The Synchronization of Periodic Routing Messages

Sally Floyd, *Member, IEEE,* and Van Jacobson

Abstract— **The paper considers a network with many apparently-independent periodic processes and discusses one method by which these processes can inadvertently become synchronized. In particular, we study the synchronization of periodic routing messages, and offer guidelines on how to avoid inadvertent synchronization. Using simulations and analysis, we study the process of synchronization and show that the transition from unsynchronized to synchronized traffic is not one of gradual degradation but is instead a very abrupt 'phase transition': in general, the addition of a single router will convert a completely unsynchronized traffic stream into a completely synchronized one. We show that synchronization can be avoided by the addition of randomization to the traffic sources and quantify how much randomization is necessary. In addition, we argue that the inadvertent synchronization of periodic processes is likely to become an increasing problem in computer networks.**

I. Introduction

A SUBSTANTIAL, and increasing, fraction of the traffic in today's computer networks comes from periodic traffic sources; examples include the periodic exchange of routing messages between gateways or the distribution of real-time audio or video. Network architects usually assume that since the sources of this periodic traffic are independent, the resulting traffic will be independent and uncorrelated. For example, even though each routing process might generate a packet at fixed, 30 second intervals, the total routing traffic observed at any point in the network should be smooth and uniform since the processes are on separate nodes and started with a random relative phase. However, many network traffic studies [22], [26], [15], [3] show that the total traffic is not uniform but instead is highly synchronized.

This paper argues that the architect's intuition that independent sources give rise to uncorrelated aggregate traffic is simply wrong and should be replaced by expectations more in line with observed reality. There is a huge body of research on the tendency of dynamic systems to synchronize in the presence of weak coupling [2]. As far back as the mid-seventeenth century, Huygens noticed that two unsynchronized pendulum clocks would keep in time if hung on the same wall, synchronized by the barely-perceptible vibrations each induced in the wall. As reported in [2], synchronization has been studied in electronic circuits, a wide range of mechanical objects, and biological systems such as cell populations and communities of fireflies. Most of these systems exhibit a tendency towards synchronization that is independent of the physical constants and initial conditions of the system [7]. This research suggests that a complex coupled system like a modern computer network evolves to a state of order and synchronization if left to itself. Where synchronization does harm, as in the case of highly correlated, bursty routing traffic, it is up to network and protocol designers to engineer out the order that nature tries to put in.

This paper investigates one means by which independent sources of periodic traffic can become synchronized. An analytic model is developed that shares many of the features observed in simulations and in real traffic measurements. There are two main results from this model:

- The transition from unsynchronized to synchronized behavior is very abrupt. The traffic does not gradually 'clump up' and become more synchronized as network parameters change. Instead, for each set of protocol parameters and implementation interaction strengths there exists a clearly defined transition threshold. If the number of sources is below the transition threshold, the traffic will almost certainly be unsynchronized and, even if synchronized by some external force,[1] it will unsynchronize over time. Conversely, if the number of sources is above the threshold, the traffic will almost certainly be synchronized and, even if placed in an unsynchronized state by some external force, will evolve to synchronization over time.
- The amount of randomness that must be injected to prevent synchronization is surprisingly large. For example, in the Xerox PARC internal network, measurements [6] show their cisco routers require roughly 300 ms. to process a routing message (1 ms. per route times 300 routes per update). From the results in Section 5, the routers would have to add at least a second of randomness to their update intervals to prevent synchronization.

There are many examples of unanticipated synchronized behavior in networks:

- **TCP window increase/decrease cycles.** A well-known example of unintended synchronization is the synchronization of the window increase/decrease cycles of separate TCP connections sharing a common bottleneck gateway [32]. This example illustrates that unless we *actively* engineer to avoid synchronization, such as by injecting randomness into the network, synchronization is likely to be the equilibrium state. As an example of injecting randomness, the synchronization of window increase/decrease cycles can be avoided by adding randomization to the gateway's algorithm for choosing packets to drop during periods of congestion [9]. (This randomization has the advantage of avoiding other unintended phase effects as well.)

[1] For example, by restarting all the routers at the same time because of a power failure.

Manuscript received January 1994; revised February 1994; approved by IEEE/ACM TRANSACTIONS ON NETWORKING Editor L. Zhang. This work was supported by the Director, Office of Energy Research, Scientific Computing Staff, of the U.S. Department of Energy under Contract DE-AC03-76SF00098.

The authors are with the Lawrence Berkeley Laboratory, Berkeley, CA 94720 USA (floyd@ee.lbl.gov, van@ee.lbl.gov).

IEEE Log Number 9402099.

• **Synchronization to an external clock.** Two processes can become synchronized with each other simply by both being synchronized to an external clock. For example, [22] shows DECnet traffic peaks on the hour and half-hour intervals; [23] shows peaks in ftp traffic as several users fetch the most recent weather map from Colorado every hour on the hour.

• **Client-server models.** Multiple clients can become synchronized as they wait for service from a busy or recovering server. For example, in the Sprite operating system clients check with the file server every 30 seconds; in an early version of the system, when the file server recovered after a failure or after a busy period, a number of clients would become synchronized in their recovery procedures. Because the recovery procedures involved synchronized timeouts, this synchronization resulted in a substantial delay in the recovery procedure [1].

• **Periodic routing messages.** Unlike the client/server model or the external clock model, the synchronization of periodic routing messages involves seemingly independent periodic processes. There are many routing protocols where each router transmits a routing message at periodic intervals. Assuming that the routers on a network are initially unsynchronized, at first glance it might seem that the periodic messages from the different routers would remain unsynchronized. This paper explores how initially unsynchronized routing messages can become synchronized.

We examine the details of router synchronization to give a concrete example of inadvertent synchronization, to underline the necessity of actively designing to avoid synchronization, and to emphasize the utility of injecting randomization as a method of breaking up synchronization. When a particular instance of synchronization is observed, it is usually easy to suggest protocol changes that could prevent it. This misses the point. Synchronization is not a small problem caused by minor oversights in protocol design. The tendency of weakly coupled systems to synchronize is quite strong and changing a deterministic protocol to correct one instance of synchronization is likely to make another appear.

Various forms of periodic traffic are becoming an increasingly-large component of Internet traffic. This periodic traffic includes not only routing updates and traffic resulting from the increasing use of periodic background scripts by individual users [22], but real time traffic (such as video traffic) that has a periodic structure. Although the periodic structure of video traffic is generally not affected by feedback from the network, there are still possibilities for synchronization. For example, individual variable-bit-rate video connections sharing a bottleneck gateway and transmitting the same number of frames per second could contribute to a larger periodic traffic pattern in the network. As periodic traffic increases in the Internet, it becomes increasingly important for network researchers to consider questions of network synchronization.

We use both simulation and analysis to explore the synchronization of periodic routing messages. The first goal of the analysis is to examine the role that random fluctuations in timing play in the synchronization of routing messages. These random fluctuations contribute to both the formation of synchronization and to the breaking up of synchronization after it occurs.

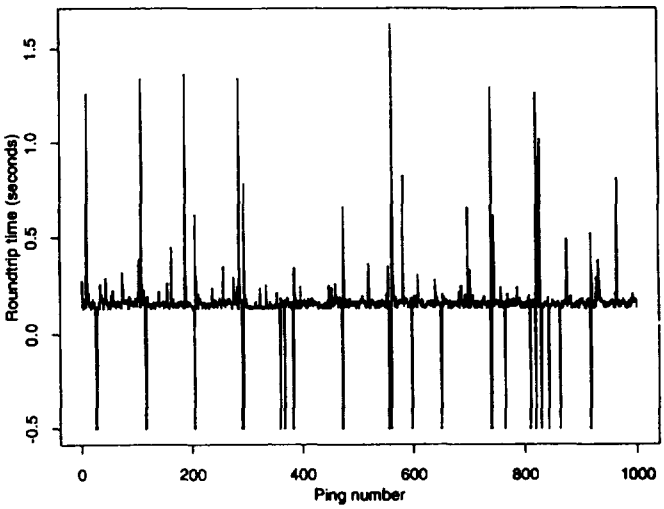


Fig. 1. Periodic packet losses from IGRP routing messages.

One way to break up synchronization is for each router to add a (sufficiently large) random component to the period between routing messages. A second goal of our analysis is to investigate this explicit addition of a random component to the routing timer, and to specify the magnitude of the random component necessary to prevent synchronization.

Section II gives examples of periodic traffic patterns in the Internet; Section III describes our model of periodic routing messages on a network. Section IV explains the results of our simulations. Section V describes a Markov chain used to analyze some aspects of the Periodic Messages model. Section VII presents conclusions and discusses alternatives for preventing routing message synchronization.

II. Periodic Traffic Patterns In the Internet

This section gives an example of synchronized routing messages, and several examples of periodic traffic patterns in the Internet (some of which are caused by periodic routing messages). While we do not have direct evidence of operational problems in the Internet related to synchronized routing messages, we show indirect evidence that such problems could exist. In general, there are significant patterns of periodic packet drops and delays in the Internet.

We began this investigation in 1988 after observing synchronized routing messages from DECnet's DNA Phase IV (the DIGITAL Network Architecture) [31] on a local Ethernet at LBL (Lawrence Berkeley Laboratory). Each DECnet router transmitted a routing message at 120 second intervals. Within hours after bringing up the routers on the network after a failure, the routing messages from the various routers were completely synchronized.

In May 1992, in the course of investigating packet loss rates in the Internet, we conducted experiments sending runs of a thousand pings each, at roughly one-second intervals, from Berkeley and other sites to destinations across the Internet. For all of the runs to destinations at Harvard or MIT, at least three percent of the ping packets were dropped, regardless of the time of day. Fig. 1 shows a particular run of a thousand

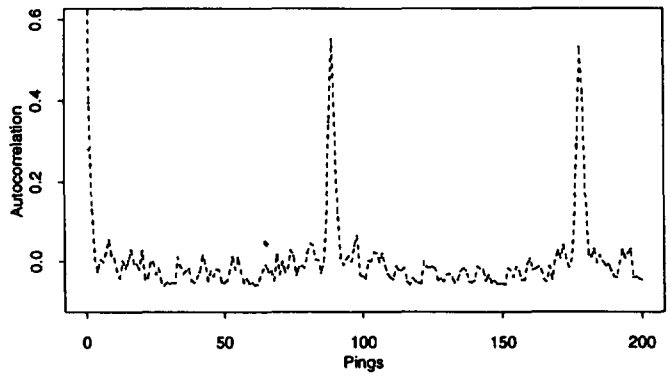


Fig. 2. The autocorrelation of roundtrip times.

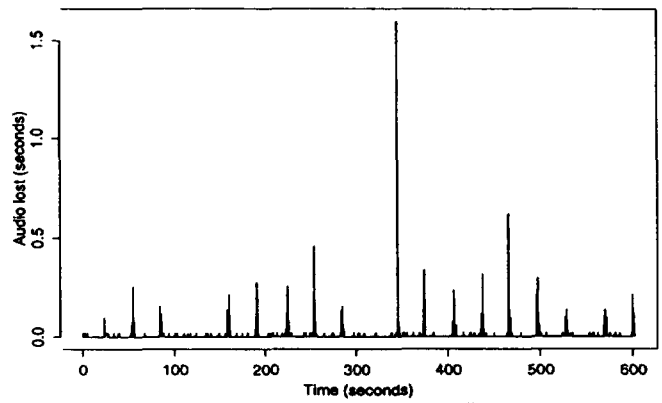


Fig. 3. Periodic packet losses at 30-second intervals.

pings from Berkeley to MIT; the x-axis shows the ping number and the y-axis shows the roundtrip time. Dropped packets are represented by a negative roundtrip time. Fig. 2 shows the autocorrelation function for the roundtrip times in Fig. 1, where the dropped packets are assigned a roundtrip time of two seconds (higher than the largest roundtrip time in the experiment). The pattern of periodic packet drops at 90-second intervals is illustrated in both figures. Further experiments determined that these packet drops were occurring at the NEARnet (New England Academic and Research Network) core routers. Earlier investigation of Internet behavior had also reported a degradation in service with a 90-second periodicity on paths to MIT [26].

These packet drops were determined to becaused by IGRP (the Inter-Gateway Routing Protocol [14]) routing updates at the NEARnet routers [27]. The routers were unable to forward other packets while large routing updates were being processed. The particular problem of periodic packet losses on NEARnet has since been resolved; the router software has been changed so that normal packet routing can be carried out while the routers are dealing with routing update messages. Although it has been speculated that these packet drops were also connected with synchronization, it is unclear and there is no direct evidence [27], [18].

Periodic packet drops have been demonstrated associated with RIP (the Routing Information Protocol [13]) as well as with IGRP. Fig. 3 shows audio packet losses during an audiocast[2] of the December 1992 Packet Video workshop [15]. The x-axis shows the time in seconds; the y-axis shows the duration of each audio outage in seconds. The little blips more-or-less randomly spread along the time axis represent single packet losses. The larger loss spikes are strongly periodic; they occur every 30 seconds and last for several seconds at a time. During these events the packet loss rate ranges from 50 to 85% and there are frequent single outages of 100-500 ms. These periodic losses are almost certainly due to the source-routed (tunneled) multicast packets competing with routing updates and losing. Because 30 seconds is the default update time for RIP, these long intervals of packet losses are conjectured to result from RIP routing updates; it is not known if this problem involves synchronization. In other instances periodic 30-second audio packet losses have been conclusively traced to RIP routing updates [6], and there is some indirect evidence of synchronization.

In our "ping" experiments of the Internet in May 1992 we found many examples of periodic packet drops for which we have no explanation. For example, we found paths with packet drops every 318 seconds, paths with packet drops every 15 seconds, and paths with large delays every 45 seconds. We found different periodic patterns on the local path from LBL to the UC Berkeley campus at different times of the day. From our "ping" experiments, we conjecture that a significant number of packet drops in the Internet are associated with periodic processes of one type or another.

[2] For a report on the first such audiocast, see [5].

III. The Periodic Messages Model

This section describes a general model of periodic routing messages on a network; we call this the Periodic Messages model. This model was initially patterned after DECnet's DNA Phase IV, but other routing protocols that can conform to this model include EGP (Exterior Gateway Protocol) [21], Hello [20], IGRP, and RIP. In these routing protocols, each router on a network transmits a routing message at periodic intervals. This ensures that routing tables are kept up-to-date even if routing update messages are occasionally lost.

The Periodic Messages model behaves as follows.

1) The router prepares and sends a routing message. In the absence of incoming routing messages, the router resets its timer T_c seconds after step 1 begins. Other routers receive the first packet of this router's routing message T_d seconds after step 1 begins.
2) If the router receives an incoming routing message (or the first packet of an incoming routing message) while preparing its own outgoing routing message, the router also processes the incoming routing message. The router takes T_{c2} seconds to process an incoming routing message.
3) After completing steps 1 and 2, the router sets its timer. The time until the timer next expires is uniformly drawn from the interval $[T_p - T_r, T_p + T_r]$ seconds, where T_p is the average period and T_r represents a random component; this could be a (small) random fluctuation due to unavoidable variations in operating system overhead or a (larger) fluctuation due to a random

component intentionally added to the system. When the timer expires, the router goes to step 1.

4) If the router receives an incoming routing message after the timer has been set, the incoming routing message is processed immediately. If the incoming routing message is a "triggered update" caused by a major change in the network such as the failure of a link, then the router goes to step 1, *without* waiting for the timer to expire.

Because the router resets its timer only after processing its own outgoing routing message and any incoming routing messages, the timing of one router's routing messages can be affected by the routing messages from other nodes. This gives the weak coupling between routers, allowing the synchronization of routing messages from several routers.

The Periodic Messages model ignores properties of physical networks such as the possibility of collisions and retransmissions on an Ethernet. The Periodic Messages model is not intended to replicate the exact behavior of periodic routing messages, but to capture some significant characteristics of that behavior.

RIP and IGRP are intradomain routing protocols that use periodic routing messages. In RIP each router transmits periodic routing messages every 30 seconds. In IGRP, routers send routing messages at 90-second intervals.

EGP (Exterior Gateway Protocol) is used in some places between the NSFNET backbone and its attached regional networks; EGP routers send update messages every three minutes.[3] In the 1988 LBL network, DECnet routers implementing DNA Phase IV sent routing messages every two minutes. IGRP, RIP, and DECnet's DNA Phase IV all incorporate triggered updates, where routing messages are sent immediately in response to a network change such as the removal of a route. The first triggered update results in a wave of triggered updates from neighboring routers.

Not all implementations of these routing protocols correspond to the Periodic Messages model in this paper. The RFC for RIP [13] mentions that when there are many gateways on a single network, there is a tendency for the periodic routing messages to synchronize. The RFC specifies that in order to avoid this synchronization, either the routing messages must be triggered by a clock that is not affected by the time required to service the previous message, or a small random time must be added to the 30-second routing timer each time, though the magnitude of the random time is not specified. As an example of implementations that don't conform to the Periodic Messages model, in some implementations of IGRP and RIP routers reset their routing timers before the outgoing routing message is prepared, and routers don't reset their routing timers after triggered updates [17].

Thus the Periodic Messages model illustrates only one possible mechanism by which routing messages can become synchronized. Wherever there are interactions between routers, or between a router and the network, there could exist mechanisms that lead to synchronization.

[3]With BGP (Border Gateway Protocol), which runs on top of TCP, incremental update messages are sent as the routing table changes.

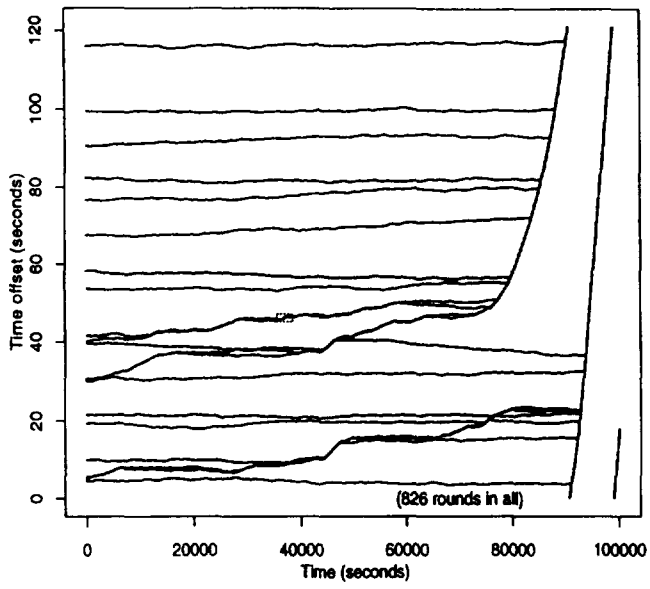


Fig. 4. A simulation showing synchronized routing messages

IV. Simulations

This section describes simulations of the Periodic Messages model. These simulations show the behavior of a network with N routing nodes on a single broadcast network, for $N = 20$. In the first set of simulations the periodic routing messages for the N nodes are initially unsynchronized; in the second set the periodic messages are initially clustered. The simulations show that the behavior of the Periodic Messages system is determined by the random overhead added to each node's periodic timer. As the level of randomization increases, the system's ability to break up clusters of synchronized routing messages also increases.

Definitions: T_p, T_r, T_c, T_{c2}, and T_d. The time T_p is the constant component of the periodic timer and T_r is the magnitude of the random component. Each router's routing timer is drawn at each round from the uniform distribution on $[T_p - T_r, T_p + T_r]$ seconds. Each router requires T_c seconds of computation time to process an outgoing routing message, and T_{c2} seconds of computation time to process an incoming routing message; each routing message could consist of multiple packets. In this paper we assume that T_{c2} and T_c are the same. T_d seconds after a router's routing timer expires, other routers receive the first packet of the routing message. □

For the simulations in this section, T_p is 121 seconds, T_c is 0.11 seconds, and T_d is set to zero; for the initial simulations in this section T_r is set to 0.1 seconds. The average timer-value of 121 seconds was chosen to give a minimum timer-value comparable to the 120 second timer used by the DECnet routers on our local network. The value of 0.11 seconds for T_c was chosen somewhat arbitrarily to model an estimated computation time of 0.1 seconds and transmission time of 0.01 seconds for a router to compute and transmit packets for an outgoing routing message after a timer expiration; these values are not based on any measurements of actual networks. Section V-C discusses how the results scale with different values for the various parameters.

When a node's routing timer expires, the node takes T_c seconds to prepare and transmit its routing message. We call this time the *busy period*. For each routing message received while a node is in its busy period, that node's busy period is extended by the $T_{c2} = T_c$ seconds required to process an incoming routing message.

For simplicity, in the simulations in this section T_d is set to zero; that is, when node A's timer expires the other nodes immediately receive the first packet of node A's routing message. Thus in the simulations, when node A's timer expires node A immediately spends T_c seconds preparing and transmitting its routing message, and at the same time the other routing nodes each spend $T_{c2} = T_c$ seconds receiving and processing the routing message from node A. This assumption most plausibly reflects a network with low propagation delay, where a router's routing message consists of several packets transmitted over a T_c-second period. Section V-D shows the results of simulations with $T_d > 0$.

The first set of simulations investigates the process by which initially unsynchronized routing messages become synchronized. The routing messages for the N nodes are initially unsynchronized; for each node the time at which the first routing message is sent is chosen from the uniform distribution on $[0, T_p]$ seconds. For the simulation in Fig. 4, T_r is set to 0.1 seconds. Each jittery line in Fig. 4 is composed of hundreds of points, and each point represents one routing message sent by a routing node. The x-axis shows the time in seconds at which the routing message was sent, and the y-axis shows the *time-offset*, i.e., the time modulo T, for $T = T_p + T_c$ seconds. This time-offset gives the time that each routing message was sent relative to the start of each *round*.

The simulation in Fig. 4 begins with unsynchronized routing messages and ends with the $N = 20$ routers transmitting their routing messages at essentially the same time each round. At the left-hand side of the figure the twenty jittery lines represent the time-offsets of the transmit times for the twenty nodes. In the absence of synchronization each router's timer expires, on the average, $T_p + T_c$ seconds after that router's previous timer expiration. These successive timer expirations give a jittery but generally horizontal line for the timer expirations for a single router. However, as we explain below, when routers become synchronized this increases the time interval between successive routing messages from a single router. At the end of the simulation the routing messages are fully synchronized, and all of the nodes set their timers at the same time each round. In this case each router has a busy period of $20 * T_c$ seconds rather than of T_c seconds, increasing the time interval between successive routing messages.

Fig. 5 is an enlargement of a small section of Fig. 4. This figure illustrates the synchronization of routing messages from two routers; each "x" marks a timer expiration, and each "□" marks the timer being reset. In the first five rounds of Fig. 5 the two nodes are independent, and each node sets its timer exactly T_c seconds after its previous timer expires. However, in the sixth round, node A's timer expires, say, at time t, and node A begins preparing its routing message. Before node A finishes preparing and sending its routing message, node B's timer expires; node A has to finish sending its own routing message *and* to process node B's routing message before it can reset its own timer. These two tasks take $T_c + T_{c2} = 2T_c$ seconds, so node A resets its timer at time $t + 2T_c$.

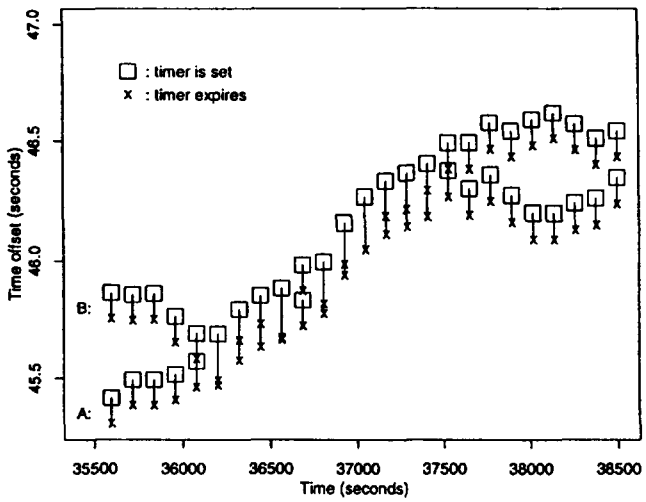


Fig. 5. An enlargement of the simulation in Fig. 4.

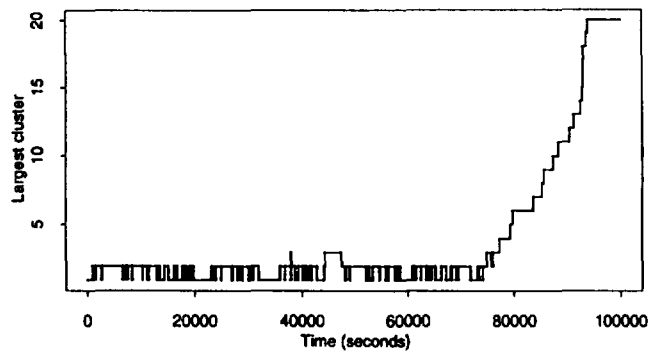


Fig. 6. The cluster graph, showing the largest cluster for each round.

In our model node B begins processing node A's routing message at time $t+T_d$, and in the simulation in Fig. 5 T_d is set to zero. While node B is receiving and processing node A's routing message, node B's own timer expires; node B has to prepare and send its own outgoing routing message *and* finish processing node A's routing message before resetting its timer. These tasks take $T_c + T_{c2} = 2T_c$ seconds, so for $T_d = 0$ node B also resets its timer at time $t + T_d + 2T_c = t + 2T_c$. At this point node A and node B are synchronized and we say that they form a *cluster*; node A and node B set their timers at the same time. The two nodes remain synchronized, setting their timers at roughly the same time, as long as the timers expire within $T_c - T_d$ seconds of each other each round. The cluster breaks up again when, because of the random component, node A and node B's timers expire more than $T_c - T_d$ seconds apart.

More generally, a *cluster of size* i refers to a set of i routing messages that have become synchronized. Each of the i nodes in a cluster is busy processing incoming routing messages and preparing its own outgoing routing message for iT_c seconds after the first timer in the cluster expires. For $T_d = 0$, the i nodes in a cluster reset their timers at exactly the same time.

One way to think of the simulation in Fig. 4 is as a system of N particles, each with some random movement in a one-dimensional space. For a particle in a *lone cluster* (a cluster of

size one), each timer-offset differs from the previous round's timer-offset by an amount drawn from the uniform distribution on $[-T_r, +T_r]$ seconds. In Fig. 4 the successive timer-offsets for an unsynchronized routing node (the movement of a single particle) are represented by a jittery but generally horizontal line.

For particles (or routing nodes) in a cluster of size i, $T_c + (1-i)T_{c2} = iT_c$ seconds are spent processing routing messages after the first timer of the cluster expires; then the nodes in the cluster all reset their timers. A cluster of i particles moves ahead a "distance" of roughly $(i-1)T_c$ seconds in each round. In Fig. 4 the movement of a cluster is represented by an irregular line with positive slope; the larger the cluster, the steeper the slope. When two clusters meet, the nodes in the two clusters all reset their timers at the same time; the two clusters merge, for the moment, into a larger cluster.

As Fig. 4 shows, a cluster of i particles can sometimes break up into two smaller clusters. Even though the i nodes set their routing timers at the same time, it is possible for one node's routing timer to expire more than $T_c - T_d$ seconds before any of the other nodes in the cluster, because of the random component in the timer interval for each node. When this happens, the first node *breaks out* of the cluster, as discussed further in Section V-A. The break-up of a cluster can be seen in Fig. 5 where a cluster of size two forms and then breaks up again.

The first part of the simulation in Fig. 4 shows small clusters occasionally forming and breaking up. Towards the end of the simulation a sufficiently large cluster is formed, moving rapidly across the space and incorporating all of the unclustered nodes that it encounters along its path. As the cluster size increases, the average period of the cluster also increases; the larger the cluster, the more quickly it "bumps into" and incorporates the smaller clusters.

A simulation at any point in time can be partially characterized by the size of the largest cluster of routing messages. Fig. 6 shows a cluster-graph of the simulation in Fig. 4. The x-axis shows time and the y-axis shows the size of the largest cluster in the current round of N routing messages.

Fig. 7 shows a simulation identical to that in Fig. 4, except that the simulation was started with a different random seed. Unlike the simulation in Fig. 4, the simulation in Fig. 7 ends with unsynchronized routing messages. For the simulation in Fig. 7, a cluster as large as five occasionally forms but each time the cluster breaks up again.

Fig. 9 shows the cluster graphs from several simulations that start with unsynchronized routing messages. The parameters are the same as the previous simulations, except that the random component T_r ranges from $0.6T_c$ to $1.4T_c$. Note that the time scale is different from the cluster graphs on previous pages; in Fig. 9 the simulations run for 10^7 seconds (115 days) instead of 10^5 seconds (just over 1 day). As the random component increases, the simulations take longer and longer to synchronize.

These simulations do not specifically include updates triggered by a change in the network. We can instead begin our simulations with synchronized routing messages, which can result from triggered updates. These simulations are shown in Fig. 10; the random component T_r ranges from $2.3T_c$ to $2.8T_c$. As the random component increases, the simulations return more quickly to the unsynchronized state.

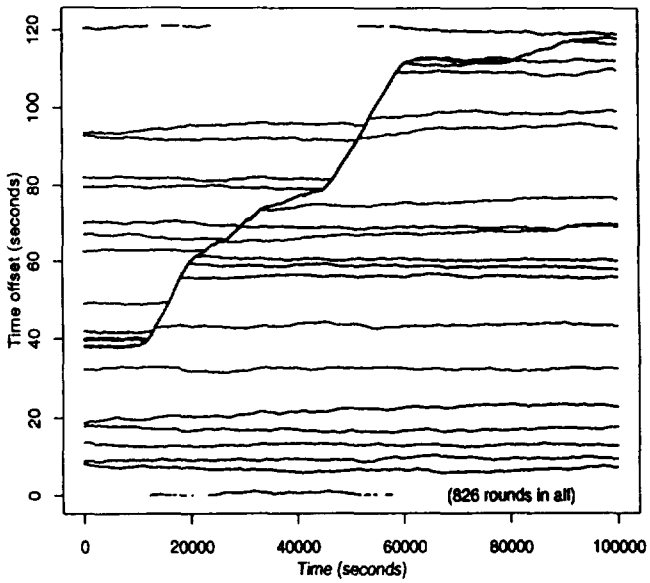


Fig. 7. A simulation showing unsynchronized routing messages.

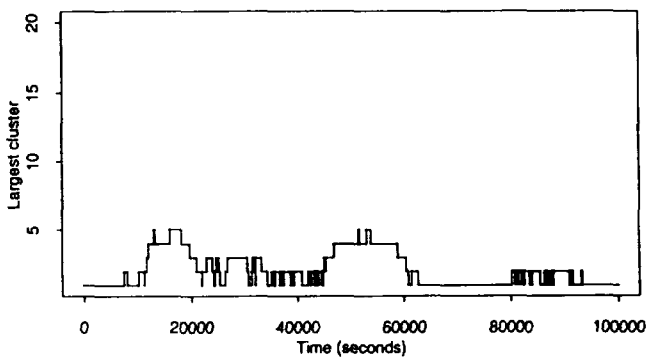


Fig. 8. The cluster graph, showing the largest cluster for each round.

Our simulation results are consistent with simulations of the same model in [29]. In addition to simulations, preliminary results from experiments by Treese have shown synchronization of systems on an Ethernet [30]. The experiments use an algorithm similar to the Periodic Messages model. The results suggest that the Periodic Messages model captures a realistic possible behavior of real computer systems.

V. The Markov Chain Model

This section uses a Markov chain model to further explore the behavior of the Periodic Messages system. The Markov chain explores the behavior of a system of N routers that each implement the Periodic Messages model described in the previous section. The Markov chain model assumes that each router receives a periodic routing message from every other router; this would be the case, for example, for N routers on a broadcast network. Section V-F discusses the issues in extending these results to N routers connected in an arbitrary topology, Section V-D discusses the effects of a nonzero transmission and propagation delay between routers, and Section VI discusses other analytical approaches to synchronization.

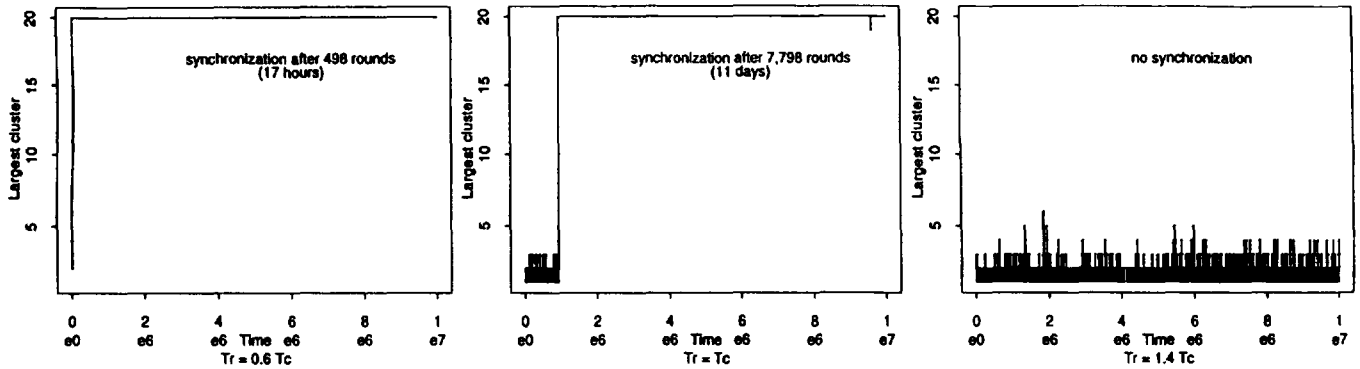


Fig. 9. Simulations starting with unsynchronized updates, for different values for T_r.

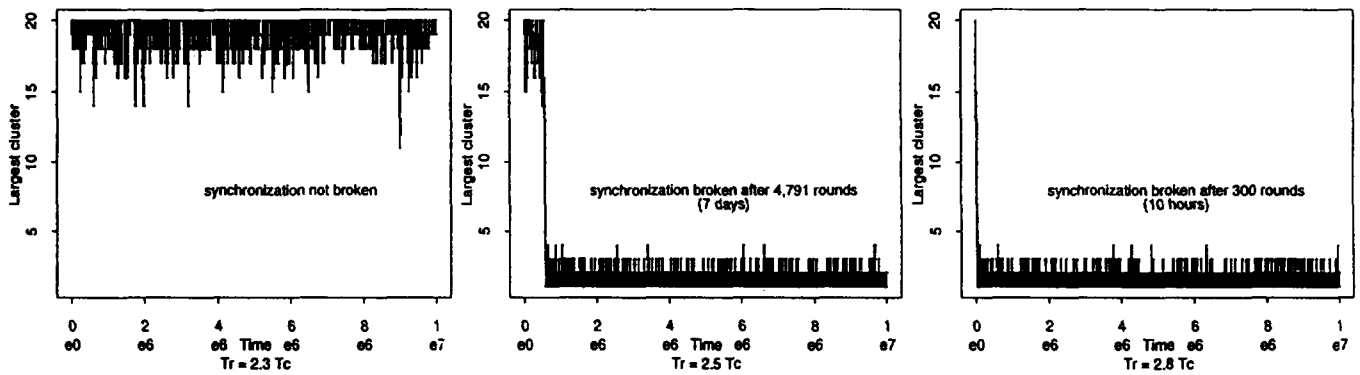


Fig. 10. Simulations starting with synchronized updates, for different values for T_r.

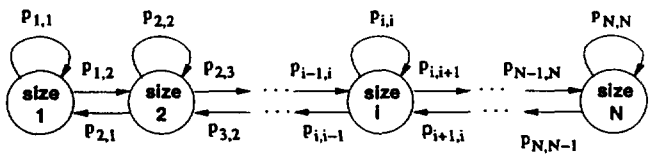


Fig. 11. The Markov chain.

The Markov chain model is used to compute the expected time for the system to move from an unsynchronized state to a synchronized state, and vice versa. This Markov chain model uses several simplifying assumptions and therefore only approximates the behavior of the Periodic Messages model. Nevertheless, the Markov chain model illustrates some significant properties of the simulations of the Periodic Messages model.

The Markov chain has N states;when the largest cluster from a round of N routing messages is of size i, the Markov chain is defined to be in state i. Fig. 11 shows the Markov chain, along with the transition probabilities. The transition probability $p_{i,j}$ is the probability that a Markov chain in state i moves to state j in the next round.

The Markov chain model is based on several simplifying assumptions:

- The first simplifying assumption of the Markov chain model is that the future behavior of the system depends only on the current state and is independent of past states. This assumption is clearly not true for the Periodic Messages model, where the future behavior of the system depends not only on the size of the largest cluster but on the transmit times of the other routing messages.
- The second simplifying assumption is that the size of the largest cluster changes by at most one from one round to the next. Again, this assumption is not strictly accurate, particularly for large values of N or T_r. For example, in the Periodic Messages model it is possible for two clusters of sizes i and 2 respectively to merge and form a cluster of size $i+2$ in the next round.
- The analysis of the Markov chain model assumes that except for the largest cluster of size i, all other clusters are lone clusters of size one; again, this conservative assumption is not strictly accurate. Given a cluster of size i, the *following* cluster is defined as the cluster that follows the cluster of size i in time. At each round, we assume that the "distance" between the largest cluster of size i and the following lone cluster is given by an exponential random variable with expectation $T_p/(N-i+1)$. This distance is defined as the wait between the time when the nodes in the cluster of size i *set* their timer and the time when the timer *expires* for the node in the following lone cluster. This expected value is based on the average distance between $N-i+1$ clusters.

As in the Periodic Messages model, we assume that each node's timer expires after a time drawn from the uniform distribution on $[T_p - T_r, T_p + T_r]$ seconds. For a node in a cluster of size i, the node takes $T_c + (i-1)T_{c2} = iT_c$ seconds to process the incoming and outgoing routing messages in the cluster, and other nodes receive the first packet of the routing message T_d seconds after the timer expires. In this section we

assume that $T_c < 2T_r + T_d$; if not, then a cluster never breaks up into smaller clusters.

The next two sections define the transition probabilities for the Markov chain. Given these transition probabilities, we compute the average time for the Markov chain to move from state 1 to state N, and the average time for the Markov chain to move from state N back down to state 1. This analysis shows that when T_r is sufficiently large, the Markov chain moves quickly from a synchronized state to an unsynchronized state.

A. Cluster Breakup and Growth

This section estimates $p_{i,i-1}$, the probability that the Markov chain moves from state i to state $i-1$ in one round. The second half of this section estimates $p_{i,i+1}$. In the Markov chain, a cluster of size i can break up to form a cluster of size $i-1$ either by breaking up into a cluster of size one followed by a cluster of size $i-1$, or vice versa. Because the first of the two cases is more likely,[4] for simplicity we only consider this case. We say that the first node *breaks away* from the head of the cluster.

Thus, $p_{i,i-1}$ is the probability that the node whose timer expires first, node A, resets its timer before it receives any routing messages from any of the other $i-1$ nodes in the cluster. For i nodes in a cluster, the i timers are all set within T_d seconds of each other; in this analysis we estimate that the i timers are all set at the same time and the timers expire at i times uniformly distributed in a time interval of length $2T_r$. Let L be the time from the expiration of the first timer until the expiration of the second of the i timers. In the absence of incoming messages, node A resets its timer T_c seconds after its timer expires, and receives notification of a routing message from another node in the cluster $L+T_d$ seconds after its timer expires. Because we assume that $T_c < 2T_r + T_d$, there is always a nonzero probability that a cluster of size i breaks up into smaller clusters.

From [8, p. 22],

$$p_{i,i-1} = Prob.(T_c < L + T_d) = \left(1 - \frac{T_c - T_d}{2T_r}\right)^i \qquad (1)$$

for $i > 1$.

Now we estimate $p_{i,i+1}$, the probability that the system moves from state i to state $i+1$ in one round. We leave $p_{1,2}$ as a variable; $p_{1,2}$ depends largely on T_r, the random change in the timer-offsets from one round to the next. For simplicity, this analysis assumes that $T_c = T_{c2}$.

The probability that a cluster of size two or more incorporates additional routing nodes, forming a larger cluster, depends largely on the fact that larger clusters have larger average periods than smaller clusters. After some time the larger cluster "collides" with a smaller cluster, and the two clusters merge.

For a cluster of size i, each node in the cluster sets its timer $T_c + (i-1)T_{c2} = iT_c$ seconds after the first timer in the cluster expires (or after it receives the first packet from that node's routing message). For $T_d = 0$, each of the i timer expirations is uniformly distributed in the interval $[T_p - T_r, T_p + T_r]$. Given i events uniformly distributed on the interval [0, 1], the expected value of the smallest event is $1/(i+1)$ [8, p.24]. Thus the first of the i timers expires, on average, $T_p - T_r + 2T_r/(i+1) = T_p - T_r(i-1)/(i+1)$ seconds after the timers are set. The average total period for a node in a cluster of size i is therefore $T_p - T_r(i-1)/(i+1) + iT_c$ seconds.

In one round the timer-offset for a cluster of size i moves an average distance of $(i-1)T_c - T_r(i-1)/(i+1)$ seconds relative to the timer-offset for a cluster of size one. For simplicity, in estimating $p_{i,i+1}$ we assume that the timer-offset for a cluster of size i moves in each round *exactly* $(i-1)T_c - T_r(i-1)/(i+1)$ seconds relative to the timer-offset for a cluster of size one. (This assumption ignores the somewhat remote possibility that a cluster of size i could "jump over" a smaller cluster.) What is the probability that, after one round, the timer-offset for a cluster of size i moves to within T_c seconds of the timer-offset for a cluster of size one?

The Markov chain model assumes that the distance between a cluster of size i and the following small cluster is an exponential random variable with expectation $T_p/(N-i+1)$. Thus for a Markov chain in state i, $p_{i,i+1}$ is the probability that an exponential random variable with expectation $T_p/(N-i+1)$ is less than $(i-1)T_c - T_r(i-1)/(i+1)$. For $2 \le i \le N-1$, this gives

$$p_{i,i+1} = 1 - e^{-((N-i+1)/T_p)((i-1)T_c - T_r(i-1)/(i+1))}. \qquad (2)$$

For all i, $p_{i,i} = 1 - p_{i,i-1} - p_{i,i+1}$.

B. Average Time to Cluster, and to Break Up a Cluster

This section investigates the average time for the Markov chain to move from state 1 to state N, and vice versa.

Definitions: $t_{i,j}$ and $f(i)$. Let $t_{i,j}$ be the expected number of rounds until the Markov chain moves from state i to state j, given that it moves from state i directly to state j. Let $f(i)$ be the expected number of rounds until the Markov chain first enters state i, given that the Markov chain starts in state 1. We leave $f(2)$ as a variable. □

We give a recursive definition for $f(i)$ for $i > 2$. The expected number of rounds to first reach state i equals the expected number of rounds to first reach state $i-1$, plus the additional expected number of rounds, after first entering state $i-1$, to enter state i. After state $i-1$ is first reached, the next state change is either to state $i-2$, with probability $(p_{i-1,i-2})/(p_{i-1,i-2} + p_{i-1,i})$, or to state i, with probability $(p_{i-1,i})/(p_{i-1,i-2} + p_{i-1,i})$. The expected number of rounds to reach state i, after first entering state $i-2$, is $f(i) - f(i-2)$. This leads to the following recursive equation for $f(i)$:

$$f(i) = f(i-1) + \frac{p_{i-1,i-2}}{p_{i-1,i-2} + p_{i-1,i}}(t_{i-1,i-2} + f(i) - f(i-2)) + \frac{p_{i-1,i}}{p_{i-1,i-2} + p_{i-1,i}} t_{i-1,i}.$$

[4] The second of the two cases occurs only if the last node transmits its routing message after it has had time to process routing messages from *all* previous nodes in the cluster.

Thus for $c(i) = t_{i-1,i} + (p_{i-1,i-2}/p_{i-1,i})t_{i-1,i-2}$,

$$f(i) - \frac{p_{i-1,i-2} + p_{i-1,i}}{p_{i-1,i}} f(i-1) + \frac{p_{i-1,i-2}}{p_{i-1,i}} f(i-2) = c(i). \tag{3}$$

From Appendix A, (3) has the solution[5]:

$$f(i) = f(2)\left(1 + \sum_{m=3}^{i}\left(\prod_{j=2}^{m-1}\frac{p_{j,j-1}}{p_{j,j+1}}\right)\right)$$

$$+ \sum_{m=3}^{i}\sum_{k=3}^{m}\left(c(k)\prod_{j=k}^{m-1}\frac{p_{j,j-1}}{p_{j,j+1}}\right). \tag{4}$$

Consider $t_{j,j+1}$, the expected number of rounds to move from state j to state $j+1$, given that the Markov chain in fact moves from state j to state $j+1$. Let $P_{j,x}$ be the probability that the Markov chain in state j first moves to state $j+1$ on round x, given that the Markov chain moves from state j to state $j+1$. The equation for $t_{j,j+1}$ is as follows [25, p. 37]:

$$t_{j,j+1} = \sum_{x=1}^{\infty} x\, P_{j,x}$$

$$= \sum_{x=1}^{\infty} x(p_{j,j})^{x-1} p_{j,j+1} = \frac{p_{j,j+1}}{(p_{j,j-1} + p_{j,j+1})^2}.$$

Similarly, the equation for $t_{j,j-1}$ is as follows:

$$t_{j,j-1} = \frac{p_{j,j-1}}{(p_{j,j-1} + p_{j,j+1})^2}.$$

Next we investigate the average time for the Markov chain to move from state N to state 1.

Definitions: $g(i)$. Let $g(i)$ be the expected number of rounds for the Markov chain to first enter state i, given that the Markov chain starts in state N.

Thus $g(N) = 0$ and

$$g(i) = g(i+1) + \frac{p_{i+1,i+2}}{p_{i+1,i+2} + p_{i+1,i}}(t_{i+1,i+2} + g(i) - g(i+2))$$

$$+ \frac{p_{i+1,i}}{p_{i+1,i+2} + p_{i+1,i}} t_{i+1,i}.$$

For $d(i) = t_{i+1,i} + (p_{i+1,i+2}/p_{i+1,i})t_{i+1,i+2}$, this gives the recursive equation

$$g(i) - \frac{p_{i+1,i+2} + p_{i+1,i}}{p_{i+1,i}} g(i+1) + \frac{p_{i+1,i+2}}{p_{i+1,i}} g(i+2) = d(i). \tag{5}$$

Equation (5) has the solution below:

$$g(i) = \sum_{m=i}^{N-1}\sum_{k=m}^{N-1}\left(d(k)\prod_{j=m+1}^{k}\frac{p_{j,j+1}}{p_{j,j-1}}\right) \tag{6}$$

[5]This solution could also be verified by the reader by substituting the right-hand side of (4) into (3) .

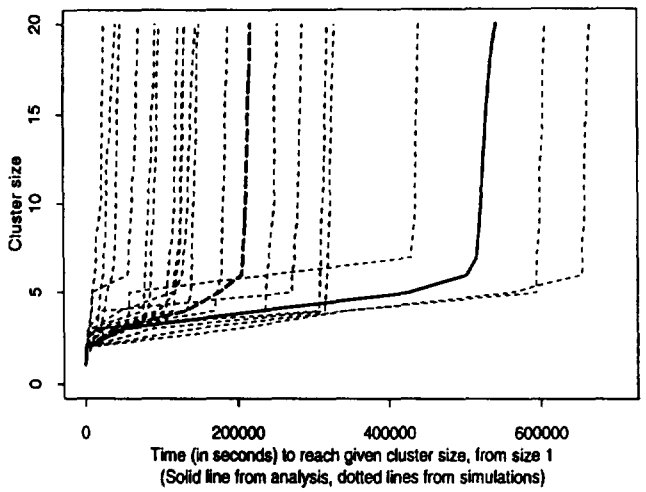


Fig. 12. The expected time to reach cluster size i, starting from cluster size 1, for $T_r = 0.1$ seconds.

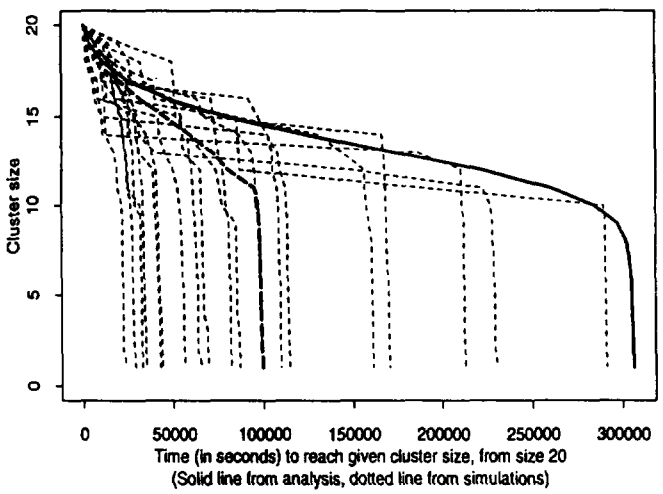


Fig. 13. The expected time to reach cluster size i, starting from cluster size N, for $T_r = 0.3$ seconds.

The derivation of this equation is similar to that of $f(i)$. Note that this equation does not depend on the values of $p_{1,2}$ or of $f(2)$.

The solid line in Fig. 12 shows $f(i)$, computed from (4), for $N = 20$, $T_p = 121$ seconds, $T_c = 0.11$ seconds, $T_r = 0.1$ seconds, $T_d = 0$ seconds, and $f(2) = 19$ rounds. (This value for $f(2)$ is based both on simulations and on an approximate analysis that is not given here.[6]) The x-axis shows the time in seconds, computed as $(T_p + T_c)f(i)$. The y-axis shows the cluster size i; a mark is placed at cluster size i when the system first reaches that cluster size. The results of twenty simulations are shown by light dashed lines. Each simulation was started with unsynchronized routing messages, with the values for N, T_p, T_c, T_d, and T_r described above; these simulations differ only in the random seed. The heavy dashed line shows the results averaged from twenty simulations.

[6]The dynamics for moving from a cluster of size two or more to a larger cluster is based largely on the fact that larger clusters have larger average periods. In contrast, the dynamics for moving from a cluster of size one to a cluster of size two depends on how frequently two clusters of size one 'collide', where all clusters of size one have the same average period; this requires a different analysis.

The solid line in Fig. 13 shows $g(i)$, computed from (6), for the same parameters for N, T_p, T_d, and T_c as in Fig. 12, and for $T_r = 0.30$ seconds; for the value of T_r in Fig. 12, the system takes a long time to unsynchronize, making simulations unrealistic. The heavy dotted line averages the results from twenty simulations.

Figs. 12 and 13 show that the average times predicted by the Markov chain are two or three times the average times from the simulations. This discrepancy is not surprising, because the Markov chain is only a rough approximation of the behavior of the Periodic Messages system. Aside from the difference in magnitude, however, the functions predicted from the Markov chain and computed from the simulations are reasonably similar. Thus the Markov chain model does in fact capture some essential properties of the Periodic Messages system.

C. Results from the Markov Chain Model

This section explores the general behavior of the Markov chain model. We compute the expected time for the Markov chain to synchronize and to unsynchronize, for a range of values for N, T_c, and T_r, and compare these analytical results to the results of simulations. This comparison shows that the Markov chain model is *explanatory* rather than *predictive*; the Markov chain model and the simulations exhibit the same qualitative behavior, and the Markov chain model can be used to *explain* the behavior of the simulations, but the Markov model is not sufficiently accurate to *predict* the exact results of the simulations.

The analysis in this section, along with the simulations, shows that for a wide range of parameters, choosing T_r as a small multiple of T_c ensures that the system is almost always unsynchronized. The analysis further shows that for fixed values for T_c and T_r, the transition to synchronization is an abrupt function of the number of nodes N. Finally, in this section we consider a system of routers in an arbitrary topology, where each router only receives periodic messages from its immediate neighbors. We suggest that the model of synchronized routing messages in this paper is likely to hold in arbitrary topologies only for connected subsets of nodes with similar degree.

Fig. 14 considers the expected time for the Markov chain to synchronize or to unsynchronize, as a function of the parameter T_r. Fig. 14 gives $f(N)$, from (4), and $g(1)$, from (6), for T_r ranging from zero to $4.5T_c$, given $N = 20$, $T_p = 121$ seconds, $T_c = 0.11$ seconds, and $T_d = 0$ seconds. The solid line on the right shows the expected time for the Markov chain to move from state N to state 1; the solid line on the left shows the expected time for the Markov chain to move from state 1 to state N. The dotted line on the left was computed using values for $f(2)$ based on an approximate analysis that is not given here; the solid line on the left uses $f(2)$ set to zero. For $T_r < 0.5(T_c - T_d)$, clusters never break up once they have formed, and the time to synchronize depends largely on the time to first form a cluster of size two; this time increases as T_r approaches 0.[7] Note that the y-axis is on a log scale, and

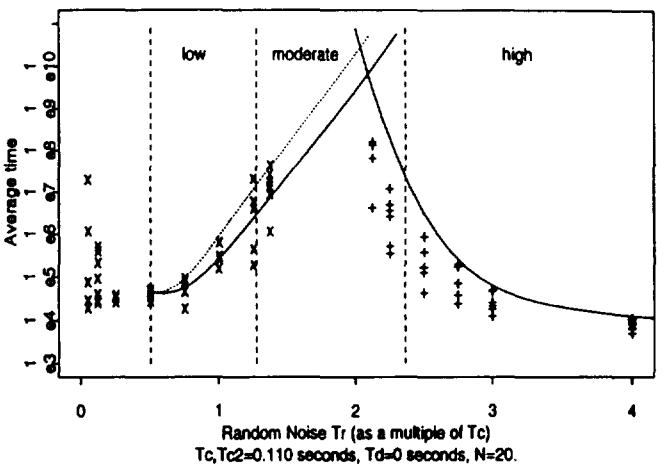


Fig. 14. Expected time to go from cluster size 1 to cluster size N, and vice versa, as a function of T_r.

ranges from less than 10^3 seconds (roughly 16 minutes) up to 10^{11} seconds (over 3 thousand years).

Fig. 14 can be used as a general guide in choosing a sufficiently large value of T_r, given the values for the other parameters in a system, so that the system moves easily from state N to state 1 and rarely moves from state 1 back to state N. The figure shows the regions of low, moderate, and high randomization. In the region of low randomization the system moves easily from state 1 to state N; in the region of high randomization the system moves easily from state N to state 1. In the region of moderate randomization the system takes a significant period of time to move either from state 1 to state N, or from state N back to state 1. In the low and moderate regions $f(N)$, the expected time for the Markov chain to move from state 1 to state N, grows exponentially with T_r. The "X" marks on Fig. 14 show simulations that start with unsynchronized routing messages and the "+" marks show simulations that start with synchronized routing messages.

Fig. 15 shows the same analytical results as in Fig. 14 for the number of nodes N ranging from 10 to 30, and for a range of values for T_c. These simulations were performed to check how accurately the analytical results predict the simulation results for a range of parameters. Note that for larger values of T_c and of N, the analytical results significantly overestimate the time required by the simulations to go from state N to state 1. The analytical results use the simplifying assumption that the size of the largest cluster changes by at most one from one round to the next. As the parameters T_c and N increase, this assumption becomes less applicable.

The figures show that for a wide range of parameters, choosing T_r at least ten times greater than T_c ensures that clusters of routing messages will be quickly broken up. For any range of parameters, choosing T_r as $T_p/2$ should eliminate any synchronization of routing messages. This would be equivalent to setting the routing timer each time to an amount from the uniform distribution on the interval $[0.5T_p, 1.5T_p]$ seconds. This introduces a high degree of randomization into the

[7] As Fig. 14 shows, for extremely small values of T_r there is little randomness in the system, and it can take some time for two nodes to first form a cluster.

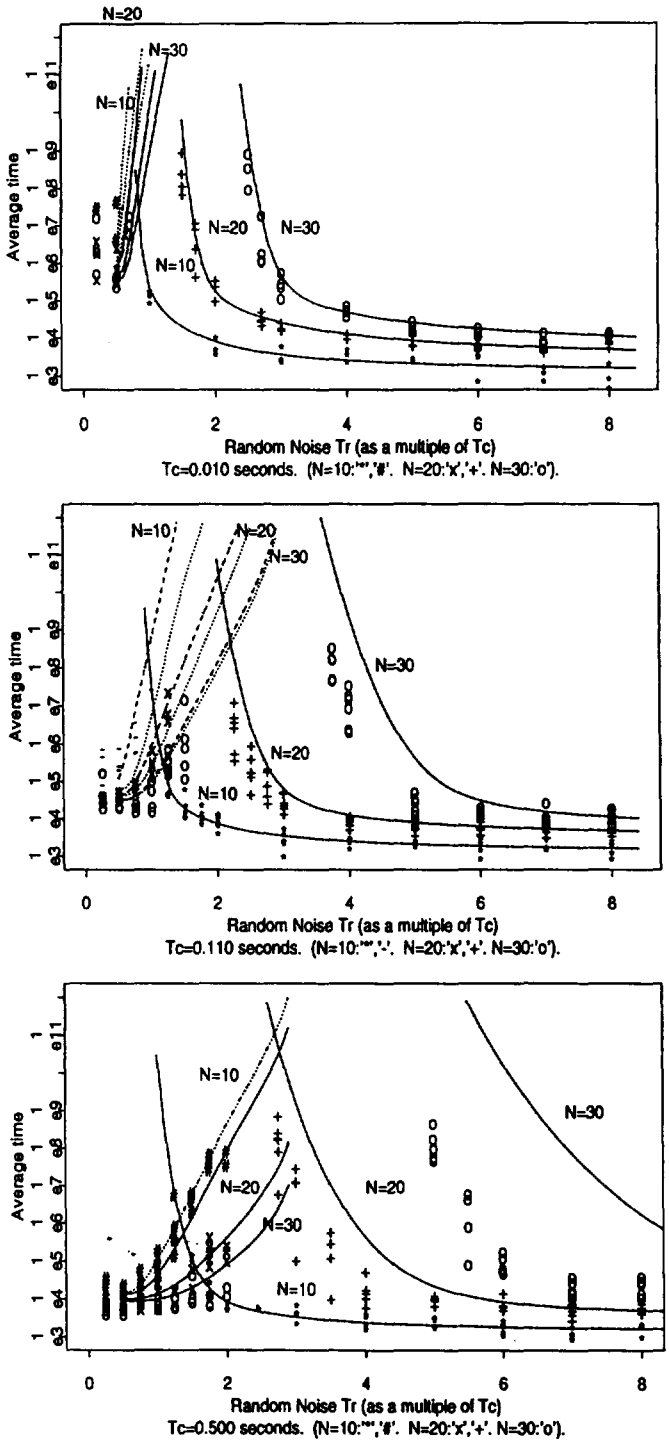


Fig. 15. Expected time to go from cluster size 1 to cluster size N, and vice versa, as a function of N and of T_r.

system, yet ensures that the interval between routing messages is never too small or too large.

D. Incorporating Delays Between Routers

The analysis and simulations in the paper so far have assumed that $T_d = 0$; that is, that when a node's timer expires, other routers are immediately notified of the timer expiration. While small values for T_d accurately reflect a model of routing nodes where propagation delay is low and

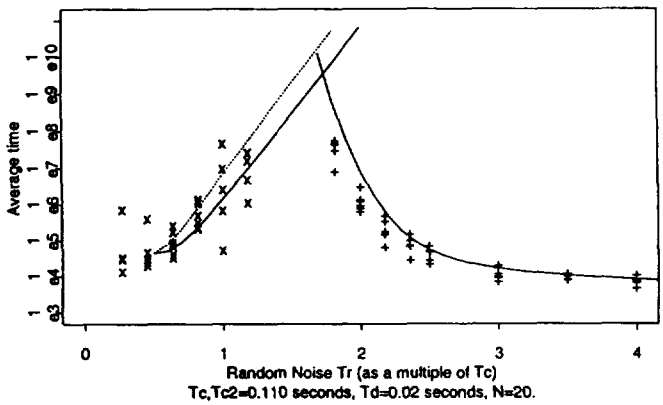


Fig. 16. Time to go from a cluster of size 1 to a cluster of size N, and vice versa, for $T_d = 0.02$ seconds.

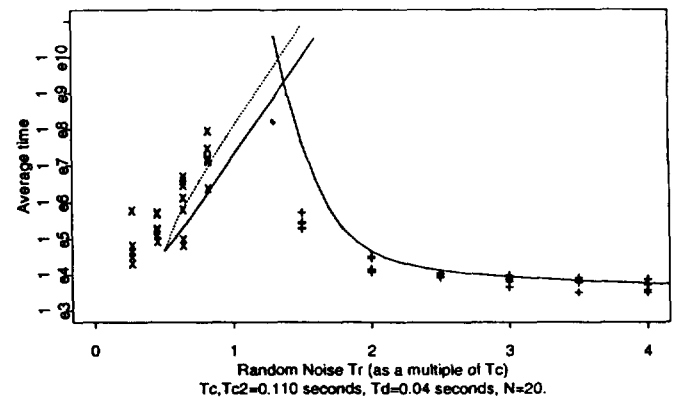


Fig. 17. Time to go from a cluster of size 1 to a cluster of size N, and vice versa, for $T_d = 0.04$ seconds.

each routing message consists of a number of packets, it is physically impossible for T_d to be zero. In this section we explore simulations with small nonzero values for T_d.

Recall that, in the absence of incoming routing messages, router A resets its timer T_c seconds after its timer expires, and router B is notified of router A's incoming routing message T_d seconds after router A's timer expires. If $T_d > T_c$ (for example, because router A resets its timer before it transmits the first packet of the routing timer), then there is little coupling between adjacent routers. In this case, if two routers' timers expire at the roughly same time, then each router resets its timer before receiving a routing message from the other router, and clusters break up quickly. In this section we explore simulations with $0 < T_d < T_c$. This reflects a model where each routing message consists of multiple packets, and neighboring routers receive the first packet of a routing message before the source router resets its timer.

Fig 16 shows the results of simulations with $T_d = 0.02$ seconds. The lines show the same analytical results given in Fig. 14, but computed for $T_d = 0.02$ seconds. From the analysis in Section V, $p_{i,i-1}$, the probability that a cluster of size i breaks into a cluster of size $i-1$ in one round, can be estimated by

$$p_{i,i-1} = \left(1 - \frac{T_c - T_d}{2T_r}\right)^i \tag{7}$$

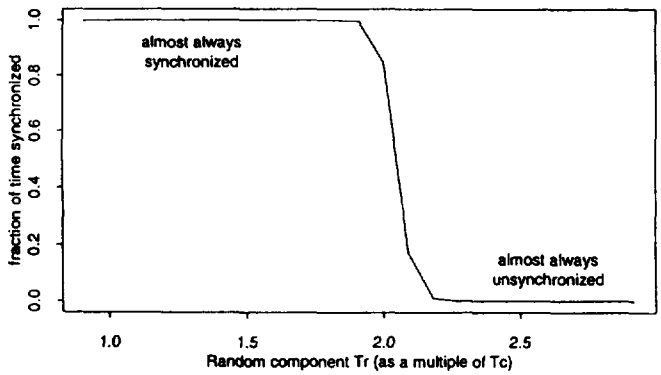


Fig. 18. The fraction of time synchronized versus the random component T_r.

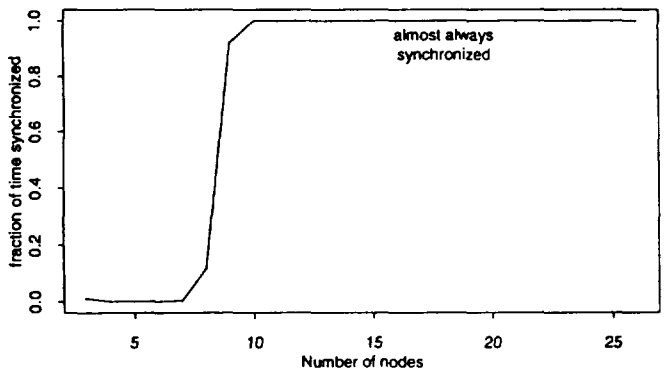


Fig. 19. The fraction of time synchronized versus the number of nodes, for $T_r = 0.11$ seconds.

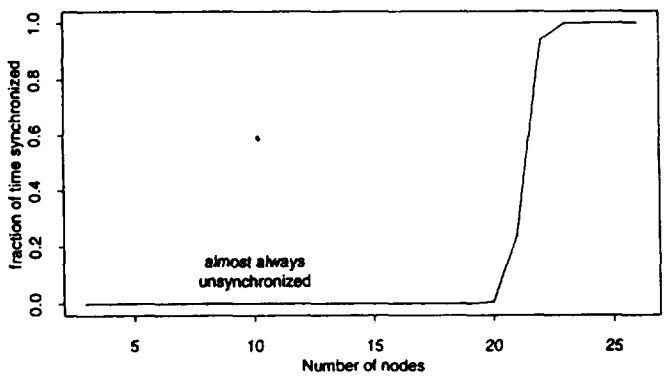


Fig. 20. The fraction of time synchronized versus the number of nodes, for $T_r = 0.30$ seconds.

for $i > 1$ and $T_c - T_d < 2T_r$. As this equation describes, the main effect of increasing T_d is to increase $p_{i,i-1}$.

In general, five simulations were run for each value of T_r, and each simulation was terminated after 10^8 seconds. The "X" marks show simulations that start with unsynchronized routing messages and the "+" marks show simulations that start with synchronized routing messages. Note that with T_d set to 0.02 seconds rather than to zero, the simulations take longer to synchronize and less long to unsynchronize. Nevertheless, the basic behavior of synchronization is preserved. As Fig. 17 shows, increasing T_d from 0.02 seconds to 0.04 seconds further increases the time required for synchronization.

The simulations and analysis show that the time to synchronize increases as $T_c - T_d$ increases. After a node's timer expires, this is the time between when other nodes are notified of the timer expiration, and when the node resets its own timer (in the absence of incoming routing messages). This interval can be affected by a number of factors, such as the propagation delay, the number of packets in the routing message and the timing between the transmission of these packets, and the promptness with which nodes reset their routing timers.

E. Steady-State Behavior

One quantity of interestis the fraction of time that the Markov chain spends with low cluster sizes. We were only able to estimate the equilibrium distribution for the Markov chain by further approximating the transition probabilities. However, one simple way to estimate the fraction of time that the Markov chain spends in synchronized states is to compute $g(1)/(f(N)+g(1))$. Recall that $f(N)$ is the expected number of rounds for the system to move from state 1 to state N; for most of this time the system is largely unsynchronized. Similarly, $g(1)$ is the expected number of rounds for the system to move from state N to state 1; for most of this time the system is largely synchronized.

In Fig. 18 the x-axis shows T_r; the other parameters are $N = 20$, $T_p = 121$ seconds, $T_d = 0$, and $T_c = 0.11$ seconds. The y-axis is $g(1)/(f(N) + g(1))$, the estimated fraction of time for which the system is synchronized. As Fig. 18 shows, as T_r is increased, the system makes a sharp transition from predominately-synchronized to predominately-unsynchronized. The simulations and analysis in Fig. 15 show that for a wide range of values for N and T_c, the transition from predominately-synchronized to predominately-unsynchronized occurs for T_r a small multiple of T_c.

Figs. 19 and 20 show the estimated fraction of time that the Markov chain spends synchronized as a function of the number N of nodes in the network. The parameters for these figures are $T_p = 121$ seconds, $T_d = 0$, and $T_c = 0.11$ seconds; T_r is 0.11 seconds in Fig. 19 and 0.3 seconds in Fig. 20. For each figure, as the number of nodes is increased the system makes a sharp transition from predominately-unsynchronized to predominately-synchronized. This corresponds in practice to a network that moves from an unsynchronized to a fully synchronized state when one additional router is added to the system.

Figs. 19 and 20 show that the number of nodes where the transition to synchronization takes place is a function of the other parameters of the system; in Figure 15 the transition occurs for $N = 9$, and for Fig. 20 the transition occurs for $N = 22$. Recall that because of the simplifying assumptions, the analysis is explanatory rather than predictive; the analysis explains why the transition to synchronization is an abrupt function of the number of nodes, but the analysis does not necessarily accurately predict the exact number of nodes at which this transition takes place.

F. Topologies with Point-to-Point Links

The analysis in this paper applies to a network of N routers where each router receives a periodic routing message from

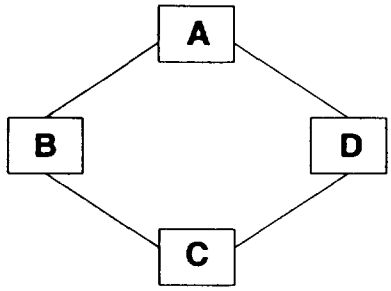


Fig. 21. A ring topology of routers.

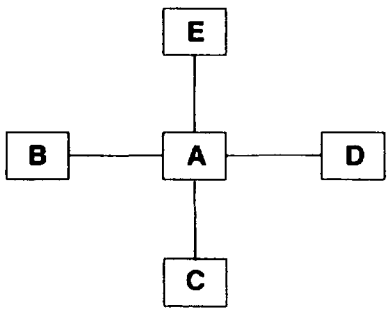


Fig. 22. A star topology of routers.

each other router. Consider instead N routers in an arbitrary topology, where the routers are connected by point-to-point links and each router only receives periodic messages from its immediate neighbors. It is still possible for all N routers in an arbitrary topology to become synchronized, using the mechanisms described in this paper, but the synchronization dynamics depend strongly on the particular network topology.

For example, for the topology of routers in Fig. 21 each router receives a routing message from two neighboring routers. Thus each router has to prepare its own outgoing routing message and to process two incoming routing messages. If all four routers become synchronized, then each router's timer expires on the average $T_p+T_c+2T_{c2} = T_p+3T_c$ seconds after its busy period begins, and each router's busy period begins when either its timer or one of its neighbors' timers expires. The process of synchronization in a network where each routing node has constant degree is similar to the process described in Section V-B, and for small values of T_r such a network is likely to remain synchronized once synchronization occurs.

For the star topology of routers in Fig. 22, however, given the Periodic Messages model discussed in this paper, the synchronization of routing messages is less likely to occur. In Fig. 22 Router A has to process four incoming routing messages, while each of the other routers processes only one incoming routing message. Assume that all five routers transmit a routing message at the same time. Then router A's timer expires on the average $T_p + 5T_c$ seconds later, while each of the other routers' timers expires on the average $T_p + 2T_c$ seconds later. This difference in period should break up synchronization fairly rapidly.

Thus for routers in an arbitrary topology the tendency towards synchronization can depend heavily on the details of the topology. Current internet topologies where most routers have a similar and low degree should lend themselves to synchronization.

VI. Other Approaches to Synchronization

The Markov chain model in Section V was designed to capture some key properties of the routing messages model such as the role of randomization in producing an initial cluster of size two, the role of changes in period in enabling the formation of larger clusters, and the role of randomization in breaking up clusters once they have formed. In addition, this Markov model seems (to us) fairly simple and straightforward. However, there are many possible approaches to the analysis of synchronization; in this section we discuss briefly coupled nonlinear oscillators and coupling methods for Markov processes.

The approach to synchronization in [2] concerns coupled nonlinear oscillators and uses the tools from nonlinear dynamic systems. The book shows that the tendency of synchronization is characteristic of a broad class of dynamic systems. Because [2] does not treat the question of synchronization in the presence of randomization, the results are not immediately applicable to the synchronization of routing messages.

Mathematical models have recently been developed to analyze "pulse-coupled" oscillator systems [28] such as communities of Thai fireflies, where the periodic pattern of one firefly's flashing is affected by the others nearby. At dusk male fireflies gather in trees by the edge of the river and flash on and off in an unsynchronized fashion but, as the night progresses, whole trees of fireflies will flash in synchronization for hours. These "pulse-coupled" oscillator systems are similar to our model of routing messages where the timing of one router's routing messages can be affected by the arrival of routing messages from a neighboring router.

A different approach to synchronization comes from the literature on coupling methods for Markov processes [12]. In the classical coupling [12, p. 13], two independent copies of a Markov process evolve until they reach a common state then, from that point, the two Markov processes use the same transition mechanism and follow the same path. It might be possible to express our model for the synchronization of routing messages as an example of coupled stochastic processes, with each routing node a separate Markov process, but the analysis that we use does not come from the literature of coupled Markov processes.

VII. Conclusions and Open Questions

In this section we give some specific conclusions about the synchronization of routing messages, then discuss conclusions and open questions about the more general problem of unanticipated structure in the Internet.

As the simulations and analysis in this paper demonstrate, periodic routing messages from a system of routers in a network can easily become synchronized. The simulations and analysis both show that this synchronization is an emergent property at a particular scale with an abrupt transition from unsynchronized to synchronized behavior. Thus the behavior

of the system with a small number of routers can not necessarily be extrapolated to explain the behavior of the system with a large number.

Synchronization can be avoided with appropriate implementation of the routing protocols. One possible method is the addition of a random component to the routing timer intervals. Our analysis provides general guidelines on determining the magnitude of the random component necessary to avoid synchronization. In particular, setting the timer each round to a time from the uniform distribution on the interval $[0.5T_p, 1.5T_p]$ seconds is a simple way to avoid synchronized routing messages.[8] Adding a random component to the routing timers changes the strict periodicity of the routing messages but does not affect the convergence properties of the underlying routing protocols.

A second method for avoiding synchronization is to implement a routing timer that is independent of external events (as mentioned in the specifications for RIP) [13, p. 23]. If each router resets its timer immediately after the timer expires (regardless of its activities when the timer expires) and if routers don't reset their timers after triggered updates, then the process of timer synchronization described in this paper might be avoided. There are, however, drawbacks to this approach: if routers are initially synchronized (either by chance, or because they were restarted at the same time) then they will remain synchronized since there is no mechanism to break up synchronization if it does occur.

It is also possible to reduce the negative impact of synchronized routing updates by modifying routers to give acceptable performance in the presence of large or synchronized routing updates. While it is often more efficient to exploit traffic structure rather than to engineer it out, this does not seem to be the case for synchronized routing messages. Even with tuned router implementations, synchronized routing messages place an unnecessary burden on the network, and a preferable solution is to avoid synchronization in the first place.

Periodic routing messages are not the only example of unintended structure in the Internet. Our "ping" experiments suggest that many periodic processes are at work. From these experiments we conjecture that a significant number of the packet drops in the Internet are associated with some form of periodic process, but the causes of structure in current Internet traffic are only beginning to be explored.

One model of large-scale structure comes from the kinetic theory of automobile traffic [24]. Individual drivers, each seeking to optimize their own goals, can produce collective traffic patterns akin to the coordinated motion of flocks of birds or schools of fish. Individual strategies of reducing speed during congestion result in a collective decrease in traffic flow as the vehicle concentration increases past a certain density. [16] shows that given an initially homogeneous traffic flow, regions of high density and low average velocity (clusters of cars) can spontaneously appear. These high-density regions can move either with or against the flow of traffic, and two clusters with different velocities, widths, and amplitudes merge when they meet, resulting in a single cluster. We believe there are analogous interactions between packet streams and we intend to explore these mechanisms for the emergence of large-scale structure in packet-switched networks.

As the Internet expands to new types of traffic (e.g., voice and video), new routing patterns (multicast distribution), and new gateway scheduling disciplines (Quality-of-Service for realtime traffic), it is important to anticipate the large-scale structure that might be introduced by these changes. But, as our analysis of routing shows, large scale structure is often an emergent property that cannot be observed or inferred from small scale simulations or measurements. Although they do not currently exist, we feel that large scale simulators are a necessary tool for exploring questions of large scale structure. Large scale simulations can help us build intuition about the behavior of large-scale networks, better understand behavior of current Internet traffic, and predict how this behavior might change as Internet traffic types, routing patterns, and gateway scheduling disciplines evolve.

[8] Pseudo-random numbers can be efficiently generated by a random number generator [4].

ACKNOWLEDGMENT

We thank T. Li, S. McCanne, M. Moran, V. Paxson, and L. Zhang for their comments on earlier drafts of this paper. Some of the simulation and analysis in this paper was done several years ago, after observing synchronized DECnet routing messages in 1988. After a report in 1992 by Professor Agrawala on packet drop rates in the Internet referred to long delays at periodic 90-second intervals on paths to MIT, we reported to the End-to-End Research Group that we found a similar problem of packet drops at 90-second intervals on paths to MIT. We thank Dave Clark for pursuing these problems further; initial conjectures that the problems of periodic packet drops at MIT were related to synchronized routing messages pushed us to finish our draft of this paper. We have enjoyed discussions with Win Treese, who has also investigated the synchronization of routing messages.

APPENDIX
SOLUTION TO RECURSIVE FORMULA

This section gives the solution for the following recursive equation for $f(i)$:

$$f(i) - \frac{p_{i-1,i-2} + p_{i-1,i}}{p_{i-1,i}} f(i-1) + \frac{p_{i-1,i-2}}{p_{i-1,i}} f(i-2) = c(i),$$

where $c(i) = t_{i-1,i} + (p_{i-1,i-2}/p_{i-1,i})t_{i-1,i-2}$. We express this, using operator notation, as

$$(E^2 - \frac{p_{i-1,i-2} + p_{i-1,i}}{p_{i-1,i}} E + \frac{p_{i-1,i-2}}{p_{i-1,i}}) f_{i-2} = c(i)$$

$$\Leftrightarrow \quad (E - \frac{p_{i-1,i-2}}{p_{i-1,i}})(E-1) f_{i-2} = c(i).$$

For an explanation of operator notation in solving recursive equations, see [11] or [19]. Let

$$z_i = (E-1) f_{i-2}.$$

This gives the first order difference equation

$$(E - \frac{p_{i-1,i-2}}{p_{i-1,i}})z_i = c(i)$$

$$\Rightarrow \; z_{i+1} = \frac{p_{i-1,i-2}}{p_{i-1,i}} z_i + c(i).$$

Therefore

$$z_i = b(i)z_{i-1} + c(i-1) \quad \text{for } b(i) = \frac{p_{i-2,i-3}}{p_{i-2,i-1}},$$

with the initial condition $z_3 = f_2 - f_1$, and for $f_1 = 0$. This has the solution [11, p. 18]

$$z_i = z_3 \prod_{j=4}^{i} b(j) + \sum_{k=4}^{i} \left(c(k-1) \prod_{j=k+1}^{i} b(j) \right).$$

Now solving for

$$(E-1)f_{i-2} = f_2 \prod_{j=4}^{i} b(j) + \sum_{k=4}^{i} \left(c(k-1) \prod_{j=k+1}^{i} b(j) \right),$$

we have that

$$f_{i-1} = f_{i-2} + f_2 \prod_{j=4}^{i} b(j) + \sum_{k=4}^{i} \left(c(k-1) \prod_{j=k+1}^{i} b(j) \right),$$

with the initial condition f_2. This has the solution

$$f_i = f_2 + \sum_{m=3}^{i} \left(f_2 \prod_{j=4}^{m+1} b(j) \right) + \sum_{m=3}^{i} \sum_{k=4}^{m+1} \left(c(k-1) \prod_{j=k+1}^{m+1} b(j) \right).$$

Substituting for $b(j)$, we get

$$f_i = f_2 + \sum_{m=3}^{i} \left(f_2 \prod_{j=2}^{m-1} \frac{p_{j,j-1}}{p_{j,j+1}} \right) + \sum_{m=3}^{i} \sum_{k=3}^{m} \left(c(k) \prod_{j=k}^{m-1} \frac{p_{j,j-1}}{p_{j,j+1}} \right).$$

REFERENCES

[1] M. Baker, private communication, 1992.

[2] I. I. Blekhman, *Synchronization in Science and Technology*. New York: ASME Press Translations, 1988.

[3] H. Braun, B. Chinoy, K. Claffy, and G. Polyzos, "Analysis and modeling of wide-area networks: Annual status report," CSL, Univ. of California, San Diego, Feb. 1993.

[4] D. Carta, "Two fast implementations of the 'minimal standard' random number generator," *Commun. ACM*, vol. 33 no. 1, p.87-88, Jan. 1990.

[5] S. Casner and S. Deering, "First IETF Internet audiocast," *Comput. Commun. Rev.*, vol. 22 no. 3, pp. 92-97, July 1992.

[6] S. Deering, private communication, 1993.

[7] E. M. R. A. Engel, *A Road to Randomness in Physical Systems*. New York: Springer-Verlag, 1992.

[8] W. Feller, *An Introduction to Probability Theory and Its Applications, V. II*. New York: Wiley, 1966.

[9] S. Floyd and V. Jacobson, "On traffic phase effects in packet-switched gateways," *Internetworking: Research and Experience*, vol. 3, no. 3, Sept. 1992. p.115-156.

[10] ——, "The synchronization of periodic routing messages," presented at *SIGCOMM 1993*, Sept. 1993, pp. 33-44.

[11] D. H. Greene, and D. E. Knuth, *Mathematics for the Analysis of Algorithms*. New York: Birkhauser, 1982.

[12] D. Griffeath, "Coupling methods for markov processes," in *Advances in Mathematics Supplementary Studies*. New York: Academic, 1978, vol. 2, pp. 1-43.

[13] C. Hedrick, "Routing information protocol," Request For Comments (RFC) 1058, June 1988.

[14] ——, "An introduction to IGRP," Aug. 1991. Available by anonymous ftp from shiva.com:doc/igrp.ps.Z.

[15] V. Jacobson, "Audio losses during yesterday's packet video workshop audiocast," Nov. 11, 1992. Message-ID 92121120398.AA24349@rx7.ee.lbl.gov. archived in nic.es.net:ietf/rem-conf.

[16] B. S. Kerner and P. Konhauser, "Cluster effect in initially homogeneous traffic flow," *Phys. Rev. E*, vol. 48, no. 4, pp. 2335-2338.

[17] T. Li, private communication, 1993.

[18] D. Long, private communication, 1993.

[19] R. E. Mickens, *Difference Equations*. New York: Van Nostrand Reinhold, 1987.

[20] D. L. Mills, "DCN local-network protocols," Request For Comments (RFC) 891, Dec. 1983.

[21] ——, "Exterior gateway protocol formal specification," RFC 904, Apr. 1984.

[22] V. Paxson, "Empirically-derived analytic models of wide area TCP connections," LBL Tech. Rep. LBL-34086, May 1993.

[23] ——, "Growth trends in wide-area TCP connections," submitted to *IEEE Network*, 1993.

[24] I. Prigogine and R. Herman, *Kinetic Theory of Vehicular Traffic*. New York: American Elsevier, 1971.

[25] S. M. Ross, *Introduction to Probability Models*. New York: Academic, 1985.

[26] D. Sanghi, A. Agrawala, O. Gudmundsson, and B. Jain, "Experimental Assessment of End-to-End Behavior on Internet," Univ. of Maryland, Tech. Rep. UMIACS-TR-92-62, June 1992.

[27] J. Schiller, private communication, Sept. 1992.

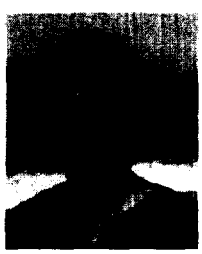
[28] S. Strogatz and I. Stewart, "Coupled oscillators and biological synchronization," *Sci. Amer.*, pp. 102-109, Dec. 1993.

[29] W. Treese, "Self-synchronization phenomena in computer networks," MIT class project, Dec. 1992.

[30] ——, private communication, 1994.

[31] VMS Networking Manual, Version 5.0, AA-LA48A-TE, Digital Equipment Corporation, Maynard MA, Apr. 1988.

[32] L. Zhang and D. Clark, "Oscillating behavior of network traffic: A case study simulation," *Internetworking: Research and Experience*," vol. 1 no. 2, pp. 101-112, 1990.



Sally Floyd (S'86–M'89) received the B.A., M.S., and Ph.D. degrees from the University of California at Berkeley, the undergraduate degree in sociology and mathematics in 1971, was followed by classes in electronics and then by nearly a decade of work on computer systems for Bay Area Rapid Transit (BART). She received the graduate degrees from 1987 and 1989, in theoretical computer science.

She is a research scientist at Lawrence Berkeley Laboratory.

Van Jacobson photograph and biography not available at time of publication.