International Journal of Hybrid Information Technology
Vol. 1, No. 2, April, 2008

Differentiating Congestive and Non-congestive Losses on TCP Endpoint

Nanjun Li
Hasso Plattner Institute at the University of Potsdam
nanjun.li@hpi.uni-potsdam.de

Werner Zorn
Hasso Plattner Institute at the University of Potsdam
zorn@hpi.uni-potsdam.de

Abstract

Packet loss in IP networks can be congestive due to IP interrupt queue overflow on the nodes, and non-congestive due to hardware failure, signal-fading or obstacles. Recognition of two types of losses can largely help TCP endpoints in making right decisions. In this paper we apply the Visualized IP-based Network Simulator (VINS) to characterize the key differences between congestive packet losses and that of due to link disruption, which is non-congestive and commonly seen in wireless networks. Based on the analyses, suggested improvements for wireless TCP endpoints are proposed on both kernel and application levels in terms of avoiding useless keep-alive probing and congestion window tuning, but re-routing and re-establishing new connections.

1. Introduction

Congestive and non-congestive packet losses are caused by different factors and can impact the performance of TCP endpoints differently. In wired networks the majority of packet losses are congestive because of the limited capacity of IP interrupt queue on the routers in processing intensive packet arrival, while the losses on links are rare as links are rather stable and are not the bottleneck of end-to-end communications [1]. In wireless networks, packet losses on links can be massive due to link disruption, signal fading or obstacles between wireless agents. Recognition of these two types of losses can help TCP endpoints setting up more efficient strategies in dealing with packet loss. For congestive loss, tuning the congestion window to avoid further congestive loss is a well-known scheme in many TCP implementations; while on persistent link disruption, it might be expected to re-route the network to find a new path to its peer and re-establish the connection, instead of adjusting congestion window.

Among some existing simulation tools, packet losses on links and nodes are not well-distinguished. In Section 2 we briefly review the gap between the simulation done with these tools and the case in real networks. Section 3 introduces the recent contribution of Visualized IP-based Network Simulator (VINS, [2]), which is a discrete event tool re-implementing TCP/IP stack in the user space closely matching relevant RFCs, providing more detailed statistics and runtime visualization. In Section 4 we set up a scenario and apply VINS to analyze the loses in wired and wireless networks, characterizing the key distinction between the two types of losses by logging the entire TCP control block on each packet/segment arrival and departure. In Section 5, suggested solutions are proposed in both kernel and user

spaces to have TCP modules more effectively differentiate two types of losses. We conclude in Section 6 and introduce the work in future.

2. Existing Network Simulators

In the past decades some network simulators provide much needed tools in research and are widely used. However there are some reality concerns:

NS-2 (Network Simulator 2, [3]) is a discrete event tool based on the work of REAL [4]. NS-2 and REAL employ a similar model constructing scenarios with nodes and links, while: 1) NS-2 node has no address (Classful [5] or Classless [6]), no prefix for inter-domain routing. Routes of packets are predetermined by simulator, which is not the case in real networks; 2) service time on node and statistics, such as node service rate, packet arrival and departure rates, discards on TTL (Time-To-Live) expiry, cannot be gained because the packets are enqueued on the links; 3) socket interface is missing.

OMNeT++ [7] is a free and relative new network simulator. Its scenarios are built with modules that are integrated with a FreeBSD kernel, thus avoid the re-implementation of a protocol stack; the inter-domain simulation issue was partially solved. However, 1) OMNET++ employs an "Internet-like topology" [7] because the IP network architecture is not included; 2) the tight integration with a single FreeBSD kernel prevents it from running different versions of TCP in one scenario;

OPNET [8] is a commercial one with convenient user interface allowing people easily design and run complex simulation. While the concerns might be: 1) packet delay and loss rate are artificially set, which shall be traffic-dependent; 2) IP network architecture and socket interface are missing (like previous ones).

3. Contribution of VINS

3.1. What is New in VINS?

VINS is a WIN32 application simulating network entities (applications, sockets, nodes and links) with C++ objects. It is designed to follow a number of key requirements by Internet standards and RFCs which have not been well-considered in existing simulators. By re-implementing TCP/IP stack in the user space, VINS provides extensive statistics for performance analysis of IP networks and TCP endpoints. In comparison with the existing tools mentioned above, VINS has the following new features:

1) Socket interface is devised, through which applications can access network services as it was implemented in the BSD operating system [9]. Thus VINS supports full-duplex communication and delayed acknowledgment by removing fictional TCP Source and Sink (NS-2, OMNeT++ and OPNET have no socket);

2) In VINS, overflow is reproduced on the node and packet loss is traffic-dependent (OPNET imposes a artificial loss ratio in dropping packets, and ignores the dependency between losses);

3) Support dynamic TCP payload encapsulation (NS-2, OMNeT++ and OPNET packets have fixed size to 1 MSS);

4) Node performs LPM (Longest Prefix Match) algorithm to lookup and forward packets as Classful networking, which enables building-up scenarios in large-scale and real

Internet topology (NS-2 and OMNeT++ automate the route while OPNET uses Path Editor to determine packet route, which is not realistic enough).

3.2. Key Features in TCP Implementation

Though TCP has many implementations, some key requirements by RFC 793 and 1323 are fundamental and valid today (e.g. in FreeBSD8.0). They are implemented in VINS too:

3.2.1. Initial Values

A number of control variables in TCP control block are initialized with same values, according to [9]:

ssthresh←65535 bytes (Slow-Start-Threshold)
RTO←6000 ms (Retransmit-Time-Out)
RTTVAR←3000 ms (Round-Trip-Time Variation)
SRTT←0 (Smoothed-Round-Trip-Time)
MSS←1460 bytes (Maximum Segment Size)
cwnd←1 MSS (Congestion Window)

3.2.2. RTO Calculation

RTO is renewed on each ACK received, which is the well-known Jacobson algorithm [9], which is still valid in many recently released operating systems, such as FreeBSD 8.0. VINS follows this algorithm by setting the gain applied to RTT, ***g***, to 1/8 and the gain applied to mean deviation estimator, ***h***, to 1/4. SRTT is initialized to 0, adjusted on the first ACK's arrival: SRTT ← RTT, where RTT is the measured value of the first round trip time. In BSD this value is measured during connection establishment, while in VINS it is measured on receiving the ACK of the first data PDU. Hence on each ACK's arrival, RTO will be recalculated as in the following steps:

Delta ← RTT - SRTT
SRTT ← SRTT + g*Delta
RTTVAR ← RTTVAR + h*(|Delta| - RTTVAR)
RTO ← SRTT + 4*RTTVAR

3.2.3. Loss Detection and Retransmit

TCP keeps all bytes received in order by checking the sequence number of each arrival segment. It sends duplicate ACK on receiving out-of-order sequence numbers to trigger a re-transmit on the sender. Sender re-sends the possible lost segment on RTO occurs, and shall be able to re-send expected data on duplicate ACKs are received, instead of waiting for RTO. This mechanism is known as Fast Retransmit.

3.2.4. TCP Segment Payload Size

TCP Maximum Segment Size (MSS) is limited by an estimated Maximum Transfer Unit (MTU), which is recommended to be 1500 bytes. TCP maximum data payload is up to 1460 bytes. Since hosts are generally not the bottleneck, VINS ignores flow control and focuses on congestion control. On each ACK arrival, snd_una will advance to right and congestion window will be inflated. Let new available in cwnd be n bytes and socket so_snd buffer be full of outgoing data, constraints in determining the payload size m are:

m ← min(SB_MAX-cwnd, n); // not to exceed cwnd
m ← min(m, MSS); // not to exceed MSS

SB_MAX is default to 65535 bytes as the length of so_snd buffer in bytes, limiting the maximum size of congestion window as well TCP flight size. When available data size is less than MSS/4, VINS has the sender hold for a larger m. This control is known as "small packet avoidance" [9] to utilize network service more efficiently.

3.2.5. Delayed and Piggybacking ACK

As a full-duplex protocol, TCP allows the ACK segments to be sent in piggybacking of outgoing data. VINS follows 4.4BSD: an ACK can be held up to 200 ms in hope to catch on with an outgoing segment during this period, known as Fast Timeout [9]. Currently we assume all acknowledgments are delayable.

4. Characterizing Packet Losses

In this section we investigate two types of packet losses: 1) losses due to intermediate node overflow; 2) those due to link disruption by obstacles or fading in wireless networks. A scenario is designed and shown in Figure 1, the "X-Test", consisting of a pair of connected TCP Reno peers and one-way background traffic implemented with SUPPLIER (X1) and CONSUMER (Y2) [2] nodes. Background traffic is tuned as by varying the mean departure intervals between 2 successive packets that sent by the X1. The stochastic departure interval is exponentially distributed (executable file and scripts from [10]).

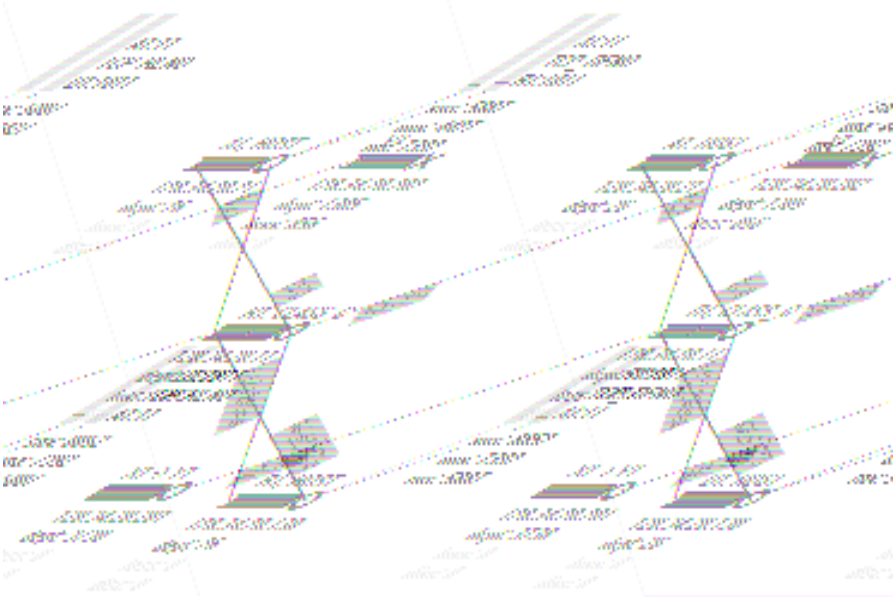

Figure 1 Screenshot of X-test

4.1. Packet Losses in Wired Networks

As the interval between each two successive packets is exponentially distributed, the background traffic results in Poisson Arrival at node R1. VINS logs the entire TCP control block, and presents the following variables in graphs: the congestion window in bytes, pending segment numbers and slow-start thresholds. Figure 2 presents samples of the evolutions when background traffic mean rate is {100, 50, 25, 12.5, 6.25, 3.125} packet/s, as intuitive insights into TCP module's behaviors.

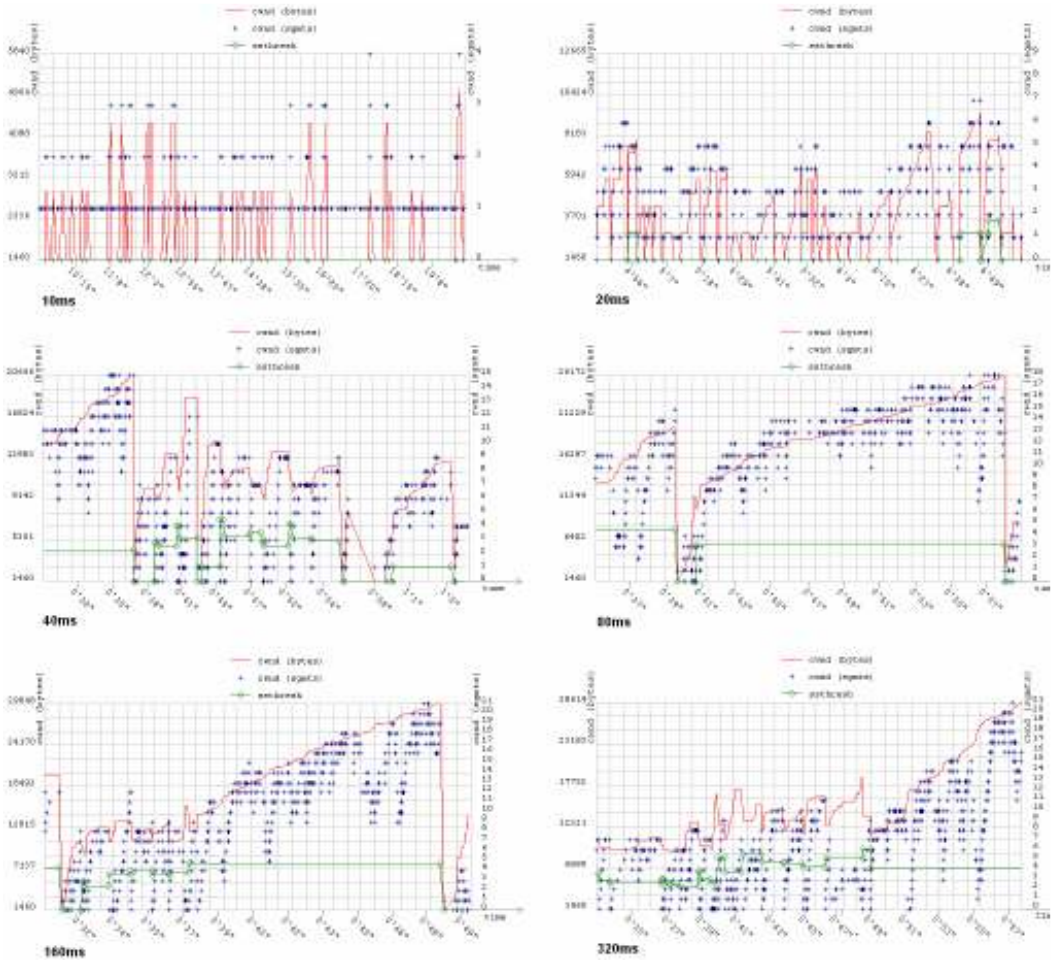

Figure 2 Congestion Window, Number of Pending Segments and Slow-Start-Threshold in X-Test

We tune the mean departure rates of background traffic departing from X1 as {100, 83.33, 71.43, 62.5. 50, 40, 31.25, 27.78, 25, 12.5, 6.25, 3.125} packet/s, and gain statistics of both discrete packets and byte-streams. The three primary counters of each node (dept, lost and disc) indicate the quantities of the departed, lost and discarded packets on the node. Details can be gained from VINS System Report at simulation run time.

Figure 3 and Figure 4 present the three traffic's mean rates of arrival (at R1) and departure (from R1):

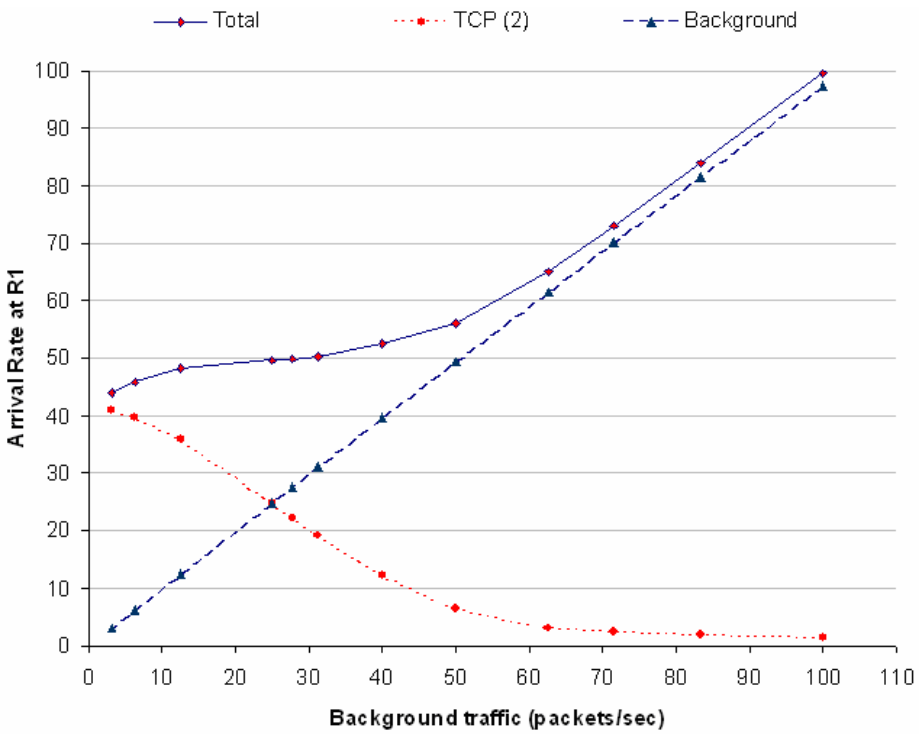

Figure 3 Packet Arrival Rates at Node R

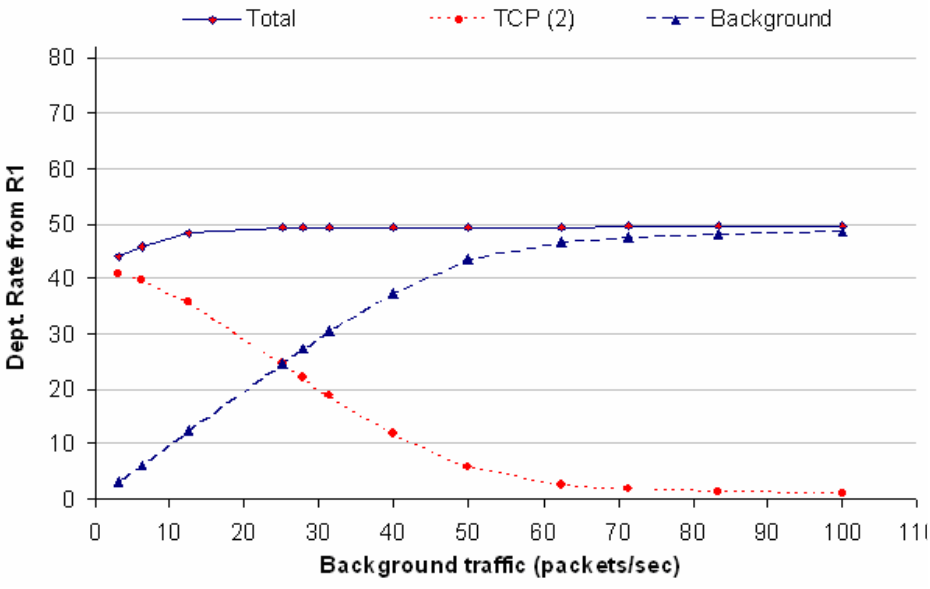

Figure 4 Packet Departure Rates from Node R

Figure 4 demonstrates that with the increment of background traffic rate, the total departure rate from R1 is approaching to the maximum output rate of R1, which is 50 packets/sec; meanwhile the departure rate of TCP shrinks for congestion avoidance.

The relation between TCP throughput and packet loss ratio are shown in Figure 5, showing that the impact of packet loss ratio on TCP is severe. We characterize the TCP control block on overflow loss in the following:

1) throughput is non-zero, since the congestive loss ratio is always under 100%
2) snd_una advances on expected ACK arrival
3) retransmit timeout and ssthresh get updated on any valid segment arrival

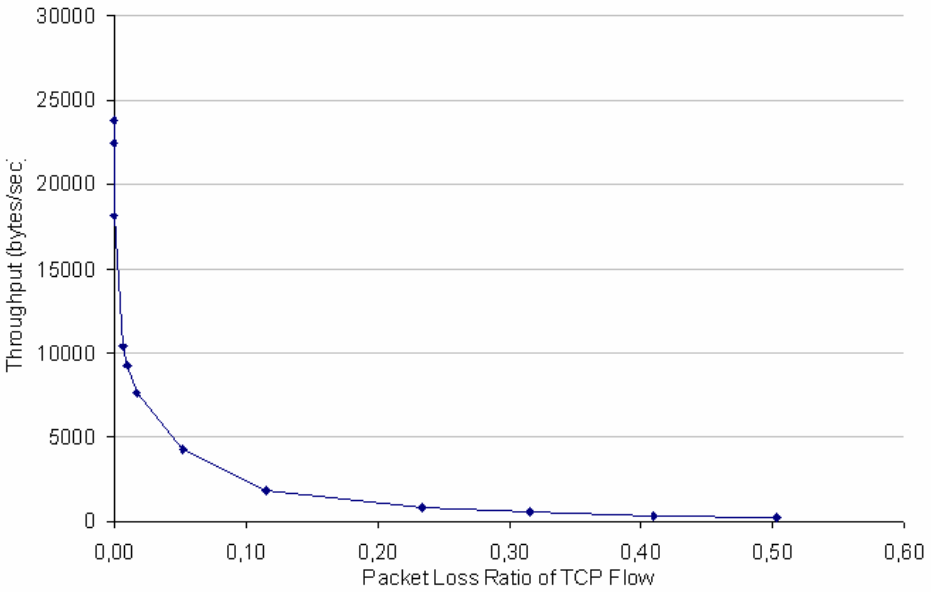

Figure 5 Relation of TCP Throughput with Packet Loss Ratio

The evaluation in Figure 5 can be compared with other simulation-based and experimental results, e.g. [11], showing the loss ratio can slow-down TCP throughput drastically.

4.2. Packet Losses on Wireless Link Disruption

In wireless networks, link disruption due to hardware failure, obstacles or fading etc., can be frequent. Differentiating the losses due to link disruption may help the TCP module in making right decisions: instead of tuning the congestion window and entering congestion avoidance states, it would be expected introducing a new strategy in handling packet loss which matters not congestion.

To simulate the packet loss on link disruption, we employ the same scenario as Figure 1 presents: suspend the router R1 and keep two TCP hosts running. Figure 6 illustrates a segment of TCP control block log, recoding event of link disruption and TCP's response: congestion window size in bytes, segments ssthresh, snd_nxt and RTO.

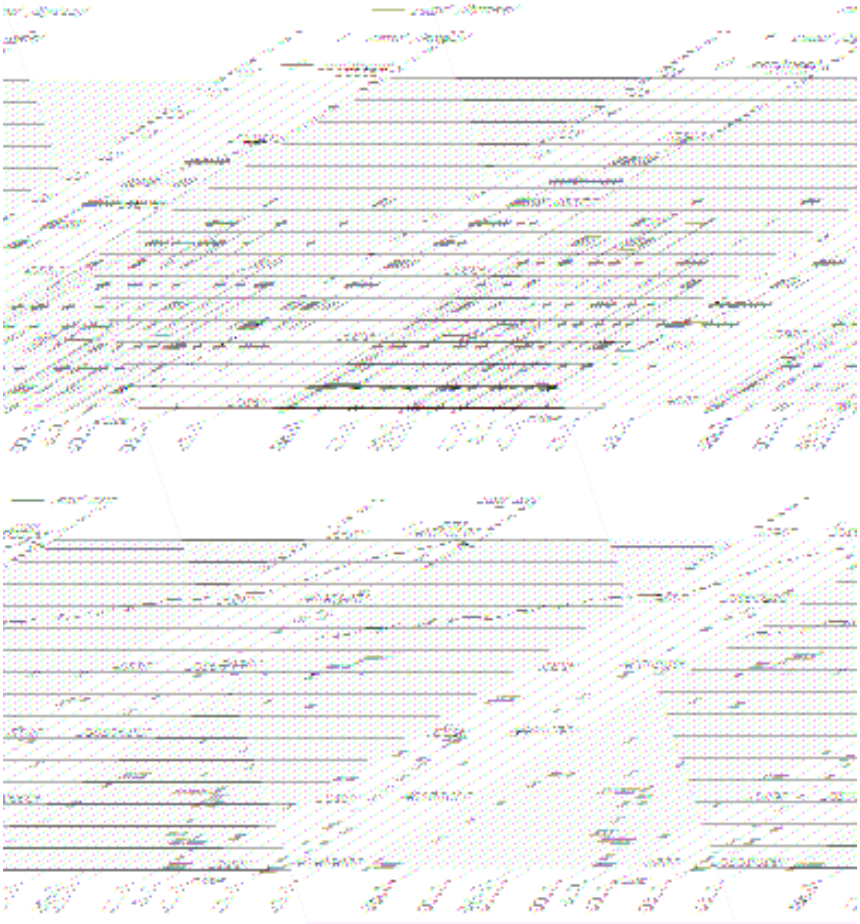

Figure 6 TCP Control Block on Link Disruption

We can characterize the TCP control blocks on link disruption in the following:

1) throughput is cut to 0
2) snd_nxt stops advancing
3) RTO gets no more updated
4) congestion window size remains as 1 MSS

5. Differentiating Two Types of Losses

Many today's TCP/IP stacks are derived from the implementation in 4.4BSD, which was developed to avoid congestive packet losses in wired networks. On RTO occurrence, traditional TCP modules adjust the congestion window size to slow-down its sending rates, and may send probing segments on Keep-alive [9] timeout. Retransmit may be performed many times before reporting a connection disruption to the user application (e.g., connection timeout). The cwnd tuning and connection-alive probing processes may last considerable time, depending on the version of TCP and contemporary RTO value. While this strategy helps nothing if a link gets disrupted.

In comparison with the case in wired networks, packet losses on wireless links are hardly to be predicted with RTT or sporadic losses, but easily to be recognized with the four characteristics we presented in the last section. Nevertheless, people may expect to find a new gateway as early as possible. Two potential improvements are proposed to have TCP more adaptive in the wireless environment:

1) In the kernel space: we suggest employing a "Gateway Timer" to enhance existing TCP modules. Like the TCP retransmit timer, Gateway Timer gets reset on any segment arrival on the current connection. Its value can be fixed or associated with RTO (e.g., 1.5 times of current RTO). On Gateway Timeout, TCP silently re-route to find other possible gateways instead of tuning cwnd or keeping-alive. If the re-routing succeeds, all existing connections can be immigrated onto the new path. These activities are transparent to the users, which may raise the least negative impact to the user's session (web-browsing, email, FTP download etc).

2) In the user space: user application may disable Keep-alive timer and set socket in non-blocking mode in sending and receiving. Collaborations are required on both client and server sides: they both need a timer to monitor the duration of 0-throughput. On client end, it may recommend user to reconnect to the server on a certain timeout, without terminating the current session; while server accepts the recovering re-connection without reset the current session. This solution does not require changes to the OS kernel.

One open issue in suggestion 1) is the optimal timer in the kernel. Too frequent changing routes may bother other connections and slow down network's performance. For example, temporary obstacles may cause only a short period of link disruption, and is tolerable to the user application. On this case, it is not necessary to re-route or re-establish the connection.

6. Conclusion

In this paper we analyze the different impacts of congestive and non-congestive packet losses with VINS, and propose two suggested improvements that may help in developing wireless-adaptive OS kernels and TCP-based applications. VINS has a set of new aspects in comparison with other simulators'. It provides socket interface to the applications, through which user can access the services provided by the networks. Nodes in VINS are modeled closely matching to what they are in the real world, with a 32-bit address, forwarding prefix, routing-table, IP interrupt queue and characterized service time distribution. Being the bottleneck in today's Internet [12], node's performance might have been expected for quantitative analysis.

As a new simulator, development of VINS is still ongoing. In a short future we will propose a variety of mathematical models in simulating sorts of link layer packet losses, support the fast-growing technologies such as IP tunneling and Ad Hoc; Human and other peripheral events are being modeled and parameterized, such as triggering frequent topological changes in mobile networks. Meantime we are also trying to establish analytical models and tools for complex and cross-layer protocol analysis, which has been presented in [13].

References

[1] S. Floyd, "A Report on Some Recent Developments in TCP Congestion Control", IEEE Communications Magazine, June 2000

[2] N. Li. "Node-Oriented Modeling and Simulation of IP Networks", Proc. of 14th Annual IEEE International Conference and Workshop on the Engineering of Computer Based Systems, ECBS 2007

[3] K. Fall and K. "Varadhan, The NS Manual", September, 2006.

[4] S. Keshav, "Real: A Network Simulator", Technical Report TR-88/472, University of California, Berkeley, December 1988

[5] J. Postel, "Internet Protocol", RFC 791

[6] V. Fuller, "Classless Inter-Domain Routing, CIDR: an Address Assignment and Aggregation Strategy", RFC 1519

[7] F. Baumgartner, M. Scheidegger and T. Braun, "Enhancing Discrete Event Network Simulators with Analytical Network Cloud Models", International Workshop on Inter-domain Performance and Simulation (IPS), 2003

[8] OPNET Technologies, Inc. http://opnet.com/

[9] G. W. Wright and W. R. Stevens, "TCP/IP Illustrated Volume II - The Implementation", Addison Wesley, 1994

[10] VINS download package: https://www-fgks.hpi.uni-potsdam.de/fileadmin/user_upload/VINS/VINS.zip

[11] S. Floyd and Kevin Fall, "Promoting the Use of End-to-End Congestion Control in the Internet", IEEE/ACM Transactions on Networking, Vol. 7, No. 4, August 1999

[12] P. Molinero-Fernández, "Circuit Switching in the Internet", Ph.D. Thesis, Stanford University, 2003

[13] W. Zorn, "FMC-QE: A New Approach in Quantitative Modeling", Proc. of 2007 World Congress in Computer Science, Computer Engineering, & Applied Computing, 2007

Authors

Li, Nanjun (1972) got his B.S. in 1995 and M.E. in 2001, both from Zhejiang University, China. He worked in Destiny Electronics Co. (1995 – 1998) and later in General Electric Co. (2001 – 2002). Since 2002 he has been working as a research assistant and toward the Ph.D. degree in the Communication System Research Group, led by Werner Zorn in Hasso Plattner Institute, the University of Potsdam.

Zorn, Werner (1942) studied electrical engineering (Dipl.-Ing.) at Karlsruhe University (TH) 1962-67, gaining his doctorate in 1971 (Dr.-Ing.). He became the director of the Informatics Computing Center (IRA) in 1972, the professor for Computer Science both at Karlsruhe University in 1979. Since 06/2001 he has been the head of Communication Systems Research Group at the Hasso-Plattner-Institute, the University of Potsdam.