# Random Linear Network Coding for Do-it-Yourself Networks

**6.1850 Formal Project Proposal**
**Katrina LaCurts, Jessie Stickgold-Sarah**

Lili Wilson
lmwilson@mit.edu

Sean Cheng
scheng12@mit.edu

October 30, 2023

## 1 Introduction

Over the past couple of decades, the Internet has grown crucial to daily life. Its ubiquity has facilitated the dissemination of information, and it is becoming increasingly important in modern-day jobs, schooling, and personal life. However, the Internet's significance does not equate to universality: a study by the Pew Research Center found that as of 2021, 33% of American adults do not have home broadband internet access. [13] This percentage is higher amongst racial minorities and people with lower levels of income and education.

Internet access is largely mediated by Internet Service Providers (ISPs), which often charge users a monthly subscription fee in return for providing an internet connection to their homes. As for-profit companies, many of these ISPs focus their attention on providing service to high-income and urban areas. In practice, this often leaves lower-income and rural households without many options for ISPs: 200 million Americans only have a choice between two ISPs, and consumers without a choice pay up to five times more than people in areas with more options. [2] Not only do ISPs have significant power in setting a price tag on internet connectivity, they play a critical role in determining what content their customers can access on the internet and have a wealth of personal customer data.

In an effort to reclaim power over internet access, grassroots efforts to provide free and unmediated internet connectivity have become more prominent in recent years. Do-it-Yourself Networks (also known as DIY networks) have sprung up around the world, run by local community volunteers. These networks often leverage wireless mesh network topology to provide internet access in a more decentralized manner.

However, little work has been done to compare the baseline performance of DIY mesh networks to that of traditional ISP-based internet access. Our first contribution in this

project will be to provide comprehensive measurements of each system, focusing on bandwidth, throughput, and energy efficiency.

After this initial measurement, we aim to improve DIY mesh network performance by introducing "network coding" into our network. Proposed in 2000, network coding is an alternative way to transmit data throughout a network that aims to improve network throughput, energy efficiency, robustness, and security. Instead of sending packets sequentially, network coding schemes share linear combinations of packets and eliminate the need for maintaining packet queues at each switch. Network coding has been shown to be particularly powerful in networks that are richly interconnected, making DIY mesh networks an optimal target for a network coding implementation [8]. In this project, we examine the impact of integrating one specific network coding scheme, random linear network coding (RLNC), into DIY networks.

The remainder of this paper explores previous examples of DIY networks (Section 2), explains our motivations for using RLNCs (Section 3), lists our proposed contributions and computations (Sections 4 and 5), and lists potential concerns and fixes (Section 6).

# 2 Do-it-Yourself Networks

## 2.1 What are community-based networks?

DIY networks are part of a larger movement towards a free, open, and neutral internet. The goal is to connect communities through a network that exists outside of commercial ISPs, decentralizing internet access and returning power to the hands of internet users. In addition to providing ubiquitous and equal internet access, these networks also aim to reduce censorship and promote free speech, as traffic through them is not regulated by a central body like an ISP.

Wireless community networks have been deployed successfully around the world. These networks have served a variety of purposes in the communities they have been deployed in. These include providing connectivity to rural communities, empowering low-income households or refugees in cities, and forming robust internet architecture in case of natural disaster. The most prominent examples include Guifi.net in Spain [6][15], NYC Mesh in New York City [10], and Freifunk in Germany [5][9].

## 2.2 How DIY networks work: Wireless Mesh Networks

Each of these DIY networks use wireless mesh networks as their underlying structure. A wireless mesh network is a richly interconnected network topology, where every node in the network is connected to most other nodes. Due to the redundancy created by this inter-connectivity, mesh networks are self-healing: if any individual node fails, data can be sent easily via another route. It is also easy to add new nodes to a mesh network, making this network structure scalable.

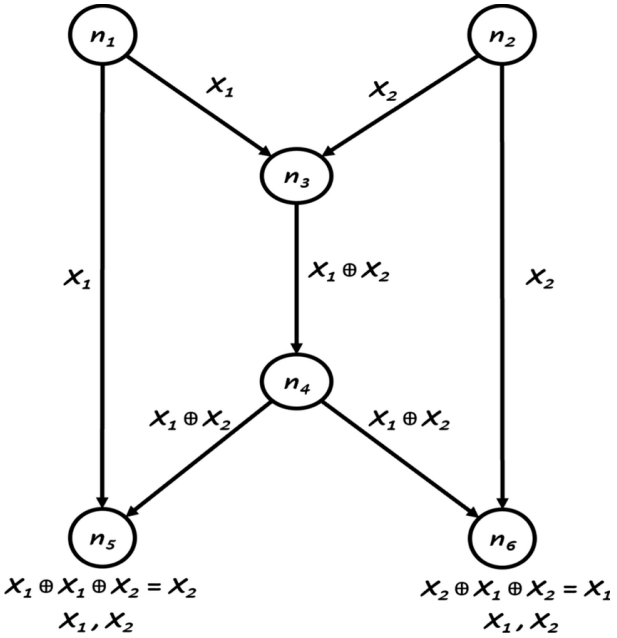


Figure 1: Butterfly Network [14]

Mesh networks are well-suited for DIY networking because they are decentralized and take advantage of the presence of a large and physically-proximal set of participating nodes. In the case of DIY community networks, every user in the mesh helps route traffic for all participating users. In this way, if one node in the network is connected to the larger internet, all nodes in the network can benefit from access, routing requests through one another. Guifi, NYC Mesh, and Freifunk all rely on these webs of interconnected nodes to provide broad internet access. As of October 2023, Guifi had 37,838 participating nodes [6], NYC Mesh had 1,256 [10], and Freifunk had 36,897 [4].

Understanding how to make these wireless mesh networks more effective at delivering content to users will have significant impact not only in the existing systems mentioned above, but could inspire the creation of more community DIY networks.

# 3 Random Linear Network Coding

## 3.1 What is it?

Linear network coding was proposed in the early 21st century, and offered an alternative to traditional methods of data transfer. The motivating example is the Butterfly Network, shown in Figure 1.

In a traditional network when the two nodes at the top of the network are sending packets $x_1$ and $x_2$, it will take 4 timesteps for both $x_1$ and $x_2$ to reach the two bottom nodes. This is because the link connecting $n_3$ and $n_4$ can only send one packet at a time,

so in the second timestep, this link can only hold one of $x_1$ or $x_2$, while the other one is queued. The second packet will then be sent in this link during the third timestep, and reach the final receiver after the fourth timestep.

In linear network coding, instead of each link sending a single packet, they send a linear combination of packets, where the linear combinations are taken over some finite field. In this case, in the first timestep, $n_1$ sends $x_1$ to all possible receivers, tells the receivers that they are getting $x_1$, and $n_2$ does the same with $x_2$. The difference lies in the link from $n_3$ to $n_4$ - instead of waiting two steps to send two different packets, $n_3$ instead adds the two packets modulo 2, notes that the information it is sending is the XOR of $x_1$ and $x_2$, and sends it to $n_4$. $n_4$ then sends the resulting information to $n_5$ and $n_6$. This process in total takes 3 timesteps, and both $n_5$ and $n_6$ receive all the information, as $n_5$ can XOR packet $x_1$ to $x_1 \oplus x_2$ to get $x_2$ out, and $n_6$ can do the same.

The main way that linear network coding is implemented in practice is with random linear network coding (RLNC). In this case, instead of simply adding two packets in a field, nodes instead choose two random numbers in the pre-determined finite field. The nodes then multiply packet one by the first random number, multiply packet two by the second random number, and then add the two together, noting the random numbers before sending out the information. To receive $n$ packets then, destination nodes just need to receive $n$ linearly independent packet combinations and then perform Gaussian elimination to get each packet.

### 3.2 How can it help DIY networks?

While sending an $n$-packet message in both traditional and RLNC techniques require the receiver to get $n$ packets, the amount of time spent sending these packets over a lossy network is much shorter for RLNC [3].

This is because in a normal TCP connection, source nodes send out sequential packets, and if they do not receive an ACK in a reasonable amount of time, they will resend the packet and all subsequent packets. After many dropped packets, this adds up, causing a slow connection. Normal TCP connections also require a queue at each switch, so busy switches often cause network slowdowns.

In RLNC, the packets arriving at the destination are linear combinations of the true source packets, so the order of transmission does not matter. Thus, in the case of a dropped packet, the source node(s) can simply send another packet, and with high probability due to the random coefficient selection at each node, the packet received by the destination node will contain unique information. Queuing is also not an issue here - nodes can take a linear combination and send both packets at the same time.

One possible counterargument against RLNC is the added computational power required by the destination to decode the linearly combined packets. While this cost is non-negligible, preventing unnecessary transmission of packets greatly outweighs the computational requirement of Gaussian elimination. Devices using RLNC coding have

been shown to use 2 to 3 orders of magnitude less energy than Wi-Fi transmission, leading to longer battery life and less power usage [1].

Finally, DIY networks can take full advantage of RLNC's strengths. As shown earlier, DIY networks are typically very highly connected, resembling mesh networks. Network coding performs best in graphs with a high degree of connectivity [8], can be implemented to have a degree of fault tolerance, [12], and has even been shown to achieve the upper bound in multicast problems with one source [7].

## 4 Proposed Contribution

Current research on the performance of DIY mesh networks is very limited, perhaps because these networks have emerged from such a grassroots movement. We could not find a comprehensive comparison of the experience users have connecting to internet via an ISP versus a community-based DIY network. This makes it difficult to understand how these mesh networks can be improved to better serve their users, and to understand what changes can be made to make free DIY network service comparable to the type of paid access you might get from an ISP.

While the use of network coding in mesh networks has been explored, it has not been applied in a community-based networking context. We believe that, after measuring differences between community mesh networking and ISP-based networking approaches, we will be able to further improve upon DIY mesh networks using RLNC. Our goal here is to see if we can achieve lower latency, lower energy consumption per node in the mesh network, and/or higher throughput with a network coding implementation.

In this project, we are proposing a new DIY mesh network that implements a network coding scheme (RLNC) to improve performance for users that rely on community networks. The system would be structured identically to existing wireless mesh networks, with nodes in the system routing traffic through one another. We will be modeling this new system closely off of the existing NYC Mesh architecture [11]. The NYC Mesh network architecture is well-documented and is currently in use, making it a good choice of system to base ours off of.

The DIY mesh network will consist of interconnected mesh routers throughout a community, which communicate with one another wirelessly (see Figure 2). In the case of the NYC Mesh network, these routers live on the tops of buildings and connect with all other routers that are in their line of sight. Some of these mesh routers will connect to a "supernode," which is a router that is connected to an internet exchange point (IXP) via a fiber optic cable. This node will serve as the gateway for the mesh network's internet access, bypassing any ISPs by interfacing with the IXP directly.

When a user wants to connect to the mesh network, they set up an internal router within their household that connects to a nearby mesh router, either in their building or in a neighbor's building. They can then connect their personal devices to the internal router and use Wi-Fi as they would if they were getting internet access from an ISP.

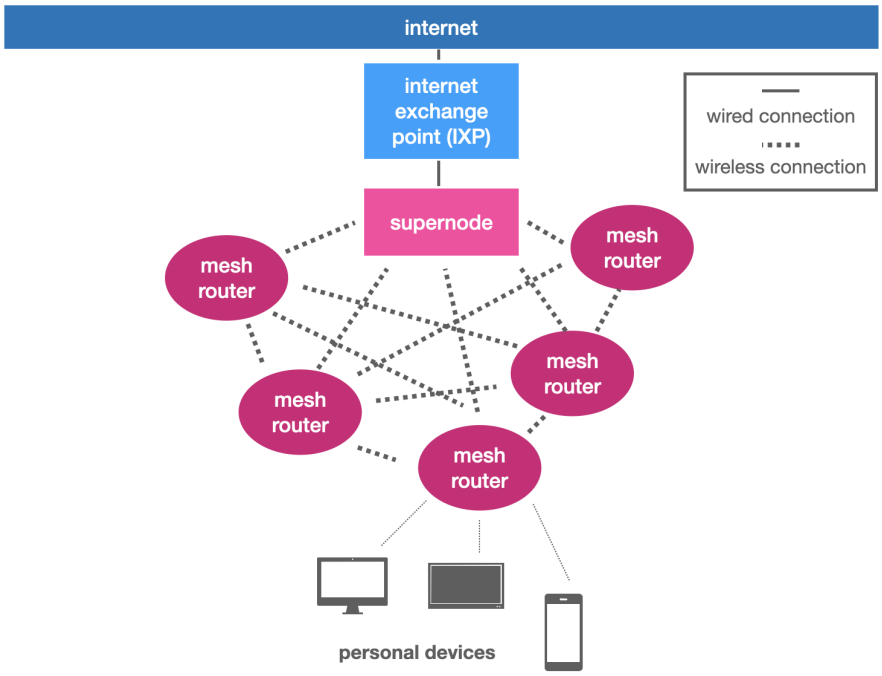


Figure 2: DIY Mesh Network System Architecture

The key change in our DIY mesh network implementation will occur when users want to make requests, since we will be transmitting traffic throughout the network using network coding. A user will begin to request data by sending out a request normally. The request will be sent through the mesh network to the supernode, where gateway internet access is enabled. At each step in the network as the request is sent to the supernode, individual mesh router nodes will have an opportunity to encode or re-encode the request packets they receive. Upon arrival at the supernode, packets are decoded and then sent to the internet using typical TCP protocol. Upon the supernode's receipt of responses, it will then encode packets using RLNC and broadcast the information back out to the mesh network.

Converting between network coding and typical TCP packet transmission at the supernode allows our network coding system to interface with existing network protocols without requiring changes external to the system.

## 5 Proposed Computation

Our computation will contain two parts: a simulation of the system, and an implementation of the system on a small network. We plan to compare the latency, throughput, and energy efficiency of the above system with both a more traditional DIY networking system that does not use RLNC, as well as a direct Wi-Fi connection.

For the simulation, we will create a program modeling a network. This network will have a single source sending information to other nodes. We fix default parameters such as maximum transmission rate in the wires and upload/download rates for the source that is connected to the internet. Then, we use these default parameters to create a similar network for the non-RLNC case, and compare the metrics between the two.

To implement this system, we propose using some number of Raspberry Pi machines as nodes. We will compare latency, throughput, and energy efficiency metrics of personal devices connected to an RLNC DIY network, a traditional DIY network, and Wi-Fi.

# 6 Possible Concerns and Questions

Here, we list possible issues with the project, and if we have an idea, possible remedies. We also compile a list of questions we have for future research.

## 6.1 People already use DIY networks for illegal/illicit purposes. Will improving DIY networking worsen the problem?

Yes - if this technology improves DIY networking, illicit uses of DIY networks will also be improved. However, technologies like the Internet itself are also used for nefarious purposes, but it is still being improved anyway to make lives better. Similarly, we believe that the benefits gained from improving DIY networking will outweigh the harms that it causes.

## 6.2 What will happen if some nodes become adversarial? What if the adversarial node is the one providing internet access?

We do not know yet - more work will be done about the security and robustness of the system to attacks.

## 6.3 Does network coding assume everything is multicast? If network coding is implemented in DIY networks, how would we ensure that private information only gets to one person?

We also do not know this yet. This was a particularly confusing question that requires more research.

## 6.4 Is the supernode of a DIY network a single point of failure? Does it ever get overwhelmed with requests?

From what we can tell, the supernodes of the NYC Mesh network each handle more than 400 nodes' requests each, and they are still working fine.

## References


[1] Acevedo, J., Scheffel, R., Wunderlich, S., Hasler, M., Pandi, S., Cabrera Guerrero, J., Fitzek, F., Fettweis, G., and Reisslein, M. Hardware acceleration for rlnc: A case study based on the xtensa processor with the tensilica instruction-set extension. *Electronics 7* (2018), 180. https://www.mdpi.com/2079-9292/7/9/180.

[2] Campisi, N. Millions lack broadband access. *Forbes* (2021). https://www.forbes.com/advisor/personal-finance/millions-lack-broadband-access/#:~:text=More%20than%20200%20million%20Americans,a%20White%20House%20fact%20sheet.

[3] CodeOnTechnologies. Random linear network coding-faq. https://www.codeontechnologies.com/technology/rlnc-faq.

[4] Dietel, T. Freifunk-karte.

[5] Freifunk. What is freifunk about? https://freifunk.net/en/what-is-it-about/.

[6] guifi.net. What is guifi.net? https://guifi.net/en/what_is_guifinet.

[7] Li, S.-Y., Yeung, R., and Cai, N. Linear network coding. *IEEE Transactions on Information Theory 49*, 2 (2003), 371–381. https://ieeexplore.ieee.org/document/1176612.

[8] Lima, L., Ferreira, D., and Barros, J. Topology matters in network coding. *Telecommunication Systems* (2012).

[9] Lopez, O. 'hacktivists' keep refugees online in dead zone germany. *Newsweek Magazine* (2016). https://www.newsweek.com/hacktivists-refugees-online-germany-478914.

[10] NYCMesh. Frequently asked questions. https://www.nycmesh.net/faq.

[11] NYCMesh. Typical installations. *NYC Mesh Docs*. https://docs.nycmesh.net/intro/typical-installs/.

[12] Peng, Y., Song, Q., Yu, Y., and Wang, F. Fault-tolerant routing mechanism based on network coding in wireless mesh networks. *Journal of Network and Computer Applications 37* (2014), 259–272. https://www.sciencedirect.com/science/article/pii/S1084804513000532.

[13] Pew. Internet/broadband fact sheet. *Pew Research Center* (2021). https://www.pewresearch.org/internet/fact-sheet/internet-broadband/#who-uses-the-internet?tabId=tab-b59d5c71-53f5-4023-9955-78a8bacc4e56.

[14] Qassim, Y. Cooperative diverse opportunistic network coding for wireless networks. *Wireless Personal Communications* (2023). https://www.researchgate.

net/figure/Butterfly-network-model-showing-network-coding_fig1_350154926.

[15] Roger Baig, Ramon Roca, F. F., and Navarro, L. guifi.net, a crowdsourced network infrastructure held in common. *Computer Networks* (2015). https://doi.org/10.1016/j.comnet.2015.07.009.