Topology of Bidirectional Mesh Networks and Subsequent Systems

James Sherman
School of Computer Sciences and Computer Engineering
Hattiesburg MS, United States
James.Sherman@usm.edu

Dr. James L. Clark
School of Computer Sciences and Computer Engineering
Hattiesburg MS, United States
James.l.Clark@usm.edu

***Abstract*— With the coming surge of smart devices in the next decade, a new system of network architecture will need to be built to secure and accommodate the IoT. One of the methods explored is that of Bidirectional Network Systems. This topology uses the individual device as a routing point to two or more other devices at any given time. This allows a device to send and transport packets along a chain of devices to eventually reach an outside connection and access data from the web or from other devices in the network.**

Keywords*— *IoT, IT, UDP, TCP, IP, Packet, Cyber Security, Bidirectional Network, Branch Node, Trunk Node, Queue, Priority, Device

I. Modern networks

In the modern technological world, we rely upon tested and verified IT architectures to provide network connected services to consumer devices. Schemas for routers, switches, firewalls and most other devices have increased in complexity over the years but not in topology or theoretical limit. While this works with a steady incline of devices, the world is expecting a massive boom in IoT based devices in the coming years [1].

Table 1.

Year	***Devices Per Billion***
2018	23.14
2019	26.66
2020	30.73
2021	35.82
2022	42.62
2023	51.11
2024	62.12
2025	75.44

From table 1 above, you can see that device per billion boom over the coming years grows exponentially. In 2018 alone we have close to 23.14 billion IoT devices in the world. In less than a decade, that number is expecting to skyrocket in 2025 to a staggering 75.44 billion devices that are all IoT [1]. This rapid explosion of devices poses a myriad of challenges and new risks in the current environment. For one, these new devices are all going to require connection and access to numerous numbers of networks all across the planet. This is billions of new and unique IP addresses, higher volumes of network traffic, and more variables for network engineers to take into consideration.

Scalability and viability of entire networks comes into play when this sheer amount of devices come online. Modern routers and access points can support 255 total devices across a single router but bandwidth is then decreased per user as it is shared across a single connection [2]. This can lead to increased bottlenecks, costlier topologies for businesses, and possible new and costly solutions to handle the sheer amount of IoT devices. This is where the theory of Bidirectional Network Systems play their part.

II. introduction to bidirectional systems

With these problems being introduced and time being an important factor, network engineers and programmers have been taking a look at different ways to handle these mounting problems. One of the more creative solutions is that of bidirectional networks. The name in itself gives a heavy connotation as to what it does. Bidirectional meaning functioning in two directions and network meaning a group or system of interconnected people or things, one can assume the systems application. In traditional networks and network packet movement we use TCP (Transmission Control Protocol) and UDP (User Datagram Protocol). These systems make up the direct backbone of synchronous and asynchronous network use [4].

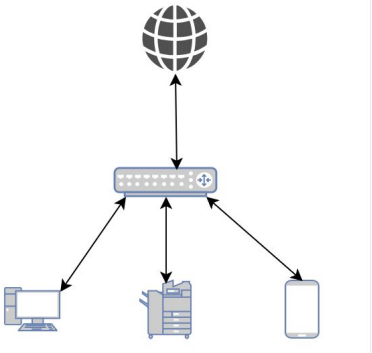
Bidirectional networking combines both TCP and UDP with an internal device routing program to make a chain system for devices to funnel packets across a network. Each device can create its own customizable routing table, and packet storage system to chain along the packet requests to the next device node. Eventually the packet will hit the intended device, fail, or reach the world wide web to go onto another destination. This allows for IoT devices to be more secure as it is only sending the intended information in a

singular line to trusted device that can be monitored . The interesting facet of this technology is that engineers and system administrators can pull a device off or replace it on the network and alter the device routing table. This allows for the flow of network traffic to be alleviated at congested points or low processing power devices. Another important note is that the system is a mesh network. It uses numerous outgoing connections that are not linked to the central device, but to all devices in a branch pattern and can be connected directly, dynamically, or non-hierarchically to a indeterminate amount of other device nodes.

III. Single Point Router Topology vs Bidirectional Topology

In an interesting turn of events, modern internet connection and device routing rely upon a single point router topology for connection to the internet. In most systems, packets are sent to a primary distributor, such as the router, and then routed to their respective device destination via IP[4]. Part of the problem with this is that if a router is compromised, and not every device is secure or has hardware capabilities to be, a hacker can watch all incoming and outgoing traffic [3]. If an attacker has managed to infiltrate your system, and more specifically your router, then issues are going to arise based upon the fact that your network service is now being watched and manipulated. Any packet of information that is picked up on the router can now be accessed, read, or manipulated by a skilled hacker.

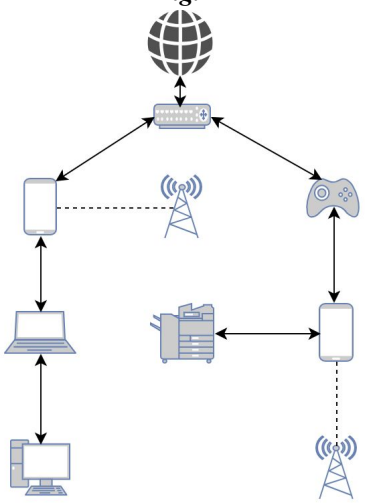
Fig 1.



Standard Device to World Wide Web Connection

In figure 1, we see that computers, smart devices, and phones can connect to an outgoing router via Wifi or direct cable connection. This allows us a single point of interface to access the world wide web or our other machines. This system is laden with issues and possible vulnerabilities but this makes up a large amount of modern internet connected topologies in various formats.

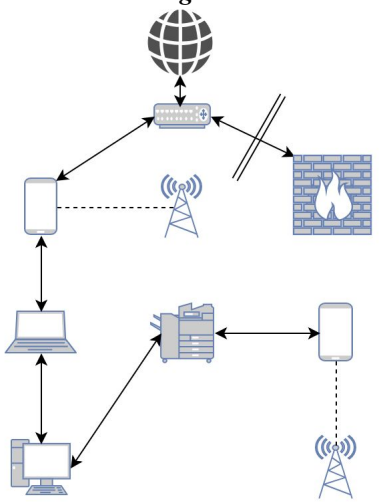
Fig. 2



Bidirectional Connection to World Wide Web

The interesting part of bidirectional connection, shown in figure 2, is that it allows for multiple outlets to be used for packet transmission in case of faulty or compromised devices. In figure 2, we can see that different chain systems link out to the world wide web in different forms. We have the traditional router interface we spoke of earlier but we also have phone connection to a cellular tower in the same figure. One of the main reasons for this is internal security. Linking and pushing packets between individual devices is safe in a closed local area network, but having the ability to bring data into our secure network via different outside connections is an interesting prospect. This is also bolstered by the idea of having a security node to look over the incoming data. In case one of our outside linking devices is compromised, we can simply reroute the network to feed into a new node and pass our checkpoint going in and out again like so:

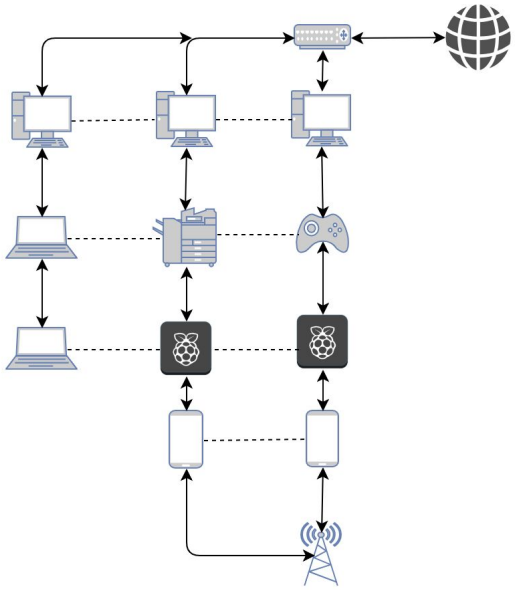
Fig. 3



Bidirectional Network Change

In figure 3, we have a small demonstration at how bidirectional change works in comparison to figure 2. In our mock figures we determined that one of the devices became compromised. A simple adjustment of connection configuration allows for redirection of the traffic that the device was passing and allows it to move onto another device node. Both the router and the smartphones here allow for inward and outward bound connection and can allow or deny access into our closed local area network. Because we have the ability to change what device was linking and transporting our data, we still have a secure network and can monitor incoming or outgoing connections without having the corrupted device to worry about. This can be scaled to larger topologies as in the next figure.

Fig. 4



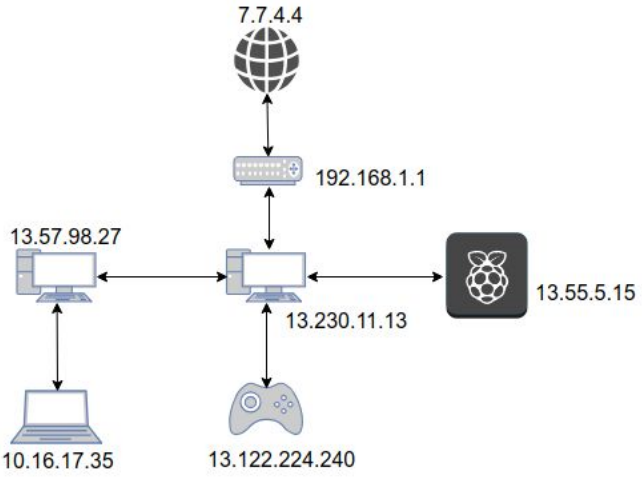
Possibilities in a Larger IoT Local Network

In figure 4, we can see that bidirectional networks can support a host of devices and can have a multitude of branches. In the figure, dotted lines are representing the possible connections in case a device fails or if rerouting is necessary. While it is not inclusive of all possible connections it does show the power and configurability of a network that is able to change how packet traffic is directly accessed, may it be in a traditional device or in an IoT system. Solid lines with arrow tips are showing where and how traffic may possibly be filed to reach the outside world or to other devices. In our figure, both our phones and standard desktops are acting as our closed local area network security nodes that are surveilling packet by packet in case of issues. These same machines are also the last transport device before hitting both our router and our outgoing tower connection. With this full topology we can message and access the other devices on our network without having to worry about rebounding the connection from a singular source device depending upon configuration. Another key note is that the full system allows us to employ all the standard methods of routing and security ,all the while, ensuring that our incoming and outbound packets are safe by using security nodes..

IV. SECURITY PRINCIPALS OF BIDIRECTIONAL SYSTEMS

As seen in the previous figure set 1 - 4, bidirectional networks can be connected in innumerable configurations. Since the system does not require true router access to move packets between devices, we have to have a security stack to help secure the bidirectional network. For reference purpose, as we are exploring the parts of the bidirectional system inductively we will use a reference to a stream of data packet incoming from an outside service with an ipv4 address of 7.7.4.4 to a local machine inside the bidirectional network with ipv4 10.16.17.35. These simple ipv4 addresses in the coming figure 5 are used to distinguish where and how the packet is moving and how it interacts with each device and the systems involved in checking, routing, and delivering the information to the correct device.

Fig 5.



Test Case Bidirectional Home Network

Figure 5 represents our mock network for testing with a simple edition that the desktop computer [13.230.11.13] acts as our security hub and a trunk device. All devices bi-directionally access this node to interact with outside connections and other devices inside of the network topology.

The first request needed from our system is the initial call from [7.7.4.4]. This call, routing through the respective devices originating from our sender is going to send packets of data that cross our security boundary and then rerouted to be delivered to [10.16.17.35]. As [7.7.4.4] sends the packets, our router at 192.168.1.1 routes the incoming connection past any network security measures at the physical hardware level and then arrives at our security node device [13.230.11.13].

This security node device is unique in how it functions as a sentinel inside of the bidirectional network set. First when the incoming data packet arrives, standard security measures are taken to analyze the incoming packets. The main security tools included in this step fall into two choices for bidirectional network security in testing [5]. The first of these tools is that of the network protocol analyzer. As for a formal definition, a network analyzer (also called a protocol analyzer or packet analyzer) is a combination of hardware and programming, or in some cases a stand-alone hardware device, that can be installed in a computer or network to

enhance protection against malicious activity. Network analyzers can supplement firewalls,anti-virus programs, and spyware detection programs [5]. This invaluable tool helps monitor incoming connections and packets into the system and , along with standard security tools such as a firewall, watches over all incoming traffic to help prevent malicious attacks or dubious entry into the closed bidirectional system.

The next topic at hand that helps secure the bidirectional network is that of the encrypted traffic analysis system. Most modern day systems do not have a solution to detect malicious content in encrypted traffic. Networks lack the security tools and resources to implement a solution that can be deployed throughout the network infrastructure without slowing down the network. Traditional threat inspection with bulk decryption, analysis and re-encryption is not always practical or feasible, for performance and resource reasons [6]. In many cases, however, advanced analytic techniques can be used to identify malicious flows for further inspection using decryption techniques [6]. On any given day, no one knows how much of their digital traffic is in the clear versus encrypted. If traffic is encrypted, the encryption is typically done to meet standard compliance requirements that mandate specific security policies may it be in home, business, or large infrastructure [6]. This is where a lightweight solution like Encrypted Traffic Analytics comes into play.

Encrypted Traffic Analytics focuses on identifying malware communications in encrypted traffic through passive monitoring, the extraction of relevant data elements, and supervised machine learning to detect future threats [6]. Encrypted Traffic Analytics extracts four main data elements: the sequence of packet lengths and times, the byte distribution, TLS-specific features, and the initial data packet [6]. Encrypted Traffic Analytics also identifies encryption quality instantly from every network conversation providing the visibility to ensure enterprise compliance with cryptographic protocols. It delivers the knowledge of what is being encrypted and what is not being encrypted on the network so that a system can be fully covered and secured.

The importance of having these low weight analytic and watch based tools on our security node system may be proverbial “overkill”, but there are reasons behind having such heavy layered security. While bidirectional networks are a good solving point to the bulk of the problems regarding IoT security and overproduction, major flaws are opened up to the network when outside connection is added in. Because of how packets are stored, redirected, encapsulated, and sent there are major vulnerabilities in how the system passes data along its branch lines.

V. HOW PACKET BIDIRECTION WORKS

As we have discussed in earlier sections, part of the explanation for how bidirectional systems operate needs to be explained inductively as to see the full scale of the system. If we recall, we used figure 5 to show how a mock network would operate. So to continue on with our packet stream example, from this point the packets coming from 7.7.4.4 have passed our router, crossed the checkpoint, and are being setup to be moved into the next device in the chain from ip’s [13.230.11.13] to [13.57.98.27].

This bidirectional movement occurs with a couple different aspects. The first we are going to cover is the routing system. The routing system inside of a bidirectional network is incredibly important to each device. All devices in the system have a stored list location of all other connected devices in specific order so that they can pass information along a line based system. Other possible backup devices for configuration or rerouting traffic are also stored in a redirection list for attempted contact in case of continuous failure of service to the intended normal node. These related lists are used to store individual addresses that map to one another. As a reference to our figure five, we will use the device ip [13.57.98.27] as an example of related lists.

Device [13.57.98.27] knows the location of the next two device nodes [10.16.17.35] as well as [13.230.11.13]. These are the only devices that we can pass a packet to under normal circumstances as a “branch” of the network to reach our goal. A third address or what is labeled as a “safety” address is also stored from [13.55.5.15] in a separate list for emergency connection. In case one of the primary nodes fails, the [13.57.98.27] device can attempt to contact the [13.55.5.15] and establish a bidirectional connection and continue moving network traffic through the new node device. If a route outside this layer of singular connection safety fails, it should be noted that administrators should be able to use simple interfacing commands to help reconfigure the system and reroute to other nodes. It is also possible to increase the emergency list size to hold more than one emergency contact ip in the network.

Other features include handling rerouting and turning devices into what have been dubbed “trunk” devices that act as pseudo routers for packet transportation. Each device has a primary and secondary function as either a “branch” that only has two connections, or a “trunk” that acts as a rerouting device to pass packets to a correct destination. Trunk devices are similar to that of our security node device [13.230.11.13]. This style of configuration differs in the sense that, instead of only two nodes of connection, this device can support as many logical connections that can be attached to the device. The trunk device has a few key pieces of information that allow this to happen. First, the trunk device acts similar to a road intersection. The device has multiple routes that it can take to redirect traffic or move packets to and fro in hopes of reaching the intended direction. The main way this is handle is by having, similar to the standard two node connection of the primary bidirectional function, a system of lists in which the routes can take. When a packet or stream is passed into the trunk, the system then identifies what intended ip addresses are on local network and performs a check to see if the intended location is one within the bidirectional network. If the intended location is found in one of the lists in the system, the device redirects the packet or stream onto the intended route to reach its destination. To give reference to how this works, remember our packet analogy from figure 5 is being moved from [13.230.11.13] and is now in queue to be directed to [13.57.98.27]. The trunk system of device [13.230.11.13] checks the packets and notes the intended address is [10.16.17.35]. It then goes through these lists to determine if our node is present.

- List 1: [13.55.5.15] >> [end of list]
- List 2: [13.122.224.240] >> [end of list]
- List 3: [13.58.98.27] >> [10.16.17.35] >> [end of list]
- list 4: [192.168.1.1] >> [outbound connection]

Our trunk node spots that on list three that 13.58.98.227 has a direct connection to [10.16.17.35]. It then chooses the list three connections to route the packet. The data is then loaded onto our next step, a queue, and is put in line to be sent to our next device.

VII. BIDIRECTIONAL QUEUES

The next feature of bidirectional devices is how the individual system handles data and sending packets onto the next device in the chain. As we have talked about, each device is either a branch or a trunk and each of these modes carry different lists that identify various locations of nodes in the network. A system implementation of how the packets are handled before being sent out is a large portion of what makes the bidirectional system viable. There are two primary methods of handling this system. First is that of the standard queue, and the second is that of the manually configurable priority queue. These systems work just like any standard queue data structure but have included metadata and systems in place to help the machine to decide what order the packet will be sent to and where it will be sent.

For most topologies, the queue is the most applicable option to handle how packets and moving across the network. To show why this important, reference back to the current stage of our figure 5 packet movement.
We identified that list 3 was the correct path to reach our intended destination with the list values being:

- List 3: [13.58.98.27] >> [10.16.17.35] >> [end of list]

Our trunk device then handles the routing by scheduling the packet into the queue. The table below shows the metadata used in storing the packet in the devices queue before it is sent, along with metadata about the packet, off to the next device to be read or sent along the line.

Table 2.

Destination	List Entry Address	Packet Value in Queue
13.122.224.240	13.122.224.240	1
13.57.98.27	13.57.98.27	2
10.16.17.35	13.57.98.27	3

Each packet to be sent out is pushed into the queue consecutively and then is reordered according to what data is received and generated to handle traffic on the device. In table 2 we see that one of the packets in our stream has generated metadata to be stored and used as a control operator for our queue. Of the data elements generated about the packet, there are three primary data objects used to determine what packet is to be sent, and what information is also passed along to the next device.

- Destination - the destination is an obvious important characteristic of what to pass to the next packet. This is labeled in the packets data as intended destination. Once the destination address has been identified, it's an easily packaged and passable characteristic that can aid in getting the packet to the correct address.

- List Entry Address - this part of the queue functions as an entry point for where the packet to go. Having this list entry address helps to identify what direction the trunk device needs to send the packet. This value is also updated on each device to help with console logging possibly corrupt or ruined packets. Each device can also check to see if it is part of the concatenated list of address entries to assure packet destination arrival happens. A check can be performed to see if this is the case. If so the information continues as usual, else the packet can be resent back to the trunk device or completely thrown away.

- Packet Value in Queue - this part of the queue is more for error handling and console logging of information. It is not sent as metadata as it is only useful for the specific device. Nevertheless, if a device system is faulty knowing if the queue system is not performing duties adequately is incredibly important.

VII. PRIORITY QUEUES

The dynamic queue is the alternative to the standard queue. It is a manual configuration to facilitate device precedence for certain nodes in the bidirectional system. Any time a packet or packet stream is sent from this device, it is given precedence over all other sending requests on any other device.

It contains two extra data pieces that the standard packet queues do not have.

Table 3.

Priority	Signature
True	1
True	2
False	0

The priority queue contains passable values of priority and signatures of priority value. Outside of the passed packet and packet metadata, these values tell if this piece of information needs to be sent out ahead of any other packet.

The priority value is a Boolean value signifying if the packet has precedence. If the device is deemed as a priority system, then this extra data is tacked on to the information passed to the next node in the system as metadata. Since

there are only true and false values, non-priority devices only carry a False flag as their assignment. The signature is the second passed value in a priority queue. This value is a checking value in case there are multiple priority systems sending packets simultaneously. This allows certain devices to maintain order of importance when sending data across a queue. As a note, numeric positive integers are the main value of Signature and denote precedence of being sent in case of priority device conflict.

To show an example of how the priority and signature interact we have made an updated table two to show how these variables for the queue interact.

Combined Table 1.

Destination	List Entry Address	Packet Value in Queue
13.122.224.240	13.122.224.240	1
13.57.98.27	13.57.98.27	2
10.16.17.35	13.57.98.27	3

Priority	Signature
False	0
False	0
False	0

In the next combined table, we see that table 2 has been recreated but with the inclusion of the priority and signature category. The original packets that are in the queue have priority Boolean values of false and their signatures are 0.

Combined Table 2.

Destination	List Entry Address	Packet Value in Queue
7.7.4.4	192.168.1.1	1
7.7.4.4	192.168.1.1	2
13.122.224.240	13.122.224.240	3

Priority	Signature
True	1
True	2
False	0

These outgoing packets to [7.7.4.4] have been labeled with positive value signatures and a priority value of True. They replace the current leading outgoing packet of [13.122.224.240] as the primary concern of the queue. This allows these express packets to move across the network with an expedited pace over any other non-precedent machine.

As to end the transport of our packet stream example, after identifying the correct address list, harvesting metadata, and loading the queue the packet stream is moving to [13.58.98.27] and to [10.16.17.35] subsequently. Sent with them is any relevant metadata that is needed for the queue of [13.59.98.27] and to eventually check that the destination at [1-.16.17.35] is correct before being read.

VIII. Connection System of the Network

Thus far we have discussed the topology, list location, and queue architecture of the bidirectional system but there is one more facet that is intrinsic to how the system functions. This factor is the connection medium.

As we mentioned earlier TCP and UDP sockets are the primary backbones of synchronous and asynchronous network systems. For bidirectional networks these factors of how data is transported out of the queue is incredibly important. As a brief overview let's look at what these socket types are.

UDP - UDP is a simple transport-layer protocol. The application writes a message to a UDP socket, which is then encapsulated in a UDP datagram, which is further encapsulated in an IP datagram, which is sent to the destination [7].

There is no guarantee that a UDP will reach the destination, that the order of the datagrams will be preserved across the network or that datagrams arrive only once.
The problem of UDP is its lack of reliability: if a datagram reaches its final destination but the checksum detects an error, or if the datagram is dropped in the network, it is not automatically retransmitted [7].
Each UDP datagram is characterized by a length. The length of a datagram is passed to the receiving application along with the data.
No connection is established between the client and the server and, for this reason, we say that UDP provides a connectionless service [7]. TCP is a connection-oriented protocol, which means a connection is established and maintained until the application programs at each end have finished exchanging messages. It determines how to break application data into packets that networks can deliver, sends packets to and accepts packets from the network layer, manages flow control, and—because it is meant to provide error-free data transmission—handles retransmission of dropped or garbled packets as well as acknowledgement of all packets that arrive [8].

For example, when a Web server sends an HTML file to a client, it uses the HTTP protocol to do so. The HTTP program layer asks the TCP layer to set up the connection and send the file. The TCP stack divides the file into packets, numbers them and then forwards them individually to the IP layer for delivery. Although each packet in the transmission will have the same source and destination IP addresses, packets may be sent along multiple routes. The TCP program layer in the client computer waits until all of the packets have arrived, then acknowledges those it receives and asks for the retransmission on any it does not (based on missing packet numbers), then assembles them

into a file and delivers the file to the receiving application [8].

With this information in hand, the bidirectional network has to setup two distinct protocols for how to handle TCP and UDP based socketing. To ease into this topic, we will start with the UDP side first.

Bidirectional UDP protocol is a very simple and straightforward handling of the system. Since the data is encapsulated with multiple layers of datagram with the IP destination datagram on top, we only have to worry about datagram package size and how it will be held on each device. The normal metadata is generated and is used to push the packet set into the queue may it be standard or priority queue. As the datagram is popped off the queue and sent to the next device it stays in its abstracted datagram wrapper until it reaches its host destination and is then de-wrapped and used.

The problem with this is if all packets do not arrive then it is nigh impossible to recreate or restore the full message without making the request again. Because a bidirectional system inherently has multiple stops and redirections to make it allows for a higher margin of error.

The bidirectional TCP protocol is a bit trickier in how it handles package acceptance and delivery. Because the system has the three-way handshake ideology of sending, delivering, and verifying we have to incorporate into each destination device a return notification of what packets were lost and the verification of delivered packets. This can work on a standard queue but may be incredibly slow as the return package has to wait through all other packets in order of all other device queues. The upside of this is it works catalytically if the device receiving the TCP based information is registered with priority and a high signature value. This allows for quicker movement through the queues and eventually receiving the necessary information.

IX. Downsides of the Bidirectional Network

Despite the large boons of the bidirectional network there are downsides to using the system. The primary detriment of the system is speed. Moving high volumes of data even with Ethernet network standards across a vast mesh network of devices can be tedious and genuinely downright inefficient. Then network is limited by the speed at which the data can move.

Another major issue is how redirection of moveable devices is handled. In theory, portable and IoT devices, should be trunk based devices but ends of branches of solitary and stationary devices. If an outside connection is present, as such in smartphones, these devices could possibly act as less secure nodes for sending and receiving data into a network. Careful planning is needed by system engineers to implement larger scale bidirectional networks as portable devices can cause major problems.

Lastly, if a device is pre-infected before it enters the safety net of the bidirectional network, all of the systems could be easily compromised by a simple worm. Standard security practices are recommended but a solid addition may be multiple security node devices that act as trunks of the bidirectional network.

X. Side Notes

A few last minute pieces of information need to be addressed. First things first, a bidirectional network is a mesh network topology able to transmit data from one device to another, store the data in an organized manner, resend it so it can be interpreted in its original form, and be read by the intended recipient. The talking points in this paper regarding bidirectional systems are intended to be helpful tips from a higher level of abstraction to assist in implementation. These systems can be implemented in whatever way needed to solve the problem in hand with whatever creative ways are needed. The inclusion of list integration, emergency route switching, queue architecture, TCP and UDP specific protocols can all be handled in greater or lesser detail than what is explained here. By no means is it a definitive guide on how to handle the system, just a singular take on a proverbial network data structure.

Second, bidirectional system offers an axiomatic solution to a number of IoT device boom based problems with preserving bandwidth and router connection slots via the branch and trunk system. Having the devices act as a mesh network is only as viable as the speed and accuracy of the data moved inside the network. Testing and work should preferably be done in an environment with solid network infrastructure already in place and systems designed to check the bidirectional network for accuracy and data integrity. As more advanced mesh network topologies are explored, using the device as a primary routing node will be an important sub system of how our networks connect.

Finally, bidirectional networks aid a lot of the issues in device based cyber security problems. Transport encryption, some privacy concerns, security configurability, and poor physical security are all aspects touched by the bidirectional system in one way or the other. Having devices that are hot swappable for how network traffic is delivered is a revolutionary tool to cut down on cyber security based issues for all size topologies.

XI. References

[1] billions), I. (2018). *IoT: number of connected devices worldwide 2012-2025 | Statista*. [online] Statista. Available at: https://www.statista.com/statistics/471264/iot-number-of-connected-devices-worldwide/ [Accessed 30 Nov. 2018].

[2] Actiontec.com. (2018). *Wireless Network Capacity…How Many Devices Can Connect to my WiFi Network? - Actiontec.com*. [online] Available at: https://www.actiontec.com/wifihelp/wireless-network-capacityhow-many-devices-can-connect-wifi-network/ [Accessed 30 Nov. 2018].

[3] Countermeasures, T. (2018). *The Internet of Things (IoT) – Threats and Countermeasures*. [online] Cso.com.au. Available at: https://www.cso.com.au/article/575407/internet-things-iot-threats-countermeasures/ [Accessed 30 Nov. 2018].

[4] Beej.us. (2018). [online] Available at: http://beej.us/guide/bgnet/pdf/bgnet_A4.pdf [Accessed 30 Nov. 2018].

[5] Grimes, R. (2018). *6 Network Protocol Analyzers*. [online] IT Pro. Available at: https://www.itprotoday.com/compute-engines/6-network-protocol-analyzers [Accessed 30 Nov. 2018].

[6] (DNA), D. and Security, E. (2018). *Enterprise Network Security - Encrypted Traffic Analytics (ETA)*. [online] Cisco. Available at: https://www.cisco.com/c/en/us/solutions/enterprise-networks/enterprise-network-security/eta.html#~stickynav=1 [Accessed 30 Nov. 2018].

[7] Holm Security. (2018). *What is the difference between TCP and UDP?*. [online] Available at: https://support.holmsecurity.com/hc/en-us/articles/212963869-What-is-the-difference-between-TCP-and-UDP- [Accessed 30 Nov. 2018].

[8] SearchNetworking. (2018). *What is TCP (Transmission Control Protocol)? - Definition from WhatIs.com*. [online] Available at: https://searchnetworking.techtarget.com/definition/TCP [Accessed 30 Nov. 2018].

[9] Shekhar, A. (2018). *Wi-Fi vs Ethernet: Which Internet Connection Is Better And Why?*. [online] Fossbytes. Available at: https://fossbytes.com/wi-fi-vs-ethernet-comparison-features/ [Accessed 30 Nov. 2018].

[10] Lendino, J. (2018). *What is 802.11ac Wi-Fi, and how much faster than 802.11n is it? - ExtremeTech*. [online] ExtremeTech. Available at: https://www.extremetech.com/computing/160837-what-is-802-11ac-and-how-much-faster-than-802-11n-is-it [Accessed 30 Nov. 2018].