FairOM: Enforcing Proportional Contributions among Peers in Internet-Scale Distributed Systems

Yijun Lu[1], Hong Jiang[1], and Dan Feng[2]

[1] Department of Computer Science and Engineering
University of Nebraska–Lincoln, Lincoln, NE 68588, USA
{yijlu,jiang}@cse.unl.edu
[2] Department of Computer Science and Engineering
Huazhong University of Science and Technology, Wuhan, 430074, China
dfeng@hust.edu.cn

Abstract. The viability of overlay multicasting has been established by previous research. However, in order to apply overlay multicast to Internet-scale distributed systems, such as the Grid and Peer-to-Peer systems, the issue of effectively enforcing fairness among peers so as to optimize overall performance remains as a challenge. This paper argues that simply applying a multiple-tree scheme does not provide sufficient fairness, in terms of performance. Instead, we believe that a better way to define fairness, for performance's sake, is to factor in peers' proportional contributions as it provides the opportunity to support many simultaneous multicasting sessions. This paper then presents a protocol, called FairOM (Fair Overlay Multicast), to enforce proportional contributions among peers in Internet-scale distributed systems. By exploiting the notion of staged spare capacity group and deploying a two-phase multicast forest construction process, FairOM enforces proportional contributions among peers, which enables more simultaneous multicasting sessions and alleviates potential hot-spots. The simulation results of a large multicast group with 1000 members show that FairOM achieves the goal of enforcing proportional contributions among peers and does not overwhelm the peers, including the multicast source. FairOM also achieves low delay penalty for peers and high path diversity.

1 Introduction

In Internet-scale distributed systems, such as the Grid and Peer-to-Peer (P2P) computing, reliable and efficient data dissemination plays a very important role, with examples ranging from the massive data delivery in data Grid to multimedia delivery in P2P environment. In these systems, overlay multicasting [4, 5, 11] is a better choice than IP level multicast for several reasons. First, overlay multicasting does not need the support from the network infrastructure. Second, it can be configured on top of the application level, thus providing opportunities to capture the semantics of the applications. And finally, it is easy to use and configure in practice.

The biggest challenge in applying overlay multicasting to an Internet-scale environment, such as the Grid and P2P environment, is to meet the peers' requirement of fairness [3], which stems from the equal status of peers in the distributed systems (in

Grid environment, different sites can be treated as equal status peers; in P2P environment, each node can be treated as a peer). In these environments, no one is supposed to contribute dramatically more or less than others.

The conventional single-multicast-tree structure does not satisfy the fairness requirement as the leaves in the tree have no contribution to the multicast effort while the interior nodes contribute all the forwarding bandwidth [3]. To tackle this problem, the notion of multicast forest, or multiple multicast-trees, has been explored in several studies [3, 9]. A good example of these systems is SplitStream [3], which builds a multicast forest and ensures that each peer only serves as an interior node once (as a contributor in one tree) on average and is a receiver in all other trees.

In this paper we revisit the issue of fairness requirement by asking the question of how to properly define fairness so as to increase overall performance. Even if we have a multicast forest in which each peer contributes some (by being an interior node in one multicast tree, for example) and no peer is overwhelmed, is there any chance that the multicast is still unfair in the sense that it results in relatively poor performance?

We argue that simply letting each peer contribute once and satisfying each peer's outgoing bandwidth constraint is not enough for enforcing fairness for the sake of performance. A better way to define fairness, we believe, is to enforce that peers' contributions are proportional to their total available outgoing bandwidths, which is analogous to taxation or donation. In taxation or donation, it is desirable for people to give the same percentage of their available capital as their contributions to the society (here, we assume all the people are in the same tax bracket).

Performance-wise, enforcing proportional contribution provides an environment to support multiple simultaneous multicasting sessions that may not otherwise be achievable by simply asking every peer to contribute *arbitrarily*. Consider the following example in which peers *A* and *B* are both going to multicast a movie and each multicast will span all the peers in the network. Suppose that *A* builds its multicast forest first and one peer, *C*, is assigned to contribute 90% of its outgoing bandwidth to it. Then when *B* tries to establish its multicast forest, chances are that *C* just does not have enough bandwidth to support it because it has contributed too much to the first multicast session. In this case, the construction of a forest for *B* becomes either infeasible or, barely feasible by saturating *C*'s outgoing bandwidth and making C a hot-spot/bottleneck. In this case, if we instead let each peer contribute roughly the same percentage of its outgoing bandwidth, say 20%, then *C* has a chance to support the two multicasting sessions simultaneously.

Moreover, as alluded to in the previous paragraph, enforcing proportional contribution among peers can reduce the probability of hot-spots. Using the same example, if multiple multicasting sessions are forced upon *C* when it barely has enough bandwidth, then *C* will become a hot-spot of the system and packets will be delayed, or worse yet, lost.

We present a protocol, called FairOM (Fair Overly Multicast), to enforce proportional contributions among peers through a two-phase forest construction process, with the assumption that all peers play by the rules. The case where peers may not be trustworthy is beyond the scope of this paper and will not be considered any further except for a brief discussion in Section 3.7.

The performance of FairOM is evaluated in a large size multicast group with 1,000 members through simulations. Simulation results demonstrate that FairOM achieves

the goal of enforcing proportional contributions among peers, does not overwhelm peers, including the source, has low delay penalty, and achieves high path diversity.

Before we move on to the next section, it is noteworthy that, in terms of bandwidth constraint, we only concern about the outgoing bandwidth for two reasons. First, current broadband technologies, such as ADSL, have limited outgoing bandwidth and larger incoming bandwidth. Second, each peer should have enough incoming bandwidth to accept all the stripes otherwise it cannot benefit from the multicast system.

We also recognize that, if the network bandwidth is so constrained that all the bandwidth is needed, there is no need to concern about the proportionality. However, with the proliferation of wireless-enabled laptops and high-speed Internet connections, we believe that there will be certain amount of excessive bandwidth available within Internet-scale distributed systems in the near future that should be effectively exploited to benefit the overall performance.

The rest of the paper is organized as follows. Section 2 formulates the problem and section 3 discusses the design of FairOM. The evaluation of FairOM is discussed in section 4 and related work is discussed in section 5. Finally, section 6 concludes this paper and discusses future work.

2 Problem Formulation

We represent each peer's total outgoing bandwidth as its total contribution capacity. Because of the design goal of minimizing the standard deviation of contribution proportions, we make the following three assumptions:

- Each data package to be multicast is encoded into *n* equal sized stripes and each peer has enough incoming bandwidth to absorb all the *n* stripes. This is essential to successfully build a multicast forest because otherwise the receiver cannot receive all the stripes no matter what multicast scheme is used.
- The total available outgoing bandwidth of peers is sufficient to build a forest to multicast data to the peers. Again, this assumption is to make the forest building feasible.
- There is excessive outgoing bandwidth in this multicast group. While this assumption is not essential to the correctness of the protocol, it provides the opportunity to show its advantages. If there is little excessive bandwidth left, all peers will have to contribute almost all their capacities, thus reducing to a special case of this protocol and making it identical or similar to other schemes.

Before we state our design goal, let us first formally define several terms with the assumption that there are a total of *n* peers in this multicast group.

- ***Ti:*** Total available outgoing bandwidth for each peer *i*, or, the maximum number of stripes it is capable of forwarding.
- ***Ci:*** The forwarding load of peer *i*, in term of the number of stripes it is assigned to handle.
- ***Ri:*** defined as *Ci/Ti*, is the contribution ratio of peer *i*.
- ***StdR:*** The standard deviation of the contribution ratios (R) of all the *n* peers. That is,

$$StdR = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(R_i - \overline{R})^2}$$

A complete multicast forest must satisfy the following two conditions:

- Multicast satisfaction: each peer should receive all the n stripes.
- Bandwidth limitation: the forwarding load of each peer i should be less than or equal to its total available outgoing bandwidth, or $Ci \leq Ti$.

The design goal is to minimize the standard deviation of all the peers' co ntribution ratio *StdR* in a complete multicast forest.

Goal: minimize *StdR*.

3 Design of FairOM

The basic idea of FairOM is to build a multicast forest in two phases. In the first phase, the peers join the multicast group and establish the neighborhood by a pair - wise neighborhood establishment procedure and use this neighborhood information to build an initial multicast forest that may not be complete. In the second phase, a peer contacts the source to ask for any missing stripes to make the forest complete.

FairOM assumes that a new peer knows at least one other member in the current multicast group when it joins, implying that FairOM does not directly deal with boo t- strap mechanism. Further, FairOM assumes that all the peers know when the forest construction starts and the number of trees they need to join. In practice, the source and the peers can exchange this information through web page announc ements or emails. As well, a peer can learn this information from its neighbors.

3.1 Establishment of Neighborhood

After joining the multicast group, a new peer will eventually establish its neighbor list by running a periodical neighborhood establishment procedure. In each round of this procedure, the peer contacts its neighbors (there is at least one bootstrap neighbor by assumption) and checks this neighbor's neighbor list. If its neighbor's neighbors do not appear in this peer's own neighbor list, it acts as follows. When its neighbor list is not full (each peer defines its length of neighbor list), it puts the new peers into its neighbor list. Otherwise, it compares the new peers with the ones a lready in its neighbor list according to the routing latency between the peers and itself. If a new peer has smaller latency, this peer replaces a current neighbor by the new one with a certain probability (currently we use 0.8) to prevent hot spot. If this peer adds a new peer to its neighbor list, it sends a notice to this new peer about this. While not imm e- diately clear here, the purpose of this operation will become obvious later when we discuss the staged quota relaxation next.

In this way, each peer will establish its ow n neighbor list after a certain number of rounds. After that, this periodical process servers as a way to adjust peers ' neighbor lists and maintain the neighborhood among peers.

3.2 Staged Spare Capacity Group

Staged spare capacity group is a key data structure in FairOM to enforce proportional contributions. Suppose that the spare capacity group has five stages, where each stage represents a percentage range of the capacity (e.g., stage 1 represents [0%, 20%], stage 2 (20, -40%], etc), then the source will put each of the registered peers into an appropriate stage. To illustrate this concept, we consider a simple example as illustrated in Figure 1.

In Figure 1, suppose that peer *A* has a total outgoing bandwidth of 20 (i.e., it can forward 20 stripes of data) and has already contributed 3 units of the total, then its current contribution is 15% (3/20). Because *A*'s contribution is less or equal to 20%, it is put into stage 1. *B* is put into stage 2 because its contribution is in the range (20%, 40%]. Follow the same criteria, *C* and *D* are put in stage 1 and 5, respectively.

It is worth noting that the source maintains an independent staged capacity group for each stripe. So if a peer has contribution for more than one stripe, it needs to register the contribution information for each stripe independently.

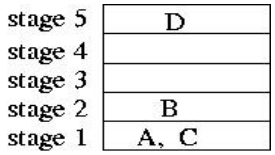

Figure 1. Layout of the staged spare capacity group for A, B, C and D while the contributions of them are 15%, 25%, 10% and 82%, respectively.

3.3 Phase I: Initial Forest Construction

Now we illustrate the first phase of the multicast forest construction among all the peers. The purpose of the initial forest construction is by no means to build a complete forest. Instead, it servers as a good start and provides a skeleton on which the second phase can improve. Because this is a quota-driven system, the system has a predefined initial quota. Each peer is willing to contribute as much as it can within this predefined quota.

More specifically, the source first sends all the stripes out, which are then forwarded to different neighbors to achieve path diversity. For each peer that receives a stripe, it forwards the stripe to as many neighbors as it can within the predefined quota. If a peer receives multiple transmissions of the same stripe, it picks one and rejects others. At this stage, let us assume that a peer picks the parent that notices it first.

When a multicast relationship between a parent and a child has been established (the parent picks the child and the child accepts it), the parent and the child both record this relationship locally. Then both of them start to run a heartbeat checking procedure to detect any failure.

For each peer, when it receives a stripe and has gotten all the responses from the children candidates it picked, it calculates its contribution and registers it to the staged spare capacity group by sending a message to the multicast source.

There is no clear line between the first phase and the second one in a global scale. Instead, it is each peer's own decision on when to start the second phase that we describe next.

3.4 Phase II: Making the Forest Complete

After the forest building process starts, each peer checks with those peers that treat it as a neighbor (recall that a peer sends a notice to its neighbor after the neighbor is added to its neighbor list in the establishment of neighborhood procedure). If all peers it contacts have already gotten some stripes and did not choose it as a child in the initial forest construction, it will seek help from the source. Moreover, a peer contacts the source anyway if a predefined deadline has passed.

In the message it sends to the source, the peer indicates the number of times it has requested for spare capacity, starting with 1 (the first time). When the source receives the message (with number 1), it only looks for parents for this peer in the first stage of the spare capacity group by randomly picking one eligible parent which has this stripe. Then it calculates what the new contribution for the parent would be. If the new contribution ratio is beyond the quota limit of this stage (20% for the first stage), the parent's record is moved from the current stage to the next higher one (stage 2 in this case).

If a parent is found, the parent will receive the adoption request of a potential child from the source and then send a request to the potential child that needs to be adopted. Thus the peer with missing stripes can get what it wants.

If the source cannot find a parent in this stage, the peer with missing stripes waits a predefined period of time before it starts the next round of request. When the next round starts, the peer sends another message that has a request number of 2, which tells the source to search for an adoption up to stage 2. However, the source still starts to look for adoption from the first stage in a hope to find some new spare capacity from smaller contributors. Thus each round provides opportunities for the source to find a peer with smaller contribution ratio to adopt. By following this protocol, the source relaxes the quota gradually and finally builds a complete forest in which every peer is in all trees. A pseudo code of this process is illustrated in Figure 2.

We need to mention that the delay between each round of requests for spare capacity and the way the source looks for adoption (always starts from the first stages, and climbs to higher stages gradually) are essential to the effectiveness of FairOM to enforce proportional contributions because they provide an opportunity to utilize resource from newly joined smaller contributors.

```
Peer:
  If (current > deadline && has not received all stripes) {
      num = 1;
      while (num_of_try <= 5) {
           send (source,id of all missing stripe, num);
           wait(waiting_time);
           if( still has missing stripes) {
               num++;
           } else {
               return success;
      }}
      return fail;
  }
Source:
  While (1) {
    recv(peer, ids of missing strips, num);
    foreach stripe in missing stripes {
        // find adoption from stage <= num;
        parent = find_adoption_up_to(num);
        if (parent != NULL) {
          send(parent, peer, id of missing strip);
          if (parent.contribution > limit of the current stage)
              move parent to the next higher stage
  }}}
```

Figure 2. **Pseudo code for making the forest complete**

3.5 Incorporating Multicast Delay Information into Consideration

The algorithm discussed so far does not consider multicast delay when it performs the forest construction. In this section, we try to minimize the multicast delay of the forest by incorporating the delay information into each of the two phases of the forest construction. Here, the delay perceived by a peer in regard to a particular stripe is defined as the time delay for it to receive the stripe from the source. Performance-wise, the shorter the delay, the better the performance is perceived by this peer.

In the initial forest construction process, each peer sends its delay information along with the message it sends to its neighbors. When a peer receives multiple transmissions of the same stripe, it picks the one with the smallest delay and drops others. Because the dropping process is based on delay, it will not create cycles as proved informally by following example.

Consider three peers *A*, *B* and *C*. Suppose that Peer *A* chooses *B* as parent, *B* chooses *C* as parent, and *C* chooses *A* as parent. In this case, the delay D_a of *A* is larger than delay D_b of *B*, that is, $D_a > D_b$. Similarly, we have $D_b > D_c$ and $D_c > D_a$, implying that $D_a > D_a$, which is impossible. Thus the scheme to incorporate delay information is cycle free.

In the second phase of the forest construction, when a peer requests missing stripes from the source, the source chooses several parents for this child (we current use three parents) and the child chooses the parent that gives it the smallest delay.

3.6 Handling Peer Join and Departure

When a peer joins the multicast group after the forest has been built, it first establishes its neighbor list by following the neighborhood establishment procedure. Then it seeks for adoption from its neighbors. Upon receiving this request, a neighbor grants the request if this adoption will still keep itself in the same stage in the spare capacity group. Otherwise, it rejects this request. If no neighbor is willing to adopt the new peer, it contacts the source for spare capacity.

In the case of peer departure, we differentiate two kinds of departures: decent departures and failures. For a decent departure, the departing peer notifies its multicast parents and children so that the parents can reclaim their contribution and its children can start seeking for adoptions by first contacting their neighbors and then contacting the source if none of their neighbors grant their requests. In the case of a failure, this failure will eventually be discovered by the heartbeat checking procedure which is run between each pair of multicast parent and child. After the failure is discovered, the failed peer's parents and children can react accordingly, similar to the case of a decent departure.

3.7 Discussions

Stress Put on the Source: In FairOM, the source is leveraged to manage the spare capacity group. One concern of this is the stress put on the source. We have considered two possible solutions to reduce such stress, but neither of them is completely satisfactory.

The first solution uses a source pool that includes several servers to share and distribute the burden of the source. While this scheme is simple, it assumes that the other peers in the pool are as trustable and stable as the source, which is not usually the case in a dynamic environment such as P2P. The second solution assigns several peers the same responsibility and uses Byzantine protocol to make the final decision and prevent cheating. However, this design complicates the system substantially.

The Security Issue: With the assumption that the peers are trustworthy, FairOM performs well and finally builds a fair-sharing multicast forest. However, when some peers do not work within the rule and cheat on their contributions, the multicast forest would not be fair any more. To prevent this from happening, distributed audit mechanism, as proposed by Ngan et. al [8], can be deployed to detect cheating and remove the peer that cheats from the multicast group.

4 Performance Evaluation

In order to best evaluate the system performance, we choose the Transit-Stub model [10] to simulate a physical network. The Transit-Stub model generates 1452 routers that are arranged hierarchically, like the current Internet structure. Then we generate 1,000 end nodes and attach them to routers randomly with uniform distribution. Further, our simulator models peers' bandwidths by assigning each peer a number that refers to the maximum number of stripes it can forward, which serves as the peer's

total outgoing bandwidth. For all the simulations, each peer's total outgoing bandwidth is randomly chosen between 10 and 20.

4.1 Effectiveness of Enforcing Proportional Contributions

We measure the effectiveness of enforcing proportional contribution by *StdR*, the standard deviation of the peers' contributions. In this simulation, we run three configurations with numbers of stripes being 2, 4 and 8, respectively.

In all the simulations, the algorithm satisfies the requirement to build a complete forest and satisfies all peers' bandwidth constraints. Then the mean value and *StdR* are calculated and summarized in Table 1. This result clearly shows that FairOM performs very well when we change the number of stripes from 2 to 8.

Table 1. Mean and Std of contribution ratios

Statistics	FairOM (2)	FairOM (4)	FairOM (8)
Mean	0.131	0.257	0.521
StdR	0.047	0.090	0.106

4.2 Improvement on the feasibility for multiple simultaneous multicasting sessions

In this section, we qualitatively argue the effectiveness of FairOM in terms of improving the feasibility for multiple simultaneous multicasting sessions as follows.

Because we assume that each multicast session needs to span the whole network, the feasibility of scheduling multicasting session is determined by the weakest peer in the network. Here, the weakest peer refers to the peer that has the least available outgoing bandwidth when a new multicasting session starts. When its available outgoing bandwidth is not enough to support one stripe, scheduling of that multicasting session becomes infeasible.

We consider two schemes, the SplitStream-like systems, and FairOM. In SplitStream-like systems, when the weakest peer contributes more in one session, it can render the scheduling for additional sessions infeasible. While in FairOM, it enforces that when each new multicast session is scheduled, the contribution ratio of each peer increase proportionally, which implies that the more powerful peers will contribute more to support the multicast, thus leaving enough bandwidth at the small peers' side for them to support more simultaneous sessions.

4.3 Forest Construction Overhead

To evaluate the overhead of the forest construction in FairOM, we use the number of messages received by each peer during the forest construction phase. In the relevant literature, this metric is also denoted as "node stress".

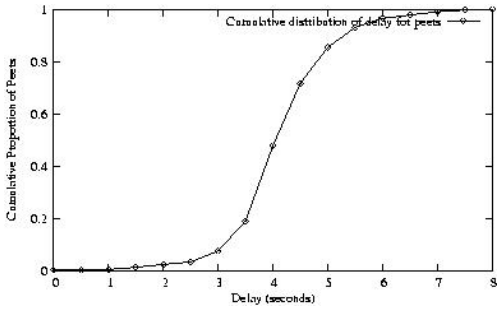

Figure 3. Cumulative distribution of delay for peers

In a typical run, all the peers, except the source, have node stress less than 300. The node stress for the source is 6585. While 6585 appears extremely high compared with other nodes' stress, it is amortized during the whole forest construction and does not induce much bandwidth cost for the source as shown in the following analysis.

In that particular run, the time of the forest construction was 192 seconds. Let us conservatively assume that each packet is of size 1KB, which is much larger than is really needed for the purpose of forest construction according to our experience. These 6585 messages amount to a total size of 6.6MB and receiving these packets in 192 seconds requires a bandwidth of 34.4KB. There is even less data sent out from the source during the forest construction phase (because certain messages, such as spare capacity registration, do not require response but account for a large portion of total messages received by the source). So this should not be a burden for a media server that usually has high-speed Internet connection.

4.4. Multicast Performance

Figure 3 shows the cumulative distribution of peers' delay in a typical run with 4 stripes. In this figure, a point (x, y) indicates that a fraction y of all the peers have delay less than or equal to x. In this simulation, all peers receive all the stripes within 8 seconds and the average delay is 4.1 seconds. This clearly shows the effectiveness of incorporating multicast delay information into the forest construction process.

4.5. Path Diversity of the Multicast Forest

Path diversity refers to the diversity between the paths from each node to the multicast source. Ideally, the paths should be disjoint with each other so that one peer's failure only causes the loss of one stripe for the receiver. While FairOM mainly uses randomization to achieve diversity, the enforced delay between quota relaxation requests also contributes to the path diversity because it provides opportunities for a receiver to get stripes from different parents.

In the simulation, we randomly fail one peer (not source, of course) in the multicast group. We run the simulation with two configurations. First, we run FairOM with four stripes and then FairOM with eight stripes. The result is shown in Table 2. This result validates the effectiveness of the path diversity of FairOM, which successfully builds such a forest where one peer's failure only costs the loss of a small number of stripes.

Table 2. Max, mean and median # of lost stripes with a single node failure

Statistics	FairOM (4)	FairOM (8)
Max	2	3
Mean	1.02	1.66
Median	1	1

5. Related Work

Seminal work such as Overcast [5] and End System Multicast (ESM) [4] builds a single multicast tree for a multicast source. While these systems proved the feasibility and validity of overlay multicast, the asymmetric nature of tree implies that a single-tree approach can not satisfy P2P's requirement of fairness as the leaves in a tree have no contribution to the multicast transmissions at all.

CoopNet [7, 9] uses a centralized mechanism to build multiple trees. To enforce fairness, CoopNet uses randomization in the tree construction process. There are two main differences between CoopNet and FairOM. First, FairOM utilizes both decentralized initial forest construction and centralized forest improvement while CoopNet is based on a purely centralized algorithm. Second, there is no mechanism in CoopNet to enforce proportional contributions among peers.

SplitStream [3] builds a multicast forest in which each peer only serves as an interior node once, and serves as leaves in all other trees, so is a fair system in the sense that each peer contributes once and only once. The main difference between FairOM and SplitStream lies in the way fairness is defined. Instead of defining fairness by letting each peer that contributes share certain forwarding load, FairOM defines fairness as peers' contributions being proportional to their total available bandwidths. SplitStream also has the concept of spare capacity and uses it as a backup mechanism to build a complete forest; however, the spare capacity group in FairOM is staged and plays a central role to enforce proportional contribution.

Bullet [6] is a representative of mesh based multicast protocol, which builds an overlay mesh to disseminate data. Comparing with a single-tree based multicast, Bullet has the benefit of removing forwarding bottleneck, which helps achieve high bandwidth. The philosophy behind Bullet is to exploit excessive bandwidth while the primary design goal of FairOM is to enforce fair contribution among peers.

Recently, Bharambe et. al [2] present the impact of heterogeneous bandwidth to DHT-based multicast protocols, such as Scribe, the origin of SplitStream. However, their work is based on DHT-based multicast while ours on an unstructured multicast.

6. Conclusion and Future Work

This paper presents the design and evaluation of FairOM, a fair overlay multicasting scheme for Internet-scale distributed systems. Through a two-phase forest construction process, FairOM enforces proportional contribution among peers. Simulation results show that FairOM achieves this design goal and puts low node stress to all the peers. Furthermore, it achieves path diversity which makes it robust to node failure.

In the future, we plan to investigate mechanism to reduce the source's heavy duty and investigate the security issues. Finally, we plan to make this forest adaptive to the dynamic changes of bandwidth after it is initially built.

Acknowledgements

This work is partially supported by the National Basic Research Program of China (973 Program) under Grant No. 2004CB318201.

References

[1] Planet-Lab, http://www.planet-lab.org

[2] R. Bharambe, S. G. Rao, V. N.Padmanabhan, S. Seshan, and H. Zhang. The impact of heterogeneous bandwidth constraints on DHT-based multicast protocols, In IPTPS'05, 2005

[3] M. Castro, P. Druschel, A.-M. Kermarrec, A. Nandi, A. Rowstron, and A. Singh. Splitstream: High-bandwidth multicast in cooperative environment. In *Proc. of the SOSP*, Bolton Landing, New York, USA, October 2003.

[4] Y.-H. Chu, S. G. Rao, S. Seshan, and H. Zhang. A case for end system multicast. IEEE Journal on Selected Areas in Communication (JSAC), Special Issue on Networking Support for Multicast, 20(8), 2002.

[5] J. Jannotti, D. Gifford, K. Johnson, and M. Kaashoek. Overcast: Reliable multicasting with an overlay network. In *Proc. OSDI*, San Diego, CA, 2000.

[6] D. Kostic, A. Rodriguez, J. Albrecht, and A. Vahdat. Bullet: Hight bandwidth data dissemination using an overlay mesh. In *Proc. of the ACM Symposium on Operating System Principles (SOSP)*, October 2003.

[7] V. N. Padmanabhan, H. J. Wang, and P. A. Chou. Supporting heterogeneity and congestion control in peer-to-peer multicast streaming. In *IPTPS*, 2004.

[8] T.-W. J. Ngan, D. S. Wallach, and P. Druschel. Enforcing fair sharing of peer-to-peer resources. In Proc of IPTPS, 2003

[9] V. N. Padmanabhan, H. J. Wang, P. A. Chou, and K. Sripanidkulchai. Distributing streaming media content using cooperative networking. In *NOSSDAV'02*, Miami, Florida, USA, May 2002.

[10] E. Zegura, K. Calvert, and S. Bhattacharjee. How to model an inter network. In *INFOCOMM*, San Francisco, California, 1996.

[11] S. Zhuang, B. Zhao, A. Joseph, R. Katz, and J. Kubiatowicz. Bayeux: An architecture for scalable and fault-tolerant wide-area data dissemination. In *NOSSDAV'2001*, June 2001.