ISSN 2047-3338

# Comparisons of Network Coding with Basic Routing Techniques

Naushad M. PEYRYE and Mussawir A. HOSANY

***Abstract*—The Internet of today is based on a store-and-forward mechanism for packet transfer where routers have the ability of only forwarding packets to most appropriate links without modifying the data in the packets. However with the advent of Network Coding, data are encoded and decoded by routers before packets are forwarded, in view to reduce the bandwidth and thus to improve throughput. This paper presents the network coding principle as practically conceivable for routing by comparing it with basic static routing techniques through simulation. The result shows that network coding performance varies with network size and density. We also investigated a dynamic version of selective flooding and selective network coding through Routing State Information Ageing (RSIA) and the result shows some improvements to Flooding and inherently, but to some lower extent, to Network Coding.**

***Index Terms*—Network Coding, Routing, Flooding, Shortest Path and RSIA**

## I. Introduction

TODAY'S networking devices work on the assumption that routers and other relaying devices can only forward bunch of data to most appropriate lines without information encoding. This scenario makes the relaying nodes' operations simple and such devices require relatively small processing and memory capabilities. However, with the increase in hardware capabilities at reduced cost, routers can be equipped with higher capabilities allowing the prospect for network coding.

Network coding is an approach whereby intermediate nodes within a network encode and decode information in an attempt to improve the throughput utilization of that network topology [1], [2].

Network coding has opened up new possibilities or methods to transfer information from a source node to a destination node. Network coding is considered as a revolutionary technology onto which the speedier Internet of tomorrow will be fully reliant on [3].

Naushad M. Peyrype is with Infosys Limited, Mauritius, (Email: naushad_peyrye@infosys.com)

Mussawir A. Hosany is with the Department of Electrical and Electronic Engineering, Faculty of Engineering, University of Mauritius, Mauritius, (Email: m.hosany@uom.ac.mu)

Recent researches on Network Coding showed that it can achieve bandwidth savings [1], max-flow from source in multicast scenarios through linear coding [2]. However, we still need to show, in practice, how Network Coding behaves when compared to other routing techniques.

Network Coding is still at rudimentary stage, and most implementations are based on flooding or multicasting principles to distribute combined information packets. In other terms, Network Coding has been thought of as a static solution for routing. However, adding some congestion control technique to routing in network coding may lead to a more efficient network coding routing solution. This can be achieved by using a dynamic selective version of flooding thought Routing State Information Ageing (RSIA).

The goals of this paper are to:

- Show the configurations where Network Coding is more suitable as a routing solution when compared to basic routing algorithms like flooding and shortest path.
- Show how basic Network Coding can be improved in a dynamic environment through RSIA.

Section II illustrates the Network Coding Routing Principle with an example. Section III discusses the RSIA improvement techniques applicable to Flooding and Network Coding. Section IV describes the Experimental Set-up and section V contains the simulation results and discussions.

## II. Network Coding Routing Principle

### *A. Network Coding Example*

Basically a node in a network coding scheme encodes all the received information from the input links into a single encoded packet and forward it to all the possible output links [4]. As a packet propagates down the network, it gets encoded with the packets available at each node to form a new encoded packet. When a node receives an encoded packet, it first checks whether the packets available with this encoded packet have already been received previously. If so, this encoded packet is discarded. Otherwise, it is decoded and all the new information received is queued in a list of received packets. Finally all the received packets from the list are encoded into a single packet and this packet is sent to the neighboring nodes [1].

Consider the following network scenario with node A and node B having to send two different packets pk-A and pk-B to

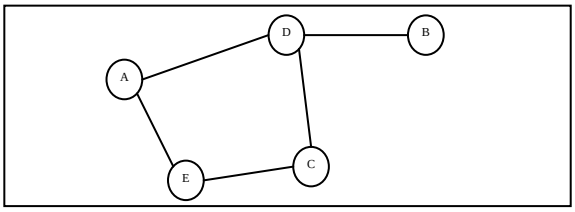


Figure 1: Network Example

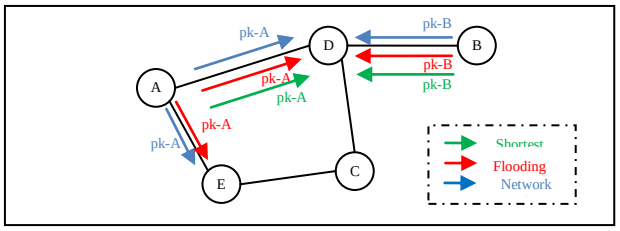


Figure 2: Network Example after flow iteration 1

| Table 1: Nodes Information after iteration 1 | | | |
|---|---|---|---|
| *Nodes* | *Shortest Path* | *Flooding* | *Network Coding* |
| A | pk-A | pk-A | pk-A |
| B | pk-B | pk-B | pk-B |
| C | | | |
| D | pk-A, pk-B | pk-A, pk-B | pk-A, pk-B |
| E | | pk-A | pk-A |

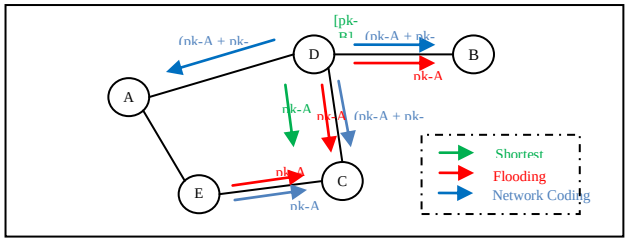


Figure 3: Network Example after flow iteration 2

node C, and assuming similar edge weights, and no delay at intermediary nodes. For shortest path routing from node A to C, we will consider route ADC to be the chosen shortest path.

The packet flows for the first hop is shown in Fig. 2.

At this stage, the data present at the different nodes are (Table 1).

The packet flows for the second hop is shown in Fig. 3.

Please note that during the second hop, packet pk-B is kept in the out buffer at Node D for edge DC for both Shortest Path and Flooding, to be sent out in the next hop. After the second hop, the packets present at the different nodes are (Table 2).

Since with network coding, node C have the ability to decode pk-B from (pk-A + pk-B) and pk-A, no further hop is required. However, with both shortest path and flooding, one more hop is required for both packets pk-A and pk-B to reach node C.

| Table 2: Nodes Information after iteration 2 | | | |
|---|---|---|---|
| *Nodes* | *Shortest Path* | *Flooding* | *Network Coding* |
| A | pk-A | pk-A | pk-A, (pk-A + pk-B) |
| B | pk-B | pk-A, pk-B | pk-B, (pk-A + pk-B) |
| C | pk-A | pk-A | pk-A, (pk-A + pk-B) |
| D | pk-A, pk-B | pk-A, pk-B | pk-A, pk-B |
| E | | pk-A | pk-A |

### *B. Packet Encoding*

Consider a set of original messages $M_1$, $M_2$,…, $M_n$ arriving at an encoding node. Each message $M_i$ is associated to a unique encoding vector $g = \{g_1, g_2,\ldots,g_n\}$ [5]. The Information Vector for the encoded data is achieved by first concatenating the encoding vector to the message and for all positions merging the characters using, for example, XOR operation.

$$X_i = \sum_{i=1}^{n} g_i M_k^i \quad [3]$$

Consider a set of three messages,

$M_1 = 100$

$M_2 = 011$

$M_3 = 000$

gM can be represented as:

$$gM = \left[\begin{array}{ccc|ccc} 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 0 & 0 & 0 \end{array}\right]$$

Decoding all encoded messages will require at least n number of different messages with no linear dependency among them. However, it can also be possible that having only two encoded messages (both containing multiple packets) may be enough to decode a single packet. Since it is required to keep all innovative messages, we will use a decoding matrix structure Dm, and decoding can be achieved by using Gaussian Elimination [5].

For each new row added to Dm (either received or inferred packet), we need to apply the XOR operation to every other row in Dm to obtain a new potential packet p. The following rules then apply:

1) The decoded packet is discarded if

$$\exists_i (\forall_j (a_{ij} = p_j))$$

2) The decoded packet is discarded if

$$\sum_{j=1}^{|g|} p_j = 0$$

3) The decoded packet is an original packet if

$$\sum_{j=1}^{|g|} p_j = 1$$

Consider that at the receiving end, we first receive X.

$$Dm = \left[\begin{array}{ccc|ccc} 1 & 1 & 1 & 1 & 1 & 1 \end{array}\right]$$

No informative packet can be inferred yet, since $\exists i \left(\sum_{j=1}^{|g|} a_{ij} = 1\right)$ does not hold. Now we receive another packet ($M_1 + M_2$) as 110111.

$$Dm = \left[\begin{array}{ccc|ccc} 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 1 & 1 & 1 \end{array}\right]$$

By applying XOR, we can infer p as 001000. Since this fails rules 1 and 2, it is added to Dm.

$$Dm = \left[\begin{array}{ccc|ccc} 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 0 & 0 & 0 \end{array}\right]$$

P passes rule 3, therefore is an original packet ($M_3$). Furthermore, applying XOR operation on p to the rows in Dm fails for rules 1 and 2, therefore no new packet is inferred. Another packet ($M_2 + M_3$) is received as 011011.

$$Dm = \left[\begin{array}{ccc|ccc} 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 1 & 1 \end{array}\right]$$

By applying XOR to all other rows in Dm, we have

$$Dm = \left[\begin{array}{ccc|ccc} 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 1 & 1 \\ 1 & 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 1 \end{array}\right]$$

All three packets $M_1$, $M_2$ and $M_3$ are decoded.

## III. Routing State Information Ageing

In conventional networks, routing is where a decision is made on which output line to forward a packet when it arrives. Nodes (routers) in such networks need to know the whole or part of the network map in order to achieve efficient routing. In contrast, with basic network coding scenario (and basic flooding), nodes do not need to know the network map, and just forwards all the packets it receive to all the output lines.

In some scenario, flooding can be considered very inefficient, when it is clear-cut known that sending a packet to some lines will never reach a particular destination. If this information is known to the node, then opting for a selective flooding will be more efficient in terms of bandwidth usage. Selective flooding is where an incoming packet is forwarded to some (most) of the output lines, with this decision being based on some high-level information. This means that for such kind of scenario, some information about the network needs to be known and stored at the node, adding up complexity to this routing technique. In case this information changes with time during packet routing process, a dynamic version of selective flooding is attained.

For network coding, if a node knows that one of his neighbors already has a packet or that this neighbor will not help to make a packet reach its final destination, then there is no need to include this packet in the encoded packet to be sent to that node. This may reduce bandwidth utilization and may considerably reduce the decoding effort of that neighbor.

Routing State Information Ageing (RSIA) is a technique whereby nodes have some routing information in memory that decays with time. It mimics the forgetting process in human psychology. RSIA works by assigning a knowledge value for each possible path flow at the node, and routing happens based on these knowledge values. The knowledge values vary during the lifetime of the routing process based on:

1) information/knowledge gained through reception of new messages
2) information/knowledge decay.

Consider the following network where Node A needs to send a packet to Node E.

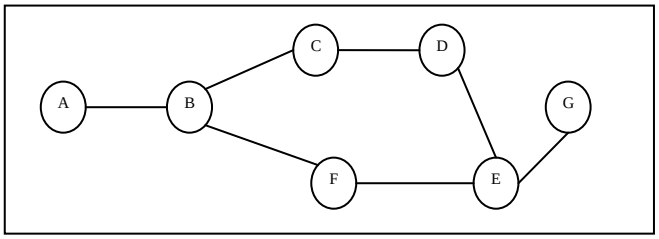


Figure 4: Network Example 2

Through flooding, A will first send the packet to Node B.

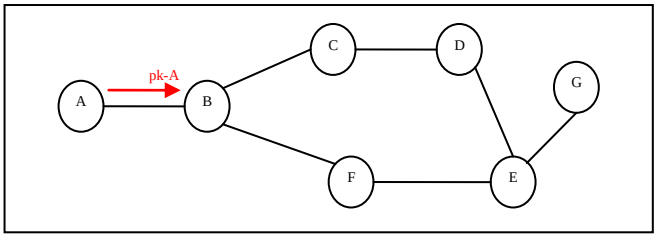


Figure 5: Network Example 2 after flow iteration 1

At B, the packet will be flooded to C and F.

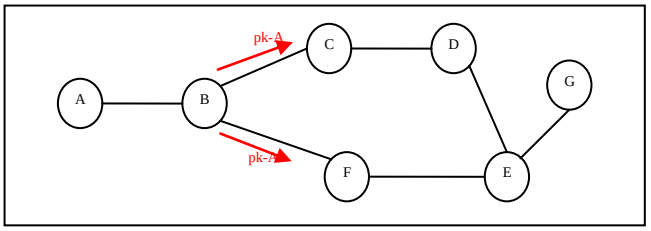


Figure 6: Network Example 2 after flow iteration 2

In next step, the packet will flow to D and E.

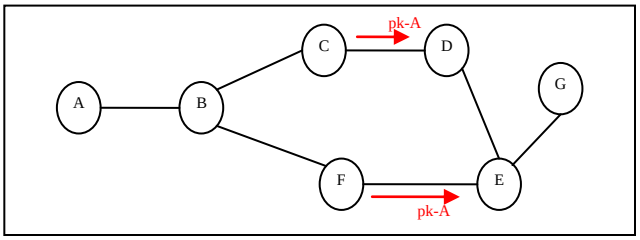


Figure 7: Network Example 2 after flow iteration 3

At this stage:

1. pk-A with source data A has searched its destination E (data).
2. this packet came from incoming line F (information).
3. to send a packet from E to A, sending the packet to F is advantageous rather than sending the data to D (knowledge).

In the next iteration, E will receive a second copy of pk-A, but will discard it. Now consider that G needs to send a packet pk-G to A. After the packet reaches E, by normal flooding, it will be flooded to both F and D.

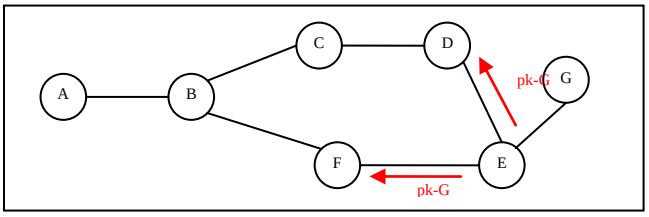


Figure 8: Network Example 2 after reverse flow iteration 1

However since E already has the knowledge about a quick path to A via F, it is better to send the packet to F only. This will control the unnecessary replication of packets during flooding.

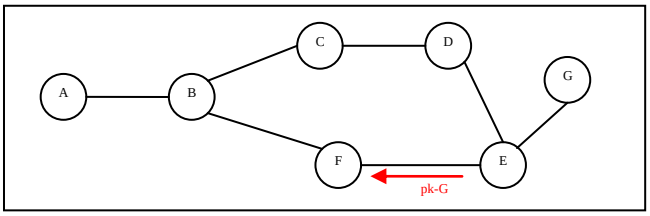


Figure 9: Network Example 2 after reverse flow iteration 1 with Improvement

In most cases, this optimized flooding resembles static shortest path in practice, and likewise it can happen that, with time, network traffic increases in the selected path, and by this, sending the packet through D could have been a better choice.

We can also argue that if another packet pk-A2 with source A comes from D to E, then we can replace our best path in E for A via D. But having the second packet pk-G always sent towards F for A, we can never have a new packet pk-A2 at B sent for E towards C. It means that our selective option for flooding will tend towards the same static decisions (like shortest path) with time.

Logically, the probability of F as the best route for pk-G at node E is very high when pk-A reached E, but this probability tends to decrease with time, as things change over time. We can achieve this principle by introducing knowledge decay (forgetting) at node E. This is governed by a knowledge decay rate kdr (e.g., 0.09), a knowledge insignificant value kiv (e.g., 0.1) and a knowledge significant value ksv (e.g., 0.6).

The knowledge decay rate is the amount of knowledge (for each unit) that is lost or forgotten per unit time. For example, at time t, we have a knowledge value of 2.0 and during the subsequent periods of time, we do not have any knowledge gain concerning the measured property truth. At the subsequent time t+1, considering a kdr of 0.09, the knowledge value will become 2.0 - (2.0 x 0.09) = 1.82. At time t+2, the knowledge value will decrease to 1.82 – (1.82 * 0.09) = 1.6562.

The knowledge insignificant value is defined as the minimum knowledge value that is deemed important enough to be considered different as zero value (may however be not enough for positive decision making). It also means that the system does not need to store a knowledge value which is equal or below the kiv, but can replace this value with zero.

The knowledge significant value is the minimum knowledge value that allows a positive decision making to be taken. In other words, equal or above this threshold, the solution being measured can be considered to be a valid solution. The value may be computed as a factor ksf (Knowledge Significant Factor) of the maximum Knowledge value for a particular node and destination pair. This means that in a well dispersed configuration, the ksf represents the ratio of links to be chosen among possible links for a particular flow.

$(K\text{-}Value(t\text{-}1) > kiv) \rightarrow K\text{-}Valuet = K\text{-}Value(t\text{-}1) * (1 - kdr)$
$\wedge \neg(K\text{-}Value(t\text{-}1) > kiv) \rightarrow K\text{-}Valuet = 0$

The diagram below shows an example of the Knowledge value for a particular solution starting at value 2.0 and another one at 1.5 with kdr at 0.09, kiv at 0.1, ksf at 0.6 (assuming that the first knowledge value is also the maximum for the destination) and with no further positive influence.

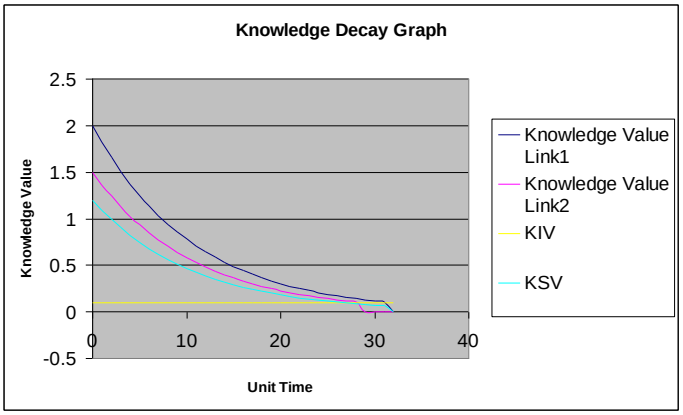


Figure 10: KDR, KIV and KSV Influence on Knowledge Value

In the above graph, it can be seen that any incoming packet initially will be forwarded to both Link 1 and Link 2. When Knowledge value for Link2 drops to KIV and subsequently to zero, any incoming packet will be forwarded to only Link 1. However when the knowledge value for Link 1 drops to KIV and subsequently to zero, any incoming packet will be forwarded to both Link 1 and Link 2.

Consider that we attribute a knowledge value of 1 when knowledge is gained (a new packet is obtained) at Node E.

Table 3: Knowledge Value Table Example at iteration 0

| *Destination* | *Selected Next* | *Knowledge Value* |
|---|---|---|
| A | D | 0 |
| A | F | 0 |
| A | G | 0 |
| B | D | 0 |
| B | F | 0 |
| B | G | 0 |
| C | D | 0 |
| C | F | 0 |
| C | G | 0 |

When pk-A is obtained via F, the table is updated as follows (Table 4).

Table 4: Knowledge Value Table Example at iteration 1

| *Destination* | *Selected Next* | *Knowledge Value* |
|---|---|---|
| A | D | 0 |
| A | F | 1 |
| A | G | 0 |
| B | D | 0 |
| B | F | 0 |
| B | G | 0 |
| C | D | 0 |
| C | F | 0 |
| C | G | 0 |

For each iteration, the knowledge value for each row is decreased by a decay rate of kdr = 0.09. After two iterations, the table will be updated as (considering no other packet has been received) (Table 5).

An incoming packet is then forwarded to all selected path for that particular destination for which

$$\text{K-Value} \geqslant \max(\text{K-Value}) * \text{ksf}$$

Properties of Flooding/Network Coding with Ageing:

Table 5: Knowledge Value Table Example at iteration 2

| *Destination* | *Selected Next* | *Knowledge Value* |
|---|---|---|
| A | D | 0 |
| A | F | 0.8281 |
| A | G | 0 |
| B | D | 0 |
| B | F | 0 |
| B | G | 0 |
| C | D | 0 |
| C | F | 0 |
| C | G | 0 |

- Node does not need to know the structure of the network as compared to Shortest Path.
- The best path is selected from statistics of real packets; no predefined path computation is required, as compared to Shortest Path.
- It has the capability of load balancing and multiple path selections.
- No control packet needs to be circulated in the network, as compared to existing dynamic routing algorithms.
- Each node requires enough memory to store the knowledge table.

## IV. Experimental Set-up

### *A. Network Simulation Application*

A Network Simulation Application (NSA) has been built that allows any routing technique to be plugged in for evaluation. The NSA is composed of N number of similar structured networks (same nodes and links) and each of the networks is composed of a number of nodes n and bi-directed links l, and where each node has at least one link to another node making the network always connected.

Packets (bunch of data, also referred as messages) traverse the network from node to node via links. At a unit time (referred to as one hop), only one packet can travel a link. Other packets that need the currently occupied link have to wait for the next hop. Furthermore, if more than one packet reaches a node, the node can process only one packet at a time, therefore making the other incoming packets to wait for their turn to get processed. These waiting process have been implemented using queues because of the First Come First Serve characteristics required.

Each node has three queues associated with it:

1) *Incoming queue*

   This queue contains the messages received by the node that are waiting to be processed. Messages may be received by a node from external nodes (via links through routing) or by higher protocols within the node (message creation).

2) *Outgoing queue*

This queue contains messages that are waiting to go through the links to other nodes. For each link associated to a node, we have one outgoing queue. Therefore one node has the same number of outgoing queue as links connected to it.

3) *Processing queue*

This queue contains messages that have reached their ultimate destination. Messages put in this queue are meant to be read by higher protocols, and therefore no further processing happens at the NSA for the packets in this queue.

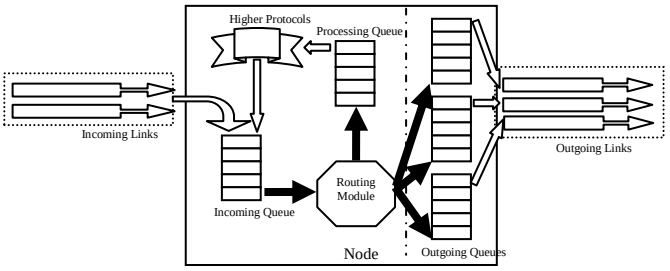


Figure 11: Network Simulation Application Architecture

During each hop, the following processes happen at each node:

1) The routing module consumes one packet from incoming queue (may read more than 1 packet for network coding)
2) The routing module decides what to do with the current packet: sending to one or more outgoing queues or processing queue or discard the packet.
3) The first packet in the outgoing queue of a link is transferred to the incoming queue at the other end of the link (to another node except for self-directed links). This process is called Hopping.

Each message packet has enough information for routing modules within nodes to decide how to route the packet till it reaches its ultimate destination. The data that constitute a message packet is:

1) Message Id: a unique number in each network that represents a particular message. Two packets in the same network can however have the same message id if they are replicates (e.g. for flooding).
2) Message Data: the information content of the packet. This data is important to higher level protocols.
3) Source: the node id where the packet was introduced in the Network. This is useful for determining the next paths for selective flooding.
4) Destination: the node id where the packets needs to be routed.
5) Time to Live (TTL): the amount of hop remaining after which the packet needs to be destroyed. This is useful for flooding to ensure that packets do not over-propagate.
6) Last Hop: indicates the last node the packet visited. This is useful for flooding and network coding in reducing unnecessary network consumption.
7) Merged: indicates a flag to determine whether a packet is an original packet or a merged packet in network coding routing.

The Network Simulator Application has been built with the following functions:

1) Random creation of N number of similar structured networks with n number of nodes and $l_{avg}$ average link per node ($l_{avg}$ is only used as an indicative parameter).
2) Perform preliminary computations for different routing algorithms (E.g. Building Shortest Path tree).
3) Injecting M number of messages/packets at random (source, destination and data length) in all networks (same packets injected in all networks for better comparison).
4) Perform routing for all networks with each network associated with one routing module, until all packets reach their respective destination.
5) Provide measurement of network generated like complexity, average max path length, etc., and results data like number of hops count required to complete transfer for each network.

### *B. Shortest Path Algorithm*

The shortest path for each node to other nodes have been computed by having two arrays hops and paths with size the same as the number of nodes (each row representing one node). The hops array (all elements) is initialized with the number of nodes, and the paths array is set as empty strings. Next, the hops record for the source node is set to zero. Starting from the first hop record to the last one, and if the hops value is less than the number of nodes, the hops value of the neighbors of the node is set to one more than the current hop record if this value is less than the existing one. When a hop records value is changed, the path of the changing index is also updated by appending the path of the hop under consideration with the same hop. This process is repeated until no further change happens.

### *C. Flooding Algorithm*

The flooding module has been implemented by taking care of circulation problem that can arise with there is loop in the network. Unnecessary redundant packets have been removed from the network by the following mechanisms:

1) Time to Live (TTL) of packets
   An original packet when inserted in the network has a maximum hop count value known as TTL value. At each node where the packet is forwarded, the TTL is decreased by 1. When the TTL for a packet reaches 0, it is discarded.
2) Known packets
   When a packet reaches a message for the first time, its reference is stored in a list called history at the node. Each packet that needs to be processed is first checked within the history (which contains the message id), and in case present, is discarded. This also ensures that no packet is delivered twice to the higher protocol at the receiving node.

### *D. Network Coding*

Network coding has been implemented by combining two packets (for simplicity of implementation) using XOR operation on bytes. The network coding routing module peeks (without removing from the queue) into the incoming queue for the first two packets. It then checks whether it is possible to encode (merge) these two packets. Two packets are merge-able when all of the below conditions are met:

1) None of the two packets are already merged.
2) None of the two packets have reached their destination.
3) None of the two packets are already known packets to the node (present in history).
4) The size of both packets (data part) are less than or equal to 1944 (maximum packet size of 2048 – 2 times the added header information of size 52).
5) The merged packet does not already exist in the node history.

If merging is not possible, the first packet is read from the input buffer and is processed exactly as flooding. In case merging is possible, the merged packet is sent over the network similarly like flooding but only the first packet is removed from the incoming buffer.

Merging is done on each byte as per the location of the bytes in the data streams. In other words, byte at position 1 in data part of packet 1 is XORed with byte at position 1 in data part of packet 2. The header information for the original packets is inserted in the data part of the merged packet to allow the latter to be undistinguished from other packets during hopping process.

In the Merged packet, only the minimum length is kept, since it is enough to deduce both packets length given the following information is already known during the decoding process:

1) Length of merged packets (length of the bigger packet).
2) Minimum length stored in the merged packets (length of the smaller packet).
3) Length of the known packet (at least one of the packet needs to be known to allow decoding).

Having the decoded packet length together with the merged data and the know packet data, it is easy to decode the new packet.

### *E. Routing State Information Ageing*

Decision on which links to flood a packet is based on some knowledge values for each possible flow. In simpler terms, each node is required to store a knowledge value (truth value) for each destination link pairs in a table called knowledge table. For example, in a network of n nodes, and one particular node has l links, the number of rows in the knowledge table is (n-1) x l.

One new table has been introduced in the NSA with three predominant fields denoting the destination, the link to use and a knowledge value. When the network is built, the knowledge table is filled for each node with possible destinations and the node's available links. The knowledge value is set to zero since at this time there is no preference for particular paths to reach a particular destination. We implemented RSIA with kdr of 0.09, kiv at 0.1, ksf at 0.6.

### *F. Metrics*

For the purpose of comparing the routing strategies, the number of iterations (unit time) or hops required for all injected packets to reach their destinations and the thickness lower bound of the graph have been considered.

The thickness of a graph is the minimum number of planar sub-graphs needed such that a union of these sub-graphs makes the original graph. More number of edges with the same number of hosts may imply a thicker graph. Measuring thickness is an N-P Hard problem and a good estimate for it is the thickness lower bound [6].

## V. Simulation Results and Discussions

### *A. Network Coding Routing Comparison Experiment*

The shortest path, flooding and network coding algorithms have been implemented in the Network Simulation Application, and have been executed with number of nodes n ranging from 10 to 50, $l_{avg}$ ranging from 2 to 6 and number of messages injected M = 50. The experiment has been done 150 times and the network metrics generated are as follows (Table 6):

Table 6: Routing Algorithm Comparison Experiment – Node Frequency Distribution

| *Nodes* | *Frequency* | *Percent* | *Cumulative Percent* |
|---|---|---|---|
| 10 | 25 | 20 | 20 |
| 20 | 25 | 20 | 40 |
| 30 | 25 | 20 | 60 |
| 40 | 25 | 20 | 80 |
| 50 | 25 | 20 | 100 |

Table 7: Routing Algorithm Comparison Experiment – Thickness Lower Bound Frequency Distribution

| *Thickness Lower Bound* | *Frequency* | *Percent* | *Cumulative Percent* |
|---|---|---|---|
| 1.3 | 2 | 1.6 | 1.6 |
| 1.4 | 2 | 1.6 | 3.2 |
| 1.5 | 3 | 2.4 | 5.6 |
| 1.6 | 3 | 2.4 | 8 |
| 1.9 | 10 | 8 | 16 |
| 2 | 62 | 49.6 | 65.6 |
| 2.1 | 33 | 26.4 | 92 |
| 2.2 | 6 | 4.8 | 96.8 |
| 2.3 | 4 | 3.2 | 100 |

The graph below shows the general trend for each experiment performed.

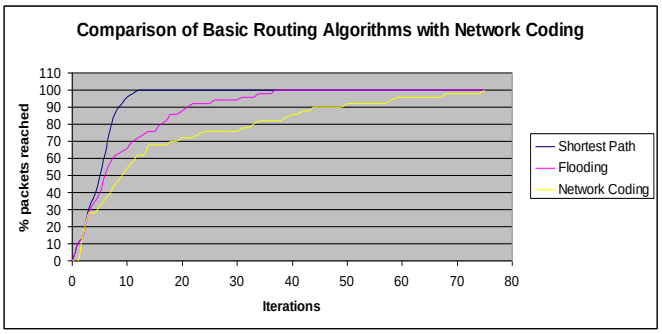


Figure 12: Basic Network Coding Iterations Comparison

In the initial stage of the experiment, Network Coding tends to send merged packets. Since no node has received any original packet yet, they need to wait for at least one original packet to be received to generate two packets. In comparisons, in the initial iterations, both Shortest Path and flooding are already sending real packets, making their start performance better than network coding.

In the next rounds, the original packets start to reach their destination which means that when one original packet is received at a node already having a merged packet of the original packet, another packet can be inferred. This explains why Network Coding equals the performance of both Shortest Path and Flooding after few iterations.

However, after consequent rounds, both Flooding and Network Coding tends to send unnecessary packets (original or merged packets) all round the network, allowing Shortest Path to demark from them. Also, when comparing Network Coding with Flooding, the number of packets (and proportionally unnecessary packets) to distribute throughout the network increases for network coding. Thus, flooding performs better and completes around half number of iterations than network coding does.

Table 8: Routing Algorithm Comparison Experiment – Network Metrics Correlations with Node

| *Node* | | *Mean* | *Standard Deviation* | *Coefficient of Variation* |
|---|---|---|---|---|
| Pearson | Correlation Coefficient | 0.026 | 0.163 | .370** |
| | Sig. (2-tailed) | 0.778 | 0.069 | 0 |
| | N | 125 | 125 | 125 |
| Kendall's tau_b | Correlation Coefficient | -0.03 | 0.082 | .256** |
| | Sig. (2-tailed) | 0.647 | 0.213 | 0 |
| | N | 125 | 125 | 125 |
| Spearman's rho | Correlation Coefficient | -0.037 | 0.109 | .339** |
| | Sig. (2-tailed) | 0.682 | 0.227 | 0 |
| | N | 125 | 125 | 125 |

**. Correlation is significant at the 0.01 level (2-tailed).

For each experimental result, the Mean, Standard Deviation and Coefficient of Variation of the number of iterations required for shortest path, flooding and network coding routing have been computed. These statistical metrics have been compared with Node and Thickness Lower Bound.

The Table 8 shows the different correlation coefficients of Node with Mean, Standard Deviation and Coefficient of Variation.

The correlations above show that network size does not impact the mean and standard deviation of iterations required for completion by the three routing methods. However, all three correlation methods show significant correlation values (above 0.01) for coefficient of variation.

To further illustrate this relationship, the average for variation has been computed for each node value as shown in the graph below (Fig. 13).

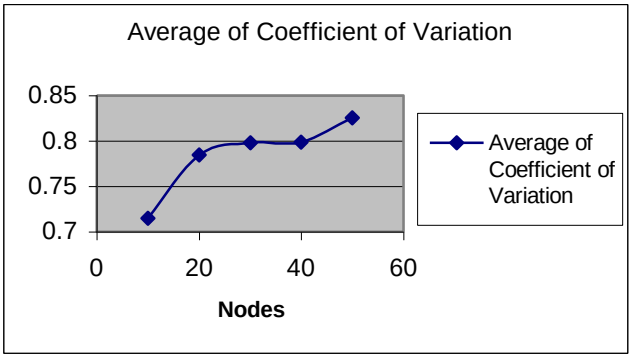


Figure 13: Basic Network Coding - Node against Coefficient of Variation

This implies that, as the network size increases, distances between the numbers of iterations required increases between shortest path, flooding and network coding.

The table below shows the different correlation coefficients of Thickness Lower Bound with Mean, Standard Deviation and Coefficient of Variation.

Table 9: Routing Algorithm Comparison Experiment – Network Metrics Correlations with Thickness Lower Bound

| *Thickness Lower Bound* | | *Mean* | *Standard Deviation* | *Coefficient of Variation* |
|---|---|---|---|---|
| Pearson | Correlation Coefficient | -0.171 | -0.11 | -0.028 |
| | Sig. (2-tailed) | 0.057 | 0.222 | 0.756 |
| | N | 125 | 125 | 125 |
| Kendall's tau_b | Correlation Coefficient | -.137* | -.137* | -0.081 |
| | Sig. (2-tailed) | 0.045 | 0.044 | 0.235 |
| | N | 125 | 125 | 125 |
| Spearman's rho | Correlation Coefficient | -.185* | -.178* | -0.104 |
| | Sig. (2-tailed) | 0.039 | 0.047 | 0.25 |
| | N | 125 | 125 | 125 |

*. Correlation is significant at the 0.05 level (2-tailed).
**. Correlation is significant at the 0.01 level (2-tailed).

The above table shows little linear correlation between thickness lower bound and mean, standard deviation or coefficient variation (Pearson coefficient). However Spearman's rho correlation coefficient shows a negative relationship between thickness and mean or standard deviation, but no significant relationship for coefficient of variation. This is further illustrated by plotting thickness against average mean and average standard deviation.

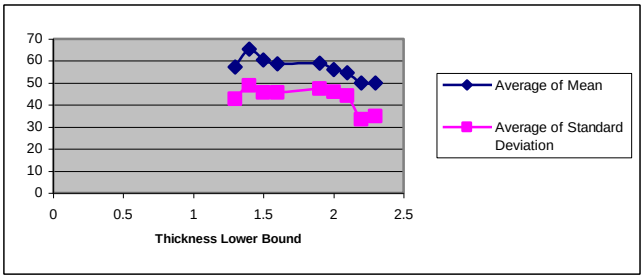


Figure 14: Basic Network Coding - Thickness Lower Bound against Mean and Standard of Deviation

The above graphs show slight decreases in standard deviation and mean as thickness increases. This can be explained as the denser a network is, more redundant path exists, and better is flooding and network coding performances. Since there is no consequent change in coefficient of variation, an improvement to Shortest Path can be deduced when thickness increases. The reason for this is that fewer nodes are bottlenecks, and therefore packets flow with minimum delay.

Shortest Path performs better in general as compared to Flooding and Network Coding because it does not pollute the network and its nodes possessing the advantage of having the complete network picture. Figure 12 illustrate this.

Since Shortest Path is considered as a benchmark, for similar network experiments for other routing techniques, the minimum deviation of the number of iterations for other routing techniques together with Shortest Path is preferable. Figure 13 shows that the lesser the number of node, the less deviation exists. Furthermore, figure 14 shows that after a threshold of thickness lower bound of 2, the deviations are less.

Therefore, we can conclude that Network Coding performs better when the network size is small (less than 10 nodes) and the network is dense (thickness lower bound is greater than 2, making the average number of edges per node 4.8 as minimum).

### *B. Routing State Information Ageing Experiment*

Both flooding and network coding have been implemented with RSIA (kdr of 0.09, kiv at 0.1, ksf at 0.6) in the Network Simulation Framework, and have been executed together with the original flooding and network coding algorithms with number of nodes n ranging from 10 to 40, lavg ranging from 2 to 6 and number of messages injected M = 50. The experiment has been done 100 times and the input metrics generated are as follows (Table 10):

Table 10: RSIA Experiment – Node Frequency Distribution

| *Nodes* | *Frequency* | *Percent* | *Cumulative Percent* |
|---|---|---|---|
| 10 | 25 | 25 | 25 |
| 20 | 25 | 25 | 50 |
| 30 | 25 | 25 | 75 |
| 40 | 25 | 25 | 100 |

Table 11: RSIA Experiment – Thickness Lower Bound Frequency Distribution

| *Thickness Lower Bound* | *Frequency* | *Percent* | *Cumulative Percent* |
|---|---|---|---|
| 1.8 | 1 | 1 | 1 |
| 1.9 | 6 | 6 | 7 |
| 2 | 49 | 49 | 56 |
| 2.1 | 32 | 32 | 88 |
| 2.2 | 6 | 6 | 94 |
| 2.3 | 3 | 3 | 97 |
| 2.4 | 3 | 3 | 100 |

For each of the experiment, the reduction rate for both flooding and network coding has been computed as the improvement (improved version iteration – normal version iteration) to normal version ratio. This value gives a statistical comparison performance measure for RSIA improvement.

The Table 12 shows correlations coefficients for flooding improvement reduction rate and network coding reduction rate as compared to node.

Table 12: RSIA Experiment – Improvement Reduction Rate – Node Correlation

| *Node* | | *Flooding Improvement Reduction Rate* | *Network Coding Improvement Reduction Rate* |
|---|---|---|---|
| Pearson | Correlation Coefficient | 0.022 | -0.068 |
| | Sig. (2-tailed) | 0.828 | 0.503 |
| | N | 100 | 100 |
| Kendall's tau_b | Correlation Coefficient | 0.082 | -0.018 |
| | Sig. (2-tailed) | 0.282 | 0.831 |
| | N | 100 | 100 |
| Spearman's rho | Correlation Coefficient | 0.124 | -0.021 |
| | Sig. (2-tailed) | 0.217 | 0.835 |
| | N | 100 | 100 |

*. Correlation is significant at the 0.05 level (2-tailed).
**. Correlation is significant at the 0.01 level (2-tailed).

Pearson correlation coefficient shows a minor linear negative relationship between Network Coding Improvement Reduction Rate and Node, whereas Spearman's rho coefficient

shows some positive relationship between Flooding Improvement Reduction Rate and Ratio.

The graph below shows the rate changes with node (Fig. 15).

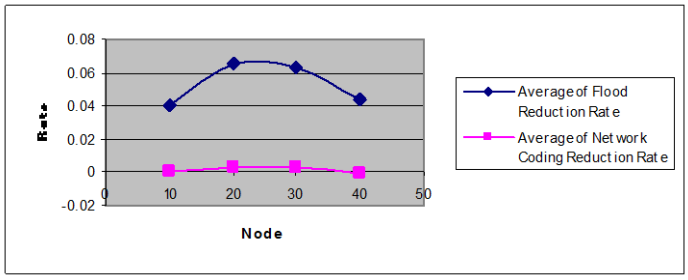


Figure 15: RSIA Experiment – Reduction Rate Percentage against Nodes

The figure above shows a considerable reduction rate for flooding with medium network size (between 20 and 30). However, the rate improvement for network coding is always less than 0.01, and barely changes with network size, thus disqualifying RSIA as a good optimization for network coding routing.

Thickness has also been compared to reduction rate as shown in the coefficient Table 13.

Table 13: RSIA Experiment – Improvement Reduction Rate – Thickness Lower Bound Correlation

| *Thickness Lower Bound* | | *Flooding Improvement Reduction Rate* | *Network Coding Improvement Reduction Rate* |
|---|---|---|---|
| Pearson | Correlation Coefficient | -0.076 | -0.066 |
| | Sig. (2-tailed) | 0.454 | 0.512 |
| | N | 100 | 100 |
| Kendall's tau_b | Correlation Coefficient | -0.118 | -0.125 |
| | Sig. (2-tailed) | 0.13 | 0.148 |
| | N | 100 | 100 |
| Spearman's rho | Correlation Coefficient | -0.137 | -0.147 |
| | Sig. (2-tailed) | 0.175 | 0.145 |
| | N | 100 | 100 |

*. Correlation is significant at the 0.05 level (2-tailed).
**. Correlation is significant at the 0.01 level (2-tailed).

The coefficients in table above show very little negative significant relationship between thickness lower bound and improvement reduction rates. The average improvement reduction rate curves in the figure below shows this relationship.

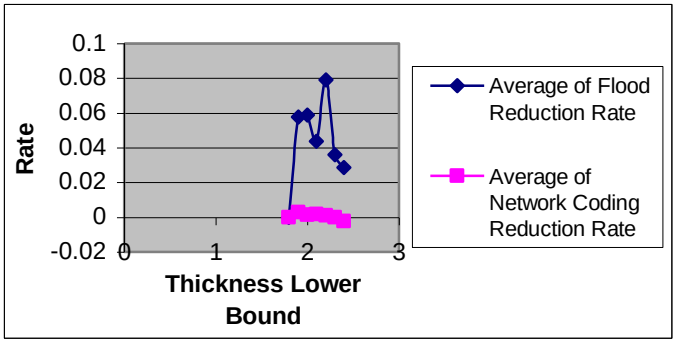


Figure 16: RSIA Experiment – Improvement Reduction Rate against Thickness Lower Bound

When thickness lower level is minimal, both the average reduction rate for flooding and network coding are very low. For flooding, this however increases as the graph gets denser, whereas for network coding, there is no significant change. When the thickness lower bound goes above a threshold, flooding improvement rate starts to decrease. The same behavior is for network coding, except that network coding goes to a negative rate implying that with dense networks, network coding without RSIA performs better than network coding with RSIA.

Network Coding uses flooding technique inherently during its routing process. Figure 14 shows that RSIA can increase flooding performance to 6 to 7 percent for medium size network (20 to 30 nodes) and figure 15 shows that flooding with RSIA is more effective when the thickness lower bound is between the medium ranges (2 to 2.25). It is also seen that the improvement brought by RSIA to network coding is very minimal, and inadequate for high consideration.

## VI. Conclusion and Future Works

The research revealed that in a noiseless environment, despite the fact that Network Coding does not need to know the complete network structure and therefore no setup required, its performance comes closer to Shortest Path when the network size is small and network density is high. This makes network coding a good candidate for small LAN setups with multiple links to nodes, achievable through fixed wireless arrangements. This can be further investigated by performing in-depth experiments on such networks for Network Coding.

The results show that Flooding can be sufficiently improved by introducing RSIA, producing a dynamic selective version of flooding. This works better in a medium range for both network size and network thickness for Flooding. Network Coding, however, is less impacted by RSIA since Network Coding required some redundancy for it to be effective and RSIA tries to decrease redundancy in networks.

For this study, RSIA has been modeled using Knowledge Decay Rate 0.09, Knowledge Insignificant Value 0.1 and Knowledge Significant Factor 0.6. Further research can be done by varying the RSIA parameters to investigate the correct parameters to achieve larger medium ranges for network size and network density for both selective flooding and selective network coding.

## REFERENCES


[1] R. Ahlswede, N. Cai, S.Y.R. LI, R.W. Yeung, "Network information Flow", IEEE Transactions on Information Theory, 46, 1204-1216, 2000.

[2] Z. Li, B. Li, "Network Coding in Undirected Networks", In: ed. Proceedings of the 38th Annual Conference on Information Science and Systems (CISS), Princeton, NJ, USA, 257-262, March 2004.

[3] P. Chou, Y. Wu, K. Jain, "Practical network coding", In Allerton Conference on Communication, Control, and Computing, Monticello, IL, September 2003.

[4] S.Y.R Li, R.W. Yeung, N. Cai, "Linear Network Coding". IEEE Transactions on Information Theory, 49 (2), 371-381, 2003.

[5] C. Fragouli, J. Le Boudec, J. Widmer, "Network Coding: An Instant Primer". ACM SIGCOMM Computer Communication Review, 36(1), 63-68, 2006.

[6] S. Skiena, "Implementing Discrete Mathematics Combinatorics and Graph Theory with Mathematica". Reading MA: Addison-Wesley, 1990.


**Naushad Muhammad Peyrye** (naushad_peyrye@infosys.com) received his BSc (Hons) degree in Computer Science and Engineering with First Class Honors from University of Mauritius in 2005. From 2006 onwards, he is with Infosys Limited, initially as Software Engineer, Technology Analyst, and currently as Technology Lead. During this time, he worked with large international corporations in Netherlands and France and had the opportunity to appreciate the setup, practical workings and challenges of large network implementations

**Mussawir Ahmad HOSANY** (m.hosany@uom.ac.mu) received his B.Eng (Hons) degree in Electrical and Electronic Engineering with First Class Honors from University of Mauritius in 1997, MSc degree in Electronics and Communications Engineering with Distinction from University of Ulster (Belfast) in 2001 and PhD from University of Mauritius in Communication Theory in 2004. In 2009, he was awarded the USA Fulbright postdoctoral fellowship and preceded to California. He worked on a 9-month research project at the San Diego State University and successfully designed and implemented a cross-layer approach for H.264/AVC over fading channels. He is presently senior lecturer with the department of Electrical and Electronic Engineering, Faculty of Engineering of the University of Mauritius.