In Proc. 8th Workshop on Hot Topics in Operating Systems (HotOS-VIII)

Reconsidering Internet Mobility

Alex C. Snoeren, Hari Balakrishnan, and M. Frans Kaashoek
MIT Laboratory for Computer Science
Cambridge, MA 02139
{*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 any system that attempts to address them.

In particular, we argue that a good system architecture should (i) eliminate the dependence of higher protocol layers upon lower-layer identifiers; (ii) avoid prescribing a particular naming scheme; (iii) handle unexpected network disconnections in a graceful way, exposing occurrences to applications; and (iv) provide these services at the mobile nodes themselves. 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, 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 fundamental issues preventing the widespread deployment of mobile applications and services on the Internet. 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.

This research was supported in part by DARPA, NTT Corporation, and IBM. Alex C. Snoeren is supported by a National Defense Science and Engineering Graduate (NDSEG) Fellowship.

There are five fundamental problems caused by mobility:

1. **Host or service location:** 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 communicating peers.
3. **Disconnection:** Communicating applications should be able to rapidly discern when a disconnection at either end, or a network partition, causes communication to be disrupted.
4. **Hibernation:** If a communicating host is unavailable for a significant period of time, communications should be suspended, and resources appropriately reallocated.
5. **Reconnection:** Resumption of network connectivity between connecting peers should be detected in a timely manner. Upon re-attachment to the network, the communicating peers should resume 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. We argue that a complete—and useful—solution must address all these issues.

One need look no further than `telnet`, one of the Internet's oldest applications, for a practical example of the continuing lack of support for the latter three 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, despite using Mobile IP, if there was any activity on the session whatsoever in her absence, she will find the connection aborted. 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 [1, 3, 11, 24] and documented reports of others' work spanning several years [6, 10, 13, 16, 26], 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. These guidelines are:

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. *Don't 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 host.* Proxies are attractive due to their perceived ease of deployment. It becomes markedly more difficult to ensure they are appropriately located when hosts are mobile, however.

We elaborate upon these principles 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, leveraging domain-specific naming services (e.g., DNS, service discovery schemes [1, 9], etc.) for endpoint 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 articulate a number of design guidelines for obtaining end-to-end application mobility.

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, Vint Cerf wrote [5]: "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 also 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 it should allow for it to change, and continue to function across such changes.

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 [17, 19]. Similar results have also been found in the networking domain [6, 10, 28], and in the area of transport optimization over wireless links [3, 4, 24].

2.2 Beware the Siren song of naming

Most proposals for managing Internet mobility attempt to provide naming and location services as a fundamental part of the mobility system. For example, Mobile IP assumes that the destination of each packet needs to be *independently* located, thereby necessitating a home agent to in-

tercept and forward messages to a mobile host. Alternatively, proposals have been made to use an agile naming scheme [1] or IP multicast [18] to provide mobility support.

We believe inexorably binding mobility handling with naming unnecessarily complicates the mobility services, and restricts the ability to leverage advances in name resolution techniques. 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. The problem with using a new idealized naming scheme is twofold: 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 many applications are likely to end up using a naming scheme that best suits it (e.g. INS [1], DNS, JINI, UPNP), rather than suffer the inadequacies of a universal one. Second, if this tracking is done through the same name resolution mechanism, possibly every packet could invoke the resolution process, adding significant overhead and degrading performance.

We therefore suggest an application use whichever naming scheme is sufficiently adept at providing the appropriate name-to-location binding in a timely fashion. This service is used at the beginning of a session between peers, or in the (unlikely) event all peers were to 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 migration [24].

2.3 Handle unexpected disconnections

An area of Internet mobility that has almost entirely been ignored is disconnections. While significant work has been done in the area of disconnected file systems [13, 17], less attention has been paid to preserving application communication when a disconnection occurs, allowing it to quickly resume upon reconnection.

Our 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 [3, 7, 8, 15, 20, 26]. 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 [15], 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 [7, 25]. 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 very complex problem [26]. We observe this issue can be completely avoided if the support is collocated with the mobile host itself.

3 Migrate architecture

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 mo-

bility support needed varies dramatically across modalities [6]. By and large, applications utilize one of three well-known modalities: asynchronous, atomic, RPC-like operations; sequenced, reliable sessions; or loosely ordered, quasi-reliable data streams.

Typically the first class of applications require little in the way of mobility support from the system, and often find it quite simple to handle mobility themselves, when provided with sufficient status information by the system. The latter two paradigms are session-based, and can often benefit from more extensive system support for robust communication between application end points. Due to the disparate performance and reliability requirements, however, it is important that a mobility service does not specify a single policy, but enables the application to dictate its requirements through its choice of transport protocols.

Hence we propose an optional session layer. The layer presents a unified abstraction to the application layer: a session. 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 applications so that they can control them, if they are inclined to do so.

3.2 Session layer

The session layer exports a session abstraction to applications. Applications specify a set of connections destined for the same end point, which the session layer manages as a unit, 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. The session layer provides a locally-unique token, or Session ID, which allows the application to identify the session across any changes in network or transport identifiers.

3.2.1 Disconnection. If a host can no longer communicate with an end point due to mobility, as signaled by changes in the network layer state or transport layer failure (e.g., loss of carrier, power loss, change of address, etc.), 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 [21]. Specifically, it may block all blockable sockets, buffer non-blocking TCP sockets, selectively drop unreliable packets, 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. System policy may dictate trying multiple network interfaces [10] 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.

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.

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 [21, 24]. Similarly, the performance of even traditional transport protocols can be enhanced when the network layer exposes the appropriate state [3, 4]. Similarly, grouping multiple transport instances between the same end points into sessions can provide additional performance improvement [2, 22].

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, however, through the use of IP encapsulation. The corre-

spondent session layers establish an IP tunnel to the new end point, and continue to send application data using the old address.

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

Perhaps the greatest contribution of our architecture is its approach to disconnection: It is not an error state to be treated with hostility and disdain; instead, it is a natural transient occurrence, and 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 useability at the disconnected client, however, and has not addressed the scalability of servers. If approaches similar to ours become widespread, 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 actively considering a variety of services the session layer should implement, and briefly hypothesize about some possible techniques below.

3.3.1 State migration. Researchers have examined hoarding techniques both in the context of application data [17] and code [11] to bring state all the way to the mobile node. Dahlin *et al.* [7] proposed HTTP-based mobile extensions to allow various mobile extensions to be hosted at proxies throughout the network, or in the case of disconnection, at the client itself.

The striking similarity of the approaches taken by a variety of applications enabled for disconnected operation leads us to believe there is commonality in the services required, and, further, that the number of different services required is quite small. So small, in fact, that it may be possible to handle them in a generic fashion.

Other researchers have attempted to completely eliminate application-based state from the servers, resulting in so-called stateless servers, but many applications are difficult to implement in such a fashion. We believe the session abstraction may be a useful way to compartmentalize small amounts of connection state, reducing the amount of context applications need to store themselves. Furthermore this state could be tagged as being associated with a particular communication session, and managed in an efficient fashion. 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 great deal of context associated with a communication session, and it may be the case some (or all of it) will be invalidated by disconnection and/or reconnection. This has been handled in a variety of fashions by applications, including through the use of contextual objects, which allows one name to correspond to a variety of data objects based upon the current context [12]. This concept was extended in Active Names, which maps names to a chain of mobile programs that can customize how a service is located [25]. In both cases the application or service must be able to determine the current context, and be notified of any changes.

We note that transport and session layer information can be equally contextual. In particular, previous work has shown that context changes in the transport layer can be leveraged to adapt application protocol state [23]. 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 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 [20] and multicast-based mobility [18]. 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 [15], where connection redirection was achieved using a split-connection proxy, providing so-called Transport-Layer Mobility.

The integration of name resolution and message routing in an Intentional Naming System to implement a “late binding” option that tracks highly mobile services and nodes was proposed in [1]. A similar approach has been proposed in the TRIAD architecture [8].

TCP-specific solutions for preserving communication

across network-layer changes have been proposed in [21, 24]. However, they 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 in [27]. 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 was explored in the context of moving entire sessions in [14], where a global naming service provides endpoint bindings and address changes are affected through a TCP-specific protocol extension.

5 Conclusion

In this paper, we have defined what we believe to be the five salient issues of mobility in the Internet, and posit that one, naming, should not be addressed in a general fashion. We further presented a set of design guidelines for building a system to provide the remaining four services, distilled from a decade of research in mobile applications and system support for mobility on the Internet. Following these principles, we have outlined 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, an reconnection provided by the session layer define and appropriate set of interfaces to enable more advanced system support for mobility.

References

[1] ADJIE-WINOTO, W., SCHWARTZ, E., BALAKRISHNAN, H., AND LILLEY, J. The design and implementation of an intentional naming system. In *Proc. ACM SOSP* (1999).

[2] BALAKRISHNAN, H., RAHUL, H., AND SESHAN, S. An Integrated Congestion Management Architecture for Internet Hosts. In *Proc. ACM SIGCOMM* (Aug. 1999).

[3] BALAKRISHNAN, H., SESHAN, S., AND KATZ, R. H. Improving reliable transport and handoff performance in cellular wireless networks. *ACM Wireless Networks 1*, 4 (1995).

[4] CACERES, R., AND IFTODE, L. Improving the performance of reliable transport protocols in mobile computing environments. *IEEE JSAC 13*, 5 (June 1995).

[5] CERF, V. G., AND CAIN, E. The DoD Internet architecture model. *Computer Networks 7* (1983).

[6] CHESHIRE, S., AND BAKER, M. Internet mobility 4x4. In *Proc. ACM SIGCOMM* (Aug. 1996).

[7] DAHLIN, M., ET AL. Using mobile extensions to support disconnected services. Tech. rep., UT Austin, Apr. 2000.

[8] GRITTER, M., AND CHERITON, D. An architecture for content routing support in the internet. In *Proc. USENIX USITS* (Mar. 2001).

[9] GUTTMAN, E., PERKINS, C., VEIZADES, J., AND DAY, M. Service Location Protocol, Ver. 2. RFC 2608, 1999.

[10] INOUYE, J., BINKLEY, J., AND WALPOLE, J. Dynamic network reconfiguration support for mobile computers. In *Proc. ACM/IEEE Mobicom* (Sept. 1997).

[11] JOSEPH, A. D., TAUBER, J. A., AND KAASHOEK, M. F. Mobile computing with the rover toolkit. *IEEE Trans. on Computers 46*, 3 (Mar. 1997).

[12] KERMARREC, A.-M., COUDERC, P., AND BANATRE, M. Introducing contextual objects in an adaptive framework for wide-area global computing. In *Proc. ACM SIGOPS European Workshop* (Sept. 1998).

[13] KISTLER, J., AND SATYANARAYANAN, M. Disconnected operation in the Coda filesystem. In *Proc. ACM SOSP* (Oct. 1991).

[14] LANDFELDT, B., LARSSON, T., ISMAILOV, Y., AND SENEVIRATNE, A. SLM, a framework for session layer mobility management. In *Proc. IEEE ICCCN* (Oct. 1999).

[15] MALTZ, D., AND BHAGWAT, P. MSOCKS: An architecture for transport layer mobility. In *Proc. IEEE Infocom* (1998).

[16] MILOJICIC, D., DOUGLIS, F., AND WHEELER, R. *Mobility: Processes, Computers, and Agents*. Addison Wesley, Reading, Massachusetts, 1999.

[17] MUMMERT, L., EBLING, M., AND SATYANARAYANAN, M. Exploiting weak connectivity for mobile file access. In *Proc. ACM SOSP* (Dec. 1995).

[18] MYSORE, J., AND BHARGHAVAN, V. A new multicasting-based architecture for internet host mobility. In *Proc. ACM/IEEE Mobicom* (Sept. 1997).

[19] NOBLE, B., SATYANARAYANAN, M., ET AL. Agile application-aware adaptation for mobility. In *Proc. ACM SOSP* (Oct. 1997).

[20] PERKINS, C. E. IP mobility support. RFC 2002, Oct. 1996.

[21] QU, X., YU, J. X., AND BRENT, R. P. A mobile TCP socket. In *Proc. ICSE* (Nov. 1997).

[22] SAVAGE, S., CARDWELL, N., AND ANDERSON, T. The Case for Informed Transport Protocols. In *Proc. HotOS VII* (March 1999).

[23] SNOEREN, A. C., ANDERSEN, D. G., AND BALAKRISHNAN, H. Fine-grained failover using connection migration. In *Proc. USENIX USITS* (Mar. 2001).

[24] SNOEREN, A. C., AND BALAKRISHNAN, H. An end-to-end approach to host mobility. In *Proc. ACM/IEEE Mobicom* (Aug. 2000).

[25] VAHDAT, A., AND DAHLIN, M. Active names: Flexible location and transport of wide-area resources. In *Proc. USENIX USITS* (Oct. 1999).

[26] ZENEL, B., AND DUCHAMP, D. A general purpose proxy filtering mechansism applied to the mobile environment. In *Proc. ACM/IEEE Mobicom* (Sept. 1997).

[27] ZHANG, Y., AND DAO, S. A "persistent connection" model for mobile and distributed systems. In *Proc. IEEE ICCCN* (Sept. 1995).

[28] ZHAO, X., CASTELLUCCIA, C., AND BAKER, M. Flexible network support for mobile hosts. *ACM MONET 6*, 1 (2001).