Multimedia QoS Adaptation for Inter-tech Roaming

Cristian Hesselman, Henk Eertink and Arjan Peddemors
Telematica Instituut, The Netherlands
{hesselman, eertink, peddemors}@telin.nl

Abstract

We introduce a scalable application-level QoS adaptation service for roaming between wireless networks that are based on different technologies ('inter-tech' roaming). The service is part of a platform that supports the distribution of multimedia streams (e.g., a streamed TV channel) to mobile clients operating in a heterogeneous environment. Central to our approach is the notion of a service class, which is a domain-specific perceptual QoS level. Each domain in a wireless infrastructure must support a limited number of these service classes. Our adaptation service handles inter-tech roaming by handing a client off from one service class to another. In this paper, we focus on the design of the adaptation service's client-side components. They combine the loss characteristics of the client's network interfaces with configurable policies to decide when to initiate a handoff to a target service class and when to complete it. We conclude with some experimental results.

1. Introduction

The distribution of a multimedia stream to a large number of mobile users often involves a variety of client devices with different processing and communications resources [6, 11, 12, 29, 45]. In this setting, it is usually difficult to deliver a stream at a Quality of Service (QoS) level that is fine-tuned to the capabilities and the current resource availability (e.g., in terms of available network bandwidth) of individual mobile devices ('individual-best' QoS [44]). An extreme solution to this problem is to provide the same QoS level to all clients ('all-worst' QoS [44]), but this will usually yield a significant number of users receiving an unsatisfactory perceptual QoS.

We are developing a platform that strikes a balance between these two extremes (for more details, see [32]). The platform divides the coverage area of a wireless infrastructure into domains and restricts the amount of available 'QoS spectrum' in access domains to a small number of discrete perceptual QoS levels. We call these QoS levels service classes. Service classes are domain-specific in that domain administrators are responsible for defining and managing their nature, number and ordering. We feel that this approach scales well for individual domains and therefore for a future wireless system [45] as a whole. The price that we pay is that we cannot deliver 'individual-best' QoS levels.

In this paper, we present the design of our platform's QoS adaptation service. It is an application-level service that adapts the QoS that a client receives by handing it off to another service class. We focus on handoffs across different network technologies as a result of roaming. We assume a best-effort IP service.

The rest of this paper is organized as follows. In Section 2, we briefly explain the organization of our platform. In Section 3, we introduce the mechanism that we use to achieve handoffs between QoS levels. We discuss an implementation of this mechanism in Section 4 and some qualitative results of its performance in Section 5. We present our conclusions in Section 6.

2. Multiparty Sessions

Our platform allows a server to distribute a raw audio-video stream to two or more clients. As an example, consider an application that broadcasts a TV channel from server S_{tv} to clients C_1 through C_7 (see Figure 1).

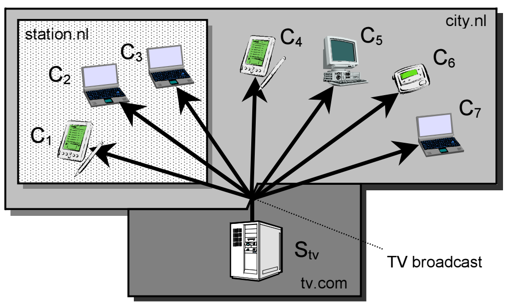

Figure 1. Television broadcast.

Figure 1 shows two domains that each operate a wireless network. Domain station.nl operates a LAN (short range, high speed) and city.nl operates a MAN (medium range, medium speed). We assume that each client C_i ($1 \leq i \leq 7$) is equipped with at least two network interfaces so that they can connect to station.nl's LAN as well as to city.nl's MAN. We assume that clients C_1 through C_3 receive the TV stream over their LAN interfaces, while C_4 through C_7 receive the stream over their MAN interfaces.

Figure 2 shows how our platform decomposes the TV broadcast of Figure 1. The players P_i in this figure model the presentation resources of a client C_i and consume the stream that source S produces. Our platform connects the players to the source through a *session*. A session is a *high-level notion of connectivity* that deals with the heterogeneity of client devices and networks. A session encapsulates type j decoders (D_j) and encoders (E_j), as well as proxies (X). Proxies [1, 4, 12, 14, 27, 28, 37, 39] perform functions such as rate adaptation, transcoding [10], audio and video filtering, and so on [13]. We assume that a proxy belongs to a domain (e.g., X_s of station.nl and X_c of city.nl) and runs on a *gateway* host at a domain border. A session also encapsulates multicast connections (labeled M) that interconnect decoders, encoders and proxies. We assume that an encoded audio-video stream is packetized using RTP [15, 36].

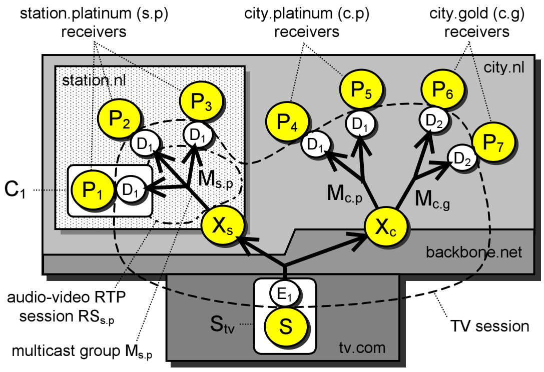

Figure 2. Decomposition of the TV broadcast.

The proxies in Figure 2 create domain-specific *service classes* from the stream they receive from S. A service class defines a *discrete perceptual QoS level* [13] of the raw audio-video stream that a player P_i receives (e.g., platinum quality video). The capabilities and the current resource availability of client C_i largely determine the service class that its player P_i will get. To allow our system to scale sessions up to large numbers of players, we propose that the administrator of each domain defines and manages *its own small number* of service classes. To emphasis that a service class is *domain-specific*, we denote it as domain.class (e.g., station.platinum) from now on.

Inside the platform, administrators define the requirements for a client to receive a service class in terms of an audio and video codec type (e.g., MPEG-4 [9]), a set of codec QoS characteristics (e.g. sampling rate, number of coding layers), a packetizer type (e.g., an RTP profile) and required transport-level QoS characteristics (e.g., in terms of bandwidth and loss). The platform associates a set of *site-local* RTP sessions with each service class. The size of such a set typically depends on the number of coding layers the domain administrator has configured for a service class [2, 3, 35, 38]. The platform realizes each RTP session as a site-local multicast group [8, 17] to maximize scalability. For simplicity, we assume one RTP session and one multicast group per service class in this paper (e.g., RTP session $RS_{s.p}$ and its multicast group $M_{s.p}$ for station.platinum).

Similar approaches that combine proxies and multicast groups for fixed communications can be found in [5, 7, 10, 31]. [6] uses proxies and multicast groups to provide reliable multiparty communications to mobile clients.

In the remainder of this paper, we concentrate on the issues involved in a player switching from one service class to another as a result of roaming between networks.

3. QoS Adaptation

The resources available to a mobile client typically fluctuate as a result of roaming, increased network load, and RF interference [30]. As a result, the QoS of the stream that the client's player receives needs to be *adapted* [34, 42, 43].

3.1 Inter-tech Service Class Handoffs

Our platform transfers a player to another service class whenever the QoS level of the stream that it receives no longer fits the available communications resources. We call this a *service class handoff*. As an example, consider player P_3 (Figure 2) and assume that the client device that hosts it (C_3) roams from station.nl to city.nl (Figure 3).

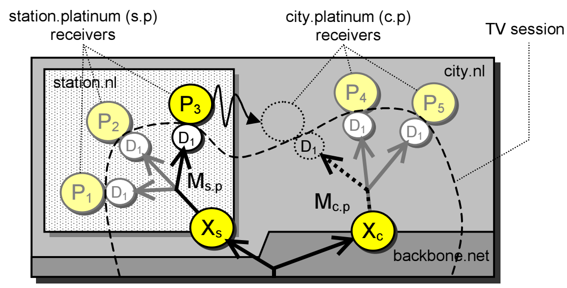

Figure 3. A service class handoff.

To perform the handoff for P_3, the platform unsubscribes C_3 from the multicast group associated with station.platinum ($M_{s.p}$) and joins it to the multicast group of city.platinum ($M_{c.p}$). The platform furthermore configures C_3's decoder to the QoS level of city.platinum.

Observe that our approach deals with device mobility at the application level. This is similar to the SIP solution for mobility [33]. It is however unlike the Mobile IP [19] approach that deals with device mobility at the IP level. Also notice that we cannot guarantee that a client has an invariant IP address at its disposal (cf. [19]) because we follow an application-level approach. Our handoff approach can therefore only be used for live and scheduled multimedia broadcasts that are transmitted over UDP.

We call the handoff of Figure 3 an *inter-tech* [27] service class handoff because it involves two different network technologies (cf. [4, 26, 41]). It is also an *inter-domain* [14] service class handoff because it transfers P_3 from the station to the city domain. Service class handoffs can also occur when a client roams in a network based on a single technology (intra-tech [40]) or within a single domain (intra-domain). Combinations of these types of handoffs are also possible, but we will not consider them in this paper.

In the example of Figure 3 we have assumed that there are other players in city.nl (P_4 and P_5) that already receive the TV stream at class platinum. A proxy (X_c) and a multicast group ($M_{c.p}$) that realize city.platinum are therefore already in place when C_3 roams into city.nl. If this had not been the case, our platform would have had to *dynamically* allocate these resources. A service class handoff may thus require our platform to dynamically create or destroy proxies and multicast groups. Observe that each domain must always be able to accommodate its lowest service class for incoming clients (cf. city.nl in Figure 3), preferably at different encodings for good client coverage.

The inter-tech service class handoffs that our platform supports are *mobile-controlled* [24]. In the example of Figure 3, this means that there is a *handoff component* on C_3 that decides whether a handoff to $M_{c.p}$ is required. This handoff component is an application-independent part of our platform.

The decision to handoff to another service class can be based on several metrics [29], for instance on the transmission delay on the paths between C_3 and the two proxies. Our platform uses the *packet loss* characteristics of these paths. We make use of *beaconing* [26] to determine these characteristics. In the example of Figure 3, this means that proxies X_s and X_c multicast beacon messages into their respective domains at proxy-specific intervals I_s and I_c. They include the domain they belong to as well as the interval that they use in their beacons and transmit the beacons onto a well-known multicast group M_b (cf. [5, 7]). Observe that RTCP [15] sender reports from the proxies could also be used to act as 'beacons'. However, RTCP requires the interval between subsequent reports to be at least 5 seconds. This would yield unacceptable handoff detection delays in our system.

Beacons act as input for the handoff component on the client. The handoff component consists of three sub-components that cooperatively detect and execute handoffs (see Sections 3.2 through 3.4 for more details):

- The *Network-specific Monitor* (NM). An instance of this component monitors a single network interface and analyzes its loss characteristics. There exists exactly one NM for each network interface of a mobile client. An NM determines if the quality of its associated network interface is currently 'acceptable', 'questionable' or 'unacceptable'.
- The *Interface Monitor* (IM). An IM selects the most appropriate interfaces based on the quality information maintained by the NMs. There exists exactly one IM in a mobile client.
- The *Content Switch* (CS). A CS realizes service class handoffs. It is aware of the codecs that a client supports and is capable of configuring them. A CS uses buffers to smooth the handoff process. It uses the information maintained by the IM to decide if a handoff is necessary. A CS resolves any ties if the IM indicates that multiple network interfaces can be used. A mobile client hosts one CS per application.

3.2 Network-specific Monitor

A *Network-specific Monitor* (NM) keeps track of the quality of a specific network interface. The input for an NM consists of beacon messages that it receives on M_b over the interface that it monitors. An NM uses mechanisms similar to those of Mobile IP [19, 25] and the ones described by Stemm *et al.* [26]. However, instead of keeping track of the number of *consecutively* lost beacons, we use a sliding 'averaging window' [23, 24] in which we maintain a history of lost and received beacon messages. We combine the averaging windows with threshold values. The result is a handoff mechanism that is based on common RF-level techniques [23, 24, 27], but applied on the application level.

We denote the NM for interface j as NM_j and the *averaging window* that it maintains as W_j. An entry in W_j represents a beacon and indicates whether the beacon was received or not. We denote the size of W_j as Z_j ($Z_j \geq 1$).

An NM uses *timeouts* to detect lost beacons. We denote the timeout value that NM_j uses as T_j. The advantage of timeouts over sequence numbers is that W_j gets updated at fixed (i.e., T_j) intervals, even when a lot of beacons are lost. The downside is that it introduces additional processing due to reoccurring timeouts when very few beacons are being received.

In addition to T_j, we also define a *rejoin timer* R_j for each NM_j. Whenever there is no connectivity to the infrastructure via interface j, NM_j attempts to join the mobile client to M_b on interface j every R_j seconds.

As an example, consider the high-speed LAN interface of C_3 and assume that C_3 is positioned well inside the station domain. We denote the associated NM as NM_l (j equals 'l' for 'LAN'). At start-up time, NM_l assumes that there is no connectivity to the infrastructure and initializes T_l to R_l. NM_l joins C_3 to M_b on the LAN interface every R_l seconds until it receives a beacon from X_s. When this happens, NM_l updates W_l and sets T_l to 1.5 * I_s. NM_l sets T_l to this value every time it receives a beacon to compensate for jitter in station.nl's network.

Next, assume that C_3 roams into the city domain. If NM_l does not receive a beacon from X_s within T_l seconds after the last beacon, it considers the beacon that it was supposed to receive lost. In this case, NM_l again updates W_l, but sets T_l to I_s rather than to 1.5 * I_s because it finds itself roughly in the middle of X_s' beacon interval.

Observe that the above behavior allows a beacon to experience half a beacon interval of jitter before NM_l considers it lost.

Each NM_j compares the entries in its window W_j with two interface-specific *threshold* parameters H_{j1} and H_{j2} ($0 \le H_{j1} < Z_j$, and $0 \le H_{j2} < Z_j$). We require H_{j2} to be larger than or equal to H_{j1}. This divides the loss range of W_j into three regions, each with its own quality label. If we assume that L_j denotes the number of lost beacons in W_j, these regions are:

- A region delimited by $0 \le L_j \le H_{j1}$ where the quality of interface j is *acceptable*;
- A region delimited by $H_{j1} < L_j \le H_{j2}$ where the quality of interface j is *questionable*; and
- A region delimited by $H_{j2} < L_j \le Z_j$ where the quality of interface j is *unacceptable*.

The values of H_{j1} and H_{j2} can be set by the application. The application can also control the window size Z_j and the value of the rejoin time R_j. It cannot control the beacon interval time and T_j because the domain administrator determines them. Using these policies, the application can configure allowable beacon losses and perceptual distortions. We will discuss this in more detail in Section 5.

Figure 4 shows that there are 4 typical signaling moments when roaming in and out of an overlay domain. In this case, client C_3 roams from station.nl to city.nl and back. As a result, the loss on C_3's LAN interface (L_l) first increases, and then decreases. The loss on C_3's MAN interface (L_m) remains relatively constant.

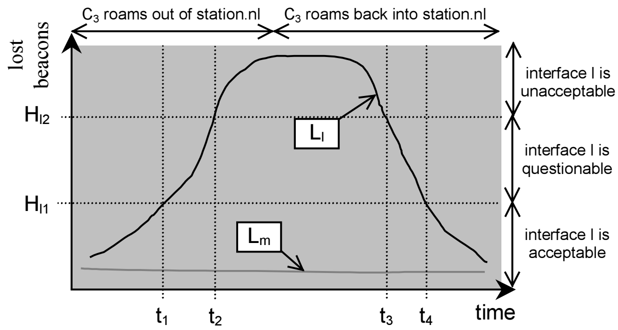

Figure 4. Abstract view of loss characteristics.

3.3 Interface Monitor

The *Interface Monitor* (IM) selects the most appropriate network interface. It bases its decision on the information maintained by the NMs. The IM can be partly configured by the application through a *policy*. An IM policy defines a total *preference ordering* across the network interfaces of a mobile client. A policy is usually easy to define, for instance by preferring a wireless LAN over Bluetooth over UMTS, etc.

In the following we assume that a certain mobile client has n ($n \ge 1$) network interfaces that are represented by an equal number of NMs. We also assume that the NMs are ordered as $NM_1...NM_n$ with NM_1 representing the interface with the highest preference and NM_n the one with the lowest preference.

To determine the most appropriate interface, the IM locates the NM_i ($i \ge 1$) with the highest preference whose interface provides an acceptable or questionable quality. The IM selects the interface of NM_i as the one to use if its quality is acceptable. However, if NM_i indicates that the quality of interface i is questionable, the IM also selects an alternative interface NM_j ($j > i$) that provides an acceptable or questionable quality. When the IM has found an alternative NM_j, it considers both NM_i and NM_j appropriate. In this case, the IM leaves it up to the CS to resolve the tie because this component has more application-level knowledge (e.g., about coding formats). The IM thus merely gives handoff *hints* in this situation.

As an example, assume that client C_3 is on the boundary of station.nl and city.nl (cf. Figure 3). At this point, player P_3 receives the audio-video stream at class station.platinum over multicast group $M_{s.p}$. When C_3 moves out of range of station.nl's LAN, NM_l ('l' for 'LAN') will start to loose the beacons that X_s transmits. The quality of NM_l will eventually become questionable at t_1. The quality of NM_m, ('m' for 'MAN') on the other hand, will very likely remain acceptable because the MAN of city.nl overlays the LAN of station.nl. As a result, the IM considers both of C_3's interfaces appropriate at t_1. This situation lasts until t_2. After that, the IM considers C_3's LAN interface unacceptable and sees the MAN interface as the only appropriate one.

When C_3 roams back into the station domain, the IM finds the MAN interface most appropriate until t_3, both interfaces during t_3-t_4 and the LAN interface from t_4 onwards.

3.4 Content Switch

A *Content Switch* (CS) hands the player of a client device off from one service class to another. It uses the information maintained by the IM to select a network interface to use. If this is not the interface that is currently in use, the CS selects a service class on the new interface and hands the player off to this class. To accomplish the handoff, the CS subscribes the client to the multicast group associated with the target service class on the new interface. It then starts to *buffer* the multimedia data that it receives on the target multicast group to smooth the handoff process. Once there is sufficient data in the buffer, the CS (re)configures the decoder that is appropriate for the target class and feeds the data in the buffer through the (new) decoder to the player. Finally, the CS unsubscribes the client from the multicast group over which it used to receive the multimedia stream.

In our roaming example, the CS may join a target multicast group and start to buffer the data that this group carries at all t_i in Figure 4 (i = 1, 2, 3, 4). Similarly, there are also various points at which the CS can complete a handoff (i.e., start to feed the data in the target buffer through the decoder to the display of the client). In particular, these points are t_2, t_4, or as soon as there is sufficient data in the buffer associated with the target multicast group. We use a *policy* to define the points where the CS joins the client to the target multicast group and where it completes a handoff. Table 1 shows what policies are possible.

	from $M_{s.p}$ to $M_{c.p}$		from $M_{c.p}$ to $M_{s.p}$	
Policy	join $M_{c.p}$ at	HO to $M_{c.p}$ at	join $M_{s.p}$ at	HO to $M_{s.p}$ at
1	t_1	Asap	t_3	asap
2	t_1	Asap	t_3	t_4
3	t_1	Asap	t_4	asap
4	t_1	t_2	t_3	asap
5	t_1	t_2	t_3	t_4
6	t_1	t_2	t_4	asap
7	t_2	Asap	t_3	asap
8	t_2	Asap	t_3	t_4
9	t_2	Asap	t_4	asap
HO = handoff; t_i (i = 1, 2, 3, 4) correspond to the t_i in Figure 4				

Table 1. CS handoff policies.

Policies may be classified. For instance, policies 4 through 9 may be classified as 'greedy' because they attempt to stick with $M_{s.p}$ as long as possible.

In Section 5, we will discuss policy number 9 in more detail. At that point, we will also present some quantitative results of experiments with this policy.

4. Implementation

We implemented the handoff components and mechanisms in our testbed (see Figure 5). Figure 5 also illustrates how the proxies and the player of Figure 3 are distributed over the machines in the testbed.

The Solaris server hosts proxies X_s and X_c. For simplicity, we have implemented X_s and X_c to act as servers. That is, they generate the stream containing the TV channel locally rather than from a stream coming from server S (cf. Figure 2). X_s and X_c each consist of a QuickTime Darwin streaming server [18]. In Figure 5, they are labeled S_s and S_c, respectively.

S_s and S_c run synchronously as indicated by the arrow between them and loop continuously. S_s locally reads a high quality movie from a hinted (i.e., encoded and RTP-packetized) QuickTime file and transmits it onto the multicast group that represents class station.platinum, $M_{s.p}$. Similarly, S_c locally reads a low quality version of the same movie from a different hinted file and transmits it onto the multicast group that represents city.platinum, $M_{c.p}$. X_s and X_c each also contain a process (not shown in Figure 5) that multicasts beacon messages onto M_b every I_s and I_c seconds, respectively.

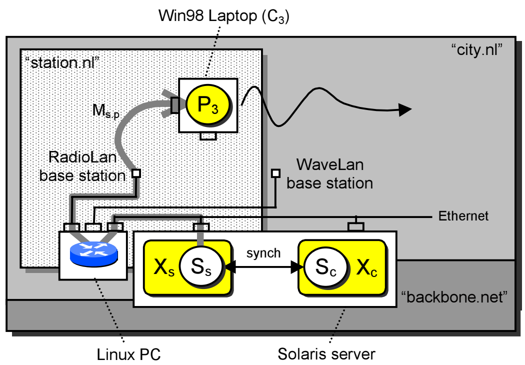

Figure 5. Testbed.

The RadioLan [22] and WaveLan [21] base stations mimic the LAN of station.nl and the MAN of city.nl, respectively. The RadioLan base station provides a gross bandwidth of 10 Mbps and has an indoor range of approximately 15 meters. The WaveLan base station offers a 1 Mbps gross bandwidth at a range of approximately 30 meters. In our testbed, the WaveLan cell completely *overlays* the RadioLan cell.

Our client runs on a standard Windows laptop. It uses the QuickTime client software package [20] to receive the streams that servers S_s and S_c transmit. The QuickTime package exposes a Java API through a thin wrapper that our CS component uses to realize inter-tech service class handoffs. We have implemented the CS, the IM and two NMs as separate Java threads. One NM (NM_r) monitors the loss characteristics of the laptop's RadioLan interface, the other (NM_w) that of the WaveLan interface. Observe that when the client has roamed out of the RadioLan network, it receives its stream from S_c over $M_{c.p}$ (WaveLan) as a result of a service class handoff.

5. Experiments

We have conducted experiments with several policies. However, in this paper we only consider policy 9 of Table 1 (chosen because of its asymmetry). Other policies show other handoff delays, but work similarly. In the experiments that we discuss here, we set the IM policy to favor the RadioLan network over the WaveLan network. We also froze the policy for NM_w since we do not consider roaming out of the WaveLan coverage area. Finally, we set the beacon interval of both proxies to 100 ms. In the following, L_r and L_w denote the number of lost beacons as seen by NM_r and NM_w, respectively.

Figure 6 shows the behavior of L_r and L_w when the window size for RadioLan is 20 (Z_r) and its thresholds are set to 5 (H_{r1}) and 15 (H_{r2}). As the client roams out of the RadioLan network, L_r first becomes larger than H_{r2} (i.e., of an unacceptable quality) at t_2'. At this point, the IM informs the CS that the WaveLan interface is the only appropriate interface. As a result, the CS initiates a handoff from station.platinum to city.platinum (policy 9). It therefore orders QuickTime to join the laptop to $M_{c.p}$ and to begin buffering data. However, L_r fluctuates around H_{r2} at this stage and drops back to H_{r2}. The IM detects this and informs the CS that the RadioLan interface can be used again. In response, the CS cancels the handoff in progress. At t_2, L_r again becomes larger than H_{r2} and the CS initiates a handoff for the second time (policy 9). This time, L_r stays above the threshold. As soon as QuickTime is done buffering, the CS completes the handoff by putting the stream from $M_{c.p}$ on screen (policy 9). In Figure 6, this is at t_{e1}. We consider the handoff to have begun at t_{b1} when NM_r (RadioLan interface) missed the first of the last $H_{r2} + 1$ beacons it lost at t_2, so the total handoff delay in this experiment equals $t_{e1} - t_{b1}$ = 5.12 seconds. This includes the time it takes IP multicast to establish a route to the laptop as well as QuickTime's initialization and buffering delay. It must be noted that QuickTime is responsible for a large portion (almost 70%) of the total handoff time.

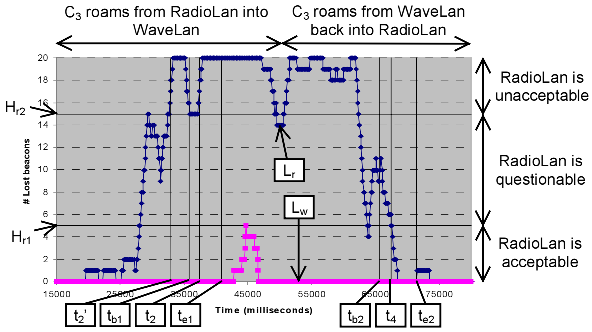

Figure 6. CS policy 9 (see Table 1).

Similarly, the handoff from city.platinum back to station.platinum (right side of Figure 6) starts at t_{b2}. The CS initiates the handoff at t_4 by joining the client to $M_{s.p}$ (policy 9). It completes the handoff (at t_{e2}) when QuickTime is done buffering (policy 9) by putting the stream from $M_{s.p}$ on screen. The total handoff delay in this case is $t_{e2} - t_{b2}$ = 5.88 seconds.

Both handoffs resulted in perceptual distortions. When roaming out of the RadioLan coverage area, the play out of the stream from S_s became rather bumpy some time before the CS put the stream from S_c on screen. On the way back, we also noticed some distortions due to the behavior of the QuickTime libraries. They apparently stop buffering data after a fixed amount of time, even when they have not yet received enough data to display the stream from S_s. Unfortunately, the QuickTime API does not allow us to change this. The problem can be however be overcome with another NM policy. In a subsequent experiment, we therefore set NM_r's lower loss threshold (H_{r1}) to 2. In this case, the CS does not join the multicast group of the WaveLan network ($M_{c.p}$) until there are 2 or less lost beacons in NM_r's averaging window. This usually means that the client has already received more data on its RadioLan interface, which fixes this problem.

6. Conclusions and Future Work

We presented a platform that revolves around the notion of domain-specific application-level service classes. Our QoS adaptation service dynamically hands off clients from one service class to another, for instance as a result of inter-tech roaming. The adaptation service can be configured through various policies that can be set by the application to define the moments of handoff initiation and completion. We also discussed the results of handoff experiments that we have conducted using different policies. Depending on the chosen policy, we found that an inter-tech service class handoff can be realized in a perceptually smooth manner. We also found that handoff delays of our implementation are primarily determined by the (non-configurable) initialization and buffering time of the QuickTime package. Handoff detection is however fast and effective.

We believe that our handoff policies are useful for developers of mobile multimedia applications. They allow them to configure our platform's logic by selecting the handoff behavior that best matches their application.

Our future work will deal with the *establishment* of configurations such as the one shown in Figure 3. In particular, we will concentrate on the protocols that assign a service class to a client based on its capabilities (e.g., screen size) and available resources (e.g., bandwidth). We will also investigate the extensions to support inter and intra-domain roaming.

References

[1] S. Cho, Y. Shin, "Multimedia Service Interworking over Heterogeneous Networking Environments", IEEE Network, March/April 1999

[2] S. McCanne, V. Jacobson, M. Vetterli, "Receiver-driven Layered Multicast", Proc. of ACM SIGCOMM, Stanford, USA, August 1996

[3] X. Li, M. Ammar, S. Paul, "Video Multicast over the Internet", IEEE Network, March/April 1999

[4] E. Brewer, R. Katz, Y. Chawathe, S. Gribble, T. Hodes, G. Nguyen, M. Stemm, T. Henderson, E. Amir, H. Balakrishnan, A. Fox, V. Padmanabhan, S. Seshan, "A Network Architecture for Heterogeneous Mobile Computing", IEEE Personal Communications, Oct. 1998

[5] E. Amir, S. McCanne, R. Katz, "An Active Service Framework and its Application to Real-time Multimedia Transcoding", Proc. of ACM SIGCOMM'98, Vancouver, Canada, Sept. 1998

[6] Y. Chawathe, S. Fink, S. McCanne, E. Brewer, "A Proxy Architecture for Reliable Multicast in Heterogeneous Environments", Proc. of ACM Multimedia'98, Bristol, UK, Sept. 1998

[7] K. Jonas, M. Kretschmer, J. Moedeker, "Get a KISS — Communication Infrastructure for Streaming Services in a Heterogeneous Environment", Proc. of ACM Multimedia'98, Bristol, UK, Sept. 1998

[8] D. Meyer, "Administratively Scoped IP Multicast", RFC 2365, July 1998

[9] R. Koenen, "MPEG-4 — Multimedia for Our Time", IEEE Spectrum, Feb. 1999

[10] E. Amir, S. McCanne, H. Zhang, "An Application Level Video Gateway", Proc. of ACM Multimedia, San Fransisco, USA, Nov. 1995

[11] A. Fasbender, F. Reichert, E. Geulen, J. Hjelm, T. Wierlemann, "Any Network, Any Terminal, Anywhere", IEEE Personal Communications, April 1999

[12] A. Fox, S. Gribble, E. Brewer, E. Amir, "Adapting to Network and Client Variability via On-Demand Dynamic Distillation", ASPLOS-VII, Oct 1996

[13] N. Yeadon, F. Garcia, D. Hutshison, D. Shepherd, "Filters: QoS Support Mechanisms for Multipeer Communications", IEEE Journal on Selected Areas in Comm., Sept. 1996

[14] A. Balachandran, A. Campbell, M. Kounavis, "Active Filters: Delivering Scaled Media to Mobile Devices", 7th Int. Workshop on Network and Operating System Support for Digital Audio and Video (NOSSDAV'97), St. Louis, USA, May 1997

[15] H. Schulzrinne, S. Casner, R. Frederick, V. Jacobson, "RTP: A Transport Protocol for Real-Time Applications", RFC 1889, Jan. 1996

[16] X. Xiao, L. Ni, "Internet QoS: A Big Picture", IEEE Network, March/April 1999

[17] D. Lee, D. Lough, S. Midkiff, N. Davis, P. Benchoff, "The Next Generation of the Internet: Aspects of the Internet Protocol Version 6", IEEE Network, Jan/Feb 1998

[18] Darwin Streaming Server, http://www.publicsource.apple.com/projects/streaming/

[19] J. Solomon, "Mobile IP — The Internet Unplugged", Prentice Hall, 1998

[20] Apple QuickTime Client 4.1, http://developer.apple.com/quicktime/

[21] WaveLan homepage, http://www.wavelan.com

[22] RadioLan homepage, http://www.radiolan.com

[23] G. Pollini, "Trends in Handover Design", IEEE Communications Surveys, http://www.comsoc.org/pubs/ surveys/pollini/pollini-org.html

[24] N. Tripathi, J. Reed, H. VanLandingham, "Handoff in Cellular Systems", IEEE Personal Communications, Dec. 1998

[25] A. Seneviratne, B. Sarikaya, "Cellular Networks and Mobile Internet", Computer Communications, Sept. 1998

[26] M. Stemm, R. Katz, "Vertical Handoffs in Wireless Overlay Networks", ACM Mobile Networking, Special Issue on Mobile Networking and Internet, Spring 1998

[27] K. Pahlavan, P. Krishnamurthy, A. Hatami, M. Ylianttila, J. Makela, R. Pichna, J. Vallström, "Handoff in Hybrid Mobile Data Networks", IEEE Personal Communications, April 2000

[28] B. Zenel, D. Duchamp, "A General Purpose Proxy Filtering Mechanism Applied to the Mobile Environment", Proc. 3rd ACM/IEEE International Conference on Mobile Computing and Networking, Budapest, Hungary, Sept. 1997

[29] U. Varshney, R. Vetter, "Emerging Mobile and Wireless Networks", Communications of the ACM, June 2000

[30] A. Campbell, "A Research Agenda for QOS-aware Mobile Multimedia Middleware", NSF Wireless and Mobile Communications Workshop, Virginia, USA, March 1997

[31] I. Kouvelas, V. Hardman, J. Crowcroft, "Network Adaptive Continuous-Media Applications Through Self Organised Transcoding", Proc. of Network and Operating Systems Support for Digital Audio and Video (NOSSDAV'98), July 1998, Cambridge, UK

[32] C. Hesselman, H. Eertink, "A Scalable QoS Adaptation Service for Mobile Multimedia Applications", 6th EUNICE Open European Summer School, Sept. 2000, Enschede, The Netherlands, http://wwwtgs.ctit.utwente.nl/Docs/ eunice/summerschool/papers/programme.html

[33] E. Wedlund, H. Schulzrinne, "Mobility Support Using SIP", 2nd ACM/IEEE International Conference on Wireless and Mobile Multimedia (WoWMoM'99), Seattle, USA, Aug. 1999

[34] D. Chalmers, M. Sloman, "A Survey of Quality of Service in Mobile Computing Environments", IEEE Communications Surveys, http://www.comsoc.org/pubs/surveys, 2nd Quarter 1999

[35] L. Wu, R. Sharma, B. Smith, "Thin Streams: An Architecture for Multicast Layered Video", 7th Intl. Workshop on Network an Operating Systems Support for Digital Audio and Video (NOSSDAV97), St. Louis, USA, May 1997

[36] K. Brown, S. Singh, "Extensions to RTP to support Mobile Networking", 3rd Intl. Workshop on Mobile Multimedia Communications, Princeton, USA, Sept. 1996

[37] A. Kassler, A. Neubeck, P. Schulthes, "Filtering Wavelet Based Video Streams for Wireless Interworking", IEEE Intl. Conference on Multimedia and Exposition 2000 (ICME2000), New York, USA, August 2000

[38] H. Schulzrinne, S. Casner, R. Frederick, V. Jacobson, "RTP: A Transport Protocol for Real-Time Applications", Internet draft, draft-ietf-avt-rtp-new-08.txt, July 2000

[39] J. Meggers, T. Strang, A. Park, "A Video Gateway to Support Video Streaming to Mobile Clients", ACTS Mobile Communications Summit 1997, Aalborg, Denmark, Oct. 1997

[40] A. Campbell, "Mobiware: QoS-Aware Middleware for Mobile Multimedia Communications", 7th IFIP International Conference on High Performance Networking, White Plains, NY, April 1997

[41] J. Meggers, A. Park, R. Ludwig, "Roaming between GSM and Wireless LAN", ACTS Mobile Summit, Granada, Spain, Nov. 1996

[42] X. Wan, H. Schulzrinne, "Comparison of Adaptive Internet Multimedia Applications", IEICE Transactions on Communications, June 1999

[43] G. Fankhauser, M. Dasen, N. Weiler, B. Plattner, B. Stiller, "The WaveVideo System and Network Architecture: Design and Implementation", Technical Report, Zürich, Switzerland, June 1998

[44] L. Delgrossi, C. Halstrick, D. Hehmann, R. Herrtwich, O. Krone, J. Sandvoss, C. Vogt, "Media Scaling for Audiovisual Communication with the Heidelberg Transport System", ACM Multimedia, Anaheim, USA, August 1993

[45] M. Haardt, W. Mohr, "The Complete Solution for Third-Generation Wireless Communications: Two Modes on Air, One Winning Strategy", IEEE Personal Comm., Dec. 2000