Energy-Efficient Frame Dropping Policies for Multimedia

Robin Snader, Albert F. Harris III, and Robin Kravets

{rsnader2,aharris,rhk}@cs.uiuc.edu

University of Illinois at Urbana-Champaign

Introduction

Traditionally, multimedia applications have been primarily constrained by a lack of bandwidth. In mobile systems, however, energy constraints must be considered as well. While these constraints are related, they differ in one key way: bandwidth is renewable, but energy is not. This implies that techniques for managing bandwidth, typically performed at the transport layer, must be reconsidered in terms of their energy efficiency as well. Traditionally, these transport layers are evaluated in terms of *application goodput*, or the amount of useful data received by the application. We propose that they should also be evaluated in terms of the total amount of data sent per useful data unit received at the application, which we term *data expansion*.

In light of this observation, we have designed and implemented a new multimedia transport layer, which we call Reaper, to meet application-level reliability requirements while minimizing data expansion and therefore energy consumption. Any frame that is transmitted by the sender that cannot be used by the receiver (*e.g.*, because it arrived late) constitutes energy wasted by the transport layer. This wasted energy includes even successfully received fragments of those frames which arrive incomplete or late. Given these observations on the relationship between data expansion and energy, our research investigates the use of intelligent frame-dropping policies to conserve the energy associated with frames that would not be useful.

On the Tolerance of Losses: When a Frame Isn't Worth Saving

There are two unique characteristics of multimedia data that make retransmission-based loss recovery challenging. First since multimedia applications have strict timing requirements, frames that arrive past their deadlines waste energy. Early identification of these frames can save both energy and bandwidth. Second, multimedia applications can tolerate a low loss rate. This opens the possibility of opportunistically suspending loss recovery to save energy.

Since application data units (*i.e.*, frames) are often much larger than the maximum packet size of the network, a transport protocol must divide them into fragments. 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. In this section, we examine three scenarios where a multimedia transport layer might drop a frame, and the subsequent effects on energy consumption.

Reactive Dropping

Reactive methods leverage the fact that the receiver knows the deadline of a frame and can therefore tell if that deadline has passed during its reception. The receiver can then signal the sender to stop sending any further fragments of that frame [3]. 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 by preventing the sender from transmitting useless fragments depends on the frame size, the bandwidth, and latency of the link. If all of the fragments of a frame are in flight before the drop signal is received by the sender, this mechanism results in no energy savings. Thus, reactive dropping is most effective in low-latency networks, and with a large playout buffer. This mechanism, together with its concomitant limitations, has been well studied.

Predictive Dropping

In contrast to reactive dropping, which is initiated by the receiver, predictive dropping is initiated and controlled by the sender. To do this, the transport layer at the sending side estimates the deadline of the frame associated with each fragment it sends. Using its knowledge of network conditions, the sender can then predict whether a fragment will arrive at the receiver in time to be useful. Since a frame only partially received by its deadline lacks even partial utility, however, this prediction must be based not on the current fragment, but the last fragment of the associated frame in order to be accurate. Since channel characteristics vary unpredictably with time, such a prediction is inherently inaccurate; however, it has the potential to save much more energy than reactive methods. For details of how Reaper controls the optimism of the prediction, see [1]. To the best of our knowledge, using cross-layer information to save energy in this way has not been previously studied.

Opportunistic Dropping

It is obvious that suspending loss recovery for frames that are already late or predicted to be late can save energy and bandwidth. However, it is also possible to use information about current performance (*i.e* the current on-time delivery rate) to suspend loss recovery even if the frame might arrive on time, improving energy efficiency and successful frame delivery rate[2]. This approach has two effects. First, energy that would have been used in the retransmission of the lost fragments is saved. Second, the stream can catch up in time by moving on to the next frame, possibly preventing subsequent frames from being late. For the details of how Reaper implements opportunistic dropping, see [1]. Conserving energy by taking advantage of variable reliability has been studied previously, but not, as far as we are aware, in this context.

Evaluation and Analysis

To validate and evaluate this combination of novel and previously known techniques, we implemented them as a user-space library under Linux, which we called Reaper. This implementation of Reaper takes data from applications and sends it over UDP, with the timing of TCP New Reno [5] and the reliability described as above. To evaluate Reaper's performance, we use two metrics. First, the transport layer must deliver enough frames to match the application's loss tolerance level. To measure this, we use the number of frames received on time by the application before their playout deadlines. Second, the data expansion caused by the transport layer must be kept to a minimum to increase energy efficiency. Using these metrics, we show that a coordinated combination of the mechanisms presented above results in gains in both energy efficiency and the application goodput over using other subsets of the mechanisms.

Performance Analysis

To understand the impact of combining the mechanisms used by Reaper, the we present goodput and energy analysis for six protocols: Reaper, Reaper without opportunistic drops, TCP with reactive drops (with and without opportunistic drops), early drop, and TCP. For the details of the protocols tested, and the exact testing scenario, see [1].

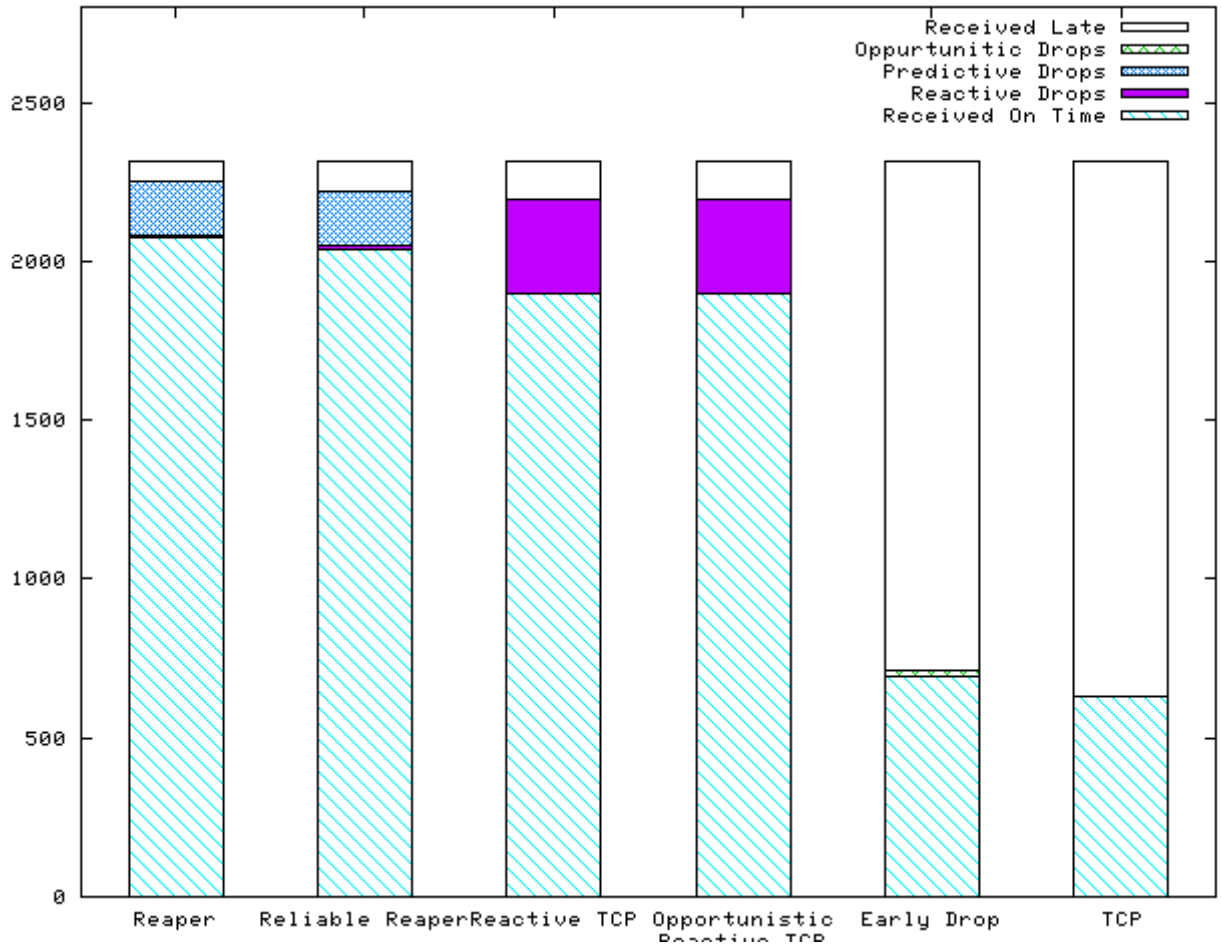

Figure 1: Frames Attempted and Their Disposition

We first evaluate the application goodput for each protocol. As seen in Figure 1, TCP and early drop perform poorly. This is due to their inability to tell when they are falling behind-once they start missing deadlines, they never catch back up. Reaper and the reactive TCP variants do much better (with Reaper performing the best), since they can skip frames to catch up when they fall behind.

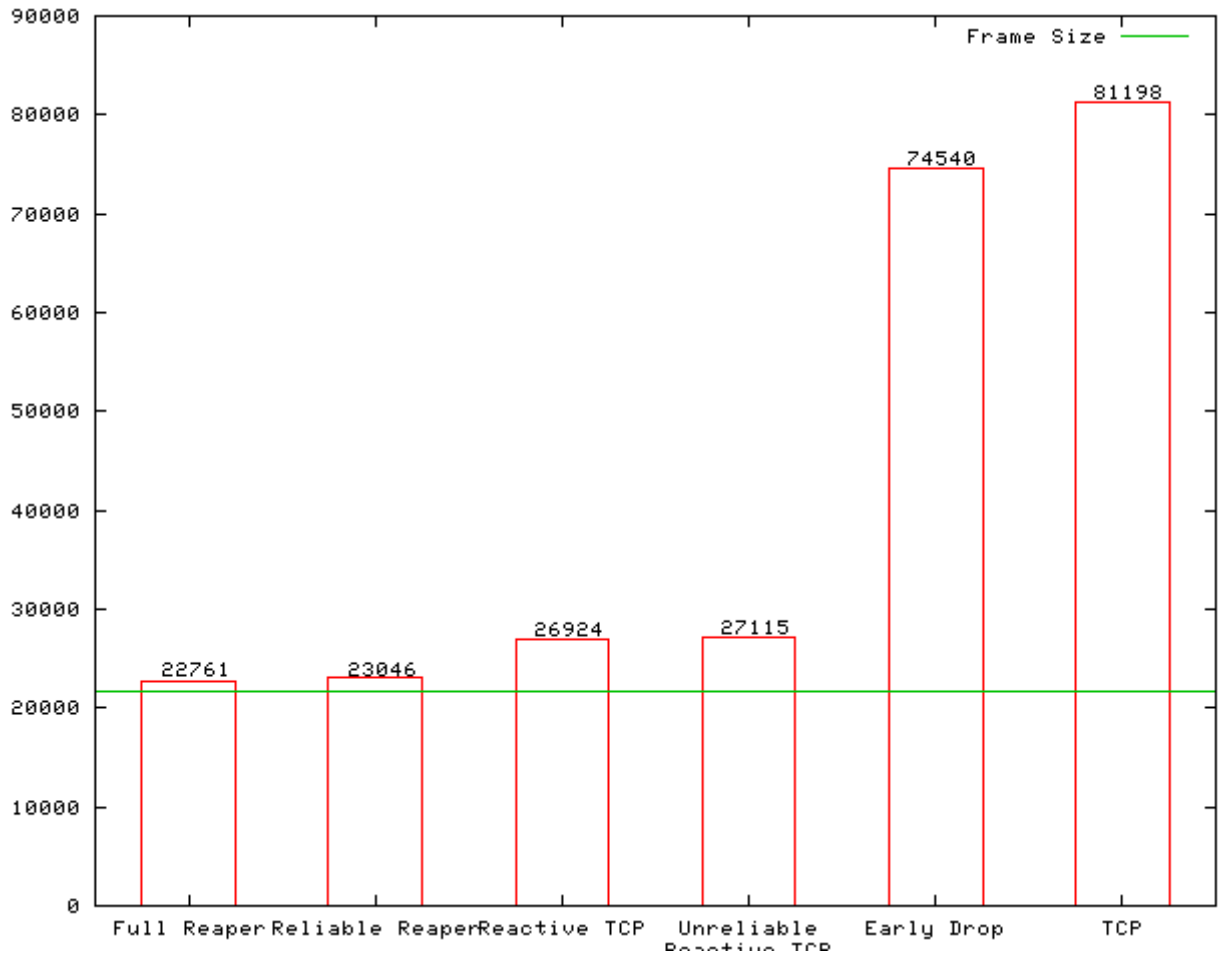

Figure 2: Bytes per Good Frame

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. Figure 2 depicts the average bytes sent per good frame for each of the protocols. The horizontal line in the graph represents the average number of bytes per frame, and the space above it represents the average data expansion per frame. As expected, Reaper has the lowest data expansion factor and therefore the best energy efficiency of all protocols tested.

Conclusions & Future Directions

This research examined techniques, both new and previously known, to maximize application goodput and minimize data expansion by controlling the reliability of the transport layer. We showed that our combination of techniques performs better than any subset via experimentation.

As future work, it would be interesting to explore the impact of alternative loss-notification algorithms, such as ELN [6] and SACK [4]. Such algorithms provide more feedback to the transport layer, allowing more intelligent loss-recovery decisions as well as more accurate feedback to the application. Additionally, it would be interesting to examine ways to dynamically set and tune the parameters controlling Reaper's exact operation[1].

Bibliography

1

A. F. Harris III, R. Snader, and R. Kravets.
On energy, adaptation, and the death of frames.
Technical Report UIUCDCS-R-2006-2744, UIUC Dept. of Comp. Sci., June 2006.
`http://mobius.cs.uiuc.edu/~rsnader2/pubs/reaper.ps`.

2

R. Kravets, K. Calvert, P. Krishnan, and K. Schwan.
Adaptive variation of reliability.
In *HPN 1997*, 1997.

3

R. Sinha and C. Papadopoulos.
An adaptive multiple retransmission technique.
In *NOSSDAV*, 2004.

4

IEEE Computer Society.
TCP selective acknowledgement options, RFC 2018, October 1996.

5

IEEE Computer Society.
TCP RFC 2581, April 1999.

6

IEEE Computer Society.
The addition of explicit congestion notification (ECN) to IP RFC 3168, September 2001.