Background TCP data transfer with Inline network measurement

Tomoaki Tsugawa, Go Hasegawa, and Masayuki Murata
Graduate School of Information Science and Technology, Osaka University
1–5 Yamadaoka, Suita, Osaka, 560–0871, Japan

Abstract—**In the present paper, ImTCP-bg, a new background TCP data transfer mechanism that uses an inline network measurement technique, is proposed. ImTCP-bg sets the upper limit of the congestion window size of the sender TCP based on the results of the inline network measurement, which measures the available bandwidth of the network path between the sender and receiver hosts. ImTCP-bg can provide background data transfer without affecting the foreground traffic, whereas previous methods cannot avoid network congestion essentially. ImTCP-bg also employs an enhanced RTT-based mechanism so that ImTCP-bg can detect and resolve network congestion, even when reliable measurement results cannot be obtained. The performance of ImTCP-bg is investigated through simulation experiments, and the effectiveness of ImTCP-bg in terms of the degree of interference with foreground traffic and the link bandwidth utilization is also investigated.**

I. Introduction

Due to the rapid development of networking technologies in both access and core computer networks, as well as the sudden increase of the Internet population, various IP-based network services are emerging and currently co-exist on the Internet. Although these services compete for network link bandwidth, Transmission Control Protocol (TCP) currently plays a major and important role for avoiding and solving network congestion collapse by using the congestion control algorithm [1] between the sender/receiver endhosts. Thus, TCP provides effective usage and fair sharing of network resources among competing data transmission flows.

However, some of the Internet services do not necessarily require fair resource allocation with respect to other flows and should be operated in the background through prioritization mechanisms. For example, in Content Delivery/Distribution Networks (CDNs) [2] such as Akamai [3], Web servers transfer various types of data (e.g. backup, caching [4], and prefetching [5]), in addition to the data in response to the document transfer request from Web clients. In this case, the user-requested data should be transferred with the higher priority than the other traffic. Data backup and synchronization in Storage Area Networks (SAN), online updating of operating systems (e.g. Microsoft's Background Intelligent Transfer Service [6]), and data caching in peer-to-peer network [7] are other examples of tasks that should be performed without affecting the foreground traffic.

In previous studies, such prioritized behaviors were realized by either IP-based mechanisms or application-based mechanisms. In IP-based mechanisms, such as DiffServ [8], the Internet routers are equipped with prioritization mechanisms and process the incoming packets according to pre-defined prioritization policies. For instance, Assured Forwarding in DiffServ has four classes and three dropping levels at the router buffer to differentiate the incoming flows. However, such mechanisms have well-known shortcomings in scalability, because the prioritization mechanisms should be implemented on all routers between the sender and receiver endhosts.

In application-based mechanisms, the prioritization mechanisms are provided by upper-level programs, or sometimes by service administrators. For example, cache synchronization and prefetching in CDNs is performed when there is little user-requested foreground traffic. Data backup is usually scheduled to be performed at midnight in order to avoid degrading the throughput of other higher-prioritized flows during the daytime. In such cases, the programs/administrators must monitor the network traffic to determine the time during which the network is underutilized. However, successfully realizing such mechanisms is difficult due to large fluctuations in Internet traffic.

Therefore, TCP-based approaches such as TCP Nice [9] and TCP-LP [10], which are herein referred to as background TCP, have been introduced in order to handle background (lower-prioritized) data transfer on the Internet. These approaches observe the Round Trip Times (RTTs) of the data packets of a TCP connection and decrease the congestion window size when the RTTs increase, whereas the original TCP Reno continues to increase its congestion window size until packet loss occurs, regardless of increases in RTTs. By this mechanism, background data transfer against the TCP Reno connections is achieved because in practical networks the RTTs increase before the packet loss occurs.

Although both TCP Nice and TCP-LP can realize data transfer without affecting the foreground (higher-prioritized) traffic, these protocols are unable to utilize the available bandwidth of the network efficiently. This is because the degree to which the congestion window size can decrease when the RTTs increase is fixed, and is too large, regardless of the network condition, similarly to TCP Reno which halves the window size when packet loss occurs.

In the present paper, a novel background TCP mechanism based on bandwidth measurement is proposed, with the goal of achieving both background transfer and network bandwidth utilization. The proposed background TCP variant uses the inline network measurement mechanism proposed in [11, 12], which can measure the available bandwidth of the network path between sender and receiver endhosts. This inline network measurement mechanism uses the data and ACK packets of a TCP connection for the measurement task, without injecting addi-

tional traffic, which is ideal for background data transfer. The proposed mechanism sets the maximum value of the congestion window size of the sender TCP by using the measurement results of the available bandwidth. In addition, an RTT-based mechanism that dynamically determines the degree to which the congestion window size can decrease according to the observed RTT value is employed, whereas TCP Nice and TCP-LP use a constant degree for the possible decrease.

II. Background data transfer with TCP

TCP adjusts the data transmission speed by changing the congestion window size in response to network congestion. The TCP algorithm allows a TCP sender to continue to increase its congestion window size additively until network congestion is detected. TCP decreases the window size multiplicatively when network congestion occurs. As a indicator of the network congestion, TCP Reno uses packet losses in the network (referred to herein as a loss-based mechanism). On the other hand, TCP Nice and TCP-LP introduce another congestion indicator, namely the increase of RTTs for data packets (RTT-based mechanism). These protocols provide background data transfer without affecting the foreground traffic based on the following assumption. Consider an output link of an Internet router equipped with an output buffer. When the packet incoming rate of the traffic destined for the output link is larger than the output link bandwidth, the excess traffic is stored in the output buffer, which causes some queuing delay, and eventually results in the packet losses when the buffer becomes full. That is, for a TCP connection, the RTTs usually increase before packet losses occurs when the network is congested. Therefore, TCP Nice and TCP-LP connections can detect network congestion earlier than TCP Reno connections.

The present paper considers the following two objectives for background data transfer: (1) unaffected foreground traffic and (2) fully utilization of the network link bandwidth. That is, a perfect background data transfer mechanism can fully utilize the bandwidth that is unused by the foreground traffic, while not degrading the performance of the foreground traffic. However, realizing such a complete mechanism is quite difficult because a trade-off relationship exists between these two objectives. The difficulty in realizing a good background data transfer mechanism lies in balancing this trade-off relationship. For example, TCP Nice and TCP-LP are unable to efficiently utilize the available bandwidth of the network path, especially when the number of background TCP connections is small [9, 10]. This is mainly because these protocols use fixed parameters in detecting network congestion and decreasing the congestion window size. That is, these two background TCP variants set the parameters by which to satisfy objective (1), while sacrificing objective (2). RTT-based mechanisms such as those of TCP Nice and TCP-LP encounter this trade-off problem due to their trial-and-error nature. These mechanisms continue to increase the window size when RTT becomes increased, and then decrease the window size to some degree.

In order to satisfy the above two objectives, another congestion indicator is proposed, i.e. the available bandwidth of the network path between the sender and receiver hosts (bandwidth-based mechanism). The available bandwidth is the most straightforward information by which to describe background data transfer. If the TCP sender obtains the available bandwidth information exactly and quickly, then an ideal background data transfer mechanism, in terms of both the background nature and link utilization can be created.

Many algorithms and tools by which to measure the available bandwidth of network paths have been proposed in the literature [13–17]. However, the existing methods cannot be directly employed for the newly proposed background TCP because these methods utilize numerous test probe packets and require too much time to obtain a single measurement result. In order to address this problem, a method referred to as Inline measurement TCP (ImTCP) has been proposed in [11, 12]. ImTCP does not inject extra traffic into the network, but rather estimates the available bandwidth of the network path from data and ACK packets transmitted by an active TCP connection in an inline fashion. Since the ImTCP sender obtains bandwidth information every 1–4 RTTs, ImTCP can follow the traffic fluctuation of the underlying IP network well. In addition, because the ImTCP mechanism is implemented at the bottom of the TCP layer, various types of TCP congestion control mechanisms can include this measurement mechanism. Therefore, the ImTCP mechanism is integrated into the proposed background TCP in order to obtain the available bandwidth information of the network path.

However, the RTT-based mechanism cannot be discarded even when the bandwidth-based mechanism is employed, because ImTCP does not always provide accurate measurement results for the available bandwidth. For example, when the congestion window size of the ImTCP sender is small, ImTCP cannot measure the available bandwidth [12]. Furthermore, the measurement accuracy of ImTCP depends on the network environment, e.g. the RTT, the physical link bandwidth, and the number of active connections. When the measured available bandwidth value is inaccurate, the background data transfer based on the measured value may affect the foreground traffic. Therefore, the RTT-based mechanism should be used in conjunction with the bandwidth-based mechanism.

III. ImTCP-bg mechanisms

ImTCP-bg consists of two major mechanisms: a bandwidth-based mechanism with inline network measurement and an enhanced RTT-based mechanism for adjusting the window size when the first mechanism does not work well.

A. Bandwidth-based mechanism

In ImTCP-bg, the congestion window size is controlled using the available bandwidth information of the network path between the sender and receiver hosts, as measured by the ImTCP mechanism. ImTCP-bg smoothes the measurement results of ImTCP using a simple exponential weighted moving average, as follows:

$$\bar{A} \leftarrow (1-\gamma) \times \bar{A} + \gamma \times A_{cur} \quad (1)$$

where A_{cur} denotes the current result of the available bandwidth measured by ImTCP mechanism, γ ($0 < \gamma < 1$) is a smoothing parameter, and $\bar{A}$ is the smoothed available bandwidth. The ImTCP-bg sender then sets the upper limit of the congestion window size ($maxcwnd$) using the following equation:

$$maxcwnd \leftarrow \bar{A} \times RTT_{min} \tag{2}$$

where RTT_{min} is the minimum RTT experienced throughout the lifetime of the connection.

As shown above, the proposed bandwidth-based mechanism is quite simple: the upper-limit of the congestion window size is simply set as the product of the measured available bandwidth and the minimum RTT. In Section 4, this simple mechanism is demonstrated to be effective for background data transfer.

B. Enhanced RTT-based mechanism

The effectiveness of the above-described bandwidth-based mechanism depends largely on the accuracy of the measurement by ImTCP of the available bandwidth. In [12] the authors demonstrated that ImTCP can give the reasonably accurate measurement results every 1–4 RTTs. However, ImTCP does not always provide reliable measurement results, as explained in Section 2, and may result in the congestion of the bottleneck link. Therefore, the RTT-based mechanism is employed to quickly detect and resolve the undesirable network congestion. The RTT-based mechanism is enhanced in order to be used with the bandwidth-based mechanism.

ImTCP-bg detects network congestion using only the current and minimum values of RTT, whereas TCP Nice and TCP-LP also use the maximum RTT, which is difficult to observe in the actual network. When an increase in RTT is detected, the ImTCP-bg sender decreases its congestion window size immediately in order to resolve the congestion. Next, denote $\overline{RTT}$ as the smoothed RTT value that is calculated by the traditional TCP mechanism and RTT_{min} as the minimum RTT. Here, δ (> 1.0) is the threshold parameter to judge whether network congestion occurs. The ImTCP-bg sender detects the network congestion when the following condition is satisfied:

$$\frac{\overline{RTT}}{RTT_{min}} > \delta \tag{3}$$

When Equation (3) is satisfied, the ImTCP-bg sender decreases its congestion window size ($cwnd$) according to the following calculation.

$$cwnd \leftarrow cwnd \times \frac{RTT_{min}}{\overline{RTT}} \tag{4}$$

Equation (4) implies that ImTCP-bg determines the degree of decrease of the congestion window size based on the ratio of the current value of RTT and its minimum value. Thereby, ImTCP-bg avoids unnecessary the underutilization of the link bandwidth while maintaining the background-based data transfer. Note that this modification of the RTT-based mechanism is effective because the bandwidth-based mechanism is used concurrently.

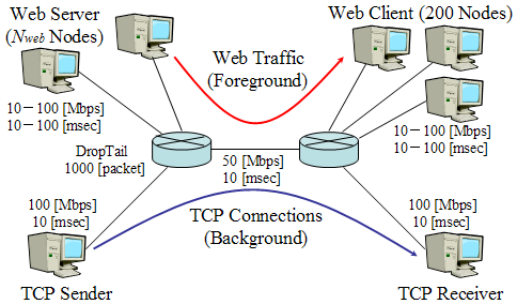


Fig. 1. Network model

IV. Performance Evaluation

In this section, simulation results are shown to evaluate the performance of ImTCP-bg proposed in Section 3. ns-2 [18] is used for the simulation experiments. Traditional TCP Reno, TCP Nice and TCP-LP were chosen for performance comparison.

The network model used in the simulation is depicted in Figure 1. This model consists of sender/receiver hosts, two routers, and links between the hosts and routers. The bandwidth of the bottleneck link is set to 100 Mbps, and the propagation delay is 10 msec. A DropTail discipline is deployed at the router buffer, and the buffer size is set to 1000 packets. The packet size is 1500 Bytes. Web traffic is assumed to be foreground traffic. N_{web} Web servers transfer Web documents to 200 Web clients. The bandwidth of the access link of each Web node is set randomly between 10 and 100 Mbps, and the propagation delay is also a random value between 10 and 100 msec. The amount of foreground Web traffic is adjusted by changing N_{web}. In addition, one or more TCP connections are established in order to perform background data transfer. The performance of the background TCP variants are compared with respect to the following: the transfer time of the foreground Web traffic, the queue length of the bottleneck link buffer, the throughput of the background data transfer, and utilization of the available bandwidth. The control parameters for ImTCP-bg are set as $\gamma = \frac{1}{8}$ and $\delta = 1.2$ which are determined by simulation trials, and for TCP Nice and TCP-LP are configured according to [9, 10].

A. Case of one connection

First, the simulation results are presented for the case in which one background data transfer connection is established, and the degree of interference with the foreground traffic and the utilization of network bandwidth are evaluated. The number of Web servers, N_{web}, is changed from 10 to 50. Figures 2, 3 show the change in the average throughput of the background TCP connection and the average download time of foreground Web documents. The results labeled as “available bandwidth” in Figure 2 and “no background traffic” in Figure 3 show the results for the case in which no background data transfer exists.

Figure 2 shows that although the TCP Reno connection achieves the highest throughput, the throughput exceeds the available bandwidth. Furthermore, Figure 3 shows that the av-

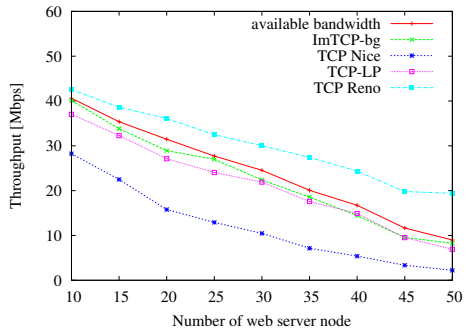


Fig. 2. Average of throughput in the case of one connection

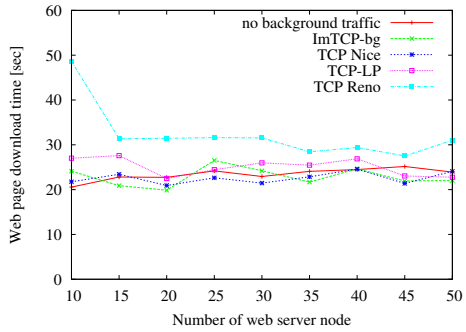


Fig. 3. Average of download time in the case of one connection

erage download time of the foreground Web documents is larger than for the case in which no background traffic exits. That is, TCP Reno cannot be used for background data transfer. On the other hand, for TCP Nice, TCP-LP and ImTCP-bg, the average download time in Figure 3 is almost identical to the case of no background traffic. This means that these protocols do not affect the foreground Web traffic, satisfying one of the objectives of the background data transfer. Furthermore, Figure 2 shows that the average throughput of the ImTCP-bg connection is the closest to the available bandwidth. Therefore, ImTCP-bg is determined to have the most ideal characteristics for background data transfer, which satisfies objectives (1) and (2) in Section 2.

B. Case of multiple connections

Finally, the results for the case in which two or more background data transfer connections are established are shown, and the effect of multiple background TCP connections is evaluated. In this simulation, five background TCP connections join the network at 0, 50, 100, 150, and 200 seconds, and end data transmissions at 500, 450, 400, 350, and 300 seconds. This means that the number of active background TCP connections in the network is as follows: 1, 2, 3, 4, 5, 4, 3, 2, and 1. Table I shows the average queue length at the output link buffer of the bottleneck router and Figure 4 shows the throughput of background data transfer connection as functions of time. Here, N_{web} is set to 20.

TCP Reno shows the worst behavior for the background data transfer, in terms of large queue length at the bottleneck link (Table I) and over-utilization of the available bandwidth of the network (Figure 4(a)). Table I also shows that the average of queue length of TCP Nice is the smallest among the four variants, meaning that TCP Nice is the best choice for satisfying objective (1), described in Section 2. However, Figure 4(b) shows that the throughput of the background data transfer is the lowest among the four variants, especially when the number of connection is small.

Figure 4(c) shows that when TCP-LP is used for background data transfer, packet losses occur immediately after a new connection is established. This is because TCP-LP needs the maximum RTT to control its congestion window size. TCP Nice and TCP-LP detect network congestion by using the minimum and the maximum RTT (or one-way packet delay). However, essentially, monitoring the maximum RTT by background TCP is difficult because these TCP variants decrease the congestion window size at an early stage of network congestion. Therefore, TCP-LP intentionally continues to increase its congestion window size until packet losses occur at the first slow start phase, while monitoring the maximum RTT value. Furthermore, in Figure 4(c) we can see that the throughput of TCP-LP connections are shown to be quite low for some time after the packet loss. This is because the fast retransmission and fast recovery mechanism of TCP Reno is activated.

Figure 4(d) shows that ImTCP-bg connections can utilize the available bandwidth of network path effectively even when only one connection exists. This is because ImTCP-bg controls its congestion window size appropriately using the results of inline network measurement. Furthermore, when multiple connections exist in the network, ImTCP-bg connections can maintain high utilization of the available bandwidth and the change in the throughput of each ImTCP-bg connection is stable compared with other background TCPs. That is because ImTCP-bg dynamically changes the degree of decrease congestion window size according to the change in the RTT. From these simulation results, the bandwidth-based algorithm with inline measurement and the RTT-based algorithm are determined to co-exist well in ImTCP-bg to realize background data transfer.

V. Conclusion

In the present paper, ImTCP-bg, a new background TCP data transfer mechanism that uses an inline network measurement technique, was proposed. ImTCP-bg provides a background data transfer without interfering with the foreground traffic by setting the upper limit of its congestion window size based on the results of the inline network measurement. ImTCP-bg also employs an enhanced RTT-based mechanism, which dynamically determines the control parameters. ImTCP-bg can detect and resolve network congestion even when reliable measurement results cannot obtained. Through simulation evaluations, the effectiveness of ImTCP-bg in terms of the degree of interference with foreground traffic and utilization of the available bandwidth was confirmed. In future studies, ImTCP-bg will be implemented and its performance will be evaluated in an actual network.

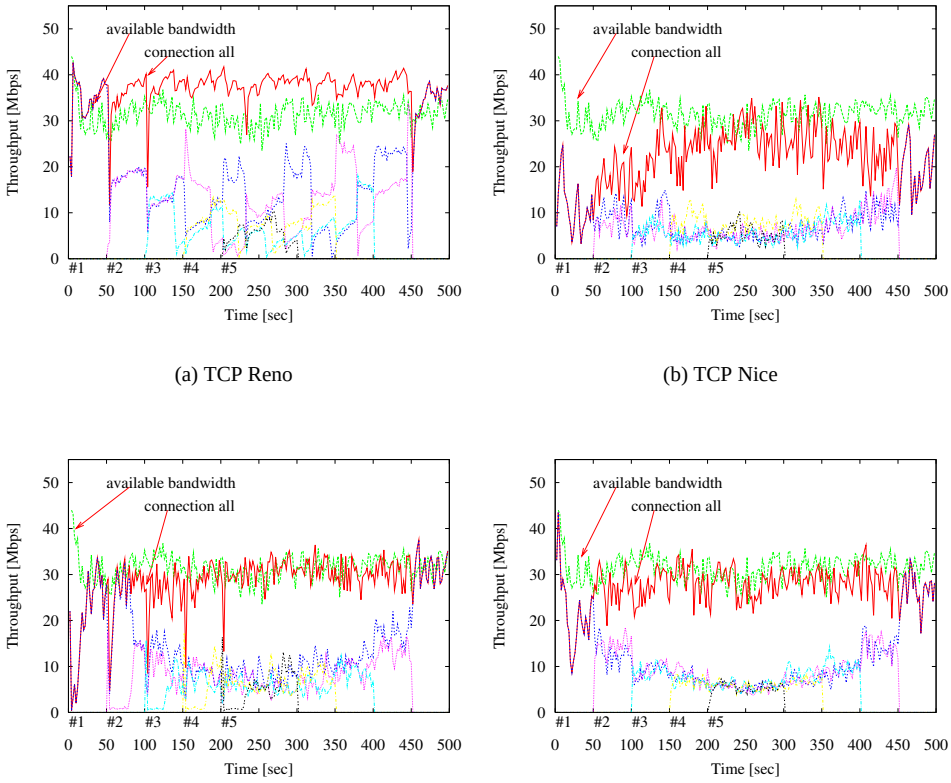
References

[1] W. R. Stevens, *TCP/IP Illustrated, Volume 1: The Protocols*. Addison-Wesley, 1994.

TABLE I

AVERAGE OF QUEUE LENGTH

	TCP Reno	TCP Nice	TCP-LP	ImTCP-bg
Average queue length [packets]	583.34	8.44	64.63	44.11



(a) TCP Reno

(b) TCP Nice

(c) TCP-LP

(d) ImTCP-bg

Fig. 4. Change of throughput in the case of multiple connections

[2] B. Krishnamurthy, C. Wills, and Y. Zhang, "On the use and performance of content distribution networks," in *Proceedings of ACM SIGCOMM 2001 Internet Measurement Workshop*, Nov. 2001.

[3] Akamai Home Page. available at `http://www.akamai.com/`.

[4] L. Breslau, P. Cao, L. Fan, G. Phillips, and S. Shenker, "Web caching and Zipf-like distributions: Evidence and implications," in *Proceedings of IEEE INFOCOM 1999*, Mar. 1999.

[5] A. Venkataramani, P. Yalagandula, R. Kokku, S. Sharif, and M. Dahlin, "The potential costs and benefits of long term prefetching for content distribution," *Computer Communication Journal*, vol. 25, pp. 367–375, Mar. 2002.

[6] Microsoft Corporation, *Background Intelligent Transfer Service in Windows Server 2003*, Sept. 2002. available at `http://www.microsoft.com/windowsserver2003/techinfo/overview/bits.mspx`.

[7] A. Rowstron and P. Druschel, "Storage management and caching in PAST, a large-scale, persistent peer-to-peer storage utility," in *Proceedings of SOSP 2001*, Oct. 2001.

[8] S. Blake, D. Black, M. Carlson, E. Davies, Z. Wang, and W. Weiss, "An architecture for differentiated services," *RFC 2475*, Dec. 1998.

[9] A. Venkataramani, R. Kokku, and M. Dahlin, "TCP Nice: A mechanism for background transfers," in *Proceedings of OSDI 2002*, Dec. 2002.

[10] A. Kuzmanovic and E. W. Knihtly, "TCP-LP: A distributed algorithm for low priority data transfer," in *Proceedings of IEEE INFOCOM 2003*, Apr. 2003.

[11] M. L. T. Cao, "A study on inline network measurement mechanism for service overlay networks." Master's thesis, Graduate School of Information Science, Osaka University, Feb. 2004.

[12] M. L. T. Cao, G. Hasegawa, and M. Murata, "Available bandwidth measurement via TCP connection," in *Proceedings of IFIP/IEEE MMNS 2004 E2EMON Workshop*, Oct. 2004.

[13] R. L. Carter and M. E. Crovella, "Measuring bottleneck link speed in packet-switched networks," *International Journal on Performance Evaluation*, vol. 27–28, pp. 297–318, Oct. 1996.

[14] B. Melander, M. Bjorkman, and P. Gunningberg, "A new end-to-end probing and analysis method for estimating bandwidth bottlenecks," in *Proceedings of IEEE GLOBECOM 2000*, Nov. 2000.

[15] M. Jain and C. Dovrolis, "End-to-end available bandwidth: Measurement methodology, dynamics, and relation with TCP throughput," in *Proceedings of ACM SIGCOMM 2002*, Aug. 2002.

[16] C. Dovrolis and D. Moore, "What do packet dispersion techniques measure ?," in *Proceedings of IEEE INFOCOM 2001*, Apr. 2001.

[17] V. Ribeiro, R. Riedi, R. Baraniuk, J. Navratil, and L. Cottrell, "pathChirp: Efficient available bandwidth estimation for network paths," in *Proceedings of NLANR PAM 2003*, Apr. 2003.

[18] The VINT Project, "UCB/LBNL/VINT network simulator - ns (version 2)." available at `http://www.isi.edu/nsnam/ns/`.