FogQN: An Analytic Model for Fog/Cloud Computing

Uma Tadakamalla
Computer Science Department
George Mason University
Fairfax, VA, USA
utadakam@gmu.edu

Daniel A. Menascé
Computer Science Department
George Mason University
Fairfax, VA, USA
menasce@gmu.edu

***Abstract*—Several tradeoffs need to be considered when determining the optimal fraction f of data processing executed at the cloud versus at fog servers. The processing capacity of fog servers is typically smaller than that of cloud servers. On the other hand, it may be more expensive to use cloud resources as opposed to fog servers. As f increases, more data has to be sent and received from the cloud. On the other hand, if too much processing is left for the fog servers, they may not have enough capacity to handle requests from sensors and other IoT devices and may become a bottleneck. This paper presents an analytic model and a publicly available tool, called *FogQN*, based on open multi-class Queuing Networks (QN) for fog and cloud computing. *FogQN* was validated with the JMT simulation tool using both distribution-based arrival rates and inputs from real IoT applications.**

***Keywords*-fog computing; cloud computing; queuing theory; queuing networks; IoT applications**

I. Introduction

A key resource management issue when analyzing and optimizing fog computing systems is the determination of the impact of the fraction f of data processing executed at the cloud versus at fog servers. Several tradeoffs need to be considered. The processing capacity of fog servers is typically smaller than that of cloud servers because the latter have ample capacity, which can be increased in an elastic way. On the other hand, it may be more expensive to use cloud resources than fog servers, which are typically owned by the people in charge of the application. As f increases, more data has to be sent and received from the cloud. Given that fog servers communicate with the cloud over the Internet, more roundtrip times and transmission times add up to the cloud processing time. On the other hand, fog servers may not have enough capacity to handle requests from sensors and other IoT devices and may become a bottleneck. Figure 1 illustrates this tradeoff by indicating the variation of the average response time of requests as a function of f when the average arrival rate is 8 requests/sec. The figure indicates that for lower values of f (i.e., 0.2 to 0.4), the average response time decreases as more processing is shifted to the cloud. However, for $f > 0.4$, the response time starts to increase due to excessive use of cloud services and data transmission over the Internet. Thus, there is an optimal value of f, f_{opt}, that minimizes the average response time for a given set of parameters.

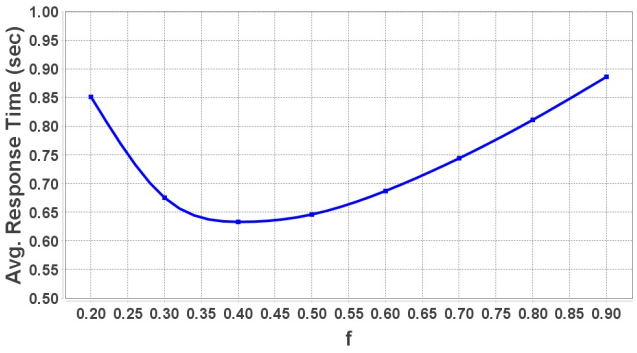

Figure 1. Average response time (in sec) vs. fraction of cloud processing for an arrival rate of 8 request/sec.

The work in [11] discusses the challenges of modeling the Internet of Things. We address some of these challenges here. Specifically, the contributions of this paper are: (1) an analytic model and corresponding tool, called *FogQN*, based on open multi-class Queuing Networks (QN) for fog and cloud computing; *FogQN* is publicly available at [18]. (2) A validation of *FogQN* with the simulation modules of the Java Modeling Tool (JMT) [2] using probability distribution-based inputs and inputs from real applications.

II. Background

Fog computing extends cloud computing capabilities to the edge of the network [14]. Fog resources reside closer to end-user devices and consist of routers, servers, and switches. In this paper, we reference all fog resources holistically as fog servers. Fog servers act as an intermediate layer between cloud datacenters and end-user devices and provide compute, storage, and networking services between these devices and traditional clouds [4].

Unlike cloud computing, which is more centralized, fog computing is more distributed as it uses both fog and cloud resources. Fog servers are generally not as robust as cloud servers; at peak loads, the data that cannot be processed by fog servers is processed by cloud servers. The data that need to be processed by the cloud is sent over a wide

area network (WAN). Therefore, only a fraction of the total data sent and received by end-user devices needs to travel through the WAN as compared with a pure cloud computing paradigm. Additionally, the fog/cloud computing paradigm reduces the cloud processing load when compared with the pure cloud computing model. Thus, fog computing can perform efficiently in terms of service latency, power consumption, network traffic and operational expenses [7], [14] and improve response times. This paradigm is more suitable for applications with low latency requirements such as Internet of Things (IoT) services, including eHealth, autonomous cars, and smart cities. The work in [4] demonstrated the role of fog computing in connected vehicles, smart grids, wireless sensor and actuator networks (WSAN), smart building controls, and software-defined networking (SDN).

Figure 2 illustrates a hierarchical structure of a simple fog computing environment. From the bottom, the first layer is called Edge and consists of devices such as mobile phones, sensors, autonomous cars, and cameras. The second layer, the fog layer, is closer to the edge devices and consists of fog servers. The top layer represents the cloud consisting of cloud servers. The devices in the edge layer send data via WiFi or a Local Area Network (LAN) to the fog servers, which store and process a fraction of the data and send the data that could not be processed locally to the cloud via a WAN, for further processing. The cloud servers process the data and send responses back to the fog servers.

Fog computing requires an effective and efficient resource management of fog and cloud resources to improve quality of service (QoS). This paper proposes an analytical model to analyze the cost and performance of requests submitted to a fog/cloud infrastructure.

III. *FogQN*: A Queuing Model for Fog Computing

FogQN is an analytic model and corresponding tool based on multi-class open Queuing Networks (QN) [16] to analyze the performance and cost of requests processed by a fog/cloud computing environment. This model can be used in the design of an autonomic controller that dynamically determines the best partition between fog and cloud processing.

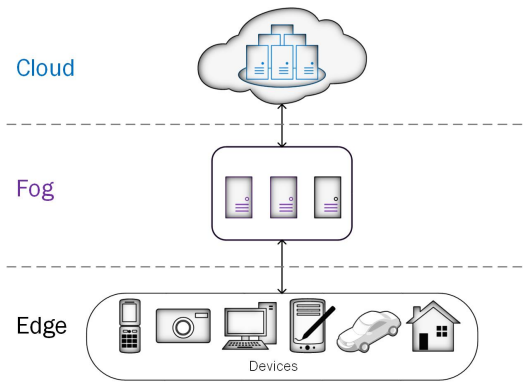

Figure 2. A Fog/Could Environment

A. The QN Model

Multiclass open QN models have been well studied (see e.g., [16]). A QN is a network of K *queues* where a queue consists of a single waiting line and one or more servers. For instance, a queue can represent a server, a network, a processor, or an I/O device. In open QNs, arriving requests visit various queues, some more than once, until they complete processing and leave the system. Requests are grouped into clusters, aka *classes* in the QN literature, of requests that have similar demands on the various queues. The average total service time spent by a class r ($r = 1, \cdots, R$) request at a queue k ($k = 1, \cdots, K$) is called the *service demand* and is denoted by $D_{k,r}$.

The input parameters of an open multiclass QN are (1) the matrix of service demands $\mathbf{D} = [D_{k,r}]$ and (2) the vector $\vec{\lambda} = (\lambda_1, \cdots, \lambda_r, \cdots, \lambda_R)$ where λ_r is the average arrival rate of class r requests.

The average response time R_r of class r requests is simply the sum of all *residence times* $R'_{r,k}$, i.e., the time spent by a class r request at queue k, receiving service or waiting for it, over all visits to the queue. Thus,

$$R_r = \sum_{k=1}^{K} R'_{k,r}. \tag{1}$$

The residence time $R'_{k,r}$ is given by

$$R'_{k,r} = \frac{D_{k,r}}{1 - \sum_{s=1}^{R} \lambda_s D_{k,s}} \tag{2}$$

if queue k is load independent queue, i.e., it has a finite capacity of work and its processing rate does not depend on the queue size. The term $\sum_{s=1}^{R} \lambda_s D_{k,s}$ in Eq. (2) is the utilization U_k of queue k due to all classes and must be less than 1. If queue k has an ample processing capacity (aka delay queue), its residence time is simply the service demand $D_{k,r}$.

Figure 3 shows an open QN that models the Fog/Cloud environment shown in Fig. 2. The QN shows an arrival stream of requests that can be generated for example from IoT devices. These requests go through a local area network (LAN) and reach a fog server (FS), of which there may be more than one, even though only one is shown in the figure for simplicity. Each FS consists of various queues including one or more CPUs and disk queues. Requests may cycle through the devices of a FS and may complete processing after using only FS devices or may require additional processing by the cloud. In the latter case, such cloud requests have to go through the WAN and be processed by the devices (CPU and disks) of a cloud service. For simplicity, only one cloud server is shown in the figure.

Readers familiar with QN theory will recognize that any number of queues can be easily modeled by an open QN (see [16]).

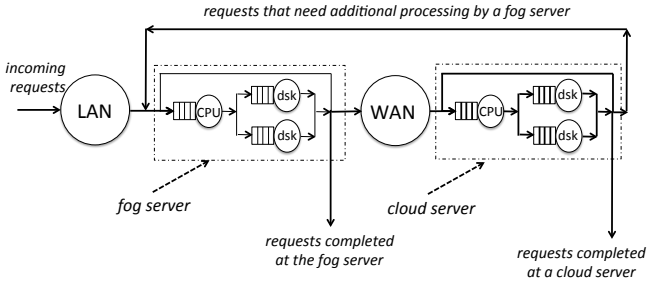

Figure 3. Queuing model for fog computing

For the purpose of this paper and without loss of generality we abstract the fog and cloud servers as single queues each. Thus, the queues considered by *FogQN* are: (a) LAN: represents the time (i.e., latency plus transmission time) spent by a request in the local area network that connects sources of requests (e.g., IoT devices) to a fog server; (b) Fog Server: represents the waiting and processing time at a fog server. Note that a request may have to visit a fog server more than once and wait in line each time; (c) WAN: represents the time (i.e., latency plus transmission time) spent at the Wide Area Network to send requests from a fog server to the cloud and receive replies; and (d) Cloud Server: represents the time spent by a request waiting and being processed by a cloud server; As the figure illustrates, a request may perform several cycles in which it visits a fog server, the WAN, and a cloud server, until it completes. A request may complete at the fog server or at the cloud server.

B. Notation and Parameters

The notation and parameters used in our model are divided into application-level and infrastructure-level parameters.

1) Application-level Parameters:

- R: number of classes of requests
- λ_r: average arrival rate of class r requests (req/sec)
- L_r: amount of data to be processed by class r requests at either a fog server or cloud server (MB)
- f_r: fraction of the data L_r processed at the cloud
- $\vec{f} = (f_1, \cdots, f_R)$
- d_r: average size of a data packet sent from the cloud to the fog server for class r requests (MB)
- $I_r^{\max}$: maximum number of cloud service invocations per class r request.
- I_r: average number of cloud service invocations per class r request. We assume that $I_r = f_r \times I_r^{\max}$.
- γ_r^{fc}: total amount of data transmitted by class r requests from the fog server to the cloud (MB); we assume that $\gamma_r^{fc} = f_r \times L_r$.
- γ_r^{cf}: total amount of data transmitted by class r requests from the cloud to the fog server (MB); we assume that $\gamma_r^{cf} = I_r \times d_r = f_r \times I_r^{\max} \times d_r$.

The following computed application-level parameters are obtained in subsection III-D from the basic parameters listed above and from the parameters listed in subsection III-B2:

- $D_{f,r}$: average service demand of class r requests at a fog server (sec)
- $D_{c,r}$: average service demand of class r requests at a cloud server (sec)
- $D_{\mathrm{LAN},r}$: average service demand of class r requests at the LAN (sec)
- $D_{\mathrm{WAN},r}$: average service demand of class r requests on the WAN (sec)

2) Infrastructure-level Parameters:

- B_{LAN}: bandwidth of the LAN connecting an IOT device and the fog server (MB/sec)
- B_{WAN}: bandwidth of the WAN connection between a fog server and the cloud (MB/sec)
- LAT_{LAN}: round-trip latency of the LAN connecting an edge device to the fog server (sec)
- LAT_{WAN}: round-trip latency of the WAN connection between a fog server and the cloud (sec)
- δ: scaling factor used to multiply any portion of a request's processing time that has to be processed at the cloud
- C_{LAN}: cost of sending data through the LAN connecting edge devices to a fog server ($/MB)
- C_{WAN}: cost of sending data through the WAN connecting a fog server to the cloud ($/MB)
- C_f: Processing cost per unit time at the fog server ($/sec)
- C_c: Processing cost per unit time at the cloud ($/sec)

C. Model Outputs

The outputs of FogQN are denoted as:

- $R'_{\mathrm{LAN},r}$: residence time of class r requests at the LAN (sec)
- $R'_{f,r}$: residence time of class r requests at a fog server (sec)
- $R'_{\mathrm{WAN},r}$: residence time of class r requests at the WAN (sec)
- $R'_{c,r}$: residence time of class r requests at the cloud (sec)
- R_r: average response time of class r requests (sec)
- C_r: average processing cost of class r requests ($)

D. Computing Service Demands

The service demand per class r request at the LAN is the sum of the LAN's latency and the transmission time. Thus,

$$D_{\mathrm{LAN},r} = \frac{L_r}{B_{\mathrm{LAN}}} + LAT_{\mathrm{LAN}}. \tag{3}$$

We assume that the average total processing time of class r requests (sec) at a fog server or at the cloud is equal to the sum of two terms: a constant term K_1 (in sec) and another proportional to the amount of data processed. The proportionality factor is denoted as K_2 (in sec/MB).

The service demand at the fog server per class r requests can be written as

$$D_{f,r} = K_1 + K_2(1 - f_r)L_r. \quad (4)$$

The second term in Eq. (4) is the processing time at a fog server that is proportional to the amount of data $(1 - f_r)L_r$ processed at the fog server.

The service demand at the WAN for class r requests is the sum of the latency $I_r \times LAT_{\text{WAN}}$ and the transmission time between the fog server and the cloud and vice-versa. Each interaction with the cloud incurs in a latency. Hence,

$$\begin{aligned} D_{\text{WAN},r} &= f_r \; I_r^{\max} \; LAT_{\text{WAN}} + \frac{\gamma_r^{fc} + \gamma^{cf}}{B_{\text{WAN}}} \\ &= f_r \left[I_r^{\max} LAT_{\text{WAN}} + \frac{L_r + I_r^{\max} d_r}{B_{\text{WAN}}} \right] \quad (5) \end{aligned}$$

Finally, the service demand for class r requests at the cloud can be written as

$$D_{c,r} = \delta(K_1 + K_2 f_r L_r) \quad (6)$$

because δ is the scaling factor for processing time at the cloud and $f_r L_r$ is the portion of the data processed at the cloud.

E. Response Time Calculation

The residence time equations for the LAN, fog server, WAN, and the cloud are computed based on Eq. (2) and on the service demands computed above.

$$R'_{\text{LAN},r} = D_{\text{LAN},r}; \quad R'_{\text{WAN},r} = D_{\text{WAN},r} \quad (7)$$

$$R'_{f,r} = \frac{D_{f,r}}{1 - \sum_{s=1}^{R} \lambda_s D_{f,s}} \quad (8)$$

$$R'_{c,r} = \frac{D_{c,r}}{1 - \sum_{s=1}^{R} \lambda_s D_{c,s}} \quad (9)$$

Finally, the response time of class r requests is given by the sum of the residence times at the LAN, fog server, WAN, and the cloud. So,

$$R_r = R'_{\text{LAN},r} + R'_{f,r} + R'_{\text{WAN},r} + R'_{c,r}. \quad (10)$$

Because R_r is a function of the service demands of all classes at the fog server and at the cloud (see Eqs. (8) and (9)) and these service demands depend on all values of f_r for $r = 1, \cdots, R$, we will use the notation $R_r(\vec{f})$ to make that dependence explicit.

F. Cost Calculation

The total cost C_r of processing a class r request is the sum of the costs of transmitting data through the LAN and the WAN plus the processing costs at the fog server and the cloud. The transmission cost at the LAN is obtained by multiplying the total amount of data transmitted over the LAN by the cost per MB. Thus, this cost is $L_r \times C_{\text{LAN}}$. The cost of data transmission over the WAN is $(\gamma_r^{fc} + \gamma_r^{cf}) \times C_{\text{WAN}}$ following the same approach as for the LAN. The processing cost at the fog server is obtained by multiplying the residence time at the fog server by the cost per unit time C_f at the fog server. Thus, this cost is $R'_{f,r} \times C_f$. Using a similar reasoning, the processing cost at the cloud is $R'_{c,r} \times C_c$. Therefore, the average cost of class r requests is

$$\begin{aligned} C_r &= L_r \times C_{\text{LAN}} + (\gamma_r^{fc} + \gamma_r^{cf}) \times C_{\text{WAN}} + \\ &\quad R'_{f,r} \times C_f + R'_{c,r} \times C_c \quad (11) \end{aligned}$$

As above, we denote C_r as $C_r(\vec{f})$.

G. Sensitivity to f

This section analyzes the sensitivity of the response time R_r and cost C_r as a function of the proportion of cloud processing f. We consider in this subsection the case of a single class (i.e., $R = 1$) since the purpose here is to analyze the sensitivity of the response time and cost with respect to f. We drop all subscripts relative to class to simplify the notation. We start by considering the partial derivative of R with respect to f.

$$\begin{aligned} \frac{\partial R}{\partial f} &= \frac{\partial R'_{\text{LAN}}}{\partial f} + \frac{\partial R'_f}{\partial f} + \frac{\partial R'_{\text{WAN}}}{\partial f} + \frac{\partial R'_c}{\partial f} \\ &= \frac{\partial D_{\text{LAN}}}{\partial f} + \frac{\partial D_f}{\partial f} \frac{1}{(1 - \lambda D_f)^2} + \\ &\quad \frac{\partial D_{\text{WAN}}}{\partial f} + \frac{\partial D_c}{\partial f} \frac{1}{(1 - \lambda D_c)^2} \quad (12) \end{aligned}$$

The partial derivatives of the service demands are:

$$\begin{aligned} \frac{\partial D_{\text{LAN}}}{\partial f} &= 0; \quad \frac{\partial D_f}{\partial f} = -K_2 L \\ \frac{\partial D_c}{\partial f} &= \delta K_2 L \\ \frac{\partial D_{\text{WAN}}}{\partial f} &= LAT_{\text{WAN}} \; I^{\max} + \frac{L + I^{\max} \; d}{B_{\text{WAN}}} \quad (13) \end{aligned}$$

As the equations in (13) show, the service demand at the LAN does not vary with f, the service demand at the WAN and at the cloud grow with f and the service demand at the fog server decreases with f. So, the bottleneck (i.e., the queue with the largest service demand [16]) changes with f. Therefore, there is an optimal value of f that minimizes the response time as illustrated in Fig. 1. Using Eqs. (13)

in (12) we get

$$\frac{\partial R}{\partial f} = \frac{-K_2 L}{(1-\lambda D_f)^2} + \frac{\delta K_2 L}{(1-\lambda D_c)^2} + LAT_{\text{WAN}}\ I^{\max} + \frac{L + I^{\max}\, d}{B_{\text{WAN}}}. \quad (14)$$

Equation (14) shows that, except for its first term, all its terms are positive . So, the derivative of the response time can start as negative, reach zero, and become positive as f increases. This explains the behavior seen in Fig. 1.

The partial derivative of C with respect to f is

$$\begin{aligned}\frac{\partial C}{\partial f} &= \frac{\partial[(\gamma^{fc}+\gamma^{cf})C_{\text{WAN}}]}{\partial f} + C_f \frac{\partial D_f}{\partial f}\frac{1}{(1-\lambda D_f)^2} + C_c \frac{\partial D_c}{\partial f}\frac{1}{(1-\lambda D_c)^2} \\ &= C_{\text{WAN}}(I^{\max} d + L) + \frac{-K_2 L C_f}{(1-\lambda D_f)^2} + \frac{C_c \delta K_2 L}{(1-\lambda D_c)^2}. \end{aligned} \quad (15)$$

The same observations we made with respect to $\partial R/\partial f$ can be made with respect $\partial C/\partial f$. Thus, depending on the values of the parameters, there is an optimal value of f for the response time and cost. The optimal value may not be the same for each metric.

IV. Validation

We validated the equations of the *FogQN* model presented in the previous section with two types of simulations using JSIM*wiz* and JSIM*graph* [2] from Java Modeling Tools (JMT) [2]. In the first type of validation we used Poisson arrivals and in the other we used interarrival time distributions obtained from traces of a smart city application.

A. Validation with Poisson Arrivals

The parameters used for the validation of a two-class model are shown in Table I and the validation results are shown in Table II. The arrival process is assumed to be Poisson, i.e., the interarrival time of requests is assumed to be exponentially distributed. Note from the top of Table II that we kept λ_2 constant at 3.0 req/sec through the experiments reported in that table and increased λ_1 from 1.0 req/sec to 3.0 req/sec. The percent absolute error, ϵ, between the analytical and simulation models is computed as $\text{abs}(100 \times (simulation - analytical)/simulation)$. As Table II clearly shows, the absolute percent error is very small in most cases and ranges from 1.3% to 11.9% but stays below 10% in the vast majority of cases. The absolute average errors are approximately 10.0% and 2.8% for classes 1 and 2, respectively. The overall average absolute error is approximately 6.4%, which is very low. Therefore, we consider the analytic expressions used by *FogQN* as validated using Poisson arrivals.

Table I
Model Parameters Utilized and Computed Service Demands

[A] Application-level Parameters		
R	2	
$\vec{\lambda}$	[varies, 3 req/sec]	req/sec
$\vec{L}$	[4.0, 7.0]	MB/req
$\vec{f}$	[0.30, 0.40]	
$\vec{d}$	[0.05120, 0.1024]	MB/req
$\vec{I}^{max}$	[1, 1]	
K_1	0.015	sec/req
K_2	0.032	sec/MB
[B] Infrastructure-level Parameters		
B_{LAN}	1250	MB/sec
B_{WAN}	5.00	MB/sec
LAT_{LAN}	0.0008	sec/req
LAT_{WAN}	0.04	sec/req
δ	0.4	
[C] Computed values		
$\vec{I}_r$	$= f_r \times I_r^{max}$ = [0.30, 0.40]	
$\vec{\gamma}^{cf}$	$= I_r \times d_r$ = [0.01536, 0.04096]	MB/req
$\vec{\gamma}^{fc}$	$= f_r \times L_r$ = [1.200, 2.800]	MB/req
[D] Computed Service Demands		
$\vec{D}_{LAN}$	[0.0040, 0.0064]	sec/req
$\vec{D}_f$	[0.1046, 0.1494]	sec/req
$\vec{D}_{WAN}$	[0.2551, 0.5842]	sec/req
$\vec{D}_c$	[0.0214, 0.0418]	sec/req

Table II
Mean and 95% confidence intervals of average response times using *FogQN* and simulation (JMT)

λ_1= varies; λ_2= 3.0 req/sec						
	R_1 (sec)			R_2 (sec)		
λ_1	FogQN	Simulation	ϵ(%)	FogQN	Simulation	ϵ(%)
1.0	0.518	0.559 ± 0.017	7.4	0.974	0.955 ± 0.027	2.0
1.2	0.530	0.580 ± 0.012	8.7	0.990	0.971 ± 0.018	2.0
1.4	0.542	0.595 ± 0.013	8.8	1.009	0.996 ± 0.029	1.3
1.6	0.557	0.613 ± 0.017	9.2	1.029	1.005 ± 0.015	2.4
1.8	0.572	0.634 ± 0.011	9.8	1.052	1.009 ± 0.022	4.2
2.0	0.590	0.652 ± 0.017	9.5	1.077	1.054 ± 0.027	2.2
2.2	0.610	0.686 ± 0.020	11.0	1.106	1.087 ± 0.025	1.7
2.4	0.633	0.711 ± 0.021	11.0	1.138	1.103 ± 0.020	3.2
2.6	0.659	0.748 ± 0.020	11.9	1.176	1.134 ± 0.029	3.7
2.8	0.689	0.781 ± 0.022	11.7	1.219	1.183 ± 0.032	3.0
3.0	0.725	0.819 ± 0.021	11.5	1.270	1.207 ± 0.034	5.2
Avg			10.0			2.8

B. Validation with Real Traces

We wanted to explore the robustness of *FogQN* for interrarrival time distributions obtained from real IoT applications. The City of Seattle Open Data portal releases several datasets that are generated by IoT devices in Seattle. We used the "NW 58th St Greenway at 22nd Ave NW Bike Counter" dataset [8] for the second *FogQN* model validation. The dataset is generated by two IoT devices, called tube sensors, which are installed across the street. These tubes count the number of bicyclists and sends the information to an eco-counter server. The dataset has records that contain the number of bikes that traverse the east and west directions

of the street at every hour, which will be referred as classes 1 and 2, respectively, heretofore.

We first analyzed the dataset by grouping the data by hour of the day for 1,644 days while ignoring invalid data (both nulls and zero counts) as shown in Table III. The table shows for each hour of the day for each direction and for all days the total number #Rec of records, the total number $\Sigma(\mathrm{cnt})$ of bikes counted in the #Rec records. For each hour, we built a randomized list of interarrival times using the bike counts cnt for each hour and computed the average interarrival time IAT for that hour. We then computed the average rate λ as 1/IAT and computed the COV of the interarrival times. The numbers in Table III reflect the arrival of requests at the fog server from one sensor in each direction, i.e., east and west. However, for the purpose of this analysis we considered that the arrival rate is the aggregate of 3,600 similar sensors.

The table shows that the hour of 5:00pm-6:00pm had the maximum bike counts (i.e., the highest load) for both East and West. Therefore, we used that hour for validation purposes. Because the COV of the interarrival times for both directions is greater than 1, we fitted a Coxian distribution to the data in each direction and used these Coxian distributions in the JMT simulation. The other input parameters are the same as those in Table I [A] and Table I [B], except for the vector $\vec{L}$, which is $[1.0, 1.0]$ MB/req and the vector λ which is $[12.192, 9.205]$ req/sec.

The response times from the JMT simulation are compared with those computed using *FogQN* and the results are shown in Table IV. The errors are very small (around 10%) for both classes of requests, which indicates the robustness of the equations derived here under more realistic arrival processes.

Table III
STATISTICS OF THE SEATTLE "NW 58TH ST GREENWAY AT 22ND AVE NW BIKE COUNTER" DATASET BY THE HOUR OF THE DAY

Hour	East					West				
	#Rec	$\Sigma(\mathrm{cnt}_1)$	IAT_1 (hr)	λ_1 (cnt/hr)	COV_1	#Rec	$\Sigma(\mathrm{cnt}_2)$	IAT_2 (hr)	λ_2 (cnt/hr)	COV_2
0:00	580	1,479	0.392	2.550	0.756	527	945	0.558	1.793	0.590
1:00	400	968	0.413	2.420	0.733	415	687	0.604	1.655	0.545
2:00	298	624	0.478	2.094	0.640	337	545	0.618	1.617	0.541
3:00	232	404	0.574	1.741	0.490	274	394	0.695	1.438	0.440
4:00	252	415	0.607	1.647	0.484	297	394	0.754	1.327	0.395
5:00	482	1,080	0.446	2.241	0.639	654	1,044	0.626	1.596	0.514
6:00	852	3,251	0.262	3.816	1.007	1,031	2,589	0.398	2.511	0.713
7:00	1,248	8,601	0.145	6.892	1.269	1,217	4,945	0.246	4.063	0.883
8:00	1,341	10,901	0.123	8.129	1.342	1,354	8,181	0.166	6.042	0.968
9:00	1,411	12,523	0.113	8.875	1.367	1,408	8,903	0.158	6.323	0.974
10:00	1,378	11,758	0.117	8.533	1.399	1,406	9,415	0.149	6.696	1.039
11:00	1,420	13,959	0.102	9.83	1.474	1,438	10,974	0.131	7.631	1.144
12:00	1,456	16,381	0.089	11.251	1.521	1,440	11,870	0.121	8.243	1.172
13:00	1,455	16,518	0.088	11.353	1.526	1,441	11,757	0.123	8.159	1.106
14:00	1,467	16,347	0.090	11.143	1.506	1,436	11,283	0.127	7.857	1.113
15:00	1,478	15,381	0.096	10.407	1.397	1,434	11,116	0.129	7.752	1.049
16:00	1,487	17,124	0.087	11.516	1.382	1,437	11,692	0.123	8.136	0.994
17:00	**1,486**	**18,118**	**0.082**	**12.192**	**1.364**	**1,446**	**13,310**	**0.109**	**9.205**	**1.033**
18:00	1,467	16,544	0.089	11.277	1.424	1,427	12,482	0.114	8.747	1.054
19:00	1,397	12,901	0.108	9.235	1.371	1,372	9,183	0.149	6.693	1.041
20:00	1,307	9,050	0.144	6.924	1.284	1,263	5,658	0.223	4.480	0.937
21:00	1,211	6,796	0.178	5.612	1.220	1,074	3,752	0.286	3.493	0.830
22:00	974	4,113	0.237	4.223	1.040	924	2,401	0.385	2.598	0.779
23:00	841	2,697	0.312	3.207	0.912	744	1,639	0.454	2.203	0.710

Table IV
MEAN AND 95% CI OF RESPONSE TIMES USING *FogQN* AND SIMULATION (JMT) FOR THE SEATTLE "NW 58TH ST GREENWAY AT 22ND AVE NW BIKE COUNTER" DATASET

λ_1 (East) = 12.192 req/sec; COV_1 (East) = 1.364 λ_2 (West) = 9.205 req/sec; COV_2 (West) = 1.033					
R_1 (sec)			R_2 (sec)		
FogQN	Simulation	$\epsilon(\%)$	*FogQN*	Simulation	$\epsilon(\%)$
0.234	0.259 $\pm$ 0.007	9.7	0.268	0.299 $\pm$ 0.007	10.4

V. RELATED WORK

The goal of fog computing is to address cloud computing's problem of high latency and network congestion. This is achieved by extending the cloud computing paradigm to the edge of the network [4], [14], [15]. Bittencourt *et al.* presented a general architecture that supports virtual machine migration in fog computing [3]. The work in [6] presented decentralized design patterns on elasticity in IoT and cloud-based systems. [9] presents a simulation-based toolkit for fog computing.

The authors of [13] proposed resource management strategies at each fog node to improve quality of service (QoS). The authors of [1] presented fog resource management techniques based on user characteristics. An approach to optimize web page performance within a fog computing architecture combined with knowledge available at the fog nodes is discussed in [20].

The authors of [5], [10], [17], [19] used queuing theory models to study service performance in cloud computing. The work in [10] used an M/G/1 queue to model energy consumption and response time tradeoffs for an edge device powered by solar energy that sends messages to cloud services.

VI. CONCLUDING REMARKS AND ONGOING WORK

This paper introduced a multi-class open queuing network model for fog computing environments. The model equations were incorporated into our publicly available *FogQN* tool [18]. Analytic models are orders of magnitude faster than simulation models and therefore can be used by autonomic controllers [12]. We are now working on an autonomic controller that uses *FogQN* to dynamically control how much processing is done at the fog servers vs. at the cloud. The controller optimizes a utility function of the average response time and cost.

ACKNOWLEDGEMENT

The work of Daniel A. Menascé was partially supported by the AFOSR grant FA9550-16-1-0030.

REFERENCES

[1] M. Aazam and E. N. Huh. Dynamic resource provisioning through fog micro datacenter. In *2015 IEEE Intl. Conf. Pervasive Computing and Communication Workshops*, pages 105–110, March 2015.

[2] M. Bertoli, G. Casale, and G. Serazzi. JMT: performance engineering tools for system modeling. *SIGMETRICS Perform. Eval. Rev.*, 36(4):10–15, 2009.

[3] L. F. Bittencourt, M. M. Lopes, I. Petri, and O. F. Rana. Towards virtual machine migration in fog computing. In *0th Intl. Conf. P2P, Parallel, Grid, Cloud and Internet Computing (3PGCIC)*, pages 1–8, Nov 2015.

[4] F. Bonomi, R. Milito, J. Zhu, and S. Addepalli. Fog computing and its role in the internet of things. In *Proc. MCC Wkshp. Mobile Cloud Computing*, MCC '12, pages 13–16, New York, NY, USA, 2012. ACM.

[5] T. Bures, V. Matena, R. Mirandola, L. Pagliari, and C. Trubiani. Performance modelling of smart cyber-physical systems. In *2018 ACM/SPEC Intl. Conf. Performance Engineering*, pages 37–40. ACM, 2018.

[6] V. Cardellini, T. G. Grbac, M. Nardelli, N. Tanković, and H.-L. Truong. QoS-Based elasticity for service chains in distributed edge cloud environments. In *Autonomous Control for a Reliable Internet of Services*, pages 182–211. Springer, 2018.

[7] A. V. Dastjerdi, H. Gupta, R. N. Calheiros, S. K. Ghosh, and R. Buyya. Fog computing: Principles, architectures, and applications. *CoRR*, abs/1601.02752, 2016.

[8] Dataset. https://data.seattle.gov/browse?category= Transportation.

[9] H. Gupta, A. Dastjerdi, S. Ghosh, and R. Buyya. iFogSim: A toolkit for modeling and simulation of resource management techniques in the internet of things. *J. Software Practice and Experience*, 6, 2017.

[10] P. G. Harrison and N. M. Patel. Optimizing energy-performance trade-offs in solar-powered edge devices. In *Proc. 2018 ACM/SPEC Intl. Conf. Performance Engineering*, New York, NY, USA. ACM.

[11] G. Kecskemeti, G. Casale, D. Jha, J. Lyon, and R. Ranjan. Modelling and simulation challenges in internet of things. *IEEE Cloud Computing*, 4:62–69, 2017.

[12] J. O. Kephart and D. M. Chess. The vision of autonomic computing. *Computer*, 36(1):41–50, Jan 2003.

[13] J. Li, C. Natalino, D. P. Van, L. Wosinska, and J. Chen. Resource management in fog-enhanced radio access network to support real-time vehicular services. In *Fog and Edge Computing (ICFEC), IEEE 1st Intl. Conf.*, pages 68–74. IEEE, 2017.

[14] R. Mahmud, R. Kotagiri, and R. Buyya. *Fog Computing: A Taxonomy, Survey and Future Directions*, pages 103–130. Springer, Singapore, 2018.

[15] E. Marin-Tordera, X. Masip, J. Garcia Almiana, A. Jukan, G.-J. Ren, J. Zhu, and J. Farre. What is a fog node a tutorial on current concepts towards a common definition. Nov. 2016.

[16] D. A. Menascé, V. A. F. Almeida, and L. W. Dowdy. *Performance by Design:computer capacity planning by example*. Prentice Hall, 2004.

[17] T. S. Sowjanya, D. Praveen, K. Satish, and A. Rahiman. The queueing theory in cloud computing to reduce the waiting time. *Intl. J. Computer Science Engineering & Technology*, 1(3), 2011.

[18] U. Tadakamalla and D. A. Menascé. FogQN: A tool for modeling fog computing environments. available at https: \\www.cs.gmu.edu\~menasce\fogqn\.

[19] J. Vilaplana, F. Solsona, I. Teixidó, J. Mateo, F. Abella, and J. Rius. A queuing theory model for cloud computing. *J. Supercomputing*, 69(1):492–507, 2014.

[20] 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 *IEEE 7th Intl. Symp. Service-Oriented System Engineering*, March 2013.