Handling High Bandwidth Aggregates By Receiver Driven Feedback Control (Extended Abstract)

Chee-Wei Tan
Princeton University

Dah-Ming Chiu John C. S. Lui
The Chinese University of Hong Kong

David K. Y. Yau
Purdue University

I. Introduction

Network congestion control solutions fall into one of two categories: *end-to-end* without involving intermediate routers, and *hop-by-hop* to shorten the control loop but at the expense of more costly router design and operation. The Internet protocol suite requires intelligence mainly at the network end points. TCP, for example, performs end-to-end control, where *senders* adjust their allowed windows of outstanding packets according to acks of previous data sent and receive windows advertised by receivers. Since the absence of a timely ack can be the result of either a congestion event or a corrupted packet, such *window-based* congestion control may be unduly affected by the goal of error control [2]. To decouple congestion control from error control, a *rate-based* design (e.g., ATM ABR) typically feeds back a binary congestion control to multiple senders, who then adjust their sending rates accordingly.

In this work, we consider a variant congestion control problem in which a *receiver* computes an explicit allowed traffic rate and uses it to regulate multiple *routers* sending to the receiver. Our design is thus rate-based, but uses an explicit rate signal instead of a binary signal. Such congestion control has many applications in practice, such as a web server regulating its Internet clients in a flash crowd scenario, or a server trying to protect itself from aggressive sources during a distributed denial-of-service (DDoS) attack. Intelligence is mostly required at the receiver since all the essential control decisions are made there. A subset of the routers participate in the control but do not make control decisions. As sources may not be trusted (e.g., in a DDoS attack), their participation is neither required nor assumed.

In our approach, a network receiver, say S, experiencing resource overload installs a *router throttle* [6] at a set of upstream routers that are several hops away. The throttle specifies the maximum rate (in bits/s) at which packets destined for S can be forwarded by each router. Traffic that exceeds the rate limit will be dropped by the router. An installed throttle will generally change the load experienced at S. S can then adjust the throttle rate in multiple rounds of feedback control until it achieves its load target. Our goal is to derive a throttle algorithm that is (i) highly *adaptive* by avoiding unnecessary control parameters that would require configuration, (ii) able to *converge* quickly to a fair resource allocation, (iii) highly *robust* against extrinsic factors beyond the system's control (e.g., dynamic input traffic, and number/locations of current sources), and (iv) *stable* under given delay bounds.

II. Solution Overview

We assume that a receiver, say S, aims to export its full service capacity U_S (in kb/s) to the network, but no more. An important research question is then how to determine proper throttle rates such that the objective is achieved under heavy loading. It is clear that the throttle rate should depend on the current demand distributions and therefore must be negotiated dynamically between the receiver and the throttle routers. As stated earlier, the negotiation approach is *receiver* driven. A receiver operating below the designed load limit needs no protection, and need not install any router throttle. As receiver load, denoted by ρ, increases and crosses the designed load limit U_S, the receiver may start to protect itself by installing a rate throttle, denoted as r_S, at the throttle routers. If a current throttle rate fails to bring down the load at S to below U_S, then the throttle rate can be further reduced. On the other hand, if the receiver load falls below a low-water mark L_S (where $L_S < U_S$), then the throttle rate is increased to allow more packets to be forwarded to S. If an increase does not cause the load to significantly increase over some observation period, then the throttle is removed. The goal of the control algorithm is to keep the receiver load within $[L_S, U_S]$ whenever a throttle is in effect.

In fairly allocating the receiver capacity among the deployment routers, we define the following notion of fairness:

Definition *1* (distributed max-min fairness) *A resource control algorithm achieves* **distributed max-min fairness** *among the deployment routers, if the allowed forwarding rate of traffic for S at each router is the router's max-min fair share of some rate r satisfying $L_S \leq r \leq U_S$.*

III. Control-theoretic Fair Throttle Algorithms

We have studied several fair throttle algorithms with increasing sophistication. Each algorithm is a probing algorithm that tries to determine a *common* throttle rate, r_S, to be sent by the receiver to all the deployment routers using negative feedback [1]. They are all designed to converge to the desired fairness criterion. The difference between the algorithms is in the way the receiver computes r_S, which will affect the speed of convergence and the stability of the system.

A. Binary Search Fair Algorithm

The possible range for the throttle rate r_S is $[0, U_S]$, the two extremes corresponding to an infinite number of throttles and one throttle, respectively. A familiar binary search algorithm [3] can be applied: Reduce the search range by half at every iteration of the algorithm. This algorithm uses the aggregate rate ρ to determine the direction of adjustment. When the aggregate rate is more than U_S, r_S is reduced by half. When the aggregate rate is less than L_S, r_S is increased to the mid-point between the current r_S and U_S.

Binary search is highly efficient if we assume that the traffic loading is static. When traffic can vary, however, the algorithm must detect situations in which the shrunken search range for r_S can no longer deal with the changed traffic conditions, and *re-initialize* the search range accordingly, which is challenging.

B. Proportional control with number of throttle estimate

Binary search directly adjusts r_S, without using the magnitude of the overshoot (or undershoot) of the aggregate rate ρ. We now present an algorithm that makes use of the information, as well as an *estimate* of the number of deployment routers that actually drop traffic. For clarity of exposition, we will call a deployment router whose offered traffic rate exceeds the throttle rate (and hence will drop traffic because of throttling) an *effective throttling router*. We also call the throttle at such a router an *effective throttle*. If the number of effective throttles is n, then it is reasonable to set the change of r_S to

$$\Delta r_S = \frac{|\rho - C|}{n} \quad (1)$$

where C represents the target rate (a value within $[L_S, U_S]$) of aggregate traffic at S. By keeping track of the last throttle rate r_{last} and the last aggregate rate ρ_{last}, we can estimate n as $n = |\rho - \rho_{last}|/|r_S - r_{last}|$. Given n, we can compute a suitable change to r_S using Eqn. 1.

We wish also to address our fourth objective, namely system *stability* despite delay in the feedback control. The objective can be achieved by trying to correct only a *fraction* K_P of the discrepancy between the aggregate rate and the rate target. The fraction K_P should be *proportional* to the discrepancy; hence, we also refer to such control as *proportional control* [4].

We found that for stationary traffic demand, the proportional controller's rate of convergence is monotonic and relatively fast [5]. For dynamic traffic, the speed of convergence depends on the setting of K_P. How the setting of K_P can affect system stability motivates the design of our next algorithm.

C. Proportional-derivative control fair algorithms

We now introduce an optimization to the proportional controller. When there is a change in the offered load at the deployment routers, a higher proportional gain can adapt faster to the change. However, such "strong" control may result in large fluctuations in the throttle rate. On the other hand, a "weak" control signal can cause prolonged congestion at the receiver. Hence, the goal is to reduce fluctuations while still achieving fast convergence. The stability problem that arises when a high proportional gain is used can be mitigated by considering a derivative control parameter, K_D. as shown in Fig. 1. The new algorithm is called a *proportional and derivative* (PD) controller [4].

In the algorithm, if $\rho > U_S$ or $\rho < L_S$, the total expected change in throttled traffic rate in all of the deployment routers at the next control interval is

$$\triangle\rho'_{i+1} = -K_P(\rho_i - C) - K_D\triangle\rho_i; \qquad \triangle\rho'_1 = 0, \quad (2)$$

where $\triangle\rho_i$ is the actual change and $\triangle\rho'_i$ is the expected change in the control interval i.

In Eqn. 2, the first term is necessary for the mismatch regulation and the second term tracks the rate of change in the feedback. The first term has the largest impact on the aggressiveness of the probe used in the algorithm. A smaller K_P will make the system slower to converge. But a larger K_P may result in larger overshoots, sometimes even causing the system to become unstable – i.e., the system oscillates indefinitely around the band

Proportional-Derivative Fair Throttle Algorithm

$r_S \doteq (U_S + L_S)/N$; /* Init $N = |$throttle routers$|$ */
$\rho_{last} := -\infty$
While(1)
  sends current rate-r_S throttle to deployment routers;
  monitor traffi c arrival rate ρ for time window w;
  If ($\rho \geq U_S$)
    $C := L_S$
  elif ($\rho \leq L_S$)
    If ($\rho - \rho_{last} < \epsilon$)
      remove rate throttle from routers;
      break;
    else
      $C := U_S$;
    fi;
  else
    break;
  fi;
  $\phi := -K_P \times (\rho - C) - K_D \times (\rho - \rho_{last})$
  $r_S := r_S + \phi/$est(ρ, r_S);
  $\rho_{last} := \rho$;
end while;

Fig. 1. Proportional-Derivative fair throttle algorithm specifi cation.

$[L_S, U_S]$ without convergence. The second term gives a signal proportional to the rate at which the mismatch changes. It is more responsive to changes in the mismatch than the first term. It also has a damping effect on the amount of rate throttle fluctuation. A larger K_D implies more damping; it helps the stability of the system, and allows the proportional gain to be increased. Our simulation results [5] show that the term can reduce the amplitude of overshoot significantly during transient response while maintaining fast convergence.

The PD controller has the best performance among the controllers presented [5]. It is robust against a wide range of settings – not only does it guarantee convergence, but also it has much smaller over/under-shoot for the aggregate rates to the receiver S. Distributed max-min fairness is also guaranteed for all the deployment routers.

IV. Conclusion

A control-theoretic analysis of the presented algorithms can be found in [5]. In addition, extensive simulation results illustrate algorithm performance under different system parameters, including the number of traffic sources, their traffic patterns, variant delays of the feedback loops, and the choice of K_P and K_D [5]. Our results show that the PD algorithm has robust performance under diverse operating conditions.

References

[1] D. M. Chiu and R. Jain, "Analysis of the increase/decrease algorithms for congestion avoidance in computer networks", *Journal of Computer Networks and ISDN*, 17(1), pp. 1-14, June 1989.

[2] D. Clark, M. Lambert, and L. Zhang "NETBLT: A high throughput transport protocol", *Proc. ACM SIGCOMM*, 1987.

[3] T. H. Cormen, C. E. Leiserson and R. L. Rivest, "Introduction to Algorithms", *Second Edition*, MIT Press, 2001.

[4] B. C. Kuo, "Automatic Control Systems", Prentice Hall, 1975.

[5] C. W. Tan, D. M. Chiu, J. C. S. Lui, and D. K. Y. Yau, "A Distributed Throttling Approach for Handling High Bandwidth Aggregates". Technical Report CSE-04-110, Chinese University of Hong Kong, November 2004.

[6] D. K. Y. Yau, J. C. S. Lui, F. Liang and Y. Yeung, "Defending against distributed denial-of-service attacks with max-min fair server-centric router throttles", *IEEE/ACM Trans. Networking*, 13(1), Feb 2005.