CONTROL SYSTEM ANALYSIS OF A TELEMETRY NETWORK SYSTEM (TMNS)

Maria S. Araujo[1], Myron L. Moodie[1], Ben A. Abbott[1], Thomas B. Grace[2]

[1]Southwest Research Institute®
San Antonio, Texas

[2]Naval Air Systems Command (NAVAIR)
Patuxent River, Maryland

maria.araujo@swri.org, myron.moodie@swri.org, ben.abbott@swri.org, thomas.grace@navy.mil

ABSTRACT

On the surface, network-based telemetry systems would appear to be simple, stateless, information collecting entities. Unfortunately, the reality of networking technologies brings a hierarchy of control loops into the system setup. At the top level, the command and status collection data loop that users manipulate the system with is a feedback loop. The commands themselves are transmitted across the network through competing streams of data, which are guided and controlled by Transmission Control Protocol (TCP) mechanisms. TCP mechanisms themselves have control loops in order to avoid congestion, provide reliability, and generally optimize flow. These TCP streams flowing across a network fabric compete at choke points, such as network switches, routers, and wireless telemetry links – all of which are also guided by control loops. This paper discusses the hierarchy of control loops present in a TmNS, provides an analysis of how these loops interact, and describes key points to be considered for telemetry systems.

KEYWORDS

iNET, Control Systems, TCP, Throughput, Latency

INTRODUCTION

Network-based telemetry systems are not simple, stateless, information collecting entities. The reality of networking technologies brings a hierarchy of control loops into the system setup. At the top level, the command and status collection data loop that users manipulate the system with is a feedback loop. The commands themselves are transmitted across the network through competing streams of data, which are guided and controlled by TCP mechanisms. TCP mechanisms themselves have control loops in order to avoid congestion, provide reliability, and, from a fairness perspective, optimize flow. These TCP streams flowing across a network fabric compete at choke points, such as network switches, routers, and wireless telemetry links – all of which are also guided by control loops.

With the recent completion of the integrated Network Enhanced Telemetry (iNET) standards, it is important to conduct a control system analysis of the TmNS to determine whether areas of counteracting systems exist, which could degrade the end-to-end throughput and latency of the TmNS, and whether the TmNS can be optimized in ways that are simple, yet not considered. Before exploring the iNET details, it is important to review control theory concepts.

CONTROL SYSTEMS BACKGROUND

Control theory is concerned with the behavior of dynamic systems and how to control the inputs to those systems to achieve the desired outputs. Examples of control systems include thermostats, cruise control systems, electric motor controllers, flight and propulsion systems in aircrafts, among others. In fact, control systems are part of life itself. For instance, maintaining appropriate blood pressure, body temperature, and metabolism require control systems. Given their wide range of applications, it is no surprise that control systems are also present in network-based systems. Not only are control systems present in communication networks, they are at their core.

A clear example of control systems applied to communication networks is the Internet. One of the most common protocols used for the Internet is known as TCP/IP, which is based on two of the most important protocols in it: the Transmission Control Protocol (TCP) and the Internet Protocol (IP). TCP is a classical example of a control system. To motivate this discussion, Figure 1 shows the throughput of a TCP/IP network versus time. This graph was obtained using Wireshark during an iNET Reliability Critical (RC) data transfer session between two end nodes.

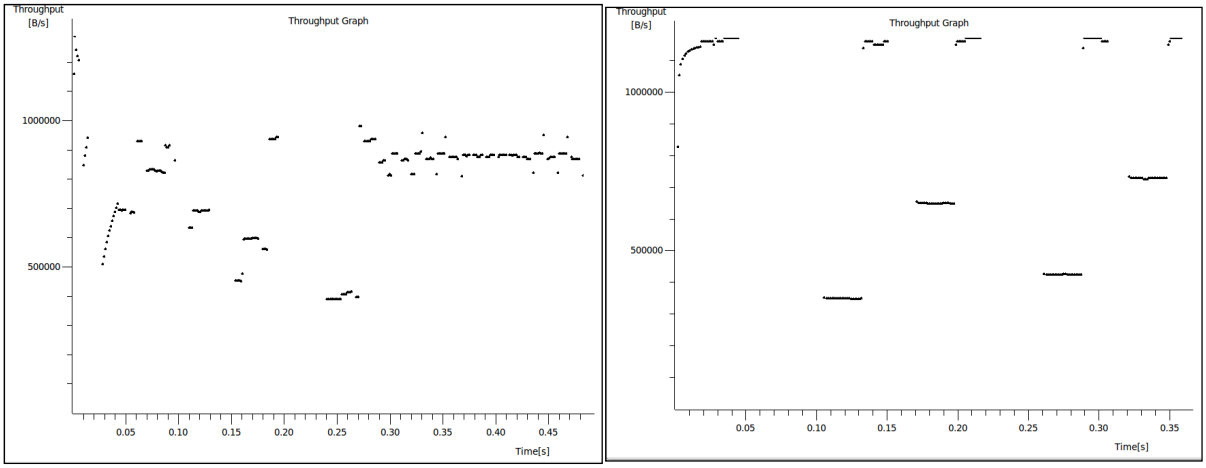

Figure 1. TCP Instantaneous Throughput Graphs over a 500-millisecond and 350-millisecond periods

It is clear from Figure 1 that there are complex dynamics involved in IP-based networks. Because the underlying mechanisms supporting IP-based networks are control systems, it is critical that those control mechanisms and their impact on overall latency and throughput be analyzed as a network-based system is designed. Failure to look at those mechanisms closely may result in an inefficient, poorly planned system. For instance, if a certain type of data has a latency requirement of 100ms and it will be transferred with protocols having the throughput

characteristics shown in Figure 1, it might be a good idea to understand the causes for the many areas of low link utilization and how to eliminate them.

CONTROL THEORY BASICS

Before delving into the details of the control loops present in the TmNS, it is important to establish a basic foundation in control theory. Over the past centuries, many control techniques have been developed, including classical feedback control [e.g. Proportional, Integral, Derivative (PID) Controller; optimal control, where a cost functional is defined and the controller tries to either maximize or minimize that cost; adaptive control, in which the controller must "adapt" to a controlled system with varying parameters; fuzzy control, where the controller is based on fuzzy logic; among others]. Controllers can be open-loop, feedback, or feed-forward. The control loops described in this paper will be primarily feedback control loops, where the measured output of the system is used in the computation of the control actions, in addition to other parameters. A basic feedback control system is depicted in Figure 2. The reference signal is the desired set point for the system. The difference between the reference signal and the measured output (i.e. error) is computed and fed into the controller. The control law in the controller then governs the system to achieve the desired system output. Control theory is concerned with system modeling, control law specification, system stability, controllability, and observability.

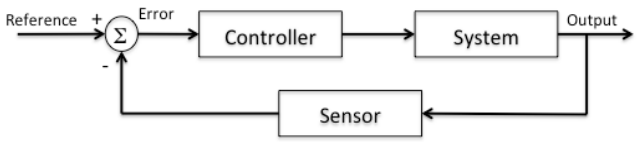

Figure 2. Basic Control System Diagram

CONTROL SYSTEMS IN THE TMNS

The TmNS is an IP-based telemetry network that uses the same building block concepts as typical enterprise networks and the Internet. The TmNS, however, has stringent performance requirements that are not otherwise present in typical networks, such as low latency constraints and high levels of available bandwidth utilization. Furthermore, the available bandwidth in the TmNS is significantly constrained (around 8Mbps), when compared to typical networks, due to the radio frequency (RF) link. The TmNS can be viewed as a big control system. Referring back to Figure 2, possible outputs of interest could be latency and throughput, with the reference points being the desired latency and throughput, for example.

The TmNS consists of multiple control loops operating autonomously, possibly in competition with each other. Data transfer in the TmNS is accomplished using two transport protocols: 1) the Latency/Throughput Critical (LTC) protocol, which transfers data using the User Datagram Protocol (UDP), and 2) the Reliability Critical (RC) Protocol, which uses the TCP protocol. While it is envisioned that LTC will be largely used for transporting data within the Test Article (TA), most data transfer occurring over the bi-directional RF link will use the RC protocol.

One of the key concepts of iNET is correcting pulse-code modulation (PCM) dropouts using the RF network. To achieve this, telemetry processors on the ground receiving PCM data will identify dropouts and initiate RC sessions with the recorder on the TA in order to request the missing data. The data will then be transferred to the ground (telemetry processors) using the RC protocol, which is TCP-based. Because TCP transfers are at the core of the TmNS, it is important to analyze them in more detail. There are several variants of TCP implementations (Tahoe, Reno, Vegas, NewReno, etc). The concepts presented in this paper will focus on the Tahoe and Reno versions, which are the most common TCP implementations, unless otherwise specified.

TCP FLOW CONTROL

TCP is composed of two main control systems: flow control and congestion control. Flow control prevents the sender from sending amounts of data that are too large for a given receiver to receive and process. An example of a situation where flow control is needed is when machines of different network speeds are involved in the data transfer. The data flow is then "controlled" in order to prevent the receiver from getting overwhelmed. In summary, the sender keeps transmitting TCP segments to a receiver as long as the receiver has enough capacity in its buffer to receive the segment (which is conveyed in the rwnd field in the TCP ACK). If the receiver's buffer is full (which is conveyed through zero window advertisements), the sender pauses transmission until the receiver's buffer can receive more data. Clearly, TCP's flow control mechanism is a feedback control system. The sender sends data to a receiver and waits for feedback from the receiver to determine how much data to send next [1].

It has been shown that the connection rate or throughput of TCP based on flow control alone is [1]

$$TCP\ Rate = {}^{1}/_{RTT} * \min\left(Bo, \int_t (PR - TCP\ Rate)\ dt\right) \quad (1)$$

where RTT is the round trip time of the connection, Bo is the receiver's buffer capacity, and PR is the processing rate of the receiver (i.e. the rate at which it can empty its buffer). Equation (1) implies that the flow control mechanism in TCP consists of an integral controller (from PID control theory), in which the TCP throughput rate is dependent solely in its deviation from the processing rate of the receive node [1]. Now, the application processing rate is, in general, not a constant. In fact, it is a varying parameter. Assuming a processing rate at the receiver of PR = $Ao(1 + sin\omega t)$ (where ω is the frequency of oscillations in radians and Ao is the amplitude of the oscillations), it can be further shown that [1]

$$TCP\ Rate = \min\left({}^{1}/_{RTT}\, Bo, Ao\left[1 + \frac{1/RTT}{\sqrt{1/RTT^2 + \omega^2}} * \sin(\omega t - \varphi)\right]\right) \quad (2)$$

From (2), the TCP rate will be a constant Bo/RTT or will grow as demanded by the receiving node. The key takeaway is that the TCP rate is dependent not only on the receiving buffer size, but also on the receiving node's processing frequency fluctuations and the amplitude of the fluctuations. One of the obvious limitations observed from (1) and (2) is the integral only aspect of flow control. Integral only controllers present very slow responses to errors/disturbances. Another key limitation is that if the receiver's buffer capacity (Bo) is not large enough to

accommodate the processing rate of the receiving node, the TCP throughput cannot exceed Bo/RTT. It is then of utmost importance that these constants be characterized in the TmNS in order to better understand the end-to-end performance and possible problem areas.

TCP CONGESTION CONTROL

The other control mechanism present in TCP is congestion control. Congestion control in TCP consists of sources slowly "probing" the network for available capacity by linearly increasing their transmission rates and exponentially reducing those rates when congestion is detected, as depicted in Figure 1 [2]. TCP "senses" network congestion through the detection of packet losses. This is very important to consider when analyzing throughput and latency across the TmNS.

The TCP connection can be analyzed in phases. The first phase, called slow-start, is the phase where the source slowly starts transmitting packets (with a small congestion window size) and then increases the window size linearly as acknowledgements (ACKs) of those packets are received. Again, it is clear from this description that a feedback control system is in place, with the congestion window size being controlled based on feedback on the state of the system (i.e. acknowledgements from the receiver, which are an indication of network congestion). As ACKs are received, the receive window continues to increase linearly until a threshold is reached, at which point the second phase of the TCP connection, the congestion avoidance phase, starts.

During the congestion avoidance phase, the window size continues to increase, but in a much slower fashion, as long as ACKs continue to be received. Whenever a loss is detected, the protocol adjusts the congestion window (i.e. reduces it) and enters the slow-start phase again (TCP Tahoe). Refinements in TCP Reno improve this process to allow for a faster recovery after a loss is detected [2]. It is important to note, however, how packets losses (or timeouts caused by a slow network) can significantly impact the performance of a TCP connection.

Now that the basic congestion control concepts of TCP are clear, let us analyze the control system involved in more detail. We can model the network as L communication links shared by N sources. Each source i transmits at a rate $y_l(t)$, so that the aggregate flow at each link is [2]

$$y_l(t) = \sum_i R_{li} x_i(t - \tau_{li}^f) \quad (3)$$

$$\text{where } R_{li} = \begin{cases} 1 & if\ source\ i\ uses\ link\ l \\ 0 & otherwise \end{cases} \quad (4)$$

It is convenient to analyze this system as an optimal control problem. The optimal control problem comprises of finding a control law for a certain system that minimizes a pre-determined cost functional (or maximizes a utility functional). The cost functional is a function of the states of the system and the control law. As such, the cost index serves the purpose of penalizing deviations of the system states and control law from a previously defined operating point. In the TCP control problem, the goal is to maximize the combined utilization of a link across all sources while being subject to certain link constraints (2). Equation (5) depicts the control

problem, where $U_i(x_i)$ can be interpreted as the benefit the source obtains for transmitting at rate x_i, and q_i is the price the source "pays" for using the link. As such, the optimal control problem maximizes the "profit" for the source [2].

$$\max_{x_i} U_i(x_i) - x_i q_i \quad (5)$$

Therefore, as we look at the TCP congestion control problem, the network resources represented by the links l assign a price (p_l) to each link based on the aggregate rate of the link. The prices associated with each link can then in turn be used by the sources to set their transmission rates $x_i(t)$. Thus, in order to implement the congestion control mechanisms, the following need to be accomplished: sources need to adjust their rates based on the link prices, and links must adjust their prices based on their aggregate rates [2]. In real networks, the price assignment of each link p_i is accomplished through Active Queue Management (AQM) algorithms present in routers. The setting of the transmission rates x_i for each source is accomplished using the TCP protocol [3].

It can be shown [2] that the average model for TCP Reno is (please refer to [2] for derivation details).

TCP	Model	
Reno	Source control	$\dot{x}_i = \frac{1-q_i(t)}{\tau_i^2} - \frac{1}{2}q_i(t)x_i^2(t)$
RED	Link control	$\dot{b}_i = \begin{cases}(y_i(t)-c_i) & \text{if } b_i(t)>0\\ [y_i(t)-c_i]^+ & \text{if } b_i(t)=0\end{cases}$ $\dot{r}_i = -\alpha_i c_i\,(r_i(t)-b_i(t))$ $p_i = m_i(r_i)$
	Utility	$U_i(x_i) = \frac{\sqrt{2}}{\tau_i}\tan^{-1}\left(\frac{\tau_i x_i}{\sqrt{2}}\right)$

Figure 3. TCP/AQM Models [2]

The source control model captures how sources adjust their rates based on link prices, where $q_i(t)$ are the end-to-end loss probabilities. The link control model captures how the links adjust their prices based on link aggregate rates, using, in this example, the Random Early Detection (RED) AQM mechanisms. Finally, the utility model captures the utility functional that is to be maximized in the optimal control problem (i.e. congestion control problem). Similarly to flow control, the constants associated with the congestion control models need to be characterized for the TmNS. That characterization will in turn assist in the identification of areas that could improve the end-to-end performance of the TmNS.

OTHER CONTROL MECHANISMS

Clearly, TCP control systems are of critical importance when analyzing network performance in the TmNS. However, many other control systems exist, such as Quality of Service (QoS), memory management, scheduling mechanisms, among others. Some of these control systems will be briefly discussed in this section.

The TmNS is a heterogeneous network, as it needs to support different types of applications and diverse traffic shapes and characteristics. This heterogeneity may require that appropriate QoS

mechanisms be in place in order to support acceptable end-to-end network performance (i.e. latency). While many QoS mechanisms exist, they all can be modeled using the basic control elements: sensors, controllers, and actuators. The main QoS mechanism used in the TmNS is Differentiated Services (DiffServ) technology. DiffServ uses a field in the header of IP packets (Differentiated Services Code Point field) in order to classify packets. This classification of packets can then be used by network devices (such as routers) to prioritize certain classes of traffic over others.

Examining QoS in the TmNS as a control system, routers are clearly the key components in the system. Sensor data in the system include queue length, loss rate, bandwidth sensors, among others. Based on sensor data and DiffServ rules, routers compute forward and drop decisions and subsequently take the appropriate action. Referring to Figure 1, the sensor data (queue length, etc.) is the feedback data in the system. The controller is achieved by the DiffServ rules, which then act upon the system (the queues in the router) to determine which packets are forwarded (or dropped) and in what order of priority they are forwarded.

Another type of control system in the TmNS is real-time scheduling. Scheduling involves deciding how to allocate system resources between multiple applications and threads. Scheduling can also involve network scheduling. There are many examples in literature showing not only how control theory applies to scheduling, but how it can optimize scheduling [4].

Yet another control system present in the TmNS is the Link Manager, which is a time division multiple access (TDMA) controller. The Link Manager is responsible for optimized control and coordination of radio operation across multiple ground and airborne radios in the Radio Access Network (RAN). It dynamically allocates the RF network capacity provided to all tests using one TDMA domain using one frequency, so as to efficiently share the constrained RF resource. The Link Manager allocates slots for RF transmission among all uplinks and downlinks between radios located in Ground Stations and on Test Articles. The dynamic allocation of RF network capacity is a control system. Based on several types of sensor data, such as queue lengths in the radios, and a given set of rules, such as priorities, the link manager needs to control the allocation of slots for RF transmissions.

NOISE AND PERTURBATIONS

It is clear that many control loops coexist in the TmNS. While there are additional control loops existent in the TmNS, the control systems discussed in this paper are the ones that ultimately may impact end-to-end performance of the TmNS. It is important to now consider sources of noise in the system, as they perturb the control loops in the TmNS.

A key source of noise in the TmNS is the bidirectional RF link. The RF link introduces losses, and therefore noise, that would otherwise not happen in wired networks. In wired networks, losses are caused almost exclusively due to congestion. The introduction of the wireless link introduces losses and errors that are native to the wireless medium.

Competition among TCP streams is another source of perturbation. It has been shown [5] that the congestion control mechanisms of some TCP implementations have scalability problems as the number of connections increase. Furthermore, without proper design, a network with multiple TCP connections may experience a phenomenon called TCP global synchronization, in which sources will simultaneously enter the slow-start phase in highly congested networks (i.e. whenever a loss is detected) and then enter the congestion-avoidance phase and cycle back and forth in this fashion, significantly impacting network performance and utilization. As a result, tests need to be conducted to determine the amount of parallel TCP connections the TmNS can support, and, in the event the number of parallel connections is not sufficient for the TmNS needs, appropriate mitigating strategies need to be devised.

Other sources of noise include interactions between link layer Automatic Repeat reQuest (ARQ) and the TCP protocol; encryption mechanisms, which add delay and overhead to all packet transmissions; IP stack perturbations; among others.

CONSIDERATIONS FOR THE TMNS

Now that the main control loops and sources of noise in the TmNS have been identified, the obvious question comes to mind: What does this all mean when it comes to developing the TmNS demonstration system (or any network-based instrumentation system) and why should we be concerned with these control loops?

A first consideration is the interactions between the RF link and the TCP protocols. As discussed in previous section, TCP determines network congestion based on packet losses. That is because, in wired networks, packet losses occur almost exclusively due to network congestion. However, once the RF link is introduced into the picture, misinterpretations of packet losses (or timeouts due to a very slow connection) over RF links as congestion losses may lead to significant network throughput degradation [6]. Recalling the congestion control mechanism discussion, anytime a loss is detected, certain TCP implementations will exponentially decrease the window size and return to the slow-start phase of the connection. If many losses/timeouts are incurred in the RF link of the TmNS, those losses could significantly affect network throughput and end-to-end latency of PCM dropout recovery data. While there are many TCP implementations that improve on this issue, through fast recovery and other mechanisms, it is important to look at which TCP implementation (Tahoe, Reno, Vegas) best suits the needs of the TmNS.

Another key consideration is how packet losses occur. Typical AQM router techniques, such as DropTail, drop packets once they arrive at a full buffer. However, this can lead to TCP global synchronization issues. A possible alternative is to use RED queuing disciplines on the routers, which drop packets based on probabilities that increase based on queue length [7]. As such, router queuing disciplines need to be investigated to optimize the performance of the TmNS.

Yet another consideration related to TCP is how it "senses" congestion. Since the congestion control of the most widely used TCP implementation (Reno) relies on packet loss to "sense" congestion, by the time congestion is detected, packet losses must already have occurred. In other words, TCP *reacts* to packet losses versus *preventing* them. In order to improve on this shortcoming, the use of Explicit Congestion Notification (ECN) can be considered. ECN is an optional feature of the TCP/IP suite that is only used when both endpoints support it and are willing to use it. When ECN is successfully negotiated, an ECN-aware router may set a mark in the IP header instead of dropping a packet in order to signal impending congestion. The receiver of the packet echoes the congestion indication to the sender, which must react as though a packet was dropped [8]. If congestion becomes frequent over the TmNS RF link, the use of ECN may be beneficial.

Directly related to effects of RF links on TCP connections is link layer ARQ. It has been shown [9] that the error rate seen by the TCP layer can be improved by the use of Forward Error Correction (FEC) and ARQ at the link layer. However, suitable parameters for ARQ need to be considered in order to achieve performance enhancements. In fact, inappropriate ARQ settings could be detrimental to overall TCP performance [9]. As such, the use of ARQ at the link layer simultaneously with TCP at the transport layer in the TmNS needs to be further investigated.

TCP flow control could also affect the overall network performance of the TmNS due to fluctuating application processing rates, zero window advertisements and network fluctuating rates. However, these issues can be more easily mitigated by ensuring end nodes, such as telemetry processors, have the appropriate resources to handle the required amounts of data. Nevertheless, it is important to ensure that those resources are allocated appropriately, as inappropriate buffer sizes at receiving nodes have been shown to introduce more than 60% degradation in TCP throughput [1].

Finally, another key point to consider in the TmNS is QoS mechanisms. While DiffServ presents a promising mechanism for achieving end-to-end performance requirements for low latency data, such as Voice over IP (VoIP), studies need to be conducted in order to ensure those requirements can be met with DiffServ alone. If not, the use of more refined QoS mechanisms, such as QoS controllers (as discussed in [10], [11]), that aim at maximizing performance across multiple applications based on high level priority requirements specified by a user, may need to be considered.

CONCLUSION

This paper exposes a network as a variety of control systems, some of which are in competition with each other. It might appear that, given the advanced networking technologies available today, building and setting up a network is not very complex. However, once the low-level systems that "power" networks are analyzed, it should be clear that designing a network with limited bandwidth (i.e. RF link) and harsh end-to-end latency constraints (100ms – 200ms) requires a deep understanding and mitigation of the limitations of the underlying processes.

Let us take a look again at an instantaneous TCP throughput graph of RC sessions across a constrained link (e.g. RF link), as shown in Figure 1. Imagine that data needed to be transferred from an end node to another across that network in a time not to exceed 150ms. Figure 1 should clearly indicate that, unless the significant drops in throughput are mitigated, the 150ms end-to-end latency requirement may be jeopardized. As such, a proper analysis of the control systems involved needs to be conducted, which would aid in identifying the root causes of throughput drops and implementing the proper mitigation techniques to achieve the required performance.

The success of iNET depends on a system that delivers data and does so in a timely fashion. Data delivery will include RC data (to fix PCM dropouts), VoIP data, etc. As such, the ongoing control system analysis study being performed will provide information that is critical to understanding the capabilities of the TmNS in providing the resources for data to be delivered on time. The study will also identify areas of possible shortcomings, along with recommendations to address them.

ACKNOWLEDGEMENTS

The work described in this paper would not have been possible if not for the contributions from the many members of the iNET Standards Working Groups. We gratefully acknowledge this effort and the funding and guidance provided by the iNET program.

REFERENCES

[1] Sanadhya, S., and R. Sivakumar. "Adaptive Flow Control for TCP on Mobile Phones."
[2] Low, S.H., F. Paganini, and J. C. Doyle. "Internet Congestion Control." IEEE Control Systems Magazine, pp. 28-43, February 2002.
[3] Papachristodoulou, A., J. C. Doyle, and S. H. Low, "Analysis of Nonlinear Delay Differential Equation Models of TCP/AQM Protocols using Sums of Squares."
[4] Stankovic, J.A., Chenyang Lu, Son, S.H., and Gang Tao. "The case for feedback control real-time scheduling." www.cs.virginia.edu/~stankovic/psfiles/fcedf.ps
[5] Ohsaki, H., M. Masayuki, and H. Miyahara. "Application of control theory to a window-based flow control mechanism with ECN routers."
[6] Bai, Y., A.T. Ogielski, and G. Wu. "Interactions of TCP and Radio Link ARQ Protocol."
[7] Floyd, S., and V. Jacobson. "Random Early Detection Gateways for Congestion Avoidance." IEEE/ACM Trans. Networking, vol. 1, pp. 397-413, Aug. 1993. Available: ftp://ftp.ee.lbl.gov/papers/early.ps.gz.
[8] http://en.wikipedia.org/wiki/Explicit_Congestion_Notification.
[9] Chockalingam, A., M. Zorzi, and V. Tralli. "Wireless TCP Performance with Link Layer FEC/ARQ."
[10] Kim, W., P. Sharma, J. Lee, S. Banerjee, J. Tourrilhes, S. Lee, and P. Yalagandula. "Automated and Scalable QoS Control for Network Convergence."
[11] Hoang, D. "Application of Control Theory in QoS Control."