Proceedings of the UK e-Science All Hands Meeting 2005, ©EPSRC Sept 2005, ISBN 1-904425-53-4

Multicast Application Sharing Tool for the Access Grid Toolkit

S Mehmood Hasan[1], Gareth J Lewis[1], Vassil N Alexandrov[1], Martin T Dove[2], and Matt G Tucker[3]

[1]Advanced Computing and Emerging Technologies Centre, School of Systems Engineering, University of Reading, Whiteknights, P.O. Box 225 Reading, RG6 6AY, UK
[2]Department of Earth Sciences, University of Cambridge, Downing Street, Cambridge CB2 3EQ
[3]ISIS Facility, Rutherford Appleton Laboratory, Chilton, Didcot,OX11 0QX

Abstract

Multicast Application Sharing Tool (MAST) allows geographically distributed participants to share arbitrary legacy applications. MAST supports scalable group to group collaboration by using multicast. It is being used within the eMinerals project to augment the Access Grid functionality. In this paper we describe MAST and its deployment as an Access Grid node service within the eMinerals Virtual Venue.

1 Introduction

The e-Science community is benefiting from research derived from multi-institutional collaborations. These collaborations maximise the potential for sharing of expertise and experience between partners and must be supported by collaborative tools. These tools aim to complement human face-to-face communication by providing various modes of interaction between geographically distributed peers. Collaborative tools create a virtual work environment on multiple computer systems connected over the Internet.

Multicast Application Sharing Tool (MAST) enables sharing of legacy applications in a group to group setting. It has been developed as part of the eMinerals project [1] to be used specifically with the Access Grid [2] environment. Access Grid is an advanced collaborative environment, which is used for group to group collaboration across the Internet. The communication offered by the Access Grid can be enriched by an ability to share, modify and collaboratively create data and information. MAST allows geographically distributed participants to share anything from subject specific applications, such as a molecular viewer to generic office programs, such as a PowerPoint presentation. This paper describes the integration of MAST into the Access Grid environment by deploying it as a node service. This allows participants or group of participants within an Access Grid session to engage in various collaborative activities beyond audio/video conferencing.

The paper is organised as follows: Section 2 describes MAST in detail, including various implementation issues, and provides a comparison with the state of the art. In Section 3, we discuss the Access Grid Toolkit and the concept of node services. This discussion leads us to the deployment of MAST as an Access Grid service in Section 4. The penultimate section presents our initial experiences, and we conclude with remarks on the work presented in this paper and a roadmap for our future work in Section 5.

2 Application Sharing

Audio/Video conferencing is essential for an effective collaborative experience, allowing participants to mimic natural human inter-

action. Another major component involves sharing of material between participants. This includes activities such as collaborative viewing of visual data, document editing by various colleagues and collaborative code development between geographically distributed peers. Application sharing is the ability to share and manipulate desktop applications between multiple participants, thus facilitating the aforementioned activities.

Application sharing provides functionality that is essential in some group meetings. There are two main approaches to implementing application sharing; collaboration aware and collaboration unaware sharing. Collaboration unaware sharing involves an application developed to share the graphical representation of the programs running on the system. These tend to be less responsive to user interactions and generate relatively large amounts of network traffic. Due to the applications being unaware of the collaboration, there is no need for them to be adapted to work in a cooperative manner. Collaboration aware applications, on the other hand, are developed to support cooperative work by providing mechanisms to synchronise the "shared view" between users. The synchronisation puts less demand on network resources and so the interactive response is generally better. However, as mentioned earlier either applications are specifically developed for this purpose or they are adapted to work in a cooperative manner, which may not be possible for many commercial applications.

2.1 Related Work

The Access Grid Toolkit provides some default collaboration aware applications, such as, shared browser, shared question tool and shared presentation. It also includes the functionality to deploy newly created applications. These suffer from the disadvantages mentioned in the previous section. Several collaboration unaware application sharing solutions are also currently used with the Access Grid, these include; Virtual Network Computing (VNC) [5], inSORS IGPix [6] and Distributed PowerPoint (DPPT) [7]. The main problem with these application sharing systems is their scalability within a group environment.

Virtual Network Computing (VNC) has traditionally been used with the Access Grid in order to share applications during meetings. It shares the entire desktop with the participants in the group, which may not be desirable. However, it can be 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 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 - one participants runs the VNC server within the group and all other participants connect unicast to that server. Each participant establishes a separate TCP connection with the server, and the server sends out a copy of the same data on each connection. This increases the load on the server with each extra participant and results in inefficient use of network resources. VNC approach clearly does not scale well and is therefore unsuitable for group collaboration.

- All the participants wishing to be part of a VNC session must acquire the IP address of the VNC server machine prior to the session. Furthermore, trust relationships between hosts must already be in place for the communication to take place. This puts a restriction on spontaneous collaboration between peers.

- Sharing several applications within a group becomes complicated with VNC. The situation becomes quite confusing with several people sharing at the same time because it is difficult to keep track of which peer is sharing which desktop. Furthermore, one must keep track of the IP addresses of each VNC server machine in the group.

There is a multicast extension to VNC which attempts to overcome the scalability problem associated with the original VNC tool. MulticastVNC [8] was developed as part of a TeleTeachingTool (TTT) and hence

restricts participants (students) to be mere viewers in a session. The TTT server acts as a proxy, receiving unicast traffic from the teacher's machine running the VNC server and multicasting the traffic to all the students within a particular multicast group. This overcomes the problem of bandwidth, but adds the inconvenience that other participants can act only as viewers. There is another restriction which means each client is must establish a TCP connection to the TTT server to initialise the Remote Framebuffer Protocol (RFB) [10] (used in VNC) and get the multicast address and port for the session. This connection is terminated upon initialisation and the client then connects to the multicast group.

inSORS IGPIX is a commercial application sharing software. It works by grabbing the graphical data associated with the shared application and sending it to the inSORS server based at their site. The clients then make individual connections to this server over http, using the URL circulated by the presenter prior to the meeting. This clearly does not provide a scalable model as individual connections are made by each client to the server. Moreover, clients are mere viewers in this session, which can be quite restrictive.

DPPT allows the sharing of PowerPoint slides between geographically dispersed peers. It is more responsive and makes more efficient use of the network resources as it only transmits the events rather than the graphical data. However, there are several drawbacks with this approach, these include:

- All participants must download the PowerPoint slides before the collaborative session can commence
- It is designed for sharing PowerPoint presentations and therefore no other applications can be shared
- It does not work well with animations on the slides, this is because only slide change and mouse/keyboard events are propagated to all the participant

2.2 Multicast Application Sharing Tool (MAST)

Our approach is based on providing a scalable solution for group to group collaboration. This is achieved by using multicast - where interested parties join a group and messages are propagated only to the group members. The important point to note here is that the sender only needs to send one copy of the message which is then delivered to all interested participants. In a group scenario this is clearly a better approach than unicast, where a separate copy of the same message is sent out for each participant thus increasing network traffic and load on the sender. Broadcast, on the other hand, involves propagating the message to everyone on the local area network (LAN) thereby flooding the network unnecessarily.

2.2.1 Update Mechanism

MAST shares an application's visual representation among participants within a group, allowing changes to the application to be propagated to all members. Sending visual data associated with the whole application would be inefficient. If only a small proportion of the screen is changing, it would make more sense to send only the changes. MAST achieves this by splitting the visual representation of the legacy application into sections. Each section is checked to see if there is a change. If the section has changed, it is sent to the other participants. The receiving participants identify the section that has changed and update their local view of the application. Therefore, MAST reduces the network load by sending only updated regions and also compressing the data before transmission.

Another important issue involves the frequency of obtaining the graphical data. The visual data for the whole application could be acquired and sent to other participants each time an event occurs or after a set interval. Changes to the visual representation of an application can occur due to hardware or software events. If the user clicks a button on the application, the view of the application will change in response to the event. Similarly if an application initialises an event, such as, an animation, or a progress bar, the visual representation of the application would change. There are two possible methods for reacting to these events - the first is to check the visual representation after a default interval. This method works well for software events, by updating the screen to show changes not

induced by external events, such as mouse or keyboard events. The interval between checking for updates is extremely important. The default interval value must attempt to provide good responsiveness, whilst ensuring relatively low overhead. To reduce wastage of processor time, the interval must be relatively high. This makes using the interval method unacceptable for changes due to external events. In a standard operating system users would expect the visual response to an event within several hundred milliseconds. If the interval is set much higher, then the shared application would appear "sluggish" to the user. The alternative to setting the interval is to check sections after an external event. It is sensible to assume that the visual representation of the shared application will change once it receives an external event. Using this method ensures that the shared application will remain responsive.

MAST combines the two methods to achieve an ideal balance, achieving high responsiveness and avoiding unnecessary overhead. The update interval can be set relatively high to avoid undue wastage of processor time, whilst still capturing changes that are not due to external events. Checking the segments after each external event (associated with the shared applications window) means that visual changes are processed quickly, improving the interactive performance.

2.2.2 Using Multicast: Benefits and Restriction

In an application sharing scenario, one participant has the application running on their desktop which is then shared within the group. This means the same data must be transmitted to many participants. It is a one-to-many connection and therefore the typical multicast situation. Using multicast makes MAST a scalable system for sharing applications. However, using multicast means that MAST must also overcome some restrictions. Multicast is based on UDP (User Datagram Protocol), which is a connectionless transport-layer protocol. This means the transmission of data is unreliable and there are no quality of service guarantees.

2.2.3 Unreliable Transmission

With IP Multicast data packets can arrive out of sequence and there is no guarantee of delivery. If some packets are lost then the remote participants' view of the shared application could be incomplete. MAST does not have any knowledge of packet loss and therefore it would assume that the section is updated and will not resend the section. To overcome this problem, and to accommodate latecomers, MAST must resend sections periodically even if they appear to be unchanged since the previous send. Resending sections could be done all at once after a set intervals. However, at that moment it could take a relatively long time to obtain visual data for the entire application not to mention the increased load on the network. During this high overhead period external events from the user could take longer to process and so the responsiveness of the shared application would be affected. MAST attempts to balance this load by resending a few sections after each interval, this reduces the overhead associated with refreshing the shared application, and maintains the responsiveness to the application user.

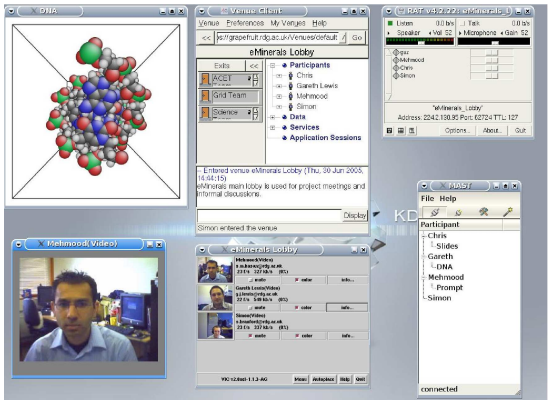

Figure 1: Screen shot of a molecular viewer being shared using MAST within an Access Grid session on a Linux machine

A crucial factor while designing MAST was to ensure that there were no dependencies between data packets i.e. all the information about a packet was self contained. It is also important to tolerate missing packets and deal with packets received out of sequence. Each segment in MAST is compressed into one packet and each packet contains a header. The header provides the receiver all the information needed to display that particular section on the screen appropriately.

2.2.4 Session Management

MAST provides each participant with a list of all other participants in the session

and their respective shared applications (see Figure 1). An important aspect of the transport system is the identification of the each participant and their shared applications. This is achieved by providing each participant with a unique participant id and generating a unique application id for each shared application in the session. MAST does not use a central server, which means that the participant list cannot be managed centrally. Each participant must manage their own list, identifying each participant and application stream. When a new application stream is received, MAST checks if the owner of the shared application is present in its own participant list. If the participant is present, then this application is added to the list. If the participant is not present, then the participant and the application name is added to the list.

Each instance of MAST must be responsible for detecting participants that leave the session or applications that are no longer being shared. MAST supports explicit leave/delete messages, these are sent when a participant leaves the session or stops sharing an application. When a participant receives such a message, list is updated appropriately. It may be possible that a peers was not able to send a leave/delete message before exiting the session. This could happen due to abnormal program termination, system shutdown or network related problems. In this case, detecting leaving participants is relatively simple - while a stream is being received a flag is set to indicate that a participant and their application is active. After a set interval, these flags are cleared implying that the entries have become inactive. List entry flags are checked periodically and if any entries have a cleared flag at the next interval then these would be removed from the list.

2.2.5 Unicast Support

As mentioned earlier, multicast provides a scalable solution for group collaboration. However, the use of multicast is not ubiquitous and it is currently not available to many institutions as well as home users (due to the reluctance of the Internet Service Providers in adopting this technology). This is expected to change with the adoption version 6 of the internet protocol (IPv6). In the meanwhile, in order to create an inclusive application sharing environment MAST provides supports for unicast transmission. This is achieved by using with a multicast bridge, such as, QuickBridge [9]. A bridge works by joining the appropriate multicast group on behalf of the unicast participants interested in a multicast session. The bridge then receives data from the multicast group and forwards it unicast, sending a separate copy to each of the unicast participants. There must be a possibility to detect if unicast participants are still interested in receiving packets. In the case of multicast transfer this is done automatically by the network, but with unicast transmission the UDP packets are transmitted even if the client is no longer running. A bridge accomplishes this by only forwarding packets to active participants. In order for participants to be active, they must send a keep alive message periodically. MAST has been configured to keep the state of the participant active with the bridge by sending short messages periodically. This guarantees that only needed unicast traffic is produced.

3 Access Grid Toolkit

The Access Grid Toolkit (AGT) is an advanced collaborative environment providing audio and video conferencing facilities, services, shared data and shared applications. This technology has been widely adopted within the academic community and is used by the eMinerals team to hold regular meetings. MAST is being used within the eMinerals project to augment the Access Grid functionality.

The Access Grid Toolkit consists of two major components; the Venue Client and the Venue Server. The Venue Server hosts the Virtual Venues, which act as a meeting point for peers with similar interests. These can be modified using a Venue Management Tool. The Venue Server is responsible for managing the Virtual Venues and accepting connections from participants via Venue Clients. The Venue Client runs on the client machine and allows the user to move between Virtual Venues to share data and collaborate using the available media tools.

There are two main mechanisms available for extending the functionality of the Access Grid; the creation of shared applications [11] or deployment of services using the

node service infrastructure [12]. The node service infrastructure is used for the two main media tools VIC [3] for video and RAT [4] for audio. Each node has a single AGNodeService, each machine in the node has a AGService Manager and each media tool requires a AGService. The shared application architecture allows the AGT functionality to be enhanced by providing an interface to which collaborative tools can be added. The architecture consists of a Application Service (for registering participants) and an Event Channel for propagating messages to other registered participants.

A major difference between the two mechanisms is the origin of the messages. In the shared application infrastructure, the messages originate from one of the client machines (for example the IP address of a client machine wishing to share its desktop). The message is propagated to all the registered participants via the Event Channel and each client can act on this information. This model works well in a situation where one of the clients is acting as a server and any changes made by the client can be sent to all registered participants. The node service infrastructure propagates messages that originate from the Venue Server. Amongst other information sent to the current AGT media tools is the multicast address and port (resolved by the Multicast Address Allocator). This information is used to allow each client's media tools to connect to the correct multicast group. These tools provide the scalability that is synonymous with the Access Grid philosophy.

3.1 Deploying MAST as an Access Grid Service

The Access Grid Toolkit provides default services for the video (VIC) and audio (RAT) tools. These tools run on the client machine using information provided by the Venue Server. Each of the Virtual Venues (VV) on the server must provide a multicast address and port for the different media tools. There are two ways in which this can be configured; the default method for the AGT is for the Venue to dynamically allocate addresses using the Multicast Address Allocator. The same dynamic address must be used for each type of media tool within a VV. The second method is to allocate static addresses for each media tool within the VV. A Virtual Venue can be configured using the Venue Management Tool and modified to use static multicast addresses. The Venue Management Tool only allows static addresses to be specified for audio and video. It is possible to extend the python scripts on the server side to configure other services with static addresses, but we felt using the dynamic address mechanism provided a more elegant solution.

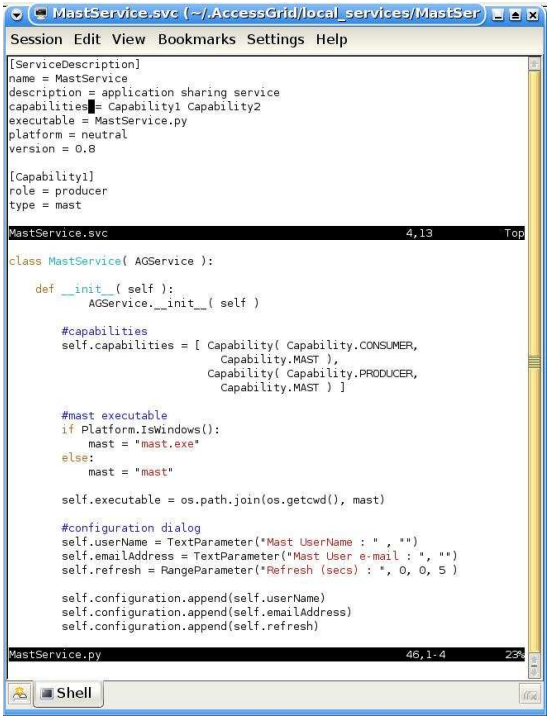

Figure 2: Screen shot showing part of the MastService python script and the MastService configuration file

MAST is intended to work with the Access Grid, employing the same scalable model. Like VIC and RAT, MAST requires all the participants to join a common multicast group. The node service infrastructure provides the most appropriate solution for deploying MAST within the Access Grid. To achieve this, there are several steps which are needed. A service for MAST must be created within the AGT and MAST must be adapted to accept the information sent to it by the Venue.

A service can be constructed by creating two files, a MastService python script - which is used to control the MAST media tool on the client machines and a MastService configuration file which describes

attributes of the service, such as its name and capability. A snapshot of these files can be seen in Figure 2. To create the MAST Service, a MAST capability must be declared in the Types python script so that it can be used within the MAST service script. This capability describes the type of information being transferred. The MAST Service has two roles; it acts as both a consumer and a producer of data. There are several important parts to the Service script, including the initialisation of the AGService, configuration, starting and stopping the service. Once completed, the script files are packaged in a zip file with the MAST executable and moved to the appropriate directory (depending on the platform).

The MAST application must also be adapted to work with the MAST Service and accept the command line arguments in the format "multicast address/port". This information is used to automatically connect to the correct multicast group when a client enters a new Venue. In such a situation, the MAST menu items used for opening saved connections and creating new connections are disabled to ensure that control of an application's connection is exclusively restricted to the Venue Server.

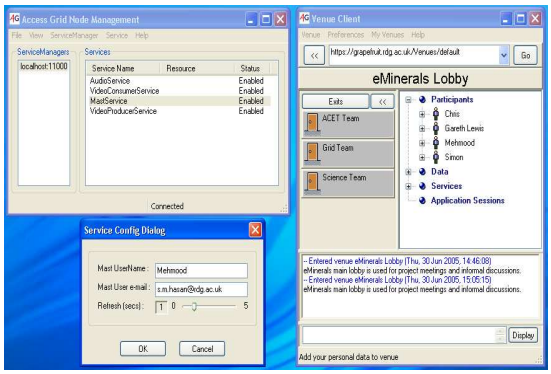

Figure 3: Screen shot showing MastService enabled on a node and the MAST configuration options on a Windows machine

User can add the MAST Service using the Service Manager within the Venue Client. Figure 3, shows MAST Service enabled on a node together with a MAST configuration dialog. If the MAST service is enabled when entering a new Venue, the connecting client will be allocated the multicast address and port necessary to share applications using MAST within that particular Virtual Venue.

4 Initial Experiences

The initial deployment of MAST across the eMinerals project has required tweaking of various options and parameter within the tool. MAST uses unreliable transmission and therefore packet loss on the network becomes a major consideration. As mentioned earlier in the paper, MAST sends the graphical representation of the shared application periodically as well as after external events. If a small percentage of packets are lost then the periodic update interval can be set to a relatively high value, thus reducing the network traffic and the load on the sender. On the other hand, if a large percentage of packets are lost, all group members may not have the same view of the shared application - a situation which is highly undesirable in a synchronous collaborative environment. In this scenario, the update frequency is increased to counter the packet loss on the network. This increases the CPU usage but allows all the participants to have an identical view of the shared application. Theoretically, it is possible that even with high update frequency the users' experience is compromised due to a very large percentage of packets lost on the network. However, we have not encountered that scenario during the testing process.

Segment size is an important factor in determining the amount of traffic generated to share an application. MAST divides the shared application's window into several segments. Each segment is compressed into one packet and transmitted to the entire group. As each packet is compressed into one packet, the size of each packet depends on the size of the initial segment and the rate of compression. In our experience, sharing document editing applications generates less traffic and average packet size is quite small in general. However, packet size tends to be relatively large with graphical applications. This is due to the rate of compression - with high resolution graphics the compression is comparatively less effective resulting in larger packet sizes. The segment size is determined dynamically depending on the size of the shared application and the host machine display resolution. This ensures packet size is kept small enough and the packets are not discarded.

Deploying MAST as an Access Grid node ser-

vice provides a convenient way of interaction for the participants. Prior to this, the multicast address and port had to circulated via other means, such as email etc. However, if MAST Service is enabled when a participant enters a Virtual Venue, MAST automatically connects to the multicast group allocated by the Venue Server. The point to note is that this is a seamless process to the users, which in our initial experience has proved to be a very useful feature specially for novice users.

5 Conclusion and Future Work

In this paper, we have presented the deficiencies of existing application sharing tools and provided a detailed discussion on design and implementation of the Multicast Application Sharing Tool. MAST has been specifically developed for group collaboration and complements the Access Grid functionality. The use of MAST with Access Grid Toolkit builds a virtual work environment allowing geographically distributed peers to collaborate in a meaningful manner. Deploying MAST as an Access Grid node service provides a convenient way for participants to interact. It enables use of the tool in a similar manner to other default services, such as audio and video services.

In the future, we plan to optimise various subsystems of MAST in order to enhance the user experience, and use machine and network resources more efficiently. This can be achieved by using separate channels for session information and application related data, allowing application data to be processed more quickly. Additionally, the used encoding could be improved to reduce network traffic in order to support low bandwidth networks.

Acknowledgements

We are grateful to the National Environmental Research Council (NERC) for their financial support

References

[1] Dove, M.T. et. al.: Environment from the molecular level: an escience testbed project. AHM 2003 (Nottingham 2-4/9/2003)

[2] The Access Grid Project website. Available on: http://www.accessgrid.org Last accessed on 30 June 2005

[3] Videoconferencing Tool (VIC) website. Available on: http://www.mice.cs.ucl.ac.uk/multimedia/software/vic/ Last accessed on 30 June 2005

[4] Robust Audio Tool (RAT) website. Available on: http://www.mice.cs.ucl.ac.uk/ multimedia/software/rat/ Last accessed on 30 June 2005

[5] Richardson, T., Stafford-Fraser, Q., Wood, K.R., Hopper, A.: Virtual Network Computing. IEEE Internet Computing, Volume 2, Number 1 January/February 1998

[6] The inSORS website. Available on: http://www.insors.com Last accessed on 30 June 2005

[7] von Hoffman, J.T.: Guide to Distributed PowerPoint. Retrieved on 30 June 2005 from http://accessgrid.org/agdp/guide/dppt.html

[8] Ziewer, P., Seidl, H.: Transparent Teleteaching. In Proceedings of ASCILITE 2002, Auckland, NZ, December 2002

[9] Daw, M., von Hoffman, J.T.: Guide to Netwrok Bridging on the Access Grid. Retrieved on 30 June 2005 from http://www.accessgrid.org/agdp/guide/network-bridging/1.0/html/book1.html

[10] Richardson, T.: The RFB Protocol. Version 3.8. RealVNC Ltd. Retrieved on 30 June 2005 from http://www.realvnc.com/docs/rfbproto.pdf

[11] Futures Laboratory: Programmer's Manual - Shared Application. Retrieved on 30 June 2005 from http://www-unix.mcs.anl.gov/fl/research/accessgrid/documentation/developer.html

[12] Lefvert, S.: Programmer's Manual - Node Services. Retrieved on 30 June 2005 from http://www-unix.mcs.anl.gov/fl/research/accessgrid/documentation/developer.html