A Network Architecture for Heterogeneous Mobile Computing

Eric A. Brewer, Randy H. Katz
Elan Amir[1], Hari Balakrishnan[2], Yatin Chawathe, Armando Fox[3], Steven D. Gribble, Todd Hodes,
Giao Nguyen, Venkata N. Padmanabhan[4], Mark Stemm, Srinivasan Seshan[5], and Tom Henderson
University of California at Berkeley

This paper summarizes the results of the BARWAN project, which focused on enabling truly useful mobile networking across an extremely wide variety of real-world networks and mobile devices. We present the overall architecture, summarize key results, and discuss four broad lessons learned along the way. The architecture enables seamless roaming in a single logical overlay network composed of many heterogeneous (mostly wireless) physical networks, and provides significantly better TCP performance for these networks. It also provides complex scalable and highly available services to enable powerful capabilities across a very wide range of mobile devices, and mechanisms for automated discovery and configuration of localized services. Four broad themes arose from the project: 1) the power of dynamic adaptation as a generic solution to heterogeneity, 2) the importance of cross-layer information, such as the exploitation of TCP semantics in the link layer, 3) the use of agents in the infrastructure to enable new abilities and to hide new problems from legacy servers and protocol stacks, and 4) the importance of soft state for such agents for simplicity, ease of fault recovery, and scalability.

1 Introduction

The BARWAN project was a three-year DARPA-funded project at the University of California at Berkeley that focused on enabling truly useful mobile networking across an extremely wide variety of real-world networks and mobile devices. This paper presents the overall architecture, summarizes key results, and discusses broad lessons learned along the way. We attacked nearly all aspects of the problem, including link-layer and transport protocols, mobile routing, power management, application support, multimedia streaming, and local and global network services.

For quite some time, our group has been driven by a simple but compelling mantra:

Access is the Killer App

Originally, in 1994, it was an argument against the prevailing trend towards mobile devices that were essentially islands of computation—mobile computers looking for a "killer app" similar to a personal organizer. But the phrase is really an argument that what matters is not so much what the device can do, but rather the data and computing power to which it has access. It is possible to harness terabytes of data and the power of supercomputers even while mobile—as long as you access them over a ubiquitous network.

The resulting architecture reveals how to build this network out of an overlaid collection of networks, how to seamlessly roam around it, how to make it perform well, and how to leverage computing in the infrastructure to enable new abilities and services for even the simplest mobile devices.

We believe in *ubiquitous localized networking*: the network as a whole has broad coverage, but connection quality and services vary greatly by location. The connection may vary from wired or infrared in-room networks with great performance, to metropolitan cellular networks, to satellite networks with high latencies but tremendous coverage. Services vary greatly as well: from home/office printer access, to local driving directions, to global services such as search engines and web access. This is an alternative to the idea of ubiquitous computing [Wei93], in which we care far more about nearby connectivity than nearby computing.

We also believe in a powerful infrastructure that is highly available, cost effective, and sufficiently scalable to support millions of users. In general, computation, storage and complexity should be moved from the mobile devices into the infrastructure, thus enabling powerful new services, better overall cost performance, and small, light-weight, low-power, inexpensive mobile devices with tremendous functionality. We have built such an infrastructure and will show how it provides new abilities for a very simple mobile client, the PalmPilot PDA.

We have implemented and evaluated the architecture over a three-year period; many of the specific results have already been evaluated in the literature. We will often refer to such papers for more detail and for complete coverage of the related work. This paper evaluates the architecture as a whole and summarizes the lessons we have learned.

The architecture is probably best introduced in terms of some of the challenging problems for which it offers solutions:

- We provide seamless mobility both within a network and across heterogeneous networks. The detection and setup of network connections is transparent and automatic, as is the selection of the best network in range. For several networks, we also provide very low-latency handoffs (below 200ms), thus enabling smooth audio and video even while roaming.
- We provide a reliable transport layer, based on TCP, that works with legacy servers, while hiding the effects of wireless losses and asymmetry that typically ruin TCP performance.

1: Now at Fast Forward Networks
2: Now at MIT
3: Now at Stanford
4: Now at Microsoft Research
5: Now at IBM T. J. Watson Research Center

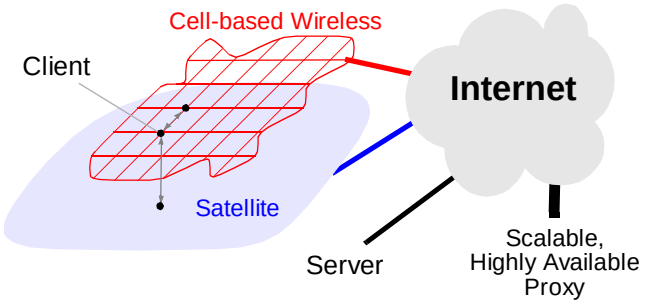

Figure 1: Elements of the BARWAN Architecture. In addition to heterogeneous clients, networks and servers, we provide a *proxy* to assist with mobility and dynamic adaptation. The three black circles indicate points visible from the client; *horizontal* handoffs occur between two cells of the same network, while *vertical* handoffs occur between different networks, e.g., radio-frequency wireless to satellite. The architecture also enables high-performance reliable data transport for these networks.

- We provide automatic discovery and configuration of local network services, such as local printers and DNS servers. We also provide remote control of local environments even if they are new to the user, such as the A/V equipment another's lab.
- We support thin clients through dynamic adaptation of content, tuning the content specifically for the device, the user, and the current quality of the network connection, often reducing latency by more than a factor of five. We also enable new abilities for such devices by leveraging computation in the infrastructure.
- We provide cluster-based support for scalable, manageable and highly available infrastructure services. We also provide a layered programming model for implementing services that allows authors to focus on service *content* rather than the challenges of availability and scalability.

1.1 Basic Approach

The there two basic views of the architecture: the physical view of a client roaming through an overlaid collection of networks, shown in Figure 1, and the network-layer view, shown in Table 1.

A key concept is *overlay networking*, which is the unification of several heterogeneous networks, of varying coverage and performance, into a single logical network that provides coverage that is the union of the networks' coverage with performance corresponding to the best network in range. In addition to traditional cell-based roaming (*horizontal* roaming), we provide transparent roaming *across* the constituent networks (*vertical* roaming), even when this requires changing the IP address or network interface.

Given an overlay network, the next immediate challenge is the poor performance delivered by standard TCP over the wireless and asymmetric networks that enable mobility. TCP assumes that all losses are due to congestion, and that the connection is symmetrical, both of which may be false. We surmount these issues through a combination of TCP enhancements and corrective agents in the infrastructure.

Network Services Section 5	Automatic discovery and configuration of localized services
Presentation Section 4	Dynamic Adaptation of content for heterogeneous networks and devices (the *proxy architecture*)
Transport Section 3	Reliable data transport for wireless and asymmetric networks
Routing Section 2	Roaming in heterogeneous networks (*Overlay IP*)
Link Layer Section 3.1	Transport-aware link-level retransmission (*Snoop* protocol); management of shared wireless links

Table 1: Layered view of the Architecture

Another key concept and primary contribution is the *proxy architecture*, which uses a powerful agent in the infrastructure to adapt content dynamically to support heterogeneous networks and mobile devices that (nearly all) servers handle poorly. The proxy enables useful access even from very simple devices.

Finally, we define an architecture for *adaptive network services*. IP connectivity is not sufficient: mobile users need to discover, configure and use local services, such as DNS servers, local printers, and in-room control of lights and A/V equipment. This enables a far richer environment for mobile users, and when combined with the proxy greatly simplifies the creation and use of new services.

After looking at broad themes and our network testbed, we examine each of the four major areas in Sections 2 through 5. We close by revisiting our themes and summarizing our conclusions in Sections 6 and 7.

1.2 Four Key Themes

There are some ideas and techniques that are used repeatedly throughout the architecture. In particular, there are four general themes that merit higher-level discussion, and that we revisit throughout the discussion.

Dynamic Adaptation is the general way we deal with varying network conditions and with client heterogeneity. This includes new adaptive mechanisms for TCP, on-demand format conversion, real-time video transcoding, dynamic quality/performance tradeoffs, and even dynamic generation of customized user interfaces for small devices.

Cross-layer Optimization: There are several places where we explicitly break the OSI networking model to enable smarter adaptation or better performance. These include using physical layer information to trigger handoffs, and exploiting transport-layer information to guide link-level retransmission and application-level dynamic adaptation.

Agent-Based Capabilities: In order to maximize our ability to work with legacy clients, servers and TCP stacks, we often use agents within the network infrastructure to add

new capabilities or to improve performance. Fundamentally, the role of these agents is to hide to vagaries of mobility, wireless networking and extreme heterogeneity from all of the legacy clients and infrastructure; the agents make everything look like it is "supposed to". The alternative is take an end-to-end approach in which you are allowed to modify all components, including clients and servers. This has the advantage of a potentially better overall solution, but it is nearly impossible to deploy, since it requires upgrading most clients and servers. We will show that the agent-based solutions not only have a better deployment path, but in many cases perform better and in some cases are the only option, because they leverage information outside of the particular end-to-end connection. We use agents to handle mobility and vertical handoff, to enhance TCP performance for wireless or asymmetric networks, and as the basis of proxy architecture.

Exploit Soft State: *Soft state* is state that aids in performance or adds capabilities, but that is not required for correctness. Soft state need not be recovered after failures, can be thrown out at any time, and can typically be stale or inconsistent without violating correctness. In general, most of our agents use soft state to make them simpler and trivially fault tolerant. We exploit soft state in all aspects of the design.

1.3 BARWAN Testbed

We concentrate on four networks in the overlay testbed:

IBM Infrared Wireless LAN: This infrared network forms our basic in-room network [IBM95]. Infrared provides trivial spectrum reuse, and thus enables very high bandwidth per cubic meter in buildings. It is also the lowest cost network.

Lucent WaveLAN: We have used both the 900-MHz and 2.4-GHz versions [ATT94] as our building-size network. We have coverage of four floors of Soda Hall, with one or two basestations per floor.

Metricom Ricochet: This is a packet-based radio network for metropolitan areas [Met96], including the Bay Area. We also deployed a private network in Soda Hall for experimentation.

Hughes Direct-Broadcast Satellite (DBS): We have one dish on top of Soda Hall and have also used a portable dish at remote sites [Dir98].

Figure 2 shows the typical bandwidths, latencies, cell sizes, and registration times for these networks. The Effective Bandwidth is the throughput measured by the user, excluding overhead and retransmissions. Latency measures the round-trip time for a simple packet. Cell Size is the approximate diameter of the coverage area of one transmitter (basestation). Registration Time is the time to set up the initial connection. The registration times were measured by sending a stream of UDP packets to a mobile host, turning on the network interface, and marking the time between when the network interface was turned on and when the first data packet was received by the mobile. The key conclusions are that there is wide variation in all dimensions, and that there is no single "best" network.

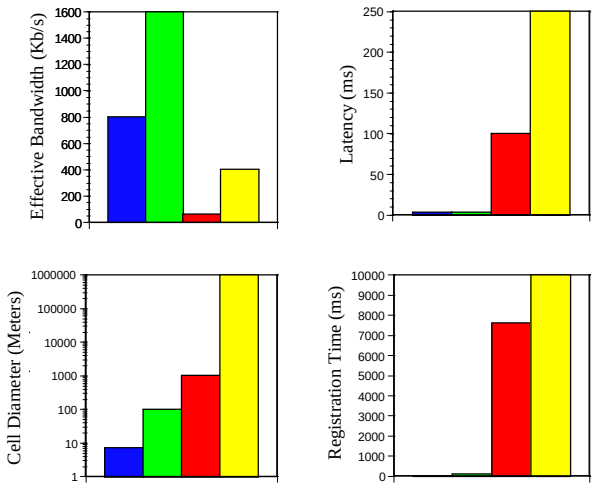

Figure 2: Testbed Networks
Latency is the round-trip time; *Registration Time* is the time to set up the initial connection.

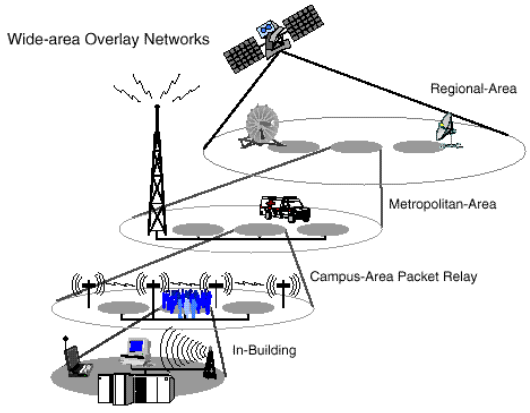

Figure 3: Wireless Overlay Network Structure

2 Overlay Networking

Figure 3 shows an example of a wireless overlay network. Lower levels are comprised of high-bandwidth wireless cells that cover a relatively small area. Higher levels in the hierarchy provide a lower bandwidth per unit area connection over a larger geographic area. In our system, we have three overlay levels. The lowest level is comprised of a collection of disjoint room-size high-bandwidth networks, which provide the highest bandwidth per-unit-area: one Mb/s or more per room. The second level consists of building-size high-bandwidth networks that provide approximately the same bandwidth as the room-size networks, but cover a larger area, for example, a single floor of a building. The final level is a wide-area data network, which provides a much lower bandwidth connection, typically tens of kilobits, over a much wider geographic area. We can also support "mobile" wired networks, such as dynamically connecting to a local ethernet subnet via DHCP.

We must depend on existing networking technologies and wireless data providers to have a full range of wireless networks at our disposal. Although we would ideally modify some components of these systems (e.g. basestation software), this is not always possible. For example, while we can modify and experiment with the basestations for the room-size and building-size overlays, the wide-area data overlay is owned and administered by a third party. As a result, we cannot directly control the overlay's infrastructure. This is an important consideration because it limits the modifications we can make (in these cases) to support vertical handoff.

2.1 Vertical Handoffs

The primary objective of a vertical handoff system is to minimize the handoff latency for an individual user while keeping bandwidth and power overheads as low as possible.

There are some important differences between the horizontal and vertical handoffs. First, there is a distinction between *up* and *down* in vertical handoffs: an *upward vertical handoff* is a handoff to an overlay with a larger cell size and lower bandwidth/area, and a *downward vertical handoff* is a handoff to an overlay with a smaller cell size and higher bandwidth/area. Downward vertical handoffs are less time-critical, because a mobile can always stay connected to the upper overlay during handoff. Second, in a heterogeneous network, the choice of the "best" network can be challenging; for example, an in-building RF network with a weak signal may yield better performance than a wide-area data network with a strong signal. Finally, there may also be financial differences that do not arise in a single network; some networks charge per minute or byte.

The primary technical objectives in the design of a seamless vertical handoff system are the balance among handoff latency, power consumption, and wasted bandwidth.

Low-Latency Handoff: make the switch between networks as seamless as possible with minimal interruption or data loss. As a user roams among areas of good and poor connectivity, the only user-visible change should be due to the limitations of the specific networks. For example, lower levels may support full-motion video and high-quality audio, while higher overlays may support only audio.

Power Savings: minimize the power drain due to multiple simultaneously active network interfaces. The simplest approach to managing multiple network interfaces (NIs) is to keep them on all the time. However, measurements of the network interfaces [SGHK96] show that the IBM Infrared and WaveLAN interfaces together consume approximately 1.5 watts *even when not sending or receiving packets*.

Bandwidth Overhead: minimize the amount of additional network traffic used to implement handoffs. Implementing vertical handoffs consumes bandwidth in the form of beacon packets and handoff messages.

There are many inherent tradeoffs in meeting these objectives. For example, reducing power consumption by keeping network interfaces off when not in use increases handoff latency. Similarly, zero-latency handoff could be achieved by simply sending and receiving data across all network interfaces simultaneously at all times, but this results in an inordinate waste of bandwidth and power. Our goal is to balance low-latency handoffs with the issues of power and bandwidth efficiency.

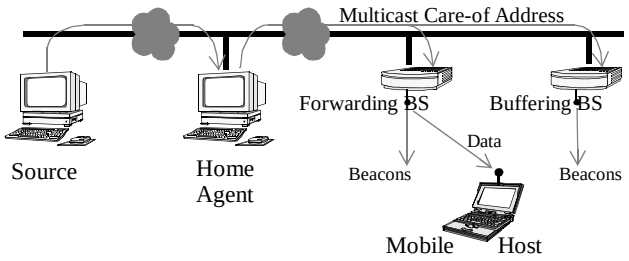

Figure 4: Elements of the Handoff System

2.2 The Handoff Mechanism

Handoffs are built on top of the mobile routing capabilities of Mobile IP [Per96], shown in Figure 4. *Mobile Hosts* (MHs) connect to a wired infrastructure via *Basestations* (BSs) which act as *Foreign Agents*. A *Home Agen*t (HA) encapsulates packets from the source and forwards them to the basestations.

One important difference is that the *care-of address* (for basestations we control) is a multicast rather than unicast address. The mobile selects a small group of BSs to listen to this multicast address for packets forwarded by the HA. One of the BSs is selected by the MH to be a *forwarding* BS; it decapsulates the packets sent by the HA and forwards them to the MH. The others are *buffering* BSs; they hold a small number of packets from the HA in a circular buffer. These buffers are soft state: they improve performance and can be lost at any time without affecting correctness.

Basestations send out periodic beacons similar to Mobile IP foreign-agent advertisements. The MH listens to these packets and determines which BS should forward packets, and which should buffer packets in anticipation of a handoff; the rest simply don't listen to the multicast group at all. When the mobile initiates a handoff, as shown in Figure 5, it instructs the old BS to move from forwarding to buffering mode,[1] and the new BS to move from buffering to forwarding mode. The new BS forwards the buffered packets that the mobile has not yet received.

An upward vertical handoff is initiated when several beacons from the current overlay network are not received. The MH decides that the current network is unreachable and hands over to the next higher network. The arrival of beacons on a lower overlay initiates a downward vertical handoff. The MH determines that the mobile is now within range and switches to the lower overlay. The handoff starts when the lower overlay becomes reachable or unreachable, and ends when the first packet forwarded from the new overlay network arrives at the MH. We depend only on the presence or absence of packets to make *vertical* handoff decisions.

For horizontal handoffs, the MH uses a channel-specific metric to compare different BSs and connects to the best one according to that metric. This allows the horizontal handoff

1: The arrows represent the logical endpoints of a message, not the path that the message takes from source to destination. Messages go through the new basestation if the old one is unreachable.

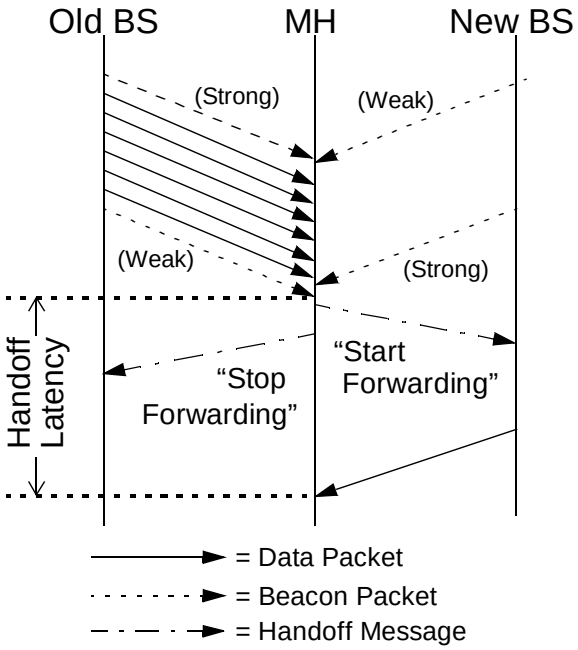

Figure 5: Breakdown of a Handoff

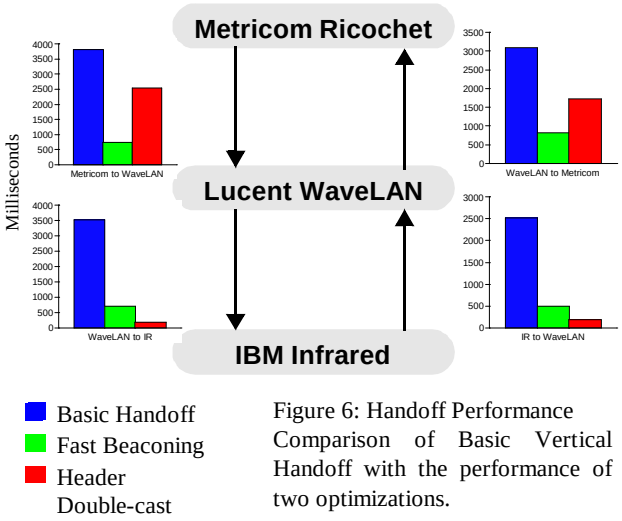

Figure 6: Handoff Performance Comparison of Basic Vertical Handoff with the performance of two optimizations.

system to operate seamlessly underneath the vertical handoff system. For an overlay network that handles mobility directly, such as Metricom, our system does nothing and lets the network make its own horizontal handoff decisions.

2.3 Performance

The primary metric we use to evaluate handoff performance is the *handoff latency*, as pictured in Figure 5. More detailed descriptions of this work [SK98] evaluate not only more optimizations, but also power management and bandwidth overhead. They also provide breakdowns of the performance numbers and analytical models for each method.

Here we examine the basic mechanism and the two most useful optimizations:

Fast Beaconing (FB): The MH can selectively instruct a subset of the BSs that are listening to its multicast group to transmit beacon packets at a higher frequency, thus reducing the time to discover connectivity status. However, this reduces the effective bandwidth on the wireless interface and may cause media access problems on the wireless subnet.

Header Double-casting (HD): Two BSs are placed in forwarding mode simultaneously; the current BS and a BS of the next higher overlay. The secondary BS sends only the headers of the packets (since the data is received on the primary interface); this provides up-to-date status for the secondary network. When more than *n* consecutive packets are received on a new interface with none received on the old interface, the MH decides that the old overlay is unreachable and initiates a handoff to the new interface.

Figure 6 compares these three approaches for four vertical handoff scenarios. The beacon period was one second for the Basic and HD versions, and 200ms for the FB version. For HD, we switched networks when we saw 10 consecutive packets on the new interface with none on the old. Key results include:

- Fast beaconing can result in subsecond handoff latency proportional to the bandwidth overhead. This approach consumes bandwidth whether or not data is being sent to the mobile device.
- Header double-casting results in very low latency (170ms) with a modest bandwidth overhead (about 15% for WaveLAN).
- The impact of HD on Metricom handoffs was limited because the high base latency negates the "up to date" status providing by receiving packets on the secondary interface.
- These low handoff latencies enable very smooth audio and video across the handoff. Handoff is detectable but not disruptive during an audio stream, and nearly undetectable during a video stream.

2.4 Conclusions

We designed and implemented an overlay networking system that enables largely transparent handoffs from in-room through metropolitan-area networks. In addition to the basic handoff mechanism, we developed two enhancements that significantly reduce handoff latency, thus enabling smooth handoffs even for intolerant applications such as audio playback.

Not all transitions between levels in the overlay network hierarchy can be treated identically. In our system, the best choice was specific to the pair of networks that were chosen. For our testbed, header double-casting performs the best for transitions between in-room and in-building networks, and fast beaconing works best for transitions between in-building and wide-area networks.

Depending on the presence or absence of data packets rather than channel measurements allowed us to rapidly add new network interfaces to our hierarchy. In addition, by depending only on packet reception, we can handle in an

identical way causes for disconnection other than mobility, such as the insertion or removal of PCMCIA network cards.

Our solution touches all four themes. Transparent vertical handoff is an agent-based dynamic adaptation to changes in network quality, including disconnection and the discovery of a better network. It uses cross-layer information both for horizontal handoff (in networks that we have control of the basestations) and for enabling upper layers to adapt to network changes (covered in Section 4).

This is among the earliest work in the architecture, and it does not uniformly exploit soft state. We do use soft state for the low-latency handoffs, since we can lose data at the buffering basestations. However, we use hard state for the forwarded connections, a decision we inherited from Mobile IP. Nonetheless, both still provide support for the power of soft state, as we encountered many problems that could have been avoided if we didn't have to maintain hard state at the agents. For example, there are complex race conditions between the recovery of hard-state at the agents (after a failure) and the mobile moving out of range that would not exist with soft state. If we redid the implementation today, we would use only soft state at the agents.

For more info: overlay networking [SK98][KB96]; low-latency handoffs [Ses95][SBK97]; power management [SGHK96].

3 Reliable Data Transport over Wireless Networks

Reliable data transport underlies many of the critical end-user applications supported in today's Internet, including file transfer, electronic mail, and access to the World Wide Web. It is extremely challenging to achieve high transport performance in an environment characterized by heterogeneous, wireless access technologies. Wireless links bring the challenges of 1) high and bursty losses in addition to network congestion, 2) asymmetric effects, and 3) unfair and poor utilization of shared links. In the next three sections, we discuss each of these challenges in more detail and present several novel solutions.

The de facto standard protocol for reliable data transport in the Internet is the Transmission Control Protocol (TCP). TCP provides a reliable byte-stream abstraction to the higher layers, delivering bytes in-order to the application. TCP has evolved to achieve excellent performance in conventional wired networks by adapting to end-to-end delays and congestion losses. For this purpose, TCP maintains a running average of the estimated round-trip delay and the mean linear deviation from it. The TCP sender uses the *cumulative acknowledgments (acks)* it receives to determine which packets have reached the receiver, and provides reliability by retransmitting lost packets. TCP uses these regular acknowledgments to pace out new data at a consistent rate, a procedure called *ack clocking*.

3.1 Combating Wireless Packet Losses

TCP is tuned to perform well in traditional networks where packet losses occur almost entirely due to congestion. However, networks with wireless or other lossy links also suffer from significant losses due to bit errors and handoffs.

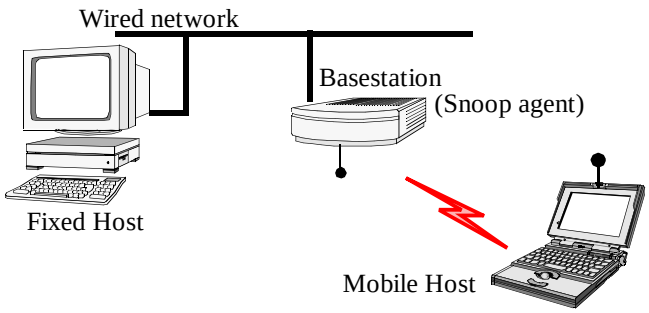

Figure 7: Network topology of the single-hop cellular wireless network based on the WaveLAN. The snoop agent runs at the basestations in the network.

TCP responds to all losses by invoking congestion control and avoidance algorithms, resulting in degraded end-to-end performance in wireless systems. In this section, we present the design of a transport-aware link protocol, the Berkeley Snoop protocol, to overcome these problems and shield fixed senders on the Internet from the vagaries of the wireless link. We also compare our protocol to alternatives to understand the precise reasons for improved performance.

3.1.1 The Snoop Protocol

The Snoop protocol is founded on three of our design themes: agent-based adaptation, cross-protocol interactions, in which the link agent is aware of transport semantics and messaging, and the use of soft state. Due to the agent approach, we require no modifications to TCP implementations on fixed hosts in the Internet, and we completely preserve the end-to-end semantics of TCP. The network topology and agent location are shown in Figure 7.

For transfer of data from a fixed host *to* a mobile host, we deploy a *snoop agent* at the basestation. The agent caches unacknowledged TCP segments and performs local retransmissions of lost segments, which it detects based on duplicate TCP acks from the mobile host and time-outs of locally maintained timers. Furthermore, it suppresses duplicate acknowledgments of wireless losses, so they are *not* seen by the TCP sender, thus shielding the sender from these non-congestion-related losses. In particular, the agent completely hides transient situations of very low communication quality and even temporary disconnection.

For data transfer *from* a mobile host to a fixed host, we exploit additional local knowledge because we control both the basestation and mobile sender. The snoop agent detects missing packets at the basestation by snooping on packets arriving from the mobile host and identifying holes in the transmission stream. By using heuristics based on local queue-length information, the agent classifies wireless losses as being due to congestion or corruption, and adds corruption-induced losses to a list of holes. When it eventually sees duplicate acknowledgments for these holes arriving from the fixed receiver, signifying a loss of this packet, it sets a bit in its TCP header and forwards it on to the mobile sender. The sender uses this *Explicit Loss Notification (ELN)* bit to identify that the loss was unrelated to congestion, and retransmits the packet without taking any congestion-control actions.

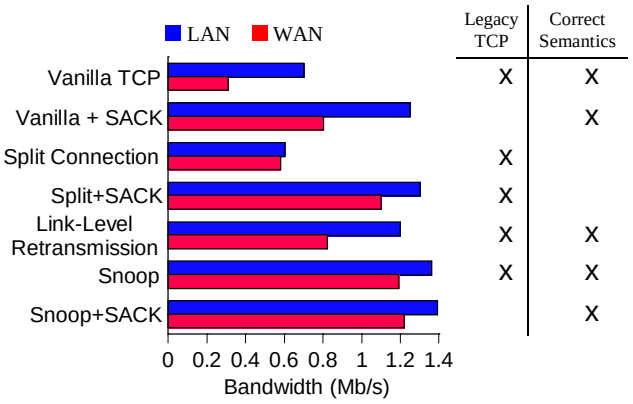

Figure 8: TCP Performance over Wireless Links

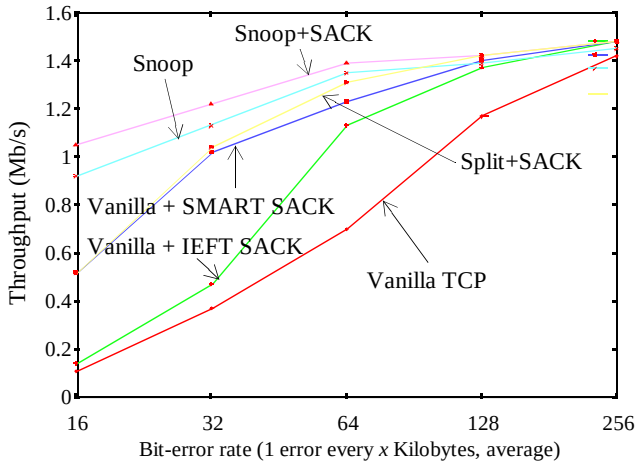

Figure 9: Performance of six protocols (LAN case) across a range of bit-error rates, ranging from 1 error every 16 KB to 1 every 256 KB shown on a log-scale.

3.1.2 Snoop Evaluation

A variety of protocols have been proposed in the literature to overcome the effects of wireless bit-errors on TCP performance. We classify these schemes into three broad categories: 1) end-to-end protocols, where loss recovery is performed by the sender; 2) link-layer protocols, that provide local reliability; and 3) split-connection protocols, that break the end-to-end connection into two parts at the basestation to shield the sender from the wireless link.

Figure 8 compares the solutions for performance and indicates which solutions work with legacy TCP stacks and which provide correct end-to-end semantics. The full papers on this topic examine more protocols and a wider array of bit-error rates, and describe the Snoop protocol in detail [BSK95] [BPSK97].

We focus on data transfer from a fixed host to a mobile host. Each test consists of an 8MB transfer over a combined network with a wired section and a single-hop wireless section using 915-MHz WaveLan. The LAN numbers use a single 10 Mb/s ethernet hop and have a bandwidth-delay product of about 1.9KB, while the WAN numbers use 16 Internet hops over the wide area, with a bandwidth-delay product of about 23KB. We ensured that there were no losses due to congestion to isolate corruption from congestion. Wireless losses were emulated by IP checksum corruption according to an exponential distribution of bit errors with an average bit-error rate of 1/64KB (about 1.9×10^{-6}), which corresponds to a packet loss rate of 2.3% with 1400-byte packets. Figure 9 graphs the performance against a range of bit-error rates.

End-to-end Protocols attempt to make the TCP sender handle losses through the use of two techniques. First, they use some form of selective acknowledgments (SACKs) to allow the sender to recover from multiple packet losses in a window without resorting to a coarse time-out. Second, they attempt to have the sender distinguish between congestion and other forms of losses using an *Explicit Loss Notification* (ELN) mechanism.

The simplest end-to-end approach is vanilla TCP (Reno), which performs poorly as expected, with only 50% of the possible throughput on the LAN and 25% on the WAN. The basic problem is that the window size remains small all of the time, and performance only gets worse with a larger bandwidth-delay product (as in the WAN case).

Adding selective acknowledges (SACKs) helps to some extent, but only when error rates are low or medium (see Figure 9). In the LAN case, we used a simple SACK scheme based on a subset of the SMART proposal [KM97]. This was the best of the end-to-end protocols that we tried, but it is still slower than the Snoop protocol. In the WAN case, we based our SACK implementation [BSK95] on RFC 2018, which was 65% of optimal. SACKs often allow the sender to recover from multiple losses without timing out. However, the sender's congestion window decreases every time there is a packet dropped on the wireless link, causing it to remain small. This is precisely the situation that the Snoop protocol avoids.

Split-connection approaches completely hide the wireless link from the sender by terminating the TCP connection at the basestation. The second connection can use techniques such as negative or selective acknowledgments, rather than standard TCP, to perform well over the wireless link. This class does not preserve the end-to-end semantics of TCP.

We present two examples: when the wireless connection uses vanilla TCP and when it uses the SMART-based SACK scheme described above. With vanilla TCP, the performance improvement is not as good as the other schemes, especially in the WAN case. The congestion window of the wired side remains large and stable compared to an end-to-end TCP connection where the congestion window size fluctuates rapidly. However, the wired connection doesn't make rapid progress because it periodically stalls due to the advertised window from the basestation being zero. When SACKs are used over the wireless connection, performance is improved. However, even this enhancement performs about 10% worse than Snoop, in addition to not providing correct end-to-end semantics and causing slower, more complicated handoffs.

Link-layer solutions attempt to hide link-related losses from the TCP sender by using local retransmissions and possibly forward error correction. Since the end-to-end TCP connection passes through the lossy link, the TCP sender

may not be fully shielded from wireless losses. This can happen either because of adverse timer interactions between the two layers [Cox95], or more likely because of TCP's duplicate acknowledgments causing fast retransmissions at the sender even for segments that are locally retransmitted.

Our base link-layer algorithm uses cumulative acknowledgments to determine losses and retransmits them locally from the basestation to the mobile host; it leverages TCP acknowledgments instead of generating its own. It is equivalent to the Snoop protocol except that it does not suppress duplicate acknowledgments. Thus, although this protocol successfully prevents sender time-outs, it does not prevent the sender from reducing its window in response to wireless losses. In the LAN experiments, the throughput difference is about 10%, but rises to 30% in the WAN case due to the higher bandwidth-delay product. The rapid fluctuations in sender window size lead to an average size that is smaller than the WAN bandwidth-delay product, which is the reason for the significant performance degradation. This situation is avoided by the TCP-aware Snoop protocol that suppresses duplicate acknowledgments. Finally, the best protocol, Snoop enhanced with SACK information, improves the performance of basic Snoop, especially in burst error situations (not shown, see [BPSK97]).

To summarize, Snoop enhanced with SACK provides the best performance among the protocols we studied, maintains end-to-end semantics and involves no changes to TCP stacks on the fixed hosts. Its *relative* performance improves significantly as the bit-error rate increases (up to 20 times improvement for the error rates we investigated), and also improves as the bandwidth-delay product increases. Our results also demonstrate that the end-to-end connection need not be split at the basestation to achieve good performance. SACKs help in recovering from burst losses, but are not sufficient when windows are small. Finally, the use of ELN helps in separating congestion control from loss recovery since the sender can now be made aware of non-congestion-related losses.

The snoop protocol clearly reinforces our focus on agent-based approaches since it works with legacy servers and provides the best performance, even over end-to-end solutions. Snoop exploits cross-layer information: in particular, we use both a transport-aware link layer (with Snoop) and a link-aware transport protocol (with ELN). The snoop agent also exploits soft state; its loss or corruption does not affect the protocol's correctness. We achieve this by taking advantage of TCP's cumulative acknowledgments, which makes it trivial to refresh out-of-date state. If the state associated with the protocol is lost, it can be rebuilt upon the arrival of the next packet and acknowledgment. The snoop protocol should thus be viewed as a performance booster for networks with lossy links. This contrasts it with hard-state approaches such as split connections.

Finally, the use of soft state dovetails nicely with the low-latency handoff scheme of Section 2.2. It takes advantage of the intelligent multicast buffering at the basestations, designed to achieve low-latency handoffs, to perform the mirroring of the Snoop agent's soft state. When a handoff completes, TCP connections thus resume with their snoop state intact, which further enables smooth handoffs and helps to hide the transient drop in performance from the sender.

3.2 Combating the Effects of Asymmetry

TCP relies on the ack clocking mechanism to pace data transmissions into the network. Although this leads to robust congestion control in most wired networking scenarios, it degrades performance in networks that exhibit asymmetry. A network is said to exhibit *asymmetric characteristics* with respect to TCP, if the data transport performance in one direction significantly affects the traffic and network characteristics in the opposite direction. The traffic in the reverse direction could just be the TCP acks. This general definition leads to three kinds of asymmetry:

Bandwidth: The bandwidth in the forward direction (towards the user) is 10 to 1000 times larger than that in the reverse direction. Networks based on Direct Broadcast Satellite (DBS) systems or wireless cable-modem technology exhibit such asymmetry.

Latency: In certain wireless networks, such as Cellular Digital Packet Data (CDPD), the underlying media-access control (MAC) protocol often results in a significantly larger one-way latency from a basestation to a mobile client than in the reverse direction. In half-duplex networks, such as the Metricom Ricochet multi-hop wireless network, there is a large variable delay when a node switches from sending to receiving mode or vice versa, a problem that arises in the presence of bidirectional traffic including TCP acks. These factors reduce overall throughput and increase the variation in round-trip times, which delays retransmissions.

Loss rates: Packet loss may be much greater upstream than downstream. This could be inherent in the network technology or the result of distinct upstream and downstream networks. Ack losses limit the performance of TCP in the primary direction.

We identify three specific challenges that network asymmetry presents for TCP performance. We discuss end-to-end techniques (i.e., enhancements to TCP) and agent-based techniques to alleviate these problems.

First, the ack stream corresponding to a data transfer could interfere with forward progress either 1) indirectly by the breakdown of ack clocking due to reverse-channel congestion or wireless losses, or 2) directly by competing with the flow of data packets (as in half-duplex networks like Metricom). Our basic solution is to decrease the frequency of acks. Our end-to-end approach, called *ack congestion control*, allows the receiver to reduce the frequency of acks adaptively, while ensuring that the sender is not starved for acks. This requires the TCP sender to tell the receiver the size of its window. Our agent-based approach, called *ack filtering*, operates at the router or basestation on the far side of the reverse channel. It exploits the cumulative nature of TCP acks to filter redundant acks from the queue when a new ack arrives for the same connection.

This decreased frequency of acks leads to the second problem: since each new ack could cause the sender's window to slide forward by several segments, the sender could emit a burst of several packets back-to-back, and induce congestion. Since the TCP sender increments its congestion

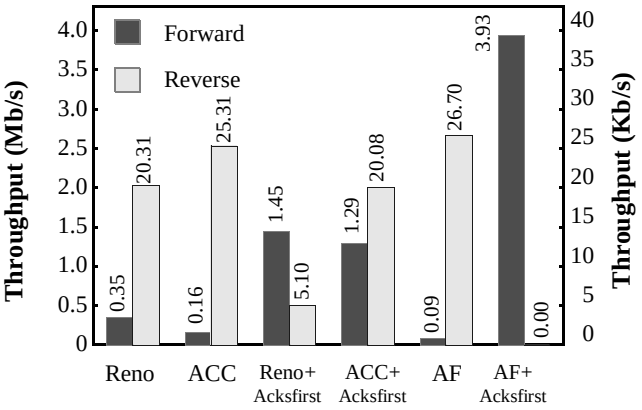

Figure 10: Throughput of a 3 MB forward transfer and a 100 KB reverse transfer; 10 Mb/s forward and 28.8 Kb/s reverse bandwidth. Note that the reverse throughput is 0 for the "AF+Acksfirst" case.

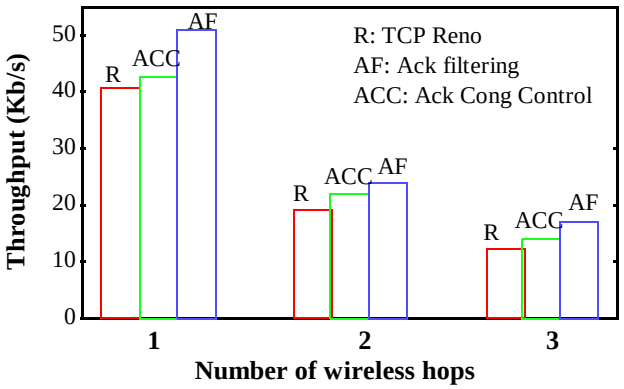

Figure 11: TCP throughputs from simulations of Reno, ACC and AF, as a function of the number of wireless hops. Simulation parameters are derived from the Metricom packet-radio network.

window based on the *number* of acks and not on how many bytes are acknowledged with each ack, window growth could be much slower. An end-to-end solution, called *sender adaptation*, breaks up a potentially large burst into several smaller bursts and regulates these smaller bursts according to the effective data transfer rate of the connection. Our agent-based solution, called *ack reconstruction*, hides the infrequent ack stream from the TCP sender by deploying an agent at the *near side* of the reverse channel. Once the acks have traversed the constrained network, the ack reconstructor regenerates the original smooth ack stream by inserting new acks to fill in gaps and by spacing apart acks that have been compacted. The end result is that an unmodified TCP sender can continue to rely on standard ack clocking to sustain the data transfer at a consistent rate.

The final problem arises specifically in an asymmetric-bandwidth situation where the acks of the forward-direction data transfer must share the constrained reverse channel with a reverse-direction data transfer (simultaneous bidirectional data transfers). With FIFO queuing, the large data packets of the reverse transfer could block the transmission of acks for long periods of time, thereby starving the sender of the forward transfer. Our agent-based solution to this problem, called *acks-first scheduling*, involves giving acks a strictly higher priority than data packets. The rationale is that the small acks packets when sent infrequently would have little impact on the performance of the reverse data transfer, but their timely transmission is critical to sustaining good performance for the forward data transfer. Since the resolution of this problem requires control over the scheduling of data and acks at the reverse bottleneck link, it is *not possible* to solve this problem without an agent.

We have performed a detailed evaluation of these solution techniques both via *ns* simulation as well as in our network testbed. These results are discussed in detail in [BPK97] and [BPK98]. We present here sample results for both the asymmetric-bandwidth and asymmetric-latency configurations. Figure 10 shows the performance of various schemes when there is two-way traffic in an asymmetric network. We make two observations: 1) acks-first scheduling helps the forward transfer significantly by avoiding the long delays on the reverse channel, and 2) ack congestion control (ACC) reduces the impact of forward acks on reverse transfers.

Figure 11 shows the performance of various schemes obtained via a simulation of the Metricom packet-radio network. The important observation is that ack filtering (AF) and to a lesser extent ack congestion control (ACC) result in improved performance. The reduced frequency of acks decreases MAC contention, reducing latency variation and eliminating overhead for switching directions. The performance improvement is between 20% and 35%.

To summarize, the set of end-to-end and agent-based techniques we have developed overcome the adverse effects of asymmetry in a variety of asymmetric situations.

As with Snoop, our solutions combine agent-based approaches with cross-layer information and extensive use of soft state. In the best, version combine two agents to provide good performance with legacy TCP endpoints: the agent on the far side of the asymmetric link provides ack compression and better ack scheduling relative to bidirectional transfers, while the near-side agent reconstructs a smooth ack stream to enable smooth ack clocking in unmodified senders. Both of these agents exploit cross-layer information and assumptions (such as importance of a smooth ack stream) as well as soft state for simple implementations that improve performance while maintaining correctness even in the presence of faults.

3.3 Summary

In summary, our general approach to improving data transport performance in mobile, wireless networks has been to identify the specific characteristics of such networks, such as wireless errors, asymmetry, and unfairness, that impact TCP performance, and then to explore both TCP enhancements and agent-based mechanisms that alleviate them.

The agent-based approaches are often as good or better than end-to-end TCP modifications, and provide the critical additional advantage of immediate deployment, thus allowing us to integrate wireless technologies cleanly into the rest of the global Internet. The keys to making agents work well

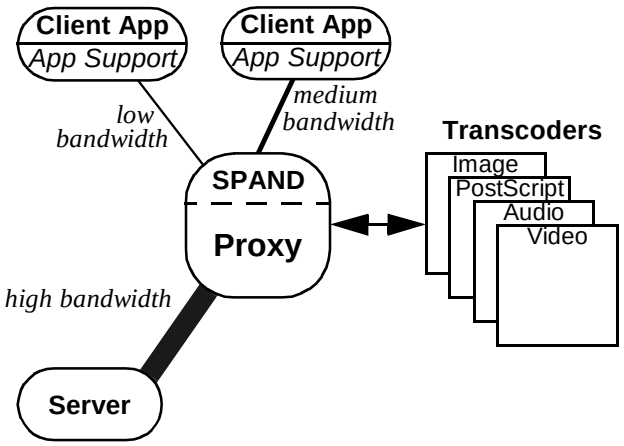

Figure 12: Basic proxy architecture. The proxy uses the transcoders to optimize the quality of service for the client in real time. SPAND monitors end-to-end bandwidth and connectivity to the clients (and servers) and notifies the proxy of any changes, which may result in changes in transcoding to adjust the quality of service.

are to exploit cross-layer information, such as the ack semantics of TCP, and to build them based on soft state for simplicity, scalability (since there is no consistency issue), and availability (since they are trivially restartable). Finally, agent-based approaches are the only solution when we need to exploit cross-layer information from multiple connections, as illustrated by ack-first scheduling for asymmetric links.

For more info: Wireless TCP [BPSK97], Snoop and handoffs [BSK95]; handling asymmetry [BPK98].

4 The Proxy Architecture

The key to applications in a mobile, wireless, very heterogeneous environment is the proxy architecture, which uses a proxy as a smart intermediary between traditional servers and heterogeneous mobile clients. The fundamental driver for this architecture is the inability of (nearly all) servers to handle the incredible variation in software, hardware and networking capabilities of mobile clients. Through various forms of data transformation, the proxy can dynamically translate between the needs of clients and the formats and abilities of the servers. The basic architecture is shown in Figure 12.

In addition to the wide variation in networks covered in previous sections, there are also significant variations in software and hardware. Software variations include the data formats that a client can handle, such as JPEG, PostScript and video formats, and also protocol issues such as HTTP/ HTML extensions and multicast support. Hardware variations are summarized in Table 2.

The key idea in on-demand transformation is *distillation*, which is highly lossy, real-time, datatype-specific compression that preserves most of the semantic content of a document or stream. For example, a color image can be scaled

Platform	SPECint92/ Memory	Screen Size	Bits/pixel
High-end PC	198/64M	1280x1024	24, color
Midrange PC	157/8M	1024x768	16, color
Midrange laptop	51/8M	640x480	8, color
Palm Pilot	low/1M	160x160	2, gray

Table 2: Physical Variation Among Clients

	On-Demand Image Transformation	Real-Time Video Transcoding
Hardware	Reduce image resolution, color depth, convert to gray-scale or monochrome with halftoning.	Reduce resolution, color depth, frame rate, frame quality (e.g. quantization factors), convert to gray-scale or monochrome with halftoning.
Software	Format conversion: GIF, JPEG, PICT, Pilot format, PostScript; multicast to unicast conversion	Format conversion: NV, H.261, Motion JPEG, MPEG, InfoPad VQ [N+96]; multicast to unicast conversion
Network	Dynamic network variation due to RF effects or overlay handoffs.	Dynamic network variation due to RF effects or overlay handoffs; variation in jitter or latency.

Table 3: Dynamic adaptation in response to variations across client hardware and software, and dynamic network conditions.

down and the number of colors reduced, thereby reducing the size of the representation. For video, we can reduce the frame size, frame rate, color depth, and resolution to create a smaller but reduced-quality representation. Intimate knowledge of each datatype allows distillation to achieve tremendous compression, sometimes better than a factor of ten, that preserves much of the semantic content of the original.

The first purpose of the distilled object is to allow the user to receive *some* version of an object quickly; they can then evaluate the value of downloading the original, or some part of the original. For instance, zooming in on a section of a graphic or video frame, or rendering a particular page containing PostScript text and figures without having to render the preceding pages. We define *refinement* as the process of fetching some part (possibly all) of a source object at increased quality, possibly the original representation.

The second and more powerful purpose is to *enable new abilities* in clients, which arises out of dynamic adaptation to formats and data rates that the client can handle. For example, we have enabled: 1) web browsing with graphics on the PalmPilot (covered in Section 4.4.1), 2) PostScript viewing via distillation to HTML, 3) MBone video over ISDN, transcoded from 400 Kb/s down to 128 Kb/s; and 4) participation in multicast-based white-board sessions on the PalmPilot, which supports neither multicast nor the basic

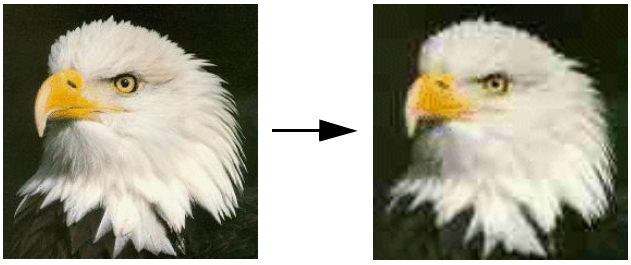

Figure 13: Scaling this JPEG image by a factor of two in each dimension and reducing JPEG quality to 25% results in a size reduction from 10KB to 1.5KB.

white-board formats such as GIF images. These examples illustrate the power of dynamic adaptation via an agent (the proxy), and show how very simple clients can leverage a powerful infrastructure to achieve capabilities that they could not achieve on their own.

4.1 On-Demand Transformation

Although we have looked at several data types for on-demand transformation, we will focus on just two examples, images and video, since they are challenging and significantly reduce the required bandwidth. Table 3 shows several kinds of dynamic adaptation that the proxy can make in response to hardware, software and network heterogeneity.

The image transcoder takes several input formats, including GIF and JPEG, and produces an optimized output. Figure 13 shows an example transformation. The output depends on the network conditions and the client's hardware and software capabilities. Typically we use a format that is both small and easy for the client to render.

The graphs in Figure 14 depict end-to-end client latency for retrieving the original and each of four distilled versions of a selection of GIF images. The images are fetched using a 14.4 modem through the Berkeley PPP gateway, via a proxy that runs each image through a GIF distiller. Each bar is segmented to show the distillation latency and transmission latency separately. Clearly, even though distillation adds latency at the proxy, it results in greatly reduced end-to-end latency, in addition to better looking output for that client.

As with image transformation, video-stream transcoding mitigates all three forms of client variation. Figure 15 exhibits the tradeoffs that can be made in video distillation. The video transcoder, called *vgw*, handles several video formats and can transcode video in real time at frame rates up to 30 frames/second [AMZ95]. Since each frame is converted in real time, the proxy can make frame-by-frame tradeoffs and can respond very dynamically to changing network conditions. As an example of a new ability enabled by the architecture, vgw makes it possible to watch MBone broadcasts over ISDN via adaptation from about 400 Kb/s to 128 Kb/s.

In both cases, the original servers are unaware of the transformations or of the limited capabilities of the clients or networks. Thus we can combine very heterogeneous clients and networks with sophisticated legacy servers that target stationary desktop PCs and workstations.

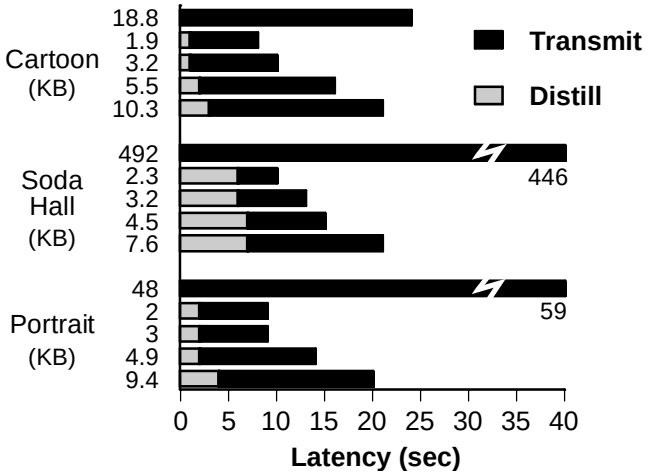

Figure 14: End-to-end latency for images with and without distillation. Each group of bars represents one image with 5 levels of distillation; the top bar represents no proxy at all. The y-axis number is the distilled size in kilobytes (so the top bar gives the original size). Note that two of the undistilled images are off the scale; the Soda Hall image is off by an order of magnitude. These numbers are from a Sun SparcStation 20.

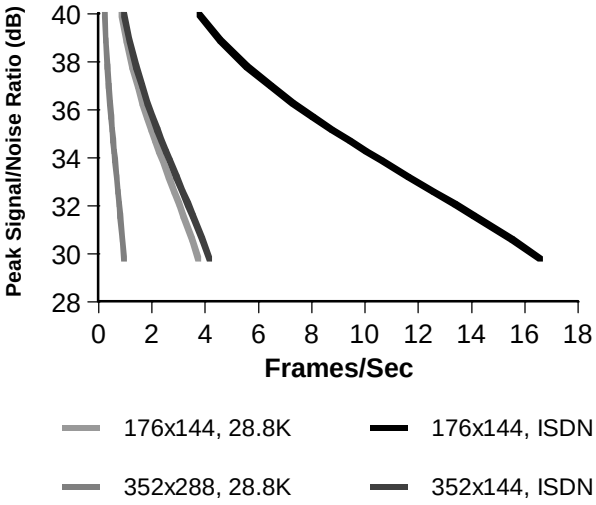

Figure 15: The four-way tradeoff among bandwidth, frame resolution, quality (measured in peak signal/noise ratio), and frames per second. Dynamic adaptation enables fine-grain control of these tradeoffs, and can provide consistent quality or frame rates even as network conditions vary over time.

4.2 Measuring Changes in Network Characteristics

To use transcoding to adapt to network variation, the proxy must have an estimate of the current network conditions along the path from the proxy to the client. We have developed a network measurement system called SPAND [SSK97], *Shared Passive Network Performance Discovery*, that allows a measurement host to collect the actual application-to-application network performance (e.g., available bandwidth and latency) between proxies and clients. The

Service: Service-specific code
- Workers that present human interface to what TACC modules do, including device-specific presentation
- User interface to control the service
- complex services: implement directly on SNS layer.

TACC: Transformation, Aggregation, Caching, Customization
- API for composition of stateless data transformation and content aggregation modules
- Uniform caching of original, post-aggregation and post-transformation data
- Transparent access to Customization database

SNS: Scalable Network Service support
- Incremental and absolute scalability
- Worker load balancing and overflow management
- Front-end availability, fault tolerance mechanisms
- System monitoring and logging

Figure 16: Scalable Network Service Layered Model

proxy consults this database of performance information to determine the appropriate amount of distillation to use for a particular client. We have also added hooks for clients or third parties, such as the overlay networking software, to notify the proxy of changing network conditions. All new transformations compute the degree of distillation based on these new capabilities.

4.3 Scalable Cluster-Based Proxies

Because the proxy is a permanent part of the network infrastructure, it must meet all of the *utility requirements* of the rest of the network, which include very high availability, ease of management, and scalability to support millions of users. In addition, these capabilities are difficult to obtain, so we would like to isolate the utility requirements from the functionality of the proxy; this allows authors of proxy-based services to focus on the *content* of their service rather than on the challenges of availability and scalability.

Towards this end, we break the proxy into three distinct layers, as shown in Figure 16. The lowest layer, *Scalable Network Service* (SNS) support, provides all of the availability, scalability, load balancing and monitoring functionality. The middle layer, *TACC*, described below, provides a programming model for authors that simplifies service creation and that hides the details of the SNS layer, thus providing scalability and availability "off the shelf." The top layer handles the actual presentation of data and client protocols such as HTTP, and manages all of the open connections. Typically service authors need only write TACC modules to implement their service, the SNS and service layers provide everything else.

To explore these issues, we built and deployed a scalable transformation proxy called TranSend, which is available publicly at `http://transend.cs.berkeley.edu`. It follows the layered architecture and all of its functionality resides in the TACC layer via a collection of about a dozen modules. It has had over 10,000 unique users since its launch

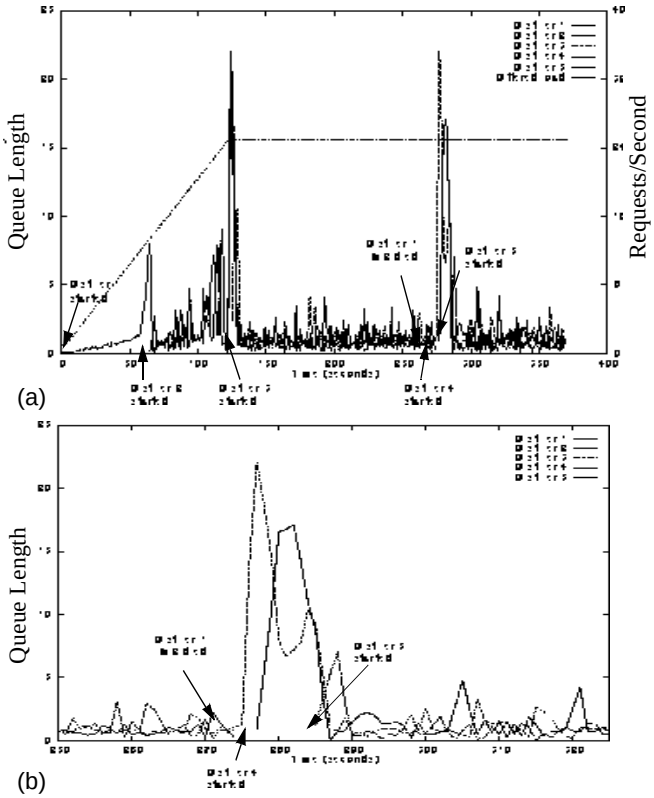

Figure 17: Distiller queue lengths observed over time as the load presented to the system fluctuates, and as distillers are manually brought down. (b) is an enlargement of (a).

in April 1997, and many people use it all of the time even from wired networks such as modems and ISDN.

The SNS layer is implemented on a network of workstations to provide these benefits; the architecture and analysis of this layer are beyond the scope of this paper, but are covered in [FGCB97]. Using a cluster of ten Sun 140-MHz Ultra-1 workstations, we were able to show that even the image-transformation part of TranSend, which is relatively heavy-weight, scales linearly to handle over 170 transformations per second, which is sufficient for more than 10,000 simultaneous users, and more than 10 million transformations/day.

We have also demonstrated very high availability, with very low downtime since the TranSend service began operation. The fault-recovery capabilities enabled this uptime without any dedicated system-administration staff. Figure 17 shows a trace of the availability, load balancing, and burst-recovery mechanism in action. Failed TACC workers are cleanly restarted in a few seconds and new workers are automatically deployed as load rises; together these mechanism ensure low latency (short queue lengths), load balancing, and excellent overall response time, even with faults.

Finally, we found that the cost of the computing infrastructure is extremely low, about 25 cents per user per month for all of the equipment [FGCB97]. The basic reason for better cost performance is the statistical multiplexing enabled by moving resources from client devices into the infrastructure, where they can be shared. For example, ISPs typically have

1/20th as many modems as their customers, even though every connection requires one on each end. Similarly, computing power in the infrastructure is more than twenty times more efficient simply because it is not idle 80% of the time (or more), as in mobile devices, which are often off and have bursty utilization even when in use [GB97].

4.4 Generalized Proxies: The TACC Model

So far, we have motivated the proxy architecture as a mechanism for on-demand transformation. However, the proxy architecture is far more general and powerful. In particular, we can generalize the proxy into a powerful platform for composable infrastructure services.

This more general model is called *TACC*, for transformation, aggregation, caching and customization. As described above, *transformation* is an operation on a single data object that changes its content; examples include filtering, format conversion, encryption, and compression.

Aggregation involves collecting data from several objects and collating it in a prespecified way. For example, one aggregator collects all listings of cultural events from a prespecified set of Web pages, extracts the date and event information from each, and combines the results into a dynamically generated "culture this week" page. The combination of transformation and aggregation by TACC workers results in an extremely general programming model; in fact, transformation proxies [Bro+96], proxy filters and caches, customized aggregators [SB97], and search engines are all subsumed by the TACC model.

Caching plays an interesting role in TACC, because in addition to caching original Internet content, the caches in our architecture can store post-transformation data, post-aggregation content and even intermediate results. Like all caches, its contents are entirely soft state that can be regenerated at the expense of network latency or computation.

Customization allows many users to share a smaller number of services, by parameterizing the services and recalling each user's parameters as they invoke the service. This is similar to allowing each user to pass different "command-line arguments" to the service. The *customization database*, in most respects a traditional ACID database, maps a user-identification token, such as an IP address or HTTP cookie, to a list of key-value parameters.

A key strength of our architecture is the ease of *composition* of tasks; this affords considerable flexibility in the transformations and aggregations the service can perform, without requiring workers to understand task-chain formation, load balancing or burst management., any more than programs in a Unix pipeline need to understand the implementation of the pipe mechanism. We now look at some illustrative examples of TACC programs that reveal the power of worker reuse and composition.

4.4.1 Top Gun Wingman

Bringing web access to resource-poor devices such as PDAs, smart phones, or set-top boxes is fundamentally difficult because these devices are ill-suited to the tasks of parsing, decoding, and rendering web content. These devices do not directly support web data types, and the development and runtime environments of the devices preclude their implementation. Even if data types such as GIF or JPEG images are supported, most Web images remain awkward to use due to the limitations in the screen, the memory/storage capacity, and the CPU performance. For example, the 3Com PalmPilot has a 160x160x2 gray-scale display, 1MB of NVRAM, and a 16-MHz Motorola Dragonball processor.

The proxy architecture solves this problem by moving nearly all of the complexity and cost into the infrastructure, where we can apply multiple workstations and leverage powerful libraries and development environments. By shifting complexity off of the impoverished device to the TACC server, we were able to implement the *Top Gun Wingman* web browser on the Palm Pilot, the first (and so far only) graphical web browser for the Pilot. Wingman has more than 10,000 regular users.

The client-side code knows nothing about the web—it merely knows how to receive and render various high-level data types such as bitmapped images and formatted text segments. All of the HTTP/HTML processing, image parsing, image distillation, and page layout are implemented in the TACC server as a pipeline of workers. The pipeline's output is a ready-to-render page, with both graphics and text.

Our partitioning of the web browser has enabled a number of innovative features that are not possible without the proxy architecture:

Accelerated Download Time: one worker in the Wingman TACC pipeline performs the same on-the-fly image distillation that we described above. This feature combined with more the efficient representation of the entire page as a single easy-to-parse data block (versus multi-connection HTML) makes download times on Top Gun Wingman faster than even text-only PDA web browsers.

Transparent Functionality Upgrades: because most of the web browser is implemented in the TACC server, it has been possible for us to upgrade the browser's functionality without redistributing software to all of the clients. For example, when we implemented a better graphics transformer that performed high-frequency enhancement of images to increase edge visibility, we simply dropped in this worker as a replacement to the old image transformer, and all of our clients instantaneously began receiving higher quality images.

Aggregation Services: the Wingman browser has access to all of the aggregation services implemented in our TACC server, since they compose. We have also built and deployed several aggregators that present condensed versions of popular search engine and directory services that are better suited to the limited screen real estate on the Palm Pilot.

Support for Complex Data Types: there are many data types that would be simply infeasible to handle on the client side, such as ZIP files and PostScript. However, we *can* handle these data types in the TACC server. For example, we implemented a worker that accepts a ZIP compressed archive, and presents its contents to the browser as a list of hyperlinks. When the user selects a link, the TACC worker extracts the relevant object, performs any required data-type specific transformations, and then sends only that transformed element to the client. Since Pilot applications are

Figure 18: Top Gun MediaBoard allows the PalmPilot to participate in shared-whiteboard sessions, despite its small display and inability to support the Scalable Reliable Multicast protocol on which the original MediaBoard is based.

normally stored as ZIP files, this allows users to install applications directly off of the Internet with a single click.

4.4.2 Other TACC Services

In addition to Top Gun Wingman, there are many other TACC-based services. To illustrate the variety of applications, we highlight a few that are being deployed at Berkeley for widespread public use.

Group Annotation for the Web: Inspired by prior work on annotating the Web [SMB96], we built a scalable, highly available group-annotation service. The annotations are kept in a separate database and are combined with source documents on the fly via transformation; this allows annotation of any page, even without write access to the original. Controls are added with annotations, which allow the user to invoke a Java-based annotation viewer/editor.

Top Gun MediaBoard: MediaBoard is a modular reimplementation of the shared Internet whiteboard *wb* [JM95]. Top Gun MediaBoard [CF97] is a TACC-based implementation of this program for the PalmPilot, which is partitioned so that the heavyweight Scalable Reliable Multicast (SRM) protocol is run at the TACC server, by a MASH worker representing a particular *mb* session. A compact unicast datagram protocol is used for communication between the PalmPilot clients and the TACC server, which also does coordinate transformations and image transformations (reusing many of the Wingman TACC modules) corresponding to the MediaBoard draw operations occurring in the session. Top Gun MediaBoard thus performs both data adaptation and protocol adaptation, making it a prime example of the flexibility of the TACC toolkit for building adaptive applications. Figure 18 shows a screenshot of the TGMB client.

Charon Indirect Authentication: We have built a proxied version of Kerberos IV, called *Charon* [FG96], for use with resource-poor clients such as PDAs and smart phones. This partitioning allows an *untrusted* proxy to interact with the Kerberos infrastructure on behalf of the clients, giving the clients full authenticated (and optionally encrypted) access to the services but relieving them of the significant burden of having a full Kerberos implementation. Even though nearly all of the Kerberos functionality is implemented by the proxy, neither the user's Kerberos password nor their keys for new services ever leave the client device, thus enabling use of an untrusted proxy. Charon has the same immunity to protocol-based attacks as Kerberos, and is more immune to certain end-to-end attacks because of the nature of the client devices. In fact, Charon turns such devices into a powerful form of "smart card."

The success of the TACC platform in supporting Wingman and other applications is strong evidence for its role as an enabling technology for deploying complex, scalable, highly available, and composable services.

4.5 Summary

We have presented a uniform architecture that addresses extreme variation in the areas of client hardware, software, and connectivity, via powerful dynamic adaptation of protocols and content. Because the architecture is structured around agent-based on-demand transformation, we require no changes to the existing infrastructure and servers.

We use cross-layer information to control the degree of distillation very dynamically based on current network conditions. For example, as a mobile user walks outside, not only do we perform a seamless vertical handoff, but we also notify the proxy of the change in network quality. In response the proxy increases the amount of distillation, thus maintaining the same response time (within about 10-20%) even though the network conditions have changed by more than an order of magnitude in bandwidth and latency (assuming a handoff from WaveLAN to Metricom).

Although not discussed, the cluster-based proxy implementation makes extensive use of soft state [FGCB97], including all of the management and worker state except for user profiles, which need to be persistent. In fact, we originally implemented the manager with mirrored hard state and failover, but later replaced it with a much simpler and more effective soft-state based approach.

The proxy architecture includes many significant contributions:

- On-demand transcoding reduces end-to-end latency, often by an order of magnitude, for a wide range of network media. Dynamic adaptation allows clients to *refine* content selectively and spend more bandwidth for improved quality as needed, e.g., zooming in on part of an image or video frame.
- Knowledge of client limitations allows transcoders to optimize content for each client: images, video, and rich text all look significantly better after distillation because they target the client's optimal colormap and screen fonts. This property leaves servers free to target the high end.
- Knowledge of changing network conditions, via cross-layer information, allows adaptation of transformation parameters to maximize the overall quality of service, including both representation quality and performance.
- Moving complexity into the infrastructure enables significant new abilities for clients, as illustrated by Wingman, Charon indirect authentication, and MBone participation over ISDN, among many other examples.
- The TACC model simplifies development of complex services, and hides the very challenging problems of scalability, availability and load balancing.

- We have deployed for public use several TACC-based services that not only provide compelling services, but also show that the proxy architecture really can enable both powerful new services and a very heterogeneous mobile and wireless infrastructure.

For more info: transformation [FGBA96]; video transcoding [AMZ95]; cluster-based services [FGCB97]; TACC applications [UCB97]; network monitoring and SPAND [SSK97].

5 Adaptive Network Services

The Network Services component of the architecture builds on our routing, transport, and the proxy architecture. It provides support for locale-specific discovery of available resources and "on-the-move" reconfiguration of, and adaptation to, these resources. It additionally provides mobile clients with a generic "remote control" for their environment. This supports seamless interaction with the current environment as clients roam in overlaid networks.

We have developed for our building the following example services:

- untethered interaction with lights, video and slide projectors, a VCR, an audio receiver, and other A/V equipment from a wireless laptop computer;
- automatic "on-the-move" reconfiguration for use of local DNS/NTP/SMTP servers and proxies;
- audited local printer access;
- visualization of physical geography and object locations via a protocol for interactive floor maps;
- advertising available local services (or other data) to unregistered ("anonymous") clients.

Our premise is that providing basic IP access to clients is not enough. We must augment basic IP connectivity with an adaptive *network services infrastructure*. Distillation proxies are one example of a network service that may require discovery and reconfiguration. The challenge is to develop an *open* service architecture that allows heterogeneous clients to discover what they can do in a new environment and interact with it while making minimal assumptions about standard interfaces and control protocols.

We provide mechanisms for 1) service discovery, 2) remote-control interfaces, and 3) access control based on physical locality.

5.1 Service Discovery

The service discovery protocol allows for the maintenance of dynamic repositories of service information, advertises its availability, and supports queries against it. The Service Location Protocol [VGPK97], under development by the IETF Service Location working group, is one proposal for implementing local-area service discovery.

We have implemented local-area service discovery and are in the process of extending it for use in the wide-area. The latter introduces additional challenges: scalability, security, hierarchies, and multiple administrative domains.

Local-area service discovery is based on *embedding location information into existing data networks* rather than using external information such as GPS coordinates. Outfitted subnets periodically broadcast a beacon packet to the local subnet broadcast addresses (and in some cases multicast with limited scope).

The local server for a client's current attachment point is encoded in the beacons. Notifications of a change are implicit. Further service information is obtained by querying this *service discovery server*. The list of local services can be obtained by requesting the well-known meta-service "INDEX". This minimal bootstrap standardizes the mechanism without constraining its interpretation, thereby allowing variation in resource discovery protocols as they evolve. We unified the service discovery bootstrap and mobility beacons by observating that mobility beaconing is simply a form of announce-listen protocol. The announce rate is one that provides acceptable handoff latency.

Client-based beaconing is used in both the Active Badge [HH94] and ParcTab systems [SAG+93], while Mobile IP uses infrastructure-based beaconing, as do we. This is the correct design choice for three reasons: the support for user (rather than infrastructure) anonymity, better bandwidth scalability in a network where there are many MHs per BS, and because power is more precious on mobile devices.

There is one additional field in the beacon packets: an application-specific payload. This provides two advantages. First, it makes data available to mobiles *before* registration (in the Mobile IP sense), allowing the possibility of "anonymous" access to this broadcast data. Second, it acts as a vehicle for announce/listen protocols that require only limited announcement timer adaptability (discrete multiples of the base beaconing rate), such as control applications. We currently piggyback a list of some or all of the available services in the cell, allowing anonymous assessment of them and reducing discovery latency.

5.2 Remote Control

Providing users with local services exposes new challenges. One is supporting group meetings in which some participants are physically clustered together in a meeting room and others attend from distributed desktops via remote conferencing tools.

For a room participant to interact with the local instantiations of the conferencing tools yet still participate in the meeting, it is necessary to extend these (local) tools so they can be controlled remotely. This requirement has two parts: it must be possible to control the devices in the room (i.e., local services for each), and it must be possible for participants to control these things *while participating* in a meeting.

To address these requirements, we extend both room devices and existing collaborative applications with shared, simplified, remote-control interfaces. These interfaces maintain two key departures from the current suite of collaboration tools. Rather than focus on use at the desktop, with local control of each application through its user interface (a one-to-one mapping of participants to controllers), we generalize this mapping, allowing a group of users to *share* control

through the use of their personal mobile devices. Additionally, we specify a set of "standard configurations" of these applications and devices, thereby reducing the required complexity. Details of our experience with "remote-controlled vic" (rvic) are available [HN97].

An important area of ongoing work is in *interface specification languages (ISL)*, which we are using as a way to generate a functional user interface for devices and services that have "never met". That is, given an ISL for a service and a simple device specification, we can generate the UI for that service/device pair automatically. An extreme version of this would allow a service to have the same ISL for web use, for touch-tone telephones, and for speech-based control.

5.3 Scoping Access Control

Making services available to visitors raises general security issues, including those specific to the wireless domain [Bro95]. Local services require additional access control, because we extend local services for remote use. Global access control is necessary, but not sufficient; the expected behavior that environmental changes can only be affected by people *in that environment* has been broken. For example, lights should not able to be turned off by a person across the country. Maintaining this expectation is important when extending existing social metaphors into an environment with controllable objects.

We address this by embedding *tickets*, random fixed-length bit vectors, in the mobility beacons and requiring the current ticket to be included in all communications to servers. Periodically changing the tickets in an unpredictable way and limiting the scope of the broadcast (e.g., to the local subnet) prevents remote access from nodes *on the access control list* that are not actually local. This ticket-based exclusion can be overridden by explicitly giving remote participants the ticket. However, by restricting access by default, we better emulate the existing paradigm.

5.4 Adaptive Network Services Summary

In summary, the adaptive network services layer of the architecture provides *functionality beyond IP dial-tone* to both roaming and local users. The themes that guide our work in this area include:

- Taking advantage of the availability of non-intrusive mobile client devices to allow users to use *their personal devices* in a manner comfortable to them;
- Making the environment *responsive and customizable* via remote-control access to devices and the to the services that make the devices useful; and
- Utilizing *dynamic discovery and reconfiguration* at multiple levels (service availability, service interfaces) to address the inherent variability exposed in mobility-enabled environments.

For more info: localized services [HKSR97][HK97]

6 Discussion

Returning to a broad view of the architecture, we examine some of the issues that we did not address, contrast BARWAN with related work, and revisit our four themes.

6.1 Key Issues not Addressed by BARWAN

There are several important areas that we did not directly address in the architecture. First, we limited ourselves to off-the-shelf networks and mobile devices. Although we added agents, we did not modify the base software for our networks, nor did we develop any new networking hardware. We have learned several lessons that should enable better future networks, such as the importance of network-interface power management, the negative impact of half-duplex networking, and the importance of augmented infrastructure-based beacons.

Similarly, we avoided developing our own mobile device, in part in response to the significant effort involved with the InfoPad project. However, we view the current set of devices as sufficient only for prototyping, and have high expectations for the next generation, which should be designed with the proxy architecture in mind. We have had discussion with device manufacturers along these lines.

We also dealt only tangentially with the infrastructure required for global authentication. The Charon work enables proxy-based authentication, and we can restrict access to local users via beacon-based tickets, but those both prematurely assume that a global authentication system is in place. As such systems are deployed, our architecture extends them to support heterogeneous clients and local services.

Finally, an important area of ongoing work is fairness and utilization for shared wireless networks such as wireless LANs. In addition to traditional MAC-layer issues, wireless networks suffer from time-varying network performance per host and little or no mechanisms for collision detection. We are looking at channel dependent link-scheduling algorithms based on class-based queueing to mitigate these effects.

Finally, we did not look at disconnected operation, instead assuming connectivity of varying quality. We believe that disconnected operation is an important capability for mobile users; we left it out only because there are several major projects that address the issues [MES95][Kle97][JTK97], and because we believe that those systems are complementary to our work.

6.2 Related Projects

There are at least several hundred useful papers that relate to some aspect of the architecture; here we focus only on broad projects of similar scope. For detailed coverage of related work in a specific area, please refer to the papers listed in the "**For More Info"** list at the end of the corresponding section.

The Coda and Odyssey projects at CMU focus on file-system based support for mobile devices. The Coda system [MES95] manages file-system consistency across intermittent and disconnected access. BARWAN did not look at disconnected access, although we believe the Coda approach is largely orthogonal and thus could be supported directly. Odyssey [NSN+97] originally defined multiple quality levels

of file-system objects to support variation in network quality; their current approach adopts our notion of dynamic adaptation and distillation (and the use of proxies) to provide both application-aware and application-transparent adaptation at the file-system level. Much of the rest of the BARWAN architecture is complementary, such as overlay networking and reliable data transport, and in fact there has been some joint work [NKNS96]. One major distinction is the service focus of BARWAN, rather than the file-system focus of Odyssey, which enables not only localized services, but also complex TACC-based services (and protocols) such as the Wingman browser.

The InfoPad project [Bre+96][N+96] preceded BARWAN, had some overlap in personnel, and was very influential on several of our network and application-support elements. The InfoPad is a hand-held wireless tablet that depends on both high-bandwidth networking (for bitmapped graphics and video) and extensive application support in the infrastructure. For example, it depends on a modified version of X Windows on the server that renders all of the *pixels* (not vector graphics) over the wireless connection. Although very bandwidth intensive, this approach has the benefit of simple low-power hardware, since it requires only simple processing on the device. A major distinction is thus our focus on low-bandwidth links and some processing power on the devices. We can also view the InfoPad as just another heterogeneous device; for example, the video transcoder can convert video streams into InfoPad's custom VQ format.

UCLA's work on nomadic computing [Kle97] addresses many of the same issues as BARWAN. In particular, it shares the service focus of BARWAN. It provides strong support for wide-area data management (predictive caching and data consistency) and disconnected operation. However, it does not (so far) address the issues of overlaid networks, reliable data transport, or scalable complex infrastructure services. The proxy architecture has emerged as fundamental to the nomadic computing.

Rover [JTK97] is a toolkit for developing mobile applications that support disconnected operation. As with Coda, the Rover work is complementary, and we expect to integrate the Rover technology into our future work in this area. The proxy architecture and network services work simplify the infrastructure side of the Rover toolkit.

6.3 Four Themes Revisited

We finish our discussion by revisiting our four themes.

Dynamic Adaptation is the general way we deal with varying network conditions and with client heterogeneity. Vertical handoff is a dynamic adaptation in response the presence or absence of packets on a particular overlay, while horizontal handoff is an adaptation based on physical-layer feedback such as signal strength.

The proxy architecture is an extreme form of dynamic adaptation: we change the content dynamically depending on network conditions, device software and hardware capabilities, and user preferences. We demonstrated dynamic distillation for both images and real-time video streams, in both cases enabling smooth tradeoffs between quality and performance across a range of devices. Similarly, by defining remote-control interfaces via an ISL, we can dynamically adapt the user interface to the capabilities of the client device. These techniques give us tremendous flexibility and are the key to supporting networks and clients that vary by orders of magnitude across many dimensions.

Cross-layer Optimization: There are several places where we explicitly break the OSI networking model to enable smarter adaptation or better performance. For example, we use physical-layer information to trigger (horizontal) handoffs.

A compelling example is the interaction of the transport and link layers in our solutions to reliable data transport over wireless and asymmetric links. Snoop uses detailed TCP knowledge to detect time-outs and duplicate acks, and it suppresses duplicate acks to the sender to control when the sender enters congestion control. Similarly, the ELN mechanism allows the transport layer to know that a particular loss was not due to congestion. The control of ack clocking in our support for asymmetric connections is another example of controlling the transport layer from the link layer; we reduce acks on the reverse channel and then reconstruct them for smooth flow from legacy TCP senders. We also exploit transport-layer information when we use "acks first" scheduling, since we have to understand the purpose of the packets. In all cases, we achieve significantly better performance and maintain the end-to-end semantics.

Finally, we use transport-layer information to control the proxy (essentially the presentation layer). This enables dynamic adaptation and thus maximizes the tradeoff between quality and performance (response time for HTTP requests; frame-rate for video streams), even in the presence of varying network conditions.

Agent-Based Capabilities: To maximize our ability to work with legacy clients, servers and TCP stacks, we use agents within the network infrastructure to add new capabilities or to improve performance. Fundamentally, the role of these agents is to hide to vagaries of mobility, wireless networking and extreme heterogeneity from all of the legacy servers and infrastructure; the agents make everything look like it is "supposed to". The alternative is take an end-to-end approach in which you are allowed to modify all components, including clients and servers. This has the advantage of a potentially better overall solution, but it is nearly impossible to deploy, since it requires upgrading most clients and servers.

The overlay networking solution uses multiple agents, one at the home base to forward packets, and one near the mobile device to handle low-latency handoffs locally. These agents primarily hide mobility from legacy servers by acting as a level of indirection. The local agent also enables fast handoffs by maintaining state at multiple basestations via local multicast.

In the case of transport issues, the agent-based approaches usually performed as well or better than the end-to-end approaches, and provided the critical additional advantage of immediate deployment, thus allowing us to integrate wireless technologies cleanly into the rest of the global Internet. The snoop agent hides wireless losses from TCP senders, thus enabling them to work well for wireless

links. Similarly, we use a pair of agents to hide significant problems with reverse channels in asymmetric links from TCP, again enabling new levels of performance for those links without modifying all of the servers on the Internet. For both acks-first scheduling and shared wireless links, we showed that agents can perform link optimizations across a collection of connections, which is not possible without a local presence at the link or with just enhancements to the TCP protocol.

The most powerful agents, of course, are the cluster-based TACC servers, which can provide mobile users with the combined power of several workstations (even for a single HTML page). We demonstrated many capabilities that are not possible without the proxy, and produced several applications that depend on tremendous leverage from the infrastructure to meet deliver their functionality, most notably the PalmPilot web browser and whiteboard applications. Even for web browsing from home, the proxy enables 3-7 times faster downloads and the receipt of (degraded) MBone video. We expect powerful proxies to be fundamental to future internet access.

Exploit Soft State: Soft state is state that aids in performance or adds capabilities, but that is not required for correctness. Soft state need not be recovered after failures, can be thrown out at any time, and can typically be stale or inconsistent without violating correctness. In general, most of our agents use soft state to make them simpler and trivially fault tolerant. We exploit soft state in all aspects of the design, and in the few places that we didn't use it, we either regretted it (overlay networking) or rewrote the subsystem to use soft state (TACC manager).

Fundamentally, soft state goes well with agents because the latter are typically added on to systems that already have a solution for end-to-end data integrity (such as TCP). Thus agents tend to be both optional and not responsible for the state of the system that they extend. Limiting the agent's state to soft state is thus not that difficult, and it enables trivial fault recovery and scalability. Perhaps more importantly, soft state tends to avoid many difficult issues regarding fault tolerance and data consistency, thus making the implementation of the agents far more simple overall. For example, even though we use two agents to manage the acks on a asymmetric link, our use of soft state means that either agent (or both) can fail without effecting the correctness of the connection. In fact, they can restart in any or order or not at all and still maintain correctness. Finally, soft state works well for agents because it focuses on availability rather than consistency, which is the opposite of most fault-tolerant systems, such as ACID databases.

7 Conclusions

The BARWAN project achieved nearly all of its goals: we deployed a working network that really is useful. It is possible to roam among several networks and the performance of the networks is quite good. The proxy is highly available and supports thousands of users, several quality services, and a wide range of mobile devices; two of these applications (TranSend and Wingman) have thousands of users each. The support for localized network services enables automatic configuration of mobile devices so they can exploit local resources, and the remote control functionality makes it possible to control your environment with your personal PDA, even if you've never been in the room before. These are powerful capabilities; we look forward to the day when they are regularly available to everyone.

8 References

[AMZ95] E. Amir, S. McCanne and H. Zhang. An Application-Level Video Gateway. In *Proc. of ACM Multimedia '95*, November 1995.

[ATT94] AT&T Corporation. *WaveLAN: PC/AT Card Installation and Operation*. AT&T manual, 1994.

[Bre+96] E. Brewer *et al.* The Design of Wireless Portable Systems [InfoPad]. *Proc. of Spring COMPCON '95*, March 1995.

[Bro+96] C. Brooks, M.S. Mazer, S. Meeks and J. Miller. *Application-Specific Proxy Servers as HTTP Stream Transducers*. Proc. WWW-4, Boston, May 1996. http://www.w3.org/pub/Conferences/WWW4/Papers/56.

[BSK95] H. Balakrishnan, S. Seshan, and R.H. Katz. Improving Reliable Transport and Handoff Performance in Cellular Wireless Networks. *ACM Wireless Networks*, 1(4), December 1995.

[BPK97] H. Balakrishnan, V. N. Padmanabhan and R. H. Katz. The Effects of Asymmetry on TCP Performance. In *Proc. of ACM Mobicom '97*, September 1997.

[BPK98] H. Balakrishnan, V. N. Padmanabhan and R. H. Katz. The Effects of Asymmetry on TCP Performance. In *ACM Mobile Networking (MONET), Special Issue on Mobile Networking in the Internet*, Spring 1998

[BPSK97] H. Balakrishnan, V. N. Padmanabhan, S. Seshan and R. H. Katz. A Comparison of Mechanisms for Improving TCP Performance over Wireless Links. *IEEE/ACM Transactions on Networking*, December 1997.

[Bro95] D. Brown. Techniques for Privacy and Authentication in Personal Communication Systems. *IEEE Personal Communication*s, pp. 6–10, August 1995.

[CF97] Y. Chawathe and S. Fink. ***MASH Meets TACC: Top Gun MediaBoard.*** **http://http.cs.berkeley.edu/~yatin/courses/isvc/report/.**

[Cox95] D. C. Cox. Wireless Personal Communications: What is it? *IEEE Personal Communications*, pages 20–34, April 1995.

[Dir98] The DirecPC Home Page. http://www.DirecPC.com, 1998.

[FG96] A. Fox and S. Gribble. Security on the Move: Indirect Authentication Using Kerberos, *Proceedings of MobiCom '96*, November 1996.

[FGBA96] A. Fox, S. Gribble, E. Brewer and E. Amir. Adapting to Network and Client Variability via On-Demand Dynamic Distillation, *Proceedings of ASPLOS-VII*, October 1996.

[FGCB97] A. Fox, S. Gribble, Y. Chawathe and E. Brewer. Cluster-Based Scalable Network Systems, *Proceedings of SOSP '97*. October 1997.

[GB97] S. Gribble and E. Brewer. System Design Issues for Internet Middleware Services: Deductions from a Large Client Trace. *Proc. of the USITS '97*, December 1997.

[HH94] A. Harter and A. Hopper. A Distributed Location System for the Active Office. *IEEE Network Magazine*, 8(1), January 1994.

[HK97] T. Hodes and R. Katz. "Composable Ad-hoc Mobile Services for Universal Interaction", *Proceedings of MobiCom '97*, September 1997.

[HKSR97] T. D. Hodes, R. Katz, E. Servan-Schreiber, and L. A. Rowe. Composable Ad-hoc Mobile Services for Universal Interaction. *Proc. of The Third ACM/IEEE International Conference on Mobile Computing*. Budapest, Hungary, September, 1997.

[HN97] T. Hodes and M. Newman, *rvic: A vic Variant for Remote Control of Shared Video Monitors*, http://http.cs.berkeley.edu/~hodes/rvic/index.html.

[IBM95] IBM Infrared Wireless LAN Adapter Technical Reference. IBM Microelectronics - Toronto Lab, 1995.

[JM95] V. Jacobson and S. McCanne. *Public Domain Network "Whiteboard"*. ftp://ftp.ee.lbl.gov/conferencing/wb.

[JTK97] A. D. Joseph, J. A. Tauber and M. Frans Kaashoek. Mobile Computing with the Rover Toolkit. *IEEE Transactions on Computers: Special issue on Mobile Computing*, 46(3). March 1997.

[Kle97] L. Kleinrock. Nomadicity: Anytime, Anywhere in a Disconnected World, *Mobile Networks and Applications*, 1(4), pp. 351–357, January 1997.

[KB96] R. H. Katz and E. A. Brewer. The Case for Wireless Overlay Networks. In *Proc. of the SPIE Multimedia and Networking Conference (MMNC'96)*, San Jose, CA, January 1996.

[KG97] M. Kornacker and R. Gilstrap. Group Annotation Using TACC. http://kemet.berkeley.edu/ratogi/class/cs294-6.

[KM97] S. Keshav and S. Morgan. SMART Retransmission: Performance with Overload and Random Losses. In *Proc. of INFOCOMM '97*.

[MES95] L. B. Mummert, M. R. Ebling and M. Satyanarayanan. Exploiting Weak Connectivity for Mobile File Access. *Proc. of the 15th ACM Symposium on Operating Systems Principle*s, December 1995.

[Met96] Metricom web page. http://www.metricom.com, 1996.

[N+96] S. Narayanaswamy *et al.* Application and Network Support for InfoPad. *IEEE Personal Communications*, March 1996.

[NKNS96] G. T. Nguyen, R. H. Katz, B. Noble, and M. Satyanarayanan. A Trace-Based Approach for Modelling Wireless Channel Behavior. *Proc. of the Winter Simulation Conference*, December 1996.

[NSN+97] B. Noble, M. Satyanarayanan, D. Narayanan, J. E. Tilton, J. Flinn, and K. Walker. Agile Application-Aware Adaptation for Mobility. *Proceedings of SOSP '97*, St. Malo, France, October 1997.

[Per96] C. Perkins. *IP Mobility Support*. RFC 2002, October 1996.

[SAG+93] B. N. Schilit, N. Adams, R. Gold, M. Tso, and R. Want. The PARCTAB Mobile Computing System. In *Proceedings of the Fourth Workshop on Workstation Operating Systems (WWOS-IV)*, October 1993.

[SB97] B. Schilit and T. Bickmore. Digestor: Device-Independent Access to the World Wide Web. *Proc. of WWW*-6, Santa Clara, CA, April 1997.

[SBK97] S. Seshan, H. Balakrishnan, and R. H. Katz. Handoffs in Cellular Wireless Networks: The Daedalus Implementation and Experience. *Kluwer Journal on Wireless Personal Communications*, January 1997.

[Ses95] S. Seshan. *Low Latency Handoffs in Cellular Data Networks*. Ph.D. thesis, University of California at Berkeley, December 1995.

[SGHK96] M. Stemm, P. Gauthier, D. Harada, and R. H Katz. Reducing Power Consumption of Network Interfaces in Hand-Held Devices. In *Proc. of the Third Workshop on Mobile Multimedia Communications (MoMuC-3)*, September 1996.

[SK98] M. Stemm and R. H. Katz. Vertical Handoffs in Wireless Overlay Networks. In *ACM Mobile Networking (MONET), Special Issue on Mobile Networking in the Internet*, Spring 1998.

[SMB96] M. A. Schickler, M. S. Mazer, and C. Brooks. Pan-Browser Support for Annotations and Other Meta-Information on the World Wide Web. *Proc. of the Fifth International World Wide Web Conference*, May 1996.

[SSK97] S. Seshan, M. Stemm, and R. H. Katz. SPAND: Shared Passive Network Performance Discovery. In *Proc. of the First Usenix Symposium on Internet Technologies and Systems (USITS '97)*. December 1997.

[UCB97] University of California at Berkeley, *CS294-6: Internet Services*, Fall 1997. Class proceedings at http://www.cs.berkeley.edu/~fox/cs294.

[VGPK97] J. Veizades, E. Guttman, C. Perkins and S. Kaplan. *Service Location Protocol, Internet Draft #17*, IEFT 1997. draft-ietf-srvloc-protocol-17.txt.

[Wei93] M. Weiser. Some Computer Science Issues in Ubiquitous Computing. *Communications of the ACM*, 36(7), July 1993.