DYNAMIC ADAPTATION OF NETWORK CONNECTIONS in Mobile Environments

Most network applications assume continuous connectivity—an assumption that does not "migrate" to wireless environments. The authors present the design of a communication layer for mobile computing that dynamically adapts to changes in network connections.

JØRGEN S. HANSEN, TORBEN REICH, BIRGER ANDERSEN, AND ERIC JUL
University of Copenhagen



Software engineers who want to migrate existing applications from fully connected environments to mobile environments face a significant problem in obtaining and maintaining network connectivity. In fully connected environments, connectivity is static and permanently established through wired networks. Applications simply "assume" that they are online continuously. This assumption is clearly invalid in mobile environments that use wireless connections. While decreases in computer size and cost have made it more practical to use a portable computer as a mobile workstation, and advances in wireless technologies have made it possible to access networks outside the reach of wired connections, the development of software for use in a mobile environment has lagged behind these technologies. In general, support for mobile computing is poorly integrated in operating systems and network software.

In this article, we discuss some of these problems. We then describe a wireless design of a communication layer for mobile computing and a prototype based on that design. Our work was part of AMIGOS (Advanced Mobile Integration in General Operating Systems), a col-

1089-7801/98/$10.00 © 1998 IEEE

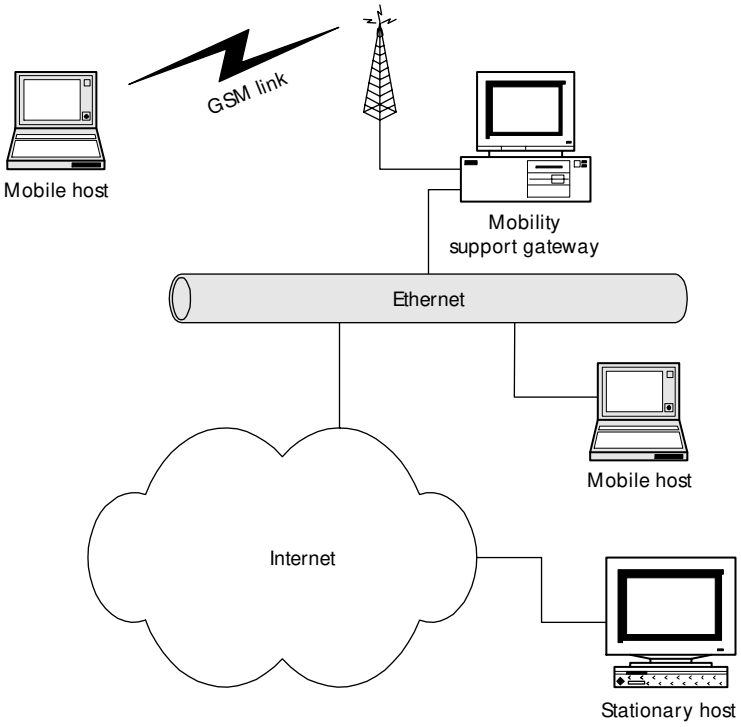

Figure 1. Diagram of a mobile system.

laboration between researchers at the University of Copenhagen in Denmark and the University of Minho in Portugal. The AMIGOS project provides transparent support for semiconnected operations on mobile computers running a standard operating system; the project home page is at http://www.econ.cbs.dk/people/birger/AMIGOS/.

DESIGN OVERVIEW

Briefly, our design lets a mobile user connect a mobile host to a LAN, then disconnect the host from it. The user then can reconnect, for example, via a Global System for Mobile Communications (GSM) cellular modem without losing TCP/IP connections. We want to allow existing TCP/IP-based applications to be used in a mobile environment, without application modifications. The assumption we make here is that a user wishes to maintain network connectivity of, for example, TCP sessions, despite the intermittent disconnect that slow, less reliable serial links offer. Thus we wish to mask the less-than-ideal connectivity provided especially by wireless links and to allow switches between different links. Such links, and switching between them, may become more common as new types of wireless communication systems, such as low-orbiting satellites, are deployed.

Our goal is to keep alive, for example, TCP/IP connects that would otherwise die for those applications that can tolerate longer delays in communication. An alternative would be to modify TCP/IP directly; our approach allows TCP/IP to be used unchanged. Users can work with their laptops virtually continuously. At the office, the laptop connects to the office LAN via a PC card LAN adapter. After work, users board their train or climb into their car, pop in a GSM PC card, and continue working. (This applies to most applications; obvious exceptions include backup or printer spooling, which are a waste of the expensive GSM phone connection.) Once at home, the user can pop out the GSM PC card and replace it with a PC card modem, still without service interruptions. However, because the phone line is cheaper and faster than the cellular connection, printer spooling would be allowed although backup would not be.

Our design addresses mobile systems that consist of a number of mobile hosts (laptops) and at least one stationary host, which provides support for the mobile hosts. Figure 1 shows the stationary support host, or *mobility support gateway*. A mobile host may either connect by dial-up serial links, via the gateway (cellular GSM or ordinary telephone links), or directly connect to a LAN. We assume that normal use of the mobile host would be as a workstation but that it can sometimes be mobile.

To let the mobile host move freely, our communication layer hides the complexity of using and switching between different means of communication. It also supports disconnected operation and makes packet scheduling possible on low-bandwidth communication channels. The communication layer thus adapts to different underlying media. Because our goal is to support existing applications that are migrated to mobile environments, we have limited our work to TCP/IP-based applications.

WIRELESS TCP PROBLEMS

Other research has used TCP/IP atop wireless links to provide mobile connections. A key problem is the generally inferior quality of wireless links to that of wired links, resulting in higher packet loss probability.

Packet Loss and Congestion

Wireless network connections for mobile hosts let the mobile host move freely. Because wireless network quality varies considerably over time, however, protocols supporting reliable data transport, such as TCP, may experience a performance loss. When used on links with drop-outs and periods of heavy noise, TCP interprets the cause of packet loss as network congestion and initiates congestion control. This, together with TCP's retransmission policy, can seriously degrade throughput.[1,2]

The data service provided by GSM includes support for the Radio Link Protocol, which provides reliable transmission on the wireless part of a serial GSM link. RLP reduces actual packet loss to a level comparable to ordinary telephone lines, but RLP may delay packet delivery due to data-link–level retransmission. The TCP layer will also interpret these delays as packet loss and thus will initiate congestion control as just described.

Fast Retransmission

Cáceres and Iftode[1] focused on improving the usability of interactive applications by reducing the data transmission pauses introduced by the hand-offs between wireless transmitters. They did so by forcing the TCP layer to retransmit IP packets as soon as each hand-off completed. Such retransmissions trigger the fast retransmission procedure, which is a part of most contemporary TCP implementations. This approach requires only a few modifications to the network protocol layer, but it does not ameliorate spurious transmission errors or the effect of erroneously instigated congestion control.

Split TCP Connections

To isolate the troublesome wireless link, Bakre and Badrinath[2,3] suggested using a split (also called indirect) connection. Other researchers have also adopted this approach,[4,5] which splits the TCP connection into two: one connection for the wired network, one for the wireless network. The data transfer may thus proceed independently on both. The wired network connection uses the ordinary TCP protocol, allowing communication with stationary hosts using a regular TCP/IP stack. The wireless link connection uses either a modified TCP protocol or a special-purpose protocol optimized for wireless data transfer, which improves performance.

Such improvement comes at the cost of increased latency and buffer requirements. Furthermore, splitting the TCP connection makes it more vulnerable to errors or a crash at the intermediate host.

IP Packet Caching

Another approach can isolate the data transfer on the wired path of a TCP connection from the effects of packet loss on the wireless link. This involves the insertion of an IP layer-caching mechanism between the wired and the wireless network.[6] As most packets are lost on the wireless link, IP packet caching is most useful when the data flow is directed from a stationary to a mobile host. The idea behind IP packet caching at the base station is to perform local retransmissions on the connection's wireless part without affecting the sender. In this way, packet loss on the wireless link—which is not caused by congestion—does not trigger the congestion avoidance-and-control mechanism at the sending host.

To improve transmission performance from a mobile host, the monitor detects packet loss on the wireless link and sends negative acknowledgments to the mobile host. This reduces the time before packets are retransmitted, especially if several packets are lost.

DESIGN CONSIDERATIONS

We based the design of our communication layer on five major considerations, described here.

Communication Model

In our model, a mobile host has three operational modes: connected to the LAN, connected to the mobility support gateway via serial link, and disconnected. In the first mode, we refer to the mobile host as fixed. In the latter two modes, we refer to the mobile host as roaming.

A fixed mobile host functions as a normal stationary host. A roaming mobile host requires special support. The troubles facing a roaming mobile host include wireless links with largely variable quality, which practically guarantees packet loss; periods of disconnection; and changes in point-of-attachment to the network. Our design addresses all these problems.

Disconnected Operation

Our design must also let a mobile host remain disconnected for an extended period of time without jeopardizing its network connections. Note that not all types of applications benefit from keeping the network connection during the period of disconnectivity. If the cost of redoing the work done by the application using the network connection is small, if the number of resources reserved by the application at both ends of the connection is large, and if the application is unlikely to be used while the mobile host is roaming, then it would be a waste of resources to keep the network connection open. We currently do not support this differentiation in the ways applications use network connections, but extending our design to do so is certainly possible.

With TCP, the primary problem during disconnected operation is that the TCP connection may be closed down by a connection time-out. A TCP connection's endpoint may terminate the connection in two cases:

- One or more packets have been retransmitted several times without receiving an acknowledgment.
- Keep-alive probes, which are sent on idle connections at regular (commonly, two-hour) intervals, remain unacknowledged.

Terminating the TCP connection is not always appropriate in a mobile environment, because we want to maintain the TCP connection across interruptions of the underlying data link. We can avoid undesired terminations by means of a split

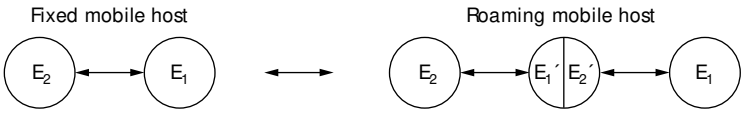

Figure 2. Connection schemes for fixed and roaming hosts.

TCP connection. When the mobile host has no connection to the wired network, we can let the mobility support gateway acknowledge the packets—although they have been delivered to the gateway, not to the mobile host—thus avoiding the TCP time-out.

Several solutions to the split TCP connection exist on the connection handling the wireless link. We can use a special-purpose protocol, or we can modify the TCP layer on the mobility support gateway and the mobile host so that time-outs correctly handle disconnected operation.

Application-level protocols based on the User Datagram Protocol or other connectionless services pose more difficulty because time-outs are not handled at the UDP level. Instead, a UDP application must provide a time-out protocol. Therefore, it is impossible to devise a generally applicable solution to disconnected operation. Instead, special considerations for each application must be built into a design. For example, Zenel and Duchamp propose application proxies at the dividing line between the wired and wireless network.[7]

Triple-Point Connection

We decided on using the split-connection approach to handle many of these problems. For simplicity, and to limit our system to the lower layers of the protocol stack, we don't support specialized handling of application-level protocols. Instead, we simply forward all non-TCP packets at the mobility support gateway as a regular IP router would do.

Our design specifies each TCP connection of the roaming host as two separate TCP connections: One handles the wired network path from the peer host to the mobility support gateway, and the other handles the serial link from the gateway to the mobile host. The gateway acknowledges and passes on packets to and from mobile hosts. From both the mobile and the stationary hosts' viewpoint, a packet has been passed to its respective peer when the gateway has acknowledged the packet.

Unlike the point-to-point connections of regular TCP, this can be viewed as a triple-point connection, in which the mobility support gateway is the center point. This center point consists of two endpoints, which are proxy endpoints for the real endpoints. Figure 2 shows this, in which the endpoints E_i' are proxies for the endpoints E_i. These proxy endpoints act as the real endpoints, and use the same sequence numbers for packet transfer as the real endpoints. This makes it possible to insert and remove the proxy endpoints without changing the state of the real endpoints. In this way, the TCP connection at the mobile host can be switched dynamically between the point-to-point and triple-point connections. This improves performance in the case where the mobile host is in the fixed state, compared with the previously mentioned split-TCP connection approaches that always incur the extra overhead of double connections.

A split-TCP connection approach gives us the option of using a special-purpose protocol for use on the serial link. Measurements have shown that special-purpose protocols only moderately improve performance, compared to regular TCP, when used on wireless-link connections only.[5] As a result, we decided to use TCP/IP with point-to-point protocol (PPP)[8] as the serial-link protocol. This also lets us apply the same TCP, UDP, and IP layers on the mobile host, both when fixed and when roaming.

PPP includes TCP/IP header compression, which reduces the header overhead to an acceptable level on the serial links. To cope with disconnects, our design modifies the TCP layers of the mobile host and the mobility support gateway. This allows the TCP timers of the wireless link connection to be suspended while the mobile host is disconnected, which prevents time-outs.

Our design specifies each TCP connection of the roaming host as two separate TCP connections.

The additional copying and buffering caused by the mobility support gateway introduces extra communication overhead. Because the serial links have substantial round-trip times (hundreds of milliseconds), however, the extra overhead is insignificant.

Figure 3 summarizes the communication model, where TCP′ is our modified TCP layer. This figure shows how we modified the mobile host protocol stack and both sides of the mobility support gateway protocol stack to enable the synchronization described later. Our TrAnsparent COmmunication (TACO) layer handles packet scheduling and serial-link management.

Mobile Hosts on the Move

Supporting mobile hosts in TCP/IP-based networks means facing the problem of supporting host migration between different IP networks. The network-address-based hierarchical routing employed by most IP networks makes it very inconvenient for a mobile host to keep its original IP address. When moving to a new IP network, the mobile host very likely uses a different network address than it had on the previous IP network, thereby making the routing of packets to the mobile host difficult. Conversely, if a mobile host changes its IP address after migration, IP packets from the host's existing network connections will not be routed to the mobile host.

The early 1990s saw several solutions proposed for sup-

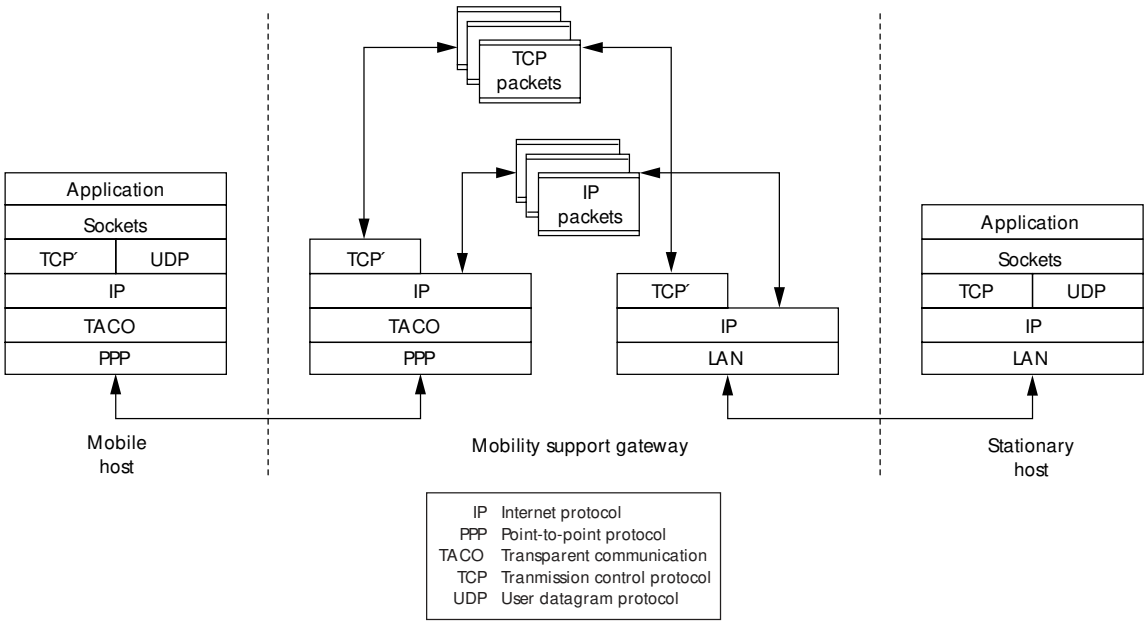

Figure 3. Protocol layers used by roaming host.

porting mobile hosts in IP-based networks. One of the earliest was the Mobile*IP[9,10] (also known as Columbia Mobile IP). The main drawback of their widely used approach was its lack of scalability—it supported only campuswide mobility.

Recently, the IETF completed the Mobile IP scheme[11]—the current standard for supporting mobile hosts in IP networks. This scheme solves the routing problem through a level of indirection. In the home network of the mobile host, a home agent is responsible for forwarding IP packets for the mobile host to the mobile host's current location. When a mobile host connects to a foreign network—that is, to a LAN other than the home LAN—it must establish contact with a foreign agent belonging to that network. This foreign agent informs the home agent of the mobile host's new location. In some cases, the mobile host may function as its own foreign agent. The mobile host is free to send IP packets directly to any host, but packets to the mobile host must travel through the home agent. To eliminate the home agent route, the IETF is extending the Mobile IP protocol to optimize the path taken by IP packets to the mobile host.[12]

In our design, we need to redirect IP packets when a mobile host moves between fixed and roaming states. To retain TCP connections across state transfers requires that the mobile host does not change IP address, as this is part of the unique TCP connection identifier. We have implemented an IP packet redirection mechanism based on the address resolution protocol,[13] but this supports only redirection to other hosts connected to the same LAN. Later on, we may include the Mobile IP protocol.

The mobile host can, after connecting via the GSM network, roam freely in the GSM's area. The mobile host's range, however, may be limited by factors other than the GSM network's coverage. Increased distance from the mobility support gateway may significantly increase the cost of maintaining a GSM connection, especially when crossing the various boundaries of cellular service providers. To support long-range roaming, our design lets the mobile host change its mobility support gateway.

TCP Connection Transfer

When the mobile host goes from fixed to roaming, the following disconnect procedure inserts the proxy endpoints of the center. The mobile host freezes its TCP connections and transfers their states to the mobility support gateway. These states initialize the proxy endpoints at the gateway. The mobile host then redirects the mobile host's IP packets to the gateway, and the gateway restarts the mobile host's TCP connections. The mobility support gateway now assumes responsibility for the mobile host's TCP connections on the wired network. This process of reintroducing a disconnected stateful entity is similar to the process of performing a membership change in group communication systems as described by Birman.[14]

When the mobile host goes from roaming to fixed, it connects directly to the home network, and informs the mobility support gateway of its presence. The mobile host and the gateway then must synchronize the states of the TCP connections. If the gateway has buffered packets, these must be delivered and acknowledged by their destinations before the connections' center point can be removed. The gateway

enters a synchronization mode in which the proxy endpoints send packets but accept only acknowledgments. The gateway discards all data arriving in new packets. When the buffers at the gateway are empty, the real endpoints of the TCP connections are in a consistent state. This is because all packets that one endpoint believes to be acknowledged actually have been acknowledged by the peer. Finally, the mobile host redirects the IP packets to itself, and the mobile host again functions as a fully connected workstation.

Our design uses a simple tunneling mechanism during the connect and disconnect phases. This lets the mobile host use TCP/IP protocols for the communication described earlier without interference from IP packet redirection and TCP connection freezing.

PACKET SCHEDULING

Bandwidth is scarce when a user dials up a serial link for data transmission. For one thing, bandwidth provided by serial interfaces is an order of magnitude less than that provided by LAN interfaces. Moreover, the cost of communication and the availability of physical connections limit the time during which a serial interface may be kept connected.

Our TACO layer sends packets between the mobile host and the mobility support gateway, based on information about priority and latency requirements.

The objective of packet scheduling is partly to ensure that the most important data are sent first, partly to ensure that transmissions are cost-effective. The idea is to send the highest priority packets first, then to delay the transmission of lower priority packets. The delay continues until the packets cannot wait any longer or until the underlying connection medium is switched to a faster, cheaper medium. The approach we discuss now relies on the packet scheduling design in our TACO layer. The packets are sent between the mobile host and the mobility support gateway, based on information about the packets' relative priority and latency requirements.

Interface Profiles

TACO needs a set of transmission requirements for each application. Typically, users (or the person installing the application on the laptop) provide these.

TACO recognizes an application by its transmission endpoint (TEP), which is merely the tuple (IP address, protocol type, port number). We require the following information for each TEP:

- Minimum required bandwidth (MRB).
- Maximum latency (ML).
- Maximum cost per minute (MCM).
- Minimum idle time before disconnect (MIT).
- Preferable maximum time before connect (PTC).

The first three items are an *interface profile*. Each communication interface known to the system, for example, GSM modem, ordinary modem, and Ethernet, has such a profile, based on the respective interface's technical specifications. (Future interfaces may have different profiles.) The last two items provide additional information about the TEP and determine, for example, when to establish and shut down a physical link.

Two-Level Scheduling

The applications running on the mobile host may include a mix of applications designed for mobile computing and applications totally unaware of changes in the underlying media. As a result, we propose using packet scheduling both at the system level and at the application level. System-level scheduling arbitrates the (possibly scarce) transmission resources among competing applications, whereas application-level scheduling lets the application adapt to changes in resources.

Essentially, interface profiles avoid choking low-bandwidth connections with data that could just as soon wait until later. TACO may even shut down expensive cellular connections despite the presence of unsent packets, if those packets are of lower priority. For example, these would include those packets to be transmitted only at a low cost per minute. Lower priority applications may easily experience longer periods of disconnection than higher priority applications.

Note that we do not modify any application code; we merely need to set up appropriate interface profiles for each application.

Scheduling Implementation

System-level scheduling is based on three data structures: a *wait queue*, an *active set*, and a *pending set*. A queue in one of these structures represents each active TEP. TACO appends any packets sent to the corresponding TEP queue. If no queue is associated with the TEP in any of the data structures, TACO will create a new queue and place it in a structure, depending on the TEP's requirements.

Wait Queue. TACO places any new TEP queue here, whenever PTC is greater than zero. It remains here until the timer value reaches zero. When a time-out occurs, TACO moves the corresponding TEP queue to either the active set or the pending set. The location depends on whether or not items 1 through 3 of the requirements match the current interface profile. If the requirements match, TACO moves the queue to the active set structure; otherwise, the queue is moved to the pending set.

Active Set. Any TEP queue, where PTC equals zero, will stay in the active set as long as the queue is nonempty and MIT is greater than zero. When an MIT time-out occurs, TACO deletes the corresponding empty TEP queue from the active set. If an interface profile changes, TACO moves any queues, where items 1 through 3 of the related requirements do not match the new profile, to the pending set . TACO sets a physical link's idle timer to the maximum MIT among TEPs in the active set, which is adjusted as the active set changes.

Pending Set. Any TEP queue, where PTC equals zero, and the items 1 through 3 of the related requirements set do not match the interface profile, will appear in the pending set. In case a change in interface profile occurs, TACO moves any queues, where items 1 through 3 of the related requirements do match the new interface profile, to the active set.

Because link requirements are associated with TEPs rather than packets, all packets sent on a given TEP will have identical link requirements. This means that within TEPs, packets can be scheduled on a simple FIFO basis. Round-robin TEP scheduling in the active set allows a TEP, which is granted access to the communication interface, to send one packet only from its associated data queue. This ensures that no TEP experiences starvation, but bandwidth is not necessarily divided equally among TEPs, as packets from various TEP queues may differ in size.

When an active set is no longer empty, TACO establishes a physical network link. The link is broken when the physical link's idle timer generates a time-out. Involuntary disconnects cause attempts to reconnect whenever the interface profile matches the requirements of at least one of the TEPs in either the active or the pending set. The reason for searching the pending set is that disconnects might occur due to a change of interface.

The system will use spare bandwidth by transmitting packets from any TEP, from either the wait queue or the pending set. Because all TEPs in the pending set have link requirements not matched by the interface profile, only packets associated with these connections are sent. This assumes, however, that no TEPs from the wait queue match the interface profile. Any transmission initiated to utilize spare bandwidth leaves the idle timer, associated with the physical link, unaffected. During periods of disconnection, application-level protocols may retransmit packets, causing many duplicate packets to be buffered in the TEP queues. To remedy this, we let the user specify that IP packets for certain TEPs be discarded during disconnected operation.

The system will use spare bandwidth by transmitting packets from any transmission endpoint, from either the wait queue or the pending set.

Application-Level Scheduling

Dynamic application-level adaptation lets individual applications adapt to their environment, which is beneficial in several ways. Dynamic adaptation can reduce the data transfer rates of individual applications to match the available link. This lessens the burden on the system-level packet scheduler, if not making it altogether superfluous. Such adaptation can minimize the data actually transmitted, because applications may prevent stale data from being transferred to the communication layer. If, for example, no communication interface is present and an application wants to send updates to a server, later updates may overwrite earlier ones and thus waste bandwidth. If, instead, the application waits until the updates actually can be delivered, then only the latest, valid ones must be transmitted.

Our design's approach to application-level adaptation is that applications may request notification when a certain link quality is available. The application specifies, through a system call, the link quality threshold to generate a notification. When the quality deviates from the threshold, TACO notifies the application. This lets applications adapt to changes without having each application implementing a link quality monitoring system.

Note that our design does not guarantee that an application may use the available resource exclusively. Numerous applications may be notified that a resource's availability has crossed a certain threshold. If all applications try to acquire the resource, the resource's availability as seen by the individual application possibly no longer exceeds the specified threshold. Applications requesting the same type of interface may thus still compete for the available resources.

Our approach resembles the one taken by Brian Noble and colleagues in their design of Odyssey,[15,16] but where ours focuses on network connections, theirs is more general. Odyssey can work with different types of system resources, with different characteristics and different operations performed on them. Odyssey consists of two components: wardens handle the operations specific for a single type, and the viceroy takes care of the centralized resource management. Applications that want to use these facilities do it through a special system interface. They do not thereby have system-level scheduling, which prevents nonadapting applications from using system resources.

Packet-Scheduling Summary

Our design has a simple scheduling mechanism that provides both system-level scheduling and application-level scheduling. The specification of link requirements is a natural way to state

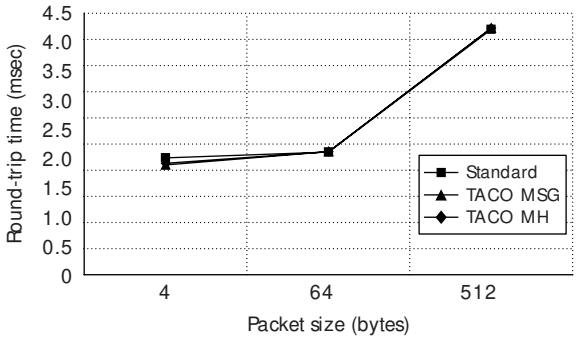

Figure 4. TCP round-trip time on 10-Mbps Ethernet.

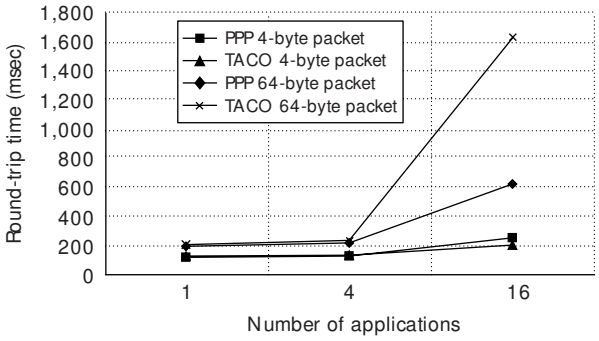

Figure 5. Concurrent TCP round-trip time on a 14-Kbps modem.

an application's communications needs, perhaps with the exception of the PTC. The main disadvantage of our scheduling approach is that it is packet based, which means that bandwidth may not be divided equally among the active queues.

TACO IMPLEMENTATION

We built a prototype of our communication layer design around the Linux 1.1.12 operating system. The prototype shows our technical design's viability and gave us experience with packet scheduling complexities. In this section we discuss several experiments we performed. We performed them on a stationary 60-MHz Pentium PC as the mobility support gateway and a 75-MHz Pentium notebook PC as a mobile host. The mobile host could connect by 10-Mbps Ethernet, by 14,400-bps ordinary telephone connection, or by wireless 9,600-bps GSM data connection. We measured the effects of both the TACO layer and of packet scheduling on network performance.

To evaluate the TACO layer, when the mobile host connects directly to the fixed network, we measured the round-trip time of TCP packets between the mobile host and the mobility support gateway. We did this with three different configurations. One had the TACO layer installed in neither host; one had it installed in only the mobile host; and one had it installed only in the mobility support gateway. As Figure 4 shows, the overhead contribution from the TACO layer is negligible when directly connected to a fixed network.

To evaluate packet scheduling, we measured the round-trip time of TCP packets between the mobile host and the mobility support gateway when connected by a 14,400-bps modem. To analyze how the scheduling dealt with multiple, concurrent connections, we used one, four, or sixteen concurrent applications. As Figure 5 shows, the overhead increases substantially with an increase in concurrent applications. This is particularly true for the larger, 64-byte packets, which cause a substantially longer round-trip time when using sixteen concurrent applications.

A key reason for using system-level scheduling is to maximize the use of the currently available connection. We thus conducted an experiment where a high-priority interactive UDP application competed with a low-priority TCP bulk transfer application. We conducted the tests between the mobile host and the mobility support gateway, which were connected by a 14,400-bps modem.

We expected the experiment to show minimal effects of the bulk transfer. As Figure 6 shows, this was not the case. With a low-priority bulk transfer running together with our round-trip time measurements, the round-trip time increased sharply. We tracked this down to a problem concerning buffering in the serial link driver. The TACO layer cannot distinguish between a packet transmitted on the serial link and a packet buffered in the serial link driver. Consequently, the buffer in the serial link driver filled with packets from the bulk transfer application as soon as there were no queued packets from the interactive session.

The primary use of the TCP timer-disabling mechanism is to eliminate TCP connection time-outs during periods of disconnected operation. In our experiments, a side effect of the retransmit timer's disabling was that we avoided the retransmission time-outs and the exponential back-off of the retransmission timer. This made the data transfer proceed faster when a connection was reestablished.

Disconnects rarely happen on modem links but regularly occur on cellular GSM data links. We thus tested the effects of disconnects on bulk transfer connections on a GSM data link. The GSM link was forced to disconnect twice (after one minute, then after three) during a TCP bulk data transfer that would take approximately three minutes to complete without disconnects. It took between 60 and 90 seconds to reconnect after each of the forced disconnects. This was primarily due to the long connect times of a GSM link (typically on the order of one-half to one minute). As Figure 7 shows, the timer disabling generally provides a slight improvement in throughput compared to regular TCP, but we also see that the worst case of the regular TCP connection is considerably worse.

As our experiments indicated, we achieved only some of our goals with this prototype. Clearly, the implementation of pack-

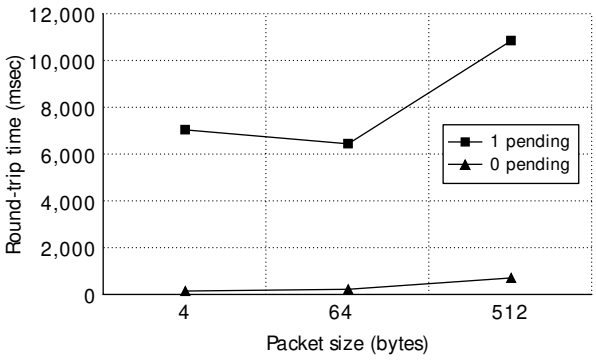

Figure 6. Interactive UDP connection and pending bulk transfer TCP connections concurrently on 14,400-bps modem.

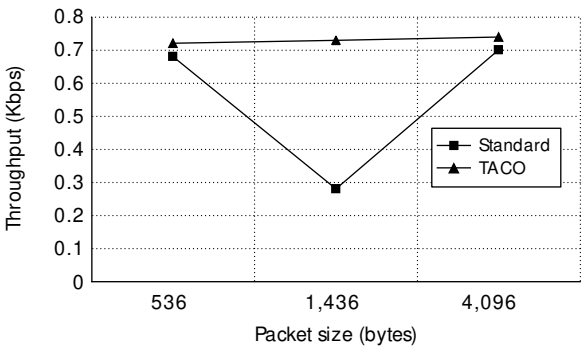

Figure 7. Effects of disconnects on throughput on TCP connections using 9,600-bps GSM.

et scheduling needs to be improved, but at the same time the prototype shows that it is possible to provide dynamic adaptation when switching between being fully connected through a fixed network and an intermittent, wireless network.

CONCLUSION

The major advantage of our design is that many existing applications (for example, those using TCP/IP) can be used unchanged in weakly connected mobile environments. This functionality comes at a reasonable price—communication is slightly slower; however, in most cases this is not a problem because most mobile hosts use low-bandwidth media.

It might seem a bit late to focus on very low-bandwidth media. However, low-bandwidth media will also exist in the future, even though what constitutes low bandwidth will change. Today 10 Kbps is very low bandwidth; in a couple of years, 100 Kbps will be. As applications evolve to require more bandwidth for multimedia transmissions and so on, better Quality of Service will always be more expensive and sometimes so expensive that it becomes acceptable to stay with cheaper media. Hence, the work presented in this article will remain viable as long as different ranges of bandwidth exist.

We expect to extend TACO with support for several new network/link types, such as wireless LAN. However, these extensions will not alter our design, which is prepared for new link types. Integration with Mobile IP could solve the problems of connecting to a foreign LAN in the case when the foreign LAN is connected to the home LAN.

Other major future work includes support for protocols other than TCP/IP with focus on real-time and resource reservation techniques needed for multimedia streams. ■

REFERENCES

1. R. Cáceres and L. Iftode, "Improving the Performance of Reliable Transport Protocols in Mobile Computing Environments," *IEEE JSAC*, special issue on Mobile Computing Networks, Vol. 13, No. 5, June 1995.
2. A. Bakre and B.R. Badrinath, "I-TCP: Indirect TCP for Mobile Hosts," Tech. Report DCS-TR-314, Dept. of Computer Science, Rutgers Univ., Oct. 1994.
3. A.V. Bakre and B.R. Badrinath, "Implementation and Performance Evaluation of Indirect TCP," *IEEE Trans. Computers*, Mar. 1997, pp. 260-278.
4. M. Kojo, K. Raatikainen, and T. Alanko, "Connecting Mobile Workstations to the Internet over a Digital Cellular Telephone Network," Tech. Report C-1994-39, Dept. of Computer Sci., Univ. of Helsinki, Sept. 1994.
5. R. Yavatkar and N. Bhagawat, "Improving End-to-End Performance of TCP over Mobile Internetworks," *Proc. Workshop on Mobile Computing Systems and Applications*, IEEE Computer Society Press, Los Alamitos, Calif., 1994.
6. H. Balakrishnan et al., "Improving TCP/IP Performance over Wireless Networks," *Proc. 1st ACM Int'l Conf. Mobile Computing and Networking*, ACM Press, New York, Nov. 1995.
7. B. Zenel and D. Duchamp, "General Purpose Proxies: Solved and Unsolved Problems," *Proc. 6th Workshop on Hot Topics in Operating Systems*, IEEE CS Press, Los Alamitos, Calif., 1997.
8. W. Simpson, "The Point-to-Point Protocol (PPP)," RFC 1661, July 1994, ftp://ds.internic.net/rfc/rfc1661.txt.
9. J. Ioannidis, D. Duchamp, and G.Q. Maguire Jr., "IP-based Protocols for Mobile Internetworking," *Proc. SIGCOMM 91* ACM Press, New York, Sept. 1991, pp. 235-245.
10. J. Ioannidis, D. Duchamp, and G.Q. Maguire Jr., "The Design and Implementation of a Mobile Internetworking Architecture," *Proc. 1993 Winter Usenix Tech. Conf.*, Usenix, Berkeley, Calif., Jan. 1993, pp. 491-502.
11. C. Perkins, "IP Mobility Support," RFC 2002, Oct. 1996, ftp://ds.internic.net/rfc/rfc2002.txt.
12. C. Perkins and D.B. Johnson, "Route Optimization in Mobile IP," IETF draft, July 1997, ftp://ftp.ietf.org/internet-drafts/draft-ietf-mobileip-optim-06.txt (working draft).
13. D.C. Plummer, "An Ethernet Address Resolution Protocol," RFC 826, Nov. 1982, ftp://ds.internic.net/rfc/rfc826.txt.
14. K.P. Birman, "The Process Group Approach to Reliable Distributed Computing, *Comm. ACM*, Vol. 36, No. 12, Dec. 1993, pp. 37-53.

15. B.D. Noble, M. Price, and M. Satyanarayanan, "A Programming Interface for Application-Aware Adaption in Mobile Computing," *Computing Systems*, Vol. 8, No. 4, Fall 1995, pp. 345-363.
16. B.D. Noble et al., "Agile Application-Aware Adaption for Mobility," *Proc. 16th ACM Symp. Operating System Principles*, ACM Press, New York, Oct. 1997.

Jørgen S. Hansen is a PhD student at the Department of Computer Science, University of Copenhagen. His research interests include operating system support for high-speed networking, distributed systems, and mobile computing. He received an MS in computer science from the University of Copenhagen in 1996.

Torben Reich is a software engineer at DSC Communications A/S, where he is currently working with translation between proprietary and standardized Telecommunications Management Network (TMN) information models for modeling Synchronous Digital Hierarchy (SDH) equipment. Reich received an MS in computer science in 1996 at the University of Copenhagen.

Birger Andersen is associate professor at Copenhagen Business School and member of the Distributed Systems Group. His work focuses on distributed systems, including mobility and consistency issues in mobile computing. Andersen received an MS in 1988 and a PhD in 1992 at the University of Copenhagen. He is a member of the ACM and the IEEE Computer Society.

Eric Jul is associate professor at the Department of Computer Science, University of Copenhagen. His research interests include distributed operating systems, object-oriented languages and systems, and object-oriented design. He received an MS in computer science from the University of Copenhagen, and a PhD in computer science from the University of Washington.

Readers may contact Hansen and Jul at Dept. of Computer Science, University of Copenhagen, Universitetsparken 1-3, DK-2100 Copenhagen, Denmark, e-mail {cyller, eric}@diku.dk; Reich at DSC Communications A/S, R&D, Dept. of Embedded Software, Lautrupbjerg 7-11, DK-2750 Ballerup, Denmark, e-mail treich@dscc.dk; and Andersen at Copenhagen Business School, Dept. of Informatics and Management Accounting, Distributed Systems Group, Howitzvej 60, DK-2000 Copenhagen, Denmark, e-mail birger.andersen@cbs.dk.