Enhancement of the scalability of an infrastructure for support of a large scale virtual and agent environment

A Dissertation submitted in partial
fulfilment of the requirements for the
degree of

MASTER OF SCIENCE

in Parallel and Scientific
Computation

in the

Faculty of Science

University of Reading

by

Oliver Otto

23. January 2001

Supervisors:

Dr. Dave Roberts
University of Reading
Great Britain

Dipl.-Ing. Gottfried Junghanns
Fachhochschule für Technik und
Wirtschaft Berlin
Germany

ACKNOWLEDGMENTS

The work in this thesis is a part of the COMRIS research project at the Reading University. I would like to take this opportunity to thank all those who have contributed to the work. Firstly, I would like to express my gratitude to my supervisor Dr. Dave Roberts for all his support, guidance, constructive criticism and ideas.

Furthermore, I am grateful to Robin Wolff who was very helpful whenever I needed advice and information about all sorts of practical questions.

In addition, I would like to thank Iain Werry and Farshid Amirabdollahian for reading through the whole dissertation and helping me with my English.

Finally, I am grateful to my parents for their encouragement and support over my whole study.

ABSTRACT

The dramatic improvements in global interconnectivity due to intranets, extranets, and the Internet has led to an explosion in the number and variety of new data-intensive applications. One of these applications are virtual reality systems, with their virtual environments. Current research in large-scale virtual environments can link hundreds of people and artificial agents with interactive three-dimensional (3D) graphics, massive terrain databases, global hypermedia and scientific datasets. One of these research projects is the Co-Habited Mixed Reality Information Spaces (COMRIS). A variety of network elements is required to scale up this virtual environment to an arbitrarily large size and simultaneously connecting thousands of interacting agents. The goal of this work is to show which elements can be used to enhance the scalability of the communication infrastructure. It is showing a generic overview of different design possibilities like the ring or the hypercube topology.

TABLE OF CONTENTS

LIST OF FIGURES

ABBREVIATIONS

COMRIS	Co-Habited Mixed Reality Information Spaces
3D	three-dimensional
CC	Conference Center
SP	Secretary Pool
TCP	Transmission Control Protocol
UDP	User Datagram Protocol
VE	Virtual Environments

GLOSSARY

Address	A name or token that identifies a network component. In local area networks (LANs), for example, every node has a unique address.
Agent	Autonomous Software which is searching and summarize information to send it in the real world
Broadcast	One-to-all, unreliable communication
Conference Center	The controlling central entity of the COMRIS system that manages global conference information including registration
Degree	The degree of a node is defined to be the number of its neighbours
Diameter	The longest path between any two nodes
Entity	A client of the infrastructure (For example an agent, a database or a text generation system
Links	A connection between nodes
Multicast	One-to-many, unreliable communication
Node	In networks, a processing location. A node can be a computer or some other device, such as a printer. Every node has a unique network address
Port	In TCP/IP and UDP networks, an endpoint to a logical connection. The port number identifies what type of port it is.
Publish Group	Abstraction of multicast group where only the creator is allowed send messages and add members

Scalability	Refers to how much a system can be expanded. Capable of being changed in size and configuration. The term by itself implies a positive capability. For example, "the device is known for its scalability" means it can be made to serve a larger number of users without breaking down or requiring major changes in procedure
Secretary	Interface between agent and infrastructure
Secretary Pool	Interface for a pool of agents and the infrastructure
Socket	The mechanism for creating a virtual connection between processes
Subspace	A region of the virtual space, with given properties and functionality to aid group interaction between agent interaction.
Unicast	One-to-one, reliable communication
Virtual Space	The space through which agents meet and interact

Chapter 1

1 Introduction

1.1 Overview

The dramatic improvements in global interconnectivity due to intranets, extranets, and the Internet has led to an explosion in the number and variety of new data-intensive applications. One of these applications are virtual reality systems, with their virtual environments. Current research in large-scale virtual environments can link hundreds of people and artificial agents with interactive three-dimensional (3D) graphics, massive terrain databases, global hypermedia and scientific datasets. One of these research projects is the Co-Habited Mixed Reality Information Spaces (COMRIS).

The COMRIS project aims to develop, demonstrate and experimentally evaluate a scalable approach to integrating the Inhabited Information Spaces schema with a concept of software agents. The COMRIS vision of co-habited mixed-reality information spaces emphasizes the co-habitation of software and human agents in a pair of loosely coupled spaces, a virtual and a real one. However, this project does not pursue the perceptual integration of real and virtual space into an augmented reality. Instead, the coupling aims at focusing the large potential for useful social interactions in each of the spaces, so that they become more manageable, goal-directed and effective.

The COMRIS project uses the conference center as the thematic space and concrete context of work. The conference center is a structure of places for registration, presentation, refreshment, and so on. At a conference, people gather to show their results, see other interesting things, find interesting people, meet officials in person, or engage in any kind of discussion. The possibilities of interaction at such an event are enormous, it is very

information-intensive, and the great diversity of topics and purposes that are being addressed make it difficult to get everything done.

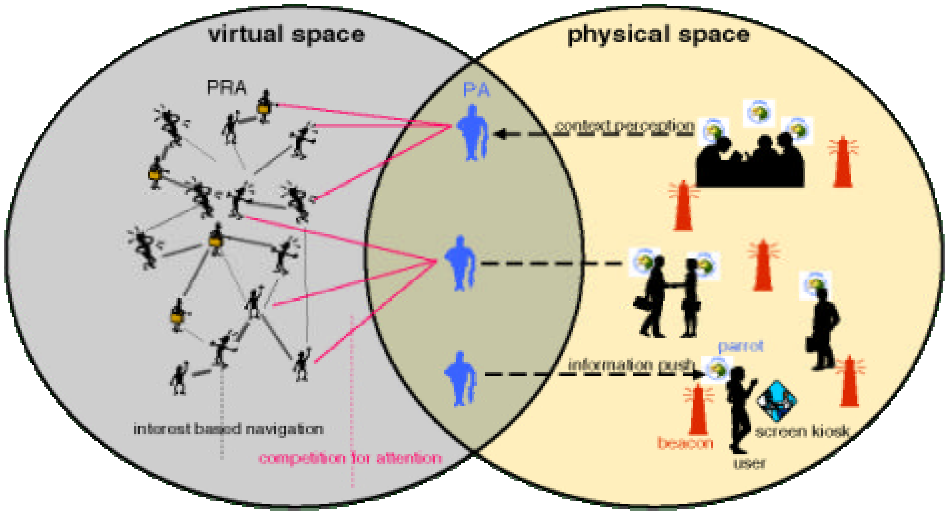

Figure 1-1 COMRIS overview

In the mixed-reality conference center, real and virtual conference activities are going on in parallel (as presented in Figure 1-1). Each participant wears its personal assistant (PA), an electronic badge and earphone device, wirelessly hooked into an Intranet. This personal assistant – the COMRIS parrot - realizes a bi-directional link between the real and virtual spaces. It observes what is going on around its host (whereabouts, activities, other people around), and it informs its host about potentially useful encounters, ongoing demonstrations that may be worthwhile attending, and so on. Several personal representatives (PRA), the software agents that participate on behalf of a real person in the virtual conference, gather this information. Each of these has the purpose to represent, defend and further a particular interest or objective of the real participant, including those interests that this participant is not explicitly attending [3].

To accomplish this task it is necessary that each agent communicate with the other agents in the virtual space. For this purpose, the agents send unicast messages (a point-to-point connection). Each agent can only

communicate with known agents, which they acquaint in special interest groups (also called subspaces). Such a subspace (which is centrally managed) is combing agents with a familiar topic and a place to get the name of other agents. Subspaces are like a chat room on the Internet, everyone can join, leave, listen and speak. In contrast to the subspaces exists publish groups. They are like a radio channel, everyone can listen but only one agent can send (the owner of this group).

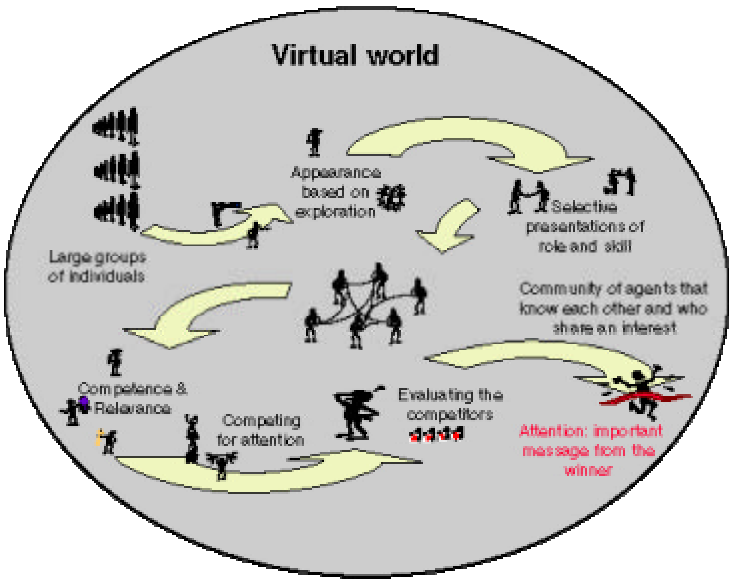

Figure 1-2 The Virtal world of COMRIS

Over these places in the virtual world, it is possible to find people in the real world with similar interests and to make real conversation (as presented in Figure 1-2).

1.2 Conceptual formulation

A variety of network elements are required to scale up virtual environments (VEs) to arbitrarily large sizes, simultaneously connecting thousands of interacting agents and all kinds of information objects. VE construction can include concepts and components from nearly any subject

area. The variety of desired connections between people and agents can be summarized by the slogan “connecting everyone to everyone”. As diversity and detail of virtual environments increase without bound, network requirements become the primary bottleneck.

The virtual space is the environment through which COMRIS agents will communicate and function. The virtual space is not only a conceptual home to hundreds or even thousands of agents, but also to other components with which these agents communicate. All such components and agents are termed entities and communicate via the virtual space communication infrastructure.

If n agents work in the virtual space and each agent sent each other agent one message, then n*(n-1) messages are generated to transmit over the infrastructure. For example, a conference with 100 members (per member exists 5 interest based agents) can generate 250.000 messages and a conference with 1000 members can generate 25 million messages [8]. This $O(n^2)$ transport problem needs a very scalable infrastructure to transmit all these messages with a small delay. Within the context of COMRIS, scalability concerns the ability to increase the number of users and agents without degrading the usability of the system [2] [9].

This work is trying to show ways to enhance the scalability of the current COMRIS infrastructure. It is discussing over theoretical possibilities and implementing some of then. The COMRIS Project started in 1998 and will finish in December 2000. Due to this condition, this work is trying to implement some changes to enhance the scalability and is talking about alternatives for projects with similar facts. The tests in chapter 4 show the effect of a small change on the whole infrastructure.

1.3 Organization of this thesis

This thesis is organised as follows:

- a short overview of COMRIS
- a conceptual formulation of my work
- what is scalability and why is it useful
- an overview about the conditions for a system like COMRIS
- a short brief introduction to addressing and routing (to understand the later sections)
- an overview about different kind of topologies and their influence on COMRIS
- further possibilities to enhance the scalability (direct connection, address multicast, etc.)
- analysis of some implementations (direct connection, address multicast, special name concept)
- conclusion

Chapter 2

2 Problems of network scalability

2.1 What is scalability?

Scalability is a popular buzzword that refers to how well a hardware or software system can adapt to increased demands. For example, a scalable network system would be one that can start with just a few nodes but can easily expand to thousands of nodes. Scalability can be a very important feature because it means that you can invest in a system with confidence you won't outgrow it.[1]

2.2 Conditions for a large scale virtual and agent environment

Each system has conditions, which describe the behaviour and the restrictions for the system. The following section defines and identifies the required specification, which is needed by the communication infrastructure of a virtual environment like COMRIS. It focuses on the communication architecture and on different technical aspects that play a role in the implementation of the virtual space of COMRIS [2].

2.2.1 General Architecture

The communication infrastructure provides support for dialogue between agents. An Agent interface to the infrastructure exists through secretaries and an individual secretary supports each agent. Both the implementation of the infrastructure and the location of peers are invisible to agents. In order to communicate to a peer or group of peers, an agent simply passes a message to the secretary.

[1] http://webopedia.internet.com/TERM/s/scalable.html

An overview of the infrastructure is presented in Figure 2-1, shown from the perspective of the Infrastructure Developer.

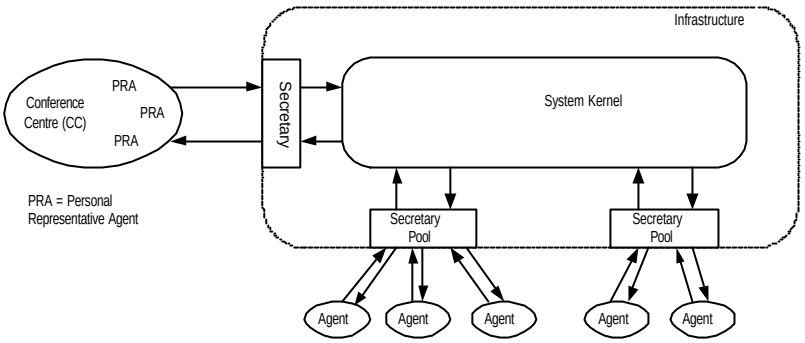

Figure 2-1 COMRIS infrastructure

The sequential initialisation of the infrastructure (in terms of communications from new agents), different type of communications services available from the infrastructure, is outlined in Figure 2-2.

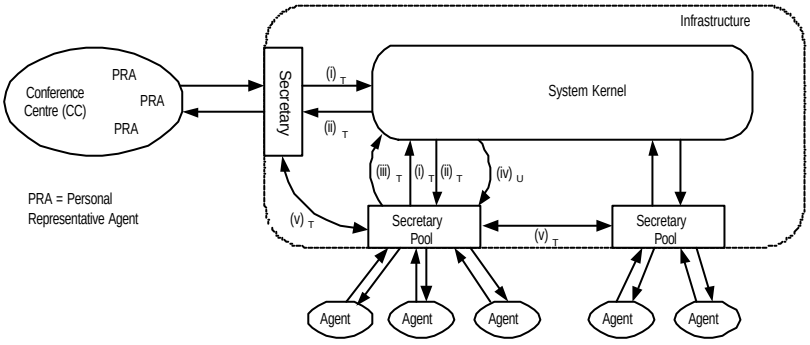

Figure 2-2 Secretary communication within the infrastructure

The numbers in parenthesis indicate the logical sequence of messages and the subscript indicates the type of communications, where U indicates unreliable communications protocol (Multicast), while T denotes a reliable communications protocol (TCP). The sequence is as follows:

(i) An agent communicates with kernel to register

(ii) The kernel accepts the registration.

(iii) An agent joins in a subspace via the central kernel.

(iv) The infrastructure sends a message to all member of the subspace, with the information about the new agent.

(v) Subsequent messages requiring reliable communications are sent via the central server or direct connections.

2.2.2 Subspaces

Subspaces represent (possibly overlapping) interest-based regions of the virtual space. A subspace may be thought of as a message forwarding service for groups of agents with a common interest. An agent may communicate to one or, sequentially, to all other members of a subgroup to which it belongs.

2.2.3 Publish groups

A publish group supports unidirectional one-to-many communication and is analogous to a mail distribution service. In contrast to Subspaces (to which an agent may request to join and leave, see below) membership of a Publish Group is under the direct and sole control of the creator agent.

2.2.4 Communication across the infrastructure

The infrastructure only supports the reliability of communication between secretaries. The secretary is not responsible for whether or not the agent actually reads or processes the message.

One-to-one communication between secretaries is based on TCP streams and may thus be considered reliable. Guaranteeing the reliability of one-to-many communication can have implications on timeliness, scalability and error recovery. For this reason reliable one-to-many communication is provided as an option that may be selected by the creator of the group. Both Subspaces and Publish groups will be given this facility.

2.2.5 Communication related requirements

Communication - Agents must be able to communicate with each other. Point to point messaging is a requirement. The Personal Agents are able to uniquely identify the senders and receivers of messages.

Communication reliability - When an agent receives a message it must also process the message. The agent is not obliged to really act according to the instruction in the message. It may decide not to perform what is stated in the message, but the agent may not simply dump or ignore the message before parsing it. This shall result in a non-repudiation communication where an agent can not falsely deny that it ever received the message.

2.2.6 Agent specific related requirements

The following requirements are described from the perspective of the communication infrastructure.

Identification – It must be possible to uniquely identify each agent. Personal Agents must have a unique ID (e.g. name). This uniqueness of this name is managed by the virtual space (i.e. the Agent Management System where agents can register themselves)

Autonomous – Personal Agents must be able to reason about their own state and act according to it. The internal state of the agent is not only changed through environmental stimuli, i.e. input and outputs, but also through internal processes.

Scalability – The agent performance may not drop unacceptably, if the number of competing personal representation agents grows. The personal agent must efficiently handle the management and communication

2.3 Why do we need network scalability?

A network must provide connectivity among a set of computers. Sometimes it is enough to build a limited network that connects only a few select machines. In fact, for reasons of privacy and security, many private (corporate) networks have the explicit goal of limiting the set of machines that are connected. In contrast, other networks (of which the Internet is the prime example) are designed to grow in a way that allows them the potential to connect all the computers in the world. A system that is designed to support growth to an arbitrarily large size is said to be scalable [1].

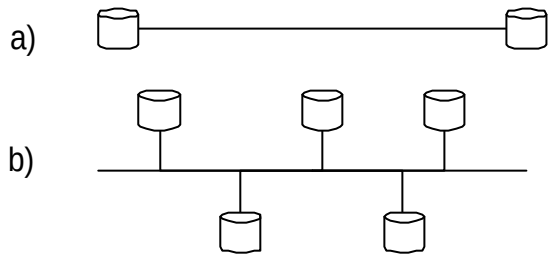

Figure 2-3 Direct links: a) point-to-point; b) multiple access

Network connectivity occurs at many different levels. At the lowest level, a network can consist of two or more computers directly connected by some physical medium, such as a coaxial cable or an optical fiber. Such a physical medium is called a link or connection and the computers that are connected through such a link, are called nodes. As illustrated in Figure 2-3, physical links are sometimes limited to a pair of nodes (such a link is said to be point-to-point), while in other cases, more than two nodes may share a single physical connection (such a connection is said to be multiple-access). Whether a given connection supports point-to-point or multiple-access connectivity depends on how the node is attached to the connection. It is also the case that multiple-access connection are often limited in size, in terms of both the geographical distance they can cover and the number of nodes they can connect. In the COMRIS project, the

links are not physical connections, they are logical links, but they have the same interpretation as the application layer (see Figure 2-4). This layer is represented by the user interface, agent intelligence, and so on. In contrast to physical networks (which are to find on IP Level 1 to 3), the COMRIS communication infrastructure is resided on IP Level 4. However, this is not a reason not to use physical network designs for a logical infrastructure.

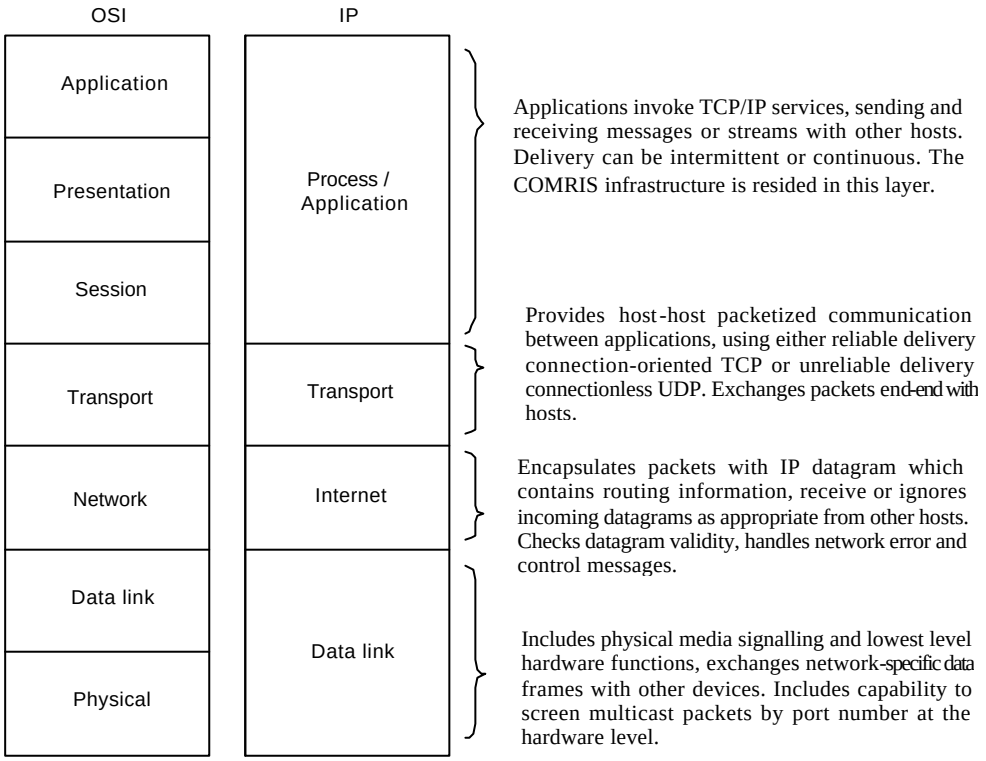

Figure 2-4 Correspondence between OSI and IP protocol layer model

A network with nodes can be described by two properties. Two nodes are neighbours if there is a link connecting them. The degree of a node is defined to be the number of its neighbours. A high degree means that a node has many direct connections to other nodes. For example, in Figure 2-3a the degree is 1 and in Figure 2-3b the degree is n (number of nodes on the bus). The diameter of a network is the longest path between any two nodes. A small diameter is preferable, because it decreases the delay of a message through the infrastructure. In Figure 2-3 both examples have a

diameter of 1. These properties are useful to recognise the scalability of a communication infrastructure.

If computer networks were limited to situations in which all nodes are directly connected to each other over a common physical medium, then networks would be very limited in the number of computers they could connect. On the other hand, the number of wires coming out of the back of each node would quickly become both unmanageable and very expensive. For a more logical environment, like the COMRIS infrastructure, is it meaning, that there are more open socket connections between the nodes and that needs more memory (which is not endless available). Fortunately, connectivity between two nodes does not necessarily imply a direct physical connection between them. Indirect connectivity may be achieved among a set of cooperating nodes. Consider the following two examples of how a collection of computer can be indirectly connected.

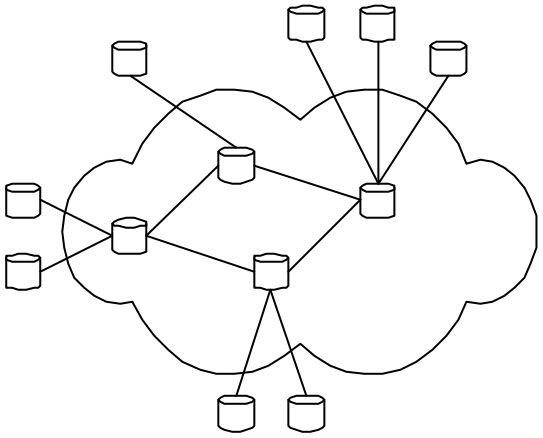

Figure 2-5 Switched network

Figure 2-5 shows a set of nodes, each of which is attached to one or more point-to-point links. Those nodes that are attached to at least two connections run software that forwards data received on one link out on another. If organized in a symmetric way, these forwarding nodes form a switched network. There are numerous types of switched networks, of which the most common are packet-switched and circuit-switched. The

important feature of packet-switched networks is that the nodes in such a network send discrete blocks of data to each other. Think of these blocks of data as corresponding to some piece of application data such as a file, a piece of email, or an image. Each block of data is called either a packet or a message.

Packet-switched networks typically use a strategy called store-and-forward. As the name suggests, each node in a store-and-forward network first receives a complete packet over some link, stores the packet in its internal memory, and then forwards the complete packet to the next node. In contrast, a circuit-switched network first establishes a dedicated circuit across a sequence of links and then allows the source node to send a stream of bits across this circuit to a destination node. The major reason for using packet switching rather than circuit switching in a computer network is efficiency.

The cloud in Figure 2-5 distinguishes between the node on the inside that implement the network (they are commonly called switches, and their sole function is to store and forward packets) and the nodes on the outside of the cloud that use the network (they are commonly called hosts, and they support users and run application programs). In general, this document uses a cloud to denote any type of network, whether it is a single point-to-point link, a multiple-access link or a switched network.

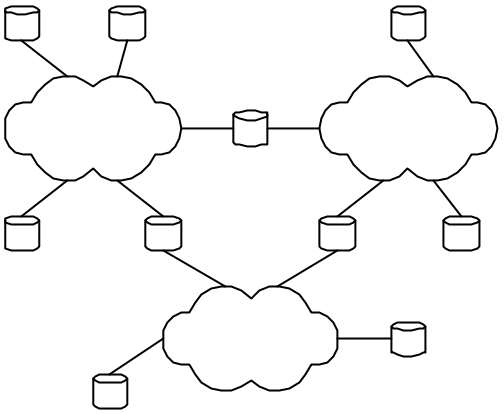

Figure 2-6 Interconnections of networks

A second way in which a set of computers can be indirectly connected is shown in Figure 2-6. In this situation, sets of independent networks (clouds) are interconnected to form an internet work or Internet for short. A node that is connected to two or more networks is commonly called a router or gateway and it plays much the same role as a switch. It forwards messages from one network to another. Note that an internet can itself be viewed as another kind of network, which means that an internet can be build from an interconnection clouds to form lager clouds. Router, gateways and switches are terms from physical networks. The COMRIS infrastructure takes over this part of the kernel, but the rule is the same.

Just because sets of hosts are directly or indirectly connected to each other does not mean that we have succeeded in providing host-to-host connectivity. The final requirement is that each node must be able to say which of the other nodes on the network it wants to communicate with. This is done by assigning an address to each node. An address is a byte string that identifies a node. The network can use a node's address to distinguish it from the other nodes connected to the network. When a source node wants the network to deliver a message to a certain destination node, it specifies the address of the destination node. If the sending and receiving nodes are not directly connected, then the switches and routers of the network use this address to decide how to forward the message toward

the destination. The process of determining systematically how to forward the message toward the destination node based on its address is called routing.

This brief introduction to addressing and routing has presumed that the source node wants to send a message to a single destination node (unicast). While this is the most common scenario, it is also possible that the source node might want to broadcast a message to all the nodes on the network. Or a source node might want to send a message to some subset of the other nodes, but not all of them, a situation called multicast. Thus, in addition to node-specific address, another requirement of a network is that it supports multicast and broadcast addresses [1]. This is used in the publish groups and subspaces, which are explained at section 2.2.2 and 2.2.3. The nodes can be agents, secretaries or kernels in the COMRIS project and the address is the name of an agent. It is not necessary for an agent to know the path (routing way) in the infrastructure, only the receiver name is important (as described in section 2.2.1). The infrastructure is searching for the shortest path to transmit the message. Nevertheless, for the infrastructure design it is important to think about the routing path. Chapter 3.1 describes different possibilities to keep this path as small as possible.

3 Possibilities to enhance the scalability

With a look at the specification of COMRIS (see section 2.2), we can imagine that a scalable system has to handle thousands or millions of messages, because each agent has to be able to communicate with each other agent. Also it is necessary to have a central point for the name management, only with such a point it is possible to uniquely identify the agent. Consider the following two examples of what situations the infrastructure has to handle in the real system.

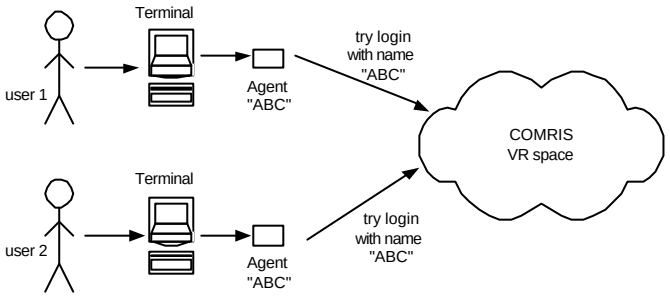

Figure 3-1 two user try to login with the same name

Figure 3-1 shows two users, which try to login to the VR space at the same time and with the same name. If we look to the COMRIS specification, we can see that this situation is not allowed, which means that one user should get an error message and this user has to try again with another name.

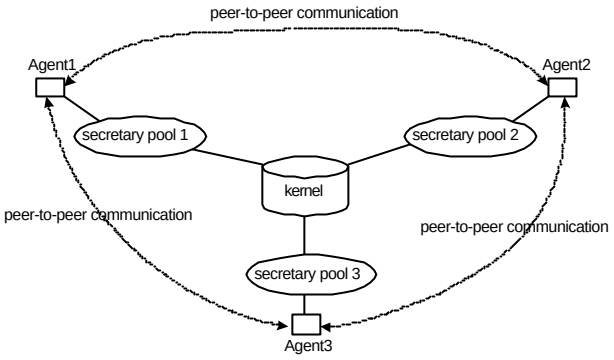

Figure 3-2 peer-to-peer communications between agents

The second example is a usual communication between two agents, which is shown in Figure 3-2. There are no restrictions, which can cut off a connection between any two agents.

There are different ways to enhance the scalability of a system like COMRIS. The following sections describe some ideas, which can be useful to make the infrastructure more flexible and faster.

3.1 Different network structure

The following discussion of the properties of different topologies is based on a collection of nodes that communicate via links. These sections speak about some ways to distribute the traffic or even to enhance the scalability for the nodes.

3.1.1 Star topology, the current COMRIS topology

The star topology is a very popular network structure. Most computer centers use the star. This topology is based on a central node (sometimes called the host) in the "middle" and all other nodes are arranged around this central point. The central node may just route all transmissions to their respective destinations or also carry out some processing of its own. Data is transmitted through the networks by the host calling each node in turn to determine if it has any data to transmit. (In some star networks, the

transmission is instigated by the individual nodes, which send an interrupt to the host to signal that it has data ready to be transmitted).

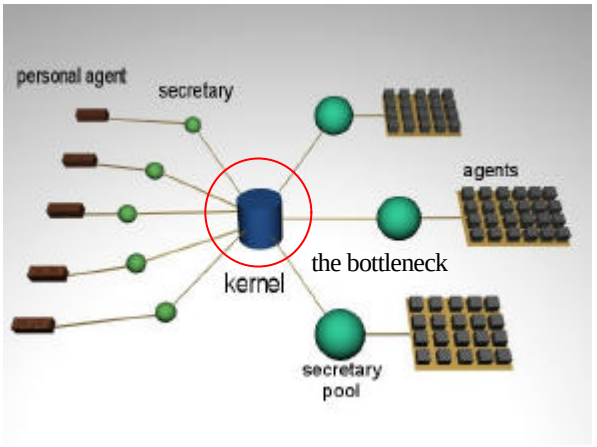

Figure 3-3 Example for a star topology

The degree of such a system is usually one (except the degree of the central node, which is depend of the number of nodes) and in addition, the diameter is very small with two. If an agent wants to speak with other agents, the communication goes through the central node. However, this is the bottleneck of this topology.

In this system, there is no problem with both the examples (see the beginning of this chapter). Even with the central node, there exists only one entity-list (in this list all agents are registered, with name, address and port number) and a double registration with the same name is not possible. The star topology is the current COMRIS network structure.

This topology is very easy to implement with all the COMRIS specifications, but unfortunately not very scalable. Chapter 4.6.1 shows the speed of the registration of up to 5000 agents and transmitting up to 20000 messages as soon as possible. In comparison with further methods this infrastructure is very slow (which are described in 3.1.5).

Star advantages:

- The response time is generally very fast, but this depends mostly upon the power of the host.
- Concurrent processing by the host is possible.
- Doubtlessly the uniquely identification (because the central node has the entire entity-list)

Star disadvantages:

- If the host fails, the whole network fails.
- The central node can only handle a certain number of nodes and the network cannot be expanded beyond this number (section 4.6.1 shows some results).
- As each node has its own communication line, there is a large cost in its initial installation.
- The central node is the bottleneck (entire communication is going over this point)

3.1.2 Ring topology, expandable as the star

Ring networks consist of nodes directly linked to each other by a single communication line. Messages travel from node to node around the ring until it reaches its correct destination. As with the bus network (described in the next section), each node must be capable of recognizing its own address to receive a message. If a message is passed to a node, which is not the correct destination, the message is transmitted to the next node in the ring.

The ring with a token-based concept

To control access to the line such that two messages are not transmitted simultaneously, a method called "token passing" is implemented. A token is a frame of bits, which is passed from one node to the next. The token may be "empty" or it may contain a message. If an empty token is received

and the node wishes to transmit data, it holds the token and writes into it: the destination address, its own address and the message itself. The token is then passed onto the next node. As the token is no longer marked as "empty", it ensures that nodes cannot transmit messages at the same time. When the token is finally passed to the node which has an address corresponding to the token's destination address, that node reads the message and then marks the token as being read. The token is then passed on to the next node and continues to be passed around the ring until it completes a full circuit and reaches the node, which originated the message. It is only at this point that the message is erased and the token is again marked as being "empty".

A ring of n nodes has a degree of two, because the node is always connected with two other nodes. The diameter of a ring grows as more nodes are added, so it is n/2 for a bi-directional ring. [5]

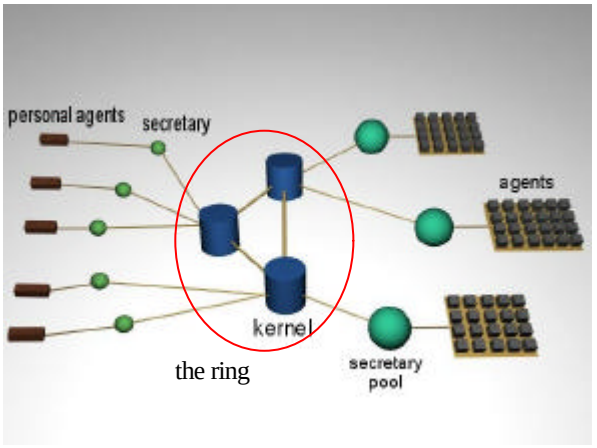

Figure 3-4 Example for a ring topology

The ring with atomic-broadcast concept

If the ring uses a token to synchronize the message transfer in the network, there is no problem with the agent registration or with peer-to-peer communication. Nevertheless, a disadvantage of the token system is the delay-time. A message can only be sent if the node has the token. Another

way to use the ring network topology and to be sure that the unique name concept of COMRIS is still possible is to use a reliable broadcast [17].

The reliable broadcast performs two phases of broadcast communication. A node knows that a message is reliably broadcast if the message (phase 1) and the related acknowledgment (phase 2) of all nodes are received. The atomic broadcast is an extension of the reliable broadcast. An additional numbering algorithm and sorting technique are used in the atomic broadcast. It ensures a globally unique order of broadcasted messages at all nodes, so that each node knows that every node got the message in an unique order, i.e. who tried the registration at first, if two secretaries try to register the same name.

The atomic broadcast concept allows communication between nodes without waiting for a token, i.e. for forwarding a message. However, it allows also the unique name concept for agent and publish group / subspace registration. With this concept it is possible to decrease the delay for the whole message transfer and thus increase the scalability.

The ring is a good way to enhance the scalability of the COMRIS infrastructure. This topology is easy to implement and much more scalable than the star network. Additional nodes can be added without effort and these nodes can be used for new secretary pools or kernels (which increase the number of members). If a node gets a message from a connected secretary or from a neighbour node, then this node has to wait for the token. With the token it is possible to send all messages, which arrived since the last time the node got the token. To prevent a node from keeping a token for a long time (while sending lots of messages) it is useful to use a maximum time to hold the token to send or to use instead the atomic broadcast concept. However, with the ring it is easy to use more than one kernel node or to use a distributed system (see section 3.2). Only the

overhead of transmitting a message is higher than the star topology, because the message transport needs additional operating cost.

Ring advantages:

- Very good for a medium number of nodes which require very high transmission speeds.
- Expansion is easily achieved.
- Unique registration is not a problem, due to using a token or atomic broadcast.

Ring disadvantages:

- Transmission delays are long even with light traffic.
- Each node must be turned on for the network to operate. (Or each node's attached network interface must be continually active.)
- Failure of a single node will halt a unidirectional ring network.

3.1.3 Bus topology, an impractical topology

A bus topology is one in which all devices connect to a common, shared connection (sometimes called the backbone). Most bus networks broadcast signals in both directions on the backbone connection, enabling all nodes to directly receive the signal. Some buses, however, are unidirectional: signals travel in only one direction and can reach only downstream devices.

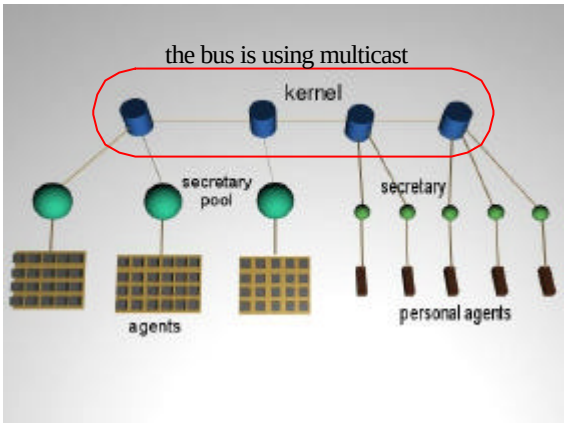

Figure 3-5 Example for a bus topology

Any node can communicate with any other node by broadcasting its message on the bus. All nodes continuously monitor the bus and when a message is detected which has the correct address code attached, that node acts upon the transmission. (Each node has its own network address.) Before any node may transmit a message, it must first "listen" to the bus to determine if any other transmissions are currently being broadcast. Once it determines that the bus is clear, it commences transmission. This process is known as "contention".

The degree of this network structure is unpredictable, because it depends on the number of nodes, which are connected to the bus. On the other side, the diameter is one due to all nodes are connected on one link.

Due to a missing central node (like in the star network), it is a problem to register two agents with the same name. A solution for this problem is to specify a node as the registration node (only this node can register an agent) or to use a token. With using a token, this structure is no longer a bus topology it is a ring topology (discussed in the previous section).

The bus is unsuitable for the COMRIS infrastructure. Not only is the name management difficult. The key problem is the information transmission at the bus, every message is broadcast to the other nodes on the bus. This creates a huge overhead for each message. For a small system with not more then 100 agents and an average of 100 messages per minute, this system maybe possible. For example, if we suppose that the broadcast of a message needs 10ms, than it needs 1 sec to send 100 messages but 100 sec for 10000 messages. However, a huge amount of messages makes the whole infrastructure extremely slow.

Bus advantages:

- Good for small networks with low traffic. (ie: Data is not frequently transmitted by the nodes.)
- Easy initial installations and easily expanded by adding extra nodes.
- If any one node fails, only the part behind that node is affected.

Bus disadvantages:

- The response time degrades rapidly as the data transmission load increases.
- Tapping into the bus causes transmission signals to be distorted.
- Unique registration is not warranted (no central registration).
- The bus is broadcasting each message and this increase the overhead.
- For logical infrastructures this is not suitable, the bus is more a physical topology.

3.1.4 Tree topology, not recommendable for COMRIS

The tree topology is similar to the star. A tree has three different types of nodes, namely a root node (or central node), interior node and leaf node, each with different degree. Usually, only the leaf node is connected to other component of the network (i.e. to the secretaries).

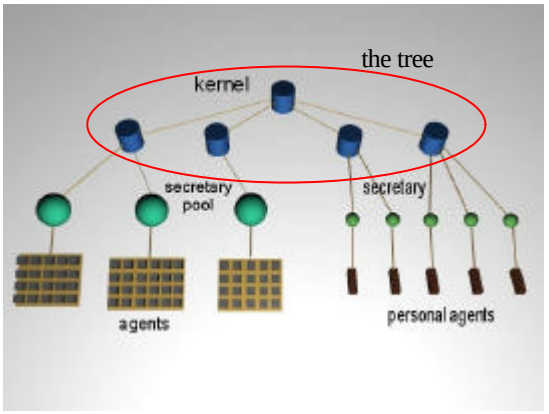

Figure 3-6 Example for a tree topology

The degree of this network structure is three (for a bi-directional tree) and the diameter is 2*log N (where N is the number of nodes). A high diameter makes the communication path for a message from one agent to another one very long. That means that the delay for a message can be very high.

The registration example (see beginning of this chapter) can be a problem, if the registration is not in the root node. If the registration in the leaf or interior node, than is a system needed, which prevents two different nodes registering the same name at the same time. Registration should be done in the root node, otherwise we might assign two different nodes a unique. After a successful registration or deregistration, the root has to send an update of the entity-list to all the other nodes.

A big problem in the tree is to find the direct way to another node of the tree. However, this is important to transmit a message or to register / deregister an agent. A solution to this problem is to use a special name concept for the nodes. Each level of the tree has it own number and in addition there exists a consecutive numbering from the left to the right side of each level (as presented in Figure 3-7).

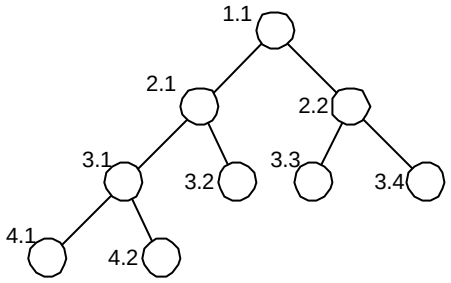

Figure 3-7 consecutive numbering of a tree

The extension of this tree is only to the right side and downward possible (it avoids a conflict with the consecutive numbering). For communication between two nodes it is now possible to calculate the path (it is necessary to go up or down). Without numbering, it is still possible to send the message to the root (which should have all addresses) to forward the message.

Multicast method, which is often used for physical trees, is not useful for logical trees as it will create huge amount of traffic over the whole tree. That would make the system very busy and decrease the scalability.

How useful is the tree for the COMRIS infrastructure? This is depending on the kind of traffic. The star, bus and ring topology have one thing in common, all nodes have contact with each message. The tree has a different concept and with a lot of “local” traffic (only over one or two nodes) is the tree to prefer. However, systems like COMRIS, which generate traffic through the whole infrastructure all the time, has a big

bottleneck – the root. It is like the star, but the overhead for each message is bigger.

Tree advantages:

- Expansion is easily achieved by adding extra nodes.
- The diameter of a tree (2*logN) is smaller then the diameter of a ring (N/2)

Tree disadvantages:

- Transmission delays can be long with a big tree.
- The network traffic near the root increase with a higher number of messages through the network.
- If any one node fails, the part beyond this node is affected (If the root fails, the whole network fails).
- It is not easy to find a specific node in the tree, it needs a special routing algorithm.
- Unique registration is not warranted, unless the root node is used for it.

3.1.5 Hypercube topology, high debatable

In a highly scalable topology, more nodes can be added without severely increasing the amount of logic required to implement the topology and without increasing the diameter. Such a topology is the hypercube. A link connecting two nodes defines a 1-dimensional "cube". A square with four nodes is a 2-dimensional cube, and a 3D cube has eight nodes. This pattern reveals a rule for constructing an n-dimensional cube: begin with an (n-1)-dimensional cube, make an identical copy, and add links from each node in the original to the corresponding node in the copy. Doubling the number of nodes in a hypercube increases the degree by only one link per nodes, and likewise increases the diameter by only 1 path [1].

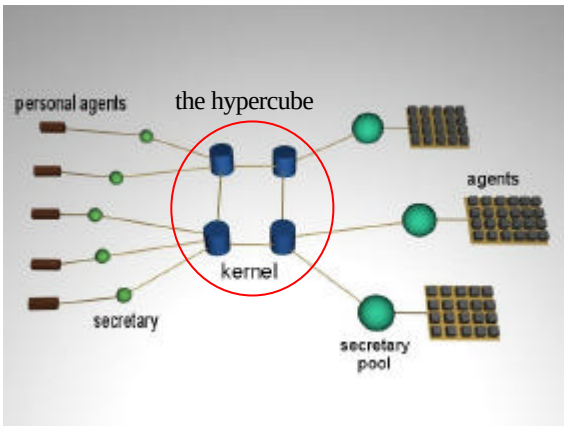

Figure 3-8 Exapmle for a hypercube topology

Communication in a hypercube is based on the binary representation of node IDs. The nodes are numbered so that two nodes are adjacent if and only if the binary representations of their IDs differ by one bit. For example, nodes 0110 and 0100 are immediate neighbours but 0110 and 0101 are not. An easy way to label nodes is to assign node IDs as the cube is constructed. When you œpy an (n-1)-dimensional cube, make sure the corresponding nodes in the two copies have the same IDs. Then extend all the IDs by one bit. Append a 0 to the IDs of nodes in the original cube, and append a 1 to the IDs of nodes in the copy.

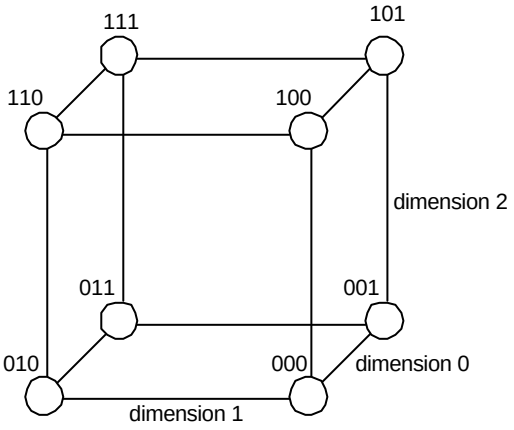

Figure 3-9 A 3-dimesional hypercube

Node IDs are the basis for a simple algorithm for routing information in a hypercube. An *n*-dimensional cube will have *n*-bit node IDs. Sending a

message from node A to node B can be done in n cycles, where on each cycle a node will either hold a message or forward it along one of its links. On cycle i the node that currently holds the message will compare bit i of its own ID with bit i of the destination ID. If the bits match, the node holds the message. If they don't match, it forwards the message along dimension i, where dimension i is the dimension that was added in the i^{th} step of the construction of the cube (i.e. it is the same "direction" at all nodes) [5].

The hypercube has the problem that a unique registration is only possible with a central node (which is not existing in a hypercube). A solution is a combination from hypercube and central node. It means that a series of nodes assume the central node part. All registrations and deregistration has to proceed on this special nodes. To avoid double registrations, it is necessary that all these special nodes be connected together, i.e. in a ring structure (see section 3.1.2). For the first example (see beginning of this chapter) it means that the registration request has to forward to the nearest central node, which can execute the request. However, to speed-up this procedure it is useful for each node to know the nearest central node. This can be achieved by calculating the shortest path to this special node. For example, each node has to know the address of these nodes and can compute the shortest path, using their own address (using a XOR over both addresses).

Unfortunately, the hypercube is not linear scalable. Each new dimension doubles the number of nodes, therefore, it is not possible to add only one node. To avoid one sided network load, it is necessary to distribute the load over the new nodes. This needs a special algorithm to spread the existing infrastructure over the whole hypercube. However, with the knowledge about the prospective infrastructure size (i.e. the maximum of conferee) it is also possible to start the hypercube with a stable dimension.

Broadcasting from Node x in an n dimensional binary hypercube can be performed as follows. First, Node x makes n copies of the broadcast traffic and forwards it to its n outgoing links. Then, a node receiving broadcast traffic on its dimension-k link forwards that traffic to its outgoing links that correspond to dimensions 0 through k – 1. Optimal and reliable broadcasting algorithms in hypercube can be found in [6] and [7].

Hypercube advantages:

- Fault-tolerant
- Very good for a high number of nodes.
- The shortest path between any two nodes is the dimension of the hypercube.
- If any one node fails, there are (n-1) paths left.

Hypercube disadvantages:

- Expansion needs a increasing of the dimension by one and doubles the number of nodes.
- It is only useful for big network structure.
- Unique registration is not warranted, unless using a special registration model.
- It needs a complex algorithm, to calculate the spreading of the network (if there a change to the next dimension).

3.1.6 Summary topologies

The previous sections described some different infrastructures. The word ‘node’ can be replaced by the word ‘kernel’, if we look at the COMRIS project. For this case, all these network structures are a substitution for the single kernel from the current existing infrastructure. The decision over the correct infrastructure should be chosen at the beginning of such a project. Only with the knowledge of the purpose and the specification, is it possible to make the correct choice of the kind of topology. This choice has an influence on the flexibility and scalability. It is a difficult decision to decide which topology is the best for an infrastructure like COMRIS. Only the star was implemented in the COMRIS infrastructure. Unfortunately, this work is written at the end of the COMRIS project and therefore another topology not implementable yet (except the star network). However, on the point of this work it is possible to say that the star topology is suitable for the COMRIS infrastructure. Until a few thousand agents (1000-10000) are connected over the secretaries to the infrastructure, the star works well (this is equivalent to a conference with 1000 members). Everything that is bigger should use another topology, like the ring or the hypercube.

Another desirable property of interconnection networks is node symmetry. A node symmetric network has no distinguished node, that is, the “view” of the rest of the network is the same from any node. Rings and hypercubes are all node symmetric. Trees and stars are not. When a topology is node asymmetric, a distinguished node can become a communications bottleneck [5].

Table 3-1 gives an overview of the different topologies and their assets and drawbacks.

Table 3-1 complete topology overview

	Star	Ring	Bus	Tree	Hypercube
Degree	for central node = n-1 otherwise = 1	= 2	= n	= 3 (for bi-directional)	= dimension (x)
Diameter	for central node d = 1 otherwise d = 2	d = n / 2	d = 1	d = 2 * log n	$d = \log_2 n$ with $n = 2^x$ 2)
Subspaces 1)	yes	yes	yes	yes	Yes
Publish groups 1)	yes	yes	yes	yes	Yes
Cross communication 1)	yes	yes	yes	yes	Yes
Uniquely identification 1)	yes	yes	not warranted	not warranted 3)	not warranted 3)
Scalable 1)	limited	limited	limited	limited	unlimited
Advantages	- easy to implement - currently COMRIS structure - a small diameter	- good for medium networks -expansion is easy - identification is firmed - degree is small -	- a small diameter - easy to initial -	- usable for bigger networks - expansion is easy - degree is small -	- for unlimited networks - fault tolerant - a small diameter and degree, as well for big networks -
Disadvantages	- central node is the bottleneck - not unlimited scalable - a high degree -	- delay longer for big networks - one node can block the whole ring - a high diameter	- for a virtual structure, difficult to implement -need multicast (on the bus) for peer-to-peer communication - a high degree	- it needs a special routing algorithm and names concept - the bottleneck is near the root -	- it needs a special concept to expand the dimension -

1) COMRIS Conditions possible or not
2) n = Number of nodes
3) only with special concepts

3.2 Centralized vs. decentralized architectures

The debate between centralized versus decentralized (also called distributed) architectures for multi-user applications is an old one. The two primary issues are performance and consistency. Decentralized architectures have been lauded for good performance. They require less network bandwidth since only input or state-changing information must be transmitted between nodes. Decentralized architectures also provide good feedback to the agents since locally initiated input is handled locally. There is no wait for the input to be processed by a central node and then transmitted out to the agents. In comparison, centralized architectures appear better at maintaining consistency among the other nodes. The central portion of the system sequences the various inputs from the other nodes (or agents) and ensures that every client sees the same changes at the same time [9].

Rendezvous is a good example of a centralized approach to building multi-user systems [11] [12]. Rendezvous relies on a central abstraction connected via bundles of constraints, or links, to multiple views. This is called the abstraction-link-view paradigm (ALV) [13]. In Rendezvous, the abstraction and the views all run as lightweight processes within the same heavyweight operating system process. Every user has access to a virtual terminal. From this terminal thy have access to a program called the Rendezvous Access Point (RAP), which is their entry into Rendezvous and allow them to use multi-user applications or to make user-user communication. Assume that there are n users in a conference. If every user provides some sort of non-conflicting input (such as scrolling a window or clicking the mouse at the virtual terminal), then $O(n^2)$ messages are sent through the network. Any single message requires one transmission to the abstraction and n-1 transmissions from the central abstraction to the other views. For each user of n users to send a message (n messages), this becomes $n*(n-1) = O(n^2)$.

The price in network usage, though, is not without merit. Rendezvous provides a reliably consistent view to each user. In fact, the communication mechanism worked so well that some applications relied on the reliable, sequenced broadcast of state changes even for updating the interface of the user who made the change [14]. This proved to be a simple and elegant way to write applications.

The Rendezvous abstractions and views described above actually ran within one process on a single processor. Assume that a distributed constraint system was implemented (as described in [12]) and that views ran on the users machines and not on the machine running the abstraction. Network traffic is still $O(n^2)$ as described above. However, if this system is then implemented on a network providing reliable, sequenced multicasting, the network usage is vastly improved O(n) = 2*n. Any single message from a client would be sent over a reliable connection to the central abstraction and is then multicast to every other client, resulting in two network transmissions. For n clients, this becomes 2*n, or O(n). However, the overall message latency is high because the abstraction is still processing every message.

In contrast, MMConf is a good example of the replicated approach to multi-user applications [15]. Although its performance is good (O(n) network messaging traffic in theory but no centralized bottleneck to add latency), in practice, applications built on top of MMConf quite often lost synchronization. In addition, applications were arbitrarily limited in their functionality. For example, MMConf explicitly used rigid floor control and token passing to avoid some of the synchronization problems. This meant that some users would have to wait to interact with the application or would not be allowed to interact with it at all. Besides user dissatisfaction, this floor control policy was a complicated piece of code that relied on unique tokens and sequence numbers to work properly--it often did not. As

another example, certain user-oriented features such as continuous scrolling were disabled, again to alleviate some synchronization problems. As a result, application programs presented unnatural interfaces to users or were less powerful than their single-user counterparts. Much of this is due to the fact that MMConf was not implemented with true, reliable multicast--instead it was implemented as best as possible on top of TCP/IP.

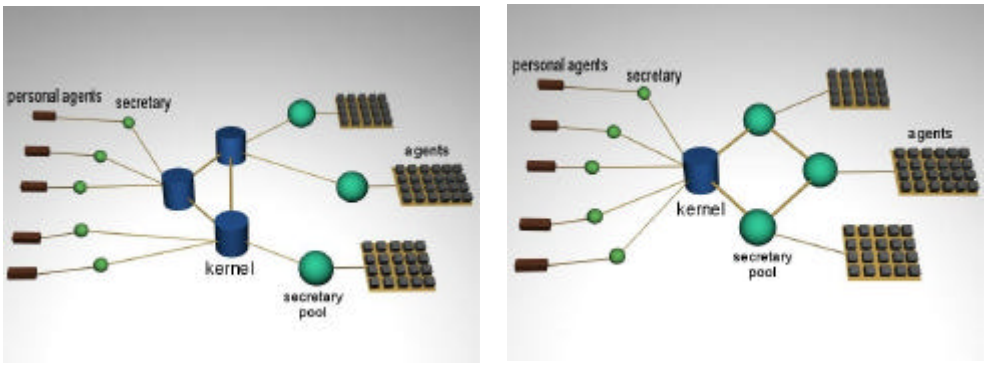

Figure 3-10 Centralized vs. decentralized architectures

The COMRIS infrastructure has three key conditions: communication (from each agent to each agent), identification and subspaces / publish groups. With a centralized architecture is this reachable. One problem of this architecture is that it is not endlessly scalable. There is a point on which the central node has to manage too much and the latency increases rapidly. A decentralized architecture has not this trouble, but in the COMRIS case another key problem exists. The identification and subspaces have to be unique and this is difficult to manage in a totally decentralized system, like at the MMConf project. A solution is a mixed architecture. For example, the Internet is such an architecture. Each Internet address is unique and it is possible to communicate with other members on the world wide web. Section 3.1 describes different architectures and each node means an independent kernel. If each kernel is connected to an amount of secretaries (and secretary pools), then it is possible to speak about a centralized system, because from a more abstract

point of view all kernels are together like one central node. However, what happens if the nodes are replaced by kernels and secretaries (or better secretary pools)? This is a mixed architecture, with some central parts (the kernels) but altogether it is decentralized.

3.3 Direct connections

Up to now, we have spoken about the distribution of traffic. All these structures are useful and needed for huge systems (with thousands or perhaps millions of agents). Another network structure, which was not explained, is a fully connected network. In such a network, each node is connected with each other node. This system has a diameter of one, but a degree of n (n equals number of nodes). Such a system is only workable for small networks and it is not very scalable. A variant of a fully connected network is a limited connected network. This means only a finite number of direct connections are possible. For example, if we say a secretary pool can have ten direct connections, than it is necessary that the secretary pool after ten connections closes the oldest one, to open a new direct connection.

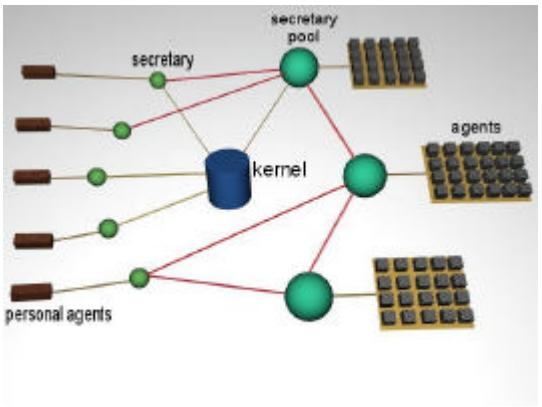

Figure 3-11 Example for a direct connection

Using direct connections, it is possible to bypass a lot of traffic around the kernel. The direct connections are an important part of the COMRIS infrastructure. Section 4.6 shows the influence to scalability. Through the exoneration of the kernel, the infrastructure is much more scalable (at 3 times). In addition, the direct connections are the first step to a decentralized architecture. The kernel takes over just the identification task and the management of the subspaces / publish groups, and the forwarding of messages in these areas.

3.4 Use subspace for address transport

The original version of the COMRIS infrastructure for subspaces used a simple method to send the name of each new member. Only the name was transmitted, with a single message to each member of this subspace (called unicast), and saved in the entity-list of the secretaries.

This method guaranteed that each member of the subspace got the information about a new agent. However, this is also a problem, it generates many messages, which have to be transferred over the infrastructure. For example, if there are 10 agents and they join one after one in the same subspace, it creates 45 messages (1+2+3+..+9) to tell the existing members of the subspace "There is a new agent". For 100 agents in a subspace it creates 4950 messages and this makes the whole infrastructure very busy.

$$\frac{n-1}{2}\cdot\left(a+a_{n-1}\right)$$

i.e. with: n=100, a=1 ,a_{n-1}=99
msg=4950

Each member of the subspace should get the information about a new member and therefore it is possible to use a multicast (see 2.3) for the transition. Now it creates only one message per entry and this unburdens

the infrastructure. In contrast to unicast the multicast is not reliable, but the probability is very high that an agent will gets this information (about the new agent) over another way (another subspace or direct communication).

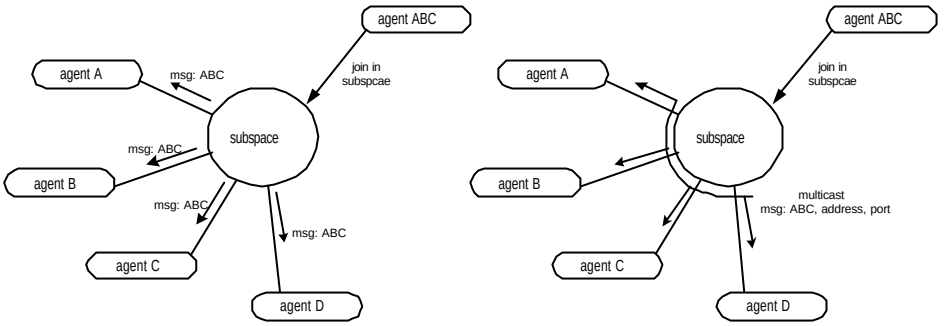

Figure 3-12 unicast vs. multicast entry message

Another key problem of the original method is, there is not enough information to create a direct connection (see section 3.3), which needs an address with a port number. Without this information, the secretary has to ask the kernel all the time for the address, or the kernel to forward the message. To avoid this situation it is better to multicast the name, the address and the port of a joining agent (as presented in Figure 3-12). Obviously, it makes the entry message bigger, but it removes the kernel load. This is an important feature for the infrastructure. The subspace is usually used as an exchange platform for names from other agents with similar interests. It helps to decrease the kernel load and therefore the kernel is less busy for an important job. Section 4.6.3 compares the difference between both variants.

3.5 Special name concept

There are different ways to enhance the scalability of a system like COMRIS. One way is to increase the capacity of the network structure, to handle all the traffic over the network. Another way is it to reduce this traffic. Such a way is to give each agent name a special extension, to

recognise the source of this agent. For example, in the COMRIS project each agent can talk with each other agent, but the secretaries do not know where in the network the receiver agent is. Of course, in the current project each agent has an entity, which should include the name, the address and the port. Nevertheless, sometimes this information is unknown and they need a lot of memory. That sometimes only a name is known is conditional on agent intelligence or through external input (from the real world). The only thing which is always available is the name of the agent.

The idea of this special name concept is an extension of the agent name. For example, if agent ABC is a member of secretary pool 1, and then this agent gets the extension 1 (ABC#1). If the agent moves to another secretary pool, i.e. to secretary pool 2 the new extension is 2 (ABC#2). In case that this agent is a personal agent (directly connected to a real person), the extension is 0 (ABC#0). Normally, there is a direct connection between the secretary pools (see section 3.3), but this connection has the name of the creator agent (conditional on the programming of COMRIS). The problem is, if an agent (ABC) from the same secretary pool tries to make a connection to an agent from the other secretary pool, the infrastructure does not know that an older connection (to DEF) exists. The secretary pool 1 sends the message to the kernel and this forwards the message to secretary pool 2, which submits the message to the mailbox of agent DEF. With this extension it is possible to save the connection under the universal secretary pool name, instead of the agent name. Now it is possible for the infrastructure to recognize that an agent is trying to make a link to a secretary pool, which already exists. The main benefit can be achieved for agents in secretary pools, because they are the majority in the infrastructure and they create the primary traffic between secretary pools.

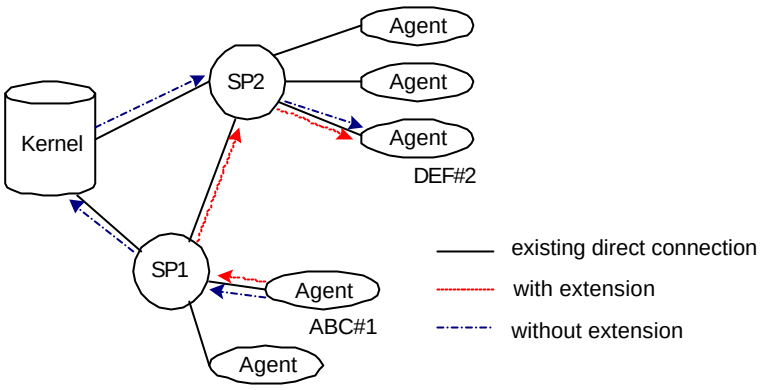

Figure 3-13 expample for a message path

To get the full effect for the COMRIS infrastructure it is necessary that this extension is always bonded with the agent name. Unfortunately, at this moment it is not longer completely realizable. The COMRIS project is to advance to expect of the other COMRIS partner that they implement this concept now. The current version of the infrastructure is using this concept, but the extension to the outside world is hidden. That makes it not so effective (see section 4.6.4). Nevertheless, this concept was very useful for the visualization module of the COMRIS infrastructure It is now possible to associate an agent to a secretary or secretary pool [10].

Chapter 4

4 Implementation, Integration and Tests

4.1 Description of existing COMRIS infrastructure

After three years of development the main part of the COMRIS infrastructure was already implemented and working at the beginning of this work. The infrastructure of COMRIS is written in JAVA and each kernel, secretary and secretary pool is running in its own java virtual machine.

The communication infrastructure enfolds 65 class-files with over 500 methods. Due to the fact that the infrastructure was originally written in C and then transformed to JAVA, the object oriented approach of JAVA is not complete. This makes changes at the source code more complicated or requires implementations two or three times. At the beginning of this work, the COMRIS infrastructure was basically done, but only the kernel was used for communications between the secretaries. For the following implementations it was partly necessary to make some changes to the flow diagram. Especially the send and receive message needed a partial redesign.

The general functioning of the communication infrastructure is described in Figure 4-1. It should show the event trace for different activities. The first step, after the initiation, is the registration of the secretaries and secretary pools and, of course, of the agents. Then it is possible to register the subspaces and publish groups, in which the agents can join and start to exchange information. If an agent finds other agents with similar interests, the direct communications are started. After the information exchange, it is necessary to send a message to the real world and to give the status of the information exchange to the conference member.

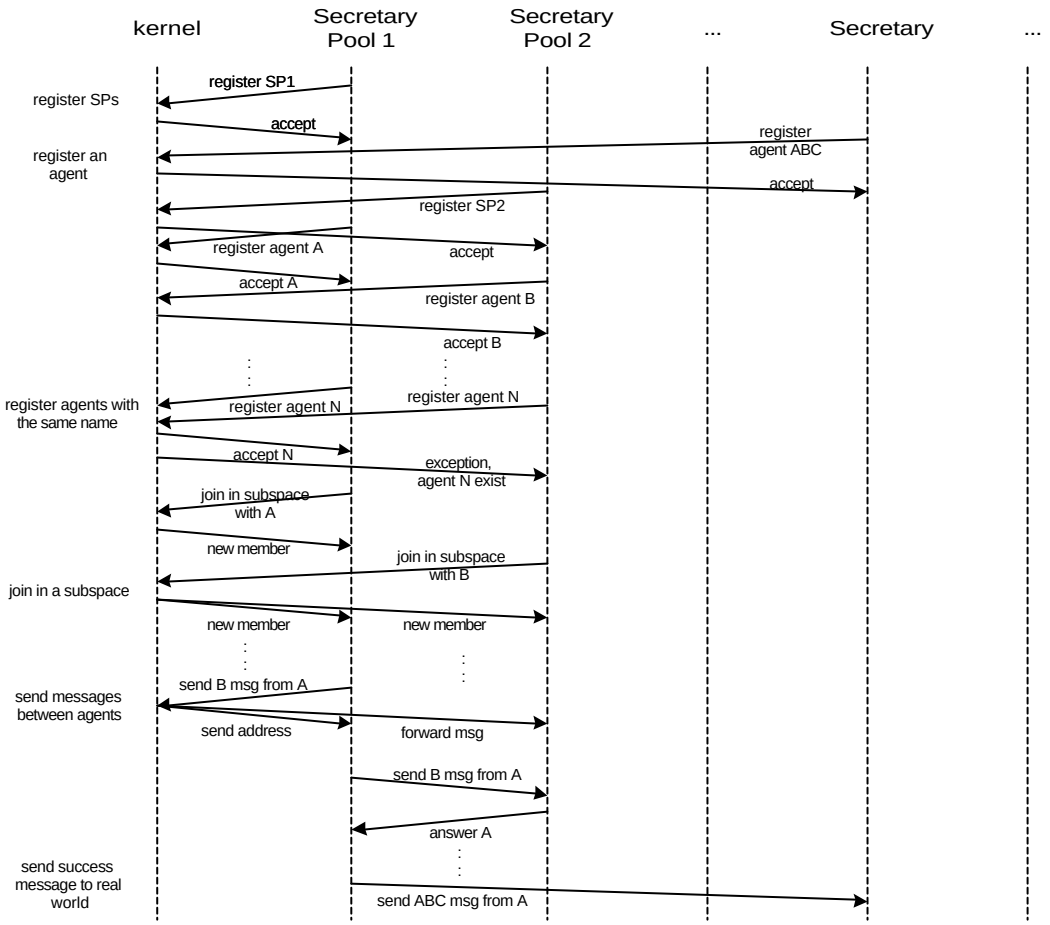

Figure 4-1 Event trace of the COMRIS infrastructure

The message format is XML, which implies an XML parser is needed to translate the message and to figure out the next action according to this message. With this proper format, it is also possible to have an interface to other agent systems and to exchange information with them (provided in later versions). For example, the Visualisation Tool is using parts of the basic secretary method and sends and receives messages (which are based on XML) to get the status information about the infrastructure [10]. Due to the fact that the Visualisation Tool is like a secretary, but with another assigned task, it was necessary to implement some service routines for this tool. The additional methods replay the request for information of the visualisation tool.

4.2 Implementation of direct connection

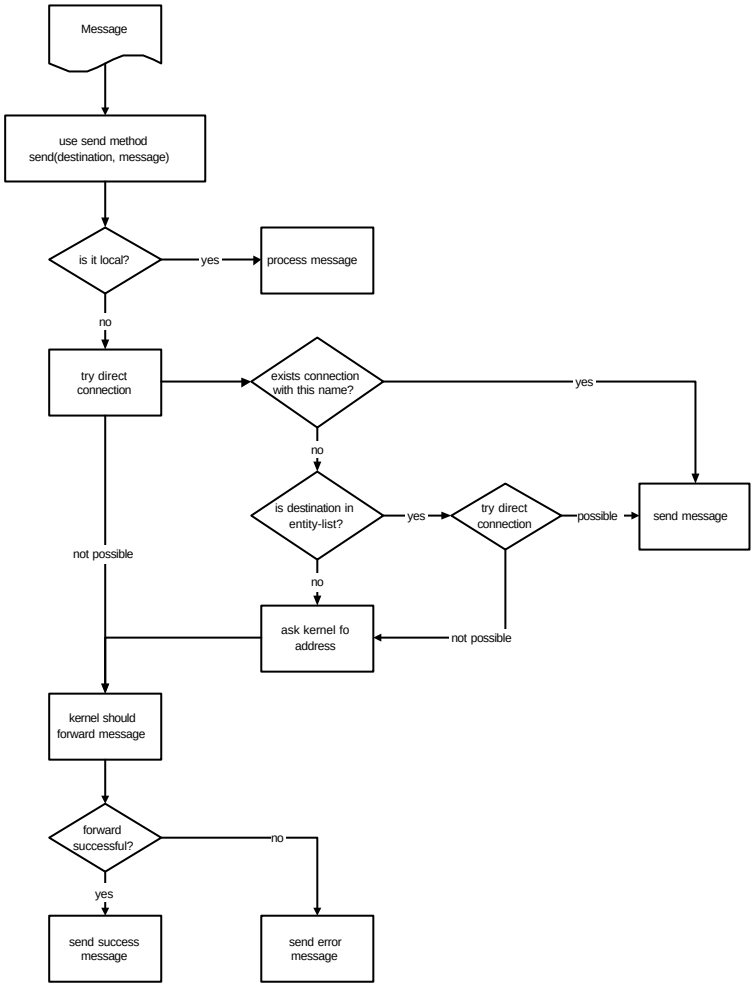

Figure 4-2 Message flow diagram for direct connetion

The first thing that was implemented was the direct connection. For this purpose the direct connection method was integrated in the send method. Now, when an agent sends a message to another agent, the send method tries to make a direct connection. Figure 4-2 shows a message flow diagram. The success of the direct connection is shown later on in this chapter.

4.3 Implementation of address multicast

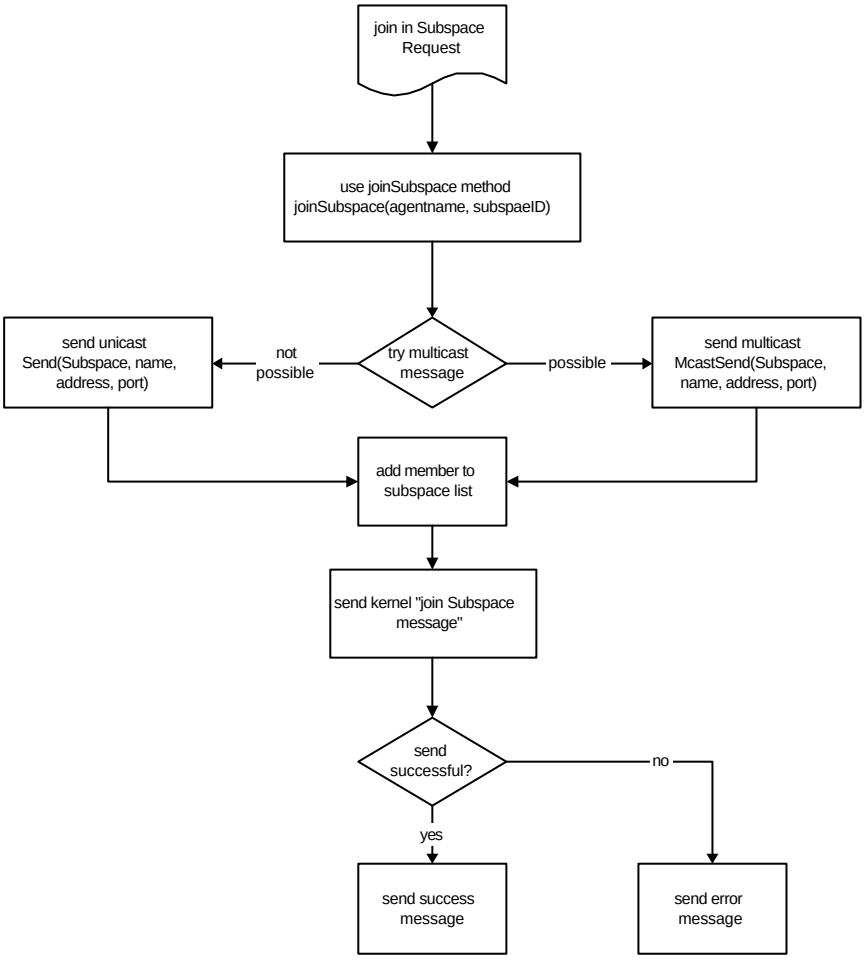

Figure 4-3 Flow diagram for address multicast

The second implementation was the address multicast. The working principle is quit simple, as seen in Figure 4-3. If an agent joins in a subspace it is important, for the existing agents in this subspace to know not only the name but also the address and the port of the new agent. Obviously, this makes the multicast message bigger and takes more memory at the entity-list, but the multicast is much faster than a unicast message to each subspace member. It makes the direct connection easier

and reduces the traffic over the kernel, because now it is no longer necessary to ask the kernel for the address of the destination agent.

4.4 Implementation of special name concept

Another update was the special name concept. This function can be implemented in different ways. One possibility is to enhance the agent name with an extension, which is represents the secretary or the secretary pool. For example, instead the name ABC is the new name ABC#0. The extension number uses the number of the secretary pool (which is always starting with 1) and an agent from a secretary has the number 0. With this model (as presented in Figure 4-4), all personal agents in a secretary have zero, but it does not matter, because these agents do not produce so much traffic and they are in the minority.

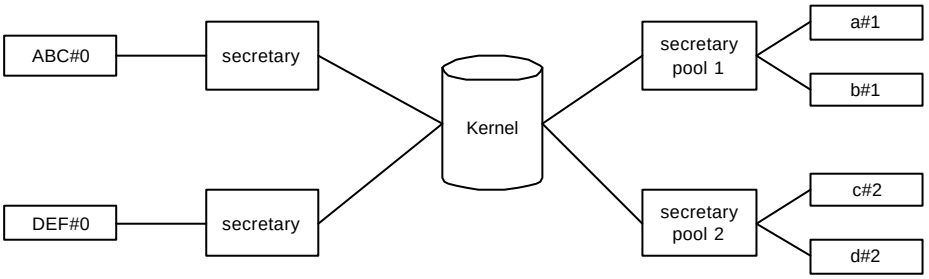

Figure 4-4 each agent has an extension

The old model: If an agent tries to send a message, the direct connection method has to compare the name of the destination and the existing connection. If such a connection is not available, the method has to search for a connection with the same address and port number. The problem is, if the address and the port number are unknown by the agents. In this case, the message has to be sent to the kernel and forwarded to the destination.

The new model: Now, if an agent tries to send a message, the direct connection method looks for the extension and compares just this extension with existing connections to other secretary pools. This should reduce the comparisons and the requests at the kernel.

Unfortunately this model is not applicable now, because the COMRIS project is at its end and this model needs changes not only at the infrastructure level. Another possibility to implement this concept is to expand the data set entry. The name, the address and the port number of an agent is saved in the dataset entry. A new entry is the extension number. The difference to the upper model is, this entry is saved by the secretary and is not visible to the outside (as presented in Figure 4-5).

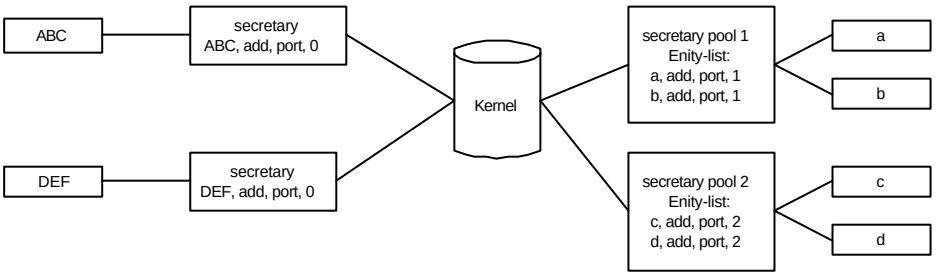

Figure 4-5 the extension is saved at the secretary

4.5 Description of Controlled Experiments

The Infrastructure layer provides the COMRIS agents with many different communication protocols. However, much of the functionality is hidden to provide simplicity of use. To initiate communications, the agents only need to talk to their dedicated Secretary, which contains an interface into the Infrastructure. The Secretaries and the rest of the Infrastructure, i.e. the central Kernel, can then deal with the mechanism of sending or receiving messages. It is clear therefore that the underlying speed, reliability and scalability of the Infrastructure are vital to the upper layers of the project.

The following experiments aim to test the characteristics of the Java Infrastructure with different versions. The most important factors are scalability, i.e. the relationship between the size of the Infrastructure and its performance, and reliability. The scalability can be broken down into the following key areas for experimentation:

- number of registrations
- determine outcome of direct connection
- determine outcome of address multicast
- determine outcome of special name concept

The size of the Infrastructure layer is limited by the performance/capabilities of the machine(s) that it resides on. There is, of course, a limit to the amount of hardware that can be dedicated to this task so realistic numbers of registered agents running on a single machine are required. Currently the Kernel acts as a central name server and resides on a single machine. As the Infrastructure grows the load on the machine running the Kernel will also grow. Measuring the performance of different machines, whilst running a Kernel through a large registration session, should give a quantified answer to how the above constraint will restrict

the Infrastructures performance, and how many agents can successfully register in a set period of time.

The size of the Infrastructure also affects the frequency of message sending as there are more entities wishing to send messages. A critical point or bottleneck for the message sending will occur at the registration phase, numerous agents wishing to register with the Kernel at once. Another critical time is when agents join a subspace.

When dealing with the message sending itself there are two main variables; the size of the messages and the frequency at which they are being sent. Both of which affect the message delay. Whether the messages are coming from one sender or from many senders, from the viewpoint of the receiving Secretary the only difference is the frequency of the incoming messages. This is because each communication socket has an independently threaded Connection class, which calls the Secretary when it has received a message. For these tests the role of the COMRIS agent has been replaced by a test agent to replicate all the typical functions of an agent.

To prove that one solution makes the system more scalable than the other solution, it is necessary to simulate a realistic behaviour. The only way to do this, is to use random factors, which means that the whole test is based on random parameters. Of course, there are some static conditions like number of agents or number of sent out messages, but settings like which agent is in which secretary pool or which agent is doing a peer-to-peer connection are random. With random variables it is essential to repeat the tests more than one time. With enough repetitions it is possible to get a good average of testing results.

The tests were ran on equivalent machines with 128Mbyte RAM, 500 MHz processor and 100 MBit Network card on a network with the same speed. On the machines was installed Windows 2000 and Java 1.2.2.

The following test pattern was used to compare the different COMRIS versions.

```
Start Kernel
Start Secretary Pool 1 to xxx
for SP1 to SPxxx
        create yyy Agents
        save time (start time)

for SP1 only
        create Subspace 1 to www

for SP1 to SPxxx
        for Subspace 1 to www
                create random number R1
                for R1
                        select random agent R2
                        join with R2 in Subspace
        save time (time to join in a subspace)

        for zzz number of messages
                select random agent A1 (from own SP)
                select random agent A2 form entity-list
                send message from A1 to A2
        save time (time to send zzz messages)
```

this are the variable parameters:
www – number of subspaces
xxx – number of secretary pools
yyy – number of agents per secretary pool
zzz – number of messages

All tests were repeated five times, to get a rough average. In summery, the tests have resulted in thousands of measured values (ca. 25000 results). Each test needed synchronisation between the secretary pools and the kernel, because it was important that each test was started and stopped at the same time. Furthermore, each test needed a long execution time (more than 36h), due to the different parameters and recurrences. With the identical hardware and a fast network it is possible to disregard the

influence of the physical network and hardware. Performance tests with low speed machines (166 MHz) increased the time per measuring by a factor of four, but the general difference between the different infrastructures versions was the same as before.

4.6 Results of Experiments

4.6.1 Registration Performance

An earlier test was trying to show the registration time for several agents. The test in this document could confirm these results. Figure 4-6 shows the time taken for up to fifty thousand registrations from one Secretary Pool to the Kernel. Both the Kernel and Secretary Pool are running on the same machine (Windows 2000). As the number of registrations increases so the processing of these requests slows down, this is quantified in Figure 4-7.

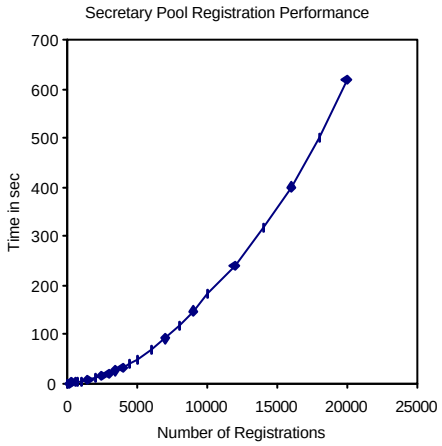

Figure 4-6 Secretary Pool registration times

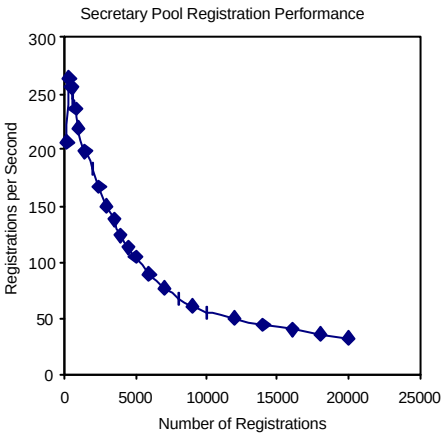

Figure 4-7 Number of registrations per second

In the results the registrations take increasingly longer time beyond one thousand registrations. As long as registration is reliable when overworked

which the results prove as no registrations failed, then such behaviour is acceptable. A perfectly scalable system would produce a horizontal line.

4.6.2 Test direct connections

To prove that direct connection can disburden the kernel, it was necessary to test an infrastructure version with and without implemented direct connections.

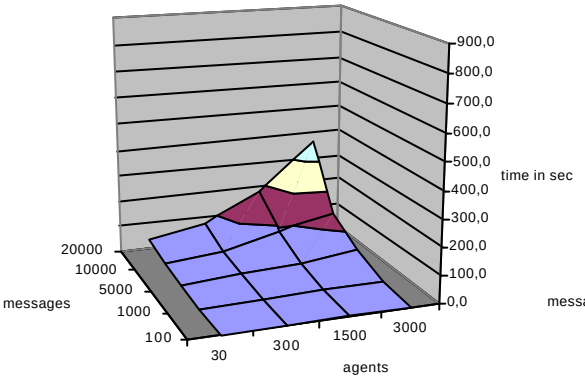

Figure 4-8 using direct connections

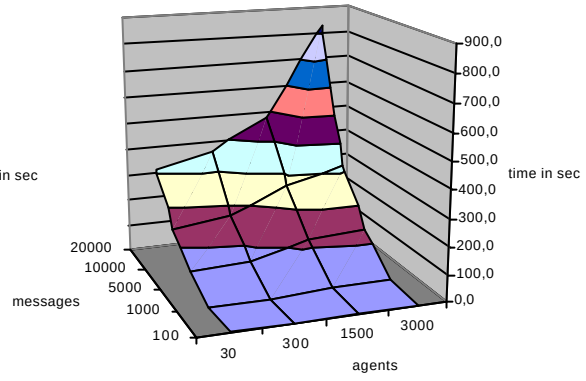

Figure 4-9 no direct connections

Figure 4-8 shows the time to send 100 to 20000 messages between 30 to 3000 agents, using direct connections. In comparison with Figure 4-9, which is using the old COMRIS version without direct connections, it needs 2 ½ less time to send 20000 messages between 3000 agents. In the Figures it is good to see the influence of the number of messages. A small amount of traffic does not need so much capacity from the kernel and so there is no big difference between both versions. In contrast, a high level of traffic needs too much capacity from the kernel and it is not able to forward all messages fast enough.

4.6.3 Test address multicast

The implementation of the address multicast was not so complicated, but the influence to the efficiency and scalability is very high. Figure 4-10 shows the curves of an infrastructure version with and without address multicast over the subspaces.

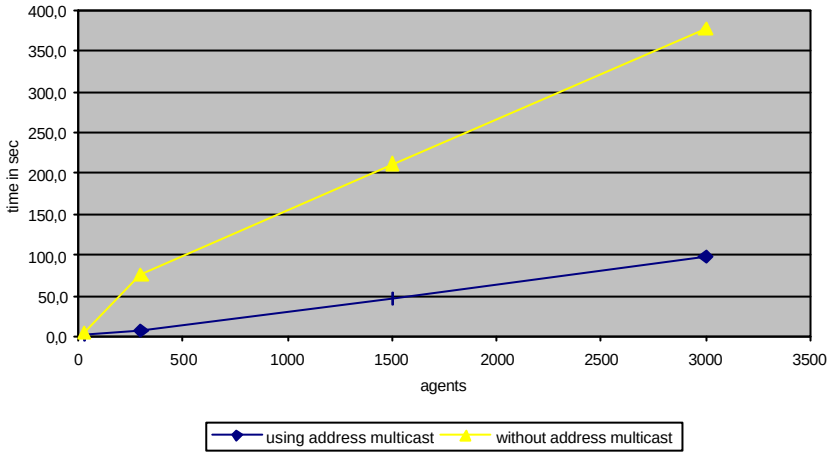

Figure 4-10 influence of address multicast

It is easy to see that the old version increases faster than the new version and the time to send the multicast message is less than a 1/3 of sending unicast messages to each subspace member.

4.6.4 Test special name concept

The special name concept should reduce the check-ups to find existing direct connections between secretary pools. Figure 4-11 to Figure 4-14 shows the influence of this concept for three and for five secretary pools.

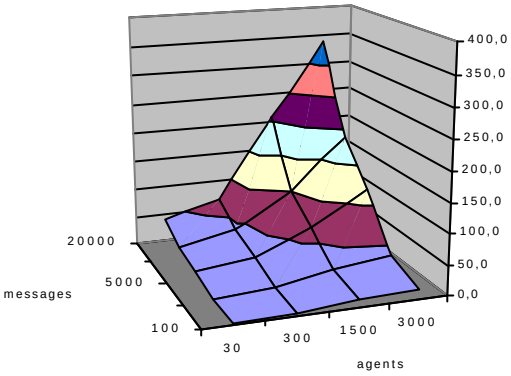

Figure 4-11 using name concept with 3 SPs

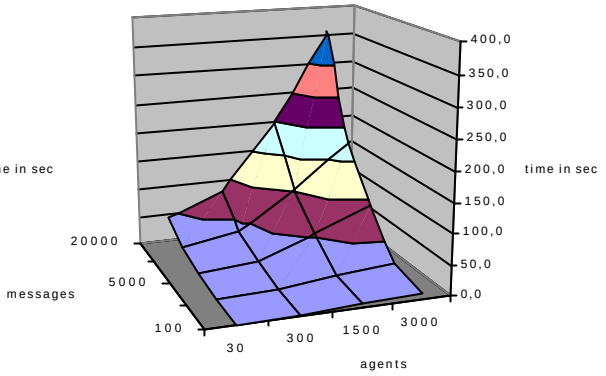

Figure 4-12 no name concept with 3 SPs

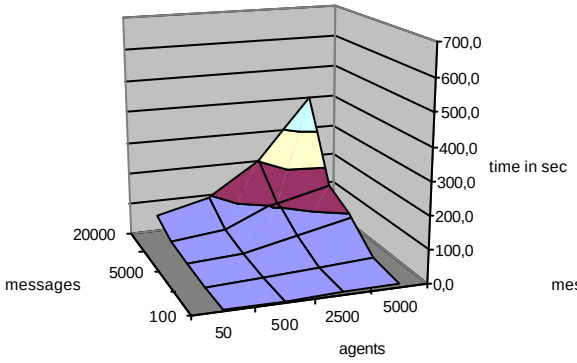

Figure 4-13 using name concept with 5 SPs

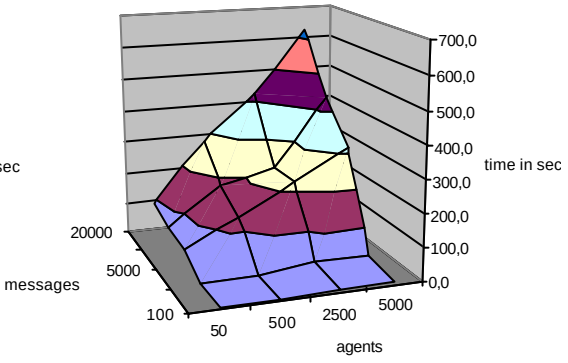

Figure 4-14 no name concept with 5 SPs

The influence of using the name concept in Figure 4-13 is higher than in Figure 4-11. The reason for this behaviour lies in the number of secretary pools. More secretary pools mean more connections between them and this improves the search effects of this concept.

The efficiency of this concept is not as high as the direct connections. One reason is that this is not the ideal implementation (the version which is using the name for the secretary pool number), but also that this concept needs more memory to save the additional information.

4.6.5 Test number of applications per machine

Another interesting question was the influence of the java virtual machine (JVM) to the performance. Is it useful to start more than one secretary pool on the same machine, only to separate different interest groups or similar things? The answer is, no. Each secretary pool has its own JVM and using more than one secretary pool takes more memory and processing time from the machine. A solution is to allow to more than one secretary pool with the same JVM (it is to implement in the source code of the secretary pool). However, why should it be useful? For example, two secretary pools need two separate entity-lists, more memory and more processor time than one pool with the same number of agents. Figure 4-15 and Figure 4-16 show the difference between one and two secretary pools on a machine with the same number of agents.

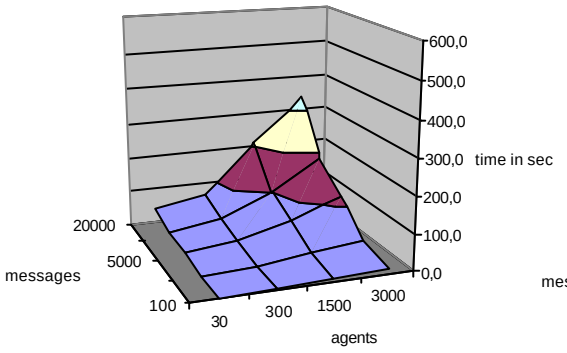

Figure 4-15 running 1 SP per machines

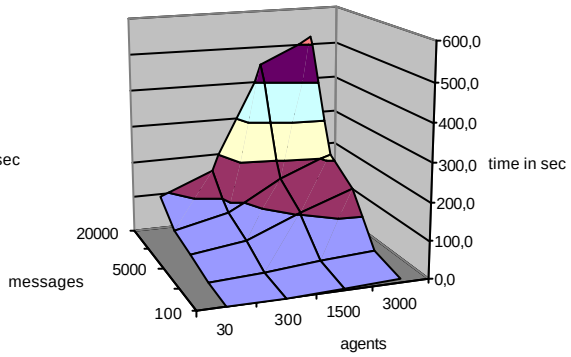

Figure 4-16 running 2 SPs per machines

4.6.6 Summery tests

A number of observations and interpretations can be made from these graphs. Registration of small amounts of agents is fast. Scaling up to a larger number of agents and also messages shows drastic increase in time. Also the different kind of improvements make the communication

infrastructure more scalable and let the star network work well until there are more then 5000 agents (which is enough for a conference with 1000 members).

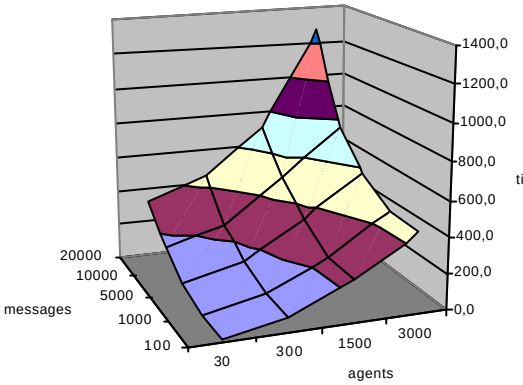

Figure 4-17 original COMRIS version

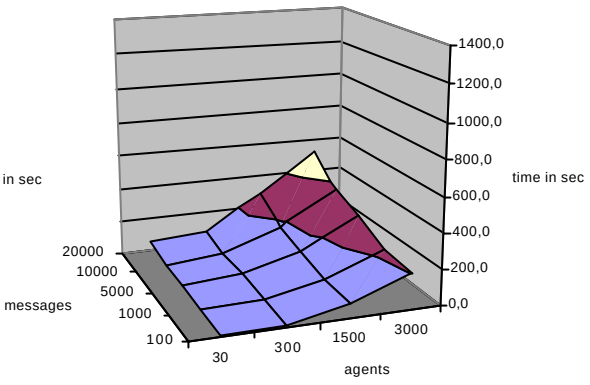

Figure 4-18 COMRIS version with all updates

The key question after all improvements is always: “Was it useful?”. Figure 4-17 and Figure 4-18 shows the difference between the original and the last updated COMRIS version. It is clear to see the new version is 2 ½ faster than the original. Noticeable is that the trend is obviously lower, which means that the load does not increase so fast.

The kernel requires enough memory to keep a direct connection to every secretary in the infrastructure. Whilst the infrastructure represents a centralised model this is an important consideration, even though the secretaries are fairly independent of the kernel once direct connections are established. If such connection fails or if secretaries are allowed a maximum number of direct connections to free up their machine’s resources, then the kernel will still be required. Maybe the connection to the kernel could be an intermittent one, turned off when not required, freeing up the kernel’s resources.

The charts do not show a nice linear or exponential curve. This is caused by two main factors. First, the fact that the curve is a result of more than one (kernel and secretary pools) separated java virtual machine (JVM) processes on top of Windows NT, which is not a real-time system and has to divide and schedule its resources with other applications. In addition, the JVM can garbage collection calls enter into a memory during execution of the test. Second, and more important, is the fact that the implementations are thread based. This means that in some cases a large number of threads (the application itself, direct connection, etc.) can have to perform a lot of pre processing but are only able to produce their output result at the next activation cycle.

5 Conclusion

5.1 Achieved status

The biggest problem on this work was the time, due to the knowledge that the COMRIS project is near the end it is difficult to make elementary changes. Nevertheless, appropriate work was done and the scalability of the COMRIS communication infrastructure was increased.

The address multicast is noticeable right at the start of the infrastructure, thereby the whole registration process in the subspaces and publish groups is faster (see 4.6.3). The direct connections are perceivable, especially in the running system, because the messages find the specified destination without any detours. The third implemented idea, the special name concept is only useful for bigger infrastructures with many secretary pools and due to the shortened implementation, not so effective. The kernel itself is now able to handle a significantly higher load than at the beginning of this work. Therewith the whole system is able to handle higher load and the message delay time is reduced.

Altogether, the scalability of the communication infrastructure was increased by a factor of two and a half to three. There were also some methods implemented, which are necessary for the visualisation tool [10]. The test process discovered some errors at the programming of the infrastructure (some functions were used twice), which were removed immediately.

5.2 Commendation

Over time, it showed that some ideas are useful and some are not so useful for such an agent system. Therefore, it is adverse to use bus or tree topologies. They have a potential bottleneck and they have some implementation problems. At the end it always depends on the size of the agent system. For example, a small conference with 100 – 1000 members is not a problem for the star topology, especially not with some additional concepts like the direct connection and the address multicast. Should it be a bigger conference, it is necessary to have a more flexible infrastructure. In the star network the central node is always the bottleneck and, in addition, direct connections are not endlessly useful. Each direct connection takes the same memory and it needs time to search an existing direct connection. Thus it is better to use a more decentralized architecture, like the ring or the hypercube.

The work on the infrastructure has shown that it is easy to say "we enhance the scalability", but it is difficult to realise this aim. How scalable an infrastructure can be is decided in the design and with the specification. It is always difficult to implement changes in a sophisticated stadium. Another important conclusion is that not only the look of the infrastructure can enhance the scalability. This means that the complete system has to be well balanced. In addition to the infrastructure it is also important that, for example, the agents use techniques to reduce the traffic. So it is perhaps possible to implement message-filter algorithms to make the $O(n^2)$ transport problem to a O(n) transport problem.

5.3 Perspective

With the end of this work, the COMRIS project has already finished. Thus there is no further work to do in direct connectivity to COMRIS. However, there are some ideas that can be prosecuted in additional studies. This document tries to show the different design possibilities for the COMRIS infrastructure. Further work could try to test the different topologies and to prove that a hypercube is more scalable than a star or a ring.

Another concept, which was not described in this work, is load balancing for distributed architecture [16]. Load balancing collects system state information and assigns or redistributes the application tasks among the processors of a parallel computing system in order to maximise overall throughput and stabilise response times. However, it could be used also to assigns the traffic of a communication infrastructure like COMRIS. This can be performed either by a central component supervising the entire system, by cooperating pre-processor load balancing agents or by cooperation of load balancing agents, each of them controlling a part of the processing system.

It is also imaginable that the traffic through the infrastructure is finding its way with a special routing strategy. For example, if a connection between two secretary pools overloaded, it needs more time for sending a message between two agents in this pools. A special algorithm could analyse the traffic and could bypass new messages over another path. This path maybe longer, but free and thus the message can reach the destination earlier.

BIBLIOGRAPHY

[1] *"Computer Networks: A system approach"*, Larry L. Peterson & Bruce S. Davie; 600 pages 2nd Ed (1 November, 1999) Morgan Kaufmann, ISBN: 1558605770

[2] "*COMRIS: Virtual space definition and implementation plan"*, unpublished internal document

[3] Starlab (Belgium) checked 21.03.2001
http://comris.starlab.org/

[4] Webopedia checked 21.03.2001
http://webopedia.internet.com/TERM/s/scalable.html

[5] Computational Science Education Project;
http://csep1.phy.ornl.gov/csep.html checked 21.03.2001

[6] *"Reliable broadcast in hypercube computers"*, P. Ramanathan and K. G. Shin; IEEE Trans. Comput., vol. C-37, no. 12, pp. 1654–1656, Dec. 1988.

[7] *"Optimum broadcasting and personalized communications in hypercubes"*, S. L. Johnson and C. T. Ho; IEEE Trans. Comput., vol. C-38, no. 9, pp. 1249–1268, Sept. 1989.

[8] *"RING: A Client-Server System for Multi-User Virtual Environments"*, Thomas A. Funkhouser; Proceedings of the 1995 symposium on Interactive 3D graphics, 1995, Page 85

[9] *"High Performance Infrastructure for Visually-intensive CSCW Applications"*, Stephen Zabele, Steven L. Rohall and Ralph L. Vinciguerra; Proceedings of the conference on Computer supported cooperative work, 1994, Pages 395 - 403

[10] *"Virtual Reality Visualization of a Communication Infrastructure and Large Scale Agent Interaction"*, Robin Wolff unpublished Dissertation, Jan 2001

[11] *"Rendezvous: An Architecture for Synchronous Multiuser Applications"*, John F. Patterson, Ralph D. Hill, Steven L. Rohall and Scott W. Meeks; Proceedings of the conference on Computer-supported cooperative work, 1990, Pages 317 - 328

[12] *"The Rendezvous Language Architecture"*, Ralph D. Hill, Tom Brinck, John F. Patterson, Steven L. Rohall and Wayne T. Wilner; Commun. ACM 36, 1 (Jan. 1993), Pages 63 - 67

[13] *"The Abstraction-link View Paradigm, Using Constraints to Connect User Interfaces to Applications"*, Ralph D. Hill; Conference proceedings on Human factors in computing systems, 1992, Pages 335 - 342

[14] *"A Collaborative Medium for the Support of Conversational Props"*, Tom Brinck and Louis M. Gomez; Conference proceedings on Computer-supported cooperative work, 1992, Pages 171 - 178

[15] *"MMConfi An Infrastructure For Building Multimedia Applications"*, Terrence Crowley, Paul Milazzo, Ellie Baker, Harry Forsdick and Raymond Tomlinson; Proceedings of the conference on Computer-supported cooperative work, 1990, Pages 329 - 342

[16] *"Scalability and Potential for Optimization in Dynamic Load Balancing - Centralized and Distributed Structures"*, Wolfgang Becker; http://www.informatik.uni-stuttgart.de/cgi-bin//makehtml-ncstrl.cgi?document=TR-1992-01 checked 21.03.2001

[17] *"Distributed Peer-to-Peer Control for Harness"*, Christian Engelmann, unpublished Dissertation, Jan 2001

Declaration of authorship

I hereby declare that this submission is my own work and that, to the best of my knowledge and belief, it contains no material previously published or written by another person nor material which to a substantial extent has been accepted for the award of any other degree or diploma of the university, except where due acknowledgement has been made in the text.

Erklärung zur Urheberschaft

Ich versichere hiermit, dass ich die vorliegende Arbeit selbständig und nur unter Verwendung der angegebenen Literatur bzw. Hilfsmittel ohne fremde Hilfe angefertigt habe.

Die Arbeit wurde bisher in gleicher oder ähnlicher Form keiner anderen Prüfungsbehörde vorgelegt.

Reading (UK), den 27.März 2001

(Oliver Otto)