An adaptive AIMD congestion control protocol for communication networks

Robert Shorten* , Douglas Leith** , Peter Wellstead

Hamilton Institute, NUI Maynooth, Ireland

Abstract. We present a new adaptive TCP protocol for the control of congestion in communication networks. The key innovative idea in our protocol is to combine results from Frobenius-Perron theory with online adaptation to realise a TCP variant that: (i) allocates the network pipe fairly amongst competing flows; (ii) coexists with other TCP variants; (iii) responds rapidly to changes in available bandwidth; and (iv) which strives to achieve a high data rate through the bottleneck link. When implemented, the new protocol requires only sender-side modification of the standard window congestion control scheme and is transparent to network buffers and network sources operating other TCP-variants. Results from the network simulator NS are presented to illustrate the key features of our scheme.

Key Words : *Adaptive congestion control; Network congestion control; Communication networks*

1 Introduction

In this paper we propose a new congestion control protocol for networks of sources employing additive-increase multiplicative-decrease (AIMD) congestion control algorithms. We show that using this protocol it is possible to obtain networks that achieve: (i) fair allocation of the network pipe among competing flows; (ii) TCP-friendliness toward other conventional TCP sources; (iii) rapid allocation of available bandwidth among sources (network responsiveness); and (iv) an efficient use of the bottleneck link capacity. We have shown in an earlier paper [1] that for conventional AIMD networks operating under-provisioned links, high data throughput cannot be achieved without adversely affecting the responsiveness characteristics of the network. Roughly speaking, networks whose sources multiplicative-decrease (backoff) factors are close to zero are very responsive whereas those networks with backoff factors close to unity are highly efficient in terms of data throughput. In this paper we show that both of these objectives can be achieved simultaneously by means of a new type of AIMD algorithm. In a similar manner to TCP-Westwood [2], our key idea is to use available measurements to adapt the network backoff factors to reflect prevailing network conditions. Crucially, we use results from positive matrices and Frobenius-Perron theory to ensure that our adaptation strategy for the AIMD parameters is constrained to result in fairness amongst competing operating conventional TCP-Reno.

The adaptive control ideas build upon the analysis of the dynamic properties of congestion control mechanisms presented in [3, 1] for networks of synchronised sources. The results in these papers shows that the dynamic congestion avoidance properties

* Joint first author; Email: robert.shorten@may.ie
** Joint first author; Email: robert.shorten@may.ie

of networks whose sources employ AIMD congestion control algorithms can be deduced from the study of a positive matrix [4]. At a more general level however, the use of adaptive control in congestion control is a quite natural sequel to Jacobson's original analysis [5]. In particular, the increase/backoff format of AIMD flows has a strong family resemblance to Dual Control concepts [6] of probing (to gather information concerning a system), combined with control (to effectively use a system). Many of these issues are particularly topical in the context of high-speed networks [7, 8]) where a number of proposals for modifications to TCP are currently being studied (it seems likely that high-speed networks will have queues that are small relative to the window sizes of the flows that utilize the network due to cost and to reduce network latency and jitter).

This paper is structured as follows. An overview of the results derived in [3, 1] is given in Section 2. In Section 3 we discuss why adaption is required to achieve the three criteria, of fairness, efficiency and responsiveness. In Section 4, an adaptive scheme is proposed. The paper concludes with a series of examples of the adaptive method for various traffic and network scenarios.

2 Positive matrices and the internet

The algorithm presented in this paper is motivated by the insights obtained from the analysis framework for synchronised communication networks developed in [3, 1][1]. A summary of the main results in these papers is presented here for completeness.

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, and that all of the sources are operating a TCP-like congestion control algorithm. TCP (transmission control protocol) operates a window based congestion control algorithm. The TCP standard defines a variable *cwnd* called the congestion window. Each source uses this variable to track the number of sent unacknowledged packets that can be in transit 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 its send rate when congestion is detected as depicted in Figure 1. 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 $0 < \beta_i < 1\ \forall i \in \{1, ..., n\}$. Then, under the assumptions of source synchronisation and uniform round-trip-time (RTT)[2] for all sources, a convenient network model can be found using elementary algebra.

Let $w_i(k)$ denote congestion window size of source i immediately before the kth network congestion event is detected by the sources. 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 $t_a(k)$ is the time 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.

[1] The problem of developing an analysis framework for designing communication networks has become topical in the context of internet congestion control [5, 9–13], but is also of relevance in a variety of problems where a number of devices compete for a shared resource.

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

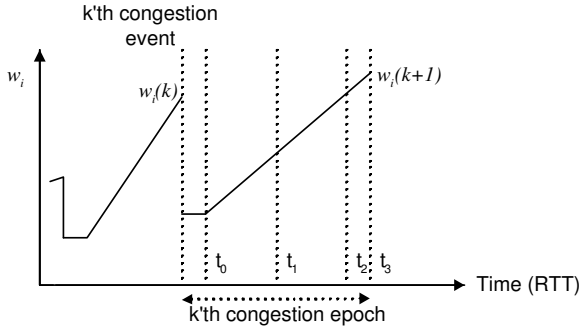

Fig. 1. Evolution of window size

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 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 $q_{max} + BT$ where q_{max} is the maximum queue length of the congested link, B is the service rate in packets per second and T is the round-trip time when the queue is empty. At the $(k+1)$th congestion event

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

and

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

Hence, it follows that

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

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

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

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 [4]. Many

results are known for positive matrices and we will 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 stability; (iii) convergence rate; and (iv) and network throughput. Roughly speaking, window or pipe fairness refers to a steady state situation where n sources operating $AIMD$ algorithms have an equal number of packets in flights at each congestion event; stability refers to the existence of a unique fixed point to which the network dynamics converge; and throughput refers to the objective that the network operates at the bottleneck-link capacity. It is shown in [1] that these properties can be deduced from the network matrix A. We briefly summarise here the relevant results in these papers.

Theorem 1. *[3] 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 1 and properties of non-negative matrices [14, 4].

Corollary 1. *[3] 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 $\rho(A) = 1$. It follows that all other eigenvalues of A satisfy $|\lambda_i(A)| < \rho(A)$. The network possesses 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 are easily deduced from the above discussion.

(i) **Fairness and friendliness:** Window fairness is achieved when the Perron eigenvector is a scalar multiple of $x_p^T = [1, ..., 1]$. Consequently, if $\frac{\alpha_i}{1-\beta_i}$ does not depend on i then the matrix is symmetric and the system is fair. Further, since for conventional TCP-flows we have that

$$\alpha = 2(1 - \beta), \tag{7}$$

it follows that any new protocol operating an AIMD variant that satisfies $\frac{\alpha_i}{1-\beta_i} = 2$ will be TCP-fair at each congestion event.

(ii) **Network responsiveness:** It follows from the Corollary that the second largest eigenvalue of the matrix A determines the convergence properties of the entire network. It is therefore important to determine this eigenvalue. We show in [1] that the 95% network rise-time when measured in number of congestion epochs is bounded by $n_r = \frac{log_e(0.05)}{log_e(\lambda_{n-1})}$ where λ_{n-1} is the second largest eigenvalue of A. For the frequently encountered situation where $\beta_i = 0.5$ for all i, we show in [3] that $\lambda_{n-1} = 0.5$; hence $n_r \approx 4$. 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. We also show in [1] that all the eigenvalues of A are real and positive and lie in the interval $[\beta_1, 1]$ (where we assume that $\beta_1 \leq \beta_2 \leq .... \leq \beta_n$). In particular, the second largest eigenvalue is bounded above by β_n and below by β_{n-1}. Consequently, fast convergence to the equilibrium state (the Perron eigenvector) is guaranteed if the largest backoff factor in the network is small.

(iii) **Network throughput :** It follows that immediately before a congestion event the network bottleneck is operating at link capacity and the the total data throughput through the link is given by

$$R(k)^- = \frac{\sum_i^n w_i(k)}{T + \frac{q_{max}}{B}}, \tag{8}$$

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

$$R(k)^+ = \frac{\sum_i^n \beta_i w_i(k)}{T} \tag{9}$$

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. A simple method to ensure maximum throughput is to equate both rates yielding the following equation for the β_i:

$$\beta_i = \frac{T}{T + \frac{q_{max}}{B}} = \frac{RTT_{min}}{RTT_{max}}. \tag{10}$$

(iv) **Maintaining fairness :** Note that setting $\beta_i = \frac{RTT_{min}}{RTT_{max}}$ requires a corresponding adjustment of α_i to maintain network fairness. TCP fairness is ensured by adjusting α_i according to $\alpha_i = 2(1 - \beta_i)$.

In summary, if $\frac{\alpha_i}{1-\beta_i}$ does not depend on i then the matrix is symmetric and the system is fair. Since A is positive all the non-Perron eigenvalues are in the interior of the unit circle so the network has a unique stationary point (the Perron eigenvector) that is stable. More generally, it is shown in [1] that even when the matrix A is not symmetric, its eigenvalues are real and positive, and all the non Perron eigenvalues lie inside the unit circle and interlace the β_i's. Hence, the rate of convergence to the stationary solution, which depends on the second largest eigenvalue of A, is bounded from above by the largest β_i. Thus to obtain rapid convergence one is always interested in ensuring that the β_i's are uniformly small (close to zero). On the other hand the link utilisation is highest for β_i close to unity since this ensures that network buffers do not empty following a congestion event.

Finally we note that many of our results extend in a natural manner to the case of non-synchronised networks in the following way. Networks without synchronisation can be modelled by letting $\beta_i(k)$ be either β_i or 1 for each source; i.e. $\beta_i(k) = 1$ when the i'th source does not experience a packet drop at the k'th congestion event and $\beta_i(k) = \beta_i$ otherwise. This yields the following description of the network:

$$W(k+1) = A(k)W(k), \tag{11}$$

where $A(k)$ is one of $2^n - 1$ possible matrices over each congestion epoch. Here the asymptotic properties of the matrix product $A(k)A(k-1)....A(0)$ as $k \to \infty$ determine network fairness, stability, convergence and throughput properties.

3 Adaptive congestion control

In this section we discuss how the results quoted in the previous section can be used to guide the design of an adaptive scheme that ensures efficient bandwidth utilisation and responsiveness to disturbances. Our basic strategy is to continuously adapt

the AIMD flow parameters to ensure that the network is responsive and that the bottleneck link capacity is fully utilised and consists of two main components: (A) adaptation to ensure bandwidth utilisation; and (B) adaptation to ensure network responsiveness.

(A) Adaptation to achieve 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 when, for example, each flow sharing a common bottleneck link has a different round-trip time. Moreover, in high-speed networks large high speed buffers are problematic for technical and cost reasons.

The solution is an adaptive backoff mechanism along the lines suggested in Item (iii) in Section 2 in which the provisioning of each TCP flow is estimated on-line and the backoff factor set such that the throughput on a per flow basis is matched before and after backoff. In ideal circumstances this should ensure that the buffer just empties following congestion and the link remains operating at capacity[3]. The parameters required for such an adaptive mechanism can be easily obtained at each flow by measuring the maximum and minimum round trip time. Since we have that:

$$RTT_{min,i} = T_i,$$
$$RTT_{max,i} = \frac{q_{max}}{B} + BT_i.$$

then the multiplicative backoff factor β_i that ensures efficient use of the link is $\beta_i = \frac{RTT_{min,i}}{RTT_{max,i}}$ where $RTT_{min,i}$ and $RTT_{max,i}$ are the minimum and maximum RTT's seen by the i'th source, and T_i is the minimum round trip time as seen by the i'th source when the queue is empty.

In summary: each source operates the adaptive backoff mechanism operates as follows.

(i) Estimate $\frac{RTT_{min,i}}{RTT_{max,i}}$.
(ii) Set the multiplicative backoff factor to be equal to $\frac{RTT_{min,i}}{RTT_{max,i}}$.
(iii) Adjust the corresponding additive increase parameter according to $\alpha_i = K(1-\beta_i)$, $K > 0$, $K \in R$: $K = 2$ gives TCP fairness.
(iv) Monitor continuously the ratio $\frac{RTT_{min,i}}{RTT_{max,i}}$ to check for dynamic changes in the link provisioning.

Comment : In real networks, $RTT_{min,i}$ and $RTT_{max,i}$ are noisy quantities. It is therefore prudent for each source to estimate the ratio of these quantities indirectly; for example using filtered RTT measurements, or by matching throughput (averaged over an RTT) immediately before and immediately after backoff (see [15] and [16] for further details).

Comment : This adaptation strategy is similar TCP-Westwood. However, it differs from Westwood in a number of key areas. In particular: (i) we make no attempt to estimate the packet rate of the bottleneck link; (ii) our adaptation scheme is based upon easily obtained network measurements and does not require complex

[3] In circumstances where the buffer does not empty the link will still operate at capacity.

filtering strategies; and (iii) and network fairness properties are guaranteed in our scheme by controlling the Perron eigenvector (adjusting the parameters according to $\alpha_i = 2(1 - \beta_i)$ gives TCP fairness).

(B) Adaptation to achieve responsiveness

The ratio $\frac{RTT_{min,i}}{RTT_{max,i}}$ may approach unity on highly under provisioned links. However values of β_i close to one will give slow convergence after a disturbance (e.g. traffic joining or leaving the route associated with the link, see examples below). It follows that we need a further adaptive mechanism which continuously adjusts the trade-off between network responsiveness and efficient link utilisation. This requires a network quantity that changes sensibly during disturbances and can be used to trigger an adaptive reset. One variable that does this is the minimum of the mean Inter Packet Time for each source ($IPT_{min,i}$), where the mean is taken over a round-trip time. The $IPT_{min,i}$ is a measure of the link bandwidth allocated to a particular flow. This in turn 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 $IPT_{min,i}$ for 'our' flow to increase. On the other hand the value of $IPT_{min,i}$ will decrease when the traffic decreases. Thus by monitoring $IPT_{min,i}$ 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 for each source is:

(i) Continually monitor the value of $IPT_{min,i}$.
(ii) When the measured value of $IPT_{min,i}$ moves outside of a threshold band, reset the value of β_i to β_{reset}.
(iii) Once $IPT_{min,i}$ 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 $\beta_i = \frac{RTT_{min,i}}{RTT_{max,i}}$.

The two adaptive mechanisms (backoff and reset) that comprise the adaptive control algorithm are shown schematically in Figure 3.

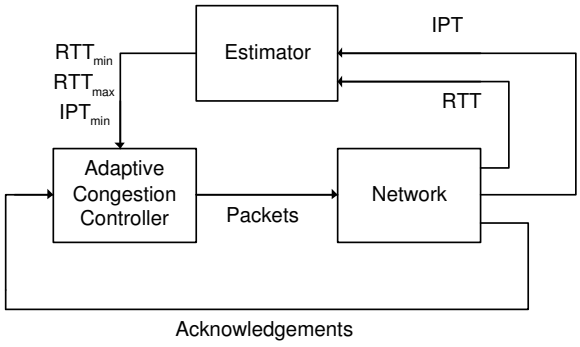

Fig. 2. Adaptive congestion control scheme

The figure depicts a classical adaptive control paradigm [17] in which an online estimator continuously estimates relevant system parameters and uses these to con-

tinuously update the settings in a control law. In this particular case we argue that the parameters which control the adaptation can be determined exactly. The value of $IPT_{min,i}$ may vary slightly due to other network mechanisms, therefore the band of variations that are allowed before reset must be selected by the user. At the moment we consider this to be a user 'tuning knob', although in future work we are considering sequential change detection methods which can automate this process.

4 Examples

In this section we present examples that illustrate some of the main points of the paper.

Example 1: Fair coexistence of flows. In this example we consider two flows, each with different choices of α_i and β_i. Both flows satisfy the formula $\alpha_i = 2(1-\beta_i)$. NS simulation results are shown in Figure 3. Evidently, each of the flows coexist fairly at the congestion event, i.e. the have the same window size at each congestion event.

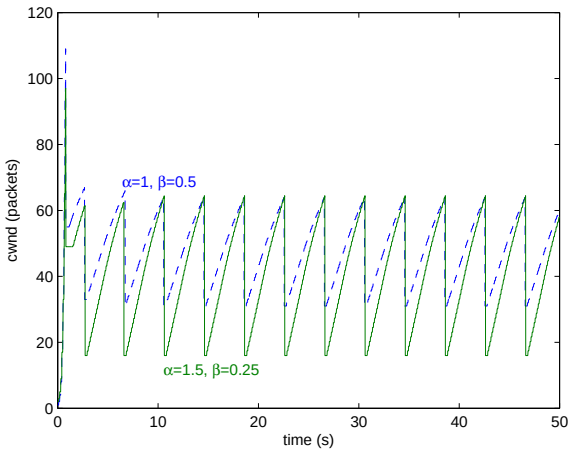

Fig. 3. Example of fair coexistence of two AIMD sources (network simulation parameters are: 10Mb bottleneck link, 100ms delay, maximum queue size is 40 packets).

Example 2: Dependence of convergence on backoff factor. Figure 4 shows NS simulation results for three flows operating the adaptive backoff mechanism $\beta_i = \frac{RTT_{min,i}}{RTT_{max,i}}$. To illustrate the behaviour of the adaptive backoff mechanism in the context of recent proposed modifications to TCP, we show results for the high speed variant of TCP proposed in [3] (this modifies the TCP AIMD algorithm to achieve a faster than linear rate of increase while ensuring backward compatibility with standard TCP on slower networks). For the choice of network parameters used, we have that $\beta_i = 0.77$. For this value of backoff factor, the foregoing positive matrix analysis indicates a convergence time (95% rise time) of 11 congestion epochs and it can be seen from the figure that this is in good agreement with packet-level simulation results.

Example 3: Variation of ITP_{min} with traffic. Figure 5 shows the $IPT_{min,i}$ time history of flow 1 in the previous example (the congestion window evolution is also

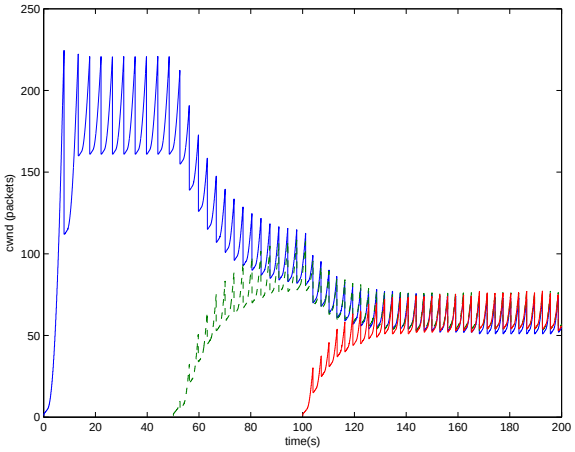

Fig. 4. Illustrating poor responsiveness with adaptive back off (network simulation parameters are: 20Mb bottleneck link, 100ms delay, maximum queue size is 50 packets).

shown for comparison). Notice that the IPT for the first flow increases as the second flow start to seize bandwidth.

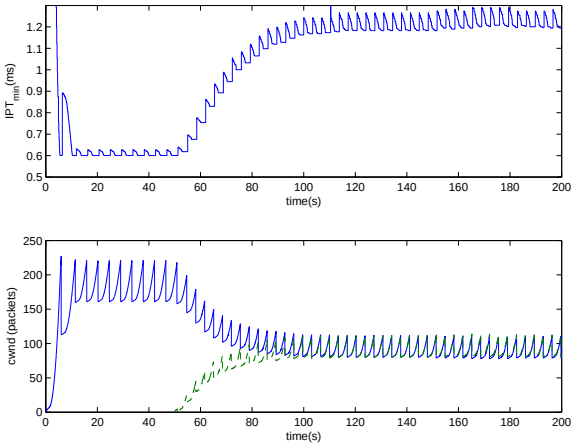

Fig. 5. Illustrating the change in IPT with traffic (network simulation parameters are: 20Mb bottleneck link, 100ms delay, maximum queue size is 50 packets).

Example 4: Adaptive congestion control. Figure 6 repeats the simulation of Example 2 but now with the both adaptive backoff and reset, as described in Section 3B. 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). This behaviour is repeated when a third flow starts.

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.

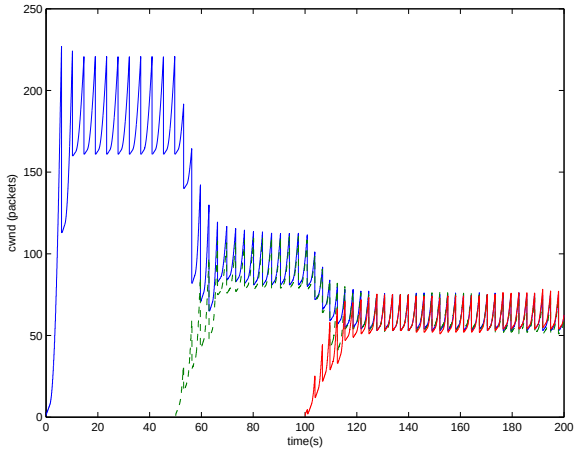

Fig. 6. Adaptive congestion control. Notice that the effective backoff is reset in response to new flows starting (network simulation parameters are: 20Mb bottleneck link, 100ms delay, maximum queue size is 50 packets).

5 Discussion and concluding remarks

In this paper we have discussed the dynamic performance of communication networks where each source operates an AIMD congestion control algorithm. We have shown how adaptive control has a clear role in achieving the twin objectives of efficient bandwidth use and responsiveness. For rapid response to disturbances, low values of β_i are required. In the case of underprovisioned bottleneck links this may lead inefficient use of the link bandwidth. To overcome this problem we have proposed an adaptive congestion control protocol that improves both bandwidth utilisation and network responsiveness. Our protocol, which is based on adapting the backoff factor of each flow to reflect the prevailing network conditions, involves only sender side modification of TCP. Simulations are presented to illustrate the efficacy of the proposed algorithm.

Acknowledgements

This work was supported by Science Foundation Ireland grant 00/PI.1/C067 and the Walton Visitor Programme. This work was also partially supported by the European Union funded research training network *Multi-Agent Control*, HPRN-CT-1999-00107[4] and by the Enterprise Ireland grant SC/2000/084/Y. Neither the European Union or Enterprise Ireland is responsible for any use of data appearing in this publication.

References

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

[4] This work is the sole responsibility of the authors and does not reflect the European Union's opinion

2. M. Gerla, M. Sandaidi, M. Valla, and R. Wang, "TCP Westwood with adaptive bandwidth estimation to improve efficiency/friendliness tradeoffs." To appear in Computer Communication Journal, 2003.
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 and R. Plemmons, *Nonnegative matrices in the mathematical sciences*. SIAM, 1979.
5. V. Jacobson, "Congestion avoidance and control," in *Proceedings of ACM SIGCOMM*, pp. 314–329, 1988.
6. A. A. Feldbaum, "Dual control theory: Part I," *Automation and Remote Control*, vol. 21, no. 9, pp. 874–880, 1960.
7. S. Floyd, "High-Speed TCP for large congestion windows," tech. rep., Internet draft draft-floyd-tcp-highspeed-02.txt: Work in progress, February 2003.
8. T. Kelly, "On engineering a stable and scalable TCP variant," tech. rep., Cambridge University Engineering Department Report CUED/F-INFENG/TR.435, 2002.
9. S. Floyd and K. Fall, "Promoting the use of end-to-end congestion control in the internet," *IEEE/ACM Transactions on Networking*, vol. 7, no. 4, pp. 458–472, 1999.
10. Various authors, "Special issue on TCP performance in future networking environments," *IEEE Commuications magazine*, vol. 39, no. 4, 2001.
11. S. Low, F. Paganini, and J. Doyle, "Internet congestion control," *IEEE Control Systems Magazine*, vol. 32, no. 1, pp. 28–43, 2002.
12. Various authors, "Special issue on internet technology and convergence of communication services," *Proceedings of the IEEE*, vol. 90, no. 9, 2002.
13. J. Hespanha, S. Hohacek, K. Obrarzka, and J. Lee, "Hybrid model of TCP congestion control," in *Hybrid Systems: Computation and Control*, pp. 291–304, 2001.
14. R. Horn and C. Johnson, *Matrix Analysis*. Cambridge University Press, 1985.
15. R. Shorten, D. Leith, and P. Wellstead, "Adaptive congestion control of the internet." Submitted to Automatica, 2004.
16. D. Leith and R. Shorten, "H-TCP:TCP for high-speed and long-distance networks," in *Web-proceedings of the Second International Workshop on Protocols for Fast Long-Distance Networks (http://www-didc.lbl.gov/PFLDnet2004/)*, Argonne National Laboratory, Argonne,, Illinois USA, 2004.
17. P. E. Wellstead and M. B. Zarrop, *Self-Tuning Systems: Control and Signal Processing*. Wiley, 1995.