CLAMP: Differentiated Capacity Allocation in Access Networks

Lachlan Andrew, Stephen Hanly and Rami Mukhtar
ARC Special research Centre for Ultra-Broadband Information Networks (CUBIN)
Department of Electrical and Electronic Engineering
University of Melbourne, Victoria 3010, Australia
{lha,hanly,rgmukht}@ee.mu.oz.au

***Abstract*— This paper presents a solution for providing differentiated capacity allocation in an access network. The system is based on CLAMP, an algorithm that can differentiate between flows sharing the same FIFO queue. The system is suitable for small access networks, such as those based on DSL and HFC modems and wireless LAN access points. The deployment of CLAMP is completely contained within the access network; no changes to the remainder of the network are required. CLAMP provides the opportunity to enforce local policies on TCP flows that originate from sources distributed globally. The performance of CLAMP is verified by both simulation and analysis.**

***Index Terms*— End-to-end flow control, differentiated services, low bandwidth access networks, distributed resource allocation.**

I. Introduction

Shared access networks, in which one to tens of flows simultaneously share a low bandwidth connection to the internet, are increasingly common. In such networks, the low bandwidth connection will typically be the bottleneck. Examples include home and office networks with low-bandwidth connections to the internet such as digital subscriber line (DSL) or hybrid fibre/coax connections, private connections between branch offices, and most importantly, wireless LANs, in which the wireless link is the bottleneck. In such networks, with limited resources, the apportioning of bandwidth is especially important. It may be desirable to give preference to certain applications over others, for example giving priority to HTTP traffic over FTP traffic. In other circumstances, it may be desirable to distinguish between users (e.g., CEO versus intern, or parents versus children).

The deployment of a mechanism to provide differentiated bandwidth allocation within the access network must be completely independent of the rest of the internet. Accordingly, it has to inter-operate with existing TCP sender-side implementations. Furthermore, the mechanism must be cheap and easily implementable.

This paper presents an algorithm for differentiated capacity allocation by *C*urtailing the *L*arge TCP *A*dvertised windows to *M*aximize *P*erformance, which we refer to as CLAMP. It is applicable to access networks in which all receivers and the last hop router can all be configured to run CLAMP. It is most useful when there is no control over the sender (as is usually the case) and when there are one to tens of TCP flows simultaneously sending packets through the same access point.

CLAMP does not need any per flow state information, and it is totally distributed, making it very easy to implement. This paper assumes that all flows share a single first-in-first-out (FIFO) queue. This is the simplest approach and will be shown to be sufficient to provide differentiated service at the transport layer. However it is not a critical assumption for CLAMP, which is still useful in more general contexts [1].

This paper presents the CLAMP algorithm, describes how it is to be implemented, and provides analytical and simulation results illustrating its performance.

II. Related Work

Many end-to-end solutions exist for enforcing Quality of Service (QoS) on the Internet. The bulk of the proposals are end-to-end models that fall into two main categories: 1) the integrated services model (Intserv) [2] and 2) the differentiated services model (Diffserv) [3].

Intserv is capable of providing two classes of service, fixed delay for applications requiring a bounded transmission delay, and enhanced best effort for applications requiring low loss transmission of their datagrams. Intserv is based on the widespread implementation of RSVP [4], a signalling protocol that facilitates the reservation of resources along the end-to-end path in order to meet the

This work was in part funded by the Australian Research Council (ARC).

flow's QoS requirements. In addition to the widespread deployment of RSVP, Intserv requires per flow state information to be maintained by routers. Finally, senders must take part in the initiation of the resource reservation.

The aim of Intserv is totally distinct from that of CLAMP. Intserv is an Internet wide solution that focuses on providing a specific level of service to certain flows. Its deployment must be Internet wide, and its use is usually selected by the sender. In contrast CLAMP provides the access network administrator with a means to allocate the capacity of the access network in any desired proportion. It does not rely widespread Internet deployment, and the allocation policy is under the control of the access network administrator.

Diffserv attempts to overcome the complexity and scalability issues attributed to Intserv by dividing all flows into aggregate service classes [5] as indicated by setting combinations of the type-of-service bit in the IP header corresponding to either low delay, high throughput or low loss rate service. The application of Diffserv would provide a means to apply a particular allocation policy to the access network. However, it would have to be done by the sender, which is out of the control of the access network's administrator. Since CLAMP is a receiver side solution, it overcomes this limitation.

The operation of CLAMP is closest to the idea of a work conserving Round Robin scheduler with weighted service rates [6–8]. A Round Robin scheduler maintains a separate queue at the access point for each flow. In a weighted service regime, it will provide a different proportion of the service time to each queue. Unlike a multi-queue solution, CLAMP only requires the use of a single FIFO queue. Although the complexity of a multi-queue system for a relatively small number of flows is not unreasonable, the implications of a multi-queue system make its implementation impractical. The main difficulty lies in determining when to assign and de-assign a "virtual" queue to an identified flow. IP flows are stateless, and determining when a particular flow starts and stops is not trivial. CLAMP avoids this complexity by letting the receiver dynamically limit the proportion of data buffered in the single FIFO queue attributed to a particular flow, in a decentralised mannar. Accordingly, the access point does not need to maintain any state information. Furthermore, since CLAMP works with a single FIFO queue at the access point, its implementation is simple, and it can be implemented in a variety of different access network scenarios.

The goals of CLAMP are similar to those outlined in [9], which proposes the eXplicit Control Protocol (XCP). However, unlike CLAMP, XCP requires modifications to the sender and estimates of the RTT. Since RTT estimates are a key part of the operation of XCP, it would be difficult to apply it to a receiver-side implementation. XCP is more suited to a future internet when widespread changes are made to all clients and routing architectures.

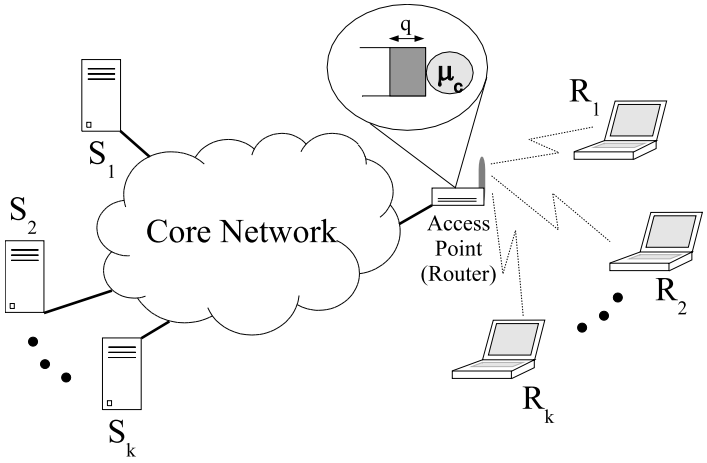

Fig. 1. Model of k flows sharing a single bottleneck access link.

III. System Topology

The topology of interest is shown in Fig. 1. There are k flows of data packets controlled by TCP that share a single bottleneck link from an access point, with output rate μ_c. Each flow, i, has a sending node, S_i, that is located somewhere in the Internet and a receiving node R_i located within the access network. Each flow is controlled by TCP.

Some flows carrying inelastic data, such as UDP traffic, that pass through the access point will not be controlled by TCP or any other congestion control mechanism. Our interest does not lie in controlling these flows, and so we assume they take a fixed fraction of the access link capacity, and the remaining bandwidth is to be shared among elastic flows via TCP. Our aim is to use CLAMP to ensure that these flows conform to a desired capacity allocation policy.

IV. The Algorithm

The CLAMP algorithm assumes that each sending node implements TCP flow control. Under the assumption that sources are greedy, the total number of packets and acknowledgements in flight at any time, t, is equal to the minimum of the sender's current congestion window (CWND) and the receiver's current advertised window (AWND). CWND is solely controlled by the sender, and can not be explicitly set by an element located in the

access network. However, the value of AWND is controlled by the receiver. We propose CLAMP, an algorithm that will select the current value of AWND for each node i, denoted $w_i(t)$, in a decentralized way, such that each flow obtains a proportional share of the channel rate, μ_c, and the equilibrium buffer occupancy of the access router, $q(t)$, can be controlled as discussed below. CLAMP is an enhancement of an algorithm that we proposed in [10]. It has been modified here to provide differentiated rate allocation, and operation compatible with non-greedy sources.

CLAMP retains compatibility with the *end-to-end* Internet framework [11,12]. It only requires aggregate (as opposed to flow-by-flow) feedback from the access point, and is fully compliant with existing transport layer standards.

The CLAMP system is comprised of two distinct components, both of which reside in the access network. The first component is a software agent that is integrated into the access point, be it a DSL central office modem or wireless LAN access point. The second component is a software agent located in the receiver. It may take the form of a network interface driver, network interface card adapter or a modification to the operating system kernel. Both of these components will now be described in more detail.

A. Access Router Agent

The software agent in the access point simply samples the queue length, q, of the FIFO queue at regular intervals. It then computes a convex monotonic increasing function of q, $p(q)$, which is then passed to each receiver. This may be achieved by inserting the value into the TCP header of each packet leaving the access router (for example in the TCP options field). Alternatively each receiver can explicitly request the value from the access point as required. This paper paper focuses on the affine function

$$p(q) = \frac{bq - a}{\mu_c}, \tag{1}$$

where μ_c is the rate of the bottleneck link, and the constant b determines how sensitive the bottleneck queue size is to the number of flows. The parameters a and b control the equilibrium mean queue size, q^*, as will be seen in Section VI.

B. Receiver Agent

A software agent located at the receiver intercepts the value of $p(q)$ advertised by the access point and sets the advertised window value, $w(t)$, in all outgoing TCP acknowledgments according to the following algorithm. For simplicity, the algorithm will be described for flow i in the case of equal-length packets. Let t_k denote the time instant when the kth packet is received by the receiving client. The algorithm is:

$$w(t_k) = \begin{cases} w(t_{k-1}) - 1 & \text{if } \Delta w(t_k) < -1 \\ w(t_{k-1}) + \Delta w(t_k) & \text{if } -1 \le \Delta w(t_k) \le \overline{\Delta} \\ w(t_{k-1}) + \overline{\Delta} & \text{if } \Delta w(t_k) > \overline{\Delta} \end{cases} \tag{2}$$

where

$$\Delta w(t_k) = [\phi_i \tau - p(q(t_k))\tilde{\mu}(t_k)](t_k - t_{k-1}) \tag{3}$$

and $\phi_i > 0$ is a positive constant, $\tau > 0$ (packets/sec) is a constant, and $\tilde{\mu}$ (packets/sec) is an estimate of the received rate. The term in τ tries to increase the window at a constant rate, while the term in $\tilde{\mu}$ reduces it at a rate which increases with the occupancy of the queue and with the proportion of traffic due to the flow. The current received rate, $\tilde{\mu}$, is estimated using a sliding window averaging function,

$$\tilde{\mu}(t_k) = \frac{\alpha}{t_k - t_{k-\alpha}}, \tag{4}$$

where the integer α is a smoothing factor. This choice of estimator is somewhat arbitrary, and other estimators may prove to be more effective.

The maximum window increase in (2) is limited to a constant, $\overline{\Delta} > 0$. This prevents large $t_k - t_{k-1}$ from causing large changes in w when packets arrive infrequently, such as when a source becomes idle for an extended period of time.

Furthermore, the limit on the window decrease in (2) reflects that fact that a packet can only be removed from the network when it is received. This remains compliant with the recommendations of Section 3.7 of RFC 793 [13].

The flow control algorithm can provide non-uniform sharing of the bottleneck bandwidth by appropriately setting the constants ϕ_i. It will be shown by simulation in Section V and by analysis in Section VI that, under certain conditions, flow i will obtain the proportion $\phi_i / \sum_{j=1}^{k} \phi_j$ of the bottleneck capacity. The remainder of the paper is dedicated to analyzing the performance of the system (2) and how to configure its parameters for stable operation.

Finally, note that the algorithm as described does not prevent the window size falling to zero. Since window updates only occur on receiving a packet, this would cause the window to remain at zero indefinitely. There are several techniques that can be used to deal with this special case. However, a simple solution is to limit the minimum window size to a constant, $w_{\min}$. The simplest case is to take $w_{\min} = 1$.

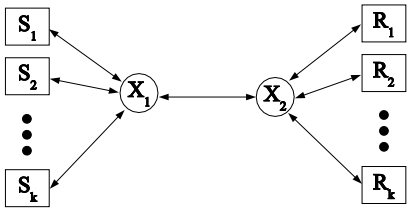

Fig. 2. Simulation network topology

V. Simulation Results and Discussion

This section demonstrates that the proportion of capacity each flow obtains can be set by appropriately choosing values of ϕ_i of each flow i. This is done by simulating a test network topology running the CLAMP system.

The algorithm was implemented as an event driven network simulator. The main objective of the simulation was to simulate a true system as accurately as possible. Each sender implemented TCP flow control, all routers employed drop tail queues.

The simulation topology used for experimentation is illustrated in Fig. 2, in which X_1 and X_2 are routers, $S_1, S_2, \ldots, S_k$ are sending clients, and $R_1, R_2, \ldots, R_k$ are receiving clients. All links are bidirectional, with characteristics as shown in Table I.

Node 1	Node 2	Capacity	Delay
S_i	X_1	10 Mb/s	$d_i/2$
X_1	X_2	1.5 Mb/s	1 msec
X_2	R_i	10 Mb/s	1 msec

TABLE I
Link configuration

For simplicity, all packets were of fixed size. An agent was placed in the router, X_1, that constantly monitored the total FIFO queue size, and inserted the value into the header of all outgoing packets. Agents at the receiving nodes ran the algorithm as described in Section IV, placing the computed window size (rounded to the nearest packet) into the `awnd` field of outgoing acknowledgements. Except when stated otherwise, all simulation parameters are indicated in Table II.

A. Differentiated Sharing of Access Capacity

The first experiment was to verify that CLAMP does in fact provide differentiated sharing of the access point's capacity. Accordingly, the system shown in Fig. 2 was simulated for $k = 4$. The sources S_1, S_2, S_3 and S_4 were started at 0, 50, 100 and 150 seconds respectively. The results as shown in Figs. 3-4 illustrate how CLAMP is effective in dividing the access point's capacity independently

TABLE II
Simulation Parameters

Parameter	Value
TCP Packet Size	500 Bytes
CLAMP τ	10000 Bytes/s
CLAMP $\overline{\Delta}$	80000 Bytes
CLAMP b	$2\,\mathrm{s}^{-1}$
CLAMP a	2000 Bytes/s
d_1	0.41224 s
d_2	0.2943 s
d_3	0.075644 s
d_4	0.0001421 s

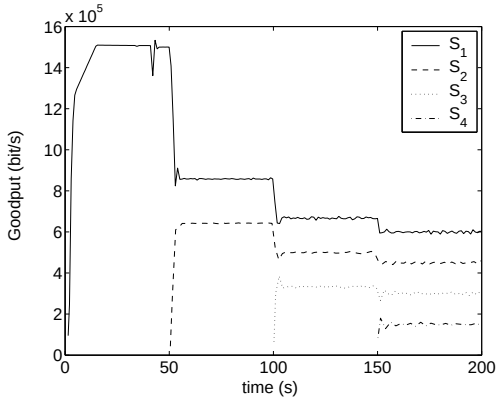

Fig. 3. Simulation of CLAMP providing differentiated allocation of the access point's capacity to each flow, $\phi_1 = 0.5, \phi_2 = 1, \phi_3 = 1.5, \phi_4 = 2$.

of the respective RTTs of each flow. Note that, at each stage, the allocation of capacity is in proportion to the ratio of the ϕs. For example, in Figure 3, the ratio of the capacities at 75 s is 2 : 1.5 : 1 and at 175 s it is 2 : 1.5 : 1 : 0.5. The values of ϕ_i for all i in each experiment are given in the captions of the respective figures.

It is important to highlight that, given disparate propagation delays, TCP would have provided greater capacity to the flows with the shorter RTTs. In contrast CLAMP avoids this limitation, by controlling TCP to provide any given proportional allocation of capacity.

B. Bottleneck in the Core

This section investigates the effect of a bottleneck elsewhere in the core as a result of congestion in the core. In this situation, CLAMP will not be able to provide the desired allocation of capacities to all flows. This is due to the effect that the flows which pass through a bottleneck in the core will be constrained by that bottleneck. However, CLAMP will divide the remaining capacity between the unconstrained flows according to the desired proportion allocation.

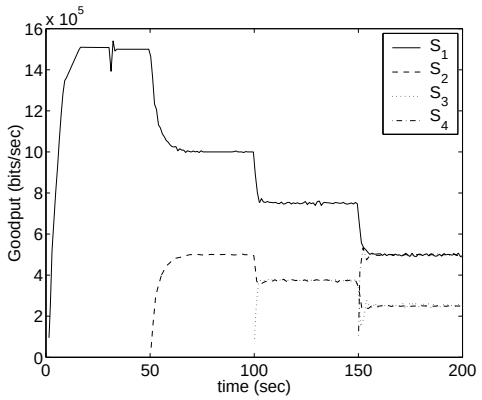

Fig. 4. Simulation of CLAMP providing differentiated allocation of the access point's capacity to each flow, $\phi_1 = 1, \phi_2 = 0.5, \phi_3 = 0.5, \phi_4 = 1$.

Congestion in the core is simulated by starting a cross traffic TCP session within the core.

The network topology that was simulated is illustrated in Fig. 5. $S_i, i = 1, 2, 3, 4$ are greedy TCP sources and R_i are the matching receivers, running CLAMP. Both X_1 and X_2 are core routers, X_3 the access router running the CLAMP router agent, and X_3 a LAN switch. All link capacities and delays are listed in Table I and other simulation parameters are listed in Table II. For this experiment $\phi_i = 1$, for all i, so that each flow gets an equal share of the bottleneck capacity.

The cross traffic source is another TCP source fed by a greedy traffic source, which was started at 100 s and stopped at 300 s. Fig. 6 is plot of the sequence numbers of the received acknowledgments versus time.

The parallel lines before the 100 second mark indicate that all four flows are receiving an equal share of the bottleneck link capacity. After the introduction of cross traffic within the core, link X_2–X_3 replaces X_3–X_4 as the bottleneck for connections 1 and 3. At this time CLAMP automatically relinquishes control of these connections back to TCP, but continues to control connections 2 and 4. Connection 1, with its very long RTT, gets almost no capacity from link X_2–X_3, while connections 2 and 4 share the spare bandwidth on link X_3–X_4 almost fairly. After 300 seconds, the cross traffic source ceases transmission, restoring X_3–X_4 as the bottleneck, and subsequently CLAMP resumes control of all the flows. After a transient period, once again each flow obtains an equal share of the bottleneck link capacity (all four lines becoming parallel again). While connection 1 is controlled by TCP, its $w(t)$ becomes very large. As a result, it gets a greater share of the bandwidth during the transient after 300 seconds, increasing the long-term fairness of CLAMP. If fairness on short timescales is desirable, then the maximum value of $w(t)$ can be clipped.

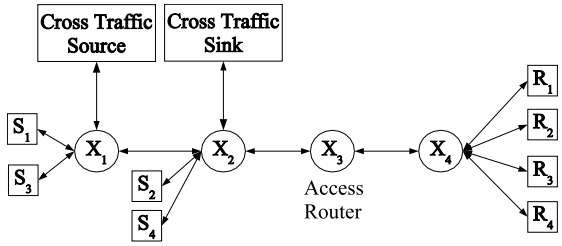

Fig. 5. Simulated topology that includes cross traffic.

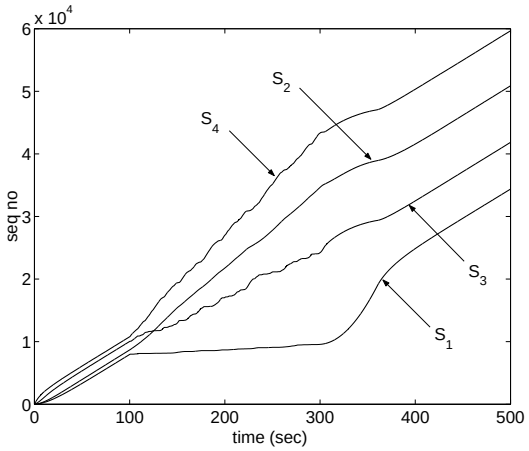

Fig. 6. Plot of sequence number when two flows are controlled by a bottleneck in the core.

VI. The Fluid-Flow Model

In order to provide deeper insight into the algorithm's performance, this section presents the analysis of a fluid flow model of the system. In particular, this model is used to determine the system's equilibrium point, which verifies the proportional sharing that was observed in the simulations. Conditions required for stable operation are also obtained, which are necessary for setting the parameters of the algorithm.

For each flow i, let d_i be the propagation time as defined in Section IV, $w_i(t)$ be the window size, and $B_i(t)$ be the buffering at the bottleneck queue attributed to the flow, satisfying

$$q(t) = \sum_{i=1}^{k} B_i(k). \tag{5}$$

Consider a fluid flow approximation to the system described in Section IV, with perfect mixing of fluid at the bottleneck queue, i.e., the proportion of fluid leaving the queue attributed to flow i is $B_i(t)/\sum_{j=1}^{k} B_j(t)$.

The total number of packets in the network at time t attributed to flow i is equal to the number buffered in the bottleneck queue plus the number in flight. The number in flight at time t is equal to the number that left the queue in the interval $(t, t - d_i]$ plus or minus added or removed due

to changes in window size during that interval. Hence,

$$w_i(t) = B_i(t) + \int_{t-d_i}^{t} \left(\frac{B_i(s)}{q(s)}\mu_c + \left.\frac{dw_i}{dt}\right|_{t=s} \right) ds. \quad (6)$$

Differentiating (6) gives the rate of change of buffering at the bottleneck queue attributed to flow i:

$$\frac{dB_i}{dt} = \left.\frac{dw_i}{dt}\right|_{t-d_i} + \frac{B_i(t-d_i)}{q(t-d_i)}\mu_c - \frac{B_i(t)}{q(t)}\mu_c. \quad (7a)$$

Under the flow control algorithm (2), a fluid model of the window size evolution is

$$\frac{dw_i}{dt} = \begin{cases} -\frac{B_i(t)}{q(t)}\mu_c & \text{if } g_i(t) \le -\frac{B_i(t)}{q(t)}\mu_c \\ g_i(t) & \text{if } -\mu_c < \frac{q(t)}{B_i(t)} g_i(t) \le \mu_c\overline{\Delta} \\ \frac{B_i(t)}{q(t)}\mu_c\overline{\Delta} + \epsilon & \text{otherwise} \end{cases} \quad (7b)$$

where ϵ is a small constant, and

$$g_i(t) = \phi_i\tau - p(q(t))\frac{B_i(t)}{q(t)}\mu_c. \quad (7c)$$

The term $(B_i(t)/q(t))\mu_c\overline{\Delta}$ reflects the maximum increment to the window size, $\overline{\Delta}$, that can occur with every packet that arrives at the receiver in the physical model. The extra term ϵ serves the same purpose as $w_{\min}$ in the actual algorithm; it prevents the window from being stuck at 0. A small value of ϵ will provide the necessary compensation; a large value will unduly distort the model.

The term $-B_i(t)/q(t)\mu_c$, for the case when $g_i(t) \le -\mu_c B_i(t)/q(t)$, represents the fact that in the physical model, the window cannot be decreased at the sender faster than the rate of arrival of acknowledgements from the receiver, which is the rate of arrival of packets at the receiver from the bottleneck queue.

Another discrepancy between the physical model and the fluid model concerns delay. In the fluid model, the only delay in the system is the propagation delay in the network, modelled as a constant. In the real, physical system, packets are also delayed in the bottleneck queue. However, information regarding the total queue size does not suffer this latter delay.

Clearly, there are discrepancies between the physical model and the fluid model. Nevertheless, the fluid model offers many advantages from the point of view of analytical understanding. It will be shown later that insights from the fluid model do carry over to simulation results for the physical model that were presented in Section V.

Although the fluid model has been described by differential equations, it is important to recognize that there are points of discontinuity in the fluid model. These occur whenever the bottleneck queue empties, for then the total output rate of the bottleneck queue switches from μ_c to zero. Then, when the queue starts to fill again, the output rate switches back to μ_c, another point of discontinuity. To be consistent with this physical description, we define $0/0 \equiv 0$ in the last terms of (7a). It is easy to see that when the queue empties, the first two terms of (7a) sum to a non-negative value, so no B_i will go negative during this period, and at some later point the queue will begin to increase again and become strictly positive. The point in time when the queue starts to increase again is another point of discontinuity of the fluid model, as the total output rate of the queue switches back to μ_c. However, apart from these isolated points of discontinuity, the system is continuous, and is properly described by the differential equations between points of discontinuity.

The change in the total queue size is obtained by summing (7a) over i,

$$\frac{dq}{dt} = \sum_{i=1}^{k} \left.\frac{dw_i}{dt}\right|_{t-d_i} + \sum_{i=1}^{k} \frac{B_i(t-d_i)}{q(t-d_i)}\mu_c - \rho(t)\mu_c, \quad (8)$$

where $\rho(t)$ is the utilisation of the bottleneck channel at time t.

Lemma 1: If $0 \le B_i(t) \le w_i(t)$ for all $t \le 0$ and all $i = 1, \ldots, k$, then under (7),

$$0 \le B_i(t) \le w_i(t) \quad (9)$$

for all $t \ge 0$ and all $i = 1, \ldots, k$. Moreover, $B_i(t) > 0$ whenever $q(t) > 0$.

Proof: Assume first that $B_i(t) \ge 0$ for all i and t, from which it follows that $q(t) \ge 0$. Then (7b) implies $dw_i/dt \ge -\mu_c B_i(t)/q(t)$, and the integrand in (6) is always non-negative, giving the second inequality.

To see that $B_i(t) \ge 0$ for $t \ge 0$, note that this is trivially true during the periods when the queue is empty. For during these periods, the last term of (7a) is by definition zero, and the sum of the other two terms is non-negative.

The other intervals to consider are when the queue is nonzero. In this case, suppose that $B_i(t)$ decreases, passing through zero at some time $\hat{t}$, at which point its derivative must be negative. But then $B_i(\hat{t}) = 0$, so the last term of (7a) is zero. But the sum of the first terms is non-negative by (7b), providing a contradiction. Hence, $B_i(t)$ cannot become non-positive during these periods. ■

The next part of the analysis characterises the equilibrium behaviour of the fluid model. Let

$$T = \tau \sum_{j=1}^{k} \phi_j \quad (10)$$

denote the aggregate rate at which users would increase their window sizes if $p(\cdot)$ were zero.

Lemma 2: There is a unique equilibrium point defined, for all i, by

$$B_i^* = \frac{\phi_i}{\sum_{j=1}^k \phi_j} p^{-1}\left(\frac{T}{\mu_c}\right). \quad (11)$$

Proof: Substituting (11) into (7) shows it is an equilibrium. To see it is unique, assume there is another equilibrium point, B^+, with $q^+ = \sum_{i=1}^k B_i^+$ and $dw_i/dt = 0$ for all i. At B^+, $g_i(t) = 0$ for all i by (7b) (noting that $\epsilon > 0$), whence for all i, (7c) implies

$$B_i^+ = \frac{\phi_i \tau q^+}{\mu_c p(q^+)}. \quad (12)$$

Summing (12) over i yields

$$q^+ = p^{-1}\left(\frac{T}{\mu_c}\right). \quad (13)$$

Substitution of (13) into (12) shows that $B_i^+ = B_i^*$, establishing uniqueness. ■

Lemma 2 says that under the assumptions that all sources are greedy and the system is stable, flow i will obtain the proportion $\phi_i / \sum_{j=1}^k \phi_j$ of the bottleneck link capacity.

VII. Stability Analysis: Necessary Conditions

This section presents some necessary conditions for the stability of (7) when $p(\cdot)$ is given by (1). Note that for a system to be asymptotically stable, its linearisation must also be stable, and that in any linearised system $p(\cdot)$ must take this form.

A. Equal Delays

Theorem 1: Let $d_i = d$ for all i, $B_i(t) \geq 0$ for all $t < 0$, and $p(\cdot)$ be given by (1). Then a necessary condition for the total queue size, $q(t)$, to converge under (7) is:

$$b < \frac{\pi}{2d}. \quad (14)$$

Proof: Near equilibrium, the window evolution is governed by the equations

$$\frac{dw_i}{dt} = g_i(t)$$

since locally the other terms in (7b) are not biting. Substituting this expression into (8) and summing over i gives the linear equation

$$\frac{dq}{dt} = T + a - bq(t-d), \quad (15)$$

which can be analyzed by standard techniques [14]. Taking the unilateral Laplace transform gives

$$Q(s) = \frac{q(0^-)}{s + be^{-ds}}. \quad (16)$$

The (infinite number of) poles of (16) determine the limiting behaviour of (15). For $d = 0$ the system has a single real pole located at $s = -b$, hence is stable for all $b \geq 0$. For $b \geq 0$ as d is increased, an infinite number of poles will appear from infinity on the left half plane, and ultimately cross the imaginary axis. Applying the method shown in [14], p26, (14) is obtained as a necessary and sufficient constraint for stability of (15). ■

B. Arbitrary Delays: Constant Queue Size

It will be seen that significant insight into the behaviour of (7) can be obtained by considering the simplified system defined by (7a) for the case where $q(t) = q^*$ is constant, $q^* = (T + a)/b$ is the equilibrium value. In this case equations (7a) decouple to form the linear constant coefficient delay equations,

$$\frac{dB_i}{dt} = \phi_i \tau + \frac{(\mu_c - T)}{q^*} B_i(t - d_i) + \frac{\mu_c B_i(t)}{q^*}. \quad (17)$$

Proposition 1: A necessary and sufficient condition for (17) to converge for all $b < b_0$, is that either

$$T \leq 2\mu_c, \quad (18)$$

or

$$d_i \leq \frac{1}{b_0} \cos^{-1}\left(\frac{\mu_c}{\mu_c - T}\right) \frac{T + a}{\sqrt{T^2 - 2\mu_c T}}. \quad (19)$$

Proof: The stability of equation (17) for any value of b, is equivalent to the poles of the unilateral Laplace transform of the unforced equation,

$$\begin{aligned} M_i(s) &= \frac{B_i(0^-)/q^*}{s + (\mu_c/q^*) + (\mu_c - T/q^*)e^{-sd_i}} \quad (20) \\ &= \frac{B_i(0^-)/q^*}{A(s) + C(s)e^{-sd_i}} \end{aligned}$$

all being in the left half plane. The stability analysis now follows that of [14]. For sufficiently small d_i this system is stable, as it is a retarded system. Let $d_i(b)$ be the smallest positive delay such that $s = j\omega$, $\omega \in \mathbb{R}$, is a pole. By the continuity of poles, the system will be stable for all $d_i < d_i(b)$. Since poles occur in complex conjugate pairs, this is further equivalent to $A(j\omega)A(-j\omega) = C(j\omega)C(-j\omega)$ [14], or equivalently

$$\omega^2 = \frac{T^2 - 2\mu_c T}{(q^*)^2}.$$

This has no non-zero real solution for $T \leq 2\mu_c$, and therefore (17) is always stable in this case. This establishes the sufficiency of condition (18). It remains to show that, if $T > 2\mu_c$ then (19) is necessary and sufficient.

When $T > 2\mu_c$, $d_i(b)$ can be found by equating the real part of the denominator of (20) to zero:

$$\frac{\mu_c}{q^*} - \frac{\mu_c - k\tau}{q^*} \cos\left(\frac{d_i(b)\sqrt{T^2 - 2\mu_c T}}{q^*}\right) = 0$$

or equivalently

$$\cos\left(\frac{d_i(b)\sqrt{T^2 - 2\mu_c T}}{q^*}\right) = \frac{\mu_c}{\mu_c - T}.$$

Taking the smallest positive argument argument of $\cos(\cdot)$,

$$d_i(b) = \cos^{-1}\left(\frac{\mu_c}{\mu_c - T}\right) \frac{q^*(b)}{\sqrt{T^2 - 2\mu_c T}}.$$

Note that $d_i(b)$ increases to infinity as b decreases to zero. Define

$$\beta \equiv \cos^{-1}\left(\frac{\mu_c}{\mu_c - T}\right) \frac{T + a}{\sqrt{T^2 - 2\mu_c T}}.$$

Necessity: if $d_i > \beta/b_0$ then there exists a $b < b_0$ such that $d_i(b) = d_i$, and hence (17) is not stable. *Sufficiency:* if $d_i < \beta/b_0$ then for all $b < b_0$, $d_i < \beta/b$, and hence (17) is stable. ■

Corollary 1: The condition ((18) or (19)) is necessary for the stability of the unique equilibrium of (7).

VIII. Conclusions

This paper has presented CLAMP, an algorithm for differentiated proportional allocation of the capacity of a bottleneck link. The algorithm fits into the Internet end-to-end framework, requiring only aggregate congestion information from the final router in the network. It is completely compatible with existing TCP senders and routers in the core network.

Simulation results have indicated that CLAMP is an effective access network modification that provides differentiated proportional sharing of the access point's capacity.

A fluid flow approximation of the system was presented and analyzed in order to determine conditions for which the algorithm is stable. These results can be used to configure the algorithm's parameters for stable operation.

References

[1] L. L. H. Andrew, S. V. Hanly, and R. G. Mukhtar, "CLAMP: Maximizing the performance of TCP over low bandwidth variable rate access links," *Submitted for publication in 2002*.

[2] R. Braden, D. Clark, and S. Shenker, "Integrated services in the internet architechture: an overview," RFC 1633, IETF, 1994.

[3] S. Blake, D. Black, M. Carlson, E. Davies, Z. Wang, and W. Weiss, "An architecture for differentiated services," RFC 2475, IETF, 1998.

[4] R. Braden, L. Zhang, S. Berson, S. Herzog, and S. Jamin, "Resource reservation protocol (RSVP) – version 1 functional specification," RFC 2205, IETF, 1997.

[5] J. Postel, "Service mappings," RFC 795, IETF, 1981.

[6] S. Morgan, "Queueing disciplines and passive congestion control in byte-stream networks," in *Proc. INFOCOM*, vol. 2, pp. 711–720, IEEE, IEEE, 1989.

[7] L. Kleinrock, *Queueing Systems*. New York, NY: John Wiley and Sons, 1975.

[8] E. Hahne, "Round-robin scheduling for max-min fairness in data networks," *IEEE Journal on Selected Areas in Communications*, vol. 9, pp. 1024 –1039, September 1991.

[9] D. Katabi, M. Handley, and C. Rohrs, "Internet congestion control for future high bandwidth-delay product environments," in *Proc. ACM Sigcomm 2002*, August, 2002.

[10] L. Andrew, S. Hanly, and R. Mukhtar, "Analysis of rate adjustment by managing inflows," in *Proc. 4th Asian Control Conference*, (Singapore), pp. 47–52, 2002.

[11] J. H. Saltzer, D. P. Reed, and D. D. Clark., "End-to-end arguments in system design," *ACM Transactions in Computer Systems*, vol. 2, no. 4, pp. 277–288, 1984.

[12] S. H. Low, "A duality model of TCP and queue management algorithms," in *Proc. ITC Specialist Seminar on IP Traffic Measurement, Modeling and Management*, (Monterey, CA), 2002.

[13] Information Sciences Institute University of Southern California, "Transmission control protocol," RFC 793, IETF, 1981.

[14] J. E. Marshall, H. Górecki, K. Walton, and A. Korytowski, *Time-Delay Systems: Stability and Performance Criteria with Applications*. New York, NY: Ellis Horwood, 1992.