Energy-Aware Multi-Source Video Streaming

Danjue Li* Chen-Nee Chuah* Gene Cheung† S. J. Ben Yoo *
*University of California, Davis †HP Laboratories, Japan

***Abstract*— In a multi-source video streaming system, premature draining of low-power nodes can cause sudden failures of peer connections and degrade streaming performance. To solve this problem, we propose an energy-aware scheduling (EAS) scheme to better distribute the streaming load among different peers by jointly considering network conditions and node energy levels. We model the proposed scheme using a rate/energy-distortion optimization framework and heuristically solve it using the concept of asynchronous clocks. Simulation studies show that the proposed EAS scheme can achieve comparable streaming quality while consuming less energy.**

I. INTRODUCTION

This paper considers a multi-source video streaming (MUVIS) system [1] that leverages source diversity to enhance the wireless video streaming quality. As illustrated in Figure 1, MUVIS uses proxy located next to the access point to periodically request data units from participating peers and server, and subsequently forward them to the streaming client. To perform sender selection, for each sender the proxy needs to set up a clock, which wakes up at regular intervals of Δ_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. In such a system, most peers operate on limited battery supplies while streaming video usually tends to be a lengthy process that can consume a significant amount of energy. Considering that the client streams different portions of video content from different senders, unexpected power depletion in some peers can cause peer failure and greatly degrade the received video quality. Therefore, it is important to design an energy efficient scheduling scheme to avoid completely draining low-power nodes while still providing satisfying video quality. Here, we are mainly concerned with the energy consumed by the wireless network interface cards (WNIC), which is determined by how much information is sent or received and how long the WNIC is activated.

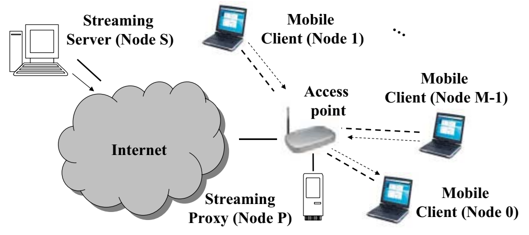

Fig. 1. Multi-Source Video Streaming over WLAN

Several studies address the energy problem at various protocol layers [2–4]. Ramos et al. [2] proposed a link layer adaptation scheme to dynamically adapt the fragmentation threshold, transmission power, and retry limit according to the channel conditions. Chandra et al. [3] explored ways to transmit data packets in a predictable fashion, allowing the clients to transition the WNIC to a low power sleep state. Tamai et al. [4] proposed an energy-aware video streaming system based on battery capacity, desired playback duration, and relative importance of video segments. Most of these works focus on minimizing energy consumption on a single wireless station (STA) and do not consider the multi-source scenario. This paper aims to design an energy-aware scheduling scheme for multiple source video streaming over WLAN.

Our work makes the following contributions toward this goal. First, we design an *asynchronous-clock-based* [1] power saving strategy that puts the WNIC in a low power state [3] whenever possible. By using asynchronous clocks to make different senders transmit data packets at regular predictable intervals, our approach can provide better accuracy in predicting packet arrivals. Second, we model energy consumption in each source node transmission cycle, and formulate the energy-aware scheduling problem as a rate/energy-distortion optimization problem. Third, we present NS [5] simulations to show how the proposed scheduling scheme can help reduce total energy cost while maintaining satisfying video quality.

II. ENERGY SAVING IN MULTI-SOURCE VIDEO STREAMING SYSTEM

Figure 2(a) illustrates the phases in one transmission cycle of sender $j \in \{1, ..., M-1\}$ without any energy-saving strategies. Here a transmission cycle refers to the time period between two adjacent request arrivals on sender j. The various tasks performed by the MAC protocol in one transmission cycle will correspond to different radio modes, namely, *receive*, *transmit*, *idle*, or *sleep* modes. Although power consumption for each mode might vary for different WNICs, all of them share the characteristic property that the $transmit$, $receive$, and $idle$ modes always consume a similarly large amount of power while $sleep$ mode consumes the least amount of power [6]. Ignoring the energy consumed for channel sensing, the total energy consumed by sender j during one transmission cycle will include: the energy spent to receive request and acknowledgement, e_j^r, the energy spent transmitting requested data packets, e_j^t, and the energy drainage during the idle mode, e_j^i, i.e. $e_j = e_j^t + e_j^r + e_j^i$.

To compute e_j^t, e_j^r and e_j^i, we first define the following variables. Let ϵ_j be the bit error rate perceived by sender j, R be the physical layer data transmission rate of the WNIC, b_r be the request packet size, b_a be the acknowledgment packet size, b be the average RTP packet size, n be the number of RTP

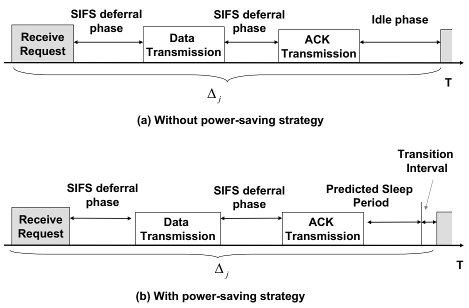

Fig. 2. A transmission cycle of a selected sender j

packets requested each time from sender j, and T_{SIFS} be the length of Short Interframe Space(SIFS). Let P_t, P_r, P_i, and P_s be the power of the WNIC under *transmit*, *receive*, *idle*, and *sleep* modes, respectively. We define Δ_j as the period to request packets from sender j, and it is bounded by $\Delta_j \geq \frac{b}{R_{jP}}$, where R_{jP} is the maximum streaming rate allowed by current network conditions between sender j and the proxy. Therefore, we can compute e_j^t, e_j^r and e_j^i as:

$$e_j^t = \frac{n \cdot b}{R} \cdot \frac{1}{(1-\epsilon_j)^b} \cdot P_t \tag{1}$$

$$e_j^r = \frac{b_r + n \cdot b_a}{R} \cdot P_r \tag{2}$$

$$e_j^i = (\Delta_j - \frac{\frac{nb}{R}}{(1-\epsilon_j)^b} - \frac{n \cdot b_a + b_r}{R}) \cdot P_i \tag{3}$$

From Figure 2(a), we find that a WNIC stays *idle* for most of its transmission cycle, which contributes to a significant part of total energy consumption. If a power-saving mechanism is supported and the WNIC can switch among different communication modes for arbitrary durations of time, transitioning to low-power mode (i.e. *sleep* mode) when the WNIC is not sending/receiving data can result in large energy savings. Choosing *sleep* time, T_j^s, for sender j is a trade-off between saving power and increasing the packet loss rate: with a smaller T_j^s, the WINC spends lots of time in idle state, potentially missing on further energy saving; on the contrary, with a larger T_j^s, the WINC might be asleep while a packet was being delivered, potentially missing a packet. Note that in the MUVIS system, a selected sender j only receives requests and sends data when its associated asynchronous clock wakes up. By taking advantage of the asynchronous clock settings, we propose a power-conserving strategy as illustrated by Figure 2(b). When an asynchronous clock wakes up, the proxy selects a sender j from the joint sender group and generates a request with the clock period information, Δ_j piggybacked. Upon receiving a new request, the sender strips off the clock period information. Since packets might experience different delays before arriving in sender j, we use a sliding window with size equal to W requests to smooth out the effect of delay jitter and improve the estimation accuracy of T_j^s:

$$T_j^s = (\Delta_j - \frac{\frac{n \cdot b}{R}}{(1-\epsilon_j)^b} - 2n \cdot T_{SIFS} - \frac{n \cdot b_a + b_r}{R} - T_j^t + \frac{\sum_{k=2}^{W} |I_j^{(k)} - I_j^{(k-1)}|}{W-1} \cdot \frac{I_j^{(W)} - I_j^{(W-1)}}{|I_j^{(W)} - I_j^{(W-1)}|}) \cdot \alpha \tag{4}$$

where $I_j^{(k)}$ is the packet arrival time at sender j, and T_j^t is the WNIC waking-up time. α is *estimation adjust factor*, which scales from 0 to 1, as a measure of how conservative we choose the sleep period. A smaller *estimator adjust factor* means a more conservative estimation. Since the WNIC will be placed into *sleep* mode for T_j^s instead of staying *idle*, the total energy consumption e_j will be adjusted:

$$e_j = e_j^t + e_j^r + e_j^i - T_j^s(P_i - P_s) \tag{5}$$

III. Energy-aware Scheduling

In this section, we will present our proposed energy-aware scheduling scheme used by the proxy to decide when and to which peer/server it should send each data unit request. Since streaming is a time-constraint application, the scheduling scheme must be computationally efficient.

A. Problem Formulation

We first model the source using a directed acyclic graph (DAG) [7] and represent each frame i using a data unit DU_i associated with three constants: the size of the DU_i in RTP packets n_i, the playout buffer deadline T_i, and the distortion reduction D_i. Using the source model, we can formulate the energy-aware scheduling problem as a joint rate/energy-distortion optimization problem, i.e. under certain rate and energy constraints, we want to schedule packet transmissions to achieve optimal distortion on the client side. As in our previous work [1], we assume that the optimization will be performed every T_o seconds, and at each optimization instant t, there are N data units under consideration for (re)transmission. For each DU_i in the optimization window, we define a transmission policy, $\pi_i = \{h_i, c_i\}$, which records its transmission history h_i before t and transmission decision c_i for next T_o seconds. h_i is defined as: $h_i = \{(t_i^{(1)}, s_i^{(1)}), (t_i^{(2)}, s_i^{(2)}), ..., (t_i^{(l_i)}, s_i^{(l_i)})\}$, where l_i is the number of attempts, $t_i^{(k)}$ and $s_i^{(k)}$ are the time stamp and the sender id for k^{th} attempt, respectively. c_i is defined as: $c_i = \{(\tau_i^{(1)}, \xi_i^{(1)}), (\tau_i^{(2)}, \xi_i^{(2)}), ..., (\tau_i^{(z_i)}, \xi_i^{(z_i)})\}$, where z_i is the number of transmission requests to be sent in next T_o seconds. Thus, we can get the probability that DU_i can be received correctly at the client before its deadline as:

$$q_i(\pi_i) = 1 - \prod_{k=1}^{l_i} [1 - p_{s_i^{(k)}}(T_i - t_i^k)] \cdot \prod_{k=1}^{z_i} [1 - p_{\xi_i^{(k)}}(T_i - \tau_i^{(k)})] \tag{6}$$

where $p_{s_i^{(k)}}(T_i - t_i^k)$ is the probability that a request sent by the proxy at time t_i^k can result in a data unit DU_i from sender $s_i^{(k)}$ arrival at the client ($j = 0$) before its playout deadline T_i, which can be computed via the i.i.d loss model and delay model in [1]. $p_{\xi_i^{(k)}}(T_i - \tau_i^k)$ is similarly defined and computed. Given the source model, (5), and (6), the energy-aware scheduling problem can be formulated as: minimizing the expected overall distortion for N data units, i.e.

$$D(\pi) = D_0 - \sum_{i=1}^{N} D_i \prod_{l \preceq i} q_l(\pi_l) \tag{7}$$

subject to:

$$\sum_{i=1}^{N}\sum_{k=1}^{z_i} n_i\delta(\xi_i^k - j) \le \left\lfloor \frac{T_o \cdot R_{jP}}{b} \right\rfloor, j = S, 1, 2, ..., M-1 \quad (8)$$

$$\sum_{j=1}^{M-1}\sum_{i=1}^{N}\sum_{k=1}^{z_i} n_i\delta(\xi_i^k - j) + \sum_{i=1}^{N}\sum_{k=1}^{z_i} n_i\delta(\xi_i^k - S) \le \left\lfloor \frac{R_{0P} \cdot T_o}{b} \right\rfloor \quad (9)$$

$$\sum_{i=1}^{N}\sum_{k=1}^{z_i} e_j(n_i)\delta(\xi_i^k - j) \le E_j, j = 1, 2, ..., M-1 \quad (10)$$

where $\delta(x)$ is the delta function: $\delta(x) = 1$ if $x = 0$ and 0 otherwise. R_{0P} is the maximum streaming rate between the proxy and the client. E_j is the total amount of energy available on sender j, which will be updated at each optimization instant. If sender j is selected at attempt k of DU_i, i.e. $\xi_i^{(k)} = j$, then the requesting time must be one of the transmission opportunities of pair j.

B. Solution

Directly solving the optimization problem defined by (7), (8), (9), and (10) is computationally prohibitive. Instead, we rely on asynchronous clocks [1] to decouple the combinatorial problem into a set of point-to-point rate/energy-distortion optimization problem, i.e. selecting a sender and then picking a data unit to request from that sender. Here, the period of each clock will be jointly decided by the quality of the link between sender j and the proxy, represented by R_{jP}, and the energy level of sender j, E_j. To implement the idea of asynchronous clocks, the proxy needs to track energy level of each sender based on its updated energy information, and decide whether remaining energy is higher than its preset threshold, which would be, for example, 10% of its initial energy level. It will remove those senders that have energy lower than the threshold. Then for remaining senders, by jointly considering R_{jP}, and E_j, the period of asynchronous clock, which regulates the connection between sender j and the proxy, will be set as:

$$\Delta_j = \begin{cases} \max\left\{ \frac{b}{\frac{R_{jP}\cdot E_j}{\sum_{l=1}^{M-1} R_{lP}\cdot E_l}\cdot(R_{0P}-R_{SP})}, \frac{b}{R_{jP}} \right\}, & j = 1, 2, ..., M-1 \\ \frac{b}{R_{SP}}, & j = S \end{cases} \quad (11)$$

(11) shows that the amount of information streamed from j will be proportional to its R_{jP} and E_j. A sender with better connection and higher energy will get more opportunities to be queried by the proxy to send out data units. After a Δ_j expires, a transmission opportunity is granted to sender j. The proxy will then select a data unit to request using the simplified RaDiO framework [7], i.e. the optimal data unit DU_i for (re)transmission is the one with the largest $\lambda_i = \lambda_i' S_i / n_i$, where S_i is data sensitivity, and λ_i' is the increase in successful delivery likelihood given one transmission is sent at the optimization instant t. λ_i' and S_i can be defined as:

$$\lambda_i' = q_i(\pi_{i,1}) - q_i(\pi_{i,0}) \quad (12)$$

$$S_i = \sum_{k \succeq i} D_k \prod_{\substack{l \preceq k \\ l \neq i}} q_l(\pi_l) \quad (13)$$

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

IV. Evaluation

Using NS2.29, we simulate an IEEE 802.11 WLAN that consists of one base station and three wireless nodes (i.e. two peers and one client) as well as a wired server node. For wired connections, we set the available bandwidth for streaming to be 200 kbps, one hop delay to be 5ms, and the packet loss rate to be 4%. For wireless connection, we set the transmission range to be 250 m, data rate to be 11 Mbps, and the link layer delay to be 25μs. The bit error rate(BER) of the wireless link is 2e-5 and average RTP packet size is 4000 bit. We choose the initial energy of peer 1 to be 100 *Joule* and peer 2 to be 10 *Joule*. Each WNIC is configured as shown in Table I [6].

We encode test sequences *Foreman* and *Container* using H.264 standard into 300 frames each, and loop them to generate video traffic for the 25s simulation. The bit rate for each sequence is 120 kbps and frame rate is 15 fps. We choose the I-frame frequency to be 1/25. For background traffic, we consider 400kbps constant bit rate traffic running between the wired server and the streaming client.

P_t	P_r	P_s	P_i	T_j^t
1.425 W	0.925 W	0.045 W	0.80 W	0.75 ms

TABLE I
WNIC PARAMETERS

Using the above simulation settings, we implement both streaming schemes (with and without energy-aware scheduling). Figure 3 shows the energy consumption on each peer throughout the simulation. From Figure 3, we can see that by transitioning the WNIC from *idle* to *sleep* mode, each peer node can save energy and last longer. To minimize the randomness of simulation results, we run the simulation fifty times with different random seeds. By averaging simulation results across fifty runs, we can get the average total amount of energy spent during one simulation run, as shown in Table II.

From Figure 3, we can also see that the available energy of peer 2 is not enough for it to participate throughout a simulation run. To examine how it is going to affect the system performance with different schemes implemented, we measure the peak-signal-noise-rate (PSNR) of the received video sequence at the client side. From Figure 4, we can see that implementing energy-aware scheduling can help achieve smoother video quality by better distributing the streaming load among peers based on node energy level as well as network condition. Since the initial energy of peer 2 cannot

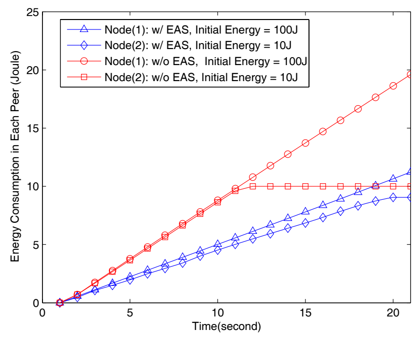

Fig. 3. Cumulative Energy Consumption on Each Peer ('*Foreman*')

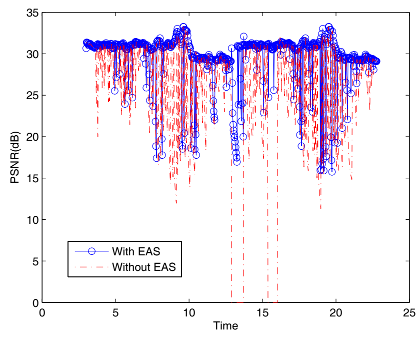

Fig. 4. Instantaneous PSNR vs. Time ('*Foreman*')

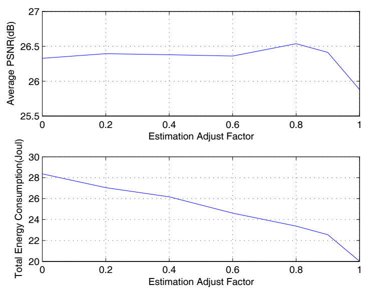

Fig. 5. Sensitivity Measures of Sleep Period Estimation ('*Foreman*')

last through the simulation, eventually the proxy realizes that peer 2's energy is beneath a certain threshold, i.e. 10% of the initial energy, and excludes it from the sender group before the actual energy depletion happens. Therefore, the system can react earlier to prevent the performance degradation caused by sudden failure of a peer connection. However, when the scheme is not energy aware, the proxy schedules the packet transmission based on only network conditions, which might end up putting more load on peer 2 and drain it faster. Since the proxy is not aware when peer 2 fails because of energy depletion, the proxy will still keep sending requests to it. The proxy finally detects the node failure by noticing the sudden increased packet loss on the connection to peer 2 and excludes it from the sender group. Unfortunately in the time this has taken, many packets have already been dropped. As a result, the video quality is degraded greatly, as shown in Figure 4. We repeat our PSNR measurement 50 times and summarize our results in Table II. Again, we can see that using energy-aware scheduling, the multi-source streaming system can achieve better performance by quickly reacting to quality degradation caused by a dying peer connection.

Video Sequence	***Foreman***	***Container***
PSNR w/o Loss (dB)	30.71	34.32
PSNR w/ Loss w/ EAS (dB)	25.88	31.97
PSNR w/ Loss w/o EAS (dB)	24.12	30.32
Consumed Energy w/ EAS (J)	20.01	22.65
Consumed Energy w/o EAS (J)	35.28	33.55

TABLE II
PERFORMANCE COMPARISON

In our simulations, we also run a sensitivity analysis on how the sleep period estimation affects the system performance. In other words, we examine how the system performance changes when we choose different values for *estimation adjust factor* α in (4). Previous simulation results are obtained when we set α to be 1. From Figure 5, we notice that when the estimation adjust factor starts decreasing from 1, the system performance first increases because more conservative estimation helps to lower the probability that the WINC might be asleep while a packet was being delivered but as the value keeps decreasing, this effect becomes more trivial. Instead, too conservative estimation makes the WINC spend more time in the *idle* state and increases its energy consumption. Eventually, some peer will be excluded from the system and the client has to update its sender group, resulting in performance fluctuation. This is why performance starts decreasing after certain point in Figure 5. One thing to emphasize is that even when we choose the estimation adjust factor to be zero, i.e. never putting peer nodes into sleep, we can still achieve a performance improvement by monitoring node energy level and reacting earlier to void the quality degradation caused by sudden energy depletion.

V. CONCLUSION

In this paper we propose an energy-aware scheduling scheme for supporting multi-source video streaming over IEEE802.11 WLAN. By jointly considering network conditions and energy levels of each peer node, the proposed scheme will stream more information from high-power peers and periodically transit sender WNICs into *sleep*. As a result, it can prevent the premature draining of low-power nodes. It can also avoid sudden peer failure by monitoring node energy level and switching to new peer nodes once it detects energy levels dropping lower than a preset threshold. NS-based simulation results show that our proposed energy-aware scheduling scheme can efficiently cut down energy costs and meanwhile provide comparable quality streaming service.

REFERENCES

[1] D. Li, C-N. Chuah, G. Cheung, and S.J Ben Yoo, "MUVIS: Multi-Source Video Streaming server over WLANs," *IEEE/KICS JCN*, 2005.
[2] N. Ramos, D. Panigrahi, and S. Dey, "Energy-efficient link adaptations in IEEE 802.11b wireless LAN," in *Wireless and Optical Communications*, July 2003.
[3] S. Chandra and A. Vahdat, "Application-specific network management for energy-aware streaming of popular multimedia formats," in *USENIX*, 2002.
[4] M. Tamai, T. Sun, K. Yasumoto, N. Shibata, and M. Ito, "Energy-aware video streaming with qos control for portable computing devices," in *NOSSDAV*, 2004.
[5] "Network simulator," http://www.isi.edu/nsnam/ns/.
[6] Y. Jiao and A. R. Hurson, "Adaptive power management for mobile agent-based information retrieval," in *Proc. of Advanced Information Networking and Application*, 2005.
[7] P. Chou and Z. Miao, "Rate-distortion optimized streaming of packetized media," Tech. Rep. MSR-TR-2001-35, 2001.