# Streaming Agent: A Network Proxy for Media Streaming in 3G Wireless Networks

Gene Cheung
Hewlett-Packard Laboratories, Japan
Hewlett-Packard Japan, Ltd

Takeshi Yoshimura
Multimedia Laboratories
NTT DoCoMo, Inc.

**Abstract**

Streaming media in 3G wireless networks means the streaming server must simultaneously adapt the media content to two mediums of drastically different characteristics: an often congested wired network and an unreliable wireless link. To improve the server's media adaptiveness, we enhance a previously proposed network proxy, located at the junction of the wired network and wireless link, to provide timely feedbacks to the server. The enhanced proxy is called Streaming Agent (SA). SA timely feedbacks provide partial path information for the server to track the wired client state — which packets have arrived correctly and on-time at SA. Armed with knowledge of wired client state, the server can in real-time better adapt the streaming content accordingly. We demonstrate SA's usefulness in the context of the format-adaptation problem: two video streams with the same bit rate but different I-frame frequencies are switched back and forth depending on SA and/or client timely feedbacks. We show that using a scheme that adapts to both SA and client feedbacks, visual quality can be improved up to 0.78dB at 230kbits/s over a scheme that adapts to client feedbacks only.

## I. Introduction

The goal of this paper is to improve streaming media quality in next generation 3G wireless networks [1]. Streaming media in our context means a piece of media content is delivered from a streaming server in a wired network to a mobile client via a last-hop wireless link. The content is decoded and viewed by the client before the entire content has been downloaded. The non-interactivity of the streaming session means that a tolerable initial playback delay at the client — in addition to the transmission delay — can be used to absorb some delay jitter in the network. However, media packets nevertheless have playback deadlines, and excessively late packets due to large delay jitter will be rendered useless at the client.

Unique to wireless media streaming is the presence of two drastically different transmission mediums: an often congested wired network and an unreliable wireless link. Despite this difference from wired network streaming such as the Internet, one straightforward approach to adapt wireless streaming media is to simply ignore the medium distinction and adapt the media content based solely on end-to-end observed conditions, whatever the cause maybe. This approach to wireless streaming media has been shown to be sub-optimal in [2], where the key observation is that end nodes cannot distinguish wireless link loss from wired network congestion, and so conventional network congestion control algorithms [3] [4], operating based on end-to-end packet-loss rate and round trip time (RTT), cannot be performed properly. The undesirable result is often a reduction in sending rate even though the wired network is not congested and only the wireless link is lossy. The proposed solution in [2] is the RTP monitoring agent — a network proxy located at the intersection of wired core network and wireless link, that sends *statistical feedbacks* (RTCP reports [5] in particular) back to the sender to help the sender estimates the wired *network state*. Statistical feedbacks such as conventional RTCP reports are observable network characteristics over a window of packets, such as mean packet loss rate and mean and variance of RTT. The sender, using RTCP reports from the monitoring agent and the client, can now distinguish packet loss in the wired network apart from packet loss in the wireless link and can perform the correct wired network congestion control.

RTCP reports from RTP monitoring agent is an example of *partial path information* — information collected by network proxies along the packet delivery path from the sender to the client. In general, providing

partial path information to the sender can only be beneficial; the more feedback information the sender receives, the better the sender can adapt the media content. From an academic point of view, how to use partial path information to optimize end-to-end performance is an interesting problem. The important engineering tradeoff, however, is how expensive the deployment cost of collecting and sending partial path information compare to the increase in performance. In the case of RTP monitoring agent, the tradeoff is convincingly worthwhile as it is impossible for the sender to perform correct congestion control without some form of partial path information.

Given the necessity of a RTP monitoring agent, in this paper we propose to enhance the described RTP monitoring agent to send a new type of partial path information. The enhancement is minimal in terms of implementation cost, and the enhanced network proxy is called a *streaming agent* (SA). The new partial path information is *timely feedbacks*, which are positive/negative acknowledgment packets (ACKs/NAKs) that tell the sender which packets has/has not arrived at SA correctly and on time. This is termed the wired *client state*, juxtaposing with wired network state tracked by SA's statistical feedbacks. Armed with the knowledge of wired client state, the sender can better adapt the media content. For example, to notify the sender that an important media packet has been dropped in the wired network, SA can send an NAK to the sender quickly informing him of the loss event using only the wired network infrastructure.

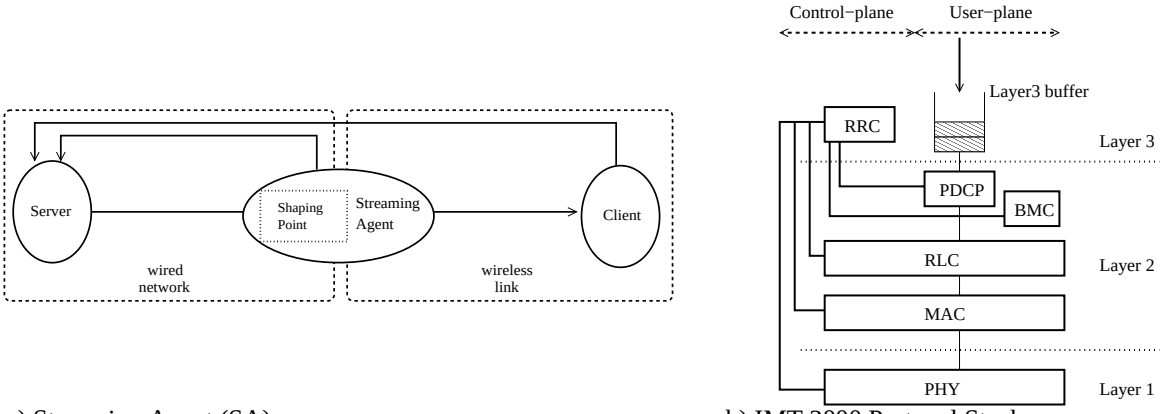
Although SA can conceivably be advantageous in many transmission schemes, we argue the usefulness of SA in the context of a simple problem — the format-adaptation problem: two video streams coded in the same bit rate but different I-frame frequencies are switched back and forth depending on SA and/or client feedbacks. While the format-adaptation problem is a practical solution as it does not require heavy computation such as transcoding [6], it is not the purpose of this paper to claim its performance. Rather, we simply use it as an illustration to show that many transmission schemes, such as format-adaptation, can benefit substantially from using SA's partial path information. To make our argument for SA in the context of the format-adaptation problem, we first show that timely feedbacks are important in general by showing that a format-adaptation scheme using both timely feedbacks and statistical feedbacks outperforms a scheme using statistical feedbacks only. We then show that a scheme using both SA and client timely feedbacks can outperform a scheme using only client timely feedbacks by a noticeable amount.

The organization of the paper is as follows. We first discuss related work in section II. We then discuss the details of SA in section III. We present an analytical model that explores the conditions under which SA is most useful in section IV. We then briefly discuss the context problem — format-adaptation problem, in section V. We present results in section VI. We conclude with a discussion of implementation issues in section VII and future work in section VIII.

## II. Related Work

We can trace the root of RTP monitoring agent [2] (later version in [7]) to Snoop agents in [8]. In [8], the key observation is that unreliable wireless links do not fit the TCP assumption that packet loss is caused only by network congestion. To avoid confusing link failure with congestion and resulting in unnecessary congestion control, transport-layer aware link-layer protocol called the Snoop Protocol is employed by Snoop agents located at intersection of wired network and wireless links. Snoop agents maintain a cache of TCP packets and retransmit lost packets when detected, while suppressing duplicate ACKs to the sender. While Snoop agents were designed specifically for optimizing TCP performance over last hop wireless links, RTP monitoring agent is the extension of the idea to media streaming. Unlike Snoop, which isolates the application layer from the network particulars — TCP's window-based congestion control and retransmission — as much as possible, RTCP reports from the monitoring agent are sent to the sender's application so that the application can perform proper congestion control while avoiding receiver buffer starvation by reducing source coding rate. The application can use a host of signal processing techniques to accomplish this, such as data-scheduling of scalable coding [9], transcoding [6] etc.

Streaming media is a popular topic, and an extensive body of previous research [10] [11] [9] [12] exists




a) Streaming Agent (SA) b) IMT-2000 Protocol Stack

Fig. 1. Streaming Agent and 3G link-layer Protocol Stack

focusing on the case when the entire delivery path is abstracted as one independent memoryless channel. As we have pointed out earlier, for the special case of wireless streaming media, one cannot distinguish wired network congestion from wireless link loss by looking only at end-to-end observed states, which will result in loss in performance when bit rate is reduced due to improper congestion control. However, many of these optimizations can conceivably be modified or extended to benefit from SA's partial path information, resulting in better end-to-end performance.

The general notion of timely feedbacks has been known to be useful in the streaming literature, and many transmission schemes [10] [12] rely on the availability of such feedbacks. Recently, the Audio/Video Transport Working Group in IETF has also recognized the importance of timely feedbacks and submitted an Internet draft [13] to modify RTCP reports in current RTP RFC [5] to include such feedbacks.

## III. Streaming Agent

Streaming agent (SA), enhanced version of the RTP monitoring agent [2], is a network proxy located at the basestation of the transmitting wireless link, or more generally, at the intersection of the wired core network and the transmitting wireless link. During a server-client streaming session such as a RTP session, SA identifies the stream by looking at selected fields of IP packet headers, such as source and destination addresses and source and destination port numbers, and monitors the packet stream. SA periodically sends statistical feedbacks and timely feedbacks to the sending server. Statistical feedbacks, such RTCP reports, are sent in mid-term (in the order of seconds to minutes) and contain observable statistical information such as mean and variance of RTT and packet loss rate. Timely feedbacks are sent in short-term (within a second) and contain packet acknowledgments or user-defined event notifications. The exact frequencies of timely and statistical feedbacks can be preset by the network operator, or by the sender prior to the start of the streaming session via API handshake. If it is the latter, then the duration of the stream must also be specified so the associated session state at SA can be removed when expired. See Figure 1a for an illustration of SA.

In order not to overwhelm the wireless link, a *shaping point*, located topologically just prior to SA, is used to limit the sending rate so that the packet rate is no larger than the wireless link bandwidth. Basically, a layer 3 IP packet queue is located at the basestation to store packets waiting to be fragmented and transmitted in lower layers. If the wireless link condition is poor, the number of retransmissions until successful transmission will be large, causing the IP packet queue to build up. The shaping point reacts to the fullness of the queue by dropping packets prior to packet arrival at SA to avoid eventual queue overflow. See Figure 1b for an illustration. Details of the shaping point are also discussed in [2].

In the same spirit as Snoop agent, SA also maintains the soft state property: while the presence of SA enhance the streaming performance, the absence of it does not lead to catastrophic errors. In the event of

SA system failure and sender fails to receive SA feedbacks, the sender simply adapts the media content to client feedbacks only. While this mode of operation is sub-optimal due to lack of partial path information, it is not catastrophic.

### *A. Deployment Cost of Streaming Agent*

As previously mentioned, streaming agent is an enhanced version of the RTP monitoring agent [2], which sends statistical feedbacks to the sender for proper wired network congestion control. For RTP monitoring agent to function properly, it must perform packet classification to identify flows and maintain session states to generate periodic RTCP reports. Given such an entity already exists, streaming agent simply enhances it by sending additional timely feedbacks. Therefore the deployment cost of streaming agent, given RTP monitoring agent already exists, is minimal.

### *B. Usefulness of Partial Path Information*

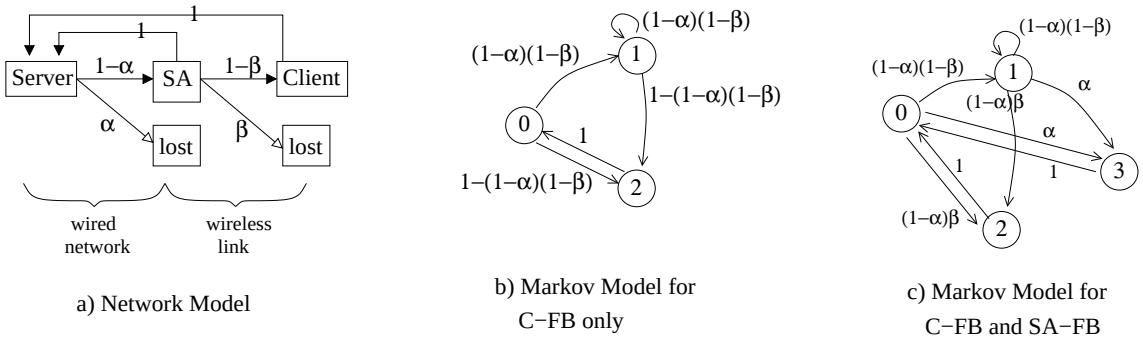
How useful is the partial path information provided by SA's statistical and timely feedbacks? As mentioned before, statistical feedbacks can be used to perform proper wired network congestion control. We do not advertise any particular congestion control algorithm here; we simply claim that since most end-to-end congestion control algorithms [4] [3] operate as a function of mean and/or variance of round-trip-time (RTT) and/or packet loss rates, most can operate properly using SA statistical feedbacks.

In contrast to statistical feedbacks, timely feedbacks deal with information unique to individual packets. For example, SA can monitor the RTP sequence number space in the stream and can send acknowledgments (ACK) when it notices a group of $K$ consecutive packet arrivals. Using the ACKs, we can track the wired client state. Given the wired client state, the server can then better adapt the streaming content in a timely fashion. As an instantiation, we use this type of feedback to perform format adaptation in this paper.

It is interesting to note that SA's timely and statistical feedbacks are complementary in the sense that they are most useful in opposite network conditions. When the wired network is congested and the wireless link is error-free, the timely feedbacks are particularly valuable in reporting packet drops to the sender quickly as the wired client state perfectly mimics the *end client state* — which packets have arrived correctly and on time at the client. On the other hand, when the wired network is not congested but the wireless link is poor, the statistical feedbacks are useful to distinguish wireless link loss from wired network congestion to avoid unnecessary congestion control.

Of course, one can track the end client state using frequent timely feedbacks from the client. In contrast, SA timely feedbacks leads only to wired client state — an estimated end client state at best. Knowing the end client state, which is ultimately what the sender is concerned with, is tracking the wired client state still necessary? Before answering this question, we first note that there are two important scenarios when frequent timely feedbacks from client is not possible:

1. According to current RTP protocol [5], the RTCP reports that receiver generates periodically during an RTP streaming session contain only statistical feedbacks, not timely feedbacks. This means the client must use a non-standard protocol to provide timely feedbacks. The Third Generation Partnership Project (3GPP) has already completed standardization called 3G packet streaming services (3G-PSS) [14] [15] that uses existing RTP and RTCP specifications. Deployment of 3G wireless networks has already begun in Japan, and 3G-PSS compliant handsets have been manufactured and available to the general public since Autumn 2001. Though IETF has begun discussion on modifications to RTP to include timely feedbacks [13], it will be a long lag time before proposed Internet drafts are propagated to new streaming standard, and manufacturers make handsets according to new published standard. Hence, it is important to develop technology flexible enough to support existing standard-compliant handsets and future generation ones.
2. Mobile client typically has severe power constraint due to limited battery life. Because sending data consumes more power than receiving, having client sends frequent feedbacks during the course of the streaming session may not be desirable.




a) Network Model

b) Markov Model for C−FB only

c) Markov Model for C−FB and SA−FB

Fig. 2. Abstract Network Model and Markov Chains for Feedback Systems

Even if the client can send frequent timely feedbacks, SA timely feedbacks are still useful. The reason is that in reality, sender can only track a delayed version of the end client state using client timely feedbacks, where the delay is the RTT between the sender and the client ($RTT_{client}$). Similarly, the sender can track a delayed version of the wired client state where the delay is the RTT between the sender and SA ($RTT_{SA}$). This means that if the transmission delay in the wireless link is large, then knowing a more recent wired client state can still be useful next to a less recent end client state. Moreover, if the wireless link happens to be relatively error free, then the wired client state mimics the end client state well, and having a more recent wired client state becomes all the more useful.

### B.1 Streaming Agent in 3G Wireless Network

To summarize the above argument in a nutshell, how useful SA timely feedbacks depends on how fast SA timely feedbacks are relative to client timely feedbacks, and how precise the wired client state mimics the end client state. It turns out that in the case of today's 3G wireless network, typical one-way delay of radio links is quite large — on the order of 100ms — without link layer retransmission [16]. Assume conservatively that $RTT_{SA}$ is 100ms, SA timely feedbacks are faster then client timely feedbacks by $67\%$. If link-retransmission is performed, as often required due to the unreliable nature of the wireless link, then the difference in delay between SA timely feedbacks and client timely feedbacks is even more significant.

Also unique to the 3G wireless network is the different available transmission modes at the link layer [1]. As an IP packet is passed down from layer 3 to layer 2, the radio link control protocol (RLC) provides segmentation and retransmission services. See Figure 1b for an illustration. Whether and how many retransmissions are done depends on the RLC modes. There are three modes: transparent, unacknowledged and acknowledged. For acknowledged mode, an automatic repeat request (ARQ) is used for error control. The tradeoff between link quality and link delay can be set by the parameter that sets the number of retransmissions. This parameter is set by Radio Resource Control (RRC) during configuration. According to the specification, one can easily adjust the link quality/delay tradeoff by setting the parameter.

Typical 3G wireless service providers use forward error protection in lower layers to lower packet loss rate to a reasonable level under normal conditions (less than $5\%$). If a small number of link-layer retransmission is used as described above, then the wireless link is close to lossless. Given the intended application is streaming media, where a fixed initial delay up to several seconds is tolerable, performing a small number of link-layer retransmission is very beneficial toward improving end-to-end performance. In such case, wired client state very closely mimics end client state, and SA timely feedbacks become very useful.

## IV. Analysis of SA Timely Feedbacks

From previous discussion, it becomes obvious that the region of operation where SA timely feedbacks are most useful, as compare to client timely feedbacks, is where the wired network is congested and the wireless

link is almost lossless. In this section, we analyze the usefulness of SA timely feedbacks over client timely feedbacks in other regions of operation in the *best case scenario* — the most improvement we can expect from using SA feedbacks. Toward this goal, we first construct a simple network model shown in Figure 2a. Essentially, we divide the network into wired and wireless part, each described by a memoryless packet loss model with loss parameter $\alpha$ and $\beta$ respectively. Each packet loss is detected at SA and client and a negative acknowledgment packet (NAK) is sent back to the server. We assume here the feedback packets are never dropped.

For source coding, we first make the assumption that we are using predictive coding, such as a "IPPP. . .P" group of pictures (GOP). In other words, one independently coded frame is followed by a series of dependently coded frames. This assumption seems reasonable as virtually all video coding standards use some form of predictive coding. The second assumption on source coding is that the source coder continues to code frames predictively (P-frames) until a NAK is received from SA/client, at which point a non-predictive frame (I-frame) is used followed by a series of predictive frames. This assumption is made to maximize both the utility of SA and client feedbacks in order to accentuate their difference in performance. Assume further that $RTT_{client}$ is $M$-frame time and $RTT_{SA}$ is $N$-frame time. Let the objective measure be the fraction of frames correctly decoded, where frame $i$ is correctly decoded iff all dependent frames up to the last independent frame are all received correctly.

For the client feedback (C-FB) only case, we constructed the corresponding discrete-time Markov chain in Figure 2b. Starting at initial state 0, with probability $(1-\alpha)(1-\beta)$ client receives frame 1 correctly and we move to state 1 — state representing frames are correctly decoded at the client. With probability $(1-\alpha)(1-\beta)$, each subsequent frame being decoded correctly is also correctly received, until a frame $i$ is lost in the network with probability $1-(1-\alpha)(1-\beta)$. We then move to state 2, representing frames being incorrectly decoded at the client. Note that the subsequent $M-1$ frames, $i+1,\ldots,i+M-1$, are also decoded incorrectly, regardless of whether they are transmitted correctly. The NAK for frame $i$ reaches the server after $M$-frame time, upon which we return to initial state 0 and the process repeats. The transition matrix $P$ for this Markov chain is:

$$P = \begin{bmatrix} 0 & (1-\alpha)(1-\beta) & 1-(1-\alpha)(1-\beta) \\ 0 & (1-\alpha)(1-\beta) & 1-(1-\alpha)(1-\beta) \\ 1 & 0 & 0 \end{bmatrix} \tag{1}$$

Given the discrete time Markov chain described by the transition matrix $P$ is aperiodic and recurrent [17], we can find the invariant probability distribution $\bar{\pi} = [\bar{\pi}_0 \ \bar{\pi}_1 \ \bar{\pi}_2]$. The fraction of correctly decoding frames is scaled by the cost of each state: $f_{client} = \bar{\pi}_1/(\bar{\pi}_1 + M\bar{\pi}_2)$.

For the case when both client and SA send feedbacks, the transition matrix $P$ is:

$$P = \begin{bmatrix} 0 & (1-\alpha)(1-\beta) & (1-\alpha)\beta & \alpha \\ 0 & (1-\alpha)(1-\beta) & (1-\alpha)\beta & \alpha \\ 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix} \tag{2}$$

where state 3 is the state when frames are decoded incorrectly but detected first at SA.

The cost associated with state 2 is now more complicated, because a frame $i$ lost in the wireless link may not necessarily entail a loss of $M$ frames. The reason is that if frame $i+k$ is lost in the wired network and $N << M$, then SA NAK for frame $i+k$ will reach server before client NAK for frame $i$ so server can recover quickly. The cost for state 2 in this case is:

$$M' = M(1-\alpha)^{M-N-1} + \sum_{i=1}^{M-N-1} (N+i)(1-\alpha)^{i-1}\alpha \tag{3}$$

We now get a similar cost expression for $f_{SA}$: $f_{SA} = \bar{\pi}_1/(\bar{\pi}_1 + M'\bar{\pi}_2 + N\bar{\pi}_3)$. To compare the performance of the two cases, we plotted $f = f_{SA} - f_{client}$ in Figure 3. The top plot shows function $f(\alpha,\beta)$ for $N=4$, $M=10$, and the bottom plot shows the function for $N=8$, $M=10$.

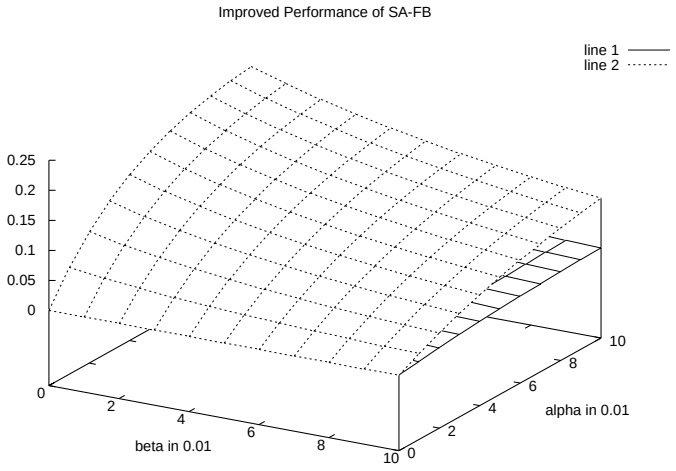
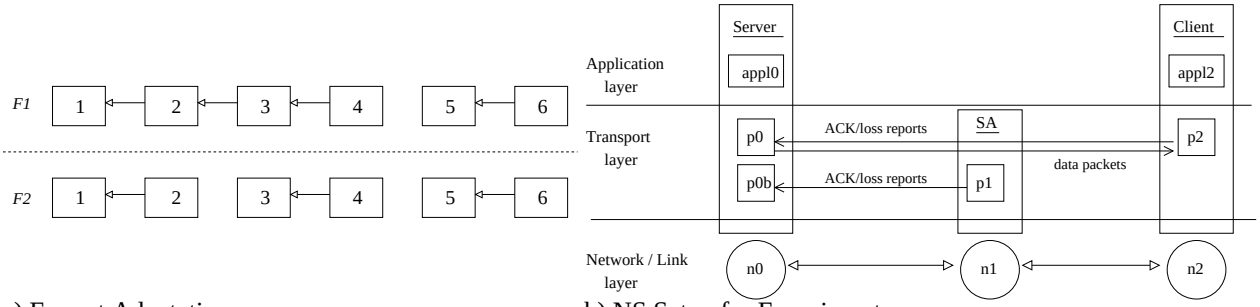



Fig. 3. Improvement of SA-FB plus Client-FB over Client-FB Only



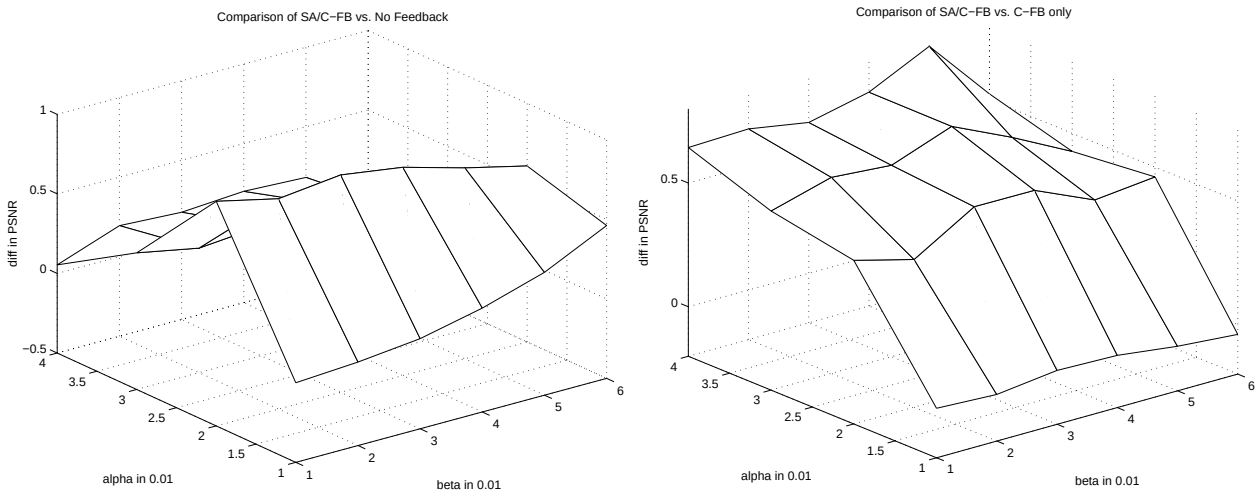

a) Format Adaptation b) NS Setup for Experiment

Fig. 4. Experimental Setup

Several observations can be made. First, notice that SA-FB is most useful when wired network is poor and the wireless link is good. This agrees with our previous discussion. Second, SA-FB usefulness is strongly correlated to the delay difference $RTT_{client} - RTT_{SA}$. This is also intuitive. The more surprising observation is that the usefulness of SA-FB decreases slowly as $\beta$ increases. From this, we can remark on SA-FB usefulness: even if wired client state no longer mimics end client state, as long as the wired network is sufficient lossy, SA-FB is still informative and can improve end-to-end performance.

## V. Context Problem: Format Adaptation

As pointed out in the introduction, we show the effectiveness of SA experimentally in the context of the format adaptation problem. The problem assumes two video streams with same bit rate but different I-frame frequencies are available at the server. A format adaptation strategy is used to switch between the two formats depending feedbacks from client and/or SA. Format switching occurs only at I-frame boundary to avoid drifting problem due to mis-prediction. The actual format adaptation strategy is a simple one: whenever we determine a frame has been lost, we switch to the format with an I-frame in the nearest future. If more than one format satisfies this condition, we select the format with lowest I-frame frequencies (i.e. highest quality). For example, in Figure 4a we see two video stream encoded with different I-frame frequencies. Suppose we start with format $F1$. If we detect a loss in frame 2, we switch to $F2$, since $F2$ has I-frame in frame 3. If we now detect frame 4 is lost, we would switch to $F1$.




a) Comparison of SA/C-FB vs. No Feedback b) Comparison of SA/C-FB vs. C-FB Only

Fig. 5. Comparisons of Feedback Schemes

## VI. EXPERIMENT

### A. Experiment Setup

To demonstrate the effectiveness of our proposed network agent SA in the context of the format adaptation problem, using Network Simulator 2 [18], we implemented a new application inside ns2. It has a transport layer duplex connection (`p0-p2`) from the sender node `n0` to the client node `n2`, and a simplex connection (`p1-p0b`) from the SA node `n1` to the sender node `n0`. See Figure 4b for an illustration.

For our experiment, the links `n0-n1` and `n1-n2` has constant delays and uniform drop rates. An instance of the application, `app0`, sits at sender node and periodically sends packets to the client using the first connection. Each packet has a sequence number in the packet header. There is a filter at the link from `n1` to `n2` that sniffs out the packets target to the client and forwards it to `p1`, who sends ACKs back to the sender using the second connection. The sender decides which format to send based on received ACKs.

### B. Results

The objective measure we are using is average PSNR, calculated relative to the uncompressed original video sequence. A frame $i$ can be incorrectly decoded because that frame packet is dropped during transmission, or a dependent frame earlier in the sequence is dropped — we assume a one-to-one mapping between frame (I- or P-frame) and RTP packet in the experiment. In such incorrect decoding case, the most recently correctly decoded frame $j$ is used for display for frame $i$, and so we calculate the PSNR using original frame $i$ and encoded frame $j$ instead. If no such frame $j$ is available, then PSNR is simply 0.

To generate actual data for our experiment, we use H.263 video codec to encode the first 50 frames of the `carphone` sequence into two streams, $F1$ and $F2$, encoded at QCIF size, 230kps and 20frames/s, and at I-frame frequencies of 5 and 25 frames/s. The same bit rate requirement is achieved by finding a combination of I-frame quantization parameter and P-frame quantization parameter that maximizes PSNR given the bit rate. The resulting average PSNR for the two compressed streams compared to the original uncompressed sequence are 37.01dB and 38.82dB respectively. For each stream, a distortion matrix $d(j, i)$ is generated offline where $d(j, i)$ is the distortion between correctly decoded frame $j$ and original frame $i$, for $j < i$. The two distortion matrices are inputted into ns2 for calculation in the experiment.

We set up parameters for network simulation as follows. We first let one-way wired network delay be

50ms. We let the wired network loss by identically and independently distributed (iid) with loss parameter $\alpha$. For the parameters in the wireless link, we first assume the number of retransmission be 2. So the probability that a packet is correctly delivered across the wireless link is $(1-\beta)+\beta(1-\beta)$. Assuming one-way delay in the link is 100ms, the average delay for the link given a packet is delivered correctly is $\frac{(1-\beta)100+\beta(1-\beta)200}{(1-\beta)+\beta(1-\beta)}$.

We first argue that timely feedbacks are useful in general in the format adaptation problem. With the absence of timely feedbacks, scheme 1 sends either format $F1$ or $F2$ only with no switching — scheme 1 uses the better of the two formats given the network condition. For given $\alpha$ and $\beta$, we find the performance of scheme 1 by finding the average PSNR of the video stream at the client for the `foreman` sequence. We repeated this experiment 1000 times for an averaging effect. We then repeated the experiment for $0 \leq \alpha \leq 0.1$ and $0 \leq \beta \leq 0.05$ at $0.01$ increment. We performed the same procedure for scheme 2 which employs a format adaptation strategy with client and SA timely feedbacks. In Figure 5a, we plotted the PSNR of scheme 2 minus scheme 1 in the region when scheme 2 has higher PSNR than scheme 1. Outside this region, the network condition is so poor that switching to $F1$ does not make sense, and using $F2$ at all time is the best strategy. We found that within the region, scheme 2 is better than scheme 1 with maximum improvement of 0.91dB and an average improvement of 0.33dB. We can safely conclude then that timely feedbacks are indeed useful in the region where format adaptation makes sense. The $(\alpha, \beta)$ parameter space over which scheme 2 is better than scheme 1 is the test region we use for the next part of the experiment.

We now compare the performance of case 2 with the case when the format-adaptation strategy uses only client timely feedbacks (case 3). Using the same procedure, we generated two surfaces corresponding to the two cases for the parameter space determined in the previous part of the experiment. In Figure 5b, we plotted the difference between the two surfaces. We see that the performance of case 2 is better in every point in the test parameter region, with maximum improvement 0.78dB and an average improvement 0.42dB. Notice that performance is most pronounced with $\alpha$ is large — when the SA timely feedbacks are most informative. Notice also the general shape of the surface is similar to our derived surface in Figure 3. We can now conclude that SA timely feedbacks improve end-to-end performance even if client timely feedbacks are available.

## VII. Discussion

The recent Internet Draft titled "Extended RTP Profile for RTCP-based Feedback" [13] discussed extensively how to extend existing RTCP protocol to include timely feedbacks. In particular, a complex timing algorithm is proposed so that the combined timely/statistical RTCP feedbacks do not exceed the $5\%$ session bandwidth that is recommended in the most recent RTP Internet Draft. Since the purpose of the experiment was to show the usefulness of SA timely feedbacks, not statistical feedbacks, we did not perform congestion control and no statistical feedbacks were sent from either the client or SA. For deployment, however, one can follow [13] and have the client and SA each share $5\%$ feedback bandwidth between statistical and timely feedbacks. An alternative option is to limit combined timely/statistical feedbacks from *both* SA and the client to $5\%$. The problem is then how to divvy up the $5\%$ between SA and the client for optimal performance. Instead of solving this difficult problem, we believe that having each of SA and client use at most $5\%$ should not cause adverse network problems. The reason is that for congestion control in wireless media streaming, in order to distinguish wireless link loss from wired network congestion, the server requires partial path information — RTCP reports from SA. This is fundamentally different from the design in [5], where only client's RTCP reports are required. Given this is the case, we are not increasing the feedback bandwidth by having SA and client each share $5\%$ between timely/statistical feedbacks.

Also specified in [13] are the three types of feedback messages: transport layer, payload-specific, application layer. Both payload-specific and application layer messages require the feedback sender to look inside the RTP payload to determine if feedback is necessary. As a network proxy, we require SA to look only at

packet headers of IP packets, and hence SA does not have that knowledge and cannot provide these types of feedbacks. On the same token, SA sidesteps end-to-end security issues by never touching the payload. Hence SA can be used even if the payload is encrypted as described in the secure RTP [19].

## VIII. Conclusion

In this paper, we have proposed extending the capabilities of a network agent, located at the boundary of wired / wireless network, to provide timely feedbacks to the streamer server. Using the agent feedbacks, the sender can track the wired network state and client state, and using these states, can better adapt the media content. In the context of the format-adaptation problem, the improvement of using SA feedbacks can be quite substantial.

Although we argue in this paper SA's usefulness in the context of the format adaptation problem, the usefulness of SA feedbacks is not limited to just that. So a possible future direction is to use SA feedbacks to perform application-level retransmission along the line of [10] [12]. We conjecture that by enabling feedback-retransmission mechanism in the wired part of the network only using SA's partial path information, the performance improvement can be even more dramatic.

## IX. Acknowledgments

The paper benefitted greatly from valuable comments and suggestions from Wai-tian Tan and John Apostolopolous of HP Labs Palo Alto.

## References

[1] H. Holma and A. Toskala, Eds., *WCDMA for UMTS: Radio Access for Third Generation Mobile Communications*, Wiley, 2001.

[2] T. Yoshimura et. al., "Qos control architecture with rtp monitoring agent for mobile mumtimedia streaming," in *IPSJ DICOMO 2001*, June 2001.

[3] S. Floyd et. al., "Equation-based congestion control for unicast applications," in *SIGCOMM*, 2000.

[4] R. Rejaie, M. Handley, and D. Estrin, "Rap: an end-to-end based congestion control mechanism for realtime streams in the internet," in *INFOCOM*, March 1999.

[5] H. Schulzrine et. al., "Rtp: A transport protocol for real-time application," January 1996, IETF RFC 1889.

[6] S. Wee, J. Apostolopoulos, and N. Feamster, "Field-to-frame transcoding with temporal and spatial downsampling," in *ICIP*, October 1999.

[7] T. Yoshimura et. al., "Rate and robustness control with rtp monitoring agent for mobile multimedia streaming," in *IEEE ICC*, April 2002.

[8] H. Balakrishnan et. al., "A comparison of mechanisms for improving tcp performance over wireless links," in *IEEE/ACM Trans. Networking*, December 1997, vol. 5, no.6.

[9] Z. Miao and A. Ortega, "Optimal scheduling for streaming of scalable media," in *Asilomar*, 2000.

[10] M. Podolsky, S. McCanne, and M. Vetterli, "Soft arq for layered streaming media," Tech. Rep. UCB/CSD-98-1024, University of California, Berkeley, November 1998.

[11] V. Chande and N. Farvardin, "Progressive transmission of images over memoryless noisy channels," in *IEEE JSAC*, June 2000, vol. 18, no.6.

[12] P. Chou and Z. Miao, "Rate-distortion optimized streaming of packetized media," in *submitted to IEEE Trans. MM*, February 2001.

[13] S. Wenger et. al., "Extended rtp profile for rtcp-based feedback," July 2001, IETF Internet-draft: draft-ietf-avt-rtcp-feedback-00.txt.

[14] *3GPP TS 26.233 Transparent End-to-End Packet Switched Streaming Services (PSS); General description (Release 4)*, ftp://ftp.3gpp.org/Specs/2001-03/Rel-4/26_series/26233-400.zip, March 2001.

[15] *3GPP TS 26.234 Transparent End-to-End Packet Switched Streaming Services (PSS); Protocols and codecs (Release 4)*, ftp://ftp.3gpp.org/Specs/2001-03/Rel-4/26_series/26234-400.zip, March 2001.

[16] G. Montenegro et. al., "Long thin networks," January 2000, IETF RFC 2757.

[17] G. Lawler, *Introduction to Stochastic Processes*, Chapman & Hall, 1995.

[18] "The network simulator ns-2," June 2001, release 2.1b8a, http://www.isi.edu/nsnam/ns/.

[19] R. Blom et. al., "The secure real time transport protocol," February 2001, IETF Internet-draft.