

Broadband Wireless Local Loop Evaluation Via an Active TCP Connection Monitoring Methodology

Harish Sitaraman, Jesse M. Davis, Joseph B. Evans
Information & Telecommunication Technology Center
Department of Electrical Engineering & Computer Science
University of Kansas
evans@ku.edu

Abstract

Active monitoring of TCP connections aids in the study of the stability of such connections over a given setup. With the advent of access technologies aiming to provide solutions to the last mile problem, there needs to be a method by which an ISP can study TCP connections over such equipment. TCP in itself has been designed to be stable and has been proven to be stable over many access technologies. But in many cases, the connections are not stable enough from a customer's perspective.

Our effort has been to develop a TCP connection monitoring methodology that the ISP can study the rate at which connections are lost. We develop a connection monitoring tool (ConMon) that measures round trip time and throughput apart from detecting connections drops. The path to the end point is also traced in order to study reasons for possible connection drops. We test this tool over wired LAN, cable modem and broadband wireless technologies.

1. Introduction

With the ever-increasing demand for faster and reliable Internet connections, newer technologies and equipment are being deployed by Internet Service Providers (ISPs). Technologies that have been used to provide Internet access to customers and the final mile include the dial-up modem, ISDN, cable modem, and lately DSL. Apart from cable modem, the rest use the telephone network to connect the customer to the ISP. The cable modem on the other hand uses the coaxial cable provided by the cable TV company to access the Internet.

The University of Kansas has been involved in a Broadband Wireless Local Loop (B-WLL)¹ trial in which we have studied the workings of a particular broadband wireless technology in an attempt to address the last mile problem.

The technology is relatively new and has not yet been widely used to provide Internet access to customers. The performance over such an outdoor wireless setup is subject to various factors including line of sight issues, interference, and weather conditions, to name a few.

We have developed an active TCP connection monitoring tool (ConMon) to provide a methodology for measuring the stability of the TCP connection to the service provider. Additionally ConMon measures the throughput, round trip time and makes an effort to detect the cause of a connection loss if it is a network problem. The tool would also help in understanding any routing anomalies between the source and the destination.

The need to study the performance of TCP connections from a customer's perspective was considered necessary to observe the stability over such a new setup. As ISPs provide various services to customers, it becomes necessary to understand the stability of the connection provided by the ISP and monitor other parameters such as throughput and round trip times. In addition to providing these capabilities, ConMon would also enable the service provider to study parameters such as average connection holding time and number of connection drops over a period of time.

2. Design and Features

We next explain the structure and design of ConMon, and elaborate on the design considerations and features of the tool.

2.1. Overview

ConMon can be functionally divided into the several daemons, namely:

- ConMon – Core server and client portions.
- Data transfer - Data request and transfer for throughput measurements.
- Traceroute - Path trace and connection drop path trace to detect anomalies.

¹ This work is sponsored by Sprint Corporation.

Figure 1 shows the overall block diagram of the ConMon tool.

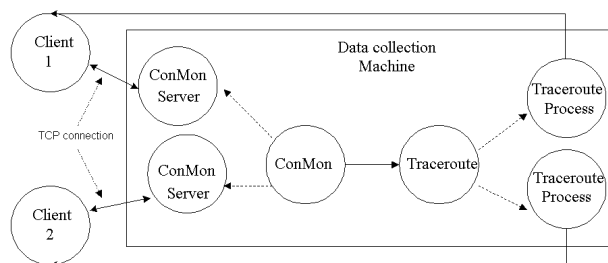


Figure 1. Connection Monitor (ConMon).

2.2. Selection of ConMon Parameters

The design of ConMon is based on a client-server architecture. The primary tasks of ConMon include:

- Monitor the TCP connection to check the frequency of connection losses.
- Measure the RTT on the connection.
- Measure the throughput over the connection.

Measuring the parameters above would supply the ISP with information regarding the connection that is provided to the customer. This would enable the ISP to study and possibly improve the connection behavior that is experienced by the customers. Any problems occurring outside the ISPs domain may not be of concern to the service provider since it may not be possible to rectify these problems.

2.2.1. Connection Loss The provision of diverse services such as data, voice and video over access technologies such as cable modem, DSL, etc., necessitates a stable connection from the ISP to the customer. By observing the frequency of connection losses, any behavioral patterns could be noted. For example, disconnects could happen more frequently during certain periods of the day when more users are connected to the provider network. This could probably be during evenings in the case of customers accessing the Internet from their residence or during the afternoons in the case of an enterprise.

A connection could be considered lost due to various reasons. The most common reasons are due to excessive congestion along the path of the connection, route changes, or failure of network equipment. From the customer's perspective, excessive congestion or network failure would reflect on the performance of the connection. If the performance deteriorates considerably, the customer will most likely disconnect and attempt to reconnect. This tolerance level can be used in the form of a waiting period before deciding that the connection is lost.

In ConMon, both the server and the client monitor their end of the socket. Since data is sent over the connection for the measurement of RTT and throughput, the connection is

timed out and considered lost if data cannot be sent over the connection within the waiting period. The client reestablishes the connection to the server after detecting a connection loss and the connection monitoring starts again.

2.2.2. Round Trip Time. The RTT is measured as the difference in time between when a packet is sent and when an acknowledgement returns when all the data has been received at the receiver end. During periods of heavy traffic, the RTT would provide estimates of turn around time over different connections (if multiple clients are available) and hence these could be compared.

In ConMon, a sample is defined as the number of RTT measurements over which the minimum, maximum, mean and median can be calculated. For example, if the sample size is 10, then after every 10 RTT measurements, the minimum, maximum, mean and median from that sample are calculated. At first, it might seem that we can use the mean value of the RTT as an estimate of that sample. However, it is generally observed that there are one or two RTT values in the sample that are much higher than the remaining values [13]. These values skew the mean of the sample. Hence the median of the sample is considered since it is a more robust value, being the central value of the sample.

2.2.3. Throughput ConMon can be used to measure throughput from either end of the connection. Options for throughput measurement such as packet size, number of packets and the frequency to perform a throughput measurement can be specified in the configuration file. The throughput measurement is performed only at the end of a RTT sample measurement i.e. if an RTT sample size is 10, then the throughput measurement can be performed only after the sample is completed. This would preserve the periodicity with which the RTT is measured. The throughput is measured at both ends of the connection, but only the value measured at the receiver side is stored.

2.3. Features of ConMon

The following are the salient features of ConMon:

- The ConMon server is designed to be a concurrent server. Hence it can accept connection requests from multiple clients.
- If a loss of connection is detected, the client will attempt to reconnect to the server. If this is not performed, but instead the client tries for a specified period of time and stops, then it is a tedious task to restart all the clients once the server is reachable
- Each client can use a different set of parameters as configured on the server side.
- It provides the ability to transfer all the collected data files to a requesting host. This eases the burden of manually copying files from the clients to the server.

- ConMon can use the functions of a traceroute daemon to trace the path to the clients that it is currently monitoring. This would assist the user to detect any routing problems along the path of the connection when a connection is lost.

2.3.1. Traceroute Daemon A traceroute daemon (tracertouted) can be used to complement the working of ConMon. Some of the initial motivation for introducing this feature was achieved when ConMon was detecting connection losses, but the reason for these losses were not known. There could be various reasons for observing a connection loss. It could be due to a failure on the end hosts, or perhaps due to a network problem along the path of the connection. In the Internet, most of the connection instabilities occur due to reasons such as a router flapping between different interfaces or due to a temporary network outage on a certain link. It is in general a rarity for connection failures to originate at the end hosts (unless the end host is shutdown or rebooted).

There is also a possible occurrence of routing asymmetries on the Internet e.g. if the upstream and downstream routing path is different. This could affect the perceived throughput and round trip times over the connection.

2.3.1.1. Tracertouted The basic working of traceroute was explained in the previous section. Considering the method by which traceroute works, it can be modified to execute as a daemon and accept connections from ConMon. The traceroute daemon (tracertouted) could also be used to observe any routing asymmetries between the client and the server. The daemon could be run on either end of the connection, and hence the path can be traced from both ends of the connection.

Tracertouted is an independent entity from ConMon. When a client connects to the ConMon server, the server notifies tracertouted providing it with information about the connecting client. The path to the client is traced and output logs are stored for reference. A separate pathtrace file is also maintained for each client and contains a chronological log of changes in the path to the client. The file displays the date and time at which a change was noted, the number of hops to the destination and the hop at which a change was detected.

Since tracertouted is primarily a version of traceroute modified to accept requests from ConMon, it can also be used to trace the path to the client when a connection loss is observed. This could provide some insight into possible problems along the path to the client. Tracertouted can also be used on the client side to provide similar functionality to trace the path to the server.

2.3.2. Copy Daemon In an effort to integrate a method to copy data along with ConMon, a copy daemon was designed for this purpose since other copying methods such as rsync

and ssh may not be installed. ConMon runs a separate process that accepts connections from hosts requesting for data that it has collected. Effectively both the ConMon client and server run this daemon so that the data that has been collected can be copied to the requesting host. The two basic features of the daemon include:

- A file transfer utility (netpipe²) that is used to transfer the data files to the requesting host.
- A method to find and copy only files that were not copied since the last request. This would prevent unnecessary copying of files.

The host that requests the data uses a copy request client that uses the clients file generated by the ConMon server. The clients file contains the IP address of the clients currently being monitored. In case a particular client was being monitored in the past, but is not being monitored currently, then this is reflected in the file. When requesting for data, the clients file is referred and each client that is currently monitored is contacted and data is requested. The copy request client can be run from a crontab and hence data can be periodically requested from the clients.

A basic authentication mechanism is provided in the daemon. The requesting host provides a username and password that should be verified by the copy daemon. Once the authentication succeeds, the data transfer is allowed. This mechanism however is not secure as information is exchanged in a clear text manner. A better and improved mechanism is ultimately required. The sequence of requests and actions is shown in Figure 2.

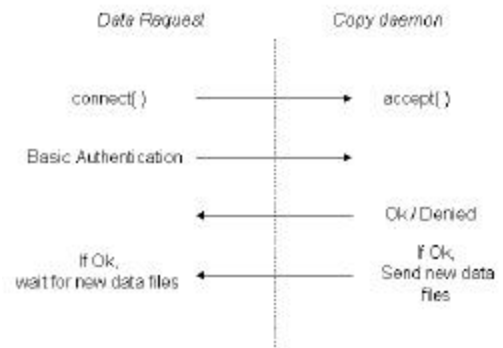


Figure 2. Copy daemon.

2.3.3. Netpipe Netpipe is a file transfer utility written by Roel Jonkman at the University of Kansas. It was written from an inspiration of netcat and ttcp. Netpipe accepts input from stdin, which implies that the input can be provided from the command line or can be piped to it. This input data is further sent over a TCP connection to the requesting host. A typical usage of netpipe is as follows:

- Sending end: `tar -cvf - * | netpipe -s 192.168.4.1`

² Originally written by Roel Jonkman, University of Kansas [15]

- Receiving end: netpipe | tar -xvf -

On the sending end, the data collection directory tree is archived into a tarfile and piped to netpipe, which sends it over the TCP connection to the receiving end, which untars the tarfile.

Netpipe uses the primary mechanism of the poll system call in Unix and reads data from stdin.

2.3.4. Data Storage Format The collected data is stored in the Netlogger message format [11] developed at the Lawrence Berkeley National Lab (LBNL) as part of the Netlogger toolkit. The format derives from the Universal Logger Message (ULM) format from an IETF standard:

- a list of "field=value" pairs
- Required fields: DATE, HOST, PROG, LVL
 - DATE: YYYYMMDDhhmmss
 - PROG: Name of the software component that issued the ULM.
 - HOST: Name of the host that generated the ULM.
 - LVL: Security level (Emergency, Alert, Error, Usage, etc.)
- Optional user defined fields
- Netlogger adds a required field, namely NLEVENT, which is a unique identifier for the event being logged.

The PROG field was modified to signify the component within ConMon for which this ULM is generated. Since certain client hostnames at customer premises do not resolve into the fully qualified domain name (FQDN), the IP address of the host was used in these cases as the value for HOST. The destination address is also stored in the file.

For the RTT measurement, the minimum (Min), maximum (Max), mean (Avg) and median (Median) are also stored along with the packet size (pktsize) and number of packets per sample (packets). In the throughput measurement, the throughput measured is stored (throughput) instead of the sampled values. For logging information about connection drops, the time at which the connection was established (connect), the time at which it was disconnected (disconnect) and the connection drop number (number) are stored. From the connection establishment and disconnect times, the duration of time for which the connection was stable and connected can be calculated.

All data files are stored in a compressed format, using the zlib compression library. The data file is of the format < t3m3st3mp > dat.gz, where timestamp is the time at which this file is created.

2.3.5. The Plotter The collected data is plotted using a custom plotter. The initial efforts in the development of the plotter started in the AAI project at the University of Kansas. The plotter was primarily developed to read and plot from data files that are based on the Netlogger message format.

Certain enhancements were made to the plotter to plot connection drops.

The different components that comprise the plotter kit are:

- Creation of Images (GIF) using the gd graphics library.
- Reading of compressed files achieved using zlib, a data compression library.
- Ability to generate daily, weekly, monthly and user specified timerange plots using external scripts.

gd is a graphics library which allows the user to draw images using lines, rectangles etc. from the source code. It provides function calls using which the user can generate GIF images.

zlib is a free lossless data compression library that supports reading and writing files in gzip (.gz) format with an interface similar to that of stdio. It provides in-memory compression and decompression functions.

2.3.6. Usage Scenario A typical scenario in which ConMon could be used includes running the server at the ISP head end and the client on the customer end. Hence multiple clients can connect to the server at the head end, and the performance of the connections can be observed.

Figure 3 depicts a scenario in which ConMon can be used. The figure depicts a cable service provider and the customer end. The ConMon server can be run on a machine attached to the CMTS. The client can be run on the customer machine. This will enable monitoring of the connection provided by the service provider to the customer. It can also be noted that the customer's traffic passes through the CMTS on the service provider's end, and is further routed by the router onto the Internet.

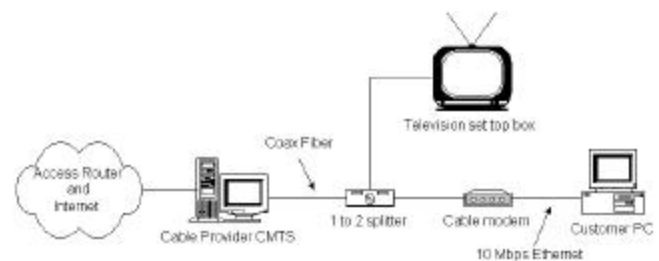


Figure 3. Cable network usage scenario.

Figure 4 depicts another scenario in which the link to the service provider is a wireless link. In this case, the base station is located at the ISP end. This base station is responsible for providing service to the customers in its range. The ConMon server can be run on the server attached to the base station and the client could be run on the customer machine. This would enable the ISP to measure the stability of the connections over the wireless link. The tool can also be used in other cases, where the user expects to measure the stability of TCP connections between two end points. Yet

another scenario would be using ConMon between end points that traverse the public Internet.

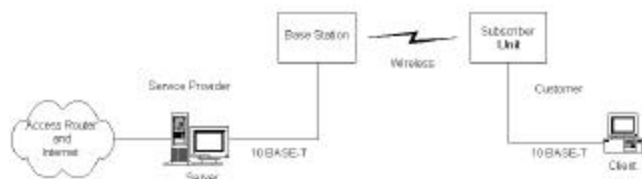


Figure 4. Broadband Wireless usage scenario.

3. Implementation

This section explains the implementation details of ConMon. It highlights the working of ConMon along with an explanation of how the traceroute daemon complements the working of ConMon.

The ConMon server is a concurrent server that runs on the central monitoring site. It is capable of handling requests from multiple clients.

3.1. Monitoring and Measurements

The primary objective of the server is to monitor the TCP connection end-point and observe the stability of the connection. A connection is meant to be stable and alive if data can be sent over the connection at the required times. Hence, when a RTT or throughput measurement is performed and it is not possible to send all the data within a certain period of time, then the connection is declared to be lost. The amount of time before which we can decide that the connection is lost, depends on modeling the user's patience level.

ConMon uses a timeout value of three times the waiting time required for the next expected packet. e.g. If the time between sending RTT packets is 60 seconds, then the timeout value is set to 180 seconds. If it is not possible to send or receive data over the connection while measuring throughput, then the connection can be considered lost after waiting for the timeout value. The calculation of this timeout value can be altered depending on the testing scenario. If the timeout is calculated to be too small, then a connection will be declared lost even before all the data can arrive in a reasonable amount of time. Hence there must be a certain tradeoff with which this timeout value is initialized.

The server periodically measures the RTT to each connected client using the parameters in the configuration file, if provided. The RTT is measured only by the server. After the sample size number of RTT's have been measured, the minimum, maximum, mean and median values from the sample are stored in the data collection directory for that client.

The throughput measurement can be configured to be measured from either the server or client side, considering that the service provider has configured asymmetric

bandwidth allocation for the upstream and downstream links. The measurement is performed only after a sample of RTT measurements.

3.2. Tracerouted Details

The traceroute daemon (tracerouted) can be used to complement the working of ConMon. It can be used with both the ConMon client and server. The freely available traceroute utility originally written by Van Jacobson and extensively rewritten by Eric Wassenaar [7] was modified to function as a concurrent server that accepts requests from ConMon and traces the routing path to multiple clients simultaneously. The primary functions of tracerouted are as follows:

- On connection establishment, ConMon sends a request to tracerouted over a TCP connection to trace the path to the end-point of the connection. e.g. ConMon client requests a trace to the server and vice versa for the server.
- On a connection disconnect, ConMon requests tracerouted to perform an immediate traceroute to the end-point in an attempt to detect the reason for the connection loss.
- Once the above request has been sent, ConMon sends a message to tracerouted requesting it to stop the currently performed trace to the end-point. The traceroute is stopped only after waiting for a certain period of time.

The tracerouted creates and maintains a trace file for each monitored host. This file contains information regarding any fluctuations or problems on the route to the host. This would assist in the detection of any routing problems.

Hence tracerouted provides a path tracing functionality in which the path to the client is traced periodically.

Since both the client and server part of ConMon monitor the connection, it is possible for the client to detect a connection loss by timing out earlier than the server. When this occurs and if the connection is reestablished before the server times out, the server will request a second traceroute for the same end host. Hence tracerouted ignores such multiple requests for the same end host when the trace is currently being performed. Whenever a trace is requested, the status variable is incremented. If another request arrives before the current trace is stopped, the status variable is incremented again. Such multiple requests arrive only when the client reconnects prior to the server timing out. When the server times out, the status variable is decremented.

Tracerouted stores all the periodic traceroute logs and traceroutes that are done immediately after a connection loss.

3.3. Operations

The ConMon server runs on the central monitoring site. Tracerouted can be used to trace the path from both the client

and server side. The ConMon client runs on the customer end. When the client connects to the server, and if the path needs to be traced, the server notifies tracerouted with parameters about the connecting client. The server monitors the TCP connection apart from measuring the round trip time. Depending on whether throughput measurement is required or not, the measurement is performed by either the server or the client and the receiving side logs the measured throughput.

The traceroute daemon traces the path to the end host and generates a file containing any fluctuations or problems on the route. In addition, ConMon also notifies the traceroute daemon in the event of a connection loss. In this case, an immediate traceroute is performed to the end host in an attempt to find if there is any routing problem towards the destination that could be a possible reason for the loss of connection.

4. Experimental Setup and Results

4.1. Motivation for Tests

The B-WLL equipment was deployed at a customer premises in Lawrence, Kansas to test the performance of connections over the wireless link. The cable modem is a popular access technology used for Internet access and hence it was interesting to compare the stability of the connections over both these technologies. The B-WLL equipment was fairly new and aimed at providing Internet access including data, voice and video. ConMon was also run on the wired on-campus local area network to observe connection behaviors on the local network. The test setup and results pertaining to each access technology are explained in further sections.

4.2. B-WLL Experimental Configuration

The B-WLL setup involved the deployment of a SU at the rooftop of a customer's residence. The Access Point (AP) was setup on the rooftop of ITTC and was behaving as a point-to-point connection to the customer SU. The SU was at a distance of 2.4 km from the AP. The antennae were directional and the SU was within the line of sight of the AP. The logical topology setup is depicted in Figure 5.

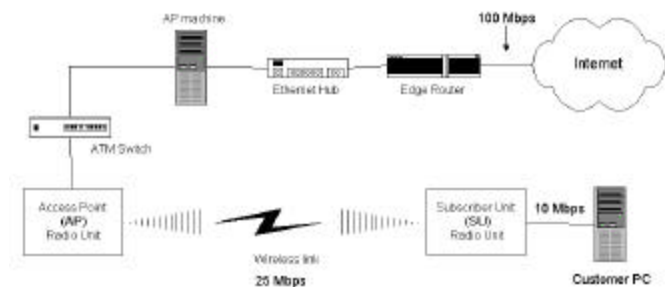


Figure 5. B-WLL Network test setup.

The wireless channel was capable of achieving 25 Mbps and used the un-licensed frequency channels for transmission. The upstream and downstream channel bandwidth allocations were symmetric and so was the routing. The customer used the SU as the gateway from the residence and accessed the public Internet using ITTC as the gateway. The connection was an always-on connection.

The ConMon server was run on the machine connected to the AP and the client was run on the customer machine connected to the SU. The measurement was equivalent to observing the behavior on the first hop connection to the ISP.

4.2.1. Wireless Channel Verification Before the experiments for the ConMon client and server on the wireless channel took place, experiments to test the capacity and behavior of the wireless channel were conducted. This involved using netperf [14] to test the reported bandwidth of the channel with differing socket and message sizes. It was assumed at the beginning of the tests that the behavior of the channel would mostly model standard Ethernet behavior with the socket and message sizes we were modeling. With the wireless channel behaving like Ethernet, the results from LAN tests and the wireless test could be more easily compared.

The first subscriber unit (SU #2) was located 2.4 kilometers away from the base station, while the second unit (SU #3) was located approximately 10 meters from the base station.

As is seen in the plots below, some erratic behavior is seen on SU #2 (Figure 6). This behavior was never fully explained. However it does seem likely that more retransmissions were needed due to errors occurring from the stability of the wireless link. However, SU #3 demonstrated Ethernet-like behavior for tests that sent a maximum amount of traffic through the link, as most socket and message sizes result in an 85-90% utilization of the link. SU #2 demonstrated this as well for the small message size test.

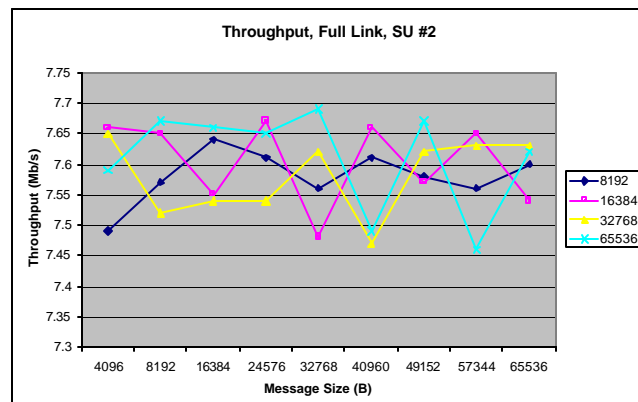


Figure 6. Unidirectional Flood Test Results for SU #2.

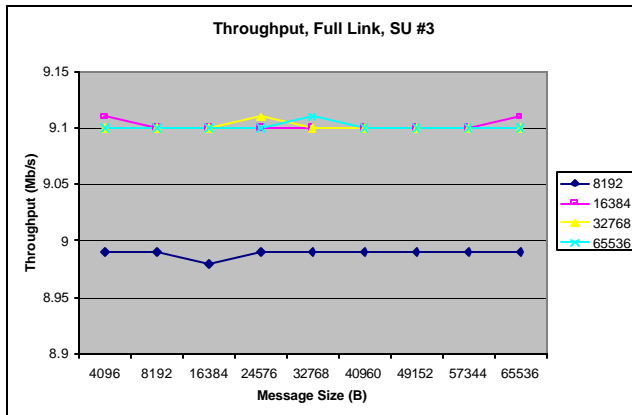


Figure 7. Unidirectional Flood Test Results for SU #3.

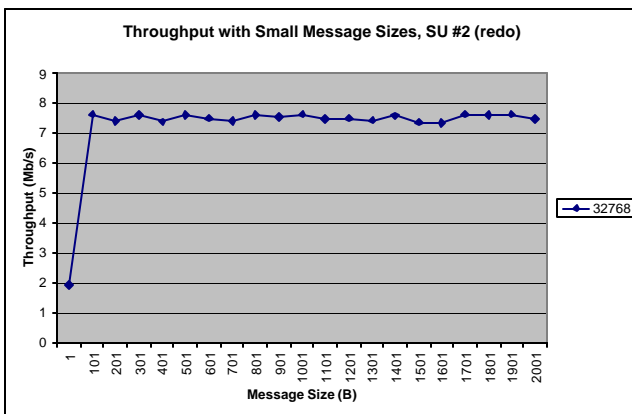


Figure 8. Small Message Test Results for SU #2.

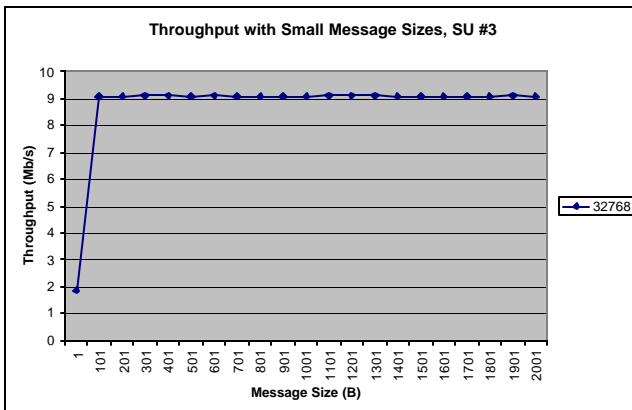


Figure 9. Small Message Test Results for SU #3.

4.3. Cable Modem Experimental Configuration

The ConMon client was run at three customer's home machines that use the cable modem to access the public Internet. Our efforts to run a ConMon server from the cable provider's CMTS were not successful, as we could not obtain permission from the local cable provider. With this constraint, the cable modem setup involved running the

ConMon server on a Solaris machine at ITTC. The logical topology of this setup is depicted in Figure 10.

The routes to the clients were across the public Internet. The traffic from the client passed through the cable provider's network and then through Sprint- link to finally reach the server. The routes were symmetric and normally took 12 hops between the client and the server. Even though the server runs on a single machine, it can be assumed to be a connection to a particular end-point (e.g. website). This also aims at testing the connection to the cable provider in addition to observing routing problems when traffic passes the public Internet.

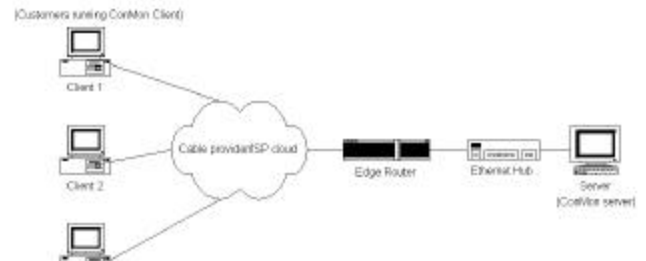


Figure 10. Cable Modem Network test setup.

4.4. Local Campus Experimental Configuration

This setup used the ConMon server running on the Solaris machine used for the cable modem setup. The clients were run from two Linux machines in the department network. The route involved passing through the default gateway and reaching the destination machine from either end. The topological setup is depicted in Figure 11.

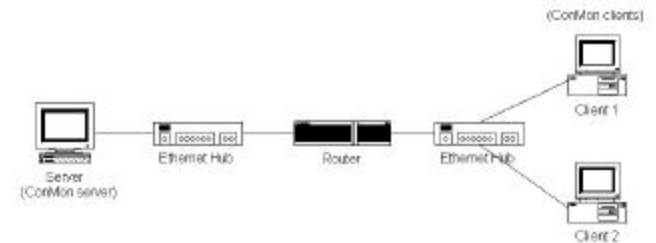


Figure 11. Wired LAN test setup.

The link to the first hop gateway was a 100 Mbps link. The departmental network used 10 Mbps Ethernet. The parameters used for each of the clients were initially same for an even comparison. The clients.conf configuration file had a parameter set for each client similar to the following line:

```
192.168.4.32 -r 1024 -n 10 -f 30 -C -t 1024 -N 2000 -F 1
-sbuf 65535 -rbuf 65535
```

To elaborate on the details of the above line, the RTT packets used a packet size of 1024 bytes and were sent every

30 seconds. The sample size was 10. The throughput measurement was performed by the client. This was changed at frequent intervals, so that the throughput in the other direction could be measured. The packet size used for the throughput measurement was 1024 bytes and 2000 such packets were used. The measurement was done once after every RTT sample.

4.5. Observations on the B-WLL Experiments

The AP and the SU that were used in the testing were in the beta stage of development. Due to this, the units did not operate in a stable manner continuously. The reason for the instability can also be attributed to external conditions such as low temperatures and heavy winds though this could not be confirmed.

The number of connection drops in this setup was relatively large and this was primarily due to the failure of the wireless link. When the link became unstable, the TCP connection often timed out. When the connection was reestablished, a disconnect used to occur shortly. Hence during the period of instability, the number of drops would be high. When the link failed, the units could not reach each other, and hence communication ceased. During this time, the ConMon client periodically tried to reconnect. Once the link started working, the TCP connection was established and the monitoring resumed. It should also be noted that the SU was connected to the customer machine using 10 Mbps Ethernet and hence the peak achievable throughput was constrained even though the link was 25 Mbps.

Figure 12 displays the throughput achieved over a period of time from the AP to the SU (downstream).

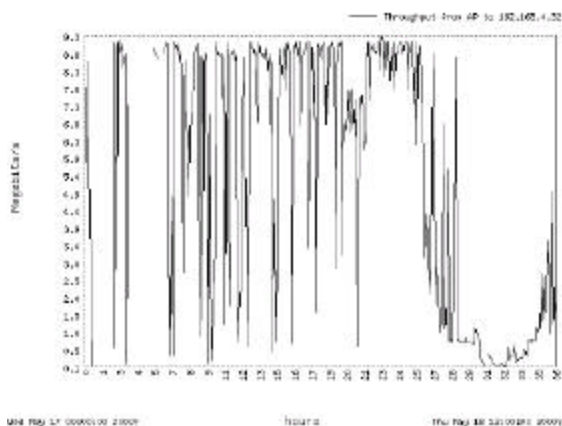


Fig. 12. WLL - Throughput from the Access Point to Subscriber Unit (Downstream).

It can be observed that the throughput peaks within 10 Mbps and mean throughput during this range is 5.9 Mbps. There were no connection drops observed during this period and the connection was very stable. When the wireless channel was stable, the throughput observed on the

downstream and upstream direction were around 8 Mbps and 6 Mbps respectively. This provides an aggregate throughput of 14 Mbps. It should be observed that in spite of only one user utilizing the channel, the Media Access Control (MAC) layer in the radio units constrain the amount of bandwidth that can be utilized by a single user.

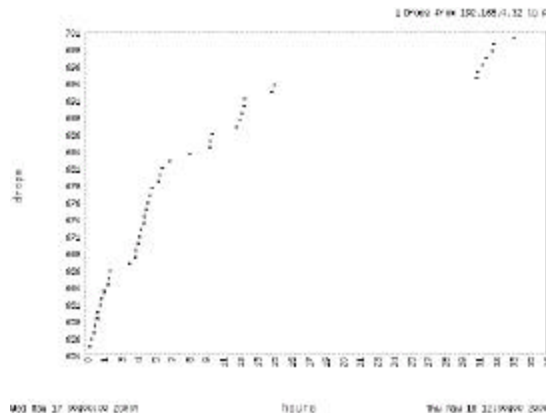


Figure 13. WLL - Connection drops observed at the Subscriber Unit.

Figure 13 shows the connection drops during this period. The numbers along the Y-axis signify the drops since the monitoring started. By observing the graph it can be seen that the 655th drop occurred at around midnight on 17 May. It can be observed from figure 6.5 that there is break in the measurement at around 6:29am on May 18. In this case, the link failed for a short period of time. Once the link was restored, the SU was reachable from the AP.

4.6. Observations on the Cable Modem Experiments

Most of the problems seem to arise due to outages and fluctuations along the route between the end points. During this time the connections were lost and were reconnected back after the duration of the outage. During this time the customer would not have been unable to reach any destination address that was routed via that router, unless there was an alternate route.

Using traceroute to find the reason for a connection drop aims at finding possible routing problems and it is possible that the route could have stabilized by the time the traceroute is performed. This would not indicate any failure along the path, where there might have been one.

An example of observed performance is due to asymmetric routing between the end-points. The downstream path to the client was over the usually observed 12-hop route, but the upstream route was over a much higher latency route of 9 hops.

```
traceroute to server (server), 64 hops max, 40 byte packets
1 10.204.254.254 21.253 ms 12.965 ms 20.248 ms
```

2 24.124.5.254 28.871 ms 18.794 ms 22.317 ms
 3 205.171.48.245 47.161 ms 67.907 ms 46.578 ms
 4 205.171.16.101 43.1 ms 47.520 ms 46.580 ms
 5 205.171.16.98 43.416 ms 47.777 ms 45.894 ms
 6 205.171.48.58 512.718 ms 308.600 ms 202.178 ms
 7 164.113.232.201 336.186 ms 330.787 ms 333.170 ms
 8 164.113.201.250 199.898 ms 259.492 ms 326.153 ms
 9 server 440.620 ms * 375.196 ms

The above traceroute denotes an upstream route passing through Qwest, and the downstream route using the normal route through Sprintlink. It can also be observed from the above traceroute that the latency steeply increases at hop 6. The downstream route had an average latency of around 85 ms. Due to this asymmetric routing, the observed throughput was only around 4-6 Kilobytes per second. However, when this route fluctuation happened midstream from the 12-hop route to the 9-hop route, the throughput observed was much higher than the normally observed throughput. The throughput peaked at 1.8 Mbps, but later the latency increased and hence the performance decreased.

Figure 14 displays the throughput from the clients to the monitoring server. It can be seen from the initial part of the graph that the connection was unstable due to which the throughput had declined.

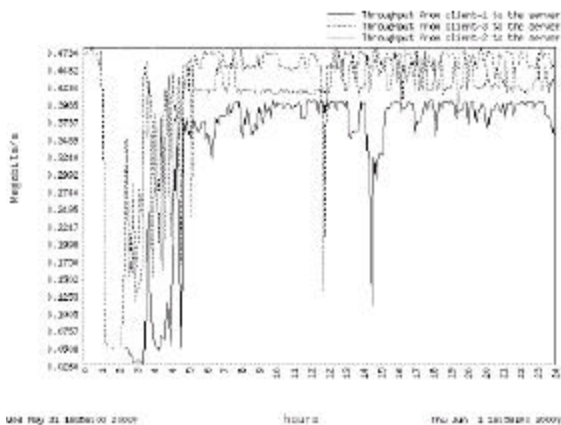


Figure 14. Cable Modem Network - Throughput from Clients to Server.

4.7. Observations on the Campus Experiments

The connections were stable most of the time on the on-campus setup. Only once did the connections disconnect due to the first hop gateway failing.

4.8. Comparisons

Measurements were conducted over a period of one month and the throughput and connection drops on the different access technologies were studied. Table 1 displays the upstream throughput observed on the different clients over the period of one month. It must be noted that since the cable modem is a shared network, the performance depends

on the number of customers using the service. The upstream limit set by the cable provider was 0.5 Mbps.

Client	Minimum (Mb/s)	Maximum (Mb/s)	Mean (Mb/s)
WLL Client	0.017	7.631	4.413
Cable Client 1	0.026	0.402	0.354
Cable Client 2	0.115	0.479	0.434
Cable Client 3	0.036	0.498	0.440
LAN Client 1	0.093	0.739	0.545
LAN Client 2	0.190	0.721	0.557

Table 1. Upstream Throughput Comparison.

Table 2 displays the connection drops observed at the client side over this period. It can be observed that the number of connection drops over the wire- less link is relatively large. This is primarily due to the instability of the wire- less link. The drops observed on the cable modem network is primarily due to routing problems on the public Internet, and the drops on the LAN were due to first hop gateway failures and client host reboots. The table also displays the average connection hold time that is the average duration for which the connection is stable and connected.

Client	Connection drops	Average Connection hold time (minutes)
WLL Client	1219	18.93
Cable Client 1	18	1648.93
Cable Client 2	14	759.67
Cable Client 3	21	1207.05
LAN Client 1	3	7395.38
LAN Client 2	4	1381.496

Table 2. Client side Connection drops.

Table 3 compares the downstream throughput observed from the server to the clients. A downstream measurement was not performed for the third cable client. The downstream bandwidth limit provided by the cable provider was 2 Mbps. The wireless link was able to achieve up to 9.25 Mbps.

Client	Minimum (Mb/s)	Maximum (Mb/s)	Mean (Mb/s)
WLL Client	0.027	9.251	6.488
Cable Client 1	0.014	1.205	0.756
Cable Client 2	0.023	2.019	1.380
LAN Client 1	0.226	4.012	1.816
LAN Client 2	0.203	3.421	1.679

Table 3. Downstream Throughput Comparison.

The variation of round trip time is shown in Table 4. It can be seen that the RTT varies widely over the wireless link. The high values are due to an unstable link or temporary link failures. Though all the cable clients were using the same service provider, the RTT to client 1, was higher than to the remaining two cable clients.

It has been observed that the wireless test setup that was used provided a fast connection, but the link became unstable during certain periods. Whenever the wireless link was stable, the performance over this setup with respect to speed was better than the cable modem setup. The cable modem connection provided a reasonably stable throughput with few connection drops.

Client	Minimum (ms)	Maximum (ms)	Median (ms)
WLL Client	4.87	13640.03	5.39
Cable Client 1	119.92	6064.82	175.84
Cable Client 2	66.43	4016.16	84.74
Cable Client 3	53.93	1939.88	72.21
LAN Client 1	3.57	101.75	3.7
LAN Client 2	3.58	27.01	3.7

Table 4. Round trip time Comparison.

5. Conclusion and Future Work

5.1. Summary

We have developed a TCP connection monitoring tool that has been used to demonstrate the necessity and usefulness of connection monitoring for service providers. The different tests indicate some of the various circumstances under which a TCP connection can be disconnected by the end user. Some of the conclusions are as follows:

- Service providers are deploying new technologies to provide better and faster connections to the Internet. When deploying new equipment, there could be many reasons due to which the equipment may not perform up to the expected level of performance. From the end user's perspective, the connection over which the Internet is accessed must be stable and perform as expected. The process of making sure that this is possible can be achieved using ConMon.
- We have demonstrated using tests on the BWLL network that a fast connection to the Internet is achievable using this setup. Further, since the equipment that we used was at the beta stage of development, it was unstable. The results using ConMon provide the necessary information to realize the performance of the connection. Comparisons against an already existing and popular cable modem connection provide insights on stability and performance.
- Network outages and route fluctuations occur on the Internet due to various reasons. We have also shown using the cable modem network that most of the connection failures were due to these routing problems. This however cannot be generalized, since we were unable to test the connection between the customer and the cable provider.

5.2. Future Work

ConMon is capable of performing active monitoring of the TCP connection to the service provider. Some possible future work is discussed below:

- The ability to send traffic based on a certain distribution can be useful in simulating operations such as an FTP download, sending email etc.
- A mechanism that allows the detection and maintenance of host identifiers would be helpful in dealing with DHCP clients.
- A method to detect connection performance deterioration would be useful.
- Currently, the connection monitoring is done until one of the end-points (client or server) is killed. A time bound test in which the connection is monitored for a fixed period of time and then the monitoring stopped would prove to be useful.

References

1. Allman, M., Paxson, V., Stevens, R., TCP Congestion Control, RFC 2581, April 1999.
2. Amitava Dutta-Roy., Cable: it's not just for TV, IEEE Spectrum, May 1999.
3. Andrew Tridgell, Paul Mackerras., rsync, <http://rsync.samba.org>
4. Anthony R. Noerpel, Yi-Bing Lin., Wireless Local Loop: Architecture, Technologies and Services, IEEE Personal Communications, June 1998.
5. Chadi Barakat, Eitan Altman, Walid Dabbous., On TCP Performance in a heterogeneous network: A survey, IEEE Communications Magazine, January 2000.
6. Douglas E. Comer., Internetworking with TCP/IP, Volume I, Principles, Protocols, and Architecture, Prentice Hall, 3rd edition, 1997
7. Eric Wassenaar., Traceroute, <ftp://ftp.nikhef.nl/pub/network/>, May 1999
8. Ganymede Software., Qcheck, <http://www.qcheck.net>
9. Information Sciences Institute., Transmission Control Protocol, RFC 793, September 1981
10. Jacobson, V., Braden, R., Borman, D., TCP Extensions for High Performance, RFC 1323, May 1992
11. Lawrence Berkeley National Laboratory., NetLogger: A Methodology for Monitoring and Analysis of Distributed Systems, <http://www.didc.lbl.gov/NetLogger/homepage.html>
12. Mark Gates., Iperf, <http://dast.nlanr.net/Projects/Iperf>, February 2000
13. Paxson V., Measurements and Analysis of End-to-End Internet Dynamics, Ph.D. thesis, University of California, Berkeley, April 1997.
14. Rick Jones., Netperf, <http://www.netperf.org>, February 1995
15. Roelof J.T. Jonkman., Netspec: Philosophy, Design and Implementation, Masters thesis, University of Kansas, February 1998
16. Stevens R., Unix Network Programming Prentice Hall, August 1997
17. Terry Slatery, Mike Muuss., TFTP, <http://ftp.arl.mil/pub/tftp>, October 1985. Modified further at Silicon Graphics, Inc.