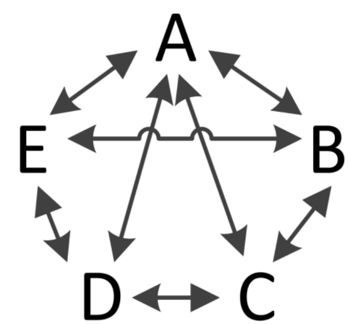
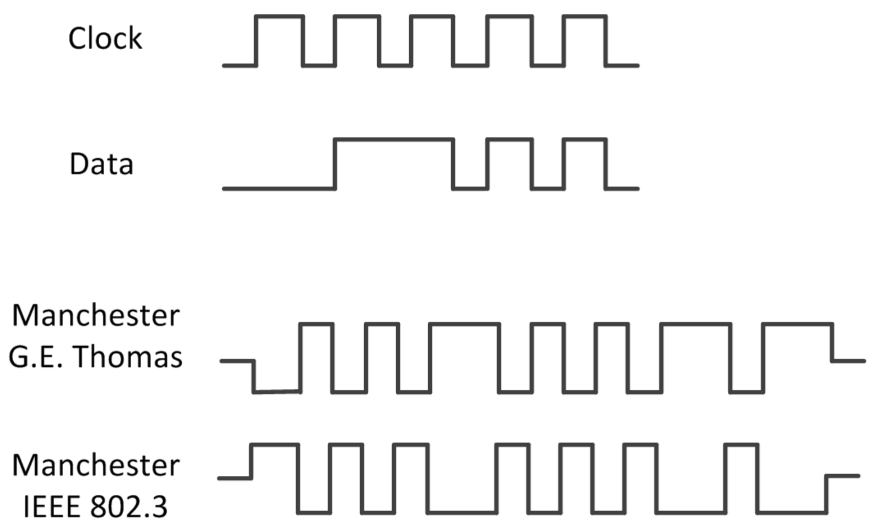
COIS 4310H – Assignment #2
Simon Willshire (0491272)



1.
    a. A->B: A->B->**C**, A->B->**E**
    b. B->A: B->A->**C**, B->A->**D**, B->A->**E**

2. MACA Protocol grants access for sender to be received. No, the nodes whilst in range request to send data using "Request to Send" (RTS), and the receiver if able must accept using "Confirm to Send" (CTS) allows the packet to be sent. The other nodes "hear" and defer until the network is clear and reply to incoming or request to send once more.

3. Manchester Encoding (Classic Ethernet) => Encode 0001110101



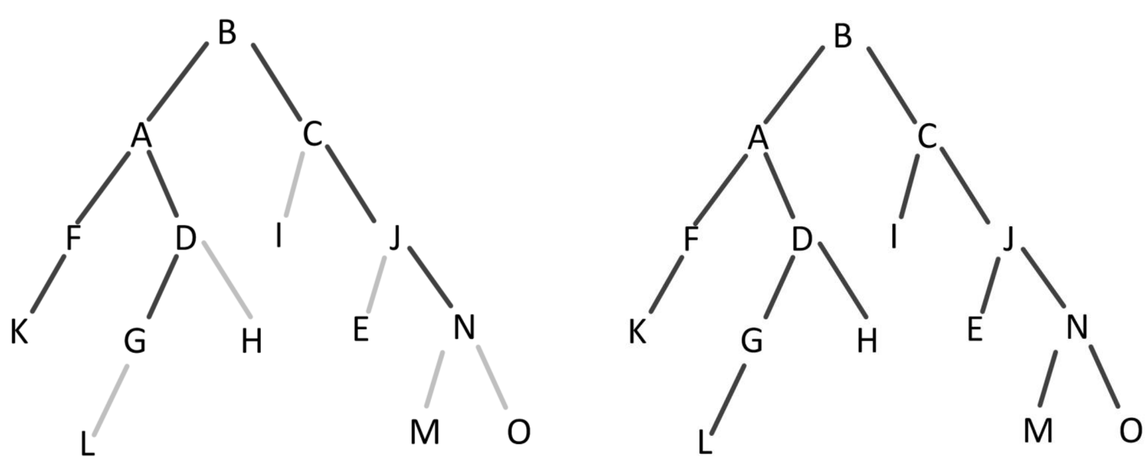
4.
    a. A->C: B1: 2,3,4
    b. E->F: H1: B2: 2, H1 rebroadcasts back to F as well.
    c. F->E: H1, B2: 2, H1 rebroadcasts back to E as well.
    d. G->E: B2: 3 (Recv), 1,2,4
    e. D->A: B2: 1 (Recv), 2,3,4 -> (B1: 4(Recv), 1, 2, 3)
    f. B->F: B1: 2 (Recv), 1,3,4 -> (B2: 4 (Recv), 1, 2, 3)

5. C->B: 6, C->D: 3, C->E: 5

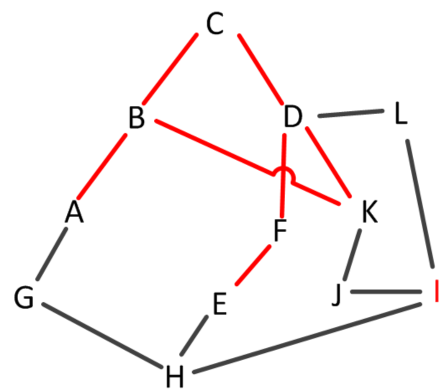
	A	B	C	D	E	F	As C	Route
A		5		16	7		5	B
B		0		12	6		6	B
C		8		6	5		3	E
D		12		0	9		3	D
E		6		9	0		5	E
F		2		10	4		2	B

6. Broadcast from B
   a. Reverse path forwarding: Node receives broadcast packet, if packet arrived on shortest path to B, flood neighbours. Otherwise, ignore broadcast packet.
   = 28 (right figure)

   b. Sink Tree = 14 (left figure)

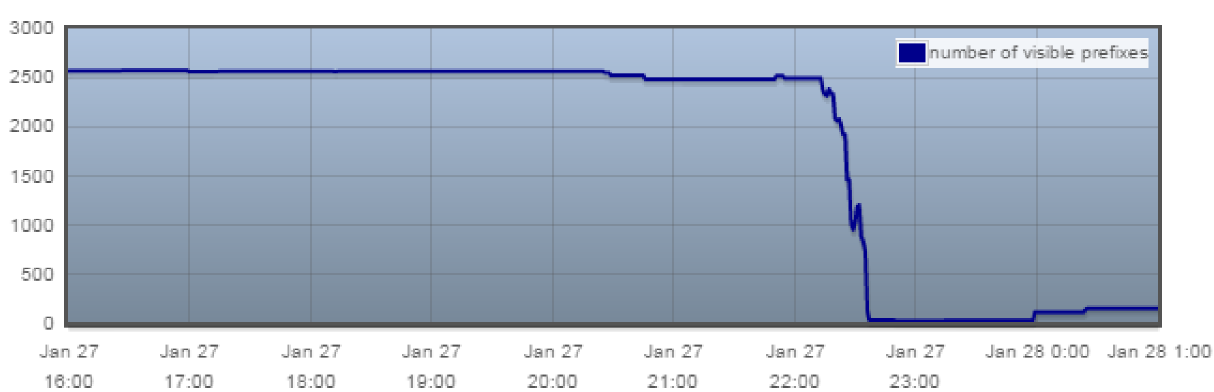


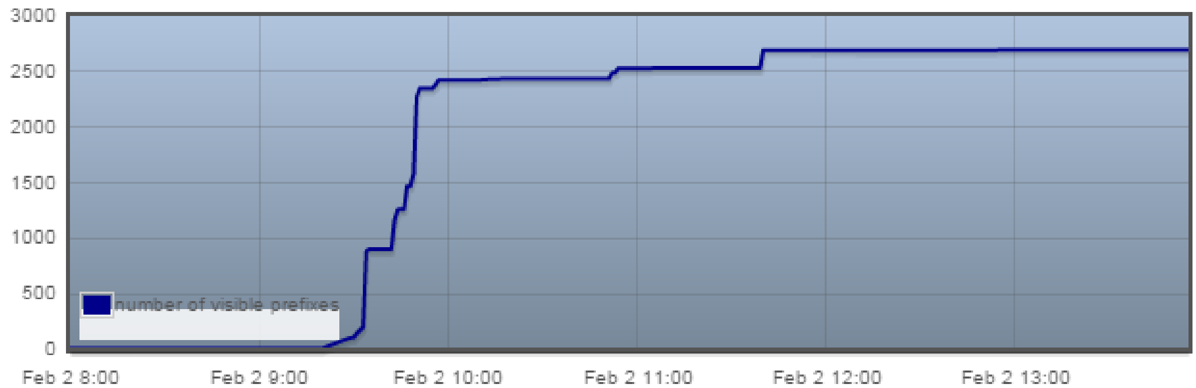
7. Multicast Spanning Tree for C in {A, B, C, D, E, F, I, K}



I: Does not multicast, no immediate multi-casted parent.
{G, H, L}: Nodes do not receive. (leaf nodes)

8. Egypt was able to blackout their internet coverage by removing border gateway protocol (BGP) routing information. This was accomplished by mandating minor adjustments to their ISPs (Four main ones: Telecom Egypt, Link Egypt, Vodafone/Raya, and Etisalat Misr, as well as its three main mobile carriers: Vodafone Egypt, Mobinil, and Etisalat). This modification essentially removed pathing and addressing information from the sender so that Egypt was unable to receive network traffic. Days earlier Twitter and Facebook were blocked. However the BGP method of disconnection is most likely the most effective and efficient way of taking down a large network, as simply disconnecting cables may be unpredictable and cause liability issues. The change was able to be accomplished in the matter of minutes to propagate between routers, for a grand impact of wiping out 88% of the countries internet. Interestingly only a single ISP was left standing during the collapse, Noor Data Networks was left functional. It is speculated that this carrier was left running in order to maintain the Egyptian stock exchange [3].





Static view on the number of visible prefixes originating from Egyptian organisations on 2 Feb between 8:00 and 14:00 UTC

[4]

In doing this, piggybacking providers (with as afore questioned countries; Sudan and Eretria) would obviously become disconnected. ISPs to these downstream countries were slowly restored within hours, for those that were affected by Egyptian carriers. The beauty of the BGP methodology is that the ISPs still have complete control of designated areas, ie they may selectively allow the BGP broadcast to resume in certain areas of their network. The truly astonishing factor being that, this outage was by no means legal and following people's rights. It was done in short to attempt to stabilize and maintain order during the chaotic riots in 2011.

References

[1] http://www.computerworld.com/article/2512745/internet/how-egypt-pulled-its-internet-plug.html
[2] http://en.wikipedia.org/wiki/Internet_in_Egypt
[3] http://www.telegraph.co.uk/news/worldnews/africaandindianocean/egypt/8288163/How-Egypt-shut-down-the-internet.html
[4] https://stat.ripe.net/events/egypt

9. Ad hoc mobile phone networks provide a way for mobile devices to connect without any centralised infrastructure; it is self-configuring and consists of entirely mobile nodes. This network topology makes use of multi-hop relays, similar to peer-to-peer networking, however distance restricted and completely wireless. The implementation of such a network would be most suitable to densely populated areas that is flat and above ground (unless lots of people using the network were underground…). These limitations are implied by the use of mobile technology, as sparse density would lead to dropped network transmissions, or signals generally getting obstructed by terrain, metal or concrete. However, with all this said, this topology is able to be deployed completely by the devices using it, and as such remote or impoverished regions it becomes very useful for communication. Many ethical/legal problems also may arise, as the network is completely unrestricted and unable to be censored and screened by the previous centralised solution. As with any network system, more acronyms have been used to describe this self-configuring topology,

consisting of but not limited to Vehicular Ad hoc Networks (VANET), Smart Phone Ad hoc Networks (SPAN), Internet based mobile ad hoc networks (iMANET), or more generally mobile ad hoc network (MANET). Some other protocols are capable of dealing with short periods of delay (Delay-tolerant networking), provided each node is capable of storing incoming data to later broadcast once able [2].

Other than typical communication via voice or messaging, other data may be carried through this topology, there are systems that are used for data monitoring, where nodes with sensors may interact with one another at mobile locations, this allows for spatial correlation between sensors, making it a very powerful and low cost technique [3].

As previously mentioned, security is very limited in these networks, it relies much on the same basis that the internet originally was created on, “trust based security”. There is a difficult trade-off to consider as with most software, the age-old security vs. performance dilemma, considering that the standard node in a typical communication network is a smartphone, with limited power and resources (even with the drastic performance increase over the last couple of years).

Overall this technique is a great solution for low-cost, high density, low interference solution, where the typical centralised solution is either non-existent, or not within the interest of the user. All of this comes at severe implications to speed, reliability, and security.

References

[1] http://en.wikipedia.org/wiki/List_of_ad_hoc_routing_protocols
[2] http://en.wikipedia.org/wiki/Delay-tolerant_networking
[3] http://en.wikipedia.org/wiki/Mobile_ad_hoc_network

10. IPv4 to IPv6 Address Conversion

```
// COIS-4310H - Assignment #2
// Simon Willshire (0491272)
// #10: IPv4 >> IPv6 Address Converter

#include <stdlib.h>
#include <stdio.h>
#include <string.h>
#include <stdint.h>

// Assumes valid IPv4 Address
// http://stackoverflow.com/questions/10513841/using-strtok-to-split-the-string
//      IN:     @addr: IPv4 address (#.#.#.#)
//      OUT:    @ipv4: unsigned char array [4] with byte values
void strtoIPV4(char* str, unsigned char* ipv4)
{
        int i = strlen(str);
        char* addr = new char[i+1];
        memcpy(addr, str, i);
        i = addr[i] = 0;

        // Loop through segments sep by '.', convert to uchar
        addr = strtok(addr, ".");
        do ipv4[i++] = (unsigned char)(atoi(addr) % 256);
        while(addr = strtok(NULL, "."));

        delete addr;
}

// Assumes argv[1] = IPv4 address with correct formatting (#.#.#.#)
//      OUT: Outputs IPv6 address, uses 0-15 hex conversion
int main(int argc, char** argv)
{
        const char* hl = "0123456789abcdef";
        unsigned char* ipv4 = new unsigned char[4];

        strtoIPV4((argc > 2) ? argv[1] : "127.0.0.1", ipv4);
        char ipv6[] = {
                hl[(ipv4[0] / 16)], hl[(ipv4[0] % 16)],         // <uchar / 16><uchar % 16> per byte
                hl[(ipv4[1] / 16)], hl[(ipv4[1] % 16)],
                ':',                                            // IPv6 colon formatting in centre
                hl[(ipv4[2] / 16)], hl[(ipv4[2] % 16)],
                hl[(ipv4[3] / 16)], hl[(ipv4[3] % 16)]
        };
        printf("IPv4: %s\nIPV6: 0:0:0:0:0:ffff:%s\n", argv[1], ipv6);
        return 0;
}
```

Output { 174.142.32.131, 192.168.1.1, 127.0.0.1, 74.125.29.94 }

```
root@alpha:~/Documents/IPConverter# g++ -std=c++11 ipconv.c -o ipconv
root@alpha:~/Documents/IPConverter# ./ipconv 174.142.32.131
IPv4: 174.142.32.131
IPV6: 0:0:0:0:0:ffff:ae8e:2083
root@alpha:~/Documents/IPConverter# ./ipconv 192.168.1.1
IPv4: 192.168.1.1
IPV6: 0:0:0:0:0:ffff:c0a8:0101
root@alpha:~/Documents/IPConverter# ./ipconv 127.0.0.1
IPv4: 127.0.0.1
IPV6: 0:0:0:0:0:ffff:7f00:0001
root@alpha:~/Documents/IPConverter# ./ipconv 74.125.29.94
IPv4: 74.125.29.94
IPV6: 0:0:0:0:0:ffff:4a7d:1d5e
```

11. Traced trentu.ca stream, selected initial HTTP GET request with text/html response: