Dissemination of Address Bindings in Multi-substrate Overlay Networks

Jörg Liebeherr Majid Valipour
Department of Electrical and Computer Engineering
University of Toronto, Canada

***Abstract*—Self-organizing overlay networks have emerged as a new paradigm for providing network services. While most overlay networks are built over a single substrate network (mostly, the Internet), recently the construction of overlay networks over multiple heterogeneous substrate networks has received increased attention. Such networks seek to interconnect mobile or fixed devices using a diverse set of networking modalities. Here, a key challenge arises from the more complex address bindings, where a single logical identifier is bound to multiple substrate addresses. In this paper, we evaluate the design and inherent trade-offs of mechanisms for exchanging information on address bindings in a multi-substrate overlay network. The evaluation is done using measurement experiments of an overlay network software system. The measurement data provides insights into the scalability of dissemination methods. An important finding is that gossip-based address dissemination is less effective than an on-demand dissemination of address bindings.**

I. Introduction

Self-organizing application-layer overlay networks have emerged as a powerful paradigm for deploying network services. Elements of self-organizing networks have been incorporated in peer-to-peer file sharing networks, content distribution and streaming networks, wireless mesh and wireless sensor networks, and vehicular networks. An overlay network is a virtual network built on top of an existing *substrate network*, also called *underlay network*. A link in the overlay network corresponds to a path of one or more links in the substrate network.

Application-layer overlay networks emerged in the late 1990s for rapid deployment of services not globally available on the Internet, such as multicast [1], distributed lookup [5], fault resilient delivery [4], and many more. The networks are often self-organizing in the sense that they create and maintain a network topology in a distributed fashion without need for manual or central configuration or management.

Application-layer overlay networks generally assume that the Internet provides a permanently available and universally accessible substrate network. More recently, researchers have tried to relax this assumption and considered self-organizing application-layer overlay networks over multiple heterogeneous substrate networks, where any data link, network layer, or even overlay network can constitute a separate substrate network [7], [8]. The potential of such networks is a seamless interconnection of applications running on mobile and fixed endsystems using a diverse set of networking modalities without requiring access to a particular network infrastructure.

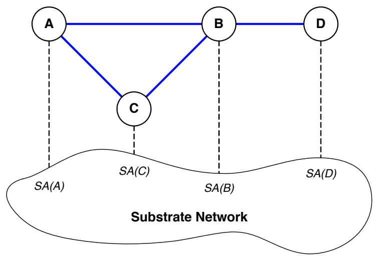
Multi-substrate overlay networks impose many challenges with respect to addressing, discovery, network topology maintenance, and many more. In this paper, we address a key problem in multi-substrate networks that arises from the complex address bindings, where a single local identifier for a node in the overlay network is bound to multiple substrate addresses. In a multi-substrate network, a node may be connected to a large number of substrate networks. Since nodes can be neighbors in the overlay network only if they share at least one common substrate, the challenge lies in building a connected network graph that contains all devices that wish to establish communication.

In this paper, we devise distributed protocol mechanisms that assist nodes with learning about remote nodes accessible through alternative substrates. We present and evaluate a set of mechanisms, referred to as *cross-substrate advertisement (CSA)*, by which overlay nodes exchange address information of one substrate network across another substrate network. We consider different approaches for address dissemination, including a gossip-based approach, proactive dissemination, and on-demand address resolution. The discussed CSA methods are implemented in an existing software system for self-organizing networks [12], and evaluated on a local cluster of servers. Our experiments show that properly designed CSA methods can scale to large networks, and quickly react to changes of the network topology. Our experiments indicate that an on-demand approach appears superior in terms of effectiveness and overhead.

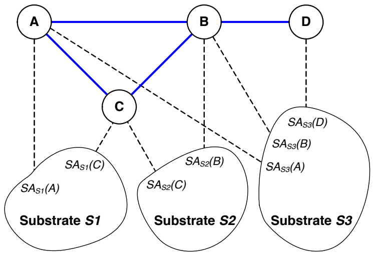
The remaining paper is structured as follows. In Section II, we discuss address bindings in single- and multi-substrate overlay networks. In Section III, we motivate the need for cross-substrate advertisements. In Section IV, we discuss methods for disseminating address bindings. In Section V, we present the experimental evaluation of CSA methods. Finally, we present conclusions in Section VI.

II. Address Bindings in Overlay Networks

We refer to an overlay node, or simply *node*, as an application that participates in an overlay network. In a self-organizing overlay network, nodes cooperate to establish the network topology graph. Application data in the overlay network is then routed along the edges of the graph. Establishing and maintaining an overlay topology involves (1) a discovery (rendezvous) process, by which a node not connected to the overlay network can find nodes that are part of the overlay



(a) Single-substrate overlay network.



(b) Multi-substrate overlay network.

Fig. 1. Bindings of overlay and substrate addresses.

topology; (2) a neighbor selection method, by which a node determines the subset of reachable nodes that become its neighbors in the overlay topology; and (3) a routing protocol that determines the forwarding paths for application data in the overlay network. In many structured overlay networks, e.g., Chord [5], the forwarding table is implied once the nodes have selected their neighbors in the topology.

Each node of an overlay network has an identifier, called the *overlay address*, which, in some topologies, may also play the role of a locator for the routing protocol. Dependent on the overlay network topology, overlay addresses can be binary strings [5], coordinates [16], or arbitrary identifiers [1]. The overlay address serves as a unique identifier in the overlay network. When the overlay address of a node is derived from its geolocation, uniqueness of the overlay address can be asserted only for given periods of time.

Each node has one *substrate address* for each attachment point to a substrate network. In application-layer overlay networks with the Internet as substrate network, a substrate address generally consists of an IP address and a port number. We refer to the association of an overlay address with a substrate address as an *address binding*.

A. Address Bindings in Single-substrate Overlays

In a single-substrate scenario, each overlay address of a node is associated with one substrate address, resulting in a one-to-one binding. Fig. 1(a) depicts such an overlay network with four nodes with overlay addresses A, B, C, and D. Nodes connected by a link are neighbors in the overlay topology. Each node has an attachment to the same substrate network, with substrate addresses $SA(A)$, $SA(B)$, $SA(C)$, and $SA(D)$, respectively.

When node A sends a message to its neighbor B, the message is addressed to $SA(B)$ in the substrate network. With a single substrate, any two nodes can potentially become neighbors in the overlay. (This may not hold in a wireless network, since nodes can communicate only if they are within transmission range of their radios. Such a situation can be viewed either as a single, but disconnected substrate network, or as multiple connected substrate networks.)

Since nodes must be able to exchange messages with their neighbors, each node should hold the address bindings of its neighbors. Nodes learn about potential new neighbors by exchanging address bindings. In a single-substrate network, a node that receives a message can often infer the address binding of the sending node. For example, when A sends a message to B, B can extract $SA(A)$ from the source address in the encapsulating header to create the address binding $\langle A; SA(A)\rangle$.

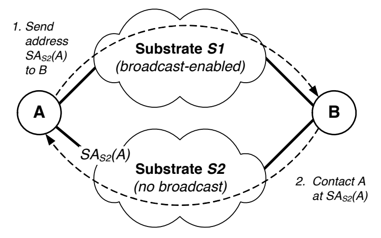
B. Address Bindings in Multi-substrate Overlays

In a multi-substrate network, nodes can communicate with each other directly only if they share a common substrate network. Nodes may have multiple *address bindings*, with one substrate address for each connected substrate network, resulting in a one-to-many mapping of overlay to substrate addresses.

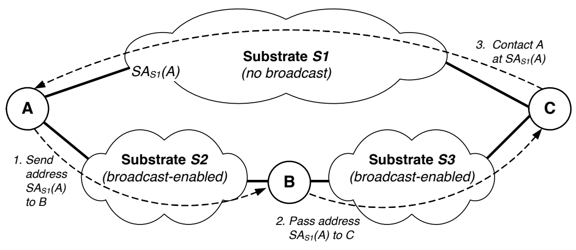
In Fig. 1(b) we depict the topology of a multi-substrate overlay network with three substrate networks $S1$, $S2$, and $S3$. Denoting by $SA_{S1}(A)$ the substrate address of node A on substrate $S1$, the complete address bindings of node A are given by $\{\langle A; SA_{S1}(A)\rangle, \langle A; SA_{S3}(A)\rangle\}$, which is referred to as A's *(substrate) address list*. We assume that nodes specify preferences for the substrates in their address list. When nodes are connected to each other by more than one substrate network (e.g., nodes A and B are both connected to substrates $S1$ and $S3$), the nodes select one of the available substrate based on their configured preference. While, in a single substrate network, an address list can be extracted by inspecting encapsulation headers of incoming messages, this is not the case in a multi-substrate network. Thus, for nodes to take full advantage of multiple substrates, additional mechanisms are required by which nodes can disseminate address lists. In the simplest case, each message sent between overlay nodes contains the complete address list of the sender, however, this may incur unreasonable overhead. In practice, the dissemination of address lists must trade-off the benefit of having available address list information with the cost to disseminate the lists. The purpose of this study is to explore these trade-offs, and provide insights into the design of effective dissemination methods.

C. Related Work

While the problem addressed in this paper, i.e., the dissemination of substrate address information across multiple substrates, has not been studied before, there exists an extensive literature on networks over heterogeneous substrates. Traditionally, network architectures have dealt with heterogeneous substrate networks by providing multiple one-to-one address bindings. For example, in the IP network architecture a multi-homed host has a different IP address for each configured network interface. The resulting problems, e.g., the need to



(a) Example 1: Exchange over connected broadcast substrate network



(b) Example 2: Exchange over multiple substrate networks

Fig. 2. Exchange of substrate addresses across substrate networks.

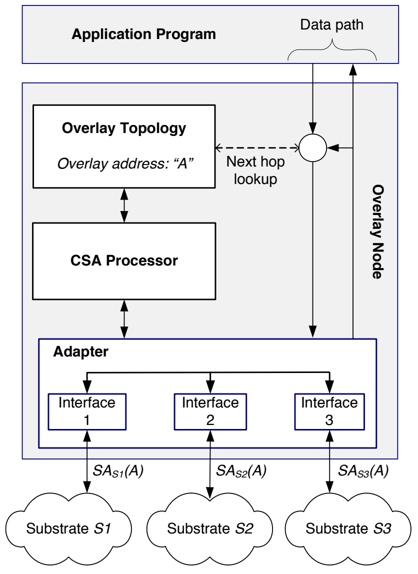


Fig. 3. Overlay Node.

separate identifier and locator functions of an IP address, have been extensively studied, and numerous solutions have been proposed, e.g., [13], [14]. A different set of works address the heterogeneity due to private IP networks and the resulting loss of end-to-end connectivity, e.g., [9]. Plutarch [2] defines a framework for interconnecting substrate networks and address realms by making heterogeneity explicit, where middleboxes perform translation between address realms. A self-organizing approach for interconnecting applications across heterogeneous collections of substrate networks is presented in [8], which uses a multi-substrate enabled distributed hash table (DHT) for routing. The approach has been further developed in SpoVNet [7], which uses the DHT only for lookup of a locator, and routes traffic separately.

III. Cross Substrate Advertisement

We are interested in suitable protocol mechanisms for disseminating substrate address information in a multi-substrate overlay network across a diverse set of broadcast and non-broadcast substrate networks, which we refer to as *cross-substrate advertisement* or CSA. Such mechanisms assist overlay nodes with learning about nodes accessible through alternative substrates. We only consider fully distributed solutions without the need for special purpose nodes or access to an external service.

A. Motivation for CSA

The following examples motivate the need for protocol mechanisms for exchanging address bindings in multi-substrate overlay network.

Example 1. In Fig. 2(a), two nodes, A and B, are both attached to substrate networks $S1$ and $S2$. Only $S1$ supports a broadcast delivery, while $S2$ is a non-broadcast network. Suppose nodes A and B prefer to connect in the overlay via $S2$, possibly because it offers a higher capacity or a higher level of security. In this scenario, A and B can rendezvous over $S1$ with broadcast messages. Then A can use $S1$ to send its substrate address in $S2$, $SA_{S2}(A)$, to B (either by unicast or broadcast). Once B receives A's address in $S2$, it can contact A using the preferred substrate.

Example 2. In Fig. 2(b), there are three nodes (A, B, C) and three substrate networks $(S1, S2, S3)$, where $S1$ is a non-broadcast network. Suppose, A has a neighborhood relation established with B over substrate $S2$, and B is a neighbor of C over substrate $S3$. Further, the preferences of nodes A and C are such that they rather connect over $S1$, however, A and C require each others' substrate addresses in $S1$ to establish communications over this non-broadcast substrate. In this scenario, CSA can be used to deliver the substrate address of A on substrate $S1$, $SA_{S2}(A)$, to node C across substrates $S2$ and $S3$. This scenario is more difficult than Example 1, since it requires the support of the intermediate node B.

An important design decision is whether CSA should be realized as part of the protocol that maintains the overlay network topology, or as a separate functional component that is shared by all overlay topology protocols. We consider CSA as a set of common protocol mechanisms that support any overlay topology, with the rationale that the advantages of a topology independent design outweigh possible drawbacks, e.g., not being able to optimize the exchange of address bindings for a specific topology.

B. Overlay Node with CSA Processor

Fig. 3 illustrates the main functional components of an overlay node in an application-layer overlay network. The overlay node appears as middleware between an application program and a set of substrate networks. Any specific design or implementation of an overlay node may differ from this depiction, however, the same basic functions are found in every implementation of an overlay node.

The overlay node plays the role of an application-layer router. For each outgoing application message, it performs a lookup for the next hop and then forwards the message using one of the connected substrates. For an incoming application message, the overlay node determines if the message should be forwarded to other nodes and/or delivered to the application. The overlay topology component comprises the control path of the overlay node, which includes discovery, neighbor selection, and the forwarding table. For each connected substrate network, the overlay node has a component, referred to as *interface*, for access to the substrate network and encapsulation/decapsulation of messages. Each interface is associated with a substrate address. In Fig. 3, the node has overlay address A, and three interfaces with substrate addresses $SA_{S1}(A)$, $SA_{S2}(A)$, and $SA_{S3}(A)$. In the figure, the interfaces are collected in an *adapter* component.

In our design, the tasks of cross-substrate advertisement are performed by a single component, referred to as *CSA processor*, which is placed between the overlay topology component and the adapter, as shown in Fig. 3. The CSA processor inspects incoming and outgoing messages to and from the overlay topology component, and initiates the resolution of address bindings.

CSA processors only need to exchange two types of messages: (1) a request for an address list, and (2) an update containing one or more address lists. Updates of address lists can be sent as standalone messages, e.g., in response to a request, or piggybacked to an outgoing message from the overlay node. The rules for transmitting request and update messages are discussed in the next section.

We have implemented a CSA processor as an enhancement to an existing (open source) software system for single-substrate application-layer overlay networks, called HyperCast [12]. We have modified the HyperCast software to support multi-substrate overlay networks. The modified implementation follows the design shown in Fig. 3. Details of the CSA processor specification, such as packet formats, and its implementation are given in [17].

IV. Methods for Address List Dissemination

We distinguish two methods for exchanging address bindings. We refer to a *direct address list exchange* when two nodes connected to the same substrate network exchange their address lists with each other. This is the scenario depicted in Fig. 2(a). In a *relayed (indirect) address list exchange* a node forwards address bindings for other nodes in the overlay network. The scenario in Fig. 2(b) depicts a relayed address list exchange, where B relays address information of A to C.

Since a direct address exchange is relatively straightforward, we will not discuss it in detail. Essentially, a node can attach its address list to each outgoing message. Alternatively, whenever a node receives a message from a remote node, it can check whether it has recent information on the address list of the sender of the message, and, if required, request the address list. In a broadcast network, there is the additional option to broadcast a request for an address list, similar as in ARP [15]. In the following we discuss CSA methods for a relayed address list exchange.

A. Gossip Communication

An obvious candidate for disseminating address binding information is a gossip protocol. In gossip communication, a piece of information, originally known by a single node is spread through the entire network. Each node that holds the information periodically exchanges it with a randomly selected node. Protocols for gossip communication have been extensively studied, and we refer to [10] for a survey. With gossip, each overlay node periodically sends one or more randomly selected address list(s) to one or more randomly selected destination(s). Without memory constraints, all nodes eventually obtain the address lists of all nodes in the network.

While the concept of gossip communication is simple, running an efficient gossip protocol can be difficult. In large networks, nodes may not be able to store the complete set of address lists, thus, requiring a replacement strategy of stored address entries. Further, the frequency of gossiping, the amount of data gossiped, and the destination set should adapt to the number of nodes in the overlay network.

For our evaluation of CSA methods, we use a basic gossip protocol, which does not require information on the size or topology of the network. The gossip protocol operates in rounds, where the time between rounds, the *gossip interval*, is constant. In each round, a node selects one address list as destination and transmits to this destination a set of address lists (By default, one address list is transmitted). The selection of destinations is random, but weighted to favor newly added nodes. Also, the more an address list has been gossiped in the past, the less likely it will be gossiped in future rounds.

A drawback of gossip communication is that it does not take into consideration whether a disseminated piece of information is actually needed by the receiver. Conversely, there is no guarantee that an address list of a remote node is available when needed.

B. Protocol-driven Dissemination

In virtually all protocols for maintaining an overlay topology, neighboring nodes advertise information about other (non-neighbor) nodes to each other. Such third-party node advertisements are used to learn about other nodes in the network, and identify potential new neighbors in the overlay topology. Such considerations suggest to attach substrate addresses to messages containing third-party node advertisements. This is referred to as *protocol-driven dissemination*. For example, in Fig. 2(b), node B is a neighbor of A, and therefore knows its address lists (due to a direct address exchange). If B sends a message to C containing a third-party advertisement of A and attaches the address list of A, node C obtains the substrate address needed to send a message to node A using $S1$.

We consider two methods of protocol-driven dissemination, one is pro-active and the other operates in an on-demand fashion.

Push. In the pro-active approach, referred to as *Push*, the complete address list of an advertised node is piggybacked to each third-party node advertisement. With *Push*, nodes need not maintain state information about cross-substrate advertisements, however, attaching a complete address list to each third-party node advertisement results in redundant transmissions.

Pull. In the on-demand approach, referred to as *Pull*, address lists must be explicitly requested by a node. When a node wants to send a message to an advertised node, i.e., a node whose overlay address was obtained by receiving a third-party node advertisement, it sends a request for the address list to one of its neighbors. When the node receiving a request holds the requested address list, it replies to the requesting node. Otherwise, the receiving node itself issues a request to resolve the address list. In this fashion, the request is iterated until a node can respond to the query.

There are two variations of the *Pull* approach. One can issue the request to the neighbor that earlier sent the third-party advertisement for the requested node. Here, the rationale is that address information about a node is more likely to be found at the node that has sent an advertisement for this node. Alternatively, the request can be made to the next-hop neighbor on the path to the requested node. Here, the rationale is that address information about a node is more likely to be found closer to the location of the requested node.

The steps performed by *Pull* share aspects with address resolution protocols in non-broadcast networks. For example, the Next-Hop-Resolution-Protocol (NHRP) [6], which has been used for IP-to-ATM address resolution when multiple IP subnets are realized on a common ATM substrate, follows the IP routing table to the subnet where the requested node is located.

C. Variants and Hybrid Methods

With *Push*, appending address lists to all third-party node advertisements incurs substantial overhead. To evaluate whether *Push* can be effective with less overhead, we consider a modified version of *Push*, where only the substrate address with the highest preference is piggybacked to a node advertisement, using the preference settings of the 'owner' of an address. This method is referred to as *Push-Single*.

We refer to hybrid methods as CSA mechanisms that combine elements of the approaches above. In this paper, we evaluate a hybrid method that supplements gossip communication with features of *Push*. The main drawback of (pure) gossip communications is that the dissemination of address information is not deterministic, and the main drawback of *Push* is the incurred overhead. In the considered hybrid method, referred to as *Push-Single+Gossip*, we enhance *Push-single* with the gossip protocol. This methods transmits the most preferred substrate address pro-actively and disseminates complete address lists via gossiping. In the experimental evaluation in Section V, the hybrid method is shown to significantly enhance the effectiveness of a pure gossip solution.

V. Evaluation

In this section, we present an experimental evaluation of the cross-substrate advertisement mechanisms from Section IV to assess and compare the effectiveness of CSA in supporting self-organizing overlay networks in a multi-substrate environment. We organize substrate networks in a layout that allows us to predict their connectivity. Also, we use an overlay network with a structured topology, where convergence of the topology can be asserted.

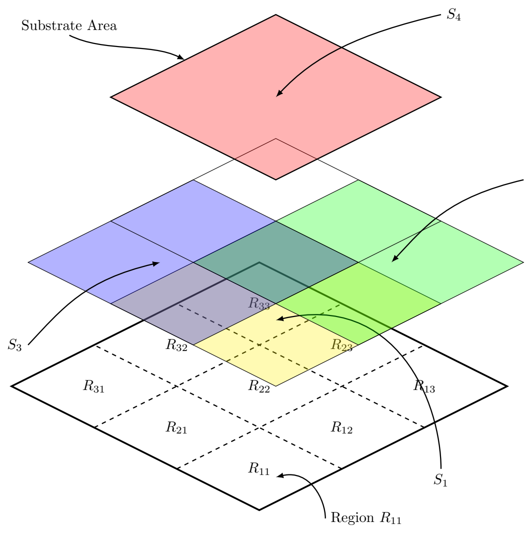
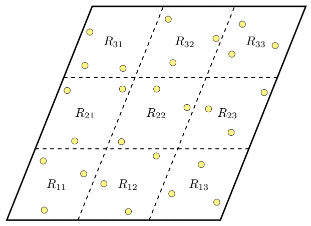


Fig. 4. Substrate arrangement for DT.

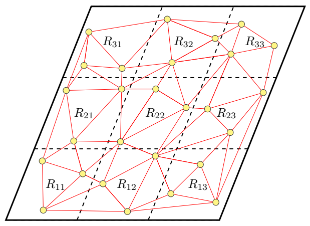
Testbed Network. The experiments are conducted on an Emulab [3] cluster at the University of Toronto, consisting of 22 servers with Intel Xeon 2 GHz processors, and GigE interfaces. In the experiments, workload is distributed approximately equally across the servers.

Emulating Multiple Substrates. The substrate networks in our experiments are set up as UDP/IP networks, which are labeled with an additional substrate identifier. (The identifier becomes part of the substrate address.) This enables us to conveniently create a large number of substrates. Two nodes share a common substrate network and can exchange messages directly only if they have a substrate address with the same substrate identifier. The UDP/IP substrate networks are set up as non-broadcast networks.

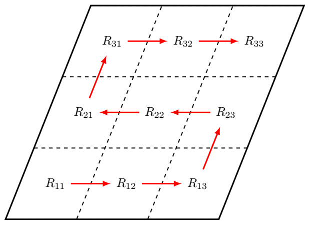
Arrangement of Substrate Networks. Each substrate is associated to a square in a 2-dimensional plane, with an area of $(\ell \times \ell)$ for each substrate networks. The areas of substrates are laid out to form an overlapping tiling, with the length of overlap given by $\ell/2$. In this fashion, a system of $N \times N$ substrate networks creates $(N+1) \times (N+1)$ tiles, which we refer to as *regions*. The scheme for arranging the overlapping substrates is illustrated in Fig. 4. The top of the figure shows the area associated with substrate S_4. Below, we show how the other three substrates, S_1, S_2, and S_3, overlap to form the overlapping tiling. The overlapping areas of substrate networks $S_1, \ldots, S_4$ create nine regions, where each region is associated with one or more substrate networks. In the figure, the regions are labeled as $R_{11}, \ldots, R_{33}$. By distributing overlay nodes to the regions, we associate the nodes with substrate networks. For example, a node located in the labeled region R_{11} is only attached to substrate network S_1, while a node in region R_{22}



(a) Placement of nodes.



(b) Delaunay triangulation.



(c) Buddy configuration.

Fig. 5. Delaunay triangulation overlay topology in a (2x2) grid of substrate networks with 9 regions (In (c), the arrows indicate the neighboring region where a pre-configured buddy is located).

is connected to all four substrates $S_1 - S_4$.

Overlay Topology. For our experiments, we use a structured overlay protocol that establishes a Delaunay triangulation topology. The DT protocol from [11] is an application-layer overlay topology protocol that creates a Delaunay triangulation for overlay nodes with (x,y)-coordinates in a non-broadcast substrate network. In our experiments, the coordinates of nodes are matched to the regions of the substrates, and nodes are evenly distributed across the regions formed by the substrate networks (see Fig. 5(a)).

Since we establish non-broadcast substrates, each node must be pre-configured with a set of substrate addresses of overlay nodes, the *buddies*, to make an initial rendezvous with an existing overlay network.[1] After the rendezvous, a node maintains a list of neighbors in the overlay topology. Periodically, every *HeartBeat* (we use *HeartBeat* $= 500$ msec), a node sends to each neighbor a *HelloNeighbor* message, which includes node advertisements of the closest neighbors in a clockwise and counter-clockwise direction. When each node updates its neighbor table with incoming *HelloNeighbor* messages, the network eventually converges to the desired topology (see Fig. 5(b)).

In the multi-substrate experiments with the DT protocol, each node is configured with two *buddies*. One buddy is a node in the same region, the other buddy is a node in a neighboring region, following a scheme indicated in Fig. 5(c). The arrows in the figure indicate where a buddy in a neighboring region is located, i.e., the second buddy of nodes in region R_{11} is located in R_{12}, and so forth. As seen in the figure, the buddies are selected such that they form a chain between regions. This ensures that it is in principle feasible to form a single connected overlay network.

[1]The version of the DT protocol described in [11] uses a central server for the rendezvous process. The version used in the experiments is as described here.

A. Measurement Methodology

In our experiments, we start a given number of overlay nodes and measure the effectiveness of CSA methods in terms of the ability to establish a multi-substrate overlay network. As performance metrics we use *stability* and *connectivity* of the overlay topology, which are measured as follows:

Stability: In the DT protocol, a node is *locally stable* if it satisfies a local stability condition (see [11]), which can be computed from its neighborhood table. If all nodes are locally stable, it is assured that the overlay network has formed a stable Delaunay triangulation topology. We use the percentage of locally stable nodes to measure the progress towards completing a stable topology. However, this measure does not detect if multiple disconnected topologies have formed.

Connectivity: In the DT protocol, a node is a *leader* if it does not have a neighbor with a larger coordinate[2]. Since each Delaunay triangulation graph can have only one leader, the number of leaders indicates the number of partitioned (unconnected) overlay topologies. When an experiment is started, each node is initially a leader of an overlay network with itself as the only member. We say that a set of nodes is *connected* when all overlay nodes have the same leader. We use the number of leaders as a measure of progress towards establishing a single connected overlay network.

A single stable overlay network with a Delaunay triangulation topology has formed if and only if all nodes are locally stable and there is only one *leader*.

Our experiments compare the CSA methods for relayed address list exchange from Section IV. *Gossip* refers to the gossip protocol, where we evaluate gossip intervals of $250, 500$ and 1000 msec. *Push*, *Pull*, and *Push-single* denote the protocol-driven dissemination methods, and *Push-single+Gossip* refers to the previously discussed hybrid method.

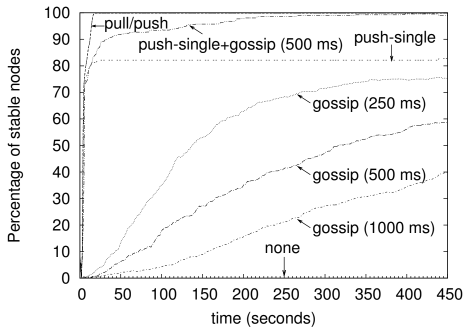
The length of all experiments is 450 sec. Each experiment is repeated three times, and we plot the average of the results in graphs.

B. Experiment 1: 64 Substrate Networks

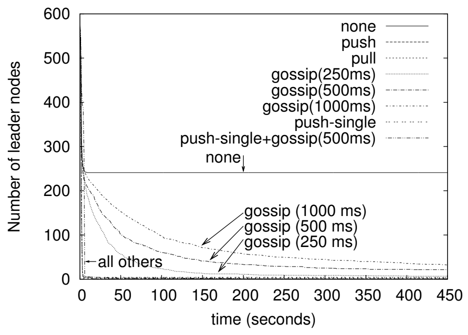
We first evaluate the CSA methods in a network of 64 substrate networks, laid out as an 8×8 overlapping grid, creating 81 regions (as shown in Fig. 4). We place eight nodes into each region, resulting in an overlay network of 648 nodes.

Figs. 6(a) and 6(b), respectively, depict the percentage of stable nodes in the overlay and the number of partitioned topologies (measured by counting 'leaders') as functions of time. Fig. 6(c) depicts the connectivity metric for a smaller range of values, which allows us to validate whether a method can establish a single topology (with one 'leader'). The graphs show that *Push* and *Pull* quickly establish a single stable overlay network in less than 20 sec. The *None* option does not lead to a stable network, thus providing evidence that CSA is needed to establish a multi-substrate overlay network. Gossip communication shows a slow increase of the stability and connectivity measures, with better results for shorter gossip

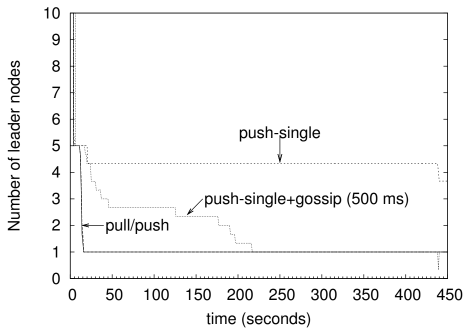
[2]$coord(A) < coord(B)$, if $y_A < y_B$, or $y_A = y_B$ and $x_A < x_B$.



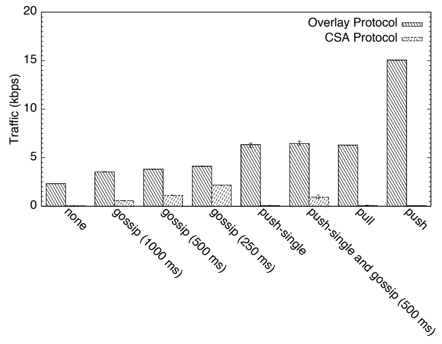
(a) Stability.



(b) Connectivity.



(c) Connectivity for selected methods.

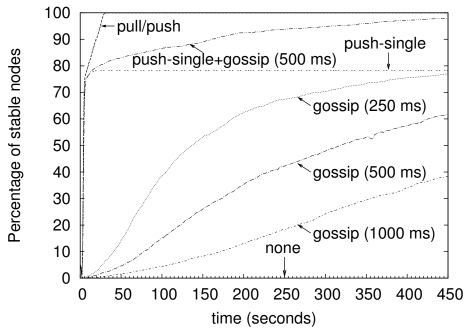


(d) Protocol overhead.

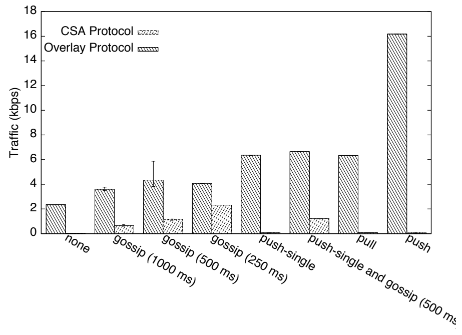
Fig. 6. Experiment 1: Overlay topology with 64 substrate networks and 648 nodes.

interval. The *Push-single* increases sharply initially, but cannot achieve stability for all nodes, and also cannot form a single connected network. At the same time, the hybrid method that enhances *Push-single* by gossiping can create a single overlay containing all nodes.

Fig. 6(d) compares the overhead of protocol messages incurred by the CSA mechanisms in the experiment. We measure the average amount of control traffic received by a node, averaged over the length of the experiment. Since CSA messages are frequently piggybacked to control messages of the overlay protocol, we present the total control traffic, which includes messages of the DT protocol with or without piggybacked CSA information (labeled as *Overlay Protocol*), as well as standalone messages sent by the CSA processor (labeled as *CSA Protocol*). The amount of traffic with *None* gives a lower bound for the control overhead without CSA. As expected, *Push* incurs the most overhead, while the overhead for all other methods is modest. In summary, the protocol-driven dissemination methods appear most effective. With *Pull* generating considerably less protocol traffic than *Push*, it offers the best tradeoff overall.



(a) Stability.

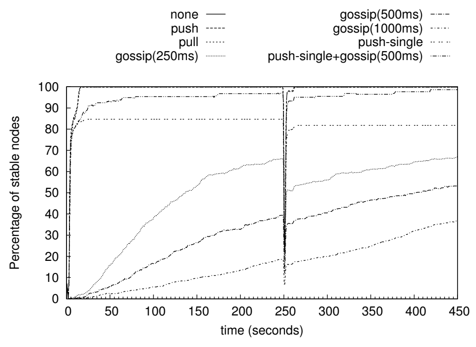


(b) Protocol overhead.

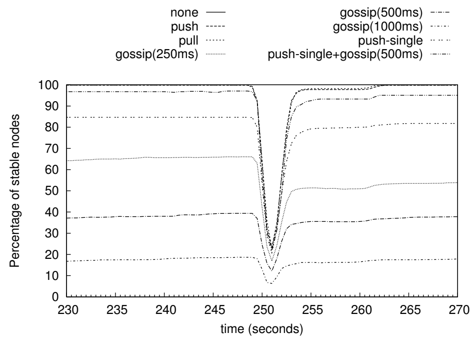
Fig. 7. Experiment 2: Overlay topology with 289 substrate networks and 2592 nodes.

C. Experiment 2: 289 Substrate Networks

We repeat the same experiment, in a larger network, where we quadruple the number of regions and nodes. The substrates are arranged in an overlapping 17×17 grid, resulting in 324 regions, that each receive eight overlay nodes for a total of 2592 overlay nodes. Figs. 7(a) present the results for stability (we omit the graph on connectivity for lack of space). For all methods, the outcomes are similar to Experiment 1. Fig. 7(b) shows that the average protocol overhead in the larger network is similar to that observed in Experiment 1. We conclude that, for the chosen substrate and overlay network configuration, all CSA methods scale well.



(a) Result of the entire experiment.



(b) Snapshot for $230\,\mathrm{sec} < t < 270\,\mathrm{sec}$.

Fig. 8. Experiment 3: Stability in a scenario with churn (64 substrate networks, 648 nodes; at t=250 sec, 25% of nodes in each region leave the network).

D. Experiment 3: Performance of CSA under Churn

In this experiment, we examine how CSA methods perform when the network experiences a major disruption. The setup for this experiment is as in Experiment 1. Half-way through the experiment, at time $t = 250$ sec, 25% of randomly selected nodes in each region leave the network. We are interested in studying the impact of this event on the ability to maintain the topology of the overlay network.

Fig. 8(a) presents the stability measure with different CSA methods over the duration of the entire experiment, and Fig. 8(b) depicts the same data for a period starting 20 sec before the departure event until 20 sec after the event. The figures show that the departure event majorly disrupts the overlay topology, as the percentage of stable nodes falls to 30% or below. With *Push* and *Pull*, the network quickly recovers to full stability. The *Push-single* method, with or without gossiping, returns quickly to the stability level before the departure event. With gossip, we observe that after an initial quick recovery, the stability settles at a level that is below that before the departure event. This difference between the gossip and protocol-driven methods is noteworthy. After the departure event, the overlay topology protocol repairs the topology, resulting in increased protocol traffic in areas where a repair is needed. Since protocol-driven methods piggyback CSA information to protocol messages, they deliver address information where it is needed. In contrast, gossip dissemination does not specifically direct address lists to nodes affected by the departure event. Thus, the address information lost in the departure event must be slowly re-acquired.

VI. Conclusions

We have studied mechanisms for resolving address bindings in a self-organizing overlay network over multiple substrate networks. We presented methods for cross-substrate advertisement, by which overlay nodes exchange address information of one substrate network across another substrate network. We have evaluated the effectiveness and the overhead of CSA methods in a set of measurement experiments on a testbed network. Our experiments show that CSA improves the ability to establish a single connected overlay network, even when the number of substrate networks grows large. Overall, the on-demand *Pull* method presented the best trade-off in terms of effectiveness and overhead, compared to proactive methods such as *Push* or a gossip protocol. While the performance of CSA methods depends on the particular properties of substrate networks and the overlay topology, our experiments show that CSA is a crucial protocol component in large-scale multi-substrate overlay networks.

Acknowledgment

This work is supported in part by the Natural Sciences and Engineering Research Council of Canada.

References

[1] Y.-H. Chu, S. G. Rao, S. Seshan, and H. Zhang. A case for end system multicast. In *Proc. ACM Sigmetrics*, pages 1–12, June 2000.
[2] J. Crowcroft, S. Hand, R. Mortier, T. Roscoe, and A. Warfield. Plutarch: an argument for network pluralism. In *Proc. ACM SIGCOMM FDNA Workshop*, pages 258–266, August 2003.
[3] B. White et. al. An integrated experimental environment for distributed systems and networks. In *Proc. OSDI*, pages 255–270, Dec. 2002.
[4] D. G. Andersen et. al. Resilient overlay networks. In *Proc. ACM SOSP*, pages 131–145, October 2001.
[5] I. Stoica et. al. Chord: A scalable peer-to-peer lookup service for Internet applications. In *Proc. ACM SIGCOMM*, pages 149 – 160, August 2001.
[6] J. Luciani et. al. NBMA next hop resolution protocol (NHRP). IETF RFC 2332, 1998.
[7] R. Bless et. al. The underlay abstraction in the spontaneous virtual networks (SpoVNet) architecture. In *Proc. 4th Euro-NGI Conf. on Next Generation Internet Networks*, pages 115–122, April 2008.
[8] B. Ford. Scalable Internet routing on topology-independent node identities. Technical report, MIT, October 2003.
[9] P. Francis and R. Gummadi. IPNL: A NAT-extended Internet architecture. In *Proc. ACM SIGCOMM*, pages 69–80, August 2001.
[10] S. M. Hedetniemi and A. L. Liestman. A survey of gossiping and broadcasting in communication networks. *Networks*, 18(4):319–349, 1988.
[11] J. Liebeherr, M. Nahas, and W. Si. Application-layer multicasting with Delaunay triangulation overlays. *IEEE J. on Sel. Areas in Comm.*, 20(8):1472–1488, October 2002.
[12] J. Liebeherr, J. Wang, and G. Zhang. Programming overlay networks with overlay sockets. In *Proc. 5th COST 264 NGC Workshop, LNCS 2816*, pages 242–253, September 2003.
[13] D. Meyer. The locator identifier separation protocol (LISP). *Internet Protocol Journal*, 11(1):23–34, March 2008.
[14] R. Moskowitz and P. Nikander. Host Identity Protocol (HIP) architecture. RFC 4423, IETF, May 2006.
[15] D. C. Plummer. An Ethernet address resolution protocol. RFC 826, IETF, November 1982.
[16] S. Ratnasamy, P. Francis, M. Handley, R. Karp, and S. Shenker. A scalable content-addressable network. In *Proc. ACM SIGCOMM*, pages 161–172, August 2001.
[17] M. Valipour. Cross-substrate advertisement: Building overlay networks for heterogeneous environments. Master's thesis, Univ. of Toronto, 2010.