Web Agent Supporting Transport Layer Mobility

Yong-Jin Lee[1] and M. Atiquzzaman[2]

[1] Department of Technology Education,
Korea National University of Education, 363-791
yjlee1026@daum.net
[2] School of Computer Science, University of Oklahoma,
200 Felgar Street, Norman, OK 73019, USA
atiq@ou.edu

Abstract. Typical transmission control protocol (TCP) based web agents in mobile wireless environment have several deficiencies, such as performance degradation, head-of-line blocking, and unsupported mobility. The Stream Control Transmission Protocol (SCTP) is a new transport protocol, which provides multi-streaming and multi-homing features. SCTP extensions for dynamic address reconfiguration will support transport layer mobility. We address SCTP based web agents supporting seamless transport layer mobility in the wired and wireless environment. Our proposed SCTP based 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 investigated and described the components 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 web agent reduces the mean response time of a typical TCP based web agent by 12 % on the average.

1 Introduction

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

A web agent, which we address in this paper, works in the ubiquitous environment and provides mobile users with customized web services to use hypertext transfer protocol (HTTP) automatically without user intervention. For example, portable web agents installed in personal digital assistants (PDA) or notebook computers of roaming users can download objects from web servers based on a predefined profile. Since the vast majority of IP traffic is transmitted using TCP, typical web agents use Transmission Control Protocol (TCP) Application Programming Interface (API). However, TCP-based mobile web agents suffer from the following three deficiencies: performance degradation, Head-of-Line (HOL) blocking, and unsupported mobility [1,2].

Stream Control Transmission Protocol (SCTP) [3] has been proposed by IETF to overcome the above deficiencies of TCP. The performance degradation problem is

T. Magedanz et al.(Eds.): MATA 2005, LNCS 3744, pp. 1 – 11, 2005.

alleviated in SCTP by incorporating the enhanced features of TCP congestion control schemes. For example, SCTP uses the Selective Acknowledgement (SACK)-based Fast Retransmit algorithm used in TCP. This scheme speeds up loss detection and increases bandwidth utilization [4]. Use of SACK is mandatory in SCTP. In addition, SCTP 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, we use SCTP's multi-streaming feature to speed up the transfer of web objects. By transmitting each object in a separate stream, the HOL effect between different objects is eliminated. If one object is lost during the transfer, others can be delivered to the web agent at the upper layer while the lost object is being retransmitted from the web server. This results in a better response time to users with only one SCTP association for a particular HTML page.

Finally, to deal with the unsupported mobility problem of TCP, we utilize the extended SCTP 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 communicating endpoints. The extended SCTP multi-homing feature (called dynamic IP address reconfiguration [7]) is capable of supporting transport layer mobility. This feature provides a mechanism that allows 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] utilize the dynamic IP address reconfiguration feature of SCTP, without requiring any modification to the IP infrastructure. Secure SCTP [10] deals with the traffic redirection attack problem that can arise from dynamic IP address reconfiguration.

While SCTP can solve several deficiencies of TCP for mobility in the ubiquitous environment, it does not provide the location management function that Mobile IP supports intrinsically. Hence, location management in SCTP is performed by the Domain Name Server (DNS) in the application layer [9] or Mobile IP in the network layer [8]. Use of DNS can cause scalability problem, and the scheme used 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 setup to web servers. Location management is, therefore, outside the scope of this paper.

We propose a web agent based on SCTP and its dynamic IP address reconfiguration feature. The most important performance measure in a web environment is the mean response time between HTTP requests and replies. To compare the performance of our proposed SCTP based web agent and a typical TCP based web agent, we have carried out simulation on our experimental testbed. Results show that our web agent's mean response time is less than a TCP based web agent by about 12% on average.

The main contributions of this paper are: (i) propose an SCTP based mobile web agent architecture; (ii) describe the functions of the components of the web agent; and (iii) evaluate the performance of the proposed web agent and compare with a typical TCP-based web agent.

The rest of the paper is organized as follows. Section 2 describes the architecture and functions of the proposed SCTP based web agent. Section 3 presents results of performance evaluation, and Section 4 concludes the paper.

2 Architecture of the SCTP-Based Mobile Web Agent

We present the architecture of the SCTP-based mobile web agent in Fig. 1. It consists of an *application engine* performing the function of user interface and an SCTP-based *protocol engine* supporting transport mobility. An SCTP API serves as the interface between the two engines. The application engine exchanges HTTP message with the web server in the application layer. The protocol engine, located in the transport layer, has the mobility support component and the data component. The functions of the mobility support component are movement detection and handover management. The data component includes Transmission Control Block (TCB) and Common Data Block (CDB), which maintain related information regarding the current association and mobility support.

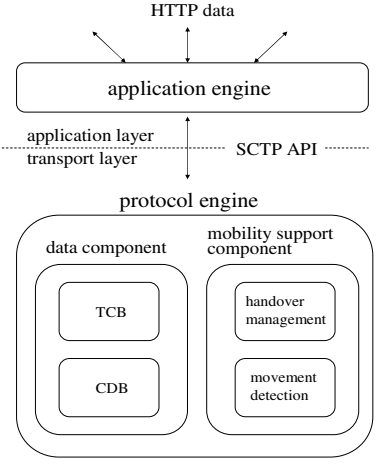

Fig. 1. Architecture of our proposed SCTP based mobile web agent

2.1 Application Engine

The application engine provides users with HTTP service, such as web content delivery. Since the application engine interacts with SCTP in the transport layer, it avoids HOL blocking (which TCP based applications experience) 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 transmission, the receiver does not display the previously received object until the lost object has been completely received. 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 evades the HOL blocking.

TCP web agent receives an HTML file with M embedded objects after 3-way-handshake using HTTP (it hereafter refers to HTTP 1.1 with persistent connection and pipelining its requests: Fig.2 (a)). That is, TCP web agent sends M requests simultaneously using pipelining. A server sends its responses to those requests in the same order that the requests were received.

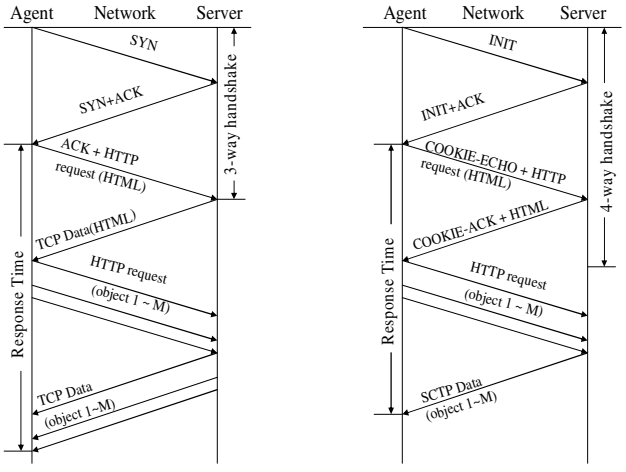

Fig. 2. Timelines for TCP web agent and SCTP web agent

The initialization of an association in SCTP web agent using HTTP (Fig.2 (b)) 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 web agent has the same connection-establishment delay as TCP web agent, namely one round trip time. Furthermore, SCTP web agent does not limit the maximum number of objects for pipelining. To summarize, the main difference between TCP web agent and SCTP web agent is that SCTP web agent can receive multiple objects in parallel using multi-stream.

Furthermore, SCTP web agent has the following additional advantages when compared with a TCP web agent: (i) The security of an application is increased remarkably. Since TCP web agent establishes the connection setup using 3-way-handshake, it is defenseless against blind SYN attacks. On the other hand, our SCTP based 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 web agent. (ii) The fault tolerance of an application is enhanced. TCP web agent 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

application into the SCTP 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 the linux kernel stream control transmission protocol (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 protocol implementation. CDB 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 correspondent 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 correspondent node; (v) Local transport address list indicating the local IP address list; (vi) Primary path indicating the primary destination address of the correspondent 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 the mobile node (MN) to identify and trace its own location change. Location management is a function of the correspondent node (CN) to trace the current location of MN in order to initiate the connection establishment with MN. Handover management is a 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. 3.

(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. 3). Web agent acquires its own IP address by using stateless auto-configuration of IPv6 based on the network prefix (when using IPv6) or querying the dynamic host configuration protocol (DHCPv4/v6) server (when using IPv4/IPv6). Web agent and web server establish the association by exchanging IP addresses. At this time, each end point specifies the primary IP address on which data is sent (2, Fig. 3). Information related to the association is recorded in each TCB of web agent and web server, followed by exchange of data.

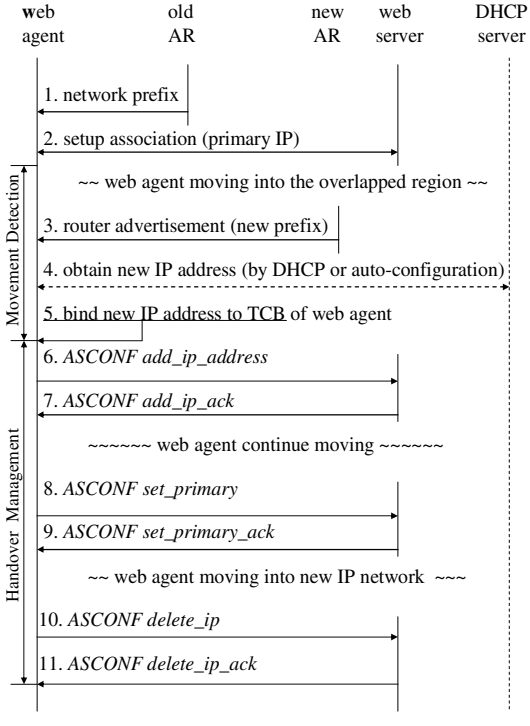

Fig. 3. Timeline for the mobility support component of web agent

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. 3), and detects its

movement into new network by comparing its current network prefix (1, Fig. 3) with new network prefix (3, Fig. 3). If web agent uses IPv6, it can itself configure a new IP address using stateless auto-configuration based on the network prefix. Alternatively, it can acquire a new IP address from the DHCPv4/v6 server (4, Fig. 3), which increases the required signaling time. Anyway, newly obtained IP address is bound to the local transport address list in the TCB of web agent (5, Fig. 3). 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. 3). 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. 3). 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 the 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 informs the web server to use the new IP address as the primary address for data communications (8, Fig. 3). Web server replies by sending an *ASCONF set_primary_ack* chunk to the web agent (9, Fig. 3).

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 an *ASCONF delete_IP* chuck to the web server, which eliminates the previous primary IP address (10, Fig. 3). The reason to delete the obsolete IP address is as the follows: We assume that the new set primary path is broken in the coverage of new AR. If the previous primary IP address was not deleted from the binding list, it might become an alternate path. Thus, data from the web server may be redirected to the alternate path. However, the previous primary IP address cannot receive any data in the coverage of the new AR. This results in unnecessary traffic in the network. Handover is completed when the web server responds by sending *ASCONF delete_ip_ack* to the web agent (11, Fig. 3).

3 Performance Evaluation

3.1 Experimental Setup

In this section, we compare the performance of SCTP web agent with TCP web agent in terms of the mean response time. Fig. 4 shows our testbed for collecting the data. Table 1 shows the host and network configurations of the testbed.

For this experiment, we wrote two Linux C server programs that mimic HTTP over TCP and SCTP servers, respectively. We also wrote two Linux C web agent programs

to simulate pipelining (TCP/SCTP) and multi-streaming (SCTP), respectively. The main reason for writing our own HTTP server and agent is due to the lack of adequate support of SCTP by current HTTP client/server.

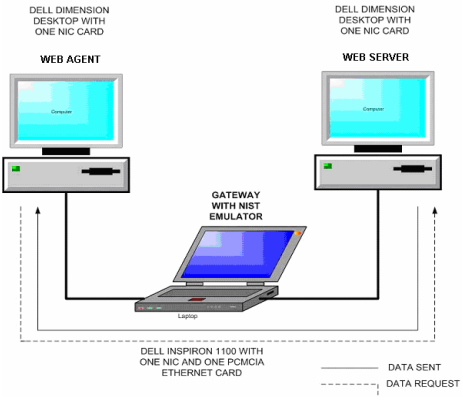

Fig. 4. Testbed for the simulation

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

In order to mimic the pipelining of TCP web agent, we used *M* threads that send requests simultaneously. TCP web agent receives HTTP reply sequentially from the server. The procedure for SCTP web agent is as follows: we first fork two processes: parent and child. In the parent process, *M* threads simulate the pipelining for sending *M* HTTP requests to the web server. In the child process, *M* threads simulate multi-streaming for receiving *M* HTTP replies from the web server. To summarize, the main difference between the TCP and SCTP web agents is that SCTP web agent can receive multiple objects in parallel using multi-streaming.

We used the NIST Net emulator [12], between the web agent and web server, to compare the performance under varying packet loss rates. We measured the response time using Ethereal protocol analyzer [13] that captures packets flowing between the server and agent. Table 2 represents test parameters used for our experiment. We have

carried out the experiment using the following common parameters for all scenarios: the number of objects embedded in the HTML file (M) = 5, size of one object = 13. 5 KB, and maximum segment size (MSS) = 1,500 B. In scenario 1 of Table 2, we investigate the mean response time at packet loss rates (p) of 0.4 %, 1 %, 2 %, and 5 %. Bandwidth (bw) and round trip time (RTT) were set at 40 Kbps and 256 ms, respectively. In scenarios 2 and 3, we varied the bandwidth and round trip time, respectively.

Table 2. Test parameters for the 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. 5~7 show the mean response times corresponding to the scenarios in Table 2. Mean response times are computed for 50 trials for each test parameter for TCP web agent and SCTP web agent, respectively.

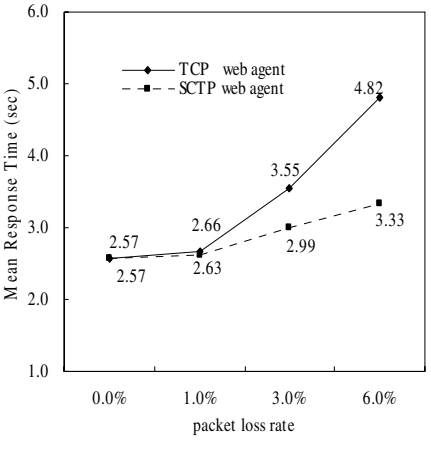

Fig. 5. Mean response time as a function of packet loss

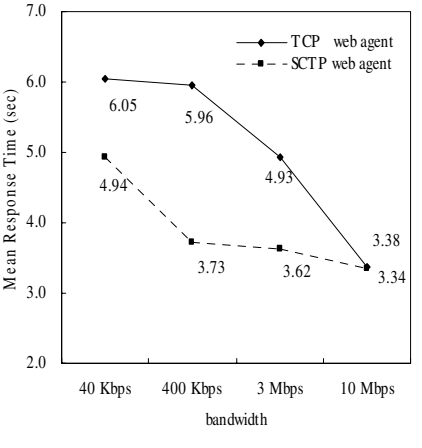

Fig. 6. Mean response time as a function of bandwidth

Fig. 5 represents the mean response time as a function of packet loss rate. When the packet loss rate is zero (p = 0 %), there is no HOL blocking in a TCP web agent. Therefore, there is no difference in the mean response time between SCTP and TCP web agents. In addition, we found that the gap between TCP and SCTP web agents increases as the packet loss rate increases. This is due to the multi-streaming feature

and the effective error recovery of SCTP. As the packet loss rate increases, the TCP web agent must retransmit the lost packets more frequently; moreover, it suffers from the head-of-line blocking causing the large response time. On the other hand, SCTP web agent does not suffer from the HOL blocking due to the use of the multi-streaming feature. Furthermore, the proposed SCTP web agent can detect losses more quickly than TCP web agent. Thus, it can perform fast recovery from the error using the SACK mechanism.

Fig. 6 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 web agent over TCP web agent becomes smaller. This is because of the minimal effect of fast retransmit for high bandwidth.

In Fig. 7, we found that the rate of increase of the mean response time in TCP web agent is about 900 % between 0.055 s and 1s, while it is 800 % for SCTP web agent. Since TCP web agent cannot utilize the multi-streaming feature, all objects experience large *RTT*. On the other hand, multiple objects in SCTP web agent experience only one *RTT*. Thus, as the number of objects and *RTT* increase, the performance gap between SCTP web agent and TCP web agent increases.

We define the savings rate of SCTP web agent over TCP web agent by Eq. (1),

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

T_{TCP} and T_{SCTP} represent the mean response time of TCP and SCTP-based web agents, respectively. Fig. 8 depicts the mean savings rate in our experiment, where the mean savings rates of SCTP agent over TCP agent for packet loss rate, bandwidth, and round trip time are 11.9 %, 20. 8 %, and 5.5 %, respectively. To summarize, we can reduce the mean response time of TCP web agent by 12 % on average using SCTP web agent.

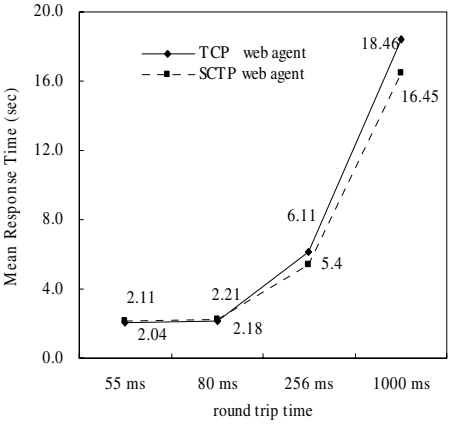

Fig. 7. Mean response time as a function of RTT

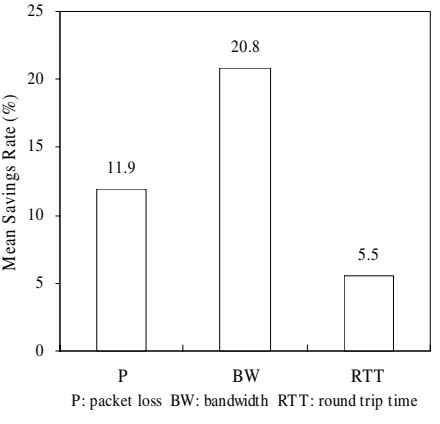

Fig. 8. Mean savings rate of SCTP over TCP

4 Conclusions

In this paper, we have proposed a Stream Control Transport Protocol-based web agent. The web agent supports transport layer mobility and can overcome many of the limitations of network layer-based schemes. To demonstrate the efficiency of our web agent, we have carried out experiments in our experimental testbed. Simulation results show that our SCTP-based web agent can reduce the mean response time over a typical TCP-based web agent by 12 %, on average. Future work includes performance evaluation of multi-homing effect in a wireless web environment.

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. (2003) 20-27.
4. Fu, S. and Atiquzzaman, M.: SCTP: State of the Art in Research, Products, and Challenges. IEEE Communication Magazine. (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. Linux kernel stream control transmission protocol (lksctp): http://lksctp.sourceforge.net/
12. NIST NET Emulator: http://snad.ncsl.nist.gov/nistnet/.
13. Etherreal Protocol Analyzer: www.ethereal.com/.