

Some Observations on Fairness of Bandwidth Sharing

Dah Ming Chiu

Some Observations on Fairness of Bandwidth Sharing

Dah Ming Chiu

SMLI TR-99-80

September 1999

Abstract:

This paper reviews the engineering solutions and economic models for fair allocation of network bandwidth to elastic flows. By using examples, it gives insight to Proportional Fairness, and how it compares to the fairness achieved by TCP.

The second topic is a discussion of fair bandwidth allocation between multicast and unicast flows. By separating the discussion into economic and engineering viewpoints, this paper discusses ideal solutions and suggests practical approaches to achieving reasonable fairness.



M/S MTV29-01
901 San Antonio Road
Palo Alto, CA 94303-4900

email address:
dahming.chiu@east.sun.com

© 1999 Sun Microsystems, Inc. All rights reserved. The SML Technical Report Series is published by Sun Microsystems Laboratories, of Sun Microsystems, Inc. Printed in U.S.A.

Unlimited copying without fee is permitted provided that the copies are not made nor distributed for direct commercial advantage, and credit to the source is given. Otherwise, no part of this work covered by copyright herein may be reproduced in any form or by any means graphic, electronic, or mechanical, including photocopying, recording, taping, or storage in an information retrieval system, without the prior written permission of the copyright owner.

TRADEMARKS

Sun, Sun Microsystems, Java, and the Sun logo are trademarks or registered trademarks of Sun Microsystems, Inc. in the U.S. and other countries. All SPARC trademarks are used under license and are trademarks or registered trademarks of SPARC International, Inc. in the U.S. and other countries. Products bearing SPARC trademarks are based upon an architecture developed by Sun Microsystems, Inc.

For information regarding the SML Technical Report Series, contact Jeanie Treichel, Editor-in-Chief <jeanie.treichel@eng.sun.com>.

Some Observations on Fairness of Bandwidth Sharing

Dah Ming Chiu

Sun Microsystems Laboratories

Burlington, Massachusetts

1. Introduction

Background

The design of today's Internet was heavily influenced by the doctrine *Keep It Simple*. The network concentrates on routing and forwarding packets, and wastes no time in deciding how to regulate traffic. To control congestion, simple algorithms are used in the transport (TCP) in the end systems to react to congestion. Relying on the elastic nature of much of the traffic, the sharing model is to accommodate all incoming traffic and to degrade service gracefully and fairly when there is congestion, rather than to adopt admission control as the telephone system does. As a result, there is no need for any other bandwidth sharing mechanisms; the economic policy of bandwidth sharing is built into the transport algorithms, almost as a side effect of congestion control. Based on the success of the Internet, this minimalist approach has worked very well.

The engineering

There was little study of the economic ramifications of the bandwidth sharing policy affected by the congestion control algorithms at the time they were designed. The simple notion of sharing the bottleneck bandwidth fairly was the only guiding consideration. The fairness of TCP-style congestion control algorithms was first analyzed in [Chiu]. The rather awkward terminology of *Additive Increase Multiplicative Decrease*¹ used to describe these algorithms stuck, and is often used to refer to the TCP congestion control algorithms by the performance analysis community. The analysis in [Chiu] was based on a single bottleneck resource. The generalized model of network fairness is the *Max-Min fairness*, which had been discussed in earlier literature and described clearly in [Bertsekas]. For an Additive Increase Multiplicative Decrease algorithm to approximately result in Max-Min fairness, it would require the competing flows to traverse the same number of links (or have approximately the same roundtrip time).

It is known that TCP favors short flows (defined as flows that pass through a small number of links, or with a smaller roundtrip time). Since TCP is window-based, each flow's minimum window size is 1. Under congestion, the window sizes for short flows have a lower bound of 1 whereas the windows for long flows are reduced much more (proportionally); with more equalized window size, short flows achieve higher throughput than long flows due to the shorter roundtrip time to refresh the window. Closed-form equations of TCP's throughput have been studied under static conditions [Floyd], [Mathis], [Padhye]. It roughly approximates to

¹ A more sensible terminology, used by some people today, is *linear increase/exponential decrease*.

$$x = \frac{c}{T\sqrt{q}}$$

where T is the roundtrip time, q is the fraction of packets lost, and c a constant. If the value of q is approximately the same for all flows sharing the same bottleneck, then the ratio of the throughput of two TCP flows is inversely proportional to the ratio of their roundtrip times.

These engineering successes have led many to believe (apparently) that TCP fairness is the perfect policy for elastic traffic to share network resources, and new protocols must strictly emulate TCP.

The economics

More recent research re-examined the economic models of network bandwidth sharing. In [Kelly1], [Kelly2] and [Gibbens], Kelly and others proposed a model of optimizing a utility function associated with network flows. Kelly further applied operations research techniques to solve the optimization problem, and illustrated that different notions of fairness would result from different utility functions. In particular, when applying a logarithmic (or *diminishing returns*) utility function for each flow, the resultant solution tends to favor short flows more than long flows. Kelly terms this *Proportional Fairness*, and proposes that this is a more suitable form of economic policy (fairness) for bandwidth sharing. In applying a Lagrangian technique for solving the optimization problem, Kelly was also able to apply other economic interpretations to his models; for example, certain variables in the model correspond to shadow prices and usage charges of network bandwidth.

Subsequently, [Massoulié] took Kelly's framework and considered other possible utility functions, such as maximizing the sum of the throughputs of all flows, or minimizing the sum of the delays (inverse throughput) of all flows. The comparison of the resultant sharing policies (fairness criteria) to those of Max-Min and Proportional fairness shed light on the diversity of policies that might be considered.

This paper

Our observations in this paper discuss some misconceptions that seem to come up when trying to explain successful engineering with economic interpretations, and when applying the elegant economic models to engineering reality.

The first topic of discussion involves a claim that TCP implements Proportional Fairness. This claim is euphoric in that it brings together the engineering success and a newly found economic theory. We show that while what TCP implements and Proportional Fairness have some similar properties, they are different; we give a counter example to illustrate this. To further explain the difference, we sketch a constructive algorithm that derives a Proportional Fair solution from a Max-Min solution.

The second topic of discussion involves a widely proposed policy that a multicast flow should be TCP friendly in its bandwidth sharing algorithms. Since TCP is believed to implement Proportional Fairness, some argue that the bandwidth share allocated to a multicast flow should be proportionally fair in the Kelly sense. Using Kelly's model, we explain the meaning of making multicast proportionally fair. This demonstrates the naivete of applying proportional fairness to flows that justifiably should have different utilities than a regular TCP flow. This leads us to

discuss the question of how unicast and multicast flows should share network bandwidth, from both an economic and engineering point of view.

2. TCP and Proportional Fairness

TCP is observed to favor short flows in its resultant bandwidth allocation. Proportional Fairness suggests allocating more bandwidth to shorter flows in many scenarios. Therefore, does TCP implement proportional fairness? Not necessarily.

Proportional Fairness

We briefly review Kelly's derivation of Proportional Fairness.

The network has a set of J resources, and C_j is the finite capacity of resource j . A flow i is represented as a non-empty subset of J , and the set of all flows is I . Set $A_{ji}=1$ if flow i goes through resource j and set $A_{ji}=0$ otherwise. This defines a 0-1 matrix A . Associate each flow with a user, and suppose that if a rate x_i is allocated to user i then this has a utility $U_i(x_i)$ to the user. Assume that utilities are additive, so that the aggregate utility of all users is the sum of each user's utility; then the economic problem is to maximize the aggregate utility:

$$\begin{aligned} & \text{Maximize} && \sum_{i \in I} U_i(x_i) \\ & \text{Subject to} && Ax \leq C \\ & \text{over} && x \geq 0 \end{aligned}$$

Any rate allocation x that satisfies $Ax \leq C$, and $x \geq 0$ is called a feasible allocation.

Assuming elastic traffic [Shenker], a reasonable utility function for $U()$ is $\log()$. Using this utility function and Lagrangian multiplier techniques, Kelly found that the solution to the above problem must satisfy the following criteria: x must be feasible, and for any other feasible allocation y :

$$\sum_{i \in I} \frac{y_i - x_i}{x_i} \leq 0$$

Such an allocation x is called *proportionally fair*.

This definition of the proportional fairness solution is descriptive but non-constructive. An often used example (for example in [Massoulié]) is as shown in the following figure:

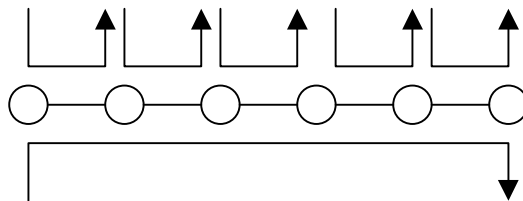


Figure 1. Proportional Fairness Example

There are n concatenated links, and $(n+1)$ flows in the network. Each link has bandwidth of 1 unit. Flow 0 goes through all the links; flow 1 goes through link 1; flow 2 goes through link 2;

and so on, and flow n goes through link n as shown in Figure 1. The Max-Min fair solution for this network is to assign the same rate to all the flows:

$$x_0 = x_1 = x_2 = \dots = x_n = \frac{1}{2}$$

where x_i is the rate allocated to flow i . The Proportional Fairness solution, on the other hand, is to allocate the bandwidth as follows:

$$x_0 = \frac{1}{n+1}$$

$$x_1 = x_2 = \dots = x_n = \frac{n}{n+1}$$

Let us take for granted that this solution is correct (a derivation will be shown later). You can check this solution against the necessary and sufficient conditions for Proportional Fairness stated earlier for verification.

A quick observation one can draw is

$$\frac{x_0}{x_i} = \frac{1}{n} \quad \forall i$$

Or the bandwidth allocation is (inversely) proportional to the length of the flow. Without going through more examples, it is tempting to generalize this observation for this particular example as a general property of Proportional Fairness.

Counter example

Let us consider a slightly different scenario as depicted below:

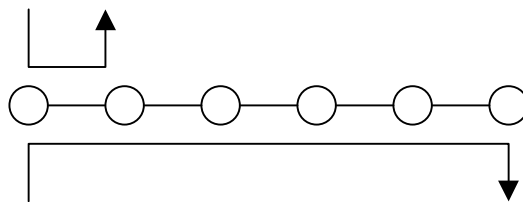


Figure 2. Proportional Fairness Counter Example

The network in this example is the same as in the previous example, but there are only two flows. Flow 0 still traverses all links, while flow 1 goes through only link 1 . This time the Proportional Fairness solution is the same as the Max-Min fair allocation:

$$x_0 = x_1 = \frac{1}{2}$$

This shows that the “property” that the allocation is inversely proportional to the length of the flow is not generally true.

In order to ensure that we have the correct proportional fair solutions for these two examples, let us go through the derivations as they are relatively simple for these examples.²

For the example in Figure 1, by symmetry, the solution for x_1 through x_n must be the same, and we let that be equal to p . This means x_0 equals $(1 - p)$. Kelly’s utility function thus becomes

$$\sum_0^n U_i(x_i) = \log(1 - p) + n \log(p) = f_1(p)$$

Since this is a strictly concave function, the maximum occurs when the derivative of this function is 0:

$$\frac{df_1}{dp} = -\frac{1}{1 - p} + \frac{n}{p} = 0$$

Hence

$$p = \frac{n}{n + 1}$$

For the example in Figure 2, however, the utility function becomes

$$\sum_0^1 U_i(x_i) = \log(1 - p) + \log(p) = f_2(p)$$

Hence the different proportional fair solution.

In fact, in Kelly’s original paper [Kelly1], it was clearly stated on page six that “...if each flow x_s passes through a single bottleneck, then the solution x is necessarily Max-Min fair.” So this counter example is only applicable to a misconception that is not Kelly’s.

Kelly went on to say “...This conclusion does not, however, apply when flows pass through multiple bottlenecks.” And these are indeed the scenarios when proportional fairness differs from Max-Min fairness. Here lies the intuitive explanation. Only when a connection passes through multiple bottlenecks does it have the potential of *sacrificing* its own utility with a greater return in terms of social welfare (the utility of all flows as sum).

A future research exercise is to study both randomly generated networks and real networks to analyze how often flows go through multiple bottlenecks.

² The derivation of a slightly more general form of this problem is in [Massoulie]. We are including the simple derivation for convenience rather than novelty.

Constructing a proportional fair solution

Not all networks are amenable to the kind of simple analysis afforded by examples 1 and 2 (illustrated in Figures 1 and 2) above. Is there a simple way to construct a proportional fair solution by iteration rather than by solving the mathematical model? Here we discuss a simple idea and a sketch of a centralized algorithm.

There is a centralized algorithm to derive a Max-Min solution without much complexity, for example see a description in [Bertsekas]. The starting point of our algorithm is the Max-Min solution. At this point, each flow makes an altruistic move, starting from the longest flow. If the flow goes through a single bottleneck, it is exempted; otherwise, it gathers all other flows competing with it (for the multiple bottlenecks) and finds an amount of reduction in its own flow that would cause the maximum increase in the total utility of this subset of flows.³ For relatively simple network scenarios, we conjecture that this algorithm will converge quickly to a proportional fair solution.

The problem is, of course, the cyclic situations as illustrated by the following example:

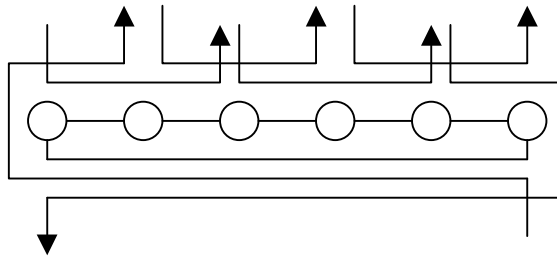


Figure 3. Example of a network of cyclic flows

When a cyclic flow pattern is discovered, it should help the altruistic flow to terminate its offer to reduce rate appropriately. The details of this aspect of the algorithm have not been fully studied and are left for further research.

3. Bandwidth Sharing between Unicast and Multicast Flows

Multicast in the Internet, known as IP multicast, experienced its share of success failure. Its success was in demonstrating several realistic and suitable applications that can greatly benefit from the multicast technology⁴: live audio/video channel dissemination to widely spread receivers, and collaboration among widely separated users. Due to whatever reasons, it also earned a bad reputation for hogging the network bandwidth and negatively impacting the operation of the rest of the Internet. Perhaps because of this reputation, network managers are often very cautious with further experimentation with multicast. In addition, the Internet Engineering Task Force (IETF) orchestrated a very strong guideline for developing a TCP-friendly multicast congestion control scheme [RFC2357], realizing that congestion control is where network bandwidth sharing policies are implemented on the Internet today.

The IETF guideline suggests that a multicast flow be treated as a unicast flow. The actual interpretation of how to make the unicast equivalent of the multicast flow TCP-friendly has been

³ This can be thought of as the point of diminishing returns for further sacrifice.

⁴ We are referring to the experimental services provided by the MBONE.

a subject of research, often a topic of heated debate in the Reliable Multicast Research Group (chartered by the IETF).

We give our observations in two parts below. First, we take an economic perspective on this question. The emphasis in this case is not on the practicality for implementation, but on explaining different alternatives based on reasonable (global) models. Then, we get into the engineering aspects by first discussing some of the current proposals, then propose a new idea.

The economic discussion

Let us take Kelly's model and consider what it means to make a multicast flow proportionally fair as if it were a unicast flow. First, consider the following simple example network:

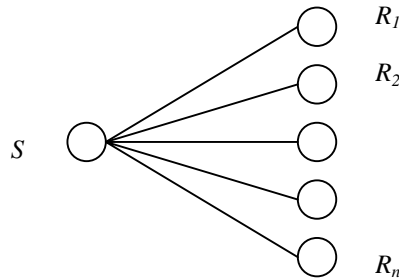


Figure 4. Multicast example

Suppose there is one multicast flow that goes from S to R_1, R_2, \dots, R_n . And there are n unicast flows from S to R_i for $i = 1$ to n , respectively. If we treat the multicast flow as if it were a unicast flow, then the application of Kelly's model to this problem reduces it to an identical problem as the problem in Figure 1, assuming the multicast flow is flow 0 . Hence the solution is also identical:

$$x_0 = \frac{1}{n+1}$$

$$x_1 = x_2 = \dots = x_n = \frac{n}{n+1}$$

This implies that the larger the multicast group the smaller its share of the proportional bandwidth it would get. We are confident that no one would consider this as fair, simply from intuition.

To recapitulate, we started wanting to make multicast share bandwidth fairly with unicast; since proportional fairness is a good criterion for unicast fairness, we applied that to multicast; what we got is intuitively unfair. So, what went wrong?

The mistake is to assume that the utility function for the multicast flow is the same as that of a unicast flow.

For the sake of argument, let us consider a multicast flow that goes to n receivers gives the same utility as if the n receivers had n different senders. This changes the sum of the utilities to be

$$\sum_0^n U_i(x_i) = n \log(1-p) + n \log(p) = f_4(p)$$

and the optimal p for this problem is the same as that using $f_2(p)$, hence

$$x_0 = x_i = \frac{1}{2} \quad \forall i$$

This now seems to be a quite satisfactory solution, intuitively. As we learned our lesson from the previous section, it is quite dangerous to make a conclusion based on one example.

Let us consider another example where we have the same network as in Figure 4, but only two flows: the same multicast flow plus only one unicast flow from S to R_I . Again, assuming the utility of the multicast is the same as n unicast flows, the sum of the utility is now:

$$\sum_0^1 U_i(x_i) = n \log(1-p) + \log(p) = f_5(p)$$

Applying the same technique, we see the solution is

$$x_1 = p = \frac{1}{n+1}$$

$$x_0 = \frac{n}{n+1}$$

This time, the multicast flow gets n times the bandwidth of the sole unicast flow that is competing with it. Somehow, this does not seem intuitively right.

It is not clear that the utility of n receivers receiving a multicast flow is the sum of (n times) the utility of each receiver receiving from a unicast flow. There are various efforts and costs that are associated with being part of a multicast session. A receiver has to receive at the same time, the same rate, and exactly the same content as other receivers, which may not necessarily be convenient for any given receiver. There are perceived cost-savings from *sharing a ride* such that if a multicast receiver is asked to pay the same amount as a unicast receiver, it may not feel it is justified. For a variety of reasons of this kind, it seems economically not quite appropriate to equate a multicast to n unicast flows. Therefore, a sensible utility function of a multicast is probably something in-between the utility of a single unicast flow and n times of a single unicast flow. A recent paper [Legout] studied a number of possibilities, among them is a utility function that depends on $\log(n)$ instead of n . More research on this topic should perhaps try to factor in the costs associated with multicast as we alluded to earlier.

Once a function of n is added to the multicast flow's utility function, the resultant fairness solution becomes a *Weighted Proportional Fairness* solution as described in Kelly's study (with the weight being a function of n). An approximate interpretation of weighted proportional fairness is that it gives the flows with weights in their utility functions a larger share of the congested bandwidth, in an amount roughly proportional to the weights.

Our economic observation is non-conclusive. By showing two extreme examples, we seemed uncertain in what we are trying to say. Actually there is a clear message; that is, from an economic point of view, the key is to know the utility function for a multicast flow. When an appropriate utility function for multicast flows is used, the resultant fairness solution is then *incentive-compatible*.

The engineering discussion

The primary focus of the following discussion is not to arrive at an engineering solution, but to discuss what constitutes a set of reasonable fairness requirements for implementing multicast congestion control.

The challenge is to adhere to the *Keep it simple* doctrine, yet be as incentive-compatible as possible.

Observation 1: can't afford to use n

One key factor limiting the design of an engineering solution is the following. Multicast in the Internet (known as IP multicast) is currently designed as an *anonymous* service. In other words, the network does not tell the sender how many receivers are receiving its data. This design is clearly consistent with the *Keep it simple* doctrine. This means, however, that even if we come up with a perfect utility function for an n -receiver multicast flow (presumably a function of n), it is difficult if not infeasible to implement the corresponding fairness solution in a congestion control mechanism.⁵ So our first engineering observation is that for the sake of network simplicity, it is acceptable to forego the use of a weight for the multicast utility function that is dependent on n . But at the same time, we take note that we are possibly mistreating the multicast flows.

Observation 2: representative is okay

Most proposals for multicast congestion control fairness amount to the following. Given a multicast flow with n receivers, try letting only one receiver work with the sender to implement a TCP-compatible congestion control algorithm,⁶ and repeat this for all receivers. Then pick the slowest throughput (produced by one *representative* receiver working with the sender) as the multicast flow's fair share. See [Strawman] as an example proposal.

Consider applying this representative approach to the two extreme examples based on the network in Figure 4 that we discussed earlier. For both examples, the multicast flow gets allocated half of the bottleneck bandwidth. This is not too bad, although in the two-flow example we would like to see the multicast flow get a higher share (but not as much as n times).

Observation 3: relaxed TCP emulation

Once we assume there is a representative,⁷ we have roughly reduced the multicast flow down to a unicast equivalent,⁸ and the question is how this unicast equivalent should share bandwidth with other flows. The TCP *purist* would advocate an exact emulation of TCP. This is clearly attempted by [Strawman] for example. We suggest this is not necessary for two reasons:

⁵ Of course, one can go through the trouble of building a separate service that derives the receiver population information and feeds it into the congestion control mechanism somehow. There are also some engineering proposals for the multicast network to keep receiver population information for accounting purposes.

⁶ These proposals vary in how the *representative* receiver and the sender actually implement a TCP-compatible congestion controller. Some use almost the exact TCP windowing mechanism; some use a rate-based equivalent approach; some sample network loss and use a formula to determine the TCP-equivalent rate; yet others use a combined window and rate-based approach. For our discussions, the actual mechanism is not important.

⁷ This is a logical notion for the discussion of fairness. We are not mandating that a reasonable protocol must explicitly elect a representative, as long as the same kind of behavior is achieved.

⁸ We say roughly because the multicast flow still incurs traffic that is not on the path from the sender to the representative receiver. For the engineering discussion, we are not going to be this exact.

1. Recall in observation 1 above we explained that for simplicity reasons, these (TCP-friendly) proposals do not use a receiver population-based weighting factor, hence under-allocating bandwidth to multicast flows. Therefore, we argue we should go out of our way now to make sure we treat the unicast equivalent (of the multicast flow) as fair as possible. Specifically, we should allow the multicast flow to get an equal share of the bottleneck bandwidth (Max-Min fairness) if that is easy to implement engineering-wise.
2. The effort that goes into emulating TCP exactly can be high. In adopting a TCP-formula based approach (for example [Strawman]), it should not be necessary to use a measured roundtrip time (corresponding to the unicast equivalent flow). Instead, we suggest that a roundtrip time that roughly corresponds to the bottleneck delay would suffice. This ensures that the unicast equivalent gets no less bandwidth than other competing TCP flows.

Of course, if emulating TCP exactly is the easiest way to implement in your protocol, that should also be allowed.

Observation 4: Max-Worst fairness

Let us call the fairness criterion discussed here *Max-Worst* fairness (for multicast). For simplicity, the "*Worst*" corresponds to a decision not to include a weighting factor that is a function of the receiver population in the solution. The "*Max*" corresponds to the extra effort in attempting to ensure fairness for the unicast equivalent of the multicast flow.

Let us illustrate the Max-Worst fairness with an example:

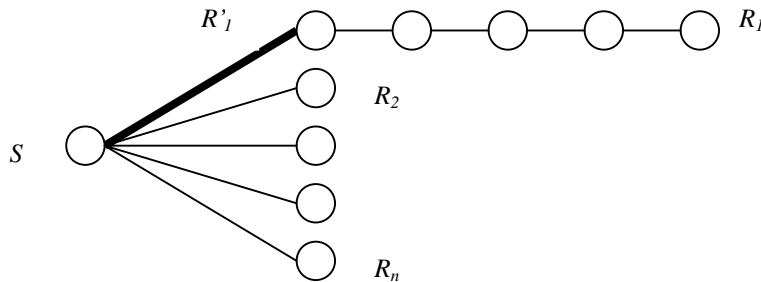


Figure 5. Example to illustrate TCP emulation with Max-Worst fairness

As in earlier examples, there is a single multicast flow with n receivers, except receiver R_l is m links further away from the sender. The darker link is the bottleneck where n other flows traverse. Because of the bottleneck location, the unicast equivalent is the path from S to R_l . When competing with other TCP flows traversing the bottleneck, the Max-Worst fairness lets the unicast equivalent *pretend* that R'_1 is the receiver in order to ensure the multicast flow gets at least $1/n$ of the bottleneck bandwidth.

4. Conclusions and Future Work

In this paper, we explained some of the subtle differences between Max-Min fairness, Proportional Fairness, and the fairness TCP implemented. In doing so, we distinguished between an economic view and an engineering view of the subject: the former tries to prescribe, often abstractly and taking a global view, whereas the latter implements, often approximately, what is feasible or simple. We clarified some misconceptions that often come up in engineering

discussions. For the difficult problem of allocating bandwidth to both multicast and unicast flows, we presented some observations, both from an economic and an engineering point of view.

At various places in the paper, we suggested areas for further research. For example:

- The work on a procedure to derive a Proportional Fair solution starting from a Max-Min fair solution is sketchy and incomplete.
- Analysis of how often and by how much Max-Min and Proportional Fairness differ, in both random and selected real networks, would be interesting information.
- Investigation into what constitutes a suitable multicast utility function that factors in both the receiver population, as well as some of the costs associated with multicast in comparison to its unicast equivalents, needs to be undertaken.
- The engineering approach to multicast congestion control is still an active area of research, and much work remains to be done.

Acknowledgements

The author wishes to thank Professors Frank Kelly and Jon Crowcroft for valuable discussions; and thanks to Guy Steele, Miriam Kadansky, Joe Wesley, and Suchi Raman for various suggestions.

5. References

- [Bertsekas] D. Bertsekas and R. Gallager, *Data Networks*. Prentice Hall, 1987.
- [Chiu] D. M. Chiu and R. Jain, Analysis of the increase and decrease algorithms for congestion avoidance in computer networks. *Computer Networks and ISDN Systems*, Vol 17, pp 1-14, June 1989.
- [Floyd] S.Floyd and K. Fall, Promoting the use of end-to-end congestion control in the Internet. <http://www-nrg.ee.lbl.gov/floyd/end2end-paper.html>, 1998.
- [Kelly1] F. Kelly, Charging and rate control for elastic traffic, *European Transactions on Telecommunications*, Vol 8, pp 33-37. 1997.
- [Kelly2] F. Kelly, et al., Rate control for communication networks: shadow prices, proportional fairness and stability, *Journal of the Operational Research society*, 49:237-252, 1998.
- [Gibbens] R.J.Gibbens, F. Kelly, Resource pricing and the evolution of congestion control, <http://www.statslab.cam.ac.uk/~frank/evol.html>, 1999.
- [Legout] A.Legout, et al., Bandwidth Allocation Policies for Unicast and Multicast Flows, INFOCOMM 99, 1999.
- [Massoulié] L. Massoulié and J. Roberts, Bandwidth sharing: objectives and algorithms, INFOCOMM 99, 1999.
- [Mathis] M.Mathis, et al., The macroscopic behavior of the TCP congestion avoidance algorithm. *Computer Communication Review*, 27, 1997.
- [Padhye] J. Padhye et al., Modeling TCP throughput: A simple model and its empirical validations. *Proceedings of SIGCOMM 98*, 1998.
- [RFC2357] A Mankin, et al., IETF Criteria for Evaluating Reliable Multicast Transport and Application Protocols, *IETF RFC2357*, June 1998.

[Shenker] S. Shenker, Fundamental design issues for the future Internet, *IEEE Journal Selected Areas Communication* 13, 1176-1188, 1995.

[Strawman] M. Handley and S. Floyd, Strawman Specification for TCP Friendly (Reliable) Multicast Congestion Control (TFMCC), working paper of Reliable Multicast Research Group.

About the Author

Dah Ming Chiu is a researcher at Sun Microsystems Laboratories in Burlington, Massachusetts. His recent research is on reliable multicast, and multicast flow/congestion control. Prior to Sun, he worked at Digital Equipment Corporation, and AT&T Bell Labs. His research interests include performance modeling and analysis of network protocols, distributed systems, and WWW-based applications. He received a Ph.D. degree from Harvard University and a B.Sc. degree from Imperial College, University of London.