Peer-to-Peer Group Communication for City-Scale Mesh Networks

by

William A. Sussman

Bachelor of Science, Yale University (2021)

Submitted to the Department of Electrical Engineering and Computer Science
in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

May 2024

Authored by: William A. Sussman
Department of Electrical Engineering and Computer Science
May 17, 2024

Certified by: Hari Balakrishnan
Fujitsu Professor of Computer Science and Artificial Intelligence
Thesis Supervisor

Accepted by: Leslie A. Kolodziejski
Professor of Electrical Engineering and Computer Science
Chair, Department Committee on Graduate Students

Peer-to-Peer Group Communication for City-Scale Mesh Networks

by

William A. Sussman

Submitted to the Department of Electrical Engineering and Computer Science
on May 17, 2024 in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE

ABSTRACT

The Internet has become extremely centralized. The benefits of centralization have thus far outweighed the drawbacks, but users today are much more concerned about privacy, and reachability is increasingly threatened by natural disasters, political repression, cyberattacks, and human error. CityMesh provides an answer to this problem, constructing a decentralized mesh network out of wireless access points. To test our unicast routing protocol, we built a discrete-event network simulator using SimPy. However, we make several simplifying assumptions, and unicast is not sufficient for many applications. In this thesis, I show that our simulator nevertheless achieves 67.4% correlation with real data that we collected, and I generalize our simulator for multicast. Specifically, I compose our unicast primitive into multicast trees using three different topologies, and surprisingly find that Steiner trees perform worse than minimum spanning trees on average.

Thesis supervisor: Hari Balakrishnan
Title: Fujitsu Professor of Computer Science and Artificial Intelligence

Acknowledgments

My advisor, Hari, has been very patient with me. It has been an honor to work with him and Manya Ghobadi on this project, alongside James Lynch, Chenning Li, and Charlie Liu. I had a helpful conversation with David Karger, and my undergraduate advisor, Wenjun Hu, provided invaluable feedback. On a personal note, MIT Hillel has been a reliable source of support, and I can always count on my friends and family. Thank you all.

Contents

List of Figures

Chapter 1

Introduction

The Internet was supposed to be decentralized. Today, it is more centralized than ever: Google alone has nine services with over 1 billion users, including YouTube (2.1 billion), Gmail (1.8 billion), and Drive (1 billion) [1]. Meta owns Facebook (3 billion), WhatsApp (2 billion), Instagram (2 billion), and Messenger (979 million) [2]. Fundamentally, each of these applications has a central server for all its clients; an application may be sharded across many physical servers and datacenters, but it is still logically centralized.

For example: When `<userA>` sends an email to `<userB>@gmail.com`, the email is not sent directly to `<userB>`. Instead, the email is sent to `gmail.com`, and `<userB>` downloads a copy. This centralized architecture has benefits and drawbacks. On one hand, `<userB>` can download a copy onto many devices, or lose all copies without losing the email, and Google is responsible for maintaining the server (including security). On the other hand, Google can read all of `<userB>`'s emails, and if `gmail.com` is down or unreachable by `<userA>` or `<userB>`, then email will not work.

The benefits of centralization have thus far outweighed the drawbacks. But this is changing. Users today are much more concerned about privacy, and reachability is increasingly threatened by natural disasters, political repression, cyberattacks, and human error.

CityMesh provides an answer to this problem. By leveraging the density of wireless access points (APs) in cities, we can construct a decentralized mesh network for peer-to-peer communication. `<userA>` would send email directly to `<userB>`'s mailbox at `<buildingB>`, rather than `gmail.com`. `<userB>` can query `<buildingB>` remotely, enabling mobility. We assume that user addresses are exchanged out-of-band, obviating the need for a location service.

To test our unicast routing protocol, we built a discrete-event network simulator using SimPy. However, we make several simplifying assumptions. In Chapter 3, I show that our simulator nevertheless achieves 67.4% correlation with real data that we collected.

Additionally, unicast is not sufficient for many applications, such as group chat and social media. Group communication requires multicast. In Chapter 4, I compose our unicast primitive into multicast trees using three different topologies: star, minimum spanning tree (MST), and Steiner tree. Conventional wisdom suggests that Steiner would outperform MST, but surprisingly I found that Steiner performs worse than MST on average.

I begin with a discussion of related work in Chapter 2.

Chapter 2

Related Work

This thesis touches on three broad topics: peer-to-peer, group communication, and city-scale mesh networks.

2.1 Peer-to-Peer

Much has been written about peer-to-peer applications such as Napster, Gnutella, FastTrack, and BitTorrent. A survey can be found in [3]; interestingly, Skype used to be peer-to-peer but has since moved to the cloud. CityMesh differs from these applications because it is not an overlay on top of IP.

2.2 Group Communication

Multicast is also a well-studied problem. Despite proposals like SRM [4], RMTP [5], MTCP [6], and PGMCC [7], network-layer multicast has not been embraced. As a result, application-layer multicast protocols have been developed, such as Narada [8], Yoid [9], HMTP [10], NICE [11], Scribe [12], and CAN Multicast [13]. A survey can be found in [14].These would be interesting to implement in CityMesh; for simplicity, however, I focus on basic tree algorithms.

2.3 City-Scale Mesh Networks

Greedy routing is very scalable, but gets stuck at dead ends. One solution is GPSR [15], which recovers by routing around the perimeter of the local region using a right-hand rule. Geographic Source Routing [16] is the most similar to CityMesh, computing the shortest path through a map.

Roofnet [17] was one of the first mesh networks deployed at city-scale. Like CityMesh, it uses shortest-path source routing, but its goal is to route to Internet gateways.

Chapter 3

Simulator

Ideally we would build CityMesh using the real APs in a city, but since we do not have access to these APs, we turn to simulation. Existing discrete-event network simulators such as ns-3 [18] are more sophisticated than we need for our initial exploration, so we built our own simple simulator using SimPy. In particular, we make the following simplifying assumptions:

- The number of APs in a building is proportional to the building footprint area. This ignores building height; our model is 2D.
- Every building has at least one AP.
- APs are spread uniformly within each building footprint.
- Two APs can communicate bidirectionally if and only if they are within a global constant distance. This ignores wireless effects.

We download building footprints from OpenStreetMap (OSM) [19]. For each building footprint, we multiply the area by the global parameter `ap_density` to get the number of APs in the building, and place them randomly within the footprint. For example, if `ap_density` = 0.005 APs/m^2, then a building with 1,000 square meters will be assigned 5 APs. All building are assigned at least one AP. Each AP can communicate with other APs within `tx_range` meters.

3.1 Measurement Setup

We measured real APs along several bike routes in Boston and Cambridge, and compared the real results with the simulated results. Data was collected using a WiFi Pineapple [20] in a backpack while biking, as shown in Figure 3.1. The bike routes are shown in Figures 3.2–3.4; we focus on the `downtown` dataset (Fig. 3.2) because it is the largest and most heterogeneous. At each sampling point, a GPS device recorded the location, and the Pineapple recorded a "fingerprint" of MAC addresses, similar to VTrack [21].

Figure 3.1: Measurement setup.

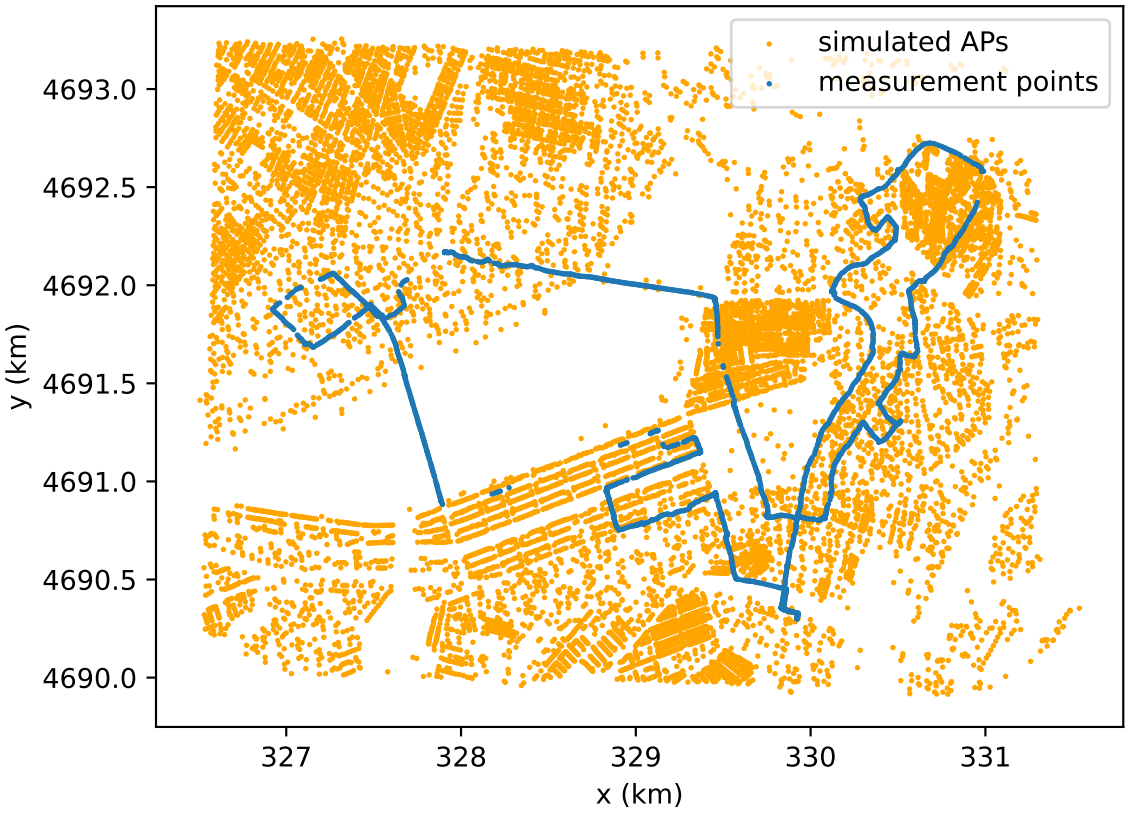


Figure 3.2: Downtown route.

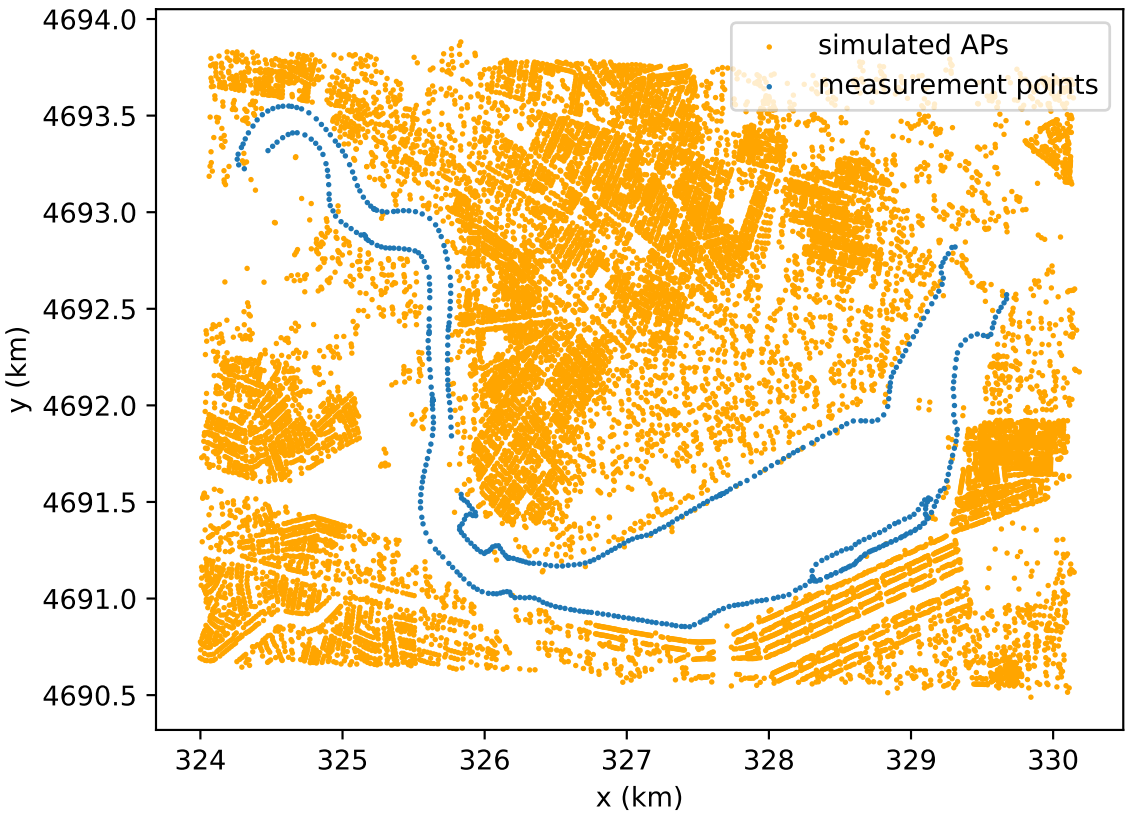


Figure 3.3: River route.

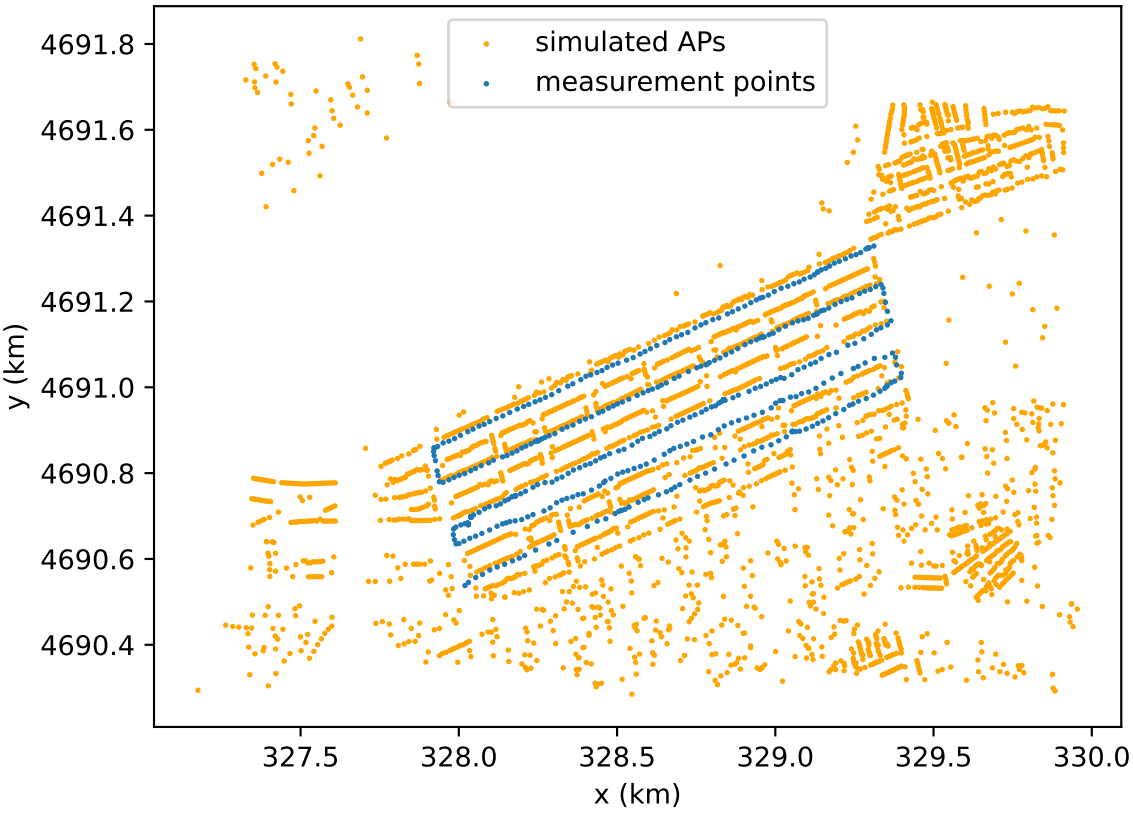


Figure 3.4: Cityblock route.

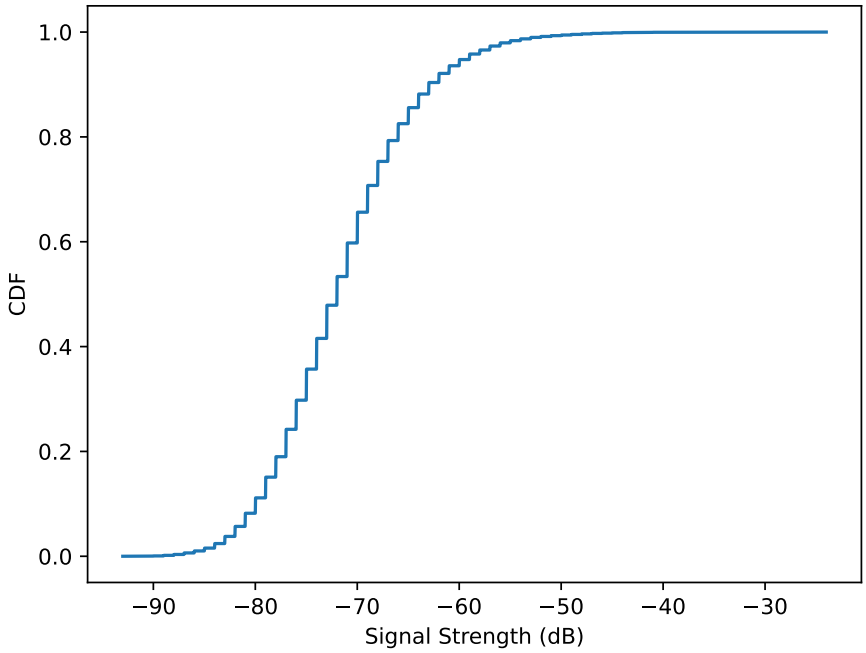


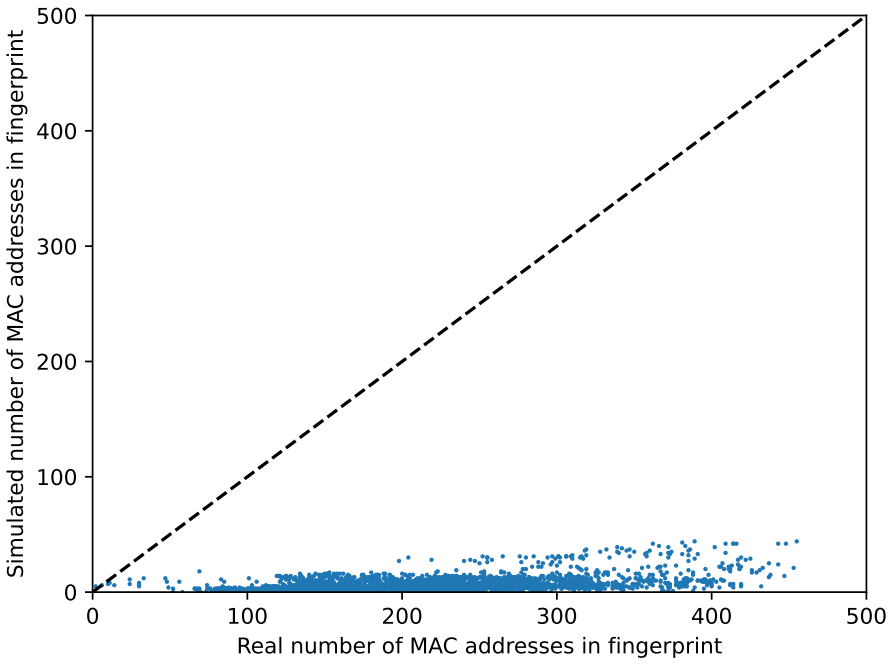
Figure 3.5: Distribution of measured signal strengths.

3.2 Results

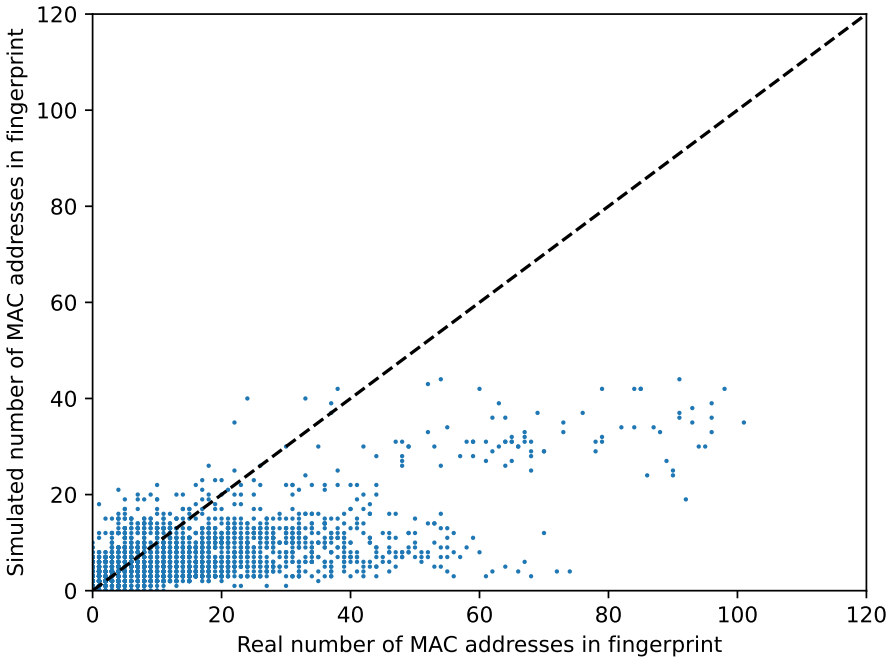
At first, the correlation is poor. For example, with `ap_density` = 0.001 APs/m^2 and `tx_range` = 55 m, the correlation is 43.0% (Fig. 3.6a). However, most of the real APs have weak signal strengths; more than 90% of the measured signal strengths were below −60 dB (Fig. 3.5). Instead of improving the simulator, we can improve our interpretation of reality!

Applying a minimum strength to the real data significantly improves the correlation. With `min_strength` = −60 dB, the same parameterization yields a correlation of 67.4% (Fig. 3.6b). This was the highest correlation encountered during a parameter sweep. Therefore, we set `ap_density` and `tx_range` to 0.001 APs/m^2 and 55 m, respectively.

Perhaps even more compelling is the number of MAC addresses as a function of distance traveled (Fig. 3.7). Filtering alters the shape of the real data, resulting in a closer match to the simulated data.

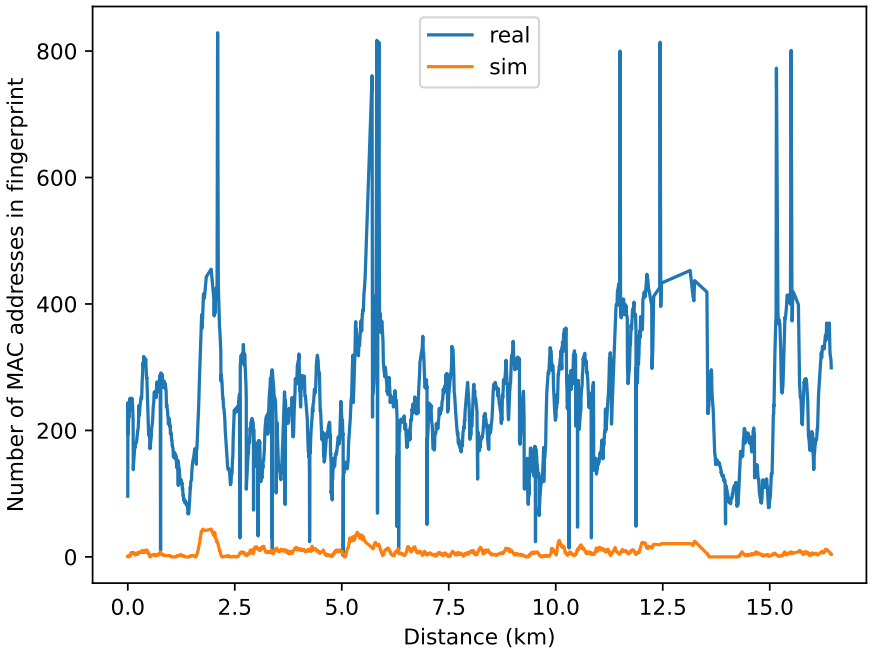


(a) Unfiltered. Correlation = 43.0%.

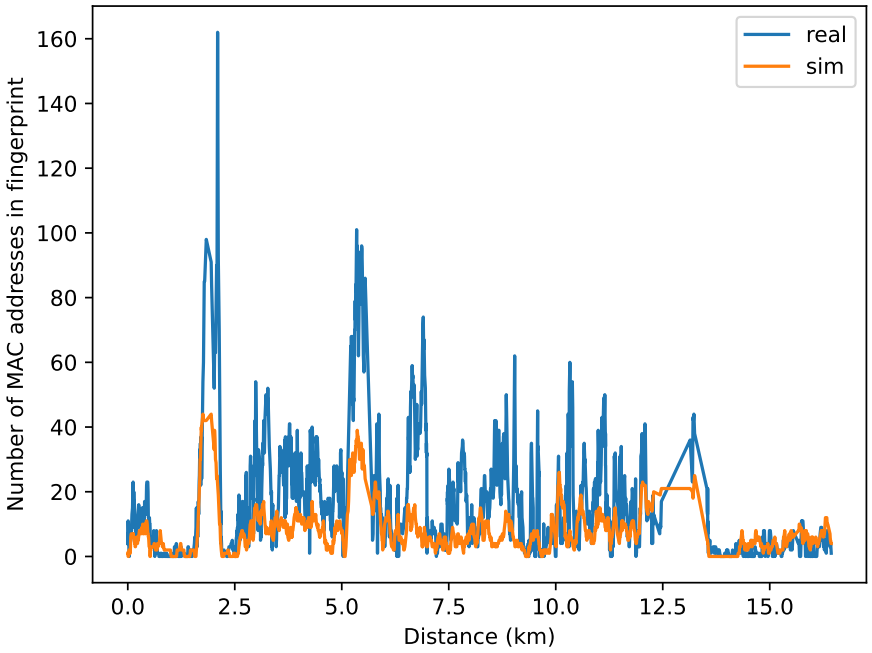


(b) Filtered. Correlation = 67.4%.

Figure 3.6: Real vs. simulated number of MAC addresses in fingerprint.



(a) Unfiltered. Correlation = 43.0%.



(b) Filtered. Correlation = 67.4%.

Figure 3.7: Number of MAC addresses as a function of distance traveled.

Chapter 4

Multicast

Group communication is essential for many applications. Users often send one email to many addresses, one text message to many members of a group chat, or one post to many followers on social media. The key question is how to construct a tree connecting the terminals (source + destinations).

4.1 Star Graph

A naive solution is to unicast from the source for all destinations, forming a star graph. This barely qualifies as a multicast tree; it is included only as a baseline in Figure 4.1.

4.2 Minimum Spanning Tree

I construct a complete graph $G = (V, E)$, where V is the set of terminals, and the weight of each edge $e = (u, v) \in E$ is the cost of the weighted shortest path between u and v in the building graph. The minimum spanning tree of G is the multicast tree.

4.3 Steiner Tree

Steiner tree is similar to minimum spanning tree, except any building can be a Steiner point. According to conventional wisdom, Steiner tree should be better than minimum spanning tree. I use the Mehlhorn algorithm to compute the Steiner tree.

4.4 Experimental Setup

Let k be the number of destinations. For each k in a geometric sequence, I generate 100 random pairs $(s, \{t_1, ..., t_k\})$, and for each pair I generate a star graph, a minimum spanning tree, and a Steiner tree. Each data point in Figures 4.1–4.3 is an average of 100 pairs, with error bars showing ± 1 standard deviation.

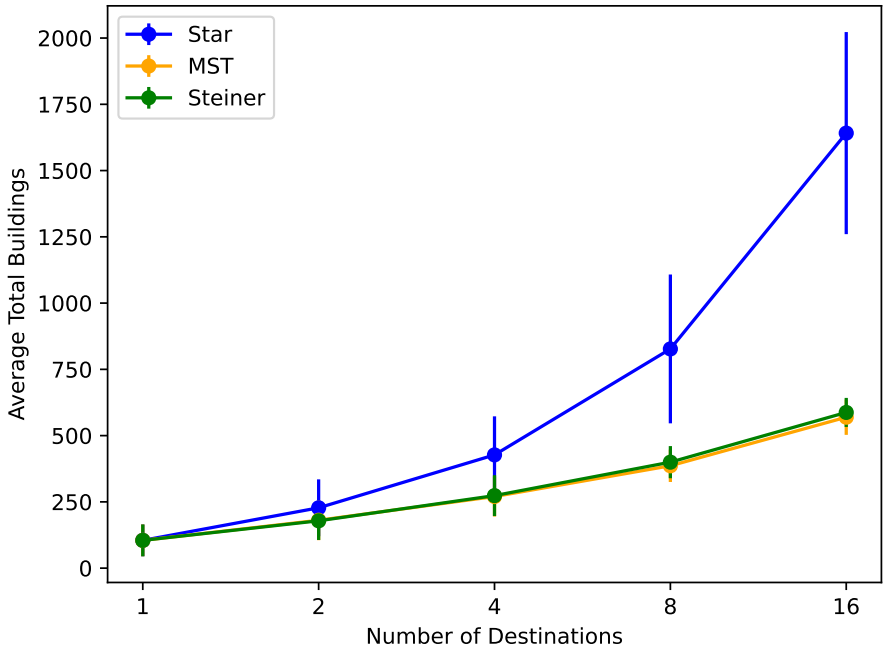


Figure 4.1: Effect of scaling on average total buildings (semi-log).

4.5 Results

As the number of destinations increases, the average total buildings increases (Fig. 4.1). MST and Steiner perform about the same, and much better than star.

Steiner trees are large even for unicast ($k = 1$), and get worse as the number of destinations increases (Fig. 4.2). This is because the Steiner algorithm selects many Steiner points along each route; the more destinations, the more routes, and the more Steiner points. By contrast, minimum spanning trees are small and scale well. This difference is illustrated in Figure 4.4; the Steiner tree in Figure 4.4b is much more dense than the minimum spanning tree in Figure 4.4a for the same 4 destinations.

As the number of destinations increases, the average buildings per route decreases for MST while Steiner remains flat (Fig. 4.3). Shorter route length is not necessarily better; very short routes do not benefit from compression, while very long routes remain long after compression.

There is a tradeoff between the average tree size (Fig. 4.2) and the average buildings per route (Fig. 4.3). As the tree gets more dense, the number of routes (the denominator) increases, but the number of buildings (the numerator) does not change, so the buildings per route decreases. In CityMesh, we prefer small trees, since the tree must be stored in the packet. Therefore, MST is better than Steiner for CityMesh.

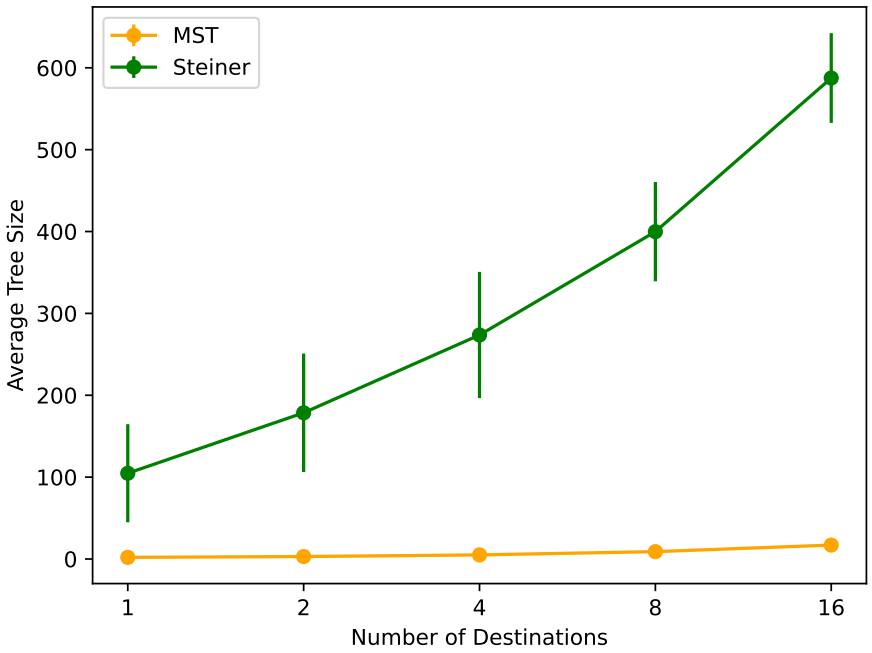


Figure 4.2: Effect of scaling on average tree size (semi-log).

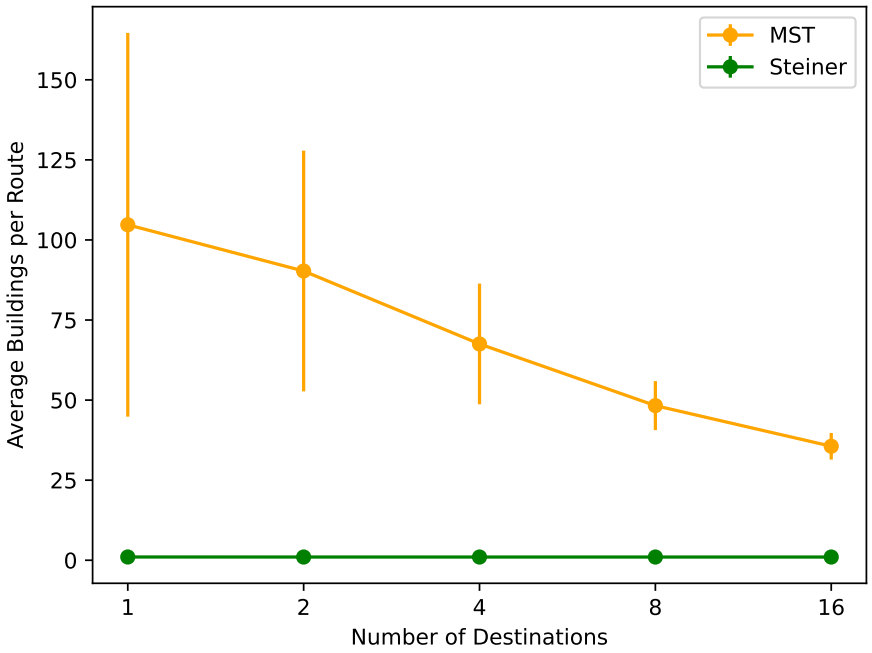
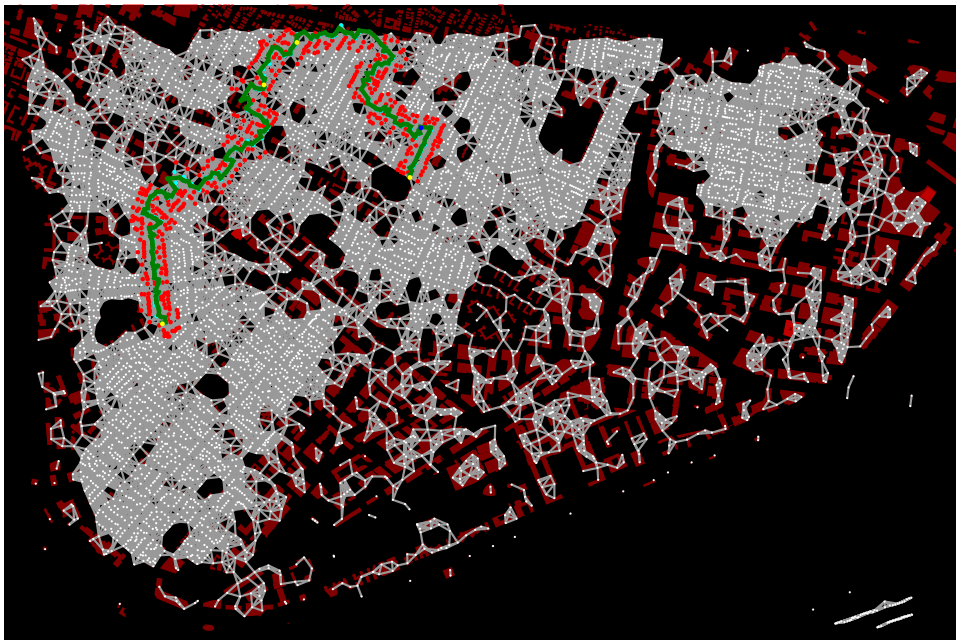
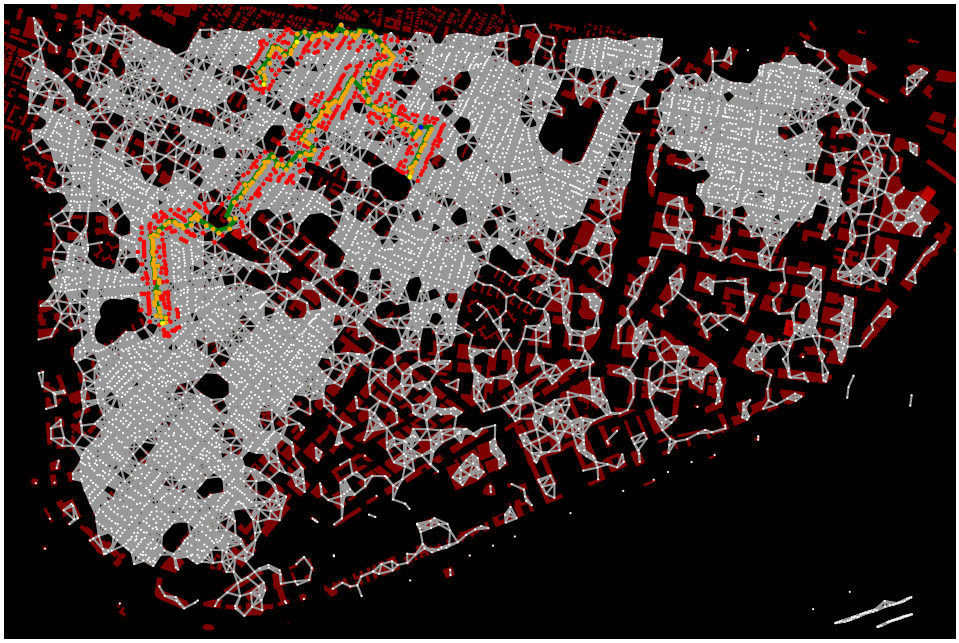


Figure 4.3: Effect of scaling on average buildings per route (semi-log).



(a) Minimum spanning tree.



(b) Steiner tree.

Figure 4.4: Minimum spanning tree and Steiner tree for the same 4 destinations.

Chapter 5

Conclusion

Although CityMesh is a long way from replacing the centralized Internet, our initial steps are encouraging. We made several simplifying assumptions, yet I was able to show that our simulator nevertheless achieves 67.4% correlation with real data that we collected. Then, I composed our unicast primitive into multicast trees using three different topologies, and surprisingly found that Steiner trees perform worse than minimum spanning trees on average.

The takeaway is not that Steiner trees are bad; rather, minimum spanning trees are surprisingly good, and Steiner trees are similar but with higher overhead.

One idea to improve the performance of Steiner trees is to constrain the Steiner points so that many fewer are selected. Another possibility is to penalize the overhead in the edge weights. These remain future work.

We are just beginning, but peer-to-peer group communication in CityMesh promises to enable decentralized multicast applications.

References

[1] B. Buchanan, *Google has 9 products with over 1 billion users - how effectively is it monetizing those hours?* https://01core.substack.com/p/google-has-9-products-with-over-1, 2022.

[2] Kepios, *Global Social Media Statistics*, https://datareportal.com/social-media-users, 2024.

[3] J. Li, "On peer-to-peer (P2P) content delivery," 2008.

[4] S. Floyd, V. Jacobson, C.-G. Liu, S. McCanne, and L. Zhang, "A Reliable Multicast Framework for Light-Weight Sessions and Application Level Framing," *IEEE/ACM Transactions on Networking*, 1997.

[5] J. C. Lin and S. Paul, "RMTP: A Reliable Multicast Transport Protocol," in *INFOCOM*, 1996.

[6] I. Rhee, N. Balaguru, and G. Rouskas, "MTCP: Scalable TCP-like Congestion Control for Reliable Multicast," *Computer Networks*, 2002.

[7] L. Rizzo, "pgmcc: a TCP-friendly single-rate multicast congestion control scheme," in *SIGCOMM*, 2000.

[8] Y. Chu, S. Rao, S. Seshan, and H. Zhang, "Enabling Conferencing Applications on the Internet using an Overlay Muilticast Architecture," in *SIGCOMM*, 2001.

[9] P. Francis, *Yoid: Extending the Internet Multicast Architecture*, https://www.icir.org/yoid/docs/ycHtmlL/htmlRoot.html, 2000.

[10] B. Zhang, S. Jamin, and L. Zhang, "Host Multicast: A Framework for Delivering Multicast to End Users," in *INFOCOM*, 2002.

[11] S. Banerjee, B. Bhattacharjee, and C. Kommareddy, "Scalable application layer multicast," in *SIGCOMM*, 2002.

[12] M. Castro, P. Druschel, A.-M. Kermarrec, and A. Rowstron, "SCRIBE: A large-scale and decentralized application-level multicast infrastructure," *IEEE Journal on Selected Areas in Communications*, 2002.

[13] S. Ratnasamy, P. Francis, M. Handley, R. Karp, and S. Shenker, "A Scalable Content-Addressable Network," in *SIGCOMM*, 2001.

[14] S. Banerjee and B. Bhattacharjee, "A Comparative Study of Application Layer Multicast Protocols," 2002.

[15] B. Karp and H.-T. Kung, "GPSR: Greedy Perimeter Stateless Routing for Wireless Networks," in *MobiCom*, 2000.

[16] C. Lochert, H. Hartenstein, J. Tian, H. Fussler, D. Hermann, and M. Mauve, "A Routing Strategy for Vehicular Ad Hoc Networks in City Environments," in *Intelligent Vehicles Symposium*, 2003.

[17] J. Bicket, D. Aguayo, S. Biswas, and R. Morris, "Architecture and Evaluation of an Unplanned 802.11b Mesh Network," in *MobiCom*, 2005.

[18] ns-3, *ns-3 Network Simulator*, https://www.nsnam.org/, 2024.

[19] OpenStreetMap, *OpenStreetMap provides map data for thousands of websites, mobile apps, and hardware devices*, https://www.openstreetmap.org/about, 2024.

[20] Hak5, *WiFi Pineapple*, https://shop.hak5.org/products/wifi-pineapple, 2024.

[21] A. Thiagarajan, L. Ravindranath, K. LaCurts, S. Madden, H. Balakrishnan, S. Toledo, and J. Eriksson, "VTrack: Accurate, Energy-Aware Road Traffic Delay Estimation Using Mobile Phones," in *SenSys*, 2009.