H-TCP: TCP for high-speed and long-distance networks

D. Leith*, R. Shorten*
Hamilton Institute, NUI Maynooth

Abstract

In this paper we present a congestion control protocol that is suitable for deployment in high-speed and long-distance networks. The new protocol, H-TCP, is shown to fair when deployed in homogeneous networks, to be friendly when competing with conventional TCP sources, to rapidly respond to bandwidth as it becomes available, and to utilise link bandwidth in an efficient manner. Further, when deployed in conventional networks, H-TCP behaves as a conventional TCP-variant.

1 Introduction

It is generally accepted that future communication and computer networks will be characterised by high-speed and long-distance connectivity, and by the requirement to carry a wide variety of network services and traffic types. These demands create new challenges for network designers and researchers. Clearly, the problem of designing future networks may be addressed by a joint optimization of link-layer, transport layer and application layer technologies. Unfortunately, the option to completely redesign networks with a view to such a joint optimization is not feasible due to a strict backward compatibility constraint; namely, that any new algorithms designed to operate in future networking environments must also operate in existing and older network types in a way that co-exists with existing and older transport protocols and supports incremental rollout. The constraint of backward compatibility is particularly severe in the transport layer and it is the design of transport layer protocols, in particular TCP, that is the principal concern of this paper. It is widely recognised that transport layer enhancements are essential if high performance next generation networks are to be realised [1]. Our objective here is to develop a systematic framework for modifying the basic TCP algorithm that renders it suitable in a variety of network types. In this paper we report an important first step in this direction. We describe a new TCP-variant that is suitable for deployment in high speed and long distance networks, as well as conventional networks. The new TCP variant, H-TCP, is shown to be fair when deployed in homogeneous networks, to be friendly when competing with conventional TCP sources, to rapidly respond to changes in available bandwidth, and to utilise link bandwidth efficiently. Further, H-TCP, is shown to behave as a conventional TCP-variant when deployed on conventional network types.

This paper is structured as follows. In Section 2 we develop a positive systems network model that captures the essential features of communication networks employing drop-tail queuing and AIMD congestion control algorithms. In Section 3 we use the insights gained from the analysis of the dynamic properties of this model to develop H-TCP.

*Joint first author

2 Nonnegative matrices and communication networks

A communication network consists of a number of sources and sinks connected together via links and routers. We assume that these links can be modelled as a constant propagation delay together with a queue, that the queue is operating according to a drop-tail discipline, and that all of the sources are operating a TCP-like congestion control algorithm. The links and queues along a network path form a 'pipe' that contain packets in flight. TCP operates a window based congestion control algorithm. The TCP standard defines a variable *cwnd* called the congestion window. Each source uses this variable to determine the number of packets that can be in transit, but not yet acknowledged, at any time. When the window size is exhausted, the source must wait for an acknowledgement before sending a new packet. Congestion control is achieved by dynamically adapting the window size according to an additive-increase multiplicative-decrease (AIMD) law. The basic idea is for a source to gently probe the network for spare capacity and rapidly back-off the number of packets transmitted through the network when congestion is detected, as depicted in Figure 7. Each source is parameterized by an additive increase parameter and a multiplicative decrease factor, denoted α_i and β_i respectively. These parameters satisfy $\alpha_i \geq 1$ and $0 < \beta_i < 1 \ \forall i \in \{1, ..., n\}$.

It is informative to begin our discussion by considering networks for which the following assumptions are valid: (i) at each congestion event every source experiences a packet drop i.e. the drops are synchronised; and (ii) each source has the same round-trip-time (RTT)[1]. In this case an exact model of the network dynamics may be found using elementary algebra. Let $w_i(k)$ denote the congestion window size of source

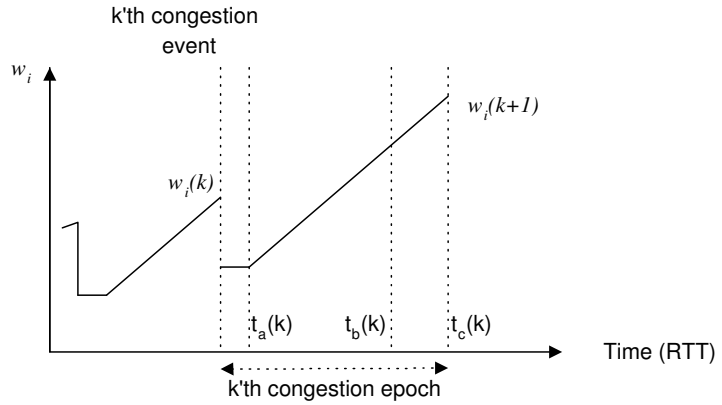

Figure 1: Evolution of window size

i immediately before the kth network congestion event is detected by the source. Over the kth congestion epoch three important events can be discerned: $t_a(k), t_b(k)$ and $t_c(k)$ in Figure 1. The time $t_a(k)$ denotes the instant at which the number of unacknowledged packets in the pipe equals $\beta_i w_i(k)$; $t_b(k)$ is the time at which the pipe is full; and $t_c(k)$ is the time at which packet drop is detected by the sources, where time is measured in units of RTT. It follows from the definition of the AIMD algorithm that the window evolution is completely defined over all time instants by knowledge of the $w_i(k)$ and the event times $t_a(k), t_b(k)$ and $t_c(k)$ of each congestion epoch. We therefore only need to investigate the behaviour of

[1]One RTT is the time between sending a packet and receiving the corresponding acknowledgement when there are no packet drops.

these quantities.

We have that $t_c(k) - t_b(k) = 1$; namely, each source is informed of congestion exactly one RTT after the first dropped packet was transmitted. Also,

$$w_i(k) \geq 0, \sum_{i=1}^{n} w_i(k) = P + \sum_{i=1}^{n} \alpha_i, \ \forall k > 0, \tag{1}$$

where P is the maximum number of packets which can be held in the pipe; this is usually equal to $q_{max} + BT_d$ where q_{max} is the maximum queue length of the congested link, B is the service rate of the congested link in packets per second and T_d is the round-trip time when the queue is empty. At the $(k+1)$th congestion event

$$w_i(k+1) \quad = \quad \beta_i w_i(k) + \alpha_i [t_c(k) - t_a(k)]. \tag{2}$$

and

$$t_c(k) - t_a(k) \quad = \quad \frac{1}{\sum_{i=1}^{n} \alpha_i} [P - \sum_{i=1}^{n} \beta_i w_i(k)] + 1. \tag{3}$$

Hence, it follows that

$$w_i(k+1) \quad = \quad \beta_i w_i(k) + \frac{\alpha_i}{\sum_{j=1}^{n} \alpha_i} [\sum_{i=1}^{n} (1 - \beta_i) w_i(k)], \tag{4}$$

and that the dynamics an entire network of such sources is given by

$$W(k+1) = AW(k), \tag{5}$$

where $W^T(k) = [w_1(k), \cdots, w_n(k)]$, and

$$A \quad = \quad \begin{bmatrix} \beta_1 & 0 & \cdots & 0 \\ 0 & \beta_2 & 0 & 0 \\ \vdots & 0 & \ddots & 0 \\ 0 & 0 & \cdots & \beta_n \end{bmatrix} + \frac{1}{\sum_{j=1}^{n} \alpha_i} \begin{bmatrix} \alpha_1 \\ \alpha_2 \\ \cdots \\ \alpha_n \end{bmatrix} \begin{bmatrix} 1-\beta_1 & 1-\beta_2 & \cdots & 1-\beta_n \end{bmatrix}. \tag{6}$$

The matrix A is a positive matrix (all the entries are positive real numbers) and it follows that the synchronised network (5) is a positive linear system [2]. Many results are known for positive matrices and we exploit some of these to analyse the properties of synchronised communication networks. In particular, from the viewpoint of designing communication networks the following properties are very important: (i) network fairness and TCP-friendliness; (ii) network convergence; (iii) network responsiveness; and (iv) throughput efficiency. Roughly speaking, window or pipe fairness refers to a steady state situation where n sources operating $AIMD$ algorithms have an equal number of packets P/n in flight at each congestion event; convergence refers to the existence of a unique fixed point to which the network dynamics converge; responsiveness refers to the rate at which the network converges to the fixed point; and throughput efficiency refers to the objective that the network operates at the bottleneck-link capacity. It is shown in [3, 4] that these properties can be deduced from the network matrix A. We briefly summarise here the relevant results in these papers.

Theorem 2.1 *[4] Let A be defined as in Equation (6). Then, a Perron eigenvector of A is given by $x_p^T = [\frac{\alpha_1}{1-\beta_1}, ..., \frac{\alpha_n}{1-\beta_n}]$.*

The following corollary follows from Theorem 2.1 and properties of non-negative matrices [5, 2].

Corollary 2.1 *[4] For a network of synchronised time-invariant AIMD sources: (i) the network has a Perron eigenvector* $x_p^T = [\frac{\alpha_1}{1-\beta_1}, ..., \frac{\alpha_n}{1-\beta_n}]$*; and (ii) the Perron eigenvalue is* $\rho(A) = 1$*. All other eigenvalues of A satisfy* $|\lambda_i(A)| < \rho(A)$*. The network converges to a unique stationary point* $W_{ss} = \Theta x_p$*, where* Θ *is a positive constant such that the constraint (1) is satisfied;* $\lim_{k\to\infty} W(k) = \Theta x_p$*, and the rate of convergence of the network to* W_{ss} *is bounded by the second largest eigenvalue of A (*$max|\lambda|, \lambda \neq 1 \in$ *spec(A)).*

The following facts may be deduced from the above discussion.

(i) **Fairness and friendliness:** Window fairness is achieved when the Perron eigenvector x_p is a scalar multiple of the vector $[1, ..., 1]$; that is, when $\frac{\alpha_i}{1-\beta_i}$ is a constant that does not depend on i. Further, since it follows for conventional TCP-flows that $\alpha = 2(1-\beta)$, any new protocol operating an AIMD variant that satisfies $\alpha_i = 2(1-\beta_i)$ will be both fair and TCP-friendly. See for example Figure 2

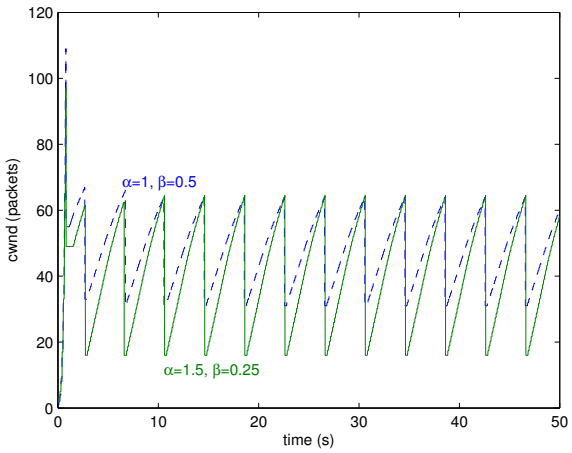

Figure 2: Example of window fairness between two TCP sources with different increase and decrease parameters (NS simulation, network parameters: 10Mb bottleneck link, 100ms delay, queue 40 packets.

(ii) **Network responsiveness:** The second largest eigenvalue λ_{n-1} of the matrix A bounds the convergence properties of the entire network. We show in [4] that the network rise-time when measured in number of congestion epochs is bounded by $n_r = \frac{log(0.95)}{log(\lambda_{n-1})}$. With $\beta_i = 0.5$ for all i, $n_r \approx 4$; see for example Figure 2. Note that n_r gives the number of congestion epochs until the network dynamics have converged to 95 % of the final network state: the actual time to reach this state depends on the length of the congestion epochs which is ultimately dependent on the α_i. It is shown in [4] that all the eigenvalues of A are real and positive and lie in the interval $[\beta_1, 1]$, where the β_i are ordered as $0 < \beta_1 \leq \beta_2 \leq .... \leq \beta_{n-1} \leq \beta_n < 1$. In particular, the second largest eigenvalue is bounded by $\beta_{n-1} \leq \lambda_{n-1} \leq \beta_n$. Fast convergence to the equilibrium state (the Perron eigenvector) is guaranteed if the largest backoff factor in the network is small.

(iii) **Network throughput :** At a congestion event the network bottleneck is operating at link capacity

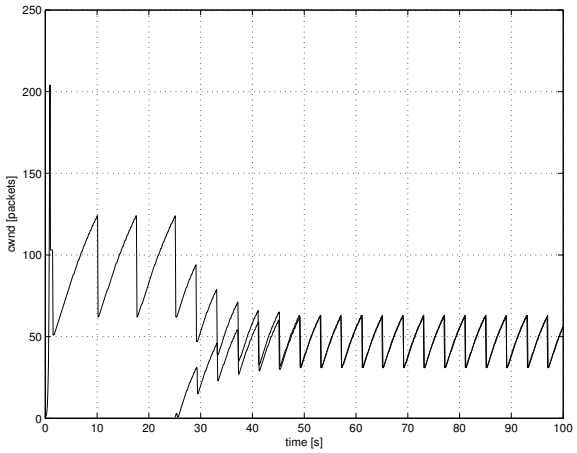

Figure 3: NS packet-level simulation ($\alpha_i = 1$, $\beta_i = 0.5$, dumb-bell with 10Mbs bottleneck bandwidth, 100ms propagation delay, 40 packet queue).

and the total data throughput through the link is given by

$$R(k)^- = \frac{\sum_i^n w_i(k)}{T_d + \frac{q_{max}}{B}} \tag{7}$$

where B is the link capacity, q_{max} is the bottleneck buffer size, T_d is the RTT when the bottleneck queue is empty and $T_d + q_{max}/B$ is the round-rip time when the queue is full. After backoff, the data throughput through the link is given by

$$R(k)^+ = \frac{\sum_i^n \beta_i w_i(k)}{T_d} \tag{8}$$

under the assumption that the bottleneck buffer empties. Evidently, if the sources backoff too much, data throughput will suffer as the link operates below its maximum rate and the queue remains empty for a period of time. A simple method to ensure maximum throughput is to equate both rates yielding the following equation for the β_i:

$$\beta_i = \frac{T_d}{T_d + \frac{q_{max}}{B}} = \frac{RTT_{min}}{RTT_{max}}. \tag{9}$$

(iv) **Maintaining fairness :** Note that setting $\beta_i = \frac{RTT_{min}}{RTT_{max}}$ requires a corresponding adjustment of α_i if it is not to result in unfairness. Both network fairness and TCP-friendliness are ensured by adjusting α_i according to $\alpha_i = 2(1 - \beta_i)$).

2.1 Models of unsynchronised network

The objective of the preceding discussion is to illustrate that important network properties may be related to the properties of certain positive matrices. Unfortunately, the assumptions under which this model was derived, namely of source synchronisation and uniform RTT, are extremely restrictive (although they may be valid in many long-distance networks). It is therefore of great interest to extend our approach to more general network conditions.

Consider the general case of a number of sources competing for shared bandwidth in a generic dumbbell topology (where sources may have different round-trip times and drops need not be synchronised). The evolution of the *cwnd* of a typical source as a function of time, over the $k'th$ congestion epoch, is depicted in Figure 4.

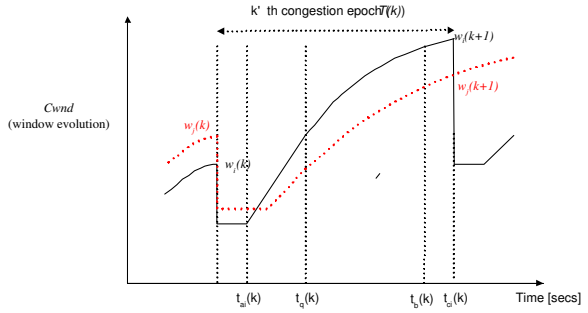

Figure 4: Evolution of window size over a congestion epoch. $T(k)$ is the length of the congestion epoch in seconds.

As before a number of important events may be discerned; $t_{ai}(k)$ is the time at which the number of packets in flight belonging to source i is equal to $\beta_i w_i(k)$; $t_q(k)$ is the time at which the bottleneck queue begins to fill; $t_b(k)$ is the time at which the bottleneck queue is full; and $t_{ci}(k)$ is the time at which the i'th source is informed of congestion. Note that the evolution of the i'th window size is not linear after t_q. This is due to the fact that the RTT of the i'th source increases according to $RTT_i = T_{d_i} + \frac{q_{max}}{B}$ after t_q where T_{d_i} is the RTT of source i when the bottleneck queue is empty. Note also that we do not assume that every source experiences a drop when congestion occurs. For example, a situation is depicted in Figure 4 where the i'th source experiences congestion at the end of the epoch whereas the j'th source does not.

Given these general features it is clear that the modelling task is more involved than in the synchronised case. While this is certainly the case, it is possible to relate $w_i(k)$ and $w_i(k+1)$ using a similar approach to the synchronised case as follows.

(i) Non-uniform RTT : The evolution of the i'th window w_i does not evolve linearly with time. However, we may relate $w_i(k)$ and $w_i(k+1)$ linearly by defining the average rate $\bar{\alpha}_i$ over the k'th congestion epoch:

$$\bar{\alpha}_i(k) = \frac{w_i(k+1) - \beta_i w(k)}{T(k)}, \tag{10}$$

where $T(k)$ is the duration of the k'th epoch; namely, $w_i(k+1) = \beta_i w_i(k) + \bar{\alpha}_i(k)T(k)$.

(ii) Unsynchronised source drops : We may account for the effect of unsynchronised behaviour as follows. Consider again the situation depicted in Figure 4. Here, the i'th source experiences congestion at the end of the epoch whereas the j'th source does not. This corresponds to the i'th source reducing its window variable to $\beta_i w_i(k+1)$ after the $k+1$'th congestion event, and the j'th source not adjusting its window size at the congestion event. This may be modelled by allowing the back-off factor of the i'th source to take one of two values at the k'th congestion event:

$$\bar{\beta}_i(k) \quad \in \quad \{\beta_i, 1\} \tag{11}$$

corresponding to whether the source experienced a packet loss or not.

Then by proceeding as described in the previous discussion one obtains the following description of the network dynamics

$$W(k+1) = A(k)W(k),\ A(k) \in \mathbb{R}^{n\times n}, \tag{12}$$

where the time between congestion events is now measured in seconds rather than number of RTT's. The matrix $A(k)$ takes the form of (6) with α_i and β_i replaced with $\bar{\alpha}_i(k)$ and $\bar{\beta}_i(k)$ respectively. An important simplification occurs when $q_{max} << BT_{d_i}\ \forall\ i$. In this case, the average $\bar{\alpha}_i$ are (almost) independent of k and given by $\bar{\alpha}_i \approx \frac{\alpha_i}{T_{p_i}}$. This situation corresponds to the practically important case of a network whose buffer is small compared with the delay-bandwidth product for all sources utilising the congested link. Such conditions prevail on a variety of networks; for example networks with large delay-bandwidth products, and networks where large jitter and/or latency cannot be tolerated. Then, Equation (12) reduces to

$$W(k+1) = A(k)W(k),\ A(k) \in \mathcal{A} = \{A_1, ..., A_m\},\ , A_i \in \mathbb{R}^{n\times n},\ \ m = 2^n - 1, \tag{13}$$

where

$$A_1 \quad = \quad \begin{bmatrix} \beta_1 & 0 & \cdots & 0 \\ 0 & \beta_2 & 0 & 0 \\ \vdots & 0 & \ddots & 0 \\ 0 & 0 & \cdots & \beta_n \end{bmatrix} + \frac{1}{\sum_{j=1}^{n} \bar{\alpha}_i} \begin{bmatrix} \bar{\alpha}_1 \\ \bar{\alpha}_2 \\ \cdots \\ \bar{\alpha}_n \end{bmatrix} \begin{bmatrix} 1-\beta_1 & 1-\beta_2 & \cdots & 1-\beta_n \end{bmatrix}. \tag{14}$$

as in the case of synchronised networks. The non-negative matrices $A_2, .., A_m$ are constructed by taking the matrix A_1 and setting some, but not all, of the β_i to 1. This gives rise to $m = 2^n - 1$ unique matrices associated with the system (13) corresponding to the different combinations of source drops that are possible.

We have from (13) that $W(k) = \Pi_k W(0)$ where

$$\Pi_k = A(k)A(k-1)....A(0). \tag{15}$$

The evolution of the vector of window sizes is governed by the asymptotic properties of the matrix product Π_k as $k \to \infty$. Consequently, the asymptotic behaviour of this product also determines the network fairness, convergence, responsiveness and throughput efficiency properties. While it can be immediately seen than the unsynchronised case is considerably more difficult to analyse that the synchronised case, it is shown in [6] that for the system (13) the structural properties of the matrices in $\mathcal{A}$ make the product Π_k amenable to study. Specifically, assuming sufficient randomization of drops (induced for example by a small amount of background web traffic; see [6] for details), it may be shown that the unsynchronised network (13) exhibits the same qualitative features as the synchronised system (5). In particular, we show in [6] that under the assumption that the probability that $A(k) = A_i \in \mathcal{A}$ is independent of k and equals ρ, then the qualitative properties of (13) are identical to (5); namely that the the empirical mean of the source congestion windows converges to a fixed point; and that this fixed point is fair if $\bar{\alpha}_i = k(1-\beta_i)$ for all i (TCP fairness corresponds to k =2); and finally that the bottleneck link will be used efficiently provided $\beta_i = \frac{RTT_{min,i}}{RTT_{max,i}}$. Full details of these results can be found in [6].

3 Protocols for high-speed and long distance networks

Recently, the design of congestion control protocols for deployment in high speed and long distance networks has been the subject of much interest in the networking community [7, 8, 9]. This interest

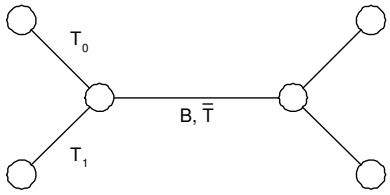

Figure 5: Dumbbell topology used in Figure 6.

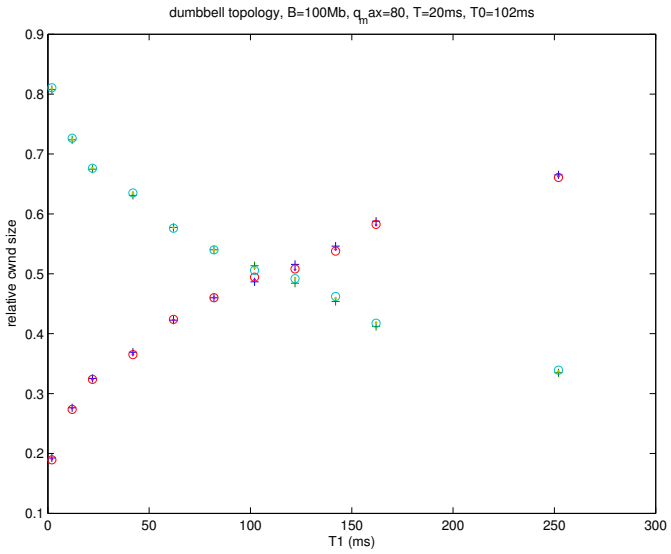

Figure 6: Asymptotic behaviour of the empirical mean of $W_i(k)$: Key: + NS simulation result; · prediction of unsynchronised model (13); ∘ analytic prediction.

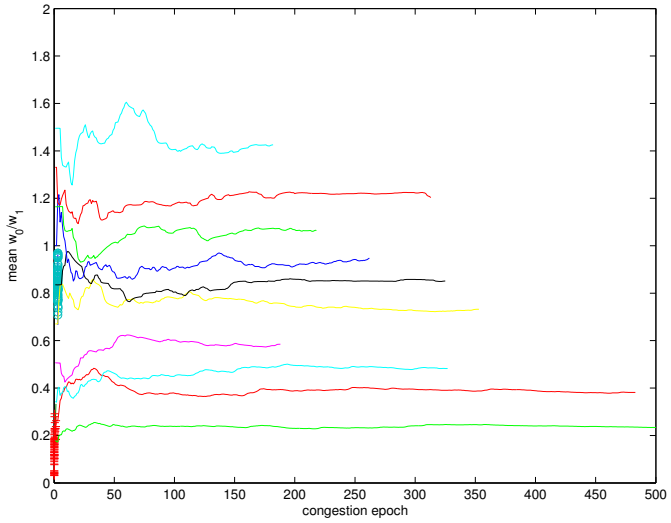

Figure 7: Convergence of the empirical mean of the window size to asymptotic values.

stems from the fact that the conventional TCP congestion control algorithm is ineffective in networks where window sizes may become very large. In these networks, following a congestion event, it may take an excessively long time for a source window size to recover. This leads to slow network convergence properties and poor bandwidth utilisation in links whose queues are small compared with the delay-bandwidth products as seen by sources served by the congested link. It is therefore essential to revise the TCP congestion control algorithm to operate efficiently in such environments. This task is non-trivial due to the backward-compatibility constraints discussed in the Section 1, and due to a number of performance related constraints. In particular, it is desirable that any new protocol exhibit the following features.

(i) High speed protocols should behave as a conventional TCP-variant when deployed in low-speed/short-distance networks.

(ii) High speed protocols should be TCP friendly; that is, should not completely starve TCP flows of available bandwidth when competing on high speed links

(iii) High speed protocols should be fair in some suitable sense. For example, high-speed sources competing against each other should on average have an approximately equal number of packets in flight in the network at each congestion event. The extension of our work to design for this and other types of fairness can be achieved with minor modifications to our analysis and algorithms.

(iv) High speed protocols sources should be responsive. That is, they should respond quickly to changes in available bandwidth (following start-up or death of a network flow, or in response to other network disturbances).

(v) High speed protocols sources should ensure that the bottleneck link is being used efficiently at all times.

In the remainder of this section we demonstrate that H-TCP realises all of these design objectives.

3.1 H-TCP

Several approaches have been proposed for designing protocols for high-speed and long distance networks [7, 8, 9] ranging from minor modifications to conventional TCP, to a complete protocol redesign. Our approach belongs to the former category and represents an evolution of conventional TCP rather than a radical departure from it. Our motivation for adopting this approach is twofold: (i) TCP has proved to be remarkably effective and robust in regulating network congestion and it seems sensible to retain as many aspects of TCP as possible; and (ii) it seems likely that TCP will continue to be deployed in a variety of networks into the future and any new protocol should therefore both co-exist and be backward compatible with conventional TCP.

Our design, referred to as H-TCP, is motivated by the simple observation that the α_i should be small is conventional networks (for backward compatibility) and large in high-speed and long distance networks (for short duration congestion epochs even with large pipe sizes). We therefore concentrate on modifying the basic TCP paradigm by adjusting the rate α_i at which a source inserts packets into a network to reflect the prevailing network conditions. This is similar to the work advocated by Floyd and Kelly in [7, 8]. The key innovative idea in our approach is to make the α_i increase as a function of the time elapsed since the last packet drop experienced by the i'th source.

Specifically, H-TCP amends conventional TCP in the following manner. In the high-speed mode the increase function of source i is $\alpha_i^H(\Delta_i)$ and in the low-speed mode α_i^L. The mode switch is governed by:

$$\alpha_i \quad = \quad \begin{cases} \alpha_i^L & \Delta_i \leq \Delta^L \\ \alpha_i^H(\Delta_i) & \Delta_i \geq \Delta^L \end{cases} \tag{16}$$

where Δ_i is the time elapsed since the last congestion event experienced by the ith source, α_i^L is the increase parameter for the low-speed regime (unity for backward compatibility), $\alpha_i^H(\Delta_i)$ is the increase function for the high-speed regime, β_i is the decrease parameter as usual and Δ^L is the threshold for switching from the low to high speed regimes.

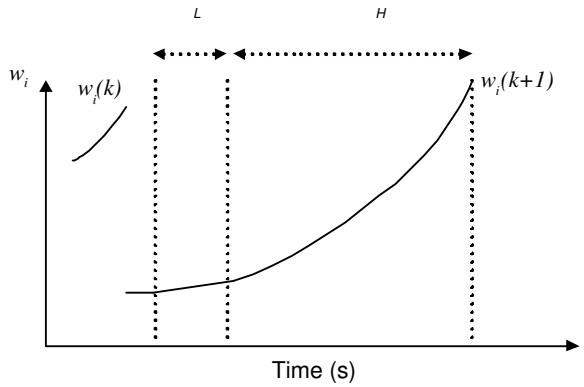

Figure 8: Evolution of window size

The increase function α_i^H is a design parameter that can be chosen according to desired objectives. In the rest of the present paper we set α_i^H according to:

$$\alpha_i^H(\Delta_i) \quad = \quad 1 + 10(\Delta_i - \Delta^L) + (\frac{\Delta_i - \Delta^L}{2})^2. \tag{17}$$

This choice of α_i^H yields a response function similar to that of HS-TCP [7]. In terms of the congestion epoch duration for large pipe sizes, the impact of increasing α in this manner is evident from Figure 9. A typical window evolution time history is illustrated in Figure 8. This approach has several advantages over evolving the α_i as a function of w_i as advocated in [7]. Firstly, the function governing the rate at which α_i is increased can be tuned to ensure that H-TCP operates as standard TCP in conventional networks where the time between successive congestion events is small, and to evolve more aggressively in high speed and long-distance networks where the time between congestion events may be long. We use a simple mode switch to guarantee that H-TCP operates as a conventional TCP variant for a short period after every congestion event. This guarantees both backward compatibility on low speed networks, and TCP-friendliness when deployed in high-speed networks. Secondly, because the mode switch is based on time since the last back-off, the sources behave symmetrically; that is, sources already in high speed mode do not gain a long term advantage over new flows starting up. This maintains symmetry in the network thereby guaranteeing fairness with other H-TCP sources.

Comment 1: We note that H-TCP is not an AIMD congestion control strategy. Nevertheless, by defining an effective linear $\bar{\alpha}_i$ for each source,

$$\bar{\alpha}_i(k) = \frac{w_i(k+1) - \beta_i(k)w(k)}{T(k)}, \tag{18}$$

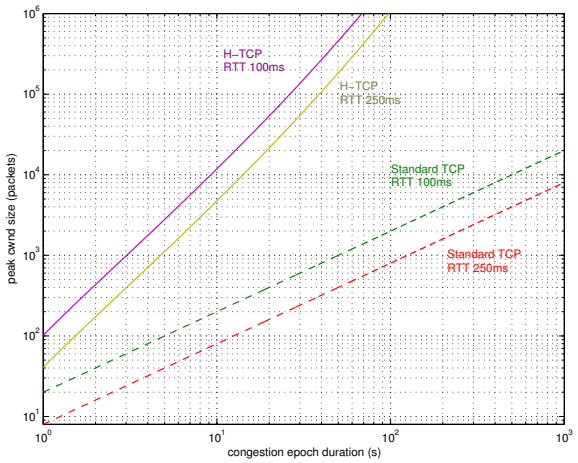

Figure 9: Peak window size achieved vs duration of congestion epoch with standard TCP and with H-TCP

where $T(k)$ is the duration of the k'th epoch, the behaviour of a network of H-TCP sources may be modelled in exactly the same manner as in Section 2. See for example, Figures 10-13.

Comment 2: Recall that for standard TCP we have that the effective increase rate is inversely proportional to round-trip time, $\bar{\alpha}_i \approx \frac{\alpha_i}{T_{p_i}}$. A similar situation holds for the high-speed mode switch (17). In both cases, we note that $\bar{\alpha}_i$ can be effectively made invariant with round-trip time by simply scaling α_i by the respective round-trip time T_{p_i}. With such scaling[2], the congestion epoch duration (see Figure 9) also becomes invariant with round-trip time. Combining this observation with the convergence results above that establish the convergence rate in terms of number of congestion epochs, it then becomes an option to specify a required convergence time in seconds that is independent on round-trip time.

3.2 Adaptation to achieve efficient bandwidth utilisation

In standard TCP congestion control the AIMD parameters are set as follows: $\alpha_i = 1$ and $\beta_i = 0.5$. These choices are reasonable when the maximum queue size in the bottleneck buffer is equal to the delay-bandwidth product, and backing off by a half should allow the buffer to just empty. However, is is generally impractical to provision a network in this way; for example, when each flow sharing a common bottleneck link has a different round-trip time. Moreover, in high-speed networks large buffers are problematic for both technical as well as cost reasons. When the queue sizes is small, the effect of backing off by 0.5 can lead to the queue being empty for a significant period of time and thereby to an under utilisation of the bottleneck link. An example showing this effect is given in Figure 12. The solution is an adaptive backoff mechanism that exploits the following observation. At congestion the network bottleneck is operating at link capacity and the total data throughput through the link is given by

$$R(k)^- = \sum_i^n \frac{w_i(k)}{RTT_{max,i}} \tag{19}$$

[2]It is of course prudent to restrict such scaling to lie in some interval, say [0.5,10], to prevent misbehaviour on paths with very short or very long round-trip times.

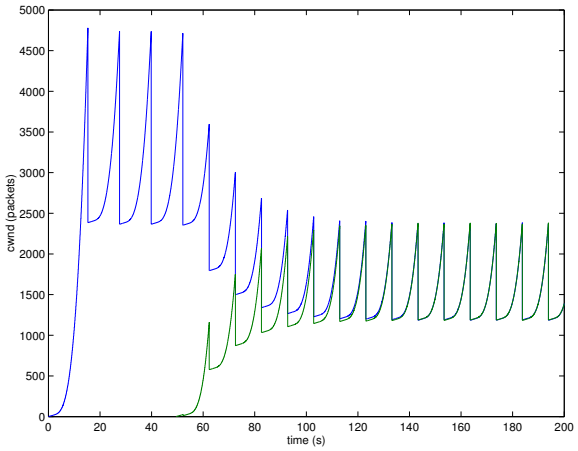

Figure 10: Example of two H-TCP flows illustrating rapid convergence to fairness - taking approximately 4 congestion epochs which is in agreement with the rise-time analysis for $\beta_i = 0.5$ (NS simulation, network parameters: 500Mb bottleneck link, 100ms delay, queue 500 packets).

where B is the link capacity, n is the number of network sources, and $RTT_{max,i}$ is the maximum RTT experienced by the i'th source. After backoff, the data throughput through the link is given by

$$R(k)^+ = \sum_i^n \frac{\beta_i w_i(k)}{RTT_{min,i}} \tag{20}$$

under the assumption that the bottleneck buffer empties. Clearly, if the sources backoff too much, data throughput will suffer. A simple method to ensure maximum throughput is to equate both rates yielding the following equation for the β_i:

$$\beta_i = \frac{RTT_{min,i}}{RTT_{max,i}}. \tag{21}$$

Based on the above observation we propose an adaptive strategy under which each source estimates $\frac{RTT_{min,i}}{RTT_{max,i}}$ and uses this quantity to determine β_i such that the throughput is matched before and after backoff, thereby ensuring that the buffer just empties following congestion and the link remains operating at capacity [10].

Comment : Alternatively, the backoff factor can be expressed as

$$\beta_i(k+1) \quad = \quad \min_{\text{j}} \ \beta_i(j) \frac{B_i^-(j)}{B_i^+(j)} \tag{22}$$

where $B_i^-(k)$ is the throughput of flow i immediately before the k'th congestion event, $B_i^+(k)$ the throughput of flow i immediately after the k'th congestion event. Both quantities are readily measured from packets ACK'ed over an RTT. This avoids the need to measure the ratio $RTT_{min,i}/RTT_{max,i}$ directly and is the approach currently employed in test implementations.

3.3 Adaptation to achieve responsiveness

As mentioned previously, in AIMD-like algorithms a trade-off exists between responsiveness and throughput efficiency. The back-off factor may need to approach unity on links with small queues to achieve

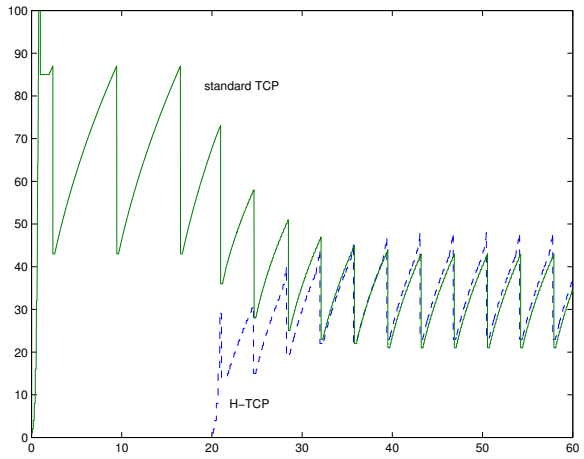

Figure 11: Example of standard TCP and H-TCP flows co-existing on a low speed link (NS simulation, network parameters: 5Mb bottleneck link, 100ms delay, queue 44 packets; H-TCP parameters: $\alpha^L = 1, \alpha^H = 20, \beta = 0.5, \Delta^L = 19$).

efficient utilisation. However values of β_i close to one will lead to slow convergence after a disturbance (e.g. traffic joining or leaving the route associated with the link, see examples below). We therefore need to adapt the source back-off factors to reflect the need to respond rapidly to changes in network conditions or to utilise bandwidth efficiently. This requires a network quantity that changes sensibly during disturbances and which can be used to trigger an adaptive reset that adjusts the β_i to ensure responsiveness. One quantity that can be used to achieve such an adaptive strategy is the throughput achieved just before a congestion event, B_i^-. B_i^- is determined by the link service rate B, which we assume is constant, the number of flows, and the distribution of bandwidth among the flows. Thus as new flows join we expect the B_i^- to decrease. On the other hand the value of B_i^- will increase when the traffic decreases. Thus by monitoring B_i^- for changes it is possible to detect points at which the flows need to re-adjust and reset β_i to some suitable low value for a time.

In summary, an adaptive reset algorithm is as follows.

(i) Continually monitor the value of B_i^-.

(ii) When the measured value of B_i^- moves outside of a threshold band, reset the value of β_i to β_{reset}.

(iii) Once B_i^- returns within the threshold band (e.g. after convergence to a new steady state, which might be calculated from β_{reset}), re-enable the adaptive backoff algorithm $\beta_i = \frac{RTT_{min,i}}{RTT_{max,i}}$.

In our experiments we reset β_i to 0.5 when B_i^- changes by more that 20% from one congestion epoch to another. Figure 14 illustrates the operation of the adaptive back-off and reset algorithm. It can be seen that the backoff factor of flow 1 is reset to 0.5 temporarily when flow 2 starts, ensuring rapid convergence (in around 4 congestion epochs, consistent with the eigenvalues of the A matrix with backoff factor of 0.5). Notice that the flows now converge quickly to the fair allocation, at which time the adaptive reset is disabled and the value of the β_i that utilises the link bandwidth effectively is used instead.

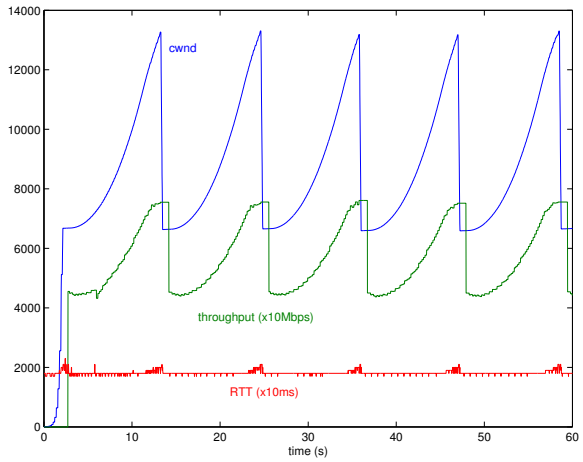

Figure 12: H-TCP with $\beta_i(k) = 0.5$ for all sources.

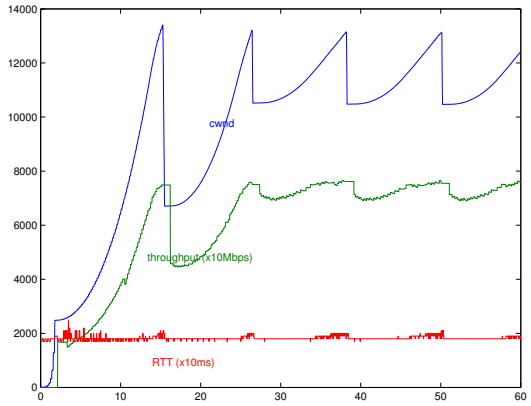

Figure 13: H-TCP with adaptive backoff.

3.4 Complete H-TCP algorithm

H-TCP can be implemented with minor modifications to the existing TCP congestion control algorithm as follows.

Let $\Delta_i(k)$ be the time since the last congestion event as experienced by source i, $\frac{RTT_{min,i}}{RTT_{max,i}}$ be the ratio of minimum and maximum RTT's as experienced by source i, and B_i^- is the throughput achieved by source i immediately before a congestion event

(a) On each acknowledgement set:

$$\alpha_i \leftarrow \begin{cases} 1 & \Delta_i \le \Delta^L \\ 1 + 10(\Delta_i - \Delta^L) + (\frac{\Delta_i - \Delta^L}{2})^2 & \Delta_i > \Delta^L \end{cases} \tag{23}$$

and then set

$$\alpha_i \leftarrow 2(1 - \beta_i)\alpha_i. \tag{24}$$

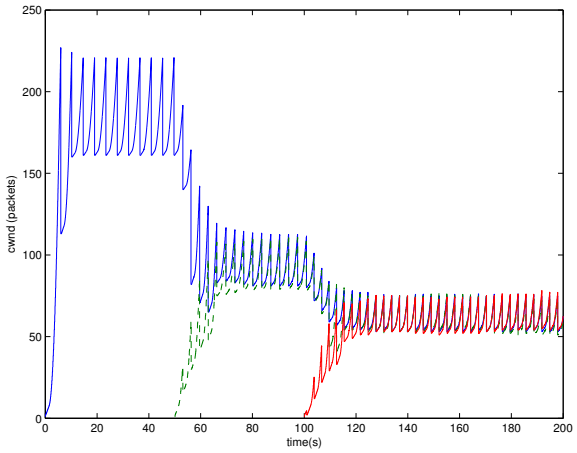

Figure 14: Adaptive congestion control. Notice that the effective backoff is reset in response to new flows starting (network simulation parameters are: 20Mb bottleneck link, 100ms delay, maximum queue size is 50 packets).

(b) On each congestion event set :

$$\beta_i(k+1) \quad \leftarrow \quad \begin{cases} 0.5 & |\frac{B_i^-(k+1)-B_i^-(k)}{\bar{B}_i^-(k)}| > 0.2 \\ \frac{RTT_{min,i}}{RTT_{max,i}} & \text{otherwise.} \end{cases} \tag{25}$$

Comment 1: It is prudent to restrict the $\beta_i(k)$ to the interval $[0.5, 0.8]$ since for very small queues $\frac{RTT_{min,i}}{RTT_{max,i}}$ may approach unity.

Comment 2: In line with Comment 2 in Section 3.1, we additionally advocate scaling the α_i by the respective round-trip time T_{d_i} to achieve a congestion epoch duration, and thus convergence time, that is effectively independent of round-trip time.

Acknowledgements

This work was supported by Science Foundation Ireland grant 00/PI.1/C067.

References

[1] R. Mukhtar, S. Hanly, and L. Andrew, "Efficient internet traffic delivery over wireless networks," *IEEE Communications Magazine*, vol. 41, no. 12, pp. 46–54, 2003.

[2] A. Berman and R. Plemmons, *Nonnegative matrices in the mathematical sciences*. SIAM, 1979.

[3] R. Shorten, D. Leith, J. Foy, and R. Kilduff, "Analysis and design of synchronised communication networks," in *Proceedings of 12th Yale Workshop on Adaptive and Learning Systems*, 2003.

[4] A. Berman, R. Shorten, and D. Leith, "Positive matrices associated with synchronised communication networks." Submitted to Linear Algebra and its Applications, 2003.

[5] R. Horn and C. Johnson, *Matrix Analysis*. Cambridge University Press, 1985.

[6] R. Shorten, F. Wirth, and D. Leith, "Positive matrices and communication networks." Technical Report, Signals and Systems Group, NUIM, 2004.

[7] S. Floyd, "High speed TCP for large congestion windows," tech. rep., Internet draft draft-floyd-tcp-highspeed-02.txt, work in progres, February 2003.

[8] T. Kelly, "On engineering a stable and scalable TCP variant," tech. rep., Cambridge University Engineering Department Technical Report CUED/F-INFENG/TR.435, 2002.

[9] C. Jin, D. Wei, and S. Low, "FAST TCP: Motivation, Architecture, Algorithms, Performance." Caltech CS Report CaltechCSTR:2003:010, 2003.

[10] R. Shorten, D. Leith, and P. Wellstead, "Adaptive congestion control of the internet." Submitted to Automatica, 2004.