A Framework for P2P Application Development

James Walkerdine, Lee Melville, Ian Sommerville
Computing Department, Lancaster University, Lancaster, UK
{walkerdi, l.melville, is} @comp.lancs.ac.uk

ABSTRACT

Although Peer-to-Peer (P2P) computing has become increasingly popular over recent years, there still exist only a very small number of application domains that have exploited it. This can be attributed to a number of reasons including the rapid evolution of P2P technologies, coupled with their often-complex nature. This paper describes a framework that has been developed that seeks to aid developers in building P2P applications and therefore cut short development time.

1. Introduction

There is no denying that P2P has become an increasingly popular technology over the last decade. As well as the development of well known P2P applications such as Napster [1], ICQ [2] and MSN Messenger [3], there has also been considerable research within the area. In particular, research has focused on aspects such as improving routing strategies, new overlay structures, and tackling quality of service issues.

Despite these many developments the actual number of application domains that have exploited this technology has remained a niche technology, typically focusing on groupware such as instant messaging, forums, shared workspaces and file sharing. Although P2P technology can obviously provide significant benefits to such domains, it does seem that the potential of P2P is not being fully utilised.

Why P2P technology has not been so readily adopted is most likely due to a number of reasons, in particular the lack of design support for developers of P2P applications and the complexity of the underlying P2P technology. The vast majority of P2P research has focused on the low level aspects of P2P technology. Little work has been carried out on the actual design process for P2P application development. To an extent this has been addressed within the EC funded P2P ARCHITECT [4] project that has developed a methodology, reference architectures, notations and guidelines for P2P application development, with a particular focus on business environments.

Although there has been and continues to be P2P technological developments, they often evolve rapidly or require sufficient understanding of their workings before they can be successfully applied. In our own experiences with Sun's JXTA API [5], we found we needed to gain a clear understanding of the JXTA concepts and workings before we could successfully build a P2P application - a task that involved several months of effort. Similar experiences have also been reported by others [6].

It is not surprising that given the complexity of existing P2P technology and the lack of support for the design process, developers can find the prospect of building P2P applications a daunting task. Ultimately this not only has a detrimental effect on the research area as a whole with its potential exploitation being reduced, but also on its uptake within industrial domains with businesses being less likely to commit the money and effort in order to utilise it.

This paper presents the P2P Application Framework, a model and implementation that seeks to reduce the overhead in developing P2P applications by providing a set of generic and protocol dependant *application* orientated services. As a result it is hoped that this will encourage the rapid development of more applications (potentially in different application domains), and also make the use of P2P technologies more feasible within business environments.

The paper begins by providing an overview of the P2P Application Framework, describing its overall structure and key concepts. A publicly available Java based implementation of the framework is then presented, before finally ending with a discussion of developer's experiences in using the framework. This discussion also demonstrates the framework in use.
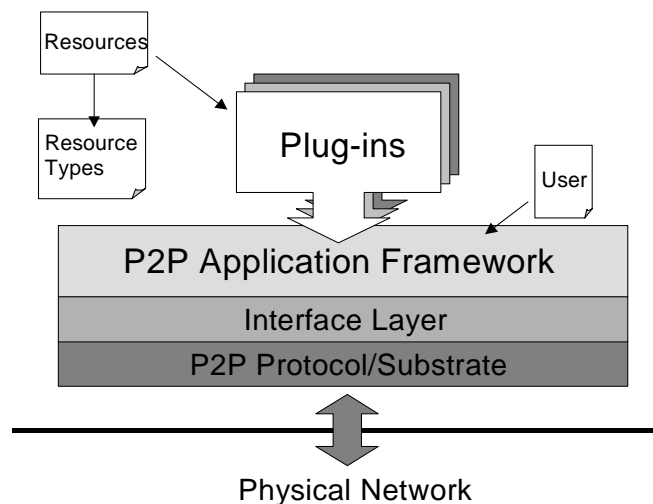
Figure 1 - The P2P Application Framework model

2. The P2P Application Framework

The P2P Application Framework is intended as a mechanism to help developers in building P2P applications. It achieves this by providing in-between layers that further removes the developer from the underlying P2P technology. A consequence of this is that the developers who use the framework do not need to understand how the P2P technology functions, and instead they can focus their time purely on building the applications (referred to as *plug-ins*) that will utilise it. Furthermore, the additional abstraction means that developed plug-ins are independent of the underlying P2P protocol/substrate that is used. So for example, using the framework the same Instant Messenger plug-in could operate over Pastry [7] or JXTA without being changed.

To assist the developer the framework provides a set of commonly used and generically accessible services, for example message communication or resource searching. However, unlike typical middleware that tends to operate at a low level, these services operate at a higher level and are provided specifically for use by GUI styled applications.

The design of the P2P Application Framework is structured around a semi-centralised P2P network model, in that there exists one or more peers within the system that have an increased role and capabilities. Within the P2P Application Framework these nodes essentially act as index peers in a similar vein to those that exist in P2P applications such as Napster or ICQ. Their objective is to support the running of the network (by providing look-up styled services), rather than to control it.

A semi-centralised structure was chosen as it was deemed to be better suited for providing many of the framework services [8]. There is no reason, however, why the framework could not be modified so that it could operate over a fully decentralised approach. It would simply mean that such functionality would have to be spread over all the peers within the network.

Figure 1 shows the structure of an individual peer within the P2P Application Framework model. It should be noted that both standard and index peers possess this structure. A breakdown of this model will now be provided.

P2P Protocol/Substrate - this layer within the model represents the underlying P2P technology that is being used within the system. This could be, for example, JXTA, Pastry or Gnutella [9].

Interface Layer - a key characteristic of the P2P Application Framework is that it is sufficiently generic and abstract that it can be utilised on top of different types of P2P system. In order for the framework to communicate with the underlying technology there needs be a special interface layer that translates the communication between the two. For example, if it is desired for the framework to be built on top of Pastry then an interface layer that is tailored for Pastry would be required to allow the framework to make use of the protocol. The difficulty in building interface layers will be dependent on the individual protocol/substrate, however existing work has already shown such layers can be created for a number of protocol types [10]. As will be discussed later, the implementation of the framework that we have developed is based on JXTA

and so in this case the interface layer supports the communication between the framework and JXTA.

P2P Application Framework - the framework itself sits on top of the Interface Layer and provides the foundation for application development. The framework provides a number of generic services that are accessible to developers.

- *Message Communication* - the framework handles all communication aspects within the P2P system. Developers do not need to worry about how to construct messages specific to the underlying P2P technology or how peers within the network are addressed. Instead messages are simply comprised of attribute name: value pairs (e.g., 'Message: Hello there'), and are sent to targets based on their User ID. If a sending peer does not know the location of a target peer a look up request is made with the index peers. The framework can also deal with communication with a user/peer that is off-line. In this case the message is stored by an index peer and forwarded on the next time the target user/peer comes back on-line.
- *File Sharing* - the framework provides generic facilities for file sharing between peers. Obviously the capabilities of the file sharing support will be dependent on the underlying P2P technology that is used. This means that, depending on the technology, NAT and firewalls may be an issue.
- *Search* - the framework provides generic search facilities that allow plug-ins and users to search the network for resources and other users. These search facilities are tied in with the index peers that exist within the network and operate in a manner similar to applications such as Napster. The index peers maintain a catalogue of the users/resources within the network, and individual peers can interrogate it.
- *Awareness* - the framework provides awareness facilities that allow peers and their respective plug-ins to stay up to date on the status of users and resources they have an interest in [11]. Again this is tied in with the index peers, and can operate in bi-directional manner - the index peers can inform interested peers of status changes, and the peers can also interrogate the index peers.
- *Monitoring* - the framework provides support for monitoring capabilities which allows users/plug-ins to access information about the P2P system. This could, for example, be information about peer communication, peer resources, peer availability or peer location. The network monitoring information is collected by and stored on the index peers, and can be interrogated by individual peers.

- *Favourites List* - the framework is able to maintain a favourites list for each user. This can be used to capture favourite/interested users and also potentially some types of resources (for example, shared disk space). Users can use the contents of the favourites list as shortcuts or as reminders. As well as being held locally a copy of a user's favourites list can also be stored on the index peers.
- *Front-end* - the framework provides a general front-end in which the user is able to manage and activate developed plug-ins, access the frameworks search service, and manage their user details (including favourites list).

User - as with some existing P2P systems such as Instant Messengers (ICQ, MSN Messenger, etc), within the framework it is the *user* rather than the peer that is the unique identity within the system. This means that a user can switch between different peers whilst still maintaining their customisations (such as favourites list, plug-in settings, etc). All users are assigned a unique ID by the index peer, which identifies them within the network

Plug-ins - a key concept within the framework is that of plug-ins. A plug-in can essentially be thought of as a P2P application (such as an instant messenger, file sharing tool, etc) that is built to make use of the P2P Application Framework. Plug-ins draw upon the services provided by the framework in order to carry out their operation. Because a significant amount of the required functionality is already catered for, plug-ins can be developed quickly and typically require less code than their standalone equivalents.

A flexible and independent relationship exists between the plug-ins and framework as a result of a standardised two-stranded communication protocol. This is comprised of a:

- *Common Plug-in Interface* - All plug-ins that wish to be apart of the framework must conform to a specified interface. This ensures that suitable access points exist through which the framework can interrogate the plug-in. Information that the framework may want to have access to could range from the plug-in's name through to what resources it contributes to the network. These access points also provide the means in which the framework passes along messages relevant to the plug-in that it has received. Implementing the interface would typically involve the plug-in providing a set of publicly accessible methods. An example of this is provided later in the paper.
- *P2P Application Framework API* - In order to provide a means for the plug-ins to access the

services that are provided, the framework possess an API. This API provides plug-ins (and their developers) with a simple way to access the frameworks functionality. The scope of the API is obviously dependent on the implementation of the framework but would typically include access to aforementioned services and possibly utility methods to help plug-in developers (for example, Byte to Object conversion methods). Again an example of our implementation's API is provided later in the paper.

Because the relationship between the plug-ins and framework is quite generic and loosely coupled, a fair degree of adaptability is provided for (as will be discussed later).

Plug-ins possess a unique ID that is consistent across the whole system. These IDs are assigned by the index peer during the plug-in's creation and help to determine where messages are from and where they should be routed. For example, a message could be sent to a plug-in (Plug-in ID:2) on a users machine (User ID: 5).

Resources - as with many existing P2P applications, plug-ins are able to contribute resources to the network. For example, a file sharing plug-in may contribute MP3 files to the network. The term 'resource' is used in a very broad sense in that a resource could be, in theory, anything that a plug-in can contribute. Furthermore a contribution does not necessarily need to be a physical entity (for example, a file), but could, for example, also represent the willingness for that peer/user to take part in an activity. For example, the willingness for the user of a peer to take part in a P2P based game. Contributed resources are registered with the index peer so that a consistent and easily searchable network wide catalogue is maintained (c.f. Napster).

Resource Types - to help identify resources within the network, they can be assigned a Resource Type. A Resource Type essentially represents a broad classification of the resource. For example, an MP3 resource could be assigned an 'Audio' Resource Type. In a sense they are similar to Mime Types, although they are not just restricted to file based resources and are more open ended.

By semantically classifying resources in this way it becomes easier for the framework and other plug-ins to identify and utilise the resources that exist within the network. One of the key advantages of such an approach is that it allows the possibility of a resource that is being made available by one plug-in to be then utilised by another plug-in. For example, a file sharing plug-in contributes 'Audio' Resource Type resources and an audio player plug-in plays them.

In order to keep aware of what plug-in:resource type relationships exist, the framework interrogates each plug-in to discover what Resource Types it is interested in. For example, the audio player plug-in would inform the framework that it is interested in 'Audio' Resource Types. Again this provides the framework with a degree of flexibility.

A key factor with the relationships that exists between the plug-ins, its resources and the framework is that of adaptability. Because there exists a generic and loosely coupled connection between these three entities it means that plug-ins and resources can be added and removed from the system at will, with the framework being able to adjust itself accordingly. For example, a user may have located a resource on the network of 'Audio' Resource Type. However, they do not possess a suitable plug-in to handle that type and the framework indicates this to them. After installing the audio player plug-in, the framework will now highlight that this can handle the particular resource.

Our implementation for the framework highlights other ways in which this adaptability can be utilised, and this will be discussed in the following section.

3. Implementing the P2P Application Framework

This section provides a brief overview of our implementation of the P2P Application Framework. Fully working releases have been made publicly available and have been used within a number of projects. Experiences in using the P2P Application Framework are provided later in this paper.

Implementation development work has been spread over a year, building on our existing developments as part of the P2P ARCHITECT project. Within this project we built a simple instant messenger application that ran on top of Sun's JXTA P2P API, in order to gain experience in P2P application development.

Based on our somewhat frustrating experiences with JXTA and the need for an interface layer for the P2P Application Framework, an abstraction was then developed. This captured the commonly used functionality of JXTA, provided support for a semi-centralised architecture structure, and acted as a bridge between the framework and JXTA.

The framework itself was written in Java and makes use of reflection to interrogate the available plug-ins. Again this reinforces an adaptable nature by ensuring that no plug-ins or resources are hardwired into the framework itself. Developed plug-ins are stored within a directory structure that is searched by the framework on initialisation. Any plug-ins it finds are loaded and instantiated via reflection. Currently there exists a single

index peer that makes use of a MySQL database in order to capture details about the state of the network. There is no reason why additional index peers could not be used, although important decisions would need to be made with regards to ensuring database consistency. The incorporation of multiple index peers into our implementation is an area we intend to examine in the future.

The current stable release of the framework possesses all the services that have been previously described, apart from the monitoring support that is to be incorporated into the next release. A number of plug-ins have been developed to test the framework, with more still in development. A summary of these plug-ins will be provided later in the paper.

P2P Application Framework Front-end

It was desired to develop a front-end for the framework that was similar in appearance to many existing instant messenger applications, primarily focusing around a buddy list (a favourites list), but with buttons to access the framework's functionality and installed plug-ins. Figure 2 provides a screenshot of the framework implementation's front-end.

In addition to the features common to instant messenger applications (for example, user awareness within the favourites list), the screenshot also illustrates the plug-ins that have been loaded into the framework, as well as showing one way in which the framework implementation can adapt to the plug-ins. In this
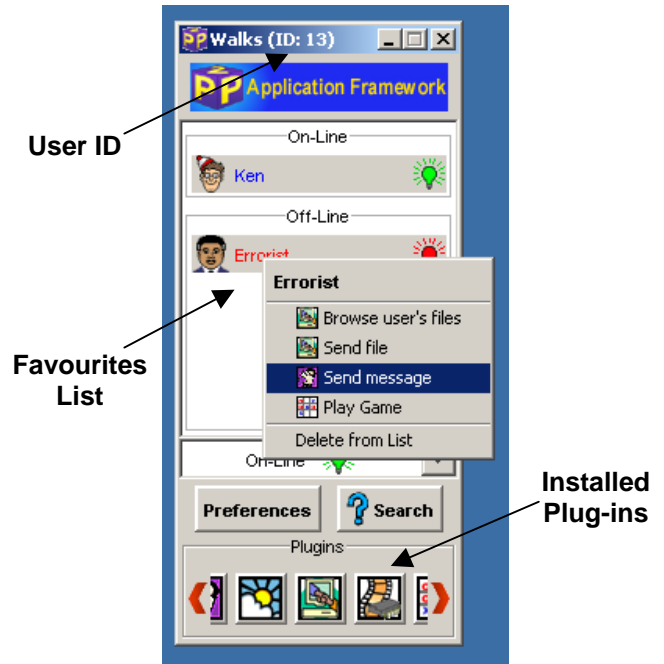
Figure 2 - Front-end of the framework implementation

instance the options that appear in the pop-up menu reflect what is specified by the plug-ins (i.e., whether or not the individual plug-ins desire to add a menu entry). As new plug-ins are added or old ones removed the options within the pop-up menu adapt accordingly.
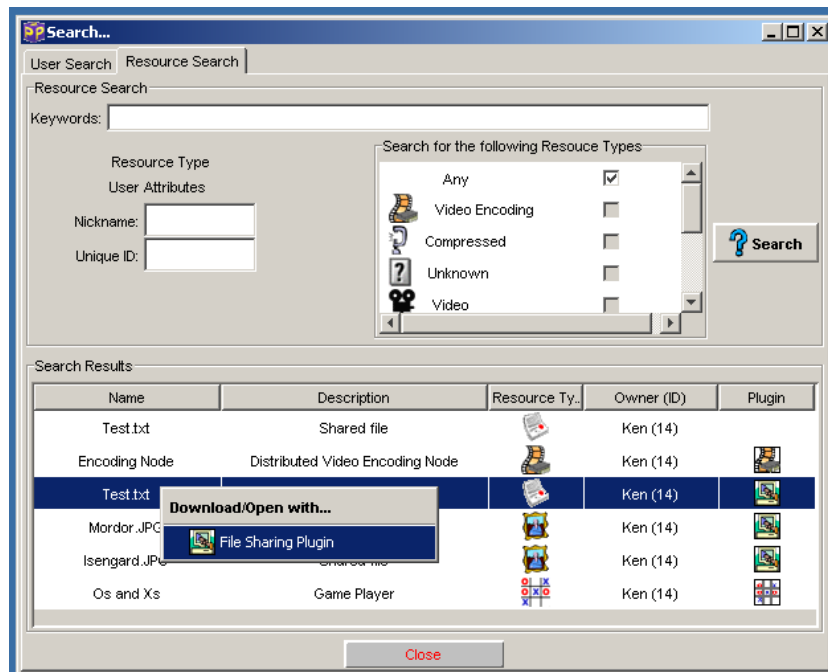

Figure 3 - Searching with the framework

5

Searching with the framework

Our implementation allows both plug-ins and users to search the network for other users and available resources. As previously discussed, when a search is performed a query is sent to the index peer, which interrogates its associated database. Figure 3 provides a screenshot of the implementations search facilities. In this example the user has performed a general resource search and so a list of all currently available resources is returned. As can be seen, the Resource Type for each resource is displayed, along with an indication of the plug-in that is providing the resource (if it is known by the user's framework installation). By selecting a resource the user can bring up a menu that displays all the plug-ins that can access this resource (based on which plug-ins have registered an interest in that Resource Type). Selecting a plug-in would then invoke it with the selected resource. In this example, by selecting the File Sharing plug-in the user can download the file to their peer.

The implementation's API also provides a search method that allows plug-ins to access the frameworks search service transparently to the user.

4. Using the P2P Application Framework

A pre-release of the implemented framework was released to developers in early 2004. This allowed for testing, refinement of the framework API and development to begin on an initial set of plug-ins.

The first full release was made publicly available in April 2004, accompanied with API documentation, user guide and examples. To date six plug-ins have been developed for use with it, with four more currently in development. Not only has the framework been used by colleagues within the department, but also as part of a number of BSc and MSc student projects. Furthermore there has also been interest expressed from external institutions, although currently we are only able to provide limited support. Table 1 summarises (a) currently completed and ongoing plug-in developments, and (b) the Resource Types that have so far been created.

After initial bugs were fixed, feedback from users and plug-in developers' has been particularly positive. Developers have commented that once they have understood how the framework functions, plug-in development has been quick and straightforward. In general developers have found the framework to considerably reduce development time (weeks rather than months). We intend to perform a more quantitative analysis in due course.

In order to demonstrate the use of the framework from the developers' perspective, we will now provide a simple breakdown of the developed Instant Messenger plug-in. This will focus on highlighting the aspects in which the plug-in and framework interact.

Table 1 - (a) Completed and ongoing plug-ins (b) Current Resource Types

Icon	Plug-in	Description
	Instant Messenger	A typical instant messenger application similar to ICQ
	File Sharing	Napster styled file sharing application
	Network Mood	Simple visualisation of overall user 'mood' within the network
	O's and X's Game	Simple P2P based Noughts and Crosses game (or Tic-Tac-Toe)
	Distributed Video Encoder	P2P based video encoding application
	Digital Library	P2P based Digital Library with natural language processing facilities
N/A	Network Visualiser	Provides various types of 2D/3D visualisations of the system
N/A	Intelligent Video Streaming	Streaming video between peers with video quality determined by peer resources (CPU, etc)
N/A	RADP2P	An adaptable P2P system
N/A	Requirements Analysis	A P2P based system for designers to capture and analyse requirements

(a)

Icon	Resource Type	Description
	Video Encoding	Donating CPU resources for video encoding
	Audio	Audio Files
	Compressed	Compressed Files
	Document	Document Files
	Program	Program Files
	Picture	Picture Files
	Video	Video Files
	Unknown	Unrecognised Files
	0's and X's	Availability for a Noughts and Crosses game

(b)

Case Study: Instant Messenger Plug-in

The Instant Messenger plug-in was the first plug-in to be developed for use within the P2P Application Framework. It was intended that it would provide simple text based instant messaging support, though over time this has been extended. The current version of the plug-in allows images and sounds to be also sent within a message (as illustrated in figure 4).

The plug-in interacts with the framework in a number of ways:

- Plug-in activation
- Sending of a message
- Receiving of a message
- User status changes

Activating the plug-in

Although a plug-in can be activated in a number of ways it was decided that it would only be possible to initialise an instant message conversation with users who were currently on your favourites list. Consequently when the plug-in is first initialised and the framework interrogates it, the plug-in informs the framework to add a 'Send Message' entry to the pop-up menu (as illustrated in figure 2). When a user selects this menu entry the framework informs the plug-in which displays a messaging window similar to that shown in figure 4. As this plug-in is not interested in any resources around the network it does not register any Resource Type/Plug-in relationships with the framework.

Sending a message

When a user wishes to send a message the plug-in takes the message contents (text, image, sound) and wraps it up within a Java HashMap object. The HashMap also contains additional details about the sender and target of the message (for example, the plug-ins involved). A method within the framework API is then called passing the HashMap message and User ID of the target as parameters.

```
framework.sendMessage(targetID,
        hashMapMessage);
```
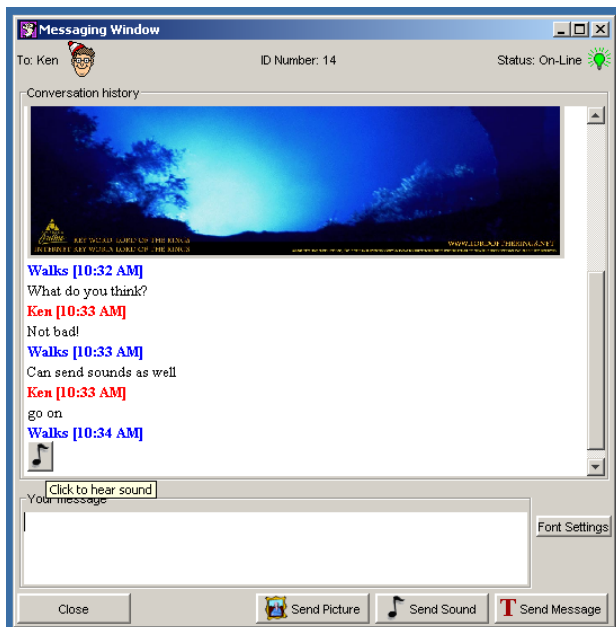
The framework then sends the message to the target.

Figure 4 - The Instant Messenger Plug-in

Receiving a message

When the framework of a peer receives a message it first determines the destination plug-in by looking at the relevant name:value pair within the HashMap message. In this case the framework routes forward any messages containing the ID for the Instant Messenger plug-in. The framework uses reflection to achieve this, as illustrated in Figure 5. The method *'messageArrived'* is part of the common plug-in interface that all plug-ins must implement.

When the Instant Messenger plug-in receives a message it deconstructs it and acts accordingly based on its content. Typically this would involve taking the contents of the message and displaying it within the relevant messaging window.

```
// Obtain Plug-in Class
Class pluginClass = pluginInstance.getClass();

// Specify class of parameters
Class param[] = {java.util.HashMap.class};

// Obtain MessageArrived method from class
Method m = pluginClass.getMethod
("MessageArrived", param);

// Invoke method within plug-in, passing the
message as a parameter
m.invoke(pluginInstance, (Object)
HashMapMessage);
```

Figure 5 - The framework routing a message to the plug-in

Reacting to user status changes

Obviously a user's state can change and other users need to be kept informed of this. When the framework of a peer receives a user status change message (from the index peer) it alerts all installed plug-ins. Again reflection is used to call the *'changeOfPeerStatus'* method within each plug-in.

When this method is called within the Instant Messenger plug-in it updates the status within the relevant messaging window (the status is shown in the top right hand corner).

As can be seen, although the Instant Messenger plug-in and P2P Application Framework only communicate with each other for a few operations, these operations represent a significant part of the plug-ins overall functionality. Because the framework takes care of this functionality the developer's time can instead be spent on developing the higher-level application functionality of the plug-in (such as GUI or providing additional features). The other plug-ins that have so far been developed are similar in structure, using the framework to perform operations where required.

5. Future work

The P2P Application Framework is a work in progress and will continue to be further developed and refined.

An important addition will be the completion of the monitoring support. This is ongoing and is being developed alongside a number of plug-ins that utilise this functionality (the distributed video encoder, network visualiser and the intelligent video streaming plug-ins). Methods to grant plug-ins access to monitoring information will be incorporated into the framework's API.

As has been discussed, there are already a number of ongoing plug-in developments. It is hoped that as the framework matures and is further used, such developments will continue resulting in a broad range of P2P applications. Future possible plug-ins include P2P based database systems and dependability tools that can draw upon monitoring information.

It would also be beneficial to carry out detailed evaluations and testing of the framework. Such evaluations would most likely be qualitative in nature, with user feedback also helping to further refine the framework.

Finally, we shall continue to encourage the uptake of the P2P Application Framework as a general resource for P2P application development. Not only will this be internally as part of research and student projects, but also externally with interested third parties.

6. Conclusions

This paper has presented the P2P Application Framework, a generic and flexible mechanism to assist developers in the building of P2P applications. The framework focuses on lowering development time and seeks to achieve this by reducing the amount of effort developers need to spend on comprehending the underlying P2P technology. Essentially the framework wraps around the underlying technology and provides the developers with a set of generic application centric services. A consequence of the framework is that P2P applications can be more readily developed and, in turn, this will hopefully encourage expansion into new application domains. Furthermore, the additional abstraction allows for applications that have been developed using the framework to be utilised over different underlying P2P protocols/substrates.

Initial releases of the framework's implementation have been made publicly available and have been used within a number of P2P application developments. Currently the framework has and still is being tested and used within a number of research and student projects, covering a number of application domains. Initial experiences have shown that the framework has provided significant benefits to the P2P application development process.

To help illustrate the simple operation of the framework, the paper has also provided a brief breakdown of how the developed Instant Messenger plug-in and framework liases with each other.

The P2P Application Framework is available for download, along with documentation, from our departmental P2P website - ***http://polo.lancs.ac.uk/p2p***

7. Acknowledgements

8. References

[1] Napster. MP3 file sharing application. More information at the URL http://www.napster.com

[2] ICQ. Instant Messenger Application. More information at the URL http://www.iqc.com

[3] MSN Messenger. Instant Messenger Application. More information at http://specials.msn.com/ms/default.asp

[4] P2P ARCHITECT: Ensuring dependability of P2P applications at architectural level, EU Project IST-2001-32708. More information at http://www.atc.gr/p2p_architect

[5] Project JXTA, P2P API, Sun Microsystems Inc. More information can be found at http://www.jxta.org/

[6] Halepovic, E., Deters, R., Building a P2P Forum System with JXTA. *In the proceedings of P2P 2002,* Linkoping, Sweden, 2002.

[7] Rowstron, A., Druschel, P., Pastry: Scalable, distributed object location and routing for large-scale peer-to-peer systems, *IFIP/ACM International Conference on Distributed Systems Platforms (Middleware)*, Heidelberg, Germany, November, 2001, pp. 329-35

[8] Walkerdine, J., Melville, I., Sommervile, I., Dependability Properties of P2P Architectures. *In the proceedings of P2P 2002*, Linkoping, Sweden, 2002.

[9] The Gnutella protocol specification v0.4. Clip2 Distributed Search Services. Available from http://www9.limewire.com/developer/gnutella protocol 0.4.pdf

[10] Coulson, G., Grace, P., Blair, G., Mathy, L., Duce, D., Cooper, C., Yeung, W. K., Cai, W., Towards a component-based middleware framework for configurable and reconfigurable grid computing. *To be presented at ETNGRID-2004*, Italy, June 2004.

[11] Walkerdine, J., Melville, I., Sommerville, I., Designing for Presence within P2P Systems. *Technical Report COMP-003-2004*, Computing Department, Lancaster University, 2004