EdgeToll: A Blockchain-based Toll Collection System for Public Sharing of Heterogeneous Edges

Bowen Xiao[1], Xiaoyi Fan[2], Sheng Gao[3] and Wei Cai[1]
[1]School of Science and Engineering, The Chinese University of Hong Kong, Shenzhen, Guangdong 518172 China
[2]Department of Electrical and Computer Engineering, The University of British Columbia, Vancouver V6T1Z4 Canada
[3]School of Information, Central University of Finance and Economics, Beijing 100081 China
Email: bowenxiao@link.cuhk.edu.cn, xiaoyif@ece.ubc.ca, sgao@cufe.edu.cn, caiwei@cuhk.edu.cn

***Abstract*—Edge computing is a novel paradigm designed to improve the quality of service for latency sensitive cloud applications. However, the state-of-the-art edge services are designed for specific applications, which are isolated from each other. To better improve the utilization level of edge nodes, public resource sharing among edges from distinct service providers should be encouraged economically. In this work, we employ the payment channel techniques to design and implement EdgeToll, a blockchain-based toll collection system for heterogeneous public edge sharing. Test-bed has been developed to validate the proposal and preliminary experiments have been conducted to demonstrate the time and cost efficiency of the system.**

***Index Terms*—edge, blockchain, pricing, system, testbed**

I. Introduction

Cloud computing has transformed everything as a service [1] nowadays. Nevertheless, latency sensitive cloud applications, e.g. interactive multimedia systems, are still struggling from the unacceptable delay introduced by the network round-trip time (RTT). Edge computing [2], an emerging computing paradigm in future 5G network [3], is designed to improve the quality of services (QoS) for time-critical cloud applications, especially in the mobile scenarios. In contrary to the remote cloud server, edge nodes are nearby infrastructures, a.k.a. cloudlet, providing software services to the end users. On the other words, edge serves as an intermediate between a terminal device and the cloud to facilitate computing at the proximity of data sources.

However, the state-of-the-art edge platforms are specifically designed for customized applications, rather than a public service for various applications and distinct user groups. For example, an edge node deployed for power plants will not handle video processing requests from a mobile game player, even it has been staying in an idle status for a long time. The isolation among different applications significantly reduces the utilization level of edge resource, which still requires continuous maintenance work. Despite security considerations, one critical issue in preventing public edge resources sharing is the lack of motivation for the edge infrastructure provider. An incentive mechanism is still facing challenges and technical issue from a real-world implementation. First, there is no public third-party trustworthy proxy to collect toll fees for multiple edge service providers. The heterogeneous nature of edge deployments requires a transparent resource bidding platform operated independently. Second, the toll fee for a general service request is relatively small. It may be hard to use legal tender for resource pricing. Third, distinct edge platforms may adopt different pricing schemes and credit systems, which prevent the resource consumers from leveraging available edges nearby.

On the other hand, the blockchain system [4] has introduced a transparent, trustworthy and unformed ecosystem for multiple independent parties. This feature makes it a perfect solution to the toll collection problem in heterogeneous public edge sharing. The immutable and open source smart contracts [5] driven by blockchain enables a transparent profit distribution scheme among multiple edge service providers in an autonomous manner. In addition, by leveraging cryptocurrency, the edge nodes from multiple service providers are able to use a unified, fine-granularity, and transparent pricing method to charge users. From the users' perspective, it is convenient to spend one cryptocurrency in consuming resources from multiple parties, which highly increase the availability of edge services. In fact, the blockchain-based toll system can minimize the cost for both providers and the users, given the business rules are well-defined: there will be no centralized operators to pocket the difference as its profit.

Nevertheless, existing blockchain systems are still in their preliminary stages. Most well-known blockchain systems are suffering from the high cost of gas fee and unacceptable latency introduce by the Proof-of-Work (PoW) [6], while the others, who minimize the overhead by adopting other consensus models (e.g., Delegated-Proof-of-Stake from EOS[1]), are not well recognized as full decentralized platforms. This imposes a big challenge for the toll collections systems for frequent but small amount transactions, e.g. the one we are proposing. In this work, we design and implement EdgeToll, an open source toll collection system for heterogeneous public edge sharing. By leveraging the technique of payment channel, EdgeToll provides a transparent, quick and cost-efficient solution to encourage participation of edge service providers.

The remainder of this paper is organized as follows. We reviewed related work in Section II and presented the overview of the proposed system in Section III. We then present the technical design and test-bed implementation in Section IV and Section V, respectively. Based on our development, the

[1]https://eos.io/

experiments are conducted to validate our system in Sections VI. Section VII concludes this paper.

II. Related Work

A. Cloud and Edge Integration

Integrating edge to cloud platform involves a series of research topics in data and computational offloading. Traditional approach offloading schemes adopt virtualization techniques to host multiple copies of virtual machines in both cloud and edges [7], while another group of researchers has investigated the possibility of dynamic code partitioning [8] [9]. However, despite the form of offloading, the edge nodes intrinsically provide resource services for end users through direct network connectivity. In this work, we assume the end users are requesting micro-services installed in the edge nodes to simplify our model.

B. Decentralized Applications and Payment Channel

The blockchain [4] data structure, together with the peer-to-peer (P2P) system and the proof-of-work (PoW) [6] consensus model, makes the decentralized ledger for cryptocurrencies became a reality [10]. Known as Blockchain 2.0, Ethereum [11] was implemented to facilitate decentralized applications (dApps) [12], which have been extended to various areas, including initial coin offerings, social networks, networked games, and IoT. In this work, we write smart contracts to develop a decentralized toll collection system for edge service sharing among multiple parties, which perfectly demonstrate the benefits of dApps. To overcome the long response latency and the monetory costs introduced by frequent transactions, the payment channel [13] technique is designed for "off-chain" transactions, which allow users to make multiple token exchanges with a minimum number of smart contract invocations. The state-of-the-art payment channels can be classified into two types: *uni-directional payment channel* and *bi-directional payment channel*. An uni-directional payment channel only allows single directional transactions, while a bi-directional payment channel [14] allows both parties to send transactions. The duplex payment channel is composed of two uni-directional payment channels, which allows transactions to be sent from both directions.

III. System Overview

In this section, we present the overview for the proposed system. As illustrated in Fig. 1, edge nodes, and end users should register corresponding addresses in the blockchain before they can participate in the proposed system. These addresses, being accessed with the private keys known to their owners, are the destinations of cryptocurrency tokens. From the perspective of the end user, he/she needs to discover nearby edge nodes that provide the services and pay the corresponding cost after the services are delivered. In case of multiple edge nodes available, the user can choose one from these candidates, in terms of their performance and offered price. On the other hand, the edges can select their service recipients from the perspective of task complexity and bidding price, if there are multiple end users competing for the same resources. Note that, all payments should go through a smart contract to guarantee the transparency of the system.

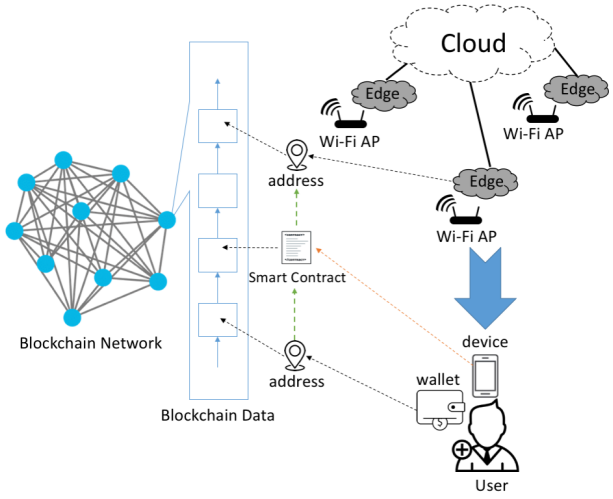

Fig. 1. The System Architecture for EdgeToll

However, the proposed system will consume significant tokens as gas fee when the users pay their toll to the edge nodes. Also, the long transaction delay will also disable high frequent service delivery to the users. Therefore, we may need to create payment channels to minimize the overhead of payment transactions. Nevertheless, it is impossible for a user to establish payment channels to a lot of edge nodes, since there will be another overhead here: the users need to lock certain among of tokens to open the channel, while he/she may only interact with one edge once.

IV. System Design

In order to solve the above issue, we employ an open source proxy as a service matching server and the payment intermediate. The first functionality of the proxy is to match the appropriate service provider and recipient. This process can be optimized with artificial intelligence (AI) algorithms. Alternatively, this process can be a result of a series of competitions and cooperation to be modeled with game theory. In our implementation, the proxy always adopts greedy algorithms to minimize users' cost and maximize the edge nodes' profit under different scenarios. The second role of the proxy is the intermediate of users and the edges. An end user only opens one payment channel to the proxy, while the proxy open payment channels to the edges. Of course, different edge nodes from the same service provider may share the same blockchain wallet, which can significantly reduce the number of payment channels. In this work, since the payment from users to the proxy, from the proxy to the edge service providers, are uni-directional, we adopt the uni-directional payment channel.

Fig. 2 illustrates the sequence diagram for the proposed EdgeToll system. Edges can be deployed by any companies or individuals. For any edge who want to join in the EdgeToll public sharing platform, a registration to the proxy is required

as its initialization process. Through the registration, an edge is requested to provide its blockchain address and its IP address: the former one is serving as the destination of toll fees and the later one is how the end user's device identify the edge.

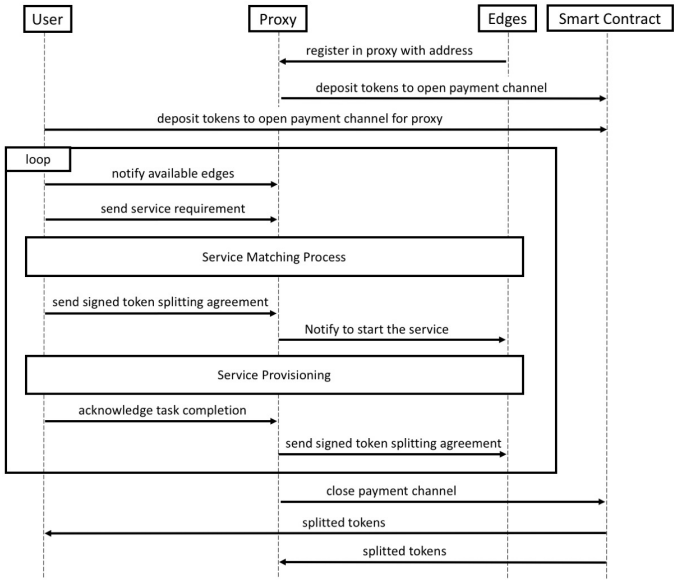

Fig. 2. Sequential Diagram for EdgeToll

After proxy receives edge's address, the proxy will evaluate edge's condition and invoke a smart contract to open a payment channel for the registered edge. The proxy deposits tokens into that contract and set the recipient to be the registered edge so that only the edge can withdraw the token. On the other hand, the end users are usually mobile terminals whose locations are changed over time. Once a user has a demand for edge resources, he/she needs to open the payment channel for the proxy through smart contracts. At the same time, the user also needs to discover nearby edges and notify the list of available candidates to the proxy. After the service requirement is sent from the user to the proxy, service matching process will be conducted to find the suitable pairs. After that, the user needs to sign a signature on an agreement to split the tokens and send it to the proxy. The proxy, the recipient of the signature, can validate the agreement with blockchain data, which is a no-cost operation, since it is a simple blockchain data reading function. After the validation, the proxy notifies the corresponding edge to deliver its service to the user. Once the user acknowledges the completion of service, the proxy will sign its token splitting agreement with the edge to deliver the edge's profit. Note that, this is another signed agreement from the proxy to the edge, which is different from the one the proxy received from the user, though the two agreements may have the same amount of tokens. In practice, the proxy may charge a small amount of transaction fee to cover its operational cost in providing service matching and payment channel intermediate service. However, the transaction fee should be written in an open source program that is agreed by both parties.

After a series of payment, the users, the proxy or the edges may choose to withdraw the tokens by closing the payment channel, which will introduce a gas fee overhead, since it is an on-chain operation. However, all payment channel based off-chain transactions, as depicted in the loop of Fig. 2, are fast data exchange without any cost.

A debatable issue for our design is that we introduced a centralized proxy which handles payments among users and edges, which violates the decentralization spirit of the blockchain. In fact, a simple trick on software engineering can minimize the impact of this concern: the proxy is a completely open source and the proxy code will be hashed and recorded in the blockchain. Any third party can audit the proxy by comparing the hash value of the running system to the blockchain recorded data, thus, maintain the transparency of the system.

V. Test-bed Implementation

A. Software Architecture

Fig. 3 illustrates the software architecture for EdgeToll.

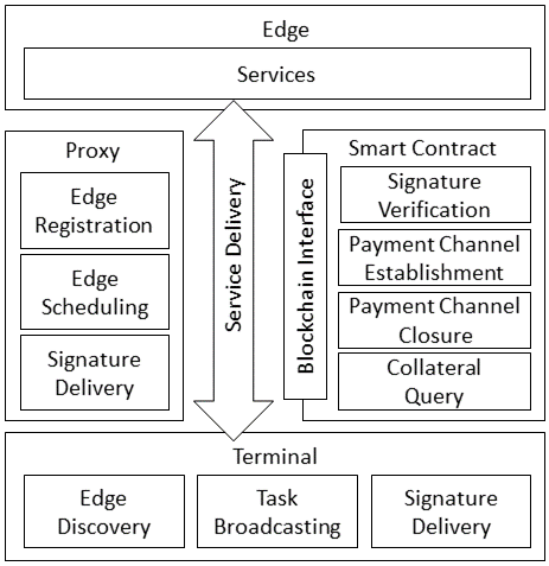

Fig. 3. Software Architecture for EdgeToll System

The *Smart Contract* hosted by Ethereum blockchain provides four major functions to facilitate the payment channel: 1) Payment Channel Establishment: the component for creating a new payment channel. To call this function, the establisher should provide the Ethereum address of channel receiver and sign a transaction to transfer deposit tokens to the smart contract account. The deposit will be locked through the lifetime of the payment channel. 2) Signature Verification: The sender address, receiver address, the value transferred and other signed signature properties should be provided. Keccak256 hash function will hash sender address, receiver address, transferred value together and resulted value will be used to recover the signer of agreement for verification. The return result indicates verification legality in boolean type. 3)

Payment Channel Closure: This module is used to withdraw tokens from the contract account. To verify the qualification of the signature holder, the provided parameters should be the same as Signature Verification. When the invocation happens, the contract will balance the tokens among channel launchers and receivers. Because of the existence of service charge (gas fee), the actual amount receiver obtains is not exactly the same as signature classified. 4) Collateral Query: This module is used to query the balance of existing channels. This qualification-free operation is designed to relieve the distrust of channel receiver. With provided parameters channel launcher address and receiver address, unique channel collateral can be return. If the query result is 0, it implies that the channel between two address has been closed or never exist.

The *Proxy* consists of 3 components: 1) Edge Registration: This component is used for edge devices to register in the database and Ethereum address should be provided. In the real world, the *Proxy* instance will evaluate edges for pricing. But to simplify our model, we fix the deposit amount as 1 ether. The database in *Proxy* will keep a record of the configuration parameters of edges and update dynamical information like location, working status, and memory usage, etc. 2) Edge Scheduling: This component can solve the decision-making problems. Users should provide available edges list attached with task description, which is the result of *Edge Discovery* component in user end. In a future design, the call to this function can match user and edge based on the analysis of data stream from *Terminal* and the dynamical status of available edges, which will provide the user a better resource scheduling. More advanced in further, a task can be decomposed into multiple steps and distributed to heterogeneous edges to improve working efficiency. Our edge selection is determined by the simulated price. 3) Signature Delivery: Users should build payment channel first before the delivery of signed agreement. A verification process will be conducted after the calling from the user. After *Signature Verification* in *Smart Contract* is done, the *Proxy* will call this function again and sign a equal-value signature to edges, which works as a signature delivery intermediate station. If the user set the withdraw pole in request parameters as True, the proxy will close the payment channel immediately.

The *Edge* is the service provider, which may be a container of various functions, such as video surveillance and face recognition. On the other hand, the *Terminal* should contain 3 components: 1) Edge Discovery: the component for discovering nearby edge services. The crucial parameters of edges and description of tasks will be sent to the interface of *Edge Scheduling*. The description of task detail is omitted and all searchable edges have been registered in the proposed simplified model. 2) Task Broadcasting: This component is used to conduct task offloading and type conversion. After the uploading, raw data will be transformed into proper type and then sent to edges for processing. In our work, the input data is an image contains a person. 3) Signature Delivery: The hash value of sender address, a receiver address and amount of tokens will be signed by the private key of the sender. The result of sender address, receiver address, transferred value and three attribute value of signed transaction form a legal signature. The agreement will be delivered to receiver server off-chain.

B. Enabling Software Packages

To facilitate the development process, we adopt a series of cutting-edge software to implement constructing components for the system. We select Ethereum[2] as our blockchain platform, due to its popularity and maturity in technical and commercial community. In our implementation, we utilize Truffle Suite[3] to simulate a private blockchain environment for software development and the Rinkeby Testnet[4] to conduct empirical experiments. Ethereum offers Solidity[5], a Turing-complete programming language for smart contract development. With solidity, we implement an uni-directional payment channel to support the transactions among users, edges and, proxy. The smart contract will be invoked by web3.py[6] library, which is a python[7] interface for interacting with the Ethereum blockchain and ecosystem. The reason for choosing web3.py rather than the web3.js framework is that our user client program and proxy server are implemented with Python. To integrate our EdgeToll system to an edge-terminal environment, we leverage the edge platform from Jiangxing Intelligence Inc.[8], an edge computing start-up located in Shenzhen, China. Each Jiangxing edge node provides a Wi-Fi signal as the portal to access its AI applications, including real-time face recognition and positioning. To facilitate dynamical edge service discovery, we adopt pywifi[9], a python library to search available edge services. The list of available edge access points will be updated to the proxy in real-time. After connecting to the edge, the client will initialize a TCP/IP request through the Application Programming Interface (API) offered by Jiangxing edges to submit the user's image in base64 format, and the edge will return the location of the face in the image in a JSON file[10].

VI. Experiments and Analysis

In this section, we validate the design and implementation of the proposed EdgeToll system with preliminary experiments.

A. Test-bed Specifications

Jiangxing edges used in our test-bed are Acorn RISC Machine architecture (ARM) computers with 8 GB RAM and Intel i5-7300 CPU. The edge is also equipped with a TP-LINK WDR5620 wireless access point, which adopts IEEE 802.11g/b standard with 1200 Mbps data rate and 2.4/5 GHz radio frequency.

[2]https://www.ethereum.org/
[3]https://truffleframework.com/
[4]https://www.rinkeby.io/
[5]https://github.com/ethereum/solidity
[6]https://github.com/ethereum/web3.py
[7]https://www.python.org/
[8]http://www.jiangxingai.com/
[9]https://github.com/awkman/pywifi
[10]https://www.json.org/

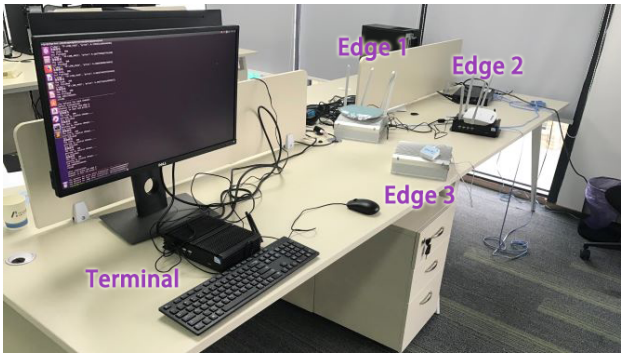

Fig. 4. Demonstration of the Implemented EdgeToll Test-bed

Fig. 4 illustrates a running demonstration of our proposed system, which consists of three edge nodes and one terminal for the end user. All experiments described in following sections are conducted over the test-bed.

B. Experiment Design

Because there are transaction latency and gas fee in Ethereum blockchain, overall service time and the monetary cost should be measured in our experiments. In addition, due to the resource competition among multiple users or multiple edges, the impact of the service requests frequency should be an important factor to be considered as well. Therefore, we design the following tw0 experiments from different perspectives.

- **Benefit of Payment Channel:** the experiment compares the time and cost efficiency with and without the utilization of payment channel technology. Our hypothesis is that with more transactions posted, payment channel will save more time and monetary cost, due to its off chain nature.
- **Cost Minimization:** this experiment is designed to discuss the monetary considerations from the perspective of users. The end users can minimize their costs if there are competitions among multiple edge service providers, given the price for a service unit is dynamic, similar to the spot instance pricing[11] available in cloud computing.

C. Experimental Settings

Here we present the default parameter settings for our following experiments. The default block rate in Rinkeby, approximately one block per 15 seconds, is adopted if no specific settings are imposed. The mobile terminal is a single board computer with Ubuntu 16.04 Linux operating system. By default, we iterate the numbers of users' tasks from 1 to 50 with a step of 5. With proposed system, we assume the users will not close the channel until they complete all of their tasks. Each set of experiments has been repeated for 100 times and their average values were derived as our final results.

[11]https://aws.amazon.com/ec2/spot/pricing/

D. Result Analysis

Fig. 5 and Fig. 6 illustrate the performance comparisons between the system with and without the support of the payment channel. We set up different parameters as depicted in the legends, where *PC* represents using the payment channel, while *WPC* represents the system without using a payment channel. The value of 5s, 10s, and 15s represent the different block intervals used in Fig. 5, while 1 Gwei, 4 Gwei, and 7 Gwei in Fig. 6 represent different gas fee required for posting one transaction to the blockchain.

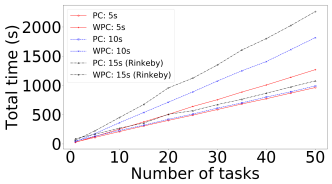

Fig. 5. Total Complete Time

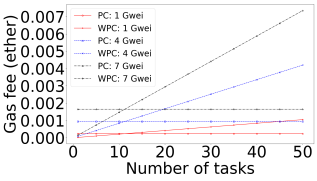

Fig. 6. Transaction Gas Fee

In Fig. 5, we study the impact of users' task numbers on the total completion time, which consists of edge service time and the blockchain transaction delay for the toll payment when applicable. From the results, it is obvious that the total time cost increases linearly as the growth of numbers of users' task. When the total number of users' tasks is small, e.g. 1 task, the total time cost of EdgeToll may be no different to that of systems without payment channel. However, the total waiting time values for conventional approaches, who directly pay tokens through blockchain transactions, are increased at a much higher speed, especially when the block interval is relatively high. For example, the difference of total service time cost between two schemes is 1185 seconds when the block time is set 15 seconds, a common Rinkeby scenario, which means the payment channel can reduce the time cost by more than 110% from *PC*. In fact, the largest overall latency reduction in different block time can be up to 31.8%, 39.6%, 110% in the ratio of *PC*, respectively. Note that, the reduction in Rinkeby is considerably more significant than other shcemes because of the existence of congestion in real world test P2P network, which may indicates the more distinguished performance in authentic production environment.

A similar phenomenon can be observed in Fig. 6, which shows the difference of gas fee the system needs to consume between the two paradigms. One significant feature is that, the gas fee for payment channel based experiments is a constant, no matter how many tasks are posted by the users. This is because all payments for their tasks are sent through the channel, which is a no-cost off-chain process. In fact, the only gas fees they need to pay are the opening and closing transactions in the beginning and the end of their service usage. Things are completely different without the help of payment channel: the total gas fee may increase as the total number of tasks increase. And if the price for the gas increase, the slope will become larger as well. When the number of user' s task is relatively small, e.g. less than 10, the operational cost of WPC

is cheaper than PC. But the condition become different as the increasement of task number.For instance, when a user has a heavy workload e.g. 50 tasks and high gas price (gas price = 7 Gwei), the total gas fee without payment channel will be 0.0057 ether nearly 345% larger than that of the proposed EdgeToll system. In fact, even in the low gas price, e.g., gas price = 1 Gwei, the gas fee without payment channel can also be 4.4 times the value of proposed system.

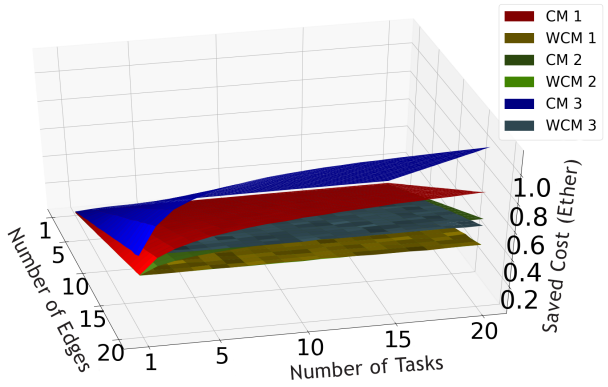

Fig. 7. The impact of task number and number of edges on the saved cost

Fig. 7 shows the result of user cost minimization. In the figure, *CM* represents the cases using user cost minimization algorithms in the proxy, while *WCM* represents the scenarios that the end users randomly select one nearby edge to post their requests. Regarding the dynamic pricing data proposed by the edges in our experiment, we are not able to find corresponding data set for a trace-drive simulation. However, we believe the spot instance price from Amazon Web Service (AWS) can be a reference for us, since they are intrinsically the same mechanism: unit prices are subject to the available resources can be provided. Therefore, we choose a number of random functions to generate dynamic prices for edge nodes, while the mean values of the normal random distribution function can be attained from observing the mean of price history in amazon web service, Linux/Unix d2.xlarge products. In Fig. 7, different schemes are corresponding to different random function. The numeric value *1* indicates a normal distribution with mean = 0.207 and standard deviation = 0.01, the value of *2* means another normal distribution with mean = 0.207 and standard deviation = 0.005, and the value *3* implies a uniform distribution with interval = [0.17, 0.23]. As shown in the figure, the system can bring remarkable benefits to the user. When the price is relatively stable, for example, when the standard deviation is 0.005, the improvement of the system is relatively insignificant. However, when the price vacillates in a uniform random function, the overall saved cost for scenarios with number or tasks = 20 and number of edges = 20 is around 1.14 ether, nearly 93.7% reduction in comparison to a traditional system with 0.59 ether.

VII. Conclusion

An trustworthy and efficient toll collection system is the key to motivate the heterogeneous edge platforms to share their vacant resource from a commercial point of view. In this work, we design and implement EdgeToll, a blockchain-based system, to fill the blank in this area. By leveraging the payment channel technique, we provide a low-latency and cost-efficient solution for a decentralized, transparent and auditable toll collection. We believe that the deployment of EdgeToll will contribute to the public popularization of edges, which can reduce the computational pressure in cloud service and accelerate the future of the Internet of Things (IoT).

Acknowledgment

This work was supported by Nature Key Research and Development Program of China (2017YFB1400700), Shenzhen Fundamental Research Fund under grants No. KQTD2015033114415450 and No. ZDSYS201707251409055, grant No. 2017ZT07X152, the Natural Sciences and Engineering Research Council of Canada (NSERC), and the National Natural Science Foundation of China (61602537, U1509214).

References

[1] P. Banerjee, R. Friedrich, C. Bash, P. Goldsack, B. Huberman, J. Manley, C. Patel, P. Ranganathan, and A. Veitch, "Everything as a service: Powering the new information economy," *Computer*, vol. 44, pp. 36–43, March 2011.

[2] W. Shi, J. Cao, Q. Zhang, Y. Li, and L. Xu, "Edge computing: Vision and challenges," *IEEE Internet of Things Journal*, vol. 3, pp. 637–646, Oct 2016.

[3] Y. Yu, "Mobile edge computing towards 5g: Vision, recent progress, and open challenges," *China Communications*, vol. 13, pp. 89–99, N 2016.

[4] M. Nofer, P. Gomber, O. Hinz, and D. Schiereck, "Blockchain," *Business & Information Systems Engineering*, vol. 59, pp. 183–187, Jun 2017.

[5] N. Álvarez Díaz, J. Herrera-Joancomartí, and P. Caballero-Gil, "Smart contracts based on blockchain for logistics management," in *Proceedings of the 1st International Conference on Internet of Things and Machine Learning*, IML '17, (New York, NY, USA), pp. 73:1–73:8, ACM, 2017.

[6] A. Back, "Hashcash - a denial of service counter-measure," 09 2002.

[7] M. Satyanarayanan, P. Bahl, R. Caceres, and N. Davies, "The case for vm-based cloudlets in mobile computing," *IEEE Pervasive Computing*, vol. 8, pp. 14–23, Oct 2009.

[8] B.-G. Chun, S. Ihm, P. Maniatis, M. Naik, and A. Patti, "Clonecloud: Elastic execution between mobile device and cloud," in *Proceedings of the Sixth Conference on Computer Systems*, EuroSys '11, (New York, NY, USA), pp. 301–314, ACM, 2011.

[9] S. Kosta, A. Aucinas, P. Hui, R. Mortier, and X. Zhang, "Thinkair: Dynamic resource allocation and parallel execution in the cloud for mobile code offloading," in *2012 Proceedings IEEE INFOCOM*, pp. 945–953, March 2012.

[10] S. Nakamoto, "Bitcoin: A peer-to-peer electronic cash system," *White Paper: https://bitcoin.org/bitcoin.pdf*, 2008.

[11] V. Buterin, "Ethereum white paper: a next generation smart contract & decentralized application platform," *https://github.com/ethereum/wiki/wiki/White-Paper*, 2013.

[12] W. Cai, Z. Wang, J. B. Ernst, Z. Hong, C. Feng, and V. C. M. Leung, "Decentralized applications: The blockchain-empowered software system," *IEEE Access*, vol. 6, pp. 53019–53033, 2018.

[13] J. Poon and T. Dryja, "The bitcoin lightning network: Scalable offchain instant payments," in *Whitepaper*, 2016.

[14] C. Decker and R. Wattenhofer, "A Fast and Scalable Payment Network with Bitcoin Duplex Micropayment Channels," in *17th International Symposium on Stabilization, Safety, and Security of Distributed Systems (SSS), Edmonton, Canada*, August 2015.