The difficulties of bandwidth estimation in highspeed network

P. Oothongsap, M. Vouk, and Y. Viniotis
North Carolina State University, Raleigh, North Carolina

October 15, 2003

ABSTRACT: Currently, the high-performance networks under development offer the promise of connectivity at speeds up to 40 Gbps or more. However, current versions of TCP/IP frequently cannot meet high throughput demands because of flow and congestion control mechanisms. For this reason, several new hybrid protocols such as RBUDP, User-Level UDP, Tsunami, and SABUL have been proposed as alternatives that take into account conditions arising in high performance networks in order to deliver better performance.

SABUL is one of the hybrid protocols that has a congestion control algorithm. Several SABUL empirical studies show that SABUL is able to achieve high bandwidth utilization in highspeed networks.

For example to achieve 1 Gbps throughput, the inter packet time between two packets is 12 μs where the packet size is fixed to 1500 Bytes and the end host system is a high speed machine. SABUL succesfully controls inter packet interval by counting the number of CPU cycles and not by using sleep command which has granurality in ms. This seems to be a good solution; however, the inter packet time is not exactly equal to 12 μs. It is always slightly greater than 12 μs since context switching may occur or the sender may need to read data from system input/output or execute other commands.

SABUL runs at the application layer and uses a UDP channel to transfer data; in the current version of SABUL, SABUL sender/receiver request 256 KBytes of a UDP buffer. Moreover, SABUL uses writev() command to place the packet to UDP channel. This is a blocking command. If UDP buffer is full, SABUL sender will be blocked until the buffer is available and the sender can place the packet to UDP buffer successfully. SABUL measures its throughput at the sender by using the formula:

$$throughput = \frac{number\ of\ sent\ packets}{time\ inteverval\ between\ two\ SYN\ packets} \quad (1)$$

From the above method, the measured throughput is essentially the rate of placing the packets to UDP buffer. This leads to several questions, e.g., does this rate match with the physical link rate?, etc. We have done a series of SABUL experiments in a very short-haul (local) and in a long-haul (Abilene) highspeed network. The following is our experiment setup.

In a local private network, we distinguish "slow" and "fast" end-clients (machines). The model of the "slow" machines is IBM Netfinity 4000R server (CPU speed of 700 MHz, 1 GB of memory, 1 Gbps Ethernet adapter). The model of the "fast" machines is IBM eServer x335 (CPU speed is 1 GHz, 1.5 GB of memory, 1Gbps Ethernet adapter). "Slow" machines are running Linux version 7.2 and "fast" machines are running 7.3. In addition to the 1 Gbps interface, each machine has a 10/100 Mbps interface which was used only for control and monitoring of the hosts. All experiments ran over the 1Gbps interfaces. Machines were interconnected via an Extreme Networks 6800 series Blackdiamond, 10Gb switch.

For a long-haul network, end hosts are located at three different Abilene end-point locations: North Carolina State University (NCSU), Georgia Institute of Technology (GT) and University of Washington (UW). At NCSU, there is one machine named localhost. At GT, there are three machines named fast1, fast2, and fast3. At UW, there is one machine named fasttcp. These are 4-way machines running Linux operating system (Kernel version 2.4.18) with CPU speeds of 3 GHz and 2 GB of memory. Each machine is equipped with 1-gigabit Ethernet card. These machines are connected to each other through the Abilene backbone network. The effective point-to-point capacity of each link between

Abilene end-points used in these experiments is 2.4 Gb/s. Then the bottleneck link in the path is 1Gb/s, to the end-host.

During the experiments, we have experienced the following:

1) Measured throughput by SABUL is higher than bottleneck capacity. Since the application layer has no idea about the physical layer or the speed of the network interface card, when SABUL sender tries to push as many packets as possible to the network, these packets will be buffered at the UDP buffer. With this behavior, SABUL sender may see sending rate higher than the speed of the network interface (note: this is a known phenomenon with tools such as ttcp, for example, and normally one would measure the actual end-to-end rate at the receiver).

2) There is an "unusual" throughput fluctuation in the measurements. SABUL sender calculates the sending rate from the number of packets sent divided by the time interval between two SYN packets that the sender receives from the receiver. The time measured between two SYN packets is random, and SABUL receiver uses a TCP channel to transmit SYN packets. SABUL sender will process SYN packets upon receipt. However, if the SABUL sender CPU is busy, then the kernel will not pass the SYN packet up to the application layer right away. With this behavior, the demominator in equation (1) is random.

Moreover, SABUL sender sometimes miscalculates the number of packets sent. With the errors in the number of packets sent and the randomness in equation (1), SABUL sending rate will increase sharply when the sender detects a large number of packets sent and the time interval between two SYN packets is very small. It will decrease sharply when the sender detects a small number of packets sent and the time interval between two SYN packets is very large.

These measurement accuracies have a direct impact on the accuracy of congestion control algorithm. Since SABUL congestion control algorithm depends upon the value of packet loss rate, the error in number of packets sent causes an error in the packet loss rate. The consequence is that the new sending rate in the next interval is also incorrect.

In addition to, measuring the throughput at the application layer, we measured SABUL throughput by using software/hardware sniffers. With these experiments, we have faced difficulties in measuring accurately throughput:

Software sniffer: There are several software packages such as Ethereal, tcpdump, that are able to capture packets and their timestamp when they are sent to network. During the experiments, we used tcpdump to capture the timestamp of each packet that is sent out. We ran tcpdump on the same machine as SABUL sender. With this capturing, we are able to explain the inter packet time between two packets. We can use this result to estimate the throughput of SABUL sender. However, tcpdump may fail completely capture all the packets. When that happens, one has to correct for the dropped packets (via hardware based counters on the adapters).

Of course, the end host CPU speed and capacity has an impact on the number of lost packets. If the endhost speed is high, the number of lost packets is smaller. This behavior is quite common since sender machine needs to send and filter the packets at the same time. This implies that software sniffer requires high amount of system resources.

Hardware sniffer: During recent experiments, we also investigated suitability of hardware sniffers, specifically Adtech 400. The sniffer is nominally a 1Gbps sniffer, and it receives a copy of every single packet that is sent by SABUL sender. Using hardware sniffer is the most accurate method of measuring the throughput since it is done at the physical layer. However, the hardware sniffer may have limited buffer size. The maximun number of packets that this particular sniffer could capture was only about 32 Kpackets (packet size of 1500 Bytes). This number of packets is too small for analysis since it only provides seconds of capture time at high speeds where thousands of seconds might be needed to recognize some protocol behaviors. We are in the process of investigating more powerfull hardware sniffers.

In the workshop, we are especially interested in discussing

(i) practically, advantages and disadvantages of direct vs indirect measurements when it comes to high volume data streams and bulk data transfer protocols

(ii) impact of application-level receiver-based measurements to control sender functions on congestion control, stability, and other properties of high-performance applications.