An Algorithm for Rate Allocation in a Packet-Switching Network with Feedback

by

Anna Charny

B.S., Mathematics, Moscow Institute of Physics and Technology, Russia, 1979

M.S., Mathematics, Kalinin State University, Russia, 1985

Submitted to the Department of Electrical Engineering and Computer Science
in Partial Fulfillment of the Requirements for the Degrees of

Master of Science in Electrical Engineering and Computer Science

and

Electrical Engineer

at the

Massachusetts Institute of Technology

May 1994

Signature of Author ______________________________

Anna Charny

Certified by ______________________________

Dr. David D. Clark, MIT Thesis Supervisor

Certified by ______________________________

Dr. Raj Jain, Digital Thesis Supervisor

Accepted by ______________________________

Frederic R. Morgenthaler, Chair, Committee on Graduate Students

ABSTRACT

As the speed and complexity of computer networks evolve, sharing network resources becomes increasingly important. Thus, the issue of how to allocate the available bandwidth among the multitude of users needs to be addressed. Such allocation needs to be in some sense efficient and fair to different users. In this work the so-called *maxmin fairness* is chosen as the optimality criterion. A new distributed and asynchronous algorithm is suggested. The algorithm is shown to converge to the optimal rate allocation in a network with general topology under dynamic changes in the set of network users, individual user load and occasional route changes. An upper bound on convergence time is given. The algorithm is shown to be well-behaved in transience. Unlike previous work, the algorithm takes bandwidth consumed by feedback traffic into account. Further, an extension of the algorithm is suggested to address the problem of policing misbehaved users.

Acknowledgements

I would like to thank my thesis advisors, Dr. David Clark and Dr. Raj Jain for all the fruitful and stimulating discussions and for the guidance they provided throughout this work.

I would also like to thank Digital Equipment Corporation for providing my financial support during graduate school through the GEEP program.

I am grateful to Lisa Felice, my supervisor at Digital Equipment Corporation for being so helpful and supportive.

I am also grateful to William Duane, my Technical Sponsor at Digital Equipment Corporation for his help and his interest in my work.

I would like to thank the staff of the GEEP program at Digital Equipment Corporation for their help and efficiency.

I thank my office mates Oumar Ndiaye and Tim Shepard for being so nice and friendly.

I also thank Tim Shepard and Chris Lefelhocz for their help and patience in answering my many questions.

Contents

1 Introduction

This section discusses design decisions adopted in this work, describes the existing results for the chosen model, summarizes the main results of this work and finally gives the brief layout of the remaining sections.

1.1 Background

There has been extensive debate in the literature about the relative merits and drawbacks of open-loop control schemes versus closed-loop control schemes. The large propagation delay to packet transmission time ratio in the modern high-speed networks poses a significant challenge for any end-to-end feedback scheme [21], [24], [25]. As a result, a number of open-loop alternatives like prior reservation and switch-based controls have been suggested.

Prior reservation schemes are generally considered to be suitable for steady stream-like traffic with a priori known resource requirements. Reservation also provides quality of service guarantees that are difficult to achieve with walk-in service. The price for this, however, is the lack of flexibility in the presence of dynamic changes in the network load leading to potential waste of precious network resources.

Switch-based controls have been shown to be necessary for achieving fairness [27]. However, if no source-based control is exercised, the sources may continue to inject excessive traffic into the network, causing more overload and wasting network resources.

The approach adopted in this work is based on the cooperation between the sources and the network in sustaining an acceptable network load from the standpoint of fairness and efficiency. This approach is similar to that of [23] and [26].

The problem of load allocation is twofold - the sources must be able to determine their optimal load, and the network must ensure that even if all sources operate at their optimal rates, these rates are enforced across the network. The mechanism of such enforcement strongly depends on the shape of source traffic, a particular flow control mechanism, and service discipline of all switches in the network. There is a vast amount of work addressing the issue of preserving feasible user rates under various assumptions on the underlying service discipline and the shape of source traffic. (see for example [3], [11], [12], [14], [25]).

This thesis addresses the first problem, i.e. how to determine the set of optimal rates in a distributed network under dynamic changes in the absence of centralized knowledge about the network and without synchronization of different network components.

We consider a system in which switches maintain their own controls. Switches communicate these controls to the source by feedback. We consider an end-to-end feedback scheme, in which the destination generates feedback packets which deliver the aggregate feedback signal from all switches on the packet's route back to the source. Upon receipt of the feedback signal, the source adjusts its load accordingly. The details of the algorithm are discussed later in this work.

We show that the algorithm is very general in nature and is applicable to a broad range of service disciplines and underlying traffic shapes. This flexibility is largely due to our choice to decouple the problems of determining the optimal rates and enforcing them.

1.2 Route Selection

It is assumed that at any time of the algorithm operation the route of each session is unique. We allow the route to change from time to time (for example in response to equipment failures or due to some routing decisions), but we disallow existence of more than one route at any given time. We assume that the changes in the route do not occur very often, and that in the absence of network failures the routes will eventually stabilize for a given set of network users.

The obvious argument against this approach is that the best load allocation for a particular choice of session routes may not be the best over all possible route choices. However, even if session routes are unique, the problem of optimal rate allocation is non-trivial and deserves proper attention. In addition, note that the algorithm is shown to be robust in the presence of dynamic route changes. Thus, it can be run in conjunction with any independent routing algorithm which will eventually stabilize to some route. As soon as the route is found, our algorithm will recover from any past changes and will converge for the optimal rates for this route.

1.3 Network Model

In the real world endnodes are interconnected through a complex network of switches. There can be many users physically located at one network node, each of them conducting perhaps several communication sessions with other users in the network. Some of those sessions can be bi-directional like a conversation, other can be uni-directional, like file transfer.

For our purposes, we assume that all sessions are independent. Moreover, we simply treat one user conducting several sessions as several independent users. Similarly, we treat any bi-directional data exchange as two independent uni-directional ones.

We assume that any two connected nodes in the network are connected by a pair of half-duplex links of identical capacity pointing in the opposite directions. In general different link pairs have different capacities.

It is assumed that each enduser is connected to exactly one switch.

Endusers are not connected directly to each other. A switch can be connected to zero or more endusers of any type and to zero or more other switches. It is assumed that there is a path going through one or more switches from any source to its respective destination.

It is convenient to make a distinction between the 'entry' links into the network and all other links. The 'entry' links are in essence artificially created in the model to separate different users located at one network node. While capacities of all other links are real physical restrictions, capacities of the 'entry' links can be chosen as we please as long as they do not impose additional restrictions on session flows. It will be seen later that it is convenient to consider these capacities to be equal to the session demand. Thus we allow these capacities to be infinite if session demand is infinite. Capacities of all other links are assumed finite.

Similarly, the model creates an artificial switch per endnode located at the entry into the network. This switch has two 'real' half-duplex links connecting it to the network and $2m$ artificial half-duplex links, connecting it to m endusers located at the real-life endnode.

1.4 Optimality Criterion

The goal of this work is to determine a fair and efficient rate allocation . The precise meaning of the terms 'efficient' and 'fair' has been a target of extensive debate in the last two decades. References [1], [21], [24], [8], [7] contain a variety of approaches and definitions of fairness and efficiency.

The approach adopted in this work chooses the so-called *maxmin* or *bottleneck* optimality criterion discussed in various modifications in [1], [12], [17], [23], [26].

This approach is based on the following intuition.

Consider a network with given link capacities, the set of sessions and fixed session routes. We are interested in such rate allocations that are feasible in the sense that the total throughput of all sessions crossing any link does not exceed the link's capacity. We would like the feasible rate allocation to be fair to all sessions. On the other hand we want the network to be utilized as much as possible.

We now define a fair allocation in the following way. We consider all "bottleneck" links, i.e the link with the smallest capacity available per session. We give a strict definition of it in section 2. We share the capacity of these links equally between all sessions crossing them. Then we remove these sessions from the network and reduce all link capacities by the bandwidth consumed by the removed sessions. We now identify the "next level" bottleneck links of the reduced network and repeat the procedure. We thus continue until all sessions are assigned their rates.

Such rate vector is known as *maxmin fair* allocation. The above global synchronized procedure for achieving maxmin optimal rates is well known and is described for instance in [1], [23].

It can be easily seen that the rate allocation obtained in such a way is fair in the sense that all sessions constrained by a particular bottleneck get an equal share of this bottleneck capacity. It is also efficient in the sense that given the fair allocation, no more data can be pushed through the network, since each session crosses at least one fully saturated link.

Assuming that packets are infinitely small and the flows are deterministic, it can be seen that maxmin fairness implies maximum efficiency in the sense that the bottleneck resource is utilized up to its capacity and no queues build up.

It is well known, however, that for packets of finite size and the general distribution of packet arrival and service times utilizing the link to its full capacity leads to infinite queue growth and causes severe performance degradation. Thus, for general distribution of arrival and service times utilizing the bottleneck to its full capacity is not good for efficiency. Thus, in this case a different efficiency criterion is called for.

Reference [26] introduces the optimal efficiency criterion for a general network configuration as the maximum power of the bottleneck resource, where

$$Resource\ \ Power = \frac{Bottleneck\ \ Resource\ \ Throughput}{Bottleneck\ \ Resource\ \ Response\ \ Time}$$

The resource capacity at which the power is maximized is called the *knee capacity*. In general, the knee capacity depends on the particular distribution of the packet arrival times and service discipline.

However, if the knee capacity is known, then applying the global procedure for determining maxmin fair rates described above to the network with the *knee capacities* replacing the original capacities, we can obtain the rate allocation which is fair in the sense that the bottleneck resources are still shared equally among their users and efficient in the sense that the bottleneck resource power is maximized.

In summary, provided the knee capacities are known, we can use maxmin optimality on the network with knee capacities for both efficiency and fairness.

In practice, the knee capacities are not known a priori. As a result, either an a priori estimate is required, or an independent algorithm for congestion detection must operate in parallel to provide this estimate "on the fly"[26]. Another approach might be to combine the two by choosing some conservative estimate of the knee capacity and then attempt to adjust it if the link detects that it is constantly underutilized.

For the purposes of this work we assume that the knee capacities are known. Moreover, we will use the word "capacity" to mean the "knee capacity" unless otherwise indicated.

1.5 Service Discipline

The only assumption we make about the service discipline employed by the switch is that the packets of each session are served in FIFO order. Thus, the switches could be strict FIFO, FIFO+, Priority, Stop-and-Go, Fair-Queuing, etc. We emphasize that the reason for such flexibility is that the algorithm presented in this work is a *calculation* algorithm and is not concerned with enforcement of the rates. Such enforcement is strongly dependent on the service discipline.

In addition, we allow the switches to drop packets as they please, as long as at least some packets of each session continue to get through. While dropping packets can cause a lot of wasteful retransmissions, it is essential to note that our algorithm will still calculate correct optimal rates even in the presence of heavy packet loss. This property seems very important, since it means that the algorithm is robust in the presence of data loss due to heavy temporary congestion.

1.6 Previous Work and Summary of Results

The procedure for achieving maxmin optimal rates described earlier used global information, which is expensive and difficult to maintain in the real-world networks.

Several feedback schemes have been proposed to achieve the same goal in a distributed network. In essence, all these schemes maintain some link controls at the switch level and convey some information about these controls to the source by means of feedback. Upon receipt of the feedback signal the source adjusts its estimate of the allowed transmission rate according to some rule.

These algorithms essentially differ in the particular choices of link controls and the type of feedback provided to the source by the network.

References [6], [15], [17] describe distributed algorithms of this type. However, these algorithms required synchronization, which is difficult to achieve.

Mosley in [23] suggested an asynchronous algorithm for distributed calculation of maxmin fair rates. The algorithm was shown to converge to maxmin optimal rates. However, the algorithm convergence time was rather slow and simulations showed poor adaptation to dynamic changes in the network.

Later Ramakrishnan, Jain and Chiu in [26] suggested a distributed asynchronous algorithm for achieving maxmin optimal rates which uses a different type of feedback. The switches still calculate fair rate allocation for all sessions crossing its outgoing links, but this allocation is not explicitly communicated to the source. Instead, a bit is set in the packet's header if its current flow across the link exceeds the current value of the link's fair allocation. When the source receives packets with the bit set, it decreases its rate, otherwise it increases it. The algorithm has an attractive property of using just one bit in the packet header for feedback. It has been extensively tested in a variety of real-life network configurations and have been demonstrated to be fair and efficient even under dynamic network changes. However, while the simulation results are extremely favorable, no theoretical guarantees on the algorithm convergence to an optimal operating point in a general network topology are available. Moreover, since the optimal rates are not provided to the sources, the algorithm produces oscillations around the optimal rate and it may take a long time to get close to the optimal solution.

The approach adopted in this work requires explicit calculation of the optimal rates. It defines a family of link control calculation policies and a feedback mechanism which ensure convergence to maxmin optimal rates from any initial conditions. An algorithm employing any of these policies is shown to be self-stabilizing in the sense that it recovers from any past errors, changes in the set of network users, individual session demands, and session routes.

It is demonstrated that convergence of the algorithm is generally faster than that of the algorithms describe earlier in this section. An upper bound on convergence time is provided.

In addition, it is shown that the algorithm is 'well-behaved' in transience. In particular, it is shown that given an upper bound on round-trip delay, the actual transmission rates can be kept feasible throughout the transient stages of algorithm operation while still providing reasonable throughput to all users.

These qualities are extremely important in a dynamic network where changes in user load caused by newly arrived sessions can cause infeasibility which must be quickly taken care of to avoid large queue buildup and performance degradation.

We also suggest a mechanism for policing misbehaved users.

In addition, unlike previous work, we take into account the bandwidth consumed by feedback traffic.

Simulation results demonstrate that the algorithm works well under dynamic changes in the network load.

1.7 Outline

Section 2 contains the formal definition of the optimality criterion and provides a global procedure for determining optimal rates in the presense of real feedback traffic.

Section 3 contains the description of the distributed algorithm.

Section 4 gives the convergence theorem.

Section 5 discusses the transient behavior of the algorithm and provides an upper bound on convergence time.

Section 6 gives the results of several simulation experiments.

Section 7 contains a discussion on some of related issues and suggests an extension of the algorithm to policing misbehaved users.

Section 8 summarizes the results and gives some suggestions for future research.

2 Optimality Criterion

2.1 Definition of the MAXMIN Optimum

It seems natural to consider only static rate allocations for possible candidates for an optimal allocation. Once such optimal allocation is defined, our goal can be formulated as finding an algorithm to dynamically control an arbitrary rate allocation to bring it as close to the static optimum as possible.

We start with defining a feasible set of rate allocations η as follows:

$$\eta_i \geq 0 \tag{1}$$

$$\sum_{i \in \mathcal{G}_j} (u_{i,j} + k w_{i,j}) \eta_i \leq C_j \tag{2}$$

where η_i is transmission rate of session i, $u_{i,j} = 1$ if session i crosses j on its forward route and 0 otherwise, and $w_{i,j} = 1$ if session i crosses j on its feedback route and 0 otherwise, and $\mathcal{G}_j$ is the set of all sessions crossing link j.

(1) simply states that we are not interested in negative transmission rates, while (2) ensures that a rate allocation is such that no link capacity is exceeded.

Now we can define the optimality criterion on this feasible set as follows. We need the following definition first.

Definition 2.1 *Consider vector* $a = (a_1, \ldots, a_n)$*. Let* $\hat{a} = (\hat{a}_1, \ldots, \hat{a}_n)$ *be a permutation of* a *such that* $\hat{a}_i \leq \hat{a}_j$ *if* $i < j$*. Vector* b *is said to be* lexicografically greater *that* a *if either* $\hat{a}_1 < \hat{b}_1$ *or* $\exists\, 1 \leq j \leq n$ *s.t.* $\hat{a}_i = \hat{b}_i \;\; \forall\, 1 \leq i < j$ *and* $\hat{a}_i < \hat{b}_i$

Now we define the *maxmin optimal* vector of transmission rates by

Definition 2.2 *Vector* $\eta = (\eta_1, \ldots, \eta_S)$ *is called* maxmin optimal *for network* $\mathcal{N}$ *if*

- *it satisfies restrictions (1) and (2)*
- *it is lexicografically greater than any other feasible solution of (1) and (2)*

It can easily be seen that this definition in fact means that the optimal vector is such that its smallest component is maximized over all feasible vectors, then, given the value of the smallest component, the next smallest component is maximized, etc.

The next section describes a global procedure to obtain maxmin optimal rates for a network with feedback traffic.

2.2 Finding Globally Optimal Rates

In this section we give a way to find the stationary optimal vector η given global information about the network. The results here are quite similar to those given in [1], [23], [15], [26]. However, this work considers a somewhat different model than the cited authors, since our model accounts for the bandwidth consumed by feedback flows. Note that it is not clear a priori whether it is legitimate to treat feedback sessions in the same way as independent forward sessions, since their rates cannot be chosen independently from their corresponding forward sessions.

For the sake of simplicity we consider the case of "greedy" sessions, i.e. sessions with infinitely large demands. Note however, that the case of finite demands can be reduced to the "greedy" case by simply adding artificial links of capacity equal to the session demand at the entry of each session to the network.

We start with the following definition.

Definition 2.3 *Link l is called* bottleneck *with respect to network* $\mathcal{N}(\mathcal{L}, \mathcal{S})$ *if* $\frac{C_l}{f_l + kb_l} = \min_{j \in \mathcal{L}} \frac{C_j}{f_j + kb_j}$

Note that this is slightly different from the traditional definition of a bottleneck link. In our definition, we allocate a link's capacity between sessions (forward and feedback) sharing this link in such a way that each session is allocated $\frac{C_l}{f_l + kb_l}$ on its forward way.

Optimal stationary rates can now be found by the following procedure.

We find all bottlenecks link of the network and set the transmission rates of all the sessions crossing these links in either direction to $\frac{C_l}{f_l + kb_l}$ and mark those sessions. Then we decrease capacities of all links by the total capacity consumed by the marked sessions crossing these links on their forward or feedback paths. We consider a reduced network with all link capacities adjusted as above and with marked sessions removed. We repeat the procedure until all sessions are marked.

This procedure can be formalized as follows.

PROCEDURE *GLOBAL OPTIMUM*

Given network $\mathcal{N}(\mathcal{L}, \mathcal{S})$

START:

Denote:

$\hat{\mathcal{L}}_1$ - set of all links $l \in \mathcal{L}$ s.t. at least one session of $\mathcal{S}$ crosses l on its forward of feedback path

$\mathcal{L}_1$ - set of all links $l \in \hat{\mathcal{L}}_1$ s.t. $\frac{C_l}{f_l+kb_l} = \min_{j\in\hat{\mathcal{L}}_1} \frac{C_j}{f_j+kb_j}$

$\tau_1 = \frac{C_j}{f_j+kb_j}$ for any $j \in \mathcal{L}_1$

$\mathcal{S}_1$ - set of sessions crossing at least one link $\mathcal{L}_1$

f_l^1 - number of sessions of $\mathcal{S}_1$ crossing link l on forward path

b_l^1 - number of sessions of $\mathcal{S}_1$ crossing link l on feedback path

ITERATION i :

Given:

$\mathcal{S}_1, \ldots, \mathcal{S}_{i-1}$,

$\mathcal{L}_1, \ldots, \mathcal{L}_{i-1}$

$b_l^1, \ldots, b_l^{i-1}$

$f_l^1, \ldots, f_l^{i-1}$

$\tau_1, \ldots, \tau_{i-1}$

Define :

$\tilde{\mathcal{S}}_{i-1} = \mathcal{S}_1 \cup \ldots \cup \mathcal{S}_{i-1}$,

$\tilde{\mathcal{L}}_{i-1} = \mathcal{L}_1 \cup \ldots \cup \mathcal{L}_{i-1}$,

$\hat{\mathcal{L}}_i$ set of all links $l \in \mathcal{L} \setminus \tilde{\mathcal{L}}_{i-1}$ s.t. at least one session of $\mathcal{S} \setminus \tilde{\mathcal{S}}_{i-1}$ crosses this link on its forward or feedback path

$\mathcal{L}_i$ - set of all links $l \in \hat{\mathcal{L}}_i$ s.t.

$$\frac{C_l - \sum_{j=1}^{i-1} \tau_j (f_l^j + k b_l^j)}{f_l + k b_l - \sum_{j=1}^{i-1} (f_l^j + k b_l^j)} = \min_{q \in \hat{\mathcal{L}}_i} \frac{C_q - \sum_{j=1}^{i-1} \tau_j (f_q^j + k b_q^j)}{f_q + k b_q - \sum_{j=1}^{i-1} (f_q^j + k b_q^j)}$$

$\mathcal{S}_i$ - set of sessions of $\mathcal{S} \setminus \tilde{\mathcal{S}}_{i-1}$ crossing at least one link on $\mathcal{L}_i$

$\tau_i = \frac{C_l - \sum_{j=1}^{i-1} \tau_j (f_l^j + k b_l^j)}{f_l + k b_l - \sum_{j=1}^{i-1} (f_l^j + k b_l^j)} \;\; \forall l \in \mathcal{L}_i$

$\tilde{\mathcal{S}}_i = \mathcal{S}_i \cup \tilde{\mathcal{S}}_{i-1}$

$\tilde{\mathcal{L}}_i = \mathcal{L}_i \cup \tilde{\mathcal{L}}_{i-1}$

f_l^i - number of sessions of $\mathcal{S}_i$ crossing link l on forward path.

b_l^i - number of sessions of $\mathcal{S}_i$ crossing link l on feedback path.

If $\mathcal{S} = \tilde{\mathcal{S}}_i$, then STOP

Else perform iteration $i + 1$

END of *GLOBAL OPTIMUM*

Theorem 2.1 1. *Procedure* Global Optimum *terminates in a finite number of iterations.*

2. *When the procedure terminates, all sessions are assigned their globally optimal rates.*

3. *Let τ_i be the optimal rates assigned at iteration i. Then $\tau_1 < \ldots < \tau_m$*

4. *Let $\mathcal{L}_i$, $\mathcal{S}_i$ and τ_i be the set of bottleneck links of the reduced network of iteration i and sessions crossing these links respectively. Then any session in $\mathcal{S}_i$ crosses at least one link in $\mathcal{L}_i$*

5. *Only sessions from $\mathcal{S}_1 \cup \ldots \cup \mathcal{S}_i$ go through any link in $\mathcal{L}_i$ $\forall\, 1 \leq i \leq m$*

6. $\forall\ 1 \leq i \leq m$

$$\tau_i \begin{cases} = \frac{C_l - \sum_{j=1}^{i-1} \tau_j (f_l^j + k b_l^j)}{f_l + k b_l - \sum_{j=1}^{i-1} (f_l^j + k b_l^j)} & \textit{if } l \in \mathcal{L}_i \\ < \frac{C_l - \sum_{j=1}^{i-1} \tau_j (f_l^j + k b_l^j)}{f_l + k b_l - \sum_{j=1}^{i-1} (f_l^j + k b_l^j)} & \textit{if } l \in \hat{\mathcal{L}}_i \end{cases}$$

where $\hat{\mathcal{L}}_i$ is the set of sessions in $\mathcal{L} \setminus \mathcal{L}_1 \cup \ldots \cup \mathcal{L}_i$ s.t. least one session of $\mathcal{S} \setminus \mathcal{S}_1 \cup \ldots \cup \mathcal{S}_i$ crosses l.

The proof of this theorem is given in Appendix 2.

3 Distributed Algorithm

3.1 Assumptions and Goals

Section 2.2 provided a way to determine the optimal rates of a fixed fixed set of sessions using the global knowledge of the network. In addition, the global algorithm described there required synchronization of stages in which the optimal rates were assigned.

This section presents an algorithm to achieve the same goal in a distributed asynchronous way.

We start with a few words about the assumptions of the model and the goals of the distributed algorithm.

We now allow sessions to exit or enter as they please. However, to make the notion of optimal rates meaningful, we must assume that the sessions enter or exit not too often, in the sense that there is an extended period of time in which the set of sessions in the network is fixed. Then for this period we can define optimal rates as in section 2.

Thus, we allow the sessions to go through a period of instability in which some sessions can enter and exit, and then to stabilize to some fixed set for an extended period of time. We want the algorithm to stabilize to the optimal transmission rates for this set. Once the network has reached its current optimal state, we want it to remain there until new sessions enter or old sessions exit. If some sessions exit or enter, the optimal rates over the new set change as well. We want the algorithm to stabilize to the new optimal rates. If the set of network users changes much slower than the time required for the algorithm to converge, then the network will spend most of the time in a currently optimal state.

3.2 High-Level Description

The essential idea of the algorithm is to emulate the iterations of Procedure *Global Optimum* in a distributed asynchronous way.

To achieve this we let all packets carry an estimate of the bandwidth available for the session. This estimate will be referred to as the packet's 'stamped rate'. We stress here that in fact the algorithm does *not* require that every packet carries the stamped rate. We use this assumption only for simplicity. It will become clear that special control packets can be used to carry the stamped rate, or only a fraction of data packets can be used for this purpose.

We let each link maintain its current estimate of the fair share of its own capacity, referred to as the link's 'advertized rate'.

Originally the source sets the packet's stamped rate to some arbitrary initial value. As the packet travels through the network, its stamped rate is reset to the smallest of the packet's initial rate and the smallest of advertized rates of all links on the packet's round-trip route.

When a feedback packet returns to the source, the source adjusts its transmission rate according to the stamped rate of the feedback packet.

Each link maintains a list of its users. It adds a session to this list when the first packet of a new session is received. It deletes a session from the list when it determines that a session has exited. We do not address the issue of exactly how this determination is done. One could define a timeout value, or let the sessions send a special “last” packet. For the purposes of this work, however, we ignore any details or issues associated with any such choice, and simply assume that there is some way a switch can recognize the fact that the session is no longer active.

For each user the link stores its last seen stamped rate. It will be referred to as the ‘recorded rate’ of the session at the link.

The link sets a bit in the session’s entry if a packet of that session is received with stamped rate below or equal to the current advertized rate of the link. We say that a session with this bit set at some link is *marked at the link*.

The link then calculates its advertized rate as

$$\frac{C - \tilde{C}}{f + kb - \tilde{f} - k\tilde{b}} \tag{3}$$

where $\tilde{C}$ is capacity used by last seen stamped rates of the sessions marked at this link; $f, b, \tilde{f}, \tilde{b}$ are the number of total and marked forward and feedback sessions at the link respectively.

It is essential that the set of marked sessions at any time must satisfy the following conditions:

1. If any session is marked, its recorded rate is less than or equal to the advertized rate of the link.

2. Advertized rate is calculated according to (3).

The above conditions will be referred to as M-consistency, for “marking” consistency.

If at any time a session violates M-consistency, it must be immediately unmarked and the advertized rate must be recalculated. It turns out that M-consistency is central to ensure convergence of the algorithm to optimal values from any initial conditions. We will discuss M-consistency in more detail later in this work.

Finally note that is possible that if the source's idea of a session's rate is below the advertized rates of all sessions in the sessions route, and the session's demand is not satisfied, "obeying" the stamped rate received in the feedback packet would cause the session to operate below its optimal rate. To avoid this condition, an extra bit in the packet header is used. The bit will be referred to as the "u-bit". A "greedy" session, (i.e. a session whose demand is infinite or unknown), set's the u-bit to 0 on all of its packets. A "conservative' session, whose demand is known and finite, sets the u-bit of all its outgoing packets to 1.If the packet's stamped rate is above or equal to the advertized rate of a link in the packet's route, the link sets the u-bit to 1. Hence, if a feedback packet returns with u-bit set to 0, it means that advertized rate of all links was higher than the stamped rate of the packet. In this case the source ignores the received stamped rate and resets its idea of allowed rate to its demand.

There is no synchronization between operation of different network components The next section contains a formal description of the algorithm.

3.3 Algorithm Description

This section describes the data structures and operation of network components. Where appropriate, 'pidgin C' code is used to describe component operation. The code is not intended to be efficient and is sometimes redundant for the sake of clarity of the underlying ideas.

3.3.1 Data Structures

Packet p :

u_p 'u-bit' used to indicate that the session's rate can be increased

ρ_p packet's stamped rate

t_p packet's type (forward or feedback)

Source s :

ρ_s stamped rate of the last feedback packet received

u_s bit indicating whether or not to set the u-bit of outgoing packets

d_s demand of the session

Destination d :

ρ_d stamped rate of the last forward packet received

u_d 'u-bit' of the last forward packet received

$count_d$ used for counting the number of unacknowledged forward packets

Link l :

C_l Capacity of the link

f_l Number of forward sessions known at the link

b_l Number of feedback sessions known at the link

$\mathcal{G}_l$ Set of sessions known at the link

For any session $i \in \mathcal{G}_l$:

a_i^l - bit used to mark the session at the link

δ_i^l - is equal to 1 if the session is forward and to k if it is feedback (Note that k is a universal constant across the network)

ξ_i^l - recorded rate of the session

μ_l - advertised rate of the link, calculated as

$$\mu_l = \begin{cases} C_l & \text{if } f_l + kb_l = 0 \\ C_l - \sum_{j \in \mathcal{G}_l} \xi_j^l \delta_j^l a_j^l + \max_{i \in \mathcal{G}_l} \xi_i^l & \text{if } f_l + kb_l = \sum_{j \in \mathcal{G}_l} \delta_j^l a_j^l \\ \frac{C_l - \sum_{j \in \mathcal{G}_l} \xi_j^l \delta_j^l a_j^l}{f_l + kb_l - \sum_{j \in \mathcal{G}_l} \delta_j^l a_j^l} & \text{otherwise} \end{cases} \tag{4}$$

3.3.2 Source Operation

source_initialize(source s) {

/* called at initialization time */

if (demand not known) set $d_s = \infty$, $u_s = 0$

$\rho_s = d_s$

$if \;\; (d_s < \infty)$

$u_s = 1;$

else

$u_s = 0;$

}

source_receive_packet(source s, packet p) {

/* called upon receipt of a feedback packet */

if $(\rho_p > d_s)$ { /* must be that demand has decreased */

$\rho_s = D_S;$

$u_s = 1;$

}

else if $(u_p == 0)$ {

```
                $\rho_s = D_S$;
                if ($D_S < \infty$) $u_s = 1$;
                else $u_s = 0$;
        }
        else {
                /* Packet passed at least one link whose advertized rate
                /* was equal to the packet's current stamped rate,
                /* so obey this rate
                $\rho_s = \rho_p$;
                $u_s = 0$;
        }
}

source_generate_packet(source s, packet p){

        create new packet p
        $\rho_p = \rho_s$
        $u_p = u_s$
        add p to the outgoing link's output queue
}
```

3.3.3 Destination Operation

```
destination_initialize(destination d){

        $count_d = 0$
        $\rho_d = 0$;    $u_d = 0$;
}
```

```
destination_receive_packet(destination d, packet p){

/* called upon receipt of a forward packet */

    count_d = count_d + 1
    /* setparameters for the feedback packet as seen in the
    /* last of the k packets to be acknowledged
    if (count_d == k) {
        ρ_d = ρ_p;    u_d = u_p;
        countd = 0;
        create new feedback packet p
        ρ_p = ρ_d;    u_p = u_d
        send p
    }
}
```

3.3.4 Switch Operation

As soon as a packet arrives to an input link of a switch, it is added to the end of the output queue of the appropriate outgoing link. If several packets are received simultaneously from different input links, they are processed in some random order.

3.3.5 Output Link Operation

```
link_initialize(link l) {
    f_l = 0;  b_l = 0;  G_l = ∅
```

```
        μ_l = C_l
}

link_action(link l, packet p){
/* called when packet p is at the head of the link's output queue */
        if any session exited
                call link_update_session_exit(link l, session s)
        if (i_p ∉ G_l) /* packet belongs to a new session */
                call link_update_new_session(link l, packet p)
        else /* packet belongs to a session already seen at the link */
                call link_update_known_session(link l, packet p)
        transmit packet p
}

link_update_session_exit(link l) {

        /* update the list of known sessions */
        if packet of forward session
                f_l = f_l − 1
        else /* feedback session */
                b_l = b_l − 1
        G_l = G_l \ {i}

        /* recalculate advertized rate μ_l with updated information */

        μ_l = calculate_adv_rate(l);
}
link_update_new_session(link l, packet p) {
```

/* update the list of known sessions */

$\mathcal{G}_l = \mathcal{G}_l \cup \{i_p\}$

$\delta_i^l = t_p + k(1 - t_p)$

$f_l = f_l + t_p$

$b_l = b_l + (1 - t_p)$

/* do not mark the new session */

$a_{i_p}^l = 0;$

/* note that we do not need to set the recorded rate of the

/* new session at this time since unmarked session's recorded rate

/* is not used in calculation of advertized rate

$\mu_l = calculate\_adv\_rate(l)$

$if\ (\rho_p \geq \mu_l)$ {

$\rho_p = \mu_l;$

$u_p = 1;$

}

/* record the new rate now */

$\xi_{i_p^l} = \rho_p;$

}

link_update_known_session(link l, packet p) {

$if\ (\rho_p \geq \mu_l)$ {

$\rho_p = \mu_l;$

$u_p = 1;$

}

if $(\rho_p \leq \mu_l)$

$a^l_{i_p} = 1;$

$\xi^l_{i_p} = \rho_p;$

$\mu_l = calculate\_adv\_rate(l);$

calculate_adv_rate(link l)

{

/* first calculate advertized rate with given set of marked sessions */

RATE_CALCULATION: {

if $(f_l + kb_l) == 0$

$\mu_l = C_l$

if $(f_l + kb_l == \sum_{j \in \mathcal{G}_l} \delta^l_j a^l_j)$

$\mu_l = C_l - \sum_{j \in \mathcal{G}_l} \xi^l_j \delta^l_j a^l_j + \max_{i \in \mathcal{G}_l} \xi^l_i$

else

$\mu_l = \frac{C_l - \sum_{j \in \mathcal{G}_l} \xi^l_j \delta^l_j a^l_j}{f_l + kb_l - \sum_{j \in \mathcal{G}_l} \delta^l_j a^l_j}$

}

unmark any session whose recorded rate is above the calculated advertized rate

repeat RATE_CALCULATION *once more and return*

}

4 Convergence Theorem

In section 3.2 we introduced the notion of M-consistent calculation of the advertized rate of the link. Essentially, M-consistency means that once the advertized rate is calculated with some set of marked sessions, no session remains marked with recorded rate exceeding the advertized rate. Function *calculate_adv_rate*() in the algorithm description provides a possible way to perform M-consistent calculation. Note that the result of the M-consistent calculation is not only the advertized rate but also the set of "marked" sessions. Lemma 4.1 below proves that the result of this function is in fact M-consistent. Convergence theorem given later in this section proves that given *any* M-consistent advertized rate calculation the algorithm described in the previous section will converge to the optimal rate vector if started from arbitrary initial conditions. Thus, function *calculate_adv_rate*() can be treated as a "black box" and can be replaced by any other function providing M-consistent result. Thus in essence, Convergence Theorem 4.1 proves convergence of a family of algorithms with M-consistent link control calculation. We will return to the issue of M-consistency in section 7, where will will also give another example of M-consistent calculation.

Lemma 4.1 *After any link state update the advertized rate of the link and the marking of the sessions known at that link are M-consistent.*

Proof of Lemma 4.1. Consider any link update. Let $\mathcal{Y}$ be the set of sessions marked at the beginning of this update. Let μ^1 be the result of the first advertized rate calculation in function *calculate_adv_rate*(). Let $\mathcal{Z}$ denote the set of sessions which happen to be marked with stamped rates greater that μ_1. By operation of function *calculate_adv_rate*() all sessions in $\mathcal{Z}$ will be unmarked. Then, if not all sessions are marked, the final advertized rate μ returned by function *calculate_adv_rate*() is calculated as

$$\mu = \frac{C - \sum_{i \in \mathcal{Y} \setminus \mathcal{Z}} \xi_i \delta_i}{f + kb - \sum_{i \in \mathcal{Y} \setminus \mathcal{Z}} \delta_i} = \frac{C - \sum_{i \in \mathcal{Y}} \xi_i \delta_i + \sum_{i \in \mathcal{Z}} \xi_i \delta_i}{f + kb - \sum_{i \in \mathcal{Y}} \delta_i + \sum_{i \in \mathcal{Z}} \delta_i} \geq$$
$$\geq \frac{C - \sum_{i \in \mathcal{Y}} \xi_i \delta_i + \mu^1 \sum_{i \in \mathcal{Z}} \delta_i}{f + kb - \sum_{i \in \mathcal{Y}} \delta_i + \sum_{i \in \mathcal{Z}} \delta_i} = \mu^1$$

The last equality can be easily checked. Since all sessions which remain marked after the second advertized rate calculation in *calculate_adv_rate*() have recorded rates below or equal to μ^1, the statement of the lemma follows. If all sessions are marked, the statement of the lemma trivially holds, since by (4) advertized rate is greater or equal to the maximum recorded rate of its sessions.

Theorem 4.1 *Given arbitrary initial conditions on the states of all links in the network, states of all sources, destinations and arbitrary number of packets in transit with arbitrary control information written on them, the algorithm given in section 3 converges to the optimal rates as long as the set of sessions, their demands and routes eventually stabilize.*

Note that essentially any change in route or demand of the session is equivalent to an old session exiting and a new session entering. Thus, without loss of generality, the proof will be given under the assumption that demands and routes are fixed, but sessions are allowed to enter or exit, as long as eventually the set of sessions stabilizes. We give the proof for the case of infinite user demand. This does not cause any loss of generality, since as it has been already mentioned, the case of finite demands is reduced to this case by adding artificial links with capacities equal to session demands at the entry to the network.

The proof of this theorem is based on the following 4 lemmae.

Lemma 4.2 *After the set of sessions stabilizes at some time t_0 and all these sessions have become known at all links in the network,*

$$\mu_l(t) \geq \frac{C_l}{f_l(t) + kb_l(t)}$$

for all links l for all times $t \geq t_0$. Here f_l and b_l are the number of forward and feedback sessions crossing link l respectively.

Proof of Lemma 4.2

In what follows the link index l and the time argument t is omitted.

Consider the time of any state update of link l after t_0. By Lemma 4.1 the result of any link update is M-consistent, so any marked session i has recorded rate $\xi_i \leq \mu$. Let $\mathcal{Y}$ denote the set of indices j s.t. $a_j = 1$, (i.e. the set of marked sessions). Then, for the case when not all sessions are marked, by M-consistency

$$\mu = \frac{C - \sum_{j\in\mathcal{Y}} \xi_j \delta_j}{f + kb - \sum_{j\in\mathcal{Y}} \delta_j} \geq \frac{C - \mu \sum_{j\in\mathcal{Y}} \delta_j}{f + kb - \sum_{j\in\mathcal{Y}} \delta_j}$$

Hence, $\mu \geq \frac{C}{f+kb}$.

If all sessions are marked, two cases are possible. If $max_{i\in\mathcal{G}}\xi_i \geq \frac{C}{f+kb}$, then by M-consistency $\mu \geq max_{i\in\mathcal{G}}\xi_i \geq \frac{C}{f+kb}$, where $\mathcal{G}$ is the set of all sessions crossing the link, and the statement of the lemma holds.

If all $\xi_i < \frac{C}{f+kb}$, then by (4) and by M-consistency $\mu = \max \xi_i + C - \sum_{i\in\mathcal{G}} \xi_i\delta_i \geq C - \mu \sum_{i\in\mathcal{G}} \delta_i = C - \mu(f + kb)$ and the statement of the lemma follows.

Lemma 4.3 *Let τ_i denote the optimal rate of sessions in $\mathcal{S}_i$, where $\mathcal{S}_i$ is the set of sessions whose optimal rates were assigned at iteration i of Procedure* Global Optimum*, and $\mathcal{L}_i$ - the set of bottleneck links of this iteration. Let t_0, f_l, b_l be as in Lemma 4.1. Then for any $t > t_0$ it must be that*

$$\mu_l(t) > \tau_1 \quad \forall\, l \in \mathcal{L} \setminus \mathcal{L}_1$$

$$\mu_l(t) \geq \tau_1 \quad \forall\, l \in \mathcal{L}_1$$

Proof of Lemma 4.3

By Lemma 4.2 $\mu_l \geq \frac{C_l}{f_l(t)+kb_l(t)} \geq \frac{C_l}{f_l+kb_l}$

By Theorem 2.1 $\tau_1 = \frac{C_l}{f_l+kb_l}$ if $l \in \mathcal{L}_1$ and

$\tau_1 < \frac{C_l}{f_l+kb_l}$ if $l \in \mathcal{L} \setminus \mathcal{L}_1$.

This should be obvious since τ_1 is the capacity per session of a first-level bottleneck link, which by definition must be smaller than $\frac{C_l}{f_l+kb_l}$ of any other link.

The statement of this lemma immediately follows.

The next Lemma states that there exists some time, after which all sessions in $\mathcal{S}_1$ will have reached their optimal rate τ_1 and will be marked with this optimal rate at all links on their routes.

Lemma 4.4 *Let* $\tau_i,\ \mathcal{S}_i,\ \mathcal{L}_i$ *be as in Lemma 4.3. Then* $\exists\, T_1 \geq 0 \quad s.t. \quad \forall\, t \geq T_1$

1. $\rho_{p_i} > \tau_1$ *for any packet* p *of session* $i \in \mathcal{S} \setminus \mathcal{S}_1$.

2. $\xi_i^l > \tau_1$ *for any session* $i \in \mathcal{S} \setminus \mathcal{S}_1$ *and link* l *in the route of* i *or its feedback.*

3. $\mu_l = \tau_1$ *for any session* $l \in \mathcal{L}_1$

4. $\rho_{s_j} = \tau_1$ *for the source* s *of any session* $j \in \mathcal{S}_1$

5. $\rho_{s_j} > \tau_1$ *for the source of any session* $j \in \mathcal{S} \setminus \mathcal{S}_1$

6. $\rho_{p_i} = \tau_1$ *for any packet* p *of session* $i \in \mathcal{S}_1$

7. $a_i^l = 1, \quad \xi_i^l = \tau_1$ *for all sessions* $i \in \mathcal{S}_1$ *and all links* l *in the route of* i *or its feedback.*

Argument t *is omitted here.*

The proof of this Lemma is given in the Appendix 3.

The result of this lemma will now be used as the base case for induction on the index i of $\mathcal{S}_i$. Note that this lemma states that not only the sessions in $\mathcal{S}_1$ have reached their optimal rates, but this rates will never change and the sessions will be marked at all links in their routes ever after (as long as the set of sessions remains the same).

The inductive step is given the by following Lemma:

Lemma 4.5 (Inductive Step). *Suppose for some* $1 \leq i < m$

$\exists\, t_i \geq 0 \quad s.t. \quad \forall\, t \geq t_i$

1. $\mu_l = \tau_j$ *for any link* $l \in \mathcal{L}_j, \;\; 1 \leq j \leq i$

2. $\rho_{s_j} = \tau_i$ *for the source* s *of any session* $j \in \mathcal{S}_j, 1 \leq j \leq i$

3. $\rho_{s_j} > \tau_j$ *for the source* s *of any session* $j \in \mathcal{S} \setminus (\mathcal{S}_1 \cup \ldots \cup \mathcal{S}_i)$

4. $\rho_{p_k} = \tau_j$ *for any packet* p *of session* $k \in \mathcal{S}_j, \;\; 1 \leq j \leq i$

5. $a^l_k = 1, \quad \xi^l_k = \tau_j$ *for all sessions* $k \in \mathcal{S}_j \;\; 1 \leq j \leq i$ *and all links* l *in the route of* k *or its feedback*

6. $\rho_{p_j} > \tau_i$ *for any packet* p *of session* $j \in \mathcal{S} \setminus (\mathcal{S}_1 \cup \ldots \cup \mathcal{S}_i)$.

7. $\xi^l_j > \tau_i$ *for any session* $j \in \mathcal{S} \setminus (\mathcal{S}_1 \cup \ldots \cup \mathcal{S}_i)$ *and link* l *in the route of* i *or its feedback.*

Then $\exists\, t_{i+1} \geq 0 \quad s.t. \quad \forall\, t \geq t_{i+1}$ *such that conditions 1-7 hold for* $i + 1$.

It is assumed that the set of sessions has stabilized by time t_i.

Proof of Lemma 4.5.

By inductive hypothesis all sessions in $\mathcal{S}_j, \;\; 1 \leq j \leq i$ have reached their optimal rates τ_j and these rates do not change as long as the set of sessions remains unchanged. Moreover, by inductive hypothesis any session in $\mathcal{S}_j, \;\; 1 \leq j \leq i$ is marked with its optimal rate τ_j at any link on its way for all times $t > t_m$. Therefore, capacity of any link l in the network available for all sessions in $\mathcal{S} \setminus \tilde{\mathcal{S}}$, where $\tilde{\mathcal{S}}_i = (\mathcal{S}_1 \cup \ldots \cup \mathcal{S}_i)$, is $\tilde{C}_l = C_l - \sum_{j \in \tilde{\mathcal{S}}_i \; crossing \; l} \delta_j \tau_j$ $\forall \; t > t_m$.

Consider a reduced network with links $\mathcal{L} \setminus \tilde{\mathcal{L}}_i$,where $\tilde{\mathcal{L}}_1 = (\mathcal{L}_1 \cup \ldots \cup \mathcal{L}_i)$, sessions $\mathcal{S} \setminus \tilde{\mathcal{S}}$, and link capacities $\tilde{C}_l = C_l - \sum_{j \in \tilde{\mathcal{S}}_i \; crossing \; l} \delta_j \tau_j$. Note that it is legitimate to consider the reduced network precisely because by inductive hypothesis all sessions in $\mathcal{S}_1 \cup, \ldots, \cup \mathcal{S}_i$ have stabilized at their optimal rates and are forever marked at all links with their optimal recorded rates.

Denote $\tilde{f}_l = f_l - \sum_{j=1}^{i} f_j^l$ and $\tilde{b}_l = b_l - \sum_{j=1}^{i} b_j^l$

Note that by Theorem 2.1 $\tau_{i+1} = \frac{\tilde{C}_l}{\tilde{f}_l + k\tilde{b}_l}$ for all $l \in \mathcal{L}_{i+1}$ and

$\tau_{i+1} < \frac{\tilde{C}_l}{\tilde{f}_l + k\tilde{b}_l}$ for all $l \in \mathcal{L} \setminus \tilde{\mathcal{L}}_{i+1}$

Now the argument of Lemma 4.3 can be repeated for the reduced network to show that $\mu_l \geq \tau_{i+1} \;\; \forall \; l \in \mathcal{L}_{i+1}$ and $\mu_l > \tau_{i+1} \;\; \forall \; l \in \mathcal{L} \setminus \tilde{\mathcal{L}}_{i+1}$

Then repeating the proof of Lemma 4.4 for the reduced network we show that all the statements of Lemma 4.5 hold.

The statement of Theorem 4.1 follows by induction.

It is important to emphasize the role of M-consistent link control calculation in the proof. The fact that after any link state update there were no marked sessions with recorded rates above the newly calculated link control value ensured that the link control was eventually increased to allow sessions with higher rates regain capacity unused by sessions with lower rates.

5 Transient Behavior

There are several important implications of the proof of Theorem 2.1.

Note that the algorithm ensures that the optimal rates of sessions ia assigned in stages analogous to the stages of Procedure Global Optimum described in section 2.2. Lemmae 4.4 and 4.5 provide the framework for obtaining a worst case bound on the number of round-trips required for convergence. In particular, these lemmae imply that provided no new changes occur in the network,

- eventually all sessions of $\mathcal{S}_1$ get their optimal rates
- once all sessions of $\mathcal{S}_1\cup,\ldots,\cup\mathcal{S}_i$ get their optimal rates set, these rates are not going to change
- once all sessions of $\mathcal{S}_1\cup,\ldots,\cup\mathcal{S}_i$ get their optimal rates set, sessions in $\mathcal{S}_{i+1}$ will eventually get their optimal rates.

It follows from the proof of Lemmae 4.4 and 4.5 that the time required for all sessions of $\mathcal{S}_{i+1}$ to obtain their optimal rates *after* all sessions in $\mathcal{S}_1\cup,\ldots,\cup\mathcal{S}_i$ obtain their optimal rates is at most the time needed by any session to complete 4 round-trips (See the comment at the end of the proof of Lemma 4.4 in Appendix 3). Suppose now that the round-trip delay for is bounded by some D. Then the following upper bound holds:

Proposition 5.1 *Given an upper bound D on round-trip delay and the number N of iterations of the global procedure, the upper bound on the algorithm convergence time from any initial conditions is given by $4ND$.*

Note that the number of iterations of the global procedure is exactly equal to the number of *different* rates in the optimal rate vector.

It can also be easily seen that if the network operates at the optimum and then a new session comes in or exits, it takes at most $4D(N-M)$ for the algorithm to converge to a new set of optimal rates. Here N is again the number of iterations of the global procedure for the new set, and M is the "seniority" of the session, i.e. the index of the iteration of the global procedure at which the optimal rate of this session is assigned. This is simply because by operation of the algorithm sessions of lower optimal rates will not be affected by the newly arrived or departed session.

A few words are due about this bound.

First this bound gives the theoretical worst-case guarantee. In practice, convergence time should be expected to be significantly better. In fact, in our simulations we were not able to produce convergence worse than $2N$ round-trips. However, this result is tight in the sense that it is possible to create a very unfortunate sequence of packet transmissions to attain this bound.

Second, the convergence time measured in round-trips does not give a good insight into the actual convergence time measured in real time units if the time of round-trip delay D is not satisfactory bounded. Given a feasible set of transmission rates, a network configuration, a particular underlying service discipline, and the source's traffic shaping mechanism, we could hope to be able to obtain such bound either from experiment or from theoretical analysis. References [25], [11] provide such upper bounds for particular service disciplines and source traffic shapes. Note also that for feasible constant-rate smooth flows of infinitely small packets D is simply the network propagation delay.

However, unless special measures are taken, the transient infeasibility of transmission rates can cause significant queue growth, and, as a result, can significantly increase the upper bound on D and slow convergence of the algorithm (as measured in real time rather than the number of round-trips). Thus it would be very important to ensure that the algorithm produces a feasible transmission vector as early as possible.

To get some intuition on this, consider a synchronized implementation of the algorithm in which all sessions start simultaneously and each link updates its state when it has received information from all of its users. Suppose that all sessions are known at all links. Then it can be easily seen that after a link updates all of its sessions (on their very first pass), and possibly resets the stamped rates of the packets, the sum of all stamped rates on output does not exceed the capacity of that particular link. Thus, in such synchronized version when all packets return to their sources, the resulting vector of stamped rates will be feasible. This feasibility will continue to hold if we look at the stamped rate as it is *received* in the incoming feedback packet rather than as it is set in the *outgoing* packets. (Note that the vector of stamped rates of the *outgoing* packets may not and will not be feasible if a source detects that it needs to increase its rate upon receiving a packet with u-bit set to zero. In this case the stamped rate of the outgoing packet will be set to potentially very large value of the session's demand).

Hence, in the synchronized implementation we could preserve the feasibility of the actual rate vector by simply setting the actual rate to the newly received stamped rate if the u-bit is set to 1 and preserving the old actual rate otherwise. If in addition we choose initial transmission rates conservatively to ensure initial feasibility, we would ensure feasibility of the algorithm in transience. Clearly, as soon as the stamped rates converge to their optimal values, this policy will reset the actual rates to the optimal values at most one round-trip later.

However, in an asynchronous implementation this policy will not necessarily ensure feasibility. This can be seen by observing that a faster session can increase its actual transmission rate before slower sessions realize that they need to decrease theirs. We have not rigorously proved, but we believe that the heuristic described below will resolve this problem. The key point is that the *actual* transmission rate does not have to be adjusted at the same time as the stamped rate, since the stamped rate calculation is completely decoupled from the underlying traffic. Suppose the value of D for the *uncongested* network is available. Then,

- if the actual transmission rate needs to be *decreased* according to the "synchronized" policy described above, decrease it immediately

- if the actual transmission rate needs to be *increased*, according to the "syncronized" policy wait for $2D$ before increasing it.

The rationale for this policy is that decreasing the rate cannot possibly violate feasibility, while increasing it can, if the other sessions are not yet aware of the increase. Since the *stamped* as opposed to the *actual* rate is in fact increased according to the original algorithm, on the next round-trip the new rate increase will be known at all links, so the other sessions will be notified about this change no later than on their next round-trip after that.

The results of our simulation in fact suggest that such policy may be too conservative. In the experiments described in the next section we were unable to produce infeasibility for more than one round-trip after a new session entered the network even without implementing the above policy. More investigation on this issue is called for.

It should be also noted that one round-trip after all sessions become known at all links, any session's stamped rate is at least as large as the minimum of its demand and the equal share of the capacity of the bottleneck session for this link. Note that while this may be less than the optimal rate of this session, this bound ensures that all sessions are guaranteed reasonable throughput even before the optimal rates are obtained.

The above considerations in combination with the simulation results presented in the next section lead us to believe that the algorithm is "well-behaved" in transience.

Finally it is instructive to compare the behavior of this algorithm to other algorithms for finding maxmin optimal rates. In particular we will look at Selective Binary Feedback Scheme (SBF) presented in [26] and the scheme presented in [23]. While extensive comparison is beyond the scope of this work, we believe that the following simple example gives a good insight on the comparative behavior of these algorithms.

Consider a network consisting of one bottleneck link of capacity C shared by two sessions with the desired demand D. We will now investigate the behavior of the three algorithms in such system. Since SBF scheme is designed to operate in the window environment, we "translate" the window size to the effective transmission rate by dividing the window size by the round-trip delay.

For simplicity, we assume synchronized operation. We look at the case when initial demands $D > C$. Suppose $C = 20$ and initial rates are $R_1 = 100$, $R_2 = 50$. The optimal rate for both session is 10

Selective Binary Feedback Scheme This scheme essentially computes the same value for link controls (called "fair allocation" in [26]) as the "advertized rate" in this thesis. However, instead of delivering the minimal value of all link controls to the source, SFB sets a bit in the packet header if the current demand of the session on the link is above the link control value. If this bit is set in the feedback signal, the source adjusts its effective rate to $R^{new} = cR^{old}$, where $0 < c < 1$. If the bit is 0, the source sets the effective rate to $R^{new} = R^{old} + b$, where $b \geq 0$. Note that the optimal rate in our example is $C/2$ for both sessions.

Clearly, if $D - C/2$ is large, it may take quite a few iterations to just reduce the rate below $C/2$. Then the algorithm will additively increase the effective rate until it gets above the current link control, etc. Note that it is impossible to ensure feasibility, since the algorithm will invariably have to eventually raise the rate above its feasible value to "feel" its way to the optimum. The following sequence gives the effective rates for the two sessions of our example for the case $c = b = 0.5$:

R_1 : 100, 50, 25, 12.5, 13.0, 6.5, 7.0, 7.5, 8.0, 8.5, 9.0, 10.0 10.5, 11.0 ...

R_2 : 50, 25, 12.5, 6.25 6.75, 7.25, 7.75, 8.25, 8.75, 9.25, 9.75, 10.25, 5.125, ...

Mosley [23] calculates link controls as $\max_i r_i + \frac{C - \sum_i r_i}{n}$, where r_i is the "observed rate" of session i across the link and n is the number of sessions crossing the link. As in our scheme, the minimal value of the link controls is conveyed to the source, which then sets its transmission rate to this value.

The following sequence gives the rates obtained at several iterations for the sessions in our example:

R_1 : 100, 25, 10, 10, 10 ...

R_2 : 50, 25, 10, 10, 10 ...

Our Scheme As we have seen earlier, our scheme gets the optimal values on the very first round-trip.

R_1 : 10, 10, 10, 10, 10 ...

R_2 : 10, 10, 10, 10, 10 ...

As indicated by this trivial example, it should be expected that the algorithm described in this thesis converges and reaches feasibility faster. This expectation should be intuitively clear for the following reasons. SBF, while taking information about individual sessions into account, does not have a way to efficiently use this information, since the source is only notified whether its rate must be increased or decreased, but is not told "how far to go". Mosley's scheme does inform the source about the rate it should transmit at, but the information needed to compute this rate is based on the aggregate rate of all sessions and the maximum rate of all sessions. In contrast, our algorithm takes full information about the individual session rates into account when calculating the rate estimate, and informs the source about this rate.

6 Simulation Results

Experiments 1-5 show that the algorithm described in this thesis allows the sessions to adapt quickly to the changes in network load due to sessions entering or exiting the network or variation of active session demands.

The MIT Advanced Network Architecture group's network simulator was used in in this work.

A number of simulation experiments have been performed on different configurations. The results of these experiments are provided in the following subsections. We demonstrate that the algorithm allows the sessions to adapt quickly to any changes in the network load and converge to the current optimal rate.

All experiments were performed with deterministic packet transmission rates on an underutilized network. The main purpose of these experiments was to investigate the number of round-trips required for convergence in the presense of dynamic network changes. Even in an underutilized network the actual length of round-trips of different sessions and even different packets of any one session can be different due to different path lengths, event scheduling, etc. Thus, the number of round-trips seems to give more insight into the algorithm behavior than the actual time. We observed that in all our experiments the time of convergence was well below the worst-case theoretical bound obtained in section 5.

In addition we investigated the feasibility of the transient solutions. We have not used the heuristic described in section 5. Even without it, the rate vector of stamped rates became feasible after one round-trip following a change in the network load.

Finally, we investigated the behavior of the algorithm with link control calculation used by the Selective Binary Feedback Scheme. Since no substantial difference was observed, the results of these simulations are not shown.

Note that the stamped rate plotted in all figures of this thesis reflect the stamped rate as seen at the input to the sources. That is, if we were to follow the "synchronized" policy described in 5 this stamped rate would in fact reflect the actual transmission rate.

6.1 Experiment 1

Network configuration of this experiments is shown in Figure 1. The purpose of this experiment is to observe how the sessions adjust to the changing network load when sessions enter or exit the network.

In this configuration 5 sessions share one bottleneck link. At time 0 all sessions begin transmitting their data. Their initial demands exceed the fair share of the bottleneck link. Eventually session 1 exits, while sessions 2-4 continue to transmit. Then, some time later sessions 2 and 3 exit. Still some time later session 3 reenters, but this time its demand is below its fair share of the bottleneck link. Finally, session 2 reenters with its initial demand.

Tables 1 and 2 contain the session demands and session optimal rates at different times of this experiment.

Figures 2 - 6 show the behavior of the stamped rates of Sessions 1-5 as a function of time. Boxes on the plots correspond to the round-trip epochs. Note that the sessions quickly adjust to the changing network load when sessions enter or exit. Table 3 gives the number of round-trips each session took to converge to the new optimal rates after each change in the network load. Note that all these numbers are within the theoretical guarantee of 4. After the very first round-trip feasibility is preserved throughout the experiment.

6.2 Experiment 2

The setting of this experiment is similar to Experiment 1 (see Figure 7).

There are 15 sessions sharing one bottleneck link. All sessions start transmission at time 0. Given infinite demand, the fair share of the bottleneck link is 10000. We consider the case when the demand of sessions 1-7 is strictly below this value, while demand of sessions 9-15 is strictly above 10000. Demand of session 8 is exactly 10000.

The demands are chosen so that the total demand is exactly equal to the bottleneck link capacity. As a result, the demand vector is also the optimal rates vector (see Table 4).

In this scenario, it should be expected that sessions 1-8 should be able to transmit at their demand immediately, while sessions 9-17 should regain the capacity not used by the less demanding sessions and should stabilize to their demanded rate. Moreover, any session of 9-17 should wait for all sessions with lower optimal rates to stabilize before it can reach its own optimal rate.

Figures 8-10 show that the sessions behave exactly as expected. Note that this time the x-axis is the number of round-trips rather than the actual time. Since different round-trips may take different time, the plots of stamped rate versus time will not necessarily be as well alligned.

We see that the worst observed convergence time is 10 round-trips (session 15) is drastically smaller than the theoretical guarantee of 60.

6.3 Experiment 3

In this experiment we investigate the behavior of the network with 2 levels of bottleneck links.

Configuration used in this experiment is shown in Figure 11. Sessions 1-4 with large initial demands start transmission at time 0. Sessions 3 and 4 share the first-level bottleneck link 2. Their optimal rates are 5000. Sessions 1 and 2 share the second-level bottleneck links 1 and 2. Their optimal rates are 1000.

Figures 12 - 15 show the behavior of sessions 1-4. As expected, the sessions sharing the second-level bottleneck links take longer to converge to their optimal rates as the sessions sharing the worst bottleneck.

Note the typical "climbing upstairs" behavior of the sessions sharing the second-level bottleneck (sessions 1 and 2). They operate at a lower rate until they realize that the first-level sessions operate at low rates and regain the free capacity.

We still observe that the worst obsered convergence time of 6 (session 2) is much smaller than the theoretical guarantee of 12.

6.4 Experiment 4

This experiment is intended to investigate the behavior of the network with different number of hops in the routes of different sessions and different optimal rates. Configuration of this experiment is given in Figure 16. It is clear that the actual number of round-trips required for all sessions to converge to their optimal rates depends on the specific timing of the events in the system. In particular, clearly session 4 should receive its optimal rate after the very first round-trip. In contrast, session 1 may have to wait till all other sessions stabilize to their optimal rates before getting its optimal rate assigned. Since session 1 also has the shortest route, it should be expected that it will take session 1 significantly more round-trips to converge than session 1. Similarly, session 2 may have to wait for session 1, and session 3 may have to wait for sessions 1 and 2.

Optimal rates of the sessions are given in table 5.

Figure 17 shows that the simulated behavior of session 4 is exactly as expected, that is, it converges to its optimal rate immediately after the first round-trip. Session 3 also converges after the first round-trip, which is faster than the worst-case behavior described above. The behavior of sessions 3 and 4 corresponds to our expectations. Session 2 converges on the third round-trip, and it takes session 3 6 round-trips to converge. This is much better than the theoretical guarantee of 16. Again, we see that feasibility is preserved throughout the experiment.

6.5 Experiment 5

Configuration of this experiment is shown in Figure 18. This experiment is intended to examine a more sophisticated case in which different sessions have different number of links in their routes. In addition, there are 3 levels of bottleneck links here. We also investigate the response of the algorithm to dynamic changes.

All 5 sessions enter at time 0. Then session 3 exits at time 15. At time 48 sessions 1 and 2 exit, and, finally, at time 67 session 3 reenters. Demands of all sessions are large.

Optimal rates of sessions at different times are given in Table 6. Figures 19-23 show that all sessions quickly determine their optimal rates after load changes. Table 7 gives the number of round-trips required by each session to stabilize to the new optimal rate.

We observe again that the maximum observed number 7 of round-trips required for the algorithm to stabilize to its optimal rate after a change in the network load is still significantly below the theoretical upper bound of 12.

Note also that the feasibility is restored after the very first round-trip following a change in the set of network users.

7 Discussion

7.1 Remarks on M-consistency

As mentioned earlier in this work, M-consistent rate calculation was essential in ensuring the algorithm convergence. We proved earlier that our advertized rate calculation policy was M-consistent. We emphasize that checking for violation of M-consistency and enforcing it at all times ensures full recovery from any previous data corruption.

It is possible to construct other M-consistent policies. In this section we will show that the scheme for link control calculation in the Selective Binary Feedback Scheme presented in [26] is also M-consistent. The SBF performs calculation on each update of its link control A_{fair} , called "fair allocation" as follows. Let J denote the set of a "allocated" sessions, and A_i denote the rate of session i as observed by the link.

- Initially it sets $A_{fair} = \frac{C}{n}$, $J = \emptyset$
- Otherwise, if $A_i < A_{fair}$, allocate i, i.e. set $J = J + \{i\}$ and calculate

$$A_{fair} = \frac{C - \sum_J A_i}{n - number\ of\ unallocated\ sessions}$$

Repeat until there are no sessions to allocate.

Note that the "allocated" sessions in SBF are analogous to our "marked" sessions.

It can be easily seen that SBF is also M-consistent. Thus, we can use SBF control calculation in our algorithm and get the same convergence results. In fact, we have done that in simulation and no significant changes in the algorithm behavior were observed.

Note significant similarity in the actions taken between the two schemes. The main difference, however, is that SBF does not keep track of previous allocations (or markings in our terminology) and recalculates them anew on each calculation. In contrast, our scheme remembers the past allocations and uses the past information for the new calculation.

Note also that the two schemes may produce different values of link controls and the sets of marked sessions. To see this, consider 3 sessions with "observed", or "recorded", rates equal to 1, 1 and 5 on a link of capacity 6. It is entirely possible that in our algorithm one of the sessions with ("recorded") rate 1 is marked, while the other 2 are not. Our algorithm will then calculate the control value ("advertised rate") as $\frac{6-1}{2} = 2.5$. The calculation of SBF for theses rates will calculate its control value ("fair allocation") as 4 and would mark ("allocate") both sessions with observed rates 1.

Thus, a family of link control calculation policies has been identified. If plugged into the algorithm suggested in this thesis, any policy of this family will ensure convergence and the transient properties described in the previous sections.

7.2 Comments on the usage of the "Stamped Rate" Field in the Packet Header

The algorithm makes substantial use of the field in the packet header, which we call the "stamped rate" of the packet. Note that if the actual transmission rates could be accurately measured on input to a switch and if newly calculated rates could be accurately enforced on output, the packets would no longer need to carry this field, thus reducing the overhead. However, despite the obvious overhead, the use of this field may be well justified for the following reasons.

- It is very difficult to measure and enforce rates with sufficient accuracy. Such measurements and enforcement strongly depend on the underlying service discipline and the shape of underlying traffic. The effects of measurement and enforcement errors are unclear and need more investigation.

- Rate measurements constitute significant operational overhead in every network node. It can be argued that measurments might be necessary anyway for rate enforcement for some service disciplines. While this is certainly true for some service disciplines, simple FIFO does not need to perform any measurements. Rate measurments may also be needed for policing misbehaved users. However, such policing can be done at the entry to the network only, thus relieving the rest of the network from the necessity of measuring the rates.

- Having the stamped rate in the packets completely decouples the processes of rate calculation and rate enforcement. In particular,
  - the algorithm can be used with any underlying service discipline (as long as the packets of each sessions are served FIFO), e.g. FIFO, FIFO+, Round-Robin, Stop-and-Go, etc, with any flow control mechanism and any traffic shaper;
  - the algorithm is robust in the presence of heavy packet loss in the sense that it converges even if only some packets continue to get through;
  - the above quality allows to the algorithm on top of any other algorithm in some specialized control packets if needed

These considerations in combination with the convergence and transient properties described earlier in this thesis suggest that the overhead of having an extra field in the packet header may be well justified by the benefits it produces.

7.3 Policing Misbehaved Sessions

The algorithm described so far suffers from the same problem as most of the other schemes using FIFO queues. That is, if a session violates the rules imposed by the algorithm due to malfunction, maliciousness or a different protocol used, it can flood the network with its data at the expense of well-behaved users.

Note that when a feedback packet returns to the source, its stamped rate is set to the current estimate of the session's allowed rate. If the packet's 'u-bit' is set, then the source is supposed to adjust its transmission rate to this stamped rate. If the 'u-bit' is not set, then the source is expected to send unmarked packets with some a-priori known initial rate.

Thus, the first switch on the session's way, which is also the last switch on the feedback's way can determine session's allowed rate by the information it the last feedback packet of that session seen. It can then measure the actual transmission rate of the session and drop packets of this session if this rate is exceeded.

Note that this is only needed at the entry to the network, so the switches in the middle of the network connected to other switches only do not need to worry about misbehaved users at all.

7.4 Network with Full-Duplex Links

The assumption of this model was that each physical connection is modeled as two half-duplex links of identical capacity.

It should be noted that the algorithm can be used for the case of the full-duplex links with the following modification. In essence, the link should no longer distinguish between the forward and the feedback sessions crossing it. Instead, it should keep track of the total number of sessions crossing it and should calculate its "advertized rate" as if all sessions were one-way only, but the total capacity of the link were $\frac{C}{(1+k)}$ rather than C. In addition, the control fields of the feedback packets should not be changed by the switches. It can be shown that with these modifications the algorithm will produce maxmin optimal rate allocation in a network with full-duplex links as well.

8 Summary and Areas for Future Research

We have described an algorithm for distributed asynchronous computation of maxmin optimal session transmission rates. We have shown that the algorithm converges to the optimal rates faster than other algorithms achieving a similar objective and that it is well-behaved in transience. The algorithm is self-stabilizing in the presence of dynamic network changes and heavy packet loss. Unlike previous work, this algorithm takes bandwidth consumed by feedback traffic into consideration.

We have argued that though a field in the packet header utilized by the algorithm constitutes undesirable overhead, it also provides significant flexibility of potential application.

Note that the algorithm can be easily extended to the case when instead of end-to-end feedback the switch sends a feedback packet to the previous switch on its way to inform it about its control value if it is smaller. Such hop-by-hop feedback will propagate the minimum link control value (advertized rate) to the source faster than the end-to-end scheme. Investigating the behavior of this algorithm with hop-by-hop feedback might be a topic for future research.

Another possibility to improve convergence time of the algorithm might be to restrict allowed transmission rates to some discrete values. While this would certainly decrease the number of potential different values of the optimal vector and thus improve convergence time, the effects of such discretization are not very clear and need further research.

We also believe that the transient behavior of the algorithm needs to be further examined in more extensive simulation and perhaps in some real-life environment. The results of such investigation may be crucial to determine practical applicability of the algorithm.

9 Appendix 1

Summary of Notation

$\mathcal{R}$ Set of switches in the network.

$\mathcal{L}$ Set of (half-duplex) links in the network (assumed even).

$\mathcal{S}(t)$ Set of sessions in the network at time t

R Number of Switches in the network.

L Number of (half-duplex) links in the network.

$S(t)$ Number of sessions in the network at time t.

$0 < k \leq 1$ Ratio of session reception rate to acknowledgement transmission rate (assumed constant throughout the network).

$\mathcal{P}(l)$ Mapping of set $\{1, \ldots, L\}$ into itself such that

- $\mathcal{P}(i) = j \Leftrightarrow \mathcal{P}(j) = i$
- $\mathcal{P}(i) \neq i$

Mapping $\mathcal{P}$ breaks the set of all indices of the half-duplex links in the network into $\frac{L}{2}$ pairs, such that each pair contains the indices of two half-duplex links corresponding to one full-duplex link

$u_{i,j}$ Set to 1 if session i crosses link j on its forward path, set to 0 otherwise.

$w_{i,j}$ Set to 1 if session i crosses link j on its feedback path, set to 0 otherwise.

$\delta_i^j = u_{i,j} + kw_{i,j}$

$\mathcal{F}_j(t)$ Set of sessions crossing link j on the forward path at time t.

$\mathcal{B}_j(t)$ Set of sessions crossing link j on the feedback path at time t.

$\mathcal{G}_j(t) = \mathcal{F}_j(t) \cup \mathcal{B}_j(t)$ Set of all sessions crossing link j at time t

$f_j(t)$ Number of forward sessions crossing link j at time t.

$b_j(t)$ Number of forward sessions crossing link j at time t.

C_i Capacity of link i.

h_i Number of links (hops) in the route of session i.

n_j^i Index of the j-th link in the route of session i.

$\mathcal{A}_i$ Set of half-duplex links incoming to switch i.

$\eta_i(t)$ Actual transmission rate of session i at time t.

$\lambda_{i,j}(t)$ Actual throughput of session i across link j at time t.

$\psi_{i,j}(t)$ Actual throughput of acknowledgement of session i across link j at time t.

$\nu_j(t)$ Actual service rate of link j at time t

$\phi_i^j(t)$ The queue in the output buffer of link i due to session j only at time t.

$\mathcal{L}_i$ Set of bottleneck links of iteration i of Procedure Global Optimum.

$\mathcal{S}_i$ Set of sessions crossing all links in $\mathcal{L}_i$ whose optimal rate is assigned iteration i of Procedure Global Optimum.

τ_i Optimal rate of sessions in $\mathcal{S}_i$

f_i^l Number of sessions of $\mathcal{S}_i$ crossing link l on the forward path

b_i^l Number of sessions of $\mathcal{S}_i$ crossing link l on the feedback path

10 Appendix 2

Proof of Theorem 2.1.

At every iteration of the procedure at least one new link is chosen as the bottleneck link of the reduced network. All sessions crossing this link are added to $\tilde{\mathcal{S}}_i$. Note that none of these sessions have been added to $\tilde{\mathcal{S}}_i$ at any iteration $j < i$. Thus, at the end of each iteration the number of sessions in $\tilde{\mathcal{S}}_i$ increases at least by one. The algorithm terminates when all sessions in $\mathcal{S}$ are in $\tilde{\mathcal{S}}_i$, so at most S iterations are needed to terminate. This proves Statement 1 of the theorem.

We now prove statement 2. First note that by the way τ_i is defined $\tau_1 < \tau_2 < \ldots < \tau_m$.

Let n_i be the number of sessions in $\mathcal{S}_i$. Renumber the sessions in $\mathcal{S}$ so that the first n_1 indices correspond to sessions in $\mathcal{S}_1$, the next n_2 indices correspond to sessions in $\mathcal{S}_2$, etc. Then the rate allocation obtained by Procedure *Global Optimum* can be written as $\beta = (\beta_1, \ldots, \beta_S) = (\underbrace{\tau_1, \ldots, \tau_1}_{n_1}, \ldots \underbrace{\tau_m, \ldots, \tau_m}_{n_m})$ Let $\alpha = (\alpha_1, \ldots, \alpha_S)$ be any other feasible rate allocation. We show that β is lexicografically greater than than α. Let a be the a permutation of α such that $a_1 \leq a_2 \leq \ldots \leq a_S$. Suppose $a_1 > \tau_1$. Then consider any link $l \in \mathcal{L}_1$. Note that by operation of Procedure *Global Optimum* $\tau_1 = \frac{C_l}{f_l + kb_l}$.

The total throughput across link l is

$$\sum_{j \in \mathcal{G}_l} a_j \delta_j^l \geq a_1(f_l + kb_l) > \eta_1(f_l + kb_l) = C_l$$

where $\delta_i^j = u_{i,j} + kw_{i,j}$.

Thus, feasibility condition is violated, so it must be that $a_1 \leq \tau_1$. If $a_1 < \tau_1$, then β is lexicografically greater than than α. Suppose $a_1 = \tau_1$.

We now show that if $a_j = \beta_j \;\; \forall\, 1 \leq j < i$ then $a_i \leq \beta_i$. Let $1 \leq k \leq m$ be the index s.t. $\beta_i = \tau_k$. Consider any link $l \in \mathcal{L}_k$. Note that by operation of the Procedure only links from $\tilde{\mathcal{L}}_k$ cross l. Also, for this l

$$\tau_k = \frac{C_l - \sum_{j=1}^{k-1} \tau_j (f_l^j + k b_l^j)}{f_l + k b_l - \sum_{j=1}^{k-1} (f_l^j + k b_l^j)} \tag{5}$$

Suppose $a_i > \tau_k$. Denote by Q the set of sessions j crossing link l s.t. $\beta_j = \tau_k$. Note that Q is just the set of sessions of $\mathcal{L}_k$ crossing l. Then the total throughput across link l is

$$\begin{aligned} \sum_{j \in \mathcal{G}_l} a_j \delta_j^l = \sum_{j=1}^{k-1} \tau_j (f_l^j + k b_l^j) + \sum_{j \in Q} a_j (f_l^j + k b_l^j) \geq \\ \sum_{j=1}^{k-1} \tau_j (f_l^j + k b_l^j) + a_i \sum_{j \in Q} (f_l^j + k b_l^j) > \\ \sum_{j=1}^{k-1} \tau_j (f_l^j + k b_l^j) + \tau_k \sum_{j \in Q} ((f_l^j + k b_l^j) = \\ = \sum_{j=1}^{k-1} \tau_j (f_l^j + k b_l^j) + \tau_k (f_l + k b_l - \sum_{j=1}^{k-1} (f_l^j + k b_l^j)) = C_l \end{aligned}$$

The last equality follows from (5). Thus, feasibility condition is violated again and it has been proved that it must be that $a_i \leq \beta_i$. If the inequality is strict, then β is lexicografically greater that α. Thus it follows by induction that β is lexicografically greater than any other feasible rate allocation vector, so τ are indeed optimal rates.

This concludes the proof of Statement 2 of the theorem. The fact of Statement 3 of the theorem that $\tau_1 <, \ldots < \tau_m$ simply follows from the definition of the bottleneck link of iteration i.

The claim of Statements 4, 5 and 6 of the theorem immediately follow from the operation of Procedure Global Optimum.

The proof of Theorem 2.1 is now complete.

11 Appendix 3

Proof of Lemma 4.4 By Lemma 4.3 $\exists\, t_1 \geq 0, \quad s.t. \quad \forall\, t \geq t_1$

$$\mu_l \geq \tau_1 \quad \forall\, l \in \mathcal{L}_1 \tag{6}$$

$$\mu_l > \tau_1 \quad \forall\, l \in \mathcal{L} \setminus \mathcal{L}_1 \tag{7}$$

It will now be shown that eventually all packets of any session in $\mathcal{S}_1$ will have stamped rate at least as great as τ_1, any packet of any session not in $\mathcal{S}_1$ will have stamped rate strictly above τ_1, all sessions in $\mathcal{S}_1$ have recorded rates at least as great as τ_1 at all links in their route, and all sessions not in $\mathcal{S}_1$ have recorded rates strictly greater than τ_1 at all links in their route. That is, $\exists \;\; t_2$ such that the following inequalities hold $\forall \;\; t \geq t_2$:

$$\begin{aligned} \xi_j^l &\geq \tau_1 \;\; \forall \;\; l \;\; in \;\; route \;\; of \;\; j \in \mathcal{S}_1 \\ \rho_{p_j} &\geq \tau_1 \;\; for \;\; any \;\; packet \;\; p \;\; of \;\; j \in \mathcal{S}_1 \\ \xi_j^l &> \tau_1 \;\; \forall \;\; l \;\; in \;\; route \;\; of \;\; j \notin \mathcal{S}_1 \\ \rho_{p_j} &> \tau_1 \;\; for \;\; any \;\; packet \;\; p \;\; of \;\; j \notin \mathcal{S}_1 \end{aligned} \tag{8}$$

Note that the last two inequalities of (8) are exactly the statements 1 and 2 of this lemma.

To see that (8) holds, consider some $j \in \mathcal{S}_1$. Consider the first packet of j to leave the source after time t_1. When feedback to that packet returns to the source at some time $t_j^1 \geq t_1$, two cases are possible:

Case 1.

The ‘u-bit’ of the packet is set. Then its stamped rate $\rho_p(t_j^1)$ must be set to the minimum of the advertised rates of the links seen on the packet's route. By (6) and (7) it then must be that $\rho_p(t_j^1) \geq \tau_1$. Hence at the source $\rho_s(t_j^1) \geq \tau_1$

Case 2.

The u-bit is not set. Then by operation of the algorithm $\rho_s(t_j^1) = \infty \;\; > \tau_1$.

Note that since individual sessions are served in FIFO order, any feedback packet arriving at the source after time t_j^1 corresponds to a forward packet which left the source after time t_1. Applying the above argument to any such packet we can see that

$$\forall\, t > t_j^1 \quad \rho_{s_j}(t) \geq \tau_1 \tag{9}$$

Note that condition (9) holds for any session in $\mathcal{S}_1$ at all times after it it has completed its first round-trip.

Consider any packet of j which left the source after t_j^1 Let t_j^2 be the time when feedback to this packet returns to the source. Any packet of j which is in the network at any time after t_j^2 left the source after time t_j^1. By operation of the algorithm, at any link l in its route its stamped rate is set to $\rho^{new} = \min(\mu_l, \rho^{old})$. By (9) its initial rate is set to $\rho^{init} = \rho_{s_j} \geq \tau_1$, so by (6) and (7) it must be that for any packet p of session j the second inequality of 8 holds for all $t > t_j^2$.

By operation of the algorithm ξ_j^l is set to the last seen ρ_p of session j. Let t_j^3 denote the time when first feedback packet to any forward packet of j sent after t_j^2 returned to source. Then the recorded rate of session j at any link on its way was set after time t_j^2. It has already been shown that all packets of j have stamped rates above or equal τ_1 for any time $t \geq t_j^2$. Hence, by operation of the algorithm and from 6 and 7 the first 8 holds for all $t \geq t_j^3$.

Thus, taking $t_2 = max_{j \in \mathcal{S}_1} t_j^3$ we have shown that the first two inequalities of (8) hold for all $t > t_2$.

By Theorem 2.1 any session $j \in \mathcal{S} \setminus \mathcal{S}_1$ does not go through any link $l \in \mathcal{L}_1$. Repeating the exact same argument as above for $j \in \mathcal{S} \setminus \mathcal{S}_1$ and taking (7) into account, it can be shown that the third and fourth inequalities of (8) hold as well.

Note that time t_2 is the time when all sessions in $\mathcal{S}_1$ have completed 2 round-trips.

To prove statement 3 of the lemma, consider any link $l \in \mathcal{L}_1$. By Theorem 2.1 only sessions from $\mathcal{S}_1$ cross any link of $\mathcal{L}_1$. Then, if not all sessions are marked,

$$\mu_l = \frac{C_l - \sum_{j \in \mathcal{S}_1} \xi_j^l a_j^l \delta_j}{f_l + kb_l - \sum_{j \in \mathcal{S}_1} a_j^l \delta_j}$$

By Theorem 2.1 $\tau_1 = \frac{C_l}{f_l + kb_l}$ for any $l \in \mathcal{L}_1$. Hence for any $l \in \mathcal{L}_1$

$$\mu_l = \frac{\tau_1(f_l + kb_l) - \sum_{j \in \mathcal{S}_1} \xi_j^l a_j^l \delta_j}{f_l + kb_l - \sum_{j \in \mathcal{S}_1} a_j^l \delta_j} \tag{10}$$

Note that (10) and (8) imply that $\forall\, t > t_1 \quad and \quad \forall\, l \in \mathcal{L}_1$

$$\mu_l = \frac{\tau_1(f_l + kb_l) - \sum_{j \in \mathcal{S}_1} \xi_j^l a_j^l \delta_j}{f_l + kb_l - \sum_{j \in \mathcal{S}_1} a_j^l \delta_j} \leq \frac{\tau_1(f_l + kb_l) - \tau_1 \sum_{j \in \mathcal{S}_1} \delta_j}{f_l + kb_l - \sum_{j \in \mathcal{S}_1} a_j^l \delta_j} = \tau_1$$

By (6) $\mu_l \geq \tau_1$ for all $l \in \mathcal{L}_1$, so statement 3 of the lemma follows for any $t \geq t_2$.

The case when all sessions are marked is quite similar and is omitted here.

To see that statement 4 holds, note that by Theorem (2.1) j crosses at least one link of $\mathcal{L}_1$, so by (8) and statement 3 of this lemma feedback to any packet which left the source after time t_2 returns to the source with the stamped rate set to τ_1. Denote the time of the return of this feedback to the source by t_j^4

It then follows that for any $t > t_4^j$ ρ_s is set to τ_1, so statement 4 of the lemma holds for all $t \geq t_3 = \max_{j \in \mathcal{S}_1} t_4^j$.

The proof of statement 5 is almost identical to the proof of statement 4. Statement 6 follows from statements 2 and 3, which are aleady proved.

Note that t_3 is the time when all sessions in $\mathcal{S}_1$ have completed their third round-trip.

Finally, it follows from conditions 1-6 and the operation of the algorithm that any session $j \in \mathcal{S}_1$ will be marked at every link on its way as soon as the first packet of j is sent after time t_j^4 passes that link and will remain marked ever after as long as the set of sessions remains unchanged. Thus, there exists time t_4 such that statement 7 of this lemma holds for all j.

This completes the proof of Lemma 4.4.

Note that t_4 is the time when all sessions in $\mathcal{S}_1$ have completed their fourth round-trip. We have just shown that it takes at most four round-trips of the "slowest" session in $\mathcal{S}_1$ to ensure that all sessions in $\mathcal{S}_1$ have reached their optimal rates and are marked with their optimal rates at all links in their routes. This result will be used in section 5 to obtain an upper bound on convergence time.

References

[1] Bertsekas, D., Galager, R., 1992. Data Networks, ch. 6, Prentice Hall, Englwood Cliffs, N.J.

[2] Gafni, E., and Bertzekas, D., 1984 "Dynamic Control of Session Input Rates in Communication Networks", IEEE Trans. Automat. Control, AC-29, pp. 1009-1016.

[3] Clark, D., Schenker, S., Zhang, L. "Supporting Real-Time Applications in an Integrated Services Packet Network: Architecture and Mechanism", Proc. ACM SIGCOMM '92, Baltimore, August 1992, pp 14-26.

[4] Clark, D., Tennenhouse, D., " Architectural Considerations for a New Generation of Protocols, ", Proc ACM SIGCOMM '90, Vol. 20, No. 4, Philadelphia, September 1990, pp 200-208.

[5] Demers, A., Keshav, S., Shenker, S. 1990. "Analysis and Simulation of a Fair Queuing Algorithm", Internetwork: Research and Experience, Vol.1, No. 1, John Wiley & Sons, pp 3-26.

[6] Gafni, E., 1982. The Integration of Routing and Flow Control for Voice and Data in Integrated Packet Networks", Ph.D. thesis, MIT, Dept of Electrical Engineering and Computer Science, Cambridge, MA.

[7] Gerla, M., Chan, W., de Marca, J.R.B "Fairness in Computer Networks", Proc. IEEE Int. Conf. Commun., June 1985, pp.1384-1389.

[8] Gerla, M., Kleinrock, L. 1980, "Flow Control: A Comparative Survey," IEEE Trans. Com., COM-28, pp.553-574.

[9] Golestani, S. 1980. "A Unified Theory of Flow Control and Routing on Data Communication Networks", Ph.D. thesis, MIT, Dept of Electrical Engineering and Computer Science, Cambridge, MA.

[10] Golestaani, S. "Congestion Free Real-Time Traffic in Packet Networks", Proc. of IEEE INFOCOM '90, San Francisco, Cal., June 1990, pp. 527-536.

[11] Golestani, S. "A Stop-and-Go Queuing Framework for Congestion Management", Proc. of ACM SIGCOM '90, Vol. 20, No.4, September 1990, pp.8-18.

[12] Hahne, E., Gallager, R., "Round-Robin Scheduling for Fair Flow Control in Data Communication Networks", (report LIDS-P-1537). Cambridge, MA: MIT Laboratory for Information and Decisions Systems.

[13] Hahne, E., 1986. "Round-Robin Scheduling for Fair Flow Control in Data Communication Networks", Ph.D. thesis, MIT, Dept of Electrical Engineering and Computer Science, Cambridge, MA.

[14] Hahne, E."Round-Robin Scheduling for MaxMin Fairness in Data Networks", IEEE J. on Sel. Areas in Comm., vol 9, No.7, September 1991.

[15] Hayden, H. 1981. "Voice Flow Control in Integrated Packet Networks', (report LIDS-TH-1152). Cambridge, MA: MIT Laboratory for Information and Decisions Systems.

[16] Jaffe, J. 1981. "A Decentralized Optimal Multiple-User Flow Control Algorithm", Proc. Fifth Int. Conf. Comp. Comm, October 1980, pp 839-844

[17] Jaffe, J. "Bottleneck Flow Control", IEEE Trans. Comm., COM-29, No 7. pp954-962.

[18] Jacobson, V. "Congestion Avoidance and Control" , Proc. ACM SIGCOMM '88, Stanford, Calif., August 1988, pp.314-329.

[19] Jain, R., Chiu D.-M., Hawe, W., " A Quantitative Measure of Fairness and Discrimination for Resource Allocation in Shared Computer System", DEC-TR-301, Digital Equipment Corporation. 1984

[20] Jain, R., Ramakrishnan, K., " Congestion Avoidance in Computer Networks With a Connectionless Network Layer. Part 1: Concepts, Goals and Methodology", DEC-TR-507, Digital Equipment Corporation. 1987

[21] Jain, R., "Myths about Congestion Management in High-Speed Network," DEC-TR-724, Digital Equipment Corporation. 1990.

[22] Nagle, J., "On Packet Switches with Infinite Storage", IEEE Trans. on Comm., Vol.35, No.4, April 1987, pp. 435-438

[23] Mosley, J. 1984. " Asynchronous Distributed Flow Control Algorithms", Ph.D. thesis, MIT, Dept of Electrical Engineering and Computer Science, Cambridge, MA.

[24] Partridge, C., 1993. Gigabit Networking, ch. 3,4,7,8,11-13, Addison-Wesley Publishing Co, Inc.

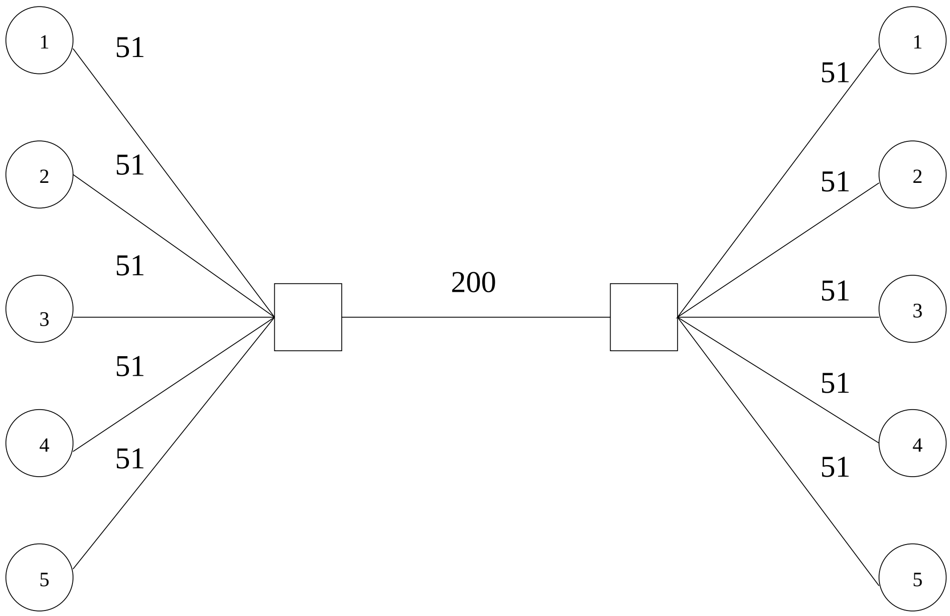

Figure 1: Experiment 1: 5 sessions sharing one bottleneck link. Capacities of the links are shown next to the links (x 10^{-3}).

[25] Parekh, A., 1992. " A Generalized Processor Sharing Approach to Flow Control in Integrated Services Networks," Ph.D. thesis, MIT, Dept of Electrical Engineering and Computer Science, Cambridge, MA.

[26] K.K.Ramakrishnan, Raj Jain, Dah-Ming Chiu. " Congestion Avoidance in Computer Networks With a Connectionless Network Layer. Part IV: A Selective Binary Feedback Scheme for General Topologies Methodology", DEC-TR-510, Digital Equipment Corporation. 1987

[27] Shenker, Scott, "A Theoretical Analysis of Feedback Flow Control", Proc. SIGCOMM'90, Philadelphia, PA, September 1990, pp.156-165.

[28] Zhang, L., Shenker, S., Clark, D., " Observations on the Dynamics of a Congestion Control Algorithm: The Effects of Two-Way Traffic," Proc. ACM SIGCOMM '91, Zurich, September 1991, pp.133-148.

Session/time	0-27	27-42	42-63	63-86	86-100
1	11000	-	-	-	-
2	11000	11000	-	-	11000
3	11000	11000	-	2000	2000
4	11000	11000	11000	11000	11000
5	11000	11000	11000	11000	11000

Table 1: Experiment 1: Demand of sessions 1-5 at different times.

Session/time	0-27	27-42	42-63	63-86	86-100
1	4000	-	-	-	-
2	4000	5000	-	-	6000
3	4000	5000	-	2000	2000
4	4000	5000	10000	9000	6000
5	4000	5000	10000	9000	6000

Table 2: Experiment 1: Optimal rates of sessions 1-5 at different times.

Session/time	0	27	42	63	86
1	2	-	-	-	-
2	2	3	-	-	4
3	2	2	-	0	0
4	1	1	1	4	4
5	2	3	2	4	2

Table 3: Experiment 1: Number of round-trips required for sessions 1-5 to stabilize to new optimal rates after changes in network load 1-5.

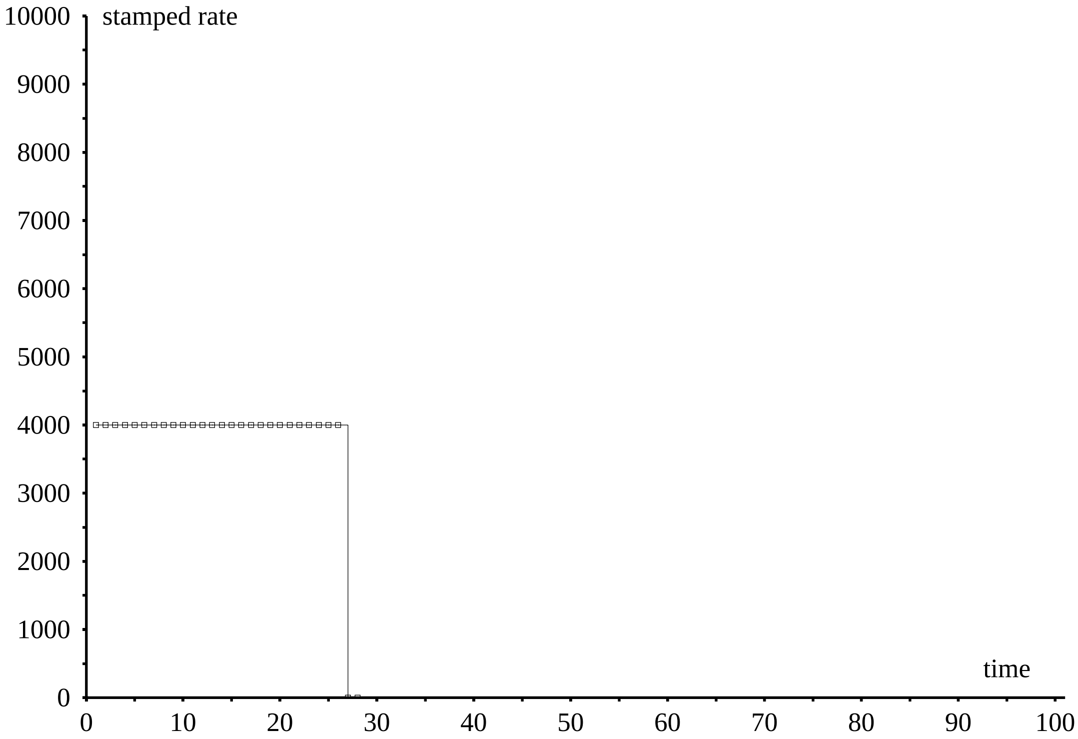

Figure 2: Experiment 1: Session 1. Boxes correspond to round-trip epochs. This session reaches its optimal rate after the first round-trip; it exits at time 27 and never returns.

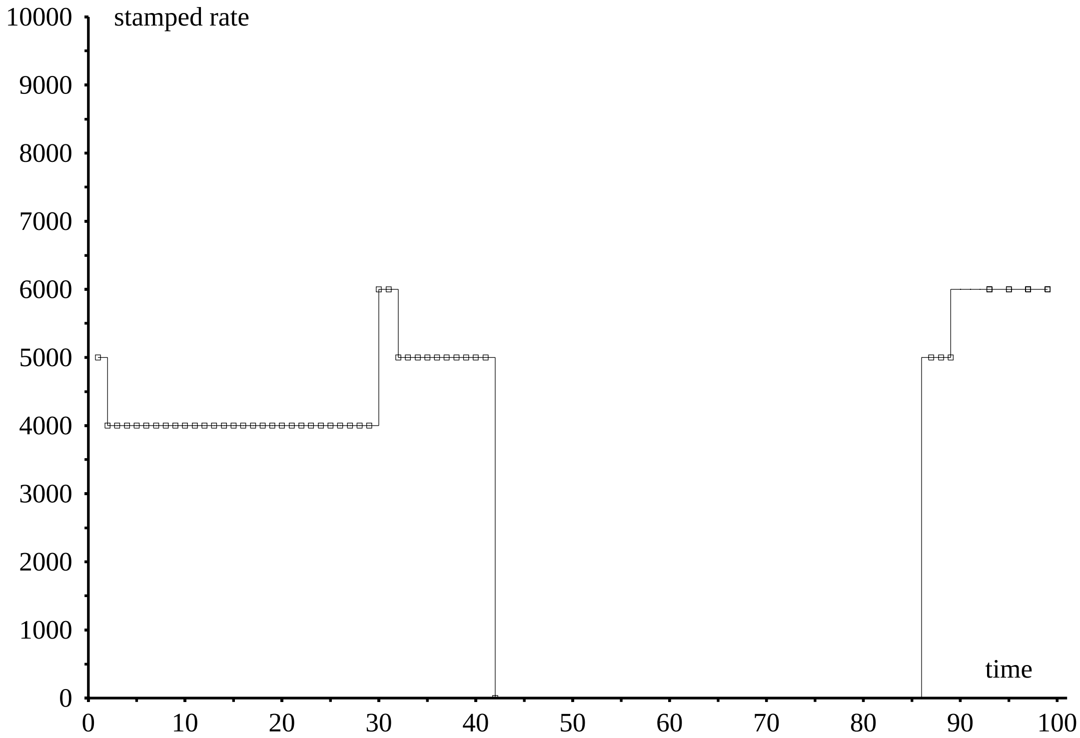

Figure 3: Experiment 1: Session 2. Boxes correspond to round-trip epochs. This session takes 2 round-trips to stabilize ti its original optimal rate of 4000. At time 27 session 1 exits, and this session restabilizes to the new optimal rate of 5000. At time 42 the session exits and re-enters at time 86, stabilizing to its new optimal rate of 6000.

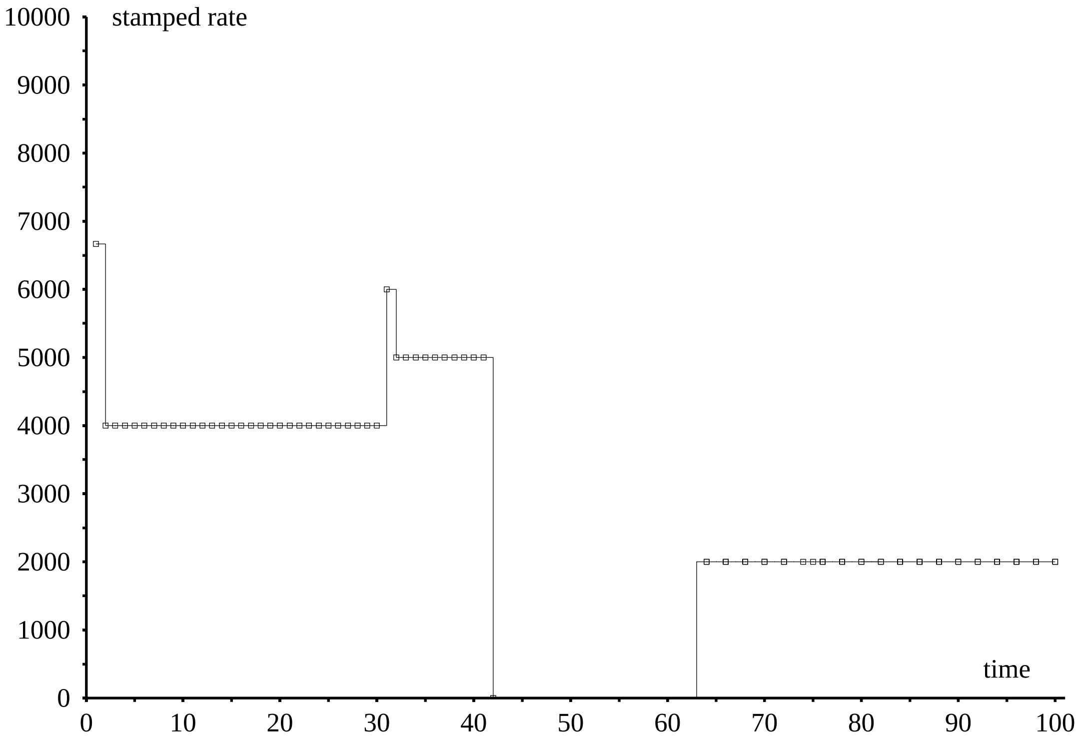

Figure 4: Experiment 1: Session 3. Boxes correspond to round-trip epochs. This session exits at time 42 and re-enters at time 63 with a low demand of 2000. Its behavior up to time 42 is similar to Session 2. Since its demand after time 2000 is lower than the equal share of all sessions in the network, its optimal rate is exactly equal to its demand.

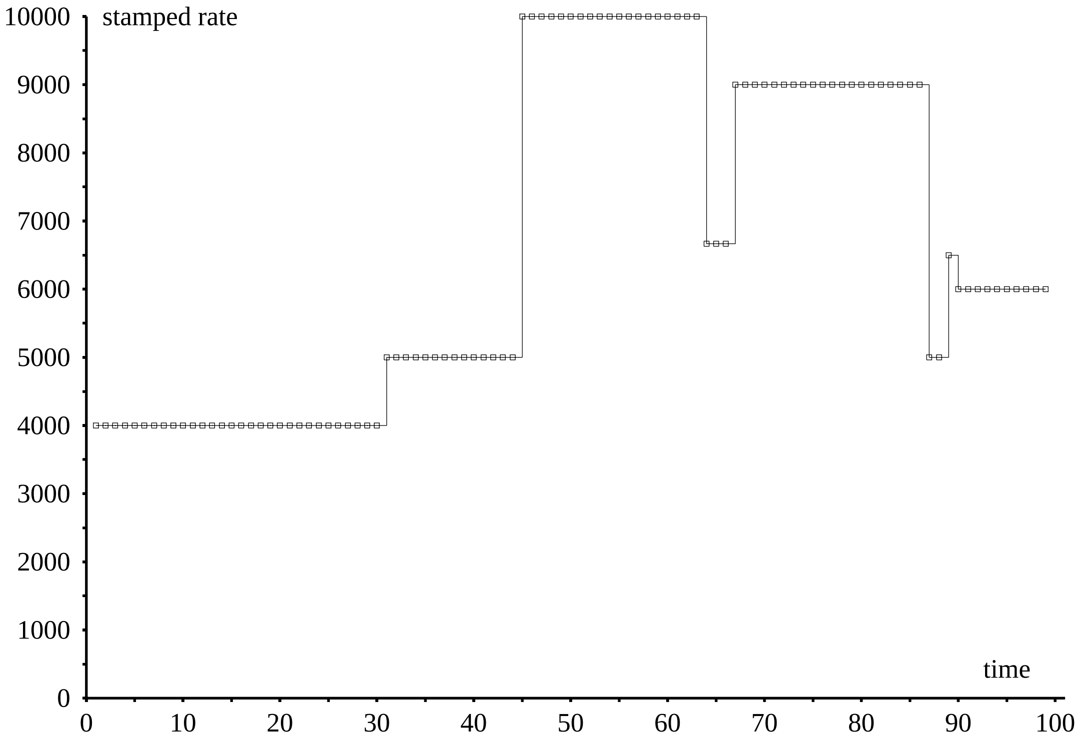

Figure 5: Experiment 1: Session 4. Boxes correspond to round-trip epochs. This session does not exit during the experiment. At times 27, 42, 63 and 86 other sessions enter or exit. It can be seen that the session quickly stabilized to the new optimal rates.

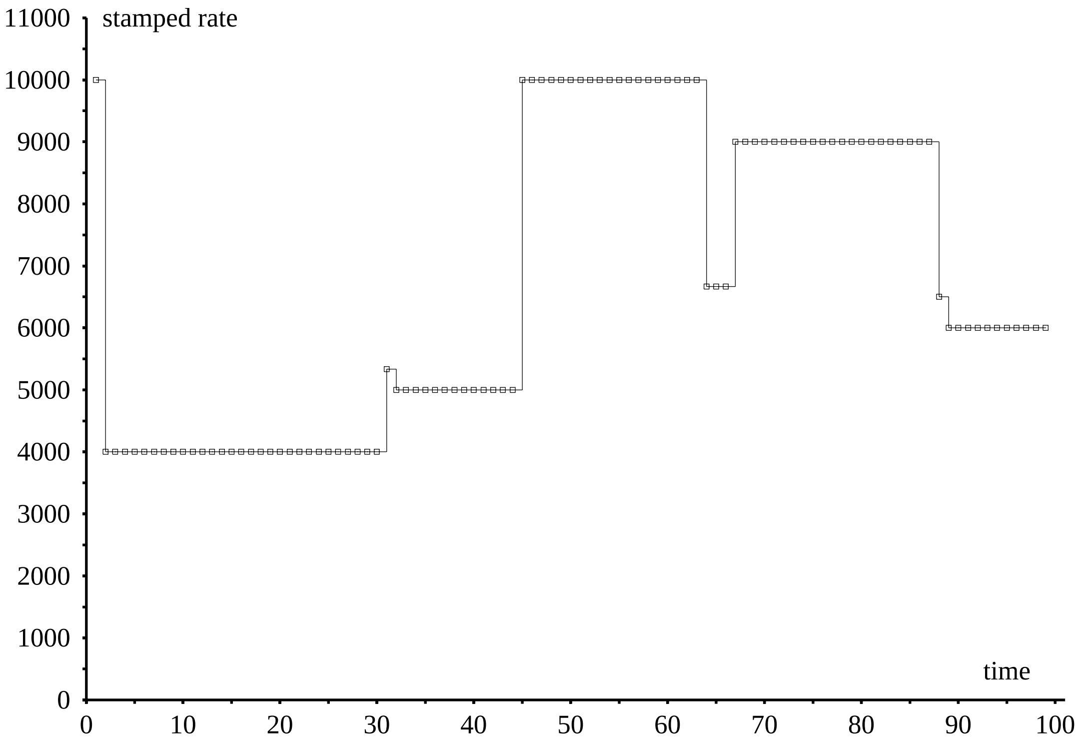

Figure 6: Experiment 1: Session 5. Boxes correspond to round-trip epochs. The behavior of this session is similar to Session 4.

session	s1	s2	s3	s4	s5	s6	s7	s8
demand	3000	4000	5000	6000	7000	8000	9000	10000

session	s9	s10	s11	s12	s13	s14	s15
demand	11000	12000	13000	14000	15000	16000	17000

Table 4: Experiment 2: Session demands (optimal rates are exactly session demands in this experiment.

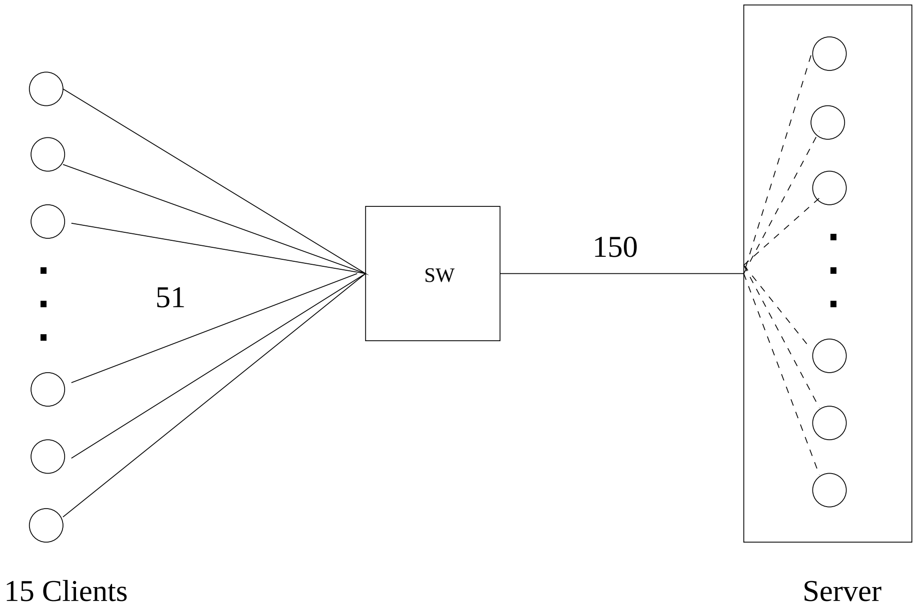

Figure 7: Experiment 2: Configuration. Numbers next to the links are capacites (x 10^{-3})

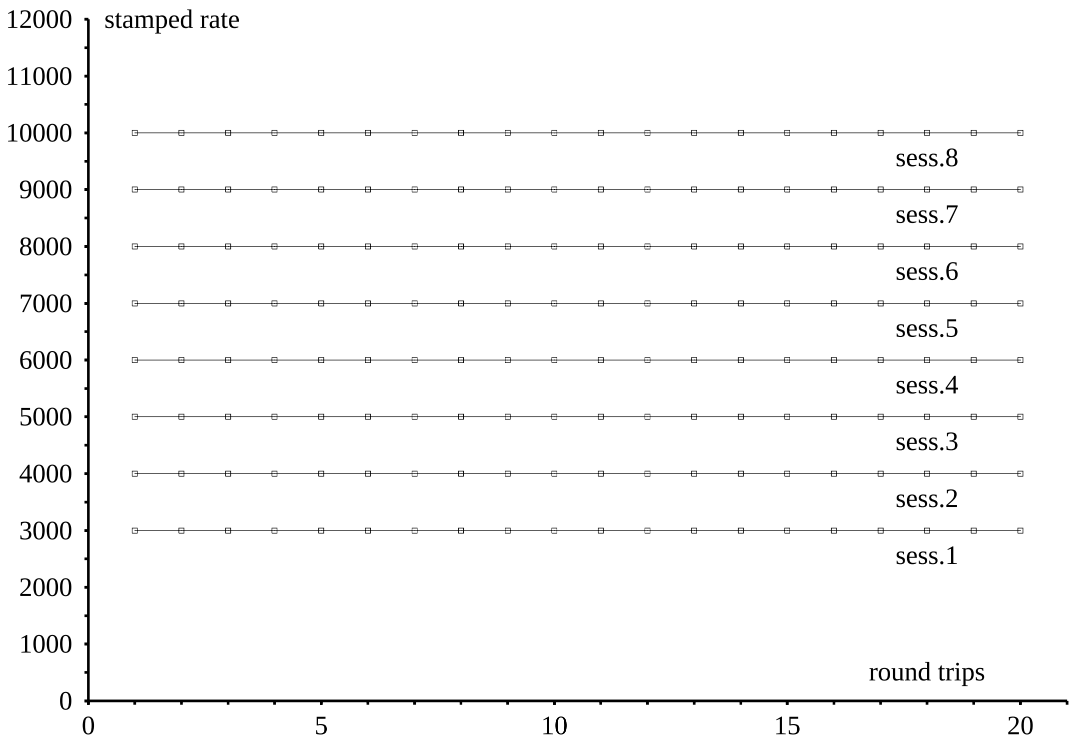

Figure 8: Experiment 2: Sessions 1-8.

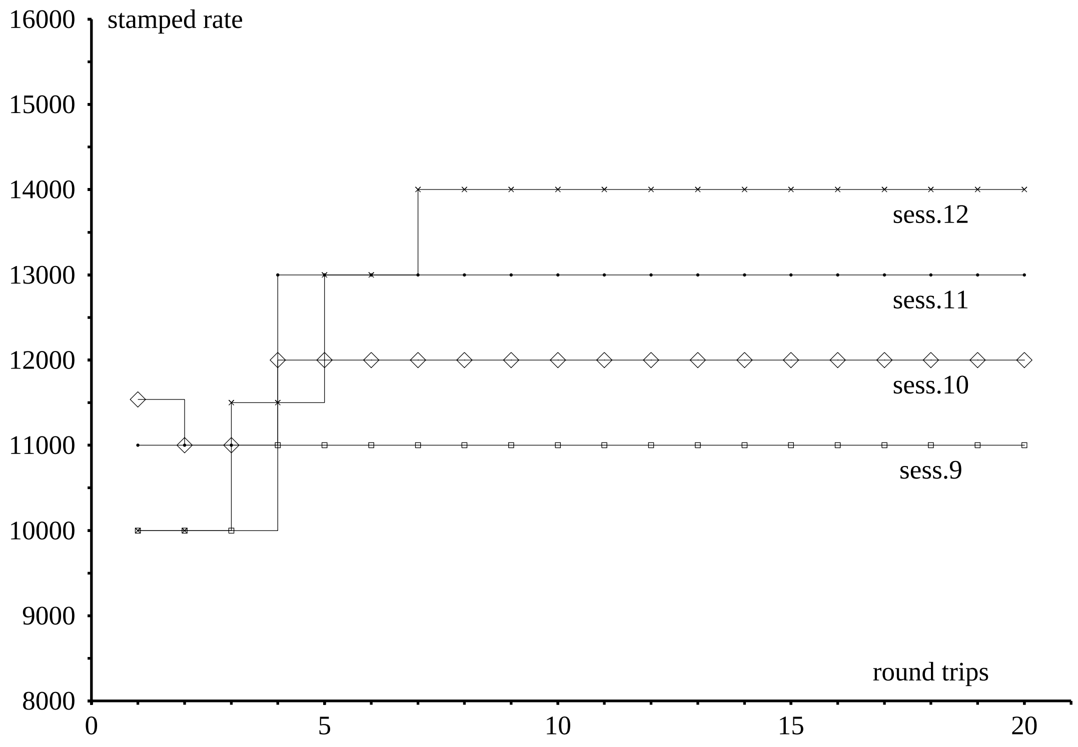

Figure 9: Experiment 2: Sessions 9-12.

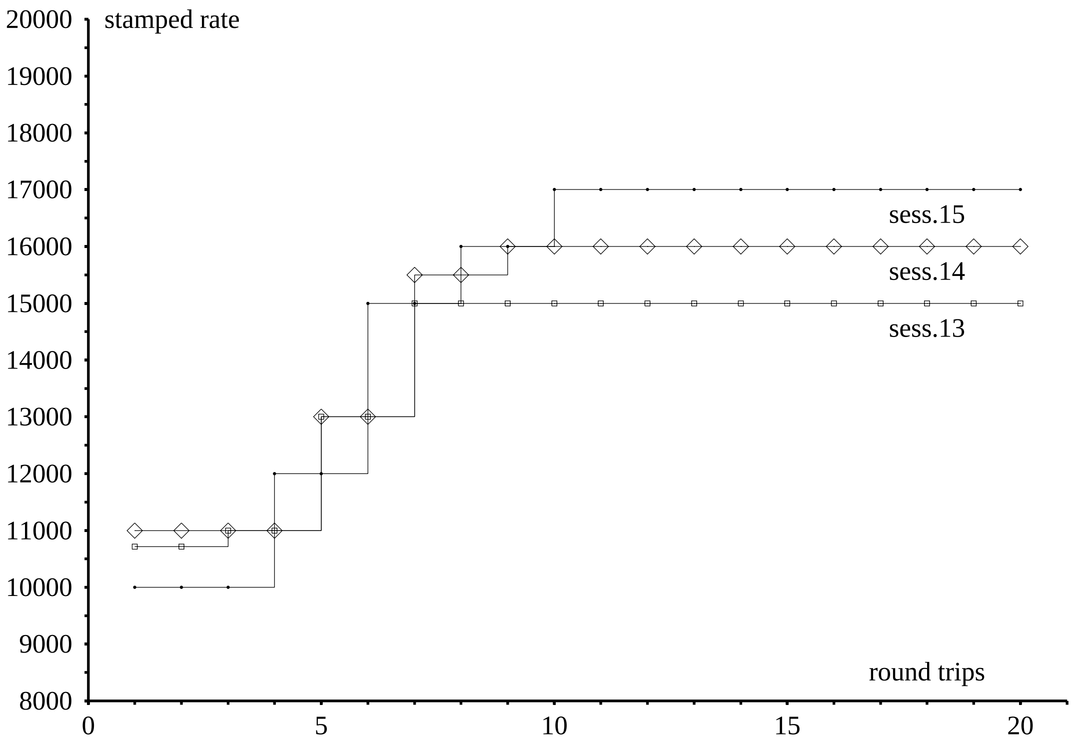

Figure 10: Experiment 2: Sessions 12-15.

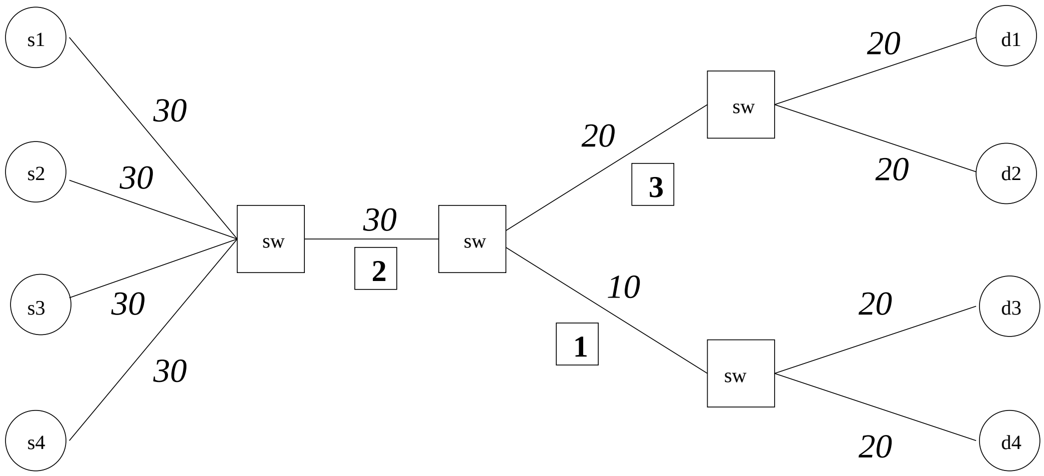

Figure 11: Experiment 3: Configuration. Demands of all session are large (40000). Numbers in italics next to the links are capacities (x 10^{-3}.) The bottleneck links are numbered by boxed bold figures in a box . Link 1 is the bottleneck for sessions 3 and 4, while links 2 and 3 are bottleneck for sessions 1 and 2. Optimal rate of sessions 1 and 2 is 10000, optimal rate of sessions 3 and 4 is 5000.

1	2	3	4
4000	3000	2000	1000

Table 5: Experiment 4: Optimal rates of sessions 1-4.

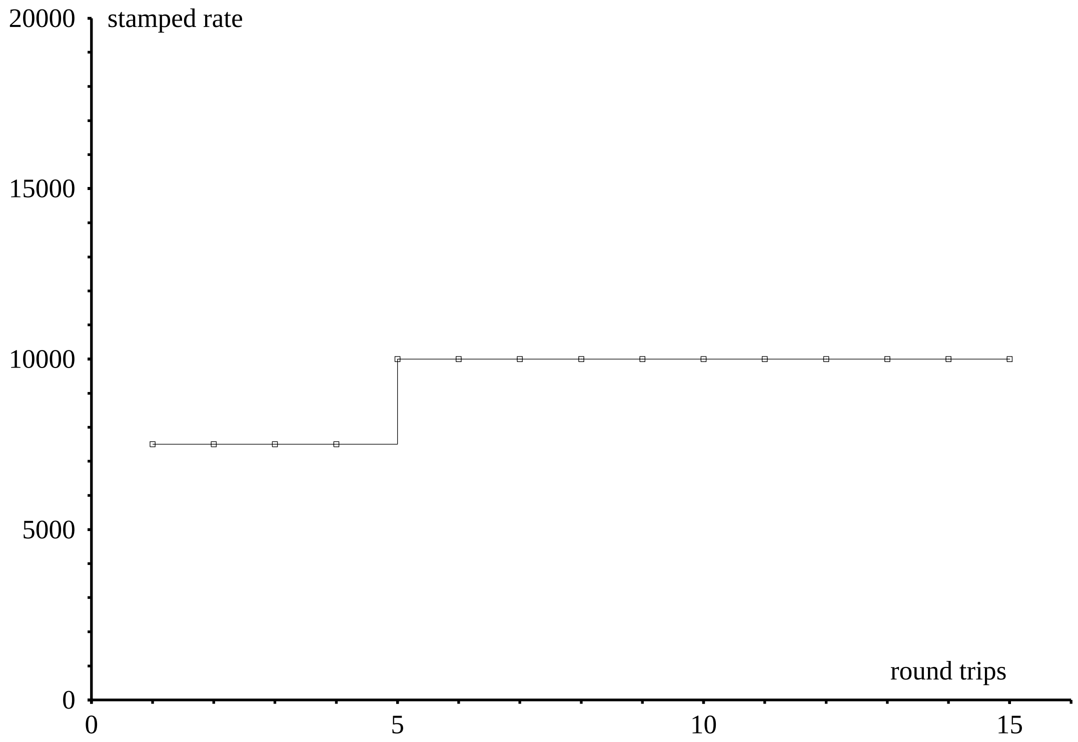

Figure 12: Experiment 3: Session 1 (optimal rate 10000).

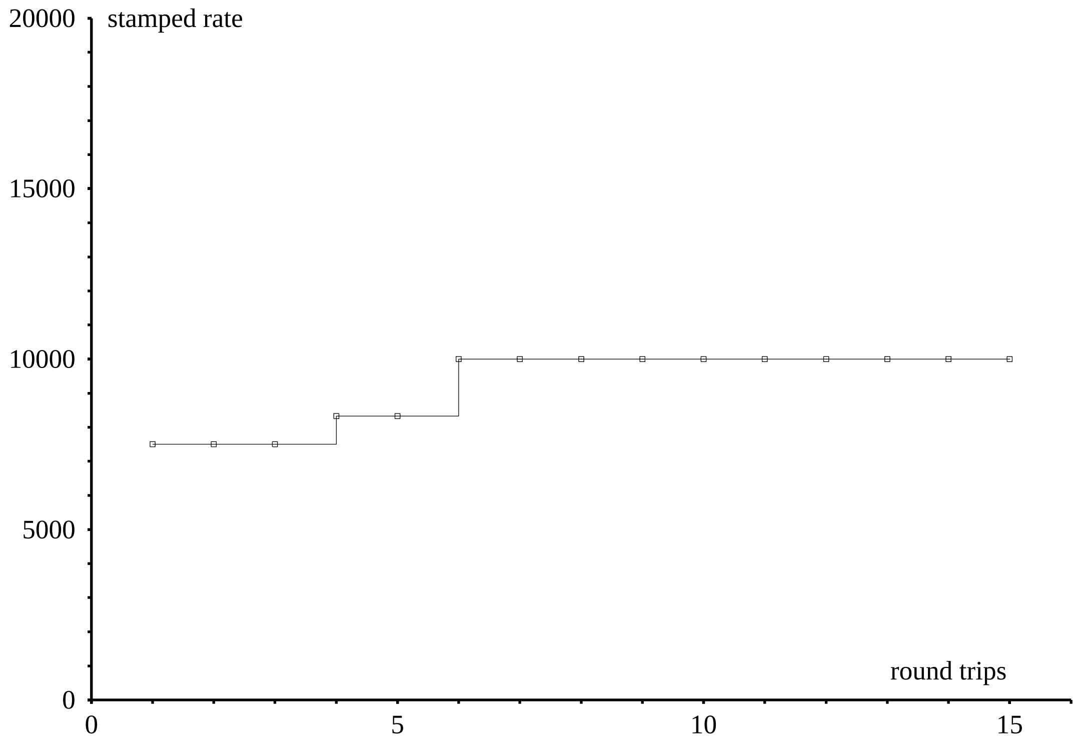

Figure 13: Experiment 3: Session 2 (optimal rate 10000).

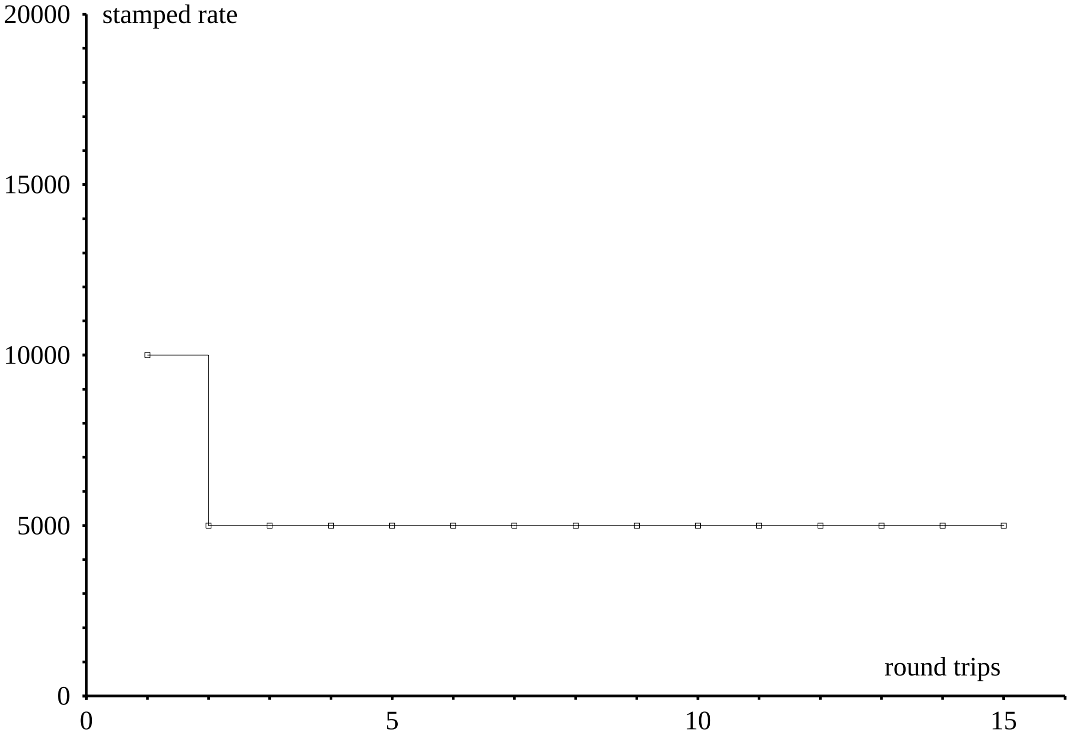

Figure 14: Experiment 3: Session 3 (optimal rate 5000).

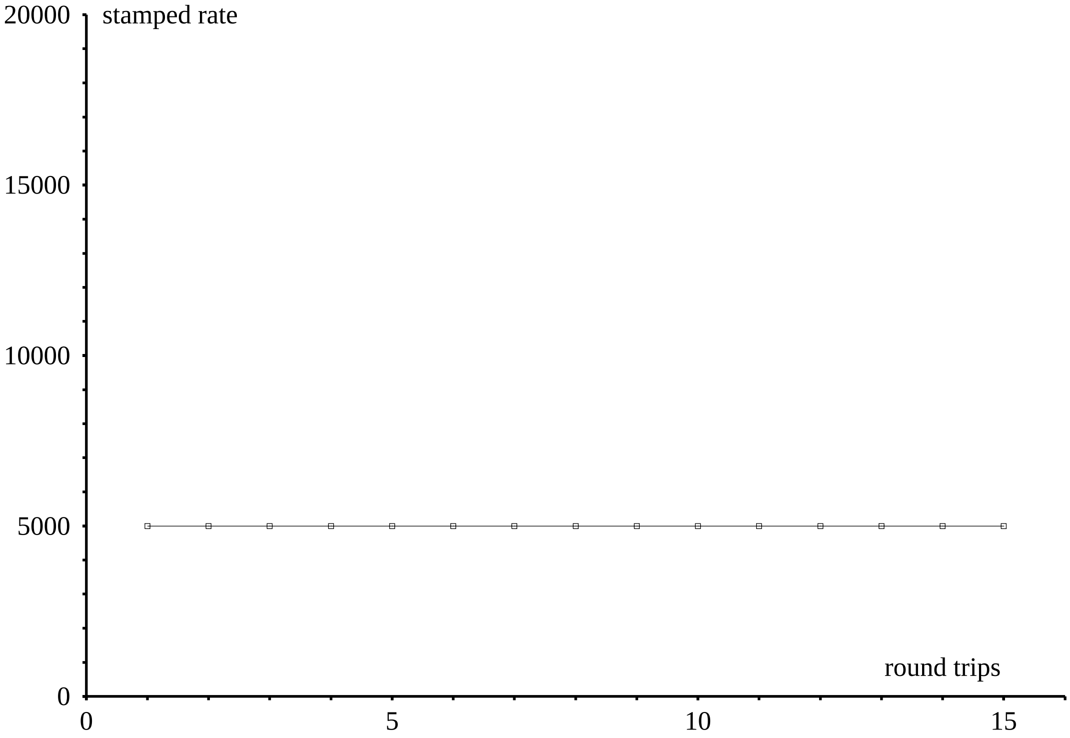

Figure 15: Experiment 3: Session 4 (optimal rate 5000).

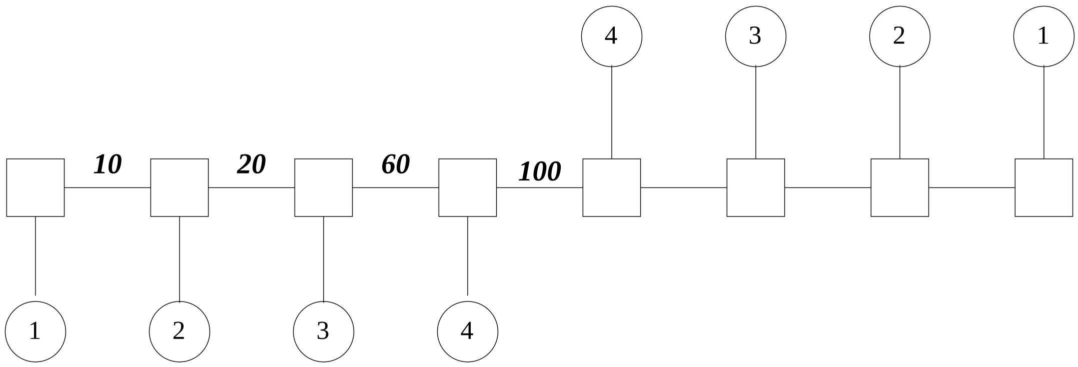

Figure 16: Experiment 4: Configuration.

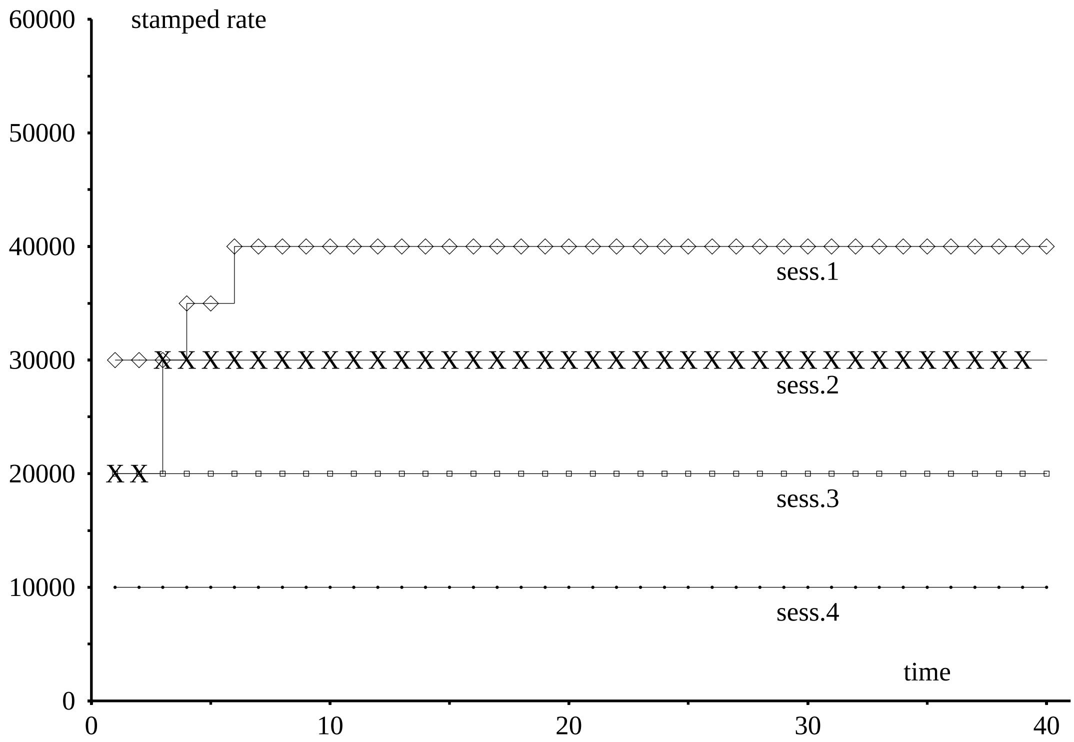

Figure 17: Experiment 4: Sessions 1-4, Simultaneous start.

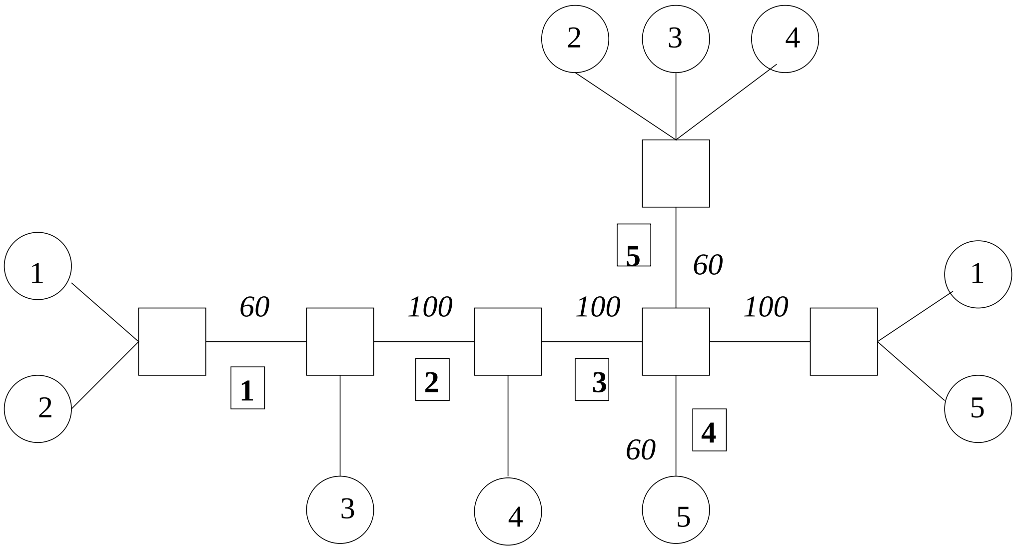

Figure 18: Experiment 5: Configuration. Sessions 1-5 enter at time 0. Then session 3 exits at time 15. Then sessions 1 and 2 exit at time 48. Finally session 1 reenters at time 67. Sessions 2, 3 and 4 share the first-level bottleneck link 5. Session 1 is bottlenecked by link 1. Finally, session 5 is bottlenecked by link 4. Capacities of the links are given in italics (x 10^{-3}). Bottleneck links are numbered in boxed bold print. Links which are not numbered are not bottlenecks.

Session/time	0-15	15-48	48-67	67-100
1	4000	3000	-	5000
2	2000	3000	-	-
3	2000	-	-	-
4	2000	3000	6000	5000
5	6000	6000	6000	5000

Table 6: Experiment 5: Optimal rates of sessions 1-5 at different times.

Session/time	0	15	48	67
1	4	5	-	2
2	2	2	-	-
3	2	-	-	-
4	1	3	2	1
5	7	6	3	1

Table 7: Experiment 5: Number of round-trips required for sessions 1-5 to stabilize to new optimal rates after changes in network load 1-5.

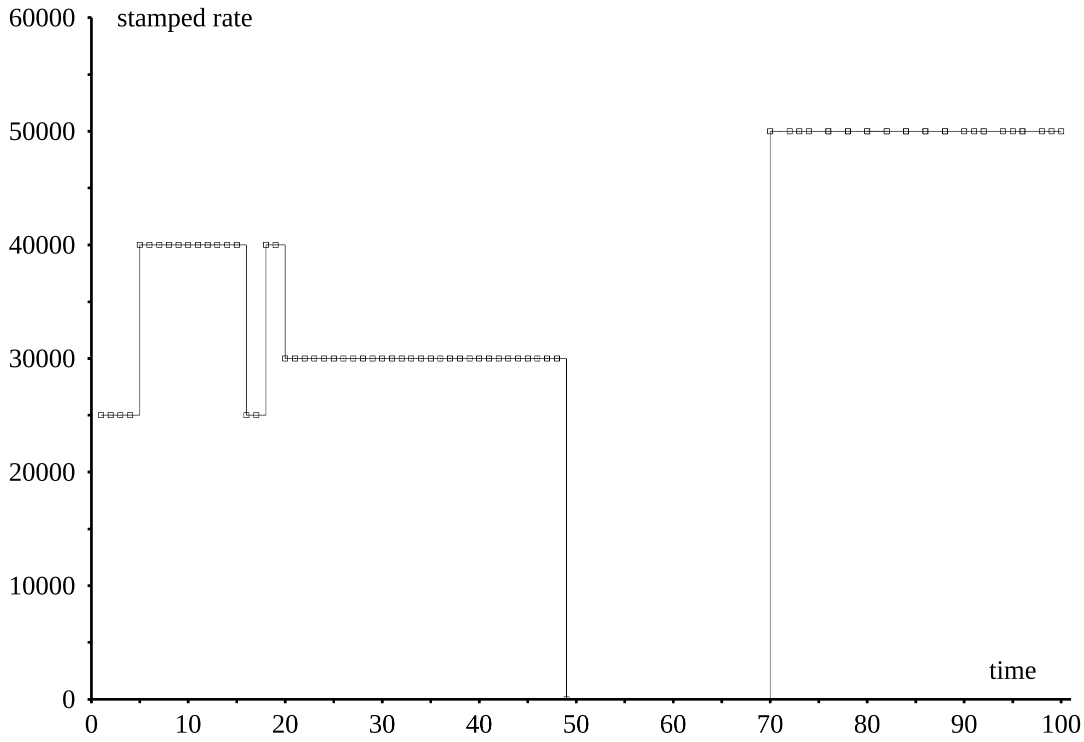

Figure 19: Experiment 5: Session 1. Exits at time 48 and reenters at time 67.

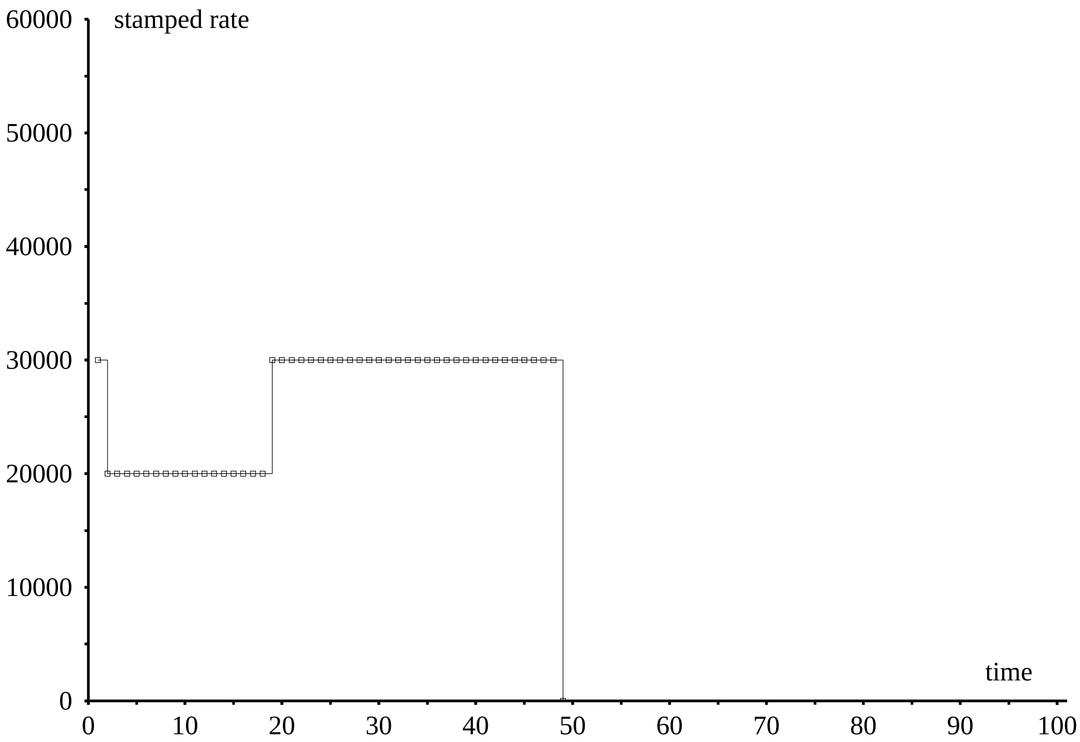

Figure 20: Experiment 5: Session 2. Exits at time 48.

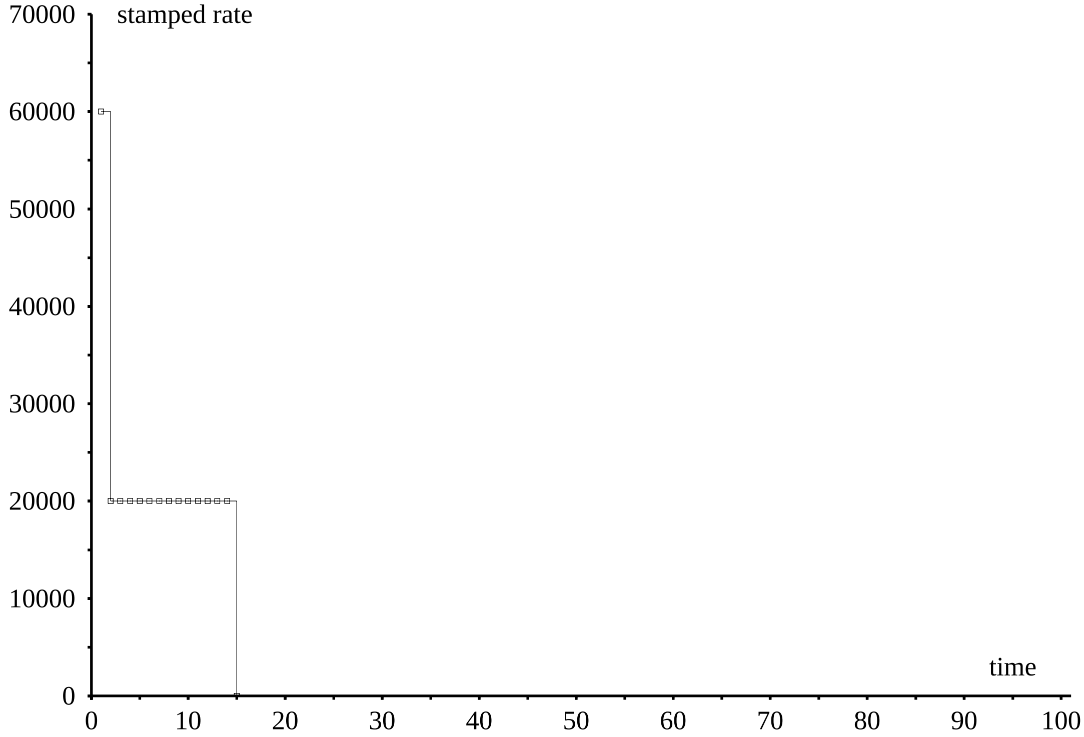

Figure 21: Experiment 5: Session 3. Exits at time 15.

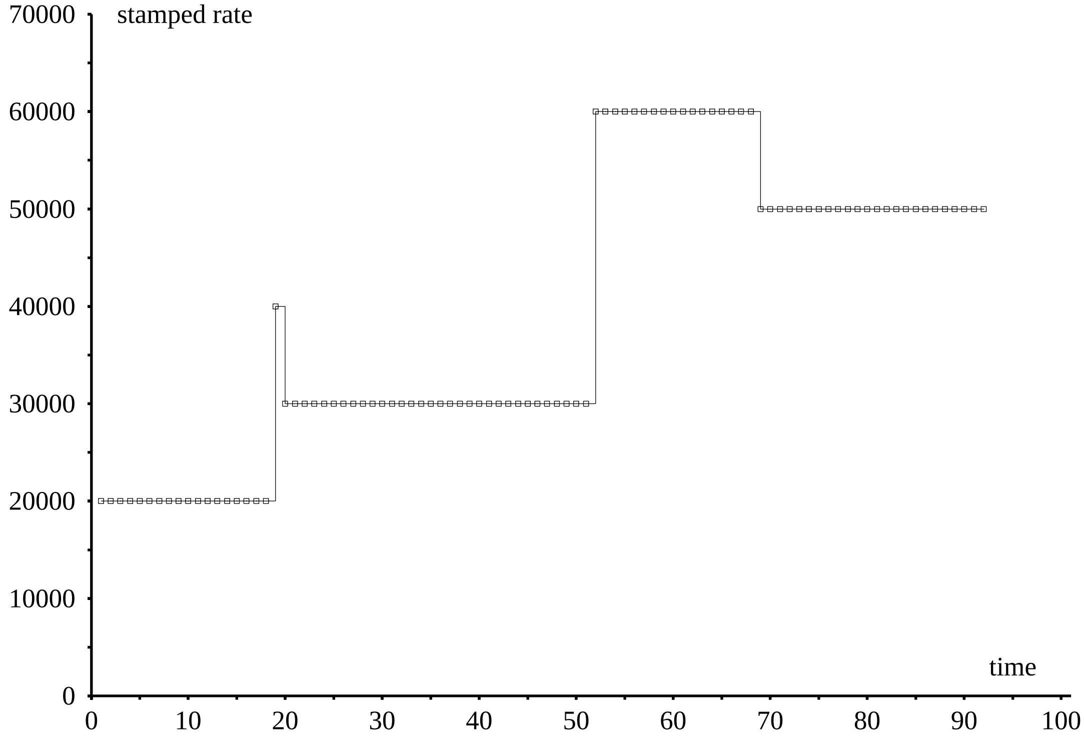

Figure 22: Experiment 5: Session 4.

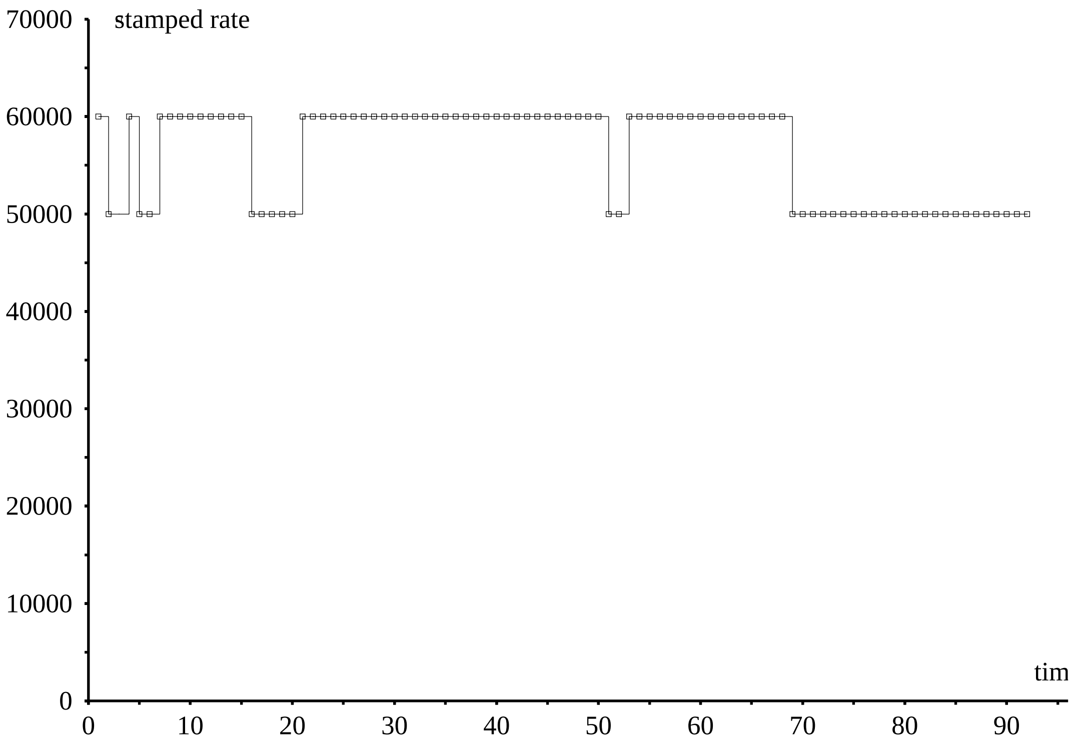

Figure 23: Experiment 5: Session 5.