Synergy: An Overlay Internetworking Architecture and its Implementation

Minseok Kwon
Department of Computer Science
Rochester Institute of Technology
Rochester, NY 14623
E-mail: jmk@cs.rit.edu

Sonia Fahmy
Department of Computer Science
Purdue University
West Lafayette, IN 47907–2066
E-mail: fahmy@cs.purdue.edu

***Abstract*— A multitude of overlay network designs for resilient routing, multicasting, quality of service, content distribution, storage, and object location have been recently proposed. Overlay networks offer several attractive features, including ease of deployment, flexibility, adaptivity, and an infrastructure for collaboration among hosts. In this paper, we explore cooperation among co-existing, possibly heterogeneous, overlay networks. We discuss a spectrum of cooperative forwarding and information sharing services, and investigate associated scalability, heterogeneity, and security problems. Motivated by these services, we design *Synergy*, a utility-based overlay internetworking architecture that fosters overlay cooperation. Our architecture promotes fair peering relationships to achieve synergism. Results from Internet experiments with cooperative forwarding overlays indicate that our Synergy prototype improves delay, throughput, and loss performance, while maintaining the autonomy and heterogeneity of individual overlay networks.**

I. Introduction

Over the past few years, overlay networks have emerged as a flexible paradigm for sharing information and rapidly prototyping disruptive technology. Without violating the end-to-end principle [1], overlays can be used to deploy new functionality between the core IP network and applications. Overlay solutions have been designed for the multicasting [2], [3], inter-domain routing [4], [5], content distribution [6], storage [7], quality of service [8], and peer-to-peer networking [9], [10] problems.

As overlay networks become pervasive, a number of questions naturally arise. First, can autonomous and *heterogeneous* overlays cooperate? Current overlay networks have both heterogeneous performance goals and heterogeneous service goals, e.g., minimum latency versus maximum bandwidth overlays, or multicast distribution versus unicast forwarding versus distributed storage. The key incentive for overlay cooperation is to exploit the presence of hosts with *unique resources* in one overlay, when such resources are absent in other overlays that need them. For example, a strategically located host can significantly improve delay performance of a flow which would otherwise traverse a sub-optimal policy-based route [4], [5]. A host which is already receiving an application-level multicast stream can distribute that stream to its neighboring hosts [11]. A host with high computational, memory or storage capabilities can be exploited by other overlays. A host with content that other hosts are seeking can supply that content to hosts in other overlays. These are only a few of the potential avenues in which *synergy* among overlays is possible.

Second, can co-existing overlays interfere? Overlay networks are currently being deployed independently. Consequently, several diverse overlay networks may simultaneously exist and compete for the same resources in the Internet [12], [13]. For example, multiple overlays may compete for resources on co-located overlay nodes (e.g., compete for CPU cycles or memory on the same host), compete for bandwidth on shared underlying links, or compete for buffer space in common underlying routers. In this context, *co-located nodes* denote a single host connected to more than one overlay network, but the definition can be extended to hosts in the same subnet, or hosts in the same autonomous system (AS).

Third, can overlay cooperation facilitate the design and deployment of revolutionary Internet architectures? Several recent studies propose bold re-designs of the current Internet architecture [14], [15]. Overlays can provide a framework to accelerate prototyping such approaches. For example, cooperative overlays can be used to implement heterogeneous contexts and interstitial functions as in Plutarch [14]. In such a framework, one overlay (or context) employs homogeneous addressing, naming, routing, or transport. Different overlays (or distinct contexts), however, are connected via interstitial functions to map functionality. Another example is [15], where regions and their relationships can exploit cooperative overlays. Cooperative overlays also provide the flexibility necessary for rapid prototyping and deployment of new routing proposals such as NIRA [16].

In this paper, we explore a spectrum of cooperative services among co-existing autonomous overlays (Fig. 1), and discuss associated scalability, heterogeneity, and security problems. As a proof-of-concept, we design and implement a utility-based overlay internetworking architecture, *Synergy*, where overlay agents facilitate inter-overlay coordination. Overlay nodes may elect to join the Synergy network which incorporates nodes from several overlay networks. Long-lived flows can utilize Synergy forwarding for better performance. Synergy tackles a number of management challenges that arise when constructing a shared overlay network comprised of autonomous overlays. The routing protocol in Synergy considers load at

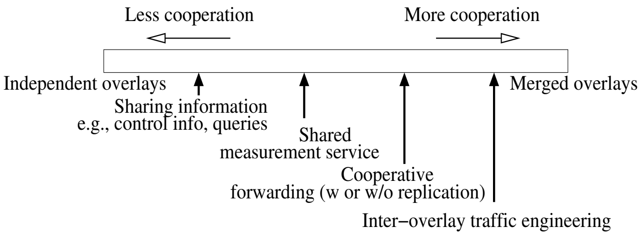


Fig. 1. The spectrum of overlay cooperation

hosts, time intervals between successive routing decisions, and peering relationships among overlays, to offer stable, scalable, and fair cooperative routing. We have implemented a prototype of Synergy cooperative forwarding, and conducted experiments on the PlanetLab wide-area experimentation platform [17]. Results indicate that Synergy improves latency, throughput, and loss, while preserving overlay heterogeneity.

The remainder of this paper is organized as follows. Section II gives examples of cooperative overlays. In Section III, we discuss routing and management in the Synergy overlay internetworking architecture. In Section IV, we describe our implementation details. In Section V, we present preliminary results from our experiments and simulations. In Section VI, we give an overview of related work. Finally, we summarize our conclusions and future work in Section VII.

II. A Spectrum of Cooperative Services

There are several possible degrees of overlay cooperation. In the extreme case, independently administered overlay networks may be merged into a single overlay network. The decision to merge overlays can be based upon whether the overlays have common goals (e.g., maximum bandwidth multicast, minimum delay routing), and the number and location of nodes in each of the overlays. In a merged overlay network, competition among overlays, and inter-overlay communication and coordination are eliminated. Hosts originally in different overlay networks, however, lose their heterogeneous performance and service goals after merging. In addition, scalability problems of certain types of overlays may be exacerbated.

Partial overlay cooperation can reduce competition and improve overall performance, while preserving heterogeneity and autonomy, at the expense of more complex management. We now give several example services that are possible with overlay synergy. In the remainder of the paper, we explore the *cooperative forwarding* service in more depth.

Proactive versus reactive cooperative forwarding. The Synergy architecture we will describe below is geared towards cooperative forwarding for long-lived flows. A set of hosts from several overlay networks comprise the Synergy network, and serve as transit nodes (with or without replication) for other overlays. Synergy allows a sender to route a long-lived flow around misconfigurations, faults or hot spots in its own overlay. This strategy can be employed *reactively*, only when faults or hot spots are detected, or *proactively*.

Control information sharing. Without coordination, multiple overlays may use network measurements to route selfishly, causing route instability. For example, route flapping may occur when several overlays select the same shortest path at one time and then determine another common shortest path the next time. In order to mitigate route instability, we can randomize routing time intervals for different overlays, and disseminate intervals via overlay agents (which facilitate coordination among overlays). We can also utilize traffic equilibrium computations as in [13]. In either case, the time scale for network measurements must be shorter than the routing time interval.

Shared measurement. As argued in [18], separate probing by overlay networks may introduce significant overhead, especially since overlay nodes can reside in the same host or subnet. Therefore, a coordinated and shared measurement service can be extremely useful. Clearly, there is a tradeoff between the granularity and freshness of information. The measurement service and the overlay routing algorithms that exploit it can also employ hierarchical aggregation [19], hashing/binning, randomization, or inference [20] mechanisms to distribute and reduce measurement overhead.

Query forwarding. Hosts in different peer-to-peer networks can cooperate to expand the potential search space. If a host in a peer-to-peer overlay network fails to find requested objects in its own overlay network, the host forwards the query to its overlay agent. The agent translates and disseminates this query with the address and type of the originator to agents in other peer-to-peer networks.

Inter-Overlay traffic engineering. Routing paths independently computed by each overlay network may degrade overall network performance and fairness. We can exploit network tomography and inference techniques to identify common bottlenecks and infer internal delays and losses. Using this inferred information, multiple overlay networks can be coordinated to maximize the bottleneck bandwidth, for example.

III. Synergy Architecture

The objective of our prototype Synergy architecture is twofold: (i) to improve routing performance via cooperative forwarding, and (ii) to be easily extensible to new services, such as those discussed in the previous section. Given an overlay network o, we define an *overlay link* or *connection* as $e_o = (d_s, \eta_0, \ldots, \eta_l, d_r)$. This comprises a host d_s, followed by a sequence of routers η_i, followed by a host d_r. An *overlay flow* is a chain of overlay links in which the first and last hosts are the sender and destination, respectively. A host in an overlay network is connected to its neighbors in the overlay via overlay links. A *home* overlay is defined as the autonomous overlay network which a host originally joins. Each overlay network executes its own routing protocol, according to its own optimization metrics, e.g., delay, bandwidth, node identifier, etc. We now describe the components of the Synergy architecture.

A. Overlay Agents

Each overlay network periodically designates one of its hosts as an *overlay agent* (for brevity, we will refer to it as an “agent”). The agent facilitates joining Synergy via identifier assignment and utility estimation (details are given in Section III-C). Highly loaded or bandwidth-constrained nodes are eliminated from consideration for becoming agents. The selected agent can be the host with the estimated (i) highest number of co-located overlay nodes, or (ii) highest number of overlays represented in its current or past neighbors in the Synergy network, or (iii) minimum maximum delay to other hosts in its home overlay network. Hosts with co-located overlay nodes (heuristic (i)) are good candidates since they can serve as agents for multiple overlays, and reduce traffic among agents. Traffic can also be reduced if the agents of different overlays are in close proximity. In heuristic (ii), a host which is close to hosts from several other overlay networks is considered a good candidate. In heuristic (iii), all hosts in the home overlay can rapidly contact their agent since the agent is closest to the overlay centroid. Primitives such as those in [18] can supply network information required for agent selection. Observe that every agent must have a backup agent, which replaces the overlay agent when it fails or leaves its home overlay.

All agents form a delay-based overlay mesh, called the *agent network*. A bootstrap mechanism allows the discovery of other overlays and their agents. We assume that the address of at least one bootstrap node is globally known. Either one of the agents (which we refer to as the *primary agent*) or an independent server can serve as the Synergy bootstrap node. Each agent is restricted to connect to less than UBA (the upper bound in the agent network) neighbors on the agent mesh, where UBA is determined based upon a tradeoff between performance and scalability.

B. Utilizing the Synergy Network

The Synergy overlay internetwork (or “Synergy network” for short) constitutes hosts exported (i.e., temporarily contributed) by autonomous, but cooperating overlays. The Synergy network is a new overlay network that incorporates *selected* hosts from more than one overlay network. This form of cooperation is analogous to the notion of code sharing in airline systems. For example, a customer who has purchased a ticket from airline A actually uses airline B if the two air carriers have an agreement to share certain routes. In this case, airline A code shares certain routes with airline B, while still operating its own routes. Airlines employ code sharing for increased efficiency and better resource utilization. From the customer perspective, the notion of code sharing can reduce costs and improve customer experience.

Overlay flows are classified based on the sender and receiver identities in the packet header. Long-lived flows are better-suited for exploiting the Synergy *cooperative forwarding service*, since short-lived flows are unlikely to have time to find Synergy routes that are better than their home overlay routes. Each sender keeps track of the duration of its flows. A flow is declared *long-lived* once its time duration exceeds a predefined threshold. In this case, all the nodes on the overlay flow path (including sender and receiver) are exported, so that the sender can utilize the Synergy network to send to the receiver. Once these hosts are exported, their overlay flows use Synergy routing paths (instead of home overlay paths) to deliver packets. Synergy routing paths are computed *completely independently* of routing protocols of home overlays. Therefore, Synergy must support all routing metrics of participating overlays, as discussed in Section III-D. Observe that Synergy forwarding need not only be unicast – with replication, multicast and peer-to-peer overlays can be supported.

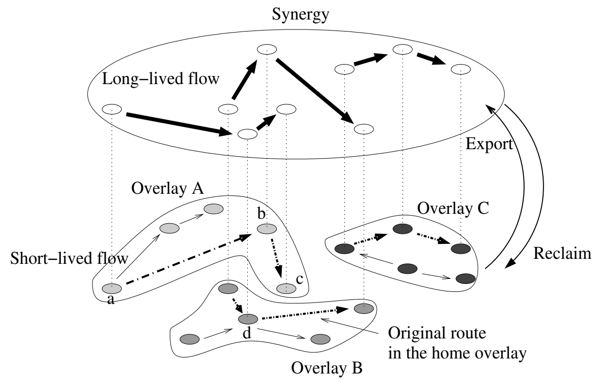


Fig. 2. Overlay internetworking architecture: An overlay internetwork is constructed among exported hosts. Long-lived flows (dash-dot lines) in home overlays are replaced by Synergy routes (thick lines) if this will improve performance.

While a host is part of Synergy, the host executes both home overlay routing (for nodes on its home overlay) and Synergy routing. A host in Synergy thus maintains two forwarding tables: one for its home overlay, and another for Synergy. An overlay flow in Synergy routing can exploit hosts from both its home overlay and other overlays as intermediate hops on routing paths. This helps circumvent network hot spots by making a larger pool of hosts available as potential transit nodes. Fig. 2 depicts an example of Synergy forwarding. In the figure, the communication from sender host a to receiver host c in overlay network A benefits from cooperation between overlays A and B. The route from a to c employs host d in overlay B as a transit node. This cooperation can yield shorter (or wider) routes than the routes that only transit hosts in overlay network A.

C. Host Export

An exported host is assigned a unique identifier (e.g, a serial number) by the primary agent. Thus, hosts with the same identifier in different overlays (e.g., same nodeId in two Pastry overlays) can be easily distinguished in Synergy. An identifier from a leaving host can later be reused by new Synergy hosts. A mapping table (SynergyId, IP_address) is maintained by agents and distributed to exported hosts as well as hosts that utilize Synergy. Packets on Synergy can be thus routed using Synergy identifiers which map to IP addresses.

Operations for exporting and reclaiming hosts are facilitated by overlay agents. Hosts, however, communicate within Synergy *without* the intervention of agents once they have successfully been exported. A host can leave Synergy by notifying its neighbors. The neighbors then disconnect the links to that host. When new hosts are exported to the Synergy network, an approximately equivalent number of hosts previously exported from the same home overlay (if any) may leave Synergy, i.e., be *reclaimed*, to maintain the Synergy network size manageable. Such hosts are clearly not allowed to leave Synergy if being used as intermediate nodes by any flow in the Synergy network.

The choice of which hosts to export (*in addition to* hosts in long-lived flows) significantly impacts the performance of Synergy. At least four heuristics may be used to make good export choices. First, an exported host should be likely to be useful to other overlays, either because it possesses unique resources/capabilities, or because of its strategic location. For example, an exported host in close proximity to several exported hosts may not significantly enhance delay performance, since it likely offers similar delays to these hosts in its close proximity. Second, if the exported host is already heavily loaded, it may not be as useful to other overlays as a lightly loaded host. Third, loads on different overlays may be considered, not just loads on hosts. For example, an overlay which is streaming audio can typically export a larger number of hosts than an overlay streaming high bandwidth video. Fourth, a trust-based priority mechanism must be used to determine which overlays are "cooperative," as we will discuss in Section III-D. The *export utility* of a host can therefore be computed based upon any subset of these factors.

In our implementation, the export utility approximates the number of hosts that are likely to transit this host when forwarding their packets on Synergy, multiplied by their performance gain when utilizing this host. To estimate this, we use the number of times this host serves as a next hop in its home overlay, which gives an indication of how strategically located this host is, divided by the number of possible connections. Unfortunately, comparing this number for all hosts in an overlay is $O(n^2)$ (where n is the number of overlay hosts), which is prohibitive for large overlays. Therefore, in our implementation, only p random hosts are used for computing the number of entries (where a host is the next hop) for q selected hosts. The number of entries examined in each routing table is limited to r. The agent estimates the export utility of each of the q hosts as the number of times they appear as next hop in the selected r entries of the routing tables of these selected p hosts. As long as $p, q, r \ll n$, the export utility computation is scalable. In the case of application level multicast, utility can be estimated as the size of the subtree rooted at that host. The utility value is periodically estimated, and hosts that exceed a specified threshold are exported, while those that fall below another threshold are reclaimed.

D. Routing and Priority Mechanisms

Our routing protocol employs three mechanisms to ensure that the Synergy network is fairly utilized. First, the inbound load at a host is limited to k overlay flows. This ensures that the host is not overloaded by many transit flows. Second, only one overlay flow is allowed to select its routes at a time. We serialize overlay flows for *route updates* in the order of flow identifiers. The primary overlay agent coordinates this process. Note that packets can still be forwarded concurrently while routing decisions are serialized. Third, an overlay flow is allowed to utilize host h as a transit node only if that host has a lower or equal priority than the source and destination of the flow, i.e., if $maximum(priority(d_i)) \geq priority(h)$, where d_i denotes source and destination hosts of a flow. We define $priority(d_i)$ as the number of overlay flows that d_i has assisted in the recent past by acting as a Synergy transit node. Our implementation computes $priority(d_i)$ using an exponentially weighted moving average (EWMA) of the number of overlay flows for a certain time duration. At time t, $priority(d_i)$ becomes $(1 - \alpha) \times priority(d_i) + \alpha \times NumFlows(t)$. Observe that $maximum(priority(d_i))$ will increase if the source and destination have recently served as transit nodes through which other overlay flows have been routed. This approach ensures basic fairness in relationships among cooperating overlays. If an overlay network has refused to export potentially high-priority hosts in the recent past, its flows are allowed to use fewer hosts from other overlays due to the lower priority of these flows.

The Synergy routing protocol can use different metrics on each link. To compute routing paths, Synergy runs a link-state routing protocol on top of a network with overlay links for all possible node pairs. Each node maintains overlay link properties to all the other nodes for different metrics. Bandwidth is the most appropriate choice for overlays for high-throughput bulk data delivery (e.g., file download service in peer-to-peer systems), while real-time streaming overlays are latency-sensitive. An overlay flow on Synergy chooses the most representative primary routing metric(s), depending on its home overlay routing goals. Since each Synergy flow determines its routes independently of routes in home overlays, overlays with heterogeneous performance goals can cooperate. The overlay link available bandwidth is estimated through the TCP throughput formula as a function of round-trip latency and packet loss.

E. Security

Non-cooperative or non-trustworthy hosts/overlays may exploit other overlays, but not contribute to them. Our export/reclaim mechanism inherently alleviates this problem, since hosts originally on a flow that now uses Synergy routing are available to be utilized by other overlays. A host is inaccessible to other overlays when its home overlay no longer uses Synergy. This formulates simple peering relationships among overlays. The Synergy architecture, however, is vulnerable to several malicious attacks, which are also possible (but easier to detect) in independent overlay networks. For example, a host may misbehave by rejecting incoming packets, lying about measurements to cause incorrect routing, rapid joining and leaving, and launching denial-of-service attacks. To mitigate such attacks, secure algorithms for node identifier assignment, routing, and forwarding can be adopted. For example, cooperation between the sender and receiver of an overlay flow can

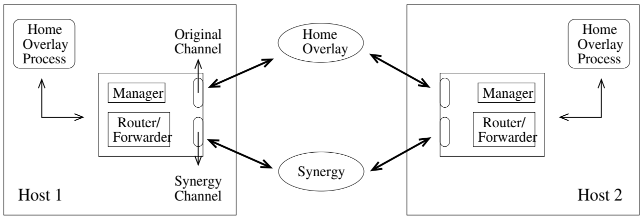


Fig. 3. Synergy implementation

help detect forwarding disruptions. The sender can periodically inform the receiver of current sending rates through direct unicast from the sender to the receiver (not via Synergy intermediate hosts). Routing attacks, where intermediate hosts modify the content of routing update information exchanged among participants, can also be similarly detected. If multiple nodes collude, however, these mechanisms cannot prevent attacks. Certified node identifiers can authenticate overlay flows in this case. Implementation and experimentation with such mechanisms are included in our future plans.

IV. Implementation

We have implemented the techniques described in Sections III-A to III-D as a proof-of-concept for Synergy, with the exception of heuristic-based agent selection (Section III-A), which is included in our future work plans. Currently, we simply assign a random overlay host as the overlay agent. The primary goal of our implementation is to *transparently* provide a host with cooperative overlay services. Synergy is a separate component running on the same hosts where home overlays are deployed. Synergy is implemented at the user-level, so as *not* to require superuser privileges or kernel modifications.

As shown in Fig. 3, Synergy comprises three modules: the channel, manager, and router/forwarder. Packets from the home overlay process are diverted to the Synergy component using divert mechanisms supported by several operating systems (e.g., ipchain on Linux). For our experiments, however, we emulate the divert mechanism via inter-process communication to avoid requiring superuser privileges. Therefore, our Synergy component sends periodic signals to a pre-specified port to which the home overlay process can listen.

The manager controls host export/reclaim, and incoming packet classification. If a host is exported, packets are sent to the Synergy channel; otherwise, the manager directs packets to the home overlay channel. Packets from the home overlay channel are transmitted following the paths established by the home overlay. The router/forwarder computes the Synergy routing paths. We reuse part of the RON implementation [5], specifically (i) classified routing and forwarding for different routing metrics (but no policy routing), (ii) ping protocols, and (iii) the performance database. The Synergy channel forwards its packets to the Synergy component at the next hop computed by this router/forwarder. When a packet arrives at the Synergy channel of the final destination, the channel passes the packet to the home overlay process at that destination.

V. Performance Evaluation

In this section, we analyze the performance of our overlay internetworking implementation via both Internet experiments and simulations.

A. Experimental Setup

We have implemented Synergy in C++ and conducted Internet experiments on PlanetLab [17]. We experiment with 8 RON overlays where each overlay contains 8 hosts. Therefore, we have a total of 64 nodes: 43 nodes belong to universities or companies in the United States, and the remaining 21 are located in Asia, Europe, Canada, and Australia. All the overlays except one have at least one host which is located outside the United States. Our experiments were conducted over several days in April−June 2004. We present in this paper a representative subset of the results.

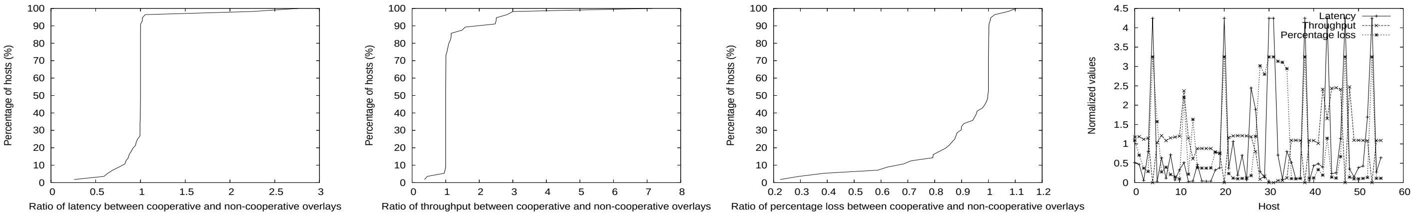
In each overlay, a randomly chosen data source generates data streams to three other overlay nodes. Two streams are generated per source-destination pair, i.e., the total number of overlay flows is $2 \times 3 \times 8 = 48$. For the two streams per pair, one stream is transmitted over the home overlay only, while the other stream is concurrently transmitted over the Synergy network. The flows are bulk data transfers with application rate 1.7 Mbps. We use $r = 5$ for computing the export utility (p and q are set to all overlay hosts). The export algorithm uses utility threshold values 0.2 and 0.02. The time duration to detect long-lived flows is set to 5 seconds. The priority is updated every 14 seconds with $\alpha = 0.1$, and we use $k = 3$ for the inbound load. Latency and loss are measured by generating probe packets, and throughput is estimated from the TCP equation.[1]

B. Experimental Results

Fig. 4(a) illustrates the cumulative distribution of the ratio of latency observed with Synergy and that observed with independent home overlays, averaged for all hosts. The x-axis indicates the value of the ratio and the y-axis denotes the percentage of hosts that have latency ratios less than this value. The results are collected from a total of 120,000 latency samples from the sources to receivers. We find that Synergy reduces the latency of approximately 50% hosts compared to home overlays (30% hosts show substantial decrease while 20% exhibit marginal decrease). Synergy improves latency by a factor of five or more for many home overlay routing paths. About 45% of hosts exhibit a ratio close to one. The latency ratio for less than 5% of hosts is larger than one, due to overhead from the Synergy divert mechanism (inter-process communication overhead) and due to overloaded hosts.

Synergy also increases throughput for most participating hosts. To compare throughput with and without Synergy, we again measured and averaged 120,000 throughput samples

[1]To compute latency, the source timestamps each data packet sent. The latency is estimated when the packet arrives at the destination, and hence clock skew can introduce inaccuracies. Since our purpose is coarse granularity comparison between home overlays and Synergy, the results obtained are considered acceptable.



(a) Cumulative distribution of the ratio of average latency with Synergy to that achieved by independent home overlays.

(b) Cumulative distribution of the ratio of average throughput with Synergy to that achieved by independent home overlays.

(c) Cumulative distribution of the ratio of average loss percentage with Synergy to that achieved by independent home overlays.

(d) A comparison of latency, throughput, and loss rate achieved by Synergy and independent home overlays.

Fig. 4. Latency, throughput, and loss comparison of cooperative and independent overlays.

at receivers. In Fig. 4(b), we plot the cumulative distribution of the ratio of the average Synergy throughput to the average throughput achieved by independent home overlays. The distribution reveals that about 52% of the hosts achieve higher throughput via Synergy than home overlays alone by a maximum factor of seven (9% hosts show substantial increase while 43% exhibit marginal increase). Among the remaining hosts, 44% use the same paths, while 4% traverse worse paths in Synergy than in their home overlays.

Fig. 4(c) illustrates the cumulative distribution of the ratio of loss percentage results obtained from Synergy and those observed with independent home overlays in the same experiment. About 50% of hosts in Synergy improve loss 1 to 5 times over independent home overlays. The remaining 50% exhibit a ratio close to or slightly larger than one. Again, Synergy packet processing overhead and overloaded hosts are possible reasons. In addition, the mechanism for RON loss measurements and averaging may give inaccurate long-term averages. Finally, RON uses bi-directional link information for uni-directional loss measurement assuming symmetric loss percentages. Network paths, however, are found to be asymmetric in losses.

We compare latency, throughput, and loss averaged for all hosts in Fig. 4(d). The results for each metric are normalized (or divided) by the mean values. In the figure, higher normalized values of latency and percentage loss indicate worse performance while higher throughput means better performance. The figure shows that the three metrics exhibit similar trends. This implies that a route optimized for one of the three metrics promises high performance in terms of the other two metrics. This result is not surprising, since overload or faults cause delays and losses, and throughput is a function of delay and loss. This correlation among delay, loss, and throughput may not, however, hold with wireless or satellite connections with random losses.

To study how heterogeneous overlays maintain their individual performance goals, we examine the performance of Synergy versus independent home overlays for the particular metric of importance to each overlay (latency, loss, or throughput) in Table I. Synergy gives improvements in all the overlays for their respective metrics, except overlay 4. Overlay 5 suffered from performance variation as conditions changed significantly among experiments. Latency exhibits the most significant improvement – approximately 10% or more for overlays 1-3. Thus, Synergy allows heterogeneous overlays to maintain their routing goals while cooperating. More complex scenarios with multiple independent constraints, and overlays with heterogeneous service goals will be investigated in our future work.

TABLE I

EXPERIMENTAL RESULTS FOR THREE PERFORMANCE GOALS

Overlay	Metric	Mean	Max	Min
1	Latency	0.8986	1.0002	0.6347
2	Latency	0.8388	1.0003	0.2587
3	Latency	0.9363	1.0000	0.6898
4	Throughput	0.9997	1.0056	0.9817
5	Throughput	1.1341	2.9992	0.3708
6	Throughput	1.0011	1.0039	0.9999
7	% Loss	0.9663	1.0235	0.8392
8	% Loss	0.9573	1.0022	0.7945

The primary objective of cooperative forwarding is to overcome path outages that are commonly experienced in the Internet. As discussed in [5], an outage can be defined as the length of time during which no packets get through a path. We consider 30 and 50% loss as points at which performance is unacceptable. To measure outages in home overlays and Synergy, we compare the averaged packet loss in both cases, following a methodology similar to [5]. Fig. 5 shows that the majority of points in the scatter plot are located below the $y = x$ line, i.e., Synergy has lower packet loss than home overlays in most cases. There is a higher number of points for home overlays above 0.3 or 0.5 loss (corresponding to 30 or 50% loss) than Synergy. This means that Synergy decreases the number of path outages. The figure also illustrates that there are approximately 10 points with $x = 1$ and $y < 1$. For such network paths, Synergy helps overlay connections route around outages existing in home overlays. Many points are concentrated around the point of $x = 0$ and $y = 0$. Some points, however, can also be found around the point of $x = 1$ and $y = 1$. The packet loss for Synergy exceeds home overlays for a few of these points.

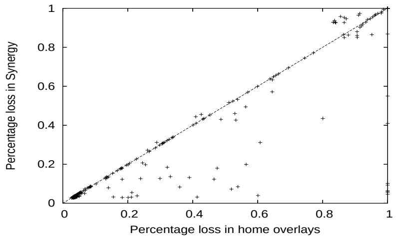


Fig. 5. Average packet loss for 660 network paths in home overlays and Synergy, where half of the paths are in home overlays and the other half is in Synergy. In the figure, 112 points exceed 0.3 for Synergy (horizontal line) while 128 points exceed 0.3 for home overlays (vertical line). In addition, 89 points exceed 0.5 for Synergy, but 105 points exceed 0.5 for home overlays.

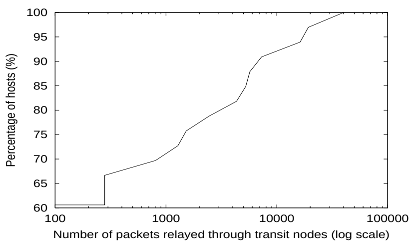


Fig. 6. Cumulative distribution of the number of packets relayed through transit nodes in cooperative overlays.

To investigate the non-uniform contribution of transit nodes in Synergy, we plot the distribution of packets transferred via Synergy channels in Fig. 6. We observe that only 10% of transit nodes participate in relaying over 10,000 packets. Most of the transit nodes (approximately 60%) deliver fewer than 100 packets. This result validates the conjecture that a few strategically located hosts can play a major role in Synergy routing, which confirms the effectiveness of our host export mechanism.

As previously discussed, Synergy imposes additional delay for diverting between the Synergy component and the ports of a home overlay. This is similar to the delay incurred by divert sockets to forward data in the home overlay itself, which is about 220 ms [5]. Since the current version of Synergy reuses the routing and probing mechanisms of RON, a similar overhead is added. Based upon [5], RON routing and probe traffic would be about 14 kbps for 32 hosts (this is the number of hosts which use Synergy in our experiment).

In addition to routing and probe traffic, Synergy injects extra traffic to broadcast the export utility and priority of each host. We analyze this overhead similar to the analysis in [5]. A Synergy export utility packet comprises 60 bytes of header information plus 20 bytes for the export utility of each peer. A host announces its export utility to the overlay agent every $\texttt{EXPORT\_INTERVAL} = 14$ seconds. For this traffic, hosts generate a total of $\frac{8m(60+20)(m-1)}{14}$ bps where m is the number of hosts in the home overlay. Upon receiving these packets from all the hosts, the agent aggregates export utilities. The agent responds to each host with an aggregated export utility packet. The aggregated packet adds $\frac{8m(80)(m-1)}{14}$ bps. The bandwidth consumed by this traffic is thus $\frac{1280m(m-1)}{14} = 91.43m(m-1)$ bps. In our experiments, $m = 8$ and the number of overlays is 8. Therefore, the total traffic is about 41 kbps, for the worst case of p and q representing all nodes.

Hosts also compute and send their priority value to the agent. This priority packet needs 48 bytes for the header and 10 bytes for the priority. Thus, a total of $58n$ bytes is created, where n is the number of hosts that run Synergy in that home overlay. The agent then forwards the aggregated priorities to the primary agent. This packet consists of 60 bytes of header information and $10n$ bytes of priority information for each peer. The primary agent replies to each overlay agent with $48 + 10l$ bytes of priorities for other hosts, where l is the total number of hosts in the Synergy network. An agent broadcasts this packet to the home overlay. Therefore, the traffic consumed by an overlay is $58n + 60 + 10n + (48 + 10l)(n + 1)$ bytes where $l = 8n$. We again use 14 seconds for exchanging priority packets. This is about 2.1 kbps when $n = 4$ in the experiment. Thus, the total traffic for 8 overlays is $2.1 \times 8 = 16.8$ kbps. Overall, the overhead created by Synergy is about $14 + 41 + 16.8 = 71.8$ kbps. This overhead is reasonable, and can be reduced by increasing the intervals, or the values of p and q.

In our experiments, we compare Synergy with individual overlays rather than with a single large overlay network constituting all nodes (which would undoubtedly yield higher performance). This is because a single large overlay faces administrative conflicts, and heterogeneity and scalability challenges. We argue that individual overlays would be more likely to participate in Synergy than in a single large overlay for the following reasons. First, individual overlays maintain heterogeneity among overlays. Second, the transparent service of Synergy facilitates the deployability of Synergy. Third, the host export mechanism tackles scalability challenges via reducing the number of Synergy-participating nodes. A single large overlay does not exhibit these three properties.

C. Simulation Results

We also simulate a simplified version of Synergy to quantify its advantages as underlying route quality varies. This version of Synergy does not include the agent selection, host export, or priority mechanisms discussed in Section III. We use the packet-level overlay network simulator myns [21]. We simulate a router-level Transit-Stub topology generated by GT-ITM [22]. The total number of nodes is 492, with 2 transit domains, 6 nodes per transit domain, 5 stub domains per transit node, and 8 nodes per stub domain. We manipulate 0-45% of the link delays after route computation, to introduce long delay links (consistent with sub-optimal inter-domain policy routes, such as those observed in [4]). We only consider latency as a routing metric in this simulation.

Fig. 7 compares the mean delays of cooperative overlays (using Synergy) and home overlays (here, they are End System Multicast [2] overlays). We create 10 different ESM overlays with 20 hosts each (total number of hosts is 200). We measure the mean end-to-end delay along overlay links among all pairs of hosts. As expected, the figure shows that Synergy

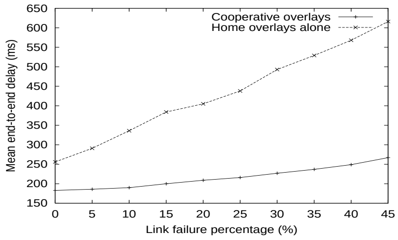


Fig. 7. Mean delay: Synergy versus home overlays.

gives lower latency than home overlays. More importantly, the gap between Synergy and home overlays increases as routing pathologies increase.

VI. Related Work

The idea of overlay cooperation has been studied in different contexts. A grand challenge in networking research discussed in a recent NSF-sponsored workshop report [12] is "simultaneously co-existing overlays." Broadcast Federation [11] investigated cooperation among multicast/broadcast overlay and IP-level networks. Our focus, however, is not limited to overlay broadcasting – we consider more general characteristics of overlay interactions. Nakao *et al.* [18] have designed a shared routing underlay that provides large-scale and coarse-grained network layer information to overlay nodes. Tomography-based overlay network monitoring that requires $O(n)$ measurements for all the $O(n^2)$ overlay paths was proposed in [20]. The approach reduces measurement overhead, at the expense of computational overhead. MACEDON [23] is an infrastructure that enables users to generate code for overlay algorithms using a concise script language. Operating system support and network management for large-scale overlay networks were studied in [24]. Open DHT [25] is a publicly accessible distributed hash table (DHT) service that clients can use instead of running their own DHT nodes. The primary focus of Open DHT is on DHT service availability while ours is on general overlay interactions.

Several theoretical studies (e.g., [26], [27]) have recently shown that selfish routing may cause suboptimal performance. In contrast, Qiu *et al.* [13] have reported that selfish source and overlay routing indeed achieve close to optimal average latency in Internet-like environments, at the expense of significantly increased congestion on certain links. Overlay networks that detect performance degradation of current routing paths and re-route through other hosts include Detour [4] and RON [5]. Rewaskar and Kaur [28] recently studied the tradeoff between performance gain in an overlay network and overhead incurred. A method to compose a resilient and high performance overlay mesh using interleaved spanning trees was proposed in [29].

Multicast and peer-to-peer systems have also been successful services on overlay networks. A number of overlay multicast algorithms have been proposed over the last four years. End System Multicast (or Narada) [2] was one of the earliest and most tested approaches. Overcast [3] provides scalable and reliable single-source overlay multicast. Castro *et al.* [30] compare a number of overlay multicast algorithms built using peer-to-peer overlay networks. Gnutella [31] and KaZaA [32] are commercial peer-to-peer file systems for music and video sharing. For better scalability, several recent peer-to-peer systems, including Chord [9] and Pastry [10], use efficient distributed hash table (DHT) lookup algorithms over overlay networks.

VII. Conclusions and Future Work

We have presented an overlay internetworking architecture for allowing autonomous, possibly heterogeneous, overlay networks to collaborate. The architecture aims at supporting transparent interactions among different overlays to improve performance and promote information sharing. We have designed Synergy – an architecture that includes three essential mechanisms: overlay agents, utility-based host export, and fair inter-overlay routing. To the best of our knowledge, this is the first work to propose a deployable and general overlay internetworking system.

We have implemented a Synergy prototype and performed experiments on PlanetLab. The key findings from our experiments are that: (i) Synergy improves the performance of overlay connections with respect to latency, throughput, and loss; (ii) Synergy reduces the number of path outages; (iii) Performances with three routing metrics – latency, throughput, and loss rate – are commensurate with one another; (iv) Synergy effectively considers heterogeneous routing metrics; (v) The additional overhead of Synergy is marginal; (vi) A few strategically located hosts play a major role in Synergy routing; and (vii) Synergy benefits increase when underlying routing pathologies increase.

We plan to conduct larger-scale experiments with heterogeneous overlays (e.g., complex constraint-based routing systems, overlay multicast, and peer-to-peer systems). These experiments will allow us to quantify the effectiveness and practicality of overlay cooperation in realistic environments. We will also extend Synergy to support other types of cooperative overlay services, and study the complexity, scalability, and security issues involved. Eventually, we hope that our Synergy system can serve as an enabler for the design and evaluation of revolutionary Internet routing and service architectures.

VIII. Acknowledgments

This research has been sponsored in part by NSF grant ANI-0238294 (CAREER), and the Schlumberger Foundation technical merit award. The authors would like to thank David G. Andersen (CMU) for the RON code; Suman Banerjee (U. of Wisconsin) for the myns simulator; Akihiro Nakao (Princeton U.) for answering many of our questions; and Kihong Park, Dongyan Xu, and David Yau (Purdue U.) for several inspiring discussions.

References

[1] J. Saltzer, D. Reed, and D. Clark, "End-To-End Arguments In System Design," *ACM Transactions on Computer Systems*, pp. 277–288, November 1984.

[2] Y. Chu, S. Rao, S. Seshan, and H. Zhang, "Enabling Conferencing Applications on the Internet using an Overlay Multicast Architecture," in *Proc. of ACM SIGCOMM*, August 2001, pp. 55–67.

[3] J. Jannotti, D. Gifford, K. Johnson, M. Kaashoek, and J. OToole, "Overcast: Reliable multicasting with an overlay network," in *Proc. of OSDI*, October 2000.

[4] S. S. et al., "Detour: a Case for Informed Internet Routing and Transport," *IEEE Micro*, vol. 1, no. 19, pp. 50–59, January 1999.

[5] D. Andersen, H. Balakrishnan, M. Kaashoek, and R. Morris, "Resilient Overlay Networks," in *Proc. of ACM SOSP*, October 2001, pp. 131–145.

[6] J. Byers, J. Considine, M. Mitzenmacher, and S. Rost, "Informed Content Delivery Across Adaptive Overlay Networks," in *Proc. of ACM SIGCOMM*, August 2002.

[7] F. Dabek, M. F. Kaashoek, D. Karger, R. Morris, and I. Stoica, "Wide-area Cooperative Storage with CFS," in *Proc. of ACM SOSP*, 2001, pp. 202–215.

[8] L. Subramanian, I. Stoica, H. Balakrishnan, and R. Katz, "OverQoS: An Overlay Based Architecture for Enhancing Internet QoS," in *Proc. of USENIX NSDI*, March 2004.

[9] I. Stoica, R. Morris, D. Liben-Nowell, D. R. Karger, M. F. Kaashoek, F. Dabek, and H. Balakrishnan, "Chord: A Scalable Peer-to-peer Lookup Protocol for Internet Applications," in *Proc. of ACM SIGCOMM*, August 2001, pp. 149–160.

[10] A. Rowstron and P. Druschel, "Pastry: Scalable, Decentralized Object Location and Routing for Large-scale Peer-to-Peer Systems," in *Proc. of ACM/IFIP Middleware*, 2001.

[11] Y. Chawathe and M. Seshadri, "Broadcast Federation: An Application-layer Broadcast Internetwork," in *Proc. of ACM NOSSDAV*, May 2002, pp. 117–126.

[12] M. Ammar and Co-authors, "Report of the National Science Foundation Workshop on Fundamental Research in Networking," April 2003, http://www.cs.virginia.edu/∼jorg/workshop1.

[13] L. Qiu, Y. R. Yang, Y. Zhang, and S. Shenker, "On Selfish Routing in Internet-Like Environments," in *Proc. of ACM SIGCOMM*, August 2003.

[14] J. Crowcroft, S. Hand, R. Mortier, T. Roscoe, and A. Warfield, "Plutarch: An Argument for Network Pluralism," in *Proc. of the ACM SIGCOMM Workshop on Future Directions in Network Architecture*, August 2003.

[15] K. Sollins, "Designing for Scale and Differentiation," in *Proc. of the ACM SIGCOMM Workshop on Future Directions in Network Architecture*, August 2003.

[16] X. Yang, "NIRA: A New Internet Routing Architecture," in *Proc. of the ACM SIGCOMM Workshop on Future Directions in Network Architecture*, August 2003.

[17] L. Peterson, T. Anderson, D. Culler, and T. Roscoe, "A Blueprint for Introducing Disruptive Technology into the Internet," in *Proc. of the HotNets-I*, October 2002.

[18] A. Nakao, L. Peterson, and A. Bavier, "A Routing Underlay for Overlay Networks," in *Proc. of ACM SIGCOMM*, August 2003.

[19] S. Banerjee, B. Bhattacharjee, and C. Kommareddy, "Scalable Application Multicast," in *Proc. of ACM SIGCOMM*, August 2002.

[20] Y. Chen, D. Bindel, and R. Katz, "Tomography-based Overlay Network Monitoring," in *Proc. of ACM SIGCOMM Internet Measurement Conference*, October 2003.

[21] S. Banerjee, "The myns simulator," http://www.cs.umd.edu/∼suman/research/myns/.

[22] E. Zegura, K. Calvert, and S. Bhattacharjee, "How to Model an Internetwork," in *Proc. of IEEE INFOCOM*, vol. 2, March 1996, pp. 594 –602.

[23] A. Rodriguez, C. Killian, S. Bhat, D. Kostic, and A. Vahdat, "MACEDON: Methodology for Automatically Creating, Evaluating, and Designing Overlay Networks," in *Proc. of USENIX NSDI*, March 2004.

[24] A. Bavier et al., "Operating System Support for Planetary-Scale Network Services," in *Proc. of USENIX NSDI*, March 2004.

[25] B. Karp, S. Ratnasamy, S. Rhea, and S. Shenker, "Spurring Adoption of DHTs with OpenHash, a Public DHT Service," in *Proc. of IPTPS*, February 2004.

[26] E. Koutsoupias and C. Papadimitriou, "Worst-Case Equilibria," in *Proc. of the 16th Annual Symposium on Theoretical Aspects of Computer Science*, 1999, pp. 404–413.

[27] T. Roughgarden and E. Tardos, "How Bad is Selfish Routing?" *Journal of ACM*, vol. 49, no. 2, pp. 236–259, 2002.

[28] S. Rewaskar and J. Kaur, "Testing the Scalability of Overlay Routing Infrastructures," in *Proc. of the Passive and Active Measurements Workshop*, April 2004.

[29] A. Young, J. Chen, Z. Ma, A. Krishnamurthy, L. Peterson, and R. Wang, "Overlay Mesh Construction Using Interleaved Spanning Trees," in *Proc. of IEEE INFOCOM*, March 2004.

[30] M. C. et al., "An Evaluation of Scalable Application-level Multicast Built Using Peer-to-peer Overlays," in *Proc. of IEEE INFOCOM*, March-April 2003.

[31] "Gnutella," http://gnutella.wego.com.

[32] "Kazaa," http://www.kazaa.com.