

# “Total Management of Transmissions for the End-User”, a Framework for User Control of Application behaviour

R. De Silva<sup>†</sup>, B.Landfeldt<sup>‡</sup>, S.Ardon<sup>‡</sup>, A. Seneviratne<sup>‡</sup>,

[ranil, bjornl, seb, aps]@eng.uts.edu.au

<sup>†</sup> School of Computing Science, <sup>‡</sup> School of Electrical Engineering

University of Technology, Sydney, PO Box 123 Broadway,

Ultimo NSW 2007, Australia

**Keywords:** QoS management, adaptive applications, configurable protocols

## Abstract

With the growth of mobile computing, users will have simultaneous access to multiple networks, each with different characteristics, services and costs. This paper introduces TOMTEN (TOtal Management Of Transmissions for the ENd-user), a framework for managing resources in this environment. TOMTEN is a reactive framework that is only active when either the application is started up or the user indicates dissatisfaction with a current session. The framework being reactive, positions it in contrast with all Quality of Service (QoS) frameworks proposed to date, since these are continuously trying to predict the user’s perceived quality and preferences.

In addition to this reactive QoS, TOMTEN also supports application adaptivity within the framework. Application adaptivity in TOMTEN does not require modifications to the application as it is done transparently using proxy modules.

This paper presents the TOMTEN framework and shows how it attempts to assist the user in maximising their resource utilisation through the usage of the reactive QoS management scheme and application adaptation. The paper discusses the different components that form TOMTEN, before introducing a prototype implementation and presenting experimental results using two applications – a MPEG player and a web browser.

## 1. Introduction

With the growth of mobile computing, users will have access to heterogeneous networks consisting of both wired and wireless networks. It is likely that these networks will be overlaid thereby allowing the user to have simultaneous access to more than one network. This environment will be characterised by variable services from both the end-system and the networks. It will also be characterised by rapid fluctuations of service levels and changes in cost. This paper introduces TOMTEN (TOtal Management of Transmissions for the ENd-user), a framework for managing resources in such an environment.

As perception of quality is finally dependent on the user, we believe it is necessary to assist the user in controlling this environment rather than having the system decide on an acceptable level of service [Lan98]. TOMTEN does not make decisions on whether the current level of service being received by the user is satisfactory, rather it leaves it up to the user to indicate when s/he is unsatisfied. Therefore, the TOMTEN framework takes a reactive approach rather than the predictive approach taken by most Quality of Service (QoS) models [Vog95].

The essential features of the TOMTEN framework are that it provides a very simple interface for the user to interact with and a large degree of flexibility and adaptivity. The interface is provided using a Graphical User Interface (GUI), through which the user interacts with TOMTEN. The flexibility and adaptivity are provided through

letting the user choose which networks are used for which applications and by being able to dynamically modify the applications' behaviour and the underlying communication subsystem to the current operating environment.

We will introduce the four components that make up TOMTEN – USA, PRIMATE, Boomerang and a network switch in Section 2. In Section 3, we discuss how these components interact within TOMTEN. Section 4, discusses a prototype implementation of the framework, and some experimental results we have obtained. Future work and related work are discussed in Sections 5 and 6 respectively, before presenting our conclusions in Section 7.

## 2. TOMTEN

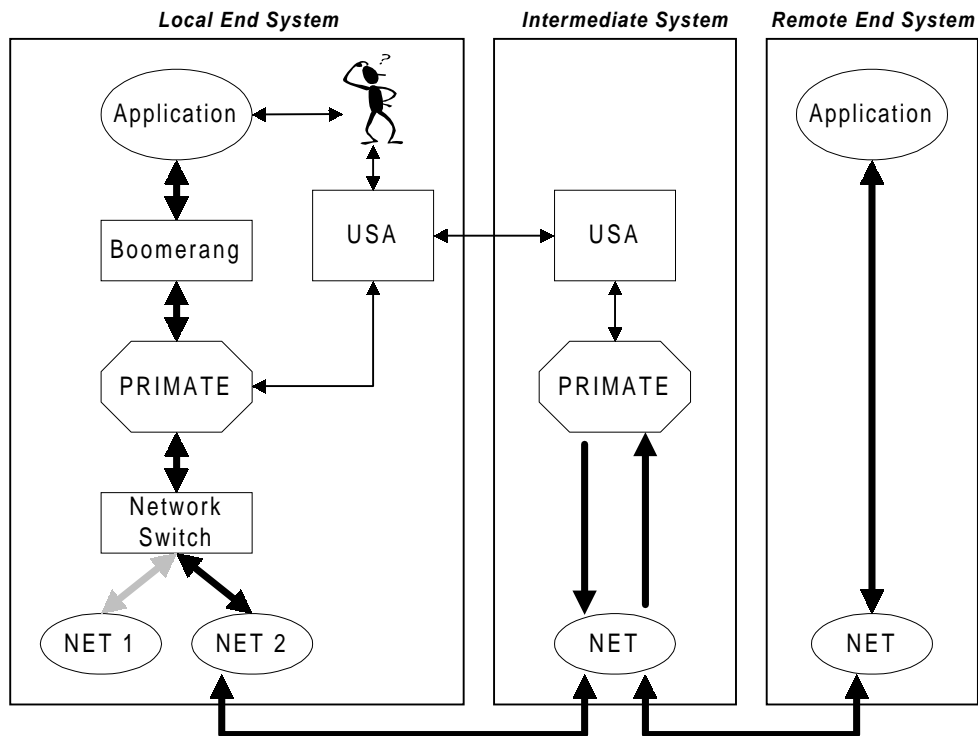
The aim of this framework is to help the user to maximise the utilisation of current resources. We try to achieve this goal firstly by supporting a highly configurable architecture and secondly by simplifying the use of this architecture for the user through a graphical user interface.

We support flexibility in three dimensions: application adaptation, communication protocols and network interfaces. In the network interface dimension - we allow different networks to be used by different applications, as well as managing resource reservations when such a network is available. This allows the user to decide on the cost versus performance trade-off. We also allow for the applications' characteristics and behaviour to be modified. The technique we propose does not require any modifications to the applications, instead we place service modules acting as proxies, in-between the two end-systems. These transparently modify the applications' behaviour similar to the work done by [Fox96]. Finally, we use tailored communication protocols that can be configured to suit the particular network environment and the application's requirements [Des98]. By working in these three dimensions, we are able to best support the requirements of the user, given the currently available resources.

The TOMTEN framework consists of four main modules as shown in Figure 1:

- A reactive quality of service management module called USA (User Services Assistant), that manages a global view of the available resources and provides suggestions to the user about their options when they are unsatisfied.
- An application specific adaptation module called PRIMATE (PROxy Intelligent Module for Adapting Traffic Efficiently) that is used to provide varying levels of service to the user without modifying the application.
- In order to allow applications to be used in the TOMTEN framework without modification, we use a module, Boomerang, which intercepts the traffic from the application and redirects it to the PRIMATE.
- A network switch that enables individual applications to be run through a chosen network interface.

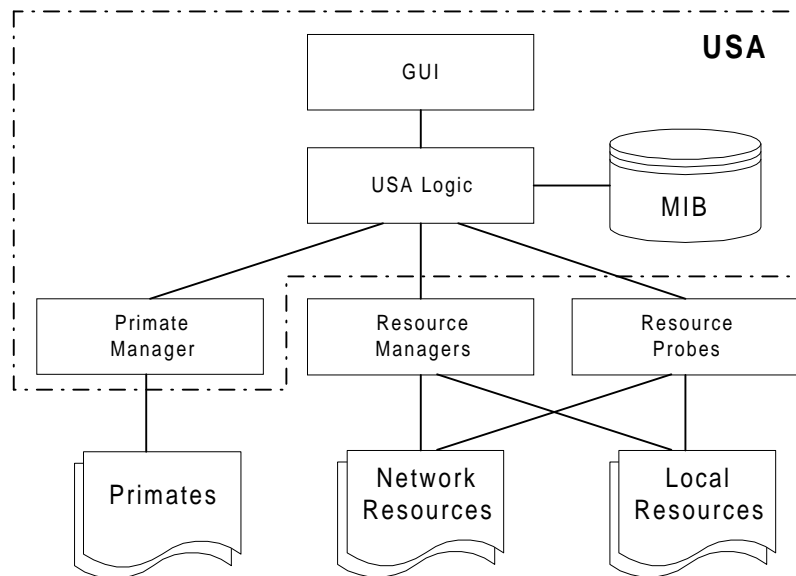
These components are illustrated in Figure 1, and will be discussed in the following sections.



**Figure 1. The TOMTEN Framework**

## 2.1 USA

USA (User Services Assistant) is the reactive quality of service manager at the centre of the TOMTEN framework. As mentioned earlier, most QoS management schemes actively monitors applications and tries to keep the resource usage within certain boundaries. If these QoS schemes detect that an application is violating the pre-determined boundaries, they try to compensate by re-negotiating local and network resources. These complex re-negotiations are done without user intervention or knowledge, in the “belief” that they will improve the perceived level of quality.



**Figure 2 USA - Reactive Quality of Service Manager**

We believe that the perception of quality is dependent on a number of factors like the user's individual preferences, the content of the media stream and surroundings. On this basis, it is not possible for the system to predict the user's perception of quality. Therefore, unlike other QoS managers, USA is not predictive; rather it is reactive and waits for the user to indicate that they are unsatisfied with the current quality of a given application. Furthermore, USA does not automatically make any changes to the negotiated resources. Instead it suggests to the user a possible set of application adaptations (achieved through PRIMATE) that may improve the performance. The user decides, from this list of options provided by USA. If the application adaptations are unacceptable or unable to improve the application performance, the user has the option to change system settings. Thus, the user is informed of the new costs of the connection – something schemes that automatically re-negotiate resources fail to do. Therefore, one of the important features of TOMTEN, is that the user is included as part of the QoS management scheme.

Figure 2 shows the different modules that constitute USA. Given that the user is an integral part of this framework, it is vital that the user can easily interact with USA and this is achieved using a simple Graphical User Interface (GUI), namely a dissatisfied button.

USA uses resource probes to ascertain the differences in the available resources in order to determine the cause of user dissatisfaction. It probes both local and network resources and supports different probing tools. USA also supports resource managers, which are able to reserve both local and network resources where possible. This way USA handles different resource reservation procedures such as with ATM and RSVP.

The PRIMATE manager in USA maintains information of the available PRIMATES and the applications that use them. The PRIMATE manager is also responsible for downloading new PRIMATES and running them, as described in section 2.2 below.

The final two components of USA are the logic module and the Management Information Base (MIB). The USA logic module is responsible for connecting the different components together and suggesting adaptation options when requested by the user. The MIB stores information about the application, this information is

optional and may be entered when the application is registered with TOMTEN. The MIB information is used to aid in the suggestion making process.

## 2.2 PRIMATE

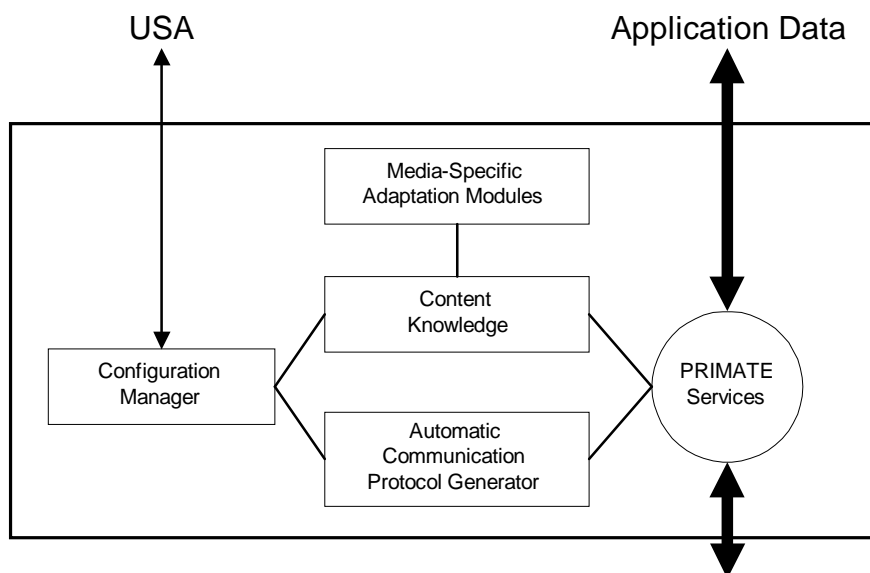
The PRIMATE (PRoxy Intelligent Module for Adapting Traffic Efficiently) provides application adaptation functionality within the TOMTEN framework. Application adaptivity has been traditionally built into the application. Consequently, each application attempts to monitor its performance and adapt on individual knowledge. This approach is limited by the fact that applications are unable to have proper knowledge of the system resources and secondly the development of application adaptivity has been done in an ad-hoc manner.

PRIMATE attempts to address these issues by separating application adaptivity from the application into an independent module. The PRIMATE modules prevent modifications to the application by intercepting application data streams, using Boomerang (see Section 0), and transparently modifying it. PRIMATES are generally implemented in a distributed manner allowing the application data to be adapted on an intermediate system similar to work conducted by [Fox96].

PRIMATES have the advantage that they are content specific, rather than application specific. For example, a PRIMATE developed to provide adaptivity for MPEG Streams could be used with any MPEG Streaming application. Moreover, as PRIMATES are developed separately to the application, they do not require access to application source and encourage development by third party vendors.

Figure 3 shows the basic composition of PRIMATE. The PRIMATE services that perform on the application data are dependent firstly on content specific knowledge, and secondly on a tailored communication subsystem.

The content specific knowledge consists of knowledge about how the application data stream can be adapted. The application content itself, can consist of a number of media types – like a web browser that can support different data types (i.e. different image, audio or video standards). The content knowledge can use a number of media-specific adaptation modules as shown in figure 3.



**Figure 3 PRIMATE**

The second component of PRIMATE's services is based on the development of a tailored communication system. A tailored communication protocol is a communication subsystem specifically developed to support the application's requirements and the network environment. It has been shown in [Des98] that these protocols can be dynamically generated, using automated techniques, to improve application performance.

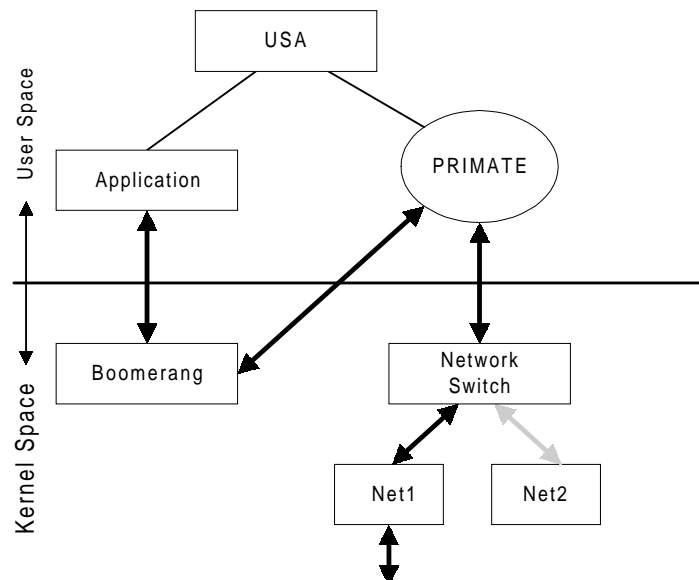
The need to support tailored communication protocols evolved because traditional communication protocols, like TCP and UDP, were shown to be unsuitable for many of today's applications and network environments. The mobile computing environment is one of the environments where traditional protocols have been shown to perform poorly [Bak95, Fla98].

PRIMATE's distributed design, allows tailored communication protocols to be easily supported between the local and remote components, without modification to either of the end applications. Thus, PRIMATE's protocol supports the development of indirect protocols as proposed by [Bak96].

The final element of the PRIMATE module is the configuration manager that generates possible configurations. These suggestions are generated based on the content knowledge and possible tailored communication subsystem configurations that can address the criteria specified by USA. These suggestions are returned to USA who will present these options to the user. When the user selects an option, the configuration manager will be responsible for reconfiguring the services provided by the PRIMATE.

### 2.3 Boomerang

Boomerang plays a simple but important role within the TOMTEN framework. It is responsible for transparently intercepting the data transmitted by the application and redirecting it to the PRIMATE. Without Boomerang, it would be necessary to modify the application to transfer its data directly to PRIMATE, which in turn would require it to support proxy processing (as done in some of the newer Internet applications) or else require modifications to the application. Boomerang provides a uniform solution to this problem as neither of the two can be guaranteed.



## **Figure 4 TOMTEN Prototype Implementation Architecture**

Figure 4 illustrates how Boomerang will intercept traffic from the application in the implementation framework of TOMTEN, which will be discussed in Section 4.1.

### **2.4 Network Switch**

We believe that mobile computing environments will support multiple overlaying networks, each associated with different performances, services and costs. Within this environment, it is necessary to allow users to run applications over a chosen network that will satisfy their requirements at the cheapest cost. The Network Switch has been designed to fulfil this obligation. In the simplest case, the Network Switch is an extended routing table that allows routing on IP address and port numbers.

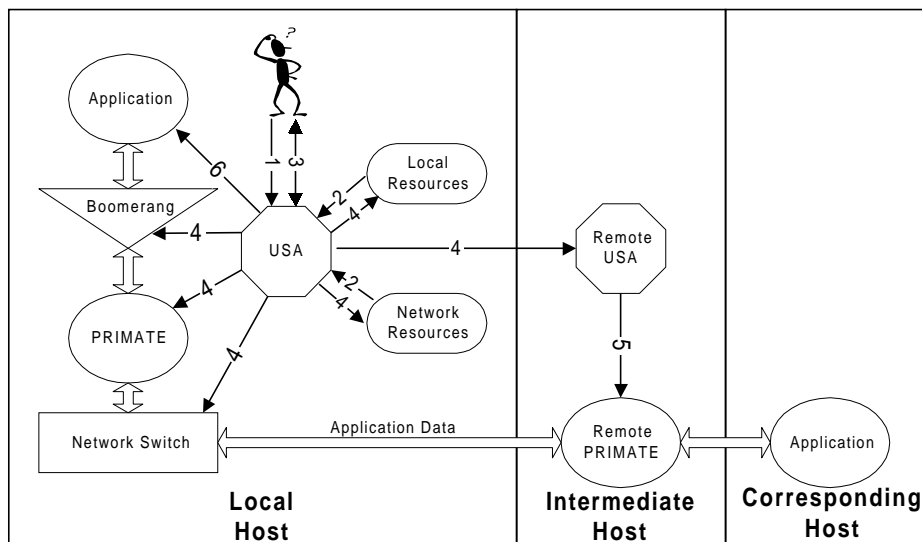
When we look closer at this environment, we notice that there are limitations to this basic solution. Firstly, it is necessary to support mobility in this environment, which can currently be achieved using IP mobility techniques like the Mobile-IP protocol. Mobile-IP hides the movement of the mobile computer by forwarding IP traffic to its current location, commonly results in triangular routing. Mobile-IP protocol also requires that the mobile computer maintain the same IP address for all interfaces – there by allowing interfaces to be “hot-swapped” without disrupting the applications.

Secondly, we require that applications can transfer data across specific network interfaces and receive the corresponding replies on the same interface. This would require that the interfaces be individually identified (i.e. using different IP addresses) to allow the return traffic to be sent on the same path. If all the interfaces maintained the same IP address, the network could not differentiate between the different interfaces and how routing for them should be handled. Thus, all return packets would be routed along the same path to the host.

Thus, it can be seen that these two points are contradictory in many ways and we currently leave it as an open research issue to develop a technique that can accommodate both these requirements.

## **3. TOMTEN Framework**

The user starts an application through USA’s GUI (1) as shown in Figure 5. The user here informs USA of the general importance of the application (high, medium or low preference). USA checks the availability of the local and network resources (2) before suggesting, to the user, a short list of the most appropriate overall configurations given the environment (3). These suggestions are based on information from USA’s MIB and by probing the available resources. We would expect the list to contain two or three choices of suitable configurations, and the final decision is then the user’s. The user has here the choice of accepting one of the recommended configurations or to change the level of importance of the application so that USA will provide a new set of choices.

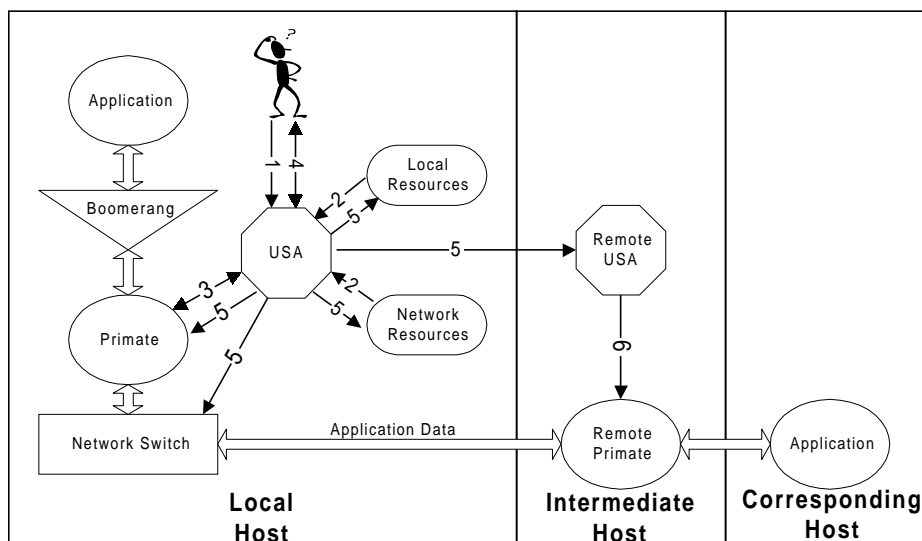


**Figure 5 TOMTEN Application Start Up Sequence**

When the user makes a choice, the USA configures the different TOMTEN components (4) – Boomerang, the Network Switch and starts a PRIMATE. USA will also make any local and/or network resource reservations. USA also contacts its peer entity on the intermediate host and requests it to start the remote PRIMATE (5). Currently to simplify our TOMTEN model, we assume that the Intermediate system is always able to support any requests made. Finally, the application is started (6) by USA.

When this is completed, USA moves into the background. As USA does not try to make any prediction of the perceived level of quality, it does not attempt to modify the environment without user intervention.

When the user is unsatisfied, they indicate it through USA’s GUI (1) as shown in Figure 6. TOMTEN then tries to determine what may have caused user dissatisfaction. This is achieved by probing the current local and network resources (2) and comparing it with the resources that were available when the application was launched. This way USA determines what resources are currently limited.



**Figure 6 TOMTEN User Dissatisfaction Sequence**

USA then requests PRIMATE to suggest application adaptations (3) that might improve the perceived level of quality given the specified limited resource(s). As PRIMATE processes the application data stream, it is able to gather its own information on the performance of the application. This information in contrast to the system level information probed by USA is at the application level. Therefore, PRIMATE will return to USA a set of possible adaptations it can provide as well as recommendations on what types of system level resources may be of benefit to the application. For example, PRIMATE might keep statistics of packet jitter for its data stream and could therefore recommend that a network with less jitter would improve the application's performance. This information would otherwise be difficult for USA to attain.

USA would then recommend to the user how they could improve the current performance of the application (4). This would consist of two sets of options. The first are the application adaptations provided by PRIMATE. The second is a list of system changes that are generated from the information gathered from the local and network resource monitoring and recommendations made by PRIMATE. For example, it could suggest switching to a different network or shutting down other CPU intensive applications. As before, the number of choices recommended to the user should be kept small.

The other consideration when presenting choices to the user is that resource re-negotiation is generally very complex when compared to application adaptation. TOMTEN therefore bases its suggestions on application adaptation and resource changes that do not require re-negotiation.

As said before, the final decision of what action to take is then made by the user (4). This has the advantage that the user can ignore inappropriate suggestions, while in a predictive system, the user is not involved in the decision making process, and is therefore unable to prevent inappropriate changes from taking place.

USA informs the appropriate components of TOMTEN of the user's decision to make the appropriate changes (5 & 6).

### **3.1 Features of the TOMTEN Framework**

The novel features of the TOMTEN framework are:

- We provide a complete framework that supports application adaptivity, configurable protocols and QoS. We do not know of any other framework that attempts to address all these issues.
- TOMTEN de-couples QoS management and application adaptation by separating the functionality into USA and PRIMATE respectively.
- We propose the use of a reactive QoS manager - USA – that differs from other QoS management models by not attempting to predict users' perception of quality.
- Application adaptation is made transparent to the application using PRIMATE and BOOMERANG. Therefore, it is not necessary to customise existing applications to be used with TOMTEN.
- TOMTEN allows PRIMATE modules to be developed by third party vendors. This will allow for a greater variety of adaptation modules to suit different environments and media to be developed.

- PRIMATE was designed to run in a distributed environment. This allows for distributed processing of the application data as well as supporting split connection protocols as proposed in [Bak95].
- The framework is designed to support dynamic downloading of PRIMATEs either to the local or intermediate systems.

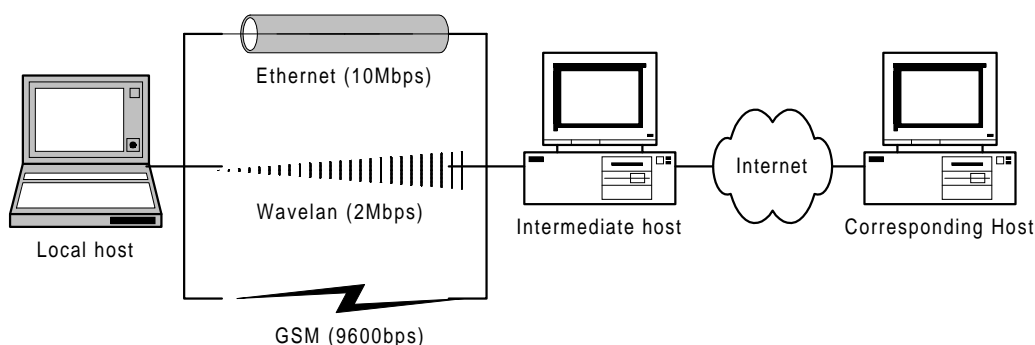
## 4. Prototype Implementation and Experimental Results

As TOMTEN, allows all final decisions to be made by the user, it eases the precision required in the suggestions it makes. Therefore, the development of a working prototype implementation of TOMTEN was considerably simpler than other QoS schemes. This section discusses the development and testing of a prototype implementation of TOMTEN. We will firstly discuss how the different components of TOMTEN were implemented before explaining experiments conducted with an MPEG application and a web browser application.

### 4.1 Prototype Implementation of TOMTEN

The prototype implementation of TOMTEN was developed on the Linux platform. We have currently implemented prototypes of the USA, PRIMATE and Boomerang components of TOMTEN. The implementation architecture of TOMTEN is shown in Figure 4. The network switch is currently emulated by dynamically changing the routing table. Although this method has some inherent limitations, it is sufficient for conducting early performance measurements.

Our basic network architecture is shown in Figure 7. It consists of a mobile computer that is connected to a base station through three different interfaces – a 10 Mbps Ethernet, a 2 Mbps Wavelan and a 9600 bps GSM link. The GSM link is simulated using a fixed serial link between the mobile host and the base station.



**Figure 7 TOMTEN Implementation Testbed**

USA has been implemented primarily using the JAVA language. Although the network probes were implemented in C for performance reasons.

The network probes operate by transferring a block of data from the local host to the intermediate host. The network probes are currently very simple and allow us to determine which is the most suitable link. Currently a Linux specific system call is made in order to bind the application to a given interface to guarantee that transfers only take place on that interface. They currently record the available bandwidth, the round-trip time and packet loss for these short transfers. Different size transfers are

done for each of the networks to take into consideration the maximum bandwidth and all transfers are bounded by a maximum delay. It is our intention within the TOMTEN framework that the probing only takes place when the user indicates dissatisfaction in order to save on over-head traffic. Hence, it is important that the monitoring process is completed within reasonable time to prevent delays in the responsiveness of TOMTEN. The implications of this will be further investigated.

The PRIMATES have been developed in C although we are planning to port them to JAVA to introduce platform independence and code portability. The PRIMATES use sockets to communicate with USA. We discuss the implementation of the PRIMATES in detail in the following sections.

Boomerang has been implemented as a module within the Linux kernel at the socket layer. It detects socket calls made by the application, and redirects them to a specified PRIMATE port. USA and PRIMATES use a library of system function calls to communicate with Boomerang. We are considering developing Boomerang as a dynamically link library which will be called instead of the system socket libraries. This will not require modifications to the kernel and will make Boomerang easier to port between different platforms.

## **4.2 MPEG Player Application**

The primary aim of this prototype implementation of TOMTEN was to verify the feasibility of the framework and to evaluate the overhead it introduces. The application we used to evaluate the overhead introduced by the framework was a MPEG player. This MPEG player is based on the MPEG-1 standard that only includes video sequences. It was the attribute of the MPEG standard, to define frames of different levels of detail (I, P and B frames), that allowed us to customise the MPEG player to provide varying levels of quality to the user [Rao96].

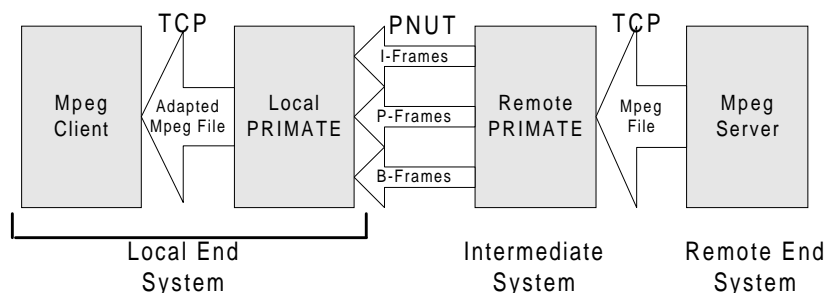
To support adaptivity for this MPEG player, we used an automatically generated tailored protocol between the local and remote PRIMATES. The tailored protocol was generated using an automated communication protocol generation model called PNUT (Protocols configured for Network and User Transmission). PNUT allows the use of multiple data channels, each supporting their own protocol. PNUT manages the data channels using out-of-band signalling on a control channel. Further details about PNUT can be found in [Des98]

The MPEG player uses three data channels to transfer the different frame types as shown in Figure 8. The channel containing the I-frames is always uses a reliable protocol (i.e. ordered and error free) as these frames are compulsory for the MPEG application to correctly interpret the data. By defining the protocols used on the remaining P and B channels, we are able to determine the level of detail viewed by the user.

Currently our MPEG PRIMATE has three levels of service – a Full service, a Partial service and a Filtered service. The Full service defines all three channels to be reliable, therefore guaranteeing that all the frames will be delivered and viewed. The Partial service defines the latter two channels to unreliable (i.e. unordered and no error recovery). Using the Partial service, if the network is congested or the application is slow processing the MPEG clip, P and B frames will be lost, reducing the level of detail viewed by the user. This level of service provides partial ordering and partial reliability as discussed in [Dia94]. The Filtered service defines the latter two channels

to be filtered (i.e. data dropped at the source). This provides a low quality video consisting of just I-frames.

Our testbed consisted of three PCs (a 486 and two Pentiums) running Linux 2.0.33 and connected through a dedicated Ethernet network. The MPEG Player was based on the Berkeley Player [Ber96] extended into a client-server application. As shown in Figure 8, the local PRIMATE and the MPEG Client coexisted on the local end-system. The MPEG Server was on a remote end-system while the remote PRIMATE ran on an intermediate system. The remote PRIMATE could also have run on the remote end-system but in these experiments by placing it at an intermediate system, we were also able to provide indirect protocol services as shown by [Bak95].



**Figure 8 MPEG Application made Adaptive**

The following measurements were taken for an MPEG file (consisting of 161 I-frames, 160 P-frames and 639 B-frames) firstly using a direct connection from the MPEG client to the server without using TOMTEN and secondly using TOMTEN with MPEG PRIMATES. The results of these measurements are shown in the table below.

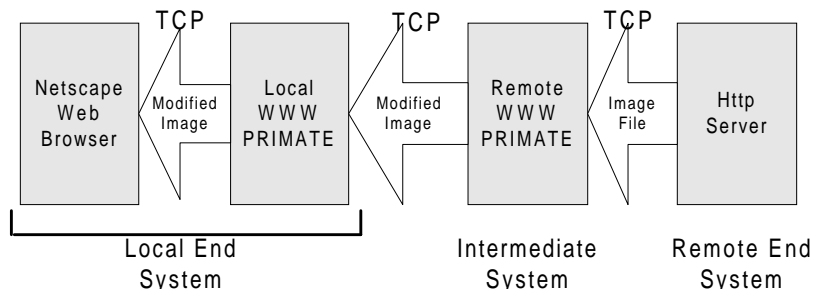
	<b>Frame rate (fps)</b>	<b>Play out Time (seconds)</b>	<b>Frames Processed Total</b>
<b>Direct Connection</b>			
full video (I,P,B)	12.4	77.23	960/960
<b>TOMTEN</b>			
Full video (I + P + B)	12.3	77.75	960/960
Partial video (I + B? + P?)	12.4	67.25	832/960
Filtered video (I)	3.6	45.63	162/960

By comparing the times of the full video play-out when using the direct connection and with the use of TOMTEN, we able to show that the overhead introduced by the PRIMATES are insignificant. Furthermore the differences between the play-out times and the number of frames processed in the three different service levels illustrate that TOMTEN was able to modify the content of the MPEG stream and hence the perceived quality to the user.

### 4.3 Web browser Application

One of the most commonly used Internet applications, today, is the web browser. We therefore decided to develop PRIMATES to allow application adaptation to be introduced into web browsers.

The Web PRIMATE processes the images that the browser receives, similar to the work conducted by [Fox96]. The processing of the images are conducted on the intermediate system as shown in Figure 9. Currently the Web PRIMATE modifies JPEG and GIF images and is capable of reducing the number of colours in the image or by converting it into greyscale. The Web PRIMATE currently does not attempt to resize the pictures as done in [Fox96].



**Figure 9 Web Browser Application made Adaptive**

When the web browser requests a file, the socket call is intercepted by Boomerang and transferred to the local Web PRIMATE. The Web PRIMATE then relays the request to the Remote Web PrimatE. With the Web PRIMATES, the main processing is done at the Remote PRIMATE and the local PRIMATE mainly servers as a relay. The Remote PRIMATE requests the HTML file from the HTTP server and checks the stream header to detect if it contains an image. If the image is not found or the stream is unrecognised, it is allowed to pass without modifications. If an image is detected and the PRIMATE has been requested to reduce the incoming image size then the file is saved locally. Currently the remote PRIMATE does not process the image stream, rather it downloads the entire file and uses image processing tools to reduce the size of the image. The modified image is then sent to the local PRIMATE who relays it to the Web browser to display.

As it can be seen from this example, it is not necessary to modify the Web browser or the HTTP server. Furthermore, the Web PRIMATE can be used with any web browser.

As the Web PRIMATE, receives the entire image file, processes it and then forwards it on to the Web browser, there is an overhead associated with the Web PRIMATE. This overhead is related to the cost of downloading the file, which is dependent on the size of the file and speed of the link between the Remote PRIMATE and the HTTP server. Then there is an additional overhead for processing the image. These overheads are countered depending on the speed of the link from the intermediate host to the local host and the amount the image size is reduced. Therefore, the benefit of using this Web PRIMATE occurs when there is a slow link between the local and intermediate hosts and the faster connections from the intermediate host to the remote host.

We conducted experiments by running a web browser using the GSM link (9600 bps) of our experimental test-bed to measure the benefits of using the Web PRIMATE. We used the Netscape Web Browser 4.04 for our experiments. Firstly, we measured the time it took from when the request was received from the web browser until it was completed. This time included the transfer from the HTTP server, the processing of the image file and the transfer from the Remote PRIMATE. We also separately measured the times on the Remote PRIMATE to record how long it took to receive the file from the HTTP server and the time to process it.

Two full colour images were transferred – a GIF and a JPEG. Colour reduction processing and grey-scale processing were done for both images. During the experiments we tried to limit the image processing so that the resultant image was still clearly recognisable as a reduced quality version of the original. The results are shown in the table below.

	Image Size	Total Time	Processing Time
GIF Image <a href="http://silicon.ee.uts.edu.au:81/images/opera1.gif">http://silicon.ee.uts.edu.au:81/images/opera1.gif</a>			
Normal	60,624 bytes	78.55 secs	None
Reduced Colours	19,377 bytes (67% reduction)	23.34 secs (70% reduction)	2.058 secs
Greyscale	16,059 bytes (73% reduction)	19.78 secs (74% reduction)	1.739 secs
JPEG Image <a href="http://silicon.ee.uts.edu.au:81/images/opera.jpg">http://silicon.ee.uts.edu.au:81/images/opera.jpg</a>			
Normal	17428 bytes	19.50 secs	None
Reduced Colours	8036 bytes (53% reduction)	10.26 secs (47% reduction)	0.338 secs
Greyscale	6912 bytes (60% reduction)	9.15 secs (53% reduction)	0.267 secs

The results show that great improvements are possible by using the Web primate on a slow link. These benefits will be reduced when using higher speed links since the size reduction techniques currently used were very simple. Further reductions could easily be achieved by scaling down the large pictures as shown in [Fox96].

## 5. Future Work

This paper has not gone into detail on how the interactions with the intermediate system will be conducted. This includes the issue on how TOMTEN will choose an intermediate system. Other issues that involve the intermediate system are how we can dynamically transfer and run our PRIMATEs on them.

As mentioned in Section 0, we are studying how we can provide routing in an environment with multiple networks and which has to support mobility. We would also want to route traffic on an application basis rather than simply on the destination.

Finally as the user interaction is currently such and important part of the TOMTEN framework, we need to conduct further studies on the development of the GUI through which they communicate with TOMTEN.

## 6. Related Work

The only research similar to TOMTEN that we currently know about is the work proposed by [Nob98]. Although their work does not support configurable protocols, the use of intermediate systems or downloadable proxies. Furthermore, their QoS management scheme is predictive as opposed to USA that is reactive.

## 7. Conclusions

In this paper, we have introduced TOMTEN, a framework for managing resources in order to provide a user with QoS management. The essential characteristics of TOMTEN are:

- It is user controlled. TOMTEN does not attempt to predict the user's perception of quality. It is only activated when the user wants to start an application or shows dissatisfaction with an ongoing session.
- TOMTEN supports application adaptation without requiring any changes to the application.
- It de-couples QoS management from application adaptivity.

We have shown through the development of the prototype implementation of TOMTEN, that it is feasible and easy to realise. We have also shown through the development of two PRIMATEs for MPEG Player applications and Web browser applications that we are able to transparently provide application adaptation.

## Acknowledgements

We would like to acknowledge Christophe Diot's help in the designing of USA. We would also like to acknowledge the financial support from Ericsson Australia and the Australian Research Council.

## References

- [Bak95] A. BAKRE AND B. BRADRINATH, *I-TCP : Indirect TCP for mobile hosts*, in Proceedings of 15<sup>th</sup> Intl. Conf. on Distributed Computing Systems, May 1995.
- [Ber96] *Berkeley Player*, Berkeley Multimedia Research Center (BMRC), University of California, Berkeley, 1996  
[Http://bmerc.berkeley.edu/projects/mpeg/mpeg\\_play.html](http://bmerc.berkeley.edu/projects/mpeg/mpeg_play.html)
- [Des98] R. DE SILVA, *PNUT - Protocols configured for Network and User Transmissions*, PhD Thesis, University of Technology, Sydney, Jan. 1998, submitted for assessment.
- [Dia94] M. DIAZ, C. CHASSOT AND A. LOZES, *From the partial order connection concept to partial order multimedia connections*, in Proceedings of HIPPARCH Workshop, Dec. 1994.
- [Fla98] A. FLADENMULLER, R. DE SILVA, *Effects of User Mobility on a TCP Transmission*, to be published in IFIP International Conference on Performance of Information and Communication Systems, Sweden, May, 1998.
- [Fox96] A. FOX, S. D. GRIBBLE, E. A. BREWER AND E. AMIR, *Adapting to network and client variability via on-demand dynamic distillation*, in Proceedings of 7<sup>th</sup> Intl. Conf. On Arch. Support of Prog. Lang. And Oper. Sys. (ASPLOS VII), Oct. 1996, Cambridge, MA.
- [Lan98] B. LANDFELDT, A. SENEVIRATNE AND C. DIOT, *User Services Assistant: an end-to-end reactive QoS architecture*, submitted to IFIP 6<sup>th</sup> International Workshop on Quality of Service (IWQoS'98).
- [Nob98] B. D. NOBLE, M. SATYANARAYANAN, D. NARAYANAN, J. E. TILTON, J. FLINN AND K. R. WALKER, *Agile Application-Aware Adaptation for Mobility*, to appear in the Proceedings of the 16<sup>th</sup> ACM Symposium on Operating System Principles, 1998.
- [Rao96] K. R. RAO AND J. J. HWANG, *Techniques & Standards for Image Video & Audio Coding*, Prentice Hall, 1996.
- [Vog95] A. VOGEL, ET AL., *Distributed Multimedia and QoS: A Survey*, IEEE Multimedia, Vol. 2, No. 1, Summer 1995.