MULTI+: Building Topology-Aware Overlay Multicast Trees

Luis Garcés-Erice, Ernst W. Biersack and Pascal A. Felber

Institut EURECOM
06904 Sophia Antipolis, France
{garces|erbi|felber}@eurecom.fr

Abstract. TOPLUS is a lookup service for structured peer-to-peer networks that is based on the hierarchical grouping of peers according to network IP prefixes. In this paper we present MULTI+, an application-level multicast protocol for content distribution over a peer-to-peer (P2P) TOPLUS-based network. We use the characteristics of TOPLUS to design a protocol that allows for every peer to connect to an available peer that is close. MULTI+ trees also reduce the amount of redundant flows leaving and entering each network, making more efficient bandwidth usage.

1 Introduction

IP Multicast seems (or, at least, was designed) to be the ideal solution for content distribution over the Internet: (1) it can serve content to an unlimited number of destinations, and (2) it is bandwidth-wise economic. These two characteristics are strongly correlated. IP Multicast saves bandwidth because a single data flow can feed many recipients. The data flow is only split at routers where destinations for the data are found in more than one outgoing port. Thus n clients do not need n independent data flows, which allows for IP Multicast's scalability. However, IP Multicast was never widely deployed in the Internet: security reasons, its open-loop nature, made IP multicast remain as a limited use tool for other protocols in LANs. The core Internet lacks of an infrastructure with the characteristics of IP Multicast.

Lately, with the advent of broadband links like ADSL and the generalization of LANs at the workplace, the *edges* of the Internet started to increase their bandwidth. Together with the ever-cheaper and yet more powerful equipment (computational power, storage capacity), they give the millions of hosts connected to the Internet the possibility of implementing themselves services that augment at the application level the capabilities of the network: the Peer-to-Peer (P2P) systems. Various application-level multicast implementations have been proposed [1–5], most of which are directly implemented on top of P2P infrastructures (Chord [6], CAN [7] or Pastry [8]). The good scalability of the underlying P2P networks give these application-level multicast one of the properties of the original IP Multicast service, that of serving content to a virtually unlimited number of clients (peers). However, these P2P networks are generally conceived as an application layer system completely isolated from the underlying IP network.

Thus, the P2P multicast systems that we know may fail at the second goal of IP Multicast: a LAN hosting a number of peers in a P2P multicast tree may find its outbound link saturated by *identical* data flowing to and from its local peers, unless those peers are somehow *aware* of the fact that they are sharing the same network. This is a critical issue for ISPs due to P2P file-sharing applications and flat-rate commercial offers that allow a home computer to be downloading content 24 hours a day. This problem also affects application-level multicast sessions if peers do not take care of the network topology. We have based our P2P multicast protocol on TOPLUS because of its inherent topology-awareness. We consider that there is a large population of peers, which justifies the utilization of Multicast, that many Multicast groups may coexist without interfering each other, and that each peer must accept to cooperate with others in low-bandwidth maintenance tasks, but they are not forced to transmit content that does not interest them.

Related Work. Some examples of overlay networks which introduce topology-awareness in their design are SkipNet [9], Coral[10], Pastry [11], CAN [12]. Application-level Multicast has given some interesting results like the NICE project [1] or End System Multicast [2]. Other application-level multicast implementations use overlay networks as we do to create Multicast trees: Bayeux [3], CAN [4] Pastry (Scribe) [5]. However, these approaches are not designed to optimize some metric like delay or bandwidth utilization. There is an interesting comparative in [13]. Content distribution overlay examples are SplitStream [14] and [15]. Recently, the problem of data dissemination on adaptive overlays has been treated in [16]. Our main contribution is the achievement of efficient topology-aware multicast trees with no or very little active measurement using distributed algorithms, while others aiming at similar goals require extensive probing [17], or rely on a much wider knowledge of the peer population [17] [18] than ours.

In the next section we present the main aspects of TOPLUS. In Section 3 we describe MULTI+. In Section 4 we comment some results on MULTI+ Multicast trees properties, before we conclude and sketch future work in Section 5.

2 TOPLUS Overview

TOPLUS [19] is based on the DHT paradigm, in which a resource is uniquely identified by a key, and each key is associated with a single peer in the network. Keys and peers share a numeric identifier space, and the peer with the identifier closest to a key is responsible for that key. The principal goal of TOPLUS is simple: each routing step takes the message closer to the destination.

Let I be the set of all 32-bit IP addresses. Let $\mathcal{G}$ be a collection of sets such that $G \subseteq I$ for each $G \in \mathcal{G}$. Thus, each set $G \in \mathcal{G}$ is a set of IP addresses. We refer to each such set G as a *group*. Any group $G \in \mathcal{G}$ that does not contain another group in $\mathcal{G}$ is said to be an *inner group*. We say that the collection $\mathcal{G}$ is a *proper nesting* if it satisfies all the following properties:

1. $I \in \mathcal{G}$.
2. For any pair of groups in $\mathcal{G}$, the two groups are either disjoint, or one group is a proper subset of the other.
3. Each $G \in \mathcal{G}$ consists of a set of contiguous IP addresses that can be represented by an IP prefix of the form $w.x.y.z/n$ (for example, $123.13.78.0/23$).

The collection of sets $\mathcal{G}$ can be created by collecting the IP prefix networks from BGP tables and/or other sources [20]. In this case, many of the sets $\mathcal{G}$ would correspond to ASes, other sets would be subnets in ASes, and yet other sets would be aggregations of ASes. This approach of defining $\mathcal{G}$ from BGP tables require that a proper nesting is created. Note that the groups differ in size, and in number of subgroups (the fanout). If $\mathcal{G}$ is a proper nesting, then the relation $G \subset G'$ defines a partial ordering over the sets in $\mathcal{G}$, generating a partial-order tree with multiple tiers. The set I is at tier-0, the highest tier. A group G belongs to tier 1 if there does not exist a G' (other than I) such that $G \subset G'$. We define the remaining tiers recursively in the same manner (see Figure 1).

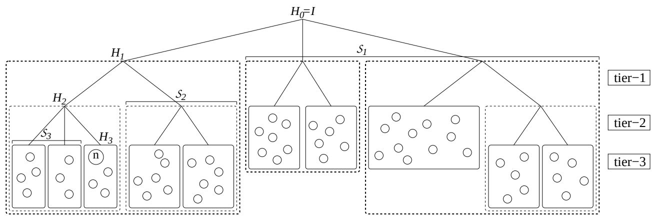


Fig. 1. A sample TOPLUS hierarchy (inner groups are represented by plain boxes)

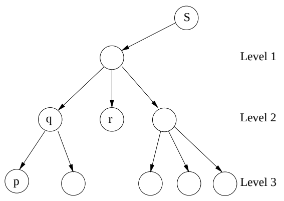


Fig. 2. A simple multicast tree.

Peer State. Let L denote the number of tiers in the tree, let U be the set of all current active peers and consider a peer $p \in U$. Peer p is contained in a collection of telescoping sets in $\mathcal{G}$; denote these sets by $H_N(p), H_{N-1}(p), \cdots, H_0(p) = I$, where $H_N(p) \subset H_{N-1}(p) \subset \cdots \subset H_0(p)$ and $N \leq L$ is the tier depth of p's inner group. Except for $H_0(p)$, each of these telescoping sets has one or more siblings in the partial-order tree (see Figure 1). Let $\mathcal{S}_i(p)$ be the set of siblings groups of $H_i(p)$ at tier i. Finally, let $\mathcal{S}(p)$ be the union of the sibling sets $\mathcal{S}_1(p), \cdots, \mathcal{S}_N(p)$.

Peer p should know the IP address of at least one peer in each group $G \in \mathcal{S}(p)$, as well as the IP addresses of all the other peers in p's inner group. We refer to the collection of these two sets of IP addresses as peer p's *routing table*, which constitutes peer p's state. The total number of IP addresses in the peer's routing table in tier L is $|H_L(p)| + |\mathcal{S}(p)|$. In [19] we describe how a new peer can join an existing TOPLUS network.

XOR Metric. Each key k' is required to be an element of I', where I' is the set of all s-bit binary strings ($s \geq 32$ is fixed). A key can be drawn uniformly randomly from I', or it can be biased as we shall describe later. For a given key $k' \in I'$, let k be the 32-bit suffix of k' (thus $k \in I$ and $k = k_{31}k_{30} \ldots k_1 k_0$). Throughout the discussion below, we will refer to k rather than to the original k'.

The XOR metric defines the distance between two IDs j and k as $d(j,k) = \sum_{\nu=0}^{31} |j_\nu - k_\nu| \cdot 2^\nu$. The metric $d(j,k)$ has the following properties, for IDs i,j and k:

- If $d(i,k) = d(j,k)$ for any k, then $i = j$.
- Let $p(j,k)$ be the number of bits in the common prefix of j and k. If $p(j,k) = m$, $d(j,k) \leq 2^{32-m} - 1$.

– If $d(i,k) \leq d(j,k)$, then $p(i,k) \geq p(j,k)$.

$d(j,k)$ is a refinement of longest-prefix matching. If j is the unique longest-prefix match with k, then j is the closest to k in terms of the metric. Further, if two peers share the longest matching prefix, the metric will break the tie. The peer p^* that minimizes $d(k,n)$, $p \in U$ is "responsible" for key k.

3 MULTI+: Multicast on TOPLUS

A Multicast Tree. First we assume that all peers are connected through links providing enough bandwidth. A simple multicast tree is shown in Figure 2. Let S be the source of the multicast group m. Peer p is receiving the flow from peer q. We say that q is the parent of p in the multicast tree. Conversely, we say that p is a child of q. Peer p is in level-3 of the multicast tree and q in level-2. It is important to note that, in principle, the *level* where a peer is in the multicast tree has nothing to do with the *tier* the peer belongs to in the TOPLUS tree.

In the kind of multicast trees we aim at building, each peer should be close to its parent in terms of network delay, while trying to join the multicast tree as high (close to the source) as possible. Each peer attempts at join time to minimize the number of hops from the source, and the length of the last hop. In the example of Figure 2, if p is a child of q and not of r, that is because p is closer to q than to r. By trying to minimize the network delay for data transmission between peers, we also avoid rearranging peers inside the multicast tree, except when a peer fails or disconnects.

Building Multicast Trees. We use the TOPLUS network and look-up algorithm in order to build the multicast trees. Consider a multicast IP address m, and the corresponding key that, abusing the notation, we also denote m. Each tier-i group G_i is defined by an IP network prefix a_i/b where a_i is an IP address and b is the length of the prefix in bits. Let m_i be the key resulting from the substitution of the first b bits of m by those of a_i. The inner group that contains the peer responsible for m_i (obtained with a TOPLUS look-up) is the *responsible inner group*, or RIG, for m in G_i (note that this RIG is contained in G_i.) Hereafter, we assume a single m, and for that m and a given peer p we denote the RIG in $H_i(p) \in$ tier-i simply as RIG-i of p. This RIG is a rendezvous point for all peers in $H_i(p)$. The deeper that a tier-i of a RIG-i is in the TOPLUS tree, the narrower the scope of the RIG as a rendezvous point (fewer peers can potentially use it).

In the simple 3-tier example of Figure 3, we have labeled the RIGs for a given multicast group (peers in grey are members of the multicast group), where all inner groups are at tier-3. The RIG-i of a peer can be found following the arrows. The arrows represent the process of asking the RIGs for a parent in the multicast tree. For example, p and q share the same RIG-1 because they are in the same tier-1 group. t's inner group is its RIG-1, but t would first contact a peer x (white) in its RIG-2 to ask for a parent. Note that this last peer is not in the multicast tree (Figure 4).

Assume a peer p in tier-$(i+1)$ (i.e., a peer whose inner group is at tier-$(i+1)$ of the TOPLUS tree) wants to join a multicast tree with multicast IP address m, which we call group m.

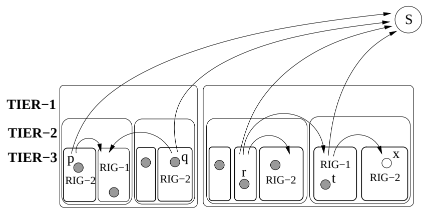


Fig. 3. The RIGs in a sample TOPLUS network.

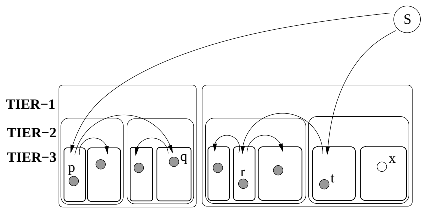


Fig. 4. Sample multicast tree.

1. The peer p broadcasts a query to join group m inside its inner group. If there is a peer p' already part of group m, p connects to p' to receive the data.
2. If there is not such peer p', p must look for its RIG-i. A look-up of m_i inside p's tier-i group (thus among p's sibling groups at tier-$(i+1)$) locates the RIG-i responsible for m. p contacts any peer p_i in RIG-i, and asks for a peer in multicast group m. If peer p_i knows about a peer p'' that is part of m, it sends the IP address of p'' to p, and p connects to p''. Note that p'' is not necessarily a member of the RIG-i inner group. In any case p_i adds p to the peers listening to m, and shares this information with all peers in RIG-i. If p'' does not exist, p proceeds similarly for RIG-$(i-1)$: p looks up m_{i-1} inside p's tier-$i-1$ group (i.e., among p's sibling groups at tier i). This process is repeated until a peer receiving m is found, or RIG-1 is reached. In the latter case, if there is still no peer listening to m, peer p must connect directly to the source of the multicast group. One can see that the search for a peer to connect to is done bottom up.

Property 1. *When a peer p in tier $i+1$ joins the multicast tree, by construction, from all the groups $H_{i+1}(p), H_i(p), \cdots, H_1(p)$ that contain p, p connects to a peer $q \in H_k$ where $k = \max\{l = 1, \ldots, i+1\} : \exists r \in H_l$ and r is a peer already connected to the multicast tree. That is, p connects to a peer in the deepest tier group which contains both p and a peer already connected to the multicast tree.*

This assures that a new peer connects to the closest available peer in the network. Notice that even in the case of failure of a peer in a RIG-i, the information is replicated in all other peers in the RIG-i. If a whole RIG-i group fails, although MULTI+ is undeniably affected, the look up process can continue in RIG-$(i-1)$. We believe this property makes MULTI+ a resilient system.

Property 2. *Using multicast over TOPLUS, the total number of flows in and out of a group defined by an IP network prefix is bounded by a constant.*

Due to lack of space, we do not further develop this important aspect of MULTI+. We refer the interested reader to the Technical Report [21]. However, in the experiments below we will notice the tight number of flows per network prefix.

Membership Management. Each peer p knows its parent q in the multicast tree, because there is a direct connection between them. Because p knows the RIG where it got its parent's address, if p's parent q in level i of the multicast tree fails or disconnects, p directly goes to the same RIG and asks for a new parent. If there is none, p becomes the new tree node at level i, replacing q. Then p must find a parent in level $i-1$ of the multicast tree, through a join process starting at said RIG. If p had any siblings under its former parent, those siblings will find p as the new parent when they proceed like p. If more than one peer concurrently tries to become the new node at level i, peers in the RIG must consensually decide on one of them. It is not critical if a set of peers sharing a parent q are divided in two subsets with different parents upon q's depart.

Join and leave is a frequent process in a P2P network, but we expect the churn to be rather low due to the fact that in a multicast tree, all peers seek the same content concurrently, throughout the duration of the session.

Parent Selection Algorithms. From the ideas exposed before, we retain two main parent selection algorithms for testing the construction of multicast trees.

– FIFO, where a peer joins the multicast tree at the first parent found with a free connection. When a peer gets to a RIG to find a parent, the RIG answers with a list of already connected peers. This list is ordered by arrival time to the RIG. Obviously, the first to arrive connects closer (in hops) to the source. The arriving peer tests each possible peer in the list starting with the first one until it finds one that accepts a connection.
– Proximity-aware, where, *when the first parent in the list has all connections occupied*, a peer connects to the closest parent in the list still allowing one extra connection.

Note that we do not always verify if we are connecting to the closest parent in the list. The idea behind this is that, while we implicitly trust MULTI+ to find a close parent, we prefer to connect to a peer higher in the multicast tree (fewer hops from the source) than to optimize the last hop delay. If MULTI+ works correctly, the difference between these two policies should not be excessive, because the topology-awareness is already embedded in the protocol through TOPLUS.

4 MULTI+ Performance

Obviously, the $O(n^2)$ cost of actively measuring the full inter-host distance matrix for n peers limits the size of the peer sets we can use [21]. P2P systems must be designed to be potentially very large, and experiments should reflect this property by using significant peer populations. Methods like [22] map hosts into a M-dimensional coordinate space. The main advantage is that given a list of n hosts, the coordinates for all of them can be actively measured in $O(Mn)$ time (the distances of the hosts to a set of M *landmark* hosts, with $M \ll n$).

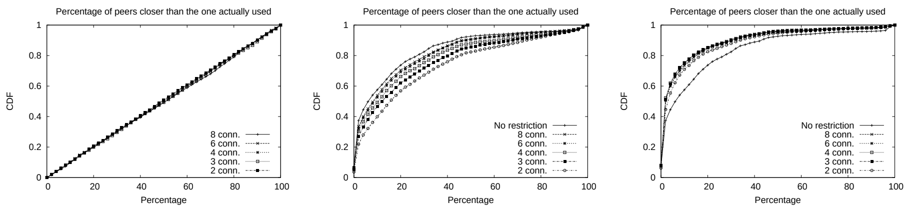
TC Coordinates. CAIDA [23] offers to researchers a set of network distance measurements from so-called *Skitter* hosts to a large number of destinations. `Skitter` is a traffic measurement application developed by CAIDA. In a recent paper [22], the authors have used these and other data to obtain a multi-dimensional coordinate space representing the Internet. A host location is denoted by a point in the coordinate space, and the latency between two hosts can be calculated as the distance between their corresponding points. The authors of [22] have kindly provided us with the coordinates of $196,297$ IP addresses for our study. Hereafter we call this space the TC (from Tang and Crovella) coordinate space. We calculate distances using a Euclidean metric, defined $D(x_i, x_k) = \sqrt{\sum_{k=1,\ldots,M}(x_{ik} - x_{jk})^2}$, for any two hosts identified by their M-coordinate vectors x_i and x_j.

5000 Peers Multicast Tree. In this experiment we test the characteristics of Multicast trees built with MULTI+ using the TC coordinate space and a set of 5,000 peers. We use the coordinate space to measure the distance between every pair of hosts. In order to make the experiment as realistic as possible, we use a TOPLUS tree with routing tables of reduced size, obtained from the grouping of small and medium-sized tier-1 groups into virtual groups, and this process introduces a distortion in the topological fidelity of the resulting tree [19]. The 5,000 peers are organized into a TOPLUS tree with 59 tier-1 groups, 2,562 inner-groups, and up to 4 tiers. We evaluate the two different parent selection policies described before: FIFO and proximity-aware. We also compare these two approaches with random parent selection. In all cases we test MULTI+ when we do not set a limit on the maximum number of connections a peer can accept, and for a limited number of connections, from 2 to 8 per peer. In the test we measure the following parameters, presented here using their CDF (Cumulative Distribution Function):

– The percentage of the peers in the total system, when the full multicast tree is built, closer to one peer that this peer's parent. Those figures *exclude* the peers directly connected to the source (Figure 5).
– The level peers occupy in the multicast tree. The more levels in the multicast tree, the more delay we incur in along the transmission path and the more the transmission becomes subject to losses due to peer failure (Figure 6).
– The latency from the root of the multicast tree to each receiving peer (Figure 7).
– The number of multicast flows that go into and out of each TOPLUS group (network) (Figure 8).

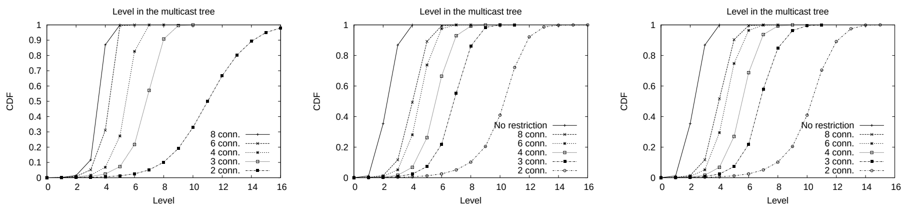
From our experiments we obtain very satisfactory results. From Figures 5 to 8 we draw a number of conclusions:

– Individual peers do not need to support a large number of outgoing connections to benefit from MULTI+ properties: three connections are feasible for broadband users, and the marginal improvement of 8 connections is not very significant.
– The proximity-aware policy performs better than FIFO in terms of end-to-end latency (Figure 7) and connection to closest parent (Figure 5). However, with respect to the number of flows per group (Figure 8) and level distribution in the multicast tree (Figure 6), they are very similar. That is because both trees follow the TOPLUS structure, but the proximity-aware policy takes better decisions when the optimal parent peer has no available connections.
– In Figure 5(c) we can see that having no connection restrictions makes closeness to parent less optimal than having restrictions, for the proximity-aware policy. This is normal, since when we have available connections, a peer's main goal is to connect



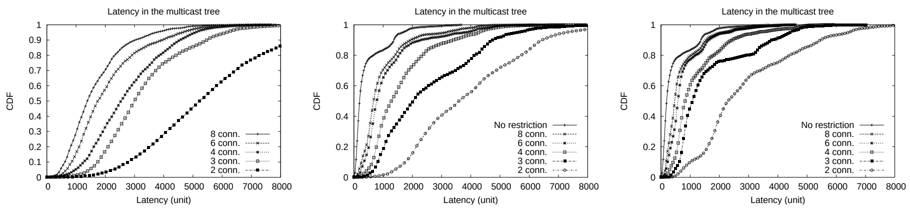
(a) Random parent selection. (b) FIFO parent selection. (c) Proximity-aware parent selection.

Fig. 5. Percentage of peers in the whole system closer than the one actually used (for those not connected to the source.)



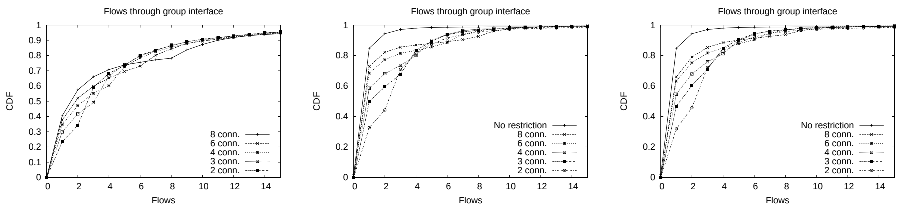
(a) Random parent selection. (b) FIFO parent selection. (c) Proximity-aware parent selection.

Fig. 6. Level of peers in the multicast tree.



(a) Random parent selection. (b) FIFO parent selection. (c) Proximity-aware parent selection.

Fig. 7. Latency from root to leaf (in TC coordinate units) in the Multicast tree.



(a) Random parent selection. (b) FIFO parent selection. (c) Proximity-aware parent selection.

Fig. 8. Number of flows through group interface.

as high in the multicast tree as possible. See in Figure 6 how peers are organized in fewer levels, and in Figure 7 how the root-to-leaf latency is better for the unrestricted connection scheme. Still, we can assert that the multicast tree is following (when possible) a topology-aware structure, because most peers connect to nearby parents.

– The random parent selection policy organizes the tree in fewer levels than the other two policies (Figure 6(a)), because connections are not constrained to follow the TOPLUS structure. However those connections are not optimized, and the resulting end-to-end delay performance in any aspect is considerably poorer.

5 Conclusion and Future Work

We have presented MULTI+, a method to build application-level multicast trees on P2P systems. MULTI+ relies on TOPLUS in order to find a proper parent for a peer in the multicast tree. MULTI+ exhibits the advantage of being able to create topology-aware content distribution infrastructures without introducing extra traffic for active measurement. Admittedly, out-of-band information regarding the TOPLUS routing tables must be calculated offline (a simple process) and downloaded (like many P2P systems today require to download a list of peers for the join process). The proximity-aware scheme improves the end-to-end latency, and using host coordinates calculated offline and obtained at join time (as is done for TOPLUS) avoids the need for any active measurement. MULTI+ also decreases the number of redundant flows that must traverse a given network, even when only few connections per peer are possible, which allows for better bandwidth utilization. As future work, we plan to evaluate the impact of leaving and failing peers on the multicast tree performance, as well as comparing its properties with other systems.

Acknowledgments. This work could not have been done without the kind help of Prof. Mark Crovella, who provided us with the TC coordinate space data for our experiments.

References

1. Banerjee, S., Bhattacharjee, B., Kommareddy, C.: Scalable application layer multicast. In: Proceedings of ACM SIGCOMM. Pittsburgh, PA, USA (2002)
2. hua Chu, Y., Rao, S.G., Seshan, S., Zhang, H.: A case for end system multicast. IEEE Journal on Selected Areas in Communication (JSAC), Special Issue on Networking Support for Multicast **20** (2003)
3. Zhuang, S.Q., Zhao, B.Y., Joseph, A.D., Katz, R.H., Kubiatowicz, J.D.: Bayeux: An architecture for scalable and fault-tolerant wide area data dissemination. In: Proceedings of NOSSDAV'01. Port Jefferson, NY, USA (2001) 124–133
4. Ratnasamy, S., Handley, M., Karp, R.M., Shenker, S.: Application-level multicast using content-addressable networks. In: Proceedings of NGC, London, UK (2001) 14–29
5. Castro, M., Druschel, P., Kermarrec, A.M., Rowstron, A.: Scribe: a large-scale and decentralized application-level multicast infrastructure. IEEE Journal on Selected Areas in Communications **20** (2003) 1489 –1499

6. Stoica, I., Morris, R., Karger, D., Kaashoek, M., Balakrishnan, H.: Chord: A scalable peer-to-peer lookup service for internet applications. In: Proceedings of ACM SIGCOMM, San Diego, CA, USA (2001)
7. Ratnasamy, S., Handley, M., Karp, R., Shenker, S.: A scalable content-addressable network. In: Proceedings of ACM SIGCOMM, San Diego, CA, USA (2001)
8. Rowstron, A., Druschel, P.: Pastry: Scalable, distributed object location and routing for large-scale peer-to-peer systems. In: IFIP/ACM International Conference on Distributed Systems Platforms (Middleware), Heidelberg, Germany (2001) 329–350
9. Harvey, N.J.A., Jones, M.B., Saroiu, S., Theimerm, M., Wolman, A.: Skipnet: A scalable overlay network with practical locality properties. In: Proceedings of the Fourth USENIX Symposium on Internet Technologies and Systems (USITS '03), Seattle, WA, USA (2003)
10. Freedman, M., Mazieres, D.: Sloppy hashing and self-organizing clusters. In: Proceedings of 2nd International Workshop on Peer-to-Peer Systems (IPTPS '03), Berkeley, CA, USA (2003)
11. Castro, M., Druschel, P., Hu, Y.C., Rowstron, A.: Topology-aware routing in structure peer-to-peer overlay network. In: International Workshop on Future Directions in Distributed Computing (FuDiCo), Bertinoro, Italy (2002)
12. Shenker, S., Ratnasamy, S., Handley, M., Karp, R.: Topologically-aware overlay construction and server selection. In: Proceedings of IEEE INFOCOM, New York City, NY (2002)
13. Castro, M., Jones, M.B., Kermarrec, A.M., Rowstron, A., Theimer, M., Wang, H., Wolman, A.: An evaluation of scalable application-level multicast built using peer-to-peer overlays. In: Proceedings of IEEE INFOCOM, San Francisco, USA (2003)
14. Castro, M., Druschel, P., Kermarrec, A.M., Nandi, A., Rowstron, A., Singh, A.: Splitstream: High-bandwidth multicast in a cooperative environment. In: Proceedings of SOSP'03, New York, USA (2003)
15. Byers, J.W., Considine, J., Mitzenmacher, M., Rost, S.: Informed content delivery across adaptive overlay networks. In: Proceedings of ACM SIGCOMM, Pittsburgh, PA, USA (2002) 47–60
16. Zhu, Y., Guo, J., Li, B.: oEvolve: Towards evolutionary overlay topologies for high bandwidth data dissemination. IEEE Journal on Selected Areas in Communications, Special Issue on Quality of Service Delivery in Variable Topology Networks (2004)
17. Rodriguez, A., Kostic, D., Vahdat, A.: Scalability in adaptive multi-metric overlays. In: Proceedings of IEEE ICDCS, Tokyo, Japan (2004) 112–121
18. Riabov, A., Liu, Z., Zhang, L.: Overlay multicast trees of minimal delay. In: Proceedings of IEEE ICDCS, Tokyo, Japan (2004) 654–664
19. Garcés-Erice, L., Ross, K.W., Biersack, E.W., Felber, P.A., Urvoy-Keller, G.: Topology-centric look-up service. In: Proceedings of COST264/ACM Fifth International Workshop on Networked Group Communications (NGC), Munich, Germany (2003) 58–69
20. Krisnamurthy, B., Wang, J.: On network-aware clustering of web sites. In: Proceedings of ACM SIGCOMM, Stockholm, Sweden (2000)
21. Garcés-Erice, L., Biersack, E.W., Felber, P.A.: Multi+: Building topology-aware overlay multicast trees. Technical Report RR-04-107, Institut Eurecom, Sophia-Antipolis, France (2004)
22. Tang, L., Crovella, M.: Virtual landmarks for the internet. In: Proceedings of the ACM SIGCOMM Internet Measurement Conference (IMC-03), Miami Beach, Florida, USA (2003)
23. CAIDA (http://www.caida.org/)