Proceedings of HotOS VIII,
Elmau/Oberbayern, Germany,
May 2001.

Reconsidering Internet Mobility

Alex C. Snoeren, Hari Balakrishnan and M. Frans Kaashoek *

MIT Laboratory for Computer Science
{snoeren, hari, kaashoek}@lcs.mit.edu

Abstract. *Despite the popularity of mobile computing platforms, appropriate system support for mobile operation is lacking in the Internet. This paper argues this is not for lack of deployment incentives, but because a comprehensive system architecture that efficiently addresses the needs of mobile applications does not exist. We identify five fundamental issues raised by mobility—location, preservation of communication, disconnection handling, hibernation, and reconnection—and suggest design guidelines for a system that attempts to support Internet mobility.*

In particular, we argue that a good system architecture should (i) eliminate the dependence of higher protocol layers upon lower-layer identifiers; (ii) work with any application-selected naming scheme; (iii) handle (unexpected) network disconnections in a graceful way, exposing its occurrence to applications; and (iv) provide mobility services at the mobile nodes themselves, rather than via proxies. Motivated by these principles, we propose a session-oriented, end-to-end architecture called Migrate*, and briefly examine the set of services it should provide.*

1 Introduction

The proliferation of laptops, handheld computers, cellular phones, and other mobile computing platforms connected to the Internet has triggered much research into system support for mobile networking over the past few years. Yet, when viewed as a large-scale, heterogeneous,

* This research was funded by DARPA (Grant No. MDA972-99-1-0014), NTT Corporation, Intel, and IBM. Alex C. Snoeren is supported by a National Defense Science and Engineering Graduate (NDSEG) Fellowship

distributed system, the Internet is notoriously lacking in any form of general support for mobile operation.

We argue that previous work has failed to comprehensively address several important issues. This paper discusses some of these issues and describes a session-oriented architecture we are developing to preserve end-to-end application-layer connectivity under various mobile conditions.

Mobility raises five fundamental problems:

1. **Locating the mobile host or service:** Before any communication can be initiated, the desired end-point must be located and mapped to an addressable destination.
2. **Preserving communication:** Once a session has been established between end points (typically applications), communication should be robust across changes in the network location of the end points.
3. **Disconnecting gracefully:** Communicating applications should be able to rapidly discern when a disconnection at either end, or a network partition, causes communication to be disrupted.
4. **Hibernating efficiently:** If a communicating host is unavailable for a significant period of time, the system should suspend communications, and appropriately reallocate resources.
5. **Reconnecting quickly:** Communicating peers should detect the resumption of network connectivity in a timely manner. The system should support the resumption of all previously established communication sessions without much extra effort on the part of the applications.

Most current approaches provide varying degrees of support for the first two problems. The last three—disconnection, hibernation, and reconnection—have received little attention outside of the file system context [19]. We argue that a complete—and useful—solution must address all these issues.

One need look no further than interactive terminal applications like `ssh` or `telnet`, one of the Internet's oldest applications, for a practical example of the continuing lack of support for these important components. A user with an open session might pick up her laptop and disconnect from the network. After traveling for some period of time, she reconnects at some other network location and expects that her session continue where it left off. Unfortunately, if there was any activity on the session during the period of disconnectivity, she will find the connection aborted upon reconnection to the network. The particular

details of the example are irrelevant, but demonstrate just how lacking current support is, even for this simple scenario.

Based on our own experience developing various mobile protocols and services [3, 5, 14, 26] and documented reports of several other researchers over several years [9, 13, 15, 18, 28], we identify four important guidelines that we believe should be followed as hints in designing an appropriate network architecture for supporting mobile Internet services and applications:

1. *Eliminate lower-layer dependence from higher layers.* A large number of problems arise because many higher layers of the Internet architecture use identifiers from lower layers, assuming they will never change during a connection.
2. *Do not restrict the choice of naming techniques.* Dynamic naming and location-tracking systems play an important role in addressing mobility. In general, whenever an end point moves, it should update a naming system with its new location—but forcing all applications to use a particular naming scheme is both unrealistic and inappropriate.
3. *Handle unexpected disconnections gracefully.* We advocate treating disconnections as a common occurrence, and exposing them to applications as they occur.
4. *Provide support at the end hosts.* Proxies are attractive due to their perceived ease of deployment. However, it becomes markedly more difficult to ensure they are appropriately located when hosts are mobile.

We elaborate upon these guidelines in Section 2. They have served as a guide in our development of an end-to-end, session-oriented system architecture, called *Migrate*, over which mobile networking applications and services can be elegantly layered. We describe our proposed architecture in Section 3, discussing how it addresses four of the five problems mentioned above: preserving communication, and handling disconnection, hibernation, and resumption. We do not provide or enforce a particular location or naming scheme, instead leveraging domain-specific naming services (e.g., DNS, service discovery schemes [3, 12], etc.) for end-point location.

An attractive feature of our architecture is that it accomplishes these tasks without sacrificing common-case performance. Migrate provides generic mechanisms for managing disconnections and reconnections in each application session, and for handling application state and context. We briefly discuss related work in Section 4 before concluding in Section 5.

2 Design guidelines

In this section, we elaborate on our four design guidelines for supporting applications on mobile hosts.

2.1 Eliminate lower-layer dependence

The first step in enabling higher-layer mobility handling is to remove inter-layer dependences. In a 1983 retrospective paper on the DoD Internet Architecture, Cerf wrote [8]: "TCP's [dependence] upon the network and host addresses for part of its connection identifiers" makes "dynamic reconnection" difficult, "a problem ... which has plagued network designers since the inception of the ARPANET project in 1968." The result is that when the underlying network-layer (IP) address of one of the communicating peers changes, the end-to-end transport-layer (TCP) connection is unable to continue because it has bound to the network-layer identifier, tacitly (but wrongly) assuming its permanence for the duration of the connection.

A host of other problems crop up because of similar linkages. For example, the increasing proliferation of network address translators (NATs) in the middle of the network has caused problems for applications (like FTP) that use network- and transport-layer identifiers as part of their internal state. These problems can be avoided by removing any assumption of stability of lower-layer identifiers. If a higher layer finds it necessary to use a lower-layer identifier as part of its internal state, then the higher layer should allow for it to change, and continue to function across such changes.

Furthermore, each layer should expose relevant changes to higher layers. In today's Internet architecture, applications have almost no control over their network communication because lower layers (for the most part) do not concern themselves with higher-layer requirements. When important changes happen at a lower layer, for example to the network-layer address, they are usually hidden from higher layers. The unfortunate consequence of this is that it makes it hard for any form of adaptation to occur.

For example, a TCP sender attempts to estimate the properties of the network path for the connection. A significant change in the network-layer attachment point often implies that previously discovered path properties are invalid, and need to be rediscovered. This consequence is not limited to classical TCP congestion management—for example, if mobile applications are notified of changes in their environment and given the power to effect appropriate changes, significant

improvements in both performance and usability can be realized [19, 21]. Similar results have also been shown in the network layer [9, 13, 30], and in the area of transport optimization over wireless links [5, 7, 26].

2.2 Beware the Siren song of naming

Many researchers have observed that the first problem raised by mobility, namely locating the mobile host or service, can be addressed through a sophisticated naming system, hence most proposals for managing Internet mobility attempt to provide naming and location services as a fundamental part of the mobility system.[1] Unfortunately, the tight binding between naming schemes and mobility support often causes the resulting system to be inefficient or unsuitable for various classes of applications. For example, Mobile IP assumes that the destination of each packet needs to be *independently* located, thereby necessitating a home agent to intercept and forward messages to a mobile host. The utility of alternative proposals to use agile naming [3] or IP multicast [20] for mobility support hinges on widespread deployment of their location systems.

We believe that inexorably binding mobility handling with naming unnecessarily complicates the mobility services, and restricts the ability to integrate advances in naming services. On the face of it, it appears attractive that a "good" naming scheme can provide the level of indirection by which to handle mobility. In practice, however, it is important to recognize and separate two distinct operations. The first is a "location" operation: The process of finding an end point of interest based on an application-specific name. The second is a "tracking" operation: Preserving the peer-to-peer communication in some way. There are two problems with using a new idealized naming scheme: First, there are a large number of ways in which applications describe what they are looking for, which forces this ideal naming scheme to perform the difficult task of accommodating them all. Experience shows that each application is likely to end up using a naming scheme that best suits it (e.g. INS, DNS, JINI, UPnP), rather than suffer the inadequacies of a universal one. Second, if this tracking is done through the same name resolution mechanism, every packet would invoke the resolution process, adding significant overhead and degrading performance.

We therefore suggest that an application use whichever naming scheme is sufficiently adept at providing the appropriate name-to-loca-

[1] Indeed, the authors of this paper are guilty of having taken this position in the past.

tion binding in a timely fashion. This service is used at the beginning of a session between peers, or in the (unlikely) event that all peers change their network locations "simultaneously." At all other times, the onus of preserving communication across moves rests with the peers themselves. In the common case when only a subset of the peers moves at a time, the task of reconnection is efficiently handled by the peers themselves. We have previously described the details of such a scheme in the context of TCP connection migration [26].

2.3 Handle unexpected disconnections

The area of Internet mobility that has received the least attention is support for efficient disconnection and reconnection. While significant work has been done in the area of disconnected file systems [15, 19], less attention has been paid to preserving application communication when a disconnection occurs, enabling it to quickly resume upon reconnection. The key observation about disconnections is that they are usually unexpected. Furthermore, they last for rather unpredictable periods of time, ranging from a few seconds to several hours (or more). Today's network stacks terminate a connection as soon as a network disconnection is detected, with unfortunate consequences—the application (and often the user) has to explicitly reinitiate connectivity and application state is usually lost.

Like all other aspects of network communication, we believe the system should therefore provide standard support for unexpected disconnection, enabling applications to gracefully manage session state, releasing system resources and reallocating them when communication is restored. Even if the duration of the disconnection period is short enough to avoid significantly impacting communication or draining system resources, the disconnection and ensuing reconnection events are often hidden by current network stacks, leaving the higher network layers and application to eventually discover (often with unfortunate results) that network conditions have changed dramatically.

2.4 Provide services at the end points

A great deal of previous work in mobility management has relied on a proxy-based architecture, providing enhanced services to mobile hosts by routing communications through a (typically fixed) waypoint that is not collocated with the host [5, 10, 11, 17, 22, 28]. It is often easier to deploy new services through a proxy, as the proxy can provide enhanced services in a transparent fashion, inter-operating with legacy

systems. Unfortunately, in order to provide adequate performance, it is not only necessary to highly engineer the proxy [17], but locate the proxy appropriately as well.

Several researchers have proposed techniques to migrate proxy services to the appropriate location, avoiding the need to preconfigure locations [10, 27]. Unfortunately, all candidate proxy locations must be appropriately preconfigured to participate. Further, in the face of general mobility, proxies (or at least their internal state) must be able to move with the mobile host in order to remain along the path from the host to its correspondent peers. This is a complex problem [28]; we observe that it can be completely avoided if the support is collocated with the mobile host itself.

3 Migrate approach

We now describe the Migrate approach to mobility, which leverages application naming services and informed transport protocols to provide robust, low-overhead communication between application end points. We describe a session-layer protocol that handles both changes in network attachment point and disconnection in a seamless fashion, but is flexible enough to allow a wide variety of applications to maintain sufficient control for their needs.

3.1 Service model

The number of communication paradigms in use on the Internet remains small, but the type and amount of mobility support needed varies dramatically across modalities [9]. In particular, the notion of a session is application-dependent and varies widely, from a set of related connections (e.g. FTP's data and control channels) to an individual datagram exchange such as those often found in RPC-based applications (e.g. a cached DNS response). As the lengths of sessions grow longer and they become more complex in terms of the system resources they consume, applications can benefit from system support for robust communication between application end points. However, due to the disparate performance and reliability requirements of different session-based applications, it is important that a mobility service enables the application to dictate its requirements through explicit choice of transport protocols and policy defaults.

Hence we propose an optional session layer. This layer presents a simple, unified abstraction to the application to handle mobility: a ses-

sion. Sessions exist between application-level end points, and can survive changes in the transport, network, and even other session layer protocol states. It also includes basic check-pointing and resumption facilities for periods of disconnection, enabling comprehensive, session-based state management for mobile-aware applications. Unlike previous network-layer approaches, our session layer exports the specifics of the lower layers to the application, and provides an API to control them, if the application is inclined to do so.

3.2 Session layer

Applications specify their notion of a session by explicitly joining together related transport-layer connections (or destinations in connectionless protocols). Once established, a session is identified by a locally-unique token, or Session ID, and serves as the system entity for integrated accounting and management. The session layer exports a unified session abstraction to the application, managing the connections as a group, adapting to changes in network attachment point as needed. The selection of network end point and transport protocol, however, remains completely under the application's control.

To assist in the timely detection of connectivity changes, the session layer accepts notification from lower layers (e.g., loss of carrier, power loss, change of address, etc.), the application itself, or appropriately authorized external entities that may be concurrently monitoring connection state [4]. Since a session may span multiple protocols, connections, destinations, and application processes, there may be several sources of connectivity information. Regardless of the source, the session manager handles notification of disconnection and reconnection in a consistent fashion.

3.2.1 Disconnection. If a host can no longer communicate with a session end point due to mobility, as signaled by changes in the network layer state, transport layer failure, or other mechanisms, it informs the application. If the application is not prepared to handle intermittent connectivity itself, the session layer provides appropriate management services, depending on the transport layers in use, including data buffering for reliable byte streams. Specifically, it may block or buffer stream sockets, selectively drop unreliable datagrams, etc. Additional application and transport-specific services can be provided, such as disabling TCP keep-alives.

Depending on the system configuration, the session layer may need to actively attempt to reestablish communication, or it may be notified by network or transport layers when it becomes available again. Sys-

tem policy may dictate trying multiple network interfaces or transport protocols. In either case, if the period of disconnection becomes appropriately long (as determined by system and application configuration), it will attempt to conserve resources by reducing the state required in the network, transport, and session layers (with possibly negative performance implications upon reconnection), and notify the application, enabling additional, domain-specific resource reallocation.

3.2.2 Reconnection. Upon reattachment, a mobile host contacts each of its correspondent hosts directly, informing them of its new location. Some transport layers may be unable to adequately or appropriately handle the change in network contexts. In that case, the session layer can restart them, using the session ID to re-sync state between the end points. In either case, the session layer informs the application of reattachment, and resynchronizes the state of the corresponding session layers.

The complexity of synchronization varies with the transport protocols in use; a well-designed transport layer can handle many things by itself. By using a transport-layer token, and *not* a network layer binding, the persistent connection model can provide limited support for changes in attachment point, often with better performance than higher-layer approaches [23, 26]. Similarly, the performance of even traditional transport protocols can be enhanced when the network layer exposes the appropriate state [5, 7]. Similarly, grouping multiple transport instances between the same end points into sessions can provide additional performance improvement [4, 24].

Legacy transport protocols may be completely unable to handle changes in network addresses. In that case, the session layer may initiate an entirely new connection, and resynchronize them transparently at the session layer. In the worst case, the application itself may be unable to handle unexpected address changes, and provide no means of system notification. Such applications are still supported via IP encapsulation. The correspondent session layers establish an IP tunnel to the new end point, and continue to send application data using the old address.

If a correspondent end point is no longer reachable (possibly because the other end point also moved), the application is instructed to perform another naming/location resolution operation in attempt to locate the previous correspondent, returning a network end point (host, protocol, port) to use for communication. The particular semantics of suitable alternative end points and look-up failure are application specific. It may be a simple matter of another application-layer name resolution (perhaps a fresh DNS query), or the application may which

wish to perform its own recovery in addition to or in place of reissuing the location query.

While the amount overhead varies with the capabilities of the available lower layer technologies, overhead is incurred almost exclusively during periods of disconnectivity and reconnection. This provides high performance for the common case of communication between static peers.

3.3 State management

In a spirit similar to Coda, our architecture considers disconnection to be a natural, transient occurrence that should be handled gracefully by end hosts. For extended periods of disconnection, resource allocation becomes an additional concern. While managing application state is outside the scope of our architecture, enabling efficient strategies is decidedly not. In particular, since disconnection often occurs without prior notice, applications may require system support to reclaim resources outside of their control.

There has been a great deal of study on application specific-methods of dealing with disconnected or intermittent operation. Most of it has focused on providing continued service at the disconnected client, and has not addressed the scalability of servers. If our approach becomes popular, and disconnected sessions begin to constitute a non-negligible fraction of the connections being served, servers will need to free resources dedicated to those stalled connections, and be able to easily reallocate them later. We are considering a variety of state management services the session layer should implement, and briefly hypothesize about two: migrating session state between the system and application, and providing contextual validation of session state.

3.3.1 State migration. We believe the session abstraction may be a useful way to compartmentalize small amounts of connection state, reducing the amount of state applications need to store themselves, and simplifying its management. Furthermore this state could be tagged as being associated with a particular communication session, and managed in an efficient fashion together with system state [6]. System support may allow intelligent paging or swapping of associated state out of core if the period of disconnection becomes too long.

3.3.2 Context management. There is a significant amount of context associated with a communication session, and it may be the case that some (or all) of it will be invalidated by disconnection and/or reconnection. In particular, previous work has shown that context changes in the transport layer can be leveraged to adapt application

protocol state [25]. Hence any state the session layer manages needs to be revalidated, possibly internally, possibly through application-specific up-calls. Changes in context may dictate that buffers be cleared, data be reformatted, alternate transport protocols be selected, etc. This requires a coherent contextual interface between the application and the session layer.

4 Related work

The focus of the Migrate architecture is on preserving end-to-end application communication across network location changes and disconnections. Much work has been done in the area of system support for mobility over the past few years; this section outlines the work most directly related to ours.

At the network-layer, several schemes have been proposed to handle mobile routing including Mobile IP [22] and multicast-based mobility [20]. Mobile IP uses a home agent as to intercept and forward packets, with a route optimization option to avoid triangle routing. The home-agent-based approach has also been applied at the transport layer, as in MSOCKS [17], where connection redirection was achieved using a split-connection proxy, providing so-called transport-layer mobility. Name resolution and message routing were integrated to implement a "late binding" option that tracks highly mobile services and nodes in the Intentional Naming System [3].

Most TCP-specific solutions for preserving communication across network-layer changes [23, 26] do not handle the problems associated with connections resuming after substantial periods of disconnectivity. A "persistent connection" scheme where the communication end-points are location independent was proposed for TCP sockets and DCE RPC [29], but the mapping between global endpoint names and current physical endpoints is done through a global clearinghouse, which notifies everyone of binding updates. Session layer mobility [16] explored moving entire sessions by utilizing a global naming service to provide endpoint bindings; address changes are affected through a TCP-specific protocol extension.

5 Conclusion

In this paper, we have defined five salient issues concerning host mobility in the Internet. We presented a set of design guidelines for building a system to address these issues, distilled from a decade of research in

mobile applications and system support for mobility on the Internet. Following these principles, we outlined *Migrate*, a basic session-based architecture to preserve end-to-end application-layer communication in the face of mobility of the end points. We believe the general abstractions for disconnection, hibernation, and reconnection provided by the session layer define an appropriate set of interfaces to enable more advanced system support for mobility.

References

1. September 1997.
2. March 2001.
3. William Adjie-Winoto, Elliot Schwartz, Hari Balakrishnan, and Jeremy Lilley. The design and implementation of an intentional naming system. In *Proc. ACM SOSP*, pages 186–201, December 1999.
4. Hari Balakrishnan, Hariharan Rahul, and Srinivasan Seshan. An integrated congestion management architecture for Internet hosts. In *Proc. ACM SIGCOMM*, pages 175–187, August 1999.
5. Hari Balakrishnan, Srinivasan Seshan, and Randy H. Katz. Improving reliable transport and handoff performance in cellular wireless networks. *ACM Wireless Networks*, 1(4):469–481, December 1995.
6. Gaurav Banga, Peter Druschel, and Jeffrey C. Mogul. Resource containers: A new facility for resource management in server systems. In *Proc. USENIX OSDI*, pages 45–58, February 1999.
7. Ramon Caceres and Liviu Iftode. Improving the performance of reliable transport protocols in mobile computing environments. *IEEE JSAC*, 13(5):850–857, June 1995.
8. Vinton G. Cerf and Edward Cain. The DoD Internet architecture model. *Computer Networks*, 7:307–318, October 1983.
9. Stuart Cheshire and Mary Baker. Internet mobility 4x4. In *Proc. ACM SIGCOMM*, pages 318–329, August 1996.
10. Mike Dahlin, Bharat Chandra, Lei Gao, Amjad-Ali Khoja, Amol Nayate, Asim Razzaq, and Anil Sewani. Using mobile extensions to support disconnected services. Technical report, UT Austin, April 2000.
11. Mark Gritter and David Cheriton. An architecture for content routing support in the internet. In *Proc. 3rd USITS* [2], pages 37–48.
12. E. Guttman, C. Perkins, J. Veizades, and M. Day. Service Location Protocol, Ver. 2. RFC 2608, June 1999.
13. Jon Inouye, Jim Binkley, and Jonathan Walpole. Dynamic network reconfiguration support for mobile computers. In *Proc. ACM/IEEE Mobicom* [1], pages 13–22.
14. Anthony D. Joseph, Joshua A. Tauber, and M. Frans Kaashoek. Mobile computing with the rover toolkit. *IEEE Trans. on Computers*, 46(3):337–352, March 1997.

15. James Kistler and M. Satyanarayanan. Disconnected operation in the Coda filesystem. In *Proc. ACM SOSP*, pages 213–225, October 1991.
16. Bjorn Landfeldt, Tomas Larsson, Yuri Ismailov, and Aruna Seneviratne. SLM, a framework for session layer mobility management. In *Proc. IEEE ICCCN*, pages 452–456, October 1999.
17. David Maltz and Pravin Bhagwat. MSOCKS: An architecture for transport layer mobility. In *Proc. IEEE Infocom*, pages 1037–1045, March 1998.
18. Dejan Milojicic, Frederick Douglis, and Richard Wheeler. *Mobility: Processes, Computers, and Agents*. Addison Wesley, Reading, Massachusetts, 1999.
19. Lily Mummert, Maria Ebling, and M. Satyanarayanan. Exploiting weak connectivity for mobile file access. In *Proc. ACM SOSP*, pages 143–155, December 1995.
20. Jayanth Mysore and Vaduvur Bharghavan. A new multicasting-based architecture for internet host mobility. In *Proc. ACM/IEEE Mobicom* [1], pages 161–172.
21. Brian Noble, M. Satyanarayanan, Dushyanth Narayanan, James Eric Tilton, Jason Flinn, and Kevin R. Walker. Agile application-aware adaptation for mobility. In *Proc. ACM SOSP*, pages 276–287, October 1997.
22. Charles E. Perkins. IP mobility support. RFC 2002, October 1996.
23. Xun Qu, Jeffrey Xu Yu, and Richard P. Brent. A mobile TCP socket. In *Proc. IASTED Intl. Conf. on Software Eng.*, November 1997.
24. S. Savage, N. Cardwell, and T. Anderson. The Case for Informed Transport Protocols. In *Proc. HotOS VII*, pages 58–63, March 1999.
25. Alex C. Snoeren, David G. Andersen, and Hari Balakrishnan. Fine-grained failover using connection migration. In *Proc. 3rd USITS* [2], pages 221–232.
26. Alex C. Snoeren and Hari Balakrishnan. An end-to-end approach to host mobility. In *Proc. ACM/IEEE Mobicom*, pages 155–166, August 2000.
27. Amin Vahdat and Michael Dahlin. Active names: Flexible location and transport of wide-area resources. In *Proc. 2nd USITS*, October 1999.
28. Bruce Zenel and Dan Duchamp. A general purpose proxy filtering mechansism applied to the mobile environment. In *Proc. ACM/IEEE Mobicom* [1], pages 248–259.
29. Yongguang Zhang and Son Dao. A "persistent connection" model for mobile and distributed systems. In *Proc. IEEE ICCCN*, pages 300–307, September 1995.
30. Xinhua Zhao, Claude Castelluccia, and Mary Baker. Flexible network support for mobile hosts. *ACM MONET*, 6(1), 2001.