International Research Journal of Engineering and Technology (IRJET) e-ISSN: 2395-0056
Volume: 05 Issue: 04 | Apr-2018 www.irjet.net p-ISSN: 2395-0072

Modeling a New Startup Algorithm for TCP New Reno

Ahmed Yusuf Tambuwal[1], Rafaello Secchi[2]

[1]*Computer Science Unit, Usmanu Danfodiyo University, Sokoto (UDUS)*
[2]*Electronics Research Group, University of Aberdeen, UK*

Abstract - *Standard TCP (New Reno) is vulnerable to startup effects that cause loss of connection setup packets or result in long round trip time (RTT) greater than 1-second. When either of these events occurs, TCP New Reno resets its congestion state by reducing initial congestion window (IW) and slow-start threshold (ssthresh) values to 1 and 2 maximum segment size (MSS) respectively. In this condition, TCP requires multiple round trips to complete delay-sensitive transactions, thus resulting in poor user-experience. This paper presents a new congestion control algorithm that makes TCP more responsive by increasing its robustness against startup losses. Our main contribution in this paper is derivation of a stochastic model that yields a simple expression for computing the latency of a short-lived transaction as a function of IW, ssthresh and bandwidth-delay product (BDP) of an uncongested link.*

Key Words: Transmission Control Protocol, Congestion Control, Startup, Responsiveness, Interactive Applications

1. Introduction

The transmission control protocol (TCP) 1 is the main protocol used on the Internet for reliable delivery of data packets between communicating hosts. A TCP client initiates a connection to a remote Internet server using a three-way handshake (3WHS) procedure. In general, the client then submits a data request, which is processed by the server resulting in a data response. Once data transmission starts, TCP attempts to maximize throughput without causing congestion on the network. Several works have focused on designing new TCP algorithms with better throughput performance such as in 2345.

Conversely, throughput performance is not the main requirement of short-lived interactive applications (e.g. web browsing and E-commerce), which account for a majority of TCP flows 67. Quite different from bulk transfers, interactive applications demand speedy delivery of few data chunks across the Internet within short delay bounds. Despite many algorithms proposed to solve this important problem 89101112, it still remains an open challenge for TCP.

This paper proposes a new algorithm that aims to make standard TCP (New Reno) more responsive by increasing its robustness against startup losses. TCP New Reno interprets the loss of connection setup packets (i.e. SYN or SYN-ACK) as a signal for serious network congestion, prompting a sender to reduce its initial congestion window (IW) to 1 maximum segment size (MSS) and its slow-start threshold (ssthresh) to 2 MSS 13. This response increases latency of short-lived interactive applications by several round trips, thus significantly reducing end-user Internet experience.

While ignoring the SYN congestion signal and starting with very large IW and ssthresh values negates TCP conservative principles, there are strong motives to use a less conservative approach. Firstly, random packet loss is quite a common occurrence when data traverses wireless/mobile network links e.g. due to high contention between multiple users sharing the radio channel, poor weather conditions, or when a mobile host is obstructed and suffers temporary link outages 1415. Also, network middle boxes such as firewalls, proxies, and network address translators, can erroneously drop SYN packets due to suspicion of unwanted or malicious traffic 1617. In a more general context, TCP inherently causes loss of packets (including the SYN and SYN-ACK) when probing for available capacity and trying to maximize throughput 181920.

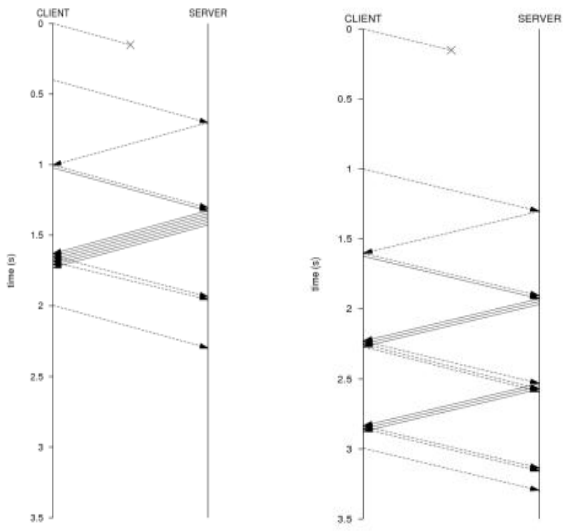
We therefore propose a new algorithm called 'TCP SYN Loss (TSL) Startup Algorithm' that uses a *halving* congestion response function during startup, which is less conservative than the current standard. After connection setup is completed, standard TCP congestion control is applied for the data transfer phase.

The rest of this paper is organized as follows: Section 2 presents the TSL startup algorithm that we propose for use when congestion is signalled during the connection setup phase of a new connection. Section 3 then develops a stochastic model of the proposed algorithm with a discussion of its impact on web transactions in section 4. Finally, section 5 concludes the paper and presents future work.

2. TSL Congestion Control Algorithm

Deviating from current standard, the proposed TSL startup algorithm reduces the default startup rate of a new TCP flow by a maximum factor of 2 when the SYN or SYN-ACK packet is dropped during connection establishment.

If a further loss occurs before the connection is established, then the algorithm reverts to the standard TCP behaviour by resetting the Initial congestion window to 1 MSS and Slow-start threshold to 2 MSS.



(a) (b)

Fig. 1 Illustration of a typical TCP transaction (a) without SYN loss and (b) with SYN loss

Our proposed algorithm uses the following set of instructions.

$$\textbf{\textit{if}}\ (\textbf{\textit{SYN Packet is Dropped during 3WHS}})\ \textbf{\textit{Then}}\ \{$$

$$\textbf{\textit{Set IW}} = \textbf{\textit{max}}\ (IW_{min}, \frac{IW}{2})$$

$$\textbf{\textit{Set ssthresh}} = \textbf{\textit{max}}\left(ssthresh_{min}, \frac{ssthresh}{2}\right)$$

}

Since the aim of the TSL algorithm is to enable a new flow to complete short data transfers quickly using slow-start, it attempts to keep startup variables as high as possible even when congestion is signalled. Hence in addition to rate halving, lower bounds (i.e. IW_{min} and $ssthresh_{min}$) are defined in case the default IW and ssthresh values are initially low e.g. when implementing standard RFC 5681, which limits IW to 3 MSS.

It can be argued that if the SYN congestion signal was really due to high network utilisation, then the TSL flow remains aggressive for only 1 round trip time (RTT) before receiving additional feedback and falling back to standard TCP congestion behaviour. Conversely, if the signal is only due to transient congestion or transmission errors, slow-start quickly grows the congestion window and completes a short data transfer quickly, to the benefit of the end-user.

Figure 2 illustrates how TSL startup enables a short flow to complete faster than a parallel TCP New Reno flow. Both flows drop the SYN-ACK packet at time t_1. However when data transmission eventually begins at t_2, the TSL flow completes faster because it sets IW to 3 MSS and ssthresh to 40 MSS (compared to 1 MSS and 2 MSS respectively for the TCP New Reno flow).

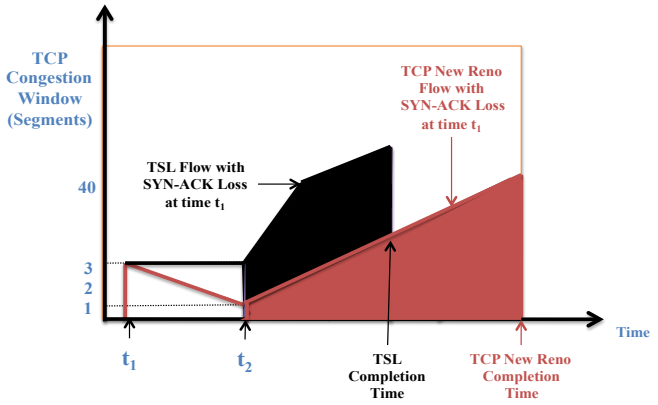


Fig. 2 Illustration of a TSL flow starting with high IW and ssthresh values after recovering from SYN-ACK loss, and completing faster than a parallel TCP New Reno flow.

3. TSL Model

This section develops a stochastic model of TSL startup algorithm that yields an analytic expression for the latency of a short-lived and uncongested TCP flow as a function of IW, ssthresh and link bandwidth-delay product (BDP). The aim is to gauge impact on response time of various TSL startup algorithms that may be used when congestion is wrongly inferred due to a SYN-ACK packet loss. A few assumptions are made:

1. The slow-start (SS) and congestion avoidance (CA) algorithms are based on TCP Reno.
2. During SS, the cwnd is increased by 1 for each ACK received i.e. no use of delayed ACK.
3. During CA, the cwnd is increased by 1 per RTT.
4. There is no data packet loss after the 3WHS, and the receiver advertises an infinite window size.
5. The bottleneck link is asymmetric.

Hence, for a given flow size L (in Bytes), the sender transfers maximum-sized segments (MSS) as fast as its congestion window (cwnd) and slow-start threshold (ssthresh) allow. But the sender rate may also be limited by the bottleneck link, which is characterized with a capacity C (in bps) and propagation delay d (in seconds).

If the TCP header size is denoted as HDR (in Bytes), then the bandwidth-delay product in packets (BDPp) is expressed as:

$$BDP_p = \frac{C.d}{8\ (MSS + HDR)} + 1 \qquad (1)$$

The flow size in packets (Lp) is expressed as:

$$L_p = \left\lceil \frac{L}{MSS} \right\rceil \quad (2)$$

The time to transmit the header is expressed as:

$$\tau_H = \frac{8.HDR}{C} \quad (3)$$

The time to transmit the payload is expressed as:

$$\tau_p = \frac{8.MSS}{C} \quad (4)$$

The time to transmit the request is:

$$\tau_R = \frac{8.RQ}{C} \quad (5)$$

And the maximum round-trip delay is given as:

$$\tau = 2\tau_H + \tau_P + d \quad (6)$$

The model considers four possible TCP phases as illustrated in Figure 3.

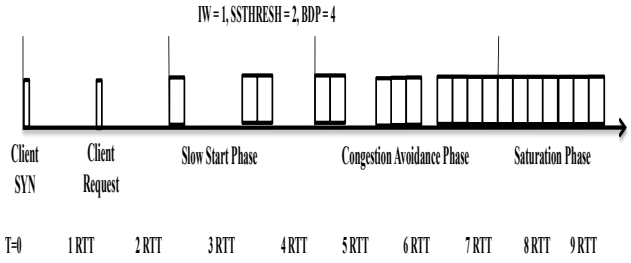


Fig. 3. Illustration of data transfer with IW=1 and ssthresh=2 and buffer capacity=4 segments

3.1 Connection Establishment Phase

Before data transfer begins, a connection must be established between the two end-hosts that wish to communicate. The connection establishment process is usually called three-way handshake (3WHS). Most TCP implementations also send the data request packet together with the last ACK of the 3WHS.

The duration of the 3WHS including sending of the data request is derived as:

$$\tau_{ex} = 2.(2.\tau_H + d) + \tau_R \quad (7)$$

3.2 Slow Start Phase

The data flow begins in slow-start using selected values of IW and ssthresh. During slow-start, the cwnd is increased exponentially until ssthresh is exceeded, the buffer becomes saturated, or when the flow is completed. For a given number of slow-start rounds (Nss), the attained congestion window size (CWNDss) and the total number of sent segments (Tss) are expressed as:

$$CWND_{SS} = 2^{N_{SS}}.IW \quad (8)$$

$$T_{SS} = \sum_{k=0}^{N_{SS}} 2^k . IW \quad (9)$$

On simplification (4) becomes,

$$T_{SS} = (2^{N_{SS}+1}).IW \quad (10)$$

a. Condition 1

If slow-start is terminated by ssthresh, the following equality holds true:

$$2^{N_{SS}}.IW \geq SS \quad (11)$$

Equation (11) simplifies to:

$$N_{SS} \geq \log_2\left(\frac{SS}{IW}\right) \quad (12)$$

b. Condition 2

Similar to above, slow-start may be terminated due to buffer saturation.

$$2^{N_{SS}}.IW \geq BDP_P \quad (13)$$

Equation (8) simplifies to:

$$N_{SS} \geq \log_2\left(\frac{BDP_P}{IW}\right) \quad (14)$$

c. Condition 3

Finally, the flow completion may end slow-start

$$2^{N_{SS}+1}.IW \geq L_p \quad (15)$$

Equation (15) simplifies to:

$$N_{SS} \geq \log_2\left(\frac{L_P + IW}{2.IW}\right) \quad (16)$$

Since slow-start must be terminated when at least one of the three conditions is satisfied, combining equations (12), (14), and (16) yields,

$$N_{SS} = \left\lceil \min\left(\log_2\frac{BDP_P}{IW}, \log_2\frac{SS}{IW}, \log_2\frac{L_P+IW}{2.IW}\right)\right\rceil \quad (17)$$

Given N_{SS} is known, the number of packets sent in slow-start (S_{SS}) and the slow-start duration (T_{SS}) can be calculated using the following equations.

$$S_{SS} = IW.(2^{N_{SS}} - 1) \quad (18)$$

$$T_{SS} = N_{SS}.\tau \quad (19)$$

3.3 Congestion Avoidance Phase

Congestion avoidance starts immediately after SS. The cwnd size in the first round of congestion avoidance is equal to ssthresh, and it is incremented by 1 MSS per RTT. Hence an expression can be derived for total number of segments sent (S_{CA}) for a given number of round trips (N_{CA}) spent in congestion avoidance.

$$S_{CA} = ssthresh + (ssthresh + 1) + \ldots (ssthresh + N_{CA} - 1) \quad (20)$$

The above expression can be written in the concise form below.

$$S_{CA} = \sum_{k=1}^{N_{CA}} (ssthresh + k - 1) \quad (21)$$

Equation (21) when expanded becomes:

$$S_{CA} = N_{CA} \,.\, ssthresh + \frac{N_{CA} \,.\, (N_{CA} - 1)}{2} \quad (22)$$

To solve for N_{CA}, two conditions are considered that terminate CA.

a. Condition 1

The number of remaining data segments (R) to be sent after slow-start is known to be,

$$R = L_p - S_{SS} \quad (23)$$

Since congestion avoidance must terminate when these remaining flow segments are transmitted, then

$$S_{CA} \geq R \quad (24)$$

Substituting for S_{CA} and R in (24), and solving the quadratic equation gives:

$$N_{CA} \geq \sqrt{\left[\left(ssthresh - \tfrac{1}{2}\right)^2 + 2.\left(L_p + S_{SS}\right)\right]} - \left(ssthresh - \tfrac{1}{2}\right) - 1 \quad (25)$$

b. Condition 2

Congestion avoidance may be terminated before the flow ends if the buffer becomes saturated. This gives a second equation for N_{CA}.

$$N_{CA} \geq \lceil BDP_P \rceil - ssthresh \quad (26)$$

Therefore the final expression for N_{CA} is stated below as:

$$N_{CA} = \min \left[\sqrt{\left(ssthresh - \tfrac{1}{2}\right)^2 + 2.(L_P + S_{ss})} - \left(ssthresh - \tfrac{1}{2}\right) - 1,\ \lceil BDP_P \rceil - ssthresh \right] \quad (27)$$

Having derived N_{CA}, the number of packets sent in congestion avoidance (S_{CA}) and the congestion avoidance duration (T_{CA}) can be calculated using the following equations:

$$S_{CA} = \frac{1}{2} \,.\, N_{CA} \,.\, (N_{CA} - 1) + N_{CA}.ssthresh \quad (28)$$

$$T_{CA} = N_{CA} \,.\, \tau \quad (29)$$

3.4 Saturation Phase

If congestion avoidance is terminated because the buffer becomes saturated, the remaining data packets are sent back to back at the link capacity, akin to constant bit rate (CBR) traffic. The total number of bytes sent in saturation (S_{SAT}) and the time spent in saturation (T_{SAT}) is derived as:

$$S_{SAT} = (L_P - S_{SS} - S_{CA}).HDR + (L - (S_{SS} + S_{CA}) \,.\, MSS) \quad (30)$$

$$T_{SAT} = \frac{8 \,.\, S_{SAT}}{C} \quad (31)$$

Combining the derived expressions in (7), (19), (29) and (31) gives the final analytical expression for startup latency (T) of short-lived TCP flows.

$$T = T_{ex} + T_{SS} + T_{CA} + T_{SAT} - \tau_H \quad (32)$$

4. Analysis and Discussion

The web is essentially a large network of interlinked documents, photos, videos, and applications that are individually referred to as web objects. Web transactions are request-response applications, which mostly use TCP for reliable delivery. However, some important applications such as Domain Name Server (DNS) queries primarily use the User Datagram Protocol (UDP). The performance impairment due to slow startup and high latency depends on the types of applications using the network. For this research response time for web object retrieval is considered are defined.

Several variants of TSL startup are possible. These can be divided into two classes based on minimum value of IW after SYN loss.

4.1 Gentle-Start TSL Variants

The gentle-start TSL variants react conservatively when loss of the SYN-ACK packet is detected. The

minimum value of IW is set to 1 segment after the 3WHS, with the standard TCP algorithm falling into this category. Standard TCP additionally sets the minimum ssthresh value to 2 segments causing a slow starting rate of 1 segment that only increases linearly per RTT. Other gentle-start TSL variants are designed so that the sending rate is initially low but is exponentially increased by setting minimum ssthresh value at an intermediate or high level. The gentle-start variants evaluated include:

- TSL 1, 2 (Standard TCP)
- TSL 1, 16
- TSL 1, 1000

4.2 Brute-Start TSL Variants

The brute-start TSL variants react aggressively when loss of the SYN-ACK packet is detected. The minimum value of IW is set to 3 segments after the 3WHS. Hence if the connection response size is less than 4KB; it is possibly completed in 1 RTT after the 3WHS. For larger connection sizes, the response time depends on the minimum ssthresh value, which may be set to 2 segments for a linear starting rate, or set arbitrarily high for an exponential starting rate. Three brute-start variants are studied:

- TSL 3, 2
- TSL 3, 16
- TSL 3, 1000

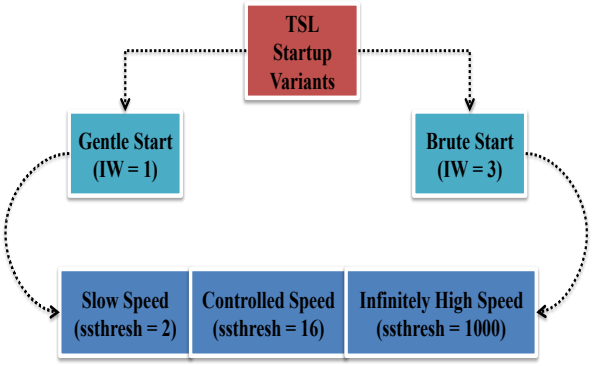


Fig. 4 TSL startup variants

TABLE-1: Impact of TSL Startup on Web Latency

TSL Startup Algorithm	Transfer Size		
	8KB	50KB	180KB
	Minimum Response Time (in RTTs)		
TCP New Reno	5	10	18
TSL 1, 16	4	7	12
TSL 1, 1000	4	7	8
TSL 3, 16	3	5	10
TSL 3, 1000	3	5	7

Table 1 shows web object and web page response times (measured in units of round trips) of different TSL configurations in an uncongested scenario. The proposed TSL startup algorithm was found capable of subtracting 2-5 round-trips from duration of short web flows in an uncongested scenario. and up to 11 round-trips for moderate flows up to 180KB. This is an important outcome because quickly completing connection setup and the initial data transfer of a few KB is critical for many current Internet applications. Web-based applications, for example, need to present the initial contents to the user in a handful of seconds to achieve acceptable performance.

5. Conclusion

Short TCP flows may suffer significant response time performance degradations when a small SYN control packet is lost or during a transient congestion period. Unfortunately, this creates an incentive for misbehavior by clients of interactive applications (e.g., gaming, telnet, web) to send "dummy" packets into the network at a TCP-fair rate even when they have no data to send, thus improving their performance in moments when they do have data to send. Even though no "law" is violated in this way, a large-scale deployment of such an approach has the potential to seriously jeopardize one of the core Internet's principles — statistical multiplexing. In this way, they become capable of developing larger congestion windows and improve their performance by avoiding long retransmission timeouts. While several solutions have been proposed to efficiently combat the problem, none has been deployed in the Internet, probably because they require non-negligible architectural changes.

This work has presented an easy to implement algorithm that will reduce startup latency of standard TCP for the benefit of both Internet users and service providers. For example, one of the top online-based commercial stores (Amazon.com) recently estimated that every 100ms of additional latency reduces profit by 1% 19. Google also reported that increasing the retrieval time of its search page from 400ms (old page with 10 results) to 900ms (new page with 30 results), decreased traffic and ad revenues by 20% 20. A different set of results shows that end-users are willing to wait no longer than 4 seconds for a web page to be displayed 2122, or 10 seconds for a video streaming session to startup A, before quitting or restarting the connection.

Our future work will perform simulation studies to measure the impact of our proposed algorithm on a congested Internet. The performance figures derived from the model in this paper account only for uncongested network scenarios, which is the normal condition of the Internet.

References

1. W. R. Stevens, TCP/IP Illustrated Volume I: The Protocols, New York:Addison-Wesley, 1994.
2. A. Baiocchi, A. Castellani, and F. Vacirca, "YeAH-TCP: Yet another highspeed TCP," in Proceedings of PFLDNET, Los Angeles, CA, February 2007.
3. S. Ha, I. Rhee, and L. Xu, "CUBIC: a new TCP-friendly high-speed TCP variant," SIGOPS Oper. Syst. Rev., 42(5), 2008.
4. R. Mittal, V. T. Lam, N. Dukkipati, E. Blem, H. Wassel, M. Ghobadi, A. Vahdat, Y. Wang, D. Wetherall, and D. Zats, "TIMELY: RTT-based Congestion Control for the Datacenter," ACM SIGCOMM Comp Commun. Review, vol. 45, no. 4, pp. 537–550, 2015.
5. N. Cardwell, Y. Cheng, C. S. Gunn, S. H. Yeganeh et al., "BBR: congestion-based congestion control," Commun. ACM, vol. 60, no. 2, pp. 58–66, 2017.
6. P. Borgnat, G. Dewaele, K. Fukuda, P. Abry, and K. Cho, "Seven Years and One Day: Sketching the Evolution of Internet Traffic," INFOCOM 2009, IEEE, Rio de Janeiro, 2009, pp. 711-719.
7. "Scaling in Internet Traffic: A 14 Year and 3 Day Longitudinal Study with Multiscale Analyses and Random Projections," IEEE/ACM Trans. on Networking, vol. 24 (4), pp. 2152-2165, 2017.
8. N. Dukkipati, T. Refice, Y. Cheng, J. Chu, T. Herbert, A. Agarwal, A. Jain, and N. Sutin, "An Argument for Increasing TCP's Initial Congestion Window," ACM Comput. Commun. Rev., 40, 2010.
9. S. Radhakrishnan, Y. Cheng, J. Chu, A. Jain, and B. Raghavan. "TCP Fast Open," In Proc. of the ACM Int. Conf. on Emerging Networking Experiments and Technologies (CoNEXT), Dec. 2011.
10. H. Sangtae, R. Injong, "Taming the elephants: New TCP slow start," Elsevier Computer Networks, vol. 55 (9), pp. 2092-2110, Jun, 2011.
11. T. Flach et.al, "Reducing web latency: the Virtue of Gentle Aggression", in Proc. of ACM SIGCOMM, (Hong Kong, China), pp. 159-170, 2013.
12. B. Briscoe et.al, "Reducing Internet Latency: A Survey of Techniques and their merits," in IEEE Commun. Surveys & Tutorials, vol. 18 (2), pp. 2149-2196, 2016.
13. M. Allman, V. Paxsons, "TCP Congestion Control," IETF RFC 5681, Sep. 2009.
14. L. Angrisani, and M. Vadursi, "Cross-Layer Measurements for a Comprehensive Characterization of Wireless Networks in the Presence of Interference," IEEE Trans. on Instrumentation and Measurement, vol. 56, (4), pp.1148-1156, 2007.
15. A. Sheth, S. Nedevschi, R. Patra, S. Surana, L. Subramanian, and E. Brewer, "Packet Loss Characterization in WiFi-based Long Distance Networks," IEEE INFOCOM, Alaska, USA, 2007.
16. A. Medina, M. Allman, and S. Floyd, "Measuring Interactions Between Transport Protocols and Middleboxes," in Proceedings of the 4th ACM SIGCOMM conf. on Internet measurement, Taormina, Sicily, 2004, pp. 336-341.
17. W. Zhaoguang, Z. Qian, Q. Xu, Z. Mao, and M. Zhang, "An untold story of middleboxes in cellular networks," ACM SIGCOMM Comp. Commun. Review, vol. 41, (4), pp. 374-385, 2011.
18. Akamai, "State of the Internet," [online]. Available: http://www.akamai.com/html/about/press/releases/2013/press_101613.html (Accessed: 14 October 2013).
19. G. Linden, "Make Data Useful" [online]. Available: http://sites.google.com/site/glinden/Home/StanfordDataMining.2006-11-28.ppt . Retrieved 20 Mar. 2014.
20. Nokia Siemens Network, http://br.nsn.com/file/2103/latencywhitepaperprintversion White paper. Retreieved 20 Mar. 2014.
21. Estimating end-to-end performance in IP networks for data applications, ITU-T Rec. G.1030, 2005.
22. G. Giambene, "QOS requirements for multimedia services," in Resource Management in Satellite Networks Optimization and Cross-Layer Design, New York: Springer, 2007, ch. 3, pp. 67-92.

A. Wang, J. Kurose, P. Shenoy, and D. Towsley, "Multimedia streaming via tcp: an analytic performance study," in MULTIMEDIA '04: Proc. of the 12th annual ACM international conference on Multimedia, New York, 2004, pp. 908-915.