Loss Profiles at the Link Layer

Kevin Brown

Department of Computer Science
University of South Carolina
Columbia, SC 29208

Suresh Singh

Department of Computer Science
Department of South Carolina
Columbia, SC 29208

Abstract

In an earlier paper [3] we proposed the implementation of Loss Profiles (a new QoS measure for Mobile Computing) as a sub-layer at the transport layer. In this paper we examine the possibility of implementing a limited form of Loss Profiles at the link layer.

1 Introduction

In [3] we identified a new QOS parameter for the mobile environment, called *loss profiles*, that ensures *graceful degradation* of service (for applications that can tolerate loss) in situations where user demands exceed the network's capacity to satisfy them. For example, consider a situation where a mobile user, who has opened a data connection with a bandwidth requirement of 32Kbps, finds herself in a cell (in the course of roaming) where the available bandwidth is only 24Kbps. It is easy to see that such a situation can develop quite easily because the mobile network can exercise no control on the movement of users. There is thus nothing to prevent several users with open connections from congregating within the same cell.

How can the network deal with such situations? Our proposal, discussed in our earlier paper [3], is to implement a sub-layer, sitting on top of UDP and CM (Continuous Media) protocols that enables the sender to put *flags* within the data stream. These flags delineate *logical data segments* that can be discarded (by an intermediate node) in the event of a bandwidth crunch. Thus, if the data stream consists of JPEG video, each data segment may represent one compressed frame. If the data stream consists of MPEG frames, a data segment may represent an I, B or P frame (Index, Bi-directionally coded and Predicted frames respectively). The logic behind inserting such flags is that the mobile network knows which parts of the data stream may be discarded *together*. Thus, if the data stream consists of compressed JPEG frames, it is not a good idea to discard half of one compressed frame and half of the next frame. This is because neither frame can be reconstructed at the destination resulting in two lost frames as opposed to only one lost frame if data between two consecutive flags was discarded instead.

Why do we insist on requiring the sender to insert flags within the data stream (via calls to the LPTSL – Loss Profile Transport Sub-Layer)? Why not, instead, require the mobile user to renegotiate the connection bandwidth with the sender? The reasons for using our approach are:

- If the data being transmitted by the sender is *multicast* to several receivers (mobile or stationary), then it is inappropriate for the sender to reduce the quality of the connection for *all* receivers just because one receiver finds herself in a congested cell.

- Periods where a user finds herself in a congested cell may only last a short time (if the user is actively roaming, she will soon wander out of the current congested cell). Thus, renegotiating the connection bandwidth ev-

ery few seconds (if the cell latency is 20–30 seconds, as in the case of a picocellular environment) may be expensive (protocol overhead as well as application overhead).

2 Why a new proposal for loss profiles?

Our solution (of implementing LPTSL) works well for most applications we have studied. However, the solution is tied-in quite closely with our three-layer mobile network architecture. To summarize, we consider our mobile network as being made up of three layers – at the lowest layer are the Mobile Hosts (MHs) who communicate with Mobile Support Stations (MSSs) over the wireless link. The MSSs are simple devices that operate at the data link and network layer only. Several MSSs are controlled by a gateway machine called the Supervisor Host (SH). The SH provides connectivity to the Internet (or other wired networks). All data connections between a MH and other hosts are split in two – one from the MH to the SH and another from the SH to the other connection endpoint. The reason for doing this is to ensure efficient implementation of transport protocols and to shield the fixed network from the idiosyncrasies of the mobile network. A complete protocol stack specification for the mobile network may be found in [2]. Observe that implementations such as I-TCP[1] and MTCP[4] also split the TCP connection in two. But do so at either the MSS or at some node 'close' to the MH. Our architecture generalizes this idea and enables the SH to exercise greater control on how data is transmitted to the MH. For example, if a MH has several open data connections, the SH can intelligently multiplex data for the various open connections to meet the negotiated QoS parameters for each connection. As the MH moves from cell to cell, the SH can keep track of losses seen, jitter experienced, etc. for all the MH's connections and ensure the the QoS parameters are met over the entire lifetime of the MH's connections.

While we believe that our architecture is very well suited for implementing mobile networks of the future, we do have to be realistic and assume that not all deployments of such networks will follow our specification! For that reason, we propose a version of loss profiles that can be implemented by the base stations (MSSs in our architecture) and, perhaps, by an ad hoc node 'close' to the MH.

3 Loss Profiles at the MSS

The central idea is to require, as in [3], that the sender insert flags within the data stream to delineate logical data segments. However, these flags are not interpreted by the MSS. Rather, the LPTSL layer at the MH keeps track of losses and requests retransmissions of partial data segments from the MSS since part of a data segment may have been discarded by the MSS prior to transmission or because that part of the data segment may have beed transmitted while the MH was in a fade. Note that the MH may have to sacrifice some other portion of its data stream in order to receive the retransmission. To explain this approach better, let us define the function of each component involved in the data transfer.

- The *sender* applications makes calls to LPTSL, rather than to the transport layer protocol such as UDP, whenever it needs to write data to a connection. Each call represents one logical data segment. LPTSL at the sender inserts flags into the data stream being transmitted. These flags may contain information such as priority of the segment (e.g., I frames in MPEG are very important because losing one will cause 12 consecutive frames to be lost) and other information.

- The data stream is transmitted to the MSS (Mobile IP or some other protocol ensures that the data stream arrives at the correct MSS).

- The *MSS* has allocated some bandwidth to the MH. Let us assume that the bandwidth allocated is B_a while the bandwidth requested is B_r. If $B_r \leq B_a$, unless the MH goes into a fade, it is unlikely that any data will be lost. On the other hand, if $B_r > B_a$,

($B_r - B_a$) bytes/sec of the data stream will have to be discarded prior to transmission to the MH. Let us consider two models of the MSS:

- *A dumb MSS:* The MSS maintains a byte-oriented FIFO buffer for the MH's connection. When the buffer overflows (this happens if $B_r > B_a$ or if $B_r = B_a$ but the MH requested a retransmission of some bytes lost during a fade), new data is discarded.
- *A smart MSS:* The MSS is aware of B_r and B_a. It discards data from the connection using some pre-defined algorithm. For instance, the algorithm may require that the MSS discard $(B_r - B_a)T$ bytes of data from the end of the FIFO buffer every T seconds. Or, another algorithm may be to discard this amount of data randomly uniformly. *The policy used is in no way connected to the requirements of the data connection itself, however.* For example, for an audio connection, a random uniform loss would be preferred, by the user, to a clustered loss at the end of every T seconds. While the latter may be appropriate for a video connection (see the user studies reported in [3]). However, we assume that the MSS does not know about the application and implements a loss function built into its design by the manufacturer.

- The LPTSL at the *MH* makes no assumptions about the MSS itself(smart or dumb) and instead keeps track of the losses as they occur in the data stream. We assume that the data stream is treated as a byte stream and each flag (that precedes a segment) contains the starting and ending byte number of data in that segment. This information is sufficient to determine which segments have suffered partial data loss.

  Assume that the application running at the MH has a predefined acceptable loss profile specified as a library function.

  - In [3] we propose that loss profiles, such as uniform loss, clustered loss, etc., be parameterized as function calls. A simple example is a function that implements uniform loss. This function takes as input the bandwidth, percentage of loss desired and a time interval over which this loss must be enforced (giving a time interval of, say, 2 seconds, ensures that the appropriate loss percentage is achieved every 2 seconds) and outputs an acceptable loss behavior (e.g., bytes 200–400, bytes 1000–1300, bytes 3000–3100 may be lost if necessary).

  LPTSL examines the output of this function and may, if necessary, request retransmission of some bytes from the MSS in order to match the loss profile.

  - A more complex strategy may be the following. If the data stream is CBR, LPTSL at the MH can easily *predict* the future composition of the byte stream for some applications and deliberately *sacrifice* some portions of this stream by sending control messages to the MSS to discard portions of its buffer. This ensures that other portions of the buffer are not discarded by the MSS. This last proposal may or may not be feasible depending on the complexity of the MSS and the existence of a protocol between MHs and the MSS that allows the MHs to control the buffer of the MSS.

  Thus, LPTSL at the MH ensures that the loss profile achieved matches the application's needs (and does so without intervention from the SH, as in our earlier paper). The request for retransmission is transmitted over the wireless link and consumes scarce

bandwidth but, we believe, it is small enough to be justified.

4 Conclusions

We have proposed an alternative approach to implementing Loss Profiles in mobile networks. This new proposal does not depend on a three-layer mobile network hierarchy. This will ensure that Loss Profiles can be implemented in most mobile networks of the future.

Acknowledgments

This work was supported by the NSF under grant number NCR-9410357.

References

[1] A. Barke and B. R. Badrinath. I-TCP: Indirect TCP for Mobile Hosts. *Technical Report DCS-TR-314, Rutgers University*, October 1994.

[2] K. Brown and S. Singh. A Network Architecture for Mobile Computing. *IEEE INFOCOM, March 24-28*, pages 1388–1396, 1996.

[3] K. Seal and S. Singh. Loss Profiles: A Quality of Service Measure in Mobile Computing. *ACM Wireless Networks*, 2(1):45–61, 1996.

[4] R. Yavatkar and N. Bhagawat. Improving end-to-end performance of TCP over Mobile Internetworks. *IEEE 1994 Workshop on Mobile Computing Systems and Applications, Santa Cruz, CA*, December 1994.