Using Destination Set Grouping to Improve the Performance of Window-Controlled Multipoint Connections

Shun Yan Cheung[*]
Department of Mathematics and Computer Science,
Emory University,
Atlanta, Georgia 30322
Email: cheung@mathcs.emory.edu
Phone: (404) 727-3823

Mostafa H. Ammar
College of Computing
Georgia Institute of Technology
Atlanta, Georgia 30332

Tech. Report GIT-CC-94-32
August 1994

Abstract

In conventional multicast communication, the source carries a single conversation with all destination nodes. If a node on the path to any destination becomes congested, the throughput to *all* destinations is reduced, thus treating some destination nodes unfairly. We consider a window-controlled multipoint connection and study the use of destination set grouping, where the destination set can be split into disjoint subgroups with the source carrying independent conversations with each subgroup. We present a static grouping heuristic that can obtain near optimum grouping for static network environments and a dynamic grouping protocol which can adjust the grouping and the window sizes per group in response to changing network conditions. The performance of the static grouping heuristic and the dynamic grouping protocol are studied using simulation and compared with single-group multicasting.

---

[*]This author was supported in part by the University Research Committee of Emory University.

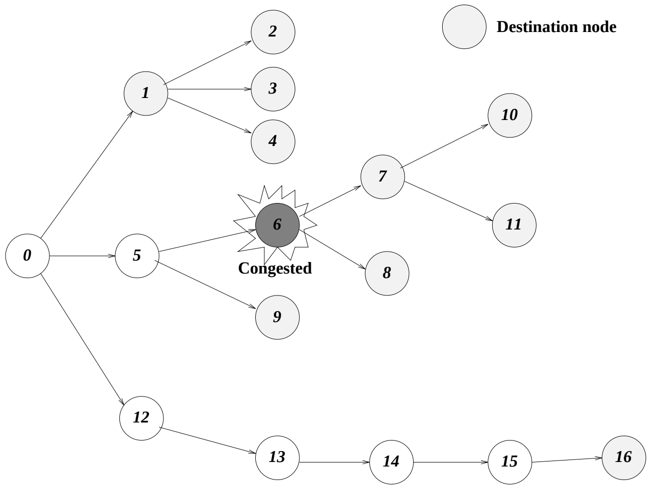
1 Introduction

We study the use of window flow and congestion control mechanisms and how they apply to point-to-multipoint communication. In window-controlled *single destination* communication a distinction is made between static and dynamic window schemes. A *static* window scheme is traditionally used to provide end-to-end flow control, where a source wants to control the rate at which it sends messages to a destination. The assumption is that the message consumption capability of the destination remains relatively stable for the duration of a conversation so it is sufficient to negotiate an initial window size (based on the destination's capacity) and continue to use it throughout. A *dynamic* window scheme provides the capability of changing the transmitter's window size in response to changes of the capacity of the destination and/or of the network path leading to the destination. Because of their ability to adjust to changing traffic situations, dynamic window schemes can be used to avoid and/or control network congestion in addition to performing an end-to-end flow control function. In dynamic window schemes, the source reacts to feedback from the network and increases or decreases its window accordingly. This feedback can be explicit in the form of choke packets [1] or congestion bits [2] returned by nodes that are congested along the path from the source to the destination. This feedback can also be implicit in the form of timeout [3, 4] or excessive packet delay [5].

Both static and dynamic window schemes can be generalized in a straightforward manner to multipoint communication. In this generalization, window sizes are determined by the lowest message consumption capacity available. For instance, in a static window scheme where the maximum window size is negotiated at connection setup, the window size will be chosen according to the slowest destination. If this destination is much slower than the others, it will needlessly reduce the throughput to the other destinations in the multipoint connection.

Similar problems arise when dynamic window sizing is used. Consider the situation where a multicast tree is used to carry traffic from a source to a set of destinations. The source adjusts its window size up or down according to feedback it receives from the nodes in this multicast tree. If one of these nodes is congested it will cause the source to reduce its window size and thus causing a degradation in the throughput experienced by *all* destinations whether or not the congested node lies on the path to the destinations. Figure 1 shows an example of such a scenario. The destination nodes are shaded in the network and node 6 is relatively congested. Multidestination packets will experience excessive queueing delay at node 6 and consequently, the destinations 7, 8, 10 and 11 will cause the source to reduce the window size. As a result, the packet transfer rate to the other destination nodes is also reduced.

FIGURE 1 Example unfairness in multidestination communication



The above type of "unfairness" problem is inherent in multipoint communication where economy (e.g., bandwidth savings) can be derived by collectively treating all the destinations as a single monolithic group. Because paths leading to different destinations may possess differing capabilities, this collective treatment can often lead to unfairness.

In this paper we consider the use of *destination set grouping* where the set of destinations in a single multipoint conversation are split into disjoint groups. The source then carries independent conversations with each group. The same set of messages are used for each conversation and, therefore, the objective of the single multipoint conversation is achieved by the set of independent multipoint conversations. This splitting of the destination set will incur a cost. But, by careful splitting of the original destination set, it can be made to not suffer from unfairness problems. For instance splitting the original destination set {1,2,3,4,7,8,9,10,11,16} in Figure 1 by having nodes 1, 2, 3, 4, 9 and 16 in one group, and nodes 7, 8, 10 and 11 in another will reduce the unfairness effect. It will, however, double the traffic associated with this multipoint conversation carried by some network links because the source has to send each packet twice; once per group.

In previous work, we have investigated similar ideas in the context of an ARQ protocol [6] being used over a broadcast channel. It was shown that destination set splitting can improve throughput despite the increase in communication cost. The basic objective of this paper is to investigate this tradeoff as it applies to a window-controlled multipoint conversation being carried over a packet switched network and to determine situations where it is beneficial to incur the cost of splitting.

In the first part of the paper, we investigate a static grouping and windowing scheme, and consider the performance of a multipoint conversation as a function of the (static) grouping of the destination nodes and the (static) window size used by each group. We show that, despite the increase in the number of packets sent, significant improvement in throughput and average delay can be achieved by splitting the destination set appropriately. In the second part we develop a dynamic destination set grouping protocol that allows dynamic splitting and merging of destination groups, as well as dynamic changing of the window size associated with each group. The performance of this dynamic method is studied extensively through simulation.

The remainder of the paper is organized as follows. In Section 2 we discuss the basic operation of our proposed protocol. Section 3 presents a static grouping method and Section 4 evaluates the performance of static grouping. In Section 5, we develop a dynamic destination set grouping method and in Section 6, we study its behavior and performance in static and dynamic network conditions. Section 7 concludes the paper.

2 Basic Operation

We consider a situation where a multicast routing tree that spans the source node s and all destinations D is used to forward packets. The construction of multicast routing trees has been studied extensively (see for example [7, 8]), and we will therefore assume that the network has an efficient multicast route discovery algorithm in place.

The multicast destination set D is split into a set of disjoint multicast subgroups. A grouping $\mathcal{G} = \{G_1, G_2, \ldots, G_K\}$ of D is a set of subgroups such that

- $\forall\, G_i, G_j, i \neq j$: $G_i \cap G_j = \phi$.
- $\bigcup_{j=1}^{K} G_j = D$.

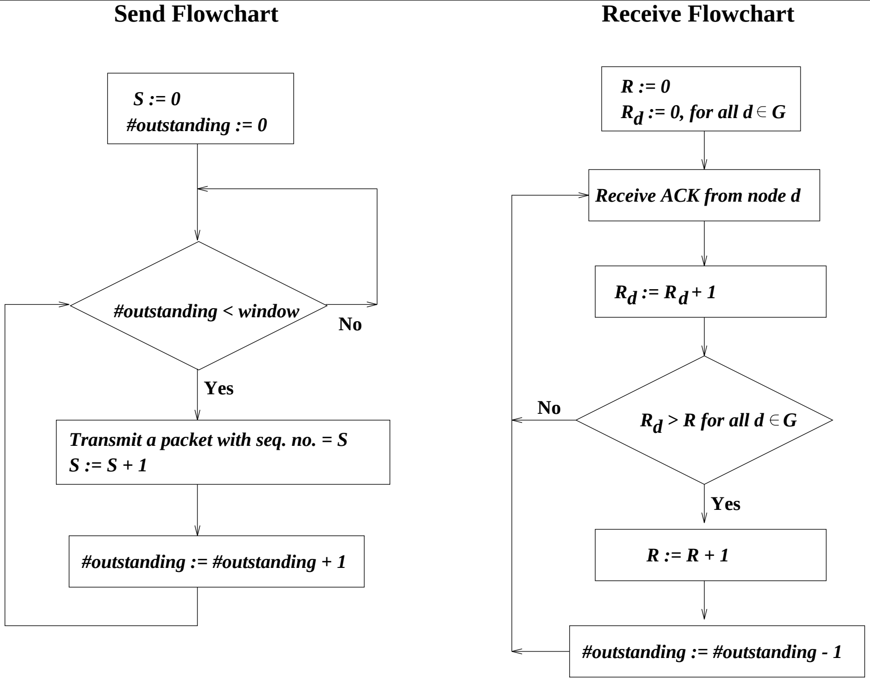
The objective of the source s is to transmit the same set of messages to all destinations. The source carries separate conversations with each multicast subgroup G_i, $i = 1, 2, \ldots, K$.

The multicast tree used to route packets to a subgroup is assumed to be a subtree of the multicast tree that would be used for transmitting to the destination set D in one group.

Window flow control is used and the source always has new packets to send but will only do so to a particular subgroup if the corresponding window is open. The destinations return acknowledgements and the window for a particular subgroup is advanced once all destinations in a subgroup acknowledge its receipt.

The window scheme implemented by the source is shown Figure 2. The source maintains a send and receive sequence number for the subgroup (R and S) , and receive sequence numbers for all members of a subgroup (R_d). Notice that the send and receive sequence numbers of different subgroups and nodal receive numbers of different members of a subgroup can be different. We assume that the network is reliable and provides sequenced delivery and, therefore, it is not possible for the source to receive any acknowledgements for packet x after it receives all acknowledgements for packet $x + 1$.

FIGURE 2 Window scheme implemented by source



The performance measure by which we judge a particular point-to-multipoint conversation is the average power [9] P:

$$P = \frac{1}{|D|} \sum_{i \in D}^{N} P(i) \tag{1}$$

where $P(i)$ is the average power of node i, $i \in D$, defined as

$$P(i) = \frac{\gamma_i}{T_i}$$

where γ_i and T_i are the throughput (i.e., the number of packets received by node i per unit time) and the average delay per packet.

3 Static Destination Set Grouping

In this section, we consider the problem of finding the grouping $\mathcal{G}$ of destinations that maximizes the overall power in a network with static load conditions. There are two components to the grouping heuristic we propose. First, a level of performance is associated with each destination. A grouping heuristic then uses this performance information to find a good grouping of the nodes.

3.1 Destination Performance

Intuitively, the optimum grouping is one that puts "similar" nodes in the same subgroup without creating an excessively large number of subgroups. The main problem in finding an optimal grouping is to determine when the performance levels of nodes in a group differ sufficiently so that splitting the group into two (or more) subgroups (and creating more independent multicast conversations) will improve performance. In order to measure "similarity" in performance, we have considered various measures including delay, throughput and power. Although the objective performance measure is average power, in our simulation experiments we have found that grouping heuristics guided by throughput perform best. This is caused by the fact that power varies over a very wide range which can cause the heuristic to create many small groups.

The performance information must be mutually consistent, i.e., the nodal performance must be measured under *similar* operating conditions. For our heuristic we use a measure which we call the *raw throughput*. For destination i, the raw throughput, h_i, is the throughput of source to destination i transmissions under the following conditions:

- The path from source to destination in the multicast tree is used.
- The source is sending packets using a window of size 1. In this way the full effect of the source to destination path length is reflected in the throughput measure.
- The source is transmitting to only this destination. Otherwise, the network traffic conditions are the same as the environment in which the multicast conversation is to operate. (See subsection 4.1 for a model of traffic conditions.)

The raw throughput is used only to guide the grouping heuristic. It is typically different from the *actual throughput*, γ_i (used in power calculations in equation (1)) that is experienced when traffic to all destinations is present and when the window size maybe large than one.

3.2 Grouping Heuristic

The grouping heuristic uses the performance level determined for each node to find a good grouping. The algorithm is recursive. It starts with the destination set in one group and in each recursion, it determines if there is an input group such that, when it is split into two subgroups, will improve average power. If there is no candidate group for splitting, the procedure terminates, otherwise the method is invoked recursively using as input the grouping that results after splitting a candidate group. The heuristic is based on a number of *grouping criteria* which are guidelines for splitting the initial destination set. We first present the grouping criteria and then discuss the static grouping heuristic. Note that the criteria are only a set of guidelines to judge if a grouping of nodes is favorable and they do not constitute necessary nor sufficient conditions for optimality.

The first grouping criterion ensures that dissimilar nodes are not put in the same subgroup and it is stated as follows:

C1: Let $\mathcal{G}$ be a grouping and, G_1 and G_2 are subgroups in $\mathcal{G}$. The throughput of *all* nodes in G_1 must be strictly less than that of *any* node in G_2.

If the nodes in D are ordered by their throughput, criterion C1 is then satisfied if each subgroup consists of a subrange of nodes. To simplify the remainder of the discussion, we will re-number the nodes in D as 1, 2, ..., $\mid D \mid$ such that $h_i \geq h_j$ for $i < j$ and $i, j = 1, 2, \ldots, |D|$ where h_i and h_j are the raw throughput of nodes i and j, respectively.

To ensure that nodes of similar performance level are grouped together, we require that the *normalized standard deviation* of throughput of each group is relatively small. The normalized standard deviation is the ratio of the standard deviation and its mean:

C2: For every subgroup G in grouping $\mathcal{G}$:

$$\frac{\sigma_h(G)}{h_{\text{ave}}(G)} \quad < \quad \tau_{sim} \tag{2}$$

where $h_{\text{ave}}(G)$ and $\sigma_h(G)$ are the sample mean and the sample standard deviation of the raw throughputs of the destinations in G, respectively.

The threshold value τ_{sim} is a design parameter and a smaller (larger) value for τ_{sim} will allow the creation of groups with smaller (larger) differences in throughput. The choice of τ_{sim} is discussed later.

Although criterion C2 asserts that only nodes of similar performance are to be grouped together, it does not necessarily guarantee that they will be put in the same group. In fact, the grouping $\mathcal{G} = \{\{1\}, \{2\}, \ldots, \{\mid D \mid\}\}$ (each subgroup is a singleton node) satisfies C2 regardless of whether the nodes have similar or different performance levels. The third and final criterion is designed to prevent nodes of similar performance levels from being put in different groups. It states that the performance levels of subgroups must be "separated" by a sufficiently large amount:

C3: For every pair of subgroups G_1 and G_2 in grouping $\mathcal{G}$:

$$\delta(G_1, G_2) \quad > \quad \tau_{dissim} \tag{3}$$

where $\delta(G_1, G_2)$ is a *distance function* on the subgroups G_1 and G_2.

The parameter τ_{dissim} is another design parameter of the method. A smaller (larger) value for τ_{dissim} will permit the creation of groups with smaller (larger) distances in performance. We discuss possible values for this threshold below. The distance function $\delta(G_1, G_2)$ measures the difference in performance levels of members in G_1 and G_2. Many functional forms are possible to measure the distance between two groups. We found that the following normalized raw throughput difference function works well:

$$\delta(G_1, G_2) = \frac{\min(h_{G_2}) - \max(h_{G_1})}{\left(\frac{\min(h_{G_2}) + \max(h_{G_1})}{2}\right)} \tag{4}$$

where

$$\min(h_G) = \min_{i \epsilon G} h_i \tag{5}$$

$$\max(h_G) = \max_{i \epsilon G} h_i \tag{6}$$

and where G_1, and G_2 are such that $\min(h_{G_2}) > \max(h_{G_1})$. (Recall that we assume that destinations are ordered according to their raw throughput and that each subgroup is a subrange of destinations.)

Grouping criteria C1, C2 and C3 will remove a large number of undesirable groupings, however, they will not determine a unique grouping. The promising groupings that satisfy criteria C1, C2 and C3 must be ranked in some way so that the most promising grouping can be determined. In general, different groupings have a different number of groups and ranking the promising groupings is difficult. Fortunately, due to the recursive nature of our grouping heuristic, we need only to compare groupings that consists of exactly two subgroups. For such groupings a ranking function is easy to define and we use a combination of the normalized standard deviation and the distance function δ. We define the *in-homogeneity* of a two-subgroup grouping as follows:

$$\Gamma(\{G_1, G_2\}) = \frac{\sigma_h(G_1)}{h_{\text{ave}}(G_1)} + \frac{\sigma_h(G_2)}{h_{\text{ave}}(G_2)} - \delta(G_1, G_2) \tag{7}$$

Notice that $\Gamma(\{G_1, G_2\})$ is minimal if nodes in the same subgroup have similar relative performance *and* nodes in different subgroups have larger differences in relative performance. The in-homogeneity measure will be used to determine the most favorable grouping among those that satisfy grouping criteria C1, C2 and C3.

Figure 3 presents our static grouping heuristic and shows how the criteria above are used.

3.3 Determining Heuristic Parameters

The performance of the static grouping heuristic depends on the choice of the thresholds τ_{sim} and τ_{dissim}. Through numerical experiments using different functions for τ_{sim} and τ_{dissim}, we found that the following values work well:

$$\tau_{sim} = f_1 \frac{\sigma_h(G)}{h_{\text{ave}}(G)} \tag{8}$$

$$\tau_{dissim} = f_2 \tag{9}$$

FIGURE 3 Group splitting heuristic

```
Split(G, τ_sim, τ_dissim)
    /* G = {n_1, ..., n_K} and h_{n_1} ≥ ... ≥ h_{n_K} */
  begin
  Find largest i such that:
    (a) G_1 = {n_1, ..., n_i}, G_2 = {n_{i+1}, ..., n_K} and
    (b) σ_h(G_1) / h_ave(G_1) < τ_sim and
    (c) δ(G_1, G_2) > τ_dissim and
    (d) Γ({G_1, G_2}) is minimal
  if successful then
    begin
    Split(G_1, τ_sim, τ_dissim);
    Split(G_2, τ_sim, τ_dissim);
    end;
  else
    Return(G);
  end;
```

where G is the input destination set to the heuristic. The parameters f_1 and f_2 are constants and they are determined empirically. If f_1 is too small or too large, then the heuristic may be too conservative (does not split off nodes affected by congestion) or too aggressive (forms too many groups). Similarly, the heuristic can be too aggressive and conservative when f_2 is large and small, respectively. In particular, the static grouping heuristic was able to find a good grouping in all test cases when $f_1 \geq 0.33$ and $f_2 \in (0.30, 0.59)$, but not for every combination of f_1 and f_2 in these ranges. In future research we plan to consider various approaches for a more systematic determination of f_1 and f_2.

4 Evaluating the Static Grouping Heuristic

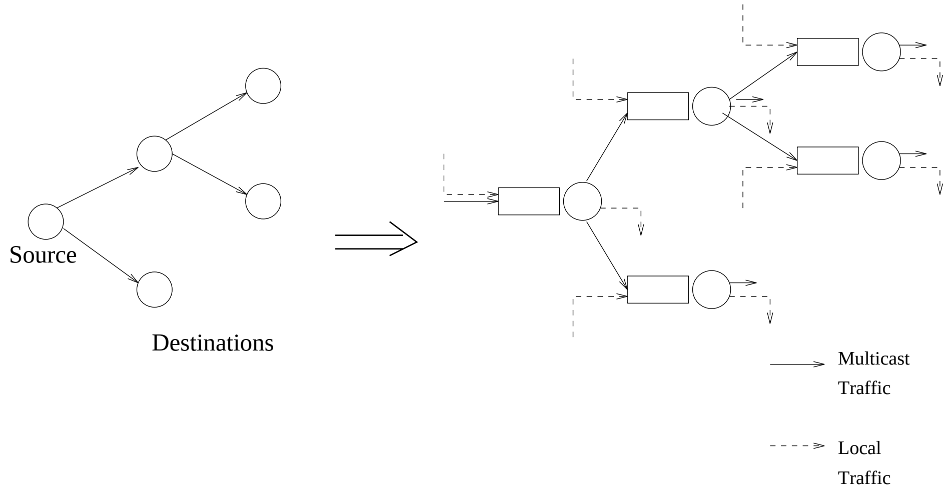
4.1 Model

Our evaluation of the static grouping heuristic is based on the simulation of a model that is based on the one described in [10]. The network is represented by a graph $G(V, E)$ where V and E are the set of nodes and links, respectively. The network is assumed to provide error-free and sequenced delivery of messages. The node $s \in V$ is the source and the subset

$D \subseteq V$ is the set of destinations.

The multicast routing tree is modelled as a queueing network with forking (see for example [11]). Each node (including the leaves) is modelled as a FCFS single server queue with deterministic service times. The output of a queue modelling a node with more than one child is forked and represents arrivals to the queues modelling the child nodes. This is illustrated in Figure 4. The service time of each node is used to represent the processing time of arriving packets as well as the transmission time of the link leading to the node. We will model congestion by increasing the load on specific queues. A congested queue will, therefore, represent the congestion of the processor queue at the corresponding node and/or the congestion of the link queue corresponding to the link from the parent of the node. The service time at node i, for $i = 1, 2, \ldots, N$, is assumed to be deterministic with rate equal to μ_i packets per time unit.

FIGURE 4 Queueing Model



As in [10], we assume that the time to return acknowledgements is negligible (or subsumed in the processing time at a destination).

Packets arriving at a queue in our model can arrive from two sources:

- they can be multicast packets from the source s enroute to a destination.
- they can be packets from sources other than s.

The arrivals from sources other than s are called "local traffic" and the node will discard them after processing. The arrival rate of such traffic to a node i is assumed to be Poisson

FIGURE 5 Multicast Routing Tree #1

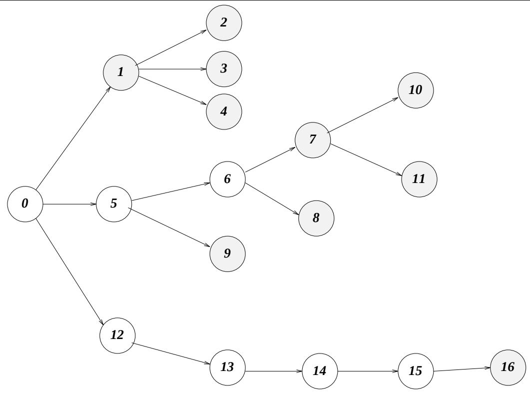
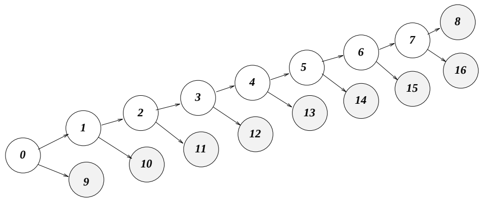


FIGURE 6 Multicast Routing Tree #2



with rate λ_i, for $i = 1, 2, \ldots, N$.

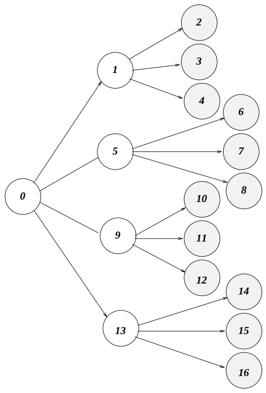
4.2 Numerical Examples

The grouping heuristic, with parameters $f_1 = 0.35$ and $f_2 = 0.50$, was used to find groupings in the networks shown in Figures 5, 6 and 7. Routing tree #1 is a somewhat arbitrary tree. Routing trees #2 and #3 represent two extreme forms of routing trees: the trees are narrow and deep, and broad and shallow, respectively. The raw throughput data for the destination nodes used by the static grouping heuristic are obtained by simulation.

Three sets of experiments are performed for each routing tree. In the first set, we place one congested node in the network, in the second set we add another one, and in the last set one of the congested nodes is made even more congested. Table 1 shows the list of nodes in the corresponding networks that are congested. All nodes in the network have equal service rate $\mu = 100$ packets per time unit. Non-congested nodes have local arrival rates of $0.2\times\mu$. The local traffic rates for congested nodes are as shown in Table 1. Table 1 also shows the groupings determined by the static grouping heuristic for the various systems.

The performance of the multicast connection using the groupings determined by the

FIGURE 7 Multicast Routing Tree #3



heuristic is studied in Table 2. The table also shows the optimum window sizes for each subgroup. These window sizes are determined by enumeration and simulation of various window size combinations and choosing the ones that give the maximum power. The power obtained using destination set splitting is compared with that obtained when no destination set splitting is used. It is clear that in these cases, the added cost of destination set splitting is outweighed by its benefit.

5 Dynamic Destination Set Grouping

We now turn our attention to the development of a dynamic protocol that adjusts the destination set grouping, as well as window sizes, in response to changing network conditions. Our proposed protocol's basic structure is shown in Figure 8. At any point in time, the protocol is maintaining K multicast conversations with K subgroups of the destination set. The window size for each subgroup may be different and is subject to adjustment as traffic conditions change. The protocol "visits" these subgroups in a round robin fashion. On each visit, a packet may be transmitted to the subgroup, any received acknowledgements will be processed and other processing will be performed as discussed later.

After a round of subgroup visits is completed, the protocol checks whether it is time to consider splitting or merging of subgroups. If so, the splitting or merging of groups is processed and another round of visits to (the now possibly revised) set of subgroups is begun. In later subsections we will discuss how the decision time is determined and how split and merge decisions are made. Observe that in our proposed protocol a maximum of one split or merge decision is taken. Also, at each decision point, we only consider splitting one group

Case	Network	Cong. Nodes	Grouping
#1	#1	16	G_1={1,2,3,4,7,8,9,10,11}, G_2={16}
#2	#1	6, 16	G_1={1,2,3,4,9}, G_2={7,8,10,11,16}
#3	#1	6, 16†	G_1={1,2,3,4,9}, G_2={7,8,10,11}, G_3={16}
#4	#2	6	G_1={9,10,11,12,13,14}, G_2={8,15,16}
#5	#2	6, 13	G_1={9,10,11,12,14}, G_2={8,13,15,16}
#6	#2	6, 13†	G_1={9,10,11,12,14}, G_2={8,15,16}, G_3={13}
#7	#3	1	G_1={6,7,8,10,11,12,14,15,16}, G_2={2,3,4}
#8	#3	1, 6	G_1={7,8,10,11,12,14,15,16}, G_2={2,3,4,6}
#9	#3	1, 6†	G_1={7,8,10,11,12,14,15,16}, G_2={2,3,4}, G_3={6}
† Local arrival rate is 0.95 $\times\mu$, all other congested nodes: local arrival rate is 0.8 $\times\mu$			

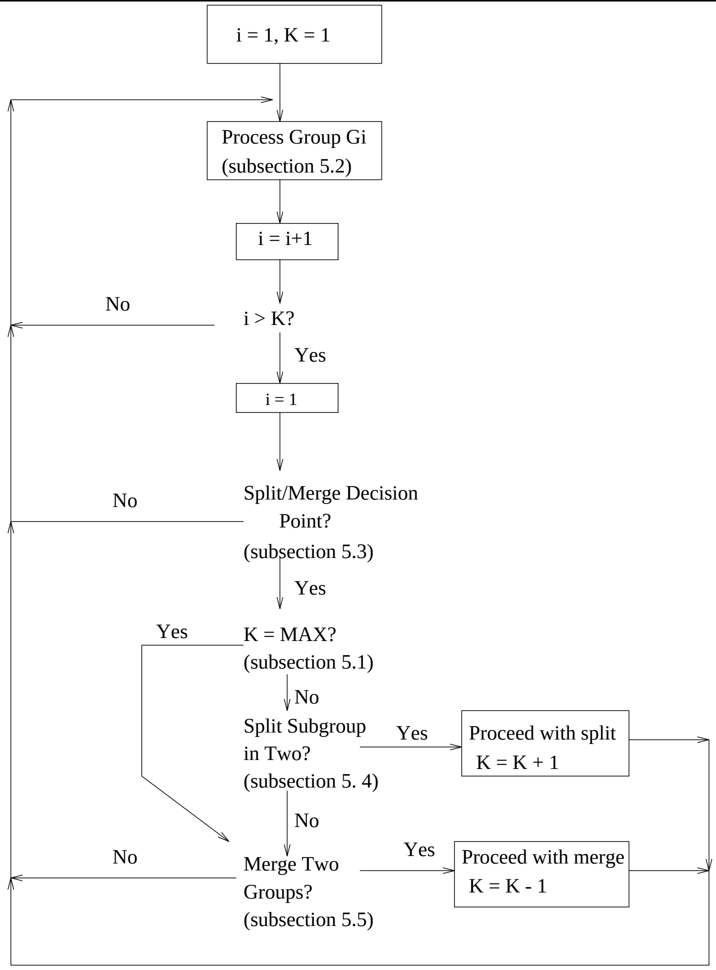
Table 1: Network configurations and grouping obtained by the static grouping heuristic

Case	$W(G_1)$	$W(G_2)$	$W(G_3)$	Power	$W(G)$	Power	Ratio
#1	5	1	N/A	999.93	4	478.65	2.09
#2	4	1	N/A	721.88	4	339.60	2.13
#3	3	1	1	662.11	3	104.57	6.33
#4	7	1	N/A	635.19	5	329.63	1.93
#5	7	1	N/A	578.27	6	287.60	2.01
#6	7	1	1	528.91	3	87.82	6.02
#7	4	1	N/A	927.96	2	467.01	1.99
#8	4	1	N/A	842.50	6	393.51	2.14
#9	4	1	1	771.73	8	108.82	7.09

Table 2: Average power of multicast connections for destination set grouping and single destination group

in two or merging two groups into one. As time progresses, however, these binary splits and merges can accumulate to result in arbitrary groupings of the destination set. Finally, we note that splitting of groups is not considered if the protocol is already maintaining some maximum number of groups MAX (see discussion in subsection 5.1 below).

FIGURE 8 The basic destination set grouping protocol



5.1 Constraining Allowed Groupings

Before proceeding with a discussion of the details of our protocol, we examine an important issue regarding multicast addressing and routing in the adaptive grouping environment. At any point in time, the source will be maintaining a set of multicast conversations, each with a subset of the initial destination set. If group addressing is used (i.e., all members of a single subgroup are made to recognize a single subgroup address), then we must have a separate multicast address designated for every possible subset of the original destination set. This is because the grouping that results from unconstrained splitting and merging can be arbitrary. For a destination set of size $|D|$, there are on the order of $2^{|D|}$ such subsets. In addition to setting up the addresses to be recognized by the destinations, the switches in the network need to know how to route to all these addresses ahead of time. Clearly, setting up these addresses "on the fly" as groups are merged and split is not an efficient approach.

To address this we impose some constraints on the allowable destination set groupings. The multicast destination set, D, is divided into a set of *basic* groups $\mathcal{B} = \{B_1, B_2, \ldots, B_L\}$ such that:

$$\begin{aligned} D &= B_1 \cup B_2 \cup \ldots \cup B_L \quad \text{and,} \\ B_i \cap B_j &= \phi, \qquad\qquad\qquad \text{for } i,j = 1,2,\ldots,L \text{ and } i \neq j \end{aligned}$$

The allowable destination set groupings are of the form $\mathcal{G} = \{G_1, G_2, \ldots, G_K\}$ where each G_i, $i = 1,2,\ldots,K$, is a union of a number of basic groups and where K is less than some value MAX. (Recall that the G_i's are also mutually exclusive.)

For example, suppose $D = \{1,2,\ldots,7\}$, $MAX = 2$ and the basic groups are $B_1 = \{1,2,3\}$, $B_2 = \{4,5\}$ and $B_3 = \{6,7\}$, then legal groupings are for example: $\mathcal{G}_1 = \{\{1,2,3\},\{4,5,6,7\}\}$ and $\mathcal{G}_2 = \{\{1,2,3,6,7\},\{4,5\}\}$. However, the following groupings are illegal: $\mathcal{G}_3 = \{\{1,2,3,4\},\{5,6,7\}\}$ (neither subgroups are unions of basic groups) and $\mathcal{G}_4 = \{\{1,2,3\},\{4,5\},\{6,7\}\}$ (number of subgroups exceeds MAX).

We assume that all possible legal groupings for a given $\mathcal{B}$ and MAX have been set up in the network [1]. The number of groups that need to be set up is given by

$$N_{groups} = S(L,1) + S(L,2) + \ldots + S(L, MAX)$$

where $S(L,i)$ is a Stirling number of the second kind [12][2]. Table 3 shows the number of groups required for some representative values of L and MAX. Notice that the number of different groupings is unchanged if $MAX \geq L$; this is because the largest possible number of subgroups is L.

[1] "Setting up" a group means all its members will recognize the multicast group address and that switches in the network have entries in their routing tables for the group.

[2] $S(n,i)$ counts the number of ways to put n distinct objects in i undistinguishable boxes such that no box is empty.

	Number of Basic Groups, L				
MAX	2	3	4	5	6
2	2	4	8	16	32
3	2	5	14	41	122
4	2	5	15	51	187
5	2	5	15	52	202
6	2	5	15	52	203

Table 3: Number of multicast groups that need to be set up as a function of L and MAX

5.2 A Multipoint Generalization of the Binary Feedback Scheme

Our protocol allows for dynamic window sizing (as well as dynamic grouping). The protocol used to control the window size with one multicast subgroup is an extension of the Binary Feedback (BF) congestion avoidance protocol [2]. The BF scheme uses an indication bit in the network layer header to signal congestion. Messages traversing the network towards a destination that encounter a congested node have the congestion indication bit set. This bit is then copied in the acknowledgement packet being returned to the source. A node is considered congested if when a packet arrives, the average queue length (over current and previous busy periods) at that node exceeds one message. The source uses the returned congestion indication bits to decide on when to increase or decrease the window size.

The BF protocol operates in *window sizing* cycles. The number of acknowledgements processed in cycle i is equal to $W_{i-1} + W_i$ where W_{i-1} and W_i are the window sizes used in cycle $i-1$ and i, respectively. The first W_{i-1} acknowledgements received are discarded and the next W_i acknowledgements are used to determine whether the window size should be increased or decreased. To distinguish the two types of acknowledgements received during a window sizing cycle, we will call the last W_i acknowledgements *meaningful*. If more than half of the congestion indication bits in the meaningful acknowledgements are set, the decision is to reduce the window size and otherwise, the window size is increased. The BF protocol uses an additive increase and multiplicative decrease scheme to update the current window size [13]. After the window is updated to W_{i+1}, the BF protocol starts a new cycle in which $W_i + W_{i+1}$ acknowledgements are processed.

Our multipoint generalization of the BF scheme operates in a similar manner. New packets are transmitted to members of subgroup G as long as the number of outstanding (unacknowledged) packets is less than the current window size for that subgroup. A packet is only considered acknowledged if all members in G have received it. The window sizing algorithm is similar to the BF protocol and differs only in the way that it filters the congestion signals. The protocol will decrease the window if a majority of the meaningful acknowledgements from *any* one node in G have the congestion-bit set, and otherwise the window size will be increased.

The multipoint dynamic window sizing protocol is implemented for each group on consecutive visits to the same subgroup in the overall protocol shown in Figure 8. On each visit:

1. acknowledgements received since the last visit are processed and the window is advanced if possible,
2. at most one packet is transmitted to the subgroup if the window is "open" and if a packet is available for transmission, and
3. if this is the end of a window sizing cycle, consideration is given to adjusting the window size.

5.3 Split/Merge Decision Point

A decision on whether groups should be split and/or merged involves the consideration of the current state of the multipoint conversation. Details of what is examined are discussed below. For efficiency and stability reasons we consider examining such a state periodically. We propose to examine the state of the conversation whenever each multicast subgroup has completed at least R window sizing cycles (as defined in subsection 5.2) since the last split/merge decision. Note that the different multicast subgroups will complete cycles at different rates since they may have different window sizes. The value of R will influence the stability of the split/merge decisions. A small R will result in frequent splitting and merging of groups, whereas a large value of R will cause the protocol to react too slowly to changes in the network traffic conditions.

To aid in the split/merge decisions, the following information is maintained for each destination, d:

- N_d, the total number of acknowledgements received from node d.

- S_d, the cumulative delay experienced by packets sent to d. The delay for each packet is estimated by subtracting the transmission time from the current time value and then dividing the result by two.
- C_d, number of acknowledgements from d which have their congestion indication bit set.

Initially, the above parameters are set to zero. At a split/merge decision point, they are used to compute throughput and power estimates as discussed below and reset to zero.

5.4 Group Splitting

A subgroup G should be split into two subgroups if some (but not all) its members are experiencing congestion that is severe enough to result in widely dissimilar performance levels between affected and non-affected nodes. We consider a node d to be affected by congestion if $C_d > \alpha \times N_d$, where α is a fraction between 0 and 1. The choice for α will affect the behavior of the split algorithm. Using this information, members of a group G are categorized into two subsets: `CongSet`, the nodes that are affected by congestion and `NCongSet`, the nodes that are not affected by congestion.

The difference in performance is severe if `Dissimilar(CongSet, NCongSet)` $>$ `SPLIT_THR`, where `SPLIT_THR` is a design parameter of the protocol. We will discuss the `Dissimilar` function below. Because we can only form subgroups that are a union of our basic subgroups (see subsection 5.1) we actually test to see if `Dissimilar(CongSet+,` $G-$`CongSet+)` $>$ `SPLIT_THR`, where `CongSet+` $=$ `CongSet` with non congested destinations added to form a union of basic subgroups.

The split criterion cannot use throughput as a guideline because all nodes in the same group have equal throughput. Nodes that are affected by congestion will have larger delay and consequently, lower power (ratio of throughput and delay). The simulation study shows that the following `Dissimilar` function, which measures a "power distance", works well:

$$\texttt{Dissimilar}(G_1, G_2) = \frac{|\min(P_{G_2}) - \min(P_{G_1})|}{\left(\frac{\min(P_{G_2})+\min(P_{G_1})}{2}\right)}$$

where $\min(P_{G_1})$ and $\min(P_{G_2})$ are the minimum power of nodes in G_1 and G_2, respectively.

All current subgroups are examined to see if they are candidates for splitting. If there are more than one split candidate group, then the one with the largest difference in performance (i.e., value of the `Dissimilar` function) is split. If a split candidate is found, the two new subgroups start out using the window size of the original group.

5.5 Group Merging

The merging of two subgroups is considered at the split/merge decision point if no split candidate was found or the number of groups in the current grouping is equal to the maximum. The protocol will consider every possible pair of groups G_1 and G_2 in the current grouping and a pair is a candidate for merging if `Similar(`G_1`,`G_2`)` $<$ `MERGE_THR` where `MERGE_THR` is another design parameter of the protocol. The `Similar` function is discussed below. If there are more than one merge candidate pair, then the pair with the smallest difference in performance (i.e., the smallest value for the `Similar` function) are merged. Note that there is no danger of forming an illegal group by merging existing legal subgroups. If a merge candidate pair is found, the new merged group starts out using the smaller window size of the merged subgroups.

We use the following expression, which measures a "throughput distance", for the function `Similar`:

$$\texttt{Similar}(G_1, G_2) = \frac{|\gamma_{G_2} - \gamma_{G_1}|}{\left(\frac{\gamma_{G_2}+\gamma_{G_1}}{2}\right)}$$

Where $\gamma_G = \gamma_i$ for any $i \epsilon G$. (Note that members of the same group have the same throughput.) The reasoning behind the above merge criterion is the following. The rates at which packets are sent to members in groups G_1 and G_2 are dictated by γ_{G_1} and γ_{G_2}, respectively. Therefore, nodes in the faster group will not be slowed down too much by members of a slower group if the throughputs of the two groups are similar.

5.6 Sequence Numbering

Recall that the source maintains a receive sequence number for each destination and one for each subgroup (see Figure 2). The subgroup sequence number is used to move the window while the destination sequence number marks the last packet received by a particular destination. As groups are split and merged, the subgroup sequence numbers are modified to reflect the split or the merge. The individual destination receive sequence numbers are increased only as a result of acknowledgment reception.

The rules by which the subgroup receive sequence number is modified are as follows:

- When a group is split into two subgroups, the receive sequence number for each of the new subgroups is set to the smallest of the destination sequence numbers of members of the group.
- When two subgroups are merged, the new group's receive sequence number is set to the smaller of the original two subgroups' receive sequence numbers.

Example: Consider a destination set $D = \{1, 2, 3\}$. Suppose that at some point in time the destinations get split because node 1 is congested into $G_1 = \{1\}$ and $G_2 = \{2, 3\}$. Now suppose that some time later node 1 is back to normal and needs to be merged back with nodes 2 and 3. If at that point the destination sequence numbers are as follows: $R_1 = 100$, $R_2 = R_3 = 400$ then the merged group will have a group receive sequence number equal to 100. If, after an additional 100 packets are sent, node 1 gets congested again and is split off, then G_1 will have a receive sequence number of 200, while G_2 will have a receive sequence number of 400.

Note that members of the faster group of the merged subgroups will temporarily receive duplicate packets which do not contribute to the throughput of a node. This represents a cost for merging two groups. The effect of this cost is primarily a function of how much longer the conversation will last. This type of information is typically not available and we, therefore, do not use it in making merge decisions. If, however, there is information about how many more packets remain to be transmitted in a conversation, our heuristic can be modified to incorporate such information.

5.7 Determining Dynamic Protocol Parameters

The performance of the dynamic destination set grouping protocol depends on the choice of R (see section 5.3), α (see section 5.4), and the thresholds `SPLIT_THR` and `MERGE_THR`. The dynamic protocol can become unstable (e.g., merge and split the same groups over and over again) if these parameters are not set correctly.

First we observe that the thresholds `SPLIT_THR` and `MERGE_THR` should be chosen to define a hysteresis algorithm to prevent split/merge cycling. Through simulation experiments using different values, we found that R must be set to at least 21 to reduce erroneous split and merge decisions due to statistical effects and we have used $R = 29$ in our numerical examples to further reduce the likelihood of erroneous split/merge decisions. We also found that for $\alpha = 0.50$, the split algorithm can sometimes categorize a congested node as non-congested and vice versa. This may be due to the fact that the window sizing scheme (subsection 5.2) uses the same fraction to indicate congestion. Consequently, we have experimented with higher values for α and found that the split algorithm performs well for $\alpha = 0.55$. For this particular α value, the split algorithm is overly aggressive when SPLIT_THR ≤ 0.45 (it splits off too many groups) and overly conservative when SPLIT_THR ≥ 0.60. We used SPLIT_THR $= 0.55$ (a reasonably conservative threshold) in our numerical experiments. The merge algorithm is too conservative when MERGE_THR ≤ 0.20 and we used (another reasonably conservative threshold) MERGE_THR $= 0.25$. The parameter settings are summarized in Table 4.

α	R	SPLIT_THR	MERGE_THR
0.55	29	0.55	0.25

Table 4: Parameter settings for the dynamic destination set grouping protocol

6 Evaluating the Dynamic Grouping Heuristic

For our evaluation of the dynamic grouping heuristic we use the same simulation model outlined in subsection 4.1. As before our simulation assumes that the source never runs out of packets to multicast.

We use the same networks shown in Figures 5, 6, and 7. The basic groups used for each network are shown in Table 5. We use different basic groups for different networks and the groups are chosen such that members of a group have multiple common parent nodes. This is because if a node becomes congested, all its children nodes will be affected by the congestion. Since we do not know in advance which nodes will become congested, these basic groups will increase the likelihood that members of the same basic group are affected by congestion reducing the chance of unfairness. The maximum number of groups MAX is set to the number of basic groups in all cases. The table also shows the number of multicast groups that need to be set up in advance prior to running the dynamic grouping protocol.

We first study the performance of the dynamic protocol under static network conditions. The objective is to ensure that the protocol will result in stable behavior in such situations.

The static network conditions we experiment with are the same ones shown in Table 1.

Network	Basic groups	L	N_{trees}
#1	$\mathcal{B}$ = {{1,2,3,4}, {7,8,9,10,11}, {16}}	3	5
#2	$\mathcal{B}$ = {{9,10,11}, {12,13,14}, {8,15,16}}	3	5
#3	$\mathcal{B}$ = {{2,3,4}, {6,7,8}, {10,11,12}, {14,15,16}}	4	15

Table 5: Basic groups used in the networks

Case	Time	Grouping	Avg. Power
#1	18.9	G_1={1,2,3,4,7,8,9,10,11}, G_2={16}	897.4
#2	23.5	G_1={1,2,3,4}, G_2={7,8,9,10,11,16}	599.5
#3	14.2	G_1={1,2,3,4}, G_2={7,8,9,10,11,16}	545.9
	53.4	G_1={1,2,3,4}, G_2={7,8,9,10,11}, G_3={16}	

Table 6: Behavior of the dynamic protocol in static networks

Table 6 summarizes the behavior of the dynamic protocol in static network conditions. Only the results for cases #1, #2, and #3 are shown. The behavior in other cases is similar. The initial grouping is the single group containing all multicast destinations. In the cases where congested nodes have $0.8 \times \mu$ arrivals per time unit (i.e., cases #1, and #2), the dynamic protocol splits the initial group into the grouping shown in the table at the time indicated in the second column. Notice that the grouping resulting from the dynamic protocol may be different from the one determined by the static grouping heuristic (see Table 1) for the same test cases due to the constraint that groups in the dynamic protocol must be unions of basic groups. In case #3, where one node is more severely congested than another, the dynamic protocol first splits the initial grouping into two groups G_1 and G_2 where the former contains only non-congested nodes and the latter is a superset of the congested nodes (that satisfies the basic group constraint). The source now carries a high and a low power conversation with G_1 and G_2, respectively. The protocol subsequently divide G_2 further if over time the more congested node reports congestion *and* the less congested node does not in some split/merge decision period. This is the case in test case #3.

The last column in Table 6 shows the performance of the dynamic protocol which is the average of the power of the individual conversation with the multicast destinations. Comparing to the results in Table 1, we can see that the average power achieved by the dynamic protocol is significantly better than single-group multicasting and is comparable to the level of performance achieved by the static grouping heuristic.

We next examine the behavior of the dynamic protocol under dynamic network conditions. The network starts out with one permanently congested node and another temporarily congested node. Table 7 summarizes the network conditions we experiment with. The local arrival rate of the permanently congested node is $0.8 \times \mu$ and that of the temporarily congested node is $0.8 \times \mu$ during the interval (100,400) and is equal to $0.2 \times \mu$ at other times.

Test case	Network	Perm. cong. node	Temp. cong. node
#1	#1	16	1
#2	#1	16	6
#3	#2	6	1
#4	#2	6	3
#5	#3	1	5
#6	#3	1	10

Table 7: Dynamic network environments

Table 8 shows the behavior of the dynamic protocol in the dynamic network settings shown in Table 7. The table shows the time at which a split or a merge occurs. Recall that the two interesting events, congestion start and end at the temporarily congested node, occur at times 100 and 400, respectively. We can see that despite the fact that the dynamic protocol splits one group or merges two groups at a decision point, a sequence of splits and merges can result in somewhat arbitrary groupings constrained only by the basic groups. Also we observe that the splits and merges occur soon after the congestion event times. Notice also that the dynamic protocol is very stable under dynamic loads and it does not cycle.

Table 9 shows the average performance of the multicast for the duration of the simulation period. For comparison reasons, we also show in Table 9, the average power of the single-group multicasting (using our multipoint generalization of the Binary Feedback scheme to adjust window size) for the same dynamic network conditions. The average performance

Case	Start time	Grouping used
#1	18.9	G_1={1,2,3,4,7,8,9,10,11}, G_2={16}
	110.5	G_1={1,2,3,4,7,8,9,10,11,16}
	145.9	G_1={7,8,9,10,11}, G_2={1,2,3,4,16}
	407.6	G_1={7,8,9,10,11}, G_2={1,2,3,4}, G_3={16}
	413.9	G_1={1,2,3,4,7,8,9,10,11}, G_2={16}
#2	18.9	G_1={1,2,3,4,7,8,9,10,11}, G_2={16}
	104.0	G_1={7,8,9,10,11}, G_2={16}, G_3={1,2,3,4}
	110.3	G_1={7,8,9,10,11,16}, G_2={1,2,3,4}
	423.1	G_1={7,8,9,10,11}, G_2={16}, G_3={1,2,3,4}
	436.4	G_1={1,2,3,4,7,8,9,10,11}, G_2={16}
#3	23.4	G_1={9,10,11,12,13,14}, G_2={8,15,16}
	137.0	G_1={8,9,10,11,12,13,14,15,16}
	445.6	G_1={9,10,11,12,13,14}, G_2={8,15,16}
#4	23.4	G_1={9,10,11,12,13,14}, G_2={8,15,16}
	106.4	G_1={9,10,11}, G_2={12,13,14}, G_3={8,15,16}
	168.5	G_1={9,10,11}, G_2={8,12,13,14,15,16}
	469.6	G_1={9,10,11}, G_2={12,13,14}, G_3={8,15,16}
	539.7	G_1={9,10,11,12,13,14}, G_2={8,15,16}
#5	13.0	G_1={6,7,8,10,11,12,14,15,16}, G_2={2,3,4}
	106.5	G_1={10,11,12,14,15,16}, G_2={2,3,4}, G_3={6,7,8}
	111.6	G_1={10,11,12,14,15,16}, G_2={2,3,4,6,7,8}
	416.8	G_1={10,11,12,14,15,16}, G_2={2,3,4}, G_3={6,7,8}
	420.8	G_1={6,7,8,10,11,12,14,15,16}, G_2={2,3,4}
#6	13.0	G_1={6,7,8,10,11,12,14,15,16}, G_2={2,3,4}
	105.7	G_1={6,7,8,14,15,16}, G_2={10,11,12}, G_3={2,3,4}
	115.4	G_1={6,7,8,14,15,16}, G_2={2,3,4,10,11,12}
	422.3	G_1={6,7,8,14,15,16}, G_2={10,11,12}, G_3={2,3,4}
	426.7	G_1={6,7,8,10,11,12,14,15,16}, G_2={2,3,4}

Table 8: Behavior of the dynamic protocol in dynamic network environments

improvement of the dynamic protocol over single-group multicasting with no destination set splitting is 86%.

	Experiment					
	#1	#2	#3	#4	#5	#6
Dynamic grouping protocol	623.5	590.1	366.6	481.8	606.5	604.6
Single-group using dyn. window	308.3	335.6	195.9	206.5	381.3	387.5

Table 9: Average power with and without dynamic destination set grouping

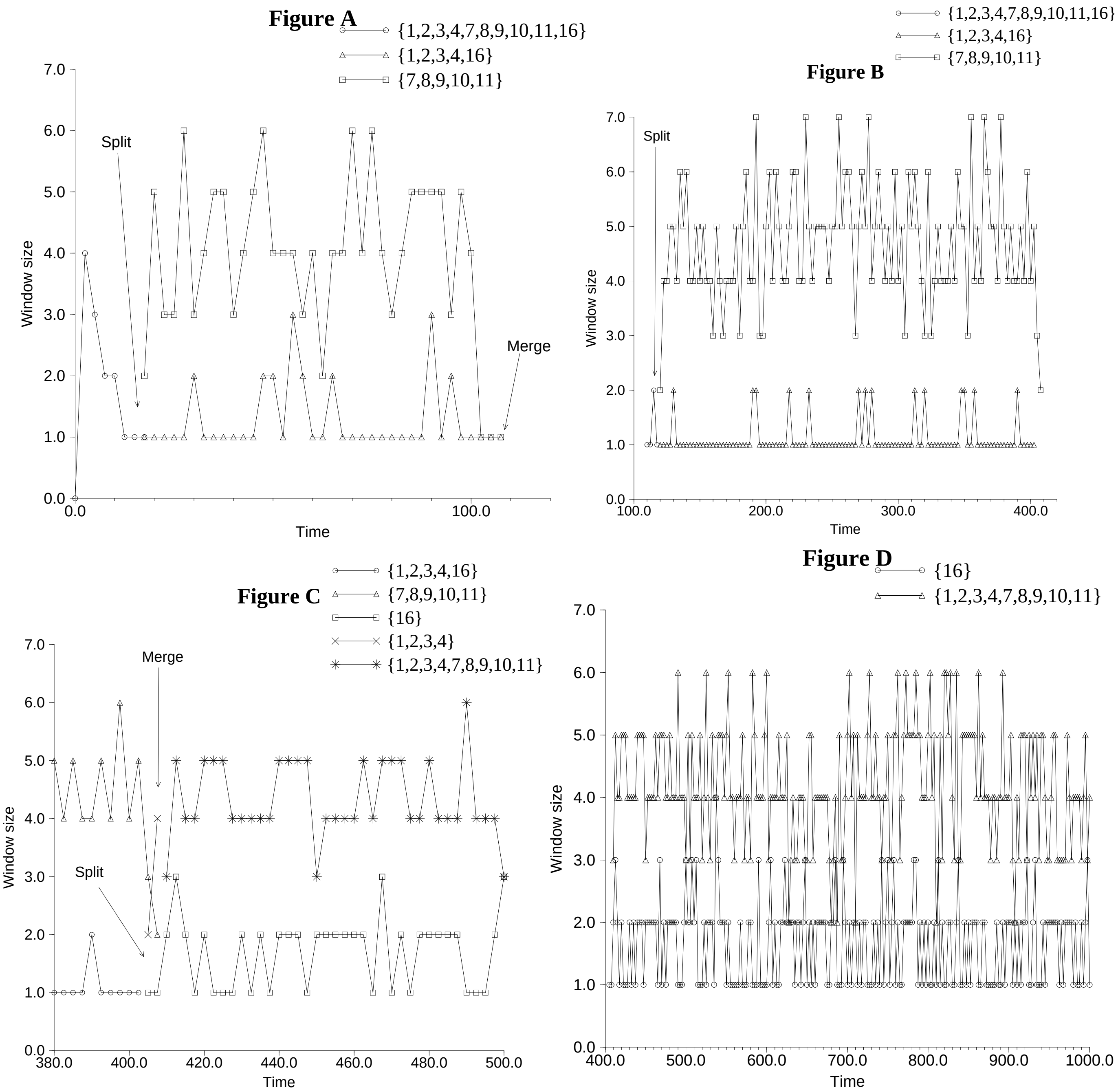
Finally, we show the window sizes used by multicast groupings in test case #1 in Figure 9 (The figure is representative of all test cases). The figure is divided in four subfigures and each subfigure shows a different simulation interval. Subfigures A, B and D shows the interval before, during and after the temporary congestion, respectively. Subfigure C shows in greater detail the behavior of the dynamic protocol during the transient period (380,500).

7 Concluding Remarks

We have studied the problem of improving the performance of window-controlled multicast communication through destination set grouping. With this technique, the set of multicast destinations is split into disjoint groups and the source carries independent conversations with each subgroup. We presented a static grouping heuristic and a dynamic grouping protocol for finding a favorable grouping of multicast destinations to improve unfairness and performance (power). Through simulation experiments, we have shown that despite the fact that the multicast traffic is increased with destination set splitting, the overall performance of the multicast conversation can be improved significantly.

By necessity, our protocols rely on heuristic arguments and use threshold parameters that sometimes need to be determined empirically. In future work we need to examine a more systematic approach to determining these parameters.

FIGURE 9 Detailed operational behavior of the dynamic grouping protocol



References

[1] J. L. Grange and M. Gien, *Experiments in Congestion Control Techniques*, pp. 211–234. in *Flow Control in Computer Networks*, IFIP, North-Holland Company, 1979.

[2] K. K. Ramakrishnan and R. Jain, "A binary feedback scheme for congestion avoidance in computer networks," *ACM Transactions on Computer Systems*, vol. 8, pp. 158–181, May 1990.

[3] R. Jain, "A timeout-based congestion control scheme for window flow-controlled networks," *IEEE Journal on Selected Area in Communication*, vol. SAC-4, pp. 1162–1167, Oct 1986.

[4] W. Bux and D. Grillo, "Flow control in local-area networks of interconnected token rings," *IEEE Transactions on Communications*, vol. 33, pp. 1058–1066, October 1985.

[5] R. Jain, "A delay-based approach for congestion avoidance in interconnected heterogeneous computer networks," *Computer Communication Reviews*, vol. 19, pp. 56–71, Oct 1989.

[6] M. H. Ammar and L. R. Wu, "Improving the throughput of point-to-multipoint ARQ protocols through destination set splitting," in *IEEE INFOCOM'92*, pp. 262–271, IEEE Communications Society, 1993.

[7] F. K. Hwang and D. S. Richards, "Steiner tree problems," *Networks*, vol. 22, pp. 55–89, Jan 1992.

[8] M. H. Ammar, S. Y. Cheung and C. Scoglio, "Routing multipoint connections using virtual paths in an ATM network," *Proceedings of INFOCOM 93*, pp. 98-105, March 1993

[9] L. Kleinrock, "Power and deterministic rules of thumb for probabilistic problems in computer communications," in *Proceedings of the International Conference on Communications*, pp 43.1.1–43.1.10, June 1979.

[10] R. Jain and K. K. Ramakrishnan, "Congestion avoidance in computer networks with a connectionless network layer: Concepts, goals and methodology," in *Proceedings of the Computer Networking Symposium*, pp. 134–143, April 1988.

[11] M. H. Ammar and S. B. Gershwin, "Equivalence relations in queueing models of fork/join networks with blocking," *Performance Evaluation*, vol. 10, pp. 233-245, December 1989.

[12] C. L. Liu, *Introduction to Combinatorial Mathematics*. McGraw Hill, 1968.

[13] D.-M. Chiu and R. Jain, "Analysis of the increase and decrease algorithms for congestion avoidance in computer networks," *Computer Networks and ISDN Systems*, vol. 17, pp. 1–14, 1989.