Network Overlay Construction under Limited End-to-End Addressability

Wenjie Wang
Department of EECS
University of Michigan
wenjiew@eecs.umich.edu

Cheng Jin
Computer Science Department
California Institute of Technology
chengjin@cs.caltech.edu

Sugih Jamin
Department of EECS
University of Michigan
jamin@eecs.umich.edu

Abstract—**Network overlay construction has found many applications in today's Internet, such as P2P networks [1][2], end-host multicast [3][4], and network security [5]. Many researchers have proposed solutions to building an overlay network among a given group of end-hosts. The usual overlay construction assumes two-way network communication—each host can initiate connections to and accept requests from other hosts. This is not always true on the Internet due to the use of Network Address Translation (NAT) and firewalls. Our experiments with one P2P file-sharing system revealed that over 34% of hosts were guarded hosts, i.e., hosts that cannot accept connections from other Internet hosts. The lack of two-way communication capability presents a challenge because only a subset of hosts can act as overlay routers. Using a cluster-base approach, we design a new overlay construction mechanism called *e**, which organizes members based on reachability and intelligently select overlay routers to reduce end-to-end latencies on the overlay. Under realistic scenarios involving guarded hosts, *e** can reduce average end-to-end latency on the overlay by 28-61% compared to existing protocols. Furthermore, *e** significantly improves the worst case end-to-end latency on the overlay.**

I. Introduction

Network overlay has become a popular technique used for communications in distributed network services, such as Peer-to-Peer (P2P) file-sharing [1][2], end-host multicast [3][4] and security overlay network [5]. Independent of the overlay construction protocol, the constructed overlay networks are generally similar. The networks consist of peers, normally end hosts on the Internet, and virtual links connecting host pairs using either TCP or UDP. The virtual links are used to relay both data and control messages. There is a rich literature on the design, implementation, and evaluation of various protocols for overlay construction, mostly in the context of end-host multicast. Examples include HMTP [6], Narada [3], NICE [7], TMesh [4], Yoid [8], and Zigzag [9]. End-host multicast, which connects members of a multicast group via transport-layer overlay, has gained popularity since IP Multicast [10] is not generally available on the Internet. These efforts focus on the protocol design to construct and optimize the overlay with all hosts considered identical in terms of processing power, bandwidth, and reachability. The Internet, however, is far more heterogeneous. With a wide variety of hardware and software configurations, constructing network overlays on the Internet faces many challenges not generally considered in simulations. The focus of this paper is to study the effect of one such challenge—the existence of guarded hosts. We call host A a *guarded* host if no TCP connections or UDP sessions can be established to A from hosts outside A's local area network. This may happen if A is behind a NAT (Network Address Translator [11]) gateway or a firewall. Similarly, we call a host that allows incoming TCP connections as well as outgoing connections an *open host*. Since guarded hosts are not able to accept incoming connections, their existence complicates the construction of efficient overlays.

The presence of guarded hosts in P2P systems not only requires modifications to the implementation of a protocol, but also poses a more fundamental challenge to the protocol design itself. The fundamental assumption of two-way communication being universally available is violated. In order to confirm that guarded hosts are indeed a tangible issue for today's P2P applications, we conducted empirical measurements on the Internet with real P2P file-sharing services. Our study indicates that 34% to 42% of hosts on P2P file-sharing networks are guarded. Our experiments also show that with such high percentages of guarded hosts, existing protocols suffer up to 37% degradation in average end-to-end latency on an overlay. These are strong evidence that guarded hosts indeed effect overlay performance. We propose a protocol named *e**, which can accommodate a large number of guarded hosts, while achieving very good performance with a relatively low overhead.

Our proposed *e** uses locality based clustering, which groups nearby member hosts into a single cluster. Each cluster consists of a *cluster center* and *clients*. Guarded hosts can only be clients, while open hosts can be clients as well as cluster centers. Each cluster must contain at least one open host to act as the cluster center. A guarded host can attach to an overlay network without introducing too much overhead by staying connected to its cluster center. Our studies on both artificial network topologies and a snapshot of the Internet topology show that compared to current overlay construction protocols, *e** achieves lower average end-to-end latency even in the presence of a large number of guarded hosts. For example, with guarded hosts making up 40% of members in a group, the average end-to-end latency with *e** is 42% lower than that of Narada. We show that *e** has low overhead and is parsimonious in its use of the underlying physical network resources.

The *e** protocol is not simply a better overlay construction algorithm. It is designed with realistic performance metrics and practical concerns in mind. Accommodating a large number of guarded hosts is only one of its main objectives. Existing protocols that use clustering may need to refine their overlay construction algorithms as they are limited to working only with open hosts. For protocols without clustering, our experiments indicate that they may experience sub-optimal performance in the presence of guarded hosts because their op-

timization techniques do not make as effective a use of the resources of open hosts as does *e**. The *e** protocol is also designed to be flexible so that it can be used in conjunction with other overlay protocols that do not explicitly minimize end-to-end latency.

The keys to *e**'s performance are its center-election protocol and its shortcut-selection heuristic. The election protocol is based on a center rank function that takes not only addressability, but also network capacity and latency into account. The shortcut-selection heuristic improves the quality of constructed overlays by intelligently adding shortcuts to improve end-to-end latencies of designated paths. The shortcut-selection heuristic can also increase the connectivity of nodes to provide reliability and better performance when necessary, even for groups with a large population of guarded hosts.

Currently, existing P2P file-sharing applications address this problem using an *ad hoc* "push" approach. A guarded host A joins a file-sharing network by connecting to a proxy-server. Search or download requests from other peers must be routed by the proxy-servers to A. Host A then pushes the results or requested files back to the actual recipients. This "push" mechanism introduces extra overhead on the proxy server and may create long access latency for those peers making the requests. Currently we are aware of only one study on the overhead introduced by guarded hosts [12], though no systematic approach on how P2P network has been proposed on how P2P network can accommodate guarded hosts with low overhead and little performance degradation.

In IP Next Layer (IPNL), Francis and Gummadi presented an extension to the current Internet architecture by introducing IPNL as a layer above IPv4 [13]. It defines IPNL header that is encapsulated in the IP header and includes extra address information (extended IP address) so that hosts behind NAT gateways can be identified and addressed. IPNL thus addresses the reachability problem for hosts behind NAT gateways by changing end hosts and NAT gateways on the Internet. However, IPNL cannot eliminate the presence of guarded hosts because of the deployment of firewalls due to security concerns. Another approach that may reduce the use of NAT is IPv6 due to its much larger address space. However, even with IPv6, firewalls are again likely to be present due to security concerns, and NAT may continue to exist because it provides the benefit of address isolation to network providers and subscribers. Our work on guarded hosts should still be relevant as the Internet evolves into a new architecture such as the one proposed by NewArch [14] or Plutarch [15]. With the increasing population of mobile IP devices, such as wireless phones, networked sensors, and power-limited handheld devices, more and more hosts have become guarded hosts and are not able or willing to serve content directly. We believe our work can be applicable in such environments similar to the case of guarded Internet hosts.

The rest of this paper is organized as follows. We first show the prevalence of guarded hosts in P2P systems and its effect on existing end-host multicast protocols in Section II. We propose our solution in Section III, evaluate its performance in Section IV, and conclude in Section VI.

II. Effect of Guarded Hosts on Network Overlay Construction

In this section, we would like to answer the question: what percentage of hosts in a typical P2P file-sharing service are guarded hosts? This study differs from existing work where the total number of hosts behind a NAT gateway is counted. Bellovin presented a technique to estimate the number of hosts behind a NAT gateway by examining the IP headers of packets from the given NAT gateway [16]. He found that when processed correctly, packets emanating from individual machines could be isolated so that the number of hosts behind a NAT gateway could be counted. What we ask here is not how many hosts there are behind any given NAT gateway but overall how many hosts in a P2P network are behind NAT gateways (and firewalls).

A. Prevalence of Guarded Hosts on the Internet

We study the prevalence of guarded hosts on the Internet by conducting empirical measurements on existing P2P networks. There are several widely used P2P file-sharing applications on the Internet, e.g., eDonkey, Gnutella, Kazaa, and Napster. We selected eDonkey and Gnutella networks because of their popularity and availability of source code.

In the eDonkey file-sharing system, a set of dedicated servers allow users to search for files. Upon startup, the new peer connects to one of the servers, which then assigns a unique ID to that peer. The server assigns two types of ID: *HighID* and *LowID*. A peer that is an open host is assigned a HighID, which is the host's IP address in decimal. A peer that is a guarded host is assigned a LowID, which is a 32-bit random number unique to that server. Each server determines whether a host should be assigned a HighID or LowID based on the result of a proprietary probing mechanism. Each eDonkey client can identify whether an assigned ID is a HighID or LowID using the threshold value of 1e6. Thus, we may use the number of HighID peers and the number of LowID peers in the eDonkey network to estimate the percentage of guarded hosts.

Since we do not have control over any eDonkey server for administrative reasons, we had to modify a popular eDonkey client [17]. Our modified client issues queries and collect lists of peers from various eDonkey servers. Specifically, our modified eDonkey client connects to a server, and asks for popular file suffixes (avi, mp3, jpg, and mpg) once every 90 seconds. From the list of files returned for each suffix, we request the top ten files with the highest availability.[1] Each request

[1] A file's availability denotes the number of peers sharing that file, as known to the connected server. Thus, selecting files with high availability yields a large number of peers.

TABLE I
NUMBER OF GUARDED HOSTS FOUND IN P2P FILE-SHARING APPLICATION

Day	1	2	3
Total Servers	52	52	52
Responsive Servers	37	32	29
HighID hosts	20200	19602	29799
LowID hosts	9832	11099	15029
% of LowID	32.7%	36.2%	33.5%

prompts the server to return a list of peers storing the requested file. The list returned by the server for the same requested file could potentially be different across requests. From these lists, it is straightforward to separate HighID and LowID peers. Our eDonkey client stays connected to each server to collect the lists of peers for 20 minutes, and then randomly tries another server.

We ran our experiments for six days with each experiment lasting about one day. Table I shows the results from the first three days. Of the 52 servers we probed, about 63% of them responded to our modified client, shown as "Responsive Servers" in the table. Some servers denied our client's connection either because the server was off-line, the server has reached its capacity, or the server was dedicated to a particular flavor of clients. Some servers noticed our unusual search activities and blacklisted our client's IP, which resulted in a smaller set of responsive servers for successive runs. In all of our experiments, we observed similar percentages of LowID (guarded) hosts, about 34% on average.

Fig. 1 shows the total number of hosts discovered on each server on Day 1 and the number of guarded hosts on that day. We sort the servers in non-decreasing order based on the percentages of LowID hosts, and assign each server a numeric rank value. As shown in Fig. 1, there are seven servers with no LowID hosts even though hundreds of hosts were found on these servers. The probability that our client has simply missed LowID hosts on these seven servers is quite low. We suspect that these servers are configured to deny guarded hosts, which is quite common for eDonkey servers. Guarded hosts impose more workload on the server than open hosts. A guarded host can only search via its TCP connection to the connected server; it also requires assistance from the connected server to upload files.

Unlike the client-server architecture on eDonkey networks, Gnutella organizes its participants into a pure peer-to-peer overlay network with no centralized servers. A peer sends or receives control messages, such as *neighbor request* or *file request*, through existing neighbors. Our study of 134,252 hosts on Gnutella network shows that 42% are guarded hosts, higher than the percentage on the eDonkey network. This difference is due to the different ways the two protocols treat guarded hosts. On the eDonkey network, in addition to the server access limitation discussed earlier, there are other evident disadvantages for guarded hosts. While an open host can request an eDonkey server to relay connection request to be initiated by

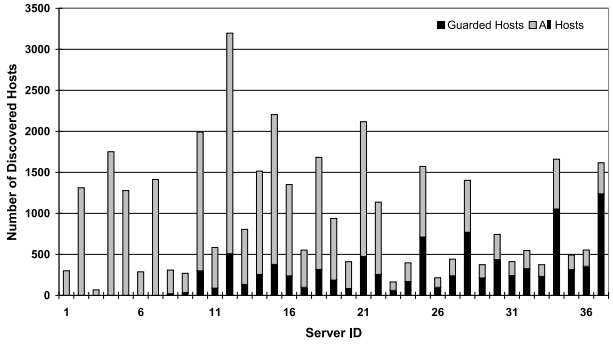


Fig. 1. Numbers of guarded hosts at different eDonkey servers.

a guarded host to the open host, a guarded host cannot make such a request because it cannot accept connections from the other guarded host. While an open host can be connected to both open and guarded hosts, a guarded host can only be connected to open hosts. Since guarded hosts can only access resources on open hosts and have limited search capability, they perceive lower quality of service (QoS) in terms of download speed compared to open hosts. There is thus strong incentives for a guarded hosts to configure its network to allow incoming connections so that it may be reachable from outside and receive the same QoS as open hosts. For P2P protocols that do not impose a difference in perceived QoS between guarded hosts and open hosts such as Gnutella, we expect a higher percentage of guarded hosts since there is no strong incentive for a host to become "open."

Our experiments indicate that on average 34% of hosts on P2P networks are guarded hosts. Due to space limitation, we only present details on the eDonkey network. The data on eDonkey network provides a lower bound on the percentage of guarded hosts one can expect to see on a general purpose P2P network because hosts on the eDonkey network have incentives to become open hosts in order to download files faster. We expect our study to be even more relevant for networks with higher percentage of guarded hosts, e.g., the Gnutella network or end-host multicast overlay networks.

B. Performance of Overlay Construction Algorithms with Guarded Hosts

We show in the previous section the prevalence of guarded hosts in P2P systems. In this section, we study how guarded hosts affect the performance of end-host multicast protocols. Before delving into the details of our evaluation setup, we briefly describe the protocols we investigate and the performance metric we use. We study four end-host multicast protocols: HMTP [6], Narada [3], TMesh [4], and Yoid [8]. HMTP and Yoid build *tree* overlays where there is a single path between any node pair,[2] while Narada and TMesh build *mesh*

[2]In our discussion, when referring to overlays, we use *node* in place of *host*, and *distance* in place of *latency* to be consistent with standard graph theory vocabulary.

overlays where multiple paths may exist between any node pair. Studying both tree-based and mesh-based overlays provides a more comprehensive picture of the effect of guarded hosts. Most of these existing protocols rely on overlay optimization algorithms to gradually improve the quality of the constructed overlays in terms of end-to-end latency. Therefore, the main performance metric we adopt in this paper is the *Average Relative Delay Penalty (ARDP)* as defined in [4]. Relative Delay Penalty (RDP) is the ratio of the latency $D'_{i,j}$ between a node pair i and j on the overlay to the latency between them on the physical network $D_{i,j}$ [3]. ARDP is then the average RDP between all node pairs:

$$ARDP = \frac{1}{N(N-1)} \sum_{i=0}^{N} \sum_{j=0, j \neq i}^{N} \frac{D'_{i,j}}{D_{i,j}}, \qquad (1)$$

where N is the number of nodes in the overlay. Smaller ARDP is an indication that most node-pair latencies on the overlay are close to latencies on the physical network. If the overlay were a full mesh, and all members were directly connected to each other, the ARDP would be 1.

We found that these existing protocols suffer about 12-32% ARDP degradation in the presence of guarded hosts, which could limit their usefulness to applications that require low end-to-end latencies. We introduce the *e** overlay construction protocol in Section III after presenting the results of our study on existing protocols in this section.

Chu *et al.* discovered that, in the real Internet, the RDP between a node pair i and j may be less than 1, that is, the distance between i and j in the overlay may be less than their distance in the network [18]. This is because the data path between two hosts is not necessarily the shortest path between them on the Internet due to various routing policies enforced by ISPs [19]. In our study, we assume that all traffic are routed through the shortest paths so the RDP between any node pair is never less than 1. The result of ignoring such effect is that our experiments likely overestimate the RDPs of various overlay protocols. On the Internet, these protocols including *e** could perform better and achieve lower RDP than reported here.

In addition to ARDP, we also use 90%-tile RDP and CDF (Cumulative Distribution Function) of RDP to give a more complete picture of the quality of end-to-end overlay latency under different construction algorithms.

B.1 Experimental Setup

To ensure that our experiments are not topology dependent, we conduct our simulations on topologies with and without power-law node degree distribution, with various sizes and configurations. For power-law based topology, we use a topology of 4,000 nodes generated by the Inet-3.0 topology generator [20]; for non-powerlaw topology, we use a transit-stub topology of 4,400 nodes generated by the GT-ITM topology generator [21].[3] We have also conducted our experiments on topologies with 6,000 and 8,000 nodes. We further evaluate our protocol on a snapshot of the Internet topology in Section IV-F. All experiments show results similar to the ones presented.

TABLE II
HOSTS WITH DIFFERENT CONNECTION SPEED

Connection speed	Maximum degree	Number of hosts
below 100Kbps	2	10%
100Kbps - 2Mbps	4	30%
2Mbps - 10Mbps	6	40%
above 10Mbps	10	20%

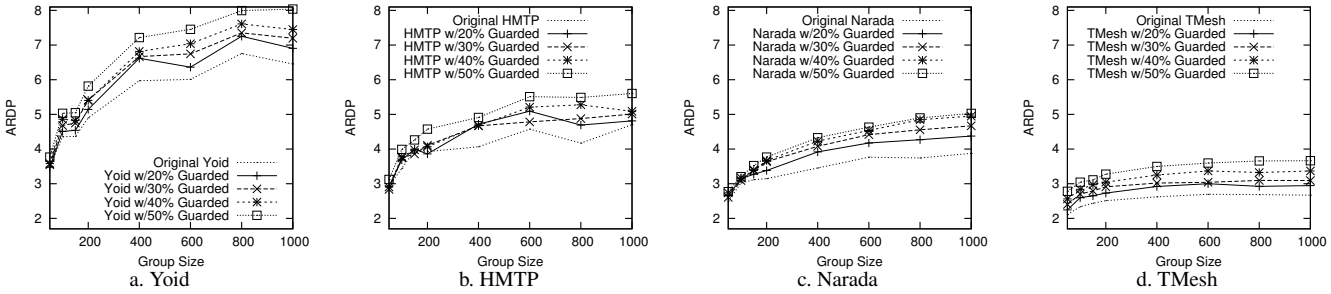
For each type of topology, we test each overlay protocol with different number of randomly picked overlay nodes. Different percentage of guarded hosts are then evenly distributed among these nodes. Following the empirical measurements presented in the previous section, we experimented with overlay networks consisting of 30% and 40% guarded hosts. In addition, to evaluate the sensitivity of our results, we further experimented with overlay networks consisting of 20% and 50% guarded hosts.

To accommodate guarded hosts, existing overlay construction protocols must be retrofitted with the constraint that *a directed link from node A to another node B can be established only if B is an open host*. In our discussion, we name the unmodified overlay construction protocols that run on topologies without guarded hosts the *original* protocols. For instance, in Fig. 2a, *original*-Yoid stands for the unmodified Yoid that runs on a topology without guarded hosts.

Besides network reachability, other factors, such as network capacity, make hosts on the Internet unequal. For instance, a host with a 56Kbps modem connection should not be treated the same as a host with a 10Mbps connection. A host's network capacity normally depends on the type of Internet connection it has. Saroiu *et al.* studied the speed of Internet connections of hosts in several P2P file-sharing networks, and found that in the Gnutella network, about 10% of hosts had connection speed less than 100Kbps, 30% had connection speed between 100Kbps and 2Mbps, 40% between 2Mbps and 10Mbps, and the remaining 20% had speed higher than 10Mbps [22]. In our experiments, we assume a similar network capacity distribution among hosts in our overlays. We also assume guarded hosts are uniformly distributed among hosts with different types of capacity.

We simulate the network capacity of a host by limiting its maximum node degree. For instance, a host with connection speed less than 100Kbps has a maximum node degree of two. A more detailed distribution of maximum node degree is shown in Table II. In addition to the distribution shown in Ta-

[3] On the Inet topology, the distance between two directly connected nodes is the Euclidean distance between the two nodes on the simulated plane. The distance between two nodes on the transit-stub topology is set automatically by the GT-ITM generator.



a. Yoid b. HMTP c. Narada d. TMesh

Fig. 2. Overlays with different numbers of guarded hosts.

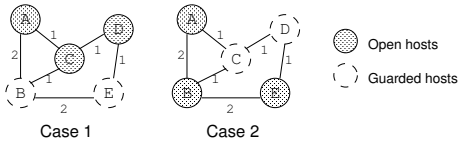


Fig. 3. Impact of guarded host placement.

ble II, we also experimented with other distributions. The distribution used may change the performance of different overlay protocols in terms of ARDP value, but it does not alter the qualitative conclusions. We only present the results for the configuration shown in Table II.

B.2 Effect of Guarded Hosts

Figs. 2a and 2b show the results for Yoid and HMTP on the GT-ITM topology respectively. The x-axis shows the multicast group size, ranging from 50 to 1,000 members. For each group size, we run the simulation ten times, selecting a different set of nodes as members and different nodes among the members as guarded hosts each time. We report the ARDP averaged over the 10 runs on the y-axis. With 30% (40%) guarded hosts, ARDP under Yoid is 12% (17%) higher than in original-Yoid. With 50% guarded hosts, the ARDP increases as much as 24%. As for HMTP, the increases in ARDP are 17%, 26%, and 32% for groups with 30%, 40%, and 50% guarded hosts, respectively.

In addition to the larger increase in ARDP, the ARDP of HMTP also fluctuates more than that of Yoid. HMTP explicitly optimizes its overlay construction, which makes it sensitive to the placement of guarded hosts. Different placements of guarded hosts may result in totally different overlay construction scenarios. Fig. 3 shows a simple example. In case 1, B and E are guarded, while in case 2, C and D are guarded. In case 2, we cannot construct a tree that performs as well as in case 1 in terms of end-to-end distance because in case 2 we have to use the longer links AB and BE. The same problem does not exist for Yoid because Yoid generally builds sub-optimal trees. Thus in terms of absolute ARDP, HMTP with 50% guarded hosts still performs better than original-Yoid. For group size of 1,000 members, HMTP with 50% guarded hosts achieves ARDP of 5.6, lower than the ARDP of 6.5 achieved by original-Yoid with no guarded host present.

In Figs. 2c and 2d, we show the performance under Narada and TMesh when guarded hosts are present. Narada and TMesh are mesh-based protocols, which generally utilize more network resources than tree-based protocols and consequently achieve better ARDP. For example, the ARDP of original-Narada is about on average 16% lower than that of original-HMTP. However, this ARDP improvement comes at a cost. Narada uses about twice as many links as HMTP to achieve the 16% latency improvement (see Fig. 4). The number of links used in an overlay is a good indication of the overlay's protocol overhead and network resource usage.

When the percentage of guarded hosts increases from 30% to 50%, Narada experiences 22% to 31% increase in ARDP compared with original-Narada (Fig. 2c). Although these percentages are comparable to that of HMTP (Fig. 2b), the nominal ARDP values are better in the Narada case because Narada uses a larger number of links than HMTP, as shown in Fig. 4. By constructing a better connected overlay, Narada increases the chance of finding a shorter path between two members. A better connected overlay is also less sensitive to the placement of guarded hosts. Hence we see that the ARDP of Narada does not fluctuate as much as that of HMTP.

Similarly, an increase in the percentage of guarded hosts on a network adversely effects TMesh (Fig. 2d). However, TMesh clearly delivers the lowest absolute ARDP values among all the protocols studied. Even with 40% guarded hosts, TMesh still out-performs original-Narada, especially for large groups. Furthermore, TMesh achieves this better performance with smaller number of links than Narada, as shown in Fig. 5. Compared to original-TMesh, the 33% increase in ARDP when 50% of the network is guarded hosts results from TMesh being restrained to using 17% fewer links than original-TMesh. Even so, these numbers compare favorably against those of the other protocols.

From Fig. 2, it is clear that TMesh provides the lowest ARDP in the presence of guarded hosts. Even with TMesh, however, when 40% of nodes on the overlay are guarded hosts, the ARDP achieved is over 3. This means that for a cross-continent node-pair with 80-100ms round-trip time (RTT), the latency between them in TMesh could be over 300ms. Studies have indicated that some applications on the Internet requires RTT of less than 300ms to be usable. Voice-over-IP and multi-

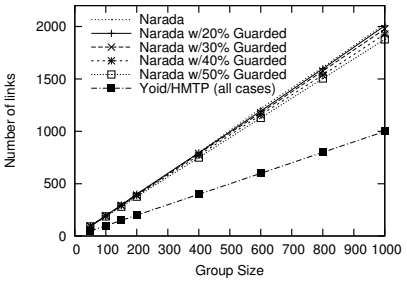


Fig. 4. Number of links used by Narada.

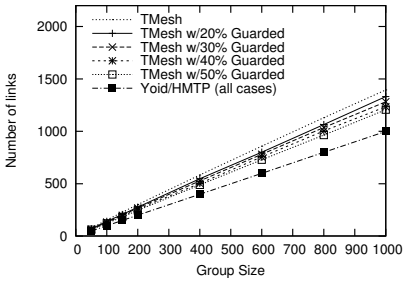


Fig. 5. Number of links used by TMesh.

player gaming are such applications [23][24]. In the remainder of this paper, we introduce and evaluate *e**, an overlay construction protocol that can achieve lower ARDP than existing protocols.

III. Guarded-Aware Overlay Construction

Our basic approach in integrating guarded hosts into an overlay is to group members into clusters based on locality, measured in latency. A center is elected in each cluster. To qualify as a cluster center, a member must meet a list of criteria, such as the speed of its network connection and its latencies to other members. By assigning an open host as the center in each cluster, the constructed overlay may accommodate a large portion of guarded hosts. In this section, we address the following two key questions: the selection of cluster centers and the connections between clusters.

Several graph-theoretic algorithms, such as k-HST [25], min K-center [26], and l-greedy [27], may be used to perform center placement. While these algorithms offer either absolute or probabilistic bounds on the latencies between nodes in a cluster and its center, there is no performance guarantee on end-to-end latencies, which is more important in an overlay. There are also algorithms that generates spanner subgraphs, such as t-spanner [28][29], which offers a geometric bound on end-to-end latency distortion, or ϵ-stretcher [30], which offers an arithmetic bound on end-to-end latency distortion.

While a geometric bound should yield spanners with fewer edges compared to those with an arithmetic bound, the multiplicative constant offers a bound that is unnecessarily restrictive for paths with very small latencies, and excessively loose for paths with long latencies. For example, in a 2-spanner, a path with real latency of 2 ms must have an overlay latency no worse than 4 ms, while a path with latency of 100 ms may be have an overlay latency as poor as 200 ms. With an additive constant of 10 ms, the same paths would be no worse than 12 and 90 ms, respectively. An additive bound is thus a more meaningful criterion to be used for overlay construction. We thus adopt the spanner algorithm of ε-stretcher in our design of *e**.

Algorithm 1 (*e**)

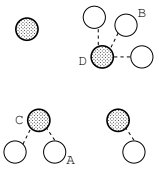
1. Let $G^* = (V^* = \emptyset, E^* = \emptyset)$
2. Let $U = \emptyset$
3. While $(V^* \neq V)$
4. Randomly select $u \in V$
5. $V^* \leftarrow V^* \cup \{u\}$
6. $\forall v \in \{v' : d_G(u, v') < \varepsilon/2\}$
7. If $(v \notin V^*)$
8. $V^* \leftarrow V^* \cup \{v\}, E^* \leftarrow E^* \cup \{(u, v)\}$
9. $U \leftarrow U \cup \{u\}$
10. Form a connected backbone among nodes in U
11. Compute the ϵ-stretcher G_ϵ of G^*

Fig. 6. The *e** algorithm.

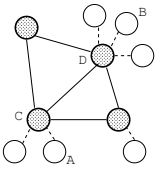
A. *Centralized* e*

The *e** algorithm consists of three phases. In the first phase, it partitions the set of hosts into clusters and selects cluster centers. This is shown in Fig. 6, lines 1-9. A node is randomly selected, and all other nodes within $\varepsilon/2$ of the selected node are grouped into one cluster. Each such cluster is a star topology (Fig. 7a). In the second phase, cluster centers inter-connect among themselves to form a connected backbone (Fig. 7b). In the current implementation, each center connects to two closest centers already on the backbone. In the final phase, the ϵ-stretcher spanner algorithm is applied to add extra links or *shortcuts* to improve the end-to-end latencies of the constructed overlay (instead of the ϵ-stretcher algorithm, another algorithm may be used for shortcut selection to optimize for different performance metric).

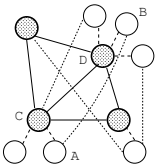
Fig. 7 illustrates the construction process of an *e** overlay. In the figure, each circle represents a host, and a line connecting two hosts represents a connection, i.e., a virtual link in the overlay network. The shaded circles in bold are the cluster centers. In Fig. 7, we only show the overlay network, with the underlying physical network concealed. We introduce the definition of latency bias δ, which is defined on a node pair. For example, for node pair AB shown in Fig. 7b, the distance between node A and B in the overlay, D'_{AB}, is the sum of distances between node pairs AC, CD, and DB. If the distance between A and B in the concealed physical network is D_{AB}, the difference between D'_{AB} and D_{AB} is the *latency bias* δ_{AB}. For a node pair that is directly connected in the overlay, its δ



a. Electing cluster centers and forming star topologies.



b. Inter-connecting cluster centers.



c. Adding extra links for performance.

Fig. 7. Example of $e*$ algorithm.

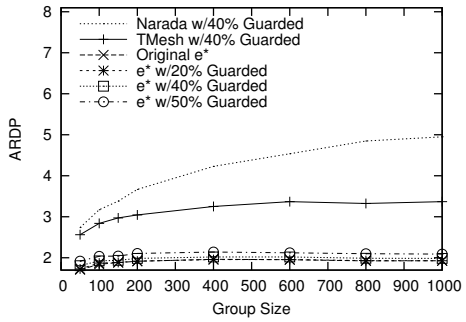


Fig. 8. $e*$ with different numbers of guarded hosts.

value is zero. A virtual link between a node pair in the group that is not directly connected in the overlay is a *potential* shortcut. For example, node pair AB is not directly connected in Fig. 7b, so the virtual link AB is a potential shortcut. If latency bias δ_{AB} is above a pre-defined boundary ε, the potential shortcut AB will be added and δ_{AB} becomes zero, as shown in Fig. 7c. All potential shortcuts are evaluated in the order of their link latencies.

To study $e*$ capability to minimize end-to-end latencies with a large percentage of guarded hosts, we first show how it can be applied to establish overlay in a centralized manner, assuming global knowledge and coordination. We then present a distributed version of the $e*$ protocol.

A.1 $e*$ Parameter Settings

There are four parameters in the $e*$ algorithm that affect the overlay performance:

- Accuracy parameter ε: ε determines the trade-off between end-to-end latency distortion and the number of links in an overlay. With highly dynamic membership in an overlay, it may be difficult to use a static ε value. A small ε on a sparse group may require a large number of links in the overlay, in the extreme case, creating a full mesh. On the other hand, a large ε on a dense group may result in a poorly connected overlay. An alternative is to dynamically adjust ε based on latency information of current overlay participants.
- Cluster radius (r): Cluster radius effects both the size of clusters and the number of clusters. Currently, we set $r = \varepsilon/2$.
- Cluster center inter-connection: We need to construct a *connected* and *efficient* overlay among the cluster centers. An efficient initial overlay is important to achieve good performance [4]. Since $e*$ algorithm does not specify the protocol to connect cluster centers, any existing overlay construction protocol may be used.
- Node degree allocation: In $e*$, each cluster center C has two types of neighbors: nodes in its own cluster and other cluster centers. Since C has limited available bandwidth in practice, it should carefully set its node degree to maintain a balance between connecting to other cluster centers for better performance and serving as many nodes as possible in its own cluster.

A.2 Performance of Centralized $e*$

To compare our results on $e*$ with the results we have on existing overlay protocols, we use the same experiment setup described in Section II-B.1. The $e*$ algorithm for this implementation assumes end-to-end latency information is known. The ε value is set to 30%-tile of end-to-end latency distribution for the group. Only open hosts are qualified as cluster centers. Each cluster center reserves at least two of its node degree for inter-connections to other cluster centers. Each cluster center connects to its closest cluster centers that have not reached their node degree limits. After cluster centers are inter-connected, as in ϵ-stretcher, each end-to-end path is evaluated in non-decreasing order of latency. If the path latency in the $e*$ overlay is longer than the actual latency by more than ε, a shortcut is added.

Fig. 8 shows the ARDP of the resulting $e*$ overlay with varying number of guarded hosts. The results of Narada and TMesh with 40% guarded host are included for comparisons. In Fig. 8, we see very small degradation in ARDP with 20% and 40% guarded hosts. Even with 50% guarded hosts, $e*$ only experiences about 7% worst-case ARDP degradation compared to the standard $e*$ algorithm with no guarded hosts. In terms of absolute ARDP, $e*$ achieves lower ARDP than existing end-host multicast protocols we have studied. Where TMesh achieves an ARDP of 3, $e*$ is able to deliver an ARDP of 2. A node-pair with 80 ms RTT experiences only around 160 ms RTT on the $e*$ overlay.

However, $e*$ uses nearly 50% more links than Narada, nearly twice as many as TMesh. A detailed study of our simulation results shows that the final one-third of the links added by $e*$ to the overlay contributes only 7% improvement to ARDP. Adding more shortcuts to an $e*$ overlay clearly has a diminishing return in ARDP improvement. To reduce the

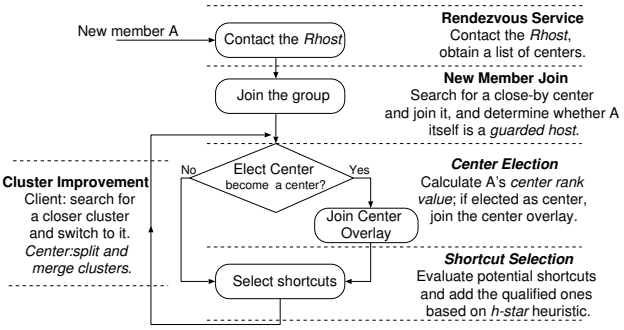


Fig. 9. Flow chart of $e*$ protocol. Center election, shortcut selection, and cluster improvement may be done in parallel.

number of links with small contributions, we modify $e*$ to stop adding links when $e*$ uses the same number of links as Narada for a given multicast scenario. The resulting algorithm achieves practically identical results as the unmodified $e*$. This observation that shortcuts improve the ARDP of $e*$ overlay with diminishing return leads us to the shortcut selection heuristic used in the distributed version of the $e*$ algorithm. Obviously, the distributed version of $e*$ requires a stopping condition independent of Narada, as described later.

B. *Distributed* e* *Protocol*

We demonstrated centralized $e*$'s potential to accommodate a large number of guarded hosts and its ability to achieve low end-to-end latencies. In this section, we present a three-phase distributed *$e*$ protocol* with each phase corresponding to the $e*$ algorithm shown in the previous section. First, a center-election protocol is applied based on a *center rank vector*. After centers are elected, they are inter-connected by an overlay protocol. Any existing overlay protocol such as HMTP, Narada, or TMesh will do. The links that inter-connects cluster centers form an overlay called *center overlay*. In the third and final phase, links are evaluated and added to the overlay by each individual node, based on a shortcut selection heuristic, to improve the quality of the final overlay.

The $e*$ protocol requires only limited local coordination and no global knowledge. The center-election protocol involves coordination among a limited number of members in the same cluster, the overlay optimization heuristic only utilizes locally measurable metrics as described later.

The $e*$ protocol consists of the following components: rendezvous service, joining procedure, distributed center-election protocol, cluster improvement, and shortcut selection. Fig. 9 shows a flow chart of the $e*$ protocol by illustrating the procedures that a new member A goes through in joining an $e*$ overlay. Host A first contacts the rendezvous service and then follows the joining procedure to join a cluster. A cluster center is elected by the center-election protocol described later. A also adds several shortcuts to other members in the group. Since members in the group comes and goes, clusters may be split to accommodate more members or be merged to reduce the number of clusters. The center election, shortcut selection, and cluster improvement tasks may be done in parallel. To detect member failure, cluster centers exchange "heartbeat" messages. Members of the same cluster also exchange "heartbeat" messages with each other.

Rendezvous Service and ε Value

We assume the presence of a well known *rendezvous host* ($Rhost$) [8], whose main purpose is to bootstrap newly joining members into the multicast group. The $Rhost$ usually serves as a directory server that replies to members' queries with a list of members already in the multicast group. In our case, the $Rhost$ returns a list of cluster centers so that the querying member can start the join process.

The $e*$ protocol uses only one layer of clusters. This simple structure reduces the protocol complexity, but for large groups, the number of cluster could be $O(N)$ in the worst case, too large for efficient and fast joining. To speed up the joining process, cluster centers can be connected by existing protocols with hierarchical properties, for example, HMTP or TMesh, whose joining complexity is $O(\log N)$.

The ε value plays an important role in the $e*$ algorithm. As we will show in Section IV-A, the performance of $e*$ is robust to varying ε values, so the ε value can be set either by the $Rhost$ or by a nominated cluster center based on measurements from previous runs of the algorithm or from current members.

New Member Joins and Guarded Host Detection

When a member A joins a multicast group, it first obtains a list of cluster centers from $Rhost$. The list includes the root of the center tree if HMTP or TMesh is used. Member A finds the closest cluster that has not reached full capacity and is within the distance of cluster radius r, and joins it. If no such cluster exists, A either randomly finds a cluster to join or forms a cluster of its own.

Another important task of member A is to figure out its addressability, i.e., whether it is a guarded host. In the query message A sends to other centers during the joining process, a "callback" bit is set so that the queried centers will attempt to connect back to A to check whether A is addressable. If any of these "callback" succeeds, the "callback" bit will be canceled in latter queries and member A considers itself an open host. If a certain number of "callback" fails, A will consider itself a guarded host.[4]

Center Election Protocol

Center election in each cluster is based on a *center rank vector* (CRV). CRV is defined by each multicast group. Each element in the CRV is a criteria for center election. A typical CRV looks like this: $CRV =<$ *open, capacity, lifetime, cluster_dist* $>$, with its schema explained in Table III.

[4]In such a scenario, false-positive is possible, but it does not compromise the $e*$ protocol. A guarded host is also allowed to convert to an open host if port-mapping [11] is configured or firewall is disabled after the host joins the group.

TABLE III
SCHEMA OF A TYPICAL CENTER RANK VECTOR

Element	Description
open	0 if the host is guarded, 1 otherwise
capacity	maximum node degree
lifetime	how long the host has stayed in the group
cluster_dist	sum of distances to other members in the cluster

Each member in a multicast group is responsible for maintaining the value of each element in its CRV. A member periodically updates its CRV by active and passive probing. For *cluster_dist*, since members in the same cluster periodically exchange "heartbeat" messages with their CRV included, the distance to other members can be measured and summed up without too much overhead. The most challenging part in computing the CRV is to determine *capacity* because *capacity* depends on the members' available bandwidth. We should be careful in selecting cluster centers to avoid those with very low bandwidth. As discussed in Section II-B.1, Internet hosts today have a known spectrum of network connections, from dial-up modem to dedicated multi-Mbps connections. Thus, instead of measuring bandwidth, which normally involves high overhead [31][32][33], each *e** client tries to estimate its connection type. To accomplish this it relies on system specification[5] and user specification. Additionally, it maintains a history of the maximum throughput of the most recent downloads and use this as an indication of its effective connection speed. In addition to connection type, the computation of capacity can also take other factors, such as estimated data rate in the group, CPU power, system load, etc, into account.

Since CRVs are part of the "heartbeat" messages exchanged among hosts in the same cluster, hosts in the same cluster can be sorted by comparing each element in their CRV vectors in order. When sorting, some elements can tolerate certain deviation. For example, two hosts with *lifetimes* differed only by minutes are considered equal. Hosts' IPs are used to break ties. The host with the highest ranked CRV is elected as the cluster center.

To qualify as a center, a host must meet some minimal requirements. Clearly, only *open* hosts can be centers. To be a cluster center, a host must at least have capacity above a threshold value.[6] We call a host that qualifies as a center but is not currently serving in that role a *backup center*. A cluster may refuse a join or split message if it does not have enough backup centers. We next discuss cluster splitting.

Cluster Improvement

Once a host A is attached to a cluster C, it begins to receive updates from members in cluster C. These updates include the information on cluster C's connectivity to other clusters (foreign clusters). It also includes a list of the foreign cluster centers. Using this list, host A searches for a closer cluster by randomly probing the foreign cluster centers. To avoid oscillation, host A switches to a new cluster C' only if C' is closer than the current cluster C by a threshold value.

Cluster centers can be re-elected due to dynamics of cluster membership or changes in CRV. If a backup center with higher ranked CRV appears, it will become the new center. The new center takes over the old center's connections in the center overlay.

Dynamic membership of a multicast group and the cluster switching mechanism can change the size of an existing cluster. Clusters can be split to accommodate newly joined members or switching members. A cluster center allows the size of its cluster to exceed the maximum number of clients it can serve only if it can potentially be split, i.e., the cluster has more than one backup centers. Cluster merge is possible if a cluster center finds a close-by cluster that is small enough such that clients from both clusters can be hosted by a center and the center still has enough free-capacity remaining to prevent immediate splitting of the merged clusters.

To split a cluster, the cluster center C notifies all its cluster members of the upcoming split. It also notifies them of the new center D. D will be selected to be the farthest backup center from C. Upon receiving the notification, cluster clients that are closer to the new center switch to D. D, on the other hand, marks its connection to C as a link in center overlay, and starts to join the center overlay according to the protocol used (Narada, TMesh, etc.). Cluster merge operation is a reverse of the split operation. The center (D, for example) with the relatively lower ranked CRV marks its connection to C as a client-center link and detaches itself from center overlay. Its also tells its clients to switch to C.

Shortcut Selection Heuristic

The final phase of the *e** protocol is to add shortcuts to reduce end-to-end latencies. To add shortcuts in order of link length will incur too high an overhead as all possible links must be evaluated. Instead, we use a shortcut selection heuristic to select shortcuts shorter than a latency bound Λ but with latency bias higher than the accuracy parameter ε. Selecting links with latency shorter than Λ complies with *e** algorithm's shortest-link first policy. For instance, for a node pair AB, if A is the node that initiates the link selection, A calculates the latency bias δ_{AB}. Only if $\delta_{AB} > \varepsilon \; and \; D_{AB} < \Lambda$ will the shortcut AB be added. D_{AB} is the latency between AB in the underlying physical network.

Calculating δ_{AB} only requires locally measurable metrics. The calculation requires both D_{AB} and D'_{AB}, where D'_{AB} is the latency between AB on the overlay. Since A is the initiating node, B must be an open host, otherwise the link addition should not be considered in the first place. With B being an open host, latency on the physical network from A to B can be obtained using any standard measurement tool. D'_{AB} can

[5]Sometimes a host's full DNS name indicates its connection type, which may contain keywords such as "dialup" or "dsl".

[6]Under extreme circumstances where there are no sufficient high-capacity nodes, low-capacity nodes become centers in order of their CRV ranks.

be obtained from the A routing table if distance vector or path vector routing is used on the overlay. Otherwise, D'_{AB} can be estimated by the latency between the cluster centers of A and B since the latency between A (B) to its cluster center is bounded by cluster radius r.

The shortcut selection heuristic also allows node with low node degree to increase the Λ value if it has extra capacity. If a node can not find any qualified links after a period of time, it can double the Λ value to increase its chances of finding shortcuts.

IV. Performance Evaluation

Accuracy parameter ε effects the *e** protocol in that it controls both the cluster radius r and the shortcut selection heuristic. We first study the sensitivity of *e** on the choice of ε. Then we investigate the quality of *e** overlay with guarded hosts constituting different percentages of the overlay population. We use HMTP, Narada, and TMesh protocols to connect the center overlays. In addition to ARDP, we also evaluate *e** along the following metrics: 90%-tile RDP, node degree distribution, protocol overhead, and network resource usage. We use the same simulation scenarios used throughout this paper. Due to space limit, we only present here the results from GT-ITM topology and from a snapshot of the Internet topology.

A. Effect of ε Value

The accuracy parameter ε serves two purposes in the *e** protocol: to determine the radius r of clusters ($r = \varepsilon/2$) and the lower bound for latency bias during link selection. In the centralized *e** algorithm, ε is set to the 30%-tile of end-to-end latency distribution. It is also desirable to calculate ε based on end-to-end latencies in our distributed protocol because intuitively it should adapt according to the group dynamics. However, as end-to-end latencies fluctuate with dynamic group membership, it is infeasible to obtain this information on the Internet in real time. An alternative is to estimate ε based on partial or historic knowledge of end-to-end latencies. Since we are likely to use inaccurate ε values for overlay construction, we need to evaluate how robust the *e** protocol is to the inaccuracy in the ε value. For this purpose, we define three types of ε values: accurate-ε, partial-ε, and historical-ε. Accurate-ε is calculated assuming complete knowledge of end-to-end latencies, partial-ε is based on the latency information collected from a subset of current overlay members, and historical-ε is ε value measured in previous runs of the *e** algorithm.

To evaluate the use of partial-ε and historical-ε in the *e** algorithm, we conducted the following experiments. We measure the accurate-ε from a group of 50 randomly chosen nodes. Let us call this group G_1 and the measured ε value $\hat{\varepsilon}$. To evaluate the feasibility of using partial-ε, we grow the group from 50 to 1,000 members, and measure the performance of *e** for various group sizes, as reported in Fig. 10, using the same $\hat{\varepsilon}$ as the ε value in all cases. To evaluate the feasibility of using historical-ε, we randomly choose another group G_2 of 50 nodes distinct from those in G_1. We then measure the performance of *e** on this group, as the group grows from 50 to 1,000 members, again using the same $\hat{\varepsilon}$ as the ε value in all cases. Fig. 10 shows the resulting ARDP. For comparison purposes, we also show the ARDP of *e** when accurate-ε values are used in all cases. The use of partial-ε and historical-ε do not give rise to a significant variation in ARDP.

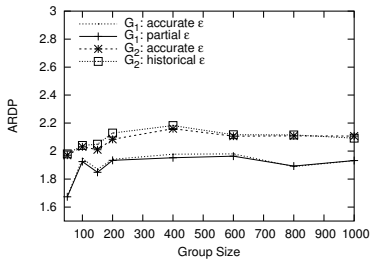


Fig. 10. Effect of partial-ε and historical-ε on *e**.

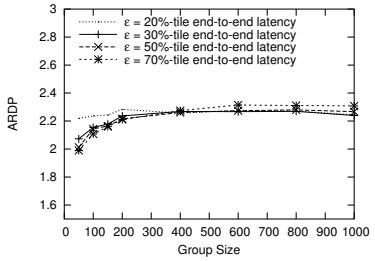


Fig. 11. The *e** with different ε values.

We next study the robustness of *e** with a range of ε values by specifying that each member determines its ε value as a fixed percentile of end-to-end latency distribution measured to all other members. Fig. 11 shows the performance of *e** for ε determined based on various percentiles of the measured end-to-end latency distribution.

Our investigation shows that *e**'s robustness to changes in ε value rests on the shortcut selection heuristic. Changes in ε can result in changes in cluster radius and number of clusters. However, the shortcut selection heuristic can still effectively select most of the short links as desired in the *e** algorithm. These shortcuts help to achieve the consistently good end-to-end latency in the *e** overlay.

B. ARDP and 90%-tile RDP

In this section, we show the performance of *e** overlays in terms of end-to-end latency when the center overlays are connected using HMTP, Narada, or TMesh protocol. Our results show that *e** can significantly improve end-to-end latencies of overlay networks. Even with 50% guarded hosts, it can still achieve lower ARDP than the original-TMesh, where there were no guarded hosts.

Figs. 12 to 14 show the ARDP of *e** overlay with center overlay constructed using HMTP, Narada, and TMesh respectively. We name the *e** overlay with centers connected by

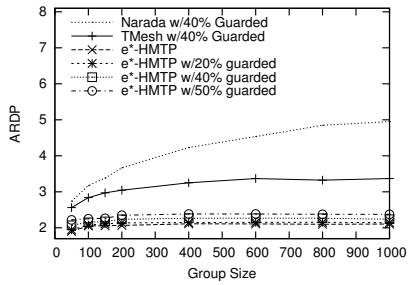


Fig. 12. ARDP of $e*$ overlay with HMTP center overlay.

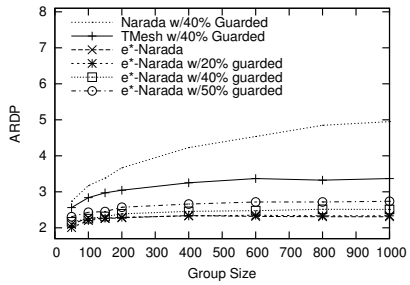


Fig. 13. ARDP of $e*$ overlay with Narada center overlay.

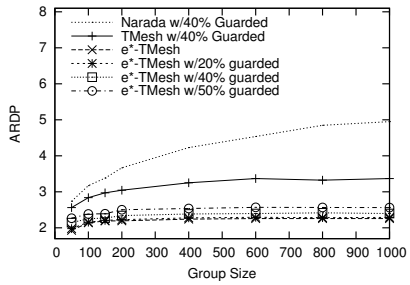


Fig. 14. ARDP of $e*$ overlay with TMesh center overlay.

HMTP $e*$-HMTP overlay, similarly, $e*$-Narada and $e*$-TMesh are for center overlays connected using Narada and TMesh. Compared to the ARDP degradation of all the protocols shown in Section II-B.2, it is clear that the use of the $e*$ protocol results in lower ARDP degradation. With 20% guarded hosts, we hardly see any change in ARDP in all three $e*$ cases. Even with 50% guarded hosts, $e*$-HMTP and e*-TMesh experience only about 12% and 6% degradation in ARDP. The distributed $e*$ protocol proves its ability to accommodate a large population of guarded hosts.

Use of $e*$ also improves ARDP in absolute terms. For groups with 40% guarded hosts, the ARDP of $e*$-HMTP is 61%, 42%, and 28% lower than that of Yoid, Narada, and TMesh without $e*$ respectively. The significant advantage of the $e*$ protocol in the performance can also be observed from the 90%-tile RDP (Fig. 15). For example, with 40% of members being guarded hosts, the 90%-tile RDP of $e*$-HMTP is 46.6% lower than that of Narada, and 33.2% lower than that of TMesh.

We notice that $e*$ in general uses more links than original Narada and TMesh. For example, $e*$-HMTP with 40% guarded host uses about 6.3% more links than Narada. However, considering that it achieves 41.6% lower ARDP, the extra number of links is reasonable. Another observation is that despite the performance difference between original HMTP and TMesh, they perform equally well in conjunction with the $e*$ protocol. The shortcuts selected by $e*$'s shortcut selection heuristic equalize the differences between $e*$-HMTP and $e*$-TMesh since TMesh and $e*$ use similar heuristics to pick shortcuts.

To provide a more complete picture of the performance advantages afforded by $e*$ in the presence of guarded hosts, we also study the CDF of RDP of various overlays. In Fig. 16 we show the results for 600-node groups with 40% of members being guarded hosts.

In the above experiments, $Rhost$ sets the ε value to 30%-tile end-to-end latency based on latency information obtained from each node. Our CRV consists of $<$*open, capacity, cluster_dist*$>$ only. The maximum node degree distribution is the same as in Table II. Nodes with maximum degree above or equal to four are qualified as cluster centers. When a node becomes a cluster center, half of its node degree is allocated for cluster clients, and the other half is reserved for inter-cluster connections and shortcuts. For shortcut selection, we set a latency bound Λ to be half of ε. A node with node degree lower than four doubles its Λ if it fails to find a qualified shortcut after five tries.

C. Node Degree and Center Distribution

Node degree distribution indicates the workload distribution of a group. Nodes with higher node degrees generally contribute resources to the group. Fig. 17 shows the node degree distribution of 600-node groups for different overlays when 40% of members are guarded hosts. In all cases, the maximum node degree is ten. Apparently, $e*$ produces more nodes with high degrees. Our investigation shows that most of the high degree nodes are centers in the $e*$ overlay. This is because the $e*$ protocol deliberately increases cluster centers' connectivities. This property can be beneficial to content providers, who can place well-provisioned, high-capacity nodes on the network. These nodes can serve as cluster centers and facilitate a more efficient overlay.

There are several plateaus in the CDF that only happen in $e*$ overlays, such as the one from degree seven to degree nine. The plateau between degree seven and nine means that there are very few nodes with these degrees. The edges of the plateaus match the maximum node degrees in our experimental setup. For example, as shown in Table II, we have maximum node degrees of six and ten, which happens to be the edges of the plateau between seven and nine. Since $e*$ prefers to place centers on high-capacity nodes, most nodes with the maximum node degree (ten) are elected as centers and reach their maximum capacities, which explains why we hardly see nodes with degree from seven to nine.

The plateaus in node degree distribution can also be explained by studying the cluster size. In Table IV, we show the distribution of cluster size for $e*$-HMTP on a 600-node group. We count the number of cluster of different sizes, and report the ratio over the total number of cluster in Table IV. With increased number of guarded hosts, the ratio of large-

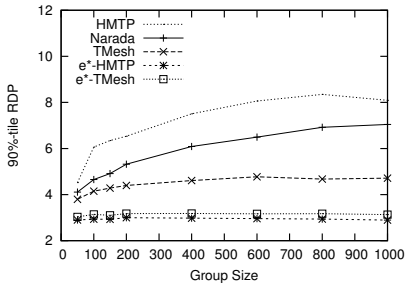


Fig. 15. 90%-tile RDP of various overlays with 40% guarded hosts.

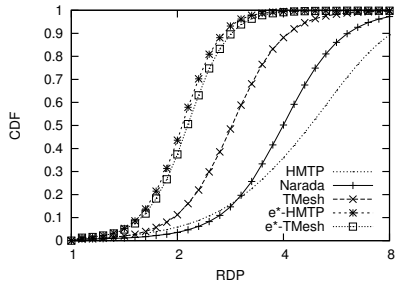


Fig. 16. CDF of RDP of various overlays with 40% guarded hosts.

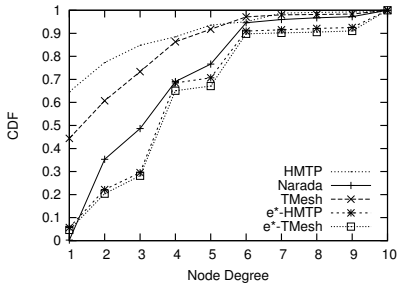


Fig. 17. Node degree distribution of various overlays with 40% guarded hosts.

TABLE IV
PERCENTAGE OF CLUSTER WITH DIFFERENT SIZES

Percentage of guarded host	Cluster Size				
	1	2	3	4	5
0%	53.2%	20.1%	18.2%	3.5%	5.0%
20%	42.8%	19.7%	26.5%	3.0%	8.0%
40%	27.5%	15.6%	40.3%	3.0%	13.5%
50%	16.1%	11.8%	52.1%	2.4%	17.6%

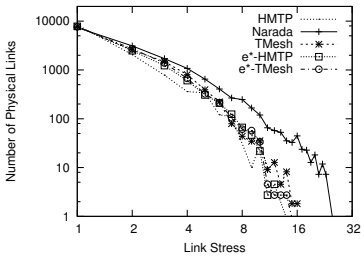


Fig. 18. Link stress.

size clusters increases. This is because the increasing number of guarded hosts have to be served by some cluster centers, and correspondingly result in more large clusters. One exception is cluster size four, whose share keeps decreasing. The reason is similar to what causes the plateau areas in node degree distribution. For nodes with maximum degrees of six and ten, their maximum cluster capacities are set to three and five respectively in our experiments. Decreased number of clusters of size four means that most of degree-ten nodes reach their full cluster capacity of five, as indicated in Fig. 17.

D. Link Stress

In overlay networks, a physical link may have to carry multiple copies of the same data sent by a member. This normally happens if a node has a higher degree on the overlay than its number of physical connections. The number of duplicate copies of the same packet carried by a link L is usually called its *link stress*. For example, if L carries only one copy of data, it has a link stress of one. Fig. 18 shows the distribution of link stress of various overlay from ten simulations of 600-node group with 40% of members being guarded hosts. In each simulation, we pick one random node to serve as a data source and count the copies of data carried by each physical link used in the overlay. A routing algorithm is implemented on the overlay so that data is forwarded through source-rooted shortest paths on the overlay. Fig. 18 shows that *e** has smaller physical number of links with link stress above ten than does TMesh. Narada shows more physical links of high link stress than the other protocols.

E. Protocol Overhead

In the *e** protocol, only one layer of cluster centers are maintained. In contrast, NICE and Zigzag use a hierarchy of clusters. Except for worst-case node degree, Zigzag has similar overhead to NICE, including a worst-case join overhead of $O(\log N))$ and a worst-case control overhead of $O(\log N)$ [9]. N is the size of the multicast group. Zigzag has a worst-case node degree of $O(d^2)$, where d is a constant (Zigzag sets $d = 3$), whereas NICE has a worst-case node degree of $O(\log N)$. In *e**, the worst-case node degree is m, where m is a constant. In our experiments we set $m = 10$, however, *e** has no special requirement on maximum node degree. Each node can specify its degree based on its own connection condition. In Zigzag all nodes are considered equal. The join overhead of *e** is the join overhead of the center overlay protocol used. Particularly, for *e**-HMTP and *e**-TMesh, the average join overhead is $O(\log N)$. For control overhead, since each node has a constant number of node degree, and periodically exchanges neighborhood status with its neighbors, the control overhead of *e** is $O(m^2)$. The cluster merge and split overhead of *e** is $O(m)$. How to build NICE and Zigzag overlay in the presence of guarded hosts and with the constraint of limited node degree, while still preserving their original properties and integrities, is currently unspecified.

We now compare *e**'s overhead against those of Narada and TMesh. We only consider two types of protocol overhead: routing overhead and overlay maintenance overhead. We assume all the protocols adopt a path vector algorithm similar to the one used in TMesh [4]. In *e**, only cluster centers

TABLE V
PROTOCOL OVERHEAD (KBPS)

Group size	50	100	150	200	400	600	800	1000
Narada	0.251	0.577	0.976	1.379	3.155	5.026	7.064	9.194
TMesh	0.208	0.489	0.780	1.111	2.659	4.336	6.114	7.985
*e**-HMTP	0.204	0.425	0.684	0.938	2.139	3.459	4.818	6.283
*e**-TMesh	0.220	0.486	0.755	1.012	2.299	3.637	4.994	6.490

exchange routing updates among themselves. Cluster clients only receive updates from their centers. We also take all the "heartbeat" messages into account. In Table V, we report the protocol overhead of various overlays on groups where 40% of members are guarded hosts. The $e*$ protocols show lower overhead than Narada or TMesh alone. Even though $e*$ incurs extra overhead to maintain its cluster structure, this is offset by the reduction in routing overhead due to relegating routing responsibility to cluster centers.

F. Experiments with Internet Topology

In addition to artificial topologies generated by Inet and GT-ITM generators, we also run our experiments on a router-level topology of a large ISP constructed from traceroute data. The traceroutes were initiated from 50 sites on the Internet to 200,000 IP addresses. The resulting topology contains 1,426 nodes. We conduct ten experiments for each overlay scheme investigated in this paper, with the same settings for guarded hosts and maximum node degree distribution as used throughout this paper. We randomly choose 200 nodes to be members of a multicast group. For groups with no guarded hosts, the ARDPs are: 1.83 (Narada), 1.75 (HMTP), 1.49 (*e**-TMesh), 1.48 (TMesh), and 1.43 (*e**-HMTP). For groups with 40% guarded hosts, the ARDPs are: 2.08 (Narada), 1.80 (HMTP), 1.72 (TMesh), 1.56 (*e**-TMesh), and 1.50 (*e**-HMTP). In terms of 90%-tile RDP, for groups without guarded hosts, the results are: 3.20(Narada), 3.11 (HMTP), 2.27 (TMesh), 2.15 (*e**-TMesh), and 2.05 (*e**-HMTP); for groups with 40% guarded hosts, 3.88 (Narada), 3.20 (HMTP), 2.84 (TMesh), 2.34 (*e**-TMesh), and 2.24 (*e**-HMTP). The $e*$ protocol performs consistently well.

V. RELATED WORK

Several existing overlay construction protocols also employ clustering technique, e.g., NICE and Zigzag. NICE organizes members of a multicast group into a hierarchy of clusters with size $[k, 3k-1]$, where k is a constant. Each member must belong to a level 0 cluster. Each cluster nominates a cluster leader that will become a member in layer 1. Members in layer 1 are grouped into bounded-size layer-1 clusters in the same way as in layer 0. This procedure repeats recursively until it reaches the top layer where there is only one member. The overlay is formed by connecting all members in the same cluster into a full mesh. For a member that appear from layer 0 to layer l, its maximum node degree may reach $l*(3k-1)$. With worst-case node degree $O(\log N)$, NICE may not scale well for applications with high date rate. Furthermore, since NICE only considers latency for cluster leader selection, it may experience sub-optimal performance if selected cluster leaders are hosts with inadequate bandwidth.

Zigzag organizes its cluster hierarchy in a similar manner to NICE, but adopts a different overlay construction mechanism. Zigzag forms the overlay by two kinds of links: links from server S to other members in the top layer, and links from a node in layer l to its foreign subordinates. Foreign subordinates of a layer-l node (A) are layer-$(l-1)$ nodes that are not in A's cluster in layer $l-1$. Zigzag assumes a single server S that is the data source for the group; server S is the cluster leader for all the clusters of which it is resident. Zigzag shares the same cluster leader selection problem as NICE. Since Zigzag assumes a predefined single data source S, and consequently builds a tree overlay rooted at S, it may suffer high average end-to-end latency, especially for nodes that appears only in layer 0, which make it not suitable for groups with multiple data sources.

VI. CONCLUSION

In this paper we present a systematic study of network overlay construction under limited end-to-end addressability. We first show the prevalence of guarded host in popular P2P file-sharing applications. Our study indicates that as high as 42% of the participants in P2P file-sharing applications can be guarded hosts. The large population of guarded hosts not only imposes a challenge on overlay protocol deployment but also on the overlay protocol design itself. We propose a distributed overlay construction protocol named $e*$ than can accommodate a large population of guarded hosts. Our evaluations prove that $e*$ can accommodate a large number of guarded hosts, and achieve low end-to-end latency with low overhead. At the same time, $e*$ has the flexibility to work with any overlay construction protocols, which leaves space for further performance improvement if used with more advanced protocols. Since we design $e*$ with practical issues in mind, such as guarded host detection and realistic node degree limit, it can provide a realistic and effective overlay construction protocol for the Internet.

REFERENCES

[1] Kazaa. http://www.kazaa.com/.
[2] GnuTella. http://www.gnutella.com/.
[3] Y. Chu, S. Rao, and H. Zhang. A Case For End System Multicast. *ACM Sigmetrics*, pages 1–12, 2000.
[4] W. Wang, D. Helder, S. Jamin, and L. Zhang. Overlay Optimizations for End-host Multicast. *Networked Group Communications*, 2002.

[5] A. Keromytis, V. Misra, and D. Rubenstein. SOS: Secure Overlay Services. In *Proc. of ACM SIGCOMM 02*, 2002.

[6] B. Zhang, S. Jamin, and L. Zhang. Host Multicast: A Framework Delivering Multicast To End Users. *Proc. of IEEE INFOCOM '02*, 2002.

[7] S. Banerjee, B. Bhattacharjee, and C. Kommareddy. Scalable Application Layer Multicast. In *Proc. of ACM SIGCOMM '02*, 2002.

[8] P. Francis. Yoid: Extending the Internet Multicast Architecture. *Unrefereed report*, Apr. 2000. http://www.aciri.org/yoid.

[9] D. Tran, K. Hua, and T. Do. Zigzag: An Efficient Peer-to-Peer Scheme for Media Streaming. *Proc. of IEEE INFOCOM 03*, 2003.

[10] S. Deering. Multicast Routing in Internetworks and Extended LANs. In *Proc. of ACM SIGCOMM '88*, pages 55–64, 1988.

[11] K. Egevang and P. Francis. The IP Network Address Translator (NAT), 1994. RFC-1631.

[12] Y. Chu, A. Ganjan, T. Ng, S. Rao, K. Sripanidkulchai, J. Zhan, and H. Zhang. Early experience with an internet broadcast system based on overlay multicast. Technical Report CMU-CS-03-214, Carnegie Mellon University, 2003.

[13] P. Francis and R. Gummadi. IPNL: A NAT-Extened Internet Architecture. *Proc. of ACM SIGCOMM '01*, 2001.

[14] D. Clark, R. Braden, A. Falk, and V. Pingali. FARA: Reorganizing the Addressing Architecture. *ACM SIGCOMM 2003 FDNA Workshop*, 8 2003.

[15] J. Crowcroft, S. Hand, R. Mortier, T. Roscoe, and A. Warfield. Plutarch: An Argument for Network Pluralism. *ACM SIGCOMM 2003 FDNA Workshop*, 8 2003.

[16] S. Bellovin. A Technique for Counting NATed Hosts. In *Internet Measurement Workshop*, 2002.

[17] eMule Client. http://www.emule-project.net/.

[18] Y. Chu, S. Rao, S. Seshan, and H. Zhang. Enabling Conferencing Applications on the Internet using an Overlay Multicast Architecture. *Proc. of ACM SIGCOMM '01*, 2001.

[19] L. Gao and J. Rexford. Stable Internet Routing Without Global Coordination. In *Proc. of ACM SIGMETRICS*, 2000.

[20] J. Winick and S. Jamin. Inet-3.0: Internet Topology Generator. Technical Report CSE-TR-456-02, EECS Department, University of Michigan, 2002.

[21] K. Calvert, M. Doar, and E. Zegura. Modelling Internet Topology. *IEEE Communications Magazine*, jun. 1997.

[22] S. Saroiu, P.K. Gummadi, and S.D. Gribble. A Measurement Study of Peer-to-Peer File Sharing Systems. In *Proceedings of Multimedia Computing and Networking 2002 (MMCN '02)*, San Jose, CA, USA, January 2002.

[23] V. Hardman, M. A. Sasse, M. Handley, and A. Watson. Reliable Audio for Use over the Internet. *Proc. of INET*, 1995.

[24] Lothar Pantel and Lars C. Wolf. On the suitability of dead reckoning schemes for games. In *Proceedings of the 1st workshop on Network and system support for games*, pages 79–84. ACM Press, 2002.

[25] Y. Bartal. Probabilistic Approximations of Metric Spaces and Its Algorithmic Applications. In *IEEE Symposium on Foundations of Computer Science*, pages 184–193, 1996.

[26] M.R. Garey and D.S. Johnson. Computers and Intractability, 1979.

[27] P. Krishnan, D. Raz, and Y. Shavitt. The Cache Location Problem. *IEEE/ACM Transactions on Networking*, 8(5):568–582, 2000.

[28] I.Althöfer, G.Das, D.Dopkin, D.Joseph, and J.Soares. On sparse spanners of weighted graphs. *Discrete and Computational Geometry*, 9:81–100, 1988.

[29] A. Liestman and T. Shermer. Additive Graph Spanners. *Networks*, 23:343–363, 1993.

[30] C. Jin, S. Jamin, D. Raz, and Y. Shavitt. *Building Scalable Internet Services: Theory and Practice*. Kluwer Academic Publishers, 2004.

[31] V. Jacobson. Pathchar - a Tool to Infer Characteristics of Internet Paths. Technical report, 1997.

[32] K. Lai and M. Baker. Measuring Link Bandwidths using a Deterministic Model of Packet Delay. pages 283–294, 2000.

[33] Manish Jain and Constantinos Dovrolis. End-to-end available bandwidth: measurement methodology, dynamics, and relation with tcp throughput. In *Proceedings of the 2002 conference on Applications, technologies, architectures, and protocols for computer communications*, pages 295–308. ACM Press, 2002.