Computing and Informatics, Vol. 24, 2005, 67–83

COLLABORATIVE VIRTUAL ENVIRONMENT FOR ADVANCED COMPUTING[*]

Gareth J. Lewis, S. Mehmood Hasan, Vassil N. Alexandrov

Advanced Computing and Emerging Technologies Centre
School of Systems Engineering
University of Reading
Whiteknights, P. O. Box 225
Reading, RG6 6AY
United Kingdom
e-mail: {g.j.lewis, s.m.hasan, v.n.alexandrov}@rdg.ac.uk

Martin T. Dove, Mark Calleja

Department of Earth Sciences
University of Cambridge
Downing Street
Cambridge, CB2 3EQ
United Kingdom
e-mail: {martin, mcal00}@esc.cam.ac.uk

Manuscript received 9 December 2004

Abstract. Synchronous collaborative systems allow geographically distributed participants to form a virtual work environment enabling cooperation between peers and enriching the human interaction. The technology facilitating this interaction has been studied for several years and various solutions can be found at present. In this paper, we discuss our experiences with one such widely adopted technology, namely the Access Grid. We describe our experiences with using this technology, identify key problem areas and propose our solution to tackle these issues appropriately. Moreover, we propose the integration of Access Grid with an Application

[*] Original title "Building Collaborative Environments for Advanced Computing", published in Proceeding of the ISCA 17th International Conference on Parallel and Distributed Computing Systems, Pages 497–502, San Francisco, California, 15–17 Sep. 2004.

Sharing tool, developed by the authors. Our approach allows these integrated tools to utilise the enhanced features provided by our underlying dynamic transport layer.

Keywords: Collaborative computing, application sharing, access grid, multicast

1 INTRODUCTION

Collaborative computing systems aim to complement human face-to-face communication by providing various tools which enhance users' experience. The emergence of multicast triggered a rush in the development of group communication software. Collaborative tools have also become popular with the widespread availability of broadband. The Access Grid [1] has become well known for high quality group-to-group collaboration across the Internet. It has been widely adopted by the academic community and uses multicast for streaming audio and video. Unfortunately, Multicast is still not available to many institutions as well as home users (due to the reluctance of the Internet Service Providers (ISPs) in adopting this technology).

The Access Grid has now become available for a single desktop via tools known as the Personal Interface to Access Grid (PIG) [2] (for Microsoft Windows Users) and PIGLET [3] (for Linux users). PIG/PIGLET tools are used by the eMinerals [4] project group to interact with geographically distributed members. Several problems have arisen as these tools use multicast, which is not widely supported. These issues are discussed in detail later in the paper.

Our work in the field of collaborative computing started with research into Collaborative Computing Frameworks (CCF) [9], which was developed at Emory University, Atlanta, USA in collaboration with the University of Reading, UK. CCF is a suite of software systems, communications protocols, and tools that enable computer-based cooperative work. It constructs a virtual work environment on multiple computer systems connected over the Internet. CCF facilitates sharing of applications and resources. With the experience gained from CCF, we have been able to carry forward our research and address the deficiencies found in the CCF system.

Person-to-person communication is enriched by an ability to share, modify, or collaboratively create data and information. Our aim at The University of Reading is to provide an Application Sharing tool which allows effortless sharing of legacy applications. The Application Sharing tool is developed by the authors, specifically to be used in a group communication environment. This tool will be integrated with the PIG/PIGLET to provide enhanced functionality. We are also interested in developing an inclusive collaborative system, allowing unicast participants to interact with multicast groups in a dynamically changing environment.

The rest of the paper is organised as follows. Section 2 describes our experience with the PIG/PIGLET tools and outlines major problem areas. Section 3 presents the Application Sharing tool developed at The University of Reading. In Section 4, we illustrate the design of the proposed dynamic transport system and look at the integration issues. We conclude with a road map for our future work in Section 5.

2 THE ACCESS GRID

The Access Grid is an advanced collaborative environment, which is used for group-to-group collaboration. A suite is used for the Access Grid to ensure effective collaboration. A typical Access Grid suite consists of several machines for the audio, video and for the display. The video streams are displayed through several high quality projectors and specialist sound equipment to enhance the sound quality and to reduce the echo. Multicast is used for transport of the video and audio data to multiple hosts. The Access Grid uses the concept of Virtual Venues (VV) to allow groups with similar interests to interact; an example of this is The University of Reading VV. Each of the VV's has a unique multicast address and port over which streams the video and the audio. The VV uses different ports and sometimes different multicast addresses to distinguish between the audio and video streams. This provides flexibility for the user to decide whether to receive audio, video or both. Unicasting and broadcasting provide extremes in addressing – unicast uses a single IP address and port as an endpoint, and broadcasting propagates to all IP addresses on a subnet. Multicast provides an intermediate solution, allowing a set of IP addresses to be identified and ensures that datagrams are only received by interested participants. The Access Grid comprises of several separate applications, of which the main tools are Robust Audio Tool (RAT) [6] for audio and a Video Conferencing Tool (VIC) [5] for video. There are many Access Grids nodes in the UK, but within a project there are often participants who do not have access to a suite. Even if a suite is available for some of the project participants, it is likely that they will have to book specific times for use and ensure that there is a trained operator available.

2.1 PIG/PIGLET

An alternative to installing an expensive Access Grid suite is to use the Personal Interface to the Grid (PIG). PIG can be used on both Microsoft Windows and Linux operating system (PIGLET). As with the Access Grid suite the two main tools are VIC and RAT. Rather than have separate machines for the video and audio, both the tools run on a single desktop computer. Using PIG allows distributed project members to participate in regular Access Grid meetings without the inconvenience of regularly booking, and traveling to an Access Grid suite. As in an Access Grid suite, multicast is used to send audio and video data to other participants. If multicast is not enabled on the local network, then it is possible to setup a multicast bridge to allow unicast participants to join in meetings.

Multicasting solves some of the issues involved with group communication by reducing the bandwidth required in connecting multiple hosts. This becomes particularly prevalent when considering the transport of multimedia traffic. Although using multicast for group communication has advantages over simply using UDP, there are some serious problems associated with multicast. These problems have been observed first hand at The University of Reading and include:

Routing Multicast Packets – In IPv4 the multicast addresses are known as the Class D addresses and range from 224.0.0.0 to 239.255.255.255. Conceptually multicast works by informing the kernel that the machine is interested in a particular multicast address. The host machine then sends an IGMP message to any associated router – this informs the router to pass datagrams related to that group. The routers communicate between each other to propagate the message that a host is interested in a particular group. To achieve the routing of multicast packets, the routers must be multicast enabled. Currently multicast support is not ubiquitous and therefore multicast is not viable for a large number of hosts.

Setting up a Multicast Bridge – Due to the problems faced with enabling multicast within existing subnets, it is possible to use a bridge machine connected to the M-Bone. The bridge joins the appropriate multicast group and unicasts datagrams to the connected participants on a different subnet. This has been used by many institutions as an alternative to enabling multicast; an example of this is the Cambridge eMinerals bridge which has been used extensively in the eMinerals project. The Reading bridge has been used successfully to attend meetings, but we have recognised that the performance of the tools diminishes as there is an increased load on the bridge.

Multicast and Firewalls – As when using UDP, problems arise when attempting to multicast between hosts separated by a firewall. The only course of action is to open certain ports for certain multicast addresses to allow traffic through. An added consideration with multicast is allowing IGMP packets to be sent through the firewall, this is essential to inform routers that a host wishes to join a certain multicast group.

Facilitation of Dynamic Meetings – One of the limitations with the PIG is the lack of functionality for dynamic meetings. The PIG works effectively for scheduled meetings, but there is no consideration to dynamic meetings between hosts. In other collaborative tools, such as instant messengers, a host is informed when his "buddies" are online and accessible for communicating.

3 APPLICATION SHARING

Video and Audio are essential for an effective collaborative experience, allowing participants to mimic natural human interaction. Another major component involves the sharing of material between participants. This includes activities, such as collaborative viewing of a representation of data, document editing by various colleagues, collaborative code development between geographically distributed peers, etc. Application sharing is the ability to share and manipulate desktop applications between multiple participants.

There are two general approaches to developing synchronous collaborative applications. First approach, collaboration transparency, facilitates application sharing by employing a mechanism that is unknown, or “transparent”, to the application and its developers. In this case, one is able to share an arbitrary single-user legacy application. The second approach, referred to as collaboration awareness, allows applications to be specifically designed to support cooperative work between multiple users.

Application sharing systems can be generally divided into two categories in terms of their architecture; centralised and replicated. An application sharing system with a centralised architecture implies that the application is shared in one location, and its graphical output is distributed to all the session participants. In this setting, the response time is comparatively slow and the amount of traffic considerably large. However, a centralised architecture is able to deal effectively with the late comer problem. The late comer problem arises when a participant joins when a collaborative session is already in progress. There are several mechanisms which can be used to deal with this problem. For example, all events can be recorded and replayed for the late comer, or current state of the application can be transferred to the late comer. There are advantages and disadvantages associated with both approaches and the appropriateness of the solution depends on the type of shared application.

Another issue that arises with a centralised architecture is floor control. Since the application is run centrally, simultaneous inputs cannot be allowed. There needs to be a mechanism whereby participants can take turns in editing a document or manipulating a certain representation of data. This is known as floor control.

In contrast to a centralised architecture, a replicated application sharing system requires that the same application and its execution environment be replicated at each site. Each replica executes locally and instead of distributing the graphical output only input events are transferred to all participating replicas. This approach allows for faster local response time and reduces network traffic as compared to a centralised architecture. However, fundamentally, the problem remains that the replicas of the transparently shared application may become inconsistent. If the application depends on the timing of an input or processor speed, replicas on different hosts may reach different states. While replicated collaboration-aware applications are designed to avoid timing or processor dependencies, single-user applications may contain such dependencies.

3.1 Related Work

The technology that facilitates collaborative work has been studied for many years and there are many products currently available. The relative merits of this technology have been evidenced by the success of commercial application sharing products such as the Microsoft NetMeeting [11] and SunForum [12]. Many research products and prototypes have also been developed, to name but a few, Dialogo [17], Flexible JAMM [16], MMConf [13], Rapport [18], SharedX [15], XTV [14] and VNC [7].

Systems such as Dialogo [17], Rapport [18], SharedX [15], XTV [14] are developed to share applications on X Windows platform. X Windows defines a network-aware graphical protocol which separates an application's display from its computation. The X Windows system uses a client server model in which the user's application, the X client, renders images on the user's display by sending messages to the X server, using TCP/IP. This separation yields a natural approach to implementing window sharing. SharedX [15] and XTV [14] adopt a centralised architecture, whereas Dialogo [17] is a replicated system. However, the main shortcoming of these systems is that they are developed to share application solely on X Windows platform.

Flexible JAMM [16] is able to support some level of independent view of the shared application and explores novel approaches, such as multi-user scroll bar and radar views. However, research in [16] is focused towards sharing only JAVA programs on different operating systems. This approach is clearly very restrictive since it does not provide collaborative support for a vast number of user applications.

Virtual Network Computing (VNC) [7] is currently one of the most popular tools used with the Access Grid in order to share applications during meeting. VNC shares the entire desktop with the participants in the group. However, it can be easily modified to share a specific application.

VNC has become the leading solution for desktop sharing. It is designed for point to point communication, consisting of a server (Xvnc) and a light-weight viewer (vncviewer). A participant wishing to share a particular application runs the application and allows the rest of the group to make individual TCP connection to his/her machine. This approach has several features that make it less suitable for group to group collaboration; these include:

- VNC is point to point – within the group (or Virtual Venue) one of the participants runs the VNC server to which all other participants connect. This is unsuitable for group communications since it does not present a scalable model.
- Multicast VNC [19] only allows participants to be viewers – The multicast plugin to VNC acts as a proxy to multicast the unicast traffic to other members of the group. This overcomes the problem of bandwidth, but adds the inconvenience that other participants can act only as viewers.
- Each participant must run the VNC server to share an application with the group. The other participants must have the IP address of the machine wishing to share.
- VNC is used for remote graphical login – sharing several applications between different participants becomes complicated. The user must also share the entire desktop rather than a single application.

3.2 Multicast Application Sharing Software

The approach to adding groupware features to single-user applications falls into either the collaboration-aware or the collaboration-transparent category. The former requires access to proprietary source code, which in practice may be impossible to acquire. Thus collaboration transparency appears a more promising alternative in many situations. By comparison, collaboration transparency allows for sharing single-user applications without modification to the source code.

In contrast to collaboration aware applications, conventional collaboration transparency systems, like Microsoft NetMeeting, are lacking in terms of efficient use of network resources and support for key groupware principles: concurrent work, relaxed What You See Is What I See (WYSIWIS), and group awareness.

Application sharing systems in general have either a centralised or a replicated architecture. The fundamental problem with replicated architecture is that the replicas of the transparently shared application may become inconsistent. The issue of synchronisation becomes more of a problem as the group size increases. Typically, there is no limit on the number of participants in an Access Grid session. Therefore, we decided to use the collaboration-transparency approach based on a centralised architecture, avoiding the synchronisation problems and facilitating sharing of an arbitrary single-user legacy application without access to its source code.

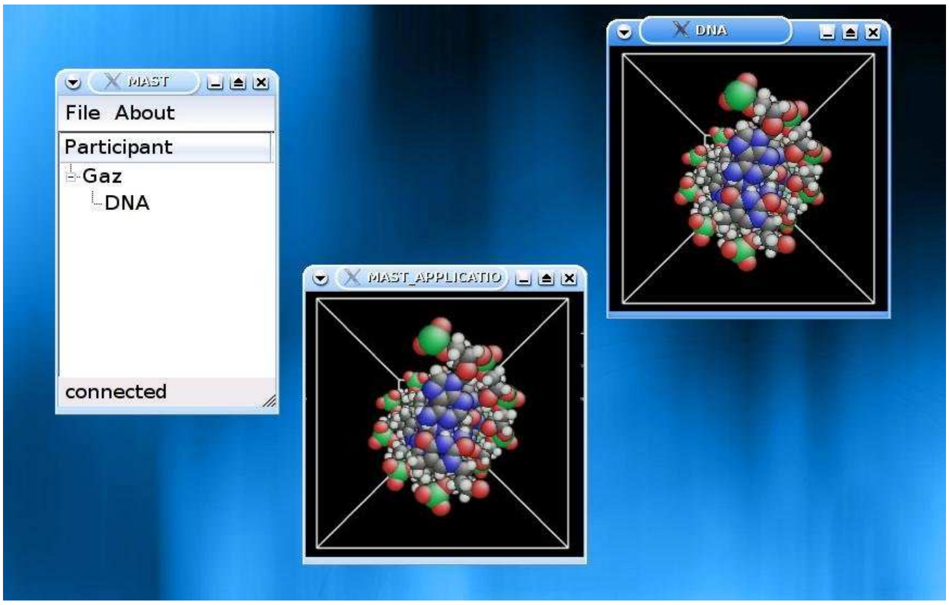


Fig. 1. Screen shot of multicast application sharing tool

VNC is traditionally used within the Access Grid to share applications. In the authors' opinion VNC is often utilised in a role it was not designed for. This led us to develop a new Multicast Application Sharing Tool (MAST). There are two versions of MAST, one for Microsoft Windows and the other for Linux, which can be used in conjunction with VIC and RAT to enhance the group-to-group collaborative experience. Figure 1 shows a screen shot of MAST being using to share an application under Linux. There are several factors that had to be considered when designing MAST:

Allows Multicasting and Unicasting – to be inline with VIC and RAT, MAST allows the data to be sent to the participants using multicast or unicast. If multicast is enabled on the network, then the application can send multicast packets, however if not multicast enabled, the application can be sent unicast to a multicast bridge.

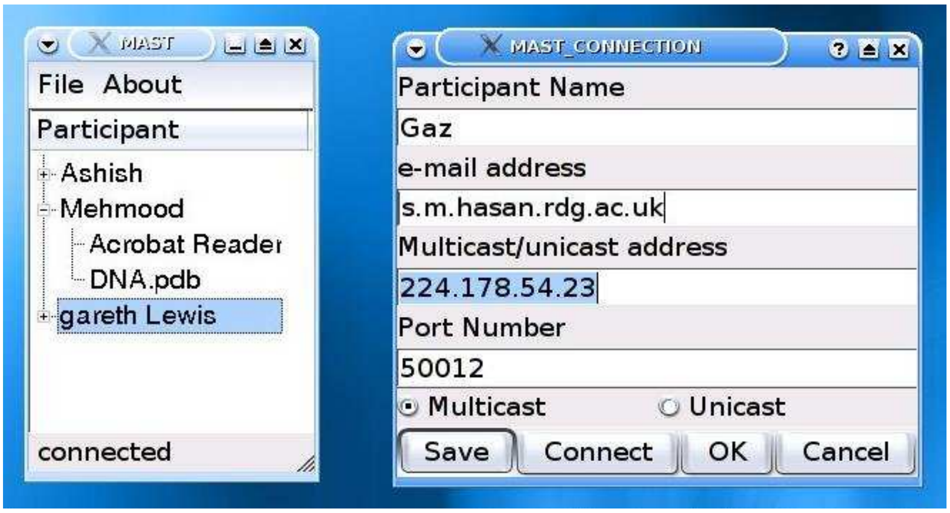
Simple Configuration – MAST has a settings menu (Figure 2) which allows a participant to select whether the data is sent multicast or unicast. The user can also add details about themselves to be seen by other participants.

Using a Single Multicast group – Access Grid has different Virtual Venues for different institutes, and within a VV each of the tools has a unique multicast group for streaming data. A single multicast group consists of a multicast address and a port number. A similar idea is used for MAST so that participants in a particular VV can share application using a single multicast group. MAST receives multiple streams, one for each of the shared applications, which must be sent over a single multicast group to be used successfully in group collaboration. MAST achieves this by uniquely identifying each of the application streams, and listing each application below the owner participant's name (Figure 2).

Reducing Screen wastage – Due to the lack of screen space when using PIG/PIGLET, we felt that it was important to reduce the screen area required by MAST. The GUI of MAST resembles that of many Instant Messengers, with a simple list of participants, that can be expanded to show all the applications currently being shared by a participant. The user can enlarge a particular application to its normal size within the sharing window. There is only one sharing window, as it was felt that having multiple sharing windows would cause wastage of valuable screen space.

It is important that the application data is transferred over the same multicast group. To achieve this, each application stream must be distinguished by a unique identifier, so that once received it can be displayed in the correct window. The application sharing will be initially restricted to having a single master and multiple viewers, where the viewers will be unable to control the application.

It is important that the application sharing allows for interaction between the various participants. Floor control becomes an extremely important consideration when designing the application sharing tool. There are several possibilities, the first is that floor control is done by deciding in the group who is going to manipulate the document. The application has no restrictions on who does what and when. The next possibility is to have a token based system, where participants must request a control token from the participant that currently has control. The token system could be extended to allow different participants to have a control token for different parts of the screen. The floor control in the application sharing software will be



Participant
- Ashish
- Mehmood
  - Acrobat Reader
  - DNA.pdb
- gareth Lewis

connected

Participant Name: Gaz
e-mail address: s.m.hasan.rdg.ac.uk
Multicast/unicast address: 224.178.54.23
Port Number: 50012
[✓ Multicast] Unicast

Fig. 2. Screen shot of multicast application sharing tool showing different participants and session settings

designed to be as flexible as possible, with this in mind it will be possible for owners of applications to specify the floor control mechanism they require for their specific application. The remote control will work over HTTP which will be available within the proposed transport layer.

4 NETWORK TOPOLOGY

The successful integration of the previously mentioned tools into an effective collaborative environment relies upon the appropriate transport system. A popular topic in recent years has been the development of peer-to-peer applications. There are several projects which strive to develop a flexible peer-to-peer platform onto which applications can be developed, some of these are discussed in detail below.

4.1 JXTA

JXTA [8] provides a base of protocols and libraries from which developers can implement their own peer-to-peer applications. The ultimate goal of the project was to develop a solution that was flexible enough to be used for a wide variety of peer services. To achieve this goal JXTA was designed at all stages to have an abstraction from the low lying network transport protocols. The abstraction from the transport layer is achieved by wrapping all the data in XML (called messages in JXTA). This abstraction is necessary for the routing of messages but it has an adverse effect on the performance.

The main problem associated with JXTA is the overhead involved in streaming audio/video. As a generic P2P solution JXTA works well; but, as a specific tool for video conferencing, the abstraction which has been key to providing its flexibility has made it unsuitable for transferring real-time multimedia. Another feature missing from JXTA is the ability to dynamically choose the transport protocol depending upon the network topology. JXTA supports various protocols, but it is the users responsibility to choose the transport protocol required.

4.2 Groove

Groove [10] is a peer-to-peer application platform for building and deploying peer-to-peer applications. Groove is not a "pure" decentralised peer-to-peer architecture since it incorporates several central servers. Groove achieves its extensibility goals through the use of XML for data exchange. Shared components in Groove can be packaged up as objects represented in XML format and transmitted over the network using protocols like XML-RPC and SOAP.

To support near real-time communications, Groove transmits a package known as delta, representing very low-level user actions such as keystrokes or brush strokes. The concept of deltas combined with internal support for SSTP (Simple Symmetrical Transmission Protocol) allows Groove to utilise network bandwidth efficiently. Groove uses several other specialised networking protocols. The Device Presence Protocol (DPP) and a form of the Rendez-Vous Protocol (RVP) to determine the presence or absence of clients on the network for making connections. On the local subnet, Groove uses UDP. In general, the design incorporates unicast, multicast, and broadcast concepts.

Groove takes care of the underlying connectivity and synchronisation issues, allowing developers to concentrate on creating applications in the peer-to-peer space. Despite its advantages, Groove is a proprietary platform and for that reason we have decided not to use it in our work. Like JXTA, Groove also lacks the ability to dynamically choose the transport protocol.

Along with JXTA and Groove, we have also looked at some other possible candidates, but have ultimately decided upon developing our own transport layer.

4.3 Dynamic Transport System

In Section 2, we discussed the problems associated with the use of Access Grid, more specifically the desktop tools – i.e. PIG/PIGLET. These problems stem mainly from the fact that multicast is used as the transport protocol. This is not a problem in a traditional Access Grid setting, where dedicated nodes are set up for videoconferencing with dedicated network connection between the nodes. The issue is not so trivial with the desktop versions of Access Grid. As mentioned earlier, multicast has not been deployed widely over the Internet. This means that participants who are unable to multicast are forced to contact a bridge which is multicast-capable. If the participants are unable to contact a bridge then there is no possibility of participating in a meeting. Another common problem is the reliability of the multicast network. In many cases network administrators have a limited understanding of the multicast protocol and subtle changes to the network can cause problems in receiving or sending multicast traffic.

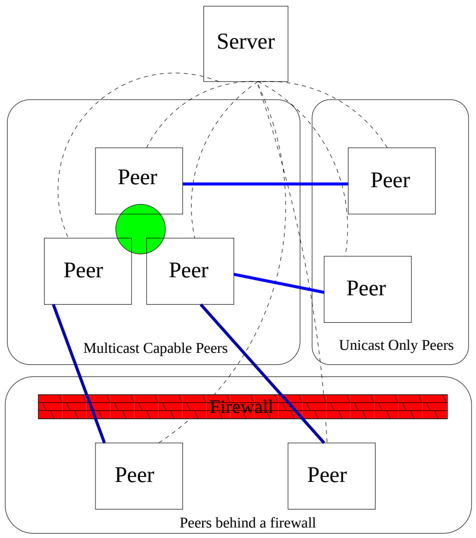


Fig. 3. Overview of the transport system's network topology

The aim of the proposed transport system is to overcome the problems with using PIG/PIGLET by incorporating network profiling. Participants that are able to multicast will join the multicast group as normal, other participants that are unable to multicast will connect to a multicast peer, which will act as a bridge and unicast datagrams to this unicast host. There are several advantages to this over using a dedicated bridge machine.

- The unicast hosts will not be connecting to a single bridge machine, they can be distributed evenly to utilise the available multicast hosts. Distributing the connections will ease the load on a single peer and so the effect on performance will be diminished.
- It will be possible for peers to dynamically change between multicast and UDP depending upon the reliability of the multicast network.
  - If the multicast fails then a multicast host can become a unicast host and associated unicast participants can be redirected to an active multicast member.
  - If the multicast is re-established then the unicast member could dynamically reconnect via multicast.

- If only one machine within a LAN has a connection to the outside, this can be used as a peer to direct the traffic to other hosts behind the firewall. This will obviously lead to problems similar to those faced with using a single bridge machine.

The proposed transport system consists of a server which controls session management. Initially, our transport system will cater for the needs of a small group of geographically distributed peers. As the number of participants increases in a session, a more distributed architecture may be used, where more servers will be added to distribute the workload. This would allow the system to be more scalable. This approach also adds robustness and resilience to the transport system.

A Session Management server will be implemented using web services to ensure that all members will have access. Once a participant has connected to the server, information about the new member will be sent to all other members of the session. Important session information such as creating sessions, joining session and leaving sessions will be sent via the server. Individual hosts will then be responsible for communicating directly in a peer-to-peer manner. The peer-to-peer portion of the transport system will take advantage of several network protocols including TCP, UDP, HTTP, and Multicast to transfer data between the peers. The controller layer will communicate with the server periodically to establish whether their connection is still active and provide information relating to other participants connection status.

The significant problem is associated with participants that are behind firewalls and have no possibility of opening ports. This poses a considerable problem, as sending the data via HTTP carries a large overhead. In such a case the audio, application sharing and possibly video could be streamed over the HTTP to a multicast host. This will act as a bridge and stream data back across HTTP. It will be interesting to observe the performance - certain concessions may have to be made, such as a picture instead of a video stream, and a freeze frame update of the shared application after set periods of time.

The Transport layer will be designed to be generic, meaning that any data can be sent or received. The Transport System will be split into three distinct sub-layers: namely, the Transport, Controller, and Session Manager layers (see Figure 4). Each of these layers will have a specific role in the transfer of data.

4.3.1 The Session Manager

The Session Manager will be responsible for deciding on the destination of the data. It will know which protocols are being used by other participants, and will provide a destination address, preferred protocol and the server address to the controller layer (1). The destinations may include multicast groups and unicast connections to other peers. The Session Manager will have an associated session GUI which will allow a host to join a VV, connect to a group containing a “buddy” and to create private sessions. It will receive state information from the controller layer

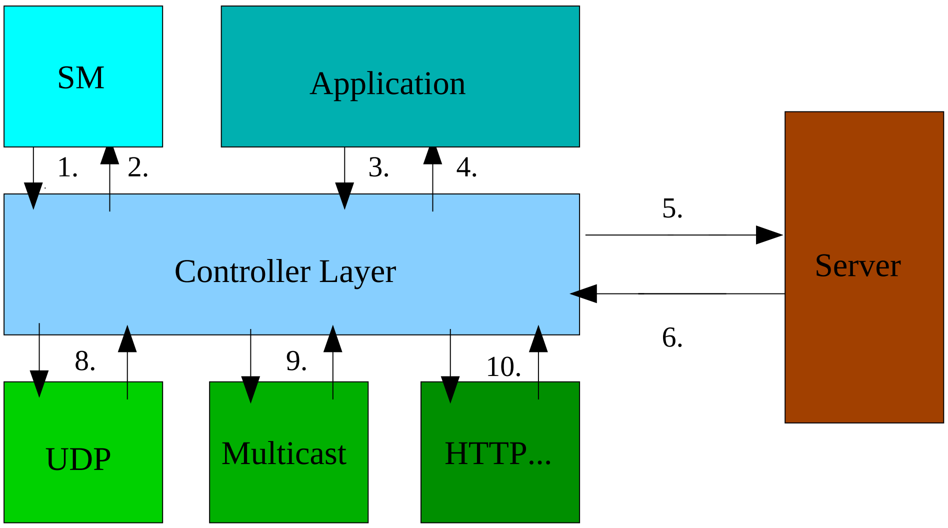


Fig. 4. Dynamic transport system architecture

and display information to the user, such as, the current transport used and the state of the multicast connection (2).

4.3.2 The Controller

The Controller will be responsible for deciding which transport protocol to use. It will know the destination and the preferred protocol from the Session Manager Layer, decide whether the preferred transport is possible, and if not, decide which protocol to use as an alternative. The controller will also be responsible for dynamically changing to another protocol if the configuration of the network changes. To achieve this the Controller connects to the server and polls for information about the connection status (5). It will also send information relating to other participants within the Virtual Venue (6). The information from the Venue participants can be used by the server to establish the state of all connections within a Virtual Venue.

4.3.3 Transport Layers

The transport layer will be responsible for actually transferring the data. The Controller module will select a protocol and a destination address based on information from the server and the Session Manager. The data will be passed to the appropriate transport module and sent either unicast (UDP), Multicast or possibly over HTTP. Using templates for the implementation of the transport layers will help ensure that any type of data can be sent or received.

4.3.4 Adding Applications

The legacy applications will be controlled by the session manager. The applications themselves will have no concept of the sessions, they will simply send and receive data of some type to the controller module (3+4). Each of the applications will have its own channel (transport system).

5 CONCLUSION

In this paper, we have discussed our experiences with using the Access Grid technologies, such as PIG and PIGLET, and have identified problem areas as well as provided solutions which overcome these issues. We have also presented an overview of the Multicast Application Sharing Tool (MAST) providing an insight into the issues involved in developing such a system. Currently there are two versions of MAST, one for Microsoft Windows and the other for Linux. Our work in the immediate future will focus on integrating these two components into a single tool, allowing application sharing between the two platforms.

We outline our plans of integrating MAST with other Access Grid technologies. These combined tools coupled with our proposed underlying dynamic transport system provide a very powerful and inclusive collaborative environment. Our target is to further develop the underlying dynamic transport system prototype into a fully functional version which would allow participants to collaborate effectively and conveniently, without an in-depth understanding of the underlying network capabilities.

Finally, although technologies described in this paper aim to create a virtual research environment allowing geographically distributed colleagues to work together towards a common goal, they have several shortcomings. It is difficult in such an environment to mimic natural human interaction effectively; there can be no physical contact between participants, and it is also difficult to use usual body language and gesture due to constraints on movement. Issues relating eye contact and facial expressions are not well addressed either. Furthermore, using multicast for communication means that messages propagate to participants at different speeds depending on the network connection between peers. This results in each participant having a different view of group members or shared applications at any particular moment in time. The above mentioned drawbacks limit the range of activities where these technologies can be effectively used. We are currently investigating these problems and plan to address some of the issues in our future work.

REFERENCES

[1] The Access Grid Project website. Available on: `http://www.accessgrid.org`.

[2] Personal Interface to Access Grid (PIG) website. Available on: `http://www.cascv.brown.edu/pig.html`.

[3] Personal Interface To Access Grid, Linux Exclusively Thanks (PIGLET) website. Available on: `http://www.ap-accessgrid.org/linux/piglet.html`.

[4] Dove, M. T.—Calleja, M.—Wakelin, J.—Trachenko, K.—Ferlat, G.—Murray-Rust, P.—De Leeuw, N. H.—Du, Z.—Price, G. D.—Wilson, P. B.—Brodholt, J. P.—Alfredsson, M.—Marmier, A.—Ptyer, R.—Blanshard, L. J.—Allan, R. J.—Van Dam, K. K.—Todorov, I. T.—Smith, W.—Alexandrov, V. N.—Lewis, G. J.—Thandavan, A.—Hasan, S. M.: Environment from the Molecular Level: An Escience Testbed Project. AHM 2003 (Nottingham 2–4/9/2003).

[5] Videoconferencing Tool (VIC) website. Available on: `http://www-mice.cs.ucl.ac.uk/multimedia/software/vic/`.

[6] Robust Audio Tool (RAT) website. Available on: `http://www-mice.cs.ucl.ac.uk/multimedia/software/rat/`.

[7] Richardson, T.—Stafford-Fraser, Q.—Wood, R.—Hopper, A.: Virtual Network Computing. IEEE Internet Computing, Vol. 2, No. 1, January/February 1998.

[8] Gong, L.: Project JXTA: A Technology Overview.

[9] Sunderam, V.—Cheung, S. Y.—Chodrow, S.—Gray, P.—Grigni, M.—Hirsch, M.—Hutto, P.—Krantz, A.—Olesen, S.—Rhee, I.—Goddard, T.—Sult, J.—Migliardi, M.—Marzilli, L.—Ano, S.—Williams, K.—Sullivan, S.: CCF: A Framework for Collaborative Computing. IEEE Internet Computing, Vol. 4, No. 1, January/February 2000.

[10] Groove Networks website. Available on: `http://www.groove.net`.

[11] Microsoft NetMeeting website. Available on: `http://www.microsoft.com/windows/netmeeting/`.

[12] SunForum website. Available on: `http://www.sun.com/desktop/products/software/sunforum/`.

[13] Crowley, T.—Milazzo, P.—Baker, E.—Forsdick, H.—Tomlinson, R.: MMConf: Aninfrastructure for Building Shared Multimedia Applications. In Proceedings of ACM CSCW '90 Conference on Computer-Supported CooperativeWork, pp. 329–342, Los Angeles, California, 1990.

[14] Abdel-Wahab, H.—Feit, M.: XTV: A Framework for Sharing X Window Clients in Remote Synchronous Collaboration. In Proceedings of IEEE Tricomm '91, pp. 159–167, Chapel Hill, NC, April 1991.

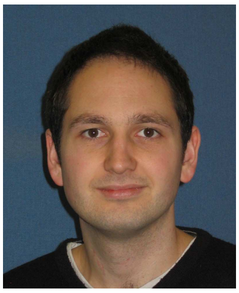
[15] Garfinkel, D.—Welti, B.—Yip, T.: HP SharedX: A Tool for Real-Time Collaboration. Hewlett-Packard Journal, Vol. 45, No. 4, pp. 23–36, April 1994.

[16] Begole, J. B.—Rosson, M. B.—Shaffer, C. A.: Flexible Collaboration Transparency: Supporting Worker Independence in Replicated Applicationsharing Systems. ACM Transactions on Computer-Human Interaction, Vol. 6, No. 2, pp. 95–132, June 1999.

[17] Lauwers, J. C.—Lantz, K. A.: Collaboration Awareness in Support of Collaboration Transparency: Requirements for the Next Generation of Shared Window Systems. In Proceedings of ACM CHI 90 Conference on Human Factors in Computing Systems, pp. 303–311, 1990.

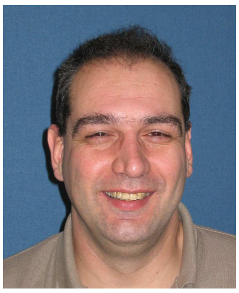
[18] Ahuja, S. R.—Ensor, J. R.—Lucco, S. E.: A Comparison of Application Sharing Mechanisms in Real-Time Desktop Conferencing Systems. In Proceedings of ACM conference on office information systems, pp. 238–248, 1990.

[19] Ziewer, P.—Seidl, H.: Transparent Teleteaching. In Proceedings of ASCILITE 2002, Auckland, NZ, Dec. 2002.



Gareth John Lewis is a research assistant within the Advanced Computing and Emerging Technologies Group at The University of Reading. His research interests are within the area of collaborative and grid computing.

S. Mehmood Hasan is a final year Ph.D. student at The University of Reading, UK. He is a member of the Advanced Computing and Emerging Technologies (ACET) group. His main areas of research are Collaborative Computing and Computer Supported Collaborative Work (CSCW).



Vassil Alexandrov is a professor in computational sciences at the School of Systems Engineering and a Director of the Centre for Advanced Computing and Emerging Technologies at the University of Reading, UK. His main interests are in the area of parallel scaleable algorithms, collaborative, cluster and grid computing and using the advances in the above mentioned areas for efficiently solving large scale scientific and industrial problems. He is currently participating in over 6 national and international projects in the area of collaborative and grid computing and e-learning.

Martin T. Dove is professor of computational mineral physics in the Department of Earth Sciences, University of Cambridge and director of the National Institute for Environmental eScience.■ His main scientific research interests are in computer simulations of the behavior and properties of materials at a molecular level. Recently he has been concerned with applying grid and escience methods to support collaborative work on molecular simulations, and leads the UK eMinerals project on this work.

Mark Callejais the grid computing specialist at the Cambridge e-Science Center (CeSC). A theoretical condensed-matter physicist by training, he is now involved in commissioning, integrating and developing grid tools and middleware for the University of Cambridge.