Chapter 1
Application Layer Multicast

Mouna Allani, University of Lausanne
Benoît Garbinato, University of Lausanne
Fernando Pedone, University of Lugano

1.1 Introduction

An increasing number of Peer-to-Peer (P2P) Internet applications rely today on data dissemination as their cornerstone, e.g., audio or video streaming, multi-party games. These applications typically depend on some support for multicast communication, where peers interested in a given data stream can join a corresponding *multicast group*. As a consequence, the efficiency, scalability, and reliability guarantees of these applications are tightly coupled with that of the underlying multicast mechanism.

At the network level, IP Multicast offers quite good efficiency and scalability but best-effort reliability only. Moreover, the deployment of IP Multicast requires all routers to be appropriately configured, which makes it quite impractical in large scale or open settings, i.e., where one does not have full control over the networking environment. For this reason, several research directions have focused on trying to offer some form of multicast communication at a higher level, typically via an application-oriented middleware. Such a higher level data dissemination support is often referred to as *Application Layer Multicast* (ALM). Of course, ALM-based solutions to data dissemination usually rely on the lower level networking mechanisms, as suggested in Figure 1.1.

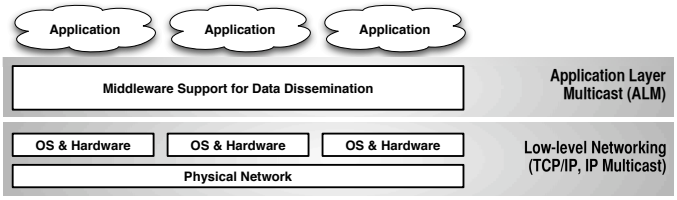

Fig. 1.1 Multicast support for data dissemination

1.1.1 From Peer-to-Peer to MANETs

Recently, mobile ad hoc networks (MANETs) have gained remarkable interest and popularity. By definition, a MANET represents a collection of wireless mobile nodes dynamically formed with the motivation that users can benefit from collaborating with each other. P2P and MANET applications share similar requirements; at the heart of these requirements, one usually find some form of reliable multicast.

While largely developed independently of each other, P2P and MANET applications share many properties: (1) Both P2P and MANET are self-organized and decentralized networks, relying on distributed components with no specific role. (2) They are both exposed to high *churn*, due to nodes frequently joining and leaving the network and, in the case of MANETs, to the mobility of the nodes. Given the similarities of both environments, their dedicated multicast protocols face similar challenges (e.g., coping with frequently changing topologies). We argue that due to their nature, multicast solutions designed to MANETs could in general be mapped to P2P. As we will discuss, the reverse is less straightforward as MANET components are usually limited on resources compared to P2P Internet components.

1.1.2 *Probabilistic* vs. *Structured Approaches*

Roughly speaking, data dissemination solutions can be categorized as following either a *probabilistic* approach or a *structured* approach. The probabilistic approach usually relies on a gossiping protocol, which consists in having each peer forward the data it receives to a set of randomly chosen neighbors. As a consequence, the path followed by the disseminated data is not deterministic. By contrast, the structured approach consists in first organizing the network peers into some overlay network and in routing disseminated data through this virtual topology.

The tradeoff between these two approaches is basically *scalability* vs. *resource consumption*. Structured dissemination uses fewer messages than probabilistic dissemination to ensure the same reliability. The reliability of a probabilistic approach hinges on the redundancy of sent messages; its scalability results from the fact that no overlay creation or maintenance is required. In this chapter, we focus on the structured approach, i.e., on multicast protocols based on an overlay network to disseminate data.

1.1.3 Content and Roadmap

In recent years, a plethora of structured multicast protocols have been proposed and there exists many different ways to classify them. Some of the protocols we survey were designed for MANETs and most of them were originally designed for Internet

P2P applications. It turns out that some P2P protocols could be used in MANETs; we discuss how applications in MANETs can benefit from them.

We characterize the proposed protocols using a small set of dimensions we consider the most relevant. To structure our discussion, we group these dimensions into three *viewpoints*, depending on their *applicative* focus, their underlying *architecture* or their *Quality of Service*. These three viewpoints are the subject of Sects. 1.2 to 1.4, respectively. Sect. 1.5 provides some details about the multicast protocols we survey, using the dimensions defined in the previous sections. Finally, Sect. 1.6 summarizes the findings of this survey and provides a bigger picture on the major trends and possible future research directions of this field.

1.2 Applicative Viewpoint

In this section, we discuss multicast solutions in terms of their *applicative* dimensions, i.e., those characterizing the applicative context in which structured dissemination is to be used. Intuitively, discussing these dimensions boils down to answering the three questions listed below.

- What is the usage focus of the multicast solution?
- What is the communication model it considers?
- Can MANET applications benefit from its use?

1.2.1 Usage Focus

Existing multicast solutions to achieving structured dissemination pursue a wide variety of goals, which are directly related to their main *usage focus*. Roughly speaking, these solutions typically focus either on some specific applicative usage (e.g., asynchronous or on-demand streaming), context-aware data dissemination, or on some system-level contraints. In turn, these contraints are usually expressed in terms of a Quality of Service (QoS) to achieve. Examples of QoS-oriented goals include minimizing latency or maximizing throughput, achieving reliability or scalability, coping with real-time constraints.

1.2.2 Communication Model

Another key applicative dimension to characterize multicast solutions is their communication model. Here we mainly distinguish between the *specific-source* communication model and the *multi-source* communication model. In a multi-source solution, the same overlay network can be used by *any peer*, while in a specific-source solution, only *one peer* can use a given overlay network.

When considering tree-based overlays for instance,[1] specific-source solutions create a separate and dedicated tree for each peer that needs to disseminate data. Therefore, these solutions tend to offer a more efficient multicast protocol than multi-source solutions. As the number of sources increases however, the overhead induced by creating and maintaining multiple trees grows. So multi-source solutions tend to make a better use of the available resources, be it memory, network bandwidth, or computing power.

1.2.3 MANET Suitability

As already suggested in Sect. 1.1, a large number of multicast solutions were devised to support P2P interactions on the Internet, out of which many, yet not all, can be transposed to MANETs. This is no surprise since interactions in a MANET are inherently peer-to-peer due to the absence of an infrastructure. For this reason, we characterize the multicast solutions discussed in this chapter using the notion of *MANET suitability* to indicate whether a multicast solution that was designed for P2P can be adapted to MANETs. In doing so, we restrict our discussion to how messages are routed. The question of how members join or leave a given multicast group is out of the scope of this survey.

Indeed, in spite of their similarities, P2P and MANET contexts tend to impose very different constraints when it comes to joining groups.[2] For example, several Internet-based P2P multicast solutions rely on a predefined set of fixed peers, known as *rendezvous points*, in order to join a multicast group. In the context of MANETs, this approach is simply impossible since by definition a MANET has no predefined set of members. With respect to how messages are routed, we rely on three criteria to decide whether a routing protocol can be used in the context of MANETs. If one or more of these criteria are not satisfied, we consider the multicast solution as not well suited for MANETs. These criteria are describer hereafter.

Fully Peer-to-Peer. This criterion states that the routing protocol must strictly assume a *peer-to-peer communication model*, i.e., it cannot assume the existence of dedicated routing nodes. Furthermore, any routing solution that relies on logical IDs associated to nodes, e.g., obtained by hashing their IP addresses, is not suitable for MANETs. Intuitively, this criterion reflects the fact that MANETs do not rely on a predefined networking infrastructure and thus cannot assume the existence of a predefined set of nodes with known IDs. The notion of node identification, with its logical and physical variants, are further discussed in Section 1.3.1.

Dynamically Adaptable. This criterion implies that the protocol must be *self-configurable* and *self-improving*, i.e., it must be able to cope with a dynamically

[1] Sect. 1.3.1 discusses tree-based overlays in detail.

[2] Discovering and joining multicast groups is a particular case of a more general problem known as *service discovery*, which is extensively discussed in Chapter **??**.

changing environment. In particular, the overlay structure it uses to route messages must be readjusted continuously as the network changes. The notions of self-configurable and self-improving networks are further discussed in Section 1.4.1.

Resource Friendly. This criterion captures the fact that mobile nodes are usually *resource constrained*, i.e., they have limited memory, processing and battery power and must use low bandwidth networks. For instance, protocols that minimize message forwarding help better share the limited wireless bandwidth of MANETs.

1.3 Architectural Dimensions

In this section, we discuss multicast solutions supporting structured dissemination in terms of their *architectural* dimensions. Intuitively, discussing these dimensions boils down to answering the four questions listed below.

- What is the topology of the overlay network?
- How is the overlay network created and maintained?
- Does the proposed solution rely on some infrastructure support?
- How is the networking environment modeled by the proposed solution?

1.3.1 Overlay Structure

As suggested in Sect. 1.1, structured dissemination usually implies the existence of some overlay network in charge of routing and delivering the data. The creation and maintenance of such overlays basically depends on two aspects: (1) how nodes are identified and (2) what is the topology of the overlay.

Node Identification

To define an overlay including all nodes in a multicast group, some approaches designate nodes using *physical IDs*, whereas other approaches rely on *logical IDs*.

Physical IDs. An overlay network based on physical IDs tends to match the underlying physical network and to better account for the resources and contraints of peers and links. With this approach, an *explicit overlay network* must be created prior to performing any data dissemination, usually adopting either a tree-based or a mesh-based topology. Multicast solutions based on physical IDs can often be adapted to MANETs precisely because they do not try to hide the physical contraints imposed by the underlying network.

Logical IDs. A logical ID may reflect some physical properties of its corresponding node (e.g., nodes proximity) or on the contrary it may be completely agnostic of

the underlying physical network. The actual structure of the overlay network then depends on the motivation to use logical IDs. When the main goal is to achieve scalability and to minimize the maintenance overhead, the overlay structure is undefined and implicitly formed by the routing mechanism performed on logical IDs. Routing schemes based on distributed hash tables are an example of such implicit overlay structures. That is, logical IDs implicitly embed a notion of logical neighborhood, which is then used to route data from one peer to another. As a consequence, the multicast protocol does not need to explicitly create or maintain an overlay network, which tends to make it more scalable. When the goal is to balance the load of forwarding data, on the other hand, the overlay structure is explicit and usually follows a well-known topology (e.g., a tree, built on top of logical IDs).

When it comes to MANETs, multicast solutions based on logical IDs are usually harder to apply because they tend to hide important contraints of the underlying physical network. For example, two peers that happen to be located in the same neighborhood at the physical level might be quite distant in terms of logical neighborhood, and vice versa. As a result, latency is difficult to control, since data might have to be routed across many peers, some of which might be slow, and across high-latency links.

Overlay Topology

While a graph topology overlay is often associated to physical ID approaches, some Logical ID approaches build a graph on top of the node abstraction aiming to routing load balancing. The widely used overlay topology is the tree.

Tree-Based topologies. Tree-based overlay networks connect any two peers via a unique path and are acyclic, which greatly simplifies data routing. In addition, the way such networks are constructed tends to make them match the underlying physical network. For example, when the outbound bandwidth of a peer is limited, the number of children of that peer in the tree tends to be proportionally limited as well.[3] On the other hand, tree topologies are very sensitive to failures or partitioning: as soon as a non-leaf peer leaves or crashes, the tree breaks, which prevents members of the partitioned subtree from receiving the disseminated data.

Interestingly, tree-based overlay networks can be used to disseminate data both from a single source or from multiple sources. In addition, some multicast solutions rely on hierarchical clusters of peers to create and maintain various source-specific delivery trees.

Mesh-Based Topologies. Various multicast solutions consist in first building a mesh-based overlay network and then deriving a data dissemination tree on top of that mesh. The mesh-based overlay network is usually constructed using some knowledge about the underlying network, so as to improve overall performance

[3] As we shall see in Sect. 1.5, many different tree construction and maintenance techniques have been proposed, resulting in overlay networks exhibiting various interesting properties.

overlay. Note that it is possible to take advantage of standard routing protocols to construct the data dissemination tree, in which case the routing protocol automatically handles potential loop problem.

Creating and maintaining an intermediate mesh-based overlay has several advantages over directly building a data dissemination tree. For example, a mesh-based overlay better tolerates peer failures than a tree structure because it contains multiple paths from the data source(s) of a multicast group to its members. In addition, the existence of multiple paths in a mesh makes it possible to optimize the overlay, e.g., for partition recovery, load-balancing.

1.3.2 Overlay Creation and Maintenance

The overlay creation and maintenance process can either be *centralized*, in which case it is the responsibility of a single peer, or *distributed*, in which case the responsibility is shared among several peers. Distributed creation and maintenance offers several advantages over a centralized approach. For example, it is more resilient to failures, since the crash of one peer will not prevent the construction and maintenance of the overlay network. It is also more scalable, since this process relies on several peers and thus tends to avoid bottleneck issues.

Building an overlay in a distributed manner, however, induces a potential loss of simplicity and optimality, because no peer has a global view on the constraints of the underlying physical network. Some multicast solutions propose an hybrid approach, based on a controller peer to manage the multicast group, e.g., for joining or leaving the group, while the overlay construction and maintenance is still performed cooperatively.

Some protocols rely on a distributed and *layered* creation and maintenance process. In such a layered architecture, the multicast group is divided into a hierarchy of clusters, each containing a set of peers and one being the leader. This layered architecture is hierarchical in the sense that clusters are grouped into layers, where all leaders of some layer are also (simple) members of a cluster in the higher-level layer. A leader has then the responsibilities to organize members of its cluster and to represent them in the whole system. On the one hand, a layered architecture is scalable and has rather low overhead in terms of network overlay management, since this is done concurrently in each cluster. On the other hand, cluster management induces an additional (but local) overhead. Furthermore, all optimizations are done locally to each cluster, so peers in different clusters are unable to establish overlay links. This may lead to a suboptimal overlay network.

1.3.3 Infrastructure Support

The majority of multicast protocols do not rely on any infrastructure, i.e, they are pure *peer-to-peer* solutions. Nevertheless, some solutions rely on strategically deployed *overlay proxies*, such proxies being typically dedicated to message routing, overlay creation and maintenance, etc.

The advantage of using proxies lies in the fact that they can be deployed on strategic hosts, those that are more reliable or have more resources. One can also make them aware of both low-level and high-level contraints. In the case of on-demand data streaming for example, proxies can arbitrate the various applicative demands, based on their knowledge of the underlying networking constraints. However, proxies may become bottlenecks or single points on failure.

1.3.4 Networking Environment Modeling

In order to create and maintain an overlay network, peers must somehow model their physical networking environments, e.g., in terms of their neighborhood, of the message loss probability for their outgoing links.[4] Usually, members of a multicast group have only a *local knowledge* of their environment, and more seldom a *global knowledge*. Typically, this knowledge is built up using techniques based on some probing interactions between peers.

For multicast solutions, the choice between global knowledge and local knowledge translates into a tension between efficiency and scalability. Indeed, the more a group of peers knows about its networking environment, the more efficient the multicast protocol will be. However, modeling a very large networking environment potentially requires storing a large amount of information, which challenges the scalability of solutions based on global knowledge.

1.4 QoS Viewpoint

In this section, we discuss multicast solutions supporting structured dissemination of data in terms of their *Quality of Service* (QoS). The notion of QoS being very broad, as well as the range of QoS guarantees offered by existing multicast solutions, it is not our intention to provide an exhaustive characterization of those solutions in terms of QoS. Rather, we focus on a limited set of QoS dimensions that we believe are the most relevant to the subject, namely *adaptivity, scalability* and *reliability*.

[4] Of course, this observation does not apply to multicast solutions based on *logical ID* overlays, discussed in Sect. 1.3.1. Indeed, the very nature of *logical ID* overlays is to be decoupled from the underlying physical network.

Intuitively, discussing these dimensions implies answering the three questions listed below.

- How does a multicast solution adapt to changes in its environment?
- How does it scale when the number of peers increases?
- How reliable is the network overlay it creates?

1.4.1 Adaptivity

Adaptivity captures the degree to which a multicast solution adapts to a changing environment, e.g., changes in the underlying physical network, changes in the membership of the multicast group. In terms of adaptability, we distinguish between three degrees of multicast solutions: *static, self-configurable* and *self-improving*.

A static solution typically builds its dissemination overlay once, assuming that all members of the multicast group are known and will not change in the future. On the contrary, a self-configurable solution builds its dissemination overlay incrementally, based on a continuous stream of peers joining and leaving the multicast group. Finally, a self-improving solution continuously improves its dissemination overlay, as ways to make a better usage of the underlying physical network are discovered. Depending on its usage focus, a given self-improving solution will aim at adapting to one or more properties of the physical network, e.g., network latency or bandwidth, links reliability.

In contrast, some solutions relying on a Logical ID overlay network can never exhibit a self-improving behavior, as it does not take the underlying physical network as its starting point. Such solutions are sometimes said to be *network agnostic*.

1.4.2 Scalability

The scalability of a solution depends on several factors, such as the resources required to create and maintain its overlay network, the scope of knowledge each peer needs to maintain in order to model its environment, etc. These factors are almost always directly correlated with the number of peers in the multicast group. For this reason, this survey characterizes the scalability of a multicast solution via the maximum size of multicast groups it can support. Such sizes can either be *small* (tens of members), *medium* (hundreds of member) or *large* (thousands of members).

1.4.3 Reliability

When reliability is considered, the majority of multicast solutions offer a reactive mechanism to recover from peer or link failures. Fewer solutions also rely on a

more proactive strategy, which aims at preventing such failures. A typical prevention mechanism consists in creating a reliable and possibly redundant overlay, which includes the most reliable network components (peers and links). The reliability of components is then usually computed by probing, as discussed in Sect. 1.3.4.

In this survey, we classify the reliability of multicast solutions as *reliable* if the solution provides an explicit failure prevention, as *fault-tolerant* if it simply proposes a mechanism for recovering from peer or link failures, and as *best-effort* if it only relies on the peer-to-peer reliability offered by low-level networking layers (e.g., TCP/IP).

1.5 A Plethora of Protocols

In the following, we illustrate the dimensions discussed in the previous sections by detailing some multicast protocols for structured dissemination. Since they are quite numerous, we structure the presentation by classifying them according to the topology of their overlay network, as discussed in Sect. 1.3.1.

1.5.1 Tree-Based Overlay

ALMA. The Application Level Multicast Architecture (ALMA) [28] targets video streaming across heterogeneous networks. It has three components: a centralized Directory Server (DS), a Web Server (WS), and the client nodes. Data streaming is undertaken over the overlay tree formed by the clients. The DS is responsible for maintaining the tree and does not participate in the actual data delivery, while the WS provides a global user interface for sources to advertise their sessions and for clients to discover the sessions available. Different dedicated trees are created for different sources. A client who wishes to join an advertised session first contacts the WS and chooses the session to join. Next, it issues a join request to the DS which in response sends back a list of potential parents. The client selects the best parent according to some metric such as the round trip time (RTT), measured at the application level. A new branch is then added to the propagation tree and the parent begins streaming data to the new client.

All clients in the tree overlay are organized into levels. Level 1 clients are between the source and the rest of the clients, and therefore are seen as synonymous to the source; clients at higher levels have joined the session at a later time. Every client on the tree monitors RTT and gossips with potential parents from the list initially sent by the DS. A client can decide to switch to another parent if this improves the overall performance. The gossip list of a client is dynamic: if possible parents become incapacitated and the list diminishes below a certain threshold, the client requests a new list from the DS. In order to improve robustness, the tree structure is

enhanced with a simple spiral mechanism that withstands node failures and overlay partitions—as long as consecutive nodes of the same branch do not fail. The spiral mechanism allows a client to maintain a connection with its grandparents in the overlay, making it possible for the client to reconnect to its grandparent if the parent fails, instead of requesting a new list of potential parents from the DS. If consecutive nodes fail, the gossip mechanism will be used by affected clients to switch parents.

ALMI. ALMI [23] is a multicast for interconnected groups of relatively small size (e.g., tens of members). Potential applications include video conferencing, multi-party games, private chat rooms, web cache replication and database/directory replication. End hosts are connected using unicast connections according to a minimum spanning tree built using application-specific performance metrics, such as the round-trip delay between group members.

Every ALMI session consists of a centralized component, the session controller, and session members. The session controller handles member registration and maintains the multicast tree; it is assumed to be accessible by all members. The session controller performs two functions. First, it ensures connectivity of the spanning tree when members join and leave the session or in the presence of network or host failures. Second, it is responsible for periodically recalculating the tree efficiency based on the measurements received from session members. To avoid single points of failure, the protocol can be extended to support multiple back-up session controllers, known to all session members, that operate in stand-by mode and periodically receive the state from the primary controller.

Although the multicast tree is calculated by a central component, and so it should be loop free, tree update messages are disseminated to all session members independently, and can be lost or received out-of-order by different group members, resulting in loops and tree partitions. To prevent such problems, the session controller assigns version numbers to trees. When generating a packet a source includes the latest tree incarnation number in the packet header. Each communication participant caches recent multicast trees. After a packet is received, the receiving node checks whether the tree version is in its cache. If so, it forwards the packet following this tree version; otherwise the packet is discarded. If a member receives a packet with a newer tree version, it contacts the controller to receive a new tree.

BTP. The Banana Tree Protocol (BTP) [12] is a simple multicast for networks of a few hundreds of nodes. BTP is unreliable; it assumes that reliability can be provided by the protocol implemented on top of it. The protocol assumes a bootstrap mechanism that allows multicast nodes to find out about some of the nodes in the system and to connect to them. The first node to join the multicast becomes the tree root. Later nodes join the group by choosing one node to connect to (e.g., the root node). A node is allowed to switch its parent to optimize the tree cost, where the cost of a link is the unicast latency, i.e., the distance between two nodes on the physical network. To avoid loops, a conservative policy limits the switch of a node only to its sibling. Moreover, a node rejects all switching attempts if it is itself executing a parent switching. Tree partitioning is resolved by having a node whose

parent has left the group reconnect to the root node. Thus, BTP is more eligible for systems where nodes do not leave or fail frequently. Two different protocols were built on top of BTP. The Banana Tree Simple Multicast Protocol (BTSMP) is a wrapper around BTP that provides many-to-many group communication for sending and receiving packets. The Banana Tree File Transfer Protocol (BTFTP) provides reliable one-to-many file transfer.

HMTP. The Host Multicast Tree Protocol (HMTP) [29] takes advantage of IP multicast infrastructure to build an overlay multicast layer. Its key design goals are: (1) to be deployable in the Internet; (2) to be compatible with IP Multicast; and (3) to be scalable. To achieve these goals, the proposed protocol automates the interconnection of IP multicast islands, i.e., designated members self-organize into a multicast tree that connects the various IP-multicast islands. The tree structure is revisited periodically to deal with changes in the network topology and the group membership.

The environment modeling in HMTP is rather static, as it assumes the existence of IP multicast islands and the protocol basically consists in bridging these islands. Members of a multicast group can join and leave the group, which triggers an algorithm aimed at repairing the tree. Furthermore, so called designated members, responsible for tunneling requests from one island to another, can also dynamically decide to quit their tunneling role. HMTP periodically reruns the join procedure to accomodate changes in the network and group membership.

MAODV. MAODV [27] is a multicast solution based on the AODV routing protocol designed for MANETs. It builds a multicast tree on-demand, connecting multicast group members. Such a multicast route is discovered on-demand using a broadcast route discovery mechanism. When a node needs a route to a destination, it broadcasts a *Route Request* (RREQ). A node with a current route to that destination or the destination itself unicast a *Route Reply* (RREP) back to the source node. Information gathered through RREQ and RREP messages defines the multicast route.

In MAODV, each process maintains two tables: The *Route Table* is used for recording the next hop for routes to other nodes in the network. This table is filled by adding a new entry at the reception of an RREQ or an RREP message for the first time. Mainly, an entry includes the destination's IP address, its sequence number, and the next hop. The *Multicast Route Table* contains entries to the multicast group the node is the router. New entries are added to this table by the reception of a *Multicast Activation* message (MACT) from the source node that previously asked for a route for a multicast group. A node that broadcasts an RREQ for a multicast group often receives more than one RREP. The source keeps the received route with the greatest sequence number and the shortest hop count to the nearest member of the multicast tree. Then, it unicasts the MACT to this selected next hop. If this node is not a member of the multicast tree, it forwards the MACT to the best next hop from which it received a RREP and enables the corresponding entry in its multicast route table. This process continues until reaching a tree member that already generated a RREP.

The *Multicast Activation* message builds a loop-free multicast tree. Consequently, nodes only forward data packets along activated routes in their multicast route tables.

NICE. NICE [2] is designed to support low-bandwidth data streaming applications with large receiver sets, such as real-time stock updates and quotes. An average member has limited knowledge about other members in the group, i.e., it maintains state for a constant number of other members. NICE creates a hierarchical arrangement of members by assigning them to distinct layers numbered sequentially with the lowest being layer zero. Hosts in each layer are grouped into a set of clusters according to some metric. This hierarchical arrangement is used to define overlay structures for control messages and data delivery paths. The control overlay is a cyclic structure where a member has connections to all peers in the layers it belongs to. The neighbors in the control topology periodically exchange soft state refresh messages. A cyclic structure with high connectivity makes the protocol converge quickly. The data delivery path on the other hand needs to be loop-free to avoid packet duplications.

NICE chooses the data delivery path to be a source-specific tree where in each cluster the topology is a star. The forwarding rule used in NICE enables a source to send a data packet to all its peers in all the clusters it belongs to. A received package will be forwarded further only if the receiving host is the cluster leader. To keep the size of each cluster bounded, whenever it exceeds the upper size limit, the cluster is split into two clusters of the same size; or if the size gets below the bottom limit a merge operation is performed. By having all members send periodic probe messages carrying new information according to the metric used, cluster refinements are enabled. It is possible for members to have distinct inconsistent views of the cluster membership due to update transition periods and for transient cycles to appear on the data path. These cycles are eliminated once the protocol restores the hierarchy invariants and reconciles the cluster view for all members. Finally, the protocol assumes a Rendezvous Point (RP) that all members must be aware of. The RP can be a member at the top layer of the hierarchy or a dedicated host that knows about the member that is the leader of the topmost layer.

OMNI. OMNI [3] targets real-time applications such as media streaming. OMNI consists of a set of service nodes named Multicast Service Nodes (MSNs) and a set of clients, i.e., end-hosts interested in receiving multicast data. MSNs are deployed on strategic positions in the network. They organize themselves in a distributed fashion to form a spanning tree that represents the multicast data delivery backbone. An important property of OMNI is its ability to adapt the overlay tree to changing network conditions and the changing distribution of clients at the different MSNs. The later is conducted by managing the MSNs' position in the overlay depending on their relative importance represented by the number of clients they are responsible for. OMNI strives to minimize the latency of the overlay structure and to bound the out-degrees of the multicast service nodes.

The multicast source is connected to a MSN called the root MSN. The protocol requires each MSN to maintain a limited state information for all its tree neighbors

and ancestors including unicast latency measurements and the overlay path from the root to itself. The initialization of the overlay structure is performed off-line, prior to the data distribution. During this initial phase every participating MSN measures the unicast latency between itself and the root MSN and sends this information to the root. After gathering all the messages from different MSNs, the root orders MSNs according to the latencies. Furthermore, it uses a centralized algorithm for constructing the initial data delivery tree respecting the maximum out-degree requests for each MSN, and distributes it to the MSNs. Constructing the tree that will represent the optimal solution to the minimum overall latency problem with an additional out-degree constraint belongs to the class of NP-hard problems.

Assuming that the out-degree of all MSNs is d, the root first chooses d closest MSNs in terms of the latency as its direct children. The next d closest MSNs are then assigned to be children of the closest child to the root, then the next d closest MSNs are assigned to the second closest child of the root, and so forth. In order to cope with dynamic environments where network conditions may change during time, OMNI proposes periodic transformations to the overlay involving nearby MSNs within two levels of each other. These transformations allow MSNs or clients to swap among themselves to improve tree properties in terms of the overall latency.

oStream. oStream [8] targets the unpredictability of user requests in on-demand media distribution. In particular it addresses *asynchrony*, where users may request the same media object at different times; *non-sequentiality*, where users do not request data sequentially (from beginning to end); and *burstiness*, where the request rate for a certain media object is highly unstable over time. The idea is to establish a minimum spanning tree and use media buffering at the hosts to aid in the distribution of asynchronous service requests for the same streaming media. To do so, oStream defines a *Temporal Dependency Model*, which determines whether user requests overlap, and can therefore be reused; the model also determines the order in which requests should be retrieved in order to maximize reuse.

oStream builds a Media Distribution Graph (MDG) that includes requests as nodes connected with weighted directed links. The weight of a link represents the number of hops between the two end hosts carrying the connected requests. MDG changes dynamically to insert new requests and remove terminated ones. Based on MDG, oStream constructs the Media Distribution Tree (MDT), which is the minimal spanning tree on the overall transmission cost of media distribution. MDT is constructed and maintained incrementally and in a distributed manner. A node (representing a request) that leaves the MDT, first deletes itself and then notifies its children to find a new parent. A node joining the MDT first finds a parent whose transmission cost is minimal and then notifies its successors to consider it as their new parent if it offers a lower cost.

Overcast. Overcast [13] is a multicast solution that builds the overlay tree incrementally. The tree includes the source as the root and dedicated nodes called overcast nodes. Using bandwidth estimation measurements, Overcast organizes the dedicated nodes with the goal of creating high bandwidth channels from the source.

To allow a quick join of clients, Overcast maintains a global status at the root, built from an information table maintained by each node about all nodes lower than itself in the hierarchy and a log of all changes in the table.

The source stores content and schedules it for delivery to the Overcast nodes. A multicast group is represented as an HTTP URL: the hostname portion names the root of an overcast netwok and the path represents a particular group in the network. Typically, once the content is delivered, the publisher at the source generates a web page announcing its availability. The final consumers of the content in an Overcast network are HTTP clients. When a client clicks on the URL for published content, Overcast redirects the request to a nearby Overcast node that serves the content. By representing a multicast group as an HTTP URL, Overcast allows an unmodified HTTP clients to join the multicast group. It also permits the archival of content so that the client can specify a starting time when joining an archived group, such as the beginning of the content.

The overlay tree is continuously reevaluated. Each node periodically reassesses its position in the tree by measuring the bandwidth to its current siblings, parent, and grandparent. As a result, nodes constantly reevaluate their position in the tree and an overcast network is inherently tolerant of nodes failures.

RanSub. RanSub [15] distributes a random subset of the participants to each node; these subsets change periodically and with a uniform representation of all participants. The execution proceeds in epochs, where an epoch consists of a distribution phase, during which data is distributed down an overlay tree, and a collection phase, during which data is aggregated up the tree. In the collection phase each participant propagates to its parent a randomly selected *collect set* containing nodes in the subtree it roots. In the distribution phase each node sends to its children a *distribute set* composed of randomly chosen participants from the collect sets gathered during the previous collect phase and the received distribute set. Once the root receives all collect sets from its children, it launches the distribution phase.

SARO (Scalable Adaptive Randomized Overlay) builds on RanSub to construct adaptive overlays constrained by degree, delay and bandwidth. During each epoch, nodes measure the delay and bandwidth between themselves and all members of their distribute set. Each SARO node A performs probes to members of the distribute set that RanSub transmits to it to determine if a remote node B would deliver better delay or bandwidth to its descendants. If so, A attempts to move under B by issueing the add request to B and waiting for the response. If the request is accepted, A notifies its old parent, communicates its new delay from the root to all its children, and notifies the new parent B of its farthest descendant. The use of RanSub for SARO is circular: SARO uses RanSub to probe peers and locate neighbors that meet performance targets, and at the same time, RanSub uses the SARO overlay for efficient distribution of collection and distribution messages. SARO requires no global coordination to perform overlay transformations. A special instance of RanSub ensures a total ordering among all participants such that no two simultaneous transformations can introduce loops into the overlay.

RMX. RMX [6] proposes to partition the heterogeneous receiver set into a number of small homogeneous data groups. Each data group is assigned a Reliable Multicast proXy (RMX). RMXs represent strategically located agents organized into an overlay network. Placing RMXs strategically to form an overlay network is a hard problem. RMX uses manual configuration of receivers into independent data groups that are bridged by a manually constructed spanning tree of RMXs.

A network of application-aware RMXs uses detailed knowledge of application semantics to adapt to heterogeneity constraints. As data flows through an RMX, it dynamically alters the content of the data or adapts the rate and ordering of data objects. Each receiver can define its own level of reliability and decide how and to what extent individual data objects can be transformed and compressed. A source or a receiver joins the specific RMX session by subscribing to its local data group. RMXs spread across the data groups organize themselves into a spanning tree via unicast interconnections. To communicate with other RMXs across wide-area links, an RMX uses a reliable unicast communication protocol such as TCP; within each data group, RMX relies on multicast forwarding. Data dissemination is performed in the following manner: a source generates data and distributes it to its local group and the corresponding RMX. The RMX forwards data from the local group towards the participants in the other groups. Whenever an RMX receives data on a link, it forwards it to all of its remaining links using spanning-tree flooding.

The recovery of lost data is performed in such a way that in the first attempt the receiver tries to retrieve the data form the local group. If this fails, the group's RMX forwards the recovery request upward along the RMX hierarchy towards the source of the data. Intermediate RMXs first attempt to recover the data from their local group, and if that fails, forward the request toward the source. Each RMX maintains a table that is used to determine the path toward sources from that RMX.

TAG/STAG. Topology-Aware Grouping (TAG) [17] is a single-source multicast protocol that exploits network topology information in the overlay construction. It assumes that it is possible to obtain the underlying path and bandwidth information to a designated member by using one of the existing route discovery mechanisms. The shortest path information that IP routers maintain is exploited when performing the path matching algorithm. This algorithm allows a TAG member to select as a parent a member whose shortest path from the source has the longest common prefix, in terms of the number of hops, with its own shortest path from the source. Alternatively, a partial matching algorithm can be used, thereby a node may join a parent that provides enough bandwidth, even if that does not follow the longest prefix matching scheme. After joining a multicast group the member maintains a family table (FT) where parent-child relationships for this node are stored.

The multicast tree management protocol encompasses three main actions: member join, member leave, and member failure. A new member that wishes to join a session sends a JOIN message to the source (the root of the tree). Upon the receipt of the JOIN message, the source computes the underlying paths to the new member and executes the path matching algorithm. If the parent of the new member is the source itself, the FT of the source is updated and the joining procedure is completed;

otherwise, the source sends a FIND message to its child that shares the longest prefix with the new member's path. The procedure continues until the appropriate parent for the new member is found. To leave a session, a member sends a LEAVE message to its parent together with its FT. Upon receipt of the LEAVE message, the parent removes the leaving member from its FT and takes over its children by adding their FT entries to its own FT. To detect failures, session members periodically exchange reachability messages with their children. The failure of a child is detected as the absence of exchange data, in which case the parent simply discards the child from its FT. When the failure of a parent is detected, the child must rejoin the session.

Subnet Topology-Aware Grouping (STAG) proposes two variations to the join procedure in TAG: reverse control and delay control. With reverse-control, in parallel to sending the JOIN message to the source node, a new node broadcasts the JOIN message in the subnet. As a result, some subnet nodes may reply back with a HELLO message. The new node accepts as a parent the node whose response arrives first and sends an OK message with the new parent's address to the other nodes. Since the join request is sent to the source as well, there is no time loss in cases where the node does not find a parent in its subnet. However, if the parent is found within the subnet, a STOP message is sent up the tree to stop the continuation of the join process. With delay-control, the two actions are separated and performed sequentially. The new node first broadcasts a JOIN message within the subnet. If it does not receive any HELLO message in response for a defined time interval, it will send the JOIN message to the source.

TBCP. In the Tree Building Control Protocol (TBCP) [22] participating members have a partial knowledge of the network topology and of the group membership before the spanning tree is computed. Every host in a TBCP tree is allowed to constrain its out degree by setting up the maximum number of children it is willing to support. In doing this, the host can control the amount of traffic it will be responsible for. When a new node wishes to join the tree and become a member of a multicast group, it uses the root of the spanning tree as a rendezvous point, assumed to be known by the node. It then sends a HELLO message to the root, which replies back with the list of its children and starts a timer waiting for the response from the new node. The root is not allowed to accept any further HELLO messages before the join procedure is completed or the timer expires. The new node estimates the distance (based on host-to-host measurements) to every child of the root and sends this information in a JOIN message to the root. If the timeout has not expired, the root evaluates all possible configurations based on the measurements received from the new node. If it judges that becoming the parent of the new node is the best configuration, it replies with a WELCOME message, the new node acknowledges it, and the joining procedure is completed. Alternatively, the root node may decide to redirect the new node to one of its children, thereby the join procedure starts again with the root's child playing the role of the new root. If on the other hand the timeout expires, the root sends a RESET message to the new node signalizing that it must restart the join procedure. Additional rules for the overlay tree construction can be used to improve the efficiency and the shape of the overlay tree. Such rules assume

a hierarchical organization of the receiver nodes in a certain number of domains, so that receivers belonging to the same domain are grouped in the same sub-tree. TBCP considers the distance metric used for estimating the score of a configuration to be an application specific.

TOMA. Two-tier Overlay Multicast Architecture (TOMA) [18] requires infrastructural support for providing the multicast service. It implements an aggregated multicast approach in which multiple groups share one delivery tree. Although this reduces the tree management overhead as well as the amount of multicast state information stored in routers, it may result in some hosts receiving information they have not subscribed for, leading to bandwidth waste.

Special service nodes referred to as member proxies (edge nodes) and host proxies (internal nodes) are connected in a tree structure. This overlay proxy placement is an optimization problem defined as follows: given the number of group members for all routers and the shortest distance between any two routers, find at most K routers as proxies such that the weighted sum of distances from each router to its nearest proxy is minimized. End hosts, both sources and receivers, are grouped in clusters according to a distance metric and connected to a tree structure called P2P multicast tree.

Each TOMA group has a unique URL-like identifier. End users subscribe to the specific group by sending a request to the group membership server, which decides about the member access and maintains the group membership information. A list of IP addresses of advertised member proxies is obtained from a DNS server, and one is selected from the list, taking into account latency and load. The selected member proxy creates the P2P multicast tree using a centralized approach.

When an end host wishes to multicast a message, it communicates with other participants and disseminates data. The message is transmitted to all other users in the same cluster and to the member proxy in charge of that group. This member proxy will disseminate the message to other member, and they will send the data further on to the group members in their clusters.

AEMD/GASR. AEMD (Adaptive Algorithm for Efficient Message Diffusion) [11] and GASR (Gambling Algorithm for Scalable Resource-Aware Streaming) [1] aim at reliable information diffusion. Both algorithms use a modular approach in two layers: while the lower layer is responsible for the environment modeling, the upper layer builds a spanning tree based on the environment modeling to reach all destinations.

To measure the reliability of links and nodes, nodes probe their direct neighbors. The environment approximation solution permits each node to have a continuously updated view of the system, including the components' reliability and the topology. The node's view is a subgraph of the system's topology, built incrementally. In AEMD this view is complete, in which case nodes try to approximate the entire topology; in GASR it is partial, defined by the maximum diameter of each node's view.

AEMD uses an overlay named Maximum Reliability Tree (MRT) built in a centralized manner by the source when a message should be disseminated. The MRT is

built in two steps: First, the source selects the most reliable components to include in the MRT, using the information from the environment modeling layer. Second, an optimize function assigns to each of the included links in the MRT the number of retransmission necessary to reach a given success probability.

GASR builds a Maximum Probability Tree (MPT) to stream data. The MPT matches the physical properties of the system and is built in a distributed manner. Some nodes, called *Incrementing Nodes*, contribute to the construction of MPT using theirs respective partial knowledge about the system. The process is incremental, starting from the source, by aggregating subtrees covering different partial views. The goal is to maximize the probability to reach all members in the neighborhood of a node given a fixed quota of messages that can be used per node.

1.5.2 Mesh-Based Overlay

Narada. Narada [7] proposes an architecture where multicast is implemented assuming only unicast IP service. The protocol is robust to failure of end systems and to dynamic changes in group membership.

Narada builds an overlay structure in a self-organized and fully distributed manner. End systems gather information of the network characteristics using passive monitoring and active measurements. An overlay structure representing a spanning tree is constructed in two steps. First, Narada builds a mesh with desirable performance properties such as the quality of the path between any pair of members, using for example delay or bandwidth, and a limited number of neighbors each member has in the mesh. For doing so, Narada requires all the nodes to have a global knowledge, i.e., to maintain a list of all other members in the group. The protocol continuously refines the overlay structure as new information is available.

An out-of-band bootstrap mechanism is assumed to exist that allows a member to retrieve a list of group members. This list needs neither to be complete nor accurate, and must contain at least one currently active group member. Dynamic changes are taken care of by allowing the protocol to incrementally improve the mesh quality by adding or dropping some of the overlay links. Members probe each other at random and gather information about utility of links. The decision on adding or dropping a link is based on these measurements. A good quality mesh must ensure that for any pair of members there exist paths along the mesh which can provide performance comparable to the performance of the unicast path between the members. The precise utility function depends on the performance metrics that the overlay is optimized for.

ODMRP. ODMRP [19] is a mesh-based multicast routing protocol. It particularly aims at mobile ad hoc wireless networks as it reduces the link/storage overhead. It is based on the concept of a *forwarding group*, which represents a set of nodes responsible for forwarding multicast data on shortest paths between any member pairs,

building consequently a *forward mesh* for each multicast group. ODMRP has two phases: a request phase and a reply phase. In the request phase, the multicast source periodically sends to the entire network an advertising packet, named *Join Request*. When a node receives a *Join Request* for the first time, it stores the upstream node ID and rebroadcasts the packet. When the *Join Request* reaches a multicast receiver, the receiver creates or updates the source entry in its member table. At the reply phase, each node sends periodically a *Join Table* to its neighbors. When a node receives a *Join Table*, it checks if one entry corresponds to its own ID. If so, that node realizes that it is on the shortest path to the source and thus is part of the *forwarding group*. Forwarded by the *forwarding group* member, the *Join Table* reaches the source via the shortest path. This implicitly constructs and maintains a mesh with the *forwarding group* ensuring the shortest path from the sources to the receivers.

OverStream. OverStream [21] uses a heuristic approach to construct both a delay- and degree-bounded application-level multicast tree in a distributed fashion. The algorithm constructs an application-level multicast tree (ALMT) intended to provide an efficient solution to live audio and video data streaming in a single source scenario. The solution takes into consideration the essential characteristics of live streaming applications, such as delay sensitivity, real-time constraints and the amount of concurrent continuous user requests. This results in an NP-hard optimization problem referred to as Maximum Sum of Nodes Multicast Tree (MSNMT) subject to delay and out degree bound constraints. The ALMT is constructed on top of the overlay network such that the tree has the source node as its root. In ALMT there is an overlay path from the source to every other node. The delay constraint requires that the overlay path delay of any node in the tree be less than a predefined delay bound. The degree constraint requires that the out-degree of any node in ALMT be less than a maximum out-degree.

OverStream works as follows: initially, each node continuously sends probe messages to other nodes in order to approximate the overlay. Nodes approximate only a portion of the network and construct an incomplete directed overlay. Afterwards, the ALMT is constructed in a fully distributed manner using a heuristic named the Powerful Propagating Ability First (PPAF). When a new node wishes to join the ALMT, it first constructs a sub-tree having itself as a root and calculates PPAF for every node in its probed node set. PPAF estimates the sum of nodes that a child node brings to a parent, so that a child can choose a node with the maximal PPAF as its parent. If the parent node is saturated and incapable of supporting more children, the child with the minimal PPAF is chosen from the set of all parent's children, the new node takes its place, and the old child is requested to perform a rejoin procedure. To leave the ALMT a node sends a leave message to its parent node and sends rejoin messages to all of its children. If a node detects missing control messages it infers the failure of its parent, and as a reaction, it performs a node join procedure and sends messages to its siblings requiring them to perform rejoin as well.

Yoid. Yoid [10] employs a single shared tree of all members of the group and creates a mesh topology to recover from tree partitions. The first member to join a

tree becomes its root. Each subsequent joining member contacts a centralized rendezvous point (RP) and obtains a list of members. These members represent the candidate parents of the joining member. The parent selection depends on a performance metric, e.g., loss rate, latency. To optimize the tree, each node periodically queries RP for updates on their initial lists of candidate parents. A node is allowed to switch to a new parent if this one provides a better latency or packet loss performance. To guarantee a non-partitioned mesh topology, each member M establishes a small number of other members as mesh neighbors. These members are randomly selected, with the exceptions that they must not include members that are tree neighbors, and must not include members that have already established a mesh link to M. The reason for this latter restriction is to prevent trivial cliques, where three or four members all use each other as mesh neighbors, thus partitioning themselves from the rest of the mesh topology.

1.5.3 Logical ID Overlay

CAN-multicast. CAN-multicast [25] is designed to scale to large groups without restricting the service to a single source. It makes use of the Content-Adressable-Networks (CAN) [24] architecture to provide an application level structured peer-to-peer overlay network whose constituent nodes form a virtual d-dimetional Cartesian coordinate space and each member owns its individual distinct zone in this space. CAN-multicast can be deployed for large group sizes where each member needs to maintain knowledge about only a small subset representing its neighbors. Neighborhood relationships are dictated by adresses assigned to hosts rather than performance. This technique gains in simplicity and scalability by not having to run routing algorithms to construct and maintain delivery trees.

To join CAN, a new node first looks up the CAN domain name in DNS to retrieve a bootstrap IP address. The bootstrap node supplies the IP address of several randomly chosen nodes currently in the system. The new node then randomly choses a point (x,y) in the space and sends a JOIN request destined to point (x,y). The current occupant node then splits its zone in half and assigns one half to the new node. In CAN-multicast, if all the nodes in CAN are members of a given multicast group, then the multicast data topology is implicitly defined by performing directed flooding on the control topology. If only a subset of the CAN nodes are members of a particular group, then the members of this group first form a mini CAN and then the multicast is achieved by flooding over the mini CAN. CAN-multicast's flooding solution aims at reducing message duplication; it is less robust than the naive flooding algorithm because the loss of a single message results in the breakdown of message delivery to several subsequent nodes.

DT. The *Delaunay Triangulation* (DT) multicast protocol [20] is based on an overlay network composed of logical coordinates assigned to the members of the underlying network topology. The DT protocol can build and maintain very large overlay

networks with relatively low overhead, at the cost of suboptimal resource utilization due to a possibly poor match of the overlay network to the network-layer infrastructure. Hence the DT protocol trades off economy of scale for increased scalability. The use of DT as an overlay has several advantages. For example, a DT has a set of alternate non-overlapping routes between any pair of vertices that represent an alternate path when nodes fail. Moreover, a DT is established and maintained in a distributed manner.

A Delaunay Triangulation for a set of vertices A is a triangulation graph with the defining property that for each circumscribing circle of a triangle formed by three vertices in A, no vertex of A is in the interior of the circle. In order to establish a Delaunay triangulation overlay, each node is associated with a vertex in the plane with given (x,y) coordinates. The coordinates are assigned via some external mechanisms (e.g., GPS or user input) and can be selected to reflect the geographical locations of nodes. Multicast forwarding in the Delaunay triangulation is done along the edges of a spanning tree that is embedded in the Delaunay triangulation overlay, and that has the sender as the root of the tree. Each node can locally determine its child nodes with respect to a given tree using its own coordinates, the coordinates of its neighbors, and the coordinates of the sender. Local forwarding decisions at nodes are done using compass routing [16]. Although compass routing in general planar graphs may result in routing loops [14], they can not happen in Delaunay triangulations [16].

Bayeux. Bayeux's multicast [32] is based on Tapestry [31], an application-level routing protocol. Nodes in Tapestry have names independent of their location and semantic properties, in the form of random fixed-length bit-sequences represented by a common base. Tapestry uses local routing maps stored at each node named neighbor maps. These maps are used to incrementally route overlay messages to the destination ID digit by digit.

Bayeux uses a source-specific model for data dissemination. Every multicast session is identified by a unique tuple. Tapestry's data location services are used to advertise Bayeux multicast sessions in the following manner: A unique session identifier is first securely mapped into a 160-bit identifier using a one-way hash function; then a file named as the identifier is created and placed on the multicast session's root node. Using Tapestry location services, the session's root (source) server advertises that document into the network. Clients willing to join a session must know the unique tuple that identifies the session. After joining a session, clients can then perform the same operations to generate the file name, and query for it using Tapestry.

A session root represents a single point of failure that can compromise the entire group's ability to receive data. It also represents a scalability bottleneck since all messages have to go through the root that has to maintain global knowledge by keeping a list of all group members. To overcome these issues, Bayeux allows the creation of multiple root nodes and partitions receivers into disjoint membership sets, each containing receivers closest to a local root in network distance. Every root contains the object to be advertised, and a new member uses Tapestry location service as before, and becomes a member of the nearest root's receiver set.

SCRIBE. SCRIBE [5] is a multi-source decentralized multicast infrastructure built on top of Pastry [26]. Pastry leverages a fully decentralized peer-to-peer model according to which a scalable and self-organizing overlay network of nodes is built. Nodes are assigned 128-bit uniformly distributed identifiers, i.e., nodeIds. Reliable message dissemination in Pastry is performed as follows: given a message and a key, Pastry routes the message to the Pastry node that has the nodeId numerically closest to the key, among all nodes. Each node maintains a routing table with entries that map the nodeId to the node's IP address. Every entry in the routing table can refer to potentially many nodes with the appropriate prefix. Among these nodes with the appropriate prefix, only the closest one to the present node is chosen according to some metric, e.g. a proximity metric such as the round trip time. Besides a routing table, each node maintains a leaf set, which represents sets of nodes with the numerically closest larger and numerically closest smaller nodeIds.

When a node wishes to join a multicast group, it chooses a nodeId and issues the request to a nearby node to route a special message using that nodeId. The message is routed by Pastry to the node that has numerically closest nodeId to the given one. The new node then initializes its state by obtaining the leaf set from the closest node. To detect node failures, neighboring nodes periodically exchange keep-alive messages. A node that does not send a response within a certain defined period of time, is assumed to be failed. The nodes from the leaf set of the failed node are then informed to update their leaf sets.

Every multicast group in SCRIBE has a unique groupId and is associated with a rendezvous point. A node that has a nodeId numerically closest to the groupId is chosen to act as the rendezvous point. The overlay tree is built upon Pastry by joining the Pastry routes from each group member to this rendezvous point that represents the root of the tree. Each forwarder, i.e., a node that is a part of a group's multicast tree, maintains a children table containing an entry for each of its children in the tree. When a SCRIBE node wishes to join a group, it asks Pastry to route a JOIN message with the groupId as the key. If on the route a node is not a forwarder, it creates an entry for the group and adds the source node as a child in its children table. The process is repeated until the JOIN message reaches the rendezvous point. SCRIBE manages group joining operations in a fully distributed manner and supports large and dynamic membership. When a SCRIBE node wishes to leave, it sends a LEAVE message to its parent initiating the appropriate structures to be updated.

In the presence of a node failure, a child node that detects the heartbeat message loss will suspect its father has failed and call Pastry to route a JOIN message which will lead to a new parent. In order to tolerate the rendezvous point's failure, SCRIBE allows its state to be replicated in a certain number of the neighboring nodes from the leaf set. SCRIBE provides best-effort delivery of messages, but leaves an open framework for applications to implement stronger reliability guarantees.

SplitStream. Based on Scribe and Pastry [26, 5] to build and maintain a tree-based application-level multicast, SplitStream [4] defines several tree structures forming a forest to duplicate and forward content. The key idea in SplitStream is to split the content into k stripes and to multicast each stripe using a separate tree. Peers

join as many trees as the stripes they wish to receive and specify an upper bound on the number of stripes they are willing to forward. The resulting forest of multicast trees distributes the forwarding load subject to the bandwidth constraints of the participating nodes.

SplitStream does not only adapt the outbound bandwidth but also the inbound bandwidth contraints. Assuming that the content has a bandwidth requirement of B and the content is split into k stripes. Participating peers may receive a subset of the stripes, thus controlling their inbound bandwidth requirement in increments of B/k. Similarly, peers may control their outbound bandwidth requirement in increments of b/k by limiting the number of children they adopt.

PeerCast. PeerCast [30] is a reliable multicast service for dynamic high-churn environments of possibly unreliable peers that may enter and depart the system at arbitrary points in time. It is based on a decentralized mechanism that utilizes peer network proximity measurements in order to efficiently cluster end hosts in the PeerCast P2P network. The protocol further uses these clusters for building efficient multicast trees that minimize latency. PeerCast accounts for the diversity of computing capacities and network bandwidth of large networks by organizing hosts into an overlay which balances host multicast workloads. PeerCast takes advantage of Internet landmark-based techniques to guarantee that peer identifiers are allocated such that peers near each other in terms of their locations in the physical network have numerically close identifiers.

Load imbalance due to node heterogeneity is addressed in PeerCast using virtual nodes. The idea is used to allow any peer to be assigned one or more virtual nodes based upon the resource availability at the peer. The resource availability of a peer is designed to represent a weighted sum of three components, namely bandwidth availability, CPU availability, and memory availability. The resource availabilities are estimated using some of the existing techniques in the literature, while the weight assignment is application specific and given by the specification of the publisher of the content. Peers periodically monitor the resource availabilities and perform a procedure called virtual node promotion that ensures corresponding multicast tree reorganization if necessary.

In order to cope with high churn-rates and avoid multiple re-subscription of many nodes in cases where a node gracefully leaves or fails, PeerCast proposes a passive service replication scheme. Each peer maintains a Replication List of other peers that can take over the role of a parent and proceed with sending data to the downstream nodes in case of a failure or a peer leave. The replication scheme is installed right after the group information is established on a peer; the selected replicas are kept consistent as hosts join or leave the end-system multicast group.

1.6 Summary

Table 1.1 shows a recapitulative view of the main aspect characterising the multicast solutions described before. With general properties (e.g. goals), this table includes architecture, service and performance properties.

Most multicast protocols surveyed focus on minimizing latency and improving scalability. These goals are justified by the nature of the environment targeted by typical application-level multicast protocols, namely loosely coupled large-area networks, such as the Internet. Indeed, most solutions aim at large systems with hundreds or thousands of nodes. In these contexts, protocols do not seem to significantly favor a design for a specific source or for multi sources, in which the same overlay network is reused by any peer willing to disseminate information.

While most of the protocols surveyed were originally designed for P2P Internet environments, some could be used in MANET settings. Yet, most of them are not suitable to MANETs. At first glance, this may look surprising due to the similarities between P2P networks and MANETs. In reality, however, MANET components are more resource-constrained then P2P Internet peers. Moreover, some P2P protocols rely on assumptions that are not feasible in MANETs (e.g., the existence of rendezvous points).

With respect to overlay structure, the large majority of protocols uses physical node IDs, as opposed to logical IDs. Moreover, by far and large most protocols build tree overlays, and the few ones based on meshes use it mainly to control information, not to disseminate data. Disseminating information over a mesh would create redundant data paths and waste bandwidth in the absence of failures. While this would provide higher reliability, as we will see next, protocols tend to resort to different mechanisms to handle peer or link failures.

It is perhaps less surprising that most solutions implement a distributed overlay creation and maintenance, where the responsibility is shared among several peers, as opposed to a centralized approach in which one peer is responsible for building the overlay. One consequence of the centralized technique is that one node must have global knowledge about the system. While most multicast protocols surveyed strive to avoid global knowledge, a few adopt a hybrid approach in which some nodes must maintain a complete view of the system. This is typically the case with rendezvous points, used by nodes to join the system.

Infrastructure support is not described in most of the techniques we surveyed. In part this is due to the fact that some of the papers considered focus on algorithmic aspects instead of systems considerations. This also has an impact on the way some of the algorithms are evaluated, often by simulation.

As for adaptivity, while most techniques are self-configurable—that is, the overlay is built incrementally, based on peers joining and leaving the system—some are also self-improving, meaning that they adapt to the physical network characteristics. Interestingly, although almost all protocols provide some support to handle failures, most prefer to recover from failures, instead of preventing them from happening. One possible reason for this is that preventing failures in the occurrence of compo-

nent errors usually involves some level of redundancy, and a consequent increase in the cost of the protocol (e.g., in terms of bandwidth).

Summing up, the prototypical ALM protocol, one that would encompass most of the features of existing systems, would be designed to *minimize latency*, probably most appropriate for *P2P* networks, based on *physical IDs* structured as an *overlay tree*, built for a *specific source*. The overlay would be created and maintained in a *distributed* manner, using *local-knowledge*. Finally, the protocol would be *self-configurable* and perhaps also *self-improving*, adapted to *medium and large networks*, and would probably be *unreliable* (i.e., reacting to failures or best effort). It turns out that five protocols in our survey match the prototypical ALM protocol: ALMA, Narada, OMNI, OverStream, and RanSub.

Given the large number of existing proposals, covering different aspects and characteristics, it is difficult to speculate on the future of structured information dissemination research. Nevertheless, one could consider ideas that have been fruitful in related areas of data dissemination (e.g., group communication for smaller-scale systems). In this context, application semantics have been used to design protocols tailor-made for specific applications. The idea is that applications can specify requirement constraints, and only pay the price (e.g., in terms of bandwidth or response time) for the requirements chosen. We have identified one protocol that builds on similar ideas (i.e., RMX) but we could expect to see others in the future.

As seen before, adaptivity is currently accounted for using mostly self-configurable and self-improving techniques. It turns out that systems can also improve reliability by adapting to the changing characteristics of the environment in which they execute (e.g., node crashes). One class of such solutions is self-stabilizing algorithms [9]. We have not identified any such a protocol in the context of structured dissemination. Among its advantages, besides handling environment changes naturally, it could provide an adequate framework for designing provable system properties.

Table 1.1 Summary of the properties of the various techniques

	Applicative viewpoint			**Architecture viewpoint**				**QoS viewpoint**		
Solution	**Usage focus**	**Communication model**	**MANET Fit**	**Node ID Topology**	**Creation & Maintenance**	**Environment modeling**	**Infrastructure support**	**Adaptivity**	**Scalability**	**Reliability**
AEMD	Reliability	Specific source	No	Physical ID Tree	Centralized	GK	No	Static	Small	Reliable
ALMA	Min. Latency RTT metric	Specific source	Yes	Physical ID Tree	Distributed	LK at clients GK at DS	No	Self-configurable Self-improving	Medium	Fault tolerant
ALMI	Min. Latency	Multi sources	No	Physical ID Tree	Centralized	GK	No	Self-configurable	Small	Fault tolerant
Bayeux	Fault tolerance	Specific source	No	Logical ID Tree on Tapestry	Distributed	LK at nodes GK at root	No	Self-configurable	Large	Fault tolerant
BTP	Min. Tree cost	Multi sources	No	Physical ID Tree	Distributed	GK	No	Self-configurable Self-improving	Medium	Fault tolerant
CAN-multicast	Scalability	Multi sources	No	Logical ID undefined	Distributed	LK	No	Self-configurable	Large	Best Effort
DT	Scalability Low overhead	Multi sources	No	Logical ID undefined	Distributed	LK	No	Self-configurable	Large	Best Effort
GASR	Resources adaptive	Specific source	No	Physical ID Tree	Distributed	LK	No	Self-improving	Large	Reliable
HMTP	Min. Tree cost	Multi sources	No	Physical ID Tree	Distributed	LK at clients GK at RP	Yes	Self-configurable	Medium	Fault tolerant
MAODV	Low overhead	Multi sources	Yes	Physical ID Tree	Distributed	LK	No	Self-configurable	Medium	Best Effort
Narada	Min. Latency	Specific source	Yes	Physical ID Mesh-Tree	Distributed	GK	No	Self-configurable Self-improving	Medium	Best Effort
NICE	Scalability Low overhead	Specific source	Yes	Physical ID Mesh-Tree	Layered	LK	No	Self-configurable Self-improving	Large	Best Effort
ODMRP	Min Latency	Multi sources	Yes	Physical ID Mesh	Distributed	LK	No	Self-configurable	Medium	Best Effort

	Applicative viewpoint			Architecture viewpoint				QoS viewpoint		
Solution	**Usage focus**	**Communication model**	**MANET Fit**	**Node ID Topology**	**Creation & Maintenance**	**Environment modeling**	**Infrastructure support**	**Adaptivity**	**Scalability**	**Reliability**
OMNI	Min. Latency	Specific source	No	Physical ID Tree	Distributed	LK	Yes	Self-configurable Self-improving	Large	Best Effort
oStream	Asynchronous streaming	Specific source	No	Physical ID Tree	Distributed	LK	Yes	Self-configurable	Medium	Best Effort
Overcast	Adapt to bandwidth	Specific source	No	Physical ID Tree	Distributed	LK at Proxies GK at root	Yes	Self-configurable Self-improving	Large	Fault tolerant
OverStream	Min. Latency Bounded out-degree	Specific source	No	Physical ID Tree	Distributed	LK	No	Self-configurable Self-improving	Large	Fault tolerant
PeerCast	Min. Latency Resource adaptive	Specific source	No	Logical ID Tree on DHT	Distributed	LK	No	Self-configurable Self-improving	Large	Reliable
RanSub	Min. Latency Max. throughput	Specific source	No	Physical ID Tree	Distributed	LK	No	Self-configurable Self-improving	Medium	Fault tolerant
RMX	Reliability	Multi sources	No	Physical ID Tree	Distributed	GK	Yes	Self-configurable	Large	Reliable
SCRIBE	Scalability Min. Latency	Multi sources	No	Logical ID Tree on Pastry	Distributed	LK	No	Self-configurable	Large	Best effort
SplitStream	Load balancing Adapt to bandwidth	Specific source	No	Logical ID Tree on Scribe	Distributed	LK	No	Self-configurable Self-improving	Large	Fault tolerant
TAG	Min. Latency	Specific source	Yes	Physical ID Tree	Distributed	GK	No	Self-configurable	Medium	Fault Fault
TBCP	Bounded out-degree	Multi sources	Yes	Physical ID Tree	Distributed	LK	No	Local Self-improving	Medium	Best effort
TOMA	Scalability	Multi sources	No	Physical ID Aggregated Trees	Distributed	LK GK for proxies	Yes	Self-configurable	Large	Best effort
Yoid	Min. Latency	Multi sources	No	Physical ID Tree-Mesh	Distributed	LK	No	Self-configurable Self-improving	Medium	Fault tolerant

References

1. M. Allani, B. Garbinato, F. Pedone, and M. Stamenkovic. Scalable and reliable stream diffusion: A gambling resource-aware approach. In *Proceedings of 26th IEEE Symposium on Reliable Distributed Systems (SRDS'2007)*, pages 288 – 297, Beijing, CHINA, 2007.
2. S. Banerjee, B. Bhattacharjee, and C. Kommareddy. Scalable application layer multicast. In *SIGCOMM '02: Proceedings of the 2002 conference on Applications, technologies, architectures, and protocols for computer communications*, New York, NY, USA, July 2000. ACM Press.
3. S. Banerjee, C. Kommareddy, K. Kar, B. Bhattacharjee, and S. Khuller. Omni: An efficient overlay multicast infrastructure for real-time applications. *Computer Networks*, 50(6):826–841, 2006.
4. M. Castro, P. Druschel, A. M. Kermarrec, A. Nandi, A. Rowstron, and A. Singh. Splitstream: high-bandwidth multicast in cooperative environments. In *SOSP '03: Proceedings of the nineteenth ACM symposium on Operating systems principles*, pages 298–313, New York, NY, USA, 2003. ACM Press.
5. M. Castro, P. Druschel, A. M. Kermarrec, and A. Rowstron. Scribe: a large-scale and decentralized application-level multicast infrastructure. *Selected Areas in Communications, IEEE Journal on*, 20, 2002.
6. Y. Chawathe, S. McCanne, and E. A. Brewer. RMX: Reliable multicast for heterogeneous networks. In *INFOCOM*, pages 795–804, Tel Aviv, Israel, 2000. IEEE.
7. Y. Chu, S. Rao, and H. Zhang. A case for end system multicast. In *Proceedings of ACM Sigmetrics, June 2000*, pages 1–12, June 2000.
8. Y. Cui, B. Li, and K. Nahrstedt. ostream: asynchronous streaming multicast in application-layer overlay networks. *Selected Areas in Communications, IEEE Journal on*, 22(1):91–106, 2004.
9. S. Dolev. *Self-Stabilization*. The MIT Press, 2000.
10. P. Francis. Yoid: Extending the internet multicast architecture. Technical report, AT&T Center for Internet Research at ICSI (ACIRI), 2000.
11. B. Garbinato, F. Pedone, and R. Schmidt. An adaptive algorithm for efficient message diffusion in unreliable environments. In *Proceedings of IEEE DSN'04 , June 2004)*, pages 507–516, June 2004.
12. D. Helder and S. Jamin. Banana tree protocol, an end-host multicast protocol. Technical report, University of Michigan, 2002.
13. J. Jannotti, D. K. Gifford, K. L. Johnson, M. F. Kaashoek, and Jr. J. W. O'Toole. Overcast: reliable multicasting with on overlay network. In *OSDI'00: Proceedings of the 4th conference on Symposium on Operating System Design Implementation*, Berkeley, CA, USA, 2000. USENIX Association.
14. B. Karp. Geographic routing for wireless networks. *Ph.D Thesis. Harvard University*, 2000.
15. D. Kostic, A. Rodriguez, J. Albrecht, A. Bhirud, and A. Vahdat. Using random subsets to build scalable network services. In *Proceedings of USITS, March 2003*, March 2003.
16. E. Kranakis, H. Singh, and J. Urrutia. Compass routing on geometric networks. In *Proc. 11 th Canadian Conference on Computational Geometry*, pages 51–54, Vancouver, August 1999.
17. M. Kwon and S. Fahmy. Path-aware overlay multicast. *Computer Networks*, 47(1):23–45, January 2005.
18. L. Lao, J. H. Cui, and M. Gerla. Toma: A viable solution for large-scale multicast service support. In *NETWORKING*, pages 906–917, 2005.
19. S. J. Lee, M. Gerla, and C. C. Chiang. On-demand multicast routing protocol (odmrp). In *Proceedings of IEEE WCNC '99*, pages 1298–1302, New Orleans, LA, September 1999.
20. J. Liebeherr and M. Nahas. Application-layer multicast with delaunay triangulations. Technical report, University of Virginia, Department of Computer Science, November 2001.
21. F. Liu, X. Lu, Y. Peng, and J. Huang. An efficient distributed algorithm for constructing delay and degree-bounded application-level multicast tree. In *ISPAN '05: Proceedings of the 8th International Symposium on Parallel Architectures,Algorithms and Networks*, Washington, DC, USA, 2005. IEEE Computer Society.

22. L. Mathy, R. Canonico, and D. Hutchison. An overlay tree building control protocol. In *NGC '01: Proceedings of the Third International COST264 Workshop on Networked Group Communication*, London, UK, 2001. Springer-Verlag.
23. D. Pendarakis, S. Shi, D. Verma, and M. Waldvogel. Almi: an application level multicast infrastructure. In *USITS'01: Proceedings of the 3rd conference on USENIX Symposium on Internet Technologies and Systems*, Berkeley, CA, USA, 2001. USENIX Association.
24. S. Ratnasamy, P. Francis, M. Handley, R. Karp, and S. Schenker. A scalable content-addressable network. In *SIGCOMM '01: Proceedings of the 2001 conference on Applications, technologies, architectures, and protocols for computer communications*. ACM Press, 2001.
25. S. Ratnasamy, M. Handley, R. Karp, and S. Shenker. Application-level multicast using content-addressable networks. In *NGC '01: Proceedings of the Third International COST264 Workshop on Networked Group Communication*, pages 14–29, London, UK, 2001. Springer-Verlag.
26. A. Rowstron and P. Druschel. Pastry: Scalable, distributed object location and routing for large-scale peer-to-peer systems. *IFIP/ACM International Conference on Distributed Systems Platforms (Middleware)*, 11:329–350, 2001.
27. E. M. Royer and C. E. Perkins. Multicast operation of the ad-hoc on-demand distance vector routing protocol. In *MobiCom '99: Proceedings of the 5th annual ACM/IEEE international conference on Mobile computing and networking*, pages 207–218, New York, NY, USA, 1999. ACM.
28. C. K. Yeo, B. S. Lee, and M. H. Er. A framework for multicast video streaming over ip networks. *J. Network and Computer Applications*, 26(3):273–289, 2003.
29. B. Zhang, S. Jamin, and L. Zhang. Host multicast: A framework for delivering multicast to end users. In *INFOCOM*, 2002.
30. J. Zhang, L. Liu, L. Ramaswam, and C. Pu. Peercast: Churn-resilient end system multicast on heterogeneous overlay networks. *Journal of Network and Computer Applications (JNCA)*, 2007.
31. B. Y. Zhao, J. D. Kubiatowicz, and A. D. Joseph. Tapestry: An infrastructure for fault-tolerant wide-area location and routing. Technical report, University of California at Berkeley, Berkeley, CA, USA, April 2001.
32. S. Q. Zhuang, B. Y. Zhao, A. D. Joseph, R. H. Katz, and J. D. Kubiatowicz. Bayeux: an architecture for scalable and fault-tolerant wide-area data dissemination. In *NOSSDAV '01: Proceedings of the 11th international workshop on Network and operating systems support for digital audio and video*, pages 11–20, New York, NY, USA, 2001. ACM.