

# A Survey and Comparison of End-System Overlay Multicast Solutions Suitable for Network Centric Warfare

Cristina Abad<sup>†‡</sup>, William Yurcik<sup>‡</sup> and Roy H. Campbell<sup>†</sup>

<sup>‡</sup>National Center for Supercomputing Applications (NCSA), Champaign, IL, USA;

<sup>†</sup>Department of Computer Science, University of Illinois at Urbana-Champaign

## ABSTRACT

Multicasting at the IP layer has not been widely adopted due to a combination of technical and non-technical issues. End-system multicast (also called application-layer multicast) is an attractive alternative to IP layer multicast for reasons of user management (set-up and control) and attack avoidance. Sessions can be established on demand such that there are no static points of failure to target in advance.

In end-system multicast, an overlay network is built on top of available network services and packets are multicasted at the application layer. The overlay is organized such that each end host participating in a multicast communication re-sends multicasted messages to some of its peers, but not all of them. Thus end-system multicast allows users to manage multicast sessions under varying network conditions without being dependent on specific network conditions or specific network equipment maintaining multicast state information.

In this paper we describe a variety of proposed end-system multicast solutions and classify them according to characteristics such as overlay building technique, management, and scalability. Comparing these characteristics across different end-system multicast solutions is a step toward understanding which solutions are appropriate for different battlespace requirements and where further research is needed.

**Keywords:** Application-layer multicast, End-system multicast, Overlay networking

## 1. INTRODUCTION

Multicasting is a one-to-many transfer service that scales sub-linearly. Such functionality is required (or desired) for applications like video streaming, distributed simulations, video-conferencing, multi-party games, content distribution, etc. Specifically for this paper, the ability of multicast technology to enable real-time collection, processing, and dissemination of sensor/command-and-control/warfighter information is crucial to the success of the future battlespace. The paradigm of network-centric warfare extends beyond sensors and weapon systems by linking communications and logistics support across traditional service boundaries (Air Force, Army, Navy) to create a fully integrated system for sharing critical operational data among commanders.

IP multicast,<sup>1</sup> which runs at the network layer, has not been widely deployed yet.<sup>2</sup> ISPs are reluctant to enable IP multicast due to a combination of “complex billing, management and security concerns”.<sup>3</sup> Furthermore, it is costly to upgrade the network infrastructure, and many organizations worldwide still use routers that do not support IP multicast. Other issues arise when reliable delivery is desired. IP multicast, as originally proposed,<sup>1</sup> is a best effort service. Reliable multicast has been proposed,<sup>4,5</sup> but the lack of standards and concerns about network congestion are an issue. Finally, IP multicast requires group set-up and maintenance, a burden in highly dynamic settings.

End System Multicast\* was proposed<sup>6</sup> as an alternative that helps overcome many of the problems that have delayed the widespread use of IP Multicast. The new battlespace paradigm of network-centric warfare fits well with end system multicast because they are both software-based (not hardware-based), dynamic (not static), easily accessible on-demand, and focused on supporting the warfighter and not on the underlying technology.

While several approaches to End System Multicast have been proposed<sup>6-16</sup> there is still a lot of research that needs to be done in the area. We present an overview of several of the End System Multicast proposed

---

Corresponding author: Cristina Abad, E-mail: cabad@ncsa.uiuc.edu, Telephone: (217) 333 2117

\*Also referred to as Application layer multicast, application level multicast and overlay multicast.

solutions and describe the issues that have not been addressed yet or that need to be further researched. The remainder of this paper is organized as follows: Section 2 surveys the current state-of-the-art in end system multicast solutions. Section 3 classifies the end system multicast schemes according to design choices. Section 4 identifies and examines the unsolved design and implementation aspects that need further study. Section 5 maps the primary design characteristics of end system multicast research systems to general application characteristics that can be mapped to the network-centric military scenarios. We conclude in Sect. 6.

## 2. SURVEY

### 2.1. End System Multicast (Narada)

The End System Multicast (ESM)<sup>6</sup> uses a two step process to build and refine the source-specific multicast tree. First, it builds a richer connected graph or mesh and tries to ensure that the mesh has some desirable performance properties. The spanning trees of the mesh, each rooted at the source, are built using a distance vector routing algorithm in which the trees are constructed from the reverse shortest path between each recipient and the source in the same way DVMRP does it. Each tree can be optimized for each source to improve performance. The efficiency of the resulting tree depends on building it on top of a “good” mesh, in which the quality of the path between two members is comparable to the quality of the unicast path between them, and each member has a limited number of neighbors in the mesh.

Narada was designed for small to medium sized groups. In that context, the overhead of maintaining the tree is not significant, but it would be unmanageable for large groups. Each member maintains a list of all other members in the group and periodically exchanges its knowledge of group membership with its neighbors in the mesh. To join the mesh, a new member randomly selects peers from a list of connected members obtained through an out-of-band mechanism and requests to be their neighbor. If a host stops receiving refresh messages from one of its neighbors, it probes it and if the member is dead it propagates this information through the mesh. To improve the quality of the mesh, peers probe each other at random and new links may be added depending on the perceived gain in utility in doing so. Links can also be dropped in this way. For stability, the gain has to be significant before changing a link. Chu et al. also explore the use of ESM for conferencing applications.<sup>17</sup>

### 2.2. ALMI: Application Level Multicast Infrastructure

The Application Level Multicast Infrastructure (ALMI)<sup>7</sup> was designed to accommodate a large number of groups, each with a small number of members. ALMI uses a centralized algorithm to improve the multicast tree. The Rendezvous Point (RP) or Session Controller calculates the minimum spanning (shared) tree (MST). A node that wishes to join the multicast tree contacts the RP and receives from it the member ID and the address of its parent in the tree. Messages go in both directions, from parents to children and vice-versa. The parent-child relationship is used for mesh control purposes. The RP also tells members the address of other peers periodically probe and evaluate the performance of the unicast paths to them. The RP uses this information to improve the spanner graph by removing bad edges and updating them with edges not currently in the graph, and improve the tree. To prevent path oscillations, the RP calculates the overall performance gain of the new multicast tree and switches tree only if the overall gain exceeds a threshold. To avoid problems that can arise from changing trees, each tree has a version number and each node maintains a small cache of recent multicast tree incarnations. Each message propagated through the tree has a field indicating to which tree version it belongs.

The centralized approach enables better reliability and reduced overhead. Communication can go on in the absence of the RP, but new members can not join the tree. To avoid failure, the use of backup-RPs is mentioned. The controller could enforce access control and act as a key distribution center, but these is left for future work.

### 2.3. Yoid: Your Own Internet Distribution

Your Own Internet Distribution (Yoid)<sup>8</sup> is DARPA-sponsored application-level multicast solution. Each group has one or more RPs. When a host wishes to join a group, it contacts its RP and receives an ID (unique within the group) and a list of some current members. The node chooses one of the members to be its parent and requests to join it. If it later loses connectivity to the parent, it selects a new parent. The RP keeps its list of

members current by explicitly checking for their liveness. The RP also selects and maintains the identity of the root of the tree. The identity of the root is maintained by using an election algorithm and a keep-alive protocol.

Yoid’s distributed (shared) tree formation algorithm may create loops. Loop detection and rapid loop termination algorithms are proposed to solve the problem. Each node keeps a list of nodes between the root and itself to easily identify loops. When a loop is formed, the node with the largest switchstamp (a number assigned to each node in increasing size from the path from the root to each node) changes parent.

When needed, the tree is refined for improved latency and loss-rate. These changes are data triggered; if there is no active sender the tree will not change. Yoid dynamically refines the tree based on observed data losses. Each host compares its losses with its neighbors’ and if they differ significantly, it decides to switch parent. Hosts occasionally test new parents to see if they can deliver messages at a significantly lower latency.

## 2.4. NICE

NICE<sup>9</sup> is a cooperative framework to scale multi-party applications developed by Banerjee et al. at the University of Maryland. Its application-layer multicast protocol is also called NICE. It uses a distributed algorithm by which nodes are self-organized into a layered topology with nodes organized at each layer into clusters. Only the leader of each cluster is part of the next layer which is also organized in clusters. Layer 0 contains all the nodes, while the last layer contains only one host, the RP. For any host wishing to send a multicast message, its data delivery path is a source-specific tree implicitly defined by forwarding each message to all other members of the cluster(s) it belongs to, except to the host the message was received from. The layered design helps it scale better, with a worst-case state and control overhead for any member of  $O(\log N)$ , and approaches the  $O(\log N)$  stretch bound possible with a topology-aware centralized algorithm. Their simulations show that for groups of size 32 or more, NICE has lower link stress, improved or similar end-to-end latencies and similar recovery properties than Narada.

Each host maintains soft state about the other members of the cluster(s) it belongs to. To join a group, a node contacts the RP which gives it a list of all the hosts that are part of the highest layer of the hierarchy. The joining host probes each and selects the “closest” one (measured by the end-to-end latency of the unicast path). It contacts this host which replies with a list of all the members of the cluster in the next layer. The process goes on until the new host finds the host “closest” to it and joins the appropriate cluster. This process takes  $O(k \log N)$  query-response pairs. To improve this, a host can temporarily join a higher layer to start receiving multicast messages right away. The leader of each cluster checks periodically the size of the cluster and if it is smaller or greater than the desired bounds it joins it with another cluster or splits it, respectively. To improve the quality of the control graph, each member in any layer periodically probes all members in its super-cluster to see if it can find a “closer” cluster to join. When a node leaves (explicitly or after failure), each of the remaining members of the cluster independently selects a leader by estimating what host is the center among the members. Then, by exchanging refresh messages, the hosts agree on a single leader.

## 2.5. Bayeux

Bayeux<sup>10</sup> is an architecture for scalable and fault-tolerant wide-area data dissemination built on top of Tapestry.<sup>18</sup> Tapestry is a structured peer-to-peer overlay routing infrastructure, in which nodes are assigned unique IDs and messages are routed incrementally through the overlay according to each node ID. Tapestry uses two secondary pointers in each neighbor map entry to allow routing in the presence of node failures. Objects to be stored in Tapestry are deterministically mapped to a node. Object look-up is done by routing request messages on the overlay to the node where the object is mapped. If a replica is found along the way, it is returned and the forwarding stops.

Not every node in the Tapestry overlay is part of a Bayeux multicast group. Even so, nodes not participating in a multicast session may have to route multicast messages if necessary. Bayeux multicast trees are source-specific trees built on top of a Tapestry overlay. To initiate a multicast session, a trivial file named as the session unique Id is created, mapped to the source (root) node, and advertised using Tapestry’s data location services. To join a session, a node sends a JOIN message to the root, which responds with a TREE message. The TREE message sets up tree forwarding state at intermediate application-level routers.

Some extensions to the protocol for improved scalability, link stress and fault-resiliency are suggested. Multiple root nodes can be used to improve scalability. The receivers are partitioned into disjoint membership sets,

each containing receivers closest to a local root in network distance. Simulations show that the load is effectively balanced among the different roots, even if they are randomly distributed. Tapestry's secondary routing path is used in Bayeux for fault-resiliency. When a backup route is taken, the node where the branching occurs forwards the necessary member state through the new path, ensuring packet delivery without the overhead of maintaining multiple multicast trees.

## 2.6. Banana Tree Protocol (BTP)

The Banana Tree Protocol (BTP)<sup>11</sup> is a “switch tree protocol” currently in use by the Jungle Monkey<sup>19</sup> peer-to-peer file sharing program and also in IDMaps,<sup>20</sup> an Internet distance measurement system. According to Helder et al., switch-trees are peer-to-peer algorithms for building and improving end-host multicast trees. The paper presents and evaluates several approaches for improving end-host multicast trees, and proposes the BTP protocol. Packets are multicasted in the BTP tree by having each node forward the packet to all of its neighbors. New nodes join directly to the root (or become the root if the tree is empty). When a node detects that its parent has left the tree it proceeds to re-join the tree as a direct child of the root. Periodically each node requests from its parent a list of the parent neighbors. By pinging each host on the list it determines if there is a “closer” node to itself than its parent. If it finds such node, it switches parents. Several preventive measures are taken to avoid creating loops: a node rejects all switching attempts if it is itself in the process of switching parents, also a switching node includes its current parent information in the switch request so the potential parent can check if the node and itself are actually sharing the same parent. Some preliminary experiments show that BTP produces low cost trees, but the testing is not thorough and future work should expand on this.

## 2.7. Overcast

Overcast provides “scalable and reliable single-source multicast using a simple protocol for building efficient data distribution trees that adapt to changing network conditions”.<sup>13</sup> An example of its use is large-scale software distribution to web clients. Overcast creates a self-organizing tree of content servers. To obtain a specific content, a client refers to an URL containing the address of the root of the group plus the name of the resource. The root selects the most appropriate content server from the tree and re-directs the client to it, thus making it transparent to the end-user. The content servers store content that is multicasted through the tree.

Nodes cooperatively self-organize into a tree overlay with the source node at its root. The tree is built with the goal of maximizing bandwidth to the root for all nodes. To achieve this, nodes are placed as far away from the root as possible without sacrificing bandwidth to the root. A node initially joins the tree directly to the root. It then tries to move farther away by comparing its bandwidth to the root through its current parent and its bandwidth to the root if attached as a child of each of its parent children. To approximate the bandwidth to the root the download time of 10 KB is measured. This is believed to be more accurate than ping. If two paths yield very similar bandwidth to the root (less than 10% difference), then the node that is closest as reported by `traceroute` is selected. Each node periodically checks if it can improve its position in the tree by measuring the bandwidth to its current siblings, parent, and grandparent. An ancestor list kept at each node is used to avoid forming loops. Soft state is kept at each node to keep track of which nodes are “alive”. Statistical information of each node is periodically reported back to the root, so it can make better choices when re-directing clients. For fault-tolerance, the root can be replicated and the replicas placed linearly at the beginning of the tree.

## 2.8. Application-level Multicast using Content-Addressable Networks (CANs)

Content-addressable networks (CANs)<sup>21</sup> are structured peer-to-peer overlays. Nodes are divided in a coordinate space, each owning a part of that space. To store a pair  $(k, v)$ , the key is deterministically mapped onto a point  $P$  in the coordinate space using a uniform hash function. Messages are routed through the space by following the straight line path through the Cartesian space from source to destination coordinates. An object is inserted by mapping its key to the space. A key could be the object name or related words.

CANs can be extended to do application-level multicasting aimed at large group sizes.<sup>12</sup> Any member can multicast messages in the CAN. Multicasting is done by intelligently flooding messages across the CAN. Heuristics are used to minimize the number of duplicate messages forwarded. Each member caches the ids of the

messages received and does not forward duplicate messages. Simulations results show that 97% of the members receive no duplicates and the other 3% receive one duplicate message on average.

A problem with the originally proposed CAN and CAN-multicast was that node mapping in the Cartesian space bears no relationship with any “closeness” metric. But, this issue was later addressed.<sup>22</sup> With the proposed extension, before joining a node has to probe a set of landmarks and based on its delay measurements join the CAN at a random point in the portion of the coordinate space associated to that landmark ordering.

## 2.9. Scribe

Scribe<sup>23</sup> is a large-scale and decentralized application-level multicast infrastructure that works on top of Pastry,<sup>24</sup> a structured peer-to-peer object location and routing substrate. A group is created by a CREATE message to the node with the ID “closest” to the groupID, which becomes the RP. A JOIN message is routed to the RP through Pastry until it reaches a node that is a forwarder to that group. Each node along the path from each subscriber to the root (RP) becomes a forwarder of the group, even if it is not a group member. Each forwarder maintains a (soft state) children table, used for tree routing. Thus, the tree building process is somewhat similar to Reverse Path Forwarding (RPF).<sup>25</sup> Any node can multicast a message to the group by sending it to the RP which is the tree root. This enables the RP to perform access control.

Scribe has some nice scalability properties. For example, the RP does not need to handle all subscriptions since the JOIN messages stop as soon as they found a forwarder. Also, multiple groups can be formed within one Pastry overlay. The tree formation is completely decentralized and Pastry’s randomization leads to well balanced trees and load. Furthermore, a bottleneck remover algorithm described in the paper allows nodes to off-load children to other nodes.

## 2.10. Bullet

Bullet<sup>14</sup> is a mesh-based data dissemination overlay. Kostic et al. argue that tree-based dissemination overlays may not be the best when high bandwidth is desired. Overlay trees limit bandwidth, are hard to optimize and lead to high overhead due to probing. The proposed solution is to use a mesh instead of a tree.

Bullet starts with an overlay tree which can be built with any of the already existing protocols. The sender selects a random subset of its children and sends them a disjoint data set of the data it wishes to transmit. How disjoint this set is depends on the bandwidth constraints of the peers. The same process is followed by the children as they propagate the data down the tree. Nodes then obtain the missing parts of the data from other peers. The paper describes an efficient way for disseminating the information of which nodes have received copies of the different parts of the data.

Another important characteristic of Bullet is that it uses TFRC<sup>26</sup> as the transport layer protocol to achieve TCP-friendly data transmissions across the overlay.

Their simulations and PlanetLab implementation results show that with a reasonable overhead (up to 30 Kbps), Bullet is able to provide a bandwidth significantly higher than that obtained with a pseudo-optimal off-line built overlay tree. Less than 10% of the packets transmitted are duplicates.

## 2.11. SplitStream

SplitStream<sup>15</sup> was designed for overlay transmission of large data such as streaming media. As the authors point out, tree-based multicast imposes high-load to interior nodes. Thus, for it to provide good results, these nodes should be highly-available and have very high-speed network connections. In a peer-to-peer environment these characteristics may not hold, leading to bad distribution trees. To cope with this problem, Castro et al. propose to strip the content and distribute each stripe through a different tree. These trees should form an interior-node-disjoint forest. SplitStream is aimed at peer-to-peer cooperative environments, as opposed to Overcast<sup>13</sup> which is designed to work on a pre-established service overlay. If appropriate encoding is used, peers can decode the data even if a stripe is lost (maybe with a penalty of lower video/audio quality at most). How to stripe and encode the data is left to the application. The expected amount of state maintained by each node is  $O(\log N)$ ; the expected number of messages to build the forest is  $O(N \log N)$  and  $O(N^2)$  in the worst case.

The paper describes how to implement SplitStream over Scribe.<sup>23</sup> The groupIds are chosen to differ in the most significant digit, ensuring that trees have a disjoint set of interior nodes. To satisfy the nodes constraints on outbound bandwidth, nodes do not reject children once they have reached their out-degree limit, but rather adopt the new child and then proceed to orphan one of their children. A child that does not share a prefix with the local node nodeId is selected. If no such child exists, the child with the shortest prefix match for that stripeId is selected. The orphaned child then tries to attach to a random former sibling from those that share a prefix match with the stripeId for which it seeks a parent. If a node cannot find a parent, it chooses one from the spare capacity group, a group that contains the nodes that have less children than their capacity limit. To avoid forming cycles each node maintains its path to the root of each stripe it receives.

The results presented show that the maximum node stress is smaller than if a centralized server is used, and that the overhead is low, even with high churn.

### 2.12. Lightweight probabilistic broadcast (lpbcast)

Gossip based broadcast protocols were first proposed by Demers et al.<sup>27</sup> for distributed databases as an efficient and scalable way for disseminating replication information. Lpbcast<sup>16</sup> is a gossip-based decentralized probabilistic broadcast protocol designed for event dissemination. Gossiping leverages reliability and scalability, and their results show that the scheme works well even when each peer has only a partial view of the overlay. Eugster et al. propose the use of gossiping to disseminate membership information and limit the amount of data disseminated and stored. Peers do not know all the other members but rather have a limited view of the state of the overlay. This allows for increased scalability at a cost of only being able to provide probabilistic reliability guarantees.

Each member maintains a partial view of the group members, a list of unsubscriptions and a list of subscriptions. Each of these lists has a maximum size limit. These last two lists are forwarded to random peers on a periodic basis to disseminate membership information and allow other members to update their views. When a member receives a gossip message, it first handles the unsubscriptions piggybacked in the message by removing unsubscribed peers from its view and adding them to the unSubs list. If unSubs exceeds its maximum limit, it removes a random entry. Then subscriptions are handled by adding (or not, depending on a fixed probability) peers not present in the current view (and subs list) and removing other peers if necessary. If the message contains new data (called notifications in publish/subscribe), it is delivered to the application. Each member periodically generates a new gossip message, and gossips it to  $F$  other random members.

To join, a new member must know some current group member and send a subscription message to it. The peer contacted gossips this new subscription information and the new member starts receiving gossip. Unsubscriptions are handled similarly.

The Microsoft Research group that proposed lpbcast has published several other papers describing improvements and variations of the gossip-based multicast and membership management protocol.<sup>28-30</sup> Gupta et al.<sup>28</sup> describe a hierarchical gossip-based multicast protocol that achieves better scalability by reducing the load imposed on the network. SCAMP<sup>29</sup> is a peer-to-peer membership management protocol for gossip-based protocols that provides each member with a partial view of the group membership. The latest paper of the set<sup>30</sup> provides in depth theoretical analysis of gossip-based protocols.

### 2.13. HostCast

HostCast<sup>31</sup> proposes adding links to grandparents and uncles in the overlay (source rooted) tree to form a richer mesh that can be used to quickly change parents, merge partitions or improve performance. Members maintain information of their primary and secondary neighbors and on their rootpath. The root periodically sends control probes through the mesh. These control messages carry timing information that can be used by every node to evaluate if a secondary path to the root provides improved performance over the primary (tree) delivery path. For this distributed probing mechanism to work, the members of the group have to be time synchronized. The max number of children (fanout) is configurable. The soft-state membership requires periodic refresh messages to prove members are alive. To avoid loops in the tree each member keeps a list of the members in its primary root path. When switching to a new parent for improved performance, there is a period when the node keeps both parents and evaluates them so it can choose the best one.

**Table 1.** Design choices for each of the application level multicast solutions

	Routing	Multicast tree construction	Tree type	Overlay management	Group size	Closeness metric	Delivery guarantees
Narada	Tree	Mesh first	Source sp.	Distributed	Small	Latency	AMO
ALMI	Tree	Mesh first	Shared	Centralized	Medium	Latency	AMO
Yoid	Tree	Tree first	Shared	Distributed	Medium	Data Loss	ZOM
NICE	Tree	Implicit	Source sp.	Distributed	Large	Latency	AMO
Bayeux	Tree	Implicit/Tree-first	Source sp.	Distributed	Large	none	AMO
CAN	Intelligent flooding	N/A	N/A	Distributed	Large	Latency	ZOM
Scribe	Tree	Similar to RPF	Shared source sp.	Distributed	Large	Latency	AMO
SplitStream	Multiple trees	Any	Source sp.	Distributed	Large	Latency	AMO
Bullet	Mesh	Any	Source Sp.	Distributed	Large	E2E bandwidth	ZOM
Lpbcast	Random flooding	N/A	N/A	Distributed	Large	none	ZOM
BTP	Tree	Tree first	Shared tree	Distributed	Medium	Latency	AMO
Overcast	Tree	Tree first	Shared	Distributed	Not comparable	10KB download time & traceroute distance	AMO
HostCast	Tree	Tree first	Source sp.	Distributed	Medium	Available bandwidth & root-path latency	AMO

## 2.14. Other

There are several other schemes for application-layer multicasting not included in this paper. oStream<sup>32</sup> is an application-layer overlay designed for on-demand media distribution. Zhu et al.<sup>33</sup> propose an application-layer multicast protocol that uses network coding that uses a 2-redundant multicast graph (a directed acyclic graph) as its topology. Zigzag<sup>34</sup> is an end system multicast infrastructure designed specifically for media streaming; in it, peers are clustered in a hierarchy and the multicast distribution tree is built on top of the hierarchical overlay. Liebeherr et al. have explored hypercubes<sup>35</sup> and delaunay triangulations<sup>36</sup> as overlay network topologies for application layer multicast. Scattercast<sup>37</sup> designed by Yatin Chawathe is a hybrid overlay multicast architecture in which Scattercast proxies form an application-level overlay that interconnects locally-scoped multicast regions, specifically designed for Internet live broadcasting (streaming). Gossamer is the name of the overlay protocol for grouping and communication among the proxies. Also by Chawathe et al., are RMX,<sup>38</sup> a reliable multicast for heterogeneous networks and “Broadcast Federation”,<sup>39</sup> an application-layer internetwork to provide end-to-end broadcast service. RelayCast<sup>40</sup> is a middleware for application-level multicast services.

## 3. CLASSIFICATION OF THE DESCRIBED SCHEMES

Table 1 shows how the different application level multicast solutions differ from each other with respect to the most important design choices.

The *routing* column refers to as how messages are routed or forwarded across the overlay network. The *multicast tree construction*, when applicable, differentiates schemes depending on if they build an overlay mesh first and then build a tree on top of it or if they build a tree and then add extra links to form a mesh. The implicit approach never builds a tree but it is implicitly determined by the routing rules. When source specific trees are used, if more than one sender is present, one needs to act as a relay and forward all packages or multiple trees need to be constructed, one for every sender. The *closeness metric* is the metric used to improve the quality of the overlay mesh. Latency, also called delay, is the most common used one and is usually obtained by actively probing peers. The *delivery guarantees* can be classified into four: at-most-once (AMO), zero-or-more (ZOM), exactly-once (EO), and one-or-more (OOM).

As can be observed in Table 1, the schemes surveyed differ from each other in several aspects. Understanding these differences will aid in the selection of an appropriate scheme for the different real-life multicasting needs.

## 4. CHALLENGES

### 4.1. Scalability

For many applications, a multicasting solution for small to medium-sized groups is enough. But some others, e.g. real-time news or stock tickers, a solution that scales to large groups is needed. Although some of the proposed end system multicast solutions provide more or less scalable solutions, more extensive simulations as well as real-life tests are needed to better understand their properties.

A multicasting solution can be un-scalable for several reasons, including the amount of state maintained at each node and the overhead due to control messages. Probing to improve the quality of the overlay network can become overwhelming as the overlay grows. Nakao et al.<sup>41</sup> propose the creation of a shared routing underlay

that different overlays can query to obtain specific information such as delay and bandwidth for different routes to aid in the construction of overlays. As different overlays would share this information that otherwise would have to be obtained independently by each probing is reduced.

## 4.2. Fault tolerance

In a distributed system end hosts have a lot of responsibilities but are prone to sudden failures. Any application-layer multicast solution must be fault tolerant. But being able to recover from failure is not the only issue, but also how it is able to do it. Recovery should be distributed, fast and should not require a large number of messages. In case of failure, the communication should be able to continue between the hosts that remain up. The group should be able to graciously recover from partitions and quickly converge to an efficient overlay.

## 4.3. Performance

The overhead of the multicast protocol should not degrade the performance of applications, in terms of throughput and delay. Using some “closeness” metric such as delay, bottleneck bandwidth, etc. to improve the quality of overlay networks is definitely needed in large-scale overlay networks to improve the performance by avoiding unnecessary network congestion.

## 4.4. Quality of Service

Some applications such as video streaming need guarantees about the delivery of its packets (in terms of bandwidth and delay). Some applications need loose guarantees while others have stricter requirements. Recent overlays have considered high bandwidth applications, but more research needs to be done before we can seamlessly distribute streaming media at the application layer.

## 4.5. Security

Current efforts in end system multicast solutions have yet to consider the security issues that arise from their use. Dondeti et al.<sup>42</sup> identify the following scalable secure multicast issues: group membership control, reliability, scalability, perfect forward secrecy, policy and mechanism separation, and secure key distribution schemes that do not trust third party hosts with keys.

### 4.5.1. Confidentiality

Only authorized end hosts should be able to read the multicast messages (*secrecy*). An encryption scheme can be used to solve this. But the problems are the complexity added by encrypting messages and efficient key distribution. Both issues have to be thoroughly analyzed to make this feasible.

There are no key distribution studies for end system multicast. Several key distribution protocols for IP multicast<sup>43,44</sup> have been proposed, but further study is needed to determine if they are appropriate for end system multicast or if new schemes are needed. While some have proposed centralized key management, for scalability issues, most recent work propose hierarchical schemes.

Some solutions<sup>45–47</sup> allow rekeying of the multicast group. To provide forward and backward secrecy<sup>†</sup>, the group should be rekeyed on membership changes. This is a potential bottleneck that affects scalability if membership changes are frequent. Kronos<sup>48</sup> and the work of Li et al.<sup>47</sup> suggest decoupling group rekeying from membership changes. As they point out, if membership changes are done periodically, the frequency of membership changes will not incur in additional overhead. The problem with this alternative, is that rekeying should be very frequent (in<sup>48</sup> it is suggested that the key should be renewed every second) to be able to provide good backward secrecy guarantees.

Group communication systems are usually small (less than 100 members) that usually aim at providing fault tolerance. While multicast usually refers to a best-effort service, group communication systems implement reliable multicast services. Rodeh et al.<sup>49</sup> describe and analyze three different rekeying schemes for group communication

---

<sup>†</sup>For backward secrecy, new members of the group should not be able to read messages multicasted before they joined. For forward secrecy, members that leave the group should not be able to understand messages multicasted after they left.

systems. Their proposed solutions use unicast messaging and do not impose a multicast tree or hierarchy, thus it is straightforward to use their proposed schemes on small-scale end system multicast solutions.

Identity confidentiality or *anonymity* may be an issue for some applications. Several projects have studied anonymity in peer-to-peer (P2P) networks.<sup>50,51</sup>

#### 4.5.2. Integrity

End hosts must have a way of knowing that the messages they receive have not been altered by an external (or internal) host. The protocol must be designed in such way that man-in-the-middle and replay attacks are not possible or are easily detected. Source-based encryption or a signature scheme can be used to prevent malicious users from modifying messages. To avoid a replay attack a nonce can be included in the message.

#### 4.5.3. Availability

The multicast service must have a high availability. This includes fault tolerance issues (see Sect. 4.2) as well as some other security related issues as denial of service (DoS) avoidance and ability to communicate even if a host has been compromised. To avoid denial of service attacks, the protocol must be as distributed as possible. In an application-layer solution it is very hard or even impossible to avoid the use of some kind of rendezvous point (RP), which can be prone to a denial of service attack. A possible solution suggested by Jannotti et al.<sup>13</sup> is to replicate the RP. Minimizing the responsibilities of the RP also helps. Another availability problem may arise if a participating host is compromised. In such case, the group should be able to continue the communication but excluding the compromised host from it. The possibility of malicious hosts participating in the communication could also be a problem. What if a host purposely does not follow the multicast protocol? Its peers should be able to notice the problem and recover from it and expel the malicious host from the group.

The Sybil attack<sup>52</sup> is particular to structured P2P systems: a single faulty peer with multiple identities in the P2P system can control a substantial fraction of it and perform attacks that would otherwise require multiple compromised peers. The solution proposed in the paper<sup>52</sup> is to have a trusted agent to certify identities.

Castro et al.<sup>53</sup> address the problem of improving availability in structured peer-to-peer overlays. They study attacks aimed at preventing correct message delivery in such networks and, describe and evaluate techniques to allow secure message forwarding in presence of malicious nodes.

## 5. APPLICATIONS

Table 2 shows several applications types and some of their multicast characteristics. As can be observed, different applications have different requirements. Thus, there is no *one-size-fits-all* solution, and the end system multicast scheme to use must be carefully selected. It is important to note that none of the schemes provide support for secure communications.

Network-centric warfare is not simply the combination of communication, intelligence, and signals but rather warfare that leverages off a common network developed to support different purposes. For example, an interlinked grid where sensors gather data, command and control systems transform this data to knowledge, and shooters then apply power - all based off the same network. Network-centric warfare saw action in Iraq where constant multi-layer overhead sensors (aircraft) were simultaneously linked to command-and-control and shooters resulting in quicker decision making and action. While none of these end system multicast schemes were designed with network-centric applications in mind, they each may be appropriate under certain scenarios. While video-conferencing, media distribution, and multi-player gaming are close analogies to envisioned network-centric applications, the most important military requirements for survivability and security may force redesign and ultimately the failure of currently proposed research end system multicast systems and thus the need for new end system multicast systems to be designed with network-centric applications as part of the initial requirements process -this is the area we intend to investigate in future study.

**Table 2.** Applications and their characteristics

Application	Source	Traffic	Group Size	Membership change rate
Whiteboard	Multi	Small	Small	Low
Multi-player games	Multi	Medium	Medium-High	Medium-High
News/Stock ticker	Single	Small	High	High
Distributed simulations	Multi/Single	Large	Medium	Medium-High
Small scale video/audio streaming	Single	Large	Small	Low-Medium
Corporate/campus-wide media streaming	Single	Large	Medium	Medium
Internet-wide video/audio streaming	Single	Large	Large	High
Media distribution	Single	Large	Large	High
Video-conferencing	Multi	Large	Small	Small

## 6. CONCLUSIONS

In this paper we have described and classified all the current state-of-the-art in end system multicast solutions. Improving the understanding of these solutions is crucial for making decisions on which to use depending on the characteristics of each application. We have also described the challenges in designing end system multicast schemes and identified several areas that need further study.

## ACKNOWLEDGMENTS

This research is funded in part by a grant from the Office of Naval Research (ONR) within the National Center for Advanced Secure Systems Research (NCASSR) <[www.ncassr.org](http://www.ncassr.org)>. We thank the members of the multicast security and survivability project and colleagues at UIUC who helped and gave valuable input for the writing of this paper: Raquel Hill, Ahmed Sobeih, Jun Wang, Jennifer Hou and Klara Nahrstedt. We also thank Brian Adamson and Joe Macker from the Naval Research Laboratory (NRL).

## REFERENCES

1. S. E. Deering and D. R. Cheriton, "Multicast routing in datagram internetworks and extended LANs," *ACM Trans. on Computer Sys* **8**, pp. 85–110, May 1990.
2. M. H. Ammar, "Why johnny can't multicast: Lessons about the evolution of the Internet." *13th Intl. Workshop on Network and Operating Sys. Support for Digital Audio and Video (NOSSDAV 2003)*, Keynote Speaker. Slides available at: <<http://www.cc.gatech.edu/fac/Mostafa.Ammar/nossdav-key.ppt>>.
3. D. Palter, "Multicast fan-out saves bandwidth," *Network World*, Sept. 2002. 09/30/02.
4. S. Floyd, V. Jacobson, S. McCanne, and L. Zhang, "A reliable multicast framework for light-weight sessions and application level framing," *IEEE/ACM Transactions on Networking* **5**, pp. 784–803, Dec. 1997.
5. J. C.-H. Lin and S. Paul, "RMTP: A reliable multicast transport protocol," in *Proc. of the 15th Joint Conf. of the IEEE Computer and Communications Societies (INFOCOM 1996)*, Mar. 1996.
6. Y.-H. Chu, S. G. Rao, S. Seshan, and H. Zhang, "A case for end system multicast," in *IEEE Jrnl. on Selected Areas in Communications (J-SAC), Sp. Issue on Network Support for Group Communication*, 2003.
7. D. Pendarakis, S. Shi, D. Verma, and M. Waldvogel, "ALMI: An application level multicast infrastructure," in *Proc. of the 3rd Usenix Symp. on Internet Technologies and Sys. (USITS 2001)*, Mar. 2001.
8. P. Francis, Y. Pryadkin, P. Radoslavov, R. Govindan, and B. Lindell, "YOID: Your Own Internet Distribution." Work in Progress.
9. S. Banerjee, B. Bhattacharjee, and C. Kommareddy, "Scalable application layer multicast," in *Proc. of ACM SIGCOMM 2002*, Aug. 2002.
10. S. Q. Zhuang, B. Y. Zhao, A. D. Joseph, R. Katz, and J. Kubiawicz, "Bayeux: An architecture for scalable and fault-tolerant wide-area data dissemination," in *Proc. of the 11th Intl. Workshop on Network and Operating Sys. Support for Digital Audio and Video (NOSSDAV 2001)*, June 2001.
11. D. A. Helder and S. Jamin, "End-host multicast communication using switch-tree protocols," in *Proc. of the 2nd Workshop on Global and Peer-to-Peer Computing on Large Scale Distributed Sys. (GP2PC 2002)*, May 2002.

12. S. Ratnasamy, M. Handley, R. Karp, and S. Shenker, "Application-level multicast using content-addressable networks," in *Proc. of 3rd Intl. Workshop on Networked Group Communication (NGC 2002)*, Nov. 2001.
13. J. Jannotti, D. K. Gifford, K. L. Johnson, M. F. Kaashoek, and J. W. O'Toole, Jr., "Overcast: Reliable multicasting with an overlay network," in *Proc. of the 4th Usenix Symp. on Operating Sys. Design and Implementation (OSDI 2000)*, Oct. 2000.
14. D. Kostic, A. Rodriguez, J. Albrecht, and A. Vahdat, "Bullet: High bandwidth data dissemination using an overlay mesh," in *Proc. of the 20th ACM Symp. on Operating Sys. Principles (SOSP 2003)*, Oct. 2003.
15. M. Castro, P. Druschel, A.-M. Kermarrec, A. Nandi, A. Rowstron, and A. Singh, "Splitstream: High-bandwidth multicast in cooperative environments," in *Proc. of the 20th ACM Symp. on Operating Sys. Principles (SOSP 2003)*, Oct. 2003.
16. P. Eugster, S. Handurukande, R. Guerraoui, A.-M. Kermarrec, and P. Kouznetsov, "Lightweight probabilistic broadcast," in *Proc. of the Intl. Conf. on Dependable Sys. and Networks (DSN 2001)*, July 2001.
17. Y. Chu, S. G. Rao, S. Seshan, and H. Zhang, "Enabling conferencing applications on the internet using an overlay multicast architecture," in *Proc. of ACM SIGCOMM 2001*, Aug. 2001.
18. B. Y. Zhao, L. Huang, J. Stribling, S. C. Rhea, A. D. Joseph, and J. D. Kubiatowicz, "Tapestry: A global-scale overlay for rapid service deployment," *IEEE Jnl. on Selected Areas in Communications (J-SAC)*, *Sp. Issue on Recent advances in Service Overlay Networks* **22**, Jan. 2004.
19. "Jungle monkey, a distributed file sharing system." Available at <http://www.junglemonkey.net/>.
20. S. Jamin, C. Jin, Y. Jin, D. Raz, Y. Shavitt, and L. Zhang, "On the placement of Internet instrumentation," in *Proc. of the 19th Joint Conf. of the IEEE Computer and Communications Societies (INFOCOM 2000)*, Mar. 2000.
21. S. Ratnasamy, P. Francis, M. Handley, R. Karp, and S. Shenker, "A scalable content-addressable network," in *Proc. of ACM SIGCOMM 2001*, Aug. 2001.
22. S. Ratnasamy, M. Handley, R. Karp, and S. Shenker, "Topologically-aware overlay construction and server selection," in *Proc. of the 21st Joint Conf. of the IEEE Computer and Communications Societies (INFOCOM 2002)*, June 2002.
23. M. Castro, P. Druschel, A.-M. Kermarrec, and A. Rowstron, "SCRIBE: A large-scale and decentralized application-level multicast infrastructure," *IEEE Jnl. on Selected Areas in Communications (J-SAC)*, *Sp. Issue on Network Support for Group Communication* **20**, Oct. 2002.
24. A. Rowstron and P. Druschel, "Pastry: Scalable, distributed object location and routing for large-scale peer-to-peer systems," in *Proc. of the ACM/IFIP/USENIX Intl. Middleware Conf.*, Nov. 2001.
25. Y. K. Dalal and R. Metcalfe, "Reverse path forwarding of broadcast packets," *Communications of the ACM* **21**, pp. 1040–1048, Dec. 1978.
26. S. Floyd, V. Jacobson, C.-G. Liu, S. McCanne, and L. Zhang, "Equation-based congestion control for unicast applications," in *Proc. of ACM SIGCOMM 2000*, Aug. 2000.
27. A. J. Demers, D. H. Greene, C. Hauser, W. Irish, J. Larson, S. Shenker, H. E. Sturgis, D. C. Swinehart, and D. B. Terry, "Epidemic algorithms for replicated database maintenance," in *Proc. 6th ACM Symp. on Principles of Distributed Computing (PODC 1987)*, Aug. 1987.
28. I. Gupta, A.-M. Kermarrec, and A. J. Ganesh, "Adaptive and efficient epidemic-style protocols for reliable and scalable multicast," Oct. 2002.
29. A. J. Ganesh, A.-M. Kermarrec, and L. Massoulié, "Peer-to-peer membership management for gossip-based protocols," *IEEE Transactions on computers* **52**, Feb. 2003.
30. A.-M. Kermarrec, L. Massoulié, and A. J. Ganesh, "Probabilistic reliable dissemination in large-scale systems," *IEEE Trans. on Parallel and Distributed Sys* **14**, Mar. 2003.
31. Z. Li and P. Mohapatra, "Hostcast: A new overlay multicasting protocol," in *Proc. IEEE 2003 Intl. Conf. on Communications (ICC 2003)*, May 2003.
32. Y. Cui, B. Li, and K. Nahrstedt, "oStream: Asynchronous streaming multicast in application-layer overlay networks," *IEEE Jnl. on Selected Areas in Communications (J-SAC)*, *Sp. Issue on Recent advances in Service Overlay Networks* **22**, Jan. 2004.

33. Y. Zhu, B. Li, and J. Guo, "Multicast with network coding in application-layer overlay networks," *IEEE Jnl. on Selected Areas in Communications (J-SAC), Sp. Issue on Recent advances in Service Overlay Networks* **22**, Jan. 2004.
34. D. A. Tran, K. A. Hua, and T. T. Do, "A peer-to-peer architecture for media streaming," *IEEE Jnl. on Selected Areas in Communications (J-SAC), Sp. Issue on Recent advances in Service Overlay Networks* **22**, Jan. 2004.
35. J. Liebeherr and T. K. Beam, "HyperCast: A protocol for maintaining multicast group members in a logical hypercube topology," in *Proc. of 1st Intl. Workshop on Networked Group Communication (NGC 1999)*, July 1999. Published in *Lecture Notes in Computer Science* **1736**.
36. J. Liebeherr and M. Nahas, "Application-layer multicast with delaunay triangulations," in *Proc. of the 6th Global Internet Symp. (IEEE GLOBECOM 2001)*, Nov. 2001.
37. Y. Chawathe, "Scattercast: An adaptable broadcast distribution framework," *ACM Multimedia Sys* **9**, pp. 104–118, July 2003.
38. Y. Chawathe, S. McCanne, and E. Brewer, "RMX: Reliable multicast in heterogeneous environments," in *Proc. of the 19th Joint Conf. of the IEEE Computer and Communications Societies (INFOCOM 2000)*, Mar. 2000.
39. Y. C. M. Seshadri, "Broadcast federation: An application-layer broadcast internetwork," in *Proc. of the 12th Intl. Workshop on Network and Operating Sys. Support for Digital Audio and Video (NOSSDAV 2002)*, May 2002.
40. N. Mimura, K. Nakauchi, H. Morikawa, and T. Aoyama, "RelayCast: A middleware for application-level multicast services," in *Proc. of the 3rd Workshop on Global and Peer-to-Peer Computing on Large Scale Distributed Sys. (GP2PC 2003)*, May 2003.
41. A. Nakao, L. Peterson, and A. Bavier, "A routing underlay for overlay networks," in *Proc. of ACM SIGCOMM 2003*, Aug. 2003.
42. L. R. Dondeti, S. Mukherjee, and A. Samal, "Survey and comparison of secure group communication protocols," Technical Report, University of Nebraska-Lincoln, 1999.
43. A. T. Sherman and D. A. McGrew, "Key establishment in large dynamic groups using one-way function trees," *IEEE Trans. on Software Engineering* **29**, May 2003.
44. S. Mittra, "Iolus: A framework for scalable secure multicasting," in *Proc. of ACM SIGCOMM 1997*, 1997.
45. M. Waldvogel, G. Caronni, D. Sun, N. Weiler, and B. Plattner, "The VersaKey framework: Versatile group key management," *IEEE journal on Selected Areas in Communications* **17**, Sept. 1999.
46. N. Weiler, "Semsomm - a scalable multiple encryption scheme for one-to-many multicast," in *Proc. of the IEEE 10th Intl. Workshop on Enabling Technologies: Infrastructure for Collaborative Enterprises (WETICE 2001)*, June 2001.
47. X. S. Li, Y. R. Yang, M. G. Gouda, and S. S. Lam, "Batch rekeying for secure group communications," in *Proc. of the 10th Intl. World Wide Web Conf. (WWW10)*, May 2001.
48. S. Setia, S. Koussih, S. Jajodia, and E. Harder, "Kronos: A scalable group re-keying approach for secure multicast," in *Proc. of the IEEE Symp. on Security and Privacy (S&P 2000)*, May 2000.
49. O. Rodeh, K. P. Birman, and D. Dolev, "A study of group rekeying," Technical Report TR2000-1791, Department of Computer Science, Cornell University, Mar. 2000.
50. M. Waldman and D. Mazi, "Tangler: A censorship-resistant publishing system based on document entanglements," in *Proc. of the 8th ACM Conf. on Computer and Communications Security (CCS 2001)*, Nov. 2001.
51. M. J. Freedman and R. Morris, "Tarzan: A peer-to-peer anonymizing network layer," in *Proc. of the 9th ACM Conf. on Computer and Communications Security (CCS 2002)*, Nov. 2002.
52. J. R. Douceur, "The Sybil attack," in *Proc. of the 1st Intl. Workshop on Peer-to-Peer Sys. (IPTPS 2002)*, Mar. 2002.
53. M. Castro, P. Druschel, A. Ganesh, A. Rowstron, and D. S. Wallach, "Secure routing for structured peer-to-peer overlay networks," in *Proc. of the 5th Usenix Symp. on Operating Sys. Design and Implementation (OSDI 2002)*, Dec. 2002.