

CONNET: Self-Controlled Access Links for Delay and Jitter Requirements

Mohamed El Gendy[†] Kang G. Shin[†] Hosam Fathy[‡]
[†]EECS Department [‡]Mechanical Engineering Department
University of Michigan, Ann Arbor, MI 48109
{mgendy,kgshin,hfathy}@umich.edu

Abstract

Access links are typically the bottleneck between a high bandwidth LAN and a high bandwidth IP network. Without a priori resource provisioning or reservation, this tends to have a serious effect on premium traffic from delay- and jitter-sensitive applications. This paper develops a new reservation-less mechanism for delay and jitter guarantees across “single FIFO queue” access links by controlling interfering non-premium traffic. The mechanism, called CONNET (CONtrolled NETwork), uses feedback regulation to create self-controlled network services capable of tracking delay and jitter references. We use a control-theoretic approach in designing the feedback loop based on a “black-box” model of the access link in both cases of single and multiple nodes. For robust control, we employ optimal Linear Quadratic Gaussian (LQG) regulator. We evaluate the performance of CONNET using a real testbed implementation showing its superiority to other rate-based schemes (such as CBQ and WFQ) in terms of simplicity, deployability, and accuracy.

1. Introduction

The Internet has become the main communication vehicle for many of our daily-life applications, such as online shopping, banking and gaming, Voice over IP (VoIP) (e.g., IP telephony), and high definition TV (HDTV). These applications impose throughput, delay, and jitter requirements on the Internet links and routers carrying their traffic. Providing predictable, as well as controllable, delay and jitter at all times is still a great challenge even with the significant increase in network bandwidth. With the existence of large amounts of competing traffic at bottleneck segments of the network, and in the absence of smart bandwidth management, applications have only limited control over the network performance.

Currently, customer networks employ high-bandwidth technologies ranging from 10/100 Mbps Ethernets to opti-

cal links, such as OC3 (155 Mbps) and OC12 (622 Mbps), to even Gigabit Ethernets. On the other hand, Internet Service Provider (ISP) networks, or more generally IP backbones, are usually equipped with even higher capacity using OC48 (2.4 Gbps), OC192 (10 Gbps) or 1 Tbps WDM fibers [12]. Access links (e.g., cable, xDSL, T1) between customer networks and their ISPs still have bandwidths of at most a few megabits per second. This creates a bottleneck between the two high-bandwidth networks, and hence, makes a strong negative impact on the performance of delay-sensitive applications at medium- and high-utilization levels of access links [17–19]. Moreover, with the current Internet’s first-in-first-out (FIFO) access links/routers there is not much to do in order to protect time-sensitive traffic from being delayed and jittered at these bottlenecks. Large background HTTP downloads can cause an annoying delay to an interactive or a real-time streaming application running through the same network.

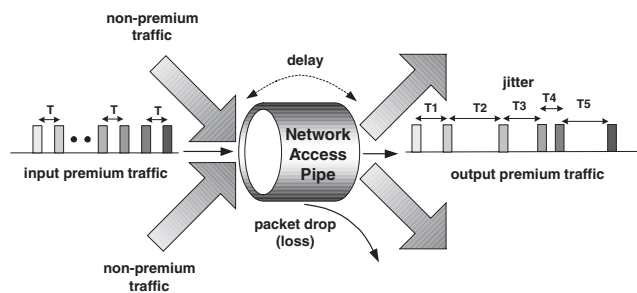


Figure 1. Delay and jitter across a network access link

This paper presents a novel way, called CONNET (CONtrolled NETwork), of providing controllable access links for delay- and jitter-sensitive premium traffic. It is based on a simple, yet intelligent bandwidth control for traffic going through the access links. Consider a network pipe—representing a single or multiple hops—shown in Figure 1, as an example access link, which carries two classes (i.e.,

premium and non-premium) of traffic. Assuming static link capacity and FIFO queueing (used in the current Internet), the delay and jitter, as well as the throughput, of premium traffic will depend directly on the amount of non-premium traffic sharing the network pipe with it. This is because, for single FIFO-queue networks, non-premium packets in front of premium packets cause them to experience variable queueing and service delays. We first characterize (with a model) the relationship between the amount of non-premium traffic and the delay and jitter of premium traffic. We then use this model to control the competing non-premium traffic in order to achieve the required premium delay and/or jitter guarantees. We take a control-theoretic—specifically model-based control—approach to the access link control problem, and present a full design and analysis of robust estimation and control.

Our contributions are two-fold: (i) development of a *reservation-less* approach for delay and jitter control that does not require prior knowledge of input traffic specification, and works well with currently-prevalent FIFO routers, and (ii) creating dynamic as well as *self-controlled* access links that react to changing traffic loads while maintaining high link utilization. We evaluate this scheme using an implementation on a real testbed network, demonstrating the correctness, accuracy, robustness, and fault-tolerance of the thus-designed controller. Its performance is compared against other well-known schemes, such as classed-based queueing (CBQ) [6] and weighted-fair queueing (WFQ) [4]. Based on this evaluation, we found CONNET making a significant (more than 40% in some cases) improvement in preserving low delay and jitter for premium traffic.

The rest of the paper is organized as follows. Section 2 presents the rationale, and introduces the main theme, of CONNET. Section 3 describes how to derive a model for the access link under study while Section 4 applies this model to the design of feedback control for delay and jitter guarantees. We describe the implementation of the control algorithm and the control components in Section 5. Section 6 presents our experimental evaluation, confirming the correctness and effectiveness of CONNET and comparing its performance against other well-known rate-control schemes. It also states the assumptions and current limitations of the current version of CONNET. Related work is discussed in Section 7, and the paper concludes with Section 8, providing insights into the results obtained as well as future directions.

2. Rationale and Main Theme

To develop a feel for the need of access link control, or CONNET, we consider our campus network, shown in Figure 2. From the EECs Department, we used traceroute to `www.google.com`, and found it tak-

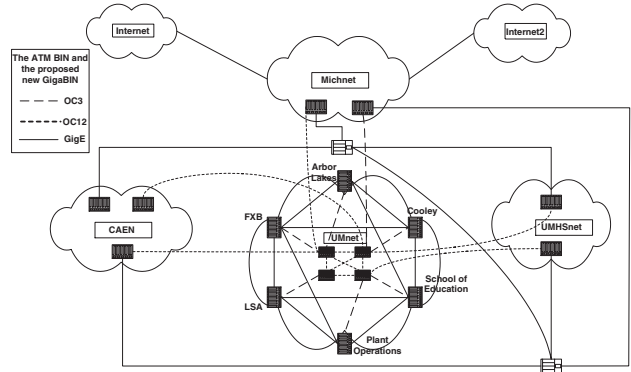


Figure 2. The University of Michigan campus network topology

ing 5 hops to reach `mich.net`, the regional ISP for the University of Michigan, at access router AA1 (`ge-1-1-0x984.aal.mich.net`). Specifically, we obtained the following result:

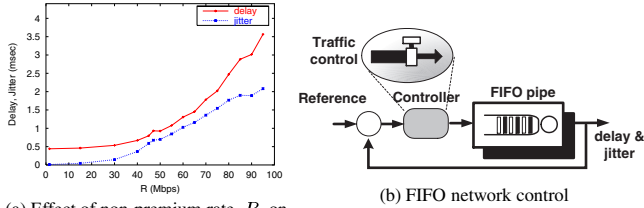
```
> 1 eecs2n-gw 0.488 ms
> 2 141.213.127.37 0.608 ms
> 3 caen-bin.r-bin-seb.umnet.umich.edu 0.711 ms
> 4 bin-arb.r-bin-arb.umnet.umich.edu 0.842 ms
> 5 ge-1-1-0x984.aal.mich.net 0.922 ms
> 6 ge-1-2-0x25.nl-chi3.mich.net 7.138 ms
> 7 198.110.131.78 7.717 ms
```

Within each segment LAN environment, there is enough bandwidth, e.g., the link speed of CAEN (Computer-Aided Engineering Network) varies between OC12 and OC48, hence a small delay. However, the delay increases significantly (~ 7 ms) at the 6-th hop between access router AA1 and the next hop in `mich.net`. This indicates that the access link (equivalently, the access router) contributes a significant percentage of the end-to-end delay and must, therefore, be controlled in order to achieve overall predictable delay and jitter performances. In what follows, we detail the two salient features of CONNET.

2.1. Reservation-less Delay and Jitter Control

CONNET requires neither advance resource reservation nor *a priori* traffic parameterization. Furthermore, it does not assume any special queueing discipline other than single-FIFO-queue routers, unlike the common scheduling algorithms that serve multiple queues for different traffic classes with the disadvantage of configuration complexity.

For a typical FIFO network path with two types of input traffic, premium and non-premium traffic, sharing the same FIFO queue as shown in Figure 1, when we vary the rate of non-premium traffic while keeping all other configurations fixed, and plot both the measured premium delay (d) and



(a) Effect of non-premium rate, R , on premium delay and jitter

Figure 3. Control criterion

jitter (j) along with the rate of non-premium traffic (R), we get a relationship similar to the one in Figure 3(a), which is the basis for our reservation-less control as well as the “physics” of the system under study. We call it the “characteristic curve,” which determines the feasible delay and jitter under the current configuration and with the available range of non-premium traffic rate. We control the rate (R) of non-premium traffic sharing the network resources with premium traffic in order to achieve the required delay and jitter for premium traffic. Note that R is not the only factor affecting the premium traffic delay and jitter. However, our previous study in [5] has shown that it has the greatest effect, and hence, suffices to be the control input.

2.2. Dynamic and Self-Controlled Links

CONNET employs *feedback control* that yields faster response to traffic changes. This creates self-controlled network paths that do not require the operator’s intervention to tune their performance each time the network workload changes. The network operator only needs to set references for the output, and then, the feedback controller will be able to track these values. This tracking also works with variable references, meaning that the operator can provide a reference signal reflecting the scenario of operation during a certain time interval (time-of-day or day-of-week). This would not be easy to achieve with operator-controlled links or other commonly-used techniques.

2.3. A Control-Theoretic Approach to Access Link Control

The resemblance of this network control problem to fluid mechanic problems motivated us to investigate the use of a control-theoretic approach to designing the control algorithm for the delay and jitter across the access link. Following this approach, we first obtain a model for the relationship between the amount of non-premium traffic, in terms of average bit rate, and the premium delay and jitter, and then use *state-space* methods in the controller design. State-space is recommended for digital control as in our network problem, and also for multiple-input-multiple-output (MIMO) systems [7]. We use a Linear Quadratic Regulator (LQR)

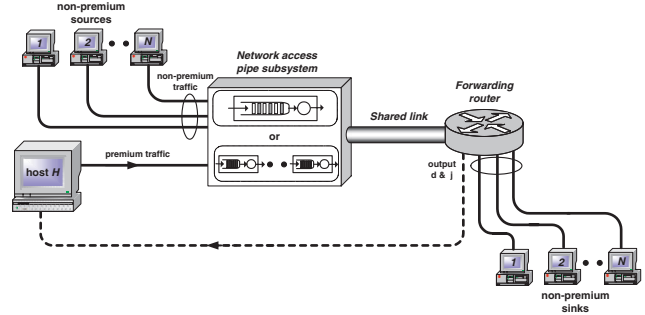


Figure 4. Network setup

controller along with the Kalman filtering as a combination of robust estimation and control [7]. Figure 3(b) illustrates the idea of control around the network pipe. We will study single- and multiple-node network pipes using the same approach, showing its generality.

3. System Modeling and Validation

We are mainly interested in a “black-box” model that describes the system behavior in terms of its inputs and outputs only, and this process is called *system identification*. Since we use the state-space method for the design and analysis of the network-control algorithm, we need a state-space model for the system under study. We use a state-space system identification method called the *subspace modeling* [14], which extracts a system model from traces of its inputs and outputs. A discrete-time “state innovation” model

$$\begin{aligned} \mathbf{x}_{k+1} &= \mathbf{A}\mathbf{x}_k + \mathbf{B}R_k + \mathbf{G}\mathbf{e}_k \\ [d_k, j_k]^T &= \mathbf{C}\mathbf{x}_k + \mathbf{D}R_k + \mathbf{e}_k \\ \mathbf{V} &= E(\mathbf{e}_p\mathbf{e}_q) = \text{cov}(\mathbf{e}_p) \end{aligned} \quad (1)$$

is the output of the subspace modeling, where the input is $u_k = R_k$, or the non-premium traffic bit rate, and the output is the premium delay and jitter, $\mathbf{y}_k = [d_k, j_k]^T$. The state vector, \mathbf{x}_k , consists of variables that describe the internal condition of the network subsystem. For example, the state variables could be the length of the queue inside the access link nodes and the rate of change of this length. For any dynamic system, there exists an infinite number of choices of state variables and system identification does not necessarily select state variables with physical meaning. Therefore, we will not be able to assign a particular meaning to \mathbf{x}_k . Only the input and output vectors, \mathbf{u}_k and \mathbf{y}_k , will have well-defined physical meanings in this paper. The subspace system identification provides the system matrices, \mathbf{A} , \mathbf{B} , \mathbf{C} , and \mathbf{D} , that describe how the input affects the state variables and the output. Moreover, the subspace modeling quantifies the measurement error, \mathbf{e}_k , and system noise, $\mathbf{G}\mathbf{e}_k$ in terms of the covariance (cov) matrix of the measurement error, \mathbf{V} . Subspace modeling also allows us to specify

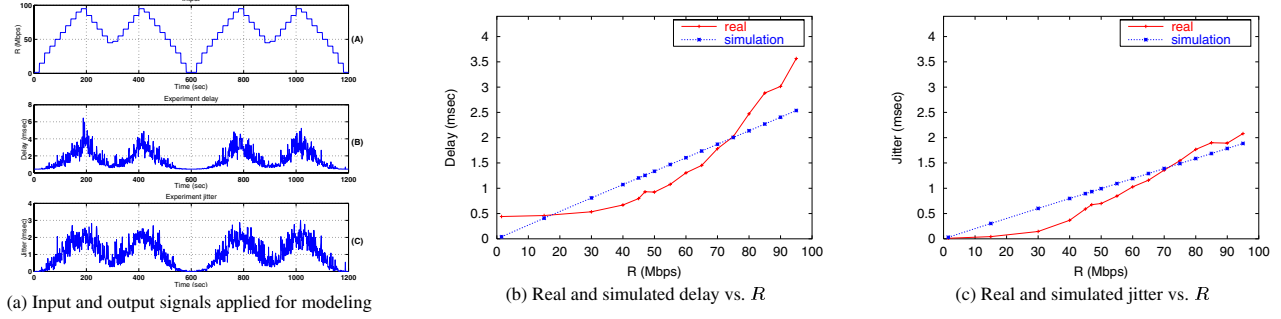


Figure 5. Model extraction and validation

a particular, as well as a recommended, system order n .¹

3.1. Experimental Setup

We use the experimental network testbed shown in Figure 4 for both modeling (i.e., calculating the above system matrices) and evaluation. It consists of Linux-based software routers and end-hosts. The traffic controls built into the Linux kernel [2] enable construction of FIFO queues as well as traffic regulators that are used to enforce the control signal. A fast Ethernet-based ring network topology, with link capacity of 100 Mbps, is used so that the one-way delay may be measured without sophisticated (and sometimes inaccurate) time synchronization such as NTP or GPS. The ring topology was built using Linux *iptables* installed on the forwarding router. Premium traffic is generated at host H , traverses one or more access routers (representing the access link under study), then the forwarding router, and finally, terminates at host H again (i.e., forming a ring topology). Hosts $1, \dots, N$ are used to generate non-premium traffic that shares the links and FIFO router(s) with the premium traffic. We use multiple non-premium sources to mimic a real network where traffic comes from multiple subnets sharing the same access link. All measurements, analysis, modeling and control calculations are done on host H . We use “non-responsive” UDP traffic sources that can be instructed to generate traffic according to a specific input signal chosen based on a given experiment scenario regardless of the losses or delays at the bottleneck. This allows us to control the exact non-premium rate without interference from either congestion- or flow-control mechanisms. However, in Section 6, we also evaluate CONNET with TCP traffic and show its effectiveness even with the TCP congestion control algorithm.

3.2. Model Extraction

In the first phase of experiments, our main goal is to calculate the system matrices of the model in Eq. (1). We use

an input signal R depicted in top plot of Figure 5(a) for estimating the model parameters. It consists of a ramp-up-ramp-down traffic sending rate that ranges from 1.5 Mbps to 95 Mbps of non-premium traffic during a small constant-rate period. This rate is always higher than that of premium traffic which is a 1 Mbps CBR.² All traffic sources use same packet length of 1000 bytes. The corresponding measured output delay and jitter for the single FIFO node case are depicted in the lower plots of Figures 5(a).

One-way delay (d) and jitter (j) are measured at host H , and sampled every T_s by a timer-operated traffic monitor listening on the receiving network interface. We use $T_s = 1$ sec, or equivalently a sampling frequency of 1 Hz. The UDP header³ in each packet carries time-stamps to enable delay and jitter calculations. Delay and jitter measurements also pass through a weighted moving average (WMA) filter calculated for jitter, as an example, as $j = j + (|d(i-1, i)| - j)/8$, where $d(i-1, i)$ is the delay variation between packets i and $(i-1)$. To determine the order of the system model, we tried several options, and found from experiments that choosing a second order model (i.e., $n = 2$) is good enough to capture the dynamics of the system and achieve the required goal. Accordingly, the thus-acquired systems matrices for the single-node case are:

$$A = \begin{bmatrix} 0.7978 & 0.6260 \\ 0.0790 & 0.2525 \end{bmatrix}, B = \begin{bmatrix} -0.0243 \\ 0.0275 \end{bmatrix}$$

$$C = \begin{bmatrix} -0.5002 & 0.3134 \\ -0.1988 & -0.1957 \end{bmatrix}, D = \begin{bmatrix} 0.0108 \\ 0.0250 \end{bmatrix}$$

$$G = \begin{bmatrix} -0.2041 & -0.3298 \\ 0.2015 & -1.2981 \end{bmatrix}, V = \begin{bmatrix} 2.8306 & 0.0694 \\ 0.0694 & 0.0747 \end{bmatrix}.$$

From this model, we calculate the open-loop system’s discrete-time poles to be 0.8770 and 0.1733, which indicates a stable system model (poles are inside the unit circle). In order to get a better view of the difference between the model behavior (from MATLAB simulation) and the real

¹By calculating the singular values of the system.

²Real-time applications usually send constant-bit-rate UDP traffic.

³We use RTP-like headers.

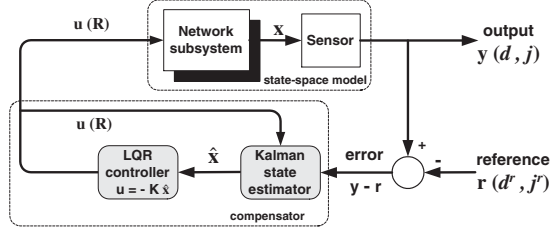


Figure 6. Feedback control loop

system behavior (from experimental data), we plot the average values of the output delay and jitter versus the input R for both outputs in Figures 5(b) and 5(c); the resulting model is linear, while the real system is not. However, the model is the best-fit for the experimental data and is good enough for building the feedback control.

4. Design of Feedback Control

The feedback-control loop around an access link is to enforce the user-/application-specified delay and jitter *references* of the premium traffic. This reference-tracking control is required to have a smooth response at an adequate speed. Large overshoots or undershoots would make a negative impact on the network traffic performance, especially with responsive congestion- and flow-control protocols, such as TCP.

In the state-space digital control design, unmeasurable states (\mathbf{x}) of the system, as in the case of access links, are estimated ($\hat{\mathbf{x}}$) using the measured input–output samples of the system.⁴ The estimated states are then used to generate the feedback-control signal to act as input to the system for the next sampling period. In order to overcome inaccuracies in the system model (e.g., caused by linearization), robust control is preferred to other regular control designs. Robust control can also deal with changes and uncertainties in the system model and input conditions. The LQR (Linear Quadratic Regulator) controller [7] fits well this design criterion and achieves a good balance between system response and the control effort required. For state estimation, we employ the Kalman filter [7], which works well with experimentally-derived models.⁵ By putting all of these blocks together, the feedback control loop is formed as illustrated in Figure 6.

4.1. Design of the LQR Controller

The control law, based on the system's estimated state vector, $\hat{\mathbf{x}}$, is given by:

$$R_k = -\mathbf{K}\hat{\mathbf{x}}_k \quad (2)$$

⁴This is possible because the system's dynamic model is observable.

⁵It accounts for process noise and measurement errors.

where \mathbf{K} is the control gain to be designed based on the system matrices, \mathbf{A} and \mathbf{B} , and two weighting matrices, \mathbf{Q}_1 and \mathbf{Q}_2 , that minimize a certain cost function [7]. We use MATLAB to calculate the controller gain as:

$$\mathbf{K} = [-15.7017, -9.3098].$$

4.2. Design of the State Estimator

The states estimates ($\hat{\mathbf{x}}$) are iteratively calculated in terms of successive samples of output \mathbf{y}_k or $[d_k, j_k]^T$, reference \mathbf{r}_k or $[d_k^r, j_k^r]^T$, previous-step input, and previous-step state estimate as:

$$\hat{\mathbf{x}}_{k+1} = (\mathbf{A} - \mathbf{L}\mathbf{C})\hat{\mathbf{x}}_k + (\mathbf{B} - \mathbf{L}\mathbf{D})\mathbf{u}_k + \mathbf{L}(\mathbf{y}_k - \mathbf{r}_k) \quad (3)$$

where $\hat{\mathbf{x}}_{k+1}$ and $\hat{\mathbf{x}}_k$ are the state estimates at step $k+1$ and k , respectively. \mathbf{A} , \mathbf{B} , \mathbf{C} , and \mathbf{D} are the system matrices from the model (i.e., Eq. (1)), while \mathbf{K} is the controller gain derived above and \mathbf{L} is the estimator gain. This formula is used to update the states in real time during control. The initial state, $\hat{\mathbf{x}}_0$, is set to zero, and the closed loop starts building up from that point. The estimator gain, \mathbf{L} , is optimally chosen to reduce the effects of both the process noise and the measurement error [7]. Using the MATLAB built-in Kalman filter function, we calculated the required estimator gain as:

$$\mathbf{L} = \begin{bmatrix} -0.1020 & -0.1649 \\ 0.1008 & -0.6491 \end{bmatrix}.$$

4.3. Simulation and Verification

In order to simulate a closed-loop system along with its controller and estimator, we manipulate Eqs. (1), (2), and (3) to get the closed-loop dynamics in terms of the original and estimated states:⁶

$$\begin{aligned} \begin{bmatrix} \mathbf{x}_{k+1} \\ \hat{\mathbf{x}}_{k+1} \end{bmatrix} &= \begin{bmatrix} \mathbf{A} & -\mathbf{B}\mathbf{K} \\ \mathbf{L}\mathbf{C} & \mathbf{A} - \mathbf{B}\mathbf{K} - \mathbf{L}\mathbf{C} \end{bmatrix} \begin{bmatrix} \mathbf{x}_k \\ \hat{\mathbf{x}}_k \end{bmatrix} \\ &+ \begin{bmatrix} 0 \\ -\mathbf{L} \end{bmatrix} \begin{bmatrix} d_k^r \\ j_k^r \end{bmatrix}, \\ \begin{bmatrix} d_{k+1} \\ j_{k+1} \end{bmatrix} &= [\mathbf{C} \quad -\mathbf{D}\mathbf{K}] \begin{bmatrix} \mathbf{x}_k \\ \hat{\mathbf{x}}_k \end{bmatrix} \end{aligned} \quad (4)$$

This system of closed-loop equations has the references $[d^r, j^r]^T$ as the input, and $[d_{k+1}, j_{k+1}]^T$ as the output, and represents the closed-loop system in Figure 6. The closed-loop system is simulated with a time-varying reference input using MATLAB, and the resulting delay and jitter are plotted in Figure 7. From this plot we observe the controller's efficiency in tracking the delay and jitter references without overshoots or undershoots. The resulting discrete-

⁶Derivation is straightforward and omitted due to space limitation.

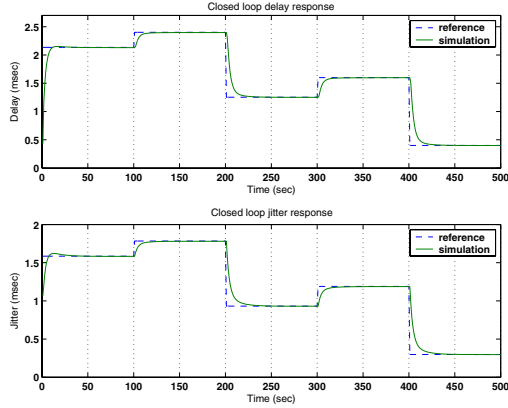


Figure 7. Closed-loop response

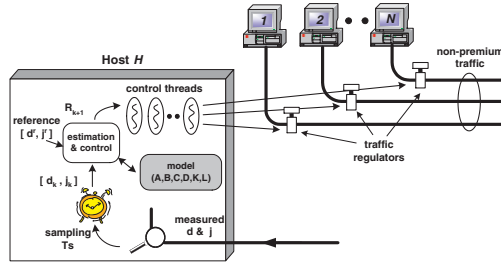


Figure 8. Control implementation

time closed-loop poles of the single-node system are 0.9169, 0.7143, 0.0081, and 0.0936, hence a stable closed loop. The closed-loop bandwidth (control frequency) is 0.7671 Hz, indicating that a sampling frequency of 1 Hz is suitable when compared to the closed-loop bandwidth.

5. Implementation of the Control and Actuation Algorithms

To control the rate of non-premium traffic, we employ traffic regulators in the form of token bucket filters applied at the non-premium sources as shown in Figure 8. This figure also illustrates the control steps executed inside host H , as described in the control algorithm of Figure 9. The traffic regulators are controlled by a signal communicated from H . The computation and communication of the control signal are to be completed within one sampling period (T_s) and hence, a separate thread is assigned to communicate the control signal to each of the non-premium traffic regulators. The measurement thread is interrupted every T_s to sample the current delay and jitter values, and calculate the control signal using a combination of Eqs. (2) and (3).

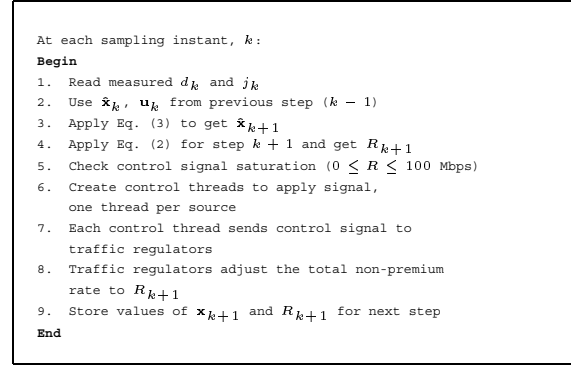


Figure 9. Control algorithm

6. Experimental Evaluation

We evaluate CONNET experimentally for different scenarios, reflecting several aspects of the controller performance. We divide the experimental results into two sets: one for a single FIFO node and the other for a multi-node FIFO pipe. Important and interesting points in the results for each scenario are identified and discussed.

6.1. Experimental Setup

We use the same testbed network illustrated in Figure 4 for experimental evaluation with 3 non-premium traffic sources, each running 2 flows⁷ with a total of 6 non-premium flows. We use 3 traffic regulators, one for each traffic source, all controlled by the control algorithm running on host H . During evaluation, unless otherwise mentioned, each non-premium traffic source sends constant-bit-rate (CBR) UDP data at the maximum rate allowed by the network, which is 100 Mbps. This allows for actual testing of the controller when the network encounters high loads.

6.2. A Single FIFO Node

For the case of a single FIFO node shared by premium and non-premium traffic, we consider the following experiment scenarios:

6.2.1. Constant Reference Tracking

In the first scenario, we study the controller's performance in tracking a constant reference for both delay and jitter. The experiment is started with traffic regulators at non-premium sources (see Figure 8) turned on while each source is sending data at the maximum link capacity of 100 Mbps, and then left on for the rest of the experiment duration of 100 seconds. We use a delay reference of 0.46 msec and a jitter

⁷To increase statistical multiplexing between packets.

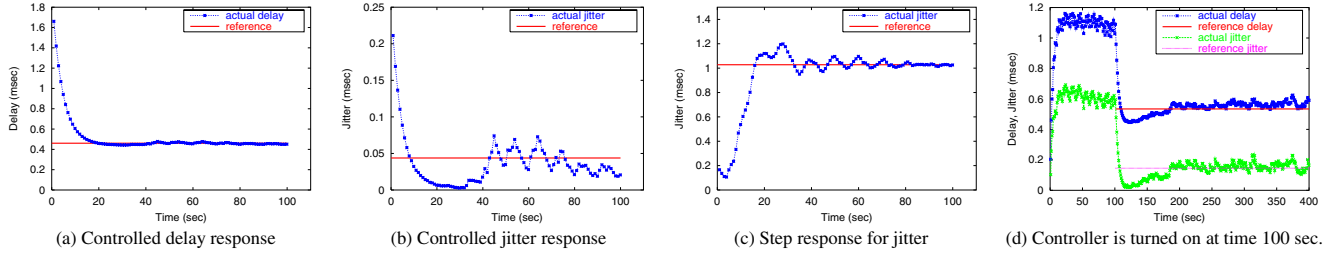


Figure 10. Constant reference tracking

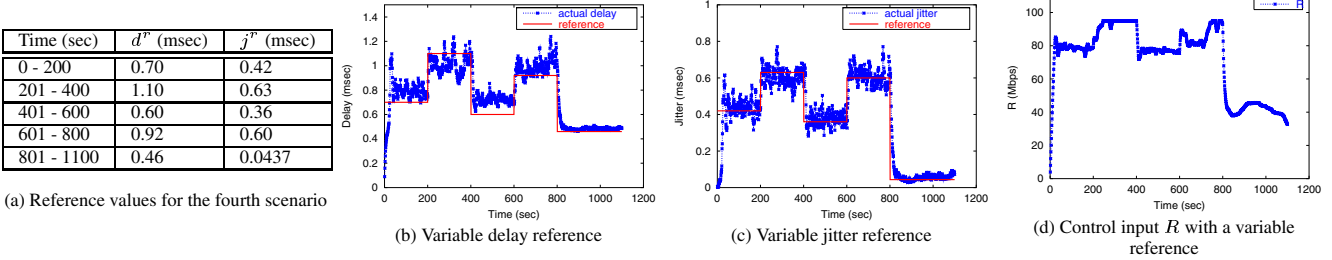


Figure 11. Variable reference tracking

reference of 0.043 msec. Figures 10(a) and (b) plot the measured delay and jitter as well as their reference values, confirming the fact that the controller can achieve the required delay.

6.2.2. Step Response

In the second scenario plotted in Figure 10(c), the step response of the system is plotted with respect to jitter as an example. A step jitter reference of magnitude 1.028 msec is applied to the controller shortly after starting the experiment, and then kept constant for the rest of the experiment duration of 100 seconds.

6.2.3. Controller Activation During Run-time

The third scenario, plotted in Figure 10(d), illustrates what happens when the controller is turned on during a heavily-loaded network operation. The experiment is started while all traffic regulators were turned off and non-premium sources are sending data at a rate of 70 Mbps resulting in a delay of around 1.1 msec and a jitter of 0.6 msec. At 100 sec (the total experiment time is 400 sec), the controller is turned on with the reference values of 0.534 msec and 0.145 msec for delay and jitter, respectively. As indicated in the figure, the controller reacts instantaneously to regulate the non-premium traffic so that the delay and the jitter may follow their reference values as closely as possible.

6.2.4. Variable Reference Tracking

To further illustrate the dynamics of the delay and jitter controllers using non-premium traffic regulation, we provide

the fourth scenario where a variable reference signal is used to reflect a certain scenario when demands and requirements change throughout the day/hour. The experiment lasts for 1100 sec, and the reference values of delay and jitter change every 200 sec. Figure 11(a) shows the reference values used in this scenario for both delay and jitter as an example of a variable reference signal. The output delay is plotted in Figure 11(b), while the output jitter is plotted in Figure 11(c). The control signal, R , is plotted in Figure 11(d), indicating saturation⁸ between times 246 and 400 sec. This is a very common case in real feedback control systems, and CONNET recovers from that.

6.3. Multi-Node FIFO Networks

For all the following experiment scenarios, we use a FIFO pipe that consists of 3 FIFO nodes connected to each other. All premium and non-premium traffic enters the pipe only via the first node, and exit from the third node. No cross traffic is introduced (see Section 6.4 for further comments).

6.3.1. Dealing with Non-premium TCP Traffic

Instead of using non-premium UDP traffic, in this scenario we use TCP sources to investigate the TCP friendliness and effectiveness of CONNET in dealing with TCP congestion control. The LQR controller works carefully not to have large overshoots or undershoots, and this feature works well with TCP. Figure 12(a) plots the performance of delay control, bringing it to the reference value. Figure 12(b) shows

⁸We set saturation level at 95 Mbps for all experiments.

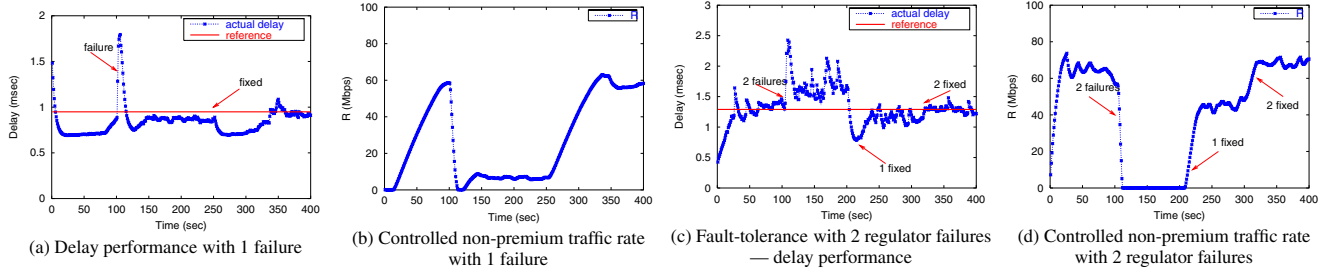


Figure 13. Fault-tolerance

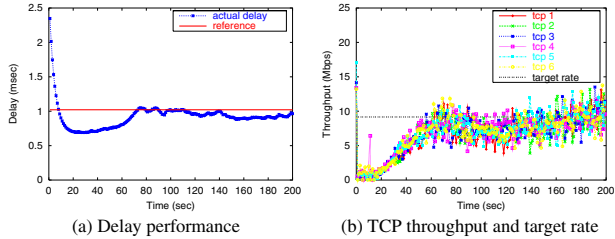


Figure 12. Non-premium TCP traffic

the resulting throughput of 6 TCP sources used in comparison with a target rate of 9.167 Mbps. The target rate indicates the actual rate needed to achieve the reference values of delay and jitter taken from Figure 3(a), divided by 6. The regulated TCP traffic could reach the same target rates and did not suffer congestion collapse or any loss of utilization, indicating that CONNET works well with TCP as well as UDP background traffic.

6.3.2. Fault-tolerance and Uneven Load Distribution to Regulators

In the following two scenarios, we present a very important and powerful feature of CONNET, “fault-tolerance.” Here we examine the case when one or more traffic regulators fail and opens the door for unregulated non-premium traffic. We first test the case of a single regulator failure, where the experiment is started with all 3 traffic regulators working, then at time 100 sec, one of the 3 regulators fails, allowing unregulated traffic at 50 Mbps to enter the network pipe from the corresponding non-premium source. Then, after another 150 sec, the failed traffic regulator is fixed and restored back to work again for the rest of the experiment duration of 400 sec. Figure 13(a) shows the delay during this scenario with the times of failure and repair indicated. Jitter has a similar performance and omitted due to space limitation. When the failed regulator is repaired at 250 sec, the network returns back to normal again and keeps delay and jitter at their reference values. Figure 13(b) shows the rate sent to the remaining operating regulators during this failure, and the controller reacts to this failure by almost shut-

ting off the other two traffic sources to keep premium traffic performance at the required level.

We investigate further what happens when 2 out of 3 regulators fail at time 100 sec, then one of them comes back to work at time 200 sec, and all three of them come back to work at time 300 sec. This is illustrated in Figures 13(c) and (d). The first figure plots the delay, showing a similar behavior at the time of failure. However, this time the controller was not able to drive the delay to the reference value during the time of 2 failures. To see why this happened, the second figure plots the output control rate from the controller. Since the control rate cannot be below zero, the controller cannot change the control signal any more and becomes helpless. When one of the failed regulators was fixed to work again, the controller catches up and sends a correct control signal to bring the output delay back to its reference value. Finally, when all the three regulators work again, the network goes back to normal. These two scenarios indicate that CONNET can survive up to one third (1/3) of regulator failures in case of our testbed network, and we also expect a similar performance when it is deployed on larger networks.

Another feature of the control algorithm is its ability to apply uneven control rates at different traffic regulators. We demonstrate this ability with the following scenario, where the control rates are distributed unevenly among 3 traffic regulators. The first one gets 50% of the control rate, the second one 35%, and the third one 15%. Again, the controller was able to work well in this case and brings both the delay and the jitter to their reference values as shown in Figure 15(a). This allows for setting up a control policy to favor some trunks over others.

6.3.3. Comparison with CBQ and WFQ

We now compare CONNET with other well-known scheduling algorithms, such as Class-Based Queueing (CBQ) [6] and Weighted Fair Queueing (WFQ) [4]. We compare the delay and jitter performance while applying a variable non-premium traffic, as shown in Figure 14(a), to the access link pipe. In this scenario, instead of FIFO queues, we set up the CBQ discipline on all of the three router nodes, where a bandwidth of 2 Mbps is allocated to premium traffic and

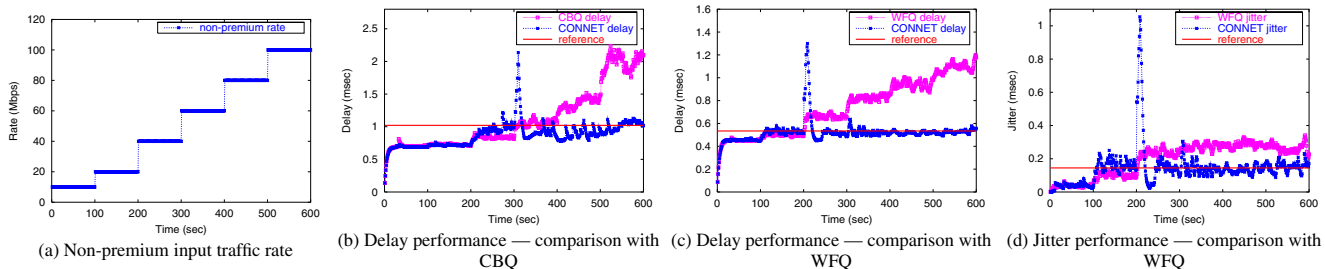


Figure 14. Comparison with CBQ and WFQ

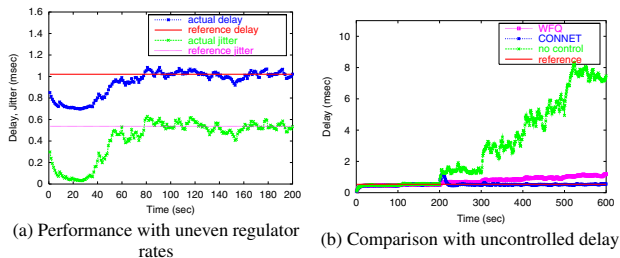


Figure 15. Controller performance

the rest is given to non-premium traffic. The delay is plotted in Figure 14(b), and in the same figure, we also plot the performance when CONNET is activated with FIFO nodes in place of CBQ. Clearly, the CBQ performs worse than CONNET, and both delay and jitter increase with the input rate, while under CONNET, they follow the reference values. CONNET’s performance improvement at a high non-premium rate is 100% for delay, and 54% for jitter.

WFQ is used to achieve a combination of good bandwidth allocation as well as delay/jitter protection. We compare the proposed control scheme with WFQ,⁹ showing that CONNET can still outperform WFQ in tracking low reference delay/jitter values. In this experiment, WFQ automatically assigns a new virtual queue to each flow with equal weights. Figure 14(c) shows a significant improvement for the controlled delay over WFQ (about 80%) using the same input in Figure 14(a). Although WFQ can achieve low jitter even with high loads, CONNET could achieve lower jitter with an almost 70% improvement as shown in Figure 14(d). The jump at time 200 sec is due to the controller transients. We also compare the delay performance with the case of uncontrolled delay in Figure 15(b) to show the degree of the absolute improvement by CONNET. The uncontrolled delay on our FIFO network could reach values up to 8 msec without WFQ or CONNET.

⁹Using WFQ implementation on Free-BSD ALTQ [3].

6.4. Assumptions, Limitations and Extensions

Beside showing the effectiveness of CONNET, we state the assumptions we made as well as its limitations.

The first assumption is that all non-premium traffic is controlled and enters the FIFO pipe via the front-end while exiting the rear-end. This reflects the actual case where access links are usually short enough (one to three hops) to have little cross traffic. Second, we assume that there is an adequate method for measuring run-time delay and jitter across the access link/pipe in order to feed the error back to the controller. We also assumed throughout the paper that the premium traffic cannot by itself exceed the access link capacity or available resources. In other words, we do not consider admission control on premium traffic.

One of the limitations of the current version of CONNET is not dealing with burst control along with rate control. We use only one control input, which is the non-premium traffic rate. In future, we would like to design a two-input controller that can control both the rate and the burst of non-premium traffic.

7. Related Work

While feedback control theory [7] has been used for decades in many engineering and scientific disciplines, such as mechanical and aeronautical engineering, only recently its effectiveness in solving network control problems has been realized (for example, see [9–11, 13]). The main goal of CONNET is different from others since we are looking at delay and jitter guarantees in networks with two or more services classes. More recent studies that use control-theoretic approaches to performance guarantees were reported in [1, 8, 15]. These studies focused on server-side control, such as web servers or Lotus Notes email servers. Although the domains of these studies are different from those of CONNET, network and traffic control, they are useful examples for us to illustrate many aspects in computer-based control design. The type of control closer to that used in this paper is Adaptive Bandwidth Control (ABC) [16] and the references thereof. CONNET differs from this type of work in that it does not apply control inside the node itself

(by changing the service rate), but uses an outside traffic controller, which is much easier to deploy without modifying the existing network routers. CONNET is also different from rate-based scheduling techniques such as that in [20], as we study single FIFO queues and do not employ any particular scheduling mechanism (other than FIFO).

8. Conclusions

We have presented a new simple, yet robust delay and jitter control mechanism, called CONNET, for premium traffic that shares bottleneck access links with non-premium traffic. CONNET is *reservation-less* and based on the relationship observed between the delay/jitter of premium traffic and the amount of non-premium traffic sharing the same access link. We took a control-theoretic approach to studying the delay and jitter control on FIFO-based access links, and presented the design and implementation of feedback control on a testbed network using the LQG regulator. CONNET creates automatic and *self-controlled* network services that require minimal operational effort. We evaluated the performance of CONNET under various experimental scenarios to show its correctness, accuracy and robustness. We also compared CONNET with other well-known rate-based disciplines such as CBQ and WFQ, revealing its significant improvement in delay and jitter performance over the other schemes. Since it is designed to handle single-FIFO-queue networks, CONNET provides a better solution for today's FIFO-based Internet without requiring any modification or any special scheduling mechanism in the network routers. In future, we would like to address the limitations of the current version of CONNET, and explore ways of integrating it into a more general traffic control framework that creates self-controlled network services across the Internet.

References

- [1] T. Abdelzaher, K. Shin, and N. Bhatti. Performance Guarantees for Web Server End-Systems: A Control-Theoretical Approach. *IEEE Trans. on Parallel and Distributed Systems*, 13(1):80–96, 2002.
- [2] W. Almesberger. Linux Network Traffic Control - Implementation Overview. In *Proceedings of the 5th Annual Linux Expo*, May 1999.
- [3] K. Cho. Managing Traffic with ALTQ. In *Proceedings of USENIX'99, Monterey, CA*, Jun. 1999.
- [4] A. Demers, S. Keshav, and S. Shenker. Analysis and simulation of a fair queueing algorithm. In *Proceedings of ACM SIGCOMM'89, Austin, Texas*, Sep. 1989.
- [5] M. El-Gendy, A. Bose, H. Wang, and K. Shin. Statistical Characterization for Per-Hop QoS. In *Proceedings of International Workshop on Quality of Service (IWQoS'03) Monterey, CA*, Jun. 2003.
- [6] S. Floyd and V. Jacobson. Link-Sharing and Resource Models for Packet Networks. *IEEE/ACM Transactions on Networking*, 3(4):365–386, Aug. 1995.
- [7] G. Franklin, J. Powell, and M. Workman. *Digital Control of Dynamic Systems*. Addison-Wesley, 1998.
- [8] N. Gandhi, J. Hellerstein, S. Parekh, D. Tilbury, and Y. Diao. MIMO Control of an Apache Web Server: Modeling and Controller Design. In *Proceedings of American Control Conference*, May 2002.
- [9] C. Hollot, V. Misra, D. Towsley, and W. Gong. On Designing Improved Controllers for AQM Routers Supporting TCP Flows. In *Proceedings of IEEE INFOCOM'01, Anchorage, Alaska*, Apr. 2001.
- [10] S. Keshav. A Control-Theoretic Approach to Flow Control. In *Proceedings of ACM SIGCOMM'91, Zurich, Switzerland*, Sep. 1991.
- [11] A. Kolarov and G. Ramamurthy. A control-theoretic approach to the design of an explicit rate controller for ABR service. *IEEE/ACM Transactions on Networking*, 7(5), Oct. 1999.
- [12] M. Kuznetsov et al. A Next-Generation Optical Regional Access Networks. *IEEE Communications Magazine*, 38:66–72, Jan. 2000.
- [13] H. Ngin and C. Tham. A Control-Theoretical Approach for Achieving Fair Bandwidth Allocations in Core-Stateless Networks. *Computer Networks, Elsevier Science*, 40(6), Dec. 2002.
- [14] P. Overschee and B. Moor. *Subspace Identification For Linear Systems: Theory, Implementation, Applications*. Kluwer Academic Publishers, 1996.
- [15] S. Parekh et al. Using Control Theory to Achieve Service Level Objectives in Performance Management. *Journal of Real-Time Systems, special issue on Control-Theoretical Approaches to Real-Time Computing*, 23(1/2), Jul./Sep. 2002.
- [16] P. Siripongwutikorn, S. Banerjee, and D. Tipper. A Survey of Adaptive Bandwidth Control Algorithms. *IEEE Communications Surveys & Tutorials*, 5(1):14–26, Third Quarter 2003.
- [17] N. Spring et al. Receiver Based Management of Low Bandwidth Access Links. In *Proceedings of IEEE INFOCOM'00, Tel-Aviv, Israel*, Mar. 2000.
- [18] VoIP Troubleshooter. <http://www.voiptroubleshooter.com/problems/access.html>.
- [19] H.-Y. Wei and Y.-D. Lin. A Survey and Measurement-Based Comparison of Bandwidth Management Techniques. *IEEE Communications Surveys & Tutorials*, 5(2):10–21, Fourth Quarter 2003.
- [20] H. Zhang and S. Keshav. Comparison of Rate-Based Service Disciplines. In *Proceedings of ACM SIGCOMM'91, Zurich, Switzerland*, Sep. 1991.