Efficient and Reliable Application Layer Multicast for Flash Dissemination

Kyungbaek Kim, *Member, IEEE*, Sharad Mehrotra, *Member, IEEE*, and Nalini Venkatasubramanian, *Member, IEEE*

Abstract—To disseminate messages from a single source to a large number of targeted receivers, a natural approach is the tree-based application layer multicast (ALM). In time-constrained *flash dissemination* scenarios, e.g. earthquake early warning, where time is of the essence, the reliable extensions of the tree-based ALM using ack-based failure recovery protocols cannot support reliable dissemination in the timeframe needed. In this paper, we propose FaReCast which exploits path diversity, i.e., exploit the use of multiple data paths, to achieve fast and reliable data dissemination. First, we design a forest-based M2M (Multiple parents-To-Multiple children) ALM structure where every node has multiple children and multiple parents. The intuition is to enable lower dissemination latency through multiple children, while enabling higher reliability through multiple parents. In order to maintain the M2M ALM structure in a scalable and reliable manner, we develop a DHT-based Distributed Configuration Manager. Second, we design multidirectional multicasting algorithms that effectively utilize the multiple data paths in the M2M ALM structure. A key aspect of our reliable dissemination mechanism is that nodes, in addition to communicating the data to children, also selectively disseminate the data to parents and siblings. As compared to trees using traditional multicasting algorithm, we observe an 80 percent improvement in reliability under 20 percent of failed nodes with no significant increase in latency for over 99 percent of the nodes. Moreover, we notice that FaReCast can reduce the network overhead more than 50 percent by tuning the M2M structure, as compared to the other reliable ALM based disseminations.

Index Terms—Overlay network, flash dissemination, fault resilience, multicast

1 Introduction

OUR work is motivated by the need for highly reliable data dissemination that delivers critical information to hundreds of thousands of receivers within a very short period of time. An example is the flash dissemination of disaster (natural or man-made) warning messages that must be rapidly delivered to urban populations in a matter of a few seconds to enable citizens to take self-protective measures (e.g., duck-cover-hold on for earthquakes, shelter underground for tornadoes). The key challenge lies in delivering messages scalably (reaching large populations), reliably (despite network outages and message losses) and efficiently (low operational cost during non-disaster times with quick ramp-up when needed). Given the rarity of these events, it is unlikely that a dedicated infrastructure, such as communication connections between peoples and the source of the messages, that is operational and available 24/7 will be deployed. Our objective, therefore, is to leverage any and all available infrastructure and exploit knowledge of network connectivity to ensure that interested recipients are able to actually receive the messages within a very short period of time.

In this paper, we develop an ALM-based solution for the flash dissemination problem above using a peer-oriented architecture where the peer nodes are formed by those interested in receiving the messages. We argue that the problem lends itself well to a peer-based architecture; an ALM-based solution is attractive due to its potential for easy deployment (no changes to lower layer network protocols at the participating nodes) [4] and its ability to deal with a variety of recipients. Challenges arise since 1) end-devices are autonomous and hence unreliable; 2) end-devices are nodes that are executing other tasks and must typically be rapidly repurposed to deal with the rare warning messages when it occurs. Our ALM-based solution must deliver the messages reliably and in time, while handling the node churn as well as minimizing the maintenance overhead.

We view the ALM approach as consisting of two key aspects—1) the ALM structure and 2) the multicasting algorithm deployed on the structure. Participating nodes in the ALM structure organize themselves into an overlay topology (typically a tree or mesh) for data delivery—each edge in this topology corresponds to a unicast path between two nodes in the underlying Internet. Once the ALM structure is constructed, data from the source node is delivered to all multicast recipients using the implemented multicasting algorithm. Note that all multicast-related functionality is implemented at the nodes (instead of at routers as in network-layer multicast).

Our target scenario consists of a single source and several receivers; A tree-based ALM structure (where every node has a single parent and multiple children) seems appropriate for fast data dissemination [4], [5], [18], [8]. A tree-based structure exploits concurrency in that all

- *K. Kim is with the Department of Electronics and Computer Engineering, Chonnam National University, South Korea. E-mail: kyungbaekkim@ jnu.ac.kr.*
- *S. Mehrotra and N. Venkatasubramanian are with Donald Bren School of Information and Computer Sciences, University of California, Irvine, CA 92697 USA. E-mail: {sharad, nalini}@ics.uci.edu.*

Manuscript received 27 Apr. 2012; revised 21 June 2013; accepted 21 Oct. 2013. Date of publication 5 Nov. 2013; date of current version 17 Sept. 2014. Recommended for acceptance by J. Cao.
For information on obtaining reprints of this article, please send e-mail to: reprints@ieee.org, and reference the Digital Object Identifier below.
Digital Object Identifier no. 10.1109/TPDS.2013.280

nodes that have received data can communicate with their children concurrently. As the *fan-out*, i.e. the number of children per node, of the tree increases, the data delivery time decreases logarithmically with the number of fan-out as the base. Unfortunately, the tree structure is highly vulnerable to failures since every intermediate node in a tree structure is a point of failure [16], [17], [25], [29] that can block delivery of the message to the entire subtree under it. In fact, this problem is further aggravated as the fan-out of a tree structure increases, since the number of nodes impacted by a failure increases exponentially.

Typical reliable multicast approaches detect such failures during operation and recover from them (usually by reconstructing parts of the tree). Protocols have been designed to recover the tree structure by substituting the failed node with an existing online node [4], [5], [6], [7], [22], [30], [29]—this requires tree reconstruction which is a time-consuming operation; techniques have also been proposed to use backup information to reduce recovery time to some extent [18], [23], [24], [25], [26], [27], [28]. Many of these approaches rely on TCP retransmission to detect and retransmit lost packets and support reliability at the communication layer. In our flash dissemination scenario, i.e., dissemination of disaster warning messages, delivery time are highly constrained; existing approaches to detect and recover from failures or employ per-packet acknowledgements require interactions and handshakes with other nodes, which makes it difficult to respond within the required time constraints.

In this paper, we present FaReCast, an ALM protocol for fast and reliable data dissemination that exploits the use of multiple data paths between nodes judiciously by 1) designing a new ALM structure and 2) developing a new multicasting algorithm that efficiently exploits the proposed structure. Specifically, we design a forest-based M2M (Multiple parents-to-Multiple children) ALM structure where each participant node has multiple parents as well as multiple children. Care is taken in the construction of the M2M structure to support path diversity (through the choice of unique parent node sets); this supports increased reliability by minimizing the probability that nodes with failed common parents lose data. To complement the M2M ALM structure, we design a multidirectional multicasting algorithm that effectively utilizes the multiple data paths in the M2M ALM structure. In addition to top-down communication characteristic of traditional multicasting, the multidirectional multicasting algorithm deploys bottom-up data flow and horizontal data flow carefully. Since there is prior knowledge of the M2M structure at each node, nodes can trigger communication and send data to parents from which it has failed to receive the expected data. Additionally, in the case of leaf nodes, FaReCast forwards data to the other leaf nodes which it anticipates may have not received the data. A key design issue for FaReCast is addressing the tradeoff between two conflicting factors—higher fan-out (increased speed) vs. higher fan-in (increased reliability).

While our primary goals are scalability, speed and reliability, our secondary goal is to reduce maintenance overhead during normal times when there is no event. In FaReCast, we minimize the client-side maintenance overhead by storing the snapshots of current network status at the configuration manager. Each individual user retrieves accurate parent/children information from the configuration manager periodically. The configuration manager detects node failure based on this periodic update request and updates the snapshot asynchronously. Through both simulation and implementation-based evaluations, we show that FaReCast tolerates over 40 percent random failures while meeting the latency constraints, i.e. FaReCast can endure high user churn as well as a certain level of snapshot inconsistency.

The rest of the paper is organized as follows. The related works are presented in Section 2. In Section 3, we describe the design and management of the M2M ALM structure having multiple fan-in and fan-out. Section 4 discusses the distributed configuration manager in order to enhance the scalability and the reliability of the management of M2M structure. The multidirectional multicasting algorithm achieving high reliability with small data latency is described in Section 5. In Section 6, we analyze the reliability and the overhead of FaReCast with a simple model. We evaluate the proposed FaReCast system in Section 7 with extensive simulations and implementation on a campus cluster platform and an emulated wide area network. Finally, we present concluding remarks in Section 8.

2 Related Works

Typical ALM applications include file sharing and content streaming; specialized protocols have been designed for these applications [5], [6], [7], [9], [10], [11], [12]. Our prior work on flash dissemination focused on fast dissemination of medium to large sized data. We leveraged a dynamic mesh-based overlay network constructed using a random walker protocol, CREW [13] to concurrently disseminate multiple chunks of a large message. Also, to enable low dissemination overhead despite catastrophic failure, we extended the random walker implementation (the Roulette protocol); it was used to implement a P2P webserver Flashback [14] to deal with flash crowds. However, the cost of metadata propagation and overlay maintenance in CREW/Roulette is unwarranted in the current scenario where 1) the end recipients are known and 2) the message size is small.

Multiple fan-in ALM structures are considered in other approaches such as [9], [11], [15], [16], [17], [20], [21]. All increase the number of data paths to a target node to stream large data concurrently and efficiently. Here, the large data is divided into smaller data chunks that are disseminated through the multiple paths concurrently. The chunking techniques are combined with loss tolerant decoding/encoding [9], e.g., erasure coding [19] to ensure that all recipient nodes get the entire large data correctly. Such expensive coding schemes are unnecessary in our scenario where the information delivered to target nodes is small (order of a few bytes/Kbytes). FaReCast exploits path redundancy to send the same data through multiple paths, and the duplicated messages are used to initiate the proposed multidirectional multicasting which utilizes the multiple data path more effectively.

A possible extension to meet the time constraint is randomized forwarding proposed in PRM (Probabilistic Resilient Multicast) [28]. By using small amount of redundant messages, PRM improves the reliability of dissemination on a tree structure. However, in the case of the flash dissemination,

its enhancement can be limited under very high failures such as 20-30 percent. In FaReCast, the combination of M2M structure and the multidirectional multicasting supports 100 percent reliability while meeting the latency constraint under very high failures such as 30-40 percent.

3 Forest-Based M2M ALM Structure

In this section, we present our forest-based M2M ALM structure. The system primarily consists of 1) target nodes interested in receiving the dissemination, 2) an originating server where the dissemination is initiated, and 3) a configuration manager that maintains and manages the structure and connectivity information (discussed later). We will construct an overlay structure consisting of the originating server and target nodes and effectively use that structure to address our dual needs of reliability and timeliness in information dissemination. To maintain separation of concerns, we distinguish two clear-cut steps in enabling reliable, fast ALM: 1) construction of an overlay structure that enables concurrency and reliability 2) use of the constructed overlay in a multicast protocol that efficiently implements concurrency (aka speed) and reliability tradeoffs in the dissemination process under dynamic conditions.

The process begins with the construction of the forest-based M2M overlay structure which consists of the originating server (or a well-established representative) as the root-node and target nodes (interested recipients) that form the sub-tree rooted at the originating server. The goal is to organize the target nodes into levels and establish overlay connections between them to enable concurrency and reliability for dissemination. The overall design philosophy is as follows: to support fast dissemination, every node in the structure will have multiple children from which concurrent content dissemination can occur. To handle reliability, every node in the structure will have multiple (redundant) parents that it receives content from. Determining the arities and connections of nodes in the M2M (Multiple parents to Multiple children) overlay structure to maximize speed and reliability while minimizing redundant transmissions is the key challenge. Prior to establishing the properties of the M2M structure, we provide some definitions and assumptions that will be used in creating the M2M structure.

Level of a node (L): The level of a node is the length of the path, expressed as number of hops, from the root to this node. The level of the root node is assumed to be 0, children of the root node are at level 1 etc.

Sibling Nodes: Nodes having same level are referred to as sibling nodes. Sibling nodes at a level belong to a *level-group*. N_L is the number of sibling nodes at level L.

Fan-in (F_i): The fan-in of a node is the number of parents of a node. All participating nodes (except those at levels 0 and 1) have F_i parents.

Fan-out factor (F_o): The fan-out factor is the ratio of N_{L+1} to N_L, and is a measure of the minimum number of nodes at a level, i.e., $N_L \geq {F_o}^L$.

The **configuration manager** is a dedicated component managed by organizations such as local authorities which are interested in reliable overlay network as a fast message dissemination medium. It enables the management of the M2M structure by performing the following tasks: 1) management of information on nodes, level-groups and sibling nodes in a level-group; 2) construction and maintenance of the overlay structure, 3) answering of queries about the M2M structure—for example, it responds to update requests from target nodes with information on their current parents/children. The update requests also serve as periodic heartbeat messages to the configuration manager indicating that a node is alive; in FaReCast, we keep the frequency of such update requests very low ($\sim$1 hour). Communications between the configuration manager, target nodes and the originating server occur via a pre-established API.

3.1 Properties of the M2M ALM Structure

- **Root-node reliability**: The root node (at level 0) is expected to be continuously available.
- **Fan-in constraint**: Each participating node should have F_i distinct parents just one level above. If the level of a node is L, all the parents are picked from the same level, $L-1$. The only exceptions are nodes at level 1 that have only one parent since the level-group for level 0 has only one node, the root node.
- **Loop-free nature**: We assume loop-free transmission when data flows from parent nodes to child nodes. In particular, we assume that the assignment of levels in the M2M ALM structure is done in such a way that all paths from the root to a node have the same length. Assuming that the latency of message transmission over any two links is not significantly different [3], the M2M ALM structure can guarantee that data reaching a node from different parents arrive close to each other in time. That is, in the M2M ALM structure with F_i parents, each node should get F_i data items within a short period of time under the failure-free situation. This property is later exploited (and selectively relaxed) by the multidirectional multicasting protocol to improve reliability and make up for missing messages from failed parents.
- **Parent set uniqueness**: All nodes in the M2M ALM structure (with $level \geq 2$) have a unique set of parents. That is, there is at least one different parent in the parent-sets of two nodes at the same level. Without this property, concurrent failures of nodes at a level can potentially block out messages to the level below; parent-set uniqueness enforces path diversity in the propagation of the message and consequently improves the chances of reliable delivery.

Achieving the parent-set uniqueness property brings out an interesting relationship between fan-in and fan-out factor. The following equation should be satisfied to guarantee parent-set uniqueness:

$$N_{L+1} \leq C(N_L, F_i) \quad \text{where } F_i > 1, L > 0. \tag{1}$$

And, to satisfy Equation (1), N_L is determined by the following equation:

$$N_L = {(F_o)}^L + F_o + F_i - 2 \quad \text{where } F_o > 1, F_i > 1, L > 0. \tag{2}$$

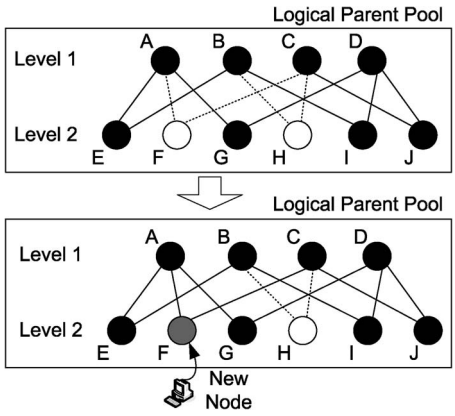

Fig. 1. Predefined logical parent pool and the uniqueness checking operation for a new node. $F_i = 2$ and $F_o = 2$.

The detail description of achieving parent uniqueness is found in Section 1.1 of the supplement file which is available in the Computer Society Digital Library at http://doi.ieeecomputersociety.org/10.1109/280.

3.2 Node Operations and Structure Maintenance

Given an initial root node (possibly the originating server), the creation of an M2M ALM structure that is compliant with the properties described above proceeds as a series of node joins. Subsequent evolution of the structure consists of a series of node join/leave requests coordinated by the configuration manager. Interspersed with node joins and node leaves, the configuration manager detects node failures (i.e. an unstructured node leave) and reorganizes the M2M ALM structure to recover from the failure. FaReCast employs periodic *update request* messages from participant nodes to the configuration manager to communicate that a participant node is still alive. The configuration manager responds to a node's *update request* with the current parents/children of the node. As discussed earlier, the frequency of *update messages* is kept low to prevent bottlenecks at the configuration manager and reduce structure maintenance cost. The details of each operation are described below.

3.2.1 Node Join

When a new node, N_{new}, joins the system, it is first given a new nodeID. The node sends a join request to the configuration manager in order to determine 1) its level and 2) its parent/children nodes. In accordance with Equation (2), the configuration manager determines current information about each level group and selects the lowest level requiring more nodes as the *level* of the new node in the M2M ALM structure. Parent selection for the node proceeds as follows. If the selected level is $L+1$, the configuration manager randomly picks F_i different nodes among the level L nodes as the parents of the new node.

The configuration manager next determines whether the selected parent set satisfies the *parent set uniqueness property* at level L. The configuration manager maintains a *parents pool* structure managed by each level-group—that contains entries of the form {child, parent-set}. If the newly selected parent-set is not unique, we choose to replace one of the nodes (the one with the maximum number of children) in the current parent set with another distinct randomly selected node at the same level (excluding the most recent selection). This uniqueness checking operation is repeated until a unique set of parents is found. If the configuration manager follows the constraint of Equation (2), every new node can find a unique set of parents.

The uniqueness checking operation can be projected to a collision problem with the birthday paradox problem. At level L, there are given N_{L+1} random parent sets drawn from a discrete uniform distribution within the set $C(N_L, F_i)$ and we can consider the probability $p(N_{L+1}; C(N_L, F_i))$ that at least two parent sets are the same. Through Taylor series expansion of the exponential function, we can approximately obtain the following equation: $p(N_{L+1}; C(N_L, F_i)) = 1 - e^{-N_{L+1}(N_{L+1}-1)/2C(N_L,F_i)}$. According to the equation, when the level L is small such as 2 or 3 $p(N_{L+1}; C(N_L, F_i))$ is higher than 0.91, but $p(N_{L+1}; C(N_L, F_i))$ decreases exponentially as the level L increases. For example, then the level L is 10, the probability of collision drops below 0.0014. That is, when the level L of a new node is big enough, the uniqueness checking operation does not incur high time cost.

However, when the level L of a node is small, the probability of collision becomes very high and the worst case time complexity such $O(N_{L+1})$ can be expected with high probability. Here, $O(1)$ means the single uniqueness checking operation including the selection of a random parent set and the test of collision. According to this, an alternative way of the uniqueness checking operation for the new nodes at lower level is required. The alternative way is using a *predefined logical parent pool* which is generated by the configuration manager in a deterministic way which follows Equation (2). The predefined logical parent pool describes the parent-child relationship between logical nodes in advance. Whenever a new node joins the system, the node is mapped into a logical node in the predefined logical parent pool and obtains the information of its parents and children. Fig. 1 depicts the predefined logical parent pool and illustrates the alternative way of the uniqueness checking operation for a new node. The usage of the predefined logical parent pool allows $O(1)$ complexity of the uniqueness checking operation.

After finding a unique set of parents, the configuration manager registers the new node with the new nodeID and its selected parent-set in the corresponding parent pool, and each parent of the new parent-set registers the new node as a child using its nodeID. The configuration manager then responds to the join request with the nodeID, the level, and parents/children information.

3.2.2 Node Leave

When a node leaves the M2M ALM structure, it notifies the configuration manager to coalesce the hole caused by it in the ALM structure. All the impacted nodes—parents-sets that include the leaving node, children of the leaving node and parents of the leaving node should be invalidated and updated. Also the number of nodes in the level-group of the leaving node reduces by one. The key idea is to quickly find a replacement node for the leaving node with minimum maintenance overhead including communication and processing cost. To do this, we invalidate the registered node information of the leaving node. The invalidated node information is retained at the configuration manager for a while (this is captured by a *retainment-timer*) with the

expectation that a new node will join quickly and it can simply replace the invalidated node. If this happens, the configuration manager validates all the impacted nodes to accommodate the newly arrived node.

If there are not enough new nodes to cover the invalidated nodes, the configuration manager must repair these holes with other existing nodes. A retainment-timer is associated with each invalidated node; when the retainment-timer of an invalidated node expires without any replacement, the configuration manager picks a random leaf node to replace the invalidated node and subsequently invalidates the leaf node information. The configuration manager does not notify the updated information to all the impacted nodes right away, but responds with it to their periodic requests for the recent structure information (update message).

3.2.3 Node Failures

Failed nodes are those that leave the M2M ALM structure without any notification. The configuration manager detects the failed nodes by an *update-timer*. Each participant node sends a periodic update request in order to refresh the parents/children information. Once a node joins the system and its information is registered, the configuration manager sets the update-timer of the node and resets the timer whenever it receives an update request from the node. If the update-timer of a node is expired, the node information is invalidated, and the repair of the M2M ALM structure proceeds similar to the case of a leaving node. The nodes in the lower level-groups use shorter update-timers than the nodes in the higher level-groups, because lower level nodes are critical to preserving the reliability of the M2M structure.

3.2.4 Maintenance Overhead and Reliability

In our target scenario, an interesting event occurs rarely and participating nodes join/leave the system far more frequently than the event. It is therefore imperative that the management overhead (e.g., network bandwidth, processing power) incurred by the M2M ALM structure is low, especially during non-event times. In order to reduce the maintenance overheads of the participating nodes, we implement a cooperative solution in which the configuration manager and nodes participate. Firstly, we effectively and efficiently use the configuration manager to maintain the M2M ALM structure and required meta-data; participating nodes send update requests to the configuration manager for current structure information. At the node end, we increase the length of timers (retainment-timer and update-timer) to keep neighborhood information updated. There is an obvious tradeoff between the overhead and reliability since delaying timers implies that the M2M ALM structure from the view of participant nodes is not as updated as desired. In Section 5, we propose an effective multidirectional multicasting protocol (when the event occurs) that can tolerate stale information in the M2M ALM structure—allowing us to achieve low maintenance overhead with high reliability.

4 Distributed Configuration Manager

The configuration manager plays an important role in the construction of the M2M ALM structure by managing the information about FaReCast nodes and communicating with them. While a centralized configuration manger is simple to implement, it presents significant scalability and reliability issues, especially when the number of users scale. This is true even if the frequency of update requests is kept very low; handling a large number of user requests in a centralized manner can incur substantial overhead. Moreover, a centralized manager is a single point of failure in generating and maintaining the M2M structure. To alleviate scalability and reliability concerns that arise from a centralized configuration manager, we design a DHT-based distributed configuration manager. The distributed configuration manager is composed of multiple machines provided by the related organization such as local authorities. DHT is used as a substrate for the distributed configuration manager primarily because it provides self-organizing properties making distributed manager more reliable. It also inherently disperses maintenance requests (join/update) across the nodes and partitions the associated information storage load across the multiple machines.

Every machine in the distributed configuration manager is associated with a machineID obtained by using the SHA-1 hash of the concatenated string of its IP address and port number. Each end user also has a nodeID generated using the similar hashing technique with any unique string of an end user (IP address or MAC address). Given machineIDs and nodeIDs, each machine is responsible for the information of nodes whose nodeIDs are numerically close to its machineID [31], [32]. That is, when a node joins, leaves or requests an update message, the request is forwarded to the machine corresponding to its nodeID. Each machine store the information of the responsible nodes and replicate it in order to provide tolerance to unexpected failures of some machines in the distributed configuration manager. The other issues related to the distributed storage such as replication and load balancing may be considered, but these issues can be resolved by using the techniques proposed in previous researches [31], [32]. The information associated with a node includes the nodeID, level, invalidation status, parents, children, leaf-links, IP address, port, and timers (update-timer/retainment-timer). Since the machinedIDs and nodeIDs are distributed uniformly on the ID space, the overhead for storing the information of nodes can be distributed to multiple machines uniformly. That is, if the number of nodes is N and the number of machines in the distributed configuration manager is M, the storage overhead is $O(N/M)$.

When a new node joins the system, it sends a join request to any one of the machines in the distributed configuration manager. The join request is forwarded to the machine corresponding to the nodeID of the new node. If there is any invalidated node in the local machine, it accommodates the new node with the invalidated instance. Otherwise, it follows the general join process; selecting its level and parents.

A machine in the distributed configuration manager cannot get the exact global status of each level of ALM structure without an expensive synchronization service, but can estimate the global status by using the local status

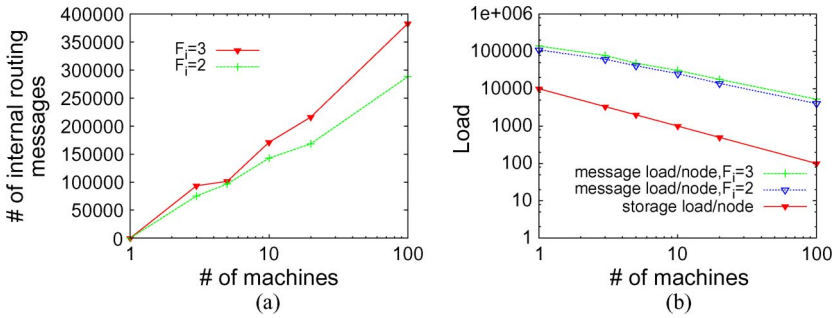

Fig. 2. Performance for the DHT-based distributed configuration manager. FreePastry Setting: Size of LeafSet $= 4$, BaseBitLength $= 2$. (a) Internal messages. (b) Average overhead of a machine.

of each level and the number of participating machines of distributed configuration manager (M). Since each machine manages around $1/M$ of the entire set of nodes, each machine also manages a potion of size $1/M$ of the nodes at each level. If the number of nodes managed by a machine for the level L is N_L^{local}, the machine can estimate the global number of nodes for the level L by multiplying N_L^{local} and M. If $MN_L^{local} < N_L$, where N_L is from Equation (2), the machine assigns the level L to the new node. Otherwise if $MN_L^{local} >= N_L$, the machine assigns the level $L+1$ to the new node and repeats this until the level $L+1$ is fully assigned.

After a new node obtains its level, it selects candidates of its parents by sending request to other machines, uniformly at random. Rather than using the centralized parent pool, the parent set uniqueness is tested by checking the existence of any shared child among the candidates of parents. If there is no shared child among the candidates of parents, it guarantees that the candidate set is a unique parent set.

The information of invalid nodes is managed by each local machine with retainment-timers. If a retainment timer is expired before accommodating any new node, the corresponding invalid node should be replaced with another valid node in the local machine.

Since all of the components for handling requests are distributed, the distributed configuration manager solves bottlenecks that occur with the centralized configuration manager when the number of nodes is large. However, it takes additional internal messages to handover a request to other machines. Let us assume that there are M machines and each machine manages $1/M$ of the entire nodes. If we assume that the target nodeID of a join/leave requet is uniformly distributed on the ID space, a machine may forward $(1-1/M)$ of requests to other machines. Generally, the routing overhead of a DHT-based system increases logarithmically with the size of participant entities, and it can be represented as $O(\log M)$. That is, the additional internal messages in the distributed configuration manager is $O((1-1/M)\log M)$. However, the additional internal messages are also distributed in M machines. That is, the cost of additional internal messages for forwarding requests of other machines is $O(((1-1/M)\log M)/M)$.

To evaluate the scalability of the proposed distributed configuration manager, we implemented a DHT (FreePastry [2])-based distributed configuration manager and ran it on ModelNet [1] setting which is used for evaluating the implementation of FaReCast. We varied the number of machines from 1 to 100. The case of only one machine represents the centralized configuration manager. In the default settings for FaReCast, we used $F_o = 3$, and $F_i = 2$ or 3. We assume that there are 10 K end users and they join the system arbitrarily. With this setting, around 10 minutes is used for every node to join the system and around 60 ms is used for handling a single message.

Fig. 2 shows the performance of the distributed configuration manager. The centralized configuration manager does not generate internal messages, but incurs the complete overheads of data storage and user request handling. As shown in Fig. 2a, the DHT based distributed configuration manager generates the internal messages which increases logarithmically with the number of machines. This is because a distributed configuration manager with more machines requires more forwarding steps to locate a machine which can handle a random request. Also, we noticed that F_i impacts the amount of generated internal messages. That is, as F_i increases, each machine initiates more requests in order to obtain parents. As the number of machines increases, the impact of F_i to the internal messages becomes substantial.

However, as shown in Fig. 2b, the overall overhead of a node decreases inversely proportional to the number of machines both in terms of the message overhead (number of messages) and storage overhead (number of instances of node information). As expected, we observed that the storage overhead decreases with the number of machines. Since the size of an instance of a node information is up to 4 KB, if there are 100 machines it spends only 400 KB rather than 40 MB which is for a centralized configuration manager. In the case of the message overhead, we measured every message between the configuration manager and end users—this includes Join request/response messages and Update request/response messages for new parents and children. We also considered the additional internal messages such as messages used for uniqueness checking operations. Despite the fact that multiple machines incur additional internal messages, we observed that the individual overhead of a machine decreases significantly along with the number of machines. According to this, we expect that the more machines are used, the more users can be handled by the distributed configuration manager gracefully.

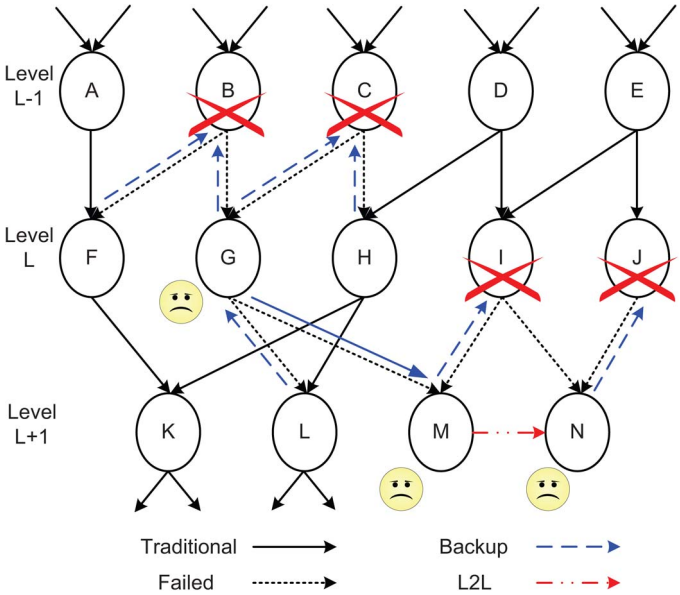

Fig. 3. Example mutidirectional multicast.

5 Multidirectional Multicasting

In this section, we describe our proposed multidirectional multicasting algorithms. The key idea here is to enable reliable message delivery despite failures with limited increase in latency and messaging overhead. We do this by pushing the multicast message along directions where failures are estimated to have occurred. Recall that when a node in the M2M ALM structure receives a message (the first one) from one parent, it is expected that messages from all the other parents will arrive shortly (due to equal path length and similar delay assumption). When messages do not arrive from parents, we conservatively assume possible failures in the direction of those parents—and enhance the multicasting algorithm to send additional messages in these directions. We propose a protocol that encompasses two forms of multidirectional multicast—*Backup* and *Leaf-to-Leaf (L2L) dissemination*.

5.1 Backup Dissemination

In our M2M ALM structure, although a node has multiple parents, the message remains undelivered to the node if all parents of a node have failed. Note that if this node is an intermediate node, its children can still receive the message from the other parents and paths. In other words, the message may bypass this node even though the node is operational—we refer to such nodes as *bypassed nodes*. Once a node is bypassed, its children lose one data path and the probability of message loss increases. For example, in Fig. 3, node G is a bypassed node and its child, node M does not get the message. One objective of our multicasting protocol design is to reduce the degradation of the reliability caused by the bypassed nodes.

Traditional multicasting protocols propagate messages unidirectionally along the tree from parents to children and cannot address the bypassing problem. If however, bottom-up propagation is permitted, i.e., messages from children to parents, the bypassed nodes can receive a message from one of its children who have received the message from alternate parents. Since this creates possibly redundant messages—careful determination of potential bypassed parent nodes by children is essential.

The detection of bypassed nodes exploits the constant-path-length property maintained by the M2M ALM structure. According to this property, when a node receives the first message from one of its parents, it expects that other $F_i - 1$ parents will send the same message within a short period of time, T_w. When a node fails to receive a message within T_w from a parent, the node assumes that the parent node has been bypassed or has failed—and forwards the message to this parent—we refer to this selective bottom-up dissemination as *backup dissemination*.

5.2 Leaf-To-Leaf Dissemination

While backup dissemination handles missing messages to bypassed intermediate nodes, it cannot be applied to a leaf node whose parents have all failed—we refer to such a leaf node as a *missing leaf node*. Given the M2M ALM structure (larger number of nodes at higher levels of the tree), a significant number (almost half) of the participating nodes are leaf nodes. Techniques to ensure that messages get to leaf nodes on time are critical to the reliable dissemination process.

We introduce the notion of *leaf-links* to address the problem of missing leaf-nodes. Here, each leaf node maintains a *leaf link pool*, i.e., a set of links to leaf nodes sharing the same parents, i.e., *leaf-links*. When a leaf node gets the first message from a parent, the following actions are performed. First, leaf links sharing that parent are masked in the leaf link pool since it is assumed that the other leaf nodes can also receive the message from the shared parent. Second, as in backup dissemination, the node waits for a stipulated period of time T_w, to receive messages from all the other parents. After the waiting time passes, the leaf node sends the message to the other leaf nodes corresponding to the unmasked leaf links. In Fig. 3, node M getting the first message from node G recognizes that node I is bypassed or failed, and sends the message to node N which shares node I as a parent. The direction of this dissemination is horizontal (leaf node to leaf node) and we refer to it as *L2L (leaf-to-leaf) dissemination*.

Unlike backup dissemination, we argue that having different waiting times for starting the L2L dissemination will help reduce redundant messages. Since every path from the root to leaf nodes at the same level has the same hop-length, transitively every message originated at the root reaches all sibling nodes within the same bounded time period T_w, assuming no failure. If all sibling leaf-nodes use the same duration of time to wait for the other messages, they start the L2L dissemination at the similar time, resulting in huge redundant messages.

To alleviate these redundant L2L disseminations, we employ differential waiting periods at leaf-nodes, similar to exponential backoff strategies. Leaf nodes that wait for longer periods now receive messages from recovered bypassed parents (due to backup dissemination) and from other leaf nodes (due to L2L dissemination). This causes related leaf links to be masked and redundant messages are avoided. To achieve random waiting time,

each leaf node generates a random integer n in the range $[1, N]$ and the waiting time is set to $n * T_W$, where T_w is the waiting time for the backup dissemination. As N increases, the variance of random waiting time increases and we more redundant messages are avoided. We set N as 4 in the evaluation.

6 Analysis of Reliability and Overhead

We analyze FaReCast with a simple model in two aspects—reliability and overhead under a flash dissemination scenario. We assume that each node has a unique set of parents and parents of a level L node are located at level $L-1$. We denote fan-in and fan-out factor as F_i and F_o, respectively. For simplicity, we assume that the probability of failure of every node is same as P_f.

At first, we consider the case of a traditional top-down multicasting. In this case, a node will not receive a message if all of its parents fail. That is, the probability that a node will not get a message can be represented as $P_{bypass} = (P_f)^{F_i}$.

However, since there is no time for repairing the failures of M2M structure in the flash dissemination scenario, P^L_{bypass} of a node at level L should consider not only the probability of failures of its parents at level $L-1$, but also the probability that parents will not get a message. Accordingly P^L_{bypass} of FaReCast only with a traditional top-down multicasting can be represented as Equation (3). Equation (3) also represents the probability that a node will not get a message in the case of multiple tree or mesh structure with the traditional multicasting. In this case, the overhead message of a level L node can be represented as $O^L = F_i - 1$

$$P^L_{bypass} = (P_f)^{F_i} + \sum_{i=1}^{F_i} \binom{F_i}{i} (P_f)^{F_i - i} (1 - P_f)^i \left(P^{L-1}_{bypass}\right)^i. \quad (3)$$

According to Equation (3), we note that P^L_{bypass} increases as L increases. It is the main reason that most of nodes which fail to receive a message in FaReCast (also in multiple tree structures) only with the traditional top-down multicasting belong to leaf nodes. The property that P^L_{bypass} increases along with L also affects the reliability of random mesh structure using epidemic dissemination. The diameter of the random mesh structure increases logarithmically with the number of participant nodes, and it is relatively longer than that of FaReCast or a multiple tree structure. Because of the long diameter, random mesh requires very high fan-in (F_i) to achieve high reliability (100 percent).

The probability that a node will not get a message, P^L_{bypass}, impacts the reliability of the multicasting system. In order to achieve high reliability, P^L_{bypass} should be low. The simplest way is increasing F_i and it is the only way for the traditional top-down multicasting in any structures. However, in FaReCast using the proposed multidirectional multicasting, the intermediate nodes using the backup dissemination and the leaf nodes using the leaf-to-leaf dissemination exhibit different P^L_{bypass} and O^L.

When using the backup dissemination, a level L node can expect additional messages from its children at level $L+1$. That is, P^L_{bypass} of an intermediate node using the backup dissemination should consider the case that all children miss a message, and it is represented as the following equation with some level of approximation. In this case, the additional messages from children are produced only if the children detect bypassed parents. Accordingly, the expected overhead of an intermediate node is $O^L = (F_i - 1) + P^L_{bypass}(1 - P^{L+1}_{bypass})F_o$

$$P^L_{bypass} = (P_f)^{F_i + F_o} + \sum_{i=1}^{F_i + F_o} \binom{F_i + F_o}{i} (P_f)^{F_i + F_o - i} (1 - P_f)^i \left(P^{L-1}_{bypass}\right)^i. \quad (4)$$

When using the leaf-to-leaf dissemination, a level L leaf node may receive messages from its sibling nodes which sharing the same parents. In our M2M ALM structure, the number of nodes in level L is around $(F_o)(L)$, and each node chooses F_i parent nodes among the $(F_o)(L-1)$ nodes in level $L-1$. According to this, an intermediate node at level $L-1$ has averagely F_oF_i children. That is a leaf node has $F_oF_i - 1$ leaf-links for each parent. For simplicity we assume that leaf-links are distinct to each other. In this case, P^L_{bypass} of a leaf node using the leaf-to-leaf dissemination should consider the case that all sibling miss a message, and it is represented as the following equation, where $F_l = (F_iF_o - 1)F_i$:

$$P^L_{bypass} = (P_f)^{F_i + F_l} + \sum_{i=1}^{F_i + F_l} \binom{F_i + F_l}{i} (P_f)^{F_i + F_l - i} (1 - P_f)^i \left(P^{L-1}_{bypass}\right)^i. \quad (5)$$

As same to the case of the backup dissemination, the leaf-to-leaf dissemination is triggered only if a child node detects bypassed parents. So, the expected overhead of a leaf node using the leaf-to-leaf dissemination is represented as $O^L = (F_i - 1) + P^L_{bypass}(1 - P^L_{bypass})F_oF_i - 1F_i$.

Fig. 4 shows the theoretical comparison of P^{15}_{bypss} along with different failure rate. In Fig. 4a, we observed that backup dissemination significantly improve the reliability and FaReCast achieves 100$ reliability. Moreover, in Fig. 4b, it is observed that as one of both F_i and F_o increases the reliability increases.

We note that when more failures are expected, more additional messages are generated to compensate the loss of the reliability in FaReCast. That is, FaReCast keeps high reliability by using additional messages incurred adaptively to the degree of failures. Moreover, unlike the general thought which is that the big F_o hampers high reliability under high probability of failures, we observed that the big F_o helps keep high reliability in FaReCast since the conditionally generated additional messages depends on F_i as well as F_o.

7 Evaluation and Implementation

To gain a better understanding of how FaReCast works under massive failures, we evaluated FaReCast along multiple dimensions, primarily in comparison to tree-based multicast, mesh-based multicast and flooding based (epidemic) protocols. These dimensions include: 1) Reliability (the ratio

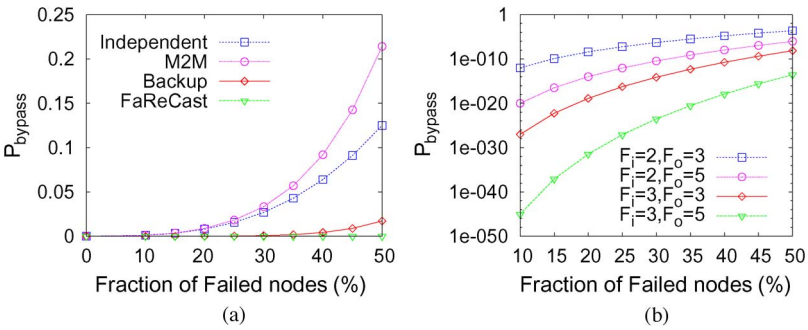

Fig. 4. Theoretical comparison of P_{bypass} at level 15. (a) Various methods. (b) Various F_i and F_o.

of unfailed nodes that finally receive the message); 2) Latency (the delay of message delivery to unfailed nodes that are able to receive the message); and 3) Efficiency/Overhead (number of duplicated/redundant messages and its impact on system overhead). The detail simulation settings are found in Section 1.2 of the supplement file available online.

7.1 Effectiveness of FaReCast under Failure

In Fig. 5, we compare FaReCast with other protocols, specifically, Tree, forest/mesh based dissemination such as SplitStream [9], MultipleTree [20], [21] and PRM [28], and flooding-based approaches such as Epidemic [11], [13], to disseminate an urgent message. Fig. 5a compares the reliability of the various protocols as a function of the fraction of failed nodes. To compare these protocols in a fair manner, we set the number of fan-in and fan-out as 3 and 3 for all protocols, respectively, except Tree which has only one fan-in. We set the number of neighbors in Epidemic to 6, since neighbors can be parents or children. For the PRM we set the number of neighbors to 20. Reliability is measured as the number of nodes receiving the message over the number of online nodes. As expected, the Tree protocol is affected adversely by the failure. Protocols having multiple parents can tolerate up to 15 percent failures reasonably well. When the number of failures exceed 15 percent, FaReCast continues to achieve 100 percent reliability, while others including M2M (uses only the M2M structure which does not employ the multidirectional multicasting) lose substantial reliability.

Fig. 5b plots the maximum latency as a function of the fraction of failed nodes. The Tree, Epidemic, PRM and M2M protocols appear to have lower maximum latencies—a closer look reveals that this comes at the cost of reliability (many nodes do not receive the message at all). SplitStream and MultipleTree lose reliability and exhibit increased latencies as the number of failures increase. The key reason is that the distance between the root and a node increases under failure in SplitStream and MultipleTree. M2M has a slightly reduced latency since its loop-free nature holds under failures. We observe that the maximum latency of FaReCast also increases along with failures. A deeper analysis indicates that this increase is caused by the waiting time to trigger the multidirectional multicasting. However, we observe that FaReCast delivers the message to around 99 percent of the total nodes with small latencies; higher latencies are incurred by a few bypassed nodes reached by backup/L2L dissemination. Epidemic exhibits good average latency behavior since the number of nodes reached in each dissemination step (in the early stage) is twice that of other protocols. We also observe that the average latency behavior of PRM is comparable to the Epidemic since PRM employs hybrid approach of tree and mesh structure.

Fig. 5c shows the message overhead as a function of the fraction of failed nodes. We define the message overhead as the average number of messages sent by an *online* node. In Tree, the message overhead decreases because most of nodes cannot get the message under failures. M2M, SplitStream, and MultipleTree have almost same overhead (around 3); Epidemic sends twice as many messages (around 6). The overhead of PRM is around 5 which is similar to Epidemic. Note that many of these protocols lose substantial reliability in spite of the high message overhead. In FaReCast, the message overhead increases along with failures. This is the unique feature of FaReCast, while other protocols usually use the same amount of message

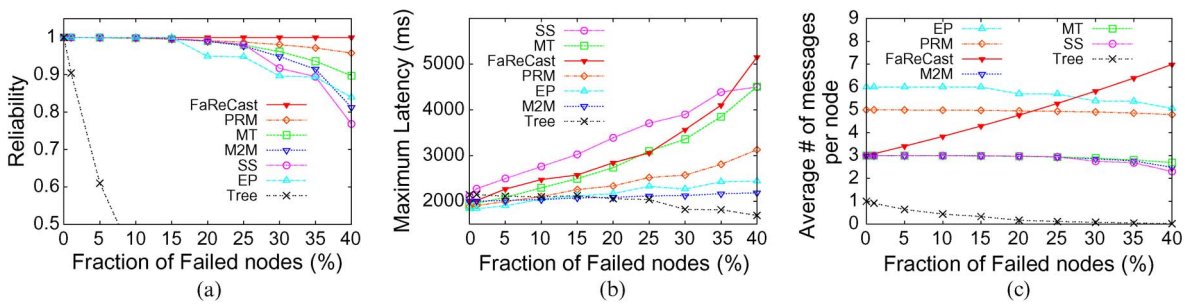

Fig. 5. Performance comparison of FaReCasts ($F_i = 3$, $F_o = 3$) with other protocols under various failures with 100,000 nodes. (SS:SplitStream, MT:MultipleTree, EP:Epidemic, PRM:PRM). (a) Reliability. (b) Maximum latency. (c) Message overhead.

TABLE 1
Comparison of FaReCast with Other Protocols under Different Settings of Fan-in and Fan-out Factor. The Target Number of Nodes Is 100,000 and the Percentage of Failure Is 30 Percent

Fan-In		2 (4 neighbors for Epidemic)			4 (8 neighbors for Epidemic)			6 (12 neighbors for Epidemic)		
Fan-Out Factor		2	4	6	2	4	6	2	4	6
Reliability	FaReCast	0.9994	0.999999	1	1	1	1	1	1	1
	MultipleTree	0.961998	0.941336	0.932087	0.998884	0.996541	0.9951	0.999964	0.999786	0.999699
	Epidemic (Mesh)	0.87516			0.999483			0.999993		
Message Overhead	FaReCast	218508	270189	313131	410858	538152	666330	592401	725890	944751
	MultipleTree	194040	192206	190775	433694	428513	425835	667016	664780	659170
	Epidemic (Mesh)	342531			846302			1315570		
Maximum Latency	FaReCast	9222	4207	2793	3956	2235	1861	3359	2030	1795
	MultipleTree	4964	3560	2921	2508	2146	1872	1881	1706	1601
	Epidemic (Mesh)	3715			2041			1610		

which depends on the initial setting. That is, FaReCast selectively use overlay links by detecting failures and triggering multidirectional multicasting. Notice that under 30 percent failure the message overhead of FaReCast is similar to Multiple Tree, but under 30 percent failure the message overhead of FaReCast is similar to Epidemic; however, FaReCast achieves 100 percent reliability, as compared to 90 percent reliability of the Epidemic protocol under 30 percent failure. For better understanding of the dynamic message overhead of FaReCast, the Section 1.4.3 of supplement file available online show the comparision between PRM and FaReCast in the aspects of message overhead and reliability.

7.2 Tuning FaReCast with Fan-in and Fan-out Factor

To understand the effect of adjustment of fan-in and fan-out factor, we compare FaReCast with other protocols under various settings of fan-in and fan-out factor in terms of reliability, message overhead (total number of sent messages) and maximum latency. Table 1 shows this comparison with 100,000 nodes and 30 percent failure. In this table, the results of SplitStream is omitted because it is very similar to that of MultipleTree. For Epidemic, fan-out factor is not considered, but the number of neighbors is two times more than fan-in value for other protocols. In Table 1, for all of protocols, as fan-in increases, message overhead as well as reliability increases. Also, bigger fan-in reduces maximum latency. That is, the effect of fan-in is similar to all of protocols. But, the effect of fan-out factor to FaReCast is different to the other protocols. The only common effect of fan-out factor to all of protocols is that bigger fan-out factor reduces maximum latency. For other protocols including MultipleTree and SplitStream, as fan-out factor increases, reliability slightly decreases and there is no change of message overhead. That is, with bigger fan-out factor, a node failure impacts more children, but there is no way to complement the failure, and reliability decreases. On the other hand, for FaReCast, as fan-out factor increases, message overhead increases and reliability also increases. According to this, to achieve 100 percent reliability, other protocols requires very big fan-in such as over 7, but FaReCast can use small fan-in with big fan-out factor such as the setting of 2 fan-in and 6 fan-out factor.

We also observed that FaReCast can achieve same level of reliability with fewer messages. In Table 1, the message overhead of FaReCast is almost half of that of MultipleTree (also SplitStream) and almost quarter of that of Epidemic. Unlike other protocols, which send the data through all designated links (children or neighbors), FaReCast selectively sends the data to the subset of all possible links (children, parents, and leafs) which suppose to be affected by node failures. But, with fewer messages FaReCast exhibits longer maximum latency. To reduce maximum latency, FaReCast can increase fan-in or fan-out factor. Note that, in FaReCast, a large fan-in just wastes messages, on the other hand a large fan-out factor is more effective in reducing dissemination latency.

7.3 Effect of Random Timer of L2L Dissemination

As discussed in Section 5.2, L2L dissemination suffers from the problem that multiple leaf nodes may send messages to the same nodes simultaneously. To diminish the effect of this conflict, we can use the random waiting time to the L2L dissemination by multiplying a random integer to T_w. Fig. 6b plots the message overhead for different policies

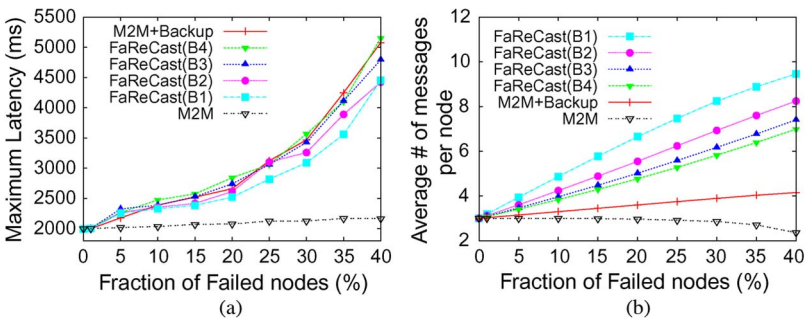

Fig. 6. Performance comparison between various waiting policies for L2L dissemination. (a) Maximum Latency. (b) Message Overhead.

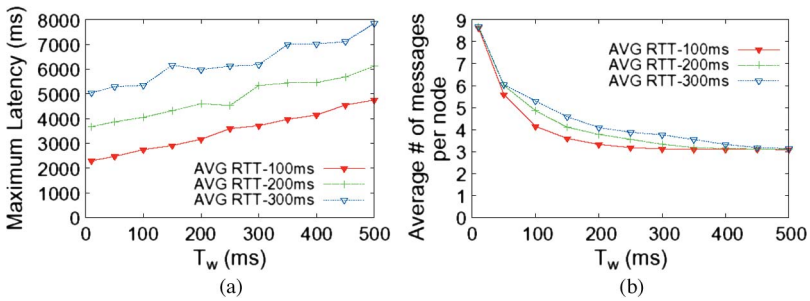

Fig. 7. Performance of FaReCast($F_i = 3$, $F_o = 3$) with various T_w under 30 percent failures. (a) Maximum latency. (b) Message overhead.

that obtain random waiting times differently. Each policy uses a different range of the random integers multiplied to T_w. B1, B2, B3, and B4 policy uses 1, 2, 3, and 4 as N, which is the maximum value the random integer can have. So, policy B1 causes all nodes wait for T_w and with policy B4, 25 percent of the nodes each wait for T_w, $2 * T_w$, $3 * T_w$, and $4 * T_w$, respectively. As we expected, B1 achieves the biggest message overhead and the lowest maximum latency. At the other end, B4 achieves the lowest message overhead and the highest maximum latency. However, the difference in maximum latency is not that big. Even though large N for the random timer helps reducing the message overhead, this improvement saturates—In Fig. 6b, B2 can save about 25 percent of the message overhead than B1, but B4 can only save about 10 percent of the message overhead than B3.

7.4 Effect of T_w

Fig. 7 shows the latency and the overhead as a function of T_w in FaReCast with 30 percent of failures. We vary the average RTT between any two nodes from 100 ms to 300 ms. As T_w increases, the maximum latency increases linearly. But, the overhead decreases exponentially as T_w increases. The main reason is that the longer T_w delays the triggering point of multidirectional multicasting. On the other hand, the delayed multidirectional multicasting reduces redundant messages. We note that if T_w is bigger than the average RTT, the overhead does not decrease significantly but the latency still increases linearly. According to this, it is good to set T_w to the average RTT.

7.5 Implementation of FaReCast

We implemented the FaReCast protocol in a middleware suite and evaluated it on a campus cluster testbed with nodes connected through a wide-area Internet emulator (ModelNet) [1]. The detail descriptions of the setting of ModelNet and the implementation of FaReCast system is found in Section 1.3 of the supplement file available online.

8 Conclusion

We proposed FaReCast, a Fast and Reliable data dissemination system for flash dissemination scenarios such as earthquake early warning where the target network is large and potentially unreliable. FaReCast constructs a M2M ALM structure where each node has multiple children guaranteeing the fast delivery of the message as well as multiple parents ensuring the reliability of the delivery. The FaReCast M2M ALM structure is accompanied by a multidirectional multicasting algorithm which sends data not only to the children but also to the selected parents or the selected siblings. Unlike ack-based communication, the multidirectional multicasting algorithms proactively figure out the direction of failures and target message resends along the direction. As compared to trees using a traditional multicasting algorithm, we observe an 80 percent improvement in reliability under 20 percent failed nodes with no significant increase in latency for over 99 percent of nodes. Under extreme failures, e.g. more than 30 percent failed nodes, we observe that existing mesh-based protocols exhibit low reliability, FaReCast achieves 100 percent reliability with a small increase in latency and overhead.

Acknowledgment

This study was financially supported by Chonnam National University, 2012.

References

[1] MODELNET Modelnet. [Online]. Available: http://modelnet.ucsd.edu/
[2] FREEPASTRY FreePastry. [Online]. Available: http://www.freepastry.org/FreePastry/
[3] L. Ciavattone, A. Morton, and G. Ramachandran, ''Standardized Active Measurements on a Tier 1 IP Backbone,'' *IEEE Commun. Mag.*, vol. 41, no. 6, pp. 90-97, June 2003.
[4] Y. Chu, S.G. Rao, S. Seshan, and H. Zhang, ''A Case for End System Multicast,'' in *Proc. ACM SIGMETRICS*, 2000, pp. 1-12.
[5] S. Banerjee, B. Bhattacharjee, and C. Kommareddy, ''Scalable Application Layer Multicast,'' in *Proc. SIGCOMM*, 2002, pp. 205-217.
[6] D.A. Tran, K.A. Hua, and T.T. Do, ''ZIGZAG: An Efficient Peer-to-Peer Scheme for Media Streaming,'' in *Proc. IEEE INFOCOM*, 2003, pp. 1283-1292.
[7] A. Rowstron, A. Kermarrec, M. Castro, and P. Druschel, ''Scribe: The Design of a Large-Scale Event Notification Infrastructure,'' in *Netw. Grp. Commun.*, 2001, pp. 30-43.
[8] S. El-Ansary, L.O. Alima, P. Brand, and S. Haridi, ''Efficient Broadcast in Structured P2P Networks,'' in *Proc. IPTPS*, 2003, pp. 304-314.
[9] M. Castro, P. Druschel, A.-M. Kermarrec, A. Nandi, A. Rowstron, and A. Singh, ''SplitStream: High-Bandwidth Multicast in a Cooperative Environment,'' in *Proc. SOSP*, 2003, pp. 298-313.
[10] D. Kostić, A. Rodriguez, J. Albrecht, and A. Vahdat, ''Bullet: High Bandwidth Data Dissemination Using an Overlay Mesh,'' in *Proc. SOSP*, 2003, pp. 282-297.
[11] V. Pai, K. Kumar, K. Tamilmani, V. Sambamurthy, and A.E. Mohr, ''Chainsaw: Eliminating Trees from Overlay Multicast,'' in *Proc. IPTPS*, 2005, pp. 127-140.

[12] BITTORRENT, BitTorrent. [Online]. Available: http://www.bittorrent.com/
[13] M. Deshpande, B. Xing, I. Lazardis, B. Hore, N. Venkatasubramanian, and S. Mehrotra, ''CREW: A Gossip-Based Flash-Dissemination System,'' in *Proc. IEEE ICDCS*, 2006, pp. 1-8.
[14] M. Deshpande, A. Amit, M. Chang, N. Venkatasubramanian, and S. Mehrotra, ''Flashback: A Peer-to-Peer Webserver for Handling Flash Crowds,'' in *Proc. ICDCS*, 2007, pp. 1-8.
[15] A.C. Snoeren, K. Conley, and D.K. Gifford, ''Mesh-Based Content Routing Using XML,'' in *Proc. SOSP*, 2001, pp. 160-173.
[16] M. Bansal and A. Zakhor, ''Path Diversity Based Techniques for Resilient Overlay Multimedia Multicast,'' in *Proc. PCS*, 2004, pp. 1-6.
[17] R. Tian, Q. Zhang, Z. Xiang, Y. Xiong, X. Li, and W. Zhu, ''Robust and Efficient Path Diversity in Application-Layer Multicast for Video Streaming,'' *IEEE Trans. Circuits Syst. Video Technol.*, vol. 15, no. 8, pp. 961-972, Aug. 2005.
[18] D. Frey and A.L. Murphy, ''Failure-Tolerant Overlay Trees for Large-Scale Dynamic Networks,'' in *Proc. P2P*, 2008, pp. 351-361.
[19] R. Rodrigues and B. Liskov, ''High Availability in DHTs: Erasure Coding vs. Replication,'' in *Proc. IPTPS*, 2005, pp. 226-239.
[20] V.N. Padmanabhan, H.J. Wang, and P.A. Chow, ''Resilient Peer-to-Peer Streaming,'' in *Proc. IEEE ICNP*, 2003, pp. 16-27.
[21] V. Venkataraman, P. Francisy, and J. Calandrinoz, ''Chunkyspread: Multitree Unstructured Peer to Peer Multicast,'' in *Proc. IPTPS*, 2006, pp. 1-6.
[22] M. Ferreira, J. Leitao, and L. Rodrigues, ''Thicket: A Protocol for Building and Maintaining Multiple Trees in a P2P Overlay,'' in *Proc. IEEE SRDS*, 2010, pp. 293-302.
[23] M. Yang and Y. Yang, ''A Peer-to-Peer Tree Based Reliable Multicast Protocol,'' in *Proc. IEEE GLOBECOM*, 2006, pp. 1-5.
[24] B. Rong, I. Khalil, and Z. Tari, ''Making Application Layer Multicast Reliable Is Feasible,'' in *Proc. IEEE LCN*, 2006, pp. 483-490.
[25] J. Zhang, L. Liu, C. Pu, and M. Ammar, ''Reliable Peer-to-Peer End System Multicasting through Replication,'' in *Proc. P2P*, 2004, pp. 235-242.
[26] T. Kusumoto, Y. Kunichika, J. Katto, and S. Okubo, ''Tree-Based Application Layer Multicast Using Proactive Route Maintenance and its Implementation,'' in *Proc. P2PMMS*, 2005, pp. 49-58.
[27] S. Birrer and F. Bustamante, ''Resilient Peer-to-Peer Multicast without the Cost,'' in *Proc. MMCN*, 2005, pp. 113-120.
[28] S. Banerjee, S. Lee, B. Bhattacharjee, and A. Srinivasan, ''Resilient Multicast Using Overlays,'' *IEEE Trans. Netw.*, vol. 14, no. 2, pp. 237-248, Apr. 2006.
[29] A. Nandi, B. Bhattacharjee, and P. Druschel, ''What a Mesh: Understanding the Design Tradeoffs for Streaming Multicast,'' in *Proc. SIGMETRICS*, 2009, pp. 1-2.
[30] X. Hou and S. Wu, ''Transient Analysis of an Overlay Multicast Protocol,'' in *Proc. IEEE CCGRID*, 2009, pp. 556-561.
[31] A. Rowstron and P. Druschel, ''Storage Management and Caching in PAST, A Large-Scale, Persistent Peer-to-Peer Storage Utility,'' in *Proc. 18th ACM SOSP*, Lake Louise, AB, Canada, Oct. 2001, pp. 188-201.
[32] K. Kim and D. Park, ''Reducing Replication Overhead for Data Durability in DHT Based P2P System,'' *IEICE Trans. Inf. Syst.*, vol. E90-D, no. 9, pp. 1452-1455, Sept. 2007.

Kyungbaek Kim received the BS, MS, and PhD degrees in electrical engineering and computer science from Korea Advanced Institute of Science and Technology (KAIST), Korea, in 1999, 2001, and 2007, respectively. He is currently an Assistant Professor in the School of Electronics and Computer Engineering at the Chonnam National University, South Korea. Previously, he was a Postdoctoral Researcher in the Department of Computer Sciences, University of California Irvine, Irvine, CA, USA. His research interests are in the design and implementations of distributed systems including peer-to-peer systems, Grid/Clouding systems and social networking systems. His current focus is on the design of geography and socieal correlation aware dissemination in the context of peer-to-peer and overlay systems. He is a member of the USENIX, the IEICE and the IEEE.

Sharad Mehrotra received the PhD in computer science from the University of Texas at Austin, in 1993. He is currently a Professor in the Department of Computer Science at the University of California, Irvine (UCI), Irvine, CA, USA and the Director of the Center for Emergency Response Technologies. Previously, he was a Professor at the University of Illinois at Urbana-Champaign (UIUC). He has received numerous awards and honors, including SIGMOD best paper award 2001, DASFAA best paper award 2004, and CAREER Award 1998 from NSF. His primary research interests are in the area of database management, distributed systems, and data analysis. He is a member of the IEEE.

Nalini Venkatasubramanian received the MS and PhD degrees in computer science from the University of Illinois in Urbana-Champaign. She is currently a Professor in the School of Information and Computer Science at the University of California Irvine, Irvine, CA, USA. She has had signifficant research and industry experience in the areas of distributed systems, adaptive middleware, mobile computing, distributed multimedia servers, formal methods and object-oriented databases. She is the recipient of the prestigious NSF Career Award in 1999, an Undergraduate Teaching Excellence Award from the University of California, Irvine, in 2002 and multiple best paper awards. She is a member of ACM, and has served extensively in the program and organizing committees of conferences on middleware, distributed systems and multimedia. Prior to arriving at UC Irvine, Nalini was a Research Staff Member at the Hewlett-Packard Laboratories in Palo Alto, CA. She is a member of the IEEE.