TCP-R: TCP Mobility Support for Continuous Operation

Daichi Funato[†], Kinuko Yasuda[†] and Hideyuki Tokuda[†‡]
[†]Graduate School of Media and Governance, [‡]Faculty of Environmental Information,
Keio University. Endo 5322, Fujisawa, Kanagawa, 252, JAPAN
{daichi, kinuko, hxt}@sfc.keio.ac.jp

Abstract

The TCP-R (TCP Redirection) is an extension of TCP, which maintains active TCP connections when the disconnection occurs due to the change of the IP address or the change of the network device.

In mobile computing environments, the IP address may be changed frequently as the host moves across subnets or changes a network device. However, it is difficult for most network applications to work continuously in such a situation. There is much research to support such dynamics in the network layer, but these attempts in the IP layer tend to incur some complexity.

This paper is intended as an investigation of the end-to-end mobility support in transport layer. We developed the simple and secure redirection mechanism in TCP, which enables us to keep our working activities without any intermediate agents. We also show that TCP-R can serve as a solution to the retransmission timeout problem which frequently occurs in mobile environments. We implemented and evaluated a prototype of TCP-R by modifying FreeBSD. The measured performance indicates that TCP-R can maintain operation continuously with minimal overhead and complexity.

1 Introduction

Recently, various types of portable computers have become very popular. Those machines often have enough computing power to run the same network applications as we use on workstations. Using them to connect the Internet is now commonplace. They are inherently designed to be carried around, so they may frequently change IP addresses when the machines are moved between cells. Changes of network devices can also occur on them, because plug-and-play technologies, such as PCMCIA and PCI, allow dynamic changes of device resources. In such situations, however, most network applications cannot work continuously, and we usually must deal with some troubles at both disconnection and re-connection time. To avoid such inconvenience, much research has been proposed in the IP layer, which achieves transparent deliveries of IP packets to mobile hosts[1, 2]. These approaches at the IP layer enable all of the upper-layer systems to work continuously without knowing about disconnection, though they tend to reduce their performance or increase protocol complexity.

In this paper, we deal with end-to-end mobility support which does not need any intermediate agents. We focus on the transport layer, especially with TCP as it is the most widely used and reliable transport protocol. Although IP layer efforts can support host migration transparency, current TCP implementation does not provide optimal end-to-end support in mobile environments. The concept of "Continuous Operation" is the key of our research, which means keeping continuous working activities in frequently changing network environments. While the IP layer approaches aim to keep the IP address of the mobile host, we intend to maintain active TCP connections whether the IP address has changed or not. We also observe that most applications do not need support for full host migration transparency after they have established their connections, which means that the active connections usually require only support for continuous operations. We also describe cooperation with other host migration techniques such as MobileIP and Dynamic DNS update[3] for establishing new connections.

The rest of this paper is organized as follows. In Section 2, we introduce some previous work related to TCP-R, and consider the role of the layers in mobile computing. Section 3 provides the design concepts of TCP-R, continuous operation and compensative operation. Section 4 describes the protocol overview and mechanism of TCP-R. In Section 5, we show the prototype implementation of TCP-R and the measured performance, and Section 6 concludes the paper.

0-8186-8061-X/97 $10.00

2 Related Work

Many approaches have been proposed for the mobile computing environments. In this section, we review these works classified into three layers and discuss their suitability for mobile computing.

Network Layer

MobileIP[1] is a networking and routing mechanism using the IP-IP tunneling protocol. In MobileIP, two types of routers are needed; Foreign Agent(FA) and Home Agent(HA). These agents work together to forward packets to mobile hosts via triangle routing, and allows mobile hosts to move from one IP subnet to another without changing their IP addresses. Unfortunately, the triangle routing tend to incur extra overhead and decrease network throughput.

There are some approaches to avoid triangle routing by modifying IP routing tables. These approaches are called "Route Optimization"[4] or "Short Cut Routing"[5]. However, they require software changes of all entities and need complex authentication mechanism for security reasons, so that it is excluded in the current MobileIP standard.

In [2], the virtual internet protocol and the propagating cache method are proposed, which also support host migration transparency but reduce the cost of the triangle routing. Although this mechanism shows one view of future Internet networking, currently it is not widely accepted because it require replacement of gateways.

Another way to support host mobility in the network layer is to cheat the source host IP address by using a so-called source routing technology. However, some routers denied supporting these methods because of security reasons. Almost all of the MobileIPs have inconsistent IP address fields so that they cannot pass such gateways.

In the IP version 6, route optimization is being proposed as a standard[6]. IPv6 also presents several proposals for mobility use, but none of them solve increasing complexity and end-to-end security problems.

Dynamic DNS update[3] does not reside in the network layer, but it has suitable functionalities related to IP address. It dynamically manages the bindings of IP address and host name which are independent from IP addressing. It means wherever the host moves, the DNS name is identical to the host.

Transport Layer

In the context of transport layer, there are also attempts to support host mobility. Generally, the idea to support mobility in the TCP layer is unlikely to receive a warm welcome, because in keeping with the layering philosophy, host-to-host connectivity is considered to be supported in the IP layer. We show in this paper that transport layer has certain possibilities to support end-to-end host mobility when there are no intermediate agents.

I-TCP[7] is one solution to the problems posed in MobileIP. The main advantage of I-TCP is that it is able to differentiate between the motion-related packet losses and the congestion related ones.

While TCP-R straightforwardly revises the connection between the mobile host(MH) and the correspondent host(CH) and connects them directly; in I-TCP technology, it is broken into two phases, one between the MH and the FA, the other between the FA and the CH. These methods are called "Split-Connection Schemes". Splitting connections allow the FA to absorb the diversity of network characteristics and achieve cost-effective data transport. But [8] shows the split-connection approach achieves less throughput than that of a well-tuned end-to-end connection approach.

T/TCP[9] is not the mechanism intended to host mobility support, but it is similar to TCP-R in terms of the extension of TCP options. In the TCP client-server model, we sometimes have to face the extra overhead on the client's end; the three-way handshake and dead connections lingered in the TIME_WAIT state of TCP. T/TCP solves these problems by bypassing the three-way handshake and shortening the TIME_WAIT state using the TCP options.

Upper Layers and OS Support

Teleporting[10] takes another approach to support mobile applications. It operates within the X Window System, and allows user to interact with their existing X applications at any X display. In Teleporting, the redirection scheme is used to make a mobile X session, which is not fixed to a particular X display, but can be materialized on demand at any suitable display.

These studies indicate that the upper layers sometime want to support host mobility themselves with minimal overhead in their specific ways, though it is still strongly required by many existing network applications to support continuous operations in lower layers.

Plug-and-play technologies, such as PAO[11] and wildboar[12], are the efforts to save us the trouble of re-configuring the network devices at re-connection. These mechanisms enable to change multiple PCMCIA cards dynamically and configure each card automatically, so that we can eliminate some procedures related to network devices, such as re-assignment of

IP addresses and updating of routing tables.

3 Design and Concept of TCP-R

3.1 Continuous Operation and Compensative Operation

"Continuous Operation" is the key concept of our model, in which we keep continuous activities on mobile hosts. Usually, when mobile hosts change their IP addresses or their network interfaces, we must handle some re-connection procedures.

We separate re-connection handling into two types. One is the support of existing connections and the other is support of new connections at the original MH's location. We call the former "Continuous Operation" and the latter "Compensative Operation".

While the compensative operation usually needs network layer supports, the continuous operation can be placed effectively into the transport layer. IP layer approaches tend to support both operations in the network layer, but it does not provide an optimal solution for continuous operation.

The support of continuous operation is generally needed by connection-oriented and state-full applications which have long connection life-time. In our approach, we focus on these applications to design efficient continuous operation support, though it can support any types of applications so long as they use the TCP protocol as the transport layer.

MobileIPs require intermediate agents, but we realize intermediate agents are required only for the compensative operation. It should not always be MobileIPs. Dynamic DNS update also supports compensative operation, because almost all the Internet applications which need compensative operation require DNS name and current IP address bindings.

3.2 TCP-R for Continuous Operation

TCP-R is intended to enable each active TCP connection to be maintained despite of the change of IP address. To retain TCP connections against each disconnection, TCP-R provides the redirection mechanism.

The concept of the TCP-R protocol is the following:

- Every network application must be able to keep its connection alive even if the IP address is changed during the connection.

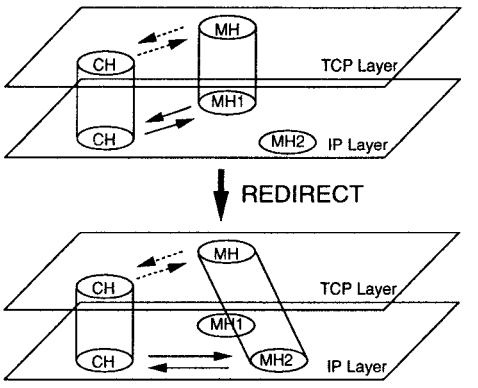

Figure 1: Redirection on Active TCP Connections

- As Mobile hosts are likely to be battery powered, any unreasonable overhead should be eliminated, and the response time should be minimized.

- IP address must always be sane so that packets can pass secure gateways.

- Connections must be secure against any forged redirection messages.

In TCP, each connection is uniquely specified by a pair of addresses identifying its two ends. Our redirection mechanism is basically straightforward; if an IP address is changed, simply revise the pair of addresses in the existing TCP connections.

Figure 1 shows the conceptual model of redirect operation provided by TCP-R. When the mobile host(MH) detects the change of IP address, the redirection message is issued by the MH in order to inform its correspondent host(CH) of its new IP address. As it is received by the CH, the pair of addresses in the existing TCP connections is revised by using it, and they resume communication each other through the revised TCP connections. In addition, the redirection operation must be done in a secure way.

We establish an assumption to provide the redirection mechanism; TCP-R assumes that each mobile host can detect the change of its IP address somehow. Some plug-and-play technologies and external daemons may be required for this assumption. TCP-R places no additional constraints on the assignment of IP addresses, the new IP address can be assigned either manually by users or automatically by external mechanisms. If users do not want to produce dead IP addresses, external mechanisms which have some address reclaiming mechanisms, such as DHCP, are recommended.

3.3 Support of Compensative Operation with Other Schemes

Each mobility support should be placed separately in its own optimal layer. The efficient support for continuous operation can be placed in the transport layer, and the active connections can keep their activities only with this support.

We provide extended functionality by cooperation among multiple protocol layers. By cooperating with IP layer supports, TCP-R does not only support continuous operation but also support new connections, compensative operaton. To support compensative operation, intermediate agents are necessary such as MobileIP and Dynamic DNS update.

If TCP-R cooperates with MobileIP, however, such new connections can provide more efficient performance during sessions. After the host moves, new sessions will have to depend on host migration support in the IP layer in order to establish their connections. They can redirect their connections using TCP-R, so that they directly communicate each other without the cost of triangle routing.

Dynamic DNS Update can also support compensative operation. Originally, DNS was designed to manage statically configured database. By dynamically assigning IP address, we can maintain the correct bindings of IP address and DNS name. It means if we retain the same DNS name, anyone can reach the host. That is the compensative operation. However, Dynamic DNS Update cannot support continuous operation. TCP-R cooperated with Dynamic DNS Update is a possible solution to the mobile computing environments.

4 Mechanism of TCP-R

In this section, we show the mechanism of TCP-R. TCP-R is a backwards-compatible extension of TCP to provide efficient support for continuous operations. TCP-R keeps all features and philosophy of the standard TCP, though it has additional merits for mobile computing. In TCP-R, we added some auxiliary operations around the standard ESTABLISHED, SYN_SENT and SYN_RECEIVED state, which treat some possible situations in the mobile computing environments. In the normal case, these additional operations do not affect standard TCP behaviors, and the other operations are not changed at all.

4.1 Initial Connection Establishment

During a three-way handshake for a new connection establishment, we add two optional procedures for TCP-R.

1. Ascertain whether the correspondent host is TCP-R aware or not.

2. Exchange the authentication keys.

We have to ascertain that the correspondent host can recognize TCP-R during the initial connection setup. First, sender issues a "SYN" piggybacked with RD_RDY option. If the correspondent host cannot understand TCP-R, RD_KEY option is not piggybacked with a "SYN, ACK". Then, it will not try to send redirection message to the host afterwards.

The redirection mechanism could be a vulnerability in the mobile computing if the redirection messages were not authenticated. Currently, we have implemented simple authentication method based on public key cryptography to keep redirection mechanism secure.

4.2 The Redirection Timeline

Figure 2 depicts a timeline of TCP-R redirection mechanism.

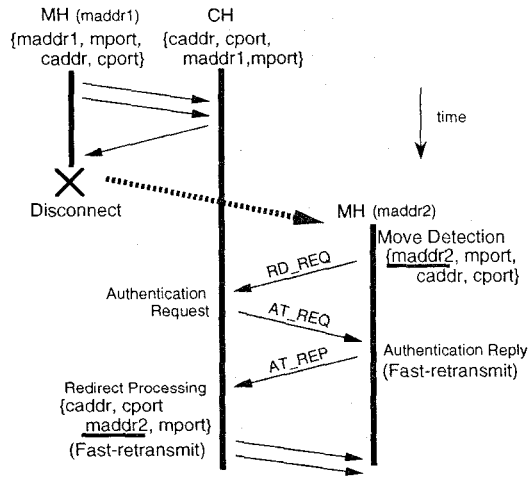

Figure 2: Timeline of TCP-R with Authentication

The steps of TCP-R redirect operation are as follows:

1. A mobile host(MH) detects that it has moved to another network or changed its network device. After the MH obtains a new IP address, the MH sends a redirect message to its correspondent host(CH).

2. When the CH receives the redirection message, it requires the connection authenticator of MH.

3. MH replies with the connection authenticator message to the CH.

4. CH checks the connection authenticator. If it is correct, CH revises the pair of addresses of the existing TCP connection, in order to redirect it to the new MH's address. Simultaneously, the MH also revises its own pair of addresses.

5. They resume to communicate with the revised TCP connection. In addition, they use the fast retransmission scheme to eliminate retransmission timeout duration.

4.3 TCP-R State Transition Diagram

Figure 3 illustrates the state transitions on the redirect operation. This figure shows only five states related to the redirect operation, and the other states of the standard TCP state machine are omitted here. Some of them are auxiliary states and need not to be implemented explicitly with the standard TCP states.

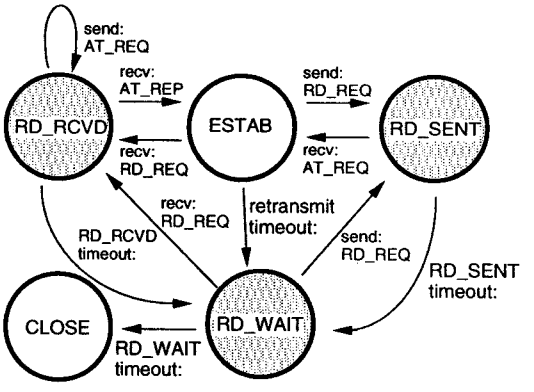

Figure 3: State Transition Diagram of the TCP-R

Each state in Figure 3 is defined as follows:

RD_SENT

After the RD_REQ is sent to the mobile host side, the mobile host enters the RD_SENT state. In this state, the mobile host revises its pair of addresses of the connection with its new IP address. When the mobile host receives the authentication segment(AT_REQ) in this state, it sends a calculated identifier to the peer. Then it returns to the ESTAB state and resumes communication to avoid the retransmission timeout delay. If the mobile host times out on the RD_SENT message before it can receive the AT_REQ, it enters the RD_WAIT state.

RD_RCVD

When the correspondent host receives the RD_REQ, it enters the RD_RCVD state. In the RD_RCVD state, there are two local modes which depend on the type of redirection. One is primary mode and the other is authentication mode. In the primary mode, it just sends an AT_REQ segment and returns to the ESTAB state, avoiding the retransmission timeout delay. In the authentication mode, the RD_REQ segment is issued and waits for the RD_REP reply containing an identifier. If the correspondent host times out before it can receive the AT_REP, it enters the RD_WAIT state.

RD_WAIT

Most of current TCP implementations apply the fixed retransmission timeout period, and the expiration of the retransmission timer causes termination of the connection. In mobile computing environments, however, both ends of the TCP connection may expire their retransmission timers while a mobile host is being taken away from the network.

In order to avoid the retransmission timeout during the host movement, we adopt the new wait state, RD_WAIT, and let hosts wait until the requested time expires. If the mobile host in this state detects that it has been re-attached to the network, it enters the RD_SENT state. On the other hand, if the correspondent host receives the RD_REQ, it enters the RD_RCVD state.

The RD_WAIT timeout should be configurable for applications and provided as a parameter at the initiation of the connections [1].

5 Implementation and Evaluation

In this section, we describe the preliminary implementation of TCP-R, and show the measured performance with our implementation. We implemented a prototype of TCP-R by modifying FreeBSD, in which we defined several new TCP options and some additional states. We also mentioned security issues and an external mobile daemon which enables network applications to operate using TCP-R netwroking environment.

[1] [13] recommended that an opening of a new connection allow a parameter specifying the total timeout period for data sent by TCP, and [14] requires that an application must be able to specify a parameter for a connection.

5.1 New TCP Options

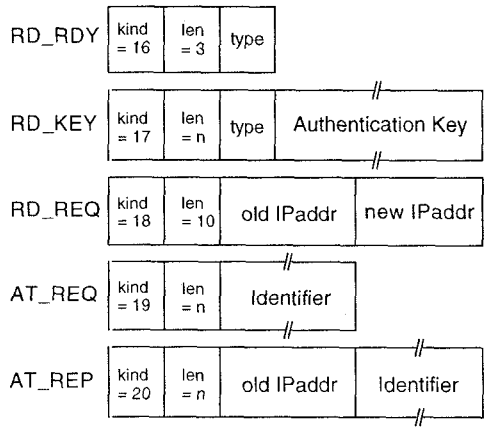

Figure 4: TCP New Options in TCP-R

Five new TCP options are used with TCP-R. Figure 4 shows these options, which will be contained in redirection segments and its echo segments. The rules for these options are as follows:

RD_RDY option
: This option may be sent in the first segment from the MH to the destination CH after MH has been moved. It contains the type of redirection which will be negotiated with CH.

RD_KEY option
: This option may be sent in CH's "SYN, ACK" segment to the MH. This option also may be sent in an "ACK" segment to the CH's "SYN, ACK" from the MH. If MH and CH support authenticated TCP-R, RD_KEY contains an authentication key. In our test implementation, an authentication key has a 64 bit field. If it only supports unauthenticated TCP-R, just a type will be sent.

RD_REQ option
: This option may be sent in a first segment from the MH to the destination CH after MH has been moved. It contains the old IP address and new IP address which will be the new destination of the CH. Note that the old IP address has to be sent because CH uses the old IP address as a hash key for the TCP control block selection.

AT_REQ option
: This option may be sent in a CH's reply segment to the MH's RD_REQ segment. CH needs the identification of MH. In our implementation, CH chose a 32 bit random number and it is encrypted by MH's public key.

AT_REP option
: This option may be sent in a MH's reply segment to the CH's AT_REQ segment. This option also contains the old IP address so that CH can select the correct TCP control block. In our implementation, MH decrypted and AT_REQ value and encrypted it using CH's public key.

By keeping the TCP options on a 4-byte boundary, performance will be improved in some architectures. These options are normally preceded by two 1-byte no-operations(NOPs).

5.2 Additional States in TCP FSM

As shown in section 4.3, we have additional three states in the TCP Finite State Machine(FSM). Table 1 shows new flags added to specify these states.

Table 1: New States in TCB

State Flag	Description
TCPS_RD_SENT	sent RD_REQ option to a peer wait for AT_REQ reply
TCPS_RD_RECV	receive RD_REQ option from a peer sent AT_REQ option to a peer wait for AT_REP reply
TCPS_RD_WAIT	transit from the retransmission, timeout RD_SENT timeout, and RD_RCVD timeout wait for RD_REQ message

In Net/3 release, if the retransmission fails, tcp_drop() function is called to tear down the connection. In TCP-R, such connection tear down caused by retransmission failure may be fatal, since mobile hosts may be taken away from the network for a long period. Instead of tcp_drop(), we implement tcp_rd_wait() to be called in the case of timeout, in order to sustain the connection until the redirection message arrives from local or peer. A user can specify the RD_WAIT timeout duration, so as not to retain undesirable zombie connections.

5.3 TCB Hash Management

TCP Control Block (TCB) keeps almost all of the status of TCP connections. TCP connections are identified by a 4-tuple of {srcaddr, srcport, dstaddr, dstport}, and this 4-tuple is used to find a proper TCB when a segment arrives at a host. In the Net/3 release, a hash table is employed to manage TCB, and the 4-tuple is used as its hash key in the in_pcblookuphash() function.

When a redirection segment arrives to a host which supports TCP-R, it changes dstaddr in its 4-tuple and rehashes the hash entry related to its correspondent host. To achieve the redirect operation correctly, these operations are needed by both ends of the connection. In addition, if the IP address changes in the same segment, we need to flush the routing table cache in the TCB.

5.4 Move Detection and Security Issues

Move detection is not included in our protocol specification, because it is highly dependent on the structure of operating systems and protocol implementations[16]. The move detection method is not specified either in MobileIP, though two primary mechanisms are recommended; One is based on Lifetime field of Agent Advertisement, and the other uses network Prefix-Lengths Extension. These mechanisms can also be applied for TCP-R.

To provide a certain framework for external move detection, we add a system call interface to notify the movement to the kernel. This can be called either explicitly by network applications or implicitly by an external daemon. Now we are implementing an external mobile support daemon, which inspects the current status of the mobile host to detect movements and issues the system call. It will also deal with the connection lifetime management.

We have two basic security modes in TCP-R. One is primary mode and the other is authentication mode. The former is an optimistic approach and the latter is a pessimistic approach.

In the optimistic approach, TCP-R provide the primary authentication method based on tuples of addresses and sequence numbers. As mentioned before, TCP connections are specified by two combinations of address and port, so we use the pairs for a part of authentication. Also, we use current sequence number for authentication, since hosts not along the path between the CH and MH will not see the sequence numbers. This general security strategy is prevalent in the Internet. When the disconnection occurs, the sequence number may slightly differ between sender and receiver, so there has to be a certain range of permission. In our implementation, the receiver window size is used for the range specification, and segments which have the sequence number within this rage will be permitted.

For the pessimistic approach, we have a explicit authentication mode in TCP-R. In our test implementation, we have coded a simple public key encryption method. But we did not support any certification because we just wanted to verify the identity of the active connection. The certification and message authentification code(MAC) should be placed within higher layers such as SSL[17] and PKDA[18].

5.5 Performance Evaluation

We have measured some basic performance of TCP-R in three areas: initial connection setup, reconnection latency and TCP throughput. We have used a Toshiba Portege 620CT (Pentium 100MHz, FreeBSD) as the mobile host(MH) and IBM PC/AT Compatibles (Pentium 166MHz, FreeBSD) as CH, FA and HA. 10BaseT ethernet is used as the data link layer. Each value of time is measured by using pentium counter. MobileIP evaluations are based on the CMU FreeBSD implementation[15].

Table 2: Connection Setup Costs (ms)

	Active Open	Passive Open
Normal TCP	1.54	3.82
TCP-R(primary)	1.84	3.97
TCP-R(auth)	2.32	4.69

Table 2 depicts the initial connection establishment costs in the kernel level. This measurement was carried out with a 1 hop network. According to the results, the overhead of the two optional procedures for redirect operation is relatively small.

Table 3: Reconnection Latency after 2 sec disconnect (ms)

	Latency (ms)
TCP-R(primary)	492
TCP-R(auth)	507
MobileIP	3147

Table 3 shows the application level reconnection latency. In this experiment, MH moved from a home network to a 1 hop network after a 2 sec disconnect, and mesurement began when the MH detected it's own movement. The results show that the actual overhead of redirection operations are extremely low compared with MobileIP. In addition, we can reduce the retransmission timeout delay, which normally would take few seconds however it can be reduced. Although we have not implemented the features of route optimization, the expected value of MobileIP with route optimization will be the same as that of MobileIP. Other measurements show the time for the primary redirection operation is 170μsec on average on the mobile host side, which occupies approximately 27% of the total time needed for redirection in the case of a 0 hop network. The remaining time is spent waiting for the

AT_REQ segment, which is sent from the correspondent host. This ratio decreases in proportion with the increase of the number of hops, for example, the ratio is only about 6% in case of 3 hops.

Table 4: Throughput Evaluation (Mbps)

	CH → MH	MH → CH
TCP-R(auth & primary)	7.10	8.07
MobileIP	2.67	3.32

Table 4 shows the measured throughput of TCP-R and MobileIP. The table shows the throughput when there is one gateway placed between the correspondent host and the mobile host. The difference between them is caused by the triangle routing, and the result shows TCP-R can reduce the overhead of triangle routing. The throughput of MobileIP with route optimization is expected to achieve slightly smaller results than that of TCP-R, because it takes binding table lookup costs for each IP packets.

6 Conclusions and Future Work

In this paper, we have demonstrated the TCP redirection mechanism and the necessity of host mobility support in the transport layer. TCP-R is a backwards-compatible extension of TCP. It provides the efficient and secure support for continuous operations in the mobile computing environments without any intermediate agents. We also show that the compensative operation such as a dynamic DNS update and MobileIP can cooperate with TCP-R for initiating new connections. A few redirect operations are added to TCP for mobility support, though TCP-R keeps all features of the standard TCP. The redirection mechanism provided by TCP-R is strong but quite simple, and overcomes the TCP retransmission timeouts. The measurement shows the overhead of redirect operations is prohibited very small.

We are currently investigating the framework of end-to-end parameter negotiation and extended socket API for TCP-R. It will negotiate the proper timer granularity and timeout duration in both sides. We are also developing an external mobile daemon, which supports network applications above TCP-R, and deals with move-detection events and connection lifetime management.

Acknowledgment

The authors would like to acknowledge Dean Nobuo Saito, Kunihito Matsumura, Yoshito Tobe, Shunichiro Okada and Tatsuya Hagino for their valuable suggestions and comments. We are also graceful to the members of the MKng project and SFC Media Center at Keio University.

References

[1] C.Perkins. "IP Mobility Support", RFC 2002, October, 1996.

[2] F.Teraoka, M.Tokoro. "Host Migration in Virtual Internet Protocol", In Proceedings of 1st International Networking Conference, June 1992.

[3] S.Thomson, Y.Rekhter, J.Bound. "Dynamic Updates in the Domain Name System", RFC 2136, April 1997.

[4] A.Myles, D.B.Johnson, C.Perkins. "A Mobile Host Protocol Supporting Route Optimization and Authentication", IEEE JSAC special issue on Mobile and Wireless Computing Networks, June 1995.

[5] T.Blackwell, et al. "Secure Short-Cut Routing for Mobile IP", In Proceedings of USENIX Summer 1994 Technical Conference, June 1994.

[6] C.Perkins, D.B.Johnson. "Mobility Support in IPv6", In Proceedings of Second Annual International Conference on Mobile Computing and Networking, November 1996.

[7] B.Ajay, B.R.Badrinath. "I-TCP: Indirect TCP for Mobile Hosts", In Proceedings of 15th International Conference on Distributed Computing Systems, May 1995.

[8] H.Balakrishnan, V.N.Padmanabhan, S.Seshan, R.H.Katz. "A Comparison of Mechanisms for Improving TCP Performance over Wireless Links", In Proceedings of ACM SIGCOMM '96, August 1996.

[9] R.T.Braden. "T/TCP – TCP Extensions for Transactions, Functional Specification", RFC 1644, July 1994.

[10] F.Bennett, T.Richardson, A.Harter. "Teleporting – Making Applications Mobile", In Proceedings of 1994 Workshop on Mobile Computing Systems and Applications, December 1994.

[11] T.Hosokawa. "PAO: FreeBSD Mobile Computing Package ", http://www.jp.freebsd.org/PAO, 1996

[12] "A PCMCIA support for BSD/OS", http://www.foretune.co.jp/wildboar/wildboar.html, 1995.

[13] W.Boulevard. "Transmission Control Protocol – DARPA Internet Program Protocol Specification", RFC 793, September 1981.

[14] R.T.Braden. "Requirements for Internet Hosts – Communication Layers", RFC 1122, October 1989.

[15] D.A.Maltz, D.B.Johnson. "The CMU Monarch Project IETF Mobile IPv4 Implementation User's Guide", ftp://ftp.monarch.cs.cmu.edu/pub/monarch/mobileip/users-guide.ps, February 1997.

[16] J.Inouye, S.Chen, C.Pu, J.Walpole. "System Support for Mobile Multimedia Applications", In Proceedings of the 7th International Workshop on NOSSDAV, May 1997.

[17] A.O.Freier, P.Karlton, P.C.Kocher. "The SSL Protocol Version 3.0", http://www.netscape.com/eng/ssl3, March 1996.

[18] M.A.Sirbu, J.C.Chuang. "Distributed Authentication in Kerberos Using Public Key Cryptography", Internet Society Symposium on Network and Distributed System Security, April 1997.