

Organization-level Control of Excessive Internet Downloads

Saad Y. Sait, Hema A. Murthy, Krishna M. Sivalingam

Dept. of Computer Science & Engg., Indian Institute of Technology, Madras, Chennai 600036, INDIA

Email: aboomaryam.saad@gmail.com, { hema, skrishnam }@iitm.ac.in

Abstract—The control of excessive downloads by rogue users in organizational LANs is the subject of this work. Two mechanisms have been used in order to accomplish this. The first mechanism, is TCP rate control (TCR); it is a receiver-based flow control technique that can be used to effectively rate limit rogue users' flows, making more bandwidth available to regular users. The second mechanism, admission control reduces the bandwidth wastage due to users disconnecting out of impatience when user goodputs are low. Using simulation-based experiments, it has been demonstrated that the composite technique, *exclusive TCP rate and admission control* (xTRAC) provides seamless control of rogue users, while improving response times and goodput by upto 58% during overload. In this way regular users are *incentivized* and rogue users are *penalized* leading to long-term control of users.

I. INTRODUCTION

Peak-hour congestion is a universal problem faced by all types of Internet subscribers, be it residential, small or home office (SOHO), campuses or large enterprises. In the absence of pricing policies, organizational access networks are prone to excessive usage by rogue users. So, suitable control mechanisms need to be applied to penalize rogue users, and incentivize responsible usage, leading to efficient use of Internet access links. This work devises a scheme that accomplishes this objective while improving overall performance.

In order to control peak-hour congestion at the organizational level, first the effects of congestion should be mitigated for regular users. In Congestion Manager [1], applications *voluntarily* adjust to congestion, while QPC [3] places packets of rogue users on a low priority queue, and schedules them on a *strict* priority basis, thereby making more bandwidth available to regular users. In order to obtain *long term control* over users, however, responsible usage should be incentivized. In [5] users are given good service quality of limited duration during peak hours, whereas in [3], quotas are used up faster during peak hours, and low priority is given to rogue users.

Standard approaches to control downloads at the receiver are by dropping packets [6], marking packets [1] and by changing the advertised window like in *TCP rate control* (TCR) [4]. While dropping packets causes delays, marking packets assumes that senders react to ECN. On the other hand, all TCP versions respond to a change in advertised window. TCR divides available bandwidth in a controlled manner by advertised window adjustment, and avoids packet retransmissions and timeouts that lead to poor performance during peak-hours.

In this work, the problem of controlling rogue downloads within an organization is performed using TCR. Although TCR reduces congestion, it may lead to poor link utilization during heavy loads, when users disconnect out of impatience; this may be resolved using admission control. The resulting scheme called *exclusive TCP rate and admission control* (xTRAC) controls rogue users seamlessly based on traffic load, while improving regular user experience and overall link utilization. All control mechanisms have been implemented at a web proxy, assumed to be on the organizational LAN and performance gains have been demonstrated by simulation-based experiments. Alternatively, as both TCR and admission control can be applied at any intermediate router, the scheme may be extended to any router through which organizational traffic flows.

In this work, excessive and wasteful usage has been referred to as *abusive* usage, rogue users as *high* users and regular users as *low* users. In our previous work [7], techniques for classifying high and low users were devised; here the category of the users is assumed to be known.

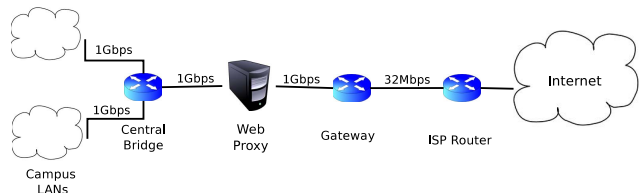


Fig. 1: Organizational access network topology

II. BACKGROUND

Fig. 1 shows the general topology of an organization's access network. All upstream traffic passes through an enterprise gateway to an ISP router, and then to the Internet, and vice versa for downstream traffic. Since all Internet traffic passes through the access network, it serves as a vantage point where control policies can be applied. Typically in organizations providing Internet access to their employees, all traffic passes through a single gateway to the ISP, and then to the Internet. This gateway is frequently a *web proxy* which forwards HTTP requests. In this work, the existence of a web proxy where the control mechanism may be deployed has been assumed.

TCR [4] is a receiver-based flow control technique that may be used in this scenario in order to control downloads. Two

important aspects of TCR are *window sizing* and *ACK spacing*. While window sizing refers to the modification of advertised window in the ACKs for flow control, ACK spacing is the spacing out of ACKs. As ACKs clock the sending of data at the sender, ACK spacing reduces the burstiness of traffic.

TCR categorizes flows as *hungry* or *non-hungry*. Hungry flows are those that are currently using all the bandwidth allocated to them. Non-hungry flows are backlogged at some intermediate router in the Internet and are therefore not in need of all the bandwidth allocated to them. TCR frequently re-allocates the bandwidth unutilized by the non-hungry flows among the hungry flows in a *rate allocation* step. This allocated rate is later used to compute the advertised window, which is set in the ACKs. This is the *rate enforcement* step. The reader is directed to [4] for further details regarding TCR algorithm.

III. CONTROL MECHANISM

The control mechanism uses TCR to control high user downloads. The general TCR approach [4] does rate allocation every time a new flow joins, at the end of an update interval and when a flow stops sending. Rate allocation complexity varies between $O(1)$ for a clever implementation and $O(n)$ for a straightforward implementation. In this work, TCR is being used to control HTTP traffic which has a high turnover rate; a $O(n)$ rate allocation performed every time a new flow joins or a flow stops sending would increase overheads. So, in this work, a straightforward implementation is used and rate allocation is performed only at the end of a sampling interval; even-though complexity is $O(n)$, it becomes $O(1)$, when the cost is amortized over the sampling interval. This implementation called *simple TCR* (STCR) will be described next.

The rate allocated to each flow by the rate allocation step is called the *allocated* rate. The observed rate of downloads for each flow is calculated at the end of a sampling interval. This is called the *calculated* rate. Fig. 2 shows the rate allocation algorithm. For each flow, the allocated rate A_i , calculated rate R_i , previous allocated rate A'_i , advertised window in packets w_i and its previous value w'_i are maintained. In the loop shown in lines (2) to (10), each flow is marked as hungry or not hungry; the number of hungry flows N_h and the total bandwidth consumed by all flows C are computed. Next, in line (11) unutilized bandwidth U is computed as $B - C$ where B is the total link capacity of the bottleneck link. In lines (12) to (17), this unutilized bandwidth U is then divided among the hungry flows.

In Fig. 2, flows are marked as hungry or non-hungry using a function *isHungry()* in line (3) by comparing R_i with A_i and A'_i (the reason for comparing with A'_i will be explained later). While allocated rates of hungry flows increase, the allocated rates of non-hungry flows are reduced to $(A_i + R_i)/2$ (line (8)). This has the effect of gradually reducing the rate in successive sampling intervals to R_i . The sampling interval is set to 300ms. This value should be large enough to judge based on packets received if the flow is hungry or non-hungry; but

```

1:  $C \leftarrow 0$ 
2: for each flow  $i$  do
3:   if newFlow[ $i$ ] or isHungry( $A_i, R_i, A'_i$ ) then
4:      $hungry[i] \leftarrow true$ ;  $C \leftarrow C + A_i$ ;
5:      $N_h \leftarrow N_h + 1$ 
6:   else
7:      $hungry[i] \leftarrow false$ ;  $C \leftarrow C + R_i$ ;
8:      $A'_i \leftarrow A_i$ ;  $A_i \leftarrow \frac{A_i + R_i}{2}$ ;
9:   end if
10: end for
11:  $U \leftarrow B - C$ 
12: for each flow  $i$  do
13:   if  $hungry[i] = true$  then
14:      $A'_i \leftarrow A_i$ ;  $A_i \leftarrow A_i + \frac{U}{N_h}$ ;
15:   end if
16:    $w'_i \leftarrow w_i$ 
17: end for
18:  $start \leftarrow t$ 

```

Fig. 2: Rate Allocation

it should not be too large to slow down the rate of increase of allocated rate for continuing flows.

Fig. 3 describes the rate enforcement step; it is performed every time the advertised window field in the ACK is set. w_i is set as shown in line (1). However, as flow control should also be performed, it is capped with W_{TCP} , the advertised window computed by TCP receiver based on available buffer space (line (7)). The value W_i , the advertised window, is then written to the ACK. It may be noted that for a flow i , the new allocated rate (and advertised window) takes effect only after one RTT_i into the sampling interval. So, the ACK spacing δ_i is computed using A'_i during the first RTT of the sampling interval, after which A_i is used (lines (2) to (6)). For the same reason, *isHungry()* (Fig. 2) computes the nett allocated rate from A_i , A'_i and compares it with R_i to determine if the flow is bottlenecked; it considers allocated rate A'_i in the first RTT and A_i in subsequent RTTs. For a new flow that has not entered upon rate allocation, w_i is set to the initial window of TCP.

```

1:  $w_i \leftarrow A_i \times \frac{RTT_i}{MSS_i}$ 
2: if  $t - start \leq RTT_i$  then
3:    $\delta_i \leftarrow \frac{RTT_i}{w'_i}$ 
4: else
5:    $\delta_i \leftarrow \frac{RTT_i}{w_i}$ 
6: end if
7:  $W_i \leftarrow \min(w_i \times MSS_i, W_{TCP})$ 

```

Fig. 3: Rate Enforcement

Even-though STCR avoids congestion at the bottleneck link by dividing the available bandwidth among the flows, it is still susceptible to poor user goodput when there are too many flows sharing the link. This leads to a wastage in link usage as users disconnect out of impatience. Typically, in a situation like this, admission control may be used to improve user

experience as performed in [2].

Fig. 4 illustrates how the proposed control mechanism works. A suitable fraction of the bandwidth is reserved for low users which will give them a good Internet experience. The remainder of the bandwidth is used by the high users. This is accomplished by implementing STCR at the TCP layer separately for both categories of users. For each category, a limit is fixed on the maximum number of external TCP connections as indicated in Fig. 4. When this limit is reached, *initiating* (new) requests of high and low users are placed in separate queues at the application layer. On the other hand, a *subsequent* request on a keep-alive connection is forwarded for better performance without being held up. As each category of users has its own *exclusive* share of bandwidth, and limit on number of flows, the composite scheme is called *exclusive TCP rate and admission control* (xTRAC).

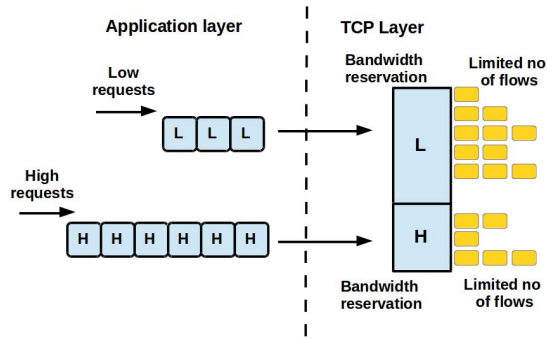


Fig. 4: xTRAC scheme

In the case of xTRAC, the total used bandwidth deviates from the total available bandwidth during the course of the sampling interval; this happens because flows are continuously terminating and new flows are joining, and maximum limit on number of flows may be exceeded when subsequent requests are admitted. When the aggregate bandwidth allocated to all the flows exceeds the capacity of the link, the link is *overbooked*. A surge in the number of requests leading to overbooking of the link is called a *request surge*. Sometimes flows are allocated a high bandwidth which they are unable to use. This is called *internal wastage* of bandwidth; it leads to ineffective utilization of link, which translates to larger overall response times.

IV. PERFORMANCE EVALUATION MODELING

Experiments have been performed by way of simulations. Fig. 5 shows the topology used in simulations; many clients in a LAN (campus computers) connect to servers through a bottleneck link with bandwidth 32Mbps. All other links in the network are allotted a much higher bandwidth, so that they are not bottlenecked. This is a typical dumbbell topology in which numerous clients connect by way of a bottleneck link to many remote servers and is suitable to analyze performance at the bottleneck link. RTTs vary between 50-200ms. The number of hosts in the LAN is varied in order to change the traffic

load on the proxy server. Listed below are other aspects of the modeling:

- *Inter-session times* - the times between successive accesses for the same user have been modelled as a mixture of 3 exponentials with weights 0.37, 0.47, 1.6 and corresponding means 1.8, 12 and 100 secs; based on IITM proxy logs.
- *HTTP response sizes* - have been modelled as a lognormal distribution with $\mu = 7.76$, $\sigma = 1.63$; based on IITM proxy logs.
- *Small and large flows* - small flows are typically web page views and have been modelled as 4 request/response non-overlapping interchanges, with an inter-request gap of 400ms; each response has less than 50MSS size. Large flows are those which involve a download (video, archive, executable etc.) and have been modeled as a single request/response interchange with response size greater than 50MSS.
- *User impatience* - for small flows, the time that a user waits before terminating a small flow has been modeled as $Q = 3.5 + Q_e$ where 3.5s represents the minimum wait time and $Q_e \sim \text{Exp}(1/3)$ is the extra wait time. For large flows, the minimum expected throughput that the user expects, (failing which he terminates the download) has been modeled as $P = 24 + P_e$, where 24kbps is the minimum expected throughput and $P_e \sim \text{Exp}(1/24)$. The user checks the progress of the download after a suitable time assuming the download happens at 160kbps.
- *Performance metrics* - *useful downloaded bytes*, *packet drop ratio* (at bottleneck link) and *average response times* (average time taken for download) are *system metrics*. They measure the aggregate performance of the system. Instead of measuring goodput, the total useful bytes downloaded have been reported. This excludes the bytes wasted due to terminated connections. *Number of completed connections* is a *control metric*; it measures the performance of low flows viz-a-vis high flows.

The time duration for each experiment was 500s. For each experiment the statistics were averaged over 10 runs with different seeds; statistics were found to have a standard deviation within 5% of the mean, indicating accuracy of the results.

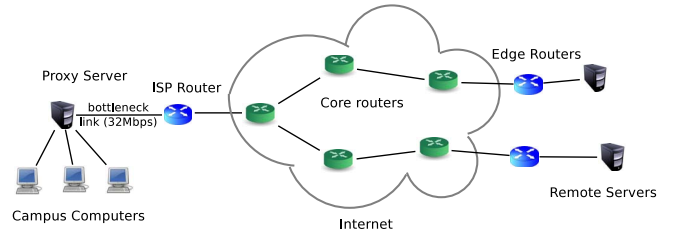


Fig. 5: Simulated LAN Topology

V. PERFORMANCE RESULTS

The xTRAC scheme controls abusive usage by applying a rate restriction on the aggregate of high and low flows

separately. This scheme is equivalent to the case when each category's flows take a distinct gateway with its own access speed to the Internet. It would be beneficial then to analyze the system performance for a single category of users under different user loads. These are presented below followed by results for xTRAC multiple category case.

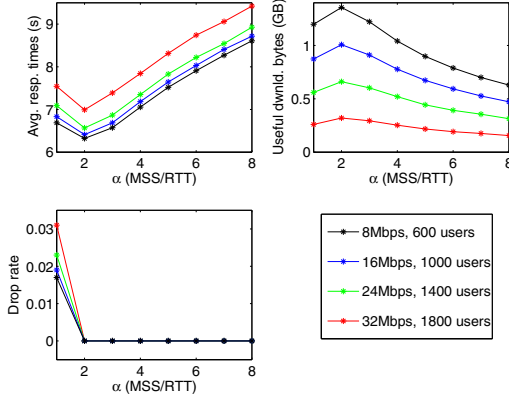


Fig. 6: Comparison of performance under heavy load for different bottleneck link capacities as α is varied

A. xTRAC Single Category

For discussion, say link capacity is B MSS/RTT, the maximum active flows is M and actual number of flows is m , $\alpha = B/M$ and $\beta = B/m$. For light loads M may be set to B . The maximum limit M is rarely reached, so wastage due to user impatience is minimal; internal wastage is always existent for light loads and cannot be reduced further.

For heavy loads $m = M + \delta_2$ and $\beta = \alpha - \delta_1$ are possible for $\delta_1, \delta_2 > 0$ due to request surge i.e. for light loads all slots are not occupied; when the user load is increased, all slots get occupied, and the limit may also be exceeded. Fig. 6 compares the performance under heavy traffic loads for different bottleneck link capacities. A peak is seen for $\alpha = 2$. This is a result of 2 trends. Firstly, when $\alpha = 1$, for heavy traffic $\beta < 1$ and $m > M$; so, user impatience and overbooking result in wasted bytes and packet drops respectively. Secondly, as α increases, internal wastage results in bad performance. Best performance is obtained for $\alpha = 2$.

B. xTRAC Multiple Category

A fraction of link capacity needs to be reserved for low users which would result in good experience for them. This means that the low user traffic would essentially be a light load for the (large) quota allocated to it, and the high user traffic would be a heavy load for the (smaller) quota allocated to it. It was noted in the previous section that for heavy loads, $\alpha = 2$ is suitable and for light loads, $\alpha = 1$ is appropriate.

An average response time of 3 seconds was chosen as the criterion for good performance for low users. The reservations corresponding to this average response were obtained, and the performance recorded and compared with plain TCP (TCP

TABLE I: xTRAC performance when compared to plain TCP across different user loads

User load	xTRAC Resrv.	Resp. time (s)		Useful bytes (GB)		xTRAC lo/hi
		No Ctrl.	xTRAC	No Ctrl.	xTRAC	
800	0.5	2.45	2.72	1.71	1.62	0.93
1000	0.52	3.21	3.21	1.80	1.82	1.04
1200	0.65	4.22	4.07	1.63	1.71	1.72
1400	0.8	5.17	4.05	1.44	1.74	3.28
1600	0.93	6.03	3.61	1.28	1.80	12.14
1800	1.0	6.86	3.03	1.15	1.82	∞

Reno) in Table I. Aggregate performance of xTRAC is better than plain TCP except under light traffic conditions. For xTRAC, useful bytes downloaded represent 85-90% of link utilization, as the maximum limit is 2GB (32Mbps for 500s); A 58% increase is observed over plain TCP for the highest load of 1800 users. Low users flows are served several times faster than high users depending on the user load (for plain TCP, both are served at the same rate). While low users experience good performance, high users' flows wait a short while after which the requests are served fairly fast. Essentially most high users' flows terminate without wastage in downloaded bytes, or completion without wastage.

VI. CONCLUSIONS

A scheme called xTRAC was devised for controlling abusive (high) users' downloads at an organizational web proxy, that also improves the performance of the system. The scheme uses TCR and admission control. Firstly, TCR mitigates the adverse effects of congestion for regular (low) users during peak hours. Secondly, TCR restricts the aggregate bandwidth of high users seamlessly depending on user load. Admission control serves to improve link utilization by marginalizing the loss due to user impatience. The scheme therefore incentivizes low users and penalizes high users during congestion. This aids in long term control of high users. xTRAC may be implemented at any intermediate router as both TCR and admission control may be applied at any intermediate router. It is also easily deployable as only a single host implementation needs to be changed.

REFERENCES

- [1] H. Balakrishnan, H. S. Rahul, and S. Seshan, "An Integrated Congestion Management Architecture for Internet Hosts," *ACM SIGCOMM Computer Communication Review*, vol. 29, no. 4, pp. 175–187, Oct. 1999.
- [2] A. Banchs, S. Tartarelli, and A. Descamps, "Intra-customer Admission Control for TCP Flows in Diffserv Assured Forwarding," in *Proceedings of IEEE GLOBECOM*, Nov. 2002, pp. 2568–2572.
- [3] S.-I. Chu and S.-C. Chang, "Time-of-day Internet-access Management by Combining Empirical Data-based Pricing with Quota-based Priority Control," *IET Communications*, vol. 1, no. 4, pp. 587–596, Aug. 2007.
- [4] S. Karandikar, S. Kalyanaraman, P. Bagal, and B. Packer, "TCP Rate Control," *ACM Computer Communications Review*, vol. 30, pp. 45–58, Jan. 2000.
- [5] D. Lee, J. Mo, J. Walrand, and J. Park, "A token pricing scheme for Internet services," in *Economics of Converged, Internet-Based Networks*, Oct. 2011, pp. 26–37.
- [6] C.-P. Li and E. Modiano, "Receiver-based flow control for networks in overload," in *Proceedings of IEEE INFOCOM*, 2013, pp. 2949–2957.
- [7] S. Y. Sait, A. Bhandari, S. Khare, C. James, and H. A. Murthy, "Multi-level anomaly detection: Relevance of big data analytics in networks," *Sadhana*, vol. 40, no. 6, pp. 1737–1767, 2015.