An Extension to the DNS-based End-to-End Mobility Scheme[*]

Sameer Tilak and Nael B. Abu-Ghazaleh
Computer System Research Laboratory
Dept. of CS, Binghamton University
Binghamton, NY 13902–6000
{sameer,nael}@cs.binghamton.edu

With the proliferation of mobile devices and the emergence of the internet as a global communication and business medium, there is significant interest in providing robust, high-performance, internet connectivity to protable devices. This requires seamless delivery of data between the peers even as they change location. The accepted solution for mobility is Mobile IP, an IETF standard supporting mobility in the network layer. Recently, Snoren and Balakrishnan suggested an end-to-end implementation that uses dynamic DNS updates and transport layer connection migration to implement seamless mobility. This approach has several advantages and it is conceivable that it will replace (or co-exist) with Mobile IP to implement mobility. One of the restrictions in the proposed implementation is that it does not support the case where both peers can migrate. In this note, we propose an extension to the end-to-end implementation to allow concurrent peer migration.

I. Introduction

Advances in VLSI have made computers faster, cheaper and lighter, triggering a shift towards ubiquitous computing where access to information and information processing tools is available anywhere and anytime. Powerful portable devices (such as PDAs, lap-tops and cellular phones) are becoming increasingly common. With the emergence of the Internet as a global communication and business medium, there is a great deal of interest in providing robust, high-performance internet connectivity to portable devices. Accordingly, this has required supporting seamless mobility within the internet protocol stack.

Traditionally, the solution to the mobility problem has been implemented at the routing layer (Mobile IP [1, 2]). In this solution, there is a home agent (HA) associated with the mobile host's (MH) network. When the MH moves to a foreign network, it contacts an agent on this network (called Foreign Agent, or FA), obtains a new care of address and notifies its HA of the new address. Subsequent traffic to the mobile is intercepted by the HA and tunneled to the care-of address. Thus, mobility is seemlessly implemented by the routing layer.

The fundamental problem in supporting mobility is decoupling the host name from its location. At the routing layer, the name (IP address) carries location information (the network of the MH). This is a necessary feature at the heart of the scalability of the Internet. Accordingly, it is necessary to implement the decoupling by having the HA redirect every packet to a new address. This results in inefficient, triangular routing. Although optimizations exist to reduce triangular routing, they require transport layer modification.

Recently, an end-to-end implementation of host mobility was developed by Snoeren and Balakrishnan [4]. In this model, mobility is supported at the transport layer. More specifically, the "name" of the mobile is used as an invariant and is decoupled from its IP address (and, thus, location information). As an MH changes location, it uses the dynamic update feature of DNS to update its A record in its home DNS server. Thus, when establishing new connections with the mobile, a host will be able to discover its current location. In addition, the scheme extends TCP to allow seamless migration of active connections. The model is outlined in more detail in Section II. This model eliminates triangular routing, allows more flexibility in reacting to location changes, and provides a more natural model for supporting mobility.

One of the limitations of the proposed model is that it does not handle concurrent migration of the two ends of the connection. More specifically, if a TCP connection is established between two mobile hosts, and both of them migrate, the proposed scheme is not able to preserve the active connection. In this work, we propose an extension to the end-to-end model to handle this case. In addition, we outline the implementation tradeoffs and their effect on performance. The remainder of this note is organized as follows. Section II presents an overview of the end-to-end mobility model. Section III presents the proposed extension, and discusses the associated implementation tradeoffs. Finally, Section IV presents some concluding remarks.

[*]This work was partially supported by NSF grant EIA-991099

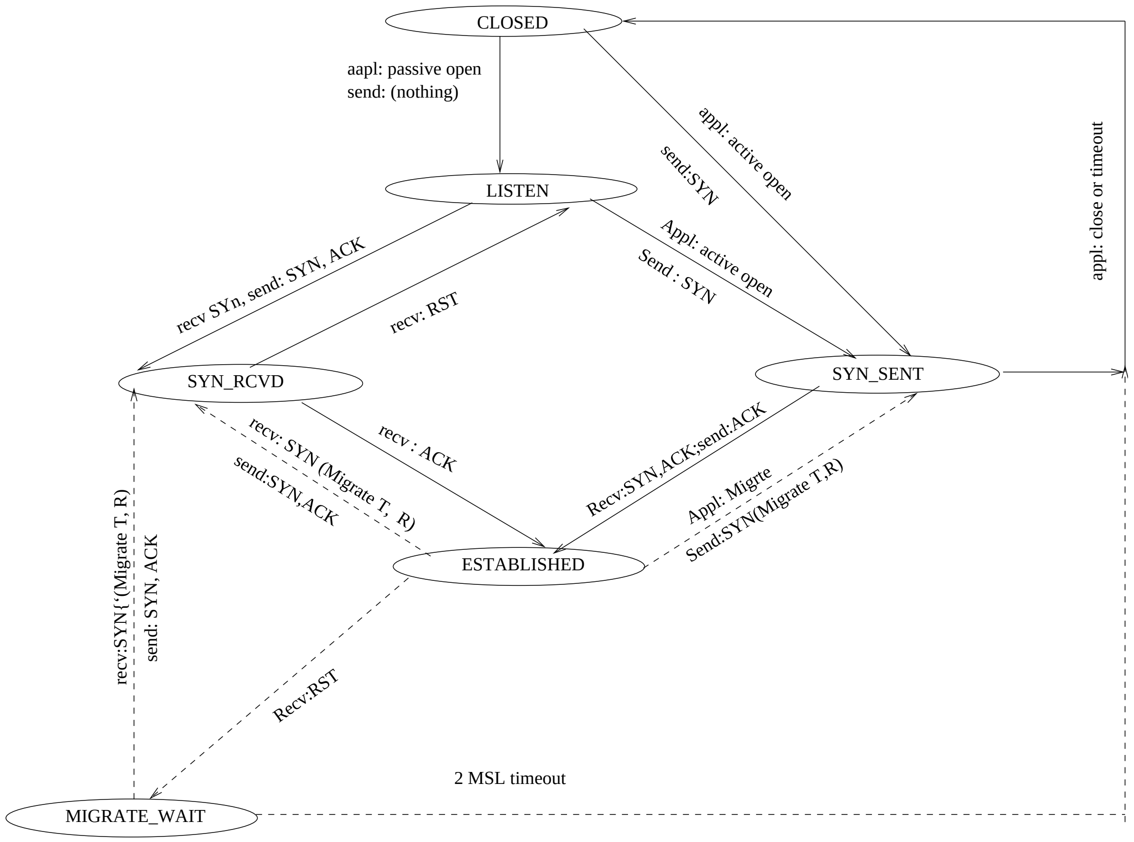

Figure 1: TCP Extension for Migration as Proposed by Snoeren and Balakrishnan

II. An Overview of the End-to-End Architecture

The End-to-End architecture solves the fundamental problem of decoupling name of the host with its location using the DNS server as invariant to implement the link between the MH's name and its location. This architecture follows the end-to-end argument in system design [3]: if the same functionality can be logically implemented in two layers in a layered system then it is advisable to implement that functionality at the higher layer. The advantage of transport layer mobility is that it enables higher layers such as TCP to learn about mobility and adapt to the changes as best fits their needs. For example the congestion control mechanism can derive significant conclusion in advance if the peer mobility is detected by it.

Addressing: Since the underlying IP substrate remains unchanged the extension does not violate the semantics of Internet addressing. When the MH migrates, it obtains a new IP address on the foreign network (e.g., using DHCP).

Mobile Host Location: When a mobile host changes its point of attachment it must update its hostname-to-address mapping with its DNS server. The entries for mobile hosts are set to have a cache TTL of 0 – they are not cached. This raises efficiency concerns if there are multiple connections to the same host. However, since the entry for the home DNS server of the mobile is cachable, the lookup is efficient. Thus, a name resolution starts with the name server of the MH's domain, which will have to latest mapping since the MH updates it whenever it migrates. While Mobile IP requires the assistance of an agent on its home network for every packet, the end-to-end scheme only requires this assistance for connection set up.

Connection Migration: With the ability to address and locate a mobile host, what remains is supporting seamless migration of connections that are active when the MH moves. Without loss of generality, the Snoeren and Balakrishnan extended TCP to allow this migration (Figure 1). A new option, called Migrate TCP, was added; it is negotiated during initial connection establishment. In order to provide protection from unauthorized migration requests, a secure token is associated with the connection. The token can be negotiated through Elliptical Curve Diffie-Hellman (ECDH) key exchange during the initial connection establishment phase. There are three facets to the architecture: (i) addressing; (ii) mobile host location; and (iii) active connection migration. Thus, the negotiated TCP connection can be identified either by a 4-tuple ¡source address, source port, dest port, dest port¿ or by a triplet ¡source address, source port, token¿. The MH can at any time resume an established TCP connection with the host using the above-mentioned triplet; the peer can then resynchronize the connection at a new end point.

There remains one case that needs to be resolved: the address allocation policy on the an MH's old network may allow reassignment of the old IP address before the MH has migrated. In that case, the new node obtaining the IP address will send an `RST` in reply to the unexpected traffic it is receiving from the peer. The `MIGRATE-WAIT` state prevents this situation from occuring; when the connection receives an RST, it stops retransmission and waits for its peer to finish migration.

The migrated connection maintains the same control block including sequence numbers space so all transmission after mobility can take place seamlessly. However this architecture limits simultaneous mobility of both the peers. We propose an extension to the scheme in order to support simultaneous mobility by both the peers.

III. Proposed Extension

The end-to-end scheme discussed above does not handle the case of connections between two mobile hosts. This is a significant draw back in a future Internet where many of the hosts are likely to be mobile. For example, ad hoc networks, an important class of wireless networks where all nodes are mobile, cannot take advantage of this scheme unless both ends are allowed to move. In this section, we propose an extension to the end-to-end mobility architecture to allow concurrent migration.

Connection Establishment

In establishing the connection, the initiating node sets the migrate option, signalling its mobility. In the acknowledgement, if the peer also sets its migrate permitted option, but also its migrate option to indicate its own mobility (only if it supports concurrent mobility). The initiator can then acknowledge with a migrate-permitted (if it supports concurrent mobility), or deny it by sending an ack with the migrate-permitted option reset. The token is negotiated as in the original scheme. Thus, a peer can restart a previously established TCP connection from the new address using ¡new-source-address, new-source-port, token¿ triplet. For the remainder of this discussion, we are assuming that both the peers support concurrent mobility.

Connection Migration

We first consider the case of a hard-handoff (i.e., by the time the MH realizes it needs to move, it has already lost its point of attachment). This is the situation considered by the original proposal. Optimizations to the scheme to handle soft-handoffs and to minimize some of the migration overhead are discussed in the next Section.

In the case of hard-handoffs, the mobile node is not able to inform about his mobility to its peer until it later reconnects to a new point of attachment. Figure 2 shows the extended TCP state diagram. We need to consider following situations:

1. One node has moved to a new network. Its notification is received just as the peer is about to move.

2. Both the peers move at the same time (each before it receives the migration notification from the other).

In the first scenario, if the MH has time to send an ACK before it moves, the situation becomes identical to two consecutive one-way migrations – no extension is necessary to handle this case. If the `ACK` was not sent before migration, then once the MH establishes a new point of attachment, it can send a `SYN/Migrate` directly to the peer's new address (which it received in the `SYN/Migrate` prior to its disconnection). The MH can move to the `SYN-SENT` state. The peer, upon receiving the `SYN/Migrate` sends a `SYN/ACK` and moves to the `SYN-RECEIVED` state (this is the reason behind the new transition from `SYN-SENT` to `SYN-RECEIVED`).

In the second scenario, each node will send a `SYN/Migrate` packet to the other from its new point of attachment (with the connection token included). Both will move to the `SYN-SENT` state. Since each is sending the message to the old point of attachment, these messages will be lost. However, the node cannot automatically conclude that the other has moved since the packet may be lost to normal network congestion or errors (especially in a wireless environment where errors are common). Thus, after sending the `SYN<Migrate, T, R>`, each waits for an `ACK` from the peer. Neither receives a `SYN/ACK`, even after one or more retransmissions.

In our extension, we add an `ADDRESS-RESOLVE` stage that is entered after the mobile determines that its peer has also moved (say, after k failed retransmissions of the `SYN/Migrate`). The purpose of this stage is to resolve the new address of the peer by contacting its DNS server similar to the case for new connections. An optimal value of k needs to be determined empirically (either statically or adaptively per connection based on RTT or previous congestion behavior). A low value of k means that we recover from concurrent mobility more quickly; however,

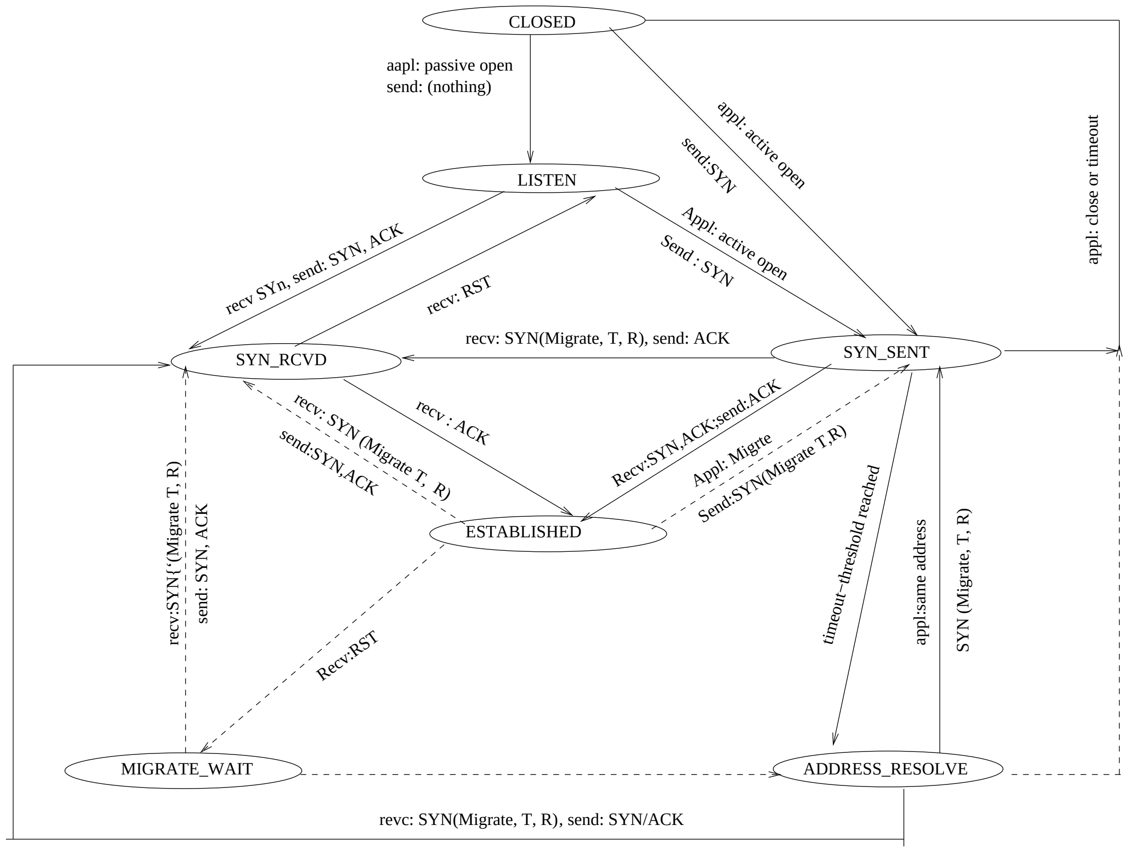

Figure 2: TCP Migration Scheme Extended for Concurrent Mobility

if the loss is due to congestion, additional traffic is generated for the superfluous address resolution. Note that if the value of k was selected too small, no harm will result: the DNS entry will return the old address, causing additional retransmissions of the `SYN/Migrate` to the peer. This takes care of network prolonged losses due to congestion, errors or temporary diasconnection.

The address resolution state contacts the DNS server and resolves the name again: the reply will include the new address of the peer, or the old address if the peer has not updated the DNS server yet. The implementation of address resolution is discussed later. After the address is resolved, TCP resends its `Migrate/SYN` request to the new IP address of the peer with the same `<new-source-address, port, token>` to the `<new-destination-addr, port>` and moves to `SYN-SENT` state. The sequence numbers are maintained form the old connection space just as in case of the original architecture. If after sufficient tries a new IP address cannot be obtained from the DNS server of the peer then the connection can be terminated. The alternation between `SYN-SENT` stage and `ADDRESS-RESOLVE` stage can be limited to a fixed number such that the connection is terminated if address resolution fails for a number of consecutive times (most likely indicating that the peer is unreachable).

Note that the address resolution is occuring at both peers; it is a race condition where either of them may follow the logical progression in the extended TCP state diagram at its own pace. Both peers may resolve the new point of attachment at the same time and then generate a `SYN/Migrate` to each other. When a `SYN/Migrate` is received, the MH can conclude that its peer discovered its new location; it can now send an `ACK` and move to the `SYN-RECEIVED` state. When it receives an `ACK` the migration is complete and the MH can move to the established case. Note that the case where one MH receives the `SYN/Migrate` from the other before it resolves the address, it can move to `SYN-RECEIVED` and reply with an `ACK` as well. It is only necessary for one MH to send the `SYN/Migrate`; this allows a performance optimization discussed later to cut down on redundant concurrent address resolution. With this extension, seamless migration of connections in presence of concurrent migration is possible. The extension enjoys the same level of security as the original scheme, and adds minimal overhead to normal TCP operation and one-way TCP migration.

Issues, Optimizations and Tradeoffs

Implementing Address Resolution: There are two approaches to implement address resolution: (i) make everything transparent to the application and push all the functionality into TCP; or (ii) Allow a minor modification in the application to accommodate address resolution. The first case requires TCP to be aware of the peer MH name; TCP will contact the DNS server and resolve the address to the new location. In the second case TCP makes an upcall to the application to resolve the new IP address. The upcall is handled by a library routine. This routine maintains a reverse translation table of IP to name for the active connections. It registers these entries for active connections (e.g., by having `gethostbyname()` update its connection cache). The upcall routine will contact the DNS server of the peer node and then get its new IP address (since the peer will update its DNS server after migrating using secure DNS update protocol). The proposed upcall mechanism does not require lot of changes to the applications; the application is linked with the modified library routine. It provides a cleaner separation between the layers since TCP remains unaware of the application level name of the host.

Soft Handoff: If soft handoff is supported, the MH determines it needs to move before it loses the current point of attachment. In this situation the mobile node can send an RST to the other host just before it leaves the old network in order to prevent further transmission from the peer to the mobile node's previous network.

Deployment issues: The scheme requires a minor modification to the existing source code with the introduction of upcall at TCP layer and handling of the call in the library routine. It retains all the advantage of the original ”End-to-End architecture”, without introducing additional deployment complications.

IV. Concluding Remarks

This is an attempt to provide an extension ”An End-to-End architecture for host mobility” to provide the feature of simultaneous mobility of both the peers. Keeping track of simultaneous mobility of hosts may be very useful in adhoc networks. It uses DNS server as an invariant to solve the problem of decoupling a host name from its current location. Security is not major concern of the extension but it uses ECC for token computation. The proposed architecture extends an End-to-End architecture of host mobility retaining all its advantages and allows simultaneous mobility by both the peers with a negligible overhead. The proposed extension does not include much overhead and we hope that it can be deployed very easily.

References

[1] PERKINS, C. RFC 2002: IP Mobility Support, October 1996.

[2] PERKINS, C. RFC 2003: IP Encapsulation within IP, October 1996.

[3] SALTZER, J., REED, D., AND CLARK, D. End-to-end arguments in system design. *ACM Transactions on Computer Systems 2*, 4 (Nov. 1984).

[4] SNOEREN, A., AND BALAKRISHNAN, H. An end-to-end approach to host mobility. In *Proceedings of the Sixth Annual International Conference on Mobile Computing and Networking (MobiCom'00)* (2000), pp. 155–166.