Differential QoS and Pricing in Networks: where flow-control meets game theory

Peter B Key and Derek R McAuley
Microsoft Research Limited,

Published in IEE Proceedings Software, March 1999 *

March 1999

Abstract

This paper looks at ways of providing Quality of Service to users based on a simple pricing scheme. It is primarily aimed at elastic traffic, and it is users rather than the network who define the flow control schemes. A framework for assesing schemes and algorithms via a distributed game is presented.

1 Introduction

In telecommunication networks, the traditional approach to Quality of Service (QoS) requires users to tightly specify or control their traffic in return for tight QoS guarantees, while the network provider turns away excess demand. The Internet offers the other extreme of accepting every demand, giving a vanilla 'best effort' QoS and relying on users behaving 'nicely' to ensure 'fairness'. Neither of these approaches is ideal, and both communities are struggling to define usable mechanisms for providing differential QoS.

Might it not be possible to learn from some other areas where pricing and tariffs are used extensively both to influence user behaviour and to protect the resources? We claim that it is, and moreover that a rich set of behaviours and optimisations can be constructed from the simplest of frameworks, rich enough to meet the needs of any user or application. Our only proviso is that we deal with demands that can react or adapt to some signal given by a resource. We show how to construct this simplest of models, and then describe how playing a sequence of distributed games could test the assertion. One of our intentions is to try to synthesise insights from different disciplines and also to encourage practitioners to design strategies that can be tested in this environment.

Our starting point is the work of Kelly, who in a series of papers elucidated some of the linkage between congestion controls and pricing in networks (eg [1]), and the experimental work of Gibbens and Kelly [2]. Other approaches can be found in Odlyzko [3] or the congestion pricing work of MacKie-Mason and Varian [4].

2 Framework

Our underlying model is a set of users $\mathcal{R}$ indexed by r (think of r indexing a route) who place demands x_r on a set of resources, where the set of resources $\mathcal{J}$ is indexed by j. Resource j can have a number of (multi-dimensional) attributes such as a capacity C_j or a queueing discipline and buffer-size B_j. If the resources are connected via a network then a route identifies a set of resources with an implicit ordering of resources. For fixed routing, if the matrix $\boldsymbol{A} = (A_{jr})$ denotes the amount of resource consumed at resource j by user r then the (offered) load to the resource vector $\boldsymbol{C}$ is $\boldsymbol{Ax}$. For convenience assume that $\boldsymbol{A}$ is a matrix of ones or zeros.

The vector $\boldsymbol{x}$ can be interpreted as a vector of flows, with x_r an amount of bandwidth or bandwidth per unit time. Now suppose each user has a utility function $U_r(x_r)$ relating bandwidth to 'worth', which we assume is concave, and hence characterises *elastic* traffic, (Shenker [5]).

For resources with finite capacity and utility functions which are additive across users, the 'Social Optimum' which maximises $\sum_r U_r(x_r)$ for concave utility functions will, in general, accept as many users as possible, sharing out resources and dividing them ever more finely. If there is some cost function associated with the loading y on a resource, $C_j(y)$ say, which is increasing and differentiable and measures the rate at which costs are incurred, then the 'social planner' seeks to

$$\begin{array}{ll} \text{maximise} & \mathcal{U}(\boldsymbol{x}) = \sum_r U_r\left(x_r\right) - \sum_j C_j\left(\sum_r A_{jr} x_r\right) \\ \text{over} & \boldsymbol{x} \geq \boldsymbol{0}. \end{array} \tag{1}$$

When $C_j\left(\cdot\right)$ are convex functions, the optimum for non-zero x_r satisfies

$$U_r'\left(x_r\right) = \sum_j \mu_j A_{jr} \tag{2}$$

where the μ_j are interpreted as the shadow prices

$$\mu_j = p_j\left(\sum_r A_{jr} x_r\right) \text{ with } p_j\left(y\right) \stackrel{def}{=} \frac{dC_j\left(y\right)}{dy}. \tag{3}$$

Suppose that users are charged an amount α_r per amount sent, then the User Optimum is

$$\text{maximise } U_r(x_r) - \alpha_r x_r \text{ over } x_r \geq 0 \tag{4}$$

which has the unique solution for a concave utility function

$$x_r = \begin{cases} x^* & \text{where} \quad U_r'(x^*) = \alpha_r \quad \text{if } U_r'(0) > \alpha_r \\ 0 & \text{otherwise} \end{cases} \tag{5}$$

and where it is worth the user accepting any amount of bandwidth offered up to x^* (assuming prices are rational, so $U_r'(0) > \alpha_r$). The 'user' and 'social' optima coincide if the prices are right, that is if $\alpha_r = \sum_{j \in r} \mu_j$.

This suggests the following: users generate a load on the network, $x(t)$ and receive feedback signals $f(t)$ where the feedback signals are related to shadow prices. For example certain packets receive 'marks'. If costs reflect loss, then packets should be marked if and only if the packet contributed to a loss event, giving *sample path* shadow prices (see [2] which describes marking for buffered and unbuffered resources). Consider for instance, a single unbuffered resource, capacity C, and a stochastic setting where the load Y is a random variable with mean y. The expected loss, $C(y)$ is then just $\mathbb{E}\left[(Y-C)^+\right]$; for Y Poisson or Gaussian, the shadow price (derivative of C w.r.t y) can be shown to be $P_{sat} = \mathbb{P}\{Y \geq C\}$, the probability of resource saturation. Hence the marking scheme for an unbuffered resource is very simple: mark all those packets contributing to the load when the resource is overloaded, otherwise mark none. Note that the probability of loss, $P_{loss} = \mathbb{E}\left[(Y-C)^+\right]/\mathbb{E}[Y]$ is typically an order of magnitude less than P_{sat}, so many more packets are marked than are lost.

With the identification $U_r(x) = w_r \log x$, the social optimum is a Nash arbitration scheme and 'proportionally fair'[1], with $w_r = x_r \sum_{j \in r} \mu_j$, and hence we take the feedback signals to be of the form

$$f_r(t) = x_r(t) \sum_{j \in r} \mu_j(t). \tag{6}$$

3 A class of Control Schemes

The choice of what is considered 'optimal', even from a system viewpoint, is heavily influenced by users' utility functions. However these are usually unknown to the network or system (and perhaps even to the user), which suggests the following: let the network determine the prices, and let the users respond and place demands upon the network which the network will try to meet. The users are then free to behave as they wish, possibly implicitly performing some user optimisation. This certainly provides a rich class of behaviours, and the question we seek to answer is whether or not this is sufficient, and whether it is possible to run a network in this manner.

The network will generate revenues, which are related to congestion, and if all of the congestion revenues are reinvested in new capacity, then capacity will be expanded to the point where its marginal value is equal to its marginal cost. In other words the prices send back the correct signals for capacity expansion. The term 'price' is suggestive of a monetary transaction, but it need not involve money. Indeed, even in the case of exchange of money, the actual charging mechanism is separate from the pricing signals, and could involve pre-purchase of credits

3.1 Redefining Flow Control Algorithms

To implement these types of Flow Control Algorithms all that is needed is a means of feeding back information to the user. The signal is just a 'mark' associated with each packet. Where there are a number of resources, packets could potentially be 'marked' at each resource, thus the mark would be a non-negative integer. However in a stochastic network setting, if the network is running in a 'sensible' region, then the probability of a packet being marked at more than one resource is small, hence the 'mark' can be approximated by just one bit. Given that many data transfer protocols have a mechanism for Congestion Indication (CI), then clearly we could fit Explicit Congestion Notification, FECN or BECN into this scheme. With more advanced protocols such as ATM, the marked scheme fits in easily (into the ABR traffic class using binary feedback). As an aside, note that for unbuffered resources, the shadow prices are just the probability of resource saturation, which will be (approximately) additive along a route provided that the resources are lightly loaded — another reason for wanting to run in an 'uncongested' region.

As a specific example, let us see how a 'TCP-like' algorithm could be constructed. To implement a suite of rate control algorithms requires certain changes to the usual TCP algorithm [6]:

- Add a *marked packets* variable, *mpkt* to the per-connection state. This is an integer counter of the number of marked packets received by the sender.
- The receipt of a non-zero *mkpt* may cause a recalculation of the congestion window *cwnd* ('may' since the updating might be done periodically, rather than at every change of state variable).
- There is no explicit slow-start mode. The *cwnd* at start or after a lost packet is determined by the congestion algorithm — so does not have to default to 1.
- On receipt of an ACK, the *cwnd* is updated according to the congestion algorithm, which can use information from the *mkpt* state. In otherwords, the Congestion Avoidance phase is altered, as is Slow Start. Both are replaced by application specific increases.
- The slow-start threshold, *ssthresh* is retained for compatibility, but there is no requirement to use it.

3.2 A TCP-like example

Gibbens and Kelly have looked at the following class of algorithms, where we are working in discrete time for convenience.

'Willingness to pay scheme' Flow control scheme with parameters κ and w

$$x_{t+1} = x_t + \kappa\,(w - f_t) \tag{7}$$

where w represents a 'willingness to pay', and κ affects the rate of convergence of the algorithm.

The case $w = 1$ could be considered an 'improvement' to TCP, or we could consider

'TCP-like' algorithm

$$x_{t+1} = x_t + \kappa\left(w - f_t\frac{x_t}{2}\right) \tag{8}$$

which replaces the 'lost packet' signal by the feedback signal, but then reacts in the same way as TCP (has $f_t \propto x_t$ and $f_t = x_t\mathbb{P}\{\sum X \geq C\} = x_t P_{sat}$ for a single unbuffered resource instead of $f_t = x_t P_{loss}$ which is TCP).

These algorithms are implicitly related to a logarithmic utility function, or a proportionally fair scheme. It is worth noting what happens as we scale by the number or sources. With strictly concave utility functions, the overall return from the network depends on the number of users — the more users or routes, the greater the return, hence the shadow price also depends on the number of users. Only if the utility functions are linear is independence of the number of users achieved. For the first scheme, from equation (7) the equilibrium value is achieved when

$$w = \mathbb{E}\left[f_t\right] \tag{9}$$

hence it is not possible to design a scheme with w fixed that gives the same network performance independent of the number of users. This illustrates some of the difficulties posed by trying to define an 'optimal' flow control scheme and mandating everyone to use it.

As w vary, the target performance varies. The following figures illustrate the effect of varying w and the number of connections n when the capacity is 100, using a stochastic model. Figure 1 has $w = 1$, and shows how the overall load increases above capacity as the number of sources rises, giving an incentive to increase capacity. Figure 2 has $w = 0.05$ and the load increase is much more gradual (in the steady state $0.05 = x\mathbb{P}\{nx \geq C\}$) with a strong incentive for the users to keep the network lightly loaded. These figures can be contrasted with those for TCP, using the same set-up of single resource of capacity 100. From the discussion around equation (8) it follows that the steady-state behaviour of a discrete-time TCP algorithm is just

$$x_t = \frac{\sqrt{2}}{\sqrt{P_{loss}}}. \tag{10}$$

This is in units of segment size, hence if R is the round trip time, dividing by R gives the throughput, which is of the form

$$\frac{c}{R\sqrt{P_{loss}}} \tag{11}$$

for some constant c, as others have found [7]. Figure 3 shows how the offered load increases dramatically with the number of users, caused by too few feedback messages being generated. Changing the feedback signal to the shadow-price

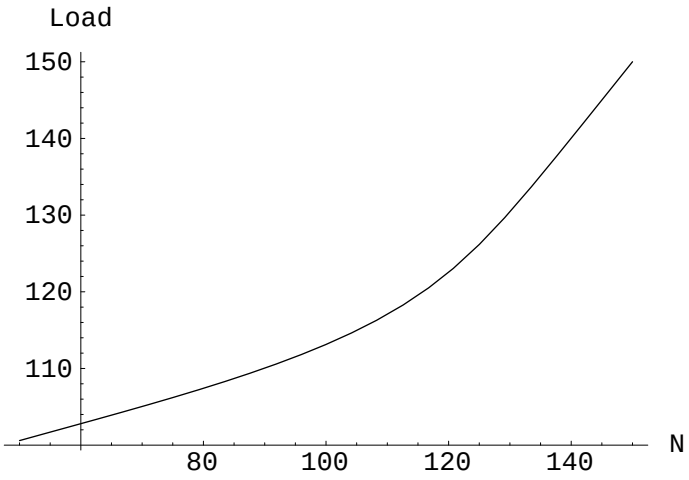

Figure 1: Elastic Flow control scheme: $C = 100, w = 1$.

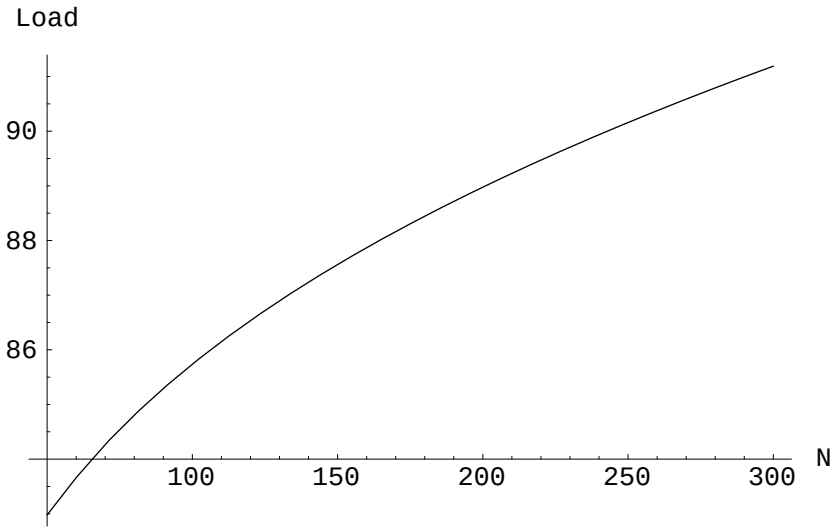

Figure 2: Elastic Flow control scheme: $C = 100, w = 0.05$.

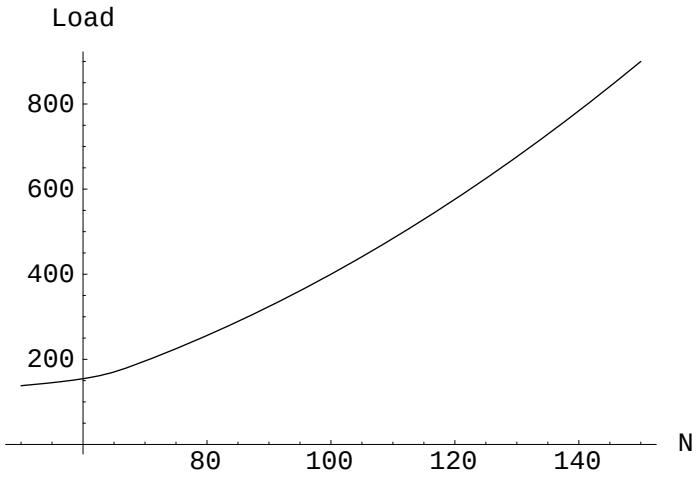

Figure 3: TCP control scheme: $C = 100$.

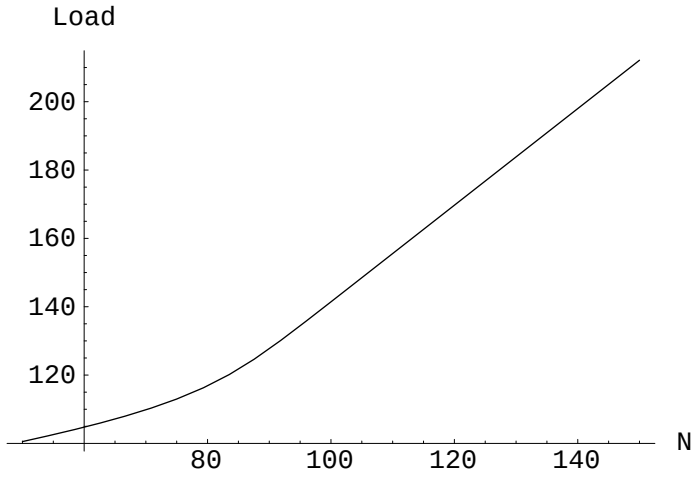

Figure 4: TCP-like with revised marks: $C = 100$

in(8) reduces the offered load, see figure 4 which has a similar shape to figure 1, the $w = 1$ flow scheme, though still with a poorer performance.

4 A Distributed Game as an Experimental Framework

How do we decided if it is possible to provide a rich-enough class of Qualities of Service to the users? One way would be to extensively simulate, however why not run a distributed emulation of a network, where users are free to invent their own algorithms or behave as they wish on a 'network'? But this is exactly like a distributed multi-user game, where each user is trying to optimise their own 'objective'. The advantage of this approach is that it both generates a mix of loads likely to be encountered in a real heterogeneous network, and also encourages users to develop 'optimal' algorithms and strategies, much as happened in Axelrod's repeated Prisoners Dilemma experiments [8].

Gibbens and Kelly[2] have tried out initial set of experiments with different

algorithms[1], which require algorithms to be implemented in Java. We propose a more general framework, where Users 'play' against a 'Network' which represents a resource system:

- The game will comprise a distributed set of users, and a 'Network', where the Network will initially be emulated on a PC or cluster of PCs at a single site.
- A protocol interfaces between the users and the Network.
- Users will be able to interact (connect) across the Internet.
- Users are free to interact in real time, or via applications or programs. These can be in any language.
- Translation stubs between users' programs and the game protocol will be provided for certain common languages (eg C, Java).
- Game Time can be slowed down or speeded up compared with real time.

4.1 Protocol for a distributed Game

The distributed game is between different users, or between a User and the Network. Hence it is necessary to design a protocol allowing users to communicate with the Network. Users generate load (packets), which are sent to the Network that generates a set of marks. A more general framework allows a user to connect to a destination, or set of destinations, the Network then decides which resources the user is allocated, and sends back marks to the user based on state of the resources. Note that the feedback signals will depend upon what resource model the 'Network' is using.

The Network sends a signal to the user when the game starts and ends, and sends information about the relative speed of the game (compared to slots or real time) which enables users playing in real time (via Telnet) to compete with compiled programs. Timing information is sent back to the user via a heartbeat.

The basic game protocol is text based, sits above TCP, and consists of just three parts:

<**packet**> : <pkt>= < destination ><token><size>
where <destination>, <token> and <size> are all unsigned single word-length. The token is generated by the user, and used to enable users to detect loss. <size> is interpreted as an integer, <destination> is used to identify a 'route' in the game Network.

<**ACK**> : <ACK>=<token><mark>
where <token> can be void or corrupted, and <mark> is in general an integer giving a number of marks on the packet.

<**time**> : <time>=<sec.μsec>
sends heartbeats back to the user giving time in the game world.

[1] http://www.statslab.cam.ac.uk/~richard/research/topics/evolution/

Additional commands are needed to stop and start the game and control the information flow. The sequence of events in time looks like

User: requests to join game
Receives an acknowledgement
Receives info on relative speed at which game played
Receives notice of start of game
    Repeat {
        Generate token
        Calculate load (packet size)
        Send <packet>= < destination ><token><size>
        Receive <ACK>=<token><mark>
        Receive <time>=<sec.μsec> }
    Until End of game
Receives end of game signal
Ends session

Note that the receiving and generating of packets, ACKS and time do not have to follow that sequence — a number of packets can be generated before any ACKs are received and time messages will be received asynchronously.

5 Objectives, Algorithms and Analysis

It is interesting to note that the analysis of this class of control schemes can be looked at from the following disciplines (amongst others!):

- Computer science: for instance the exploration of the implementation of algorithms and protocols.
- Control Theory: the algorithms themselves fall into stochastic control theory.
- Game Theory: although the real system has a lot of inherent 'noise', it is possible to pose certain abstractions as well-posed games, both co-operative and non co-operative.
- Economics: exploring the relationship between prices and optimality.
- Financial mathematics: by viewing the network as a 'smart-market' with prices reflecting valuation of assets, financial derivatives relate to questions of reservation.
- Stochastic Decision Theory.
- Evolutionary Biology.
- Optimisation / Dynamic Programming.

The specification of an 'optimal' algorithm is critically dependent upon what criterion is used to determine optimality. Here we explore optimality from the users' perspective, where loosely the aim is to maximise information transfer at minimum cost. We shall use the general framework given above, where a user has some utility associated with bandwidth, and costs are incurred, where costs are proportional to the number of packets marked. If $x_r(t)$ is the load (rate) generated at time t by user r, and the number of packets marked is $f_r(t)$, with cost factor α, then cost is incurred at rate $\alpha f_t(t)$. In the following we drop the dependence of x and f on r for clarity, and where there is no confusion write x_t instead of $x_r(t)$ etc. The objectives are shown in discrete time with the obvious extension to continuous time. Typical objectives might be:

1. **Control Costs** The simplest objective would be to keep the expected rate of cost constant — if this is w, which Kelly terms a 'willingness to pay', then in terms of Decision theory, the objective is

$$\text{Minimise } L\left(w - \alpha f_t\right) \tag{12}$$

   for some loss function L. If the user is attempting to chose w perhaps via some proxy function, then with the quadratic loss function ($L(x) = x^2$) this is minimised when $w = \mathbb{E}\left(\alpha f_t\right)$, which is an alternative interpretation of the 'elastic scheme' described earlier.

2. **Throughput** Maximise the average throughput minus the cost, maximise $\sum_{t=1}^{T}(x_t - \alpha f_t)/T$, for T finite, or as $T \to \infty$.

3. **Constrained Throughput** Maximise throughput given a cost constraint Δ, maximise $\sum_{t=1}^{T}(x_t/T)$ s.t. $\sum_t \alpha f_t \leq \Delta$.

4. **Discounted Throughput** This allows an infinite time horizon: $\max(1-\beta)\sum_{t=1}^{T}\beta^t(x_t - \alpha f_t)$ for some discount factor $0 < \beta < 1$.

5. **Maximise Net Utility** For given a utility function U, maximise $\sum_t\{U(x_t) - \alpha f_t\}$.

6. **File Transfer** Transfer an amount of data F (eg a file) at minimum cost, minimise $\sum_t \alpha f_t$ s.t. $\sum_t x_t = F$.

7. **Fixed Time File Transfer** Transfer file F in set time T at minimum cost, minimise $\sum_{t=1}^{T} \alpha f_t$ s.t. $\sum_{t=1}^{T} x_t = F$.

8. **Quick File Transfer** Transfer file F as quickly as possible at minimum cost.

Note that some of these are not necessarily well posed (eg the Quick File Transfer), but that all of them are 'nice'. However this need not be the case. For instance a user could choose to do as much damage as possible to other users at minimum cost, which is a situation the network has to cope with, and is a valid objective in this sense.

5.1 A Sequence of Games

Initially it is sensible to look at a single objective, and have a tournament or competition based on this. For various reasons, the 'Fixed Time File Transfer' is a sensible starting point, as it is amenable to analysis. Good algorithms will compete against clones of themselves, and against users trying to achieve the same objective. The Network will have to provide some background load.

The next two stages are to use a defined heterogeneous environment, where the population of types of users is fixed, at least in probability, and finally the completely heterogeneous case, where users compete against each other and try to maximise their own objectives. The latter case emulates a real network where it is also important to study the robustness of good algorithms ('genotypes') as the population or environment evolves.

6 Conclusions and Further Work

We have argued that it is possible to provide differential quality of service to users by means of simple pricing scheme, where the network sends back the correct congestion prices to the user, and the users are then free to choose how to react.

We have presented a framework for assessing the effectiveness and stability of flow control schemes by constructing a distributed game, played with users against a 'Network' where the Network provides the environment (ecosystem). The protocols and mechanisms to play the game are currently being constructed, and will also be available for download [2].

Then the 'games' will begin!

Acknowledgement

It is a pleasure to acknowledge helpful discussions with Frank Kelly and Richard Gibbens.

References

[1] Kelly, F., Maulloo, A., and Tan, D.: 'Rate control in communication networks: shadow prices, proportional fairness and stability.' *Journal of the Operational Research Society*, 1998. **49** pp. 237–252

[2] Gibbens, R. and Kelly, F.: 'Resource pricing and the evolution of congestion control.'
http://www.statslab.cam.ac.uk/~frank/PAPERS/evol.html, 1998

[3] Odlyzko, A.: 'A modest proposal for preventing Internet congestion.'
http://www.research.att.com/amo/doc/modest.proposal.ps, 1997

[2] http://www.research.microsoft.com/users/pbk/pbk.htm

[4] MacKie-Mason, J. and Varian, H.: 'Pricing the Internet.' In Kahin, B. and Keller, J., editors, *Public Access to the Internet* (Prentice-Hall, New Jersey, 1994)

[5] Shenker, S.: 'Fundamental design issues for the future Internet.' *IEEE J. Selected Area Communications*, 1995. **13** pp. 1176–1188

[6] Jacobson, V.: 'Congestion avoidance and control.' In *Proc. ACM SIGCOMM '88*. 1988 pp. 314–329, pp. 314–329. An updated version is available via ftp://ftp.ee.lbl.gov/papers/congavoid.ps.Z

[7] Mathis, M., Semke, J., Mahdavi, J., and Ott, T.: 'The macroscopic behavior of the congestion avoidance algorithm.' *Computer Communications Review*, 1997. **27** (3)

[8] Axelrod, R.: The Evolution of Cooperation (Basic Books, NY, 1984)