MUVIS: Multi-source Video Streaming Service over WLANs

Danjue Li, Chen-Nee Chuah, Gene Cheung and S. J. Ben Yoo

Abstract: **Video streaming over wireless networks is challenging due to node mobility and high channel error rate. In this paper, we propose a multi-source video streaming (MUVIS) system to support high quality video streaming service over IEEE 802.11-based wireless networks. We begin by collocating a streaming proxy with the wireless access point to help leverage both the media server and peers in the WLAN. By tracking the peer mobility patterns and performing content discovery among peers, we construct a multi-source sender group and stream video using a rate-distortion optimized scheme. We formulate such a multi-source streaming scenario as a combinatorial packet scheduling problem and introduce the concept of asynchronous clocks to decouple the problem into three steps. First, we decide the membership of the multi-source sender group based on the mobility pattern tracking, available video content in each peer and the bandwidth each peer allocates to the multi-source streaming service. Then we select one sender from the sender group in each optimization instance using asynchronous clocks. Finally, we apply the point-to-point rate-distortion optimization framework between the selected sender-receiver pair. In addition, we implement two different caching strategies,** *simple caching simple fetching (SCSF)* **and** *distortion minimized smart caching (DMSC)***, in the proxy to investigate the effect of caching on the streaming performance. To design more realistic simulation models, we use the empirical results from corporate wireless networks to generate node mobility. Simulation results show that our proposed multi-source streaming scheme has better performance than the traditional server-only streaming scheme and that proxy-based caching can potentially improve video streaming performance.**

Index Terms: **Caching strategies, multi-source video streaming, rate-distortion optimized packet scheduling.**

I. INTRODUCTION

In recent years, there has been an increasing demand to deliver high-quality and high-bandwidth video streams over wireless networks. However, wireless networks present a number of unique challenges as compared to delivering the same multimedia content through the traditional wired networks. In particular, the end-to-end perceived video quality can fluctuate vastly due to higher channel bit error rate, channel fading, interference, and end host mobility.

Past literature on video streaming schemes over wireless networks have advocated protection-based approaches to address the above challenges. Krishnamachari et al.[1] proposed an adaptive cross-layer protection strategy to enhance the quality of scalable video transmission. Majumdar et al.[2] presented a hybrid FEC/ARQ scheme to increase the robustness of video streaming in IEEE 802.11 wireless LAN (WLAN). Wang et al.[3] suggested using a video proxy in the base station to reduce ARQ delay. While these approaches enhance the performance of video streaming over wireless networks, none of them have considered the problem from the perspective of rate-distortion optimized streaming [4].

The goal of this paper is to design a video streaming scheme that can be used to provide high quality video streaming service such as video-on-demand over WLAN. Instead of using the traditional *server-client* service model, we propose a joint server/peer video streaming architecture to leverage peer resources in WLAN and optimize the streaming performance from the perspective of a single wireless client. We consider the scenario where part of the multimedia content desired by the client already resides on peers that are located in the same WLAN. Connections to these peers typically have shorter delay and better performance than communicating with a remote video server. However, the reliability of wireless connections to these peers heavily depends on the node mobility and physical channel characteristics.

Our contributions in this paper are threefold.

- First, we propose a novel multi-source video streaming (MUVIS) system for delivering high quality video content over WLANs. By leveraging nearby wireless peers and the remote media server to form a joint sender group, our architecture can shorten the transmission delay and decrease the quality degradation due to packet losses and bandwidth variation. In addition, the proposed architecture can perform on-the-fly content discovery to locate the desired video data among peers and track peer mobility to maintain a more stable joint sender group. We introduce a proxy-driven streaming scheme, which relies on a proxy collocated with the access point (AP) to coordinate among multiple senders and perform rate-distortion optimized streaming/caching. Compared with a sender-driven approach, the proposed proxy-driven streaming scheme avoids the synchronization problem among multiple senders and can more easily adapt to instantaneous changes in the network.
- Second, we formulate the multi-source (server/peers) streaming as a combinatorial packet scheduling problem and decouple it into two steps: first selecting the sender (server or one of the peers) from the chosen sender group and then applying point-to-point rate-distortion optimized streaming scheme between a specific sender-receiver pair. To solve the sender selection problem, a set of asynchronous clocks are introduced at the proxy and each clock is responsible for one particular sender-receiver pair. In addition to sender selection, asynchronous clocks can

Danjue Li is with the Department of Electrical and Computer Engineering, University of California-Davis, USA, email: dli@ucdavis.edu. Chen-Nee Chuah is with Department of Electrical and Computer Engineering, University of California-Davis, USA, email: chuah@ece.ucdavis.edu. Gene Cheung is with Hewlett-Packard Laboratories Japan, Japan, email: gene-cs.cheung@hp.com. S. J. Ben Yoo is with Department of Electrical and Computer Engineering, University of California-Davis, USA, email: sbyoo@ucdavis.edu.

1229-2370/03/$10.00

also be used to perform rate control for streaming service by properly choosing clock periods to satisfy rate constraints.

- Third, we propose a smart frame-level caching management strategy, *Distortion Minimized Smart Caching (DMSC)*, to manage the cache in the proxy. Instead of performing simple caching according to data arrival order until the cache is fully occupied, DMSC will selectively cache data units based on their relative importance so that local retransmission can maximally reduce the distortion perceived by the client under a certain cache size constraint. Moreover, by using DMSC, the proxy can fetch data units from the cache in a rate-distortion optimized way. Our simulation experiments show that under typical WLAN conditions, our proposed DMSC strategy can consistently improve the system performance by $1-3$dB compared with the traditional simple caching scheme.

The rest of the paper is organized as follows. We describe the related work in Section II and present the proposed MUVIS system architecture in Section III. In Section IV, we introduce the proxy-driven streaming scheme and analyze its performance in Section V. In Section VI, we will discuss how the proxy cache affects performance modeling and introduce a detailed cache management mechanism. In Section VII, we present NS simulation results to evaluate our proposed schemes. We conclude the paper and discuss future research directions in Section VIII.

II. RELATED WORK

In this section, we will briefly review research works related to this paper.

A. Sender diversity in video streaming

Using sender diversity to enhance the video streaming quality has been actively explored in the past literature [5][6][7][8]. Xu et al. [5] proposed a peer-to-peer video-on-demand system using multiple description and sender diversity, where they use multiple ordinary computers (peers) as servers and the client can stream different layers of the same video file from these peers. Nguyen et al. [6] applied a receiver-driven rate allocation algorithm to determine the rate for each server by taking into account available network bandwidth, channel characteristics, and a pre-specified, fixed level of forward error correction to minimize the probability of packet loss. Meanwhile, they proposed a packet partition algorithm for the sender side to ensure that no packet is sent by more than one server. Hefeeda et al. [7] proposed a video streaming system that leverages underlying peer-to-peer streaming support. Their proposed peer selection method, called *topology-aware selection*, relies on the underlying topology of sender candidates and network connection qualities to infer *goodness* of the peers to choose the best senders. The most closely related work is CoolStreaming [8], which is a gossip-based peer-to-peer streaming system. Each user can maintain connections to multiple peers and swap information among them with certain delay constraints.

However, not all the above works are specifically designed for video streaming over WLAN and none of them perform frame-wise sender selection. In contrast, our proposed scheme performs frame-wise sender selection by taking into account available network bandwidth, channel characteristics, and peer dynamics. Moreover, none of above approaches consider rate-distortion optimized streaming, which is the main idea that underpins the proposed MUVIS system. After pre-fetching a rate-distortion preamble from the media server, the MUVIS proxy will run the rate-distortion optimization algorithm to schedule packet transmissions between the client and multiple senders.

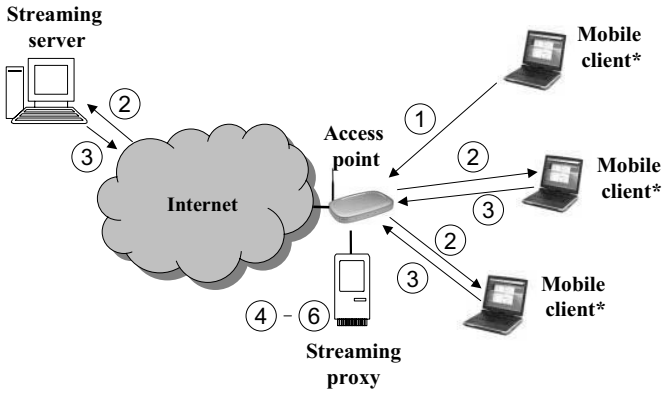
B. Caching strategies for video streaming over the Internet

Video caching has also been used in the past literature to improve the quality of multimedia streaming systems. Early papers [9][10] proposed and analyzed algorithms to cache intervals of video data in the main memory. Earlier clients can then satisfy multiple future clients that request the same video file a little bit later. Tewari et al. [11] defined a new disk-based caching policy called the resource-based caching (RBC) algorithm, which considers bandwidth as well as storage capacity constraints and caches a mixture of intervals and full files that have the greatest caching gain. As an improvement over RBC, Almeida et al. [12] proposed a pooled RBC policy to allow sharing allocated bandwidth of cached files. Whenever a new full file is added to the cache, its bandwidth allocation is added to the bandwidth pool. When a request for a cached file arrives, a new stream from the bandwidth pool is assigned (if possible) to that request. When it finishes delivering the file, the bandwidth is returned to the pool and can be assigned to a new request, even if that request is for a different cached file. Rejaie et al. [13] defined a frequency-based caching (FBC) policy to simply cache the files or partial files that are estimated to have the highest access frequency at the current time. In a complementary work, Sen et al. [14] proposed a prefix caching strategy that caches the initial frames of the video stream to reduce the client start-up delay and decrease variation in quality. Selective partial caching [15][16] has also been introduced to accommodate variations in media display rate at the client end.

Motivated by the above studies, we propose a smart frame-based caching strategy called Distortion Minimized Smart Caching (DMSC) to manage the cache. DMSC combines the design idea of RBC and selective caching to determine which video frames should be cached in order to optimally improve the quality at the client end. DMSC accomplished this using the rate-distortion optimization framework.

C. Rate-distortion Optimized Streaming

To the best of our knowledge, the work by Chou et al. [4] is the first work to systematically define and solve the point-to-point rate-distortion optimized video streaming problem. Our work heavily leverages their ideas, which we combine with the concept of asynchronous clocks introduced in our previous work [17] to implement multi-source diversity streaming. There are a number of relevant works [4] [18] [19] [20] based on the point-to-point rate-distortion optimization problem. Chakareski et al. [18] focused on single path streaming and proposed a hybrid receiver/sender driven streaming scheme, while our work focuses on streaming video from multiple paths simultaneously. Both Chakareski et al. [19] and Begen et al. [20] considered path diversity for media streaming in a receiver-driven rate-distortion optimized framework, while the former specified a single constraint on the expected overall transmission rate from the senders to the client. In contrast, we consider M rate con-



* Equipped with IEEE802.11 interface cards

(a)

- STEP1: Client sends a **SREQ(VFN, UID, SR)** message to the proxy when it wants to stream video file using MUVIS.
- STEP2: Proxy sets a timeout t_{SRTO} and broadcasts a **MREQ(VFN, UID, SR)** message to the media server and all the mobile users in the WLAN.
- STEP3: Server replies a **SACK(RDP, AB)** message once the request is authorized. Willing peers send a **PACK(CL, UID, AB)** message back to the proxy.
- STEP4: Proxy builds up a content table to save all the replies. When the timeout t_{SRTO} expires, proxy builds up a joint sender group based on the replies and mobility tracking results.
- STEP5: Proxy runs the rate-distortion optimized streaming scheme to start streaming data from different senders.
- STEP6: Peer sends out the **PUPD(CL, UID, AB)** message to the proxy if the available video content changes over time. Proxy updates content table once it gets a **PUPD(CL, UID, AB)** message or detects the peer departure.

(b)

Fig. 1. Multi-source video streaming (MUVIS) (a) system architecture (b) streaming session setup procedure

straints for M distinct delivery paths. Moreover, both of these works focused on a general static path diversity streaming problem, that is, the number of paths was fixed during the entire video streaming session. They did not consider the dynamic membership of senders. In our work, we must address the dynamic group membership problem due to the nomadic nature of wireless peers (potential senders).

III. MULTI-SOURCE VIDEO STREAMING SYSTEM ARCHITECTURE

In this section, we present our proposed architecture for streaming video over WLANs. We will start with an overview and then discuss the content and sender discovery mechanisms in details.

A. System overview

Figure 1(a) illustrates our proposed multi-source video streaming (MUVIS) system architecture, where the client can simultaneously stream from the remote media server and nearby peers in the WLAN. In this paper, we assume all the mobile users (MU) are cooperative and have subscribed to the streaming service provided by MUVIS. Traditionally, the client subscribing to a streaming service, such as video-on-demand (VoD), will completely rely on the remote media server to provide the video content. When a flash crowd appears, the media server can easily be overloaded. In addition, the quality of the connection traversing the Internet can frequently degrade due to network congestion and link failure [21]. This simple *server-client* model overlooks the possibility that the desired video content may already exist among peers located in the same WLAN. Compared with the connection to the remote media server, connections to peers usually have shorter delay and potentially higher bandwidth. In the proposed MUVIS system, we explore the possibility that the desired video content may exist among peers and we make those nearby resources accessible to video subscribers. With the coordination provided by a streaming proxy, which is physically collocated at the access point (AP), a MUVIS client can establish multiple connections simultaneously to its peers in addition to the connection with the media server. In the case when server-proxy bandwidth is smaller than proxy-client bandwidth, such multi-source streaming can obtain extra bandwidth from peer senders and the aggregated streaming rate will increase. Even in the case when server-proxy bandwidth is larger than proxy-client bandwidth, multi-source streaming can still improve quality of transmission by using peer connections to shorten the transmission delay. In contrast to our previous work [17], we only use the *infrastructure* communication mode of the IEEE 802.11 interface card when designing MUVIS system and all the traffic between the client and its peers have to go through the access point.

Figure 1(b) summarizes the steps involved in establishing a MUVIS session. When a MU joins the WLAN, it sends an ***Association Request (AREQ)*** message containing its user ID (UID) to the AP. When it leaves the WLAN, it will send another message, ***Disassociation Request (DREQ)***, to terminate the existing association. The proxy will keep track of the association/disassociation behaviors for each MU in the WLAN to monitor peer mobility. Whenever the client wants to start streaming a video file, it will send out a request to the proxy, which will be immediately forwarded to the remote media server. Meanwhile, the proxy will also broadcast it to the peers in the WLAN and set up a timeout t_{SRTO} for receiving replies. After authorizing the request, the media server will reply with a ***Server Acknowledge (SACK)*** message to the proxy with the *Rate-Distortion Preamble (RDP)* included. Here, the RDP is the directed acyclic graph (DAG) representation of the desired video file, which will be used by the proxy to perform the rate-distortion optimized packet scheduling. We will discuss the DAG in more detail in Section IV-A.1. Upon receiving the broadcast request, peers which can contribute to the multi-source streaming will send a ***Peer Acknowledge (PACK)*** message back to the proxy. When the timeout t_{SRTO} expires, the proxy will construct a joint sender group based on the replies and peer mobility history and schedule streaming from multiple senders to the client.

B. Content discovery among peers

In general, the remote media server stores a full copy of the desired video file while peers may only have part of it. This assumption about partial content availability at peers is reasonable because peers may delete part of the content after viewing due

SSID	UID	Content Portions	Allocated Bandwidth
12	server	$[0s, 60s)$	variable
	8088	$[0s, 5s)$	$100kbps$
	8092	$[50s, 60s)$	$200kbps$

Table 1. Content table

UID	association time	disassociation time
8088	$10:10:10am$	$10:23:19am$
	$10:39:10am$	$10:53:19am$
	$10:59:10am$	$11:08:09am$
8092	$10:10:10am$	$10:43:10am$
	$10:45:10am$	$11:23:19am$

Table 2. Mobility tracking table

to storage limitations or have only downloaded the beginning portions of the video due to lack of interest and/or early exit in their previous streaming sessions. Performing content discovery is necessary for locating the desired video content among peers.

The content discovery process proceeds as the following. Once the proxy gets the ***Streaming Request (SREQ)*** message from a client, it will initiate a multi-source streaming session for this client and randomly generate an unused integer number as the streaming session ID (SSID). Then the proxy will start a timer t_{SRTO} and broadcast a ***Media Request (MREQ)*** message to the remote media server and all the MUs in its transmission range, including the desired video file name (VFN), client's UID, and the preferred streaming rate (SR). The request will be rebroadcast if no reply arrives at the proxy before t_{SRTO} expires. If the proxy gets no reply before t_{SRTO} expires, it will send a ***System Busy*** message to the client. The client will wait a random time before it sends another ***SREQ*** to the proxy. When MUs receive the request from the proxy, they will search their local archives to see if they have the desired video content. Any MU that has kept full or partial copies of the desired content and is willing to share will reply to the proxy with its UID, a content list (CL) showing which portion of the requested video content it has, and the allocated bandwidth (AB) for this session. Meanwhile, the media server will also reply with its content information and allocated bandwidth after authorizing the request. After t_{SRTO} expires, the proxy will construct a content table as shown in Table 1. To keep the information in Table 1 up-to-date, the proxy will send out a periodical probing beacon to detect the existence of MUs. At the same time, whenever the available content for video file VFN changes in an MU, the MU will automatically send a ***Peer Update (PUPD)*** message to update its entry in the content table. Once an MU leaves the WLAN, the proxy will delete the corresponding entry from the content table upon receiving the ***DREQ*** message.

C. Joint sender group membership

The mobility of MUs makes membership of the candidate sender group rather dynamic. This can potentially lead to high variation of the video quality perceived by the client. To smooth the video quality variation, we try to enlist the more stable nodes into the joint sender group. In this paper, we characterize the mobility using two metrics, session duration and revisit interval. The session duration refers to the amount of time that a user stays associated with an access point (and hence served by our proxy) before moving to another access point or leaving the network, while revisit interval refers to the amount of time before the next visit of the MU. Longer session durations and shorter revisit intervals represent lower mobility. By keeping track of these two metrics, we can infer how *stable* an MU can be as a potential video source.

Our proposed streaming scheme relies on the proxy to keep track of peer mobility by recording the association/disassociation behaviors for each MU in the WLAN, as illustrated by Table 2. When the proxy gets a streaming request from the client, it will use the information in Table 2 to generate a mobility factor m_j for each MU, as illustrated by Figure 2. T_j^a is the set of association times for MU j, T_j^d is the set of dissociation times for MU j, $\Re_j$ is the set of session duration times R_j for MU j, $\Im_j$ is the set of revisit intervals V_j for MU j, and C_1 and C_2 are weighting coefficients. A larger m_j value represents higher stability.

After checking the mobility level of each peer in the content table, the proxy will select peers that are less mobile as potential senders. In this paper, we simply pick peers starting with the most stable one until the aggregated bandwidth from peers and the server reaches the link capacity between the proxy and the client. A comprehensive peer selection algorithm is out of scope of this paper and we will explore it in the future work.

```
1  if The proxy detects that MU j joins the WLAN at time t_a
2        if MU j has not visited WLAN in the past, i.e. T_j^a ≡ ∅
3              T_j^a ← dummyvalue
4        else
5              T_j^a ← {T_j^a, t_a}
6  if The proxy detects that MU j leaves the WLAN at time t_b
7        if MU j has not visited WLAN in the past, i.e. T_j^d ≡ ∅
8              T_j^d ← dummyvalue
9        else
10             T_j^d ← {T_j^d, t_b}
11
12 ℜ_j ← T_j^d − T_j^a, ℑ_j ← T_j^a(k) − T_j^d(k − 1)
13 if The proxy needs to construct a joint sender group
14       for Each MU listed in the content table
15             m_j ← E{R_j} / (E{V_j}(C_1 σ^2(R_j) + C_2 σ^2(V_j)))
```

Fig. 2. Pseudo-code for Generating Mobility Factor for Mobile User

IV. PROXY-DRIVEN RATE-DISTORTION OPTIMIZED VIDEO STREAMING

The MUVIS system uses a proxy-driven rate-distortion optimized video streaming scheme. The basic idea of this scheme is that under certain network constraints, the proxy will optimally schedule requesting transmission of data units from different senders and relaying them to the client to minimize the perceived quality degradation. In other words, for each DU, the proxy has to decide: when and from which sender to request the corresponding packet(s). Here, we use the client-perceived distortion to measure the quality degradation, which includes two parts: the distortion caused by quantization during source coding, and the distortion caused by network transmission. In

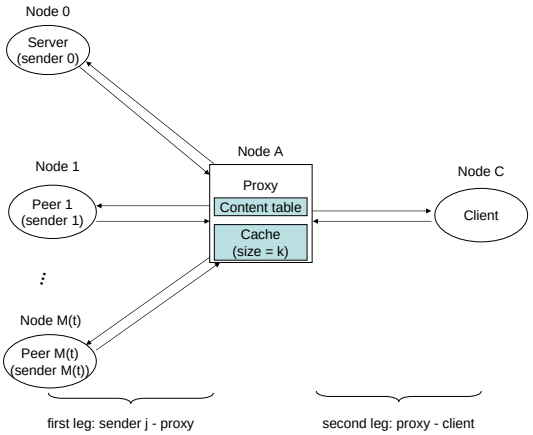


Fig. 3. System model for proxy-driven joint server/peers video streaming

this paper, we concentrate on the second type of distortion, i.e., minimizing the network-caused distortion under certain network constraints.

Directly solving such a combinatorial packet scheduling problem is hard, and the computation complexity makes it infeasible for a real-time application. Instead, we decouple it into two steps. In each transmission opportunity (to be defined later), we first select a sender (server or one of the peers) and then apply point-to-point rate-distortion optimization (RaDiO) algorithm to schedule the packet transmission between the selected sender-receiver pair. Before we start the detailed discussion of the decoupling process, we first present our system model.

A. System model

Figure 3 illustrates the system model for the proxy-driven streaming scheme, which includes a media server (node 0), a proxy (node A), a client (node C), and a set of peers (node m, $m \in \{1, ..., M(t)\}$). $M(t)$ is the number of peers in the multi-source joint sender group at the given time t. At this point, we assume that there is no cache in the proxy.

A.1 Source Model

We assume that a compressed video representation has been assembled into data units with one frame per data unit. Since the video file is predictively encoded, data units in the bit stream will be dependent upon each other. We use a *directed acyclic graph (DAG)* [4] to model such dependency, where each DU_i represents a frame i and a directed edge represents the dependency relation between two data units. In this formulation, data unit is the smallest granularity we will consider in the optimization. Parameters associated with each DU_i include: the size n_i, the decoding time T_i, and the distortion reduction D_i. The size n_i is the size of DU_i measured in number of RTP packets. The decoding time T_i is the time by which DU_i must arrive at the client. D_i is the distortion reduction if DU_i arrives at the client on time and is successfully decoded.

A.2 Network model

In Figure 3, we denote the channel between node m and node n as ζ_{mn}, $m, n \in \{0, ..., M(t), A, C\}$ and $m \neq n$. Each channel includes one forward path and a backward path that are modeled as independent time-invariant packet erasure channels with random delay. Here, we define the forward path as the path from the node m to the node n, and the backward path as the reverse. We define the packet loss as one minus the probability that a packet transmitted by node m is successfully received by the node n. Hence, for ζ_{mn}, we can characterize its forward path by random loss ϵ^F_{mn} and delay density $f^F_{mn}(x) = e^{-\gamma^F_{mn}x}$, $x \geq 0$, where γ^F_{mn} is a constant determined through measurements [22]. Then, the packet sent by node m at time T will be correctly received by node n by time T' with the following probability:

$$p^F_{mn}(T' - T) = (1 - \epsilon^F_{mn}) \int_0^{T'-T} f^F_{mn}(x)dx \qquad (1)$$

Similarly, the backward path for ζ_{mn} can be characterized by ϵ^B_{mn} and delay density $f^B_{mn}(x) = \gamma^B_{mn} e^{-\gamma^B_{mn}x}$, $x \geq 0$. The packet sent by node m at time T will be received correctly by node n at time T' with probability $p^B_{mn}(T' - T)$, which has the same form as (1).

Thus, for ζ_{mn}, the probability that a request (or data unit) sent by node m at time T to the node n will result in a data unit (or an acknowledgment) successfully arriving at node m by time T', $P_{mn}(T' - T)$, will be:

$$P_{mn}(T'-T) = (1-\epsilon^F_{mn})(1-\epsilon^B_{mn}) \int_0^{T'-T} f^F_{mn}(x) * f^B_{mn}(x)dx \qquad (2)$$

where $*$ denotes convolution.

A.3 Network constraints

To prevent link overloads or potential congestion problems, the streaming rate we use for each connection in MUVIS system has to satisfy certain constraints set by the network. The connection to the media server, since it will traverse the Internet, needs to be TCP-friendly. In order not to claim more bandwidth than what a normal TCP connection would use under the same network conditions, the maximum streaming rate Ω_j for sender j without causing congestion collapse is determined by the well-known TCP-friendly rate control (TFRC) [23]:

$$\Omega_j = \frac{L}{\mu_j\sqrt{2\alpha_j/3} + t_{RTO}(3\sqrt{3\alpha_j/8})\alpha_j(1 + 32\alpha_j^2)} \qquad (3)$$

where μ_j is the round-trip time, α_j is the loss event rate perceived by the receiver and t_{RTO} is TCP retransmission timeout value. According to Floyd et al. [23], it is reasonable and practical to estimate t_{RTO} using $t_{RTO} = 4\mu_j$. Let j in (3) be 0 for the connection to media server.

TFRC might result in bandwidth under-utilization in wireless network environment. This is due to the fact that end users cannot distinguish between packet loss due to bit error versus network buffer overflow [24]. However, the design of a high throughput rate control scheme for streaming video over WLAN is not the focus of this paper. As a starting point, we still use

TFRC to decide the maximum non-congested streaming rate over wireless channels. We segment the connection between the sender j and the client into two parts: sender j to the proxy and the proxy to the client, and implement TFRC for each segment.

Based on the above discussion, we can summarize the rate control policy as follows. For each connection between a selected sender (Node $j, j \in \{0, \ldots, M(t)\}$) and the proxy (Node A), we implement TFRC using (3) with parameters μ_j and α_j corresponding to the measured mean round-trip time (RTT) and loss event rate on that connection. Since the proxy aggregates the traffic from multiple sources, we need a separate TFRC connection between the proxy and the client to avoid overloading the channel and affecting other non-video traffic.

B. Sender selection using asynchronous clocks

There are two factors that will affect sender selection: the allocated bandwidth W_j of sender j and its maximum non-congested streaming rate Ω_j. The sender that can allocate more bandwidth for streaming and has better connection with the client will be selected to stream data to the proxy more often. Based on the above criteria for sender selection, we introduce the concept of *asynchronous clocks*. The idea is to set up a clock j ($j \in \{0, \ldots, M(t)\}$) at the proxy for each sender-receiver pair j. This clock j wakes up at regular intervals of Δ_j and is inversely proportional to the channel quality and W_j. Once a clock j wakes up, it signals that a data unit transmission opportunity is immediately granted for the proxy to request a data unit from sender j. After the proxy initiates the request, the clock will be reset to wake up after another Δ_j.

Given Ω_j, as defined by TFRC (3), and W_j (included in the ***PACK*** message), we can define Δ_j as:

$$\Delta_j = \begin{cases} \frac{L}{\Omega_j}, & j = A \\ \max\{\frac{L}{\Omega_j}, \frac{L}{W_j}\}, & j = 0, 1, \ldots, M(t) \end{cases} \quad (4)$$

where we assume that packet size L is the same for all senders.

To couple the data transmission deicsion with the awakening of asynchronous clocks, we introduce the concept of a *transmission token*, which denotes the permission to use a connection. Once an asynchronous clock wakes up, a transmission token will be assigned to the selected sender. We associate one transmission token for each TFRC connection, and whoever gets the transmission token can use that connection for transmission.

As illustrated by Figure 3, all the packets requested from different senders will be forwarded directly to the client upon arrival at the proxy. Since the end-to-end transmission path includes both the sender-proxy connection and the proxy-client connection, the proxy will wait until it collects transmission tokens in both the first leg (sender-proxy pair j) *and* the second leg (proxy-client) before sending a request to sender j. If more than one clock expire on the first leg while waiting for the clock on the second leg, the proxy will suspend all these clocks until the clock on the second leg wakes up. Once the clock on the second leg wakes up, the proxy will pick one sender for the first leg that has the best connection quality (equivalent to shortest clock period). After selecting a sender j, the proxy will then pick a data unit to request from sender j, release any obtained transmission tokens, and reset the suspended clocks.

C. Point-to-point rate distortion optimized video streaming

In each transmission opportunity, after identifying a sender using asynchronous clocks, the problem will be simplified into a point-to-point RaDiO packet scheduling problem.

The motivation of RaDiO is to provide an alternative solution to a heuristic approach to schedule time-critical data based for transmission. Specifically, it will address the following scheduling issue: given a set of interdependent video packets with their respective deadlines, a set of scheduling slots (also called transmission opportunities), and a set of streaming rates constrained by network status, which packet should be sent in each slot to yield the lowest distortion perceived by the receiver? Chou et al. [4] has systematically defined and solved this problem for a lossy packet-switched networking environment. Our work builds on this and extends it to a decoupled multi-source streaming problem.

C.1 Optimization window

Once the proxy obtains the tokens required for requesting a data unit from sender j, an optimization instance will appear. At any given optimization instance t_o, an optimization window equal to N frame-time is selected. The window is defined to be the set of data units whose delivery deadline falls within start time, $start(t)$, and end time, $end(t)$. Data units are brought into the optimization window by $start(t)$. By keeping the window small, it keeps the optimization computationally feasible and the instantaneous client buffer small. When data units cannot be reasonably be expected to be delivered to the client on time, $end(t)$ expires them. The slope of both functions — the rate at which the window advance in time — is the playback speed at the client. The optimization is performed again in P seconds, which is equivalent to the time elapsed before next transmission opportunity appears. Figure 4 is a plot of the playout deadline T of data units against the proxy running time t, where d is the playback delay at the client.

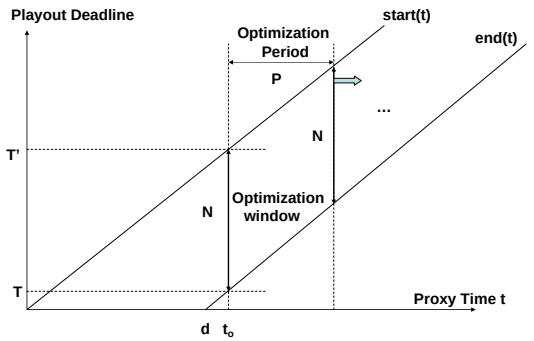


Fig. 4. Optimization Window.

C.2 Transmission policy

We now describe the transmission process of a data unit. The proxy can request the data unit from the sender for a certain period of time, which may start at the time when it first gets included into the optimization window and end at the time when it is due at the receiver, i.e., the playout deadline. A discrete set of transmission opportunities (decided by asynchronous clocks) within this period represents the times at which the data unit may be (re)transmitted from a selected sender.

For each $DU_i, i \in \{1, 2, \ldots, N\}$, we define a transmission policy π_i, where $\pi_i \in \Pi$. Π corresponds to a family of transmission schedules, which dictates when and how the data unit should be requested/transmitted. Let $\pi = \{\pi_1, \ldots, \pi_N\}$ be the transmission vector for N data units. Let π_i be defined as $\pi_i = \{H_i, c_i\}$, where H_i is the transmission history of DU_i and c_i is the transmission decision which determines if DU_i is chosen for (re)transmission at current transmission opportunity.

Given the source model defined in IV-A, the goal of point-to-point RaDiO is then to choose the transmission policy π given the set of asynchronous clocks, deadlines, distortion reduction, network feedback, packet loss, and delay distribution to minimize the distortion defined by:

$$D(\pi) = \{D_0 - \sum_{i=1}^{N} D_i \prod_{l \preceq i} q_l(\pi_l)\} \quad (5)$$

where $l \preceq i$ denotes the set of DU_l's that precede or are equal to DU_i in the DAG. D_0 is the overall expected distortion for the group given no data unit is received, D_i is the distortion reduction if DU_i is successfully decoded, and $q_l(\pi_l)$ is timely arrival probability under the transmission policy π_l.

V. PERFORMANCE MODELING

In this section, we will formulate the proxy-driven MUVIS scheme proposed in the previous sections.

A. Problem statement

Consider N data units in a selected optimization window for (re)transmission that have not yet arrived at the client correctly. As previously discussed, an optimization opportunity will appear when the proxy collects tokens at both the first leg and the second leg as shown in Figure 3. Now the remaining question is *which* data unit should be picked among the N data units in the window.

Let l_i be the number of previous transmission attempts for DU_i. For each attempt $k \leq l_i$, we define a time stamp $t_i^{(k)}$ and a sender ID $s_i^{(k)}$ to record when and from which sender the proxy requested DU_i. For each data unit $DU_i, i \in \{1, 2, \ldots, N\}$, we define its transmission policy π_i as $\pi_i = \{H_i, c_i\}$, where $H_i = \left\{(t_i^{(1)}, s_i^{(1)}), \ldots, (t_i^{(l_i)}, s_i^{(l_i)})\right\}$, $c_i = (t_o, j)$ if DU_i is selected to be requested from sender j and $c_i = \emptyset$ otherwise.

Based on the channel model in Section IV-A.2, the probability that a request sent by the proxy to sender j at time T will result in a data unit successfully arrival at the client by time T' is:

$$\begin{aligned}\Theta_j(T'-T) &= (1-\epsilon_{Aj}^F)(1-\epsilon_{Aj}^B)(1-\epsilon_{AC}^F) \\ &\quad \times \int_0^{T'-T} f_{Aj}^F(x) * f_{Aj}^B(x) * f_{AC}^F(x)dx \quad (6)\end{aligned}$$

Using $\Theta_j(T'-T)$ and the source model defined in Section IV-A.1, the probability of successfully receiving data unit i, $q_i(\pi_i)$, is:

$$q_i(\pi_i) = 1 - (1 - \Theta_{c_i}(T_i - t_o)) \prod_{k=1}^{l_i} \left(1 - \Theta_{s_i^{(k)}}(T_i - t_i^{(k)})\right) \quad (7)$$

where $\Theta_{c_i}(T_i - t_o) = 0$ if c_i is $\emptyset$.

Therefore, we can deduce the expected distortion $D(\pi)$ for N data units at the client using (5).

B. Solution

To solve the scheduling problem defined in (5), we will decouple it into two steps: first selecting a sender using asynchronous clocks and then applying point-to-point RaDiO framework to select a DU for (re)transmission. We already show how to set up asynchronous clocks in section IV. In this section, we will focus on how to select a DU using the RaDiO framework. To do that, we will leverage the work of Chou et al. [4]. Following their discussion, the optimal data unit DU_i for (re)transmission is the one with the largest $\lambda_i = \lambda_i' S_i / n_i$, where λ_i' is the increase in successful-delivery likelihood given one transmission is sent at the optimization instant and S_i is data sensitivity. λ_i' and S_i can be defined as the following:

$$\lambda_i' = q_i(\pi_{i,1}) - q_i(\pi_{i,0}) \quad (8)$$

$$S_i = \sum_{k \succeq i} D_k \prod_{\substack{l \preceq k \\ l \neq i}} q_l(\pi_l) \quad (9)$$

where $\pi_{i,1} = \{H_i, (j, t_o)\}$ is the transmission policy of DU_i given one more transmission request is sent to j at time t_o, and $\pi_{i,0} = \{H_i\}$ is the policy of DU_i given no request is sent at time t_o.

VI. PROXY ASSISTED SMART CACHING

All the previous discussions are based on the assumption that there is no cache available in the proxy. However, we notice that a wireless channel is much more error-prone compared with a wire-line channel due to high bit error rate, contention, and end host mobility. Local caching and retransmission can be performed to improve wireless channel throughput by concealing wireless losses from end users. Therefore, in this section, we revisit the problem by considering the role of caching at the proxy.

Since the cache size is finite, contention will take place when a packet arrives at a full cache. To solve the cache contention problem, a comprehensive cache management is needed. In contrast to our previous work [25], here we consider two different caching strategies: Simple Caching Simple Fetching (SCSF) and Distortion Minimized Smart Caching (DMSC).

- **Simple Caching Simple Fetching (SCSF)**: The basic idea of SCSF is that the proxy saves a copy of incoming data before forwarding it to the client, but only if its cache is not full. Otherwise, it will directly forward the incoming data to the client. When the data unit to be retransmitted is in the cache, the proxy will fetch it from its cache instead of requesting it from the joint sender group (i.e., local retransmission).
- **Distortion Minimized Smart Caching (DMSC)**: Instead of simply forwarding packets to the client when the cache is full, we propose a more intelligent caching strategy so that more important data units will have a higher chance to be cached. The basic idea of DMSC is to use the distortion reduction contributed by successful delivery of DU_i to denote its importance and cache data that will maximize the distortion reduction at the client under a certain cache size constraint. By implementing DMSC, there is no packet forwarding anymore. Instead, packets arriving in the proxy will be cached/dropped based on their importance, and the proxy cache will operate as a secondary sender, which can actively send out DUs to the client under the regulation of its associated asynchronous clock.

The introduction of cache into the proxy will affect how to associate the optimization instances with the awakening behavior of the asynchronous clocks, and as a result, affect the way we model the whole system. In the following discussion, we will present the detailed DMSC cache management mechanism and model the performance of a cache-enabled MUVIS system.

A. Transmission token reinterpretation

When the proxy implements SCSF, it still has to ensure that an end-to-end connection between the sender j and the client exists before requesting new data from sender j. Therefore, the transmission token will be similarly treated to the case where the proxy has no cache. In contrast, when the proxy implements DMSC, the end-to-end connection between sender j and the client is not critical. We do not need to collect transmission tokens on both legs. Hence, if a first leg clock j wakes up, the proxy will immediately request a data unit from sender j regardless of the second segment. The requested data units will be cached once they get to the proxy. If the second leg clock wakes up, the proxy will immediately select a data unit from the cache to send to the client.

B. Performance modeling for cache-enabled MUVIS system

B.1 Case one: SCSF is implemented

When the proxy uses SCSF strategy, we still need to collect tokens for both legs before sending a request to sender j. Since the cache size is limited, some data units will be cached while others will be forwarded directly to the client. For a data unit DU_i that has not been cached, the probability $q_i(\pi_i)$ that DU_i will arrive at the client before its deadline T_i can be defined using (7). If the data unit DU_i under (re)transmission consideration happens to be cached, the proxy will send DU_i directly from its cache to the client. Suppose that DU_i is successfully cached by the proxy at the $l_i'^{th}$ attempt ($l_i' \leq l_i$). Since DU_i has already arrived in the cache successfully, the transmission history for the past l_i attempts can be discarded. Thus, the probability of successfully receiving cached DU_i at the client by its deadline T_i, $q_i(\pi_i)$, is:

$$q_i(\pi_i) = 1 - \left(1 - P_{AC}^F(T_i - t_o)\right) \prod_{k=l_i'+1}^{l_i} \left(1 - P_{AC}^F(T_i - t_i^{(k)})\right) \quad (10)$$

where $\prod_{k=l_i'+1}^{l_i} \left(1 - P_{CA}^F(T_i - t_i^{(k)})\right) = 1$ if $l_i' + 1 > l_i$.

B.2 Case Two: DMSC is implemented

When DMSC is implemented, the optimization instance means that the proxy can request a data unit from sender j or it can stream a data unit from its cache to the client, depending on whether the awakened clock is in the first leg or the second. If it is in the first leg, the proxy will request transmission from sender j. The optimization window will contain data units that have not yet arrived at the proxy correctly. For a data unit DU_i that that has not yet correctly arrived at the cache, π_i will be the transmission request policy from proxy to senders. We can define the probability that DU_i will arrive at the proxy before its deadline as:

$$q_i(\pi_i) = 1 - (1 - P_{Ac_i}(T_i - t_o)) \prod_{k=1}^{l_i} \left(1 - P_{As_i^{(k)}}(T_i - t_i^{(k)})\right) \quad (11)$$

If it is in the second leg, a data unit transmission will be scheduled from the proxy to the client. The optimization window will then only contain data units that are already available in the cache but not available in the client yet. For a data unit DU_i in the cache, π_i will be the transmission policy from the proxy to the client. We can define the probability that DU_i will arrive at the client before its deadline as:

$$q_i(\pi_i) = 1 - \left(1 - p_{AC}^F(T_i - t_o)\right) \prod_{k=1}^{l_i} \left(1 - p_{AC}^F(T_i - t_i^{(k)})\right) \quad (12)$$

Having q_i defined by (7), (10)-(12), now we can deduce the expected distortion for N data units using (5) for both of the above cases. Note that for the first case and for proxy-client pair in the second case, $D(\pi)$ corresponds to the expected distortion at the *client*, while for the sender-proxy pair j in the second case, $D(\pi)$ corresponds to the expected distortion at the *proxy*.

C. Cache management

Cache management for a DMSC includes three parts: *cache write*, *cache read*, and *cache flush*. *Cache write* refers to the strategy for writing incoming data units to the cache. When the cache is not full, all incoming data units will be cached using the First-Come-First-Cache (FCFC) criteria. Since data units may be delayed differently before they arrive at the proxy, packet reordering is needed to guarantee that data units with shorter play-out deadlines will be saved further ahead in the cache so that they have better chances to be selected earlier for transmission. When the cache is full, the proxy has to rely on its flushing strategy to free up some space before writing an incoming data unit into the cache.

Cache read refers to the strategy for selecting a data unit to transmit to the client. In each optimization instance t_o, the proxy will pick up to N data units from the head of the cache. For

each data unit DU_i we define two parameters, last transmission time T_l^i and retransmission timeout T_O. T_l^i is the latest time when DU_i was (re)transmitted. T_O denotes the amount of time that will elapse before DU_i will be re-selected for retransmission consideration. If DU_i has been (re)transmitted in the past T_O seconds, i.e. $t_o - T_l^i \leq T_O$, it will not be considered for (re)transmission at time t_o. Once we select data units under (re)transmission consideration at time t_o, the remaining part will be equivalent to the point-to-point RD optimized streaming problem.

Cache flush regulates when and how to flush the cache. The cache will be flushed when one of the following situations happens: i) an acknowledge is received from client; ii) a cached data unit expires; and, iii) a data unit arrives at the fully occupied cache. In the first two cases, the acknowledged or expired data units will be deleted from the cache immediately. In the third case, we need to decide the relative importance of data units in the cache and the incoming data unit. Let t be the time when the contention happens. At time t, the relative importance I_i of DU_i is measured by λ_i and its time-to-live (TTL_i):

$$I_i = \min\{\lambda_i \cdot \left\lfloor \frac{TTL_i}{t_o - t + \mu_P} \right\rfloor, \lambda_i\}, \tag{13}$$

where, TTL_i is given by $TTL_i = T_i - t$, t_o is the first transmission opportunity appearing after t, μ_P is the measured mean round trip time for the connection between the proxy and the client, and λ_i is the benefit of transmitting DU_i defined in Section V. The smaller I_i, the less important DU_i will be. (13) shows that DU_i that cannot arrive at the client by its play-out deadline T_i will be the least important. With the relative importance defined by (13), if the incoming data unit is more important than some data unit in the cache, we need to flush the least important data unit from the cache to make room for the incoming data unit. If the incoming data unit is less important than any data unit in the cache, the proxy will choose to drop the incoming data unit.

VII. PERFORMANCE EVALUATION

To evaluate the proposed MUVIS system, we use the network simulator NS-2.27 to implement our schemes. We characterize the received video quality using the Peak Signal-to-Noise Ratio (PSNR). In the following discussion, we will first describe how to simulate the wireless environment and video source in Section VII-A, and then present the simulation results in Section VII-B.

A. Simulation design

We simulate a hybrid networking environment that consists of two parts: wireless LAN and wired Internet. The topology includes one server, one AP and three MNs (one client and two peers). A proxy node is co-located with the AP and communicates with all mobile nodes at 2.437GHz using IEEE 802.11 infrastructure mode. Meanwhile, the server is connected to the proxy/AP over a wired connection.

We will use the empirical results of Balazinska et al. [26] to generate more realistic mobility patterns instead of using the random way-point model. To model the mobility of an individual mobile user, we need to compute two metrics: session duration and revisit interval.

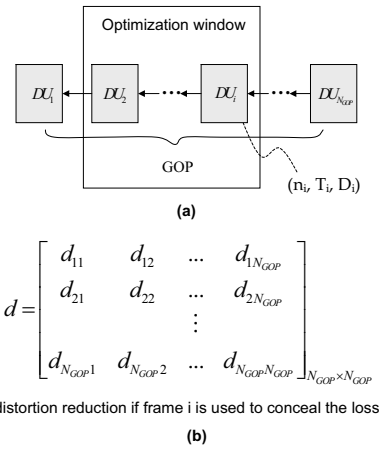


Fig. 5. Packet interdependency: (a) Linear directed acyclic graph(DAG), (b) Distortion matrix

• **Session Duration**: For MN j, we denote the session duration as R_j. Our definition of session duration is equivalent to the persistence metric in the work of Balazinska et al., which is shown to follow a power law distribution with an exponent of 1.78, i.e., $P(X = x) \sim x^{-1.78}$.

• **Revisit Interval**: After a MN leaves the AP, it is likely that it will come back and visit the AP again. The revisit interval defined in Section III-C shows how likely it is for a MN to visit an AP that it has visited recently. Balazinska *et al.* [26] presented a prevalence metric to measure the fraction of time that a user spends with a given AP. In this paper, we will derive the revisit interval based on that prevalence distribution.

Let V_j be the revisit interval. The prevalence of MN j is $Prev_j$ and the prevalence probability distribution follows a power law [26], i.e. $P(Prev_j = x) \sim 0.001x^{-1.75}$. Using $Prev_j$ and R_j, the revisit interval V_j is defined by $V_j = R_j/Prev_j$.

For the wire-line connection, we set both forward and backward packet loss rates (PLR) to be 5%, and the round trip time (RTT) to be 100ms. The bandwidth of the link between the server and the proxy node is set to be 1.1Mbps. We add some background traffic to simulate a more realistic networking environment. Here, we use constant bit rate (CBR) traffic as the cross traffic. Therefore, in the presence of background traffic, the allocated bandwidth for streaming service between the server and the proxy is at most 150 Kbps. For the wireless network set-up, we enable RTS/CTS and implement DCF. We choose the wireless bit error rate (BER) to be $1.5 * 10^{-5}$. The link layer (re)transmission delay is set to be 50μs and the maximum number of link layer retransmissions is 4. The data rate of the wireless channel is 11Mbps, and only up to 1.1 Mbps out 11Mbps will be used for streaming service. For each MU, the maximum allocated bandwidth for streaming service will be 450 Kbps for the proxy-to-client connection and 150 Kbps for all peer-to-proxy connections.

Two 300-frame Class-B[1] standard video sequences, *foreman* and *container*, are used to drive the simulation. These video

[1]Class B sequences normally have medium spatial detail and low amount of movement or *vice versa*.

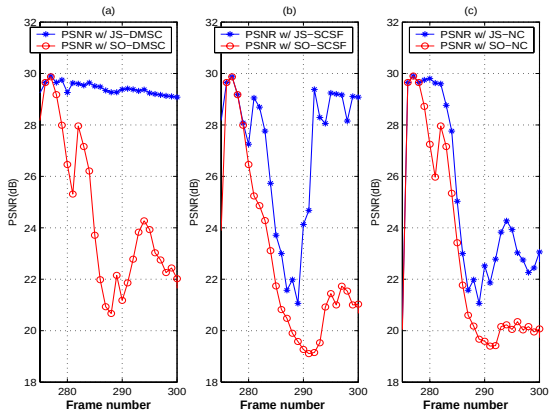


Fig. 6. Effect of multi-source streaming ('*Foreman*')

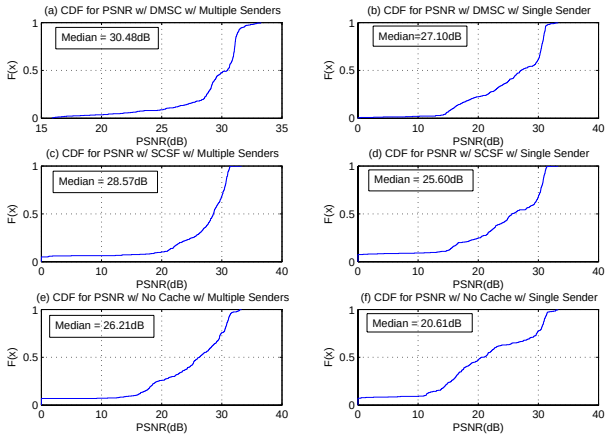


Fig. 7. Cumulative Distribution Function (CDF) of across-run average PSNR ('*Foreman*')

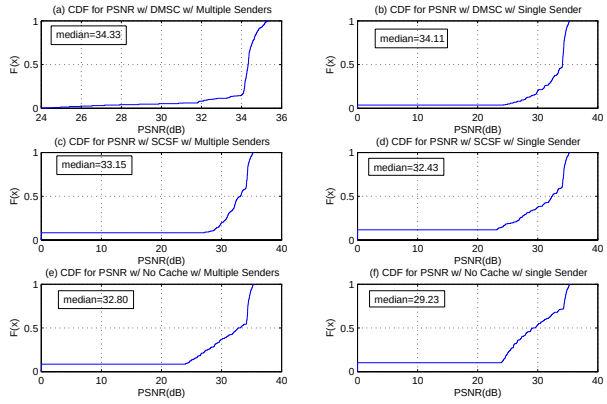


Fig. 8. Cumulative Distribution Function (CDF) of across-run average PSNR ('*Container*')

sequences are encoded at 120kps as shown in Figure 5(a) using H.263 version2 at QCIF, with 30 frames per second and a 1/25 I-frame frequency. For each sequence, the PSNR between the original frame i and the reconstructed frame j is calculated for every combinational of i and j for $i \leq j$ and saved as a matrix d illustrated by Figure 5(b). Considering the interdependency shown in Figure 5(a), the distortion reduction D_i for DU_i can be expressed as:

$$D_i = \begin{cases} d(i,i) + \sum_{j=2}^{N_{GOP}} d(i,j), & (i \bmod N_{GOP}) = 1 \\ d(i,i) + \sum_{j=i+1}^{N_{GOP}} d(i,j) - \sum_{j=i}^{N_{GOP}} d(i-1,j), & \text{otherwise} \end{cases}$$

where $d(i,j)$ is the distortion reduction if frame i is used to conceal the loss of frame j, and N_{GOP} is the number of frames in one Group of Pictures (GOP). D_i will then be used to design the scheduling scheme and estimate the performance of the client based on successfully received data units.

B. Simulation Results

To show the benefits of multi-path rate-distortion optimized streaming and to compare different caching strategies, we have implemented the following streaming schemes: Joint sender group with DMSC ***(JS-DMSC)***, Joint sender group with SCSF ***(JS-SCSF)***, Joint sender group with No Cache ***(JS-NC)***, Server-only with DMSC ***(SO-DMSC)***, Server-only with SCSF ***(SO-SCSF)***, and Server-only with No Cache ***(SO-NC)***. The first three schemes, ***JS-DMSC***, ***JS-SCSF***, ***JS-NC***, use the proposed MUVIS scheme but different caching strategies. The other three schemes, ***SO-DMSC***, ***SO-SCSF***, ***SO-NC***, do not use MUVIS and only rely on the media server to provide video contents.

B.1 Multi-Source Diversity

We test the different streaming schemes using both the *Foreman* and *Container* sequences and measure the effect of multi-source diversity on the system performance for *No Cache* , *SCSF* and *DMSC* scenarios, respectively. Figure 6 compares the performance of the multi-source streaming and single-source streaming in one typical simulation run. In this set of simulations, we consider transmitting *Foreman* over the simulated wired/wireless network, and for this specific run, we can see that using multi-source streaming improves the quality of received video at the client.

We repeat the above simulations 50 times with different random seeds. For each streaming scheme, we plot the cumulative distribution function (CDF) of the across-run average PSNR (computed across multiple runs for each frame) shown in Figure 7 and Figure 8. In each figure, we compare the CDF of six competing streaming schemes and show the corresponding median values. From these two figures, we can see that those schemes using multi-source diversity have higher median values, which means more satisfying video quality 50% of the time. We can compute the mean PSNR for each scheme by further averaging the across-run average PSNR in Figure 7 and Figure 8 over the entire sequence run-time, respectively. Table 3 and Table 4 show the mean PSNR of both video sequences for these six competing strategies in the lossy scenario. As a comparison, we show the mean PSNR in the loss-free scenario as well. From Table 3 and Table 4, we can see that for both sequences, multi-source streaming schemes achieve better perfor-

PSNR w/o Loss (dB)	PSNR w/ Loss (dB)		
30.71	w/ JS-DMSC	w/ JS-SCSF	w/ JS-NC
	29.16	25.56	24.45
	w/ SO-DMSC	w/ SO-SCSF	w/ SO-NC
	27.01	23.23	21.44

Table 3. Average PSNR ('*Foreman*')

PSNR w/o Loss (dB)	PSNR w/ Loss (dB)		
34.32	w/ JS-DMSC	w/ JS-SCSF	w/ JS-NC
	32.40	30.45	29.77
	w/ SO-DMSC	w/ SO-SCSF	w/ SO-NC
	31.95	29.86	27.13

Table 4. Average PSNR ('*Container*')

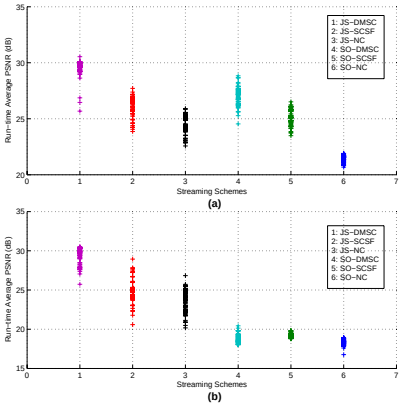


Fig. 9. PSNR variation across 50 runs ('*Foreman*')

PSNR w/o Loss (dB)	PSNR w/ Loss (dB)		
30.71	w/ JS-DMSC	w/ JS-SCSF	w/ JS-NC
	28.30	24.88	23.45
	w/ SO-DMSC	w/ SO-SCSF	w/ SO-NC
	18.84	19.18	18.31

Table 5. Average PSNR when the server-client connection is the bottleneck ('*Foreman*')

mance than schemes that only use the media server by effectively leveraging the content available at peers. Such multi-source streaming can offer additional advantages. For example, when the remote media server is overloaded or when there is congestion on the path to the server, the client can avoid quality degradation by requesting data units from its neighboring peers. Second, peer-to-peer streaming with neighboring nodes may incur less delays. To illustrate this, we change the available bandwidth between the server and the proxy for the streaming service to be 0.1 Mbps, representing a scenario where the path to the server is congested. We use the *Foreman* sequence and rerun the simulation using this new setting 50 times with different random seeds to get the mean PSNR. We show the quality variation using six different schemes in Figure 9. As a comparison, we use Figure 9(a) to show the quality variation for the case when server-client connection is not the bottleneck, while Figure 9(b) shows the case when server-client connection is the bottleneck. To obtain Figure 9, we average the PSNR over the sequence

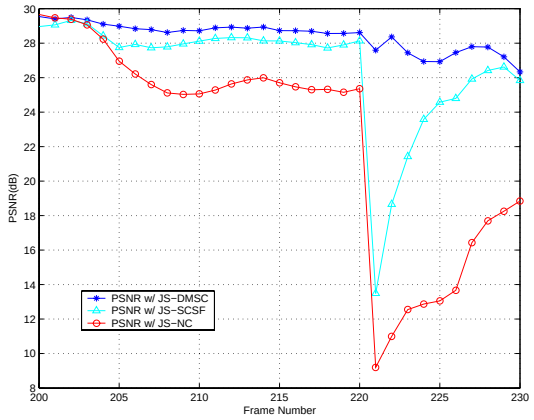


Fig. 10. Effect of different caching strategies on the system performance when multi-source streaming is supported ('*Foreman*')

run-time for each scheme. In Figure 9, the y-axis shows the sequence run-time average PSNR for each of those the 50 runs while the x-axis shows the corresponding streaming schemes. By further averaging the results in Figure 9 over the 50 runs, we can get the average PSNR for each scheme as shown in Table 5. Both Figure 9 and Table 5 clearly demonstrate that at the onset of the appearance of congestion, streaming schemes that use multi-source diversity can maintain good performance by leveraging multiple connections, while those that only rely on the server connection suffer great performance degradation.

B.2 Cache Strategies

In Section VI, we discussed how the introduction of cache in the proxy affects performance modeling. To measure the cache effect and compare different cache strategies, *SCSF* vs. *DMSC*, we evaluate the streaming performance in both the single-source and multiple-source scenarios. In our simulations, we measure the cache size in terms of the number of RTP packets, which is set to be 100 RTP packets throughout the experiments. We first study how the introduction of cache affects the streaming quality. To do this, we simulate three schemes: JS-DMSC, JS-SCSF and JS-NC. We repeat the same experiments 40 times each, and average the results over 40 runs to get the across-run average PSNR perceived by the client. In Figure 10, we observe that using limited cache also helps improve the performance for the multi-source streaming scheme. With the same cache capacity, the proposed $DMSC$ strategy can provide better system performance compared to the $SCSF$ strategy. Table 3 and Table 4 presented in Section VII-B.1 also show that using limited cache can improve the average streaming performance in both single and multi-source scenarios.

The last set of results show the average PSNR under different loss conditions. Figure 11(a) shows how different schemes perform under fixed wireless BER ($1.5 * 10^{(-5)}$) but varying wired PLR, while Figure 11(b) shows how they perform under fixed wired PLR(5%) but varying wireless BER. From Figure 11, we can see that most of time multi-source streaming can provide more satisfying performance than the server-only approach by leveraging peer connections to compensate the quality degradation caused by the increased loss. The one exception, shown in

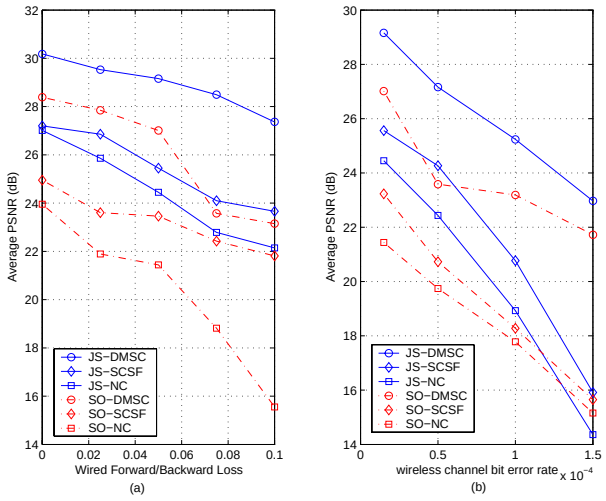


Fig. 11. Average PSNR vs. Varying Loss Conditions ('*Foreman*')

Figure 11(b), is that *SO-NC* performs better than *JS-NC* when the wireless BER is really high. This is because wireless connection is much worse than the wired connection, and without caching, the increased contention from having peer connections will cancel out the benefit of multi-source diversity and degrade the quality instead. In addition, Figure 11 also clearly shows how different caching strategies work under varying loss environments. As illustrated, *DMSC*-based schemes perform better than *SCSF*-based schemes and the schemes with no cache, especially when DMSC is jointly used with multi-source streaming.

VIII. CONCLUSION AND FUTURE WORK

In this paper, we propose the multi-source video streaming (MUVIS) system to support high-quality video streaming service over WLANs. MUVIS uses a proxy collocated with the access point to leverage the media server and mobile peers as a joint sender group (multi-source) and stream video to the client in a rate-distortion optimized way. We formulate such a multi-source, rate-distortion optimized streaming process as a combinatorial packet scheduling problem and combine the concept of asynchronous clocks and the rate-distortion optimization framework to solve it. To investigate the cache effect, we present two different caching strategies, the traditional *Simple Caching Simple Fetching (SCSF)* and our proposed *Distortion Minimized Smart Caching (DMSC)*, and model the overall system performance. The measurement results from corporate wireless networks are used to generate more realistic peer mobility models for the simulations. Numerical investigations through NS simulations show that schemes with multi-source diversity can provide better performance than those with a single sender. The simulation results also show that caching at the proxy will potentially increase the streaming performance. Compared with the simple caching strategy *SCSF*, our proposed *DMSC* strategy can provide better performance. It also achieves higher resilience to packet loss, especially when it is jointly used with multi-source streaming. In this paper, we focus on the performance of a single wireless client. In the future, we will address the performance of multiple wireless clients running multiple co-existing MUVIS sessions. In addition, we will also investigate admission control and rate allocation for multiple streaming sessions over WLANs.

ACKNOWLEDGMENTS

This work was supported in part by the National Science Foundation (NSF) under the CAREER Award No. 0238348, NSF ITR EIA-0122599 and UC Micro program with matching funds from Hewlett Packard.

REFERENCES

[1] S. Krishnamachari, M. van der Schaar, S. Choi, and X. Xu, "Video streaming over wireless LANs: A cross-layer approach," in *Proceedings of Packet Video Workshop*, Nantes, France, April 2003.

[2] A. Majumdar, D. G. Sachs, I. V. Kozintsev, K. Ramchandran, and M. M. Yeung, "Multicast and unicast real-time video streaming over wireless LANs," in *IEEE Transactions on Circuits and Systems for Video Technology*, June 2002, vol. 12, no.6.

[3] T. Wang, H. Fang, and L. Chen, "Low-delay and error-robust wireless video transmission for video communications," in *IEEE Transactions on Circuits and Systems for Video Technology*, Dec. 2002, vol. 12, no.12.

[4] P. A. Chou and Z. Miao, "Rate-distortion optimized streaming of packetized media," Tech. Rep. MSR-TR-2001-35, Microsoft Research, Feb. 2001.

[5] X. Xu, Y. Wang, S. P. Panwar, and K. W. Ross, "A peer-to-peer video-on-demand system using multiple description coding and server diversity," in *IEEE ICIP*, Singapore, Oct. 2004.

[6] T. Nguyen and A. Zakhor, "Protocols for distributed video streaming," in *IEEE ICIP*, Rochester, NY, USA, Sept. 2002.

[7] M. Hefeeda, A. Habib, B. Botev, D. Xu, and B. Bhargava, "Promise: Peer-to-peer media streaming using collectcast," in *ACM International Conference on Multimedia*, Berkeley, California, USA, Nov. 2003.

[8] X. Zhang, J. Liu, B. Li, and T.-S. P. Yum, "Donet/coolstreaming: A data-driven overlay network for live media streaming," in *IEEE INFOCOM*, Miami, FL, USA, 2005.

[9] A. Dan and D. Sitaram, "Buffer management policy for an on-demand video server," Tech. Rep. 19347, IBM Research, Jan. 1993.

[10] A. Dan and D. Sitaram, "A generalized interval caching policy for mixed interactive and long video environment," in *Proceeding of Multimedia Computing and Networking Conference*, San Jose, CA, USA, Jan. 1996.

[11] R. Tewari, H. Vin, A. Dan, and D. Sitaram, "Caching in bandwidth and space constrained hierarchical hyper-servers," Tech. Rep. CS-TR-96-30, Department of Computer Sciences, University of Texas at Austin, Jan. 1997.

[12] J. M. Almeida, D. L. Eager, and M. K. Vernon, "A hybrid caching strategy for streaming media files," in *Proceeding of Multimedia Computing and Networking*, San Jose, CA, Jan. 2001.

[13] R. Rejaie, H. Yu, M. Handley, and D. Estrin, "Multimedia proxy caching mechanism for quality adaptive streaming applications in the internet," in *IEEE INFOCOM*, Tel Aviv, Israel, March 2000.

[14] S. Sen, J. Rexford, and D. Towsley, "Proxy prefix caching multimedia streams," in *IEEE INFOCOM*, New York City, USA, March 1999.

[15] Y. Wang, Z. L. Zhang, D. Du, and D. Su, "A network conscious approach to end-to-end video delivery over wide area networks using proxy servers," in *IEEE INFOCOM*, New York City, USA, April 1999.

[16] Z. Miao and A. Ortega, "Proxy caching for efficient video services over the internet," in *Proceeding of Packet Video Workshop*, New York City, USA, April 1999.

[17] D. Li, G. Cheung, C. Chuah, and S. J. Ben Yoo, "Joint server/peer receiver-driven rate-distortion optimized video streaming using asynchronous clocks," in *IEEE ICIP*, Singapore, Oct. 2004.

[18] J. Chakareski, P. A. Chou, and B. Girod, "Computing rate-distortion optimized policies for hybrid receiver/sender driven streaming of multimedia," in *Asilomar Conference on Signals, Systems, and Computers*, Pacific Grove, CA, USA, Nov. 2002.

[19] J. Chakareski and B. Girod, "Server diversity in rate-distortion optimized media streaming," in *IEEE ICIP*, Barcelona, Spain, Sept. 2003.

[20] A. C. Begen, Y. Altunbasak, and M. A. Begen, "Rate-distortion optimized on-demand media streaming with server diversity," in *IEEE ICIP*, Barcelona, Spain, Sept. 2003.

[21] G. Cheung, C. Chuah, and D. Li, "Optimizing Video Streaming Against Transient Failures and Routing Instability," in *IEEE ICC*, Paris, France, June 2004.

[22] A. Gurtov and S. Floyd, "Modeling wireless links for transport protocols," in *ACM SIGCOMM Computer Communication Review*, April 2003, pp. 85–96.

[23] S. Floyd, M. Handley, J. Padhye, and J. Widmer, "Equation-based congestion control for unicast applications," in *ACM SIGCOMM*, Stockholm, Sweden, August 2000.

[24] T. Yoshimura, T. Ohya, T. Kawahara, and Minoru Etoh, "Rate and robustness control with rtp monitoring agent for mobile multimedia streaming," in *IEEE ICC*, New York City, USA, April 2002.

[25] D. Li, G. Cheung, C. Chuah, and S. J. Ben Yoo, "Proxy-driven rate-distortion optimized video streaming over wireless network using asynchronous clocks," in *Proceeding of Packet Video Workshop*, Irvine, CA, Dec. 2004.

[26] M. Balazinska and P. Castro, "Characterizing mobility and network usage in a corporate wirelss local-area network," in *First International Conference on Mobile Systems, Applications, and Services (MobiSys)*, San Francisco, CA, USA, May 2003.