A Practical Network Coding and Routing Scheme based on Maximum Flow Combination

Lianlong Wu, Kevin Curran

School of Computing and Intelligent Systems
Faculty of Computing and Engineering,
University of Ulster, Northern Ireland, UK
Email: kj.curran@ulster.ac.uk

Abstract

Network coding is a novel field of information theory and coding theory. It is a breakthrough over the traditional store-and-forward routing methods by allowing coding of two or more packets together. From an information flow aspect, multiple flows could be overlapped in a routing scheme. Hence, the theoretical upper bound of multicast capacity could be achieved by network coding. In this project, a complete routing and coding scheme is constructed to realize the maximum multicast transportation task. In order to implement the scheme, the paths of multiple max-flows are determined. Edges are divided into overlapped and normal type based on the merged max-flows. The transmitting data is represented using packets in a specific format. Multicast, forward and coding operations are defined to transmit data at the nodes. The nodes are classified according to the type of operations. A dynamic coding and routing algorithm is proposed to route packets gradually from source node to destinations in topological sorting order by the three operations on the path of merged max-flows. We show that the use of simple xor operations can satisfy most of network topologies. The running time of the algorithm presented here is less than one second for most of the benchmark and random data sets.

1 Introduction

In the past half century, information flow being transferred in a network is similar to highway transportation or water in a pipe system. Packets in the digital network or signals in the analogue network were transmitted independently under different transportation streams without overlap. However, unlike cars or water, information can be recombined in a transport system so that the path of different information can be overlapped in order to optimise arrival times.

Network Coding (NC) is a recent research area in information theory [1]. It dramatically changes traditional information processing. For example, in traditional computer networks, information packets are transmitted from the source through intermediate nodes by the store-and-forward method. There is no extra processing except replication. With network coding, the operations such as xor or linear combinations between two or more different packets are allowed to join different information flows, and the original binary packet could be recombined or extracted later at the receivers. The classical butterfly network model is shown in Figure 1. One source node S intends to send both packets a and b to the target node T1 and T2. For the traditional store-and-forward method demonstrated in Figure 1(a), two channels are required between V3 and V4. By using the xor coding as shown in Figure 1(b), the bandwidth between V3 and V4 is reduced to one channel. Node T1 receives packets a and a⊕b, and packet b could be obtained by a⊕(a⊕b). Similarly packet a could be decoded at T2 with b and a⊕b. Hence, half of the bandwidth between V3 and V4 is saved.
Another example of the wireless satellite multicast communication is shown in Figure 2. Node $S$ stands for the satellite, nodes $V_1$ and $V_2$ are two ground stations. $V_1$ sends packet $a$ to $V_2$ through the satellite $S$ and $V_2$ sends packet $b$ inversely. By traditional methods, four time units are required:

1. $V_1$ sends $a$ to $S$;
2. $V_2$ sends $b$ to $S$;
3. $S$ sends $a$ to $V_2$;
4. $S$ sends $b$ to $V_1$.

If there is a satellite broadcast $a\oplus b$ to both stations at the same time, then each station could decode the other packet with the one it sent. Hence one time unit is saved.
With Network Coding, the theoretical max flow upper bound indicated by Shannon can be achieved. It was proved that applying Linear Network Coding might help to achieve the multicasting upper bound under one source multiple receivers’ situation [2]. Linear Network Coding combines the coding and routing together from the physical layer and network layer. Two problems need addressing in order to apply Network Coding in Multicasting. The first is to establish the routing for transportation and the second is to adopt the coding pattern or code algorithm.

Some methods have been proposed to resolve the second issue. Random Network Coding (RNC) is a distributed implementation and widely used in current research. Compared with other coding approaches, such as heuristic, exponential or polynomial algorithms, Random Network Coding has lower complexity and could be easily applied in real network systems. The establishment of routing is a precondition of network multicasting. In traditional IP multicasting, the routing is set up through multicast Steiner Trees [3]. The following sections provide an overview of network coding and outline a novel network coding multicast scheme.

## 2 Network Coding

The benefits of network coding are increasing multicast throughput, reducing resources usage and enhanced system robustness. The butterfly network example in Figure 1 illustrated the potential gain in network throughput. With network coding, the multicast task of two packets can be achieved in one time unit. Hence, the transportation rate for each receiver is 2. It is as if each receiver has the network to themselves. Without network coding, it requires 3 time units to finish the same multicast task of two packets as the contention between node $V_1$ and $V_2$ requires one more time unit to complete the transportation. So the transportation rate for each receiver is 1.5. It has been shown that the larger the degree of each node, the more improvement of the throughput by network coding [4]. The satellite application in Figure 2 is an example of optimizing wireless resources. Network coding also has the benefit of reducing packet loss as well. As shown in the above example, assume that the packet loss rate $r_{AB}$ between node $A$ and $B$ is 0.1 and $r_{BC}$ equals to 0.2. With packet-level Forward Error Correcting (FEC) such as fountain codes, information could be transported from $A$ to $C$ at a success rate of $(1-r_{AB})(1-r_{BC}) = 0.72$. If node $B$ could decode and re-encode all the packets it received, then overall success rate could be increased to $1-r_{BC} = 0.8$, which is the same as the maximum flow from node $A$ to $C$. [5]. In addition, by applying network coding, the side effect caused by the failure of network links or nodes is minimized. Hence, the robustness of the network link is enhanced and the cost for network management is reduced [6].

The challenges of network coding are in the following aspects. For the security issue, by the nature of recombined packets and overlapped routes, the risk of wiretapping attacks is highly reduced. On the other hand, the operations of data in the intermediate nodes might affect the authenticity of the data. In order to implement Network Coding, a high quantity of computation is required at every node in the network. For certain problems with limited conditions, an upper bound of the computation can be determined. However, for universal issues, this upper bound cannot be determined at this moment, and it is confirmed to be considerably large. As computational processing resources become cheaper, the network bandwidth becomes the critical limitation. Network Coding could dramatically increase the network throughput by using increased computational power. Not all networks however can be extracted as acyclic or directed graph such as a Token Ring network. For the cyclic graph, the constructed coding is changing all the time. It is a challenge to construct coding scheme for such a graph.

The synchronization problem deserves attention as the encoder of the middle nodes might receive multiple input data flows in a short time period. For example, the node $V_3$ in Figure 1 requires synchronism before operation on two incoming packets. Synchronism is sensitive especially for real time applications such as live voice or video transportation. In an actual application of multicasting such as media stream publication, the nodes are usually changing all the time. As a result, the coding scheme is required to be rebuilt and the decoding for other nodes is effected as well. A scalable architecture is required for this dynamic changing situation. A lot of attention has been devoted to the architecture for network coding in various types of
networks and connections, in order to achieve the maximum multicast capability. Generally speaking, the algorithms could be classified in centralized network coding and distributed network coding.

There are two multicast centralized network coding algorithms. One is an algebraic structure (Koetter and Medard, 2002) which extends the previous conclusion to universal networks and enhances the robustness. This new conclusion has been proved with ringed directed graphs by the max flow min cut theorem. One transformation matrix is used to represent the relation between input information from a source and the information received at a node. The network coding is realized by constructing such a suitable transformation matrix. The other multicast centralized network coding algorithm has a polynomial time complexity algorithm (Sanders, 2003). It simplified the construction for single source multicasting in no ring and no time delay graph. The required path set is defined by the max flow min cut algorithm. Based on the set, the operation for each node is determined. This method not only reduces the construction algorithm from exponential to polynomial, but also reduces the lower bound of the alphabet required. In addition, Fragouli indicates that it is a colour painting problem for two source multicast network coding [7]. Distributed network coding methods do not require the network topology information. The previous operation history is added to the packets so that the receiver could decode the original content from source. For example, the Random Network Coding adds the randomly generated coefficients to the head of each packet so that the sink node could decode packets without the knowledge of the entire network topology. It was shown that the success rate for decoding is acceptable, if the field size is large enough [8]. [8] provide an algebraic framework for Network Coding and show that the capacity of a multicast session (and many generalizations of that) is achievable by linear network coding.

Network Coding can be used in applications like Peer to Peer (P2P) Networks, Wireless Ad Hoc and others. Moreover, the related theory and application research contributes to Informatics, Coding Theory, Complexity Theory, Graph theory, Matrix Theory and other subjects. Application Layer multicast is an alternative method to the Network Layer Multicast. While the information in network layer multicast is forwarded by the router, the information in application layer multicast is forward by the terminal host (PC or server). As the host usually has powerful computational resources, it is a compatible environment for network coding. One typical application is P2P file sharing software Avalanche developed by Microsoft [9]. The file is divided into \( n \) blocks and encoded with Random Linear Coding at every host. As the linear coefficients are added to the encoded blocks, one host could decode the entire file once sufficient coded blocks are received. Meanwhile, the possibility of complete delivery is increased despite the joining or leaving of hosts dynamically. In fact, Avalanche applies network coding on a unique time-parameterized graph as shown in Figure 3, rather than the physical network [10]. The variable \( t \) denotes the unit time and the link stands for the delivery of packets between hosts.

![Time-Parameterized Graph of Avalanche type system](image-url)

*Figure 3. Time-Parameterized Graph of Avalanche type system*
Due to the unreliability and multicasting feature in the physical layer of wireless network, network coding could resolve the issue in the traditional routing and cross-layer design. Applied network coding in the wireless network could increase the multicast throughput, reduce the number of hops and decrease the energy required for emission. In particular, with Random Network Coding, the original data could be retrieved at the terminal nodes even if some of the intermediate nodes or links are disabled. One throughput optimization framework is proposed for multi-hop networks across network and physical layers. The network coding scheme provides for data routing and wireless power allocation [11]. Another experimental on 802.11 hardware shows that one XOR-only mechanism nearly doubles the network throughput [12].

Another reduced complexity network coding is proposed for ad hoc networks. The links are divided into two types: (a) entering relay nodes and (b) entering targets. The same capacity can be achieved by applying network coding only on the type (a) nodes and keeping the traditional routing style at all the type (b) nodes [13]. The minimum energy per bit can be achieved by network coding for mobile and ad hoc networks with linear program [4]. Finally, a simple distributed method is proposed for exchange independent information in two wireless nodes using only XOR for network coding [13].

3 Network Coding Fundamental Theory

The network $G = (V, E, C)$ is a Directed Acyclic Graph (DAG), $V$ is the set of vectors and $E$ is the set of edges. For each directed edge $<i,j> \in E$ there is a positive integer capability $C_{i,j} \in C$. The source node, in which information is generated, is denoted by $s$ and the target node, also called sink node or receiver node, is denoted by $t$. The set of all the incoming edges of one node $u (u \in V)$ is $\text{in}(u)$ and all the outgoing edges of node $u$ is $\text{out}(u)$.

3.1 Min-Cut and Max-Flow

A flow $F$ of the network is a set of positive values for each edge $<i,j>$ which satisfies:

$$0 \leq F_{i,j} \leq C_{i,j}, <i,j> \in E$$

For every node except $s$ and $t$, the incoming flow is same as the outgoing flow as described below:

$$\sum_{<j,i> \in \text{in}(v)} F_{i,j} = \sum_{<j,i> \in \text{out}(v)} F_{j,i}, v \in V \setminus \{s, t\}$$

The outgoing flow of the source node $s$ is same as the incoming flow of the sink node $t$. And this value is defined as the total value of the flow $F$.

$$|F| = \sum_{<s,i> \in \text{out}(s)} F_{s,i} = \sum_{<t,i> \in \text{in}(t)} F_{i,t}$$

A max-flow is a flow $F$ with the maximum total value $|F|$ than any other flow over the network. The cut set $U$ is a subset of node set $V$ such that the source node $s \in U$ and the target node $t \notin U$. The edges across the cut set $U$ could be represented as:

$$E_U = \{<i,j> \in E : <i,j> \in \text{out}(u) \cap \text{in}(v), u \in U, v \in V\}$$

The capacity of the cut set $U$ is the total capacity of edges in $E_U$

$$|C| = \sum_{<i,j> \in E_U} C_{i,j}$$

Similarly as max-flow, the cut set $C$ with minimum $|C|$ in one network is called min-cut. Intuitively, the min-cut is the bottleneck between node $s$ and node $t$, and any max-flow could not exceed the min-cut.
There is an example network shown in Figure 4(a), the source node is \( s \) and the sink nodes are \( t_1, t_2 \in T \), the capacity \( C_{ij} \) of each edge \( \langle i, j \rangle \) is marked besides it.

The max-flow \( F^1 \) from \( s \) to \( t_1 \) could be worked out with any one of the algorithms discussed earlier. The result is shown in Figure 4(b). The actual flow values and the capacity limitation are indicated as a pair in the form of “\((F_{ij}, C_{ij})\)”. The edges with no flow \((F_{ij}=0)\) are marked with dash lines in the graph. The flows are: 1 unit along path 0-1-3-6-9; 2 units along path 0-1-4-6-9, 2 units along path 0-2-4-7-9. Total flow value is 5.

Similarly, the max-flow \( F^2 \) from \( s \) to \( t_2 \) is shown in Figure 4(c). The flows are: 1 unit along path 0-2-5-8-10; 2 units along path 0-2-4-8-10, 2 units along path 0-1-4-7-10. Total flow value is 5. All the edges shared by both flow are marked with bold lines.

Theorem 1 [14]: (Max-flow Min-cut Theorem) the max-flow of a certain network \( G \) is exactly same as the min-cut, i.e.

\[
|C_{\text{min}}| = |F_{\text{max}}|
\]
This theorem indicates that the max-flow could achieve the min-cut bottleneck. Furthermore, one possible transportation scheme is obtained if the max-flow is worked out for one source and one sink network. It is intuitive that the max-flow is the upper bound of the capability of transportation from $s$ to $t$.

In addition, in most of network coding research, the capacity of each edge is simplified to 1, or so called unit capacity link.

### 3.2 Multicast Problem

The multicast information flow problem is where most research has taken place. In the multicast problem, there is only one source node and multiple sink nodes. All messages are available at the source node $s$, each target node $t_i$ requires all the messages. We define the max-flow from $s$ to every sink node $t_i$ is $F_{\max}^i$ ($1 \leq i \leq l$).

<table>
<thead>
<tr>
<th>Problem</th>
<th>Upper bound of Rate</th>
<th>Routing Scheme</th>
<th>Algorithm</th>
<th>Time Complexity</th>
</tr>
</thead>
<tbody>
<tr>
<td>Unicast</td>
<td>$</td>
<td>F_{\max}</td>
<td>$</td>
<td>Max Flow</td>
</tr>
<tr>
<td>Broadcast</td>
<td>$\min_{v \in V} F_{v \max}$</td>
<td>Spanning Trees</td>
<td>Polynomial</td>
<td></td>
</tr>
<tr>
<td>Multicast</td>
<td>$\min_{v \in V} F_{v \max}$</td>
<td>Steiner Trees</td>
<td>NP</td>
<td></td>
</tr>
</tbody>
</table>

*Table 1. Routing scheme and algorithm for uni/broadcast & multicast*

In the unicast case, there is one source node $s$, and only one sink node $t$, the upper bound capacity is $F_{\max}$. Standard maximum flow algorithm could be applied and the routing scheme is obtained at the same time.

In the broadcast scenario, there is one source $s$ and all the other nodes in $V$ are sink nodes. Clearly, the upper bound capacity of broadcast is limited by the sink node $t$ with the minimal $F_{t \max}$. It was proved that this capacity is achievable by spanning trees [15]. The spanning trees could be found in polynomial time and packets could be routed over them to achieve the upper bound capacity $\min_{v \in V} F_{v \max}$.

In the multicast scenario, the $l$ sink nodes $t_1, t_2, \ldots, t_l$ all belong to a set $T \subseteq V$. Similarly, the upper bound of capacity is $\min_{t \in T} F_{t \max}$. Unlike the broadcast problem, the Steiner tree for multicast is a NP-hard problem which could not be resolve in polynomial time [16]. Thus, the upper bound is not achievable until the birth of network coding. It was proved that the upper bound of multicast capacity of the network is the minimum one among the max-flows to different multicast sink nodes.

Theorem 2 [1]: For a network with capacity constraints, the upper bound of the multicast capacity $w$ is

$$ w = \min_{t \in T} |F_{\max}^t| $$

(7)

In short, “network coding makes it possible to achieve maximum throughput given by max-flow min-cut theorem, which might not be achieved if only routing is allowed.” [17]. This theorem is also known as the network coding main theorem. This upper bound capacity can be achieved by Linear Network Coding (LNC) or Random Network Coding (RNC) as discussed below.

### 3.3 Linear & Random Network Coding

Linear Network Coding is a linear combination of packets received at one node into one or more outgoing packets [2]. Assume that there is $L$ bits in each packet, every $s$ sequential bits of a packet forms one symbol over the field $F_2^s$ with each packet becomes a vector of $L/s$ symbols. Addition and multiplication could be performed over the field $F_2^s$. The result encoded packets are in the same length of $L$.

Assume that a number of original packets $M^1, \ldots, M^k$ are generated at the source node. In linear network coding, a sequence of coefficients $g_1, \ldots, g_k$ is picked up from $F_2^s$. The encoded packet is

$$ X = \sum_{i=1}^{k} g_i M^i $$

(8)
However, one encoded packet does not carry all the information of the source packets. At the decoding node, sufficient number of encoded packets and coefficients pairs \((g^1, X^1), \ldots, (g^m, X^m)\) are required in order to recovery the original packets \(M^1, \ldots, M^n\). In other words, it is required to solve the \(m\) equations and \(n\) unknowns.

\[
X^j = \sum_{i=1}^{n} g_i^j M^i, 1 \leq j \leq m \tag{9}
\]

In order to solve these equations, the coefficients vectors should be linear independent, i.e. the \(n \times m\) matrix should be full rank.

\[
G = \begin{bmatrix}
g_1^1 & \cdots & g_1^j \\
\vdots & \ddots & \vdots \\
g_i^1 & \cdots & g_i^j 
\end{bmatrix} \tag{10}
\]

The original packets \(M^1, \ldots, M^n\) could be found with:

\[
M = G^{-1} \times X \tag{11}
\]

Random Linear Coding provides a distributed manner for network coding. It was shown that if the coefficients are picked up randomly from a large enough range such as \(2^{16}\), the matrix \(G\) is full rank with the probability very close to 1 \([18]\). Random network coding is independent from the network topology. So it could be constructed even if the network topology is unknown. In addition, exponential time complexity algorithm, polynomial-time algorithm and greedy algorithm could be used to construct the coding coefficients.

### 3.4 Network Coding Routing

One node is called the merging node if there is more than one incoming links. As network coding requires more than one packet to operate, the merging node is a requirement for the coding operation. In most network coding research, the coding operation takes place at all the merging nodes.

Two methods can be used to determine the coding nodes and path for a single source multiple terminates graph. Take the butterfly network for example, as the standard max flow algorithm is designed for one source and one terminate, the max flow for the two terminates are worked out separately. A coding sub graph is generated based on the two flows and the finally coding scheme is determined \([19]\).

Another method modifies the labelling algorithm by adding the target node to the mark so that different targets can be worked out by one max flow algorithm. The time complexity is increased to \(O(tmn^2)\) where \(t\) is the number of the target nodes \([20]\).
As shown in Figure 5, the two maximum flows are listed in (a) and (b) respectively. For node V3, there are two incoming path, and both max flow shares the edge between V3 and V4, hence V3 should be the encoding node for network coding.

3.5 Example Network

The network is represented by directed acyclic graph $G = (V, E, C)$. In the multicast scenario, one source node $s \in V$ transmits $w$ data blocks simultaneously to sinks nodes in a set $T \subseteq V$, where $w$ is the maximum multicast capacity defined in Theorem 2. The problem is to find a transmission scheme with network coding that all the data blocks is received by all the sink nodes. The number of data blocks at each link is less than or equal to the capacity constrain. As network coding is a novel research field, there are few complete algorithms to construct complete coding and routing schemes. Although there is efficient algorithm for linear network coding, the large number of coding nodes requires considerable computational costs. Generic Algorithm is used to optimal the number of coding node but it is very time consuming and complicated. This research focuses on designing an algorithm to solve this problem. For a given network, a complete coding and routing scheme will be constructed as a result. Therefore, the objectives of this project are as follows.

- The maximum multicast rate of a network should be achieved.
- There is no unit capacity limitation on each edge.
- The number of coding nodes should be minimised.
- If there is no coding node required for a network, the algorithm will provide a transmission scheme with traditional forward and multicast mechanism.

One example network with one source and three sink nodes 5, 6 and 9 is shown in Figure 6(a) (Fragouli, 2007), all the edges have a unit capacity. However, there is a cycle 6-7-8-6. In order to remove the cycle, one new node 10 is introduced in Figure 6(b), edge <8,6> is removed and new edge <6, 10> and <8, 10> are introduced and node 10 is regarded as the sink node t1 instead of node 6. The max flows from s to sink nodes $t_1$, $t_2$ and $t_3$ are all 2. One network coding scheme that achieve the maximum multicast capacity is
demonstrated in Figure 6(c). According to the definitions, the coding nodes are node 2 and 7, and the multicast nodes are node 1, 3, 4 and 8.

Figure 6. Unit capacity three sink node network coding scheme

4 Algorithm for the Routing Scheme

The multicast flow graph is constructed by merging all the max-flow paths to every sink node. Second, all the data blocks are packed into packets along with its destination node index. Finally, these packets are dispensed from source node to sink nodes through forward, multicast or coding operations.

4.1 Multicast Flow Graph

The max flow of one graph can be found out through two types of algorithm: augmenting path methods or pre-flow-push methods. Augmenting path method, also known as the Ford Fulkerson method, is based on the idea of finding out one path that could enhance the current flow, until no path could be found. Let |E| = n, |V|=m, and assume maximum edge capability is u. The general labelling algorithm is finding any possible path with the time complexity of O (nmu). The capacity scaling algorithm limited the path to be the one with maximum increasing improvement and reduces the time complexity to O (nm log u). Another shortest augmenting path algorithm, the Edmonds Karp algorithm, discovers the path with the smallest number of nodes, e.g. the Breath First Search (BFS). The time complexity is O (nm²).
The two max-flows to sink node 5 and sink node 6 are shown in Figure 7(a) and (b), respectively. In this figure, flow values \( F_{ij} \) and capacity values \( C_{ij} \) are separated by slash “/” and placed beside each edge. The edges with non zero flow are marked with same colour with the sink node. Both sink nodes have the same maximum flow value 2. The two paths to sink node 5 are 0-1-5 and 0-2-3-4-5. Similarly, the two paths to sink node 6 are 0-1-3-4-6 and 0-2-6.

The unit capacity of each link is a common assumption in most network coding approaches. Larger capacity is substituted with multiple parallel links in the same direction between two nodes. However, link capacity in real networks is always a quantity in a large number instead of 1. The approaches based on unit capacity encounter a problem to deal with a large number of links with unit capacity. Hence the computational complexity is increased dramatically. The proposed approach here avoids this problem. Since links are not unit capacity, the new definitions are proposed as follows.

**Definition 1:** The merged max flow to all the sink nodes is defined as

\[
F_{i,j}^* = \max(F_{i,j}^d), 0 \leq d < l, t_d \in T, <i, j> \in E
\]

That is to say, for every edge, the merged max flow is the maximal one of the single max flows overlapping on the edge. The value of the merged max flow is used as the new limitation of the edge capacity.

**Definition 2:** The overlapped edge is the edge used in two or more max-flows to different sink nodes, i.e. the edge \(<i, j>\) satisfies

\[
F_{i,j}^l > 0 \land F_{i,j}^k > 0 \ (1 \leq l < k \leq l, k \neq t, <i, j> \in E)
\]

For the example in Figure 7, the combined maximum flow for the two max-flows is shown in Figure 8. The overlapped edges are marked with bold lines. The numbers beside the edges are the merged capacity limitations.

According to Theorem 2, the upper bound multicast transportation capacity is the minimum one of the two single max-flows. For this example, it is \( \min(2,2)=2 \), i.e. there are 2 different packets that can be broadcast.
from $s$ to $t_1$ and $t_2$ in one time unit if network coding is used. At the end of this algorithm a network coding scheme under this capacity limitation will be constructed. The merged flow graph simplifies the network topology. Only the edges and nodes in the merged max-flows will be used to construct the coding and routing scheme.

**Figure 8.** Merged max flows for butterfly network

Definition 3: The target set of edge $<i,j>$ is denoted as $D_{i,j}$ for any sink node $t$ in $D_{i,j}$, the edge $<i,j>$ is on the max-flow from $s$ to $t$. The sink nodes in the target set is represented with its index number. The formal definition is as following:

$$D_{i,j} = \{d : F_{i,j}^d > 0, t_d \in T, 0 \leq d < l, <i, j> \in E\}$$

$T$ is the set of all sink nodes

$l$ is the number of sink nodes

$F_{i,j}^d$ is the flow value of edge $<i,j>$ to sink node $t_d$

For unit capacity, as $C_{i,j} = 1$, $F_{i,j}^d > 0$ is equivalent to $F_{i,j}^d = 1$

$$D_{i,j} = \{d : F_{i,j}^d = 1, t_d \in T, 0 \leq d < l, <i, j> \in E\}$$

For example, the target set $D$ for butterfly network is shown in Figure 9. It could be derived from Figure 7. There are two sink node 5 ($index=0$) and sink node 6 ($index=1$). Edge $<0, 1>$ is shared by both max-flows, so the target set $D_{0,1}$ is $\{0, 1\}$; Edge $<1, 3>$ is on the max-flow of node 6 only, so the target set $D_{1,3}$ is $\{1\}$. The other edges could be done by parity of reasoning.
For multiple capacity edges, there are multiple target sets at each edge \( <i,j> \). The number of target sets is the flow value of the merged flow \( F_{i,j}^* \). Each target set is defined as follows:

\[
D_{i,j}^k = \{d : k + 1 \geq F_{i,j}^{d}, t_d \in T, 0 \leq d < l \}, <i,j> \in E, k \leq F_{i,j}^*
\]

Intuitively, the multiple capacity edge is treated as several parallel edges, and each edge is assigned a target set. For example, if the capacity of each edge at the butterfly network is doubled. There are two target sets for each edge as shown in Figure 10.

\[Figure 9. \text{ Target sets for butterfly network}\]

\[Figure 10. \text{ Target set for double capacity edge of butterfly network}\]
4.2 Packet Representation

The task of multicasting on a network can be regarded as a performance in which all the information packets should be transferred to multiple sink nodes. For simplicity of implementation, suppose that a packet is composed of two parts: target set and data block. Target set of a packet contains a set sink nodes where the packet should be sent to. The data block is a information unit carried by the packet. A packet \( p \) is represented as the format shown as follows:

\[
p = (\text{data block list}, \text{target set})
\]

The data block is a fixed length data section. For simplicity in algorithm design, one data block is labelled with a letter from “\( a \)” to “\( z \)”. The multiple data blocks in a data block list are linear combined (this refers to network coding) and the length is equal to one data block. A packet can carry a single data block or a data block list (coded data block) and single or multiple destinations in its target set. The data block can be coded or decoded, and the target sets can be split or merged. The element in target set of a packet is the index number of sink nodes, e.g. 0, 1, 2, ..., \( l \). For example, one packet of data block \( a \) sending to the first and second sink node is represented as “(\( a, \{0, 1\} \))”. The target set of packet \( p \) is represented as \( D(p) \) and the data block list of packet \( p \) is represented as \( B(p) \). At the beginning, there are \( w \) data blocks at the source node (\( w \) is the maximum multicasting capacity of the network). Every data block should be transferred to every sink node. So there are \( w \) packets, every packet consists of a unique data block and a full target set \{\( i: 0<i<l-1 \}\}. For example, for a network with \( l = 3 \) and \( w = 2 \), all the original packets are listed below:

\[
(a, \{0, 1, 2\}), (b, \{0, 1, 2\})
\]

There are two operations on the packets: split and coding.

The split operation divides the target set of one packet according to the specified subset \( ts \), and the data block list is unchanged. The pseudo code is shown below:

```plaintext
1 procedure split(packet p, target set ts)
2 begin
3 new packet q ← (B(p), ts)
4 new packet r ← (B(p), D(p) - ts)
5 return q, r
6 end
```

For example, packet “(\( a, \{0, 1, 2\} \))” could be split to “(\( a, \{0\} \))” and “(\( a, \{1, 2\} \))” with a specified subset \{0\}.

The coding operation joins the data blocks of two packets together. The target sets are union at the same time. The pseudo code is shown below:

```plaintext
1 procedure coding(packet p, packet q)
2 begin
3 p ← (B(p) + B(q), D(p) ∪ D(q))
4 return p
5 end
```

For example, coding two packets “(\( a, \{0\} \))” and “(\( b, \{1\} \))” will result in “([\( a, b \)],\{0,1\})”. If xor operation is applied on the data blocks, the list “[\( a, b \)]” corresponds to “\( a \ xor \ b \)”.

4.3 Algorithm for the Coding and Routing Scheme

At the beginning, all the packets are generated at the source node. The idea is to transmit all the packets along the max-flow paths to its target nodes. In order to avoid back-trace, all the nodes are visited only once by the topological ordering. When visiting one node, all the packets at that node are transferred to its successor nodes through an outgoing edge by using three operations one by one: forward operation, multicast operation and coding operation. Each operation might process partial or all the packets at that
node. After the forward operation, the multicast operation is performed, only if there are any packets left. If there are still any packets after multicast operation, coding operation is applied to handle all the remaining packets.

For directed acyclic graph (DAG), topological ordering is a linear ordering of all the nodes. All the nodes in the list do not have outbound edges to any node in front of it. Obviously, the source node could be the first node. Thereby, in this upstream to downstream ordering, the coding scheme could be constructed to ensure that all the packets are passed onto the sink nodes eventually. The topological ordering could be obtained by the topological sorting algorithm (Kahn, 1962). The algorithm recursively chooses a node which has no incoming edges, puts it the output list and removes all the outgoing links of that node. Such a node will always exist unless the graph has a cyclic. The time complexity is linear as all the nodes or edges would be accessed once, $O(|V|+|E|)$. The pseudo code of the algorithm is described below:

```plaintext
1 L is the list that will contain the sorted elements
2 S is set of all nodes with no incoming edges
3 4 L ← Ø
5 S ← {s}
6 while S is non-empty do
7 pop a node i from S
8 insert i into L
9 for each node j with an edge e from i to j do
10 remove edge e from the graph
11 if j has no other incoming edges then
12 insert j into S
13 if graph has edges then
14 output error message (graph has at least one cycle)
15 else
16 output message (proposed topologically sorted order: L)
```

For example, the topological ordering for butterfly network is 0, 1, 2, 3, 4, 5, 6. Another topological sort algorithm based on Deep First Search is discussed in [21].

Using the forward operation, the packets are transferred to the outgoing edge with the exactly same target set. Usually forward operation can transfer all the packets at the nodes which the sum of incoming flows is same as the sum of outgoing flows on the merged flow graph. The pseudo code of the forward operation is:

```plaintext
1 procedure forward_operation(node i)
2 begin
3 for each packet p in node i
4 for each outgoing edge <i,j>
5 if packet target set equals to edge target set then
6 send packet p to node j through edge <i,j>
7 end
```

For example in Figure 11, at the source node of butterfly network, the packet $(a, \{0, 1\})$ is transferred to edge $<0, 1>$ with target set $\{0, 1\}$ and packet $(b, \{0, 1\})$ is transferred to edge $<0, 2>$ with target set $\{0, 1\}$.

\[
(a,\{0, 1\}) \quad (b,\{0, 1\})
\]

\[
0 \quad \{0, 1\} \quad \{0, 1\}
\]

\[
1 \quad \{0, 1\} \quad (a,\{0, 1\})
\]

\[
2 \quad \{0, 1\} \quad (b,\{0, 1\})
\]

Figure 11. Forward operation at source node
After the forward operation, all the packets with fully matched target set are allocated. So the remaining packets may be only partially matched with the target set for some edges. By enumerating all the pairs of packet and edge, the pair with maximum intersection of the target set is chosen. Then the packet is split into two packets, with one packet to be sent along the edge and the other remained at the node. This operation is repeated until there is no matching pair of packet and edge pair.

As shown in Figure 12(a), there is a multicast operation at node 1 of the butterfly network. The best matched edge for packet \((a,\{0, 1\})\) is \(<1,5>\), the intersection of target sets is \(\{0\}\). Hence, packet \((a,\{0, 1\})\) is split into two packets \((a,\{0\})\) and \((a,\{1\})\), then \((a,\{0\})\) is transferred to the outgoing link \(<1,5>\) and \((a,\{1\})\) is retained at the node 1. Thereafter, packet \((a,\{1\})\) is found to be matched with edge \(<1,3>\) with the same target set \(\{1\}\), hence it is sent to node 3. In the coding scheme, this multicast operation is illustrated in Figure 12(b). The router at node 1 replicates the data block \(a\) into two packets with different target set and send them to two outgoing links.

![Multicast operation example](image)

Figure 12. Multicast operation example

The coding operation resolves the remaining packets that could not be handled by both forward and multicast operations. Similar to the multicast operation, the coding operation iteratively find the best matched target set between packets and outgoing links. The difference is that coding operation encodes two data blocks into one data block list to make a new packet. Two packets become one packet and take one unit of the flow capacity. One example of a coding operation in the butterfly network is shown in Figure 13(a). There are two packets \((a,\{1\})\) and \((b,\{0\})\) arriving at node 3. However, there is only one outgoing link with unit capacity and target set \(\{0, 1\}\). Hence, two packets are required to be combined together. Firstly, packet \((a,\{1\})\) is sent to node 4 through \(<3,4>\) by multicast operation. At this time, the \(D_{3,4}=\{0,1\}-\{1\}=\{0\}\). Next, by a coding operation, packet \((b,\{0\})\) is going to be coded with packet \((a,\{1\})\). The new data block list is \([a,b]\) and the target set for new packet is \(\{1\} \cup \{0\}=\{0,1\}\). If xor is used as coding operation, the coding scheme is shown in Figure 13(b). In the realistic scene, the router at node 3 working out the xor result of the data block \(a\) and data block \(b\), then sends the result to the only outgoing link \(<3,4>\).
After performing the operations illustrated above, to all the nodes in topological order, the coding scheme for butterfly network is constructed as shown in Figure 14.

Based on the operations in a node, all the nodes can be classified into three types according to the type of operations performed on the node. Therefore, node types are defined as follows:

Definition 4: A node is called forward node if only forward operation is used to transfer all the received packets at the node.

Definition 5: A node is called multicast node if at least one multicast operation should be performed to transfer all the received packets and not any coding operation is required to perform at the node.

Definition 6: A node is called coding node if at least one coding operation is required to transfer all the received packets at that node.

Based on these definitions, the actual coding nodes can be found using the proposed algorithm.
To develop a visualisation of the actual network coding algorithms, we developed a tool using Python. Python is an open-source dynamic interpreted language with a considerable amount of free third party modules programs and tools running on Windows, Linux/Unix and Mac OS X. We also incorporated Graphviz which is open-source graph visualization software for representing structural diagrams of abstract graphs or networks. Graphviz takes descriptions of a graph in a text description language (Dot) and designs the layout of diagrams. Pydot is a python interface to the Graphviz Dot language. It allows the creation of figures for directed graphs from Python data structures. The adjacency matrix is used to represent the graph in the final implementation. For a network with \( n \) nodes, one \( n \times n \) matrix \( \text{net} \) is used to store the edges and capacity limitation. An edge is represented with capacity as follows:

\[
\text{net}_{i,j} = \begin{cases} 
0, & \text{if no edge } <i,j> \\
c_{i,j}, & \text{capacity for edge } <i,j>
\end{cases}
\]

A flow through edge \(<i,j>\) is represented as

\[
\text{flow}_{i,j} = \begin{cases} 
0, & \text{if no edge } <i,j> \text{ or no flow} \\
f_{i,j}, & \text{flow value for edge } <i,j>
\end{cases}
\]

A merged flow through edge \(<i,j>\) is represented as

\[
\text{merged}_{i,j} = \begin{cases} 
0, & \text{if no edge } <i,j> \text{ or no flow} \\
f_{i,j}^*, & \text{merged flow value for edge } <i,j>
\end{cases}
\]

The python code to enumerate all the edges of the network is:

```python
1 for i in range(n):
2   for j in range(n):
3     if net[i][j]:
4       do something for edge <i,j>
```

The code to enumerate all the incoming edges of the node \( i \) is:

```python
1  # at node i
2  for j in range(n):
3    if net[j][i]:
4      # do something for edge <j,i>
```

Similarly, the code to enumerate all the outgoing edges of the node \( i \) is:

```python
1  # at node i
2  for j in range(n):
3    if net[i][j]:
4      # do something for edge <i,j>
```

One path is represented with an \( n \) element predecessor array \( \text{pred} \). \( \text{pred}[i] \) which stores the previous node of node \( i \) in the path. The traverse of the path starts from the last node \( t \) until the first node \( s \). The python code is listed below:

```python
1 i = t
2 while i != s:
3   j = pred[i]
4   # do something for edge <i,j>
5   i = j
```

The first line contains four positive integers: \( n \) giving the number of nodes; \( m \) giving the number of edges; \( s \) giving the source node id and \( l \) giving the number of sink nodes. The second line contains \( l \) integers, giving the id of sink nodes. Then followed by \( m \) lines, each line contains three positive integers: \( x \) stands for the
source node of edge; y stands for the end node of edge and c stands for the capacity of edge \( <x,y> \). Log files include all the intermediate matrices, coding scheme results, warning and error messages. Furthermore, there are six types of output images files:

1. Input network
2. Max-flow to each sink node (Figure 7)
3. Merged max-flow (Figure 8)
4. Target sets at each edge (Figure 9)
5. Coding scheme with data blocks and target sets(Figure 14)
6. Coding scheme with xor operation

In order to distinguish different flows to different sink nodes, “paired12” colour scheme provided by Graphviz is chosen as shown in Figure 15. Each sink node and the related max-flow are assigned an identical unique colour.

![Figure 15. Colour scheme for different max-flows](image)

The nodes are represented with unique shapes. As shown in Figure 16, the normal circle shape represents a forward node; double circle represents a multicast node and a double octagon represents a coding node.

![Figure 16. Node shapes of output images](image)

As shown in Figure 17, two arrow shapes are selected: crow is used for overlapped flow to multiple sink nodes and normal arrow is used for single flow to single sink node.

![Figure 17. Arrow shapes of output images](image)

The program loads the input file and constructs a coding and routing scheme for the network corresponding to the input file. The program also provides coding node number and link number as well as other statistics.

## 5 Testing and Experimental Results

The max flow program works out the max flow value from one source node to one sink node. It reads the network from text files and writes the output to another text file. It was tested using 30 test cases. The format of input and output files are described as follows.

The first line of input file contains four positive integers separated by space: \( n \) \( m \) \( s \) \( t \). \( n \) is the number of nodes; \( m \) is the number of edges; \( s \) is the source node and \( t \) is the sink node. Then followed by \( m \) lines, each line contains three positive integers: \( x \ y \ c \). \( x \) is the source node of edge; \( y \) is the end node of edge and \( c \) is the capacity of edge \( <x,y> \). The first line of output file is one integer number of the maximum flow value.
Each of the following line represents one flow in the format of “(x, y) f”, it means that the flow value is f at edge <x, y>. The test cases are listed below:

<table>
<thead>
<tr>
<th>Test Case 1</th>
<th>Test Case 2</th>
<th>Test Case 3</th>
<th>Test Case 4</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input File</td>
<td>Output File</td>
<td>Input File</td>
<td>Output File</td>
</tr>
<tr>
<td>4 5 0 3</td>
<td>6</td>
<td>8 1 2 0 7</td>
<td>8</td>
</tr>
<tr>
<td>0 2 4</td>
<td>(0, 1) 2</td>
<td>0 1 7</td>
<td>(0, 1) 6</td>
</tr>
<tr>
<td>0 1 2</td>
<td>(0, 2) 4</td>
<td>0 5 2</td>
<td>(0, 5) 2</td>
</tr>
<tr>
<td>1 2 3</td>
<td>(1, 2) 3</td>
<td>1 2 3</td>
<td>(1, 6) 3</td>
</tr>
<tr>
<td>1 3 1</td>
<td>(1, 3) 1</td>
<td>2 3 3</td>
<td>(2, 7) 3</td>
</tr>
<tr>
<td>2 3 5</td>
<td>(2, 3) 5</td>
<td>2 7 3</td>
<td>(4, 7) 2</td>
</tr>
<tr>
<td></td>
<td></td>
<td>3 0 3</td>
<td>(5, 4) 2</td>
</tr>
<tr>
<td></td>
<td></td>
<td>3 4 4</td>
<td>(6, 7) 3</td>
</tr>
</tbody>
</table>

Thirty networks were created as test cases for the proposed algorithm (except for the butterfly network example). All the cases have obtained optimal coding and routing schemes successfully by our algorithm. Six typical networks among them are listed below and some comments are given for the obtained coding and routing scheme.

As shown in Figure 18, the topology on the left is the same as the butterfly network, however, all the capacity of the edges are doubled. The coding scheme is shown on the right. There are two coded data blocks “d+b” and “c+a”, they could be decoded with single data block “d” and “c” at node 5, or a single data block “a” and “b” at node 6.

As shown in Figure 19, the source node is acting as a coding node. Coded data block “a+b” is then multicast by node 3 and received by sink node 5 and node 6. Both sink nodes could decode it with another single data block “a” or “b”.

---

As shown in Figure 20, the coded data block might be coded again. The coded data block $a+b$ is coded to $a+b+b$ at node 7. Actually, $a+b+b$ is equals to $a$. Node 9 could decode $a+b$ with $a$ and node 10 receives both $a$ and $b$.

For some networks, network coding is not necessary to achieve the maximum multicast capacity. As shown in Figure 21, the maximum multicast capacity 5 is achieved by forward and multicast operation with no coding operation.
Figure 22 shows an example which has more than two sink nodes.

As shown in Figure 23, a coding scheme does not utilise all the merged flow as some max-flow to single sink node might exceed the multicast capacity. Flows 0-11 are not used because max-flow from 0 to 11 is 3, which is larger than the multicast capacity 2.

In order to test the network coding solution, random topologies are generated. The number of node \( n \), edges \( m \), number of sink nodes \( l \) and maximum capacity \( c \) of each edge are specified. There are \( m \) edges, each edge is started and ended at two different random selected nodes, with a random integer capacity no more than \( c \). As an acyclic graph is required, the Floyd-Warshall algorithm is applied to detect and break the cyclic. Firstly, the standard Floyd-Warshall algorithm is applied to work out the shortest path among all the node pairs. Then it is checked to see if there is a path starting and ending at the same node, if there is, a cycle is detected. In order to break a cycle, the path reconstruction information is recorded when finding the shortest path. If a cycle is detected, then one edge on the path of the cycle is removed to break that cycle.
As there might be more than one cycle covering the same node, and this cycle break method can break only one cycle at a time, therefore it has to iterate until there is no more cycles detected. In addition, in order to keep the edge distribution balanced, when breaking the cycle, the order of node visitation can be shuffled. One randomly generated network example is shown in Figure 24, the source node is 0 and the sink nodes are 8 to 19. Each of the sink nodes are filled with different colours. The capacity of each edge is marked at the side of it.

![Random multicast network](image)

**Figure 24.** Random multicast network

### 5.1 Validation of the Coding and Routing Scheme

Capacity validation ensures that the number of data blocks at each edge does not exceed the capacity constrain of that edge. This test has been performed in all the experiments. The results show that all the test cases have passed this test. Decoding validation checks if all the single data blocks could be decoded at every sink node after all the packets arrive at the sink nodes.

Assuming xor is used as the coding operation.

Property 1: \( a \oplus a = 0 \)

Property 2: \( a \oplus 0 = a \)

\[
\begin{align*}
a \oplus b & \oplus a \\
= a \oplus a \oplus b \ (\text{commutative law}) \\
= 0 \oplus b \ (\text{Property 1}) \\
= b \ (\text{Property 2})
\end{align*}
\]

At one sink node, \( S_1, S_n \) are the data blocks in the data block list of packet \( P_i \). Assuming xor is used as coding operation, the data block of packet \( P \) is \( S_1 \oplus S_2 \oplus \ldots \oplus S_n \). If \( S_i \) is the only data block in the data block list of another packet \( P_j \), then \( S_i \) could be eliminated from the data block list of \( P_i \) by operation \( S_i \oplus S_1 \oplus S_2 \oplus \ldots \oplus S_{i-1} \oplus S_{i+1} \oplus \ldots \oplus S_n \).
The decoding is an iteration process. It repeatedly uses a single data block to eliminate itself from the data block list of all the other packets until there is no single data block available. If every data block list contains only one data block then decoding is successful.

In our experiments, the proposed approach has been applied to 901 valid random networks with 20 nodes. 901 coding and routing schemes have been obtained. Among these schemes, if xor is used as coding operation, only 4 coding and routing schemes could not be decoded completely at the sink nodes. In other words, the decoding success rate for the xor-only coding scheme is 99.5% in this experiment. For these networks which could not be decoded completely, linear coding operation could be used.

### 5.2 Results

In most of the random topologies generated, network coding is not necessary to achieve the maximum multicast capacity. Even in the topological network with network coding, the network coding is taking place in only a few links and nodes. The proposed approach can be used to obtain an optimal coding and routing scheme for any network. The strategy in the approach leads to the optimal coding and routing scheme with less number of coding nodes. The merging node rule is usually used to identify coding nodes in many network coding approaches. As shown in Table 2, \( n \) is number of nodes, \( m \) is number of edge, \( l \) is number of sink nodes, \( w \) is number of packets and \( t \) is number of test cases for the specified \( n, m, l \) and \( w \). The number of coding nodes identified by our algorithm is significantly less than the number of merging nodes.

<table>
<thead>
<tr>
<th>Random Network</th>
<th>Coding Node Number of Proposed Methods (avg.)</th>
<th>Merging Node Number (avg.)</th>
</tr>
</thead>
<tbody>
<tr>
<td>20</td>
<td>0.5</td>
<td>5.6</td>
</tr>
<tr>
<td>30</td>
<td>0.2</td>
<td>9.0</td>
</tr>
<tr>
<td>40</td>
<td>0.875</td>
<td>12.875</td>
</tr>
</tbody>
</table>

Table 2. Coding node number of method and merging node number

Tao proposed a modified max flow algorithm in order to minimise the cost of network coding by reducing the number of key links (overlapped edges). In Figure 25, the coding scheme on the left identifies the coding nodes by the merging node rule. The coding scheme on the right uses their algorithm. The number of coding nodes is reduced by 50% [22].

![Figure 25. Minimal cost network coding example [22]](image)

The coding scheme obtained by the algorithm outlined here for the same topologies is shown in Figure 26. There is only one coding node which is the same as the results found by [22].
In order to compare results, the networks with the same parameters as Kim’s experiments are applied to test the proposed approach \[23\]. As shown in Table 3, the random network with parameters (20 nodes, 40 links, 12 sinks, 4 packets) and (40 nodes, 120 links, 12 sinks, 3 packets) is chosen in our experiments. The benchmark data (322 nodes, 1096 links, 4 sinks, 3 packets) is a real network topology of a European backbone (Ebone) from the Rocketfeul project \[24\].

The first line shows the results of our approach. The following results are the Generic Algorithm (GA) by Kim \[25\], “Minimal 1” by Fragouli et al and “Minimal 2” by Langberg \[26\]. It shows that our algorithm obtained the best number (zero) of coding links for all of the three networks. Furthermore, the proposed approach can obtain the best result in one instance of execution (hence there is no average result in the table). The best result of the random GA approach is obtained by 20 trails of running \[25\].

With regards execution time, the Generic Algorithm requires at least 15.4 seconds per generation and 1000 generations for a network with 40 nodes \[25\]. It takes less than one second for all 40 nodes network using the proposed approach in this thesis.
6 Conclusion

A novel Max-Flow-Minimum-Cut algorithm to construct a complete coding and routing scheme is outlined. Using the proposed approach, the number of coding nodes and links can be minimized and the running time of the algorithm is extremely short. The dynamic nature of transmitting packets from node to node could be easily applied at routers on existing networks. The experimental results show that based on our algorithm, most of the coding schemes could simply apply the xor as a coding operation.

We increased the assumption capacities of links from unit to integer, a set of new definitions and concepts are introduced for network coding including merged max-flows, overlapped edge, target set of packets, target set of edges, forward operation, multicast operation and coding operation. We introduced the novel concept of building a multicast scheme by following the max-flow paths to every single sink node. The original network is simplified to merged max-flow graph. An efficient algorithm is proposed to construct a network coding and routing scheme on the simplified network. We introduced the strategy of best-matching target sets between the packet and edge. This mechanism enables the algorithm to maximise the efficiency of every edge and reduce the overall network resource usage. Finally, we demonstrated a forwarding-multicasting-coding mechanism to minimise the number of coding operation. Consequently, the algorithm identifies coding nodes dynamically during the process of constructing the coding and routing scheme.

References
