

# Adaptive Fragmentation for Latency Control and Energy Management in Wireless Real-time Environments\*

Dinesh Rajan and Christian Poellabauer  
Department of Computer Science and Engineering  
University of Notre Dame  
Notre Dame, IN 46556  
{dpandiar, cpoellab}@cse.nd.edu

## Abstract

*Wireless environments are typically characterized by unpredictable and unreliable channel conditions. In such environments, fragmentation of network-bound data is a commonly adapted technique to improve the probability of successful data transmissions and reduce the energy overheads incurred due to re-transmissions. The overall latencies involved with fragmentation and consequent re-assembly of fragments are often neglected which bear significant effects on the real-time guarantees of the participating applications. This work studies the latencies introduced as a result of the fragmentation performed at the link layer (MAC layer in IEEE 802.11) of the source device and their effects on end-to-end delay constraints of mobile applications (e.g., media streaming). Based on the observed effects, this work proposes a feedback-based adaptive approach that chooses an optimal fragment size to a) satisfy end-to-end delay requirements of the distributed application and b) minimize the energy consumption of the source device by increasing the probability of successful transmissions, thereby reducing re-transmissions and their associated costs.*

## 1 Introduction

Efficient and innovative technologies for mobile wireless systems have led to their increased use in several sophisticated environments such as multi-player gaming and autonomous surveillance in battlefields. However their continued use is limited by several critical factors such as low energy resources. There have been several efficient and commonly adapted techniques that enhance the energy efficiency of these systems. Energy management tech-

niques for communication-oriented mobile systems (e.g., sensing robot swarms) are often concentrated at the physical and link layers of the protocol stack [7, 8]. A common link-layer technique for error-control and energy management is *dynamic fragmentation* that reduces the cost of re-transmissions by fragmenting the network-bound data and increasing the probability of successful transmissions in unreliable and noisy wireless environments [4, 5].

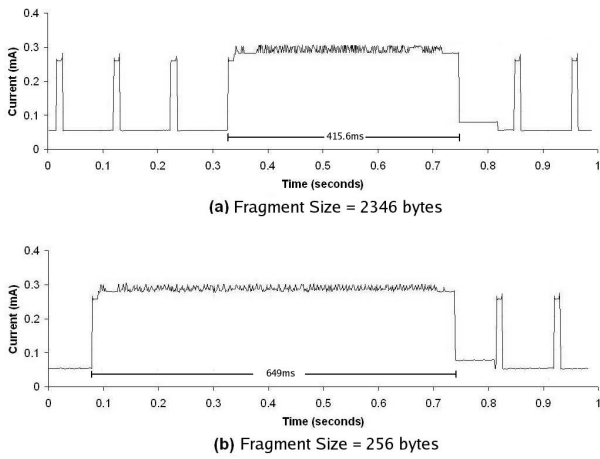
Fragmentation in retransmission-based schemes offer significant advantages in environments that are less tolerant to packet losses and errors (e.g., a remote tracking operation where packet loss significantly disrupts the quality of data analyzed). A disadvantage with re-transmission based mechanisms is that they often lead to violation of timeliness constraints due to the introduced latencies and overheads. However re-transmissions can still be employed for benefits in wireless environments with real-time constraints such as media streaming and remote surveillance. In such environments, the successful arrival of all required data is vital to application performance and effectiveness. For instance, in remote surveillance, when packets are lost or corrupted, re-transmission of packets can significantly improve the quality of surveillance, *even if re-transmitted packets arrive after their end-to-end deadlines.*

Existing fragmentation techniques often neglect the real-time requirements of the target environment and therefore lead to significant performance degradation (e.g., missed deadlines). In this work, we extensively study the latencies, overheads and costs incurred by dynamic fragment-size adaptation techniques for network-bound data in real-time environments. Based on our experimental analysis we improve on existing fragmentation techniques by introducing a feedback-based control mechanism that adapts fragment sizes with the goal to simultaneously satisfy real-time requirements and maximize achievable energy savings.

Figure 1 illustrates our experimental study of the latencies involved with fragmentation. Plots (a) & (b) in this fig-

---

\*This work is supported in part by NSF under grant number CNS-0545899.



**Figure 1. Latencies incurred with fragment sizes of 2346 and 256 bytes during transmission of an image of size 100 kb.**

ure represent the time taken in transmitting an image of size 100 kb with a *Linksys compact-flash wireless card* employing two different fragment sizes of 2346 bytes (maximum) and 256 bytes (minimum). The vertical axis represents the current drawn and hence the power consumed by this card during transmission. These measurements were made with a Picotech ADC-100 PC oscilloscope (100kSamples/s, 12 bit resolution). When the smallest fragment size of 256 bytes is used, an increase of about 55% in transmission time is observed in comparison to the transmission with a larger fragment size of 2346 bytes. Such a significant difference in the transmission latency necessitates fragmentation techniques to carefully consider the effects of introduced latencies on the real-time requirements of applications. Based on these observations, this work introduces a novel technique for dynamically tuning the fragment size to satisfy real-time (end-to-end deadlines) and performance (low packet loss, energy efficiency) guarantees.

As a motivational example, consider two users equipped with a PDA or laptop involved in streaming media content in a wireless environment. Typically, a few infrequent end-to-end deadline misses are tolerated in the form of a few dropped or delayed video-frames and therefore these applications are categorized as ‘soft real-time systems’. With existing fragmentation mechanisms [5, 8], the source device would select an optimal fragment size based on the channel conditions (such as bit error rate). In a worst case scenario with high channel error rates, the smallest fragment size (e.g., 256 bytes) would be chosen by existing mechanisms to achieve successful transmissions. As discussed earlier, smaller fragment sizes incur higher latencies. So, in the event of large propagation delays through the channel (e.g., due to interference or cross traffic) smaller fragment sizes would often lead to more deadline-misses. Thus exist-

ing fragmentation schemes can be considered sub-optimal in satisfying end-to-end real-time requirements. In such a scenario, slightly larger fragment sizes besides achieving acceptable transmission success-rates would incur lower latencies allowing deadlines to be satisfied.

This paper presents a novel mechanism that employs a feedback-based controller to dynamically determine an optimal fragment size based on *feedback of the incurred end-to-end latencies and the channel conditions*. The main contributions of this work can thus be summarized as a) study and analysis of the latencies and energy costs incurred from different fragment sizes and b) design and implementation of an adaptive mechanism to dynamically vary the fragment sizes to achieve energy efficiency while satisfying end-to-end real-time requirements of the participating applications.

## 2 Related Work

In wireless environments, energy efficiency has been of paramount importance and a multitude of research efforts have addressed it through various mechanisms developed at various layers of the protocol stack [7, 8].

Littieri et al. in [5] propose an adaptive mechanism for varying the MAC layer frame sizes used to partition the network-bound data to enhance throughput and energy savings. This is achieved by computing a frame length based on the current medium conditions characterized in terms of the bit error rate (BER). Our work extends on this related effort by also considering the real-time aspects of mobile distributed systems in fragmentation mechanisms.

Delay-control and energy efficiency with respect to dynamic fragmentation have been studied individually in [1] and [8] respectively. In [1], the time required to successfully transmit fragments of different sizes is formulated and discussed for the given data and fragment sizes. An approach that dynamically adapts fragmentation sizes, re-transmission retry limit, and transmission power for achieving energy efficiency in wireless systems was presented in [8]. However the optimal fragment sizes computed in these efforts do not consider the latency and energy costs incurred during the fragmentation process.

A feedback-based approach similar to our proposed mechanism was studied and employed in [6] where a Markov chain model was used to adapt the fragment size based on channel conditions. Our work extends the feedback received by the source node to include the experienced end-to-end latencies along with the channel error rates. This feedback is then used as an input to a controller algorithm that determines an optimal fragment size. Thus our adaptive fragmentation mechanism builds on these related efforts to simultaneously achieve end-to-end delay control and energy efficiency in mobile environments with ‘soft’ real-time requirements such as media streaming.

### 3 Adaptive fragmentation for latency control and energy efficiency

In this section, the concept of dynamically adapting fragment sizes based on observed end-to-end latencies and channel conditions is presented.

#### 3.1 System Model

Fragmentation refers to the partitioning of data received from the higher protocol layers (e.g., TCP) into smaller fragments based on the fragment size specified at the MAC layer in the IEEE 802.11 standard [2]. When fragmentation is employed on network-bound data (packet) of size  $b$  bytes, the number of resulting fragments ( $n$ ) is given by,

$$n = \lceil \frac{b}{k} \rceil \quad (1)$$

where  $k$  is the specified fragment size.

Each fragment is transmitted with a header ( $h$  bytes) to aid re-assembly of the original data at the destination. For the remainder of this paper, the term 'frame' is used to indicate a fragment (containing the partitioned data) along with its attached header. The probability of successful transmissions increases with decreasing frame size ( $k+h$ ) as given by the relation involving the bit error probability ( $BER$ ) [5],

$$p = (1 - BER)^{k+h}. \quad (2)$$

The transmission of a frame is said to have failed if it is lost in propagation through the network or is received with errors at the destination. From the computed probability in Equation 2, the average number of attempts required for the successful transmission of a frame can be expressed as

$$a = \frac{1}{(1 - BER)^{k+h}}. \quad (3)$$

The time taken for transmitting a frame of size  $k+h$  bytes depends on the transmission rate in bytes per second ( $R$ ) [1]:

$$t_{frame} = \frac{(k+h)}{R}. \quad (4)$$

The time overheads during transmission of a packet of size  $b$  consisting of  $n$  such frames can thus be formulated as

$$t_{total} = (n * a * t_{frame}) + t_{overhead} \quad (5)$$

where  $t_{overhead}$  represents the latencies incurred during partitioning of the packet into fragments of size  $k$ .

The energy overheads due to fragmentation can be analyzed similar to Equations 4 & 5. The energy consumed in transmitting a frame ( $E_{frame}$ ) is related to the transmit

power in watts ( $W$ ) and is computed by the relation

$$E_{frame} = W * t_{frame}. \quad (6)$$

It should be noted that a failure in frame transmission due to loss or errors results only in the corresponding frame being re-transmitted. Thus the energy consumed in a frame re-transmission equals  $E_{frame}$ . If  $E_{overhead}$  is the energy expended during fragmentation, then the energy consumed in successfully transmitting a packet can be given as

$$E_{total} = (n * a * E_{frame}) + E_{overhead}. \quad (7)$$

#### 3.2 Experimental Analysis

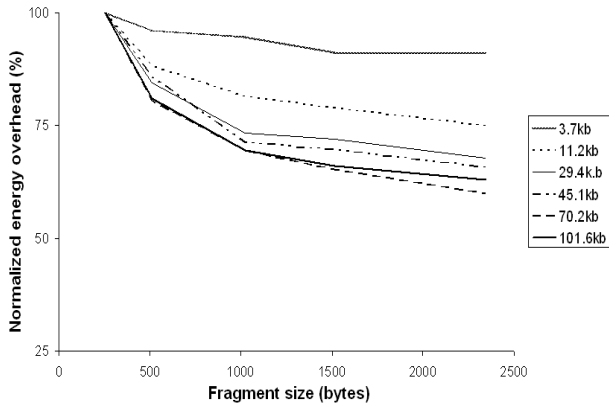
This section analyzes the energy consumed at the source node and the end-to-end latency experienced due to the overheads incurred from fragmentation. The experimental analysis and implementation in this work were made with an 11Mbps *Linksys compact-flash wireless (WCF12)* network card. This card operates at 3.3 volts and consumes 250mA when transmitting data. Frame sizes in the range between 256 and 2346 bytes were supported by the device driver provided in the kernel for this wireless card.

##### 3.2.1 Energy Consumption

The energy consumed during fragmentation is characterized by Equation 7. The factor  $E_{frame}$  which represents the energy consumed in transmitting a frame, increases linearly with the fragment size  $k$ .

The energy overheads ( $E_{overhead}$ ) incurred due to the fragmentation process were measured by considering the transmission times for a packet. The actual transmission times will include the fragmentation overheads besides other processing and buffering delays experienced before a packet is transmitted as partitioned fragments. Since the actual packet transmission times were observed to vary significantly with different frame sizes in our experiments, the latencies due to fragmentation were considered to dominate the other overheads experienced during a packet transmission. Figure 2 shows the energy overheads incurred with different fragment size settings used for transmitting data of varying sizes ranging between 3.7kb and 101kb. The plotted energy overheads for each data size ( $b_i$ ) are normalized to the energy consumed when a fragment size of 256 bytes is employed for this data size  $b_i$ . The overheads increase with decreasing frame sizes as a result of a) additional processing required in partitioning and framing due to the larger number of frames involved and b) increased transmission times resulting from higher header overheads.

In wireless environments with poor channel conditions (high  $BER$ ), the probability of failed frame transmissions is



**Figure 2. Energy overheads due to fragmentation and other actions performed on data transmitted by the Linksys WCF12 card.**

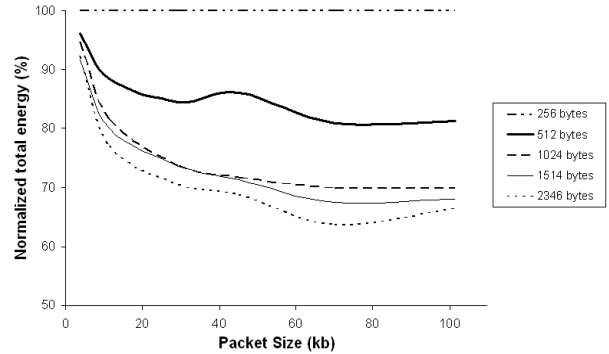
high (Equation 3). In such cases, there exists a trade-off between lowering the number of re-transmissions by employing smaller frame sizes and avoiding the energy overheads from fragmentation. Figure 3 illustrates the overall energy consumption (normalized to the energy consumed when a size of 256 bytes is used in fragmentation) for the Linksys WCF12 card under two different channel conditions.

The channel conditions (*BER*) play a significant role in determining an optimal fragment size that achieves maximal energy efficiency due to its impact on the number of re-transmissions. With good channel conditions (low BER), the overheads from fragmentation dominate the energy consumed by the system and therefore larger fragment sizes would yield lower overheads and higher energy savings. Re-transmission overheads become more significant with increasing BER and smaller fragments have higher probability of successful transmissions resulting in higher energy efficiency. This observation is illustrated in Figure 3 where a frame size of 2346 bytes consumes less energy when the BER is  $1 * 10^{-3}$  while frames of 1024 and 1514 bytes achieve higher energy savings (depending on the packet size) in channels with a BER of  $5 * 10^{-4}$ . Therefore these tradeoffs need to be carefully considered with respect to the channel conditions and the introduced overheads in computing a fragment size for optimal energy savings.

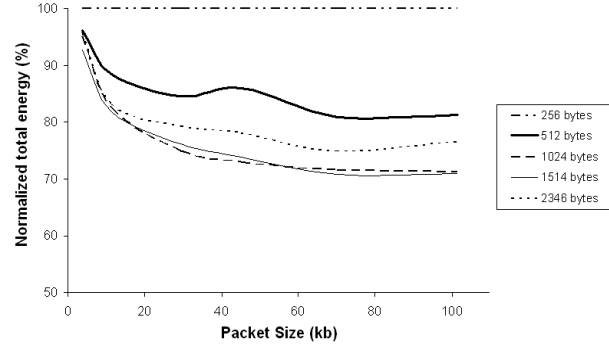
### 3.2.2 Latency

The end-to-end latency  $L$  experienced by a packet before it is delivered to the application at the destination can be expressed with the following equation, where  $n$  indicates the number of nodes traversed by the packet:

$$L_{total} = \sum_{i=1}^n (L_{node}^i) + \sum_{j=1}^{n-1} (L_{link}^j). \quad (8)$$



(a) BER =  $5 * 10^{-4}$

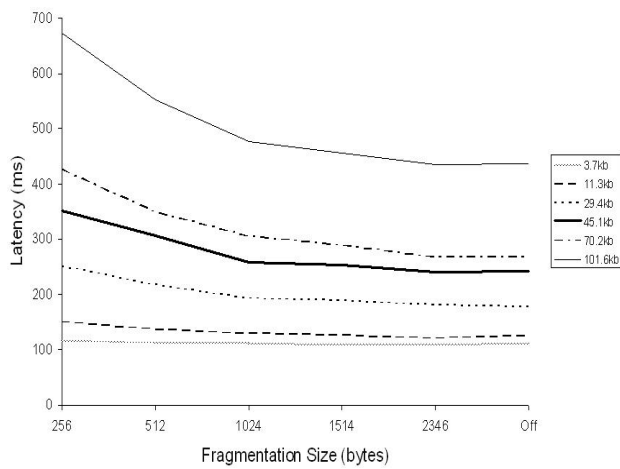


(b) BER =  $1 * 10^{-3}$

**Figure 3. Overall energy consumed by the Linksys WCF12 network card in different channel conditions represented by BER =  $5 * 10^{-4}$  and  $1 * 10^{-3}$ .**

The latencies at a node (source, destination, or any intermediate node) are expressed as  $L_{node}$  and includes the delays and overheads associated with processing (such as fragmentation or re-assembly) and buffering. The parameter  $L_{link}$  represents the transmission delays over the communication link which is determined by its bandwidth, cross-traffic, and channel conditions.

Figure 4 shows the transmission latencies experienced at the source node ( $L_{src}$ ) for the Linksys WCF12 card. It analyzes these latencies for six different frame sizes in the supported range of 256 to 2346 bytes. Note that for this wireless card, turning the fragmentation 'off' is equivalent to using the largest supported frame size (2346 bytes). The x-axis represents the fragment sizes used in partitioning packets of sizes ranging from 3kb to 100kb while the vertical axis denotes the time-overheads experienced due to fragmentation. The latencies indicated in this figure are experimental measurements of the total latency represented by Equation 5. It shows that as frame sizes decrease (from 2346 bytes to the smallest size of 256 bytes), the incurred overheads record a significant increase ranging from 7.02% for a 3.7kb packet to 54.53% for a 101kb data packet. This observation is due to the fact that for a given data size, larger

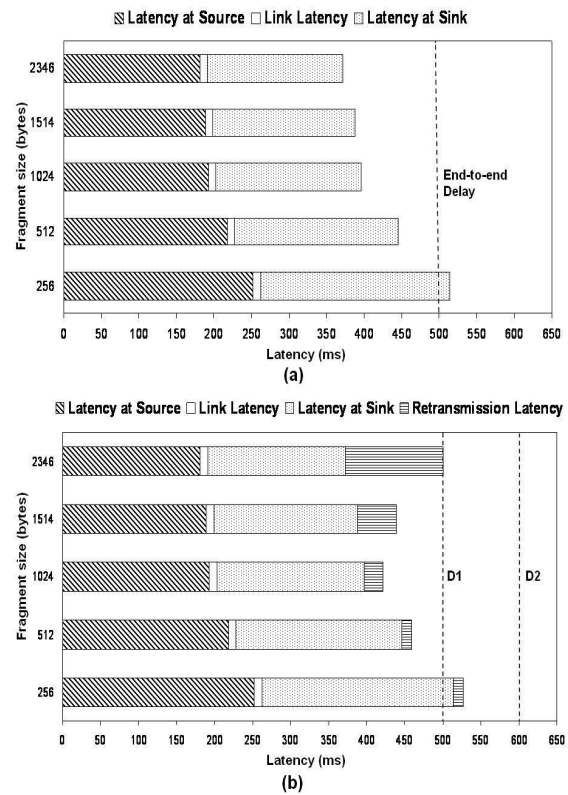


**Figure 4. Source latencies observed for different fragmentation size thresholds on various data sizes ranging from 3kb to 100kb.**

frame sizes result in fewer required frames for packet transmission compared to smaller frame sizes that produce more frames. A higher number of frames leads to increased costs during fragmentation at the source, transmission through the medium, and re-assembly at the destination.

The overheads described above need to be carefully considered in an environment which demands end-to-end timeliness. For example, consider a video streaming application involving two mobile nodes with demands on the frame arrival times to provide acceptable frame display rates. Without any loss of generality, assume that the packet (video-frames) size is fixed, say at 30kb and that five different frame sizes are supported between 256 and 2346 bytes. Consider an end-to-end deadline for these frames from the time they are generated, to their fragmentation, transmission, re-assembly and display (a deadline of 500ms is acceptable for live streaming to achieve reasonable frame display rates). The effects of employing different frame sizes on this constraint are presented in Figure 5 with the latency values at the source ( $L_{src}$ ) being derived from Figure 4. At the destination, the processing and re-assembly of frames are considered to be performed in parallel to frame receptions and therefore the latency experienced ( $L_{destn}$ ) is shown to equal the time taken at the source for transmitting the frames ( $L_{src}$ ). For simplicity, a uniform propagation delay or link-latency ( $L_{link}$ ) is assumed.

Figure 5(a) shows deadline misses to occur with the lowest allowed frame size of 256 bytes while higher frame sizes satisfy the given deadline. This is due to the earlier observation on larger latencies arising from smaller frame sizes which result in higher number of fragments. From an energy-management perspective, if the current channel conditions require frame sizes to be below 1000 bytes for higher energy efficiency, a frame size of 512 bytes needs to



**Figure 5. Illustration of end-to-end latencies from fragmentation with different fragment sizes on a 30kb data packet.**

employed for the deadline to be satisfied in the given scenario. Although a frame size of 256 bytes would achieve higher energy efficiency, it would fail to meet the end-to-end deadline thereby degrading application performance.

Multimedia systems often rely on playout buffers at the destination to smooth-out the effects of network delays and jitter on the media playback quality [3]. In cases where frame re-transmissions arrive late, thereby violating end-to-end deadline requirements, a playout buffer mechanism can still benefit from such late re-transmissions since packets are delayed and buffered before delivery to the application. The playout buffers thus allow sufficient time for re-transmissions to be requested and completed. For the streaming application discussed above, a playout buffer of length  $l$  packets would introduce a new deadline D2 and the duration in the interval D1-D2 would translate to the time allowed for re-transmissions to be completed. The length of this duration is dependent on the playout buffer size ( $l$ ) and can be adapted based on the channel error-rates and the latencies that can be tolerated by the end-user.

With re-transmission based mechanisms, smaller frame sizes incur lower overheads as a result of a) higher transmission success-rates and b) lower transmission times due to smaller size of re-transmitted data (frames). Typi-

cally, re-transmission schemes wait for a timeout interval ( $t_{to}$ ) (set to the estimated round-trip time) for frame-acknowledgments before re-transmitting un-acknowledged (lost) frames. Since the round-trip times have a uniform effect on transmissions with various frame sizes, a constant  $t_{to}$  (10ms) was assumed in our analysis. The worst-case re-transmission latency is represented as

$$L_{re-transmission} = a * (t_{frame} * n + t_{to}) \quad (9)$$

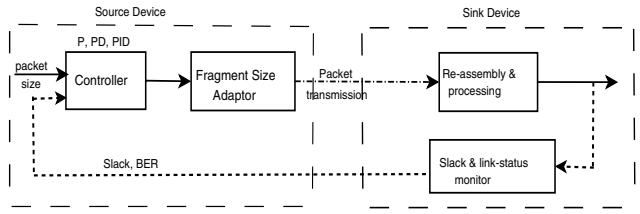
where the parameters  $n$ ,  $a$  and  $t_{frame}$  are computed using Equations 1-4 for the given fragment size  $k$ .

Figure 5(b) illustrates the timeliness of different frame sizes when the BER for the channel is given to be  $10^{-3}$ . In a worst case scenario where a fragment size of 2346 bytes is employed, each fragment would be required to be transmitted 10 times to achieve a successful transmission. Thus to allow for a re-transmissions to be received with a round-trip delay of 10ms, a deadline D2 of 100ms extending from D1 is specified. This figure shows that a frame size of 256 bytes would satisfy the playout deadline (D2) in addition to incurring lower re-transmission latencies and overheads compared to frames of larger sizes. The re-transmission latencies shown include the round-trip delays and the processing latencies. Since the acknowledgment, re-transmission and processing of frames occur in parallel to other frame transmissions and receptions, the re-transmission latencies incurred are lower compared to the latencies incurred during the first transmission attempt for a packet.

### 3.3 Feedback-based Control Mechanism

As discussed in the previous sections, fragmentation mechanisms need to carefully consider the trade-offs between latency and energy management for optimal performance and prolonged system life-time. Figure 6 outlines our dynamic fragment-size adaptation mechanism for a two node environment (our future work will extend this mechanism to multi-hop scenarios). The sink node (destination) requests and receives services (e.g., a media stream) from the source node. The term ‘slack’ ( $\tau$ ) indicates by how much the sink finishes processing a packet before the end-to-end deadline (positive slack) or by how much this deadline has been missed (negative slack). The sink monitors the conditions of its link (by tracking the number of re-transmission requests) and the slack for each received packet. This information is then fed back to the controller at the source every specified feedback interval (which can be adapted based on needs and overhead considerations).

The controller (comprised of P, PD, PI, or PID algorithms<sup>1</sup>) at the source computes an optimal fragment size for current channel conditions and the end-to-end slack ob-



**Figure 6. Controller block diagram for the adaptive fragmentation approach.**

served at the sink. The controller algorithm uses a two step mechanism in computing an optimal fragment size as explained in Figure 7. The first step involves computing a fragment size that would yield higher energy savings for the given channel conditions (BER). This corresponds to computing an energy-optimal fragment size ( $F_{E-O}$ ) that would incur the lowest energy consumption (Equation 7) for the given BER. This stage is invoked whenever there is a change in the observed BER. Based on the observed end-to-end slack, an optimal fragment size ( $F_O$ ) is then determined in the second step by appropriately adjusting the energy-optimal fragment size reported from the previous step. If the BER is unchanged from the last feedback interval, the previous selected frame size is adjusted according to the slack in computing  $F_O$ . This step thus involves the calculation of an adjustment factor ( $\Delta f$ ) based on the feedback of the observed slack. A negative slack which implies a deadline miss, will cause the controller to increase  $F_{E-O}$  (or previous  $F_O$ ) by  $\Delta f$  to avoid the higher latencies associated with smaller fragment sizes. Positive slack on the other hand would allow fragmentation of data into smaller frames by decreasing  $F_{E-O}$  accordingly with  $\Delta f$ .

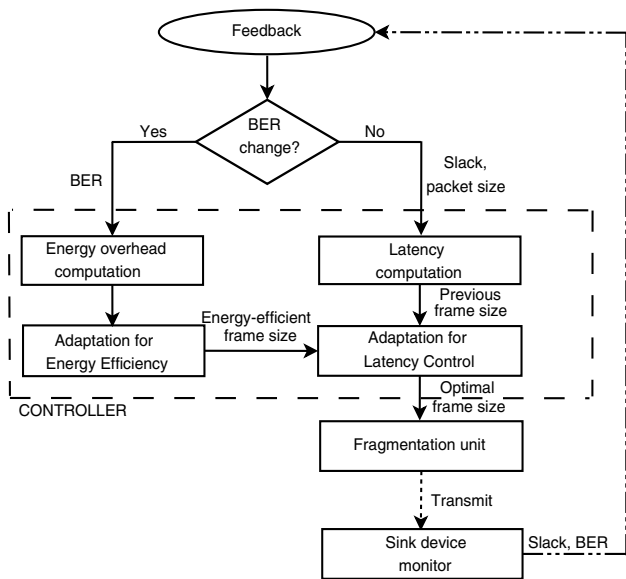
Multiple control algorithms can be used to implement the controller. As part of our experimental analysis we implemented the PD control algorithm. Since our goal is to quickly adapt to changes in the end-to-end slack, we chose to ignore control algorithms with ‘integral’ components (PI, PID) which are associated with longer response times and oscillations in controller response.

Each component (i.e., P and D) in the PD control algorithm consists of a controller gain or constant  $kp$  for the proportional component (P) and  $kd$  for the derivative component (D). The controller output can be expressed as a function of these constants. In our adaptation mechanism, the controller output  $\Delta f$  (which is used to compute  $F_O$ ) is formulated as

$$\Delta f = \begin{cases} kp_{slow} * fn(b, \tau) + kd_{slow} * \Delta\tau & \text{if } \tau \geq 0 \\ kp_{fast} * fn(b, \tau) + kd_{fast} * \Delta\tau & \text{otherwise} \end{cases}$$

The PD control algorithm computes  $\Delta f$  as a function of the controller gains ( $kp$ ,  $kd$ ), packet size ( $b$ ), the observed slack ( $\tau$ ) and, the difference between successive slack values ( $\Delta\tau$ ). The packet size is used as an input to the con-

<sup>1</sup>P = Proportional, I = Integral, D = Derivative



**Figure 7. Flowchart illustrating the two-step approach to adaptive fragmentation.**

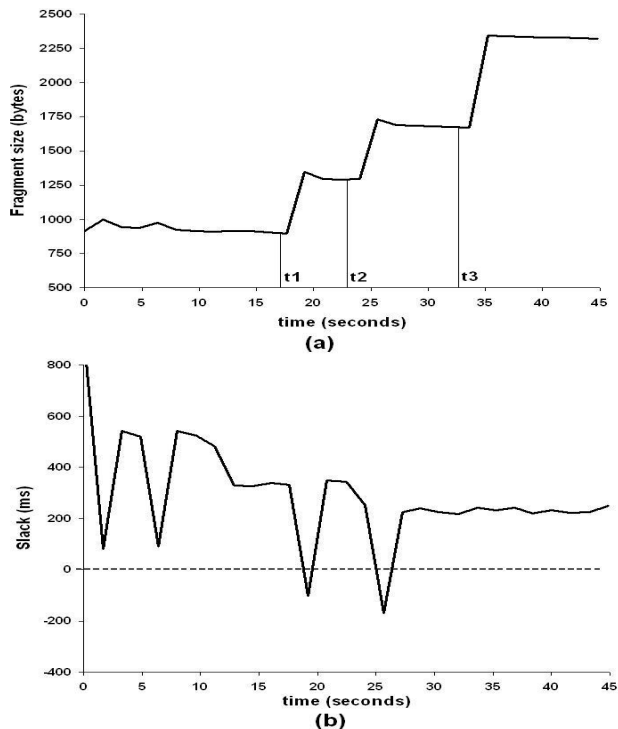
troller to model the relationship between latencies and the size of the packet. It should be noted that with a positive slack, a fragment size close to the energy-optimal fragment size needs to be employed to achieve higher energy efficiency while satisfying deadline constraints. Thus the value of the constants  $kp_{slow}$  and  $kd_{slow}$  used with positive slack are smaller than  $kp_{fast}$  and  $kd_{fast}$  which are set to a large value to quickly adapt fragment sizes and thereby reduce the introduced latencies in the event of any deadline misses.

## 4 Experimental Results

This section describes the evaluations of our proposed adaptive fragmentation mechanism with the PD control algorithm. This controller was implemented as part of a Linux kernel module with access to the wireless network interface settings. To model a streaming application, a custom-built image sharing tool which allows transfer of images using a client-server model was employed. The server (source) on a request from the client (sink), transmits images every 400ms to simulate good frame display rates in a streaming environment. The client records the slack observed for each successfully received image and sends an averaged slack to the server every 4 (image) packets (i.e., every 1600-1700ms). To simulate and vary between different channel conditions (BER), the client was set-up to drop a specified number of random packets at different intervals. Based on the number of dropped packets within a feedback interval, the BER was computed and sent as feedback along with the observed slack to the controller at the server.

The energy-overheads for different packet and fragment sizes resulting from fragmentation was modeled as an ex-

ponential relationship based on observations from Figure 2. For simplicity, a constant image size of 45kb was used.



**Figure 8. PD-Controller response with  $kp_{slow} = 0.01$ ,  $kp_{fast} = 0.05$ ,  $kd_{slow} = 0.07$ ,  $kd_{fast} = 0.5$ .**

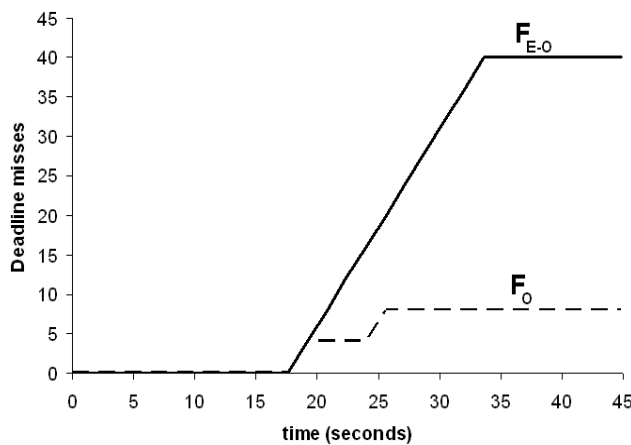
### 4.1 Controller Response

Figure 8(a) describes the fragment size adapted by the PD-controller in response to the observed end-to-end slack shown in Figure 8(b). The figure also includes adaptations in response to load disturbances and changes in the estimated channel conditions (BER). A load disturbance is introduced at time t1 (17s) by delaying the reception and processing of the packet by the application in order to simulate variations in latencies due to cross-traffic, interference. The introduced disturbance causes a deadline miss (negative slack) and the controller in response increases the fragment size accordingly to reduce the latencies incurred from fragmentation. Another deadline miss occurs when an additional disturbance is further induced at time t2 (23s) and maintained for the entire duration of the experiment. The controller again increases the fragment size to maintain the latencies within the specified constraint. The channel conditions are improved at time t3 (by decreasing the number of packets dropped intentionally) which results in a decrease in the BER (from  $10 \times 10^{-3}$  to  $5 \times 10^{-4}$ ). The controller recomputes an energy-optimal fragment size thereby allowing the mechanism to adapt to changing channel conditions as well. Our mechanism therefore achieves end-to-end latency

control while conserving energy at the source node.

## 4.2 Evaluations

Figures 9 & 10 describe the performance of the proposed mechanism (for the scenario described in Figure 8) with respect to achieving end-to-end latency management and energy efficiency. Figure 9 shows that a mechanism that selects a fragment size based only on the channel conditions for achieving energy savings ( $F_{E-O}$ ) leads to a higher number of deadline misses. The fragment size  $F_{E-O}$  is the size computed from Equation 7 in the first stage of the controller mechanism. By also including the feedback on the observed end-to-end latencies in the final selection of an optimal fragment size  $F_O$  (in the second stage of the controller mechanism), our proposed mechanism adapts to dynamic variations in the end-to-end latencies (at 17 & 23s) thereby reducing the total number of deadline misses.

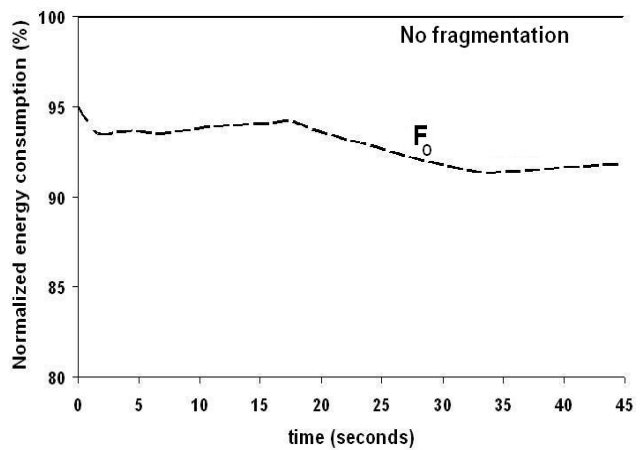


**Figure 9. Comparison of the number of deadlines missed for the energy-optimal and latency-aware fragmentation mechanisms.**

At the same time, energy efficiency is achieved in our approach by lowering the re-transmission overheads in noisy environments. By computing an energy-optimal fragment size based on BER and adjusting it depending on the observed slack ( $\tau$ ), the proposed mechanism conserves higher energy compared to a mechanism that always selects the default fragment size. Figure 10 shows the energy consumption of the two approaches normalized to the energy consumed at the default fragment size of 2346 bytes.

## 5 Conclusions

In this work, the overheads associated with the commonly adapted link layer energy management technique of dynamic fragmentation were studied extensively. Existing fragmentation approaches adapt fragment sizes ignoring the



**Figure 10. Comparison of the energy consumption for no fragmentation and  $F_O$ .**

effects of these overheads on the real-time requirements of the distributed application. Such approaches will often lead to violation of the real-time and QoS constraints resulting in performance degradation. This paper presents a novel feedback-based mechanism that dynamically adapts the fragment sizes based on the observed end-to-end latencies and channel conditions to achieve energy efficiency while satisfying the real-time constraints of the application.

## References

- [1] R. Han and D. Messerschmitt. A progressively reliable transport protocol for interactive wireless multimedia. *Multimedia Systems*, 7(2):141–156, March 1999.
- [2] IEEE Wireless Standards. <http://standards.ieee.org/wireless/overview.html#802.11>.
- [3] M. Kalman, E. Steinbach, and B. Girod. Adaptive playout for real-time media streaming. In *Proc. of IEEE International Symposium on Circuits and Systems*, May 2002.
- [4] B. S. Kim, Y. Fang, T. F. Wong, and Y. Kwon. Dynamic fragmentation scheme for rate-adaptive wireless lans. In *Proc. of IEEE Conference on Personal, Indoor and Mobile Radio Communications*, September 2003.
- [5] P. Littieri and M. B. Srivastava. Adaptive frame length control for improving wireless link throughput, range and energy efficiency. In *Proc. of IEEE Conference on Computer Communications*, March 1998.
- [6] E. Modiano. An adaptive algorithm for optimizing the packet size used in wireless ARQ protocols. *Wireless Networks*, 5:279–286, 1999.
- [7] T. Nadeem and A. Agrawala. IEEE 802.11 fragmentation-aware energy-efficient ad-hoc routing protocols. In *Proc. of IEEE International Conference on Mobile Ad-hoc and Sensor Systems*, October 2004.
- [8] N. Ramos, D. Panigrahi, and S. Dey. Energy-efficient link adaptations in IEEE 802.11b wireless lan. In *Proc. of International Conference on Wireless and Optical Communications*, July 2003.