

MaxNet: Theory and Implementation

Lachlan L. H. Andrew*, Krister Jacobsson†, Steven H. Low*, Martin Suchara*, Ryan Witt*, Bartek P. Wydrowski*

*California Institute of Technology † Royal Institute of Technology (KTH)
Pasadena, CA 91125, USA Stockholm, SE-129 32, Sweden

{lachlan,slow,suchara,witt,bartek}@cs.caltech.edu, krister.jacobsson@s3.kth.se

Abstract—This paper presents the first implementation of the MaxNet TCP network congestion control protocol. MaxNet uses explicit multi-bit signalling from routers to achieve high throughput and low latency over networks of arbitrary capacity and topology, and virtually any delay. The MaxNet algorithm is extended in this paper to give both provable stability and rate fairness. The implementation is based on the Linux Traffic Control framework. The system consists of a sender and receiver TCP algorithm as well as a router module. Performance was tested for capacities up to 1Gbit/s and delays up to 180 ms using the WAN-in-Lab facility. With no overhead but 24-bit price signals, MaxNet can scale from 32 bit/s to 1 Peta-bit/s with a 0.001% rate accuracy. The MaxStart algorithm introduced in this paper allows MaxNet to fill the transmission pipe with data just one round trip time (RTT) after the SYN packet. We detail the MaxNet TCP architecture and discuss various implementation challenges.

I. INTRODUCTION

The aim of network congestion control is to adjust source rates so that they fully utilize and fairly share the network path capacity. Besides from efficiency and fairness, good congestion control performance also requires stable rates to reduce delay jitter, and a fast response to adapt to changes in network load. For Internet-like networks, where links and sources can only have local information, the challenge is to control the source rates in a fully distributed manner.

The approach by most new Transmission Control Protocol (TCP) proposals is to control the source rate based on a congestion signal fed back by the network. Each bottleneck link on the path generates a congestion signal to control the aggregate of traffic on the link. The signal can be computed actively, by an Active Queue Management (AQM) algorithm or it can be generated passively, such as by packet loss or queuing delay in a drop-tail queue.

For ‘loss-based’ congestion control algorithms such as TCP Reno [1], BIC [2], HS-TCP [3], H-TCP [4], S-TCP [5] and TCP Westwood [6] this congestion signal is the packet loss rate. Other ‘delay-based’ proposals such as Vegas [7] and FAST TCP [8] use the queuing delay as the congestion signal.

Explicit-signalling protocols use additional fields in the packet header to communicate the congestion level explicitly. The Explicit Congestion Notification (ECN) standard [9] uses a single bit mark; the rate of sending ECN marks signals the congestion level. Several protocols use multi-bit feedback. XCP [10] explicitly signals the required *changes* in congestion windows. In contrast, RCP [11] and JetMax [12] signal the desired source rate. MaxNet [13] communicates the congestion

level of the *most* congested bottleneck link on the path, from which the desired rate is calculated. All of these except MaxNet require extra fields in addition to the congestion level field, as discussed in Section IX.

Using an explicit multi-bit signal instead of packet loss or delay improves congestion control in several ways. The variability of source rates is reduced compared to using binary loss information due to the increased resolution of the signal. This improves link utilization and delay jitter. An explicit signal also allows periods of congestion to be decoupled from increased packet latency or loss; source rates can be forced to decrease before the onset of these impairments.

As observed in [14], the signal received by a sender using a TCP scheme based on packet loss, delay or ECN marking to signal congestion, is approximately the sum of the signals generated by each bottleneck link on the end-to-end path. These are called SumNet networks. It has been shown [14] that the source rate allocation achieved by SumNet networks maximize a utility function. MaxNet, on the other hand, communicates only the maximum congestion level from the most congested link on the path. In [15] it was proven that MaxNet has faster convergence properties than SumNet networks. This results in low delay jitter and high efficiency.

MaxNet has been shown to have desirable fairness and stability properties. The original MaxNet [13] yields max-min fairness for a network of homogeneous sources, or general weighted max-min fairness for heterogenous sources. However, using homogeneous source functions sacrifices either performance at low Round Trip Times (RTTs) or stability at high RTTs. Alternatively, MaxNet can be made stable on networks of arbitrary capacities, delays and routing by varying the source function with the RTT [16]. However, this approach loses the fairness of the original proposal [13]. The theoretical contribution of this present paper is to add a source dynamic adapted from [17] to achieve both stability and fairness.

Simulation studies of MaxNet have also demonstrated that it is possible to combine MaxNet with other explicit-signalling protocols [18] and that its faster dynamics improve fairness relative to SumNet networks [19].

MaxNet operates with very low queuing delays as it is able to target a controlled link utilization. This results in significantly lower RTTs than loss-based protocols such as Reno which operate with full buffers due to the Additive Increase Multiplicative Decrease (AIMD) probing action. Furthermore, unlike delay based schemes such as Vegas or FAST, the queuing delay does not grow with load.

Whilst many properties of MaxNet have been proven, an implementation of MaxNet for a real network has been lacking. This paper describes an implementation of MaxNet based on the Linux operating system. The implementation includes a Linux router AQM module that can mark packets with an explicit signal, and modifications to TCP to control the window and echo the explicit signal. To address the short flow and low traffic aggregation case, we also integrate the MaxStart concept into MaxNet to allow sources to attain a high transmission rate within two RTTs.

After a description of the principles behind MaxNet in Section II, Sections III and IV describe the MaxNet framework. In Section V and VI the framework's implementation in the Linux Kernel is introduced. Experimental results demonstrating the stability, fairness and convergence speed are presented in Section VII. Section VIII describes how to select provably-stable parameters giving rapid convergence. MaxNet is then compared with related protocols in Section IX.

II. MAXNET BACKGROUND

In this section we summarize the key features of MaxNet introduced in [13], [15], [16]. We will describe the control framework and highlight the main results concerning the equilibrium and stability properties.

The MaxNet control loop consists of the traffic source and the router AQM algorithm. The source rate is controlled by a congestion signal or 'price', denoted $q_i(t)$, which is communicated explicitly from the AQM algorithms on the network. The source rate is set according to

$$x_i(t) = D_i(q_i(t)), \quad (1)$$

where $D(\cdot)$ is the demand function. The demand function is a convex function that describes the source's bandwidth requirement. If all sources have the same demand function, it was shown in [13] that MaxNet achieves max-min fairness. Weighted max-min fairness can be achieved by scaling the demand function.

As illustrated by Figure 1, the price communicated to the source is $q_i(t)$, the maximum of all link prices $p_l(t)$ on the source to destination path of the connection,

$$q_i(t) = \max\{p_l(t); l \in L_i\}, \quad (2)$$

where L_i is the set of links on source i 's path. To communicate the maximum price, each packet has a price field. If link l 's current congestion price $p_l(t)$ exceeds the value q_j in the

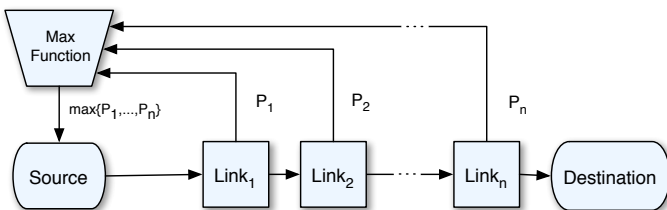


Fig. 1. Conceptual price communication scheme of MaxNet. The price at each link, P_1, P_2, \dots, P_n , is fed back to the sender, which then uses the max of the values.

price field of packet j , the router overwrites the price field with $p_l(t)$. The receiver echoes back the final value q_j in acknowledgements.

A link l computes its price signal $p_l(t)$ using the AQM algorithm

$$p_l(t + dt) = p_l(t) + dt \frac{y_l(t) - \mu_l C_l}{C_l}, \quad (3)$$

where $y_l(t)$ is the traffic rate traversing link l , C_l is the link's capacity and μ_l is the target link utilization. Note that in equilibrium $y_l(t) = \mu_l C_l$, leaving $(1 - \mu_l)C_l$ capacity to absorb traffic bursts. This makes the buffer empty in equilibrium and results in very low network latency.

In [20] it was shown that control-theoretic stability is achieved for a network of any topology, RTT or capacity if price updates have the form (3) and the slope of the demand function

$$\frac{\delta D_i}{\delta q_i} \leq -\frac{\alpha_i x_i}{\tau_i}, \quad (4)$$

where τ_i is the RTT of source i and $\alpha_i \in (0, \pi/2)$ is a gain parameter. As discussed in [17], the second condition places a constraint on the shape of the demand function which is satisfied by the demand function

$$x_i(t) = D_i(q_i(t)) = x_{\max} e^{-\alpha_i q_i(t)/\tau_i}. \quad (5)$$

Whilst (5) satisfies the stability constraints, the rate now depends not just on $q_i(t)$ but also on τ_i which means that sources with different RTTs will not achieve max-min fairness. In [17] a dynamic source algorithm was introduced to implement fairness on slow time scales separately from the fast time scale response which determines stability. On a fast time scale the rate changes are bounded by (4) by setting

$$x_i(t) = x_{\max} e^{\xi_i(t) - \alpha q_i(t)/\tau_i} \quad (6a)$$

and on a slower time scale which does not affect stability ξ_i is adjusted to make the equilibrium rate follow the designer's choice of demand function

$$\dot{\xi}_i = \frac{\alpha \eta}{\tau_i^2} (U'_i(x_i) - q_i), \quad (6b)$$

where $U'_i(x_i)$ is a utility function which relates to the desired demand function by $U'_i(x) = D_i^{-1}(x)$, and η determines the rate of convergence to fairness.

III. SOURCE ALGORITHM

This section describes the design of the source algorithm. The source algorithm is based on the dynamic controller (6) from [17] to achieve stability and weighted max-min fairness.

The key choice in designing the controller is selecting a demand function. Consider the exponential demand function

$$D(q_i) = x_{\max} e^{-q_i/T} \quad (7)$$

which removes the dependence on RTT from (5). In this discussion of equilibrium properties, we drop the time dependence in the variables. By (6a), $U'_i(x_i)$ used in (6b) is

$$U'_i(x_i) = D^{-1}(x_i) = -T \log(x_i/x_{\max}) \quad (8)$$

$$= -T(\xi_i - q_i\alpha/\tau_i). \quad (9)$$

Per ACK, if dt_{\min} has elapsed, update ξ and calculate W :

$$\begin{aligned}\xi &\leftarrow \xi + \frac{\alpha\eta}{\tau^2} dt \left(\left(\frac{T\alpha}{\tau} - 1 \right) q - T\xi \right) \\ W &\leftarrow \tau x_{\max} \exp \left(\xi - \frac{q\alpha}{\tau} \right)\end{aligned}\quad (10)$$

Parameters:

x_{\max}	maximal supported transmission rate
T	parameter that determines speed of convergence
α	overall loop gain
η	η/τ is the zero of the lead-lag compensator
dt_{\min}	minimum update interval

Variables:

ξ	state variable used in window calculation
q	price received in the most recent packet
τ	minimum RTT measurement of the flow
W	window corresponding to q
dt	interval since last update

Fig. 2. Pseudocode of the source algorithm.

1) Every dt_p seconds:

$$\begin{aligned}y_{\text{.dt}} &\leftarrow y_{\text{.dt}} + Q/T_0 \\ p &\leftarrow \max(p + y_{\text{.dt}}/C_l - \mu dt_p, 0) \\ y_{\text{.dt}} &\leftarrow 0\end{aligned}$$

2) On packet arrival:

$$\begin{aligned}y_{\text{.dt}} &\leftarrow y_{\text{.dt}} + \text{pkt.size} \\ \mathbf{if } p > \text{pkt.price } \mathbf{then} \\ &\quad \text{pkt.price} \leftarrow p \\ \mathbf{end if}\end{aligned}$$

Parameters:

μ	target utilization
C_l	link capacity
dt_p	price update interval
T_0	timescale in compensation for virtual queue overflow

Variables:

p	link price
$y_{\text{.dt}}$	aggregate arrivals in update interval dt_p
Q	instantaneous queue length

Fig. 3. Pseudocode of the router algorithm.

The pseudocode of the source algorithm is shown in Figure 2. The current implementation performs the window update on ACK arrivals, at most every dt_{\min} seconds. The calculation is packet driven, thus the calculation is executed at most once per packet, but at least every RTT.

IV. ROUTER ALGORITHM

In this section we describe the router algorithm. The router algorithm consists of the price update law and MaxStart.

A. Virtual queue AQM

The router price update is performed according to (3). The update occurs only every dt_p seconds, to limit the

computational burden. The only per-packet operations are a single addition, comparison and assignment. The pseudo code of the router algorithm is shown in Figure 3.

The increment of $y_{\text{.dt}}$ in step 2 may seem to deviate from the virtual queue of (3), but for suitable T_0 , it implements a virtual queue of the *target* rates of the sources, when the link is saturated. To understand this, consider a bottleneck carrying a single flow.

The aggregate traffic arriving in interval dt_p is $y_{\text{.dt}}$. As MaxNet is a sliding window protocol, the arrival rate at the router is limited by the ‘‘ACK clock’’ not to exceed the output rate. However, the rate the flow seeks to achieve, W/τ , can exceed this rate, yielding a physical queue. The size of this physical queue is the number of packets in the window less the number of packets in the pipe of capacity $C_l = y_l$ and delay τ ; that is $Q = W - (y_l \times \tau)$. The router then knows that the source is attempting to send at aggregate rate $y_l + Q/\tau$.

For the multiple-flow case, T_0 should be a weighted harmonic mean of the τ values of the flows. Since this is not known at the router, a conservative (large) value is chosen. Note that in equilibrium, no physical queue exists because $\mu C < C$, and so this mechanism does not affect the linear stability of the system. However, it may limit the range over which the linear model applies.

Despite its similar form, this does not correspond directly to the queue term in RCP [11], since RCP is rate-based, and flows send at their actual target rates.

B. MaxStart – Replacing slow start

TCP Reno’s slow start mechanism [1] prevents excessive congestion when a new flow starts. Its exponential increase provides a ramp-up time which scales linearly with RTT and logarithmically with the bandwidth, which scales well for unknown bandwidths.

However, it is possible to start faster with explicit signalling. For example, QuickStart [21] enables sources to determine the allowed sending rate on the path and to start almost immediately transmitting at the allowed rate.

Without some form of slow start, explicit signalling protocols would start immediately transmitting at the same rate as the other flows in the link, overloading the link. This problem is especially severe for the first flow arriving at an idle link. RCP simply allows this overload to occur, and aggressively reduces the advertised rate in response to the resulting congestion. MaxNet’s MaxStart protocol seeks instead to avoid the congestion.

MaxStart starts a new flow at the minimum spare capacity of any link on its path, and then ramps up linearly to the advertised rate over several RTTs. In the current implementation, flows ramp up over two RTTs, and the spare capacity is defined as the difference between $(\mu + (1 - \mu)/4)C_l$ and the current transmission rate, where C_l is the physical link capacity and μ is the target utilisation.

MaxStart must perform two tasks: signalling the starting rate from the routers, and senders subsequently increasing their rates. These will be described in turn.

The first packet a sender transmits is flagged to indicate that the sender wants to be informed of the spare capacity instead

of the price; the signalling will be described in Section V. Routers then mark the packet with the lowest spare capacity on the route.

MaxStart terminates as soon as the MaxStart rate exceeds the “price rate” (the rate corresponding to the advertised price). Until that time, the sender increases its target sending rate approximately 16 times per RTT, each time by approximately $1/32$ of Δ , the difference between the price rate and the *initial* MaxStart rate.

The above updates are modified slightly to compensate for burstiness. Specifically, an update occurs on receipt of the first ACK which arrives at least $\text{RTT}/16$ after the previous update. At each update, the MaxStart rate is increased by $\Delta dt/(2\tau)$, where τ is the base RTT and dt is the interval since the previous update.

Further enhancements could be made to extend the MaxStart algorithm to ensure that capacity is not over allocated for multiple flows starting at almost the same time by keeping track of the already allocated capacity. Furthermore, the rate allocation to all flows could be made equal when a new flow starts, even when N is small. This is, however, beyond the scope of this paper.

V. PRICE ENCODING

In contrast to implicit signalling protocols, the price encoding of explicit signalling protocols explicitly determine the range and precision of the achievable rates. This encoding must allow the protocol to scale well beyond current network bandwidths. The demand function (7) gives a uniform relative precision (minimum rate increment as a fraction of the rate) if the price has uniform absolute precision, such as using fixed-point encoding.

The range of rates is determined by B_i , the number of bits allocated to the integer part of the price, and the precision is determined by B_f , allocated to the fractional part. To achieve $x_{\max} = 10^{15}$, (1 peta-bit/s) and $x_{\min} = 32$ bit/s with the demand function (7) and $T = 0.4$, it suffices that $B_i \geq \lceil \log_2(T \log(x_{\max}/x_{\min})) \rceil = 4$. To achieve a relative precision of $\epsilon = 10^{-5}$, $B_f \geq \lceil -\log_2(T \log(1 + \epsilon)) \rceil = 18$. With just these 22 bits, MaxNet achieves rapid convergence to fairness over this future-proof dynamic range with high precision, with no signalling of RTTs, current rates or bottleneck links.

A signalling mechanism for this price must also be specified. The current implementation of MaxNet uses TCP options, although IPv4 options or IPv6 per-hop options could also be used. The option format is shown in Figure 4. The price field is changed by routers to accumulate the price signal on the path from source to destination. The echo field is untouched by routers and returns the price to the sender in the returning ACKs, enabling symmetric communication. The highest bit of the 24 bit price field is 0 if the remaining 23 bits contains a price or 1 if they contain a MaxStart rate.

VI. DETAILS OF LINUX IMPLEMENTATION

The current implementation of MaxNet is a patch relative to Linux 2.6.11. This version pre-dates the modular TCP architecture introduced in 2.6.13. The sender window calculations

are implemented as a hook in the `tcp_cong_avoid` function in file `net/ipv4/tcp_input.c`. When MaxNet is enabled, all changes to the congestion window in other parts of the kernel are immediately overwritten by the MaxNet window.

Sender parameters were set using the `sysctl` interface to $x_{\max} = 10^{15}$ bit/s, $T = 0.4$ seconds, $\alpha = 0.66$, $\eta = 0.06$ and $dt_{\min} = 1 \mu\text{s}$ to update on every ACK. On the router, $dt_p = 1$ ms and $T_0 = 130$ ms were set using the `tc` interface.

Being an equation-based algorithm, MaxNet frequently manipulates fractional values. Linux kernel code cannot use floating point operations, and so MaxNet uses fixed-point arithmetic. The exponential function in (6a) is implemented by a lookup table; this can be optimised using interpolation and bit shifting.

The update for ξ in (10) is a discrete time approximation for (6b). This discretisation can overshoot the equilibrium value given q , namely $\xi = \alpha q/\tau_i - q/T$, although (6b) cannot. This is prevented by clipping ξ to this value if (10) overshoots.

For a software router to operate at 1 Gbit/s on a non-real-time operating system such as Linux, unnecessary large processes must be avoided. Processes such as graphical user interfaces can cause delays of several milliseconds, resulting in large transient queues.

Prices were averaged at the receiver over one RTT. The average was weighted by the interval since the previous price signal, to reduce the impact of burstiness on the averaging.

According to (3), when links first become bottlenecks, their prices have to rise gradually from 0. During this time, sources would be told to transmit at almost $x_{\max} = 10^{15}$ bit/s. To prevent this, routers’ prices are clipped below at $p_{\min,l} = D^{-1}(C_l)$, with D given by (7).

VII. EXPERIMENTS

The performance of MaxNet in two situations will now be described. The first demonstrates its fairness, convergence speed and scalability, and the second investigates its response to cross traffic.

A. Multiple flows and links

Internet flows typically contain two congested links, one in the sender’s access network and one in the receiver’s access network. This experiment evaluates how MaxNet responds in a multi-flow multi-link environment. This demonstrates its fairness and scalability, and how it behaves when bottleneck links change.

Figure 5 shows the topology for this experiment. Link 1 is 622 Mbit/s, with a RTT of 29 ms provided by an OC-48 link of WAN-in-lab [22], and Link 2 is a 400 Mbit/s link with RTT 150 ms provided by a dummynet. The target utilisation was 90% ($\mu = 0.9$). Flow Figure 6 shows when different flows start, dividing the experiment into six intervals.

opt	optsize		
42	6	echo	price
(1 byte)	(1 byte)	(3 bytes)	(3 bytes)

Fig. 4. MaxNet option format.

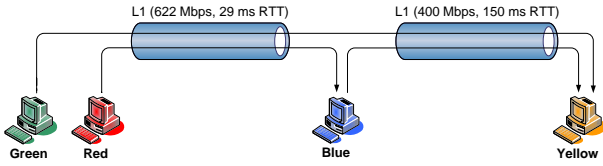


Fig. 5. Topology for multi-flow experiments.

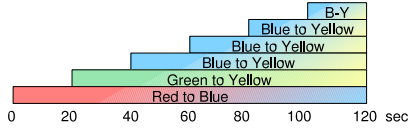


Fig. 6. Start times and durations of flows for multi-flow experiments.

Figures 7 and 8 show the rates of the flows, and the queue sizes of the links, respectively. The rates are averaged over one second intervals. On a faster timescale, there is noticeable burstiness because the implementation is window based not rate based; this can be overcome by better pacing of window increases, without the expense of packet pacing.

This simple experiment illustrates many important properties of the protocol, many of which are not tested by the traditional “dumbbell” (single bottleneck) topology.

1) *Convergence speed*: Due to MaxStart, MaxNet shows rapid convergence to full utilisation. These results show that the initial rise time of each flow is less than the 1 s sampling interval, in keeping with the nominal rise time of two RTTs.

2) *Fairness*: Reno is known to give significantly unfair rates to flows with different RTTs [23], and many proposed TCPs for large bandwidth-delay product networks are even less fair [2]. In contrast, interval 2 shows that MaxNet converges to fairness within 20 s (after a fast convergence to full utilisation), for flows with RTTs differing by a factor of 6.

Even protocols such as H-TCP [4] and FAST [8] which do not suffer from RTT unfairness give higher rates to flows traversing fewer bottlenecks, because the congestion measure (delay or loss) is summed over all links on the path. Interval 3 shows that MaxNet converges within 20 s to fair allocation between the flow from Green to Yellow, traversing two bottlenecks, and that from Blue to Yellow, traversing one.

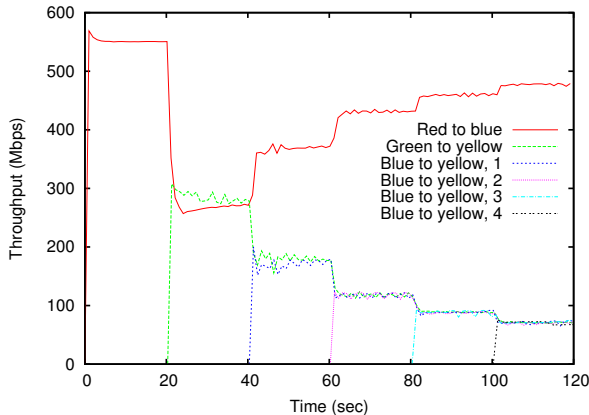


Fig. 7. Rates of flows in the two-hop experiment.

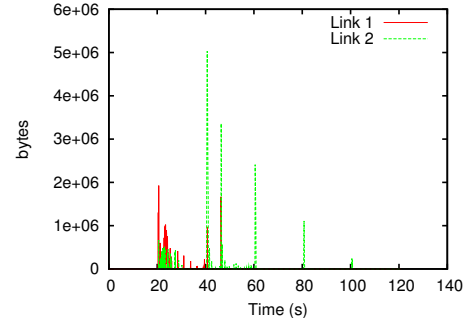


Fig. 8. Queue sizes in the two-hop experiment.

3) *Queueing*: MaxNet’s virtual queue mechanism gives an equilibrium queue size of zero. This both improves the performance of real-time services and reduces the need for memory needed in routers. When the number of flows is very small, transient queues exist when flows arrive, but the magnitude of these queues decreases rapidly as the number of flows increases. To quantify this, note that if there are already N flows in equilibrium bottlenecked at a link, then a new flow will transmit at rate at most $\mu C_l/N$ causing overload of at most $((1 + 1/N)\mu - 1)C_l$ for up to the longest RTT of any flow using the link. The overload drops to zero for $N > \mu/(1 - \mu)$.

The cause of the queue at 45 s is not clear.

4) *Switching bottlenecks*: Max-min protocols, such as MaxNet, RCP and JetMax, undergo discrete transitions when the bottleneck link for a flow changes. At 40 s, the bottleneck for the flow from Green to Yellow switches from Link 1 to Link 2. Significantly, this does not cause instability in the form of “ping-ponging” between bottlenecks as the prices stabilise. However, it does result in the highest queueing in the experiment, 5 MByte or 90% of the bandwidth-delay product of the flow from Green to Yellow before the switch.

5) *Increase and decrease in available bandwidth*: As the load on of Link 2 increases, the bandwidth available to the flow from Red to Blue increases. MaxNet quickly increases its window to use the extra bandwidth within around 2 s.

B. Cross traffic

MaxNet was run for 30 s on a single 1 Gbit/s link with 29 ms RTT and target utilisation 94% ($\mu = 0.94$). From 10 s to 20 s, a 400 Mbit/s constant bit rate (CBR) flow shared the link. Figure 9 shows the rates achieved by each flow. Note that this is a heavier CBR load than most encountered in practice, and provides an arduous test.

At the start, MaxNet again converges rapidly to the target 94% utilisation.

When the CBR flow starts, the MaxNet flow relinquishes bandwidth almost immediately, because of the ACK-clocking inherent in window-based protocols. After a few seconds, the target rate drops to the available bandwidth and the total utilisation drops back to 94%, observable as a slight dip in the MaxNet flow’s rate.

When the CBR flow ends, the dynamics of (6) can be observed. Over two thirds of the spare capacity is reclaimed

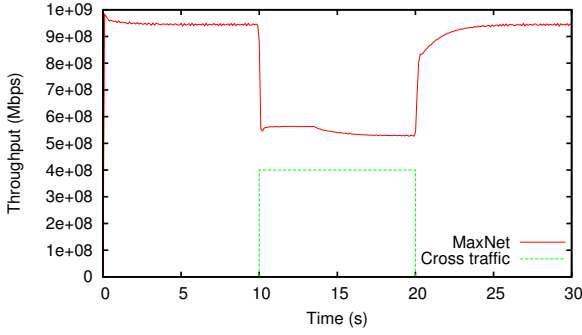


Fig. 9. Rates of MaxNet and 400Mbit/s CBR flow with a target rate of 940Mbit/s

by the MaxNet flow within 0.3 s due to the rapid drop in price. The remaining rate increase is slower due to the interaction between the price and the variable ξ used to ensure stability, but the target utilisation is still reached within a few seconds.

VIII. STABILITY AND TRANSIENTS

MaxNet is the first practical algorithm designed based on the theory developed in [17]. MaxNet’s stability can be established by using that theory. Subsection VIII-E at the end of this section is aimed at designers wishing to change MaxNet’s parameters.

A. Background

Let us now review the relevant results of [17], adapted to the case of MaxNet. These results apply to general multi-link networks with heterogeneous RTTs. Reference [17] aimed to provide stability for a range of RTTs for flows with *arbitrary* utility functions, since the utility function determined the equilibrium of the networks considered there. Since the equilibrium of MaxNet is independent of the utility function, provided that all sources use the same utility function, our aim is to determine the utility function and parameters which will improve the transient performance. In particular, some combinations of parameters are theoretically stable but give a lightly damped (highly oscillatory) response; these must be avoided for practical networks.

Stability in the presence of delay requires that the loop gain be sufficiently low. It was shown in [17] that the loop gain is determined by the slope of the demand function, and placing restrictions on the family of (static) demand functions which are stable. It was further shown that the stability of network using lead-lag¹ controllers, such as MaxNet, can be determined from the function

$$F(s, \tau_i; \alpha) = \alpha \frac{e^{-s\tau_i}}{s\tau_i} \frac{s+z}{s+z\kappa_i/\nu_i}, \quad (11)$$

where s is the complex frequency, τ_i is the RTT of the i th flow, $\kappa_i = \alpha_i x_{0i}/\tau_i$ is the slope of a static demand function which would result in stability, and ν_i is the slope of the “true”

¹Strictly, the controller considered here is either a pure-lead controller or a pure-lag controller, depending on the values of T and τ . As it is unknown which is the case, we follow [17] and call it a lead-lag controller.

demand function, D_i , at the operating point. This function appears as the elements of a diagonal matrix in the loop gains of the system. Although [17] applied to a different structure of flow control, the triangulation approach of [16] can be used to show that the result also applies to MaxNet, provided that no two bottleneck links have equal equilibrium prices.

The stability proof for MaxNet is based on the following result, which follows easily from the results of [17], [24]:

Lemma 1: Let $H(\omega; \alpha) = \text{Co}\{F(j\omega, \tau_i; \alpha)\}$ be the convex hull of F evaluated at the RTTs of the individual flows, at a given frequency ω . The system will be stable if the trajectory of $H(\omega; \alpha)$ for $\omega \in \mathbb{R}^+$ does not intersect the negative real axis to the left of $-1 + 0j$. \square

The trajectory of H is a generalised form of Nyquist curve. In [17], the speed of dynamics was set according to the flow with the longest RTT, by setting $z = \eta/\bar{\tau}$, for a sufficiently small η , where $\bar{\tau}$ is an upper bound on τ_i . The stability proof considered the system at two timescales. It was shown that for frequencies $\omega < 1/\max_i(\tau_i)$, the convex hull $H(\omega; \alpha)$ lies entirely below the real axis, while for larger frequencies, it is contained in a particular spiral which is bounded away from $-1 + 0j$.

B. Obtaining consistent dynamics

An important parameter governing the transient behaviour is the ratio κ_i/ν_i . This depends on the demand function and, in general, on the equilibrium operating point of the non-linear system. When $\kappa_i/\nu_i > 1$, the compensator is a lead-compensator which can improve the rise time and settling time of the system. However, when $\kappa_i/\nu_i < 1$, the resulting “lag-compensator” adds more phase lag (analogous to delay). This produces a “resonance peak” resulting in high gain at a particular frequency, which makes the system highly oscillatory. Note that the systems which do not suffer this excess oscillation are exactly those which would be stable even without the lead-lag compensator of [17]. Thus, the compensator is useful as an insurance against extreme RTTs, rather than extending the range of demand functions that can be deployed in practice.

Since MaxNet’s equilibrium is independent of the (common) demand function, the demand function can be chosen to improve the transients. In particular, the ratio κ_i/ν_i can be made independent of the operating point by using a demand function

$$x(q) = x_{\max} e^{-q/T}, \quad (12)$$

giving $\kappa_i/\nu_i = \alpha T/\tau_i$. This ensures that the rate of convergence will not depend on the capacity of the bottleneck link.

C. Need for a new stability proof

In contrast to [17] which uses a lead-lag parameter, z , dependent on the largest RTT in the network, the current implementation of MaxNet adapts z to each flow’s own RTT, setting $z_i = \eta/\tau_i$ as opposed to $z_i = \eta/\bar{\tau}$ as in [17]. This yields

$$F(s, \tau_i; \alpha) = \alpha \frac{e^{-s\tau_i}}{s\tau_i} \frac{s\tau_i + \eta}{s\tau_i + \eta\alpha T/\tau_i}. \quad (13)$$

-
- 1) Find $\underline{\omega} = \omega_0(\bar{\tau})$ by (14)
 - 2) Using (13), construct the Nyquist spiral

$$\mathcal{S} = \{F(j\omega, \bar{\tau}; 1) : \omega > \underline{\omega}\}.$$
 - 3) Similarly, construct the tail

$$\mathcal{T} = \{F(j\underline{\omega}, \tau; 1) : \tau < \bar{\tau}\}.$$
 (Note that this is not the tail of any Nyquist plot, as ω is fixed.)
 - 4) Construct a line entirely to above each curve, and denote the point at which this intersects the real axis by $-1/\alpha_{\max}$. That is, construct $\mathcal{L} = \{x + jy : y = \gamma(x + 1/\alpha_{\max})\}$ with α_{\max} and γ such that $(x + jy_1) \in \mathcal{L}$ and $(x + jy_2) \in \mathcal{S} \cup \mathcal{T}$ imply $y_1 \geq y_2$.
-

Algorithm 3. Determining stable parameters.

In this case the stability proof of [17] needs modification, since spiral used in that proof no longer encloses the Nyquist curves for all frequencies $\omega > 1/\bar{\tau}$. However, the same principle of studying the system at two timescales can again be used. There is again a threshold frequency $\underline{\omega}$ (depending on $\bar{\tau}$ and η) such that the convex hull $H(\omega; \alpha)$ is below the real axis for frequencies below $\underline{\omega}$. It is also possible to choose α small enough such that $H(\omega; \alpha)$ is strictly below a slanted line through $-1 + 0j$ for frequencies above $\underline{\omega}$. The theoretical complication arising from adapting z to each flow's RTT is that, unlike in [17], $\underline{\omega} \neq 1/\bar{\tau}$.

Using this approach, it can be shown that MaxNet is stable for all RTTs up to $\tau = 1000$ seconds using the parameters of Section VII, namely $T = 0.4$ s, $\alpha = 0.66$ and $\eta = 0.06$. If z were independent of τ_i as in [17], ensuring stability for $\tau = 1000$ seconds would require $z < 10^{-3}$, and it would take a quarter of an hour for flows to achieve their equilibrium (fair) rates, in contrast to the 20 s shown in Figure 7.

D. Determining stable parameters

The first step in choosing suitable parameters is finding the provably stable combinations. Following [17], it will be assumed that an upper bound, $\bar{\tau}$, on the RTT of any flow is known; the system will be designed to be stable for all $\tau < \bar{\tau}$.

For a given value of αT , and a given lead-lag coefficient η , the following is a method to find the range of overall gain α which gives stability.

Define

$$\omega_0(\tau) = \min\{\omega : \text{Im}(F(j\omega, \tau; 1)) = 0\} \quad (14)$$

to be the lowest frequency at which the spiral for τ crosses the real axis, where Im denotes the imaginary part. A given ω and τ are said to be “in the tail” if $\omega < \omega_0(\tau)$, and “in the spiral” otherwise.

Given $\alpha T \in (0, \alpha\bar{\tau}]$ and $\eta > 0$, the steps to choose α yielding a stable system are given in Algorithm 3. Figure 10 shows the construction.

Proposition 1: Under the construction of Algorithm 3, MaxNet is stable for any $\alpha < \alpha_{\max}$ for any number of flows and any network topology with maximum delay $\bar{\tau}$. \square

For a maximum RTT of $\bar{\tau} = 1000$ seconds, the parameters of Section VI satisfy this proposition with \mathcal{L} having slope

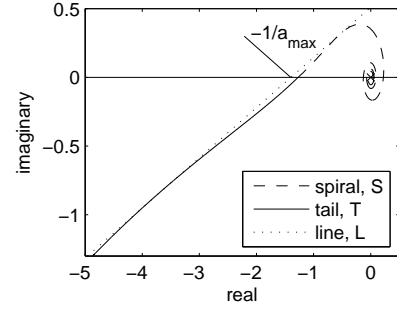


Fig. 10. Spiral \mathcal{S} , Tail \mathcal{T} , and line \mathcal{L} together with resulting α_{\max} for the illustrative case of $\bar{\tau} = 3$ s $\eta = 0.7$ and $\alpha T = 0.1$ s

$\gamma = 0.3504$, and $\underline{\omega} = 0.001525$.

The proof of Proposition 1 is in two parts. The first shows that for $\omega < \underline{\omega}$, all Nyquist curves are below the real axis. The second has two subparts. The first shows that the spiral for $\tau < \bar{\tau}$ is within \mathcal{S} (the spiral for $\bar{\tau}$) by showing that the magnitude of F is a decreasing function of τ for a fixed argument. The second shows that the portion of the tail with $\omega \geq \underline{\omega}$ is within the convex hull of \mathcal{T} by showing that the magnitude of F is a decreasing function of ω for a fixed argument.

The proof involves studying the functional relationship between F and several variables. With the obvious abuse of notation, these functions will all be called F , but with different argument lists. Let $\phi = \omega\tau$ and

$$\theta(\tau, \phi) = \text{Arg}\left(\frac{\eta + j\phi}{\eta\alpha T/\tau + j\phi}\right) - \frac{\pi}{2} - \phi \quad (15)$$

so that $F(j\phi/\tau, \tau; 1) = |F(j\phi/\tau, \tau; 1)| \exp(\theta(\tau, \phi))$. The following lemma is proven in [25].

Lemma 2: For any given $\omega > 0$, $\tau > 0$, $\phi > 0$ and $\theta < 0$,

- 1) $d\omega_0(\tau)/d\tau < 0$
- 2) $d|F(\theta, \phi)|/d\phi < 0$ if $\theta < -\pi/2$
- 3) $d|F(\theta, \tau)|/d\tau > 0$ if $\theta < -\pi/2$
- 4) $d|F(\theta, \omega)|/d\omega < 0$
- 5) $d\arg(F(\phi, \tau))/d\tau < 0$
- 6) $d|F(\phi, \tau)|/d\tau > 0$

where the derivative of $\arg(\cdot)$ is defined modulo 2π . \square

Proposition 1 can now be proved.

Proof: By Lemma 1, it is sufficient to prove that, for any ω , $H(\omega; \alpha)$ does not intersect the real axis to the left of $-1 + j0$. Since α merely scales F , this is equivalent to $H(\omega; 1)$ not intersecting the real axis to the left of $-1/\alpha$, which is left of the intercept of \mathcal{L} .

For any ω and τ in the tail, $F(j\omega, \tau; 1)$ is below the real axis; this follows from $\lim_{\omega \rightarrow 0} \arg(F(j\omega, \tau; 1)) = -\pi/2$, the continuity of F and the definition of the tail.

It will now be shown that (i) for any $\omega < \underline{\omega}$, $H(\omega; 1)$ will be entirely below the real axis, and (ii) for any $\omega \geq \underline{\omega}$, $H(\omega; 1)$ will be entirely below the oblique line \mathcal{L} .

(i) Consider an $\omega < \underline{\omega}$. By Lemma 2(1), $\omega < \omega_0(\tau)$ for all $\tau < \bar{\tau}$. Thus ω and τ are in the tail for all $\tau < \bar{\tau}$, and hence $F(j\omega, \tau; 1)$ is below the real axis for all τ , implying $H(\omega; 1)$ is also.

(ii) It remains to show that, for all $\omega \geq \underline{\omega}$, $H(\omega; 1)$ is below \mathcal{L} if $\tau < \bar{\tau}$. The cases of τ and ω being in the tail and in the

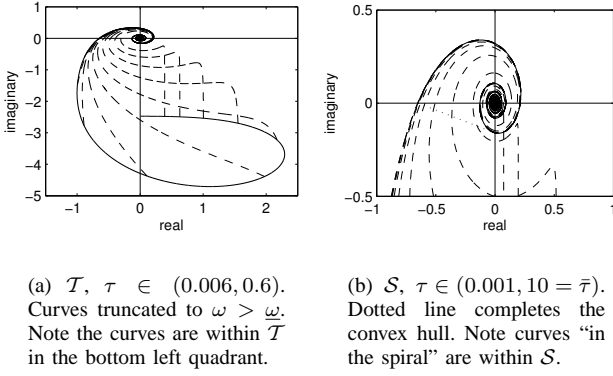


Fig. 11. Plots of \mathcal{T} and \mathcal{S} (solid lines), and Nyquist curves for varying τ .

spiral will be considered separately.

If τ and ω are in the tail, then $F(j\omega, \tau; 1)$ is below the real axis, and will be below \mathcal{L} unless it is in the bottom left quadrant, corresponding to $\theta \in [-\pi, -\pi/2]$. In that quadrant, $F(j\omega, \tau; 1)$ will be in the convex hull of $\mathcal{T} \cup \{0 + j0\}$ by lemma 2(4), which lies completely below \mathcal{L} by construction. This is illustrated in Figure 11(a), and establishes the result of this paragraph.

Conversely, if τ and ω are in the spiral, then $F(j\omega, \tau; 1)$ is within the convex hull of \mathcal{S} , by lemma 2(3). This is illustrated in Figure 11(b). Since $\text{Co}(\mathcal{S})$ is entirely below \mathcal{L} , it follows that $F(j\omega, \tau; 1)$ also is.

For a given $\omega > \underline{\omega}$, $F(j\omega, \tau; 1)$ is in the convex region below \mathcal{L} for all $\tau < \bar{\tau}$, and thus their convex hull is also within that region. This establishes case (ii) and hence the proposition. ■

It is not necessary to construct the complete sets \mathcal{S} and \mathcal{T} . It is only necessary to construct the outermost arc of \mathcal{S} in the upper left quadrant. Determining how much of \mathcal{T} is required is more complex. Given a line $\mathcal{L}' = \{x + jy : y = \gamma'(x + x')\}$, and a bounded subset $\mathcal{T}' \subseteq \mathcal{T}$, it is desirable to know whether \mathcal{L}' is above \mathcal{T} . A sufficient condition is provided by the following result.

Proposition 2: Consider a line \mathcal{L}' . Let $\mathcal{T}' = \mathcal{T} \cap \{x + jy : y > \gamma x\}$ be \mathcal{T} truncated to $\tau > \tau'$, where τ' is the largest value in the tail for which the line between the origin and $F(j\omega, \tau'; \alpha)$ is parallel to \mathcal{L}' . If \mathcal{L}' is above \mathcal{T}' then \mathcal{L}' is also above \mathcal{T} . □

Proof: This follows from the fact that $\arg(F(j\phi/\tau, \tau; \alpha))$ increases as τ decreases, by lemma 2(5). ■

E. Parameters for rapid convergence

The parameters used in Section VII are suitable for most networks. Networks with unusually high RTTs, or the need for particularly fast dynamics, may require other parameter sets. The following empirical procedure considers practical performance, as well as theoretical stability.

- 1) Let τ be the maximum RTT, τ , for which rapid convergence is required. Set $\alpha T = \tau$.
- 2) For $\eta \approx 0.1$, use (13) to select α to give a phase margin of 45° ; that is, $\text{Arg}(F(j\omega, \tau; \alpha)) > -3\pi/2$ for all ω such that $|F(j\omega, \tau; \alpha)| > 1$.

- 3) Empirically adjust η to balance the initial rise time for a single flow against convergence to equilibrium; lower η reduces the initial rise, but increases the settling time.
- 4) For the selected parameters, use Algorithm 3 to verify stability for a sufficiently high $\bar{\tau}$.

IX. COMPARISON WITH OTHER PROTOCOLS

In this section we compare MaxNet with other prominent explicit signalling protocols, XCP, RCP and JetMax. Experimental comparisons are not performed, none of these appear to have released implementations capable of operating at 1 Gbit/s, but NS-2 simulations of RCP are reported. Flow control protocols developed for Asynchronous Transfer Mode (ATM) are also not considered here since they generally require per-flow information at the switch, which is discouraged in IP networks.

The XCP protocol and MaxNet differ in several regards. XCP only achieves constrained max-min fairness [26], which can yield source rates an arbitrarily small fraction of the max-min fair rates, in contrast to MaxNet's bound of μ_l . Furthermore, linear stability of XCP has so far only been proven for a single link with sources of homogeneous RTT. Indeed recent results indicate that XCP can exhibit oscillatory behaviour under more diverse circumstances [27]. In this paper we prove the linear stability of MaxNet for arbitrary networks topologies. In terms of implementation, XCP is also more complicated at the router, requiring 12 operations per packet compared to 2 for MaxNet and requires 16 bytes compared to 6 bytes in the packet header.

RCP [11] has a similar structure to MaxNet, which yields similar dynamics for homogeneous delays. One difference is in how they avoid equilibrium queues. MaxNet uses a virtual queue with capacity marginally below the true link capacity, while RCP has a parameter β which, when non-zero, explicitly reduces the sending rates in the presence of a queue.

The relationship between RCP and MaxNet is clearly seen by considering a network with homogeneous delays, τ , and setting $\beta = 0$ for RCP, and the virtual queue capacity to the true link capacity for MaxNet. In this case, RCP updates the advertised rate every small dt by

$$R(t) = R(t - dt) \left(1 + dt \frac{\alpha(C - y(t))}{\tau C}\right) \quad (16)$$

Taking the log of (16) and using $\log(1 + x) \approx x$ gives

$$\log(R(t)) = \log(R(t - dt)) + dt \frac{\alpha(C - y(t))}{\tau C} \quad (17)$$

Changing variables using demand function $R(t) = e^{-\alpha p(t)/\tau}$ yields MaxNet's price update law (3) with $\mu_l = 1$.

More fundamentally, RCP and MaxNet differ in how they trade off speed of convergence with stability. Delayed feedback systems need to scale their feedback down for long RTTs. In MaxNet, this is done at each source, since the sources know their own RTTs. In RCP, this is done by the routers based on the traffic-weighted average RTT advertised in each packet header.

The drawback of MaxNet's approach is that a global parameter, αT , must be set to ensure acceptable performance

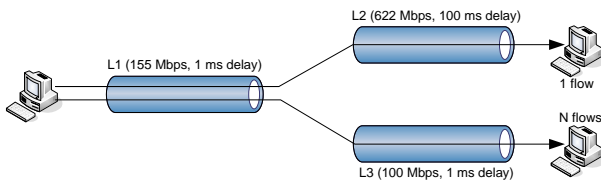


Fig. 12. Network for which RCP is unstable.

for high-RTT flows, which limits the speed of response for low-RTT flows. The drawback of RCP's approach is that it can be unstable. We use $\alpha = 0.4$ and $\beta = 1$, but believe that for any set of parameters, an unstable topology can be found.

Consider a network shown in Figure 12 with a 155 Mbit/s bottleneck (L_1) with 1 ms delay carrying $N+1$ flows, of which Flow 1 continues over a 622 Mbit/s link (L_2) with 100 ms delay and the remaining N split off onto a 100 Mbit/s link (L_3) with 1 ms delay. For $N = 1$, RCP stabilises after 3 s, after overshooting by around 50%. For $N = 9$, the rate of the Flow 1 oscillates between 10 Mbit/s and up to 100 Mbit/s for the first 10 seconds, and then settles down to steady aperiodic oscillations of about about 30% of the link capacity. For $N = 49$ the rate switches rapidly between 0 and the link capacity. Graphs are presented in [25]. MaxNet has been confirmed to be stable for this network.

The reason for this is that the dynamics of the price of L_1 are on a fast timescale dominated by the traffic which is bottlenecked by L_3 , while the mean RTT of the flows actually bottlenecked at L_1 is much larger. Note that L_1 precedes L_3 in the path, and so packets cannot easily signal to L_1 that they are bottlenecked elsewhere. One remedy would be to include packet information identifying the controlling bottleneck.

In JetMax [12], routers calculate a target rate by estimating the number of flows bottlenecked at that link, and estimating the capacity used by non-bottlenecked flows. For this, it uses four 32-bit fields to signal current rate and congestion information, and three 8-bit fields to identify the bottleneck router explicitly. This does not include fields to communicate the control information back from the receiver to the sender. It is not clear how JetMax estimate which flows are "responsive".

Protocols in which routers change state based on explicit signals from sources may also create vulnerabilities to denial of service attacks. Such issues are beyond the scope of this paper.

X. CONCLUSION

Explicit signalling allows flow control to maintain high utilisation with small average queues, to rise to full line rate within one or two RTTs and share bandwidth fairly. MaxNet is such a protocol which has been designed to be easily implemented and provably stable, while minimising signalling overhead. Experiments on an initial implementation of MaxNet have shown that it achieves the above goals, and that it scales well to large numbers of flows.

XI. ACKNOWLEDGMENT

This research is part of the WAN-in-Lab project, supported by NSF grant no. 0303620.

REFERENCES

- [1] V. Jacobson, "Berkeley TCP evolution from 4.3-tahoe to 4.3-reno," in *Proc. 18th Internet Engineering Task Force*, Vancouver, Aug. 1990.
- [2] L. Xu, K. Harfoush, and I. Rhee, "Binary increase congestion control (BIC) for fast long-distance networks," in *Proc. IEEE Infocom*, Mar. 2004.
- [3] "Highspeed TCP for large congestion windows," RFC 3649, Mar. 2003.
- [4] D. Leith and R. Shorten, "H-TCP: TCP for high-speed and long-distance networks," in *Proc. PFLDnet*, Argonne, 2004.
- [5] T. Kelly, "Scalable TCP: Improving performance in highspeed wide area networks," *ACM Comp. Commun. Rev.*, vol. 33, no. 2, Apr. 2003.
- [6] S. Mascolo, C. Casetti, M. Gerla, M. Sanadidi, and R. Wang, "TCP Westwood: end-to-end bandwidth estimation for efficient transport over wired and wireless networks," in *Proc. ACM Mobicom*, Rome, July 2001.
- [7] L. S. Brakmo, S. W. O'Malley, and L. L. Peterson, "TCP Vegas: new techniques for congestion detection and avoidance," in *Proc. SIGCOMM*, London, UK, Sept. 1994.
- [8] C. Jin, D. X. Wei, and S. H. Low, "FAST TCP: motivation, architecture, algorithms, performance," in *Proc. IEEE Infocom*, Mar. 2004.
- [9] K. K. Ramakrishnan and S. Floyd, "Proposal to add explicit congestion notification (ECN) to IP," RFC 2481, Jan. 1999.
- [10] D. Katabi, M. Handley, and C. Rohrs, "Congestion control for high bandwidth-delay product networks," in *Proc. SIGCOMM*, Pittsburgh, PA, 2002.
- [11] N. Dukkupati and N. McKeown, "Why flow-completion time is the right metric for congestion control," *ACM SIGCOMM Computer Communication Review*, vol. 36, no. 1, pp. 59–62, Jan. 2006.
- [12] Y. Zhang, D. Leonard, and D. Loguinov, "JetMax: Scalable max-min congestion control for high-speed heterogeneous networks," in *IEEE INFOCOM*, Barcelona, Spain, April 26, 2006.
- [13] B. Wyrowski and M. Zukerman, "MaxNet: A congestion control architecture for MaxMin fairness," *IEEE Commun. Lett.*, vol. 6, pp. 512–514, Nov. 2002.
- [14] F. P. Kelly, "Charging and rate control for elastic traffic," *European Transactions on Telecommunications*, vol. 8, pp. 33–37, 1997.
- [15] B. Wyrowski, L. L. H. Andrew, and I. M. Y. Mareels, "MaxNet: Faster flow control convergence," in *Proc. Networking 2004*, 2004, Springer Lecture Notes in Computer Science LNCS 3042.
- [16] B. Wyrowski, L. L. H. Andrew, and M. Zukerman, "MaxNet: A congestion control architecture for scalable networks," *IEEE Commun. Lett.*, vol. 7, pp. 511–513, Oct. 2003.
- [17] F. Paganini, Z. Wang, J. Doyle, and S. Low, "Congestion control for high performance, stability and fairness in general networks," *IEEE/ACM Trans. Networking*, vol. 13, no. 1, pp. 43–56, Feb. 2005.
- [18] L. L. H. Andrew and B. P. Wyrowski, "Performance of networks containing both MaxNet and SumNet links," in *HetNets '03*, Ilkley, UK, 2003, pp. 71–7/10. [Online]. Available: <http://netlab.caltech.edu/lachlan/abstract/hetnets.pdf>
- [19] D. Nguyen, J. Wang, L. L. H. Andrew, and S. Chan, "MaxNet: A more efficient max-min fair allocation scheme," in *Proc. Intl. Teletraffic Congress (ITC)-19*, Beijing, China, 2005.
- [20] F. Paganini, J. C. Doyle, and S. H. Low, "Scalable laws for stable network congestion control," in *Proc. IEEE Conf. Decision Contr. (CDC)*, Orlando, FL, 2001, pp. 185–90.
- [21] A. Jain, S. Floyd, M. Allman, and P. Sarolahti, "Quick-start for TCP and IP," Internet draft draft-ietf-tsvwg-quickstart-05.txt, July 2006.
- [22] "WAN in Lab," July 2005. <http://wil.cs.caltech.edu>
- [23] T. V. Lakshman and U. Madhow, "The performance of TCP/IP for networks with high bandwidth-delay products and random loss," *IEEE/ACM Trans. Networking*, vol. 5, no. 3, June 1997.
- [24] G. Vinnicombe, "On the stability of end-to-end congestion control for the internet," University of Cambridge, Tech. Rep. CUED/F-INFENG/TR.398, Dec. 2000. [Online]. Available: <http://www-control.eng.cam.ac.uk/gv/internet/TR398.pdf>
- [25] L. L. H. Andrew, K. Jacobsson, S. H. Low, M. Suchara, R. Witt, and B. P. Wyrowski, "MaxNet: Theory and implementation." [Online]. Available: <http://netlab.caltech.edu/maxnet/MaxNet.Implementation.TechReport.pdf>
- [26] S. H. Low, L. L. H. Andrew, and B. P. Wyrowski, "Understanding XCP: Equilibrium and fairness," in *Proc. IEEE Infocom*, Miami, Florida, Mar. 2005.
- [27] L. L. H. Andrew, B. P. Wyrowski, and S. H. Low, "An example of instability in XCP." [Online]. Available: <http://netlab.caltech.edu/lachlan/abstract/xcpInstability.pdf>