2021 IEEE International Conference on Pervasive Computing and Communications Workshops and other Affiliated Events (PerCom Workshops) | 978-1-6654-0424-2/21/$31.00 ©2021 IEEE | DOI: 10.1109/PERCOMWORKSHOPS51409.2021.9430977

A Latency-Defined Edge Node Placement Scheme for Opportunistic Smart Cities

Oluwashina Madamori*, Esther Max-Onakpoya*, Gregory D. Erhardt†, Corey E. Baker*
*Department of Computer Science, †Department of Civil Engineering, University of Kentucky, Lexington, KY USA
shina@uky.edu, esther.max05@uky.edu, greg.erhardt@uky.edu, baker@cs.uky.edu

***Abstract*—Smart city projects have the potential to improve the management of environmental and public infrastructure. However, the operational and capital expenditures of smart cities can prevent cities from becoming smarter. A notable factor that influences the cost is providing cellular Internet connectivity to IoT devices. 5G has been proposed as a possible solution, but projections show that 5G will not be able to support the load of billions of IoT devices coming online. To mitigate this, people, vehicles, and other nodes in transportation networks can be exploited to transmit non-urgent data by leveraging device-to-device communication in order to reduce cellular connectivity costs associated with smart city sensors. Hence, this paper addresses cost-effective edge node placement in smart cities that opportunistically leverage public transit networks. We introduce an algorithm that selects a set of edge nodes that provide minimal delivery delay within a budget. The algorithm is evaluated for two public transit network data-sets: Chapel Hill, North Carolina and Louisville, Kentucky and results show that our algorithm outperforms betweeness and in-degree centrality metrics with a reduction in latency of over 20 minutes.**

***Index Terms*—smart cities, opportunistic networks, delay tolerant networks, internet of things, edge computing, wireless**

I. Introduction

Development of smarter cities has been proposed as a means of combating the challenges arising from the increasing rate of urbanization within many cities in the world [1]. One major characteristic of most smart city designs is the deployment of a vast number of IoT devices/sensors across the city to monitor and sometimes control the state of public infrastructure such as water and gas pipes [2]. These IoT devices, which include weather sensors and traffic monitors, generate large amounts of data that need to be forwarded to the Cloud for processing and storage. To achieve this, deployed IoT devices typically rely on cellular connectivity.

However, the additive operating cost incurred from each sensor's cellular subscription plan is high [3]. The price of deploying and maintaining smart city projects is a huge deterrent for city officials, especially when the sustainability and impact of such projects are uncertain [4]–[6]. In addition, since sensors generate large amounts of data and connect to the same base stations that facilitate cellular connectivity for personal mobile devices, using cellular networks for smart city data can quickly lead to network congestion and poor user experience. Though 5G has been proposed as a viable solution, projections show that 5G will not be able to support the load of billions of IoT devices coming online [7]–[9].

Hence, there is need for cost-effective smart city communication networks that reliably and efficiently forward sensor data to the cloud without over burdening cellular infrastructure. In response to aforementioned needs, various researchers have historically explored delay tolerant networks (DTNs) or opportunistic networks for smart city applications that can tolerate high latency [10]–[14]. Messages are delivered with some delay which is directly correlated with the layout, density, and mobility of nodes in the network [15], [16]. Consequently, a question that has been marginally addressed is: **where should edge nodes be placed to minimize latency in a low-cost smart city?**

Edge placement in opportunistic networks is not intended to perform better than 5G or other centralized communication schemes, but rather offer a low-cost alternative for delivering non-urgent data, thus enabling municipalities to become smart cities at a fraction of the cost. In this paper we: (i) introduce the Minimal Delivery Delay (MDD) edge node placement problem; (ii) formulate the MDD problem as an Influence Maximization problem; (iii) develop an approximation algorithm for solving the MDD problem; (iv) compare the results of our algorithm with traditional network centrality measures and finally; (v) develop a simulation tool that models a vehicular communication network consisting of data generators (sensors), intermediate carriers (buses) and destination devices (edge nodes/gateways) within any real-world city, by directly using transit network information provided in the General Transit Feed Specification (GTFS) format.

The rest of the paper is structured as: Section II discusses related work; Section III introduces the network model; Section IV defines the MDD problem and describes its solution; Section V explains the simulation design and environment; Section VI offers the numerical evaluation; and finally Section VII discusses and concludes the work.

II. Related Works

Various researchers have investigated the opportunistic use of vehicular transportation networks for data forwarding in smart city communication networks. The network architecture often consists a set of vehicular data mules (e.g. buses, boats, train, etc) that encounter IoT devices, opportunistically collect sensed data and deliver the data to edge nodes with wired Internet connectivity [17], [18]. However, the vast

This material is based upon work supported by the National Science Foundation under Grant No. 1952181.

majority of research in this area has focused on the design of data forwarding schemes. Some works have proposed new routing schemes that harness the quasi-deterministic nature of public transportation networks and utilize metrics such as intercontact times to reduce latency and improve delivery [10], [11]. Others have explored routing in the context of different wireless communication media such as, LoRa and WiFi [17]. Some others have investigated routing and forwarding schemes in the context of preserving privacy [14]. Nevertheless, none of the aforementioned work addresses the question of edge node placement.

While placement algorithms are typically unique to the network, certain techniques from other network domains are relevant to this problem. A previous work on node placement optimization exploits the principle of submodularity to tackle the problem of sensor placements in water distribution networks [19]. In another work, the authors explore several optimization techniques for access point placement in wireless mesh networks [20]. Unlike prior research, this work formulates the edge node placement problem as an Influence Maximization problem, provides an algorithmic solution, and evaluates the algorithm using GTFS-data derived from real transits in multiple cities. The resulting solution can be applied to low-cost smart cities that leverage opportunistic networks.

1) Classic Centrality Analysis: In the field of network analysis, several popular centrality measures exist. These centrality measures are usually computed as real-valued functions and reflects a node's significance or importance within the respective network [21]. Centrality measures have been used in many kinds of networks including the Internet, social networks, biological networks, and transportation networks. Unfortunately, centrality measures work best with simple static networks [22] and not dynamic networks. Since our network model is more complex, containing not just nodes (stops) and edges (trips), but also vehicle schedule information across each node, centrality measures may not prove very useful for optimizing edge node placement.

2) Evaluation of Opportunistic Networks: A number of real-world test-beds have been developed to facilitate research in the field of vehicular communication networks and vechicular opportunistic networks [3], [23]. Although real-world deployments of various network architectures provide the most accurate results, oftentimes these results are not sufficiently generalized due to insufficient geographic diversity as well as the limited scale of the experiments. In addition, such deployments are typically financially expensive. A popular alternative to real-world evaluations has been the use of mathematical models and simulations. Simulations are advantageous due to their cost-effectiveness, scalability, and geographical and hardware heterogeneity. Examples of some widely used open-source generic simulators in the field of vehicular communication and DTN include the ONE simulator [16] and ns-3 [24]. However, it can be difficult to fully incorporate real-world data into currently available simulation environments, or adapt the environment to match the designed network model. Hence, this work develops a simulator that closely emulates the movement of real world transit vehicles, using GTFS, in order to evaluate the performance of our algorithm across many cities.

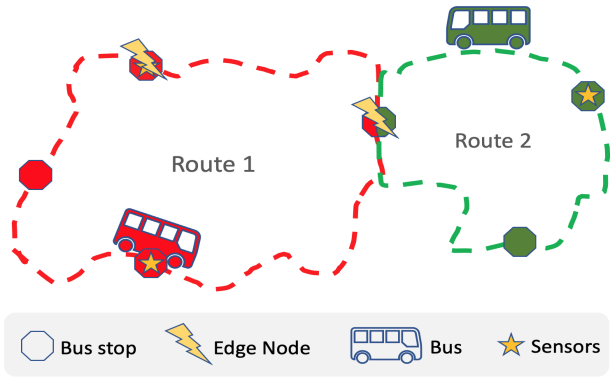

Figure 1. Simplified Example of Network Architecture

III. Model

The entities within the smart city are depicted in Figure 1 and described below:

1) Bus routes: Every public transit network has a list of predefined routes on which buses move/operate. Each route is primarily defined by its list of bus-stops. A route also contains a list of trips which specifies the arrival and departure times for buses operating on that route, as well as the sequence of stops the bus moves through during each trip. Note that a stop may service more than one route.

2) Sensors: Sensors are located at select bus stops. Each sensor generates non-urgent data at a specific periodic rate around-the-clock. The data is stored locally until it can be forwarded to a bus. Every sensor is equipped with a device that allows it to opportunistically connect to a bus when the bus is within a specified geographic range of the sensor. In addition, we are only considering sensors with data packets that are small enough to justify an assumption of transmitting to a bus in a short time window. Local storage of a sensor is considered to be infinite, since data is expected to be picked up by a bus periodically, thereby eliminating the need for a policy for dropping packets.

3) Edge nodes: These are stationary devices which, when active, are continuously connected and able to forward data directly to remote servers via the Internet for post processing and analysis. They act as the destination for all data generated by sensors. Not all bus stops are edge nodes, rather edge nodes are placed at selected bus stops. Similar to sensors, edge nodes are assumed to always be within transmission range of buses traveling on routes for the stop on which the edge node is located. Each edge node is also equipped with a device that allows it to opportunistically connect to a bus when the bus is within a specified geographic range of it.

4) Buses: These move along predefined routes on a fixed schedule. Hence, the specific geographic position of buses moving on a specific route at any time can be estimated. Each

bus is equipped with the necessary hardware to connect to sensors, retrieve and store sensor data packets, and forward the data to edge nodes. We currently do not consider the buffer/queue sizes of data in buses and so buses are assumed to have buffer/queue sizes of infinity since they are expected to deliver data to at least one edge node within a day (buses park at transit stations which are expected to have an active edge node), so a drop-policy for stored data is not necessary.

IV. Edge node Selection Problem

In this section, we present Minimal Delivery Delay (MDD) problem. The objective of MDD *is to select the set of locations, $G \subset S$, to place edge nodes such that the average network latency for data generated across the network is minimized, without exceeding a budget constraint, k*. Where, the budget constraint refers to the number of edge nodes that can be added to the network and S is the set of stops. In considering network latency, we account for both delivered and undelivered data. We also assign a penalty value to data that is undelivered, which is explained in Section IV-A. Further, the edge node selection process is conducted without prior knowledge of the stops in which the sensors will be placed in the city.

We formulate the problem as an Influence Maximization (IM) problem, which is defined: Given a network with n nodes and given a propagation process on that network, choose a set of nodes called the *seed set* D of size $b < n$ that maximizes the number of nodes in the network that are ultimately influenced [25]. However, our problem differs from the traditional IM problem because the set of nodes we want to select are not seed/source nodes but destination nodes. Hence, we consider each potential edge node location, $s \in S$ to possess an influence value, $\sigma(s)$, which describes its impact on the network latency if it is added to an existing set of edge nodes. Thus, our problem objective is to select a set of edge nodes, G, below a specified cardinality, $k < |S|$, that together decreases the network latency the most. Since, $G = D$, $k = b$ and $|S| = n$, the problem can be defined as an influence maximization problem. Given a decision variable, $y_s \in \{0, 1\}$, which indicates whether a stop, $s \in S$, was selected as an edge node, a budget constraint, k and an influence value, $\sigma(s)$ associated with every stop $s \in S$, the ILP formulation is thus:

$$\max \quad \sum_{s \in S} y_s \sigma(s) \tag{1}$$
$$\text{s.t.}$$

$$\sum_{s \in S} y_s \leq k \tag{2}$$

$$y_s = \{0, 1\} \quad \forall\ s \in S \tag{3}$$

Finding the set of edge nodes that maximize influence is NP-Hard. However, the solution can be approximated using the *Greedy* and *Cost-Effective Lazy Forward* (CELF) algorithms [26]. Nevertheless, defining the influence function is critical to the success of the algorithm.

A. Influence Function

Current IM algorithms require an influence function that simulates the propagation process and computes the marginal influence that each potential seed has on the overall propagation. Therefore for our algorithm, we develop an influence function (σ) that computes the expected latency across the network for any potential set of edge nodes.

For a given stop, s, the influence function is defined as:

$$\sigma(s) = T - \frac{1}{|S|-1} \sum_{\mathscr{s} \in S, s \neq \mathscr{s}} (\tau(s, \mathscr{s}) + \mathfrak{t}(s, \mathscr{s})) \tag{4}$$

Where T is the time penalty (maximum latency possible in the network — in our evaluation, we picked a fixed value outside the range of simulation window), $\tau(u, v)$ is the minimum time it takes to get from stop v to stop u and $\mathfrak{t}(s, \mathscr{s})$ is the lag between the time at which data is generated and the time at which the next bus for the route arrives at the stop where the sensor $\mathscr{s}$ is located. From equation 4, one can see that the influence of a bus stop is determined by its network latency. Since the influence value is calculated by subtracting the latency from the time penalty, magnitude of the latency is inversely correlated with the influence value of a bus stop. Consequently, the influence function for a set of edge nodes, G is:

$$\sigma(G) = T - \frac{1}{|S|-G} \sum_{\mathscr{s} \in S, \mathscr{s} \notin G} min(\mathcal{T}(G, \mathscr{s})) \tag{5}$$

Where,

$$\mathcal{T}(G, \mathscr{s}) = \{(\tau(g, \mathscr{s}) + \mathfrak{t}(g, \mathscr{s})) \mid \forall\ g \in G\} \tag{6}$$

Each element in the set, $\mathcal{T}(G, S)$, is the sum of the time it takes for a vehicle to forward data to an edge node and the time it takes for the vehicle to get to the sensor. In addition, given that there is no prior knowledge of the sensor locations, we compute the network latency over multiple sensor placement scenarios rather than just one, through a Monte Carlo simulation. This method increases the likelihood of the influence function returning an unbiased estimation of network latency for a specific set of edge nodes [25]. We generate a set of sensor placement scenarios, in which each scenario is essentially a simulation environment consisting of the transit network graph, where n sensors are randomly placed at various stops, and each sensor is randomly assigned a message generation frequency.

B. Influence Function Submodularity

We postulate that our influence function is a monotonically increasing submodular function.

Definition: Given a finite ground set, S, and a set function f: $2^S \rightarrow \mathbb{R}$, σ is a submodular function if $\forall\ A \subseteq B \subseteq S,\ a \in S \setminus B$.

$$\sigma(A + a) - \sigma(A) \geq \sigma(B + a) - \sigma(B) \tag{7}$$

Let the ground set be the set of bus stops in the network, S where, each edge node placement scenario, G, can be

expressed as a subset of S and the stops in G are edge nodes. Also, let $\sigma(G)$ be the maximum influence that can be acquired from G. From equation 6, if $G = \emptyset$, then the time it takes for any sensor to get to an edge node in G is the time penalty T. The average latency would be T and $\sigma(\emptyset)$ would be 0 (equation 5), indicating σ is normalized. Next, we ask the question: *Is $\sigma(G)$ monotonically increasing?*

Recall that a function σ: $2^S \rightarrow \mathbb{R}$ is monotonically increasing if $\forall\ A \subseteq B \subseteq S,\ \sigma(A) \leq \sigma(B)$. Let $c = B \setminus A$. Suppose $\sigma(B) < \sigma(A)$, then, there has to be a stop, s, which when added to B causes its data latency to be greater than that of A. Since the influence function depends on the stop that provides the minimum latency in B, and $A \cup c$ is equal to B, this statement holds: $min(\mathcal{T}(B,s)) = min(\mathcal{T}(A,s) \cup \mathcal{T}(c,s)$. As such, the minimum latency in B, is either equal to the minimum latency in A or c (whichever is lower). Hence, $min(\mathcal{T}(B,s))$ cannot be greater than $min(\mathcal{T}(A,s))$ [1]. Thus, $\sigma(B) \geq \sigma(A)$ and σ is monotonically increasing. Following this, we ask the question: *Is $\sigma(G)$ submodular?*

Given a single stop, $s \in S$, and the edge node placement scenarios, A, B and a, as described above. When a is added to G, its minimum latency for any single stop s, $min(\mathcal{T}(G + a, s))$, can be rewritten as $min(min(\mathcal{T}(G,s)), min(\mathcal{T}(a,s)))$. This indicates that the result of $min(\mathcal{T}(G + a, s))$ is either equal to $min(\mathcal{T}(G,s))$ or $min(\mathcal{T}(a,s))$.

Case 1: $min(\mathcal{T}(B + a, s)) = min(\mathcal{T}(B,s))$

In this case, the minimum delay derived from the set a has to be greater than that of B for any stop, s. As a result, the marginal influence, derived from the right side of equation 7 is 0 and since the marginal influence derived from the left side of the equation must be non-negative, the conjecture of submodularity holds.

Case 2: $min(\mathcal{T}(B + a, s)) = min(\mathcal{T}(a,s))$

Here, the minimum delay derived from the set B has to be greater[2] than that derived from a for a subset of stops, $L \subset S$. The influence derived by adding a to any placement set G is:

$$\sigma(G + a) = \sigma(G) - \frac{1}{|L|}\sum_{l \in L}(T - min(\mathcal{T}(G,l))) + \frac{1}{|L|}\sum_{l \in L}(T - min(\mathcal{T}(a,l))) \tag{8}$$

The first term represents the influence derived from the original set, the second term represents the influence contributed by L to the original set and the third term represents the marginal influence derived from the added set, a. By rewriting the left and right hand sides of equation 7 using equation 8, it is reduced to:

$$min(\mathcal{T}(A,l)) \geq min(\mathcal{T}(B,l)) \quad \forall\ l \in L \tag{9}$$

[1]This is the case because our model does not assume bus-to-bus communication, all added edge nodes in an edge node placement set provide new elements to $\mathcal{T}(G,s)$)

[2]This is because the influence an edge node placement set has on each stop is added together and averaged to find the influence- Equation 5

Since the minimum latency derived from B cannot be greater than that of A for any $s \in S$, σ is submodular.

C. CELF-MDD Algorithm

Since the problem is reduced to the maximization of a monotone submodular function, the greedy algorithm provides a $(1 - 1/e)$-approximation of the optimal edge node set [26]. Although the greedy algorithm is much quicker than a brute-force approach, it is still very slow when considering the size of actual transit networks. Therefore, we use the cost-effective lazy forward (CELF) approach. CELF significantly reduces the running time by exploiting the submodular property of our influence function while still providing the same solution set as the Greedy algorithm [27]. It eliminates the need to compute the marginal influence value of all potential edge nodes at each iteration.

In the first round, we calculate the influence for all stops (like Greedy), select the stop with the greatest influence, and store the influence values of the other stops in a max heap. In subsequent iterations, the marginal influence of the top stop in the heap is computed and added back to the heap. If the stop remains at the top of the heap, then it must have the highest marginal influence of all remaining stops, due to the sub-modular property of the influence function. If a different stop is on top of the heap, the process continues until a stop remains on top after two iterations, after which that stop is added to the edge node set. This process is repeated until the edge node budget has been met.

Algorithm 1 CELF Minimal Delivery Delay (CELF-MDD)

Input — graph N, influence function σ, budget k
Output — selected edge node set G

procedure CELF-MDD(N, σ, k)
  $G \leftarrow \emptyset$
  $Q \leftarrow \emptyset$
  for $v \in N.nodes$ **do**
    $u \leftarrow v$
    $u.gain = \sigma(\{v\})$
    `add` u `to` Q `in descending order`
  while $|G| < k$ **do**
    $u \leftarrow Q.top$
    if $u.flag = |G|$ **then**
      $G \leftarrow G \cup \{u\}$
      $Q \leftarrow Q \setminus u$
    else
      $u.gain \leftarrow \sigma(G \cup \{u\}) - \sigma(G)$
      $u.flag \leftarrow |G|$
      `Re-sort` Q `in descending order`
  return G

V. Simulation Design

We developed a simulation tool[3] that models a vehicular communication network consisting of sensors, buses and edge

[3]The code for the simulator is available on GitHub - https://github.com/netreconlab/low_cost_smart_city_optimization

nodes within any real-world city, by directly using real transit network information provided in the General Transit Feed Specification (GTFS) format [28]. GTFS handles information on transit routes, stops, and timetables [28]. Incorporating GTFS information into the simulator enables us to evaluate the performance of our model for many cities around the world.

1) Transit feed to graph conversion: The GTFS transit feed data for a transit agency is converted into a directed graph via an open-source library called *peartree* [29]. The graph contains: (i) Nodes representing stops, with each node containing the departure times for all vehicles from that stop, and (ii) Edges representing a bus path from one stop to another. The edge weight is the average time it takes for a bus to get from one stop to a neighboring stop on a trip.

2) Sensor Placement: Sensors are placed at randomly selected stops in the transit network. Each stop has a maximum of one sensor and the total number of sensors to be placed in the network is defined for each simulation. In addition, each sensor is assigned a frequency at which it generates data.

3) Data Delivery Delay: For each data packet generated at a sensor, the shortest duration it takes for the data packet to get to an edge node is computed. A subgraph (consisting of only stops in a single route) is extracted from the main graph for each route. Next, we iterate through each route the sensor is on and compute the shortest path length from the sensor to any edge node on that route. Since the edge weight is the average travel time for vehicles between a node pair, the computed shortest path length represents the estimated time it takes for a vehicle to forward the data to an egde node after departing from the sensor. The total estimated delay is the sum of the shortest path length and the waiting time. After completing the iterations, the path with the shortest total estimated delay is selected as the path through which the data will travel.

VI. Numerical Evaluation

A. Simulation Setup

In order to evaluate the performance of our algorithm, we make use of the GTFS data of two bus transit agencies: Chapel Hill Transit (CHT) in Chapel Hill, North Carolina, and Transit Authority of River City (TARC) in Louisville, Kentucky [30]. Chapel Hill measures 55 km^2 (21.3 mi^2) with an estimated population of about 60,988. While, Louisville has a population of 620,118 and land area of 171.70 km^2 (66.29 mi^2) [31].The difference in city size also translates to the differences in public transit networks present in both cities (Table I).

Table I
Bus Network Characteristics CHT and TARC

Statistics	**CHT**	**TARC**
Routes	26	46
Stops	571	4391
Total trips	1252	1917
Betweenness centrality avg.	0.04896	0.00829
In-degree centrality avg.	0.00210	0.00025

B. Minimizing Latency

The performance of CELF-MDD was compared to betweenness centrality (BC) and in-degree centrality (IC) in terms of network latency minimization. For each algorithm, the top k stops generated were selected as edge nodes, where k is the budget. The centrality metrics were computed on graph weighted with latency. Simulations were run with the selected edge node set using the simulation parameters outlined in Table II. Consistent with the influence function in IV-A, the upper bound value specified in Table II is assigned as the penalty value (delay) for undelivered data. Figure 2a and 2b show the average network latency of various budgets using each algorithm.

For CHT, we observe that CELF-MDD consistently outperforms both BC and IC by $\approx$ 20 minutes or higher. In addition, there is very little decrease in delay after first 5 edge nodes have been selected when using CELF-MDD. For TARC, we observe that CELF-MDD consistently outperforms both BC and IC by $\approx$ 45 minutes or higher. There is also minimal decrease in delay after the first 9 edge nodes have been selected when using CELF-MDD. For CHT and TARC, the CELF-MDD algorithm can effectively serve the whole network with 5 and 9 well placed edge nodes, respectively.

Table II
Simulation and Scenerio Parameters

Random generator seeds	0:1:100
Simulation time window	1:00:00-24:00:00
Time penalty/maximum-latency (hours)	25
Number of sensors	$30\% \times \lvert stops \rvert$
Sensor data generation frequency (minutes)	$U(1, 120)$
Number of sensor scenarios	5
Number of sensors for scenarios	$U(30, 40)$
Budget	1:1:15

VII. Conclusion

This work addresses the problem of efficient edge node placement in low-cost smart cities that opportunistically utilize public transit networks as data mules. We introduced the MDD optimization problem, formulated it as an IM problem and proposed an IM heuristic that considers the layout of the city and various sensor placement options in minimizing latency. Experiments were carried out using public transport networks in two cities in the United States; Chapel-Hill and Louisville. The results show that our algorithm outperforms traditional centrality measures by reducing network latency at a lower cost, indicating that our algorithm is effective in determining the best locations to place edge nodes such that delivery delay is minimized without exceeding a budget.

References

[1] P. Hayat, "Smart cities: A global perspective," *India Quarterly*, vol. 72, no. 2, pp. 177–191, 6 2016. [Online]. Available: http://journals.sagepub.com/doi/10.1177/0974928416637930

[2] L. Guevara and F. Auat Cheein, "The role of 5g technologies: Challenges in smart cities and intelligent transportation systems," *Sustainability*, vol. 12, no. 16, p. 6469, 2020.

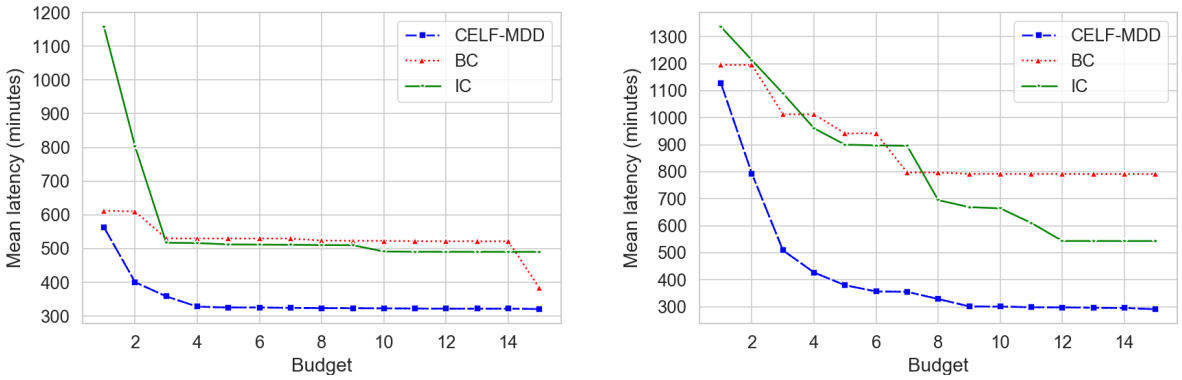

Figure 2. (a) Average network latency for CHT (b) Average network latency for TARC.

[3] J. Paradells, C. Gómez, I. Demirkol, J. Oller, and M. Catalan, "Infrastructureless smart cities. Use cases and performance," *2014 International Conference on Smart Communications in Network Technologies, SaCoNeT 2014*, 2014.

[4] Deloitte, "The challenge of paying for smart cities projects," Tech. Rep., 2018. [Online]. Available: http://dx.doi.org/10.14257/ijfgcn.2014.7.1.15

[5] T. R. Dillahunt and T. C. Veinot, "Getting there: Barriers and facilitators to transportation access in underserved communities," *ACM Trans. Comput.-Hum. Interact.*, vol. 25, no. 5, Oct. 2018. [Online]. Available: https://doi.org/10.1145/3233985

[6] SmartCitiesWorld, "Smart cities : understanding the challenges and opportunities," Tech. Rep.

[7] Cisco, "Cisco annual internet report 2018-2023," 2020. [Online]. Available: https://www.cisco.com/c/en/us/solutions/collateral/executive-perspectives/annual-internet-report/white-paper-c11-741490.pdf

[8] D. Loghin, S. Cai, G. Chen, T. T. A. Dinh, F. Fan, Q. Lin, J. Ng, B. C. Ooi, X. Sun, Q.-T. Ta *et al.*, "The disruptions of 5g on data-driven technologies and applications," *IEEE Transactions on Knowledge and Data Engineering*, vol. 32, no. 6, pp. 1179–1198, 2020.

[9] J. Navarro-Ortiz, P. Romero-Diaz, S. Sendra, P. Ameigeiras, J. J. Ramos-Munoz, and J. M. Lopez-Soler, "A survey on 5g usage scenarios and traffic models," *IEEE Communications Surveys & Tutorials*, vol. 22, no. 2, pp. 905–929, 2020.

[10] O. Choi, S. Kim, J. Jeong, H. W. Lee, and S. Chong, "Delay-Optimal Data Forwarding in Vehicular Sensor Networks," *IEEE Transactions on Vehicular Technology*, vol. 65, no. 8, pp. 6389–6402, 2016.

[11] C. Giannini, P. Calegari, C. Buratti, and R. Verdone, "Delay tolerant network for smart city: Exploiting bus mobility," in *2016 AEIT International Annual Conference (AEIT)*, Oct 2016, pp. 1–6.

[12] C. E. Baker, A. Starke, T. G. Hill-Jarrett, and J. McNair, "In vivo evaluation of the secure opportunistic schemes middleware using a delay tolerant social network," in *Distributed Computing Systems (ICDCS), 2017 IEEE 37th International Conference on*. IEEE, 2017, pp. 2537–2542.

[13] O. Madamori, E. Max-Onapkoya, C. Grant, and C. E. Baker, "Using Delay Tolerant Networks as a Backbone for Low-cost Smart Cities," in *5th IEEE International Conference on Smart Computing (SMARTCOMP)*, Washington D.C., USA, 2019.

[14] T. E. Amah, M. Kamat, K. A. Bakar, W. Moreira, A. Oliveira Jr, and M. A. Batista, "Preparing opportunistic networks for smart cities: Collecting sensed data with minimal knowledge," *Journal of Parallel and Distributed Computing*, vol. 135, pp. 21–55, 2020.

[15] P. Hui and A. Lindgren, "Phase transitions of opportunistic communication," in *Proceedings of the third ACM workshop on Challenged networks*. ACM, 2008, pp. 73–80.

[16] A. Keränen, J. Ott, and T. Kärkkäinen, "The one simulator for dtn protocol evaluation," in *Proceedings of the 2nd international conference on simulation tools and techniques*. ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering), 2009, p. 55.

[17] R. Almeida, R. Oliveira, M. Luís, C. Senna, and S. Sargento, "A multi-technology communication platform for urban mobile sensing," *Sensors*, vol. 18, no. 4, p. 1184, 2018.

[18] F. Raissi, S. Yangui, and F. Camps, "Autonomous cars, 5g mobile networks and smart cities: Beyond the hype," in *2019 IEEE 28th International Conference on Enabling Technologies: Infrastructure for Collaborative Enterprises (WETICE)*. IEEE, 2019, pp. 180–185.

[19] A. Krause, J. Leskovec, C. Guestrin, J. VanBriesen, and C. Faloutsos, "Efficient sensor placement optimization for securing large water distribution networks," *Journal of Water Resources Planning and Management*, vol. 134, no. 6, pp. 516–526, 2008.

[20] F. Xhafa, C. Sánchez, A. Barolli, and M. Takizawa, "Solving mesh router nodes placement problem in wireless mesh networks by tabu search algorithm," *Journal of Computer and System Sciences*, vol. 81, no. 8, pp. 1417–1428, 2015.

[21] M. Newman, *Networks: An Introduction*. New York, NY, USA: Oxford University Press, Inc., 2010.

[22] W. Chen and S.-H. Teng, "Interplay between social influence and network centrality: a comparative study on shapley centrality and single-node-influence centrality," in *Proceedings of the 26th international conference on world wide web*, 2017, pp. 967–976.

[23] H. Soroush, N. Banerjee, A. Balasubramanian, M. D. Corner, B. N. Levine, and B. Lynn, "DOME: a diverse outdoor mobile testbed," *Proceedings of the 1st ACM International Workshop on Hot Topics of Planet-Scale Mobility Measurements*, pp. 1–6, 2009.

[24] T. R. Henderson, M. Lacage, and G. F. Riley, "Network Simulations with the ns-3 Simulator," in *SIGCOMM'08*, 2008, p. 527. [Online]. Available: http://www.isi.edu/nsnam,

[25] D. Kempe, J. Kleinberg, and Tardos, "Maximizing the spread of influence through a social network," *Theory of Computing*, vol. 11, pp. 105–147, 2003.

[26] G. L. Nemhauser, L. A. Wolsey, and M. L. Fisher, "An analysis of approximations for maximizing submodular set functions-I," *Mathematical Programming*, vol. 14, no. 1, pp. 265–294, 1978. [Online]. Available: https://www.researchgate.net/publication/242914003

[27] J. Leskovec, A. Krause, C. Guestrin, C. Faloutsos, J. Vanbriesen, and N. Glance, "Cost-effective outbreak detection in networks," in *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2007, pp. 420–429.

[28] "GTFS Static Overview — Static Transit — Google Developers." [Online]. Available: https://developers.google.com/transit/gtfs/

[29] K. Butts, "Peartree: A library for converting transit data into a directed graph for sketch network analysis." 2018. [Online]. Available: https://github.com/kuanb/peartree

[30] OpenMobilityOrg, "OpenMobilityData - Public transit feeds from around the world." [Online]. Available: https://transitfeeds.com/

[31] U.S. Census Bureau, "U.S. Census Bureau QuickFacts: UNITED STATES," 2016. [Online]. Available: https://www.census.gov/quickfacts