Mean Response Time Estimation for HTTP over SCTP in Wireless Environment

Y.J. Lee, **Mohammed Atiquzzaman** , **S.K. Sivagurunathan**

TR-OU-TNRL-05-115
September 2005

Mean Response Time Estimation for HTTP over SCTP in Wireless Environment

Y.J. Lee
Korea National University of Education
Dept. of Technology Education
San 7, Darakri
Chungbuk, 363-791, Korea

M. Atiquzzaman, S.K. Sivagurunathan
University of Oklahoma
School of Computer Science
200 Felgar Street, Room 120
Norman, OK 73019-6151

***Abstract*-HTTP (Hyper Text Transfer Protocol) and TCP (Transmission Control Protocol) are usually used to retrieve objects in the Internet. Stream Control Transmission Protocol (SCTP) has attractive multi-streaming feature, which allows for independent delivery among streams, thus can avoid the head-of-line blocking experienced by TCP and reduce the mean response time of users. We present an analytical model and algorithm to estimate the mean response time for HTTP over SCTP and compare with that of HTTP over TCP in the wireless environment. We validate the accuracy of our model using experiments .It is shown that mean response time for HTTP over SCTP is less than that for HTTP over TCP by 20% on average.**

I. INTRODUCTION

HTTP is a transfer protocol used by the World Wide Web distributed hypermedia systems to retrieve objects in the Internet. HTTP is a connection-oriented protocol and uses TCP as the transport protocol.

The current HTTP version reduces response time by using persistent connections and pipelining. However, there is a mismatch between the requirements of HTTP and the functionality provided by TCP. When multiple embedded objects are being transferred using HTTP, it is desired that each of the objects be reliably transferred to the destination independently, Rather than ordered delivery, , it is more important to reduce the perceived latency of HTTP users. An average user is only interested in a fast response time.

Stream Control Transfer Protocol (SCTP) [8] is a new transport protocol that provides a message oriented and reliable transport service to Internet users. In addition, SCTP offers advanced delivery options, particularly desirable for multimedia applications that TCP does not provide. We can partition data into multiple streams that can be delivered independently using the multi-streaming function provided by SCTP. Message loss in any of the streams will thus only affect delivery within that stream, and not in other streams.

The research reported in this paper was funded by NASA Grant NAG3-2922.

Unlike SCTP, TCP provides a single stream of data and strict ordered delivery. For a number of applications, this characteristic of strict sequence preservation is not truly necessary. Especially, for web application, it is generally not necessary to maintain sequence between the presentation of objects, and in some cases, it may be possible to present parts of a single object out of sequence. Eventually the goal is to deliver all objects as soon as possible. Delivery of out of sequence objects may result in better perceived performance, as parts of the web page can be displayed rather than waiting for all of the information to be received. This is called the head-of-line blocking. Multi-streaming can be used for independent delivery among streams within an SCTP association, thus reducing the risk of head-of-line blocking [3]. This will result in a better response time to users through simultaneous retrieval of multiple objects [4].

The TCP model of Padhye *et al.* [7] assumed steady state bulk data TCP transfers. However, most TCP connections carrying HTTP data in Internet today are short transfers rather than being bulk transfers. Thus, startup effects such as connection establishment and slow start dominate the performance of Web. In order to describe these effects, Cardwell *et al.* [2] extended the previous steady-state model; it, however, does not consider the TCP latency due to the slow start after timeout. Jiong *et al.* [5] presented a simple model of short TCP transfers by considering the slow start period after retransmission timeout, and improved the model [2] on the influence of slow start period after retransmission timeout. However, they assumed a single packet loss. In addition, the above studies did not consider the head-of-line blocking effect of TCP which affects the mean response time perceived by end users.

Previous studies have considered the performance of Session Initiated Protocol (SIP) [1] and File Transfer

Protocol (FTP) over SCTP [6]. In this paper, we present the analytical model and experimental results for HTTP over SCTP with a view to comparing the mean response time of end users. We also compare the mean response time of HHTP over SCTP with that of HTTP over TCP.

The rest of the paper is organized as follows. In Section II, we describe our analytical model and algorithm. In Section III, we describe our experimental setup and compare the experimental results with those obtained from the analytical model, followed by concluding remarks in Section IV.

II. MODELING AND ALGORITHM

A. Terminology and assumptions

In this section, we describe the terminology and assumptions that have been used to derive the analytical model for HTTP over SCTP.

***A.1** Terminology*

RTT: round trip time from the server to the client
R: transmission rate of the link from the server to the client (bps)
O: size of the object to be transferred (bits)
MSS: maximum segment size (bits)
K: number of windows that cover the object
Q: number of times the server would stall
M: number of reference objects
N: total number of packets in object
P_l: maximum number of pipelines
p: packet loss probability
a: expected number of lost packet
x: expected value of packet number when the loss occurs
C: the window number in which x is included
y: window size that covers the expected value of x
D_T: data transfer time
T_R: data retransmission time in the case of data loss
SC: slow start time per object
HOB: head of line blocking time
I_T: initial connection setup and HTML transfer time
th: number of packets for threshold in the congestion control
A_{th}: amount of packets sent until T
L: amount of packets sent during the linear phase of congestion control
S_x: slow start time until x when packet loss occurs during slow start phase
S_{th}: slow start time until threshold T when packet loss occurs during the linear phase
T_x: total time until packet loss occurs during slow start phase
T_L: total time when packet loss occurs during congestion avoidance
TO_{detect}: expected time for retransmission timeout

***A.2** Assumptions*

(1) All objects to be transferred using HTTP are of identical size.
(2) Packets are sent to the upper layer based on the window unit.

***B.** Modeling*

Figure 1 illustrates sending an HTTP request for an object and receiving an acknowledgement from the server. If the total number of packets in an object is $N = \lceil O/MSS \rceil$, and the packet loss probability is p, the expected number of packet loss, according to the Binomial distribution, is $a = \lceil Np \rceil$. In Figure 1, $S_1, S_2,.., S_a$ represent the slow start times until the first, second,…, a^{th} packet loss, respectively. $D_{T1}, D_{T2},.., D_{Ta}$ show the data transfer time for part of object until the first, second,.., a^{th} packet loss, respectively. $HOB_1, HOB_2,., HOB_a$ represent the head-of-line blocking time for the client to wait for the completion of retransmission. Thus, the total response time = RTT + object transfer time = $RTT + (S_1 + D_{T1} + R_{T1} + HOB_1) + \cdots + (S_a + D_{Ta} + R_{Ta} + HOB_a)$. For a constant RTT in a given environment, the total response time depends on the object transfer time. Since, the sum of the data transfer times can be treated as a constant, the slow start time, retransmission time and head-of-line blocking time contribute to the total response time. If there is no packet loss, the additional slow-start time, retransmission time and head-of-line blocking time due to loss are not necessary. We, therefore, only need the slow start time (SC) for normal transmission and data transfer time (O/R).

Next we consider the case of packet loss. To find the slow start time (S_1), the expected number of packets sent before a loss (not including the lost packet) is given by

$$x = \frac{1-(1-p)^N}{p} + (1-p)^N \tag{1}$$

Thus, the expected number of packets sent before the first loss is (x-1). Slow start time for an object is given by

$$SC = v \cdot \left[RTT + \frac{MSS}{R} \right] - (2^v - 1) \cdot \frac{MSS}{R} \tag{2}$$

Here, v = min [Q,K-1]. Q represents the number of times the server would stall and is given by

$$Q = \left\lfloor \log_2 \left(1 + \frac{RTT}{MSS \;/\; R} \right) \right\rfloor + 1 \tag{3}$$

K represents the number of windows that cover the object without packet loss and given by

$$K = \left\lceil \log_2 \left(\frac{O}{MSS} + 1 \right) \right\rceil \tag{4}$$

In case of packet loss, since we send only x packets before the loss, K is given by

$$K \approx \lceil \log_2 (x+1) \rceil \tag{5}$$

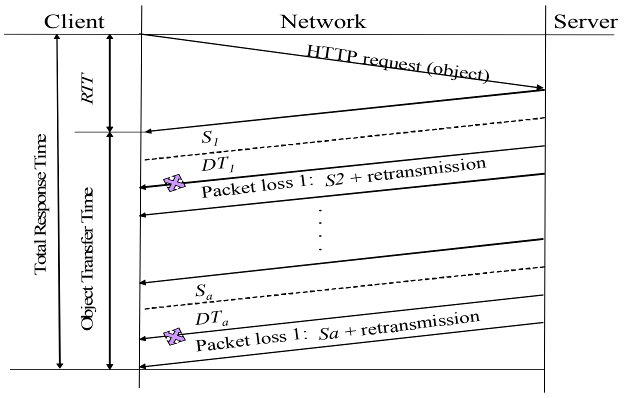

Figure 1. HTTP Request for an object.

Substituting the above values into SC, we can obtain the slow start time until x (S_x) as

$$\begin{aligned} S_x &= v \cdot \left[RTT + \frac{MSS}{R} \right] - (2^v - 1) \cdot \frac{MSS}{R} \\ Q &= \left\lfloor \log_2 \left(1 + \frac{RTT}{MSS/R} \right) \right\rfloor + 1 \\ K &= \lceil \log_2 x + 1 \rceil \\ v &= \min[Q, K-1] \end{aligned} \tag{6}$$

Now, let's consider the initial threshold (T). The total number of packets sent until T is

$$A_{th} = \begin{cases} 2^{\lceil \log_2 (th+1) \rceil} - 1, & \text{if } th = 2^k \\ 2^{\lceil \log_2 (th+1) \rceil} - 1 + th, & \text{if } th \neq 2^k \end{cases} \tag{7}$$

In Figure 1, S_1 can be different depending on the following two cases: In the case of $x+1 \leq A_{th}$, $S_1 = S_x$, otherwise, $S_1 = S_{th}$.

If $x+1 \leq A_{th}$, it means that packet loss occurred in the slow start phase as shown in Figure 2. S_x represents the slow start time for x packets.

Next, the current window number is given by

$$\begin{aligned} C &= \min\{K : 2^0 + 2^1 + \cdots + 2^{K-1} \geq x\} \\ &= \min\{K : 2^K - 1 \geq x\} \\ &= \min\{K : \log_2 (x+1)\} = \lceil \log_2 (x+1) \rceil \end{aligned} \tag{8}$$

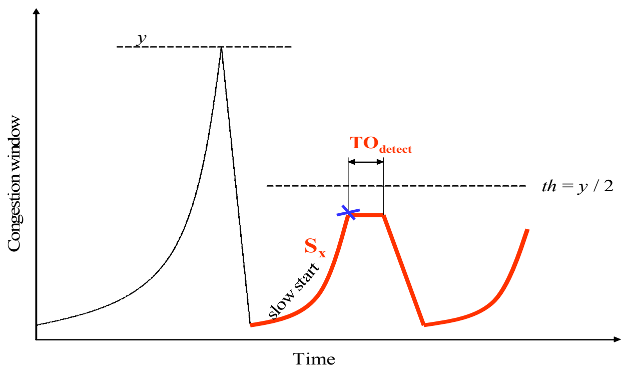

Figure 2. Packet loss during slow-start phase.

Thus, the window size (y) which covers the expected number of packets when loss occurs (x) is given by

$$y = \begin{cases} 2^{C-1} - (2^C - N) + 1, & \text{if } N < 2^C \\ 2^{C-1}, & \text{otherwise} \end{cases} \tag{9}$$

If $y > th$, we set $th = y$. The number of packets sent before the loss in the C^{th} windows is

$$\begin{aligned} b &= x + 1 - (2^0 + 2^1 + \cdots + 2^{C-2}) \\ &= x + 1 - 2^{C-1} \end{aligned} \tag{10}$$

The number of packets to be resent, including the lost packet, is (y-b). This number will be added to the remainder and processed in the next step as shown in the algorithm. Since b packets must wait in the C^{th} window, and (y-b) packets are to be sent in the next slow-start time, head-of-line blocking time is given by

$$HOB_1 = (y - b) \cdot \frac{MSS}{R} \tag{11}$$

Thus, total time for slow start phase, including the transmission time of the lost packet is

$$T_x = S_x + (x+1) \cdot MSS/R + HOB_1 + TO_{detect} \tag{12}$$

In Equation (12), $TO_{detect} = 3/2\ RTT$ which represents receiving three duplicate ACKs for detecting a lost packet for fast retransmission. Before proceeding to the next step, we set $N = N - x + (y - b)$ and $T = \lceil T/2 \rceil$.

$x+1 > A_{th}$ implies a packet loss in the congestion avoidance phase, as shown in Figure 3, where $x = A_{th} + L$. So, the number of packets sent before the loss during this phase is

$$L = x - A_{th} \tag{13}$$

In this phase, each packet is followed by an acknowledgement. The additional time to send L packets is $L \times RTT$. Slow start time until th (S_{th}) is

$$S_{th} = v \cdot \left[RTT + \frac{MSS}{R} \right] - (2^{v} - 1) \cdot \frac{MSS}{R}$$

$$\begin{aligned} Q &= \left\lfloor \log_2 \left(1 + \frac{RTT}{MSS\ /\ R} \right) \right\rfloor + 1 \\ K &= \left\lceil \log_2 A_{th} \right\rceil \\ v &= \min[\, Q, K - 1] \end{aligned} \tag{14}$$

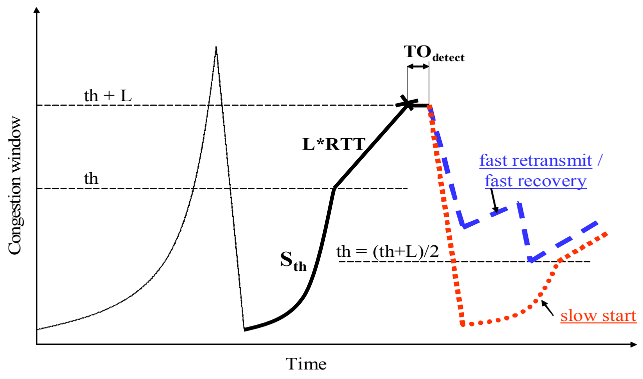

Figure 3. Packet loss during congestion avoidance Phase.

Thus, total time for the congestion avoidance phase, including the transmission time of the lost packet is

$$T_L = S_{th} + (x+1) \cdot MSS/R + L \cdot RTT + TO_{detect} \tag{15}$$

In Eqn. (15), $TO_{detect} = 3/2\ RTT$ corresponding to the reception of three duplicate ACKs for detecting a lost packet. By setting $N = N - x + 1$ and $th = \lceil (th + L)/2 \rceil$, we can obtain the rest of the values, such as T_x or T_L, by using the same method iteratively. It should be noted that S_x or S_{th} can be found for i=1,.., a+1, but HOB_i can be found for i=1,..,a.

C. Algorithm

Based on the model in the previous section, the complete procedure to find the object transfer time is given as follows.

ALGORITHM FOR OBJECT TRANSFER TIME

```
VARIABLE    $i$: count variable, $tot\_hob$: sum of $HOBi$
            $O_T$: total object transfer time
(1) Set $O_T = 0$ and $tot\_hob = 0$.
(2) Compute the total number of packets in object is
    $N = \lceil O/MSS \rceil$ and the expected number of packet loss
    $a = \lceil Np \rceil$.
(3) Set $i = 0$.
(4) Set $i = i + 1$.
(5) If $i = a+1$ or $p = 0$, Set $x = N$ and go to (7).
(6) Compute the expected number of packets sent
    before the loss
        $x = \frac{1-(1-p)^N}{p} + (1-p)^N$
(7) Compute the object transfer time for slow start
    If ($x+1 \le A_{th}$) {
      Compute slow-start time ($S_x$);
       If $a = 0$ or $i = a+1$ {
         Set $T_x = S_x + (x+1) \cdot MSS/R$;
         Set $O_T = O_T + T_x$}
       else { Compute $C$, $y$, $b$, and $HOB_i$;
         Set $T_x = S_x + (x+1) \cdot MSS/R + HOB_i + TO_{detect}$;
         Set $O_T = O_T + T_x$ ; Set $T = T/2$ ;
         Set $tot\_hob = tot\_hob + HOB_i$;
         Set $N = N - x + (y-b)$ }
    };
    If ($x+1 > A_{th}$) {
      Compute slow-start time ($S_{th}$), $L$ ;
       If $a = 0$ or $i = a+1$ {
         Set $T_L = S_{th} + (x+1) \cdot MSS/R + L \times RTT$;
         Set $O_T = O_T + T_L$ ;}
       else {
         Set $T_L = S_{th} + (x+1) \cdot MSS/R + L \cdot RTT + TO_{detect}$;
         Set $O_T = O_T + T_L$; Set  $th = \lceil (th + L)/2 \rceil$;
         Set $tot\_hob = tot\_hob + HOB_i$;
         Set $N = N - x + 1$; }
     };

(8) Set $i = i+1$.
(9) If $i = a+1$ or $p = 0$, Set $x = N$ and go to (11).

(10) Compute the expected number of packets sent before
      the loss
        $x = \frac{1-(1-p)^N}{p} + (1-p)^N$
(11) Compute the object transfer time after initial loss.
      Compute $C$, $y$, $b$, and $HOB_i$
       If $a = 0$ or $i = a+1$ {
         Set $T_L = (x+1) \cdot MSS/R + L.RTT$ ;
         Set $O_T = O_T + T_L$ ; Go to (12). }
       else {
         Compute $L$, $HOB_i$;
         Set $T_L = (x+1) \cdot MSS/R + L \cdot RTT + HOB_i + TO_{detect}$ ;
         Set $O_T = O_T + T_L$; Set  $th = \lceil (th + L)/2 \rceil$;
         Set $tot\_hob = tot\_hob + HOB_i$;
         Set $N = N - x + 1$; }
     };

(12) If $i = a+1$, then proceed to (13). Otherwise, go to
     (11).
(13) Find the total object transfer time ($O_T$).
```

When the number of packets is N, time complexity of our algorithm is $O(N)$. Based on our model, Table 1 shows the response time of HTTP over TCP and SCTP, respectively.

TABLE I:
Comparison of response time

Protocol	Total Response Time
HTTP over TCP	$I_T+RTT+O_T$, if $PI \geq M$ $I_T + M/PI * (RTT+O_T)$, Otherwise *(PI–Pipeline Index (depth of the pipeline))*
HTTP over SCTP	$I_T + RTT+(O_T - tot\_hob)$

III. PERFORMANCE EVALUATION

A. Experimental Setup

In this section, we validate our model presented in Sec. II. Our objective is to compare the results obtained analytically to those obtained experimentally. Since the current HTTP servers do not yet support the new SCTP protocol, we simulated the web server and client by transferring objects between two machines using TCP and SCTP, but emulating the HTTP protocol. (We didn't use TCP based HTTP server, because we want to make a fair comparison between TCP and SCTP using the same experimental setup). Since the model developed in this paper is concerned with object transfer time between a web server and a user, our simulated experiment consisting of just transferring objects to find object transfer time, is appropriate for validation of our model.

Figure 4 shows our experimental setup. Dell desktops have been used as servers and clients to transfer data between machines. NIST emulator [9] was used to simulate various network conditions, such as packet loss, bandwidth, RTT, etc., between the server and client.

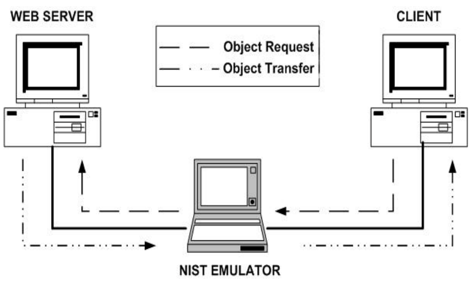

Figure 4. Experimental Setup.

The various object transfer times were calculated from packets captures by Ethereal [10] during transfer of objects between the client and the server for different network conditions. From the the experiment setup, the mean transfer time of HTTP over TCP and HTTP over SCTP were obtained. The collected results are compared and analyzed in the next section.

B. Results Analysis

Table II shows the mean response time (sec), for the proposed model and for the experiment as a function of the packet loss ratio (p) for RTT = 256 ms, O = 13.5 KB, R = 40 Kbps, MSS = 536 B, and M = 5. As p decreases, head-of-line blocking becomes very clsoe to zero. This is due to the fact that head-of-line blocking occurrs only in the case of packet loss. Because the size of object for HTTP application is very small, head-of-line blocking time of the HTTP application is very small. Accordingly, the head-of-line blocking time becomes very small as the transmission rate of the link (R) increases, thus overcoming any gain of HTTP over SCTP (when compared against HTTP over TCP). Nevertheless, since HTTP over SCTP can be supported over only one association, it has the advantages of requiring less resources than HTTP over TCP.

TABLE II
Result for varying packet loss ratio (p)

p / **Protocol**	**0.4 %**	**1%**	**2 %**	**5 %**
TCP (Model) (Sec)	21.27	21.33	21.77	25.84
TCP (Experimental) (Sec)	18.70	18.73	19.00	21.77
SCTP (Model) (Sec)	20.76	20.81	21.26	24.86
SCTP (Experimental) (Sec)	18.62	18.64	18.85	19.56

Table III shows the total response time according to RTT when p = 1 %, O = 13.5 KB, R = 40 Kbps, MSS = 536 B, and M = 5. Table 3 shows that the mean response time increases sharply with an increase of RTT.

TABLE III
Result for varying RTT

RTT(Sec) / **Protocol**	**0.055**	**0.08**	**0.256**	**1.0**
TCP (Model) (Sec)	14.89	14.94	18.73	21.56
TCP (Experimental) (Sec)	13.82	14.63	21.84	28.17
SCTP (Model) (Sec)	14.83	14.92	18.64	19.40
SCTP (Experimental) (Sec)	13.71	14.71	21.33	26.17

Table IV shows the total response time as a function of R for p = 1 %, O = 13.5 KB, RTT= 0.256 sec, MSS = 536 B, and M = 5. In Table 5, as R increases, meanresponse times for persistent connection with pipelining of HTTP/1.1 and HTTP over SCTP are almost same

This implies that the head-of-line (HOB) blocking time is close to zero. This can be explained by the fact that the value of *MSS*/*R* to affect the HOB decreases as *R* increases.

TABLE IV
Result for varying bandwidth (R)

R / **Protocol**	**40 Kbps**	**400 Kbps**	**3 Mbps**	**10 Mbps**
TCP (Model) (Sec)	18.73	5.42	4.20	2.59
TCP (Experimental) (Sec)	21.84	5.60	4.50	4.40
SCTP (Model) (Sec)	18.64	4.92	3.79	2.59
SCTP (Experimental) (Sec)	21.33	5.09	3.99	3.89

The small differences between the results from model and those from experimental were due to the inaccuracies of the NIST emulator we used in our experiment, to simulate the packet loss ratio, bandwidth, RTT, etc for small objects. We transferred small objects to reflect the real world scenario of HTTP transfers.

IV. CONCLUSIONS

We investigated the mean response time of HTTP file transfers over SCTP that had not been considered in the literature. The multi-streaming feature of SCTP allows a web server to simultaneously send several objects in an SCTP association without degrading the server performance as observed with pipelining and parallel connections for HTTP over TCP. We derived an analytical model for the mean response time for HTTP over SCTP. Previous traffic models for HTTP over TCP only assumed packet losses during the slow start phase; our model also considers the loss during the congestion avoidance phase. Furthermore, it estimates the head-of-line blocking time that is not included in the response time of HTTP over SCTP. Our results show that the mean end user response time for HTTP over SCTP is better than HTTP over TCP.

REFERENCES

[1] G. Camarllo, R. Kantola and H. Schulzrinne, “Evaluation of Transport Protocols for the Session Initiation Protocol”, *IEEE Network*, Vol. 17, No. 5, pp. 40-46, 2003.

[2] N. Cardwell, S. Savage and T. Anderson, “Modeling TCP Latency”, *IEEE Infocom*, Tel Aviv, Vol. 3, pp. 1742-1751, March 2000.

[3] A. L. Caro, J. R. Iyengar, P. D. Amer, S. Ladha, G. Heinz and K. Shah, “SCTP: A Proposed Standard for Robust Internet Data Transport”, *IEEE Computer*, Vol. 36, No. 11, pp. 56-63, November 2003.

[4] S. Fu and M. Atiquzzaman, “SCTP: State of the art in Research, Products, and Technical Challenges”, *IEEE Communication Magazine*, pp. 64-76, 2004.

[5] Z. Jiong, Z. Shu-jing and Qi-gang, “An Adapted Full Model for TCP Latency”, *Proceedings of IEEE TENCON ’02*, Beijing, Vol. 2, pp. 801-804, October 2002.

[6] S. Ladha and P. Amer, “Improving Multiple File Transfer Using SCTP Multistreaming”, University of Delaware, TR, 2003.

[7] J. Padhye, V. Firoiu, D. F. Towsley and J. F. Kurose, “Modeling TCP Reno Performance: A Simple Model and Its Empirical Validation”, *ACM Transactions on Networking*, Vol. 8, No. 2, pp. 133-145, 2000.

[8] R. Stewart, Q. Xie, et al, Stream Control Transmission Protocol, RFC 2960, 2000.

[9] http://snad.ncsl.nist.gov/itg/nistnet/

[10] www.ethereal.com