7 Conclusion and Recommendations

7.1 Merits and Drawbacks of the System

Measuring user-perceived performance is a key to determine end-user satisfaction. This is vital for establishments, like LEARN, who always pay attention to give the best possible service to all categories of users within budgetary limitations.

The main objective of this project is to build a tool that enables the Network Administrator to measure and compare client-side web performance without user involvements. In this thesis, we have presented a framework that can be used to measure and compare user-perceived web performance. This will be helpful to Network Administrators in fine-tuning network parameters.

7.1.1 Merits of the System

High stability

This system is based on a standard packet-sniffing program TCPDUMP; hence the stability of the system is high. This packet sniffer is widely used and has been tested in different networks around the world and most of the bugs have been fixed. The other important factor is that this packet-sniffer is constantly evolving.

Configurability

The system was designed and implemented in such a way that it is very easy to conduct measurements. And the ultimate result is in Spreadsheet Application format. So Network Administrators are free to analyze them any way they wish.

Portability

The system is completely written in Perl so that it can be easily ported to other platforms, such as Solaris, Windows NT etc. where there is a Perl interpreter and TCPDUMP packet sniffer. No extra software is needed.

Maintainability

It is easy to install and maintain because of its layered approach and modular design. Users can easily add or modify features for their own needs, since source codes are available in public. This also helps in troubleshooting the system.

7.1.2 Drawbacks of the System

Capturing IP packets

TCPDUMP is used as the packet-sniffer of this system. Whole IP packets have to be captured in order to grab all necessary information required by the system, rather than capturing only part of the packet. When the system is monitoring the traffic of single user, it will not be a problem. But it has to monitor a heavy load of traffic, TCPDUMP may dump some of the data packets that should be captured.

Availability

The system is currently tested and available only for the Linux platform. It will not require major changes in order to use this in UNIX versions such as Solaris. But using this system in Microsoft Windows will require some modifications because file-naming conventions in Windows are different from Linux.

7.2 Future Enhancements

7.2.1 Possible solutions to problems in 6.2

As discussed in section 6.2, the present system ignores effect of If-Modified-Since Headers. But this system could be developed at least to remove those situations in calculation. This will help to increase the accuracy of the ultimate results. This can be done by making changes to the coding of comparison.pl. The same thing could be applied to the involvement of Cache HIT and MISS.

It will definitely be worth, if the system is developed to support persistent connections. Because this is one of the new features added to the HTTT to enhance the performance.

7.2.2 Enhance the GUI.

This system only gives GUI to insert basic configuration settings. And the system gives the ultimate result in Spreadsheet format, keeping Network Administrators free to analyze it. But, it is possible to develop GUI for analyzing the results. This will be helpful to enhance the user-friendliness of the present system.

7.2.3 Data Validation

Although the present system does data validation to a certain extent, it is not enough. This also reduces the chances of an administrator making mistakes in employing the system. Even if he makes mistakes, it will be easy to identify and fix them.

7.2.4 Re-writing

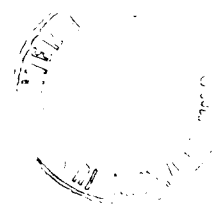
Due to the system performance issues, one should also re-write the most effective parts of the system in order to increase its performance. Use of a different programming language such as 'C' may help increase the performance of such codes.

Alternatively, one should revise the algorithms of sub-routines that have more effect on performance.

Both solutions are possible because, the system itself and all other re-used components are open source packages or programs.

7.2.5 Developed to fully support HTTP/1.1

The present system does not support some features that are available in the HTTP/1.1 version. Persistent connections and chunked transfer encoding are also among them. The present system should be developed to support those features to enhance the accuracy of the system.



Appendix A

Excerpt from User Manual of TCPDUMP

-a Attempt to convert network and broadcast addresses to names.

-c Exit after receiving count packets.

-d Dump the compiled packet-matching code in a human readable form to standard output and stop.

-e Print the link-level header on each dump line.

-F Use file as input for the filter expression. An additional expression given on the command line is ignored.

-i Listen on interface. If unspecified, tcpdump searches the system interface list for the lowest numbered, configured up interface (excluding loopback). Ties are broken by choosing the earliest match.

-l Make stdout line buffered. Useful if you want to see the data while capturing it. E.g., ``tcpdump -l | tee dat" or ``tcpdump -l > dat & tail -f dat".

-n Don't convert addresses (i.e., host addresses, port numbers, etc.) to names.

-N Don't print domain name qualification of host names. E.g., if you give this flag then tcpdump will print ``nic" instead of ``nic.ddn.mil".

-O Do not run the packet-matching code optimizer. This is useful only if you suspect a bug in the optimizer.

-p Don't put the interface into promiscuous mode. Note that the interface might be in promiscuous mode for some other reason; hence, `-p' cannot be used as an abbreviation for `ether host {local-hw-addr} or ether broadcast'.

-q Quick (quiet?) output. Print less protocol information so output lines are shorter.

-r Read packets from file (which was created with the -w option). Standard input is used if file is ``-".

-s Snarf snaplen bytes of data from each packet rather than the default of 68 (with SunOS's NIT, the minimum is actually 96). 68 bytes is adequate for IP, ICMP, TCP and UDP but may truncate protocol information from name server and NFS packets (see below). Packets truncated because of a limited snapshot are indicated in the output with ``[|proto]", where proto is the name of the

protocol level at which the truncation has occurred. Note that taking larger snapshots both increases the amount of time it takes to process packets and, effectively, decreases the amount of packet buffering. This may cause packets to be lost. You should limit snaplen to the smallest number that will capture the protocol information you're interested in.

-S Print absolute, rather than relative, TCP sequence numbers.

-t Don't print a timestamp on each dump line.

-tt Print an unformatted timestamp on each dump line.

-v (Slightly more) verbose output. For example, the time to live and type of service information in an IP packet is printed.

-vv Even more verbose output. For example, additional fields are printed from NFS reply packets.

-w Write the raw packets to file rather than parsing and printing them out. They can later be printed with the -r option. Standard output is used if file is ``-".

-x Print each packet (minus its link level header) in hex. The smaller of the entire packet or snaplen bytes will be printed.

When TCPDUMP are employed to dump network traffic, expressions will be used to select which packets will be dumped. If no expression is given, all packets on the net will be dumped. Otherwise, only packets for which expression is `true' will be dumped.

The expression consists of one or more primitives. Primitives usually consist of an id (name or number) preceded by one or more qualifiers. There are three different kinds of qualifier:

type: qualifiers say what kind of thing the id name or number refers to. Possible types are host, net and port. E.g., `host foo', `net 128.3', `port 20'. If there is no type qualifier, host is assumed.

dir: qualifiers specify a particular transfer direction to and/or from id. Possible directions are src, dst, src or dst and src and dst. E.g., `src foo', `dst net 128.3', `src or dst port ftp-data'. If there is no dir qualifier, src or dst is assumed. For `null' link layers (i.e. point to point protocols such as slip) the inbound and outbound qualifiers can be used to specify a desired direction.

proto: qualifiers restrict the match to a particular protocol. Possible protos are: ether, fddi, ip, arp, rarp, decnet, lat, sca, moprc, mopdl, tcp and udp. E.g., `ether src foo', `arp net 128.3', `tcp port 21'. If there is no proto qualifier, all protocols consistent with the type are assumed. E.g., `src foo' means `(ip or arp or rarp) src foo' (except the latter is not legal syntax), `net bar' means `(ip or arp or rarp) net bar' and `port 53' means `(tcp or udp) port 53'.

More complex filter expressions are built up by using the words and, or and not to combine primitives. E.g., `host foo and not port ftp and not port ftp-data'. To save typing, identical qualifier lists can be omitted. E.g., `tcp dst port ftp or ftp-data or domain' is exactly the same as `tcp dst port ftp or tcp dst port ftp-data or tcp dst port domain'.

Primitives may be combined using:
A parenthesized group of primitives and operators (parentheses are special to the Shell and must be escaped).

Negation (`!' or `not').
Concatenation (`&&' or `and').

Alternation (`||' or `or').

Negation has highest precedence. Alternation and concatenation have equal precedence and associate left to right. Note that explicit and tokens, not juxtaposition, are now required for concatenation.

Expression arguments can be passed to TCPDUMP as either a single argument or as multiple arguments, whichever is more convenient. Generally, if the expression contains Shell metacharacters, it is easier to pass it as a single, quoted argument. Multiple arguments are concatenated with spaces before being parsed.

Output Format

The output of tcpdump is protocol dependent. The following gives a brief description of output of TCP packets.

The general format of a tcp protocol line is:
src > dst: flags data-seqno ack window urgent options

Src and dst are the source and destination IP addresses and ports. Flags are some combination of S (SYN), F (FIN), P (PUSH) or R (RST) or a single `.' (no flags). Data-seqno describes the portion of sequence space covered by the data in this packet (see example below). Ack is sequence number of the next data expected the other direction on this connection. Window is the number of bytes of receive buffer space available the other direction on this connection. Urg indicates there is `urgent' data in the packet. Options are tcp options enclosed in angle brackets (e.g., <mss 1024>).

Src, dst and flags are always present. The other fields depend on the contents of the packet's tcp protocol header and are output only if appropriate.

Here is the opening portion of an rlogin from host rtsg to host csam.

rtsg.1023 > csam.login: S 768512:768512(0) win 4096 <mss 1024>
csam.login > rtsg.1023: S 947648:947648(0) ack 768513 win 4096 <mss 1024>
rtsg.1023 > csam.login: . ack 1 win 4096
rtsg.1023 > csam.login: P 1:2(1) ack 1 win 4096

csam.login > rtsg.1023: . ack 2 win 4096
rtsg.1023 > csam.login: P 2:21(19) ack 1 win 4096
csam.login > rtsg.1023: P 1:2(1) ack 21 win 4077
csam.login > rtsg.1023: P 2:3(1) ack 21 win 4077 urg 1
csam.login > rtsg.1023: P 3:4(1) ack 21 win 4077 urg 1

The first line says that tcp port 1023 on rtsg sent a packet to port login on csam. The S indicates that the SYN flag was set. The packet sequence number was 768512 and it contained no data. (The notation is `first:last(nbytes)' which means `sequence numbers first up to but not including last which is nbytes bytes of user data'.) There was no piggy-backed ack, the available receive window was 4096 bytes and there was a max-segment-size option requesting an mss of 1024 bytes.

Csam replies with a similar packet except it includes a piggy-backed ack for rtsg's SYN. Rtsg then acks csam's SYN. The `.' means no flags were set. The packet contained no data so there is no data sequence number. Note that the ack sequence number is a small integer (1). The first time TCPDUMP sees a tcp `conversation', it prints the sequence number from the packet. On subsequent packets of the conversation, the difference between the current packet's sequence number and this initial sequence number is printed. This means that sequence numbers after the first can be interpreted as relative byte positions in the conversation's data stream (with the first data byte each direction being `1'). `-S' will override this feature, causing the original sequence numbers to be output.

On the 6th line, rtsg sends csam 19 bytes of data (bytes 2 through 20 in the rtsg -> csam side of the conversation). The PUSH flag is set in the packet. On the 7th line, csam says it's received data sent by rtsg up to but not including byte 21. Most of this data is apparently sitting in the socket buffer since csam's receive window has gotten 19 bytes smaller. Csam also sends one byte of data to rtsg in this packet. On the 8th and 9th lines, csam sends two bytes of urgent, pushed data to rtsg.

If the snapshot was small enough that TCPDUMP didn't capture the full TCP header, it interprets as much of the header as it can and then reports ``[|tcp]" to indicate the remainder could not be interpreted. If the header contains a bogus option (one with a length that's either too small or beyond the end of the header), TCPDUMP reports it as ``[bad opt]" and does not interpret any further options (since it's impossible to tell where they start). If the header length indicates options are present but the IP datagram length is not long enough for the options to actually be there, TCPDUMP reports it as ``[bad hdr length]" [11].

References:

[1]. Lanka Education and Research Network, http://www.learn.ac.lk [31/03/2004]

[2]. P. Pushpakumara and Gihan Dias, "LEARNStat: A Network Traffic Monitoring Utility", Proceedings of the 20th National Information Technology Conference, Colombo, Sri Lanka, 2001

[3]. Gihan V Dias, "Managing Internet Bandwidth: the LEARN Experience", INET 2001, Stockholm, Sweden, June 2001.

[4]. Sidath Bandara and Gihan Dias, "A Smart Cache for a Central Off-line Download System," Proceedings of the 5th Research for Industry Symposium, University of Moratuwa, 1999.

[5]. Gihan Dias and Chamara Guneratne, "Using Dynamic Delay Pools for Bandwidth Management", Proceedings of the Seventh International Web Caching and Distribution Workshop, Boulder, Colorado, August 2002.

[6]. M.S.D. Fernando and Gihan V. Dias, "An Architecture for Advanced Proxying Based on User Requests", Proceedings of the 8th Research for Industry Symposium, University of Moratuwa, August 2002.

[7]. R. Rajamony and M. Elnozahy, "Measuring Client-Perceived Response Times on the WWW", Proceedings of the 3 rd USENIX Symposium on Internet Technologies and Systems, March 2001.

[8]. R.Liston and E.Zegura, "Using a proxy to measure client-side web performance", Proceedings of the 6th Int. workshop on Web Caching and Content Distribution, June 2001.

[9]. M. Koletsu and G.M. Voelker, "The Medusa Proxy: A tool for exploring user-perceived web performance", In the Sixth International Workshop on Web Caching and Content Distribution, Boston, MA 2001.

[10]. Leeann Bent and Geoffrey M. Voelker, "Whole Page Performance", Proceedings of the Seventh International Web Caching and Distribution Workshop, Boulder, Colorado, August 2002.

[11]. TCPDUMP, http://www.tcpdump.org/ [15/03/2004]

[12]. R. Fielding, J. Gettys, J.C. Mogul, H. Frystyk, L. Masinter, P. Leach, T. Berners-Lee, "Hypertext Transfer Protocol—HTTP/1.1", RFC 2616, June 1999.

[13]. Paul S. Hethmon, "Illustrated Guide to HTTP", Manning Publications Co. 1997.

[14]. D.E. Comer, Internetworking with TCP/IP, Vol. 1: Principles, Protocols, and Architecture. Prentice-Hall, 1995.

[15]. Stallings, William. (1993) "*SNMP, SNMPv2, and CMIP*" Massachusetts: Addison-Wesley.

[16]. Stevens, W. Richard. (1993) "*TCP/IP Illustrated, Volume 1- The Protocols*" Singapore: Addison Wesley Longman.

[17]. Tanenbaum, Andrew S. (1996) "*Computer Networks, 3rd Edition*" New Jersey: Prentice-Hall, Inc.

[18]. The Medusa Proxy http://ramp.ucsd.edu/medusa/ [15/02/2003]

[19]. Gnuplot, http://www.gnuplot.info/ [15/02/2003]

[20]. Gnuplot Manual, http://www.ucc.ie/gnuplot/gnuplot.html [15/02/2003]

[21]. Squid web proxy cache server, http://www.squid-cache.org/ [15/02/2003]