

CLAMP: Maximizing the Performance of TCP over Low Bandwidth Variable Rate Access Links

Lachlan L. H. Andrew, Stephen V. Hanly and Rami G. 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

CUBIN technical report # 2004-02-01

Abstract—This paper presents CLAMP, a distributed algorithm to enhance the performance of TCP connections that terminate in a wireless access network. CLAMP works at a receiver to control a TCP sender by setting the TCP receiver’s advertised window limit. Coupled with existing TCP sender side implementations and active queue management, CLAMP provides a complete end-to-end flow control scheme optimised for wireless access networks. CLAMP provides explicit control over wireless link utilisation and queueing delay, and allows a controlled proportion of the link capacity to be allocated to each flow sharing the same radio bearer. The only information required from the access network is the aggregate bottleneck queue size; specifically, it does not rely on propagation delay estimates. No modifications are required to the sender side or the core network. The performance of CLAMP is investigated under a range of network conditions and conditions for convergence are found.

Index Terms—TCP Performance Enhancement, differentiated services, low bandwidth access networks, distributed resource allocation, delay differential equations.

I. INTRODUCTION

TCP is the dominant transport layer protocol for data transfers in the internet. It has been a remarkably successful protocol; its simple and scalable bandwidth probing, coupled with reliable packet delivery, have provided the robustness required for rapid growth. Nevertheless, recent trends have revealed shortcomings that need to be remedied in order to maintain the rapid expansion of internet services. One of these is the need to provide a range of heterogeneous services, where latency, as well as throughput, is an important issue, and where it is possible to provide “differentiated services”. The other is the need to provide these services over heterogeneous physical networks; in particular, the modern, time-varying, wireless access networks, such as 3G cellular, and wireless LANs. The purpose of the present paper is to introduce a new TCP-compatible algorithm that is suited to this modern, heterogeneous environment.

Traditional wireless cellular networks were voice telephony networks, and were designed accordingly, with latency being a key performance metric. With the advent of today’s wireless LANs, and cellular networks providing 3G services, the emphasis has changed to providing differentiated services,

including high rate data services, where throughput is also a key concern. A large body of research on the wireless physical layer (surveyed in [1]) and link and MAC layers (see [2] and the references therein) has allowed this transition, with new coding techniques, signal processing algorithms, and link and MAC layer protocols designed to control the tradeoff between throughput and latency as dictated by service requirements.

Voice requires very low latency and jitter, but many others are prepared to trade latency for throughput. Streaming video also requires low latency, but high throughput is also a priority. Services such as background data transfers (*ftp*) and interactive services (human-machine interactions such as web browsing) can tolerate more delay, and typically operate over the Transport Control Protocol (TCP). While much recent research has focussed on the provisioning of these services at the lower layers, not much attention has been paid to the interaction with the network, including higher layer protocols, and the effect of long propagation delays in the core network. TCP itself is not sympathetic to these lower-layer optimizations.

On the other hand, there has been a large amount of research on TCP over wireless channels, mainly focusing on the important issue of packet loss due to wireless channel impairments, and ways to improve this at the link layer ([3], [4], [5], [6], [7], and many other references), but the particular problem of controlling the trade-off between latency and throughput has not been considered. Although TCP has been designed for elastic traffic, where latency is not important, interactions with lower layers can reduce the effectiveness of lower layer mechanisms for increasing throughput ([8], [9], [10]). Moreover, for many interactive services that operate over TCP, latency is still an important issue, and delay-sensitive flows may share the same access buffer as TCP traffic.

In the present paper, we present a higher-layer flow control algorithm, that allows control over the tradeoff between latency and utilization for individual services, and which can provide service differentiation between traffic classes that share a common radio bearer. It uses the fact that retransmissions at the link layer provides low packet loss rate at the expense of variable packet transmission times [5], [11]. An important purpose of the algorithm is to prevent undesirable interactions between layers. Indeed, the demonstrations of the algorithm consider the impact of propagation delay in the core network, fading at the wireless link, and channel-

state-aware scheduling. The algorithm runs as a receiver-side software agent that interworks with standard TCP, and does not require widespread deployment in the Internet. It controls transmission by adjusting the advertised window set by the receiver, and is called “CLAMP” since it works by “Curtailing Large Advertised-windows to Maximize Performance”. The algorithm is described here as a transport layer enhancement, to benefit interactive data traffic operating over TCP. However, the algorithm itself could also be implemented at the application layer to control latency and throughput for streaming traffic.

In order to avoid starvation of the link layer scheduler, CLAMP attempts to maintain a set mean queue at the wireless access point. This is particularly important due to the variability in the capacity of the wireless link, induced by fading. Links with a constant capacity need only buffer against the randomness of the traffic, while wireless links should buffer enough packets to take advantage of good channel conditions when they occur ([12], [13], [14]).

CLAMP also aims to ensure that all services sharing the wireless bandwidth obtain a fair share of the capacity commensurate with their requirements. Uploading a web page should have priority over a background file transfer, regardless of the geographic locations of the servers for each. This aspect of CLAMP makes it of interest for wireline access as well, although this is not the subject of the present paper.

CLAMP is distributed; the only information required from the network is a measure of the aggregate queue size at the bottleneck node; no per flow information is required, nor are estimates of RTTs. CLAMP works in conjunction with existing Internet congestion control algorithms, providing a performance enhancement when the access link has a low and time varying capacity.

II. MOTIVATION AND RELATED WORK

Cutting edge wireless technologies are designed specifically with data in mind. Enhancements such as link rate adaptation ([2]), incremental redundancy ([15] and references therein), and channel-state-aware scheduling ([13], [14]) change the focus of TCP research from mitigating the effect of errors on TCP to controlling TCP to ensure that queueing latency is low and link utilisation is high.

Techniques for improving TCP’s performance over wireless links can be grouped into three categories [3]: sender-side modifications, receiver-side modifications, and intermediate Performance Enhancing Proxies (PEP).

Sender-side modifications require TCP senders to modify their behaviour when communicating to a peer host that is connected through a wireless link. For example, TCP Westwood [16] and the proposal by Samaraweera [17] attempt to distinguish between congestion-related losses and wireless losses by estimating the bandwidth delay product. The key drawback of sender side modifications is that they require widespread changes to Internet hosts.

PEP based schemes [18] address an essentially link layer issue at the transport layer. They attempt to hide wireless-related losses from the end hosts by locally retransmitting

packets lost at the air interface from an intermediate proxy node located within the network. These schemes fall into two sub-categories: split connection and packet interception. Split connection schemes terminate the TCP connection at the PEP, and form a new connection to the mobile host. This effectively reduces the TCP control loop between the sender and receiver. In contrast, interception based schemes [19] buffer TCP packets that pass through the PEP and look for duplicate acknowledgements, which the receiver uses to indicate packet loss. The PEP withholds these acknowledgements and retransmits the lost packets over the air interface. The main drawback of using PEPs is that they break end-to-end semantics. This limits the application of IP layer security (since IP payloads must be accessed), scalability, asymmetric routing, and failure detection [18]. For these reasons PEPs have not seen any widespread deployment.

The limitations of PEPs and sender-side modifications raise the question: Can we fundamentally change the behavior of TCP without making widespread changes to the Internet? Fortunately, receiver-side modifications provide a third, more practical alternative.

Receiver-side modifications only require changes to wireless hosts, which is far easier to achieve. Examples that focus on improving the performance of a low bandwidth access link by setting the receivers AWND value are presented in [20–23]. They use feedback from the access router coupled with a software agent at the receiver to set the TCP receiver window. Experiment and analysis have shown that this can increase utilization [24], [25], improve fairness and provide the ability to prioritize between flows.

CLAMP is another receiver side TCP enhancement using AWND to control flow rates. Unlike [23], it adapts the window size to the congestion level.

A problem with many enhancements of TCP is their need for accurate estimates of the propagation time devoid of any queueing delays. This is particularly so in schemes such as TCP Vegas [26–28] which aim to control the total data buffered in the network, or schemes like [21,29] which aim to control a bottleneck queue size. These schemes all introduce an approximately constant queueing delay in proportion to the number of flows using each queue. Late arriving flows will incorrectly attribute some queueing delay to propagation delay, giving an unfair bias towards long lasting flows. Unlike these schemes, CLAMP does not need accurate estimates of the propagation delay.

In terms of comparing CLAMP with TCP-Vegas, it should be pointed out that Vegas is strictly an end-to-end protocol, whilst CLAMP does require the intervention of the last access router (base station in the wireless context) and the user agent at the receiver side. Thus, it is not fair to make a direct comparison of, say, TCP Reno + CLAMP, against TCP Vegas.

The eXplicit Control Protocol (XCP) [30] has similar objectives to CLAMP. However, like Vegas, XCP requires estimates of the RTT and modifications to the sender-side. Since RTT estimates are a key part of the operation of XCP, it would not be possible 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.

A scheme [31] that pursues similar objectives has recently been proposed for asynchronous transfer mode (ATM) networks. It uses a rate based controller, which is centralised at the bottleneck. The controller maintains a target queue at the bottleneck link while providing fair capacity allocation to each flow. Unlike [31], CLAMP does not require per flow information, making it more suitable for deployment on the Internet.

CLAMP meets objectives similar to those of [30], [31], [20], using a technique similar to that of [21]. By controlling the congestion window at the receiver, CLAMP maintains a fixed mean queue at the bottleneck that prevents the precious access link from being starved of packets, while minimizing queueing delays. The size of the bottleneck queue with respect to the number of flows can be controlled. Its decentralized architecture makes it ideal for deployment on a range of access networks, including those with a wireless access point, DSL connection, etc.

Some features of CLAMP have been motivated by prior work on AQM, whereby core network routers use probabilistic marking of packets to provide feedback to rate-controlled end nodes [32–34]. A fundamental difference between these and CLAMP is their requirement for internet-wide deployment. In contrast, CLAMP can work effectively in conjunction with any TCP-compatible AQM scheme. However, there will still be a role for CLAMP in a future internet, even with such mechanisms controlling core network congestion. Unlike the above schemes, CLAMP maintains a queue at the bottleneck router, which can be tuned to the requirements of the access network. As will be shown in Section V, such queueing can be useful in handling time varying link capacity. This control is difficult to achieve if the control strategy is rate-based rather than window-based, or if the control is not local to the receiver’s access network.

A traditional cwnd-based AQM scheme tailored for use at a wireless base station [11] has been shown to outperform the popular Random Early Discard AQM. This has the desirable property of needing to be implemented only at a single point. It works by limiting the minimum spacing between packets losses. However, because its signalling is still by dropping packets, it cannot totally avoid TCP’s variations of a factor of two in window size.

Although scaling behaviour is not the topic of this paper, we remark that CLAMP scales in such a way that the queueing delay at the bottleneck router does not tend to zero in the limit of large access link capacity, in contrast to the above AQM schemes (see also [35]). There are actually two scalings we can consider: 1) the channel rate can be scaled up holding the number of nodes fixed, or 2) the channel rate can be scaled up, increasing the number of nodes in the same proportion, and hence holding the rate per node fixed. In either case, using the fluid model in Section VII, we can compute the equilibrium total queue size in terms of the scaling parameters and show that it must itself scale up in proportion with the channel rate, as seen in Section VII. A more intuitive explanation is the following: as the receiver has no knowledge of RTTs, CLAMP cannot translate between data rate and window size. Instead, it relies on the existence of queueing delay at the bottleneck

router to perform its control. However, queueing delay can be made small by appropriate choice of parameters.

Because CLAMP does not require the access router to store information about each connection, implementation also does not suffer from scaling problems.

III. SYSTEM TOPOLOGY AND DESIGN CONSTRAINTS

A. Design Constraints

The objective behind this work is to devise a scheme that will only require modifications to the access network itself, i.e., clients that reside in the access network, and possibly edge routers. This is essential if a scheme is to be deployed on the current internet.

Additionally, the scheme should retain compatibility with the *end-to-end* Internet framework [36]. To scale to high bandwidth access links, with many flows, it should only require aggregate (as opposed to flow-by-flow) feedback from the network, and should be compatible with the existing transport layer protocols. Since TCP does not provide any mechanism to measure RTTs from the receiver, the scheme should not rely on such measurements, as the only practical way to obtain such information would be to generate explicit probe packets.

Finally, the scheme should be distributed, and as simple as possible. Simplicity minimizes the cost of the access router, which is critical, as it is likely to be a commodity device. It must be distributed, since flows may terminate in separate machines, and communication between them would incur an overhead.

B. Topology and Notation

The access network topology of interest is illustrated in Fig. 1. For the sake of illustration we have depicted the access link as a wireless network access point, however it can be any one of a variety of wireless technologies as discussed in the introduction. There are u active users sharing the radio resources. In the example scenarios studied in this paper, the access point maintains a separate queue for each user, to allow channel-state-aware scheduling between users, but users may be involved in multiple TCP sessions which then share the same queue, as is the case for user 1 in the figure. Alternatively, the access point may use only a single queue for all users (not depicted, but relevant for wireline applications).

The service rate of a queue, μ_c , depends on the channel statistics of the users, and the scheduling policy at the wireless access point. Referring to Fig. 1, each flow, i , has a source node, S_i , and an associated round trip transmission delay, which includes all propagation and queueing delays, except for the queueing delay at the access router.

Each sending node implements sliding window flow control. Under the assumption that sources are greedy, the total number of packets and acknowledgements for flow i , in flight at any time, t , is equal to the window size, $w_i(t)$. The CLAMP algorithm selects $w_i(t)$ in a decentralized way, such that each flow sharing the same queue obtains a proportional share of the service rate, μ_c , and the equilibrium buffer occupancy of the queue, $q(t)$, can be controlled as discussed below.

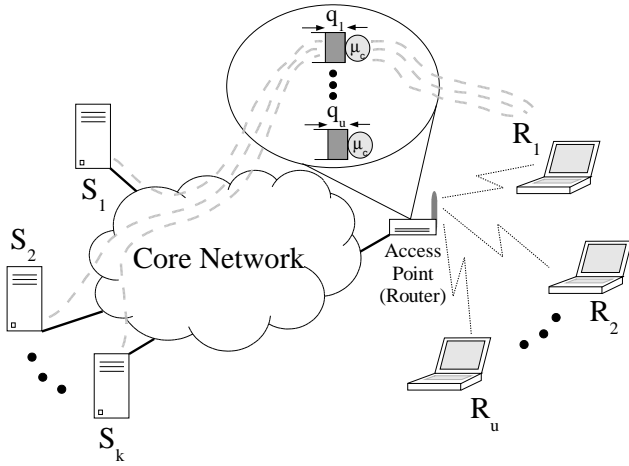


Fig. 1. System topology.

IV. THE CLAMP PROTOCOL

In order to implement CLAMP, the access network should be modified by inserting a software agent in both the access router and the client, as described below. (As an aside, note that it is also possible to implement CLAMP as a PEP, with the associated inability to use IPsec. Indeed, the precursor to CLAMP described in [37] has been implemented as a GPRS PEP, with very promising results [38].)

A. Access Router Agent

The software agent in the access router periodically samples each user's queue length, q , and evaluates a measure of "congestion", $p(q)$: the monotonic increasing function of q given below. The value is then passed to each receiver, either by inserting the value into the IP header of each packet leaving the access router, or by each receiver explicitly requesting the value from the access router as required. (Note that using the IP header rather than the TCP header allows IP security to remain unaffected.)

This paper uses the smooth function

$$p_s(q) = \begin{cases} (bq - a) / \mu_c & \text{if } q > 2a/b \\ b^2 q^2 / 4a\mu_c & \text{otherwise,} \end{cases} \quad (1)$$

where μ_c is the average aggregate rate of the bottleneck link in bytes per second, and the dimensionless constant $b < \pi/2$ determines how sensitive the bottleneck queue size is to the number of flows, as seen in Section VII. The parameters a (in bytes) and b control the equilibrium mean queue size, q^* , as will be seen in Section VII. They can be statically set or dynamically tuned to obtain a desired queueing behavior.

This function is smooth, monotonic increasing and has bounded derivative. Smoothness has been found to increase the fairness of the algorithm, while stability considerations require the bounded derivative, as discussed in Section VII.

B. Client Side Agent

The client agent's function is to receive the router's congestion measure, p , and set the receiver's TCP AWND value

according to the algorithm described below, hence clamping the sender's TCP transmission window.

If IP security is not required, this agent can be implemented as a wrapper for the software driver for the wireless network interface by modifying packets' payloads. Alternatively, using an open-source protocol stack, the price information can be passed up from the IP layer to the TCP layer, and the agent can operate securely at the transport layer.

C. Window Update Algorithm

For simplicity, the algorithm will be described for a single flow. Let t_k denote the time instant when the k th packet, of size s_k bytes, is received by the receiving client. CLAMP calculates the change in window size (in bytes [39]) as

$$\Delta w(t_k) = \left[\frac{\phi\tau - p(q(t_k))\hat{\mu}(t_k)}{\hat{d}(t_k)} \right] (t_k - t_{k-1}) \quad (2)$$

where $\hat{\mu}$ is an estimate of the received rate of the flow in bytes per second, $\hat{d}(t_k)$ is given below, $\phi > 0$ is a positive constant expressing the flow's need for bandwidth, and $\tau > 0$ (bytes) is a constant for all flows. The term in τ tries to increase the window at a constant rate, to balance the term in $\hat{\mu}$ which reduces it at a rate which increases with the occupancy of the queue and with the proportion of traffic due to the particular flow.

The current received rate, $\hat{\mu}$, is estimated using a sliding window averaging function,

$$\hat{\mu}(t_k) = \frac{\alpha \sum_{i=k-\alpha}^k s_i}{t_k - t_{k-\alpha}}, \quad (3)$$

and

$$\hat{d}(t_k) = (1 - \beta)\hat{d}(t_{k-1}) + \beta \frac{\text{AWND}(t_k)}{\hat{\mu}(t_k)}, \quad (4)$$

where the integer α and $\beta \in (0, 1)$ are smoothing factors, and $\text{AWND}(t_k)$ is the actual advertised window (see below). These estimators were chosen for their simplicity, and other estimators may prove to be more effective. In equilibrium, \hat{d} will be the RTT of the flow in seconds, i.e., propagation plus queueing delay. However, computing $\hat{d}(t_k)$ does not require an explicit measurement of such delay, and the algorithm achieves its objectives even if it is not an accurate estimate of the RTT.

Note that for notational ease we have omitted the index of the flow in the above notation. However, each flow will maintain its own $\phi, w, \hat{\mu}$, and \hat{d} .

Preliminary descriptions of CLAMP [37,40] did not contain the the division by $\hat{d}(t_k)$ in (2). This factor improves stability of the algorithm, as demonstrated in Sections V–VII.

Before updating the window size, w , Δw is clipped, giving

$$w(t_k) = \begin{cases} w(t_{k-1}) - s_k & \text{if } \Delta w(t_k) < -s_k \\ w(t_{k-1}) + \Delta w(t_k) & \text{if } -s_k \leq \Delta w(t_k) \leq \bar{\Delta} \\ w(t_{k-1}) + \bar{\Delta} & \text{if } \Delta w(t_k) > \bar{\Delta} \end{cases} \quad (5)$$

The maximum window increase, $\bar{\Delta} > 0$, 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. Limiting the decrease in window

size to the received packet size ensures the right edge of the window is monotonic non-decreasing [39, p. 42].

The AWND value advertised to the sender is then

$$\text{AWND}(t_k) \equiv \min(w(t_k), \text{AWND}), \quad (6)$$

where the AWND on the right hand side of the assignment is the value received by the client side agent from the receiver's operating system. The assigned value on the left hand side is the one to be used in (4).

The flow control algorithm can provide non-uniform sharing of the bottleneck bandwidth by appropriately setting the constants ϕ_i . Section VII shows that, under certain conditions, flow i will obtain the proportion $\phi_i / \sum_{j=1}^k \phi_j$ of the bottleneck capacity, where k is the number of flows sharing the bottleneck.

Finally, it is important to note that the algorithm as stated 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, the simplest solution is to limit the minimum window size to a constant, w_{\min} . The simulations presented here take $w_{\min} = 1$.

D. Slow Start

CLAMP's update rule uses $p(q)$ as feedback about the suitability of the current window size, $w(\cdot)$. When the sender's CWND is less than $w(\cdot)$, this feedback path is broken, and $w(\cdot)$ may grow too large. For this reason, it is useful for CLAMP to mimic TCP's *slow start* algorithm [41]. Briefly, TCP starts with a window size of one, and during its "slow start" phase, it increases its window by one segment for each acknowledgment received, doubling the window every RTT. Slow start terminates when buffer overflow causes a packet loss, and the window size is halved.

To track this, a new CLAMP connection starts with $w(0) = MSS$, where MSS is the connection's maximum segment size, and for each packet received, it sets

$$w(t_k) \leftarrow w(t_k) + MSS,$$

instead of using (5). CLAMP stops tracking slow start when congestion is detected, i.e., $\Delta w(t_k) < 0$ in (2) or three duplicate acknowledgments are received, indicating packet loss. The receiver then sets $w(t_{k+1}) \leftarrow w(t_k)/2 \approx w(t_k - \text{RTT})$. The reason for this is that the observations at time t_k which indicate congestions reflect the AWND size one round trip time earlier.

Although buffer dimensioning is beyond the scope of this paper, note that if the bottleneck buffer size is large enough relative to the equilibrium queue size then the condition $\Delta w(t_k) < 0$ will be true before a buffer overflow occurs, and the CLAMP will terminate slow start without causing a buffer overflow (see Section V). By taking advantage of the additional control information from the access router, CLAMP receiver side slow start tames the behavior of TCP sender side slow start.

V. PERFORMANCE EVALUATION

The aim of this section is to evaluate if CLAMP does in fact enhance access link throughput and provide differentiated capacity allocation. In order to ascertain whether these design goals are achieved, the network topology shown in Fig. 3 is simulated under various conditions.

Referring to Fig. 3, sources S_1, S_2, \dots, S_k represent servers which are TCP rate controlled, and the links from S_i to X model entire paths through the internet. Node X is the wireless access router which contains the CLAMP router software agent. It monitors the queue size, computes $p(q)$ and inserts the value into an IP header option field, using an additional 32 bits. The wireless access router transmits to the TCP/CLAMP clients, R_1 – R_4 , over a common broadcast radio channel. Clients R_3 and R_4 are both running in the same physical device and so have a common queue at router X , and common radio propagation conditions, as described below. On completion of each successful packet transmission, the base station selects the next user with whom it should transmit to be the user with the highest SNR. The base station implements adaptive coding and modulation, and stop and wait ARQ retransmissions. If all queues are permanently full (so that the scheduler can pick the best of three channels), the peak channel throughput is slightly more than 1 Mbps, with precise value depending on the packet loss rate considered. There is a limit on the number retransmission attempts, and hence a non-zero probability of packet error. All links are bidirectional, with characteristics as shown in Table II. Other simulation parameters are given in Table III.

To describe the details of the adaptive coding and modulation used in our simulation, we first make the point that there are many possible coding and modulation, link layer adaptation, and MAC layer scheduling standards to choose from. It is not the purpose of the present paper to undertake a detailed study of TCP over any particular current standard; our objective here is to capture key features of modern wireless link layers in a simulation that is fast and efficient for our purposes. The key features are adaptive, variable rate transmissions, and very small, but non-negligible, packet loss probabilities.

The path loss fading process between the base station and each receiver is a stationary, stochastic process with marginal statistics given by the Rayleigh distribution; thus the SNR has exponential statistics. Jakes' model is used with $G = 8$ generators and a Doppler frequency of 0.6 Hz (1.8 GHz carrier, 0.1 m/s mobile speed). All users' processes are independently and identically distributed. Frame transmissions over the physical channel occur in slots of $\tau_{\text{slot}} = 3$ ms; a slot is the time period of one frame transmission attempt. The process is sampled on a slot by slot basis; the SNR is deemed to be constant over a slot interval. Rate matching is used, and in slot s , $B(s)$, bits are transmitted:

$$B(s) = \tau_{\text{slot}} \min(W_0 \log(1 + N(s)), C_{\max}),$$

where C_{\max} is a parameter representing the maximum possible rate that can be transmitted over the channel, $N(s)$ denotes the SNR over the frame duration, and W_0 is a constant to

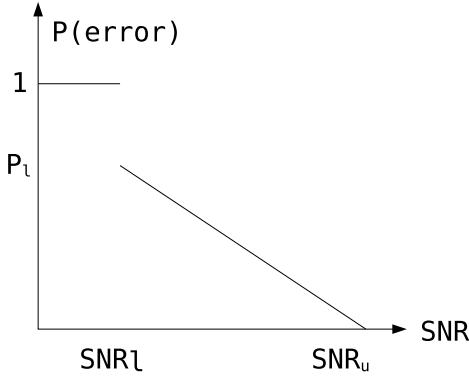


Fig. 2. Frame error rate as a function of SNR.

account for bandwidth; the assumption is that the rate will vary logarithmically with the SNR up to some maximum value.

Given that slot errors can be due to many factors beyond the scope of the present paper (Doppler spread, rate of channel state feedback, measurement error, available rates, burstiness of the bit error process) and that the optimal point on the tradeoff between rate and loss probability may depend on complex cross-layer effects, we have chosen a simple, if somewhat arbitrary, way of assigning loss probability to frames: we assume that the conditional probability of frame error, conditioned on the SNR, is a piecewise linear, decreasing, function of the SNR , as depicted in Figure 2. The term SNR_l is the lowest tolerated SNR: a frame is certainly corrupted if the SNR falls below SNR_l . We denote the frame error rate at SNR_l , the limiting frame error rate as we approach SNR_l from the right, by P_l . The SNR can exceed SNR_u , but this is the value corresponding to the maximum channel rate C_{max} . We assume that the frame error rate is zero for any SNR exceeding SNR_u . Between SNR_l and SNR_u , we assume that the frame error probability is a linear, decreasing function.

If a frame is received in error, then the base station retransmits it exactly as it was before, with no recoding. We assume that the next attempt is independent of the first, but the same probability of error is used as was computed for the first attempt; i.e., we assume that the channel hasn't changed from the preceding attempt. A maximum of $R = 3$ slot transmissions are allowed. If after R attempts the frame is still received in error, then the erroneous frame is delivered as is to higher layers, ultimately resulting in a packet error. A packet will typically contain more bits than can be transmitted in a single slot interval. Hence, a packet transmission will span a variable number of slot intervals. The number of slot intervals is random, depending on the realizations of the channel processes, and chosen parameters.

At the completion of a packet transmission, the base station selects the user with the best channel quality that has at least one packet queued at the base station pending transmission. The scheduling decision is only made once the previous packet transmission has completed, either successfully or unsuccessfully.

The above model contains many parameters and both packet loss probability, and packet throughput, are functions of these

TABLE I
ERROR MODEL PARAMETERS

| G | Doppler (Hz) | C_{max} (Mbps) | R | SNR_l | SNR_u | P_l | Packet loss rate |
|-----|--------------|------------------|-----|---------|---------|-------|------------------|
| 8 | 0.6 | 1.44 | 3 | 0.004 | 4 | 0.44 | 0.001 |
| | | | | | | 0.8 | 0.01 |

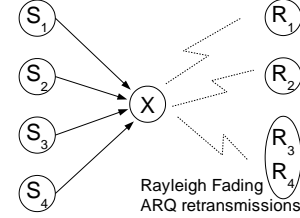


Fig. 3. Simulation network topology

parameters. The experiments used the parameters in Table I, which corresponds to data rates up to 1.44 Mbps at SNR_u . When the probability of error reaches 1, at SNR_l , the data rate is 3.57 kbps. Because the channel rate, μ_c , was not constant, the value used in calculating $p(q)$ from (1) was the mean value over the preceding 100 packets. In the experiments below we varied the parameter p_l to achieve average packet loss rates of 0.01 and 0.001.

To demonstrate how CLAMP can be used to control the tradeoff between queuing delay and radio link utilisation, we ran several simulation based on the topology depicted in Figure 3. Experiments were run for two sets of channel statistics, giving packet loss rates of 0.01 and 0.001, with a wide range of values for parameter a for a corresponding range of mean queue sizes. The resulting aggregate throughput of node X (excluding retransmissions) is plotted against the mean queue size in Figure 4. For comparison, the performance of TCP is also plotted, varying the buffer size to adjust the mean queue size. When the packet loss rate is 0.001, a mean queue size of 5000 bytes (a delay of around 40 ms) gives a performance improvement due to CLAMP of around 10%, while for 1000 bytes (10 ms) it is almost 30%.

Corresponding results for the average time to download

TABLE II
LINK CONFIGURATION

| Node 1 | Node 2 | Capacity | Delay | Pk loss rate |
|--------|--------|----------|---------|--------------------|
| S_i | X | 10 Mb/s | $d_i/2$ | 0 |
| X | R_i | see text | 1 msec | $10^{-2}, 10^{-3}$ |

TABLE III
SIMULATION PARAMETERS

| Parameter | Value |
|----------------------|--------------|
| TCP Packet Size | 500 Bytes |
| CLAMP τ | 5000 Bytes/s |
| CLAMP $\bar{\Delta}$ | 10000 Bytes |
| CLAMP b | 1 |
| CLAMP a | variable |
| d_1 | 0.1345 s |
| d_2 | 0.041 s |
| d_3 | 0.2843 s |
| d_4 | 0.002525 s |

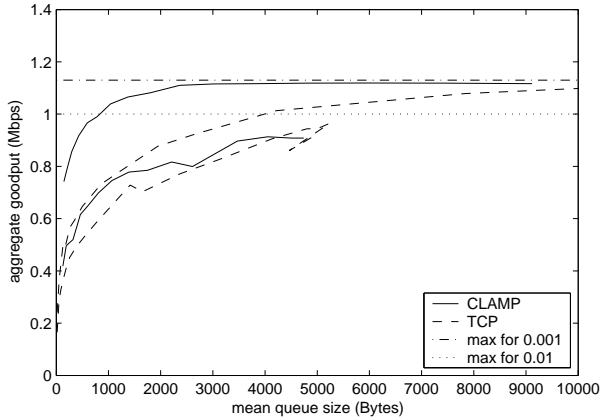


Fig. 4. Goodput/Latency tradeoff for packet loss rates of 0.001 (upper curves) and 0.01 (lower curves). A queue size of 5000 bytes introduces a delay of approximately 40 ms.

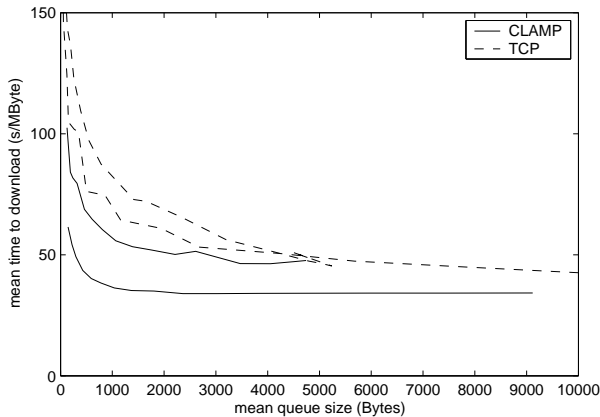


Fig. 5. Time/Latency tradeoff for packet loss rates of 0.001 (lower curves) and 0.01 (upper curves). A queue size of 5000 bytes introduces a delay of approximately 40 ms.

1 MByte of data are presented in Figure 5. With a packet loss rate of 0.001, these results indicate an improvement of 35% by using CLAMP at 30 ms delay. The reason for a larger gain is that these results correspond to the harmonic mean of the bandwidths of the sources, rather than their arithmetic mean, thus giving greater weight to the slowest connections. CLAMP’s improved fairness has a greater impact on the user with the longest round trip time, and hence worst performance under TCP. This causes the significant reduction in mean download time.

Returning to Figure 4, an important point to emphasize is the benefit to utilization provided by CLAMP at low latencies. It is well known that most TCP flows are small (“mice” [42]) in which case CLAMP does not directly have an impact on window evolution for these flows, since they terminate in slow-start. However, it is also well known that most packets belong to large TCP flows (“elephants” [42]; see [43], [44] for details) so it is reasonable to consider the scenario in which there is one large flow, sharing the access queue with bursty traffic from small flows. Figure 4 indicates that the throughput of the large flow need not be compromised significantly by the requirement that the latency for the mice flows be small.

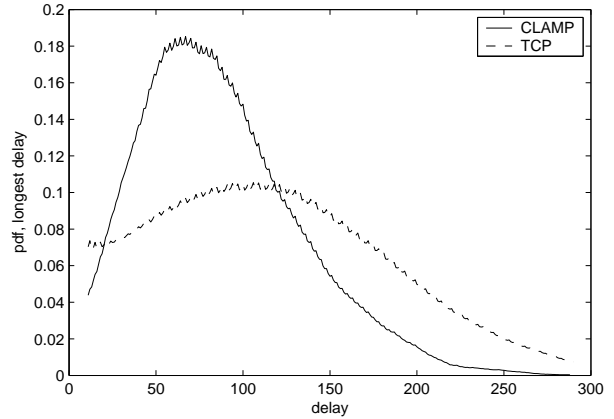


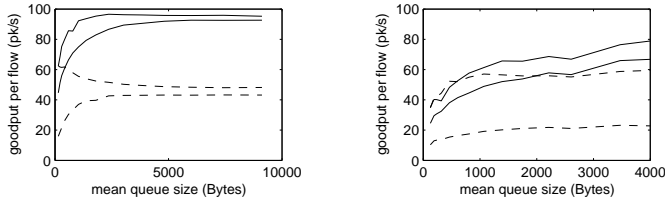
Fig. 6. Probability density of delay for flow 1, under CLAMP and TCP, with a mean of 85 ms for each.

The same is true if the small flows belong to real-time services, such as voice. While we have not provided an explicit experiment, we assume that the small flows will have negligible impact on the elephants. It is beyond the scope of the present paper to provide a full treatment of the impact of delay sensitive flows (particularly non-responsive non-TCP flows) on flows controlled by CLAMP. If the above assumption is violated, further mechanisms at the access router may be required to protect the TCP flows from the non-responsive flows. This is true for TCP with or without CLAMP.

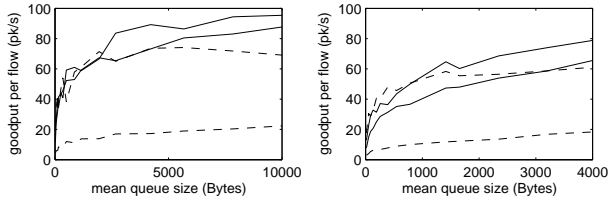
For a given mean delay, real time services fare better under CLAMP than TCP, due to the reduced jitter. This is seen in Figure 6, which shows the pdf of the delay at the queue for flow 1, which is the source with the largest jitter. (In general, the jitter under TCP is larger for flows with larger round trip times, as the halving of the window causes a greater change in queueing; however, since flow 3 shares a queue with flow 4, which is very responsive due to its small RTT, it has a smaller jitter than flow 1.) Figure 6 shows that, by reducing the jitter, CLAMP reduces the probability that the queueing delay will exceed an acceptable maximum delay, such as 120 ms. The results for other flows are qualitatively similar, although less pronounced.

CLAMP’s fairness is shown explicitly in Figure 7, which shows the throughput of each flow against the mean queueing delay (averaged over all flows). For low packet loss rates, flows 3 and 4 share their common queue fairly under CLAMP, despite a difference in round trip time of two orders of magnitude. Under TCP, these flows have very unequal throughputs, except when the queueing delay swamps the propagation delay giving them more equal round trip times (not depicted). When the packet loss rate is higher, CLAMP is no longer entirely fair, but still more so than TCP alone.

Note that the curves for packet loss rate 0.01 terminate at a queue size of around 6000 bytes. This is the limit imposed by TCP halving its window on packet loss; further increases in buffer size do not significantly improve utilization beyond this point. This is illustrated in Fig. 8, which shows the mean AWND and CWND sizes for a system running CLAMP for 8000 seconds with varying levels of packet loss. CLAMP



(a) CLAMP, $p_e = 0.001$ (b) CLAMP, $p_e = 0.01$



(c) TCP, $p_e = 0.001$ (d) TCP, $p_e = 0.01$

Fig. 7. Throughput per flow for packet loss rates of $p_e = 0.001$ and $p_e = 0.01$. Dashed lines are for flows on the same host.

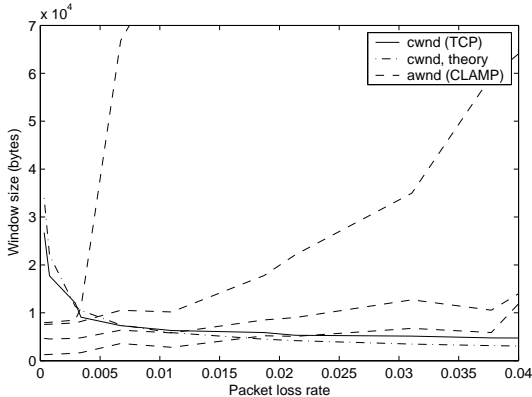


Fig. 8. Window sizes for varying packet loss rates

prevents buffer overflow, and so the CWND size is determined purely by wireless packet loss. Since the window size increases by $1/\text{CWND}$ per packet, and the probability p_e decreases by $\text{CWND}/2$, the equilibrium CWND size is approximately $\sqrt{2/p_e}$ packets, where p_e is the packet loss probability. More accurate analysis [45] yields $\sqrt{3/2p_e}$, shown in Fig. 8 as “cwnd, theory”, which is a very close match to the observed value. In contrast, the AWND size is not determined externally and depends on the a and b parameters. When CWND is less than AWND, CLAMP no longer senses the change in rate caused by increasing AWND, which causes AWND to increase without bound. In this scenario, CLAMP becomes ineffective (but harmless), and performance improvement requires more link layer retransmissions (or techniques such as PEPs). The important conclusion to draw from Fig. 8 is that for packet loss rates less than 0.01, AWND is smaller than CWND, and hence CLAMP is in charge of flow control.

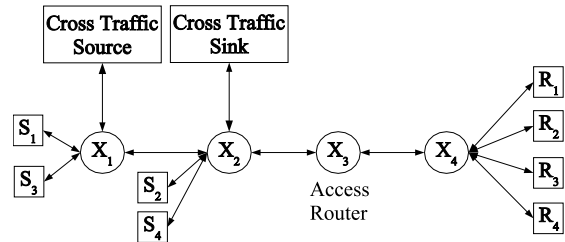


Fig. 9. Simulated topology that includes cross traffic

VI. CONGESTION IN THE CORE NETWORK

A key strength of CLAMP is that it can be implemented in a TCP network by changing only the router at the access links and the receivers served by that access link. Section V showed that the algorithm does produce the desired results when the access link is the bottleneck. In this section will further investigate the interaction between CLAMP and TCP by investigating a scenario where another bottleneck arises in the core network. This is achieved by starting a cross traffic TCP session within the core. As in the case with packet loss, this will result in TCP’s CWND limiting CLAMP’s performance.

The network topology that was simulated is illustrated in Fig. 9. $S_i, i = 1, 2, 3, 4$ 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_4 a LAN switch. All link capacities and delays are listed in Table IV and other simulation parameters are listed in Table III.

The cross traffic source is another TCP source fed by a greedy traffic source, which was started and stopped at 100 and 200 seconds respectively. Fig. 10 is plot of the sequence numbers of the received acknowledgments versus time.

The parallel lines before the 100 second mark indicates that all four flows are receiving 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.

VII. A FLUID MODEL FOR CONVERGENCE ANALYSIS

With any feedback control system, the issue of stability must be addressed. The following sections define and analyse a sim-

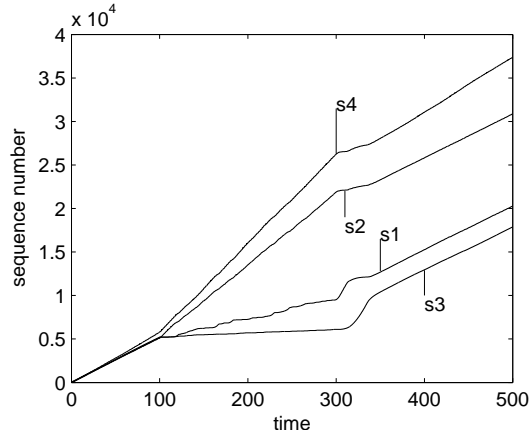


Fig. 10. Simulated topology that includes cross traffic

TABLE IV
LINK CONFIGURATION

| Link | Capacity | Delay |
|------------------------------|----------|---------|
| $S_i - X_j$ | 10 Mb/s | $d_i/2$ |
| $X_1 - X_2$ | 10 Mb/s | 1 msec |
| $X_2 - X_3$ | 10 Mb/s | 1 msec |
| $X_3 - X_4$ | 1.5 Mb/s | 1 msec |
| $X_4 - R_i$ | 10 Mb/s | 1 msec |
| Cross Traffic Source - X_1 | 10 Mb/s | 5 msec |
| X_2 - Cross Traffic Sink | 10 Mb/s | 5 msec |

plified model of CLAMP in order to establish the conditions under which it converges, and the nature of the equilibrium. There are two versions of the CLAMP algorithm: the one described in this report, and the one of [37], [40]. A general model will be derived which describes both algorithms. Some general results which apply to both versions will first be derived, followed by some limited results for the current version and then more comprehensive results for the simpler version.

Note that CLAMP cannot cause the system to become *unbounded*, because the actual transmission rate cannot exceed that specified by TCP's congestion control algorithm. Even if the maximum AWND size were not bounded by (6), the actual transmit window would be bounded by TCP's congestion window (CWND). However, the throughput will be maximised when the system does converge.

A. Fluid model

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

$$q(t) = \sum_{i=1}^k B_i(t). \quad (7)$$

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

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 packets that left the queue in the interval $(t, t-d_i]$ plus or minus packets added or removed due to changes in window size during that interval. Hence,

$$w_i(t) = B_i(t) + \int_{t-d_i}^t \left(\frac{B_i(s)}{q(s)} \mu_c + \frac{dw_i}{dt} \Big|_{t=s} \right) ds. \quad (8)$$

Differentiating (8) gives the rate of change of buffering at the bottleneck queue attributed to flow i :

$$\frac{dB_i}{dt} = \frac{dw_i}{dt} \Big|_{t-d_i} + \frac{B_i(t-d_i)}{q(t-d_i)} \mu_c - \frac{B_i(t)}{q(t)} \mu_c. \quad (9a)$$

Under the flow control algorithm (5), a fluid model of the window size evolution is

$$\frac{dw_i}{dt} = \begin{cases} -\frac{B_i(t)}{q(t)} \mu_c & \text{if } g_i(t) \leq -\frac{B_i(t)}{q(t)} \mu_c \\ g_i(t) & \text{if } -\mu_c < \frac{q(t)}{B_i(t)} g_i(t) \leq \mu_c \bar{\Delta} \\ \frac{B_i(t)}{q(t)} \mu_c \bar{\Delta} + \epsilon & \text{otherwise} \end{cases} \quad (9b)$$

where ϵ is a small constant, and $g_i(t)$ has one of two forms:

$$g_i(t) = \frac{1}{d_i} \left[\phi_i \tau - p(q(t)) \frac{B_i(t)}{q(t)} \mu_c \right] \quad (10a)$$

for the algorithm described in this report, and

$$g_i(t) = \phi_i \tau - p(q(t)) \frac{B_i(t)}{q(t)} \mu_c \quad (10b)$$

for the older version of clamp described in [37], [40]. Note that the two algorithms have the same equilibrium, and it is only the transient behaviour which is changed.

The term $(B_i(t)/q(t))\mu_c\bar{\Delta}$ reflects the maximum increment to the window size, $\bar{\Delta}$, 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 $(-B_i(t)/q(t))\mu_c$, in the case when $g_i(t) \leq -\mu_c B_i(t)/q(t)$, 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 occurs in 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.

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, define $0/0 \equiv 0$ in the last terms of (9a). It is easy to see that when the queue empties, the first two terms of (9a) 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 (9a) over i ,

$$\frac{dq}{dt} = \sum_{i=1}^k \frac{dw_i}{dt} \Big|_{t-d_i} + \sum_{i=1}^k \frac{B_i(t-d_i)}{q(t-d_i)} \mu_c - \rho(t) \mu_c, \quad (11)$$

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

Lemma 1: If $0 \leq B_i(t) \leq w_i(t)$ for all $t \leq 0$ and all $i = 1, \dots, k$, then under (9),

$$0 \leq B_i(t) \leq w_i(t) \quad (12)$$

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

Proof: Assume first that $B_i(t) \geq 0$ for all i and t , from which it follows that $q(t) \geq 0$. Then (9b) implies $dw_i/dt \geq -\mu_c B_i(t)/q(t)$, and the integrand in (8) is always non-negative, giving the second inequality.

To see that $B_i(t) \geq 0$ for $t \geq 0$, note that this is trivially true during the periods when the queue is empty. For during these periods, the last term of (9a) 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 (9a) is zero. But the sum of the first two terms is non-negative, 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

$$T = \tau \sum_{j=1}^k \phi_j \quad (13)$$

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 by, for all i ,

$$B_i^* = \frac{\phi_i}{\sum_{j=1}^k \phi_j} p^{-1} \left(\frac{T}{\mu_c} \right). \quad (14)$$

Proof: Substituting (14) into (9) 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 (9b) (noting that $\epsilon > 0$), whence for all i , (10a) implies

$$B_i^+ = \frac{\phi_i \tau q^+}{\mu_c p(q^+)}. \quad (15)$$

Summing (15) over i yields

$$q^+ = p^{-1} \left(\frac{T}{\mu_c} \right). \quad (16)$$

Substitution of (16) into (15) shows that $B_i^+ = B_i^*$, establishing uniqueness. ■

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

B. Stability of new CLAMP algorithm

Unlike its predecessor, the current version of CLAMP divides the total change in window size by d_i , which approximates the round trip time in equilibrium. As will be shown in this section, this allows parameters to be set such that the system is stable for arbitrary round trip times. However, it complicates the analysis considerably, and results are only known in the special case of equal propagation delays.

Letting $d_i \equiv d$ for all i , and ignoring the boundary conditions in (9b), equation (11) takes the simple, linear form

$$\frac{dq}{dt} = \frac{1}{d} [T + a - bq(t-d)] \quad (17)$$

where $d_i = d$ for all i is assumed.

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 the necessary and sufficient condition for the total queue size, $q(t)$, to converge under (17) is:

$$b < \frac{\pi}{2}. \quad (18)$$

Proof: Summing (11) over i gives the linear equation:

$$\frac{dq}{dt} = \frac{T+a}{d} - \frac{b}{d} q(t-d), \quad (19)$$

which can be analyzed by standard techniques [46]. Taking the unilateral Laplace transform gives

$$Q(s) = \frac{q(0^-)}{s + (b/d)e^{-ds}}. \quad (20)$$

The (infinite number of) poles of (20) determine the limiting behaviour of (19). For $d = 0$ the system has a single real pole located at $s = -\infty$, 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 [46, p 26], we search for potential points where the poles cross the imaginary axis by solving:

$$W(\omega^2) = \omega^2 - (b/d)^2 = 0,$$

for ω^2 . The solution $\omega^2 = (b/d)^2$ indicates that the poles cross the imaginary at $s = \pm j(b/d)$. Equating the characteristic equation of (20) to zero and substituting in $\omega = b/d$ reveals that the first time two poles cross into the positive half of the

imaginary axis occurs when $d = (\pi d)/(2b)$, i.e., the system will be stable for $b < \pi/2$. ■

If $q(t)$ converges to a constant, the equations (9) decouple, and each is stable if and only if (18) holds. Thus (18) is a necessary and sufficient condition for the asymptotic stability of (9) for the case of equal delays.

C. Region of stability of old CLAMP

The simpler form of the original CLAMP algorithm allows more comprehensive results to be obtained. This will be achieved by studying the linearisation around the equilibrium point.

The equilibrium solution of (9) can be shifted to the origin by substituting $B_i(t) \rightarrow B_i^* + x_i(t)$. Noting that $dx_i/dt = dB_i/dt$, and neglecting the clipping of dB_i/dt , (9a) becomes

$$\begin{aligned} \frac{dx_i}{dt} &= \phi_i \tau - \left(p \left(q^* + \sum_{j=1}^k x_j(t - d_i) \right) - 1 \right) \\ &\quad \times \frac{B_i^* + x_i(t - d_i)}{q^* + \sum_{j=1}^k x_j(t - d_i)} \mu_c \\ &\quad - \frac{B_i^* + x_i(t)}{q^* + \sum_{j=1}^k x_j(t)} \mu_c. \end{aligned} \quad (21)$$

The system (21) is not amenable to analysis, but insight into its dynamics can be obtained from its linearisation. Using $1/(1+x) = 1 - x + o(x)$, and $f(x+\Delta)f(x) + f'(x)\Delta + o(x)$, (21) becomes

$$\begin{aligned} \frac{1}{\mu_c} \frac{dx_i}{dt} &= \sum_{j=1}^k \theta_i \left(x_j + \left(p'(q^*) - \frac{C}{q^*} \right) x_j(t - d_i) \right) \\ &\quad - \frac{1}{q^*} x_i(t) - \left(\frac{C}{q^*} \right) x_i(t - d_i) \end{aligned} \quad (22)$$

where $\theta_i = \phi_i / \sum_{j=1}^k \phi_j$ and $C = p(q^*) - 1$.

The stability of the linear system (22) is necessary for the asymptotic stability of (21) and hence (9). The stability region is easily characterised when $p(q^*) \approx 1$, as then $C \ll p'(q^*)q^*$, and the terms at $t - d_i$ are dominated by the first sum. When $C = 0$, (22) becomes

$$\frac{q^*}{\mu_c} \frac{dx_i}{dt} = \theta_i \sum_{j=1}^k (x_j(t) - p'(q^*)q^* x_j(t - d_i)) - x_i(t). \quad (23)$$

Proposition 1: If $b < \pi/(2d_i)$ for all i , then $x(t)$ is stable under (23).

This is a corollary of the following two lemmas, using the continuity with respect to b of solutions.

Lemma 3: A necessary and sufficient condition for (23) to be stable is that there be no solution, s , of

$$s + b \sum_{i=1}^k \theta_i e^{-sd_i} = 0 \quad (24)$$

with non-negative real part.

Proof: Let E be the $k \times k$ matrix of all '1's and $r = p'(q^*)q^* = bq^*/\mu_c$. Writing (23) in matrix form, and taking the Laplace transform gives the stability requirement

$$\det \left(\left(\frac{rs}{b} + 1 \right) I - \text{diag}(\theta_i(1 - re^{-sd_i}))E \right) \neq 0 \quad (25)$$

for any s with non-negative real part, where I is the identity. Equivalently, $(1 + rs/b)$ must not be an eigenvalue of $\text{diag}(\theta_i(1 - re^{-sd_i}))E$. However, this matrix has rank 1, and its only eigenvalues are 0 and $\sum_{i=1}^k \theta_i(1 - re^{-sd_i})$. The result follows from the fact that $\sum_{i=1}^k \theta_i = 1$. ■

Lemma 4: No solution of (24) can lie on the imaginary axis if $b < \pi/(2d_i)$ for all i .

Proof: Assume, with a view to obtaining a contradiction, that there is a solution at $s = j\omega$, giving

$$j\omega + b \sum_{i=1}^k \theta_i e^{-j\omega d_i} = 0.$$

Writing $\omega' = \omega/b$, and equating real and imaginary parts,

$$\sum_{i=1}^k \theta_i \cos(d_i b \omega') = 0 \quad (26a)$$

$$\sum_{i=1}^k \theta_i \sin(d_i b \omega') = \omega'. \quad (26b)$$

Since $\sum_{i=1}^k \theta_i = 1$ and $\theta_i > 0$, (26b) implies that ω' is the weighted mean of samples of the sine function, and hence $-1 \leq \omega' \leq 1$. However, by hypothesis $0 \leq d_i b < \pi/2$. Thus

$$-\frac{\pi}{2} < d_i b \omega' < \frac{\pi}{2}$$

whence $\cos(d_i b \omega') > 0$ for all i . This contradicts (26a), and the lemma is proved. ■

D. Two flows

In the special case of two flows with $\phi_1 = \phi_2$ and $p(q^*) = 1$, the stability region of (23) can be completely characterised. The next section will show that this stability region seems to play a role in characterising the stability region of the fluid model (9) even when $p(q^*) \neq 1$.

Consider the case of $k = 2$ users, with delays d_1, d_2 . Denote $\text{sinc}(x) = \sin(\pi x)/(\pi x)$.

Proposition 2: When $k = 2$, $\phi_1 = \phi_2$ and $p(q^*) = 1$, a necessary and sufficient condition for (23) to be stable is

$$b < \frac{1}{d_1 \text{sinc}(d_1/(d_1 + d_2))}. \quad (27)$$

Proof: For $0 < h < 1$, define $d_i(h) = d_i h$, for $i = 1, 2$. To show sufficiency, suppose instead that (27) holds and there is a solution of (24) in the right half plane. By continuity of the solutions with respect to h , there exists a value of h , denoted by $h' < 1$, which gives a solution on the imaginary axis, for the system with delays replaced by the $d_i(h')$ values. Then there exists an $\omega_0 > 0$, such that

$$j\omega_0 + \frac{b}{2} e^{-j\omega_0 d_1 h'} + \frac{b}{2} e^{-j\omega_0 d_2 h'} = 0$$

Since $\cos(x) = -\cos(y)$ only for $y = (2n+1)\pi \pm x$, equating real parts gives

$$\omega_0(d_1 \mp d_2)h' = (2n+1)\pi. \quad (28)$$

Equating real parts, and using (28), gives

$$\omega_0 = (b/2)(\sin(\omega_0 d_1 h') + \sin((2n+1)\pi \pm \omega_0 d_1 h')).$$

Since $\omega_0 \neq 0$, and by the symmetry of $\sin(\cdot)$,

$$\omega_0 = b \sin(\omega_0 d_1 h') = b \sin((2n+1)\pi - \omega_0 d_1 h') \quad (29)$$

$$= b \sin\left(\frac{(2n+1)\pi d_1}{d_1 + d_2}\right). \quad (30)$$

The smallest value of h' corresponds to $n = 0$, and substituting (30) into $\omega_0(d_1 + d_2)h' = \pi$ gives

$$h' = \frac{1}{bd_1 \operatorname{sinc}(d_1/(d_1 + d_2))}.$$

But by assumption the right hand side is greater than unity, which is a contradiction. Thus (27) is sufficient.

To show necessity, it suffices to show that solutions of (24) only ever cross from the left half plane to the right as h increases [46]. Treating (24) as an implicit definition of $s(h)$, it suffices that $\operatorname{Re}(ds/dh) > 0$ whenever $s = j\omega_0$ and $h = h'$. Differentiating (24) gives

$$\frac{1}{ds/dh} = \left(\frac{sb}{k} \sum_{i=1}^k d_i e^{-sd_i h'} \right)^{-1} - \frac{h'}{s}.$$

Since the final term is purely imaginary, and $\operatorname{Re}(x) > 0 \Leftrightarrow \operatorname{Re}(1/x) > 0$, it is sufficient to show that

$$\omega_0 \sum_{i=1}^k d_i \sin(\omega_0 d_i h') > 0.$$

By (29), ω_0 has the same sign as $\sin(\omega_0 d_i h')$ whenever $s = j\omega_0$, and the proposition follows. ■

Note that as $d_1/d_2 \rightarrow 0$ the condition (27) becomes $b_0 < 1/d_1$, and the stability is dominated by the shortest delay, rather than the longest delay, which seems somewhat surprising.

As expected, the bound in (27) lies between the bounds obtained for the systems $d'_1 = d'_2 = \min(d_1, d_2)$ and $d''_1 = d''_2 = \max(d_1, d_2)$.

These analytical results are verified by the numerical results obtained in [37], [40].

VIII. CONCLUSIONS

A system called CLAMP is presented for improving the performance of TCP over low bandwidth, variable rate access networks. It has been shown, by both analysis and simulation, that CLAMP provides differentiated proportional allocation of the capacity of a bottleneck link and simultaneously improves throughput. The algorithm fits into the Internet end-to-end framework, requiring only modifications to the access network, and the mobile users. In particular, only aggregate congestion information from the access router is required by the users. It works in conjunction with current TCP sender implementations and AQM routers in the core network.

Detailed packet level simulations of CLAMP running over test network topologies with multiple flows, and fading wireless channels, were performed. The results of the simulations indicate that total throughput, and individual file download times, can be substantially improved whilst simultaneously decreasing latency for all the flows. CLAMP parameters can be set to obtain service differentiation (more detailed

investigations can be found in [40]) and a fluid model is derived to support this claim, and which enables an analysis of stability. Future work will focus on how to tune the algorithm's parameters to achieve a desired operating point in terms of throughput and latency.

REFERENCES

- [1] E. Biglieri, J. Proakis, and S. Shamai, "Fading channels: information-theoretic and communications aspects," *IEEE Transactions on Information Theory*, vol. 44, pp. 2619–2692, October 1998.
- [2] S. Nanda, K. Balachandran, and S. Kumar, "Adaptation techniques in wireless packet data services," *IEEE Communications Magazine*, vol. 38, pp. 54–64, Jan. 2000.
- [3] H. Balakrishnan, V. N. Padmanabhan, S. Seshan, and R. H. Katz, "A comparison of mechanisms for improving TCP performance over wireless links," *IEEE/ACM Trans. Networking*, pp. 756–769, 1997.
- [4] H. M. Chaskar, T. Lakshman, and U. Madhow, "TCP over wireless with link level error control: Analysis and design methodology," *IEEE/ACM Trans. Networking*, vol. 7, pp. 605–615, Oct. 1999.
- [5] D. A. Eckhardt and P. Steenkiste, "Improving wireless LAN performance via adaptive local error control," in *Proc. IEEE Int. Conf. Netw. Protocols (ICNP)*, pp. 327–338, Mar. 2003.
- [6] R. Ludwig, A. Konrad, A. Joseph, and R. Katz, "Optimizing the end-to-end performance of reliable flows over wireless links," *Kluwer/ACM Wireless Netw. J.*, vol. 8, pp. 289–299, Mar.-May 2002.
- [7] M. Meyer, "TCP performance over GPRS," in *Proc. IEEE Wireless Commun. Netw. Conf. (WCNC)*, pp. 1248–1252, Mar. 2003.
- [8] Z. Kostic, X. Qiu, and L. F. Chang, "Interactions between TCP and RLP protocols in a cellular system," *IEEE Vehic. Technol. Conf.*, vol. 3, pp. 2244–2248, May 2001.
- [9] M. Malkowski and S. Heier, "Interaction between UMTS MAC scheduling and TCP flow control mechanisms," in *Proc. International Conference on Communication Technology*, pp. 1373–1376, 2003.
- [10] R. G. Mukhtar, S. V. Hanly, and L. L. H. Andrew, "Efficient internet traffic delivery over wireless networks," *IEEE Commun. Mag.*, vol. 41, pp. 46–53, Dec. 2004.
- [11] M. Sagfors, R. Ludwig, M. Meyer, and J. Peisa, "Queue management for TCP traffic over 3G links," in *Proc. IEEE Wireless Commun. Netw. Conf. (WCNC)*, pp. 1663–1668, Mar. 2003.
- [12] A. J. Goldsmith and P. P. Varaiya, "Capacity of fading channels with channel side information," *IEEE Trans. Inform. Theory*, vol. 43, pp. 1986–1992, Nov. 1997.
- [13] P. Bender, P. Black, M. Grob, R. Padovani, N. Sindhushyana, and A. Viterbi, "CDMA/HDR: a bandwidth efficient high speed wireless data service for nomadic users," *IEEE Communications Magazine*, vol. 38, pp. 70–77, July 2000.
- [14] P. Viswanath, D. Tse, and R. Laroia, "Opportunistic beamforming using dumb antennas," *IEEE Transactions on Information Theory*, vol. 48, pp. 1277–1294, June 2002.
- [15] D. Cygan and E. Offer, "Short linear incremental redundancy codes having optimal weight structure profile," *IEEE Trans. Inform. Theory*, vol. 37, pp. 192–195, Jan. 1991.
- [16] C. Casetti, M. Gerla, S. Mascolo, M. Y. Sanadidi, and R. Wang, "TCP Westwood: Bandwidth estimation for enhanced transport over wireless links," in *Proc. ACM MOBICOM*, pp. 287–297, 2001.
- [17] N. Samaraweera, "Non-congestion packet loss detection for TCP error recovery using wireless link," *IEE Proceedings Communications*, vol. 146, no. 4, pp. 222–230, 1999.
- [18] J. Border, "Performance enhancing proxies intended to mitigate link-related degradations," RFC 3135, IETF, 2001.
- [19] H. Balakrishnan, S. Seshan, and R. H. Katz, "Improving reliable transport and handoff performance in cellular wireless networks," *ACM Wireless Networks*, vol. 1, no. 4, pp. 469–481, 1995.
- [20] L. Kalampoukas, A. Varma, and K. K. Ramakrishnan, "Explicit window adaption: A method to enhance TCP performance," *IEEE/ACM Trans. Networking*, vol. 10, pp. 338–350, June 2002.
- [21] N. T. Spring, M. Chesire, M. Berryman, V. Sahasranaman, T. Anderson, and B. N. Bershad, "Receiver based management of low bandwidth access links," in *Proc. IEEE INFOCOM*, pp. 245–254, 2000.
- [22] J. Aweya, M. Ouellette, D. Y. Montuno, and Z. Yao, "Enhancing network performance with TCP rate control," in *Proc. IEEE Globecom*, (San Francisco, CA), pp. 1712–1718, 2000.

- [23] R. Mukhtar, S. Hanly, M. V. Ivanovich, H. Vu, and P. G. Fitzpatrick, "Analysis of TCP performance over hybrid fast fixed-to-slow wireless links," in *Proc. IEEE Globecom 2001*, (San Antonio, TX), pp. 1816–1820, 2001.
- [24] H.-K. Shiu, Y.-H. Chang, T.-C. Hou, and C.-S. Wu, "Performance analysis of TCP over wireless link with dedicated buffers and link level error control," in *IEEE International Conference on Communications*, pp. 3211–3216, IEEE, 2001.
- [25] Y. Bai, A. Ogielski, and G. Wu, "Interactions of TCP and radio link ARQ protocol," in *IEEE Vehic. Technol. Conf.*, vol. 3, pp. 1710–1714, 1999.
- [26] R. Jain, "A delay-based approach for congestion avoidance in interconnected heterogeneous computer networks," *ACM Comp. Commun. Rev.*, vol. 19, pp. 56–71, Oct. 1989.
- [27] L. S. Brakmo, S. W. O'Malley, and L. L. Peterson, "TCP Vegas: New techniques for congestion detection and avoidance," in *Proc. ACM SIGCOMM*, (London, UK), pp. 24–35, 1994.
- [28] L. S. Brakmo and L. L. Peterson, "TCP Vegas: End-to-end congestion avoidance on a global internet," *IEEE J. Select. Areas Commun.*, vol. 13, pp. 1465–1480, Oct. 1995.
- [29] J. Mo and J. Walrand, "Fair end-to-end window-based congestion control," *IEEE/ACM Trans. Networking*, vol. 8, pp. 556–567, Oct. 2000.
- [30] D. Katabi, M. Handley, and C. Rohrs, "Internet congestion control for future high bandwidth-delay product environments," in *Proc. ACM Sigcomm 2002*, August, 2002.
- [31] P. F. Quet, B. Ataşlar, A. İftar, H. Özbay, S. Kalyanaraman, and T. Kang, "Rate-based flow controllers for communication networks in the presence of uncertain time-varying multiple time-delays," *Automatica*, vol. 38, pp. 917–928, 2002.
- [32] F. Kelly, A. Maulloo, and D. Tan, "Rate control in communication networks: Shadow prices, proportional fairness and stability," *J. Op. Res. Soc.*, vol. 49, pp. 237–378, 1998.
- [33] S. H. Low and D. E. Lapsley, "Optimization flow control I: Basic algorithm and convergence," *IEEE/ACM Trans. Networking*, vol. 7, pp. 861–875, Dec. 1999.
- [34] S. Kunniyur and R. Srikant, "End-to-end congestion control: utility functions, random losses and ECN marks," in *Proc. IEEE INFOCOM*, pp. 1323–1332, 2000.
- [35] F. Kelly, "Models for a self-managed internet," *Philosophical Transactions of the Royal Society*, vol. A358, pp. 2335–2348, 2000.
- [36] S. H. Low, "A duality model of TCP and queue management algorithms," *IEEE/ACM Trans. Networking*, vol. 11, pp. 525–536, Aug. 2003.
- [37] L. L. H. Andrew, S. V. Hanly, and R. G. Mukhtar, "Analysis of rate adjustment by managing inflows," in *Proc. 4th Asian Control Conference*, (Singapore), pp. 47–52, 2002.
- [38] R. Chakravorty, S. Katti, J. Crowcroft, and I. Pratt, "Flow aggregation for enhanced TCP over wide-area wireless," in *Proc. IEEE INFOCOM*, pp. 1754–1764, 2003.
- [39] Information Sciences Institute University of Southern California, "RFC 793: Transmission control protocol," RFC 793, IETF, 1981.
- [40] L. L. H. Andrew, S. V. Hanly, and R. G. Mukhtar, "CLAMP: Differentiated capacity allocation in access networks," in *Proc. IEEE Int. Performance Computing and Communications Conf.*, (Phoenix), pp. 451–458, Apr. 2003.
- [41] W. Stevens, "TCP slow start, congestion avoidance, fast retransmit, and fast recovery algorithms," RFC 2001, IETF, 1997.
- [42] S. H. Low, F. Paganini, and J. C. Doyle, "Internet congestion control," *IEEE Control Systems Magazine*, vol. 22, pp. 28–43, Feb. 2002.
- [43] V. Paxson and S. Floyd, "Wide area traffic: The failure of Poisson modeling," *IEEE/ACM Transactions on Networking*, vol. 3, pp. 226–244, June 1995.
- [44] M. E. Crovella and A. Bestavros, "Self-similarity in world wide web traffic: Evidence and possible causes," *IEEE/ACM Trans. Networking*, vol. 5, pp. 835–846, Dec. 1997.
- [45] J. Padhye, V. Firoiu, D. F. Towsley, and J. F. Kurose, "Modelling TCP reno performance: A simple model and its empirical validation," *IEEE/ACM Trans. Networking*, vol. 8, no. 2, pp. 133–145, 2000.
- [46] 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.