

# Can coexisting overlays inadvertently step on each other?

Ram Keralapura\*, Chen-Nee Chuah\*  
University of California, Davis

Nina Taft, Gianluca Iannaccone  
Intel Research Labs

## Abstract

*By allowing end hosts to make routing decisions at the application level, different overlay networks may unintentionally interfere with each other. This paper describes how multiple similar or dissimilar overlay networks making independent routing decisions could experience race conditions, resulting in oscillations in both route selection and network load. We pinpoint the causes for synchronization in terms of partially overlapping routes and periodic path probing processes and derive an analytic formulation for the synchronization probability of two overlays. Our model indicates that the probability of synchronization is non-negligible across a wide range of parameter settings, thus implying that the ill-effects of synchronization should not be ignored. Using the analytical model, we find an upper bound on the duration of traffic oscillations. We validate our model through simulations that are designed to capture the transient routing behavior of both the IP- and overlay-layers. We use our model to study the effects of factors such as path diversity (measured in round trip times) and probing aggressiveness on these race conditions. Finally, we discuss the implications of our study on the design of overlay networks and the choice of their path probing parameters.*

## 1 Introduction

Application-layer overlay networks are becoming very popular due to the fact that they can often offer better services catered to different applications than the traditional IP networks. This concept has been exploited in building content delivery networks like Akamai [1], resilient networks like RON [2], multicast services like SplitStream [3], and distributed hash table services like Bamboo [14], among others. All of these networks have multiple nodes that collaborate with each other at the application layer to provide features that are not readily supported by IP layer routing services. For example, RON [2] and Detour [15] demonstrate that end-to-end route selection can often find better

alternative paths by relaying traffic among overlay nodes.

Numerous of these overlays are being deployed over the Internet and the volume of traffic that they carry is increasing [16]. Since most overlay networks are designed independently with different target applications in mind, our suspicion is that as overlay traffic load increases, different overlays may unintentionally interfere with each other. It is therefore very important to examine the impacts of the co-existence of multiple overlay networks when their traffic represents a significant, if not dominant, portion of the total traffic.

Based on our previous work in [8], we arrive at a hypothesis that two (or more) overlays can experience race conditions and become synchronized leading to route and traffic oscillations. This hypothesis is formulated based on two key observations. First, in the work by Floyd et al [4], the authors discuss how there are many examples of seemingly independent periodic processes in the Internet that can inadvertently become synchronized. They warned that the phenomenon of inadvertent synchronization of period processes would most likely become an increasing problem in computer networks. Typically overlay networks use a periodic probing process to detect events that deteriorate path performance, and to identify alternate paths between pairs of source and destination overlay nodes. IP layer events such as failures can trigger an overlay network to detect a performance problem and move its traffic to an alternate path. The work in [4] would suggest that a situation in which two different overlays, both using *periodic* probing, that can react to the same IP layer triggering event, is a candidate scenario for the synchronization problem. A second motivation for suspecting that synchronization might arise comes from control theory. Different overlays are simultaneously and independently conducting routing control at the application layer. This corresponds to a situation in which multiple independent control loops coexist, yet react to the same events (e.g, failures). This is a classic situation for race conditions.

This paper seeks to explore this hypothesis and to gain a better understanding on the likelihood of oscillations, how long they can last, and the conditions under which they occur. The contributions of this work are:

---

\* The authors were supported by NSF CAREER Grant No.0238348

- We pinpoint the reasons for oscillations in terms of partially overlapping paths and periodic probing process (Section 3).
- We develop an analytical method to compute the probability of synchronization between two overlay networks as a function of the path probing parameters. We also provide an upper bound on the number of oscillations that the two overlays will experience after they are synchronized, in the absence of external events acting as stop triggers (Section 4).
- We validate our hypothesis and analytical model using simulations that are carefully designed to capture the IP-layer transient routing dynamics and the generic properties of overlay routing strategies. We do indeed see a variety of scenarios in which oscillations happen when the overlay traffic is a significant portion of the total traffic. The scenarios differ in the length of oscillations, the number of overlays involved, and in the manner by which oscillations are eventually stopped. The simulation results closely match the predictions from our analytical model both in terms of probability of synchronization and the number of oscillations (Section 5).
- We show that the synchronization probability is not negligible for a wide range of parameters, suggesting that particular care must be given by network designers to the configuration of overlay routing protocols. We illustrate that oscillations can occur even for two overlays that deploy considerably dissimilar path probing parameters. We study the impact of path diversity and probing aggressiveness on the probability of synchronization (Section 6).

We summarize the implications of this study on the design of overlay routing strategies in Section 7 and discuss some future directions in Section 8.

## 2 Related Work

Interactions between multiple co-existing overlay networks were first addressed by Qiu et al. [13], where the authors investigate the performance of selfish routing after the system reaches the Nash equilibrium point (when network-level routing is static). They also show that selfish routing can achieve optimal average latency at the cost of overloading certain links. Liu et al. [10] model the interaction between overlay routing and IP traffic engineering as a two-player game, where the overlay attempts to minimize its delay and IP traffic engineering tries to minimize network cost. In our current work, we focus instead on dynamics of the overlay routing layer before the system reaches the equilibrium. Instead of static network-layer routing, we consider events such as link/router failures, flash crowds and network congestions that lead to dynamic re-computation of routes in overlay applications and/or IGP protocols.

Other works have studied the overlay network probing

process, a crucial component of overlay routing. Nakao et al. [12] proposed a shared routing underlay that exposes large-scale, coarse-grained static information (e.g., topology and path characteristics) to overlay services through a set of queries. They advocate that the underlay must take cost (in terms of network probes) into account and be layered so that specialized routing services can be built from a set of basic primitives. However, sharing network-layer path information may induce synchronized routing decisions in overlay networks and unintentionally lead to route/traffic oscillations, an aspect not addressed in [12]. We hope to shed some light on this problem through our modeling of overlay and IP-layer dynamics in response to failures.

While our work was in part inspired by the work in [4], the particular periodic process we focus on is different from the one considered in [4]. They focused on routing protocols such as EGP, IGRP and RIP that send a periodic update message to ensure routing tables are kept up to date. The process we explore uses two types of periodic probes, and only reacts to external triggers. Also, they study the scenario of many routers participating in the same protocol, whereas as we consider two different instances of the protocol, with non-identical parameters that only partially overlap in the underlying physical network.

## 3 Why do race conditions occur?

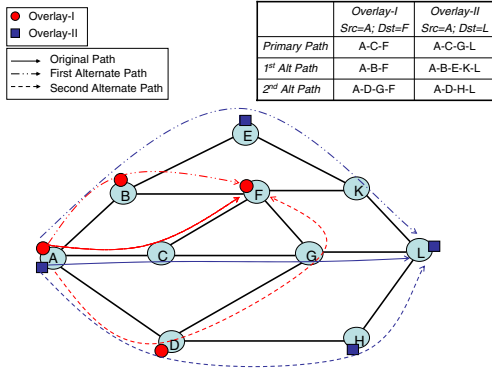
Before we present the arguments for why and when race-conditions (and hence traffic oscillations) between multiple co-existing overlays occur, we describe how we model the overlay networks in both our analysis and simulation.

### 3.1 Modeling Overlay Networks

While different overlay networks, designed for a wide range of applications, may differ in their implementation details (like choice of topologies or performance goals), most of them provide a common set of functionalities, including periodic path/performance monitoring, failure detection and restoration. In this paper, we attempt to model the most generic properties of an overlay network, as summarized below:

- Most overlay routing strategies select a path between a source-destination pair with the best performance based on end-to-end delay, throughput, and/or packet loss. Our model assumes the overlay path with the shortest end-to-end delays will be selected, but can be extended to include other metrics.
- Most overlay networks monitor the actively used paths by sending frequent probes to check if the paths adhere to acceptable performance bounds. If the probing event detects a problematic path (due to failures, congestion, etc. at the IP-layer), then the overlay network sends probes at a

higher rate to confirm the problem before selecting an alternate path. Our model assumes regular probes are sent out every  $P$  seconds. If a probe does not receive a response within a given *timeout* (or  $T$ ) value, then the path is probed at a higher rate (every  $Q$  seconds). If a path remains bad after  $N$  such high frequency probes, the overlay will find an alternate path (or the next best path) between the source and destination nodes. For instance, RON [2] can be modeled with  $P=12s$ ,  $Q=3s$ , and  $N=3$  while Akamai network can be modeled with much smaller values of  $P$ ,  $Q$ , and  $N$  [9]. As soon as an alternate path is found, the traffic is moved to the alternate path, which is now probed every  $P$  seconds to ensure that it is healthy<sup>1</sup>.



**Figure 1. Two overlay networks that partially share primary and alternate paths**

### 3.2 Conditions for Traffic Oscillations

Oscillations are initiated when coexisting overlays satisfy the following conditions:

**Path Performance Degradation.** An event must trigger a perturbation of the network state that leads overlay networks to revisit their routing decisions and look for alternate paths. This event can be an increase in the traffic demand of the IP network or one of the overlay networks, or a link failure event that results in a reduction of capacity. Since different overlays are controlled by autonomous timers and routing algorithms, a path performance degradation event could provoke independent reactions from different overlays.

**Topology (i.e., Primary and Backup Paths).** The node locations determine how the paths of coexisting overlay networks overlap. Oscillations may occur when the primary and alternate paths share at least one common link. Figure 1 illustrates this case with overlays on top on an IP network. The node pair  $A - F$  in *Overlay-I*, and pair  $A - L$

<sup>1</sup>As long as the current path adheres to the performance bounds, an overlay does not shift traffic to an alternate path even if the alternate path starts to exhibit better performance.

in *Overlay-II* share the link  $A - C$  on their primary paths. Assume, for simplicity of discussion, that the “top” path is their first alternate choice. If link  $A - C$  fails, then the first alternate path for  $A - F$  and that for  $A - L$  would share link  $A - B$ . If this link becomes a bottleneck, forcing the overlay networks to move their traffic again, then the overlay source-destination pairs  $A - F$  and  $A - L$  would now move to the “bottom” path. However, they would still share link  $A - D$  that could itself become a bottleneck. Hence the topology criteria for synchronization to occur is: there is a pair of overlay nodes in each of two different overlay networks, such that the two primary paths share at least one common bottleneck link. This condition is intuitive. Two overlays will not get synchronized if they do not share portions of physical paths. For oscillations to sustain, the two overlays must share bottleneck link/s in both their first and second alternate path choices.

**Probing Parameters.** As mentioned in Section 3.1, overlays periodically probe their paths. Consider the two overlays in Figure 1 and a failure on link  $A - C$  that is common to their primary paths ( $A - F$  in *Overlay-I* and  $A - L$  in *Overlay-II*). Suppose the timing of the probing processes for two overlays is such that the last high frequency probes, for each of the two overlays, expire within a short time window of the other. Then both overlays will shift their traffic to their first choice alternate path roughly at the same time. When this occurs we say that two overlays get *synchronized*. This happens when the window of time is so short that the overlay that moves second does not have time to re-probe its path to realize that some traffic load from the other overlay has already moved. Now, if the traffic load on the first choice alternate path becomes high, then the overlays could react again moving their traffic to the second choice alternate path. Such reactions can continue and overlays move their traffic in a lock-step fashion between the two alternate paths until the distance between the probes grows large enough to end the synchronization. When this happens we say that the two overlays *disentangle* themselves.

Since overlay networks have no control over performance degradation events inside an ISP (first condition), and since they may not have much control over the placement of overlay nodes that eventually determines the overlap of paths (second condition), we focus our calculation of the probability of synchronization in terms of just the probing process parameters.

## 4 Analyzing Oscillations

In this section, we focus on how the path probing parameters affect synchronization. First we develop an analytic formulation of the probability of synchronization for two overlays as a function of the parameters in the path probing procedure. Second we derive an upper bound on how long

two overlays can remain synchronized, for a given set of parameters.

#### 4.1 Probability of Synchronization

For our analysis, we assume the first two conditions for synchronization hold, i.e., the two overlays share at least one link on their primary paths and one event on the shared links occurs (e.g., failure or congestion) that causes probe packets to be lost or excessively delayed.

As described in Section 3.1, overlay networks probe their paths at regular intervals of  $P$  seconds. If the path is healthy, the probe should return in one round trip time, with a measure of the path delay (or an assessment of another chosen metric). If the probe does not return before the timeout  $T$  expires, then the overlay starts sending its high-frequency probes ( $N$  will be sent) every  $Q$  seconds. Thus, the probing procedure for each overlay  $i$  on path  $j$  is specified by five parameters: the probe interval  $P_i$ , the high frequency probe interval  $Q_i$ , the timeout  $T_i$ , the number of high frequency probes  $N_i$ , and the round trip time  $R_{ij}$  over path  $j$ . Note that  $T_i$  is the same for low- and high-frequency probes. By definition  $P_i \geq Q_i \geq T_i \geq R_{ij}$ .

The probing procedure implies that (under normal circumstances) on a given path there will be exactly one probe in every time period of length  $P$ . Now suppose that an event (e.g., a link failure) occurs at time  $t_l$ . We assume that a probe sent on path  $j$  in overlay  $i$  at time  $t_0$  “senses” the state of the path at  $t_0 + R_{ij}/2$ , i.e., the probe is dropped if the path is not operational at that time<sup>2</sup>. Hence, the overlay network will detect the failure event with the probe sent at  $t_0$  if  $t_0 \in [t_l - R_{ij}/2, t_l - R_{ij}/2 + P_i]$ . We call this period the *detection period*. The overlay will then react at time  $t_0 + T_i$  sending the high frequency probes as discussed above.

Consider two overlay networks,  $O_1$  and  $O_2$ . Let  $t_1$  and  $t_2$  be the actual times at which the initial probes are sent during the detection period. We assume that  $t_1$  and  $t_2$  are equally likely to occur anywhere in their detection period and hence are uniformly distributed in their detection period. Once an overlay network detects the failure, it begins sending the high frequency probes every  $Q_i$  time units. The final high frequency probe will be sent out at  $f_i = t_i + N_i Q_i$  for  $i = 1, 2$ .

An overlay network actually moves its traffic to an alternate path immediately after the final high frequency probe has timed out ( $f_i + T_i$  for  $O_i$ ). Two overlay networks will synchronize if they both move their traffic to the same bottleneck link (shared by their chosen alternate paths) in a “short” window of time. By “short” we mean here that the window is small enough that when one overlay moves, the

second overlay does not see or detect that move through its probing process and thus moves itself onto the same link(s).

There are two cases for synchronization - in one case  $O_1$  moves its traffic first and  $O_2$  moves shortly thereafter, or vice versa. Consider the case of  $O_1$  moving first. Suppose that  $O_2$  sends out the final high frequency probe after  $O_1$  sends its final high frequency probe, but before  $O_1$  moves its traffic. We assume that  $O_2$  decides at time  $f_2$  what its alternate path will be if this last probe does not return, and hence it doesn’t have time to detect the traffic move by  $O_1$  before it moves its own traffic. Hence if we have the timing  $f_1 < f_2$  and  $f_2 - f_1 < T_1$ , then both overlays move their traffic without being aware of the other’s reaction. This is the condition for synchronization when  $O_1$  moves first. Similarly, if  $O_2$  moves first, the networks will synchronize if  $f_2 < f_1$  and  $f_1 - f_2 < T_2$ . Hence the condition for synchronization of two overlays is:

$$\begin{aligned} -T_1 &< f_1 - f_2 < T_2 \\ -T_1 &< (t_1 + N_1 Q_1) - (t_2 + N_2 Q_2) < T_2 \\ b &< t_1 - t_2 < a \end{aligned} \quad (1)$$

where  $a = N_2 Q_2 - N_1 Q_1 + T_2$ ;  $b = N_2 Q_2 - N_1 Q_1 - T_1$

We assume that  $t_1$  and  $t_2$  can occur anywhere in their detection period with a uniform probability. It is important to notice that the actual value of  $t_l$  is irrelevant and hence for the ease of understanding we consider  $t_l = 0$ . Thus the range of  $t_1$  is  $[-R_1/2, P_1 - R_1/2]$  and the range of  $t_2$  is  $[-R_2/2, P_2 - R_2/2]$ , where  $R_1$  is the RTT for the primary path in overlay  $O_1$  and  $R_2$  is the RTT for the overlapping primary path in the other overlay<sup>3</sup>,  $O_2$ .

For a specific set of parameters ( $P_i, Q_i, T_i, N_i, R_i$ ) for  $i = 1, 2$ , we can represent the system as a two dimensional graph with the x-axis representing probe  $t_1$  and the y-axis representing probe  $t_2$ . All the allowable values for the tuple  $(t_1, t_2)$  lie inside the rectangle with the vertices:  $(-R_1/2, -R_2/2)$ ,  $(P_1 - R_1/2, -R_2/2)$ ,  $(P_1 - R_1/2, P_2 - R_2/2)$  and  $(-R_1/2, P_2 - R_2/2)$  (see Figure 3). This geometric representation allows us to compute the probability of synchronization,  $P(S)$ , of two overlay networks in an intuitively simple way. We define the *region of conflict* to be the portion of this rectangle in which synchronization will occur, i.e., the region that satisfies the two constraints specified in Equation 1. The boundaries of the *region of conflict* are thus determined by the boundaries of the rectangle and their intersection with the two parallel lines of slope 1:

$$\text{Line 1: } t_1 - t_2 = a \quad (2)$$

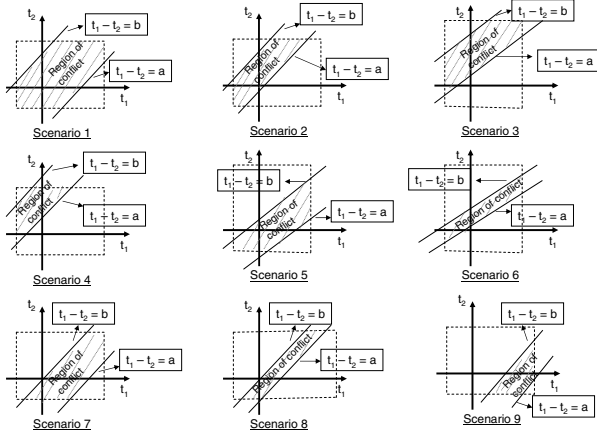
$$\text{Line 2: } t_1 - t_2 = b \quad (3)$$

Since  $t_1$  and  $t_2$  can occur anywhere in their detection period with uniform probability, synchronization will occur if

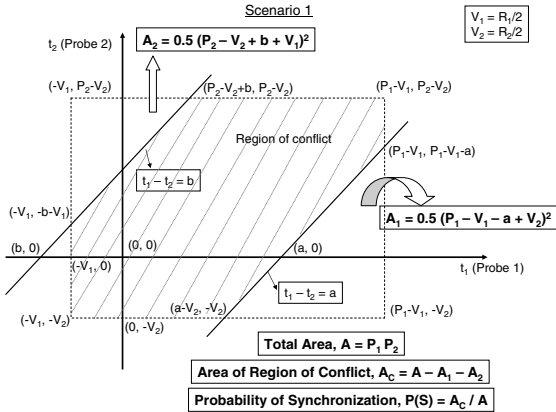
<sup>2</sup>To simplify our analysis during failures we ignore the exact values of propagation delays between the source, the failed spot, and destination. Thus we approximate the instant at which a probe is dropped by  $R_{ij}/2$ .

<sup>3</sup>Since we focus only on the primary path in both the overlays, we drop the second subscript in  $R_{ij}$ .

the point  $(t_1, t_2)$  lies inside the region of conflict. Now the probability of synchronization,  $P(S)$ , can be defined to be the ratio of the area of the region of conflict to the total area of the rectangle. This 2-dimensional representation captures the influence of all the parameters  $(P_i, Q_i, N_i, T_i, R_i)$  since these quantities ultimately define all the corners and line-intersection points needed to compute the relevant areas.



**Figure 2.** All possible scenarios to calculate the region of conflict



**Figure 3.** Scenario 1 ( $V_1 = R_1/2$  and  $V_2 = R_2/2$ )

There are a number of ways in which *Line-1* and *Line-2* intersect the boundaries of the rectangle. All possible scenarios that determine this area are shown in Figure 2. Consider Figure 3 that represents *Scenario 1* in detail. *Line-1* intersects the bottom and right edges, while *Line-2* intersects the left and top edges. As evident in Figure 3, we can clearly see that the area  $A$  of the rectangle is composed of three distinct regions:  $A_1$  (area of the region below *Line-1* and the rectangle boundaries),  $A_2$  (area of the region above *Line-2* and the rectangle boundaries) and the *region of conflict*. Hence the *region of conflict*,  $A_C$ , can be expressed as,

$$A_C = A - A_1 - A_2 \quad (4)$$

Thus we can express the probability of synchronization as

$$P(S) = \text{Probability}(b < t_1 - t_2 < a) = \frac{A_C}{A} \quad (5)$$

In *Scenario 1*,  $A_1$  and  $A_2$  are triangular regions with two equal edges (due to the fact that both *Line-1* and *Line-2* have a slope of 1). From Figure 3, we see that  $A_C$  can be computed using

$$A = P_1 P_2 \quad (6)$$

$$A_1 = 0.5(P_1 - R_1/2 - a + R_2/2)^2 \quad (7)$$

$$A_2 = 0.5(P_2 - R_2/2 + b + R_1/2)^2 \quad (8)$$

Even though the above equations for  $A_1$  and  $A_2$  are valid for *Scenario 1*, they do not hold for scenarios where *Line-1* intersects boundaries other than the right and bottom edges of the rectangle, and/or *Line-2* intersects boundaries other than the left and top edges of the rectangle. For example, if we consider *Scenario 2* in Figure 2, *Line-1* intersects the top and bottom edges of the rectangle. If we use Equation 7 to calculate  $A_1$ , then  $A_1$  is larger than it should actually be since it includes space outside the rectangle. We thus need to add another term,  $E$ , to  $A_C$  in Equation 4 to compensate for the *excess* included in  $A_1$ . In the case of *Scenario 2* (when *Line-1* intersects the top edge of the rectangle), the excess term to be removed is  $E = 0.5(P_1 - R_1/2 - a - P_2 + R_2/2)^2$ . We do not include the excess terms for each of the nine scenarios since it is a straightforward computation. Although this model results in 9 different scenarios, each of them with a different equation for  $P(S)$ , it is still attractive since it is conceptually very simple.

## 4.2 How long do oscillations last?

Suppose that two overlays react to an event within a short window of time, and land up on alternate paths that share at least one common link. As depicted in Figure 1, such a reaction by both overlays could overload the common link, prompting them to find another alternate path. These reactions lead to oscillations that last until the overlay networks disentangle themselves or are influenced by an external event (e.g., a shift in the traffic loads). If no external events stop the oscillations, then it is important to ask how long these oscillations will last. In this section, we derive an upper bound on the number of oscillations.

To break synchronization what matters is the temporal spacing between the two probing processes that govern their reaction times (moving traffic). Let  $s_0 = t_1 - t_2$  denote the difference in time between the initial detection of the path problem. Since we are concerned with the difference between the reaction times, we can map the  $\mathbb{R}^2$  region of

conflict space onto  $\mathbb{R}^1$  space on a real line for all possible scenarios. In other words, we can represent all the points in the *region of conflict* by the value of  $t_1 - t_2$  at that point. Here all the points in the *region of conflict* are mapped to the region between the points  $b$  and  $a$  on the real line.

Every time an overlay network shifts traffic to an alternate path it starts probing the new overlay path every  $P$  seconds. If both the overlays shift their traffic almost simultaneously resulting in performance degradation on the first choice alternate path, then this will trigger another response from both overlays. They will shift their traffic to their second choice alternate path. If these second choice paths become overloaded, each overlay may move back to its first choice path, thus entering into oscillations. The time for an overlay to detect a problem on a new path and then move its traffic is given by  $P_i + N_i Q_i + T_i$ . Thus each time the synchronized overlays move together from one set of alternate paths to another, the spacing between the probes change by  $c = P_1 + N_1 Q_1 + T_1 - P_2 - N_2 Q_2 - T_2$ . After  $k$  ( $k = 1, 2, \dots$ ) such reactions, the spacing between the probes can be expressed as:

$$s_k = s_0 + k.c \quad (9)$$

Note that we have implicitly assumed here that the parameter values for the primary and alternate paths remain the same for both the overlays; hence the value of  $c$  is the same regardless of whether we are on the first or second set of alternate paths. We make this assumption for two reasons. First, it allows our model to remain tractable; without this the size of the box (feasible region for probe values) in our model for  $P(S)$  would change with each traffic shift. Second, this is not unreasonable for scenarios in which two overlay networks select locations for their nodes that are similar either because they are strategic or resident where the traffic demands are high. Also, if the values of  $c$  are different for each of the alternate paths then the *rate* at which the spacing moves towards the boundary condition could either increase or decrease, thus introducing the possibility that the number of oscillations could be better or worse than the case that we consider here.

From Eqn 1 the stop condition for the oscillations is given by  $|s_k| > a - b$  (note that  $a > b$ ). The worst case in the number of oscillations happens when  $s_0$  is equal to  $a$  (or  $b$ ) and moves towards  $b$  (or  $a$ ) by  $c$  seconds at each step. It is then straightforward to derive  $\bar{k}$ , the upper bound on the number of oscillations,  $\bar{k} = \left\lceil \frac{a-b}{|c|} \right\rceil$ . Hence,

$$\bar{k} = \left\lceil \frac{T_1 + T_2}{|P_1 - P_2 + N_1 Q_1 - N_2 Q_2 + T_1 - T_2|} \right\rceil \quad (10)$$

Notice that when the overlays have identical parameters, they remain synchronized forever. The model thus follows our intuition that once two overlay gets synchronized, if the

spacing between the probes never changes, they remain synchronized always.

## 5 Simulating Multiple Co-existing Overlays

To validate our hypothesis and the model, we built a simulator that implements the control planes at both the overlay and IP layers. The simulator allows us to analyze the conflicting decisions made by multiple overlays and their impact on both overlay and other IP traffic.

### 5.1 Simulating IP and Overlay Network Dynamics

Within each ISP domain, we emulate an IP-layer interior gateway protocol (IGP) that implements Dijkstra's shortest path algorithm. To simulate real-world network scenarios, we introduce link failures and carefully model the IGP dynamics in response to failures as outlined in [6] and [7]. In any IP network, the link utilization determines the delay, throughput, and losses experienced by traffic flows that traverse the link. To simulate realistic link delays, we model delay on any link as a monotonically increasing piecewise linear convex function of its utilization (as in [5] and [13]).

In our simulations, a single IP network could have multiple overlay networks with nodes resident in its domain. Each overlay can have a different topology and routing strategy. Note that we do not require that all of the nodes participating in an overlay be resident in the same domain. All overlay networks adopt the same routing strategies and path probing mechanisms described in Section 3.1.

We assigned the background traffic between various IP nodes in the network based on the findings in [11]. The traffic between overlay nodes in various overlay networks was assigned such that *the overlay traffic accounts for a significant portion of the IP traffic* (an implicit assumption made in all of our discussions). In our simulations the combined overlay traffic from all the overlay networks was typically 30-50% of the overall traffic. We generated numerous events at the IP layer and observed the reactions of overlay and IP networks, both at the control plane and the data forwarding plane. In the following section, we present some of our key observations.

### 5.2 Illustrating Race Conditions

As mentioned before, our hypothesis is that the co-existence of multiple independent overlays can exhibit race conditions that lead to unexpected network instability. To test our hypothesis, we consider a scenario with five overlay networks deployed on top of a tier-1 ISP backbone topology (similar to [7]), as shown in Figure 4. To simulate a realistic scenario with heterogeneous overlay networks, we chose different timers values (i.e.,  $P$ ,  $Q$ ,  $T$ , and  $N$ ) for each of the

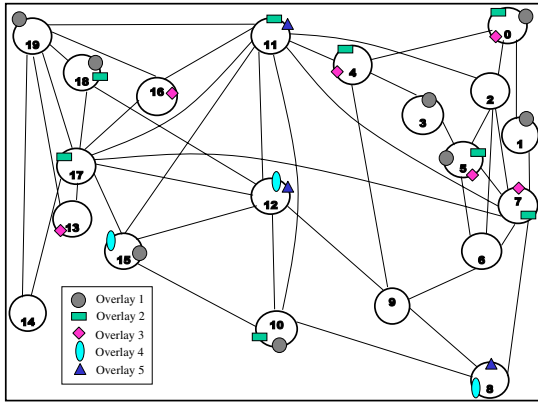


Figure 4. Simulation topology

overlay networks (Table 1). Notice that the timers of the first two overlays have significantly larger values compared to the other three. Even though our simulation topology has a single domain, it is straightforward to see that the observations and results that we present in the rest of the paper can occur even when overlay networks span multiple domains.

Timer	$P$ (ms)	$Q$ (ms)	$T$ (ms)	$N$
Overlay-1	2000	600	300	3
Overlay-2	2000	1000	350	3
Overlay-3	1000	500	200	3
Overlay-4	800	400	120	3
Overlay-5	700	300	100	3

Table 1. Timer Values for the overlays in simulation

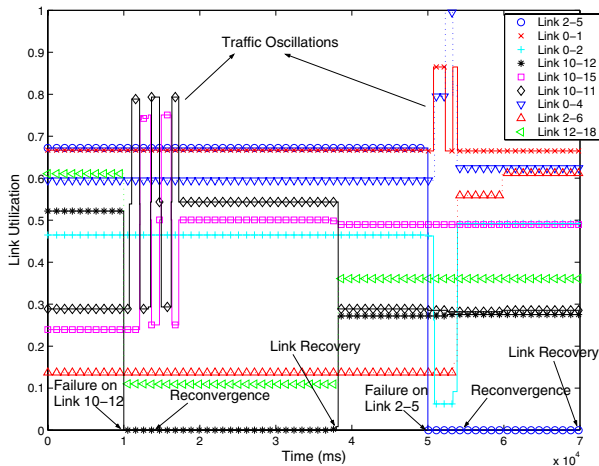


Figure 5. Link utilization as a function of time

We ran numerous simulations by generating different IP-level events and various traffic loads in the overlay networks. Results reveal many different possible interactions between overlay networks triggered by different events. For ease of presentation, we only show the dynamics of multiple overlays in response to a very common event, link failures.

Figure 5 shows the utilization of a subset of links in one of the simulation runs that lasted for 70s. On the x-axis we mark the timeline of various IP-layer events such as link failures, routing *re-convergence* (IGP routing protocol has converged and identified alternate path), and link recovery (failed link becomes operational again). We consider two link failure events: (i) *link 10-12* fails at  $t = 10$ s, and (ii) *link 2-5* fails at  $t = 50$ s. The failures are far apart such that the first link is operational (i.e. recovered) before the second failure event. Note that we do not show multiple simultaneous link failures for ease of illustration. However, multiple failures will exacerbate the race conditions described here.

**Traffic Oscillations:** In Figure 5, we can clearly see that soon after the link failure events, loads on some of links start oscillating. There are two sets of oscillations: one corresponding to the failure of *link 10-12* and the other for the failure of *link 2-5*. During these two sets of oscillations, the paths between some source-destination pairs in the overlay layer change constantly. The first set of oscillations involves two overlay networks (*Overlay-1* and *Overlay-2*) while the second set involves three (*Overlay-1*, *Overlay-2*, and *Overlay-3*). We observe that the overlay paths involved in these oscillations do share at least a few common links in the IP layer. Note that the two sets of oscillations involve different number of overlays, implying that different number of overlay networks can interact with each other resulting in oscillations.

**Stop Triggers for Oscillations:** Even though both sets of oscillations in Figure 5 are triggered by failure events, the stop trigger for the oscillations are different. In the first set, the oscillations stop when the overlay networks de-synchronize, or disentangle themselves, with no external trigger. In the second set, oscillations stop when IGP re-converges. Different runs of our simulations clearly indicate that there is a variety of events that act as stop triggers for these oscillations (e.g., IGP re-convergence, link recovery, self-disentanglement, etc.). An important observation here is that certain IP layer events that act as stop triggers for oscillations at some point in time might not affect the oscillations at another point in time. Also, most of the IP layer events that act as stop triggers are heavily dependent on the network conditions at the IP layer. For example, IGP convergence depends not only on timer values set by the ISPs, but also on the location of BGP peering points [7]. The order of occurrence of these stop triggers is not deterministic, thus introducing unpredictability in the duration of the oscillations. In essence, the end of oscillations depends on numerous factors, thus making it non-trivial to accurately estimate the impact of oscillations on overlay or non-overlay traffic.

### 5.3 Validation of Analytical Model

To validate both our analytic formulation and our implementation in the simulator, we compare  $P(S)$  computed using the model (Section 4) versus that seen in simulation. We consider two similar networks (i.e., all the parameters are identical), but vary the value of the probe interval. The results are given in Figure 6. The simulation results are based on running the simulation 1000 times and calculating the number of times the overlay networks got synchronized. We can see that the analytical results closely match the simulation results.

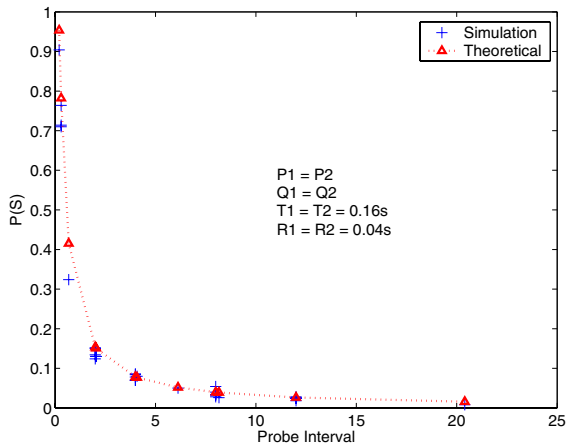


Figure 6. A comparison of theoretical and simulation results for  $P(S)$  between two identical overlay networks with different values of  $P$

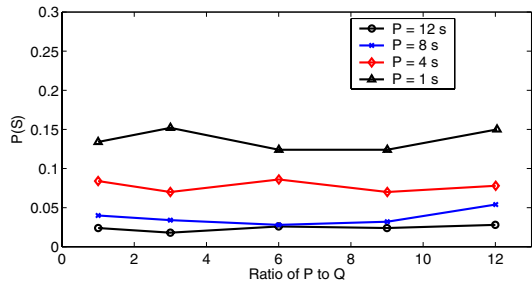


Figure 7.  $P(S)$  as a function of the ratio  $P/Q$  with different values of  $Q$  for a given value of  $P$  in two identical overlay networks

When the two overlay networks are identical (i.e.,  $P_1 = P_2 = P$ ,  $Q_1 = Q_2 = Q$ ,  $T_1 = T_2 = T$ ,  $N_1 = N_2 = N$ , and  $R_1 = R_2 = R$ ), it is easy to see that we have  $a = T$  and  $b = -T$ . Hence the probability of synchronization (from Eqns 5, 6, 7, 8), collapses to the simple equation  $P(S) = T(2P - T)/P^2$ . If our model is correct, this implies that  $P(S)$  is independent of  $Q$ ,  $N$  and  $RTT$ . Figure 7 shows the variation of  $P(S)$  (generated using the simulator)

as a function of  $Q$  for constant values of  $P$ . We can clearly see that for a given probe interval, varying  $Q$  does not impact the probability of synchronization between two identical networks thus confirming the accuracy of our model.

To verify the correctness of the theoretical upper bound on the number of oscillations (Eqn 10), using our simulator we simulate oscillations in two synchronized overlays that are dissimilar (Figure 8). We run the simulations 50 times for each set, but the figure represents only those cases where the overlays synchronize and oscillate. We can clearly see that the theoretical upper bound on the number of oscillations before the overlays disentangle is larger than the actual number of oscillations in our simulations, yet lies near the values observed in simulation. Note that we are exploring the number of oscillations in a small parameter space, but the main purpose of this figure is to validate our model. We look at a wider parameter space in Section 6.2.

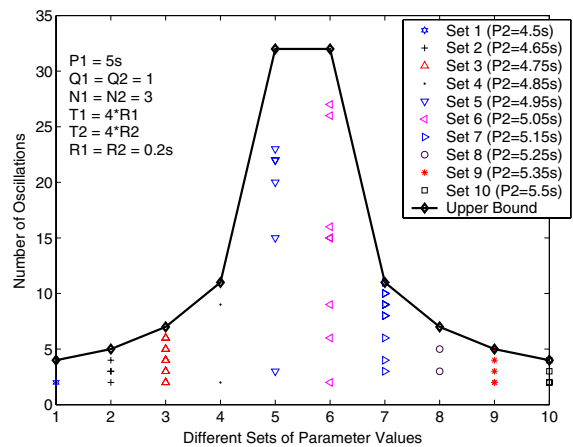


Figure 8. Comparison between theoretical and simulation results for number of oscillations.

## 6 Sensitivity to Probing Parameters

To assess whether or not these race conditions pose a problem in the design of overlay networks, we need to understand whether such situations are pathological (and thus very unlikely to occur) or if there is a reasonable chance that this can happen over some non-narrow range of the parameter space. For the case of two overlays, we have ten parameters, and thus  $P(S)$  describes a probability in 10-dimensional space. We now study how  $P(S)$  varies with respect to some of these parameters, or combinations of them. Due to the complexity of the parameter space, we direct our attention to address the following questions: (i) Is  $P(S)$  non-negligible in operating regions that can occur in the Internet? (ii) Can we count on naturally occurring variations in RTT (due to path length diversity) to reduce  $P(S)$  to negligible values? (iii) If not, which parameter settings

can drive  $P(S)$  to low values? In other words, how should an overlay network designer choose the probing parameters to reduce the likelihood of synchronization, especially when the behavior of other overlays are not known?

### 6.1 Aggressiveness Factor and Probe Parameter Setting

We start with the simplest case of two overlays with identical parameter setting because this case provides some insight about overlays in general. Recall from Section 5.3 that for two identical overlays, we have  $P(S) = T(2P - T)/P^2$ . We see that  $P(S)$  depends only on the probe interval and the timeout values of the overlays. The maximum value of  $P(S) = 1$  occurs when  $T = P$ , i.e., for these parameter settings, the overlay networks will definitely synchronize. If  $P = 2T$  then  $P(S) = 0.75$ . In order to decrease the probability of synchronization to less than 0.05 (i.e. 5% chance of synchronization) we need to use  $P \approx 40T$ .

We are thus motivated to characterize overlay networks by their probing frequencies. We consider overlays that probe frequently and move their traffic quickly as aggressive. We define an *aggressiveness factor*,  $\alpha$ , of an overlay network as the ratio of the timeout and probe interval,  $\alpha_i = T_i/P_i$ . Note that  $RTT \leq T \leq P$ , hence  $0 < \alpha \leq 1$ . For two identical overlay networks we have  $P(S) = 2\alpha - \alpha^2$ , which shows that as the networks increase their aggressiveness (i.e. as  $\alpha \rightarrow 1$ ),  $P(S)$  increases.

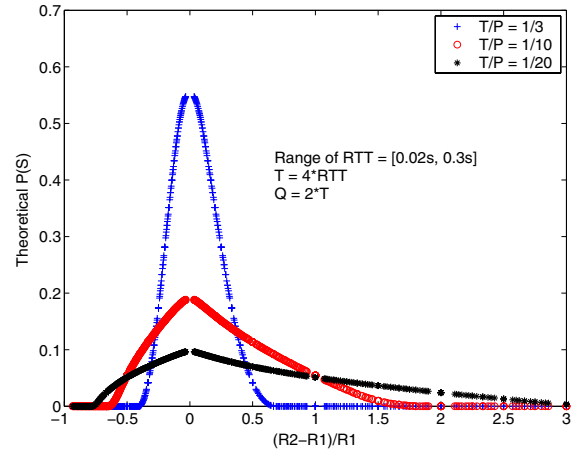
In many of our sample scenarios in the next section, we varied RTT values between 20ms and 300ms to capture a variety of realistic Internet overlay paths that span either a small or large geographic distance (i.e. nearby cities to international routes). We choose the timeout value to be four times the RTT. This is motivated by the type of approach usually followed in TCP in which timeout values are set to be the mean RTT plus 3 or 4 times the standard deviation. Assuming the standard deviation is similar to the mean, we use  $T = 4 * RTT$  in our calculations.

The other probing parameters  $P$  and  $Q$  can be set in two ways: (i) *Proportional* values, where  $P$  and  $Q$  are set to be multiples of  $T$  (and hence RTT) and are different for each path in the overlay; (ii) *Fixed* values, where  $P$  and  $Q$  are constants independent of  $T$  and RTT and therefore the same for all the paths.

### 6.2 Results and Discussion

In Figure 9, we explore the impact of variations in RTT values on  $P(S)$  for two proportional parameter overlays. Although both overlays have the same aggressiveness in this example, the actual values of  $P$ ,  $Q$ , and  $T$  will differ for each overlay because the parameters ultimately depend on the particular RTT. We observe in this figure that when both overlays are aggressive (e.g.,  $T/P = 1/3$ ),  $P(S)$  can

be as high as 55%. When both are non-aggressive (e.g.,  $T/P = 1/20$ ),  $P(S)$  never gets above 10%. In this figure, we plot  $P(S)$  versus the relative difference in RTT values between two overlays. The figure indicates that when one RTT is more than twice the value of the other, then this synchronization issue is not a concern as  $P(S)$  is near or at zero. However when the RTTs are less than 50% different from one another, then we can have non-negligible probability of synchronization. This could happen for two overlay networks that both span a similar geographic region. Since the dependence here is on the relative RTT's, the actual size of this geographic region does not matter.



**Figure 9.  $P(S)$  Vs  $(R_2 - R_1)/R_1$  for proportional parameter overlays with similar aggressiveness and varying RTT**

As overlays are not widely deployed and their performance requirements are not yet well understood, it is not clear how to decide for which values should  $P(S)$  be considered “significant”, or “non-negligible”. In this paper, we consider  $P(S)$  to be non-negligible if it exceeds 10%. Admittedly, this number is subjective, however we will see plenty of scenarios in which  $P(S)$  is considerably far away from zero to indicate that synchronization problems should not be neglected.

In Figure 10, we examine some scenarios in which the two overlays have different aggressiveness factors. In these scenarios, the first overlay is set to be aggressive, while the second overlay varies from aggressive ( $\alpha_2 = 1/5$ ) to non-aggressive ( $\alpha_2 = 1/20$ ). We see that even in the case of one aggressive and one non-aggressive overlay network,  $P(S)$  can still be non-negligible for a wide range of relative RTT values. However, this figure indicates that an overlay might benefit from using non-aggressive parameters even if another overlay behaves aggressively. To further explore this hypothesis, we consider a wider variety of cases in Figure 11.

Figure 11 shows the value of  $P(S)$  as a function of the

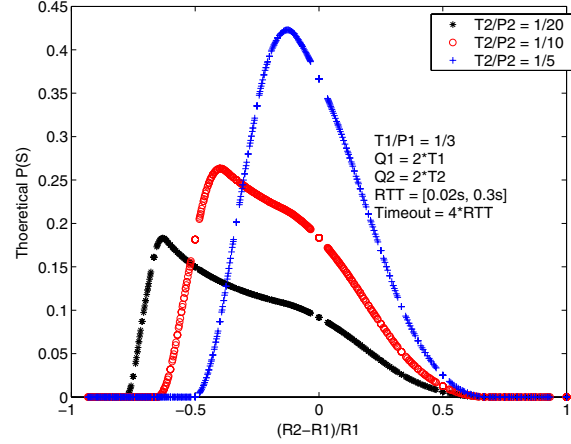
aggressiveness factors of the two overlays. Each curve in the graph represents the value of  $P(S)$  for a fixed value of  $T_1/P_1$  but different values of  $T_2/P_2$ . We can clearly see that as the aggressiveness of both the overlays increase, there is a higher chance of synchronization. This probability significantly decreases when the overlays are non-aggressive. This confirms that as long as one of the overlays is non-aggressive, the probability of synchronization is low. In other words, setting a high value of  $P$  is critical to reducing  $P(S)$ . We wish to point out that there could be fairness issues when one overlay is very aggressive, and exploits the non-aggressive parameter settings of the other overlay. We defer the study of fairness to future work.

We now look at the impact of using a fixed parameter approach to choosing  $P$  and  $Q$ . In Figure 12, we consider the case of two RON networks. The pattern in this figure is explained as follows. Each straight line of points belongs to the cases of a fixed  $R_1$  as  $R_2$  is varied through its entire range. An interesting observation from this plot is that even when the relative difference between the RTT's is zero,  $P(S)$  does *not* take on a single value, but instead can take on any of a number of values.  $P(S)$  is at its minimum when both RTT values are small ( $R_1 = 20\text{ms}$ ,  $R_2 = 20\text{ms}$ ), and achieves its maximum when both RTT values are large ( $R_1 = 300\text{ms}$ ,  $R_2 = 300\text{ms}$ ). This suggests that in fixed parameter overlays, unlike proportional parameter overlays, the absolute value of  $RTT$  is an important factor in determining  $P(S)$ . We observe that RON networks are designed to be non-aggressive ( $T/P$  varies between 0.007 and 0.1) and this results in low synchronization probabilities. However, there still do remain a number of cases in which  $P(S)$  exceeds 10%.

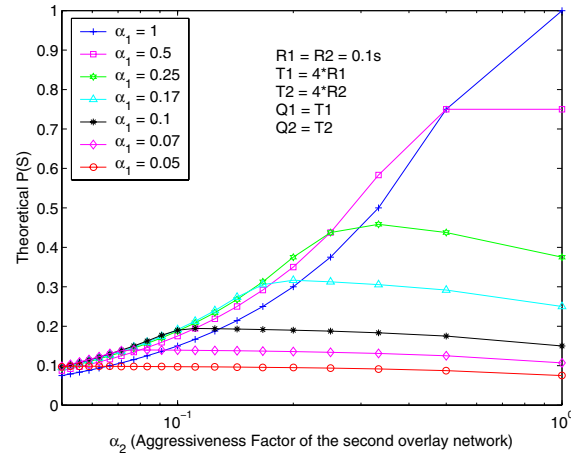
In Figure 13, we consider a case of two fixed parameter overlays with different values of  $P$  and  $Q$ . We see a similar behavior as in the previous case of fixed parameter overlays. We wish the point out that in these two cases  $P(S)$  is significant for a wider range of values on the x-axis when compared to Figure 9. In other words,  $P(S)$  does not reach zero once the relative difference exceeds 60% or 70%. Using a fixed approach to parameter selection means  $P$  is constant, however, since the aggressiveness is determined by  $\alpha = T/P$ , the aggressiveness is varying per path (since  $T$  is proportional to RTT). The overlay is more aggressive on long paths and less so on shorter paths. The increased aggressiveness on longer paths could explain why  $P(S)$  does not disappear when the relative difference of RTTs is high (e.g., 300%).

The above results show the influence of  $P$  and  $T$  on the probability of synchronization. Figure 14 explores the influence of  $Q$  on  $P(S)$  for fixed parameter overlays. For a given value of  $P_2$ , as  $Q_2$  grows,  $P(S)$  decreases. For a given value of  $Q_2$ , the range of cases (i.e., values of  $P_2$ ) in which synchronization can happen are more numerous

when  $Q_2$  is small than when it is large. Thus, in general, it seems beneficial to use a higher value of  $Q$  to reduce the chances of synchronization. The tradeoff here is a delayed reaction by the overlay. However, most importantly, based on Figure 14 we can conclude that the influence of the high frequency probes ( $Q$ ) on  $P(S)$  is far less significant compared to the influence of the probe interval  $P$  or the timeout  $T$ .



**Figure 10. Proportional parameter overlays with mixed aggressiveness and varying RTT**



**Figure 11. Proportional parameter overlays with mixed aggressiveness and a chosen value of RTT**

Finally, Figure 15 shows the upper bound on the number of oscillations between two overlays after they are synchronized for both proportional and fixed parameter overlays. Even though we are exploring a small subset of the parameter space, there are a considerable number of cases where the number of oscillations is more than 5.

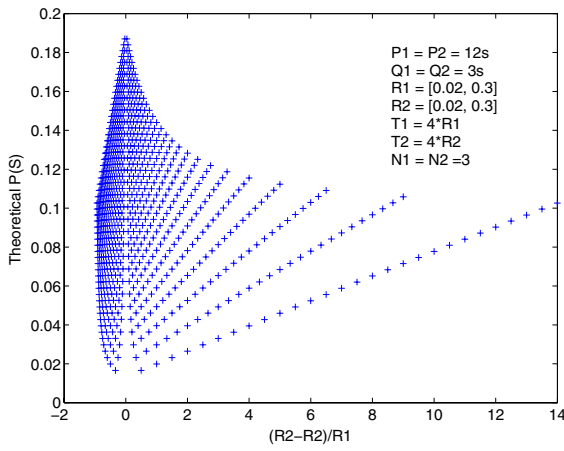


Figure 12. Fixed parameter overlays (RON-like) with the same values of  $P$  and  $Q$ , and varying RTT

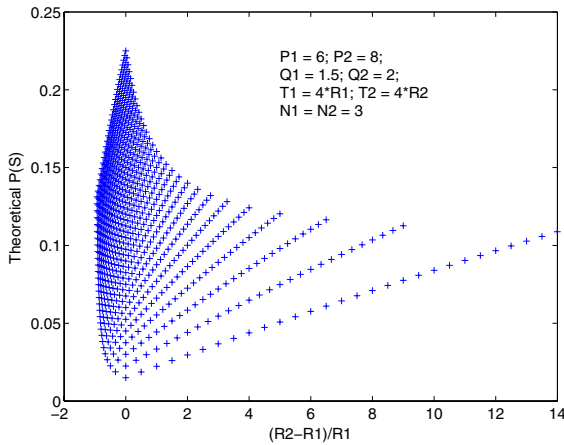


Figure 13. Fixed parameter overlays with the different values of  $P$  and  $Q$ , and varying RTT

## 7 Implications of synchronization in overlays

Today's Internet does not have multiple overlay networks deployed, that is, not the type that use continuous probing to do path selection. We have explored the possibility of race conditions occurring should multiple overlays get deployed so that we may accumulate some wisdom about how to design such networks before they become widely used. We have seen a variety of scenarios in which the probability of synchronization exceeds 10%. These scenarios included cases in which RTTs were varied, probe rates were varied, and the relationships among the parameters were varied. We thus believe that there does exist a non-narrow range of the parameter space in which synchronization is non-negligible. We also illustrated that once synchronization occurs, the resulting oscillations can sometimes last for a long time. We thus believe that these issues should be taken into consider-

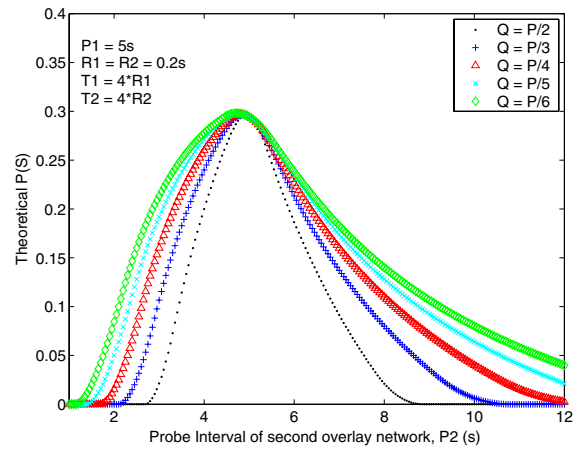


Figure 14.  $P(S)$  as a function of  $P_2$  with  $P_1 = 5s$

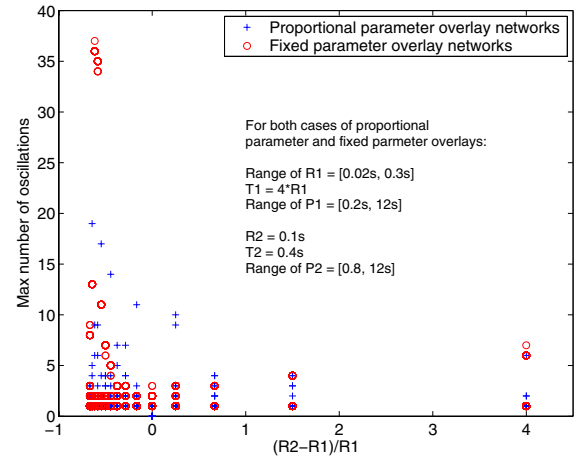


Figure 15. Maximum number of oscillations as a function of relative RTT

ation when overlay networks are designed and configured.

One of the questions we set out to explore was whether we can count of variations in RTT alone to avoid synchronization. When using a proportional approach to parameter selection,  $P(S)$  can be significant (e.g., 40%) when the relative difference in RTTs is small (roughly less than 70%) and the overlays are aggressive. When using a fixed approach to parameter setting,  $P(S)$  can exceed 10% even when the relative difference is as high as 300%. Moreover, when using the fixed approach, the absolute value of RTTs matter and large RTTs can bring about larger  $P(S)$ . This implies, for example, that designing a cross-continental overlay network is more challenging than designing one in a single country. Since synchronization can occur, even when overlay paths have dissimilar RTTs (whether large or small), overlay network designers should *not* rely upon differences in RTTs to avoid synchronization.

We believe that overlay networks should be designed

with care so as to mitigate race conditions as much as possible. This is non-trivial as we have shown there isn't any "ideal" set of parameters that ensures avoidance of synchronization. Our results indicate that using a proportional approach to parameter selection might be better than using a fixed one. The proportional approach narrows down the range of relative RTTs in which synchronization can occur. Also, proportional parameter overlays depend only on the *relative* RTTs and can exploit the path diversity in the Internet better than fixed parameter overlays. But in reality, different overlay networks could easily end up choosing the same strategic locations to place their nodes, resulting in similar RTT values for various paths in different overlays, thus making it harder to achieve and exploit path diversity. In other words, using either a proportional or fixed parameter approach could result in a fair chance of experiencing oscillations.

We saw that  $P(S)$  is more sensitive to the low-frequency probe  $P$  than the higher frequency probe  $Q$ . It appears that the best approach for averting race conditions, is for overlays to be nonaggressive in their probing, i.e., by using large values of  $P$ . The tradeoff here is a slower reaction time. We showed that being nonaggressive can result in smaller values of  $P(S)$  even if other overlays are aggressive. There may be implications here in terms of fairness. We also show that it is beneficial to be nonaggressive, as we suspect that the trend is towards building more aggressive overlays because of the popular belief that overlays can outperform layer-3 networks in terms of their reaction time to performance degradation events. We wish to point out that when many overlays start to co-exist, aggressive probing can have negative consequences and overlays can inadvertently step on each other.

## 8 Future Directions

In addition to nonaggressive probing, there are other ways to mitigate oscillations. One obvious method is to add randomness into various parameters. Although this is a simple approach, we believe that it may not be very effective in solving the problem due to the fact that  $P(S)$  and  $\bar{k}$  depend on the difference terms (like  $P_1 - P_2$  in Eqn 10) and randomness added to different overlays could cancel each other.

An approach to quickly stop oscillations would be to employ a back-off approach, i.e., to successively increase the timeout and/or probe parameters each time an overlay source node decides to switch routes to the same destination. Another technique is to have overlays share information in order to ensure that harmful interactions are minimized. A detailed analysis of the above issues are a part of our future work.

We also plan to perform asymptotic analysis to under-

stand what happens as the number of overlays increase and these overlays may or may not use similar overlay routing algorithms. We also hope to explore the issue of fairness when distinct overlays are configured with different levels of aggressiveness.

Another direction of future work is to gain a comprehensive understanding of *stop triggers* for oscillations. In other words, we plan to investigate: (i) what are the various possible stop triggers?, (ii) what is the stop trigger for a given set of oscillations?, (iii) frequency of occurrence of different stop triggers, (iv) the influence of network topology and conditions on the occurrence of stop triggers, etc. Since this is the first work of its kind, we have many assumptions in our model and analysis, but we plan to relax these assumptions and improve our model in the future.

## References

- [1] Akamai. <http://www.akamai.com>.
- [2] D. Anderson, H. Balakrishna, M. Kaashoek, and R. Morris. Resilient Overlay Networks. In *SOSP*, Oct. 2001.
- [3] M. Castro, P. Druschel, A. Kermarrec, A. Nandi, A. Rowstron, and A. Singh. SplitStream: High-Bandwidth Multicast in Cooperative Environments. In *SOSP*, Oct. 2003.
- [4] S. Floyd and V. Jacobson. The Synchronization of Periodic Routing Messages. In *ACM SIGCOMM*, Sept. 1993.
- [5] B. Fortz and M. Thorup. Internet Traffic Engineering by Optimizing OSPF Weights. In *IEEE INFOCOM*, Mar. 2000.
- [6] G. Iannaccone, C.-N. Chuah, S. Bhattacharyya, and C. Diot. Feasibility of IP Restoration in a Tier-1 Backbone. *IEEE Network*, Mar. 2004.
- [7] R. Keralapura, C. Chuah, G. Iannaccone, and S. Bhattacharyya. Service Availability: A New Approach to Characterize IP Backbone Topologies. In *IWQoS*, June 2004.
- [8] R. Keralapura, N. Taft, C. N. Chuah, and G. Iannaccone. Can ISPs take the heat from overlay networks? In *ACM HotNets*, Nov. 2004.
- [9] T. Leighton. The Challenges of Delivering Content and Applications on the Internet. In *NSDI Keynote*, May 2005.
- [10] Y. Liu, H. Zhang, W. Gong, and D. Towsley. On the interaction between overlay routing and traffic engineering. In *IEEE INFOCOM*, Mar. 2005.
- [11] A. Medina, N. Taft, K. Salamatian, S. Bhattacharyya, and C. Diot. Traffic Matrix Estimation: Existing Techniques and Future Directions. In *ACM SIGCOMM*, Aug. 2002.
- [12] A. Nakao, L. Peterson, and A. Bavier. A Routing Underlay for Overlay Networks. In *ACM SIGCOMM*, Aug. 2003.
- [13] L. Qiu, Y. Yang, Y. Zhang, and S. Shenker. On Selfish Routing in Internet-Like Environments. In *ACM SIGCOMM*, Aug. 2003.
- [14] S. Rhea, D. Geels, T. Roscoe, and J. Kubiatowicz. Handling Churn in a DHT. In *USENIX ATC*, June 2004.
- [15] S. Savage, A. Collins, E. Hoffman, J. Snell, and T. Anderson. The end-to-end effects of internet path selection. In *ACM SIGCOMM*, Aug. 1999.
- [16] K. Sripanidkulchai, B. Maggs, and H. Zhang. An Analysis of Live Streaming Workloads on the Internet. In *ACM IMC*, Oct. 2004.