Application and TCP measurements

TkL Markus Peuhkuri

2007-04-04

Lecture topics

- What to measure in applications
- Application traffic analysis
- Protocol analysis
  - RTP / RTCP
  - TCP
  - how about secure encapsulation
- Host-based diagnostics
- After this lecture you should know how to
  - do application-specific measurements
  - extract quality information from protocol headers
  - analyse application logs

What can be measured from a network

- QoS performance
- Applications used
- Protocol extensions used
- Protocol parameters
- Protocol and implementation anomalies
- Implementation and devices used

What is important for applications

- Throughput
  - file and document transfers
  - should get a fair share of resources
- Delay
  - interactive or real-time applications
  - maximum upper bound for delay (or for some fraction of traffic)
- Loss
  - packet loss results a loss of application fidelity
- Applications can have complex interrelations between these
  ⇒ must consider carefully before concluding

How to test for throughput

- Just transfer a large file
  - `time wget http://site.example/latest.iso`
- Benefits
  - easy to do and analyse
  - 640 MiB in 3701 seconds ⇒1,45 Mbit/s
- Problems
  - depends on other systems and network
    ⇒ tells very little on *network*
  - gives only present performance for *additional* traffic
  - results additional load on network
  - depends on TCP implementations used
    * Reno, Vegas, BIC, Westwood, ...
    * window size
  - does not pinpoint problem locations

Throughput: flow-based passive

- Use flow data: eg NetFlow
  - calculate average throughput
- Benefits
  - may be readily available
  - continuous measurements
- Problems
  - many flows have lulls, either receiver or sender initiated [18]
- A possible algorithm
  1. select only the largest flows
  2. group by network topology i.e. either by sender or receiver
  3. build throughput histogram possibly grouping multi-day measurements by time of day

Throughput: packet-based passive

- Capture packets, group by flows
- Determine if a flow is
  - host-limited
  - network-limited
  - by analysing TCP headers (see page 4)
- Benefits
  - one is able to select only bulk transfers
  - extract bulk transfer periods
- Problems
  - a large amount of data
  - additional hardware

Active measurements classes

SO Sender Only measurements depend on standard functionality

- ICMP echos and diagnostic methods
- depends on other system functioning properly

SRP Sender and Receiver Paired measurements

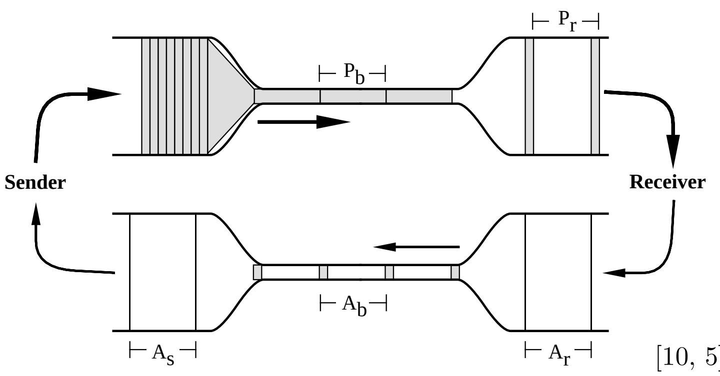
- possible to use measurement specific packets
- accurate time stamps, sequence numbers

RO Recipient Only: most limited functionality

- depends the sender to behave as expected
- packet-pair sending
- passive analysis

Throughput: active probes

- Try to estimate
  - bottleneck capacity
  - available bandwidth
- Packet pairs (trains)
- Estimate TCP capacity
  - loss process
  - maximum delay
  - optimum window size



[10, 5]

Delay measurements

- Easy to do active RTT
  - no need to synchronise clocks
  - ICMP echo
  - TCP handshake
  - UDP response
  - take end system delay into account
- One-way delays more difficult
  - software support
  - clock synchronisation or clock skew estimation
- Take account possible classification
- Low bandwidth requirements
  - get sufficient number of samples

Passive delay measurements

- Packet-level measurements
- TCP
  - time difference between data and corresponding ACK
  - ack may be lost too
  - end system characteristics[14]
- RTP [17]
  - has timestamp for samples: if there is standard PCM voice, the timestamp counter is incremented by 8000 every second
  - if monitored on far end, delay variation can be identified
  - note possible clock skew
- Flow-based analysis possible
  - short flows
  - 1st packet of flow

When an average is not the perceived average

- Example: measure delay by sending one packet every second
  - you get 86400 measurement samples for delay each day
  - average over samples 75 ms
    ⇒ network ok for VoIP?
- However, users complain that there is quite lot of delay
- Performance problems in 9 – 15 peak use
  - one quarter of samples from that period of time: average 200 ms
  - off-peak samples have average of 33 ms
- Performance must be weighted by use

Loss measurements

- Active similarly to delay measurements
- Passive measurements
  - TCP retransmissions
  - RTP sequence number monitoring
  - RTCP receiver reports

TCP header analysis

- Sequence numbers and ack numbers
- TCP options used
- Interesting analysis
  - delay
  - throughput received
  - spurious retransmits: note that even if we see duplicate segments at our measurement point, the segment may be lost between us and host

– other implementation problems [14]

<table>
<tr><td>0</td><td colspan="9"></td><td>1</td><td colspan="9"></td><td>2</td><td colspan="9"></td><td>3</td><td></td></tr>
<tr><td>0</td><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td><td>6</td><td>7</td><td>8</td><td>9</td><td>0</td><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td><td>6</td><td>7</td><td>8</td><td>9</td><td>0</td><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td><td>6</td><td>7</td><td>8</td><td>9</td><td>0</td><td>1</td></tr>
<tr><td colspan="16">Source Port</td><td colspan="16">Destination Port</td></tr>
<tr><td colspan="32">Sequence number</td></tr>
<tr><td colspan="32">Acknowledgment Number</td></tr>
<tr><td colspan="4">Data Offset</td><td colspan="6">Reserved</td><td>URG</td><td>ACK</td><td>PSH</td><td>RST</td><td>SYN</td><td>FIN</td><td colspan="16">Window</td></tr>
<tr><td colspan="16">Checksum</td><td colspan="16">Urgent Pointer</td></tr>
<tr><td colspan="24">Options</td><td colspan="8">Padding</td></tr>
<tr><td colspan="32">payload</td></tr>
</table>

TCP RTT delay

- Every data sent should be acknowledged
- Pair sent sequence numbers to received acknowledgement numbers. Make note that the sequence number indicates the first octet while corresponding acknowledgement number is the sequence number of one following *last* octet. Thus, if seq=1000 and segment has 500 bytes, then ack=1501.
- Measure the time between
- Some caveats
  – not every segment is acknowledged
  – ack may not be immediate as the receiver may wait for additional packets to arrive or the end system may be busy
  – 1st ack you see may not be the first ack sent
  – asymmetric routing
  – normal passive measurements analysis

1-way TCP loss analysis

- Problem: determine if there are lost packets in TCP connections without keeping full TCP state
- Get a rough number: count only retransmissions
  1. for each packet received, check if it is part of existing flow
  2. check if sequence number is less than in one before (beware seq number wrap)
  3. if so, count as retransmission
- One gets some lower estimate for packet loss
- Spurious retransmits may be an issue

RTP header analysis

- RTP provides synchronisation of different media
- Identifier of different senders
- Sequence numbers provide message ordering
  – one can identify on-wire if some packets are lost
- Delay estimation using timestamps
  – one must know timestamp rate

0										1										2										3	
0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1

V=2	P	X	CC	M	PT	sequence number
timestamp						
SSRC						
CSRC list						
extension header, data						

RTCP report analysis

- Provides statistic information (uses maximum of 5 % data rate)
- Sender reports
  - NTP timestamp and RTP timestamp mapping
  - sent packets and payload bytes
- Reception reports for each source
  - fraction lost (indicated as 1/256 units)
  - cumulative packets lost that is calculated as difference between expected number of packets received and actual number of packets received. Duplicate or late packets are not counted as late.
  - highest sequence number received as 32-bit extended number
  - jitter value

$$J(i) = J(i-1) + (|D(i-1,i)| - J(i-1))/16 \tag{1}$$

$$D(i,j) = (R_j - R_i) - (S_j - S_i) = (R_j - S_j) - (R_i - S_i) \tag{2}$$

    where S_i is RTP timestamp in packet i and R_i is arrival time (in timestamp units).

  - information about last SR: timestamp and 1/65536 seconds from the last one

Voice performance

- Human tests: MOS (Mean Opinion Score)
  - a set of test users
  - for each test, a grade $1-5$ is given
    1. bad
    2. poor
    3. fair
    4. good
    5. excellent
  - expensive and time-consuming
  - codecs and systems language-dependent. There are significant differences between both languages and genders how well each compression method performs.
- Automated tests
  - characterise network performance
  - estimate MOS based on those parameters

E-model

- A computational model for use in transmission planning [9]
- Takes a set of parameters

  R_o basic signal-to-noise ratio

  I_s simultaneous impairment factors

  I_d delay impairment factors

  I_e equipment impairment factors, for example voice codec and bit rate used has an effect on here. PCM at 64 kbit/s has value of 0 while GSM full-rate codec has value 20 (half-rate 23)

  A advantage factor to take into account user's expectations $(0-20)$, results approximately MOS difference of one unit

$$R = R_o - I_s - I_d - I_e + A \tag{3}$$

E-model, MOS, GoB, PoW

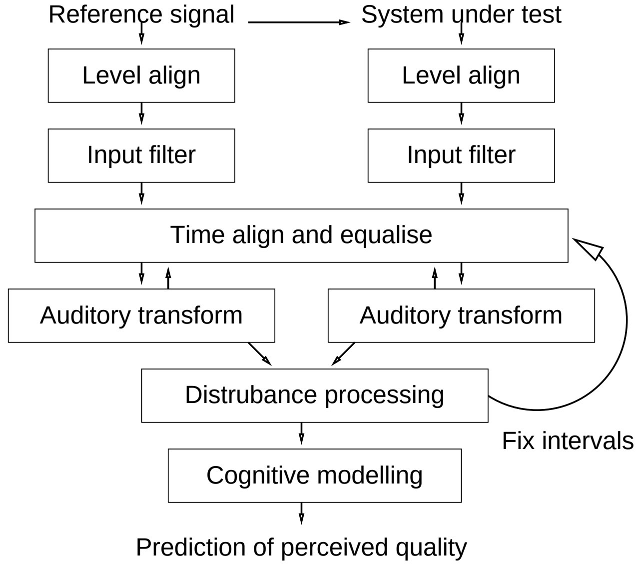
R	MOS	GoB	PoW	Users
≥ 100	4.5			(maximum)
90	4.34	97	≈ 0	Very satisfied
80	4.03	89	≈ 0	Satisfied
70	3.60	73	6	Some dissatisfied
60	3.10	50	17	Many dissatisfied
50	2.58	27	38	Nearly all dissatisfied
≤ 0	1			(minimum)

GoB Good or Better, the percentage of users who think connection to be better than reference connection

PoW Poor or Worse

Algorithm-based VoIP measurements

- PESQ[6] most appropriate for VoIP
  - older: PSQM, PSQM+,
  - errors are different in VoIP than GSM or PSTN
- Network measurements
  - RTP timestamps, sequence numbers and reports from live traffic
  - use active measurements tools to estimate parameters
- Feed network-affected voice to analysis



IP Performance Metrics (ippm) [15]

- IETF working group developing a set of standard metrics for Internet data delivery services
  - quality
  - performance
  - reliability
- Can be used by all parties: network operators, end users, or independent testing groups
- Metrics defined:
  - connectivity [12]
  - one-way delay and loss [1, 2]
  - round-trip delay and loss [3]
  - delay variation [7]
  - loss patterns [11]
  - packet reordering
  - bulk transport capacity [13, 16]
  - link bandwidth capacity

The IPPM WG will develop a set of standard metrics that can be applied to the quality, performance, and reliability of Internet data delivery services. These metrics will be designed such that they can be performed by network operators, end users, or independent testing groups. It is important that the metrics not represent a value judgement (i.e. define "good" and "bad"), but rather provide unbiased quantitative measures of performance.

Flow data

- Cisco has used Netflow export format
  - incompatibles between vendors
- IPFIX (IP Flow Information Export)
  - based on Netflow v9
  - specification mostly done

Accounting and AAA information

- Accounting systems collect information about network traffic
  - CRANE [19]
- AAA systems
  - Diameter [4]
- Highly aggregate data
  - total bytes, packets
  - can be used to estimate traffic demand

Protected data: IPSec

- Is it possible to conclude anything about those
- In general: no
- It may be possible to conclude something
  - traffic volume
  - single-application VPN characteristics

Non-network measurements

- Network application logs
  - http servers
    * client IP address and model
    * document size
    * date, transfer time
    * request correlation

    ```
    72.30.110.140 - - [06/Apr/2006:07:58:46 +0300] "GET /korso2005/ HTTP/1.0" 200
    737 "-" "Mozilla/5.0"
    ```
  - mail servers
    * message sizes
    * service times: some email servers currently wait some time before accepting email to identify some spammer software. Also there may be delay resulting from blacklist lookups etc.
  - ftp servers
- Response time for application, for example to monitor database server; includes both network and application delays. These can be used as part of SLA verification tools, especially if "whole service" (i.e. both the network and the server) is provided by one service provider.
- Mostly appropriate for estimating traffic demand

Application performance

- In addition to data transmission QoS
- Call setup time
  - PDD (Post Dialling Delay) [8]
- Channel change time for IPTV
- System responsiveness
- These are best measured on end systems
  - instrumented application
  - test equipment

Statistics from end systems

- End systems collect protocol statistics
  - OS dependent
  - counter wrap
- Provides indication of network quality
  - TCP retransmits
  - TCP reorders

Small system, low traffic

```
Ip:
    48967 total packets received
    0 forwarded
    0 incoming packets discarded
    48831 incoming packets delivered
    33700 requests sent out
Icmp:
    9 ICMP messages received
    0 input ICMP message failed.
    ICMP input histogram:
        destination unreachable: 9
    0 ICMP messages sent
    0 ICMP messages failed
    ICMP output histogram:
```

Small system, low traffic

```
Tcp:
    192 active connections openings
    16 passive connection openings
    0 failed connection attempts
    9 connection resets received
    7 connections established
    48253 segments received 232 segments / connection
    33257 segments send out 160 segments / connection
    13 segments retransmited 0.04 % retransmits
    0 bad segments received.
    60 resets sent
Udp:
    434 packets received
    0 packets to unknown port received.
    0 packet receive errors
    442 packets sent
```

Small system, low traffic

```
TcpExt:
    12 packets pruned from receive queue because of socket buffer overrun
    40 TCP sockets finished time wait in fast timer
    1356 delayed acks sent
    Quick ack mode was activated 23 times
    9 packets directly queued to recvmsg prequeue.
    535 of bytes directly received from backlog
    60 of bytes directly received from prequeue
    37205 packet headers predicted
    3 packets header predicted and directly queued to user
    453 acknowledgments not containing data received
```

```
    312 predicted acknowledgments
    1 congestion windows recovered after partial ack
    0 TCP data loss events
    5 other TCP timeouts
    341 packets collapsed in receive queue due to low socket buffer
    6 connections reset due to unexpected data
    6 connections reset due to early user close
    1 connections aborted due to timeout
```

Busy system, Linux

```
Ip:
    49526913 total packets received
    9316140 forwarded
    0 incoming packets discarded
    37552025 incoming packets delivered
    55090221 requests sent out
    2952 outgoing packets dropped
    10 fragments dropped after timeout
    52771775 reassemblies required
    5966080 packets reassembled ok
    152 packet reassembles failed
    11693318 fragments received ok
```

Busy system, Linux

```
Icmp:
    259266 ICMP messages received
    463 input ICMP message failed.
    ICMP input histogram:
        destination unreachable: 37001
        timeout in transit: 307
        source quenches: 2
        echo requests: 221591
        echo replies: 3
    463977 ICMP messages sent
    0 ICMP messages failed
    ICMP output histogram:
        destination unreachable: 236219
        time exceeded: 72
        redirect: 6095
        echo replies: 221591
```

Busy system, Linux

```
Tcp:
    41926 active connections openings
    650042 passive connection openings
    24499 failed connection attempts
    41515 connection resets received
    15 connections established
    27207697 segments received 39 segments / connection
    35352653 segments send out 51 segments / connection
    107436 segments retransmited 3% restansmits
    2851 bad segments received.
    56397 resets sent
Udp:
    9925464 packets received
    144656 packets to unknown port received.
    402 packet receive errors
    32968468 packets sent
```

Busy system

```
TcpExt:
    267353 resets received for embryonic SYN_RECV sockets
    45 ICMP packets dropped because they were out-of-window
    77499 TCP sockets finished time wait in fast timer
    3 time wait sockets recycled by time stamp
    36 packets rejects in established connections because of timestamp
    385649 delayed acks sent
    2925 delayed acks further delayed because of locked socket
    Quick ack mode was activated 13198 times
    646595 packets directly queued to recvmsg prequeue.
    3271571 of bytes directly received from backlog
    549815762 of bytes directly received from prequeue
    5340998 packet headers predicted
    401429 packets header predicted and directly queued to user
    2676410 acknowledgments not containing data received
    14962075 predicted acknowledgments
    127 times recovered from packet loss due to fast retransmit
    10782 times recovered from packet loss due to SACK data
    Detected reordering 20 times using reno fast retransmit
```

Busy system

```
    TCPDSACKUndo: 12
    4141 congestion windows recovered after partial ack
    9406 TCP data loss events
    TCPLostRetransmit: 1
    117 timeouts after reno fast retransmit
    4379 timeouts after SACK recovery
    337 timeouts in loss state
    23084 fast retransmits
    489 forward retransmits
    4864 retransmits in slow start
    52291 other TCP timeouts
    TCPRenoRecoveryFail: 37
    863 sack retransmits failed
    367 times receiver scheduled too late for direct processing
    16671 DSACKs sent for old packets
    1482 DSACKs sent for out of order packets
    2845 DSACKs received
    900 connections reset due to unexpected data
    615 connections reset due to early user close
    2366 connections aborted due to timeout
```

Busy system, Solaris

```
UDP
udpInDatagrams      =116690321 udpInErrors        =     0
udpOutDatagrams     =126637248

TCP tcpRtoAlgorithm     =     4 tcpRtoMin           =   400
tcpRtoMax           = 60000 tcpMaxConn          =    -1
tcpActiveOpens      =11063565 tcpPassiveOpens     =8857655
tcpAttemptFails     =5605680 tcpEstabResets      =333124
tcpCurrEstab        =   284 tcpOutSegs          =2433610475
tcpOutDataSegs      =1829582137 tcpOutDataBytes     =4291730024
tcpRetransSegs      =4618387 tcpRetransBytes     =4194709197
tcpOutAck           =603500275 tcpOutAckDelayed    =20033072
tcpOutUrg           =     0 tcpOutWinUpdate     =136839
```

```
tcpOutWinProbe      =151290 tcpOutControl       =35759699
tcpOutRsts          =864119 tcpOutFastRetrans   =542751
```

Busy system, Solaris

```
tcpInSegs           =2508775648
tcpInAckSegs        =1214188950 tcpInAckBytes       =2161643669
tcpInDupAck         =26633792 tcpInAckUnsent      =     0
tcpInInorderSegs    =1567697694 tcpInInorderBytes   =735960060
tcpInUnorderSegs    =181743 tcpInUnorderBytes   =154652942
tcpInDupSegs        =589996 tcpInDupBytes       =36748252
tcpInPartDupSegs    =  4194 tcpInPartDupBytes   =1802910
tcpInPastWinSegs    =  1693 tcpInPastWinBytes   =122699007
tcpInWinProbe       =  5726 tcpInWinUpdate      =132144
tcpInClosed         = 82623 tcpRttNoUpdate      =1365731
tcpRttUpdate        =1201346033 tcpTimRetrans       =5016742
tcpTimRetransDrop   = 69051 tcpTimKeepalive     = 60025
tcpTimKeepaliveProbe=  8128 tcpTimKeepaliveDrop =    34
tcpListenDrop       =256703 tcpListenDropQ0     =     0
tcpHalfOpenDrop     =     0 tcpOutSackRetrans   =988463
```

Busy system, Solaris

```
IP ipForwarding        =     2 ipDefaultTTL        =   255
ipInReceives        =2477584094 ipInHdrErrors       =     0
ipInAddrErrors      =     0 ipInCksumErrs       =     0
ipForwDatagrams     =     0 ipForwProhibits     =    33
ipInUnknownProtos   =    64 ipInDiscards        =     2
ipInDelivers        =2617639644 ipOutRequests       =2455379708
ipOutDiscards       =   183 ipOutNoRoutes       =     0
ipReasmTimeout      =    60 ipReasmReqds        = 10802
ipReasmOKs          = 10748 ipReasmFails        =    54
ipReasmDuplicates   =     3 ipReasmPartDups     =     0
ipFragOKs           =3626344 ipFragFails         =     1
ipFragCreates       =76043592 ipRoutingDiscards   =     0
tcpInErrs           =  4166 udpNoPorts          =4498522
udpInCksumErrs      =  1092 udpInOverflows      = 34219
rawipInOverflows    =     0
```

Busy system, Solaris

```
ICMP icmpInMsgs          =3320175 icmpInErrors        =     0
icmpInCksumErrs     =  1753 icmpInUnknowns      =     0
icmpInDestUnreachs  =597470 icmpInTimeExcds     =220863
icmpInParmProbs     =     0 icmpInSrcQuenchs    =     0
icmpInRedirects     = 35481 icmpInBadRedirects  = 35481
icmpInEchos         =2464564 icmpInEchoReps      =    44
icmpInTimestamps    =     0 icmpInTimestampReps =     0
icmpInAddrMasks     =     0 icmpInAddrMaskReps  =     0
icmpInFragNeeded    =   567 icmpOutMsgs         =2937414
icmpOutDrops        =    90 icmpOutErrors       =     0
icmpOutDestUnreachs =472841 icmpOutTimeExcds    =     9
icmpOutParmProbs    =     0 icmpOutSrcQuenchs   =     0
icmpOutRedirects    =     0 icmpOutEchos        =     0
icmpOutEchoReps     =2464564 icmpOutTimestamps   =     0
icmpOutTimestampReps=     0 icmpOutAddrMasks    =     0
icmpOutAddrMaskReps =     0 icmpOutFragNeeded   =     1
icmpInOverflows     =     0
```

Busy system, Solaris

```
IGMP:
     222062 messages received
          0 messages received with too few bytes
          0 messages received with bad checksum
     222014 membership queries received
          0 membership queries received with invalid field(s)
         32 membership reports received
          0 membership reports received with invalid field(s)
         32 membership reports received for groups to which we belong
         48 membership reports sent
```

Conclusion

- Possible to estimate application throughput using network measurements
- Applications can collect performance data
- Perceived quality estimation
  - Quality of Experience
  - data, voice, video quality

References

[1] G. Almes, S. Kalidindi, and M. Zekauskas. A One-way Delay Metric for IPPM. Request for Comments RFC 2679, Internet Engineering Task Force, September 1999. (Internet Proposed Standard). URL:`http://www.ietf.org/rfc/rfc2679.txt`.

[2] G. Almes, S. Kalidindi, and M. Zekauskas. A One-way Packet Loss Metric for IPPM. Request for Comments RFC 2680, Internet Engineering Task Force, September 1999. (Internet Proposed Standard). URL:`http://www.ietf.org/rfc/rfc2680.txt`.

[3] G. Almes, S. Kalidindi, and M. Zekauskas. A Round-trip Delay Metric for IPPM. Request for Comments RFC 2681, Internet Engineering Task Force, September 1999. (Internet Proposed Standard). URL:`http://www.ietf.org/rfc/rfc2681.txt`.

[4] P. Calhoun, J. Loughney, E. Guttman, G. Zorn, and J. Arkko. Diameter Base Protocol. Request for Comments RFC 3588, Internet Engineering Task Force, September 2003. (Internet Proposed Standard). URL:`http://www.ietf.org/rfc/rfc3588.txt`.

[5] Robert L. Carter and Mark E. Crovella. Measuring bottleneck link speed in packet-switched networks. *Performance Evaluation*, 27&28:297–318, 1996.

[6] Adrian E. Conway and Yali Zhu. A simulation-based methodology and tool for automating the modeling and analysis of voice-over-IP perceptual quality. *Performance Evaluation 54 (2003) 129 147*, 54(2):129–147, October 2003.

[7] C. Demichelis and P. Chimento. IP Packet Delay Variation Metric for IP Performance Metrics (IPPM). Request for Comments RFC 3393, Internet Engineering Task Force, November 2002. (Internet Proposed Standard). URL:`http://www.ietf.org/rfc/rfc3393.txt`.

[8] Service quality assessment for connection set-up and release delays. ITU-T Recommendation E.431, International Telecommunication Union, 1992.

[9] The e-model, a computational model for use in transmission planning. ITU-T Recommendation G.107, International Telecommunication Union, 2000.

[10] Van Jacobson. Pathchar: How to infer the characteristics of internet paths. Lecture at Mathematical Sciences Research Institute, April 1997. URL:`ftp://ftp.ee.lbl.gov/pathchar/msri-talk.pdf`.

[11] R. Koodli and R. Ravikanth. One-way Loss Pattern Sample Metrics. Request for Comments RFC 3357, Internet Engineering Task Force, August 2002. (Informational). URL:`http://www.ietf.org/rfc/rfc3357.txt`.

[12] J. Mahdavi and V. Paxson. IPPM Metrics for Measuring Connectivity. Request for Comments RFC 2678, Internet Engineering Task Force, September 1999. (Internet Proposed Standard) (Obsoletes RFC2498). URL:`http://www.ietf.org/rfc/rfc2678.txt`.

[13] M. Mathis and M. Allman. A Framework for Defining Empirical Bulk Transfer Capacity Metrics. Request for Comments RFC 3148, Internet Engineering Task Force, July 2001. (Informational). URL:`http://www.ietf.org/rfc/rfc3148.txt`.

[14] V. Paxson, M. Allman, S. Dawson, W. Fenner, J. Griner, I. Heavens, K. Lahey, J. Semke, and B. Volz. Known TCP Implementation Problems. Request for Comments RFC 2525, Internet Engineering Task Force, March 1999. (Informational). URL:`http://www.ietf.org/rfc/rfc2525.txt`.

[15] V. Paxson, G. Almes, J. Mahdavi, and M. Mathis. Framework for IP Performance Metrics. Request for Comments RFC 2330, Internet Engineering Task Force, May 1998. (Informational). URL:`http://www.ietf.org/rfc/rfc2330.txt`.

[16] V. Raisanen, G. Grotefeld, and A. Morton. Network performance measurement with periodic streams. Request for Comments RFC 3432, Internet Engineering Task Force, November 2002. (Internet Proposed Standard). URL:`http://www.ietf.org/rfc/rfc3432.txt`.

[17] H. Schulzrinne, S. Casner, R. Frederick, and V. Jacobson. RTP: A Transport Protocol for Real-Time Applications. Request for Comments RFC 3550, Internet Engineering Task Force, July 2003. (Internet Standard) (Obsoletes RFC1889) (Also STD0064). URL:`http://www.ietf.org/rfc/rfc3550.txt`.

[18] Matti Siekkinen, Guillaume Urvoy-Keller, Ernst W Biersack, and Taoufik En-Najjary. Root cause analysis for long-lived TCP connections. In *Co-NEXT 2005, 1st ACM/e-NEXT International Conference on Future Networking Technologies, 24-27 October, 2005, Toulouse, France*, October 2005.

[19] K. Zhang and E. Elkin. XACCT's Common Reliable Accounting for Network Element (CRANE) Protocol Specification Version 1.0. Request for Comments RFC 3423, Internet Engineering Task Force, November 2002. (Informational). URL:`http://www.ietf.org/rfc/rfc3423.txt`.