Churn-Resilient Protocol for Massive Data Dissemination in P2P Networks

Zhenyu Li, *Member*, *IEEE*, Gaogang Xie, *Member*, *IEEE*,
Kai Hwang, *Fellow*, *IEEE*, and Zhongcheng Li, *Member*, *IEEE*

Abstract—Massive data dissemination is often disrupted by frequent join and departure or failure of client nodes in a peer-to-peer (P2P) network. We propose a new churn-resilient protocol (CRP) to assure alternating path and data proximity to accelerate the data dissemination process under network churn. The CRP enables the construction of proximity-aware P2P content delivery systems. We present new data dissemination algorithms using this proximity-aware overlay design. We simulated P2P networks up to 20,000 nodes to validate the claimed advantages. Specifically, we make four technical contributions: 1). The CRP scheme promotes proximity awareness, dynamic load balancing, and resilience to node failures and network anomalies. 2). The proximity-aware overlay network has a 28-50 percent speed gain in massive data dissemination, compared with the use of scope-flooding or epidemic tree schemes in unstructured P2P networks. 3). The CRP-enabled network requires only 1/3 of the control messages used in a large CAM-Chord network. 4) Even with 40 percent of node failures, the CRP network guarantees atomic broadcast of all data items. These results clearly demonstrate the scalability and robustness of CRP networks under churn conditions. The scheme appeals especially to web-scale applications in digital content delivery, network worm containment, and consumer relationship management over hundreds of datacenters in cloud computing services.

Index Terms—P2P networks, data dissemination, multicast algorithms, distributed systems, cloud computing.

1 Introduction

Many Internet applications need to perform large-scale data dissemination from multiple sources. However, this process is often disturbed by network churn conditions caused by unexpected node failures. Good examples include wide-area worm signature distribution to prevent network epidemic outbreaks, massive data mining in consumer relationship management (CRM), multiplayer gaming, remote design collaboration among many participants, etc. [10]

Distributed IP multicast was not widely deployed due to various technical and marketing reasons [7]. Application layer multicast using peer-to-peer (P2P) overlay networks is a promising alternative to IP multicast. This paper proposes a new churn-resilient protocol (CRP) to achieve fast and reliable data dissemination in P2P networks. We consider dynamic P2P networks under churn conditions. Moreover, we assume skewed distribution of the node capacity.

Our ultimate goal is to achieve sustained throughput and fast data dissemination rate from any data source with small overhead even under frequent node failures. To this end, we propose a *churn-resilient protocol* for fast massive multisource data dissemination.

- *Z. Li, G. Xie, and Z. Li are with the Institute of Computing Technology, Chinese Academy of Sciences, No. 6, South Road, Kexueyuan, Zhongguancun, PO Box 2704, Beijing 100190, P.R. China. E-mail: {zyli, xie, zcli}@ict.ac.cn.*
- *K. Hwang is with the Department of Electrical Engineering, University of Southern California, EEB-212, Los Angeles, CA 90089-2560. E-mail: kaihwang@usc.edu.*

Manuscript received 30 Oct. 2009; revised 11 Feb. 2010; accepted 30 July 2010; published online 3 Jan. 2011.
Recommended for acceptance by Y. Hu.
For information on obtaining reprints of this article, please send e-mail to: tpds@computer.org, and reference IEEECS Log Number TPDS-2009-10-0538.
Digital Object Identifier no. 10.1109/TPDS.2011.15.

The topology of the CRP-enabled overlay design looks like a ring augmented with extra chord links. However, the overlay topology differs from the DHT Chord [20] in that the extra links are not fixed fingers from nodes. Instead, the chord links are established among nodes satisfying certain proximity properties.

We consider two proximity factors to enhance date delivery performance. This proximity-aware overlay enables the optimal choice of overlay links for fast data dissemination. Inspired by the Chord structure, we associate with each node a successor list of $\log n$ nodes along the ring, in order to improve churn resilience, where n is the overlay size. The CRP-enabled overlay is self-adjusted after any node failure.

A shared spanning tree is extracted from the CRP overlay. This tree structure leverages links with minimized latency. Instead of using pure tree broadcast, we apply scope flooding in the first few hops and tree traversing in the remaining hops. This hybrid design saves delivery time with very small amount of redundancy. The delivery tree takes advantage of using alternate overlay links to achieve fault tolerance. The tree can be recovered from node failures, even when the failure rate of every single node is as high as e^{-a} $(a > 1)$.

We report both theoretical findings and simulated experimental results. The contributions of our work are summarized below in four technical aspects:

1. The CRP-enabled overlay design promotes proximity awareness, dynamic load balancing, and resilience to network churns by node failures.
2. CRP networks reduce the average delivery delay by 28 to 50 percent, compared with using the scope flooding [6] and epidemic broadcast trees [13], [21].
3. Even with 40 percent of node failures, the CRP still guarantees atomic broadcast in data dissemination.

4. With 10,000 nodes, the CRP network requires to use only 1/3 of the control messages used in a large CAM-Chord [25] network.

2 Related Work

There are two types of data dissemination methods: *mesh-based pull* and *tree-based push methods*. The pull methods use swarming content delivery. Each node advertises to its neighbors which messages it has received, and the neighbors explicitly request messages if needed. Two representative pull schemes are Chainsaw [17] and PRIME [15]. PRIME incorporates swarming into streaming applications and points out the design tradeoffs of such systems. To achieve a low delay, a node in such systems should advertise as soon as it receives a new message, increasing the control overhead greatly [22].

The tree-based push methods achieve a fast dissemination with low overhead. The main concern is the vulnerability to node failures. Our CRP method falls into this type. The CRP extracts dissemination tree from a churn-resilient overlay, which provides sufficient alternative overlay links for tree to achieve good fault resilience. The preliminary idea was reported in an IEEE *IPDPS-2008* paper [14].

CAM-Chord [25] builds on Chord [20] overlays. It takes the heterogeneity of node capacities into account. A node's degree is proportional to its capacity. DHT-based structured overlays, such as Chord and Pastry [19], are hard to be adapted for optimization and the maintenance overhead is high.

ACOM [6] implements data dissemination by combining scope-limited flooding and random walks on overlay networks. It requires $O(n/\log n)$ replicated messages to send a data item, where n is the number of nodes. Considerable amount of message replication and long delivery delay are the major concerns.

The Scribe [3] and SplitStream [4] apply multicast services on top of the Pastry. The Scribe builds one tree for each multicast group, while SplitStream builds multiple trees to achieve a higher bandwidth multicast service by using Scribe. These methods are all designed for the data distribution from a single source node.

GoCast [21] and Plumtree [13] apply a tree-based gossip scheme for data dissemination. While GoCast pays attention to network proximity, both methods ignore the disparity in node capacities. High overhead due to periodic gossip is also a major concern here. Moreover, they do not exploit the benefit of scope flooding and thus causing a long delivery delay.

SMesh [11] builds a topology-aware mesh overlay as a control plane for P2P streaming. It constructs tree stuctures for multiple groups (i.e., channels). Our CRP differentiates SMesh in three aspects: heterogeneity awareness, churn resilience, and fast data dissemination using hybrid delivery algorithms. Probabilistic gossip-based methods [12] are limited by using too many replicated messages. SCAMP [8] applies a P2P membership management as a gossip-based protocol.

3 Proximity-Aware Overlay Networks

In this section, we use an example to illustrate the basic concepts of proximity awareness in constructing a powerful overlay for massive data dissemination in a P2P environment. The idea is to map the physical network into the virtual overlay by considering the fact that different links may experience different latencies or link delays and nodes may have different capacities.

TABLE 1
Notations and Basic Definitions

n	The number of nodes involved in the overlay network
c_x	Capacity of node x
d_x	Node degree of node x, $d_x \leq c_x$
D_x	Degree threshold for node x to build chord links
d	Average node degree over all nodes.
q	Target message redundancy ratio
α	Weighting factor between network and capacity proximity
t	Hop count in scope-flooding method
x_f	Node x's farthest neighbor with the highest link weight
m_x	The size of x's successor list

In a physical network, the link latency is measurable. The node capacity of a node represents the maximum number of adjacent nodes to which it can forward the data items, concurrently. A node's out degree is bounded by its capacity. Table 1 summarizes the notations used in this paper.

The proximity-aware overlay is built around a unidirectional ring with extra bidirectional chord links. The overlay is heterogeneous since links are associated with different weights. A node's neighbors are those that are directly connected by either ring or chord links with it, while its chord neighbors are those that are connected by chord links. Initially, the base ring is assumed empty before any node joining the system. The first node is located at any position on the base ring.

An n-node ($n > 0$) base ring is split into n equal-distance intervals. The $(n+1)$th node is placed on any interval with a probability $1/n$. Each node x has two immediate neighbors: the predecessor and successor along the ring, denoted as prd_x and suc_x, respectively.

We apply both *network proximity* and *capacity proximity* in CRP protocol. The network proximity is measured by the latency or closeness of two nodes in physical IP networks. This proximity enables faster transferring data items. The capacity proximity is measured by the closeness of nodes with respect to node capacities. The capacity proximity allows us to put high-capacity nodes at higher positions on delivery trees, which reduces the delivery hop count.

Fig. 1a shows the initial base ring with three nodes. The ring is assumed unidirectional with clockwise links shown by dotted edges. We assume that all ring links are weighted with 1 unit. Suppose the node capacities for A, B, and C are 3, 4, and 2, respectively. Node A's successor and predecessor are B and C, respectively. The fourth node D with capacity 3 is inserted into the interval between A and C in Fig. 1b. A chord link weighted with 0.7 is added between nodes B and D.

In Fig. 1c, the fifth node E with capacity 4 joins between nodes C and D. Two new chord links are added, with weights 0.6 and 0.8. The added links should be limited by the node capacity of all end nodes. A newly inserted node

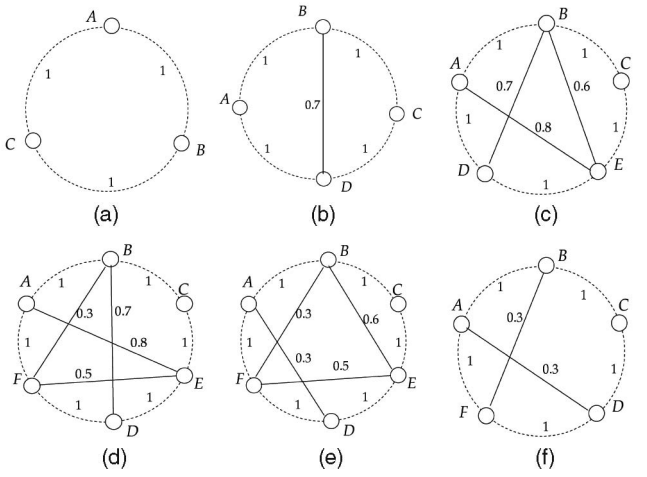


Fig. 1. Example proximity-aware overlay design. (a) Initial overlay with three nodes; (b), (c), and (d) after the joining of the fourth (D), fifth (E), and sixth (F) nodes, respectively; (e) the overlay after link weight adjustment; and (f) after the departure of node E.

may establish new chord links with other nodes by removing some existing chord links.

An existing chord link is removed when it has higher weight than the added new links. In Fig. 1d, a new node F is inserted between nodes A and D. Two new chord links are added to connect F with nodes B and E replacing the old link between B and E in Fig. 1c. The newly added chord links must have lower weights than those of the links being replaced.

The overlay is dynamically adapted by replacing higher weight chord links with lower ones, without changing any node's degrees. For example, in Fig. 1e, the links between B and D, A and E in Fig. 1d are replaced with two new lower weight links between A and D, B and E. Fig. 1f shows the overlay after the departure of node E in Fig. 1d. The links associated with E are removed.

The base ring is always connected as nodes keep their correct successors. To improve the connectivity, the CRP borrows the idea of successor list from Chord. Each node maintains a list of $O(\log n)$ successor nodes down the base ring. If a node detects a faulty successor, it replaces that successor with the first live node in the list. In Fig. 1e, for example, node C keeps two successor nodes E and D. If node E fails or leaves, C takes D as its new successor.

4 Node Joining and Departures

This section specifies the node joining and departure processes for using the CRP-enabled P2P networks.

4.1 Node Joining Process

The first node joins the overlay by choosing any position on the empty base ring. It sets both the predecessor and the successor to itself. An n-node ($n > 0$) base ring is split into n intervals by the nodes. The $(n+1)$th node x uniformly chooses each interval at random to join with equal probability $1/n$. Suppose x chooses the ring interval which starts from z, then x joins the base ring as z's successor.

Each node x keeps a successor list, which contains the first m_x nodes following x in a clockwise direction on the ring. These nodes keep the same order as they appear in the ring. The first entry in x's list is its successor and the ith entry is i hops away from x in the clockwise direction of the ring. A new node x copies the list from its predecessor z as its own list, while z updates its list by adding x at the first position of the list. Thus, z's successor list size is increased by 1.

The addition of x on the ring affects the successor lists of other existing nodes that contain x's successor. The nodes preceding $2m_x$ hops from the new node x along the base ring, update their lists without changing the list sizes. Since the size of the successor list on each node is estimated equally due to the *successor list balance method*, these $2m_x$ covers all the nodes whose successor lists should be updated.

In the successor list balance method, each node periodically sends a balancing message to a randomly chosen neighbor to even their list sizes. The balancing messages are piggybacked on heartbeat messages between neighbors.

We leverage a controlled random walk on the overlay for predecessor chosen for the new node. The new node contacts any existing node w when joining. The node w uniformly chooses a neighbor at random to start a random walk for $\log n$ hops. The last node of the walk is selected as the new node's predecessor. The proofs of the following two theorems can be found in [14].

Theorem 1. *The average size of node successor list converges to $O(\log n)$, where n is network size.*

Theorem 2. *If any node fails with a probability e^{-a} $(a > 1)$ independently, then the ring structure is connected with a high probability more than $1 - n^{1-a}$, where e is base of natural logarithm.*

4.2 Node Departure Process

Node departure process is a reverse procedure of node join. The departure node x asks a randomly chosen node y to remove the last (i.e., m_yth) entry from y's successor list. Then, it notifies the nodes, preceding it $2m_x$ hops on the ring, to update their lists by removing x and adding the successor of the last entry at the end of the lists, leaving the list sizes unchanged. If x fails, its predecessor can detect the failure by heartbeat messages and takes over the departure operation for it.

5 Overlay Update Operations

We begin this section with the description of link proximity weight (PW), and then specify the overlay management operations.

Each node contributes to the system performance by enlarging its node capacity. If a node x's degree falls below a preset threshold D_x, it falls into the *degree interval*. Within this interval, the node periodically calls *connection* operation to increase its degree by building more chord links. Otherwise, if x's degree exceeds D_x, it falls into the *proximity interval*. Within this interval, the node periodically calls *adjustment* operation to replace high-weight chord links with low-weight ones.

The threshold D_x for a node x is set at $c_x \cdot \beta$, where $0 < \beta \leq 1$. To achieve a higher system capacity, the default value of β is set to be 0.8.

5.1 Computing the Chord Link Weight

Let $e_{x,y} = (x, y)$ denotes the overlay link that connects node x and node y ($x \neq y$). Following the works [9], [18], [23], we consider network proximity when establish chord links, aiming at saving bandwidth and delivery time. We define *network proximity weight (NPW)* of a link as the one-way latency of that link normalized by the latency of a ring link. NPW of link $e_{x,y}$ is expressed as follows, where $d(x, y)$ is the link's one-way latency and suc_x is x's successor.

$$NPW(e_{x,y}) = \frac{d(x,y)}{d(x, suc_x)}. \quad (1)$$

We also consider node capacity proximity, which allows high-capacity nodes highly connected. A high-capacity node is chosen as the root of the delivery tree. Thus, high-capacity nodes are placed at higher locations in the tree, which yields a reduced tree depth [2].

We define *capacity proximity weight (CPW)* of a link $e_{x,y}$ as the capacity difference between x and y, normalized by the capacity difference between x and suc_x. It is expressed as follows, where $c(x, y) = |c_x - c_y|$.

$$CPW(e_{x,y}) = \frac{c(x,y)}{c(x, suc_x)}. \quad (2)$$

We use above two link weight function to establish low-weight chord links and prune high-weight ones. However, these two weight functions may be conflicting. A link with smaller latency may have a larger capacity difference. We combine them to yield a single proximity metric, called *Proximity Weight*, as below

$$PW(e_{x,y}) = \alpha \times CPW(e_{x,y}) + (1 - \alpha) \times NPW(e_{x,y}), \quad (3)$$

where α is a *weighting factor*, thus $0 \leq \alpha \leq 1$.

The configurable parameter α gives us the flexibility to fine tune the PW function for different application scenarios. A lower value of α implies higher network proximity, while a higher α implies higher capacity proximity. We further extend the weight functions to a set of two links, $S = \{(x, y), (r, s)\}$. The NPW, CPW, and PW for the link set S are defined as follows:

$$NPW(S) = \frac{1}{2}\left[\frac{d(x,y)}{d(x, suc_x)} + \frac{d(r,s)}{d(r, suc_r)}\right], \quad (4)$$

$$CPW(S) = \frac{1}{2}\left[\frac{c(x,y)}{c(x, suc_x)} + \frac{c(r,s)}{c(r, suc_r)}\right], \quad (5)$$

$$PW(S) = \alpha \times CPW(S) + (1 - \alpha) \times NPW(S). \quad (6)$$

One implement concern is the estimation of link latencies. We employ a landmark based scheme [18], [23]. We randomly choose b public accessible sites as landmarks. A node x measures the distances to the landmarks, yielding a landmark vector $<d_1^x, d_1^x, \ldots, d_b^x>$. Physically close nodes are likely to have similar vectors. Thus, the link latency is computed as follows:

$$d(x, y) = \sqrt{\sum_{i=1}^{b} \left(d_i^x - d_i^y\right)^2}. \quad (7)$$

Landmark vectors are piggybacked on other messages if possible. The results in [23] have shown that using 15 landmark nodes can yield good latency predication. Thus, we set the value of b to 15. The landmark-based scheme produces coarse-grain proximity measurements. Latency-sensitive applications can use Htrae [1] to get more accurate predication.

In practice, landmark nodes can be taken from well-known DNS servers. The capacity of a node may change due to the variation of the node's available resource. The corresponding links' CPWs should be updated when a node's capacity changes.

5.2 Neighborhood Operations

The CRP introduces two management operations for neighborhood management, namely *connection* and *adjustment*. Every single node periodically calls these two operations. The connection operation increases the caller node's degree, while the adjustment one builds low-weight links without changing any node's degree.

Suppose that, among all chord links of a node x, the link between x and x_f has highest weight. We denote x_f as x's *farthest neighbor*, formally defined by

$$\forall e_{x,i}, \quad PW(e_{x,x_f}) \geq PW(e_{x,i}), \quad x_f, i \in R_{chd}(x), \quad (8)$$

where $R_{chd}(x)$ is the set of x's chord neighbors. In Fig. 1d, B's furthest neighbor is D.

Connection operation. This operation is called periodically by a node x when its degree falls below the threshold D_x. The operation builds chord links to increase node degree. The node x randomly chooses a neighbor to send a *connection message*. When a node y receives a connection message initialized by node x, it either builds a link with x or forwards the message, depending on its current neighborhood information and the link weight between x and y. The message is forwarded by $2\log n$ hops at most. The pseudocode of this procedure is given in Algorithm 1.

Algorithm 1. Insertion of Chord Links
Input: node y, y's neighborhood, the connection message initialized by node x, x's degree
Output: New chord links to be established between node x and other existing nodes
Procedure:
1. ***if*** $x \in R(y)$ ***then***
2: ***goto*** Step 13
3: ***if*** $d_y < D_y$ ***then***
4: Establish a chord link between x and y
5: ***else if*** $PW(y, x) < PW(y, y_f)$ ***then***
6: ***if*** $D_y \leq d_y < c_y$ ***then***
7: Establish a chord link between x and y
8: ***else if*** $x \notin R(y_f)$ and $d_x + 2 \leq c_x$ ***then***
10: Remove the chord link between y and y_f
11: Establish a chord link between x and y
12: Establish a chord link between x and y_f
13: ***else*** *forward* the message randomly to a neighbor of y

The rationale behind the algorithm is that, for the node y, if it is in its *degree interval*, it builds a chord link with x. Otherwise, it builds a chord link with x only if the link has

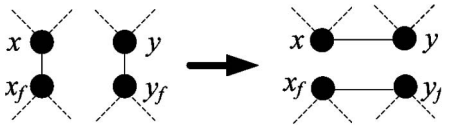


Fig. 2. Adjustment operation without changes of node degrees.

lower weight. In Fig. 1b, after the node D joins, a chord link is established between D and B since both of them are in their degree intervals. In Fig. 1d, F builds chord links with B and E by removing the chord link between B and E since the chord link between B and E has a higher weight than the newly added links.

Adjustment operation. This operation is called periodically by a node x when its degree exceeds D_x. The operation adapts x's neighborhood according to proximity information, without changing any node's degree. The node x uniformly selects a two-hop away neighbor y at random by sending an *adjustment message* with hop limited at 2 for neighborhood adaptation.

Fig. 2 depicts the adjustment operation. The node y is two-hop away from x. Suppose x is not a neighbor of y and x_f is not a neighbor of y_f. The neighborhood transition in Fig. 2 is performed with a probability p_{trs}.

$$p_{trs} = \begin{cases} 1, & PW(S_1) > PW(S_2), \\ e^{(PW(S_1)-PW(S_2))/0.01}, & otherwise, \end{cases} \quad (9)$$

where $PW(S_1)$ is the proximity weight of the link set S_1, computed according to (6). The set S_1 consists of link e_{x,x_f} and e_{y,y_f}, while S_2 consists of link $e_{x,y}$ and e_{x_f,y_f}.

This process is inspired by Localiser [16]. It is similar to *simulated annealing* [5] with a fixed temperature 0.01, which helps prevent getting stuck in local optimum. The selection of the "temperature" is based on the fact that, with a smaller value, fewer transitions would be performed, saving the control overhead.

The adjustment operation is also illustrated in Fig. 1e. The chord links between B and D, A and E in Fig. 1d are replaced with two new lower weight links between A and D, B and E. During the adjustment process, node degrees remain unchanged.

In a stable network (no node joins and leaves), once a node x's degree exceeds the threshold D_x, the degree will never drop below that threshold. This guarantees a high system capacity, and reduces the number of adaptation iterations required, saving the control overhead.

5.3 Adaptive Management Period

Each node periodically calls *connection* or *adjustment* operation to adapt its neighborhood. The key problem is to decide the interval between two successive operations. Each node adaptively varies its management period based on its neighbor churn. The process is illustrated in Fig. 3. If the last two management operations bring no change on its neighborhood, its period grows. The growth of the period is affected by its degree and the degree threshold. When the node falls into its degree interval, the period grows linearly. Otherwise, the period is doubled. If its last management operation changes its neighborhood, the period is halved.

The period variation scheme is based on two insights. First, the linear growth of a node's period increases the number of connection operations. This benefits the degree growth when a node's degree falls below the degree threshold. Second, in order for the overlay to respond to node churn promptly, the period shrinks exponentially to ensure that neighbor sets are updated in a prompt fashion. In Fig. 3, the maximum period $T_{\max}$and the minimum period T_0 are set to 60 s and 1 s, respectively. Each node performs at least one management operation every 60 s.

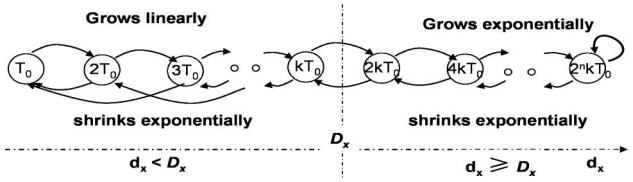


Fig. 3. The switching of adaptive management period before and after the threshold on node degree.

6 Multisource Data Dissemination

In this section, we specify the data dissemination process, which is improved from the scope-flooding and tree-traversing schemes.

6.1 Shared Tree Construction

The shared tree is a spanning tree of the overlay graph. The first joining node is chosen as the root. A new joining node selects the physically closest neighbor in the tree as its parent. Periodically (e.g., every 2 minutes), the root node selects another node with high capacity as a new root to build a new tree. A flooding-like scheme is used for tree construction.

The root sends *tree construction messages* to all its neighbors except its successor, since its successor is a random neighbor, not preferred as a tree neighbor. On receiving such a message for the first time, a node y asks the node z which forwards the message to it as its parent, effectively making this overlay link a tree link.

The node y further forwards the message to its overlay neighbors, except its successor and its parent node z. Since each node forwards the message to its predecessor, all nodes can receive the construction messages. Thus, the tree structure spans all nodes.

Fig. 4 shows the spanning tree embedded in the overlay illustrated in Fig. 1e. The root node is B with a capacity 4. The tree construction is similar to DONet [26]. Following the analysis in DONet, we summarize the average height of the delivery tree is bounded by $O(\log_{d-2} n)$, where d is the average node degree.

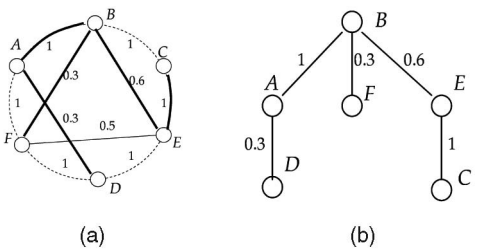


Fig. 4. (a) The spanning tree with root B embedded in the overlay. The thick overlay links are taken as tree links. (b) The delivery tree in (a) expressed as a tree. (a) Overlay links applied. (b) A tree representation.

We choose a high-capacity node as the tree root. Since each node prefers the nodes with similar capacities as its proximity neighbors, the high-capacity nodes are likely to be placed at higher locations in the tree, effectively reducing the tree depth.

In CRP, each node is a possible data source. A simple choice for data dissemination is to broadcast the data items from any node on the tree structure. However, if a leaf node is the data source, then the message only reaches a very small portion of nodes within the initial several hops. For example, in Fig. 4b, suppose D is the source, then only two new nodes (A and B) are reached within two hops.

6.2 Fast Data Dissemination Algorithm

The initial few hops of scope flooding exponentially increase the number of visited nodes at the cost of very limited replicated messages. Motivated by this, CRP leverages scope flooding for the first t hops, then delivers the data messages on the tree from the nodes that have received the data messages. We analyze below the proper choice of t for a given message replication ratio. The message replication ratio is the ratio of duplicated messages to all messages.

Theorem 3. *If a data item is flooded for t hops on the overlay, and then broadcasted on the tree structure, the average number of messages used is bounded by* $n + 2d(d-1)^{t-1}$.

Proof. The flooding uses Q_1 messages, which at most reach Q_1 nodes. The nodes that reach at the last hop simultaneously start to broadcast the message on the tree. The number of these nodes is $n_1 \leq d(d-1)^{t-1}$. To ensure all the left nodes to receive the messages, the tree broadcast requires to use $Q_2 \approx n - Q_1 + 2n_1$ messages. Thus, in total $Q = Q_1 + Q_2 \leq n + 2d(d-1)^{t-1}$ messages are required. □

Given the message replication ratio q, following Theorem 3, the value of t is computed below

Corollary. *Given the message replication ratio q and the average node degree d, the number of hops for flooding t is*

$$t = \begin{cases} \frac{\log(n \times \frac{q}{1-q} \times \frac{d-1}{2d})}{\log(d-1)}, & q > 0, \\ 0, & otherwise. \end{cases} \quad (10)$$

Let $M(id, obj, k)$ be the message for the data item obj, where k is the flooding hop counter initialized as t according to (10). The flooding hop number computed may be noninteger. The pseudocode for the message forwarding by a typical node x is given in Algorithm 2.

Algorithm 2. Multi-source Data Dissemination
Input: A node x, its neighbors, the message $M(id, obj, k)$ received from neighbor y identified by id.
Output: Messages to be sent to all x's neighbors
Procedure:

```
1: if x has not received the message M with id then
2:     if k > 0 then
3:         p = min(1, k)
4:         R' ← R(x) − {y} with prob. p //scope-flooding//
5:         R' ← R_t(x) − {y} with prob. 1 − p //tree traversing//
6:         for each node z ∈ R' do
7:             forward M(id, obj, k − 1) to z
8:     else // tree traversing //
9:         for each node z ∈ R_t(x) − {y} do
10:            forward M(id, obj, k − 1) to z
11: else discard the message M // already processed by x //
```

In (10), the replication ratio q is set by the application users. We take a node's successor list size as the value of $\log n$, and take the average degree of a node's neighbors as an estimation of d.

In Fig. 4b, suppose D is the data source, the target redundancy q is set at 34 percent. According to (10), t is set close to 1 Thus, D first floods the data messages to all its neighbors, A, E, and F. The node A and E then forwards the message to their own tree neighbors B and C, respectively. Thus, all the five nodes are aware of the data item within two hops.

6.3 Churn Resilience and Recovery

Every single node keeps the information of its ancestor nodes on the tree. This information is used to determine whether a particular node is an ancestor node of it or not. The information is updated when the node's ancestors change.

A new node selects a physically closest neighbor already in the tree as its parent. When a node leaves, it notifies its tree neighbors (i.e., its parent and children). Once a node x receives a departure notification from its parent or detects the failure of its parent by heartbeat messages, it actively switches its tree parent.

The node x prefers its proximity neighbor as the new parent, in order to save delivery time. If all its proximity neighbors cannot be selected as its new parent, the node x forces its successor suc_x on the ring as its parent node on the tree. The node suc_x actively switches its parent as x does if it is also a descendant node of x on the tree. The ancestors' information stored on each node is used for loop avoidance. For example, in Fig. 4b, if node A fails, its child node D actively takes its successor F on the base ring as its new parent. If the root node fails, its predecessor takes over the root role.

When building the delivery tree, a node does not forward tree construction message to its successor on the ring. Therefore, in most cases a node's successor is not a descendant of that node. Thus, a node can quickly find its new parent.

The delivery tree is rebuilt in every 2 minutes by selecting another higher capacity node. The purpose is to take advantage of the newly added proximity links on the overlay network. In practice, the interval between tree reconstructions can be adjusted dynamically according to node churns.

If each node on the ring of a CRP-enabled overlay keeps a correct successor, the delivery tree can be recovered from any node failure. Thus, following Theorem 2, the delivery tree can recover from any node failures, even when the node failure rate is as high as e^{-a} ($a > 1$).

7 Simulated Experimental Results

We developed an event-driven P2P simulator to compare CRP with four tree-based designs: CAM-chord, ACOM,

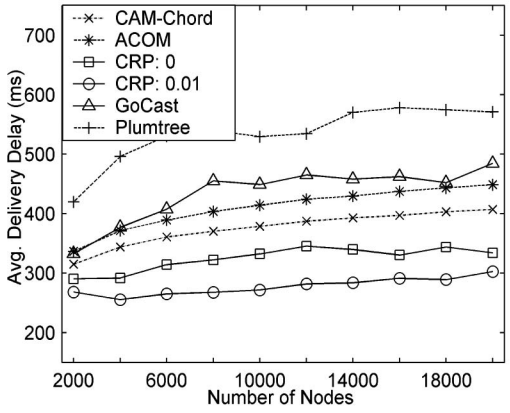


Fig. 5. Average delivery times of five data dissemination schemes over large P2P networks.

GoCast, and Plumtree. The simulator runs on a Linux SMP server consisting of eight processors. We simulate a P2P network up to 20,000 nodes. After any node joins the system, the timer starts to run. All nodes join at once. From the 300th second, 1,000 nodes are randomly chosen to send data items simultaneously.

We apply TS-topology [24] with about 2,500 routers to simulate IP networks. The topology consists of 16 transit routers in four transit domains. Each transit router is attached by five stub domains with 30 stub routers each. Nodes in P2P networks are attached to stub routers randomly. Node capacity follows a bounded Pareto distribution, with a shape 2 and scale $c/2$. The samples outside the range $[c/2, 8c]$ are discarded. The average node capacity c was set at 8. In our CRP, the default weighting factor α was set at 0.3. We focus on the average delivery time, message replication ratio, and control overhead. Each data point represents the average value of five trials.

Figs. 5, 6, and 7 depict the relative performances. The notation "CRP:q" denotes a CRP system with message replication ratio q. It can be found in Fig. 5, CRP outperforms all other designs in terms of delivery time. The CRP has lower delivery hop count due to fast data dissemination and lower average link latency due to the proximity awareness.

Besides, increasing q from 0 to 0.01 significantly reduces the delivery time, showing the benefit of scoped-flooding at the first few hops. From Fig. 6, we see that the message

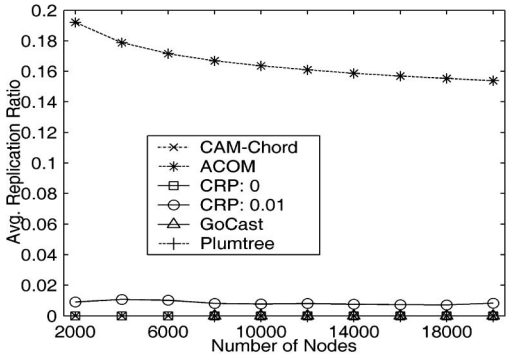


Fig. 6. Average message replication ratio of five data dissemination schemes over large P2P networks.

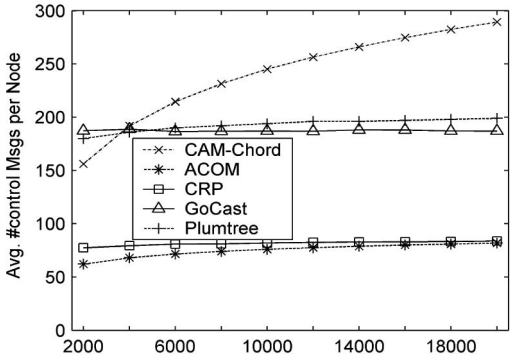


Fig. 7. Control message counts of five data dissemination schemes over large P2P networks.

replication in CRP is close to 0 and can be controlled by adjusting q. Fig. 7 demonstrates that the CRP and ACOM designs require to using the least control messages.

Specially, as the number of nodes grows beyond 10,000, the CRP:0.01 design reduces the average delivery time by 28-50 percent (Fig. 5) from that experienced in other four tree designs. In addition, the CRP requires only 1/3 of the control messages used CAM-Chord (Fig. 7).

Fig. 8 shows the impacts of massive node failures on data delivery time and on the number of unreached live nodes in CRP. It can be found that at the failure rate of as high as 40 percent, all live nodes are reached by the data messages from any source. Thus, the CRP guarantees atomic data dissemination (defined in [12]) even when 40 percent of nodes fail, simultaneously.

8 Conclusions

The newly developed CRP protocol is shown effective to perform massive data dissemination in P2P networks under the churn conditions. The CRP design promotes network proximity and maximizes node capacity for fast data dissemination. It combines the advantages of scope flooding and tree traversing to achieve high performance.

The simulation results show that the CRP design reduces the average delivery time by 28-50 percent with only 1 percent message redundancy, compared with those experienced in scope flooding and epidemic broadcast tree designs. It guarantees atomic broadcast data dissemination, even with 40 percent of node failures.

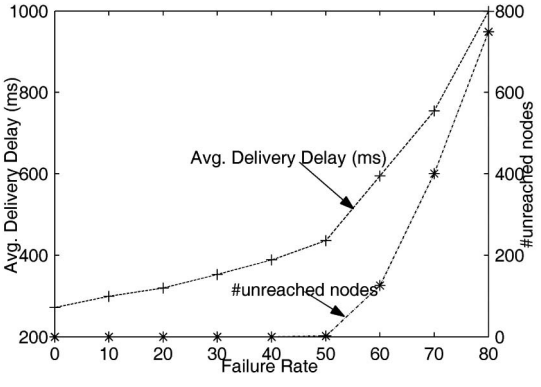


Fig. 8. Impact of the average delivery times of the CRP-enabled P2P networks.

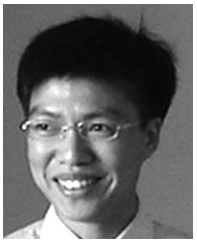
ACKNOWLEDGMENTS

This work was supported by the National Basic Research Program of China under grant No. 2007CB310702, by the National Natural Science Foundation of China under grant No. 60873242 and No. 60903207, by the Instrument Developing Project of the Chinese Academy of Sciences under grant No. YZ200926. An early version of this work [14] was presented in the *Proceedings of IEEE IPDPS '08*.

REFERENCES

[1] S. Agarwal and J.R. Lorch, "Matchmaking for Online Games and Other Latency-Sensitive P2P Systems," *Proc. ACM SIGCOMM,* Aug. 2009.
[2] M. Bishop, S. Rao, and K. Sripanidkulchai, "Considering Priority in Overlay Multicast Protocols under Heterogeneous Environments," *Proc. IEEE INFOCOM,* Apr. 2006.
[3] M. Castro et al., "SCRIBE: A Large-Scale and Decentralized Application-Level Multicast Infrastructure," *IEEE J. Selected Areas in Commun.,* vol. 20, no. 8, pp. 1489-1499, Oct. 2002.
[4] M. Castro, P. Druschel, A.-M. Kermarrec, A. Nandi, A. Rowstron, and A. Singh, "SplitStream: High-Bandwidth Multicast in Cooperative Environments," *Proc. ACM Symp. Operating Systems Principles,* Oct. 2003.
[5] V. Cerny, "Thermodynamical Approach to the Traveling Salesman Problem: An Efficient Simulation Algorithm," *J. Optimization Theory and Applications,* vol. 45, pp. 41-51, 1985.
[6] S. Chen, B. Shi, S. Chen, and Y. Xia, "ACOM: Capacity-Constrained Overlay Multicast in Non-DHT P2P Networks," *IEEE Trans. Parallel and Distributed Systems,* vol. 18, no. 9, pp. 1188-1201, Sept. 2007.
[7] C. Diot, B.N. Levine, B. Lyles, H. Kassem, and D. Balensiefen, "Deployment Issues for the IP Multicast Service and Architecture," *IEEE Network,* vol. 14, no. 1, pp. 78-88, Jan./Feb. 2000.
[8] A.J. Ganesh, A.M. Kermarrec, and L. Massoulié, "Peer-to-Peer Membership Management for Gossip-Based Protocols," *IEEE Trans. Computers,* vol. 52, no. 2, pp. 139-149, Feb. 2003.
[9] H. Hsiao, H. Liao, and C. Huang, "Resolving the Topology Mismatch Problem in Unstructured Peer-to-Peer Networks," *IEEE Trans. Parallel and Distributed Systems,* vol. 20, no. 11, pp. 1668-1681, Nov. 2009.
[10] K. Hwang, G. Fox, and J. Dongarra, *Distributed Systems and Cloud Computing,* to be published Morgan Kauffmann.
[11] X. Jin, G. Chan, W.-C. Wong, and A.C. Begen, "A Distributed Protocol to Serve Dynamic Groups for Peer-to-Peer Streaming," *IEEE Trans. Parallel and Distributed Systems,* vol. 21, no. 2, pp. 216-228, Feb. 2010.
[12] A.-M. Kermarrec, L. Massoulie, and A.J. Ganesh, "Probabilistic Reliable Dissemination in Large-Scale Systems," *IEEE Trans. Parallel and Distributed Systems,* vol. 14, no. 3, pp. 248-258, Mar. 2003.
[13] J. Leitao, J. Pereira, and L. Rodrigues, "Epidemic Broadcast Trees," *Proc. 26th IEEE Int'l Symp. Reliable Distributed Systems,* Oct. 2007.
[14] Z. Li, G. Xie, and Z. Li, "Towards Reliable and Efficient Data Dissemination in Heterogeneous Peer-to-Peer Systems," *Proc. IEEE Int'l Symp. Parallel and Distributed Processing,* Apr. 2008.
[15] N. Magharei and R. Rejaie, "PRIME: Peer-to-Peer Receiver-Driven MEsh-Based Streaming," *IEEE/ACM Trans. Networking,* vol. 17, no. 4, pp. 1052-1065, Aug. 2009.
[16] L. Massoulie, A.-M. Kermarrec, and A.J. Ganesh, "Network Awareness and Failure Resilience in Self-Organizing Overlay Networks," *Proc. IEEE Symp. Reliable Distributed Systems,* Oct. 2003.
[17] V. Pai, K. Kumar, K. Tamilmani, V. Sambamurthy, and A. Mohr, "Chainsaw: Eliminating Trees from Overlay Multicast," *Proc. Int'l Workshop P2P Systems,* Feb. 2005.
[18] S. Ratnasamy, M. Handley, R. Karp, and S. Shenker, "Topologically -Aware Overlay Construction and Server Selection," *Proc. IEEE INFOCOM,* June 2002.
[19] A. Rowstron and P. Druschel, "Pastry: Scalable, Decentralized Object Location and Routing for Large-Scale Peer-to-Peer Systems," *Proc. IFIP/ACM Int'l Conf. Distributed Systems Platforms (Middleware),* Nov. 2001.
[20] I. Stoica et al., "Chord: A Scalable Peer-to-Peer Lookup Service for Internet Applications," *ACM/IEEE Trans. Networking,* Feb. 2003.
[21] C. Tang, R.N. Chang, and C. Ward, "GoCast: Gossip-enhanced Overlay Multicast for Fast and Dependable Group Communication," *Proc. Int'l Conf. Dependable Systems and Networks,* June 2005.
[22] V. Venkataraman, K. Yoshida, and P. Francis, "Chunkyspread: Heterogeneous Unstructured Tree-Based Peer to Peer Multicast," *Proc. IEEE Int'l Conf. Network Protocols,* Nov. 2006.
[23] Z. Xu, C. Tang, and Z. Zhang, "Building Topology-Aware Overlays Using Global Soft-State," *Proc. Int'l Conf. Distributed Computing Systems,* May 2003.
[24] E.W. Zegura, K.L. Calvert, and S. Bhattacharjee, "How to Model an Internetwork," *Proc. IEEE INFOCOM,* Mar. 1996.
[25] Z. Zhang, S. Chen, Y. Ling, and R. Chow, "Capacity-Aware Multicast Algorithms in Heterogeneous Overlay Networks," *IEEE Trans. Parallel and Distributed Systems,* vol. 17, no. 2, pp. 135-147, Feb. 2006.
[26] X. Zhang, J. Liu, B. Li, and P. Yum, "DONet/Coolstreaming: A Data-Driven Overlay Network for Live Media Streaming," *Proc. IEEE INFOCOM,* Mar. 2005.

Zhenyu Li received the PhD degree from the Institute of Computing Technology (ICT), Chinese Academy of Sciences (CAS), in 2009, where he serves as an assistant researcher. His research interests include future Internet design, P2P systems, and online social networks. He is a member of the IEEE and the IEEE Computer Society.



Gaogang Xie received the PhD degree in computer science from Hunan University, in 2002. He is the professor at ICT/CAS. His research interests include future Internet architecture, programmable virtual router platform, Internet measurement, and modeling. He is a member of the IEEE and the IEEE Computer Society.

Kai Hwang received the PhD degree from the University of California, Berkeley in EECS. He is a professor at the University of Southern California. He is the founding editor-in-chief of the *Journal of Parallel and Distributed Computing*. His current research interests lie primarily in cloud computing and distributed systems. He was awarded IEEE Fellow for making significant contributions in computer architecture, digital arithmetic, and parallel processing.

Zhongcheng Li received the PhD degrees from ICT/CAS, in 1991, where he serves as a full professor. His research interests include next generation Internet, wireless communication. He is a member of the IEEE and the IEEE Computer Society.