Race Conditions in Coexisting Overlay Networks

Ram Keralapura, Chen-Nee Chuah, *Senior Member, IEEE*, Nina Taft, *Member, IEEE*, and Gianluca Iannaccone

***Abstract*—By allowing end hosts to make independent 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 could experience race conditions, resulting in oscillations (in both route selection and network load) and cascading reactions. We pinpoint the causes for synchronization 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 also show that the model can be easily extended to include a large number of co-existing overlays. 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 path probing process in overlay networks and examine strategies to reduce the impact of race conditions.**

***Index Terms*—Interaction between multiple overlay networks, race conditions, synchronization, traffic oscillations.**

I. 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 [4], 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 [5] 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 [6]. 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 [7], we arrive at a hypothesis that two (or more) co-existing overlays can experience race conditions and become synchronized leading to route and traffic oscillations, and cascading reactions (i.e., an event in one overlay that triggers an event in a second overlay, that triggers an event in a third, and so on). This hypothesis is formulated based on two key observations. First, Floyd *et al.* [8] 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 periodic processes would most likely become an increasing problem. Overlay networks typically use a periodic probing process to detect events that deteriorate path performance and to identify alternate paths between source and destination pairs. Events such as failures can trigger an overlay network to move its traffic to an alternate path. The work in [8] would suggest that when two different overlays that both use *periodic* probing react to the same IP layer triggering event, it results in 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 quantify the likelihood of oscillations and cascading reactions, how long they can last, and the conditions under which they occur. The contributions of this work are:

- We pinpoint the reasons for race conditions (oscillations and cascading reactions) in terms of partially overlapping paths and periodic probing process (Section IV).
- 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 V). We also show that our analytical model is extensible when an arbitrary number of overlay networks coexist (Appendix I).
- 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 occur when the overlay

Manuscript received February 23, 2006; revised December 2, 2006; approved by IEEE/ACM TRANSACTIONS ON NETWORKING Editor K. Ross. This work was supported by the National Science Foundation under NSF CAREER Grant 0238348.

R. Keralapura and C.-N. Chuah are with the Rubinet Research Group, University of California, Davis, CA 95616 USA (e-mail: rkeralapura@ucdavis.edu; chuah@ucdavis.edu).

N. Taft and G. Iannaccone are with the Intel Research Laboratory, Berkeley, CA 94704 USA (e-mail: nina.taft@intel.com; gianluca.iannaccone@intel.com).

Digital Object Identifier 10.1109/TNET.2007.897959

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 (Sections III and VI).

- 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 VII).
- We explore techniques to reduce the impact of race conditions. We show that adding randomness to the probing parameters does *not always* help in avoiding synchronization. However, a random back-off mechanism that suppresses 'reaction time' of overlays could help in significantly reducing the chances of oscillations. We also show that it is harder to avoid cascading reactions and hence application and network providers should revisit deployment strategies of overlay networks. (Section IX).

In Section VII, we summarize the implications of this study on the design of overlay routing strategies. In contrast to today's practice we show that it is beneficial for overlays to be nonaggressive since it lowers the likelihood of synchronizing. We present our conclusions in Section X.

An earlier version of this work appeared in [9]. This paper improves that work and extends it with additional materials: (i) discussion of cascading reactions as another form of race condition that can occur when several overlays co-exist; (ii) extension of the analytical model into a generic form to include several overlays instead of just two overlays; and (iii) examining strategies like adding randomness and back-off mechanisms to reduce the impact of race conditions.

II. Related Work

Interactions between multiple co-existing overlays were first addressed by Qiu *et al.* [10], 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.* [11] model the interaction between overlay routing and IP traffic engineering as a two-player game, where the overlay attempts to minimize its delay and traffic engineering minimizes network cost. In this 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 congestions that lead to dynamic re-computation of routes.

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 [8], the particular periodic process we focus on is different from the one considered in [8]. 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 we consider two different instances of the overlay protocol with nonidentical parameters and overlay topologies that only partially overlap in the underlying physical network.

III. Simulating Multiple Co-Existing Overlays

To validate our hypothesis (and the analytical model presented in Section V), we built a simulator that implements the control planes at both the overlay and IP layers. The simulator allows us to assess the impact of the conflicting decisions, made by multiple overlays, on both overlay and other IP traffic. In this section, we describe how we model overlay networks, and present the details of the simulator design.

A. Modeling Overlay Networks

While different overlay networks, designed for a wide range of applications, may differ in their implementation detail, most of them provide a common set of functionalities, including periodic path/performance monitoring, failure detection and restoration. In this paper, we 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 $P = 12$ s, $Q = 3$ s, and $N = 3$ while Akamai network can be modeled with much smaller values of P, Q, and N [13]. 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.[1]

B. 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 [14] and [15]. 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 [16] and [10]).

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 III-A.

We assigned the background traffic between various IP nodes in the network based on the findings in [17]. 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.

C. Race Conditions in Multiple Overlays

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 [15]), as shown in Fig. 1. 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 overlay networks (Table I). The timers of the first two overlays have significantly larger values compared to the other three. Even though we only consider a single domain, it is clear that the observations and results presented in the rest of the paper can occur even when overlay networks span multiple domains.

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.

Fig. 2 shows the utilization of a subset of links in one of the simulation runs that lasted for 70 s. On the x axis we mark the timeline of various IP-layer events such as link failures, routing

[1]As long as the current path adheres to the performance bounds, an overlay does not look for alternate paths even if they may have better performance.

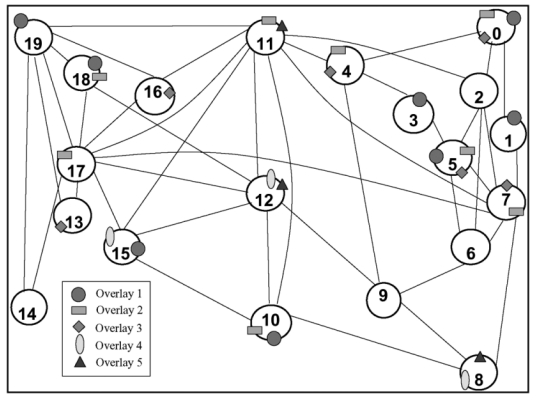


Fig. 1. Simulation topology.

TABLE I
TIMER VALUES FOR THE OVERLAYS IN SIMULATION

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

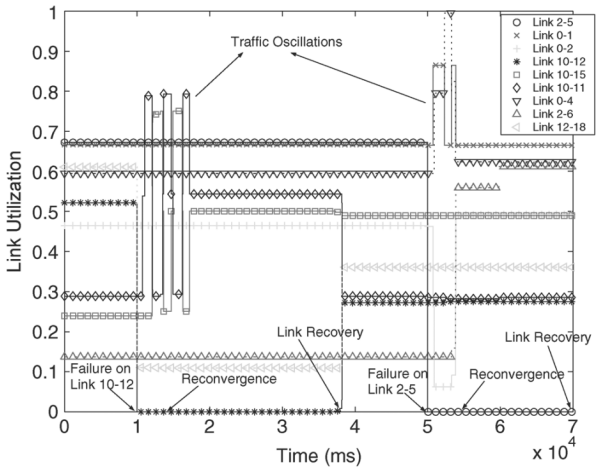


Fig. 2. Link utilization as a function of time.

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: Fig. 2 shows how loads on some of links start oscillating soon after the link failure events. 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*). The overlay paths involved in these oscillations do share at least a few common IP-layer links.

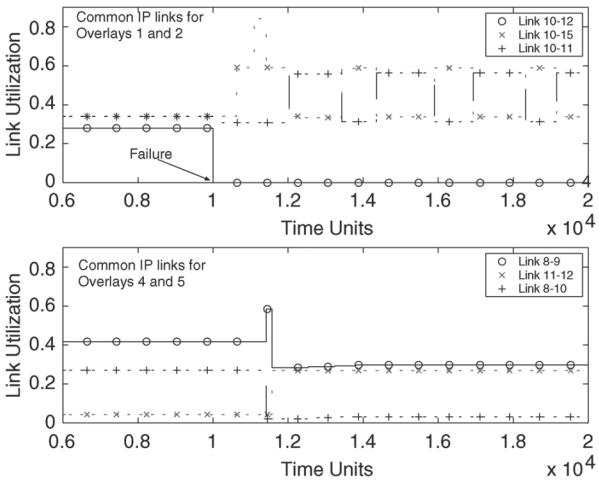


Fig. 3. Second simulation run: cascading reactions.

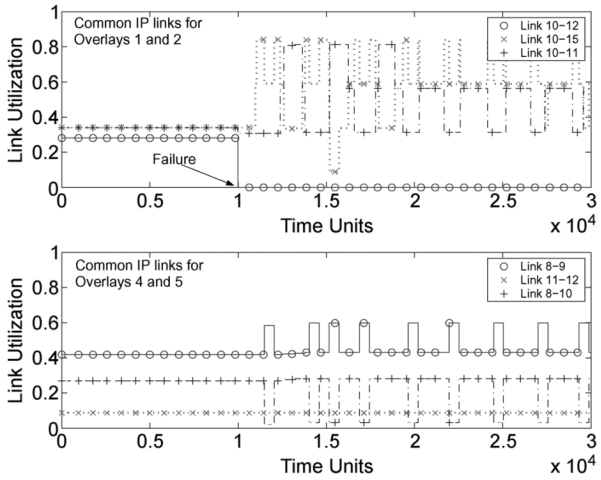


Fig. 4. Third simulation run: cascading reaction leading to oscillations.

Note that the oscillation events involve different number of overlays, implying that different number of overlay networks can synchronize with each other.

We also observed that different runs of our simulations had different stop triggers for oscillations (e.g., IGP re-convergence, link recovery, self-disentanglement). It is important to note that certain IP-layer events that act as stop triggers for one set of oscillation 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 current network conditions at the IP layer. For example, IGP convergence depends on the timer values set by the ISPs and the location of BGP peering points [15]. The order of occurrence of these stop triggers is not deterministic, thus introducing unpredictability in the duration of the oscillations.

Cascading Reactions: We also observed that a reaction from one overlay network can trigger a series of reactions in other overlay networks. We refer to such a domino effect as *cascading reactions* or *cascading route changes*.

We ran the same simulation as before but with a slightly different traffic load in the overlay networks. The top graph in Fig. 3 shows the load on IP links that were common to the overlay paths between nodes 10 and 18 in *Overlay-1* and *Overlay-2*. The bottom graph shows the load on some of the links used by *Overlay-4* and *Overlay-5* between nodes 8 and 15, and, 8 and 12, respectively.

By observing the link loads on these IP links (Fig. 3) and correlating it with the path changes at the overlay control layer, we found the following. Soon after the failure of *link 10-12*, *Overlay-1*, which uses the failed link as a part of the primary path between nodes 10 and 18, moves its traffic on to an alternate path through *link 10-15*. We can see this as an increase in the load on *link 10-15* in the top graph of Fig. 3. A little later *Overlay-2* also moves its traffic onto the same alternate path. We can see this as a further increase in the load on *link 10-15* in the top graph. This increase in load on *link 10-15* deteriorates the performance of the path from node 8 to node 15 in *Overlay-4*. The primary path for this source-destination pair is 8-10-15. *Overlay-4* reacts to this performance deterioration and moves its traffic to an alternate path (i.e., 8-9-12-15). This results in a decrease in load on *link 10-15* as in the top-graph and a simultaneous increase in load on *link 8-9*, as shown in the bottom graph. This traffic shift overloads *link 8-9* and *link 9-12*, prompting the traffic between nodes 8 and 12 in *Overlay-5* to find an alternate path. Note that the primary path from node 8 to node 12 in *Overlay-5* was 8-9-12. *Overlay-5* finds an alternate path (i.e., 8-7-11-12) and moves its traffic. This leads to a decrease in load on *link 8-9* and an increase on *link 11-12* (bottom graph). This shows that a reaction in one or more overlay networks could lead to a series of reactions from other overlays. A critical observation here is that such cascading reactions could become common as the number of overlays that co-exist increases.

Fig. 4 shows the results from our third simulation run which differs from the second run only in the overlay traffic load. The cascading reactions observed are similar to the second simulation run except that: (i) the alternate path found by *Overlay-5* between nodes 8 and 12 is 8-10-15-12 instead of 8-7-11-12; and (ii) the reaction of *Overlay-4* to the routing decisions made by *Overlay-1* and *Overlay-2* results in oscillations in both *Overlay-4* and *Overlay-5*. The resulting load fluctuations on *link 10-15*, *link 10-11*, *link 8-10*, and *link 8-9* can be observed in the top and bottom graphs of Fig. 4. Note that all four overlays (*Overlay-1*, *Overlay-2*, *Overlay-4*, and *Overlay-5*) start oscillating after the failure of *link 10-12* resulting in highly unstable network condition. The only difference between the simulation runs in Figs. 3 and 4 is the overlay load conditions. In other words, an increase in load in a overlay network could completely change the complexity of the problem. It is worth noting that a gradual increase in overlay traffic load over time could be a major factor that could lead to such cascading reactions.

Our observation of cascading reactions in multiple overlay networks is similar to the *bistable* behavior detected in traditional nonhierarchical telephone [18]. Typically, these networks are built as a full-mesh of 4ESS switches with logical links between them. By default, under low-load conditions, all the calls use the direct (one-hop) path between switches. Under high loads, when the direct path is full, the network picks a two-hop alternate path to carry a call. These two-hop calls consume more resources in the network, and could force other potential direct calls to use two-hop paths. This might result in most of the calls

using two-hop paths thus reducing the network capacity by half. This is very similar to cascading reactions observed in overlay networks. The solution for this problem in telephone networks was to use *trunk reservation* schemes that prevent a link to be used for two-hop calls when its utilization exceeds a threshold value. This scheme is inapplicable in the context of overlays since (i) there is no single administrative control for different overlays, and (ii) IP networks lack admission control and hence reservation of bandwidth on different links is not feasible.

IV. Why Do Race Conditions Occur?

In this section we present the arguments for why and when race-conditions (hence traffic oscillations and cascading reactions) between multiple co-existing overlays occur.

A. 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. Fig. 5 illustrates this case with overlays on top on an IP network. The node pair *A-F* in *Overlay-I*, and pair *A-L* 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 overlays, such that their 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.

Periodic Probing Process: As mentioned in Section III-A, overlays periodically probe their paths. Consider the two overlays in Fig. 5 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 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.

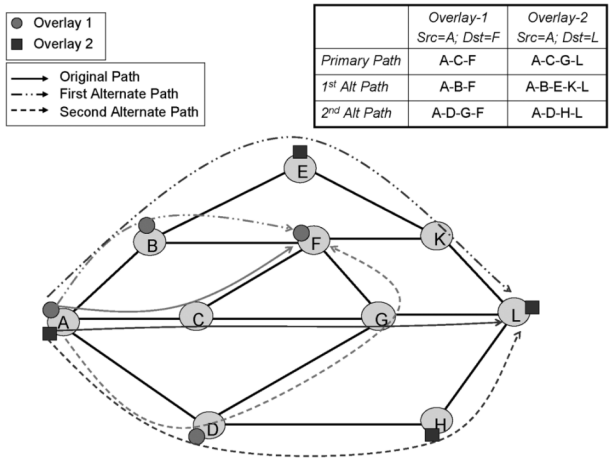


Fig. 5. Two overlay networks that partially share primary and alternate paths.

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.

B. Conditions for Cascading Reactions

Cascading reactions tend to occur when a large number of overlays coexist and satisfy the "path performance degradation" and "periodic probing process" conditions similar to case of oscillations, and also satisfy the following:

Topology (i.e., Primary and Backup Paths): Although oscillations also require the topology condition to be satisfied, the requirement for cascading reactions are quite different from that of oscillations. Cascading reactions could occur when the alternate path of first overlay overlaps with the primary path of the second overlay, the alternate path of the second overlay overlaps with the primary path of the third overlay, and so on. Fig. 6 illustrates a scenario with three overlays spanning multiple domains that could lead to cascading reactions. Consider the primary and alternate paths between the node pairs as indicated in the figure. The alternate path between the node pair *A-G* in *Overlay-1* partially overlaps with the primary path between the node pair *C-K* in *Overlay-2*. Same is the case for the node pair *C-K* in *Overlay-2* and node pair *B-H* in *Overlay-3*. For simplicity of discussion assume that the link *D-G* fails. Soon after the failure, *Overlay-1* will move the traffic between the node pair *A-G* from its primary path (*A-D-G*) to the alternate path (*A-C-F-G*). This alternate path shares the link *C-F* with the primary path between node pair *C-K* in *Overlay-2*. If this link gets overloaded degrading the performance, then *Overlay-2* could move the traffic between node pair *C-K* to the alternate path, *C-B-E-H-K*. Notice that *Overlay-2* and *Overlay-3* now share links *B-E* and *E-H*.

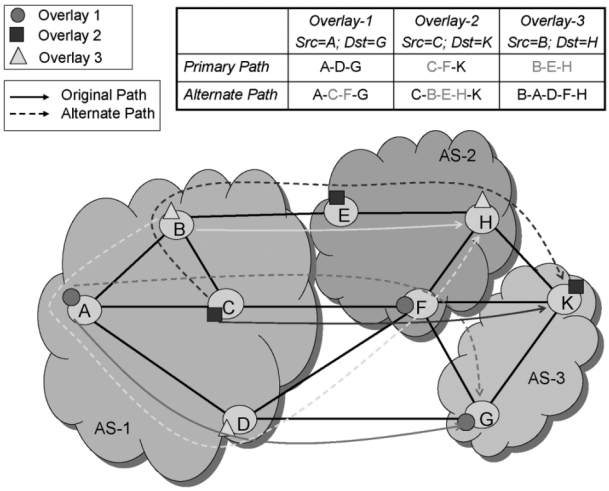


Fig. 6. Three overlay networks in multiple domains that partially share their primary and alternate paths.

If anyone of these links gets overloaded, then *Overlay-3* could move its traffic to the alternate path, *B-A-D-F-H*. If there are more overlays in the system that satisfy the topology condition then these reactions could percolate to several overlays affecting their performance.

Since a particular overlay network cannot know whether its paths overlap with those of another overlay, and since it cannot predict performance failures, avoiding these conditions is beyond the control of an overlay network. However an overlay can control its own probing process, thus in the rest of this work we focus on modeling the impact of the path probing process on oscillations.

V. 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.

A. 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 III-A, 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.[2] 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 two overlays get synchronized if *any one* of the following two conditions are satisfied:

$$\text{Sync Condition} - 1: \quad 0 < f_2 - f_1 < T_1 \tag{1}$$

$$\text{Sync Condition} - 2: \quad 0 < f_1 - f_2 < T_2. \tag{2}$$

Since the above two conditions are independent of each other, we can combine them as follows:

$$\begin{aligned} -T_1 &< f_1 - f_2 < T_2 \\ -T_1 &< (t_1 + N_1 Q_1) - (t_2 + N_2 Q_2) < T_2 \\ b &< t_1 - t_2 < a \end{aligned} \tag{3}$$

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

[2]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 $R_{ij}/2$.

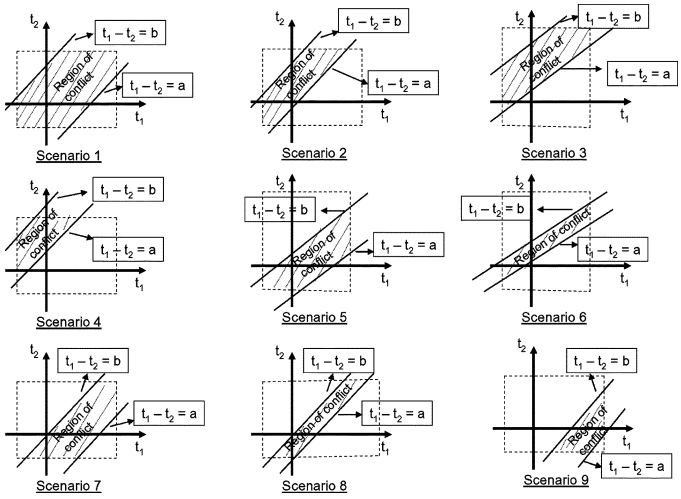


Fig. 7. All possible scenarios to calculate the *region of conflict*.

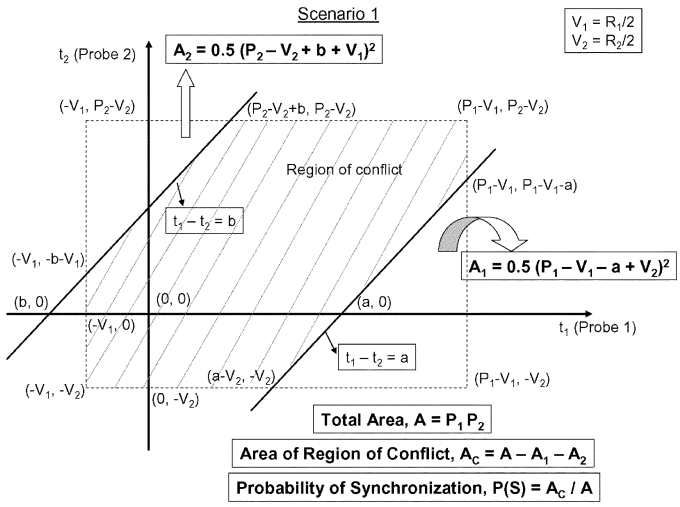


Fig. 8. Scenario 1 ($V_1 = R_1/2$ and $V_2 = R_2/2$).

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,[3] 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 Fig. 8). This geometric representation allows us to compute the probability of synchronization, $P(S)$, of two overlays in an intuitively simple way. We define *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 (3). 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:

$$\text{Line 1}: \; t_1 - t_2 = a \tag{4}$$

$$\text{Line 2}: \; t_1 - t_2 = b. \tag{5}$$

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

There are a number of ways in which *Line-1* and *Line-2* intersect the boundaries of the rectangle. The nine different scenarios for intersection are illustrated in Fig. 7. Consider Fig. 8 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 Fig. 8, 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

$$A_C = A - A_1 - A_2. \tag{6}$$

Thus we can express the probability of synchronization as

$$P(S) = Probability(b < t_1 - t_2 < a) = \frac{A_C}{A}. \tag{7}$$

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 Fig. 8, we see that A_C can be computed using

$$A = P_1 P_2 \tag{8}$$

$$A_1 = 0.5(P_1 - R_1/2 - a + R_2/2)^2 \tag{9}$$

$$A_2 = 0.5(P_2 - R_2/2 + b + R_1/2)^2. \tag{10}$$

Although 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 Fig. 7, *Line-1* intersects the top and bottom edges of the rectangle. If we use (9) to calculate A_1, then A_1 is larger that it should be since it includes space outside the rectangle. We thus need to add another term, E, to A_C in (6) to compensate for the *excess* included in A_1. In the case of *Scenario 2*, 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 compensation areas for each of the nine scenarios since it is a straightforward computation. Although this model results in 9 different scenarios, each with a different equation for $P(S)$, it is still attractive due to its simplicity. This model can be extended to an arbitrary number of overlays, as shown in the Appendix.

[3]Since we focus only on the primary path in both the overlays, we drop the second subscript in R_{ij}.

B. 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 Fig. 5, 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. 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 $\Re^2$ *region of conflict* space onto $\Re^1$ 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 $(k = 1, 2, \ldots)$ such reactions, the spacing between the probes can be expressed as

$$s_k = s_0 + k.c. \tag{11}$$

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 overlays 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, introducing the possibility that the number of oscillations could be better or worse than the case that we consider here.

From (3) 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, $\bar{k} = \lceil (a-b)/|c| \rceil$. Hence

$$\bar{k} = \left\lceil \frac{T_1 + T_2}{|P_1 - P_2 + N_1 Q_1 - N_2 Q_2 + T_1 - T_2|} \right\rceil. \tag{12}$$

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.

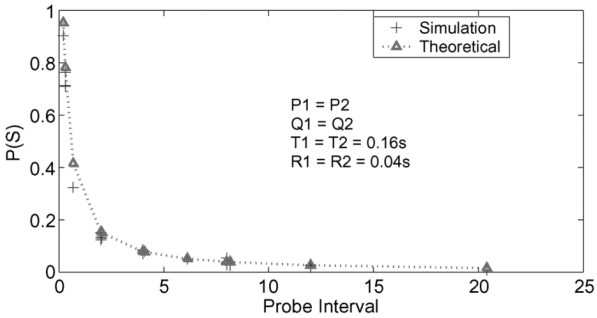


Fig. 9. Comparison of theoretical and simulation results for $P(S)$ for two identical overlay networks with different values of P.

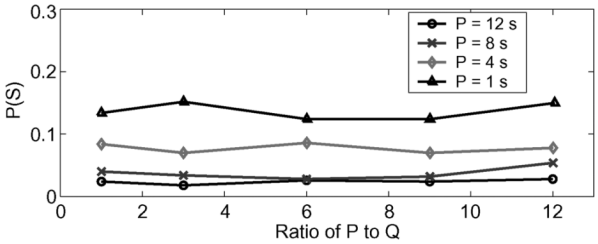


Fig. 10. $P(S)$ as a function of the ratio P/Q with different values of Q for a given value of P in two identical overlay networks.

VI. 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 V) versus that seen in simulation. We first consider two similar networks (i.e., all the parameters are identical), but vary the value of the probe interval. The results are given in Fig. 9. 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.

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 (7), (8), (9), (10)], 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. Fig. 10 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 [(12)], using our simulator we simulate oscillations in two synchronized overlays that are dissim-

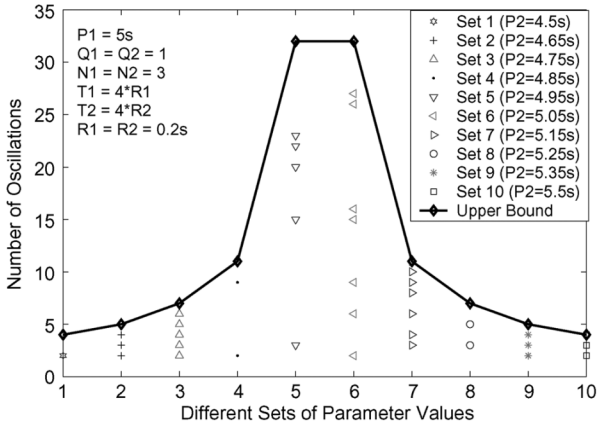


Fig. 11. Comparison: theoretical versus simulations for number of oscillations.

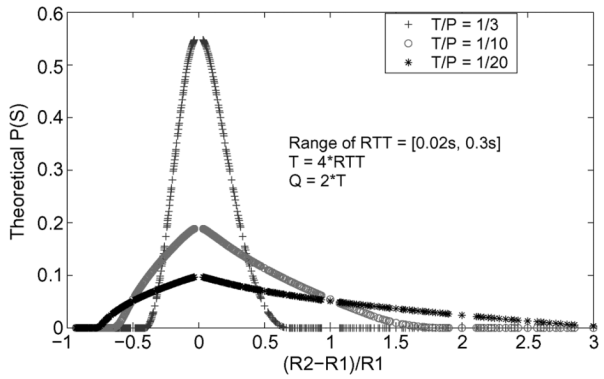


Fig. 12. $P(S)$ versus $(R_2 - R_1)/R_1$ for proportional parameter overlays with similar aggressiveness and varying RTT.

ilar (Fig. 11). 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 VII-B.

VII. 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? and (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?

A. Aggressiveness Factor and Probe Parameter Setting

We start with the simplest case of two overlays with identical parameter setting since it provides some insight about overlays in general. Recall from Section VI that for two identical overlays, we have $P(S) = T(2P - T)/P^2$. Hence, $P(S)$ 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., the overlay networks will definitely synchronize. If $P = 2T$ then $P(S) = 0.75$. To decrease the probability of synchronization to less than 0.05 (i.e., 5% chance of synchronization), we need to choose $P \approx 40T$.

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*, α, of an overlay network as the ratio of the timeout and probe interval, $\alpha_i = T_i/P_i$. Since $RTT \leq T \leq P$, hence $0 < \alpha \leq 1$. For two identical overlay networks, $P(S) = 2\alpha - \alpha^2$, which shows that as the networks increase their aggressiveness (as $\alpha \to 1$), $P(S)$ increases.

In many of our sample scenarios in the next section, we varied RTT between 20 ms and 300 ms to capture a variety of realistic Internet overlay paths that span either a small or large geographic distance. 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; and (ii) *Fixed* values, where P and Q are constants independent of T and RTT and therefore the same for all paths.

B. Results and Discussion

In Fig. 12, 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 that when both overlays are aggressive (e.g., $T/P = 1/3$), $P(S)$ can be as high as 55%. When both are nonaggressive (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.

As overlays are not widely deployed and their performance requirements are not yet well understood, it is not clear how

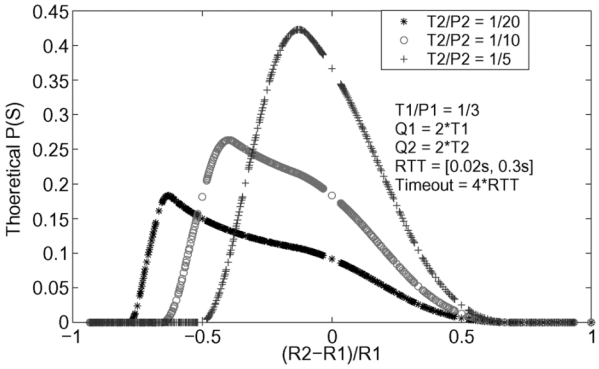


Fig. 13. Proportional parameter overlays with mixed aggressiveness and varying RTT.

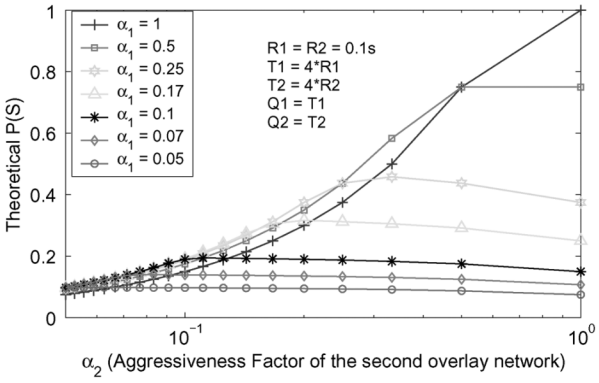


Fig. 14. Proportional parameters, mixed aggressiveness, and constant RTT.

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 Fig. 13, 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 nonaggressive ($\alpha_2 = 1/20$). We see that even in the case of one aggressive and one nonaggressive 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 nonaggressive parameters even if another overlay behaves aggressively. To further explore this hypothesis, we consider a wider variety of cases in Fig. 14.

Fig. 14 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. As the aggressiveness of both overlays increases, there is a higher chance of synchronization. This probability significantly decreases when the overlays are nonaggressive. This confirms that as long as one of the overlays is nonaggressive, 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 nonaggressive parameter settings of the other overlay. We defer the study of fairness to future work.

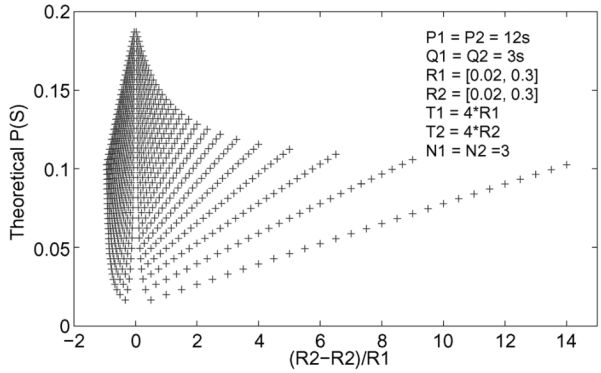


Fig. 15. Fixed parameter overlays (RON-like) with the same values of P and Q, and varying RTT.

We now look at the impact of using a fixed parameter approach to choosing P and Q. In Fig. 15, 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 ($R_1 = 20$ ms, $R_2 = 20$ ms), and achieves its maximum when both RTT values are large ($R_1 = 300$ ms, $R_2 = 300$ 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 nonaggressive (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 Fig. 16, we consider two fixed parameter overlays with different values of P and Q. We see similar behavior as in the previous case of fixed parameter overlays. We point out that in these two cases $P(S)$ is significant for a wider range of values on the x axis as compared to Fig. 12. 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 aggressiveness is $\alpha = T/P$, 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%).

We also examine the influence of Q on $P(S)$ and conclude that the influence of Q is far less significant compared to that of P or T. Due to lack of space we omit these results.

Finally, Fig. 17 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.

VIII. 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

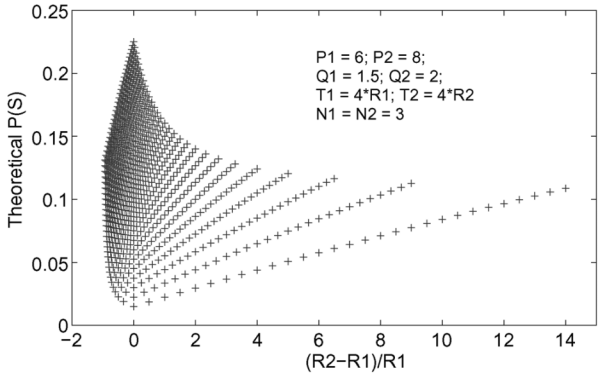


Fig. 16. Fixed parameter overlays with the different values of P and Q, and varying RTT.

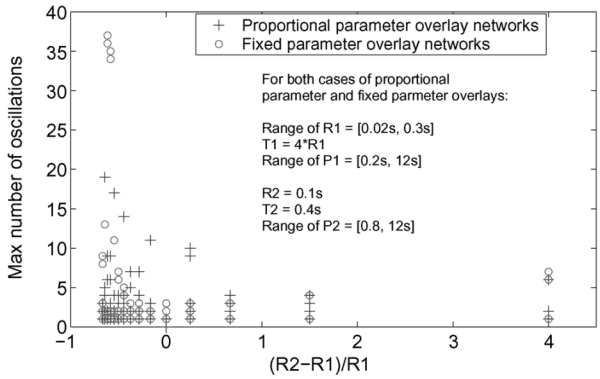


Fig. 17. Maximum number of oscillations as a function of relative RTT.

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 consideration when overlay networks are designed and configured.

One of the questions we explored was whether we can count on 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. With 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 nontrivial 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.

IX. Limiting the Impact of Race Conditions

To limit the impact of synchronization among multiple overlay networks we can take two approaches: (i) reduce $P(S)$ among overlays, and/or (ii) reduce the number of oscillations once the overlays get synchronized.

Reducing the Probability of Synchronization: Intuitively, one way to make it less likely that two overlays synchronize would be to add randomness into the probing procedure. The idea of adding randomness was illustrated to help in the case of periodic routing protocols in [8]. Here we study the resulting behavior of overlays when we add randomness to their probing parameters. The hope is that these random values will drive the reaction times of the overlays far apart, thus reducing the possibility of synchronization.

Fig. 18 shows the effect of adding randomness into overlay probing parameters on the likelihood of overlay network synchronization in our simulator. The parameters values for the two overlay networks are shown in the figure. In the results (in Fig. 18), the value of P is randomized by a certain percentage of its own value (for example, 10%, 20%, etc.). Let us consider the case of adding 10% randomness. In this case, we add a random value chosen from a uniform distribution between 0 and $0.1P$ to the original value of P. Although we use a similar strategy to add randomness to both overlays, the random values chosen for the two overlays are independent of each other.

In Fig. 18, we can see that adding randomness to the probe interval, P, does not help in decreasing the $P(S)$. Based on the

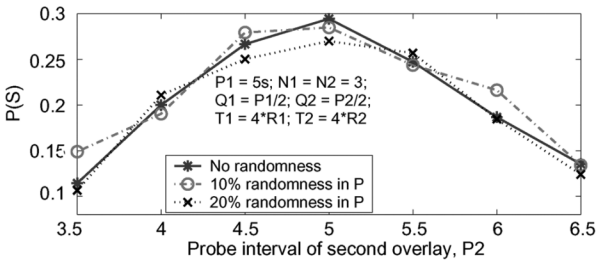


Fig. 18. $P(S)$ for randomized probing parameters.

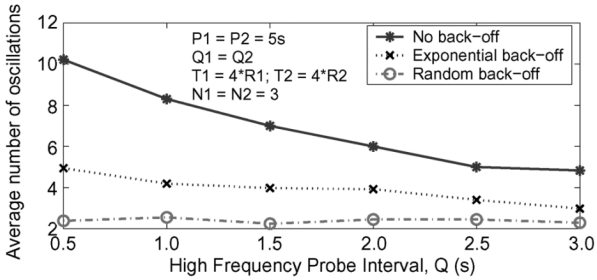


Fig. 19. Effect of using back-off technique on the number of oscillations.

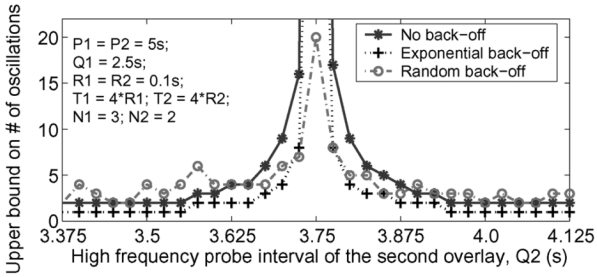


Fig. 20. Effect of using back-off technique on the number of oscillations when the overlays are using different values for N.

fact that $P(S)$ depends on the difference terms (like $P_1 - P_2$ and $N_1Q_1 - N_2Q_2$ in (3), (6)–(10)), the randomness added to the same parameters in two overlays could either increase or decrease the value of $P(S)$. We have repeated these experiments by adding randomness to other probing parameters (i.e., Q and T), but found very similar results (not shown due to space constraints). Hence, adding randomness does not always ensure that two overlays are less likely to synchronize.

Reducing the Number of Oscillations: In order to reduce the number of oscillations after a synchronization event, we propose an approach based on the well-known behavior of TCP. Whenever a flow using TCP experiences a packet loss due to congestion the protocol backs off from using an aggressive packet transfer rate. Typically this back-off occurs at an exponential rate to reduce the impact of congestion. In our case of multiple overlays, we propose to use a similar back-off technique where an overlay network successively increases the reaction time each time it decides to switch routes between the same source and destination nodes (if the reactions occur in a small time interval). In other words, this is similar in spirit to dampening i.e., to slow down the reaction time of a protocol so as to avoid responding too quickly. Note that the back-off technique is also similar to the idea of nonaggressive probing. The main difference is that while using nonaggressive probing, the parameter (or timer) values are always large, but while using back-off strategy the parameter values are increased only when oscillations are detected.

Fig. 19 shows the effect of using back-off techniques on the number of oscillations for two synchronized overlays. In these simulations, we use two different approaches to accomplish back-off: (i) "Deterministic" exponential back-off where the amount of time that an overlay waits to change its path increases exponentially (but in a deterministic manner) when the overlay has changed its path in the recent past. In Fig. 19 both overlays double the reaction time when the overlay has changed its path in the previous 20 seconds. (ii) Random back-off where the overlays wait for a random amount of time before deciding to move to an alternate path.

We can clearly see in Fig. 19 that, on an average, using random back-off reduces the impact of synchronization more than exponential back-off. Since both overlays use the same back-off parameters there is a higher chance that the overlays will remain synchronized for a longer period when they use exponential back-off strategy. However, while using random back-off strategy, both overlays are more likely to wait for different amounts of time before reacting, and hence it results in reducing the impact of synchronization significantly.

Fig. 20 explores the impact of using back-off technique on the *upper bound* of the number of oscillations. When the number of oscillations is low, the two back-off techniques perform similarly. Also, we see a huge spike at 3.75 on the x-axis, illustrating extreme sensitivity to a particular parameter setting. From (12), we can see that there are several combinations of parameter settings that will make the denominator zero, or close to zero, causing such spikes. Since overlays today do not coordinate their parameter choices, uncovering such situations is beyond the ability of an overlay network. Hence, there are many isolated cases where the race conditions we describe could be severe. For these cases, random backup-off appears more effective in quickly curbing the spike by moving the parameters away from the sensitive setting.

X. Conclusions and Future Directions

We have shown that co-existing overlay networks can experience race conditions, affecting both the overlay and nonoverlay traffic in the ISP network. We analytically modeled the probability of synchronization between overlay networks and showed that overlay networks could get synchronized even when they use dissimilar sets of probing parameters. We also explored strategies to reduce the impact of race conditions, which can be used as a guideline for designing or deploying future overlay networks.

Further studies are required to gain a comprehensive understanding of *stop triggers* for oscillations. We plan to investigate the following: (i) What are the various possible stop triggers, and the frequency of occurrence for each of them? (ii) What is the stop trigger for a given set of oscillations? (iii) How do network topologies and operational conditions affect the occurrence of stop triggers? Since this is the first work of its kind, we made many assumptions in our model and analysis, but we plan to relax these assumptions in future work.

APPENDIX I
EXTENSION OF THE ANALYTICAL MODEL

We now illustrate how our existing framework can be extended to scenarios in which more than two overlays exist. When m (where $m > 2$) overlays coexist, there are a variety of questions from different perspectives that one can ask about combinations and subsets of overlays that synchronize. For example, we may ask "What is the probability that all m overlays synchronize?" If m were very large, the probability that *all* of these overlays synchronize simultaneously is likely to be very small. It seems more meaningful to ask "What is the probability that more than a large number, say h, of overlays synchronize?" (where $h \leq m - 1$). The synchronization of a large number (but not all) of overlays, is more likely than synchronization of all of them. We call this *global synchronization*, and leave the choice of h up to the questioner. Another interesting perspective is from that of a single overlay who may like to ask "What is the probability that I will experience a synchronization event with g (or with less than or equal to g) other overlays?". If g is a small number (i.e., $g < h$), then we refer to this as *local synchronization*. In Section V we computed the case when $g = 1$.

We first address the following question: "Given the coexistence of m overlays $O_1, O_2, \ldots, O_m$, what is the **probability that *all* m synchronize**?" This can be viewed as a network-wide perspective on global synchronization. We show how our analytical framework can be extended to the case of m coexisting overlays when all the overlays satisfy the conditions outlined in Section IV-A.

Let $f_1, f_2, \ldots, f_m$ be the times at which the final high frequency probes are sent by overlays $O_1, O_2, \ldots, O_m$ after a path performance degradation event is detected. Let $\mathtt{S} = \{s_1, s_2, \ldots, s_m\}$ be an ordering of $\{1, 2, \ldots, m\}$ such that $f_{s_i} < f_{s_j}$ if $i < j$. Similar to the conditions for synchronization of two overlays outlined in (1) and (2), the condition for the synchronization of m overlays is given by a set of $m - 1$ inequalities:

$$0 < f_{s_m} - f_{s_i} < T_{s_i} \quad \text{for } i = 1, 2, \ldots, m-1. \tag{13}$$

The above equations represent the required conditions for m coexisting overlays to synchronize. The ordering in $\mathtt{S}$ depends on the actual times at which the final high frequency probes are sent out by different overlays and hence there could be several different combinations. However, from (13), we can clearly see that the order in which the final high frequency probes are sent out by the first $m - 1$ overlays does not affect the synchronization condition. The only requirement is that the final high frequency probe from O_{s_m} should be sent out after all the other overlays have sent their final high frequency probes. In other words, $\mathtt{S} = \{1, 2, 3, 4 \ldots m\}$ results in the same synchronization condition as $\mathtt{S} = \{2, 1, 3, 4 \ldots m\}$ or $\mathtt{S} = \{4, 2, 1, 3 \ldots m\}$ as long as $s_m = m$. Hence, the different possible combinations of $\mathtt{S}$ depends on the value of s_m. Note that s_m can take any one of the m values ($\{1, 2, 3 \ldots m\}$) and hence there can be m different combinations of $\mathtt{S}$. This implies that in a system with m overlays there are m independent synchronization conditions for all the overlays to synchronize. In each of these synchronization conditions there will be $m-1$ sets of inequalities (as we can see in (13)) that are required to be satisfied in order for overlays to synchronize.

To illustrate the above reasoning let us first consider the case where there are only two overlays, i.e., $m = 2$. In this case there should be m, i.e., 2 independent synchronization conditions and $m-1$, i.e., 1 set of inequality in each of the synchronization conditions. We can clearly see that this is in fact true based on (1) and (2). Similarly in the case of three overlays there are 3 independent synchronization conditions with 2 sets of inequalities in each of these conditions.

Equation (13) can be rewritten as

$$\begin{aligned} 0 &< t_{s_m} + N_{s_m} Q_{s_m} - t_{s_i} - N_{s_i} Q_{s_i} \\ &< T_{s_i} \quad \text{for } i = 1, \ldots, m-1 \\ d_{im} &< t_{s_m} - t_{s_i} < T_{s_i} + d_{im} \quad \text{for } i = 1, \ldots, m-1 \end{aligned} \tag{14}$$

where $d_{im} = N_{s_i} Q_{s_i} - N_{s_m} Q_{s_m}$. The set of inequalities in (14) involves m overlays and contains m uniform random variables $(t_{s_1}, t_{s_2}, t_{s_3}, \ldots, t_{s_m})$. The range of these variables are similar to the definitions in Section V-A, i.e., the range of t_{s_i} is $[-R_{s_i}/2, P_{s_i} - R_{s_i}/2]$ for $i = 1, 2, \ldots, m$. Hence, this system of inequalities represents a m-dimensional space and the range of values for each random variable ensures that the allowed values for the variables are bounded inside a finite volume (i.e., $\mathtt{v} = P_{s_1} P_{s_2} \ldots P_{s_m}$) enclosed by the hyperplanes $t_{s_i} = -R_{s_i}/2$ and $t_{s_i} = P_{s_i} - R_{s_i}/2$ for $i = 1, 2, \ldots, m$.

A key observation in the inequalities defined in (14) is that every inequality depends on only two variables. In other words, the two hyperplanes defined by $t_{s_m} - t_{s_i} = d_{im}$ and $t_{s_m} - t_{s_i} = T_{s_i} + d_{im}$ are parallel to all other axes except the axes in the direction of t_{s_m} and t_{s_i}. Note that with only two varying dimensions for each inequality, the situation is similar to the one described in Section V-A. There are 9 possible scenarios (similar to Fig. 7) that could occur for each inequality and there are $m - 1$ such inequalities in a synchronization condition. Hence, for each of the m synchronization conditions there are 9^{m-1} possible scenarios, each one with its own *compensation volumes*. The probability of synchronization for a particular synchronization condition is the ratio of the volume enclosed by the hyperplanes defined by the inequalities in (14), and the total volume $\mathtt{V}$. Since the synchronization conditions are independent of each other, the *total* probability of synchronization of m overlays is the sum of the probabilities of synchronization of all the m synchronization conditions. We denote the probability of synchronization of m overlays, $O_1, O_2, \ldots, O_m$, as $P_{1,2,\ldots,m}$.

We now consider local synchronization from the perspective of a particular overlay O_i. We assume a system with m overlays where all the overlays satisfy the conditions outlined in Section IV-A. **The probability that overlay O_i synchronizes with just one other overlay network** can be expressed as

$$P^{i,1} = \sum_{\substack{j=1 \\ j \neq i}}^{m} \left[P_{i,j} \prod_{k \neq i,j} (1 - P_{i,k})(1 - P_{j,k}) \right] \tag{15}$$

where $P_{i,j}$ is the probability of synchronization between overlays O_i and O_j as computed in Section V. The term $(1 - P_{i,k})$ represents the probability that the overlay O_i does not synchronize with O_k. The second part of the above equation represents the probability that O_i and O_j do not synchronize with any other overlay networks in the system.

Generalizing the above equation, the **probability that O_i synchronizes with exactly r other overlay networks** can be written as

$$P^{i,r} = \sum_{\substack{j_1=1 \\ j_1 \neq i}}^{m} \sum_{\substack{j_2=j_1+1 \\ j_2 \neq i}}^{m} \cdots \sum_{\substack{j_r=j_{r-1}+1 \\ j_r \neq i}}^{m} [P_{i,j_1,\ldots,j_r}\Phi] \quad (16)$$

$$\Phi = \prod_{\substack{k \neq i \\ k \neq j_1,\ldots,j_r}} [(1-P_{i,k})(1-P_{j_1,k})\ldots(1-P_{j_r,k})] \quad (17)$$

where $P_{i,j_1,\ldots,j_r}$ represents the probability of synchronization of O_i with r other overlays (i.e., the result from the discussion earlier in this section where the total number of overlays is $r+1$).

We can now define the **probability that O_i synchronizes with less than g other overlays**. Using (16) and (17), we can express this probability, called the probability of local synchronization of O_i, namely P^i_{LS}, as

$$P^i_{LS} = \sum_{k=1}^{g} P^{i,k}. \quad (18)$$

From the perspective of a single overlay, one can also ask "what is the probability that I experience a global synchronization event?" Defined to be the probability that an overlay synchronizes with more than h overlays, we can write this as

$$P^i_{GS} = \sum_{k=h}^{m-1} P^{i,k}. \quad (19)$$

Our initial exploration with these models indicates that global synchronization of a large number of overlays becomes quite small. However, from the perspective of a single overlay, when there are many co-existing overlays, the likelihood of local synchronization with any one of them is even higher than the probabilities presented in our results herein.

REFERENCES

[1] Akamai. [Online]. Available: http://www.akamai.com

[2] D. Andersen, H. Balakrishnan, M. Kaashoek, and R. Morris, "Resilient overlay networks," in *Proc. ACM Symp. Operating Systems Principles (SOSP)*, Oct. 2001, pp. 131–145.

[3] M. Castro, P. Druschel, A. Kermarrec, A. Nandi, A. Rowstron, and A. Singh, "SplitStream: High-bandwidth multicast in cooperative environments," in *Proc. ACM Symp. Operating Systems Principles (SOSP)*, Oct. 2003, pp. 298–313.

[4] S. Rhea, D. Geels, T. Roscoe, and J. Kubiatowicz, "Handling CHURN in a DHT," in *Proc. USENIX ATC*, Boston, MA, Jun. 2004.

[5] S. Savage, A. Collins, E. Hoffman, J. Snell, and T. Anderson, "The end-to-end effects of Internet path selection," in *Proc. ACM SIGCOMM*, Aug. 1999, pp. 289–299.

[6] K. Sripanidkulchai, B. Maggs, and H. Zhang, "An analysis of live streaming workloads on the Internet," in *Proc. ACM Internet Measurement Conf. (IMC)*, Oct. 2004, pp. 41–54.

[7] R. Keralapura, N. Taft, C. N. Chuah, and G. Iannaccone, "Can ISPs take the heat from overlay networks?," in *Proc. ACM HotNets-III*, San Diego, CA, Nov. 2004.

[8] S. Floyd and V. Jacobson, "The synchronization of periodic routing messages," in *Proc. ACM SIGCOMM*, Sep. 1993, pp. 33–44.

[9] R. Keralapura, C. N. Chuah, N. Taft, and G. Iannaccone, "Can overlay networks inadvertently step on each other?," in *Proc. IEEE ICNP*, Nov. 2005, pp. 201–214.

[10] L. Qiu, Y. Yang, Y. Zhang, and S. Shenker, "On selfish routing in Internet-like environments," in *Proc. ACM SIGCOMM*, Aug. 2003, pp. 151–162.

[11] Y. Liu, H. Zhang, W. Gong, and D. Towsley, "On the interaction between overlay routing and traffic engineering," in *Proc. IEEE INFOCOM*, Mar. 2005, pp. 2543–2553.

[12] A. Nakao, L. Peterson, and A. Bavier, "A routing underlay for overlay networks," in *Proc. ACM SIGCOMM*, Karlsruhe, Germany, Aug. 2003.

[13] T. Leighton, "The challenges of delivering content and applications on the Internet," presented at the NSDI Keynote, May 2005.

[14] G. Iannaccone, C.-N. Chuah, S. Bhattacharyya, and C. Diot, "Feasibility of IP restoration in a tier-1 backbone," *IEEE Network*, vol. 18, no. 2, pp. 13–19, Mar. 2004.

[15] R. Keralapura, C. Chuah, G. Iannaccone, and S. Bhattacharyya, "Service availability: A new approach to characterize IP backbone topologies," in *Proc. IWQoS*, Jun. 2004, pp. 232–241.

[16] B. Fortz and M. Thorup, "Internet traffic engineering by optimizing OSPF weights," in *Proc. IEEE INFOCOM*, Mar. 2000, pp. 519–528.

[17] A. Medina, N. Taft, K. Salamatian, S. Bhattacharyya, and C. Diot, "Traffic matrix estimation: Existing techniques and future directions," in *Proc. ACM SIGCOMM*, Pittsburgh, PA, Aug. 2002.

[18] G. Ash, *Dynamic Routing in Telecommunication Networks*. New York: McGraw-Hill, 1997.

Ram Keralapura received the B.E. degree in electrical engineering from Bangalore University, Bangalore, India, in 1998 and the M.S. degree in computer science from the University of Alabama, Huntsville, in 2000. He is currently working towards the Ph.D. degree in electrical and computer engineering at the University of California at Davis.

His current research interests are in designing, building, analyzing, and managing distributed networks. He is also interested in Internet routing, traffic engineering, security, and overlay/P2P networks.

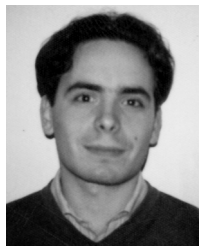
Chen-Nee Chuah (SM'06) received the B.S. degree in electrical engineering from Rutgers University, New Brunswick, NJ, and the M.S. and Ph.D. degrees in electrical engineering and computer sciences from the University of California at Berkeley in 1997 and 2001, respectively.

She is currently an Associate Professor in the Electrical and Computer Engineering Department at the University of California at Davis. Her research interests include Internet measurements, overlay/peer-to-peer systems, network security, and wireless/mobile networking.

Dr. Chuah received the NSF CAREER Award in 2003.

Nina Taft (M'94) received the B.S. degree from the University of Pennsylvania, Philadelphia, in 1985, and the M.S. and Ph.D. degrees from the University of California at Berkeley in 1990 and 1994, respectively.

She is a Senior Researcher at Intel Research, Berkeley, CA. From 1995 to 1999, she worked at SRI International; and from 1999 to 2003, she was a member of the IP Group at Sprint Advanced Technology Labs. Her interests are in traffic characterization and modeling, performance evaluation, network design, and enterprise network security.



Gianluca Iannaccone received the B.S. and M.S. degrees in 1998, and the Ph.D. degree in 2002, all in computer engineering from the University of Pisa, Pisa, Italy.

He joined Sprint in October 2001 working on network performance measurements, loss inference methods, and survivability of IP networks. In September 2003, he joined Intel Research, Cambridge, U.K. His current interests include system design for fast prototyping of network data mining applications, privacy-preserving network monitoring, and routing stability for overlay networks.