Amorpheus: A Peer-to-Peer Approach to Resilient Overlay Routing

Ben Leong, Ji Li, Dah-Yoh Lim
{benleong, jli, dylim}@mit.edu

Abstract— **In [1], Anderson et. al. identified a serious problem with the current state of inter-domain routing in the Internet and proposed Resilient Overlay Networks[2] (RONs) as a way to allow the network to deliver robust Internet communications even under cases of isolated link failures by routing around the failures.**

In this paper, we describe the architecture and operation of the Amorpheus overlay network, which is a new peer-to-peer approach to resilient overlay routing, and provide a preliminary evaluation of its performance. The Amorpheus network is a self-organizing peer-to-peer network that uses topological information to dynamically grow from a nucleus network into an overlay routing network. The network uses three reference nodes to generate a system of three gradient fields which are used by nodes to make stateless routing decisions based only on local information without the need for any global routing tables.

Our simulation results for the Waxman and 2-Level Hierarchical (*Transit-Stub*) topology models suggest that an Amorpheus network between 10 to 80 nodes (and comparable to RON) in size is able to recover from isolated link failures with a high probability of success. Our results suggest that our approach to resilient overlay routing is a viable one, though some further refinements to the current routing algorithm will be needed to allow the system to scale beyond 80 nodes.

Keywords—**Internet, peer-to-peer, overlay network, routing.**

I. Introduction

In [1], Anderson et. al. identified a serious problem with the current state of inter-domain routing in the Internet – because of the complex routing policies of the ISPs managing the various Autonomous Systems (ASs) and the nature of the Border Gateway Protocol (BGP), it is possible for path outages to cause significant disruptions to communications[3]. Anderson et. al. proposed Resilient Overlay Networks[2] (RONs) as a way to deliver robust Internet communications even under cases of isolated link failures by routing around the failures. A RON is essentially a centrally managed, tightly-coupled overlay network where the individual nodes continuously monitor the state of the inter-nodal links. It allows Internet traffic to be re-routed by identifying existing redundant paths in the underlying Internet in the event that a link failure does not completely cut off a node from the rest of the Internet.

In this project, our goal is to build a resilient overlay network using a peer-to-peer approach. In our approach, hosts that wish to use the overlay network as a resilient forwarding network will join the network. Under normal circumstances, they communicate directly with each other. However, in the event of path outages, nodes will attempt to communicate over the overlay network. If the network is not partitioned, the network will attempt to route packets between the 2 nodes using available links.

Our peer-to-peer overlay network incorporates topological information in its formation and routes using a novel gradient routing algorithm. The virtual routing topology is a planar triangulation and individual nodes interact with each other according to a fixed set of rules to set up 3 gradient fields that allow routing decisions to be made based only on local information.

The rest of the paper is organized as follows. In Section II, we review existing related work on resilient overlay networking, peer-to-peer systems and routing algorithms. In Section III, we present an overview of the architecture and operation of an Amorpheus network. In Section IV, we discuss our gradient routing algorithm. In Section V, we present our evaluation of the Amorpheus network using our network simulator. Finally, in Section VI, we present our conclusions and expected future work.

II. Related Work

In his thesis[4], Anderson reported that 8 major outages, each lasting over thirty minutes, were observed in a 71-hour period on 156 communication paths across the Internet in his experiment. All of them were recoverable with RON, which demonstrates that resilient overlay networking is viable and that physical redundant paths are in fact quite common in the Internet. The fact that the major outages are in the order of minutes motivates us to attempt to solve the same problem using a peer-to-peer approach, since it should not require the tight-coupling between nodes to route packets on an overlay network in a time shorter than it takes for the underlying network to recover by itself.

RON nodes are organized as a clique and every node has to continously monitor the status of all connections to all the other nodes. Since there are $\frac{n}{2}(n-1)$ links that have to be monitored at any one time, the topology maintenance overhead is $O(n^2)$. RON has 2 membership managers: one is a static membership mechanism that loads all members from a file, while the other is a dynamic flooding-based announcement protocol. Both these factors limit the scalability of RON in terms of number of nodes[1].

We note that for RON[4], universities, labs and companies located in various ASs were convinced to set up RON nodes to deploy the overlay network. We believe that the scalability of RON is also somewhat limited by the need for centralized coordination at the user level. If not for existing relationships, what is the incentive for institutions like universities, labs and companies agree to let regular users set up nodes on their premises? The question then arises – how can we deploy overlay routing networks without centralized coordination? This is not simply a technical problem, but also a question of how to set up the correct incentive structure so that various parties have the incentive to deploy the overlay network. In this regard, we believe that a peer-to-peer approach is the right answer because in such an approach, hosts which want to use the services offered by the network will have the incentive become members of the network and by joining it, they will contribute to the routing resources of the network.

Many peer-to-peer systems have been proposed and developed since the advent of file sharing applications like Napster[5], Gnutella[6] and Freenet[7]. Neither Napster nor Gnutella employ any form of routing, but Freenet is more structured and can also be considered an overlay network. Bandwidth utilization in these networks are usually highly inefficient as they are not optimized for routing.

The other major category of peer-to-peer systems are Distributed Hash Tables[8][9][10] [11][12] (DHTs) . These systems are usually more concerned with the efficient lookup of the node corresponding to a specific key rather than in ensuring that nodes in the network are connected together according to any form of proximity measure or that routing on the underlying IP network is efficent. As a result, the routing of packets (when the systems are used for any form of routing) in such systems following the sequence of a key-lookup can be excruciatingly inefficient.

In our view, the routing algorithm for the major DHT systems can be divided into 2 major classes: one class of such systems use one-dimensional namespaces and maintain some finger-like pointers for routing. They include Chord[8] and Plaxton-based[13] systems (Pastry[9] and Tapestry[10]). The average number of routing hops for such systems is $O(\log n)$ and the size of the routing tables at each node is also $O(\log n)$. The other major class is embodied by CAN[14]. In CAN, nodes are each mapped into a region within d-dimensional coordinate space and addresses are points inside the coordinate space. At each step, each node simply routes a packet to the neighbor that makes the most progress towards the destination coordinate. Therefore, CAN maintains a routing table of size $O(d)$ (constant in n), which significantly smaller than the previous schemes, but the tradeoff is that the average routing path is now greater in terms of number of hops at $O(d \cdot n^{1/d})$.

In our scheme, every node only maintains information about its immediate neighbors. Even though in the worst case for a random triangulation network, a node can have $n-1$ edges, the average number of edges per nodes is usually less than six[15]. This means that each node in our network maintains on average $O(1)$ state and our storage requirements are thus similar to CAN. Because our routing algorithm is stateless, there is no need to maintain state for routing and the routing table is equivalent to the neighbourhood table ($O(1)$).

In general, DHT-based routing is inefficient because topology information is not incorporated into the network formation process by any of the above schemes, though Ratnasamy et. al. have recently proposed an enhanced version of CAN[2][16]. In this light, the claim that the average number of routing hops is $O(\log n)$ does not guarantee good routing performance since adjacent hops in the overlay could involve nodes located at opposite ends to the Internet. In Amorpheus, we attempt to ensure that adjacent nodes are indeed physically close to their neighbours in terms of latency. The key difference between Amorpheus and the enhanced-CAN is that in CAN, the virtual routing topology is a virtual d-dimensional Catesian coordinate space on a d-torus, while in Amorpheus, the virtual routing topology is a 2-dimensional mesh. Our hypothesis is that a triangulation mesh is able to capture the structure of the underlying Internet topology better than the above schemes. Our hope is that by choosing a virtual topology that is more similar to the underlying routing infrastructure, we will be able to achieve greater routing efficiency. We also believe that in the event of node departures and losses, our repair algorithm is simpler and consequently allows for faster recovery.

[1] It is stated in [4] that the the size of any given RON is limited to between two and fifty nodes to allow RONs to be more aggressive in path exploration and maintenance.

[2] In the enhanced CAN, measured round-trip times are used as a signature to help determine if 2 clients belong to the same *bin* (i.e. are likely to situated in close proximity).

There are many existing conventional routing algorithms, such as Bellman-Ford[17] and link-state[18]. However, such algorithms are not likely to work well in a peer-to-peer environment where nodes join and leave frequently, because it is impossible to get a globally-consistent view of the entire system at given moment in time under such circumstances. In fact, we believe that in a rapidly-changing transient environment, the joining and/or leaving of a network node should only result in localized changes in the network and to route in such a system, the routing algorithm has to be stateless[3]. In our gradient routing approach, we hope to address transients by sending packets in a correct general direction and perform fine-grained high-precision routing only in the neighbourhood of the destination node.

In some ways, our gradient routing algorithm is similar to the greedy routing algorithms employed by geographic routing schemes for ad hoc routing networks and is almost identical in spirit to Compass Routing[19], proposed by Kranakis et. al. . Unlike the geographic ad hoc routing scheme proposed by Karp and Kung[20], our overlay network does not require perimeter routing for correctness because our routing topology is well-connected convex hull. In addition, the nodes in our network need not be GPS-enabled or have any explicit knowledge of their geographic location.

As an overlay topology, our topology is most similar to that adopted by Liebeherr et. al. in their application-layer multicast overlay network[15][21]. Their overlay is formed incrementally from a Delaunay triangulation[4] and Compass Routing is used to make local forwarding decisions at each node. In our network, our overlay is a triangulated planar graph, but the triangles are not constrained to be Delaunay triangulations. In this light, the constraints on our overlay topology are somewhat more relaxed.

III. System Overview

In Amorpheus, we start with an initial nucleus network consisting of 3 nodes, which we call reference nodes. The network is formed incrementally as new nodes are connected to the core network and all nodes are identified by their virtual distances from these 3 reference nodes. These virtual distances form natural gradient fields which we will use to route packets using our gradient routing algorithm. Routing is discussed in Section IV.

We have assumed that a bootstrapping mechanism exists for all new nodes wishing to join an Amorpheus network and they can find the IP address of some node that is already connected to the network. Perhaps a list of reference nodes can be published or DNS can be used. We believe that this is a reasonable assumption in view that there are some games like FreeCiv[22], where meta-servers have been set up as dynamic registries. The key intuition is that although many nodes will be joining and leaving at any one time, not all nodes are equally volatile. There will certainly be some nodes which are more “stable” and remain in the network for extended periods of time.

In Amorpheus, the nodes are classified as either *core* or *boundary* based on their location in the virtual routing topology. *Core* nodes are logically enclosed by other nodes while *boundary* nodes demarcate the logical boundary (i.e. convex hull) of the triangulated mesh.

A. Node Identifier

The identifier for each node (node-ID) is specified by two components: a location identifier and a zone identifier. These identifiers are determined when the node joins the network, and are not generated randomly like the identifiers for most other DHT systems. We believe that in order to route efficiently in a stateless manner, it is crucial that destination nodes be identified with location-dependent identifiers that reflect the point of connectivity to the network. Hence, we attempt to create a logical, virtual, mesh routing topology with location-dependent addressing in order to model and approximate the logical connectivity in the real underlying network. Nodes attempt to connect to the virtual routing topology at points which are physically closest (in terms of latency measures) by systematically probing the network and measuring roundtrip times (RTTs).

The location identifier is a distance vector which contains the relative distances from this node to 3 reference nodes. Distance is represented by a n-bit integer where n is arbitrary and chosen to accommodate the maximum number of nodes that are expected to join the overlay network. We will represent the location identifier as triples of the form $\{x,\, y,\, z\}$. The location identifiers of the 3 initial reference nodes, which we shall call x^*, y^* and z^*, are $(0, 2^{n-1}, 2^{n-1})$, $(2^{n-1}, 0, 2^{n-1})$ and $(2^{n-1}, 2^{n-1}, 0)$ respectively.

Although we have a location identifier for each node in our system, naive routing based solely on the location identifier will often cause packets to get stuck in some potential minima within the gradient routing field. To address

[3]By stateless routing, we mean that a node can decide to which neighbour it should forward a given packet based only on local information, i.e. the information on its neighbours and information contained on the packet itself.

[4]A Delaunay triangulation for a set of vertices $\mathcal{A}$ is a triangulation graph with the defining property that for each circumscribing circle of a triangle formed by three vertices in $\mathcal{A}$, no vertex of $\mathcal{A}$ is in the interior of the circle.

this problem, we introduced a zone identifier to allow us to perform coarse-grained routing initially towards the vicinity of the destination node before routing is done using the location identifier.

The zone identifier divides the network namespace into three logical zones according to the virtual, logical "proximity" to the 3 reference nodes, x^*, y^* and z^*. Not surprisingly we call the 3 zones corresponding to these reference nodes x, y and z respectively. Nodes on the boundaries separating any two zones are represented by a combination of two basic zone identifiers, i.e., $\{xy,\ xz,\ yz\}$. In our notation, we denote the zone of a node in the location identifier by putting a bar across the appropriate coordinate. For example, the node $(\overline{131}, 162, 156)$ belongs to the x-zone while the node $(162, \overline{131}, 156)$ belongs to the y-zone. The node $(\overline{55}, \overline{55}, 89)$ lies on the boundary of the x-zone and the y-zone.

B. Supported Network Primitives

The following operations available to all nodes at all times:

- `ping(`*IP-address*`)` : To measure RTT to the node at *IP-address*.
- `probe(`*IP-address*`)` : To ask the node at *IP-address* for its node-ID.
- `probe2(`*IP-address*`,` *ID*`)`: To ask the node at *IP-address* for the following information for node closest to node-ID *ID*:
  - node-ID
  - node-ID and IP addresses of connected neighbours
  - RTTs to connected neighbours
- `connect(`*IP-address*`)` : To attempt to make a connection to an existing Amorpheus network node at IP-address *IP-address* (if it is not already connected to the network).

The following operations are available to a node after it joins the network:

- `send(`*ID*`,` *data packet*`)`: To send the *data packet* to the node at node-ID *ID*.
- `disconnect()`: To initiate a departure sequence and leave an Amorpheus network gracefully.

C. Membership Operations

C.1 Locating Nearest Neighbour

As mentioned, we assume that a prospective Amorpheus node knows the IP address of another node that is already in the network through some out-of-band mechanism. Through this node, the new node attempts to discover the IP addresses of the nodes that most closely match $(\overline{2^{n-2}}, 3.2^{n-2}, 3.2^{n-2})$, $(3.2^{n-2}, \overline{2^{n-2}}, 3.2^{n-2})$, $(3.2^{n-2}, 3.2^{n-2}, \overline{2^{n-1}})$ and $(\frac{1}{3}2^{n-1}, \frac{1}{3}2^{n-1}, \frac{1}{3}2^{n-1})$ (See Figure 1). The intuition here is that the nodes nearest to these coordinates are in different regions of the topology and spread relatively widely apart. Once the new node obtains the IP addresses of the 4 nodes, it pings them and measures the RTTs. The node with the smallest RTT will be chosen as the base node for the next iteration of the search process.

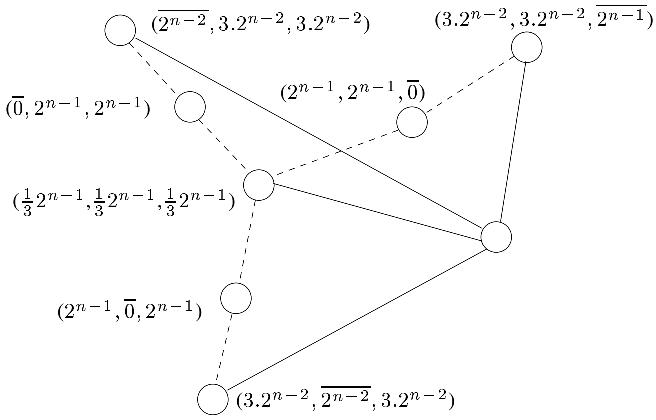

Fig. 1. Finding the nearest neighbour by recursive search.

From the chosen base node, the new node performs a recursive search by probing systematically smaller and smaller regions. More specifically, after the selection of the first base node (0th round), the search radius r_1 is 2^{n-2}. Given that (x,y,z) is the base node for the $(n-1)$th round and r_n is the search radius for the nth round, we search the following 8 locations in the nth round and pick the one with the shortest RTT as the new base node: $(x \pm r_n, y \pm r_n, z \pm r_n)$. After each round, $r_{n+1} = \frac{r_n}{2}$. Eventually, this process will converge.

Once the process converges on a base node, the new node will ask the node for information on its neighours and ping the neighbours. From the list of RTTs, it chooses the node with the smallest RTT as the next base node and repeats this process. The process ends when the new node finds a base node that has the smallest RTT to it compared to all the other nodes in its RTT cache. This base node is then assumed by the new node to be its nearest neighbour.

C.2 Joining the Network

Once a new node finds its nearest neighbour, which we will call n, it will probe all its neighbours to obtain information about their estimated RTTs to each other as well as estimates of its RTT to each of them. Using a simple test to determine whether a point is inside or outside a triangle based on the size of the triangle and distance of the point from the triangle, we can decide where and how the new node should connect to the network.

Case I: New node is inside triangle with n as one of the vertices (See Figure 2)

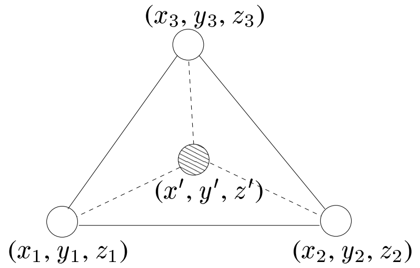

Fig. 2. New node is inside triangle with n as one of the vertices.

This means that the new node is located within the core of the network and that the new node should connect to all 3 vertices of the triangle found. The following values will be used for the location identifier of the new node:

$$x' = \frac{1}{3}(x_1 + x_2 + x_3) \quad (1)$$

$$y' = \frac{1}{3}(y_1 + y_2 + y_3) \quad (2)$$

$$z' = \frac{1}{3}(z_1 + z_2 + z_3) \quad (3)$$

The final configuration is shown in Figure 3.

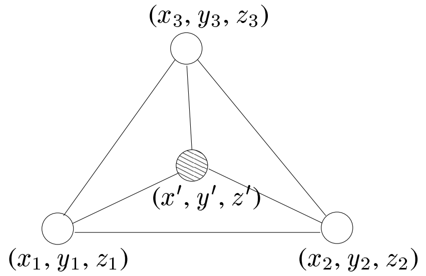

Fig. 3. Final configuration after new node joins the network inside a triangle.

Case II: n lies on a boundary link and the new node is outside the network (See Figure 4)

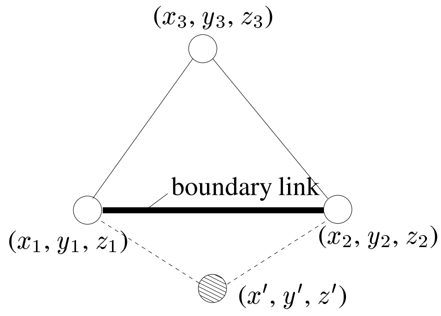

Fig. 4. n lies on a boundary link and the new node is outside the network.

In this case, the new node should connect to the 2 nodes along the boundary link. The 2 new links are set as boundary links and the original link connecting the 2 neighbours becomes a core link. To compute the location identifier for this new node, we consider the new node to be bounded by its 2 boundary nodes and the virtual node at infinity, (x_∞, y_∞, z_∞), as shown in Figure 5

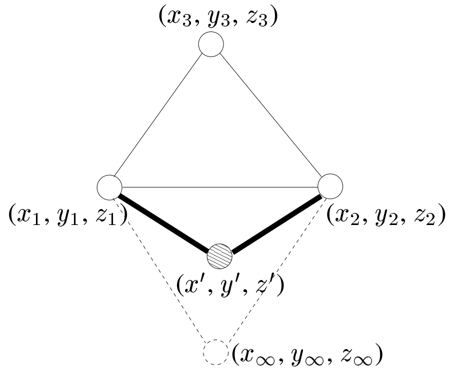

Fig. 5. Final configuration for network when new node joins 2 nodes along a boundary link.

The following values will be used for the location identifier of the new node:

$$x' = \frac{\alpha}{2}(x_1 + x_2) + (1 - \alpha)x_\infty \quad (4)$$

$$y' = \frac{\alpha}{2}(y_1 + y_2) + (1 - \alpha)y_\infty \quad (5)$$

$$z' = \frac{\alpha}{2}(z_1 + z_2) + (1 - \alpha)z_\infty \quad (6)$$

where (x_∞, y_∞, z_∞) = $(2^n, 2^n, 2^n)$. We have chosen $\alpha = \frac{1}{2}$, but it is not entirely clear whether this is indeed the best choice.

Case III: Neither Case I or Case II is satisfied.

If neither Case I nor Case II is satisfied, we conclude that that the new node should not connect to node n. During the initial nearest neighbour search phase and in the initial attempts to join the network above, the new node would have cached the RTT values of all the nodes it has pinged. Once it determines that it should not connect to n, chooses the next nearest node based on its RTT cache and repeats the above process. In certain situations, this process can fail because of transient changes in the system and the new node cannot determine using this algorithm the ideal connection point within the existing topology. In such case, it gives up trying to find the ideal connection point and takes the easy way out by simply joining the network at the nearest node in some logically consistent fashion.

C.3 Choosing a Zone Identifier

After a new node joins the network, it derives its zone identifier from the nodes that is connected to it. To derive the zone identifier, we count the number of characters (x, y and z) for all its new neighbours and adopt the modal character as its zone identifier. In the event of a tie between 2 or more characters, the new node will use both or all 3 respectively as its zone identifier. For example, if the new node joins 3 nodes with identifiers x, y and y, it adopts y as its zone identifier since there are 2 y's and only one x. In another example, if the new node joins 3 nodes with identifiers xy, x and y, it adopts xy as its zone identifier since there are 2 y's and x's.

D. *Topological Repair/Maintenance*

In addition to the join protocol described in Section III-C.2, the network also has to perform a number of other operations when nodes leave the network and perform self-repair in case of nodal failures (both regular and reference nodes). Due to space constraints, these algorithms are not described in this paper. Details are found in [23].

IV. Routing

Packets to be delivered by the Amorpheus network are tagged with a simple header consisting of the node-IDs and the IP addresses of the both the source and destination nodes. When a node receives a packet, it checks whether the destination of the packet matches its IP address. If so, the packet has been delivered correctly. If not, the packet has to be forwarded to one of its neighbouring nodes.

Next, it compares the destination IP address to that of its neighbours and its neighbours' neighbours. If the IP address is found, the forwarding path is obvious. Suppose the packet is to be forwarded beyond 2 hops. Then, to decide which node to forward the packet to, the node compares the destination address of the packet with the location and zone identifiers of all its neighbours to decide which node to forward the packet to. The zone identifier takes first priority. If the forwarding node is not in the same zone as the destination of the packet, the node will forward the packet to the neighbour which has the minimum coordinate in the required zone-coordinate.

Once the packet is routed into the destination zone, the packet will be forwarded to the next neighbour which has a location identifier which changes in the correct direction for at least 2 of the coordinates with the following rules for tie-breaking when there are 2 or more nodes which provide gradient changes in the correct direction:

- Nodes which do not cause any coordinate to "overshoot[5]" the destination are favoured over those that do.
- Nodes that are "closer" to the destination node by a Cartesian measure are favoured over those that are "further" away.

Clearly, nodes also ensure that a packet is not forwarded back to the node from which it was received. Consider the network shown in Figure 6. The routing sequence to route a packet from $(\overline{131}, 162, 156)$ to $(162, \overline{131}, 156)$ is given in Table I.

[5] By "overshoot," we mean that the change in gradient for the coordinate is in the correct direction, but it exceeds the coordinate of the destination.

V. Evaluation

In this section, we present an evaluation of the performance of our Amorpheus network using our network simulator[24]. Our simulator is written in Java and currently has about 9,000 lines of code. The simulation is run on 3 different network topologies generated by the BU Brite Topology Generator[25], namely the Router Waxman model[26], the Power Law model[27] and the 2-Level Hierarchical model[28] (*Transit-Stub*).

A. *Overlay Network Formation*

To understand how our overlay network forms over different network topologies, we generated some simple 20-node networks as shown in Figures 7 to 15. Overall, it seems that the overlay network maps quite naturally into the Waxman and Transit-Stub topologies, but it does not to map quite well into the Power Law topology. The most probably reason is that the Power Law topologies tend to be trees and there is no natural way to map a triangulated mesh into a tree.

B. *Path Optimality of Overlay Network*

It is our hope that by choosing a virtual routing topology that is more similar to the underlying infrastructure, we will be able to achieve greater routing efficiency. In order to measure how well our network performs in this regard, we generated 500-node Waxman and Transit-Stub topologies and formed overlay networks ranging from 40 nodes to 150 nodes over them. For each pair of nodes in the overlay network, we then found the ratio of the minimum latency path between to the minimum latency path in the underlying topology. In the ideal case, if the minimum path in the overlay network matches the minimum path in the underlying topology exactly, the ratio would be one. The results are shown in Figure 16.

Our results suggest that our overlay network is able to capture the structure of the underlying network topology with low path latencies. The path latencies for Waxman topologies tend to be significantly lower than those for Transit-Stub topologies and the average path latency for both topologies tend to decrease as more nodes are added. The latter result is rather intuitive since with more nodes,

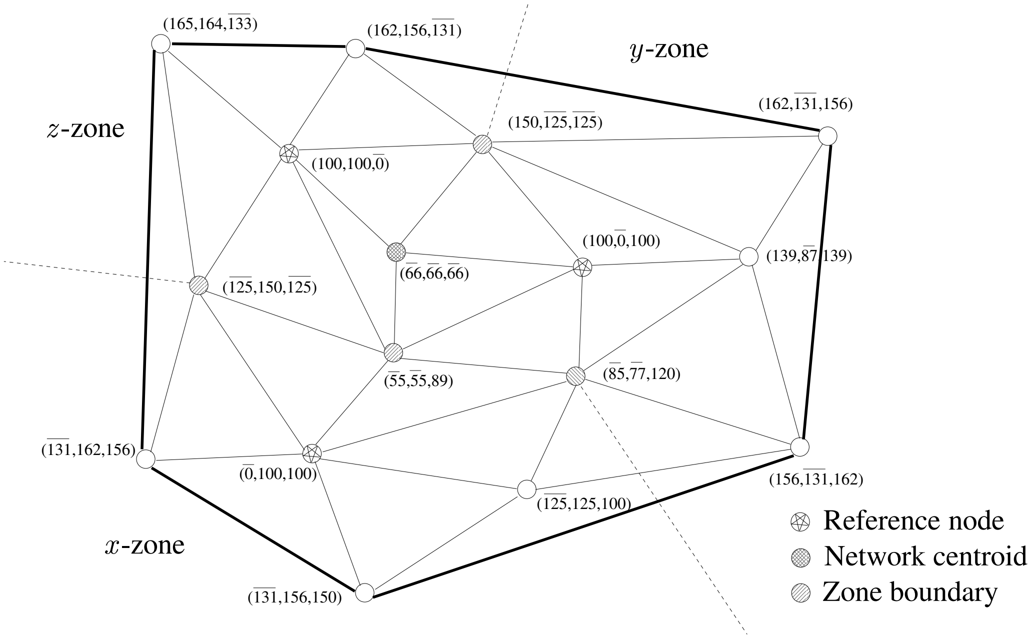

Fig. 6. Sample network.

TABLE I
SAMPLE ROUTING SEQUENCE.

Sequence	Node	Required Gradient	Neighbour	Available Gradient	Comment
1	$(\overline{131}, 162, 156)$	$\uparrow\downarrow -$	$(165, 164, \overline{133})$	$\Uparrow\Uparrow\Uparrow$	Wrong zone
			$(\overline{125}, 150, \overline{125})$	$\downarrow\downarrow\downarrow$	Wrong zone
			$(\overline{0}, 100, 100)$	$\downarrow\downarrow\downarrow$	Minimum y
			$(\overline{131}, 156, 150)$	$- \downarrow\downarrow$	Wrong zone
2	$(\overline{0}, 100, 100)$	$\uparrow\uparrow\uparrow$	$(\overline{125}, 150, \overline{125})$	$\uparrow\Uparrow\uparrow$	Wrong zone
			$(\overline{55}, \overline{55}, 89)$	$\uparrow\downarrow\downarrow$	
			$(\overline{85}, \overline{77}, 120)$	$\uparrow\downarrow\uparrow$	
			$(\overline{125}, 125, 100)$	$\uparrow\uparrow -$	Wrong zone
			$(\overline{131}, 156, 150)$	$\uparrow\Uparrow\uparrow$	Wrong zone
3	$(\overline{85}, \overline{77}, 120)$	$\uparrow\uparrow\uparrow$	$(\overline{55}, \overline{55}, 89)$	$\downarrow\downarrow\downarrow$	
			$(100, \overline{0}, 100)$	$\uparrow\downarrow\downarrow$	
			$(139, \overline{87}, 139)$	$\uparrow\uparrow\uparrow$	
			$(156, \overline{131}, 162)$	$\uparrow\uparrow\Uparrow$	
4	$(139, \overline{87}, 139)$	$\uparrow\uparrow\uparrow$	$(100, \overline{0}, 100)$	$\downarrow\downarrow\downarrow$	
			$(150, \overline{125}, \overline{125})$	$\uparrow\uparrow\downarrow$	
			$(162, \overline{131}, 156)$	$\uparrow\uparrow\uparrow$	
			$(156, \overline{131}, 162)$	$\uparrow\uparrow\Uparrow$	
5	$(162, \overline{131}, 156)$	$- - -$			Done!!

there are more links and hence more choices, so shorter paths are more likely to be available.

C. Routing Performance

In order to evaluate the routing performance of our network, we again generated 2000-node topologies. For each topology, we generated overlay networks with sizes varying from 10 to 80 nodes, with nodes chosen at random. We believe that this is a reasonable model for the deployment of this system on the Internet. For each of these networks, we generated packets between every pair of nodes and the packets are routed in the system. We compared the routing performance of the network in terms of 3 metrics: average latency stretch[6], average number of hops and packet loss rate. The results are shown in Figures 17 to 19.

It can be seen from these graphs that the average latency stretch and the average number of hops increase with the size of the network (which is not surprising). It is however

[6]Latency stretch is a measure of the optimality of routing. The latency stretch for a packet is defined as the ratio of the total delivery latency over the minimum delivery latency assuming that the shortest possible path in the underlying topology is taken by the packet.

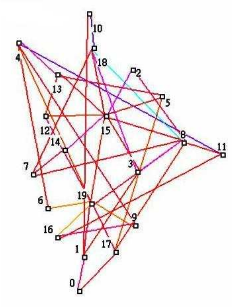

Fig. 7. 20-node Waxman topology generated by Brite.

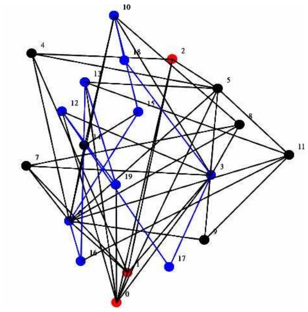

Fig. 8. Overlay network for 20-node Waxman topology.

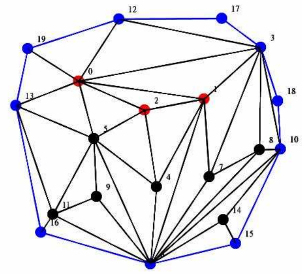

Fig. 9. Virtual routing topology for 20-node Waxman topology.

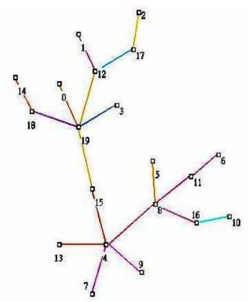

Fig. 10. 20-node Power Law topology generated by Brite.

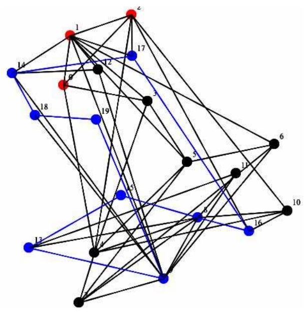

Fig. 11. Overlay network for 20-node Power Law topology.

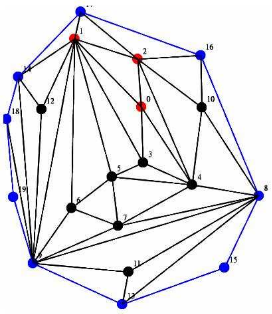

Fig. 12. Virtual routing topology for 20-node Power Law topology.

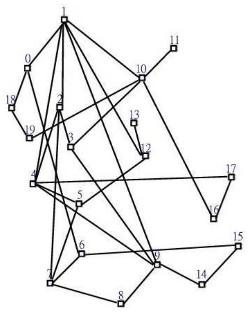

Fig. 13. 20-node Transit-Stub topology generated by Brite.

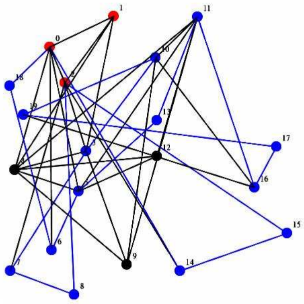

Fig. 14. Overlay network for 20-node Transit-Stub topology.

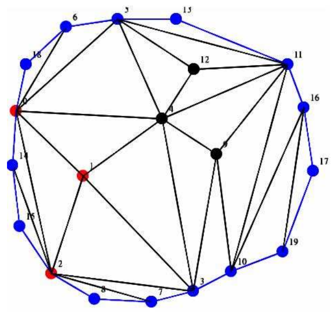

Fig. 15. Virtual routing topology for 20-node Transit-Stub topology.

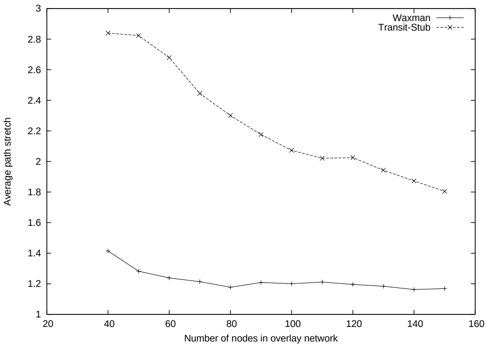

Fig. 16. Path stretch for overlay networks over 500-node topologies

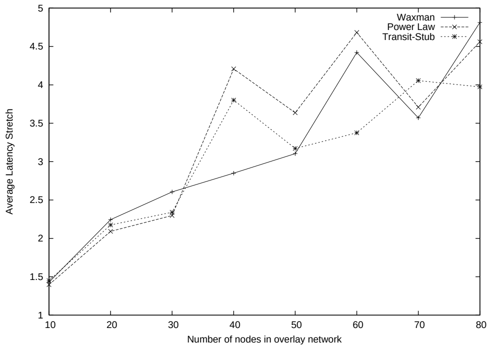

Fig. 17. Comparison of average latency stretch for 2000-node network topologies.

quite surprising that the average latency stretch and the average number of hops is approximately the same for all 3 topologies, even though from the results in Section V-B, it is clear that the path stretch for the Waxman model is significantly better than that for the Transit-Stub model. The results also clearly show that the packet routing failure rate increases as the network size increases. For networks of size similar to RON (below 50 nodes), packet loss rate is less than 10 %. Detailed analysis revealed that packet losses occur because packets can still sometimes get stuck in potential minimas during routing and are not delivered correctly to the destination.

D. Resilient Routing Performance

To evaluate the performance of Amorpheus as a resilient overlay routing network, we generated overlay networks with sizes varying from 10 to 80 nodes over 2000-node for the Waxman and Transit-Stub topologies. We did not use a Power Law topology because for a tree topology, any link failures will result in a irrecoverable network partition. With the generated networks, we randomly chose pairs of nodes and ascertained that we were able route packets successfully from one node to the other. We then shut down half the connections (chosen at random) at the destination node and determined if the network was still able to deliver packets from the source node to the destination. We repeated this process 100 times to measure the success rate of the network in recovering from these link failures. The results for this experiment are shown in Figure 20. We also repeat the above experiment for the case where all connections at the destination, except one, are shut down to see if the network is still able to route through the last remaining connection. The results are shown in Figure 21.

VI. Conclusion and Future Work

The results of Section V-B demonstrate that our overlay topology is able to capture the structure of the underlying network topology reasonably well as suggested by the low path stretch (below 1.4 for Waxman and below 2.9 for Transit Stub). As further work, we plan to obtain corresponding data for other DHT systems to see how our network compares to existing networks.

We are quite disappointed in our routing performance as described in Section V-C, but we believe that it is possible to modify the routing algorithm to perform better. At present, packets tend to get stuck in potential minimas when the number of nodes exceeds 50. Although for the purposes of our resilient routing application, this is acceptable (though not desirable), this significantly limits the scalability of our network. We hope to improve the routing algorithm to enable the system to scale to much larger networks with lower packet loss rates. Nevertheless, we believe that our approach to resilient overlay routing shows promise. We have not been able to obtain data to compare our performance to RON, but we believe that for network sizes similar to RON, we should be able to obtain performance reasonably close to RON in terms of link recover rate, though our network is likely to be able to route with slightly higher latencies.

From our results in Section V-D, it is clear that an Amorpheus network between 10 to 80 nodes (and comparable to RON) in size is able to recover from isolated link failures with a high probability of success. These results suggest that our routing algorithm can indeed be used for resilient overlay routing as anticipated.

Finally, as further work, we plan to explore other topology formation algorithms and packet routing algorithms as well. In particular, we will explore other algorithms for the nearest neighbour search and network connection search. Once we have resolved all network formation and routing issues, our goal is to build and deploy a real application to evaluate the effectiveness of our overlay network over the Internet.

Acknowledgments

The authors wish to thank John Wroclawski, Roger Moh, Steve Bauer, Archit Shah and Indraneel Chakraborty for their most valuable comments and inputs.

References

[1] D. Andersen, H. Balakrishnan, M. Kaashoek, and R. Morris, "The case for resilient overlay networks," in *Proceedings of the 8th Annual Workshop on Hot Topics in Operating Systems (HotOSVIII)*, May 2001.

[2] David G. Andersen, Hari Balakrishnan, M. Frans Kaashoek, and Robert Morris, "Resilient overlay networks," in *Symposium on Operating Systems Principles*, 2001, pp. 131–145.

[3] Bharat Chandra, Mike Dahlin, Lei Gao, and Amol Nayate, "End-to-End WAN service availability," in *Proceedings of the 3rd USITS*, 2001, pp. 97–108.

[4] David G. Andersen, "Resilient overlay networks," M.S. thesis, MIT, May 2001.

[5] Napster, "http://www.napster.com," .

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

[7] Ian Clarke, Oskar Sandberg, Brandon Wiley, and Theodore W. Hong, "Freenet: A distributed anonymous information storage and retrieval system," *Lecture Notes in Computer Science*, vol. 2009, pp. 46+, 2001.

[8] Ion Stoica, Robert Morris, David Karger, Frans Kaashoek, and Hari Balakrishnan, "Chord: A scalable Peer-To-Peer lookup service for internet applications," in *Proceedings of the 2001 ACM SIGCOMM Conference*, 2001, pp. 149–160.

[9] Antony Rowstron and Peter Druschel, "Pastry: Scalable, decentralized object location, and routing for large-scale peer-to-peer systems," *Lecture Notes in Computer Science*, vol. 2218, pp. 329–350, 2001.

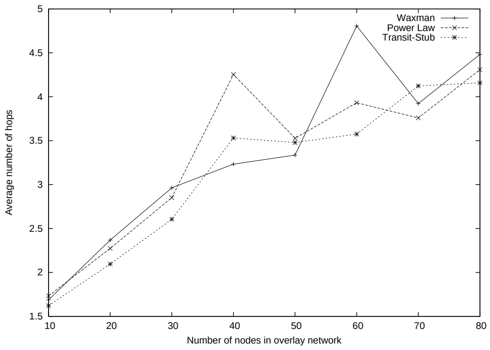

Fig. 18. Comparison of average number of hops for 2000-node network topologies.

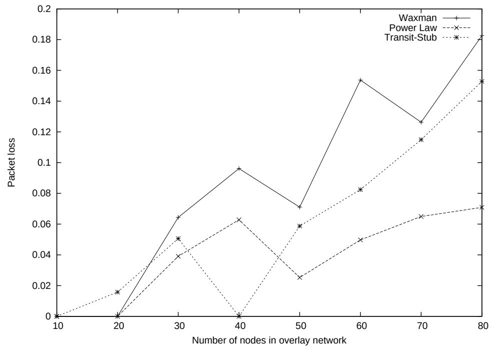

Fig. 19. Comparison of packet loss rate for 2000-node network topologies.

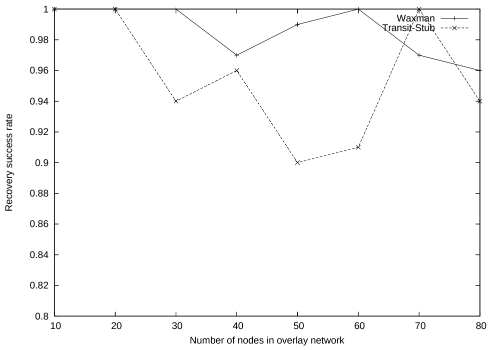

Fig. 20. Connection loss recovery with connections halved for 2000-node network topologies.

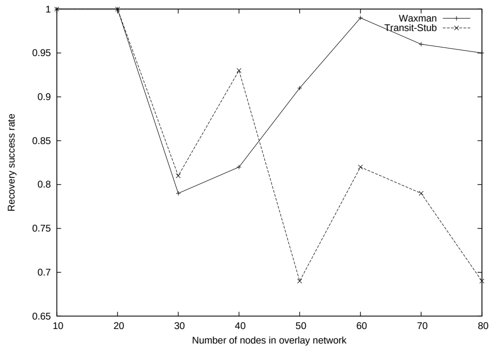

Fig. 21. Connection loss recovery with one connection left for 2000-node network topologies.

[10] B. Y. Zhao, J. D. Kubiatowicz, and A. D. Joseph, "Tapestry: An infrastructure for fault-tolerant wide-area location and routing," Tech. Rep. UCB/CSD-01-1141, UC Berkeley, April 2001.

[11] John Kubiatowicz, David Bindel, Yan Chen, Patrick Eaton, Dennis Geels, Ramakrishna Gummadi, Sean Rhea, Hakim Weatherspoon, Westly Weimer, Christopher Wells, and Ben Zhao, "Oceanstore: An architecture for global-scale persistent storage," in *Proceedings of ACM ASPLOS*. ACM, November 2000.

[12] Sylvia Ratnasamy, Paul Francis, Mark Handley, Richard Karp, and Scott Shenker, "A scalable content addressable network," Tech. Rep. TR-00-010, UC Berkeley, Berkeley, CA, 2000.

[13] C. Greg Plaxton, Rajmohan Rajaraman, and Andrea W. Richa, "Accessing nearby copies of replicated objects in a distributed environment," in *ACM Symposium on Parallel Algorithms and Architectures*, 1997, pp. 311–320.

[14] S. Ratnasamy, P. Francis, M. Handley, R. Karp, and S. Shenker, "A scalable content-addressable network," in *Proceedings of the 2001 ACM SIGCOMM Conference*, San Diego, September 2001, pp. 161–172.

[15] J. Liebeherr, M. Nahas, and W. Si, "Application-layer multicasting with delaunay triangulation overlays," Tech. Rep. CS-2001-26, University of Virginia, May 2001.

[16] S. Ratnasamy, M. Handley, R. Karp, and S. Shenker, "Topologically aware overlay construction and server selection," in *Proceedings of IEEE INFOCOM'02*, June 2002.

[17] R. Bellman, "On a routing problem," *quarterly of Applied Mathematics*, vol. 16, pp. 87–90, 1958.

[18] J. Moy, "OSPF version 2," *RFC 2328*, April 1998.

[19] Evangelos Kranakis, Harvinder Singh, and Jorge Urrutia, "Compass routing on geometric networks," in *Proc. 11 th Canadian Conference on Computational Geometry*, Vancouver, August 1999, pp. 51–54.

[20] Brad Karp and H. T. Kung, "GPSR: greedy perimeter stateless routing for wireless networks," in *Mobile Computing and Networking*, 2000, pp. 243–254.

[21] J. Liebeherr and M. Nahas, "Application-layer multicast with delaunay triangulations," in *Proceedings of Globecom 2001*, San Antonio, Texas, USA, 2001.

[22] FreeCiv, "http://www.freeciv.org," .

[23] Ben Leong, "Amorpheus: A proposal for a self-organizing peer-to-peer overlay network," *Unpublished manuscript*, 2002.

[24] Gradient Routing Simulator, "http://clementi.mit.edu/GRsim.html," December 2002.

[25] Alberto Medina, Anukool Lakhina, Ibrahim Matta, and John Byers, "Brite: Universal topology generation from a user's perspective. (user manual)," Tech. Rep. BU-CS-TR-2001-003, Boston University, April 2001.

[26] Bernard M. Waxman, "Routing of multipoint connections," *IEEE Journal on Selected Areas in Cummunications*, vol. 6, no. 9, pp. 1617–1622, 1988.

[27] Tian Bu and Don Towsley, "On distinguishing between internet power law topology generators," in *Proceedings of IEEE INFOCOM'02*, June 2002.

[28] K. Calvert, M. Doar, and E. Zegura, "Modeling internet topology," *IEEE Communications Magazine*, June 1997.