An Adaptive Multipath Protocol for Efficient IP Handoff in Mobile Wireless Networks*

Ken C.K. Tsang, Roy S.C. Ho, Mark C.M. Tsang, Cho-Li Wang, and Francis C.M. Lau
Department of Computer Science
The University of Hong Kong, Hong Kong
{cktsang, scho, cmtsang, clwang, fcmlau}@cs.hku.hk

Abstract

Achieving IP handoff with a short latency and minimal packet loss is essential for mobile devices that roam across IP subnets. Many existing solutions require changes to be made to the network or transport layer, and they tend to suffer from long handoff latency in either soft or hard handoff scenario, or both; and some are difficult to deploy in practice. We propose a new protocol, called the adaptive multipath protocol, to achieve efficient IP handoff. Based on link-layer signal strength measurements, two different schemes are used to handle soft and hard handoff respectively. Seamless IP handoff is achieved by using multiple transport layer connections on top of persistent link-layer connectivity during soft handoff. To achieve low handoff latency during hard handoff, a set of distributed sessions repositories (SRs), which are independent of the end hosts, are employed. Simulation results clearly support our claims. In particular, the latency for hard handoff is found to be as low as 50% of that of Fast handoff.

1. Introduction

Mobile handheld devices today need a continuous, uninterrupted Internet service for its communication. They would experience a *handoff* when their connectivity changes from one wireless access point to another. In the Internet environment, every node is identified by an IP address which is used for packet routing. In order to communicate with other devices through the Internet, a mobile device joins the Internet as a node. When a handoff occurs across access points of different IP subnets, the mobile device (node) would have to obtain a new IP address. When this happens, all the existing transport layer connections to the mobile node need to terminate and then be re-connected. This IP handoff process presents a hurdle to mobile applications needing a continuous, uninterrupted Internet service. An IP handoff can be soft or hard [6]—*hard* if the mobile node can only be assigned to one particular access point when it switches connectivity; *soft* otherwise.

In general, solutions tackling the IP handoff problem should satisfy two basic requirements. First, the new location, i.e., the IP address, of the mobile node should be communicated to the correspondent node (or host). This requirement is easily met in soft handoff with a static correspondent host. The mobile node may directly notify its peer the new location using the existing connection. However, if such direct notification is not possible (e.g., hard handoff, simultaneous node movements, or unidirectional data traffic), certain "middle agents" become necessary in order for the correspondent node to reach the mobile node through *indirection*. Second, the handoff latency (i.e., the time during which data delivery is interrupted due to handoff) and packet loss during handoff should be minimized. Data of an application instance should reach the mobile node with minimal loss as soon as the mobile node's connectivity to a new access point is established. To meet this requirement, the mobility solution should consider selecting the "best middle agent" that minimizes the indirection effect.

Some mobility solutions are implemented in the network layer. They introduce a level of indirection in the IP layer, which is supported by "network agents" located within the routing infrastructure [9]. Location notification to the correspondent host by the mobile node lessens the adverse effect of packet indirection [14]. In some approaches handoff latency is reduced by either selecting a "network agent" close to the mobile node [13], or exploiting the capability of link layer soft handoff [7]. These approaches, however, rely on existing routing infrastructures, which introduces performance problems such as triangle routing during packet transmission [9], or timing ambiguity problem during handoff [8]. In addition, network layer approaches try to mask mobility from the transport layer, which could lead to a long handoff latency due to false packet retransmissions initiated by the TCP layer on top; these retransmissions are a re-

*This research was supported by an HKU CRCG grant (No.10205867).

sult of out-of-order packet arrivals (due to indirection) in the mobile node. The exponentially increasing TCP's retransmission timeout (RTO) during hard handoff further adds to this latency [3]. Indeed, network layer approaches are generally difficult to deploy because a consensus on deployment needs to be reached within the users community.

Other solutions are implemented in higher layers, which take the end-to-end approach and try to maintain a continuous session with the end systems. Techniques like connection re-establishment [11] and connection migration [12] are adopted for hard handoff. Yet, these approaches lack support to make use of link-layer soft handoff, since the connection re-establishment or migration are always done *after* soft handoff in the link layer completes. Some approaches targetting for soft handoff scenario could still suffer from long hard handoff latency [5] since they do not address the issue of "middle agent" selection. None of the higher layer approaches, therefore, is completely suitable for both soft and hard handoff scenario.

We adopt the end-to-end design principle [10] and propose an *adaptive multipath protocol* on top of the transport layer to tackle the IP handoff problem. Based on signal strength measurements from the link layer, the protocol dynamically uses two different schemes to handle soft and hard handoff respectively. In soft handoff, we exploit the overlapping of link layer networks and use multiple transport layer connections, i.e., *multipath*, to reduce packet loss. A set of distributed *sessions repositories* (SRs) are incorporated for hard handoff or handoff after simultaneous node movements, to reduce the handoff latency. As a system residing above the transport layer, our approach is free from any performance or deployment problems common in network layer solutions.

The rest of the paper is organized as follows. Section 2 gives an overview of the system's architecture. Several specific designs of the system are discussed in Section 3. Performance evaluation is presented in Section 4. Section 5 reviews some related work, and Section 6 concludes the paper.

2. Architecture overview

Figure 1 shows an architectural overview of the proposed system. There are four key components, namely, *sessions repositories* (SRs), *location tracker*, *session maintainer* and *session mapping table*. The globally distributed *SRs* serve as the middle agents for *location indirection*. One SR would be selected for two endpoints of a session, which forwards location updates from one endpoint to another using the *subscribe/publish* mechanism (see Section 3.2). The SR selected could be changed within a session's lifetime, which adds flexibility to the system (see Section 3.3). In each node, a *location tracker* tracks self and peer location

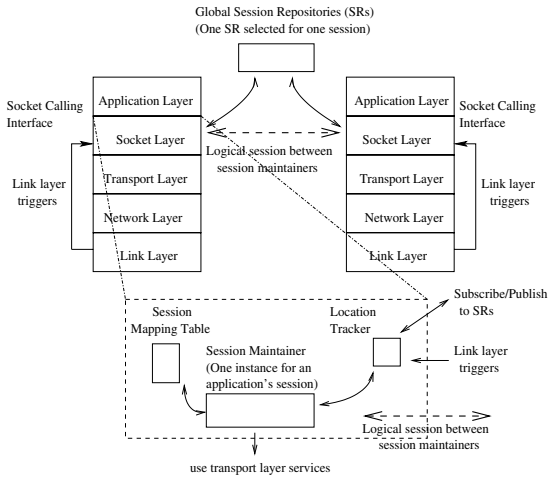


Figure 1. Architectural overview

by means of updates from the local link layer for self location, and updates (in terms of publication) from global SRs for peer location. A *session maintainer* instance is created when an application issues a transport layer connection for a new session. It manages *multiple* transport layer connections for the application. To the application, only one persistent transport layer connection exists and remains unchanged during the session's lifetime. A *session mapping table* is used by the session maintainer to identify the corresponding transport layer connections for data transmission. It stores *one-to-many* mappings between a session ID and the connection identifiers representing *all* the active transport layer connections for each session. The session ID could be derived by applying a hash function on the application-provided connection identifiers when a session starts, and is unique for each application's session.

As shown in Figure 1, the components in the endpoint reside within the socket layer. Also, the application's socket calling interfaces (e.g., socket(), bind(), etc.) remain unchanged. Thus, no or little modification is needed for the TCP/IP protocol stack and existing applications. This avoids interfering with the interaction between the underlying TCP and IP, and thus eases deployment of the solution.

Figure 2 shows the state transition diagram of the system. The system starts at the *anticipation state* when an application starts a session (i.e., sets up a transport layer connection to its peer application). In the anticipation state, the system dynamically selects an SR for the session. It also obtains signal strength measurements from the link layer and determines whether soft or hard handoff is about to take place.

When handoff occurs, the system either transits to the *direct notification handoff state* or the *indirect notification handoff state*, depending on whether peer notification is possible within a session. Peer notification is impossible if (1) the mobile node experiences a hard handoff, and thus

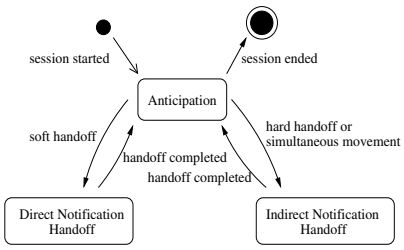


Figure 2. System's state transition diagram

cannot inform the correspondent host of its new IP address (in the new IP subnet), or (2) the two communicating endpoints move simultaneously. In the direct notification handoff state, multiple transport layer connections are used to reduce packet loss during soft handoff. In the indirect notification handoff state, location update is done via the selected SR. Transport layer connection is then re-established for the application. When handoff completes, the system returns to the anticipation state, and optionally changes the SR for the sessions. With the two different handoff states, the link-layer soft handoff capability is exploited.

3. System design

In this section, we present three specific designs of the system. We use the term *initiator* to refer to the endpoint where an application starts a session (i.e., issues the transport layer connection), and *listener* to refer to the endpoint where an application accepts the session (i.e., accepts the incoming transport layer connection).

3.1. Session-level abstraction

The session maintainer manages multiple simultaneous transport layer connections associated with different IP addresses, and uses one or more active connections to communicate with the peer. In system's handoff state after transparent connection re-establishment, user applications are masked from any network layer changes due to mobility and perceive a continuous session with its peer. Session information, including a session ID for identifying a session and the session data sequence number and acknowledgment number for data regulation, is exchanged along with the data with session maintainers in both ends by encapsulation and decapsulation.

Data from different connections of a session are regulated by keeping track of the session sequence number of expecting data. Duplication or loss of data are avoided by looking at the arriving data's session sequence number. When a new connection is set up, the *initiator* should transmit the unacknowledged data, while the *listener* should wait for data arrival at the beginning. Such ordering needs to be

```
1   Select−SR (CH−A)
2     chosen−SR <−− the SR "closest" to CH−A
3     if (total−SR < threshold) return chosen−SR
4     else
5       // re−select SRs for some existing sessions (e.g. 3), more sessions if
6       // MH starts new sessions frequently, to avoid changing SRs too often
7       CH−set <−− 3 CHs from all active CHs that are closest to one another
8       new−SR <−− null
9       min−avg−cost <−− infinity
10      for each CH−i in CH−set
11        SR−i <−− original SR of CH−i
12        avg−cost <−− average of path cost from MH via SR−i to the 3 CHs
13        if (avg−cost < min−avg−cost)
14          min−avg−cost <−− avg−cost
15          new−SR <−− SR−i
16      for each CH−i in CH−set          // assign new SR to the 3 sessions,
17        SR of CH−i <−− new−SR    // s.t. they share one SR
18      return chosen−SR
```

Figure 3. An algorithm for SR selection

defined, as a new connection with new states is used to resume the original application's session.

3.2. Location indirection

Peer location update information may arrive at a node from a peer via an SR. Such location update transmission takes place in a *subscribe/publish* manner. The initiator *subscribes* to a session record created at the SR for this session at the beginning when it is in the anticipation state. If a listener changes its IP address later in the session, the SR would be informed through the *publication* by the listener. The SR then propagates the update to the subscriber (i.e., the initiator). In the opposite direction, since an initiator can use any IP address to (re-)connect to the listener, its IP address needs not be published to the SR. As a result, an SR needs just to propagate the location information from a publisher to the corresponding subscribers according to the session records stored. On-going sessions can be kept even if the moved party cannot notify its peer the new location. With subscribe/publish, shortest latency arising from the location notification process during handoff can be achieved: the transmission of location updates takes only one path from the listener to the initiator through publication, while subscription is done in system anticipation state before handoff actually occurs.

3.3. Dynamic selection of SR

Dynamic changes of SR engaged by endpoints can be done easily through new subscriptions by the endpoints. By selecting the "best" SR in the middle, the indirection path of location updates can be shortened. In practice, the algorithm described in Figure 3 can be done in the mobile node when a new session starts. We select an SR that is "closest" to the correspondent host (line 2) since the resulting indirection path is the shortest. This can minimize the time

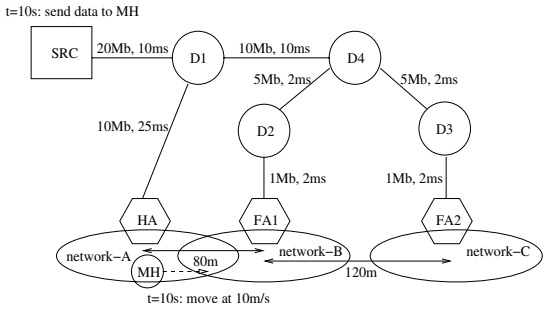


Figure 4. Network topology used in simulation experiment

for publishing the location update to the correspondent host through the SR. As more sessions are established, more SRs will be selected. This leads to a larger overhead incurred in multicasting the mobile node's publications when its IP address changes. To reduce such overhead, we reduce the total number of SRs once it reaches a threshold (line 4). We reselect the SRs for some sessions (line 7) so they share one SR (line 16-17). The shared SR should result in the shortest indirection path on average among the sessions that require re-selection (line 12-15). As a result, the length of the indirection paths for all sessions are kept minimal.

4. Performance evaluation

We evaluate the performance of our system through simulation. We compare our system with Mobile IP, IPv6, and Fast handoff.

4.1. Simulation model

The ns-2 network simulator [2] (ns-allinone-2.27 package) is used in the simulation. We patched the software with the MobiWan IPv6 extension [1] (distrib-mobiwan) and the Fast handoff protocol based on [8]. The changes include modification to the ns-2 mobile and base station node, and the inclusion of messages in the Fast handoff protocol. For fairer comparison using similar implementation, the MobiWan IPv6 extension with route optimization disabled is used to simulate Mobile IP's performance, instead of the original Mobile IP module in ns-2.

In the simulation, a network topology as shown in Figure 4 is constructed. SRC is the correspondent host for MH. HA, FA1 and FA2 are the three base stations for three different wireless networks network-A, network-B and network-C respectively, each covering an area with radius of 50m. HA and FA1 are 80m apart, while FA1 and FA2 are 120m apart. Thus, network-A and network-B overlap with each other, while network-B and network-C do not. For all network layer approaches, HA is the home agent of MH. All base stations can forward packets if Fast handoff is used.

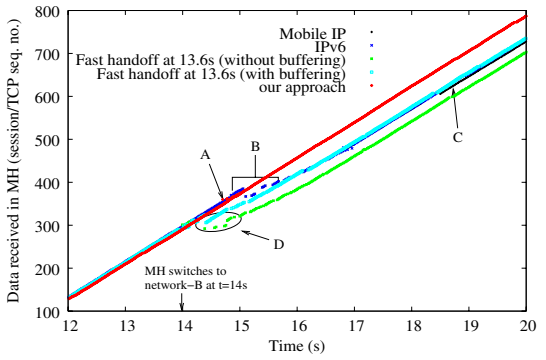
In the simulation, data is sent using TCP from SRC, starting from t=10s, and received by MH, which is initially located at the centre of network-A. At t=10s, MH moves from network-A towards network-C at a speed of 10 m/s. Soft handoff occurs when MH enters network-B, while hard handoff occurs when MH moves from network-B to network-C. In the following description, TCP-X refers to the TCP segment with sequence number X.

4.2. Performance results

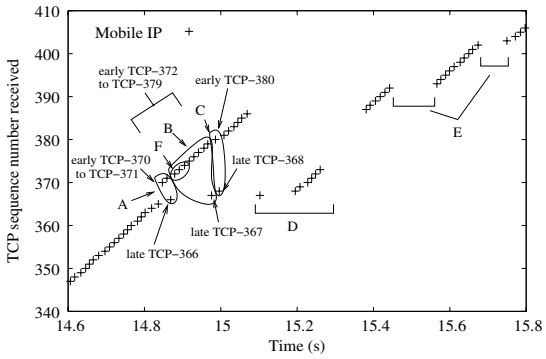
Figure 5a shows the progression of data received at MH during soft handoff (MH moves from network-A to network-B) using different schemes. Soft handoff occurs at t=14s, when signal from FA1 becomes stronger than that from HA. As shown in the figure, our approach provides a smooth data transfer during soft handoff. A new transport layer connection using the path via FA1 is established at t=14.67s (point A), in the presence of the existing connection via HA. Contrarily, data arrival at MH is not smooth in both Mobile IP and IPv6 (point B), and in Fast handoff (point D), primarily due to packet indirection done by HA which leads to out-of-order packet arrival at MH. During handoff, IPv6 and Mobile IP perform similarly (point B), as the transmission of binding updates from MH to SRC only avoids triangle routing in data transmission (point C), but does not help in reducing the handoff latency.

For Mobile IP, MH sends the binding update at t=14.67s, which is received by HA at t=14.75s. HA then starts to propagate packets (starting from TCP-370) to MH's care-of address which is under the subnet of network-B. Three observations can be obtained from our simulation, as shown in Figure 5b. First, out-of-order arrival of segments at MH occur after TCP-365 (group A, B and C), while TCP-369 is missing before 15s. Second, TCP-367 to TCP-373 are received in MH after 15s (group D). Third, some gaps exist in data transmission at about t=15.5s and t=15.7s (point E).

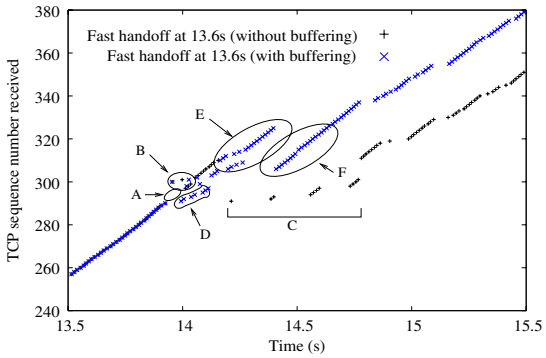
As MH moves closer to FA1, wireless data transfer to MH via HA takes more time than that via FA1. If the *wired route* from HA to FA1 is *shorter* and allows faster data transmission (we believe this is common in the real world), the indirected segments would arrive earlier at MH as compared with segments routed through HA without indirection. In particular, the transmission of segment TCP-369 is so slow that it cannot reach MH before MH leaves network-A. This explains our first observation above. As a result of the out-of-order TCP segments, the duplicate acknowledgments sent by MH trigger SRC to perform retransmissions (7=1+2+4 in total). It also drives TCP to enter the slow start phase and limits the data transmission rate. These explain the second and the third observations above.



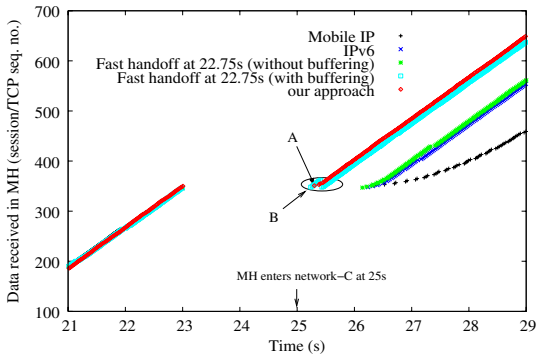
(a) Soft handoff performance comparison between Mobile IP, IPv6, Fast handoff and our approach



(b) Soft handoff behaviour of Mobile IP (so as IPv6) in detail



(c) Soft handoff behaviour of Fast handoff (with and without buffering) in detail



(d) Hard handoff performance comparison between Mobile IP, IPv6, Fast handoff and our approach

Figure 5. Simulation results

We minimize the timing ambiguity problem [13] of Fast handoff in our simulation: Fast handoff starts (i.e., MH sends out RtSolPr message to HA) at t=13.6s, which is an optimal time as it allows MH to switch network at t=14s, the time when MH is at the mid-point between HA and FA1. When MH receives FBack (the last message in Fast handoff to be received by MH before it switches network) from HA in network-A at t=14s, it immediately switches to network-B and sends out FNA to FA1. Thus, the time during which MH is disconnected to both network is made negligible.

As shown in Figure 5c, out-of-order segments arrive (group B) due to a reason similar to that for the Mobile IP case. In addition, TCP-291 to TCP-296 are lost (group A), as they are forwarded by HA to FA1 too early. While FA1 does not buffer these TCP segments but directly delivers them to MH; at the time the delivery is done, MH has not received FBack from HA yet and stays at network-A. These lost TCP segments trigger the retransmission of TCP-291 (which is received by MH at t=14.21s) and the ensuing segments (group C), and results in a long handoff latency.

The handoff performance of Fast handoff with buffering in FA1 is better than that without buffering, as shown in Figure 5c. It can be observed that TCP-291 to TCP-296 (group D) are received at MH after MH enters network-B at t=14s. However, MH receives segments coming from two different paths during handoff: one an end-to-end path from SRC (group E) and another indirect one HA to FA1 (group F). Such out-of-order segments trigger retransmissions incorrectly, causing slower data transmission rate subsequently.

Figure 5d shows the progression of data received at MH during hard handoff (MH moves from network-B to network-C) using different schemes. There is no signal received by MH from t=23s to t=25s because it is outside both network-B and network-C. When MH moves into network-C after t=25s, transmission of data can resume. In the simulation, Fast handoff starts at t=22.75s, which is optimal as MH manages to receive FBack from FA1 (at t=22.997s) *just before* it leaves network-B (at t=23s).

As can be seen from Figure 5d, our approach is the earliest to resume data delivery after hard handoff. Reconnection via a new network path starts at t=25.3s, after MH detects the presence of FA2 (at t=25.2s) and 0.5 MH-SRC RTT ($\sim$100ms) for the location update publish/subscribe process via an SR (point A). However, in Mobile IP, IPv6 and Fast handoff without buffering, MH receives the first retransmitted segment after t=26s (retransmitted by SRC at t=26.02s). This is due to the exponential backoff effect in TCP's RTO during data retransmission in SRC, as suggested in [3]. SRC's retransmission of segment at t=23.02s and t=24.02s cannot reach MH.

In Fast handoff with buffering, although the buffered data stored in FA2 can be delivered immediately to MH after MH detects the presence of FA2 (at t=25.2s) and sends

out FNA, SRC is not aware of the handoff in MH. As a result, it retransmits the unacknowledged segments as usual, causing MH to receive a duplicate set (group B) of those buffered segments (from TCP-347 to TCP-363). Data transmission returns to normal at t=25.6s, i.e., 300ms later than our approach. In other words, our approach achieves a hard handoff latency which is 50% shorter than Fast handoff.

5. Related work

Most network layer approaches are based on the Mobile IP [9] or IPv6 standards [14]. In Mobile IP, packets are forwarded to the mobile node by a home agent. Such indirection introduces the triangle routing effect which can be reduced by using a binding cache in the correspondent host in IPv6. Various extensions to IPv6 have been proposed to minimize the long handoff latency suffered in IPv6. Hierarchical Mobile IPv6 [13] uses a Mobile Anchor Point to handle intra-domain host mobility. Fast handoff [8] supports pre-registration of IP addresses prior to handoff which reduces the network registration time. S-MIP [7] solves the timing ambiguity problem of Fast handoff with the concept of "simultaneous bindings" [4]. Nevertheless, these approaches still give a long handoff latency especially in hard handoff situations due to TCP's false retransmissions and long retransmission timeout.

Other approaches work in the higher layers, following the end-to-end design principle [10]. Migrate [12] is a session-based mobility solution with modifications in the transport layer. Connection migration is done to preserve applications' sessions. Yet it does not tackle the long handoff latency problem during handoff, or support simultaneous movements of end hosts. Guo et al. proposed a mobility management system [5] between TCP and IP to handle mobility. One of the components in the system, called *virtual connectivity*, is responsible for connection maintenance and handoff latency reduction. However, it suffers from the same performance problem as network layer approaches with the same reasons, since the system resides below the transport layer.

6. Concluding remarks

In this paper, we propose an end-to-end protocol to tackle the IP handoff problem. The proposed protocol uses different schemes to handle soft handoff and hard handoff based on the wireless network coverage. Seamless soft handoff is achieved using multiple transport layer connections. For hard handoff, the effect of indirection is minimized using a dynamically selected SR as the middle agent. The proposed system resides on top of the transport layer and does not require any modification to the TCP/IP protocol stack or the network core. Simulation results show that seamless handoff can indeed be achieved in soft handoff scenarios; and our system's handoff latency can be as low as 50% of that of Fast handoff for hard handoff.

There remain a number of important issues before the benefits of the proposed system can be fully realized. First, data regulation done by the session maintainer can be enhanced to utilize a higher data rate perceived by end applications during soft handoff, when simultaneous connections are present. Moreover, we can achieve smoother data delivery for say a multimedia application during hard handoff using certain caching technique, taking into account both the location of the mobile node and the required quality of service of the multimedia application.

References

[1] MobiWan: NS-2 extensions to study mobility in Wide-Area IPv6 Networks. http://www.inrialpes.fr/planete/pub/mobiwan.

[2] The Network Simulator - ns2. http://www.isi.edu/nsnam.

[3] R. Caceres and L. Iftode. Improving the Performance of Reliable Transport Protocols in Mobile Computing Environments. *IEEE Journal on Selected Areas in Communications*, 13(5):850–857, June 1995.

[4] K. El-Malki and H. Soliman. Simultaneous Bindings for Mobile IPv6 Fast Handovers. IETF Internet Draft, October 2003. Work in Progress.

[5] C. Guo, Z. Guo, Q. Zhang, and W. Zhu. A Seamless and Proactive End-to-end Mobility Solution for Roaming Across Heterogeneous Wireless Networks. *IEEE Journal on Selected Areas in Communications*, 22(5):834–848, June 2004.

[6] B. Hamdaoui and P. Ramanathan. A Network-Layer Soft Handoff Approach for Mobile Wireless IP-Based Systems. *IEEE Journal on Selected Areas in Communications*, 22(4):630–642, May 2004.

[7] R. Hsieh, Z. G. Zhou, and A. Seneviratne. S-MIP: A Seamless Handoff Architecture for Mobile IP. In *Proc. of IEEE Infocom 2003*, pages 1774–1884, March 2003.

[8] R. Koodli. Fast Handovers for Mobile IPv6. IETF Internet Draft, October 2003. Work in Progress.

[9] C. Perkins. IP Mobility Support. Request For Comments 2002, October 1996.

[10] J. H. Saltzer, D. P. Reed, and D. D. Clark. End-to-end Arguments in System Design. *ACM Transactions on Computer Systems*, 2(4):277–288, November 1984.

[11] H. Schulzrinne and E. Wedlund. Application-Layer Mobility Using SIP. *ACM SIGMOBILE Mobile Computing and Communications Review*, 4(3):47–57, July 2000.

[12] A. C. Snoeren and H. Balakrishnan. An End-to-End Approach to Host Mobility. In *Proc. of the 6th Annual International Conference on Mobile Computing and Networking*, pages 155–166, Boston, MA, USA, August 2000.

[13] H. Soliman, C. Castelluccia, K. El-Malki, and L. Bellier. Hierarchical Mobile IPv6 mobility management. IETF Internet Draft, June 2003. Work in Progress.

[14] W. Stallings. IPv6: The New Internet Protocol. *IEEE Communications Magazine*, 34(7):96–108, July 1996.