Organization-level Control of Excessive Internet Downloads

Saad Y. Sait
Computer Science & Engineering
Indian Institute of Technology, Madras
Chennai 600036
Email: aboomaryam.saad@gmail.com

Hema A. Murthy
Computer Science & Engineering
Indian Institute of Technology, Madras
Chennai 600036
Email: hema@iitm.ac.in

Krishna M. Sivalingam
Computer Science & Engineering
Indian Institute of Technology, Madras
Chennai 600036
Email: skrishnam@iitm.ac.in

Abstract—**Peak-hour congestion is a common problem faced by Internet subscribers across the world. Heavy traffic leads to poor Internet experience for all users. The objective of this work is to provide a solution to peak-hour congestion due to download traffic in flat-rate organizational LANs. 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 per-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 regular users, while improving response times and goodput by upto 58% during overload. In the simulations, xTRAC is implemented at a web proxy. A further advantage is that implementation changes need be made only to the web proxy, leading to an easy deployment.**

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. ISPs are concerned about this issue since efficient utilization is necessary for good revenue; good revenues lower the cost. It is also a concern for subscribers who are unable to have a good Internet experience during peak hours. It is particularly problematic for organizations where a large number of users share the same link unrestrictedly without controls in place. This work addresses this problem. A control mechanism is provided that seamlessly restricts the bandwidth of rogue users, while at the same time improves the goodput at the bottleneck link.

Organizations typically provide their users free Internet access; such unrestricted access leads to misuse of resources. The problem is further exacerbated in developing nations where effective bandwidth is an expensive resource and should be shared fairly among the users [5]. ISPs provide their subscribers Internet access, but control it using pricing policies. In the absence of pricing policies, suitable control mechanisms need to be applied in organizations to penalize rogue users, and incentivize responsible usage, which can lead to efficient use of Internet access links. Approaches to control Internet usage within an organization may be *voluntary* like in [2], *quota-based* like in [12] [15] [4] or *virtual pricing* schemes like in [10].

The current work focuses on controlling excessive download traffic in an organizational network. This cannot be done with traditional schemes like per-flow queueing (PFQ), class-based queueing (CBQ) or random early drop (RED) [18] since this would have to be performed at the ISP gateway connecting to the organization. This is not feasible because intra-organizational control is outside the purview of the ISP. Control of sender's rate is usually performed using *receiver-based flow control*. For well behaved sources, receiver-based flow control may be accomplished by modifying the advertised window field in ACKs. *TCP Rate Control* (TCR) [6] is one such technique to rate control TCP flows by manipulating the advertised window. Similar techniques of controlling TCP flows by adjusting the advertised window have been used in different domains [7] [8] [16] [1].

In this work, a TCR based algorithm called *simple TCR* (STCR) has been devised for the specific case of rate limiting HTTP traffic at the web proxy, a key feature of this algorithm being the simplicity of implementation. Then two schemes - *relative TCP rate control* (RTCR) and *exclusive TCP rate and admission control* (xTRAC) have been devised for control of excessive usage during peak hours in organizational LANs. While RTCR works by controlling the relative throughput of rogue viz-a-vis normal users' flows, xTRAC allocates mutually exclusive aggregate bandwidth to both classes and augments the algorithm with admission control. Using simulation-based experiments, the performance of both techniques have been compared with vanilla TCP Reno, both in terms of control of rogue users as well as overall system performance. xTRAC controls rogue users seamlessly based on traffic load, while improving goodput by upto 58%. 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. It may be noted that the scheme may be extended to any router through which organizational traffic flows. An additional benefit is the easy deployment - only the web proxy or router implementation needs to be changed.

The approach to control peak-hour congestion in this work is to incentivize responsible usage with good performance and penalize excessive and wasteful usage with limited performance, while improving aggregate performance of the system. 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 [17], techniques for classifying high and low users were devised; here the category of the users is assumed to be known.

II. Background and Related Work

Fig. 1 shows the general topology of an organization's access network. All upstream traffic passes through a 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. In order to control downloads, it is essential for a receiver-based flow control technique like TCR to be applied; admission control of flows is essential for good performance. This section gives an overview of these techniques in the context of controlling abusive Internet access. Since this work demonstrates the control mechanism at the web proxy, an overview of web proxies is also included in this section. Finally, prior work related to this area has been discussed.

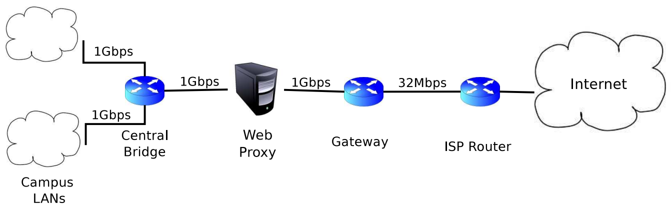
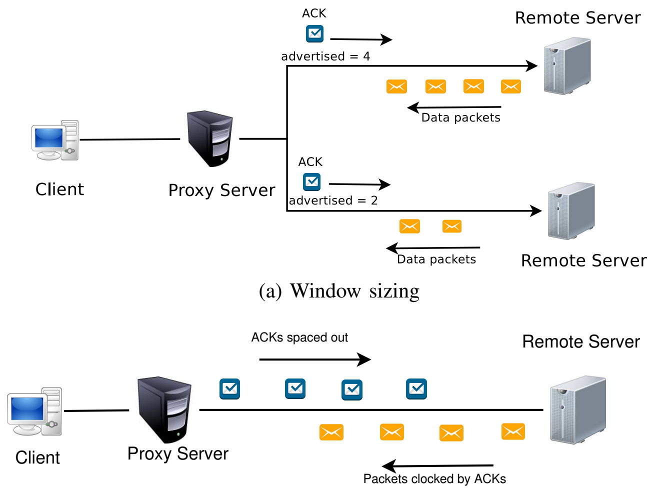


Fig. 1: Organizational access network topology



(a) Window sizing

(b) ACK spacing

Fig. 2: TCP Rate Control illustrated

A. TCP Rate Control (TCR) Approach

When a large number of flows transmit packets over a bottleneck link, packet retransmissions and timeouts result in poor per-user goodput. In order to avoid this, it has been recommended to rate control the flows [13]. For download traffic, rate control may be achieved by receiver based flow control. The TCR approach [6] is an example of this mechanism that can be used to control the download traffic on a congested link. It controls the flows by manipulating the advertised window. By dividing available bandwidth in a controlled manner by advertised window adjustment, packet retransmissions and timeouts leading to poor performance during peak-hours can be avoided.

The 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. They are used in the control mechanism at the web proxy as shown in Fig. 2.

TCR gives new flows an equal share of the bandwidth, and frequently reallocates bandwidth unutilized by backlogged flows to other flows which need it. In this way, TCR approach achieves max-min fairness with the aid of TCP closed loop feedback. It may be applied at any intermediate point between the source and the destination. A detailed description of an implementation of TCR will be described in Sec. III.

B. Admission Control

TCR approach reduces retransmissions and timeouts by controlling the total bandwidth allocated to all flows; in this way, it avoids bad per-user goodputs. Yet, bad per-user goodputs are still possible during peak hours when flows overcrowd the bottleneck link. Internet users terminate or lose interest in delayed responses; the result is that the *useful bytes* downloaded is drastically reduced. Prior work [3] [9] [14] used admission control in order to improve performance for the case of a large number of flows sharing a bottleneck link. In this work too admission control is used for effective link utilization and to boost system performance.

C. Web Proxies

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 *forward* proxy; it receives requests from clients on the LAN, and forwards them to the remote server; responses sent back from the remote server to the proxy are then returned to the client. The forward proxy may filter/modify requests and cache/modify responses.

A *web proxy* forwards HTTP requests. Fig. 3 shows the sequence of steps when a user accesses a web page through a web proxy. First, a TCP connection is established with the web proxy, and then the HTTP request is sent on the connection. The web proxy parses the HTTP request, obtains the domain name (URL) on which the request is being performed. Then, it performs a DNS lookup in order to obtain the IP address of the (remote) web server. It then establishes a TCP connection with the web server, and forwards the HTTP request to it. The

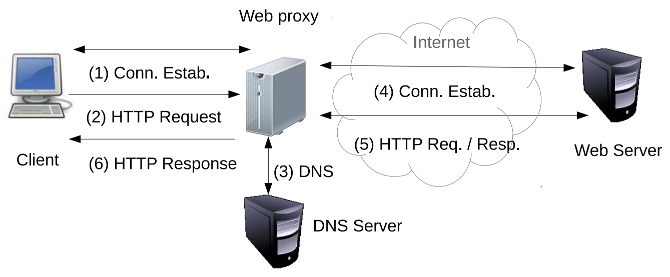


Fig. 3: Web access though proxy

web server in turn sends the HTTP response back to the proxy server. The proxy server then forwards the response back to the client. Once a connection is established, the connection may be *kept alive*, and may exchange any number of HTTP request/response pairs. The first HTTP request of a web access is called an *initiating* request. Requests that follow will be referred as *subsequent* requests.

In this work both TCR approach and admission control have been applied at the web proxy to control downloads.

D. Related Work

ISPs have attempted to control peak-hour congestion using pricing policies. In organizations, pricing is non-existent and such networks are essentially *flat-rate* networks. In order to control abusive usage, suitable control mechanisms need to be applied in organizations to penalize high users, and incentivize responsible usage, which will lead to efficient use of Internet access links. Congestion Manager [2] is an example of an end-to-end congestion management system. It is a *voluntary* approach in which applications adapt to congestion in the network. Other approaches control subscribers by applying control policies at the bottleneck link. In *quota-based* systems, a quota is maintained for each user, and when the quota is exceeded, the flows of the user are controlled at the bottleneck link. In [12] [15] such flows are placed in a low priority queue and forwarded on a *strict priority* basis. This is called *quota-based priority control* (QPC); this however does not control peak-hour congestion. A *time-of-day* pricing scheme may be applied in order to control peak-hour congestion. In this scheme, quotas are used up at different rates during different hours of the day [4]. Other schemes *incentivize responsible usage* by giving users limited but good service quality during peak-hours [10].

Organizational download traffic may be controlled by using traditional schemes like PFQ, CBQ, RED [18] or QPC [12] in the downstream direction at the ISP gateway connected to the organization. However, this is not feasible - firstly because ISPs are responsible for enforcing SLAs rather than intra-organizational control, which is outside the purview of the ISPs. Secondly, ISPs are not always able to distinguish between users because of network address translation (NAT).

Download traffic is normally controlled by *receiver-based* mechanisms. In Congestion Manager [2], the receiver controls download traffic by echoing standard congestion indicators like explicit congestion notification (ECN) and packet drops back to the sender. The sender in turn adjusts the sending rate. This is an end-to-end control mechanism which involves changes to both source and destination. A mechanism for controlling misbehaved sources which do not react to congestion control measures is found in [11]; they call their mechanism *back-pressure routing*; the difference in queue lengths for packets of the same class (source, destination pair) on adjacent routers close to the destination is used to decide the traffic of that class that should flow on the link. Excessive packets are dropped; the technique may be used for TCP as well as UDP traffic. For well behaved sources, *receiver-based flow control* may be accomplished by modifying advertised window field in ACKs. TCR [6] is one such scheme to control TCP flows by manipulating the advertised window. Firstly poor per-user goodputs during congestion due to packet retransmissions and timeouts are avoided; secondly, unfairness between flows is avoided. Similar techniques for controlling TCP flows by adjusting the advertised window have been used in different domains. In the WLANs domain, unfairness due to different data transmission rates of stations, and also due to the imbalance between upstream and downstream traffic may be resolved by manipulating the advertised window when the station is the receiver [7]. The advertised window has also been used in [8] [16] to rate limit elastic TCP traffic for better performance of real-time applications. Bluecoat's PacketShaper [1] is a commercial software based on TCR that provides different policies which may be set by the network administrator to rate limit different traffic classes e.g. rate limit greedy traffic to protect latency sensitive applications or rate limit top *talkers* and *listeners* on the network. In this work, two variants of the general TCR approach, called *relative TCP rate control* (RTCR) and *exclusive TCP rate and admission control* (xTRAC) have been used to rate control flows belonging to high users.

Control of high users should be accomplished while preserving the aggregate performance of the bottleneck link. As mentioned before, when too many flows share a link, it leads to low per-user goodput; users terminate the connection out of impatience resulting in a reduction of total useful bytes transferred at the bottleneck link. Prior work has used admission control in order to improve performance for the case of a large number of flows sharing a bottleneck link. Admission control for TCP flows may be performed at any intermediate router, and a flow may be admitted if there is room for it [14]. It has also been performed to reduce the peak-hour TCP traffic in a campus network [9]. Admission control for intra-customer *DiffServ* flows on ingress into an ISP network using the Assured Forwarding (AF) per-hop behavior (PHB) has been studied in [3]. They argue based on simulation results that admission control is essential in such a scenario. In this work as well, admission control is used to overcome the deterioration in goodput arising out of user impatience.

III. Control Mechanism

Control of peak-hour congestion at a bottleneck link connecting an ISP to an enterprise subscriber is the subject of this work. In this work, this is performed by controlling high users' flows, thereby making bandwidth available to low users' flows. The aggregate performance should be preserved while controlling the former category's flows. In this section, a simple implementation of TCR, called *simple TCR* (STCR) is described for controlling HTTP traffic at the web proxy. Then, two schemes based on STCR to control abusive usage - RTCR and xTRAC have been described.

A. Simple TCP Rate Control (STCR)

Simple TCR (STCR) is a simple implementation of TCR for controlling HTTP traffic at the web proxy, which is the receiver for the download traffic. STCR performs two important functions called *rate allocation* and *rate enforcement*. In the rate allocation step, STCR divides the bottleneck bandwidth among the flows; in the rate enforcement step, this rate is implemented by setting the advertised window field in the ACKs, and spacing out the ACKs.

STCR performs rate allocation at the end of each sampling interval; Fig. 4 shows the rate allocation algorithm. For each flow, the allocated rate A_i, observed rate R_i, previous allocated rate A'_i, advertised window in packets w_i, its previous value w'_i and weight c_i are maintained. In the loop shown in lines (2) to (10), each flow is marked as hungry or not hungry based on whether the flows are in need of more bandwidth or not; the sum of weights of hungry flows N_h is counted and the total bandwidth consumed by all flows C is 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 a weighted manner. The default value of c_i is 1.

In Fig. 4, flows are marked as hungry or non-hungry using a function *isHungry()* in lines (2) to (10). 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.

TCR rate allocation complexity varies between O(1) and O(n) depending on the algorithm chosen [6]. The general TCR approach does rate allocation every time a new flow joins, at the end of the sampling interval and when a flow stops sending. As HTTP traffic has a high turnover rate, STCR rate allocation is done only at the end of the sampling interval in order to avoid overheads. In the algorithm shown above the complexity is O(n), but it is only O(1) when the cost is amortized over the entire sampling interval.

Fig. 5 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

```
1: $C \leftarrow 0$
2: **for** each flow i **do**
3:     **if** $newFlow[i]$ **or** $isHungry(t, A_i, R_i, A'_i)$ **then**
4:         $hungry[i] \leftarrow true$; $C \leftarrow C + A_i$;
5:         $N_h \leftarrow N_h + c_i$
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:         $A'_i \leftarrow A_i$; $A_i \leftarrow A_i + U \times \frac{c_i}{N_h}$;
15:     **end if**
16:     $w'_i \leftarrow w_i$
17: **end for**
18: $start \leftarrow t$
```

Fig. 4: Rate Allocation

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. The sending rate should then be matched with the ACK spacing at the receiver since the rate of the flow is controlled by the ACK spacing. This 'synchronization' is performed in lines (2) to (6). The function *isHungry()* (Fig. 4) compares the allocated rate with the calculated rate in order to determine if the flow is bottlenecked; it considers allocated rate A'_i for the first RTT and A_i for subsequent RTTs.

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

Fig. 5: Rate Enforcement

During the course of the sampling interval, STCR allocates an initial window w_0 packets to new flows in their first RTT; terminated flows release bandwidth; the total used bandwidth is not explicitly restricted below the link capacity B, and this leads to bursty traffic if new flows use more bandwidth than what is released by terminated flows. 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. Schemes based on STCR will be discussed next.

B. Relative TCP Rate Control (RTCR)

Rate limiting of high users' flows may be performed by tweaking the c_i value in Fig. 4. Each low user's flow may be associated with a weight c_l, and each high user's flow may be associated with a weight c_h, $c_l > c_h$. Unutilized bandwidth is then divided among the flows proportionate to their weights. We refer to the ratio c_l/c_h as the *relative preference*. This scheme is called *relative TCP rate control* (RTCR). Although RTCR helps to control high users' flows, large values of relative preference exacerbate internal wastage of link capacity.

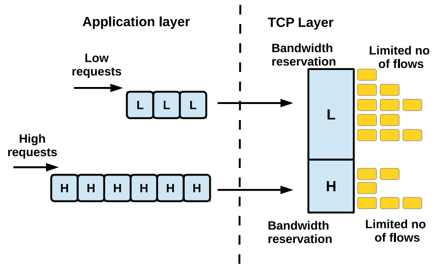


Fig. 6: xTRAC scheme

C. Exclusive TCP Rate and Admission Control (xTRAC)

As mentioned before in Sec. II-B, admission control may be performed in order to increase the per-user goodput of currently active flows, and thereby increase the useful bytes transmitted on the link. So, rate control should be augmented with admission control. It is illustrated in Fig. 6. A dedicated fraction of the bandwidth is allocated to the low users, and likewise for the high users. For each category, a limit is fixed on the maximum number of external TCP connections. When this limit is reached, new (initiating) requests of high and low users are placed in separate queues at the application layer. On the other hand, if a subsequent request arrives, it is is forwarded for better performance without being held up. Even-though admission control mitigates overbooking of the link to a large extent, overbooking is still possible in the case of xTRAC because subsequent requests may arrive on an inactive connection, when the maximum limit on the number of external connections has been reached.

IV. Performance Evaluation Modeling

Experiments have been performed by way of simulations. Fig. 7 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 dumbell 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.

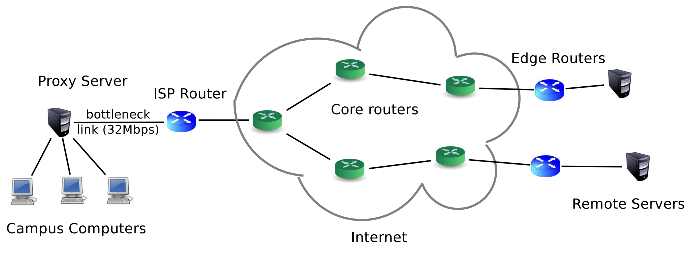


Fig. 7: Simulated LAN Topology

A. Traffic Generation

The *inter-session* time is the time between successive web accesses for the same user. The inter-session times and HTTP response sizes were collected from IITM proxy server logs and distributions matching them were chosen. The distribution of inter-session times for users has been modelled as a mixture of 3 exponentials with weights 0.37, 0.47, 1.6 and corresponding means 1.8, 12 and 100(s). The distribution of response sizes has been modelled as a lognormal distribution with $\mu = 7.76$, $\sigma = 1.63$.

Flows may be *small* or *large* flows. A small flow is one which is involved in a web page view, and consists of a few request/response interchanges with a remote server through a proxy; these interchanges are non-overlapping i.e. the next request is sent only after the current response has been received, and after a certain delay. Therefore small flows have been modelled as 4 request/response interchanges, with an inter-request gap of 400ms. Each of the 4 request/response interchanges goes over the same TCP connection simulating the case of a connection with keep-alive header enabled. Each response has a size less than 50MSS.

A large flow, on the other hand is one which involves a download (video, archive, executable etc.). A large flow has been modelled as a single request/response interchange with response size greater than 50MSS.

Equal volumes of high and low user traffic were generated.

B. Performance Metrics

Both *control* metrics as well as *system* metrics have been chosen as a measure of performance. While control metrics measure the performance of low flows viz-a-vis high flows, system metrics measure the aggregate performance of the system. Control metrics used are *number of completed connections* for small flows and *throughput* for large flows. *Goodput*, *packet drop ratio* (at bottleneck link) and *average response times* (average time taken for download) are aggregate metrics. Instead of measuring goodput, the total useful bytes downloaded for each algorithm have been reported. This figure excludes the bytes wasted due to terminated connections.

C. User impatience

User impatience has also been modelled. It has been assumed that a user terminates a web access (a small flow) after waiting for a suitable time, modelled as a random variable

$$Q = q_{min} + Q_e$$

where q_{min} is a constant representing the minimum wait time and $Q_e \sim Exp(1/\mu_Q)$ is the extra wait time. In our experiments q_{min} has been set to 3.5s, and μ_Q, the average extra wait time has been set to 3s.

On the other hand, for large downloads it has been assumed that the user checks the status of the download at a time when he/she expects the download to complete; this is the time taken for the download to complete at a rate of p_{sat}, a satisfactorily high rate. The user accurately estimates and checks at this time; if he/she is not satisfied with the throughput of the flow, he/she disconnects. Satisfactory throughput is modelled as

$$P = p_{min} + P_e$$

where p_{min} is a constant representing the minimum satisfactory throughput, $P_e \sim Exp(1/\mu_P)$ is the extra expected throughput. In our experiments p_{sat} has been set to 160 kbps, p_{min} has been set to 24kbps, and μ_P, the average extra satisfactory throughput has been set to 24kbps.

D. Simulation Setup

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.

V. Performance Results

In the following sections, the performance of the different schemes is analyzed.

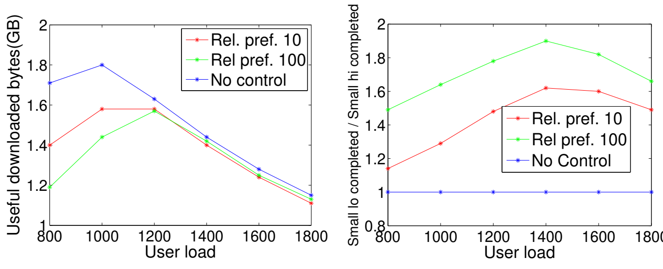


Fig. 8: RTCR performance for a 32 Mbps bottleneck link across different user loads for relative preference values 10 and 100

A. Relative TCP Rate Control (RTCR)

RTCR is applied with relative preference set to 10 and 100 for different user loads. The results in this case are shown in Fig. 8. Results indicate limited control over high users; useful bytes downloaded is also affected. Severe rate restriction of high users leads to user impatience at heavier loads, leading to reduction in useful bytes. On the other hand, for lighter loads, internal wastage of bandwidth allotted to low users leads to reduction in useful bytes. Useful bytes is between 55-80% of link capacity (maximum is 2GB - 32 Mbps for 500s) for all loads. Based on this, it may be inferred that restriction with relative preference leads to inefficient link utilization and limited control over high users.

B. Exclusive TCP rate and admission control (xTRAC)

In the previous section it was noted that the relative preference offered limited control, and at the price of efficient link usage. The xTRAC scheme offers an alternative way of controlling 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. The results for xTRAC single and multiple category case are presented below.

1) xTRAC Single Category: The following scheme is presented to enable analysis of this mechanism. The average RTT between the hosts and the servers, obtained from our simulations is 136ms; in that case a bottleneck link of 32Mbps would transfer $32 \times 10^6 \times 0.136/(8 \times MSS)$ packets in a single RTT. If MSS is assumed to be 1440, this value would be approximately 378 packets. If a maximum of M active flows are allowed, then the minimum average allocated rate for each flow would be $378/M$ MSS/RTT. This *minimum per-flow bandwidth* then is a significant parameter that influences the performance of the system. For ease of analysis, minimum per-flow bandwidth may be represented as α, effective per-flow bandwidth as β, the bottleneck link capacity as B MSS/RTT, the maximum active flows as M and actual number of flows as m. Then we have the relation

$$\alpha = \frac{B}{M} \leq \beta = \frac{B}{m} \quad (1)$$

It should be noted from (1) that for light loads $\beta \geq \alpha$, $m \leq M$ whereas for heavy loads $\beta \approx \alpha$ and $m \approx M$ i.e. for light loads all slots are not occupied; when the user load is increased, all slots get occupied frequently so that $\beta \rightarrow \alpha$, $m \rightarrow M$.

Fig. 9 compares the performance for light traffic loads for different bottleneck link capacities. The performance is similar in all cases. As α is increased, there is more scope for internal wastage, and thereby larger response times. Larger response times translate to a reduction in useful bytes because the rate at which users are serviced reduces. In each case shown, optimal performance is obtained when $\alpha = 1$ for light loads.

Fig. 10 compares the performance under heavy traffic loads for different bottleneck link capacities. A peak is seen for $\alpha = 2$. This is a result of 3 trends. First, when $\alpha = 1$, for heavy traffic it follows that $\beta \rightarrow 1$ and $m \rightarrow M$; so, user impatience with reduction in useful bytes results. This does not happen in the case of light loads (Fig. 9), as the effective per-flow bandwidth $\beta > \alpha$ and $m < M$. Secondly, note the large drop rate for $\alpha = 1$. This is because of request surge resulting in overbooking of the link, and resulting bad performance.

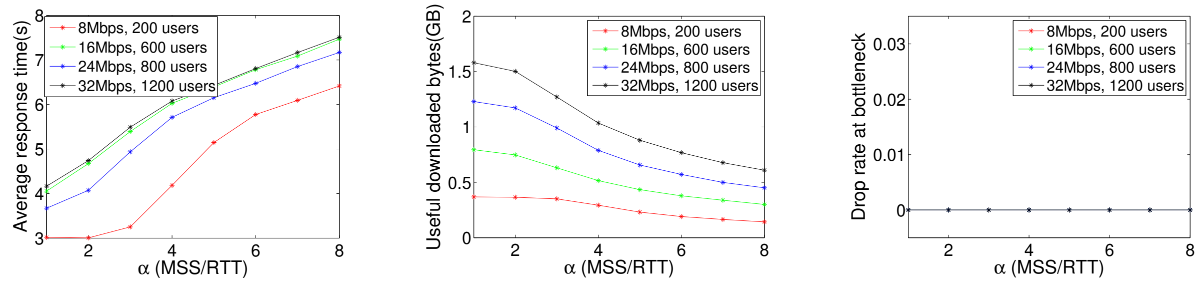


Fig. 9: Comparison of performance under light load for different bottleneck link capacities as α is varied

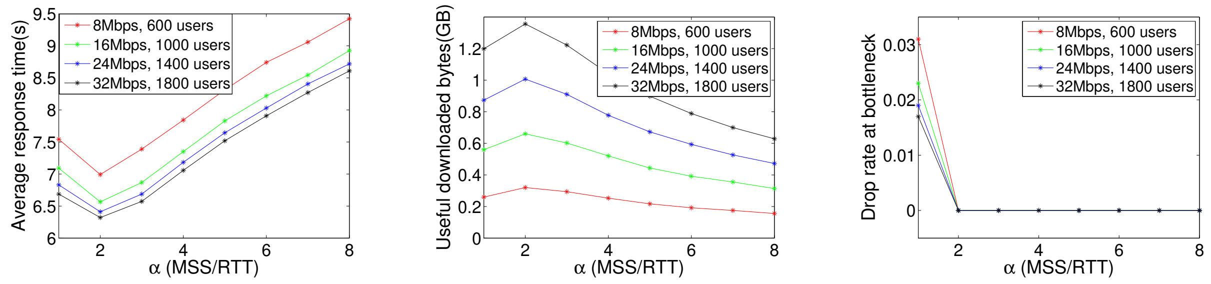


Fig. 10: Comparison of performance under heavy load for different bottleneck link capacities as α is varied

Thirdly, as α increases, internal wastage and bad performance results. Note that in the case of heavy load, consistently good performance is obtained for $\alpha = 2$ case.

Next, the results for the multiple category case are analyzed.

2) 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 more suitable. From (1), this means that $M = B/2$ and $M = B$ are suitable for high and low users respectively.

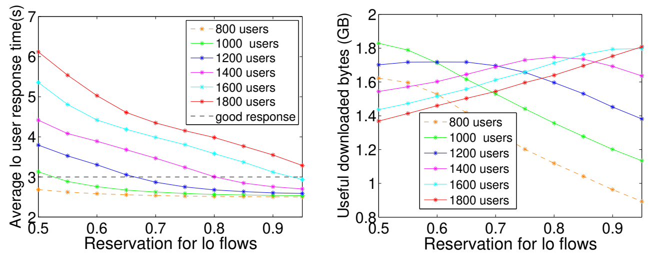


Fig. 11: Performance of xTRAC for a 32 Mbps link and different user loads across different values of reservation

In Fig. 11 the average low user response time and the useful bytes have been plotted as a function of the fraction of bandwidth reserved for low users; this is done for different user loads for a bottleneck link of 32Mbps. It may be noted from the plots that as expected, low user response times decrease when reservation is increased for them. The plot for useful bytes indicates that peaks for total useful bytes are obtained at different reservation values for different loads. This may be understood as follows - at heavy loads, a suitably large reservation for low users results in good performance for them; at the same time, this results in long wait good service (LWGS) treatment for high users - high users' flows wait a short while after which the requests are served fairly fast. Essentially most flows terminate without wastage in downloaded bytes, or completion without wastage.

TABLE I: Optimal reservation values for different user loads using xTRAC

User load	Resrv. lo users	Resp. time lo users(s)	Useful. bytes(GB)	$\frac{small_{lo}}{small_{hi}}$	$\frac{tput_{lo}}{tput_{hi}}$
800	0.5	2.68	1.62	0.92	1.14
1000	0.52	3.02	1.82	1.03	1.18
1200	0.65	3.05	1.71	1.72	1.29
1400	0.8	3.00	1.74	3.28	1.32
1600	0.93	3.01	1.80	12.25	1.28
1800	1.0	3.03	1.82	∞	NA

Next, an average response time of 3 seconds was chosen as the criterion for good performance for low users. Table I tabulates the reservations which correspond approximately to an average response time of 3 seconds for different traffic loads, and the corresponding performance. It indicates that a suitable reservation for low users leads to effective link usage, while controlling the flows of high users. Response times for low users are approximately 3s, useful bytes downloaded represent 85-90% of link utilization, as the maximum limit is

2GB (32Mbps for 500s). Low users flows are served several times faster than high users depending on the user load, eventhough the throughputs of high users remain nearly the same as low users.

TABLE II: Comparison of aggregate performance of xTRAC with plain TCP for different user loads

User load	Control Method	Resp. time (sec.)	Useful bytes (GB)	Drop rate
800	No control	2.45	1.71	$1.07 * 10^{-3}$
	xTRAC	2.72	1.62	0.0
1000	No control	3.21	1.80	$1.64 * 10^{-2}$
	xTRAC	3.21	1.82	$3.28 * 10^{-5}$
1200	No control	4.22	1.63	$4.40 * 10^{-2}$
	xTRAC	4.07	1.71	$2.3 * 10^{-4}$
1400	No control	5.17	1.44	$7.0 * 10^{-2}$
	xTRAC	4.05	1.74	$1.60 * 10^{-4}$
1600	No control	6.03	1.28	$9.0 * 10^{-2}$
	xTRAC	3.61	1.80	$2.13 * 10^{-4}$
1800	No control	6.86	1.15	$1.1 * 10^{-1}$
	xTRAC	3.03	1.82	$3.69 * 10^{-4}$

A comparison of aggregate performance of xTRAC with the no control case in Table II indicates that aggregate performance is better always except under light traffic conditions. A 58% increase in useful downloaded bytes is observed for the highest load of 1800 users.

VI. Conclusion

A technique to control excessive and wasteful usage of Internet during congestion has been devised; the additional benefit of the scheme is that it incentivizes regular users and penalizes rogue users during congestion, thereby motivating responsible usage and efficient usage of Internet access links. STCR, a simple algorithm based on the general TCR approach for rate controlling HTTP download traffic at the web proxy has been described. Then two schemes RTCR and xTRAC based on STCR were devised. While RTCR rate controlled the rogue users by dividing the unused bandwidth among individual flows in a weighted manner, xTRAC reserved a fraction of the bandwidth exclusively for each category and used TCR to rate control the flows of each category; it further performed admission control for each category. The former was limited in its control and aggregate performance; the latter on the other hand controlled rogue users seamlessly depending on the user load. During overload regular users are served several times faster than rogue users. The aggregate performance of the system is also better with xTRAC except in case of light loads. We have coined the term long wait good service (LWGS) to describe the treatment meted out to rogue users by xTRAC. They perceive a long wait followed by quick service when the flow is admitted. This serves to improve the goodput of the system.

The scheme incentivizes regular users and penalizes rogue users during congestion. This ultimately leads to control of rogue users, and easens out the congestion during peak hour.

Even-though we have implemented xTRAC at a web proxy, the same scheme may be implemented at any intermediate router. xTRAC, like TCR may be applied at any intermediate router through which Internet traffic egresses out of / ingresses into the network; this is true as long as it is possible to modify the advertised window field in the TCP header using deep packet inspection. On the other hand, admission control at a router would involve terminating new connections from the LAN to remote servers with a TCP RST when the maximum number of active connections has been reached.

A further benefit with xTRAC is the ease of deployment of such a scheme. Only a single host implementation needs to be changed; all other devices on the network are left intact.

References

[1] (2016) Packetguide 11.2. [Online]. Available: https://bto.bluecoat.com/packetguide/11.2/

[2] 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.

[3] 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.

[4] 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.

[5] A. Haq, H. B. Salau, A. Aibinu, and E. Onwuka, "A Survey of Bandwidth Utilization: Case Study of Federal University of Technology, Minna," in *IOP Conference Series: Materials Science and Engineering*, vol. 53, no. 1, 2013, p. 012054.

[6] S. Karandikar, S. Kalyanaraman, P. Bagal, and B. Packer, "TCP Rate Control," *ACM Computer Communications Review*, vol. 30, pp. 45–58, Jan. 2000.

[7] K. Kashibuchi, A. Jamalipour, and N. Kato, "Channel occupancy time based TCP rate control for improving fairness in IEEE 802.11 DCF," *IEEE Trans. Veh. Technol.*, vol. 59, pp. 2974–2985, Jul. 2010.

[8] H. Koga, K. Iida, and Y. Oie, "Receiver-based flow control mechanism with interlayer collaboration for real-time communication quality in w-CDMA networks," in *IFIP TC6 9th International Conference*, Sep. 2004, pp. 286–300.

[9] A. Kumar, M. Hegde, S. Anand, B. Bindu, D. Thirumurthy, and A. Kherani, "Nonintrusive TCP connection admission control for bandwidth management of an internet access link," *IEEE Commun. Mag.*, vol. 38, pp. 160–167, May 2000.

[10] 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.

[11] C.-P. Li and E. Modiano, "Receiver-based flow control for networks in overload," in *Proceedings of IEEE INFOCOM*, 2013, pp. 2949–2957.

[12] T.-C. Lin, Y. Sun, S.-C. Chang, S.-I. Chu, Y.-T. Chou, and M.-W. Li, "Management of abusive and unfair internet access by quota-based priority control," *Computer Networks*, vol. 44, pp. 441–462, Mar. 2004.

[13] R. Morris, "TCP behavior with many flows," in *Proceedings of IEEE International Conference on Network Protocols*, Oct. 1997, pp. 205–211.

[14] R. Mortier, I. Pratt, C. Clark, and S. Crosby, "Implicit admission control," *IEEE Journal on Selected Areas in Communications*, vol. 18, no. 12, pp. 2629–2639, Dec. 2000.

[15] T. A. Paine and T. J. Griggs, "Directing traffic: Managing internet bandwidth fairly," *EDUCAUSE Quarterly*, vol. 3, pp. 66–70, 2008.

[16] C. E. Palazzi, M. Brunati, and M. Roccetti, "An openWRT solution for future wireless homes," in *IEEE International Conference on Multimedia and Expo*, Jul. 2010, pp. 1701–1706.

[17] 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.

[18] H.-Y. Wei and Y.-D. Lin, "A survey and measurement-based comparison of bandwidth management techniques," *IEEE Communications Surveys*, vol. 5, pp. 10–21, 2003.