0	6
0	7
0	8
0	9
0	10
0	11
0	12
0	13
1	14
1	15
0	17
1	18
0	20

1	21
0	23
0	24
0	25
0	26
1	28
0	30
1	31
0	33
1	34
1	35
1	37
1	38
0	41
1	42
0	44
1	45
1	46
0	49
0	50
0	51
1	52
0	53
1	55
0	57
1	59
0	61
1	62
1	64
0	66
1	67
0	68
1	69
0	70
1	73
1	74
0	76
1	77
1	80
1	82
0	84
1	85
1	88
0	90
0	91
1	92
1	93
1	94
1	96
1	98

1 100
0 101
0 102
0 103
0 104
1 107
0 109
0 110
0 111
1 112
1 115
1 117
1 118
0 119
1 120
0 121
0 123
1 124
0 126
1 127
0 129
0 131
0 132
0 133
0 134
0 135
0 136
-0.0203769 -0.00434578 -0.010424 14 17 14 15 16
0.0033742 -0.0126119 0 15 14 15 14 130
-0.0017896 -0.011299 -0.239767 18 20 18 17 19
-0.0006184 -0.00404192 -0.296794 21 23 21 20 22
-0.0011377 0.000826601 -0.250192 28 20 35 28 29
-0.0113955 0.00801751 -0.00242501 31 23 38 31 32
0.0087791 0.0137938 -0.020849 46 34 34 35 40
-0.000826601 0.0011377 -0.250192 35 28 40 35 36
0.00156299 0.0076754 0 42 37 37 38 43
0.000826599 -0.0011377 -0.00784898 38 31 45 38 39
0.00585635 0.000320099 0 48 42 37 42 54
0.00782242 -0.0240749 0 45 38 51 45 27
0.00434578 0.0077198 -0.020849 46 34 52 46 47
-0.0108644 0.0031271 -0.250192 52 53 52 46 48
0.000869161 0.0026749 -0.00484902 55 50 62 55 56
0.000826601 0.0011377 -0.250192 59 53 64 59 60
0.000826599 0.0011377 -0.00756103 62 55 68 62 63
0.0011377 0.000826601 -0.250192 64 59 73 64 65
0.0049331 -0.00662212 -0.00242501 76 67 67 68 72
-0.0121404 0.0121404 0 69 14 59 69 58
-0.0108678 0.00202096 -0.359651 73 64 74 73 71
0 0 -0.239767 74 73 82 74 75
0.0126794 0 0 77 67 85 77 78

0.000786901 0.0039112 0.00271297 80 45 87 80 79
0.0013375 -0.0004346 -0.250192 82 74 88 82 83
0.000677399 -0.0047895 -0.00484902 85 80 91 85 86
0.0011377 -0.000826601 -0.250192 88 82 94 88 89
0.0035956 0.0070567 0 92 45 92 85 87
0.0077946 0.0152977 0 93 14 82 93 81
-0.0042699 -0.0037345 -0.130308 94 88 98 94 99
0.0016532 -0.0022755 -0.00484902 96 85 103 96 97
-0.0104298 -0.0030582 -0.010425 105 98 98 94 106
-0.00387271 0.0241929 -0.367315 100 101 100 94 95
0 -0.0122339 -0.148397 107 101 119 107 108
0.00739476 0.022335 -0.218918 112 111 119 112 113
0.00086916 0.0026749 -0.00542498 115 109 121 115 116
0.0169577 -0.0026859 0 117 14 117 110 105
-0.0066404 0.0086047 0.010425 130 118 118 123 122
-0.0003961 -0.013493 0 120 121 120 119 114
0.0193598 0.0138038 -0.218918 124 123 133 124 125
0.0016624 -0.0036267 -0.00756103 127 121 134 127 128

Number of split-vertices in the batch: 41

Figure I-2 Refinement batch 1 (LOD=1) of the *baseball bat* object progressive representation

Batch 2 content.

0 0
0 1
0 2
0 3
0 4
0 5
0 6
0 7
0 8
0 9
0 10
0 11
0 12
0 13
1 14
0 16
0 17
1 18
0 20
0 21
1 22
0 24
0 25
1 26

0	28
0	29
1	30
0	32
1	34
0	36
1	37
0	38
1	39
0	41
1	42
1	43
0	45
0	47
1	48
0	50
1	52
0	54
0	55
1	56
0	58
0	59
1	60
1	61
0	63
1	65
1	66
0	68
1	70
0	72
1	73
0	75
0	76
0	77
1	78
1	79
0	81
1	82
0	84
1	85
1	88
0	90
1	91
0	93
1	94
1	96
0	97
1	99
1	102
1	106

0	107
0	108
1	109
0	110
0	111
1	112
0	113
1	114
1	115
0	117
1	118
0	120
1	121
0	122
1	124
0	126
1	127
0	129
0	131
1	132
1	133
1	136
0	138
0	139
0	142
1	143
0	144
1	145
0	147
1	148
0	149
1	150
1	151
1	154
0	156
1	157
0	160
1	161
0	162
0	163
0	165
0	166
0	167
1	168
1	169
1	172
0	173
1	174
0	178
1	179

```
0       180
0       181
1       182
0       183
1       184
0       185
1       186
0       187
1       188
0       190
0       192
1       193
0       194
0.0267499  -0.00869155  0     14  16  33  14     15
-0.0044646  -0.0104299  -0.02085     18  20  18  17     19
0.0221404  -0.011299  -0.218918     22  24  22  21     23
-0.0120589  0.00378048  0     26  25  39  26     27
0.0012389  0.0078224  0     29  30  30  32     31
-0.0022754  0.0016532  -0.02085     34  20  43  34     35
0.00279509  0.00289371  0     45  37  38  37     46
-0.0022755  0.0016532  -0.00484902     39  26  48  39     40
-0.0062225  -0.0062225  0     42  14  34  42     33
-0.0016532  0.0022755  -0.02085     43  34  52  43     44
-0.0016532  0.0022755  -0.00484902     48  39  57  48     49
-0.0008691  0.002675  -0.02085     52  43  61  52     53
-0.00391817  0.0116351  0     56  47  66  56     57
-0.00869156  -0.0013767  0     60  14  52  60     51
0  0.0028127  -0.02085     61  52  70  61     62
0.00390422  -0.0006401  0     73  65  66  65     74
0  0.0028127  -0.00484902     66  56  75  66     67
0.000869201  0.002675  -0.02085     70  61  79  70     71
-0.00390422  0.0006401  0     73  63  65  73     64
-0.00784077  0.003995  0     78  14  70  78     69
0.0016532  0.0022755  -0.02085     79  70  88  79     80
0  0  -0.296794     82  65  91  82     83
-0.0016532  -0.0022755  -0.00542498     85  76  94  85     86
0.0022755  0.0016532  -0.02085     88  79  97  88     89
0  0  -0.296794     91  82  100  91     92
-0.0022755  -0.0016532  -0.00542498     94  85  103  94     95
-0.003995  0.00784075  0     96  14  96  97     87
-0.0227588  -0.00725708  -0.218918     99  90  109  99     100
0.0026749  0.00086915  -0.00484902     102  93  112  102     103
0.0010231  0.011299  -0.02085     106  107  106  97     98
-0.0006184  0.00404192  -0.296794     109  110  109  99     101
-0.0040515  0.0078224  -0.00542498     112  113  112  102     104
0.0013766  0.00869155  0     114  14  106  114     105
0.002675  -0.0008691  -0.02085     115  107  124  115     116
0  0  -0.296794     118  110  127  118     119
-0.003995  -0.00784075  0     129  121  121  131     130
0.0022755  -0.0016532  -0.02085     124  115  133  124     125
```

```
0  0  -0.296794     127  118  136  127     128
0.0062225  0.0062225  0     132  14  124  132     123
0.0016532  -0.0022755  -0.02085     133  124  143  133     134
0  0  -0.296794     136  127  145  136     137
0.010193  0.0051936  -0.239767     143  144  143  133     135
0  0  -0.296794     145  136  154  145     146
0.00618758  0.0062705  -0.00542498     148  149  148  139     140
0.00869153  0.0013767  0     150  14  142  150     141
0  -0.0028127  -0.02085     151  142  161  151     152
0  0  -0.296794     154  145  163  154     155
0  0.0028127  -0.00542498     157  149  167  157     158
0.011299  -0.0017896  -0.239767     161  162  161  151     153
0.00784074  -0.003995  0     168  14  160  168     159
-0.0016532  -0.0022755  -0.02085     169  160  179  169     170
0.00352206  -0.0018579  -0.296794     172  173  172  163     164
-0.0016532  -0.0022755  -0.00484902     174  165  184  174     175
0.0080892  -0.0080892  -0.239767     179  180  179  169     171
-0.0017945  0.00364627  0     190  182  183  182     191
0.0078757  -0.003947  -0.00542498     184  185  184  174     176
0.0039951  -0.00784075  0     186  14  186  187     177
0  0  -0.239767     188  180  18  188     189
-0.0013766  0.00869155  0     28  193  193  30     195
```

Number of split-vertices in the batch: 59

Figure I-3 Refinement batch 2 (LOD=2) of the *baseball bat* object progressive representation

Movements of user C grid 3x3 *random* scenario
S R LOD Time(s)
3,1,0,61
3,2,0,62
3,4,2,63
3,1,2,317
3,2,1,318
3,4,1,319
3,1,0,393
3,2,0,394
3,4,1,395
3,1,1,425
3,2,0,426
3,4,1,427
3,1,0,564
3,2,0,565
3,4,2,566
3,1,-1,687
3,2,-1,688
3,4,-1,689
3,1,0,727
3,2,1,728
3,4,2,729
a)

Movements of user C grid 9x9 *random* scenario
S R LOD Time(s)
3,1,4,31
3,2,4,32
3,4,7,33
3,1,3,61
3,2,4,62
3,4,7,63
3,1,3,130
3,2,4,131
3,4,8,132
3,1,6,317
3,2,4,318
3,4,5,319
3,1,4,393
3,2,4,394
3,4,6,395
3,1,4,425
3,2,3,426
3,4,7,427
3,1,2,564

3,2,2,565
3,4,7,566
3,1,-1,687
3,2,-1,688
3,4,-1,689
3,1,1,727
b)

Figure II-1 Fragments of the movement scripts. User C, *random* scenario, 4 users group (a) 3 LOD, b) 9 LOD)

Movements of user D grid 9x9 *far* scenario
S R LOD Time(s)
4,1,0,3
4,2,1,4
4,3,5,5
4,1,2,317
4,2,0,318
4,3,2,319
4,1,0,499
4,2,1,500
4,3,3,501
4,1,0,560
4,2,0,561
4,3,1,562
4,1,2,697
4,2,0,698
4,3,2,699
4,1,2,811
4,2,0,812
4,3,0,813
4,1,2,983
4,2,1,984
4,3,1,985
4,1,1,1025
4,2,0,1026
4,3,1,1027

Figure II-2 Fragment of the movement script (User D, *far* scenario, 4 users group, 9LOD)

Movements of user A grid 9x9 *centre* scenario
S R LOD Time(s)
1,2,6,4
1,3,7,5
1,4,6,6

1,2,6,77
1,3,7,78
1,4,7,79
1,2,6,129
1,3,6,130
1,4,7,131
1,2,7,214
1,3,7,215
1,4,7,216
1,2,6,317
1,3,6,318
1,4,7,319
1,2,8,379
1,3,7,380
1,4,7,381
1,2,7,449
1,3,7,450
1,4,7,451
1,2,6,502
1,3,6,503
1,4,7,504
1,2,7,576
1,3,5,577
1,4,8,578

Figure II-3 Fragment of the movement script (User A, *centre* scenario, 4 users group, 9LOD)

Movements of user B grid 9x9 *join* scenario
S R LOD Time(s)
2,1,4,79
2,3,4,80
2,4,1,81
2,1,4,136
2,3,5,137
2,4,1,138
2,1,2,190
2,3,4,191
2,4,0,192
2,1,-1,258
2,3,-1,259
2,4,-1,260
2,1,1,445
2,1,0,504
2,3,2,633
2,4,0,634

2,3,-1,747
2,4,-1,748
2,3,4,792
2,4,1,793
2,1,-1,843
2,3,-1,844
2,4,-1,845
2,1,0,926
2,3,5,927
2,4,0,928

Figure II-4 Fragment of the movement script (User B, *join* scenario, 4 users group, 9LOD)

Movements of user C grid 9x9 *centre* scenario 6 users
S R LOD Time(s)
3,1,7,7
3,2,6,8
3,4,7,9
3,5,6,10
3,6,8,11
3,1,8,145
3,2,5,146
3,4,7,147
3,5,6,148
3,6,8,149
3,1,8,271
3,2,7,272
3,4,7,273
3,5,6,274
3,6,7,275
3,1,7,322
3,2,7,323
3,4,6,324
3,5,6,325
3,6,7,326

Figure II-5 Fragment of the movement script (User C, *centre* scenario, 6 users group, 9LOD)

Movements of user D grid 9x9 *join* scenario 8 users
S R LOD Time(s)
4,1,1,3
4,2,1,4
4,3,4,5
4,5,1,6

4,6,6,7
4,7,3,8
4,8,2,9
4,3,-1,149
4,5,-1,150
4,6,-1,151
4,7,-1,152
4,8,-1,153
4,3,1,211
4,5,1,212
4,6,2,213
4,7,1,214
4,8,4,215
4,2,-1,548
4,3,-1,549
4,5,-1,550
4,7,-1,551
4,2,0,683
4,3,1,684
4,5,1,685
4,7,0,686

Figure II-6 Fragment of the movement script (User D, *join* scenario, 8 users group, 9LOD)

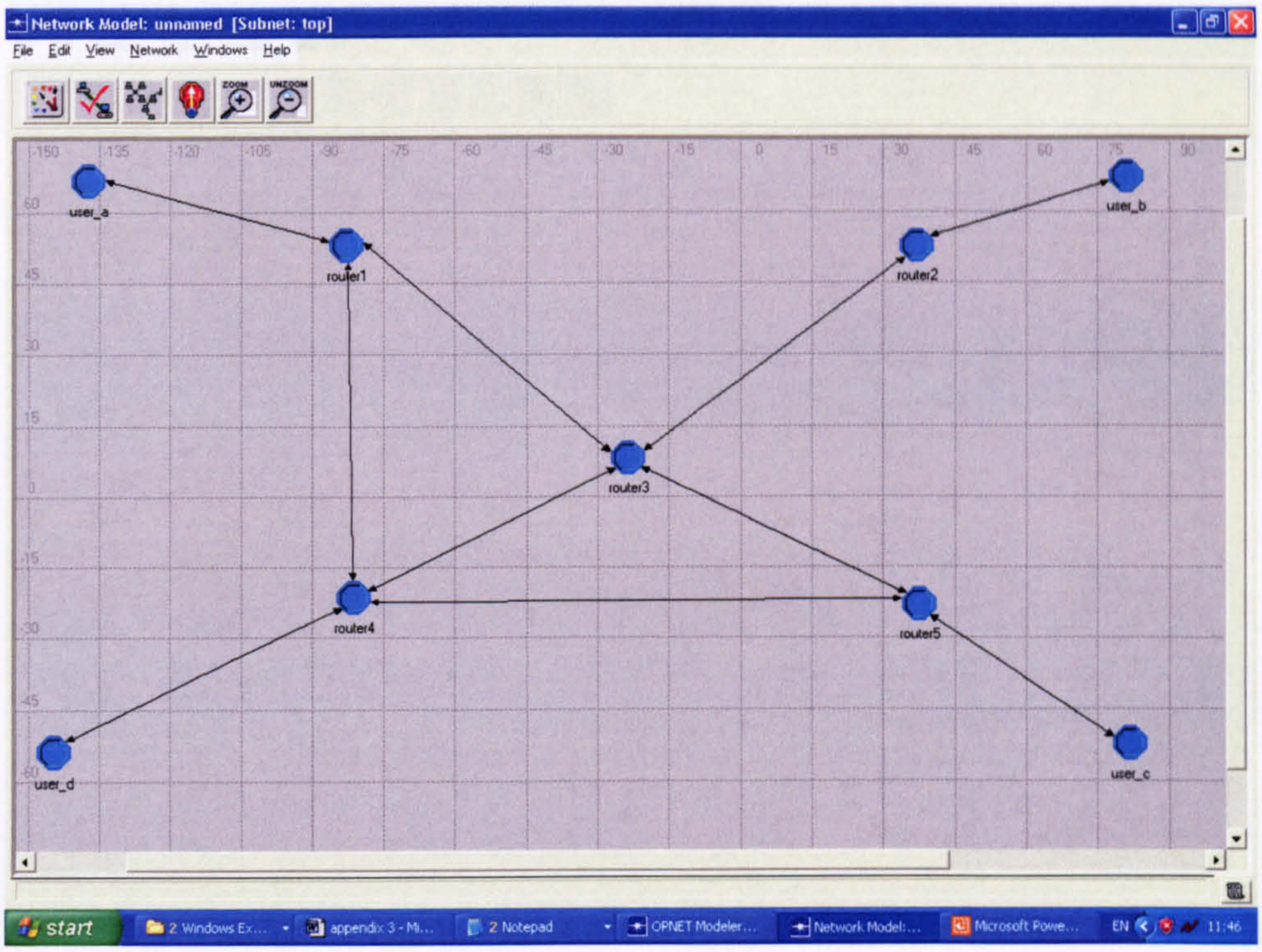


Figure III-1 Network model (non-active, 4 users)

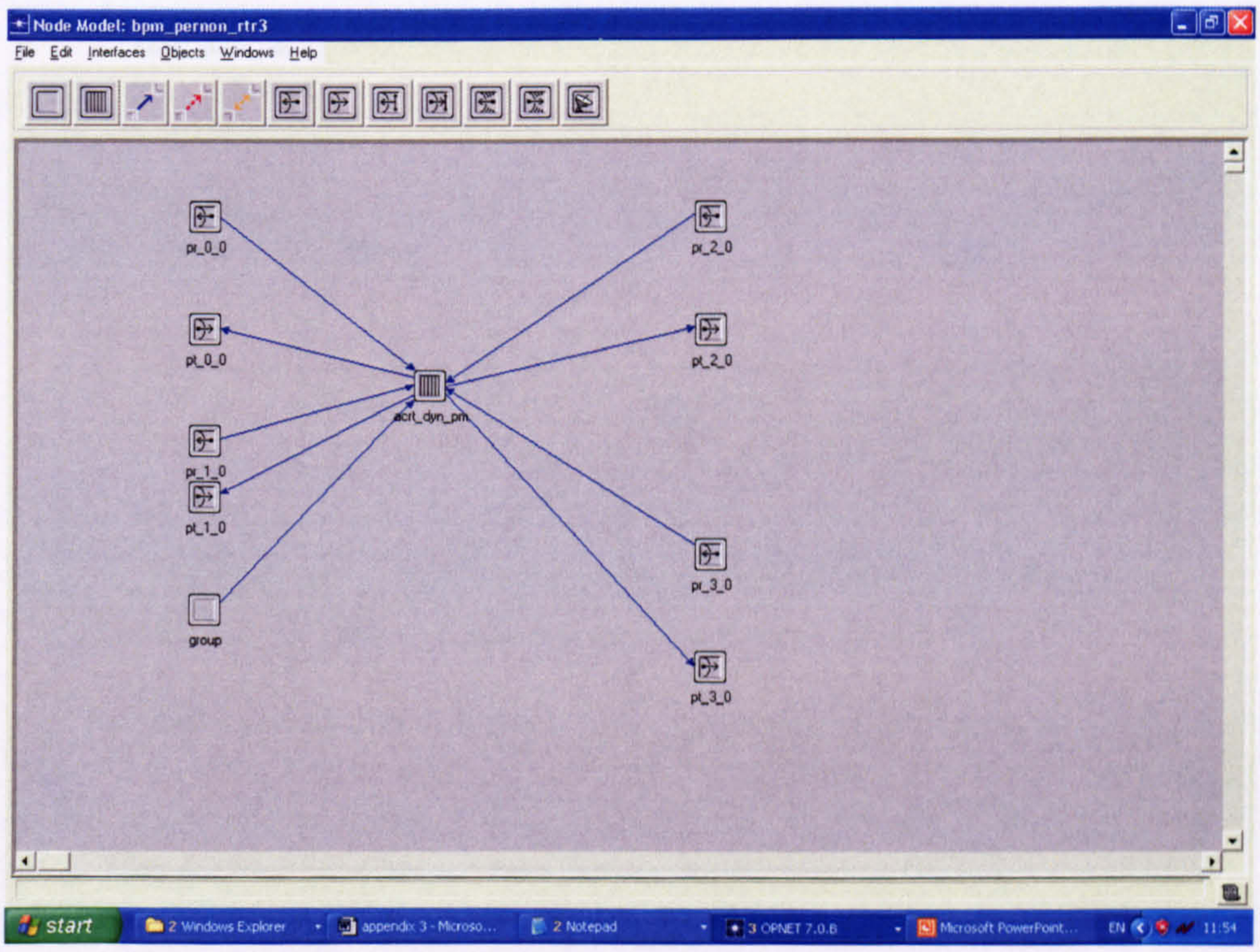


Figure III-2 Router model (non-active)

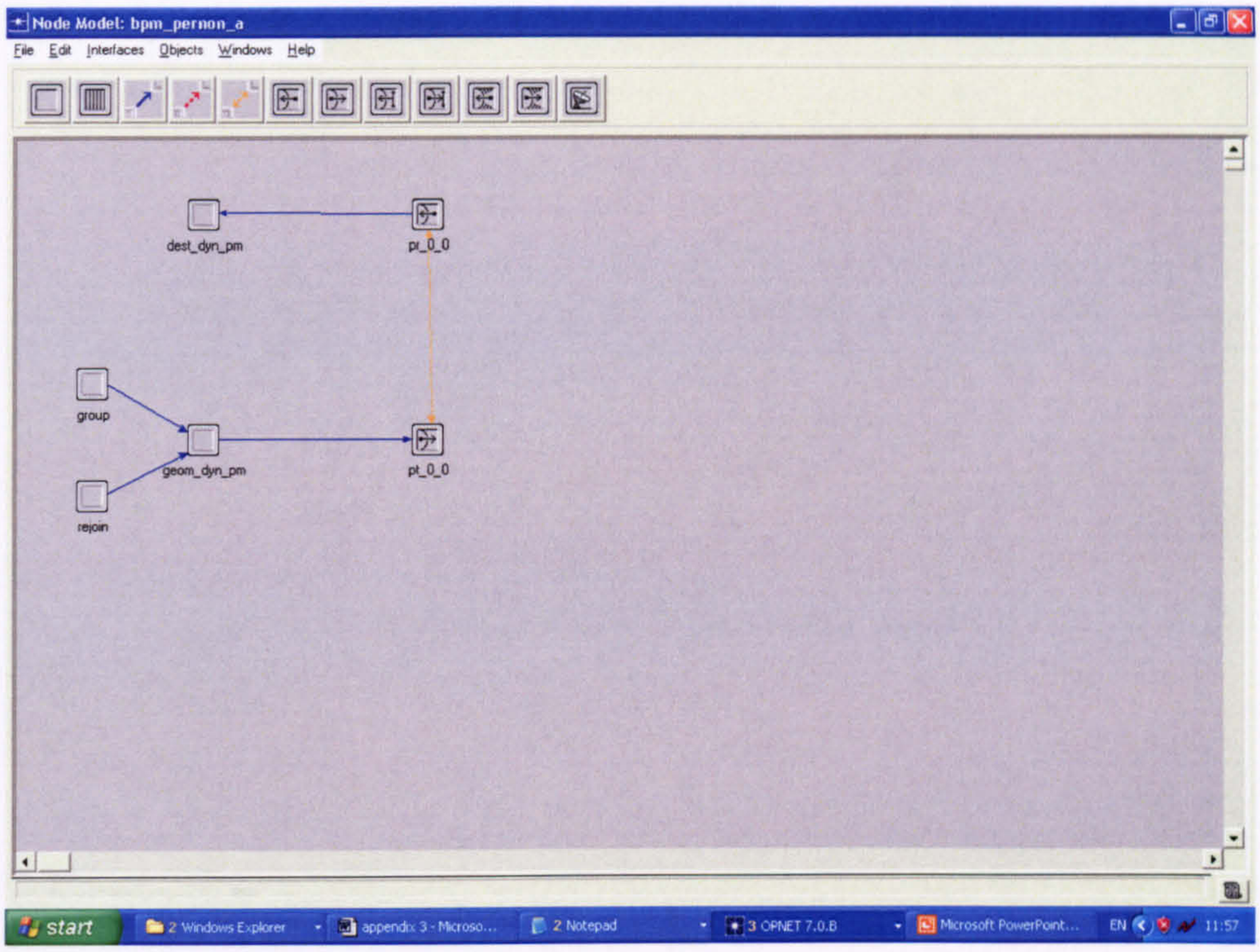


Figure III-3 Host model (non-active)

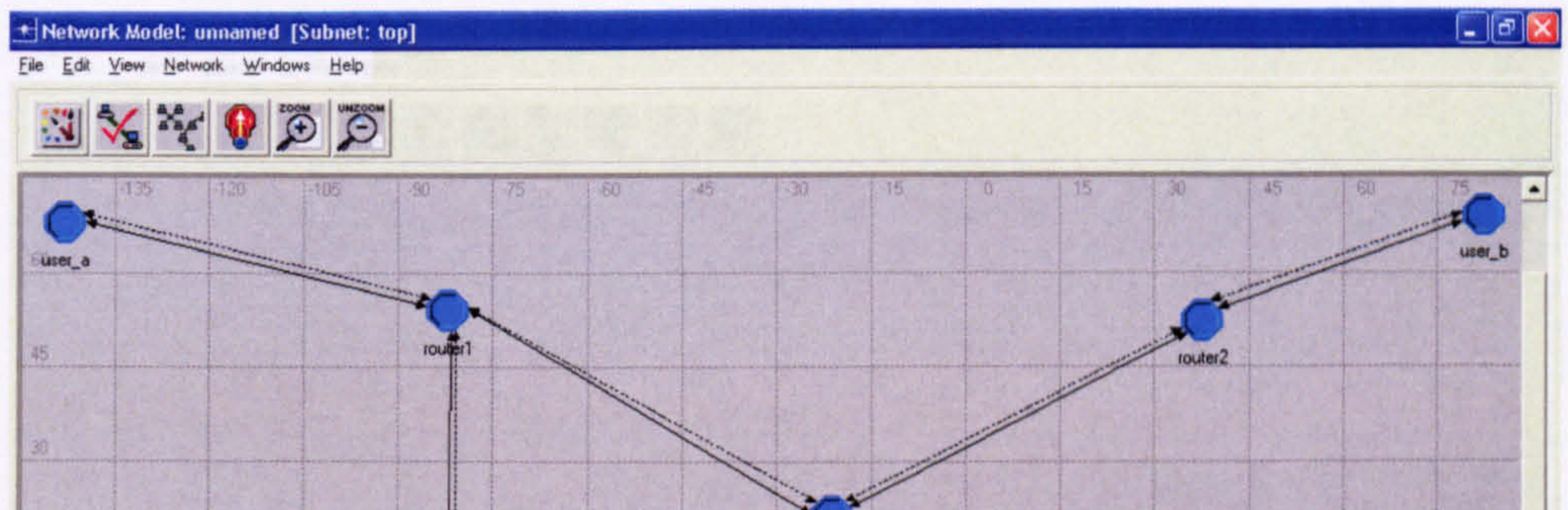


Figure III-4 Network model (active)

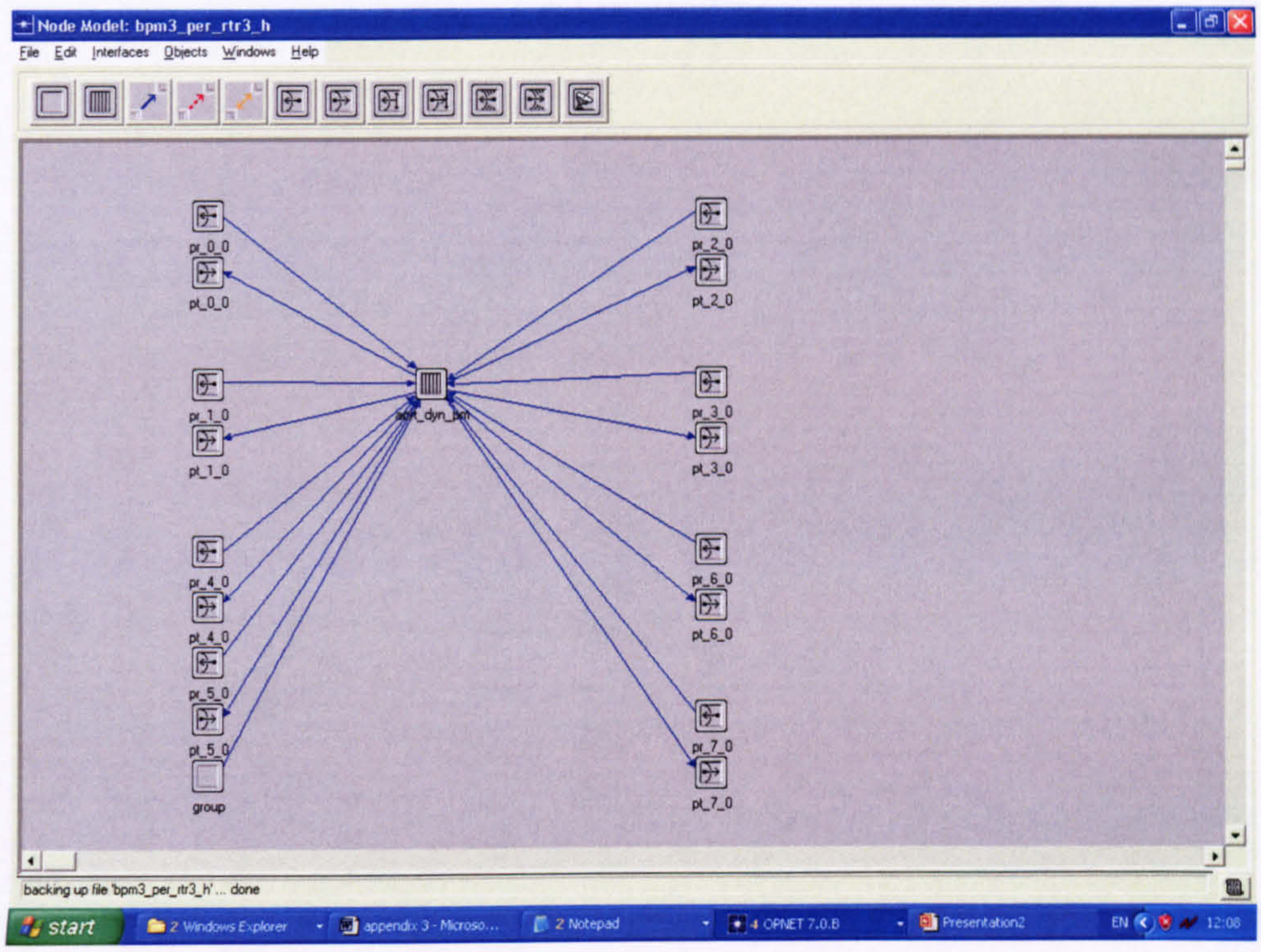


Figure III-5 Router model (active)

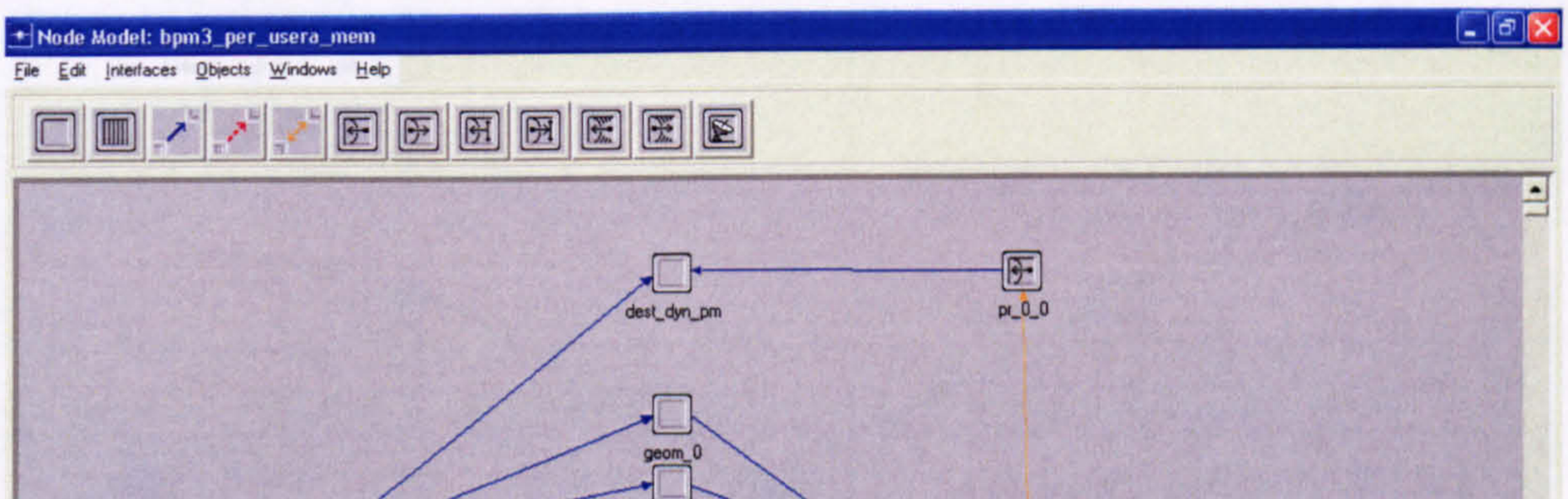


Figure III-6 Host model (3 LOD active)

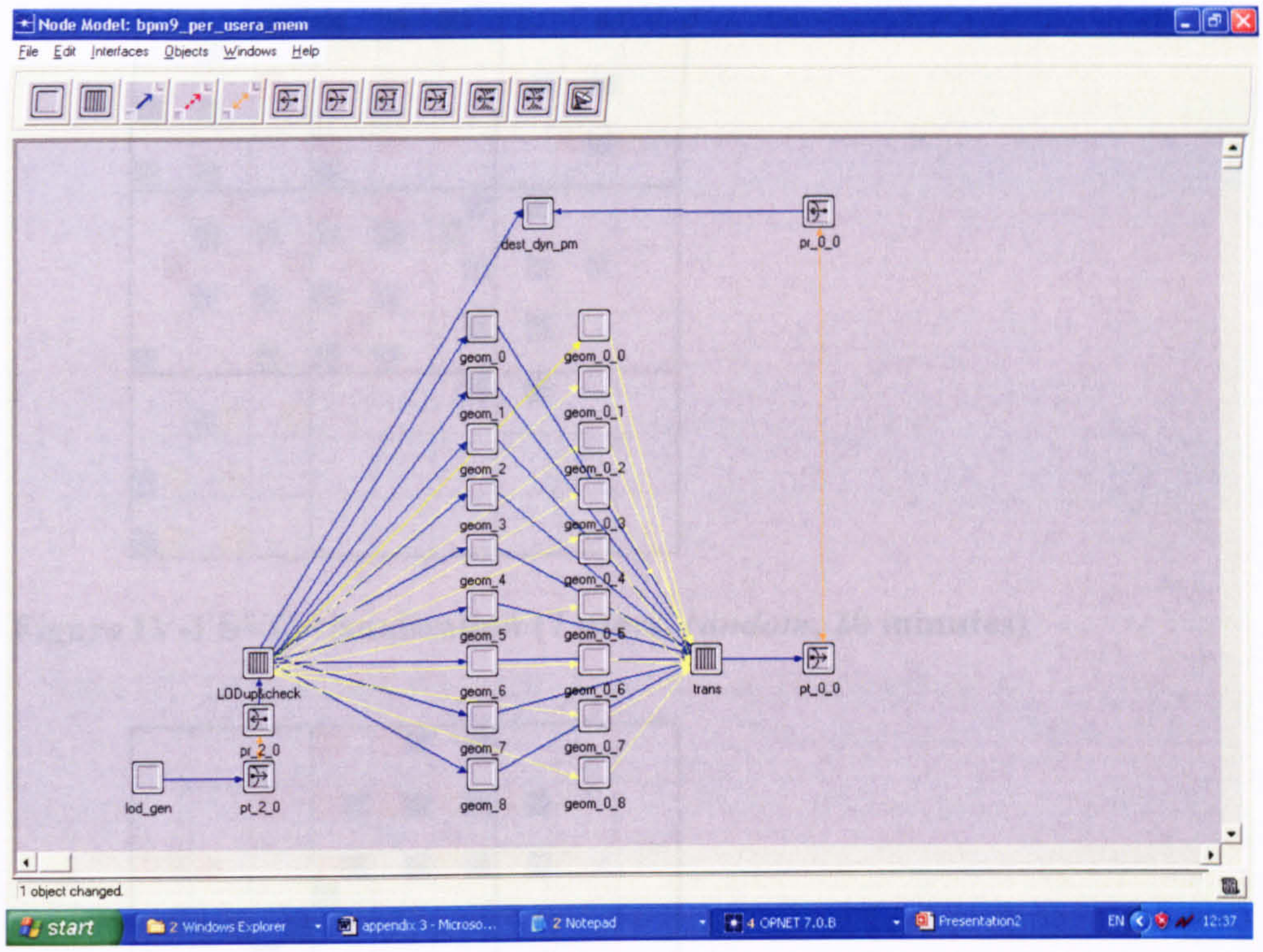


Figure III-7 Host model (9 LOD active)

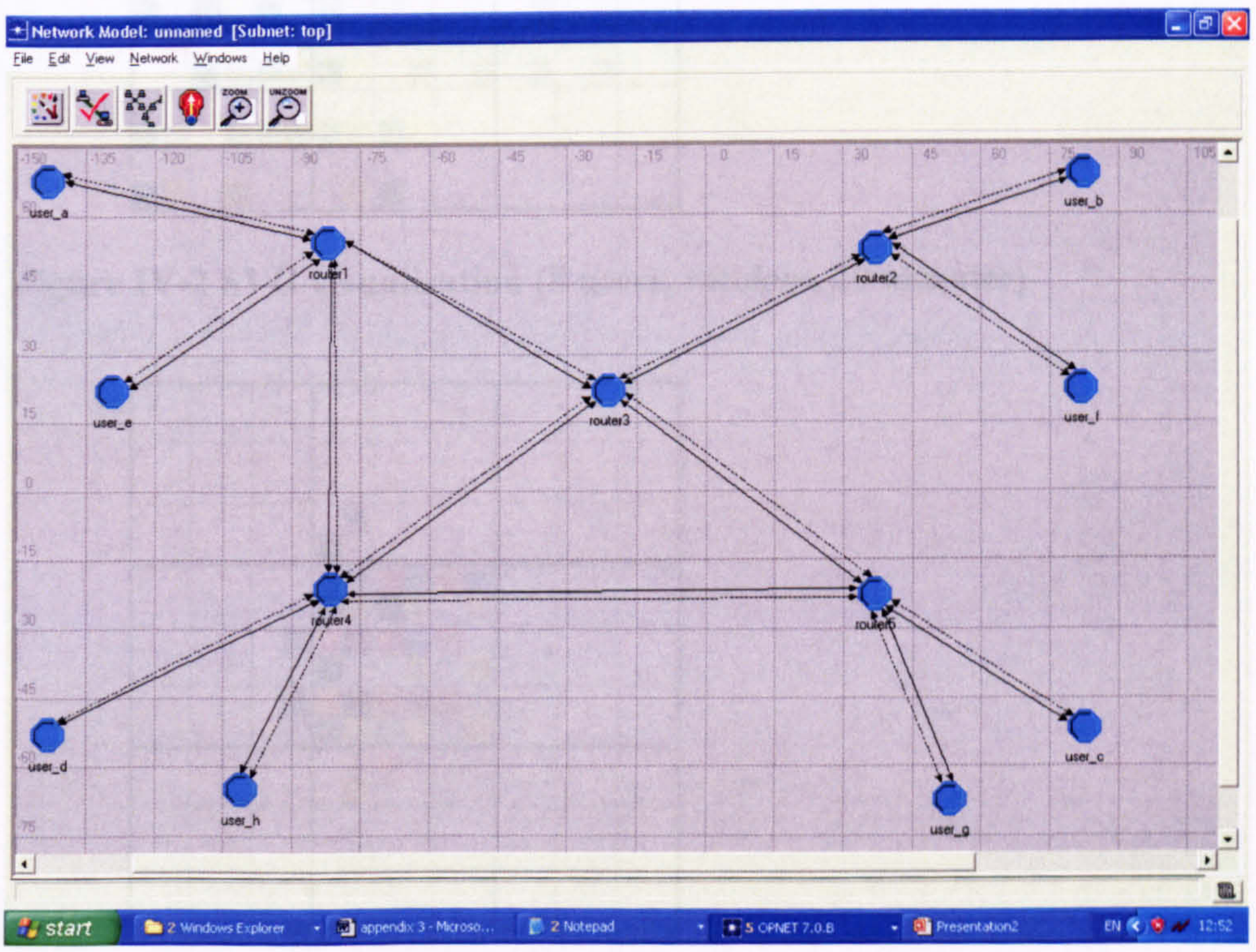


Figure III-8 Network model (active, 8 users)

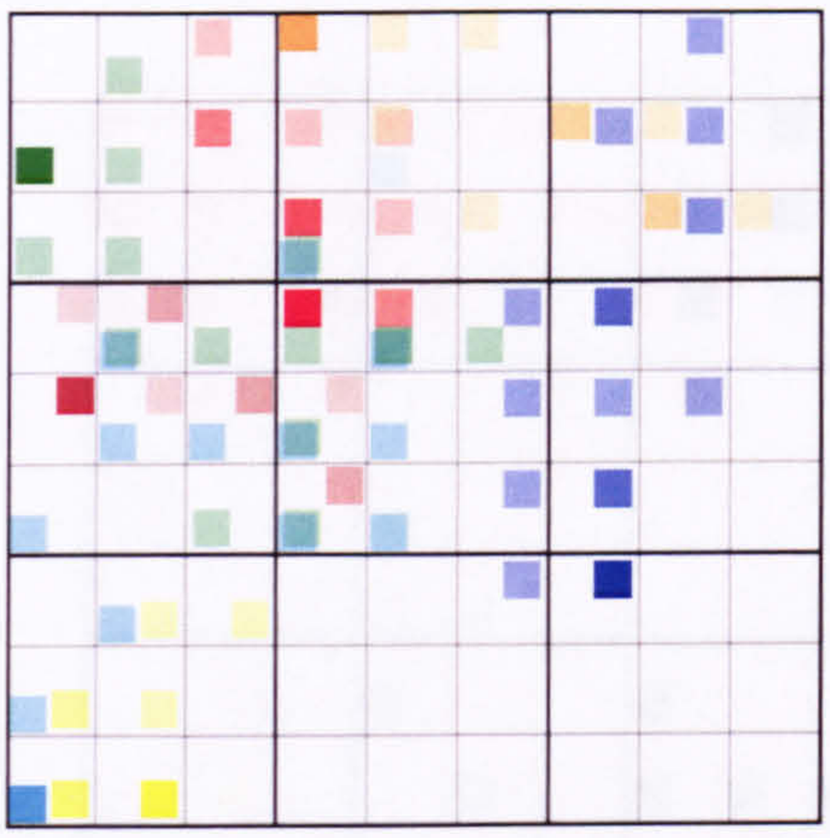


Figure IV-1 SVG visualization (7 users, *random*, 20 minutes)

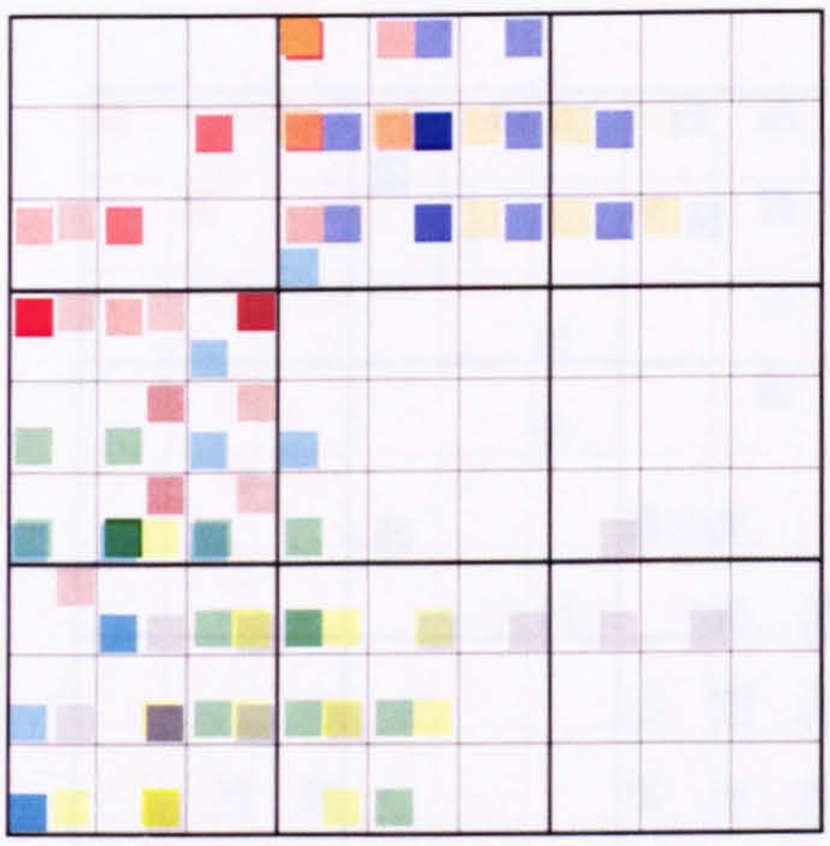


Figure IV-2 SVG visualization (8 users, *random*, 20 minutes)

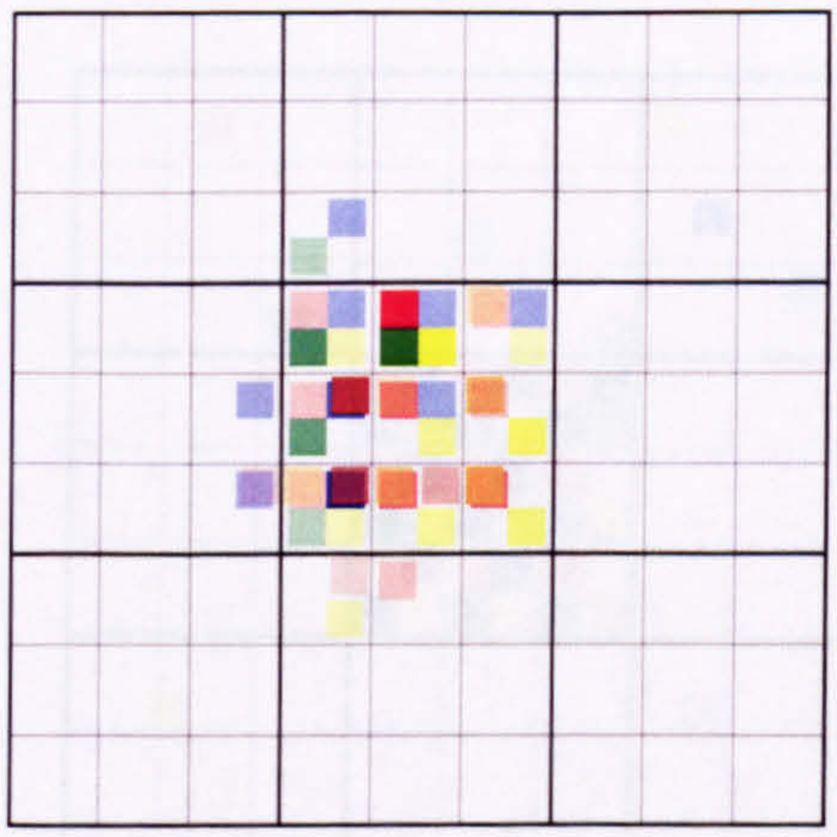


Figure IV-3 SVG visualization (6 users, *centre*, 20 minutes)

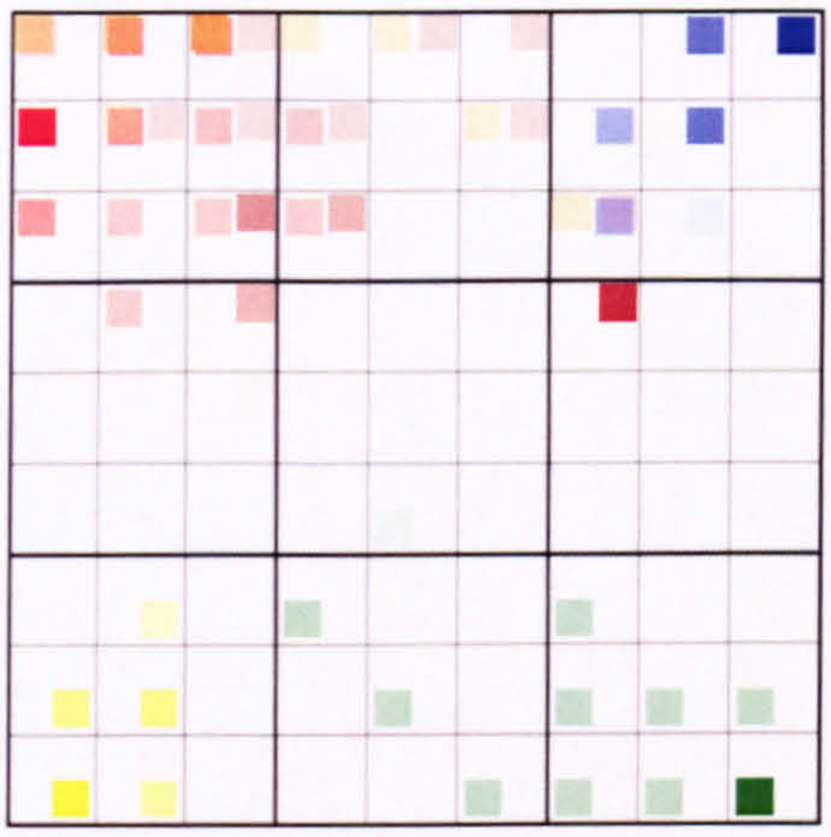


Figure IV-4 SVG visualization (6 users, *far*, 20 minutes)

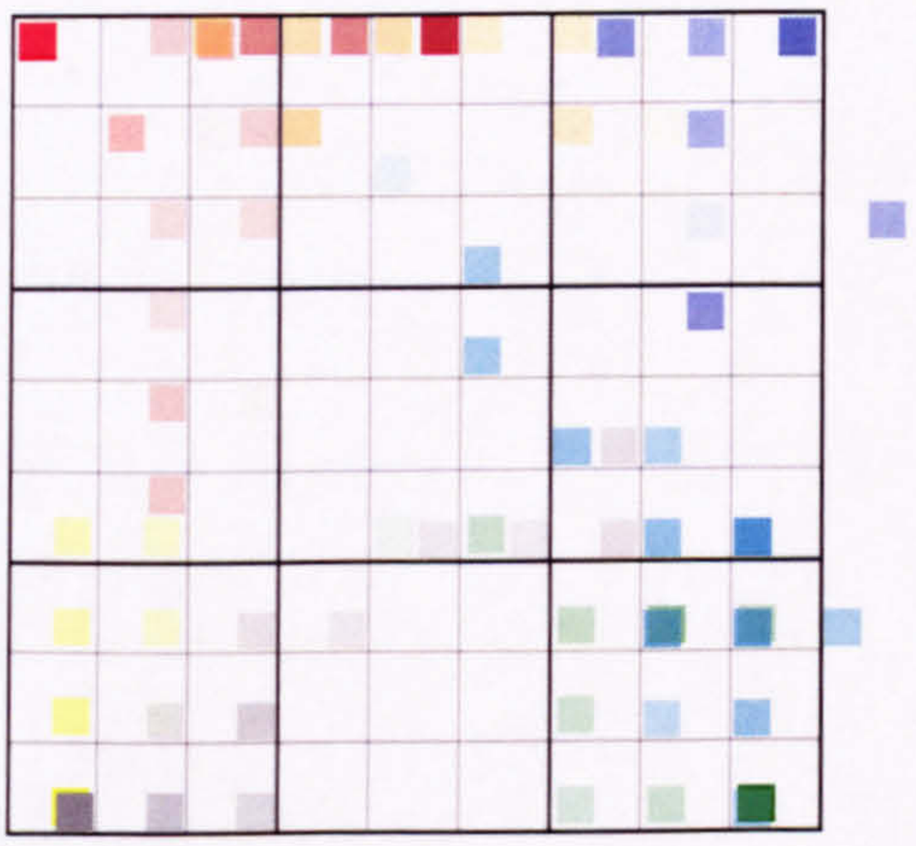


Figure IV-5 SVG visualization (8 users, *join*, 20 minutes)

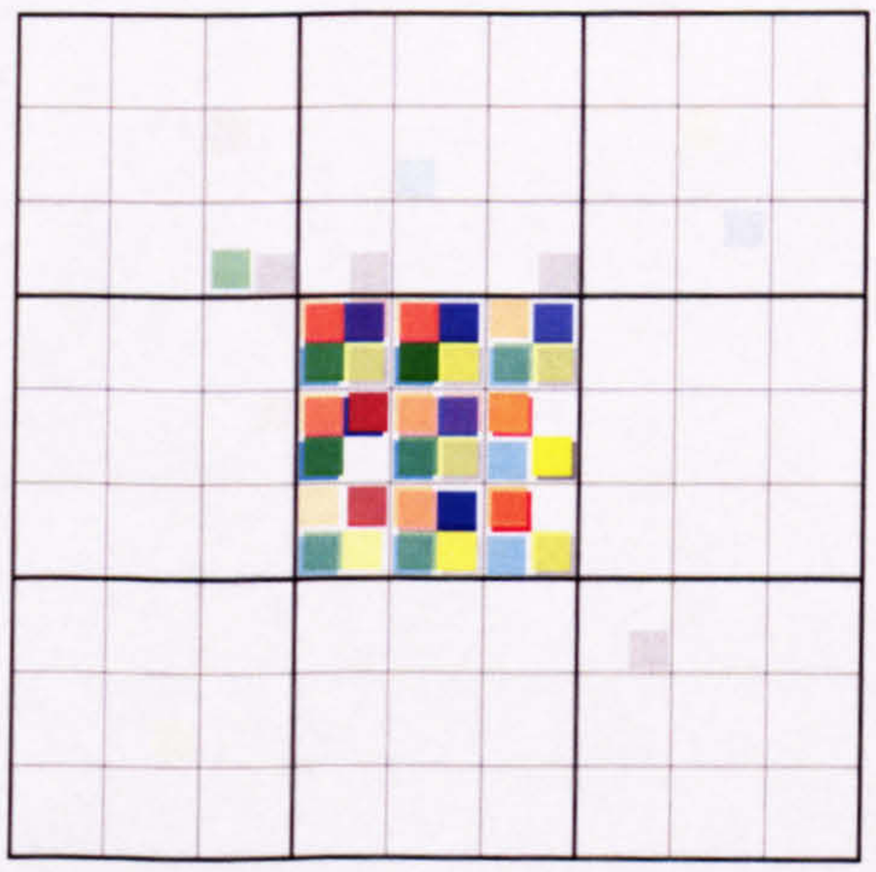


Figure IV-6 SVG visualization (8 users, *centre*, 20 minutes)