This full text paper was peer reviewed at the direction of IEEE Communications Society subject matter experts for publication in the IEEE GLOBECOM 2005 proceedings.

Improved Data Distribution for Multipath TCP Communication

Yohei Hasegawa Ichiro Yamaguchi Takayuki Hama Hideyuki Shimonishi Tutomu Murase
System Platforms Research Laboratories, NEC Corporation
e-mail: {y-hasegawa@bk, i-yamaguchi@ap, t-hama@cd, h-shimonishi@cd, t-murase@ap}.jp.nec.com

Abstract—**Multi-homed environments are increasingly common, especially for mobile users. To efficiently utilize multiple access lines for single file transfer, multipath TCP communication methods have been proposed. A multipath TCP enables simultaneous distributed data transfer between two end-points on multiple TCP connections. However, these methods cannot fully utilize the available bandwidth of multiple paths because they do not properly consider the end-to-end delay of packet transmission, so out-of-order data arrival at a receiver causes a bottleneck in data sort operations. This problem is more severe in environments where the quality of each path is different or unstable, such as in wireless environments. To solve this problem, we propose a multipath TCP communication method that includes a data distribution method to enable in-order delivery at a receiver. We call this Arrival-Time matching Load-Balancing (ATLB). ATLB continuously calculates the delay of each path, including the TCP queuing delay at a sender and the network delay, and then sends a data segment through the TCP connection with the lowest end-to-end delay. Simulation results show that ATLB improves end-to-end throughput, especially in heterogeneous environments where the quality of paths differs. For example, ATLB enabled twice the throughput with the conventional multipath TCP. We also report performance evaluation results from our ATLB test bed system in a wireless network environment. Our ATLB test bed system was able to fully utilize the aggregate available bandwidth of unstable multiple wireless links.**

I. INTRODUCTION

Network environments with multiple paths are becoming common between pairs of hosts [1]. The number of multi-homed users is increasing especially fast since mobile or wireless users often have multiple access channels. Conventionally, multiple access lines were prepared to ensure reliability. Today, in addition to high reliability, high performance is desired, especially for mobile or wireless users whose access lines are slow and unstable.

Although the number of users who have prepared multiple paths is increasing, the single TCP cannot provide sufficient throughput if packets are delivered via multiple network paths [2, 3]. This is because TCP is designed for connections that traverse a single path between a pair of hosts. Out-of-data delivery via multiple paths degrades TCP throughput.

To improve TCP data communication performance on multiple network paths, modifications of TCP [4, 5] and communication schemes with multiple TCP connections [6, 7] have been proposed. The former are modifications of single TCP behavior. These are mainly aimed at improving TCP performance in multipath forwarding networks where there are out-of-order packet deliveries. These approaches cannot always utilize each path, though, because their congestion control mechanisms remain almost the same as those of ordinary single TCP to enable friendliness and inter-operability with ordinary TCP.

The latter are parallel data transfer techniques in the application layer and multipath TCP communication methods in a new sub-layer on top of the TCP layer. The application-layer techniques are based on partial and parallel requests; for example, the range request of http1.1 and gridftp [8]. The sub-layer techniques [6, 7], which are called multipath TCP communication, are based on the distribution of data and parallel data transfer with multiple TCP connections between two end-points. In many cases multipath TCP approaches can provide higher throughput than the former approaches, because congestion control for each path can be used to determine the appropriate available rate of each path. Unfortunately, application-layer approaches involve a trade-off between data distribution efficiency and request data length – a larger request data size for each request makes the data distribution cruder. On the other hand, multipath TCP communication enables packet-by-packet data distribution, so data distribution efficiency can be higher than with application layer approaches.

Conventional multipath TCP methods still encounter a bottleneck in the data-sorting process at a receiver host. Conventional multipath TCP methods send data segments to each connection almost equally despite each path having a different delay or bandwidth. As a result, out-of-data arrival occurs at a receiver host. A conventional multipath TCP receiver needs a huge receiving buffer to sort the data segments from each path; otherwise an exhausted receiving buffer will result in a small TCP receiving window and degraded throughput. In addition, in high-speed communication the data-sorting bottleneck becomes more serious because a larger TCP window in each path increases end-to-end delay. Thus, high-performance multipath TCP communication requires an efficient data distribution method.

In this paper, we propose a data distribution method for multipath TCP communication that improves end-to-end performance and reduces the data-sorting cost. We call this method Arrival-Time matching Load-Balancing (ATLB). The method calculates the data arrival time for each path, considering the time that data segments spend in the TCP queue at a sender and the time needed for data segments to pass through the network. ATLB enables in-order data delivery to a receiver, and the data sort cost at a receiver is reducible. Thus, ATLB enables high throughput communication.

The rest of this paper is organized as follows. We briefly review related work in Section II. We describe the ATLB multipath TCP communication method in Section III. We present performance evaluation results in Section IV, and conclude in Section V.

II. MULTIPATH TCP COMMUNICATION

This section describes a multipath TCP communication scheme [6, 7].

The composition of multipath TCP communication assumed in this paper is shown in Figure 1. An end-host with multipath TCP includes an application layer and a transport layer. The transport layer is divided into two sub-layers – the multipath TCP layer and the TCP layer.

When a sender starts multipath TCP communication with a receiver, the sender opens N concurrent TCP connections to the receiver, where N is the number of paths the network provides between the sender and the receiver. The application layer then sends a data stream to the multipath layer, and the multipath TCP layer divides the data stream into multiple data segments, adding a control header to each segment. The control header contains a sequence number from the original data stream with which the receiver can reconstruct the data stream. The multipath TCP layer next transfers the data segments to the TCP layer, distributing some of the data segments to each TCP connection. When a destination TCP connection receives segments from its TCP sending peer, the multipath TCP receiver then reads the TCP data stream to recover the control data. Referring to the control data, the multipath TCP receiver reconstructs the original data stream. This data stream is then returned to the receiving application layer.

Figure 1 also shows an example of multipath TCP communication. In Figure 1, the sender uses multipath TCP to send a data stream to the receiver. The figure shows that the sender multipath TCP opens a TCP connection on each of the N paths provided by the network. The sender passes a data

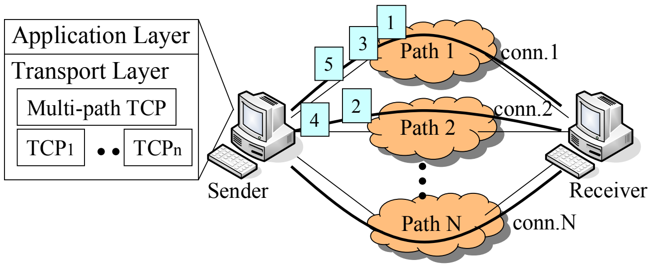

Fig. 1: Example of multipath TCP communication

stream to multipath TCP which is divided into five segments, labeled 1 through 5. The multipath TCP sequence numbers each segment and sends them onto the different TCP connections. The boxes above each TCP connection in Figure 1 denote TCP frames being transmitted by that connection. The number in the box denotes the multipath TCP sequence number which is contained in a control header.

As a multipath TCP communication data distribution strategy, the method in [6] uses dynamic data distribution where data segments are transmitted via the connection with the shortest TCP queue length, and the method in [7] uses a round-robin technique. These methods are aimed at maximizing the throughput of each path by putting data segments into each TCP connection to prevent buffer-under-run. These methods, though, suffer from the disadvantage of data sorting at the receiver. Along the various network paths, data segments transmitted from a sender experience different delays. In addition, the time that data spends in the TCP queue will differ, because each TCP's congestion control and retransmission mechanism become delay factors. Ultimately, the end-to-end delay of each TCP connection differs. Thus, data segments are not sequentially delivered to a receiver. A receiver must wait until a series of data segments arrives from each path.

Consequently, conventional multipath TCP cannot provide the desired application level throughput, because out-of-order data arrival on the receiver side causes a bottleneck in the data-sorting process.

III. PROPOSED METHOD (ATLB)

To solve the above problem, we developed the ATLB multipath TCP communication method. ATLB consists of a data distribution method to reduce the cost of data alignment in a receiver and a path-failure detection and recovery mechanism to prevent stalling of the data transfer.

1) *Data distribution method*

We show how ATLB calculates the data arrival time in Figure 2. With ATLB, the data arrival time at a receiver is estimated by taking into consideration the queuing delay in the sender's transmitting buffer and the network path delay (Figure 2). The queuing delay is dynamically calculated from each TCP connection's throughput history. The network path delay is calculated from the TCP's smoothed round-trip time (RTT). In addition to these two types of delay, there is a delay in the receiver buffer of each TCP connection. Since this is usually negligible, though, we do not take it into consideration here.

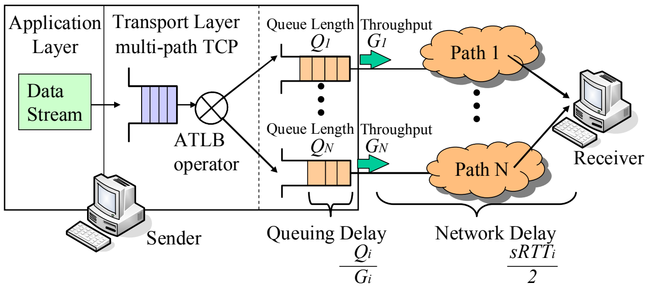

Fig. 2: ATLB Data distribution

The ATLB data distribution method is based on Eq. (1). ATLB calculates a *score* for each connection and sends a data segment via the connection having the lowest *score*. The first term in Eq. (1) is the queuing delay of the transmitting buffer, and the second term is the network path delay. Thus, ATLB dynamically sends a data segment through the connection having the lowest delay.

$$score_i = \frac{Q_i}{G_i} + \frac{sRTT_i}{2} \qquad (1)$$

where Q is the data length in the sender's transmitting buffer, *sRTT* is the smoothed RTT, and G is the smoothed throughput of connection$_i$, calculated as

$$G_j = \alpha \times G_{j-1} + (1 - \alpha) \times TPUT_j \qquad (2)$$

where α ($0 < \alpha < 1$) is a constant, and $TPUT_j$ is the throughput of a TCP connection which is continuously measured for every β milliseconds.

Parameters α and β determine whether ATLB should follow the latest throughput of each path. In a situation where throughput is steady from a long-term viewpoint, a larger α should be set. Otherwise, a smaller α is reasonable. β should be twice as large as RTT to stably measure the throughput of each connection in case of TCP's fast retransmission. We chose 1/2 and 100 for α and β, respectively, for the network environment we assumed in the performance evaluation because the network had a delay of 10 ms or more and the link bandwidth varied rapidly.

The buffer amount needed for sorting data segments in a receiver is reducible through the above method because the data from a sender will arrive at a receiver in an almost in-order sequence. Of course, momentary out-of-order data arrival will still occur, especially at the beginning of communication and due to TCP data retransmission, TCP congestion control, and network delay dynamics. If these dynamics are reflected in the TCP throughput, ATLB will react automatically by considering the queuing delay and network delay.

2) *Path-failure detection and recovery method*

ATLB has the following path-failure detection and recovery mechanism to prevent data transfer stalling.

a) *Path-failure detection*

ATLB maintains a failure-detection timer which expires after timeout *T*. *T* should be carefully chosen because a small *T* can be useful for detecting failure quickly, but misdetection becomes more likely. We assume that if a data segment just after a retransmit timeout (RTO) is lost, the path has failed or is heavily congested. We therefore make T equal to RTO + $\theta \times$ RTT ($\theta > 2$).

b) *Failure recovery*

When path failure is detected, ATLB starts to send probe packets via the failed path every *P* milliseconds. ATLB also calculates the packet loss rate every *S* milliseconds. If the packet loss rate is less than *R* %, ATLB assumes that the path has recovered. ATLB then restarts data transmission along that path. For example, we made *P*, *S*, and *R* respectively equal to 50, 500, and 10 to enable recovery within a second.

IV. PERFORMANCE EVALUATION

In this section, we look at the ATLB performance evaluation results. First, we show NS2 simulation results [9] to show that ATLB can improve end-to-end throughput when there is a data-sorting bottleneck in a receiver host. Second, we show experimental results with our test bed system in a wireless LAN.

A. *Simulation*

To test the effectiveness of ATBL when there is a data-sorting bottleneck in a receiver host, we compared the ATLB system to a conventional system.

1) *Simulation settings*

The simulation was done with the topology shown in Figure 3. We assumed that multipath TCP was applied to proxy servers to measure the end-to-end throughput, including the data sorting at a multipath TCP receiver, rather than to measure the aggregate throughput of each connection.

In the communication between sender H1 and receiver H2, GW1 and GW2 operated as proxy servers. Sender H1 tried to communicate with receiver H2 via GW1 and GW2. Multipath TCP communication was then established between GW1 and GW2.

To see whether ATLB improved performance, we compared the ATLB system to a conventional system with data distribution based on minimum queue length. In addition, to test the efficiency of data sorting at a multipath TCP communication receiver, we gave GW2 an infinitely large receiving buffer for sorting data segments from each TCP connection. The following two systems were prepared in this simulation.

-- ATLB

GW1 and GW2 enable the proposed ATLB communication.

-- MinQue

GW1 and GW2 enable data distribution where data segments are transmitted through the TCP connection with the shortest queue length.

Other basic simulation settings are shown in Figure 3 or stated elsewhere. The maximum size of the congestion window (the TCP transmitting buffer) was 2 Mbyte for each TCP connection. The maximum segment size was 1000 bytes.

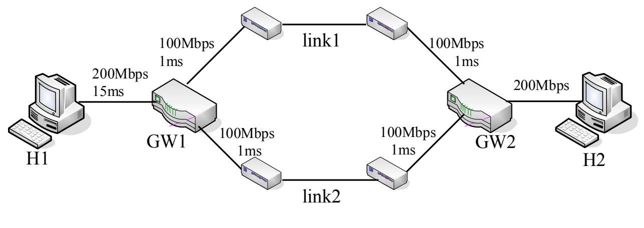

Fig. 3: Network topology in the simulation

2) *Robustness for link bandwidth differentiation*

Figure 4 compares the application level throughput at receiver H2 when the link 2 bandwidth was varied between 10 and 100 Mbps. The link 1 bandwidth was fixed at 100 Mbps. Each point of the graph is average throughput during a 10-s data transfer. We also show the ideal throughput, which is the aggregate throughput of links 1 and 2, and is not affected by the data sorting overhead at receiver H2. The results show that ATLB can fully utilize the aggregate bandwidth of links 1 and 2, considering the roughly 5% overhead of the TCP/IP header and multipath TCP control header. For example, when the respective bandwidths of links 1 and 2 were 100 and 20 Mbps, ATLB enabled throughput of about 115 Mbps, while the MinQue throughput was only about 63 Mbps.

The MinQue throughput was limited to about twice the bandwidth of the bottlenecked link 2. When the data transfer started, MinQue sent data segments at an almost equal rate to each TCP connection. This caused a data-sorting bottleneck at GW2. If we assume that the TCP congestion window of each path grew to 1 Mbyte in GW1, then MinQue would have sent 1 Mbyte data segments over each path. Since link 2 was limited to 10 Mbps, the data transfer on link 2 would take at least 800 ms, while the data transfer on link 1 would be done in about 80 ms. Until the data transfer on link 2 was complete, though, the data segments from link 1 had to wait at GW2. This means that data transfer from GW2 to H2 would be limited to roughly twice the bandwidth of bottlenecked link 2. Thus, the throughput between sender H1 and receiver H2 would be limited along with the bandwidth of the bottlenecked link.

3) *Robustness against link delay differentiation*

Figure 5 compares throughput at receiver H2 when the link delay of link 2 was varied between 7.5 and 75 ms. The link delay of link 1 was fixed at 15 ms. Each point of the graph is average throughput during a 10-s data transfer. The results show that both ATLB and MinQue can utilize almost the entire aggregate bandwidth.

When the link delay differs between paths, GW2 with MinQue has some data segments waiting to be sorted. However, if each path's throughput is the same, the queue length in GW2 almost remains at a fixed size *Qsb* which can be approximately calculated as

$$Qsb = || D1 - D2 || * B$$

where *D1* is the delay of link 1, *D2* is the delay of link 2, and *B* is the single-path bandwidth.

The ATLB and MinQue results were therefore almost the same in environments where only the delay differed between links. Both MinQue and ATLB have the same overhead at communication start up before the TCP congestion window on link 2 grows. These results help confirm that the simulation results are correct.

4) *Robustness against packet loss*

Figure 6 compares the robustness against packet loss. The packet loss rate of link 2 was set between 0.001% and 1%. We also show the ideal throughput, which is the aggregate throughput of links 1 and 2.

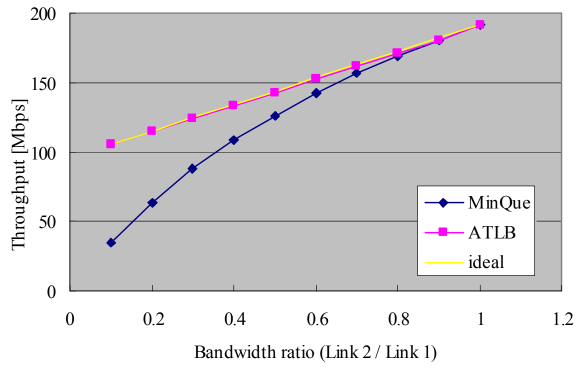

Fig. 4: Robustness for link bandwidth differentiation

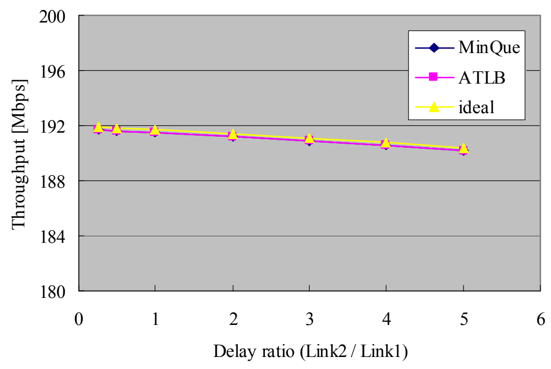

Fig. 5: Robustness for link delay differentiation

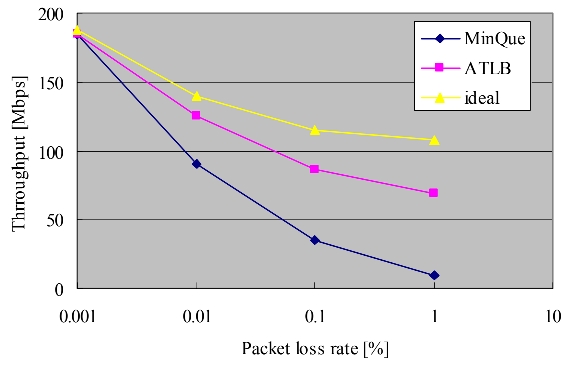

Fig. 6: Robustness for link packet loss rate differentiation

ATLB achieved higher throughput than MinQue, especially when the packet loss rate was 1%; in that case, the ATLB throughput was ten times that of MinQue. However, the ATLB throughput was below the ideal throughput. Since heavy packet loss causes frequent TCP fast retransmission and retransmission timeouts, the queuing delay of data transfer on link 2 gradually became larger. If a TCP retransmission timeout occurred, data transfer would have been stalled for at least 1 s. That would cause a lot of data segments to queue in GW2, so the data sorting in GW2 would be inefficient.

B. *Performance Evaluation on the Test Bed System*

In this section, we report the performance evaluation results for our ATLB test bed system on Linux in a real wireless LAN environment. To test whether our method can utilize all available bandwidth, we compared the ATLB throughput to the throughput of a conventional TCP data transfer established independently via each wireless link.

The test environment is shown in Figure 7. The network topology was the same as in the former simulation tests. Links 1 and 2 were IEEE802.11g wireless LAN links, and the others were 100Base/TX links. We introduced two background TCP connections on each wireless link as shown in Figure 7. Although each background TCP worked independently on links 1 and 2, each connection was supposed to utilize the available bandwidth of the wireless link. If the ATLB and background TCP throughput shared the bandwidth of links 1 and 2 fairly, this would show that the ATLB test bed system can utilize all of the available bandwidth of multiple wireless links.

To measure the ATLB performance in unstable wireless links, we introduced a test scenario which included a variety of wireless link speeds and wireless links being down or up.

The test scenario was as follows:

Time 0: The lengths of link 1 and link 2 were set to 4 m. A sender started data transfer.

Time 150: Link 1 went down.

Time 180: The length of link 1 was set to 60 cm, and then link 1 was up.

Time 250: Link 2 went down.

Time 300: The length of link 2 was set to 60 cm, and then link 2 was up.

Figure 8 shows the throughput trace under the test scenario. The bold line is the throughput trace on H2 where ATLB was introduced (ATLB-TCP). The bold broken line is the aggregate throughput of the background TCP connections (B.G.TCP1, B.G.TCP2).

The aggregate throughput was almost the same with the background TCP connections and with ATLB-TCP under this test scenario. This confirms that our test bed system utilized all available bandwidth, and the data sorting at GW2 worked well without degrading performance. In addition, when the link speeds were drastically changed, as when links went down or came up, the ATLB data distribution and failure recovery followed properly.

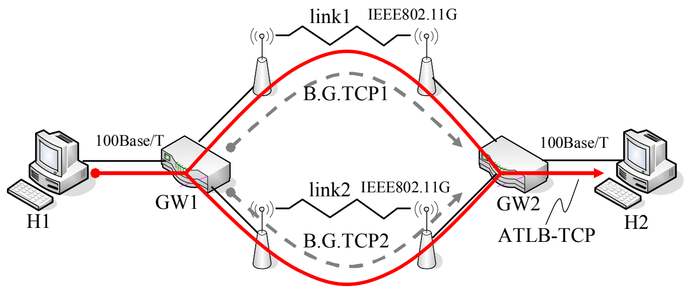

Fig. 7: Network topology

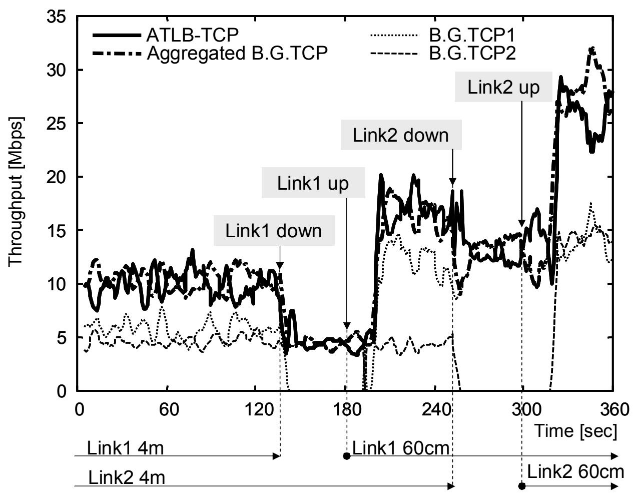

Fig. 8: Throughput on Wireless LAN

V. CONCLUSION

ATLB is a multipath TCP communication method that consists of a distributed data transfer method and a path-failure detection/recovery method.

Simulation results show that the ATLB data distribution method improves end-to-end throughput, especially in environments where the quality differs between paths. For example, the throughput of a multipath TCP system with ATLB was twice that of a conventional multipath TCP system.

We also evaluated the performance of our ATLB test bed system in a wireless LAN environment. The ATLB test bed system utilized all available bandwidth, even though it was applied in an unstable wireless LAN environment.

VI. REFERENCES

[1] S. Savage, A. Collins, and E. Hoffman, “The end-to-end effects of Internet path selection,” in Proceedings of ACM SIGCOMM, Aug. 1999.

[2] C. Barakat, E. Altman, and W. Dabbous, "On TCP Performance in a Heterogeneous Network: A Survey," IEEE Communications Magazine, vol. 38, no. 1, pp. 40--46, 2000.

[3] M. Zhang, B. Karp, S. Floyd and L Peterson, “RR-TCP: A Reordering-Robust TCP with DSACK,” in Proceedings of the Eleventh IEEE International Conference on Networking Protocols (ICNP 2003), November 2003.

[4] Youngseok Lee, Ilkyu Park, and Yanghee Choi, "Improving TCP Performance in Multipath Packet Forwarding Networks," Journal of Communication and Networks (JCN), pp. 148 - 157, vol. 4 no. 2, June 2002.

[5] Johnny Chen, “New Approaches to Routing for Large-Scale Data Networks,” Rice University, TR99-344, pp.119-144, June 1999.

[6] M. Zhang, A. Krishnamurthy, L. Peterson, R. Wang, “A Transport Layer Approach for Improving End-to-End Performance and Robustness Using Redundant Paths,” USENIX 2004, June 2004.

[7] Kultida Rojviboonchai and Hitoshi Aida, “An Evaluation of Multipath Transmission Control Protocol (M/TCP) with Robust Acknowledgement Schemes,” Internet Conference 2002, Oct. 2002.

[8] Pablo Rodriguez and Ernst W. Biersack, "Dynamic Parallel Access to Replicated Content in the Internet," IEEE/ACM Transactions on Networking, vol. 10, num. 4, Aug. 2002.

[9] The network simulator– ns-2, http://www.isi.edu/nsnam/ns