Fog Computing: Optical Scheme to Improve Mobile Users in MMVEs

Zhongtao Li，Kai Wang，Xiangyu Kong
School of Information Science and Engineering University of Jinan
Shandong Provincial Key Laboratory of Network Based Intelligent Computing
Jinan, P.R. China，250022
ise_lizt@ujn.edu.cn

***Abstract*—With the explosive growth of mobile user in Massively Multiuser Virtual Environments (MMVE)，it faces a challenge, how to deal with the request of rapid growth and reduce traffic to shorten the response time. Fog computing is a novel solution to free users from the requirement of hardware. We introduce a tree network to manage connecting layer in fog computing, which is termed as CAN (content-addressable network) Tree. All resources are mapped in a 2-D space. Each fog server handles a zone in the space. CAN Tree manages fog servers joining and departure to avoid fragment, and provide better routing in fog servers.**

Keywords—fog computing; MMVE; CAN Tree

I. INTRODUCTION

By meaning of cloud computing, Massively Multiuser Virtual Environments (MMVE) has capability to meet the highest requirements relative to flexibility, robustness and consistency [1]. Cloud computing provides Software-as-a-Service (SaaS), Infrastructure-as-a-Service (IaaS), or Platform-as-a-Service (PaaS) to meet distinguished needs. It was a perfect solution in last few years. As mobile device become more and more ubiquitous, it's been part of people's daily lives. The mobile devices become the major service objects. As reported, over 90% mobile users are directly or indirectly relied on cloud service.

Cisco has predicted that each person will has 6.58 online devices on average in 2020. The key issue is how to deal with a mass of demand and accommodate the rapid increasing of mobile network traffics [2]. Compared with cloud computing, fog computing extends the location-based services. Fog computing is a middleware between cloud computing and mobile applications to assure quick response and reduce traffic [3]. Hence fog computing could provide better service for mobile traffics. In MMVE, mobile user demand on high-quality mobile services at anywhere [4]. It's a challenge to address fragment and routing issues.

In this article, we outline the main features of Fog computing MMVE and describe its concept, architecture and routing mechanism. Lastly, we show the simulations and evaluate it.

Supported by Doctoral Foundation of University of Jinan

II. SCENARIOS

The architecture of MMVE in fog computing is shown in Fig. 1. Each fog server maintains three kinds of connections: connection between fog server and mobile user, connection between fog servers and connection between fog server and cloud server.

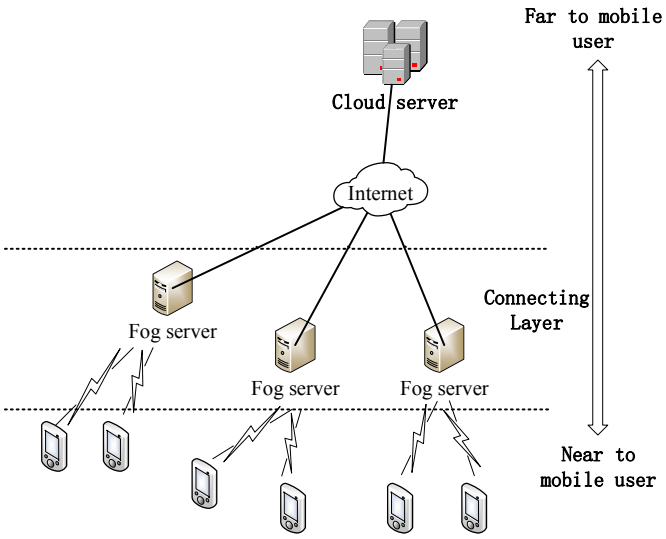

Fig. 1 Fog computing architecture

A. *Connection between fog server and mobile user*

The fog server is responsible for the mobile user via local connection. Retrieval resources from fog server instead of cloud server reduced cloud server load and shortened response time by orders of magnitude, simultaneously improved the user experience dramatically [5].

The fog server played the role of a bridge between mobile users and cloud [6]. Fog servers directly communicate with mobile users via local connections, i.e., Wi-Fi, ZigBee or Bluetooth. Fog server performs as a local cloud server, however, it is much smaller and only server its local connected mobile users. It's termed as fog cluster [7].

B. Connection between fog servers

Fog servers are located at edge of MMVE and proximity of mobile user. A fog server could be a dedicated device or virtualized device. It has enough storage, computing and communication ability [8]. And then, each fog server caches a portion of resource from cloud server. All fog servers cache all the resource, and they are responsible for providing services to mobile users in the edge of the network [9].

If mobile user requests a resource which cached by its fog server, fog server provides the resource immediately. Otherwise, fog server must retrieve the resource from other fog server. A key issue is how to organize fog servers and allocate resources with high efficiency and low cost [10].

C. Connection between fog server and cloud server

Cloud server is responsible for the storage of resources. As the MMVE center, it connects with fog servers over Internet so as to leverage the rich computing and provide content resources of cloud [11].

III. DESIGN FROM RESOURCES DISTRIBUTION AND ROUTING

In order to organize resources, we mapped resources into a two-dimensional space. Each resource is allocated a unique coordinate via hash function. A fog server caches a portion of resources, i.e., a fog server handles an area in this two-dimensional space (It's termed as zone). Fog servers need to communicate and synchronize with each other. In order to avoid fragment and improve routing efficient, we introduce CAN and CAN Tree to manage connecting layer.

A. Resources Distribution

As our design, the resources mapped in a 2-D CAN space. The CAN space is divided amongst the fog server currently in MMVE. To allow the connecting layer to grow incrementally, a new fog server which joins the system must be allocated its own coordinate space (a portion of CAN space). An existing fog server will split its own zone in half, retaining half and handing the other half to the new fog server [12].

The new fog server needs three steps to join in connecting layer as follows (shown as Fig. 2):

1. Firstly, the new fog server has to detect a fog server already in the connecting layer (Bootstrap).

2. Secondly, via the CAN routing mechanisms and optimal algorithm, it must find a fog server whose zone will be shared.

3. Finally, the new fog server must notify the neighbors.

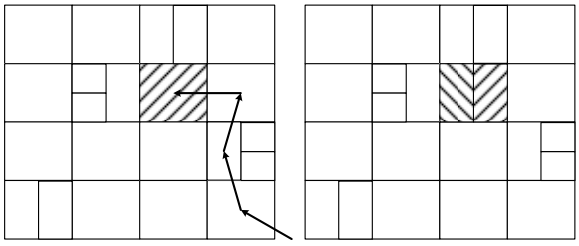

Fig. 2 New fog server joining

B. Routing

A fog server is able to retrieve resources from other fog servers so as to provide global resources to local mobile users. As we know, the efficiency of routing with greedy algorithm is low. So it is unwise to adopt a greedy algorithm. Instead of greedy routing, we proposed a tree network for routing. The core idea is to build a P2P tree [13], which is termed as "CAN Tree" [14] via long links (shown as Fig. 3). Each fog server in connecting layer is a node of the CAN tree. Shown as Fig. 4, each fog server only connects with its direct relative, i.e., parent and children, in the CAN Tree.

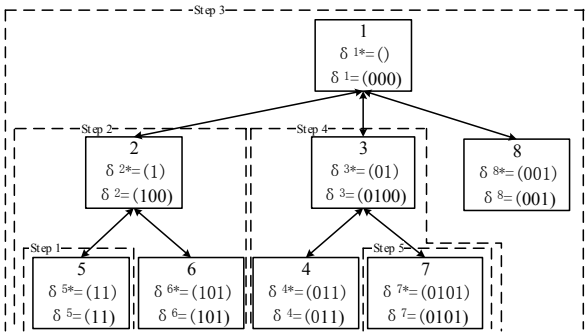

Fig. 3 CAN Tree

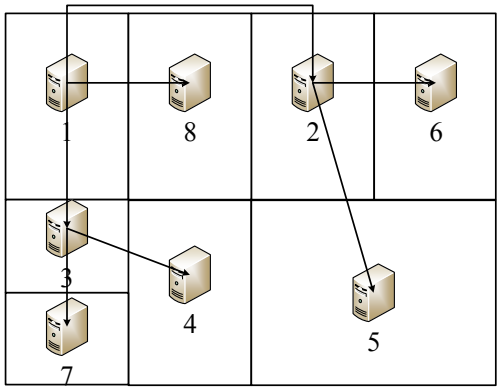

Fig. 4 The fog servers in connecting layer

For routing, it needs "zone-code" to decide where is the routing target [11]. We have proposed the zone-code in "Using Zone Code to Manage a Content-Addressable Network for Distributed Simulations" [15]. A zone-code is a bunch of binary numbers, which is the serialization of historical records.

In paper [15], we have deduced three facts as following:

"Fact 1: The zone-code is a prefix code." [16]

"Fact 2: In CAN Tree, the original zone-code of node p is the prefix of zone-codes of all nodes in the sub-tree rooted at node p."[16]

Let δ^{p*} denote the node p original zone-code. δ^{p*} is the node p first zone-code, and δ^{p*} is constant. After splitting, $\delta^{p*} \neq \delta^{p}$.

As described above, we could deduce the following: When the current node *p*'s δ^{p*} is the prefix of the target node *c*'s δ^{c*}, the node *c* is in the sub-tree rooted at node *p*. Hence, the routing table is composed by the short neighbor links, the long direct relative links, and its original zone-code δ^{*} (shown as Fig. 5).

Fog server 1 δ^{1}=(000) δ^{1}*=null

Short Link

Node ID
3
8

Long Link

Node ID	Zone code	
-	-	Parent
2	100	Child
3	0100	Child
8	001	Child

Fig. 5 Routing table

When a fog server *s* desires a resource, it retrieves via routing tables. Firstly, it obtains the coordinate of target resource (point *p* in CAN space) via hash function. Secondly, according to routing tables, send a "request" message to the fog server *e* whose zone covers point *p*. Finally, server *s* received reply and retrieve the desired resource.

Each fog server maintains its routing table to keep short and long links. Hence the current node *c* does not have any information (including its zone-code) about target fog server *e*. In order to routing in CAN Tree, we also need the zone-code of target fog server *e*. Our solution is calculating target zone-code in each hop as follows:

1. Calculate δ^{e} : δ^{e} is deduced from Equation 1. However, Equation 1 depends on $|\delta^{e}|$ (length of zone-code of fog server *e*), which is an unknown factor. In calculating, it assumes $|\delta^{e}| = |\delta^{c}|$ (fog server *e* and *c* have same length zone-codes). Consequently, δ^{e} can be deduced from Equation 1.

$$Z^{p}_{x,y} = \left\{ (x,y) \left| \begin{array}{l} (\delta^{p}_{x})_{10} \cdot \frac{w}{2^{|\delta^{p}_{x}|}} \le x < ((\delta^{p}_{x})_{10}+1) \cdot \frac{w}{2^{|\delta^{p}_{x}|}} \\ (\delta^{p}_{y})_{10} \cdot \frac{h}{2^{|\delta^{p}_{y}|}} \le y < ((\delta^{p}_{y})_{10}+1) \cdot \frac{h}{2^{|\delta^{p}_{y}|}} \end{array} \right. \right\}$$

Equation 1 Zone point set in 2-dimensional space

2. Choose next hop: The current fog server *c* checks if its δ^{c*} is prefix of δ^{e} , which is from last step. If it is, it sends the "request" message to its child which shares the longest common prefix with δ^{e} . Otherwise, it sends the message to its parent.

Repeat the above steps, the message will eventually reach the target fog server. Because each hop make current fog server δ^{p*} has longer common prefix with target δ^{e}, the routing process is simple and efficient.

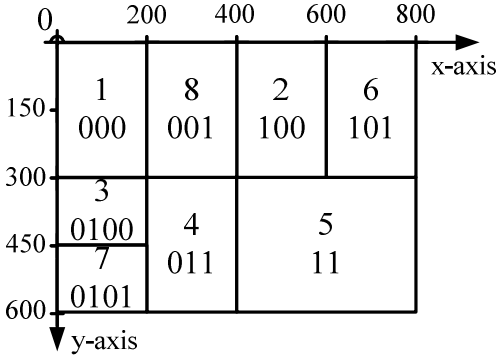

Fig. 6 Resources distribution in connecting layer

C. Routing Example

For example, a mobile user desired a resource whose coordinate is (100,500) via hash function. It directly connected to fog server 5. Shown as Fig. 3 and Fig. 6, the fog server 5 does not cover (100,500) and has $\delta^{5*} = (1,1)$ $\delta^{5} = (1,1)$. The routing procedure is as follows:

1. The fog server 5 assumes that server *e* is the target. Since $|\delta^{5}| = 2$, set $|\delta^{e}| = 2$. Thus, calculate δ^{e} as follows:

$$Z^{e}_{x,y} = \left\{ (x,y) \left| \begin{array}{l} (\delta^{e}_{x})_{10} \cdot \frac{w}{2^{|\delta^{e}_{x}|}} \le x < ((\delta^{e}_{x})_{10}+1) \cdot \frac{w}{2^{|\delta^{e}_{x}|}} \\ (\delta^{e}_{y})_{10} \cdot \frac{h}{2^{|\delta^{e}_{y}|}} \le y < ((\delta^{e}_{y})_{10}+1) \cdot \frac{h}{2^{|\delta^{e}_{y}|}} \end{array} \right. \right\}$$

$$\delta^{5} = \underset{x\ y}{11} \Rightarrow \begin{cases} \delta^{5}_{x} = 1 \\ \delta^{5}_{y} = 1 \end{cases} \Rightarrow \begin{cases} |\delta^{e}_{x}| = |\delta^{5}_{x}| = 1 \\ |\delta^{e}_{y}| = |\delta^{5}_{y}| = 1 \end{cases}$$ then

$$\Rightarrow (100,500) \in \left\{ (x,y) \left| \begin{array}{l} (\delta^{e}_{x})_{10} \cdot \frac{800}{2^{1}} \le x < ((\delta^{e}_{x})_{10}+1) \cdot \frac{800}{2^{1}} \\ (\delta^{e}_{y})_{10} \cdot \frac{600}{2^{1}} \le y < ((\delta^{e}_{y})_{10}+1) \cdot \frac{600}{2^{1}} s \end{array} \right. \right\}$$

$$\Rightarrow (\delta^{e}_{x})_{10} = 0 \quad and \quad (\delta^{e}_{y})_{10} = 1$$

$$|\delta^{5}| = 2 \Rightarrow |\delta^{e}| = 2$$

$$\Rightarrow \begin{cases} \delta^{e}_{x} = 0 \\ \delta^{e}_{y} = 1 \end{cases}$$

$$\Rightarrow \delta^{e} = (0,1)$$

Since $\delta^{5*} = (1,1)$ is not the prefix of $\delta^{e} = (0,1)$, it sends the message to the parent(fog server 2).

2. The fog server 2 repeated calculation in last step and obtains $\delta^e = (0,1,0)$ dependent on $|\delta^2| = 3$. Since $\delta^{2*} = (1)$ is not the prefix of $\delta^e = (0,1,0)$, it send the message to the parent (fog server 1).
3. The fog server 1 repeated calculation in last step and obtains $\delta^e = (0,1,0)$ dependent on $|\delta^1| = 3$. Since it's the root of CAN Tree, it sends the message to the child (fog server 3) which shares the longest common prefix with $\delta^e = (0,1,0)$.
4. The fog server 3 repeated calculation in last step and obtains $\delta^e = (0,1,0,1)$ dependent on $|\delta^3| = 4$. Since $\delta^{3*} = (0,1)$ is the prefix of $\delta^e = (0,1,0,1)$, it send the message to the fog server 7 which is the target.

IV. EVALUATION

Our solution extended CAN routing Via long links. Simultaneously, routing is optimized such as smaller routing path and more routing flexibility. Since each routing hop moves to fog server which is closer to the target, the routing is convergent procedure.

Fig. 7 show CAN Tree and greedy respectively. The number of nodes correspond to network size. The average path length is measured by the number of hops to target fog server. In simulations, CAN tree routing are better than in the original greedy routing, the path length increases much slower.

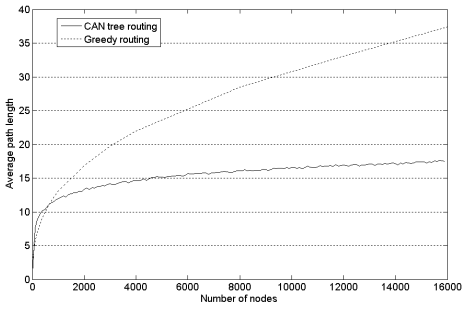

Fig. 7 Evalution routings

V. CONCLUSION

Fog computing is designed to cope with quick response edge device. It perfectly meets the requirements of MMVE. The fog servers in connecting layer cached all resource, and provide service to geographically close mobile users. An important issue is how to distribute and retrieve resources in connecting layer. In this paper, we proposed a novel solution with CAN and CAN Tree. We can see that CAN and CAN Tree significantly improves routing performance in simulations.

REFERENCES

[1] Z. Li, "Content-Addressable Network for Distributed Simulations," University of Duisburg-Essen, 2014.

[2] Y. Lin, H. Shen, "Leveraging Fog to Extend Cloud Gaming for Thin-Client MMOG with High Quality of Experience," IEEE 35th International Conferen on Distributed Computing Systems (ICDCS), 2015:734-735.

[3] C. Bezerra and C. Geyer. A load balancing scheme for massively multiplayer online games. Multimedia Tools Appl, 2009.

[4] Y. Lin, H. Shen, "Cloud Fog: Towards High Quality of Experience in Cloud Gaming," 44th International Conference on Parallel Processing (ICPP), 2015:500-509.

[5] Tom H. Luan, Longxiang Gao, Zhi Li, Yang Xiang, Limin Sun, "Fog Computing: Focusing on Mobile Users at the Edge," CoRR abs/1502.01815, 2015

[6] F. Bonomi, R. Milito, J. Zhu, S. Addepalli, "Fog Computing and Its Role in the Internet of Things," in Proceedings of ACM MCC, pp. 13–16, 2012

[7] L.M. Vaquero, L. Rodero-Merino, "Finding your Way in the Fog: Towards a Comprehensive Definition of Fog Computing," Computer Communication Review (CCR) 44(5):27-32, 2014

[8] F.Bonomi, R. Milito, P. Natarajan, J. Zhu, "Fog computing: A platform for internet of things and analytics," Big Data and Internet of Things: A Roadmap for Smart Environments. Springer International Publishing, 2014: 169-186.

[9] J. Su, F. Lin, X. Zhou, L. Xing, "Steiner Tree Based Optimal Resource Caching Scheme in Fog Computing,". Wireless Communication Over Zigbee for Automotive Inclination Measurement China Communications, 2015, 12(8):161-168.

[10] M. Aazam, E. N. Huh, "Fog computing and smart gateway based communication for cloud of things," Future Internet of Things and Cloud (FiCloud), 2014 International Conference on. IEEE, 2014: 464-470.

[11] Z. Li, K. Wang, X. Kong, et al. "Zone-code based optimal connecting layer scheme in fog MMVE" in Informative and Cybernetics for Computational Social Systems (ICCSS), 2016 3rd International Conference on. IEEE, 2016: 93-96.

[12] Sylvia Ratnasamy, Paul Francis, Mark Handley, Richard Karp, Scott Shenker, "A Scalable Content Addressable Network". In Proceedings of ACMS1GCOMM, 2001.

[13] D. Heutelbeck, "Distributed space partitioning trees and their application in mobile computing," Fernuniv., Fachbereich Informatik, 2005.

[14] Z. Li, T. Weis, "CAN Tree Routing for Content-Addressable Network," Sensors & Transducers, 2014, 162: 124-130.

[15] Z. Li, T. Weis, "Using zone code to manage a Content-Addressable Network for Distributed Simulations," in 2012 IEEE 14th International Conference on Communication Technology (ICCT): IEEE, 2012, pp. 1350-1357.

[16] Z. Li, S. Zhao, C Ge, S Gao "A Novel CAN Tree Coordinate Routing in Content-Addressable Network," Sensors & Transducers Journal, 2014, 178:99-105.