ELSEVIER

Advances in Engineering Software 31 (2000) 717–722

ADVANCES IN ENGINEERING SOFTWARE

www.elsevier.com/locate/advengsoft

A Windows-NT virtual collaboratory for technical computing[☆]

Dhruv Khettry[a], Xian-He Sun[a,b,*]

[a]Department of Computer Science, Louisiana State University, Baton Rouge, LA 70803-4020, USA
[b]Department of Computer Science, Illinois Institute of Technology, Chicago, IL 60616, USA

Abstract

Virtual Collaborative Environment (VCE) addresses enabling technologies to support collaborative, distributed, computer-based problem solving for engineering applications. The goal is to simulate a virtual laboratory, based on cooperative computing substrate, that integrally supports a shared workspace, high performance computing, distributed data management, as well as graphical scientific communication. In this study we present the design and development of a Windows-NT based VCE for technical computing. © 2000 Elsevier Science Ltd. All rights reserved.

Keywords: Virtual collaboratory; Collaborative computing; IP multicast

1. Introduction

A Window-NT Technical (computing virtual) Collaborative Environment (WTCE) has been developed enabling technologies to support collaborative, distributed, computer-based problem solving for engineering applications. WTCE simulates a virtual laboratory for cooperative computing work for engineering and scientific applications. It enables researchers, educators and industrial developers to work in conjunction with each other by accessing and sharing common resources and by collaborating over the Internet. This WTCE environment is for Windows based machines and the entire development is done using C/C++. It has been designed for supporting high performance compute intense applications. C/C++ applications provide excellent support for high performance applications and hence have been chosen for the WTCE project.

WTCE is a suite of software systems, communications protocols, and tools that enable computer-based cooperative work. WTCE constructs a virtual work environment on multiple computer systems connected over the Internet, to form a collaboratory. In this setting, participants interact with each other, simultaneously access and operate computer applications, refer to global data repositories or archives, collectively create and manipulate documents, perform computational transformations, and conduct a number of other activities via telepresence. WTCE is built upon several existing software systems. It combines the SNOW (Scalable Network of workstations) environment [1–3] and collaborative computing. SNOW delivers the high performance computing power via parallel computation and process migration. WTCE is also combined with SCALA (performance and SCALability Analyzer) for performance optimization [6,7]. The collaboration interface of WTCE is designed based on Collaborative Computing Frameworks (CCF) which is an environment on UNIX, developed at the Emory University [4,5]. CCF uses multicasting, session level protocols, atomic and consistent message passing schemes. All these ideas have been extended to WTCE. The APIs used for multicasting in WTCE are significantly different from the one used in CCF. CCF uses UNIX Berkeley sockets while WTCE uses Windows sockets. Other differences between WTCE and CCF have been discussed in the other sections of the paper. One of the main concepts implemented in the WTCE project is IP multicast and hence it is discussed in detail throughout this paper.

2. Related projects

Some of the other projects implementing collaborative environments include the Mbone tools, ISIS/Horus [8], and RMP [9]. The advances in network based parallel computing in projects like SNOW [3] extends such collaborative computing models. TeamWave is a groupware system that combines synchronous groupware technologies such as whiteboards and chat sessions. However TeamWave

[☆] This work was supported in part by National Science Foundation under grant ASC-9720215 and CCR-9972251.

[*] Corresponding author. Department of Computer Science, Illinois Institute of Technology, Chicago, IL, USA.

E-mail addresses: dhruv@csc.lsu.edu (D. Khettry), sun@cs.iit.edu (X.-H. Sun).

is a closed system that is based on the GroupKit [12] groupware toolkit, and only applications developed for GroupKit can be used with TeamWave. TeamWave is used for simple applications that are easy to develop and require facilities like text windows and simple graphical interfaces. Another class of collaborative systems is those based on the World Wide Web. Tango [11] is a prototype of an open system that provides mechanisms to rapidly integrate applications into a multiuser collaborative environment on the web. Tango uses a client–server architecture and uses a central server to control and distribute information among the clients. Local daemons on each participating machine intercept application traffic and distribute them via server and browser plugins. This tool consists of a session manager that controls the operation of various tools within a web browser and includes audio and video conferencing, televiewers and electronic instrument control. Netscape's Conference and Microsoft Netmeeting can also be considered as collaborative systems, but they are limited in their functions with regard to computation and shared data space support.

Various projects address a subset of the goals defined by WTCE. It combines collaborative computing with high performance computing, performance evaluation and optimization, and process migration. It is developed over other existing software and incorporates the features of other software too. WTCE provides all elements of collaboration and is based on a distributed architecture. One of its main advantages over other traditional methods is that it is capable of supporting existing as well as collaboration-aware applications. WTCE provides insight into the effectiveness of the benefits of scientific computing. Various distinguishing features of WTCE over other existing systems include domain specific focus, open interfaces and security. CCF had problems in processing 'channel leave' messages. Dynamic maintenance of channel membership was a problem and has been overcome in WTCE.

3. WTCE architecture

WTCE operates over standard Internet protocols, but builds its own fast and efficient multiway protocols thereupon—these core modules are identified as VCTL (Virtual Collaboration Transport Layer) and VCE-API. VCTL is a communication layer consisting of a suite of multiparty protocols, providing varying service qualities among process groups. VCTL explicitly supports distributed collaborative and multimedia applications. VCTL is based on a two-level group hierarchy. Logical interconnections among entities, called channels, define an efficient and light weight group mechanism. Channels support a variety of service qualities such as reliability and message ordering. Related channels can also be combined to form sessions, heavy weight groups which provide a default atomic multicast service. VCTL supports membership protocols tailored to the quality of service offered by a channel. The VCTL group module consists of channel membership and QoS submodules. The channel membership module enforces ordering and reliability guarantees for regular messages. The QoS module also provides an interface to lower level network protocols such as IP multicast or UDP and handles inter-network routing (IP-multicast to a LAN, UDP over WANs). The VCTL group caters to distributed and collaborative applications. The protocol stack of the reliable channel is implemented through three threads: sender, receiver and acker. The sender thread takes messages from the send queue, and fragments them into packets. The fragments are transmitted using the VCTL routing protocol. VCTL adopts a TCP-like flow and congestion control. The receiver thread acquires packets through a multicast socket and reassembles fragments into messages. Based on the channel ID contained in a message, it multiplexes the message into the receive queues of channels. It sends acknowledgements and passes them to the sender thread which adjusts its flow and congestion windows based on these acknowledgements. VCTL is more efficient when there are more clients because TCP uses unicast whereas VCTL uses multicast to transfer messages. A hierarchical architecture is used for WTCE. The lowest layer of WTCE is the physical network and over that UDP and IP multicast is implemented. The VCTL library implements multiway transport protocols supporting different QoS. This library provides the collaborative environment necessary for cooperative work. This library is used by the higher level of WTCE. An API is developed over VCTL to encapsulate all the low-level network functions. This API is called VCE-API and is responsible for running a session name server and maintains all the channel membership information. This layer is responsible for maintaining the multicast group and it sends queries to other members in the group for dynamically detecting group members. Applications are the highest layer in the WTCE architecture. Some VCE tools have been implemented. Other existing applications can use the VCE-API and perform collaborative work across the network. Fig. 1 shows the overall architecture of WTCE.

The entire communication layer (VCTL) has been designed and developed as a library consisting of different types of protocols supporting varying QoS. The design of the VCTL library is similar to the one used for CCF [5]. Applications have been developed to support WTCE and they just need to link with the VCTL library to be able to support collaborative computing. In the current version of WTCE, a chat tool has being developed. Due to the modular design, tools like audio, clearboard, and other multimedia tools can be developed over VCTL to support collaborative computing. Fig. 2 shows the architecture of VCTL. VCTL is the main component of WTCE and hence has been discussed on detail in this paper. A hierarchical design has also been used for VCTL. The channel membership module is the main module inside VCTL. This runs over the physical network and the application modules run over it. When an application comes up, it sends a join message to the

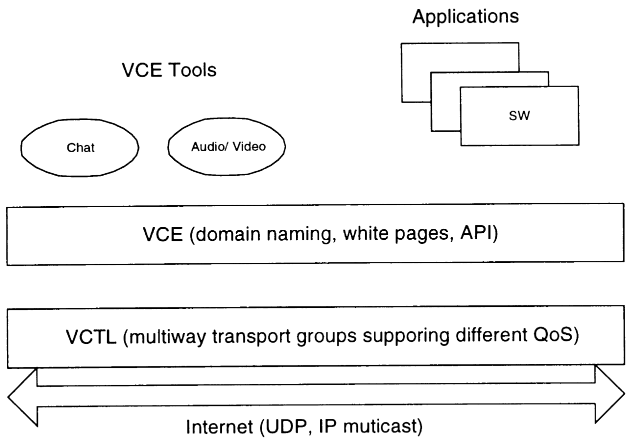
Applications
VCE Tools
Chat
Audio/ Video
SW
VCE (domain naming, white pages, API)
VCTL (multiway transport groups supporing different QoS)
Internet (UDP, IP muticast)

Fig. 1. WTCE system architecture.

channel membership (CM) module on its node. It uses the channel communication (COM) module to inform all the other members in its group. The COM module either uses UDP or IP multicast to inform all the other members in the group about this new member. Similarly a session module is also implemented and it is responsible for the name service and reliable delivery over the network. Fig. 2 shows how messages are transmitted from one module to another.

VCTL is logically implemented as a group module, interposed between applications (clients) and the physical network. This module implements the VCTL API and provides session and channel abstractions to clients. The group module consists of channel membership (CM), communication (COM), and session sub-modules. The session module supports the services that a session provides. The CM module implements membership protocols for channels using the reliable message delivery service of the session module. The COM module implements data communication services of channels. The session module implements a virtually synchronous reliable multicast group that provides group management services, such as virtual synchronous membership, failure detection, and authorization. Applications can join and leave a session through the API of the session module, and the CM module communicates with the CM modules of other session members through the session module. The session module notifies the CM module of session membership updates including member failures. A process must join a session to participate in a collaboration session. Every channel created after session join becomes a child group. The CM module implements membership protocols for different types of channels. It uses the session module to communicate with the CM modules of other session members, and uses the COM module to enforce virtual synchrony for the channels that require virtual synchrony (e.g. reliable channels). If a channel needs virtual synchrony, the application module uses the API of the CM module to receive and send messages. A VCTL channel currently supports one of three data communication services: total-ordering reliable, FIFO reliable, and unreliable. These services are implemented within the COM module.

The COM module implements a protocol stack for each channel type. Multiple channels of the same type may share a protocol stack or they may create their own. Channel ID is included in messages sent through the protocol stack and the COM module multiplexes messages to the appropriate channel based on this ID. The COM module discards messages not intended for the current process. Sockets are allocated on a per-stack basis so sharing protocol stacks saves descriptors.

Each protocol stack uses bounded IP multicast. If a channel contains members on two or more isolated subnets, messages are sent to the remote subnet via a UDP tunnel and then multicast locally. Fig. 3 illustrates VCTL's routing mechanism. A limited form of pruning is implemented. When a packet does not contain a member in a remote subnet, its stack does not send the packet to the subnet.

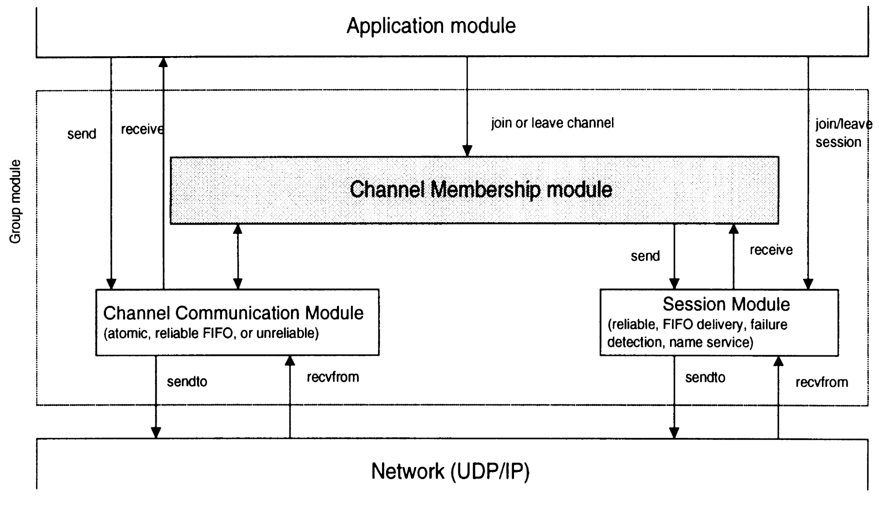
Application module
Group module
send
receive
join or leave channel
join/leave session
Channel Membership module
send
receive
Channel Communication Module
(atomic, reliable FIFO, or unreliable)
Session Module
(reliable, FIFO delivery, failure detection, name service)
sendto
recvfrom
sendto
recvfrom
Network (UDP/IP)

Fig. 2. VCTL architecture.

This technique reduces interference between channels sharing a stack but with distinct memberships.

One of the main features in the implementation of VCTL is the use of IP multicast. IP multicast is widely used for collaborative computing applications where sessions can be dynamically added to the multicast group. Researchers can collaborate with each other by sending multicast packets to all members connected over a session. This is far more efficient than requiring the source to send an individual copy of a message to each requester (point-to-point unicast), in which case the number of receivers is limited by the bandwidth available to the sender. It is also more efficient than broadcasting one copy of the message to all nodes (broadcast) on the network, since many nodes may not want the message, and because broadcasts are limited to a single subnet. The concepts of IP multicasting have been discussed in detail because this is one of the main components of the WTCE project.

IP multicast uses Class D Internet Protocol addresses, with 1110 as their high-order four bits, to specify multicast host groups. In Internet standard "dotted decimal" notation, host group addresses range from 224.0.0.0 to 239.255.255.255 for multicasting. The range of addresses between 224.0.0.0 and 224.0.0.255 is reserved for routing protocols and other low-level topology discovery or maintenance protocols. Other addresses and ranges have been reserved for applications, such as 224.0.13.000 to 224.0.13.255 for Net News. All multicast addresses in that range that are not reserved can be used by user applications for multicasting.

To send an IP multicast datagram, the WTCE sender specifies an appropriate destination address, which represents a host group. IP multicast datagrams are sent using the same "Send IP" operation used for unicast datagrams. Compared to sending of IP multicast datagrams, reception of IP multicast datagrams is much more complex, particularly over a WAN. To receive datagrams, a user's host application requests membership in the multicast host group associated with a particular multicast. This membership request is communicated to the LAN router and, if necessary, on to intermediate routers between the sender and the receiver. As another consequence of its group membership request, the receiving host's network interface card starts filtering for the LAN-specific hardware (data-link layer) address associated with the new multicast group address. WAN routers deliver the requested incoming multicast datagrams to the LAN router, which maps the host group address to its associated hardware address and builds the message (for example, an Ethernet frame) using this address. The receiving host's network interface card and network driver, listening for these addresses, pass the multicast messages to the TCP/IP protocol stack, which makes them available as input to the user's application, such as a video viewer or other collaborative applications.

Whereas an IP unicast address is statically bound to a single local network interface on a single IP network, an IP host group address is dynamically bound to a set of local network interfaces on a set of IP networks. An IP host group address is not bound to a set of IP unicast addresses. Multicast routers do not need to know the list of member hosts for each group—only the groups for which there is one member on the subnetwork. A multicast router attached to an Ethernet need associate only a single Ethernet multicast address with each host group having a local member.

Each IP multicast packet uses the time-to-live (TTL) field of the IP header as a scope-limiting parameter. The TTL field controls the number of hops that an IP multicast packet is allowed to propagate. Each time a router forwards a packet, its TTL is decremented. A multicast packet whose TTL has expired (is 0) is dropped, without an error notification to the sender. This mechanism prevents messages from needless transmission to regions of the worldwide Internet that lie beyond the subnets containing the multicast group members.

A local network multicast reaches all immediately-neighboring members of the destination host group (the IP TTL is 1 by default). If a multicast datagram has a TTL greater than 1, the multicast router(s) attached to the local network take

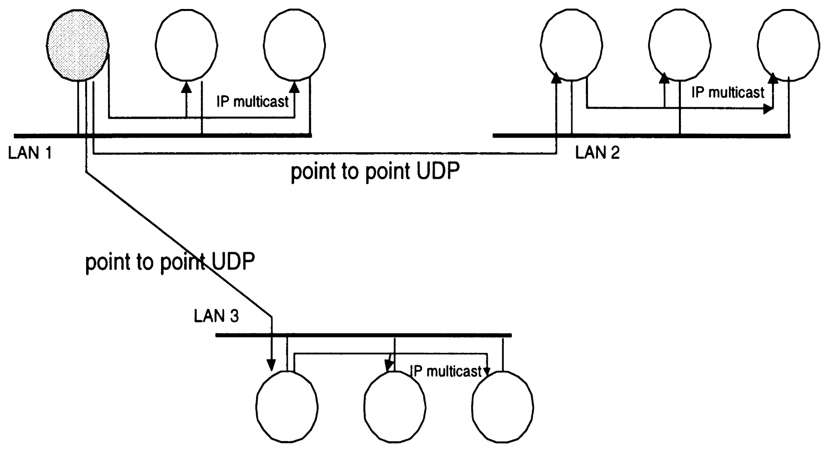

Fig. 3. Routing in VCTL.

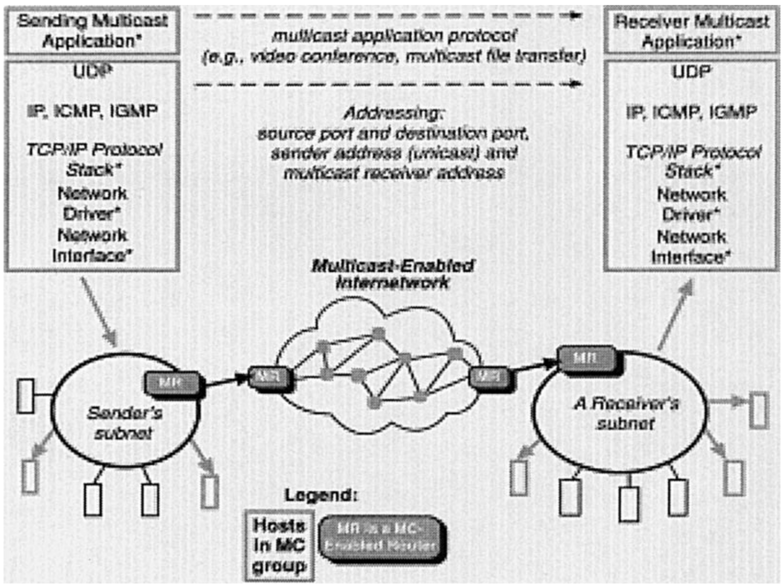
Sending Multicast Application*
UDP
IP, ICMP, IGMP
TCP/IP Protocol Stack*
Network Driver*
Network Interface*
multicast application protocol (e.g., video conference, multicast file transfer)
Addressing: source port and destination port, sender address (unicast) and multicast receiver address
Receiver Multicast Application*
Multicast-Enabled Internetwork
MR
Sender's subnet
A Receiver's subnet
Legend:
Hosts in MC group
MR is a MC-Enabled Router

Fig. 4. Multicast-enabled components.

responsibility for internetwork forwarding. The datagram is forwarded to other networks that have members of the destination group. On those other member networks that are reachable within the IP time-to-live, an attached multicast router completes delivery by transmitting the datagram as a local multicast. TTL thresholds in multicast routers prevent datagrams with less than a certain TTL from traversing certain subnets. This can provide a convenient mechanism for confining multicast traffic to within campus or enterprise networks. Several standard settings for TTL are used for multicasting: 1 for local net, 15 for site, 63 for region and 127 for world [10].

4. Implementation

WTCE has been implemented for Windows platform. It has been tested on Windows NT v4.0. Microsoft Foundation Classes (MFC) have been extensively used to implement all user interfaces. The applications are ‘Dialog based applications’ supporting MFC. The software is implemented in C/C++ using the Visual C++ environment. The VCTL library is built over Windows Sockets version 2 (WinSock 2). It provides an API for communication over the network using UDP, TCP and IP multicasting. WTCE uses Windows socket multicast APIs. WinSock 2 has been used. Multicasting is also possible on a WinSock 1.1 implementation but it should support BSD multicast APIs. These APIs are used for the following:

- to join a multicast group;
- to leave a multicast group;
- to set the IP time to live (TTL) on a multicast datagram to adjust the scope;
- to set the local interface for multicast transmission and receipt;
- to disable loopback of outgoing multicast datagrams.

5. Requirements and challenges

To support native IP multicast, the sending and receiving WTCE nodes and network infrastructure between them must be multicast-enabled, including intermediate routers. Requirements for native IP multicast at the end node hosts are [10]:

- support for IP multicast transmission and reception in the TCP/IP protocol stack;
- software supporting Internet Group Management Protocol (IGMP) to communicate requests to join a multicast group(s) and receive multicast traffic;
- network interface cards which efficiently filter for LAN data link layer addresses mapped from network layer IP multicast addresses;
- IP multicast application software such as video conferencing, collaborative computing applications etc.

The above are some of the pre-requisites for running WTCE. To run or evaluate IP multicast on a LAN, only the above are needed; no routers need to be involved for a host’s adapter to create or join a multicast group and share multicast data with other hosts on that LAN segment. For IP multicast over a WAN, traffic needs to be expanded as follows:

- All intermediate routers between the sender(s) and receiver(s) must be IP multicast-capable. Many new routers have support for IP multicast; older ones may require memory before they can be upgraded.
- Firewalls may need to be reconfigured to permit IP multicast traffic.

Fig. 4 depicts the components that must be multicast-enabled. The direction of traffic shown is for multicast datagrams.

In Fig. 4, different forms of multicasting have been illustrated. It shows how multicasting can be implemented across a WAN as well as on a LAN. Within a subnet, it is relatively simple to implement multicasting but across the Internet, it becomes subsequently difficult. Routing over the Internet is much more complicated and it employs IP, IGMP or ICMP. This transfer is performed at different layers. TCP/IP is the highest layer followed by network driver and the network interface. Communication is through UDP or through IP multicast over the internetwork. Multicasting over a WAN is a challenge as lots of networks do not provide support for multicasting. Security features may also be turned on by a network thus not allowing other networks to communicate with it. These are significant challenges for WTCE. Connections to WTCE are not fully secure. Significant steps might be taken in the future to provide secure connections when transferring data to and from insecure networks.

6. Conclusion

WTCE environment was thoroughly tested on Windows platforms for applications collaborating on the same network as well as on networks with different subnets. Certain performance evaluation is yet to be done for the WTCE system. It works very efficiently with increasing number of members in a session.

Current implementation of WTCE is designed for Windows based machines only. CCF had been implemented for UNIX based architecture. WTCE can be combined with CCF to support collaborative computing across UNIX and Windows architecture. As an improvement, heterogeneous platforms could be supported by making use of platform independent features of Java. Currently heterogeneity has been sacrificed because of the poor performance of Java on computation intensive applications. Java is currently in its rudimentary stage and its status as a scientific programming language has been continuously under question due to its poor performance on computationally intense applications. As an enhancement, a Java 3D browser interface for WTCE could be designed. Java would then be adopted for visualization and telepresence, C/FORTRAN will be used for computing, and SNOW will serve as a computing layer that facilitates parallel processing under MPI/PVM interface.

References

[1] Chanchio K, Sun X. MpPVM: a software system for non-dedicated heterogeneous computing. Proceedings of the International Conference on Parallel Processing 1996:III;215–22.

[2] Leutenegger S, Sun X. Limitations of cycle stealing of parallel processing on a network of homogeneous workstations. Journal of Parallel and Distributed Computing 1997:169–78.

[3] Sun X, Naik V, Chanchio K. A coordinated approach for process migration in heterogeneous environments. Proceedings of the Ninth SIAM Conference on Parallel Processing for Scientific Computing 1999.

[4] Sunderam V, et al. Collaborative computing frameworks for natural sciences research. http://emily.mathcs.emory.edu/ccf/, 1997.

[5] Rhee I, et al. Group communication support for distributed collaboration systems. ICDCS Proceedings, 1997.

[6] Sun X-H, Zhu J. Performance prediction: a case study using a scalable shared-virtual-memory machine. IEEE Parallel and Distributed Technology 1996:36–49.

[7] Sun X, Pantano M, Fahringer T. Integrated range comparison for data-parallel compilation systems. IEEE Transactions on Parallel and Distributed Systems 1999:448–58.

[8] Birman K, Joseph T. Reliable communication in the presence of failures. ACM Transactions on Computer Systems 1987;5(1):47–76.

[9] Whetten B, Montgomery T, Kaplan S. A high performance totally ordered multicast protocol. Proceedings of the Dagstuhl Distributed Systems Workshop 1994:33–57.

[10] Johnson V, Johnson M. Implementing IP multicast in different network architectures, www.ipmulticast.com.

[11] Beca L, et al. TANGO—a collaborative environment for the World Wide Web. http://trurl.npac.syr.edu/tango/, Syracuse University, 1996.

[12] Roseman M, Greenberg S. Building real time groupware with GroupKit, a groupware ToolKit. ACM Transactions on CHI 1996;3(1):66–106.