Mutualcast: An Efficient Mechanism for One-To-Many Content Distribution

Jin Li, Philip A. Chou and Cha Zhang,
Microsoft Research, Communication and Collaboration Systems Group
One Microsoft Way, Bld. 113, Redmond, WA 98052
Email: {jinl, pachou, chazhang} @microsoft.com

ABSTRACT

In this paper, we propose Mutualcast, a new delivery mechanism for content distribution in peer-to-peer (P2P) networks. Compared with prior one-to-many content distribution approaches, Mutualcast achieves full utilization of the upload bandwidths of the peer nodes, thereby maximizing the delivery throughput. Mutualcast splits the to-be-distributed content into many small blocks, so that the more resourceful nodes may redistribute more blocks, and the less resourceful nodes may redistribute fewer blocks. Each content block is assigned to a single node for distribution, which can be a content-requesting peer node, a non-content-requesting peer node, or even the source node. The throughput of the distribution is controlled by redistribution queues between the source and the peer nodes. Furthermore, Mutualcast can be reliable and synchronous. Thus, it can be applied to file/software downloading, media streaming, real-time audio/video conferencing, etc.

Categories and Subject Descriptors

C.2.4 [Distributed Systems]: Distributed Applications; C.2.5 [Local and Wide-Area Networks]: Internet.

General Terms

Algorithms, Design, Performance

Keywords

One-to-many content distribution, file distribution, software distribution, peer-to-peer networks.

1. INTRODUCTION

A number of applications such as software distribution, Internet TV/video streaming, video conferencing, multiplayer gaming, personal media distribution, and P2P web content duplication distribute content from one source node to many destination nodes (one-to-many content distribution). A network-level solution to efficiently distribute the content is IP Multicast [1], where a single packet transmitted from the source is duplicated at routers along a distribution tree rooted at the source node, and is thereby delivered to an arbitrary number of receivers. Though IP multicast is an efficient solution, its deployment is slow in the real world because of issues such as inter-domain routing protocols, ISP business models (charging models), congestion control along the distribution tree, security and so forth. As a result, the vast majority of traffic in the Internet today is unicast based, where two computers directly talk to each other.

Since IP multicast is not generally feasible, various approaches have been developed to let peer computers, instead of routers, distribute the content from the source. The general approach is application-level multicast (ALM) [4], where a multicast distribution tree is formed and overlaid on the existing network. Instead of using IP multicast, each peer computer in the distribution tree implements all multicast related functionalities including packet replication, membership management and content delivery on the overlaid network. Some sample systems include Scattercast [2] and Overcast [3], both of which use a single tree to distribute the content. Compared with letting the source node directly send its content to all other clients, the distribution tree approach reduces the network load of the source, thus achieving more efficient content distribution. In a distribution tree, the intermediate nodes redistribute the content, while the leaf nodes only receive the content. The upload bandwidths of the leaf nodes are not utilized for content distribution. CoopNet [5] and SplitStream [6] overcome this inefficiency by splitting the content into multiple stripes and distributing the stripes across separate multicast trees with disjoint interior nodes. Any peer computer could be an interior node in one of the multicast trees, and contribute to forwarding the content. FastReplica [17] and Bullet [18] investigate the issue of efficient and reliable replication of large files. If there are *n* nodes, FastReplica first partitions the file into *n* subfiles of equal size. Each subfile is then transferred to a different peer in the group, which is subsequently replicated and transferred to the other peers. In Bullet, peer nodes are organized into an overlay tree. Each node splits the content received from the parent into a disjoint set of blocks, with each set sent to a different child node. The child nodes then discover the missing blocks and the nodes that hold the missing blocks, and send requests to recover the missing blocks. A related scheme, using erasure coded blocks, is proposed in [19]. A practical P2P system has been implemented by BitTorrent [20] with sharing incentive so that each pair of peers roughly sends and receives an equal amount of content. These are just a few examples of the many recent schemes for application-level multicast.

Although the above ALM distribution strategies are more efficient than directly sending content from the source to the peers, they fail to achieve maximally efficient content distribution in the network. None of the above schemes adequately considers the differences in bandwidth between the peer nodes. They also fail to fully engage the bandwidth resources of all the peer nodes to distribute the content.

Furthermore, the above ALM distribution strategies are either asynchronous or unreliable. Systems such as CoopNet and SplitStream, which are used for synchronous delivery of content (e.g., streaming audio or video), while *resilient*, are *unreliable* in the sense that they cannot guarantee delivery of every byte. Con-

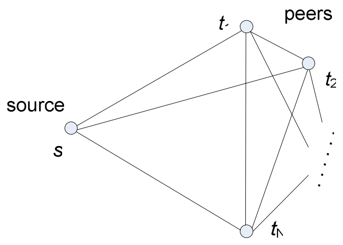


Figure 1 One-to-many content distribution.

versely, systems such as FastReplica, Bullet, and BitTorrent, which are used for reliable file distribution, are *asynchronous* in the sense that content may arrive at each receiver at a different time and at a different rate. Although there are prior multicast solutions that are both reliable and synchronous (as typified by SRM [21], RMTP [22], and the like [23]), in these prior solutions reliability and synchrony are achieved, ultimately, by slowing down the transmission to the rate of the lowest bandwidth link, which is a very inefficient solution in a heterogeneous environment.

In this work, we propose Mutualcast, which is a maximally bandwidth-efficient mechanism for content distribution in the P2P network, and which furthermore is synchronous and can be reliable. In contrast to existing approaches, Mutualcast achieves the maximum possible throughput for its content by 1) engaging as many nodes as possible to distribute the content and 2) fully utilizing the nodes' available upload bandwidths. Mutualcast also adjusts the content sending rate dynamically to match the maximum throughput under the prevailing network conditions. Mutualcast is simple and flexible, with three distinct features. First, it splits the to-be-distributed content, be it a file or media stream, into many small blocks, and distributes each block separately. The blocks are pipelined for delivery, allowing for overall synchronous communication. Second, in a Mutualcast group, each block of content is assigned to a single peer node for redelivery. The assignment and redelivery can be done reliably, allowing for overall reliable communication. Third and most importantly, Mutualcast employs an optimal bandwidth allocation strategy, which is implemented via redistribution queues between the source and the peer nodes. The redistribution queues accommodate the bandwidth differences between the peer nodes, and ensure maximum delivery throughput even if there are packet loss and transmission jitter on network links. This ensures maximum efficiency. We have implemented Mutualcast for file/software distribution, e.g., distribution of software patches/update. Nevertheless, the Mutualcast protocol can be used to distribute other content, such as streaming media or multimedia collections.

2. BACKGROUND AND PRIOR WORKS

The one-to-many content distribution problem that we are considering is illustrated in Figure 1. The network consists of a source node s, which holds the content to be distributed, and multiple peer nodes $t_i, i = 1,2,\cdots,N$, each of which may or may not request a copy of the content. The number of peer nodes N is on the order of tens. Both the source node and the peer nodes are computers connected to the Internet through an internet service provider (ISP), using an ADSL, cable modem, campus, or corporate network link. We do not consider the case that the nodes are infrastructure nodes, such as the routers, on the backbone of the Internet. Our target is to distribute the content with maximum throughput to all the destinations.

The simplest approach for a source node to distribute content in this setting is to let the source node send the content directly to the destination nodes. Though straightforward, the throughput of the content distribution is bounded by the upload bandwidth of the source node, which is usually fairly limited. Naturally, we want to enlist the help of the peer nodes, and use their upload bandwidths to aid the content distribution.

Application-level multicast (ALM) [4] has attracted a lot of interest recently. A few ALM systems, such as NICE [8], Scribe [9], Bayeux [10] and CAN-multicast [11], are designed for large-scale applications. Among them, NICE has its own node management mechanism, while Scribe, Bayeux and CAN-multicast are implemented on structured peer-to-peer overlay networks such as Pastry [7], Tapestry [12] and CAN [13]. The main objective of these large-scale systems is to reduce/balance the link and node stress while limiting the delay and duplication [16]. On the other hand, when the system scale is relatively small (as shown in Figure 1), maintaining the group status is relatively simple. The focus is thus to manage the multicast routing such that the system is reliable, efficient and has low end-to-end delay.

We next examine a few prior small-scale ALM approaches that are directly related to our Mutualcast framework. Scattercast [2] and Overcast [3] each form a single distribution tree. A sample distribution tree is shown in Figure 2(a). In this configuration, the source node sends the data to node t_1, which forwards the data to nodes t_2 and t_3. The ALM distribution tree utilizes the upload bandwidth of the intermediate node t_1, whereas the upload bandwidths of the leaf nodes t_2 and t_3 are not utilized. To better utilize the bandwidths of the peer nodes, nodes with higher upload bandwidths should be placed upstream, while nodes with lower upload bandwidths should be placed downstream.

Both CoopNet [5] and SplitStream [6] stripe the content and distribute the stripes using separate multicast trees with disjoint interior nodes. CoopNet uses a centralized tree management scheme, while SplitStream relies on Pastry [7] to maintain the distribution tree. CoopNet further utilizes multiple description coding (MDC) and forward error correction (FEC) to protect from packet loss and node failure. The CoopNet/SplitStream configuration with two application-level multicast trees is illustrated in Figure 2(b). The content is divided into two equal stripes. The first stripe is sent to node t_1, which forwards the stripe to nodes t_2 and t_3. The second stripe is sent to node t_2, which forwards the strip to nodes t_1 and t_3. We notice that the system utilizes the upload bandwidths of nodes t_1 and t_2, but fails to utilize the upload bandwidth of node t_3. The systems are resilient but not reliable.

FastReplica [17] is specifically designed for file download. For an N node P2P network, FastReplica distributes the file with N height-2 multicast trees with intermediate degree N-1. A sample FastReplica configuration of three peer nodes is illustrated in Figure 2(c). FastReplica distributes the file in two steps: the distribution step and the collection step. In the distribution step, the file is split into three subfiles and sent to nodes t_1, t_2 and t_3 (along solid, dashed, and dotted lines), respectively. After the distribution step, the collection step kicks in. Each peer node forwards its subfile to the other peer nodes. All peer nodes are engaged in content distribution in FastReplica. FastReplica is reliable, but

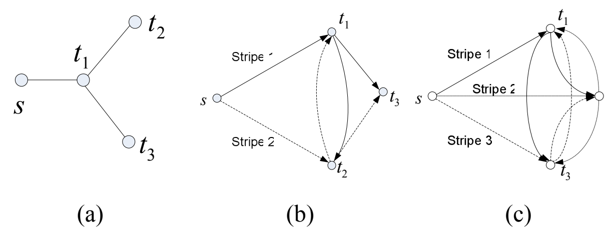


Figure 2 Prior one-to-many content distribution schemes: a) Scattercast/Overcast, b) SplitStream/CoopNet, c) FastReplica.

because the distribution is split into two stages, it cannot be applied to synchronous applications such as streaming media.

All of the above one-to-many content distribution approaches adapt to the capabilities of the peer nodes, i.e., their upload/download bandwidths, by establishing a suitable network topology. Nodes with high bandwidth are placed in the center of the distribution network, and are in charge of more content distribution. Once the network topology is established, it is often fixed throughout the session. Such distribution strategy leaves the distribution network less flexible to adapt to changes in the network conditions, e.g., congestion of certain nodes/links.

An alternative approach is to constantly update the network topology by measuring the network characteristics during the session. For instance, in End System Multicast [14], a protocol called Narada is developed to construct an overlay structure among participating end systems in a self-organizing and fully-distributed manner. End systems gather information of network path characteristics using passive monitoring and active measurements. Narada continually refines the overlay structure as more network information is available. ALMI [15] is another protocol that can adapt its structure to the network condition. Each ALMI session has a session controller which constructs a shared tree. The tree is periodically re-calculated based on the end-to-end measurements collected by session members. Adapting the network topology to network conditions makes Narada and ALMI robust to network condition variations. Unfortunately, both Narada and ALMI may suffer from inefficiency or turbulence during the adjustment of the network topology.

3. MUTUALCAST DISTRIBUTION

3.1 Framework

Mutualcast differs from the previous one-to-many content distribution approaches in that it uses a fixed network topology, but adapts by letting peer nodes with different capabilities distribute different amount of content. Mutualcast is simple to implement, with several distinct features. First, Mutualcast splits the to-be-distributed content, be it a file or a media stream, into many small blocks. The number of blocks redistributed by a certain node can thus be proportional to the resource (upload bandwidth) of the node. The node with larger upload bandwidth may redistribute more blocks, and the node with smaller upload bandwidth may redistribute fewer blocks. Second, in a Mutualcast group, each block of content is assigned to a single node for redelivery. The node in charge of the redelivery can be a content-requesting peer node, a non-content-requesting peer node, or even the source node itself. Third, employing redistribution queues between the nodes, Mutualcast can effectively deal with dynamic changes in the network condition, and copes with variations in the upload

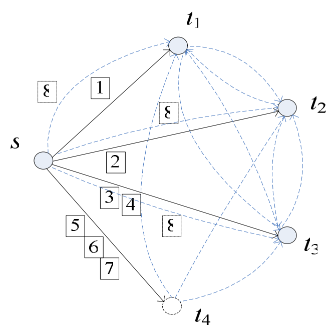


Figure 3 Mutualcast content distribution network: an example.

bandwidth, packet loss and packet jitter on an ongoing basis. An additional feature is that the network topology of Mutualcast is generic and fixed. Mutualcast relies on bandwidth reallocation to adapt to the network condition variations, which is preferred to adapting the network topology itself.

The basic distribution framework of Mutualcast is as follows. The content being distributed is chopped into blocks B_j, j=1,2,$\cdots$,M. For each block B_j, one unique node is selected to distribute the block to the rest of the peer nodes. Frequently, the node in charge of redistributing the block B_j is a peer node t_i. In such a case, the source node sends one copy of the block B_j to the peer node t_i, which then redistributes the block B_j by sending a copy of the block to the rest of the peer nodes. However, when the source node has abundant bandwidth resources, the node in charge of distributing the block B_j can be the source node s itself. In that case, the source node will directly send one copy of block B_j to each peer node t_i.

We show an example Mutualcast distribution network in Figure 3. In this network, there are one source node s and four peer nodes t_1, t_2, t_3 and t_4. Among the peer nodes, the nodes t_1, t_2 and t_3 request a copy of the content from the source node s. The node t_4 does not request a copy of the content. Nevertheless, it contributes its upload bandwidth to help distributing the content to the other peer nodes. When the block is assigned to the content-receiving peer nodes t_1, t_2 and t_3 for redistribution, such as the blocks 1, 2, 3 and 4, the block is first sent by the source node to the peer node in charge, which then forwards the block to the other two peer nodes. When the block is assigned to a non-content-receiving peer node t_4 for redistribution, such as the blocks 5, 6 and 7, the block is first sent by the source node to the peer node t_4, which forwards the block to the other three peer nodes. The source node may also choose to directly distribute the block, such as the block 8. In that case, the block is sent directly from the source node to the peer nodes t_1, t_2 and t_3.

Mutualcast chops the content into a large number of small blocks for distribution. The size of the Mutualcast block is a compromise between the granularity of distribution and the overhead required for identifying the block. During the implementation, it is preferable that the size of the Mutualcast block is a little bit less than the maximum transmission unit (MTU) of the network, so that each Mutualcast block can be sent as a single packet over the network. In the current work, we set the block size as 1KB.

3.2 Mutualcast: distribution routes

Mutualcast assigns each block to a certain node for redistribution. The number of blocks assigned to the peer node is proportional to its capacity, which is evaluated by its upload bandwidth. The reason is the following. In terms of the contribution of a peer node to the network, it is the upload bandwidth of the peer node that counts. Thus, to efficiently distribute content in a Mutualcast network, we should make use of the upload bandwidths of the peer nodes as much as possible. We also notice that for a file distribution session, the primary parameter that governs the speed of the distribution is the throughput of the network link. If a client can choose multiple servers to serve it the file, it should choose the server that provides the fastest network throughput between the two. The other network parameters, such as round trip time (RTT), packet loss ratio, network jitter, are less relevant. In a networks composed of the end-user nodes, we may characterize the network by assigning an upload bandwidth constraint on each node, a download bandwidth constraint on each node, and a link bandwidth constraint between any two nodes or any two group of nodes. However, the bottleneck is usually the upload bandwidths of the nodes. This is because in Mutualcast, a peer node sends content to multiple destinations. The output of the peer node thus splits among multiple receivers. As a result, the link bandwidth required between the two peer nodes is only a fraction of the upload bandwidth of the sending node, which usually does not become the bottleneck. The required download bandwidth for a node to receive the content is always less than the total available upload bandwidths of all the nodes in the network divided by the total number of receiving nodes. In increasingly common networks, such as cable modem and ADSL networks, the total upload bandwidths of the end-user nodes are much smaller than the total download bandwidths. Even for user nodes on the campus networks or the corporate networks, the download bandwidth can still be much larger than the available upload bandwidth because the user may cap the upload bandwidth to limit participation in the P2P activity. In the following discussion, we will assume that the receiving nodes have enough download and link bandwidths to receive content from the Mutualcast. We will briefly discuss nodes of limiting download and link bandwidths in Section 3.7.

Table 1 Link bandwidth and download bandwidth requirement for Mutualcast network of Figure 3.

Receiving node	Sending node, and Link Bandwidths					Download Bandwidth
	s	t_1	t_2	t_3	t_4	
t_1	0.83B	-	0.5B	B	B	3.33B
t_2	0.83B	0.5B	-	B	B	3.33B
t_3	1.33B	0.5B	0.5B	-	B	3.33B
t_4	B	-	-	-	-	B
Upload BW	$4B$	B	B	$2B$	$3B$	

In the example of Figure 3, let us assume that the upload bandwidths of the peer nodes t_1 and t_2 are B; that of the peer node t_3 is $2B$; that of the peer node t_4 is $3B$; and that of the source node is $4B$, where B is a unit of bandwidth. An optimal strategy of fully utilizing the upload bandwidths of the source and peer nodes is shown in Table 1. We will discuss the bandwidth allocation problem in the next section.

If the Mutualcast group includes a source node, N_1 content-requesting peer nodes (N_1>1 as otherwise the problem is trivial) and N_2 non-content-requesting (but willing to participate) peer nodes, the Mutualcast network will distribute the content through N_1 height-2 trees with intermediate degree N_1-1 (with the intermediate node being one of the content-requesting nodes), N_2 height-2 trees with intermediate degree N_1 (with the intermediate node being one of the non-content-requesting nodes), and one height-1 tree with degree N_1, all rooted at the source node. The network topology employed by Mutualcast bears some resemblance to the FastReplica scheme of [17]. Nevertheless, there are a number of distinct features of Mutualcast. First, Mutualcast does not separate the distribution and the collection steps. Instead, the content blocks are distributed continuously by the source and the peer nodes. Second, in Mutualcast, the amount of content being redistributed by a particular peer is not fixed, but varies according to the capabilities (the upload bandwidths) of the peer nodes. Finally, Mutualcast may involve the source node and non-content-requesting peer nodes in the redistribution of content.

The Mutualcast network distributes the content through three routes: 1) through content-requesting peer nodes, 2) through non-content-requesting peer nodes, and 3) directly from the source node. Each distribution method demands different amounts of network resource from the participating nodes. Again, the network resource of chief concern is the upload bandwidth consumed. To distribute a portion of content having bandwidth B in a Mutualcast network of N_1 content-requesting peer nodes, the first distribution route demands upload bandwidth B from the source node, and upload bandwidth (N_1-1)B from each content-requesting peer node. The second distribution route demands upload bandwidth B from the source node, and upload bandwidth $N_1 \cdot B$ from each non-content-requesting peer node. The third distribution route demands upload bandwidth $N_1 \cdot B$ from the source node. In the first and the second routes, Mutualcast uses the upload bandwidths of the peer nodes (including the content-requesting peer nodes and the non-content-requesting peer nodes) to alleviate the upload bandwidth burden on the source node. This has the effect of speeding up the maximum rate of content distribution.

It is interesting to notice that for the same route, the amount of network resource consumed is independent of the individual upload bandwidth of each peer node. Thus we may consider the bandwidth allocation problem with respect to each route category instead of each peer node.

3.3 Mutualcast: bandwidth allocation

In the Mutualcast network, the most precious resource is the upload bandwidth of the source node, where the content originates. If the upload bandwidth of the source node is used up, we cannot further speed up content distribution, even if there are still peer nodes with available upload bandwidths. It is apparent that if the source node sends content blocks at rate B through the delivery links to all N_1 content-requesting peer nodes, it will consume $N_1 \cdot B$ of the upload bandwidth of the source. On the other hand, if the source node sends content blocks at rate B to a peer node t_i, which in turn distributes the blocks to the rest of the content-requesting peer nodes, only an amount B of the upload bandwidth of the source node is needed. Apparently, as long as there are more than one content-requesting peer nodes, the source node should forward as many content blocks as possible to the peer nodes for redelivery. Between the content-requesting and non-content-requesting peer nodes, the content-requesting peer nodes have a slight edge in efficiency, as the content blocks sent to the nodes in the forward links are not wasted. As a result, among the three distribution routes outlined above, the most preferred route is route 1, followed by the route 2. Only when the source node still has upload bandwidth left, it may choose route 3 to distribute content directly to the peer nodes.

We assume that the Mutualcast network consists of a source node of upload bandwidth B_s, N_1 (N_1>1) content-requesting peer nodes with average bandwidth B_1, and N_2 non-content-requesting peer nodes with average bandwidth B_2. Applying the distribution route selection strategy above, the distribution throughput of the Mutualcast network, which is defined as the amount of content sent to the content-requesting peer nodes per second is:

$$\theta = \begin{cases} B_s & B_s \le B_{s1} + B_{s2}, \\ (B_{s1} + B_{s2}) + \dfrac{B_s - (B_{s1} + B_{s2})}{N_1} & B_s \ge B_{s1} + B_{s2}, \end{cases} \qquad (1)$$

with

$$B_{s1} = \frac{N_1}{N_1 - 1} B_1 \quad \text{and} \quad B_{s2} = \frac{N_2}{N_1} B_2 .$$

This shows that before the upload bandwidths of all the peer nodes have been exhausted, the distribution throughput is limited only by the upload bandwidth of the source node. All N_1 content-requesting peer nodes receive content at the rate of the upload bandwidth of the source node. After the upload bandwidths of all the peer nodes have been exhausted, the distribution throughput becomes $(1/N_1)^{th}$ of the sum of the upload bandwidths of the network (N_1B_1+N_2B_2+ B_s) minus a small portion (N_2B_2/N_1) wasted in the distribution through non-content-requesting peer nodes.

3.4 Mutualcast: distribution route selection through redistribution queue

With the priority outlined in Section 3.3, if we know the available upload bandwidths of the source and all the peer nodes, we may explicitly calculate the bandwidth allocated between any two peer nodes, and distribute content blocks accordingly. However, there is an even simpler method that works in a distributed fashion. We may use a queue to estimate the bandwidth on any connection link, and govern the selection of the distribution routes of the content blocks based on the status of the queues, thus achieving implicit bandwidth allocation without knowing the bandwidths of the network.

The key idea is to establish a queue to buffer content being sent from one node to another, and to use the queue to control the speed of distribution between any two nodes. In our implementation of Mutualcast, the links between nodes are established via TCP connections. The redistribution queues are thus simply the TCP send and receive buffers. An additional advantage of using TCP is that the flow control, reliable data delivery and node leave events are all automatically handled by TCP. Reliable data delivery in Mutualcast is inherited through these reliable TCP connections. Congestion control in Mutualcast is likewise inherited.

We call the TCP connection carrying blocks to be redistributed the *forward* link, and the TCP connection that carries blocks not to be further redistributed the *delivery* link. We establish one TCP connection (the delivery link) from each peer node to every other content-requesting peer node. We establish one TCP connection (the forward link) from the source node to every non-content-requesting peer node, and two TCP connections (the forward and

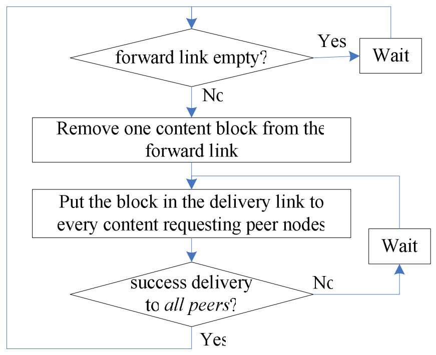

forward link empty?
Yes
Wait
No
Remove one content block from the forward link
Put the block in the delivery link to every content requesting peer nodes
Wait
success delivery to all peers?
No
Yes

Figure 4 The forward link thread of the peer node.

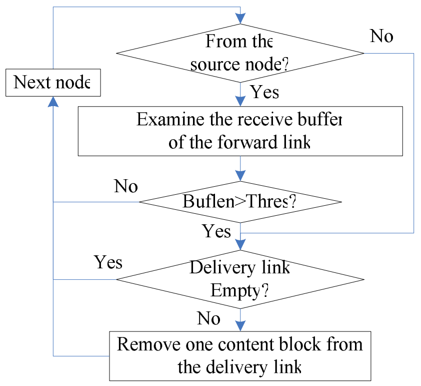

From the source node?
No
Next node
Yes
Examine the receive buffer of the forward link
No
Buflen>Thres?
Yes
Yes
Delivery link Empty?
No
Remove one content block from the delivery link

Figure 5 The delivery link thread of the peer node.

the delivery links) from the source node to every content-requesting peer nodes. The selection of the distribution routes becomes finding available slots in the TCP connections.

Let us now examine the workflow of the Mutualcast source and peer nodes. Each content-requesting Mutualcast peer node consists of two threads, where one thread receives the content blocks from the delivery link, while the other thread receives the content blocks from the forward link and redistributes them to the rest of the content-requesting peer nodes through their delivery links. For the non-content-requesting peer nodes, only the forward link thread is operated.

The operational flow of the forward link thread of a Mutualcast peer node (both content-requesting and non-content-requesting) is shown in Figure 4. In each iteration loop of the forward link thread, the peer node removes one content block from the incoming forward link, and copies the block onto the outgoing delivery links of all the other content-requesting peer nodes. The thread does not remove another content block from the incoming forward link until it has successfully copied the last content block onto all the delivery links. That way, if the outgoing delivery links are blocked, possibly resulted from reaching the limit on the upload bandwidth of the peer node, the peer node will stop removing the content blocks from the incoming forward link, thus effectively regulate the receiving rate of the forward link to be $1/M^{th}$ of the upload bandwidth of the peer node, where M is the number of nodes that the content block is redistributed to, which is N_1-1 for content-requesting peer node and N_1 for non-content-requesting peer node.

The operational flow of the delivery link thread of the content-receiving peer node is shown in Figure 5. For the content blocks arriving on delivery links from nodes other than the source node, the operation is simply to remove the content blocks from the link as soon as they arrive. For content blocks arriving on the delivery link from the source node, we put an additional constraint that we only remove content blocks from the delivery link when the receiving buffer length[1] of the forward link from the same source node is above a certain threshold. The rationale is that the delivery link and the forward link are two separate TCP connections sharing the same network path from the source to the peer node. The content blocks sent through the forward link have higher priority, as they are to be redelivered to the other content receiving peers. The receiving buffer length policy guarantees that the bandwidth of the forward link to be at least $1/M^{th}$ of the upload bandwidth before the delivery link from the source node to the peer node is activated.

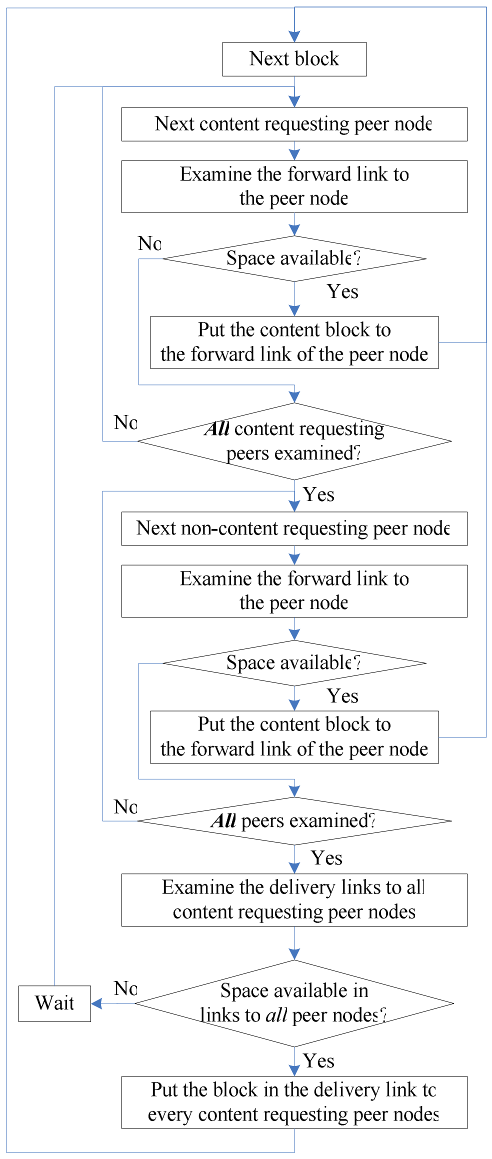
The operational flow of a Mutualcast source node is shown in Figure 6. For each content block, the source node selects one of the distribution routes based on the status of the redistribution queue. The route selection is based on the following order of priorities. The redistribution by a content-requesting peer node has the highest priority. The redistribution by a non-content-requesting peer node has the second highest priority. The distribution directly from the source node to all the content-requesting peer nodes has the lowest priority.

As shown in Figure 6, the source node first checks if there is space available for the content block in any TCP connection of the forward link from the source node to the content-requesting peer node. If the send buffer of one of the TCP connections is not full, the content block is put into that TCP buffer to be sent to the corresponding content-requesting peer node, which then redistributes the content block to the other content-requesting peer nodes through the corresponding delivery links. If no space on the forward links to the content-requesting peer nodes can be found, the source node checks the forward links to the non-content-requesting peer nodes. If space is found available on a link, the content block is put into the TCP buffer for the corresponding link. If there is still no space available even on the links to the non-content-requesting peer nodes, the source node pursues the final distribution route, and checks if there is space for one block available in all the delivery links to all the content-requesting peer nodes. Combined with the receiving buffer length policy in Figure 5, this ensures that the bandwidth of the forward link does not get squeezed by the traffic of the forward link, If space is found, the content block is replicated and put into the delivery link to each content-requesting peer node. If there is no space on any of the distribution routes, the source node will wait for a short amount of time before it will retry to find an available route for the content block again.

3.5 Operational analysis of Mutualcast: role of the redistribution queue

Using redistribution queues and the above operational strategy for the peer and source nodes, Mutualcast handles anomalies such

[1] In socket programming, the receiving buffer length may be obtained through ioctl() function call with parameter FIONREAD.



Next block
Next content requesting peer node
Examine the forward link to the peer node
No
Space available?
Yes
Put the content block to the forward link of the peer node
No
All content requesting peers examined?
Yes
Next non-content requesting peer node
Examine the forward link to the peer node
Space available?
Yes
Put the content block to the forward link of the peer node
No
All peers examined?
Yes
Examine the delivery links to all content requesting peer nodes
Wait
No
Space available in links to *all* peer nodes?
Yes
Put the block in the delivery link to every content requesting peer nodes

Figure 6 The operation flow of the source node.

as packet loss and network congestion during content distribution by adjusting the upload bandwidths of the nodes to achieve the maximum content distribution throughput by fully utilizing the upload bandwidth resources of the source and peer nodes. We explain the optimality of the Mutualcast in the following.

The content blocks between any two nodes are distributed through a redistribution queue, which in our current implementation is a TCP connection with a certain size sending and receiving buffer. We notice from Section 3.4 that the Mutualcast source and peer nodes push as many content blocks as possible into the TCP connections, until the TCP sending buffer is full. The content blocks that are pending in the sending buffers of the TCP connections ensure that the network paths between any two peer nodes are fully utilized, even considering network anomalies such as packet loss and network congestion. If there are no packet losses, new content blocks will be sent to the destination peer nodes through the TCP connections. If there are packet losses or other network anomalies, TCP will try to recover from the network errors through retransmissions, and the content blocks that are pending in the TCP sending buffers will not be sent out. The content blocks that are pending in the TCP receiving buffer of the forward link ensures that the upload bandwidth of the corresponding peer node is fully utilized. After the peer node pushes the last content block into the TCP sending buffer of the delivery links, it can retrieve the content block pending in the TCP receiving buffer, thus continue the activity of pushing blocks into the delivery links, and not wasting the upload bandwidth.

In addition, the operational flows of Figure 4-6 ensure that the upload bandwidths of the source node and peer nodes are fully utilized, and the content distribution routes are selected in favor of the distribution through content-requesting peer nodes, then the distribution through non content-requesting peer nodes, and finally the direct distribution from the source node.

When we use Mutualcast to distribute content to N_1 content-requesting peer nodes, if the upload bandwidth of the source node is low and the delivery links from the source to the peer nodes are not activated, then the content distribution throughput of Mutualcast will be the upload bandwidth B_s of the source node. In this case, the content is sent out of the source node at rate B_s, where the peer nodes have sufficient upload bandwidth to send content to all content-requesting peer nodes. Each content-requesting peer node is receiving content at the rate of B_s, as if the source node is only sending the content to it alone. If the upload bandwidth of the source node is high, and the delivery links from the source to the content-requesting peer nodes are activated, then the content distribution throughput of Mutualcast will be the sum of the upload bandwidths of the source and peer nodes, minus a small portion of bandwidth wasted by sending content blocks to the non-content-requesting peers for redelivery, all divided by the number N_1 of content-requesting nodes. As a result, Mutualcast achieves the maximum content distribution throughput calculated in equation (1), no matter what the network resource (upload bandwidth) configuration of the network is. Mutualcast also easily adapts to the changes in network bandwidth through the redistribution queues of the TCP links. If a certain peer node slows down, the content blocks in its delivery links will move slowly, prompting the peer node to retrieve fewer content blocks from its forward link. This in turn causes the source node to send fewer content blocks to this now slowed down peer node, and to redirect the content blocks to other faster peer nodes. Alternatively, if a certain peer node speeds up, Mutualcast can likewise adjust by sending more content blocks to it.

3.6 Theoretical analysis of Mutualcast: maximizing content distribution throughput

In this section we prove that Mutualcast is optimal for peer-to-peer networks with constrained upload bandwidths. Mutualcast achieves the maximum possible throughput in such networks; no other system can do better.

Let the graph (V,E) represent the network, with V being the set of nodes and E being the set of links (directed edges). Let s in V denote the source node and let T denote the subset in E of content-requesting nodes. Let the remaining nodes be non-content-requesting nodes. Consider two types of capacities. Let $c(e)$ be

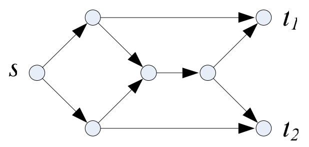


Figure 7 Edges have unit capacity. Broadcast capacity is two units. Multicast can achieve only one unit of throughput.

the capacity of each edge e in E, and let $c_{out}(v)$ represent the upload bandwidth (output capacity) of each node v in V, such that for each node v, the sum of the capacities of the edges leaving v is at most $c_{out}(v)$.

A *cut* between two nodes v_1, v_2 in V is a partition of V into two sets V_1, V_2 such that v_i is in V_i, i=1,2. The *value* of the cut is the sum of the capacities $c(e)$ on the edges e from V_1 to V_2.

It is well known that the maximum flow between s and any sink t in T achieves the minimum value over all cuts between s and t. Let C_t be the value of the maximum flow (the maxflow) between s and t. Note that $C_t = C_t(c)$ depends on the edge capacity function c:$E\rightarrow[0,\infty)$.

Definition. The *broadcast capacity* between s and T is the minimum maxflow between s and any t in T, that is, $C = \min_t C_t$. Note that like C_t, $C = C(c)$ depends on the edge capacity function c.

Clearly, the broadcast capacity C is an upper bound on the maximum rate at which common information can be broadcast from s to all nodes in T. Unfortunately, C is not achievable in general using multicast routing, as the example in Figure 7 illustrates. Although C can always be achieved using network coding [25], network coding requires the intermediate nodes to *code*, not merely *route*, their input packets to produce output packets. If only routing is used, the maximum throughput C_0 from s to T via multiple multicast trees can be a factor of log N lower than C [26]. Moreover, determining the optimal collection of multicast trees (achieving C_0) is NP-hard, while the tightest known bound on the gap between C_0 and the throughput $C_{00} \le C_0$ achievable in polynomial time is relatively loose [27]. On the other hand, if there are no Steiner nodes in the network (a Steiner node is a node v for which $C_v < C$) then the broadcast capacity C can be simply achieved by greedily packing multiple multicast trees, as implied by Edmonds' theorem [28].

Mutualcast, which is a particularly structured collection of multiple multicast trees, achieves the broadcast capacity $C = C(c)$ for some edge capacity function $c(e)$. Furthermore, it achieves the maximum such broadcast capacity, as the following theorem shows.

Theorem. The Mutualcast throughput θ achieves the maximum possible broadcast capacity subject to the node output capacity constraints, that is, $\theta = \max_c C(c)$ over all edge capacity functions c:$E\rightarrow[0,\infty)$ such that for all nodes v, the sum of $c(e)$ over all edges e leaving v is at most $c_{out}(v)$.

Proof. We have separate proofs for networks in which $B_s \le B_{s1} + B_{s2}$ and networks in which $B_s \ge B_{s1} + B_{s2}$. We prove the former with a cut separating s from V–s and we prove the latter with cuts separating V–t from t.

First assume $B_s \le B_{s1} + B_{s2}$. For any edge capacity function c, the broadcast capacity $C(c)$ can be at most equal to the value of the cut separating s from $V–s$. Since this is at most $B_s \equiv c_{out}(s)$, we have $\max_c C(c) \le B_s$. Of course, a throughput θ must satisfy $\theta \le \max_c C(c)$. On the other hand, according to (1), Mutualcast achieves throughput $\theta = B_s$. Hence $\theta = \max_c C(c) = B_s$.

Now assume $B_s \ge B_{s1} + B_{s2}$. For any edge capacity function c, the sum of $c(e)$ over all edges entering nodes in T must be at least N_1 times the broadcast capacity $C(c)$. Thus we have (denoting $U = V–T–s$ as the set of non-content-receiving nodes):

$$\begin{aligned} N_1 C(c) &\le \sum_{t\in T}\sum_{e\in In(t)} c(e) \\ &= \sum_{v\in V}\sum_{e\in In(v)} c(e) - \sum_{u\in U}\sum_{e\in In(u)} c(e) \\ &= \sum_{v\in V}\sum_{e\in Out(v)} c(e) - \sum_{u\in U}\sum_{e\in In(u)} c(e) \\ &\le \sum_{v\in V} c_{out}(v) - \sum_{u\in U}\sum_{e\in In(u)} c(e). \end{aligned}$$

On the other hand, from (1) we have (denoting $B_v = c_{out}(v)$):

$$\theta = \frac{1}{N_1-1}\sum_{t\in T} B_t + \frac{1}{N_1}\sum_{u\in U} B_u + \frac{1}{N_1}\left(B_s - \frac{1}{N_1-1}\sum_{t\in T} B_t - \frac{1}{N_1}\sum_{u\in U} B_u \right).$$

Hence

$$\begin{aligned} N_1\theta &= \frac{N_1}{N_1-1}\sum_{t\in T} B_t + \sum_{u\in U} B_u + B_s - \frac{1}{N_1-1}\sum_{t\in T} B_t - \frac{1}{N_1}\sum_{u\in U} B_u \\ &= \sum_{t\in T} B_t + \sum_{u\in U} B_u + B_s - \frac{1}{N_1}\sum_{u\in U} B_u \\ &= \sum_{v\in V} c_{out}(v) - \sum_{u\in U}\frac{B_u}{N_1} \end{aligned}$$

Of course, $\theta \le \max_c C(c)$, so $N_1\ \theta \le N_1 \max_c C(c) = N_1 C(c^*)$, where c^* is an optimizing capacity function. Thus

$$\sum_{v\in V} c_{out}(v) - \sum_{u\in U}\frac{B_u}{N_1} = N_1\theta \le N_1 \max_x C(c) = \sum_{v\in V} c_{out}(v) - \sum_{u\in U}\sum_{e\in In(u)} c^*(e).$$

We are done if we can show that the inequality holds with equality. Certainly this is true if U is empty. To show this when U is not empty, we argue that for each u in U, $\frac{B_u}{N_1} \le \sum_{e\in In(u)} c^*(e).$ Otherwise, any flow through u to the N_1 content-receiving nodes would be insufficient to use up the upload bandwidth B_u, whence we could achieve a higher throughput by re-allocating some capacity from edges between s and T to edges between s and U. □

Corollary. In a file download scenario, Mutualcast minimizes the maximum download time experienced by any content-receiving peer node, and in a streaming media scenario, Mutualcast maximizes the minimum quality experienced by any content-receiving peer node.

Thus Mutualcast is ideal in situations where a distributed group of friends wishes to experience downloaded or streamed content at the same time with the same quality.

3.7 Mutualcast: throughput under download bandwidth or link bandwidth constraints

The above sections assume that the only bottleneck in Mutualcast is the upload bandwidths of the peer nodes. Here we give a brief discussion on the Mutualcast throughput under link bandwidth or download bandwidth constraints.

Consider a peer node i with upload bandwidth B^u_i. Let its link bandwidth to the content-receiving peer node j be B^l_{ij}, j=0,⋯,M-1, where M is the number of content-receiving nodes other than itself. The link bandwidth between node i and j will not be the bottleneck as long as: $B^l_{ij} \ge B^u_i / M$. If the above inequality is not satisfied, the upload bandwidth of node i cannot be fully utilized in the current Mutualcast scheme. The effective upload bandwidth of node i becomes: $B^u_i{}' = M \min_j B^l_{ij}$. This effective upload bandwidth can be used in equation (1) to obtain the new Mutualcast throughput.

When a content-receiving peer node has download bandwidth less than the throughput given in equation (1) (which is based only on the upload bandwidths), such a node will also be a bottleneck of Mutualcast. In such scenario, the overall Mutualcast throughput will be the minimum download bandwidth of all the content-receiving peer nodes. This is because all nodes have to wait for the slowest node to finish before they can resume delivery.

An alternative strategy to the current Mutualcast implementation is to let the slow peer nodes skip certain content blocks, so that they will not slow down the receiving operation of the remaining peer nodes, which can still proceed at full speed. In a file download scenario, the slow peer nodes may be able to receive the skipped content after all the remaining nodes have finished downloading. In a streaming media scenario, the slow peer nodes may be able to receive their content with lower quality, if layered media coding is used. In comparison to the alternative approach, the current Mutualcast implementation maximizes the throughput of common information to all content-receiving peer nodes. It maximizes the minimum quality experienced by any content-receiving peer node in a streaming media scenario, or minimizes the maximum download time experienced by any content-receiving peer node in a file download scenario (for example, if a distributed group of friends wishes to experience downloaded or streamed content at the same time with the same quality). We believe in most applications, this is the preferred solution.

4. EXPERIMENTAL RESULTS

A Mutualcast file distribution solution has been implemented. The solution includes a sender module run by the source node and a receiver module run by each of the peer nodes. To verify the performance of Mutualcast, we set up a Mutualcast content delivery network with one source node and four content-receiving peer nodes. Each node is attached with an upload bandwidth gag which may control its upload bandwidth. A media file of size 16MB is then broadcasted via Mutualcast from the source to all the peer nodes.

In the first experiment, we evaluate the overall effectiveness of the Mutualcast system. The upload bandwidths of the source and the content-receiving peer nodes are kept constant throughout the entire experimental session. We then compare the theoretical broadcast capacity versus the actual Mutualcast throughput, which is measured by dividing the size of the distributed content by the time of distribution. A total of four different experiments have been conducted with difference source and peer node bandwidths,

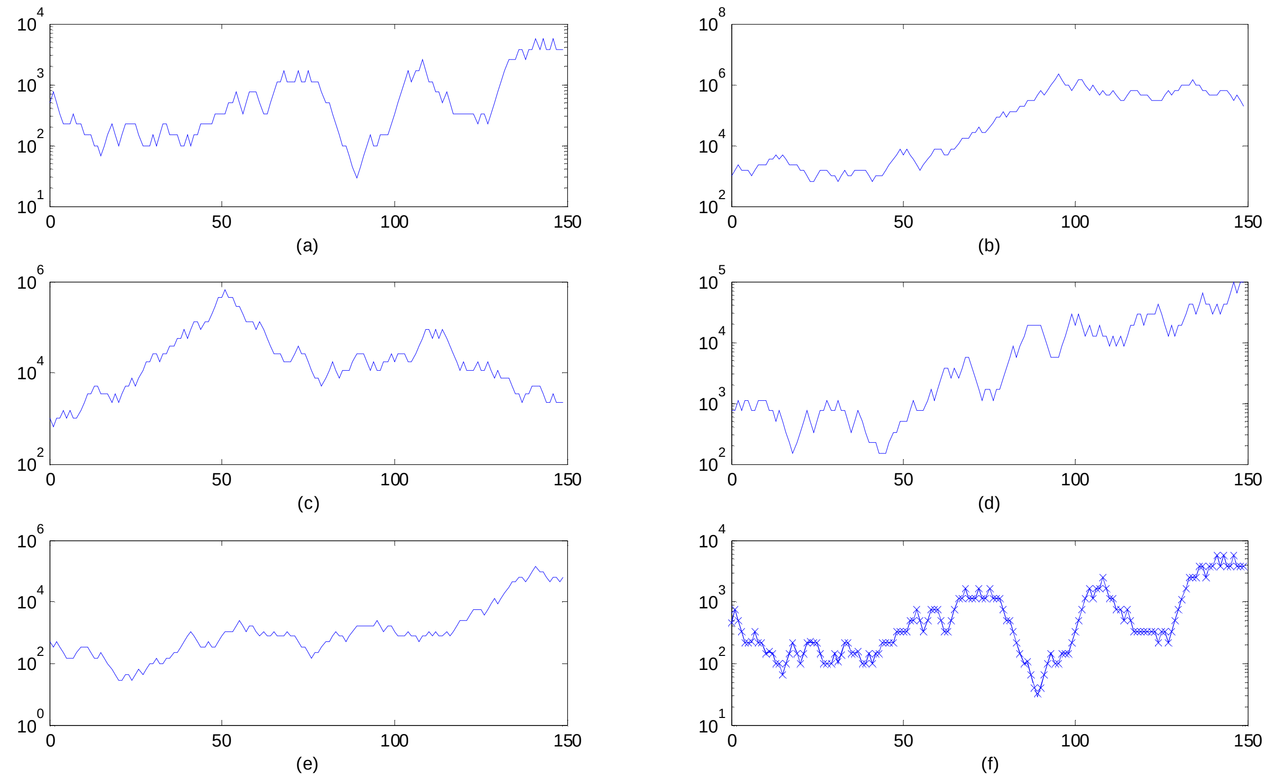


Figure 8 Mutualcast throughput analysis. The horizontal axis records the time (in seconds), and the vertical axis records the bandwidth (in kbps). The figures are: a) throughput of the source node *s*, b)-e) throughput of the content-receiving peer nodes t_1-t_4, and f) the analytical throughput (marked with a solid line) and the actual throughput (marked with an 'x').

and the results are shown in Table 2. With just the simple implementation of the Mutualcast sender and receiver components according to Figure 4-6, the actual Mutualcast throughput is remarkably close to the analytical broadcast capacity of the peer-to-peer network.

Table 2 The throughput of content distribution: the analytical broadcast capacity vs. actual Mutualcast throughput

No.	Upload Bandwidths (kbps)					Throughput (kbps)	
	S	t_1	t_2	t_3	t_4	Analytical	Mutualcast
1	500	1000	1000	750	500	500	500.08
2	1000	1000	1000	750	500	1000	999.43
3	500	250	1000	750	500	500	499.95
4	1000	750	1000	750	500	1000	1001.2

In the second experiment, we further demonstrate the capability and the flexibility of the Mutualcast network in response to network change. In the experiment, the upload bandwidths of the source and the peer nodes may randomly adjust every second with one of three modes: 50% upward, 33% downward, or constant, all with a 1/3 probability. The instantaneous upload bandwidths of the source node and the four content-receiving peer nodes are shown in Figure 8 a)-e), where the horizontal axes denote elapsed time in seconds, and the vertical axes denote the upload bandwidths in kbps. In Figure 8f), we again compare the theoretical broadcast capacity (shown with a solid line) versus the actual Mutualcast throughput (shown with an 'x'), which is measured by counting the number of bytes delivered every second. The horizontal axis is again the timeline, and the vertical axis is the throughput in kbps. It is clearly demonstrated that the Mutualcast not only can achieve the analytical broadcast capacity, but also can achieve the capacity under complex and varying network conditions. By using the TCP sending and receiving buffer as a redistribution queue, Mutualcast can adapt to changing network conditions by flexibly assigning more content blocks to nodes experiencing better network conditions at the time. It achieves the broadcast capacity throughout the content distribution session.

5. CONCLUSIONS

A simple yet flexible content distribution approach called Mutualcast is developed in the paper. Mutualcast splits the to-be-distributed content into many small blocks, and assigns the distribution of each content block to a single node, which can be a content-requesting node, a non-content-requesting node or the source node. Nodes with more upload bandwidth can distribute more blocks, and nodes with less upload bandwidth can distribute fewer blocks. TCP connections with their sending and receiving buffers are used by Mutualcast to control the throughput of the distribution, and ensure that the upload bandwidths of the all the peer nodes and source node are fully utilized even with network anomalies such as packet losses and delivery jitters. Though simple, Mutualcast achieves the broadcast capacity of the peer-to-peer network.

6. REFERENCES

[1] "Internet protocol multicast", ch 43 of Internetworking Technology Handbook, http://www.cisco.com/univercd/cc/td/doc/cisintwk/ito_doc/index.htm.

[2] Y. Chawathe, "Scattercast: an architecture for internet broadcast distribution as an infrastructure service". *PhD thesis*, University of California, Berkeley, August 2000.

[3] J. Jannotti, D. K. Gifford, K. L. Johnson, M. F. Kaashoek, and J. W. O'Toole Jr., "Overcast: reliable multicasting with an overlay network". In *Proc. of the Fourth Symposium on Operating System Design and Implementation (OSDI)*, October 2000.

[4] Y. Hua Chu, S. Rao, H. Zhang, "A case for end system multicast", In *Proc. of Sigmetrics 2000, Santa Clara, CA, June 2000.*

[5] V. N. Padmanabhan and K. Sripanidkulchai, "The Case for Cooperative Networking", In *Proc. of the First International Workshop on Peer-to-Peer Systems (IPTPS)*, Cambridge, MA, USA, March 2002.

[6] M. Castro, P. Druschel, A-M. Kermarrec, A. Nandi, A. Rowstron and A. Singh, "SplitStream: High-bandwidth content distribution in a cooperative environment", In *Proc. of the International Workshop on Peer-to-Peer Systems*, Berkeley, CA, February, 2003.

[7] A. Rowstron and P. Druschel, "Pastry: scalable, distributed object location and routing for large-scale peer-to-peer systems", In *Proc. of IFIP/ACM International Conference on Distributed Systems Platforms (Middleware)*, Heidelberg, Germany, pages 329-350, November, 2001.

[8] B. Banerjee, B. Bhattacharjee and C. Kommareddy, "Scalable application layer multicast", *Proc. SIGCOMM'02*, 2002.

[9] M. Castro, P. Druschel, A. M. Kermarrec and A. Rowstron, "SCRIBE: a large-scale and decentralized application-level multicast infrastructure", *IEEE Journal on Selected Areas in Communications*, pp.100-110, Vol. 20, No. 8, 2002.

[10] S. Q. Zhuang, B. Y. Zhao, A. D. Joseph, R. H. Katz and J. Kubiatowicz, "Bayeux: an architecture for scalable and fault-tolerant wide-area data dissemination", *Proc. of NOSSDAV'01*, 2001.

[11] S. Ratnasamy, M. Handley, R. Karp and S. Shenker, "Application-level multicast using content-addressable networks", *Proc. of NGC'01*, 2001.

[12] B. Zhao, J. Kubiatowicz, and A. Joseph, "Tapestry: An infrastructure for fault-resilient wide-area location and routing," *U. C. Berkeley, Tech. Rep.* UCB//CSD-01-1141, April 2001.

[13] S. Ratnasamy, P. Francis, M. Handley, R. Karp, and S. Shenker, "A Scalable Content-Addressable Network," *Proc. of ACM SIGCOMM*, Aug. 2001.

[14] Y. Chu, S. Rao and H. Zhang, "A case for end system multicast", *Proc. of ACM SIGMETRICS'00*, 2000.

[15] D. Pendarakis, S. Shi, D. Verma and M. Waldvogel, "ALMI: an application level multicast infrastructure", *3rd USENIX Symposium on Internet Technologies*, 2001.

[16] M. Castro, M. Jones, A.-M. Kermarrec, A. Rowstron, M. Theimer, H. Wang, and A. Wolman, "An evaluation of scalable application-level multicast built using peer-to-peer overlay networks", *Proc. of INFOCOM'03*, 2003.

[17] L. Cherkasova and J. Lee, "FastReplica: Efficient Large File Distribution within Content Delivery Networks", In *Proc. of the 4-th USENIX Symposium on Internet Technologies and Systems*, Seattle, Washington, March 26-28, 2003.

[18] D. Kostic, A. Rodriguez, J. Albrecht, A. Vahdat, "Bullet: High Bandwidth Data Dissemination Using an Overlay Mesh", In *Proc. 19th ACM Symposium on Operating Systems Principles*, October 19-22, 2003, the Sagamore, New York.

[19] J. Byers, J. Considine, M. Mitzenmacher, and S. Rost, "Informed Content Delivery across Adaptive Overlay Networks," *Proc. of ACM SIGCOMM,* Aug. 2002.

[20] B. Cohen, "Incentives build robustness in BitTorrent", http://bitconjurer.org/BitTorrent/bittorrentecon.pdf

[21] S. Floyd, V. Jacobson, C. Liu, S. McCanne, and L. Zhang, "A Reliable Multicast Framework for Light-weight Sessions and Application Level Framing," *IEEE/ACM Trans. Networking,* Vol. 5, No. 6, pp. 784-803, Dec. 1997.

[22] S. Paul, K. K. Sabnani, J. C. Lin, and S. Bhattacharyya, "Reliable Multicast Transport Protocol (RMTP)," *IEEE J. Selected Areas in Communications,* Vol. 15, No. 3, pp. 407-421, Apr. 1997.

[23] K. Obraczka, "Multicast Transport Protocols: A Survey and Taxonomy," *IEEE Communications Magazine,* pp. 94-102, Jan. 1998.

[24] S. B. Wicker, V. K. Bhargava, *Reed-Solomon Codes and their Applications*, IEEE Press, New York, 1994.

[25] R. Alswede, N. Cai, S.-Y. R. Li, and R. W. Yeung, "Network information flow," *IEEE Trans. Information Theory,* Vol 46, No. 4, pp. 1204-1216, July 2000.

[26] S. Jaggi, P. Sanders, P. A. Chou, M. Effros, S. Egner, K. Jain, and L. Tolhuizen, "Polynomial time algorithms for network code construction," to appear in *IEEE Trans. Information Theory.*

[27] K. Jain, M. Mahdian, and M. R. Salavatipour, "Packing Steiner trees," *14th ACM-SIAM Symp. Discrete Algorithms (SODA),* 2003.

[28] J. Edmonds, "Edge-disjoint branchings," in *Combinatorial Algorithms,* R. Rustin, ed., pp. 91-96, Academic Press, NY, 1973.