Supplemental Materials:- Faster Fog Computing based Over-the-air Vehicular Updates: A Transfer Learning Approach

VIII. Additional Experimental Results and Analysis

A. Traffic clustering and fog nodes distribution

In order to comprehend the pattern of traffic and to efficiently distribute the fog nodes, we analyzed the NYC taxi drive dataset [29] for the month of March (31 days). The following features were extracted from this data set: id, dropoff_datetime, pickup_datetime, dropoff_latitude, dropoff_longitude, pickup_latitude, pickup_longitude, speed, distance, and trip time.

Initially, based on datetime and traffic location, we determine the number of clusters. For each day's data of the dataset, we execute the k-means algorithm to determine the number of clusters and cluster sizes. Consider the elbow curve of Figure 4(a). We use this curve to find the optimal number of clusters in the NYC data set. The optimal number of clusters is hence defined as five. We find the maximum size of each cluster post application of the k-means algorithm for each day. Figure 4(b) shows the distribution of traffic in each cluster, where each cluster has a different number of vehicles. According to our system model, we assume that each cluster size has the maximum amount of traffic. In addition, we find the centroids of each cluster following Algorithm 1, and calculate the distance of each vehicle from its centroid position. Finding the maximum distance for a vehicle from a centroid in each cluster, we calculate the radius of each cluster. Figure 12 shows the range of each cluster and sample points of vehicles in each cluster with the centroids.

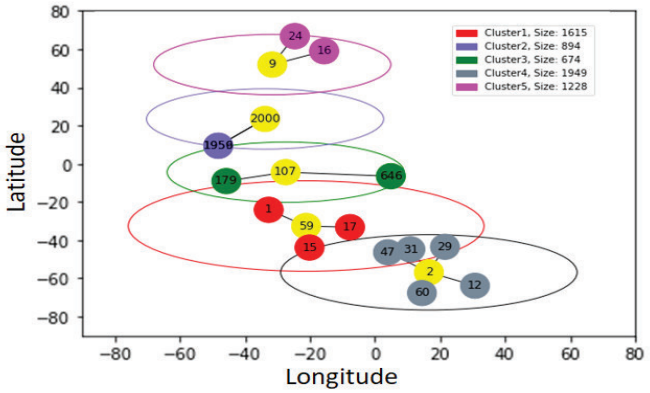


Figure 12: Range and size of each cluster

Using Algorithm 2, we calculate the initial cluster size of diverse locations at specific times. Please see Figure 13(a) for the results. This experiment was done with the number of fog nodes equal to 17. The capacities of various fog nodes are shown in Figure 13(a).

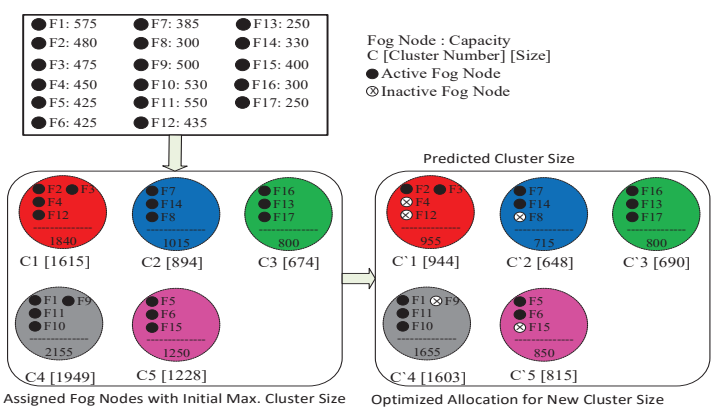
The fog nodes are sorted in a descending order based on their capacities and then distributed among different clusters. This process is carried out till all the cluster requests are satisfied by the fog nodes. We do note that the traffic size in a cluster does not always take the maximum value. Hence, using neural network machine learning, we predict the cluster size. This has the effect of optimizing the fog resources at hand. The assumption here is that the predicted cluster size is not greater than the initial cluster size, leading to a smaller number of fog resources. The number of available fog nodes in a cluster is enough to handle all incoming traffic to that cluster. If the fog nodes do not have enough capacity to accommodate the vehicles in that cluster or if any fog node(s) becomes faulty, the traffic is offloaded to the cloud data center. The proposed framework can handle variable traffic. If the incoming load in any cluster is less than its capacity, then the idle fog nodes will be put on inactive mode. Later on, when the traffic increases, the inactive fog nodes will become active according to the new requirement.

It is our advice that the fog nodes that are not needed to service the requests of the newly sized cluster be kept in an inactivate state. These inactive fog nodes become part of the net reserve of fog resources, as shown in Figure 13(a). We note that this optimization of fog nodes maximizes resource utilization and minimizes power consumption. For the next experiment, we define three different indicators in Figure 13(b): fog resource utilized, fog resource unused, and fog net reserve. The resource utilized symbolizes the total amount of used resource of all fog nodes in a cluster at a particular time. The unused resource determines the amount of resource that is allocated, but not in use. It is inversely proportional to the resource utilization of fog nodes. On the other hand, the net reserve resource is the actual resource (not allocated) that can be used for other purposes. Therefore, if the resource utilization and unused resource decrease, then the net reserve resources will also increase, and it can be distributed following Algorithm 2. The symbols C1 - C5 correspond to the clusters before application of our proposed algorithms and the symbols C'1 - C'5 refer to the clusters after application of our proposed algorithms. From Figure 13(b), we observe that for each cluster, we see an average of 93.69% fog node utilization upon allocation, 6.31% of unused fog resources, and the net reserve (after optimization) of fog resources. We observe that there is a 26.57% increment in the overall net reserve fog resources on average, which reflects good system performance as a result of our proposed approach.

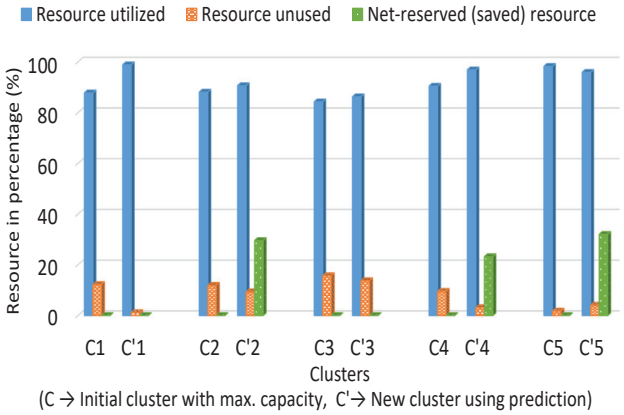
B. OTA update time calculation using Mininet-WiFi

This experimental subsection discusses the implementation of the proposed OTA update scheme on the Mininet-WiFi [26], which supports vehicle mobility and wireless communication under the simulated fog architecture. Mininet provides a simple and inexpensive network testbed for developing OpenFlow applications compared to other emulators. The architecture provides the specification of flows following the software-defined network (SDN) and allows us to add a network of virtual hosts, switches, controllers, and links. The flow specification contains the link delay, bandwidth, vehicle speed, network topology, and switch information. The simulation calculates the OTA update time for different software sizes exploring the required handover delay.

This experiment was performed employing five cars/vehicles and three fog nodes in a 250x250 meter area. The Mini net-WiFi network uses the 802.11g standard for transmitting data over a wireless network. The bandwidth of the WiFi-network is 2.4 GHz and the maximum data transfer rate is 50 Mbps. We attempt to find out the OTA update time for software of various sizes within a cluster, which is cluster



(a) Resource allocation example



(b) Allocated fog nodes utilization in different clusters

Figure 13: Fog nodes distribution using Algorithm 2 and resource utilization calculation in different clusters

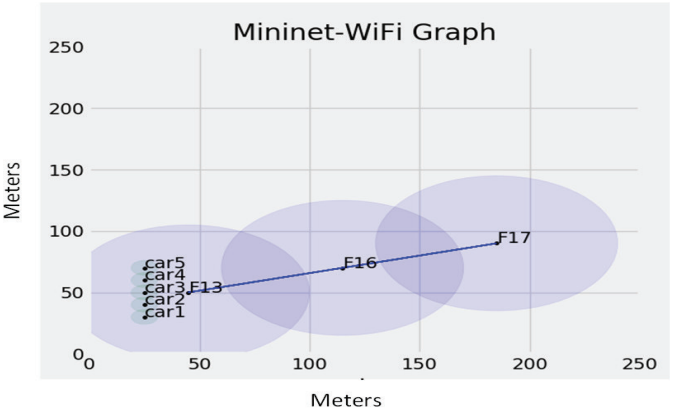


Figure 14: Mininet-WiFi network architecture for cluster 3

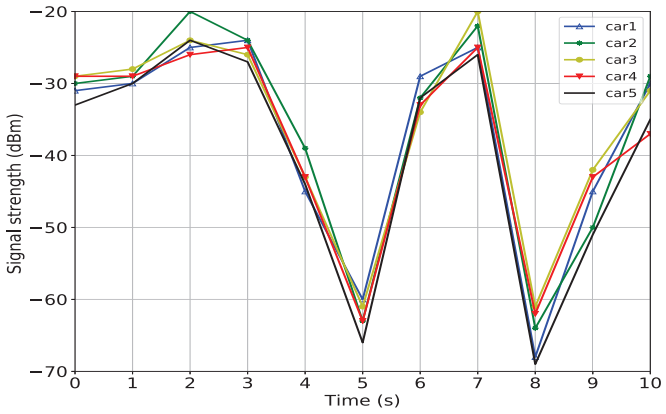


Figure 15: Signal strength of each car over the time

Artifacts/Parameters	Values
Number of cars/vehicles	5
Number of fog nodes	3
Vehicle speed	14m/s or 36km/hr
Bandwidth of links	50Mbps
Predicted propagation delay (ms)	76.030876, 70.1916, 64.80096, 72.4002 and 97.80364
Propagation model	logDistance
Association control	ssf (Strongest-Signal-First)
Range of each fog node	30m
Interface	Wlan0, Wlan1

Table II: Mininet-WiFi network parameters

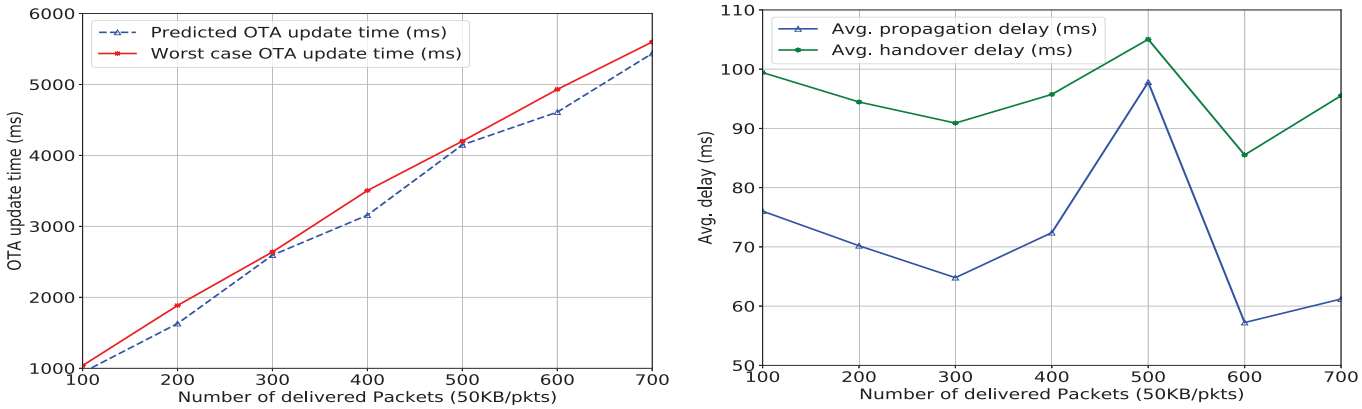
3 shown in Figure 13(a). The details of the experimental architecture are shown in Figure 14.

Our proposed solution evaluates the effectiveness of the predicted communication delay by calculating the handover delay and OTA update time in the presence of multiple traffic requests in the network. Table II lists all the network parameters used.

For this experiment, we take into account five cars moving at a speed of 14m/s (meters/second) from their pickup locations to their drop-off locations. Each car will be connected to a fog node when it goes under its coverage area. Once the vehicles start moving from one position to another, the signal strength of the fog node connections also varies over time. The ranges of fog nodes overlap where the vehicles select the strongest-signal connection. We find the signal strength after every second for each car, when it changes its position towards the drop-off location. Figure 15 shows the variations of signal strength for 10 seconds. During times equal to 5 and 8 seconds, the signal strength goes to its lowest, and each car performs a handover to connect with the strongest fog node signal. Thus, we see a gradual increase in fog node signal strength. This indicates a successful handover in the overlapping region.

Simultaneously, the mobility starting and ending points are added for each car and the speeds of the cars are set at 10m/s. When the cars start moving while following their route, they get connected in the middle of the route with different fog nodes, such as fog node 13, 16, and 17. At the beginning, all the cars remain under the coverage of fog node 13 through interface Wlan0, while the other interfaces remain off. The bandwidth of the links is set to 50 Mbps.

During the time the car is mobile, we calculate the total OTA update time by varying the number of transmitted data packets. We note that owing to variations in the distance, the data transmission rates may fluctuate. We assume that an update consisting of a number of data packets starting from 100 to 500, is pushed from fog nodes to the cars, where each data packet is equivalent to 50Kb. We set the predicted propagation delays in the transmission flow, which are 76.030876, 70.1916, 64.80096, 72.4002, and 97.80364 ms, respectively. Based on the OTA update Algorithm 4, we calculate the data transmission time measuring the time taken to complete the update transfer. As the last step, the propagation time and transfer time are subtracted in order to



(a) OTA update time for different data size

(b) Handover and propagation delay

Figure 16: OTA update time and handover delay

obtain the handover delay. The handover delay (Δ_i) includes the following delays:

$$\Delta_i = D_{rang} + D_{req} + D_{res} + D_{ex} \tag{10}$$

Here,

- D_{rang} = Time required for initial ranging process. This finds out the fog nodes with a maximum coverage area.
- D_{req} = Time required for requesting to connect to a new node.
- D_{res} = Time required to register with target fog node.
- D_{ex} = Time required for message exchange.

In order to make a comparison with the worst-case scenario of propagation delay, we calculate the OTA update time maintaining the same procedure as before. In this case, the worst-case time is selected from the maximum of all the predicted propagation delays.

The results of this experiment are shown in Figure 16. Looking at Figure 16(a), we see that the OTA update time is 940.31 ms for 100 data packets, and 5437.18 ms for 700 data packets. With the worst-case propagation delay, the OTA update time increases to 1037.34 ms and 5599.45 ms for 100 and 700 packets, respectively. We observe that the average overall OTA update time is reduced by 5.34%.

Figure 16(b) shows the comparison of the average predicted propagation delay and the average handover delay for all five cars. When the number of delivered packets varies, the average handover delay changes with respect to propagation delay. We observe from the figure that the handover delay fluctuates, leading to a high value when the propagation delay is high, and low value when the propagation delay is low. The intuition is that the correct prediction of the propagation delay can help to determine the handover delay and overall OTA update time. Therefore, the proposed scheme using transfer learning offers a better solution in calculating the OTA update time for making any decision at an earlier time. It helps to decide whether the OTA update process should proceed or not. The transfer learning approach benefits in training the model to find the expected communication delay for which the OTA update time calculation becomes faster.

We conduct another experiment to see the effect of varying the number of vehicles in the proposed architecture. The number of vehicles was varied from 10 to 100, and the corresponding OTA update time, the sum of propagation delay and handover delay were recorded. The result of this experiment is shown in Table III. The experiment sends 100 data packets for each run where each data packet comprises a 50 Kilobytes of zero inside of it. We observe that the average OTA update time increases when the number of vehicles increases. When the number of fog nodes increases from three to four, the propagation delay decreases, but the handover delay increases lightly. Therefore, the OTA update either remains close to the previous result or starts rising when the number of cars increases. The reason behind the increased OTA update time is an increase in the overall resource utilization and communication delay. With a 95% confidence level in OTA update times listed in Table III, the upper limit of the confidence interval (CI) is 1130.75ms, and the lower limit is 1099.00ms. In the case of delay calculation, the standard mean of delay is 184.64ms, and the error is almost 3.5% where the estimated upper CI is 192.09ms and the lower CI is 177.19ms for a 95% confidence level. The confidence interval at 95% produces a small bound for OTA update time and delay calculations. From these results, we conclude that our proposed approach is scalable and can offer the desired performance, even for a large number of vehicles.

# Vehicles	# Fog Nodes	T_i (ms)	Delay (ms) ($T_i^p + \Delta_i$)
10	3	1060.82	174.5
20	3	1081.43	184.2
30	3	1064.39	163.1
40	3	1095.34	175.3
50	3	1130.10	205.1
60	3	1125.21	169.4
70	3	1136.26	167.6
80	3	1159.77	186.4
100	3	1168.43	175.4
10	4	1067.34	176.79
20	4	1074.67	174.69
30	4	1092.3	190.1
40	4	1106.5	198.57
50	4	1100.28	177.5
60	4	1117.23	186.34
70	4	1128.52	203
80	4	1136.41	204.88
90	4	1149.38	214.13
100	4	1150.54	206.7

Table III: Effect on OTA update time for increasing number of vehicles

C. Effect upon varying packet sizes and vehicles

To understand the impact of varying the number of vehicles on the software update process, we measure the average throughput after latency calculation. The throughput (DR_i) is the total amount of transferred data from fog nodes to any vehicle with respect to time. Equations 4 and 5 determine the throughput for sending different number of packets (Pkt^j_{size}) against varying traffic size.

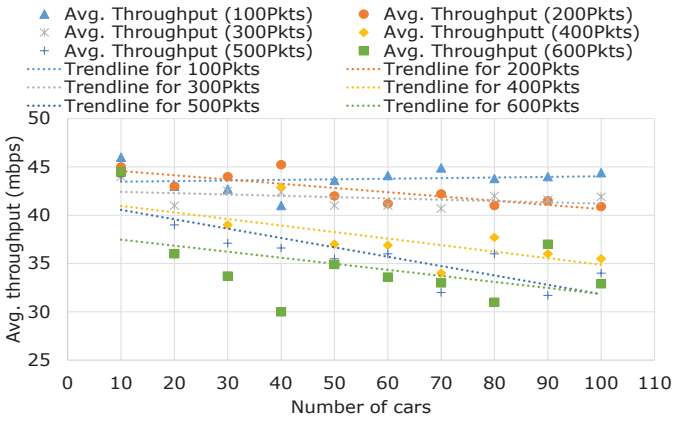


Figure 17: Effect on throughput for increasing number of vehicles and software sizes

Figure 17 show the average throughput against different software sizes for an increasing number of cars. It is visible that the throughput changes while varying the number of vehicles. Moreover, we observe that the throughput is maximized in most cases when the traffic size (number of cars) remains small. For example, when the number of cars is 10, the average throughput for different software sizes is about 45 Mbps. The standard deviation among these throughputs is also small. However, when the number of cars increases, we see that the average throughputs for small software sizes are comparatively higher than the larger software sizes. For updating the software sizes of 500 and 600 packets, the average throughput is always below 40 Mbps due to larger resource demand with propagation delay. On the other hand, the trend line for 100 packets shows how the throughput changes with a small deviation for low resource utilization and low propagation delay. Similarly, the trend line for 800 packets has a larger deviation than the others.

Apart from this, we compare the predicted communication delay for an individual car with the actual communication delay (simulated), where the predicted communication delay is either close to the actual value or a little higher than the actual. Figure 18 shows the comparison between the predicted communication delay using transfer learning and the actual communication delay (simulated) with neural network.

In the next experiment, we try to find out how the OTA update time changes when the other key parameters like: the number of vehicles, latency, and the number of fog nodes change. We use a software tool called TuringBot for Symbolic Regression. We perform the simulation in TuringBot, varying the number of fog nodes between three to five and recording the OTA update time for 100 packets, while increasing the number of cars from 10 to 100. From Figure 19(a), (b) and (c), we find that the OTA update time increases in all three cases, for increasing the number of cars. However, the OTA

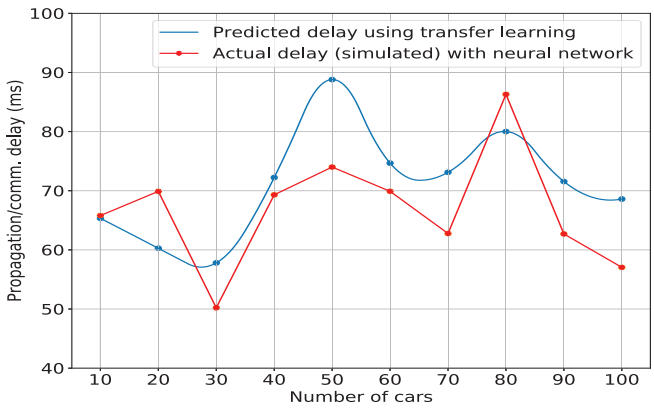
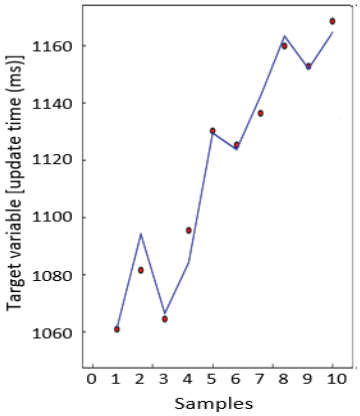


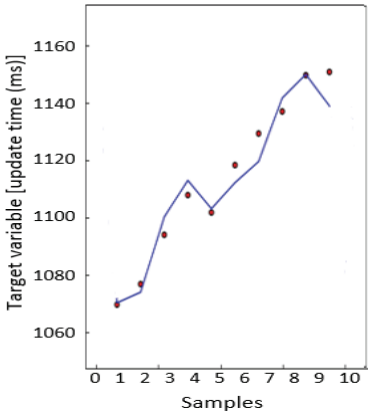
Figure 18: Actual Communication delay comparison with predicted delay using Transfer learning

update time for three fog nodes is slightly higher than the other two cases. On the other hand, the OTA update time for setting fog nodes to four or five is a little lower. Thus, the possible equation for OTA update time calculation can be Equation 11, where A (1182) and B (867) are integer constant variables. This equation tries to calculate the possible nearest value of the actual experimented data. The mean error for predicting the OTA update time is 3.31965, where mean error relative is 0.300923%.

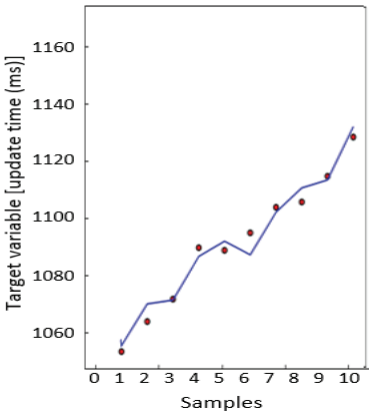
$$T_i = \frac{T_i^{lt}}{(cos(A * T_i^{lt}) + n} + T_i^{lt} + B \tag{11}$$



(a) OTA update time for using 3 fog nodes



(b) OTA update time for using 4 fog nodes



(c) OTA update time for using 5 fog nodes

Figure 19: OTA update time comparison against different samples that includes number of cars, latency