Cadence: A Simulator for Human Movement-based Communication Protocols

Harel Berger
Georgetown University
USA

Micah Sherr
Georgetown University
USA

Adam J. Aviv
The George Washington University
USA

ABSTRACT

Unfettered access to the Internet is unfortunately not universal —- studies show that more than half of the world's population is subject to at least some censorship. Even in regions without censorship, Internet outages (e.g., during natural disasters) hinder the ability to communicate online. Avoiding censorship and communicating during Internet outages have inspired a number of proposals for communicating via a class of decentralized routing protocols based on *sneakernets*. In a sneakernet, messages are passed between human-carried devices (usually smartphones), completely avoiding the Internet. Importantly, the movement of messages in a sneakernet is dictated by the movements of the (human) device owners; these networks tend to be opportunistic in the sense that messages are exchanged between parties only when those parties encounter one another through happenstance.

Understanding the security, performance, and privacy properties of various sneakernet protocols remains an open problem, with individual proposals inventing their own metrics and evaluation methodology, and is further challenged by a lack of unified evaluation and simulation frameworks. This paper presents *Cadence*, a simulator for decentralized human movement-based communication protocols that provides a unifying environment for evaluating sneakernet protocols under a variety of conditions. We describe the architecture of Cadence and argue its benefits to network and security researchers. Cadence is free open-source software.

ACM Reference Format:
Harel Berger, Micah Sherr, and Adam J. Aviv. 2023. Cadence: A Simulator for Human Movement-based Communication Protocols. In *2023 Cyber Security Experimentation and Test Workshop (CSET 2023), August 07–08, 2023, Marina del Rey, CA, USA.* ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/3607505.3607507

1 INTRODUCTION

The Internet is a global network of computers and other electronic devices interconnected through a system of routers, switches, and other equipment. While in much of the world, the Internet is freely accessible and unrestricted, this is not uniformly the case. The Internet has also become a target for censorship by both totalitarian and democratic nation-states [19, 27]. Notably, more than half of the world's population lives in countries that censor parts of the Internet [14, 19]. Additionally, the ability to communicate online can be disrupted in times of crisis, either due to natural (e.g., natural disasters) or human causes (e.g., war).

This has motivated a number of proposals for decentralized, delay-tolerant communication that avoids the Internet entirely [1, 2, 5, 7]. In this paper, we call such schemes *sneakernets*[1]. There are two defining properties of a sneakernet: First, communication is relayed through a series of device-to-device links. Examples of such links include Bluetooth Low Energy (BLE), near-field communication (NFC), and mesh-based Wifi. Second, messages are exchanged during *encounters* between the physical carriers of the devices —-i.e., the device's owners—when the devices are in close physical proximity. For example, two devices may relay messages while their owners visit the same coffee shop. In general, the challenge of sneakernets is designing routing protocols that maximize message delivery rates while minimizing latency and overall network load.

The obvious tradeoff of sneakernets is that the benefit of avoiding centralization (including the use of the Internet) comes at an enormous latency cost. Messages no longer move at the speed of electrons, but are rather dictated by human movements and happenstance encounters. Although they are ill-suited for many forms of communication, sneakernets present an important tool for communicating in totalitarian regimes and/or when natural or human-made disasters make access to the Internet untenable.

To date, sneakernets have been deployed only in a handful of instances [4, 5]. Perhaps the best-known example was during the Hong Kong protests in 2014 with the use of Firechat [4], showing their power in the right circumstances. We posit that a necessary *(and up to now, missing)* step towards wider adoption is a mechanism for consistently evaluating and testing the performance of sneakernets under varied adverse scenarios. In this paper, we present *Cadence*, a simulator that attempts to fill this gap by providing a common framework for evaluating and comparing different sneakernet protocols under various adverse scenarios. Cadence has the following high-level goals:

Protocol expressiveness. Cadence should enable researchers to evaluate customized sneakernet protocols. We aim for a simple yet flexible API that enables researchers to concisely specify their routing algorithms, including how and when messages are exchanged and how routing decisions are made. For example, in our initial prototype of Cadence, we can specify probabilistic broadcast, random walk, and geographic-based routing protocols each in only a few lines of code.

CSET 2023, August 07–08, 2023, Marina del Rey, CA, USA

ACM ISBN 979-8-4007-0788-9/23/08…$15.00
https://doi.org/10.1145/3607505.3607507

[1]Traditionally, sneakernets have been used to convey passing messages between physical devices, e.g., disks or USB drives. Here, we use the term to denote protocols that use device-to-device communication links without centralized infrastructure. Sneakernets are also sometimes referred to as *opportunistic networks*.

Behavior modeling. The simulator should not assume homogeneous behavior and should instead allow for different types of nodes (participants), including potentially malicious actors.

Mobility modeling. The core functionality of Cadence is its ability to simulate messaging in a dynamic network consisting of mobile, human users. Cadence exposes a simple API for importing various human movement datasets so that experiments can be conducted using multiple dynamic graphs.

Adversarial settings. Cadence allows for the integration of adversarial settings, such as malicious operators, to evaluate the robustness of protocols against active and passive adversaries that may seek to disrupt the opportunistic network.

Metrics. Cadence collects statistics throughout the simulation and features an interactive reporting module in which investigators can examine the results of their experiments. We aim to maintain a simple statistics collection interface that enables researchers and/or developers to easily augment Cadence to support additional metrics and reports.

Performance. Cadence is designed to operate on computing platforms with varying resources. We are constructing Cadence in Golang due to the language's strong support of parallelism. The simulation speed should scale proportionally to the computation resources on the host machine.

Repeatability and scientific validity. We aim for repeatable experiments and design Cadence to allow deterministic execution.

In what follows, we describe our initial design and experience with the Cadence simulator, which we will release as free, open-source software.

2 RELATED WORK

Sneakernets are a form of *opportunistic networks*—delay-tolerant networks in which message exchanges occur only when mobile devices come into contact with each other. Opportunistic networks have received considerable attention over the past several decades, often motivated by the potential of vehicular networks [17, 28]. Huang et al. survey opportunistic networks [13], and Sachdeva and Dev provide a more recent retrospective on the proposed systems and their deployments [24].

There already exist several simulators for modeling mobility and communication over opportunistic networks. GEMSim [25] simulates the activities of vehicles and pedestrians (using 5.2 million agents) on a GPU in several minutes. Guzman et al. [10] propose a robot mobility simulator for the research of robotic movements. ANVEL [9] is a simulator for unmanned ground vehicles. HumaNets [1] describes a simulator for a specific human movement protocol, but unfortunately, their simulator is not publicly available. Dede et al. survey the few available opportunistic network simulators [8]. Most notable among these—and most similar to Cadence—is Keränen's Opportunistic Network Environment (ONE) simulator [15, 16]. Like Cadence, the ONE simulator is designed to be an extensible platform for modeling different routing protocols.

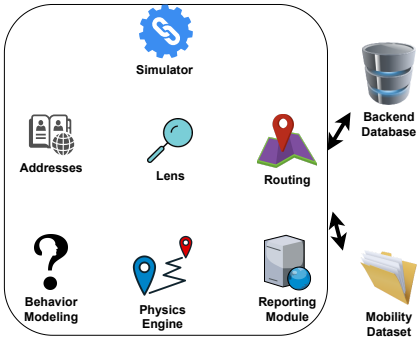

Figure 1: High-level architecture of Cadence.

However, ONE appears to be no longer maintained (the latest release is eight years old) and does not offer the very fine-grained customizations that Cadence supports, including Cadence's ability to maintain arbitrary node state and codify highly customizable routing logic and adversarial behavior.

More generally, understanding human mobility has received considerable and long-standing attention from the research community [3, 6, 11, 12, 17, 18, 20, 23, 28–30]. Mobility research spans from measuring vehicular mobility [17, 28] to human mobility datasets [3, 30] and even to animal mobility [11]. In recent years, there has been a surge in mobility data research focusing on COVID-19 and the role that human mobility plays in the spread of the disease [6, 12, 18, 20].

3 SYSTEM DESIGN

The high-level architecture of Cadence is shown in Figure 1. Cadence is composed of five main modules: lenses, a physics engine, a simulation engine, a reporting service, and a backend database. The lens module preprocesses mobility datasets and normalizes geographic locations (explained in more detail below). It is designed to be extensible to support a variety of mobility datasets. The physics engine assists in determining *encounter points*—locations and times in which two nodes are considered to be in contact. The event-driven simulation engine manages the core logic of the simulation, including node and routing behaviors, and is also designed with a simple API to enable easy customization. The reporting service allows graphic visualizations for analysts and researchers. The database maintains imported datasets and simulation results; results (e.g., dates and location of message exchanges and deliveries) are stored in a simple database schema to enable researchers to quickly construct queries that are not already supported by the reporting service.

Cadence is written in Golang and leverages the language's strong support for parallelism to achieve high simulation efficiency. In the following subsections, we describe Cadence's core components and concepts in more detail.

3.1 Lenses

The lens module, which imports and preprocesses mobility datasets, exposes a simple API as shown in Figure 2. The `Import()` function parses file(s) from a mobility dataset and imports *events*

```
type Lens interface {
    Init(logger *logger.Logger)

    Import(path string, dataSetName string) error

    // gets the type of location used for this dataset
    GetLocationType() model.LocationType
}
```

Figure 2: Lens API.

into the database. An event is a reported position and consists of a node identifier, a location, and a time. The lens logic normalizes geographic locations; in more detail, locations that are specified using latitude, longitude, and (optionally) altitude are converted to three-dimension Euclidean space according to the World Geodetic System 1984 [21], a standardized approach used for terrestrial navigation. This enables more performant simulations by reducing the cost of performing distance calculations. We have implemented lenses for the Geolife [30] and Cabspotting [22] datasets.

3.2 Physics Engine

Mobility datasets are often sparse, reporting nodes' locations only sporadically. The physics engine uses these discrete location events and infers nodes' locations at any given time using customizable heuristics. Cadence currently supports a linear movement model: given a node n and two consecutive events $(t_i, \vec{v}_i)$ and $(t_j, \vec{v}_j)$ from the mobility dataset where $t_i < t_j$ are times and $\vec{v_x}$ is the location of the node at time t_x, Cadence infers a node's position at time t_α where $t_i < t_\alpha < t_j$ as:

$$\vec{v_\alpha} = \vec{v_i} + (t_\alpha - t_i)\frac{|\vec{v_j} - \vec{v_i}|}{t_j - t_i}.$$

The physics engine also determines when *encounters* take place. Cadence allows users to declare the criteria for what constitutes an encounter. Currently, we support a distance criterion in which the user specifies a maximum distance between nodes for an encounter to take place. (Cadence users specify encounter conditions via a simple JSON file. Appendix A presents an example.) Since the mobility datasets are sparse and position information is not specified for all nodes for all times in the mobility dataset, the physics engine must also compute the times at which encounters occur. This involves taking the derivative of a variant of the above equation; the details are omitted for brevity. In summary, Cadence determines the closest encounter point between any two nodes during a given time period and then considers whether their distance meets the encounter criteria. If an encounter is determined to occur, Cadence applies the routing logic (see §3.3).

3.3 Routing

We are constructing Cadence to be capable of simulating a variety of sneakernet protocols. To provide easy extensibility, we constructed a simple addressing scheme and routing API:

Address versatility. Cadence supports multiple forms of addressing, including addressing messages toward a particular node identifier (i.e., unicast) and routing messages toward geographic locations (i.e., geocast).

```
type Logic interface {
  Init(log *logger.Logger)

  // stores the initial copy of a message at a node
  PlaceMessage(id model.NodeId, message *Message)

  // callback function for every time a node finds itself in a new position
  // `b` is the marshalled form of the location
  NewPositionCallback(nodeid model.NodeId, t model.LocationType, b []byte)

  // potentially does a message exchange between nodes when an encounter occurs.
  HandleEncounter(encounter *model.Encounter,
       messageDBChan chan *model.MessageDB,
       receivedmessageDBChan chan *model.DeliveredMessageDB) float32

  // ... supporting routines omitted for brevity ...
}
```

Figure 3: Routing logic API.

Routing versatility. The routing in our simulator is controlled by the logic engine. The logic engine is invoked whenever a node updates its position or an encounter occurs between two nodes (see §3.2). The logic engine implements the interface listed in Figure 3.

Researchers using Cadence implement the above interface to create their custom routing logic. The `PlaceMessage` function bootstraps messaging by specifying the origination of a message. Cadence supports an arbitrary number of messages during simulation. The messages themselves are specified in a simple JSON format that lists their creation time, originating node, destination, and (optionally) payload. See Appendix B for an example message specification.

The `NewPositionCallback` callback function is called whenever a node is in a new position. This is useful for simulating certain protocols (e.g., Aviv et al.'s HumaNets protocol [1, 2]) in which nodes update their internal states whenever their positions change.

Cadence invokes the `HandleEncounter` function whenever a node encounters another node. The `encounter` variable passed into this function embeds the time and location of the encounter, as well as the identity of the encountered node. Cadence is extremely flexible in terms of what happens during an encounter: implementing a routing protocol involves encoding the logic of what happens during an encounter, with a potential effect of that logic being the movement of a message between the two encountering nodes. For example, for a simple flooding algorithm, the `HandleEncounter` function determines if the encountered node already has a copy of the message; if not, it copies it to the encountered node. For random walk, the same occurs, but the `HandleEncounter` function also removes it from the node that previously carried the message.

The logic engine is versatile and can consider different factors for allowing or prohibiting a message transfer:

General factors. There are general factors that the logic engine can examine during an encounter, including:

- *Distance:* The distance between the two nodes;
- *Probabilistic:* Either a global, per message, or per encounter probability. This can be used to model probabilistic routing protocols as well as unreliable messaging channels; and
- *Connection duration:* The duration of a connection. When two devices synchronize, there is a connection that terminates after a period of time. A too-short connection may prevent some/all of the messages from transferring between the nodes. Connections may be torn down intentionally or

```
type Attacker interface {
  Init(log *logger.Logger)

  // runs an attack of the attacker node, using its queue of messages,
  // during an encounter
  Attack(encounter *model.Encounter, attacker model.NodeId,
     attackerMap *sync.Map)

  // ... supporting routines omitted for brevity ...
}
```

Figure 4: Attacker API.

due to physical obstacles or adversarial agents. Cadence models encounter durations, and support routing policies that use this duration as a factor to determine the success or failure of a message transfer.

Device factors. Cadence also supports protocols that consider device factors, including:

- *Battery consumption:* If one of the participants in the encounter has a low battery, a message may not transfer; and
- *Memory usage:* Messages consume storage and memory, the lack of which could prevent the transfer of a message.

3.4 Behavior Modeling

In Cadence, the movement behavior of nodes is dictated by the mobility dataset in use. For example, the Geolife dataset describes humans walking in the street [30], while the Cabspotting dataset describes the routes of cab drivers [22]. These movements are considered "ground truth" and immutable, and reflect human movement networks in which messages are exchanged while people go about their ordinary routines; that is, we model protocols in which humans do not intentionally alter their movements in order to participate in the sneakernet protocol.

However, the behavior of each node (or more specifically, the software running the sneakernet protocol) is more fluid. Cadence allows customization of node *behaviors*. Such functionality can be used, for example, to model different app behaviors on different devices or to simulate adversarial nodes.

To date, we have introduced two attacker models: a flooder and a dropper. The flooder injects spurious messages whenever it meets a new node. A sufficiently large flooder population can create a version of the Coremelt attack [26], a denial-of-service attack that aims to disrupt the entire network (as opposed to a particular victim). The dropper, on the other hand, does not pass any message to any node and instead drops every message it receives.

New (adversarial) behaviors can be integrated into the simulator by implementing the attacker API listed in Figure 4. The `Attack()` function specifies the attacker's behavior (e.g., changing the target of a message, dropping a message, etc.) when it is part of an encounter.

3.5 Reporting Module

Finally, Cadence operates a web service that allows researchers to run various reports and plot statistics regarding the simulation runs. Currently supported reports include plots of delivery rates, contact rates, and node lifetimes.

As with other components of the simulator, we are designing the reporting module to be easily extensible. Adding a new report requires registering a unique URL (e.g., "/avg-throughput") and an event handler, the latter of which implements the desired reporting functionality. Formatting reports is accomplished through Golang's built-in HTML template language.

We emphasize that the reporting module is intended to provide easy access to what we anticipate will be the most commonly performed types of data analyses. Alternatively, researchers can perform their own forms of analyses by directly retrieving simulation outputs from the database and examining Cadence log output.

4 DISCUSSION

Cadence is an important component of our larger ongoing effort to measure the efficacy of various decentralized routing protocols in the face of nation-state censorship and/or widespread Internet outages. We have already implemented a number of routing protocols and adversarial behaviors across several human mobility datasets using Cadence, and are using the results of our simulations to inform the development of new apps for fully decentralized, censorship-resistant messaging.

However, Cadence is still a work-in-progress with several planned improvements: We are exploring methods of optimizing our code and improving simulation speed. Mobility datasets are often enormous, and computing encounters and simulating message exchanges between hundreds of nodes over the course of timespans of up to several years yields very long simulation execution times. The challenge here is determining how to best balance memory requirements (since the datasets are large and there may be millions of messages in transit) with computing speed. Our current approach uses a traditional relational database backend (to maximize flexibility, we use an object-relational mapper to abstract away the particular database being used) with significant indexing in place to efficiently retrieve data. While this limits memory usage, the database has become a bottleneck in our simulations. We are considering switching to memory-backed data stores (e.g., Redis) to reduce I/O bottlenecks and decrease simulation times.

Second, we are planning to enable the operation of different protocols in parallel. Currently, although Cadence is itself highly parallelized, the simulator supports only a single configuration per execution. The ability to run larger simulations with "parallel worlds" and then compare their results is a planned future enhancement.

Finally, we plan to focus more attention on documentation, especially as it pertains to extending Cadence to support additional routing protocols and node behaviors.

Our goal is to support cybersecurity experimentation and testing of sneakernet routing protocols. Towards that aim, we are releasing Cadence as free open-source software, and plan to do so in conjunction with the publication of this paper. We note that our project has several years of funding and has dedicated personnel for developing, improving, and maintaining Cadence throughout that time period. We also plan to release Docker-based releases (i.e., "snapshots in time of Cadence's operating environment") for ensuring that Cadence remains usable for the long term.

5 AVAILABILITY

Cadence is available as free and open-source software and can be downloaded from https://github.com/GUSecLab/cadence.

ACKNOWLEDGMENTS

This material is based upon work supported by DARPA under Contract No. FA8650-22-C-6424. Any opinions, findings, conclusions, or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of DARPA.

REFERENCES

[1] Adam J Aviv, Matt Blaze, Micah Sherr, and Jonathan M Smith. 2014. Privacy-aware message exchanges for HumaNets. *Computer communications* 48 (2014), 30–43.

[2] Adam J. Aviv, Micah Sherr, Matt Blaze, and Jonathan M. Smith. 2012. Privacy-Aware Message Exchanges for Geographically Routed Human Movement Networks. In *European Symposium on Research in Computer Security (ESORICS)*.

[3] Hugo Barbosa, Marc Barthelemy, Gourab Ghoshal, Charlotte R James, Maxime Lenormand, Thomas Louail, Ronaldo Menezes, José J Ramasco, Filippo Simini, and Marcello Tomasini. 2018. Human mobility: Models and applications. *Physics Reports* 734 (2018), 1–74.

[4] Archie Bland. 2014. FireChat–the messaging app that's powering the Hong Kong protests. *The Guardian* 29 (2014).

[5] Briar Project. 2023. Briar: Secure messaging, anywhere. https://briarproject.org/.

[6] Caroline O Buckee, Satchit Balsari, Jennifer Chan, Mercè Crosas, Francesca Dominici, Urs Gasser, Yonatan H Grad, Bryan Grenfell, M Elizabeth Halloran, Moritz UG Kraemer, et al. 2020. Aggregated mobility data could help fight COVID-19. *Science* 368, 6487 (2020), 145–146.

[7] Brendan Burns, Oliver Brock, and Brian Neil Levine. 2008. MORA routing and capacity building in disruption-tolerant networks. *Ad Hoc Networks* 6, 4 (June 2008), 600–620. https://doi.org/10.1016/j.adhoc.2007.05.002

[8] Jens Dede, Anna Förster, Enrique Hernández-Orallo, Jorge Herrera-Tapia, Koojana Kuladinithi, Vishnupriya Kuppusamy, Pietro Manzoni, Anas bin Muslim, Asanga Udugama, and Zeynep Vatandas. 2018. Simulating Opportunistic Networks: Survey and Future Directions. *IEEE Communications Surveys & Tutorials* 20, 2 (2018), 1547–1573. https://doi.org/10.1109/COMST.2017.2782182 Conference Name: IEEE Communications Surveys & Tutorials.

[9] Phillip J Durst, Christopher Goodin, Chris Cummins, Burhman Gates, Burney Mckinley, Taylor George, Mitchell M Rohde, Matthew A Toschlog, and Justin Crawford. 2012. A real-time, interactive simulation environment for unmanned ground vehicles: The autonomous navigation virtual environment laboratory (ANVEL). In *2012 Fifth international conference on information and computing science*. IEEE, 7–10.

[10] José Luis Guzmán, Manuel Berenguel, Francisco Rodriguez, and Sebastián Dormido. 2008. An interactive tool for mobile robot motion planning. *Robotics and Autonomous Systems* 56, 5 (2008), 396–409.

[11] Daniel B Hayes and Michael J Monfils. 2015. Occupancy modeling of bird point counts: implications of mobile animals. *The Journal of Wildlife Management* 79, 8 (2015), 1361–1368.

[12] Tao Hu, Siqin Wang, Bing She, Mengxi Zhang, Xiao Huang, Yunhe Cui, Jacob Khuri, Yaxin Hu, Xiaokang Fu, Xiaoyue Wang, et al. 2021. Human mobility data in the COVID-19 pandemic: characteristics, applications, and challenges. *International Journal of Digital Earth* 14, 9 (2021), 1126–1147.

[13] Chung-Ming Huang, Kun-chan Lan, and Chang-Zhou Tsai. 2008. A Survey of Opportunistic Networks. In *22nd International Conference on Advanced Information Networking and Applications - Workshops (aina workshops 2008)*. 1672–1677. https://doi.org/10.1109/WAINA.2008.292

[14] Sanja Kelly, Mai Truong, Adrian Shahbaz, and Madeline Earp. 2016. *Silencing the Messenger: Communication Apps Under Pressure*. Freedom on the Net. Freedom House. Available at https://freedomhouse.org/report/freedom-net/2016/silencing-messenger-communication-apps-under-pressure.

[15] Ari Keranen. 2008. Opportunistic network environment simulator. *Special Assignment report, Helsinki University of Technology, Department of Communications and Networking* (2008).

[16] Ari Keränen, Jörg Ott, and Teemu Kärkkäinen. 2009. The ONE simulator for DTN protocol evaluation. In *Proceedings of the 2nd International Conference on Simulation Tools and Techniques (Simutools '09)*. ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering), Brussels, BEL, 1–10. https://doi.org/10.4108/ICST.SIMUTOOLS2009.5674

[17] Xiangjie Kong, Feng Xia, Zhaolong Ning, Azizur Rahim, Yinqiong Cai, Zhiqiang Gao, and Jianhua Ma. 2018. Mobility dataset generation for vehicular social networks based on floating car data. *IEEE Transactions on Vehicular Technology* 67, 5 (2018), 3874–3886.

[18] Moritz UG Kraemer, Chia-Hung Yang, Bernardo Gutierrez, Chieh-Hsi Wu, Brennan Klein, David M Pigott, Open COVID-19 Data Working Group†, Louis Du Plessis, Nuno R Faria, Ruoran Li, et al. 2020. The effect of human mobility and control measures on the COVID-19 epidemic in China. *Science* 368, 6490 (2020), 493–497.

[19] Arian Akhavan Niaki, Shinyoung Cho, Zachary Weinberg, Nguyen Phong Hoang, Abbas Razaghpanah, Nicolas Christin, and Phillipa Gill. 2020. ICLab: A Global, Longitudinal Internet Censorship Measurement Platform. In *IEEE Symposium on Security and Privacy (SP)*.

[20] Pierre Nouvellet, Sangeeta Bhatia, Anne Cori, Kylie EC Ainslie, Marc Baguelin, Samir Bhatt, Adhiratha Boonyasiri, Nicholas F Brazeau, Lorenzo Cattarino, Laura V Cooper, et al. 2021. Reduction in mobility and COVID-19 transmission. *Nature communications* 12, 1 (2021), 1090.

[21] Office of Geomatics. 2014. *World Geodetic System 1984 (WGS 84)*. Standard NGA.STND.0036_1.0.0_WGS84. National Geospatial-Intelligence Agency. Available at https://earth-info.nga.mil/?dir=wgs84&action=wgs84.

[22] Michal Piorkowski, Natasa Sarafijanovic-Djukic, and Matthias Grossglauser. 2022. CRAWDAD epfl/mobility. https://dx.doi.org/10.15783/C7J010

[23] Chiara Renso, Stefano Spaccapietra, and Esteban Zim'anyi. 2013. *Mobility data*. Cambridge University Press.

[24] Rahul Sachdeva and Amita Dev. 2021. Review of opportunistic network: Assessing past, present, and future. *International Journal of Communication Systems* 34, 11 (2021), e4860.

[25] Aleksandr Saprykin, Ndaona Chokani, and Reza S Abhari. 2019. GEMSim: A GPU-accelerated multi-modal mobility simulator for large-scale scenarios. *Simulation Modelling Practice and Theory* 94 (2019), 199–214.

[26] Ahren Studer and Adrian Perrig. 2009. The coremelt attack. In *Computer Security–ESORICS 2009: 14th European Symposium on Research in Computer Security, Saint-Malo, France, September 21-23, 2009. Proceedings 14*. Springer, 37–52.

[27] Ram Sundara Raman, Prerana Shenoy, Katharina Kohls, and Roya Ensafi. 2020. Censored Planet: An Internet-wide, Longitudinal Censorship Observatory. In *Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security*. ACM, Virtual Event USA, 49–66. https://doi.org/10.1145/3372297.3417883

[28] Sandesh Uppoor, Oscar Trullols-Cruces, Marco Fiore, and Jose M Barcelo-Ordinas. 2013. Generation and analysis of a large-scale urban vehicular mobility dataset. *IEEE Transactions on Mobile Computing* 13, 5 (2013), 1061–1075.

[29] Kai Zhao, Sasu Tarkoma, Siyuan Liu, and Huy Vo. 2016. Urban human mobility data mining: An overview. In *2016 IEEE International Conference on Big Data (Big Data)*. IEEE, 1911–1920.

[30] Yu Zheng, Hao Fu, Xing Xie, Wei-Ying Ma, and Quannan Li. 2011. *Geolife GPS trajectory dataset - User Guide* (geolife gps trajectories 1.1 ed.). https://www.microsoft.com/en-us/research/publication/geolife-gps-trajectory-dataset-user-guide/

A ENCOUNTER CONDITIONS

Figure 5 depicts an example encounter condition file with two conditions. An encounter must meet both conditions (that is, the conditions in the file should be considered using logical conjunction). In the example in Figure 5, the first condition (of type `distance`) specifies that an encounter takes place when nodes are less than 200 meters apart. The second constraint specifies a random variable such that the condition is true with a probability of 0.25.

```
{
  "conditions" : [
    {
      "name" : "distance200m",
      "type" : "distance",
      "params" : {
        "dist" : 200.0
      }
    },
    {
      "name" : "probability25",
      "type" : "probability",
      "params" : {
        "prob" : 0.25
      }
    }
  ]
}
```

Figure 5: Example encounter condition containing two conditions.

B EXAMPLE MESSAGE DEFINITION FILE

An example message definition file is provided in Figure 6. Here, the simulation will consider two messages (with unique IDs `1` and `2`) that respectively originate from nodes `10` and `30` and are addressed to nodes `20` and `40`. The messages are created at time `100` and `200`, respectively.

```
[
  {
    "id" : 1,
    "sender" : 10,
    "destination" : 20,
    "type" : 10,
    "time": 100,
    "payload": "Hello.",
    "shards": 0,
    "shardid": 0,
    "ms": false,
    "ttl_hops": 20,
    "ttl_secs": 1000000,
    "size": 30
  },
  {
    "id" : 12,
    "sender" : 30,
    "destination" : 40,
    "type" : 10,
    "time": 200,
    "payload": "Goodbye.",
    "shards": 0,
    "shardid": 0,
    "ms": false,
    "ttl_hops": 20,
    "ttl_secs": 1000000,
    "size": 30
  }
]
```

Figure 6: Example message definition file containing two message definitions.