2009

Comparison of routing and network coding in group communications

Yangyang Xu
University of South Florida

Comparison of Routing and Network Coding in Group Communications

by

Yangyang Xu

A thesis submitted in partial fulfillment
of the requirements for the degree of
Master of Science in Electrical Engineering
Department of Electrical Engineering
College of Engineering
University of South Florida

Major Professor: Ravi Sankar, Ph.D.
Jing Wang, Ph.D.
Xiang-dong Hou, Ph.D.

Date of Approval:
March 24, 2009

Keywords: Optimization, Throughput, Undirected Networks, Subgradient Algorithm, Multicast Protocols

ACKNOWLEDGEMENTS

I would like to gratefully acknowledge the guidance and support from my advisor Dr. Ravi Sankar for my study at USF. I would also like to thank Dr. Jing Wang and Dr. Xiang-dong Hou for serving in my committee. I acknowledge the funding support from Raytheon Company and Florida High Tech Council for my research. I also want to thank Scalable Network Technologies for providing the QualNet simulation tool. I appreciate all the help coming from the iCONS group members for my study and life in US. Finally, I thank my wife Jing, for her encouragement, support and love.

TABLE OF CONTENTS

LIST OF FIGURES

Comparison of Routing and Network Coding in Group Communications

Yangyang Xu

ABSTRACT

In traditional communication networks, information is delivered as a sequence of packets from source to destination by routing through intermediate nodes which only store and forward those packets. Recent research shows that routing alone is not sufficient to achieve the maximum information transmission rate across a communication network [1]. Network coding is a currently researched topic in information theory that allows the nodes to generate output data by encoding their received data. Thus, nodes may mix the input packets together and send them out as fewer packets. Potential throughput benefit is the initial motivation of the research in network coding.

Group communications refers to many-to-many communication sessions where multiple sources multicast independent data to the same group of receivers. Researchers always treat group communications as a simple problem by adding a super source which is connected to all the sources with unbounded capacity links. However, it cannot control the fairness between different sources in this method. Additionally, the method may be incorrect in some scenarios. In this research, we will present an example to illustrate that and analyze the reason for that.

The maximum multicast throughput problem using routing only is NP-complete. Wu *et al.* introduced a greedy tree-packing algorithm based on Prim's algorithm as an alternate sub-optimal solution [2] . This algorithm is modified in this work for group communications problem with routing in undirected networks. The throughput benefit for network coding has been shown in directed networks. However, in undirected networks, researchers have only investigated the multiple unicast sessions problem and one multicast session problem. In most cases, network coding does not seem to yield any throughput benefit [3] [4]. Li *et al.* introduced a c-flow algorithm using linear programming to find the maximum throughput for one multicast session using network coding [3] . We adapted this algorithm for group communications with network coding in undirected networks to overcome the disadvantage of the traditional method. Both algorithms were simulated using MATLAB and their results were compared. Further, it is demonstrated that network coding does not have constant throughput benefit in undirected networks.

CHAPTER 1

INTRODUCTION

1.1 Background

Communication network is a directed graph model for the distribution of data from source to destination via intermediate nodes. Routing is the process of selecting paths in a network along which to send network traffic (voice/video/data packets). The network session is considered to be unicast when sending data from one source to one destination while it is termed multicast when sending data from one source to multiple destinations.

The desired goal in any network is to fully utilize its capacity. Communication networks today share the same fundamental principle of operation. Independent data streams may share network resources, but the information itself is separate. Routing, data storage, error control, and generally all network functions are based on this assumption. In the traditional communication networks, the intermediate nodes only copy and forward packets using the routing mechanism. Optimization problem for routing has been widely researched with graph theory. Generic augmenting path algorithm is one of the basic algorithms used in unicast communications [5]. Several other algorithms were developed to minimize the computation complexity. For multicast communications, Steiner tree algorithm is the state-of-the-art for achieving optimal throughput. However, it is NP-hard.

Several sub-optimal algorithms have been developed to get the maximum throughput and the best ratio to the optimal throughput is 1/1.55 [6].

Ahlswede *et al.* showed that achieving multicast capacity of a network requires network coding which allows intermediate nodes to generate output data by combining the received data [1] . The output flow at a given node is obtained as linear combination of its input flows when we use linear network coding. Linear network coding is sufficient to guarantee the multicast throughput to be the same as the maximum throughput from the source to each individual destination in unicast session.

The initial expected benefit from network coding is the throughput benefit which has been shown in directed networks. However, from various studies reported in the past eight years, there is no throughput benefit in most of the scenarios in undirected networks.

Li *et al.* showed that the network coding does not have any throughput advantage for one multicast session in 1,000 randomly generated undirected networks (the number of links are less than 35) [3] . They claimed that "the fundamental benefit of network coding is not higher optimal throughput, but to facilitate significantly more efficient computation and implementation of strategies to achieve such optimal throughput." Dougherty *et al.* concluded that there was no multiple unicast undirected network for which the coding capacity was larger than the routing capacity [4]. However, group communications problem in undirected networks has not been studied.

1.2 Motivation and Goals

The traditional method of solving the group communications has disadvantages. Also, there is nothing in the current literature on comparison of network coding and routing for group communications in undirected networks. So, the primary goals of this thesis research are as follows:

- Present the disadvantages of traditional method for group communications and analyze the reason for it.
- Develop algorithms to get the maximum throughput for group communications in undirected networks.
- Compare network coding and routing for group communications in undirected networks.

1.3 Outline of the Thesis

In this thesis, the basic idea of unicast is first introduced in Chapter 2. The max-flow min-cut theorem which is one of the most important theorems for network optimization problem is then presented. Max-flow min-cut theorem is also the foundation of the new subgradient algorithm for network coding in group communications which will be discussed later. The generic augmenting path algorithm which is the basic algorithm for calculating the maximum flow in one unicast session with routing only (intermediate nodes only copy and forward packets) is presented as well.

Then, the multicast capacity problem is introduced in Chapter 3. Steiner tree is the optimal solution for one multicast session problem. However, it is NP-hard. So, Prim's algorithm and greedy algorithm designed by Wu for one multicast session problem is

discussed [2]. Then, the proposed greedy algorithm for group communications is presented. It is based on modification to Wu's algorithm by making sure that all sources can have the same fairness which other algorithms cannot guarantee.

In Chapter 4, the real world Internet Protocol (IP) multicast with its advantages and disadvantages comparing to IP unicast is introduced. The current widely used IP multicast protocols IGMP v1/v2/v3 for hosts to join and leave the multicast group and PIM SM/DM for routing scheme are discussed. The simulation results for those protocols are compared to routing and network coding algorithms that are proposed in this thesis.

In Chapter 5, the network coding theory is introduced. Then, the linear network coding and practical random network coding which can be the coding scheme for the subgradient algorithm are discussed. The subgradient algorithm for one multicast session in undirected network designed by Li *et al.* [3] is then introduced. The proposed algorithm for group communications in undirected networks using network coding is presented. It is based on modification to Li's subgradient algorithm by making sure that all sources can have the same fairness.

The simulation results of our greedy algorithm with routing and our subgradient algorithm with network coding for group communications in undirected networks are presented in Chapter 6. The results show that in most of the time, network coding does not have obvious throughput benefit for group communications in undirected networks.

CHAPTER 2

UNICAST CAPACITY

2.1 Network Introduction

Network is a directed graph model for the distribution of goods, data, or merchandise, etc., from their production centers, to their destination. Each directed edge has a limited capacity which is the maximum number of data that can be transmitted through that channel per time period. The diagram indicates the capacity as a positive number associated with each edge. The actual number of data called the flow on that edge. It is a non-negative number less than or equal to the capacity. Data cannot accumulate at any node; therefore the total in flow at each node must equal to the out flow at that node. The problem is to find the distribution of data that maximizes the net flow from *s* to *t*.

This can be modeled mathematically as follows. When the edges of a graph have a direction, the graph is called a directed graph or digraph. A network *N* is a directed graph with two special nodes *s* and *t*; *s* is called the source and *t* is called the destination. All other nodes in set *I* are called intermediate vertices. The edges of a directed graph are ordered pairs *(u,v)* of vertices, which we denote by $\overrightarrow{uv}$. Each edge $\overrightarrow{uv}$ has a positive capacity which we denoted by $c(\overrightarrow{uv})$.

If *f* is a real valued function defined on the edge set *E*, and if $K \subseteq E$, we denote $f(K) = \sum_{\overrightarrow{uv} \in K} f(\overrightarrow{uv})$. When $K = (S, \bar{S})$, we define $f^+(S) = f(S, \bar{S})$, $f^-(S) = f(\bar{S}, S)$. If

the function *f()* satisfies the following conditions, we define it as the flow of the network.

- Capacity constraint, $0 \leq f(\overrightarrow{uv}) \leq c(\overrightarrow{uv}), \forall \overrightarrow{uv} \in E$;
- Conservation condition, $f^+(v) = f^-(v), \forall v \in I$

In general, there may be in edges as well as out edges at *s*. The net flow from *s* to *t* will then be the out flow at the source minus the in flow. This is called the value of the flow, $VAL(f) = f^+(s) - f^-(s)$. Any flow *f* that has maximum value for the network *N* is called a max-flow of *N*. This problem was first formulated and solved by Ford and Fulkerson [6], [17].

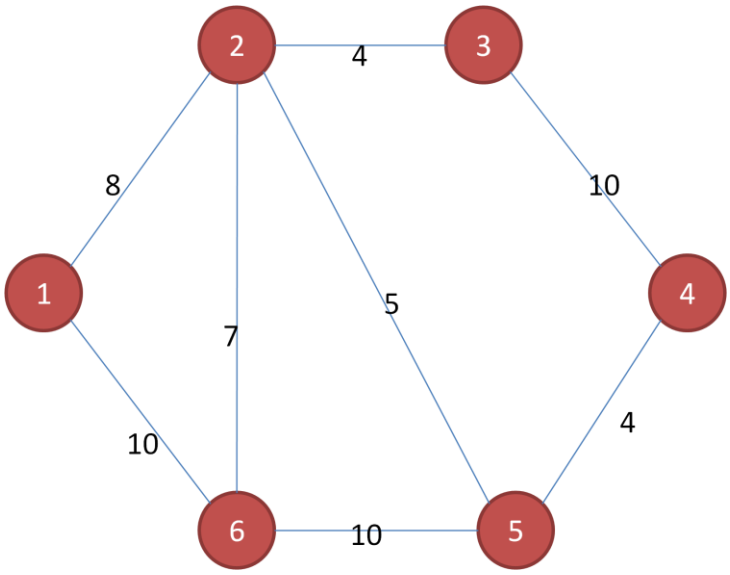


Figure 2.1 A Simple Network Topology

Since $f^+(v) = f^-(v)$, if $v \neq s$, this sum equals VAL(f). On the other hand, $f^+(v)$ is the total out flow at $v \in S$. Consider an out edge $\overrightarrow{vu}$ at v. Its flow $f(\overrightarrow{vu})$ contributes to $f^+(v)$. It also contributes to $f^-(u)$. If $u \in S$, then $f(\overrightarrow{vu})$ will appear twice in the sum, once for $f^+(v)$ and once for $f^-(u)$, and will therefore cancel. If $u \notin S$, then $f(\overrightarrow{vu})$ will appear in the summation as part of $f^+(v)$, but will not be canceled by $f^-(u)$. A similar argument holds if $v \in \bar{S}$ and $u \in S$. Therefore

$$VAL(f) = \sum_{v \in S} \left(f^{+}(v) - f^{-}(v) \right) = \sum_{\overrightarrow{vu} \in [S,\bar{S}]} f(\overrightarrow{vu}) - \sum_{\overrightarrow{vu} \in [\bar{S},S]} f(\overrightarrow{vu}) \qquad \text{Eq. 2.1}$$

This says that the value of the flow can be measured across any edge cut $[S, \bar{S}]$, such that $s \in S$ and $t \in \bar{S}$. [7]

In Chapter 2, the maximum flow problem is considered subject to the following assumptions.

- Assumption 2.1: The network is directed.
- Assumption 2.2: The network does not contain a directed path from node s to node t composed only of infinite capacity edges.
- Assumption 2.3: The network is delay-less and error-free.

2.2 Cuts

Let *N* be a network with a single source *s* and a single destination *t*. A cut in *N* is a set of edges of the form $(S, \bar{S})$. An *s*-*t* cut is an cut with $s \in S$ and $t \in \bar{S}$.

The capacity of a cut is the sum of the capacities of its edges. We denote the capacity of cut *K* by *cap K*. Thus, $cap\ K = \sum_{\overrightarrow{uv} \in K} c(\overrightarrow{uv})$.

The *s*-*t* cut whose capacity is the minimum among all *s*-*t* cuts is a minimum cut.

- Theorem 2.1: Let $K = [S, \bar{S}]$ be a *s*-*t* cut and flow *f*. Then $VAL(f) \leq CAP(K)$. If $VAL(f) = CAP(K)$, then *f* is a max-flow and *K* is a min-cut. (Max-Flow Min-Cut Theorem).

As shown in Figure 2.2, the cut is the min-cut with node 1 as the source and node 4 as the destination. As a result, the maximum throughput from node 1 to node 4 will be 4+4 = 8 units/sec.

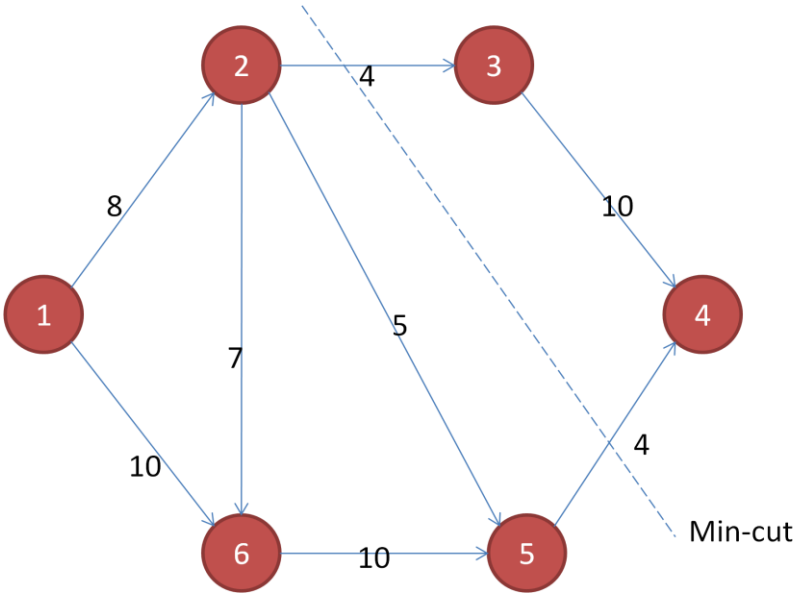


Figure 2.2 Min-Cut (Node 1 is the Source and Node 4 is the Destination)

2.3 Generic Augmenting Path Algorithm

Let us discuss one of the most intuitive algorithms for solving maximum flow problem, the augmenting path algorithm.

A directed path from the source to the sink in the residual network is referred as an augmenting path. We define the residual capacity of an augmenting path as the minimum residual capacity of any edge in the path. By definition, the capacity δ of an augmenting path is always positive. Consequently, whenever the network contains an augmenting path, we can send additional flow from the source to the destination. The generic augmenting path algorithm is essentially based on this simple observation. The algorithm proceeds by identifying augmenting paths and augmenting flows on these paths until the network contains no such path. The following is the description of the generic augmenting path algorithm.

Begin

$f := 0$;

While *G(f)* contains a directed path from node *s* to node *t* do

Begin

Identify an augmenting path *P* from node *s* to node *t*;

$\delta := min\{r_{ij}: (i,j) \in P\}$;

Augment δ units of flow along *P* and update *G(f)*;

End

End

Algorithm 2.1

The maximum flow problem given in Figure 2.3(a) illustrates the algorithm. Suppose that the algorithm selects the path s-2-*3*-*t* for augmentation. The residual capacity of this path is $\delta = min\{r_{13}, r_{34}\} = min\{8, 4, 10\} = 4$. This augmentation reduces the residual capacity of edge (2, 3) to zero (thus we delete it from the residual network) and increases the residual capacity of edge (3, 2) to 4 (so we add this edge to the residual network). The augmentation also decreases the residual capacity of edge (*s*, 2) from 8 to 4, edge (3, *t*) from 10 to 6 and increases the residual capacity of edge (2, *s*) from 0 to 4, edge (*t*, 3) from 0 to 4. Figure 2.3(b) shows the residual network after the first augmentation. In the second iteration, suppose that the algorithm selects the path *s*-6-5-*t*. The residual capacity of this path is $\delta = min\{10, 10, 4\} = 4$. Augmenting 4 units of flow along this path makes the residual network shown in Figure 2.3(c). In the third iteration, the algorithm augments 4 units of flow along the path 1-6-5-*t*. Now the residual network contains no augmenting path, so the algorithm terminates.

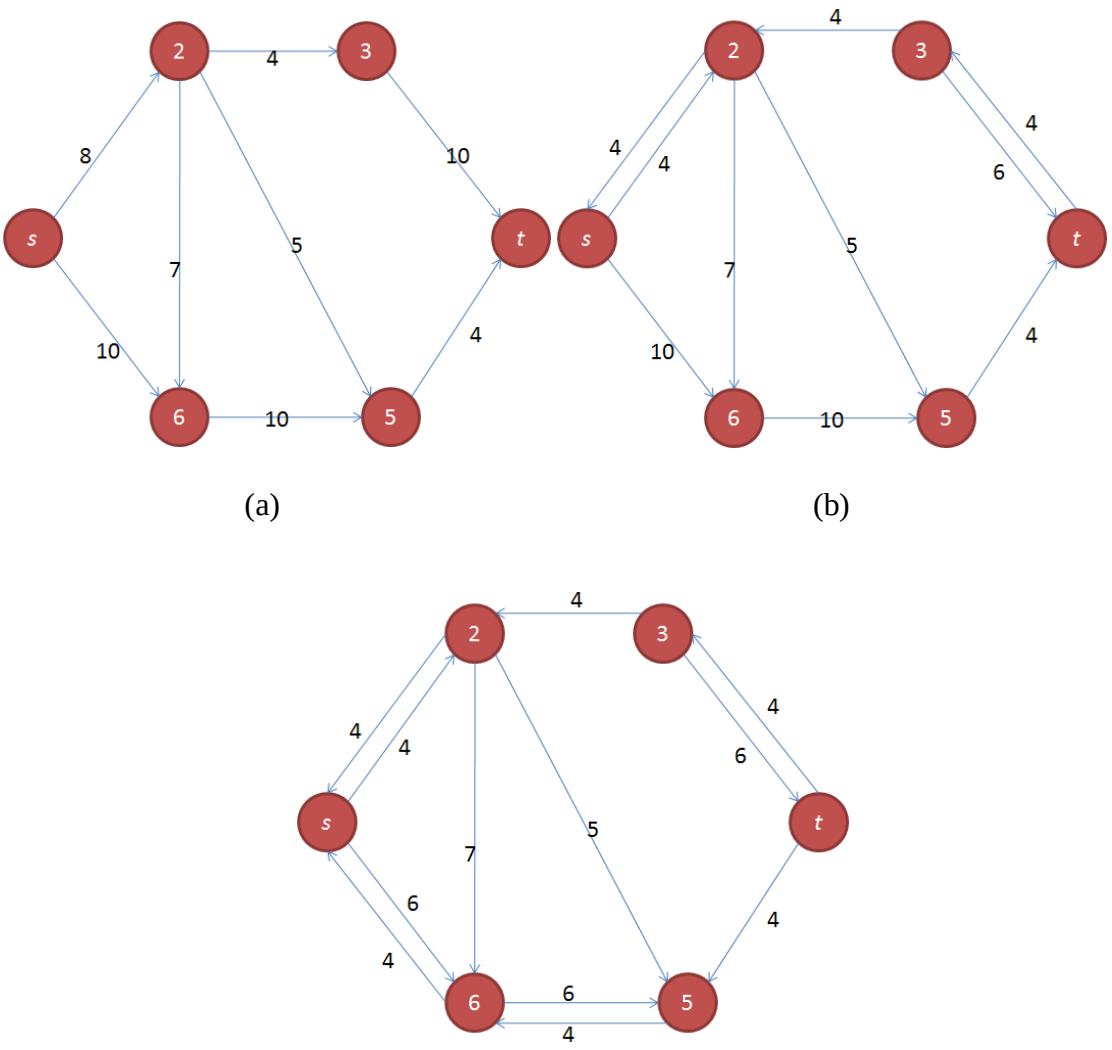


Figure 2.3 Illustrating the Generic Augmenting Path Algorithm. (a) Residual Network for the Zero Flow; (b) Network after Augmenting Four Units along the Path *s*-2-3-*t*; (c) Network after Augmenting Four Unit along the Path *s*-6-5-*t*

In implementing any version of the generic augmenting path algorithm, we have the option of working directly on the original network with the flows f_{ij} or maintaining the residual network *G(f)* and keeping track of the residual capacities r_{ij} and when the algorithm terminates recovering the actual flow variable f_{ij}. To see how we can use

either alternative, it is helpful to understand the relationship between edge flows in the original network and residual capacities in the residual network.

The concept of an augmenting path in the original network is considered next. An augmenting path in the original network *G* is a path *P* (not necessarily directed) from the source to the destination with $f_{ij} < c_{ij}$ on every forward edge (*i, j*) and $f_{ij} > 0$ on every backward edge (*i, j*). It is easy to show that the original network *G* contains an augmenting path with respect to a flow *f* if and only if the residual network *G(f)* contains a directed path from the source to the destination.

Assuming we update the residual capacities at some point in the algorithm. What is the effect on the edge flows f_{ij} by δ units on edge (i, j) in the residual network corresponds to (1) an increase in f_{ij} by δ units in the original network, or (2) a decrease in f_{ji} by δ units in the original network, or (3) a convex combination of (1) and (2). We use the example given in Figure 2.4(a) and the corresponding residual network in Figure 2.4(b) to illustrate these possibilities. Augmenting four units of flow on the path s-6-2-3-t in the network produces the residual network in Figure 2.4(c) with the corresponding edge flows shown in Figure 2.4(d). Comparing the solution in Figure 2.4(d) with that in figure 2.4(a), we find that the flow augmentation increases the flow on edge (1, 2), (2, 4), (3, 5), (5, 6) and decreases the flow on edge (3, 4). Finally, suppose that we are given values for the residual capacities. How should we determine the flow f_{ij}? Observe that since $r_{ij} = c_{ij} - f_{ij} + f_{ji}$, many combinations of f_{ij} and f_{ji} correspond to the same value of r_{ij}. We can determine one such choice as follows. To highlight this choice, we rewrite

$r_{ij} = c_{ij} - f_{ij} + f_{ji}$ as $f_{ij} - f_{ji} = c_{ij} - r_{ij}$. Now, if $c_{ij} \geq r_{ij}$ and $f_{ji} = 0$; otherwise, we set $f_{ij} = 0$ and $f_{ji} = r_{ij} - c_{ij}$.

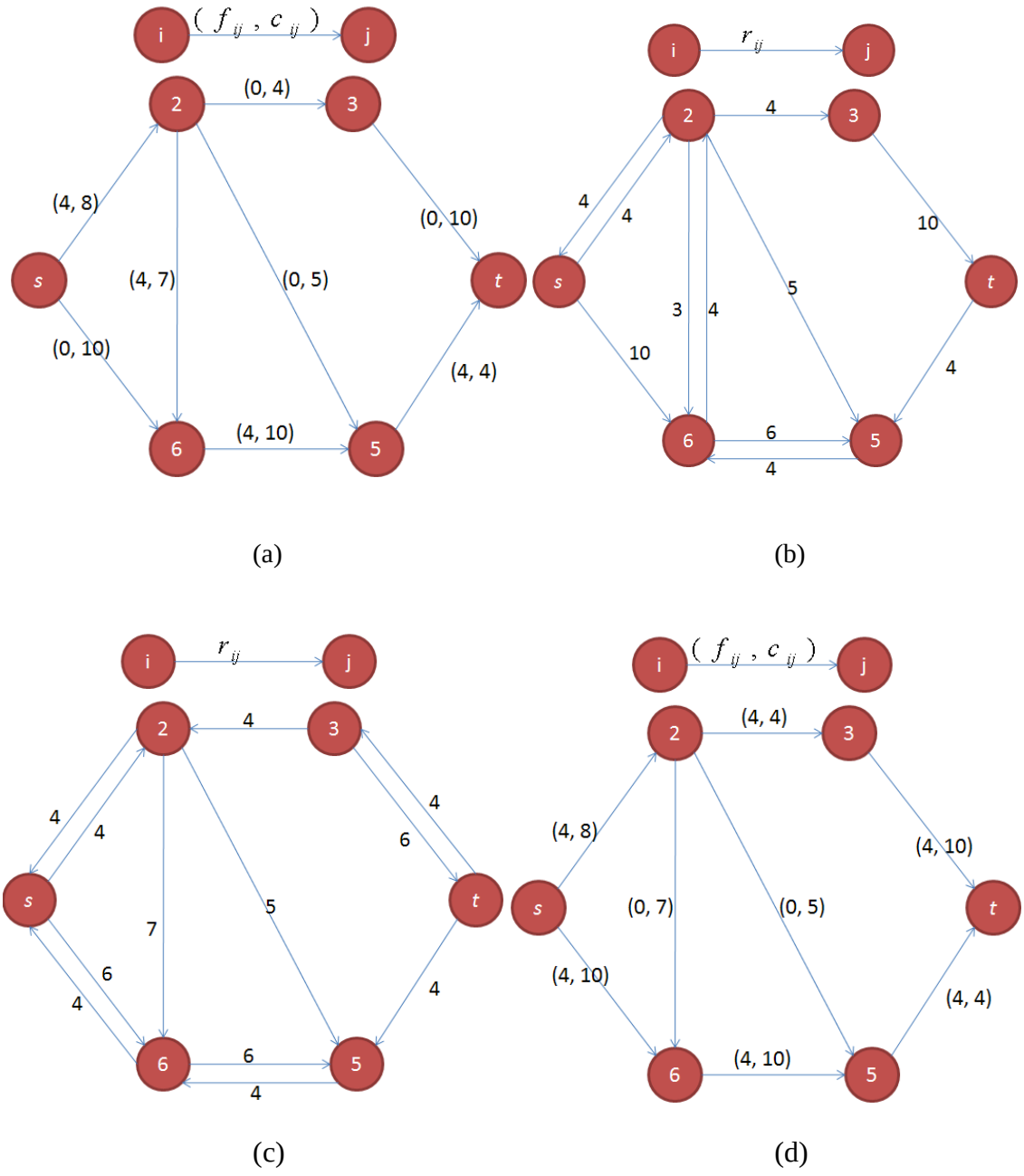
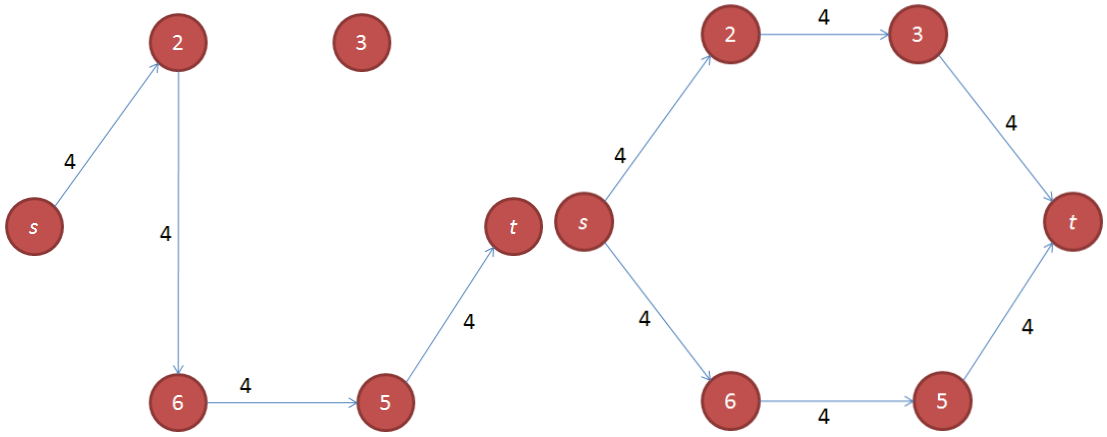
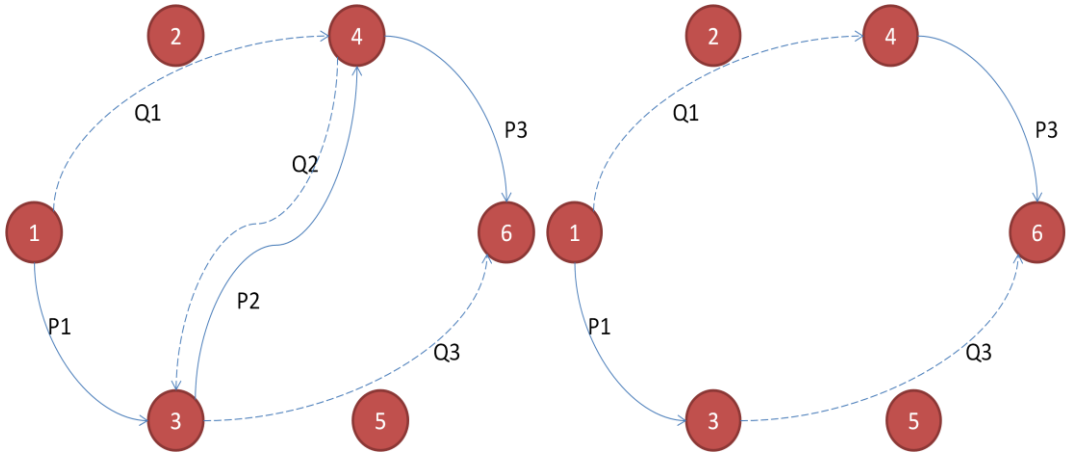


Figure 2.4 The Effect of Augmentation on Flow Decomposition.
(a) Original Network with a Flow *f*; (b) Residual Network for flow *f*;
(c) Residual Network after Augmenting Four Units along the Path *s*-6-2-3-*t*; (d) Flow in the Original Network after the Augmentation



(a) (b)

Figure 2.5 Flow Decomposition of the Solution in (a) Figure 2.4(a) and (b) Figure 2.4(d)



(c) (d)

Figure 2.6 The Effect of Augmentation on Flow Decomposition.
(a) The Two Augmentations *P1-P2-P3* and *Q1-Q2-Q3*; (b) Net Effect of These Augmentations

To obtain better insight concerning the augmenting path algorithm, we illustrate the effect of an augmentation on the flow decomposition on the preceding example. Figure 2.4(a) gives the decomposition of the initial flow and Figure 2.4(d) gives the decomposition of the flow after we have augmented four units of flow on the path s-6-2-

3-t. Although we augmented 4 units of flow along the path s-6-2-3-t, the flow decomposition contains no such path.

The path *s-2-6-5-t* defining the flow in Figure 2.4(a) contains four segments: the path up to node 2, edge (2, 6) and edge (6, 5) as a forward edge, and the path up to node t. We can view this path as an augmentation on the zero flow. Similarly, the path *s-6-2-3-t* contains four segments: the path up to node 6, edge (6, 2) as a backward edge, edge (2, 3) as a forward edge and the path down to node t. We can view the augmentation on the path *s*-2-3-*t* as linking the initial segment of the path *s*-2-6-5-*t* with the last segment of the augmentation *s*-6-2-3-*t*, linking the last segment of the path *s*-2-6-5-*t* with the initial segment of the augmentation *s*-6-2-3-*t*, and canceling the flow on edge (2, 6), which then drops from both the path *s*-2-6-5-*t* and the augmentation *s*-6-2-3-*t*. In general, we can view each augmentation as "pasting together" segments of the current flow decomposition to obtain a new flow decomposition [5].

- Property 2.1 A flow f is a maximum flow if and only if the residual network *G(f)* contains no augmenting path.

Augmenting path algorithm and max-flow min-cut theorem are solutions for max flow problem with single source and single destination. They are the fundamental of the subgradient algorithm for network coding in Chapter 5. The following is the pseudo code of the algorithm we used for the max-flow min-cut problem.

Begin

f := 0 (flow 0 on all edges)

opt := false

While not *opt* do

Construct the residual graph *G(f)*

Find a directed path *P* from *S* to *T* in *G(f)*

If such an augmenting path *P* exists

Then update flow *f* along *P*

Else set *opt* := true; and *S* := the set of vertices in *G(f)* reachable from *S*

End while

Return *f* as the max flow, and *(S , V-S)* as the min-cut

End [8]

Algorithm 2.2

CHAPTER 3

MULTICAST CAPACITY

The single multicast session from a source to a set of destinations is first considered. It can be theoretically shown that achieving optimal throughput via multiple multicast trees is equivalent to the problem of Steiner tree packing, which seeks to find the maximum number of pair wise edge disjoint Steiner trees, in each of which the multicast group remains connected. An intuitive explanation to such equivalence is that, each unit throughput corresponds to a unit information flow being transmitted along a tree that connects every node in the group. The maximum number of trees we can find corresponds to the optimal throughput for the session. Unfortunately, Steiner tree packing problem has been shown to be NP-complete. Several sub-optimal algorithms have been developed to get the maximum throughput and the best ratio to the optimal throughput is 1/1.55 [6].

While we consider multiple multicast sessions, the problem gets more complicated. Each multicast session in isolation and independently may cause congestion on some links and reduce network utilization. Any participant can be a multicast source. As a result, a multicast distribution routing graph that connects multicast members is shared by them. The choice of a multicast routing graph has several significant impacts both on the protocol performance and network utilization. In multicast packing problem, the choice of routing graph is more important since network resources need to be shared.

There are two proposals for multicast routing backbone: tree based and ring based. A comparison of the optimal multicast tree and ring topology reports that closing the cycle may require as many as 25% more links. Thus, we focus on the packing of multicast trees.

The underlying problem for the optimum shared tree is the Steiner tree problem which is NP-hard as we introduced before. One approach is based on finding a median or core node and building a shortest path tree rooted at the core (e.g., Core-Based Trees (CBT) and Protocol Independent Multicast (PIM)). There are many variants of CBT approach depending on the definition of the "core". Furthermore, results provide a comparison basis between a single shared tree and multiple source specific trees. [9]

3.1 Group Communications

Group communication refers to many-to-many communication sessions where multiple sources multicast independent data to the same group of receivers. The algorithm for group communications with routing is proposed here. First, the maximum-rate spanning trees are found based on Prim's algorithm. Then, those edges to non-destination nodes are pruned. This idea is demonstrated in [2] . We will include additional steps in to solve the group communications problem. Every multicast session can have the same priority and share the network resources fairly.

3.2 Prim's Algorithm

Prim's algorithm finds a minimum (or maximum) spanning tree for a connected weighted graph. It finds a subset of the edges that forms a tree that includes every vertex,

where total weight of all the edges in the tree is minimized (or maximized). The algorithm continuously increases the size of a tree starting with a single vertex until it spans all the vertices.

Algorithm 3.1

Input: A connected weighted graph with vertices *V* and edges *E*;

Initialize: *Vnew* = *{x}*, where *x* is a node from *V*, *Enew* = {};

Repeat until *Vnew* = *V*

Choose edge (*u*, *v*) from *E* with minimal (maximal) weight such that *u* is in *Vnew* and *v* is not as shown in Figure 3;

Add *v* to *Vnew*, add *(u,v)* to *Enew* ;

Output: *Vnew* and *Enew* describe the minimal (maximal) spanning tree. [2]

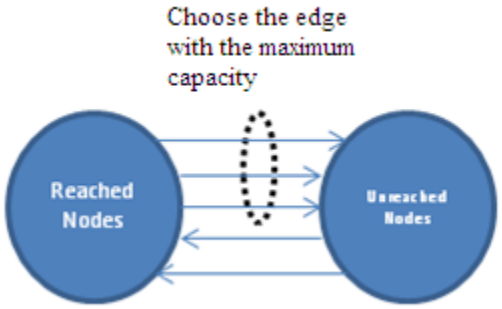
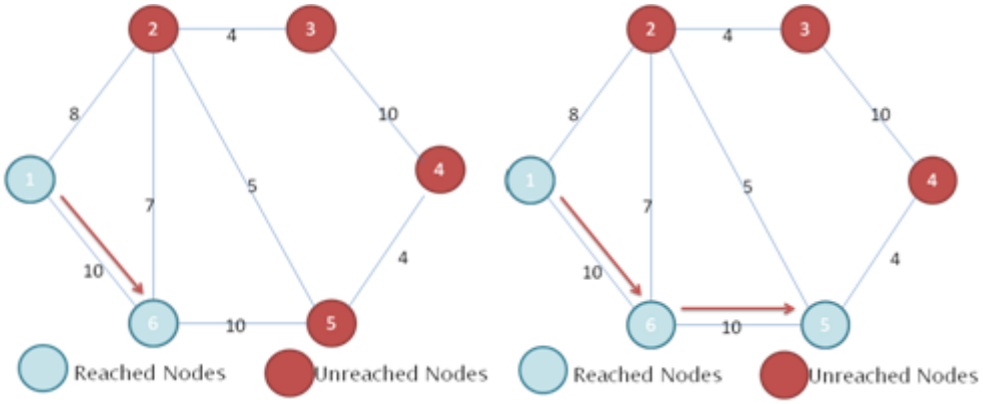
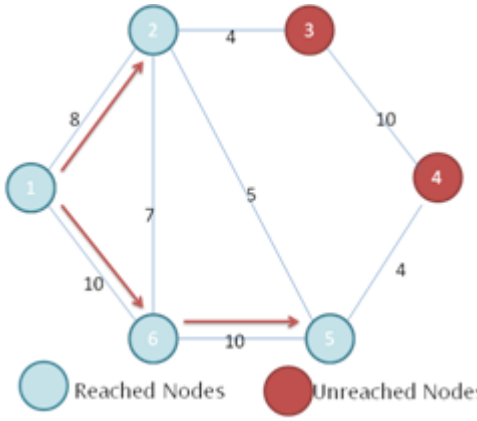


Figure 3.1 Prim's Algorithm

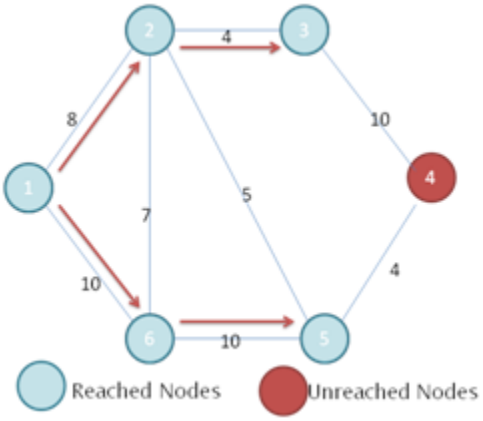


(a)

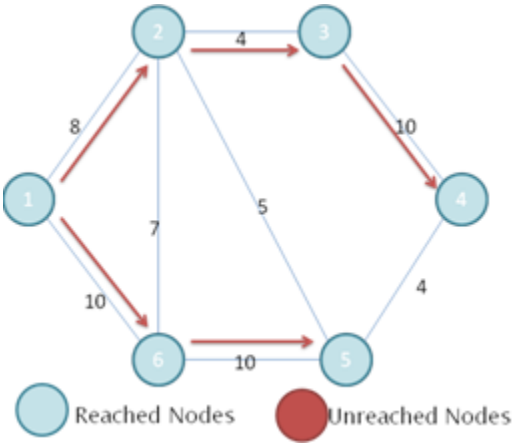
(b)



(c)



(d)



(e)

Figure 3.2 Example of the Prim's Algorithm

Figure 3.2 shows an example of the Prim's algorithm to find a maximum capacity spanning tree from the source node 1 to all other nodes in the network. First, as shown in Figure 3.2(a), the algorithm compares two links 1→2 and 1→6 from the reached node 1 to unreached nodes. Link 1→6 with the larger capacity has been chosen. Then, node 6 becomes a reached node. As shown in Figure 3.2(b), the algorithm compares the links 1→2, 6→2 and 6→5 from reached node 1 and 6 to unreached nodes. Link 6→5 with the largest capacity has been chosen. Then, node 5 becomes a reached node. The algorithm keeps running in the same way until all nodes become reached nodes.

3.3 Algorithm for Group Communications

Based on the Prim's algorithm, our greedy algorithm for the group communications is as following.

Algorithm 3.2

Step 1: Every source node generates its own maximum bandwidth tree with Prim's algorithm. Prune the branches to the non-destination nodes;

Step 2: Every tree decreases the bandwidth of the links by d_i;

Step 3: If the bandwidth of every links are greater than zero, go to step 1;
If the bandwidth of at least one link is less than 0, undo *step 2*. Go to step 4;

Step 4: If d_i is greater than *w*, $d_{i+1} = d_i/m$ and *i = i+1*. go to *step 1*;
Else, stop;

Step 5: Paste all the trees that every node has generated together.

Here, $d_i > 0$ is the bandwidth reduction step size, *i* is the times that step 4 has been run, the initial *i = 0*. *m > 1* is the step size reduction factor, and $0 < w < d_0$ is the

stop threshold. The result will be better but the calculation will be more complex if d_0 decreases, *m* increases or *w* decreases.

First of all, in order to get a fair result for each multicast session in group communications, every session run each step of the algorithm at the same time. We first let each source find a maximum capacity spanning tree by Prim's algorithm. Then, each spanning tree prunes those links to non-destination nodes. So, after the step 1, each session has a tree can cover from the source to all the destinations. Then, each tree decreases the capacity of the links it has by a same number d_i. After this step (step 2), each session has reserved a same number of capacity of the network, which means each session can transmit the same number of data with the reserved capacity of the network. Then, the algorithm reruns step 1 and step 2 until any of the remained bandwidth is less than zero. In step 4, if the stepsize d_i is greater than the stop threshold *w*, we cut the stepsize to $d_{i+1} = d_i/m$ and rerun the step 1 and step 2. This is because if the stepsize is too large, there will be too much waste of the capacity which will never be reserved and used by the group communications. If the stepsize d_i is less than the stop threshold *w*, the whole algorithm stops. After the whole algorithm stops, every session has its own distribution trees and all the sessions have the same maximum distribution rates.

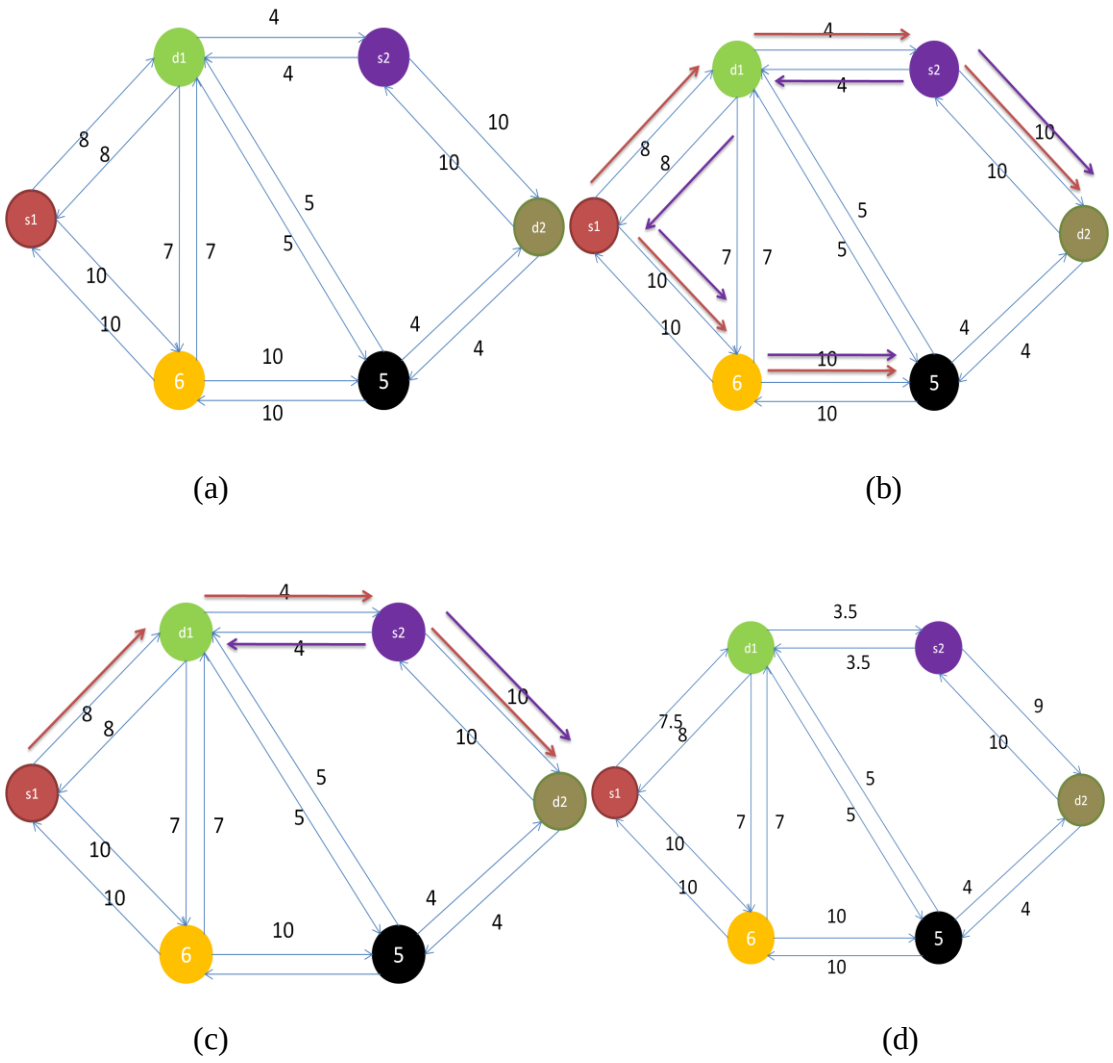


Figure 3.3 Example of our Algorithm for Group Communications

Figure 3.3 shows an example of our algorithm for group communications. Figure 3.3(a) shows the topology of the network. We suppose all links are bidirectional here. So, we divide them into two links with opposite directions. In Figure 3.3(b), both sessions (session 1 with the source s1 and destinations *d1* and *d2*, and session 2 with the source *s2* and destinations *d1* and *d2)* generate their own maximum capacity spanning trees with Prim's algorithm. In Figure 3.3(c), the algorithm prunes the un-useful leaves of the trees. So, after the step 1, session 1 has its distribution tree *s1*→*d1*→*s2*→*d2* and session 2 has

its distribution tree *s2→d1* and *s2→d2* as shown in Figure 3.3(c). Then, each tree reserves the same amount of capacity of the network which is 0.5 here in the example. Figure 3.3(d) shows the remained capacity of the network after step 2. So, with the reserved capacity, the group communications can guarantee 0.5 units/sec throughput for both sessions from source 1 and source 2 to all the destinations. For every round of the step 1 and step 2, the algorithm reserves some capacity from the network. After the algorithm stop, each session can combine the trees with the reserved capacity together and get a routing scheme. This routing scheme can guarantee each session has the same throughput.

CHAPTER 4

MULTICAST ROUTING PROTOCOLS

Internet Protocol (IP) network can transmit many types of data such as document files, audio and video within devices. However, when the data is sent via unicast, the source has to give each destination a copy which makes the whole network very inefficient. For example, when the manager wants to send his speech to the whole company, he has to send out one data flow for each employee. Obviously, this will cost a huge amount of the network resources including valuable Wide Area Network (WAN) capacity.

By using IP multicast technology, we can send data to a group of destinations through a very efficient way. Data flow is sent out from the source and tries to reach as far as possible in the network. Devices only copy the data when it has to send the data out through more than one interface in order to help the data reach all its destinations.

4.1 Comparison between Multicast and Unicast

When using unicast, the source needs to send out many copies of the data, each copy for each destination as shown in Figure 4.1.

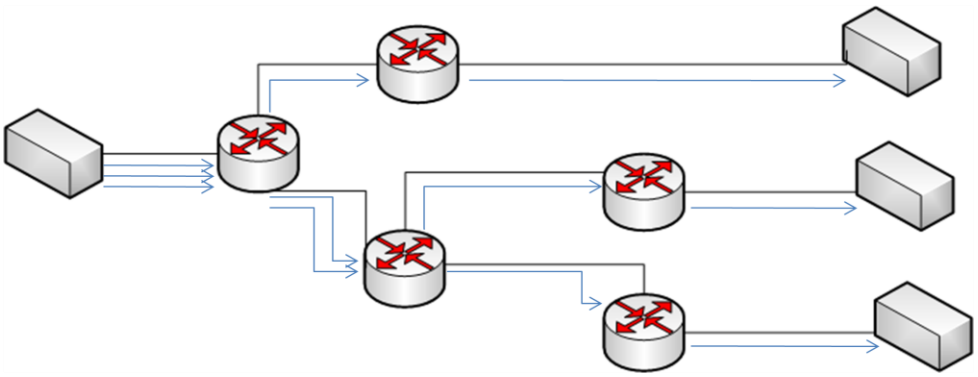


Figure 4.1 Example of Unicast

When using multicast, the source sends out data through one data flow as shown in Figure 4.2.

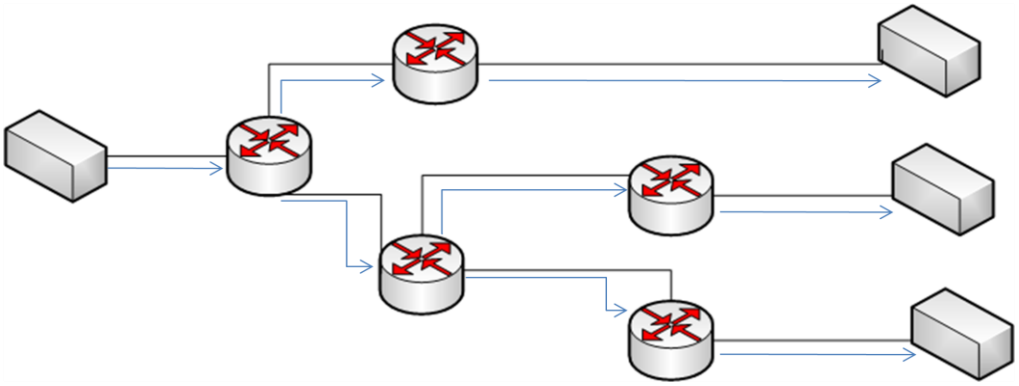


Figure 4.2 Example of Multicast

4.1.1 Multicast's Advantages

- Increase the efficiency: Because the multicast does not transmit several flows for each session, the network capacity has been used more efficiently.
- Optimize the performance: Multicasts avoids data flow redundancy. Data needs to be forwarded and proceeded becomes less.
- Support Distributed Application: When the topology increases, distributed applications becomes hard for unicast because it's lack of scalability. Multicast has made a lot of new applications available such as Webcasting, Web TV,

distance learning, telemedicine, Web radio, real-time videoconferencing, and other bandwidth- and time-critical information services.

4.1.2 Multicast's Disadvantage

Most of the multicast applications use UDP. Comparing to unicast with TCP, it has several disadvantages.

- UDP may cause multicast group loss. So, multicast application has to count the unreliable factor into consideration.
- UDP does not have congestion control function. So, if UDP becomes more and more popular on network, the network will become more congest and the whole performance of the network will drop.
- When the network topology changes, there is possibility that redundant multicast group will appear.
- There is potential security risk because the unwanted listener may find a way to join the multicast group.

So, when designing the multicast application, we have to take these disadvantages into consideration.

4.2 Multicast Models

Any-Source Multicast (ASM), Source-Filtered Multicast (SFM), and Source-Specific Multicast (SSM) are three multicast models based on how the receivers treat the multicast sources.

4.2.1 ASM Model

In the ASM model, any sender can be a multicast source and send information to a multicast group. Receivers can join a multicast group which is identified by the group IP address and obtain multicast information being sent to the group address. In this model, receivers can join or leave the multicast group at any time without regarding the sources.

4.2.2 SFM Model

Not all multicast sources are valid in the SFM model. The upper layer software checks the source address of received multicast packets and makes decision to permit or deny the traffic from specific sources. Therefore, the receivers can only receive the data from part of the sources because some sources may be filtered.

4.2.3 SSM Model

In the practical life, users may be interested in the multicast data from only some specific multicast sources. The receivers can specify their interested multicast sources in the SSM model.

4.3 IGMP

IGMP is the current widely used protocol for hosts to join or leave a specific multicast group. Routers know that which multicast group's data should be forwarded to the hosts based on the IGMP request it received from the hosts. There are three versions of IGMP.

4.3.1 IGMPv1

IGMPv1 is defined in RFC 1112 (Host Extensions for IP Multicasting).

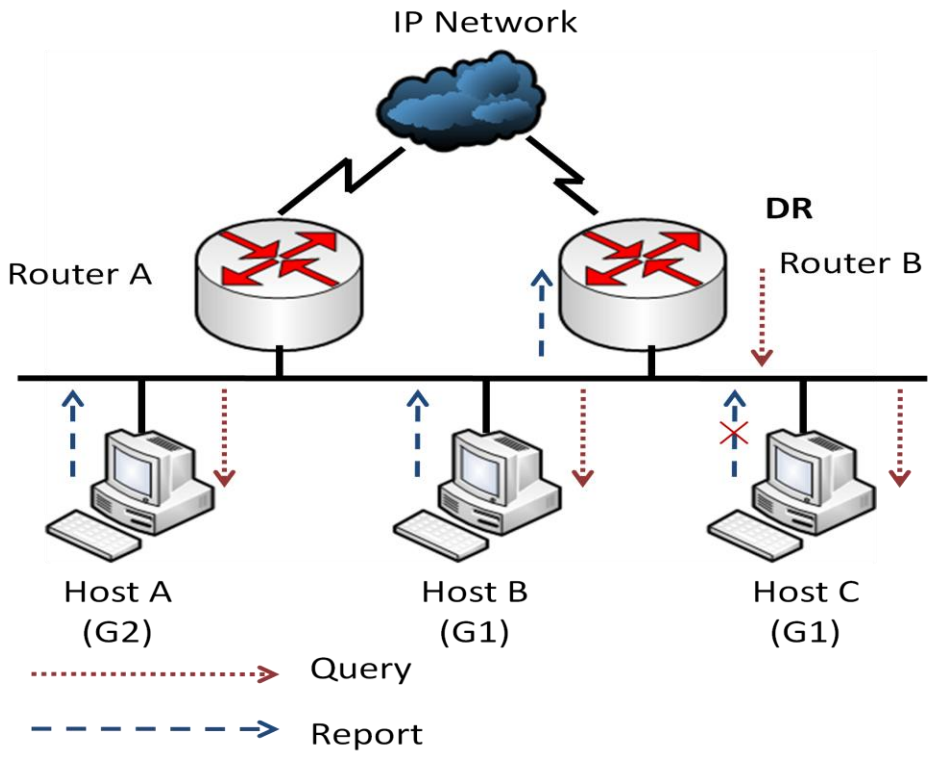


Figure 4.3 Example of IGMPv1

Assume that Host B and Host C are expected to receive multicast data to multicast group G1 with its group IP address. Host A is expected to receive multicast data to multicast group G2, as shown in Figure 4.3. The following shows how the hosts join the multicast groups and how the IGMP querier (Router B in Figure 4.3) maintains the multicast group memberships:

- The hosts send unsolicited IGMP reports to their interested multicast groups' addresses without having to wait for the IGMP queries from the IGMP querier.

- The IGMP querier periodically multicasts IGMP queries (with the destination address of 224.0.0.1) to all hosts and routers on the local subnet.
- Upon receiving a query message, Host B or Host C (the delay timer of whichever expires first) announces its membership to G1 by sending an IGMP report to the G1 multicast group address.
- At the same time, Host A sends a report to the multicast group address of G2.
- The IGMP routers learn the memberships of G1 and G2 attached to the local subnet through the above query/report process. The multicast routing protocol like PIM generates (*, G1) and (*, G2) multicast forwarding entries on the router where * represents any multicast source.
- When the multicast data addressed to G1 or G2 reaches the IGMP router, the router forwards the data to local subnets according to the multicast forwarding entries (*, G1) and (*, G2).

IGMPv1 does not support Leave Group message for hosts to leave the multicast group. Hosts stop sending IGMPv1 report to the IGMP router whenever it wants to leave the multicast group. So, the IGMP router will delete the multicast forwarding entries for one multicast group only after a period of time without receiving any IGMPv1 report from hosts regarding that multicast group.

4.3.2 IGMPv2

Compared with IGMPv1, IGMPv2 has introduced a querier election mechanism and a Leave Group mechanism.

4.3.2.1 Querier Election Mechanism

In IGMPv1, the Desinated Router (DR) elected by the Layer 3 multicast routing protocol (such as PIM) serves as the querier among multiple routers on the same subnet.

In IGMPv2, an independent querier election mechanism is introduced. The querier election process is as follows:

- Initially, every IGMPv2 router assumes itself as the querier and sends IGMP general query messages to all hosts and routers on the local subnet (the destination address is 224.0.0.1).
- Every IGMPv2 router compares the source IP address of the query message with its own interface address upon receiving a general query. After comparison, the router with the lowest IP address wins the querier election and all other IGMPv2 routers become non-queriers.
- All the non-queriers start a timer, known as "other querier present timer". If a router receives an IGMP query from the querier before the timer expires, it resets this timer; otherwise, it assumes the querier to have timed out and initiates a new querier election process.

4.3.2.2 "Leave Group" Mechanism

In IGMPv1, when a host leaves a multicast group, it does not send any notification to the multicast router. The multicast router relies on timeout of the host responding time to know whether a group no longer has members. This makes the leave group latency larger.

In IGMPv2, on the other hand, when a host leaves a multicast group:

- A Leave Group message (often referred to as leave message) is sent by the host to all routers (the destination address is 224.0.0.2) on the local subnet.
- Upon receiving the leave message, the querier sends a configurable number of group-specific queries to the group being left. The destination address field and group address field of the message are both filled with the address of the multicast group being queried.
- One of the remaining members, if any on the subnet, of the group being queried should send a membership report within the maximum response time set in the query messages.
- If the querier receives a membership report for the group within the maximum response time, it will maintain the memberships of the group; otherwise, the querier will assume that no hosts on the subnet are still interested in multicast traffic to that group and will stop maintaining the memberships of the group.

4.3.3 IGMPv3

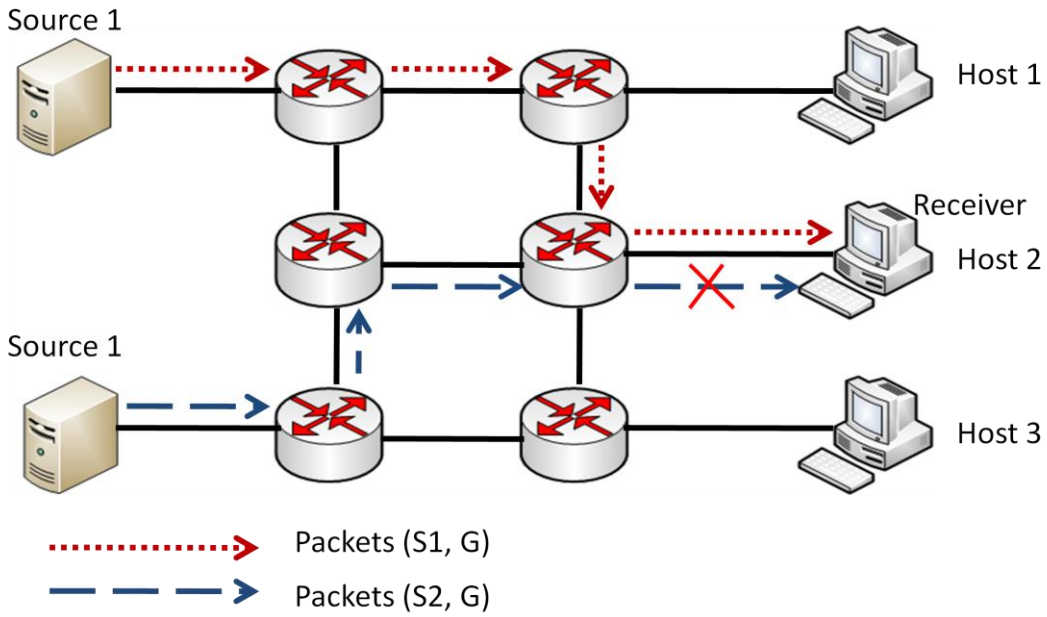


Figure 4.4 Example of IGMPv3

Built upon and being compatible with IGMPv1 and IGMPv2, IGMPv3 provides hosts with enhanced control capabilities and provides enhancements of query and report messages.

IGMPv3 Enhances control capability of hosts. IGMPv3 has introduced source filtering modes (Include and Exclude), so that a host not only can join a designated multicast group but also can specify to receive or reject multicast data from a designated multicast source. When a host joins a multicast group:

- If it needs to receive multicast data from specific sources like S1, S2, …, it sends a report with the Filter-Mode denoted as “Include Sources” (S1, S2, ……).
- If it needs to reject multicast data from specific sources like S1, S2, …, it sends a report with the Filter-Mode denoted as “Exclude Sources” (S1, S2, ……).

As shown in Figure 4.4, the network comprises two multicast sources, Source 1

(S1) and Source 2 (S2), both of which can send multicast data to multicast group G. Host B is interested only in the multicast data that Source 1 sends to G but not in the data from Source 2.

In the case of IGMPv1 or IGMPv2, Host B cannot select multicast sources when it joins multicast group G. Therefore, multicast streams from both Source 1 and Source 2 will flow to Host B whether it needs them or not.

When IGMPv3 is running between the hosts and routers, Host B can explicitly express its interest in the multicast data Source 1 sends to multicast group G (denoted as (S1, G)), rather than the multicast data Source 2 sends to multicast group G (denoted as (S2, G)). Thus, only multicast data from Source 1 will be delivered to Host B [10].

Currently, IGMPv2 is the most widely used protocol for hosts to joining the multicast group.

4.4 PIM Overview

Protocol Independent Multicast (PIM) provides IP multicast forwarding by leveraging static routes or unicast routing tables generated by any unicast routing protocol, such as routing information protocol (RIP), open shortest path first (OSPF), intermediate system to intermediate system (IS-IS), or border gateway protocol (BGP). Independent of the unicast routing protocols running on the device, multicast routing can be implemented as long as the corresponding multicast routing entries are created through unicast routes. PIM uses the Reverse Path Forwarding (RPF) mechanism to implement multicast forwarding.

Based on the implementation mechanism, PIM falls into two modes:

- Protocol Independent Multicast–Dense Mode (PIM-DM);
- Protocol Independent Multicast–Sparse Mode (PIM-SM).

4.4.1 PIM-DM

PIM-DM is a type of dense mode multicast protocol. It uses the "push mode" for multicast forwarding, and is suitable for small size networks with densely distributed multicast members.

PIM-DM assumes that at least one multicast group member exists on each subnet of a network, and therefore multicast data is flooded to all nodes on the network. Then, branches without multicast forwarding are pruned from the forwarding tree, leaving only those branches that contain receivers. This "flood and prune" process takes place periodically, that is, pruned branches resume multicast forwarding when the pruned state times out and then data is re-flooded down these branches, and then are pruned again.

When a new receiver on a previously pruned branch joins a multicast group, to reduce the join latency, PIM-DM uses a graft mechanism to resume data forwarding to that branch.

Generally speaking, the multicast forwarding path is a source tree, namely a forwarding tree with the multicast source as its "root" and multicast group members as its "leaves". The tree is also called Shortest Path Tree (SPT).

PIM-DM uses the "flood and prune" principle to build SPTs for multicast data distribution. Although an SPT has the shortest path, it is built with a low efficiency. Therefore the PIM-DM mode is not suitable for large and medium size networks.

4.4.2 PIM-SM

PIM-SM is a type of sparse mode multicast protocol. It uses the “pull mode” for multicast forwarding, and is suitable for large and medium size networks with sparsely and widely distributed multicast group members.

PIM-SM assumes that no hosts need to receive multicast data. In the PIM-SM mode, routers must specifically request a particular multicast stream before the data is forwarded to them. The core task for PIM-SM to implement multicast forwarding is to build and maintain Rendezvous Point Trees (RPTs). An RPT is rooted at a router in the PIM domain as the common node, or Rendezvous Point (RP), through which the multicast data travels along the RPT and reaches the receivers.

When a receiver is interested in the multicast data addressed to a specific multicast group, the router connected to this receiver sends a join message to the RP corresponding to that multicast group. The path along which the message goes hop by hop to the RP forms a branch of the RPT.

When a multicast source sends multicast streams to a multicast group, the source-side Designated Router (DR) first registers the multicast source with the RP by sending register messages to the RP by unicast until it receives a register-stop message from the RP. The arrival of a register message at the RP triggers the establishment of an SPT. Then, the multicast source sends subsequent multicast packets along the SPT to the RP. Upon reaching the RP, the multicast packet is duplicated and delivered to the receivers along the RPT [10].

4.5 Simulation of Current Multicast Protocols

The current multicast protocols were simulated using QualNet [16] version 4.5 with a randomly generated 30 nodes wired network topology. Link capacities are randomly picked between 10 Mbps and 20 Mbps. All link delays have been set to 0. Input and output buffer for all nodes is 16 Kbytes which is default. Packet size has been set to 1518 bytes/packet because in general larger packet size can get larger throughput in practice. Multicast with PIM DM and IGMPv2 has been enabled in all nodes as well as the whole environment. The unicast routing protocol is RIP which is good for small network that we simulated here. All destination nodes join the multicast group from 0 s to 30 s of the simulation. The data in group communications starts transmit from 10 s to 30 s to make sure that all multicast groups have been set up before the data transmission.

As packet size has been set up, the interval time between each packet was modified to adjust the throughput with binary search to get the maximum throughput for group communications. No packet loss is allowed and throughput for each source and each destination must be the same.

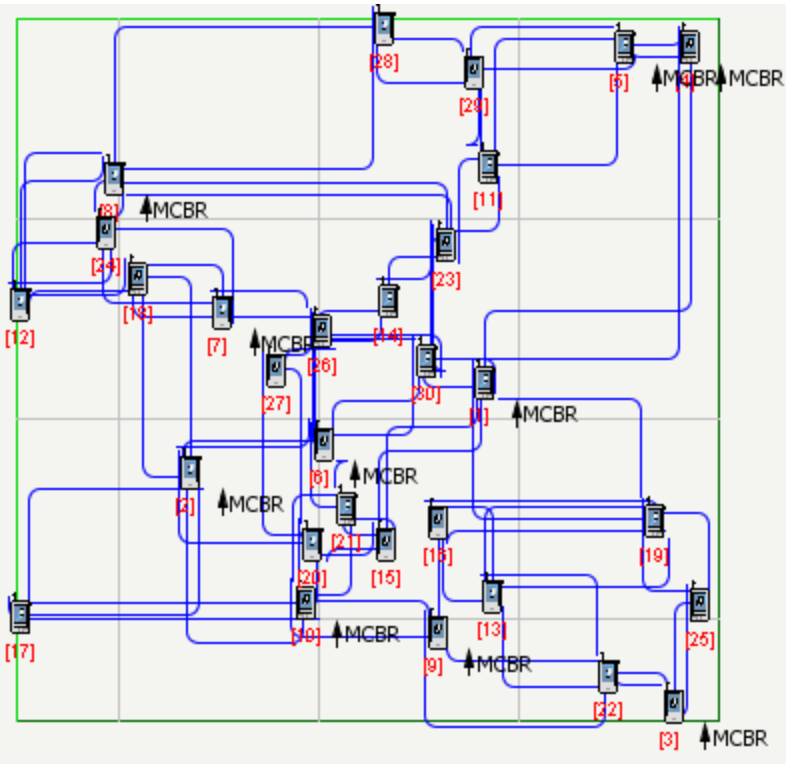


Figure 4.5 Network Topology for the Simulation of
Current IP Multicast Protocols

Figure 4.5 shows the topology of the network. As the simulated protocol does not have the function to adjust the capacity in undirected networks, we set bidirectional links for all connected nodes.

The maximum throughput from source 5 and 15 to destinations 1 and 2 is 8.1 Mbps; the maximum throughput from source 5 and 15 to all other 28 nodes as destinations is 4.8 Mbps; the maximum throughput from source 1, 2, 3, 4 and 5 to destinations 6 and 7 is 2.7 Mbps; the maximum throughput from source 1, 2, 3, 4 and 5 to all other 25 nodes as destinations is 2.5 Mbps; the maximum throughput from source 1 to 10 to destinations 11 and 12 is 2.2 Mbps; the maximum throughput from 1 to 10 to all

other 20 nodes as destinations is 1.4 Mbps. (All source and destination numbers mentioned here are the node numbers in the network topology).

The simulation result will be compared with routing and network coding algorithms' simulation results later in Chapter 6.

CHAPTER 5

NETWORK CODING

Network coding is a recent topic in information theory that allows the nodes to generate output data by encoding its received data. Thus, nodes may mix the input packets together and send them out as fewer packets.

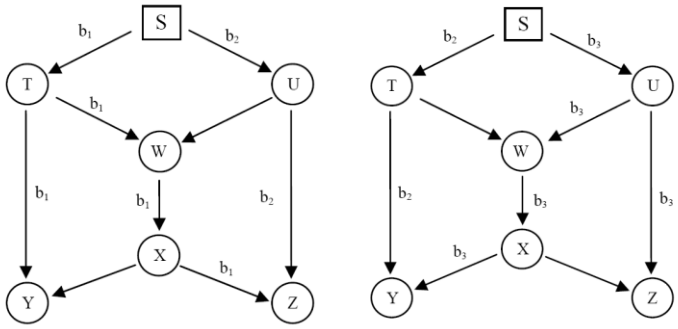


Figure 5.1 Multicast with Routing

Figure 5.1 shows the multicast routing mechanism. We assume the network is a delay-free and error-free network. S is the source while Y and Z are the sinks. All the links are with unit capacity. As shown in the graph, each sink could receive 3 units in 2 seconds. So, the maximum throughput for this multicast application is 1.5 units/sec here by using routing. The bound here is the cut T→Y, W→X and U→Z is shared by two sinks. So, the maximum throughput is 3/2 which is 1.5 units/sec.

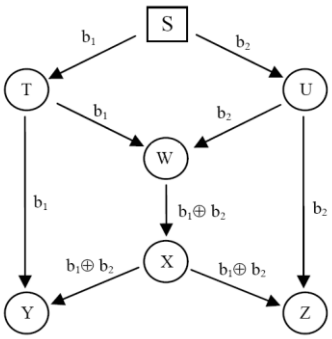


Figure 5.2 Multicast with Network Coding

Figure 5.2 shows the multicast with network coding mechanism. The network condition and the application requirement are both the same as Figure 5.1. While we allow node W encode its two input packets into one packet, the throughput becomes to 2 units/sec here. Node Y can receive two independent packets b1 and $b_1 \oplus b_2$ at the same time. Y can decode these two independent packets to get both b_1 and b_2. Also, Z can get both b_1 and b_2 by independent packets $b_1 \oplus b_2$ and b_2. The bound here is the minimum min-cut for each sink which decide how many independent packets a sink could receive.

5.1 Linear Network Coding

The output flow at a given node is obtained as linear combination of its input flows when we use linear network coding [11]. Linear network coding is sufficient to guarantee the multicast throughput as the same as the maximum throughput from the source to each individual destination in unicast session. When the packets to be combined have different sizes, the shorter ones are padded with trailing 0s. Assume that each packet consists of L bits. We can interpret s consecutive bits of a packet as a symbol over the finite field F_{2^s}, with each packet consisting of a vector of L/s symbols. With linear

network coding, outgoing packets are linear combinations of the original packets, where addition and multiplication are performed over the finite field F_{2^s}

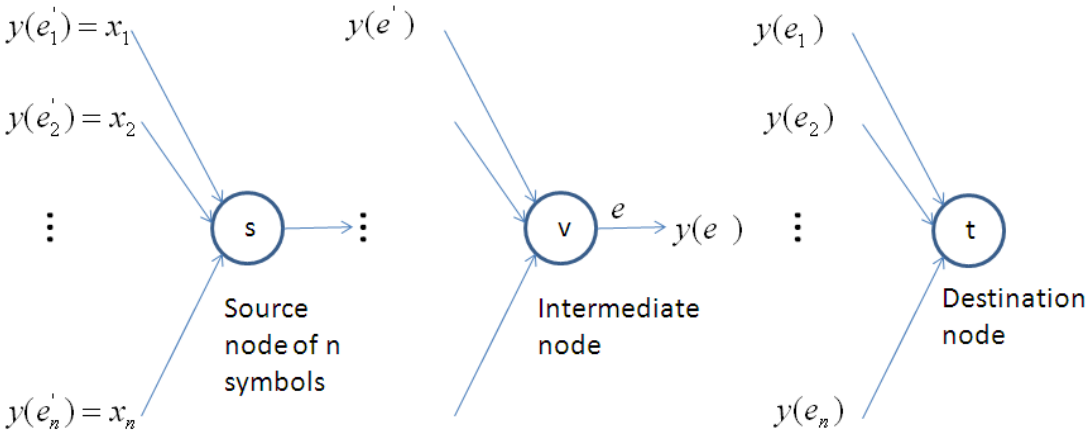


Figure 5.3 Network Coding [12]

Following are the two equations for intermediate nodes to encode the packets.

$$y(e) = \sum_{e':out(e')=v} m_e(e')y(e') \quad \text{Eq. 5.1}$$

$$\overrightarrow{m(e)} = [m_e(e')]_{e':out(e')=v} \quad \text{Eq. 5.2}$$

where $\overrightarrow{m(e)}$ is the local encoding vector. Each intermediate node has its own local encoding vector which can be both assigned by an algorithm or random generated. The intermediate node uses its local encoding vector to combine the input data together and send them out. Following two equations are global view of the linear network coding

$$y(e) = \sum_{i=1}^{n} g_i(e)x_i \quad \text{Eq. 5.3}$$

$$\overrightarrow{g(e)} = [g_1(e), \dots, g_n(e)] \quad \text{Eq. 5.4}$$

$$g(e) = \sum_{e':out(e')=v} m_e(e')g(e') \quad \text{Eq. 5.5}$$

where $\overrightarrow{g(e)}$ is the global encoding vector. The global encoding vector is calculated by local encoding vectors. With the global encoding vector, the destination nodes can decode the received packets to those symbols from the source. Following are the equations for decoding.

$$\begin{bmatrix} y(e_1) \\ \vdots \\ y(e_n) \end{bmatrix} = \begin{bmatrix} g_1(e_1) & \cdots & g_n(e_1) \\ \vdots & \ddots & \vdots \\ g_1(e_n) & \cdots & g_n(e_n) \end{bmatrix} \begin{bmatrix} x_1 \\ \vdots \\ x_n \end{bmatrix} = G_t \begin{bmatrix} x_1 \\ \vdots \\ x_n \end{bmatrix} \quad \text{Eq. 5.6}$$

$$\begin{bmatrix} x_1 \\ \vdots \\ x_n \end{bmatrix} = G_t^{-1} \begin{bmatrix} y(e_1) \\ \vdots \\ y(e_n) \end{bmatrix} \quad \text{Eq. 5.7}$$

Each destination node t can recover the source symbols $x_1, \ldots, x_n$ as long as the matrix G_t, formed by the global encoding vectors, has full rank n.

5.2 Practical Random Network Coding

There are several ways to generate the local encoding vector. In practice, distributed algorithm is preferred because it may be difficult to get global knowledge of the whole network. So, practical random network coding has been introduced by Chou [13].

When using practical random network coding, intermediate nodes select the linear coefficients in a finite field of opportune size in a random way. The encoding vector is included within the encoded packet. Nodes store within their buffers the received packets. This allows asynchronous packets arrivals and departures with arbitrarily varying rates, delay and loss. Simulation results indicate that even for small field sizes (for example, s = 8) the probability of selecting linearly dependent combinations becomes negligible.

When a node receives a packet, it decides whether to store the packet or discard it. If the new packet increases the current rank of the matrix, it is an innovative packet. If it does not increase the rank of the matrix, it means that the packet contains redundant information and it is not needed for decoding the original source packets. Hence, non-innovative packets are dropped.

Generally, we hope to invert G_t by collecting n or more packets. However, we can use the early decoding mechanism which is recommended here. Nodes perform Gaussian elimination after receiving each packet. Every node detects and discards non-innovative packets. G_t tends to be lower triangular, so it is typically possible to decode $\mathbf{x}_1, \ldots, \mathbf{x}_k$ with fewer more than k packets. This can make much shorter decoding delay than block decoding.

It has been shown that, in a directed network with network coding scheme, a multicast rate is feasible if and only if it is feasible for a unicast from the sender to each receiver [1]. Thus, there is an explicit max-flow min-cut capacity bound for the single-source multicast network coding problem. Also, the research work proves that linear coding usually suffices in achieving this maximum rate [11] . There exist directed graphs where the throughput gains of network coding for multicasting can be very significant. However, in undirected graphs the throughput gain is at most a factor of two [3] . Experimental results over the network graphs of six Internet service provides showed a small throughput gain in this case [9].

Li *et al.* introduced a c-flow algorithm using linear programming to find the maximum throughput for one multicast session in undirected networks using network coding [3] . Then, they introduced a sub-gradient algorithm with less complexity and

distributed implementation. They also gave the c-flow algorithm for multiple multicast sessions which they called m-flow algorithm. However, the m-flow algorithm cannot guarantee the same throughput for each source in group communications and it does not allow inter-session coding. Li *et al.* showed that, the throughput advantage with network coding is always 1.0 for one multicast session in 1,000 randomly generated undirected networks (the number of links are less than 35). They claimed that "the fundamental benefit of network coding is not higher optimal throughput, but to facilitate significantly more efficient computation and implementation of strategies to achieve such optimal throughput."

5.3 Subgradient Algorithm

Step 1: Choose initial orientation (e.g., balanced orientation)

Step 2: Repeat

Compute S → Ti max-flow, for all i

Refine orientation:

Increase bandwidth share for saturated links

Decrease bandwidth share for under-utilized links

Until convergence

As a result, optimal orientation obtained

Step 3: Compute S → Ti max-flow, for all i

As a result, optimal multicast rate and routing strategy obtained

Step 4: Randomized code assignment

As a result, complete transmission strategy obtained [2].

Algorithm 5.1

Primal variables in the orientation *c* are updated in two steps. First, we compute a new orientation vector *c'* as follows:

$$c' = c[k] + \theta[k]y[k] \qquad \text{Eq. 5.8}$$

where θ is a prescribed sequence of step sizes. A simple sequence is $\theta[k] = a/(bk + c)$, for some positive constants a, b and c. $y(\overrightarrow{uv})$ is valued to 1 when the link $\overrightarrow{uv}$ is the min-cut or 0 when the link $\overrightarrow{uv}$ is not the min-cut.

$$c[k+1](\overrightarrow{uv}) = \begin{cases} c'(\overrightarrow{uv}) & \Delta'(\overrightarrow{uv}) \le 0 \\ \dfrac{c'(\overrightarrow{uv})}{c'(\overrightarrow{uv}) + c'(\overrightarrow{vu})} C(uv) & \Delta'(\overrightarrow{uv}) > 0 \end{cases} \qquad \text{Eq. 5.9}$$

where $\Delta'(uv) = c'(\overrightarrow{uv}) + c'(\overrightarrow{vu}) - C(uv)$.

We can get the final $c(\overrightarrow{uv})$ after it converges.

During each iteration of orientation refinement, the algorithm computes the max-flow/min-cut from the sender to each receiver, and increases the capacity shared for more saturated links, while decreases the capacity shared for under-utilized links.

5.4 Algorithm for Network Coding in Group Communications

Group communication refers to many-to-many communication sessions where multiple sources multicast independent data to the same group of receivers. Researchers always treat group communication as a simple problem by adding a super source which is connected to all the sources with unbounded capacity links. However, it is not able to control the fairness between different sources in this method. Take the following network as an example.

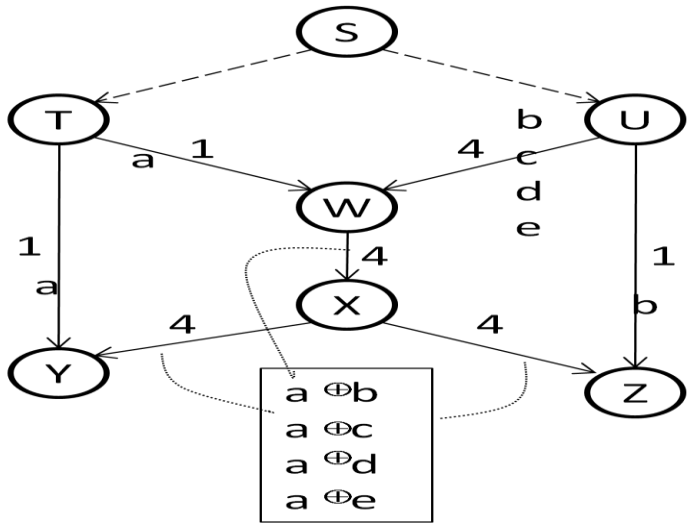


Figure 5.4 Group Communications with Network Coding

In the Figure 5.4, T and U are the sources while Y and Z are the sinks. If we add a super source S and connect it to T and U with unbounded capacity link, we can get the maximum multicast throughput from S to Y and Z with network coding is 5 units/sec based on the max-flow min-cut bound. This result comes from the cut T→Y and X→Y or the cut U→Z and X→Z. However, the maximum sending rate for the source T is only 1 units/sec because of the cut T→W and Y→ X and the maximum sending rate for the source U is 4 units/sec because of the cut U→W and Z→X. The two independent sources are not fair here.

In the example of Figure 5.4, the network is a directed network. If we extend the discussion to undirected networks, we can find that different sources may compete on the link capacity. As a result, we cannot treat group communications as a simple one multicast session problem by just adding a super source.

Now, let's take a look at another example to show that the method of adding only a super source and considering the group communication with network coding as a single session multicast problem is even incorrect.

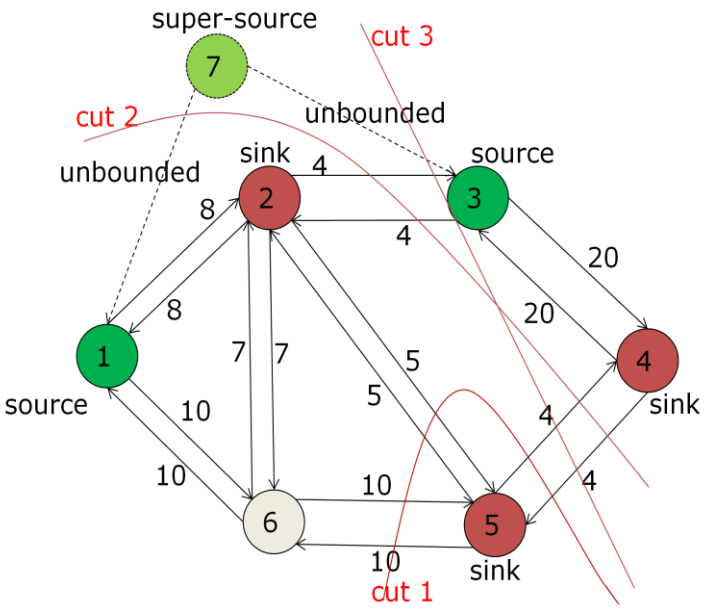


Figure 5.5 Six Nodes Bi-directional Network

In the network shown in Figure 5.5, node 1 and 3 are the sources while node 2, 4 and 5 are the sinks. The minimum min-cut from the super source to the three destinations is cut 1 which means the maximum sending rate of the super source is 19. The minimum min-cut from the source node 1 to the three destinations is cut 2 which means the maximum sending rate of the source node 1 is 8. The minimum min-cut from the source node 3 to the three destinations is cut 3 which means the maximum sending rate of the source node 3 is 8. The interesting thing is that 19 is larger the 8+8. As we know, the max-flow min-cut theory shows that the sink node 4 will never be able to receive more than 8 independent units per second from the source node 1 because of the cut 2 whenever we use network coding or not. The same thing happens from the source node 3

to the sinks node 2 and 5 because of the cut 3. As a result, in the original network (without the super source), the source 1 and source 3 will never be able to send out information with a total sending rate that is larger than 16. So, the result we get from the method that adding a super source is incorrect.

We need additional steps to design this new subgradient algorithm

Step 1: Choose initial orientation (e.g., balanced orientation)

Step 2: Put the super source S in and connect it to all sources with unbounded capacity links

Step 3: Repeat

Compute *Si* → *Tj* max-flow, for all *i* and *j*

Find the minimum max-flow *fsingle* for all sources to all sinks

Compute *S*→*Tj* max-flow for all *j*

Find the minimum max-flow *fsuper* for *S* to all sinks

Refine orientation:

Increase bandwidth share for saturated links

Decrease bandwidth share for under-utilized links

Algorithm 5.2

Until convergence

As a result, optimal orientation obtained

Step 4: Compute *S* → *Tj* max-flow *fsuper*, for all *j*. Compare this with *n* times the minimum max-flow for all sources and sinks *fsingle* and choose the less one.

As a result, optimal multicast rate and routing strategy obtained

Step 5: Randomized code assignment

As a result, complete transmission strategy obtained

Primal variables in the orientation *c* are updated in two steps. First, we compute a new orientation vector *c'* as follows:

$$c' = c[k] + \theta[k] \sum y[k] \qquad \text{Eq. 5.10}$$

If fsuper <= n*fsingle

$y(\overrightarrow{uv}) = 1$ when $\overrightarrow{uv}$ belongs to the min-cut of the super source

If fsuper > n*fsingle

$y(\overrightarrow{uv}) = 1$ when $\overrightarrow{uv}$ belongs to the min-cut for fsingle

Where θ is a prescribed sequence of step sizes and i indicates the ith source. A simple sequence is $\theta[k] = a/(bk + c)$, for some positive constants a, b and c. f_i is the maximum flow for the source i.

$$c[k+1](\overrightarrow{uv}) = \begin{cases} c'(\overrightarrow{uv}) & \Delta'(\overrightarrow{uv}) \le 0 \\ \dfrac{c'(\overrightarrow{uv})}{c'(\overrightarrow{uv}) + c'(\overrightarrow{vu})} C(uv) & \Delta'(\overrightarrow{uv}) > 0 \end{cases} \qquad \text{Eq. 5.11}$$

Where $\Delta'(uv) = c'(\overrightarrow{uv}) + c'(\overrightarrow{vu}) - C(uv)$.

We can get the final $c(\overrightarrow{uv})$ after it converges.

When there are *n* independent sources in the network, the main concern is trying to maximize the capacity from the super source to the destinations. However, when the maximum flow for the super source comes more than *n* times the maximum flow for any single sources, we have to balance the capacity for those sources. During each iteration of orientation refinement, the algorithm computes the max-flow/min-cut from each sender (include the super source) to each receiver. After that, it increases the capacity shared for more saturated links, while decreases the capacity shared for under-utilized links. This method will maximize the capacity from the sources to the sinks and make different

sources as fair as possible. After getting the final $c(\overrightarrow{uv})$, if the *fsuper* is less than *n* times *fsingle*, we can run the well designed random coding scheme by treating this as a one session multicast from the super source to all the receivers while each sources sends the information with rate *fsuper/n*. If the *fsuper* is larger than *n* times *fsingle*, we use the same random coding scheme while each source can only sends the information with rate *fsingle*.

CHAPTER 6

SIMULATION AND COMPARISON OF RESULTS

The main differences between the subgradient algorithm for network coding and the greedy algorithm for routing in group communications are as follows.

- The algorithm for routing tries to help each session (transmission from each source) reserve some capacity from the network in each iteration until there is no more available capacity. There is unused capacity in the network. However, it requires more computation. So, we can modify the parameter of the algorithm based on the network scenario.
- The algorithm for network coding is based on the idea that network coding can guarantee each session (transmission from each source) reaching the throughput obtained by max-flow min-cut algorithm to each destination. So, the algorithm assigns the half and half capacity on each direction of every link in the network initially. Then, different sessions compete for the capacity of the network until it converges.

The main contribution of these algorithms is that they can both guarantee each sessions having the same throughput.

Figure 6.1 shows the network topology for simulation. The topology is randomly generated by the network simulation tool QualNet version 4.5. Thirty nodes are randomly connected through links with random integer capacities between 10 and 20 units/sec.

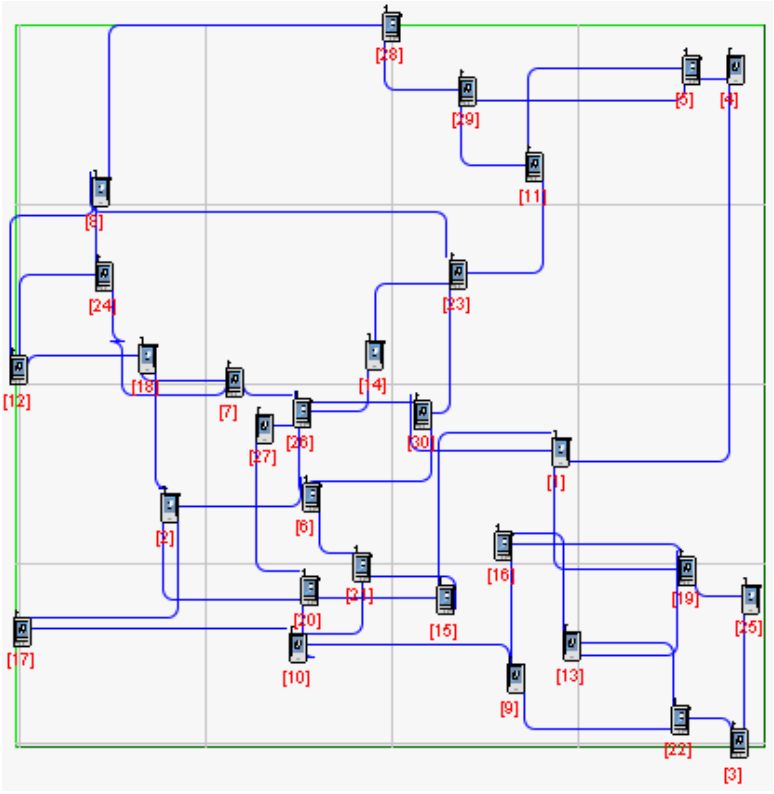
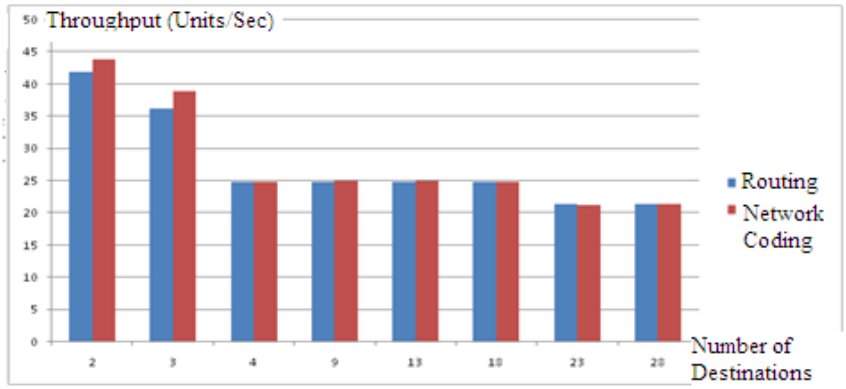


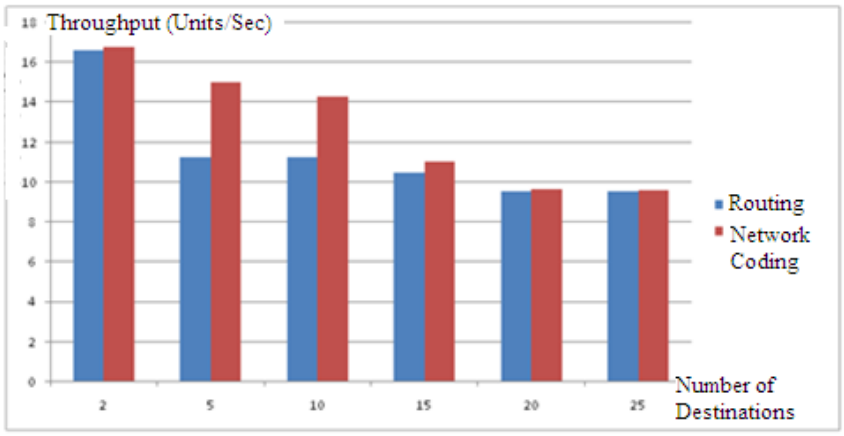
Figure 6.1 Simulation Network Topology

Both of the algorithms were simulated in MATLAB with the following network parameters

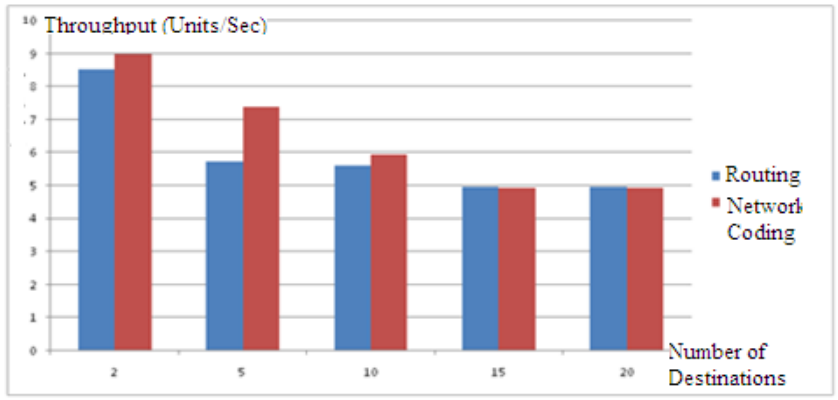
- Routing: initial stepsize 0.1, bound 0.1;
- Network coding: k <= 1000, stepsize 1000/ (50+k), k is the iteration number.



(a)



(b)



(c)

Figure 6.2 Simulation Results of the Comparison of Network Coding and Routing. (a) Group Communication with 2 Sources; (b) Group Communications with 5 Sources; (c) Group Communications with 10 Sources

Figure 6.2 (a) shows the result of group communications with 2 sources which are node 5 and node 15 here. Figure 6.2 (b) shows the result of group communications with 5 sources which are node 1, node 2, node 3, node 4 and node 5 here. Figure 6.2 (c) shows the result of group communications with 10 sources which are node 1, node 2, node 3, node 4, node 5, node 6, node 7, node 8, node 9 and node 10 here.

As shown in Figure 6.2, network coding does have obvious throughput advantage sometimes, but not always. We changed the topology, the source nodes as well as the destination nodes for the simulation and results were always similar. At most of the time, network coding does not have obvious throughput advantage which is the same as one multicast session communications simulated by Li *et al.* [2].

The following results are for comparison with current IP multicat protocols.

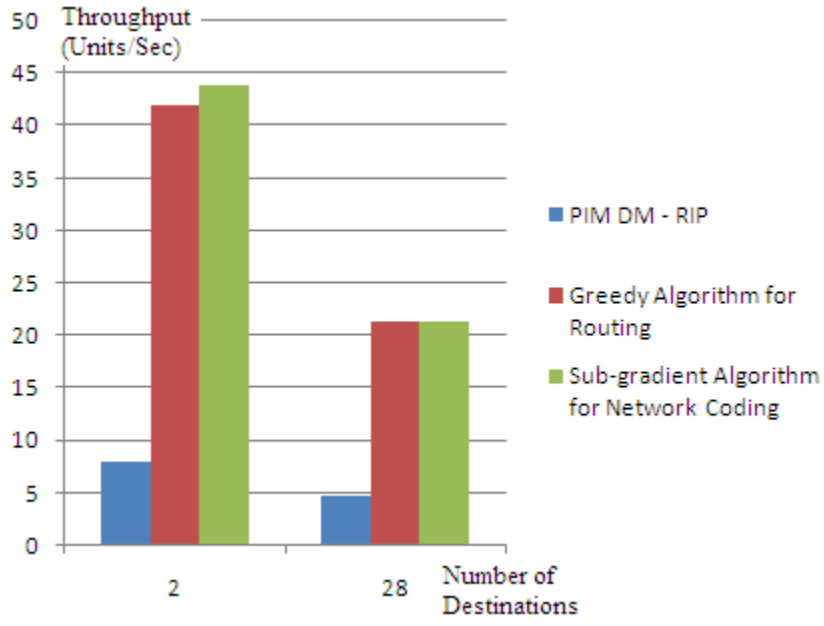


Figure 6.3 Maximum Throughputs from Source Node 5 and 15

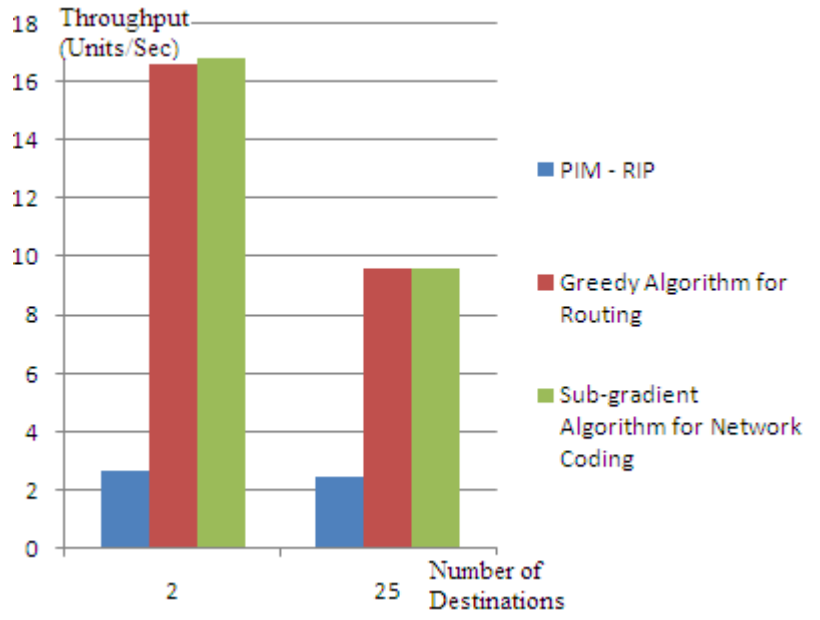


Figure 6.4 Maximum Throughputs from Source Node 1, 2, 3, 4 and 5

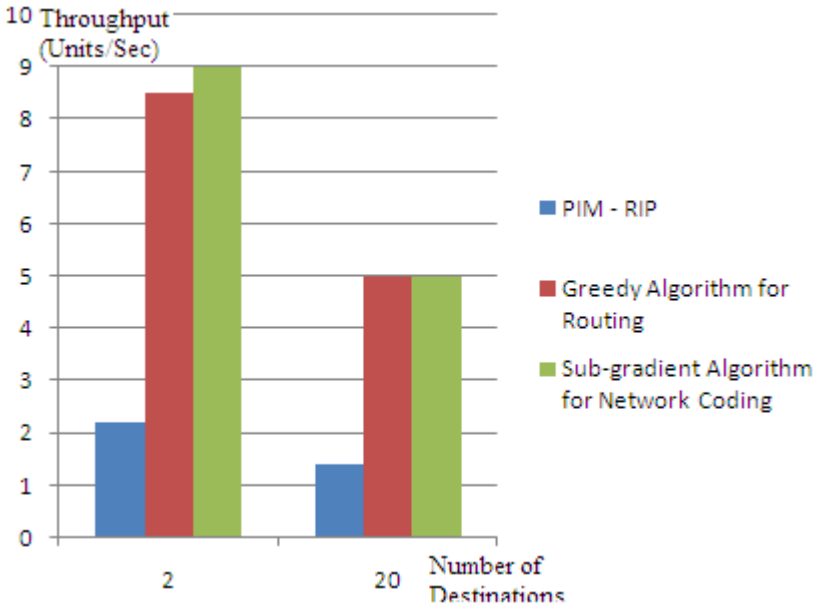


Figure 6.5 Maximum Throughputs from Source Node 1 to Node 10

So, our simulation showed a throughput comparison between network coding and routing for group communications in undirected networks. Current multicast protocols are much more scalable and easier to deploy in real network. However, the throughput with those protocols will be much less than the algorithms we simulated here. Figure 6.3 to Figure 6.5 show that both algorithms being introduced here (one for routing only and one for network coding) have huge throughput benefit to current IP multicast protocols. Also, although we can set up QOS policy in routers and switches, current protocols cannot optimize the network to make different sessions as fair as possible. Trade-off is always there in engineering problems. Researchers are doing huge effort on network coding and hope that network coding can support the maximum throughput with routing and becomes easier to deploy.

CHAPTER 7

CONCLUSION AND FUTURE WORK

7.1 Summary

The traditional method to solve the group communications problem is putting a super source with unlimited bandwidth to all sources. In this thesis, it is shown that this method cannot guarantee the fairness within different sources for routing. Also, the method can be incorrect in certain scenarios. Two algorithms are presented in this thesis, one for routing and one for network coding to guarantee that each source has the same fairness and get the sub-optimal throughput for group communications in undirected networks. All current widely used routing protocols are topology-based. The throughputs using both these algorithms (one for routing only and one for network coding) are much better than current widely used IP multicast protocols. Between the two proposed algorithms, the algorithm for network coding can have throughput benefit in some scenarios but not always. Here, we show that network coding does not have constant throughput benefit in undirected networks in group communications scenario with the consideration of fairness within different sources.

7.2 Recommendations for Future Research

Both our algorithms are not distributed algorisms. So, it is hard to deploy them into large networks. Also, they will not work well if there are link failures or topology

changes in the network. Further research is necessary to improve the scalability and robustness of these algorithms. Cluster-based algorithms might be a good approach.

We only simulated these algorithms in one randomly generated thirty nodes network scenario. Simulation in large network is necessary for future research.

REFERENCES

[1] R. Ahlswede, N. Cai, S. Li, and R. W. Yeung, "Network Information Flow," IEEE Trans. on Information Theory, vol. 46, pp. 1204-1216, 2000

[2] Y. Wu, P. A. Chou, and K. Jain, "A Comparison of Network Coding and Tree Packing," in Information Theory, ISIT 2004 Proceedings, pp. 143-149, 2004

[3] Z. Li, B. Li, D. Jiang, and L. C. Lau, "On Achieving Optimal Throughput with Network Coding," in Proceedings IEEE INFOCOM 2005, vol. 3, pp. 2184-2194, 2005

[4] R. Dougherty, Chris Freiling, and Kenneth Zeger, "Unachievability of Network Coding Capacity," IEEE Trans. on Information Theory, vol. 52, pp. 2365-2372, June 2006

[5] J. B. Orlin, T. L. Magnanti, and R. K. Ahuja, "Network Flows: Theory, Algorithms, and Applications," 1993

[6] G. Robins, and A. Zelikovsky, "Improved Steiner Tree Approximation in Graphs," in Proceedings of the Eleventh Annual ACM-SIAM Symposium on Discrete Algorithms, pp. 770-779, 2000

[7] U. S. R. Murty, and J. A. Bondy, "Graph Theory with Applications," 1976

[8] A. Mirzaian, http://www.cse.yorku.ca/~aaw/Wang/MaxFlowMinCutAlg.html

[9] S. Chen, O. Gunluk, and B. Yener, "The Multicast Packing Problem," IEEE/ACM Trans. Networking, pp. 311-318, June 2000

[10] Hangzhou H3C Technologies Co., Limited, Document Team, www.h3c.com

[11] S. Li, R. W. Yeung, and N. Cai, "Linear Network Coding," IEEE Trans. on Information Theory, vol. 49, pp. 371-381, 2003

[12] E. Fasolo, Slides from www.cs.virginia.edu/~yw5s/Network%20coding.ppt

[13] P.A. Chou, T. Wu, and K. Jain, "Practical Network Coding," the 51st Allerton Conf. Communication, Control and Computing, Oct. 2003

[14] R. Koetter, and M. Medard, "An Agebraic Approach to Network Coding," IEEE/ACM Tran. on Networking, vol. 11, pp. 782-795, Nov. 2003

[15] N. J.A. Harvey, R. Kleinberg, and A.R. Lehman, “On the Capacity of Information Networks,” IEEE Trans. Information Theory, pp. 2345-2364, June 2006

[16] Scalable Network Technologies, Inc., http://www.scalable-networks.com/

[17] L. R. Ford, and D. R. Fulkerson, “Maximal Flow through a Network”, Canadian Journal of Mathematics 8, pp. 399-404, 1956