Using DEVS for Modeling and Simulating a Fog Computing Environment

Mohammad Etemad, Mohammad Aazam and Marc St-Hilaire
Department of Systems and Computer Engineering
Carleton University, Ottawa, Canada
mohammadetemad@sce.carleton.ca, aazam@ieee.org, marc_st_hilaire@carleton.ca

Abstract— **With the increase in popularity of Internet of Things (IoT), pervasive computing, healthcare services, sensor networks, and mobile devices, a lot of data is being generated at the perception layer. Cloud is the most viable solution for data storage, processing, and management. Cloud also helps in the creation of further services, refined according to the context and requirement. However, being reachable through the Internet, cloud is not efficient enough for latency sensitive multimedia services and other time-sensitive services, like emergency and healthcare. Fog, an extended cloud lying within the proximity of underlying nodes, can mitigate the issues traditional cloud cannot solve being standalone. Fog can provide quick response to the requiring applications. Moreover, it can preprocess and filter data according to the requirements. Trimmed data is then sent to the cloud for further analysis and enhanced service provisioning. However, how much better is it to have a fog in any particular scenario instead of a standalone cloud working without fog is a question right now. In this paper, we provide an answer by analyzing both cloud-only and cloud-fog scenarios in the context of processing delay and power consumption according to increasing number of users, on the basis of varying server load. The simulation is done through Discrete Event System Specification (DEVS). Simulation results demonstrate that by the use of fog networks, users experienced lower waiting times and increased data rates.**

Keywords—Fog computing; cloud computing; performance evaluation; Simulation; DEVS

I. INTRODUCTION

Clouds have been used to provide a variety of online services. Cloud is a paradigm that provides infrastructure and other resources, such as memory, storage, and processing, to the users on-demand. Such services alongside better performance, accessibility, and scalability have made the use of cloud ever more popular. Given the increasing number of users on the cloud, heterogeneity in the services offered, advent of Internet of Things (IoT), and the evolution of Big Data, the conventional cloud is no longer sufficient as a standalone strategy, specially when it comes to delay-sensitive services. The limitations of the cloud such as requirement of high bandwidth, reliable connectivity, and sometimes multihoming has led to the development of a new concept, the fog [1-5].

Cisco estimates that by 2020, there will be 50 billion connected devices. This means that using the current methods and available concepts, networks might fail to handle the massive data demand with a guaranteed QoS. To be able to cope with future demand, low power, small cell networks are necessary. This will not only reduce path loss, but will result in a more efficient use of the spectrum. This issue has lead Cisco to propose a new concept for computing called "Fog Computing". Fog is a localized cloud with relatively limited processing power which allows for reliable, low-latency, data processing in the proximity of the user. Cisco suggests the use of fog networks in three scenarios [6]: 1) Data is collected from the edge (for example: vehicles, ships, sensors, and roadways); 2) A very large number of devices are in the network sending data; 3) Data processing and decision making should happen in less than a second.

The fog infrastructure is proposed to be used in association with a cloud. This collaboration enables a fog to forward data to the cloud for processing in case of an overload and handle time-sensitive data on the spot. Doing so offloads tasks running on the underlying nodes, reduces network traffic at the core, reduces latency, and improves reliability. This not only avoids the need for costly bandwidth, but also protects sensitive data by processing it within the local network. Cisco proposes the following architecture to be used with fog [6]:

1- Most time-sensitive data is processed and analyzed by the fog allowing a decision to be made within less than a second.

2- Data that is less time-sensitive and can be processed in seconds or minutes, is forwarded to an aggregation node for processing and decision making.

3- Even less time-sensitive data is sent to the cloud for analysis and storage.

There are multiple scenarios in which the use of fog computing can be beneficial. Some of these scenarios are, but not limited to: smart grids, vehicular networks, smart buildings, and wireless sensors. Taking smart grids for an example, fog networks can process some data at the edge and filter the data that needs to be sent out to the cloud for further processing.

Keeping in mind that the fog and the cloud serve different purposes, it can be implied that the concept of fog is not a replacement for the cloud, but really a complement to it. In this paper, we implement a simulation based on Discrete Event System Specification (DEVS) in order to evaluate how the cloud and the fog can work together to enhance user experience. Different metrics such as the number of users, average occupancy, waiting time, data rates, and power consumption are evaluated. In the following sections, we first talk about the architecture of fog followed by the related work.

Then, Sections IV and V describe the simulation setup and the associated results. Finally, Section VI concludes the paper.

II. Fog Architecture

Fog computing is known to be a system level architecture which builds on the basic capabilities of the cloud. The main element of this type of architecture is called the fog node. These nodes are connected within a network using different connection media: wired and wireless. The architecture of the nodes can be further investigated by considering two categories: hardware architecture and software architecture. While the connection media may be different, the connections rely on the IPv6 protocol because of its wide range of addresses. It should be pointed out that Ethernet is also an option for the connection. Being a system-level architecture, fog computing aims to optimize the distribution of computational and storage capabilities amongst the hierarchy levels of the network by supplementing the cloud. Figure 1 describes a sample architecture for fog layers which are located between the cloud and the smart objects [3].

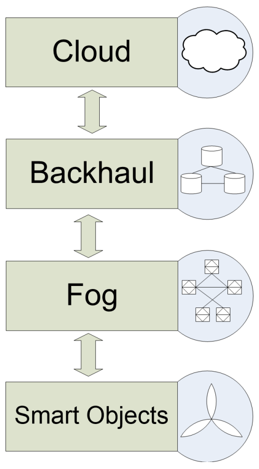

Figure 1. Architecture of the fog [3]

Fog computing is a highly virtualized platform enabling the user to store and analyze data [7]. Cisco recommends the use of fog in collaboration with IoT. It outlines a number of advantages of integrating fog into IoT services. “Expanded Portfolio with Unified Infrastructure, Data Management from the Cloud to the Fog, Redundancy and Failover, Increased Agility and Innovation, Improved Security, Deeper Insights with Improved Data Privacy, and Lower Operating Expenses” are all benefits of using IoT applications on the fog [7].

III. Related Work

The area of fog computing is relatively new. Different researchers have explored different uses of the fog architecture in different fields. Hong et al. [8] propose the concept of mobile fog and outline two main benefits for their architecture: 1) high-level programming model 2) application scaling based on demand. They assume that the fog consists of a programming platform through which one can manage computing instances similar to Infrastructure as a Service (IaaS). They conclude that mobile fog can use the instances created in the fog to execute code. Wang et al. present a concept called the Mobile Cloud Computing (MCC). They define MCC as an infrastructure where data storage and processing happens outside the mobile device. They claim moving the mobile applications to the cloud can make them available not only to smartphone users, but all mobile users [9]. By the use of Mobile Edge Computing (MEC), content providers and application developers can take advantage of the cloud computing characteristics while focusing on the edge of the network. This will allow them to take advantage of very low latency and high bandwidth for their applications. Due to limited resources for processing data on the fog network, having the fog assisting the cloud can improve the network's performance.

Fog computing has also been used in smart devices. For instance, fog computing can be used to manage smart traffic lights. In this case, image processing techniques can be used to identify emergency vehicles and smart lights can change signals based on the current situation. The neighboring traffic lights serving as fog devices can coordinate with one another to create a wave of green lights for the emergency vehicle [2]. Furthermore, many applications such as smart meters can be implemented at the edge of the network [10]. These devices can switch to other energy sources such as wind and solar based on a variety of factors such as demand, availability, or costs. They can also categorize the received data to either be processed locally or sent out to the cloud for processing [11, 12]. Smart gateways are another domain in which fog computing can be used in association with smart devices. Bonomi et al. [2] investigate the appropriateness of fog as being a platform for different applications such as smart cities and connected vehicles. Aazam and Huh [13] talk about fog computing and smart gateway based communication. They point out that at certain periods of time, the fog can help analyze the data and decide whether to upload or not. This way, the cloud will not be burdened. In such cases, the gateway, referred to as “Smart Gateway”, would be responsible for deciding on what to do with the data based on the feedback received from the application. They suggest the use of such gateways in the fog and present the architecture shown in Figure 2.

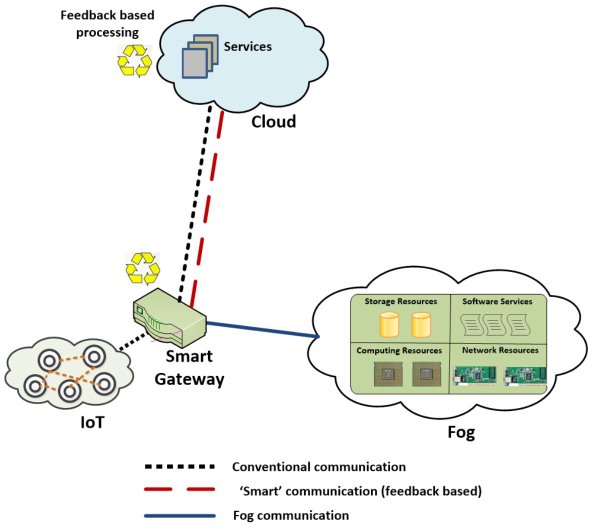

Figure 2. Smart Gateway with Fog computing/Smart network [13]

Zhu et al. [14] propose ways to improve website performance via fog boxes. They assume users are connected to the Internet via edge servers and propose an algorithm that caches commonly requested webpages for future use resulting in the requests being fulfilled faster. The fog server can also determine the browsing speed of the user based on the feedback it receives and can adjust the resolution of graphic files being sent accordingly.

Stojmenovic and Wen [11] focus on the security issues of fog computing. They suggest that the major security threat posed by the use of such networks is authentication at different levels of gateways. They claim that the man-in-the-middle attack can be simple to launch but difficult to address and conclude that fog computing is vulnerable to such type of attacks. Madsen et al. [15] conclude based on their analysis that building fog computing based projects is a challenging task but fault tolerant techniques and specific conditions make these projects possible. Hong et al. [8] discuss the migration costs of fog placements in the network for providers of infrastructures. They define a probabilistic data structure and create a distributed algorithm to create these types of structures. They also propose a migration algorithm which minimizes the network utilization. Finally, they investigate the costs imposed by executing such a plan. Satyanarayanan et al., in [16], propose the concept of cloudlets. Very similar to fog computing, cloudlets are computers with high processing power which are connected to the Internet and are available for use by nearby users. Yi et al. [17] review some challenges faced by offloading data to fog networks. They list three main challenges encountered in doing so. First is the fact that wireless networks are highly dynamic, second, the nodes in the network are dynamic, and third, the resources in the fog network are highly dynamic. This dynamic architecture of the fog allows it to be flexible, hence making it very useful, but also imposes the challenges listed above. Dubey et al. [18] propose, validate, and evaluate a service oriented architecture for fog computing called Fog Data. They characterize their architecture by 3 layers. First is a body sensor network (BSN) for gathering the data, second is a fog gateway which is responsible for on-site data processing, and third is the backend cloud which takes care of storage and heavy processing. They define the Fog Data as: "a generic architecture that allows a wide variety of wearable sensors such as smartwatch, wearable ECG system, and pulse glasses to be used for acquisition of health data" [18]. The paper validates the use of Fog Data for two main healthcare issues which are: speech disorder and ECG. They conclude that the use of Fog Data can reduce requirements for telehealth applications.

To the best of our knowledge, we did not find any papers on simulation of fog networks and how they compare to classic cloud networks. As a result, the next section introduces the simulation that we developed based on DEVS to demonstrate the effects of using fog networks.

IV. Simulation Setup

From the previous sections, it can be concluded that the use of fog can greatly improve user experience. As we did not find any simulators which show how fog can assist the cloud in managing raw data locally and pre-process it before sending it to the cloud, we decided to develop one using DEVS [19]. DEVS consists of a two layer structure which enables the developer to separate the simulator from the model. Furthermore, the model layer can be further categorized into sub-models to further reduce the complexity of each model. This allows the developer to test each model individually in the beginning and as a whole system towards the end of the development process. On the other hand, DEVS is a message based specification meaning that all communication between models is done through customizable messages. This enables DEVS to mimic the behavior of networks in which all events are triggered by messages. All DEVS models are written in C++ and consist of 6 functions: a constructor, initialization function, external function, output function, internal function, and a destructor. The input and output ports that the model will use to communicate and the time required to process a message are defined in the constructor. It should be noted that a model can have multiple input and output ports. The initialization function is used to initialize parameters that the model will use, for example the ID. The external function is called whenever a new message arrives. The programmer uses this function to extract any information required from the received message through the input ports as the other functions are not able to access the received message. The output function is used for sending the desired information on any of the defined output ports in the constructor. The internal function is used to define internal state transitions for the model. Finally, the destructor is used to release managed and unmanaged resources.

The order of execution of any DEVS model starts with the constructor followed by the initialize functions. The model will then oscillate between the output and internal functions until a message is received. As soon as a message is received by the model, the model will execute its external function. Once the external function execution is done, the model will go back to oscillating between the output and internal functions. The simulation will end once the time specified by the user through the run command is reached. The run command should be initialized through the terminal application in the Linux environment.

The above mentioned features of DEVS makes it a good fit for simulating complex systems such as fogs and clouds. Figure 3 demonstrates a high level architecture of the simulation. To connect the models to one another, the links between the models are defined in a .ma file. The .ma file also contains the name of all models and any number of arguments which can be used later in the initialization function of the models. As shown in Figure 3, different instances of the models (User, Cloud/Fog, and Broker) have been used for this purpose. The following paragraphs describe each model and the number of instances used.

User: Each user consists of two sub-models: a queue and a processor. The queue sub-model is responsible for queuing all incoming messages. Once the processor requests a message through the Req port, the queue will forward the message to the processor. The processor will then process the message and send an output message on its output port to the Broker. The user will not decide which fog/cloud will be responsible for processing its request, but it will specify the type of service it is

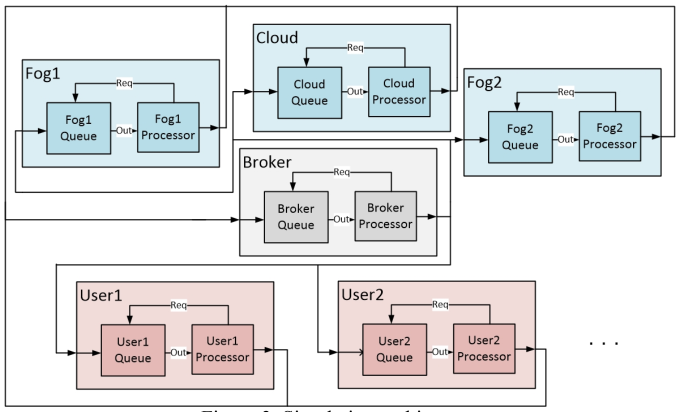

Figure 3. Simulation architecture

requesting in the message. This process will be performed by the Broker.

Broker: This model also consists of two sub-models similar to the User model. The broker (which has only 1 instance) is responsible for gathering the messages from all users and clouds/fogs and broadcasting it to all models just like a switch. Furthermore, this model has a map-key data structure which allows it to keep track of the service each fog/cloud offers. Once a message is received from the user, the broker will decide on where the message should be processed. Once the decision is made, the broker will set a new destination for the request and broadcasts the message over the network. On the other hand, if a message is received from the fog/cloud, the broker will simply change the destination to the user whom the message is intended to and broadcasts the message over the network. This is known as all incoming messages from the fog/cloud contain the user ID. The IDs being unique ensures the correct delivery of every message.

Fog/Cloud: This model also consists of two sub-models similar to the User model. There are 3 instances of this model used in our simulations, two of which act as fogs and one acts as a cloud. In the simulation, the difference between the fog and the cloud is the CPU capacity and the service they provide. The CPU capacity of the cloud has been set to be double the CPU capacity of the fogs. For simplicity but without loss of generality, we only used CPU as the available resource. However, the model could be easily extended to include memory and storage. The fogs are each responsible for one specific service where the cloud processes all requests despite the type of service requested. Finally, if a fog cannot handle a request because of shortage of capacity, the broker will be informed and the request will be sent to the cloud to be processed.

It should be noted that since all messages are broadcasted through the network by all models, the queue of each model will only queue the messages intended for that model and will drop all other messages.

The process which happens for a sample message sent by the user to be processed is as follows:

1- The user generates a request with a specific type of service, CPU requirement, and time required to stay connected to the cloud/fog.

2- This request is sent to the broker by the user.

3- The broker receives the request and extracts the service type.

4- If the service type matches a service handled by a fog, the broker will send the request to that fog. Go to step 6.

5- If the service type does not match a service handled by any of the fogs, the broker sends the request to the cloud.

6- The fog/cloud receives the request, checks its CPU occupancy and decides if it has enough CPU capacity to handle the request. A grant or reject message is sent to the broker based on its decision.

7- The broker receives the decision and forwards it to the user who requested the service.

8- The user receives the decision. If its request is rejected, the user waits for 5 minutes [20]. Go to step 2.

9- If the request of the user is granted, the user will connect to the fog/cloud through the broker.

10- The fog/cloud will track the time each user has been connected. If the requested time is up, the fog/cloud will send a leave message for the user to the broker.

11- The broker receives the message from the fog/cloud and sends it to the user.

12- The user receives the message from the broker and disconnects from the fog/cloud.

It should be noted that to simplify and add randomness to the simulation, the normal distribution has been used to generate values for the request process time, request start time and CPU occupancy. Table 1 shows the simulation parameters used to collect the results.

Finally, we would also like to mention that the DEVS simulation code developed in this paper is made available to the wider community. The source code can be downloaded from: http://www.csit.carleton.ca/~msthilaire/FogDEVS.

V. Results and Discussion

To evaluate the benefits of the fog, two different scenarios were created. In the first scenario, only one cloud is present and all requests will be processed by the cloud. In the second scenario, we still have the cloud from scenario 1 but now two fogs are added to support the cloud in processing requests. For the scenario with the fogs, a linear distribution was used to generate the type of service required by each user. The services generated were set to match the service offered by either fog 1, fog 2, or neither of them forcing the message to go directly to the cloud. In this case, not only the cloud was responsible for handling its own requests, but also helped process requests which were targeted for fog 1 and fog 2 at times when they

Table 1. Initial conditions for the simulations

Parameter	Value
Number of Users	300
Request Process Time (by Fog/Cloud)	Normal Distribution: Min: 60 minutes Max: 180 minutes Mean: 120 Standard deviation: 20 min and max are 3 standard deviations away from the mean value.
Request Start Time (by User)	Normal Distribution: Min: 0 min – representing 12:00 AM Max: 1439 min – representing 11:59 PM Mean: 720 Standard deviation: 240 min and max are 3 standard deviations away from the mean value.
Cloud CPU	32 CPUs at 2.4 GHz for each CPU Total: 76.8 G CPU
Fog CPU	16 CPUs at 2.4 GHz for each CPU Total: 38.4 G CPU
CPU Occupancy Request (by User)	Normal Distribution: Min: 768 M Max: 11.5 G Mean: 6.1 G Standard deviation: 1792 M min and max are 3 standard deviations away from the mean value. (1% to 15% occupancy of the cloud CPU equivalent to 2% to 30% occupancy of each fog)

were busy. To get more accurate results and to minimize anomalies, 10 different simulations were done with different request process time, request start time and CPU occupancy for the users. These values were generated using a normal distribution function in C++ (Table 1 shows the min., max., mean, and standard deviation values for each parameter). Figure 4 shows a side by side comparison of the average occupancy. It can be observed in Figure 4a that when fogs are not present, the load on the cloud increases and is mostly occupied above 90%. However, when the fogs are present (Figure 4b), the cloud is mostly operating below 50% occupancy. Also, it should be noted that in the case were fogs were present, an average of 99.7% of the users were able to fully complete their tasks in a given day, while in the scenario where fogs were not present, an average of 99.5% of the users were able to finish their assigned tasks. Furthermore, the average time for the users to connect to the cloud/fog was 26.5 seconds without the presence of the fogs, whereas with the presence of the fogs, the average time to connect went down to 2.1 seconds. This shows that the use of fogs can noticeably decrease the waiting time before users get connected.

To further assess the performance of the two scenarios, the total consumed power and the average processing delay were also calculated. To see the effect with respect to the number of users, data was collected for 50, 100, 150, 200, 250 and 300 users. As mentioned previously, the fog has been assumed to have half the capacity of the cloud. In this scenario, we set the fog to consume 65% of the energy of the cloud (at full capacity). This hypothetical assumption was made to observe how the use of a fog will impact the energy consumption if a fog provides half the capacity of the cloud while consuming more than half the energy of the cloud at full capacity. In this case, the power consumption for powering up and shutting down services has been assumed to be a transient phase, hence it has been ignored. The results are shown in Figure 5a with the 95% confidence interval. Furthermore, the overall delay of the system was also analyzed as shown in Figure 5b (also with the 95% confidence interval). As can be seen, there is a tradeoff between the power consumption and the average processing delay. In the approach where no fogs are present, the total power consumption outperforms the case in which fogs are available while the presence of the fogs helps reduce the average processing delay of the system.

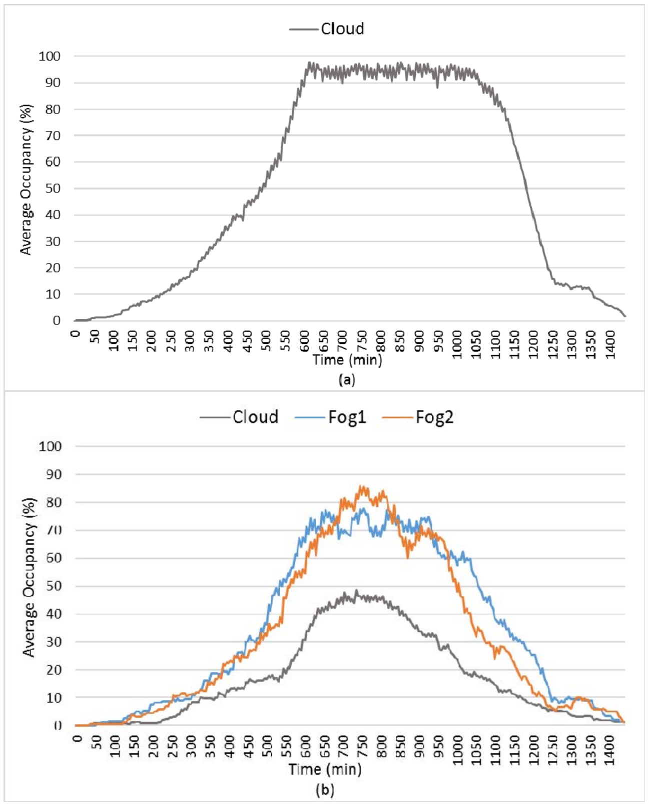

Figure 4. Average occupancy over 10 runs for a) cloud-only scenario vs. b) cloud-fog hybrid scenario.

It can also be concluded from Figure 5 that in the presence of fog, adding users does not severely impact the average processing delay. This is because the load is spread between the fogs and the cloud resulting in minimal effect on the overall system. On the other hand, increasing the number of users in the cloud-only scenario imposes significant delay on the system while keeping the power consumption similar to the scenario where fogs are present. Without the presence of fog, the average processing delay increases exponentially. This is because once the cloud is at 100% capacity, users will not be granted access to connect to the cloud. At this point, users will re-transmit the connection request after a certain period of time. This results in a jump in the number of requests received by the cloud which will further worsen the processing delay.

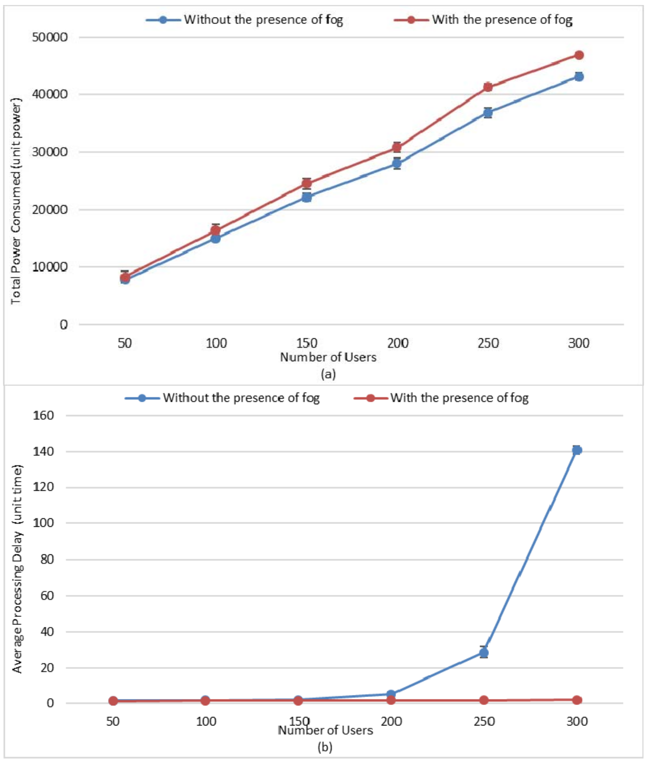

Figure 5. Results for a) total power consumed and b) average processing delay.

VI. Conclusion and Future Work

In this paper, we developed a simulation based on DEVS in order to evaluate the impact of deploying fogs. Based on the preliminary experiments, it can be concluded that the use of fogs combined with the cloud can result in an improved user experience. Doing so will not only help the cloud offload some of its tasks, but will also result in the users experiencing higher data rates. We also showed that the use of fogs can significantly decrease the amount of delay and reduce the number of outdated packets. As future steps, the simulation can be updated to incorporate more factors for the cloud, fog, and users. Additionally, load balancing algorithms can be implemented to minimize the processing time of a request. Another interesting algorithm that may be applied to the current simulation is BitTorrent. By the use of this algorithm, the cloud and fogs can perform parallel processing resulting in potentially better user experience.

References

[1] E. O. Yeboah-Boateng and K. A. Essandoh, "Factors Influencing the Adoption of Cloud Computing by Small and Medium Enterprises in Developing Economies," *International Journal of Emerging Science and Engineering (IJESE),* vol. 2, no. 4, pp. 13-20, 2014.

[2] F. Bonomi, R. Milito, J. Zhu and S. Addepalli, "Fog Computing and Its Role in the Internet of Things," in *First Edition of the MCC Workshop on Mobile Cloud Computing*, Helsinki, 2012.

[3] C. C. Byers and P. Wetterwald, "Fog Computing Distributing Data and Intelligence for Resiliency and Scale Necessary for IoT: The Internet of Things (Ubiquity symposium)," *Ubiquity* , vol. 2015, no. November 2015, pp. 4:1-4:12, 2015.

[4] R. P. Padhy and M. R. Patra, "Managing IT operations in a cloud-driven enterprise: Case studies," *American Journal of Cloud Computing,* vol. 1, no. 1, pp. 1-18, 2013.

[5] K. Kaur and A. K. Rai, "A Comparative Analysis: Grid, Cluster and Cloud Computing," *International Journal of Advanced Research in Computer and Communication Engineering,* vol. 3, no. 3, pp. 5730-5734, 2014.

[6] CISCO, "Fog Computing and the Internet of Things: Extend the Cloud to Where the Things Are," 2015. [Online]. Available: http://www.cisco.com/c/dam/en_us/solutions/trends/iot/docs/computing-overview.pdf. [Accessed 17 January 2016].

[7] CISCO, "Internet of Things (IoT)," CISCO, [Online]. Available: http://www.cisco.com/c/en/us/solutions/internet-of-things/iot-fog-computing.html. [Accessed 27 February 2016].

[8] K. Hong, D. Lillethun, U. Ramachandran, B. Ottenwälder and B. Koldehofe, "Mobile Fog: A Programming Model for Large–Scale Applications on the Internet of Things," in *Second ACM SIGCOMM Workshop on Mobile Cloud Computing*, Hong Kong, 2013.

[9] Y. Wang, I.-R. Chen and D.-C. Wang, "A Survey of Mobile Cloud Computing Applications: Perspectives and Challenges," *Wireless Personal Communications,* vol. 80, no. 4, pp. 1607-1623, 2015.

[10] C. Wei, Z. M. Fadlullah, N. Kato and I. Stojmenovic, "On Optimally Reducing Power Loss in Micro-grids With Power Storage Devices," *Selected Areas in Communications, IEEE Journal on,* vol. 32, no. 7, pp. 1361-1370, 2014.

[11] I. Stojmenovic and S. Wen, "The Fog computing paradigm: Scenarios and security issues," in *2014 Federated Conference on Computer Science and Information Systems (FedCSIS)*, Warsaw, 2014.

[12] M. Aazam and E.-N. Huh, " Fog computing micro datacenter based dynamic resource estimation and pricing model for IoT," in *2015 IEEE 29th International Conference on Advanced Information Networking and Applications.*, Gwangju, 2015.

[13] M. Aazam and E.-N. Huh, "Fog Computing and Smart Gateway Based Communication for Cloud of Things," in *2014 International Conference on Future Internet of Things and Cloud*, Barcelona, 2014.

[14] J. Zhu, D. S. Chan, M. S. Prabhu, P. Natarajan, H. Hu and F. Bonomi, "Improving Web Sites Performance Using Edge Servers in Fog Computing Architecture," in *Service Oriented System Engineering (SOSE), 2013 IEEE 7th International Symposium on*, Redwood, 2013.

[15] H. Madsen, G. Albeanu, B. Burtschy and F. Popentiu-Vladicescu, "Reliability in the utility computing era: Towards reliable Fog computing," in *20th International Conference on Systems, Signals and Image Processing (IWSSIP)*, Bucharest, 2013.

[16] M. Satyanarayanan, P. Bahl, R. Cáceres and N. Davies, "The Case for VM-Based Cloudlets in Mobile Computing," *Pervasive Computing, IEEE,* vol. 8, no. 4, pp. 14-23, 2009.

[17] S. Yi, C. Li and Q. Li, "A Survey of Fog Computing: Concepts, Applications and Issues," in *Mobidata '15 Proceedings of the 2015 Workshop on Mobile Big Data*, Hangzhou, 2015.

[18] H. Dubey, J. Yang, N. Constant, A. M. Amiri, Q. Yang and K. Makodiya, "Fog Data: Enhancing Telehealth Big Data Through Fog Computing," in *ASE BD&SI '15 Proceedings of the ASE BigData & SocialInformatics 2015*, Kaohsiung, 2015.

[19] G. A. Wainer, Discrete-Event Modeling and Simulation: A Practitioner's Approach, CRC Press, 2009.

[20] D. G. A. S. David Hucaby, CCNP Security FIREWALL 642-617 Official Cert Guide, Cisco, 2011.