

.NET and JXTA, and their support for P2P

P2P Architect Project
Lancaster University

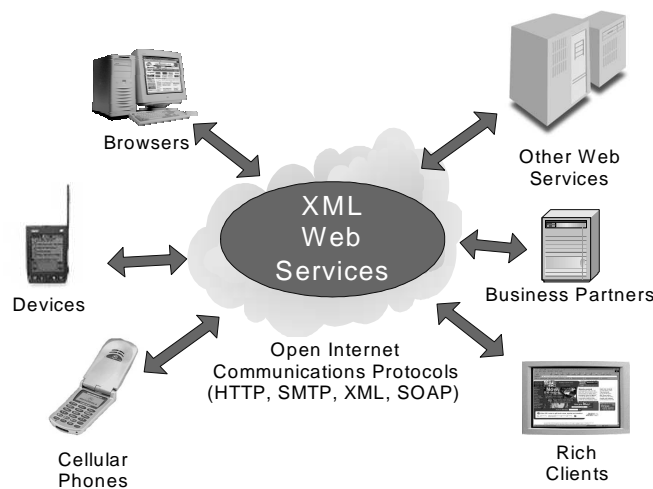
1 Introduction

This document reviews two new networking technologies and analyses the support they provide for the development of Peer-to-Peer applications (P2P). One of these technologies is Microsoft's .NET that aims to revolutionise the Internet by providing a universal interface for resources and services to connect to each other. The other technology is JXTA, an open source API for building P2P applications that is produced by Sun Microsystems Inc. A brief overview of each technology is provided followed by an analysis on its potential for P2P support.

2 .NET

Microsoft's .NET has been developed as a result of two projects. One of these was to improve the Component Object Model (COM)¹, and the other was to deliver software as a service. The combined result has created XML Web Services, a way of allowing applications to communicate with each other using well-defined protocols such as HTML, XML and SOAP. An XML Web service is an application that exposes its functionality programmatically over the Internet or intranets using standard Internet protocols, such as HTTP.

Figure 1 - XML Web Services²



In any P2P network peers must be able to discover and query other peers and also share content with other peers. The .NET framework contains certain characteristics that allow programmers to accomplish such goals.

¹ <http://msdn.microsoft.com>

² Figure copyright of Microsoft

2.1 The .NET Framework

There are three main parts to the .NET framework: the common language runtime, a hierarchy of unified class libraries, and ASP .NET, which is a component-based version of Microsoft's Active Server Pages³.

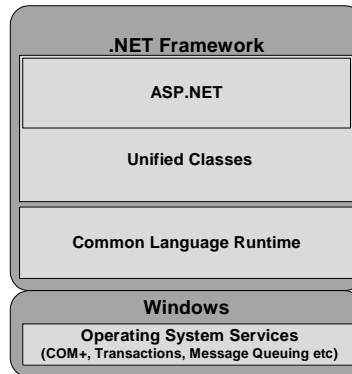
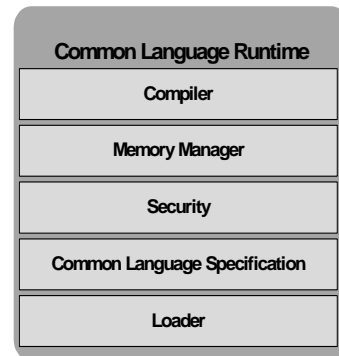


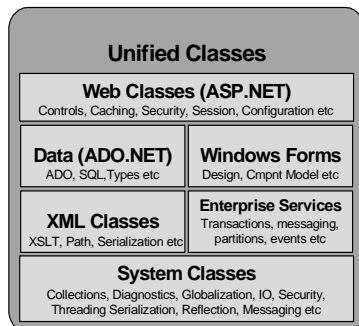
Figure 2 - The .NET framework from the application view⁴

2.1.1 Common Language runtime

The Common Language runtime sits on top of operating system services and is responsible for running any downloaded code. Like the Java Virtual Runtime environment it manages memory, security, garbage collection and so on. The Common language runtime is not directly accessible to programmers; instead they interact with it through a set of classes that lay directly above the Common Language runtime. A JIT compiler is included within the CLR, this allows application source to be downloaded and compiled just prior to being executed.



2.1.2 Unified Classes



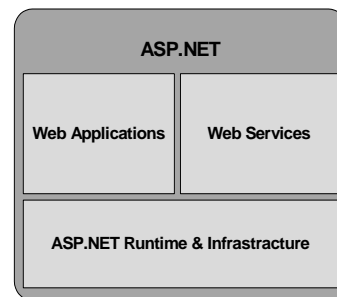
Sitting above the Common language runtime is a hierarchy of object-oriented classes. These are the entry points for utilising the CLR and are what programmers can interface their applications with. A developer can, in theory, use any language he/she wishes, as long as a compiler exists that can integrate the developed source code into something that the unified classes and the CLR can understand.

³ <http://msdn.microsoft.com>

⁴ Figure 2 and other diagrams on this and the following page are copyright of Microsoft

2.1.3 ASP .NET

The heart of ASP .NET is a HTTP runtime execution engine that is responsible for parsing incoming HTTP requests, resolving the location that the requests are intended for, and sending them to the appropriate application that will deal with them.



Its notable features include:

- The exposure of functionality through a standard web protocol.
- Web Services Description Language (WSDL) – a language that is used to describe interfaces that a user can use to build client applications.
- Universal Discovery Description and Integration (UDDI) - enables the registration of web services to allow users to discover them.
- Soap messages are built from XML fragments and are transported using HTML protocols.

SOAP ‘Simple Object Access Protocol’ is an XML based protocol, it allows the exchange of information between any applications or services that can produce or consume SOAP messages. A wide range of applications can take advantage of SOAP, from messaging to RPC style applications.

2.2 .NET support for P2P

Although direct support for the development of P2P applications is not provided, elements of the .NET API can be used for P2P application programming. In particular, the .NET API has class namespaces that can be utilised for P2P development:

System.Net namespace

This namespace has a set of classes for integrating Internet protocols into an application. It uses a layered approach so varying levels of control can be added to an application from pure socket control to general request/response actions.

System.Web.Services namespace

Here web services can be created and exposed to the outside world. A web service in the .NET framework is a programmable application logic, that can be accessed with SOAP messages. Peers connecting to the service only need to know how to send and receive SOAP messages.

System.WinForms namespace

System.WinForms is the framework for building GUI components that can be added to a P2P application.

2.3 Discussion

With .NET Microsoft have created a loosely coupled asynchronous message passing environment on which to inter-connect remote applications. Furthermore, Microsoft has extended this with a range of other features that build on the .NET architecture. For instance, the .NET My Services platform (also known as “Hailstorm”) is a user-centric application development platform. At an abstract level it allows the creation of private and secure digital safety deposit boxes where individuals can hold their credentials and access them from any location.

The Microsoft .NET framework does not have any specific peer-to-peer capabilities such as the JXTA API (described below). It is not designed simply for building P2P networks, however P2P applications and their network support can be built with the .NET framework (although this will require additional work from the developers). Much like JXTA and the Java layers underneath it .NET has been designed to run on any platform that has a “Common Language Runtime” and it’s associated JIT compiler. This leaves developers open to write applications in the language of their choice, for the platform of their choice, and still allow interoperability with other platforms.

Security, regardless of what review literature may say, is strongly considered by Microsoft, and this can be seen with the .NET framework and its support for user centricity. This certainly seems to be an evolution of Microsoft’s passport service.

Similar to JXTA the Microsoft’s .NET platform is still under construction but it does look like a real contender in the P2P revolution.

3 JXTA

JXTA is a set of generalised peer-to-peer (P2P) protocols that have been developed by Sun Microsystems⁵. The intention of these protocols is to allow any device that is available on the network to communicate and collaborate as peers. Within the last seven months the JXTA project has become open source and a host of people and institutions from around the world now contribute to its development.

One of the main objectives of JXTA is to provide a platform that supports the development of P2P systems and provides much of the basic networking functionality that is required in such development. In particular JXTA aims to achieve:

- *Interoperability* – JXTA is designed to enable peers to easily locate each other, communicate to each other and offer services to each other and to achieve this transparently across different networks and communities.
- *Platform independence* – JXTA is designed to operate independently of programming languages (such as C or Java), system platforms (such as Microsoft Windows and UNIX), and networking platforms (such as TCP/IP or Bluetooth)
- *Ubiquity* – JXTA is designed to be implementable on any device ‘with a digital heart beat’. This can include sensors, consumer electronics, PDA’s, network routers, desktop computers and storage systems.

3.1 Architecture overview

JXTA is designed to provide a layer on top of which services and applications can be built. This layer makes available to a developed application a range of sophisticated P2P tools that handle issues such as communication between peers and security, whilst at the same time maintaining interoperability by being thin and small. To achieve this the designers have further split the P2P software stack into three key layers (summarised in Figure 1):

JXTA Core

The core layer, which resides at the bottom of the P2P software stack, deals with the creation of peers and peer groups, ensures security within the network and manages the communication between peers (for example, routing).

JXTA Services

⁵ More information can be found at <http://www.jxta.org>

The services layer builds on top of the core layer and provides services such as indexing, searching and file sharing. Typically these services make up components of an overall P2P system.

JXTA Applications

The application layer represents the actual applications that have been built to make use of the JXTA P2P services. Such applications could be messaging or document management systems.

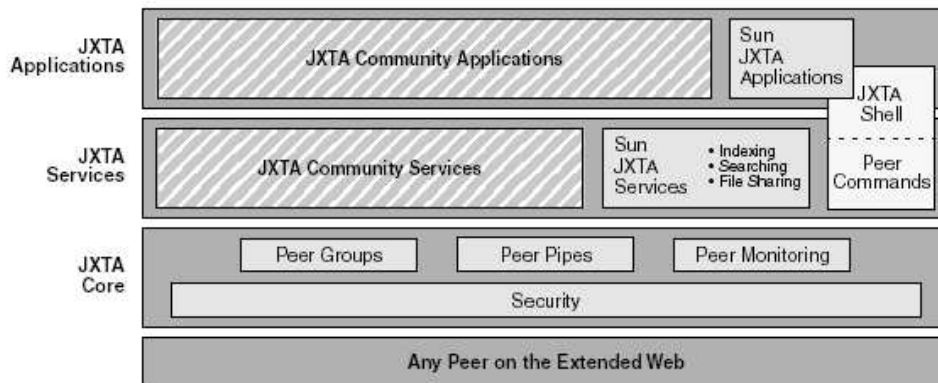


Figure 3 - JXTA Architecture⁶

Although JXTA has been split into three layers some features, such as security, can play a part in all layers and throughout a P2P system.

3.2 Supporting communication between peers

To support communication between peers JXTA makes use of *pipes* and *advertisements*. When a peer connects to the JXTA network it publishes an advertisement that represents a summary of the peer (name, location, etc) and what services it can offer. In order for one peer to communicate with another it first needs to obtain the advertisement for the target peer. JXTA provides a discovery service that trawls through the P2P network to locate an advert. Once obtained it can then setup a pipe connection between the two peers.

A pipe is a communication channel that can be used for sending and receiving messages in an asynchronous fashion. Pipes are uni-directional, in that messages can only be passed down them in one direction, and are also virtual in that they can be created and destroyed at whim and do not belong to any one peer.

Within JXTA, pipes form the basis of all communication between peers (on the application level). Currently they are intentionally unreliable in order to provide the lowest overhead. Overtime additional properties such as reliability, security and quality of service will be built into them.

3.3 Discussion

JXTA has the potential to become a powerful API for use in the development of P2P applications. The fact that it is free and open source means that a vast number of developers can (and already do) contribute to its rapid and continuous development. However, being open source also introduces a host of difficulties.

⁶ Figure copyright of Sun Microsystems, Inc

Although the JXTA project is managed, due to the openness of its development enforcing strict control is difficult. The very nature of JXTA means that its development has to be spread over many machines. These machines can be located all over the world, be operated by individuals or companies, and be operational at irregular times of the day. These factors can have a dramatic effect on the systems reliability. For example, by a peer removing itself from the JXTA network any other peers who used it as a connection point to the network will suddenly find themselves isolated from the network (partitioned).

The current version of JXTA also provides very simplistic security support. Although the developers readily acknowledge this, if JXTA is to become a viable option (especially for businesses) then this needs to be addressed. It is indicated on the JXTA website that an improved security protocol is being developed.

Despite these concerns the JXTA API is a flexible and powerful package that can be used as a basis for P2P application development. Furthermore it is still at a relatively early stage in its development and a lot of the issues raised above will be addressed.

4 Summary

This document has provided a brief overview of two technologies that can play a significant role in future P2P developments. .NET is a more general networking initiative developed by Microsoft that can be used as a basis for P2P applications. Sun's JXTA is a specific P2P API that takes care of much of the networking and management issues that a P2P application would have to deal with.

These two technologies have been discussed with regards to their applicability for P2P systems. A summary of identified issues is provided in Table 1.

Technology	Advantages	Disadvantages
.NET	Platform independent. Language independent. Allows internet based applications to be integrated to offer a wide range of services. Uses an intermediate language that is translated to native code at runtime.	Not specifically created for P2P development.
JXTA	Free and open source. Platform independent Provides specific support for P2P development. On going project with a large development base – API being constantly updated.	Not strictly managed due to its open source nature, consequently this has an effect on its reliability and availability. Lacks decent security mechanisms Currently only supports the Java programming language

Table 1- Comparison of the two technologies