Energy-Efficient Transmission for Multimedia Streams in Last-hop Wireless Internet

Invited Paper

Albert F. Harris III, Robin Snader, and Robin Kravets
Computer Science Department
University of Illinois at Urbana-Champaign
{aharris,rsnader2,rhk}@cs.uiuc.edu

***Abstract*— Multimedia applications have unique characteristics that can be leveraged to design energy-efficient loss recovery mechanisms. Given their loss tolerance and strict timing requirements, various frame dropping policies can be used to affect both application performance and energy consumption. However, the effects on energy consumption of such frame dropping policies depends on not only high-layer effects, but also the MAC layer underlying them. Therefore, we first present an analysis of an energy-efficient MAC-layer protocol, called Fast Transmit MAC Protocol (FTMP). We then explore three frame dropping policies: proactive, reactive, and opportunistic. We present the EELR (energy-efficient loss recovery) protocol, which uniquely combines all three frame dropping policies to provide both high energy efficiency and high application performance. EELR is implemented in a real system on top of an energy-saving MAC layer that exposes information about transmission costs to the transport layer for use in frame dropping decisions. Through analysis and real-world experiments, we show that the combination of an energy-efficient MAC layer with a transport protocol using all three frame-drop policies outperforms any other combination.**

I. Introduction

One of the primary functions of a transport protocol is to provide end-to-end loss recovery. However, the level of loss recovery provided depends on the requirements of the application, while its effectiveness depends on the characteristics of the end-to-end channel. Because the transport layer sits between the application layer and the network layer, it has knowledge of both the needs of the application and the condition of the network, which naturally leads to the development of cross-layer protocols to provide effective loss recovery.

Loss recovery techniques have traditionally been evaluated by the cost they impose on the network in terms of additional bandwidth or the delay imposed on the application, which may cause violations in application timing requirements. However, in mobile environments, it is also necessary to consider the cost of loss recovery in terms of the energy consumed by the additional transmissions as well as the energy lost in the transmission of data that arrives past its deadline. Energy consumption is a function of the amount of application data and the amount of redundancy needed to recovery from losses. While the amount of data sent is obviously determined by the application, the redundancy used for loss recovery is determined by the transport protocol based on the needs of the application (*e.g.*, timing) and the channel characteristics (*e.g.*, loss rate). Additionally, the effects of loss recovery mechanisms at the transport layer on energy consumption can either be aggravated or ameliorated by the design of the MAC layer. Therefore, the design of an energy-efficient transport layer is predicated on the design of an appropriate MAC layer underneath it.

Multimedia applications present unique opportunities for the design of energy-efficient loss recovery due to their relatively high tolerance of loss and low tolerance of lateness. Additionally, multimedia data typically consists of a stream of large frames, each of which are packetized for transmission. While the division into packets could be left to the network layer, the transport layer can use this dependency information to make intelligent loss recovery decisions. These characteristics of multimedia data lead to the possibility of using frame dropping to increase energy efficiency. Essentially, frames that are either incompletely received or received late are useless to the receiver and therefore constitute wasted energy.

In this paper, we first present the design and analysis of an energy-efficient MAC layer, called Fast Transmit MAC Protocol (FTMP). We then explore the space of frame dropping policies, including proactive, reactive, and opportunistic. All of these policies leverage knowledge about the state of the channel and the interdependencies among application data to intelligently drop packets that would be useless to the receiver and so unnecessarily consume energy. To fully explore the energy saving potential of frame dropping techniques we present the design of an energy-efficient MAC layer that exposes energy costs to the transport layer for use in frame dropping decisions. The main contributions of this paper are a novel, proactive frame-dropping policy, a novel, opportunistic frame-dropping policy, and the design and implementation of an energy-efficient loss recovery protocol (EELR) that uses three frame dropping mechanisms in combination. This protocol is compared to a number of other protocol variants and shown to outperform them in terms of both goodput and energy efficiency.

The rest of this paper is as follows. Section II presents adaptation techniques used in the MAC layer. Section III presents an analysis of the effects of those techniques on energy con-

This work is partly funded by NSF CNS 0347468.

sumption. Section IV presents our energy-efficient MAC layer protocol. Section V examines two families of strategies for loss recovery. Section VI describes the three dropping policies used in our protocol. Section VII presents the implementation of our energy-efficient loss recovery transport protocol, EELR. Section VIII evaluates two EELR variants and four other protocols. Finally, Section IX discusses conclusions and future directions.

II. MAC-layer Adaptation Techniques

One component of the costs of reliability mechanisms is the retransmission cost. This cost at the transport layer is strongly dependent on the underlying MAC layer. Accordingly, the MAC layer must be analyzed before the transport layer can be understood. Therefore, this section and those immediately following present the design, analysis, implementation, and evaluation of an energy-efficient MAC protocol.

A. Communication Costs

Wireless communication devices consume some base energy when they are activated and idle. The act of transmitting a packet from one node to another uses power P_{card} as determined by the base costs of the wireless interface P_{base}, the extra costs to drive the components of the card used for transmission P_{xmit}, and the transmit power level P_t as follows:

$$P_{card} = P_{base} + P_{xmit} + P_t. \quad (1)$$

With CSMA-based protocols, the wireless interfaces are required to continuously sense the medium. Therefore, P_{base} is equal to the idle costs. In the next sections, the effects of bandpass modulation and transmit power control on energy consumption are examined.

B. Bandpass Modulation Adaptation

BPM compensates for noise on a wireless channel by encoding application-layer bits into "raw" bits to be sent through the air. The number of raw bits per application data bit, or the coding rate, along with other code-specific overhead, defines the transmission rate of the channel. More efficient codes achieve higher data rates but can tolerate less noise than lower efficiency codes. This tolerance can be captured by a noise threshold, defined for each specific scheme.

Since the choice of a bandpass modulation scheme determines the data rate, it also impacts energy efficiency. For a given bandpass modulation scheme, transmission energy E_t is determined by the data rate R, the amount of data to send in bits D, and the power required for transmission P_{card} according to:

$$E_t = \frac{D}{R} \times P_{card}. \quad (2)$$

For a fixed P_{card}, it is clear that higher values of R yield a lower E_t since D is not affected by the choice of a bandpass modulation scheme. Faster rates yield more energy-efficient communication.

To compensate for variations in interference levels in wireless links, wireless LAN communication supports a range of bandpass modulation schemes. For example, IEEE 802.11b [1] supports four schemes: 1 Mbps (DBPSK), 2 Mbps (DQPSK), 5.5 Mbps (CCK), and 11 Mbps (CCK). Current protocols for selecting a bandpass modulation scheme focus on choosing the fastest rate given the currently perceived channel quality, typically measured in terms of SNR. Ramanathan and Steenstrup [2] present a dual-channel slotted-aloha MAC-layer protocol that uses a separate control channel over which the receiver transmits feedback about losses encountered to the sender. The sender then chooses the bandpass modulation scheme to use based on the loss rate. Auto Rate Fallback (ARF) [3] adapts IEEE 802.11 to allow the sender to pick the best rate based on channel quality information from the last data packet. Gass, *et al.* [4] propose a protocol for point-to-point links that selects rates based on cached, per-link information about the previous channel quality. Vaidya, *et al.* [5] propose a receiver-side rate-adaptation scheme in which the receiver picks the best rate at which to send during the RTS/CTS packet exchange in the IEEE 802.11 protocol. They show that using receiver-side information improves the performance of the rate-adaptation algorithm. The general assumption of all of this work is that the sender has no control over the SNR at the receiver and so chooses the most effective data rate, which is essentially the fastest data rate that can tolerate the current SNR.

C. Transmission Power Control

TPC compensates for noisy channels by altering the transmit power level of the wireless interface with the goal of changing the SNR at the receiver to match the threshold SNR of the chosen bandpass modulation scheme. As the transmit power level increases, the signal strength at the receiver increases, improving the quality of the signal. However, higher transmit power levels increase the energy cost of communication. The energy efficiency of a TPC scheme is directly related to the transmit power level P_t (see Equation (1)) chosen for the data transmission. In the case of TPC alone, R is constant and lower values of P_t result in lower values of E_t (see Equation (2)). Therefore, given a fixed transmission rate, the lowest transmit power level that results in a sufficient SNR for a given bandpass modulation scheme yields the highest energy efficiency.

With the shared characteristics of wireless communication channels, any node within transmission range of the receiver can interfere with reception. The use of heterogeneous power levels among the nodes can lead to an increase in collisions. However, by using homogeneous transmit power levels for the RTS/CTS and ACK, these collisions lead to minimal performance degradation in wireless LAN environments [6].

D. Combination BPM & TPC

To combine BPM and TPC, the interactions between the two techniques must be understood. The goal of TPC is to choose the lowest possible transmit power level that will not violate some SNR threshold. However, such thresholds are dependent on a particular bandpass modulation scheme. Therefore, a bandpass modulation scheme must be chosen before TPC can

be used to set the transmit power level to match the SNR threshold. When choosing a bandpass modulation scheme, there are three possibilities: always choose the slowest scheme, choose varying schemes based on some algorithm, or always choose the fastest scheme. To understand the impact of these options on energy efficiency, it is necessary to look into the relationship between the capacity of the wireless channel and the transmit power level.

Shannon's law can be used to define the relationship between channel capacity and signal strength at the receiver, where C is channel capacity in bits per second, B is bandwidth in Hertz, S is signal strength in decibels, and N is noise level in decibels as follows [7]:

$$C = B \times log_2(1 + \frac{S}{N}). \quad (3)$$

However, C represents the raw capacity of the channel. To capture the achievable data rate, R, it is necessary to include the characteristics of the coding scheme. If α represents the overhead of the coding scheme (with values between zero and one, zero representing the highest overhead), the relationship between C and R can be stated as:

$$R = \alpha \times C. \quad (4)$$

This relationship captures the fact that coding schemes with higher overhead achieve lower data rates given the raw capacity of the channel.

The first possibility for choosing a bandpass modulation scheme is to pick the slowest one possible, as hinted at by Prabhakar, *et al.* [8], who use these relationships to choose the most energy efficient data rate and transmit power level. They note that if B and N are held constant, S is monotonically decreasing and convex in $\frac{1}{C}$ (see Equation 3). This implies that lower channel capacities require lower signal strengths. If a fixed α is assumed, the channel capacity defines the data rate, implying that slower rates require lower signal strengths. Finally, since signal strength at the receiver is directly related to the transmit power level used by the sender, slower data rates require lower transmit power levels [8].

While this implies that the slowest data rate is the most energy efficient, these results cannot be directly applied to the bandpass modulation schemes used in wireless LAN environments. A key assumption for Prabhakar, *et al.*, is that data rates are defined by the transmit power level and so reducing the transmit power level directly reduces the data rate. However, IEEE 802.11 interfaces use different bandpass modulation schemes to achieve different bit rates. Additionally, these interface cards have a discrete number of transmit power levels available. This, coupled with the fact that higher rate bandpass modulation schemes use more efficient encodings, and so have lower values of α, leads to the fact that always using the slowest rate bandpass modulation scheme does not result in the most energy efficient communication.

Consider a scenario where the noise in the environment is very low, 10^{-20} decibels, and the minimum transmit power level is 1 mW. The minimum transmit power level defines the minimum possible channel capacity given the noise level according to Equation (3). For Prabhakar, *et al.*, this in turn defines a transmission rate. However, actual wireless LAN environments use different bandpass modulation schemes to achieve different rates. Therefore, while an optimal channel capacity may have been located by using the lowest transmit power level, the *transmission rate* has not. Consider an IEEE 802.11b card with 11 Mbps CCK, 5.5 Mbps CCK, 2 Mbps DQPSK, and 1 Mbps DBPSK. With a noise level so low, the channel capacity is large enough to handle any of these bandpass modulation schemes. However, the transmitter must be on for a longer time at exactly the same transmit power level for the slower rates. Therefore, slower rates do not result in energy efficient communication. In fact, even if all transmission rates are not available at the lowest power setting, the slowest transmission rate does not yield the most energy efficient communication. Consider another scenario using a Cisco Aironet 350 series interface [9] with modes shown in Tables I and II. If the noise level in the environment is 1.8 dB, the channel capacity can sustain 1 Mbps and 2 Mbps rates at 5 mW. At 20 mW it can sustain 5.5 Mbps and 11 Mbps [9]. Because the energy consumption is four times greater at the higher rates, but the rates themselves are 5 to 10 times greater, the fastest rate results in the most energy efficient communication.

Transmit Power (mW)	5	20	30	50	100
Signal Level (dB)	7	13	15	17	20

TABLE I
CISCO AIRONET 350 POWER SETTINGS

Card Mode	Transmit	Receive/Idle	Sleep
Power (W)	2.24	1.35	0.075

TABLE II
CISCO AIRONET 350 MODE COSTS

The second possibility for choosing a bandpass modulation scheme is to use different rate schemes depending on the characteristics of the communication. Miser [6] attempts to choose the most energy efficient transmission rate, transmit power level combination for each packet by calculating a table offline containing transmission rate/power combinations indexed on factors such as how many times a particular packet has been retransmitted, etc. This results in varying choices of bandpass modulation schemes, not necessarily the fastest or the slowest. However, their approach is not practical since the network configuration must be known at the time the table containing the transmission rate/power combination is calculated offline. This network configuration information must indicate the number of contending stations and determine the RTS collision probability.

The final possibility for choosing a bandpass modulation scheme is to always choose the fastest available. This approach cannot be supported if only the energy consumption due to the transmit power level is considered. However, P_t accounts for only one part of P_{card}. Since P_t is typically on the order of 10 mW to 100 mW and P_{base} and P_{xmit} are typically on the order of 1 W [9], the cost of transmission is dominated by the energy consumed by the other components of the transmission costs.

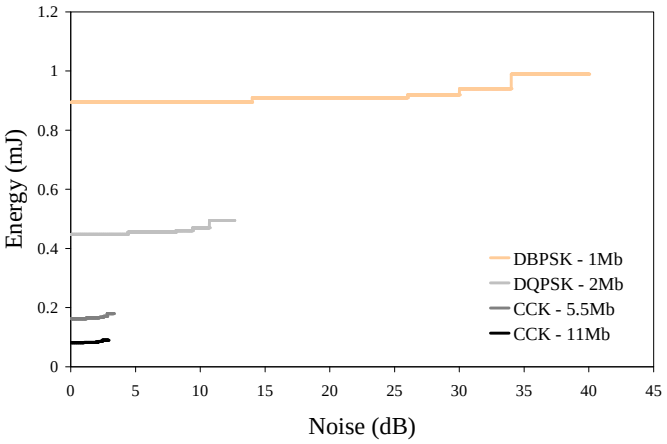


Fig. 1. Total communication energy consumption

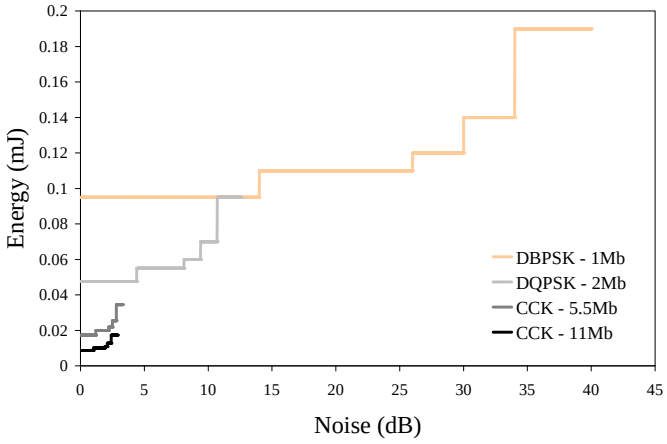


Fig. 2. Efficient card energy consumption

Therefore, solutions only targeting P_t minimize the smallest component of P_{card}. Once these costs are considered, we show that fast data transmission yields the most energy-efficient communication.

III. Fast Data Transmission

The benefits from fast data transmission stem from the relationship between P_{base}, P_{xmit}, and P_t (recall Equation (1)). However, the energy consumption of any wireless interface depends on the specific values of each. Since current wireless interfaces have a wide range of values, we present our analysis of fast data transmission on a Cisco Aironet 350 series card (recall Tables I and II). We then explore the impact of potential improvements in wireless interfaces to determine the range of cards for which fast data transmission is the most energy-efficient. Our analysis shows that major breakthroughs in wireless communication are necessary before something other than fast data transmission needs to be considered.

To show that fast data transmission produces the most energy efficient communication for wireless LANs, it is necessary to look at the total energy consumption for transmission, E_t, for each bandpass modulation scheme. Recall that E_t factors in P_{xmit} and P_{base} (see Equation (2)). To factor out the energy needed to keep the card idle and ignore the effect of power management, let $P_{base} = 0$ mW and $P_{xmit} = 980$ mW. Assuming $D = 1$ Kb, then energy consumption for each bandpass modulation scheme is represented by the step functions in Figure 1. The x-axis represents the noise at the receiver in dB and the y-axis represents the total energy consumption in mJ. Again, the most energy efficient strategy chooses the bandpass modulation scheme, transmit power level with the lowest total energy consumption. As Figure 1 depicts, once the total energy consumption of the interface card is considered, even with $P_{base} = 0$ mW, the fastest-rate bandpass modulation scheme always has the lowest energy consumption and therefore is the most energy efficient.

A power management scheme would increase the energy savings achieved by using the fastest rate bandpass modulation scheme. To see this, consider a perfect power management scheme that places the card in sleep mode whenever it is not transmitting. This essentially allows P_{base} to be factored into E_t (see Equation (2)), yielding $P_{base} = 1275$ mW, $P_{xmit} = 980$ mW, and $D = 1$ Kb. It is clear that, factoring in P_{base}, the gaps between the energy consumptions of the bandpass modulation schemes must grown, making the fastest bandpass modulation scheme clearly the most energy efficient.

This analysis of total energy costs leads to the consideration of the point where fast data transmission is no longer efficient. The gap between transmit and idle drives the fact that faster data transmission is always better. If this gap becomes much less, faster may not always be better. However, cards would have to become much better for that to be the case and radio technology is a mature discipline. RF transmitters are not becoming rapidly more efficient. Furthermore, there is a tradeoff between device size and efficiency [10]. The smaller RF devices become, the less efficient they are. This is a real issue for mobile devices as there is an obvious push to reduce the size of the devices. Therefore, P_{base} and P_{xmit} are not going to become negligible in the near future and faster data transmission (*i.e.*, keeping the time spent in transmit mode to a minimum), will continue to lead to energy efficient communication.

To quantify how much more efficient an interface would have to become, we find the minimum value of P_{xmit} for which the fastest bandpass modulation scheme is always the most efficient. E_t for each bandpass modulation scheme is represented in Figure 2 for $P_{base} = 0$ and $P_{xmit} = 90$ mW. E_t for DQPSK and DBPSK converges for noise at the receiver around 12 dB. Therefore, if P_{xmit} was reduced further, DBPSK would have lower E_t than DQPSK at 12 dB and the fastest rate would not always be the most power efficient. However, 90 mW is less than the highest transmit power level. In fact, this value is less than the hardware MAC processing costs and almost two orders of magnitude less than the cost of today's RF amplifiers [11]. Therefore, the faster-is-better approach is not likely to be outdated in the near future.

IV. Fast Transmit MAC Protocol

In this section, we present the Fast Transmit MAC Protocol (FTMP), a wireless communication protocol that achieves energy efficiency through fast data transmission. At a high level, FTMP first uses information about the SNR at the

receiver to determine the bandpass modulation scheme with the fastest possible rate. It then uses TPC to match the SNR at the receiver to the threshold SNR of the chosen bandpass modulation scheme. By choosing the fastest data rate and minimizing the time spent in transmit mode, FTMP complements idle-time power management [12], [13], [14], [15].

A. Adapting the Bandpass Modulation Scheme

FTMP bases its choice of bandpass modulation scheme on the receiver-indicated channel quality sent during channel negotiations (*i.e.*, RTS/CTS in IEEE 802.11). Each bandpass modulation scheme has a threshold SNR (SNR_t) for which it can maintain communication. We measured the performance of a Cisco Aironet 350 series card to find the thresholds for each of the four bandpass modulation schemes support.

To choose a bandpass modulation scheme, FTMP sends an RTS at the lowest data rate and highest power level (P_{max}) to the receiver. The receiver puts its perceived SNR in the CTS response. FTMP merely compares the current SNR perceived by the receiver (SNR_r) to the values in the table and uses the bandpass modulation scheme with the highest rate for which the SNR is above the threshold.

B. Choosing the Transmit Power Level

Since the choice of bandpass modulation schemes is relatively coarse-grained, it is likely that the SNR at the receiver is above SNR_t for the chosen bandpass modulation scheme. Therefore, FTMP uses TPC to reduce P_t to achieve SNR_t.

Current wireless interfaces do not have continuous power control; therefore, FTMP uses discrete power levels to find thresholds within each bandpass modulation scheme, yielding a table indexed on noise containing bandpass modulation scheme/transmit power level pairs. When a packet is ready to be sent, according to IEEE 802.11, the sender transmits an RTS packet at P_{max}. The receiver responds with a CTS containing SNR_r sent at P_{max}. The sender uses SNR_r to perform a table lookup to choose the appropriate bandpass modulation and P_t. The data is sent by the sender using the chosen bandpass modulation scheme and P_t. Finally, the receiver responds with an acknowledgment sent with P_{max}.

C. Implementation

To demonstrate that FTMP can be easily built into a system, we have implemented a prototype of FTMP on an Dell Latitude laptop running Linux with a Cisco Aironet 350 series interface. Implementing FTMP is very straightforward. The correct bandpass modulation scheme and transmit power level are selected according to the table lookup as described in Section IV. Finally, wireless interface driver calls are used to set the bandpass modulation scheme and the transmit power level of the wireless interface.

V. Loss Recovery Techniques

Loss recovery, critical to the needs of most applications, involves increasing the number of bytes transmitted. This redundant data allows the receiver to reconstruct the original application data in the event of loss, either by itself through the use of forward error correction (FEC) or more directly by requesting retransmissions from the sender. Both of these methods require an additional energy investment to transmit the redundant data, creating a tradeoff between reliability and energy efficiency.

The unit of loss recovery at the transport layer is the packet; however, typical multimedia data units (*i.e.*, frames) are much larger than the maximum packet size of the network. Therefore, transport protocols typically divide these frames into fragments, which are then transferred across the network and reassembled before delivery to the application. For a multimedia frame to be useful to the receiving application, all of its fragments must be received and reassembled before the frame's deadline.

FEC proactively achieves loss recovery by adding redundancy to each fragment of the transmitted data. If D is the size of an application frame, then FEC will increase the size to $D \times (1 + \mu)$, where μ is the amount of redundancy added per frame and so defines a correcting threshold. Since creating the error correcting codes requires computation and so incurs an energy cost, E_{comp}, the total energy cost for FEC-based loss recovery is

$$E_{FEC} = \frac{D \times (1 + \mu)}{R} \times P_{card} + E_{comp},$$

where R is the data rate and P_{card} is the power draw of the interface in transmit mode. FEC can only correct errors in the data if the number of lost fragments remains below the correcting threshold. A larger value of μ implies a larger correction threshold and a larger number of fragments that can be recovered, but requires sending more data. The main benefit of such proactive approaches is immediate data recovery at the receiver. In the face of a loss, no time is lost retransmitting the data. The cost of FEC comes from the extra μD bytes added to each frame and from E_{comp}. If the number of lost fragments is below μ, bandwidth and energy are wasted in the transmission of the unneeded redundancy. On the other hand, if μ is too small, loss recovery will fail and backup mechanisms will have to be used to repair the data, increasing E_{FEC}.

Retransmission-based loss-recovery strategies reactively repair losses by retransmitting lost fragments when a loss is detected. Such approaches have the advantage of increasing the number of bytes transmitted, and so consuming extra energy, only when there is a loss. Therefore, the amount of data sent by an automatic repeat request (ARQ) scheme is given by:

$$D\left(1 + \frac{n}{N}\right),$$

where N is the number of fragments and n is the number of retransmissions of fragments needed to recover the data. Since there is little computational overhead for retransmitting fragments, the total energy cost for retransmission-based loss recovery is

$$E_{ARQ} = \frac{D(1 + \frac{n}{N})}{R} \times P_{card}.$$

However, since losses typically cannot be detected before a round trip time has elapsed, it is possible that any retransmissions will arrive after the frame deadline has expired, wasting bandwidth and energy. The bandwidth consumed by these retransmitted packets may also cause subsequent frames to be late.

To compare ARQ with FEC, consider the channel loss rate. When the redundancy added by the FEC matches the loss rate of the channel exactly, μ is equivalent to $\frac{n}{N}$, since all of the redundancy added by the FEC is used for repair. Even in this case, however, the overhead of E_{comp} makes FEC more energy-expensive than a retransmission-based strategy. Additionally, in bursty environments like the Internet and wireless networks, the loss fraction $\frac{n}{N}$ varies, making it impossible to match the FEC correction threshold to the loss rate. This results in either extra redundancy or overhead from using other repair techniques like retransmissions. By overestimating the loss rate, FEC can work well in wired environments (either in conjunction with application frame-encoding techniques [16] or in multicast environment [17], [18]) where energy constraints are not an issue. In such environments, the rapid recovery outweighs the extra overhead for unused redundancy, which has little impact on performance. However, in wireless environments where energy is a major constraint and network conditions vary rapidly, the overhead for FEC makes it unsuitable for multimedia.

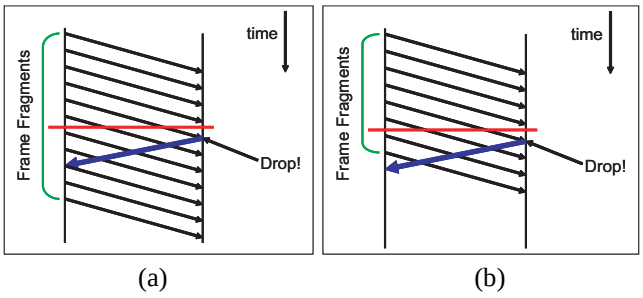
While FEC-based loss recovery is unsuitable for energy-constrained applications, neither is retransmission-based loss recovery a perfect solution. When a retransmission arrives at the receiver in a timely fashion, the fragment is still beneficial to the application and so the energy overhead of the retransmission was not wasted. However, if the retransmission arrives too late or was not necessary, the energy put into the retransmission was wasted. This raises the questions of when, if ever, retransmissions should be attempted.

VI. When a Frame Isn't Worth Saving

There are two situations in which it might be desirable to drop a frame at the sender. First, if a frame will not arrive at the receiver before its deadline, the energy and bandwidth associated with its transmission are wasted. Second, if the amount of loss is less than an application-specified threshold, it is possible to opportunistically suspend loss recovery to save energy. In this section, we discuss both of these scenarios and their impact on energy consumption. In the following section, we show that while each of these approaches in isolation can save energy, the benefits of combining them is greater than the sum of the individual improvements.

A. Timing Violations

Although it is not possible to entirely eliminate wasted energy due to frames missing their deadlines, two approaches can be used to reduce it. *Reactive* methods let the receiver notify the sender when a fragment of a frame is received late. The sender can then avoid sending any further fragments of that frame. *Predictive* methods allow the sender to predict if a frame or a fragment of a frame will be late. The sender can then avoid sending anything that it predicts will arrive late. The design and implementation of this method is one of the contributions of our work.



(a) (b)

Fig. 3. Reactive frame-dropping mechanism: (a) success, (b) failure

1) Reactive Dropping: Reactive methods leverage the fact that the receiver knows the deadline of a frame. Therefore, the receiver can tell if that deadline has passed during the reception of the fragments of a frame and signal the sender to stop sending any further fragments of that frame [19]. For example, the last two frames in Figure 3(a) could be dropped at the sender. While this can save the transmission energy of the unsent fragments, the energy to transmit the initial fragments is wasted. Additionally, the ability of this mechanism to save energy depends on the frame size relative to the bandwidth-delay product. If all of the fragments of a frame are in flight before the drop message is received by the sender, this mechanism results in no energy savings (see Figure 3(b)).

2) Predictive Dropping: Predictive methods leverage the fact that the sender has some information about frame deadlines and the round trip time (RTT). When a sender predicts that a frame being sent will arrive late at the receiver, the sender drops any unsent fragments of that frame. The specific deadline for a frame depends on the inter-frame spacing (IFS) of the multimedia stream and the size of the playout buffer at the receiver (B_p). However, to predict whether a frame will be useful to the receiver, an estimate of the arrival time of that frame is also needed. Thus, the sender must use an estimate of the one-way network latency in combination with the calculated deadline of the frame to predict whether a frame will arrive late. The challenge of predictive methods is that this estimation of frame timeliness must be accurate. If the estimation algorithm is too pessimistic, frames that could have arrived on time may be dropped, and if it is too optimistic, frames that can never arrive on time may still be transmitted.

For each fragment of a frame, it is necessary to calculate the playout time of the associated frame and to estimate the arrival time of the last fragment of that frame. The sender requires B_p, IFS, the number of fragments in each frame, and the frame number for each fragment, in addition to a continuously updated measurement of the network latency, which is calculated from the current round trip time estimate ($\overline{RTT}$) as $\overline{RTT}/2$. Because the transmission time is typically negligible compared to the network latency, an accurate estimation does not require a calculation of the network latency that separates out the bandwidth component of the round trip time.

If a frame fits in a single packet, then it is only necessary to

determine whether the packet containing the frame will arrive on time at the receiver. For a given frame F_i, this is equivalent to determining whether the following inequality holds:

$$t + \frac{\overline{RTT}}{2} < t_s + IFS \times i + B_p,$$

where t is the current time and t_s is the time at which the receiver began playout.

However, in practice, multimedia frames are split into multiple fragments, and simple network latency cannot be used to make this lateness prediction. The determination of whether a fragment will be useful depends not on its arrival time, but on the arrival time of the last fragment of its frame. If there are f fragments left to be sent in frame F_i, then for the last fragment to arrive on time, the following inequality must hold:

$$t + (T_f + \frac{\overline{RTT}}{2}) < t_s + IFS \times i + B_p,$$

where T_f is the amount of time to transmit the remaining fragments of the frame.

In the best case, T_f is negligible compared to $\overline{RTT}/2$. In the worst case, each of the $f-1$ fragments after the current one will take as much time to send as the current fragment. These edge cases represent the endpoints of the prediction interval

$$\left[\frac{\overline{RTT}}{2}, f \times \frac{\overline{RTT}}{2}\right].$$

Predictions based on the left-hand side of this interval will be optimistic and may cause fragments to be sent that will arrive late, while those based on the right-hand side will be pessimistic, and may cause frames to be dropped that would have arrived on time.

As will be shown in Section VIII, the largest gains in the number of good frames delivered and minimizing data expansion are due to predictive dropping. Predictive drops both conserve the energy to send frames that would have been late and allow subsequent frames to be sent sooner, building back some playout buffer at the receiver.

B. Opportunistic Dropping

Even if the sender estimates that there is enough time to recover a lost fragment, it may not be desirable to recover the frame. This decision can be based on network specific information, such as knowledge about the current level of congestion in the network, or on application-specific information, such as knowledge about the reliability requirements of the application.

If the receiver has been tracking the quality of the received multimedia stream, it may determine that the current quality is sufficient. A receiver could then decided not to ask for retransmission of the lost fragments [20], saving energy from the retransmission [21]. This can be implemented by allowing the receiver to send a false acknowledgment of lost fragments when the quality is sufficiently high. We take this approach one step further and allow the receiver to send a false acknowledgment for the whole frame, allowing the sender to avoid sending any unsent fragments in the frame.

Section VIII demonstrates that, although an opportunistic dropping policy wastes energy transmitting the initial fragments, it frequently prevents energy loss from subsequent frames that would have otherwise been late. Additionally, opportunistic drops allow a multimedia stream with very few frames left in its playout buffer to build back up a reserve, preventing large skips in the stream, and reducing the number of predictive and reactive drops needed later.

VII. EELR: Energy-Efficient Loss Recovery Transport Protocol

Given these insights into loss recovery, we designed an energy-aware transport protocol, EELR, that uses a novel combination of loss-recovery mechanisms to reduce energy consumption while maintaining reliability at a level specified by the application. By incorporating opportunistic dropping, EELR can more effectively use predictive dropping to minimize the number of reactive drops and increase the application goodput. EELR also supports adaptive applications by exposing relevant information about network cost and characteristics. In this section, we briefly describe our implementation of EELR and the parameters that control its frame-dropping policies.

We implemented EELR as a user-space library under Linux. This library implementation takes data from applications and sends it over UDP, with the timing and reliability described as above. A final implementation of EELR would, of course, be done at the kernel level to take advantage of the decreased packet overhead and system latency.

From the application, EELR needs information about frame deadlines and the playout buffer to support predictive dropping. The Real-Time Protocol (RTP) [22] can be used to support similar functionality without interfering with the reliability mechanisms. If the multimedia streams have heterogeneous timing and reliability needs, more sophisticated techniques [23] can be used. Additionally, to support opportunistic dropping, EELR needs information about the reliability requirements of the application. If the application does not provide a reliability level, EELR defaults to 100% reliability so that applications that are not EELR-aware can still function.

EELR passes information about the cost and quality of the multimedia stream and the current average number of bytes sent per good byte received to the application. This information, combined with data exposed by the MAC layer about interface costs, allows the application to estimate the energy cost of sending data. EELR also informs the application of frame drops so that it can react to the loss of frames if needed. For example, if the encoder is using a predictive method of encoding, the loss of one frame may imply the loss of following frames that are already encoded based on that frame. Therefore, the sooner the encoder is notified of frame loss, the fewer frames will be lost. This cross-layer design allows for a much more energy-efficient protocol than a traditional, strictly layered approach.

EELR uses the TCP New Reno congestion-control mechanisms [24], ensuring TCP-friendly behavior. However, the decisions about what data to send and whether to retransmit

a lost packet are based on the mechanisms described above. When retransmissions are used, EELR uses the TCP New Reno's retransmission mechanisms, including timeouts based on estimates of RTT, and the Fast Recovery/Fast Retransmit mechanisms. Since these are well studied mechanisms, we do not go into further detail here.

To control the optimism of EELR's predictive dropping algorithm (see Section VI-A.2), we provide a parameter ω to shift it within the prediction interval. Setting $\omega = 0$ corresponds to the left endpoint of given interval, and setting $\omega = 1$ corresponds to the right endpoint. Section VIII-A presents an evaluation of the effects of the choice of ω on the total number of dropped frames.

EELR must also calculate which frame fills the playout buffer and so triggers the multimedia stream to start playing. This frame, F_s, is calculated as follows:

$$F_s \ : \ s \geq \left\lceil \frac{B_p}{IFS} \right\rceil .$$

Given F_s, EELR can predict t_s, the playout start time, by estimating the arrival time of F_s. Any frame sent before the stream starts will arrive on time. Once frame F_s has been sent, EELR can start predicting whether subsequent frames will be late.

Thus, estimating whether a frame F_i, $i > s$, will arrive on time is accomplished by checking whether the following inequality holds.

$$(i \times IFS) - (t - t_s) + B_p \ > \\ ((\omega \times \overline{RTT}/2 \times f) + ((1 - \omega) \times \overline{RTT}/2).$$

Finally, to support application reliability requirements, EELR provides a reliability threshold, Γ. If the current reliability level is above Γ, lost fragments are never retransmitted and the associated frames are dropped. Disabling such retransmissions has two effects. First, dropping frames that are unnecessary instead of repairing them through retransmission conserves the energy that would have been spent on the retransmissions. This is the primary function of Γ. The secondary effect of disabling retransmissions is that each frame that is dropped allows the next frame to be sent sooner, potentially preventing subsequent late frames.

VIII. Evaluation

We evaluated the effectiveness of EELR using two metrics. The first metric is application goodput, the number of complete frames received on time. The second metric is the amount of data expansion caused by loss recovery, which shows the energy efficiency of the protocol. Using these metrics, we show that a complete, coordinated combination of the mechanisms presented above results in gains in both energy efficiency and application goodput over using other subsets of the mechanisms. The greatest gains come from predicatively dropping late frames, when combined with opportunistic drops.

For the experiments in this section, our testbed consisted of a Linux laptop running our implementation of EELR sending data via IEEE 802.11b through a base station to a wired node at various locations on the Internet. Here we present the results from representative case, where the wired node is one hop away from the base station. The other results are similar and due to space constraints are omitted.

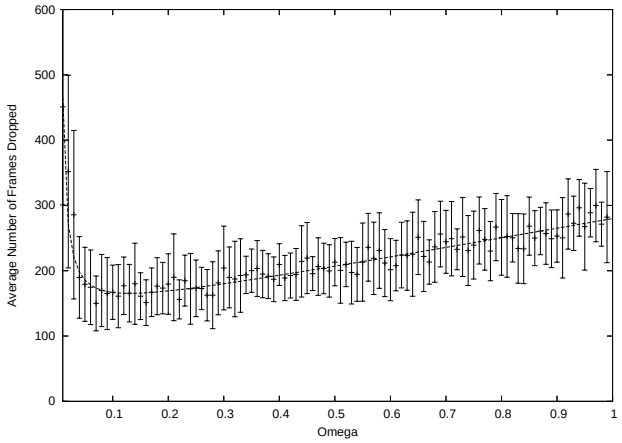


Fig. 4. Frame drops as a function of ω, with the mean over 50 trials $\pm$ standard deviation.

A. Effects of the Parameters

EELR has two parameters that are used to alter the behavior of the protocol. This subsection analyzes the effects of each one in turn.

To support predictive dropping, ω affects how optimistically EELR predicts that a frame will arrive on time. If EELR is over-optimistic, it transmits frames that end up being late at the receiver. This aggressiveness actually results in the successful transmission of more frames because it drops fewer frames than would have actually gotten through. Being over-optimistic, however, results in a *decreased* energy efficiency because the number of frames sent that are late outweighs the extra frames that are pushed through, resulting in a much higher data expansion for the multimedia stream. Being over-pessimistic, on the other hand, reduces the number of frames that are actually late to near zero, but causes frames that could have arrived before their deadlines to be dropped. If many of these frames are partially sent before they are dropped, this pessimism can also cause a decrease in energy efficiency. If the frames are all dropped at the beginning of the frame, energy efficiency increases, but the total number of frames received, and therefore the user experience, suffers.

To evaluate the effect of ω on EELR, we ran a large number of experiments on our real system (see Figure 4). These experiments confirm our intuition and indicate a good operating point of ω. With high values of ω, EELR is too optimistic about whether there is time to get a frame through. This has the effect of causing many reactive drops, and increases the total number of drops, decreasing goodput. If ω is too low, on the other hand, EELR is too pessimistic and predicatively drops a large number of frames unnecessarily, again decreasing goodput. We see from Figure 4 that the optimal value for ω should be approximately 0.13. Although EELR is relatively insensitive to the exact value chosen, values between 0.05 and 0.3 all give a total number of drops within

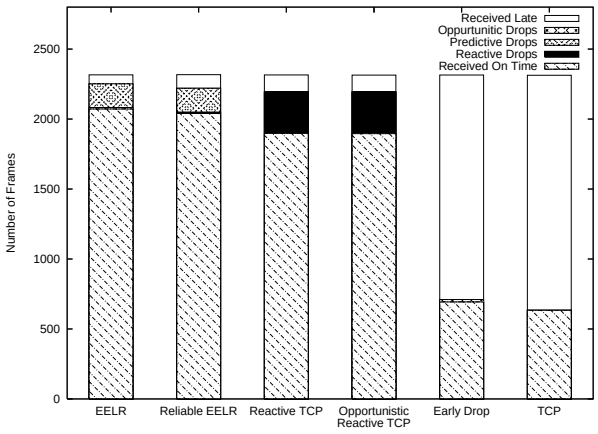


Fig. 5. Frames attempted and their disposition

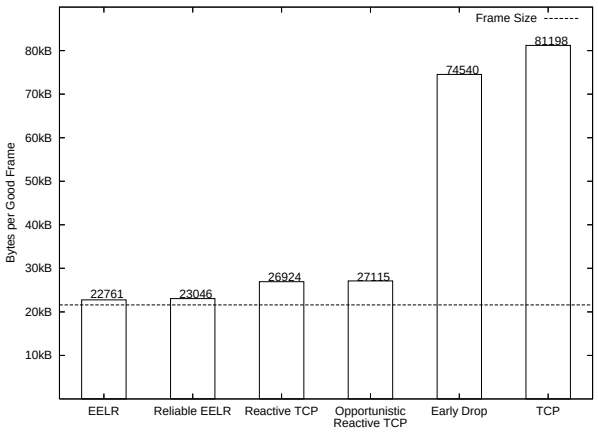


Fig. 6. Bytes per good frame

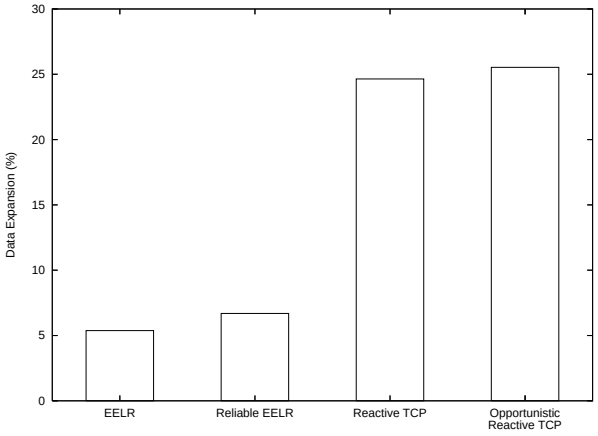


Fig. 7. Data expansion

10% of the optimal value. Choosing a value far from this range, however, can drastically increase the number of frames dropped. For our experiments, we use the optimal value of 0.13. Because these experiments were performed over a wide range of network conditions over the period of a number of days, we have confidence that the value of ω chosen will perform well in a wide variety of network conditions.

The use of opportunistic dropping is determined by the reliability threshold (Γ) desired by the application. If the long-term frame delivery ratio is above Γ, EELR disables retransmission of lost packets, and drops the associated frame to conserve energy. Essentially, lower thresholds allow more frames to be dropped during times when the network is in a particularly good state. This allows a buffer to be built that can be consumed during times of poor network performance. The ability of opportunistic dropping to reduce the number of predictive drops is shown in the next section.

B. Performance Analysis

To understand the impact of combining the mechanisms used by EELR, the following analysis presents goodput and energy analysis for six protocols: EELR, EELR without opportunistic drops, TCP with reactive drops, TCP with reactive and opportunistic drops, early drop, and TCP. TCP yields full reliability with no concern for timing constraints. Early drop aborts frames if losses are encountered, but has no mechanism to account for late frames. TCP with late frame drop sends all frames reliably but drops frames if notified by the receiver that they will be late. The unreliable variant disables retransmissions when the fraction of frames successfully received is above Γ. EELR is run on top of FTMP, which exposes the current transmission rates and costs up the stack. Finally, EELR uses all of the mechanisms described in the previous section, with either full or application-specified reliability. The following results are averages over 50 runs. Each run transfers a 50MB file over an IEEE 802.11b wireless card in a laptop through a base station to a wired receiver one hop away. The wireless conditions and the network load vary over time.

We first evaluate the application goodput for each protocol. As seen in Figure 5, TCP and early drop perform poorly. While TCP gets all of the frames to the receiver, less than 30% of the frames are on time. Early drop performs slightly better, due to its ability to use opportunistic dropping to catch up in the event of a packet loss. However, many frames are still received late. Reliable TCP with reactive dropping performs significantly better, getting roughly 82% of the frames through and on time. This improvement comes from the ability to abort sending frames that are arriving late. TCP with reactive and opportunistic drops performs slightly better. EELR's predictive dropping mechanism avoids nearly all reactive drops and increases the goodput nearly 10% over the opportunistic, reactive TCP. EELR without opportunistic dropping gets approximately 88% of the frames through successfully. However, with opportunistic drops and Γ set to 98%, EELR gets over 89% of the frames through to the receiver on time and intact. The majority of the unsent frames are due to EELR predicting that the frame will be late and skipping ahead, and thus saving the associated transmit energy.

Next, we evaluate the energy efficiency of the protocols. Recall that the energy consumption of a transport protocol is directly affected by the number of bytes used to transmit the data usable by the receiver. Essentially, this is the stream-level data expansion for the protocol. Figure 6 depicts the average bytes sent per good frame for each of the protocols. The line in the graph at about $\sim$22K bytes represents the average

number of bytes per frame. Therefore the space above the line represents the average data expansion per frame. Figure 7 shows the data expansion for the multimedia protocols only. As expected, EELR has a significantly lower data expansion and therefore better energy efficiency compared to the other protocols.

In addition to sending significantly fewer bytes per good byte, EELR actually sends at least 8% fewer total bytes than any of the other protocols, thereby using less total network energy. It is able to do this through its predictive dropping mechanism: since EELR does not have to wait for the receiver to notify it about timing violations, EELR can avoid sending even the first fragments of a frame which will be late.

IX. Conclusions and Future Directions

In this work, we have presented both an energy-efficient MAC layer protocol called Fast Transmit MAC Protocol (FTMP) and an energy-efficient transport protocol EELR. FTMP was designed based on the fact that the base energy costs to keep network interfaces active dominate the transmit energy consumption. EELR uses a novel combination of frame dropping policies for streaming multimedia data. We combined reactive, predictive, and opportunistic methods into our protocol, EELR. We further presented a Linux implementation of EELR and presented the results of a number of experiments that demonstrated EELR superior performance when compared with a number of other protocols in terms of energy efficiency and goodput.

EELR contains the hooks to allow different levels of reliability to be given to different frame types (*e.g.*, higher reliability for MPEG I-frames than B-frames). We intend to run a large battery of experiments to determine the effects of interleaving frames with different reliability levels on the efficiency of the frame dropping policies.

With respect to the specifics of EELR, it would be interesting to explore the impact of alternative loss-notification algorithms, such as ELN [25] and SACK [26]. Such algorithms provide more feedback to the transport layer allowing more intelligent loss recovery decisions as well as more accurate feedback to the application.

Finally, it would be interesting to develop a rate-based version of EELR, evaluating the interactions of rate-based mechanisms with the mechanisms presented in Section V.

References

[1] I. C. Society, "802.11: Wireless LAN medium access control (MAC) and physical layer (PHY) specifications," June 1997.

[2] R. Ramanathan and M. Steenstrup, "Hierarchically-organized, multihop mobile wireless networks for quality-of-service support," *Mobile Networks & Applications*, vol. 3, no. 1, pp. 101–119, June 1998.

[3] A. Kamerman and L. Monteban, "WaveLAN-II: A high-performance wireless LAN for the unlicensed band," *Bell Labs Technical Journal*, 1997.

[4] J. Gass, M. Pursley, H. Russell, R. Saulitis, C. Wilkins, and J. Wysocarski, "Adaptive transmission protocols for frequency-hop radio networks," in *Proc. 1998 IEEE Military Communications Conference*, vol. 2, October 1998.

[5] G. Holland, N. Vaidya, and P. Bahl, "A rate-adaptive mac protocol for multi-hop wireless networks," in *Proc. 7th ACM International Conference on Mobile Computing and Networking (MobiCom '01)*, 2001.

[6] D. Qiao, S. Choi, A. Jain, and K. G. Shin, "Miser: An optimal low-energy transmission strategy for ieee 802.11a/h," in *Mobicom*, 2003.

[7] W. Stallings, *Data and Computer Communications*. Prentice Hall, 2004.

[8] B. Prabhakar, E. Uysal-Biyikoglu, and A. El Gamal, "Energy-efficient transmission over a wireless link via lazy packet scheduling," in *Infocom*, 2001.

[9] Cisco, "Cisco Aironet 350 client data sheet," http://www.cisco.com/.

[10] J. Staudinger, "Issues and trends in mobile cellular transmitter power amplification," in *Wireless Circuits, Interconnection, and Assembly Workshop*, 1996.

[11] J.-P. Ebert and A. Wolisz, "Combined tuning of RF power and medium access control for WLANs," *Mobile Networks & Applications*, vol. 5, no. 6, pp. 417–426, Sept. 2001.

[12] L. M. Feeney, "An energy consumption model for performance analysis of routing protocols for mobile ad hoc networks," *Mobile Networks and Applications*, vol. 6, no. 3, pp. 239–249, June 2001.

[13] R. Kravets and P. Krishnan, "Power management techniques for mobile communication," in *Proc. Fourth ACM International Conference on Mobile Computing and Networking (MOBICOM'98)*, 1998.

[14] T. Simunic, L. Benini, P. Glynn, and G. De Micheli, "Dynamic power management for portable systems," in *Proc. 6th ACM International Conference on Mobile Computing and Networking (MobiCom '00)*, 2000.

[15] M. Stemm, P. Gauthier, D. Harada, and R. Katz, "Reducing power consumption of network interfaces in hand-held devices," in *Proc. 3rd. International Workshop on Mobile Multimedia Communications*, 1996.

[16] B. Wah, D. Lin, and X. Su, "A survey of error-concealment schemes for real-time audio and video transmission over the Internet," in *Proc Int Symposium on Multimedia Software Engineering*, 2000.

[17] S. Pejhan, M. Schwartz, and D. Anastassiou, "Error control using retransmission schemes in multicast transport protocols for real-time media," *IEEE/ACM Transactions on Networking*, 1996.

[18] J. Seo, N. Sung, S. Lee, N. Park, H. Lee, and C. Cho, "An adaptive type-i hybrid-ARQ scheme in a TDMA system over a non-stationary channel," in *First Workshop on Resource Allocation in Wireless Networks*, 2005.

[19] R. Sinha and C. Papadopoulos, "An adaptive multiple retransmission technique," in *NOSSDAV*, 2004.

[20] R. Kravets, K. Calvert, P. Krishnan, and K. Schwan, "Adaptive variation of reliability," in *HPN 1997*, 1997.

[21] L. Donckers, P. Havinga, G. Smit, and L. Smit, "Enhancing energy efficient TCP by partial reliability," in *13th IEEE PIMRC*, 2002.

[22] I. C. Society, "RTP RFC 3550," July 2003.

[23] J.-R. Li, S. Ha, and V. Bharghavan, "HPF: A transport protocol for supporting heterogeneous packet flows in the Internet," in *IEEE INFOCOM*, 1999.

[24] I. C. Society, "TCP RFC 2581," April 1999.

[25] ——, "The addition of explicit congestion notification (ECN) to IP RFC 3168," September 2001.

[26] ——, "TCP selective acknowledgement options, RFC 2018," October 1996.