

# ENABLING FLEXIBLE SERVICES USING XML METADATA

Luis Velasco, Ian Marshall

BT-Labs, Martlesham Heath, SUFFOLK, IP5 3RE, UK

[velascol@drake.bt.co.uk](mailto:velascol@drake.bt.co.uk), [marshall@drake.bt.co.uk](mailto:marshall@drake.bt.co.uk)

**Abstract.** Combining eXtensible Markup Language (XML) and Active Layer Networking may yield strong benefits for networked services. A Wide range of new Multimedia applications can be developed based on the flexibility of XML and the richness of expression afforded by metadata. XML is an emerging technology that will constitute the foundation of an Object Oriented Web. A system of network intermediaries based on caches, which are also active and driven by XML metadata statements, is described.

**Keywords** XML, Cache, Active Node, Proxylet, HTTP

## 1 INTRODUCTION

The characteristics and behaviour of future network traffic will be different from the traffic observed today, generating new requirements for network operators. Voice traffic will become another form of data, most users will be mobile, the amount of traffic generated by machines will exceed that produced by humans, and the data traffic will be dominated by multimedia content. In the medium term the predominant multimedia network application will probably be based around electronic commerce capabilities. Operators will therefore need to provide a low cost service, which offers an advanced global trading environment for buyers and sellers of any commodity. The e-trading environment will be equipped with all the instruments to support the provision of a trusted trading space. Most important is the ability to support secure transactions over both fixed and mobile networks. Networks will thus need to be robust, contain built in security features and sufficiently flexible to address rapidly evolving demands as other unforeseen applications become predominant.

## 2 MOTIVATION

Existing networks are very expensive, and the deployment of new communication services is currently restricted by slow standardisation, the difficulties of integrating systems based on new technology with existing systems, and the overall system complexity. The biggest cost is management. The network of the future will need to be kept as simple as possible by using as few elements as possible, removing duplication of management overheads, minimising signalling, and moving towards a hands off network.

The simplest (and cheapest) current networks are multiservice networks based on powerful ATM or IP switches. New transport networks are designed on the basis that nearly all applications will eventually use internet-like connectionless protocols. The difficulties of adding services such as multicast and QoS to the current internet demonstrate that even these simpler IP based networks will require additional mechanisms to enhance service flexibility. The simple transport network will thus need a flexible service surround. The service surround will provide a trusted environment, with security features (for users, applications and hosts), QoS support, application specific routing, automatic registration and upgrade for devices connected to the transport network. It will also enable Network Computing facilities such as secure gateways, application layer routers, cache/storage facilities, transcoders, transaction monitors and message queues, directories, profile and policy handlers. Such a service surround will likely be based on some form of distributed middleware, enabling the features to be modular and interoperable. The service surround must enable rapid introduction of new features by the operator. In order to minimise the management overhead clients will directly control which features should be used for a particular session, without operator intervention. The network will thus need to know nothing of the semantics of the session. To achieve this a middleware based on some form of active services or active networks will be required.

### 2.1 Active Networking Proposals

Active networking was originally [TENN] a proposal to increase flexibility by adding programmes to packet headers which run on forwarding devices that lie in the path of the packet. However service operators are unlikely to permit third party programmes to run on their equipment without prior testing, and a strong guarantee that the programme will not degrade performance for other users. Since it will be extremely hard to create interesting programmes in a language, which is simple enough to enable termination guarantees, we regard this approach as unrealistic.

A somewhat different flavour of active networking, in which the packets do not carry programmes but flags indicating the desirability of running a programme, has also been proposed [ALEX] based on a dynamic interpretation of programmable network ideas. This would enable the service operator to choose a programme to load, which matches the indicated need, but is sourced from his own database of tested programmes. We believe that this second approach may be valuable in the long term, however in order not to degrade router performance the number of flags will need to be small and the number of possible programmes will thus also be small. In

addition it will not resolve the immediate need for increased flexibility, as it will require standardisation, which will take time.

We have proposed a third alternative, which we call application layer networking [ALAN]. A similar proposal [AMIR] was described as active services. In this system the network is populated with application level entities we refer to as service nodes, or dynamic proxy servers. These could be thought of as equivalent to the HTTP (HyperText Transfer Protocol) caches currently deployed around the Internet, but with a hugely increased and more dynamic set of capabilities. This approach relies on redirecting selected packets into the application layer device, where programmes can be run to modify the content, or the communication mechanisms. The programmes can be selected from a trusted data source (which is normally a cache and has minimal restoration overhead), and can be run without impacting router performance or requiring operator intervention. Packets are redirected on a session by session and protocol by protocol basis, so there is no need for additional flags or standardisation. Programmes are chosen using the mime type of the content (in the application layer header), so again no additional data or standards are required.

There is a further proposal [PEI98] that allows servers to supply cache applets attached to documents, and requires proxies to invoke the cache applets. Although this provides a great deal of flexibility, it lacks important features like a knowledge sharing system among the nodes of the network (It only allows interaction between the applets placed in the same page). The functionality is also severely restricted by the limited tags available in HTML (HyperText Markup Language). Most importantly the applets do not necessarily come from a trusted source, and clients do not have the option of invoking applets from 3<sup>rd</sup> party servers. Which may be trusted.

Using mime-types (as in ALAN) provides more flexibility than HTML tags, but still restricts the range of applications that can be specified by content providers, as, different operations are often required for content with identical mime types. It is therefore necessary to find a better way to specify new services. XML [WOO99] provides a very promising solution, since the tags are extensible and authors can embed many different types of objects and entities inside a single XML document. In this paper we present a design for a modified ALAN based on XML in order to enable an object-oriented approach to Web information [MAN98]. This will show how a range of services could be implemented. We demonstrate feasibility by implementing and measuring a simple example service. A brief introduction to XML is provided in Appendix 3 for those who are new to this technology.

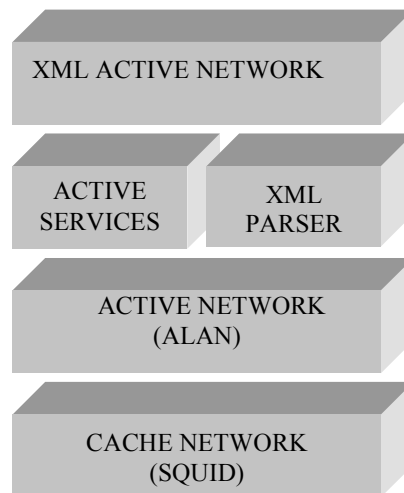
### **3 ALAN & XML**

Our design built in several layers based on existing technology:

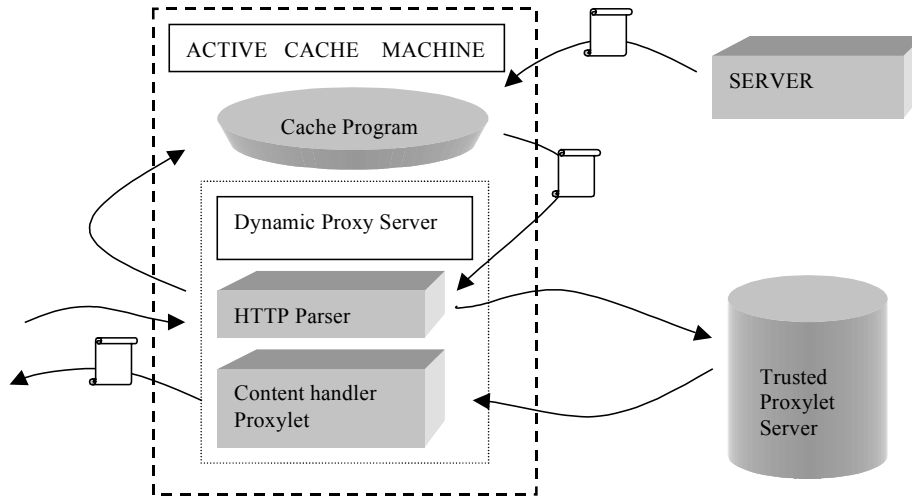
The ALAN Platform is a Java RMI based system built by the University of Technology (Sydney) in collaboration with BT-Labs to host active services. It provides a host program (Dynamic Proxy Server) that will dynamically load other classes (Proxylets) that are defined with the following interface: Load, Start, Modify, Stop [ALAN].

The platform provides a proxylet that analyses the HTTP headers and extracts the mime-types of the objects passing through the machine (HTTP Parser). After determining the mime-type of the object, the program chooses a content handler, downloads the appropriate proxylet from a trusted host to handle that mime-type, and starts the proxylet with several parameters extracted by the HTTP parser. Using this model, a wide range of interesting services can be provided. However, this original model cannot support the whole range of services we plan to implement. There is a need for additional data (not included in the HTTP headers) to manage interoperability among the services and to expand the flexibility and range of applications that can be developed. XML provides a mechanism to implement these improvements and appears a perfect complement to the architecture [MORG99].

We have built a simple XML Parser in Java that will work in collaboration with an HTTP parser to utilise all the Metadata needed in the active applications. The HTTP Parser provided by the University of Technology (Sydney) has been completely rewritten in order to integrate the XML parser seamlessly into the processing.



**Fig. 1.** shows the architecture or the prototype. The first layer is the cache network. For the prototype, we have used squid v1.19 [WESS97] for the cache. The second layer and the upper layers constitute the core of our system and will be discussed thoroughly within this paper. An Application Layer Active Network Platform (ALAN) implements the active services. One of these services is an XML parser that provides the functionality to handle the active objects and Metadata.



**Fig. 2.** Functionality of an active Node. The functionality of an active node (figure 2) is described as follows. Upon the arrival of an object into the node, the HTTP parser examines the header and gets its corresponding Mime-Type. If the object is an XML object, then the XML Parser is called and it will extract the Meta-Data. The metadata specifies which proxylets should be invoked, in which order, with which parameters, and under what circumstances. The parser then makes the appropriate calls to the DPS, which loads the proxylets.

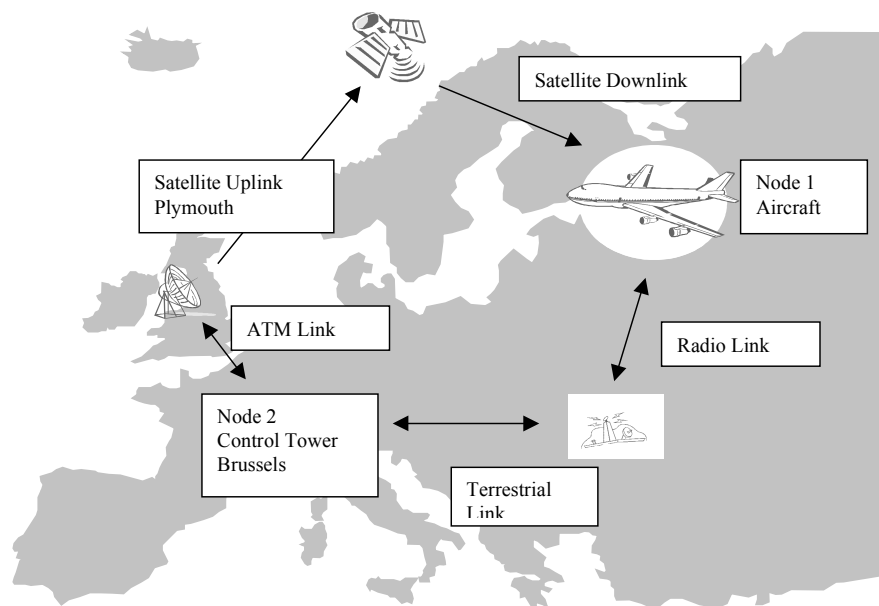
## 4 MULTIMEDIA SERVICES ENHANCED BY THE ACTIVE NETWORK

We provide a set of examples where active services can be applied to give new functionality and versatility, together with outline designs for each service.

### 4.1 Alternate Path Routing for QoS.

In order to deploy Multimedia Applications over Internet, it is necessary to establish the mechanisms needed to guarantee QoS. QoS-based routing has been recognised as a missing piece in the evolution of QoS-based service offerings in the Internet. Alternate Path Routing algorithms allows the network, or end systems, to determine a path that supports the QoS needs of one or more flows across the network. Current Internet routing protocols, e.g. OSPF, RIP, use "shortest path routing", i.e. routing that is optimised for a single arbitrary metric, administrative weight or hop count. These routing protocols are also "opportunistic," using the current shortest path or route to a destination [CRAW98]. Our aim is to enable route decisions to be made on multiple metrics and fixed for the duration of the flow. For example a node could

have a connection via landline with low latency, low bandwidth and high security, and a satellite connection with high bandwidth, high latency and low security. The choice of best route will be application, user and context specific. The XML Metadata is rich enough to express the application layer requirements and force a correct choice. Using our scheme it also allows return traffic to be correctly routed using an appropriate proxylet at the head end of the satellite link. As part of our research program we plan to apply application layer active networking in a scenario like the one presented in Figure 3. The aim is to show how mobile multimedia applications can be deployed.



**Fig. 3.** The aircraft application is an excellent example that shows diverse communication paths based on different physical layers: Aircrafts (Node 1) will have a radio based bi-directional link to the control tower. Meanwhile, the control tower can use COIAS wide area connectivity to establish a high bandwidth link to the aircrafts using ATM and Satellite combined. Path selection is performed at the Control Tower by examining the meta-information of the packets, deciding which is the most appropriate path and adding security if needed. Users can use meta-information to mark their packets with their needs of bandwidth, latency, needed degree of security and needed degree of guarantee. This meta-information will help the system to classify the packets so a smart path selection. By doing this, the packets will be routed using the optimum path.

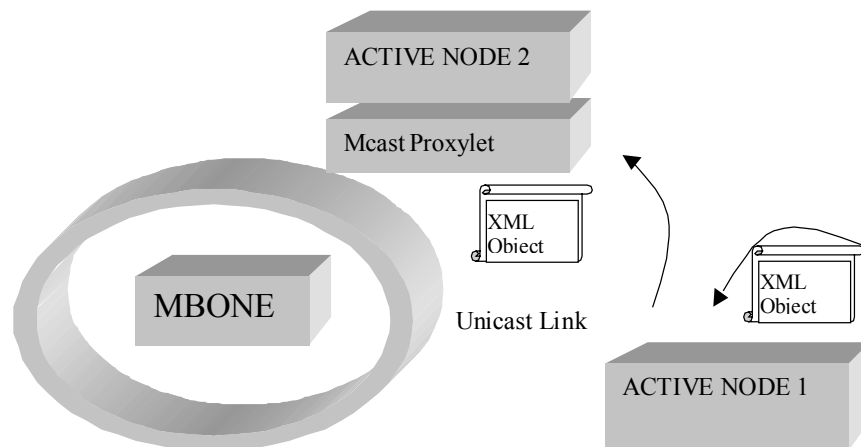
The design of the system is quite straightforward and involves tunnelling content between active nodes/end systems. By pushing an XML object, the source machine is able to provide the information needed to make correct routing decisions. The XML Object will be parsed, at any active nodes in the path, and the corresponding router

proxylet will be downloaded and executed. A tunnel proxylet will then choose the best available route to a point near the client, which can support the remote end of the tunnel. Any downstream active nodes can obviously perform further local route optimisations in a similar manner

With respect to the XML object, it is quite straightforward to produce it from the template provided in Appendix 2. We need to change the proxylet to be loaded and provide the arguments relevant to the QoS required (basically maximum RTT and minimum Bandwidth, some security specifications and the IP address of the client requiring the service). Then the router proxylet will be able to check the available possibilities and choose the most optimum path.

## 4.2 Multicast over unicast lines

Using multicast carries some disadvantages. The main disadvantage is that many routers don't support it yet. As a consequence, people built *multicast islands* in their local networks [ERIK93]. Then, they discovered that it was difficult to communicate with people doing similar things because if only one of the routers between them didn't support multicast it was impossible to establish the multicast connection.



**Fig. 4.** Briefly, when an active node (Node 1) receives an XML-based request demanding the multicast service over a unicast link, it will download a multicast reflector proxylet. The Proxylet will obtain information about possible active nodes connected to the Mbone. With that information, the proxylet will be able to select an active node (Node 2) from the list and to forward a modified XML object demanding a reflection tunnel.

To overcome the difficulty of using Multicast over unicast lines, it has been shown [ALAN] that running a reflector (as shown in figure 4) at each end of a unicast link to tunnel the multicast over unicast is useful. Our system provides the same facility for XML based multicast, with the advantage that we can specify conditions requiring

multicast in the Metadata rather than assuming multicast is always required for a particular mime type as in ALAN.

When the active node connected to the Mbone (Node 2) receives the XML object sent by the first node (Node 1), it will parse it and download a proxylet that will reflect the desired Multicast data onto the Mbone. Node 2 is thus a proxy for a multicast address, which actually originates at node 1. Reversing the order makes node 1 the proxy able to perform a local multicast of Mbone material.

The XML object needed to transmit the Metadata is very easy to build. Basically, we have to adapt the template in Appendix 2 by changing the name of the proxylet and the arguments we need to pass to it. In this case, the multicast group to be subscribed, the type of information that is going to be multicast (STREAM for example), the IP address of an active node connected to Mbone and the reflector proxylet that is required.

### **4.3 COOPERATIVE PROCESSING**

There are several jobs and services, offered in the network that are carried out quite slowly. This is due to servers that cannot cope with the demand and transferring part of the load to other machines can help. This requires co-operation between nodes, which can take the form indicated by policies expressed in the XML metadata. Nodes need to have some knowledge of other nodes and this too can be conveyed in metadata or can be obtained by other means

An XML object specifying the different tasks to be performed by the nodes is sent to them. This XML object will transmit the semantics needed to establish the collaboration and interaction policies that are going to apply to rule the distributed process uses this approach. The active layer network can produce a distributed computation scheme that may help solve complex calculus. Search engines are an interesting example because they rely on physically distributed data, but physical location of an item needs to be computed in the shortest possible time to give a quick response.

The main XML object will be composed of a hierarchy of several objects like the one described in the Appendix 2. Each embedded XML will contain the needed proxylet and the task to perform and the active node where it must be used. In addition, it would contain metadata describing how to implement the interaction among the nodes.

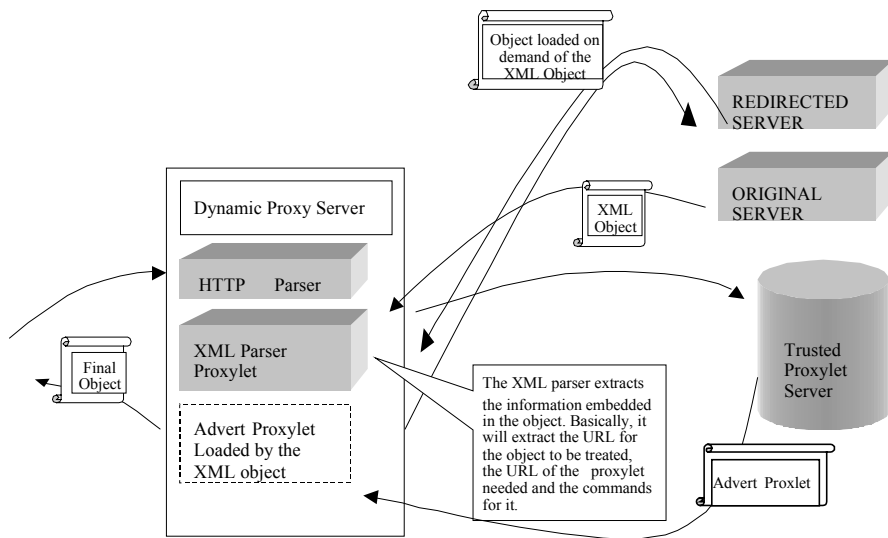
### **4.4 ADVERT ROTATION**

Studies show that among the most popular dynamic contents are advert banners that are dynamically rotated so the same HTML page can show different adverts for each request. These dynamically created HTML pages are just slight modifications of an original template page. The changes usually consist of sets of graphics of the same size that will appear consecutively in the same position of the page. To achieve this, the server executes a cgi program that generates the HTML text dynamically. This

dynamic behaviour tends to make this content un-cacheable. It is preferable to make simple dynamic pages containing rotating banners cacheable since this will;

- a) Allow a distributed service and eliminate the critical failure points.
- b) Improve the usage of the existing bandwidth.

This task requires Meta-Data that is not provided in the HTTP headers. XML will enable information like the rotation policy, the adverts that are going to be rotated, and perhaps a watermark to protect copyright, to be embedded.

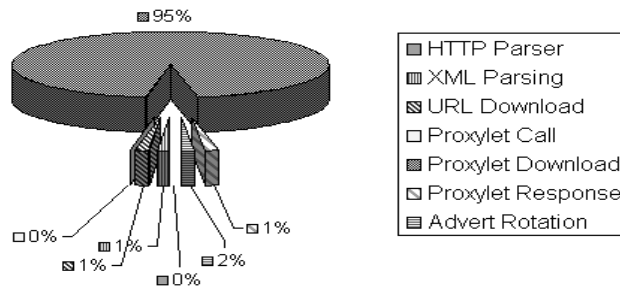


**Fig. 5.** As the object is requested and passes through the active node, it will be parsed in the http parser and then by the XML Parser. This analysis will extract the generic html page that is going to serve as a static template for the rotated adverts, the list of images which should be rotated and the rotation policy. The information is used as a parameter list in the invocation of a rotator proxylet, which will download the objects as needed. Subsequent requests for the page will be passed to the proxylet by the cache at the active node, and the proxylet will execute the rotation policy. The complete XML specification is provided in Appendix 2.

## 5 IMPLEMENTATION OF AN EXAMPLE & RESULTS

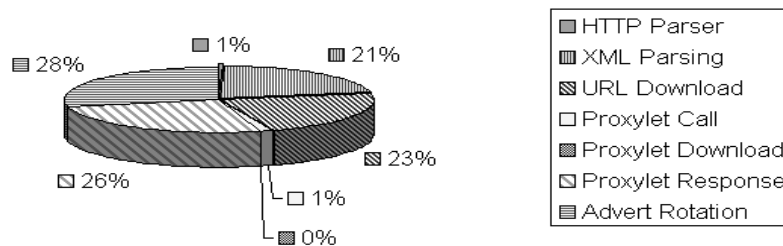
In order to get some preliminary performance measurements of the architecture we implemented the advert rotator example. The objectives are: Demonstrate the feasibility of the model and show the weak parts of the implementation in order to improve releases in the future.

### First Request Time Distribution



**Fig. 6.** The graphic shows the average distribution of times during the first ten experiments. It appears that most of the time is spent downloading and starting a new service (setting up a new proxylet in the node).

### Further Requests Time Distribution



**Fig. 7.** Shows that once the active service is set up, we can use it for further requests and avoid that undesirable delay. Caching can perform a very important role in active networks.

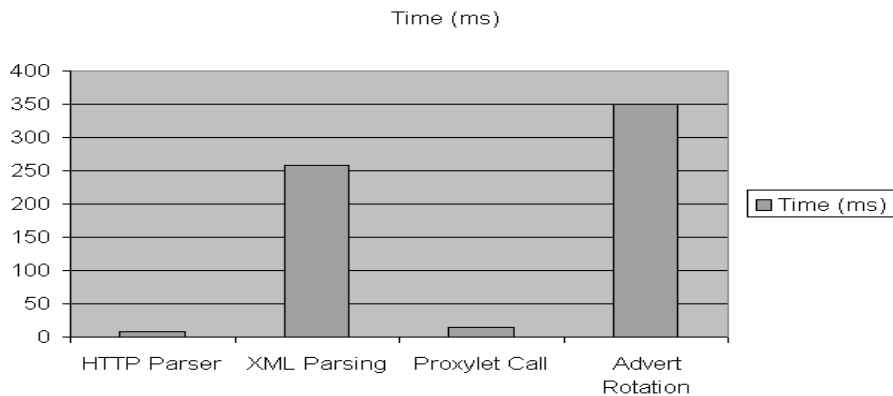
Our experiment consisted of running an active node on a Sun Sparc Station 10 with 64 MB running SunOS Release 5.5.1. The Java version for both programming and

testing was JDK 1.1.6. The active node program was started and was already running the Proxylets needed for HTTP and XML Parsing.

We conducted 20 experiments. For the first ten, the whole process ran each time a new advert rotation was requested, the advert proxylet was loaded. The subsequent ten utilised caching. When a new request arrived, a proxylet that was already running was used. We measured the times needed to accomplish the different tasks. The numerical results of these experiments are shown in the graph below.

The proportional results are illustrated in the figures 6 and 7. The analysis tries to show the times needed to perform the processes and tasks during the normal operation of the system. The functionality of these processes is described as follows:

1. HTTP Parsing. Time needed to determine analyse the HTTP header and determine the Mime-Type.
2. XML Parsing. Time needed to get the XML Object, parse it and extract all the Metadata Embedded.
3. URL Download. Time needed to download the HTML.
4. Proxylet Call. Time need to generate the query to load the Proxylet in our Active Node.
5. Proxylet Download. Time needed to download and start the proxylet; it requires a lot of time because of the ALAN platform design.
6. Advert Rotation. Time needed to perform the demanded task. In this case the advert rotation.



**Fig. 8.** The graphic in figure 8 shows the results of the experiments when the service was cached. The proportion of time spent downloading the proxy has disappeared. The URL download time can vary depending the object to be downloaded and the bandwidth to the server. In our testing, all the objects are available in our LAN so we can expect greater values for this part of the process in wide area tests. However this increment will only be important for the first request, thereafter the URL object is cached and is made locally available.

The most important variable is the times due to the additional processing of the proxylets. It appears that the XML-Parse Proxylet and the Advert Rotator Proxyler are

taking most of the time. Nevertheless the total delay is below one second. We can expect better results if a faster computer is used as a server with a non-interpreted language. However the purpose of this paper was to demonstrate the feasibility of active caching nodes based on XML and throughput was not a priority. This prototype shows that it is possible to provide active services with delays of just several hundred milliseconds.

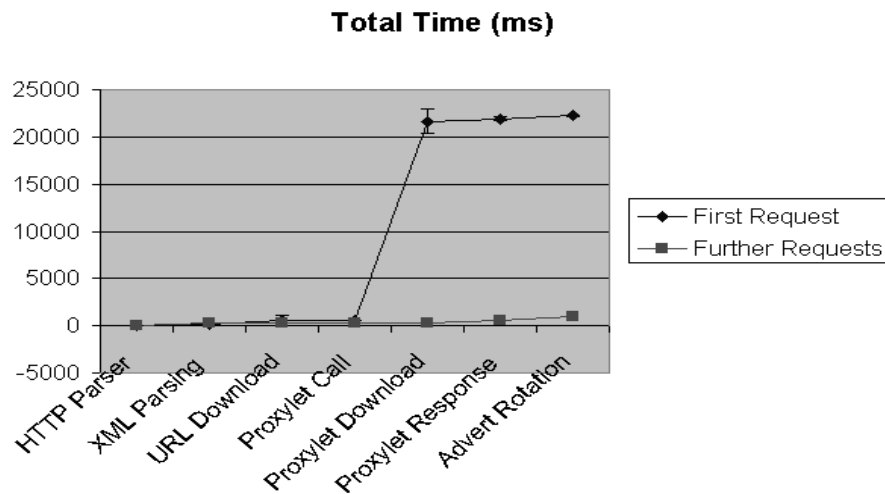


Fig. 9. illustrates the difference in the delay between the experiments that needed the proxylet to be downloaded and started and the experiments where the proxylet was already downloaded and running in the active node.

## 6 FUTURE WORK

In the immediate future we intend to build and test all the services outlined in this paper. In addition there are three major longer-term issues, which we are concentrating our efforts on solving;

It would be beneficial to specify behaviour and other non-functional aspects of the programmes requested in the metadata, using a typesafe specification language. One possibility is to use SPIN], which is c-like and has some useful performance and time related primitives. The use of a language of this kind would provide greater flexibility and interoperability, and go some way towards solving our second issue.

For complex services it will be necessary to invoke several proxylets. At present the question of how multiple proxylets interact, and how interactions such as order dependencies can be resolved, is open. We anticipate attempting to use metadata

specifications (using the language from issue a)) to maximise the probability of avoiding problems.

The performance and scalability of the DPS is currently far from ideal. Our colleagues in Sydney [ALAN] are addressing these issues, and we anticipate significant improvements will be available before our tests are complete.

## 7 CONCLUSIONS

Application layer active networks will play a crucial role in networked applications that can tolerate a delay of around a hundred milliseconds. They will extend the functionality and versatility of present networks to cover many future customer needs. HTTP caches help reduce the use of bandwidth in the Internet and improve responsiveness by migrating objects and services closer to the client. They are also ideally placed to evolve into the active nodes of the future. XML is a perfect complement to Application layer active networks based on http caches, since it will allow active nodes to be driven by enriched Metadata requests and at the same time will introduce the mechanisms for sharing knowledge between nodes. We have implemented a prototype, which demonstrates the feasibility of this approach and has enabled some initial performance measurements to be made. The results can easily be improved by using a non-interpreted language and a more powerful server.

## REFERENCES

- [ALAN] Application Layer Active Network M. Fry and A. Ghosh, "Application Level Active Networking" , Fourth International Workshop on High Performance Protocol Architectures (HIPPARCH '98), June 98. <http://dmir.socs.uts.edu.au/projects/alan/prog.html>
- [ALEX] Alexander, Shaw, Nettles and Smith "Active Bridging" Computer Communication Review, 27, 4 (1997), pp101-111
- [AMIR] E.Amir, S.McCanne, R.Katz "An active service framework and its application to real time multimedia transcoding" Proc SIGCOMM '98 pp178-189
- [CRAW98] A Framework for QoS-based Routing in the Internet. E. Crawley. R. Nair. B. Rajagopalan. H. Sandick. Copyright (C) The Internet Society (1998). <ftp://ftp.isi.edu/in-notes/rfc2386.txt>.
- [ERIK93] "Mbone - The Multicast Backbone", Eriksson, Hans, INET 1993
- [WOO99] Lauren Wood. Programming the Web: The W3C DOM Specification. IEEE Internet Computing January/February 1999
- [MAN98] F. Manola, "Towards a Web Object Model", tech report, Object Services and Consulting Inc, 1998. <http://www.objs.com/OSA/wom.htm>
- [MORG99] JP Morgenthal. "Portable Data / Portable Code: XML & JavaTM Technologies", tech report. NC. Focus
- [PEI98] Active Cache: Caching Dynamic Contents (Objects) on the Web P. Cao, J. Zhang and K. Beach University of Wisconsin-Madison, USA. <http://www.comp.lancs.ac.uk/computing/middleware98/papers.html#PCao>.
- [TENN98] "Towards an active network architecture". Computer Communication Review, 26,2 (1996) D. Tennenhouse, D. Wetherall.

[WESS97]“Configuring Hierarchical Squid Caches”, Duane Wessels AUUG'97, Brisbane, Australia.  
[XML]Extensible Markup Language (XML) W3C Recommendation 10-February-1998  
<http://www.XML.com/aXML/testaXML.htm>.

## APPENDIX 1

```
<?XML version="1.0" encoding="UTF-8" standalone="no"?>
<!DOCTYPE novel PUBLIC "-//ACTIVE-XML-VEL98//DTD
ACTIVE-XML//ES" "activeXML.dtd" []>
<dynamic initiator>
<front>
<title>active-XML</title><author>Luis Velasco</author>
</front>
<body>
<programms to load>
  <program>
    <name> activebann.Activeban </name>
    <orders>
      <load>
        <trusted program server>
          http://midway/proxylet/activebann.jar
        </trusted program server>
        <host machine>
          midway.drake.bt.co.uk:1998
        </host machine>
      </load>
    <start>
```

```

        <arguments>
        banner1 banner2 banner3 banner4 RANDOM
        </arguments>
        </start></orders>
        <orders_other_caches></orders_other_caches>
    </program>
</programms to load>
<redirect object></redirect object>
</body>
</dynamic initiator>

```

## APPENDIX 2

	HTTP Parser	XML Parsing	URL Download	Proxylet Call	Proxylet Download	Proxylet Respon se	Advert Rotation
Average	7.6	257.4	278.0	14.9	21735.4	317.2	350.8
Standard Error	18.7	180.7	563.6	1.8	1307.6	198.5	63.1

**Table 1.** Numeric result table with the time in milliseconds. It shows the results in milliseconds for the different experiments deployed in the last part of the paper. It is important to note that the figures are in milliseconds and that the Proxylet download time only includes the values for the first ten experiments, then as this step is skipped, this time is reduced to zero.

## APPENDIX 3

The Web community is increasingly targeting the W3C's Extensible Markup Language as its next generation data representation. XML defines a data format for structured document interchange on the Web and is a simple subset of Standard Generalized Markup Language.

XML is syntax for developing specialised markup languages, which adds identifiers, or tags, to certain characters, words, or phrases within a document so that they may be recognized and acted upon during future processing. "Marking up" a document or data results in the formation of a hierarchical container that is platform-,

language-, and vendor-independent and separates the content from any environment that may process it.

- 1 Structure to model data to any level of complexity
- 2 Extensibility to define new tags as needed
- 3 Validation to check data for structural correctness
- 4 Media independence to publish content in multiple formats
- 5 Vendor and platform independence to process any conforming document using standard commercial software or even simple text tools.

The XML definition provides specifications for both XML documents and XML Document Type Definitions (DTD). A DTD is a formal definition of a document. It specifies the element tags that can appear in the document. XML documents have both a logical and physical structure. Physically, the document is composed of units called entities. An entity may refer to other entities to cause their inclusion in the document. It begins in a “root” document entity that is logically composed of declarations, elements, comments, character references and processing instructions. If both physical and logical structure merge well together, then the document is well formed.

XML is a key technology for increasing the Web’s support of complex applications through the use of metadata, which describes or interprets other resources located either on the Web or elsewhere. XML can be interpreted as representing information in the form of properties and their values. This possibility of embedding behaviour associated to specific application elements using metadata makes XML the right technology to compliment active networking.