The Synchronization of Periodic Routing Messages

Sally Floyd and Van Jacobson,*
Lawrence Berkeley Laboratory,
One Cyclotron Road, Berkeley CA 94720,
floyd@ee.lbl.gov, van@ee.lbl.gov

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. We give examples of the harmful effect of these synchronized updates on other network traffic, 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.

1 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. E.g., 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 [Pa93a] [SaAgGuJa92] [Ja92] [BrChClPo93] 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 [Bl88]. 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 [Bl88], 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 [En92]. 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 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

*This work was supported by the Director, Office of Energy Research, Scientific Computing Staff, of the U.S. Department of Energy under Contract No. DE-AC03-76SF00098.

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 0-89791-619-0/93/0009/0033...$1.50

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

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 [De93] 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 [ZhCl90]. 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[FJ92]. (This randomization has the advantage of avoiding other unintended phase effects as well.)

• **Synchronization to an external clock.** Two processes can become synchronized with each other simply by both being synchronized to an external clock. For example, [Pa93a] shows DECnet traffic peaks on the hour and half-hour intervals; [Pa93b] 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 [Ba92].

• **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.

In addition, the specific problem of synchronized routing messages is of considerable practical interest. Synchronized routing messages put an unnecessary load on the network, and cause unnecessary delays and dropped packets. (See Section 2.)

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 [Pa93a], but realtime 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.

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.

2 Measurements suggesting synchronization

This section presents measurements of Internet traffic that demonstrate the harmful consequences of synchronized routing messages. We began this investigation in 1988 after observing synchronized routing messages from DECnet's DNA Phase IV on our local Ethernet. On this network 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 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. Figure 1 shows a particular run of a thousand pings from Berkeley to MIT, made at 3:30 AM on Wednesday, May 27, 1992; the x-axis shows the ping number and the y-axis shows the roundtrip time. Dropped packets are represented by a negative roundtrip time. Figure 2 shows the autocorrelation function for the roundtrip times in Figure 1, where the dropped packets are assigned a roundtrip time of two seconds (higher than the largest roundtrip time in the experiment). The high autocorrelation for roundtrip times separated by 89 pings (roughly 90 seconds, because pings were sent at 1.01-second intervals) reflects the fact that at 90-second intervals several successive pings would be dropped. Further runs of pings to intermediate locations 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 [SaAgGuJa92].

These packet drops were reported by the network manager to be caused by synchronized IGRP (the Inter-Gateway Routing Protocol [He91]) routing updates at the NEARnet routers [Sc92]. The routers were prevented from routing other packets while the synchronized 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. Nevertheless, the underlying problem of synchronized routing updates, along with the unnecessarily-heavy load that these synchronized updates can place on a network, still remains, on NEARnet and on other networks.

Synchronized routing updates have been demonstrated with RIP as well as with DECnet and IGRP. Figure 3 shows audio packet losses during a December 1992 Packet Video audiocast workshop[2] [Ja92]. 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 (the Routing Information Protocol [He88]), these long intervals of packet losses are conjectured to result from synchronized RIP routing updates. In other instances periodic 30-second audio

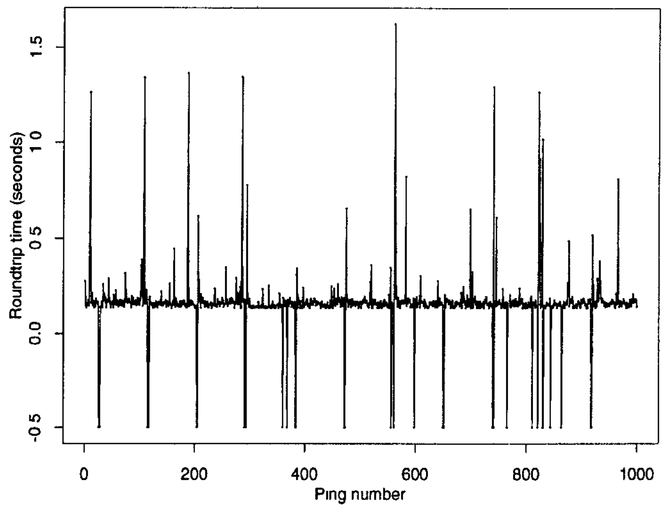

Figure 1: Periodic packet losses from synchronized IGRP routing messages.

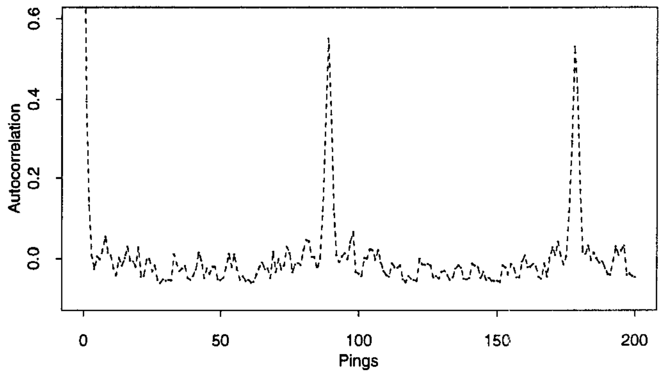

Figure 2: The autocorrelation of roundtrip times.

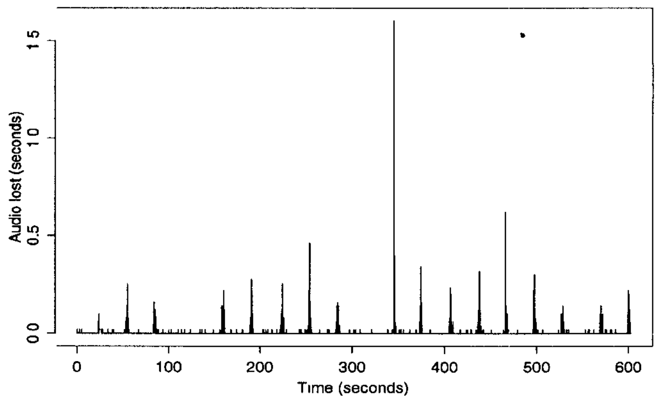

Figure 3: Periodic packet losses from (conjectured) synchronized RIP routing messages.

[2]For a report on the first such audiocast, see [Ca92].

packet losses have been conclusively traced to synchronized RIP routing updates [De93].

3 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 (the DIGITAL Network Architecture) [VMS88], but other routing protocols that can conform to this model include EGP (Exterior Gateway Protocol) [M84], Hello [Mi83], 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.

2. If the router receives an incoming routing message while preparing its own outgoing routing message, the router also processes the 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 assumes that the transmission time for routing messages is zero. The 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 TCP/IP internal gateway 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 our system, 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 [He88] 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, 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 [Li93].

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.

4 Simulations

This section describes simulations of the Periodic Messages model. These simulations show the behavior of a network with N routing nodes, 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: $\mathbf{T_p}$, $\mathbf{T_r}$, and $\mathbf{T_c}$. 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 each round is drawn 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 incoming or outgoing routing message. □

For the simulations in this section, T_p is 121 seconds and T_c is 0.11 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 to model an estimated computation time of 0.1 seconds and transmission time of 0.01 seconds a router to compute and

[3]BGP (Border Gateway Protocol), which is also used, only requires routers to send incremental update messages.

transmit an outgoing routing message after a timer expiration; these values are not based on any measurements of actual networks. Section 5.3 discusses how the results scale with different values for the various parameters.

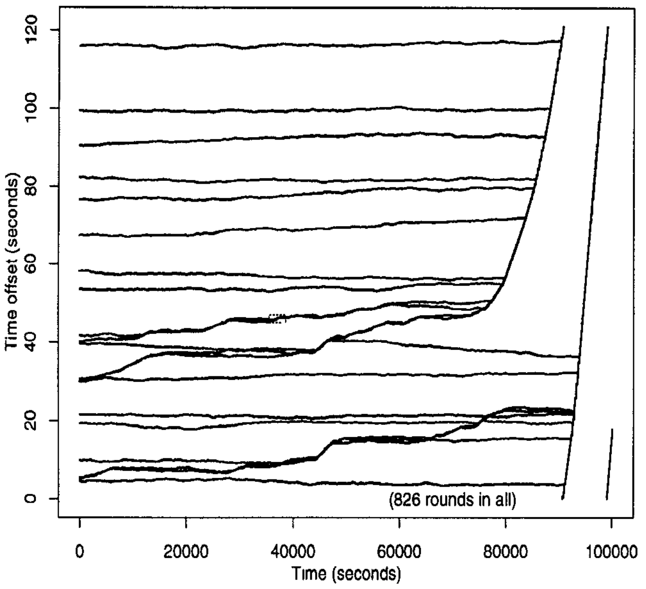

Figure 4: A simulation showing synchronized routing messages

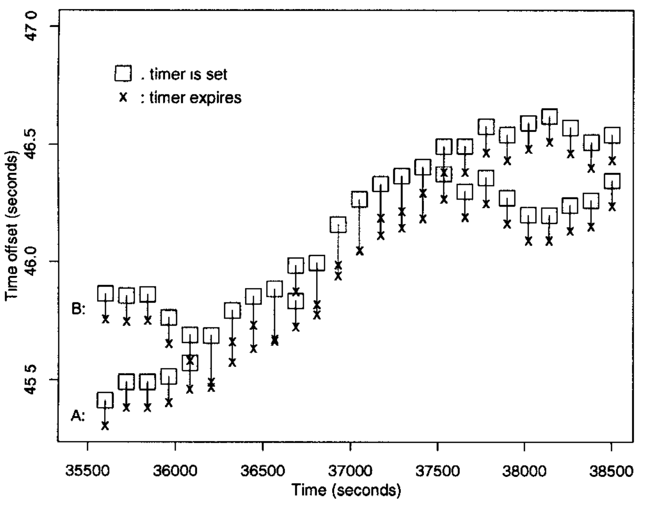

Figure 5: An enlargement of the simulation above.

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_c seconds required to process an incoming routing message.

For simplicity, when node A's timer expires we assume that the other nodes are immediately notified that node A will be sending a 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_c seconds receiving and processing the routing message from node A. (This assumption most plausibly reflects a network in which a router's routing message consists of several packets transmitted over a T_c-second period.)

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 transit time for the first routing message is chosen from the uniform distribution on $[0, T_p]$ seconds. For the simulation in Figure 4, T_r is set to 0.1 seconds. Each jittery line in Figure 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 that the routing message was sent, and the y-axis shows the *time-offset*, the time mod 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 Figure 4 begins with unsynchronized routing messages and ends with the N=20 routing messages transmitted 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.

Figure 5 is an enlargement of a small section of Figure 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 Figure 5 the two nodes are independent, and each node sets its timer exactly T_c seconds after its previous timer

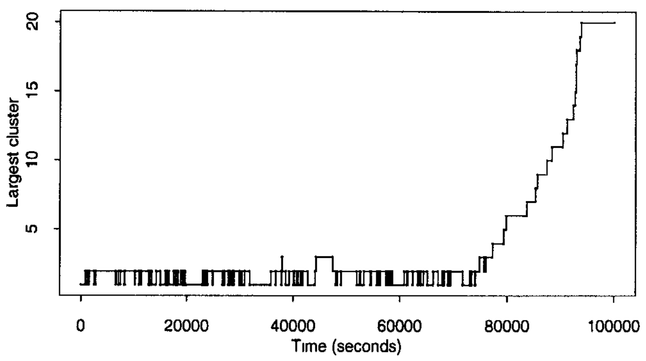

Figure 6: The cluster graph, showing the largest cluster for each round.

expires. However, in the sixth round, node A's timer expires, say, at time t, and node A begins transmitting 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 $2T_c$ seconds, so node A resets its timer at time $t + 2T_c$.

In our model node B also begins processing node A's routing message at time t. (Recall that in the simulations, node B is notified immediately when node A's timer expires.) While node B is receiving and processing node A's routing message, node B's own timer expires; node B has to prepare its own outgoing routing message *and* finish processing node A's routing message before resetting its timer. These tasks take $2T_c$ seconds, so node B also resets its timer at time $t + 2T_c$. 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 the same time, as long as the timers expire within T_c 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 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. The i nodes in a cluster reset their timers at the same time.

One way to think of the simulation in Figure 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 Figure 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, 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 Figure 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 Figure 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 seconds before any of the other nodes in the cluster, because of the random component in the timer interval for each node. The break-up of a cluster can be seen in Figure 5 where a cluster of size two forms and then breaks up again.

The first part of the simulation in Figure 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. A simulation at any point in time can be partially characterized by the size of the largest cluster of routing messages. Figure 6 shows a cluster-graph of the simulation in Figure 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.

Figure 7 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 Figure 7 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 simulation results are consistent with simulations of the same model in [Tr92].

These simulations do not specifically include triggered 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 Figure 8; 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.

5 The Markov chain model

This section uses a Markov chain model to further explore the behavior of the Periodic Messages system. 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 and of the Periodic Messages model.

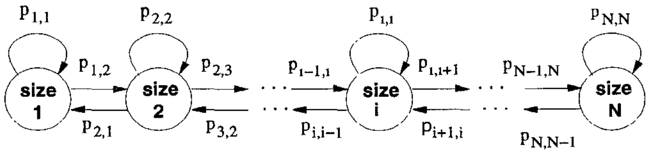

Figure 9: The Markov chain.

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. Figure 9 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

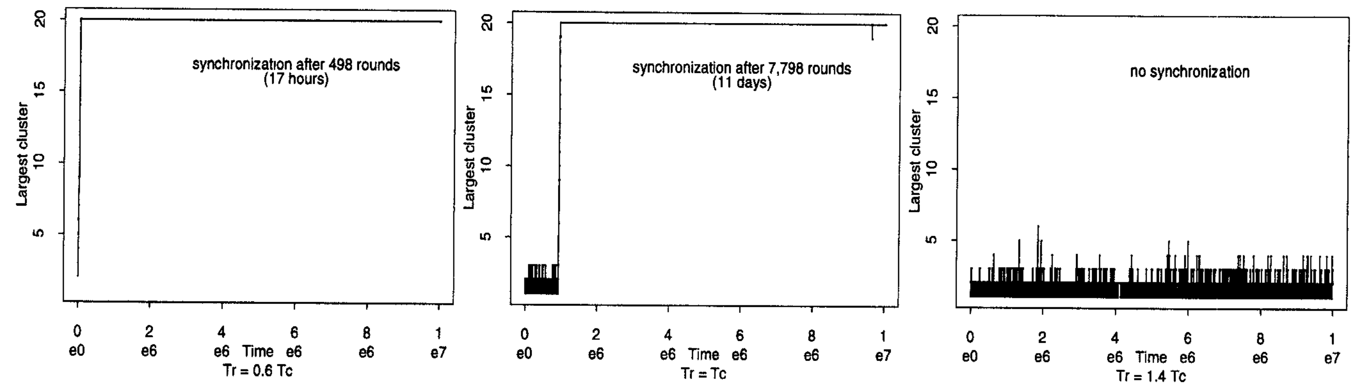

Figure 7: Simulations starting with unsynchronized updates, for different values for T_r.

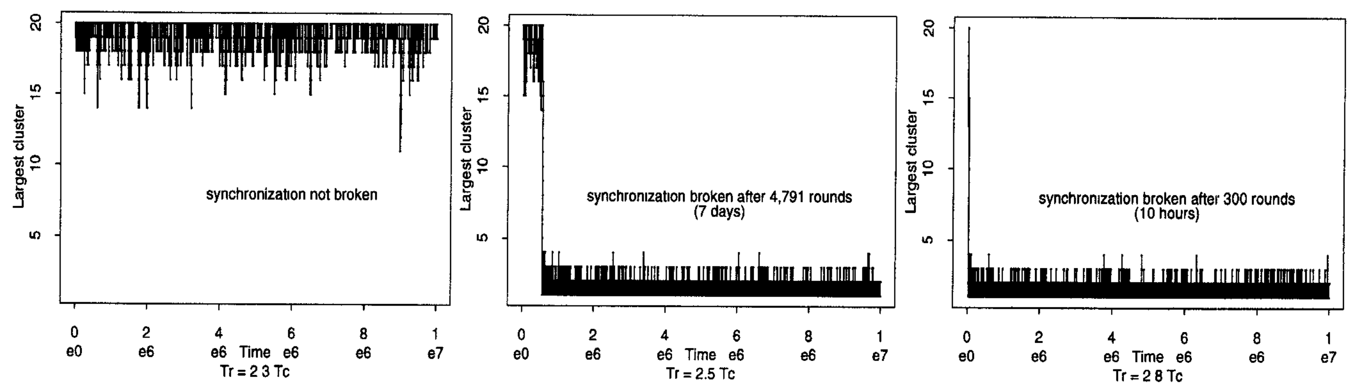

Figure 8: Simulations starting with synchronized updates, for different values for T_r.

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 all 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; 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 iT_c seconds to process the incoming and outgoing routing messages in the cluster. In this section we assume that $T_r > T_c/2$; 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.

5.1 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, 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, transmits its routing message, and the next timer in the cluster expires more than T_c seconds later. In this case, node A resets its timer before it receives any routing messages from any of the other $i-1$ nodes. Because we assume that $T_c < 2T_r$, there is always a nonzero probability that a cluster of size i breaks up into smaller clusters.

For i nodes in a cluster, 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. From [Fe66, p.22],

$$p_{i,i-1} = Prob.(L > T_c) = \left(1 - \frac{T_c}{2T_r}\right)^i \tag{1}$$

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.

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 that 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 iT_c seconds after the first timer in the cluster expires. 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 $1/(i+1)$ [Fe66, p.24]. Thus the first of the i timers expires, on average, $T_p - T_r(i-1)/(i+1)$ 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

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

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

5.2 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 the next state after state i is 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 $f(i)$. 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 $f(i)$:

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

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

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

Equation 3 has the solution [4]:

$$f(i) = f(2)\left(1 + \sum_{m=3}^{i}\left(\prod_{j=2}^{m-1}\frac{p_{j,j-1}}{p_{j,j+1}}\right)\right)$$
$$+ \sum_{m=3}^{i}\sum_{k=3}^{m}\left(c(k)\prod_{j=k}^{m-1}\frac{p_{j,j-1}}{p_{j,j+1}}\right). \tag{4}$$

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 $j+1$. The equation for $t_{j,j+1}$ is as follows [FJ93]:

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

[4]The derivation for this equation is given in [FJ93]. This solution could also be verified by the reader by substituting the right-hand side of Equation (4) into Equation (3).

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

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

Equation 5 has the solution below:

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

The derivation of this equation is similar to that of $f(i)$, given in [FJ93]. Note that this equation does not depend on the values of $p_{1,2}$ or of $f(2)$.

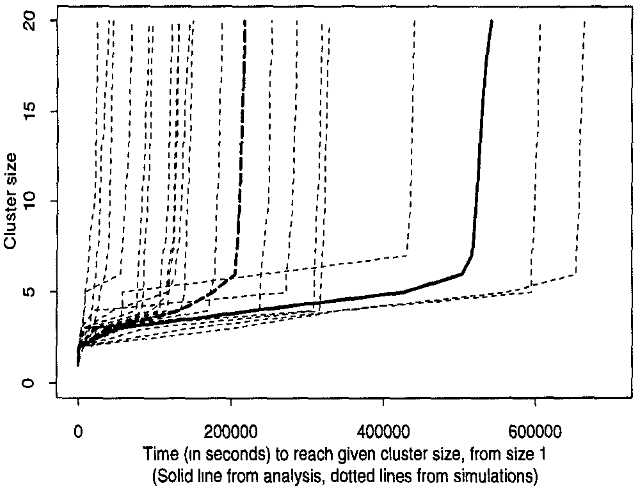

Figure 10: The expected time to reach cluster size i, starting from cluster size 1, for $T_r = 0.1$ seconds.

The solid line in Figure 10 shows $f(i)$, computed from Equation 4, for $N = 20$, $T_p = 121$ seconds, $T_c = 0.11$ seconds, $T_r = 0.1$ 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.) The x-axis shows the time in seconds, computed as $(T_p + T_c)f(i)$. The y-axis shows

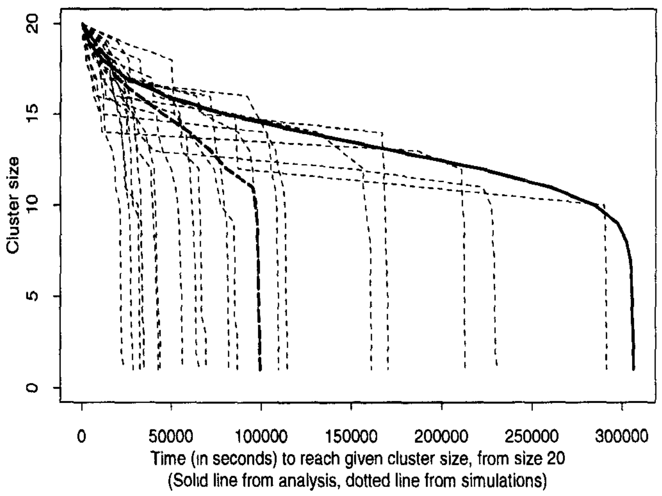

Figure 11: The expected time to reach cluster size i, starting from cluster size N, for $T_r = 0.3$ seconds.

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, and T_r described above; these simulations differ only in the random seed. The heavy dashed line shows the results averaged from twenty simulations.

The solid line in Figure 11 shows $g(i)$, computed from Equation 6, for the same parameters for N, T_p, and T_c as in Figure 10, and for $T_r = 0.30$ seconds; for the value of T_r in Figure 10, the system takes a long time to unsynchronize, making simulations unrealistic. The heavy dotted line averages the results from twenty simulations.

Figures 10 and 11 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.

5.3 Results from the Markov chain model

This section explores the general behavior of the Markov chain as a function of the parameter T_r. Figure 12 gives $f(N)$, from Equation 4, and $g(1)$, from Equation 6, for T_r ranging from zero to $4.5T_c$, given $N = 20$, $T_p = 121$ seconds, and $T_c = 0.11$ seconds. The solid line shows the expected time for the Markov chain to move from state N to state 1 and the dashed line shows the expected time for the Markov chain to move from state 1 to state N. The dashed line was computed using values for $f(2)$ based on an approximate analysis that is not given here; the dotted line shows the expected time for the Markov chain to move from state 1 to state N computed using $f(2)$ set to zero. Note that the y-axis is on a log scale, and ranges from less than 10^4 seconds (less than three hours) up to 10^{12} seconds (over 32 thousand years).

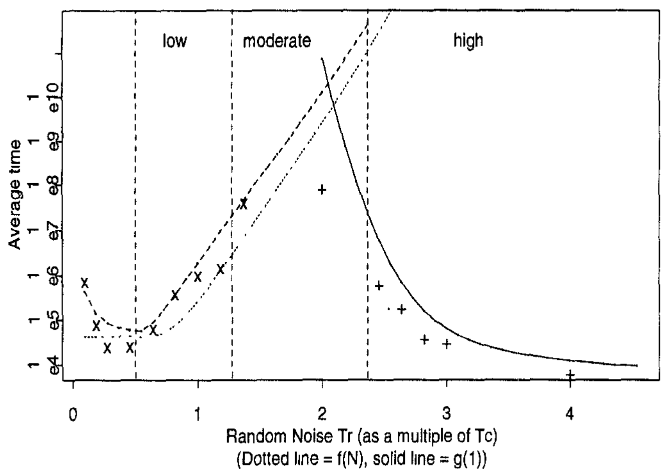

Figure 12: Expected time to go from cluster size 1 to cluster size N, and vice versa, as a function of T_r.

Figure 12 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 $f(N)$ grows exponentially with T_r. The "X" marks on Figure 12 show simulations that start with unsynchronized routing messages and the "+" marks show simulations that start with synchronized routing messages.

Figure 13 shows the same analytical results as in Figure 12 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 verify that the analytical results predict the simulation results reasonably accurately for a range of parameters. Because of time constraints, we ran only one simulation for most sets of parameters. We ran two simulations for each set of parameters for $T_r < 0.5T_c$; for these parameters the time to synchronize depends largely on the time to first form a cluster of size two, which can be quite different from one simulation to the next. Simulations with $T_c = 0.5$ are discussed in [FJ93].

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

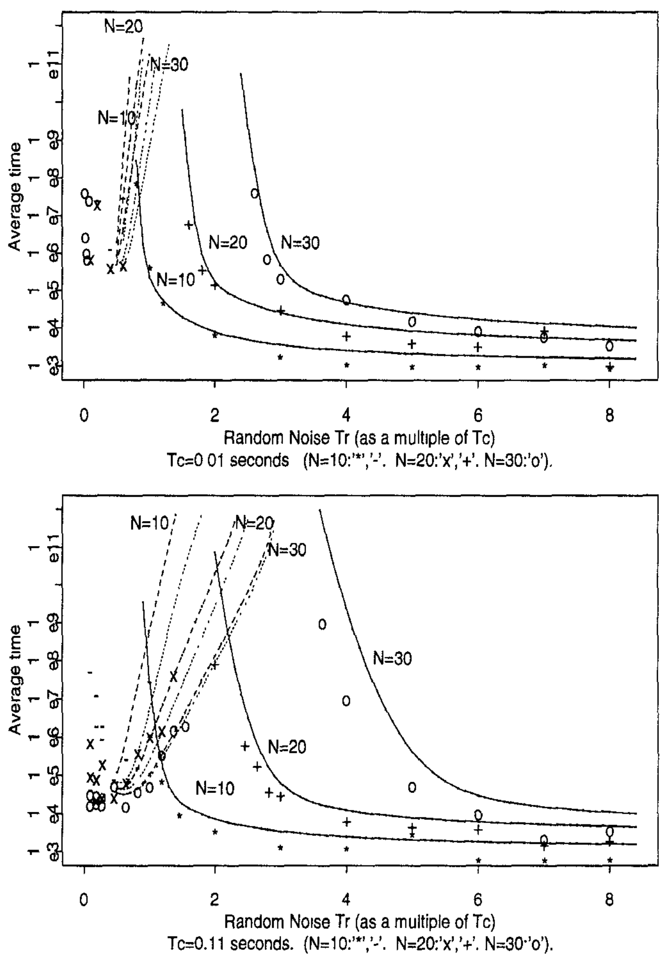

Figure 13: Expected time to go from cluster size 1 to cluster size N, and vice versa, as a function of N and of T_r.

from the uniform distribution on the interval $[0.5T_p, 1.5T_p]$ seconds. This introduces a high degree of randomization into the system, yet ensures that the interval between routing messages is never too small or too large.

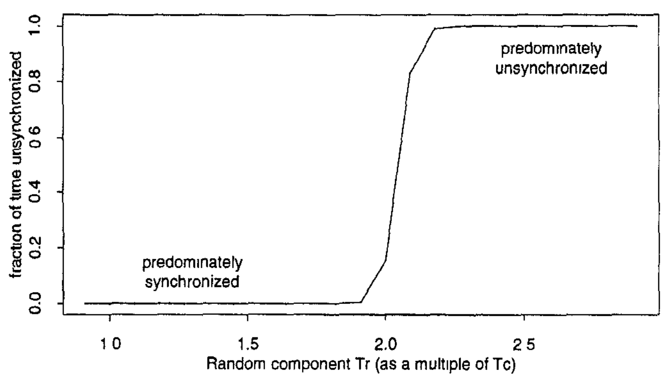

Figure 14: The fraction of time unsynchronized, as a function of the random component T_r.

One quantity of interest is the fraction of time that the

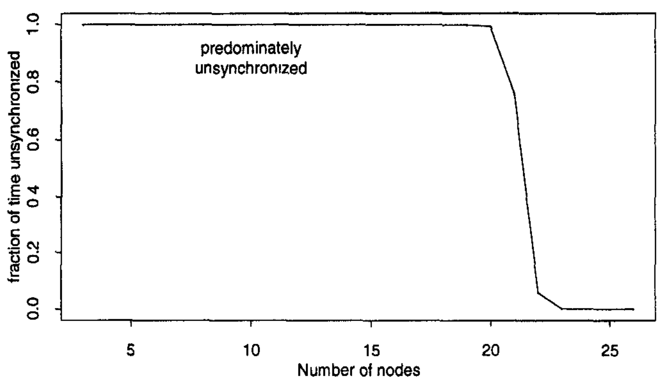

Figure 15: The fraction of time unsynchronized, as a function of the number of nodes.

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 unsynchronized states is to compute $f(N)/(f(N) + g(1))$. 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 Figure 14 the x-axis shows T_r; the other parameters are $N = 20$, $T_p = 121$ seconds, and $T_c = 0.11$ seconds. The y-axis is $f(N)/(f(N) + g(1))$, the estimated fraction of time for which the system is unsynchronized. As Figure 14 shows, as T_r is increased, the system makes a sharp transition from predominately-synchronized to predominately-unsynchronized.

Figure 15 shows the estimated fraction of time that the Markov chain spends with low cluster sizes as a function of the number N of nodes in the network. The parameters for this figure are $T_p = 121$ seconds, $T_c = 0.11$ seconds, and $T_r = 0.3$ seconds. 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.

6 Conclusions

As the simulations and analysis in this paper demonstrate, periodic routing messages from a system of routers in a network can easily become synchronized. When the random component of the routing timer intervals is low, as is probably the case in the current Internet, an initially-unsynchronized network can become synchronized fairly quickly. If nodes add a sufficient random component to the routing timers, this synchronization can be avoided, and routing messages that become synchronized as a result of triggered updates can quickly return to an unsynchronized state. The analysis provides general guidelines on determining the magnitude of the random component necessary to avoid synchronization.

Adding a random component to the routing timers is not the only possible method for discouraging the synchronization of routing updates. As is suggested in the specifications for RIP, another possibility is to implement a routing timer where the time between successive routing messages is not affected by the time required to service previous timer expirations [He88, p.23]. Thus, 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 could be avoided. There are, however, some drawbacks with this proposal. If all routers have the same default period, then routers that are initially synchronized (either by chance, or because routers were restarted at the same time) will remain synchronized; there is no mechanism to break up synchronization if it does occur.

Some methods for discouraging synchronization that might be technically feasible are unattractive because they require intervention from the system administrator. This would include having the system administrator manually set the routing update intervals for different routers on the same network to different values. For networks with a small number of routers, an alternate strategy might be to set the routing update interval at each router to a different random value. The consequences of having a slightly-different fixed period for each router would require further investigation.

There are also approaches that reduce the negative impact of synchronized routing updates. This includes the immediate response to the synchronized routing updates on the NEARnet system mentioned in Section 3, which involved modifying routers so that they can give acceptable performance even in the presence of synchronized routing updates. Nevertheless, synchronized routing updates place an unnecessary burden on the network, even assuming that routers are modified to improve their performance in the presence of synchronized routing updates. The preferable solution is to avoid synchronized routing updates in the first place.

The method of adding a random component to the routing timer is attractive because it provides a prompt break-up of synchronization, including the synchronization that could result from triggered updates or from other forms of feedback than those described in the Periodic Messages model, and it does not require manual intervention from the system administrator. Each routing node would add a random time to the routing timer each time that the routing timer is set. Section 5 provides guidelines for choosing the random component. In particular, setting the timer each round to a time from the uniform distribution on the interval $[0.5T_p, 1.5T_p]$ seconds would be a simple way to avoid synchronized routing messages. Pseudo-random numbers could be stored in memory or could be efficiently generated by a random number generator [Ca90].

Periodic routing messages are not necessarily the only example of synchronized periodic messages in the Internet. Our "ping" experiments indicate that many periodic processes are at work in the Internet, in addition to the (now-corrected) every-90-second packet drops on paths to MIT. More work is needed in exploring the effects of synchronization in the Internet. The tendency towards synchronization is likely to become more relevant in the future, as the Internet carries more traffic with a strong periodic structure.

7 Acknowledgements

We thank Tony Li, Steven McCanne, Vern Paxson and Lixia Zhang for their comments on earlier drafts of this paper; we thank Lixia Zhang for encouraging us to submit this paper in time for SIGCOMM. Much of the simulation and analysis in this paper was essentially completed some 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, mentioning long delays at periodic 90-second intervals on paths to MIT, we reported that we found a similar problem of packet drops at 90-second intervals on paths to MIT. We thank Dave Clark for contacting Jeff Schiller about the problem, who reported back that the problem was related to synchronized IGRP routing messages in NEARnet. This documentation of the practical problems caused by synchronized routing messages pushed us to finish our draft of this paper.

References

[Ba92] Baker, M., private communication, 1992.

[Bl88] Blekhman, I.I., Synchronization in Science and Technology, ASME Press Translations, 1988.

[BrChClPo93] Braun, H., Chinoy, B., Claffy, K.. and Polyzos, G., "Analysis and Modeling of Wide-Area Networks: Annual Status Report", CSL, University of California at San Diego, February 1993.

[Ca90] Carta, D., "Two Fast Implementations of the 'Minimal Standard' Random Number Generator", Commun. ACM, V.33 N.1, January 1990, p.87-88.

[Ca92] Casner, S., and Deering, S., "First IETF Internet Audiocast", Computer Communication Review, V.22 N.3, July 1992, p.92-97.

[De93] Deering, S., private communication, 1993.

[En92] Engel, E.M.R.A., A Road to Randomness in Physical Systems, Springer-Verlag, 1992.

[Fe66] Feller, W., *An Introduction to Probability Theory and Its Applications, V. II*, John Wiley & Sons, 1966.

[FJ92] Floyd, S., and Jacobson, V., *On Traffic Phase Effects in Packet-Switched Gateways*, Internetworking: Research and Experience, V.3 N.3, September 1992, p.115-156.

[FJ93] Floyd, S., and Jacobson, V., *On the Synchronization of Periodic Routing Messages*, LBL Tech Report (available by anonymous ftp from ftp.ee.lbl.gov: sync*.ps.Z), May 1993.

[He88] Hedrick, C., "Routing Information Protocol", Request For Comments (RFC) 1058, June 1988.

[He91] Hedrick, C. L., "An Introduction to IGRP", August 1991, available by anonymous ftp from shiva.com in /doc/igrp.ps.Z.

[Ja92] Jacobson, V., Audio losses during yesterday's packet video workshop audiocast, Message-ID 9212112039.AA24349@rx7.ee.lbl.gov archived in nic.es.net:ietf.rem-conf, Nov. 11, 1992.

[Li93] Li, T., private communication, 1993.

[Mi83] Mills, D.L., "DCN Local-Network Protocols", Request For Comments (RFC) 891, Dec. 1983.

[M84] Mills, D.L., "Exterior Gateway Protocol Formal Specification", RFC 904, April 1984.

[Pa93a] Paxson, V., "Empirically-Derived Analytic Models of Wide Area TCP Connections", LBL Tech. Report LBL-34086 , May 1993.

[Pa93b] Paxson, V., "Growth Trends in Wide-Area TCP Connections", submitted to IEEE Network, 1993.

[SaAgGuJa92] Sanghi, D., Agrawala, A., Gudmundsson, O., and Jain, B., "Experimental Assessment of End-to-End Behavior on Internet", University of Maryland Report UMIACS-TR-92-62, June 1992.

[Sc92] Schiller, J., private communication, Sept. 1992.

[Tr92] Treese, W., "Self-Synchronization Phenomena in Computer Networks", MIT class project, Dec. 1992.

[VMS88] VMS Networking Manual, Version 5.0, AA-LA48A-TE, Digital Equipment Corporation, Maynard Massachusetts, April 1988.

[ZhCl90] Zhang, L., and Clark, D., "Oscillating Behavior of Network Traffic: A Case Study Simulation", Internetworking: Research and Experience, Vol. 1 N.2, 1990, pp. 101-112.