Implementation of Provably Stable MaxNet

Martin Suchara[1,2], Lachlan L. H. Andrew[1], Ryan Witt[1], Krister Jacobsson[3], Bartek P. Wydrowski[1], Steven H. Low[1]

[1]California Inst. of Tech. [2]Princeton Univ. [3]Royal Inst. of Tech. (KTH)

Pasadena, CA 91125, USA   Princeton, NJ 08544, USA   Stockholm, SE-129 32, Sweden

{suchara,lachlan,slow}@caltech.edu, ryan@fastsoft.com, kringlan@ee.kth.se, bwydrowski@gmail.com

Abstract—**MaxNet TCP is a congestion control protocol that uses explicit multi-bit signalling from routers to achieve desirable properties such as high throughput and low latency. In this paper we present an implementation of an extended version of MaxNet. Our contributions are threefold. First, we extend the original algorithm to give both provable stability and rate fairness. Second, we introduce the MaxStart algorithm which allows new MaxNet connections to reach their fair rates quickly. Third, we provide a Linux kernel implementation of the protocol. With no overhead but 24-bit price signals, our implementation scales from 32 bit/s to 1 peta-bit/s with a 0.001% rate accuracy. We confirm the theoretically predicted properties by performing a range of experiments at speeds up to 1 Gbit/sec and delays up to 180 ms on the WAN-in-Lab facility.**

I. INTRODUCTION

The aim of congestion control is to adjust source rates so that they fully utilize link capacities and respond quickly to changes in network load while avoiding delay jitter. An additional goal is to share link capacities fairly. The typical approach of Transmission Control Protocols (TCP) is to control the source rates based on a congestion signal which is generated by each link and fed back by the network. The signal can be calculated actively by an Active Queue Management (AQM) algorithm, or the source can passively observe packet loss or delay.

Congestion control algorithms which decrease sending rates with increasing packet loss include TCP Reno [1], CUBIC [2], or H-TCP [3]. Other proposals that react to queuing delay include Vegas [4] and FAST TCP [5].

Routers in explicit-signalling protocols mark packet headers with information about congestion. The Explicit Congestion Notification (ECN) standard [6] uses a single bit mark; the rate of sending ECN marks signals the congestion level. Protocols that use multi-bit feedback include XCP [7], RCP [8] and JetMax [9]. MaxNet [10] differs from these protocols in that it signals the congestion level of the most congested bottleneck link on the path, which is used to calculate the source rate.

Using an explicit multi-bit signal instead of packet loss or delay is advantageous in several ways. The increased resolution of the signal reduces variability of source rates, which improves link utilisation and decreases delay jitter. Explicit signalling can also prevent packet latency or loss: rates can be decreased before the impairments occur.

As observed in [11], the congestion signal received by a sender using a TCP scheme based on packet loss, delay, or ECN marking, is approximately the sum of the signals generated by each bottleneck link on the end-to-end path. Thus we call these networks SumNets. MaxNet, on the other hand, communicates only the maximum congestion level from the most congested link on the path. In [12] it was proven that MaxNet has faster convergence properties than SumNets. This results in low delay jitter and high efficiency.

Another advantage of MaxNet is that it operates with very low queuing delays as it is able to target a controlled link utilisation. This results in significantly lower RTTs compared to loss-based protocols such as Reno which have to fill buffers to observe losses. Furthermore, unlike delay based schemes such as Vegas or FAST, the queueing delay does not grow with load.

MaxNet has also been shown to have desirable fairness and stability properties. The original MaxNet [10] yields max-min fairness for a network of homogeneous sources, or general weighted max-min fairness for heterogeneous 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 [13]. However, this approach loses the fairness of the original proposal [10]. The theoretical contribution of this paper is the addition of source dynamics adapted from [14] to achieve both stability and fairness. We call the resulting protocol MaxNet 2.0.

The practical contribution of this paper is an implementation of the protocol in the Linux kernel and the experimental evaluation of its properties on WAN-in-Lab [15]. The advantage of using WAN-in-Lab, a hardware testbed with real carrier-class networking hardware, is that it offers unprecedented realism in a laboratory setting. However, using such a complicated infrastructure necessarily implies our experiments are simpler than ones typically performed in software simulations.

Our kernel implementation of MaxNet consists of (i) a Linux router AQM module that marks packets with the explicit congestion signal, and (ii) a modification to the TCP end-hosts to control the congestion window in response to the signal.

After a description of the principles behind MaxNet in Section II, Section III describes the MaxNet algorithm and Section IV its implementation. Experimental results demonstrating the stability, fairness and convergence speed are presented in Section V. Section VI describes how to select provably-stable parameters that result in rapid convergence. Related protocols are analyzed in Section VII.

II. MaxNet Background

In this section we summarize the key features of MaxNet introduced in [10], [12], [13]. We 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

$$x_i(t) = D_i(q_i(t)), \tag{1}$$

where $D_i(\cdot)$ is called 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 [10] that MaxNet achieves max-min fairness. Weighted max-min fairness can be achieved by scaling the demand function.

Figure 1 illustrates that $q_i(t)$, the price communicated to the source, is the maximum of all link prices $p_l(t)$ on the source to destination path.

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

where L_i is the set of links on source i's path. A single price field in each packet header is used to communicate the maximum price. If link l's congestion price $p_l(t)$ exceeds the value q_j in the price field of packet j, the router corresponding to link l overwrites the price field with $p_l(t)$. The receiver echoes back the final value q_j in acknowledgements.

The AQM algorithm calculates the price signal of link l as

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

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 utilisation. Note that in equilibrium $y_l(t) = \mu_l C_l$, leaving $(1 - \mu_l)C_l$ of free capacity to absorb traffic bursts. This makes the buffer empty in equilibrium and results in very low network latency.

In [16] 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

$$\frac{\delta D_i}{\delta q_i} \geq -\frac{\alpha_i x_i}{M_i \tau_i}, \tag{4}$$

where M_i is the number of bottlenecks giving feedback to flow i, τ_i is the RTT of source i and $\alpha_i \in (0, \pi/2)$ is a
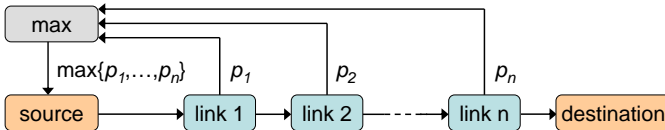

Fig. 1. Conceptual price communication scheme of MaxNet. The maximum of the prices $p_1, p_2, \cdots, p_n$ at links on the end-to-end path is fed back to the traffic source.

gain parameter. MaxNet reacts to the single most congested bottleneck, hence $M_i = 1$. Let $x_{\max}$ be the maximal supported rate for a source. As discussed in [14], condition (4) places a constraint on the demand function which is satisfied by

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

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 would not achieve max-min fairness. The dynamic source algorithm of [14] implements fairness on slow time scales, separate from the fast time scale response which determines stability. On a fast time scale the rate changes are bounded by (4) by setting

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

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:

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

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. MaxNet 2.0 Algorithm

This section describes the MaxNet 2.0 algorithm which consists of the source algorithm responsible for adjustment of source rates, the router algorithm responsible for congestion price calculation, and the MaxStart algorithm which allows new flows attain high transmission rates quickly.

A. Source algorithm

To achieve stability and weighted max-min fairness, the source algorithm is based on the dynamic controller (6) from [14]. The key choice in designing the controller is selecting a demand function. Consider the exponential demand function with a constant T

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

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

$$U_i'(x_i) = D^{-1}(x_i) = -T \log(x_i/x_{\max}) \tag{8}$$
$$= -T(\xi_i - q_i \alpha/\tau_i). \tag{9}$$

The resulting 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.

B. Router algorithm — 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

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

$$\xi \;\leftarrow\; \xi + \frac{\alpha\eta\,dt}{\tau^2}\left(\left(\frac{T\alpha}{\tau}-1\right)q - T\xi\right) \qquad (10)$$

$$W \;\leftarrow\; \tau x_{\max}\exp\left(\xi - \frac{q\alpha}{\tau}\right)$$

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	congestion window of the source
dt	interval since last update

Fig. 2. Pseudocode of the source algorithm.

1) Every dt_p seconds:

```
y_dt ← y_dt + Q/T0
p ← max(p + y_dt/Cl − μ dtp, 0)
y_dt ← 0
```

2) On packet arrival:

```
y_dt ← y_dt + packet.size
if p > packet.price then
  packet.price ← p
end if
```

Parameters:

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

Variables:

p	link price
y_dt	aggregate arrivals in update interval dt_p
Q	instantaneous queue length

Fig. 3. Pseudocode of the router algorithm.

single addition, comparison and assignment. The pseudo code of the router algorithm is shown in Figure 3.

The increment of y_dt by Q/T_0 may seem to deviate from the virtual queue (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 link carrying a single flow.

Under congestion, the rate the flow *seeks* to achieve, W/τ, exceeds the capacity C_l of the bottleneck, resulting in a physical queue. However, since MaxNet is a sliding window protocol, the packet arrival rate at the bottleneck link is limited by the "ACK clock" and is approximately equal to the bottleneck capacity; that is $y_l = C_l$. Nonetheless, we can use the queue size $Q = W - (y_l \times \tau)$ to estimate that the attempted sending rate of the source is $y_l + Q/\tau$, higher than the measured rate y_l.

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.

C. MaxStart — Replacing slow start

TCP Reno utilizes a slow start mechanism [1] that prevents excessive congestion when a new flow starts by increasing its rate exponentially. Similarly, explicit signalling protocols need a mechanism that controls the rates of newly arriving flows and prevents them starting at the same (high) rate as established flows. Conveniently, explicit signalling allows designs that solve this problem and scale faster than slow start. Inspired by QuickStart [17] which enables sources to determine the available sending rate, we introduce MaxStart.

MaxStart initiates a new flow at a rate equal to $1/4$ of the spare capacity of the most congested link on the path, and then ramps up linearly over two RTTs to the "price rate" (the rate corresponding to the advertised price). The initial rate is thus calculated as a minimum over all links of $(\mu+(1-\mu)/4)C_l - y_l$, where C_l is the link capacity, μ the target utilisation and y_l the link load. The initial rate is communicated similarly to the congestion price: the sender flags the first packet to indicate that it wants to be informed about the rate instead of the price, and the routers then mark the packet with the lowest spare capacity. MaxStart terminates as soon as the MaxStart rate exceeds the price rate. Until then, 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.

Fast ramp up time and linear increase of the sending rate are the main benefits of MaxStart. Short flows which only exist for a small number of RTTs increase their rates more rapidly and achieve a rate proportional to their fair share, rather than dependent only on their flow size. Moreover, when the equilibrium rate is reached, linear increase of rate causes less overshoot than exponential increase.

The algorithm could be further enhanced to ensure that capacity is not over allocated when multiple flows start at almost the same time. This can be done by keeping track of the already allocated capacity. Furthermore, rate allocations to all flows could be made equal when a new flow starts. This is, however, beyond the scope of this paper.

IV. IMPLEMENTATION

MaxNet 2.0 was implemented as a standalone TCP protocol in the Linux 2.6.11 kernel. We provide a description of the source and router code and details of the price communication scheme.

A. Communicating and Representing the Price

TCP options are used to carry the price. The MaxNet option format is depicted in Figure 4. As a packet propagates from the source to the destination, routers overwrite the price field with their calculated congestion price if the price advertised in the packet is lower. The echo field is used to return the price to the sender in ACKs. During MaxStart, the price field is used to communicate the desired transmission rate. The highest bit of the 24-bit price field is 0 if the remaining 23 bits contain a price or 1 if they contain a MaxStart rate.

The demand function and price encodings explicitly determine the range and precision of the achievable rates. Hence, the protocol must be designed to scale. MaxNet is able to scale over a large dynamic range with high precision. Let B_i and B_f be the number of bits allocated to the integer and fractional part of the price. 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 $\epsilon = 10^{-5}$, $B_f \geq \lceil -\log_2(T \log(1 + \epsilon)) \rceil = 18$. This analysis shows how to represent the price within the 23 available bits.

B. Implementation of the Source Algorithm

The main component of the sender code, the window calculation, is implemented in the `tcp_cong_avoid` function in `net/ipv4/tcp_input.c`. Parameters are implemented as system control variables and set using the `sysctl` interface to $x_{\max} = 10^{15}$ bit/s, $T = 0.4$ seconds, $\alpha = 0.66$, $\eta = 0.06$ and $dt_{\min} = 1 \, \mu$s to update on every ACK.

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.

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

C. Implementation of the Router Algorithm

The router code is implemented as an `iproute2` dynamically loadable module for Linux. The parameters $dt_p = 1$ ms and $T_0 = 130$ ms are set through the `tc` interface.

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).

V. Experiments

The performance of MaxNet 2.0 in two scenarios is described. The first demonstrates its fairness, convergence speed and scalability, and the second its response to cross traffic.

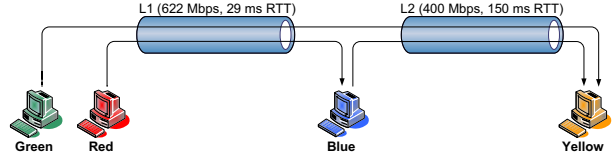| opt	optsize		
42	6	echo	price
(1 byte)	(1 byte)	(3 bytes)	(3 bytes)

Fig. 4.   MaxNet option format.

Fig. 5.   Topology for multi-flow experiments.

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 fairness, scalability, and behaviour 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 [15], and Link 2 is a 400 Mbit/s link with RTT 150 ms provided by a dummynet. The target utilisation was 90% ($\mu = 0.9$). Figure 6 shows when individual traffic flows start, dividing the experiment into six intervals.

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.
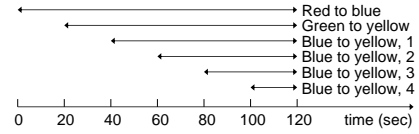

Fig. 6.   Start times and durations of flows for multi-flow experiments.
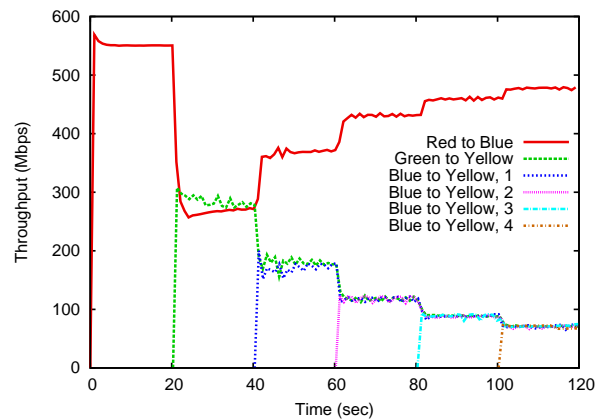

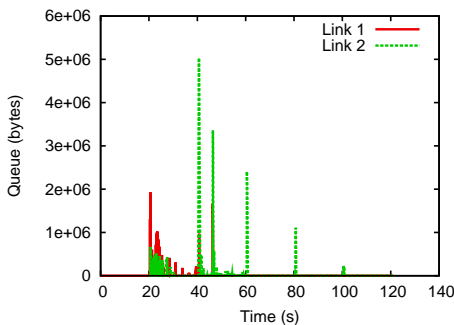Fig. 7.   Rates of flows in the two-hop experiment.

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

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

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, which is consistent with the nominal rise time of two RTTs.

2) Fairness: Reno is known to give significantly unfair rates to flows with different RTTs [18], 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 [3] and FAST [5] 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.

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 memory requirements of 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 $N > \mu/(1-\mu)$.

Note that the memory requirements of large multi-port routers with shared memory are governed by the *average* queue size, rather than the peak size, since memory can be statistically multiplexed between different ports. Thus isolated spikes occurring when a link is carrying few flows do not negate MaxNet's benefit of reducing buffering requirements.

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 fl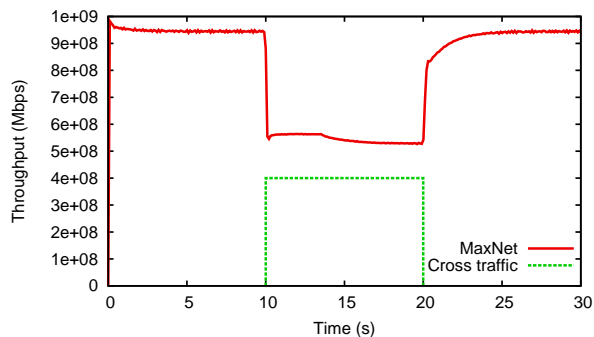ow from Green to Yellow switches from Link 1 to Link 2. Significantly, this does not cause instability in the form of "ping-ponging" between bottlenecks as the prices stabilise. However, it does result in the highest queueing in the experiment, 5 MByte or 90% of the bandwidth-delay product of the flow from Green to Yellow before the switch.

5) Increase and decrease in available bandwidth: As the load on 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 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.

VI. STABILITY AND TRANSIENTS

MaxNet 2.0's stability will now be proved using the theory developed in [14]. Subsection VI-D 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 [14], adapted to the case of MaxNet. These results apply to general multi-link networks with heterogeneous RTTs.

Stability in the presence of delay requires that the loop gain be sufficiently low. The loop gain is determined by the slope of

the demand function, which places restrictions on the family of (static) demand functions which are stable [14]. It was further shown that the stability of network using lead-lag controllers, such as MaxNet, can be determined from the function

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

where s is the complex frequency, τ_i is the RTT of the ith 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") 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. The triangulation approach of [13] shows that this result also applies to MaxNet.

The stability proof for MaxNet is based on the following result, which follows easily from the results of [14], [19]:

Lemma 1: Let $H(\omega; \alpha) = \mathrm{Co}\{F(j\omega, \tau_i; \alpha)\}$ 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 [14], the speed of dynamics was matched 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 ω, it is contained in a particular spiral which is bounded away from $-1 + 0j$.

B. Choice of demand function

The dynamics depend heavily on the ratio κ_i/ν_i. This depends on the demand function and the operating point.

The reason [14] added a lead-lag compensator was to ensure the stability when $\kappa_i/\nu_i < 1$. In this case, the resulting "lag-compensator" yields a resonance peak, causing very slowly decaying oscillations, which are unacceptable in practice. A key observation of this research is that the compensator provides "insurance" against extreme RTTs, but does not extend the range of practically feasible demand functions.

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

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

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.

In contrast to [14] which uses a lead-lag parameter, z, dependent on the largest RTT in the network, $z_i = \eta/\bar{\tau}$, the current implementation of MaxNet adapts z to each flow's own RTT, setting $z_i = \eta/\tau_i$. This yields

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

In this case the stability proof of [14] needs modification,

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 $\mathcal{T} = \{F(j\underline{\omega}, \tau; 1) : \tau < \bar{\tau}\}$. (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 $\alpha_{\max}$ 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.

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 $\omega < \underline{\omega}$. It is also possible to choose α small enough such that $H(\omega; \alpha)$ is strictly below a slanted line through $-1 + 0j$ for $\omega > \underline{\omega}$. The theoretical complication arising from adapting z to each flow's RTT is that, unlike in [14], $\underline{\omega} \neq 1/\bar{\tau}$.

Using this approach, it can be shown that MaxNet is stable for all RTTs up to $\tau = 1000$ seconds using the parameters of Section V, namely $T = 0.4$ s, $\alpha = 0.66$ and $\eta = 0.06$. If z were independent of τ_i as in [14], 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.

C. Determining stable parameters

The first step in choosing suitable parameters is finding the provably stable combinations. Following [14], the system will be designed to be stable for all RTTs $\tau < \bar{\tau}$.

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

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

to be the lowest frequency at which the spiral for τ crosses the real axis. 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 IV satisfy this proposition with $\mathcal{L}$ having slope $\gamma = 0.3504$, 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
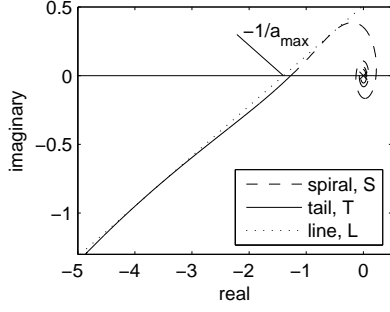
Fig. 10. Spiral $\mathcal{S}$, Tail $\mathcal{T}$, and line $\mathcal{L}$ together with resulting α_{max} for the illustrative case of $\bar{\tau} = 3\,\text{s}$ $\eta = 0.7$ and $\alpha T = 0.1\,\text{s}$.

(a) $\mathcal{T}$, $\tau \in (0.006, 0.6)$. Curves truncated to $\omega > \underline{\omega}$. Note the curves are within $\mathcal{T}$ in the bottom left quadrant.

(b) $\mathcal{S}, \tau \in (0.001, 10 = \bar{\tau})$. Dotted line completes the convex hull. Note curves "in the spiral" are within $\mathcal{S}$.

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

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 [20].

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. □

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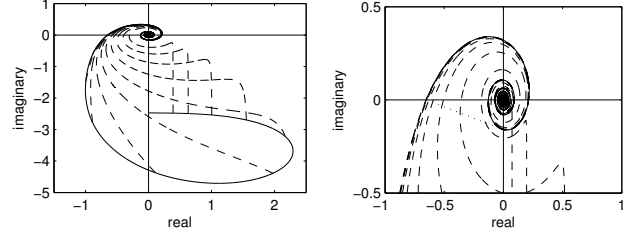
For any ω and τ in the tail, $F(j\omega, \tau; 1)$ is below the real axis; this follows from $\lim_{\omega \to 0} \arg(F(j\omega, \tau; 1)) = -\pi/2$, the continuity of F and the definition of the tail.

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), for all $\tau < \bar{\tau}$, $\omega < \omega_0(\tau)$ whence ω and τ are in the tail. Thus $F(j\omega, \tau; 1)$ 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 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). ∎

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). ∎

D. Parameters for rapid convergence

The parameters used in Section V 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 $\bar{\tau}$.

VII. Comparison with other Protocols

In this section we compare MaxNet with other prominent explicit signalling protocols: XCP, RCP and JetMax. As none of the protocols appear to have released implementations capable of operating at 1 Gbit/s, we do not perform experimental comparisons.

XCP and MaxNet differ in several regards. XCP only achieves constrained max-min fairness [21] where sources may claim an arbitrarily small fraction of the max-min fair rates, contrasting to MaxNet's bound of μ_l. Furthermore, linear stability of XCP has only been proven for a single link with sources of homogeneous RTTs, and [22] indicates that XCP can exhibit oscillatory behaviour under more diverse circumstances. In this paper we prove the linear stability of MaxNet for arbitrary network topologies. In terms of implementation, XCP is more complicated, requiring 12 operations per packet at the router compared to 2 for MaxNet, and 16 bytes compared to 6 bytes in the packet header.

RCP [8] has a similar structure to MaxNet, but differs in how equilibrium queues are avoided. 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

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

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 $R(t) = e^{-\alpha \cdot p(t)/\tau}$ 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 RTTs. In RCP, this is done by the routers based on the traffic-weighted average RTT advertised in the packets.

The drawback of MaxNet's approach is that a global parameter, αT, must be set to ensure acceptable performance 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. This is described in [20].

In JetMax [9], 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 estimates which flows are "responsive".

VIII. 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 2.0 in the realistic environment of WAN-in-Lab confirmed that it can achieve the above goals.

IX. Acknowledgment

References

[1] V. Jacobson, "Berkeley TCP evolution from 4.3-tahoe to 4.3-reno," in *Proc. 18th Internet Engineering Task Force*, Aug. 1990.

[2] I. Rhee and L. Xu, "CUBIC: A new TCP-friendly high-speed TCP variant," in *Proc. PFLDnet*, Feb. 2005.

[3] D. Leith and R. Shorten, "H-TCP: TCP for high-speed and long-distance networks," in *Proc. PFLDnet*, Feb. 2004.

[4] L. Brakmo, S. O'Malley, and L. Peterson, "TCP Vegas: new techniques for congestion detection and avoidance," in *Proc. SIGCOMM*, Sep. 1994.

[5] C. Jin, D. Wei, and S. Low, "FAST TCP: motivation, architecture, algorithms, performance," in *Proc. IEEE Infocom*, Mar. 2004.

[6] K. Ramakrishnan and S. Floyd, "Proposal to add explicit congestion notification (ECN) to IP," RFC 2481, Jan. 1999.

[7] D. Katabi, M. Handley, and C. Rohrs, "Congestion control for high bandwidth-delay product networks," in *Proc. SIGCOMM*, Oct. 2002.

[8] N. Dukkipati and N. McKeown, "Why flow-completion time is the right metric for congestion control," *ACM SIGCOMM Computer Communication Review*, vol. 36, pp. 59–62, Jan. 2006.

[9] Y. Zhang, D. Leonard, and D. Loguinov, "JetMax: Scalable max-min congestion control for high-speed heterogeneous networks," in *Proc. IEEE Infocom*, Apr. 2006.

[10] B. Wydrowski and M. Zukerman, "MaxNet: A congestion control architecture for MaxMin fairness," *IEEE Commun. Lett.*, vol. 6, pp. 512–514, Nov. 2002.

[11] F. Kelly, "Charging and rate control for elastic traffic," *European Transactions on Telecommunications*, vol. 8, pp. 33–37, Jan. 1997.

[12] B. Wydrowski, L. Andrew, and I. Mareels, "MaxNet: Faster flow control convergence," in *Proc. Networking*, 2004, Springer Lecture Notes in Computer Science LNCS 3042.

[13] B. Wydrowski, L. Andrew, and M. Zukerman, "MaxNet: A congestion control architecture for scalable networks," *IEEE Commun. Lett.*, vol. 7, pp. 511–513, Oct. 2003.

[14] 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, pp. 43–56, Feb. 2005.

[15] G. Lee, L. Andrew, A. Tang, and S. Low, "WAN-in-Lab: Motivation, deployment and experiments," in *PFLDnet*, 2007.

[16] F. Paganini, J. Doyle, and S. Low, "Scalable laws for stable network congestion control," in *Proc. IEEE CDC*, Dec. 2001.

[17] A. Jain, S. Floyd, M. Allman, and P. Sarolahti, "Quick-start for TCP and IP," Internet draft, Jul. 2006.

[18] T. 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, pp. 336–350, Jun. 1997.

[19] 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.

[20] L. L. Andrew, K. Jacobsson, S. H. Low, M. Suchara, R. Witt, and B. P. Wydrowski, "MaxNet: Theory and implementation." [Online]. Available: http://netlab.caltech.edu/maxnet/MaxNet_Implementation_TechReport.pdf

[21] S. Low, L. Andrew, and B. Wydrowski, "Understanding XCP: Equilibrium and fairness," in *Proc. IEEE Infocom*, Mar. 2005.

[22] L. Andrew, B. Wydrowski, and S. Low, "An example of instability in XCP." [Online]. Available: http://netlab.caltech.edu/˜lachlan/abstract/xcpInstability.pdf