IDENTIFYING OPPORTUNITIES FOR DETERMINISTIC NETWORK CODING IN WIRELESS MESH NETWORKS

M.J. Grobler* **and A.S.J. Helberg***

* *School for Electrical, Electronic and Computer Engineering, North-West University, Private Bag X6001, Potchefstroom South Africa, E-mail: leenta.grobler@nwu.ac.za, albert.helberg@nwu.ac.za*

Abstract: Recent advances in methods to improve network utilisation has lead to the introduction of Network Coding, a technique that can reduce local congestion in a network by combining information sent over the network. In this paper the use of deterministic Network Coding in a Wireless Mesh Network (WMN) is proposed. A method to determine where Network Coding can be implemented in a WMN is presented. It is shown that the inherent properties of WMNs provide good opportunities for the implementation of this method

Key words: Capacity, Network Coding, Network Topology, Network Utilisation, Quality of Service, Wireless Mesh Networks.

1. INTRODUCTION

The universal need for better control over resources in communication networks is a problem that is studied continuously. The network's maximum capacity needs to be defined and then utilised to ensure that as much information as possible is delivered in the most beneficial manner. One way in which this can be achieved, is by implementing a relatively new method called Network Coding. Research on Network Coding to date has lead to a wide variety of theoretical results, mainly applicable to wired networks, where Network Coding is implemented deterministically.

The capacity and resource control problem in wireless networks also needs to be addressed, but the identification of Network Coding opportunities is a bit more complicated, especially in the case of Wireless Mesh Networks (WMNs). WMNs are complex networks of which the topology may change unpredictably due to nodes joining and leaving the network, or moving around (in the case of a mobile ad-hoc network). One documented implementation of Network Coding in wireless networks is Random Network Coding [1]. Deterministic Network Coding however holds a few advantages over Random Network Coding, which will be discussed later in this paper and therefore it would be advantageous to find a way to implement Deterministic Network Coding in wireless networks.

In this paper, a method for the identification of opportunities for the implementation of deterministic Network Coding is presented. Further it is shown that the inherent properties of WMNs provide good opportunities for the implementation of this method.

The remainder of this paper is structured as follows: An introduction to WMNs is given in section 2. The maximum throughput capacity of a network in terms of the Min-cut Max-flow theorem is briefly discussed in section 3. Network Coding and the advantages it holds are presented in section 4. A new method to identify opportunities for the implementation of deterministic Network Coding in WMNs is presented in section 5. This method is evaluated in section 6. The impact on throughput capacity of WMNs is briefly discussed in section7. Finally, a conclusion is drawn in section 8.

2. WIRELESS MESH NETWORKS

WMNs are wireless ad-hoc networks, consisting of radio nodes that form a mesh topology [2]. These nodes can be either mobile or stationary and generally accommodate multiple hops. A WMN may operate standalone or be connected to other networks (like the Internet or a wireless sensor network). The nodes in WMNs can be either routers or clients and they may be self configuring. The management of WMNs can be either centralised or decentralised. The advantages and disadvantages of Wireless Mesh Networks can be summarised [2],[3] as follows:

Advantages:

- Multiple hops increase the area that the network can cover and therefore No Line Of Sight (NLOS) communication is possible.
- The mesh topology increases the robustness of the network against link failures, because there exist multiple paths to a given node.

Disadvantages:

- Multiple hops increases delay in the delivery of information.
- Routing protocols associated with WMNs tend to be either very complex or overly simple.

3. MIN-CUT MAX-FLOW THEOREM

The Min-Cut Max-Flow theorem states: *The maximum flow of information in a network is equal to the sum of the cut of the link capacities* [4].

Using the theorem, the network in figure 1 is cut, separating the sender node S and receiver nodes X and Y to cross as few of the links as possible to determine the minimum cut.

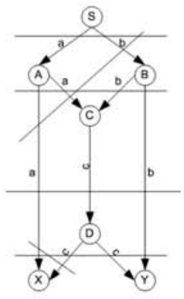

Figure 1: Min-Cut Max-Flow theorem

When each of the links have unit capacity it can be seen that the minimum cut and therefore the maximum flow (or maximum throughput capacity) of this network is equal to 2. The only way that this maximum flow can be achieved, is through the implementation of Network Coding [5],[6].

4. NETWORK CODING

Network Coding is a concept that was first introduced in [6] as a method to utilise the maximum capacity of a network and maximise the flow of information in that network. It suggested coding at packet level in wired peer-to-peer networks.

4.1 The Concept of Network Coding

The Butterfly network from [7], as depicted in figure 2, is used to explain the concept of Network Coding. The links in the figure all have unit capacity and messages *a* and *b* are binary.

Two nodes, A and B both need to transmit their messages to Nodes X and Y. Each of the nodes can forward their own message to the node that is directly connected to it, but have to route their messages through the network to reach the second node. Using traditional routing (figure 2a), node C simply replicates the information it receives from the previous sender node. In this case the two messages *a* and *b* will reach node C simultaneously. Node C will send out message *a* and then after it has been sent, message *b*. Thus, at the end of a single round of transmissions, only node Y will have received both messages, while node X still has only message *a*. This results in a throughput of 1.5 (three of the original two messages were delivered.)

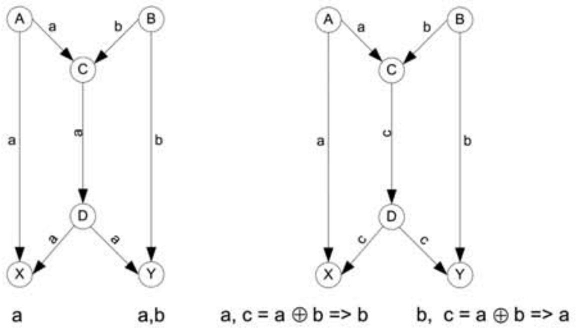

Figure 2: Butterfly Network a) without Network Coding b) With Network Coding

When Network Coding is implemented (figure 2b), node C will have the capability to transmit a linear combination (logical XOR) of the binary messages *a* and *b*. Message *c* is the same length as message *a* and *b* and is transmitted via node D to nodes X and Y. Nodes X and Y then have the capability to decode message *c* by using the message already received from the directly connected source node to solve the two linearly independent equations. In this special case, it merely means another logical XOR of the message that has already been received and the encoded message *c*. In this instance, at the end of a single round of transmissions, nodes X and Y received both messages. Four messages were delivered to two nodes in the communication round, giving a throughput of 2 [6]. This correlates to the result obtained with the Min-Cut Max-Flow theorem for a butterfly network. In effect an extra "virtual"link is created in the network, which improves the capacity of the network.

This method however changes the way node C works, because it has to form linear combinations of the messages it receives before forwarding the linear combination of these messages as a single message. It also requires nodes X and Y to have knowledge of the network topology and how the messages that reach it were encoded in order to deduce the two original messages from the messages received.

4.2 Benefits of Network Coding

The use of Network Coding in a network may provide the following benefits:

1. Throughput [6], [7], [9]: The improved throughput in networks was the first major result of Network Coding. If we refer to the throughput achieved with Network Coding in the deterministic example above, it is clear that the maximum throughput as calculated using the Min-cut Max-flow theorem has been achieved.

2. *Robustness [7], [9], [10]:* The robustness of the network refers to the ability of the network to remain

functioning even though a link has failed completely.

3. Adaptability [9], [11]: Adaptability is an important benefit when looking at WMNs, as this refers to the ability of the network to cope with nodes constantly joining and leaving the network, resulting in a constantly changing topology.

4. Security [9]: The security benefit is an inherent benefit, seeing that linear combinations of data are sent over the network and not the actual data. This benefit while useful, is however not sufficient. If a malicious entity listens long enough and receives sufficient messages to decode the information, the information can still be eavesdropped.

One approach to make Network Coding suitable for wireless networks is Random Network Coding [1]. Random Network Coding allows the nodes in a network to randomly form linear combinations of the data received. These linear combinations are then forwarded through the network (with the information of how they were combined stored in a header field) until it reaches the receiver node. The receiver node then has to wait for enough linear independent combinations to reach it before it can decode the original messages. Deterministic Network Coding however holds a few advantages over Random Network Coding:

- Less decoding delay - Decoding nodes have a predetermined number of messages to wait for and don't have the uncertainty of waiting until enough linear independant messages are received.
- Less overhead - The decoding nodes know how the message was encoded earlier in the network. Therefore no combination vector has to be sent along with the message.

We propose the opportunistic use of a predefined topology to identify localised Network Coding opportunities in WMNs. This approach may be useful to improve network utilisation of a WMN, because the deterministic approach to Network Coding requires little overhead.

5. METHOD OF IDENTIFYING NETWORK CODING OPPORTUNITIES

A method to indicate where in a WMN or how exactly at that specific location, Deterministic Network Coding could be implemented was not found in available literature. We proposed that opportunities for the implementation of Network Coding could be found by searching for "known Network Coding topologies"within larger networks.

The identification of a known network coding topology as a subset of a larger network also defines which nodes should change their forwarding function and which nodes should have the topology information necessary to enable decoding of the linear combinations of messages.

The following method to achieve local throughput gains in a WMN when using predefined Network Coding topologies, is presented:

1. Select a Network Coding topology of which the gain and capacity is known;
2. Derive the connection matrix of the larger network from a suitable distance vector routing algorithm;
3. Search the larger network matrix for the known topology structure;
4. Implement Network Coding at the appropriate nodes;
5. Re-iterate steps (3) and (4) after a routing update.

Three different search methods were examined to implement step 3.

5.1 Iterative looping

The first method was to translate the known Network Coding topology to an adjacency matrix and search for that specific smaller matrix within the adjacency matrix of the larger network.

The simplest method to do this is iterative looping. The fundamental concept of iterative looping is a nested looping structure. This effectively moves a window across the network adjacency matrix and continually compares the data in the current window to that of the desired data.

This method was implemented in MATLAB® to search networks of between 10 and 100 nodes for only one specific Butterfly adjacency matrix. The implementation therefore did not locate any Butterflies that were numbered or orientated differently.

Although this method was able to accurately locate a specific Butterfly adjacency matrix within the larger adjacency matrix, it scaled poorly and was computationally intensive.

5.2 Cross-correlation

The second method was based on the two dimensional cross correlation of the network's 1-hop adjacency matrix and the Butterfly adjacency matrix. The cross correlation of two datasets can be seen as the similarity of those two datasets and is commonly used to search for patterns in a random dataset by forming the cross-correlation.

This method was implemented in MATLAB® to search networks of between 10 and 100 nodes for all permutations of the Butterfly adjacency matrix. The concept of cross-correlation was extended to two dimensions with the network adjacency matrix being used as the random dataset and the Butterfly adjacency matrix being used as the known pattern. The result of the cross-correlation operation *xcor(adjacency, butterfly)*, a function that is based on the *xcorr* function found in the signal processing toolbox of MATLAB®, resulted in a new matrix M with values indicating the level of cross-correlation. In order to extract the correct values, the auto-correlation of the butterfly *xcor* matrix was formed. The location of the Butterfly matrix could be determined by scanning M for the correlation value.

Advantages of using the cross-correlation method is scalability, as the method remains computationally efficient when using larger matrices. A further advantage is the ability to detect where partial Butterfly structures occur within the network connection matrix. These partial structures will have the potential to become butterflies if the topology of the network changes slightly.

This method was compared to the Iterative looping method presented in section 5.1 for both accuracy and computational efficiency. For small matrices of less than 40x40 both methods performed in the same order in terms of computation time. For larger matrices of 100x100, the cross-correlation method's computation time was significantly less than that of the iterative looping method. As the matrix dimensions increase, the computational advantages of the cross-correlation method also increases.

All permutations of an adjacency matrix have to be pre-programmed as known patterns to search for in the cross-correlation method, making it cumbersome. It was however decided to implement another method because the first two methods took very long to complete a search of a network.

5.3 Concatenation

Graph theory is used to explain the third method: Concatenation.

The network is described as a directed graph *G(V,A,R)* where V represents the set of nodes in the network, A is the adjacency matrix describing the topology of the nodes V in $R = [r(e)]_{e\in A}$ that describes the link capacities. The adjacency matrix $A = (a_{ij})$ is an $n \times n$ matrix (n is the number of nodes in the network), with $a_{ij} \in \{1,0\}$. The rows and columns represent the nodes in the network and the *n'th* row and the *n'th* column represent the same node. A connection of one node to another is indicated by a *1* in the matrix and and a *0* indicates that a node is not connected (or within transmission range) to another node. In a network each node is at least connected to itself, so the main diagonal of the matrix will contain only *1's*. If these characteristics are combined, then:

$$A_{n\times n} = (a_{ij}) = \begin{bmatrix} 1 & \dots & a_n \\ \vdots & \ddots & \vdots \\ a_n & \cdots & 1 \end{bmatrix} \tag{1}$$

If the identity matrix I_n is subtracted from (1) then $B_{n\times n}$ is the true representation of the interconnectivity of the nodes:

$$\begin{aligned} A_{n\times n} - I_n &= \begin{bmatrix} 1 & \dots & a_n \\ \vdots & \ddots & \vdots \\ a_n & \cdots & 1 \end{bmatrix} - \begin{bmatrix} 1 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & & 1 \end{bmatrix} \\ &= \begin{bmatrix} 0 & \dots & a_n \\ \vdots & \ddots & \vdots \\ a_n & \cdots & 0 \end{bmatrix} \\ &= B_{n\times n}, b_{ij} \in \{1,0\} \end{aligned} \tag{2}$$

This representation of the network's topology is used to find all the paths of a certain length within a network, through concatenation. Every row of the matrix is examined to find all the *1's* present. Each time a 1 is found, a 1-hop route exists and this information is stored in an $r \times 2$ matrix $O_{r\times 2}$, where r is the number of 1-hop routes that exist, so that:

$$O_{r\times 2} = [o_{r1} o_{r2}] \tag{3}$$

where $\begin{matrix} o_{r1} = i \;\forall\; b_{ij} = 1 \\ o_{r2} = j \;\forall\; b_{ij} = 1 \end{matrix}$ when $1 \leq r \leq [(n-1)!]$ and $i < j$.

In some of the literature the Butterfly topology is called the canonical example of Network Coding, emphasising that the Butterfly typology is simple yet significant, without loss of generality. It was therefore decided to use it as a known Network Coding topology. Referring to figure 2 in section 4, it can be seen that there exists a 1-hop and 3-hop path between two sets of nodes. The two 3-hop paths share one common link, therefore to find the Butterfly topology, the 1-hop and 3-hop paths of the adjacency matrix are constructed in order to be used in the search algorithm.

The 1-hop paths are constructed and stored in (3). In order to construct the 3-hop paths, the 2-hop paths first are constructed through concatenation of the 1-hop paths. A $s \times 3$ matrix $T_{s\times 3}$ exists where s is the number of 2-hop paths, so that:

$$T_{s\times 3} = [t_{s1} t_{s2} t_{s3}] \tag{4}$$

where $[t_{s1}\ t_{s2}\ t_{s3}] = [o_{r1}\ o_{r2}\ o_{h2}]$ where $(o_{r2} = o_{h1})$ and $r < h \leq |\, O_{r\times 2} \,|$

The 3-hop paths are stored by creating a $u \times 4$ matrix $D_{u\times 4}$, where u is the number of 3-hop paths that exist, so that:

$$D_{u\times 4} = [d_{u1} d_{u2} d_{u3} d_{u4}] \tag{5}$$

where $[d_{u1}\ \ d_{u2}\ \ d_{u3}\ \ d_{u4}\ ] = [o_{q1}\ \ o_{q2}\ \ t_{s2}\ \ t_{s3}\ ]$ where $(o_{q2} = t_{s1})\ \forall\ q < s \leq |O_{r\times 2}|$,

or $[\ d_{u1}\ \ d_{u2}\ \ d_{u3}\ \ d_{u4}\ ] = [t_{s1}\ \ t_{s2}\ \ t_{s3}\ \ o_{q2}\ ]$ where $(t_{s3} = o_{q1})\ \forall\ q < s \leq |T_{s\times 3}|$.

The conditions in equation (5) respresent a concatenation of an 1-hop path with a 2-hop path or a concatenation of a 2-hop path with a 1-hop path.

To find the Butterfly topology, equations (3) and (5) are used to create a $c \times 4$ matrix $H_{c\times4}$, where c represents the number of 3-hop paths that have first and last entries that correspond with those of 1-hop paths:

$$H_{c\times4} = [d_{u1} d_{u2} d_{u3} d_{u4}] \tag{6}$$

where $(o_{l1} = d_{u1})$ and $(o_{l2} = d_{u4})$ $\forall\, u < l \leq |O_{r\times2}|$.

A search is performed in (6), the "wings"of the butterfly, i.e. the entries where:

$$(d_{u2} = d_{p2}), u < p \leq |H_{c\times4}| \tag{7}$$

which would confirm the existence of a shared link between sets of wings. An example of this method is presented in the following section.

5.4 *Example*

In this example, the Butterfly topology is used as the known topology that is the objective of the search in a random 10 node network.

One such random network, where each node is connected to exactly three other nodes (C = 3) is presented in figure 3a and the 10 x 10 connection matrix is shown in figure 3b.

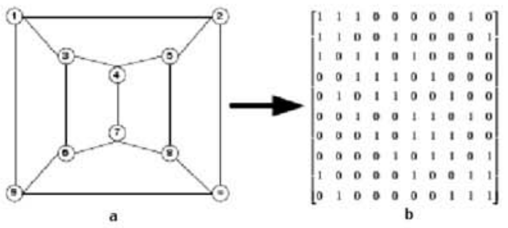

Figure 3: a) Random network b) Connection matrix of the network in (a)

Finding the known topology: The $B_{n\times n}$ matrix is created from the adjacency matrix presented in figure 3b. The 1-hop paths are obtained and stored in (8). There may be more paths if the algorithm allowed cyclic paths, but for the purpose of this example only non-cyclic paths are allowed.

$$O_{15\times2} = \begin{bmatrix} 1 & 2 \\ 1 & 3 \\ 1 & 9 \\ 2 & 5 \\ 2 & 10 \\ 3 & 4 \\ 3 & 6 \\ 4 & 5 \\ 4 & 7 \\ 5 & 8 \\ 6 & 7 \\ 6 & 9 \\ 7 & 8 \\ 8 & 10 \\ 9 & 10 \end{bmatrix} \tag{8}$$

The 3-hop paths are found by concatenation, using the conditions in equations (4) and (5):

$$D_{u\times4} = \begin{bmatrix} 1 & 3 & 6 & 9 \\ 1 & 3 & 6 & 7 \\ \vdots & \vdots & \vdots & \vdots \\ 3 & 4 & 7 & 6 \\ 4 & 3 & 6 & 7 \end{bmatrix} \tag{9}$$

Equations (5), (6) and (7) are used to determine whether there are any 1-hop paths and 3-hop paths that share the same start and end nodes. The information is stored in (10):

$$H_{u\times4} = \begin{bmatrix} 1 & 3 & 6 & 9 \\ 2 & 5 & 8 & 10 \\ 3 & 4 & 7 & 6 \\ 4 & 3 & 6 & 7 \\ 4 & 5 & 8 & 7 \\ \vdots & \vdots & \vdots & \vdots \\ 5 & 4 & 7 & 8 \end{bmatrix} \tag{10}$$

This matrix is then explored to determine whether any two of the 3-hop paths in (10) contain a shared second hop:

[1 3 6 9] and [4 3 6 7] share [3 6]; [4 5 8 7] and [2 5 8 10] share [5 8]; and [3 4 7 6] and [5 4 7 8] share [4 7].

It can therefore be concluded that there are 3 butterflies present in the network in figure 3b. The three butterflies offer three opportunities for the implementation of Network Coding. However, the three cannot co-exist, as they are not disjoint.

If the middle butterfly in nodes [3 4 7 6] and [5 4 7 8] is selected node 4 should have encoding capability and nodes 6 and 8 should have decoding capability.

6. RESULTS

The method as described in section 5.3 was implemented in networks consisting of between 10 and 20 nodes. A total of 1000 random connection matrices were generated for each size network. The created random matrix generator generated 1000 unique matrices, ensuring that a particular connection matrix is not used more than once, but not necessarily that it was not a permutation of a matrix that was already tested.

6.1 Butterfly Implementation

The generated matrices had connectivity (C) of between 3 and 7, (C = 3 meaning that each node was connected to 3 other nodes). A connectivity of 3 is the minimum connectivity that will allow the possibility of a Butterfly network to exist, but does not guarantee the existence of a butterfly network in large networks. The results of this implementation are shown in figures 4 - 6.

As can be seen in figure 4, when C = 6 or higher, all examined networks contained butterflies.

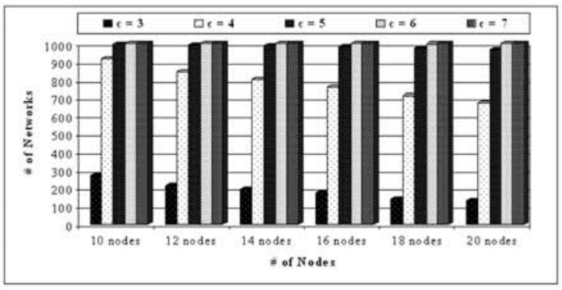

Figure 4: Number of Networks containing Butterflies

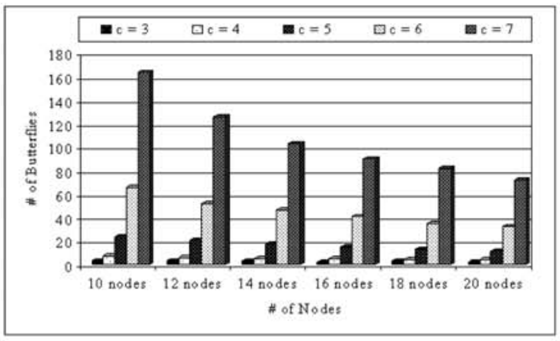

Figure 5: Number of Butterflies per Network

Further, when comparing figures 5 and 6, it can be seen that in all of the networks where butterflies were found and where C = 3, the butterflies were all disjoint. Further it was found that for low connectivity (C = 3 to 4), the number of butterflies stays relatively constant for all size networks, but with higher connectivity (C = 6 to 7), the number of butterflies decrease logarithmically with increasing network size. This may be due to the higher possibility of "islands"forming in larger networks.

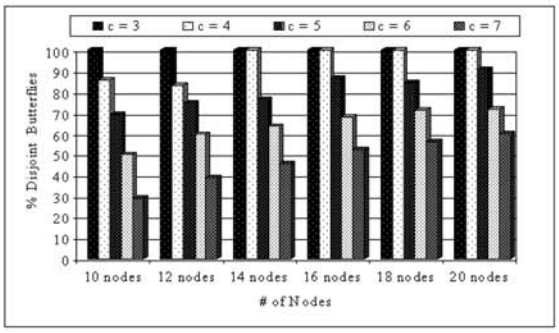

Figure 6: Percentage of Disjoint Butterflies per Network

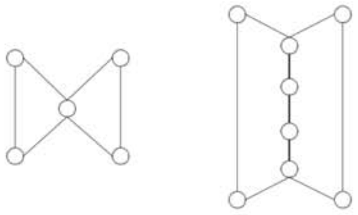

Figure 7: a) Bowtie b) Extended Butterfly

Two variations on the Butterfly topology (which test for a shared link) were also implemented. Firstly the topology was simplified to test for a shared node (refer to figure 7a), from here on referred to as a Bowtie. Secondly the topology was generalised to test for a shared path (refer to figure 7b), from here on referred to as an Extended Butterfly.

6.2 Bowtie implementation

To search for a Bowtie the parameters were selected as follows: C = 4 and higher and the shared 1-hop and 2-hop paths were used to find this topology. Again 1000 random connection matrices with between 10 and 20 nodes were explored.

Although most networks contained at least one of these Bowties, the number of Bowties per network was not very high. A connectivity of C = 6 yielded the best results in terms of presence of Bowties. For all other connectivities the number of networks containing Bowties decreased as the network size increased. This may be due to the complex nature of this topology (each node has to be connected to exactly four other nodes, two of which also have to be connected to each other on each side).

In figures 8 and 9 it can be seen that for low connectivity (C = 4 to 5), the number of bowties stay relatively constant for all size networks. With higher connectivity (C = 6 to 7) however, the number of bowties decrease logarithmically with increasing network size. Once again this phenomenon is attributed to the higher possibility of "islands"forming in larger networks.

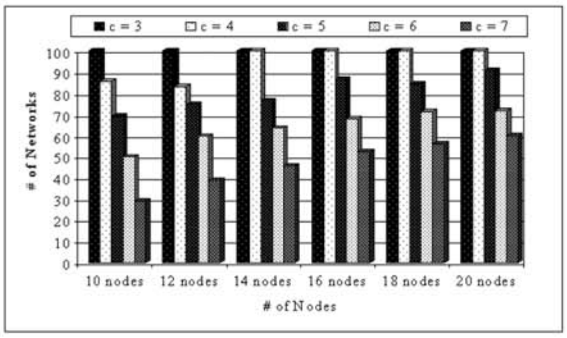

Figure 8: Number of Networks containing Bowties

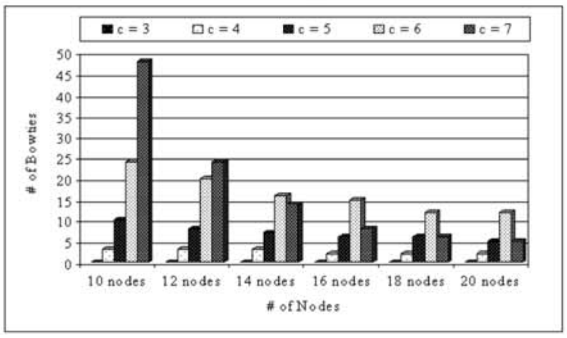

Figure 9: Number of Bowties per Network

6.3 *Extended Butterfly Implementation*

Finally, the method was implemented to find the Extended Butterfly topology. For this implementation, C = 3 and higher was selected and the shared 1-hop and 4-hop paths were used to locate this topology. The same size and same number of random networks as for the previous two simulations were used. The results of this implementation can be seen in figures 10 and 11.

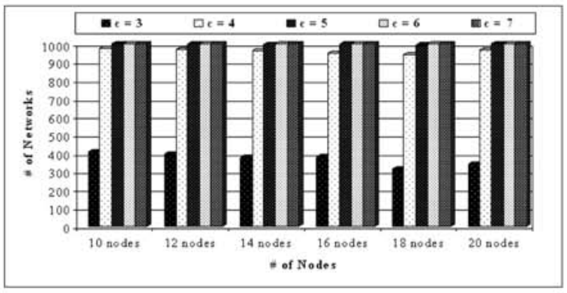

Figure 10: Number of Networks containing Extended Butterflies

This topology showed the same tendencies as the normal Butterfly, but it was found that more networks contained Extended Butterflies than normal Butterflies. When the connectivity C = 3, twice as many networks contain Extended Butterflies, but as can be seen in figure 11, there were not more Extended Butterflies than Butterflies per Network.

This may be because there are not very many opportunities for 4-hop paths in a sparsely connected network. All networks with C = 5 and higher contained Extended Butterflies. All three topologies tend to respond the same to connectivity and network size.

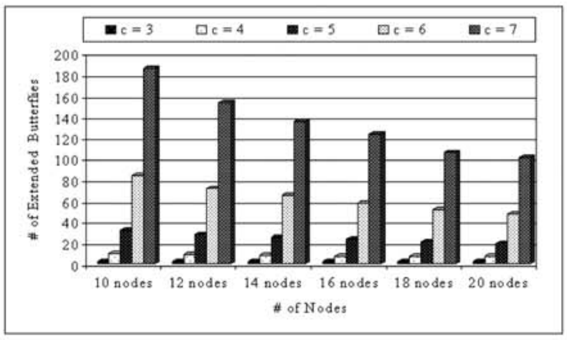

Figure 11: Number of Extended Butterflies per Network

In general it was observed that for all three topologies when connectivity C = 6, a significant number of predefined network coding topologies are probable in a random network. This seems to support [13] which suggested a connectivity of 6 for optimal throughput capacity of a mesh network.

7. THROUGHPUT CAPACITY OF A WMN

Calculating the capacity of a WMN is a complex problem due to the numerous factors that influence it [2], [3]. Studies have shown that a throughput capacity of $0.0976\sqrt{n}$ (where n is the number of nodes) is achievable for a network when C = 6 [12]. Analytical results have shown that the throughput capacity per node decreases as the node density increases [3].

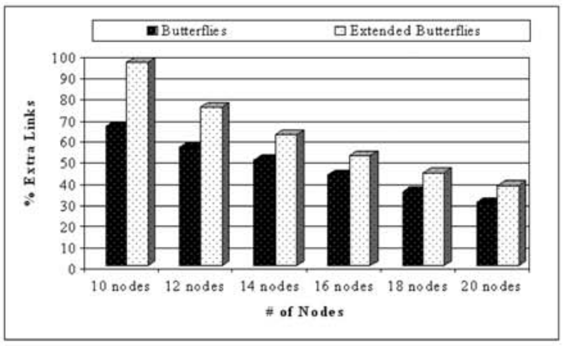

Figure 12: Percentage Extra Virtual Links

The implementation of Network Coding in a WMN can improve the throughput capacity of the network by better utilization of the available capacity. In figure 12 the improvement is shown in terms of the percentage of extra virtual links that are created in a network where C = 6 when using the Butterfly topology or Extended Butterfly topology.

8. CONCLUSION

In this paper a method was presented to determine where and how in a WMN, Network Coding can be implemented. It was implemented by means of three different appraches. The concatenation approach was more efficient than the iterative looping and cross-correlation methods, both in terms of finding topologies and in execution time. The concatenation approach was evaluated for three different topologies.

The implementation of the new method, which identifies opportunities for the implementation of network coding using predefined network coding topologies, provides more available links in the network. For each disjoint butterfly network present (or variation on this topology), one extra virtual link becomes available. Further anticipated benefits of the implementation of this concept are:

- An improvement in the total throughput (better utilisation of the network's capacity).
- Lower occupation of the total network.
- Improved Quality of Service (QoS), because of a lower delay in the network.

The implementation cost of using this method is:

- Clever nodes that can encode and decode received messages
- Route establishment delay
- A high connectivity is required
- Routing Protocol with knowledge of the whole network's topology
- Centralized network control

These costs can be overcome if the characteristics of the typical WMN are utilised, especially when the WMN is formed by nodes with significant computational power that are located within close vicinity of each other. Using laptops or mesh routers will ensure that the nodes have power to perform the computations for these clever nodes and will be able to compensate for the route establishment delay.

The nature of WMNs offer many possibilities for the opportunistic implementation of Network Coding. It is difficult to guarantee a specific increase in capacity due to the varying topology of the WMN. Theoretically this increase in topology can be as high as the maximum throughput capacity, but in practice this is seldom seen.

REFERENCES

[1] T. Ho, R. Koetter, M. Medard, D. Karger and M. Effros, "The benefits of coding over routing in a randomized setting", In Proceedings of the IEEE International Symposium on Information Theory June 2003, page 442, Yokohama, Japan, 2003.

[2] J. Jun and M. L. Sichitiu, "The Nominal Capacity of Wireless Networks", IEEE Wireless Communication. Magazine, vol. 10 pp. 8-14, 2003.

[3] I.F. Akyildiz, X. Wang and W. Wang, "Wireless Mesh Networks: A survey", Computer Networks, vol. 47, pp. 445 487, 2005.

[4] P. Elias, A. Feinstein and C. E. Shannon, "Note on maximum flow through a network", IRE Trans. Information Theory, vol. 2, pp. 117119, 1956.

[5] S. A. Aly, V. Kapoor, J. Meng and A. Klappenecker, "Bounds on the Network Coding Capacity for Wireless Rondom Networks", Information Theory and Applications Workshop, Jan. 29 2007-Feb. 2 2007, pp. 231-236.

[6] R. Ahlswede, N. Cai, S.-Y. R. Li and R. W. Yeung, "Network information flow", IEEE Transactions on Information Theory, vol. 46, July 2000, pp. 1204-1216.

[7] S.-Y. R. Li, R. W. Yeung and N. Cai, "Linear network coding", IEEE Transactions on Information Theory, February 2003, vol. 49, pp. 371- 381.

[8] A. G. Dimakis, P. B. Godfrey, M. J. Wainwright and K. Ramchandran "The Benefits of Network Coding for Peer-to-Peer Storage Systems", NetCod Workshop, January 2007.

[9] C. Fragouli, J. Widmer and J. Le Boudec "Network Coding: An instant primer", ACM SIGCOMM Computer Communication Review, January 2006, vol.36, pp. 63 - 68.

[10] R. Koetter and M. Medard, "Beyond routing: an algebraic approach to network coding", In Proceedings of the Twenty-First Annual Joint Conference of the IEEE Computer and Communications Societies, vol.1, pp. 122- 130, 2002.

[11] C. Fragouli, J. Widmer and J. Le Boudec "On the Benefits of Network Coding for Wireless Applications", Netcod, 2006.

[12] L. Kleinrock and J. Silvester, "Optimum transmission radii for packet radio networks or Why six is a magic number", in: Proceedings of the IEEE National Telecommunications Conference, Birmingham, Alabama, pp. 4.3.14.3.5, 1978.