Enabling Opportunistic Low-cost Smart Cities By Using Tactical Edge Node Placement

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

2021 16th Annual Conference on Wireless On-demand Network Systems and Services Conference (WONS) | 978-3-903176-35-5/21/$31.00 ©2021IEEE 10.23919/WONS51326.2021.9415579

***Abstract*—Smart city projects aim to enhance the management of city infrastructure by enabling government entities to monitor, control and maintain infrastructure efficiently through the deployment of Internet-of-things (IoT) devices. However, the financial burden associated with smart city projects is a detriment to prospective smart cities. A noteworthy factor that impacts the cost and sustainability of smart city projects is providing cellular Internet connectivity to IoT devices. In response to this problem, this paper explores the use of public transportation network nodes and mules, such as bus-stops as buses, to facilitate connectivity via device-to-device communication in order to reduce cellular connectivity costs within a smart city. The data mules convey non-urgent data from IoT devices to edge computing hardware, where data can be processed or sent to the cloud. Consequently, this paper focuses on edge node placement in smart cities that opportunistically leverage public transit networks for reducing reliance on and thus costs of cellular connectivity. We introduce an algorithm that selects a set of edge nodes that provides maximal sensor coverage and explore another that selects a set of edge nodes that provide minimal delivery delay within a budget. The algorithms are evaluated for two public transit network data-sets: Chapel Hill, North Carolina and Louisville, Kentucky. Results show that our algorithms consistently outperform edge node placement strategies that rely on traditional centrality metrics (betweenness and in-degree centrality) by over 77% reduction in coverage budget and over 20 minutes reduction in latency.**

***Index Terms*—smart cities, opportunistic networks, delay tolerant networks, internet of things, gateways, 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, traffic monitors, parking meters/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 costs incurred from each sensor's cellular subscription plans is expensive [3]. For example, cities such as San Diego, New Orleans, London, and Songdo have either proposed or invested in smart city projects that cost between $30 Million and $40 Billion. The costs 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, solely 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]–[13].

Opportunistic networks are attractive because of their ability to persist data with minimal infrastructure. Such networks leverage the already existing mobility of nodes within a city to retrieve data from IoT devices and either disseminate data to other devices in the network or act as intermediate data carriers which forward the data to specific locations that have edge computation, cloud connectivity, and storage resources [10], [14]. The research community has investigated the use of public transit vehicles — such as buses, trams, and light rail — for opportunistic networks; however most of the research has been limited to routing and forwarding schemes [15]. Messages are delivered with some delay which is directly correlated with the layout, density, and mobility of nodes in the network [16], [17]. Consequently, a question that has been marginally addressed is: **where should edge nodes be placed in a low-cost smart city that leverages public transit networks to improve connectivity?**

The opportunistic use of transportation networks is not intended to perform better than 5G or other centralized communication schemes, but instead offer a low-cost alternative for delivering time-insensitive data, enabling municipalities to become smart cities at a fraction of the cost. In this paper we: (i) introduce the Maximal Sensor Coverage (MSC) edge node placement optimization problem, and explore the Minimal Delivery Delay (MDD) problem; (ii) formulate the Maximal Sensor Coverage (MSC) as a set cover problem; (iii) develop approximation algorithms for solving the optimization problems highlighted; and finally; (iv) compare the results of our algorithms with traditional network centrality measures.

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

2021 16th Annual Conference on Wireless On-demand Network Systems and Services Conference (WONS)

ISBN 978-3-903176-35-5 © IFIP

The rest of the paper is structured thus: 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 of 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 [18], [19]. However, the vast majority of research in this area have 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 [18]. Some others have investigated routing and forwarding schemes in the context of preserving privacy [13]. Nevertheless, none of the aforementioned works 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. Some previous works on placement optimization for networks similar to ours include [20] which exploits the principle of submodularity to tackle the problem of sensor placements in water distribution networks. In [21], the authors explore several optimization techniques for access point placement in wireless mesh networks. Additionally, [22] looked at optimizing coverage and cost-efficiency in a smart parking network. Unlike prior research, this work proposes an edge node placement algorithm for low-cost smart cities that leverages opportunistic networks and evaluates it using GTFS-data derived from real transits from multiple cities.

A. Classic Centrality Analysis

The problem of edge node placement is similar to finding the most significant stops in a transit network. In the field of network analysis, there exist several popular centrality measures. These centrality measures are usually computed by a real-valued function and reflects a node's significance or importance within the respective network [23]. 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 [24] and not dynamic networks. Since our network model is more complex, containing not just nodes (stops) and edge nodes (trips), but also vehicle schedule information across each node, centrality measures may not provide the best solutions for optimizing edge node placement. Hence, for the optimization problems considered in this work, we explore other solutions.

B. Evaluation of Opportunistic Networks

A number of test-beds and simulation tools have been developed by the research community and presented in literature to facilitate research in the field of vehicular communication networks and vechicular opportunistic networks. In [25], the authors create the DOME testbed, to give researchers access to transit buses already furnished with necessary equipment, so that external researchers can upload their communication protocols to the buses remotely and conduct experiments within a real-world environment without having to invest in additional infrastructure. In [26], the authors also design an ad-hoc testbed using buses. 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, real-world deployments are typically financially expensive.

One popular alternative to real-world evaluations has been the use of mathematical models and simulations. Some of the advantages of this approach include: minimal financial costs, the ability to easily scale up experiments, and the heterogeneity of geography and hardware parameters during experiments. Examples of some widely used open-source generic simulators in the field of vehicular communication and DTN include the ONE simulator [17] and ns-3 [27]. However, it can be difficult to fully incorporate real-world data into simulation environments, or adapt the environment to match the designed network model. Hence, several works in literature use custom-built simulators for their models.

In [28], the authors designed a simulation environment for VANETS and intelligent traffic lights to notify vehicles of traffic and warning messages using Ad-hoc On-Demand Distance Vector (AODV). In [29], the authors conduct a feasibility study by setting up a simulation framework that relies purely on opportunistic interactions between taxi cabs. The lower accuracy of simulation environments, when compared to real-world environments is a factor that often undermines the integrity and validity of results. An additional contribution of our work is the development a simulator for our network model that closely emulates the movement of real world transit vehicles, hence we are able evaluate the performance of our network across many cities, while also maintaining high result accuracy.

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. Every route is primarily defined by the list of stops on which the route passes through. 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.

ISBN 978-3-903176-35-5 © IFIP

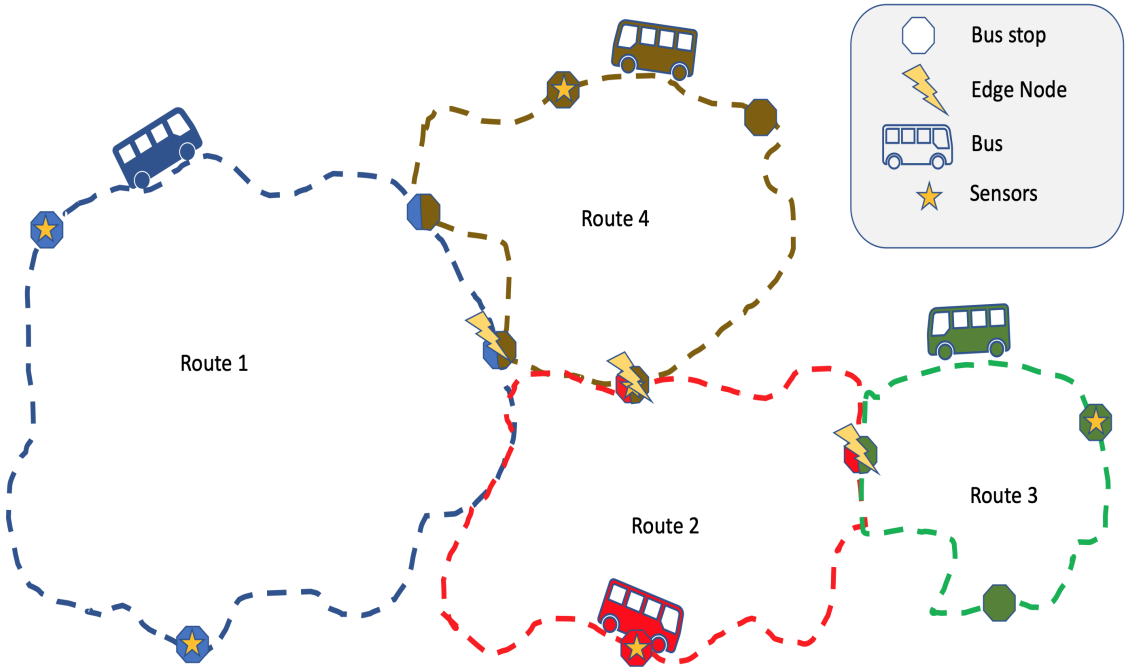


Figure 1. Network Architecture

2) Sensors: Sensors are located at select bus stops. Each sensor generates 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 whose generated data packets are small enough to justify an assumption of transmitting to a bus in a short duration. Local storage of a sensor is considered to be infinite, since data is expected to be picked up by a bus within a couple of days, thereby eliminating the need for a policy for dropping packets. Also, it is assumed that the data generated by these sensors is delay tolerant and is not relied upon to make real-time decisions.

3) Edge nodes: These are stationary and when "active (on)" are considered always-connected devices that 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. Edge nodes are equipped with edge computing and pre-processing capabilities.

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, 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 gateway within a day (buses park at transit stations which are expected to have an active gateway), so a drop-policy for stored data is not necessary.

IV. Edge Node Selection Problem

In this section, the edge node selection problem is represented as two optimization objectives and corresponding algorithmic solutions are provided. It is assumed that the cost associated with installing or activating a gateway is constant regardless of the location being considered.

A. Maximal Sensor Coverage (MSC)

The objective, Maximizing Sensor Coverage (MSC), is to *find the minimum/smallest set of locations to install or activate gateways such that there is at least one direct path in the network from all possible sensor locations to an edge node*. This would ensure that all sensors, regardless of where they are placed on the transit network will have a chance of having its data delivered.

Recall that the bus transit network consists of several routes through which vehicles move, each route comprises of a set of stops and two or more routes may share a stop. Therefore, we can define our MSC problem as such: *Given a set of routes R, and a list of stops, S, each with a subset of routes, $R_l \subseteq R$, that use that stop (each element of R is associated with at least one stop), find the minimal set of stops required to cover all routes*. This problem can be reduced to a minimal set cover problem. In the set cover problem, we are given a universal set U, such that $|U| = n$, and a family of subsets $L_1, ..., L_k \subseteq U$. A set cover

ISBN 978-3-903176-35-5 © IFIP

is a collection C of the subsets $L_1, ..., L_k$ whose union is the universal set U. Formally, C is a set cover if $\bigcup_{L_i \in C} L_i = U$. To find the minimal set cover, the objective is to minimize $|C|$.

The reduction is fairly intuitive. In our case the universal set is the set of all routes $R = U$, and the family of subsets are the set of routes each stop services, $S = L$. Given a decision variable, $x_l \in \{0, 1\}$, which indicates whether a stop in L is picked, the ILP formulation is thus:

$$\text{minimize} \sum_{l \in L} x_l \text{ st} \tag{1}$$

$$\sum_{l:r \in L} x_l \geq 1 \quad \forall\ r \in R \tag{2}$$

$$x_l = \{0, 1\} \quad \forall\ l \in L \tag{3}$$

The problem of finding the optimal set cover solution is *NP-Hard*. Nevertheless, the greedy approach is able to find a solution close to the optimal set cover. It is bounded above by a $O(log_e n)$ approximation to optimal solution of the set cover problem, where n is the number of routes in the network. The greedy MSC algorithm is described in Algorithm 1. At each iteration, we find the gateway candidate that provides the largest increase in the number of routes covered, and add it to the gateway set. This process is repeated until all routes have been covered by the gateway set.

Algorithm 1 Maximal Sensor Coverage (MSC)

Input — routes **R**, route subsets **S**
Output — selected gateway set **G**

procedure GREEDY-MSC(R, S)
  $X \leftarrow R$
  $G \leftarrow \emptyset$
  while $X \neq 0$ **do**
    Select an $S_i \subseteq R$ that maximizes $|S_i \cap X|$
    $X \leftarrow X \setminus S_i$
    $G \leftarrow G \cup S_i$
  return G

B. Minimal Delivery Delay (MDD)

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. The penalty value is selected in such a way that it indicates that the message was not delivered within the time window of the simulation. In our evaluation, we picked a fixed value outside the range of simulation window. Further, the edge node selection process is conducted without prior knowledge of the stops in which the sensors will placed in the city.

The problem is formulated 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 [30]. 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.

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. Given an undelivered messages time penalty - T, the minimum time it takes to get from stop, v, to stop, u - $\tau(u, v)$, and 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 - $\mathbf{t}(s, \mathscr{s})$, the influence function for a set of edge nodes, G is defined as:

$$\sigma(G) = T - \frac{1}{|S| - G} \sum_{\mathscr{s} \in S, \mathscr{s} \notin G} min(\mathcal{T}(G, \mathscr{s})) \tag{4}$$

Where,

$$\mathcal{T}(G, \mathscr{s}) = \{(\tau(g, \mathscr{s}) + \mathbf{t}(g, \mathscr{s})) \mid \forall\ g \in G\} \tag{5}$$

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. The influence function makes use of "data delivery delay" (Algorithm 4), to calculate the network latency. This function is submodular and its proof can found in a previous work [31]. Even though finding the set of edge nodes that maximize influence is *NP-Hard*, since the influence function is submodular, the solution can be approximated using the *Greedy* and *Cost-Effective Lazy Forward* (CELF) algorithms [32].

1) Greedy Algorithm: Since the problem is reduced to the maximization of a monotone submodular function, the greedy algorithm provides a $(1 - 1/e)$ - approximation [32]. Hence, the greedy algorithm is theoretically guaranteed to choose a gateway set whose network latency will be at least 63% of the network latency of the optimal gateway set. Our greedy algorithm is described in Algorithm 2. It starts with an empty gateway set $S = \emptyset$. In each iteration, the greedy heuristic chooses a new gateway u from the non-gateway nodes $V \setminus S$ with largest (marginal) influence gain $\sigma(S \cup u) - \sigma(S)$ and adds u to S. The algorithm terminates after selecting k gateways.

2) CELF-MDD Algorithm: Although the greedy algorithm is much quicker than a brute-force approach, the greedy algorithm is still very slow when considering the size of

ISBN 978-3-903176-35-5 © IFIP

Algorithm 2 Greedy Minimal Delivery Delay

Input — network graph **N**, influence function σ, budget **k**
Output — selected gateway set **G**

procedure GREEDY-MDD(N, σ, k)
  $G \leftarrow \emptyset$
  while $|G| < k$ **do**
    $u \leftarrow arg\ max_{\ v \in V \setminus G}\ \sigma(G \cup v) - \sigma(G)$
    $G \leftarrow G \cup \{u\}$
  return G

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 [33]. 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 submodular 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 3 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[1] that models a vehicular communication network consisting of sensors, buses and edge nodes within any real-world city, by directly using real transit network information provided in the General Transit Feed Specification (GTFS) format [34]. GTFS handles information on transit routes, stops, and timetables [34]. By building a simulation tool that incorporates GTFS information, we are able to evaluate the performance of our low-cost smart city model for hundreds of 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. The conversion is done using an open-source library called *peartree* [35]. 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 weight on each edge 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 time value representing the frequency at which it generates data. This frequency value is assigned to each sensor based on a uniform random distribution.

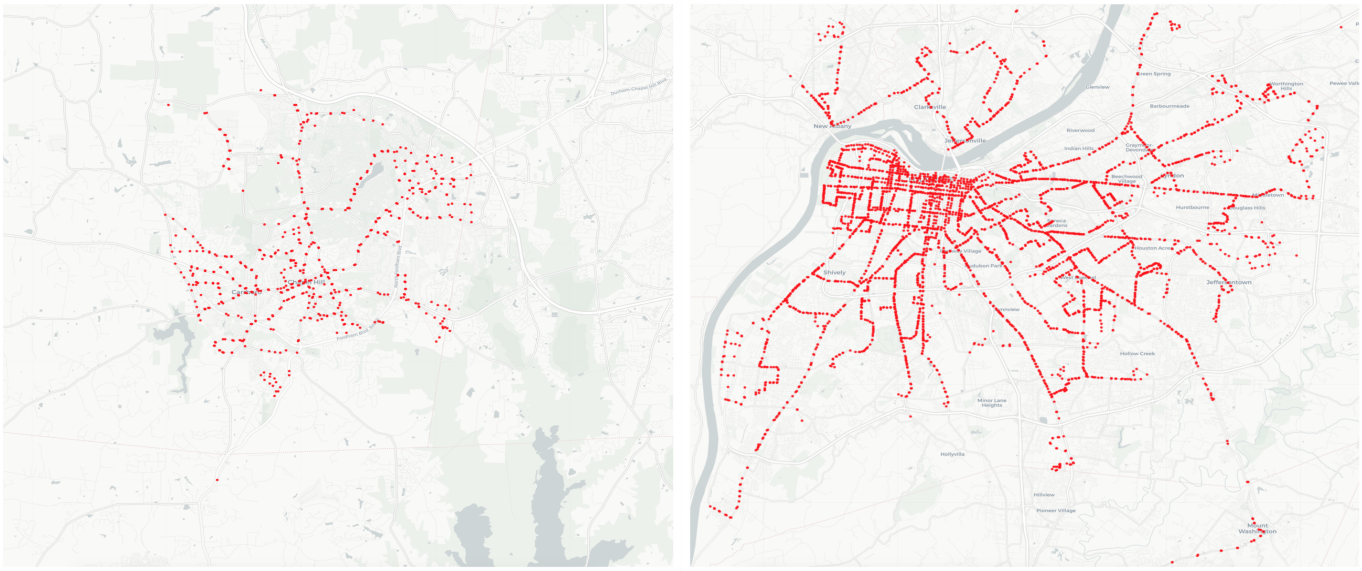
3) Data Delivery Delay: For each data packet generated at a sensor, the shortest duration it takes for the data packet to get to an edge node is computed. First, a subgraph (consisting of only stops in a single route) is extracted from the main graph for each route. Next, we iterate through each route the sensor is on and compute the shortest path length from the sensor to any edge node on that route. Since the edge node 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 edge node after departing from the sensor. The total estimated delay is calculated by adding the shortest path length to the waiting time (the lag between the data generation time and the time at which the next vehicle associated with that route arrives at the stop where the sensor is located).

After iterating through all routes for the sensor, the path with the shortest total estimated delay is designated as the path through which the data will travel. The path length of the designated path is the estimated end-to-end delay for the data packet generated. Algorithm 4 presents the pseudocode for computing the delay for each data packet generated.

4) Storing results for analysis: For each simulation, important information such as generation time, delivery time, delivery path, vehicle wait time, and vehicle travel time for each data packet generated during simulation is recorded and stored in JavaScript Object Notation (JSON) file. This helps with carrying out post-simulation analysis after the simulation has ended without having to re-run simulations.

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

ISBN 978-3-903176-35-5 © IFIP



(a) (b)

Figure 2. (a) Map of CHT bus-stop locations in Chapel Hill, (b) Map of TARC bus-stop locations in Louisville.

Algorithm 4 Pseudocode for Computing data delivery delay

Input — routes, R; sensor, E; time, t
Output — delay, D

procedure COMPUTEDELAY(R, E, t)
  $D \leftarrow \infty$
  for $r \in R$ **do**
    $G_r \leftarrow$ GETROUTESUBGRAPH(r)
    for $g \in r.gateways$ **do**
      $length \leftarrow$ DIJKSTRASHORTESTPATH(G_r, E, g)
      if $length$ **then**
        $wait \leftarrow$ TIMETILLNEXTARRIVAL(E, R, t)
        if $wait$ **then**
          $D \leftarrow min(D, length + wait)$
  return D

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 [36]. Chapel Hill is a relatively small town, measuring 55 km sq (21.3 sq miles) and an estimated population of about 60,988. Louisville, on the other hand, is a much larger city with a population of 620,118 and land area of 171.70 km sq (66.29 sq miles) [37]. The difference in city size also translates to the differences in public transit networks present in both cities as highlighted in Table I and Figure 2.

B. Maximizing Coverage

Figures 3a and 3d along with Table II give insight into the performance of the MSC Algorithm described in IV-A when compared two traditional graph centrality measures - betweenness centrality (BC), and in-degree centrality (IC). For the centrality measures, we greedily picked each gateway in order of decreasing centrality, until all routes were covered. Figures 3a and 3d show the rate of increase in route coverage compared to the number of gateways selected. We see that MSC outperforms BC and IC significantly for both CHT and TARC. For CHT, MSC requires just 4 gateways to cover all routes in the network, compared to 18 and 25 gateways in IC and BC, respectively. For a larger network like TARC, MSC requires 13 gateways to cover all routes in the network, compared to 257 and 459 gateways in IC and BC, respectively.

Table I
BUS NETWORK CHARACTERISTICS CHT AND TARC

Statatistics	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

Table II also highlights the delivery ratio for each algorithm. Since the three algorithms (MSC, BC, and IC) cover all routes, the delivery ratio is essentially the same. Delivery ratio is less than 100% because delivery also depends on a bus arriving at a sensor within the simulation time window. This means that for some data packets generated, especially in the later stages of the simulation, there were no buses moving on that route causing the data packets to remain undelivered.

C. Minimizing Latency

We evaluate the effectiveness of the CELF-MDD algorithm (described in Section IV-B2) at minimizing the overall message

ISBN 978-3-903176-35-5 © IFIP

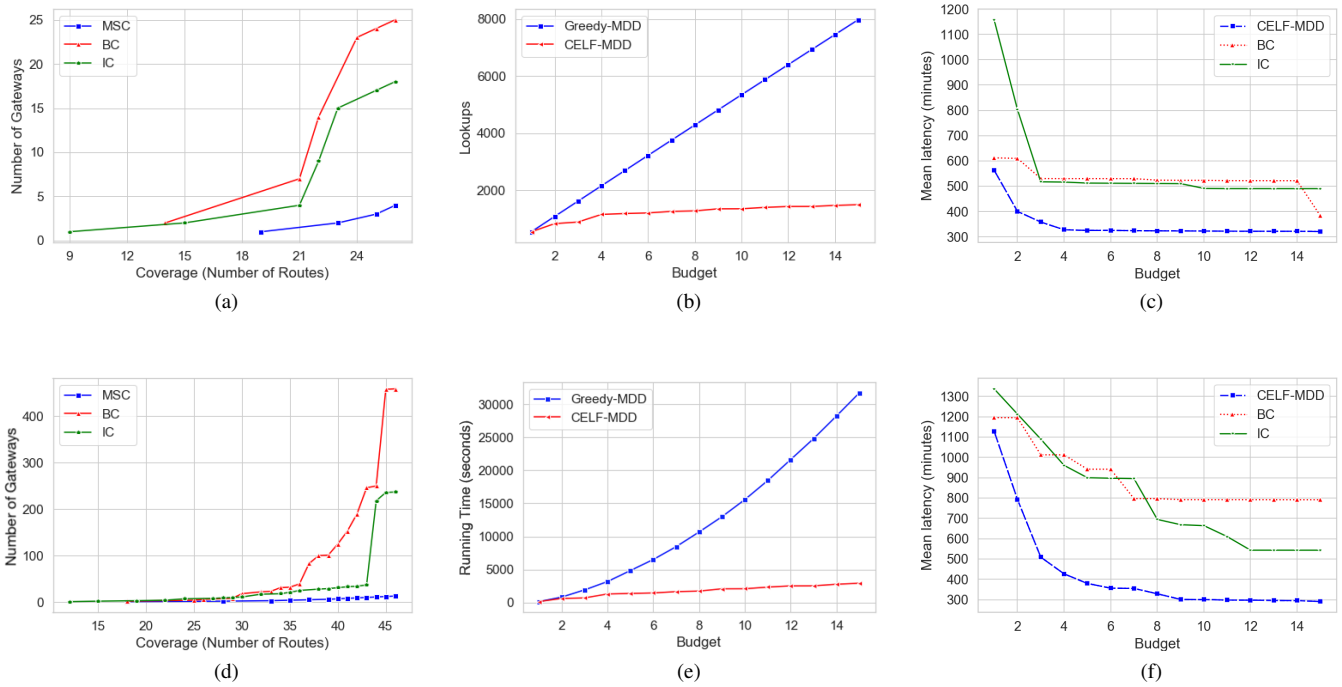


Figure 3. (a) MSC performance for CHT, (b) Graph showing the number of lookups for Greedy-MDD and CELF-MDD, (c) Average network latency for CHT (d) MSC performance for TARC. (e) Graph showing the number of running time for Greedy-MDD and CELF-MDD (f) Average network latency for TARC.

Table II
RESULTS OF MSC, BC AND IC

	CHT		TARC	
Algorithms	Cost	Delivery ratio (%)	Cost	Delivery ratio (%)
MSC	4	86.0512	13	87.0903
BC	25	84.4760	459	87.8831
IC	18	85.9515	237	88.4032

delay in the network model. We first compare the difference in run-time efficiency between Greedy-MDD and CELF-MDD. Figures 3b and 3e show the average number of lookups and running-times for a gateway budget of up to 15 when working with the CHT network. The number of lookups for each budget refers to the number of times the influence function needs to be computed before a gateway was selected at that stage. Greedy-MDD grows linearly because for each iteration, the influence gain for all gateways that have not been selected has to be computed. However for CELF-MDD, the number of lookup grows much slower due to it leveraging results from past computation as discussed in Section IV-B2.

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 III. Consistent with the influence function, the upper bound value specified in Table III is assigned as the penalty value (delay) for undelivered data. Figure 3c and 3f 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 ≈ 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 III
SIMULATION AND SCENERIO PARAMETERS

Random generator seeds	0:1:100
Simulation start time	1:00:00
Simulation end time	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	$U(30, 40)$

VII. CONCLUSION

This work addresses the problem of efficient edge node placement in low-cost smart cities that opportunistically utilize

ISBN 978-3-903176-35-5 © IFIP

public transit networks as data mules. We introduced several optimization algorithms and compared them to traditional network centrality measures. Experiments were carried out using public transport networks in two cities in the United States; Chapel-Hill and Louisville. The results show that our algorithms outperform traditional centrality measures by reducing network latency and ensuring coverage at minimal cost, indicating that our algorithms are effective in determining the best locations to place edge nodes, minimize delivery delay and minimize cost in low-cost smart cities that opportunistically utilize public transit networks as data mules.

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.

[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] 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.

[13] 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.

[14] T. M. Ho, T. D. Tran, T. T. Nguyen, S. Kazmi, L. B. Le, C. S. Hong, and L. Hanzo, "Next-generation wireless solutions for the smart factory, smart vehicles, the smart grid and smart cities," *arXiv preprint arXiv:1907.10102*, 2019.

[15] 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.

[16] 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.

[17] 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.

[18] 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.

[19] 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.

[20] 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.

[21] 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.

[22] A. Bagula, L. Castelli, and M. Zennaro, "On the design of smart parking networks in the smart cities: An optimal sensor placement model," *Sensors*, vol. 15, no. 7, pp. 15 443–15 467, 2015.

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

[24] 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.

[25] 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.

[26] J. Paradells, C. Gomez, I. Demirkol, J. Oller, and M. Catalan, "Infrastructureless smart cities. use cases and performance," in *Smart Communications in Network Technologies (SaCoNeT), 2014 International Conference on*. IEEE, 2014, pp. 1–6.

[27] 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,

[28] C. T. Barba, M. A. Mateos, P. R. Soto, A. M. Mezher, and M. A. Igartua, "Smart city for vanets using warning messages, traffic statistics and intelligent traffic lights," in *Intelligent Vehicles Symposium (IV), 2012 IEEE*. IEEE, 2012, pp. 902–907.

[29] M. Bonola, L. Bracciale, P. Loreti, R. Amici, A. Rabuffi, and G. Bianchi, "Opportunistic communication in smart city: Experimental insight with small-scale taxi fleets as data carriers," *Ad Hoc Networks*, vol. 43, pp. 43–55, 2016.

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

[31] O. Madamori, E. Max-Onakpoya, G. D. Erhardt, and C. E. Baker, "A latency-defined edge node placement scheme for opportunistic smart cities," in *2021 IEEE International Conference on Pervasive Computing and Communications Workshops (PerCom Workshops)*. IEEE, 2021.

[32] 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

[33] 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.

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

[35] 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

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

[37] U.S. Census Bureau, "U.S. Census Bureau QuickFacts: UNITED STATES," 2016. [Online]. Available: https://www.census.gov/quickfacts/fact/table/US/PST045218#qf-headnote-a%0Ahttps://www.census.gov/quickfacts/fact/table/US/PST045216

ISBN 978-3-903176-35-5 © IFIP