A Middleware Architecture for Streaming Media over IP networks to Mobile Devices

Kevin Curran
Internet Technologies Research Group
Northern Ireland Knowledge Engineering Laboratory
University of Ulster, Magee Campus, Northern Ireland, UK
Email: kj.curran@ulster.ac.uk

Gerard Parr
Internet Technologies Research Group
Northern Ireland Knowledge Engineering Laboratory
University of Ulster, Coleraine Campus, Northern Ireland, UK
Email: gp.parr@ulster.ac.uk

***Abstract* - Wireless networks differ in bandwidth, size and access costs each requiring a set of protocol functions to enable devices to communicate efficiently. Portable multimedia devices such as PDA's and laptops will also vary greatly however all these devices will require optimal multimedia delivery. A traditional method is for sources to limit their transmission rates to accommodate lower bandwidth links, even though high-bandwidth connectivity is available to many participants. This method similar to others does not provide optimum throughput to heterogeneous clients due to its quest for a common denominator bandwidth. In addition, due to the divergence of users and applications, traditional protocol stacks are frequently enriched with additional functionality such as transport protocol functionality, synchronization and presentation coding which can lead to a performance bottleneck due to the insufficient processing power and memory of portable devices.**

Chameleon is 100% Java middleware for multimedia streaming to heterogeneous mobile clients, which allows the dynamic configuration of protocols with respect to application requirements and available network resources. We evaluate the dynamic reconfigurability of the middleware in order to demonstrate runtime adaptation. We especially concentrate on the secondary quality transformation technique (SQT) of the middleware which enables clients to not only subscribe to media groups in accordance with available resources and network capacity but to also adapt media quality within each quality group dynamically in accordance to resources constraints.

***Keywords*—middleware, QoS, adaptive protocol stacks**

I. INTRODUCTION

The distinction between mobile phones and personal device assistants (PDA's) has already become blurred with pervasive computing being the term coined to describe the tendency to integrate computing and communication into everyday life [Dertouzos99]. The creation of low bit rate standards such as H.263 [H263] allows reasonable quality video through the existing Internet and is an important step in paving the way forward. As these new media services become available the demand for multimedia through mobile devices will invariably increase.

Corporations such as Intel do not plan to be left behind. Intel has created a new breed of mobile chip code named Banias. Intel's president and chief operating officer Paul Otellino states that 'eventually every single chip that Intel produces will contain a radio transmitter that handles wireless protocols, which will allow users to move seamlessly among networks. Among our employees this initiative is affectionately referred to as 'radio free Intel'".[1]

We argue that traditional monolithic protocols are unable to support the wide range of application requirements on top of current networks (ranging from 9600 baud modems up to gigabit networks) without adding overhead in the form of redundant functionality for numerous combinations of application requirements and network infrastructures. Flexible and adaptive frameworks are necessary in order to develop distributed multimedia applications in heterogeneous end-systems and network environments. Catering for this wide range of mobile devices is the focus of this paper.

II. STREAMING MEDIA TO MOBILE DEVICES

Implementing protocols from scratch is a complex and time-consuming task however frameworks are designed to ease the task of developing new protocols. They achieve this by providing a library of basic protocol functions and templates for implementation of new protocols (e.g. the X-Kernal [Peterson91], STREAMS [Unix90] and Conduits+ [Huni95]). The weakness with these protocol frameworks is that they are not portable. Many mobile manufacturers at present including Motorola, Ericcson and Nokia have devices in the marketplace, which support the Java 2 Micro-Edition (J2ME) framework. The trend towards the porting of implementations of Java on mobile devices is expected to continue[2] as it is an ideal language for protocol implementation due to its extreme portability and support for modular programming in an object oriented fashion.

Computers communicate through the use of a common set of protocols that define the set of rules to be adopted for the duration of the communication. Protocols stacks have traditionally been monolithic chunks of code where all data regardless of whether it is a continuous stream of bits with strict time dependencies between those bits, or the packets

[1] www.pcplus.co.uk (Article in May 2002 issue)
[2] http://www.javamobiles.com/

comprising an asynchronous message are sent through the same stack. The nature of the data is not considered however and therefore there is no room for optimisation of the code to create a more efficient service.

In addition, information travelling over wireless networks is prone to increased error as opposed to data over a wired local area network thus an argument exists for protocols tailored to the nature of each underlying network medium. A protocol such as TCP can be used to transport data over this medium, however, TCP applies a rate controlling mechanism, which halves the current throughput upon detecting congestion. Congestion is detecting by recording lost packets however losses are likely to occur on wireless networks due to error rather than network congestion thus the TCP congestion mechanism is inappropriate [Kojo97]. A generic monolithic protocol stack, which contained mechanisms to cope with every use case could be developed however this would lead to a solution with a large degree of redundancy as many functions would not typically be called. Additionally the amount of user space memory required to implement this solution would result in the exclusion of memory-constrained devices.

Another problem exists in the heterogeneity of the mobile devices with capabilities ranging from powerful full specification laptop PC's, to low powered PDA's. Some devices will be capable of displaying full colour 1024x800 while others may only manage black and white 100x60 screen resolution. Approaches to the problem have involved sending the lowest common denominator stream to all receivers, however this penalises the more powerful mobile clients that receive far below their true capabilities. There is also a lack of adequate development frameworks to cover the characteristics of the above kind of systems, as classical methods for distributed systems are not sufficient due to their static character. What is required is a framework that supports dynamic adaptation in a changing environment, which is systematic, yet, axiomated on practical experience.

Solutions have appeared at times to counter many of the various problems above however, these have for the most part been proprietary in nature and lacking in any standard API to enable new mechanisms to be deployed at a future time to cope with additional use cases.

III. THE CHAMELEON FRAMEWORK

Chameleon is a Middleware Framework, which supports reconfigurable dissemination oriented communication. It extends earlier research - the RWANDA framework [Parr-Curran00] - by incorporating a meta-object protocol to allow dynamic system reconfiguration. Chameleon fragments various media elements of a multimedia application, prioritises them and broadcasts them over separate channels to be subscribed to at the receiver's own choice. The full range of media is not forced on any subscriber, rather a source transmits over a particular channel, and receivers, which have previously subscribed (to the channel), receive media streams (e.g. audio, text and video) with no interactions with the source. Clients are free to 'move' between differing quality multicast groups in order to receive the highest quality (or move to a lower quality group for the greater good of minimising network congestion. This is known as Primary Quality Transformation (PQT). In addition proxies offload intensive computing on behalf of clients and dynamic reconfigurable abilities allow new components to be slotted into live systems. The new components can perform additional transcoding on streams within each group. This is known as Secondary Quality Transformation (SQT). PQT and SQT provide a rich set of features for the optimal reception of multimedia flows. Chameleon is packaged with a core API and a set of Java template classes. The object-oriented design process produces a hierarchy of classes, from which a collection of objects is instantiated to build a particular application.

Chameleon (Figure 1) addresses the network congestion and heterogeneity problem by taking into account the differing nature and requirements of multimedia elements such as text, audio and video thereby creating tailored protocol stacks which distribute the information to different multicast groups allowing the receivers to decide which multicast group(s) to subscribe to according to available memory, display resolutions and network bandwidth availability. Chameleon supports dissemination of multimedia from a source to multiple destinations however end-to-end closed-loop control can be difficult and cumbersome with multiple receivers, as the slowest receiver will impede the progress of the others.

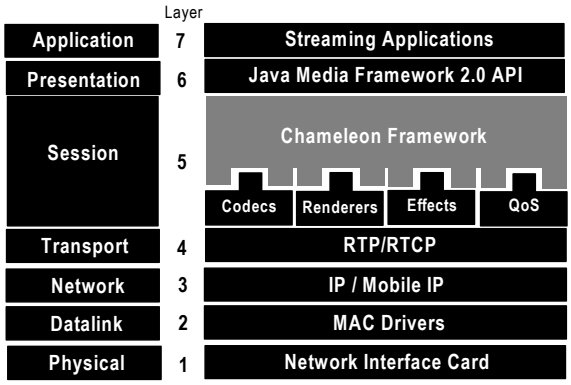

Figure 1: Chameleon in comparison to the OSI model

We believe that tight, closed-loop, end-to-end control is inappropriate for applications that expect a large number of receivers having different capabilities interconnected through networks proving different QoS. Instead, we have adopted an alternative approach that relies on very loose coupling between the source and the receivers, i.e. an open-loop approach, more suited to real-time continuous media.

Multimedia is composed of varying types such as audio, video, text, control information, etc. Within these types, exists a multitude of formats such as PCM, JPEG, and MPEG etc. Take the example of a conference application, where control information and files need to be transmitted alongside audio and video. The control information such as

who has floor control and files need reliable transport guarantees, whereas the audio and video may be transmitted with a differing QoS. Traditional transport protocols transport the media types through the same stack. If a video stream is filtered through the same stack as an audio stream, the video data will have to adopt the packet size allocated to the audio stream. Audio in general runs more efficiently with smaller packet sizes [Modiano99]. Isochronous multimedia traffic can tolerate some loss however data that misses its expected delivery time is of no use. Therefore it is more efficient to lose smaller packets than larger packets however, smaller packets demand increased header processing in routers.

Small packet sizes are not optimal for video data due to the increased size of the media involved. Using an identical protocol stack to cater for all these transport types is not an ideal scenario therefore a more efficient method would construct optimised protocol stacks for each of the media e.g. audio, text and video. Maximum benefit would be achieved if this could be implemented at run-time to cater for the applications particular preferences. A traditional stack belonging to a multimedia application, for example, would send the audio and video in packets of identical size. Research shows that optimal audio packets are smaller in size than video packets [SCTE00].

Multiple multicast multimedia groups provide a finer granularity of control compared to using a single video/audio/text stream, because a receiver may subscribe to one or more layers depending on its capabilities. If a receiver experiences packet loss as a result of network congestion, moving to a lower quality multicast group will reduce congestion, and hence will reduce potential packet loss. This is known as Primary Quality Transformation (PQT). This technique allows media to be composed into broad bandwidth encoded qualities thus all a system needs to do to increase or decrease quality is to move between multicast groups. The Secondary Quality Transformation (SQT) technique compliments this technique by providing fine-grained control of quality within each group by the insertion of transformations in the stream such as compression.

The source carries separate streams with each multicast group G_i, $i = 1,2,...,K$. The application of multiple multicast group streaming techniques to continuous media hand mobile devices the capability of allocating resources based on local specifications and priorities (see Figure 2). Multicast group streaming enables receivers to change the quality of the stream they receive, independently of one another without the source being aware of the change. Considering the feedback problems of multicast, this is a useful property and fits well with an open-loop approach to congestion control of high-speed networks, as when network congestion arises, it is possible to move between quality groups without interruption in service. Service quality should only be slightly reduced however; this technique can be highly effective as a last resort for congestion control.

Priorities can be assigned to each multicast group to allow streams to be protected against competing streams. This is an application level QoS scheme and can be implemented easily in Chameleon as all streams pass through a proxy. Pre-set priority levels overcome many problems associated with streaming over wireless links. Atmospheric conditions, physical obstacles, electromagnetic interference and other phenomena interfere with transmissions over wireless channels, ultimately introducing bit errors. Long lasting error bursts can severely impact upon applications, causing video frames to be dropped, thus effectively lowering the perceived quality. Chameleon supports the seamless operation of real-time streams over wireless links by assigning a priority and a portion of the link's resources, which are protected from being used by lower priority streams.

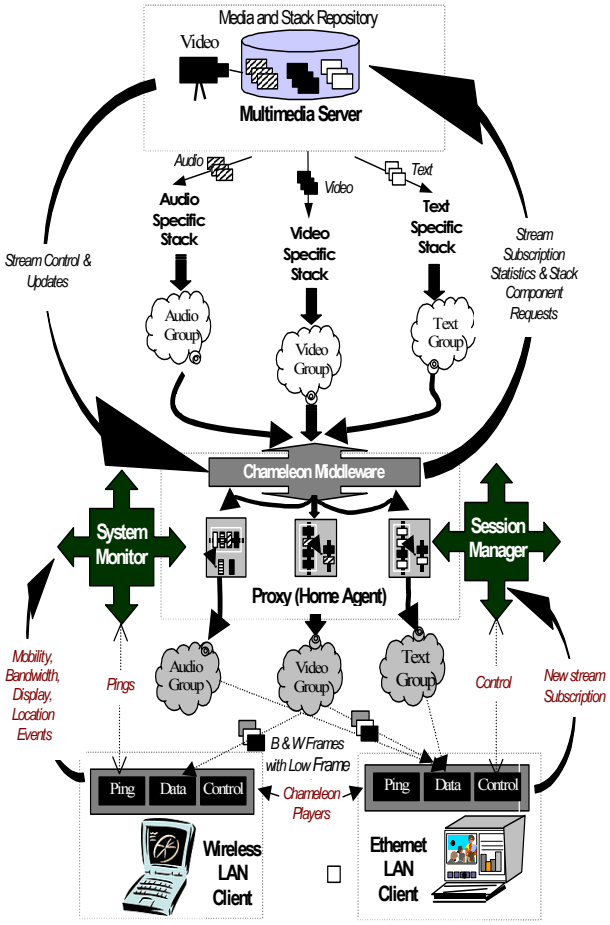

Figure 2: Multiple Multicast Multimedia Groups

As Chameleon is an open-loop system, a segment with its size defined by the application, is an independent piece of information, similar to the Application Data Unit concept described in [Clark90]. We expect that many multimedia applications, guarantees of reliable delivery will not be necessary for various media component types, and some segments could be dropped at times of heavy congestion. In addition, some of these applications may actually be quite tolerant of delays, as described in [Clark92]. Particularly for lower priority components, applications would be expected to recover gracefully from loss of segments, or adapt to

changes in the delays of their arrivals. Performing transformations on multiple streams is suited to the approach of a source transmitting multiple coded media streams from which the receivers pick according to their individual specifications and capabilities.

The benefit of this approach is that there is reduced complexity due to the absence of feedback control mechanisms, which are often redundant for continuous media. Here the source's main concern is to deliver various media streams onto a multicast channel, with no emphasis on where they end up and how they are used. A client's (or receiver's) main concern is what to extract from a channel, which is viewed as offering multiple streams, some or all of which are of interest. We believe this communication paradigm is appropriate for multimedia distribution services such as cable television systems where a single source generates video (and associated audio) distributed to a large set of receivers who generally have little or no interaction with the source.

Chameleon addresses application, application control, and transport layers. The application layer consists of the multimedia application (e.g., a video-on-demand application) which is responsible for retrieving the stored audio/video file with captions/subtitles (multilingual), composition at the sending end, and the audio/video client which is responsible for decoding and displaying the video frames at the receiving end. Application control consists of a media filter at the sending side to demultiplex each stream into several sub streams, and media filter at the receiving end to multiplex back one or more sub streams for the audio/video client. These multiplexed streams (transport layer) differ from common practice in that these streams are not logically grouped together and shipped over the wire. Instead, the media elements are divided into audio/video/text by the event filter and distributed to separate groups in accordance with application layered framing practice and then the receiving filter directs the streams to the relevant media application, thus the streams retain their distinctiveness. Media may be stored in separate files on the server and so that there is no need to split the media in real-time. The application media filter receives events from the application which may categorised them as text, audio or video. A session manager is consulted to see how many groupings of each category are required. The normal is one for text, and three each for audio and video. The text stack is composed as a reliable stack. The audio/video stacks are both UDP differing in default packet size and header sizes. Each media is sent to separate multicast groups where the well-known addresses are obtained from the session manager. Each of the three sub-groups of audio and video will require a separate multicast address. Since the network load changes during a session, a receiver may decide to join or leave a multicast group, thereby extending or shrinking the multicast trees.

The active network proposals target network programmability without being content-aware. Chameleon, in contrast targets content-aware application-level programmability. The rapid increase in media types necessitates a network infrastructure that allows clients and servers to be free from media dependency and burdens of managing content & client heterogeneity. This can be extremely important for streaming media because of its demanding resource requirements for processing, translation, and transmission thus middleware must combine media awareness with a high degree of intelligent adaptivity in order to truly serve heterogeneous clients. PQT with priority relies on third party traffic being disabled (or rate controlled). This can be achieved through the technique of blocking ports. Traffic types within Chameleon are assigned port numbers to designate media type and these work alongside the well know port numbers assigned to traffic such as FTP, TCP etc. in order to bring about prioritised media traffic streams. For the PQT with priority technique to work properly, the end-to-end path must be composed of Chameleon filters enabled in 'firehedge' mode. We adopt this title (i.e. hedges being much weaker than walls) as opposed to firewalls as we acknowledge the limited functionality of a technique such as this in comparison to industry standard firewalls which block, monitor and classify traffic in a much more detailed manner.

IV. EVALUATION OF SQT

Secondary Quality Transformation (SQT) assumes responsibility for responding to quality fluctuations within a multicast group as opposed to the Primary Quality Transformation (PQT) technique assumes responsibility for coarse grain adaptation decisions by moving between multicast groups upon violation of group bandwidth limits. SQT does not involve movement between multicast groups thus it can be seen as a fine grain adaptation technique where improvements are attempted on the media stream through the use of filters on proxies and mobile client devices.

To demonstrate the feasibility of the SQT approach, three wireless cells were operated in separate parts of the computing building at the University of Ulster. Each cell has its own dedicated IP subnet where access is provided and controlled through routers on Linux PC's connected to a number of mobile devices. All tests were performed using three similar test beds with the basic design having a client and a server communicating using a proxy and a home agent. The proxy performs the filtering as mentioned earlier, while the home agent provides connectivity for the client however due to space restrictions we only present the wireless topology which uses 10 Mbps Ethernet to connect the server, proxy and home agent, but uses a 2 MBps Zoomair wireless LAN to connect the home agent to the client. This is an environment exhibiting a reasonable amount of heterogeneity. An interesting characteristic of this environment is that Wireless LANs have a much higher packet loss rate due to the nature its medium (radio). This has various effects on test results, making them much less uniform than the other environments.

i. Retransmission Timers

Chameleon may not be able to compete with finely tuned versions of TCP and UDP, however the additional flexibility

provided by Chameleon can be useful for environments that have not been considered by standard generic protocols. One example is where a particular service which lies somewhere between TCP and UDP is required or where stronger guarantees than TCP are needed. Chameleon is ideal for circumstances where a protocol needs to be configured to match the characteristics of a specific network environment. One example could be a multimedia application, which requires congestion control but not ordered delivery, a service that TCP or UDP cannot provide.

Another scenario that we examine here is TCP over wireless networks. TCP performs well in wired networks with its low latency and low failure rate but overreacts in wireless networks where packet loss can occur for many reasons other than congestion [Wong01]. The principal problem is the congestion control algorithm. Nearly all TCP implementations nowadays assume that timeouts are caused by congestion, not by lost packets therefore when a timer expires; TCP retransmits the packet, but also invokes congestion control measures by reducing the TCP window size and throughput. The industry standards for versions of TCP such as Reno retransmit data on reception of 3 duplicate acknowledgements or the expiration of a retransmission timeout. This timeout however can be set very coarse, in the order of 350-500ms. Local Area Networks and organisations fortunate enough to be connected directly (e.g. Universities) to the backbone of the Internet quite often experience round-trips approximately 20-100ms, therefore due to the low latency of the network, faster retransmissions can be beneficial [Cheshire00].

The goal of this test was to reconfigure the TCP retransmission timer over Wireless networks to react speedier to errors rather than assume congestion and 'back-off' incorrectly. Thus we ran a series of comparisons between a Chameleon TCP Reno clone and a Chameleon TCP with a timeout value set to 150ms for Chameleon TCP and a time-out value of 400ms for TCP Reno**.**

The packet losses were simulated by a LOSSY protocol stack element. The LOSSY component simulated reordering and loss of messages (0% loss to 50% loss). We performed a series of 1000 runs of epoch sizes varying from 100-900 over our three-test bed networks. We used a Windows 98 Pentium Pro 500mhz PC Client, with 128 MB Ram connected via a 2 MB wireless LAN to a Windows 2000 Pentium III 800mhz Server with 128 MB RAM. Figure Error! No text of specified style in document.-3 illustrates that our optimised TCP outperforms a standard TCP stack when faced with losses above a 20% threshold. This is achieved through the use of a retransmission timer, which is set to respond faster to lost messages.

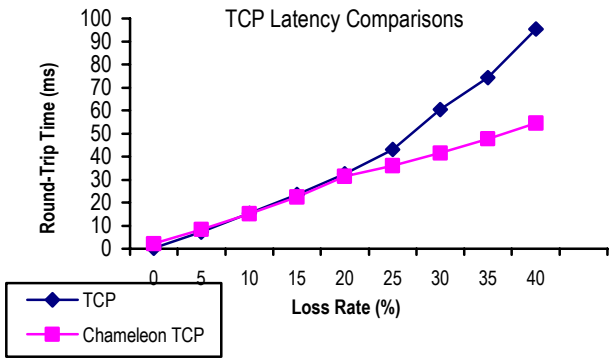

Figure **Error! No text of specified style in document.**-3 : TCP v Optimized TCP over Wireless LAN

Determining whether the application is streaming over a wired or on a wireless network can be determined at run-time by Chameleon. The assumption on the average packet loss rate for these networks is hidden in the protocol library for generic protocol stacks. Chameleon however provides the hooks for reconfiguring the environmental parameters involved. The retransmission_interval can be set in construction of the stack as follows:

IPMULTICAST (ttl=8):FRAG(Size = 2048):NAK(epochsz=32, retransmission_interval=150)

Chameleon does not require a total reloading of the stack into memory during run-time should a reconfiguration of the timer be requested. The timer interval is simply changed, buffers emptied and transmission continues as before with the new timer interval settings.

V. RELATED WORK

Problems with RSVP and other end-to-end per-flow resource reservation techniques has been well documented in the literature with regards requiring core routers to maintain individual flow states leading to scaling problems (not to mention actual deployment) [Kojo97]. In addition, existing middleware [Maffeis97, Joesph97] and multicast protocol approaches [Liljeberg96] tend to concentrate on the delivery of efficient narrowly defined solutions at the expense of generic adaptable frameworks which can be applied in wider ranging domains to cope with existing and unforeseen occurrences. Hierarchical methods [Snoeren01] do not allow for fine-grain delivery control and the ability for additional intelligence to be 'injected' into the data path to increase the transmission rate [Poger97, Zhao01]. Reflective approaches [Kojo95, Zenel95] require the use of non-standard Java virtual machines which again prevents the ready deployment of applications (which could make use of existing standard java virtual machines in routers, web browsers, JDK's etc). Proxy solutions [Todd99] are often written in slow scripting languages (hampering performance) or the filters are part of the application complicating their reuse and making it difficult to support legacy applications. Mobile frameworks [VanSteen99] seem to focus on issues such as mobile hosts migrating to

another network location, at the expense of a more complete middleware framework thus limiting users in their application development.

VI. CONCLUSIONS

The results of our research show that middleware architectures with carefully crafted generic interfaces and adaptable self-configuring traits are ideal for systems with fluctuating environment conditions. The goal was to create a framework where mobile devices can subscribe to an optimal quality of service using the minimal selection of stack components necessary and continue execution in the face of fluctuating conditions in the network and device itself. This was enabled through a platform independent and flexible framework where new components can be inserted and installed within live systems. We provide a series of template classes to aid future developers in creating mobile aware streaming solutions.

Sources and receivers are loosely coupled not needing to know the other identities with the only common object that both are aware of being a communication channel. This promotes scalability towards very large numbers of receivers and reduces the complexity of channel establishment. Receivers can have different needs, and satisfy them by individually tailoring the media streams extracted from a channel where each end system should receive the amount of data that it is capable of handling i.e. for each end system the highest possible quality is aimed for. This also helps overcome the receiver window size limiting the amount of data that can be buffered at the proxy as seen in other systems due to the decoupling of the server, proxy and client. The architecture meet the goal of creating an extensible framework where new codecs and QoS modules can be simply added without significantly perturbing any current running system.

VII. REFERENCES

[Cheshire00] Cheshire, S. For every network service there's an equal and opposite network disservice. http://rescomp.stanford.edu/~cheshire/rants/networkdynamics.html

[Joesph97] A. D. Joseph, J. A. Tauber, and M. F. Kaashoek. Mobile computing with the Rover toolkit. IEEE Transactions on Computers, 46(3):337-352, March 1997.

[Kojo97] Kojo M., Raatikainen K., Liljeberg M., Kiiskinen J., Alanko T.: An Efficient Transport Service for Slow Wireless Telephone Links. IEEE Journal on Selected Areas in Communications, Vol. 15, No. 7, September 1997.

[Maffeis96] Maffeis Silvano, Bischofberger Walter, and Mätzel Kai-Uwe. GTS: A Generic Multicast Transport Service to Support Disconnected Operation. ACM Wireless Networks Journal, 2, 1, pp. 87- 96, 1996

[Maffeis97] Maffeis, S. Building Reliable Distributed Systems with CORBA, Theory and Practice of Object Systems, Vol. 3(1), John Wiley & Sons, April 1997

[Modiano99] Eytan Modiano. An adaptive algorithm for optimizing the packet size used in wireless ARQ protocols. MIT Lincoln Laboratory, Lexington, MA 02420-9108, USA. Wireless Networks, Vol (5), No 4 (1999)

[Parr-Curran00] Parr, G., Curran, K. A Paradigm Shift In The Distribution Of Multimedia. Communications Of The ACM, Vol 43, No 6, pp 103-109, June 2000

[Peterson91] Peterson, L. & Hutchinson, N. The X-Kernal: An architecture for implementing network protocols, IEEE Transactions on Software Engineering, 17 (1), pp. 64-76 1991

[Poger97] Elliot Poger and Mary Baker, Secure Public Internet Access Handler (SPINACH)**.** Proceedings of the USENIX Symposium on Internet Technologies and Systems, December 1997.

[SCTE00] Society Of Cable Telecommunications Engineers, Inc. Engineering Committee. Audio codec requirements for the provision of bi-directional audio service over cable television networks using cable modems. Data Standards Subcommittee Document: SCTE DSS-00-01 Date of Original Issue: March 1, 2000 Date of Latest Revision: December 15, 2000

[Snoeren01] Snoeren, Alex., Balakrishnan, Hari, and Kaaeshoek, Frans. Reconsidering IP Mobility. In proceedings of the 8th IEEE workshop on Hot Topics in Operating Systems (HoTOS-VIII), Schloss Elmau, Germany, May 2001.

[VanSteen99] Van Steen, M., Tanenbaum and Homberg, P. Globe: A wide-area distributed system. IEEE Concurrency, pages 70-78, January-March 1999

[Unix90] Streams programmer's guide. Unix System V Release 4.

[Wong01] Wong, G., Hiltunen, M., and Schlichting, R. "A configurable and extensible transport protocol". IEEE Infocom 2001, April 22-26 2001, Anchorage, Alaska, April 2001