Latency Estimation for Fog-based Internet of Things

Jianhua Li[1], Tiehua Zhang[1], Jiong Jin[1], Yingying Yang[2], Dong Yuan[3], Longxiang Gao[4]
[1]Swinburne University of Technology, [2]University of Technology Sydney, [3]The University of Sydney, [4]Deakin University
{jianhuali, tiehuazhang, jiongjin}@swin.edu.au, yingying.yang@student.uts.edu.au, dong.yuan@sydney.edu.au, longxiang.gao@deakin.edu.au

***Abstract*—Low latency is critical for delay-sensitive applications such as video surveillance, live streaming, and online data analytics. Fog computing enables the emergence of the latency-sensitive internet of things (IoT) network to support real-time applications. While the distance between sensing and processing is minimized in the fog network, the cross-fog latency is yet to be determined. In this paper, we study the components of network delays and develop a latency estimation framework for fog-based IoT. The proposed framework, in particular, precisely predicts the end-to-end inter-node delay along the cloud-fog-things continuum. We investigate the benefits and use cases based on latency estimated by the proposed framework. A case study is further conducted to illustrate the validation and advantages, followed by future research directions.**

***Keywords*—Fog computing, IoT, Vivaldi algorithm, GNP, latency**

I. INTRODUCTION

Nowadays, smart environments (e.g., smart city, smart community and smart home) are built heavily relying on ubiquitous things and remote clouds promoting the emergence of the internet of things (IoT). With the mission to enhance the communication and collaboration capabilities, IoT ecosystem integrates things, data, processes, and people to form an unprecedented network along the cloud-to-things continuum. While the integration is barely in its infancy period, it brings tremendous challenges to the current internet, just to name a few, heterogeneity, mobility and latency. Such constraints drive the rising of fog computing (hereinafter fog) [1], which brings the powerful computing intelligence to the proximity of things, as shown in Figure 1. Conceptually, fog is inclusive of the global cloud, regional core, last mile access networks, clients and things. Fog, spanning from a variety of things to all level of users, is an end-to-end horizontal architecture in which processing, storage, control, management, connecting capabilities, and applications are distributed in the most efficient, logical place between data consumer and data source.

Some heterogeneous IoT applications in the smart environment are delay-sensitive and real-time. In other words, the collected data from sensing tiers must be immediately processed to trigger the respective actuator without any delay or within a tolerable time constraint, if any. For example, in the connected vehicle system, safety and traffic support data are required to be instantly processed to prevent accidents. Under this circumstance, a second even milli-second level delay is detrimental to the interest of life and asset. Initially, we briefly discuss the delay components that are responsible for the overall end-to-end latency along a particular route [2]. Later, we determine how to minimize such components theoretically and practically.

$$D_{end-to-end} = N \times \left(d_{proc} + d_{queue} + d_{seri} + d_{prop}\right) \quad (1)$$

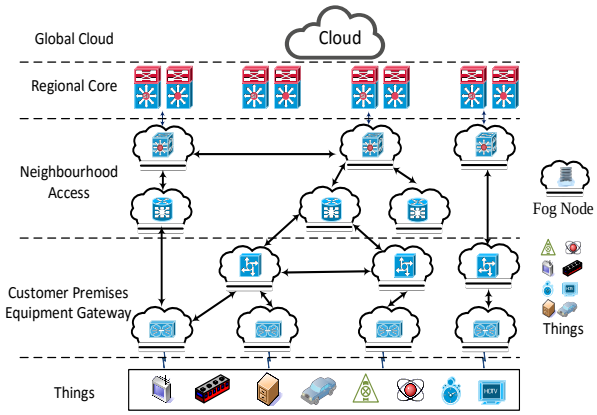
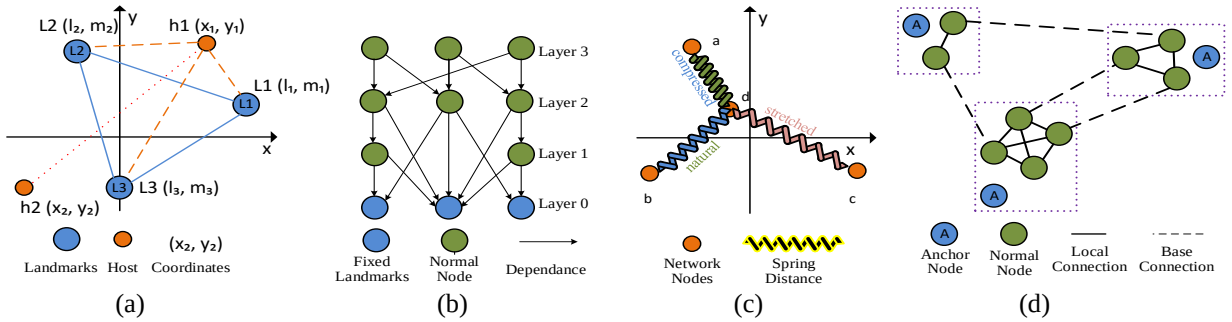


Figure 1: The Cloud-Fog-Things Continuum

In (1), d_{proc} is nodal processing delay, d_{queue} is queuing delay, d_{seri} is serialization delay, d_{prop} is propagation delay, and N is the number of network segments a packet must go through along the IoT ecosystem. Thus, the overall end-to-end delay will be approximately N times of the summation of the above four delays. With the advancement of both hardware and software, the value of d_{proc} and d_{seri} per node are on the order of microseconds [3], while the value of d_{prop} is about five micro seconds per kilometre. By the use of QoS technique, d_{queue} can be optimized for some prioritized data. Overall, fog effectively cuts latency between sensor reading and resulting actuator response [4] by minimizing the propagation delay and the number of network segments (N) within one fog network. However, when a packet travels out of fog networks where fog players have no visibility on number N, the delay and jitter may dramatically surge. And in such case, fog's promise to latency-sensitive applications can be broken by the accumulative high delays.

Accurately predicted latency brings positive inputs to many applications such as path selection. Nowadays, latency estimation is primarily conducted through the assistance of network coordinate system (NCS) that establishes a virtual positioning system for every node. Grounded on the known coordinates in geometry, a node can envisage its latency to peer nodes. Many NCS algorithms can achieve more than 90 percent accuracy of latency estimation on the internet. However, it is not appropriate in dynamic fog environment where the existence of a group of nodes is temporary. Our contribution, in this paper, is to propose an NCS algorithm for all network nodes in the fog that estimate end-to-end packet delay with higher accuracy.



(a) (b) (c) (d)

Figure 2: Network Coordinate System a) GNP b) NPS c) Vivaldi d) Pharos

The remainder of the paper is organized as follows. Section II outlines state-of-the-art approaches for latency prediction, the proposed framework of latency estimation in the fog environment is then presented in Section III. The performance of the framework is evaluated and validated through a case study by demonstrating path optimization in Section IV. Section V concludes the paper with some future research directions.

II. The Latency Estimation Approaches

As mentioned earlier, NCS estimates latency based on known network coordinates in virtual geometric vectors. An NCS virtually plots the network nodes in a multidimensional Euclidean space by mapping the delay into the measured distance. For instance, if a network node A has coordinate $(2,4)$ and knows that another node B has coordinate $(5,8)$ in a two-dimensional Euclidean model, node A can simply calculate its distance to node B as $dP_{A-B} = \sqrt{((5-2)^2 + (8-4)^2)} = 5$ without direct communication. While in an n-dimensional Euclidean model, the latency prediction formula is

$$dP_{A-B} = \sqrt{\sum_{i=1}^{n}(A_i - B_i)^2} \quad (2)$$

where dP_{A-B} is the prediction delay, A_i and B_i are their coordinates. Besides, the absolute relative error (RE)

$$RE = \frac{|E-RTT|}{\min(E,RTT)} \quad (3)$$

is used as the performance metric, where E is the estimated latency and RTT is the actual measured distance. Based on one-way delay (OWD) and/or round-trip time (RTT) measurement, various NCSs have been developed to estimate the latency among the networked nodes. OWD technique generally requires a system with highly synchronized clocking and a precise time stamping to aid the delay measurement, however, its utilization is limited because of undesired errors. For this reason, RTT is adopted for measurement and prediction of latency in the majority of NCS.

A. Landmark Based Coordinate System

1) Global Network Positioning (GNP)

Landmark based NCS relies on a small number of landmark nodes to compute synthetic coordinates. GNP [6] is a typical example in this stream that has two phases for positioning nodes, i.e., landmark phases and ordinary host phases. In the landmark phase, each landmark measures RTT to other landmarks, as shown in Figure 2(a), where three landmarks are shown in blue. In the following phase, an ordinary host measures its RTT to landmarks to work out its coordinate. The ordinary host nodes like $h1$ and $h2$, repeat the same process and get their coordinates (x_1, y_1) and (x_2, y_2) using simplex downhill method [7]. Based on the exact coordinates, each node quantifies internode distance prior to any direct communication. The performance of GNP deeply relies on the distribution of landmarks. It achieves 90 percent above accuracy when landmarks are ideally distributed, but it has poor performance with badly chosen landmarks. In the worst case, if the landmarks are not available, the entire system could potentially stop working.

2) Network Positioning System (NPS)

To overcome the constraints of GNP, Ng *et al.* upgrade their original GNP work to NPS [8], in which, NPS allows any arbitrary node to act as landmarks, managed by independent membership server. Any node can contact the membership server to query primary settings of landmarks, system hierarchy details and a list of referencing points for further probing. Then, the node engages in individual probing process for the determination of the updated positions of and the respective distance to such referencing points, until its position is stabilized. As demonstrated in Figure 2(b), fixed landmarks (layer 0) are starting points of the dependency hierarchy. By maintaining redundant dependency with interested referencing points, a landmark failure becomes much less critical. As a result, position consistency is achieved because it is more tolerant in coping with temporary landmark failures.

B. Distributed Network Coordinate System

1) Vivaldi

Unlike landmark based NCS that relies on predefined landmarks in latency estimation, Dabek *et al.* advocate another NCS called Vivaldi that does not need any dedicated infrastructure [9]. As an analogy to the natural length of mass springs, the prediction latency is assumed to be stable between any two nodes. As illustrated in Figure 2(c), the current length of each spring is treated as the distance between nodes locally. If the spring is stretched, it indicates that the RTT is over-estimated. While the spring is compressed, it suggests that the RTT is under-estimated. Likewise, the natural length of spring is a sign that the RTT is exactly estimated. According to Hooke's law, the natural length of springs among nodes a, b, c and d determine the distance between the four nodes. In such a coordinate space, Vivaldi sets synthetic coordinates to each host, so as to estimate the data transmission RTT with minimum error. A squared error function is used in [9] as below:

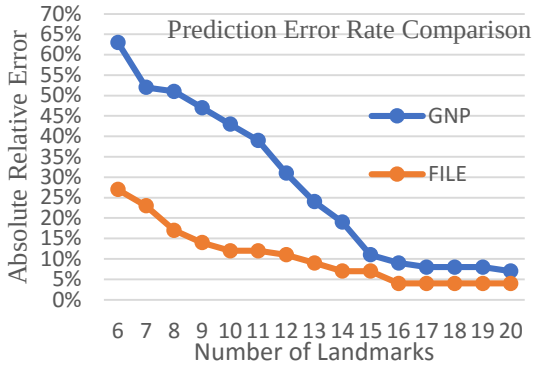


Figure 3: FILE Performance

$$E = \sum_i \sum_j \left(RTT_{ij} - ||x_i - x_j||\right)^2 \quad (4)$$

Where RTT_{ij} is the actual latency, $||x_i - x_j||$ is the estimated distance between nodes i and j. During the process of minimizing the errors, the nodes are pushed or pulled towards the perfect coordinates. In other words, Vivaldi moves each node x_i over a short distance at each interval to get close to the exact position step by step.

Initially, when a new node joins the system, Vivaldi assigns a random coordinate to it. As the new node starts communicating with another node, it surveys the RTT to that node and simultaneously learns the current coordinates of that node. This process is repeated each time whenever new nodes participate in this algorithm. Vivaldi nodes allow themselves to be moved by a small-time step(δ). With each movement, a node reduces its error with respect to others. Eventually, each node keeps its own coordinates in the hope that the prediction delay is equal to the measured delay. The authors of Vivaldi obtained 90 percent above accuracy in simulation of 1740 DNS servers on the Internet. Unfortunately, it is still an open issue to select the good value for δ in a complex network, which makes it difficult to balance prediction accuracy and convergence speed.

2) Pharos

Maintaining a global spring system with significantly changeable distances is very hard. To improve Vivaldi in short-link distance prediction, Chen *et al*. developed *Pharos* [10] that classifies nodes into two distinct groups according to their interspace. Both short-distance and long-distance coordinates are assigned to each node. As a result, both local cluster overlay for short-link distance and base overlay for medium or long-link distance are hierarchical as represented in Figure 2(d). Unlike a landmark node that must participate in the process of computing coordinates in the system, an anchor node needs only to respond to echo request regardless of actual participation of positioning process.

Whenever a new node joins Pharos system, it broadcasts its latency to locate the nearest anchor node and joins the respective cluster. After joining, it starts connecting many nodes in the same cluster, while some nodes in the cluster may establish base connection with nodes in remote clusters. At the end, Vivaldi is used to attain the two sets of coordinates for both overlays.

C. The Comparison

Each of the aforementioned NCS has advantages and disadvantages in a variety of use cases, while none can address all the challenges in computing coordinates. To begin with, the landmark system generally requires stable landmarks to serve other nodes, and the landmarks must be available and well-distributed. Furthermore, the change of distance between landmarks may not be timely updated to ordinary nodes, leading to an inaccurate estimation. Nevertheless, a good landmark based NCS empowers the accurate latency prediction without any direct communication between two nodes. Another strong point of landmark based system is its easiness to integrate a dedicate server that simplifies the positioning process and manages network coordinates for all level of nodes, regardless of their computing intelligence.

On the other hand, fully distributed NCS like Vivaldi does not require any dedicated device, instead, it relies on specific traffic patterns (piggybacking traffic) to enable self-motivated positioning. This dependency on traffic patterns limits the scope of application areas. Another weak point of Vivaldi is the shortage of mature architecture for managing network coordinates. It thoroughly relies on each node to adaptively position itself individually, which does not benefit much to low-end nodes. Despite that, such nodes can quickly respond to network changes, enabling an up-to-date latency estimation.

Algorithm 1: Landmark candidate (re)registration with FILE server through Vivaldi

// A Fog node x sends query to FILE server to determine if it is
// required to run Vivaldi.
Send ($Query$);
// FILE server responds to the node, either true or false. When
// true, it sends back an initiative coordinates x_i and a
// recommended list of other landmark nodes to probe.
Receive ($Response, x_i, (L_{1i}, L_{2i}, \dots, L_{ni})$);
// If required to run Vivaldi, the node becomes a possible
// landmark. Then it needs to evaluate its latency to other
// Landmark nodes, until the prediction error is acceptable
if ($Response == true$)
do { Vivaldi(rtt_{ij}, e_j, x_j); }
while ($e_i > AccptErr$);
// If not required to run Vivaldi, the node is a common node.
// Then it needs to calculate its coordinate towards Landmarks,
//where Dimensions defines an n-dimensional Euclidean model.
else if ($Response == false$)
Landmark ($SimplexDownhill(Dimen, Probe, Target)$);

III. FOG-BASED IOT LATENCY ESTIMATION (FILE)

As studied in Section II, all the NCS are developed on hypothesis that there will be an exact distance between any two network nodes. Besides, the NCS performs best in computing the coordinates for persistent objects, such as DNS servers on the Internet. However, the prosperity of IoT has significantly changed the environment, where myriads of things may frequently change their locations. In a fog network, enormous amounts of things and people may come and leave a from time to time. For instance, a driving vehicle is admitted as a fog node. Because lots of fog nodes intermittently join and disjoin a fog network, it is not always practical to calculate network coordinates for all of them. As such, none of the current NCSs

is immediately applicable to fog-based IoT. We therefore propose a new framework in this regard.

Although the multitude of nodes are very dynamic in fog based IoT ecosystem, many nodes are still relatively fixed along the cloud-fog-things continuum. For example, various servers (storage server, application hosting server, etc.) in both fog and cloud collaboratively reinforce other fog nodes, which generate enormous piggybacking traffic that can be used by Vivaldi for accurate internode latency estimation. When things join a fog network, such nodes can be engaged as the landmarks, to base the positioning of ordinary nodes in GNP. Hence, there are two phases in the proposed FILE system, i.e., Vivaldi phase and GNP phase. Next, we detail the algorithms in each process.

Algorithm 2: Landmark coordinate calculation in Fog (Vivaldi)

// Operation 1:
// FILE server initializes the coordinate of x_j *and inform a list*
// of nodes with their coordinates $(L_{1j}, L_{2j}, \dots, L_{nj})$ *at the*
// beginning, then the nodes measure their RTT in between to
// establish their coordinate and know their local error estimate.
Receive $(true, x_j, (L_1, L_{1j}), (L_2, L_{2j}), \dots, (L_n, L_{nj}))$;
Measure (RTT_{jL_m});
Calculate (e_j);
Update (x_j);
// Node j report its coordinate to FILE server.
Send $((j, x_j)$;
// Operation 2:
// When a new node i *join the system, the FILE server set its*
// coordinate x_i *approximately. It also informs the new node a*
// number of nodes with coordinates (including node j*) for*
// calibration.
Receive $(true, x_i, (\dots, (j, x_j), \dots))$;
// Node i *measures node j, learns an error estimate* e_j *and*
// coordinate x_j.
Measure (RTT_{ij});
Study (x_j, e_j)
// Node i *set an error estimate* e_i*, set the constants* c_e, c_c
Set (e_i, c_e, c_c);
// The main Vivaldi function
Vivaldi(rtt_{ij}, e_j, x_j)
{
// Sample weight balances local and remote error.

$$w = \frac{e_i}{e_i + e_j}$$

// Compute relative error of this sample.

$$e_s = \frac{|rtt_{ij} - ||x_i - x_j|||}{rtt_{ij}}$$

// Update weighted moving average of local errors.
$e_i = e_s \times c_e \times w + e_i \times (1 - c_e \times w)$
// Update local coordinates.
$\delta = c_c \times w$
$x_i = x_i + \delta \times (rtt_{ij} - ||x_i - x_j||) \times u(x_i - x_j)$
}
// Once this node gets accurate coordinate, it updates the
// FILE server with the latest information. report the
// up-to-date coordinate to FILE server
if $(e_i <= AccptErr)$
Send $((i, x_i)$;

Algorithm 3: Ordinary node coordinate calculation through Simplex Downhill algorithm

// A node x receives response from FILE server, required to
// run Simplex Downhill. The server also sets an original
// coordinate, recommended landmarks and their coordinates.
Receive $(False, x_i, (L_1, L_{1i}), (L_2, L_{2i}), \dots, (L_n, L_{ni}))$;
// Through the given coordinate, the node learns the
// dimension numbers used in the Euclidean model. While the
// probe is the interested landmarks, the target is IP addresses
// of any node.
Landmark$(SimplexDownhill(Dimen, Probe, Target))$
{
// Measure distance between probes
Measure (RTT_{iL_i});
// Measure distance between targets
Measure (RTT_{it_i});
// Then, run call Simplex Downhill function, to evaluated
// errors until the error estimate rate is acceptable.
while $(RE > AccepErr)$
do { $SimplexDownhill(Dimen, Probe, Target)$; }
// After that, the node reports its coordinates to FILE server
Send $((i, x_i)$;
}

A device in fog is assumed to connect and register to cloud before the device could communicate and exchange data with other entities. In regard to latency estimation, a dedicate membership server is setup in cloud to store system configuration parameters and to keep soft state about participating networked nodes. Hence, a FILE server in the cloud is liable to oversee the positioning of all level nodes encompassing the IoT ecosystem. A FILE server that provides primary system configuration information is able to dynamically selects some nodes as landmarks when the current landmark is unavailable or too heavily loaded. Initially, a node needs to query the FILE server regarding its alternative role in the system, i.e., landmark or ordinary nodes through Algorithm 1. If the FILE server determines the querying node is a candidate of landmarks, Algorithm 2 is used to calculate its coordinate, otherwise, Algorithm 3 finds the positioning of ordinary nodes.

In particular, Algorithm 1 brings two benefits to FILE system. One is to reflect the up-to-date inter-landmark latency, and the other is to update its position to FILE server that notifies the distance variation between landmarks. Next, the Vivaldi phase is presented.

A. Vivaldi Phase

This algorithm has two operations. In the first instance, a number of nodes measure their internode *RTT* to set up a reference system. Later on, the new-comer nodes figure out their coordinates using those referencing nodes.

We introduce FILE server in Algorithm 2 with management functions that significantly differentiated our approach compared to Vivaldi. FILE server initially sets the

coordinate uniformly in D-dimensional Euclidean model. Also, it assigns approximate coordinates at the beginning, resulting in the mitigation of positioning workloads. On top of this, due to the global view and thorough understanding about fog deployment, FILE server advises nodes of interest in the probe process, which improves the performance of overall latency estimation.

B. *GNP Phase*

As investigated in Vivaldi phase, each Vivaldi-running node must report its accurate coordinate to FILE server. Grounded on the harvest of nodes and their coordinates, FILE server not only provides the comprehensive supports for Vivaldi running nodes, but also supplies all level of landmarks along the cloud-fog-things continuum. As delineated in Figure 1, there are five function layers in the deployed continuum including global cloud, regional core, neighbourhood access, customer premises equipment gateway and things. Such deployment seeks to place processing where it is just-in-need, which creates a classic scenario of the utilization of landmark based NCS. Thereupon, we inspect the detailed operations.

Landmark operation and succeeding ordinary nodes operation are the components of this GNP phase. In the landmark operation, traditionally, it is to use certain number of fixed nodes as landmarks. Theoretically, there shall be at least D+1 landmarks in D-dimensional Euclidean model. Though increasing the number of landmark nodes contributes to higher accuracy, the distribution of landmarks is critical in such NCS. Since the landmarks are fixed at some location, it is hard to timely reflect the network topology change. However, the landmarks can be flexibly and dynamically selected from the potential landmark list under control of FILE server, through which, the landmarks are ideally maintained to warranty accuracy and liability.

Although the distance between landmarks has been estimated in Vivaldi phase and reported to FILE server, it is also measured in the landmark operation for further calibration. The measured and estimated values are reported to FILE server for further calibration, in this way, the server sets the landmark coordinates accurately. Subsequently, the ordinary nodes measure the RTT to the selected landmarks, calculate their coordinates using Simplex Downhill algorithm. Algorithm 3 is used in GNP phase.

As long as the coordinates are precisely estimated, the latency prediction is easily achieved through Formula 2. Beyond that, it is possible to predict latency between resource-limited nodes that cannot run FILE, e.g., a bar code scanners collecting visitor ID information. Such latency may be quantified on the uplink application server that collects the latency from the scanner based on time tag, plus the accurately predicted latency between application servers.

To sum up, FILE assumes that things require to be connected either to fog, cloud or both after their registration. A hybrid of centralized and distributed network coordinate methods is used to position nodes in the complicated ecosystem. This FILE framework empowers perfect landmark suppliers, through which, ordinary nodes calculate their coordinates individually. It enables the latency estimation among all nodes in fog environment. Thereafter, we evaluate the accuracy performance against GNP and showcase the advantages.

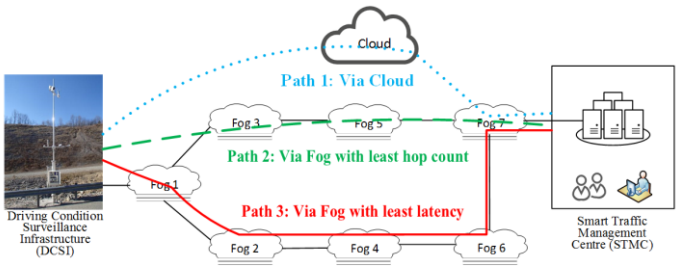


Figure 4: The Real-time Driving-condition Information System

Table 1: Data Setting in the Simulation

Data Settings	Value
Fog to Cloud Delay	500ms
Fog to Cloud Jitter	100ms
Fog to Fog Delay	10ms
Fog to Fog Jitter	2ms

IV. Performance Evaluation and Case Study

A. *Performance Evaluation*

The absolute relative error (RE) is studied as the performance metric, in comparison the accuracy of FILE with GNP. Attempt to acquire the latency from fog to cloud, we inspect twenty home routers to Amazon and Azure cloud in both Sydney and Melbourne. Over and above that, a couple of well-known cloud speed testing webpage tools [11, 12] are also used as our test-result reference. An average of 500ms of RTT is observed. Then, each router is treated as one fog node along the home routers to data center path. The average inter-node latency is about 10ms and Chang *et al*. also demonstrate same in [13]. Thus, we set the inter-fog node latency as 10ms. The inter-fog and things-to-cloud latency data are used in our experiment with 100 computers that are distributed in 6 Fog networks. These Fog networks are connected to Cloud with various distance. After that, FILE algorithm is conducted on this setting. Figure 3 plots the prediction error rate with different numbers of landmarks. FILE achieves more than 90 percent accuracy with 13 or more landmarks, while 16 landmarks are required to achieve similar accuracy in GNP. More interestingly, FILE allows to use any waypoint as referencing point for positioning to improve the estimation accuracy.

In summary, the proposed framework assists to predict the delay between things, fog and cloud nodes, before their direct communication. Because the latency prediction is one of the key parameters to optimize the fog performance, it brings comprehensive benefits to the entire IoT ecosystem including service discovery, IoT placement, content distribution, path selection, and so on. Next, a case study is conducted to demonstrate the advantages of the proposed framework further.

B. *Case Study*

Poor weather is one of the primary reasons for traffic accidents along the millions-kilometres Chinese expressways [14]. In particular, the ground fogs in lake or mountain area can suddenly cause visibility down to the meter-level in a second. In this case, the drivers brake hard subconsciously

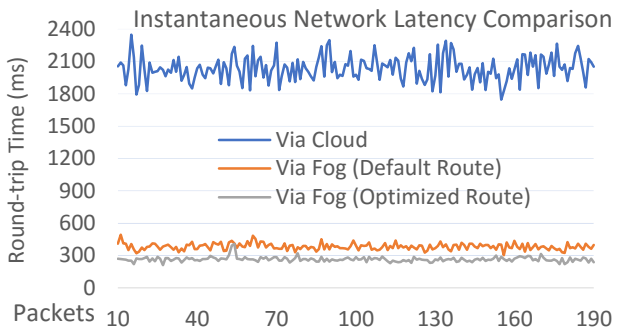


Figure 5: The Latency Comparison on Packet Size 100

Table 2: Latency with Various Packet Size

	Average Latency (milliseconds)			
Packet size	100	500	1000	1500
Case 1	2038.31	2049.82	2032.06	2057.24
Case 2	381.54	381.23	386.37	388.58
Case 3	264.80	264.65	265.57	261.93

causing disasters of rear end collisions and massive highway pileups. Due to the difficulties of forecasting such weather, RDIS (real-time driving-condition information system) is deployed to make early warnings ahead of such areas. This system is composed of weather sensors, cameras, roadside service units, infotainment systems, patrol and rescue vehicles. As displayed in Figure 4, fog nodes are deployed to enable RDIS along the roads. FILE is used for quick response to the protection of lives and assets, as it globally oversees and manages the coordinates infrastructure.

Landmarks are dynamically maintained to base accurate latency prediction for the nodes, which warrants the least latency of data transmission. We use GNS3, an open source network simulator used by networking professionals [15], to deploy our testbed. Cisco 7204 routers are used to simulate all the network nodes, except for delay and jitter generator that is simulated by WANem [16]. Table 1 presents the data set applied in the simulation. We configure the testbed to generate three data paths to carry data from DCSI (driving condition surveillance infrastructure) to STMC (smart traffic management centre). The data paths include cloud (case 1), fog with the least hop count (case 2), and fog with the least latency (case 3).

Figure 5 depicts the instantaneous end-to-end latency changes in the three cases when the packet size is 100 bytes. In case 1, when the data must go through the cloud, the maximum, minimum and average delays are 2349.25 ms, 1745.18 ms, and 2038.31 ms respectively. Case 2 illustrates the fog routing without NCS, where the maximum, minimum and average delays are 493.11 ms, 303.86 ms, and 381.54 ms respectively. When FILE predicts the delay in real-time, each node can send data over the route with the least latency. Hence, in case 3 the maximum, minimum and average delays are 406.77 ms, 210.50 ms, and 264.80 ms respectively between DCSI (sensing) and STMC (processing). Also, we examine the latency with the varying packet size of 500, 1000 and 1500 bytes. Overall, the end-to-end latency does not change much in all the cases. Table 2 presents the result of the varying size of packets.

V. Conclusion

Since we can only minimise rather than eliminate the end-to-end network latency, latency prediction is of paramount importance for the fog optimization. By taking advantage of both Vivaldi (fully distributed) and GNP (landmark-based) alogrithm, our proposed FILE can predict the latency with high accuracy for both fog nodes and things. Following the illustrated benefits, we showcase the rerouting based on the lowest end-to-end latency to further expedite the data transmission. The result concludes that FILE gives very positive input for real-time applications. As the low latency is the driving reason for fog adoption, we believe it is worth much more research, and in particular, we are going to investigate the FILE convergence behaviour on different overlays and Fog deployment strategies.

Acknowledgement

Authors wish to thank Dr. Khandakar Ahmed for his valuable contribution in the latency estimation algorithm design.

References

[1]. J. Li, J. Jin, D. Yuan, M. Palaniswami, and K. Moessner, "EHOPES: Data -centered Fog platform for smart living", in *Proceedings of ITNAC*, pp. 308-313, Sydney, Australia, Nov. 2015.

[2]. A. Abdou, A. Matrawy, and P.C. Van Oorschot, "Accurate one-way delay estimation with reduced client trustworthiness," *IEEE Communications Letters*, vol. 19, no.5, pp. 735-738, Mar. 2015.

[3]. "Bandwidth, Packets per Second, and Other Network Performance Metrics," Available: http://www.cisco.com/c/en/us/about/security-center /network-performance-metrics.html, Accessed on 9/04/2017.

[4]. F. Bonomi, R. Milito, P. Natarajan, and J. Zhu, "Fog Computing: A platform for Internet of things and analytics," in *Big Data and Internet of Things: A Roadmap for Smart Environments*, pp. 169-186, 2014.

[5]. G. Armitage, and A. Heyde, "REED: Optimizing first person shooter game server discovery using network coordinates," *ACM TOMM*, vol 8, pp. 2-20, May 2012.

[6]. T.E. Ng, and H. Zhang, "Predicting Internet network distance with coordinates-based approaches," in *Proceedings of 21st Annual Joint Conference on IEEE INFOCOM*, vol. 1, pp. 170-179, New York City, USA, Jun. 2002.

[7]. K. I. McKinnon, "Convergence of the Nelder--Mead Simplex method to a nonstationary Point," *SIAM Jounal on Optimization*, nol 9, no.1, pp. 148-158, May 1998.

[8]. T.E. Ng, and H. Zhang, "A network positioning system for the Internet," in *Proceedings of USENIX Annual Technical Conference*, pp. 141-154, Boston, USA Jun. 2004.

[9]. K. Dabek, R. Cox, F. Kaashoek, and R. Morris, "Vivaldi: A decentralized network coordinate system," *ACM SIGCOMM Computer Communication Review*, vol. 34, no. 4, pp. 15-26, Aug. 2004.

[10]. Y. Chen, Y. Xiong, X. Shi, B. Deng, and X. Li, "Pharos: A decentralized and hierarchical network coordinate system for internet distance prediction," in *Proceedings of GLOBECOM*, pp. 421-426, Nov. 2007.

[11]. M. Leonhard, "CloudPing.info," Available: http://www.cloudping.info/, Accessed on 19/04/2017.

[12]. "Cloud Network Test," Available: http://www.cloudharmony.com /speedtest, Accessed on 19/04/2017.

[13]. Y.C. Chang, and J.P. Sheu, "An energy conservation MAC protocol in wireless sensor networks," *Wireless Personal Communications*, vol. 48, no. 2, pp. 261-276, Jan. 2009.

[14]. J. Zhao, and W. Deng, "Traffic accidents on expressways: new threat to China," *Traffic Injury Prevention*, vol. 13, no. 3, pp. 230-238, Dec. 2011.

[15]. C. Welsh, *GNS3 network simulation guide*, Packt Publ., 2013.

[16]. H.K. Kalitay, and M.K. Nambiarz, "Designing wanem: A wide area network emulator tool," In *proceedings of Third COMSNETS,* , pp. 1-4, Bangalore, India, Jan. 2011