Multimedia Tools and Applications, 20, 225–236, 2003

An End to End Adaptable Architecture for Streaming Media over IP Networks

KEVIN CURRAN kj.curran@ulster.ac.uk
Internet Technologies Research Group, Northern Ireland Knowledge Engineering Laboratory, University of Ulster, Magee Campus, Northern Ireland, UK

GERARD PARR gp.parr@ulster.ac.uk
Internet Technologies Research Group, Northern Ireland Knowledge Engineering Laboratory, University of Ulster, Coleraine Campus, Northern Ireland, 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.

Micro-protocols attempt to eradicate this bottleneck by optimising the protocol stack to the functionality that is actually required by the application. A side effect of this is that it allows a device such as a PDA to offer protocol functions, which would not normally be available due to its memory constraints achievable by downloading necessary micro-protocols for new environments and discarding previous micro-protocols. Multicast media groups overcome the heterogeneous client problem where clients subscribe to different quality of services in accordance with resource availability and move between groups according to bandwidth availability.

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 primary quality transformation technique (PQT) of the middleware which enables clients to subscribe to media groups in accordance with available resources and network capacity.

Keywords: middleware, QoS, adaptive protocol stacks

1. Introduction

Mobile communications is a continually growing sector in industry and a wide variety of visual services such as video-on-demand have been created which are limited by low-bandwidth network infrastructures. 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. As 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.

2. Streaming media to end-hosts

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 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.

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 [1] and STREAMS [2]). 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.

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 [3]. 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 1024×800 while others may only manage black and white 100×60 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.

3. The chameleon framework

Chameleon is a Middleware Framework, which supports reconfigurable dissemination oriented communication. It extends earlier research-the RWANDA framework [4]—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.

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

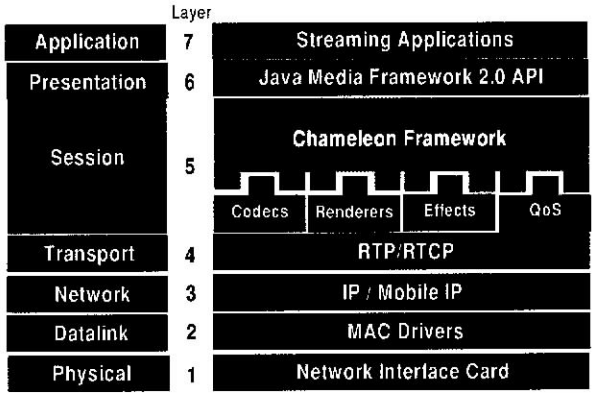

Figure 1. Chameleon in comparison to the OSI model.

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 [5]. 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 [6].

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 $G_i, i = 1, 2, \ldots, K$. 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 (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.

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 [7]. 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 [7]. 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.

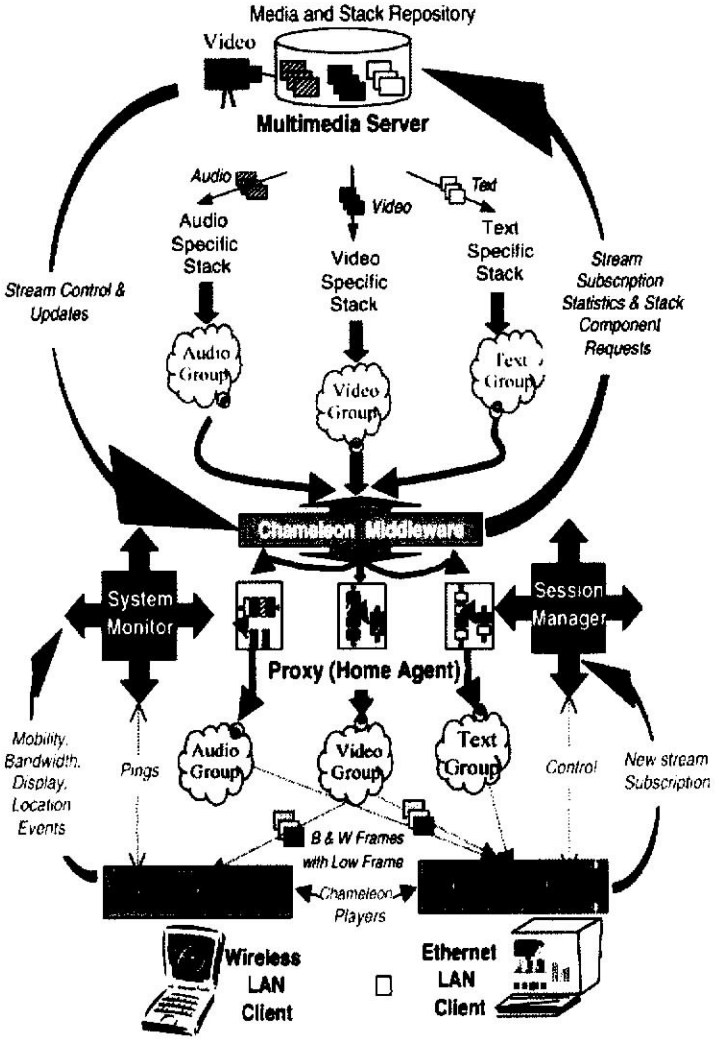

Figure 2. Multiple multicast multimedia groups.

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.

4. Evaluation of PQT

PQT has been evaluated using simulation. The basis for our simulation work is the LBL Network Simulator, 'ns', developed by the Network Research Group at the Lawrence Berkeley National Laboratory. NS provides a framework for inspecting the dynamic behaviour of network traffic, congestion, and other network characteristics. We simulate a multicast scenario of a sender with seven multicast streams (3 audio, 3 video and 1 FTP stream) to a receiver over different test bed networks with varying network bandwidth capabilities. We are primarily interested in the issue of rate control, however, as we are working on a best

effort network, we assume that the underlying network introduces data loss and error into our transmissions. We also assume that data can be delivered out of order to a receiver. We ignore the text component due to the small amount of bandwidth needed to transport supporting text.

In the Ethernet and WLAN test-beds, we modelled the audio traffic (G.711 and G.723) as CBR sources (with mean rates of 50 kbps, 100 kbps and 200 kbps respectively). In order to take the not negligible autocorrelation between successive video frames into account, we used traces of interactive and streaming video traffic as sources to our system. As the focus of the presented simulation test is on the obtained QoS of real-time traffic, the data traffic is modelled as worst-case background traffic. We used M/Pareto traffic ($\alpha = 1.2$, packet lengths 576 and 1500 Bytes) to characterise UDP/RTP compliant streams and FTP. Sources with poissonian arrival represent acknowledgements. Video streams were transmitted at rates of 600 K, 1200 K and 1800 K. FTP was 400 K.

In order to monitor the subscription progress of clients to multiple multicast groups in the face of fluctuating qualities of service we run a series of streaming broadcasts to a client connected over varying simulated network topologies to monitor its multicast subscription progress in the face of fluctuating available throughput. The server transmits video to three multicast groups. The High Quality group (HQ) is connected at throughputs up to 1800 Kbps, the Medium Quality (MQ) group up to 1200 Kbps and the Low Quality (LQ) group at 600 kbps. The Audio HQ group was set to 200 Kbps, medium quality to 100 and low quality to 50 kbps.

The four scenarios, which were examined, were:

1. NO Primary Quality Transformation and NO Priority—no dynamic subscription movement between varying quality of service groups and no priority assigned to streams.
2. NO Primary Quality Transformation WITH Priority—no dynamic subscription movement between varying quality of service groups but priority assigned to audio and video streams.
3. Primary Quality Transformation and NO Priority—dynamic subscription to higher and lower qualities of service in accordance with available bandwidth but no priority assigned to either audio or video.
4. Primary Quality Transformation WITH Priority—subscription to high & low qualities of service in accordance with available bandwidth & priority assigned to audio/video streams to enhance quality.

However, due to space restrictions, we can only present the results of the WLAN simulation as follows.

4.1. Wireless lan pqt with priority

The PQT is now enabled with priority. This allows us to assign a priority level of 5 to the audio stream and a priority level of 4 to the video stream. FTP is assigned a priority level of 2. This effectively ensures that audio and video grab their portion of the bandwidth at the expense of FTP.

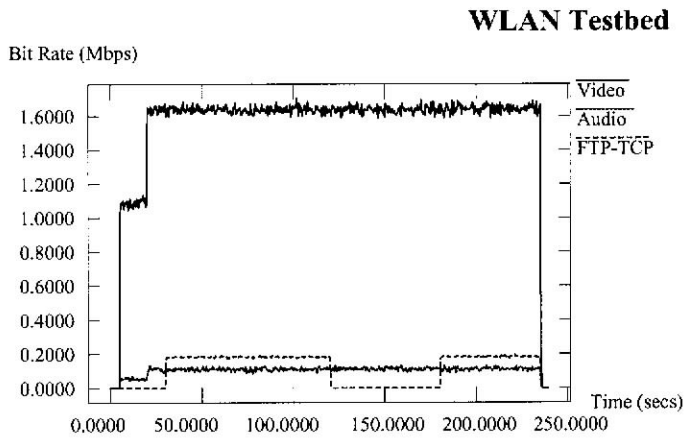

Figure 3. PQT with priority over WLAN.

PQT allows streams to move between quality group feeds in accordance with available bandwidth. Streaming of a MQ Video stream takes place at 1200 Kbps, MQ Audio at 100 Kbps and an FTP stream at 400 K. Streaming of a HQ video stream takes place at 1800 Kbps and an HQ audio at 200 Kbps as illustrated in figure 3. We see at time 20 seconds that the client moves to a higher quality video stream (1.8 Mbps) and the audio stream is increased to 200 Kbps at time 20 seconds also. Each of the FTP sessions is constrained to a max throughput of 200 Kbps. As opposed to the Ethernet PQT + Priority test case where the amount of bandwidth available negated any effects of PQT and priority, here we see the benefit of application level QoS where the quality of the audio and video stream can be prioritised and ultimately safe-guarded against 'rogue' third party streams.

Audio jitter averages out at 11 ms and video jitter at 13 ms. Jitter is relatively steady due to the amount of available bandwidth. Audio losses are negligible as too are the video losses. This method achieves the highest throughput for both audio and video streams over WLAN. This experiment demonstrates (in part) the effectiveness of using multiple multicast quality groups for streaming media to resource-constrained devices. The optimal scenario is where bandwidth is constrained and audio and video must compete with third party streams for bandwidth. PQT with priority is unnecessary where sufficient bandwidth is available for all streams such as an unloaded Ethernet. It is useful however where competition for bandwidth is necessary.

PQT is also desirable with some prioritised scheme as demonstrated earlier but it can also provide increased throughput wherever priorities are unable to be assigned. In addition a scheme such as PQT is a congestion friendly technique which aims to stream 'what is necessary' rather than simply 'dumping everything' onto the channel at the detriment of all other active receivers. We can summarise the effectiveness of PQT with and without priorities enabled over the following network types as follows:

Ethernet (assuming sufficient available bandwidth for all streams)—PQT with and without priority provides no additional benefit that best-effort streaming could not achieve.

Wireless LAN (assuming insufficient bandwidth for all competing streams)—PQT with and without priority allows priorities to be placed on specific streams and less crucial streams (e.g., File transfers) to be throttled in order to achieve an acceptable QoS. PQT serves a useful function in providing soft guarantees in bandwidth constrained wireless lans.

SLIP or GSM (assuming insufficient bandwidth for all competing streams)—PQT with and without priority allows priorities to be placed on specific streams and less crucial streams (e.g., File transfers) to be throttled in order to achieve an acceptable QoS. PQT serves an extremely useful function on severely limited bandwidth channels such as 28 K POTS or GSM, allowing streams of 'application attributed' higher importance to gain larger chunks of the channel.

5. 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) [3]. In addition, existing middleware [8] and multicast protocol approaches [9] 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 [10] 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 [11]. Reflective approaches [12] 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 [13] 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 [14] 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.

6. 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.

Abstract QoS interface functions achieve application transparency by assisting an efficient exchange of control and user data between applications and the middleware. This enables the new specification of functional requirements and agreement on application requirements in

terms of traditional QoS attributes, including numerical values for bandwidth, delay, or bit error rates. 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.

Notes

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

References

1. L. Peterson and N. Hutchinson, "The X-Kernal: An architecture for implementing network protocols," IEEE Transactions on Software Engineering, Vol. 17, No. 1, pp. 64–76, 1991.
2. Streams Programmer's Guide. Unix System V Release 4.
3. M. Kojo, K. Raatikainen, M. Liljeberg, J. Kiiskinen, and T. Alanko, "An efficient transport service for slow wireless telephone links," IEEE Journal on Selected Areas in Communications, Vol. 15, No. 7, 1997.
4. G. Parr and K. Curran, "A paradigm shift in the distribution of multimedia," Comms of the ACM, Vol. 43, No. 6, pp 103–109, 2000.
5. 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.
6. Society Of Cable Telecommunications Engineers. Audio codec requirements for the provision of bi-directional audio service over cable television networks using cable modems. Document: SCTE DSS-00-01 Date of Original Issue: March 1, 2000 Date of Latest Revision, 2000.
7. D.D. Clark and D.L. Tennenhouse, "Architectural considerations for a new generation of protocols," in Proc. ACM SIGCOMM'90, 1990, pp. 200–208.
8. S. Maffeis, "Building reliable distributed systems with CORBA, theory and practice of object systems," John Wiley, Vol. 3, No. 1, 1997.
9. A.D. Joseph, J.A. Tauber, and M.F. Kaashoek, "Mobile computing with the Rover toolkit," IEEE Transactions on Computers, Vol. 46, No. 3, pp. 337–352, 1997.
10. W. Heinzelman, Application-Specific Protocol Architectures for Wireless Networks. PhD Thesis, Massachusetts Institute of Technology, 2000.
11. Elliot Poger and Mary Baker, "Secure public internet access handler (SPINACH)," in Proceedings of the USENIX Symposium on Internet Technologies and Systems, 1997.
12. Golm, Michael and Kleinöder, Jürgen, MetaJava, Presented at STJA '97, 1997, Erfurt, Germany.
13. B. Zenel and D. Duchamp, " A general-purpose proxy filtering mechanism applied to the mobile environment," in Proceedings of MobiCom '97 Budapest, Hungary, 1997.
14. M. Van Steen, Tanenbaum, and P. Homberg, "Globe: A wide-area distributed system," IEEE Concurrency, 1999, pp. 70–78.

Kevin Curran is a Lecturer at the University of Ulster, Magee College. He is currently completing a Ph.D. in Computer Science with the University of Ulster. His research interests include distributed computing

especially emerging trends within wireless ad-hoc networks, distributed objects, dynamic protocol stacks, multimedia transport protocols and mobile systems. He can be contacted at kj.curran@ulst.ac.uk.

Gerard Parr is a professor of Telecommunications at the University of Ulster, Coleraine. He is also the Director of the Telecommunications & Distributed Systems Research Group. His research interests include ISDN Systems and Standards, Queueing Systems, Asynchronous Transfer Mode Switch Fabric and Communications network protocols. He can be contacted at gp.parr@ulst.ac.uk.