11-2018

# Using Low Power 5G IoT Devices Off-Network Utilizing Modified ProSe Communication

John Aaron Swarner
*University of Nebraska-Lincoln*, john.swarner@gmail.com

# Using Low Power 5G IoT Devices Off-Network Utilizing Modified ProSe Communication

by

John A Swarner

A THESIS

Presented to the Faculty of

The Graduate College at the University of Nebraska

In Partial Fulfillment of Requirements

For the Degree of Master of Science

Major: Telecommunications Engineering

Under the Supervision of Professor Hamid Sharif-Kashani

Lincoln, Nebraska

November 2018

# USING LOW POWER 5G IoT DEVICES OFF-NETWORK UTILIZING MODIFIED ProSE COMMUNICATION

John Aaron Swarner, M. S.

University of Nebraska, 2018

Advisor: Hamid Sharif-Kashani

**In setups to enable IoT devices, high power radios are required in each of the devices separately. In order to reduce the power requirements for each individual device, a lower power radio could be used in order to reduce the total power requirements and size for the device and increase battery life or reduce power draws for each device. To enable this, a small change removing the application server requirements would be required. A mesh network could be used to enable communication to the main network, either to a tower that is within the range or through a larger relay device. ProSe D2D standards allow for connectivity from one device to another to eventually reach the serving network.**

**This research discuss and analyzes ways for those networks to form, detect thresholds, and simulate load on a network in a standard use-case scenario. The simulation shows that the mesh network concept is performs well and can work within existing technologies.**

# I. Introduction

THIS research will go through a short background on the requirements of ProSe (D2D) connections between devices, the power and message flow requirements, separation of the Application Server (AS) requirements from those messages, and then setup of mesh style networks in places where serving networks are absent. We'll then look at the power requirements of these connections and how allowing ad-hoc 5G ProSe networks allows for lower power requirements of devices. We'll then explore how these lower power requirement can effect designs of the devices themselves and their uplinks to the internet.

While we don't think of most homes or businesses having issues with network connectivity, in mm-wave setups, basements and larger buildings may not have network connectivity to a 5G tower. In these cases, internet-of-things (IoT) setups should be allowed to setup in semi-licensed or unlicensed spectrum without authorization from the serving network. This way, the devices can establish a relay from nodes that do have serving network access from the mesh network to a serving network for internet access via a larger power or less obstructed node. To facilitate this, in some cases, the ProSe requirement of an AS must be releaxed so that if the devices are unable to connect to the serving network, they can still establish a D2D link.

To facilitate the ad-hoc network setup, we'll explore how the devices discover each other, channel scanning and selection, power requirements, and nomination/selection of "uplink" nodes back to the serving network, either via 5G cellular network, wifi, or wired connection. Then we'll move through several add and delete node scenarios to show how the network reacts to new nodes arriving and leaving the network space.

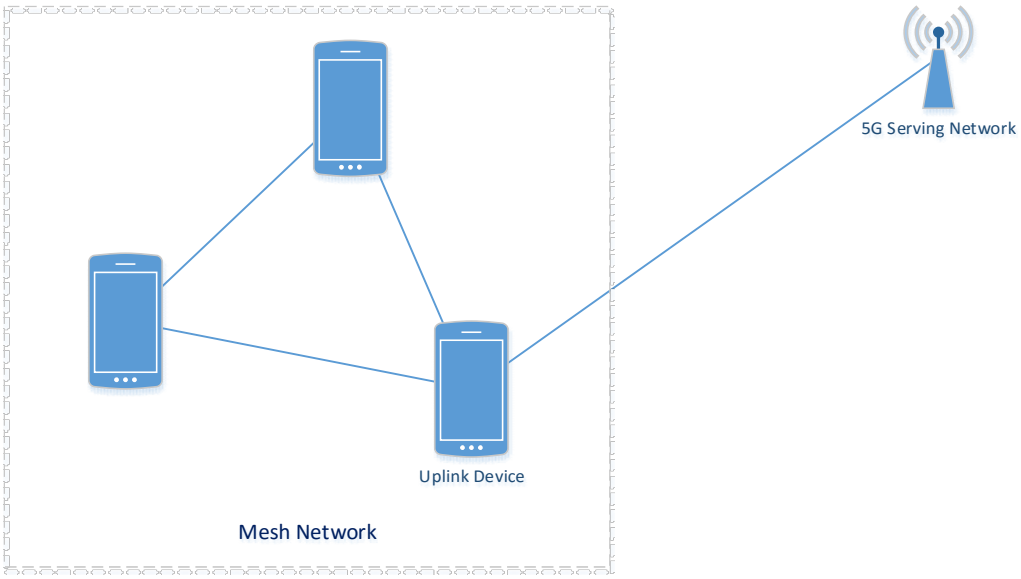



Figure 1: Example Device to Device Communication Mesh Setup and Uplink Nomination

## II. Background

### *A. 5G Network Topology*

In 5G network topologies, we are seeing a changing dynamic from the traditional cellular approach of every device attaching to the serving network.[10] While we saw a trend of this in 3G and 4G technologies with the introduction of pico-cells and femto-cells, 5G emphasizes this even more due to the introduction of the mm-wave frequencies for additional channel density and throughput requirements. Since power drop off, scattering, and reflection for these frequencies is much more problematic as distance increases, more of these types of relays are required in order to serve the same space as a lower frequency channel.

In addition, the introduction of the D2D standard in the LTE topology will be carried forward into the 5G space as well. This allows devices in the network to act as relays toward the network in the cases of certain types of data traffic.

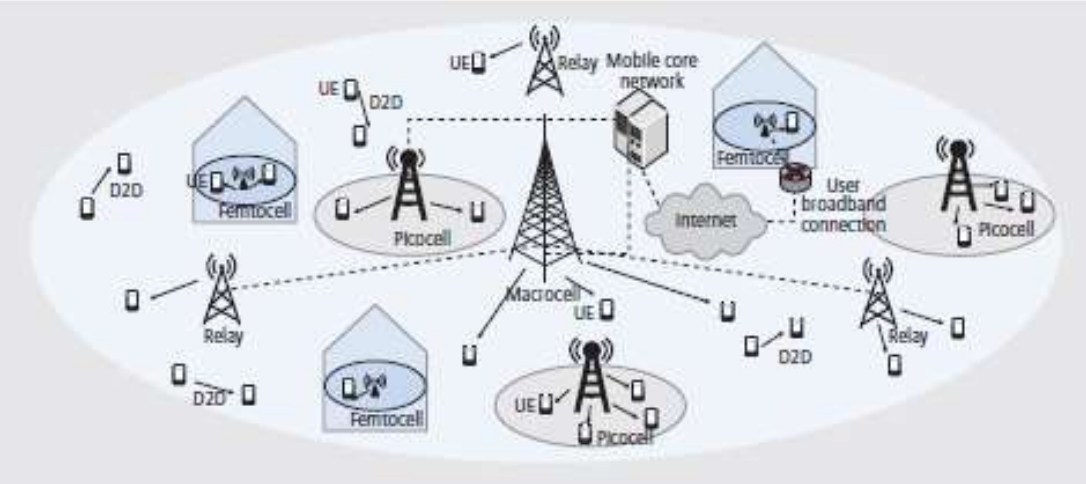



Figure 2: Multi-Tier Network setup involving all types of links in a proposed 5G setup.[10]

Driving this topology is a network designed for much higher throughput and lower latency than legacy cellular networks, allowing in urban areas an average data rate of 300 Mb/s down and 60 Mb/s up, with peaks close to over 1 Gb/s[10]. Latency is expected to be around 2-5 ms on average, which is much lower than the 60-70ms traditional 4G/LTE networks are achieving today[11].

One of the design tenets of the 5G network was developing a transport that would work will with Machine-Type Communication (MTC) devices. MTC devices are things like vehicle sensors, home appliances, security systems, or sensor networks. The throughput and latency requirements of these devices are varied, as well as placement and radio size/strength. These types of devices are now what we consider the internet of things (IoT).

### *B. Internet of Things in 5G Networks*

In a 5G network space, enabling mass IoT deployments is one of the primary focuses. In Palattella et al [1], it's discussed that to enable more IoT connections, we must establish a D2D standard that allows IoT devices to establish connections directly to the serving network, act as relay's to and from the serving network to extend the serving networks range, and setup connections between themselves without prompting from the serving network.

Facilitating these connections will help optimize the network and assist the operator in serving all the IoT devices in the coverage area. The challenges from this mostly arise from the fact that some of 5G is in licensed spectrum, so variances must be created for the latter case of direct D2D connections without serving network oversight. We can facilitate this by a market agreement that ad-hoc D2D communication happen in the semi licensed spectrum space, where only reporting use is required, and not use that for standard radio access. Since we're looking at a specific use case, for the course of this research we'll be using the 64-71GHz bands to handle these communications.

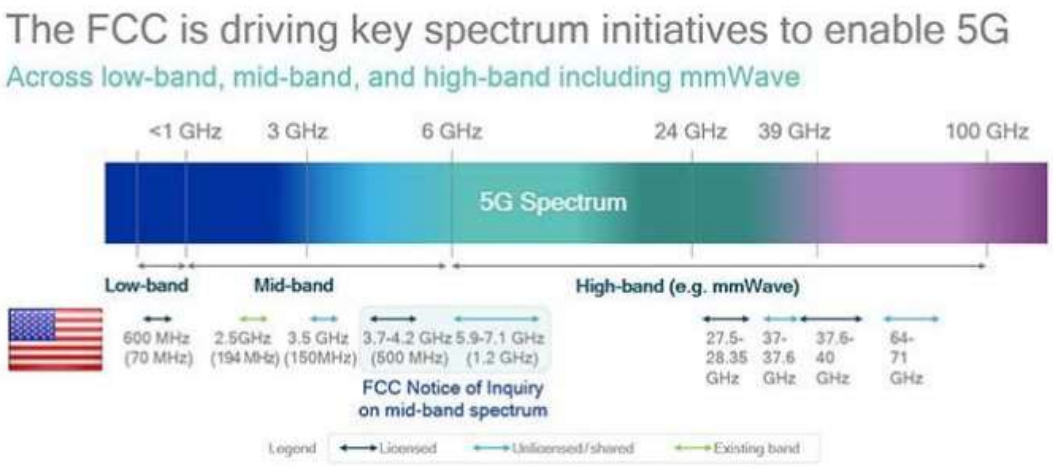



Figure 3: Spectrum Distribution of 5G Networks [9]

*C. ProSe and Required Changes*

The first item we must address to facilitate these changes is the removal of the service network control and AS requirement from the standard. While these do serve some purpose in 4G ProSe setups due to the entire setup being in licensed bandwidth and involving user devices, in 5G this is less of a problem, because we're now dealing with either semi-licensed or unlicensed spectrum and in this case we're facilitating relays for IoT devices and not user endpoints. As long as the frequency is in the unlicensed or semi-licensed spectrum, devices just need to adhere to a report requirement, so that other devices and services know that the frequency could be in use in a specific area. Also, since we're not gating based on specific applications and the need to

push phones into allowing D2D connections by the serving network, the AS requirement can be removed.

With the broad allowed range for 5G communications, radios in IoT devices need to be flexible, so that they can interact with all possible connection types of 5G networks. This flexibility allows for us to easily form IoT meshes outside of network coverage, as the radios should be able to find a small rage that is unused in almost any coverage area.

The issue with the current ProSe standard is that it was originally conceived to work in 4G licensed spectrum, which doesn't allow for D2D communication setup outside of network coverage. While it is talked about in the original write up of the discussions of enabling it, the fact that devices would talk to each other via licensed spectrum without the serving network controlling it violates a large number of FCC rules. In the 5G network setup, this is less of a problem, but still a possible issue if the frequency spectrum that the devices use falls within certain frequency ranges. Therefor the IoT manufacturers must have the Ad-Hoc mesh setup fall within certain frequency ranges that are reported to the FCC for such or in unlicensed shared spectrum. If all manufacturers do not use such ranges, the ad-hoc network setup will fail or be subject to possible FCC files due to non-compliant frequency usage.

As laid out in Pilloni et al. [2], IoT clustering will build on the existing ProSe requirements of an Mobile Management Entity (MME) and AS, but in the case of the 5G IoT using unlicensed spectrum, we can put the devices into permanent discovery mode and instead of having to have the devices be activated based on application starting the connection, automatically allowing setup of mesh networks for 5G devices and flow of data to and from the serving network. With the trend of serving networks away from application centric models and more toward transport or service models, the removal of the requirement of the AS reduces the complexity of the serving network and moves more functionality down to the devices.

This does increase the radio complexity, but this is usually only on a change to the network. In steady state, there is an additional relay function that needs to be facilitated, but with some rudimentary routing functionality on the system on a chip (SoC) we can easily facilitate setup of multiple channels and transport of the data to the nominated uplink.

While there are some cases where the data network will change drastically over time, in most cases we're talking about static setups with small numbers of node add/drops over time. We'll go over a few scenarios to

show this:

*1) Train/Plane/Truck/Car IoT Setup*

In transport configurations, there will be a mix of IoT devices and user devices. There will be a large number of sensor endpoints and cameras with a number of user and controller devices. In these cases the mesh network can be anchored through the non-user endpoints in order to achieve network stability in case of user endpoints being added or removing themselves from the network. Dedicated user relay access point to the mesh network can be utilized in order to not require the sensor or camera IoT's to have large processor or antenna/radio requirements. This allows the low compute and radio power devices that would normally constitute an IoT device to achieve efficiencies that would not be available if every device needed to reach the serving network. If every device could even reach it due to LOS/Power Loss requirements.

*2) Factory/Office*

In factory or office configurations, there will be a mix of IoT devices and user devices. Again, the large number of the devices will be sensors and cameras, but there will be a significant number of user devices such as phones/tablets/computers as people either work in the factory floor or in the office. The majority of the network can be established via the sensor and camera network to minimize impact as people move in and out of the mesh network.

*3) Home*

In home configurations, there will be a much higher ratio of user devices to IoT devices than in the other scenarios. Still, a large number of the devices will be "smart" devices (lights, thermostats, doorbells, etc), and a smaller number of devices will be sensors or video, but with large power sources (TV/Monitors, Dishwasher, Refrigerator, Oven, Washer/Dryer, etc). These large power devices can be used to establish the connection to allow the sensor network to operate in a lower power mode in order to prolong the life of the radios in those devices and battery life. The mesh network then has anchors in order to minimize disruption of the network when user devices join and leave the network.

Now that we have the building blocks for an IoT mesh network, building a practical network setup in order to have devices actually communicate to the serving network is a requirement. Selecting a frequency range for general D2D IoT communication that is outside the licensed spectrum, understanding what the limitations of that frequency ranges are, and then building some practical simulations in order to test if the hypothesis is sound will bring to light future questions to answer or prove that mesh networks can be easily developed. Also, looking at some papers on this, such as "Modeling D2D Handover Management in 5G Cellular Networks"[8], a majority of the papers done to date are around 2's or 3's and don't consider larger mesh networks, so we'll be working on much larger configurations.

First we need to understand what frequency ranges are allowed to be used. While other papers have talked about the available ranges, no one has proposed a frequency range to be targeted. If we can use a frequency range that is outside the licensed spectrum range, we can allow these devices to auto-setup and develop routing solutions that are relatively optimized depending on node to node connectivity.

Second, we need to develop the methodology of building the network. While some people have considered small numbers of devices when talking about D2D communications, I have not seen any indication that larger device meshes have been considered. Considering that in the next generation of radio access we're expecting reaching the serving network to be more difficult for radio access for a majority of the network spaces, we should consider a large sensor network or security system up-linking through a small number of devices compared to the larger mesh.

Third, determining the mesh setup and routing methods through the fabric to and from the serving network needs to be determined. If we can investigate the network setup, we can either validate that some will work or determine which ones will not and allow for future study of other routing processes.

## IV. MESH NETWORK SIMULATION

### *A. Frequency Ranges for IoT Setup*

To facilitate the setup of the mesh network, we must either allow the IoT devices to use licensed bandwidth or designate some portion of the semi-licensed spectrum for inter-IoT device setup. We are proposing to use the upper end of the designated 5G spectrum[9], which is not practical for serving network access, for

connections between IoT devices. Since the range in most relay applications isn't very high, the high10 power loss over distance of the 64-71GHz band is not an insurmountable problem as it is for Serving Network access. If we estimate 100MHz per connection, we can enable a 70 channel setup in the 7 GHz of bandwidth, which should be enough for a large majority of device layouts and enable a large number of high throughput video devices. Depending on the throughput requirements of each one of those devices, the channel size could be reduced greatly down to the standard 20 MHz, allowing for a number of additional channels.

The high power loss over distance also reduces the problem of co-channel interference in the case of mesh networks being near each other but not close enough to develop connections. Some channel scanning would need to be done in order to see what's in use and what is free, but items like that are already done in WiFi network for 802.11ac setups and is a well understood problem in much less available bandwidth applications.

When a device is installed and powered on, the first thing it will do is scan the surrounding frequency spaces and then select a channel that is not currently used. These channel use/timeslot setups are all currently defined in the 3GPP 5G standards and do not need to be re-invented for this, just used between the network spaces and without the serving network controlling it.

### *B. Power Model Selection and formula*

Using standard power loss calculations, we can calculate the power loss at 1 meter, which is about 45 dB (15 dBm) for the 64 to 71 GHz. We use the following formula to determine that:

$$\mathrm{PL}\,(\mathrm{d0} = 1\mathrm{m}) = -10 * \log\left(\frac{\mathrm{Gt} * \mathrm{Gr} * \lambda}{(4 * \pi)^2 * 1^2}\right)$$

We can just assume 1 for the gain on the receiver and transmitter to establish baselines. Running these calculations we see that the difference between the frequencies at 64 and 71 GHz is very small (45.27 dB vs 45.73 dB). We then can use the indoor power model to calculate loss between devices. We will use two calculation models in this simulation. One will be for if two devices are in similar elevations, and the other will be used if the devices are on different floors. For the inter-floor model, we'll assume there are 2 walls (5 dB loss on each[4]) between the devices and a mean loss exponent of 5.04. We'll assume transmit power of 23 dBm [3].

$$PL(d) = 15.27 + 10 * 5.04 * log(d) + 2 * 5 - 23$$

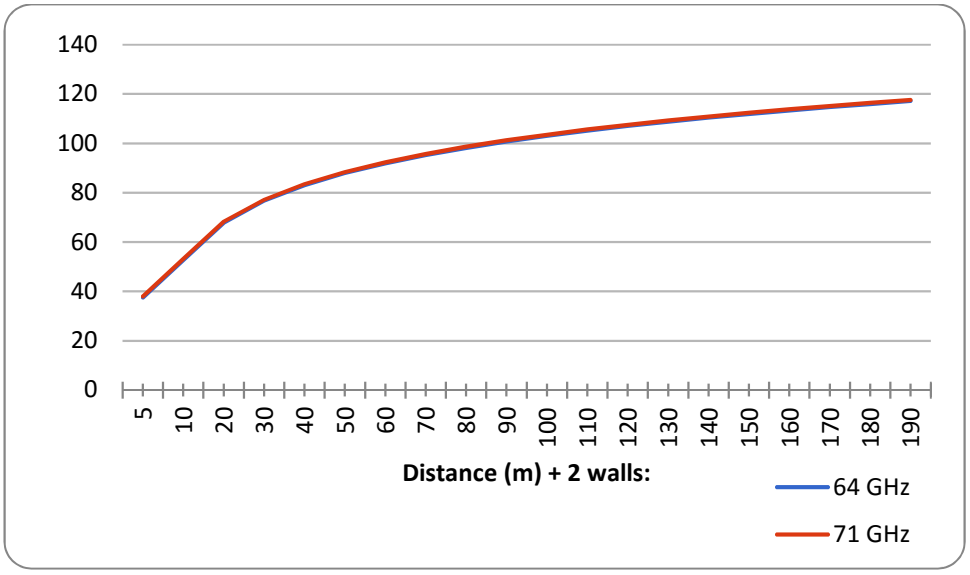



Figure 4: Power Loss between Devices on Different Floors

For devices on the same floor, we'll assume a mean loss exponent of 2.76 and the same dB loss for 2 walls.

$$PL(d) = 15.27 + 10 * 2.76 * log(d) + 2 * 5 - 23$$

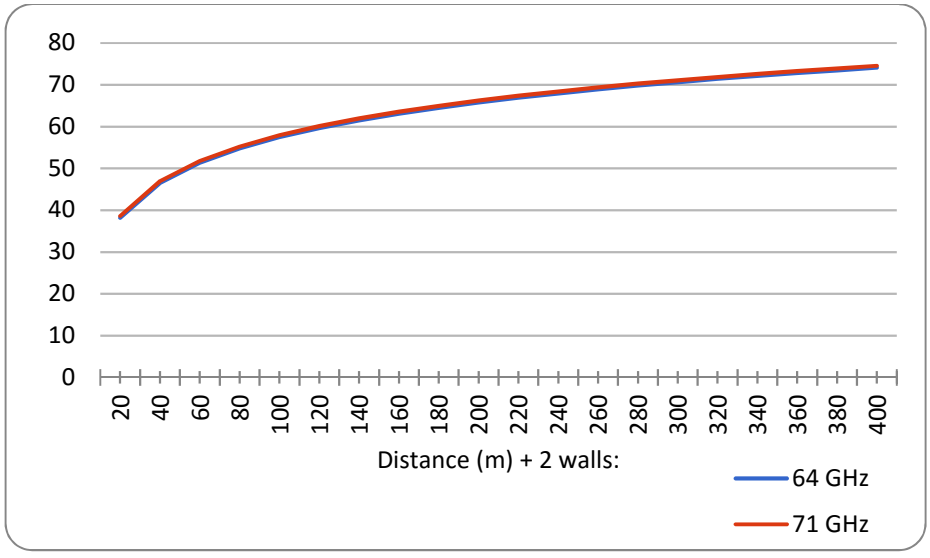



Figure 5: Power Loss Between Devices on the Same Floor

The network framework will be created in Matlab and then we can use functions to step through the configurations and make determinations around power requirements and connections. After creating the nodes in random positions, if the distance PL between devices is less than 115 dBm, the simulation assumes that the devices can talk to each other. In most applications, this means that all devices on the same floor can communicate with all other devices, but only "nearby" devices between floors can communicate. In in the case of this simulation, a 2 story, 100m by 100m building was randomly seeded with devices. On the lower floor, it was assumed that none of the devices would be able to talk to the serving network, simulating a basement or metal building lower floor with no windows. On the upper floor, a 25 % chance was given to allow for a serving network connection, simulating either being close to an outer wall, having LOS to a serving network site, or having some sort of wifi bridge or hardwire connection in the case of larger multi-purpose nodes.

```
N = Number of Nodes in the D2D Mesh Network

for N do
    define Node (Name, Floor, Position, Node Type)
    if Node Foor == 1 do
        Has Uplink = false
    else do
        Has Uplink = 50% of the time
    end if
end for
```

Figure 6: Code for Node Random Generation

We then generated a set number of nodes with random position and floor location. At lower node counts (less than 8), the number of devices and power requirements led to a lot of mesh networks that were not able to serve all devices. At 8 nodes, well over 50% of the time a network is randomly generated it is viable. That doesn't mean that lower node setups are invalid, but that careful planning would be required in order for them to be viable.

As we up the node counts to generate larger networks, once over 10 nodes almost 100% of the networks were viable. We then tested setups of those networks to determine routing and throughput requirements.

When building a 5G IoT network, the intention isn't so much to have the nodes sending application data to each other (except in some very specific sensor setups), but instead sending data to application servers on the internet for processing or sending or receiving video or audio streams from the internet to or from the end devices. This is why the original application server requirement can be relaxed.

To facilitate this communication, some sort of routing protocol must be used. In the simulation, since all nodes can talk to all other nodes on the same floor, but nodes in between floors can only talk to each other in very specific distances, the node connection parameters were compared in order to determine which nodes could talk to different floors.

```
N = Number of Nodes in the D2D Mesh Network

for N do
    Initialize Node RouteMaps
    Initialize Node Throughput Rates
end

%Now we'll create the RouteMap
for i=N do
    for j=N do
        if Node(i) is connected to Node(j) do
            Add to Total Node Connected Count
        end if
    end for
    if Node(i) is on floor 1 do
        if # connected nodes > # of nodes/floor do
            Set Node as Uplink Node
        end if
    else do
        if # connected nodes > # of nodes/floor do
            Set Node as Uplink Node
        end if
        if Node(i) has SN Uplink do
            SNUplinkNode = NetObjs(i).NodeName
        end if
    end if
end for
```

Figure 7: Code for Uplink Node Determinations

Once those are determined, we use a function to walk through the connected node variables in order to fill out a route map for each node. This route map will be used when we determine what the bandwidth requirements are for each of the connections between the nodes in order to serve its own traffic and any traffic through the node to nodes that that node is servicing.

### *E. Bit Rate Calculations and Throughput Requirements*

Now that we have a route map via uplink nodes and connections to the serving network, we can start to calculate the bit rate requirements for the inter-node communications. In an average 8 node setup, each uplink node will talk to 4 other nodes (if there is a 50/50 split on floors). The first floor uplink node will talk to the 3 other nodes on the same floor and the node it can communicate to on floor two, and the second floor node will talk to other nodes on floor two and the node it can talk to on floor one. The serving network uplink node(s) will talk to other nodes on the same floor and have all traffic to and from the serving network relayed through it. The simulation code walks through the connected node maps and adds the first or second floor uplink nodes for the nodes not populated. This allows each node object to have a "next hop" for all nodes in the network.

After the route map creation, we can then calculate the data rate requirements for each node with randomly generated network objects. In the code snipped shown in figure 6, we set a random variable to determine what kind of object each network node is and what kind of data it would either be sending or receiving. In the simulation code, we walk through each node and use that variable to set the bit rate generated by that node to and from the mesh network.

Now that we have determined what the bit rates are to and from the serving network for each one of the devices, we can create the total traffic pattern for the entire mesh network, including the rate to and from the serving network. In the simulations, we walk through the network nodes to and from the serving network. This will give us the total bit rate in and out of every node in the serving network.

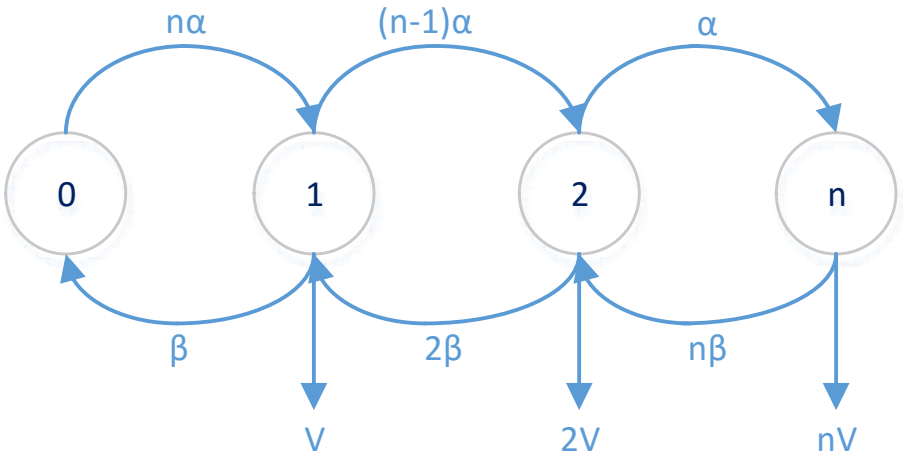
Then, we'll determine the queuing model for the nodes to determine how a larger number of nodes affect transmit/receive capabilities and service time.

### *F. Queuing Model and System Time Calculation*

Now we want to understand what pushing this traffic through the network does to each node's transmission requirements. We'll model this via Fluid Source Modeling of a bursty traffic model.

First, we need to assume some variables. We'll assume that the on time ($\alpha$) for this traffic is 2 seconds with an off time ($\beta$) of 0.5 seconds. The transmission size per state will be the total traffic relayed through

the node divided by the number of nodes on that the relay node is serving. That gives us a Markov diagram like below:




Markov State Diagram for Fluid Traffic Analysis

Now that we've defined the state diagram, we can develop the equations to determine queue time and possible packet loss. If we're modeling this like a video source, we can using 20 minisources and the following equation:

$$P = \frac{\alpha}{\alpha + \beta} \qquad \rho = \frac{Rp}{C} = \frac{20\text{PA}}{\text{C}}$$

$$r = (1 - \rho) * \left(1 + \frac{\alpha}{\beta}\right) * (1 - \frac{C}{MA})$$

$$G(x) = Am * \rho^{20} * e^{\beta * r * \frac{x}{Rp}}$$

We can generate the general queue and loss probability of data going through the system. Then we'll find out what a call admission does to the general packet throughputs. We'll use the approximate fluid flow analysis to determine this with the following equations:

$$k = \frac{\beta x}{Rp(1-P)\ln\left(\frac{1}{PL}\right)}$$

$$\frac{CL}{Rp * N} = \frac{1-k}{2} + \sqrt{((1\text{-}k)/2))\text{^}2+k*P)}$$

With the results from these equations, we can determine what additional latency would be introduced via the relay mechanism and how the modifying of each of these variables effect the total traffic. We can also determine how large the networks can get before this mesh setup will not work anymore.

## V. Simulation Results

### *A. Bitrate Requirements per Node*

The simulated network output can now be graphed by node and input/output requirements. We can quickly see that there will be no issues with actual throughput of any of the devices on a per channel basis.

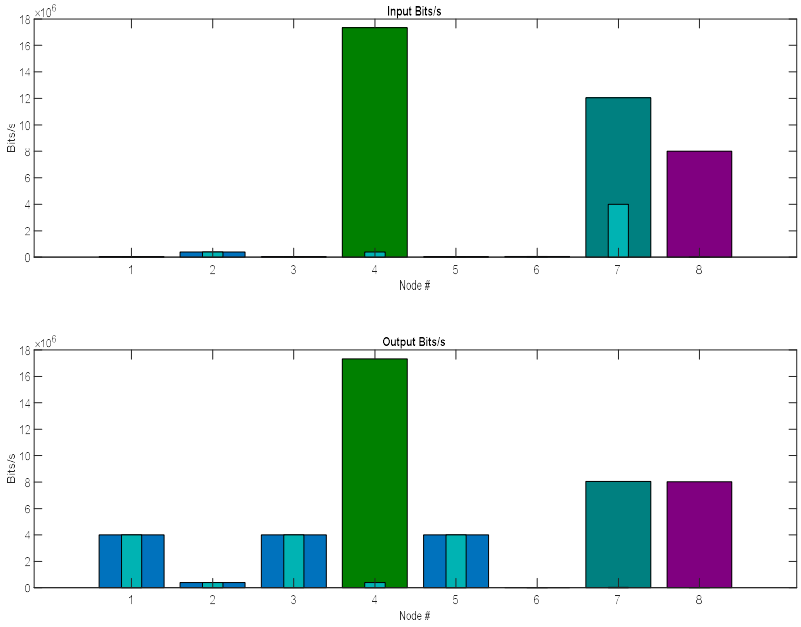



Figure 9: 8 Node Bit Rate Input/Output

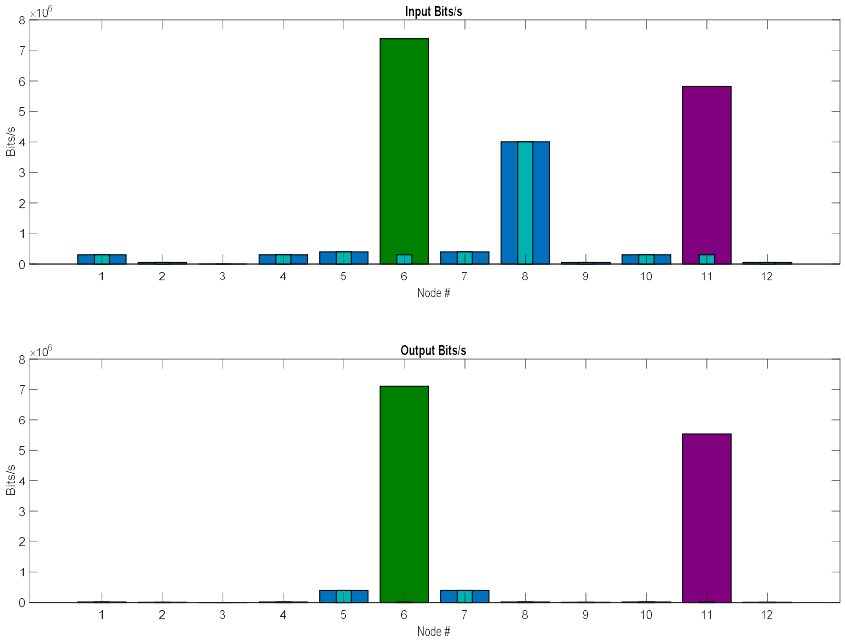



Figure 10: 12 Node Bit Rate Input/Output

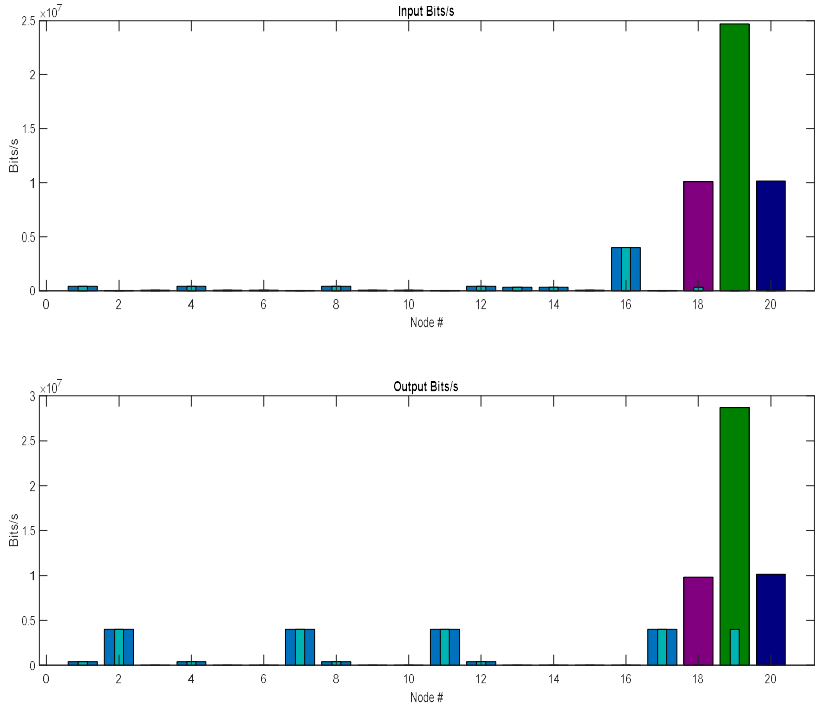



Figure 11: 20 Node Bit Rate Input/Output

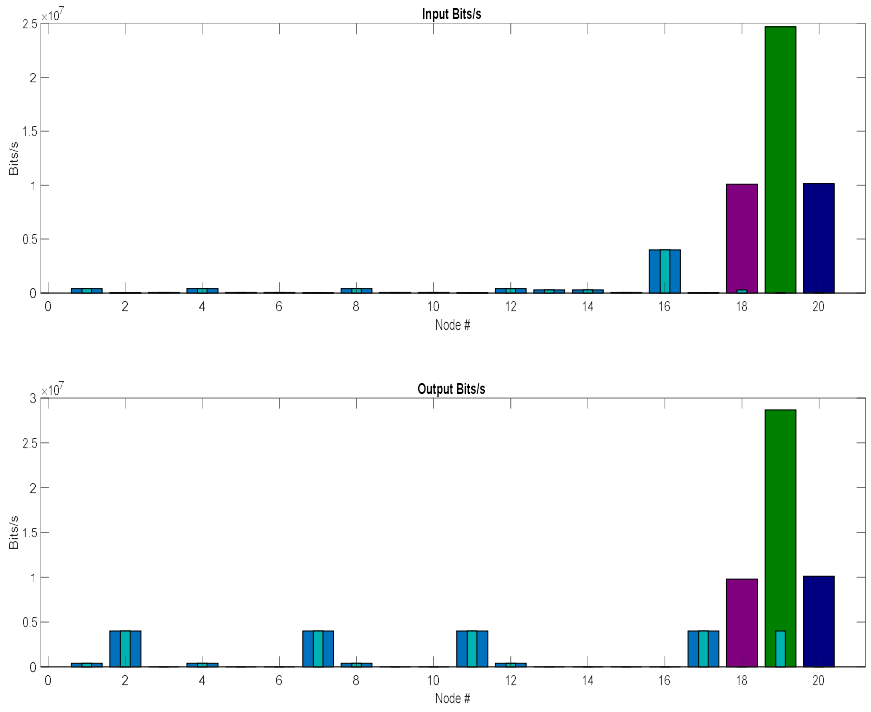



Figure 12: 25 Node Bit Rate Input/Output

In the above figures, the green nodes are the uplink nodes, and the purple nodes are the uplinks to first floor. What does become apparent that will be an issue is that without either very large processing or radio requirements on the devices, the fact that in larger configurations require more than 20 channels assigned for uplink and downlink between each devices. In the simulation, we find that in N node configurations require N channels between each of the uplink/relay nodes. In smaller node configurations like 8, this is manageable, as larger devices with larger and more complex radios will be able to handle this number of channels, but unless the network is semi-planned, this isn't always a guaranteed setup.

We then proposed looking at different routing methods that would allow for a smaller number of channels between each device. In larger node configuration this may be possible, but in the smaller configurations such the 8 node setup there is a much higher chance of only a couple of nodes being able to connect between floors. More research will have to be done into this either around routing mechanisms to reduce the number of channels per node or to develop some sort of "hub" node to enable a large number of

One thing that could resolve the channel usage is to just multiplex up a single channel set for each type of connection. That would leave us with one for all connections to each floor's uplink node, one for uplink between nodes, and one for the uplink to the main serving network. That leaves us a 1 gig fabric setup between all the nodes in each floor and the uplink node, 1 Gig uplink between the floors, and up to a 1 Gig link to the serving network, depending on serving network capabilities. This would still require some pretty complicated assignment setups from the serving nodes, but it could be handled as part of the node entry into the mesh network and by something that is much more reasonable than the N channel setups we mentioned above.

*B. Power Requirements for each*

The advantages of the configuration are readily apparent as well. The average power requirements of the channels setup between devices on the same floor is much lower than connections on the devices between floors or the connection between the uplink nodes and the serving network.

Looking at the received power for each node on the same floor, using the 20 node setup as an example:

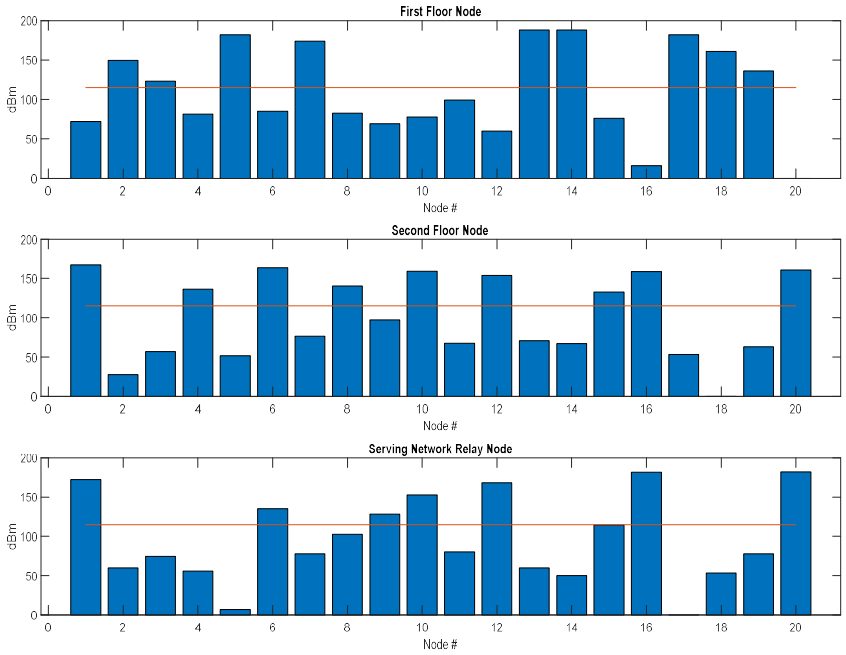


Figure 13: 20 Node Power Connection Levels

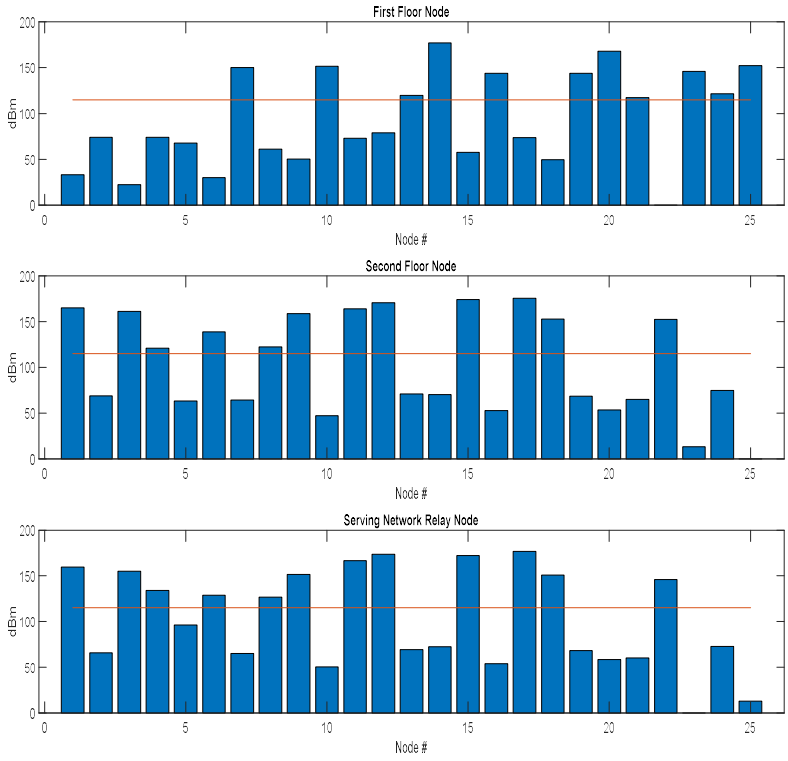



Figure 14: 25 Node Power Connection Levels

We can see that the power requirements for nodes on the same floor are well under the 115 dBm limit while the ones on different floors either are very close to the 115 dBm limit or are well above the 115 dBm detection threshold. If we go to something like a dedicated relay node (either what 4G would call a femtocell or some larger power node) with a planned layout, the secondary devices could just attach to those relay nodes and then to the serving network.

### *C. Additional Latency with Relay Functions*

Using the values that we assumed above, and using a transmission rate maximum of 1 Gbps, we can graph the following to see what effects these assumptions will make on the throughput requirements for the system. First we'll play with the buffer size to see what that does to the overall throughput requirements:

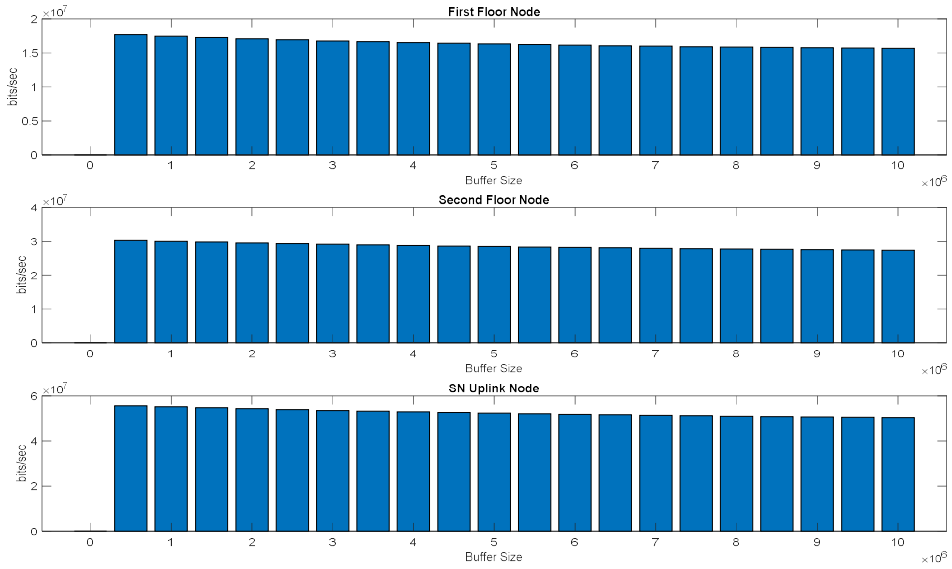



Figure 15: Throughput vs Buffer Size

As shown above, the larger the buffer makes the required throughput slightly smaller at the cost of a larger buffer wait time. Latency would increase by the buffer size divided by the transmission speed. At the largest size, it would still only increase the wait time by 10 ms per node relayed through. With the 4 megabit size buffer we used for the standard for other comparisons, it would only be increased by 4 ms per node.

I then spent some time modifying the other variables to see if changing them made any significant difference. These are shown in the appendix, but one interesting one was the following graph showing the total number of nodes that even with an order of magnitude more nodes we are well within the design specifications of the 5G network.

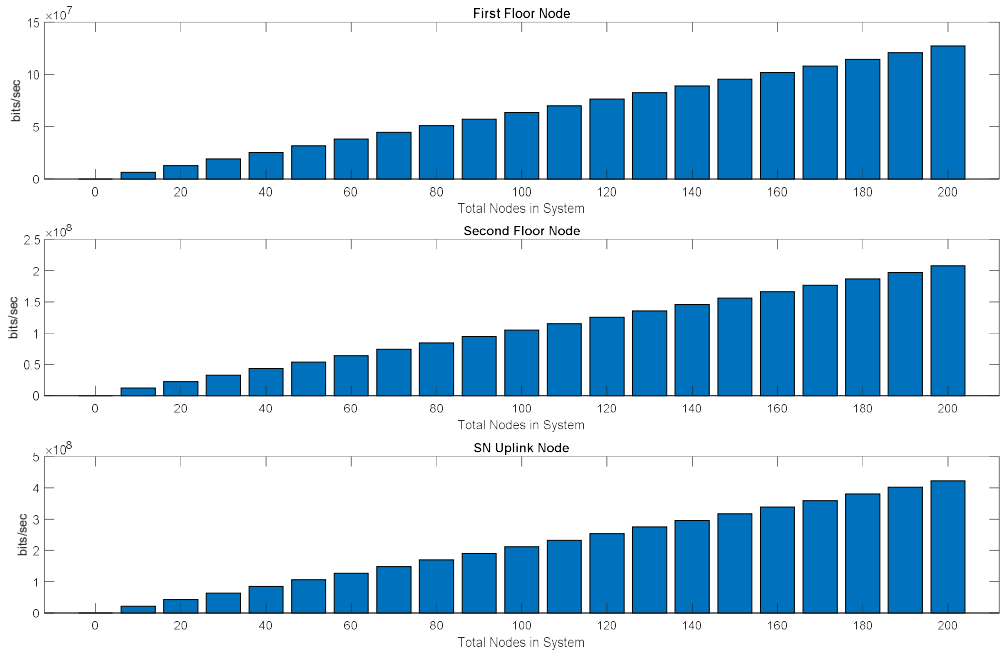



Figure 16: Throughput vs # of Nodes

*D. Nodes Entering or Leaving the Network*

The final point we wanted to look at was disruption of the network when nodes enter or leave the network. As long as the nodes leaving or joining the network are not any of the 3 targeted relay nodes that all the nodes communicate through, the amount of traffic will fluctuate, but the structure of the network will not be changed. If one of the relay nodes leaves, the network must be notified and the method needs to be re-run in order to update the routing tables to update the next-hop for all the nodes for traffic to either reach the appropriate node or the serving network.

There is a small chance, especially in smaller configurations that the new network structure will not be a valid mesh network, either because the nodes won't be able to talk between floors, or because all links to the serving network are removed.

As we have shown, a practical setup of mesh networks considering large numbers of nodes is relatively straightforward to simulate. Power requirements are lower and devices that could normally not get to the serving network now can. What we have found is that the D2D concept itself is sound for low numbers of nodes, but when establishing the mesh networks for larger node networks, more research is needed around the setup of the mesh and routing functions through it. We can use some sort of multiplexing to treat what we would consider a single 5G channel as a wifi like fabric instead of setting up individual channels for each connection. In addition, dedicated forwarding chipsets, such as application specific integrated circuits, (ASIC's) in the devices may be required to handle the data throughput required between the nodes. When these two research items are completed, we should be able to serve mesh networks in the simulated configuration that are upwards of 400+ nodes, with only the bit rate of the uplink to the network as the constraining factor.

Next steps of research would be verifying that this multiplexing is possible and generating a control scheme for multiplexing and routing in the network, similar to what is done in a Z-Wave or other mesh style network setups. Then when the data rate is determined using those methods, find what that load does to the SoC's commonly found in IoT devices, and if ASIC's are required for forwarding.

## REFERENCES


[1] Palattella, M. R, Dohler, M, Grieco, A, Rizzo, G, Torsner, J, Engel, T. Ladid, L; Internet of Things in the 5G Era: Enables, Architecture, and Business Models doi: 10.1109/JSAC.2016.2525418

[2] Virginia Polloni, Emad Adb-Elrahman, Makhlouf Hadji, Luigi Atzori, Hossam Afifi; IoT_ProSe: Exploiting 3GPP services for task allocation on the Internet of Things: Ad Hoc Networks 66 (2017) 26–39

[3] Athul Prasad, Andreas Kunz, Genadi Velev, Konstantinos, JaeSeung Song: Energy-Efficient D2D Discovery For Proximity Services in 3GPP LTE-Advanced Networks – DOI: 10.1109/MVT.2014.2360652

[4] Rappaport, Theodore S, Wireless Communications: Principles and Practice ISBN 0-13-042232-0

[5] Hossain, E, Hasan, M. 5G Cellular: Key Enabling Technologies and Research Challenges – IEEE Instrumentation and measurement Magazine DOI: https://doi.org/10.1016/j.adhoc.2017.08.006

[6] Tehrani, M.N, Uysal, M. Yanikomeroglu, H.; Device to Device Communication in 5G Cellular Networks: Challenges, Solutions, and Future Directions. IEEE Communications Magazines, May 2014.

[7] Lin, X., Andrews, J.G., Ghosh, A. Ratasuk, R.; An Overview of 3GPP Device-to-Device Proximity Services, IEEE Communications Magazine, April 2014

[8] Ouali, K. et al; Modeling D2D Handover Management in 5G Cellular Networks, DOI: 78-1-5090-4372-9/17

[9] https://www.qualcomm.com/news/onq/2017/10/04/path-opening-more-spectrum-5g-us

[10] Hossain, E. Rasti, M, Tasassum, H, Adbelnasser, A; Evolution Toward 5G Multi-Tier Cellular Wireless Networks: An Interference Management Perspective; IEEE Wireless Communications – June 2014

[11] https://www.statista.com/statistics/818205/4g-and-3g-network-latency-in-the-united-states-2017-by-provider/

[12] Maglaris, B, Anastassiou, D, Sen, P, Karlsson, G, Robbins, J, Performance Models of Statistical Multiplexing in Packet Video Communications, IEEE Transactions on Communications, 1988 DOI: 10.1109/26.2812

[13] D. Anick, et al. "Stochastic Theory of a Data Handling System with Multiple Sources", Bell System tech. J Vol. 61 No 8, pp 1871-1894, 1982

./MeshNetSimulation.m



```
clear;
clc;

alpha(1)='A';
alpha(2)='B';
alpha(3)='C';
alpha(4)='D';
alpha(5)='E';
alpha(6)='F';
alpha(7)='G';
alpha(8)='H';
alpha(9)='I';
alpha(10)='J';
alpha(11)='K';
alpha(12)='L';
alpha(13)='M';
alpha(14)='N';
alpha(15)='O';
alpha(16)='P';
alpha(17)='Q';
alpha(18)='R';
alpha(19)='S';
alpha(20)='T';
alpha(21)='U';
alpha(22)='V';
alpha(23)='W';
alpha(24)='X';
alpha(25)='Y';
alpha(26)='Z';

NumNodes=25;

%Generate Nodes
for i=1:NumNodes
    NetObjs(i).NodeName = alpha(i);
    NetObjs(i).NodeNum = i;
    NetObjs(i).FloorNum=round(rand(1))+1;
    NetObjs(i).FloorX=round(100*rand(1));
    NetObjs(i).FloorY=round(100*rand(1));
    NetObjs(i).ConnectedNode='';
    NetObjs(i).ObjType=round(4*rand(1)+1);
    if NetObjs(i).FloorNum == 1
        NetObjs(i).HasUplink = false;
    else
        if rand(1) < 0.25
            NetObjs(i).HasUplink = true;
        else
            NetObjs(i).HasUplink = false;
        end;
    end;
end;


for i=1:NumNodes
    NetObjs(i).NodePower = [0 0];

    for j=3:NumNodes
```

```
        NetObjs(i).NodePower = [NetObjs(i).NodePower 0];
    end;
end;

for i=1:NumNodes
    for j=1:NumNodes
        if i ~= j
            IsDiffFloor = abs(NetObjs(i).FloorNum - NetObjs(j).FloorNum);
            Distance = sqrt((NetObjs(i).FloorX -
NetObjs(j).FloorX)^2+(NetObjs(i).FloorY - NetObjs(j).FloorY)^2);
            if IsDiffFloor == 1
                PLoss = 0.01+10*5.04*log(Distance)+2*5-23-30;
                NetObjs(i).NodePower(j) = PLoss;
                if PLoss <= 115
                    NetObjs(i).ConnectedNode =
strcat(NetObjs(i).ConnectedNode,alpha(j));
                else
                    NetObjs(i).ConnectedNode = strcat(NetObjs(i).ConnectedNode,'n');
                end;
            else
                PLoss = 0.01+10*2.76*log(Distance)+2*5-23-30;
                NetObjs(i).NodePower(j) = PLoss;
                if PLoss <= 115
                    NetObjs(i).ConnectedNode =
strcat(NetObjs(i).ConnectedNode,alpha(j));
                else
                    NetObjs(i).ConnectedNode = strcat(NetObjs(i).ConnectedNode,'n');
                end;
            end;
        else
            NetObjs(i).ConnectedNode = strcat(NetObjs(i).ConnectedNode,'n');
        end;
    end;
end;
```

```
firstFloor=0;
secondFloor=0;

for i=1:NumNodes
    NetObjs(i).RouteMap='';
    if NetObjs(i).FloorNum==1
        firstFloor=firstFloor+1;
    else
        secondFloor=secondFloor+1;
    end;
    NetObjs(i).totalOutBitRate = 0;
    NetObjs(i).totalInBitRate = 0;
end;

%Create Routemap
for i=1:NumNodes
    lengthCN=0;
    for j=1:NumNodes
        if NetObjs(i).ConnectedNode(j) ~= 'n'
            lengthCN=lengthCN+1;
        end;
    end;
    if NetObjs(i).FloorNum == 1
        if lengthCN >= firstFloor
            firstUplinkNode = NetObjs(i).NodeName;
        end;
    else
        if lengthCN >= secondFloor
            secondUplinkNode = NetObjs(i).NodeName;
        end;
        if NetObjs(i).HasUplink
            SNUplinkNode = NetObjs(i).NodeName;
        end;
    end;
end;

for i=1:NumNodes
    for j=1:NumNodes
        if i~=j
            if NetObjs(i).ConnectedNode(j) == 'n'
                if NetObjs(i).FloorNum == 1
                    if NetObjs(i).NodeName ~= firstUplinkNode
                        NetObjs(i).RouteMap(j) = firstUplinkNode;
                    else
                        NetObjs(i).RouteMap(j) = secondUplinkNode;
                    end;
                else
                    if NetObjs(i).NodeName ~= secondUplinkNode
                        NetObjs(i).RouteMap(j) = secondUplinkNode;
                    else
                        NetObjs(i).RouteMap(j) = firstUplinkNode;
                    end;
                end;
            else
                NetObjs(i).RouteMap(j) = NetObjs(i).ConnectedNode(j);
            end;
        else
            NetObjs(i).RouteMap(j) = NetObjs(i).ConnectedNode(j);
        end;
```

```
    end;
    if NetObjs(i).FloorNum == 1
        if NetObjs(i).NodeName ~= firstUplinkNode
            NetObjs(i).RouteMap(j+1) = firstUplinkNode;
        else
            NetObjs(i).RouteMap(j+1) = secondUplinkNode;
        end;
    else
        NetObjs(i).RouteMap(j+1) = SNUplinkNode;
    end;
end;


for i=1:NumNodes
    %0=Relay (no packets to or from), 1=Sensor, 2=Camera, 3=Audio Device,
    %4=Video Device, 5=Computer/Tablet
    switch NetObjs(i).ObjType
        case 0
            NetObjs(i).DestBitRate = 0;
            NetObjs(i).OrigBitRate = 0;
        case 1
            NetObjs(i).DestBitRate = 20000;
            NetObjs(i).OrigBitRate = 20000;
        case 2
            NetObjs(i).DestBitRate = 20000;
            NetObjs(i).OrigBitRate = 4000000;
        case 3
            NetObjs(i).DestBitRate = 300000;
            NetObjs(i).OrigBitRate = 20000;
        case 4
            NetObjs(i).DestBitRate = 4000000;
            NetObjs(i).OrigBitRate = 20000;
        otherwise
            NetObjs(i).DestBitRate = 400000;
            NetObjs(i).OrigBitRate = 400000;
    end;
end;


for i=1:NumNodes
    PacketsOut = NetObjs(i).OrigBitRate;
    PacketsIn = NetObjs(i).DestBitRate;
    StartNode = NetObjs(i).NodeName;

    %Go from node out for packetsOut
    CurrNode = StartNode;

    while true
        j = strfind(alpha,CurrNode);
        NetObjs(j).totalOutBitRate = NetObjs(j).totalOutBitRate + PacketsOut;

        if CurrNode == SNUplinkNode
            break;
        end;
        CurrNode = NetObjs(j).RouteMap(NumNodes+1);

        j = strfind(alpha,CurrNode);
        NetObjs(j).totalInBitRate = NetObjs(j).totalInBitRate + PacketsOut;
    end;

    %Go from uplink node in for packetsIn
```

```
    CurrNode = SNUplinkNode;

    while true                                                                    28
        j = strfind(alpha,CurrNode);
        NetObjs(j).totalInBitRate = NetObjs(j).totalInBitRate + PacketsIn;

        if CurrNode == StartNode;
            break;
        end;

        NetObjs(j).totalOutBitRate = NetObjs(j).totalOutBitRate + PacketsIn;
        CurrNode = NetObjs(j).RouteMap(i);

    end;
end;
```

./GraphOutput.m

```
NodeInputBps=[NetObjs(1).totalInBitRate NetObjs(2).totalInBitRate];
GenInputBps=[NetObjs(1).DestBitRate NetObjs(2).DestBitRate];
NodeOutputBps=[NetObjs(1).totalOutBitRate NetObjs(2).totalOutBitRate];
GenOutputBps=[NetObjs(1).OrigBitRate NetObjs(2).OrigBitRate];
DetectPower = [115 115];

for i = 3:NumNodes
    NodeInputBps=[NodeInputBps NetObjs(i).totalInBitRate];
    GenInputBps=[GenInputBps NetObjs(i).DestBitRate];
    NodeOutputBps=[NodeOutputBps NetObjs(i).totalOutBitRate];
    GenOutputBps=[GenOutputBps NetObjs(i).OrigBitRate];
    DetectPower=[DetectPower 115];
end;

i = strfind(alpha,firstUplinkNode);
j = strfind(alpha,secondUplinkNode);
k = strfind(alpha,SNUplinkNode);

subplot(2,1,1);
IBar = bar(NodeInputBps);
IBar.FaceColor = 'flat';
IBar.CData(i,:)=[.5 0 .5];
if j~=k
    IBar.CData(j,:)=[0  0 .5];
    IBar.CData(k,:)=[0 .5 0];
else
    IBar.CData(k,:)=[0 .5 0];
end;

hold on;
bar(GenInputBps, 0.25,'FaceColor',[0 0.7 0.7]);
hold off;
xlabel('Node #');
ylabel('Bits/s');
title('Input Bits/s');

subplot(2,1,2);
OBar = bar(NodeOutputBps);
OBar.FaceColor = 'flat';
OBar.CData(i,:)=[.5 0 .5];
if j~=k
    OBar.CData(j,:)=[0 0 .5];
    OBar.CData(k,:)=[0 .5 0];
else
    OBar.CData(k,:)=[0 .5 0];
end;

hold on;
bar(GenOutputBps, 0.25,'FaceColor',[0 0.7 0.7]);
hold off;

xlabel('Node #');
ylabel('Bits/s');
title('Output Bits/s');


figure(2);
subplot(3,1,1);
PFFBar = bar(NetObjs(i).NodePower);
```

```
PFFBar.FaceColor = 'flat';
hold;
plot(DetectPower);

xlabel('Node #');
ylabel('dBm');
title('First Floor Node');


subplot(3,1,2);
PSFBar = bar(NetObjs(j).NodePower);
PSFBar.FaceColor = 'flat';
hold;
plot(DetectPower);

xlabel('Node #');
ylabel('dBm');
title('Second Floor Node');


subplot(3,1,3);
PSNBar = bar(NetObjs(k).NodePower);
PSNBar.FaceColor = 'flat';
hold;
plot(DetectPower);

xlabel('Node #');
ylabel('dBm');
title('Serving Network Relay Node');
```

./QueuingModel.m



```
txrate = 1000000000;

pFirst = [0];
pSecond = [0];
pUplink = [0];
CLFirstM = [0];
CLSecondM = [0];
CLUplinkM = [0];
NodesM = [0];
betaM = [0];

txrateM = [txrate];

for loop=1:20
    alpha=2;
    beta=0.5*loop;
    betaM=[betaM beta];

    P = alpha/(alpha+beta);
    buffer = 4000000;

    txrate = 1000000000;
    txrateM = [txrateM txrate];
    PL = 10^-5;

    RpFirst = NodeOutputBps(i)/firstFloor;
    NFirst = firstFloor;
    if j==k
        RpSecond = NodeOutputBps(j)/(secondFloor+firstFloor);
        NSecond = NumNodes;
        RpUplink = RpSecond;
        NUplink = NumNodes;
    else
        RpSecond = NodeOutputBps(j)/(firstFloor+1);
        NSecond = firstFloor+1;
        RpUplink = NodeOutputBps(k)/(NumNodes);
        NUplink = NumNodes;
    end;

    kFirst = beta*buffer/(RpFirst*(1-P)*log(1/PL));
    kSecond = beta*buffer/(RpSecond*(1-P)*log(1/PL));
    kUplink = beta*buffer/(RpUplink*(1-P)*log(1/PL));

    %NodesM = [NodesM loop*10];
    CLFirst = ((1-kFirst)/2+sqrt(((1-kFirst)/2)^2+kFirst*P))*NodeOutputBps(i);
%RpFirst*round((loop*10)/2);
    CLSecond = ((1-kSecond)/2+sqrt(((1-kSecond)/2)^2+kSecond*P))*NodeOutputBps(j);
%RpSecond*round((loop*10)/2+1);
    CLUplink = ((1-kUplink)/2+sqrt(((1-kUplink)/2)^2+kUplink*P))*NodeOutputBps(k);
%RpUplink*loop*10;

    CLFirstM = [CLFirstM CLFirst];
    CLSecondM = [CLSecondM CLSecond];
    CLUplinkM = [CLUplinkM CLUplink];

    pFirst = [pFirst CLFirst/txrate];
    pSecond = [pSecond CLSecond/txrate];
    pUplink = [pUplink CLUplink/txrate];
end;
```

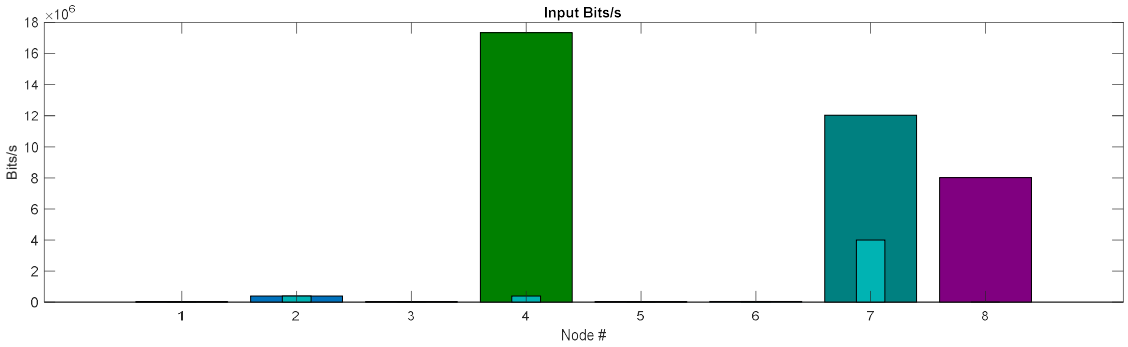
```
figure(3);

subplot(3,1,1);
CLFBar = bar(betaM,CLFirstM);
CLFBar.FaceColor = 'flat';
xlabel('Beta');
ylabel('bits/sec');
title('First Floor Node');

subplot(3,1,2);
CLSBar = bar(betaM,CLSecondM);
CLSBar.FaceColor = 'flat';
xlabel('Beta');
ylabel('bits/sec');
title('Second Floor Node');

subplot(3,1,3);
CLUBar = bar(betaM,CLUplinkM);
CLUBar.FaceColor = 'flat';
xlabel('Beta');
ylabel('bits/sec');
title('SN Uplink Node');
```

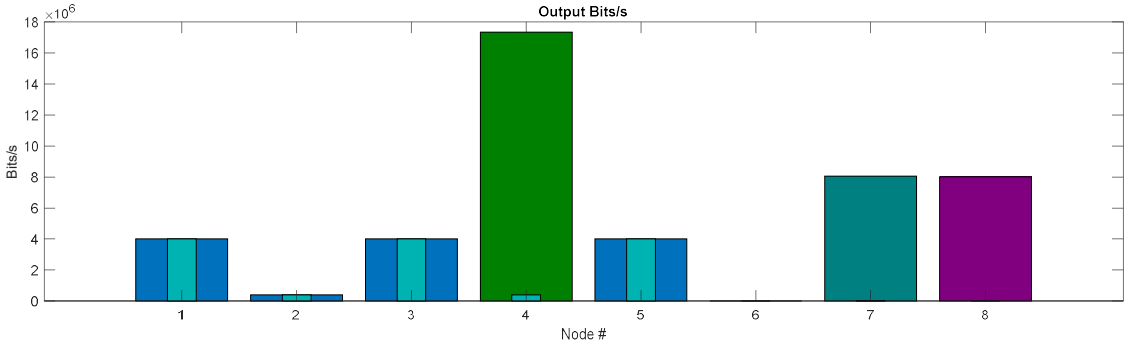
## APPENDIX B – POWER CALCULATIONS

| | 64GHz | 71GHz | | | |
|---|---|---|---|---|---|
| Lambda (meters)= | 0.0046875 | 0.004225352113 | UE Power assuming | 23 dBm | |
| PL(d0=1m) = | 45.27478447 | 45.72556822 | | | |
| mean loss exponent (n) | 5.04 | 2 Floors Any Building | 2.76 | Same Floor | |
| | | | | | |
| Distance (m) + 2 walls: | | | Distance (m) + 2 walls: | | |
| 5 | 37.50287269 | 37.95365644 | 20 | 38.18321235 | 38.6339961 |
| 10 | 52.67478447 | 53.12556822 | 40 | 46.49164023 | 46.94242398 |
| 20 | 67.84669625 | 68.29748 | 60 | 51.35175898 | 51.80254273 |
| 30 | 76.72169571 | 77.17247946 | 80 | 54.80006811 | 55.25085186 |
| 40 | 83.01860804 | 83.46939178 | 100 | 57.47478447 | 57.92556822 |
| 50 | 87.90287269 | 88.35365644 | 120 | 59.66018686 | 60.11097061 |
| 60 | 91.89360749 | 92.34439124 | 140 | 61.50791826 | 61.95870201 |
| 70 | 95.26772569 | 95.71850944 | 160 | 63.10849599 | 63.55927974 |
| 80 | 98.19051982 | 98.64130356 | 180 | 64.52030561 | 64.97108936 |
| 90 | 100.7686069 | 101.2193907 | 200 | 65.78321235 | 66.2339961 |
| 100 | 103.0747845 | 103.5255682 | 220 | 66.92565046 | 67.37643421 |
| 110 | 105.1609758 | 105.6117596 | 240 | 67.96861474 | 68.41939849 |
| 120 | 107.0655193 | 107.516303 | 260 | 68.92804888 | 69.37883262 |
| 130 | 108.8175294 | 109.2683132 | 280 | 69.81634614 | 70.26712989 |
| 140 | 110.4396375 | 110.8904212 | 300 | 70.6433311 | 71.09411485 |
| 150 | 111.9497839 | 112.4005677 | 320 | 71.41692387 | 71.86770762 |
| 160 | 113.3624316 | 113.8132153 | 340 | 72.14360258 | 72.59438633 |
| 170 | 114.6894101 | 115.1401939 | 360 | 72.82873349 | 73.27951724 |
| 180 | 115.9405187 | 116.3913025 | 380 | 73.47681174 | 73.92759549 |
| 190 | 117.123966 | 117.5747497 | 400 | 74.09164023 | 74.54242398 |

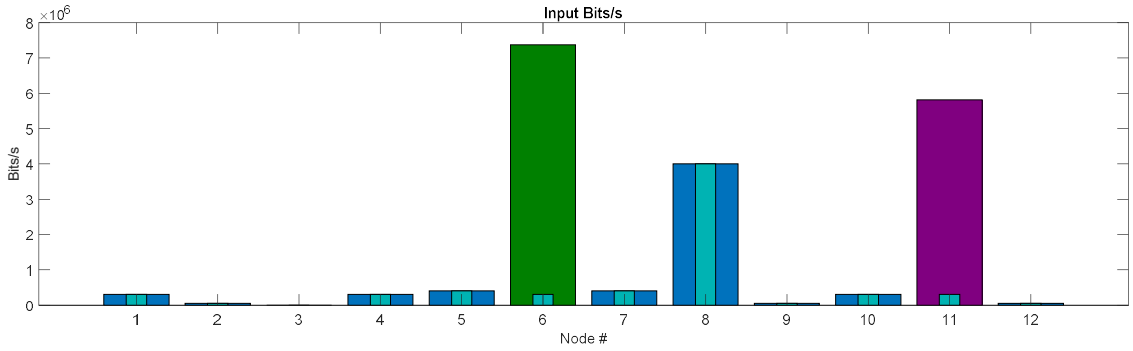
## APPENDIX C – SIMULATOR GRAPH OUTPUT

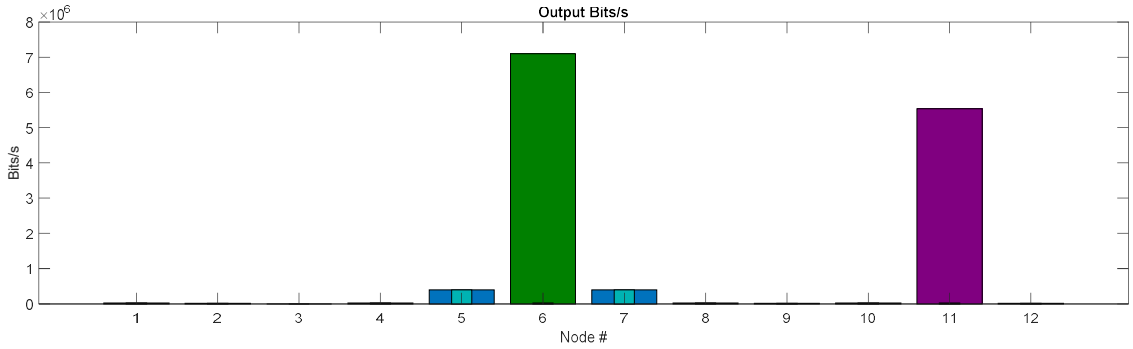
8 node Setup: 34



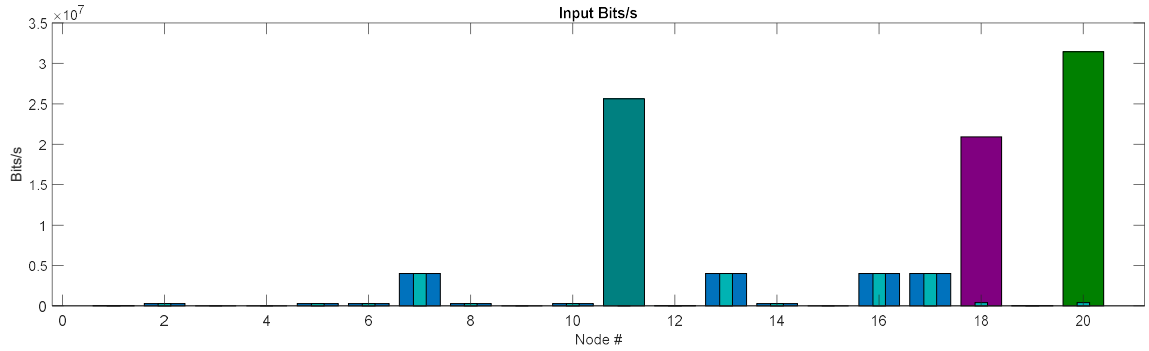

12 Node Setup

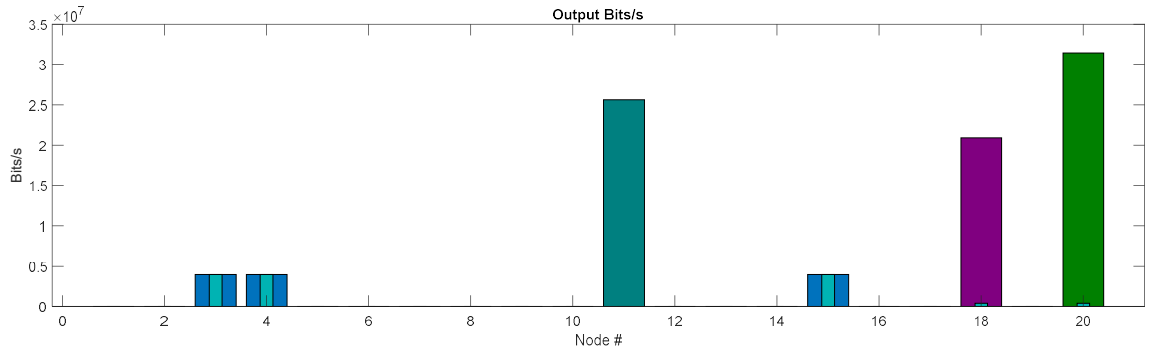


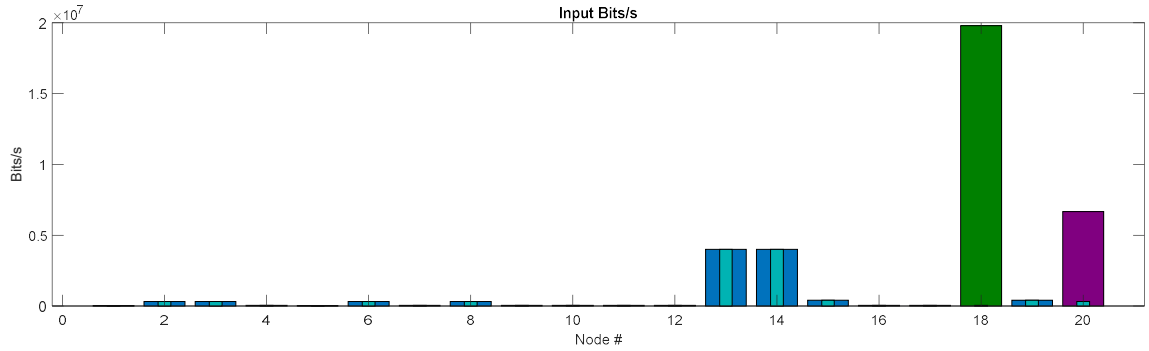



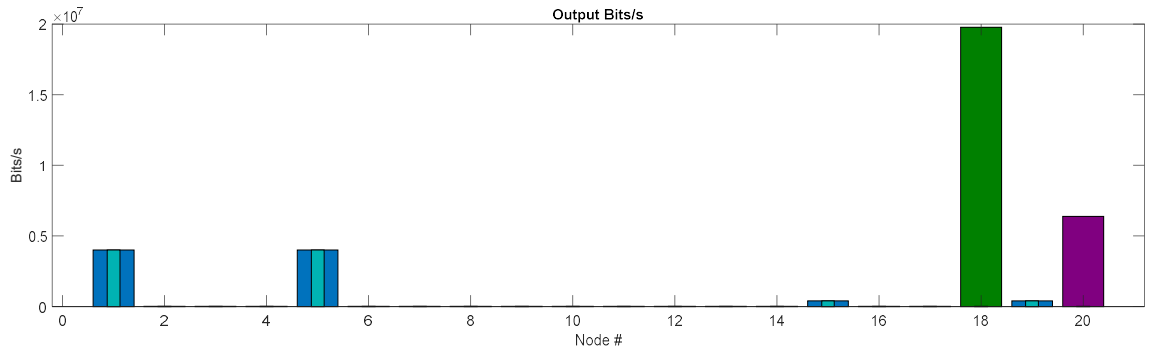
20 node Setup

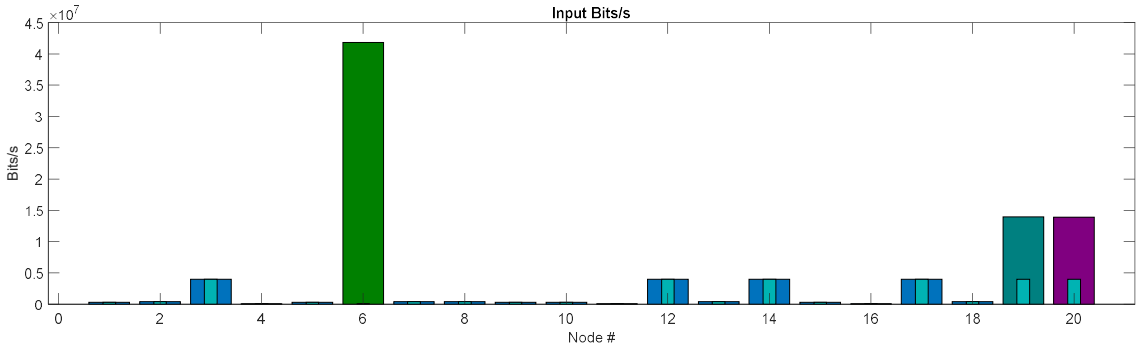


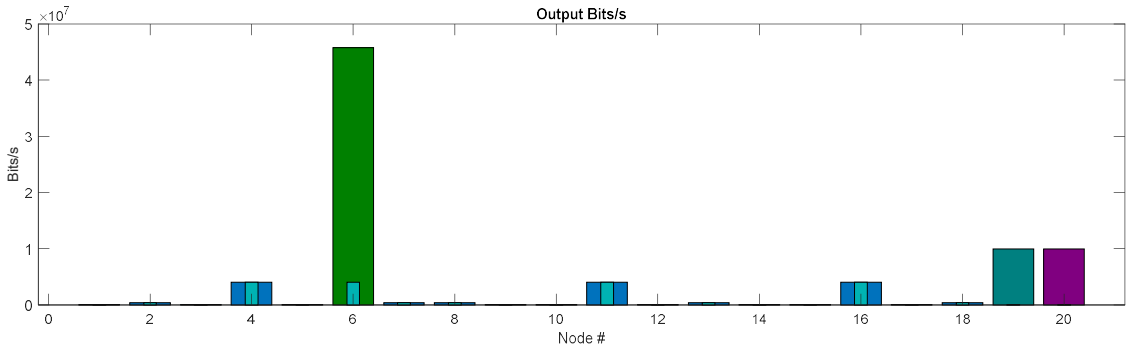


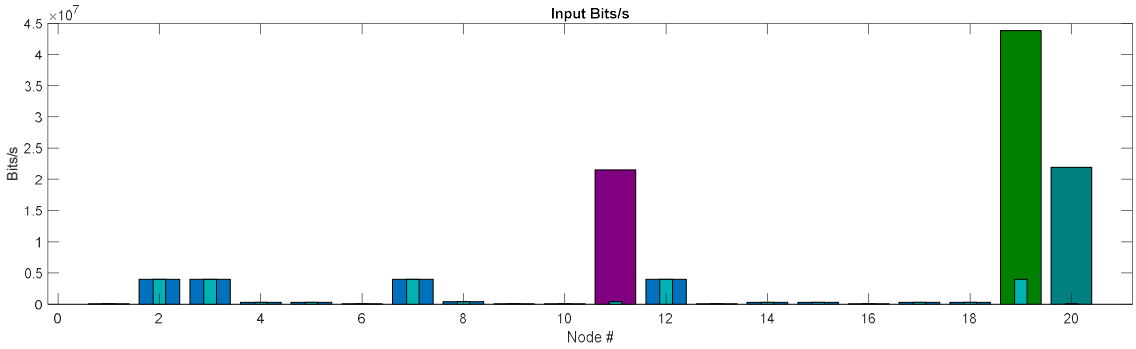



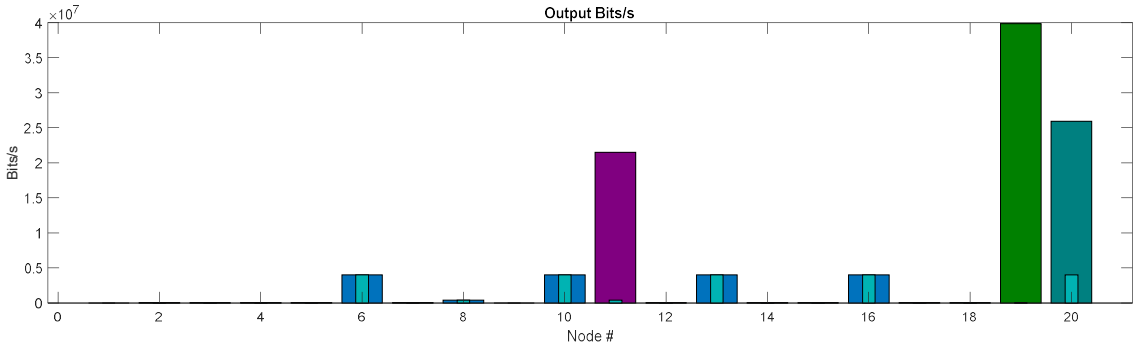



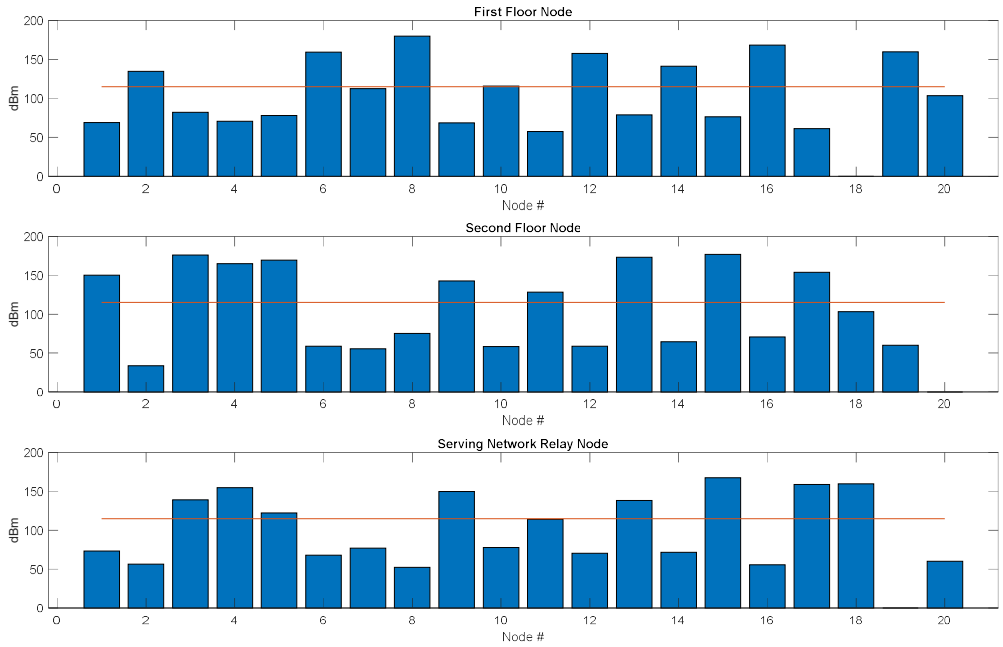



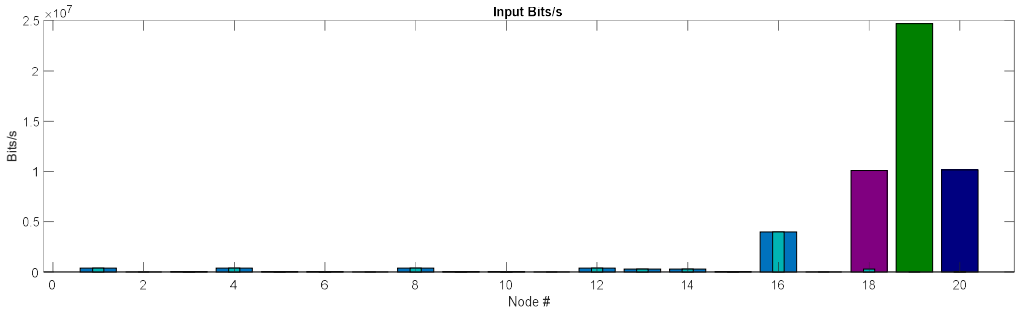

Input Bits/s
×10^7
Bits/s
Node #

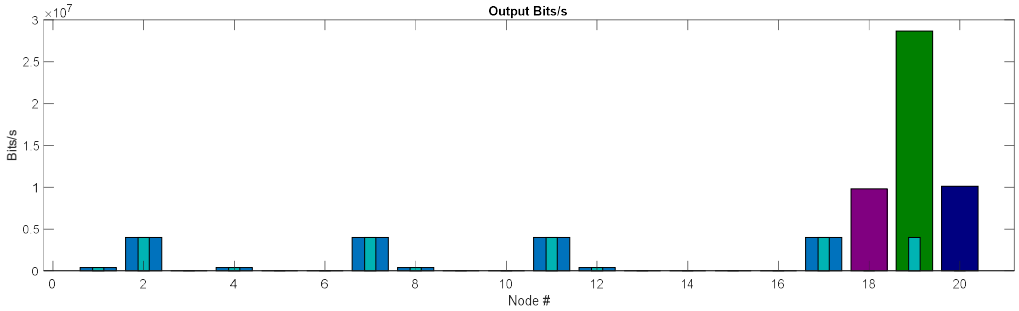

Output Bits/s
×10^7
Bits/s
Node #

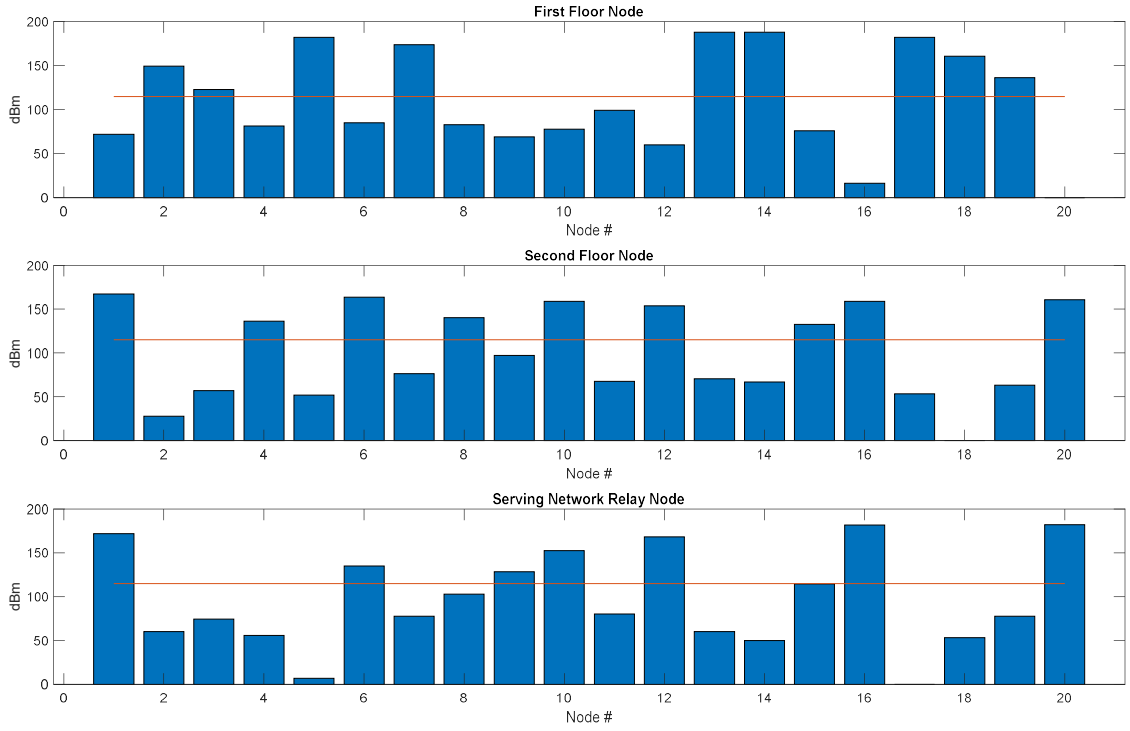

Input Bits/s
×10^7
Bits/s
Node #


Output Bits/s
×10^7
Bits/s
Node #

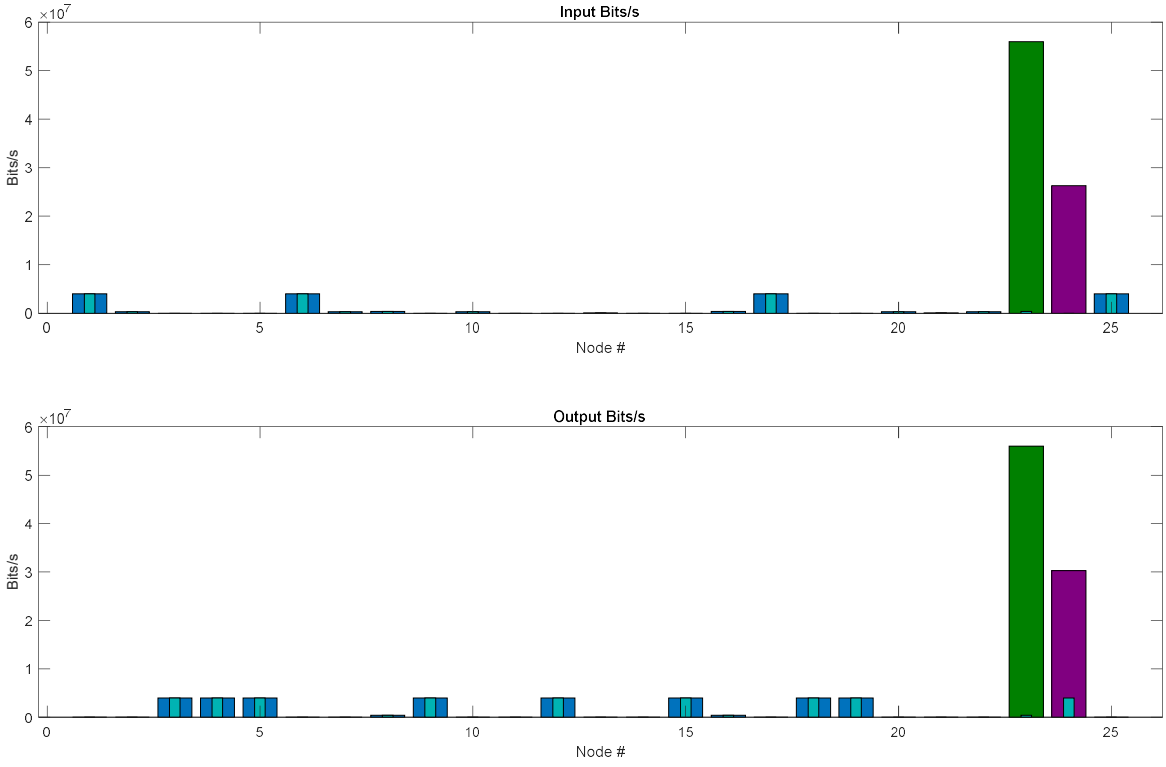

First Floor Node
dBm
Node #
Second Floor Node
dBm
Node #
Serving Network Relay Node
dBm
Node #

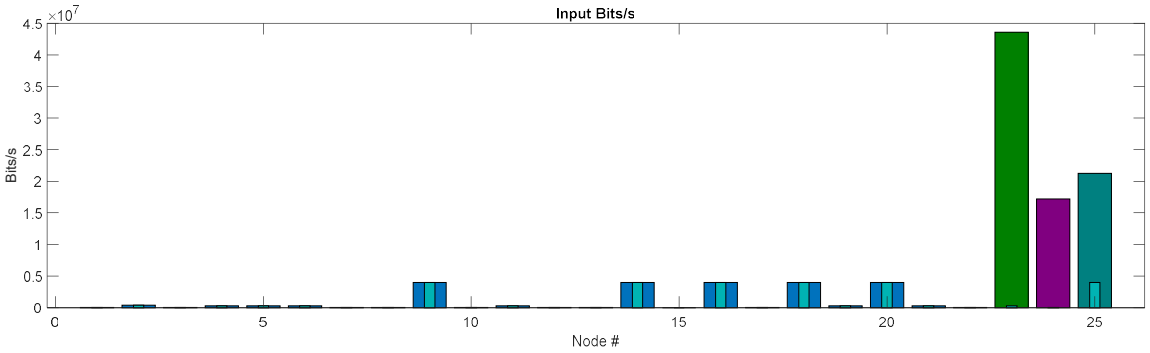

Input Bits/s
×10^7
Bits/s
Node #

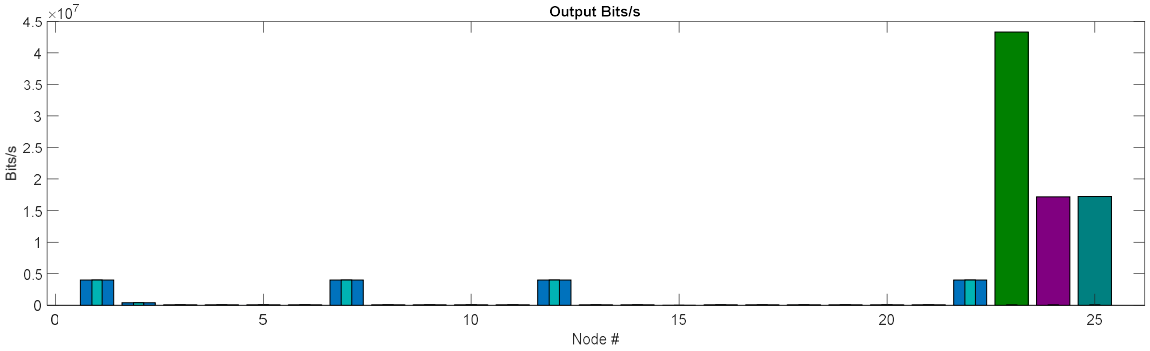

Output Bits/s
×10^7
Bits/s
Node #

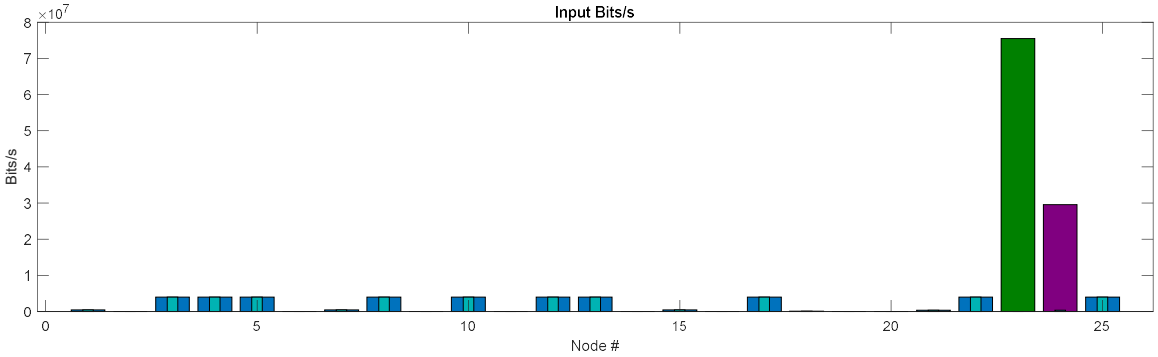


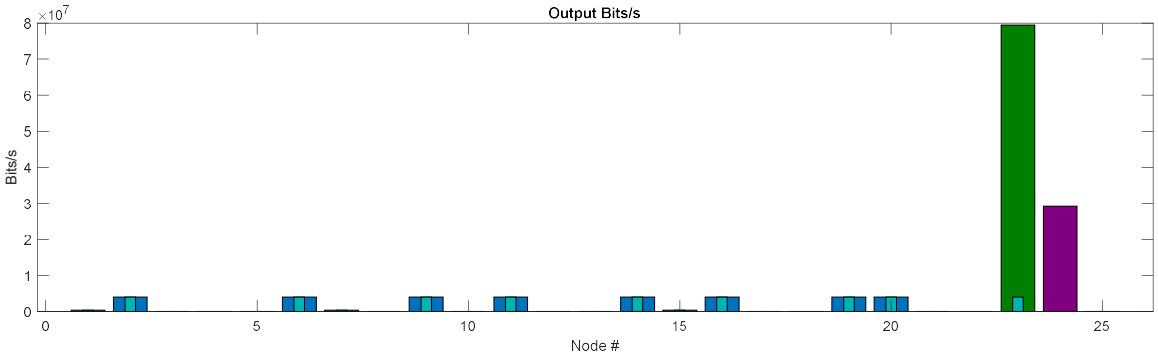

25 Node Setup

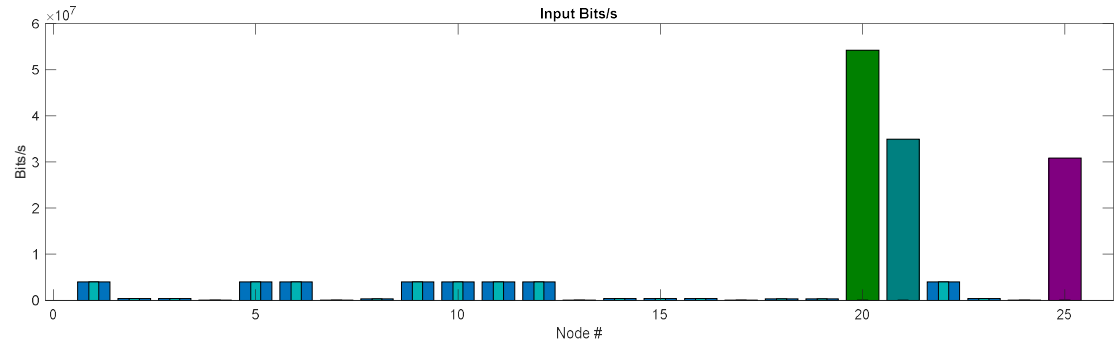


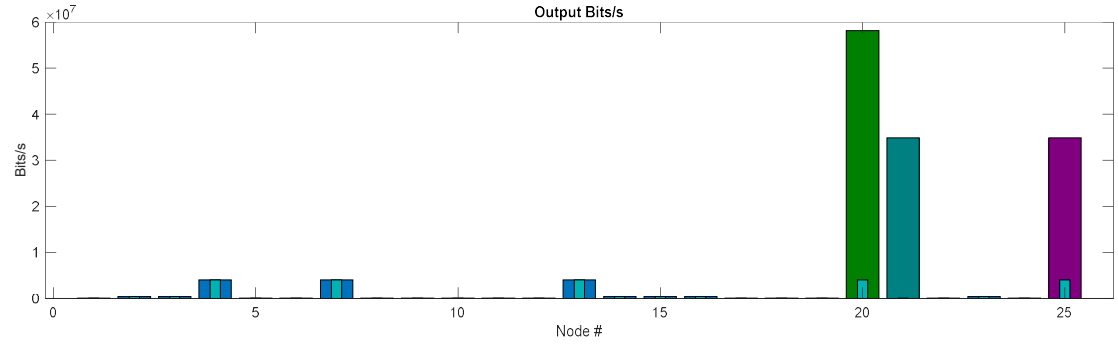

Input Bits/s
×10^7
4.5
4
3.5
3
2.5
2
1.5
1
0.5
0
Bits/s
0
5
10
15
20
25
Node #

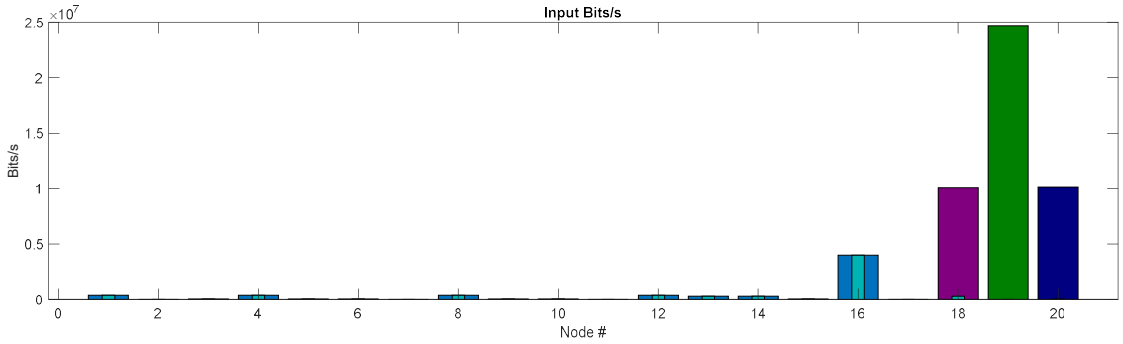

Output Bits/s
×10^7
4.5
4
3.5
3
2.5
2
1.5
1
0.5
0
Bits/s
0
5
10
15
20
25
Node #

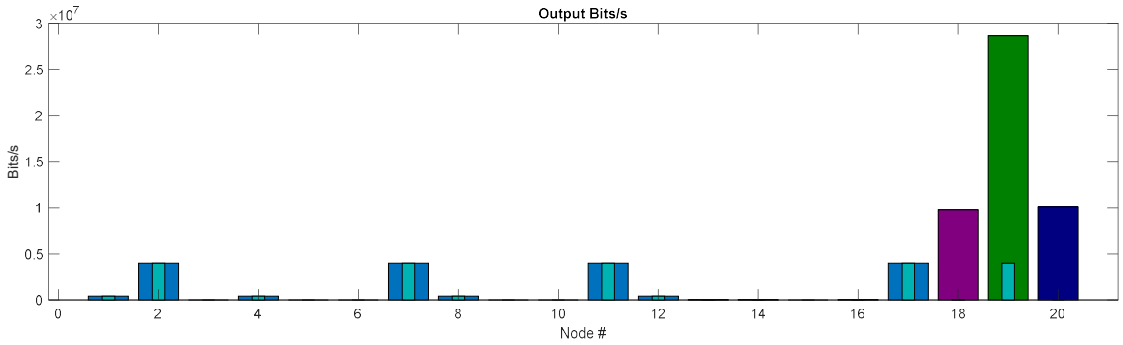

Input Bits/s
×10^7
8
7
6
5
4
3
2
1
0
Bits/s
0
5
10
15
20
25
Node #

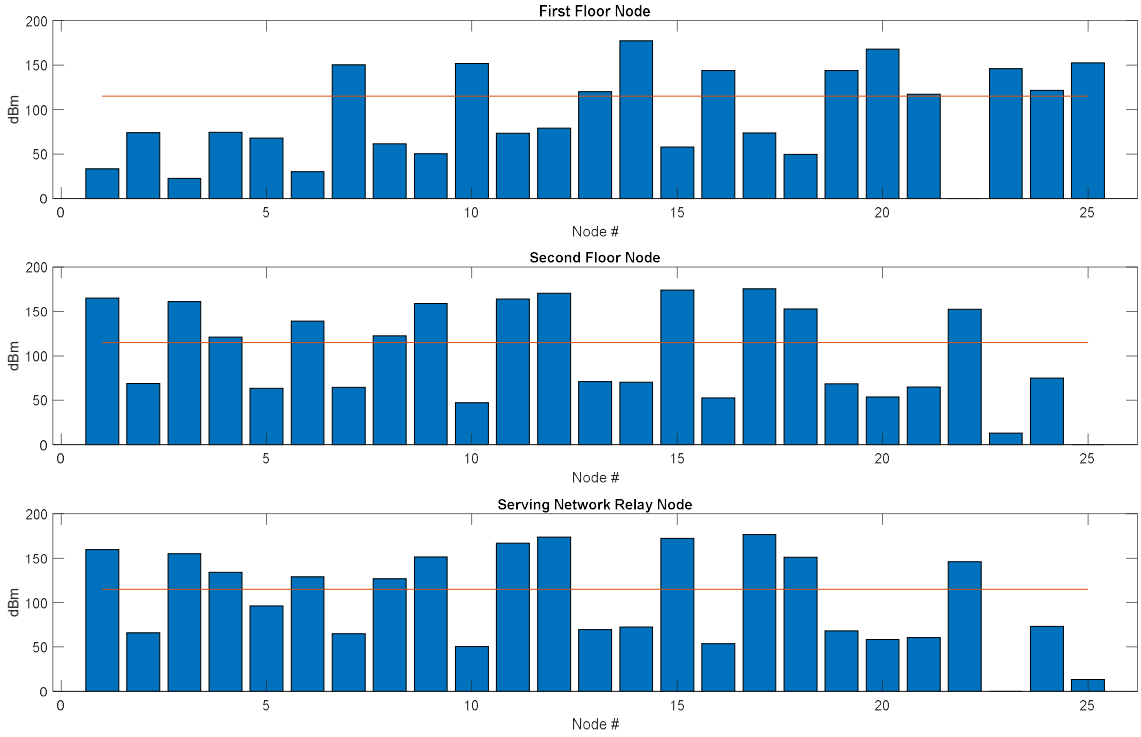

Output Bits/s
×10^7
8
7
6
5
4
3
2
1
0
Bits/s
0
5
10
15
20
25
Node #

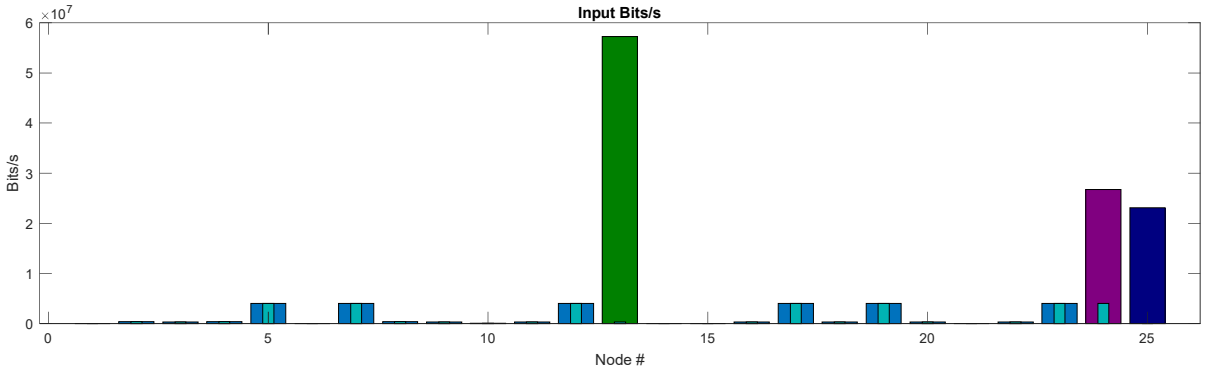

Input Bits/s
×10^7
Bits/s
Node #

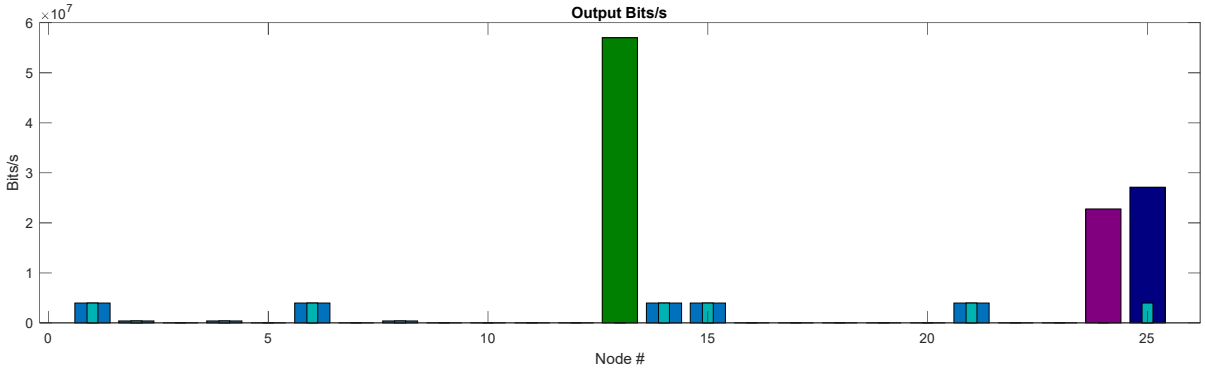

Output Bits/s
×10^7
Bits/s
Node #

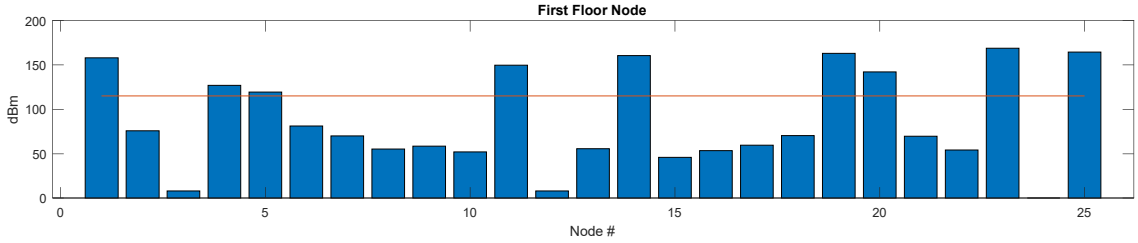

Input Bits/s
×10^7
Bits/s
Node #

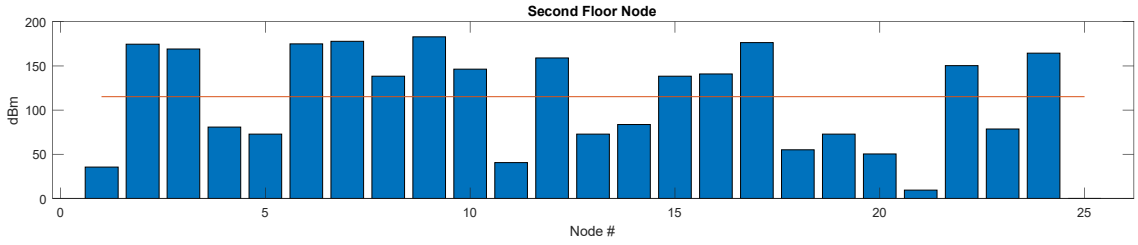

Output Bits/s
×10^7
Bits/s
Node #

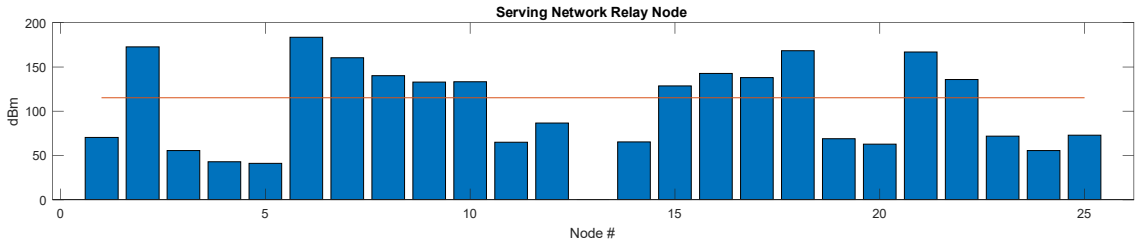

First Floor Node
dBm
Node #
Second Floor Node
dBm
Node #
Serving Network Relay Node
dBm
Node #


Input Bits/s
×10^7
Bits/s
Node #


Output Bits/s
×10^7
Bits/s
Node #

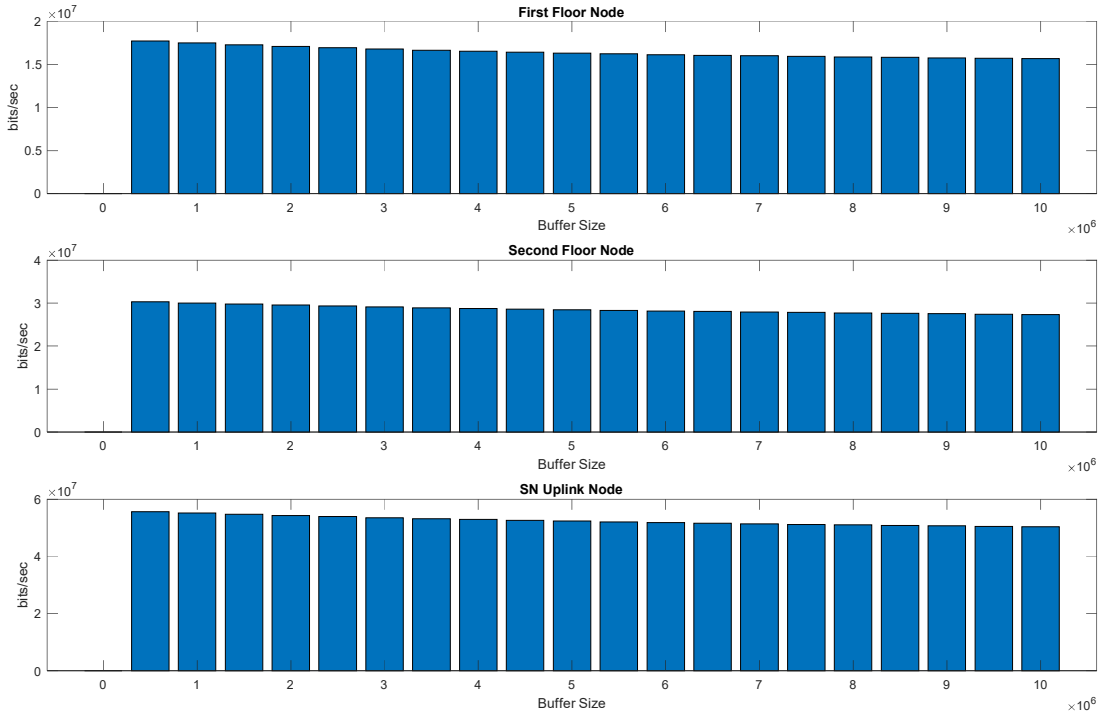

First Floor Node
dBm
Node #


Second Floor Node
dBm
Node #

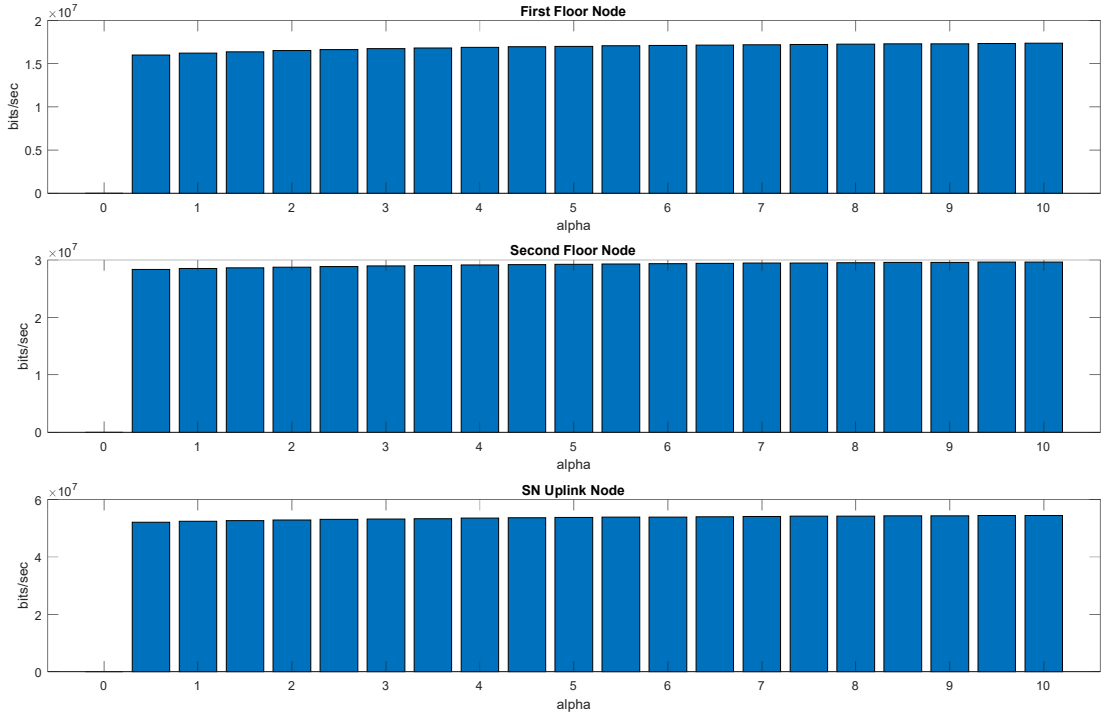

Serving Network Relay Node
dBm
Node #

Queuing Model Graphs:

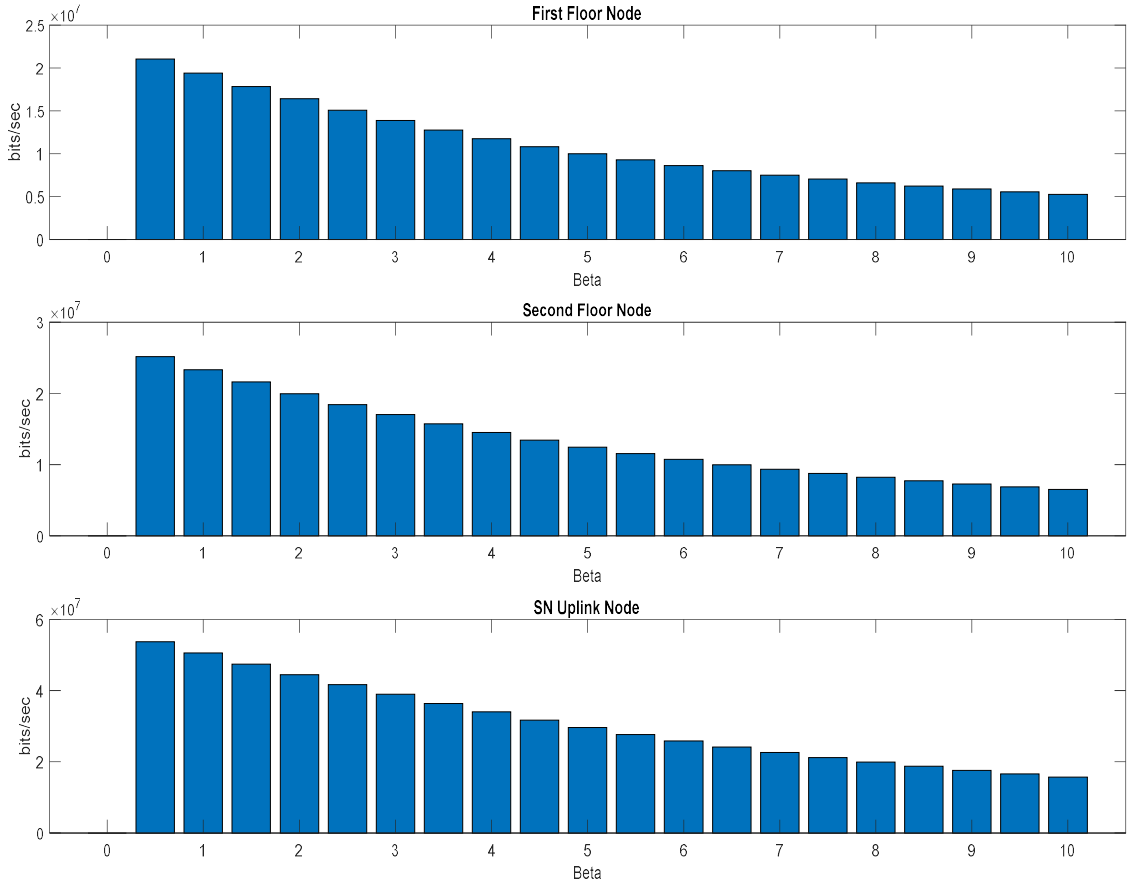




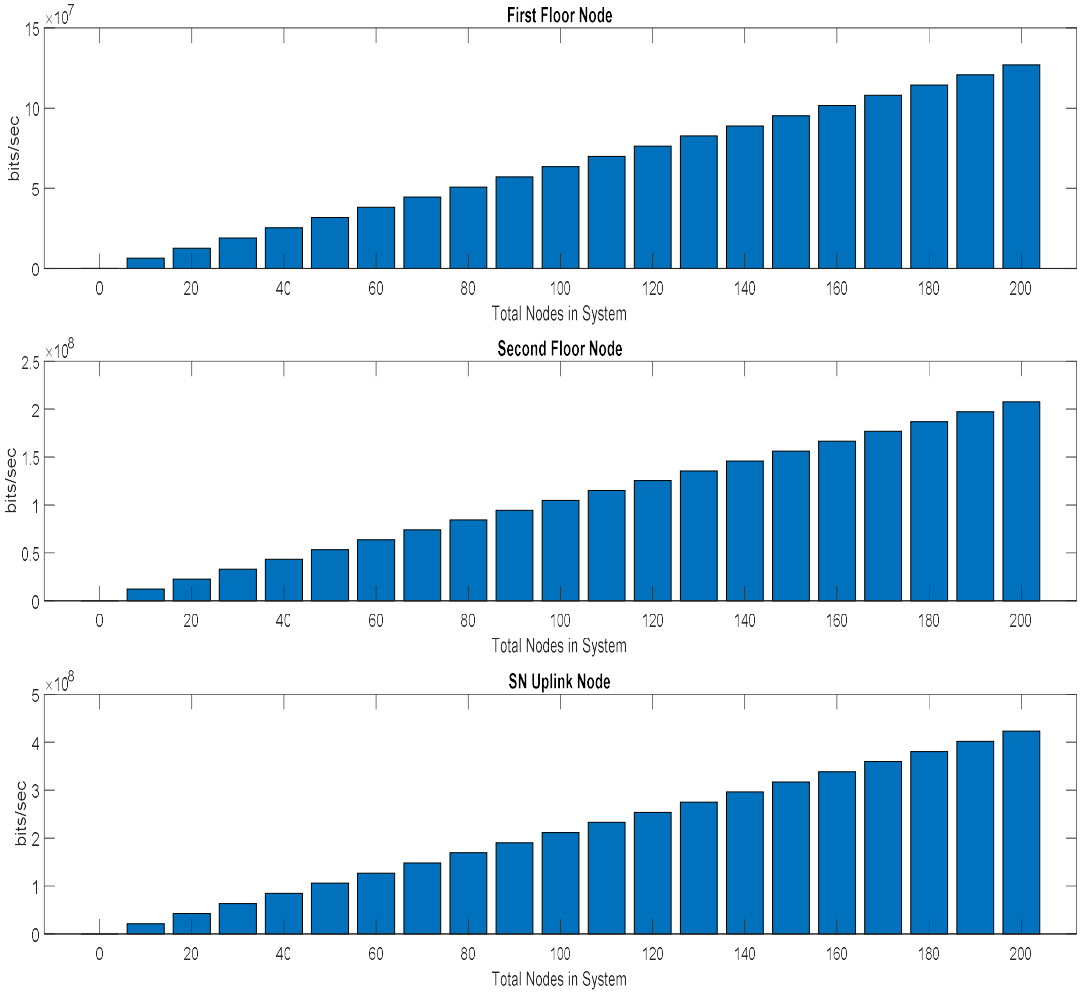

Fixed alpha=2, beta=0.5, 25 nodes, buffer size from 500k to 10 Meg




Fixed Buffer size 4Meg, beta 0.5, 25 nodes, alpha from 0.5 to 10.




Alpha set to 2, buffer fixed at 4 Meg, Number of nodes at 25, and beta varied between 0.5 and 10.




Fixed Buffer size 4Meg, alpha fixed at 2, number of nodes variable from 0 to 200. Still well under the 1 Gig limit: