

Wireless Application and API Design *

Michael Nidd, Thomas Kunz, James P. Black
{menidd,tkunz,jpblack}@uwaterloo.ca
University of Waterloo
Waterloo, Ontario, Canada

February 13, 1996

1. Abstract

The problems of maintaining a real-time connection in a high-speed network and maintaining reasonable response time in a slower network are closely related. In this position paper, we outline our plan to extend the existing continuous multimedia QoS research into design techniques that minimize user frustration on a wireless platform. This plan involves the development of reactive applications that can adapt to existing network performance, and an API that can support them.

2. Background

The transmission of data across a wireless link can be compared to the transmission of continuous multimedia data across a high-speed link. In continuous multimedia, data that arrives late is of very limited (if any) use; in wireless computing, data that arrives slowly frustrates the user. For continuous multimedia, significant research effort [1, 2, 3] has been expended to find ways to adjust network, transport, or application behaviour to maintain timely connections for as many users as possible. The users of wireless terminals also have a need for the timely delivery of information.

The goal in providing this timely delivery is to minimize user frustration. It is not sufficient to minimize the total execution time, unless the application can also present the appearance of progress. Thus, although the wireless data transfer might not be time-critical in the traditional sense, the effort to minimize user frustration level will require many applications to adapt their behaviour to get something on the screen quickly.

As an example of similar application requirements in the two areas, consider first an audio-video channel operating over a fast network. The application managing this channel can adapt the service it presents to the provided network service by performing in one of three states: colour, black & white, or audio only. This allows for application-level techniques for dealing with network problems. For example:

1. As the channel error rate increases, part of the information (e.g. colour) is dropped, and the freed bandwidth can be used for longer error correcting codes.
2. As congestion increases, the service level can be dropped without changing error control, helping to ease the channel load.
3. As the connection becomes better, the service level can be increased.

This responsiveness to connection quality allows the application to act on the understanding that for a particular case, such as video-phone, a clear audio channel with no video is preferable to poor quality audio and static-filled video.

An example of static data transfer that also requires channel maintenance is an HTML browser. On discovering that the channel error rate has significantly increased, the application might have the following options:

*This work was supported in part by the Advanced Radiodata Research Center of Motorola Canada Ltd.

1. Accept the higher error rate, perhaps displaying a user dialogue box containing an alert message, and providing an opportunity to abort the current action.
2. Adjust error correction codes to balance transmission speed with the retransmission rate to attain optimal throughput.
3. Transmit a summary of the remaining data, such as a black & white version of a picture, or just the headings of a text document. Once the summary has been received and displayed, the actual transmission can resume.

For both of the above examples, assuming that the transport layer handles all data equally, the adaptation must be initiated by the application.

The requirement for application awareness of connection quality is therefore a common link between these two classes of network applications, but before continuous multimedia research can be interpreted in the context of static data over wireless connections, the differences should also be recognised. For instance, there is no upper limit on the acceptable transmission rate; while audio should not usually be played back faster than it was recorded, unexpectedly fast transfers of static data are not dangerous. Jitter is similarly less significant.

3. Research Plan

In view of the above, it is our intent to consider the views of published research in continuous media transfer over networks. The specific parts of this work which are of interest are the application programming interface and the connection management techniques. There are three main questions to be answered about application programming interface:

1. How often should the application be notified of changes in connection characteristics?
2. What information does the application require?
3. How should this information be conveyed to the application?

The two classes of solution to the first problem can be referred to as “polling” solutions, and “call-back” solutions. The argument for polling is that the program can make more efficient decisions about when a status update is required, or when it will make no difference to the application behaviour. It is our belief, however, that this is more than the application designer should have to consider. Applications should provide acceptable bounds on network quality, and have call-back functions for handling violations of those bounds. Call-back functions can also be used for asynchronous network events, such as SNMP traps. Polling can be simulated by having the call-back functions set flags or other local variables. This is similar to the style of solution being pursued at Rutgers, with their “OBJECTS” [4] research.

The second problem, what information an application requires, might have a very simple answer. At the moment, we suspect that average throughput and error rate are sufficient by themselves. In initial implementations, we will make other values (all the SNMP MIB-II interface group objects) available to programmers examining connection management schemes. The value of this information will be assessed through its apparent usefulness in connection management experiments.

The problem of how this information should be conveyed is probably not of much theoretical interest, but it will have practical implications. If the reason that this information is required is to provide a constant feeling of speedy response, it would be particularly bad for the gathering of the information to significantly slow the program. For this reason, we imagine that the data should be passed by reference to the user program by light-weight threads that do the actual gathering. This should speed context switches, and save communication overhead.

Having designed the API, we are still faced with the problem of what to do with it. Its *raison d'être* is to allow applications to manage their own communications by making available as much information about lower layer performance as possible. Effective ways to use this information are a subject for further research. Early study will focus on varying packet size, trading header-data overhead against the all-or-nothing problems with packet data; and also on ways to automatically summarize information, transmitting a useful subset (or “abstract”) of the data requested, sometimes to be followed by the the full contents.

As a move towards general adaptation of applications, we will also develop a generic application-layer package to work with the API. This package will allow the network parameter tolerances, and the response to violations, to be adjusted at run-time. This interface can expose all the parameters available from the API, and offer general responses (e.g. level of error control, window size, packet size, information dialogue boxes, etc.) We believe that, for many applications, this modification alone will improve usability in a wireless environment.

References

- [1] A. Campbell, G. Coulson, F. García, and D. Hutchison. A continuous media transport and orchestration service. In *ACM SIGCOMM '92*, August 1992.
- [2] R. Chipalkatti, J. F. Kurose, and D. Towsley. Scheduling policies for real-time and non-real-time traffic in a statistical multiplexer. In *IEEE INFOCOM'89*, pages 774–783, April 1989.
- [3] Domenico Ferrari. Client requirements for real-time communication services. *IEEE Communications Magazine*, pages 65–72, November 1990.
- [4] Girish Welling and B. R. Badrinath. Mobjects: Programming support for environment directed application policies in mobile computing. In *ECOOP'95 Workshop on Mobility and Replication*, August 1995.

Contact author: Michael Nidd
University of Waterloo,
Waterloo, Ontario, Canada, N2L 3G1
Phone (519) 888-4567 ext. 5345
Fax (519) 885-1208
menidd@ccnga.uwaterloo.ca