

# Warp control: a dynamically stable congestion protocol and its analysis

Kihong Park  
Computer Science Department  
Boston University  
Boston, MA 02215

## Abstract

This paper presents a distributed, end-to-end congestion control protocol for use in high-traffic packet switched networks. The network is represented as a stochastic single-server queue, with arrival rates being the control variables. A time-stamp based measure of network state called *warp* is defined, and it is shown to be an estimator of network utilization. Congestion is modeled explicitly using *unimodal load-service rate functions*, and its monotonicity property is exploited to yield characterizations of stability and optimality. A protocol based on “perfect” information is analyzed, whose prowess is then shown to be emulated by one which only uses locally computable, delayed information. The main effect of a unimodal load-service function is to induce a division of the phase space into stable and unstable regions, the optimal operating point being its “boundary.” Protocols are devised for dealing with each regime separately, *rate adjustment protocol* being the control that guides the system to the optimal operating point. *Proactive rate protocol* and *reactive rate protocol* deal with the issue of the optimal operating point being near to the unstable zone. Protocols for handling fairness and structural perturbation augment the basic suite. The analysis is supported by simulations showing the global dynamical properties of the system.

## 1 Introduction

This paper presents a distributed end-to-end congestion protocol and its analysis of stability, optimality, and fairness. The present approach is closest in flavour to the rate-based access control schemes of [3, 5, 13, 15, 17, 20] where a variety of analytical methods are employed to determine the effects of various control laws under different network assumptions. This paper differs in two respects. First, we propose a time-stamp based measure of network state called *warp* and show that it estimates utilization for networks modeled

Permission to copy without fee all or part of this material is granted provided that the copies are not made or distributed for direct commercial advantage, the ACM copyright notice and the title of the publication and its date appear, and notice is given that copying is by permission of the Association for Computing Machinery. To copy otherwise, or to republish, requires a fee and/or specific permission.

SIGCOMM'93 - Ithaca, N.Y., USA /9/93

© 1993 ACM 0-89791-619-0/93/0009/0137...\$1.50

as FCFS single-server queues. This leads to a natural control law which forms the corner stone of the warp protocol suite. Preliminary experiments on a network of seven workstations linked via Ethernet and running a communication-intensive application seem to support its usefulness even in small scale systems [9]. Second, we explicitly model a network's congestion mechanism via a unimodal curve relating effective throughput to *actual* load which we call the *load-service rate function*. Similar curves have been discussed that relate throughput to *offered* load [6, 7], which necessitates the inclusion of control laws in its formulation. In the present scheme, we view the unimodal shape of the load-throughput curve as an *inherent* property shared by a multitude physical of networks being, in some sense, “independent” of any particular control algorithm. To illustrate our claim, we compare a control having access to perfect information (“Big Brother”) against warp control and show that both suffer under potential instability in the long run. The dynamics of protocols that incur a variance in their control variable due to stochastic effects or less-than-perfect information is explicated in this framework to yield qualitative insight into how difficult a problem congestion may be, and what one might expect to get achieved.

Recently, the advent of broadband packet networks has raised a special set of issues induced by the speed-mismatch between the large increase in data transfer rate on one hand and limited processing speed at routing stations on the other [4, 21]. Several methods have been proposed to remedy this situation, falling under the category of “proactive” (as opposed to “reactive”) controls [7], whose goal is to prevent congestion from occurring by continuously adjusting input rates, rather than wait for congestion to occur and then react to it [2]. The protocol presented in this paper is no exception in this respect, emphasis being placed on avoiding congestion, subject to operating at a near-optimal throughput.

Another important consideration in the larger context of computing systems is to view the communication network as a fundamental component to be managed by distributed operating systems. In this context, a principal requirement placed upon a communication network, besides its data transfer rate, is its reliability. For the network to be an effective integral part of a distributed computing environment, it has to be predictable. “Congestion collapses” [11] have highly disruptive ripple ef-

fects. Therefore maintaining a stable throughput becomes as important as providing sufficient bandwidth. Our protocol deals with these issues head on. This paper is organized as follows. First, we formalize the main variables of congestion control of interest to us. Second, we will show how two agents interacting over a communication network, each with its own local time frame, can estimate a global property of the medium, its utilization  $\rho$ . Third, we construct a distributed flow control algorithm and derive the equations governing its behavior. Fourth, we will analyze the equations with respect to stability, optimality, and fairness. Lastly, we report simulation results and conclude with refinements and extensions.

## 2 Network model and congestion control

### 2.1 Network model

Let  $N$  be a store-and-forward communication network with some fixed topology, consisting of a set of nodes  $p_1, p_2, \dots, p_n$ . Each node  $p_i$  can be a source, sink, as well as a switching station for packets traversing the network. Each node is connected to  $N$  via a network interface  $I_i$ , and all packet submissions as well as receptions are cleared through  $I_i$ . The interface executes two primary protocols, one for sending data and one for receiving data. A packet  $m_{ij}$ , destined from node  $j$  to node  $i$ , is said to be *in* the network, if  $m_{ij}$  has passed through  $I_j$ 's message generation protocol (*MGP*), but has not yet been processed by  $I_i$ 's message reception protocol (*MRP*). The *load* of the network,  $L(t)$ , at time  $t$ , is defined to be the total number of packets in the network. The message generation rate  $\lambda_i(t)$  of node  $p_i$  is defined as the number of packets sent out from  $I_i$  at  $t$ . The total message generation rate is  $\lambda(t) = \sum_{i=1}^n \lambda_i(t)$ .

The service rate  $\mu(t)$  of  $N$  is defined to be the total number of packets delivered over the entire network at time  $t$ . Although we will allow each node  $p_i$  to have its own local clock, assume the existence of an absolute time frame to which every local time frame can be mapped. The system is completely specified by indicating what the message generation and message reception protocols are, what we assume about network  $N$  including its service rate  $\mu(t)$ , and the message generation rate  $\lambda_i(t)$  at each node's interface. The goal of our protocols *MGP* and *MRP* is to adjust  $\lambda_i(t)$  as a function of  $\mu(t)$  and  $L(t)$  so as to maximize utilization  $\rho(t) = \lambda(t)/\mu(t)$ , while minimizing congestion which will be reflected in  $\mu(t)$  as a function of  $L(t)$ . Thus  $\mu(t)$  and  $L(t)$  are the quantities to be controlled, and the control variables are the  $\lambda_i(t)$ 's. Mainly out of notational convenience, all quantities will be treated as continuous variables with suitable discretizations being implicit where applicable.

In this highly abstracted viewpoint of the network, no reference to routing can be made unless the network

topology is specified. In this paper, we will divorce flow control from routing by formulating a theory that is independent of network topology. In particular, we will view the network  $N$  as a giant single-server queue, and derive our results from this assumption. Our formulation actually leads to a more general characterization of  $N$  as a coupled system of  $n^2 - n$  single-server queues  $N_{ij}$ ,  $i \neq j$ . In order to ignore the complex issue of what couplings to consider, we will assume  $N$  to be one single-server FCFS queue, with service rate  $\mu(t)$ . Congestion control can always be modeled at the link level by unifying routing with flow control. Schemes such as these have been investigated in [12, 14].

### 2.2 Utilization versus congestion

There exist many characterizations of congestion phenomena and its causes, all more or less stating that overflowing a network results in a real decrease in its service rate. This may be due to buffer-overflow and time-out of packets, deadlock, and a host of other problems. Elegant accounts of the origins of congestion can be found in [18, 19]. Such considerations allow us to conclude that network congestion is primarily a function of the *actual* load  $L(t)$  in the network. A commonly used scheme for depicting the relationship between "offered" load and effective throughput for controlled and uncontrolled networks can be found in [6, 7]. Figure 1 shows a curve relating actual load to throughput making explicit our assumption that congestion is synonymous to a drop in network service rate  $\mu$  as a function of load  $L$ . This leads

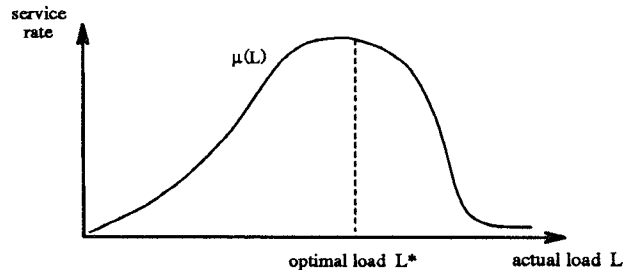


Figure 1: This figure shows the unimodal load-service function which relates actual load to service rate.

to the notion of an *inherent* optimal load  $L^*$  (in general, an interval  $(L_l^*, L_u^*)$ ) of a network  $N$ , and its associated optimal service rate  $\mu^*$ . It is not clear that this makes any sense since the load-service curve was defined independent of time and routing/flow control algorithms. Nevertheless, what we want to say is that congestion is an *inherent* property of resource-bounded traffic networks, and for a large class of "admissible" control laws, the family of associated load-service functions possesses an invariant property, its unimodal shape. We will refrain from giving formal justifications, except to say that this is purely an assumption for modeling congestion, whose validity is open to debate. Assuming this to be

true, can an optimal load be maintained? Given the conservation equality for  $L(t)$

$$\frac{dL(t)}{dt} = \lambda(t) - \mu(t), \quad (1)$$

to achieve and maintain maximum throughput,  $\lambda(t) \rightarrow \mu^*$ , as  $t \rightarrow \infty$ . For an *uncontrolled*  $M/M/1$  queue, we know that in steady state, the mean queue length (i.e., expected load) is  $E(L) = \rho/(1 - \rho)$ , hence by  $\rho^* = \lambda(t)/\mu^* = 1$ , optimal utilization cannot be maintained. Since the expected delay of the system,  $E(W)$ , is given by Little's formula as  $E(L) = \lambda E(W)$ , this is often construed to mean that there is an explosion of delay near the optimal service rate. Trade-off measures between delay and throughput such as power, weighted objective functions, and other criteria have been proposed as candidates for defining the optimal "operating point" of a network [6, 10].

Note, the above only applies to uncontrolled systems. If a control law were able to regulate the load in the queue such that  $E(L(t)) \rightarrow L^*$  and  $\text{var}(L(t))$  is kept small (analogously for  $\lambda(t)$ ) as  $t \rightarrow \infty$ , then clearly there is no a priori reason for associating a prohibitive delay with the optimal operating point  $(\mu^*, L^*)$ . That is, it is perfectly possible to operate at maximum throughput without incurring a large delay penalty. This is easily seen in the case where we assume the existence of an all-powerful Orwellian "Big Brother" who knows the value of  $L^*$ , has access to instantaneous information about  $L(t)$ , and furthermore can instantaneously change the message generation rates  $\lambda_1(t), \dots, \lambda_n(t)$  at will. To make the problem interesting, we will let  $\mu(t)$  be given as a stochastic process  $\mu(t) = \bar{\mu} + \omega(t)$  over which Big Brother has no control. Here  $\bar{\mu}$  is the mean service rate and  $\omega(t)$  is white Gaussian noise. First, let us treat the simple case where  $\bar{\mu}$  is constant. Let Big Brother's control law be given by

$$\frac{d\lambda(t)}{dt} = \epsilon(L^* - L(t)) - \frac{dL(t)}{dt}, \quad (2)$$

where  $\epsilon > 0$ . By  $\mu(t) = \bar{\mu} + \omega(t)$ , equations (1) and (2) form a system of stochastic differential equations. Let  $\tilde{\lambda} = \lambda - \bar{\mu}$ , and  $\tilde{L} = L - L^*$ . Then (1) and (2) can be written in matrix form as

$$\begin{bmatrix} d\tilde{\lambda}/dt \\ d\tilde{L}/dt \end{bmatrix} = \begin{bmatrix} -1 & -\epsilon \\ 1 & 0 \end{bmatrix} \begin{bmatrix} \tilde{\lambda} \\ \tilde{L} \end{bmatrix} + \begin{bmatrix} \omega \\ \omega \end{bmatrix}.$$

We will denote this equation by  $d\tilde{\mathbf{x}}/dt = A\tilde{\mathbf{x}} + \mathbf{w}$ . The solution can be expressed using Itô's formulation [1] as

$$\tilde{\mathbf{x}}(t) = e^{At}c + \int_0^t e^{A(t-s)}d\mathbf{w}(s) \quad (3)$$

where  $c$  is a constant that depends on the initial condition. The solution is asymptotically stable if and

only if for all eigenvalues of  $A$ , the real parts are negative. It can be easily checked that this is the case for  $\epsilon > 0$ . Whether the solution is a damped oscillation or not depends on the exact value of  $\epsilon$ , which occurs in the discriminant of the characteristic equation. In steady-state,  $E(\tilde{\mathbf{x}}) = 0$ , and  $\text{var}(\tilde{\mathbf{x}}) = O(\text{var}(\omega))$ . Thus  $\lambda(t) \rightarrow \bar{\mu}$  and  $L(t) \rightarrow L^*$ . Moreover, if  $c$  is normally distributed or constant, then the solution  $\tilde{\mathbf{x}}(t)$  is a Gaussian stochastic process. We conclude that Big Brother fares pretty well.

Let us consider what happens if we let  $\bar{\mu} = \mathcal{G}(L)$  where  $\mathcal{G}$  is strictly unimodal. That is,  $d\mathcal{G}/dL > 0$  if  $L < L^*$ ,  $d\mathcal{G}/dL = 0$  if  $L = L^*$ , and  $d\mathcal{G}/dL < 0$  for  $L > L^*$ . In general,  $\mathcal{G}(L)$  may be highly nonlinear, hence we may not be able to find closed form solutions. Nevertheless, we can exploit the monotonicity property of  $\mathcal{G}(L)$  by making piecewise linear approximations, and thus gain qualitative information regarding the stability of the more complicated system. Let  $I_l$  be a closed interval belonging to  $[0, L^*)$ , and let  $I_r \subseteq (L^*, \infty)$ . Define

$$\mathcal{G}(L) = \begin{cases} \alpha_l + \beta_l L < \mu^*, & \text{if } L \in I_l \\ \mu^*, & \text{if } L = L^* \\ \alpha_r - \beta_r L < \mu^*, & \text{if } L \in I_r \end{cases}$$

where  $\alpha_l, \beta_l$ , and  $\alpha_r, \beta_r$  are constants depending on  $I_l$  and  $I_r$ , respectively, with the restriction that  $\beta_l, \beta_r > 0$ . We obtain a new system of stochastic differential equations

$$\begin{bmatrix} \dot{\lambda} \\ \dot{L} \end{bmatrix} = \begin{bmatrix} -1 & -\epsilon + \beta_i \\ 1 & -\beta_i \end{bmatrix} \begin{bmatrix} \lambda \\ L \end{bmatrix} + \begin{bmatrix} \epsilon L^* + \alpha_i \\ -\alpha_i \end{bmatrix} + \begin{bmatrix} \omega \\ \omega \end{bmatrix}.$$

Denoting the equation by  $d\mathbf{x}/dt = A\mathbf{x} + \mathbf{b} + \mathbf{w}$ , its solution can be expressed as

$$\mathbf{x}(t) = e^{At}c + \int_0^t e^{A(t-s)}\mathbf{b}ds + \int_0^t e^{A(t-s)}d\mathbf{w}(s). \quad (4)$$

Again, asymptotic stability depends on the eigenvalues of  $A$ . The solution to the characteristic equation is given by

$$r = -\frac{1 + \beta_i}{2} \pm \frac{1}{2}\sqrt{(1 + \beta_i)^2 - 4\epsilon}. \quad (5)$$

For  $\beta_i > 0$  (i.e.  $L \in [0, L^*)$ ), we have  $\text{Re}(r) < 0$ , hence the system is stable and converges to the optimal solution  $(\mu^*, L^*)$ . For  $\beta_i < -1$ , at least one eigenvalue will have a positive real part, hence the system will become unstable. Big Brother, by incorporating a damping factor  $\gamma > 0$  via  $\gamma dL(t)/dt$  in (2), may avoid, to a certain extent, instability, but only if  $\mathcal{G}(L)$  is known. If this is not the case, or if  $\mathcal{G}(L)$  has very sharp drops in the right interval  $(L^*, \infty)$ , then even Big Brother, with all its might, stands to pay a nonnegligible price. We conclude that the left interval  $[0, L^*)$  is stable whereas the right interval  $(L^*, \infty)$  is potentially unstable.

The foregoing analysis illustrates the basic feasibility of attaining the optimal operating point, but also the “inherent” difficulty of maintaining it, even with near-perfect information, should the load-service function be unimodal. In the following sections, we will replace Big Brother by homogeneous, ignorant, but friendly group of “little folk,” and show that the resulting distributed protocol which does not have access to global information such as  $L^*$ , and which cannot respond instantaneously to changes, can nevertheless closely emulate Big Brother’s computational power.

### 3 Estimating utilization over a shared network

Let  $p_i$  and  $p_j$  be two nodes communicating over a network channel via packet switching or circuit switching. Let  $l_i$  and  $l_j$  be the local clocks of  $p_i$  and  $p_j$ , respectively. We allow  $l_i$  and  $l_j$  to run at different clock rates, but we require that the velocities be constant relative to an absolute time frame. In other words, there is a constant  $c_{ij}$  such that an event that has lasted for  $\Delta t_i$  in  $l_i$  and  $\Delta t_j$  in  $l_j$  will be related by  $\Delta t_i = c_{ij} \Delta t_j$ . If every node has access to a common global clock,  $c_{ij} = 1, \forall i, j$ . Let  $m_{ij}$  denote a packet addressed from  $p_j$  to  $p_i$ .  $m_{ij}$ , for our purposes, will contain three pieces of information. First, the actual data. Second, the source address  $i$  and destination address  $j$ . Third, the local time stamp of  $l_j$  at the time the packet was sent out at  $I_j$ ,  $p_j$ ’s network interface unit. Thus the message encoding part of the message generation protocol at  $I_j$  for  $m_{ij}$  is as follows:

#### Message encoding protocol(MEP):

1. Create a header containing  $i, j$ , and time stamped by its local clock  $l_j$ . (I.e.,  $m_{ij}.time\_stamp := l_j$ .)
2. Attach the header to the data section of  $m_{ij}$  and submit to network  $N$ .

Of course, somewhere in the process there must be a routing protocol, but from our perspective, it will be transparent. The message decoding part of the message reception protocol, now at  $I_i$  receiving  $m_{ij}$ , will update a data structure maintained for node  $p_j$  called  $hist_j$ .  $hist_j$  is a record containing three main components  $hist_j.last\_in$ ,  $hist_j.last\_out$ , and  $hist_j.warp$ . Into  $hist_j.last\_in$ , the local time at  $I_i$ ,  $l_i$ , when  $m_{ij}$  arrived is recorded. Into  $hist_j.last\_out$ , the local time at  $p_j$  when  $m_{ij}$  was sent out,  $m_{ij}.time\_stamp := l_j$ , gets recorded. The decoding part of the message reception protocol at  $I_i$  on receiving  $m_{ij}$  works as follows:

#### Message decoding protocol(MDP):

1.  $warp := \frac{1}{c_{ij}}(l_i - hist_j.last\_in) / (m_{ij}.time\_stamp - hist_j.last\_out)$ .
2.  $hist_j.last\_in := l_i$ .
3.  $hist_j.last\_out := m_{ij}.time\_stamp$ .

With the basic protocol out of the way, let us see what

properties the quantity  $warp$  has under the assumption that network  $N$  is a single-server, multiple-client, FCFS queue.

Let  $\mu(t)$  be a stochastic process given by  $\mu(t) = \bar{\mu} + \omega(t)$  where  $\bar{\mu}$  is a constant mean service rate and  $\omega(t)$  is the white noise process. The state of  $N$  at time  $t$ ,  $N(t)$ , is completely specified by  $L(t)$ ,  $\mu(t)$ ,  $\{\lambda_1(t), \dots, \lambda_n(t)\}$ , and the packets in the queue. Let  $p_j$  send two messages,  $m_{ij}^1$  and  $m_{ij}^2$ , to  $p_i$ , one at global time  $T_1$ , and the other at global time  $T_2$  ( $T_2 > T_1$ ). Let  $t_j$  be the local time of  $p_j$  at which this event occurs relative to the global time, and let  $t_i$  be the local time of  $p_i$  at which the same event occurs. Thus we have lined up the two local time frames  $l_i, l_j$  for the first event with respect to the global reference frame. Let  $t_j + \Delta t_j$  be the local time of  $p_j$  corresponding to  $T_2$ . For the subsequent treatment, it suffices to express all time-dependent quantities in terms of  $p_j$ ’s reference frame.

Assume that  $\lambda(t) = \sum_{k=1}^n \lambda_k(t)$  is a very slowly varying function of time. In fact, in the stability analysis phase after specifying the protocol, we will see that our choice of parameters will dictate this to be so. The average service rate  $E(\mu(\Delta t))$  over a time interval  $\Delta t$  is just the Brownian motion of the underlying white noise process, hence  $E(\mu(\Delta t)) = \bar{\mu}$  and its variance is proportional to  $\Delta t$ . If the initial load  $L(t_j)$  at time  $t_j$  is not too large, then  $d_1 = L(t_j)/\bar{\mu}$  is a good approximation for the time needed for  $m_{ij}^1$  to exit from the queue, i.e. get delivered to  $p_i$ . Thus the time stamp on  $m_{ij}^1$  will be  $t_j$ , and the local time at  $p_i$  when the packet arrives will be  $a_1 = t_i + c_{ij}d_1$ . What remains is to estimate the corresponding arrival time at  $p_i$  of the second packet in order to explicate  $warp$ . Let us compute the load at  $t_j + \Delta t_j$ , the time at which  $m_{ij}^2$  gets submitted to the queue. By the conservation equality (1), the load is given by

$$\begin{aligned} L(t_j + \Delta t_j) &= L(t_j) + \int_{t_j}^{t_j + \Delta t_j} (\lambda(t) - \bar{\mu}) dt \\ &\approx L(t_j) + \Delta t_j (\lambda(t_j) - \bar{\mu}). \end{aligned} \quad (6)$$

The approximation stems from the fact that we have assumed  $\lambda(t_j) \approx \lambda(t_j + \Delta t_j)$ . Using equation (6), the time needed for  $m_{ij}^2$  to exit from the queue can be estimated as  $d_2 = L(t_j + \Delta t_j)/\bar{\mu}$ . Hence, the local time of  $p_i$  at which  $m_{ij}^2$  arrives is given by

$$\begin{aligned} a_2 &= t_i + c_{ij}(\Delta t_j + d_2) \\ &= t_i + c_{ij}(\Delta t_j + \frac{L(t_j) + \Delta t_j(\lambda(t_j) - \bar{\mu})}{\bar{\mu}}) \\ &= t_i + c_{ij}(\Delta t_j + d_1 + \Delta t_j(\frac{\lambda(t_j)}{\bar{\mu}} - 1)). \end{aligned}$$

Finally, combining the previous results, the  $warp$  com-

puted at  $p_i$  by MDP upon arrival of  $m_{ij}^2$  is

$$\begin{aligned}
warp &= \frac{1}{c_{ij}} \frac{l_i - hist_j.last\_in}{m_{ij}.time\_stamp - hist_j.last\_out} \\
&= \frac{1}{c_{ij}} \frac{a_2 - a_1}{(t_j + \Delta t_j) - t_j} \\
&= \frac{1}{c_{ij} \Delta t_j} [(t_i + c_{ij}(\Delta t_j + d_1 + \Delta t_j(\frac{\lambda(t_j)}{\bar{\mu}} - 1))) \\
&\quad - (t_i + c_{ij}d_1)] \\
&= \frac{\lambda(t_j)}{\bar{\mu}} \approx \rho. \tag{7}
\end{aligned}$$

Equation (7) shows that *warp*, in the case of FCFS single-server queues with the service rate perturbed by white noise, is an estimator of the utilization of the queue. A pictorial representation of communication over a “warped” medium is shown in figure 2. *warp* has a time-lag associate with it, and its accuracy depends on several factors. First, the total delay,  $\Delta t_j + d_2$  must be not too large. Since  $\Delta t_j$  is the inverse of the message generation rate, this implies that packets carrying the control information cannot be too far in between. Second,  $\lambda(t)$  needs to be slowly time-varying for the estimates to be accurate. But  $\lambda$  is the control variable of the system, and as we shall see in the next section, for the system to be stable,  $|d\lambda(t)/dt|$  needs to be very small. Third, the steady-state load cannot be too large since this would make the steady-state delay, as reflected in  $d_1$ , become large as well. The use of time stamps to estimate net-

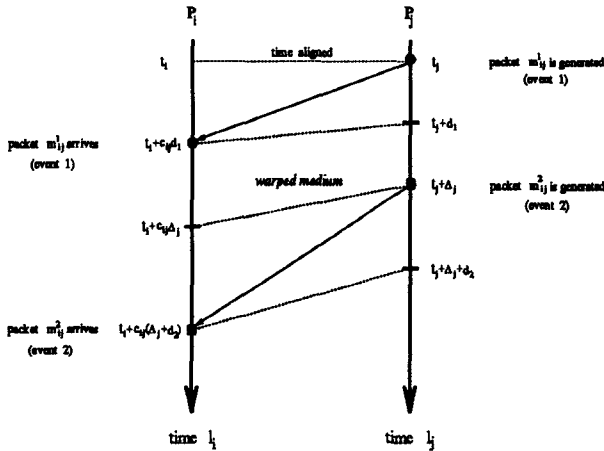


Figure 2: Communicating over a warped medium.

work delay is a common technique and other treatments can be found in [7, 13]. In the above analysis, if we let  $N$  be an arbitrary network, each pairwise interaction between two nodes  $p_i$  and  $p_j$  can be viewed as occurring over a private channel  $Q_{ij}$ , e.g. a set of virtual circuits linking  $p_i$  and  $p_j$ . In general, two such channels  $Q_{ij}$  and  $Q_{ki}$  will not be independent, and modeling their cross-coupling in a tractable way without committing

to specific network topologies may yield an interesting theory. In the next section, we present the case when all nodes share the same FCFS queue. A completely trivial extension is the case when there are  $n^2 - n$  queues  $Q_{ij}$ ,  $i \neq j$ , where all are pairwise independent.

## 4 The warp protocol

### 4.1 Protocol for rate matching

The previous section has shown that two agents interacting over a shared medium can estimate a global quantity, in our case, the network utilization  $\rho$ , making use of only “locally” available information. Having established local observability of a global quantity, the next step is to show the local controllability of this global quantity. The main goal of the first control protocol is to achieve *rate matching* between  $\lambda(t)$  and  $\mu(t)$  where  $\mu(t)$  is an unknown quantity. That is,  $E_t(\rho) = 1$ , and  $E_t((\rho - 1)^2) < \delta$ , for  $\delta$  small. Consider the case where  $\mu(t) = \bar{\mu} + \omega(t)$ , and  $\bar{\mu}$  is fixed. By equation (7) of the previous section, *warp* is an estimator of network utilization  $\rho(t)$ . This suggests using the following control law:

$$\frac{d\lambda(t)}{dt} = \epsilon(1 - warp) \approx \epsilon(1 - \frac{\lambda(t - \tau)}{\bar{\mu}}) \tag{8}$$

where  $\tau$  is a time-lag introduced by network delay. Note that we have suppressed the noise term  $\omega(t)$  which would have occurred in the denominator of the right-hand-side of (8). This is technically problematic. Even for a simpler equation,  $dx(t) = x(t - \tau)dW(t)$ , where the noise term  $W(t)$  (1-D Brownian motion) occurs multiplicatively, a closed-form solution does not seem to be known [16]. For our purposes, it suffices to extract features that describe the qualitative behavior of the system (mainly its stability), and we shall proceed with this in mind. When considering the deterministic delay-differential equation (8), it suffices to look at the homogeneous equation

$$\frac{d\lambda(t)}{dt} = -\frac{\epsilon}{\bar{\mu}}\lambda(t - \tau) \tag{9}$$

whose solution can be expressed in exponential form  $e^{\gamma t}$ . This involves finding the complex roots of the characteristic equation  $\gamma + \frac{\epsilon}{\bar{\mu}}e^{-\tau\gamma} = 0$ . The solution is a sinusoidal oscillation, and it is asymptotically stable if for all roots of the characteristic equation, the real parts lie in the negative half plane. For  $\tau = 1$ , it can be shown [8] that this is the case when  $0 < \frac{\epsilon}{\bar{\mu}} < \pi/2$ . To see the effect of any fixed  $\tau > 0$ , we will transform (9) into its “normal” form (i.e.  $\tau = 1$ ) by a change of variables. Let  $t' = t/\tau$ . Then,  $d\lambda(\tau t')/dt' = (\tau\epsilon/\bar{\mu})\lambda(\tau(t' - 1))$ . Setting  $\tilde{\lambda}(t') = \lambda(\tau t')$ , we get the normalized equation

$$\frac{d\tilde{\lambda}(t)}{dt} = -\tau\frac{\epsilon}{\bar{\mu}}\tilde{\lambda}(t - 1) \tag{10}$$

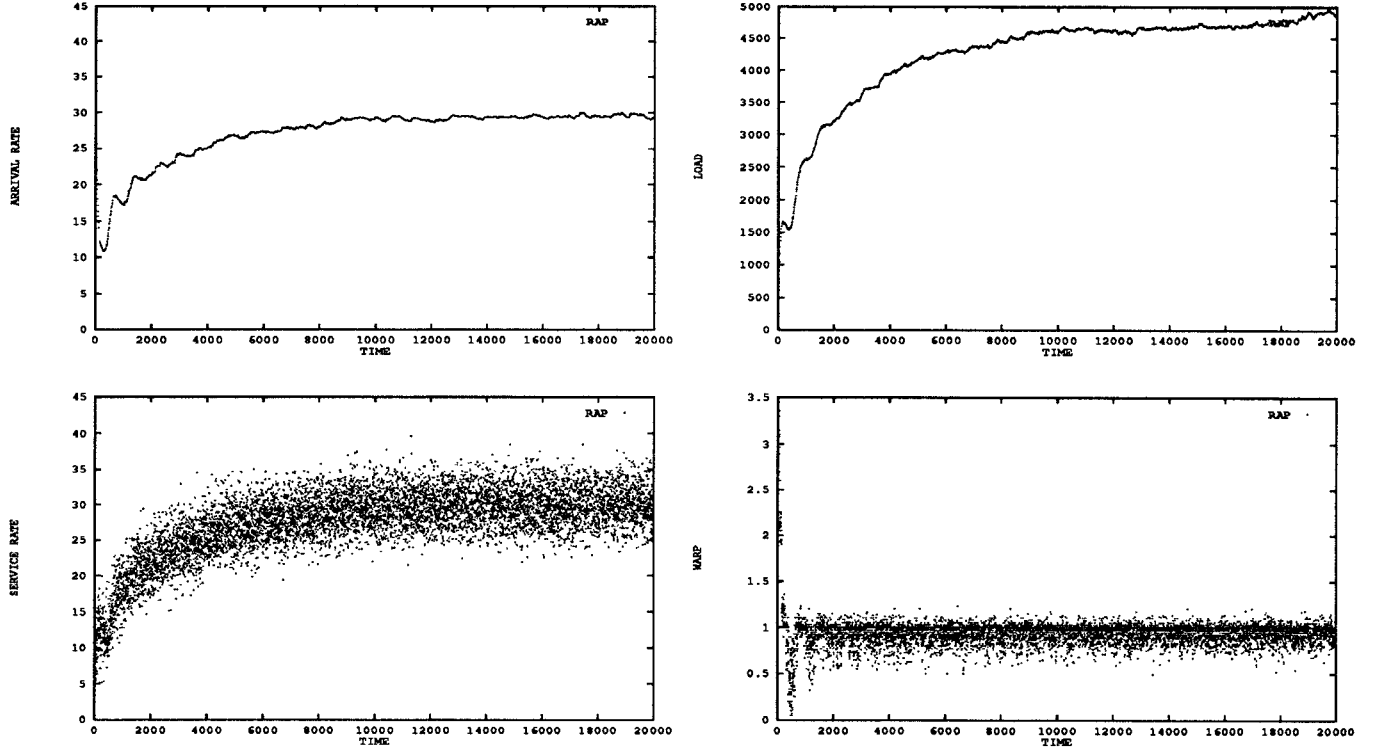


Figure 3: Simulation of a 10 node system using a unimodal service-load function running RAP.

where we have suppressed the *tilde* for notational clarity. Finally, we can state the condition for asymptotic stability of the deterministic delay differential equation as

$$0 < \tau \frac{\epsilon}{\mu} < \pi/2. \quad (11)$$

Thus the greater the network delay, the smaller  $\epsilon$  needs to be in order to keep the system stable. This agrees well with our previous requirement for making  $\lambda(t)$  a slowly varying function of time. The effect of the noise term for  $\mu$  stochastic can be gleaned from (11). If the variance of the Gaussian noise is large, then for  $\epsilon$  not sufficiently small, the stability condition may be frequently violated leading to growing fluctuation.

*Distributed control* is made possible via  $\lambda(t) = \sum_{k=1}^n \lambda_k(t)$ , where at each node, the control part of the message reception protocol executes the same control function given in (8). This we call the *rate adjustment protocol* (RAP). At each  $p_i$ :

**Rate adjustment protocol(RAP):**

1.  $\lambda_i(t) = \lambda_i(t-1) + \epsilon_i(1 - \text{warp}(t))$ .
2. If  $\lambda_i(t) < 0$ , then  $\lambda_i(t) = 0$ .

There are several points worth noting here. First, ideally,  $\lambda_i(t) = \lambda(t)/n$ ,  $i = 1, 2, \dots, n$ . As  $t \rightarrow \infty$ ,  $E_i(\lambda_i(t)) = \lambda(t)/n$ , but  $E_i((\lambda_i(t) - \lambda(t)/n)^2)$  will drift in the absence of some preventive mechanism (the expectation is taken over  $i$ ). One consequence is the breakdown

of *fairness*. By fairness, we mean the equal allocation of message generation rates over all constituents. Second, note that only one control variable,  $\lambda_i$ , is kept at each node. As we have indicated in section 3, a more general scheme is to split up  $\lambda_i$  at each  $p_i$  such that  $\lambda_i = \sum_{k \neq i} \lambda_{ik}$ . This allows for individualized communication rate control over different pairs of nodes, and is useful for the case where the interconnection network is not “uniform.” For example, node  $p_i$  may be experiencing congestion build-up with respect to its communication channel to  $p_j$ , but not to  $p_k$ . Hence the traffic to  $p_j$  should be controlled drastically whereas the communication rate to  $p_k$  may not need to change. A typical simulation run of a 10 node system using an actual unimodal load-service function (note, the previous analysis was for  $\bar{\mu}$  fixed) is shown in figure 3. The optimal load interval was set at [4700, 5000], and  $\mathcal{G}(L)$  was a straight line with positive and negative slopes on the left and right segments, respectively. The optimal service rate was 30 packets per unit time, and the service rate was subject to a white noise process with variance 9. The graphs show the stable rate matching properties of the protocol, even under nonnegligible noise conditions.

**4.2 Protocol for load matching**

In the previous section we have shown that RAP is able to maintain full utilization for the case when  $\bar{\mu}$  is

fixed. In this section, we will investigate the general case where the system is characterized by unimodal load-service functions. Basically, we will show that there is a stable region and an unstable region, and optimal utilization cannot be maintained in the long run by the rate-matching protocol alone. Consider our system of equations (1) and (8), and simplify it by ignoring the noise term  $\omega(t)$  as well as letting  $\lambda(t) = \lambda(t - \tau)$ . The latter is partly justified by our requirement for setting  $\epsilon$  very small. We get a system of nonlinear differential equations coupled via  $\mu = \mathcal{G}(L)$ ,

$$\begin{bmatrix} d\lambda/dt \\ dL/dt \end{bmatrix} = \begin{bmatrix} \epsilon - \epsilon\lambda(t)/\mathcal{G}(L) \\ \lambda(t) - \mathcal{G}(L) \end{bmatrix}. \quad (12)$$

Let us denote it by  $(\lambda, L) = F(\lambda, L)$ . To understand its dynamics, we will first linearize it, then analyze the stability of the linearized system. The gradient in the first-order term is given by

$$\frac{\partial F(\lambda, L)}{\partial(\lambda, L)} = \begin{bmatrix} -\epsilon/\mathcal{G}(L) & \epsilon\lambda(t)\mathcal{G}_L(L)/\mathcal{G}(L)^2 \\ 1 & -\mathcal{G}_L(L) \end{bmatrix}$$

where  $\mathcal{G}_L(L) \equiv d\mathcal{G}(L)/dL$ . Note, all  $(\lambda, L)$  such that  $\lambda = \mathcal{G}(L)$  are solutions to (12). Denote this set by  $\mathcal{S}$ . For any  $(\bar{\lambda}, \bar{L}) \in \mathcal{S}$ , the Taylor expansion around  $(\bar{\lambda}, \bar{L})$  is given by

$$\begin{bmatrix} d\lambda/dt \\ dL/dt \end{bmatrix} = \begin{bmatrix} -\epsilon/\mathcal{G}(\bar{L}) & \epsilon\mathcal{G}_L(\bar{L})/\mathcal{G}(\bar{L}) \\ 1 & -\mathcal{G}_L(\bar{L}) \end{bmatrix} \begin{bmatrix} \lambda - \bar{\lambda} \\ L - \bar{L} \end{bmatrix}$$

where we have dropped the higher-order terms, and  $F(\bar{\lambda}, \bar{L}) = 0$ . To determine stability, we need to solve for the eigenvalues of  $\partial F(\lambda, L)/\partial(\lambda, L)$  at  $(\bar{\lambda}, \bar{L})$ . This yields one eigenvalue of  $r = 0$  (the matrix is singular), and another given by

$$r = -\epsilon/\mathcal{G}(\bar{L}) - \mathcal{G}_L(\bar{L}). \quad (13)$$

Clearly, for  $\bar{L} < L^*$ ,  $\mathcal{G}_L(\bar{L}) > 0$ , hence  $r < 0$ . That is, the system is stable. The same holds for  $\bar{L} = L^*$ . For  $\bar{L} > L^*$ ,  $\mathcal{G}_L(\bar{L}) < 0$ , and depending on the magnitude of  $\mathcal{G}_L(\bar{L})$ , it is possible that  $r > 0$ , leading to instability. This is analogous to the situation that Big Brother encountered when faced with a unimodal load-service function (section 2.2).

Two issues need to be dealt with at this point. First, since all points  $(\bar{\lambda}, \bar{L}) \in \mathcal{S}$  with  $\bar{L} \leq L^*$  are stable solutions, if the system starts off in a state where  $L < L^*$ ,  $L$  will settle into an equilibrium far below  $L^*$ . An extra thrust, or bias, needs to be applied to give the process an upward drift. Consider the modified rate adjustment control

$$\frac{d\lambda}{dt} = \epsilon(1 + \theta(t) - \text{warp}) \quad (14)$$

where  $\theta(t)$  is a variable that controls drift. Letting  $0 < \theta(0) = \theta_a \ll 1$  has the effect of shifting the equilibrium of the control equation to  $\lambda = \mu + \theta(t)\mu$ , which in

turn will impart an upward motion on  $L$  via (1). The second issue is, in part, caused by addressing the first issue using the bias parameter  $\theta(t)$ . That is, although  $\theta(t)$  allows the system to climb up the continuous “equilibrium curve”  $\mathcal{G}(L)$  toward  $L^*$ , unless the process is terminated at some point near  $L^*$ , instability will ensue. To avoid “crossing over the line,” one, it needs to be detected that  $L(t) \approx L^*$ , and two,  $\theta(t)$  must be turned off. If we assume that for most networks the left neighborhood of  $L^*$  is relatively flat, then we can exploit this property by noting that for small  $\theta(t)$ , the history of  $\lambda(t)$  will reflect the slope of  $\mathcal{G}(L)$ . This leads to the first solution method which is of a “proactive” [7] nature. In our case, regression analysis is used to estimate the slope of the message generation rate history. If flatness is detected,  $\theta(t)$  is shut off. More sophisticated controls can be designed by making  $\theta(t)$  be a continuous function of the estimated slope. Let  $\text{rate\_hist}_i$  be an array of elements of past  $\lambda$  values at  $p_i$ .

#### Proactive rate protocol (PRP):

1.  $\text{slope} = \text{regress}(\text{rate\_hist}_i)$ .
2. If  $|\text{slope}| < \text{slope\_threshold}$ , then  $\theta(t) = 0$ .
3. If  $t \equiv 0 \pmod{\text{slope\_interval}}$ , then do
  - Discard oldest element of  $\text{rate\_hist}_i$ .
  - Record  $\lambda_i(t)$  into  $\text{rate\_hist}_i$ .

$\text{Regress}()$  is a function that performs linear regression on  $\text{rate\_hist}_i$ ,  $\text{slope\_threshold}$  is a parameter that tests whether the slope is 0 or not, and  $\text{slope\_interval}$  represents the time interval between successive stored values since keeping a large history is expensive. The longer the optimum load interval  $[L_l^*, L_u^*]$ , the more effective the proactive scheme will be since there is more room to detect the “plateau.”

An altogether different approach is the reactive scheme, which upon detection of instability, performs an “exponential back-off” type operation. For a sudden drop in service rate due to congestion or structural effects, in the absence of a rapid recovery mechanism the system will crash ( $\mu \approx 0$ ) as seen in the previous analysis. All “permanent” drops in the service rate are accompanied by a delayed, but sharp drop in the message generation rate. This can be seen in figure 4 for a 10 node system which has an optimum service rate of 60 packets per unit time. The system was simulated running only RAP. To minimize the disruptive effects of a total crash, a reactive protocol is constructed that detects a sharp increase in  $\text{warp}$ , then goes into sleep for  $\text{lock}$  number of steps after setting the message generation rate to a lower value. At each  $p_i$ ,

#### Reactive rate protocol (RRP):

1. If  $\text{warp}_i > 1 + \text{warp\_threshold}$  then do
  - Set  $\text{new } \lambda_i = \text{old } \lambda_i * (2 - \text{warp}_i)$ .
  - Set  $\text{lock}_i = \text{lock\_constant} * (\text{warp}_i - 1)$ .
2. Go into sleep for  $\text{lock}_i$  steps.

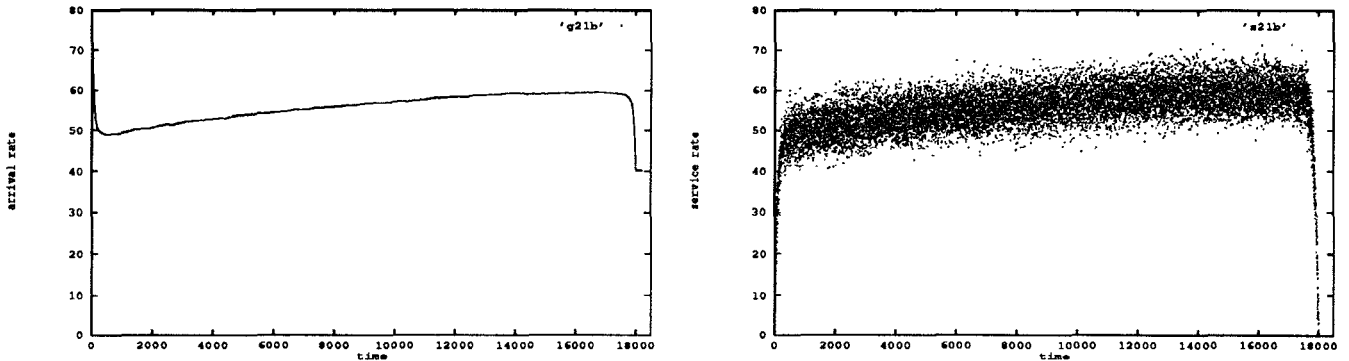


Figure 4: Simulation of a 10 node system and its crash in the unstable region.

*Warp\_threshold* is a parameter for detecting a critical change in *warp*, and *lock\_constant* determines the sleep period. It may very well be possible to incorporate the exponential back-off scheme in the control equation itself. Figure 5 shows a 10 node system with optimum service rate of 30 and optimal load interval [3700, 4000]. *Warp\_threshold* was set at 0.3 and *lock\_constant* was 60. The system was run for 30000 time steps under the RAP + RRP protocol. Note, the warp and arrival rates shown are averaged over all nodes. Each node executes its protocol independently, and it is not the case that all nodes get a *warp* value exceeding 1.3 all at the same time. If the system is at a critical level, most nodes will make a similar decision within a short interval of each other. Even if the system is not critical, a processor may decide to “hibernate” due to noise. Further improvements can be made by combining RAP, PRP, and RRP, but due to space limitations this will be omitted.

## 5 Simulation and extensions

In the following sections, we will show simulations of typical system behavior under different system configurations. All simulations were run under the following procedure: at every global time step  $t$ , dequeue  $\mathcal{G}(L(t)) + \omega(t)$  messages and deliver them to their destinations. Each receiving node executes its message reception protocol which consists of MDP, RAP, and other applicable control protocols. For each “able” (e.g. not locked) node  $p_i$ , generate  $\lambda_i(t)$  messages with destination addresses formed uniformly. This entails executing MEP, and enqueueing messages into the queue. Increment global time, and repeat. Various details have been omitted for brevity, but essentially, it is just a straightforward simulation of a multi-node system sharing a common queue. It should be mentioned that more than one message is allowed to be generated per unit time, and  $\lambda$  is kept as a real number with the fractional part being transferred over to the next round.

The top part of figure 6 shows the behavior of a

10 node system with optimum service rate 30, variance 9, running only RAP, under three different load-service functions given by [4700, 5000], [3700, 4000], and [2700, 3000], respectively. The results demonstrate the stable rate matching properties of RAP as well as convergence to the optimal load interval which agrees well with our previous analysis. All three runs were performed with the same parameters, partially showing the robustness of the protocol. Simulation runs for a 30, 60, and 90 node system is shown in the bottom part of figure 6. They all shared a common optimal load interval [5000, 6000] and an optimal service rate of 30.

### 5.1 Structural stability

The previous analysis was for the case when the system was fixed, i.e. constant number of nodes, fixed variance, and invariant load-service function. Structural variability was to some extent incorporated by the variance of the service rate, but clearly this is not enough. Figure 7 (left) shows the evolution of a 10 node system with an optimal load at 2000, and optimal service rate 30, which is structurally perturbed at time 10000 by taking away 20% of its nodes, in our case 2. There is an immediate drop in the arrival rate due to the loss, which is stably, but gradually made up. This is due to the small rate adjustment constant  $\epsilon$ . Is it possible to have a speedier recovery? The gap in the message generation rate suggests using  $\lambda$  as an indicator of sudden underutilization. There is a more uniform approach based on *warp* which in this case works opposite to what was done in RRP. Instead of watching for a sharp increase in *warp*, each node upon seeing a sharp decrease will give its generation rate a temporary boost, thereby quickening the response.

#### Boost rate protocol (BRP):

1. If  $warp_i < 1 - warp\_threshold$  then do
  - Set  $new\ \lambda_i = old\ \lambda_i * (1 + warp)$ .
  - Set  $lock_i = lock\_constant * (1 - warp_i)$ .
2. Turn off BRP and RRP for  $lock_i$  steps.

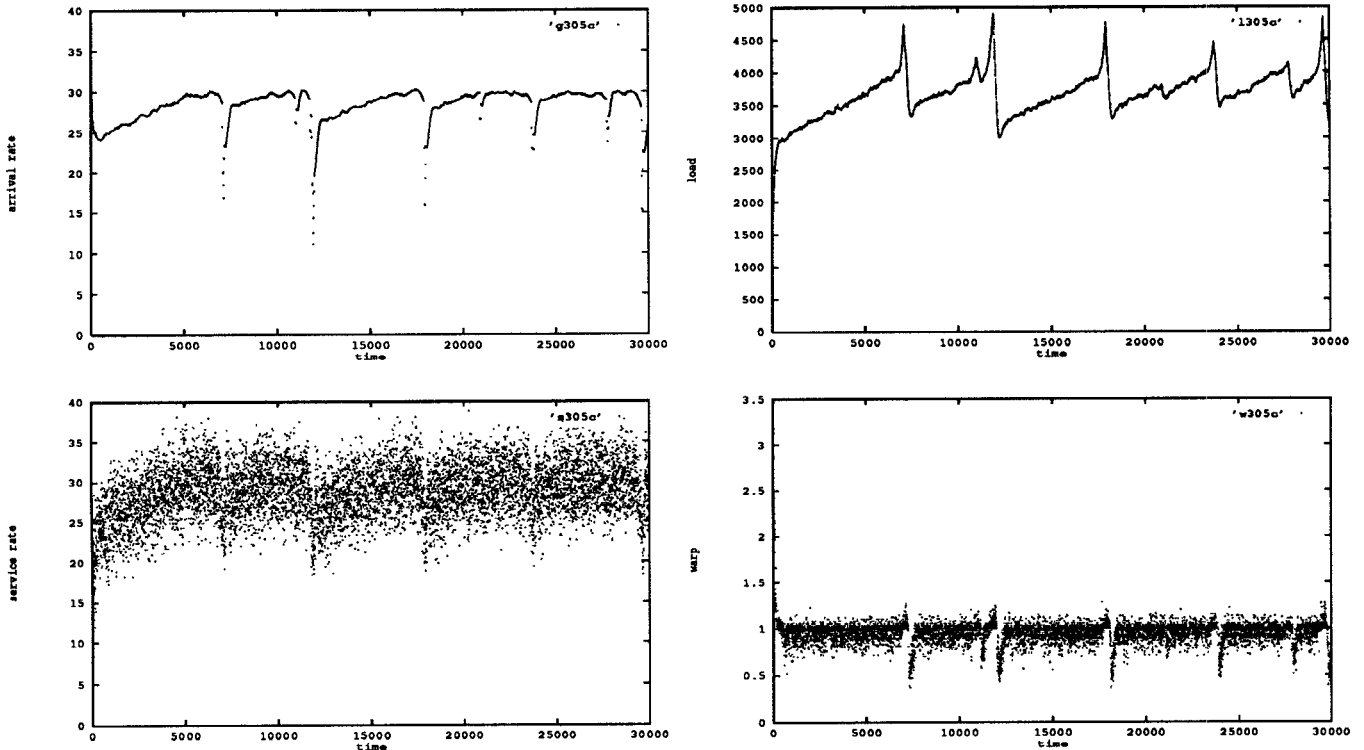


Figure 5: Simulation of a 10 node system running RAP and RRP.

Figure 7 (right) shows the effect of running RAP in conjunction with BRP for a 10 node system where 20% node perturbation is applied at time 10000. The arrival rate again experiences a sharp drop, but in contrast to figure 7 (left), the arrival rate immediately jumps back and settles to a rate close to its pre-perturbation level. It is important to shut down BRP and RRP for a short moment following the boost in order to avoid positive feedback. Structural perturbations involving the service rate are dealt with in a similar manner using RRP.

## 5.2 Fairness

Due to accumulation of random noise and increase in variance, it will not be the case that  $E_i((\lambda_i(t) - \lambda(t)/n)^2) \approx 0$  as  $t \rightarrow \infty$ . Figure 8 (left) shows the deviation in bandwidth allocation for a 10 node system with optimal service rate 30. The time evolution of the arrival rates for 5 nodes are shown. The drift is clearly visible. At time 12000, the RRP protocol performs a recovery routine, and two nodes “luck out” by performing a back-off operation that is costlier than the others, further increasing the variance. Figure 8 (right) shows a simulation run of the previous system under the exact same conditions except that a fairness protocol was active. The fairness protocol works by including  $\lambda$  in the header portion at regular intervals, which then at the receiving end are used to update the receiver’s mes-

sage generation rate. The scheme used in the above simulation is to replace the receiver’s  $\lambda$  value with the average of its sampled neighbors’, every 40 time units. There exist many variations on the same theme, but due to space constraints, the analysis part will be omitted. Suffice it to say that this is an important topic by itself which deserves separate attention.

## 6 Conclusion

A characterization of a communication network modeled as a stochastic shared single-server queue was presented, and its global dynamics analyzed with respect to stability, optimality, and fairness. Based upon the notion of an *inherent* load-service function and its unimodality assumption, stable and unstable regions in phase space were identified, and protocols were proposed for maintaining optimal utilization. The primary rate matching protocol was facilitated by a quantity called *warp* which was shown to estimate network utilization. The most interesting aspect of the rate adjustment protocol is its tendency to converge and “linger” at the optimal operating point, which then can be exploited to achieve load matching by not stepping over into the unstable region. Simulation results were presented supporting the model’s analysis, but due to space limitations, many topics were given insufficient treatment. We hope to remedy this situation in the full

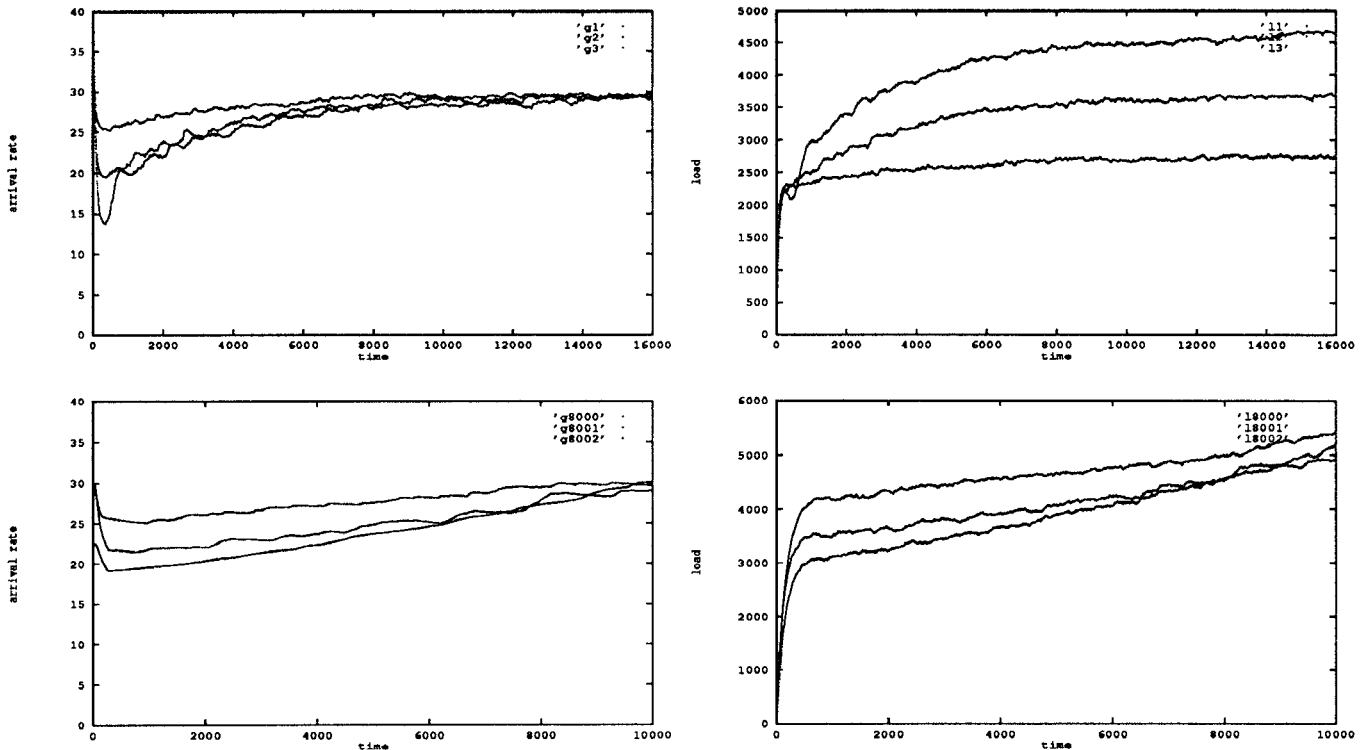


Figure 6: Top: 10 node system under three different network conditions. Bottom: 30, 60, and 90 node system under same network conditions.

paper.

### Acknowledgements

The author would like to thank Drs. Abdelsalam Heddaya, Eugene Pinsky, and Himanshu Sinha for many helpful discussions. The author can be contacted at park@cs.bu.edu.

### References

- [1] Ludwig Arnold. *Stochastic Differential Equations: Theory and Applications*. John Wiley & Sons, 1974.
- [2] K. Bala, I. Cidon, and K. Sohraby. Congestion control for high speed packet switched networks. In *Proc. IEEE INFOCOM '90*, pages 520–526, 1990.
- [3] Jean-Chrysostome Bolot and A. Udaya Shankar. Analysis of a fluid approximation to flow control dynamics. In *Proc. IEEE INFOCOM '92*, pages 2398–2407, 1992.
- [4] R. Dighe, C. J. May, and G. Ramamurthy. Congestion avoidance strategies in broadband packet networks. In *Proc. IEEE INFOCOM '91*, pages 295–303, 1991.
- [5] K. Fendick, M. Rodrigues, and A. Weiss. Analysis of a rate-based control strategy with delayed feedback. In *Proc. ACM SIGCOMM '92*, pages 136–148, 1992.
- [6] M. Gerla and L. Kleinrock. Flow control: a comparative survey. *IEEE Trans. Commun.*, COM-28:553–574, 1980.
- [7] Z. Haas and J. Winters. Congestion control by adaptive admission. In *Proc. IEEE INFOCOM '91*, pages 560–569, 1991.
- [8] Jack K. Hale. *Functional Differential Equations*. Springer-Verlag, 1971.
- [9] A. Heddaya, K. Park, and H. Sinha. Using warp to control network contention in mermera. *Submitted to HICSS '94*, 1993.
- [10] A. Hordijk and F. Spieksma. Constrained admission control to a queueing system. *Adv. Appl. Prob.*, 21:409–431, 1989.
- [11] Van Jacobson. Congestion avoidance and control. In *Proc. ACM SIGCOMM '88*, pages 314–329, 1988.

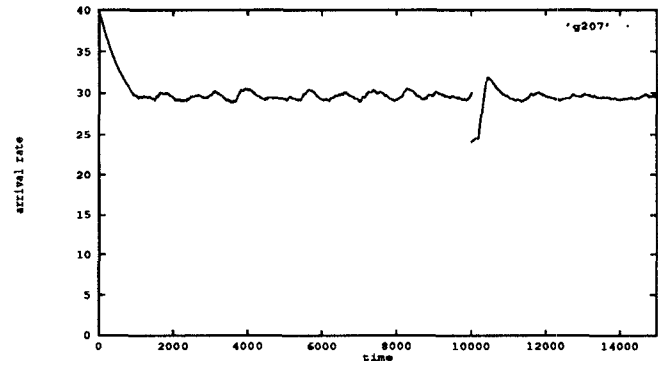
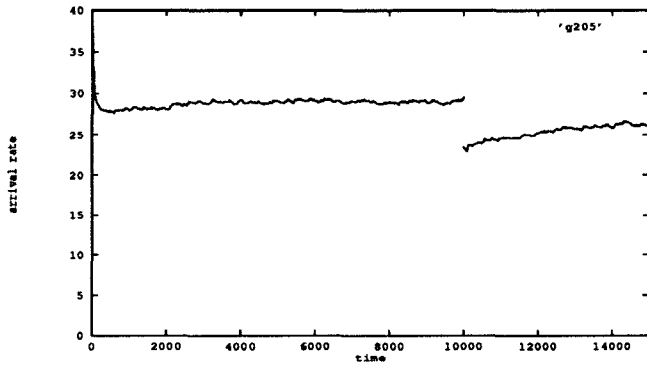


Figure 7: Simulation of a 10 node system with structural (node number) perturbation. Left: slow recovery without BRP. Right: responsive recovery with BRP.

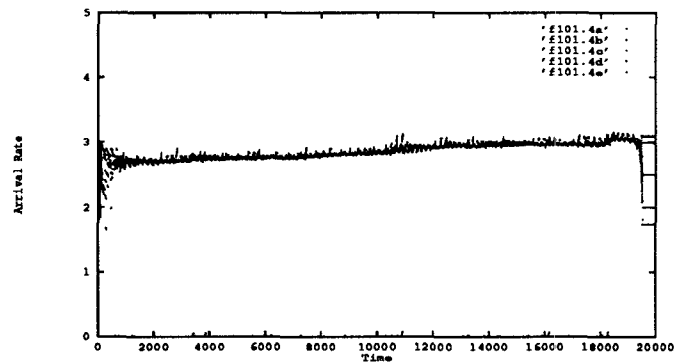
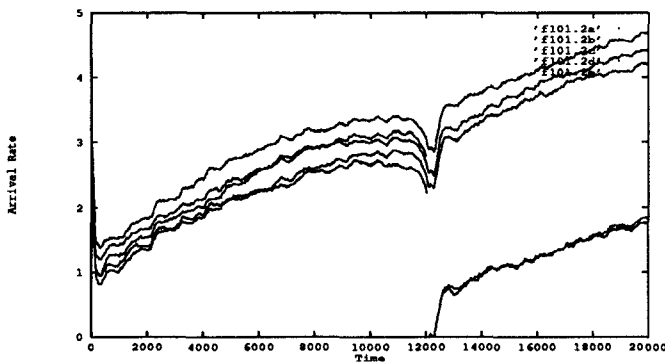


Figure 8: Simulation of a 30 node system with (right), and without (left), fairness protocol.

- [12] F. P. Kelly. The optimization of queueing and loss networks. In O. J. Boxma and R. Syski, editors, *Queueing Theory and its Applications*. North-Holland, 1988.
- [13] S. Keshav. A control-theoretic approach to flow control. In *Proc. ACM SIGCOMM '91*, pages 3–15, 1991.
- [14] H. Kobayashi. Optimal routing in closed queueing networks. *ACM Trans. Comp. Sys.*, 1(4):294–310, 1983.
- [15] D. Mitra and J. Seery. Dynamic adaptive windows for high speed data networks: theory and simulations. In *Proc. ACM SIGCOMM '90*, pages 30–37, 1990.
- [16] S-E A. Mohammed. *Stochastic functional differential equations*, volume 99 of *Research Notes in Mathematics*. Pitman Advanced Publishing Program, 1984.
- [17] A. Mukherjee and J. Strikwerda. Analysis of dynamic congestion control protocols - a Fokker-Planck approximation. In *Proc. ACM SIGCOMM '91*, pages 159–169, 1991.
- [18] G. F. Newell. *Applications of Queueing Theory*. Chapman and Hall, 2nd edition, 1982.
- [19] J. Robinson, D. Friedman, and M. Steenstrup. Congestion control in *bbn* packet-switched networks. *Computer Communication Review*, 20(1):76–90, 1990.
- [20] Scott Shenker. A theoretical analysis of feedback flow control. In *Proc. ACM SIGCOMM '90*, pages 156–165, 1990.
- [21] Y. T. Wang and B. Sengupta. Performance analysis of a feedback congestion control policy under non-negligible propagation delay. In *Proc. ACM SIGCOMM '91*, pages 149–157, 1991.