Scalable Routing in a City-Scale Wi-Fi Network for Disaster Recovery

Ziqian Liu* ✉
MIT CSAIL, Cambridge, MA, USA

Om Chabra* ✉
MIT CSAIL, Cambridge, MA, USA

James Lynch ✉
MIT CSAIL, Cambridge, MA, USA

Aaron Martin ✉
MIT CSAIL, Cambridge, MA, USA

Chenning Li ✉
MIT CSAIL, Cambridge, MA, USA

Hari Balakrishnan ✉
MIT CSAIL, Cambridge, MA, USA

Abstract

This paper presents CityMesh, a city-scale decentralized mesh network designed for disaster recovery and emergency scenarios. When wide-area Internet connectivity is unavailable or severely degraded, CityMesh leverages both static access points and mobile devices equipped with Wi-Fi to provide intra-city connectivity and reach opportunistic gateways to the Internet (e.g., via satellite links). The main contribution of this paper is a scalable routing protocol that supports millions of devices, addressing a long-standing limitation of wireless mesh and mobile ad hoc networks. Unlike prior approaches, CityMesh exploits rich building-location and building-geometry data from widely available city maps to guide route computation, improving packet delivery while significantly reducing transmission overhead. Simulation results from 70 cities show that CityMesh improves packet delivery rates by 88% over WEAVE (a state-of-the-art geographic routing protocol). A campus-scale deployment of 300 Wi-Fi devices across 31 buildings shows the practical deployability of CityMesh. These results demonstrate the promise of map-aware routing as a foundation for scalable, resilient city-wide Wi-Fi networks.

2012 ACM Subject Classification Networks → Network protocols; Networks → Ad hoc networks

Keywords and phrases mesh networking, disaster recovery, geographic routing, scalability, Wi-Fi

Digital Object Identifier 10.4230/OASIcs.NINeS.2026.10

Acknowledgements We thank Manya Ghobadi for participating in several discussions on this project. We thank John Cheng, Tejas Patel, and the NINeS reviewers for their valuable feedback. We thank Snehadeep Gayen for assisting with data analysis and visualization. This work was supported by DARPA Contract HR001120C0191.

1 Introduction

This paper presents CityMesh, a wireless routing system that aims to scale to millions of Wi-Fi devices across an urban area. Traditional mesh and ad hoc routing protocols use techniques such as link-state flooding or route advertisements, which are unscalable in dense, city-wide deployments. In contrast, CityMesh exploits accurate, high-resolution geospatial building maps, now widely available and continuously updated by commercial and open-source providers such as OpenStreetMap (OSM)

*Equal contribution.

(`https://www.openstreetmap.org/`) [35]. By reframing packet routing as a problem of computing routes between buildings rather than individual devices, we enable compact, pre-computed routing tables that can be stored on commodity Wi-Fi devices and used online with low overhead.

Before delving into how CityMesh achieves these results, it is worth addressing two questions readers may naturally ask: (1) *why does this matter?* and (2) *hasn't this problem already been solved?* We address the second question in §2, noting here that no prior approach has convincingly scaled beyond a few thousand nodes. The first question—why this matters—is central to the motivation for CityMesh.

As noted in our prior HotNets paper [68], the Internet has become increasingly centralized at both the network and application layers, driven by ISP consolidation, massive physical co-location, and the dominance of cloud computing as the application deployment model. This centralization, while efficient under normal conditions, creates acute vulnerabilities to disasters and attacks, both digital and physical. History has repeatedly shown that disasters can severely degrade or entirely sever wide-area Internet connectivity: entire regions have lost access to Internet providers [10], while in other cases only a small minority of nodes—e.g., those with access to satellite backhaul—remained connected to the Internet.

At the same time, however, many disasters leave the urban physical substrate largely intact. Power often remains available in much of the city, and the majority of buildings retain structural integrity even under stress from cyberattacks [4, 43], mass-casualty events [23, 17, 88], hurricanes [103, 13, 14], flooding [67], or moderate earthquakes [71]. In these circumstances, the primary challenge is not the lack of functioning devices and Wi-Fi routers, but rather the absence of reliable access to the wide-area Internet.

If it were possible to provide even limited communication within a region during such disruptions, a wide range of critical applications could continue to function: status updates and messaging among residents, coordination of emergency and safety services, lightweight financial transactions and local commerce, navigation, and opportunistic connectivity to nodes with functioning Internet links. Although the available bandwidth would be far lower than under normal conditions, it could be sufficient to sustain essential services during times of duress. This vision of decentralized fallback networks (DFNs) in which local connectivity persists despite the loss of the global Internet was articulated in the recent CityMesh proposal [68].

To turn that vision to reality, we develop a scalable solution for a city-scale DFN that leverages commodity Wi-Fi devices. Unlike prior work, the CityMesh protocol presented in this paper supports not only static Wi-Fi access points (APs), but also mobile devices such as smartphones, increasing the potential reach and resilience of the network. Moreover, while the HotNets paper showed the promise of using maps for routing, their approach suffered from prohibitive inefficiencies: each delivered packet required 15–20$\times$ more transmissions than an optimal unicast path, and even small payloads incurred many hundreds of bytes of per-packet header overhead. These inefficiencies render that design unscalable and unusable. Beyond that, many key details were left open for future work.

To our knowledge, this paper presents the first scalable, end-to-end design for city-scale Wi-Fi disaster recovery networks, addressing the open challenges of routing efficiency, compact forwarding state, and support for both static and mobile Wi-Fi devices.

To achieve scalability, CityMesh takes a radically different approach from prior work: *it eliminates all probe, control, and routing message exchanges between devices*. Instead, it relies on map data, now readily available for most cities worldwide, to guide route computations. CityMesh computes routes along a sequence of buildings from source to destination, requiring only that each device know its own building location. All other context is learned passively by overhearing packet broadcasts during normal operation. This design is feasible because the vast majority of Wi-Fi devices reside in

or near buildings—from fixed access points to the smartphones and other devices people use indoors.[1]

We develop algorithms to (1) compute efficient inter-building paths; (2) add redundancy to those paths using a geofence concept where nodes within the geofence rebroadcast; (3) compress those paths into compact waypoints that capture only essential turns and deviations from straight-line segments; (4) construct a grid-based addressing scheme for buildings in a city to enable small forwarding tables, which can be precomputed on a central server and distributed to the devices; and (5) suppress redundant transmissions using two randomized protocols to avoid the high overheads that plagued prior work. Taken together, these techniques yield a novel, scalable, and practical map-driven routing architecture. We also develop a new map-aware Steiner-point relay placement algorithm to compute the minimum number and placement of Wi-Fi devices needed to connect disconnected clusters when full connectivity is not achieved in a city.

We have implemented CityMesh both in the ns-3 simulator and on a campus-scale 300-node testbed, and have simulated its performance across 70 cities in 24 countries. Our results show that CityMesh achieves high delivery rates and low overhead at scales far beyond what has previously been demonstrated for Wi-Fi mesh or ad hoc routing.

This paper makes the following key contributions:

- **High deliverability at scale.** CityMesh achieves packet delivery rates 88% higher than WEAVE, the best geographic routing protocol among the ones we simulated, demonstrating the effectiveness of map-driven path computation in urban environments.
- **Low overhead.** By eliminating control traffic and suppressing redundant transmissions, CityMesh reduces data network overhead by 6×-1000× compared to state-of-the-art mesh routing schemes.
- **Compact routing state.** Each device maintains only a few kilobytes of routing state, even for city-wide networks with millions of nodes, making the design practical for commodity Wi-Fi hardware. For example, CityMesh requires only 352 entries for Los Angeles with 1,084,482 buildings.
- **Deployment feasibility.** Our analysis of 70 cities shows that the median number of additional Wi-Fi devices needed to ensure full city-wide connectivity is only 1.26% of the number already present.
- **Testbed validation.** We implement CityMesh on a 300-node campus testbed, demonstrating real-world feasibility and validating behavior in practice. Delivery rates approach an idealized flooding protocol.

This work does not raise ethical issues.

2 Background & Related Work

Network failures. As wide area networks (WANs) become increasingly centralized, failures have become more frequent and more impactful. These failures stem from diverse causes, including natural disasters [13, 103, 14], malicious attacks [43, 4], and, most commonly, human error [75, 37, 39]. For example, in 2024, Hurricane Helene left many Florida residents with restored power but no Internet connectivity [28]. In 2020, a terrorist bombing in Nashville destroyed an AT&T transmission facility, disrupting wireless and wired networks across Tennessee, Kentucky, and Alabama. The outage disabled cell service, 911 calls, and grounded flights at Nashville airport [114]. Other incidents include fiber cuts isolating entire districts [109], misconfigured BGP updates that blackholed traffic across cities, and even nationwide outages [107]. Notably, in nearly all of these cases, electricity remained available and most buildings remained intact, but Internet connectivity failed.

[1] At present, our design does not support users operating far from any building.

Static mesh networks [9, 24, 2, 20, 12] focus on throughput-optimized routing using metrics like expected transmission count/time (ETX/ETT) and topological routing (e.g., Srcr in Roofnet, which floods link-state metrics). While effective at modest scales, none demonstrate scalability beyond a few thousand nodes.

Mobile ad hoc networks (MANETs) are infrastructure-less networks where mobile nodes act as both hosts and routers. Decades of work on MANET routing [50, 7, 48, 52, 112, 18, 76, 99, 6, 22, 69, 42] falls into two classes: *topological routing*, which relies on control messaging, and *geographic routing*, which uses out-of-band location information [90].

In topological protocols, nodes know only addresses and depend on control packets to discover paths. Proactive schemes (e.g., DSDV, OLSR, B.A.T.M.A.N. [83, 106, 8, 51, 41, 27, 76, 73, 36]) maintain routing tables via periodic flooding, incurring high overhead. Reactive schemes (e.g., AODV, DSR, TORA, DYMO [18, 80, 89, 82, 81, 115, 84, 32, 47, 3, 108]) discover routes on demand, but frequent churn produces bursts of control traffic that approach proactive costs. A canonical scheme, DYMO (AODVv2), explicitly warns that it is best suitable only for sparse traffic [84]. Hybrid protocols [40, 33, 70, 46, 118, 31, 111, 110] combine proactive flooding within local clusters and reactive routing across clusters. While reducing latency compared to purely reactive schemes, they still depend on significant control traffic and inherit the same scalability limitations.

Geographic routing reduces control overhead by forwarding based on node positions, typically obtained from a location sensor such as GPS. Destination identifiers are converted to destination locations using services such as the Grid Location Service (GLS) [62]. Protocols such as GPSR [48], GDSTR [58], and WEAVE [53] forward packets greedily, switching to recovery mechanisms to route around voids. Most protocols exchange only one-hop neighbor state, and some avoid periodic control traffic altogether [48, 7, 55, 54, 58, 59, 11, 96, 93, 56, 53].

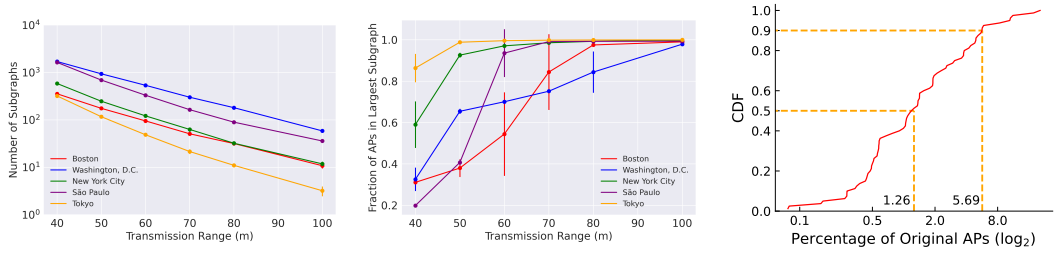
While lightweight in theory, geographic routing is fragile in practice especially in urban and indoor settings: commodity Wi-Fi localization errors (≈10-15 m indoors) significantly degrade delivery even in small networks of 100 nodes [91], and compensating for inaccuracies requires extra control packets [85], undermining efficiency. Advanced designs like WEAVE perform better in city-like topologies but still assume accurate position estimates. We compare CityMesh with WEAVE and GPSR in §5.

Vehicular ad hoc networks (VANETs) exhibit rapid topology changes and suffer from localization errors that impair geographic routing. Variants such as AGPSR, MM-GPSR, GPSR-L, AGF, and CBF [95, 117, 86, 74, 25] adapt MANET algorithms to vehicles (e.g., GPSR-L uses neighbor lifetimes to account for movement). Road map–based protocols like GSR, GPCR, GPSRJ+, A-STAR, and GyTAR [65, 66, 57, 92, 44, 45] instead route between street segments. By contrast, CityMesh uses building geometry, not roads, to construct scalable paths aligned with where Wi-Fi devices are most often located.

In summary, prior schemes on static meshes, MANETs, and VANETs either rely on heavy control messaging, assume accurate position information, or are limited to small-scale deployments. None of these approaches demonstrate scalability to millions of nodes, nor do they provide a deployable solution for disaster recovery. CityMesh introduces a map-driven routing architecture that eliminates control-plane overhead, maintains compact state, and delivers resilient connectivity at city scale.

Integration with other backhaul technologies. Many technologies, including satellites, mobile platforms, and LoRa can provide backhaul "sinks," while CityMesh focuses on routing traffic via Wi-Fi devices to whichever sinks are available, adapting software on existing hardware rather than requiring new infrastructure.

Advances in satellite communication including Starlink [101], low-cost IoT constellations [94, 98, 97], and direct-to-cell links [38] make satellites a promising fallback when terrestrial networks fail. However, city-wide coverage is impractical due to equipment and subscription costs [78], spectrum



(a) *Number of subgraphs vs. Wi-Fi transmission range, by city.*

(b) *Fraction of APs in largest connected subgraph vs. range, by city.*

(c) *CDF of percentage of additional APs (in log2 scale) needed to achieve full connectivity across 70 cities.*

Figure 1 *Impact of transmission range on AP connectivity (left, middle) and additional relay devices required for full city-wide connectivity (right).*

restrictions [94, 19], and usage limits [104]. Mobile platforms such as Cells-on-Wheels, drones, balloons, blimps, and helicopters can provide coverage [105, 72, 5, 113, 1, 61], but they are costly and slow to deploy. LoRa offers long range, but is very low-bandwidth, requires specialized hardware, and can cause significant interference at scale [26, 60].

3 Assumptions and Feasibility

The design of CityMesh is guided by two goals: (1) scalability to millions of Wi-Fi devices in a city, and (2) practical deployability on commodity hardware without specialized equipment. To achieve these goals, CityMesh eliminates traditional control-plane mechanisms (flooding, periodic probes, advertisements) and instead leverages widely available urban building maps to guide packet routing. This section describes the design assumptions, the building-based abstraction, and the components of the system.

3.1 Assumptions

We make the following assumptions:

- Wi-Fi ubiquity in and near buildings. The vast majority of access points (APs) and user devices are located in or near buildings. We show in §3.2 that existing commercial, residential, and public Wi-Fi distribution systems create an expansive network that, when joined together, provides robust, continuous coverage across most cities.
- Urban map availability. High-resolution maps of building footprints are now available from commercial and open sources (e.g., OSM), and are updated regularly.
- Partial backhaul connectivity. During disruptions, only a subset of devices (e.g., satellite-linked gateways) may retain Internet access. These serve as sinks for outbound traffic. If no such devices exist, then connectivity is only provided between devices within a city.
- Low-bandwidth, best-effort connectivity. CityMesh does not guarantee full Internet performance but aims to sustain essential low-bandwidth services under duress.

CityMesh is best suited for applications with low bandwidth requirements that tolerate high latency such as messaging and coordination, location-based status updates, emergency alerts, and local financial transactions. These are all useful applications in emergency and disaster scenarios. CityMesh requires only software changes to Wi-Fi access points and rollout of an application (or better still, software in the operating system similar to contact tracing software) for mobile devices. Access points participating in CityMesh are operated by both end residential users, as well as organizations

and campuses. CityMesh is implemented as a service within OpenWRT [79], a Linux-based OS already running on many commercially-produced access points. Users then opt-in to CityMesh with only software or settings changes on their existing hardware.

Users interact with CityMesh using their mobile devices. As the locations of these devices are not fixed, users designate stationary nodes in the network to serve as their inboxes in order to reliably receive messages from other users and services. We assume that they share the locations and unique identities of these nodes out of band or prior to an Internet outage. Periodically, each user's mobile devices retrieve new messages from their inbox(es), while more time-sensitive messages can be pushed from the inbox node to the user's current location [68].

3.2 Feasibility of CityMesh

To assess the feasibility of CityMesh, we simulate city-wide deployments using real-world building maps from OpenStreetMap [35]. The key question is whether existing access point (AP) density is sufficient to support large-scale connectivity. We model APs by uniformly distributing them within building footprints at a density of one AP per 200 m^2, with at least one AP per building. This estimate is conservative, given the prevalence of dense commercial Wi-Fi systems and residential complexes with per-unit APs. For each city, we then construct a connectivity graph by linking APs whose separation falls within a specified transmission range.

For a mesh network built from existing access points, the goal is to maximize the size of the largest connected component and minimize isolated "islands" of connectivity. Our feasibility results (Figure 1) show that, under reasonable assumptions of AP density and transmission range, the majority of APs are in one or two large connected subgraphs, though the extent varies by region. This highlights the inherent resilience of CityMesh: even without mobile nodes or supplemental infrastructure, city-wide deployments yield highly connected networks.

3.3 Bridging the Gaps

Even with dense Wi-Fi infrastructure, some regions inevitably form isolated islands of connectivity due to highways, rivers, parks, or other barriers. These gaps can be closed by strategically placing a small number of additional relay devices. In prior work, this wireless relay placement problem has been framed as a Steiner point insertion problem, where relays may only be placed within mutual transmission range and the objective is to minimize the number of added nodes [16, 64, 63, 15].

To make this computation practical at city scale, we develop a new algorithm by extending prior ideas with urban road routing and convex hull optimization, yielding significant speedups while ensuring realistic placements (e.g., no devices in water). The details are in Appendix A.

Applying our algorithm across 70 cities, we find that only a small fraction of additional relays is sufficient to achieve full connectivity: the median is 1.26% and the 90th percentile is 5.69%. Figure 1c shows a CDF of the percentage of relays added. The worst case is Venice, which requires 20.48% more relays.

4 Routing via Buildings

We devise a routing protocol that leverages building map information for cities. To compute routes, devices need to know only the current building they are in and the location of buildings nearby. These two pieces of information can be obtained out-of-band, signed and shared across the devices in the building. Notably, devices do not need to know their exact location within a building, but instead approximate their location within a building based on data packet traffic. No routing information needs to be distributed across the network.

Each transmission is a one-shot Wi-Fi link-layer broadcast without a link-layer ACK. Neighboring devices will each make a local decision to forward (broadcast) the packet or not. This approach is well-suited for dense Wi-Fi device deployment common in cities, where pairwise links are lossy, but the neighborhood degree is high.

4.1 Initial Strawman: Source Routing

One approach to using building information to send packets from a source to a destination device is source routing. The source computes a path as a sequence of building IDs (from the map database) and encodes this sequence in the packet header. Intermediate nodes that overhear the packet check whether they are near one of the buildings in the header. If not, they ignore it; if so, they consider forwarding it, suppressing their transmission only if they hear a "better" relay broadcast the packet.

This approach can deliver packets successfully if a suitable building path exists, but to make it practical we must address several challenges:

1. **Path computation.** Given building coordinates and sizes, how can we construct reliable paths? In §4.2, we propose a simple metric that avoids marginal links and yields low-cost paths. Unlike traditional mesh protocols that rely on probe packets and link flooding—both unscalable beyond thousands of nodes—our method requires no control traffic.
2. **Uncertain intra-building connectivity.** With no control messages between devices or knowledge of their precise locations, there is no guarantee that devices in two different buildings on a path are directly connected. To increase the likelihood of finding a working path we introduce *conduits* in §4.3. Comduits are geofences around the chosen paths. Nearby devices will consider broadcasting an overheard packet only if they fall within the conduit, improving reliability without link-layer retransmissions.
3. **Header scalability.** Encoding full building sequences does not scale for paths longer than a few hundred meters. In §4.4, we compress paths into a sequence of waypoint buildings that mark only turns or deviations, yielding compact headers.
4. **Global state burden.** Source routing requires each device to maintain the full map and compute every path, which may be infeasible on some devices. In §4.5, we present a grid-based addressing scheme and compact per-device routing tables, precomputed centrally and distributed to devices, eliminating the need for global state. This method also obviates the need to encode a waypoint sequence in the packet header and replaces source routing.
5. **Excess rebroadcasts.** Multiple devices in or near a given building may rebroadcast redundantly. In §4.6, we design suppression algorithms for both inter-building and intra-building forwarding that sharply reduce overhead.

4.2 Building Graph and Path Selection

From the building map, we construct a building graph $G = (V, E)$ where each vertex $b_i \in V$ represents a building with a unique ID. An edge $(b_i, b_j) \in E$ exists if the buildings are within a nominal Wi-Fi range (e.g., 100 m). Not all such edges are usable in practice, and some longer links may occasionally be feasible, but we conservatively assume communication only along these nominal edges.

Importantly, the graph G encodes only buildings, not actual Wi-Fi devices. We assume that most buildings host Wi-Fi access points (and often mobile devices), though this may not always hold. To compute usable paths (via Dijkstra's or similar algorithms), we assign edge weights that capture the likelihood of successful packet delivery between buildings.

Because connectivity decreases with distance, a natural edge cost is d^k where d is the closest distance between two buildings and $k \geq 1$ is a hyperparameter. Larger k values favor routes composed of shorter hops, while smaller values bias toward longer direct links. For three buildings A, B, C

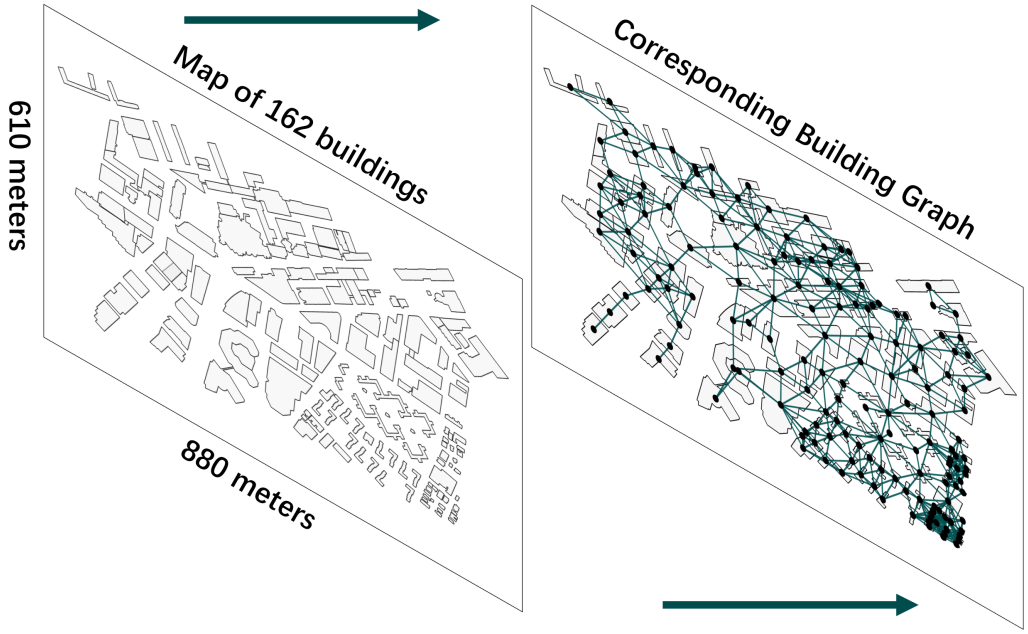


Figure 2 *Converting a transformed map to building graph.*

with pairwise distances $d_{AB} \leq d_{BC} \leq d_{AC}, d_{AC}^k - (d_{AB}^k + d_{BC}^k)$ increases with k: as k grows, indirect paths via B become increasingly preferable. For $k = 1$, the triangle inequality ensures direct links are always chosen; for $k = 2$, indirect paths are preferred when B lies within the circle of diameter AC, and as $k \to \infty$, the selected paths converge to those in the minimum spanning tree (MST). In this limit, the chosen path can be viewed as the *safest path*, maximizing packet delivery probability, but all traffic concentrates on the MST links (we prove this fact in Appendix B).

Our goal is not to model wireless connectivity precisely, nor to assume signal strength decays as $1/d^k$. Rather, we use the d^k weighting scheme simply to bias routing toward shorter inter-building links. In §5, we evaluate this strategy under varying loss rates that capture differences in materials and obstacles between buildings, even when they are geographically close.

We empirically tested the performance of different values of k for different cities, finding that $k = 10$ provides a reasonable balance between reliable delivery and avoiding traffic concentration on a small fraction of available edges (building-to-building wireless links).

4.3 Conduits to Improve Delivery

A limitation of source routing is that it assumes a working Wi-Fi link exists between every pair of successive buildings on a computed path. In practice, devices may be unevenly distributed within buildings or absent altogether, so some links may fail. To improve delivery, we extend forwarding (rebroadcasting) packets beyond only the nodes on the exact path by allowing nearby buildings within a bounded distance to also rebroadcast.

We implement this idea with *conduits*. For two buildings b_1 and b_2, we draw a rectangle of width W centered on the line segment connecting their centroids (Figure 3). The conduit is the set of all buildings whose centroids lie within $W/2$ of this line; W is the *conduit width*.

Forwarding with conduits works as follows: when a device receives a packet, it no longer checks only whether its building ID is on the specified path P. Instead, it tests whether its building's centroid lies within any conduit along P. If so, it broadcasts the packet; otherwise, it ignores it. This check

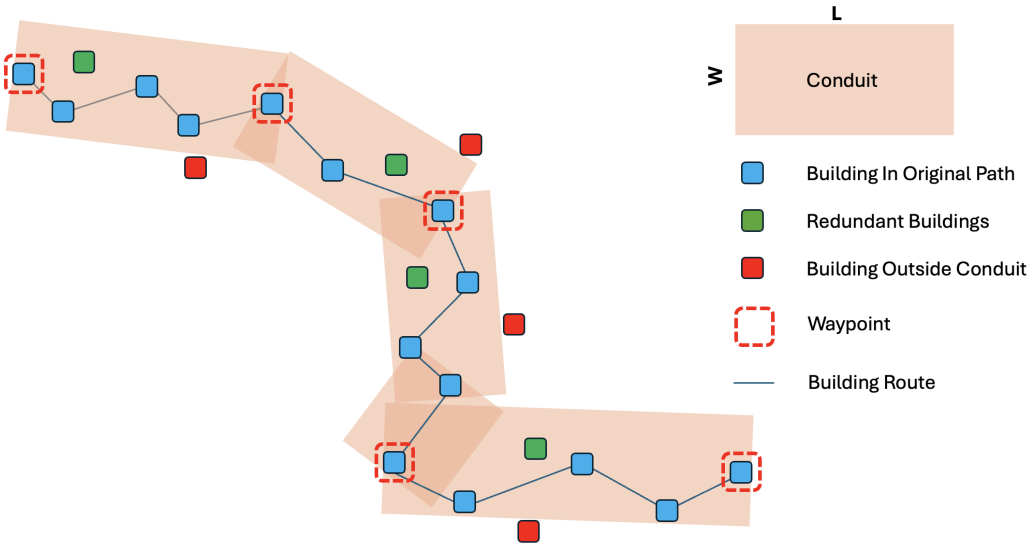


Figure 3 *A "conduit" defines the boundary within which devices should consider rebroadcasting a given packet. The conduit, bounded by waypoints, simplifies the route a packet should follow.*

requires only about ten arithmetic operations (details in Appendix C), making it lightweight for commodity devices.

4.4 Waypoints to Compress Source Routes

A direct source route requires encoding every building along the path P, which does not scale well. To reduce header size, we compress P into a new path P_C that contains only waypoints, defined as the buildings where the traversal direction changes. This compression is lossless: the original path can be reconstructed exactly from the waypoint sequence.

We identify waypoints as follows. Starting from b_0, we extend a conduit to b_i and check whether it contains all intermediate building centroids between b_0 and b_i. If so, we increment i. If not, b_i is marked as a waypoint, and the process restarts with b_i as the new origin. Figure 3 illustrates this procedure.

This scheme makes header size scale with the number of turns on a path rather than its physical length. In practice, if city routes follow long straight segments, the number of waypoints will be small. If that happens, then even multi-kilometer paths may be represented with only a handful of waypoint IDs, yielding compact headers without sacrificing correctness.

4.5 Grid-based Addressing and Routing

Waypoints shrink packet headers by up to 4× in real-city tests (from 40–80 building IDs down to 10–20). However, we find that the header size still grows with path length in practice, and requiring each source to store the full city map and compute minimum-cost paths is not scalable.

To address this problem, we *replace source routing* with a grid-based topological addressing scheme for buildings and construct hierarchical routing tables that support longest-prefix matching, similar to IPv4/IPv6 lookups [100, 102]. A centralized server precomputes routing tables for each building and distributes them to devices in advance of a disaster, eliminating the need for distributed routing computation during network failure.

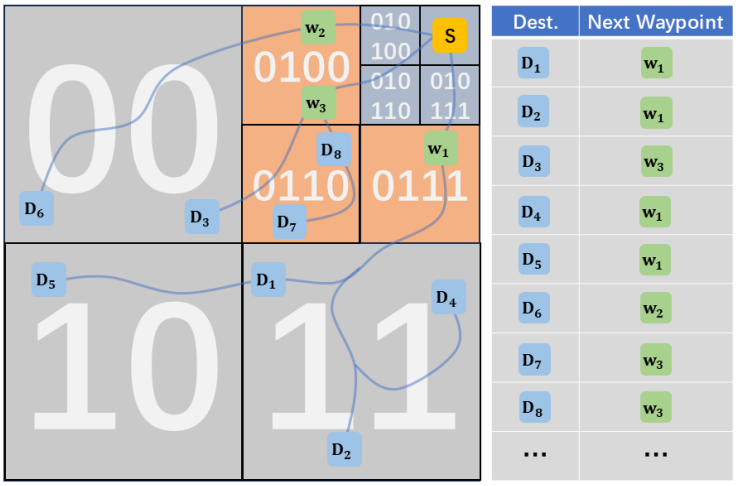


Figure 4 ***Grid-based addressing and routing table.*** *Addresses are assigned based on buildings' location. Source S stores a compressed version of (destination building, next waypoint) pairs in the routing table.*

Since building additions and removals to a city occur infrequently [77], we envision that routing table updates can be performed periodically and infrequently (e.g., once a month). After computing a routing table for a given building, the server cryptographically signs it to enable verification during dissemination. Each wireless AP connects to the server periodically (e.g., once a month) to retrieve the latest routing table. Once this pre-distribution phase is complete, the centralized server does not need to remain reachable during an outage. When mobile devices enter a building, they can verify and obtain the building's routing table from in-building devices (i.e. APs and other mobile devices) that already possess the table.

Each routing table allows a device to determine the **next waypoint** for any given destination. The tables grow only as $O(\log S)$, where S is the area of the city, and are independent of the number of devices. With this scheme, packet headers contain the destination address, the previous waypoint, and the next waypoint. The previous/next waypoint pair enables the conduit check, while the next waypoint also directs forwarding. Crucially, *the header size is now constant*, independent of path length or the number of waypoints.

4.5.1 Grid-based Addressing

We adopt a hierarchical spatial addressing scheme where each building is assigned a `gridID`, a bitstring that compactly encodes its map position. To construct these addresses, the city map is recursively divided into quadrants by halving along the x and y axes (Figure 4). This decomposition continues until the grid size is on the order of the nominal Wi-Fi range ($\approx$100 m). The scheme resembles the Grid Location Service (GLS) [62], but with one key distinction: in CityMesh, addresses are topological identifiers, not arbitrary region names.

Within this smallest grid, which we call a *cell*, we assign a unique ID to each building, so the last few bits of the addresses of each building in the cell are different (but they all share the same prefix). This approach ensures that every cell and every building in the mapped region has a unique address whose length is logarithmic in the size of the region. In addition, neighboring cells (and the buildings in them) share (long) prefixes of their grid addresses.

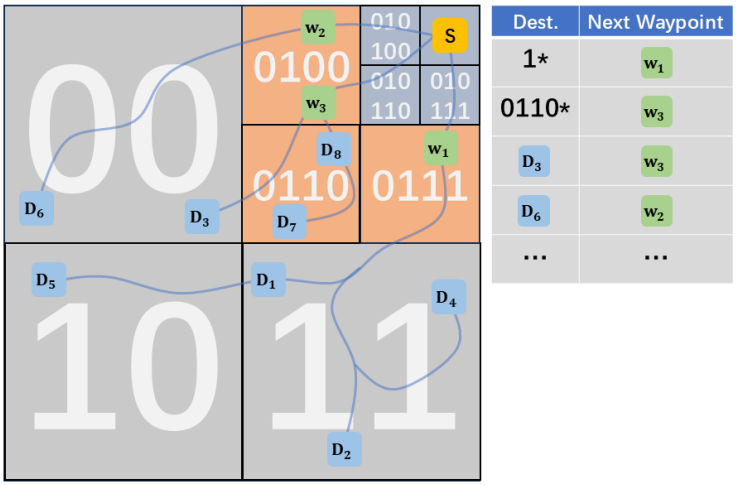


Figure 5 *Condensed routing table of source S. We can combine destination entries by specifying a coarser gridID if all destinations in that coarser grid have the same next waypoint.*

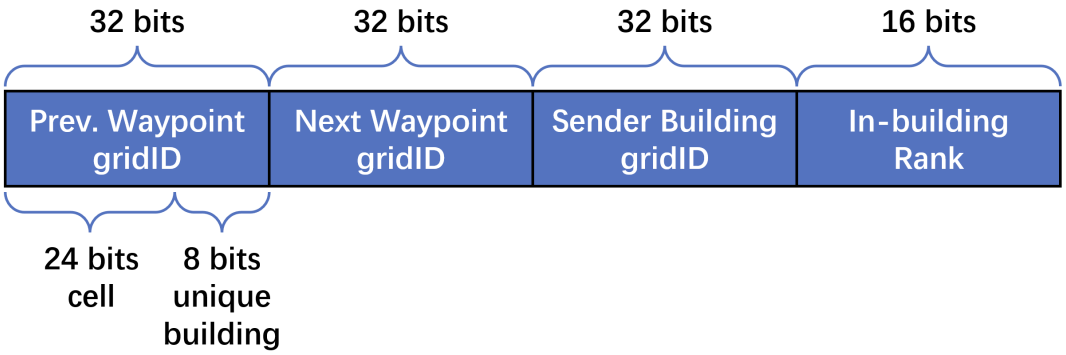


Figure 6 *CityMesh packet header format.*

Figure 5 shows how routing tables benefit from this structure. If all destinations in a larger grid share the same next waypoint, their entries can be collapsed into a single prefix entry, dramatically reducing routing state. Packet headers (Figure 6) then contain the destination address, the previous waypoint, and the next waypoint, ensuring constant-size headers independent of path length.

4.5.2 Route Calculation

Because building placement changes slowly (unlike the mobile devices within them), CityMesh precomputes routes at a central server and periodically distributes routing tables to devices. A plausible approach would be to compute all-pairs shortest paths between every pair of buildings and record the required waypoints. However, this is infeasible: all-pairs minimum-cost path computation on an n-vertex graph takes $O(n^3)$ time, and a city like New York with $\sim 10^6$ buildings makes this approach impractical.

To scale, CityMesh instead computes routes at the cell-to-cell level. For each pair of non-empty cells, it selects a random building from each cell and computes a minimum-cost path between them.

Since most cells contain 10–20 buildings, this reduces the problem size by several orders of magnitude, yielding up to a 1000× speedup (denser regions provide even greater savings). We also exclude each cell without a building from this computation.

The server then generates per-building routing tables from these cell-level paths. For each destination cell, every building has an entry pointing to its next waypoint. If some buildings lack a direct path in the precomputation, their entries are filled by copying from another building in the same cell (which is always possible). In rare cases where a waypoint falls within the same cell, we force an external waypoint by recomputing the path from that waypoint outward.

Importantly, buildings in the same cell may not all share the same waypoint for a given destination. This diversity is desirable, as it provides multiple disjoint routes. The only special case arises when the source and destination lie in the same cell; here, explicit entries are added, since intra-cell delivery requires at most one or two Wi-Fi hops by construction.

4.5.3 Condensing the Routing Table

As constructed, routing tables initially contain one entry per cell, which can reach hundreds of thousands of entries in large cities—too large for commodity Wi-Fi devices. Fortunately, many entries are redundant: multiple destination cells often share the same next waypoint. Leveraging our hierarchical addressing scheme (§4.5), we can apply standard IP forwarding table compression algorithms [87, 49, 116, 21] to aggregate these entries.

Our implementation uses the method of Draves et al. [21], which collapses ranges of destinations with identical next-hop information into single prefix entries. In §5, we show that this reduces table sizes by several orders of magnitude: for example, in Rio de Janeiro (70 km × 34 km, 115k buildings), the maximum routing table size is only 639 entries, requiring just 5.1 KB of storage.

This result shows that compressed routing tables are small enough to fit comfortably on commodity Wi-Fi devices, ensuring CityMesh 's scalability in real-world deployments.

4.6 Suppression of Redundant Transmissions

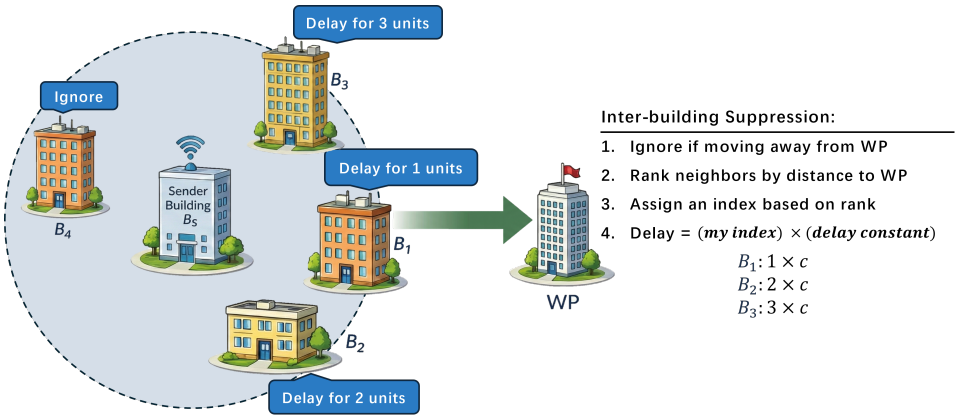
Even with compact headers and routing tables, forwarding in dense urban environments risks excessive rebroadcasting: many devices within or near the same building may rebroadcast the same packet after overhearing it. This problem grows with network density, inflating transmission overhead and increasing contention.

To address this, CityMesh incorporates suppression mechanisms that limit unnecessary rebroadcasts while preserving delivery reliability. We design two complementary techniques: inter-building suppression, which reduces redundant transmissions between neighboring buildings, and in-building suppression, which prevents multiple devices inside the same building from forwarding simultaneously. Together, these mechanisms improve scalability in dense deployments.

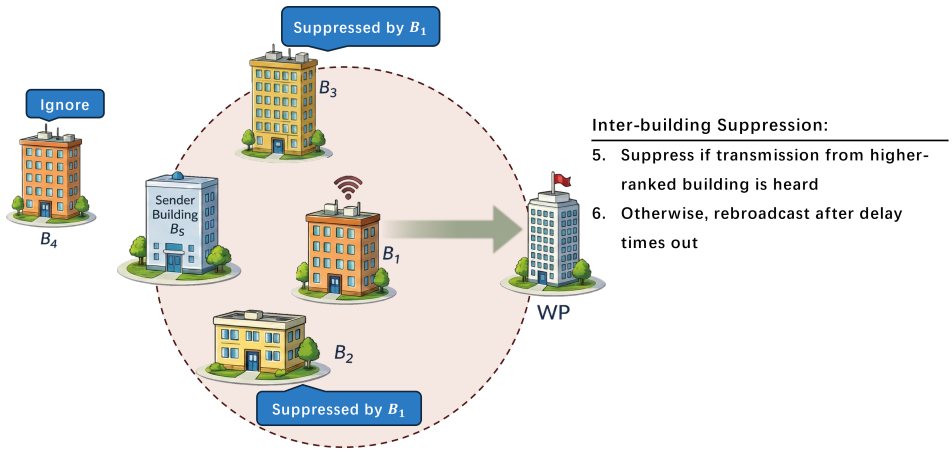
The core idea is simple: when a device receives a packet, it sets a short forwarding timer while listening for retransmissions. If a "better" device (e.g., closer to the next waypoint or destination) rebroadcasts the same packet before the timer expires, the original device suppresses its own transmission. Otherwise, it forwards once the timer fires. Both inter-building and in-building suppression operate in this fully distributed manner, requiring no central coordination.

4.6.1 Inter-building Suppression

The goal of inter-building suppression in CityMesh is to ensure that, among several candidate buildings that hear the same packet, the building that makes the most progress toward the destination is the one that rebroadcasts.



(a) *Delay computation.*



(b) *Transmission suppression.*

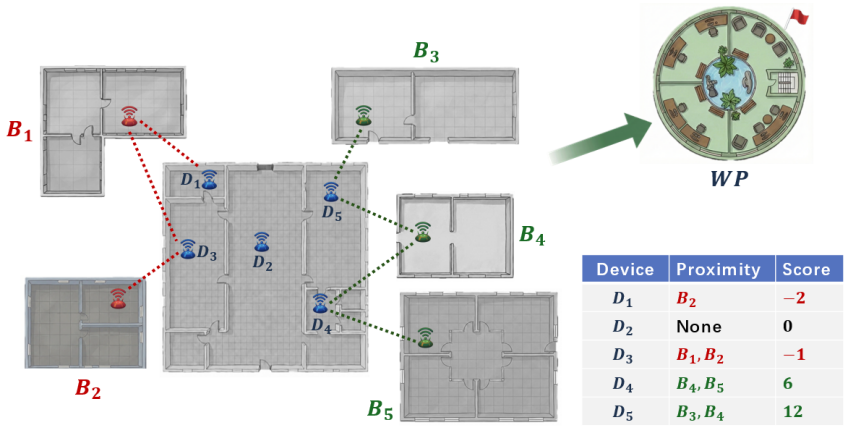
Figure 7 *Inter-building suppression.*

When a building S broadcasts a packet, each receiving device in a neighboring building consults its local building graph[2] to compute a forwarding score to decide if it should rebroadcast the packet. This score is a function of the distance between the building S and the set of S's neighboring buildings, N_S. Each receiver calculates a relative rank among the entries of N_S by sorting each building, $b \in N_S$, by the distance to the waypoint specified in the packet. Note that because the waypoint is encoded as a grid location, this calculation is simple and efficient.

The neighbor closest to the next waypoint is assigned the highest priority, corresponding to a score of 1 and a delay of one unit. Other neighbors are ranked in order of increasing distance, with the furthest building assigned a score equal to $|N_S|$ and a proportionally longer delay. Any building farther from the waypoint than S itself is disqualified from rebroadcasting.

Finally, we scale the delay for this inter-building suppression mechanism by setting the unit delay so that it is always greater than the maximum delay for in-building suppression (described in the subsection below). As a result, the devices within the same building will resolve their suppression first before the packet starts to be suppressed at the inter-building level.

[2]A local version of the graph containing the building's neighbors and their neighbors is sufficient for this task; $\approx$ 100 entries.



Device	Proximity	Score
D_1	B_2	-2
D_2	None	0
D_3	B_1, B_2	-1
D_4	B_4, B_5	6
D_5	B_3, B_4	12

Figure 8 *For in-building suppression, devices record data packets received from other buildings. Devices use the sum of each of the recorded building's scores to compute the delay. For buildings closer to the waypoint (green), there is a positive score based on the position of the recorded building. For buildings further from the waypoint (red), the score is negative.*

4.6.2 In-building Suppression

The challenge with in-building suppression is that the Wi-Fi devices have no specific location information. We considered introducing such a mechanism using prior techniques, but found that they lack sufficient precision and are resource-intensive. Moreover, we show that this is not required. Each Wi-Fi device only knows which building it is within or near.

Instead, CityMesh exploits passive overhearing: by listening to packet broadcasts, each device can estimate the set of neighboring buildings it can hear—without requiring extra beacons or control messages.

The key intuition is as follows: within a building, devices that hear better-positioned neighbors (i.e., those closer to the next waypoint) should be more likely to forward, while devices that mostly hear worse-positioned neighbors should suppress themselves. This ensures that the building contributes only one strong forwarder, minimizing redundant transmissions. This approach allows for easy storage and transmission of this information without requiring any knowledge of the link quality between devices.

To compute its in-building rank R, a device d in building b considers:

- N_b, all neighboring buildings of b, and
- $N_d \subseteq N_b$, the subset of neighbors that d can hear (assuming symmetric delivery for convenience).

The device sorts the elements in N_b in decreasing order of distance to the next waypoint, so the best next building to the next waypoint is at the end of the sorted list. Starting with a rank of 0, for each element in this sorted list that is also in N_d, we add a value of 2^i where i is the index of the element. Then, we decrease R by 1 for each neighboring building that d can reach that is farther away from the next waypoint.

Once R is calculated, the device adds an in-building delay of $c * (1 - sgn(R)\frac{\log_2 |R|}{\log_2 (best\_score)})$, where $sgn(R)$ is 1 if R is positive and -1 if R is negative. With this method, the best node will have a delay of 0, while the worst node will have an in-building delay of $2c$. If two devices have the same delay, we use a randomized jitter to break ties; a device that overhears another with the same rank broadcasting the packet will suppress its own broadcast.

4.7 Failure Recovery

We found in our simulations that the conduit width allows CityMesh to tolerate random building failures without a significant impact on deliverability. However, *large-scale regional failures* that block conduits or eliminate key waypoints can degrade performance. In its current design, CityMesh cannot handle them because as it relies solely on its precomputed routing table.

We speculate that in such cases, while alternative routes may exist, a plausible extension is to incorporate a lightweight recovery mechanism that enables dynamic rerouting under regional failures. Nevertheless, this remains a challenging open problem, as the recovery mechanism must simultaneously avoid false failure detection, operate without control packets and precise geographic information, and minimize both state and memory usage on each device.

We model the recovery problem as two separate problems: (1) failure detection and (2) alternative route finding. Although we do not completely solve the two problems above, we describe here a plausible recovery mechanism.

Failure detection. If all nodes in the building have an in-building suppression score < 0, these nodes know this conduit is a dead end. The nodes will then send a packet back to inform the previous waypoint about the dead end. Upon the reception of the message, the waypoint will send a liveness check to the next waypoint. If no response is received, the waypoint detects a failure in the current conduit, and starts the recovery process.

Alternative route finding. To compute an alternative route, the waypoint can start offsetting the conduit by some angle (i.e., rotates the conduit) until a new waypoint is found. The waypoint then learns the offset angle for the future packets going towards the original waypoint. If no waypoint was found by rotating the conduit, the packets will be sent back to the previous waypoint, and the process repeats.

We have not fully fleshed out the design of this idea, leaving it to future work.

5 City-scale Simulation Results

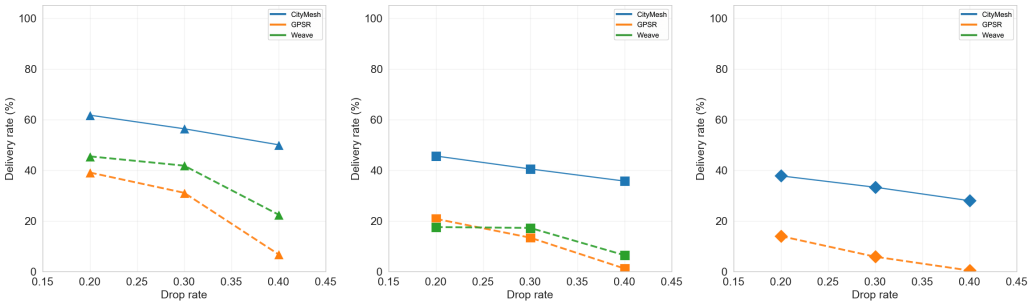
Our city-scale simulation of CityMesh addresses the following questions:

- What is the packet delivery rate of CityMesh under varying wireless conditions in various cities?
- How many total transmissions does CityMesh have, i.e., what is the bandwidth overhead?
- How does the conduit width affect performance?
- How does the exponent k affect performance?
- How much routing information are devices required to store in a real system?

We use ns-3 to model a realistic deployment of wireless devices at the city scale. We report representative results from regions of 70 cities around the world. For each city, we collect the building map from OpenStreetMap (OSM) [35]. We place wireless devices within each building at a density of roughly 200 m^2; we conducted small-scale measurements to determine that typical Wi-Fi densities in cities are usually higher than this number.

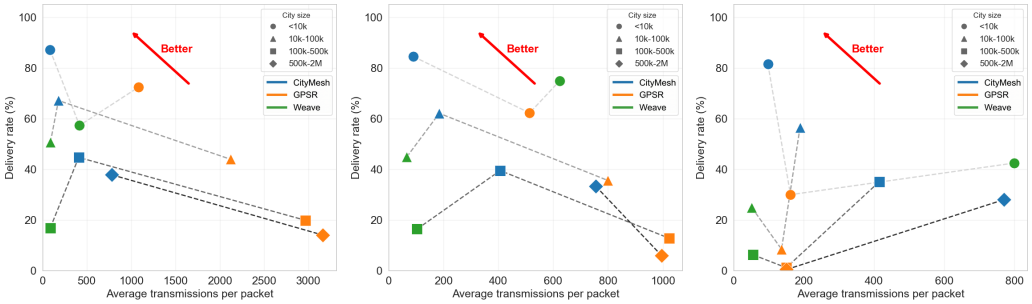
Packet loss model. We used ns-3's built-in distance-based model for a typical 2.4 GHz Wi-Fi setting, which has the property that the packet loss rate is largely zero until a range of about 70 meters, and increases rather sharply to 1 within 10 meters after that. This model does not handle other stochastic losses that we see in practice including in our testbed and in prior works like Roofnet, which see loss rates across the range from 0 to 1. Thus we add atop the distance-based ns-3 model for each link a uni-directional link-layer loss rate picked uniformly at random for each packet between 0 and a maximum of 2ℓ. We show results for multiple values of ℓ between 0.2 and 0.4.

Packet deliverability. In Figure 9, for each city, we pick 100 source-destination pairs at random and send a packet for each pair. We compute the packet delivery rate (fraction of packets delivered) for each value of ℓ. We then bucket cities of similar size together.



(a) *Number of devices: 10k-100k* **(b)** *Number of devices = 100k-500k* **(c)** *Number of devices = 500k-2M*

Figure 9 *Delivery rates as a function of ℓ for three different ranges of the number of devices in a city. Each point us an average of results over several cities, with 70 cities in total. CityMesh outperforms other approaches to a greater degree as the size of the network grows. The WEAVE code did not successfully deliver packets at over 500k devices.*



(a) $\ell = 0.2$ **(b)** $\ell = 0.3$ **(c)** $\ell = 0.4$

Figure 10 *Packet delivery rates in 70 simulated cities against the average number of transmissions. This experiment sends 100 random source–destination packets per city with a per-hop stochastic link loss uniformly averaging to ℓ, and allows GPSR up to 8 link-layer retransmissions (TTL 4096), while CityMesh uses broadcast without link-layer ACKs. Overall, the figure shows CityMesh's robustness to loss and localization error, delivering more packets with comparable or fewer transmissions city-wide.*

While CityMesh uses link-layer broadcasts and cannot benefit from link-layer retransmissions, the GPSR variants can retransmit a packet at the link layer up to 8 times. For GPSR variants, we set a TTL of 4096 hops.

In all sizes of cities, when packet loss rates are lower, CityMesh delivers a similar number of packets to GPSR with accurate location information but significantly outperforms it as packet loss rates rise.

Number of transmissions. Figure 10 compares the number of retransmissions across different routing schemes for 70 cities. As shown in Figure 10(a), when ℓ is 0.2, CityMesh outperforms GPSR in our biggest city with 33% more deliveries, with GPSR having 4.6× the amount of transmissions, with a maximum reduction of 99% transmission. This is due to GPSR's lack of global knowledge causing it to become trapped in its recovery mode. Similarly, we observe that CityMesh induces up to 28× fewer transmissions compared to GPSR with 15-meter location error. When the ℓ is increased to 0.4 (Figure 10(c)), GPSR has lower transmissions compared to CityMesh simply because it fails to deliver more than 80% of the packets.

Conduit width. We repeated the simulations in 15 cities varying the conduit width. We observed that increasing the conduit width provides an additional redundancy. However, as the conduit width increases, the original computed path is less likely to be traversed, and more drops will occur. For all other plots, we use a conduit width of 150 m.

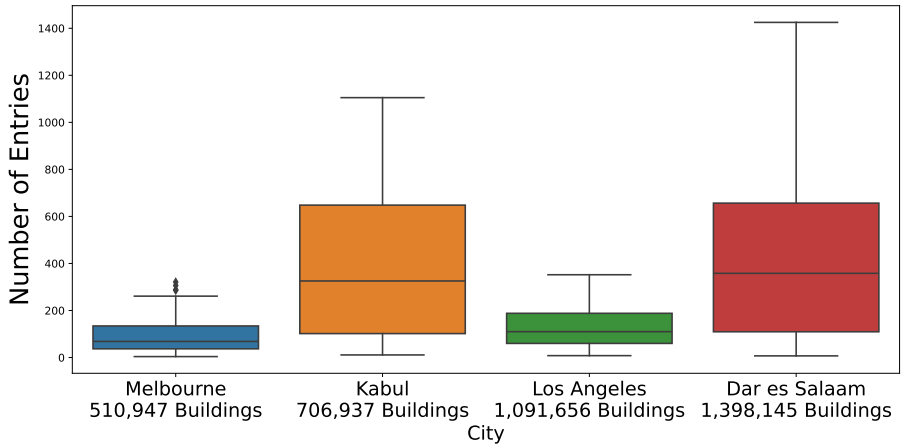


Figure 11 *The routing tables can be efficiently compressed, requiring each device to store a maximum of 1400 entries (11 KB in Dar es Salaam, a dense city with many small buildings, the worst-case for CityMesh).*

k **value.** In §4.2, we used the exponent k to control how paths are selected. An increasing value of k will select buildings that are closer together. To determine a proper value of k, we repeated the simulation over 15 cities with varying values of k. We observe that $k = 10$ provides the best deliverability, and outperforms $k = 1$ and the MST ($k \to \infty$). This is because $k = 1$ selects a path that encourages the packet to hop over distant buildings while the MST selects a path that is too long, accumulating the chance of a dropped packet over the long path. For all other results, we select $k = 10$.

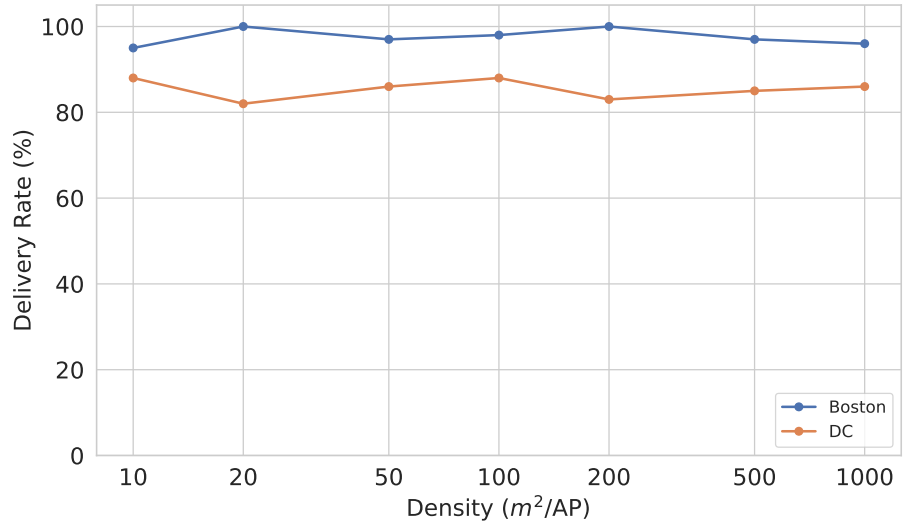


Figure 12 *Delivery performance given variation in the simulated density of CityMesh devices in two representative cities. This experiment varies the placement of devices in the city from a device being placed every 10 m^2 to a device every 1000 m^2. Small fluctuations in the results are attributable to the stochastic placement of devices within building footprints.*

Routing table size. Figure 11 plots the routing table size of 100 randomly selected buildings in four representative cities: Melbourne, Kabul, Los Angeles, and Dar es Salaam. Each entry in the table consumes up to 8 bytes. For Dar es Salaam, the mean number of entries is 456 (3648 bytes) in a

(a) *Hak5 W-Fi Pineapple Mk. VII*

(b) *GL-iNet Beryl AX (GL-MT3000)*

Figure 13 *The Hak5 Pineapple and GL-iNet devices used in the testbed, as deployed on campus.*

network of nearly 1.4 million buildings. The maximum size is 1425 entries (11.4 KB). For Melbourne, Los Angeles, and Kabul the mean number of entries are 92 (736 bytes), 127 (1016 bytes), and 383 (3064 bytes), respectively. These findings demonstrate CityMesh's compact routing tables across cities.

Device density. Figure 12 presents the impact of device density on routing performance, varying from 1 device per 10 square meters to 1 device per 1,000 square meters across two representative cities. Results indicate that CityMesh maintains consistent performance even at sparse deployments of 1 device per 1,000 square meters, demonstrating robustness to substantial reductions in infrastructure density.

6 Hardware Testbed

To evaluate CityMesh on real-world hardware, we developed a hardware testbed comprising 300 commodity access points distributed in 31 buildings around the M.I.T. campus. We wall-mounted the devices in hallways, offices, and classrooms at variable densities (Figure 13). We found that inter-device connectivity changes over the course of a day.

The testbed uses two types of devices, the Hak5 Wi-Fi Pineapple Mark VII [34], and the GL-iNet Beryl AX (GL-MT3000) [29]. We selected these devices based on their multiple radios and compatibility with OpenWRT [79], a popular open-source Linux-based access point platform. The Wi-Fi Pineapple is marketed as a penetration testing tool and is designed for active data collection using Wi-Fi testing tools. The Wi-Fi radios and MIPS processor are similar to what one would find in a low-end access point produced in the last few years. The ARM-based GL-iNet Beryl AX is marketed as a modern travel router, designed for portably connecting users' personal devices to the network.

We connected one radio on each device to the campus wireless network as a backhaul for remote access and configuration. Simultaneously, the devices formed a 802.11s mesh point network on the 2.4 GHz band with the default link learning and routing disabled in the kernel.

We built a stripped-down OpenWRT image for each device. This image used a VPN to establish a connection with a central control server that both issued software updates and received aggregated log messages from experiments running on the devices. All devices ran an agent daemon written

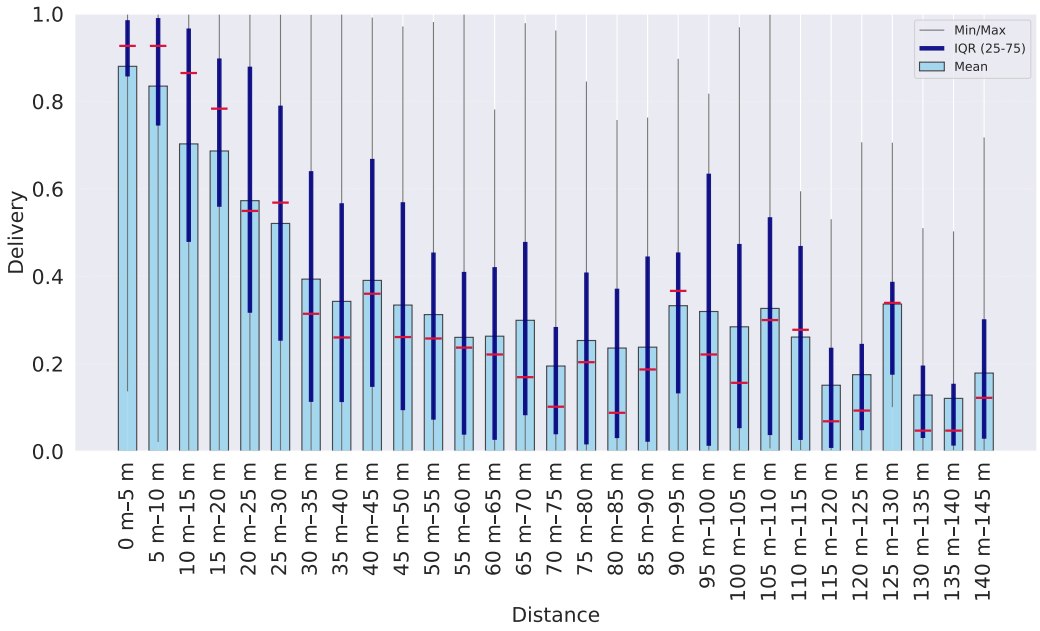


Figure 14 *Measured ping delivery as a function of distance.*

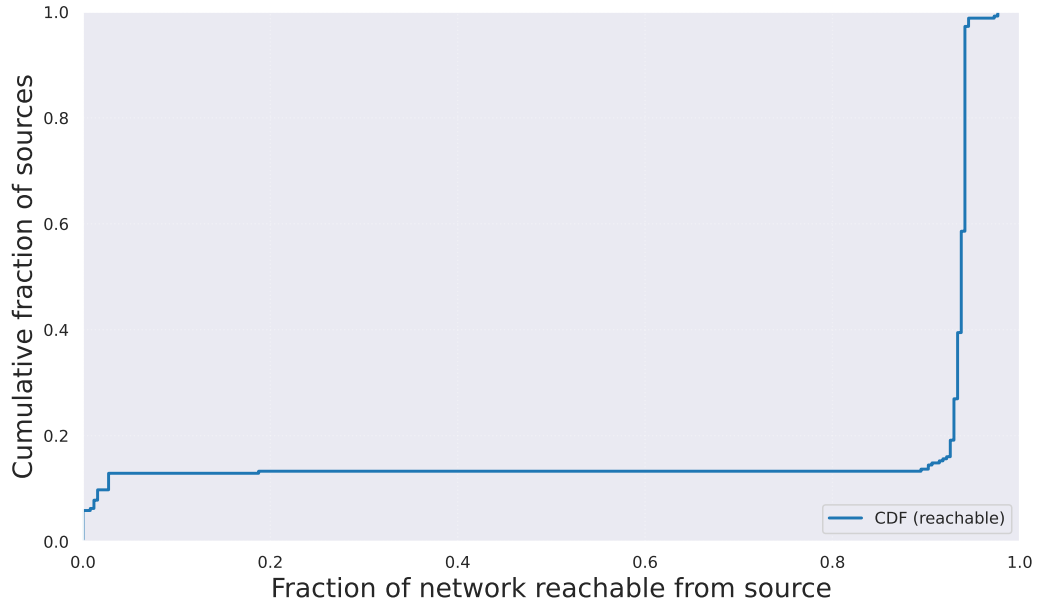


Figure 15 *Fraction of the testbed's devices reachable by source devices.*

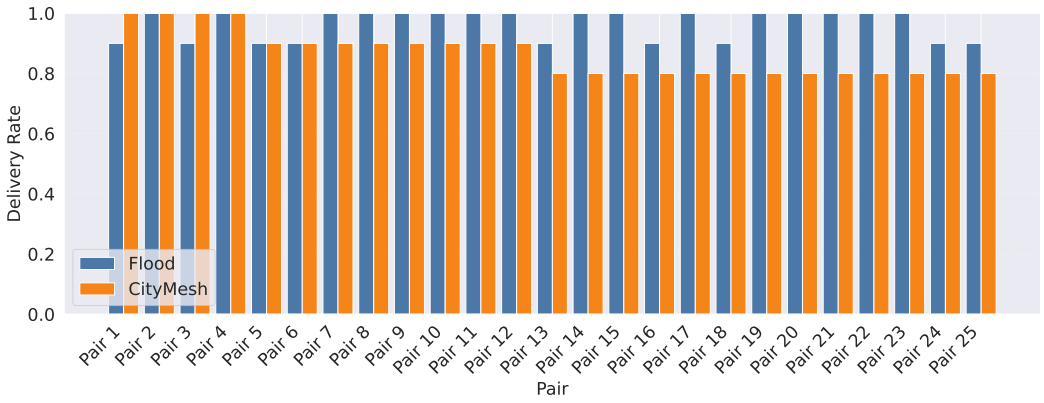


Figure 16 *CityMesh delivery rates are close to the optimal flood-based delivery rates in practice.*

in Go that performed a number of tasks. The agent verified each device's health, transmitted and received small, periodic UDP broadcasts over the 802.11s network ("pings"), and ran experiments on-demand when instructed to do so over a management gRPC socket, sending data packets between source and destination devices across the network. The agent aggregated local log messages emitted from experiments and sent them to the central log server for analysis.

To provide a realistic estimate for packet loss rate for our simulations, we aggregated ping data from the testbed network. The pings are emitted by every device once every five seconds. When received, a device simply logs that it had received the unique ping message. Transmit and receive events were aggregated to provide directional probabilities of packet delivery throughout the network. Figure 14 shows the aggregate performance of the links in the testbed binned by distance between the devices.

We evaluated the coverage of our mesh network by sending packets that are rebroadcast exactly once by all devices that hear the transmission. For each source, we are able to determine the fraction of the testbed that is reachable. Figure 15 shows the fraction of the testbed that is reachable across all sources. Most nodes are reachable from most source nodes.

We evaluated performance of CityMesh against floods, which use all devices in the network to achieve high delivery rates at the expense of significant transmission overhead. We sent packets from a given source to a selected destination ten times using CityMesh with a 50 m conduit. We selected two distant areas of the campus separated by 200 m to serve as the endpoints, and sampled devices within these buildings. Results are shown in Figure 16, arranged by descending delivery performance. These results show the CityMesh protocol performing well in practice.

7 Limitations

Building dependence. A key assumption of CityMesh is that users and devices are located in buildings, excluding devices in open areas. For instance, pedestrians near buildings, despite being capable relays, are excluded. One potential solution could be to augment the building graph with additional urban features such as road segments, walkways, and bridges. Another approach could be for devices to participate in the forwarding, acting almost as an intermediate, imaginary building.

Routing in 3D. CityMesh models cities as two-dimensional building footprints, treating each building as a single planar entity, which likely simplifies suppression decisions. In reality, buildings are three-dimensional: devices are distributed across multiple floors, basements, and rooftops, often with different levels of wireless reachability. CityMesh attempts to consider this by measuring the

connectivity to neighboring buildings (§4.6.2), and bootstrapping their score accordingly. However, it is possible in a large skyscraper, this could potentially lead to large amounts of flooding.

Clustered building failures. As noted in §4.7, while CityMesh handles random building failures, routing scalably around clustered failures is an open problem.

8 Conclusion

We presented CityMesh, a city-scale scalable wireless routing system suited for disaster recovery and emergencies. When wide-area connectivity is unavailable or significantly degraded, CityMesh enables static access points and mobile devices equipped with Wi-Fi in a city to route packets via each other for intra-city connectivity. The chief contribution of our work is a new map-based routing protocol that scales to 1.4 million buildings (and several devices in each building), a significant improvement over prior work on wireless mesh and mobile ad hoc networks. Simulation result shows sufficient packet delivery rates at modest packet overhead even with high link-layer packet loss rates with only a few hundred routing table entries for a typical city.

References

1 Abdeldime Mohamed Salih Abdelgader, Lenan Wu, and Mohammed Mohsen Mohammed Nasr. A Simplified Mobile Ad Hoc Network Structure for Helicopter Communication. *International Journal of Aerospace Engineering*, 2016(1), 2016.

2 Mikhail Afanasyev, Tsuwei Chen, Geoffrey M Voelker, and Alex C Snoeren. Analysis of a Mixed-Use Urban WiFi Network: When Metropolitan Becomes Neapolitan. In *ACM IMC*, 2008.

3 A. N. Al-Khwildi, S. Khan, K.K. Loo, and H. S. Al-Raweshidy. On-Demand Link Weight Routing Protocol with Cross-Layer Communication. In *18th Intl. Symp. on Personal, Indoor and Mobile Radio Comm.*, 2007.

4 Katy Allan. The global impact of security breaches and IT meltdowns. `https://cybermagazine.com/articles/the-global-impact-of-security-breaches-and-it-meltdowns`, October 2023.

5 Ameer Shakayb Arsalaan, Mah-Rukh Fida, and Hung X. Nguyen. UAVs Relay in Emergency Communications with Strict Requirements on Quality of Information. *IEEE Transactions on Vehicular Technology*, 74(3):4877–4892, 2025.

6 H. Badis, I. Gawedzki, and K. Al Agha. QoS routing in ad hoc networks using QOLSR with no need of explicit reservation. In *60th IEEE Vehicular Technology Conf.*, 2004.

7 Stefano Basagni, Imrich Chlamtac, Violet R Syrotiuk, and Barry A Woodward. A distance routing effect algorithm for mobility (DREAM). In *MobiCom*, 1998.

8 B.A.T.M.A.N. Advanced Development Team. B.A.T.M.A.N. Advanced (batman-adv). `https://www.open-mesh.org/projects/batman-adv`. Accessed: 2026-01-19.

9 John Bicket, Daniel Aguayo, Sanjit Biswas, and Robert Morris. Architecture and evaluation of an unplanned 802.11b mesh network. In *MobiCom*, 2005.

10 Zachary S Bischof, Kennedy Pitcher, Esteban Carisimo, Amanda Meng, Rafael Bezerra Nunes, Ramakrishna Padmanabhan, Margaret E Roberts, Alex C Snoeren, and Alberto Dainotti. Destination unreachable: Characterizing internet outages and shutdowns. In *SIGCOMM*, 2023.

11 Prosenjit Bose, Pat Morin, Ivan Stojmenović, and Jorge Urrutia. Routing with guaranteed delivery in ad hoc wireless networks. In *Discrete Algorithms and Methods for Mobile Computing and Communications (DIALM)*, 1999.

12 Vladimir Brik, Shravan Rayanchu, Sharad Saha, Sayandeep Sen, Vivek Shrivastava, and Suman Banerjee. A Measurement Study of a Commercial-grade Urban WiFi Mesh. In *IMC*, 2008.

13 Justine Calma. Brooklyn built a disaster-proof mesh Wi-Fi network just in time for Hurricane Sandy. *The Verge*, May 2023. URL: `https://www.theverge.com/c/features/23700677/wifi-mesh-network-disaster-hurricane-sandy-brooklyn`.

14 Fox Carolina. Spectrum outage: Company releases timeline for restoration. *Fox Carolina*, October 2024. URL: `https://www.foxcarolina.com/2024/10/11/spectrum-outage-company-releases-timeline-restoration/`.

15 Donghui Chen, Ding-Zhu Du, Xiao-Dong Hu, Guo-Hui Lin, Lusheng Wang, and Guoliang Xue. Approximations for steiner trees with minimum number of steiner points. *J. of Global Optimization*, 18(1):17–33, 2000.

16 Xiuzhen Cheng, Ding-Zhu Du, Lusheng Wang, and Baogang Xu. Relay sensor placement in wireless sensor networks. *Wireless Networks*, 14(3):347–355, 2008.

17 Cloudflare. Internet traffic disruption caused by the christmas day bombing in nashville. Blog post, February 2021. URL: `https://blog.cloudflare.com/internet-traffic-disruption-caused-by-the-christmas-day-bombing-in-nashville/`.

18 M. Scott Corson and Anthony Ephremides. A distributed routing algorithm for mobile wireless networks. *Wireless Network*, 1:61–81, 1995.

19 Wanja de Sombre, Arash Asadi, Debopam Bhattacherjee, Deepak Vasisht, and Andrea Ortiz. Skylink: Scalable and resilient link management in leo satellite network, 2025. URL: `https://arxiv.org/abs/2509.08455`, `arXiv:2509.08455`.

20 Richard Draves, Jitendra Padhye, and Brian Zill. Comparison of routing metrics for static multi-hop wireless networks. In *SIGCOMM*, 2004.

21 R.P. Draves, C. King, S. Venkatachary, and B.D. Zill. Constructing optimal IP routing tables. In *INFOCOM*, 1999.

22 Karim El Defrawy and Gene Tsudik. ALARM: Anonymous Location-Aided Routing in Suspicious MANETs. *IEEE Transactions on Mobile Computing*, 10(9):1345–1358, 2011.

23 Michael B. Farrell. Cellphone networks overwhelmed in blast aftermath. *Boston.com*, April 2013. URL: https://www.boston.com/news/mit/2013/04/16/cellphone-networks-overwhelmed-in-blast-aftermath/.

24 Freifunk Community. Freifunk Mesh Networking Project, 2025. Accessed: 2025-01-30. URL: https://api-viewer.freifunk.net/.

25 Holger Füßler, Hannes Hartenstein, Martin Mauve, Wolfgang Effelsberg, and Joerg Widmer. Contention-based forwarding for street scenarios. *International Workshop in Intelligent Transportation (WIT)*, 2004.

26 Amalinda Gamage, Jansen Liando, Chaojie Gu, Rui Tan, Mo Li, and Olivier Seller. Lmac: Efficient carrier-sense multiple access for lora. *ACM Transactions on Sensor Networks*, 2023.

27 J. J. Garcia-Luna-Aceves and Marcelo Spohn. Source-tree routing in wireless networks. In *ICNP*, 1999.

28 Samantha Gholar. Internet still down after Hurricane Milton? Here's how to track when WiFi might return. https://www.heraldtribune.com/story/news/local/2024/10/15/floridians-without-connectivity-await-internet-power-restoration/75673551007/, Retrieved by Sept 29th 2025.

29 GL-iNet. Beryl ax (gl-mt3000). https://www.gl-inet.com/products/gl-mt3000. Accessed: 2025-10-01.

30 Ronald L. Graham. An efficient algorithm for determining the convex hull of a finite planar set. *Inf. Process. Lett.*, 1(4):132–133, 1972.

31 P. Guangyu, M. Geria, and Xiaoyan Hong. LANMAR: Landmark routing for large scale wireless ad hoc networks with group mobility. In *MobiHoc*, 2000.

32 M. Gunes, U. Sorges, and I. Bouazizi. ARA–the ant-colony based routing algorithm for MANETs. In *Proceedings. International Conference on Parallel Processing Workshop*, 2002.

33 Zygmunt Haas. The zone routing protocol (ZRP) for ad hoc networks, 1998.

34 Hak5. WiFi Pineapple Mark VII. https://shop.hak5.org/products/wifi-pineapple. Accessed: 2026-01-19.

35 Mordechai Haklay and Patrick Weber. OpenStreetMap: User-Generated Street Maps. *IEEE Pervasive Computing*, 7(4):12–18, 2008.

36 C. Hedrick. Routing Information Protocol. RFC 1058, Internet Engineering Task Force, June 1988. URL: https://datatracker.ietf.org/doc/html/rfc1058.

37 Amir Husain and Ada Robertson. Millions of phones went offline after Ericsson's certificate expired. *The Verge*, December 2018. URL: https://www.theverge.com/2018/12/7/18130323/ericsson-software-certificate-o2-softbank-uk-japan-smartphone-4g-network-outage.

38 Apple Inc. Emergency SOS via satellite on iPhone 14 and iPhone 14 Pro lineups made possible by $450 million Apple investment in US infrastructure. https://www.apple.com/newsroom/2022/11/emergency-sos-via-satellite-made-possible-by-450m-apple-investment/, 2022.

39 Xona Partners Inc. Assessment of Rogers Networks for Resiliency and Reliability Following the 8 July 2022 Outage – Executive Summary. https://crtc.gc.ca/eng/publications/reports/xona2024.htm, July 2024.

40 A. Iwata, Ching-Chuan Chiang, Guangyu Pei, M. Gerla, and Tsu-Wei Chen. Scalable routing strategies for ad hoc wireless networks. *IEEE Journal on Selected Areas in Communications*, 17(8):1369–1379, 1999.

41 P. Jacquet, P. Muhlethaler, T. Clausen, A. Laouiti, A. Qayyum, and L. Viennot. Optimized link state routing protocol for ad hoc networks. In *IEEE International Multi Topic Conference (INMIC). Technology for the 21st Century*, 2001.

42 R. Jain, A. Puri, and R. Sengupta. Geographical routing using partial information for wireless ad hoc networks. *IEEE Personal Communications*, 8(1):48–57, 2001.

43 Vasu Jakkal. How nation-state attackers like NOBELIUM are changing cybersecurity. https://www.microsoft.com/en-us/security/blog/2021/09/28/how-nation-state-attackers-like-nobelium-are-changing-cybersecurity/#:~:text=Nation%2Dstate%20attacks%20are%20malicious,impacted%20by%20the%20NOBELIUM%20attacks., 2021.

44 M. Jerbi, S.-M. Senouci, R. Meraihi, and Y. Ghamri-Doudane. An Improved Vehicular Ad Hoc Routing Protocol for City Environments. In *IEEE ICC*, 2007.

45 Moez Jerbi, Sidi-Mohammed Senouci, Tinku Rasheed, and Yacine Ghamri-Doudane. Towards efficient geographic routing in urban vehicular networks. *IEEE Transactions on Vehicular Technology*, 58(9):5048–5059, 2009.

46 M. Joa-Ng and I-Tai Lu. A peer-to-peer zone-based two-level link state routing for mobile ad hoc networks. *IEEE Journal on Selected Areas in Communications*, 1999.

47 David B Johnson, David A Maltz, Josh Broch, et al. Dsr: The dynamic source routing protocol for multi-hop wireless ad hoc networks. *Ad Hoc Networks*, 5(1):139–172, 2001.

48 Brad Karp and H.T. Kung. GPSR: Greedy perimeter stateless routing for wireless networks. In *MobiCom*, 2000.

49 Elliott Karpilovsky, Matthew Caesar, Jennifer Rexford, Aman Shaikh, and Jacobus van der Merwe. Practical Network-Wide Compression of IP Routing Tables. *IEEE Transactions on Network and Service Management*, 9(4):446–458, 2012.

50 Young-Jin Kim, Ramesh Govindan, Brad Karp, and Scott Shenker. Geographic routing made practical. In *USENIX NSDI*, 2005.

51 K Kiran, N P Kaushik, S Sharath, P Deepa Shenoy, K R Venugopal, and Vignesh T Prabhu. Experimental Evaluation of BATMAN and BATMAN-Adv Routing Protocols in a Mobile Testbed. In *TENCON IEEE Region 10 Conference*, 2018.

52 Young-Bae Ko and Nitin H. Vaidya. Location-Aided Routing (LAR) in mobile ad hoc networks. *Wireless Network*, 6(4):307–321, 2000.

53 Michal Król, Eryk Schiller, Franck Rousseau, and Andrzej Duda. WEAVE: Efficient Geographical Routing in Large-Scale Networks. In *Proceedings of the International Conference on Embedded Wireless Systems and Networks (EWSN)*, 2016.

54 Fabian Kuhn, Rogert Wattenhofer, Yan Zhang, and Aaron Zollinger. Geometric ad-hoc routing: Of theory and practice. In *PODC*, 2003.

55 Fabian Kuhn, Rogert Wattenhofer, and Aaron Zollinger. Worst-case optimal and average-case efficient geometric ad-hoc routing. In *MobiHoc*, 2003.

56 Liangli Lai, Qianping Wang, and Qun Wang. Research on one kind of improved gpsr algorithm. In *2012 International Conference on Computer Science and Electronics Engineering*, 2012.

57 Kevin C. Lee, Jerome Haerri, Uichin Lee, and Mario Gerla. Enhanced Perimeter Routing for Geographic Forwarding Protocols in Urban Vehicular Scenarios. In *IEEE Globecom Workshops*, 2007.

58 Ben Leong, Barbara Liskov, and Robert Tappan Morris. Geographic Routing Without Planarization. In *NSDI*, 2006.

59 Ben Leong, Sayan Mitra, and Barbara Liskov. Path vector face routing: Geographic routing with local face information. In *ICNP*, 2005.

60 Chenning Li, Xiuzhen Guo, Longfei Shangguan, Zhichao Cao, and Kyle Jamieson. CurvingLoRa to Boost LoRa Network Throughput via Concurrent Transmission. In *NSDI*, 2022.

61 Jingya Li, Xingqin Lin, Keerthi Kumar Nagalapur, Zhiqiang Qi, Adrián Lahuerta-Lavieja, Thomas Chapman, Sam Agneessens, Henrik Sahlin, Daniel Guldbrand, and Joakim Åkesson. Toward Providing Connectivity When and Where It Counts: An Overview of Deployable 5G Networks. *IEEE Communications Standards Magazine*, 6(4):56–64, 2022.

62 Jinyang Li, John Jannotti, Douglas S. J. De Couto, David R. Karger, and Robert Morris. A Scalable Location Service for Geographic Ad Hoc Routing. In *MobiCom*, 2000.

63 Guo-Hui Lin and Guoliang Xue. Steiner tree problem with minimum number of Steiner points and bounded edge-length. *Information Processing Letters*, 69(2):53–57, 1999.

64 Errol L. Lloyd and Guoliang Xue. Relay node placement in wireless sensor networks. *IEEE Transactions on Computers*, 56(1):134–138, 2007.

65 Christian Lochert, Hannes Hartenstein, Jing Tian, Holger Füßler, Dagmar Hermann, and Martin Mauve. A routing strategy for vehicular ad hoc networks in city environments. In *IEEE Intelligent Vehicles Symposium*, 2003.

66 Christian Lochert, Martin Mauve, Holger Füßler, and Hannes Hartenstein. Geographic routing in city scenarios. *ACM SIGMOBILE Mobile Computing and Communications Review*, 9(1):69–72, 2005.

67 Yang Lu, Guanming Zhang, and Donglei Wang. Assessing community-level flood resilience: Analyzing functional interdependencies among building sectors. *Applied Sciences*, 15(6):3161, 2025.

68 James Lynch, Ziqian Liu, Chenning Li, Manya Ghobadi, and Hari Balakrishnan. The Case for Decentralized Fallback Networks. In *HotNets*, 2024.

69 José Alex Pontes Martins, Sergio Luis O. B. Correia, and Joaquim Celestino. Ant-DYMO: A bio-inspired algorithm for MANETS. In *17th International Conference on Telecommunications*, 2010.

70 S. Marwaha, Chen Khong Tham, and D. Srinivasan. Mobile agents based routing protocol for mobile ad hoc networks. In *GLOBECOM*, 2002.

71 Vahid Mohsenian, Nima Gharaei-Moghaddam, Stefano Mariani, and Iman Hajirasouliha. Assessment of the effects of non-structural components on the seismic reliability of structures via a block diagram method. *Structures*, 47:2050–2065, 2023.

72 Denisse Moreno. Project Loon: Stratospheric Balloons Are Connecting Hurricane Maria Victims in Puerto Rico, 2017. URL: `https://www.ibtimes.com/project-loon-stratospheric-balloons-are-connecting-hurricane-maria-victims-puerto-2604513`.

73 Shree Murthy and J. J. Garcia-Luna-Aceves. An efficient routing protocol for wireless networks. *Mobile Networks and Applications*, 1(2):183–197, 1996.

74 Valery Naumov, Rainer Baumann, and Thomas Gross. An evaluation of inter-vehicle ad hoc networks based on realistic vehicular traces. In *MobiHoc*, 2006.

75 Shaun Nichols. Why millions of Brits' mobile phones were knackered: An expired Ericsson software certificate. *The Register*, December 2018. URL: `https://www.theregister.com/2018/12/06/ericsson_o2_telefonica_uk_outage/`.

76 R.G. Ogier. Efficient routing protocols for packet-radio networks based on tree sharing. In *IEEE Intl. Workshop on Mobile Multimedia Communications (MoMuC))*, 1999.

77 Vitali Ogorodnikov. New York YIMBY's Mid-Year 2025 Report Counts 23,446 New Residential and Hotel Unit Filings from January Through June. *New York YIMBY*, August 2025.

78 Seoyul Oh and Deepak Vasisht. A Call for Decentralized Satellite Networks. In *HotNets*, pages 25–33, 2024.

79 Openwrt. OpenWRT. `https://openwrt.org/`, 2026.

80 V.D. Park and M.S. Corson. A highly adaptive distributed routing algorithm for mobile wireless networks. In *Proceedings of INFOCOM '97*, 1997.

81 Vincent Douglas Park and M Scott Corson. A highly adaptive distributed routing algorithm for mobile wireless networks. In *IEEE INFOCOM*, volume 3, pages 1405–1413. IEEE, 1997.

82 C.E. Perkins and E.M. Royer. Ad-hoc on-demand distance vector routing. In *IEEE Workshop on Mobile Computing Systems and Applications (WMCSA)*, 1999.

83 Charles E Perkins and Pravin Bhagwat. Highly dynamic destination-sequenced distance-vector routing (DSDV) for mobile computers. In *ACM SIGCOMM*, 1994.

84 Charles E. Perkins, John Dowdell, Lotte Steenbrink, and Victoria Pritchard. Ad hoc on-demand distance vector version 2 (aodvv2) routing. Internet-Draft draft-perkins-manet-aodvv2-05, Internet Engineering Task Force, November 2024. Work in Progress. URL: `https://datatracker.ietf.org/doc/html/draft-perkins-manet-aodvv2-05`.

85 Ananth Rao, Sylvia Ratnasamy, Christos Papadimitriou, Scott Shenker, and Ion Stoica. Geographic routing without location information. In *MobiCom*, 2003.

86 Sunder Aditya Rao, Manohara Pai, Mounir Boussedjra, and Joseph Mouzna. GPSR-L: Greedy perimeter stateless routing with lifetime for VANETS. In *IEEE International Conference on ITS Telecommunications*, 2008.

87 Gábor Rétvári, János Tapolcai, Attila Kőrösi, András Majdán, and Zalán Heszberger. Compressing IP Forwarding Tables: Towards Entropy Bounds and Beyond. In *SIGCOMM*, 2013.

88 Adi Robertson. AT&T recovers from multi-state outage after Nashville bombing. *The Verge*, December 2020. URL: https://www.theverge.com/2020/12/28/22202822/att-outage-nashville-christmas-bombing.

89 Elizabeth M Royer and Charles E Perkins. An implementation study of the AODV routing protocol. In *IEEE Wireless Communications and Networking Conference*, 2000.

90 Nagham H. Saeed, Maysam F. Abbod, and Hamed S. Al-Raweshidy. MANET Routing Protocols Taxonomy. In *International Conference on Future Communication Networks*, 2012.

91 Karim Seada, Ahmed Helmy, and Ramesh Govindan. On the effect of localization errors on geographic face routing in sensor networks. In *IPSN*, 2004.

92 Boon-Chong Seet, Genping Liu, Bu Lee, Chuan Foh, Kai Wong, and Keok-Kee Lee. A-STAR: A Mobile Ad Hoc Routing Strategy for Metropolis Vehicular Communications. In *International conference on research in networking*, 2004.

93 B. Shanmuga Raja, N. Prabakaran, and V. R. Sarma Dhulipala. Modified GPSR Based Optimal Routing Algorithm for Reliable Communication in WSNs. In *International Conference on Devices and Communications (ICDeCom)*, 2011.

94 Jayanth Shenoy, Om Chabra, Tusher Chakraborty, Suraj Jog, Deepak Vasisht, and Ranveer Chandra. CosMAC: Constellation-Aware Medium Access and Scheduling for IoT Satellites. In *MobiCom*, 2024.

95 Andrey Silva, KM Niaz Reza, and Aurenice Oliveira. An adaptive GPSR routing protocol for VANETs. In *IEEE International Symposium on Wireless Communication Systems (ISWCS)*, 2018.

96 Harvinder Singh. *Compass routing on geometric graphs.* University of Ottawa, 1999.

97 Vaibhav Singh, Tusher Chakraborty, Suraj Jog, Om Chabra, Deepak Vasisht, and Ranveer Chandra. Exploiting Satellite Doppler for Reliable and Faster Data Download in IoT Satellite Networks. *GetMobile: Mobile Computing and Communications*, 27(4):11–14, 2024.

98 Vaibhav Singh, Tusher Chakraborty, Suraj Jog, Om Chabra, Deepak Vasisht, and Ranveer Chandra. Spectrumize: Spectrum-efficient Satellite Networks for the Internet of Things. In *NSDI*, 2024.

99 P. Sinha, R. Sivakumar, and V. Bharghavan. CEDAR: A Core-Extraction Distributed Ad Hoc Routing Algorithm. In *INFOCOM*, 1999.

100 Keith Sklower. A Tree-based Packet Routing Table for Berkeley Unix. In *USENIX Winter Conf.*, 1991.

101 SpaceX. Starlink. https://www.starlink.com/.

102 Venkatachary Srinivasan and George Varghese. Fast address lookups using controlled prefix expansion. *ACM Transactions on Computer Systems*, 17(1):1–40, 1999.

103 Citizen Times Staff. Spectrum After Helene: Why Is Internet Still Out with Power Restored? *Citizen Times*, October 2024. URL: https://www.citizen-times.com/story/news/local/2024/10/10/spectrum-after-helene-why-is-internet-still-out-with-power-restored/75603128007/.

104 Starlink. Starlink Fair Use Policy. https://www.starlink.com/legal/documents/DOC-1469-65206-75, 2025.

105 Bruce Sterling. Project Loon Actually Working for Actual People, 2017. URL: https://www.wired.com/beyond-the-beyond/2017/11/project-loon-actually-working-actual-people/.

106 B.A.T.M.A.N. Development Team. B.a.t.m.a.n. (better approach to mobile ad-hoc networking). https://www.open-mesh.org/projects/open-mesh/wiki. Accessed: 2026-01-19.

107 Kevin Townsend. Russian Telco Hijacked Internet Traffic of Major Networks – Accident or Malicious Action?, 2020. URL: https://www.securityweek.com/russian-telco-hijacked-internet-traffic-major-networks-accident-or-malicious-action/.

108 A. Valera, W.K.G. Seah, and S.V. Rao. Cooperative packet caching and shortest multipath routing in mobile ad hoc networks. In *INFOCOM*, 2003.

109 VETRO. When the Cable Gets Cut: How a Fibre Management System of Record Saves the Day for ISPs, 2024. URL: https://vetrofibermap.com/when-the-cable-gets-cut-how-a-fibre-management-system-of-record-saves-the-day-for-isps/.

110 Guoqiang Wang, Yongchang Ji, Dan C. Marinescu, and Damla Turgut. A routing protocol for power constrained networks with asymmetric links. In *Proceedings of the 1st ACM International Workshop on Performance Evaluation of Wireless Ad Hoc, Sensor, and Ubiquitous Networks*, 2004.

111 Jianping Wang, Eseosa Osagie, Parimala Thulasiraman, and Ruppa K. Thulasiram. HOPNET: A hybrid ant colony optimization routing algorithm for mobile ad hoc network. *Ad Hoc Networks*, 7(4):690–705, 2009.

112 Nen-Chung Wang and Si-Ming Wang. An efficient location-aided routing protocol for mobile ad hoc networks. In *11th International Conference on Parallel and Distributed Systems (ICPADS)*, 2005.

113 Wentao Wang, Xianglin Wei, Yingchao Jia, and Ming Chen. UAV relay network deployment through the area with barriers. *Ad Hoc Networks*, 149, 2023.

114 Wikipedia. 2020 Nashville bombing, 2020. URL: `https://en.wikipedia.org/wiki/2020_Nashville_bombing`.

115 Qi Xue and Aura Ganz. Ad hoc QoS on-demand routing (AQOR) in mobile ad hoc networks. *J. Parallel Distrib. Comput.*, 63(2):154–165, 2003.

116 Tong Yang, Bo Yuan, Shenjiang Zhang, Ting Zhang, Ruian Duan, Yi Wang, and Bin Liu. Approaching optimal compression with fast update for large scale routing tables. In *2012 IEEE 20th International Workshop on Quality of Service*, 2012.

117 Xiaoping Yang, Mengjie Li, Zhihong Qian, and Te Di. Improvement of GPSR protocol in vehicular ad hoc network. *IEEE Access*, 6:39515–39524, 2018.

118 Xiaofeng Zhang and L. Jacob. Multicast zone routing protocol in mobile ad hoc wireless networks. In *28th Annual IEEE International Conference on Local Computer Networks (LCN)*, 2003.

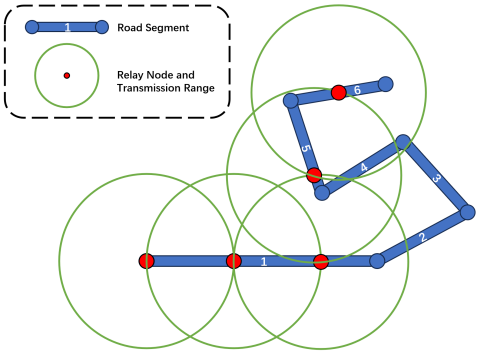


Figure 17 *An example of our method placing relay nodes on the road route. We intersect the circle with radius roughly equal to nominal transmission range, and place the relay node on the last road segment intersected. Here we are able to avoid placing relay nodes on road segment 2-4, and connect road segment 1 and 5 directly.*

A Connecting Islands with Steiner Point Insertion Algorithm

A.1 Problem Formulation

The problem of wireless relay placement has been studied before as a geometric optimization task, often modeled as a variant of the Steiner point insertion problem. In this setting, the aim is to introduce additional points, beyond the original inputs, in order to guarantee connectivity while satisfying two requirements:

1. Any two added relays must be separated by a distance less than R, the radio transmission range, so they can communicate with each other and with existing wireless devices.
2. The number of added relays should be as small as possible.

This variant is referred to as the *Steiner Minimum Tree with Minimum Steiner Points and Bounded Edge Length* (SMT-MSPBEL) problem. The problem is NP-hard [63], but a variety of approximation strategies have been proposed [16, 64, 63, 15]. Among these, Cheng et al. [16] present a 3-approximation algorithm with $O(n^3)$ complexity, where n represents the number of input nodes.

Beyond the standard SMT-MSPBEL formulation, we impose additional constraints to ensure that Steiner points are placed in practical, deployable locations. For instance, when connecting areas separated by a river, placing a relay directly on the water is neither feasible nor cost-effective. Instead, candidate points must be restricted to viable infrastructure, such as bridges or other accessible structures.

At the same time, we relax the strict objective of minimizing the total number of inserted Steiner points, given the high computational complexity of the problem. Instead, our approach focuses on keeping the number of relays reasonably small and practically justifiable.

A.2 Convex Hull Optimization

We build on the 3-approximation algorithm for the SMT-MSPBEL problem proposed in [16], which runs in $O(n^3)$ time. The algorithm proceeds in three stages. First, it sorts all pairwise edges by length and connects nodes whose edges lie within the transmission range. Next, it attempts to connect triplets of nodes by constructing 3-stars. Finally, it inserts points on the remaining edges until the graph is fully connected. The cubic complexity arises from the worst-case need to evaluate all possible node triplets when building 3-stars.

Applying the $O(n^3)$ runtime of the 3-approximation algorithm [16] directly to city-scale networks with millions of device, quickly becomes infeasible. Both storing all sorted edges and computing Steiner points are prohibitively resource-intensive at this scale. To address this issue, we introduce a convex-hull–based optimization that significantly reduces the effective input size n for the algorithm.

Our method first identifies disconnected components (or islands) using breadth-first search (BFS). For each island, we then compute its convex hull in $O(n \log n)$ time [30]. The resulting hull vertices serve as the reduced input to the original 3-approximation Steiner insertion algorithm. As shown in Table 1, in dense urban settings, this approach yields a substantial reduction in input size n, enabling more efficient computation.

City	Original Input	After Prune
Ho Chi Minh	161,207	14,399
Seoul	308,929	20,557
Busan	111,974	14,095
Karachi	269,487	14,029
Bangkok	321,741	20,927
Monaco	3,872	65
Manila	145,902	432
Washington, D.C.	32,213	335
Toronto	215,877	2,198
New York	1,581,394	11,725

Table 1 *Effect of convex hull optimization for relay generation across 10 different cities.*

We recognize that relying on convex hulls sacrifices the strict 3-approximation guarantee. In some cases, the optimal placement of Steiner points may lie inside a convex region, especially when device distributions are highly non-convex. However, since most city layouts and building clusters are approximately convex, the additional relays required are typically negligible. Given our focus on scalability while keeping relay counts modest, we accept this trade-off: the substantial performance gains outweigh the minor loss in approximation accuracy.

A.3 Map-based Relay Placement in Feasible Locations

A remaining challenge is ensuring that Steiner points are placed in practical, deployable locations. In its raw form, the approximation algorithm may insert relays in unrealistic areas—such as rivers or other inaccessible terrain, particularly when Steinerizing edges that span physical barriers. In practice, relays should be placed on buildings, roads, or bridges, where deployment is feasible and cost-effective.

To handle these real-world constraints, we again use map data for relay placement. When two nodes cannot connect directly due to an obstacle (e.g., a river), we compute a *driving route* between them using road network data. This approach ensures that relay positions follow existing infrastructure such as roads and bridges, aligning with our deployment constraints.

Since road routes often include frequent turns and detours, naively placing relays along every segment would be inefficient. To avoid over-placement, we intersect the road path with a circle representing the nominal transmission range and select the intersection point on the last road segment within the circle as the relay location. This refinement maintains reliable connectivity while minimizing the

number of additional relays. Figure 17 illustrates this method.

We acknowledge that routing relays along road networks may not always yield the theoretical minimum number of nodes. Nevertheless, by combining map data to compute the shortest driving path with our circle-intersection method for relay placement, we obtain a solution that is both practical and effective. This approach trades strict optimality for feasibility, providing a well-grounded approximation that ensures relays are placed in realistic locations while keeping their overall number reasonably low.

B Convergence of Edge-Weighted Scheme

In (§4.2), we presented a scheme which augments the edge-weighted metric with a d^k penalty. Here we show that as $k \to \infty$, optimal paths concentrate on the Minimum Spanning Tree (MST). The intuition is that $\sum_{e \in P} d(e)^k$ increasingly behaves like a "minimize the largest edge" objective: a single edge that is even slightly longer than the alternatives becomes exponentially more expensive than any bounded number of slightly shorter edges.

Formally, let T be the MST (assume distinct edge weights for simplicity) and suppose, for contradiction, that an optimal $s \to t$ path P for arbitrarily large k uses some edge $e = (u, v) \notin T$. Adding e to T creates a unique cycle C. By the MST cycle property, e is the heaviest edge on C. Let $d_{\max}$ denote the largest weight among the other edges on C, so $d(e) > d_{\max}$. The cycle consists of e plus the unique $u \to v$ path in T, call it P_{MST}, whose edges all have length at most $d_{\max}$. Replacing e in P with P_{MST} yields an alternative path P'. Since P_{MST} contains at most N edges for some finite graph-dependent bound N, its cost is at most $N \cdot d_{\max}^k$, whereas e alone costs $d(e)^k$. The ratio $\frac{d(e)^k}{N d_{\max}^k} = \frac{1}{N}\left(\frac{d(e)}{d_{\max}}\right)^k$ diverges as $k \to \infty$ because $d(e)/d_{\max} > 1$, so for sufficiently large k, $d(e)^k > \sum_{e' \in P_{\text{MST}}} d(e')^k$. Thus $\text{Cost}(P') < \text{Cost}(P)$, contradicting the optimality of P. Therefore, for large k, any minimum-$\sum d^k$ path cannot include edges outside the MST, and in the limit $k \to \infty$ the set of optimal paths is contained in T.

C Arithmetic Operations to Check a Point in a Conduit

We check whether a building lies within a conduit by computing the perpendicular distance from the building's centroid $P = (x_b, y_b)$ to the conduit centerline defined by the segment between the previous waypoint $WP_{\text{prev}} = (x_{\text{prev}}, y_{\text{prev}})$ and the next waypoint $WP_{\text{next}} = (x_{\text{next}}, y_{\text{next}})$.

This distance is derived from the magnitude of the cross product between the vectors from $WP_{\text{prev}} \to WP_{\text{next}}$ and $WP_{\text{prev}} \to P$, normalized by the length of the conduit segment.

$$d = \frac{\left|(x_{\text{next}} - x_{\text{prev}})(y_{\text{prev}} - y_b) - (y_{\text{next}} - y_{\text{prev}})(x_{\text{prev}} - x_b)\right|}{\sqrt{(x_{\text{next}} - x_{\text{prev}})^2 + (y_{\text{next}} - y_{\text{prev}})^2}}$$

For efficiency, we avoid the square root by comparing squared distances. A building centroid $P = (x_b, y_b)$ is considered to lie within the conduit if

$$\frac{\big((x_{\text{next}} - x_{\text{prev}})(y_{\text{prev}} - y_b) - (y_{\text{next}} - y_{\text{prev}})(x_{\text{prev}} - x_b)\big)^2}{(x_{\text{next}} - x_{\text{prev}})^2 + (y_{\text{next}} - y_{\text{prev}})^2} \leq \left(\frac{CW}{2}\right)^2$$

where CW denotes the total conduit width, and $\frac{CW}{2}$ is the one-sided conduit radius.