Mobile Streaming Media CDN Enabled by Dynamic SMIL

Takeshi Yoshimura[1], Yoshifumi Yonemoto[1], Tomoyuki Ohya[1],
Minoru Etoh[1], and Susie Wee[2]

[1] NTT DoCoMo, Multimedia Laboratories
3-5, Hikarinooka, Yokosuka,
Kanagawa, 239-8536, JAPAN
{yoshi, yonemoto, ohya, etoh}@spg.yrp.nttdocomo.co.jp

[2] Hewlett-Packard Laboratories
1501 Page Mill Road
Palo Alto, CA 94304-1126
swee@hpl.hp.com

ABSTRACT

In this paper, we present a mobile streaming media CDN (Content Delivery Network) architecture in which content segmentation, request routing, pre-fetch scheduling, and session handoff are controlled by SMIL (Synchronized Multimedia Integrated Language) modification. In this architecture, mobile clients simply follow modified SMIL files downloaded from a streaming portal server; these modifications enable multimedia content to be delivered to the mobile clients from the *best* surrogates in the CDN. The key components of this architecture are 1) content segmentation with SMIL modification, 2) on-demand rewriting of URLs in SMIL, 3) pre-fetch scheduling based on timing information derived from SMIL, 4) SMIL updates by SOAP (Simple Object Access Protocol) messaging for session handoffs due to clients mobility. We also introduce QoS control with a network agent called an "RTP monitoring agent" to enable appropriate control of media quality based on both network congestion and radio link conditions. The current status of our prototyping on a mobile QoS testbed "MOBIQ" is reported in this paper. We are currently designing the SOAP-based APIs (Application Programmable Interfaces) needed for the mobile streaming media CDN and building the CDN over the current testbed.

Categories and Subject Descriptors

C.2.1. [**Computer-Communication Networks**]: Network Architecture and Design – *distributed networks, store and forward networks, wireless communication.*

General Terms

Management, Design

Keywords

mobile network, streaming media, CDN, SMIL

Approximate word count

7,300 words

WWW2002, May 7-11, 2002, Honolulu, Hawaii, USA.
ACM 1-58113-449-5/02/0005.

1. INTRODUCTION

With the birth and growth of the Internet and high-speed access links, Internet users can enjoy large amounts of web content on the Internet. However, network congestion and server overload have become major concerns for content delivery. To mitigate these effects, CDNs (Content Delivery Networks) are attracting a great deal of attention. In CDNs, web content is distributed to cache servers located at the edge of the Internet close to clients, and the cache servers deliver content to clients if they cached the requested content beforehand. Since the cached content does not have to be delivered from the origin servers to the cache servers, this leads to lower latency for users, better network resource utilization for network operators, and scalable service provisioning for content providers. At the moment, several commercial companies provide CDN service on worldwide scales[1][2].

In mobile environments, Internet access has become very popular as represented by i-mode[3] and WAP (wireless access protocol)[4]. In addition, IMT-2000 (International Mobile Telecommunications-2000) has begun in Japan[5], and users can now access the Internet at up to 384 kbit/s through mobile devices. In the near future, mobile CDNs will be needed with the increasing number of IMT-2000 users.

At the same time, multimedia streaming services are becoming popular over the Internet. Mobile networks are also expected to provide IP-based multimedia streaming services in addition to web access services. From this requirement, 3GPP (3rd Generation Partnership Project), a standardization body of IMT-2000, has defined "packet streaming service (PSS) specifications[6]" as a standard for streaming services over 3G mobile networks.

Streaming services, however, present a lot of challenges for network engineers. Unlike TCP applications, streaming services require a certain amount of bandwidth to ensure the bit-rate needed by each media stream and the strict delay variation (i.e., jitter) needed to avoid buffer underflow at streaming clients. For such streaming services, CDNs are a very effective solution to relieve network congestion and to keep jitter at tolerable levels. Several papers have already considered how to cache streaming content effectively. In [7][8][9], streaming content is encoded with layered coding techniques and a proxy cache manages the number of layers to be stored for each stream. [10] proposed a prefix caching scheme that stores only the beginning of the content to minimize storage while decreasing waiting time for beginning playback. [11] also utilizes the idea of prefix caching and describes resource management issues on cache systems. Selective caching was proposed in [12] as a generalization of

prefix caching by storing various segments of the stream, specifically those that are subject to causing buffer underflow at clients. In [13], streaming content is divided into variable-sized segments in order to improve caching efficiency. Content-aware segmentation and a cache system using video summarization were proposed in [14]. In this system, streaming content is segmented at key frames and users can start streaming from any segment while watching the key frames.

Although these segment caching technologies have potential for enhancing network resource utilization and caching efficiency, they do not describe how to manage the segmented content and how to specify the locations of the proxy caches. Especially in mobile networks, since network and radio link conditions are dynamically changed depending on the time and place, it is required to specify the best cache location flexibly. In addition, the best cache location could be changed during a streaming session when a mobile client moves from one location to another location. The mechanism to update the cache location is necessary for a mobile streaming media CDN. In [15], a streaming CDN architecture that has content naming, content management, and redirection mechanisms is described, but it does not consider network dynamics and client mobility expected in mobile networks.

The main goal of our work is to design a mobile streaming media CDN (MSM-CDN) that enhances streaming media quality for mobile clients while utilizing network resources efficiently and supporting client mobility in an integrated and practical way. To achieve this, we propose a dynamic SMIL framework in which all of the CDN technologies including content segmentation, request routing, pre-fetch control, and session handoff, are controlled by SMIL (Synchronized Multimedia Integrated Language)[16] modification. The reasons for focusing on SMIL are itemized below.

- Since SMIL includes content location information, client requests can flexibly be redirected to the best cache server by replacing the original content location by the cache location selected based on the network and caching conditions.
- To improve caching efficiency, large streaming content is likely to be divided into several segments. In this case, SMIL can easily manage the temporal and spatial relation among these segments.
- SMIL makes user access patterns more predictable because it provides temporal information about when a client is likely to request each segment. This enables efficient pre-fetch control to cache servers.
- Since SMIL is based on XML (eXtensible Markup Language)[17], XML-related technologies can be applied to this system.
- 3GPP PSS adopts SMIL as the scene description language. Our dynamic SMIL framework follows the 3GPP-compliant specifications. This is important for deploying this CDN system over 3G mobile networks.
- Content providers can easily create SMIL files for describing streaming content because SMIL is a standard and popular scene description language.

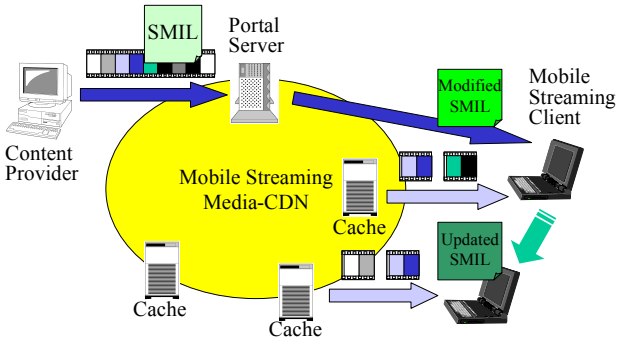

Fig. 1. Concept of dynamic SMIL framework.

Fig. 1 shows the concept of the dynamic SMIL framework. In this system, content providers register their streaming content with SMIL files describing the layout and media synchronization about the content, and do not need to worry about the existence of cache servers. Mobile streaming clients, on the other hand, download the modified SMIL files from the portal servers in the CDN. By simply following the downloaded SMIL files, the clients can receive multimedia content from the best cache servers. Even when mobile clients move to another location, the clients can continue multimedia streaming from the best cache servers by updating the SMIL files. The key elements of this system are as follows.

1. The CDN provides content segmentation functionality to enable segment caching and to improve cache efficiency. It accepts content registered from content providers and divides the content into several segments if it is too large. At the same time, it modifies the SMIL file corresponding the segmented content to describe the temporal relations hip of the segments.
2. On-demand URL rewriting described in [18] is applied to SMIL. That is, when a mobile client requests a SMIL file, the original content locations in the SMIL file are replaced by the cache server locations, and the modified SMIL file is sent to the client. The client, then, requests session setup to the cache servers according to the modified SMIL file.
3. To minimize the waste of the network resources and cache memory, streaming segments are scheduled to be pre-fetched just before a streaming client requests the segments while the client is receiving streaming segments. The request time is derived from the timing information described in SMIL files.
4. When a mobile client moves from one location to another location, the client requests an update of the SMIL file. The portal server, then, updates the SMIL file to indicate the best cache server locations and creates the SOAP (Simple Object Access Protocol)[19] message that includes the update information from the previous SMIL file. After the client reproduces the updated SMIL file from the SOAP message, it continues the streaming session according to the updated SMIL file.

We also introduce our proposed QoS control architecture[20][21] between cache servers and mobile clients. Because there is an error-prone radio link between a cache server and a mobile client, the cache server must consider the influence of radio link errors as

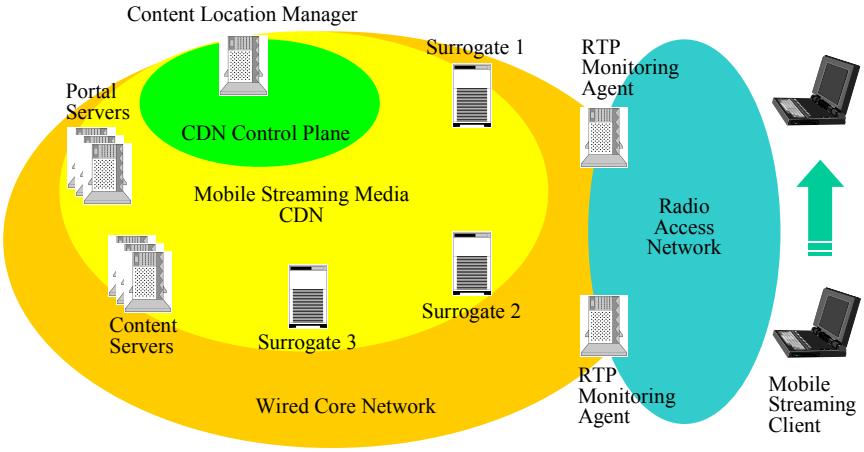

Fig. 2. Basic architecture of MSM-CDN.

well as that of network congestion. In this architecture, a network agent called "RTP monitoring agent" lies at the wired/wireless intersection, monitors RTP media streams, and feeds back information to the cache servers about the packet loss and jitter observed at the agent. From the feedback from both the RTP monitoring agent and mobile client, the cache servers can correctly determine the network and radio link conditions, and can adapt their media quality to both conditions.

The rest of this paper is organized as follows. Section 2 explains our architecture overview. Then, each key element to support our CDN architecture, i.e., content segmentation, request routing, pre-fetch scheduling, and session handoff, is described respectively in section 3 through section 6. Section 7 describes a QoS (Quality of Service) control mechanism utilizing the RTP monitoring agent. We report the current status of our prototyping work in section 8, and briefly discuss system design issues in section 9. Finally, section 10 concludes this paper.

2. BASIC ARCHITECTURE

The MSM-CDN architecture we propose is shown in Fig. 2. This architecture consists of the wired core network and the radio access network. The CDN overlays the core network, and the CDN control plane is in the middle of the CDN.

The CDN is composed of streaming portal servers, content servers, surrogates, and a content location manager (CLM). The streaming portal server accepts content registrations from content providers and provides content segmentation and SMIL modification functionality. The content servers are simple HTTP (Hypertext Transfer Protocol)[22] and RTSP (Real-Time Streaming Protocol)[23] servers and store a large number of streaming segments derived from the original media content. They do not have caching functionality, and work as HTTP servers for the surrogates and RTSP servers for the mobile clients. The surrogates are the RTSP servers that temporarily store streaming segments and serve streaming clients as streaming servers via RTSP session control. The CLM is a server located in the CDN control plane, and it takes charge of managing content locations and scheduling pre-fetches.

The mobile streaming clients can connect to the CDN via the radio access network. They are the RTSP clients that request multimedia content delivery according to the SMIL files sent from

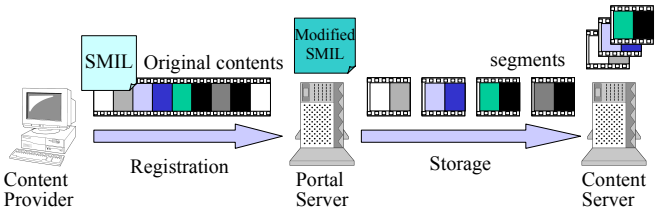

Fig. 3. Content segmentation and SMIL modification at streaming portal server.

the streaming portal server. The RTP monitoring agent is a network agent located at the intersection of the wired/wireless networks. We describe this function later.

From the next section, we describe each key element of our architecture in more detail.

3. CONTENT SEGMENTATION

Content segmentation is effective especially for large streaming content. By dividing content into several segments, cache servers can store and remove the content at the granularity of the segments, which leads to efficient cache memory and network resource utilization. Prefix caching[10], selective caching[12], variable-sized segmentation[13], and content-aware segmentation[14] have been proposed as the intelligent variants of segment caching.

We also introduce content segmentation in our CDN architecture to improve cache efficiency. The problem with content segmentation, however, is to complicate the management of these segments. In other words, content segmentation increases the number of the files network servers have to manage. And to realize such intelligent caching schemes proposed in [10]-[14], the servers have to know the relation among the segments. Thus, how to manage the segments is an important issue.

For this purpose, we bind a SMIL file to the segments divided from a streaming content. Since SMIL provides the timing information for playing or displaying, it can easily manage the temporal relation among the segments. By parsing the SMIL file, network servers can determine which segments are the next to the segments currently served and how long the segments will last.

```
<!DOCTYPE smil PUBLIC "-//W3C//DTD SMIL 2.0//EN"
          "http://www.w3.org/2001/SMIL20/SMIL20.dtd">
<smil xmlns="http://www.w3.org/2001/SMIL20/Language">
  <head>
    <layout>
      <region id="v" top="5" left="5" width="180" height="180"/>
    </layout>
  </head>
  <body>
    <par dur="60min">
      <audio src="rtsp://content-server/content-A.mp4" />
      <video src="rtsp://content-server/content-V.mp4" region="v" />
    </par>
  </body>
</smil>
```

Fig. 4. Example of original SMIL file.

```
<!DOCTYPE smil PUBLIC "-//W3C//DTD SMIL 2.0//EN"
          "http://www.w3.org/2001/SMIL20/SMIL20.dtd">
<smil xmlns="http://www.w3.org/2001/SMIL20/Language">
  <head>
    <layout>
      <region id="v" top="5" left="5" width="180" height="180"/>
    </layout>
  </head>
  <body>
    <seq>
    <par dur="10min">
      <audio src="rtsp://content-server/content-A-1.mp4" />
      <video src="rtsp://content-server/content-V-1.mp4" region="v" />
    </par>
    <par dur="20min">
      <audio src="rtsp://content-server/content-A-2.mp4" />
      <video src="rtsp://content-server/content-V-2.mp4" region="v" />
    </par>
    <par dur="30min">
      <audio src="rtsp://content-server/content-A-3.mp4" />
      <video src="rtsp://content-server/content-V-3.mp4" region="v" />
    </par>
    </seq>
  </body>
</smil>
```

Fig. 5. Modified SMIL file after content segmentation.

The procedure of content segmentation is shown in Fig. 3. First, content providers register their own streaming content with the streaming portal server. Here, the content providers also attach the SMIL files describing the layout and media synchronization about their streaming content. By registering content with SMIL files, content providers can specify how the content is displayed at streaming clients. If the size of the registered content is larger than a certain threshold, the portal server divides these contents into smaller segments. At the same time, the portal server modifies the SMIL file attached to the content to describe the temporal relations among these segments. Then, it sends the segmented content to a content server and requires it to store them.

Then, let us see an example of SMIL modification. Here, there is a streaming content whose length is sixty minutes, and its SMIL file is shown in Fig. 4. The SMIL shows that content names are "content-A.mp4" for audio content and "content-V.mp4" for video content and the duration of both content are sixty minutes.

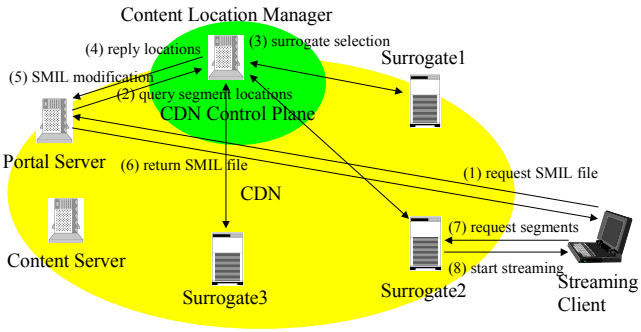

Fig. 6. Request routing procedure by SMIL modification

We assume the content is divided into three segments, and each segment is ten, twenty, and thirty minutes long, respectively. In this case, the SMIL file is modified at the streaming portal server as shown in Fig. 5. According to the modified SMIL file, these segments are renamed as "content-A-{1,2,3}.mp4" for audio segments and "content-V-{1,2,3}.mp4" for video segments, and the durations of the segments are ten, twenty, and thirty minutes long from the top segments. Since the part located between <seq> and </seq> indicates that these segments are sequentially played and displayed from the top segments, the temporal relation among them is evident.

In this way, SMIL files are modified along with content segmentation, and the modified SMIL files indicate the names, durations, and relations of the segments.

4. REQUEST ROUTING

Request routing is the most important technology for CDN. Several request routing methods have already been proposed[18]. One popular method is DNS-based request routing. However, its resolution is domain level and, therefore, fine-grain request routing at object level is difficult. Another method is using RTSP REDIRECT. Although it's simple, the REDIRECT message has to be issued from original content servers to streaming clients each time the clients request a streaming segment, and this leads to session setup latency.

In our CDN architecture, we apply on-demand URL rewriting[18] to SMIL. Request routing is done by modifying SMIL files at the streaming portal server when streaming clients request the SMIL files. In other words, the streaming portal server replaces the content locations in SMIL files by the surrogate locations, and returns the modified SMIL files to streaming clients. The advantages of this request routing are:

- Request routing is done before clients request a streaming session by RTSP. This enables faster pre-fetching described in the next section.
- There is no extra procedure such as DNS-based request routing or RTSP REDIRECT method to set up a streaming session between clients and surrogates. This leads to low latency of session setup.

Fig. 6 shows the request routing procedure by SMIL modification.

When a streaming client requests a SMIL file to start multimedia content streaming (1), the streaming portal server reads the SMIL file corresponding to the streaming content stored in its memory.

```
<!DOCTYPE smil PUBLIC "-//W3C//DTD SMIL 2.0//EN"
          "http://www.w3.org/2001/SMIL20/SMIL20.dtd">
<smil xmlns="http://www.w3.org/2001/SMIL20/Language">
  <head>
    <layout>
    <region id="v" top="5" left="5" width="180" height="180"/>
    </layout>
  </head>
  <body>
    <seq>
    <par dur="10min">
    <audio src="rtsp://surrogate2/content-server/content-A-1.mp4" />
    <video src="rtsp://surrogate2/content-server/content-V-1.mp4
      region="v" />
    </par>
    <par dur="20min">
    <audio src="rtsp://surrogate2/content-server/content-A-2.mp4" />
    <video src="rtsp://surrogate2/content-server/content-V-2.mp4"
      region="v" />
    </par>
    <par dur="30min">
    <audio src="rtsp://surrogate3/content-server/content-A-3.mp4" />
    <video src="rtsp://surrogate3/content-server/content-V-3.mp4"
      region="v" />
    </par>
    </seq>
  </body>
</smil>
```

Fig. 7. Modified SMIL file to indicate surrogate locations.

Then, the portal server queries CLM about the locations of the segments described in the SMIL file (2). At this time, the portal server notifies the estimated time when each segment will be requested, by parsing the timing information in the SMIL file. CLM utilizes this information for pre-fetching control described later. The client location (IP address) is also notified to CLM.

CLM selects the best surrogates for each segment based on the client location, network condition, caching status, surrogate load, and so on (3). In this paper, we do not specify how to select the surrogate locations. After the selection, it responds the locations of the selected surrogates to the portal server (4).

The streaming portal server, then, replaces the original content locations in the SMIL file by the surrogate locations (5). Fig. 7 shows an example of the SMIL file whose content locations are modified to indicate surrogate locations. The original URLs of the segments shown in Fig. 5 are replaced by the surrogate URLs. The locations of the segments could be different. In the SMIL file shown in Fig. 7, surrogate2 is selected for the top four segments, and surrogate3 is selected for the last two segments.

After the modification of the SMIL file, the portal server returns the modified SMIL file to the streaming client (6). The streaming client receives the SMIL file and requests streaming sessions by RTSP according to the SMIL file (7). The surrogate requested to start the streaming session delivers RTP (Real-time Transport Protocol)[24] media packets that include audio and video payloads to the streaming client (8).

As long as the streaming client follows the SMIL file downloaded from the portal server, it always requests session setup to the surrogates selected by CLM.

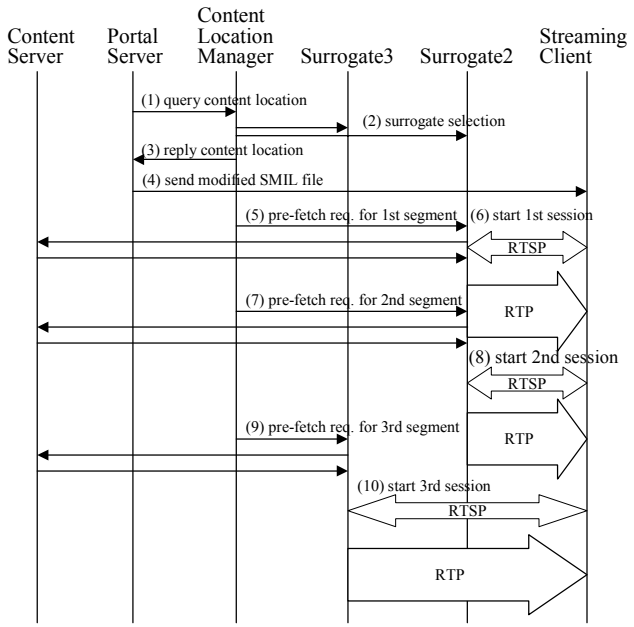

Fig. 8. Sequence of pre-fetching scheduled by content location manager.

5. PRE-FETCH SCHEDULING

To smoothly start streaming sessions, the surrogates have to pre-fetch the segments before the clients request the segments. As described in section 3, the modified SMIL files provide the temporal relations among the segments. By utilizing this information, the time when the clients will request each segment can be estimated. Based on the estimated times, pre-fetching is scheduled to be finished before the clients request the segments.

Fig. 8 shows the sequence of segment pre-fetching. In our CDN architecture, CLM takes charge of pre-fetch scheduling.

When the streaming portal server queries CLM about segment locations, it also notifies the timing information for the segments derived from the SMIL file (1). After the surrogate selection is done based on the client location, network condition, caching status, and surrogate load (2), it replies the locations of the selected surrogates (3).

At the same time, CLM creates pre-fetch scheduling table as shown in Fig. 9. The table is composed of session ID, client IP address, start time, pre-fetch status, and pre-fetch information for each segment. Pre-fetch information includes estimated request time, segment duration, segment name, original location, and surrogate selected to serve the segment.

Based on this table, CLM starts pre-fetching. As shown in Fig. 8, the first audio and video segments are pre-fetched (5) before the client requests the segments (6). Before the client finish to receive the whole first segments, the second segments are pre-fetched (7). The third segments are pre-fetched to surrogate3 while the client is receiving the second segments from surrogate2 (9).

If the client pauses the streaming, the surrogate serving the client notifies CLM of the pause. CLM, then, changes the pre-fetch status in the pre-fetch scheduling table to PAUSE and stops pre-fetch requests to surrogates. When the client restarts the streaming,

Session ID			123456789		
Client IP address			10.20.30.40		
Start time			12:34		
Pre-fetch status			ACTIVE		
#	Time	Dur.	Seg. Name	Orig. Loc.	Surrogate
1	0 m	10 m	content-A-1.mp4	content-server	surrogate2
2	0 m	10 m	content-V.1.mp4	content-server	surrogate2
3	10 m	20 m	content-A-2.mp4	content-server	surrogate2
4	10 m	20 m	content-V-2.mp4	content-server	surrogate2
5	30 m	30 m	content-A-3.mp4	content-server	surrogate3
6	30 m	30 m	content-V-3.mp4	content-server	surrogate3

Fig. 9. Pre-fetch scheduling table.

the surrogate notifies CLM of the restart. CLM recalculates the start time and the request times for each segment, and changes the pre-fetch status to ACTIVE. The same procedure is taken when the client skips or goes back to the segments.

6. STREAMING SESSION HANDOFF

When a mobile client moves from one location to another location, the best surrogate serving it could be changed. One way to redirect the client's request to the new surrogate is issuing RTSP REDIRECT method from the surrogate currently serving the client. In this case, however, the client will request the next segments from the obsolete surrogate if it is still guided by the SMIL file downloaded when starting the multimedia content delivery. Another way is client-driven session handoff. That is, the client disconnects the current surrogate and requests the new surrogate for a session setup. The problem with this is how the client gets to know the best surrogate locations for itself.

To enable session handoff for all of the streaming segments, the mechanism to update SMIL files is necessary when the best surrogates are changed due to client mobility. In our CDN architecture, a SOAP message is used for this purpose. When a mobile client moves to another location and the best surrogates are changed, the portal server sends the SOAP message that includes the locations of the new surrogates serving the client. The client, then, updates its SMIL file and continues to receive the streaming segments from the new surrogates by following the updated SMIL file.

The advantages of sending SOAP messages to update SMIL files are:

- The SOAP messages can explicitly notify the update of the current SMIL file.
- If the SMIL file to be updated is elaborate for describing the layout and its size is very large, sending the updated SMIL file itself consumes network resources. In that case, the SOAP message that tells only the update information of the surrogate locations is more cost-effective.
- Both SMIL and SOAP are based on XML. The same modules to parse XML documents can be used at mobile clients. In addition, most of the mobile clients are expected to be SOAP-capable in the near future. This requires only small modification on the clients.

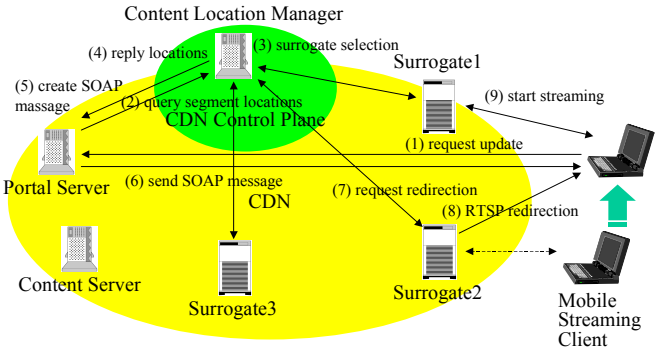

Fig. 10. SMIL update procedure when client moves.

```
<SOAP-ENV:Envelope
  xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
  SOAP-ENV:encodingStyle=
    "http://schemas.xmlsoap.org/soap/encoding/">
  <SOAP-ENV:Body>
    <m:update  xmlns:m="http://docomo.ne.jp/dynamicSMIL">
      <oldSrc>/smil/body/seq/par[2]/audio/@src</oldSrc>
      <newSrc>rtsp://surrogate1/content-server/content-A-2.mp4</newSrc>
      <oldSrc>/smil/body/seq/par[2]/video/@src</oldSrc>
      <newSrc>rtsp://surrogate1/content-server/content-V-2.mp4</newSrc>
      <oldSrc>/smil/body/seq/par[3]/audio/@src</oldSrc>
      <newSrc>rtsp://surrogate1/content-server/content-A-3.mp4</newSrc>
      <oldSrc>/smil/body/seq/par[3]/video/@src</oldSrc>
      <newSrc>rtsp://surrogate1/content-server/content-V-3.mp4</newSrc>
    </m:update>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Fig. 11. Example of SOAP message to update SMIL file.

Fig. 10 shows the procedure of SMIL update when a mobile client moves. In this figure, we assume the best surrogates are changed from surrogate2 to surrogate1.

When a mobile client moves from one location to another location and detects its movement, the mobile client requests the portal server to update the SMIL file by the SOAP request message (1). After receiving the update request, the portal server requests CLM to recalculate the locations of the best surrogates (2). As similar to the case described in section 4, CLM selects the best surrogates for each segment based on the client location, network condition, caching status, and surrogate load (3), and responds the locations of the selected surrogates to the portal server (4).

The portal server, then, creates the SOAP response message to inform the client of the new surrogate locations for each segment (5). An example of the SOAP response is shown in Fig. 11. The parts located between <oldSrc> and </oldSrc> indicate the locations to be replaced in XPath (XML Path Language)[25] description. The next parts located between <newSrc> and </newSrc> indicate the URLs of the new surrogates selected by CLM. In Fig. 11, the surrogates serving the second and the third audio/video segments are changed to surrogate1.

The SOAP message created in this way is transmitted to the client (6). From this SOAP message and the current SMIL file, the client creates the new SMIL file that includes the new surrogate locations. The client continues to request the subsequent segments from the surrogates described in the updated SMIL file.

While CLM responds the segment locations to the portal server (4), CLM also recalculates pre-fetch schedule and rewrites pre-fetch scheduling table. In addition, it may request the surrogate currently serving the client (surrogate2) to redirect the client's request to the newly selected surrogate (surrogate1) (7). Then, surrogate2 issues RTSP REDIRECT method to the client (8), and the client requests surrogate1 to send the segments currently being transmitted from surrogate2 (9). Note that surrogate2 does not need to issue RTSP REDIRECT methods for the subsequent segments because the client requests surrogate1 to deliver them directly according to the updated SMIL file.

7. QOS CONTROL AT SURROGATE

Even when surrogates close to mobile clients deliver multimedia content, network congestion still occurs between surrogates and clients. Especially in mobile networks, the bandwidth a client can use is fluctuated because of the client's mobility. In addition to network congestion, there are packet losses due to radio link errors. Therefore, some QoS control mechanisms are necessary for the surrogates.

At the moment, a number of papers have considered how to control the transmission rate of non-TCP flows. TCP-friendly rate control (TFRC)[26] and other equation-based congestion controls[27][28] have been proposed. Some rate control mechanisms utilize the RTP Control Protocol (RTCP)[24] to obtain feedback information from receivers[29][30]. RTCP is used along with RTP to provide quality information on the corresponding RTP media stream as well as some user-specific information. As shown in Fig. 12, the receiver of an RTP media stream sends back RTCP receiver reports, which include packet loss and jitter information, so that the RTP sender can identify network congestion condition and control its transmission rate accordingly.

These rate control mechanisms assume that packet loss, delay, and jitter are caused by network congestion. In mobile networks, however, packet loss and jitter may also be caused by radio link errors. When conventional rate control mechanisms are applied to the surrogates, they cannot identify the network congestion condition correctly, and this leads to inappropriate rate control. A typical symptom is that a surrogate reduces its transmission rate even if the network is not congested.

To enable the surrogates to realize appropriate QoS control, we introduce a new type of proxy named "RTP monitoring agent" and a QoS control mechanism utilizing the RTP monitoring agent. This architecture has already been proposed in [20][21].

Fig. 13 shows the network architecture with the RTP monitoring agent. The RTP monitoring agent returns feedback reports corresponding to RTCP receiver reports, but clients also return RTCP receiver reports to the surrogates. Note that the shaping point, which adjusts the outgoing rate of all packet traffic to the rate of the radio link, is located just prior to the RTP monitoring agent. This limits the occurrence of network congestion on the path between the surrogates and the RTP monitoring agent, and eliminates it on the radio link. The reports from the RTP monitoring agent indicate network congestion condition, while those from the client cover network congestion and radio link conditions. Therefore, the surrogates can determine radio link condition by comparing these two reports.

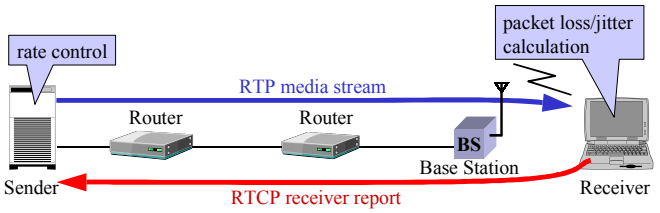

Fig. 12. Rate control using RTCP receiver reports.

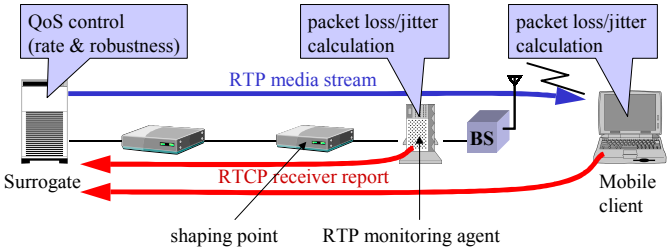

Fig. 13. QoS control architecture with RTP monitoring agent.

The RTP monitoring agent is a network agent located at the boundary between the wired core network and the wireless link. During streaming sessions, it collects statistics about packet loss, jitter, etc. for each RTP media flow and sends feedback reports to the corresponding surrogates. To do this, the RTP monitoring agent has to identify each flow by analyzing IP addresses and UDP port numbers. Although someone may point out computational burden, supposed heavy, at the RTP monitoring agents, one agent will not carry so many flows because it is located just before radio links. The supposed overhead is therefore moderately low and acceptable when considering the expected benefits. Since the feedback reports sent from the agent indicate only the wired core network condition, the surrogates can determine the correct congestion status and can do appropriate rate control. The RTP monitoring agent does not interrupt any media streams, which means streaming sessions can be continued even in case of agent's malfunction.

The surrogates have to separate the feedback reports sent from the RTP monitoring agent from those sent from the mobile clients. The pair of SSRC identifier and CNAME (canonical name) included in RTCP receiver reports can be used for this purpose. Any of the conventional rate control methods can be applied to the reports from the RTP monitoring agent. The surrogates can control their transmission rates upon analyzing the reports from the RTP monitoring agent because these include only network congestion information.

On the other hand, rate control does not work if the packet loss is caused by radio link errors. The surrogates, therefore, need to estimate radio link condition from the two reports. In the case of radio link errors, the surrogates have to make their media streams more robust against packet losses. Robustness can be controlled by several ways. One example is changing the frequency of transmitting forward error correction (FEC) packets[31]. Increasing the frequency of FEC packet transmission increases robustness against packet losses.

In this QoS control architecture with the RTP monitoring agent, the surrogates can determine network congestion and radio link conditions correctly, and adapt their media quality to both

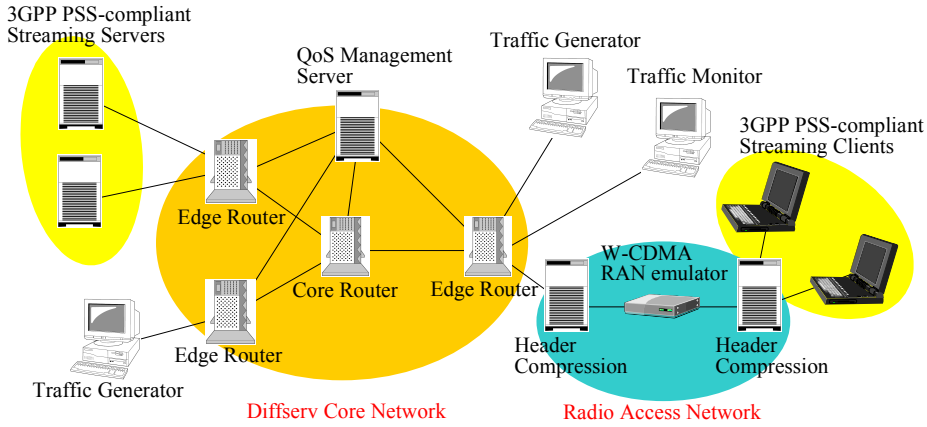

Fig. 14. Architecture of mobile QoS testbed (MOBIQ).

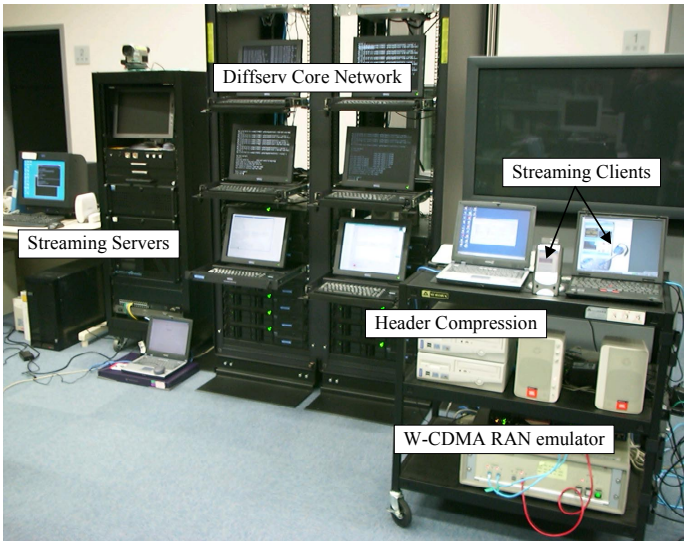

Fig. 15. Picture of mobile QoS testbed (MOBIQ).

Fig. 16. Picture of streaming client.

conditions. More details and its application to MPEG-4 video streaming are described in [20].

8. SYSTEM PROTOTYPE

We have built a mobile QoS testbed called "MOBIQ[32]" shown in Fig. 14 and Fig. 15. MOBIQ consists of Diffserv[33]-based core network and 3G RAN (radio access network).

The core network is composed of FreeBSD routers with ALTQ[34] module. ALTQ provides packet classifier, packet-scheduling disciplines including Priority Queueing, WFQ (Weighted Fair Queueing)[35], CBQ (Class-Based Queueing)[36], and queue management schemes including RED (Random Early Discard)[37] and RIO (RED with In and Out)[38]. These functions are necessary for Diffserv core routers. It also provides TOS (Type of Service) field marking and traffic meter functions, which are necessary for Diffserv edge routers. Mobile IPv6 and multicast routing are also running on these routers.

Over the Diffserv-based core network, we have developed a QoS management server, which manages QoS policies and configures all of the ALTQ routers according to the QoS policies. Its QoS policy framework is compliant with IETF Policy Framework[39]. The QoS management server has SOAP interfaces for QoS policy setup, traffic monitoring, and event notifications.

The 3G RAN includes W-CDMA (Wideband Code Division Multiple Access) RAN emulator[40] and two header compression nodes. The W-CDMA RAN emulator provides Acknowledged mode, which provides retransmission and segmentation functions, and Unacknowledged mode, which provides only segmentation function, defined in 3GPP RLC (Radio Link Control) specifications[41]. The header compression nodes implement three RTP/UDP/IP header compression algorithms defined in RFC2508 (CRTP)[42], RFC 3095 (ROHC: Robust Header Compression)[43], and our proposed method, MRC (Multiple-Reference Compression)[44]. The packets whose RTP/UDP/IP headers are compressed are transmitted over the W-CDMA RAN emulator.

In addition to the core network and the 3G RAN, we have also developed 3GPP PSS-compliant streaming server and client (Fig. 16). The protocol stack is shown in Fig. 17. The streaming server is RTSP/SDP (Session Description Protocol)[45]-capable as the session control protocols, and supports MP4 file format[46] as its input file format. The streaming client is also RTSP/SDP-capable,

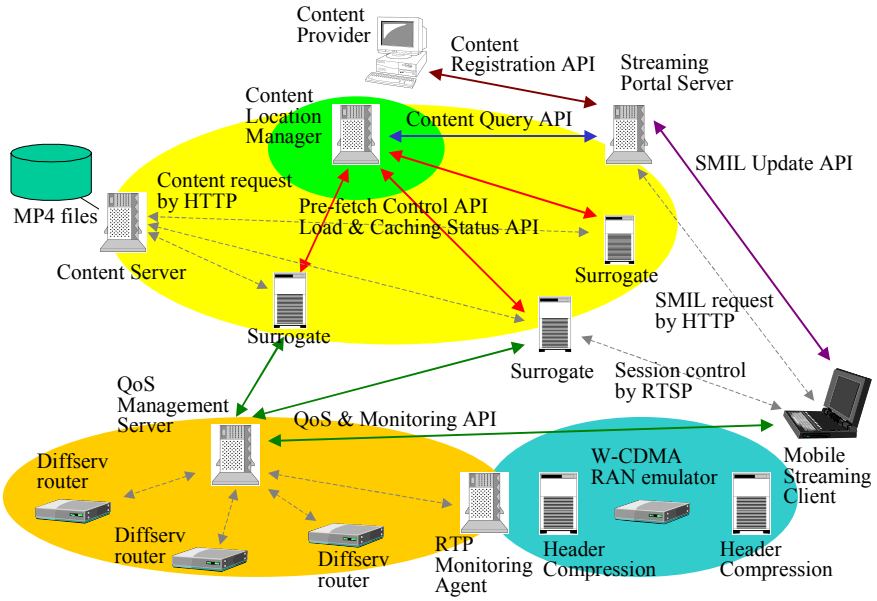

Fig. 18. API design for MSM-CDN.

Video coding		Speech coding	Audio coding	Session control	Scene description
H.263 Base-line	MPEG-4 video	AMR	MPEG-4 AAC	SDP	SMIL
				RTSP	HTTP
RTP/UDP (from MP4 file format)				TCP	
IP					

Fig. 17. Protocol stack of 3GPP PSS-compliant streaming software.

and supports AMR (Adaptive Multi-Rate), H.263 baseline, MPEG-4 video simple profile, and MPEG-4 AAC (Advance Audio Coding). It can parse SMIL files and issue RTSP requests according to the SMIL files. The RTP monitoring agent and the streaming server with QoS control mechanism utilizing the agent have also been implemented on this PSS-compliant streaming software.

Over this testbed, we are currently prototyping the streaming portal server, the content location manager (CLM), and the streaming surrogates for constructing the MSM-CDN. All of the APIs (Application Programmable Interfaces) provided by these nodes are designed as SOAP interfaces. Fig. 18 shows the API design for the MSM-CDN.

The portal server provides the content registration API for content providers and the SMIL update API for the streaming clients. CLM provides the content location query API for the portal server. The streaming surrogates have the pre-fetch API for CLM to request the surrogates to download content, and also have the load status and caching status API for CLM to select the best surrogate based on the surrogate load and caching status. The QoS setup and traffic monitoring APIs provided by the QoS management server are used by the surrogates or the streaming clients when they start streaming sessions.

9. SYSTEM DESIGN ISSUES

This paper describes a dynamic SMIL framework for a mobile streaming media CDN on an underlying base QoS network. This system performs a number of underlying algorithms, including media caching, pre-fetching, request routing, resource management, handoffs, and streaming. Each algorithm must be designed to effectively handle the combination of streaming and mobility; a number of these algorithms are currently under investigation by the authors. While a detailed discussion of these algorithms is beyond the scope of this paper, we briefly describe some examples to convey the nature of the issues at hand.

The MSM-CDN overlay performs efficient caching, pre-fetching, handoff, and resource management algorithms; these algorithms require choosing a media segment size for the content segmentation operation. An appropriately chosen segment size will consider client mobility, since each segment boundary provides an opportunity for a handoff, as well as content use statistics, since knowledge that users only tend to view the first 30 seconds of a media clip can help the system efficiently allocate its storage resources. Resource management and caching algorithms can also benefit from knowledge of content popularity, since content that is likely to be accessed by other users should be left in cache storage whenever possible.

The MSM-CDN overlay also interacts with the underlying base QoS network, and this interaction can be used for efficient algorithm design. Notice that the pre-fetch scheduling table gives a deadline for when the segment must be pre-fetched to the surrogate, however it does not specify a precise time for the pre-fetch to occur. This flexibility allows the CLM and surrogates to schedule the actual pre-fetch time while taking the underlying QoS base network resource usage into account. For example, the CLM can use the QoS monitoring API to spot potential bottlenecks in the network, and it can then use the QoS management API to compensate for these bottlenecks by decreasing or increasing the priority of different pre-fetched streams to optimize the overall system performance.

10. CONCLUSIONS

We presented a mobile streaming media CDN architecture in which all of the technologies related to CDN are enabled by SMIL modification. In this architecture, mobile clients simply follow the SMIL file downloaded from the streaming portal server, and this leads to multimedia content delivery from the best surrogates in the CDN. The key components of this architecture are as follows.

1. To improve cache efficiency, content segmentation is provided along with SMIL file modification to describe the temporal relations among the segments.
2. To redirect client requests to the best surrogates, the content locations in the SMIL file are modified to indicate the appropriate surrogate locations.
3. Segment pre-fetching is scheduled just before clients request segments in order to minimize the waste of the network resources and cache memory. The request times are derived from SMIL files.
4. SOAP messages to update SMIL file are defined. This is necessary when mobile clients move and the best surrogates for them are changed.
5. The surrogates utilizing the RTP monitoring agent can determine network congestion and radio link conditions correctly and, therefore, can control their media quality accordingly.

We reported the current status of our prototype. We have already built a mobile QoS testbed called "MOBIQ", which includes Diffserv-based core network with SOAP interfaces, and 3G RAN with W-CDMA RAN emulation and RTP/UDP/IP header compression. A 3GPP PSS-compliant streaming server and client were also developed. We are currently designing the SOAP APIs needed for the MSM-CDN and building the CDN over the current testbed.

We are now interested in personalized multimedia streaming. Since users have limited devices and time especially in mobile environments, it is important to deliver multimedia data dynamically and automatically summarized depending on user preferences or network conditions. Towards this, personalization of SMIL files[47] and MPEG-7 based personalization[48] have already been proposed. We are considering the integration of these personalization technologies into our MSM-CDN architecture.

11. REFERENCES

[1] Akamai Technologies, Inc., http://www.akamai.com/.

[2] Digital Island, http://www.digisle.com/.

[3] NTT DoCoMo, Inc., i-mode, http://www.nttdocomo.co.jp/english/i/index.html.

[4] WAP Forum, Wireless Access Protocol, http://www.wapforum.org.

[5] NTT DoCoMo, Inc., FOMA (Freedom Of Mobile multimedia Access), http://foma.nttdocomo.co.jp/english/.

[6] 3GPP TS 26.234 v 4.1.0, Transparent end-to-end packet switched streaming service (PSS); Protocol and codecs, September 2001.

[7] R. Rejaie and J. Kangasharju, "Mocha: A Quality Adaptive Multimedia Proxy Cache for Internet Streaming," In Proceedings of NOSSDAV 2001, 2001.

[8] R. Rejaie, H. Yu, M. Handley, and D. Estrin, "Multimedia Proxy Caching Mechanism for Quality Adaptive Streaming Applications in the Internet," In Proceedings of INFOCOM 2000, 2000.

[9] J. Kangasharju, F. Hartanto, M. Reisslein, and K. W. Ross, "Distributing Layered Encoded Video through Caches," In Proceedings of INFOCOM 2001, 2001.

[10] S. Sen, J. Rexford, and D. Towsley, "Proxy prefix caching for multimedia streams," In Proceedings of INFOCOM'99, 1999.

[11] E. Bommaiah, K. Guo, M. Hofmann, and S. Paul, "Design and Implementation of a Caching System for Streaming Media over the Internet," In Proceedings of Real-time Technology and Applications Symposium (RTAS 2000), 2000.

[12] Z. Miao and A. Ortega, "Proxy Caching for Efficient Video Services over the Internet," In Proceedings of Packet Video Workshop 1999, 1999.

[13] K-L. Wu, P. S. Yu, and J. L. Wolf, "Segment-Based Proxy Caching of Multimedia Streams," In Proceedings of the 10th International World Wide Web Conference, May 2000.

[14] S-J. Lee, W-Y. Ma, and B. Shen, "An Interactive Video Delivery and Caching System Using Video Summarization," In Proceedings of WCW 2001, 2001.

[15] C. D. Cranor, M. Green, C. Kalmanek, D. Shur, S. Sibal, J. E. Van der Merwe, and C. J. Sreenan, "Enhanced Streaming Services in a Content Distribution Network," IEEE Internet Computing, Vol. 5, No. 4, July-August 2001.

[16] Synchronized Multimedia Integration Language (SMIL 2.0), W3C Recommendation, August 2001. http://www.w3.org/TR/smil20/.

[17] Extensible Markup Language (XML) 1.0 (Second Edition), W3C Recommendation, October 2000. http://www.w3.org/TR/REC-xml.

[18] A. Barbir, B. Cain, F. Douglis, M. Green, M. Hofmann, R. Nair, D. Potter, and O. Spatscheck, "Known CDN Request-Routing Mechanisms," IETF Internet-Draft, draft-cain-cdnp-known-request-routing-03.txt (work in progress), November 2001.

[19] Simple Object Access Protocol (SOAP) 1.1, W3C Note, May 2000. http://www.w3.org/TR/SOAP/.

[20] T. Yoshimura, T. Ohya, T. Kawahara, and M. Etoh, "Rate and Robustness Control with RTP Monitoring Agent for Mobile Multimedia Streaming," In Proceedings of IEEE International Conference on Communications (ICC) 2002, April 2002.

[21] T. Yoshimura, T. Kawahara, T. Ohya, and M. Etoh, "QoS Control Architecture with RTP Monitoring Agent for Mobile Multimedia Streaming," IPSJ Symposium on Multimedia, Distributed, Cooperative and Mobile Systems (DICOMO 2001), June 2001.

[22] R. Fielding, J. Gettys, J. Mogul, H. Frystyk, L. Masinter, P. Leach, and T. Berners-Lee, "Hypertext Transfer Protocol -- HTTP/1.1," IETF RFC 2616, June 1999.

[23] H. Schulzrinne, A. Rao, and R. Lanphier, "Real Time Streaming Protocol," IETF RFC 2326, April 1998.

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

[25] XML Path Language (XPath) Version 1.0, W3C Recommendation, November 1999. http://www.w3.org/TR/xpath

[26] M. Handley, J. Padhye, S. Floyd, and J. Widmer, "TCP Friendly Rate Control (TFRC): Protocol Specification," Internet-Draft draft-ietf-tsvwg-tfrc-01.txt (work in progress), Internet Engineering Task Force, March 2001.

[27] S. Floyd, M. Handley, J. Padhye, and J. Widmer, "Equation-based congestion control for unicast application," In Proceedings of SIGCOMM '00, pp. 57-66, 2000.

[28] J. Padhye, J. Kurose, D. Towsley, and R. Koodli, "A model based TCP-friendly rate control protocol," In Proceedings of NOSSDAV '99, 1999.

[29] D. Sisalem and H. Schulzrinne, "The loss-delay based adjustment algorithm: A TCP-friendly adaptation scheme," In Proceedings of NOSSDAV '98, pp. 215-226, July 1998.

[30] R. Rejaie, M. Handley, and D. Estrin, "RAP: An end-to-end rate-based congestion control mechanism for realtime streams in the Internet," In Proceedings of INFOCOM '99, 1999.

[31] J. Rosenberg and H. Schulzrinne, "An RTP Payload Format for Generic Forward Error Correction," IETF RFC2733, December 1999.

[32] T. Yoshimura, T. Ohya, H. Matsuoka, and M. Etoh, "Design and Implementation of Mobile QoS Testbed MOBIQ for Multimedia Delivery Services," In Proceedings of Packet Video Workshop 2002, April 2002.

[33] S. Blake, D. Black, M. Carlson, E. Davies, Z. Wang, and W. Weiss, "An Architecture for Differentiated Services," IETF RFC 2475, December 1998.

[34] K. Cho, "Managing Traffic with ALTQ," In Proceedings of USENIX 1999, June 1999.

[35] Demers, S. Keshav, and S. Shenker, "Analysis and simulation of a fair queueing algorithm," In Proceedings of SIGCOMM '89, September 1989.

[36] S. Floyd and V. Jacobson, "Link-sharing and resource management models for packet networks," IEEE/ACM Transactions on Networking, Vol. 3, No. 4, August 1995.

[37] S. Floyd and V. Jacobson, "Random early detection gateways for congestion avoidance," IEEE/ACM Transaction on Networking, Vol. 1, No. 4, pp. 397-413, August 1993.

[38] D. D. Clark and W. Fang, "Explicit Allocation of Best Effort Packet Delivery Service," ACM Transactions on Networking, August 1998.

[39] IETF Policy Framework working group, http://www.ietf.org/html.charters/policy-charter.html.

[40] H. Inamura, T. Ishikawa, and O. Takahashi, "Evaluation of TCP traffic over W-CDMA network," IPSJ Technical Report, MBL18-33, September 2001. (in Japanese)

[41] 3GPP TS 25.322 v 3.5.0, RLC Protocol Specification, December 2000.

[42] S. Casner and V. Jacobson, "Compressing IP/UDP/RTP headers for low-speed serial links," IETF RFC 2508, February 1999.

[43] C. Bormann, C. Burmeister, M. Degermark, H. Fukushima, H. Hannu, L-E. Jonsson, R. Hakeberg, T. Koren, K. Le, Z. Liu, A. Martensson, A. Miyazaki, K. Svanbro, T. Wiebke, T. Yoshimura, and H. Zheng, "Robust header compression (ROHC): Framework and four profiles: RTP, UDP, ESP, and uncompressed," IETF RFC3095, July 2001.

[44] T. Yoshimura, T. Kawahara, T. Suzuki, and M. Etoh, "Multi-Reference Compression of IP/UDP/RTP Headers for Wireless Communications," In Proceedings of Wireless Personal Multimedia Communications (WPMC) 2000, pp. 397-402, November 2000.

[45] M. Handley and V. Jacobson, "SDP: Session Description Protocol," IETF RFC 2327, April 1998.

[46] International Standard ISO/IEC 14496-1: "Information technology - Coding of audio-visual objects -- Part 1: Systems", 2000.

[47] R. Hjelsvold, S. Vdaygiri, and Y. Leaute, "Web-based Personalization and Management of Interactive Video," the 10th International World Wide Web Conference, May 2001.

[48] S. Sekiguchi, M. Etoh, K. Emura, W. Fujikawa, K. Masumitsu, and T. Echigo, "Video Digest Delivery using MPEG-7 Media Structure Description and its Authoring System", IEICE Technical Report, PRMU2001-92, September 2001.