SCTP based Framework for Mobile Web Agent

Y.J. Lee, **Mohammed Atiquzzaman**

TR-OU-TNRL-05-116
September 2005

Telecommunication & Network Research Lab

School of Computer Science

THE UNIVERSITY OF OKLAHOMA

200 Felgar Street, Room 160, Norman, Oklahoma 73019-6151
(405)-325-0582, atiq@ou.edu, www.cs.ou.edu/~netlab

SCTP based Framework for Mobile Web Agent

Yong-Jin Lee[1] and Mohammed Atiquzzaman[2]

[1] Department of Computer Science, Woosong University,
17-2 Jayang-Dong, Dong-Ku, Taejon, 300-718, Korea
yjlee@woosong.ac.kr
[2] School of Computer Science, University of Oklahoma,
200 Felgar Street, Norman, OK 73019, USA
atiq@ou.edu

Abstract: Transmission Control Protocol (TCP) based web agents have has several deficiencies, such as performance degradation, head-of-line blocking, and unsupported mobility when applied to the mobile wireless environment. The Stream Control Transmission Protocol (SCTP) is a new transport protocol, which provides multi-streaming and multi-homing features. Recent SCTP extensions with dynamic address reconfiguration supports transport layer mobility. We propose web agent framework supporting seamless transport layer mobility in the wired and wireless environment. Our proposed framework for mobile web agent is composed of application engine to use the hypertext transfer protocol (HTTP) and protocol engine to deploy SCTP with dynamic address reconfiguration. We have described and evaluated the components of the framework necessary to implement mobile web agents in a ubiquitous environment. Mean response time is an important performance measure of web agents. Simulation results show that our SCTP based framework reduces the mean response time over a typical TCP based scheme by 12%, on the average.

1 Introduction

Ubiquitous computing environment is composed of cheap and small computers connected to the Internet and providing customized services according to user needs. A network architecture supporting ubiquitous computing environment is expected to have an IPv6 backbone network and an attached IPv4 Internet that connects mobile and wireless networks, i.e. it will be based on the all-IP network which will provide wired and wireless services while supporting host and user mobility.

The web agent we propose in this paper, works in the ubiquitous environment and provides mobile users with customized web service to use hypertext transfer protocol (HTTP) without user intervention. For example, while a user is roaming, portable web agents installed in personal digital assistants (PDA) or notebook computers can download objects, based on predefined profile, from web servers. Since the vast majority of IP traffic is transmitted using TCP, Transmission Control Protocol (TCP) application programming interface (API) is typically used to implement web agents. However, TCP based mobile web agents suffer from three problems: performance degradation, head-of-line (HOL) blocking, and unsupported mobility [1,2].

The Stream Control Transmission Protocol (SCTP) [3] has been standardized by IETF to overcome the above deficiencies of TCP. SCTP alleviates the performance degradation problem by incorporating enhanced congestion control schemes, some of which were developed for TCP. For example, SCTP uses fast retransmit in conjunction with Selective Acknowledgement (SACK), thereby speeding up loss detection and increasing bandwidth utilization [4]. In contrast to TCP, SCTP's mandatory use SACK can improve throughput

by increasing the congestion window size (*cwnd*) only when the full *cwnd* is utilized [5].

To overcome the HOL blocking problem of TCP, our proposed mobile web agent makes use of SCTP's multi-streaming feature to speed up the transfer of web objects. By sending each object in a separate stream, the HOL effect between objects is eliminated. If an object is lost during the transfer, other objects can be delivered to the web agent (at the upper layer) while the lost object is retransmitted from the web server. This results in an improved user response time with a single SCTP association for a particular HTML page.

Finally, to deal with the unsupported mobility problem of TCP, we utilize SCTP's multi-homing feature. SCTP multi-homing allows a single SCTP endpoint to support multiple IP addresses. In its current form, SCTP multi-homing support is only for redundancy. Recently, load-sharing SCTP (LS-SCTP) [6] has been suggested to aggregate the bandwidth of all active transmission paths between the communicating endpoints. The extended SCTP multi-homing feature (called dynamic IP address reconfiguration [7]) is capable of supporting transport layer mobility, and provides a mechanism for an SCTP endpoint to dynamically add and delete IP addresses during the lifetime of an SCTP association. Mobile SCTP [8] and Seamless IP Diversity based Generalized Mobility Architecture (SIGMA) [9] are handover schemes that utilize the dynamic IP address reconfiguration, and do not require any modification to the IP infrastructure. Secure SCTP [10] deals with the traffic redirection attack problem that can arise in dynamic IP address reconfiguration.

While SCTP can solve several deficiencies of TCP in a ubiquitous environment, it does not provide location management that is intrinsically supported by Mobile IP. Hence, location management in SIGMA is performed using the Domain Name Server (DNS) in the application layer [9]. However, use of DNS as location manager can cause scalability problem, while the use of home agent as location manger in mobile IP can result in complexity and inefficiency in the network. In this paper, we consider a web agent that always initiates the connection to the CN; we, therefore, do not need to consider the location management problem.

The *objective* of this paper is to propose an SCTP-based web agent framework for mobility. Mean response time between HTTP requests and replies is an important performance measure in a web environment. To compare the performance (mean response time) of the proposed framework with a TCP based scheme, we have collected results from experiments on our laboratory experimental testbed. Results show that the mean response time of our framework is lower than a TCP based scheme by about 12%, on the average.

The main *contributions* of this paper are: (i) proposing an architecture for mobile web agent framework, (ii) detailed functional description of the components in the web agent framework, and (iii) demonstration of performance enhancement of the proposed web agent framework over a typical TCP based scheme.

The rest of the paper is organized as follows. Section 2 describes the proposed framework for mobile web agent. Section 3 presents results of performance evaluation, followed by concluding remarks in Section 4.

2 Framework for the SCTP based mobile web agent

We present the framework for the mobile web agent in Fig. 1. The framework is mainly composed of the application engine performing the function of user interface and the protocol engine supporting the transport mobility based on SCTP. There exists SCTP API as the interface between two engines. Application engine exchanges the HTTP message with web server in the application layer. Protocol engine located in the transport layer has the mobility support component and the data component. Functions of the mobility support component are the movement detection and handover management. Data component in-

cludes transmission control block (TCB) and common data, which maintain the related information with the current association and mobility support.

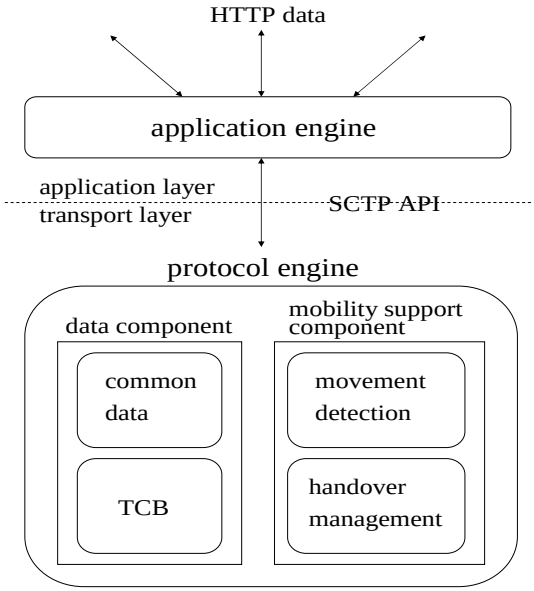

Fig. 1. Framework for the SCTP based mobile web agent.

2.1 Application engine

The application engine provides users with the HTTP service such as web content delivery. Since application engine interacts with SCTP in the transport layer, it avoids HOL blocking, which TCP based scheme experiences, and reduces the response time of an application. A typical web document contains several multimedia objects, such as image, voice and text. If any object is lost during receiving, the receiver does not display the previously received object until completely receiving the lost object. This HOL blocking increases the response time, especially in wireless networks with high packet loss. Our application engine can display objects more promptly by using the multi-streaming feature which avoids the HOL blocking.

In HTTP over TCP with persistent connection using pipelining, a client receives an HTML file with *M* embedded objects after 3-way-handshake. Then, the client that supports persistent connections pipelines its requests. That is, the client sends *M* requests simultaneously using pipelining. A server sends its responses to those requests in the same order that the requests were received.

The initialization of an association in HTTP over SCTP is completed after the exchange of four messages. The passive side of the association does not allocate resources for the association until the third of these messages has arrived and been validated. Last two messages of the four-way handshake can carry user data. With this piggybacking, SCTP has the same connection-establishment delay as TCP, namely one round trip time. SCTP's multi-streaming feature avoids the head-of-line blocking. Furthermore, SCTP does not limit the maximum number of objects for pipelining. To summarize, the main difference between HTTP over TCP and HTTP over SCTP is that SCTP can receive multiple objects in parallel using multi-stream.

Furthermore, SCTP has the following additional advantages when compared with a TCP based scheme: (i) The security of an application is increased remarkably. Since TCP based scheme establishes the connection setup using 3-way-handshake, it is defenseless against blind SYN attacks. On the other hand, our engine strengthens the security using the signed state-cookie in the 4-way-handshake for the connection setup. Although it uses four packets for connection establishment, it can combine the HTTP request into the third packet. Consequently, there is no extra overhead compared with the TCP based scheme. (ii) The fault tolerance of an application is enhanced. TCP based scheme uses only one connection path. If the TCP path is broken due to the physical layer problems, data cannot be transferred. However, our engine can continue to communicate using the alternate path which multi-homing feature provides. (iii) Seamless mobility for an application is supported. An existing TCP application suffers from the disconnection because it cannot change the currently bound IP address in TCB into the new IP address. However, our web agent continues to maintain the seamless connection with web server because it can modify the TCB using the dynamic address reconfiguration of SCTP.

SCTP API helps the application programmer who is familiar with TCP and UDP to quickly adapt to SCTP. By using the API, we can easily transform a typical TCP based application into the SCTP based application. For example, while the TCP application uses *socket*(*AF_INET, SOCK_STREAM, IPPROTO_TCP*) as the socket system call, SCTP application uses *socket*(*AF_INET, SOCK_STREAM, IPPROTO_SCTP*). System calls, *bind*(), *sctp_bindx*() are used for address binding. Several predefined events such as the new address acquisition are delivered to the application engine via SCTP notification. Application engine can obtain the status information such as contents and addresses of event by using *sctp_opt_info*(). To enhance flexibility, an implementation including lksctp [11] should provide an SCTP API to enable programmers to specify some parameters peculiar to SCTP, such as the number of outgoing streams to set up during negotiation, stream IDs used, etc.

2.2 Protocol engine

In this section, we describe in detail the various components of our proposed mobile web agent.

2.2.1 *Data component*

The data component defines necessary parameters for the protocol implementation. Common data is necessary for SCTP instance as follows: (i) Data consumer list related with the currently connected association. Data consumer means process identification information, such as file descriptor, pipe pointer, and table pointer. (ii) Secret key for the security of end-user. (iii) Address list indicating end points. (iv) Port number indicating the binding port number of end point.

Some important parameters are stored in TCB per association as follows: (i) Peer verification tag indicating the authentication value of the corresponding node (ii) My verification tag indicating the authentication value of local node (iii) State indicating the current status of SCTP such as the connection complete, shutdown, and stop (iv) Peer transport address list indicating the transport address list of the corresponding node (v) Local transport address list indicating the local IP address list (vi) Primary path indicating the primary destination address of the corresponding node.

2.2.2 *Mobility support component*

This component deals with the seamless transport mobility of web agent using the SCTP address configuration change (*ASCONF*) extension [7]. ASCONF extension defines the new IP address insertion (*add_ip_address*), the old IP address deletion (*delete_ip_address*), and primary IP address change (*set_primary_address*) chunks. According to the receipt of the above chunks, protocol engine changes the peer transport address list, local transport address list, and primary address stored in the TCB dynamically.

Generally, the mobility support problem in the wireless mobile network includes movement detection, handover management, and location management. Movement detection is a function of MN to identify and trace its own location change. Location management is a function of CN to trace the current location of MN in order to initiate the connection establishment with MN. Handover management is function of both MN and CN to provide the roaming MN with the seamless handover.

In this paper, we consider the web environment, where MN (web agent) always initiates connection setup to CN (web server). Thus, CN does not need location management. Nevertheless, if the location management function is necessary, we can add it into the mobility support component in Fig. 1. Timeline of the mobility support component is depicted in Fig. 2.

(1) *Movement detection*

Initially, mobile web agent hears router advertisement (RA) from old access router (AR), and finds the network prefix included in RA (1, Fig. 2). Web agent acquires its own IP address by using stateless auto-configuration of IPv6 based on the network prefix (when using IPv6) or inquiring to dynamic host configuration protocol (DHCPv4/v6) server (when using IPv4/IPv6). Web agent and web server establish the association by exchanging IP address. At this time, each end point specifies the primary IP address on which data is sent (2, Fig. 2). Information related with the association is recorded in each TCB of web agent and web server, followed by exchange of data.

The web agent, while communicating with the web server, moves from the coverage of old AR to the overlapped region which is covered by both the old and new AR's. Web agent hears new router prefix from new AR (3, Fig. 2), and detects its movement into new network by comparing its current network prefix (1, Fig. 2) with new network prefix (3, Fig. 2). If web agent uses IPv6, it can itself configure new IP address using stateless auto-configuration based on the network prefix. Otherwise, it can acquire a new IP address from the DHCPv4/v6 server (4, Fig. 2), which increases the required signaling time. Anyway, newly obtained IP address is bound on the local transport address list in the TCB of web agent (5, Fig. 2). These events are delivered to the application engine via SCTP notification.

(2) *Handover management*

After binding the new IP address on TCB, web agent informs the web server that it will use the new IP address by sending *ASCONF add_ip_address* (6, Fig. 2). Web server modifies its own TCB by adding the received new IP address of web agent and replies to the web agent by an *ASCONF add_ip_ack* (7, Fig. 2). At this time, web agent becomes multi-homed, and is thus reachable by two different networks. It can receive data on both old and new IP addresses. Consequently, if there is a physical problem with the path related to the primary address, the new IP address can be used as an alternate address.

As web agent leaves the overlapped region and enters the coverage of new AR, it experiences more packet loss on the primary path. If the amount of received packets on new IP address is greater than on the primary IP address, web agent sends out the *ASCONF set_primary* chuck to web server. This makes the web server to use the new IP address as primary address for data communications (8, Fig. 2). Web server replies to the *ASCONF set_primary_ack* chunk to web agent (9, Fig. 2).

As the web agent continues to move into the core coverage of new AR, the previous primary IP address becomes obsolete. Web agent sends out the *ASCONF delete_IP* chuck to web server, which eliminates the previous primary IP address (10, Fig. 2). The reason to delete the obsolete IP address is as the follows: We assume that the newly set primary path is broken in the coverage of new AR. If we did not delete the previous primary IP address in the binding list, it might become an alternate path. Thus, the data from web server may be redirected to the alternate path. However, the previous primary IP address cannot receive any data in the coverage of new AR. As a result, there exists the unnecessary traffic in the network. Handover is completed by the web server when responding by *ASCONF delete_ip_ack* to web agent (11, Fig. 2).

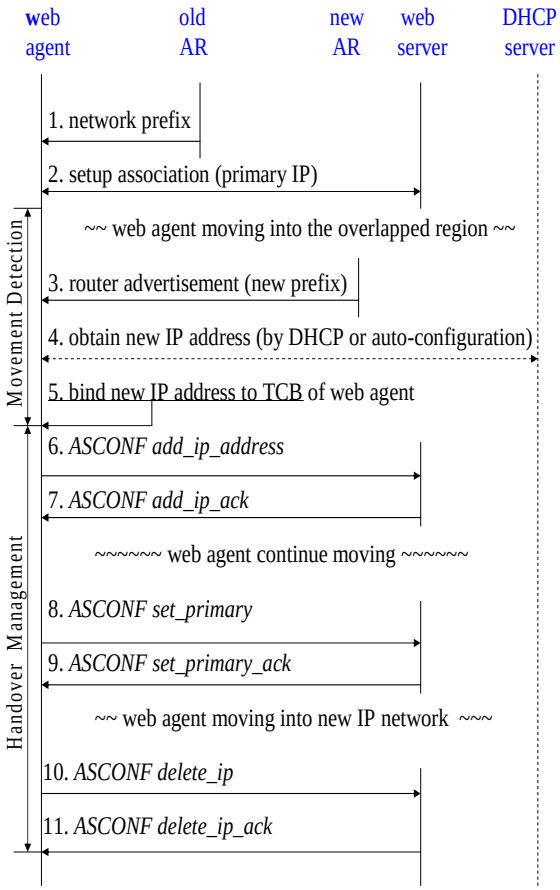

Fig. 2. Timeline for the mobility support component of web agent

3 Performance evaluation

3.1 Experimental Setup

In this section, we compare the performance of HTTP 1.1 persistent connection over our proposed SCTP based framework with a TCP based scheme. We use the mean response time as the performance criteria. Fig. 3 shows our laboratory testbed that was used to carry out the performance comparison, and Table 1 shows the host and network configurations of the testbed.

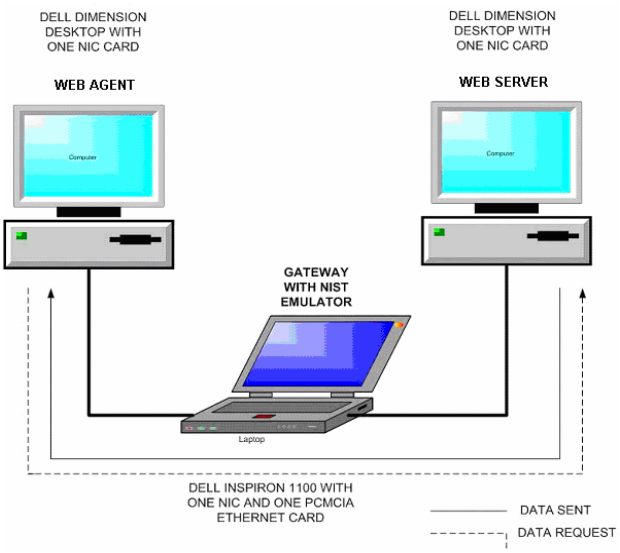

Fig. 3. Testbed for performance comparison.

Table 1 Host and network configuration of testbed.

node	*hardware*	*software*	*operating system*	*Network*
web server	Dell dimension desktop with one NIC card	TCP / SCTP server program	Redhat Linux 9 Kernel 2.6.2	eth0: 10.1.8.14, gateway: 0.1.8.5
web agent	Dell dimension desktop with one NIC card	TCP / SCTP agent program	Redhat Linux 9 Kernel 2.6.2	eth0: 10.1.10.6, gateway: 0.1.10.5
NIST emulator	Dell Inspiron-1100 laptop, one NIC card, one pcmcia Ethernet card	NIST emulator	Redhat Linux 9 Kernel 2.6.2	eth0: 10.1.8.5, gateway: default eth0: 10.1.10.5, gateway: default

To carry out the experiment for performance measurement, we wrote two Linux C server programs that mimic HTTP over TCP and SCTP servers, respectively. We also wrote two Linux C client programs to emulate pipelining (TCP/SCTP) and multi-streaming (SCTP), respectively. The main reason for writing our own HTTP server and agent is the lack of adequate support of SCTP by current HTTP client/server.

In order to mimic pipelining of TCP, we used *M* threads that simultaneously sent requests. The client sequentially receives HTTP replies from the server. The procedure for HTTP over SCTP client is as follows: a parent and a child process are forked. In the parent process, *M* threads emulate the pipelined transmission of *M* HTTP requests to the web

server. In the child process, *M* threads emulate multi-streaming for receiving *M* HTTP replies from the web server. To summarize, the main difference between HTTP over TCP and SCTP is that SCTP can receive multiple objects in parallel using multi-streaming.

We used the NIST Net emulator [11] between the web agent and web server to compare the performance for various packet losses. We measured the response time using the Ethereal protocol analyzer [12] that captured packets flowing between the server and agent. Table 2 shows test parameters that were used in our experiment. For all scenarios in the experiments, we used the following parameters: number of objects embedded in the HTML file (*M*) = 5, size of an object = 13. 5 KB, and maximum segment size (*MSS*) = 1,500 B. In scenario 1 of Table 2, we used packet loss rate (*p*) of 0.4 %, 1 %, 2 %, and 5 % to investigate the mean response time. Bandwidth (*bw*) and round trip time (*RTT*) were set at 40 Kbps and 256 ms, respectively. In scenario 2 and 3, we varied the bandwidth and round trip time, respectively.

Table 2 Test parameters for the laboratory experiment.

Test parameters	*Scenario 1*	*Scenario 2*	*Scenario 3*
Packet loss rate (*p*: *%*)	0.4, 1, 2, 5	1	1
Bandwidth (*bw: bps*)	40 K	40 K, 400 K, 3 M, 10 M	40 K
round trip time (*RTT: ms*)	256	256	55, 80, 256, 1000

3.2 Experiment Result

Figs. 4 - 6 show the mean response times corresponding to the scenarios in Table 2. Mean response times were computed for 50 trials for each test parameter for HTTP over TCP and SCTP.

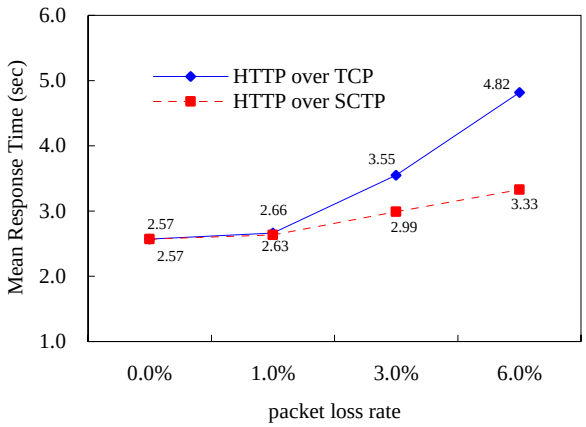

Fig. 4. Mean Response Time as a function of packet loss rate.

Fig. 4 shows the mean response time as a function of packet loss rate. For zero packet loss rate (p = 0 %), there is no HOL blocking in the TCP based scheme. Therefore, there is no difference in the mean response time between SCTP based framework and TCP based scheme. In addition, we found that the gap between HTTP over TCP and HTTP over SCTP

increased as the packet loss rate increased. We attribute this to the multi-streaming feature and the effective error recovery of SCTP.

As the packet loss rate increases, a TCP based scheme must retransmit the lost packets more frequently; moreover, it suffers from head-of-line blocking, resulting in a large response time. On the other hand, SCTP based framework uses multi-streaming, and hence does not suffer from HOL blocking. Furthermore, the proposed SCTP based framework can perform fast recovery from errors using SACK, and thus can detect losses faster than a TCP based scheme.

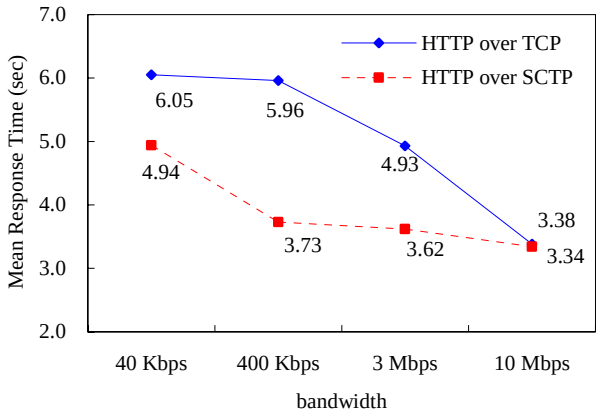

Fig. 5. Mean Response Time as a function of the bandwidth

Fig. 5 shows that the mean response time is inversely proportional to the bandwidth. It is natural that larger bandwidth reduces the total transfer time, resulting in a smaller mean response time. As the bandwidth increases, the benefit of SCTP over TCP becomes smaller. Data is transferred very quickly in a high bandwidth network, and thus the effect of fast retransmission is negligible.

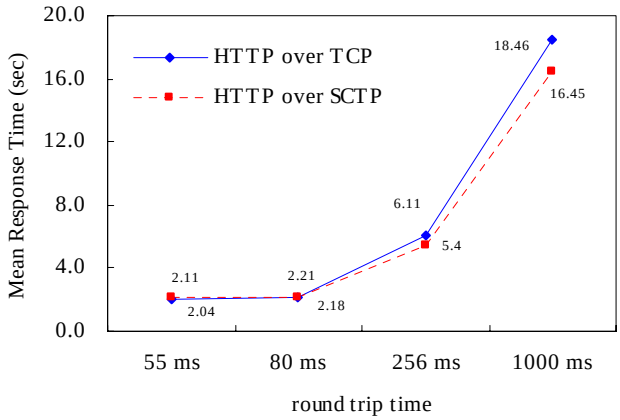

Fig. 6. Mean Response Time as a function of RTT.

In Fig. 6, the rate of increase of the mean response time in HTTP over TCP is about 900% between 0.055 s and 1s, while it is 800% for HTTP over SCTP. Since TCP cannot utilize the multi-streaming, all objects experience large *RTT*. On the other hand, multiple objects

in HTTP over SCTP experience only one *RTT*. Thus, as the number of objects and *RTT* increase, the performance gap between SCTP and TCP increases.

We define the savings rate of SCTP based framework over TCP based scheme in an HTTP environment by Eq. (1),

$$Savings\ Rate = (T_{TCP} - T_{SCTP}) / T_{TCP} \times 100\ (\%) \quad (1)$$

where, T_{TCP} and T_{SCTP} represent the mean response time of HTTP over TCP and SCTP based frameworks, respectively. Fig. 7 depicts the savings rate that was measured in our experiment, where the savings rates of SCTP over TCP are 11.9%, 20.8%, and 5.5%, for packet loss rate, bandwidth, and round trip times, respectively. To summarize, our web agent framework reduces the mean response time overTCP based scheme by 12%, on the average.

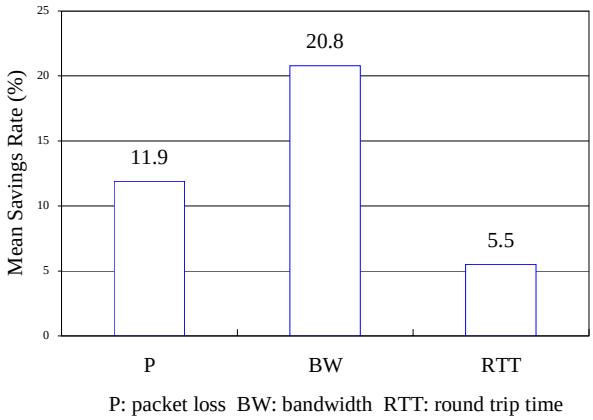

Fig. 7. Mean Savings Rate averaged over packet loss rates, bandwidths, and RTTs.

4 Conclusions

In this paper, we have proposed and evaluated a Stream Control Transport Protocol based mobile web agent framework. We have investigated and described the important characteristics and functions necessary to implement the web agent to support seamless mobility. To compare the performance of HTTP over SCTP (deployed in our web agent framework) with that of HTTP over TCP (used in a typical mobile web agent), we have carried out experiment in our laboratory experimental testbed. Results show that our web agent framework can reduce the mean response time over a typical TCP based scheme by 12%, on an average.

References

1. Bai, H, Fu, S. and Atiquzzaman, M.: Transport Layer Design in Mobile Wireless Network. Design and Analysis of Wireless Networks. Nova Science Publishers (2004).
2. Jamalipour, A.: The Wireless Mobile Internet. John Wiely & Sons Ltd. (2003) 368-384.
3. Caro, A., Iyengar, J., Amer, P., Ladha, S., Heinz, G. and Shah, K.: SCTP: A Proposed Standard for Robust Internet Data Transport. IEEE Computer, Vol. 36. (2003) 56-63.

4. Fu, S. and Atiquzzaman, M.: SCTP: State of the Art in Research, Products, and Challenges. IEEE Communication Magazine, Vol. 42. (2004) 64-76.
5. Alamgir, M, Atiquzzaman, M., and Ivancic, W.: Effect of Congestion Control on the Performance of TCP and SCTP over Satellite Network. NASA Earth Science Technology. (2002).
6. Al, A., Saadawi, T. and Lee, M.: Improving Throughput and Reliability in Mobile Wireless Networks via Transport Layer Bandwidth Aggregation. Computer Networks, Vol. 46. (2004) 635-649.
7. Stewart, R. et al.: Stream Control Transmission Protocol (SCTP) Dynamic Address Reconfiguration. IETF Internet draft, draft-ietf-tsvwg-addip-sctp-06.txt. (2003).
8. Koh, S., Chang, M. and Lee, M.: mSCTP for Soft Handover in Transport Layer. IEEE Communication Letters, Vol. 8. (2004) 189-191.
9. Fu, S., Ma, L., Atiquzzaman, M., and Lee, Yong-Jin: Architecture and Performance of SIGMA: A Seamless Mobility Architecture for Data Networks, IEEE International Conference on Communi- cations (ICC), Seoul, Korea, May 16-20. (2005).
10. Unurkhaan, E., Pathgeb, E. and Jungmaier, A.: Secure SCTP- A Versatile Secure Transport Protocol. Telecommunication Systems, Vol. 27. (2004) 273-296.
11. NIST NET Emulator: http://snad.ncsl.nist.gov/nistnet/.
12. Etherreal Protocol Analyzer: www.ethereal.com/.