Network-aware Application Adaptation for Mobile Hosts

Benjamin Atkin and Kenneth P. Birman
Department of Computer Science, Cornell University, Ithaca, NY
{batkin,ken}@cs.cornell.edu

Abstract

Mobile hosts in a wireless network can experience highly variable network performance. Adapting to network conditions can help mask this variability. For instance, a client-server application can defer inessential work and otherwise reduce communication when the quality of connection to the server is poor. This paper describes an approach to application adaptation which differs from the usual mode-based technique. It also discusses two implementations of modeless adaptation: ATP, a library for bandwidth-aware communication, and MFS, a cache manager for a distributed file system.

1 Introduction

One of the central problems in writing software to run on mobile hosts is variability in network performance. A mobile host using a wireless network can experience rapid and large-scale changes in bandwidth availability. If an application does not adapt its network communication to these changes, they can interfere with the performance seen by the user. Prior work in adapting applications to perform well on mobile or bandwidth-constrained hosts has included adaptation in file systems [6] and web browsing [3], and more general approaches for client-server systems [4] and resource-constrained applications [5].

These systems have chiefly employed what we term *modal adaptation*, in which an application has a (usually small) number of modes of operation, each with a particular bandwidth utilisation, and chooses a mode depending on the available bandwidth. For instance, a Coda file system client changes its writeback policy depending on whether it is weakly or strongly connected to the server, and a web browser might fetch degraded images from a web server in place of the highest-quality images [3].

However, modal adaptation is not always suitable. A high degree of bandwidth variation may result in rapid changes between modes, or the communication exhibited by the application may not naturally fit a coarse-grained modal division. Such a *modeless* application might send and receive multiple types of data of varying degrees of importance, and allow fine-grained adaptation to bandwidth variation. For instance, a distributed file system client could concurrently fetch files, write back shared files, write back "private" files, and prefetch files. Some of these tasks can be pushed into the background, and some serve only to improve performance, rather than affecting the semantics of the file system.

The remainder of this paper discusses ATP, a library which supports modeless adaptation through prioritised communication, and MFS, a cache manager for a distributed file system, which uses ATP in combination with file system-specific techniques to adapt itself to bandwidth availability.

2 Communication adaptation

Our initial work [1] investigated the usefulness of a Remote Procedure Call (RPC) library supporting multiple priority levels as a tool for adapting communication to network variations. This led us to develop ATP, a message-oriented network protocol which runs over kernel UDP, and does its own fragmentation and retransmissions to ensure reliable delivery. Messages can be marked with a priority and a send timeout to control how they are sent: ATP delivers queued messages in priority order and notifies the sender if a message cannot be sent before its timeout expires. We have compared the performance of ATP to various TCP-based approaches [1], and while TCP has been highly-tuned to operate well in wireless networks [2], its bytestream-oriented interface, and the uncontrollable contention for bandwidth which can arise between multiple streams, make ATP preferable for many workloads we have measured.

ATP allows either explicit or implicit adjustments to bandwidth changes. Timeouts on send operations allow an application to receive an explicit notification when there is insufficient bandwidth to transmit a message. In

The authors were supported in part by DARPA under AFRL grant RADC F30602-99-1-0532, and by AFOSR under MURI grant F49620-02-1-0233.

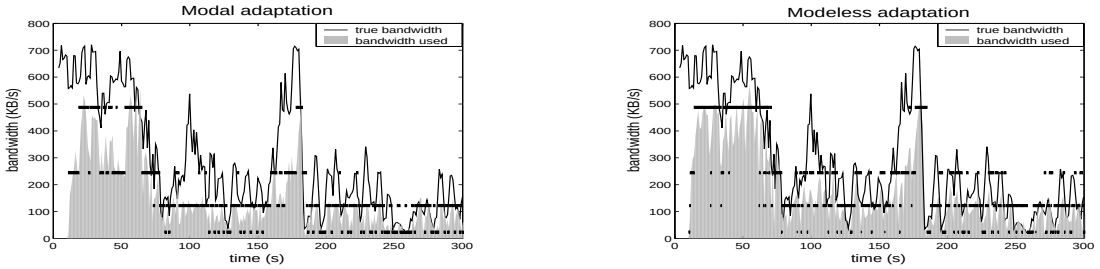

Figure 1: Modal versus modeless adaptation with ATP. *The dark horizontal lines in the graphs indicate operating modes; the left graph shows performance with modal adaptation, and the right graph shows a scheme in which there are four classes of messages being sent simultaneously, of increasing priorities (the lowest line corresponds to the highest priority). The modeless scheme achieves higher utilisation (48.5 MB of data sent) because it always has messages to send, while the modal scheme is dependent on a rapid and accurate estimate of the available bandwidth in order to select its correct operating mode (41.5 MB sent). These graphs are reproduced from [1].*

contrast, priorities allow an application to make use of *speculative communication*, in which low-priority messages can be scheduled for transmission, without interfering with the transmission of high-priority messages when bandwidth is low. Experiments have indicated that making use of priorities in ATP can allow a more efficient use of bandwidth than is possible with a modal adaptation scheme, as Figure 1 illustrates.

3 File system adaptation

After implementing ATP, we have continued by experimenting with modeless adaptation closer to the application level. Mobile file systems are well-understood and have been extensively studied [6], but even so, the caching techniques used in file systems are broadly applicable, and most existing work in mobile file systems has focused on modal adaptation.

We are implementing and evaluating MFS, a flexible cache manager for a mobile file system client. MFS uses a fetch-on-read, delayed-writeback caching scheme, but differs from previous approaches in two ways: first, it employs RPC prioritisation to adapt to changing bandwidth, and second, it makes aggressive use of file access information to improve its caching strategy.

RPC prioritisation allows the cache manager to execute RPCs for background tasks, such as prefetching and file writeback, without interfering with file validations and fetches to satisfy cache misses. File access statistics, such as associativity groups for files which are commonly co-accessed, and whether files are shared between clients or are private, affect the order in which files are prefetched and written back. If bandwidth is low, it may be acceptable that accesses to some types of files (for instance, executable files) may be serviced using a stale version in the cache, rather than initiating a fetch which will block for a long period. Priorities allow MFS to initiate RPCs to write back a shared file and a non-shared file concurrently, knowing that the non-shared file will only use "surplus" bandwidth: shared files have higher priority because they are accessed by other clients as well as the modifying client.

MFS's performance is being evaluated with a variety of workloads and bandwidth traces. Our initial experiments have demonstrated that prioritised communication can substantially outperform a non-adaptive distributed file system when bandwidth is variable, by prioritising cache validations over writeback traffic. Further experiments will compare MFS to modal schemes incorporating logging and delayed writeback.

References

[1] B. Atkin and K. P. Birman. Evaluation of an adaptive transport protocol. In *Proceedings of the 22nd Annual Joint Conference of the IEEE Computer and Communications Societies (Infocom 2003)*, Apr. 2003.

[2] H. Elaarag. Improving TCP performance over mobile networks. *ACM Computing Surveys*, 34(3):357–374, Sept. 2002.

[3] A. Fox, S. D. Gribble, E. A. Brewer, and E. Amir. Adapting to network and client variation via on-demand dynamic distillation. In *Proceedings of the Seventh International Conference on Architectural Support for Programming Languages and Operating Systems*, pages 160–170, Cambridge, Massachusetts, Oct. 1996.

[4] A. D. Joseph, J. A. Tauber, and M. F. Kaashoek. Mobile computing with the Rover Toolkit. *IEEE Transactions on Computers: Special issue on Mobile Computing*, 46(3):337–352, Mar. 1997.

[5] B. D. Noble and M. Satyanarayanan. Experience with adaptive mobile applications in Odyssey. *Mobile Networks and Applications*, 4(4), 1999.

[6] M. Satyanarayanan. The evolution of Coda. *ACM Transactions on Computer Systems*, 20(2):85–124, May 2002.