*First publ as paper in: IEEE Global Telecommunications Conference (Globecom 2003, San Francisco, CA, USA, 2003), Vol. 7, pp. 4180- 4185*

# Closed-Loop Congestion Control for Mixed Responsive and Non-Responsive Traffic

Roman Pletka[†], Andreas Kind[†], Marcel Waldvogel[†] and Soenke Mannal[‡]
[†]IBM Zurich Research Laboratory, CH-8803 Rüschlikon, Switzerland
{rap,ank,mwl}@zurich.ibm.com
[‡]ISR, Universität Stuttgart, D-70511 Stuttgart, Germany
soenke.mannal@isr.uni-stuttgart.de

***Abstract*— Today's known and widely used active queue management (AQM) schemes do not differentiate between packets from responsive (e.g., TCP sessions) and non-responsive traffic (e.g., UDP). This results in further widening the gap of unfair advantage already inherent to non-responsive traffic, as the responsive sender will significantly reduce its future transmit rate as a result of the congestion signals. As a simple work-around, responsive and non-responsive traffic are often assigned distinct AQM parameters. This approach however requires tuning for each traffic class that potentially depends on the current or expected offered load. In other words, responsiveness and TCP-friendliness cannot be estimated easily — not at last due to short-lived TCP sessions. In this paper we propose a closed-loop congestion control (CLCC) scheme on top of an existing AQM scheme to achieve fair bandwidth distribution among concurrent responsive and non-responsive traffic. The new scheme has the advantage that it does not need to estimate the level of responsiveness of traffic. We analyze our scheme on top of an existing rate-based AQM scheme known to approximate max-min fairness, and by means of simulations show that our extension significantly improves fair bandwidth allocation for responsive and non-responsive traffic. The simulation results have been verified with a prototype implementation on the IBM PowerNP 4GS3 network processor.**

## I. Introduction

AQM systems control packet-drop probabilities according to the level of congestion. Typically, packets are dropped with a higher probability when congestion increases in order to avoid undesired lock-out situations and unnecessarily high delays [1]. Past research efforts focused on configuring AQM systems such as RED [2] and numerous variants thereof [3]–[7] for service-level specifications. In particular, the difficulty in finding the appropriate parameters for RED for any combination of offered load was pointed out in [8], [9].

In general, AQM schemes are not able to achieve accurate bandwidth guarantees. Depending on the underlying hardware, a full-featured scheduler might not be available (e.g., only priority scheduler, WRR scheduler, or FIFO queue), and therefore absolute bandwidth guarantees are hard to achieve only through AQM. Even in the presence of a separate fair queuing (FQ) scheduler, the goals of AQM and FQ will partially disagree. Namely, AQM tries to keep queues short, while FQ needs longer, fixed-length queues to perform its task. Under congestion, FQ will fill its buffers, preventing them from accepting bursts and adding to the total queuing delay.

As a result, AQM is often beneficiary to FQ. Especially rate-based AQM [10], [11] seems to be more convenient for Quality-of-Service (QoS) and Service Level Agreement (SLA) environments, as it allows configuration based on packet rates rather than buffer thresholds. However, as will be shown later, even with rate-based AQM, some manual tuning is necessary to achieve fair bandwidth allocation for responsive (e.g., TCP sessions) and non-responsive (e.g., UDP) traffic at the same time. This is due to end-to-end congestion control mechanisms built into responsive protocols whereby packet drops are interpreted as network congestion somewhere on the transit path causing the sending rate to be reduced (back-off mechanism). This congestion response makes responsive protocols willfully give up bandwidth to avoid the dreaded congestion collapse. Non-responsive, greedy protocols on the other hand will grab the new bandwidth. Given sufficient bandwidth requested by non-responsive traffic, these will continuously reduce the bandwidth available to responsive, "nice" traffic, until the latter are left without any bandwidth.

In this paper we propose a new closed-loop congestion control (CLCC) scheme on top of an AQM scheme that enables fair bandwidth allocation even under heavily bursty traffic conditions with concurrent responsive and non-responsive networking traffic. In particular, we use Bandwidth Allocation Technology (BAT) [10] as the underlying AQM algorithm to show the benefit of our control scheme. We expect the new CLCC scheme to work just as easily with other types of AQM schemes, especially with those based on additive increase, multiplicative decrease (AIMD) control algorithms.

AIMD schemes require the configuration of at least two main parameters per traffic class, namely the increase and decrease constants. Thus in a setup with two traffic classes, i.e., a responsive and a non-responsive one, four parameters need to be tuned. Our new time-discrete strictly bilinear control scheme adapts and optimizes only one of these four BAT parameter that before used to be constant. Extensive simulations showed that the new CLCC scheme significantly outperforms the traditional BAT algorithm, and is effective when executed on a much slower time scale than drop probability updates with BAT are done. Therefore, the scheme needs not necessarily be implemented directly in hardware. However, an implementation in hardware should be considered if the amount of control traffic is an issue.

The paper is structured as follows. First, we briefly discuss related work, motivating the need for an improved AQM scheme. Then, in Section III we first introduce and evaluate the BAT algorithm in presence of responsive and non-responsive traffic. Section IV proposes our new scheme, CLCC, that improves fairness characteristics in this situation. Results obtained from various simulations are presented in Section V. We conclude in Section VI.

## II. Related Work

Dynamic-RED [12] uses control theory for adapting the packet-dropping probability to stabilize the queue length close to a given threshold value. Benefits are bounded delays and independence from the number of flows present in the system. As the algorithm is not rate-based, it cannot provide bandwidth guarantees.

BLUE [13] has been introduced to overcome the problems encountered by AQM schemes based on queue lengths. The algorithm uses packet-loss and link-idle events to manage congestion, and the proposed extension called *Stochastic Fair BLUE* is able to identify and rate-limit non-responsive flows by using multiple levels of independent hash functions. The probability that flows are misclassified and therefore needlessly discriminated, increases with the number of flows present in the system. A major drawback is the additional expense in implementation complexity.

The properties of congestion feedback mechanisms based on AIMD and their fairness properties were discussed extensively [14]–[17]. BAT by Bowen *et al.* [10] applies this background towards a rate-controlled AQM scheme. This proves effective at keeping the queue length to a minimum while absorbing short-term bursts. BAT approximates max-min fairness [18] by managing the drop rates using an open-loop control. Individual offered loads act as a feedback signal. Compared to other AQM algorithms, BAT also performs well for non-responsive networking traffic.

## III. Open-loop Control for Bandwidth Allocation

### A. The BAT Algorithm

The BAT active queue-management scheme [10], an open-loop control system, maintains per-flow transmit probabilities $T_i(t)$ and operates on two (coarse and fine grained) AIMD levels. For a flow $i$, the probability function is given as:

$$T_i(t+\Delta t) = \begin{cases} \min(1, T_i(t)+w) & \text{if} \quad f_i(t) \le f_{\text{i,min}} \\ T_i(t)(1-w) & \text{else if} \\ & f_i(t) > f_{\text{i,max}} \\ \min(1, T_i(t)+C_i \cdot \bar{B}(t)) & \text{else if} \;\; B(t)=1 \\ T_i(t)(1-D_i \cdot O_i(t)) & \text{otherwise} \end{cases} \tag{1}$$

The more aggressive part of the algorithm brings the actual flow rate into the maximum and minimum bounds $(f_{\text{i,min}}, f_{\text{i,max}})$ configured, while a fine-tuned control loop operates within these bounds. The aggressive part is governed by the constant $w$ $(0 < w < 1)$. The coefficients $C_i$ and $D_i$ determine the rate of increase and decrease during fine-grained control. Their choice depends on the desired convergence rate.

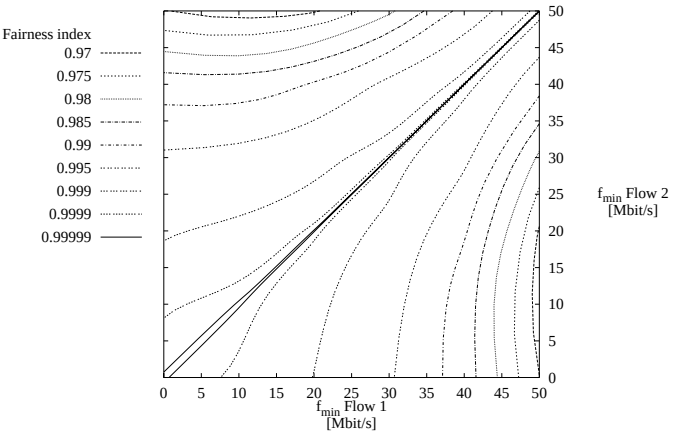



Fig. 1. Fairness index for two flows depending on their $f_{\text{i,min}}$ value using BAT. $f_{\text{i,max}}$ is set to 100 Mbit/s for both flows.

The two-level technique of BAT increases its convergence speed while retaining the stability properties of AIMD [14].

At initialization, the transmit probability for each flow $i$ is $T_i(0) = 1$, and $T_i$ are then computed periodically in intervals of $\Delta t$ from the preceding transmit probability value, the current offered rate of the flow $O_i$, the current serviced rate of the flow given as $f_i = T_i \cdot O_i$, and the exponentially weighted moving average $\bar{B}(t)$ of the excess bandwidth signal $B(t)$. $B(t)$ indicates whether transmit bandwidth is available or is expected to become available soon [10].

### B. Fairness of Non-Responsive Traffic

First we investigate the fairness of AQM in presence of non-responsive traffic only. To do so, we have chosen the BAT algorithm as our reference AQM. To illustrate the fairness of BAT, we analyze two competing CBR sources sending UDP traffic at 100 Mbit/s over a 100-Mbit/s link while varying their minimum bandwidth guarantee $f_{\text{i,min}}$ from 0 to 50 Mbit/s. In our implementation, BAT updates transmit fractions every 4 ms. The average bandwidth allocated to each flow over a simulation time of $t = 20$s is compared with the theoretical fair value of max-min fairness. As a metric for fairness, we use a weighted version of the fairness index as introduced in [19]:

$$f(x_1, x_2, ..., x_n) = \left(\sum_{i=1}^{n} \frac{x_i}{w_i}\right)^2 \Big/ n \sum_{i=1}^{n} \left(\frac{x_i}{w_i}\right)^2 . \tag{2}$$

The fairness index lies between 0 and 1, and is 1 if all users receive the theoretical throughput, given by the weights $w_i$. Thus, Equation 2 tells how close to a desired bandwidth allocation, in our case max-min fairness, the measurements are. The simulation results in Figure 1 show the weighted fairness index as a function of various minimum bandwidth guarantees. High fairness is achieved as long as the $f_{\text{i,min}}$ are close and decreases whenever they drift away from each other as well as for higher minimum bandwidth guarantees.

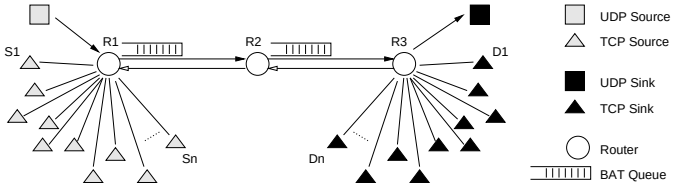



Fig. 2. Simulation topology for measuring combined UDP and TCP traffic using BAT.

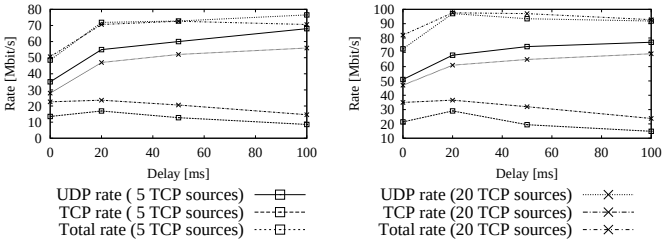



Fig. 3. Measured TCP throughput, allocated UDP bandwidth, and total throughput for different $f_{\min}$ values of BAT. $f_{\max}$ is set to 100 Mbit/s.

### C. Responsive vs. Non-Responsive Traffic

The results of [10] show that BAT performs well for non-responsive traffic and keeps the buffer occupancy level low during periods of heavy congestion. However, fair bandwidth allocation is typically not achieved in presence of *both* responsive *and* non-responsive traffic because the congestion control used by responsive protocols amplifies the control signal.

Figure 2 shows the network topology used for our simulations, which closely matches the hardware testbed with BAT implemented on an IBM PowerNP 4GS3 network processor. Traffic is divided into two flows: The first flow consists of non-responsive CBR traffic, and the second of responsive TCP connections. The CBR flow is generated using an IXIA traffic generator; TCP sources reside on a Linux computer with a 2.4.18 kernel using `ttcp` [20]. On the receiving side, the network emulator Dummynet [21] delays the packets before they are delivered to the application (`ttcp` receiver) to simulate WAN distances.

Figure 3 shows the UDP and TCP rates allocated by BAT for different $f_{\text{i,min}}$ as a function of the delay introduced for various numbers of TCP sources. The allocated TCP bandwidth increases with the number of TCP sources. This is consistent with findings from other AQM algorithms [12], [22] and can be modeled using the throughput equation of TCP [23]. Furthermore, the TCP bandwidth share decreases with increasing RTT, as it will take longer to recover from packet loss [23].

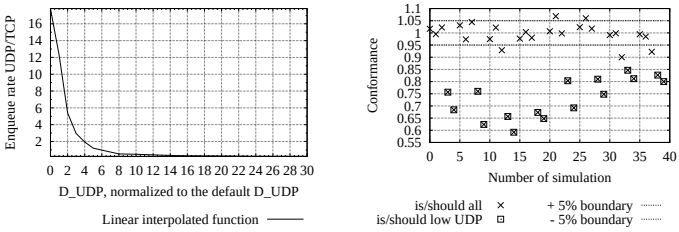
### D. A First Step to Improve Fairness

A first measure to increase fairness between responsive and non-responsive traffic would be to adapt the increase and decrease constants $C_i$ and $D_i$ depending on the traffic characteristics currently observed. Thus, for a responsive and a non-responsive traffic class, four parameters need to be adjusted. In early simulations we found that it is sufficient

TABLE I

SIMULATION PARAMETERS FOR THE OPEN LOOP CASE.

| Simulation | UDP rate [Mbit/s] | RTT | Packet size [bytes] |
|---|---|---|---|
| 0–4 | 100 | 2–20 | 1500 |
| 5–9 | 200 | 2–20 | 1500 |
| 10–14 | 100 | 2–20 | 500 |
| 15–19 | 200 | 2–20 | 500 |
| 20–24 | 100 | 2–198 | 1500 |
| 25–29 | 200 | 2–198 | 1500 |
| 30–34 | 100 | 2–198 | 500 |
| 35–39 | 200 | 2–198 | 500 |




(a) Correction factor. (b) Relative conformance.

Fig. 4. Correction factor for $D_{\text{UDP}}$ as a function of the desired bandwidth allocation given by maximum and minimum bandwidth guarantees obtained by a linear interpolation over a large set of different simulations described in Table I, and its relative conformance with the set of simulations.

to adapt the decrease constant of non-responsive traffic $D_{\text{UDP}}$ to achieve significantly better fairness results. Higher values for $D_{\text{UDP}}$ allow a faster reaction to non-responsive traffic in the case of congestion, compensating for the non-responsive characteristics of UDP traffic.

Using *ns* [24] and the simulation topology in Figure 2, we first investigate the fairness behavior of BAT. The link rate for all links is set to 100 Mbit/s. A high number of TCP sources fed by greedy FTP agents share a bottleneck link with non-responsive UDP traffic created by a CBR source. All TCP connections have different RTTs that are equally distributed within the interval given in Table I. Clearly, because of the congestion control mechanism of TCP, the two flows will not get the same bandwidth allocated. With identical increase and decrease constants $C_i$ and $D_i$, non-responsive traffic is able to grab an almost 10 times higher bandwidth share.

Figure 4(a) shows the linearly interpolated relation between the allocated bandwidth and $D_{\text{UDP}}$ gained from a large set of simulations. Detailed simulation parameters for the open-loop case are listed in Table I. Each simulation set consists of five different bandwidth allocations, namely 90, 60, 45, 30 and 25% UDP traffic, respectively. For the set of simulations given, conformance often exceeds 5% (Figure 4(b)), especially when the allocated UDP rate is relatively small (indicated with crossed boxes) compared with the offered TCP load. In summary, the method introduced above fails to provide reasonable long-term fairness.

## IV. IMPROVING FAIRNESS WITH CLCC

To improve fairness, we introduce a closed-loop control mechanism. The goal of the control process is to allocate bandwidth in a fair way as a long-term objective. When choosing

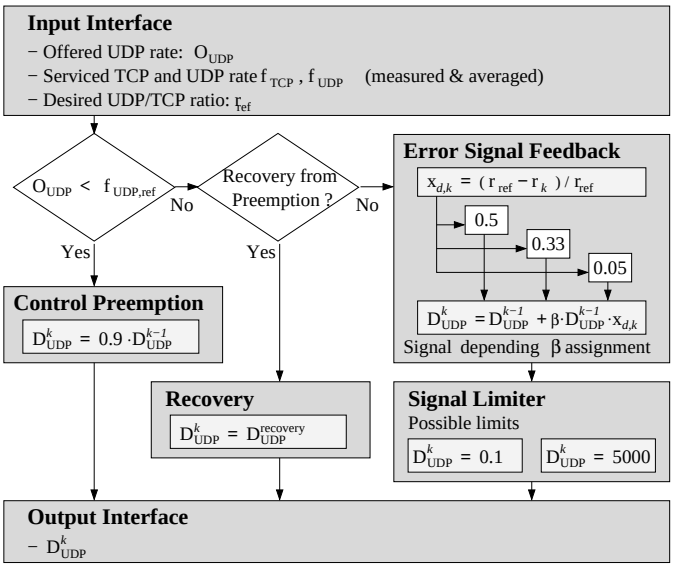



Fig. 5. Diagram of the AQM control algorithm using CLCC.

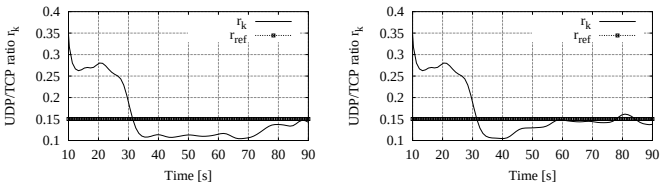



Fig. 6. UDP/TCP traffic ratio $r_k$ (a) without and (b) with limitation of the manipulated variable $D_{\text{UDP}}$.

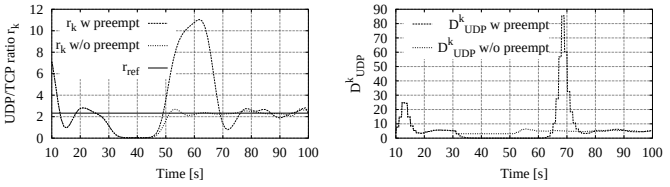



Fig. 7. Impact of control preemption on (a) the UDP/TCP traffic ratio $r_k$ and (b) the manipulated variable $D_{\text{UDP}}$.

the actual controller, we started out with the family of PID (Proportional-integral-derivative) controllers [25], which are characterized by relatively low computational requirements, suitability for hardware implementation, and their excellent theoretical performance.

An ideal PID controller is described as follows:

$$y(t) = K_R \left[ x_d(t) + \frac{1}{T_N} \int x_d(t)\, dt + T_V \frac{dx_d(t)}{dt} \right], \quad (3)$$

where the time-dependent variables are the control input $y(t)$, and the control error $x_d(t)$. $K_R$, $T_N$ and $T_V$ are system parameters. The terms in the square brackets represent the proportional, integral, and differential influence (from left to right). The discrete time notation of Equation 3 can be found by approximation of the derivatives using backward differences. Solving for the actual value of the reference input yields

$$y_k = y_{k-1} + b_0 \cdot x_{d,k} + b_1 \cdot x_{d,k-1} + b_2 \cdot x_{d,k-2} \ , \quad (4)$$

where $b_0$, $b_1$, and $b_2$ determine the basic properties of the controller. By setting $b_1 = b_2 = 0$, Equation 4 corresponds to a pure integral controller. This simplification leads to an additional reduction in both complexity and calculation effort while still ensuring efficient elimination of constant steady-state errors. We found that a proportional part ($b_1 \neq 0$) does not lead to a clear improvement of the results. Therefore we limit ourselves to integral feedback control.

The main idea of our CLCC is to adapt only the non-responsive decrease factor $D_{\text{UDP}}$, which in the traditional BAT model was a predetermined constant [10]. The now time-dependent value of $D_{\text{UDP}}$ is determined by a control process that adjusts its value autonomously according to the current traffic characteristics. This decision was made because responsive protocols react strongly to congestion indications; therefore, we need to adapt the behavior of non-responsive traffic forcibly by tuning $D_{\text{UDP}}$.

The controlled variable is the ratio of serviced UDP and TCP rates $r_k$. The reference variable is represented by $r_{\text{ref}}$ and corresponds to the desired ratio derived from max-min fairness [18]. The control error $x_{d,k}$ is defined as

$$x_{d,k} = \frac{r_{\text{ref}} - r_k}{r_{\text{ref}}} \ . \quad (5)$$

Now $D_{\text{UDP}}$ is a time-dependent value specifying the manipulated variable of the system given as

$$D^k_{\text{UDP}} = D^{k-1}_{\text{UDP}} + b_0 \cdot \frac{r_{\text{ref}} - r_k - 1}{r_{\text{ref}}} \ , \quad (6)$$

which describes the difference equation of a time-discrete, strictly bilinear system of first order.

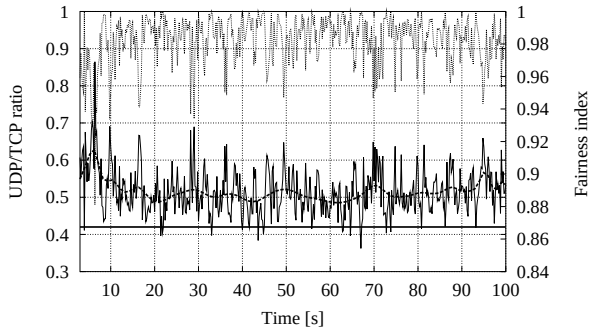
To adapt the control-error signal to the current state of $D_{\text{UDP}}$, we define $b_0 = \beta \cdot D^{k-1}_{\text{UDP}}; \quad \beta < 0$. The parameter $\beta$ depends on the absolute value of $x_{d,k}$. Three different cases for $\beta$ are distinguished that stress the importance of $x_{d,k}$ for large deviations. The detailed CLCC algorithm is shown in Figure 5. The ratio $r_k$ is measured every 100 ms and averaged over a time interval of 1 s to obtain a low-pass filter effect, hence assuring stability. The control algorithm is invoked every second, whereas the underlying AQM scheme calculates new transmit fractions every 4 ms.

The following subsections introduce further improvements integrated into the new algorithm.

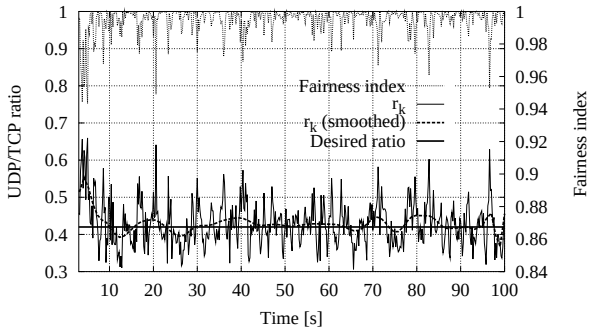
### A. Limitation of the Manipulated Variable

Equation 1 gives an implicit definition of the lower and upper limits of $D_{\text{UDP}}$. As $0 \leq T_i \leq 1$, the lower and upper bound can be written as

$$0 \leq D_{\text{UDP}} \leq \frac{1}{O_{\max}} = \frac{1}{\sum O_i} \ . \quad (7)$$

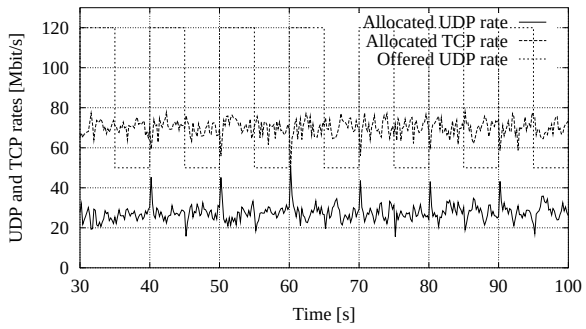



(a) BAT without additional controller.

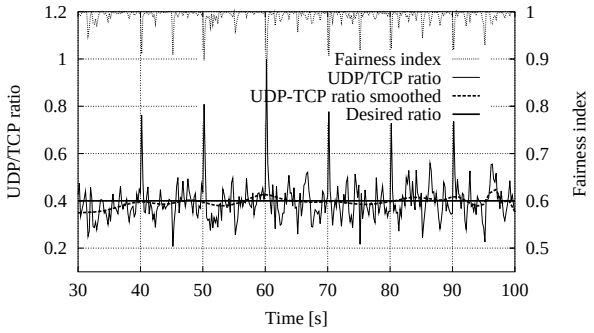



(b) Using the CLCC algorithm.

Fig. 8. Simulation results with a set of greedy TCP connections and one constant UDP flow.




(a) UDP and TCP rates.




(b) UDP/TCP ratio $r_k$ and fairness index.

Fig. 9. Simulation results with a set of 100 greedy TCP connections and one bouncing UDP flow.

The significance of the limitation of $D_{\text{UDP}}$ is illustrated by the following simulation based on the previously introduced topology: The desired ratio between UDP and TCP traffic $r_{\text{ref}}$ is set to 0.15. At the beginning, the offered UDP rate set to 200 Mbit/s. The TCP sources are the same as in the previous simulation. At $t = 30\ s$, the offered UDP rate is drastically reduced to 30 Mbit/s. The results are shown in Figure 6. The high congestion state combined with the low desired ratio forces the integral controller to increase $D_{\text{UDP}}$. If no limitation is set, the value of $D_{\text{UDP}}$ might be extremely high. After a while, when the offered load has been reduced significantly, $D_{\text{UDP}}$ is adapted to a smaller value to match the desired ratio. If the previous value is very high, a large reduction is needed, and the controller needs longer to adjust. Setting an appropriate upper limit reduces this time frame significantly.

### *B. Control Preemption*

In some special cases there is no use to adapt $D_{\text{UDP}}$, as any action would result in unnecessary integration loops that drive the manipulated variable $D_{\text{UDP}}$ to abnormal high values or zero. This is the case when (a) the offered load of UDP traffic during a certain time frame is low enough that even with a transmit fraction of one, the desired value cannot be reached, (b) the offered UDP load is below the assured minimum bandwidth, thereby the decrease loop of BAT will not be used independent of the state of the excess bandwidth signal. The system is then said to be in control-preemption state, where the previous value of $D_{\text{UDP}}$ is preserved.

On can argue that $D_{\text{UDP}}$ should be set to zero during control preemption because all packets should be forwarded during this period. As the control periods are rather long, this could be abused to push high amount of UDP traffic through the system because there is no way to reduce its transmit fraction during one control period. Such a behavior would lead to undesired oscillations. A feasible solution is to decrease $D_{\text{UDP}}$ by 10% each time step during control preemption. Figure 7 shows the impact of control preemption when the offered load is below the allocated rate between $25\ s \leq t \leq 45\ s$ and increases again after $t = 45\ s$.

To increase stability, we bound $D_{\text{UDP}}$ as shown in Figure 5. Moreover, the control loop is suspended when the offered UDP load falls below $f_{\text{UDP,ref}}$, because at this time a useful adaptation is no longer possible. The system is then said to be in control-preemption state, in which the preceding value of $D_{\text{UDP}}$ is preserved.

Next, we present results from simulations done in *ns* and verified with a prototype implementation on a IBM PowerNP 4GS3 network processor.

## V. Results

In Section III-D, we showed that $D_{\text{UDP}}$ should not be a constant value. As a consequence thereof, we introduced CLCC. Here we compare simulations of the BAT AQM

schemes with and without CLCC. We use identical simulation setups and take the default increase and decrease parameters for the open-loop case. The exact simulation setup is as follows: One UDP flow is sending at a constant bitrate of 90 Mbit/s and shares a 100-Mbit/s bottleneck link between router R1 and R2 with a set of 100 greedy TCP connections, each of them having a uniformly distributed RTT between 2 and 200 ms. The controller allocates link bandwidth in the ratio of $r_{\text{ref}} = 30/70 \approx 0.42$. The results are shown in Figure 8.

In Figure 8(a), the unacceptable difference between desired and actual ratio using traditional BAT for responsive traffic is evident. This behavior is also seen in the spread of the fairness index. The improvements obtained using our algorithm are shown in Figure 8(b). Although the UDP/TCP ratio still fluctuates, the long-term average closely matches the desired value, which is also reflected in improvements in the fairness index. Further simulations with Pareto-distributed on/off-sources for TCP traffic showed similar results, and are not presented here due to space limitations.

In a next step, we tested CLCC in presence of a UDP load alternating between 50 and 120 MBit/s every five seconds. $r_{\text{ref}}$ is set to 0.4. The results of this simulation are presented in Figure 9.

Clearly, the new control algorithm is also suitable in presence of bursty traffic. Fair bandwidth allocation is rapidly recovered even after large changes in the offered UDP rate.

## VI. Conclusion

In this paper, the problem of fair bandwidth allocation for mixed responsive and non-responsive traffic is addressed. We proposed a closed-loop congestion control scheme for an existing AQM system and showed that the fairness characteristics can be improved significantly. The time-discrete strictly bilinear control scheme is based on a proportional integral derivative controller and has been simulated in *ns* and implemented on an IBM PowerNP 4GS3 network processor for verification of the results presented.

We argue that AQM with intrinsic fairness properties such as BAT with CLCC provides advantages over combinations of other AQM schemes combined with an FQ scheduler. Specifically, the separate buffering required by FQ will partly negate the short-queue goal of AQM. To absorb bursts, FQ queue lengths need to be set to constant, high values. Under heavy load, this leads to full buffers introducing delays and preventing the queues from accepting bursts.

To our knowledge, this is the first AQM system that is able to automatically control under disparate reactiveness without having to also include an expensive fair queuing system.

## References


[1] B. Braden *et al.* Recommendations on queue management and congestion avoidance in the Internet. RFC 2309, Internet Engineering Task Force, April 1998.

[2] S. Floyd and V. Jacobson. Random early detection gateways for congestion avoidance. *ACM Trans. on Networking*, August 1993.

[3] D. Lin and R. Morris. Dynamics of random early detection. In *Proc. SIGCOMM '97*, pages 127–137, Cannes, France, September 1997.

[4] D. Clark and W. Fang. Explicit allocation of best effort packet delivery service. *ACM Trans. on Networking*, 6(4), August 1998.

[5] V. Rosolen, O. Bonaventure, and G. Leduc. A RED discard strategy for ATM networks and its performance evaluation with TCP/IP traffic. *ACM Computer Commun. Rev.*, 29(3), July 1999.

[6] W. Feng, D. D. Kandlur, D. Saha, and K. G. Shin. A self-configuring RED gateway. In *Proc. of INFOCOM '99*, pages 1320–1328, March 1999.

[7] Cisco Systems Online IOS Documentation. Weighted Random Early Detection (WRED), 1998.

[8] M. May, J. Bolot, C. Diot, and B. Lyles. Reasons not to deploy RED. In *Proc. of 7th Int. Workshop on Quality of Service (IWQoS '99), London*, pages 260–262, June 1999.

[9] V. Misra, W.-B. Gong, and D. F. Towsley. Fluid-based analysis of a network of AQM routers supporting TCP flows with an application to RED. In *Proc. SIGCOMM 2000*, pages 151–160, August 2000.

[10] E. Bowen, C. Jeffries, L. Kencl, A. Kind, and R. Pletka. Bandwidth allocation for non-responsive flows with active queue management. In *Proc. Int. Zurich Seminar on Broadband Commun., IZS 2002*, February 2002.

[11] Andreas Kind and Bernard Metzler. Rate-based active queue management with token buckets. In *Proceedings of 6th IEEE International Conference on High Speed Networks and Multimedia Communications (HSNMC'03)*, July 2003.

[12] J. Aweya, M. Ouellette, D. Y. Montuno, and A. Chapman. A control theoretic approach to active queue management. *Computer Networks*, 36:203–235, 2001.

[13] W. Feng, D. Kandlur, D. Saha, and Kang G. Shin. BLUE: A new class of active queue management algorithms. Technical Report CSE-TR-387-99, University of Michigan, April 1999.

[14] D. Chiu and R. Jain. Analysis of the increase and decrease algorithms for congestion avoidance in computer networks. *Computer Networks & ISDN Systems*, 17(1):1–14, June 1989.

[15] F. P. Kelly and A. Maulloo. Rate control in communication networks: Shadow prices, proportional fairness and stability. *Operational Res. Soc.*, (49):237–252, 1998.

[16] P. Hurley, J.-Y. Le Boudec, and P. Thiran. A note on the fairness of additive increase and multiplicative decrease. In *Proc. ITC-16, Edinburgh, UK*, pages 467–478, June 1999.

[17] S. Gorinsky and H. Vin. Additive increase appears inferior. Technical Report 2000-18, Dept. of Computer Sciences, Univ. of Texas, May 2000.

[18] D. Bertsekas and R. Gallager. *Data Networks*. Prentice-Hall, 1987.

[19] R. Jain. *The Art of Computer Systems Performance Analysis*. John Wiley, New York, 1991.

[20] M. Muuss. The story of the test TCP program ttcp. http://ftp.arl.army.mil/∼mike/ttcp.html.

[21] Luigi Rizzo. Dummynet: A simple approach to the evaluation of network protocols. *ACM Computer Commun. Rev.*, 27(1):31–41, 1997.

[22] R. Morris. TCP behavior with many flows. In *Proceedings of IEEE International Conference on Network Protocols (ICNP)*, pages 205–211, October 1997.

[23] J. Padhye, V. Firoiu, D. Towsley, and J. Kurose. Modeling TCP throughput: A simple model and its empirical validation. In *Proc. SIGCOMM*, pages 303–314, 1998.

[24] S. Bajaj *et al.* Improving simulation for network research. Technical Report 99-702b, Univ. Southern California, March 1999.

[25] G. F. Franklin, J. D. Powel, and M. Workman. *Digital Control of Dynamic Systems*. Addison-Wesley, third edition, 1997.