PAPER

Data Transfer Time by HTTP 1.0/1.1 on Asymmetric Networks Composed of Satellite and Terrestrial Links

Hiroyasu OBATA[†*], ***Student Member***, **Kenji ISHIDA**[††], **Junichi FUNASAKA**[††], ***and*** **Kitsutaro AMANO**[††], ***Regular Members***

SUMMARY Asymmetric networks, which provide asymmetric bandwidth or delay for upstream and downstream transfer, have recently gained much attention since they support popular applications such as the World Wide Web (WWW). HTTP (Hypertext Transfer Protocol) is the basis of most WWW services so, evaluating the performance of HTTP on asymmetric networks is increasingly important, particularly real-world networks. However, the performance of HTTP on the asymmetric networks composed of satellite and terrestrial links has not sufficiently evaluated. This paper proposes new formulas to evaluate the performance of both HTTP1.0 and HTTP1.1 on asymmetric networks. Using these formulas, we calculate the time taken to transfer web data by HTTP1.0/1.1. The calculation results are compared to the results of an existing theoretical formula and experimental results gained from a system that combines a VSAT (Very Small Aperture Terminal) satellite communication system for satellite links (downstream) and the Internet for terrestrial links (upstream). The comparison shows that the proposed formulas yield more accurate results (compared to the measured values) than the existing formula. Furthermore, this paper proposes an evaluation formula for pipelined HTTP1.1, and shows that the values output by the proposed formula agree with those obtained by experiments (on the VSAT system) and simulations.
key words: *web data transfer time, asymmetric networks, satellite links, terrestrial links, pipeline*

1. Introduction

The popularity of the Internet is reflected in the rise in client access rates to the Internet. Since such access is essentially asymmetric, customer satisfaction can be enhanced by using asymmetric networks, which provide asymmetric bandwidth or delay to upstream and downstream transfer. This is supported by the emergence of sophisticated network technologies. Examples include ADSL (Asymmetric Digital Subscriber Line) [1] and cable modems. One exciting approach is to combine satellite and terrestrial links. To this end, several techniques, such as the system using UDLR (Uni-Directional Link Routing) [2], [3], NTT system [4], [5], and the DirecPC system [6] have been developed. These services allow users to take advantage of asymmetric networks by sending requests for web contents via a terrestrial (land) line such as a modem, and receiving that information via a high speed, error-resilient satellite link.

Here, we provide a brief classification of asymmetric networks, because several forms are known [7].

- Bandwidth asymmetry: Typically, the downstream bandwidth is 10–1000 times the upstream bandwidth. Examples include cable modem, ADSL, and satellite-based networks.
- Media access asymmetry: This manifests itself in several ways. Example includes cellular wireless networks.
- Loss rate asymmetry: The network may inherently be more lossy in one direction than in the other.
- Propagation delay asymmetry: Propagation delay of the network is larger in one direction than in the other. Example includes satellite-based networks.

This article focuses on satellite–terrestrial networks that are characterized by both bandwidth and propagation delay asymmetry.

Asymmetric networks are most often employed by Internet users to access the WWW. Since HTTP (Hypertext Transfer Protocol) is a key technology of the WWW, it is essential to be able to predict the performance of HTTP on asymmetric networks.

HTTP is an application-level protocol for distributed, collaborative, hypermedia information systems [8], [9]. In the last few years, several articles have studied HTTP performance, but all provide only experimental evaluations [10], [11]. In a recent paper [12], the authors proposed an excellent formula for asymmetric networks. However, since the connection setup phase was not examined in detail, the formula may be inaccurate if the propagation delay is asymmetric. The authors turned their attention to the asymmetric networks that have asymmetric bandwidth, such as ADSL. Furthermore, with regard to HTTP1.1, the authors did not analyze the behavior of the congestion window in detail. Thus, the formula may inaccurate when calculating the time taken to transfer web data via HTTP1.1.

The performance of HTTP over asymmetric networks composed of satellite and terrestrial links has not been theoretically evaluated in sufficient detail. Note that the term HTTP covers both HTTP1.0 [8] and

Manuscript received January 4, 2002.
Manuscript revised April 26, 2002.

[†]The author is with the Graduate School of Information Sciences, Hiroshima City University, Hiroshima-shi, 731-3194 Japan.

[††]The authors are with the Department of Computer Engineering, Faculty of Information Sciences, Hiroshima City University, Hiroshima-shi, 731-3194 Japan.

[*]Presently, with KDDI Corp.

HTTP1.1 [9] and thus this paper proposes new performance evaluation formulas for both HTTP1.0 and HTTP1.1. Using these new evaluation formulas, we calculate the data transfer time of HTTP1.0/1.1 over asymmetric networks. The calculation results are compared to the results of an existing theoretical formula and experiments conducted using a VSAT (Very Small Aperture Terminal) satellite communication system for satellite links (downstream) and the Internet for terrestrial links (upstream). The results show that the proposed formulas better mirror the real-world performance than the existing formula.

HTTP1.1 shortens the data transmission time compared to HTTP1.0 since it doesn't force the client to set up a new TCP connection for each data request. This technique is also called "persistent connections" [9]. A persistent connection allows multiple requests to be sent without waiting for the server's response. This is useful in minimizing the total round trip delay and the number of packets, and improving performance. This technique is called "pipelining" [9], [10]. Unfortunately, current web browsers do not offer pipelining, and the performance of pipelined HTTP1.1 on asymmetric networks composed of satellite and terrestrial links has not been well investigated. Therefore, this paper also proposes a formula for pipelined HTTP1.1, and shows that the values calculated by the proposed formula agree with those obtained by conducted on a VSAT system and simulations.

The rest of this paper is organized as follows. Section 2 provides some background to this paper. Next, Sect. 3 introduces new formulas for asymmetric networks that can accurately estimate the performance (transfer time) of HTTP1.0/1.1. Numerical results and a discussion are then presented in Sect. 4. Section 5 shows the evaluation results of pipelined HTTP1.1. Finally, Sect. 6 concludes this paper.

2. Preliminary

In this section, we show the network model, and discuss related works.

2.1 Network Model

The network model used in our analysis and evaluation is depicted in Fig. 1.

The model consists of a server, a client, and two links (see Fig. 1); the downstream (satellite) link from the server to the client and the upstream (terrestrial) link† from the client to the server. The links have asymmetric bandwidth and delay. Bandwidths of the downstream and the upstream links are denoted as Q_{sat} and Q_{ter}, respectively. Propagation delays of the links are denoted as τ_{sat} and τ_{ter}, respectively. We assume that (web) data to be transmitted consists of n data sets. A data set is composed of some segment sets, and one segment set is made up of several data segments (packets). Data segment size and ack segment size are represented by d_{seg} and d_{ack}, respectively. d_{ack} is taken as the sum of TCP and IP header size. Next, d_{seg} represents the maximum segment size (MSS), which is calculated by subtracting the sum of TCP and IP header size from the maximum transmission unit (MTU) size.

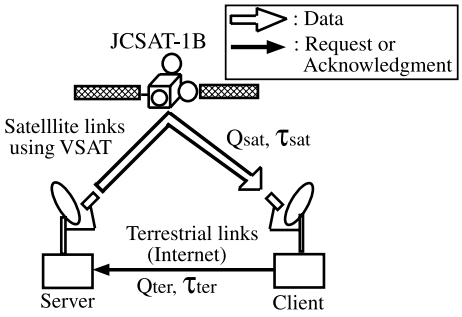


Fig. 1 Asymmetric network composed of a VSAT satellite communication system and the Internet.

2.2 Related Works

Several researchers have studied the performance of HTTP [10], [11]. However, these articles only experimentally evaluate HTTP performance.

In [12], analytical results are presented for asymmetric networks. We briefly explain the formula used below to model the transfer of data by TCP. In this formula, the transfer time of data is the sum of the connection setup time T_{setup} and the transfer time $T_{transfer}$ [12]. Here, the connection setup time T_{setup} is,

$$T_{setup} = \begin{cases} \frac{3}{2}rtt, (\text{each transfer in HTTP1.0} \\ \quad \text{and the first transfer in HTTP1.1}) \\ \frac{1}{2}rtt, (\text{the second and subsequent} \\ \quad \text{transfers in HTTP1.1}) \end{cases}$$

where, rtt is the round trip time of the connection. Finally, the total of data transfer time T_{total} is given as $T_{total} = T_{transfer} + T_{setup}$.

The authors calculated the connection setup time T_{setup} by a simple equation. Since propagation delay was assumed to be symmetrical in the numerical examples, this formula may be unable to accurately calculate the connection setup time when the propagation delay is asymmetric. Furthermore, this formula doesn't completely reflect the behavior of the congestion window when the following data flows are transferred by TCP in HTTP1.1. That is, this formula included a Slow Start phase for second and subsequent data flows. However, from experiments we found that the congestion window doesn't enter the Slow Start phase for the second and later data flows. Accordingly, this formula inaccurately

†Since the upstream is not the private line but the Internet, there is a possibility of occurring packet losses in the upstream. The discussion of this point is given in Sect. 3.1.

models HTTP1.1 performance.

3. Proposed Formulas

In order to overcome the above problem we assessed the connection setup time in detail. We create new formulas by extending the TCP formula given in [13], [14]. In this paper, we define HTTP performance as the time taken from issue of the TCP connection request from the client to the completion of receiving all data from the corresponding web server. We describe a new formula for HTTP1.0 in Sect. 3.1. A new formula for HTTP1.1 is then presented in Sect. 3.2.

3.1 Modeling HTTP1.0

Figure 2(a) shows a typical data transfer by HTTP1.0. For clarity, the TCP acks after connection establishment have been eliminated. When the client requests data from the server, a new TCP connection is established between the client and the server. Next, the client sends an HTTP request to the server, who then begins to transfer the data using TCP. When the data transfer is complete, the TCP connection is immediately closed [8], [12]. This means that the original formula can be extended to cover HTTP1.0 by adding the connection setup time to the time taken to send the data by TCP. The following paper ignores server overhead because experimental results showed that it is negligible compared to the network events.

Subsection (I) calculates the connection setup time while (II) introduces the equations that yield the data transmission time by TCP. Finally, the new formula for HTTP1.0 is presented in (III).

(I) The Connection Setup Time

The connection setup time $T_{connect}$ is the sum of the time required to transmit the packets (segments) and the propagation delay on the downstream and upstream links (see Fig. 2(a)). Thus, $T_{connect}$ is given by Eq. (1), where the packets that are transmitted for connection setup and for requesting data have the size of $P_{connect}$.

$$T_{connect} = \frac{P_{connect}}{Q_{sat}} + \frac{2P_{connect}}{Q_{ter}} + \tau_{sat} + 2\tau_{ter} \quad (1)$$

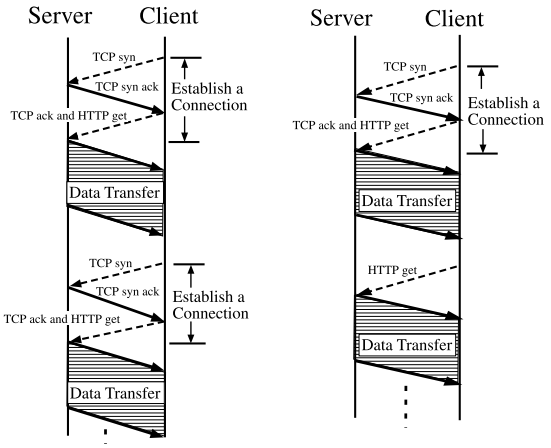
(II) Data Transmission Time by TCP

The data transmission time by TCP can be obtained using the formula in [13], [14]. In this paper, TCP version that we considered is Reno†.

We start by briefly explaining the formula in [13], [14]. In TCP, the transmission of data is based on window control; window size can vary during the session as shown in Fig. 3. Window size determines the size of each segment set which is transmitted by TCP at a time. Thus, we introduced a transmission period, i.e. the period of time between a transmission and the next transmission, is a function of the number of transmissions. Therefore, the total transmission time is the sum of the times taken to send all segment sets. Note that the formula considers the Slow Start phase and the Maximum Window Size phase (see Fig. 3).

Moreover, it is assumed that retransmission of data does not occur, meaning that there is enough buffering in the client, and no packet losses occur in both satellite links and terrestrial links. We explain this assumption for the satellite and terrestrial links. For the satellite link, comprehensive experiments discovered that packet loss is rare [13]–[15]. On the other hand, since the request and ack packets are sent over the terrestrial links, there is a possibility of their losses. However, their loss rates are insignificant because they are very small and the terrestrial link has practically sufficient bandwidth for these packets.

To begin with, the window size at the k-th transmission $w(k)$ is given by Eq. (2), where $w(0)$ equals 1. In Eq. (2), MWS denotes the maximum window size.



(a) The case of HTTP 1.0 (b) The case of HTTP 1.1

Fig. 2 Data transfer by HTTP1.0/1.1.

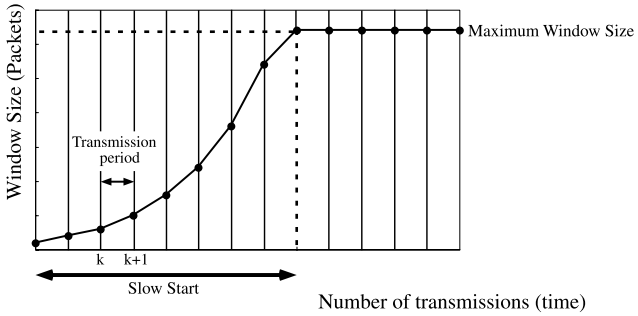


Fig. 3 Assumed change in $w(k)$.

†Our experiments involved machines running TCP Reno, while [12] considered TCP Tahoe. Therefore, we modified the equation in [12] to reflect the different characteristic of TCP Reno. That is, when packet loss occurs, the window size is halved instead of being set to zero.

$$w(k+1) = min\left(w(k) + \left\lceil \frac{w(k)}{2} \right\rceil,\ MWS\right) \quad (2)$$

$w(k)$ is assumed to follow the curve shown in Fig. 3.

Next, transmission period $t(k)$ is given by Eq. (3). rtt, $a(k)$, and $q(k-1)$ denote the round trip time, transmission time required to send segment set of size $w(k)$, and transmission time required to send ack segment corresponding to the $(k-1)$-th transmission, respectively.

$$t(k) = max(rtt, a(k), q(k-1)) \quad (3)$$

rtt, $a(k)$ and $q(k)$ are given by Eqs. (4), (5) and (6), respectively. In Eqs. (4) and (5), segment size of data P is the sum of payload data size and both TCP and IP headers.

$$rtt = \tau_{sat} + \tau_{ter} + \frac{2 \cdot P}{Q_{sat}} + \frac{d_{ack}}{Q_{ter}} \quad (4)$$

$$a(k) = \frac{P \cdot w(k)}{Q_{sat}} \quad (5)$$

$$q(k) = \frac{d_{ack} \cdot \lceil \frac{w(k)}{2} \rceil}{Q_{ter}} \quad (6)$$

Next, the threshold th which satisfies Eq. (7) is obtained. In Eq. (7), F denotes the data set size.

$$\sum_{k=0}^{th} \{w(k) \cdot d_{seg}\} \leqq F < \sum_{k=0}^{th+1} \{w(k) \cdot d_{seg}\} \quad (7)$$

Thus the time, T, required for transmitting a data set of size F via TCP (HTTP1.0) is given by Eq. (8).

$$T = \sum_{k=0}^{th} t(k) + \frac{F - \sum_{k=0}^{th} \{w(k) \cdot d_{seg}\} + d_{ack}}{Q_{sat}} + \tau_{sat} \quad (8)$$

(III) Multiple Data Set Transmission Time by HTTP1.0

Finally, the time $T_{http1.0}$ required for transmitting n data sets by HTTP1.0 is the sum of the time required for the data transmission by TCP ($T_{transmit}(k)$) and the connection setup time ($T_{connect}$), $T_{http1.0}$ is given by Eq. (9) which is derived from Eq. (1) and Eq. (8). Here, $T_{transmit}(k)$ is defined as the transmission time of the k-th data set.

$$T_{http1.0} = nT_{connect} + \sum_{k=1}^{n} T_{transmit}(k) \quad (9)$$

3.2 New Formula for HTTP1.1

Figure 2(b) shows a typical data transfer by HTTP1.1. The first request is handled in the same way as HTTP1.0. When the client requests a new data set from the same server, the client doesn't have to establish a new TCP connection. It is because, in HTTP1.1, the client retains information about the previously established TCP connection. Therefore, the client can send an HTTP request without negotiation [9], [12]. It follows that the transfer of the first data set follows Eq. (9) while the subsequent data sets are transferred more rapidly. Subsection (I) introduces a formula for these subsequent transfers. Subsection (II) introduces a formula that estimates the total transfer time of all data sets.

(I) The Time for the Second and Later Request of Web Data

The time taken to transfer the second and later requested data sets $T'_{connect}$ is the sum of the time required to transmit one packet and the propagation delay of the upstream link (see Fig. 2(b)). Thus, $T'_{connect}$ is given by Eq. (10).

$$T'_{connect} = \frac{P_{connect}}{Q_{ter}} + \tau_{ter} \quad (10)$$

(II) Transmission Time of All Web Data by HTTP1.1

When the client requests n data sets, the time, $T_{http1.1}$, taken by HTTP1.1 to transfer all sets is the sum of the time required data transfer by TCP ($T_{transmit}(k)$) and the time for the second and later request of web data ($T'_{connect}$), and the connection setup time ($T_{connect}$). Thus, $T_{http1.1}$ is described as Eq. (11) from Eq. (1), Eq. (8), and Eq. (10).

$$T_{http1.1} = T_{connect} + (n-1)T'_{connect} + \sum_{k=1}^{n} T_{transmit}(k) \quad (11)$$

Hereafter, Eqs. (9) and (11) are referred to as "the initial (1.0) formula," and "the initial (1.1) formula," respectively.

4. Numerical Evaluation

This section, applies the initial (1.0) and (1.1) formulas to an asymmetric network, the calculation results are compared to experimental data [16].

4.1 Parameters of the Asymmetric Network

Table 1 lists the parameters of the asymmetric network considered below. The asymmetric network is composed of a VSAT satellite communication system and the Internet [17] (see Fig. 1).

4.2 Evaluation

In this subsection, the values yielded by the initial formulas are compared with the experimental data.

Table 1 Network parameters.

Data Size (kbyte)	5,10,50,100,500
Data Segment Size (byte)	1448
Ack Segment Size (byte)	52
Downstream Bandwidth (kbps)	2048
Upstream Bandwidth (kbps)	900.7
Downstream Delay (msec)	258.6
Upstream Delay (msec)	17.3
$P_{connect}$ (byte)	52

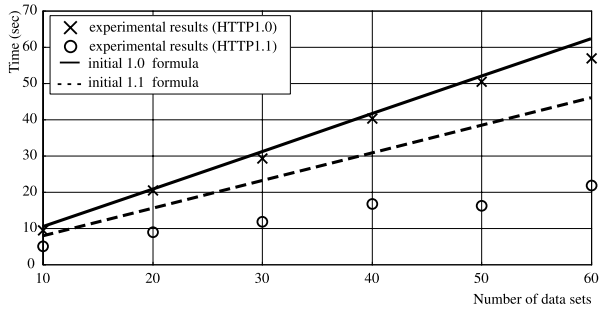


Fig. 4 Comparison of the transfer time of web data by the initial formulas and the experimental results, with various number of data sets.

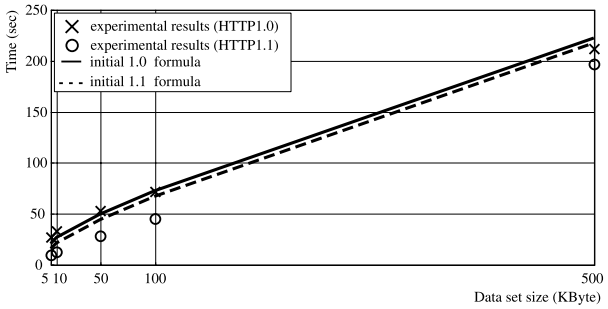


Fig. 5 Comparison of the transfer time by the initial formulas and the experimental results, for data sets of various sizes.

In the experiments, we used Apache_1.3.14 [18] as the web server and libwww robot (release 5.2.6) from W3 Consortium [19] as the client. We set up a sample web page that offered multiple images, and averaged five results obtained by downloading the same data sets following the method described in [10], [11].

Figure 4 compares the transfer time as determined from by the initial formulas and the experimental results; the data set size was 5 kbyte and the number of data sets per web page was set at 10, 20, 30, 40 50, and 60. Figure 5 indicates that similar results were recorded when the number of data sets per web page was 20 with data sizes of 5 kbytes, 10 kbytes, 50 kbytes, 100 kbytes, and 500 kbytes. In these figures, the symbols represent the experimental results and the lines plot the values predicted by the initial formulas.

4.3 Discussion

To begin with, we discuss the accuracy of the initial formulas. From Fig. 4 and Fig. 5, it is observed that the values derived from the initial (1.0) formula are consistent with the experimental results. This is not true for the initial (1.1) formula. Since the initial (1.1) formula overestimates the transfer time, we can surmise that the Slow Start phase of TCP for the second and subsequent transfers included in the initial (1.1) formula is unnecessary. To confirm this assumption, we investigated the sequence number of TCP transfer and determined whether the congestion window shrank or not. Figure 6 shows the data sequence number (taken from Fig. 5) as observed at the server; data set size is 10 kbytes. These results were obtained from the output of the *tcpdump* command in the server (see Fig. 1).

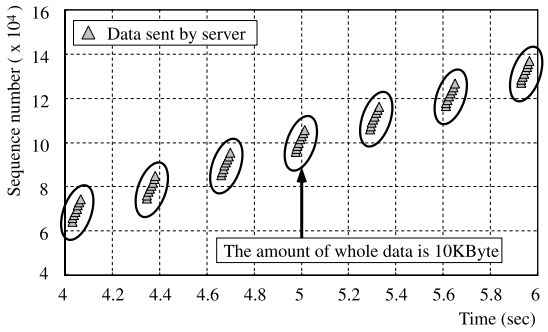


Fig. 6 Change in sequence number.

In Fig. 6, the amount of whole data which are enclosed with the circle means one data set. From Fig. 6, it is clear that multiple packets are transmitted at the same time even when starting the next web data transfer. Thus it seems reasonable to suppose that the sender's congestion window stays large, i.e. Slow Start phase is not used. This suggests that the formula of [13], [14] (Eq. (8) in this paper) should be modified. Here note that in Fig. 6, the data request interval is about 0.28 second. Based on side experiments, it appears that the congestion window doesn't enter the Slow Start phase even if the data request interval is 1 second in the environment. However, if the data request interval is very large (for example, 10 second), the server may begin Slow Start. In this case, it is not necessary to adapt the following modification.

We now introduce $(T'_{transmit}(k))$ as the transfer time of the k-th data set, whose size equals F_k, by TCP where $k > 1$. It can be determined by removing the Slow Start phase from Eq. (8).

$$T'_{transmit}(k) = \left\lfloor \frac{F_k}{MWS} \right\rfloor \cdot t_{max} + \frac{(F_k) \bmod (MWS) + d_{ack}}{Q_{sat}} + \tau_{sat} \qquad (12)$$

In Eq. (12), t_{max} denotes the transmission period in maximum window size phase that is calculated by Eq. (3) using $w(k) = MWS/d_{seg}$.

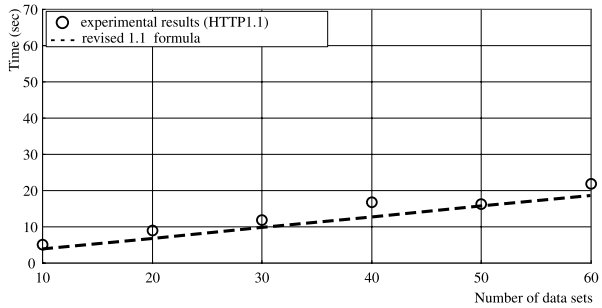


Fig. 7 Performance of revised (1.1) formula, with various numbers of data sets.

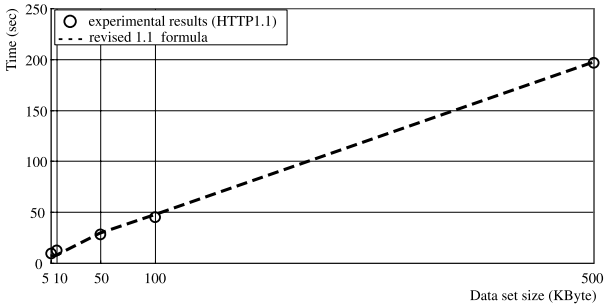


Fig. 8 Performance of revised (1.1) formula, with various data sizes.

Finally, the new equation for $T_{http1.1}$ (Eq. (13)) is derived from Eq. (11).

$$T_{http1.1} = T_{connect} + (n-1)T'_{connect} + T_{transmit}(1) + \sum_{k=2}^{n} T'_{transmit}(k) \quad (13)$$

We refer to Eq. (13) as "the revised (1.1) formula."

Next, we compare the values derived from the revised (1.1) formula to experimental results for various numbers of data sets in Fig. 7. Figure 8 presents similar results for various data set sizes. From Fig. 7 and Fig. 8, it is obvious that the revised (1.1) formula better mirrors the experimental results, which confirms that our understanding is correct.

Next, we compare the values yielded by the proposed formulas to those yielded by the formula in [12]. Figure 9 plots the transfer time using HTTP1.0, with various numbers of data sets. Figure 10 plots a similar comparison for HTTP1.1. It is obvious from Fig. 9 and Fig. 10 that the proposed formulas are more accurate than the existing formula. Now, we discuss the reason for the superiority of the proposed formulas. Figure 9 illustrates the performance of HTTP1.0. The vertical axis represents the sum of the connection setup time and the data transmission time. For the connection setup time, the formula in [12] doesn't consider asymmetric propagation delay. On the other hand, we analyzed the connection setup time in detail (See Eq. (1) and Eq. (10)). With regard to the data transmission

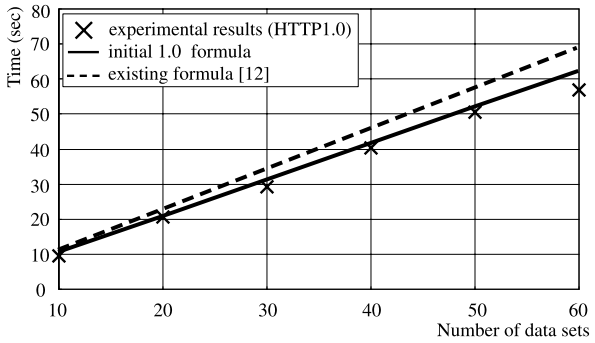


Fig. 9 HTTP1.0 comparison.

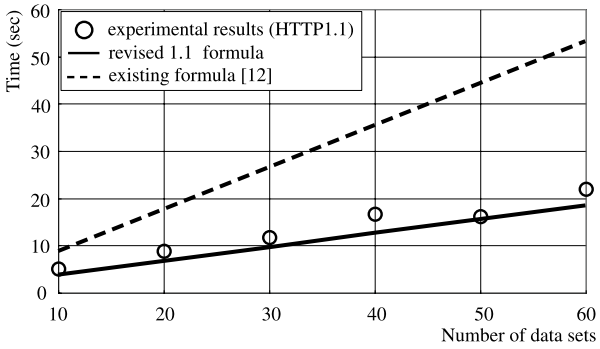


Fig. 10 Performance of revised 1.1 formula.

time, the difference between the initial 1.0 formula and the formula in [12] is small. Therefore, for HTTP 1.0, the difference in total data transmission time is mainly caused by the connection setup time. Here, the difference in connection setup time between the formula in [12] (T_{setup} is about 0.41 seconds) and our proposed formula ($T_{connect}$ is about 0.29 seconds) is about 0.12 seconds. This means that the difference in data transmission time between the formula in [12] and the initial 1.0 formula is about 1.2 seconds for 10 data sets and about 7.2 seconds for 60 data sets. This confirms that the initial 1.0 formula is more accurate than the formula in [12].

Figure 10 shows the comparable results for the revised 1.1 formula. With regard to the connection setup time, the difference between the revised 1.1 formula and the formula in [12] is relatively small. There is a large difference in terms of the data transmission time. The formula in [12] doesn't reflect the behavior of the congestion window when data sets are transmitted continuously. That is, the formula in [12] assumes that the second and later data sets are transmitted including the Slow Start phase. The proposed formula, on the other hand, considers the behavior of the congestion window in detail (see Eq. (12) and Eq. (13)).

As a result, the values from the proposed formulas better match the experimental results than the formula in [12].

The revised 1.1 formula does not consider pipelin-

ing [9] because this technique is not offered by current web browsers, however, it does appear to improve the performance of HTTP.

5. New Formula for HTTP1.1 with Pipeline

In this section, we show a formula for pipelined HTTP1.1 that is an extension of the revised 1.1 formula.

Figure 11 indicates data transfer by pipelined HTTP1.1. In the figure, the server transmits the data sets in the page requested by the client. Next, the client sends multiple requests without waiting for each response. Finally, the server sends its responses to those requests in the same order in which the requests were received [9].

Therefore, the time $T_{http1.1-pipeline}$ required for transmitting all data sets by pipelined HTTP1.1 is the sum of the time required for transmission of the first data set by TCP ($T_{transmit}(1)$), the time required for the second and subsequent data sets on the first page ($T''_{transmit}$), the time for the second and subsequent data requests ($T'_{connect}$), and the connection setup time ($T_{connect}$). $T''_{transmit}$ is given by Eq. (12) where we treat data sets 2 to n, which have size F_k, as one data. This means that $T''_{transmit}$ is given by Eq. (14).

$$T''_{transmit} = \left\lfloor \frac{\sum_{k=2}^{n} F_k}{MWS} \right\rfloor \cdot t_{max} + \tau_{sat} + \frac{\left(\sum_{k=2}^{n} F_k \right) \bmod (MWS) + d_{ack}}{Q_{sat}} \quad (14)$$

Finally, $T_{http1.1-pipeline}$ is given by Eq. (15), which is derived from Eq. (1), Eq. (8), Eq. (10), and Eq. (14).

$$T_{http1.1-pipeline} = T_{connect} + T'_{connect} + T_{transmit}(1) + T''_{transmit} \quad (15)$$

We refer to Eq. (15) as "the pipeline formula."

Figure 12 depicts the transfer time achieved by pipelined HTTP1.1 for the data set size of 5 kbytes. In the figure, the symbols represent the results of simulation experiments and experimental results, and the lines plot the pipeline formula values. Here, we used OPNET Modeler/RADIO [20] to conduct the simulation experiments. From Fig. 12, it is obvious that the pipeline formula well predicts both the experimental results and the simulation results.

We found that the benefit of pipelining increases with the number of data sets (see Fig. 12). For example, pipelining reduces the transfer time by about 1.5 seconds for 10 data sets and about 20 seconds for 100 data sets. These results lead to the conclusion that pipelining is recommended when many data sets are to be transmitted continuously on asymmetric networks using satellite and terrestrial links.

6. Conclusion

This paper has studied the transfer time of data sets by HTTP1.0/1.1 over the asymmetric networks composed of satellite and terrestrial links. Equation (9) for HTTP1.0, Eq. (13) for HTTP1.1, and Eq. (15) for HTTP1.1 with pipelining are newly derived. These formulas may be applicable to other asymmetric networks that satisfy the following conditions; the client has sufficient buffer capacity and the downstream link is error–resilient.

We used experimental results to confirm that the proposed formulas can calculate the transfer time of web data more precisely than the one already proposed.

Acknowledgments

The authors thank Prof. Matsuichi Yamada of Tokyo Engineering University for providing us with data

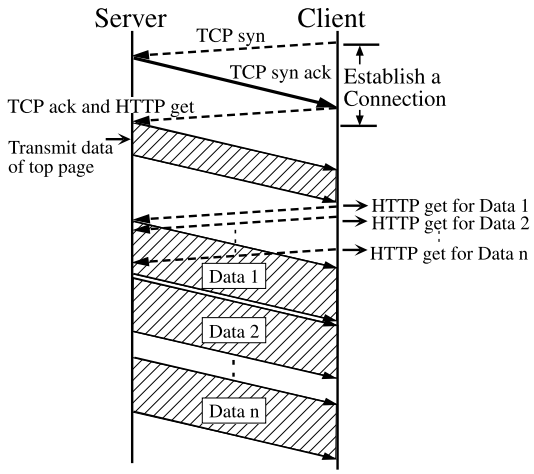


Fig. 11 HTTP1.1 with pipeline.

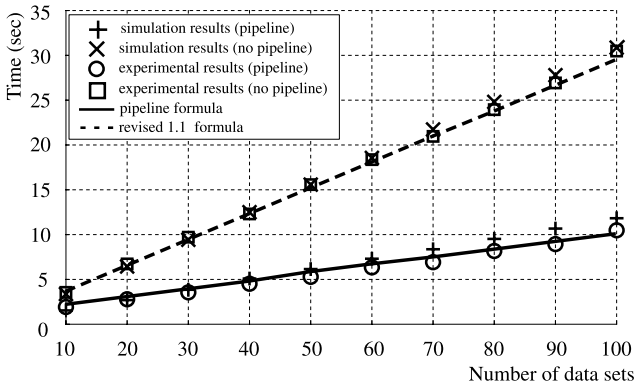


Fig. 12 The transfer time of data sets by pipelined HTTP1.1. (data set size is 5 kbytes)

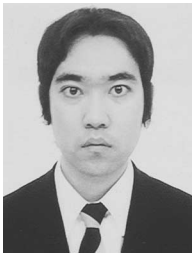
Table 2 Proposed formulas.

HTTP version	proposed formula	formula's number
HTTP1.0	$T_{http1.0} = nT_{connect} + \sum_{k=1}^{n} T_{transmit}(k)$	Formula (9)
HTTP1.1	$T_{http1.1} = T_{connect} + (n-1)T'_{connect} + T_{transmit}(1) + \sum_{k=2}^{n} T'_{transmit}(k)$	Formula (13)
$\text{HTTP1.1}_{\text{pipeline}}$	$T_{http1.1-pipeline} = T_{connect} + T'_{connect} + T_{transmit}(1) + T''_{transmit}$	Formula (15)

on the VSAT satellite communication system. This research was partially supported by the Ministry of Education, Culture, Sports, Science and Technology, Grant-in-Aid for Scientific Research (B), 10450152, and a Hiroshima City University Grant for Special Academic Research. The authors would like to thank the anonymous reviewers for their helpful comments and constructive suggestions.

References

[1] DSL Forum, "General introduction to copper access technologies," http://www.adsl.com/general_tutorial.html, April 2001.
[2] E. Duros, W. Dabbous, H. Izumiyama, N. Fujii, and Y. Zhang, "A link layer tunneling mechanism for unidirectional links," IETF RFC3077, March 2001.
[3] S. Fujieda, H. Watanabe, and H. Kusumoto, "Transparent routing in the network with unidirectional links using virtual broadcast links," IPSJ SIG Notes, 99-DPS-92, pp.49–54, Feb. 1999.
[4] M. Nakagawa, Y. Hashimoto, H. Nakashima, and T. Kon, "Performance evaluation and educational application of a multimedia interactive satellite communication system," NTT R&D, vol.47, pp.203–209, Feb. 1998.
[5] S. Miyata, T. Tanaka, and S. Kubota, "Routing management in asymmetric networks," NTT R&D, vol.47, pp.1049–1056, Oct. 1998.
[6] Hughes Network Systems, "DirecPC," http://www.direcpc.com/, April 2001.
[7] H. Balakrishnan and V. Padmanabhan, "How network asymmetry affects TCP," IEEE Commun. Mag., vol.39, no.4, pp.60–67, April 2001.
[8] T. Berners-Lee, R. Fielding, and H. Frystyk, "Hypertext transfer protocol – HTTP/1.0," IETF RFC1945, May 1996.
[9] R. Fielding, J. Gettys, J. Mogul, H. Frystyk, L. Masinter, P. Leach, and T. Berners-Lee, "Hypertext transfer protocol – HTTP/1.1," IETF RFC2616, June 1999.
[10] H. Nielsen, J. Gettys, A. Baird-Smith, E. Prudhommeaux, H. Lee, and C. Lilley, "Network performance effects of HTTP/1.1, CSS1, and PNG," Proc. SIGCOMM, vol.27, no.4, pp.155–166, Oct. 1997.
[11] S. Cheng, K. Lai, and M. Baker, "Analysis of HTTP1.1 performance on a wireless network," Technical Report CSL-TR-99-778, Stanford University, Feb. 1999.
[12] G. Hasegawa, M. Murata, and H. Miyahara, "Performance evaluation of HTTP/TCP on asymmetric networks," International Journal of Communication Systems, vol.12, no.4, pp.281–296, July 1999.
[13] H. Obata, K. Ishida, J. Funasaka, and K. Amano, "Evaluation of TCP performance on asymmetric networks using satellite and terrestrial links," IEICE Trans. Commun., vol.E84-B, no.6, pp.1480–1487, June 2001.
[14] H. Obata, K. Ishida, J. Funasaka, and K. Amano, "TCP performance analysis on asymmetric networks composed of satellite and terrestrial links," 8th International Conference on Network Protocols (ICNP2000), pp.199–206, Nov. 2000.
[15] Y. Nishida, O. Nakamura, H. Kusumoto, and J. Murai, "A broadcasting network architecture for satellite communication," IPSJ SIG Notes, 95-OS-69, pp.7–12, June 1995.
[16] H. Obata, K. Ishida, J. Funasaka, and K. Amano, "HTTP1.0/1.1 performance analysis over asymmetric networks using satellite and terrestrial links," IEICE Technical Report, IN2001-73, Sept. 2001.
[17] T. Inoue, H. Obata, K. Ishida, K. Amano, Y. Katsumi, and M. Yamada, "Experimental evaluation of TCP performance on asymmetric links using VSAT satellite communication system," IEICE, Network Architecture Workshop, pp.122–129, Feb. 2000.
[18] The Apache Software Foundation, "Apache," http://www.apache.org/, Dec. 2000.
[19] H. Nielsen, "Libwww robot," http://www.w3.org/, Dec. 2000.
[20] OPNET Technologies Inc., "OPNET moderler online documentation version 7.0B," July 2000.



Hiroyasu Obata received the B.E. and M.S. degrees in Information Sciences from Hiroshima City University, Japan, in 2000 and 2002, respectively. He is currently in the KDDI Corp. His interests include computer communications over wireless networks, such as satellite links.

Kenji Ishida received the B.E., M.Sc., and Ph.D. degrees from Hiroshima University, Japan, in 1984, 1986 and 1989, respectively. He was at Hiroshima Prefectural University from 1989 to 1997. He has been an Associate Professor in the Department of Computer Engineering, Faculty of Information Sciences, Hiroshima City University, since 1997. His interests include distributed computing systems and design of control procedures for computer networks. Dr. Ishida is a member of IEEE (U.S.A), ACM (U.S.A), IPSJ (Japan).

Junichi Funasaka received the B.S. and M.S. degrees from Tohoku University in 1993 and 1995, respectively. He also received the M.E. and Ph.D. degrees from Nara Institute of Science and Technology in 1997 and 1999, respectively. Since 1999 he has been a Research Associate in the Department of Computer Engineering, Faculty of Information Sciences, Hiroshima City University. His research interests include information processing in the Internet. Dr. Funasaka is a member of IEEE (U.S.A), IPSJ (Japan).

Kitsutaro Amano received the B.E. and Ph.D. degrees from Kyoto University, Japan, in 1955 and 1963, respectively. From 1955 to 1994, he was with Kokusai Denshin Denwa Co., Ltd. (KDD), where he last held the position of a Deputy Director of the Meguro Research and Development Laboratories of KDD. He has, since 1994, been a Professor in the Department of Computer Engineering, Faculty of Information Sciences, Hiroshima City University. His main fields of study are digital transmission systems and optical fiber undersea cable systems. Dr. Amano is a member of IEEE (U.S.A), IPSJ (Japan). He is a fellow of the IEEE.