Heuristics for Flash-Dissemination in Heterogenous Networks

Mayur Deshpande, Nalini Venkatasubramanian and Sharad Mehrotra
School of Information and Computer Science
University of California, Irvine
{mayur,nalini,sharad}@ics.uci.edu

Abstract

Flash Dissemination is a particularly useful form of data broadcast that arises in many mission-critical applications. The goal is rapid distribution of medium amounts of data in as short a time period as possible. While optimal algorithms are available for a highly constrained case (all nodes having the same bandwidth and latency), there is relatively little work in the context of heterogenous networks. Most systems and protocols today either use trees or randomized mesh-based techniques to deal with heterogeneity and mostly work with local knowledge. We argue that a protocol with global knowledge can perform much better. In this paper, we propose two centralized heuristics – DIM-Rank and DIM-Time that use global knowledge to schedule data transfer between nodes. The heuristics are based upon insights from broadcast theory. We perform detailed experimental evaluation of these two heuristics with two decentralized randomized approaches, CREW and NOD. Currently, CREW is the one of the fastest decentralized flash-dissemination protocols. Our experiments show that DIM-Rank is better among the two centralized heuristics and achieves faster dissemination than CREW and NOD across a range of heterogeneity metrics.

I. Introduction

Fast distribution of data to multiple receivers is a basic primitive and required functionality in several application domains. Differing characteristics of these application domains (the state of the network and recipients, the data to be disseminated, the situation under which dissemination occurs) pose different challenges that has a profound influence on the choice of the dissemination algorithm. For instance, in a stock trading scenario, the dissemination algorithm must be designed to optimize scalable delivery of a steady ”stream” of stock information to online traders while at the same time meeting their ‘coherence’

requirements [3]. In contrast, dissemination approaches meant to propagate updates to players in Massively Multiplayer Online Games must be designed to support scalable, in-order dissemination of bursty-data [12] (since data in such applications is generated in bursts, often as a result of players seeing/reacting to one another).

In this paper, we study a particularly useful form of dissemination that arises in mission-critical applications which we term as *flash dissemination*. Such a scenario consists of rapid dissemination of medium amounts of data to a large number of recipients in a very short period of time. Consider, for example, an organization that has geographically distributed data-centers located at various ISP points. Periodically, the data centers need to be synchronized with a global master list (or latest security patches). Fast delivery of this information to all centers is critical to avoid loss of downtime or observable 'glitch' by users. As another example, from the emergency management domain, consider a service such as "Shakecast" [2] of the USGS (United States Geographical Survey). Earthquake information sent out by Shakecast is a Shake-Map" (image-file) of 100-300KB. This information is sent to various city and county emergency management organizations that subscribe to the USGS. The goal is to provide accurate and timely data and information about seismic events as quickly as possible.

At an abstract level, both these applications fall under the network wide broadcast problem where a particular node wants to broadcast some data to all other recipient nodes as fast as possible. While network wide broadcast is a mature area with more than 20 years of research, the problem of high-speed dissemination in heterogenous networks is a new problem. In the highly constrained case of all nodes with homogenous bandwidth and latency, an optimal solution to the achievable lower bound was proposed in 1980 [7]. This was rediscovered again in 2005 [9] (in the context of overlay P2P (Peer-to-Peer) networks) and the authors also proposed an alternative approach using a hypercube to achieve the lower bound. However, when nodes are allowed to have varying bandwidths and/or latency, the problem becomes NP-hard [11]. In [11], the authors only consider a case where the data to be distributed is one single piece (or chunk). Multi-chunk distribution in heterogenous networks adds further complexity to this scenario.

Multi-chunk distribution usually leads to faster dissemination.

Current systems either use overlay trees (Narada [14], Splitstream [5]), or more recently, meshes (Bittorrent [1], CREW [6]) or a hybrid of both (Bullet [13], Bimodal-Multicast [4]) to deal with multi chunk distribution in heterogenous networks. Though not mathematically proven, randomized approaches perform quite well in real world settings and much empirical work substantiates this [9], [13]. However, many of these systems [1], [13] are either tailored towards streaming or large amounts (GBs) of data or small size events. In the scenario of interest to us, data size is usually in the middle range of hundreds of KBs to tens of MBs. As we explain later, fast dissemination of medium size data requires a protocol to do both *ramp-up* and *sustained-throughput* very well. Ramp-up is the time needed for each node to start participating in the dissemination process. In sustained-throughput, a node is able to sustain high transfer rates. Currently, CREW is a protocol that addresses this special data range.

However, all these systems use some form of randomization in their protocols and work with mostly local knowledge. We argue that when low dissemination time is of utmost importance, centralized approaches with global knowledge can make a crucial difference in performance. We propose two heuristics for multi-chunk dissemination in heterogenous networks that we call DIM-Time and DIM-Rank. The heuristics need global knowledge and a centralized ‘scheduler’ to orchestrate data transfer between nodes. By global knowledge, we mean pair-wise bandwidth and latency measures among all participating nodes. The heuristics are based upon the insights obtained from the original optimal solution to homogenous data broadcast [7]. We show via experiments that DIM-Rank achieves lower dissemination time as compared to randomized approaches across a range of heterogeneity metrics. The rest of the paper is as follows. In Sec-II, we formalize the problem of flash dissemination. In Sec-III we present our centralized heuristics, and situate them in a taxonomy of related research. We compare the heuristics to randomized mesh based approaches in Sec-IV and conclude in Sec-V

II. Problem Formalization

In this section, we first define the problem of flash dissemination more concretely. Let ν be a set of N nodes $\nu = N_1, N_2, ...N_N$ connected by an underlying fully connected network. Let $N_{seed} \in \nu$ be the seeder node with the data item D to be disseminated. The objective is to get D to all non-seeder nodes in ν as fast as possible.

A. Chunk based representation of data

We view the data item D to be disseminated as a sequence of M equal sized chunks. Note that the view of messages as a series of chunks is a generalized representation, of which single message transfer is a special case. A chunk is an aggregation of one or more bytes of data. A chunk may have a header that explicitly details the number and characteristics (e.g. checksum) of the bytes contained in it. Thus, a receiver can verify when it has got a complete chunk from the sender. Meta-information regarding the chunks contains details on how received chunks must be 'stitched' to get back the original information. A node has to receive (and verify) the whole chunk before it can transfer it further.

Chunk dissemination is advantageous and flexible because chunks can be disseminated asynchronously, be received out-of-order and then finally assembled. Furthermore, it supports increased concurrency in the dissemination process since multiple nodes can start propagating the chunks they have received so far. In fact, [7] showed how such a chunk-based dissemination leads to an optimal solution in a homogenous network and how an optimal chunksize for a given dissemination can be found. The objective of flash dissemination is therefore to deliver all M chunks of D to every non-seeder node, $N_j \in \nu, N_j \neq N_{seed}$ in the minimum amount of time.

B. Chunk based dissemination over heterogenous networks

Each node in the network has a certain *capacity* or rate at which it can transmit (or receive) data to (from) another node (also called it's *bandwidth*). Additionally, there is certain *delay* (or *latency*) defined as the time it takes for one byte of data to be transmitted from the sender to the receiver.

In a homogenous network all nodes have the same bandwidth and equal inter-node latency, so the time to transmit a message between any two peers is equal. However, this is not an accurate model to capture dissemination in the Internet where peers have different bandwidths (T3, T1, DSL, etc.) and inter-peer latencies vary considerably (from 1-1000 millisecs). We use the following characterization to describe chunk-based dissemination in heterogenous networks. Let the maximum capacity/bandwidth of a node n be $MaxBW(n)$. Different nodes can have different Max-bandwidths. Any pair of nodes, (x, y) has a latency denoted as $Lat(x, y)$. We assume $Lat(x, x)$ is minuscule and can be approximated to 0. Further $Lat(x, y) = Lat(y, x)$. A node's bandwidth may be partially reserved for an ongoing transfer and its leftover (or available) bandwidth is denoted as $AvailBW(x)$. When a pair of nodes initiates a chunk-transfer, the sustained bandwidth for the transfer is denoted as $SusBW(x, y)$ and it has the following property: $SusBW(x, y) \leq Min(AvailBW(x), AvailBW(y))$. The time required to transfer a chunk of size D between $x \rightarrow y$ is then $Lat(x, y) + \frac{D}{SusBW(x,y)}$. This time can also be higher, if for example, a connection needs to be established first between the two nodes. For instance, in TCP, a 3-way handshake is needed to establish the connection and hence the time required can be approximated as $3 * Lat(x, y) + \frac{D}{SusBW(x,y)}$. Nodes can 'split' their bandwidth into any combination of uploads and downloads. Thus a node can be engaged in multiple transfers, some upload and some download. Next, we present our heuristics for flash dissemination over heterogeneous networks.

III. Centralized Flash Dissemination Heuristics

In this section, we present our heuristics, DIM-Time and Dim-Rank, for flash dissemination in heterogeneous networks. However, we first provide a brief summary of the theoretical basis for these heuristics. We conclude with a taxonomy of research on data broadcast and situate our heuristics in it.

A. Theoretical Background

The theoretical base consists of two results for scheduling data broadcast: a multi-chunk broadcast in a homogenous network and a single-chunk broadcast in heterogenous bandwidth network.

In a homogenous network, the optimal solution for broadcast[1], which we call DIM, can be broken down into two main parts:

1) **Phase 1: Ramp-up phase** - which ensures that every node receives at least one chunk and
2) **Phase 2: Sustained-throughput phase** - which ensures that the total available capacity is used to transfer chunks.

The lower bound for the first phase is $Log(N)$. This is because, at each unit in time, the number of nodes that have atleast one chunk can be doubled, resulting in the $Log(N)$ bound. This can be done if each node that has a chunk picks a node that does not have a chunk as the receiver.

The lower bound for the second phase is $2M-1$. Achieving this involves realizing some clever insights. In the first phase, let the seeder transfers a new chunk to a new node at each point in time. The chunk that is put out earlier propagates to more nodes than later chunks. For instance the last chunk that the seeder put out is present in only two nodes (the seeder and the recipient). On the other hand the first chunk that the seeder put put has multiplied by two in each successive time step and is therefore, present in $2^{Log(N)}$ nodes. If all nodes are covered in ramp-up phase when the seeder puts out the M^{th} chunk, then half the nodes have the first chunk. This is at the end of the Ramp-up phase. After this point in time, the nodes can be cleverly partitioned into senders and receivers so that each node will either have something to receive or something to give at each point in time. More formally:

1) The seeder transfers a new chunk at each unit of time (say $chunk_1$, $chunk_2$, etc.)
2) At end of ramp-up phase, half the nodes have $chunk_1$, $1/4^{th}$ have $chunk_2$ ... 1 node has $chunk_{logN}$ (for brief $chunk_l$). *This is a key insight*, since now the whole set of nodes can be exactly partitioned into *transmitters* and *receivers*. The receivers are the nodes that contain the majority chunk and are half of total number of nodes. The other half are the transmitters.
3) Nodes having $chunk_2 ... chunk_l$ can transfer their chunk to nodes that have $chunk_1$ (and all nodes will be busy). Thus the *spread* of each of the chunk (unless it has already spread to half to nodes)

[1]detailed analysis and explanations can be found in [7], [9]

doubles in a time unit.

4) This partitioning can be repeated for $l-1$ time units, at which point all chunks have spread to half the nodes.
5) In last l time units, $chunk_1 ... chunk_l$ can become fully spread to all nodes. Here again exact partitioning is achieved. One half are transmitters of a chunk (that is spread in half) and other half are receivers.

Thus, we observe that the key challenge in multi-chunk dissemination is the **Optimal Partitioning** of nodes into transmitters and receivers so that all nodes are kept busy, not just for a particular time instant but also for all subsequent time steps.

DIM assumes a homogenous network where all nodes have the same bandwidth and equal inter-node latency, so that the time to transmit a message between any two peers is equal. This is not true in practice since individual link bandwidths and inter-peer latencies vary significantly. A recent result [11] shows that the problem of minimizing the time for broadcasting a single message in a heterogeneous network is a NP-hard problem; the authors also show that the Fastest-Node-First (FNF) heuristic is optimal in many cases for single-message broadcast. The FNF broadcast tree problem is restricted to one message and it is not entirely clear if it is also a good heuristic for broadcast of multiple messages. For example, when there is only one message to transmit, then different peers are picked for reception of message at consecutive steps from the seeder. However, if there are multiple messages, then it is not entirely clear whether the fastest node should get all the messages first or another scheme should be followed.

B. Our Heuristics: DIM-RANK and DIM-Time

Using metrics that capture the key insights of the optimal solution (in the homogenous case), we derive two heuristics, that are better than a simple adaptation of FNF. An elegant property of these heuristics is that in the case of a homogenous network, they work close to the optimal solution and in the case of single-message dissemination in heterogenous network they work like FNF. We then embed these heuristics into

a demand-driven framework, thus realizing a dynamically adaptive system for flash-dissemination system in a heterogenous network.

Any solution that addresses the key challenge of optimal partitioning must determine the following at each decision point: (1)the set of transmitters, (2)the set of receivers and (3)chunks that transmitters must send to receivers. Note that(1) and (3) are intertwined since what chunks a node possess factors into deciding the node's role. To aid in optimized partitioning, we define the following metrics:

Chunk Spread: The *spread* of a chunk is, c_i, is the total number of nodes that have c_i. More formally, $spread(c_i) = |\{P_i\}|$ where $\{P_i\} = \{x : x \text{ contains } c_i\}$. Spread of a chunk thus quantifies how rare of popular a chunk is [2].

Node Rank: The node's rank is defined as $Rank(n) = \sum_i \frac{1}{spread(c_i)}$ i.e. the summed inverse of spread of all chunks that it contains. Thus, a node's rank is higher if it either contains rare chunks or many chunks. Conversely, if a node only contains popular chunks, it's rank is low.

The choice of the above metrics is not arbitrary; it captures key insights of DIM. In the optimal solutions, the receivers were nodes that either did not have any chunks or had chunks that were in the *majority*, i.e. half of the nodes already had the same chunk. Nodes which had rare chunks were the transmitters. Thus, when deciding which chunk to transmit among a set of chunks, a node should transmit the lower-spread chunk. Similarly, if the rank of a node is high, it should be considered for transmitting a chunk and if it's rank is lower, it should instead receive a chunk.

We derive two heuristics (DIM-Time and DIM-Rank) using the metrics defined. We assume that the heuristic is to be applied to a set of nodes that have spare capacity (called $availNodes$ henceforth) and decisions have to be made on how to split them into transmitters and receivers and what chunks should be transferred. The operational flow of both DIM-Time and DIM-Rank is as follows:

1) **Select transmitter:** All nodes in $availNodes$ are sorted (highest to lowest) according to their rank (and further by capacity, for nodes of equal rank). The highest ranked node is picked to be a

[2]If so desired, spread can be normalized by the total number of nodes

transmitter.

2) **Sort receivers:** The next question that arises is who should be a receiver for *this* transmitter. For this, a set of nodes is constructed that do not have at least one chunk that the transmitter has. This set of nodes is then sorted, using one of two policies:

   - **DIM-Time:** The nodes are sorted (1) First by their capacity (highest to lowest), (2) and then by their rank **(lowest to highest)**.
   - **DIM-Rank:** The nodes are sorted (1) First by their rank **(lowest to highest)**, (2) and then by their capacity (highest to lowest).

   Note that receivers are sorted lowest to highest when using the rank metric. This is so that nodes with more popular chunks act as receivers and allow rare chunks to be propagated. Also, in DIM-Time, the goal is to give preference to the FNF intuition over the optimal-solution while the converse holds for DIM-Rank.

   Once the receiver set is sorted, nodes are picked from the list and assigned to the transmitter.

3) **Determine chunks to transfer:**

   The transmitter (selected in step-1) now starts picking receivers from the sorted list. For each receiver, the lowest-spread chunk in the intersection set is chosen. The transmitter starts sending the chunks until it has no more spare bandwidth. The transmitter is then removed from $availNodes$. Any receiver whose available bandwidth drops to zero is also removed from $availNodes$.

The process is then repeated from the start until either no more transfers can be set up or there are no more nodes in $availNodes$. The above algorithm is run by the central scheduler. Nodes report their initial capacity, any change to capacity and what chunk they received to the scheduler. The scheduler, therefore has full knowledge (about both chunks and spare bandwidth) of the network. One point in the implementation of the scheduler is how often it should be run. If it is run too often, then the number of nodes in $availNodes$ may be too small at any given point of time. If the scheduler is run far too infrequently, then nodes will waste too much time just waiting for the scheduler to tell them what to do.

Thus, there exists an optimal periodicity of the scheduler. However, we do not address this issue here. In our experiments we run the scheduler every 200ms.

Fig. 1 depicts an sample dissemination for 3 chunks (from Node 1) and 10 nodes (Nodes 1 and 2 have twice the bandwidth of the remaining nodes) for 3 centralized protocols - (i) a naive-FNF adaptation, (b) DIM-Time and (c) DIM-Rank. A chunk takes one unit of time to transfer the chunk from node 1 to node 2, and 2 units of time to transfer a chunk to any other node. The Naive-FNF adaptation for multiple chunks works as follows: (a) the highest capacity node is the transmitter, (b)the next highest capacity node with missing chunks is the receiver and (c) at every time step, the receiver node obtains the next missing chunk.

In Fig. 1(a),Node-1 first transfers all chunks to Node-2 in 3 time-steps. Thereafter, each chunk is disseminated (one by one) to the lower bandwidth nodes. This takes 15 time units in total. Fig. 1(b)depicts DIM-Time . Here, the initial steps are same (Node-1 transfers all chunks to Node-2 as per FNF) but in the later steps, different chunks are disseminated in the same round. This increases the number of concurrent transfers in the system and the total time reduces to 13 time units. Lastly, in Fig. 1(c), depicting DIM-Rank, Node-1 does not transfer all chunks to Node-2 at first. Instead, the goal here is to get all nodes ‘active’ as soon as possible (similar in spirit to the homogenous optimal solution). As can be seen, this approach achieves an even lower total dissemination time of 11 time units.

Discussion: We now discuss the optimality of DIM-Rank and DIM-Time. In single-chunk heterogenous case, DIM-Time and DIM-Rank reduce to FNF. This is because nodes that have the single chunk would all have the same rank and nodes that had no chunk would all have rank zero. Thus the transmitter is the node with the highest capacity node which also has the chunk. The highest capacity receiver is selected in either case (DIM-Time or DIM-Rank, since the ranks of all receivers are the same (i.e. 0). In short, the highest capacity sender transmits to the highest capacity receiver.

In homogeneous networks, it is easy to see how the above heuristics closely emulate DIM (up until the tail end of the sustained throughput phase). For the ramp-up phase, nodes with no chunks have rank zero

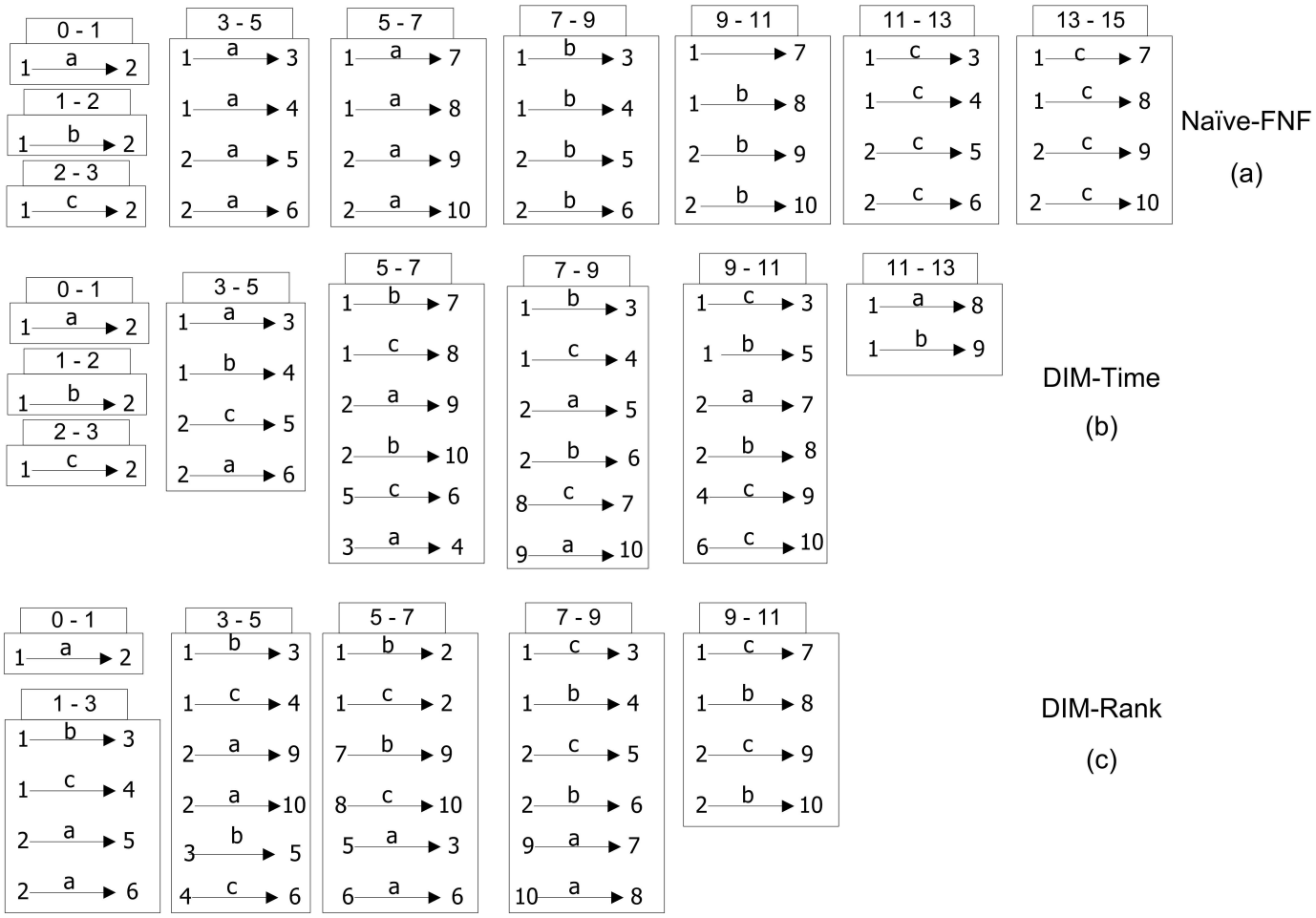


Fig. 1. Steps in Dissemination for Three Greedy Heuristics

and would get picked as receivers until there are no more zero-ranked nodes. This is identical to DIM. In the first part of sustained-throughput phase, nodes that have the most-spread chunk become the receivers till all chunks spread to half the nodes, again identical to DIM. After this, all nodes have the same rank and hence the assignment of transmitters and receivers in DIM-Rank or DIM-Time becomes randomized. We expect this randomized process to achieve high throughput, but this cannot be guaranteed optimal for all cases.

So far we have ignored the cost of computing the schedule in the central coordinator. In the best case scenario, the central scheduler has to sort the list of nodes at least once, thus incurring a cost of $O(NLogN)$. However, it is possible that the scheduler has to sort the list for the transmitters and then sort the list again for receivers for each of the transmitters. This can happen for all M chunks, leading

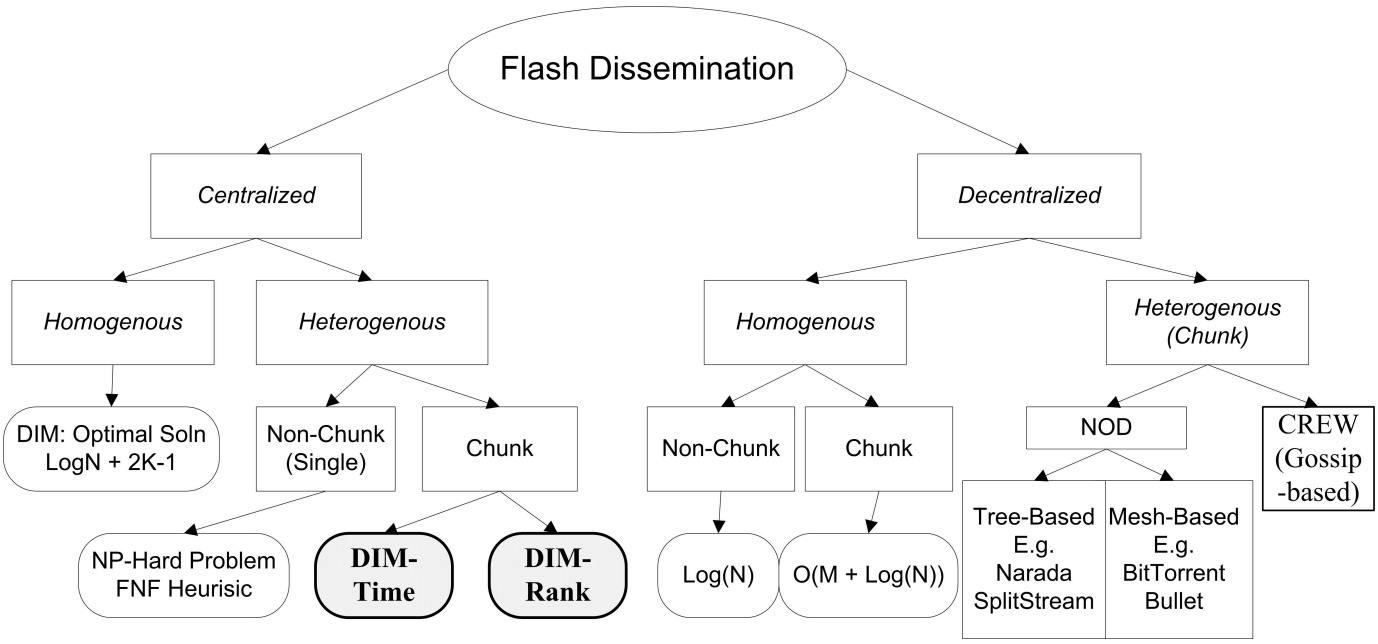


Fig. 2. Scope of our Contributions

to a worst case computing cost of $O(M * NLogN)^2$. As M and N increase, therefore, the worst case computing cost increases exponentially.

C. Taxonomy of Broadcast and our Contributions

We can view solutions to the dissemination problem along multiple dimensions as illustrated in Figure-2.

1) Centralized versus decentralized decision making for dissemination
2) Dissemination for Homogenous versus heterogenous networks
3) Non-chunk (single message) versus chunk-based dissemination

We illustrate the best solutions (that we are aware of), using the taxonomy in Fig-1. We situate our heuristics (shown in bold) in the figure. For the centralized side of the tree, we have already described the related work. For the decentralized side, we start with results in the homogenous case. The results obtained in decentralized case are stochastic in nature. Gossip-based broadcast offers bounds on how long it would take for a single chunk (message) to be broadcast to N nodes. With high probability, it would take ($O(LogN)$) rounds to disseminate a single message (with high probability) to all N nodes [10]. Theoretical results for multi-chunk gossip starting from only one seeder are not yet proven but for M

nodes each starting with a chunk, the dissemination time $O(LogN + M)$ [8]. In heterogenous networks, decentralized systems use an overlay topology to spread chunks. Within overlay based schemes, one can divide the systems into whether they are neighbor-oriented or not. In neighbor-oriented diffusion (NOD) schemes, nodes keep track of the chunks that the neighbors have and then setup exchanges. In case of a tree overlay, the flow of chunks is only in one direction. Splitstream [5], Bullet [13] and Bit-torrent [1] are all examples of NOD based systems. CREW, on the other hand does not maintain neighbor state but uses the overlay as a membership management service for its gossip-based mechanism. In [6], the authors showed CREW to be much faster for flash dissemination as compared to other overlay dissemination systems.

IV. Performance Evaluation

We now describe our empirical evaluation of the centralized heuristics compared to the decentralized mesh-based heuristics.

A. Experiments Setup

We have implemented all four approaches on top of a middleware platform that we built, called RapID. For scalability testing of thousands of peers, we needed to run multiple RapidPeers on a single host. Further, since we wanted to control the delay and throughput between peers for experiments, we developed an emulation layer that intercepts all peer-peer communications. A call to transfer a chunk to a target-peer would therefore not actually transfer the chunk but only emulate the time taken for it; both on the sender and receiver side. The emulation layer is quite detailed and provides the peer with all the details that it normally would ask from the Operating System, such as the current rate of transfer of all ongoing chunks, available bandwidth, (TCP)connection-cache of open connections, etc. When a chunk-transfer is initiated between two peers, the emulation layer on the sender side emulates an upload and the emulation layer on the receiver emulates a download. The emulation layers on both peers do a quick handshake to determine at what rate the transfer will progress. This is arrived by following the formula of $SusBW$ as

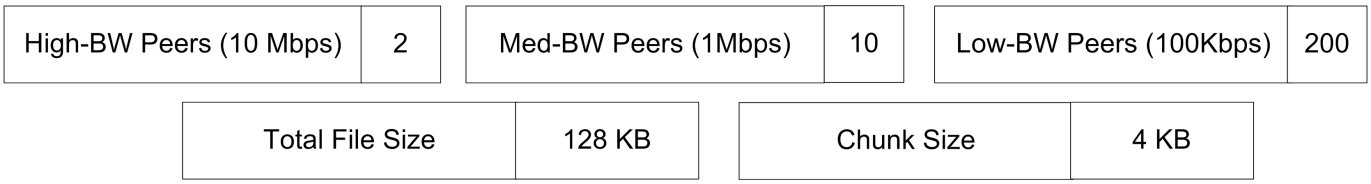


Fig. 3. Default Values Used in Experiments

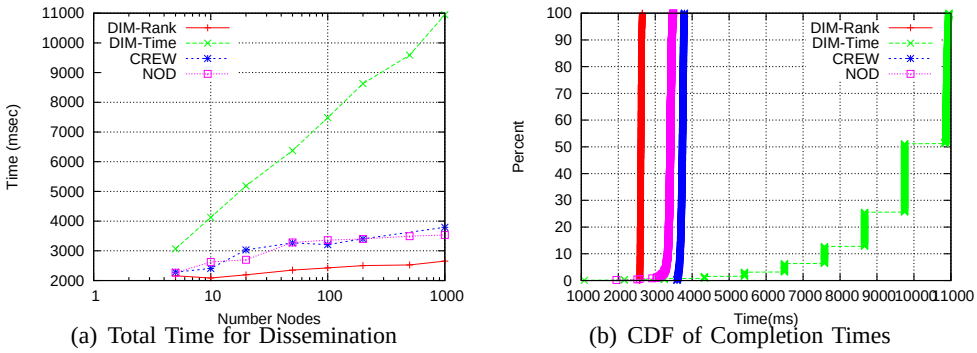
noted in Sec-II. When the appropriate time as elapsed, the emulation layers, readjust the $availBW$ and send appropriate events to the Peers that the transfer is complete. Control messages exchanged between peers are similarly emulated to reflect the latency between the peers. TCP-connection setup time is also emulated if two peers communicate for the first time. Since, there is no actual data transfer between peers, multiple peers can be run a host without hitting the maximum NIC bandwidth. Experiments are run in both homogenous and heterogenous settings. Default values for experiments are shown in Figure-3.

Total time for dissemination is our primary metric and this is calculated in an experiment as follows. The required instances of peers are started up and they all contact a known server to obtain the emulation parameters. The server assigns each peer its capacity (bandwidth) and also gives it a latency-vector to other peers. This vector contains all the latencies for this peer to contact to any other peer in the system. The latency vector is used by the emulation layer to appropriately delay inter-peer communications. The server then tags one of the peers with all the chunks. At this point the server records the $startTime$. In the centralized heuristics, the peers contact the central scheduler who then schedules the peers. As peers receive all chunks, they report the event to the server. When all peers have got all chunks, the server notes the ($stopTime$). The total dissemination time is calculated as $stopTime - startTime$.

B. Performance Results

We now show the dissemination time of the four approaches under various scenarios. We start with a homogenous case and then progressively relax the constraints, showing it's impact on the four protocols. Then, in a heterogenous environment, with a mix of different capacity nodes and real-world latencies, we show the effect to data heterogeneity on dissemination time.

1) Performance in Ideal Homogenous Network: We begin with a 'baseline' ideal-world that is homogenous; all nodes have the same bandwidth (1Mbps) and inter-node latency (2ms). The file to be disseminated is of size 128K Bytes and split into the ideal number of chunks. (as calculated from the optimal solution). The goal is to asses how well the protocols scale in a homogenous world. One expects the total time to complete is linear in $LogN$, where N is the total number of nodes. Figure4(a) shows the total time each protocol takes to disseminate files to all the nodes. Note that the x-axis is a log-scale of number of nodes and hence straight lines indicates scalability in $LogN$. Fig-4(b) shows CDF plots for disseminating the file to 1000 nodes.

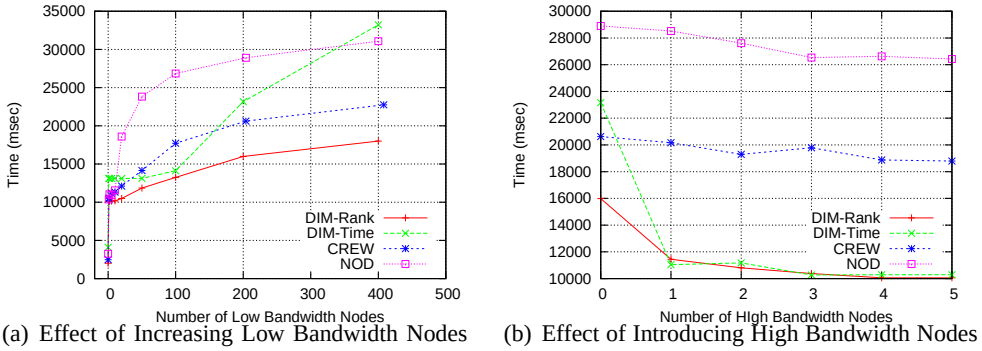


(a) Total Time for Dissemination (b) CDF of Completion Times

Fig. 4. Scalability Results

DIM-Rank performs the best while DIM-Time performs the worst (though all of them scale linearly with $LogN$). DIM-Time performs the worst since it optimizes for lower transfer time. In a homogenous network with equal latencies, the seeder sends all chunks to one node at first. The deciding factor is whether a node has an open connection to another node or not, since the latency is longer when a new connection has to be opened. This trend continues so that nodes get all chunks before they start transmitting to others. Figure4(b) shows this clearly. The number of nodes that complete in DIM-Time (rightmost plot) doubles with time but since nodes get all chunks before disseminating, DIM-Time scales as $O(M * Log(N))$ (hence the steep slope). The other protocols scale closer to $O(M + Log(N))$. This case, therefore, shows how real-world situations can affect heuristics even in the simple case of a homogenous network.

2) Effect of Bandwidth Heterogeneity: To the baseline model, we now introduce bandwidth heterogeneity. The settings for this experiment are as follows. A network of 10 nodes with medium-bandwidth (1Mbps) is the initial baseline. To this, we first add a varying number of low-bandwidth (100Kbps) nodes and study it's effects (Fig-5(a)). We now move the baseline to a network with 10 medium-bandwidth (med-bw) nodes and 200 low-bandwidth (low-bw) nodes. We then study the effect of introducing high-bandwidth (high-bw) nodes of 10Mbps into this network (Fig-5(b)).



(a) Effect of Increasing Low Bandwidth Nodes (b) Effect of Introducing High Bandwidth Nodes

Fig. 5. Performance in Bandwidth Heterogenous Networks

Intuitively, one would expect that introducing low-bandwidth nodes increases the dissemination time and conversely, introducing high-bandwidth nodes decreases the dissemination time. Fig-5(a) shows that the introduction of the first low-bw node causes a significant jump in total dissemination time. This is because the total dissemination time is dictated by when the low-bw node finishes. After this data point, the slope is more linear for all four protocols and they scale linearly in $LogN$, where N is now the number of low-bw nodes. When the first high-bw node is introduced (Fig-5(a)), there is a dramatic reduction in dissemination time for the centralized protocols. It is as if the addition of one high-bw node compensated for the addition of 200 low-bw nodes. Dim-Time and Dim-Rank are fully able to exploit high-bandwidth nodes whereas the effect is more limited for the decentralized protocols. Further introduction of high-bw servers has only marginal effect. Thus, for a given network, introducing high-bw nodes can have a significant impact initially and almost no value for later additions.

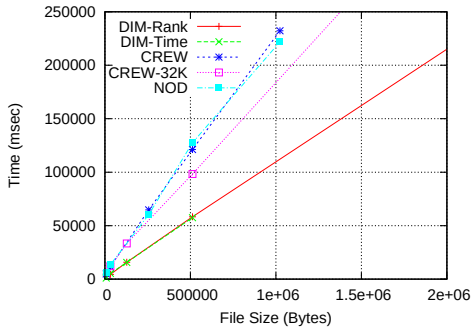


Fig. 6. Performance with Increasing Data Size

3) Effect of File Size: Fig-6 shows the effect when the total data to be disseminated is increased. DIM-Rank has a clear superior performance over all the other protocols. Moreover, as the data size is increased, the gap widens between DIM-Rank and the other protocols making it very desirable. However, note that when data size increases, M increases and this in turn increases the computing need on the central scheduler. In our experiments, we ran the scheduler on a powerful machine so that it could always finish its computation before the next cycle. With increasing node size and file size, the computation load can increase dramatically. Thus, while DIM-Rank may be a very good heuristic for disseminating medium amounts of data to hundreds (or even thousands) of peers, the cost justification for DIM-Rank has to be evaluated carefully for a particular application needs.

V. Concluding Remarks

In this paper, we presented two new heuristics, DIM-time and DIM-Rank for the flash dissemination problem in heterogeneous networks. These heuristics were developed using broadcast theory. Of the two, DIM-Rank offers much lower dissemination times across a variety of metrics. In general, the centralized approaches fare better than the randomized, local-knowledge, decentralized protocols. DIM-Rank will find most use when low dissemination time is of utmost necessity and a centralized coordinator who has global knowledge can be used. The cost of computing the schedule in the central coordinator, however, is non-trivial. It is atleast $O(NLOgN)$ and can be as bad as $O(M * NLogN)^2$ in the worst case. An interesting course of future work would be to investigate approximation techniques that achieve the same

effect as DIM-Rank without the high computation costs.

REFERENCES

[1] Bittorrent: http://bitconjurer.org/bittorrent/.

[2] Shakecast: http://earthquake.usgs.gov/resources/software/shakecast//.

[3] AGARWAL, S., RAMAMRITHAM, K., AND SHAH, S. Construction of temporal coherency preserving dynamic data dissemination network. In *Proc. of RTSS* (2004).

[4] BIRMAN, K. P., HAYDEN, M., OZKASAP, O., XIAO, Z., BUDIU, M., AND MINSKY, Y. Bimodal multicast. In *ACM TOCS* (1999).

[5] CASTRO, M., DRUSCHEL, P., KERMARREC, A.-M., NANDI, A., ROWSTRON, A., AND SINGH, A. Splitstream: High-bandwidth multicast in a cooperative environment. In *SOSP* (2003).

[6] DESHPANDE, M., XING, B., LAZARDIS, I., HORE, B., VENKATASUBRAMANIAN, N., AND MEHROTRA, S. Crew: A gossip-based flash-dissemination system. In *ICDCS* (2006).

[7] FARLEY, A. M. Broadcast time in communication networks. In *SIAM Journal on Applied Mathematics* (1980), vol. 39.

[8] FERNANDESS, C., AND MALKHI, D. On collaborative content distribution using multi-message gossip. In *IPDPS* (2006).

[9] GANESAN, P., AND SESHADRI, M. On cooperative content distribution and the price of barter. In *ICDCS* (2005).

[10] KARP, R., SCHINDELHAUER, C., S.SHENKER, AND VOCKING, B. Randomized rumor spreading. In *IEEE Symposium on Foundations of Computer Science (FOCS) 2000.* (2000).

[11] KHULLER, S., AND KIM, Y.-A. On broadcasting in heterogeneous networks. In *ACM-SIAM Symposium on Discrete algorithms* (2004).

[12] KNUTSSON, B., LU, H., XU, W., AND HOPKINS, B. Peer-to-peer support for massively multiplayer games. In *IEEE INFOCOM* (2004).

[13] KOSTIC, D., RODRIGUEZ, A., ALBRECHT, J., AND VAHDAT, A. Bullet: High bandwidth data dissemination using an overlay mesh. In *Usenix Symposium on Operating Systems Principles (SOSP)* (2003).

[14] YANG-HUA CHU, SANJAY G. RAO, S. S., AND ZHANG, H. A case for end system multicast. In *Measurement and Modeling of Computer Systems* (2001).