Available online at www.sciencedirect.com
SCIENCE DIRECT
Computer Communications 27 (2004) 1547–1568
computer communications
www.elsevier.com/locate/comcom

Short Survey

A survey of application level multicast techniques

C.K. Yeo[a,*], B.S. Lee[a], M.H. Er[b]

[a]School of Computer Engineering, Nanyang Technological University, Nanyang Avenue, Singapore, Singapore S 639798
[b]School of Electrical and Electronics Engineering, Nanyang Technological University, Nanyang Avenue, Singapore, Singapore S 639798

Received 26 April 2003; revised 8 March 2004; accepted 13 April 2004

Available online 4 May 2004

Abstract

Application level multicasting is increasingly being used to overcome the problem of non-ubiquitous deployment of IP multicast across heterogeneous networks. To the best of our knowledge, this paper is among the first papers [A comparative study of application layer multicast protocols, Unpublished report; IEEE Network, January/February, 2003] to provide a comprehensive survey of most of the various milestone research work in application level multicast in terms of both breadth and depth. The paper classifies them into different broad categories based on their topology design, service model and architecture to facilitate better understanding of their contributions and discuss their merits and limitations. As these techniques vary widely in their goals, designs, performance evaluation metrics and evaluation strategies, it is impossible to quantify their relative performance. However, this paper is able to provide a comparative insight into their performance through the use of a set of evaluation metrics identified to be common to the techniques and are directly related to their performance and whose data can be derived and inferred from their designs. The metrics used here include scaleability measured intuitively in terms of the size of the multicast receivers it can support, the protocol efficiency in terms of the quality of data paths, control overheads, amount of state information to be maintained at each member node and failure tolerance.

Keywords: Application level multicast; Overlay; Self-configurable tree; Mesh

1. Introduction

IP multicast [1–3] has been regarded as an efficient delivery mechanism for best-effort, large-scale, multi-point content delivery over the Internet as it is inherently bandwidth efficient and scaleable unlike point-to-point unicast delivery protocol. It also has built-in robustness as it does not depend on any central sources such as reflectors or mirrors for data propagation. However, IP multicast is not without its drawbacks as cited by the authors in Ref. [4]. Today's IP multicast is limited to 'islands' of network domains under single administrative control or local area networks, even though multicast has been implemented in many commercial routers. Most Internet Service Providers (ISP) are reluctant to enable it in their domains in order to reduce router load and to protect against unwanted traffic. This lack of ubiquitous multicast support more than a decade after its introduction thus limits the development of multicast applications, which in turn reduces the incentive for the network operators to enable multicast.

Simple Multicast [5], *Express* [6] and *Source Specific Multicast* [7] offer some alternate schemes to partially address the above problems by simplifying or improving multicast delivery in terms of address allocation and access control. Nevertheless, they lacked an installed base as they all require substantial changes to the network infrastructure having the same dependency on routers as their traditional IP multicast counterpart. The multicast deployment hurdles remain unresolved. There is therefore a need to provide an efficient delivery solution for multipoint communication which is ubiquitous and entails neither IP multicast support in the routers nor modifications in the network infrastructure.

Application level multicast solution is emerging as a fundamental technique to circumvent the problems of non-ubiquitous deployment of multicast across heterogeneous networks to enhance wide-area service scaleability, performance and availability [8–26]. It offers accelerated deployment, simplified configuration and better access control at the cost of additional (albeit small) traffic load

* Corresponding author. Tel.: +65-6790-4587; fax: +65-6792-6559.
E-mail address: asckyeo@ntu.edu.sg (C.K. Yeo).

0140-3664/$ - see front matter © 2004 Elsevier B.V. All rights reserved.
doi:10.1016/j.comcom.2004.04.003

in the network through an overlay-based approach. The general approach to building an application level multicast architecture involves tracking network characteristics and building appropriate topologies by having the end users to self-organize into logical overlay networks for efficient data delivery. Data delivery is accomplished via a data delivery tree which can either be the overlay itself or is embedded in the overlay. The overlay must be capable of failure detection and attempts to match the underlying network topology. Since the overlay abstracts away the details of multipoint message forwarding through the network and implements them at the end users, application level multicast is also popularly termed as end-system multicast or host multicast.

There are many ways to classify the myriad application level multicast (ALM) techniques. Here, we can broadly group them according to overlay topology design (Section 2), service model (Section 3) and architecture (Section 4) to provide a systematic insight into the contributions by the different researchers. Section 5 presents the performances of the various systems while Section 6 summarizes their properties and concludes this survey paper.

2. Classification by overlay topology design

Overlay is an integral part of application level multicast solution as it is the underlying mechanism effecting multipoint communication. The nodes in the overlays can be logically organized into two topologies, namely, the control topology and the data topology. Control topology carries control data such as heartbeat messages, refresh messages, network probes and probed data while data topology comprises the actual data delivery paths to the multipoint destinations. Nodes in the control topology may not necessarily be the members of the multicast group. Hence the control topology is a superset of the data topology. It is the norm for most techniques to adopt the tree structure for the data topology as it is simple to build and is efficient. Control topology may assume a separate physical structure in the form of a mesh where the nodes in the topology possess higher connectivity or it may share the same structure as the data topology. Depending on the approaches adopted, the resulting overlay topology can be classified into three distinct flavours: Tree, Mesh-Tree and Embedded Structure. Table 1 summarizes the topology design of the different proposals. Tree and Mesh-Tree design can be collectively grouped as topology-aware design while Embedded Structure is generally grouped under the topology-agnostic category with some exceptions as shown in Table 1. Network topology-aware design uses active measurements to infer network properties and to make an informed choice in constructing an efficient data topology to match the physical network topology as close as is practically possible. Topology-agnostic approach ignores network characteristics. Hence topology-aware algorithms have the advantage of minimizing the inefficiencies of overlays but do so at the cost of increased management overhead and potentially poor scaleability compared to its agnostic counterpart.

Table 1
Classification by overlay topology design

Topology-aware			Topology-agnostic
Tree	Mesh and tree	Embedded structure	Embedded structure
ALMA	Kudos	Bayeux	ALM-CAN
ALMI	Narada	NICE	ALM-DT
BTP	Scattercast	SCRIBE	
HM	Yoid		
OMNI			
Overcast			
TBCP			

2.1. Tree

Group members self-organize themselves into a tree by explicitly picking a parent for each new group. Nodes on the tree may establish and maintain control links to one another in addition to the links provided by the data tree. As such, the tree, with these additional control links constitutes the control topology in a tree structure. This approach is simple and is capable of building efficient data delivery trees. However, the tree building algorithm must prevent loops and handle tree partition as the failure of a single node may cause a partition of the overlay topology.

Application Layer Multicast Architecture (*ALMA*) [24–26] aims to provide best-effort multicast delivery service for real-time data streaming with the goals of reducing latency and data loss. ALMA's data topology consists of self-configuring source specific trees while its control topology comprises the data delivery trees, a centralized directory server (DS) which is the common rendezvous point for all members, refresh exchanges among a subset of members as well as updates to the DS. Hence the DS does not take part in the actual data delivery. New member queries the DS for a list of potential parents. For the earlier version of ALMA, a new member will pick the node that is closest to itself in terms of network distance approximated as Round Trip Time (RTT), to be the parent. For the improved version, ALMA is the first to incorporate loss rate with RTT as dual performance metrics to optimize the data tree [50]. Loss rate is prioritized over RTT as it directly affects the perceived quality of the received data. Hence, the potential parent is one which has a very low if not nil loss rate that is closest in RTT terms to a joining member. Unlike the improved version of Narada [10] which uses RTT and bandwidth as performance metrics, loss rate is used instead of bandwidth as the former can be implicitly estimated while data is being received by each member while the latter requires intrusive active end-to-end measurements. This involves transferring data for a period

of time using the underlying transport protocol at a rate bounded by the source rate between members, thereby incurring high overhead and latency. Members provide periodic updates to the DS of their current loss rates. Member-to-member RTT is collected via end-to-end measurements. Gossip-style algorithm (used initially in the Clearinghouse project [29] to maintain database consistency and have since been used to achieve fault tolerance and detection [30,31]) is used to improve the data tree as well as to facilitate partition recovery when a member leaves the tree. Each member will periodically try to find a better parent by exchanging loss rates and RTT data with a set of gossip candidates. These gossip candidates as well as the spiraling with their ancestors and siblings also serve as a ready list of potential parents which nodes can turn to recover from a partition. An example of the gossip and spiraling mechanisms is shown in Fig. 1. In the worst case scenario where all potential parents fail, the node can turn to the DS to request for a new list as a last recourse. State data maintained at each node is minimal as unlike HM [21], information of all members on its path to the source is not maintained and neither does it need to maintain state for the entire membership (such as Narada). Hence ALMA incurs much less overhead where data maintenance and refreshing are concerned, keeping only a small subset of information pertaining to its parent, children, grandchildren and gossip candidates. ALMA makes use of a level number for loop detection. Members update their level number each time they change a new parent to reflect their hierarchy from the source.

Application Level Multicast Infrastructure (*ALMI*) [14] provides a multicast middleware which is implemented above the socket layer. ALMI is tailored towards support of multicast groups of relatively small size (several tens) with many-to-many service mode. It employs a central controller to create a minimum spanning tree (MST) which consists of unicast connections between end hosts. Latency between members is used as the link cost of the MST thereby optimizing the ALMI overlay for low latency data delivery. Control topology takes the form of unicast connections between each member and the controller. The central controller receives updates from each member and periodically re-computes the MST. Routing information of the MST is then communicated to all the members. Ideally, since the MST is centrally computed, it will be loop-free. However, due to the losses and delays in obtaining updates from members and disseminating the different versions of MST to members, loops and partitions may occur. To prevent these problems, version number is assigned to each version of MST. Members maintain a cache of the different versions of the routing tables and only route packets with tree versions contained in the cache. If a packet with a newer tree version is received, it will re-register with the controller to receive the new MST. ALMI multicast trees have been shown [14] to be close to source-rooted multicast trees in efficiency with low performance tradeoff albeit with higher control overheads due to the maintenance of the different tree versions.

Banana Tree Protocol (*BTP*) [22] uses a receiver-based, self-organizing approach to build a shared group data tree. It is designed for distributed file sharing applications. The first host to join the group becomes the root of the tree. Subsequent new member learns of the root node and joins the tree. There is no special algorithm to prevent tree partition except for the affected node to rejoin the tree. The tree is incrementally optimized for low delay by allowing a node to switch to a sibling if the latter is closer to the node than its current parent. The proximity metric used can be the round-trip-time (RTT) between the two nodes obtained from pinging each sibling. The siblings' information maintained in each node is updated by the node's parent. Fig. 2 shows how sibling switching can lower tree cost. To prevent loops from forming during switching as a result of simultaneous switching in the case of Fig. 3 and outdated information shown in Fig. 4, two conditions must be fulfilled. First, a node will reject all attempts at switching if it is itself in the process of switching parents. Second, a node must include its current parent information

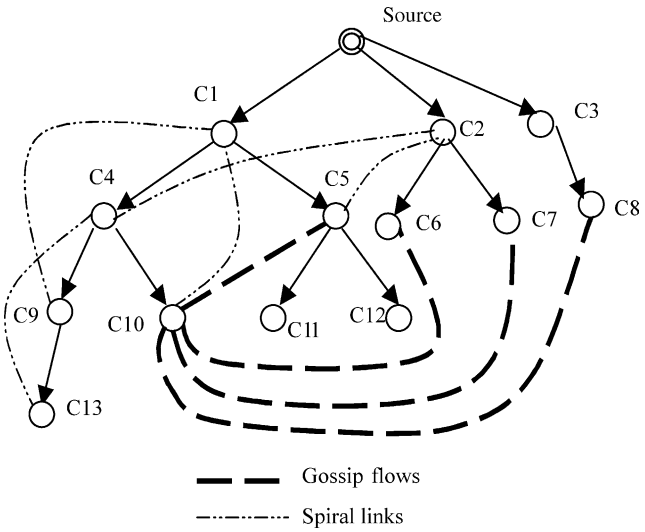


Fig. 1. An example of gossip and spiral mechanisms in ALMA. C10 gossips with C5, C6, C7, C8 while it spirals with its grandparent C1.

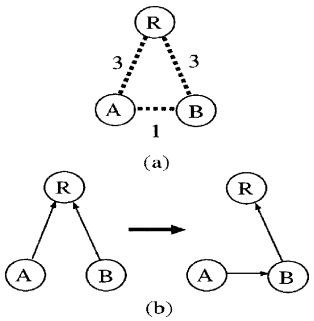


Fig. 2. Sibling switching to lower tree cost in BTP [22]. Tree cost is lowered from 6 in (a) to 4 in (b) after node A switches to node B.

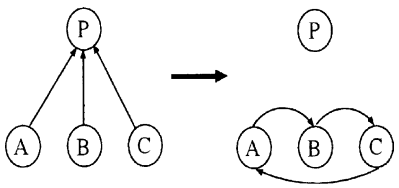


Fig. 3. Simultaneous switching creates loop in BTP [22]. A tries to switch to B, while B is trying to switch to C and the latter is attempting to switch to A.

in its switch request so that the potential parent can verify that they are indeed siblings.

Host Multicast (*HM*) [21] aims to provide best-effort multicast delivery service to applications and be compatible with IP multicast to the furthest extent possible. It automates the interconnection of IP multicast enabled islands and provides multicast to multicast-incapable end hosts via unicast tunnels. Multicast islands are connected via UDP tunnels between the Designated Members (DM) where each island elects a DM. The architecture is shown in Fig. 5. The data distribution tree is a shared tree where any member in the group can be a source. HM uses a distributed tree building protocol which scales to the number of group members (O(N) where N is the group size). As shown in Fig. 6, a new member H discovers the root A of the shared tree through the Rendezvous Point (RP). H then sets A as a potential parent and requests for the list of A's children. Among A and A's list of children, H picks the closest one. The distance metric used is the member-to-member RTT and is collected via end-to-end measurements by HM. In this case, D is a new potential parent. H repeats the process down the tree and finds the next potential parent F. In the next iteration, if H finds that F remains the closest to itself, it issues a join request. If F has not breached its degree bound for the number of children, he will accept H's request. Otherwise, H will backtrack by one level and repeat its search. Each member in HM has to maintain information about all members on its path to the root. To improve the data tree, each member will periodically try to find a closer parent by reinitiating the join process from some random member on its root path. HM uses loop detection and resolution mechanism instead of loop avoidance. Loop detection is easily effected as members know its entire root path. To recover from tree partition problem, each member can rejoin anyone in its root path or in its

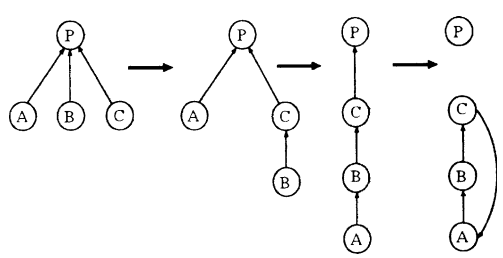


Fig. 4. Switching with outdated information creates loop in BTP [22]. Suppose B switches to C and then A switches to B. If information has not been updated in C and A and C then switches to A, a loop is formed.

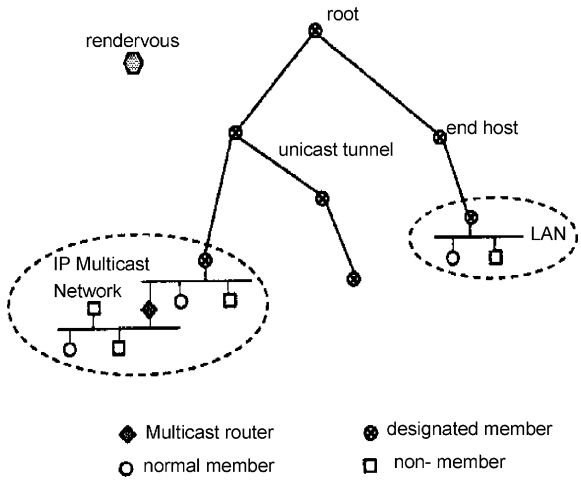


Fig. 5. Host multicast architecture [21].

cache. The cache is built up during the initial process when the member 'walks' down the tree.

Overlay Multicast Network Infrastructure (*OMNI*) [23] constructs a single-source overlay tree from a set of service nodes called multicast service nodes (MSN) deployed in the network as shown in Fig. 7. It is targeted at media-streaming applications. The root MSN is connected to the source. OMNI aims to minimize average latency of the entire overlay tree. Similar to HM, each MSN maintains state for all its tree neighbours, its ancestors and the overlay path from the root to itself. If the minimum out-degree of a MSN is two, and the number of MSNs in the group is N, then it maintains state for at most O(degree + log N) other MSNs. The root path is also used for loop detection during tree optimizations. The overlay tree building procedure comprises an off-line initialization phase before data delivery commences and the dynamic self-organizing process during data delivery. In the offline phase, the root MSN shown as A in Fig. 7 measures and sorts the list of MSNs in increasing order of unicast latencies from itself. It then builds a tree always choosing the nearest MSN to itself at each level of the tree. In other words, in Fig. 7, D–F are further from A in latency terms than B and C. This centralized approach involves O(N) latency measurements.

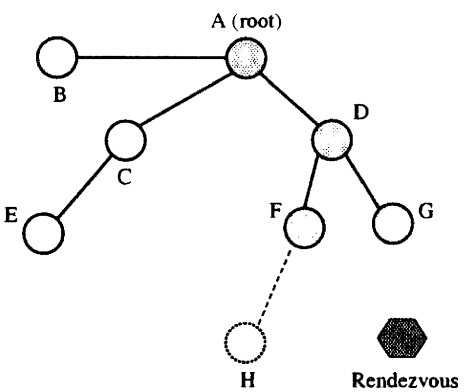


Fig. 6. Join process in host multicast [21].

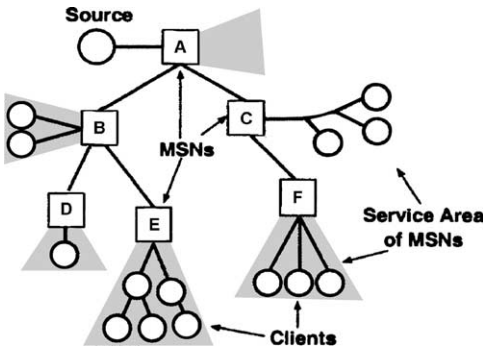


Fig. 7. OMNI architecture [23].

The key feature of OMNI's overlay tree is that it accords a dynamic priority to the different MSNs based on the size of its service set (i.e. the number of clients it serves) and iteratively optimizes the overlay tree. The dynamic self-organizing process is illustrated in Fig. 8. At its initial configuration, the overlay latency from MSN 0 to MSN x is 59 ms (Panel 0). As the number of clients increases to 7 and then to 9, the importance of MSN x increases accordingly. It first changes its parent to MSN 6 (Panel 1) reducing its overlay latency to 54 ms and subsequently to the root MSN (Panel 2) with latency of 51 ms. Thereafter, the number of clients reduces and x migrates down the tree while other MSNs with larger client sizes move up. To adjust the overlay to changing latencies, each MSN performs periodic swapping among themselves if such swapping can reduce their current average latency. This swapping is strictly local in the sense that it is confined to within two levels of each other. OMNI also uses simulated annealing to probabilistically swap one MSN with a random member not within the two levels to allow the overlay to reach a global minima in terms of latency.

Overcast's [11] objective is to maximize bandwidth to the source for all members potentially at the expense of delay increase. It aims to provide scaleable and reliable single-source multicast. It builds a source specific overlay tree which spans proxies nodes deployed across the network called overcast nodes. Overcast's tree building algorithm proceeds by placing a new node as far away from the root as possible without sacrificing bandwidth to the root. The new overcast node first contacts the root of the group. The root thereby becomes the current node. Similar to HM the node will obtain a list of children from the current node. However, unlike HM which measures the latency between itself and the list (including the root), it measures the bandwidth instead. The bandwidth is measured by observing the download time of 10 Kbytes of data. If the bandwidth through any of the children is about as high as the direct bandwidth to the current node, then one of these children becomes the current node and a new round of testing commences. In the case of multiple suitable children, the child closest (in terms of network hops) to the joining member is chosen. If no child is suitable, the search for a parent ends with the current node.

Overlay Tree Building Control Protocol (*TBCP*) [28] aims to provide an efficient and distributed protocol to build control data delivery trees for application level multicasting. It builds overlay trees which are explicitly constrained in that the host fixes an upper-limit on the number of children it is willing to support. The root of the overlay tree is the main sender. It tries to build as good a tree as possible on the outset given the partial knowledge of group membership and restricted network topology information. A new member will be bootstrapped to the root via some well-known registry. Similar to HM and Overcast, the new member will be given a list of its children. The new node will measure the rtt between itself and the given list and the root and send the measured data to the root. The root will test all local configurations to select the one which is optimal. This will require testing of $O((N+1)!)$ configurations where N is the number of children belonging to the root. Fig. 9 shows the possible configurations tested by the root (P) when new member M attempts to join the tree rooted at P with members C_1, C_2 and C_3. If the last configuration is the most optimal, M will be joined to P and C_3 will be redirected to start a join procedure with M assuming the earlier role of P. There is no special mechanism to recover from node failure.

2.2. Mesh-tree

The mesh-tree approach is a two-step design to the overlay topology. It is common for group members to first distributedly organize themselves into an overlay control

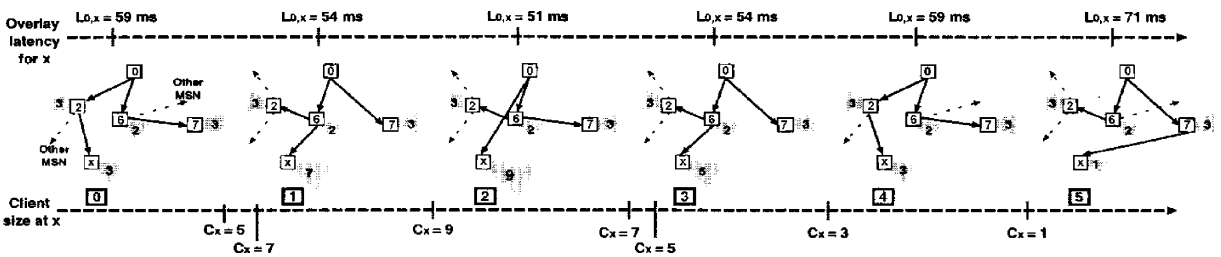


Fig. 8. Dynamics of OMNI as the number of clients change at MSNs. MSN 0 is the root, MSNs 0, 2 and 6 had out-degree bound of 2 each and MSNs 7 and x had out-degree of 3 each. The number of clients being served by MSNs is varied. The relevant unicast latencies (ms) between MSNs are as follows: $l_{0,2} = 29$, $l_{0,6} = 25$, $l_{0,7} = 42$, $l_{0,x} = 51$, $l_{2,x} = 30$, $l_{6,2} = 4$, $l_{6,7} = 18$, $l_{6,x} = 29$ and $l_{7,x} = 29$. c_x indicates the number of clients at MSN x which changes with time. The time axis is not drawn to scale [23].

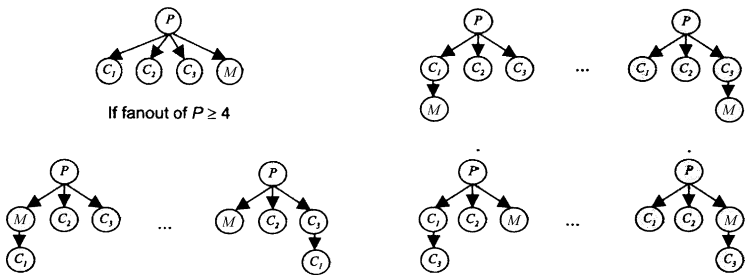


Fig. 9. Local configurations tested in TBCP [28].

topology called the mesh. A routing protocol runs across this control topology and defines a unique overlay path to each and every member. Data distribution trees rooted at any member is then built across this mesh based on some multicast routing protocols, e.g. DVMRP. Compared to tree only design, mesh-tree approach is more complex. However, it has the advantages of avoiding replicating group management functions across multiple (per-source) trees, providing more resilience to failure of members, leveraging on standard routing algorithms thus simplifying overlay construction and maintenance as loop avoidance and detection are built-in mechanisms in routing algorithms.

Narada [8,9] is one of the first application layer multicast protocols that demonstrated the feasibility of implementing multicast functionality at the end hosts. It is targeted to support collaborative applications with small group size. Narada first builds a mesh control topology across participating nodes and subsequently, the data topology is built on top of the mesh by having the group members self-organized into source-rooted multicast trees using DVMRP like routing protocol. This is shown in Panel 0 of Fig. 10. New members obtain a list of group members in the mesh via a rendezvous point (RP) which maintains state about all members joined to the multicast group. The new member randomly selects some of these members as a neighbour under constraint of maximum degree (refer to Panel 2). Each member keeps state about all other members in the group and the routing path and these states are updated via periodic refresh messages exchanges with one another. The aggregate control overhead resulting from the distribution of state messages is very high to the order of $O(N^2)$. This overhead for member discovery and the maintenance of membership in each member limit Narada to support only small to medium group multicast. However, the mesh control topology has the advantages of increased connectivity and is robust to overlay partition and node failure. This is illustrated in Panel 1. To adaptively refine the mesh which directly determines the quality of the data topology, every member periodically evaluates the utility of adding a link to another member of the control mesh and deleting existing links. An example is shown in Panel 3 where adding the edge ⟨J,G⟩ is useful as it results in a large number of shorter unicast paths being created on the mesh (e.g. between member sets {A,C,J} and {G,H}, and between member sets {C,J} and {F}). Edge ⟨A,C⟩ is removed since it has been made less useful after the addition of ⟨J,G⟩. The utility is computed based on the following heuristics:

Adding link: *A member m computes the utility gained if a link is added to member n based on* (*i*) *the number of members to which n improves the routing delay of m*, (*ii*) *how significant this improvement is in terms of delay*.

Dropping link: *The cost of a link between m and n in m's perceptive is the number of group members for which m uses n as next hop and vice versa. These are computed and m will choose the higher of the two as the consensus cost of the link to each of its neighbours. The link with the lowest consensus cost will be dropped*.

Note that original Narada [9] only uses latency as the performance metric while the improved version [10] uses both latency and bandwidth to improve performance. The latter version prioritizes bandwidth over latency by incorporating these metrics into the distance vector protocol running on the mesh. The routing protocol uses a variant of

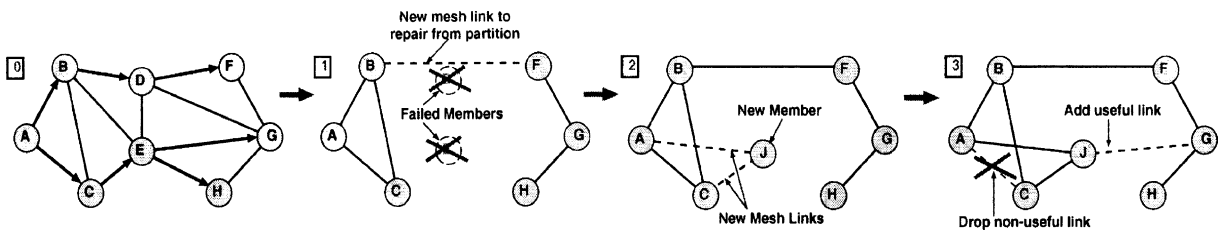


Fig. 10. Control and data topologies in Narada. Neighbours on the control path (mesh) are connected by edges. In Panel 0, the arrowed edges indicate the multicast data path where A is the source. These edges also form part of the mesh. In Panel 1, the departure of D and E leads to a mesh partition. Remaining members, detect the departure and B adds a new mesh link to F hence repairing the partition. In Panel 2, a new member joins the mesh and sets up mesh-neighbour links with two randomly chosen existing members. Periodically members evaluate the utility of different links on the mesh. In Panel 3, a non-useful link is deleted from the mesh and a potentially useful link is added [37].

the shortest widest path algorithm presented in Ref. [38]. Every member tries to pick the widest (highest bandwidth) path to every other member. If there are multiple paths with the same bandwidth, the member picks the shortest (lowest latency) path among these. The utility gain and the consensus cost in the algorithms to add and drop links described are now computed based on the number of members to which performance improves (degrades) in bandwidth and latency if the mesh link were added (dropped) and the significance of the improvement (degradation).

Kudos [27] is an extension of Narada by adding hierarchy in the overlay topology to significantly increase its scaleability. Kudos partitions overlay nodes into clusters and each cluster has a unique representative node, called the cluster head. Fig. 11 shows the two-level hierarchy. All non-head nodes in a cluster are referred to as children. At the bottom level, independent instances of Narada (the version using latency as the metric) is run within each cluster forming an overlay of children. At the top-level, an overlay spans across all cluster heads built using Narada too. In fact, any topology aware overlay building algorithm can be used. The data topology therefore comprises individual trees at each cluster in the bottom level and a tree at the top level. Likewise for the control topology, in an overlay of n nodes, Kudos form approximately $n^{1/2}$ clusters each containing $n^{1/2}$ nodes. The merits of the hierarchy lies in its superior scaleability and low management overhead since measure probes are run across smaller groups and effects of member failures are localized to smaller groups. This is achieved at the cost of efficiency as the children nodes in different clusters are unable to form overlay links to one another. The added complexity in Kudos is the clustering which involves migration, splitting and diffusion of clusters. A node joins any member using some bootstrapping mechanisms and learns of other cluster heads from its current head node. A node in a cluster can migrate to another if its periodic probes to other head nodes yield a significantly lower latency than its latency to its current head node. To reduce the load on head nodes due to such probes, a child will not probe a head node which is more than twice the latency between the child and its current head node. Splitting occurs when a cluster is more than twice as large as $n^{1/2}$. The head node will remain as head for one of the new cluster and selects a new head node from the other new cluster. A head node is chosen to minimize the average latency to all other child nodes in its cluster. As Narada maintains state data of every other member in the overlay, pairwise latency data can be easier obtained for head selection. A cluster which has shrunk in size by more than a factor of two smaller than $n^{1/2}$ due to node migration, death or departure is disbanded. The remaining nodes migrate to other cluster.

Scattercast [19] proposes an application level infrastructure to provide routing and forwarding services and customizable transport framework on top of this infrastructure that leverages application defined semantic to tune the transport protocol for Internet broadcast distribution. It is similar to Narada except for the explicit use of infrastructure service agents, called ScatterCast proXies (SCXs). These proxies which are strategically deployed within the network infrastructure use a protocol called gossamer to self-organize into an application level mesh over which a global routing algorithm is used to construct source-rooted data distribution trees using latency as the routing metric. Fig. 12 shows the scattercast architecture. Clients communicate with SCXs either via locally scoped multicast groups or via unicast. Fig. 13 shows the gossomer overlay topology and illustrates its resilience to failure and the mesh's reusability for multiple source-rooted trees. The initial mesh is randomly built as new SCXs bootstrap themselves via well-known list of rendezvous points and rely on gossip-style discovery algorithm to locate other members and join them as neighbours. The gossip protocol works by having each member periodically and independently picks some random nodes and exchanges membership information thereby propagating information rapidly across the entire system. Optimization of the mesh is performed using latency as the metric based upon which the member decides to accept others as neighbours or to change neighbours according to a predefined cost function and threshold. The cost function of a member is the cost of routing to the various sources via its individual neighbours. A member X will be accepted as a neighbour to Y if its cost function is less than the maximum cost functions among all the existing neighbours of Y. Similar to Narada, every

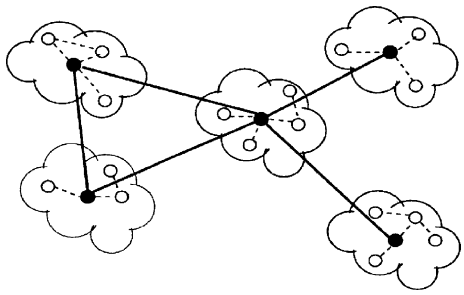


Fig. 11. The two-level hierarchy of Kudos. Clouds represent cluster with the solid nodes as head nodes. The solid lines between the heads is the top level topology [27].

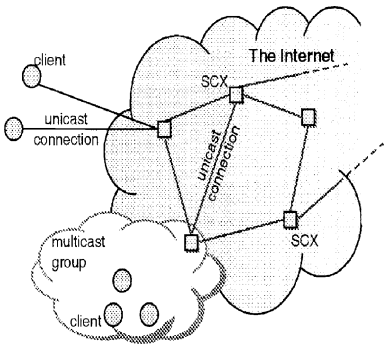


Fig. 12. The Scattercast architecture [19].

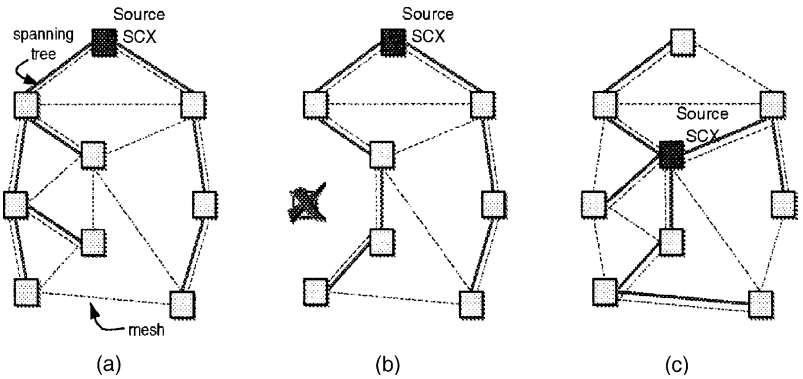


Fig. 13. A typical Gossamer overlay topology. (a) Topology consists of a mesh with a source-rooted spanning tree superimposed atop. (b) The tree-mesh provides resilience to failures by automatically re-routing the tree around a failed SCX. (c) The mesh can be reused for multiple source-rooted trees without excessive additional computation [19].

member must maintain a full state and routing tables to all the other members. Scattercast relies on centralized rendezvous points for repairing mesh partition. Although this assumption simplifies significantly the partition recovery mechanism, the partition problem will still arise in the event of failure of all rendezvous points.

Yoid [13,35] aims to extend the multicast architecture and defines a set of protocols for host-based content distribution either through tunneled unicast connections or IP multicast wherever available. It uses the tree-mesh structure differently from the other proposals categorized here in the sense that it builds a shared data tree directly and creates a mesh later to recover from tree partitions. Yoid's tree building algorithm is similar to BTP in that the first member to join a tree becomes its root and a new member queries a centralized RP which responds with a list of members (called candidate parents) which have already joined the tree. The new member then probes this list of members and decides on its parents based on a performance metric. The RP in Yoid takes on the additional roles of partition healing and group security. Yoid's objective of tree refinement is not so much to optimize performance but to avoid pathologies as its creators believe that the latter goes a long way towards acceptable level of performance. Tree refinement is accomplished through switching parents based on observed loss rates and latencies. For latency measurements, Yoid members periodically query RP for updates on their initial lists of candidate parents. Each node operates tentative links to a subset of this candidate parents list. The average latency difference between the data frames sent via these tentative links is compared to those sent via the data tree. If the latter is higher than the former by a threshold, the node will switch to the candidate parent. Note that these tentative links also serve as ready potential parents for a node to switch to in the event that it is experiencing data losses which are higher than a threshold. Yoid member passively captures data losses from the data it receives and exchanges its loss print (average losses over a period) with its neighbours. Upstream nodes will gather loss prints from its loss print nodes to decide if a downstream node is to be removed to reduce fan-out and improve loss rate. Fig. 14 illustrates such a process.

Loop detection is effected through the presence of route paths in each node of the data tree as per HM. The mesh topology in Yoid is built by having each member maintains a small cache of members (mesh members) which are randomly selected using a frame delivery mode called 'mesh anycast'. In anycast, a discovery message takes a random walk along the mesh and randomly selects

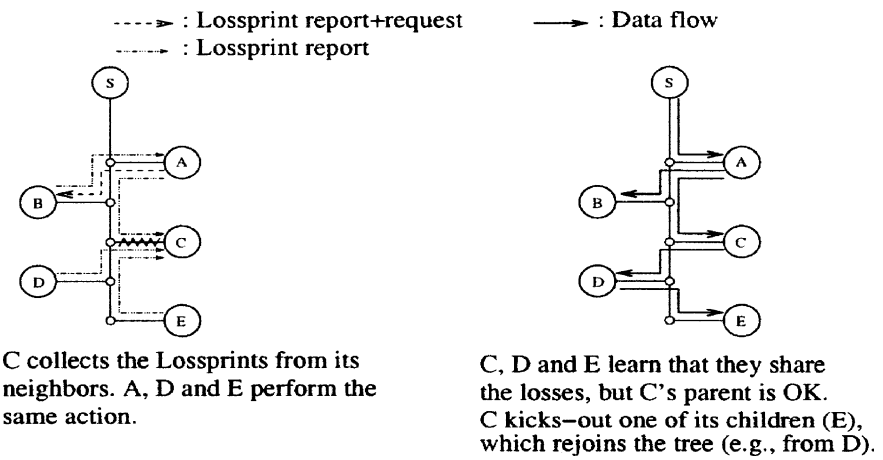


Fig. 14. Loss-rate refinement algorithm for Yoid. C, D, and E reorganize to reduce the load on the congested bottleneck link [13].

a member. Mesh members must be distinct and must not be tree neighbours. These mesh links ensure that partition resulting from node on the tree will be recovered through the additional connectivity.

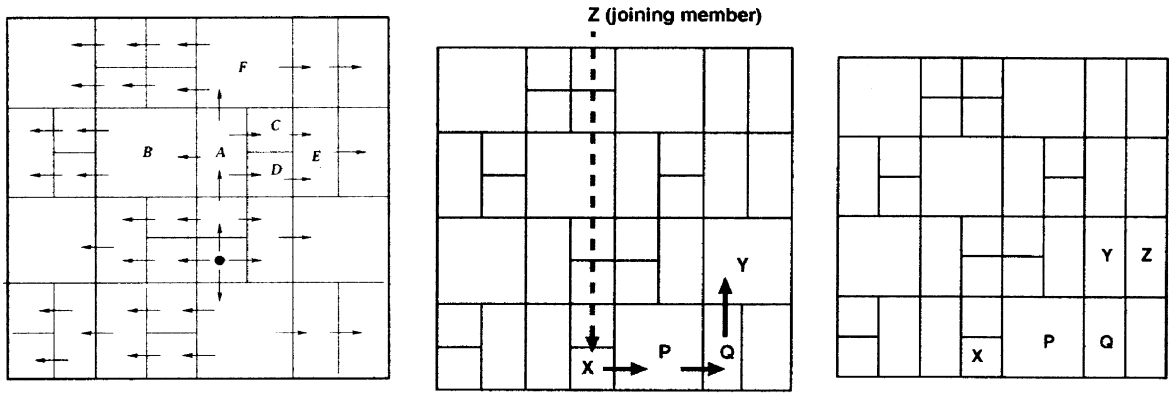
2.3. Embedded structure

Embedded structure refers to proposals who assign to members of the overlay network logical addresses from some abstract coordinate space and builds an overlay with the help of these logical address By embedding neighbour mappings in member addresses, next-hop routing of messages can be performed without the need for full fledged routing protocols such as DVMRP. Moreover, each member needs to maintain knowledge about only a small subset of members enabling the protocols to be highly scaleable. However, in contrast to tree and mesh based approaches, many of these protocols impose rules on neighbour relationships that are dictated by addresses assigned to hosts rather than performance. This may involve a performance penalty in constructed overlays as the overlays do not map well to the physical network topology. Hence a lot of overlay topologies built using embedded structure are also known as topology-agnostic designs with the exception of those which have implicit characteristics which enables them to be close to the substrate network. The following sections will first introduce the topology-agnostic solutions, namely ALM-CAN and ALM-DT and thereafter, proceed to detail the topology-aware proposals.

Application-Level Multicast using Content-Addressable Networks (*ALM-CAN*) [18] makes use of Content-Addressable Network (CAN) [39] architecture to provide an application level multicast solution. ALM-CAN is designed to scale to large group sizes without restricting the service model to a single source. CAN is an application-level structured peer-to-peer overlay network whose constituent nodes form a virtual *d*-dimensional Cartesian coordinate space and each member owns its individual distinct zone in this space. Fig. 15 shows a 2-D coordinate space partitioned into zones by 34 CAN members of which six (A–F) are marked. The black dot represents a source node. Examples of the virtual coordinate zones of A, C, D are: A(2.0–2.5,2.0–3.0), C(2.5–3.0,2.5–3.0), D(2.5–3.0,2.0–2.5). The *d*-dimensional Cartesian coordinate space therefore comprises the control topology of ALM-CAN. The multicast data topology is implicitly defined by performing directed flooding on the control topology (refer to Fig. 15a). No explicit tree construction is required. Each CAN member maintains a routing table to its neighbours as well a packet cache to identify and discard any duplicate packets. Neighbours are defined as members whose zones abut each other. Hence for a *d*-dimensional CAN, a member node maintains state for 2-D additional nodes (its abut neighbours) regardless of the number of source in the group. The data forwarding rule is summarized as follows:

The source forwards a data packet to all its neighbours in the control topology. The receiving neighbours in turn forward the data to all their neighbours except the neighbour from whom it receives the data subject to the condition that the packet has not already traversed at least half-way across the space from the source coordinates along the dimension of data forward. This is to prevent data packets from looping around the overlay. (refer to the example in Fig. 15a where D does not forward data upwards or downwards.)

A new member Z who wishes to join the multicast group queries a RP to find at least one existing member say X. Z then picks a random point in the 2-D coordinate space (say Y's space) and sends a join request through X to locate Y. This is done by routing through CAN as shown in Fig. 15b. Upon locating Y, Y's zone is split into two with Z owing one of them (see Fig. 15c). The split is done by assuming a certain order of dimension so that zones can be remerged when nodes leave. For a 2-D CAN, the split is along the X-dimension first, then the Y and so on. As ALM-CAN uses a topology-agnostic approach to build the control and data topologies without taking the relative distances between nodes into account, their data distribution paths can be very long compared to the physical network



(a)[18] (b)[37] (c)

Fig. 15. Structure of a 2-D CAN and the corresponding control and data topologies.

distance between them. Ratnasamy et al. [18] proposes to ameliorate this situation by using distributed binning by which members that are close to one another are assigned nearby zones in the coordinate space.

Application Layer Multicasting using Delaunay Triangulations (*ALM-DT*) [12] is aimed to support very large multicast group size. It assigns each member a pair of logical (x, y) coordinates in a plane. The coordinates can be assigned via some external mechanisms such as GPS or user input and can be selected to reflect the geographical locations of the nodes. Using these coordinates, the control topology which essentially comprises combinations of DTs [40] is built through angular calculations and comparisons based on the properties of DT. DT has been extensively studied in computational geometry [40]. A DT for a set of vertices A is a triangulation graph with the defining property that for each circumscribing circle of a triangle (see Fig. 16) formed by three vertices in A, no vertex of A is in the interior of the circle. The underlying mechanism used in forming the DT control topology is the neighbour test. The neighbour test is based on the locally equiangular property. Details of this property is provided in Ref. [41]. Two nodes are neighbours in the overlay if their corresponding vertices are connected by an edge in the DT that comprises the vertices associated with all the nodes in the overlay. A new node N requests to join the group by contacting the DT server who bootstraps it to an existing node D. Besides managing group membership, the DT server also helps in repairing partition in the overlay. Node D will perform the neighbour's test on N and if successful, N will become D's neighbour. Otherwise, D route the join request to one of its neighbours whose coordinates are closer to those of N. The process repeats towards the coordinates of the new node N until it passes a neighbour's test. Hence the only state information to be maintained by each node is confined to that of its neighbours.

Once the control topology is built, the source-rooted multicast data tree is embedded in the DT without requiring a routing protocol as packet forwarding information is encoded in the coordinates of a node. Compass routing [42,43] is used to determine the multicast routing tree where nodes calculate their child nodes in the multicast routing tree in a distributed fashion. No loop detection is required as compass routing in DT does not result in loops [43]. Each node can locally determine its child nodes with respect to a given tree using its own coordinates, the coordinates of its neighbours and the coordinates of

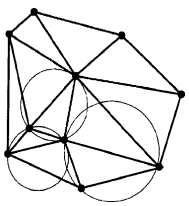


Fig. 16. A Delaunay triangulation [12].

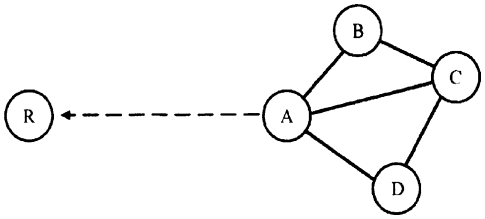


Fig. 17. Compass routing. A determines that it is the parent for C since $\angle RCA < \angle RCD$ and $\angle RCB$. Likewise, B and D determine that they are not the parents of C since $\angle RCA < \angle RCB$ and $\angle RCA < \angle RCD$ [12].

the sender. Hence average overhead of a node is a few kbps in steady state. Fig. 17 shows an example of compass routing. TCP unicast connections are adopted.

DT's merits lie in that it generally has a set of alternate non-overlapping routes between any pair of vertices which can be readily exploited for building optimal data topology and to facilitate recovery when overlay nodes fail. DT can be established and maintained in a distributed fashion where no entities maintain knowledge of the entire group unlike Narada and Scattercast. Experiments conducted showed that it can scale to 10,000 members [12]. Similar to ALM-CAN, its weakness lies in the suboptimal mapping of the overlay to the physical network given the topology-agnostic nature of the logical coordinates of the members.

Bayeux [15] focuses on fault-tolerant packet delivery as a primary goal. It can be used for Internet content distribution and is designed to scale to arbitrarily large receiver groups. The control topology of Bayeux is the Tapestry overlay. Tapestry [36] is a wide area routing and location infrastructure which embeds nodes in a well-defined virtual address space. Nodes have names independent of their locations in the form of random fixed-length bit sequences represented by a common base (e.g. 40 Hex digits representing 160 bits). These can be generated by secure one-way hashing algorithms such as SHA-1 [44]. Tapestry uses similar mechanism to the hashed-suffix

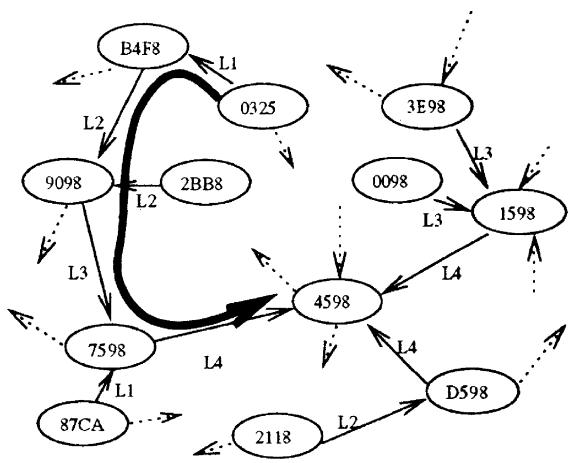


Fig. 18. Tapestry routing example showing the path taken by a message originating from node 0325 and destined for node 4598 in a Tapestry network using hexadecimal digits of length 4 (65,536 nodes in namespace) [15].

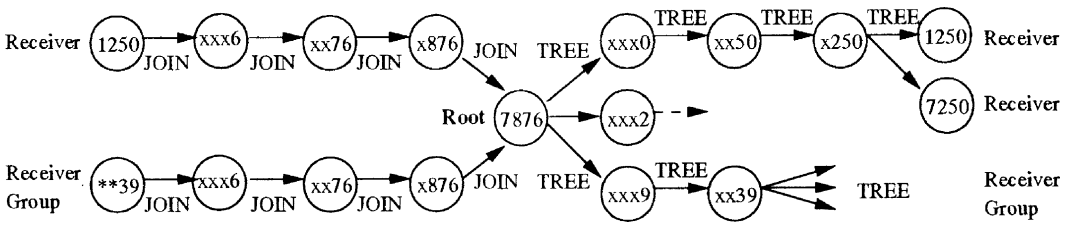


Fig. 19. Bayeux tree maintenance [15].

mesh introduced by Plaxton et al. [45]. It follows from the proof in Ref. [45] that the network distance traversed by a message during routing is linearly proportional to the real underlying network distance. Experiments performed in Ref. [36] verified that this proportionality is maintained with a small constant in real networks. Hence Bayeux's overlay is classified as topology-aware design although it is built via embedded structure approach. Tapestry uses local routing maps (neighbour map) stored at each node to incrementally route overlay packets to the destination ID digit by digit (e.g. ***8 ⇒ **98 ⇒ *598 ⇒ 4598 where * represents wildcards) as shown in Fig. 18. Assuming consistent neighbour map, Tapestry's natural hierarchy ensures that the destination can be reached within at most $\log_b N$ logical hops in a system with an N size namespace of base b. As each neighbour map at a node assumes that the preceding digits all match the current node's suffix, it only needs to keep a small constant size (b) entries at each route level, yielding a neighbour map of fixed constant size $b \log_b N$.

Unlike ALM-CAN and ALM-DT, Bayeux has to explicitly build a source-based multicast tree on top of the Tapestry control overlay. Bayeux's data tree comprises both Tapestry nodes acting as software routers as well as the multicast receivers. A JOIN request by Receiver 1250 in Fig. 19 to join a session is routed by Tapestry all the way to the root node (ID 7876 in Fig. 19). Then the root records the identity of the new member and uses Tapestry to route a TREE message back to the new member. Every tapestry node along this route (xxx0, xx50, x250) adds the identity of the new member to the list of receiver node IDs that it is responsible for and updates its routing table. Requests to leave the group are handled similarly. Bayeux therefore requires nodes to maintain more group membership information and it generates more traffic when handling group membership changes. In particular the root keeps a list of all group members and all group management traffic must go through the root which is not only a bottleneck but also a single point of failure. Bayeux proposes a multicast tree partitioning mechanism to ameliorate these problems by splitting the root into several replicas and partitioning members across them.

NICE [20] proposes a highly scaleable solution designed for low-bandwidth, data streaming applications with large receiver sets. It proposes an extremely low control overhead structure over which different low-latency data distribution paths can be built. The scaleability is achieved by organizing the members into a multi-level hierarchical control topology. Layers are numbered sequentially with the lowest layer being labelled Layer zero (denoted by L_0). Members in each layer are partitioned into a set of clusters. Each cluster is of size between k and $3k - 1$ where k is a constant and comprises members that are close to each other. NICE uses latency between members as the distance metric used in organizing the overlay. Fig. 20 shows NICE's hierarchy using $k = 3$. Note that all members are part of the lowest layer, L_0. A distributed clustering protocol at each layer partitions these members into a set of clusters with the specified size bounds. The protocol chooses a cluster leader which is the graph theoretic centre of the cluster in layer L_i to join layer L_{i+1}. The cluster leader therefore has the minimum maximum distance to all other members in the cluster. This allows new members to quickly locate its position in the hierarchy with minimum queries to other members. A new member is bootstrapped by a RP to the hosts in the highest layer (L_i) of the hierarchy. The joining host contacts these members and selects the one which is closest to it in terms of latency. This selected node will inform the joining host about its peers in layer L_{i-1}. The joining host iteratively identifies the closest member in each layer until it locates its cluster in layer L_0. Hence NICE

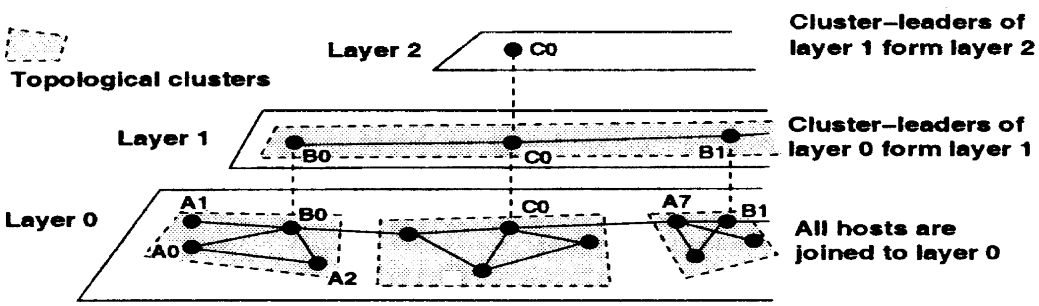
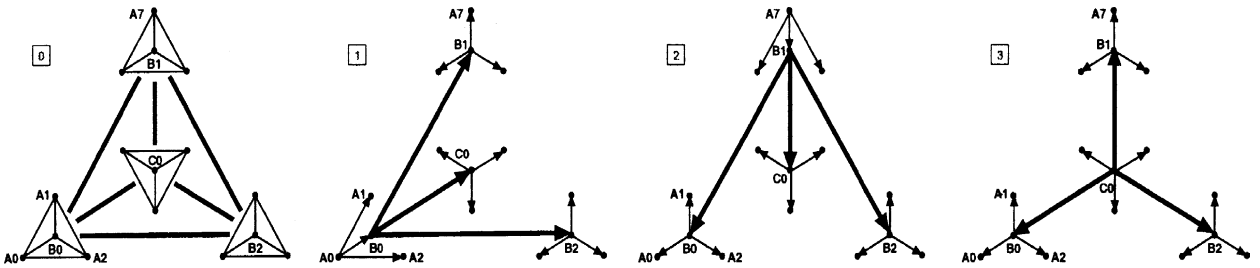


Fig. 20. Hierarchical arrangement of hosts in NICE. The layers are logical entities overlaid on the same underlying physical network.



(a) Control topology (b) Data tree rooted at A_0 (c) Data tree rooted at A_7 (d) Data tree rooted at C_0

Fig. 21. Control and data delivery paths for a two-layer hierarchy. All A_i hosts are members of only L_0 clusters. All B_i hosts are members of both layers L_0 and L_1. The only C host is the leader of the L_1 cluster comprising itself and all the B hosts [20].

is a topology-aware design as it chooses overlay peers based on network locality although its data topology is implicit in the NICE hierarchy. The NICE members hierarchy is used to define both the control and the data topologies. Using cluster size of 4, the control topology is shown in Panel 0 of Fig. 21. The edges indicate the peerings between group members on the overlay topology. Each set of 4 hosts arranged in a 4-clique in Panel 0 are the clusters in layer L_0. Hosts B_0, B_1, B_2 and C_0 are the cluster leaders of these four L_0 cluster and form the single cluster in layer L_1. Host C_0 is the leader of this cluster in layer L_1. In the control topology, all members of each cluster peer with each other and exchange periodic refresh messages incorporating the latencies between them. For example, the control path peers of A_0 in layer L_0 are all the other members in its L_0 cluster (i.e. A_1, A_2 and B_0) while those of B_0 which belongs to layers L_0 and L_1 are all the other members of its L_0 cluster (i.e. A_0, A_1 and A_2) and L_1 cluster (i.e. B_1, B_2 and C_0). In addition, all members periodically probe members in its supercluster (defined as the cluster in the next higher level where its leader belongs to) to identify a cluster leader than its current one so that the member's hierarchy can be improved to adapt to changing network conditions. Cluster leaders are also responsible for splitting and merging clusters to satisfy the cluster size bound. NICE's worst case control overhead is $O(k \log N)$ and the number of application level hops on the basic data path between any pair of members is $O(\log N)$.

The data topology is an embedded source-specific tree defined by a forwarding rule on the control topology without the need to build it explicitly or use any complex routing algorithm. A source sends a data packet to all its peers in the control topology (see Panel 1 of Fig. 21 where A_0 is the source and it forwards data to peers A_1, A_2 and B_0). A receiving host will only forward the data to its peers if and only if it is the cluster leader (see Panel 1 where cluster leaders B_0, B_1, B_2 and C_0 forward data to their respective peers). Panels 2 and 3 show different trees rooted at sources A_7 and C_0, respectively. Loops and partitioning of the data tree may occur due to stale cluster membership data and node failures, respectively. No special mechanisms are introduced except for the members and cluster leaders to restore the hierarchical relationships and reconcile the cluster view for all members.

SCRIBE [16,17] is a large-scale event notification system to disseminate data on topic-based publish-subscribe groups. It builds a source-based multicast shared tree on top of Pastry [46] which is a peer-to-peer location and routing overlay network in the Internet. Pastry uses a circular 128-bit namespace and assigns an ID based on this namespace by basing the nodeId on a secure hash of the node's public key or IP address. Given a message and a destination key, Pastry routes the message to the node whose node ID is numerically closer to the key. This is illustrated in Fig. 22. Assuming a Pastry network of N nodes, the expected number of application level hops is $\log_2^b N$ where b is a configuration parameter with a typical value of 4. With concurrent node failures, eventual delivery is guaranteed unless $l/2$ or more nodes with adjacent nodeIds fail simultaneously (l is an even integer parameter with typical value 16). NodeIds and keys can be interpreted as a sequence of digits in base 2^b. A node's routing table is organized into $128/b$ levels with 2^b entries in each level. The entry in column m at level n of a node p's routing table points to a node whose nodeId shares the first n digits with p's nodeId and whose $(n+1)$th digit is m. The routing table entry is left blank if no such node is known. In addition, each node maintains IP addresses for the nodes in its leaf set. A leaf set is the set of l nodes that are numerically closest to the current node with $l/2$ being smaller and $l/2$ being bigger.

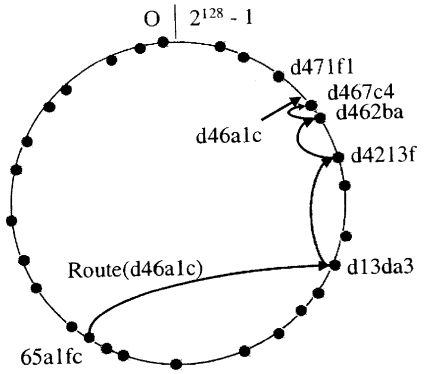


Fig. 22. Routing a message from node 65a1fc with key d46a1c. The dots depict live nodes in Pastry's circular namespace [16].

It also maintains a neighbour set which have members that are close based on the distance metric. The uniform distribution of nodeIds ensures an even population of the nodeId space with only $\log_2^b N$ levels in the routing table populated on average. Moreover, only $(2^b - 1)$ entries per level are populated as one of them is the local node. Hence the average state entries maintained by each node is only $(2^b - 1)\log_2^b N$ entries. Moreover, after a node failure or the joining of a new node, the tables can be repaired by exchanging $O(\log_2^b N)$ messages among the affected nodes. At each routing step, a node forwards the message to a node that shares a longer common prefix with the destination key than its own ID. When no such member is found in the routing table, the message is forwarded to a member in the leaf set that is numerically closer to the destination key than its own ID. Similar to Tapestry, Pastry exploits network locality to reduce routing delays by measuring the delay (RTT) to a small number of nodes when building the routing tables. For each routing table entry, it chooses the closest nodes whose nodeId satisfies the constraint for that entry. Since the constraints are stronger for the lower levels of the routing table, the average delay of each Pastry hop increases exponentially until it reaches the average delay between two nodes in the network.

The control topology of SCRIBE is essentially that of Pastry where the neighbours of any member on the control path include all in its routing table, neighbourhood set and leaf set entries. Neighbours exchange refresh messages with one another. A new node X, joining Pastry is bootstrapped to an existing node A to route a join message using X as the key. It is routed to the existing node Z with nodeId numerically closest to X. X obtains the leaf set from Z and the *i*th row of the routing table from the *i*th node encountered along the route from A to Z thereby initializing its state and notifying neighbours of its arrival. In the event of a failed node, neighbours learn of its demise and update their leaf sets. The recovering node will contact the nodes in its last known leaf set, obtain their leaf sets and updates its own. Each multicast group in SCRIBE typically consists of a subset of the members that have already joined the Pastry control topology and has its own ID (called topic identifier). In other words, a member needs to be joined to the Pastry overlay in order to join a SCRIBE multicast group. SCRIBE creates a multicast tree, rooted at the RP as its data topology. The RP is the node with nodeId closest to the topic identifier and the tree is formed by joining the Pastry routes from each group member to the RP. A Pastry node that wishes to join a SCRIBE group simply routes a join message using the topic identifier as its destination key. All members on this path that are not already part of the multicast data delivery tree for the group add themselves to the tree. SCRIBE node therefore maintains a children table for the group containing an entry (IP address and nodeID) for each of its children in the multicast tree. Upon detection of a tree partition, the SCRIBE node will call Pastry to route a join request to its topic identifier. In the process, Pastry will route the message to a new parent thus repairing the multicast tree.

Table 2
Flat versus hierarchical topologies

Flat			Hierarchical
ALMA	Bayeux	Overcast	Kudos
ALMI	HM	SCRIBE	NICE
ALM-CAN	Narada	Scattercast	
ALM-DT	OMNI	TBCP	
BTP		Yoid	

2.4. Flat topology versus hierarchy topology

A flat topology is one where there is only a single-layer overlay sitting atop the physical network. All nodes in the overlay are in the same logical level. A hierarchical design introduces hierarchy into the overlay forming a multi-layer overlay. Nodes form clusters at the lowest level where each cluster has a leader. The leaders in turn form clusters at the next level and so on. Table 2 shows that Kudos and NICE are the only advocates of hierarchical topologies. The main advantage of a hierarchical overlay lies in its ability to scale and low management overhead since measure probes are run across smaller groups. This is achieved at the cost of efficiency as the children nodes in different clusters are unable to form overlay links to one another and there is additional overhead in maintaining the clusters. As explained above, Kudos builds a 2-level hierarchical overlay while NICE is a multi-level overlay. It is noted that TBCP introduces a form of hierarchy by organizing its members into different domains based on their domain identities to cluster members of the same domain into the same subtree so as to improve tree efficiency. However, as the domain roots do not form another layer of overlay among themselves, TBCP is thus still classified as a flat topology with a single-layer overlay.

3. Classification by service models

By service model, we refer to the manner in which data is being delivered to the multiple destinations, namely best-effort delivery versus reliable delivery and source-specific delivery model (one-to-many) versus any-source delivery model (many-to-many). The service model will determine the type of applications the various multicast techniques can support. For example, real-time broadcast streaming of multimedia data and certain collaborative applications such as audio and video conferencing are tolerant of loss and hence best-effort delivery will suffice while certain types of content distribution such as data replication services will require data integrity and therefore entails reliable transfer. Content distribution usually

belongs to the source-specific model while collaborative applications use the any-source model.

3.1. Best effort versus reliable transfer

Note that the definition of reliable transfer here is more rigorous and refers to end-to-end reliability. In other words, it includes more than just the use of TCP for data transfer. TCP only ensures hop-to-hop reliability which prevents losses due to transmission error and network congestion albeit at the expense of real-time delivery. To recover from error arising from delivery tree transitions and node failures, error recovery techniques such as data duplication and retransmissions must be incorporated. These techniques, as well as hop-to-hop TCP transfer impose a back pressure on the source as the source needs to handle the retransmission requests and rate-limit itself to avoid buffer overflow. In this respect, the receivers in application level multicast techniques can help to ease the pressure on the source if they possess buffering and/or rate-limiting capability as flow control and retransmissions can be handled locally.

It is noted without prejudice that all the proposals covered herein are capable of best effort delivery be it via TCP or UDP transfer with the former being deployed at the expense of real-time performance. Table 3 shows how the various application multicast solutions can be fitted into the different service models based on the above definitions. Only a handful of proposals, namely, ALMI, Scattercast, Bayeux and Overcast incorporated additional mechanisms and protocols to support end-to-end reliable transfer while the rest choose to focus on specific applications and the efficiencies of the overlays. Yoid has included sequence number in the packet header in the Yoid Delivery Protocol specifications for in-order delivery [35] but it has, however, not defined the protocols for lost packet recovery.

ALMI and Scattercast support data delivery via both UDP and TCP. In addition, ALMI proposes out-of-band connection direct to the source for re-transmission for error recovery and in cases where application has buffering capability, retransmission can happen locally. Scattercast uses Scaleable Reliable Multicast (SRM) [32] in addition to TCP for hop-to-hop reliability. To recover lost data, a receiver issues recovery requests which get forwarded from its local proxy (called SCX) through other SCXs towards the source until one of the SCX can recover the data either from itself or its local group. It uses a scheme based on PGM (PraGmatic Multicast) [33] and BCFS (Bread-Crumb Forwarding Service) [34] to limit the scope of retransmitted data so that data is only re-transmitted to those links which have requested for it. The links are kept as soft state in each SCX.

Bayeux leverages the redundant paths to every destination provided by Tapestry, an overlay location and routing layer presented by Zhao et. al. [36] for end-to-end reliability. It proposes a suite of protocols which include 100% data duplication on the first backup route to selective duplication based on the probability of successful delivery of each link. Other protocols do away with data duplication by either using heuristics to choose the best outgoing path for each packet or predicting the shortest and most reliable outgoing link to the next hop Tapestry router. To date, Bayeux has adopted the latter approach to minimize overheads due to data duplication.

Overcast provides reliable data transfer to users via TCP over its overlay of proxy nodes (called Overcast nodes) that reside within the network infrastructure. Overcast node performs store-and-forward and failure recovery operations. It maintains a log of the data it has received and stored, and upon failure recovery, it resumes on-demand distribution from where it leaves off.

Table 3
Classification by data transfer reliability

Best effort delivery			
Without complete end-to-end reliable transfer mechanisms			Reliable delivery
ALMA	Kudos	SCRIBE	ALMI
ALM-DT	Narada	Yoid	Scattercast
ALM-CAN	NICE	HM	Bayeux
BTP	OMNI	TBCP	Overcast

3.2. Source-specific versus any-source delivery

Table 4 shows the classification of the various techniques into source specific versus any-source models. The former builds data delivery tree rooted at a single source with one tree for each source while the latter builds a group-shared tree where any members in the tree can be a source. The shared tree is usually rooted at the dominant source though it can be rooted at any node as defined by the respective tree-building algorithms. Source-specific trees are simple and are known to have latency advantage over group-shared trees although routing states associated with source-specific tree grow linearly with the number of data sources as this approach needs to deal with the overhead of maintaining and optimizing multiple trees. This is, however, less of a concern on end hosts than on routers. On the other hand, group-shared trees reduce routing inefficiency and incur less overhead in tree building and management but are not optimized for the individual source and are susceptible to a central point of failure.

Table 4
Classification by single-source versus any-source model

Source specific model		Any-source model	
ALMA	NICE	ALMI	HM
Bayeux	OMNI	ALM-DT	SCRIBE
Kudos	Overcast	ALM-CAN	TBCP
Narada	Scattercast	BTP	Yoid

The service models adopted by the various proposals are largely influenced by the targeted primary applications. ALMA, Bayeux, Narada, Kudos, NICE, OMNI, Overcast and Scattercast are aimed at media streaming and broadcast applications which are characterized by a single source. These proposals therefore fully exploit this characteristic to produce simple and efficient data delivery trees. Proposals such as ALMI are primarily targeted at collaborative applications where there can be more than one source per group while the rest are more generic in their service models. Hence, if the applications do not entail multiple sources, the proposals classified under the any-source model perform as per those in the source specific model. Any-source model is hence a superset of the single-source model.

4. Classification by architecture

Architecturally, proposals for overlay based multicast can be distinguished by whether they assume a peer-to-peer architecture or an infrastructure (proxy) based architecture. A peer-to-peer (P2P) approach constructs the overlay across end-users with all functionalities being vested with these end-users. A proxy based approach makes explicit use of infrastructure service agents or proxies or dedicated servers strategically deployed within the network to provide efficient data distribution and value-added services to a set of end hosts. The overlay is built across these proxies and end hosts attach themselves to proxies with the proxies operating on their behalf.

The diverse application layer techniques can also differ in the approaches adopted for the overlay creation and maintenance. This will determine the presence and role of a central controller in their architectures. The functions of this controller include being a centralized tree construction server, a centralized directory of all members, managing group membership such as join and leave processes and performing partition recovery.

4.1. Peer-to-peer versus proxy support

P2P architectures have the attractive property of scaleability. This is because being distributed, each peer needs only to keep state for a small number of peers. Its other advantages include the simplicity of set-up and deployment, its resource sharing capability as well as dynamicity. Their vast combined resources such as physical connectivity, computing resources may be heterogeneous but they can be individually harnessed according to their capacities. P2P systems thus provide redundancy as a single failure will not radically affect the big group. Peers can be dynamically deployed in large numbers and at hot spots quickly with minimum prior configuration.

The merits of using proxies as opposed to any end hosts lie mainly in their functional dedication. Being dedicated, homogeneous and better provisioned than individual hosts, proxies are thus more reliable and robust to failure. Proxies are persistent beyond the lifetime of individual hosts. They are more intelligent than end-hosts as they can provide value-added services such as being pre-configured with application specific components to render them application-aware. In addition, they can be positioned at strategic positions such as co-locating with IP routers or at hotspots to provide more efficient services. However, there may be problems with the acceptance and deployment of such proxy, and when compared to P2P systems, it is less responsive to changing environment conditions as proxy placement is usually static and has to be manually deployed. Table 5 distinguishes the P2P systems from their proxy based counterparts.

Table 5
Classification by the need for proxy support

Peer-to-peer based		Infrastructure support
ALMA	HM	Bayeux (uses Tapestry nodes)
ALMI	Narada	OMNI (uses Multicast Service Nodes, MSNs)
ALM-DT	NICE	Overcast (uses Overcast nodes)
ALM-CAN	SCRIBE	Scattercast (uses ScatterCast proXies, SCXs)
BTP	Yoid	
Kudos	TBCP	

OMNI deploys proxies called Multicast Service Nodes (MSNs) across the network. It is aimed at large scale data distributions where there is a single source, e.g. a webcast. Prior to the commencement of the webcast, the MSNs will organize themselves into an initial data delivery tree where clients join and leave the tree dynamically. The root MSN is the one that is connected to the source. This tree of MSN will transform itself to adapt to changing network conditions. Overcast organizes a set of proxies called overcast nodes into a distribution tree rooted at a central source for single source reliable multicast. A distinguishing feature of Overcast is the explicit provision of permanent storage capacities in the network fabric accomplished through its proxies to support applications such as broadcasting and reliable data-delivery applications where delay is not a concern. Scattercast uses its ScatterCast proxies (SCXs) to build an intelligent, application-aware proxy infrastructure to provide a customizable framework for efficient broadcasting. Application specific components such as rate adaptation are incorporated into these SCXs to provide more application functionality. SCX also serves to repair partitions in the overlay network. Clients are either statically configured with the location of their closest SCX or locate their SCX via a registry or dynamically via DHCP. Bayeux has a unique feature in that the overlay built comprises both end hosts as well as infrastructure nodes called Tapestry nodes [36]. These reliable servers are installed across the network to act as software routers with multicast functionality.

A point to note is that for HM, only designated members (DM) are involved in the building of the shared tree.

The DM can be a single host or a member which is elected to act on behalf of the members in a multicast island. The DM communicates with its island members via local multicast. Hence, some researchers may consider DMs as proxies and classify HM as a proxy-based system. However, in this paper, we classiy HM as a peer-to-peer system as the DMs are not dedicated server nodes but are specially designated end hosts.

4.2. Centralized controller, distributed approach or hybrid

A centralized approach to the overlay tree creation problem refers to the vesting of all responsibilities for group management, overlay computation and optimization with a central controller. The controller maintains group information, handles group membership, collects measurements from all members, computes optimal distribution tree and disseminates the routing tables to all members. The main advantage of a centralized controller is in the great simplification of the routing algorithms, the more efficient and simpler group management and the provision of a reliable mechanism to prevent tree partitions and routing loops. However, the centralized nature limits its scaleability and poses other reliability problem as it is more susceptible to a central point of failure while it helps to alleviate problems of tree partitions and looping.

As the name implies, distributed approach is receiver-based and it distributes the responsibilities of group membership and overlay topology computation to the individual nodes of the overlay network. It is therefore relatively more robust to failure as failure of an individual node will not impact the entire group. Having no central controller as a potential bottleneck, the distributed approach is thus more scaleable. However, a fully distributed approach causes excessive overheads and is not as optimal and efficient in building optimal overlay.

A hybrid approach is one where a lightweight controller is still used to facilitate some of the group management (e.g. join process), overlay construction and error recovery functions while the actual overlay construction is left to a distributed algorithm. Table 6 shows the classification of the techniques via their incorporation of central controller into the architecture.

ALMI is by far the only technique which advocates a completely centralized solution where a session controller has total knowledge of group membership as well as of the characteristics of a mesh connecting the members. New member only needs to learn of the controller from either online or offline means such as URL, email, etc. and simply issues a join request to the controller. Mesh characteristics are collected from periodic active probes sent by members to measure application level performance metric which in this case is round trip time. Based on this knowledge the session controller builds a minimum spanning tree and disseminates the routing tables to all the members. The use of a centralized approach simplifies the tree building process and enables the ALMI data distribution trees to be close to source-rooted IP multicast trees in efficiency with low performance tradeoff. Moreover, it makes for better reliability as tree partition and looping can be avoided and reduces overhead during a change of membership or recovery from node failure as these are more effectively managed by a central entity. However, the centralized nature of this architecture limits its scaleability and hence it is targeted at collaborative applications and scales to large number of sparse groups. The session controller also constitutes a single point of failure.

Table 6
Classification by centralized, distributed or hybrid approach

Centralized	Distributed		Hybrid
ALMI	ALM-CAN	NICE	ALMA
	Bayeux	Overcast	ALM-DT
	BTP	Scattercast	HM
	Kudos	SCRIBE	OMNI
	Narada	TBCP	Yoid

Those proposals classified under distributed approach in Table 6 adopt a distributed algorithm for overlay construction and dispense with any centralized controller or server. However, they do require some form of bootstrap mechanisms for a host to learn of some nodes in the multicast group or to learn of the root of the data delivery tree. Bayeux advertises the root nodes of its data delivery trees in the network via Tapestry's location services and its root nodes handle all join and leave requests. BTP, Kudos, Narada assume the availability of some out-of-band bootstrap protocol. Overcast nodes (proxies) who wish to join the overlay is bootstrapped via a global well known registry. Clients who wish to join the overcast session discover the root via a web registry. The root node selects the best overcast proxy to serve the clients based on the status information in its database and performs the redirection. SCRIBE requires a *contact node* in the overlay to bootstrap the join process while NICE and TBCP need a rendezvous point to learn of the highest level nodes in NICE's hierarchy and to advertise the root of the TBCP's shared tree, respectively. Scattercast provides a well-known list of rendezvous SCXs for a new SCX to bootstrap itself and then relies on gossip-style discovery to locate other SCXs. For clients joining a Scattercast session, they are statically configured with the location of its closest SCX. ALM-CAN requires a bootstrap node which maintains a partial list of members.

ALMA, ALM-DT, HM and Yoid are examples of a hybrid approach whereby a light-weight controller is incorporated in the architecture while the overlay construction algorithms are distributed. ALMA, ALM-DT and Yoid use a centralized server as a rendezvous point for new members to join the group. Membership status is maintained at these rendezvous points. In addition, ALM-DT and Yoid make use of the server to recover from overlay partition

when a member leaves the group. As for ALMA, other mechanisms, namely, dynamic gossip and spirals, have been incorporated to recover from tree partition problems [24,25]. Reverting to the centralized server is a last recourse for ALMA in the event that these mechanisms failed simultaneously. HM employs a dedicated Host Multicast Rendezvous Point (HMRP) where new members can learn about the roots of the shared trees and bootstrap themselves. The roots stored in the HMRP are kept current through periodic refreshes. OMNI operates quite differently from the other techniques. As described in section 2.1, OMNI builds an overlay across its proxies called MSNs. Prior to a session, OMNI must execute a centralized initialized procedure. The source will first be linked to a MSN identified by some bootstrap mechanism. This root MSN gathers the latency measurements between itself and the other MSNs and use this knowledge to build the initial data delivery tree and distributes the topology to the member MSNs. This centralized computation of the tree building algorithm does not impact the data delivery as it operates off-line before data delivery commences. The subsequent optimization process for the data delivery tree is performed in a distributed manner without central intervention.

5. Performance comparison

As the application layer multicast techniques reviewed here vary widely in their goals, designs, performance evaluation metrics and evaluation strategies, it is not possible to evaluate all of them comparatively. Neither is it possible to examine all the performance evaluation metrics exhaustively. Instead, this paper focuses on the few more common evaluation metrics adopted and will provide comparative data wherever available. The evaluation metric covered here includes scaleability measured intuitively in terms of the size of the multicast receivers it can support, the protocol efficiency in terms of the quality of data paths measured, control overheads, amount of state information to be maintained at each member node and failure tolerance.

5.1. Scaleability

The scaleability of a solution is limited by the protocol efficiency which comprises factors such as control overheads, time and resources used to construct the application layer multicast overlay, the amount of state kept by each node to maintain the overlay. An intuitive but simple approach which can provide a quick insight into the scaleability of each technique is via the number of receivers each of the proposals can support. This data is readily provided in all the research proposals. Table 7 summarizes their scaleability. However, a point to note is that a system which scales to small group may not necessarily be viewed as inferior to one which scales to a large group as each can serve the different needs of different applications. The group size supported is banded into the following:

Table 7
Scaleability comparison in terms of the group size supported

Small	Medium	Large	Very large
ALMI	ALMA	ALM-CAN	ALM-DT
	BTP	Bayeux	NICE
	HM	Kudos	
	Narada	OMNI	
	TBCP	Overcast	
	Yoid	Scattercast	
		SCRIBE	

- small (several tens of members);
- medium (several hundreds of members);
- large (several thousands of members);
- very large (tens of thousands of members).

5.2. Efficiency of protocols

There are many different parameters for evaluating the efficiency of the various proposals. Here we are interested in the quality of data paths generated, the amount of overhead network traffic that is attributable to overlay maintenance, network probing and other control functions, amount of state maintained by each node and the robustness to failure.

5.2.1. Performance metrics

The quality of data path is commonly evaluated using two metrics:

- *Stretch* also termed as *Relative Delay Penalty* (*RDP*) is a measure of the increase in latency that applications incur while using overlay routing. It is the ratio of a protocol's unicast routing distances to IP unicast routing distances. Assuming symmetric routing, IP multicast and naïve unicast both have a Stretch or RDP of one.
- *Stress* is a measure of how effective a protocol is in distributing network load across different physical links. It is defined on a per link or per node basis and counts the number of identical packets sent by the protocol over that link or node. IP multicast has no redundant packet replication and hence has a stress of one while naïve unicast has a worst case stress equal to the number of receivers.

In general, it is difficult to analytically compute and quantify the stretch or stress metrics for most protocols as most of these results are obtained either via simulation or empirically and they are all presented graphically. The performance also varies according to group sizes and the characteristics of the underlying topology. Nevertheless, by examining their plots, all of them have shown reasonable to good performance in either stretch or stress or both depending on their optimizing goals. To provide some feel

on the quality of the data path, the *Path Length* measured in terms of the number of application-level hops and the *Outdegree* of a node on the data delivery tree are metrics which are indirectly related to the stress and stretch metrics and can be easily analysed for some protocols. The *Path Length* is an indicator of the stretch of the protocol while the *Outdegree* provides an upper bound for the number of times that a data packet is forwarded at a node which contributes directly to the stress of the entire overlay.

Protocol Overhead refers to the total bytes of non-data traffic that enters the network. This overhead includes control and maintenance traffic required to maintain the overlay and state management as all application layer multicast solutions require nodes to exchange refresh messages with its neighbours, the probe traffic and other active measurements performed during the overlay self-organization and maintenance process. The average overhead per node is commonly used as an indicator of the scaleability of the protocol.

Another metric that is closely related to protocol overhead is the amount of information kept by each node on the overlay. Herein lies their relation. *State Information* maintained in each node has to be periodically updated to reflect dynamically changing network environment. This implies that a node maintaining more state information will generate more update messages, thereby contributing to additional protocol overhead. This metric therefore directly impacts the scaleability of a technique. Information maintained by each node includes routing tables and group state. Protocols which require the full set of member state to be stored in each node are thus not as scaleable as those which only maintain partial group member state.

Failure Tolerance is actually a multi-factorial function, being dependent on the various mechanisms put in place for error and failure recovery; whether the system is proxy-based or peer-to-peer based; whether the system relies on dedicated lower-level infrastructure support; whether the protocols are prone to a single point of failure or the failure can be contained and localized. Section 2 has detailed the mechanisms put in place for error and failure recovery in the various overlay topology designs while Section 4.1 has addressed the facts that proxy-based system is inherently more robust than peer-to-peer based system. It is acknowledged that application level multicast techniques are certainly not as robust as one based on IP multicast with lower-level dedicated infrastructure support since the end-hosts are not as well provisioned, persistent and reliable as the infrastructure set up by the Internet Service providers. This fact, notwithstanding, to facilitate a single-dimension evaluation of application level multicast solutions without being intertwined into the convolutions of the different impacting factors listed above, *Failure Tolerance* is simply defined herein as whether the protocols are prone to a single point of failure or the failure can be contained and localized. It is important to highlight that the single point of failure in the context of application level multicast protocols is not as catastrophic as it sounds. This vulnerable point refers to the centralized node or dedicated controller used by the two protocols to manage overlay construction and membership. It does not refer to the source of the multicast session as in the event that the source fails, there will be no session at all. Hence, the failure of this centralized point only prevents new members from joining the groups but does not incapacitate the entire system. Its impact on members who have already joined the groups is minimal or none depending on the respective algorithms. Hence all members and sessions in progress continue to function. Such protocols will generally strengthen this critical point with server clusters or replicated servers to minimize the probability of failure and to recover as soon as possible. Localize failure refers to failure which only impacts the failed node and its immediate neighbours and does not impact any join processes.

5.2.2. Comparison and evaluation

Table 8 shows a summary of the performance of the various protocols in terms of the performance evaluation metrics spelt out above.

Stretch: As discussed in Section 5.2.1, *path length*, which is derived from the number of application level hops, provides a good indication of the relative maximum *Stretch* of the various techniques. From Table 8, it can be observed that the number of application level hops is only bounded for those techniques which use embedded structure to construct their overlay. Tree and mesh-tree techniques possess unbounded hops and hence are likely to have longer maximum stretch than their embedded structure counterparts. The exception is OMNI which predetermines the number of nodes during its initialization phase prior to data delivery.

Stress: *Outdegree* provides an upper bound for the number of times that a data packet is forwarded at a node which contributes directly to the stress of the entire overlay. Similar to the performance for stretch, embedded structure has an intrinsic bound for *maximum outdegree per node* whereas tree and mesh-tree approaches have to impose a degree constraint to minimize stress and optimize bandwidth. This constraint can take the form of a static user defined outdegree (e.g. ALMA, ALMI, HM, Yoid) or a more dynamic one where users (Narada) or proxies (OMNI, Scattercast) set the degree constraint according to its prevailing bandwidth. The exception is BTP and Overcast where there is no degree constraint imposed.

Protocol Overhead and State Information Maintained. The overhead introduced by the protocol in the form of non-data traffic entering the overlay network is normalized as the *average control overhead per node* in Table 8 to facilitate comparison among the different techniques. This is also directly related to the *number of state entries per node* as the more states a node keeps, more updates are required to maintain its currency thereby impacting the amount of control information needed to maintain, improve and repair

Table 8
Summary of protocol performance

Techniques	Path length	Maximum outdegree	Average control overhead per node	No. of state entries per node	Failure tolerance	Group size
ALMA	Max: unbounded	User defined	O(max. deg.)	O(max. deg.)	Single point of failure at DS	Medium
ALMI	Max: unbounded	User defined	O(N)	O(N)	Single point of failure at central controller	Small
ALM-CAN	Max: O($dN^{1/d}$), Avg: $d/4N^{1/d}$ [a]	Constant of 2d	Constant of 2d	Constant of 2d	Localized failure	Large
ALM-DT	Avg : $\sqrt{N/4}$	Worst: O($N-1$),[b] Avg: approx. 6	Constant of 6	Constant of 6	Single point of failure at DT server	Very large
Bayeux	Max: O($\log_b N$)[c]	O($\log_b N$)	O($\log_b N$)	O($b \log_b N$)	Single point of failure at root	Large
BTP	Max: unbounded	Unconstrained	O(N)	O(N)	Localized failure	Medium
HM	Max: unbounded	User defined	O(max. deg.)	O(max. deg.)	Single point of failure at RP	Medium
Kudos	Max: unbounded	O($\sqrt{N}$)	O($\sqrt{N}$)	O($\sqrt{N}$)	Localized failure	Large
Narada	Max: unbounded	User defined to reflect bandwidth of user's outgoing link	O(N)	O(N)	Localized failure	Medium
NICE	Max: O($\log N$)	Between k and $3k$[d]	Constant of O(k), Max: O($k \log N$)	Constant of O(k), O($\log N$) only for the highest hierarchical layer	Localized failure	Very large
OMNI	Max: O($\log N$)	MSN defined to reflect bandwidth of MSN's outgoing link	O(max. deg + $\log N$)	O(max. deg. + $\log N$)	Localized failure	Large
Overcast	Max: unbounded	Unconstrained	O(N)	O(N)	Single point of failure at root	Large
Scattercast	Max: unbounded	User defined to reflect bandwidth of user's outgoing link	O(max. deg.)	O(max. deg)	Localized failure	Large
SCRIBE	Avg: O($\log_2^b N$)	O($\log_2^b N$)	O($\log_2^b N$)	O($(2^b-1)\log_2^b N$)	Localized failure	Large
TBCP	Max: unbounded	User defined	O(max. deg)	O(max. deg)	Localized failure	Medium
Yoid	Max: unbounded	User defined	O(max. deg)	O(max. deg)	Central point of failure at RP	Medium

N is the number of nodes in the overlay.
[a] d is the dimension of the Cartesian coordinate space of a CAN overlay.
[b] Theoretically, worse case is created when $N-1$ vertices form a circle and the nth vertex is in the center of the circle. However, the maximum outdegree is very small [12].
[c] b is the base used in the Bayeux node ID.
[d] k is a constant.

the overlay. These two metrics actually affect the scaleability of the systems and as reflected in Table 8, the highly scaleable systems with embedded structures have a non-linear bound on these control and state overheads. In other words, the overheads do not increase proportionately with the increase in the number of nodes (e.g. Bayeux, SCRIBE). In fact, such overheads and state data are constant in the case of ALM-CAN, ALM-DT and NICE which is a primary reason for their superior scaleability. Mesh-tree systems such as Narada fare the worst as it has to maintain state for the entire group. Likewise for centralized algorithm based ALMI and unconstrained outdegree protocols such as BTP and Overcast. Whereas, the other tree-based algorithms with their imposed outdegree constraint, manage to cap these overheads to an upper bound.

Failure tolerance. The fully centralized ALMI as well as most hybrid systems, namely, ALMA, ALM-DT, HM and Yoid are considered less robust due to their reliance on some form of centralized entity for group management functions and recovery from data distribution tree partitioning. To alleviate the impact of a failure in the centralized entity, HM and Yoid propose the use of a cluster of servers while ALMI uses multiple hot standby back-up servers. These standbys receive periodic updates from the primary controller so that they are ready to phase into operation once the primary controller fails. These approaches, however, often suffer from communication overheads and data consistency problems. OMNI, unlike its hybrid counterparts, does not face similar problems as its centralized entity is only involved in building the data distribution tree during the initialization phase. Its operation ceases once data distribution commences.

It is interesting to note that distributed techniques too have their Achilles heels such as the roots of the data distribution

Table 9
Summary of the properties of the various milestone application level multicast techniques

Techniques	Goals	Topology-awareness	Topology design	Topology hierarchy	End-to-end reliable delivery	Source model	Proxy	Type of algorithm
ALMA	Min. latency, min. loss	Aware	Tree	Flat	Best effort	Specific	P2P	Hybrid
ALMI	Min. tree cost	Aware	Tree	Flat	Reliable	Any-source	P2P	Centralized
ALM-CAN	Min. latency	Agnostic	Embedded	Flat	Best effort	Any source	P2P	Distributed
ALM-DT	Min. overhead, max. scaleability	Agnostic	Embedded	Flat	Best effort	Any source	P2P	Hybrid
Bayeux	Fault tolerant delivery, min. latency	Aware	Embedded	Flat	Reliable	Specific	Tapestry nodes	Distributed
BTP	Min. tree cost	Aware	Tree	Flat	Best effort	Any source	P2P	Distributed
HM	Min. tree cost	Aware	Tree	Flat	Best effort	Any source	P2P	Hybrid
Kudos	Max. scaleability	Aware	Mesh-tree	Two-level hierarchy	Best effort	Specific	P2P	Distributed
Narada	Min. latency, Max. bandwidth	Aware	Mesh-tree	Flat	Best effort	Specific	P2P	Distributed
NICE	Min overhead, max. scaleability	Aware	Embedded	Multi-level hierarchy	Best Effort	Specific	P2P	Distributed
OMNI	Min. latency	Aware	Tree	Flat	Best effort	Specific	MSNs	Hybrid
Overcast	Reliable delivery, Max. bandwidth	Aware	Tree	Flat	Reliable	Specific	Overcast nodes	Distributed
Scattercast	Min. latency	Aware	Tree	Flat	Reliable	Specific	SCXs	Distributed
SCRIBE	Max. scaleability	Aware	Embedded	Flat	Best effort	Any source	P2P	Distributed
TBCP	Max. responsiveness	Aware	Tree	Flat	Best effort	Any source	P2P	Distributed
Yoid	Min. latency	Aware	Tree-Mesh	Flat	Best effort	Any source	P2P	Hybrid

trees in the case of Bayeux and Overcast. In both Bayeux and Overcast, the root handles all join and leave requests from session members. To ameliorate the problem of the root as a single point of failure, root replication is incorporated. Bayeux creates multiple root nodes and leveraging on Tapestry's location services, receivers will be routed to the closest local root in network distance. Receivers are therefore partitioned into disjoint membership sets transparently. The technique is similar to root replication used by many existing IP multicast protocols such as CBT [47] and PIM [48,49]. Bayeux's root replication does not incur additional overhead in the sense that these replicated roots do not need to send periodic advertisements to all receivers as they can be located via Tapestry location services. Overcast uses a technique called linear roots to replicate its root. Starting with the root, some number of nodes are configured linearly, i.e. each node has only one child. All the other overcast nodes lie below this set of linear roots. These replicated roots are updated with all the information needed to handle the join operations in the event that the root node fails. The drawback of this technique is the increased latency of content distribution as data has to traverse all these extra nodes before reaching the rest of the overcast nodes as well as data consistency problems.

6. Conclusion

This paper surveys the milestone application level multicast techniques documented in the various literatures and classify them into different categories based on their topology design, service models and architecture. Table 9 summarizes their properties to facilitate a quick insight and understanding of their contributions. In addition, a performance comparison of the various techniques based on factors such as scaleability and efficiency of the protocols in terms of stress on the underlying network, relative delay penalty, protocol overheads and failure tolerance has been conducted. The results are presented in Table 8. In summary, the different techniques have been designed to cater to their different goals with their respective strengths and weaknesses as discussed in this paper. Techniques based on embedded structures tend to enjoy superior scaleability with the drawback that they may not be as network topology aware as those based on tree architecture. Tree-based techniques, being more topology aware, are better able to exploit underlying network topology for efficient data distribution except that they are less robust. Tree-mesh architectures serve to strengthen the robustness of the data distribution trees but at a cost of higher overheads. To the best of our knowledge, this paper is one of the first papers to provide a comprehensive survey of the various milestone research work in application level multicast in terms of both breadth and depth.

References

[1] S. Deering, D. Cheriton, Multicast routing in Datagram internetworks and extended LANS, ACM Trans. Comp. Syst. 8 (2) (1990) 85–100.

[2] S.E. Deering, Multicast routing in a Datagram internetwork, PhD thesis, Stanford University, December 1991.
[3] S. Deering, D. Estrin, D. Farinacci, V. Jacobson, C. Liu, L. Wei, The PIM Architecture for Wide-Area Multicast Routing, IEEE/ACM Trans. Networking December (1997) 784–803.
[4] C. Diot, B.N. Levine, B. Lyles, H. Kassan, D. Balensiefen, Deployment issues for the IP multicast service and architecture, IEEE Networks Spec. Issue Multicasting 14 (1) (2000) 78–88.
[5] R. Perkman, C. Lee, A. Ballardie, J. Crowcroft, Z. Wang, T. Maufer, C. Diot, J. Thoo, M. Green, Simple multicast: a design for simple, low-overhead multicast, IETF draft, draft-perlman-simple-multicast-03.txt, October 1999.
[6] H. Hoolbrook, D. Cheriton, IP multicast channels: EXPRESS support for large-scale single source applications, Proc. ACM SIGCOMM September (1999) September.
[7] H. Hoolbrook, B. Cain, Source specific multicast, IEFT draft, Holbrook-ssm-00.txt, March 2000.
[8] Y. Chu, S.G. Rao, S. Seshan, H. Zhang, A case for end system multicast, IEEE J. Select. Areas Commun. 20 (8) (2002).
[9] Y. Chu, S.G. Rao, H. Zhang, A case for end system multicast, Proc. ACM SIGMETRICS June (2000) 1–12.
[10] Y. Chu, S.G. Rao, S. Seshan, H. Zhang, Enabling conferencing applications on the Internet using an overlay multicast architecture, Proc. ACM SIGCOMM August (2001).
[11] J. Jannotti, D.K. Gifford, K.L. Johnson, Overcast: Reliable Multicasting with an Overlay Network, Proc. Oper. Syst. Des. Implement. (OSDI) October (2000) 197–212.
[12] J. Liebeherr, M. Nahas, W. Si, Application-layer multicast with Delaunay triangulations, IEEE J. Select. Areas Commun. 20 (8) (2002).
[13] P. Francis, Yoid: Your Own Internet Distribution, http://www.isi.edu/div7/yoid/, March 2001.
[14] D. Pendarakis, S. Shi, D. Verma, M. Waldvogel, ALMI: an Application Level Multicast Infrastructure, Proceedings of the Third Usenix Symposium on Internet Technologies and Systems (USITS), March 2001.
[15] S. Q. Zhang, B. Y. Zhao, A. D. Joseph, R. H. Jatz, J. D. Kubiatowicz, Bayeux: An architectuture for scaleable and fault-tolerant wide-area data dissemination, Proceedings of NOSSDAV, April 2001.
[16] M. Castro, P. Druschel, A.M. Kermarrec, A. Rowstron, SCRIBE: a large-scale and decentralized application-level multicast infrastructure, IEEE J. Select. Areas Commun. 20 (8) (2002) October.
[17] A. Rowstron, A. M. Kermarrec, M. Castro, P. Druschel, SCRIBE: the design of a large-scale event modification infrastructure, Proceedings of the Third International Workshop on Networked Group Communication (NGC), 2001.
[18] S. Ratnasamy, M. Handley, R. Karp, S. Shenkar, Application-level multicast using content addressable networks, Proceedings of the Third International Workshop on Networked Group Communication (NGC), 2001, pp. 14–29.
[19] Y. D. Chawathe, Scattercast: an architecture for Internet broadcast distribution as an infrastructure service, PhD thesis, Stanford University, September 2000.
[20] S. Banerjee, B. Bhattacharjee, C. Kommareddy, Scaleable application layer multicast, Proceedings of ACM SIGCOMM, August 2002.
[21] B. Zhang, S. Jamin, L. Zhang, Host multicast: a framework for delivering multicast to end users, Proceedings of INFOCOM, June 2002.
[22] D.A. Helder, S. Jamin, Banana tree protocol, an end-host multicast protocol, Technical Report, University of Michigan, CSE-TR-429-00, July 2000.
[23] S. Banerjee, C, Kommareddy, K. Kar, B. Bhattacharjee, S. Khulle, Construction of an efficient overlay multicast infrastructure for real-time applications, Proceedings of INFOCOM, April 2003.
[24] C.K. Yeo, B.S. Lee, M.H. Er, A framework for multicast video streaming over IP networks, J. Networks Comput. Applic. 26 (3) (2003) 273–289.
[25] C.K. Yeo, B.S. Lee, M.H. Er, An overlay for ubiquitous streaming over Internet, Proceedings of the Second International IFIP-TC6 Networking Conference, May 2002, pp. 1239–1244.
[26] C.K. Yeo, B.S. Lee, M.H. Er, A peering architecture for ubiquitous IP multicast streaming, ACM SIGOPS Oper. Syst. Rev. 36 (3) (2002) 82–95.
[27] S. Jain, R. Mahajan, D. Wetherall, G. Borriello, S.D. Gribble, A comparison of large-scale overlay management techniques, Technical Report UV-CSE 02-02-02, University of Washington, February 2002.
[28] L. Mathy, R. Canonico, D. Hutchison, An overlay tree building control protocol, Proceedings of the Third International Workshop on Networked Group Communication (NGC), 2001, pp. 76–87.
[29] A.J. Demers, D.H. Greene, C. Hauser, W. Irish, J. Larson, S. Shenker, H.E. Sturgis, D.C. Swinehart, D.B. Terry, Epidemic algorithms for replicated database maintenance, ACM Operating Systems Review 22 (1988) 8–32.
[30] Q. Sun, D.C. Sturman, A Gossip-based reliable multicast for large-scale high-throughput applications, Proceedings of the IEEE International Conference Dependable Systems and Networks, 2000, pp. 347–358.
[31] R. Tenesse, Y. Minsky, M. Hayden, A Gossip-style failure detection service, Proceedings of the IFIP International Conference on Distributed Systems Platforms and Open Distributed Processing (Middleware), September 1998, pp. 55–70.
[32] S. Floyd, V. Jacobson, C. Liu, S. McCanne, L. Zhang, A reliable multicast framework for light -weight session and application level framing, IEEE/ACM Trans. Network. 5 (6) (1997) 784–803.
[33] T. Speakman, D. Farinacci, S. Lin, A. Tweedly, Pragmatic General Multicast (PGM) reliable transport protocol, CISCO Systems, Internet Draft, 1998.
[34] K. Yano, S. McCanne, The Breadcrumb Forwarding Service: A Synthesis of PGM and EXPRESS to Improve and Simplify Global IP Multicast, ACM Comput. Commun. Rev. 30 (2) (2000).
[35] P. Francis, Yoid: extending the Internet multicast architecture, http://www.isi.edu/div7/yoid/, April 2000.
[36] B.Y. Zhao, J. Kubiatowicz, A. Joseph, Tapestry: an infrastructure for fault-tolerant wide-area location and routing, Technical report, UCB/CSD-01-1141, University of California, Berkeley, April 2001.
[37] S. Banerjee, B. Bhattacharjee, A comparative study of application layer multicast protocols, Unpublished report, Department of Computer Science, University of Maryland, http://www.cs.wisc.edu/~suman/pubs.html
[38] Z. Wang, J. Crowcroft, Bandwidth-delay based routing algorithms, IEEE Globecom November (1995).
[39] S. Ratnasamy, P. Francis, M. Handley, R. Karp, A scaleable content-addressable network, Proceedings of the ACM SIGCOMM, August 2001.
[40] M. Berg, M. Krevekd, M. Overmas, O. Schwarzkopf, Computational Geometry, Algorithms and Applications, Springer, Berlin, 1997.
[41] R. Sibson, Locally equiangular triangulations, Comput. J. 21 (3) (1977) 243–245.
[42] B.N. Karp, Geographical routing or wireless networks, PhD thesis, Harvard University, 2000.
[43] E. Kranakis, H. Singh, J. Urrutia, Compass routing on geometric networks, Proceedings of the 11th Canadian Conference on Computational Geometry, August 1999, pp. 51–54.
[44] M.J.B. Robshaw, MD2, MD4, MD5, SHA and other hash functions, Technical Report, TR-101 ver. 4.0, RSA Labs, 1995.
[45] C.G. Plaxton, R. Rajaraman, A.W. Richa, Accessing nearby copies of replicated objects in a distributed environment, Proceedings of the Ninth ACM Symposium on Parallel Algorithms and Architectures (SPAA), June 1997.
[46] A. Rowstron, P. Druschel, Pastry: scaleable, distributed object location and routing for large-scale peer-to-peer systems, Proceedings of the IFIP/ACM Middleware, November 2001.
[47] A. Ballardie, Core based trees (CBT) multicast routing architecture, Internet request for comments RFC 2201, September 1997.

[48] D. Estrin, D. Farinacci, A. Helmy, D. Thaler, S. Deering, M. Handley, V. Jacobson, C. Liu, P. Sharma, L. Wei, Protocol independent multicast—Sparse Mode (pim-sm): protocol specification, Internet request for comments RFC 2117, June 1997.

[49] D. Estrin, D. Farinacci, V. Jacobson, C. Liu, L. Wei, P. Sharma, A. Helmy, A protocol independent multicast—Dense Mode (pim-dm): protocol specification, Internet request for comments RFC 2117, June 1997.

[50] C.K. Yeo, B.S. Lee, M.H. Er, Application layer multicast architecture for media streaming, Proceedings of the Seventh IASTED International Conference on Internet, Multimedia, Systems and Architectures, Hawaii, USA, August 2003.