

# Access Grid Anywhere

Anja Le Blanc, Andrew Rowley, Tobias Schiebeck, Martin Turner

Research Computing Services, The University of Manchester, UK

Email address of corresponding author: Tobias.Schiebeck@manchester.ac.uk

**Abstract.** This paper describes the portalization of the Access Grid Toolkit and its impact on the e-Research Community. Presenting results from user-surveys, we discuss the issues involved in using and installing the existing Access Grid software. Then the advantages that a web portal provides are given in terms of both solving the issues that have been raised, but also providing a research environment that in the future is customizable for specific research teams and needs. We describe the creation of the new portlet called PAG (Portlet Access Grid) and its features, including the technical challenges that are presented by the integration of the two technologies. We then discuss how users are being involved in the process of development, to ensure that the software will be useful in a real life context.

## Access Grid

The Access Grid (AG)<sup>1</sup> is an advanced collaboration environment, in which users not only communicate using audio and video, but are also able to share data and collaborate using various add-on applications. It is designed around the idea of a group-to-group conferencing system, allowing users from multiple different sites around the world to join in meetings by using “virtual venues”. The Access Grid Toolkit (AGTk) software is used to navigate between these venues.

The current AG community has over 20,000 users across 56 countries worldwide. In the UK, the Access Grid Support Centre (AGSC) has over 150 registered room-based nodes, and over 200 additional users registered with desktop-based AG nodes. This is a broad user community, who can provide a constituency for feedback for developments arising from the PAG project. AGSC training workshops, documentation, and the AGSC website, all provide readily accessible means for dissemination of development outcomes.

AG is a collaboration environment useful to all researchers as research in general involves meetings, communication and data exchange. Leading research in any subject area nowadays almost always includes experts distributed all over the world, so fast and flexible meetings and worldwide collaboration are essential to create top quality results. The AG technology also allows us to reduce the carbon footprint of worldwide research by having researchers communicate efficiently remotely and thus reducing the amount of travelling required.

Traditionally the Access Grid is available either as Open Source software – the Access Grid Toolkit (AGTk) – or as a commercial application from IOCOM<sup>2</sup>. The advantage of the first is that it is free and advice is provided via an active mailing list and development community.

---

<sup>1</sup> *AccessGrid*; <http://www.accessgrid.org>

<sup>2</sup> *IOCOM*; [www.iocom.com](http://www.iocom.com)

Using the commercial product you can expect a greater availability of support with a specific level of service agreement, and it currently has a more user friendly interface. AGTk and IOCOM are mostly compatible with each other provided all connecting sites use a common audio and video codec. The AGTk is available for Windows, Linux, and MacOS whereas IOCOM is currently only available for Windows. There are also add-ons available for AGTk (Shared Applications and Node Services). The commercial software does not provide these add-ons making it more difficult to use them in general.

Access Grid is designed for group to group communication; or to be more precise for communication between several groups of people. Standard video conferencing tools often work well when there are only two sets of people involved; when there is a third party additional network traffic is produced. This is avoided with Access Grid with the use of multicast that allows the network traffic to be replicated at the routers between the sites communicating, thus requiring less overall bandwidth in general.

To remove some of the remoteness always experienced in meetings across multiple physical sites, it is recommended where possible to use dedicated room based Access Grid nodes. These dedicated nodes often allow the remote video streams to be placed on a wide projection wall in as close to real human size as possible. All Access Grid providers also support 'personal' nodes meaning that they are based on ordinary personal computers with one or two screens. This enables people to attend meetings even if they cannot reach a room based node physically.

Common uses of the Access Grid include seminars, presentations, and distant learning. The presenter is usually giving the talk from a room based node, but some of the audience might want to listen to the talk from their computer. Personal nodes are not designed to compete on the video conferencing market but are meant to enable the participation of people outside of dedicated rooms. An external tool (ScreenStreamer<sup>3</sup>) enables the sending of the local desktop as a video stream into an Access Grid venue. This is a stand-alone tool and therefore can be used regardless of the software installed on the client node.

The AG currently has a number of problems that restrict the user uptake in the wider research communities. These include installation issues as well as complex networking problems caused by unreliable networks or firewalls blocking the majority of network ports. Installing a room based node is a major commitment from an institution, but in some instances it is overlooked that you also need somebody to look after the room in order to keep it in continual good working condition. Advice and help is currently available to UK based institutions via the Access Grid Support Centre<sup>4</sup> but this can not solve remotely all local maintenance issues that may arise. Further, for inexperienced users, an operator is required to set up the meeting and assure quality throughout. An operator is always advisable so that other participants can concentrate on the content and can forget about the technology.

The non-technical users in the past have generally been frightened by the complex user interface of the Access Grid Toolkit software. This has been worsened by the non intuitive language used and a lack of simple documentation.

The criticism of the AG commonly heard is that the video quality is not very good. The reason for the less-than-ideal video quality is generally down to the codec used to transmit the streams. In the AGTk there are many codecs to choose from, but the default is a simple,

---

<sup>3</sup> *Developed in the Memetic project and available from:* <http://www.memetic-vre.net/software/ScreenStreamer/>

<sup>4</sup> <http://www.ja.net/services/video/agsc/AGSCHome/>

fast, and small bandwidth codec. New developments are already integrated into the latest release of AGTk. The AVATS<sup>5</sup> Project is set up to improve audio and video issues in the media tools VIC and RAT used within the Access Grid software.

## Portal Technologies

A web portal is a site that functions as a main point of access to streams of information often collected from the World Wide Web. Portals are often used to present information from diverse sources in a unified way. Aside from the search engine as an example of a standard portal type interface, web portals commonly offer other services such as news, stock prices, and infotainment. Portals have the advantage that they allow enterprises to provide a consistent look and feel with their own access control and structure for multiple applications, which otherwise would have been different entities altogether.

Over the last few years an informal definition of a portal has evolved to be a web-based application that offers a form of single-sign-in (although sometimes this can be anonymous) and personalisation for content aggregation as well as the ability to have multiple different layers of presentation and display. The different sources of information that are managed by the portal have been termed channels. Channels can reach in complexity from simple streams of information produced by single remote services to interactions with multiple complex remote applications. A portlet is defined as a specific set of components within a portal that deal with a particular information channel.

A key advantage of creating only a portlet interface is that the portal itself provides user authentication and authorization through some form of simplified or single-sign-in process. This means that the user can sign in with a familiar username and password, which is agreed by the enterprise that manages the portal. The portal can also store portlet specific information for each user, so the user does not have to enter details such as their name or e-mail address into the portlet.

It is worth noting that portals are not just about bringing a number of different channels of information together but have the ability to create an environment. It has been noticed that on the level of portal complexity a system could and possibly should be a complete desktop environment, hiding all the complexity of the operating system, applications and networking from the users.

Many higher education institutions now offer their students institutional portals, providing email service, access to course material in virtual learning environments (VLE), relevant exam time tables, links to other relevant sites, integrated university news, and much more. Sites can be customized either for groups of people (staff, student) or individually per person. The look and feel of an institutional portal is usually in the style of the institutional web presence.

One of the advantages of developing portlets is that they are deployable into other portals without individual integration work being done, provided they follow a portal standard. Following portal standards has advantages for both the portal developer as well as the content (portlet) developer. If the portal supports a portlet standard, vendors do not have to provide specialized portlets, and application developers can ensure that their portlet works in any

---

<sup>5</sup> <http://www.cs.ucl.ac.uk/research/avats/>

portal supporting this standard. Currently there are two widely accepted standards and one newly emerging one.

- **WSRP** (Web Services for Remote Portals) is a communication protocol that allows 'portlets' (packaged content and/or computing capabilities) from one application to work within another. In this case, the portlets are centrally hosted and only 'linked' into the remote portals.
- **JSR 168: Portlet Specification** is a collection of Java APIs for portlet developers. The specification defines a common Portlet API and an infrastructure that provides facilities for personalization, presentation, and security. The portlet is hosted on the portal site. The majority of java based portals support JSR 168. The Portlet Specification defines:
  - The portlet container contract and portlet life cycle management
  - The definition of window states and portlet modes
  - Portlet preferences management
  - User information
  - Packaging and deployment
  - Security
- **JSR 286: Portlet Specification** is a draft for an extended portlet specification based on JSR-168. It aims to close gaps identified by the community. Portal vendors were filling those gaps with custom solutions which cause incompatibility with other vendors. Improvements will be made to:
  - Events — enable portlets to communicate with each other through sending and receiving events.
  - Shared render parameters — enable portlets to specify which render parameters they can share with other portlets.
  - Resource serving — enables portlets to serve resources within the portlet context.
  - Support for asynchronous technologies, such as AJAX

These technologies are not entirely exclusive. Clearly JSR-286 will be backwards-compatible with JSR-168. Additionally, there has been some work on providing a JSR-168 compatible WSRP container. This allows existing JSR-168 hosted portlets to be accessed via WSRP without requiring any additional development.

## Need for an Access Grid Portlet

### Gathering User Feedback

In 2006, the Access Grid Support Centre carried out a survey of the Access Grid community. A similar survey has now been carried out in 2007. The 2006 survey revealed that the most common problem across sites was “*Difficulty connecting to a venue (e.g. firewall or multicast problems)*”. It was also noted that “*not being a multicast site*” was a problem as then the users had to remember to “*...switch to the unicast bridge ...*”. Then in the 2007 survey, it was further noted that “*AG Firewall rules and Multicast/Unicast are a nightmare – These need to be simplified and made more reliable*” and another user stated that “*Many*

*partners have problems connecting because they are setting up temporary nodes. Most of the problems are due to inexperience with NAT (port forwarding) & firewall configurations”, which shows that the situation had not improved.*

In the 2006 survey, users were asked what improvements they would most like to see in the Access Grid. The first most requested improvement (requested by 35% of the respondents) was *“More reliability”* with the second being *“Greater coverage of Access Grid across institutions that do not currently have it”* (requested by 16% of the respondents). Users also said that the *“Speed of the venue client”* was too slow, *“Make it more user friendly”* and *“...integration into portals ...”*. In the 2007 survey, users were asked what they would like to see the AGSC provide. The users commented that *“The AGSC should work towards providing training and advice for user managed desktop access to the system”* and *“Increase in use of the personal access grid”*.

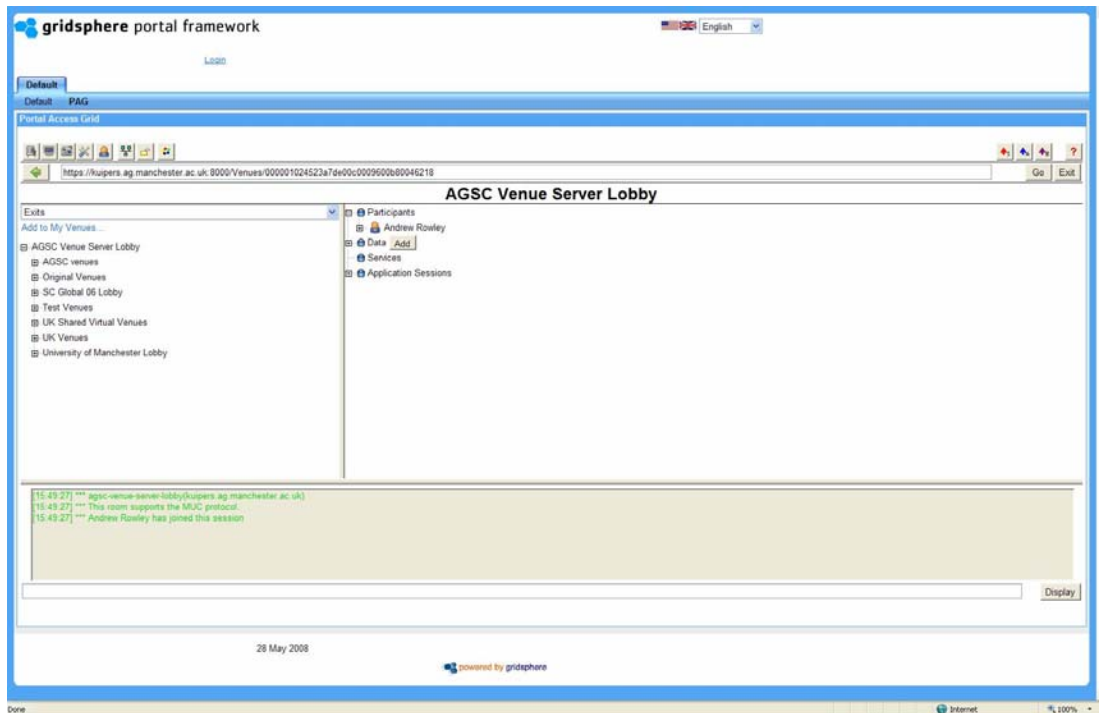
In both surveys, users were asked for additional comments. From the 2006 survey, users commented that *“Generally a lot of time is spent configuring the software ...”* and that *“...Many systems exist for supporting web seminars ... the potential of the AG in this respect seems to have been ignored.”* From the 2007 survey one user commented that *“It needs to be as easy to use and reliable as any of the commercial video conferencing software you can buy.”* Further examples of comments are *“We only use personal AG, which is a much better option for getting people interested in and using the AG ...”*, *“I have to phone from China because my student didn't manage to install the AG SW and get it working”*, *“Messenger ... ConferenceXP ... are much easier to setup, configure and to use”*, *“The process of configuring a personal node seems far more troublesome / complicated than it should”* and *“If I could connect to some meetings from the desktop, I would use the AG system more”*

This work addresses these issues by providing the specifications and functionality for a reliable, fast and highly usable portlet that requires minimal setup time and configuration. The portlet allows the Access Grid to be used more effectively from behind restrictive firewalls, and also detects and switches between multicast and unicast without user interaction. IOCOM users will also benefit from this development, as the portlet allows them to access the shared applications within a venue without installing the Access Grid Toolkit. This enables a greater degree of collaboration in meetings involving participants with IOCOM and AGTk software.

## The PAG Project

In 2007 the Open Middleware Infrastructure Institute (OMII-UK) funded “Portal Access Grid” (PAG), a project that aims to create a portlet version of the Access Grid Toolkit (AGTk) client. The idea behind the portalization for the AG is the general availability of web access even in remote places, thus increasing the use and availability of AG meetings. Most institutional firewalls don't restrict the standard ports used by web applications, so the use of a standard web browser enables the whole research community to be engaged in collaborations through the AG. All researchers have used web browsers and portal technologies, which lowers the barrier for the uptake of PAG.

As the portlet technologies are defined by generally accepted standards, this also provides a way of creating a cross-platform implementation of AG technologies. The implementation provided has been successfully tested on three of the main portal frameworks (Gridsphere, uPortal and Sakai) and is expected to run on any future portal framework that will incorporate the main standard.



**Fig 1.** The PAG portlet user interface hosted within a Gridsphere portal

Another advantage of using the portlet approach is that the software is provided centrally, so only the portal provider needs to carry out updates and installations. All tools necessary on the client side are checked and downloaded automatically as required. This resolves issues of complicated installation procedures.

The AG portlet can be provided within institutional portals as well as by services providers such as the National Grid Service (NGS) or the National Centre for e-Social Science (NCeSS). This makes the AG easily available to large sections of the UK research community.

The PAG project also aimed to ease the restrictions imposed by network firewalls and bandwidth limitations through the implementation of a TCP/IP bridge and automatic multicast/unicast detection and switching.

## Features

The Portal Access Grid is a complete replacement of the VenueClient provided by the AG Toolkit (AGTk). It is presented through the portal framework, which allows easy access through a web browser. Installation procedures are minimal for the end user, as only a working instance of Java is required. This model has the advantage that a service provider can centrally care about the portal and PAG portlet necessities, whereas an end user just worries about the Access Grid Meeting itself.

The PAG portlet allows users to use the Access Grid entirely through a portal interface. The user must have Java installed for this to work, but the portlet checks for the availability of Java when it is started, as well as JavaScript, which is also used throughout the program. As shown in Fig 1, the user interface is similar to that provided by Access Grid Toolkit. This decision was made for two reasons; firstly, the project did not have enough resources to

design a new user interface. Secondly, this should help users migrate from using the existing Access Grid Toolkit software to using PAG.

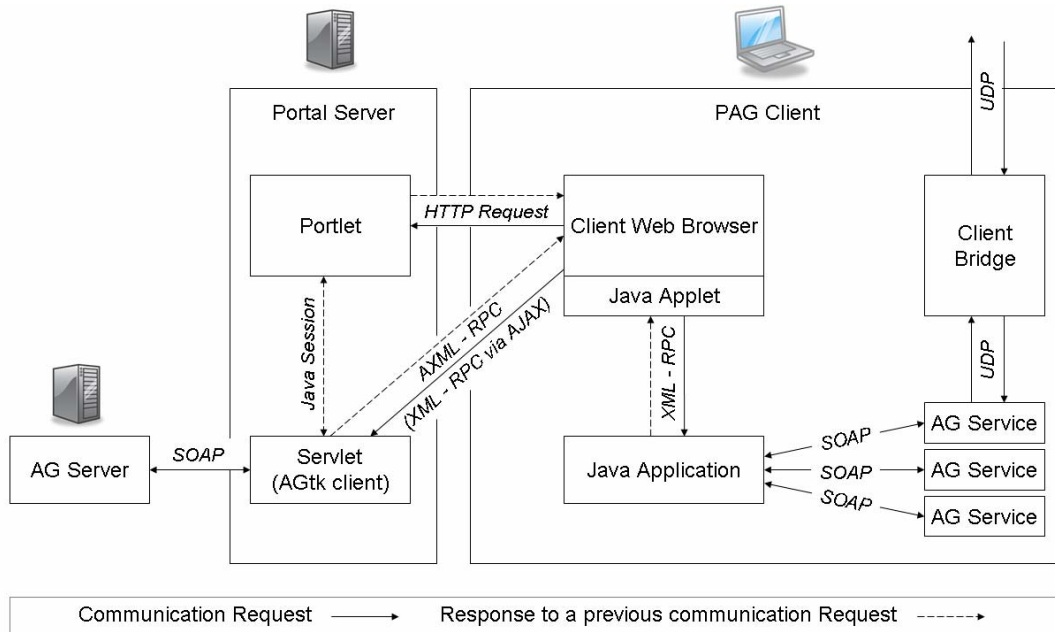
To aid this second goal, PAG has been designed to operate similarly to the Access Grid Toolkit internally. To this end, the portlet contains an implementation of the AGTk node services architecture. The node services in AGTk provide for the tools that allow communication to take place, such as the tools that transmit and receive audio and video. The purpose of the node services architecture is to communicate with these tools so that the data streams from the current Access Grid virtual venue reach the correct tool. The architecture is also designed to allow new tools to be developed and integrated (i.e. 'plugged-in') without any need to change the existing code. This makes the code somewhat dynamic, allowing additional tools to be developed and distributed without any interaction from the original developers or the hosts of the services.

PAG improves upon the node services architecture by allowing one user to add a service that every user of the portal can then use; in AGTk, each user must install the services themselves. Unfortunately, the way in which the architecture has had to be developed in PAG, it is not directly compatible with that in AGTk i.e. you cannot install an AGTk service into PAG without modification. However the intention is to provide a compatibility layer service that allows this interaction to work.

PAG also provides an implementation of the AGTk shared application architecture. The shared applications provide a way of synchronizing data between a number of instances of an application running on multiple clients within a virtual venue, such as a presentation. This behaves differently to the node services, where data is streamed and packets can be lost. In contrast, shared applications have no streaming mechanism, and require a guarantee that the data is received at all clients. Similarly to the node services, the shared applications can be plugged in without a need to recompile the code (or even restart it). Again, PAG improves upon the AGTk mechanism by allowing one user to install an application which all users of the portlet can then use.

In addition to these features that are improvements on the AGTk features, PAG provides some features that have no equivalent in AGTk. The first of these is the pluggable bridge architecture. In Access Grid, all node service traffic is communicated using multicast. In IP version 4, the version that most of the internet is built upon, multicast is an optional protocol and is therefore not guaranteed to be available at all sites. Where it is available, it is also not always reliable, and therefore AGTk provides a way of forwarding traffic without using multicast (i.e. using unicast) to a bridge, which then receives packets from and forwards packets to multicast. In AGTk, the bridging client code is embedded in the main client code, and therefore the types of bridges which AGTk supports are limited to what is provided by the developers. PAG provides a bridge architecture similar to the node services and shared application architectures that allows new bridge clients to be added without needing to change the existing client code. This allows PAG to provide support for new bridging technology that works over low bandwidth connections and from behind firewalls requiring minimal or even no ports to be opened in the firewall. This should allow the Access Grid to be used from various additional locations that would not have been accessible using AGTk.

In order to better support the communication between the bridge technology and the node services, PAG implements client-level bridging. AGTk connects the node services to the venue by passing the multicast addresses of the streams in the venue directly to the services. Most services use command-line parameters to configure the addresses to use, and therefore must be restarted whenever an address changes (e.g. when changing venues or joining a



**Fig 2.** The technical architecture of PAG.

bridge). This means that any configuration to the services, such as laying out the video streams, must be repeated each time the service is restarted. It also means that any bridges must provide unreliable UDP traffic, as this is what the services expect to receive. PAG overcomes these problems by creating a local bridge inside the client which the services then connect to. All traffic to the services is passed through this bridge, so now when changing venues, only the external connection to the bridge needs to be configured; the services can therefore be left connected to the internal side of the bridge. The bridge can also support traffic types that the services do not, such as TCP for reliable transfer of data.

With all traffic flowing through an internal bridge, this bridge can be used to monitor the traffic to determine if the current external connection is reliable. Since the services do not have to be restarted, the external connection can be changed quickly and automatically without user interaction if it is found to be unreliable. This allows PAG to control which bridges are used at any point during a meeting, and so the user does not have to understand the network connectivity issues in order to use Access Grid. The bridge can also filter the packets that are flowing to and from the tools. This means that any invalid packets are not allowed to reach the tools, ensuring smooth operation of the tools. Without this, the tools must be trusted to be able to perform this filtering themselves; experience shows that they often do not.

## Technical Challenges

Most portlets are based on web-friendly content, that is, content that is not dynamically updated without user interaction, and is likely to stay the same across multiple refreshing of the overall portal page. Access Grid therefore presents a new challenge as there is now a need to respond to events that occur remotely, such as a user joining or leaving the virtual venue, without requiring that the whole page is refreshed. Additionally, external tools are required to communicate with the portlet, such as the audio and video tools. This communication must be continuous, even when the page is being refreshed in response to input to another portlet on the same page.

In addition to the portlet related challenges, there are also technical challenges that occur from the Access Grid itself. Users expect a web-based tool to be usable from a browser without any need for network configuration. The AG as it stands can require fairly complex network configurations, especially in relation to firewalls and multicast. Access Grid is also a dynamic application that allows new tools to be plugged-in without changing the base code. As any single PAG user can upload services, applications and bridge clients that all other users of the portal may end up using, there are also security issues to consider.

Fig 2 shows the architecture used in PAG to overcome these challenges. From left to right, the PAG portlet / servlet is hosted on a portal web server (portlets in JSR-168 are hosted as a servlet with extra portlet functionality). This is the sole communication point to the AGTk server; the client does not connect to the AGTk server at all, thus removing any communication problems related to obscure port numbers and proxy servers.

When a user wants to connect to PAG, they use their web browser to talk to the portlet. The pages returned contain a Java Applet which then runs within the web browser. Standard LiveConnect JavaScript is used to talk between the web page and the applet. The JavaScript can then communicate with the portlet using Asynchronous JavaScript and XML (AJAX). This allows parts of the page to be updated without requiring that the whole page be refreshed. The portlet maintains a queue of requests and responses to these requests, and this is polled by the web browser. This means that if the user refreshes the web page (either manually, or by selecting a link within another portlet), the responses to any AJAX requests are not lost, and are simply delivered once the page has been loaded again. This also allows the web browser to display responses to events that occur remotely, such as a user joining or leaving the venue. When such an event occurs, a response is added to the queue, and then when the browser next asks for available responses, this is returned and JavaScript is used to update the user interface.

Whilst the applet that runs on the page is signed, and therefore has full access to the client computer, this cannot be used to run the node services or shared applications architecture. This is because the applet is stopped and restarted every time the user refreshes the page. To overcome this problem, a Java Application is launched by the applet. This application is the main client-side program used by PAG to run and control the node services, shared applications and client bridging. A connection is maintained between this application and the applet; this is reconnected when the applet restarts. If the connection remains disconnected for a nominal amount of time, the application assumes that the browser has been completely closed, and therefore shuts down all the services and applications and disconnects from the Access Grid.

PAG uses client-level bridging to avoid the need for the user to configure their network connection. As previously mentioned, this allows the shared applications to connect to a bridge running within the application, which then in turn forwards the data to and from the external network. This allows the client to test the network conditions on the client side and adjust the network configuration appropriately. It also allows the monitoring of the network throughout the meeting, allowing the client to adjust the network configuration should conditions change.

Security is a very big issue when using the internet and this has been taken into account with PAG, since PAG allows a single user to upload a new service, application or bridge to the portlet, which can then be used by all users. In particular, bridges will be used dynamically without user interaction. This clearly opens up the potential for users to upload malicious applications which will then be distributed and executed by PAG. To avoid this issue PAG

relies heavily on the built-in security in Java, which has been designed with exactly these issues in mind. PAG uses Java Web Start to execute node services and shared applications. There is a requirement that all Java Web Start code is signed with a certificate to enable it to do anything more than basic operations. PAG enforces this further by requiring that uploaded applications be signed, and the certificates are checked at this stage. Java Web Start then prompts the user to accept the certificate (if they have not already done so), and so the user is required to make a conscious decision to allow the code to execute. Bridges similarly are required to be signed, although this is enforced when the bridge code is loaded into the executing Java application (bridges are not separate applications in themselves). Again, the user is asked to accept the certificate when the code is first executed, and the code will not execute if the user does not accept this.

## Availability

The PAG portlet is currently accessible through a test server based at the University of Manchester<sup>6</sup>. This instance is maintained by the developers of PAG and accounts are not required to use it. The PAG team, OMII-UK and NGS are currently planning to make the PAG portlet also available to the e-Science Community via the NGS portal. The e-Social Science community may eventually access the portlet via the NCeSS portal.

For service providers there will be a download available as source and binary packages from OMII-UK. The PAG portlet will be licensed under a BSD style Open Source License<sup>7</sup>.

## AG Toolkit versus PAG

One question that may arise is why one might use PAG instead of Access Grid Toolkit. Although many of these issues have already been covered, it is useful to summarize the advantages that PAG provides.

PAG has been designed to require only that the user has Java installed (and many computers now come with Java pre-installed). In contrast, Access Grid Toolkit requires five different programs to be installed before it can be used. This process does not always go smoothly, and is one of the problems with the use of Access Grid. Even with the all-in-one installer that is now becoming available AGTk cannot be used on a computer where the user has no permission to install applications, such as public use computers.

PAG requires little or no network configuration in order for it to work. As PAG detects the current network conditions, the user should be able to join in a meeting without configuring the client. PAG also supports additional bridge types that allow users to join meetings without needing to open ports in their firewall. AG Toolkit supports only a single bridge type, and this bridge usually requires a large range of ports be opened in the firewall. Additionally, the user must select which bridge they use manually.

Upgrades to PAG are automatically downloaded and installed when the user next uses it. Because Java Web Start is used, changes to the code will be detected and upgraded. With AG Toolkit, the user must manually upgrade their client. This makes changes, both major and minor, difficult to implement, as you cannot be sure that all users will upgrade their clients.

---

<sup>6</sup> <http://memetic.ag.manchester.ac.uk/gridsphere>

<sup>7</sup> *Open Source Initiative OSI - The BSD License*; <http://www.opensource.org/licenses/bsd-license.php>

PAG allows new services, applications and bridges to be added by one user and then used by all. AG Toolkit requires that each user individually install services and applications. The advantage of using PAG here is twofold. Firstly, when using AG Toolkit, one cannot be sure that a shared application or service is available on any particular client, whereas with PAG, if you uploaded the code, you know everyone can use it. Secondly, PAG is self-evolving; no interaction is required on behalf of the host of the portal in order to enact these changes for all clients using the portal.

## User Engagement

Throughout the later development stages users were involved in testing the PAG portlet on the test server. The user groups involved were based in all research backgrounds and were based all over the UK academia. The user groups chosen as early adopters included not only experienced Access Grid users, but also researchers that had not used the Access Grid before. During the development of PAG we had a few key sites as partners that had previously had networking and firewall problems when using the AGTk client.

Many of the issues that users had with the early release version were due to a misunderstanding of what was available in the client. For example, the early release did not feature any network detection; the user had to select which type of network they were on, as with the existing Access Grid Toolkit software. Other issues were related to problems with Java installations. Overall, though, users were surprised that the software appeared to “just work”, and that the audio and video tools did not need to be installed prior to using the software.

For the communication with our users we established two mailing lists.

PAG-USERS@LISTSERV.MANCHESTER.AC.UK is a low volume announcement list to inform users about new versions, or maintenance operations.

PAG-SUPPORT@LISTSERV.MANCHESTER.AC.UK is an open mailing list to report problems with the PAG software or service. The only recipients are the PAG developers.

## Acknowledgments

The work presented here was funded by OMII-UK.

## References

*Welcome to AccessGrid.org;*  
<http://www.accessgrid.org>

*Contributed Software;*  
<http://www-new.mcs.anl.gov/fl/research/accessgrid/wiki/moin.cgi/ContributedSoftware>

*Open Source Initiative OSI - The BSD License;*  
<http://www.opensource.org/licenses/bsd-license.php>

*Portalization Process for the Access Grid*  
*Andrew Rowley, Michael Daw, Anja Le Blanc, Tobias Schiebeck, Martin Turner,*  
*Proceedings UK e-Science All Hands Meeting 2007*

*WSRP and JSR 168: Will These Portal Standards Matter to You?*

<http://www.transformmag.com/web/showArticle.jhtml?articleID=18402895>

*Introduction to JSR 168—The Java Portlet Specification*

[http://developers.sun.com/portalserver/reference/techart/jsr168/pb\\_whitepaper.pdf](http://developers.sun.com/portalserver/reference/techart/jsr168/pb_whitepaper.pdf)

*Introducing Java Portlet Specifications: JSR 168 and JSR 286*

<http://developers.sun.com/portalserver/reference/techart/jsr168/>

*The next-generation IP*

<http://www.networkmagazineindia.com/200203/200203primer.shtml>

*Core JavaScript 1.5 Guide: LiveConnect Overview*

[http://developer.mozilla.org/en/docs/Core\\_JavaScript\\_1.5\\_Guide:LiveConnect\\_Overview](http://developer.mozilla.org/en/docs/Core_JavaScript_1.5_Guide:LiveConnect_Overview)

*AJAX Tutorial*

<http://www.w3schools.com/AJAX/default.asp>

*Access Grid Support Centre – 2006 and 2007 Access Grid Survey Summaries*

<http://www.agsc.ja.net/survey/2006/surveysummary.php>

*Java SE Security*

<http://java.sun.com/javase/technologies/security/>