

Cost-effective Broadcast for Fully Decentralized Peer-to-peer Networks

Marius Portmann ^{*}, ^{**} Aruna Seneviratne ^{*}

marius@ieee.org, a.seneviratne@unsw.edu.au

^{*} School of Electrical Engineering and
Telecommunications
University of New South Wales,
Sydney, Australia

^{**} Computer and Networks Laboratory
Swiss Federal Institute of Technology ETH
CH-8092, Zürich, Switzerland

Abstract

Recently, there has been a growing interest in peer-to-peer networks, sparked by the popularity of file sharing applications such as Napster and Gnutella. A typical characteristic of a peer-to-peer system is that all the nodes are equal participants in the network. Gnutella is an example of a 'pure' peer-to-peer system, being fully distributed where all nodes are equal and no special nodes with facilitating or administrative roles are required. Due to its fully decentralized nature, Gnutella implements its services like searching and peer discovery via application-level broadcast. For this, messages are routed through Gnutella's overlay network by means of flooding. The high cost of flooding limits the scalability of fully distributed peer-to-peer systems like Gnutella. To overcome the problem of scalability, we propose Rumor Mongering (also known as Gossip) as a cost-effective alternative to flooding to implement application layer broadcast in decentralized peer-to-peer networks. We introduce a new version of Rumor Mongering, called Deterministic Rumor Mongering, which makes use of the fact that most peer-to-peer network topologies display a power-law distribution in their node degree, to implement a more intelligent routing strategy. Using simulation, we show that Deterministic Rumor Mongering performs broadcast at a significantly lower cost than flooding, at the cost of a slightly reduced reliability and increased time of completion of the broadcast.

1.0 Introduction

Recently, there has been a lot of activity in the area of peer-to-peer networking, sparked by the popularity of file sharing applications such as Napster [1], Freenet [20] and Gnutella [2]. Although the exact meaning of the term “peer-to-peer” is very debatable, these systems typically depend on a number of voluntarily participating nodes contributing resources to create some form of infrastructure. The most well known service such a peer-to-peer infrastructure can provide is certainly multimedia file-sharing, but other systems with completely different applications such as SETI@home [3] are also labeled as peer-to-peer.

Even among the peer-to-peer systems aimed at file sharing, there exist big differences in terms of architecture and implementation of key mechanisms such as searching. Napster, for instance, relies on a central indexing server to locate files that are shared. Gnutella implements searching in a fully decentralized and distributed manner, without any special nodes with facilitating or administrative roles. In that sense, Gnutella can be considered a more “pure” peer-to-peer system. The advantages of a decentralized architecture are clear. The absence of a central node that represents a single point of failure, results in increased fault tolerance and robustness. Furthermore, the only state information a Gnutella node needs to store is the address of its immediate neighbors. This allows Gnutella to deal with the very dynamic nature of a typical peer-to-peer network, where nodes frequently join and leave the network. The problem of convergence due slow propagation of routing information is therefore avoided.

Gnutella provides two types of services via its overlay network: searching for files and peer discovery. These services are implemented through broadcasting messages across the network. This application level broadcast is implemented with flooding as the underlying routing mechanism, where every node acts a router and forwarder for messages sent by other nodes. This obviously raises the question of cost and scalability. Individual nodes can be swamped with messages to forward and might not have enough resources such as CPU cycles, memory or

bandwidth required for the task. This problem has also been observed in the real Gnutella system [4], where the fact that nodes could not keep up with the rate of messages to be forwarded, has lead to a fragmentation of the network. It is therefore important for the successful deployment of decentralized peer-to-peer networks that the problem of cost and scalability is studied and more cost-effective methods for application level broadcast are explored.

In this paper, we study the cost of application layer broadcast as it is implemented in Gnutella.

The general rules for routing messages in Gnutella are as follows:

Deleted: message

- ⌘ In general, a message received by a node is forwarded to all its neighbors, except for the one, where the message was received from.
- ⌘ Each message has a Time To Live (TTL) field which is decremented by one at each visited node. If the TTL reaches zero, the message is no longer forwarded.
- ⌘ Each message has an ID to uniquely identify it on the network. Every node keeps a record of IDs of messages that it has seen in the recent past. If a node receives a message with the same ID and type as one that it has received before¹, the message is ignored and not forwarded.

To study the cost of different broadcast routing mechanisms, we define a cost metric based on the number of messages that are generated and forwarded in the network. We derive the cost for flooding based broadcast solely on the network size and the average node degree, i.e. the average number of neighbors that a node has. Based on this, we estimate the average bandwidth consumption per node for different example scenarios. This shows that the resource consumption

¹ Although the Gnutella protocol specification in [8] does not say for how long the message IDs need to be cached by the nodes, it is assumed that it is at least a multiple of the typical live time of a message.

per node of flooding based broadcast can be prohibitively high, even for networks of moderate size, which provides the motivation to search for more cost-effective broadcast methods.

As an efficient alternative to flooding to implement application-level broadcast in a fully distributed manner, we propose the use of *Rumor Mongering* (also known as Gossip) as a routing method. Rumor Mongering trades off reliability and speed for a reduction in cost. First, we study the performance and cost of Rumor Mongering, which chooses the neighbors to forward messages to uniformly at random. Then we introduce a new variant of the protocol, which we call *Deterministic Rumor Mongering*. Deterministic Rumor Mongering makes a more intelligent decision about message forwarding by making use of the fact the topology of a typical peer-to-peer network displays a power-law distribution of the node degrees. By means of simulation, we show that Deterministic Rumor Mongering can significantly reduce the cost of application level broadcast compare to flooding.

The relevance of the work in this paper is not restricted to the area of file-sharing applications. The concept of Application-layer overlay networks is increasingly being proposed for a diverse range of applications [5][6][7]. An efficient broadcast method can be an important building block in these decentralized systems, especially for ad-hoc networks and other highly dynamic environments.

The rest of the paper is structured as follows. In section 2, we introduce our simulation platform and we discuss relevant parameters such as the metric of cost. Section 3 studies the cost of flooding-based broadcast, as it is implemented in Gnutella. In section 4 and 5, we evaluate two variants of Rumor Mongering as an alternative to flooding. Finally, section 6 concludes the paper.

2.0 Simulation – Platform and Parameters

2.1 Metric of Cost

In order to evaluate the cost of application level broadcast mechanisms we first need to define a metric of cost. An obvious choice for this is the average number of messages that need to be forwarded per node reached in a single broadcast. The number of messages forwarded by a node relates directly to the amount of resources such as bandwidth and CPU cycles that are consumed.

As our metric, we define the cost c as follows:

$$c = \frac{1}{r} \sum_{i=1}^N m_i$$

- r : number of nodes reached by a broadcast
- m_i : number of messages forwarded by node i
- N : number of nodes in the network

To illustrate the cost metric defined above, Figure 1 shows a Gnutella's broadcast mechanism for two simple topologies of three nodes. The first one is a simple spanning tree with only two bidirectional links. The second topology has the same number of nodes but has an additional link between node 2 and 3.

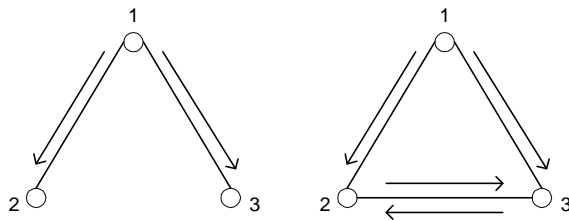


Figure 1: Routing Example to illustrate the Metric of Cost

In a first step, the initiator of the broadcast (node 1) sends messages to its immediate neighbors (2 and 3). In the case of the first topology, the broadcast is completed. The total number of messages generated is 2, one per node reached. Therefore, according to our metric, the cost is 1. In the second topology, additional messages are sent from node 2 to 3 and vice versa and the cost c is 2, twice as high as in the previous scenario.

The theoretical minimal cost for the type of broadcast considered here is one message per node to be reached, i.e. $c = 1$. This minimal cost can only be achieved for spanning tree topologies with no redundant links. It seems intuitively clear that the more redundant links a topology contains, the higher the cost of flooding based broadcasting.

2.2 The Simulator

To study the cost of application level broadcast mechanism, we built a discrete event simulator in Java. This simulator takes as input a graph file, describing the topology of the network to be simulated. Its modular design allows us to easily simulate different kinds of broadcast routing protocols.

The simulation is based on synchronous rounds. In each round, every node reads the messages from its input queue and handles them according to the specified routing rules. This means either to forward the message to one or several of its neighbors or to discard it. Each node keeps track of the number of messages it has forwarded during each round. A simulation of a broadcast is initiated by having one node in the network send a message to all its neighbors and the simulation ends when no more messages are forwarded per round. The results of the simulation may depend on which node was chosen to start the broadcast. We therefore perform a large number of simulation runs, with the node initiating the broadcast chosen at random. As the result we take the average over all the runs.

2.3 Topology

To establish the cost of application layer broadcast for peer-to-peer networks by simulation, we need to generate a network topology with the typical characteristics of peer-to-peer networks. One such characteristic that has been observed for peer-to-peer networks by measurement [9] and simulation [10], as for many other communication [11] and social networks [12], is a power-law distribution in their node degree². That means that a few nodes have a very high degree and many have a low degree. The number of nodes with a certain degree decreases with the degree according to the following power law: $f_d \propto d^{-k}$. The frequency f_d (number of nodes) of a degree d , is proportional to d to the power of a negative constant k .³

A model proposed by Barabási and Albert [13] allows to generate random topologies with a power-law characteristic. This model suggests two possible causes for the emergence of a power-law in the frequency of node degrees in network topologies: incremental growth and preferential connectivity. *Incremental growth* means that growing networks are formed by the continual addition of new nodes, and thus the gradual increase in the size of the network. *Preferential connectivity* refers to the tendency of a new node to connect to existing nodes that are highly connected or popular. Both of these are typical characteristics of peer-to-peer networks, which usually grow and evolve in such an ad-hoc fashion. We therefore believe that the Barabási-Albert model serves as a good basis to generate random topologies with the typical characteristics of peer-to-peer networks.

To generate the random topologies for our simulation, we used BRITE [14], a topology generator developed at Boston University. BRITE supports, among others, the Barabási-Albert model and allows exporting the graphs in a standard format, which can be read by our simulation tool. In the

² The degree of a node is defined as the number of immediate neighbors.

³ According to measurements in [4], the constant k is approximately -2.3 for the Gnutella network.

rest of the paper, we will refer to a topology generated with the Barabási-Albert model simply as a *Barabási topology*.

2.4 TTL

For a broadcast in Gnutella, the number of nodes that can be reached from a node depends on the TTL value of the initial message. In our analysis and simulations of Gnutella's flooding-based broadcast, we set the TTL to a maximum value, which guarantees that a broadcast can reach 100% of the network. To choose a smaller TTL value would simply result in a broadcast in smaller sub-network. Furthermore, this allows us to compare the results for flooding with the ones obtained for alternative routing methods that do not have the concept of a TTL value.

3. The Cost of Flooding-based Broadcast

In this section, we try to find the cost of flooding-based broadcast as it is implemented in Gnutella. Under the assumption that the initial TTL value is high enough for the broadcast to cover the entire network, the cost can be calculated relatively easily. During a broadcast, every node receives the message at least once. The first time the message is received, the receiving node forwards it to all its neighbors, except the one from where the message came from. Subsequent arrivals of the same message are ignored. Therefore, the number of messages that each node has to forward per broadcast is limited to the number of its neighbors, minus one. Only the node initiating the broadcast sends the message to all its neighbors, which results in one additional message to be forwarded. Hence, the cost c of a broadcast as defined in the previous section can be calculated simply as a function of the number of nodes N and the average node degree d , i.e. the average number of neighbors per node. Flooding guarantees that all N nodes of the network are being reached by a broadcast, which means $r = N$.

$$c = \frac{1 + \sum_{i=1}^N (d_i - 1)}{N} = \frac{1 + N(d - 1)}{N} = d - 1$$

c : cost in number of messages forwarded per node reached

d_i : degree of node i

d : average node degree

N : number of nodes

For N sufficiently large, the cost can be approximated with $d-1$. It is interesting to note that the cost c grows linearly with the average node degree and is virtually independent of N .

To get an idea about the scalability of Gnutella's broadcast based services we need to know the load in terms of resource usage per node, rather than the load on the whole network. For this, we calculate the average bandwidth that is consumed per node with flooding-based broadcast. We assume that the nodes in the network initiate broadcasts with an average rate of R [1/s]. This results in a total average rate of NR broadcasts per second. Multiplying by cN , the total number of messages forwarded per broadcast, yields the total number of messages forwarded in the network per second.

To get the total amount of bandwidth consumed in the network, we multiply by $2M$, the average message size in bytes. The factor 2 stems from the fact that forwarding a message consumes bandwidth on the incoming as well as the outgoing path on each node. Finally, to get the average bandwidth usage per node, the result is divided by N . Thus, the average bandwidth consumption b per node can be computed as follows:

$$b \approx 2NRM(d-1)$$

b : average bandwidth usage per node [byte/s]

N : number of nodes

R : average rate of broadcasts per node [1/s]

M : average message size [byte]

d : average node degree

Interestingly, the average load in terms of bandwidth usage per node scales linearly with the network size N . To get concrete numbers for a typical peer-to-peer scenario, we conservatively assume the average message size M to be 22 bytes, which is the minimal Gnutella message size, and the rate R to be one broadcast per minute. Table 1 shows the resulting average load per node in kBits/s for varying network size N and average node degree d .

	d = 3	d = 4	d = 5	d = 6
N=100	1.18	1.76	2.35	2.94
N=1000	11.74	17.61	23.47	29.34
N=10'000	117.34	176.01	234.67	293.34
N=100'000	1173.34	1760.00	2346.67	2933.34

Table 1: Average Bandwidth consumption (in kBit/s) per Node some P2P example scenarios

The table shows that for large networks, to provide broadcast based services, the required average bandwidth per node is quite considerable. For example, in a network of 10'000 nodes and an average node degree of 4, the required average bandwidth per node is already 176kBit/s. This is already beyond the resources of a typical user connected via a modem. In the real Gnutella network, it has been observed [4] that the high cost of broadcasting and the lack of resources of a large number of participants lead to a fragmentation of the network into smaller sub-networks. If fully decentralized peer-to-peer networks such as Gnutella are to be scalable beyond the size of a few thousand nodes, a more cost-effective way to perform application layer broadcast is required.

4. Rumor Mongering

In this section, we discuss Rumor Mongering as an alternative routing protocol to implement broadcast. Using simulation, we compare its performance with broadcast based on flooding.

4.1 The Protocol

Rumor Mongering or Gossip protocols are a class of probabilistic protocols for message routing. They are also called *epidemiological* protocols, since messages are spread in a network much like a disease in a susceptible population [15], i.e. the neighbors to which messages are forwarded by each node are chosen randomly.

Gossip protocols have been used for a wide range of applications, such as database consistency management [15], reliable multicast [16], failure detection [17], garbage collection [18] and broadcasting on small networks [19]. In most of these applications, it is assumed that every node can directly communicate with every other node or that the global topology of the network is known. This allows a specific type of graph to be superimposed on that network, on which the Gossip protocol operates. Some of the primary research in this area is concerned with finding an optimal topology graph to be superimposed in order to minimize the cost or maximize the reliability of the protocol.

In the context of decentralized peer-to-peer networks, it is not possible to superimpose an arbitrary graph on the network, since peer-to-peer networks are created in an ad-hoc fashion and new nodes can connect to any node on the network they want. Therefore, the exact global topology is not known and the individual nodes have only a very local view of the network. Little work has been done to study the performance of Gossip protocols in the context of ad-hoc peer-to-peer networks.

In this section, we evaluate one specific type of Gossip protocol, called *Blind Counter Rumor Mongering* in [19], as an alternative to flooding for implementing application level broadcast. This protocol determines how messages are routed at each node and has the following structure:

A node n initiates a broadcast by sending the message m to B of its neighbors, chosen at random.

When (node p receives a message m from node q)

If (p has received m no more than F times)

p sends m to B *uniformly randomly* chosen neighbors that p knows have not yet seen m .

The parameter B specifies the maximum number of neighbors a message is forwarded to. The parameter F determines the number of times a node forwards the same message m to B of its neighbors. For example, if F is set to one, a node forwards the message m to B of its neighbors only the first time it receives m . Subsequent arrivals of m are ignored.

A node p knows if its neighbor q has already seen the message m only if p has sent it to q previously, or if p received the message from q . In the case where the number of valid neighbors for a node is less than B , the message is only forwarded to this smaller number of neighbors.

It is often very difficult to obtain analytical expressions to describe the behavior of a Gossip protocol, even for relatively simple topologies. It is not possible to give an analytical expression for the cost parameter c as we did for flooding. However, we can give an upper limit. In Rumor Mongering, the number of messages forwarded by each node that is reached by the broadcast is bounded by BF , representing an upper limit for the cost c . We resort to simulation to find the cost of broadcast based on Rumor Mongering and to compare its performance with flooding-based broadcast.

4.2 Simulation Results

We expect Gossip protocols to implement broadcast at a lower cost than flooding. In Gossip, each node forwards a message only to a randomly selected subset of its neighbors, as opposed to all of them in flooding.

The reduced cost in number of messages forwarded comes at a certain cost. First of all, due to its probabilistic nature, a Gossip protocol cannot guarantee that all the nodes are reached by a broadcast. Secondly, Gossip requires more time to complete a broadcast than flooding. In flooding, a message is routed along all the possible paths in the network in parallel, including the shortest ones. Gossip cannot guarantee that the messages are routed along the optimal path, resulting in an increased time of completion of the broadcast. Table 2 shows how flooding and Rumor Mongering compare in terms of these three parameters: cost, reach and required time steps to complete the broadcast. The results are obtained by simulation, averaging over 1000 runs. The table shows the mean values, the minimum, the maximum and the standard deviation. For flooding, all the parameters are constant, except for the required time steps, since this value depends on where in the topology a broadcast was initiated. The simulation was done for a Barabási topology of 1000 nodes with an average node degree of 6. We chose the parameters F and B for the Rumor Mongering protocol to be 2.

	Flooding				Rumor Mongering (F=2, B=2)			
	mean	min	max	std dev	mean	min	max	std dev
cost c	4.99	4.99	4.99	0	2.78	2.73	2.82	0.01
reach	1000	1000	1000	0	918.44	894	945	8.52
time	8.94	8	10	0.36	21.33	19	30	1.32

Table 2: Flooding vs. Rumor Mongering for a Barabási Topology (N=1000, d=6)

In this case, Rumor Mongering significantly reduces the cost of a broadcast from 4.99 per node reached to a value of 2.78, while still reaching more than 90% of the network. However, the time

steps needed to complete the broadcast with Rumor Mongering, is more than twice the corresponding value of flooding.

Table 3 shows the performance of Rumor Mongering compared to flooding for different combinations of the parameters B and F . The results are shown for Barabási topologies of 1000 nodes with an average node degree d of 4,6 and 8. The mean value for the three parameters, cost, reach and time for Rumor Mongering are shown as comparative figures to the values obtained by flooding, representing 100%.

B	F	d=4			d=6			d=8		
		cost c	reach	time	cost	reach	time	cost	reach	time
2	1	54.1%	55.1%	275.2%	40.0%	67.7%	257.0%	28.6%	66.6%	276.5%
2	2	64.8%	82.3%	258.0%	55.6%	91.9%	240.4%	44.6%	92.8%	255.3%
2	3	69.9%	91.7%	242.4%	63.4%	97.1%	223.2%	48.6%	98.2%	239.7%
3	1	63.7%	76.3%	206.9%	53.0%	87.3%	192.7%	42.9%	88.3%	192.5%
3	2	74.0%	93.4%	187.0%	68.2%	97.7%	170.7%	61.7%	98.7%	174.3%
3	3	78.7%	97.1%	176.3%	74.6%	99.2%	161.3%	69.7%	99.7%	164.6%

Table 3: Performance of Rumor Mongering compared to Flooding

First of all, the table shows how the choice of B and F affects the performance of the Rumor Mongering. A low value for B and F achieves a relatively low cost for broadcast. However, the reach is also significantly reduced and the required time to complete the broadcast is high.

By increasing the parameter B , F or both, we can get arbitrarily close to the performance of flooding, achieving a high reach arbitrarily close to 100%, but also paying the price of a higher cost in terms of message forwarding. In fact, flooding is a degenerate case of Rumor Mongering, with F set to one and B set to the maximum node degree in the network. The trade-off of cost, reliability and time can be adjusted to the characteristics and requirements of the application by choosing F and B appropriately. For example, in a peer-to-peer file sharing application like Gnutella, it might be tolerable that a search for files only reaches 90% of all the nodes, if the cost of searching can be reduced by 50%.

The results also show the level of cost reduction that can be gained from Rumor Mongering depends on the average node degree of the network topology. The higher the node degree is, i.e. the more redundant links a network has, the bigger is the potential to reduce cost by replacing flooding by a Gossip protocol. For example, in a topology with an average node degree of 4, Rumor Mongering with the parameters $B=2$ and $F=1$ can reduce the cost to around 54% compared to flooding. For the same protocol, a reduction to 40% and 28% can be achieved for a topology with an average node degree of 6 and 8 respectively.

The reason why Rumor Mongering can achieve a higher cost reduction over flooding for topologies with a high average node degree is that the number of messages to be forwarded by each node in flooding increases with its node degree. For Rumor Mongering, this number is limited to BF , independent of the node's degree.

5. Deterministic Rumor Mongering

In this section, we introduce *Deterministic Rumor Mongering*, a new version of Rumor Mongering that further reduces the cost of broadcast by making use of the fact that the topology of typical peer-to-peer networks exhibits a power-law characteristic.

5.1 The Protocol

In the previous section, we showed that Rumor Mongering significantly reduces the cost of broadcasting in decentralized peer-to-peer networks compared to flooding. In Rumor Mongering, the subset of neighbors to which messages are forwarded, are chosen uniformly at random. Under the assumption that the nodes have no knowledge about the global topology of the peer-to-peer network, a uniformly random selection is the best possible strategy. As mentioned in section 2, a typical characteristic of peer-to-peer networks is a power-law distribution of the node degrees. In this section, we present a new version of Rumor Mongering, which makes use of this information to make a more intelligent decision as to which of its

neighbors to forward messages to. We call this new protocol *Deterministic Rumor Mongering*, since the selection of neighbors is not performed randomly but is based on the node degree of the corresponding nodes, i.e. the nodes with the lowest degree are chosen first. In the following, we will refer to the version of Rumor Mongering discussed in the previous section as *Probabilistic Rumor Mongering*, to differentiate it from the version introduced here.

The structure of the Deterministic Rumor Mongering protocol is as follows:

```
When (node p receives a message m from node q)
  If(p has received m no more than F times)
    p sends m to all of its neighbors of degree one, plus to B
    of the rest of its neighbors with the lowest node degree,
    that p knows have not yet seen m.
```

Messages are always forwarded to all the pendant neighbors, i.e. neighbors of degree one, since these nodes have no other chance to receive the message. Furthermore, these pendant neighbors cannot contribute to the further propagation of the message in the network and are therefore not considered for the limit of B messages to be forwarded by each node.

The rationale behind preferentially forwarding messages towards nodes with a low degree is illustrated in Figure 4. The Figure shows the average number of messages that are received by the nodes in a single broadcast as a function of their node degree. The results were obtained as averages over 1000 simulation runs for a Barabási topology of 1000 nodes with an average node degree of 6 and a Rumor Mongering protocol with the following sets of parameters: $(B=2, F=1)$, $(B=2, F=2)$, $(B=3, F=2)$.

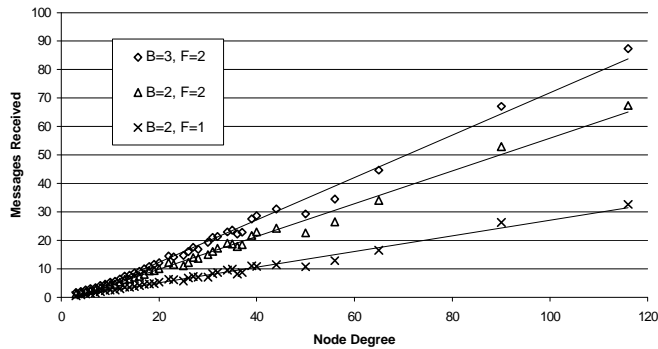


Figure 2: Average Number of redundant message received by nodes in a single broadcast implemented with Rumor Mongering, as a function of the node degree.

The graph shows that nodes of high degree receive a large number of redundant copies of the same message. This overhead grows approximately linearly with the node degree, as is indicated by linear regression trendlines. The slopes of these trendlines also increase with higher Rumor Mongering parameters B and F .

The idea behind Deterministic Rumor Mongering is to reduce the number of redundant messages that are sent to high degree nodes and therefore reduce the overall cost of broadcasts. Figure 4 shows that the high level of message overhead represents a significant potential of cost reduction.

The only requirement to implement Deterministic Rumor Mongering is that each node knows the node degree of its immediate neighbors. This requirement is not in conflict with the decentralized nature of the networks considered here and the exchange of node degree information between neighbors can easily be integrated in a system like Gnutella. For example, a one byte field in the Gnutella message header could be reserved for that purpose, increasing the minimal message size of 22 bytes by less than 5%.

5.2 Simulation Results

Again, we use simulation to study the performance and cost of Deterministic Rumor Mongering and to compare it to the probabilistic version of Rumor Mongering, discussed in the previous section. For a Barabási topology of 1000 nodes and an average node degree of 6, Table 4 provides the relevant parameters for the two versions of Rumor Mongering for a range of choices for B and F .

B	F	Probabilistic Rumor Mongering			Deterministic Rumor Mongering		
		reach	time	cost c	reach	time	cost c
2	1	67.7%	23.2	2.00	96.10%	21.1	2.00
3	1	86.9%	17.1	2.65	99.00%	16.4	2.60
2	2	91.7%	21.6	2.78	99.80%	18.4	2.82
2	3	97.1%	19.9	3.17	100.00%	18.1	3.30
3	2	97.7%	15.3	3.41	100.00%	14.2	3.42
3	3	99.2%	14.4	3.73	100.00%	14.0	3.76

Table 4: Performance of Deterministic Rumor Mongering compared to Probabilistic Rumor Mongering

For a given set of parameters B and F , Discrete Rumor Mongering achieves a significant higher reach than the probabilistic version of Rumor Mongering. It also slightly reduces the time required for a broadcast, whereas the cost is virtually the same for both versions. For example, for $(B=2, F=1)$, Discrete Rumor Mongering reaches 96% of the network compared to 67% of its probabilistic counterpart. The question is how this relates to our main goal, the reduction of cost of broadcasting.

The increase in reach at a constant cost per node translate indirectly into a reduction in cost, considering the fact that increasing the reach of Rumor Mongering by increasing the parameters B and F , results in a in a higher cost per node reached (see Table 3).

The advantage of Deterministic Rumor Mongering is more clearly illustrated in Figure 3. For different combinations of the parameters B and F , the graph shows the corresponding reach and cost parameters for both protocol variants. For a given targeted reach, Discrete Rumor Mongering implements a broadcast at a much lower cost than Probabilistic Rumor Mongering. For example, to reach 97% of the nodes with Rumor Mongering, we would choose the following protocol parameters: $B=2, F=3$. In this configuration, the cost parameter c is 3.17. To achieve a comparable reach of 96% with Deterministic Rumor Mongering, we choose $(B=2, F=1)$ and get a value of 2.00 for c , which is a reduction in cost of 37% over Probabilistic Rumor Mongering. As a comparison, the graph also shows the operating point of flooding, with a reach of 100% and a cost c of 4.99, compared to which Deterministic Rumor Mongering achieves a cost reduction of 60%. The reduction in cost for broadcast that a Rumor Mongering protocol can achieve compared to flooding, increases for topologies with a higher average node degree d , as has been shown in Table 3.

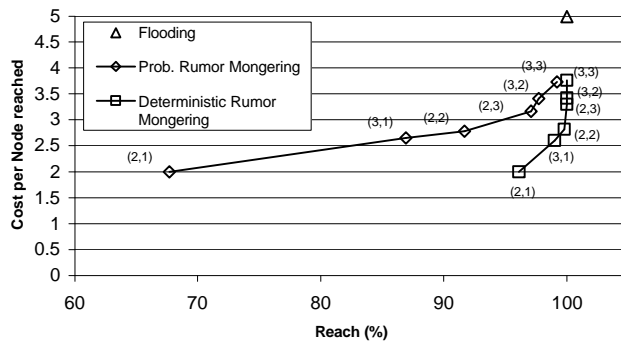


Figure 3: Cost of Broadcasting versus reach, for different sets of protocol parameters B and F

So far, all our simulations were done for networks of a constant size of 1000 nodes. To study the effect of the network size on the performance of Deterministic Rumor Mongering, we ran our simulation for the three different network sizes of 100, 1000 and 10'000 nodes. All these

networks have an average node degree of 6 and the protocol parameters were chosen as follows: $B=2$, $F=1$. The results shown in table 5 indicate that the network size has little or no effect on our cost parameter c and the proportion of the network that is reached. Only the number of time steps required to complete the broadcast increases slightly with the total number of nodes.

N	cost c	reach	time
100	2.01	97.60%	14.97
1000	2.00	97.05%	22.34
10'000	2.00	96.17%	26.01

Table 5 : Performance of Deterministic Rumor Mongering depending on the network size N, for $d=6$, $B=2$, $F=1$.

As has been shown in section 2, the cost parameter c for flooding does not depend on the network size. Table 5 indicates that this is the case for Deterministic Rumor Mongering as well, which means that its relative reduction of cost over flooding is equally independent of the network size.

6. Conclusions

For fully decentralized peer-to-peer networks such as Gnutella, application layer broadcast is often an important building block that is required to implement services such as searching. In this paper, we studied the cost of Gnutella's flooding-based broadcast mechanism. We defined a metric of cost and derived a simple formula to calculate the cost, based solely on the size of the network and the average node degree. Based on this, we made an estimation of the average bandwidth that is required per node for a few typical scenarios. This showed, as expected, that flooding-based broadcast becomes prohibitively expensive, even for relatively small networks.

As an alternative to flooding, we evaluated a probabilistic protocol called Rumor Mongering. Our simulation results show that Rumor Mongering is more scalable, since it can significantly reduce

the cost of broadcast compared to flooding. It trades off reduction in cost against reduced reach and increased time to complete the broadcast. By choosing the parameters B and F of the Rumor Mongering protocol, the level of this trade-off can be controlled almost arbitrarily within a certain range.

We further introduced a new protocol called Deterministic Rumor Mongering. This new broadcast routing protocol achieves a much better performance than the probabilistic version of Rumor Mongering by making use of the fact that typical peer-to-peer topologies show power-law characteristics in their node degree. We showed that for a typical scenario, Discrete Rumor Mongering can reduce the cost of broadcasting by around 60% compared to flooding, while still reaching more than 96% of the nodes in the network.

Furthermore, we showed that the comparative performance of Rumor Mongering versus flooding does not depend on the network size, but on the average node degree of the network topology. The more redundant links a topology has and the higher its average node degree, the higher is the potential gain of using Rumor Mongering as an alternative to flooding to implement broadcast.

References

- [1] The Napster home page, <http://www.napster.com>
- [2] The Gnutella home page, <http://gnutella.wego.com>
- [3] The SETI@home home page, <http://setiathome.ssl.berkeley.edu>
- [4] Gnutella: To the Bandwidth Barrier and Beyond, <http://www.clip2.com/gnutella.html>
- [5] David G. Andersen, Hari Balakrishnan, M. Frans Kaashoek, Robert Morris, "Resilient Overlay Networks", Proc. 18th ACM SOSP, Banff, Canada, October 2001
- [6] Y. Chu, S. Rao, and H. Zhang. "A Case For EndSystem Multicast". In Proceedings of ACM Sigmetrics, Santa Clara, CA, June 2000.

- [7] S. Ardon, P. Gunningberg, Y. Ismailov, B. Landfeldt, M. Portmann, A. Seneviratne, B. Thai, "Mobile Aware Server Architecture: A Distributed Proxy Architecture for Content Adaptation and Service Enhancement", INET2001, Stockholm, June 2001
- [8] Gnutella Protocol Specification, <http://www.clip2.com/GnutellaProtocol04.pdf>
- [9] Gnutella: To the Bandwidth Barrier and Beyond, <http://www.clip2.com/gnutella.html>
- [10] T. Hong, in "Peer-to-peer: Harnessing the Benefits of a Disruptive Technology", edited by Andy Oram (O'Reilly, Sebastopol, CA, 2001) Chap. 14, pp. 203-241
- [11] Michalis Faloutsos, Petros Faloutsos, Christos Faloutsos, "On Power-Law Relationships of the Internet Topology", ACM SIGCOMM'99, Boston, 1999.
- [12] W. Aiello, F. Chung, and L. Lu. "A Random Graph Model for Massive Graphs". In 32nd Annual Symposium in Theory of Computing, 2000.
- [13] A.L. Barabasi and R. Albert, "Emergence of Scaling in Random Networks". *Science*, pages 509-512, October 1999.
- [14] Alberto Medina, Anukool Lakhina, Ibrahim Matta, and John Byers, "BRITE: An Approach to Universal Topology Generation". In Proceedings of the International Workshop on Modeling, Analysis and Simulation of Computer and Telecommunications Systems- MASCOTS '01, Cincinnati, Ohio, August 2001.
- [15] A. Demers, D. Greene, A. Hauser, W. Irish, J. Larson, S. Shenker, H. Sturgis, D. Swinehart, and D. Terry. "Epidemic algorithms for replicated database maintenance". In Proc. ACM Symp. on the Principles of Distr. Computing, pages 1--12, August 1987.
- [16] K. Birman et al. "Bimodal multicast". Cornell University, Department of Computer Science Technical Report TR-98-1665, May 18, 1998
- [17] R. van Renesse, Y. Minsky, and M. Hayden. "A gossip-style failure detection service". In Proceedings of the IFIP International Conference on Distributed Systems Platforms and Open Distributed Processing (Middleware '98), The Lake District, England, September 1998, pp.55-70.
- [18] K. Guo, et al. "GSGC: an efficient gossip-style garbage collection scheme for scalable reliable multicast". Cornell University, Department of Computer Science Technical Report TR-97-1656, December 3, 1997

- [19] M.-J. Lin, K. Marzullo, and S. Masini. "Gossip versus deterministic flooding: Low message overhead and high reliability for broadcasting on small networks". Technical Report CS1999-0637, University of California, San Diego, Computer Science and Engineering, 1999.
- [20] I. Clarke, O. Sandberg, B. Wiley, and T.W. Hong, "Freenet: A Distributed Anonymous Information Storage and Retrieval System" in *Designing Privacy Enhancing Technologies: International Workshop on Design Issues in Anonymity and Unobservability, LNCS 2009*, ed. by H. Federrath. Springer: New York (2001).