Timeliness Evaluation of Intermittent Mobile Connectivity over Pub/Sub Systems

Georgios Bouloukakis[1] Nikolaos Georgantas[1] Ajay Kattepur[2] Valérie Issarny[1]
[1]MiMove Team, Inria Paris, France
firstname.lastname@inria.fr
[2]TCS Research & Innovation, Bangalore, India
ajay.kattepur@tcs.com

ABSTRACT

Systems deployed in mobile environments are typically characterized by intermittent connectivity and asynchronous sending/reception of data. To create effective mobile systems for such environments, it is essential to guarantee acceptable levels of timeliness between sending and receiving mobile users. In order to provide QoS guarantees in different application scenarios and contexts, it is necessary to model the system performance by incorporating the intermittent connectivity. *Queueing Network Models* (QNMs) offer a simple modeling environment, which can be used to represent various application scenarios, and provide accurate analytical solutions for performance metrics, such as system response time. In this paper, we provide an analytical solution regarding the end-to-end response time between users sending and receiving data by modeling the intermittent connectivity of mobile users with QNMs. We utilize the *publish/subscribe* (pub/sub) middleware as the underlying communication infrastructure for mobile users. To represent the user's connections/disconnections, we model and solve analytically an ON/OFF queueing system by applying a mean value approach. Finally, we validate our model using simulations with real-world workload traces. The deviations between the performance results foreseen by the analytical model and the ones provided by the simulator are shown to be less than 5% for a variety of scenarios.

Keywords

Publish/Subscribe Middleware; Mobile Connectivity; Queueing Networks; Response Time

1. INTRODUCTION

Publish/Subscribe (pub/sub) systems constitute an appealing interaction paradigm for building large-scale distributed systems. They provide a loosely coupled form of interaction that is especially useful in the case of mobile peers. Pub/sub is deemed appropriate for interaction between mobile entities and it facilitates the asynchronous and on-demand data information dissemination [1]. A large number of applications have been created and several standards have been adopted using the pub/sub interaction style, such as, web news through RSS feeds [2, 3], streaming services [4], google services [5] and real time applications [6].

To motivate our work, we provide the following illustrative use case: Metro travelers in Paris frequently utilize a transport information management system for the improvement of their travel experience. The system operates based on both authoritative and mobile crowd-sourced information from multiple heterogeneous sources. The information (i.e., events of interest) for the average user arrives with a certain rate through asynchronous pub/sub notifications. To guarantee the freshness of provided information, notifications are maintained by the system for a (limited) lifetime period. The travelers are able to access the system periodically and receive up-to-date transport information on their hand-held devices, but also publish information themselves, e.g., notifying of some incident in the transport network. They stay connected for a certain period and then disconnect, also for resource saving purposes. Moreover, actual connection/disconnection status depends on the network coverage and capacity in the metro. Under these constraints, an application designer should be able to analyze and configure certain system aspects (network and user connectivity, event lifetime period, allocated system resources) in order to guarantee the appropriate response time and delivery success rate between the sources and the mobile users. To investigate such features, it is essential to model the performance of mobile pub/sub systems by considering the above constraints.

Several existing efforts concerning the design and evaluation of mobile systems aim at guaranteeing QoS (Quality of Service) requirements under several constraints (e.g., intermittent availability, limited resources, etc). The evaluation methods used are derived mainly from the field of *Queueing Theory*. For instance, 2-Dimensional (2-D) Markov chains and quasi-birth-death (QBD) processes have been used in [7–11] to model the changing connectivity of mobile users when offloading computation or data to the cloud through either 3G or WiFi connections; the goal is to evaluate the trade-off between performance and energy consumption. However, actually applying these methods remains a complex and tedious task for system designers. On the other hand, regarding pub/sub systems, Queueing Petri Nets (QPNs) have been used in [12, 13] for accurate performance prediction. However, QPNs, while highly expressive in representing parallelism, are suited for small-to-moderate size systems and intake considerable computational resources [14].

ICPE'17, April 22 - 26, 2017, L'Aquila, Italy

ISBN 978-1-4503-4404-3/17/04…$15.00

DOI: http://dx.doi.org/10.1145/3030207.3030220

In this paper, we model the performance of a mobile pub/sub system by relying on *Queueing Network Models* (QNMs) [15, 16]. QNMs have been extensively applied to represent and analyze communication and computer systems and they have proved to be simple and at the same time powerful tools for system designers with regard to system performance evaluation and prediction. Based on QNMs, pub/sub brokers are represented as queues, called *service centers* or *queueing centers*, and the exchanged events as *jobs* served. Our performance evaluation focuses on the tradeoff between the time decoupling provided by the pub/sub paradigm and the required end-to-end timeliness in the delivery of events.

Particularly, queueing networks have simple analytical solutions and can achieve a favorable balance between accuracy and efficiency. They enable the isolation and analysis of each service center from the rest of the network. The solution of the entire network can be formed by combining these separate solutions. In this work, we analyze a service center that represents an intermittently connected mobile publisher or subscriber. This is modeled explicitly as an "ON/OFF queueing center" and relates to the publisher/subscriber being either connected (ON) or disconnected (OFF). By analyzing this queueing center, we are able to derive theoretical results relating to the expected end-to-end response time of the events. The key contributions of this work are:

1. Introducing an overall model and end-to-end event delivery model for large-scale mobile pub/sub systems as queueing networks, which provides simple and sufficiently accurate performance metrics.
2. Incorporating the "ON/OFF queueing center" into the publisher's nodes and at each broker to take into account mobile characteristics of publishers and subscribers, such as connections/disconnections.
3. Modeling the end-to-end response time from publishers to subscribers by connecting several queueing centers. This is done by following the message routing path taken via the broker overlay network and analyzing the rates and service demands at each station.
4. Providing an analytical solution to the intermittent service (ON/OFF) queueing center, which can then be used as separate component inside large queueing networks. Hence, we enrich the existing bibliography on QNMs and their solutions.
5. Validating the model using simulations based on both probability distributions and real-world workload traces. We calculate end-to-end response times from real world traces in various scenarios, subject to intermittent network connectivity or voluntary user disconnections.

As QNMs offer a simple modeling environment, this methodology can be used by system designers/developers for system tuning and capacity planning. Application developers are able to study multiple scenarios by varying parameters (i.e., arrival rates, intermittent connectivity and service demands) in our proposed model. Middleware developers are able to model middleware systems with mobile characteristics by utilizing the intermittent service center. To the best of our knowledge, our work is the first which is utilizing QNMs to model intermittently connected mobile peers as service centers and analyzing accordingly a pub/sub system. The rest

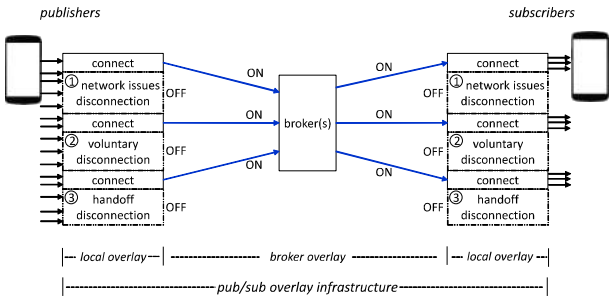

Figure 1: Peer's connectivity behaviour.

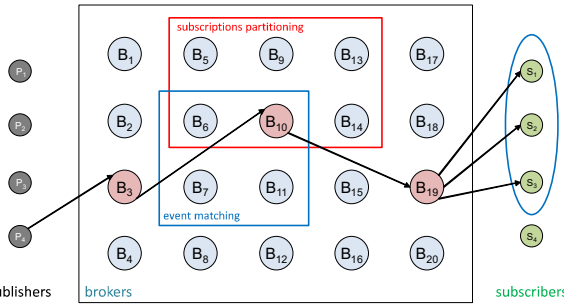

Figure 2: Network of brokers in a distributive pub/sub system.

of this paper is organized as follows: Section 2 introduces the system model of the pub/sub system and the parameters used to represent publishers, brokers and subscribers. In Section 3, the theoretical analysis of end-to-end response times for event delivery is provided. Comparison with simulations is considered in Section 4. Finally, related work is discussed in Section 5 followed by conclusions in Section 6.

2. SYSTEM MODEL

In this section, we first develop a model for mobile pub/sub systems. We then formulate the problem of calculating end-to-end delay for event delivery, which we call *end-to-end response time*, based on this model description.

2.1 Mobile Pub/Sub system

In this work, we assume mobile pub/sub systems, where multiple peers (publishers and subscribers) are mobile entities that interact with each other via a network of intermediate brokers. Each broker is linked to publishers/subscribers via wireless communication links and to other brokers via wired communication links. A pub/sub system generally builds upon an application-level overlay infrastructure. As depicted in Fig. 1, peers build upon the *local overlay* and they connect/disconnect to the *broker overlay* to send or receive events. More specifically, *publishers* produce events to their *local overlay*, which is occasionally connected to the *broker overlay*, to forward the published events. *Subscribers* connect occasionally (intermittent availability) to the *broker overlay* through their *local overlay* to receive new events. Both peers, connect and disconnect (state ON/OFF in Fig. 1) due to several reasons: *i)* there are network issues forcing disconnection (case ① in Fig. 1); *ii)* the peer disconnects from the broker overlay on a voluntary basis (e.g., to save energy) (case ② in Fig. 1); and *iii)* there is a peer's handoff between two brokers (case ③ in Fig. 1).

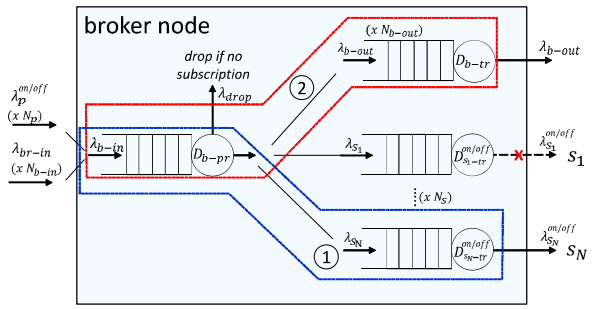

Figure 3: Pub/sub broker system model.

To support distributed applications spanning a wide-area, the pub/sub system has to be implemented as a set of independent, communicating brokers, forming the *broker overlay*. Based on [17], in such architectures, peers can access the system through any broker that becomes their *home broker*. Regardless of the subscription model (topic-based or content-based), subscribers notify interest for specific types of events (topics) to the home broker; subscriptions are spread to a subset of existing brokers (*subscription partitioning* process in Fig. 2). Such a technique of subscription partitioning provides higher scalability and improved efficiency in latency/network utilization.

On the other hand, publishers produce events characterized each by specific types (e.g., a topic) to their home broker; the subset of corresponding subscribers is determined through the *matching* process (see Fig. 2 with topic/content based filtering) which is performed by the pub/sub system. The produced events are delivered to all the determined subscribers by using the *event routing* process. This process is performed by using several algorithms, such selective routing or event gossiping. Regardless of the event routing algorithm used, produced events will pass through a specific path of brokers towards the subscribers. For example, based on Fig. 2 produced events will pass through B_3, to B_{10} and finally to B_{19}, which is the subscribers' home broker. In implementations such as Hermes [18], topic based routing of messages across brokers is carried out using a hash on the event types. Publishers wanting to publish send an advertisement message before eventual publication. A rendezvous node is set up by routing the message to a node closest to the hash of the event type name. Subscriber's subscription messages are also forwarded towards the rendezvous node. During the routing process, if an intermediary node with a particular event of interest is encountered, publications are sent using the reverse path (between the intermediary node and the subscriber), so as to prevent overloading of the rendezvous node. Such a system improves scalability and fault tolerance when compared to single broker architectures.

We model the pub/sub middleware components using simple input and output queues. An input queue is used to receive and process events and an output queue to transmit them. To tackle with mobile peer's connections and disconnections we introduce the *ON/OFF queue*. Accordingly, we model a publisher by using an *output ON/OFF queue* in its local overlay to transmit published events. If the publisher disconnects (state *OFF* in Fig. 1), events remain at the *ON/OFF queue* until its next connection (state *ON* in Fig. 1), in which are transmitted to its home broker. A subscriber is modeled as an *input queue* that receives events from the access broker. Finally, a broker (e.g., B_{19}) is modeled using a single *input queue* and multiple *output queues* as depicted in Fig. 3. Particularly, events arrive at a single input queue from generally multiple publishers/brokers with an arrival rate $\lambda_{\text{b\_in}}$. Replication or dropping of events can occur at the exit of the input queue, depending on the subscriptions. If the event is not dropped is forwarded to: *i)* generally multiple continuous *output queues* for its transmission to other brokers (case ① in Fig. 3); *ii)* generally multiple *ON/OFF output queues* for its transmission to corresponding local subscribers (case ② in Fig. 3). In case a subscriber is disconnected, the event remains in its dedicated *ON/OFF output queue* until the next connection of the subscriber.

Based on the existing literature [19], every pub/sub middleware implementation (e.g., JMS [20], Gryphon [21]) offer some specific guarantees, with regard to the underlying transport protocols of the communication links. We assume that the underlying transport protocol of our system is *reliable* and offers the following guarantees: *i) Reliable message transmissions*: the underlying infrastructure guarantees that **the delivery of each message is verified** using: acks, nacks, timeouts and re-transmissions; *ii) FIFO Event ordering*: events are served in FIFO order upon arrival at any middleware component; and *iii) Persistent subscriptions*: Subscriptions are assumed to have long validity compared to the ON/OFF periods.

Thus, based on the above the broker/broker, broker/publisher, and broker/subscriber links are reliable and there are no event losses. Moreover, we assume that each broker is always up and running (e.g., deployed in a cloud [22]) and there is no broker overload and sufficient queue capacity so that no events are dropped.

The proposed pub/sub model allows representing a variety of existing mobile pub/sub systems [4, 22–24]. Events are delivered in an asynchronous way (push-based system), but our model can represent pull-based systems [25] as well. As a future direction, we intend to extend this model by applying time-to-live lifetime periods to each published event.

2.2 End-to-end Response Time Problem

With the presented mobile pub/sub system, the question that arises from a performance perspective is the following: Given the rate of published events and the availability (in terms of connectivity) of each mobile peer (publisher and subscriber), *what is the end-to-end response time of events from each publisher to each subscriber*? To estimate the end-to-end mean response time from a publisher p to a subscriber s, we define a metric called *end-to-end mean response time*, denoted by R_s^p.

In order to analytically calculate the end-to-end mean response time, we rely on queueing theory and describe formally the proposed mobile pub/sub system. To mathematically represent the arrival rates of events, we use a topic-based subscription model, since it is efficient and simple in terms of event classification. Nevertheless, our approach can be used for any model where several classification techniques are applied. Let V be the set of all topics in the system. Each *queue* or *queueing center* processes events through a dedicated *server*. Each server supports a specific *service demand* (time needed to process one event) denoted as D, which is exponentially distributed with rate $\mu > 0$. All queueing centers apply a first-come-first-served (FCFS) queueing policy. All notations can be found in Table 1.

Variable(s)	Definition/Description
$p \in P$	a publisher in the set of all publishers
V	set of all topic names in the system
V_{p}, V_{s}	sub-set of topics that p publishes events and s is subscribed
$s \in S$	a subscriber in the set of all subscribers
$\lambda_{\mathrm{p\_in}}/\lambda_{\mathrm{p\_out}}$, $\lambda_{\mathrm{b\_in}}/\lambda_{\mathrm{b\_out}}$, $\lambda_{\mathrm{s\_in}}/\lambda_{\mathrm{s\_out}}$	input or output rate of events at each publisher p, broker b, and subscriber s
ON, OFF	mobile peers states: ON for connected, OFF for disconnected
T_{ON}, T_{OFF}	average duration of ON (connected) and OFF (disconnected) periods
$b \in B$	a broker in the set of all brokers
$N_{\mathrm{s}} \subseteq S$	number of subscribers, subscribed to b
D_{pr}, D_{tr}	the service demand for the processing and transmission of events
$R_{\mathrm{s}}^{\mathrm{p}}$	end-to-end response time from publisher p to subscriber s

Table 1: Model variables' and shorthand notation.

We model the connectivity of pub/sub peers as follows: let ON and OFF be the states where the peer is connected and disconnected, correspondingly. A given peer, is connected (ON state) for an exponentially distributed time period with parameter θ_{ON} ($\theta_{\mathrm{ON}} = 1/T_{\mathrm{ON}}$). Upon the expiration of this time, the peer disconnects (OFF state) and stops sending or receiving relevant events for an exponentially distributed time period with parameter θ_{OFF} ($\theta_{\mathrm{OFF}} = 1/T_{\mathrm{OFF}}$). To model the performance of a component that send events under the effect of the above connectivity, we introduce the "ON/OFF queueing center".

ON/OFF queueing center model : Each such component ($q_{\mathrm{on/off}}$) is dedicated to a pub/sub peer (publisher p or subscriber s) and is responsible for the transmission of events. Its server processes events when the peer is connected. Thus, based on peers' connectivity, the $q_{\mathrm{on/off}}$ server processes events according to the exponential distributions θ_{ON} and θ_{OFF}. A $q_{\mathrm{on/off}}$ queueing center is defined by the tuple:

$$q_{\mathrm{on/off}} = (\lambda_{\mathrm{in}}, \lambda_{\mathrm{out}}^{\mathrm{on/off}}, D_{\mathrm{tr}}, \theta_{\mathrm{ON}}, \theta_{\mathrm{OFF}})$$

where λ_{in} is the input rate of events to the queueing center, $\lambda_{\mathrm{out}}^{\mathrm{on/off}}$ is the output rate of events, and D_{tr} is the service demand for the transmission of events (if any) during T_{ON}. The $q_{\mathrm{on/off}}$ accepts input rate of events according to a Poisson process λ_{in}. The output process $\lambda_{\mathrm{out}}^{\mathrm{on/off}}$ will be intermittent, because no event exits the queue during T_{OFF} intervals. We assume with good approximation that the output process $\lambda_{\mathrm{out}}^{\mathrm{on/off}}$ is Poisson during T_{ON} intervals with rate $\lambda_{\mathrm{in}} * (T_{\mathrm{ON}}+T_{\mathrm{OFF}})/T_{\mathrm{ON}}$, so that the average output rate equals the average input rate. We call this an "intermittent Poisson process". Let $R_{\mathrm{p|s}}^{\mathrm{on/off}}$ be the average delay for the processing of events in $q_{\mathrm{on/off}}$. Without loss of generality, we make the following assumption: if T_{ON} expires and there is an event currently being served, the server interrupts its processing and will continue in the next T_{ON} period.

Publisher Model : Let P be the set of all publishers in the system. Each publisher ($p \in P$) forwards the published events to its home broker when connected. For this, it maintains an "ON/OFF queueing center" to its local overlay. A publisher p is defined by the tuple:

$$p = (id_{\mathrm{p}}, V_{\mathrm{p}}, \lambda_{\mathrm{p\_in}}, \lambda_{\mathrm{p\_out}}, D_{\mathrm{tr}}^{\mathrm{on/off}}, T_{\mathrm{ON}}, T_{\mathrm{OFF}})$$

where id_{p} is the publisher's identifier, $V_{\mathrm{p}} \subseteq V$ is the set of the topics on which p publishes events according to a Poisson process for each topic, $\lambda_{\mathrm{p\_in}}$ is the input rate of the published events which is also Poisson (sum of Poisson processes for the set of topics V_{p}), $\lambda_{\mathrm{p\_out}}$ is the output rate (intermittent Poisson because of the $q_{\mathrm{on/off}}$), $D_{\mathrm{tr}}^{\mathrm{on/off}}$ is the service demand for the events transmission, and T_{ON} and T_{OFF} are the average periods where p connects and disconnects.

Subscriber Model : Let S be the set of all subscribers in the system. Each subscriber ($s \in S$) receives relevant events from its home broker when connected. A subscriber s is defined by the tuple:

$$s = (id_{\mathrm{s}}, V_{\mathrm{s}}, \lambda_{\mathrm{s\_in}}, D_{\mathrm{pr}}, T_{\mathrm{ON}}, T_{\mathrm{OFF}})$$

where id_{s} is the subscriber's identifier, $V_{\mathrm{s}} \subseteq V$ is the set of the topics that s has subscribed, $\lambda_{\mathrm{s\_in}}$ is the input rate which is intermittent Poisson since s receive events when connected (events match s's topics V_{s}), D_{pr} is the service demand for the local processing of events, and T_{ON} and T_{OFF} are the average periods where s connects and disconnects.

Broker Model : Let B be the set of all brokers in the system. Each broker ($b \in B$) receives the published events to several topics from local publishers and other brokers and forwards them to the corresponding brokers or local subscribers according to their connectivity status. A broker b is defined by the tuple:

$$b = (id_{\mathrm{b}}, \lambda_{\mathrm{b\_in}}, D_{\mathrm{pr}}, N_{\mathrm{s}}, D_{\mathrm{s}}^{on/off}, \lambda_{\mathrm{s\_out}}, N_{\mathrm{b\_out}}, D_{\mathrm{b\_tr}}, \lambda_{\mathrm{b\_out}})$$

where id_{b} is the broker's identifier, $\lambda_{\mathrm{b\_in}}$ is the broker's input rate according to a Poisson process due to the following: let $N_{\mathrm{p}} \subseteq P$ be the set of publishers, and $N_{\mathrm{b\_in}}$ be the set of brokers that publish events to b. As already defined, $\lambda_{\mathrm{p\_out}}$ rate is intermittent Poisson. Accordingly, we assume that N_{p} randomly published $\lambda_{\mathrm{p\_out}}$ rates, along with the published rates of $N_{\mathrm{b\_in}}$ brokers, are also Poisson. Incoming events arrive at a single input queue, named "in queueing center" (q_{in}), to be processed. D_{pr} is the service demand to process the incoming events. Events without subscription are dropped. $N_{\mathrm{s}} \subseteq S$ is the set of subscribers subscribed to the broker b. Depending on N_{s}, the broker maintains multiple output "ON/OFF queueing centers" ($q_{\mathrm{on/off}}$) that represent subscribers connectivity. $D_{\mathrm{s}}^{on/off}$ is the service demand to transmit events to s. $\lambda_{\mathrm{s\_out}}$ is the intermittent Poisson output rate of events that match s's topics V_{s}. $N_{\mathrm{b\_out}} \subseteq B$ is the set of brokers connected to b represented by multiple output queues, named "b_out queueing centers" ($q_{\mathrm{b\_out}}$). $D_{\mathrm{b\_out}}$ is the service demand to transmit events to another broker, and $\lambda_{\mathrm{b\_out}}$ is the output rate of events towards another broker (also Poisson).

End-to-end Response Time : To evaluate the average end-to-end response time from p to s, denoted by $R_{\mathrm{s}}^{\mathrm{p}}$, it is essential to consider every peer's connectivity and calculate the delay of the published events at every middleware component they pass through. For instance, considering the path of the events published from p_1 to s_3 in Fig. 2, events pass through: B_3, B_{10} and B_{19} brokers. Thus, it is essential to calculate the delay at: *i)* p_1's local overlay (p_1's $q_{on/off}$); *ii)* each intermediate broker B_3 and B_{10} (case ① in Fig. 3); *iii)* s_3's home broker B_{10} (case ② in Fig. 3); and *iv)* s_3's local overlay.

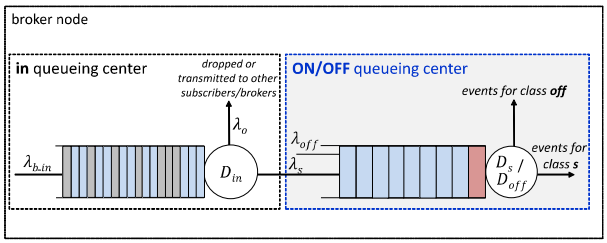

Figure 4: Queueing Network for a broker node.

3. END-TO-END RESPONSE TIME

In this section, we define the properties that allow to build a queueing network. We then model the network of our system in order to calculate the *average end-to-end response time*.

3.1 Broker Queueing Network

To evaluate the end-to-end response time, we build upon *QNMs* [15, 26]. A *queueing network* is a network of connected *service centers* which provides analytical solutions for performance measures (e.g., response time). To obtain such solutions, each service center can be isolated from the rest of the network and evaluated analytically, under specific assumptions. Essentially, service centers are considered to be M/M/1 queues, i.e., featuring Poisson arrivals and exponential service times. The solution of the entire network can then be formed by combining these separate solutions. Even if these assumptions are rarely the case in real-world systems, *QNMs* have proved to provide good approximations in most such systems. In a pub/sub system, events pass through a subset of connected brokers up till the subscriber. To evaluate the performance of such a system, we model it with a network of service centers. Our main contribution lies in enhancing common QNMs with a more accurate model for the *ON/OFF queueing center*.

We evaluate our system analytically. As a first step, we focus on the subscriber's home broker. Such a broker is represented as follows: *i)* events from multiple brokers or publishers arrive at the first queue (*in queueing center*) with rate $\lambda_{\text{b\_in}}$ to be processed; *ii)* processed events are dropped if there is no subscription in the routing table by a subscriber or a another broker; *iii)* maintained events are transmitted (after possible replication) to interested subscribers and brokers; and *iv)* hence, events that match subscriber's topics are transmitted to the subscriber with rate λ_{s}. As already mentioned in Section 2, $\lambda_{\text{b\_in}}$ is Poisson; inside $\lambda_{\text{b\_in}}$, there is an event flow for the subscriber corresponding to a subset of topics (V_{s}), which is a sum of Poisson subflows, one per topic. Moreover, the service demand for the processing of events is exponential. Thus, λ_{s} at the output of the in queueing center is also Poisson.

3.1.1 Delay Calculation at each Service Center

We express the subscriber's home broker as a network of queueing centers (Fig. 4). Based on this queueing network, to evaluate the *average delay* (R_{s}^{b}) into the broker b for a subscriber $s \in S$, it is necessary to calculate the delay at each service center.

The *in queueing center* handles the flow of incoming events to the broker b with rate $\lambda_{\text{b\_in}}$. For each event the service demand is D_{in}. We model the *in queueing center* as an *M/M/1* ($q_{\text{m/m/1}}$). Based on standard solutions for M/M/1 queues [15], the response time is given by:

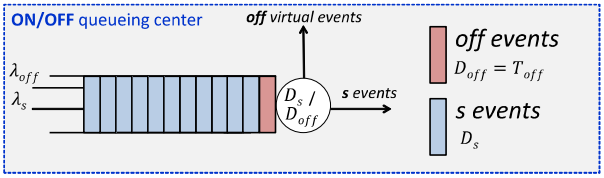

Figure 5: ON/OFF queueing center.

$$R_{\text{s}}^{\text{in}} = \frac{D_{\text{in}}}{1 - \lambda_{\text{b\_in}} D_{\text{in}}} \tag{1}$$

Regarding the *ON/OFF queueing center*, as already defined, its server is subject to an *ON/OFF* process following independent exponential $T_{\text{ON}}/T_{\text{OFF}}$ intervals. Checking existing analytical solutions in the bibliography of QNMs, there is no solution for such a queueing center. In the following, we elaborate a solution based on the *mean value approach* [27]. This approach relies on common assumptions, such as: *i) the PASTA property*, where Poisson events encounter the mean queue upon arrival [27]; *ii) the Memoryless property of the exponential distribution*, where the expected time until completion of an event in service upon the arrival of a new event is equal to the service demand of the event in service; and *iii) Little's Law*.

To evaluate the *ON/OFF queueing center* we introduce - besides normal events - virtual *off* events that follow precisely the $T_{\text{ON}}/T_{\text{OFF}}$ timing of the server: an (*off* event) arrives at the server exactly when the latter goes into its T_{OFF} interval. Moreover, we represent the server's inactivity during the T_{OFF} interval by the service demand of the *off* event. More precisely, we assume that an *off* event arrives at the system exactly at the beginning of a T_{OFF} interval and has preemptive priority over normal events. Hence, it reaches the server also at the beginning of the T_{OFF} interval, which corresponds to its service time.

Accordingly, we set the *off* events to have mean virtual service demand equal to T_{OFF} (let service demand D_{off} equal to T_{OFF}). Additionally, we assume that normal events have mean actual service demand D_{s}. This modeling allows mapping our *ON/OFF queueing center* to one with continuous service. Hence, as depicted in Fig. 5, we specify our model as a two-class model (normal events and *off* events) with preemptive priority [28]. As already defined, λ_{s} is the rate of the arriving s events, based on a Poisson process; events are served with service demand D_{s}, which is exponentially distributed. Moreover, λ_{off} is the rate of the arriving events of class *off*. To specify the λ_{off} rate, we formulate the following theorem.

THEOREM 1. *The average* λ_{off} *rate for the virtual* off *events is given by:*

$$\lambda_{\text{off}} = \frac{1}{T_{\text{ON}} + T_{\text{OFF}}} \tag{2}$$

PROOF. For an outside observer looking at the system an arbitrary point in time, a new *off* event arrives at the beginning of a T_{OFF} interval and is served for T_{OFF}. During T_{OFF} there is no other *off* event arrival. At the end of the T_{OFF} interval, a new *off* event will arrive after T_{ON} time period. Based on the above:

$$\lambda_{\text{off}} = \begin{cases} 0, & \text{during } T_{\text{ON}} \text{ intervals} \\ \frac{1}{T_{\text{ON}}}, & \text{during } T_{\text{OFF}} \text{ intervals} \end{cases} \tag{3}$$

Hence, during $T_{\rm ON}$, the λ_{off} flow is Poisson with exponentially distributed parameter $T_{\rm ON}$. The average $\lambda_{\rm off}$ rate for both intervals is given by:

$$\lambda_{\rm off} = \frac{T_{\rm OFF}}{T_{\rm ON}+T_{\rm OFF}}\ 0 + \frac{T_{\rm ON}}{T_{\rm ON}+T_{\rm OFF}}\frac{1}{T_{\rm ON}} = \frac{1}{T_{\rm ON}+T_{\rm OFF}} \quad (4)$$

Note that the overall $\lambda_{\rm off}$ flow is not Poisson: during the $T_{\rm OFF}$ interval no new *off* event is allowed to arrive. □

The following theorem exploits the PASTA property, priority queueing, and Little's law in order to evaluate the $q_{\rm on/off}$ queueing center.

THEOREM 2. *The average delay or response time $R_{\rm s}^{on/off}$ for the $q_{\rm on/off}$ queueing center is given by:*

$$R_{\rm s}^{on/off} = \frac{\frac{T_{\rm OFF}^2}{T_{\rm ON}+T_{\rm OFF}} + D_{\rm s}\frac{T_{\rm ON}+T_{\rm OFF}}{T_{\rm ON}}}{1 - \lambda_{\rm s}D_{\rm s}\frac{T_{\rm ON}+T_{\rm OFF}}{T_{\rm ON}}} \quad (5)$$

PROOF. In our queueing center, the *off* class has *preemptive priority* over the class *s*. For such a model, a new arriving *off* event has to wait and be served for time:

$$R_{\rm off} = D_{\rm off} + Q_{\rm off}D_{\rm off} \quad (6)$$

where $Q_{\rm off}$ is the number of the *off* events present in the queue. The *off* event has priority over the *s* events and thus, it has to wait only for preceding *off* events (if any). On the other hand, a new arriving *s* event has to wait and be served for time:

$$R_{\rm s} = D_{\rm s} + Q_{\rm s}D_{\rm s} + Q_{\rm off}^{\rm pre}D_{\rm off} + Q_{\rm off}^{\rm post}D_{\rm off} \quad (7)$$

In this case, despite the fact that a new *s* event has arrived, there is always the possibility that an *off* event can arrive. Thus, an event *s* must wait for any *off* and *s* class events that are already in the queue when it arrives, and any *off* class events that arrive during its residence period. Let $Q_{\rm off}^{\rm pre}$ be the average number of *off* events found in the queue when the *s* event arrived and $Q_{\rm off}^{\rm post}$ be the average number of *off* events that arrive in the queue after the arrival of *s*.

Our model has some singularities we should take into account. More specifically, according to the the Theorem 1 $\lambda_{\rm off}$ is not Poisson. Thus, the PASTA property does not hold and $Q_{\rm off}$ in eq. 6, encountered by a new arriving off event, is not the average $Q_{\rm off}$. Nevertheless, we have already defined that it can be only one *off* event in the system. Thus, a new arriving *off* event sees $Q_{\rm off} = 0$, and based on the eq. 6 it has to wait for time:

$$R_{\rm off} = D_{\rm off} \quad (8)$$

On the other hand, *s* class arrivals are Poisson. Thus, in case of a new *s* arriving event, the PASTA property holds. Hence, by taking into account that during $T_{\rm OFF}$ exists only one *off* event in the system and during $T_{\rm ON}$ none, the average $Q_{\rm off}^{\rm pre}$ number of *off* events is given by:

$$Q_{\rm off}^{\rm pre} = \frac{T_{\rm OFF}}{T_{\rm ON}+T_{\rm OFF}}1 + \frac{T_{\rm ON}}{T_{\rm ON}+T_{\rm OFF}}0 = \frac{T_{\rm OFF}}{T_{\rm ON}+T_{\rm OFF}} \quad (9)$$

Furthermore, since $\lambda_{\rm off}$ is different during $T_{\rm ON}$ and $T_{\rm OFF}$ (see eq. 3), we must express the average $Q_{\rm off}^{\rm post}$, separately at each interval:

$$Q_{\rm off}^{\rm post} = R_{\rm s}^{\rm T_{ON}}\frac{1}{T_{\rm ON}} + R_{\rm s}^{\rm T_{OFF}}0 \quad (10)$$

$R_{\rm s}^{\rm T_{ON}}$ is the portion of $R_{\rm s}$ that corresponds to $T_{\rm ON}$ and $R_{\rm s}^{\rm T_{OFF}}$ is the portion of $R_{\rm s}$ that corresponds to $T_{\rm OFF}$. Based on the eq. 7:

$$R_{\rm s}^{\rm T_{ON}} = D_{\rm s} + Q_{\rm s}D_{\rm s} \quad (11)$$

Finally, based on Little's law:

$$Q_{\rm s} = \lambda_{\rm s}R_{\rm s} \quad (12)$$

Thus, based on equations 9, 10, 7 and 12, we use equation 7 to derive the response time $R_{\rm s}$, which is denoted as $R_s^{on/off}$ for the *ON/OFF queueing center*. □

Subsequently, the home broker of the subscriber is modeled by utilizing the (*in queueing center*) connected to the *ON/OFF queueing center*. Thus, we evaluate the average delay, $R_{\rm s}^{\rm b}$, for a subscriber in broker *b* by using the eq. 1 and Theorem 2 as follows:

$$R_{\rm s}^{\rm b} = R_{\rm s}^{\rm in} + R_{\rm s}^{\rm on/off} \quad (13)$$

Based on the above analysis, in case the published events pass through multiple brokers before reaching subscriber's home broker, they are processed at the *in queueing center* and then transmitted through the broker's *out queueing center*. Thus, such a broker is modeled using two *M/M/1 queueing centers* and the average delay $R_{\rm s}^{\rm b}$ is given by:

$$R_{\rm s}^{\rm b} = R_{\rm s}^{\rm in} + R_{\rm s}^{\rm out} \quad (14)$$

3.2 End-to-end delay calculation from a mobile publisher to a mobile subscriber

In the previous subsection, we calculated the average delay of subscriber events into a broker *b*. Let *ps* be the class of events published from a specific mobile publisher *p* and that reach a specific mobile subscriber *s*. Let $\lambda_{\rm ps}$ be the rate of events for the class *ps*. To estimate the *average end-to-end response time* ($R_{\rm s}^{\rm p}$) regarding the events of the class *ps*, we feed the flow $\lambda_{\rm ps}$ alone through the sequence of queueing centers modeling the path between p and s while taking also into account the load of the queueing centers due to all the other flows of the pub/sub system. Thus, we create a path of queueing centers by considering the queueing centers of: *i)* the *p*'s local overlay; *ii)* the subset of brokers that the events pass through; and *iii)* the *s*'s local overlay.

Regarding *p*'s local overlay, Theorem 2 provides a simple analytical solution to calculate approximately the delay of a component that publishes events based on the peer's connectivity. Thus, the *ON/OFF queueing center* can be used to evaluate the performance of a publisher *p*. Regarding each broker *b*, if *b* is *s*'s home broker, the eq. 13 is used, otherwise the eq. 14 is used (see previous subsection). Finally, *s*'s local overlay is modeled by an output queue. Thus, the analytical solution of a simple *M/M/1 queueing center* (eq. 1) can be used to evaluate its performance.

For a given $\lambda_{\rm ps}$ flow, system designers should parameterize the above analytical solutions. Particularly they must define the $T_{\rm ON}$ and $T_{\rm OFF}$ connectivity periods of *p* and *s*, and the service demand (D) when *ps* events are processed at each queueing center. Additionally, they should consider the effect of other flows going through the queueing centers. Let

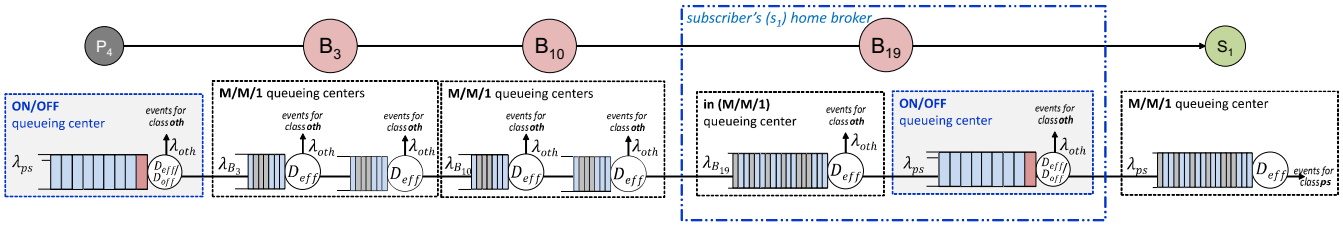

Figure 6: End-to-end queueing network from p_4 to s_1.

oth be the class of events that pass through each queueing center and are different from the events of class *ps*. Let λ_{oth} be the rate of events for the class *oth*; the service demand to serve them is D, same as for ps events. To include the effect of λ_{oth} flows going through the queueing centers, we abstract the utilization of each queueing center by its λ_{oth} flow by calculating the effective service demand D_{eff} for the λ_{ps} flow when λ_{ps} is fed alone through the queueing center. More specifically, for each queueing center, we solve:

$$R(\lambda_{\text{ps}}, D_{\text{eff}}) \;=\; R(\lambda_{\text{ps}} + \lambda_{\text{oth}}, D) \tag{15}$$

meaning that λ_{ps} flowing alone under D_{eff} should have the same response time as when flowing together with λ_{oth} under D.

In case our queueing station is an *M/M/1 queueing center*, based on eq. 1, $D^{\text{ps}}_{\text{eff}}$ can be calculated by:

$$D^{\text{ps}}_{\text{eff}}(\lambda_{\text{oth}}, D) \;=\; \frac{D}{1 \;-\; \lambda_{\text{oth}} D} \tag{16}$$

Similarly $D^{\text{ps}}_{\text{eff}}$ is calculated for the *ON/OFF queueing center* by using our analytical solution of Theorem 2.

Below we provide an algorithm which accepts as input the path of connected brokers $K \in B$ for the delivery of events from a mobile publisher p to a mobile subscriber s. The output of the algorithm is the network of queueing centers and can be used for the evaluation of its performance.

Algorithm 1: Composition of end-to-end queueing network publisher p to subscriber s.

Input: path of connected brokers $K \in B$ from p to s; rate of events λ_{ps}, λ_{oth}; and service demand D at p, s and each broker in K.
Output: end-to-end queueing network QN from p to s, λ_{k} at each queueing center; effective service demand $D^{\text{k}}_{\text{eff}}$ at each queueing center.

Connect the publisher's queueing center::
$QN \leftarrow q_{\text{on/off}}$;
$\lambda_0 \leftarrow \lambda_{\text{ps}}, \quad D^0_{\text{eff}} \leftarrow D^{\text{ps}}_{\text{eff}}(\lambda^{\text{p}}_{\text{oth}}, D^{\text{p}})$;
for $k \leftarrow 1$ *to* K **do**
  if $k = K$ **then**
    Connect the queueing centers of s's home broker:
    $QN \leftarrow QN + q_{\text{m/m/1}} + q_{\text{on/off}}$;
    M/M/1 queuing center;
    $\lambda_{\text{k1}} \leftarrow \lambda_{\text{ps}} \quad D^{\text{k1}}_{\text{eff}} \leftarrow D^{\text{ps}}_{\text{eff}}(\lambda^{\text{k1}}_{\text{oth}}, D^{\text{k1}})$;
    ON/OFF queuing center;
    $\lambda_{\text{k2}} \leftarrow \lambda_{\text{ps}}, \quad D^{\text{k2}}_{\text{eff}} \leftarrow D^{\text{ps}}_{\text{eff}}(\lambda^{\text{k2}}_{\text{oth}}, D^{\text{k2}})$;
  else
    Connect the queueing centers of broker B_{k}:
    $QN \leftarrow QN + q_{\text{m/m/1}} + q_{\text{m/m/1}}$;
    M/M/1 queuing centers;
    $\lambda_{\text{k1}} \leftarrow \lambda_{\text{ps}} \quad D^{\text{k1}}_{\text{eff}} \leftarrow D^{\text{ps}}_{\text{eff}}(\lambda^{\text{k1}}_{\text{oth}}, D^{\text{k1}})$;
    $\lambda_{\text{k2}} \leftarrow \lambda_{\text{ps}}, \quad D^{\text{k2}}_{\text{eff}} \leftarrow D^{\text{ps}}_{\text{eff}}(\lambda^{\text{k2}}_{\text{oth}}, D^{\text{k2}})$;
Connect the subscriber's queueing center:
$QN \leftarrow QN + q_{\text{m/m/1}}$;
$\lambda_{\text{k+1}} \leftarrow \lambda_{\text{ps}}, \quad D^{\text{k+1}}_{eff} \leftarrow D^{\text{ps}}_{\text{eff}}(\lambda^{\text{k+1}}_{\text{oth}}, D^{\text{k+1}})$;
return QN, $\lambda_{\text{k}}\ \forall k \in \{0, K, k+1\}$, $D_{\text{k}}\ \forall k \in \{0, K, k+1\}$;

For instance, in case we apply the above algorithm to the set of connected brokers from p_4 to s_1 in Fig. 2, the output of the above algorithm is the end-to-end queueing network of the Fig. 6. Thus, using our analytical solutions, the *average end-to-end response time* $R^{\text{p4}}_{\text{s1}}$ is given by:

$$R^{\text{p4}}_{\text{s1}} = R^{\text{on/off}}_{\text{p4}} + R^{\text{b3}}_{\text{s1}} + R^{\text{b10}}_{\text{s1}} + R^{\text{b19}}_{\text{s1}} + R^{\text{s1}}_{\text{s1}}$$

where $R^{\text{b3}}_{\text{s1}}$, $R^{\text{b10}}_{\text{s1}}$, $R^{\text{b19}}_{\text{s1}}$ are known by the equations 13 and 14.

In Algorithm 1, we do not deal with the actual routing protocols that are specified by the routing overlay. In actual middleware implementations (Hermes, Gryphon, Siena), the state of the overlay network (events, demand, topology, queue lengths) will determine the broker node routes used. Algorithm 1 aggregates the queue centers based on a selected topology and provides the aggregated end-to-end queuing model.

4. EVALUATION RESULTS

4.1 ON/OFF queueing center validation

To validate our model, we have developed a simulator for mobile pub/sub systems which can be parameterized using well-known probability distributions and actual data derived from a real setup (real traces).

4.1.1 Analytical vs. Simulated Response Time

Our simulator, *MobileJINQS*[1], is an open-source library for building simulations encompassing the constraints of mobile systems. *MobileJINQS* is an extension of *JINQS*, a Java simulation library for multiclass queueing networks [29].

We utilize *MobileJINQS* to implement the *ON/OFF queueing center* described in Section 2. Mobile peers connect and disconnect in the scale of seconds/minutes to send/receive events, depending on the application context. To represent such behavior, we set the ON/OFF system parameters as follows: *i)* the server remains in the *ON* and *OFF* states for exponentially distributed time periods $T_{\text{ON}} = T_{\text{OFF}} =$ 20/40/60 sec, thus, the server changes its state every 20, 40 and 60 sec; *ii)* events are served with a mean service demand $D = 0.125$ sec; *iii)* there is sufficient buffer capacity so that no events are dropped; and *iv)* events arrive to the queue with a mean rate varying from 0.05 to 4 events per sec ($\lambda_{\text{max}} = 4$ events/sec). By applying λ rates greater than 4 events/sec, the system saturates. Using the above settings in our simulator, we run the system and derive the simulated curve of the mean response time for several λ rates as depicted in Fig. 7. Confidence intervals of the simulation results are found to be very small (less than two order of magnitude) and are not presented in the figures.

Subsequently, we apply the same parameters to the equation of Theorem 2 in order to analytically calculate the re-

[1]http://xsb.inria.fr/d4d#mobilejinqs

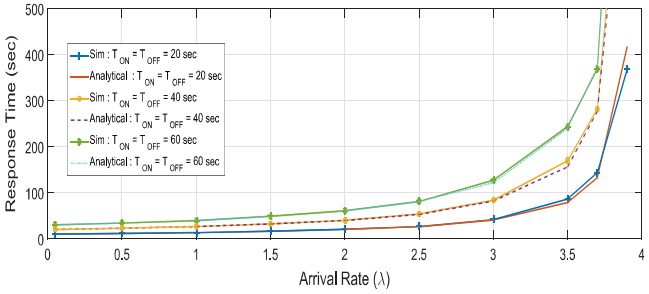

Figure 7: Analytical vs. Simulated Response Times at the *ON/OFF queueing center*.

sponse time as depicted in Fig. 7 through the analytical curve. For a service center where its server is always *ON*, the system does not saturate if $\lambda D < 1$. However, for the *ON/OFF queueing center* the system does not saturate if $\lambda D \frac{T_{\text{ON}}+T_{\text{OFF}}}{T_{\text{ON}}} < 1$ as indicated by the denominator of the Theorem 2. Thus, this confirms that $\lambda_{\max} = 4$ events/sec for a system reaching a steady state. By comparing the curves for the simulated and analytical response times, we notice that results match with high accuracy for arrival rates below to 3.5 events/sec. Differences are noticed for rates equal to or higher than 3.5 events/sec. This is acceptable, since the system is close to saturation at these rates.

4.1.2 *Validation using Arrival Rates from Real Traces*

To further validate our model, we parameterize it using input workloads derived from real traces. The real data, named the *D4D* dataset, was provided to us by Orange Labs in the context of the *D4D* challenge[2]. The *D4D* dataset contains Call Detail Records (CDRs) of users that are subscribed to the Sonatel Telecom mobile operator in Senegal. This data was collected for the whole country over a period of 50 weeks from 7 January 2013 until 23 December 2013. More details about our analysis of the *D4D* dataset and the way we have leveraged it to model the performance of large-scale mobile pub/sub systems can be found in our recent work in [30].

For our validation we used the antenna *traces*. An antenna trace reflects the number of calls made or SMS sent by many mobile users associated to this antenna, over the period of 50 weeks. We assume homogeneity in user access patterns to mobile services (antennas) and application services. For parameterizing the *ON/OFF queueing center*, we map the number of calls or SMS per 10 min interval at the selected antenna to an equal number of events published over the same time interval. Based on [30], this mapping results in a *non-homogeneous Poisson process (or input flow)*, defined as λ_{in}, with rate parameter $\lambda_{(\text{t})}$ piecewise constant in each interval $t \in T$:

$$\lambda_{(\text{t})} = \frac{N_i^t}{|t|} \tag{17}$$

where T is the 50-week period, $|t|$ equals to 10 min, and N_i^t is the number of published events for each 10 min interval at a given antenna i.

Thus, in order to calculate the rate of the input flow (λ_{in}) over the 50-week period at a given antenna i, we use the eq. 17. Subsequently we use these rates to parameterize the *ON/OFF queueing center*. To perform our experiments with representative traces, we selected input flows from a

[2]http://www.d4d.orange.com/en/Accueil

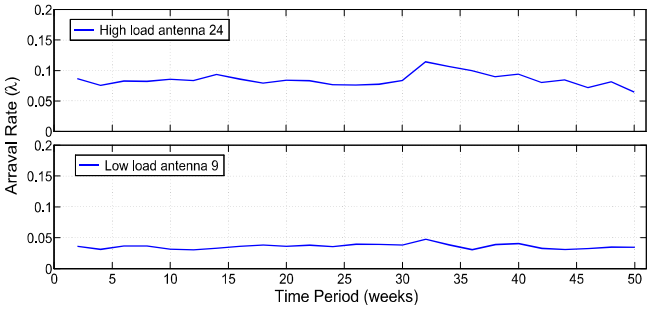

Figure 8: Low and High load Arrival Rates.

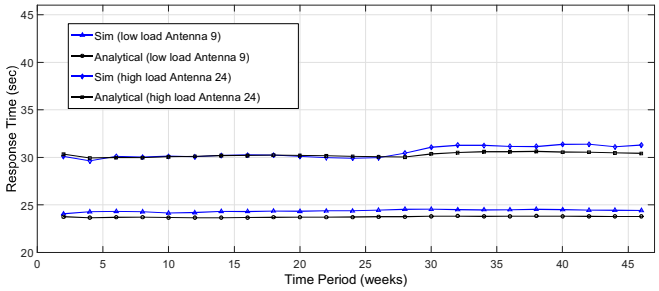

Figure 9: Analytical vs. Simulated Response Times using Arrival Rates from Real Traces.

low load antenna and a *high load antenna*. Fig. 8 depicts two antennas used for our experiments: *i)* antenna 9 has a low load input flow with overall average rate of 0.04; and *ii)* antenna 24 has a high load input flow with overall average rate of 0.075.

To perform simulations and compare the results with our analytical model, we extend the *MobileJINQS* simulator by enabling the application of non-homogeneous input flows to the *ON/OFF queueing center*. Thus, we set the system as follows: *i)* the server remains in the *ON* and *OFF* states for exponentially distributed time periods with parameters $\theta_{\text{ON}} = \theta_{\text{OFF}} = 0.025$ (i.e. $T_{\text{ON}} = T_{\text{OFF}} = 40$ sec); *ii)* events are served with a mean service demand $D = 1$ sec; and *iii)* events arrive to the queue with variable λ rate for each 10 min interval, based on the loads of antennas 9 and 24 (Fig. 8). By running the system with the above settings we derive the simulated curves of the mean response times for the input flows of antennas 9 and 24 over the 50-week period, as depicted in Fig. 9. The mean response times regarding the overall period are 24 and 31 sec, correspondingly for the two antennas.

Subsequently, we apply the same parameter values to the equation of Theorem 2 for each 10 min interval and we calculate the mean response times over the 50-week period as depicted in Fig. 9 through the analytical curves. The mean response times regarding the overall period are 23 and 30 sec, correspondingly for each antenna. Note that we selected these parameter values with respect to our condition ($\lambda_{(\text{t})} D \frac{T_{\text{ON}}+T_{\text{OFF}}}{T_{\text{ON}}} < 1$), in order to avoid the saturation of the *ON/OFF queueing center*. By comparing the curves for the simulated and analytical response times, we notice that the absolute deviation between the two is no more than 5% (approximately 1 sec).

It is worth noting that using the load of antenna 24, the mean response time is much higher in comparison to the one of antenna 9 (7 sec difference). In this case, to get a

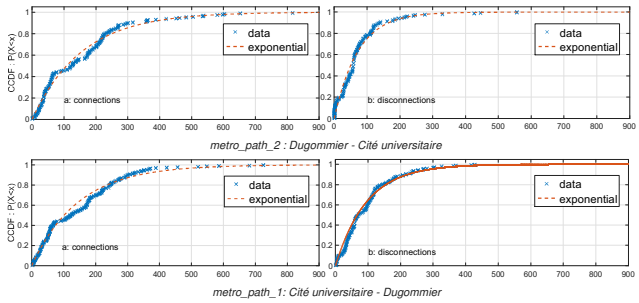

Figure 10: CCDF of connections and disconnections.

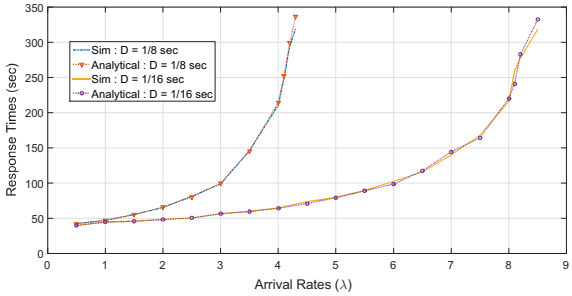

Figure 11: Response Time while traveling in metro path: Cité Universitaire - Dugommier.

lower mean response time, an application developer should set the system to process faster the published events (lower service demand D) of antenna trace 24. Assuming that user access to mobile services (or antennas) is similar to user access to application services, antenna traces can be leveraged for broker capacity planning in related areas. However, performing simulations leads to high development and computational cost in order to obtain accurate results. Therefore, our analytical model can be a useful tool for evaluating the performance of middleware systems.

4.1.3 *Validation using Connectivity Rates from Real Traces*

The analytical solution of our *ON/OFF queueing center* can be parameterized with parameters related to connectivity. To investigate solutions regarding issues in real life, we study the actual connections and disconnections in the metro. Towards this, we have developed an android application, named *Metro Cognition*[3], related to network connectivity data for metro passengers in Paris and Delhi.

Metro Cognition collects connectivity tuples using the Android *BroadcastReceiver* while the user is traveling. Additionally, using a background service it collects more tuples every 30 seconds for safety reasons. Let `con_tuple` be the connectivity tuple with the following 4 values: *i) ON/OFF* (depending on availability of Internet connection); *ii) timestamp* (the exact time when the connectivity status ON/OFF is captured); *iii) provider* (e.g. Vodafone) and; *vi) metro_path_id* (a unique identifier for the user's path from one metro station to another).

One `con_pattern` consists of many `con_tuple`. Each `con_pattern` is created as follows: *i)* the user starts the application and chooses the path between two metro stations; *ii)* the background connectivity service starts; *iii)* when the connectivity status changes and every 30 seconds, a tuple

[3]https://goo.gl/x6vuoB

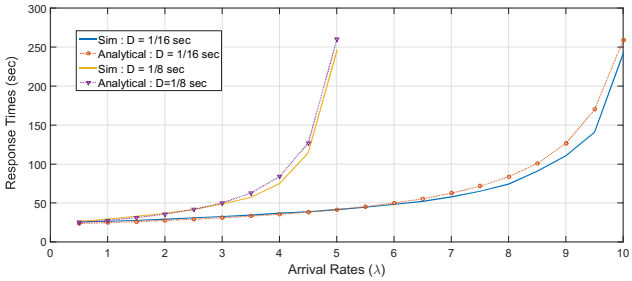

Figure 12: Response Time while traveling in metro path: Dugommier - Cité Universitaire.

(`con_tuple`) is created; *iv)* when the user's journey ends, the background service stops, the data are stored in JSON format and are sent to the cloud server *GoFlow*[4]. In [31], we initiated the creation of a dataset related to network connectivity data for metro travelers in Paris and Delhi.

To utilize our dataset for the validation of the *ON/OFF queueing center*, we concatenate all the `con_patterns` for each *metro_path_id*. So far, we have collected sufficient amount of data in the following metro paths:

-metro_path_1 : Cité Universitaire → Dugommier; *journeys* : 34; *total duration* : 15.18 hours; *average duration journey* : 26.8 min.

-metro_path_2 : Dugommier → Cité Universitaire; *journeys* : 28; *total duration* : 12.13 hours; *average duration journey* : 26 min.

Our data[5] show that metro travelers lose and recover network connection for periods equal from several seconds to 5 minutes, maximum. In average, connected periods are 1.5 times larger than the disconnected periods. As a first step, we specify the best fit of our data to existing probability distributions by applying the same method as in [9]. Fig. 10 shows the complementary cumulative distribution functions (CCDFs) of connections and disconnections at each of the above metro paths. An interesting observation from CCDFs is that our traces fit best with exponential distributions. More specifically, the measured statistics fit very well with 146.38 (i.e., T_{ON}) and 96 (i.e., T_{OFF}) seconds in average for connections and disconnections while traveling in path 1. For path 2, the average connection time is 155.88 sec (i.e., T_{ON}), and the average disconnection is 72.15 sec (i.e., T_{OFF}).

Using our simulator, we start with the 1st metro path (Cité Universitaire → Dugommier) and we generate connectivity data that follow exponential distributions with the above parameters: $T_{\text{ON}} = 146.38$ sec and $T_{\text{OFF}} = 96$ sec. Subsequently, we are able to calculate the delay of events when sent and received by metro travelers. Fig. 11, compares the response times between the theoretical analysis and the model-based simulation when applying the above connectivity parameters, various arrival rates, and various service times. We notice that results match with high accuracy. By applying service time, $D = 0.125$ sec, the system response time becomes too high for λ rates greater than 3.5 events/sec. To tune the system providing better response time, events should be processed faster. Thus, by applying service time, $D = 0.0625$ sec, the response time is too high for λ rates greater than 7 events/sec.

Regarding the 2nd path (Dugommier → Cité Universitaire), we confirm the correctness of the above approach as

[4]https://goflow.ambientic.mobi

[5]https://goo.gl/1SBiaU

follows: instead of generating data that follows exponential distribution, we directly apply the derived ON/OFF periods from the real traces. Using the same setup as previous and the average connected and disconnected periods (T_{ON} = 155.88 sec and T_{OFF} = 72.15 sec) in the 2nd path, Fig. 12 compares the response times between the theoretical analysis and the trace-based simulation. Results match with high accuracy for low rates. For higher rates, there is a quite good match between the two with the maximum difference of about 10%. The gap comes from the fact that our dataset is still small, and thus, simulation may not always reach mean response times. In our future work, we intend to continue the collection of data to perform experiments in several other paths of the metro in Paris.

4.2 End-to-end Response Time Evaluation

In Section 3, we provide an approach to model and evaluate the end-to-end response time over a pub/sub system by focusing on the connectivity of mobile peers. Based on several overlay infrastructures such as Jedi [32], Siena [33], Gryphon [21] and Hermes [18]; various topologies can be formed. In this paper, we are agnostic to the type of overlay topology used; and system designers are able to use Algorithm 1 to model all the queuing centers for end-to-end communication from the publisher p to the subscriber s. However, depending on the time scale of user's connectivity the response time might be significantly affected. In this subsection, we study multiple scenarios for parameterizing the queueing centers, by considering three types of connectivity and a specific overlay topology.

4.2.1 End-to-end System tuning

For our experimental setup, we use the end-to-end queueing network of the Fig. 6 to evaluate the response time from publisher p_4 to subscriber s_1. Application developers can utilize our proposed analytical solutions to study multiple application scenarios by varying the following parameters: λ, T_{ON}, T_{OFF} and D.

For all the experiments bellow we assume that the publisher p_4 transmits its events; and the subscriber s_1 process them to its local queue with service demand $D_{s_1} = 62.5$ ms. These values are set in order to evaluate if the delay introduced by the broker path significantly affects end-to-end latency. We define the peers' connectivity by considering disconnections: *i)* network issues; *ii)* voluntary reasons; *iii)* degraded network.

Network Issues

The actual connection/disconnection status of the mobile user depends on the network coverage and capacity. The average connection periods depend also on the type of user mobility. For example, there periods differ for pedestrians, vehicular, rail and metro passengers. For the present evaluation, we employ scenarios with metro passengers. In [31], we conclude that connectivity patterns of a path depend on both the network coverage and crowdedness of the metro. Thus, we evaluate the end-to-end response of the queueing network in Fig. 6 by considering the following scenarios:

Scenario 1 : p_4 travels on the path *Bastille* $\rightarrow$ *Cité Universitaire*. It is connected for $T_{\text{ON}} = 109$ sec and disconnected for $T_{\text{OFF}} = 121.5$ sec in average. It produces events with rate $\lambda = 1$ event/sec that reach s_1 who travels on the path *Cité Universitaire* $\rightarrow$ *Dugommier* and is connected and disconnected for $T_{\text{ON}} = 146.3$ sec and $T_{\text{OFF}} = 96$ sec in average. Both passengers travel during the morning.

Scenario 2 : p_4 travels on the path *Étienne Marcel* $\rightarrow$ *Mairie de Montrouge*. It is connected for $T_{\text{ON}} = 292.1$ sec and disconnected for $T_{\text{OFF}} = 78.7$ sec in average. It produces events with rate $\lambda = 1$ event/sec towards s_1 which travels on the path *Dugommier* $\rightarrow$ *Cité Universitaire* and is connected and disconnected for $T_{\text{ON}} = 155.8$ sec and $T_{\text{OFF}} = 72.1$ sec in average. Both passengers travel during the evening.

We assume that the broker network is reliable with no broker overload. Thus, the event delivery latency is in the scale of hundreds of milliseconds to 1 second. Accordingly, for each broker of the path (from p_4 to s_1), we define the service demand (D) at each queueing center equal to 0.0625 sec. The average connectivity periods are derived from the real traces of the dataset[6] in the metro of Paris. In Table 2, we compare the results of analytical and simulated response times of the above scenarios. The absolute deviation between the two is no more than 5%.

Based on the above results, the response time of the 1st scenario is much higher than the 2nd. This happens because the metro is more crowded during the morning and the connected periods are shorter. Also, since the scale of the peer's connectivity is in tenths of minutes; the transmission and processing delay in the broker path is negligible.

Voluntary Reasons

In this case, common practice of mobile subscribers is to remain disconnected for a long period, and than periodically connect to receive events that match their interests. In this way they are able to save energy resources to their smartphone or to consume their monthly data plan accordingly. For such connectivity periods, we consider the following scenarios: *i)* p_4 connects for $T_{\text{ON}} = 60$ sec to publish events, and disconnected for $T_{\text{OFF}} = 10$ min in average. Events can be produced while the user is disconnected and forwarded when is connected. The publishing rate is $\lambda = 0.5$ event/sec. s_1 connects to receive events for $T_{\text{ON}} = 30$ sec and disconnects for $T_{\text{OFF}} = 15$ min in average; *ii)* p_4 is a web server deployed on cloud always connected ($T_{\text{OFF}} = 0$) and produces events with rate $\lambda = 0.5$ event/sec towards s_1 which connects to receive events (e.g., news feeds) during $T_{\text{ON}} =$ 30 sec and disconnects for $T_{\text{OFF}} = 15$ min in average.

Similarly to the previous experiment, we assume that the broker network is reliable with no broker overload. Thus, for each broker we define the service demand (D) at each queueing center equal to 0.0625 sec. Results are shown in Table 2. Similarly the absolute deviation between the analytical and the simulated response time is no more than 5% and the transmission/processing delay in the broker path is negligible (the scale of the peer's connectivity is in tenths of minutes).

Degraded network

So far, we have assumed that our broker network is reliable. In this experiment, we study situation when brokers connect through a degraded network. Network degradation refers to a decrease in connectivity and an increase in latency throughout a given network. In such a network, mobile users connect and disconnect very often (every few seconds). Published events might be re-transmitted until they reach their

[6]https://goo.gl/1SBiaU

Disconnections	p_4 (T_{ON}, T_{OFF})	s_1 (T_{ON}, T_{OFF})	Simulation	Analytical model	Deviation %
Metro Travel	109, 121.5 (sec)	146.3, 96 (sec)	118.7 (sec)	116.76 (sec)	1.63
	292.1, 78.7 (sec)	155.8, 72.1 (sec)	45.3 (sec)	43.7 (sec)	3.53
Voluntary	60 (sec), 10 (min)	30 (sec), 15 (min)	27.82 (min)	28.31 (min)	1.76
	always, 0	30 (sec), 15 (min)	17.61 (min)	18.03 (min)	2.38
Network Degradation	1 (sec), 1 (sec)	1.5 (sec), 1.5 (sec)	2.41 (sec)	2.26 (sec)	6.22

Table 2: Estimated vs Measured Response Times.

final destination. The output flows of events increases at each publisher/broker.

Based on the above, we consider the following scenario: p_4 connects for $T_{ON} = 1$ sec to publish events, and disconnected for $T_{OFF} = 1$ sec in average. s_1 connects to receive events for $T_{ON} = 1.5$ sec and disconnects for $T_{OFF} = 1.5$ sec in average. Since the broker network is degraded, we increase the publishing rate at $\lambda = 2$ event/sec (due to retransmissions) with service demand D at each queueing center equal to 0.0625 sec. Results are shown in Table 2. Note that for this scale of peers' connectivity, the transmission/processing delay in the broker path is not negligible.

5. RELATED WORK

The pub/sub interaction paradigm is increasingly being used in the design of large-scale distributed systems. Consequently, investigating evaluation techniques of such systems has become crucial. In this Section, we present our survey concerning the recent efforts in the design and evaluation of mobile systems. We divide the survey into 2 categories: the 1st is general related work in queueing theory applied to performance modeling of multiple applications, and the 2nd is literature specific to the performance of pub/sub systems.

Concerning the related work in queueing theory, we begin with the work of Mehmeti et al. [8], where *WiFi offloading* is analyzed extensively by providing performance metrics to improve the efficiency. The authors model the WiFi network availability as an ON/OFF alternating renewal process, which is similar to our mobile subscriber's availability. In order to provide performance metrics of the above models, authors investigate a queueing analytical model based on the *2-Dimensional (2-D) Markov chains*. Authors in [9], also use *2-D Markov chains* to model the WiFi offloading, however they only provide numerical solutions. Moreover, a similar approach is followed in [11, 34], concerning the offloading strategies in Mobile Cloud Computing (MCC). A birth-and-death process is a Markov chain, where transitions are allowed only to the neighboring states. Using this approach we are able to express the subscriber's intermittent connectivity [10] and derive performance metrics. However, providing solutions by following this approach will result in high computational cost since the process is solved with numerical methods [35].

Based on the above, expressing the intermittent WiFi availability for a mobile user using 2-D Markov chains, is a complex and tedious procedure. Extending this approach for expressing middleware systems, such as pub/sub, is even more complicated (2nd category). In [14], *Queueing Network (QNs)* and *Performance Petri nets (PPNs)* are compared with respect to *expressive power* and *solution efficiency*. PPNs enjoy an advantage over QNs in representing synchronization (parallel systems) and are probably best suited for design purposes. On the other hand QNs: provide convenient primitives for constructing models; guarantee that are well-formed (i.e., stable, deadlock-free, etc); and can be solved efficiently. Work done in [36] makes use of QNs to estimate performance of web applications using algorithms such as Mean Value Analysis (MVA).

For the 2nd category of related work, we begin with Pongthawornkamol et al [24]. These analytical models abstract the expressiveness of pub/sub systems under unreliable, best effort public networks. In this study, the authors apply *lifetime (or deadline) periods* for each published event, and the *intermittent availability* of each subscriber in order to estimate the subscriber's reliability. Furthermore, in [37] the authors extended the above work by providing closed form expressions of reliability as a function of the number of brokers, transmission range, area dimensions and deadline parameters. Subsequently, Gaddah et al. in [23], focus on user mobility into the pub/sub interaction paradigm to investigate a pro-active caching approach. Based on this work, to design new hand-off management solutions they consider a fixed network topology where transfer/caching of events/subscriptions between brokers occur prior to subscribers' movement. In [12], a methodology for workload characterization and performance modeling of distributed pub/sub systems is presented. In this study, authors use *Queueing Petri Nets (QPNs)* for accurate performance prediction, similar to [13]. In [38], Sachs et al. analyze the SPECjms2007 benchmark using message oriented middleware. QPNs are used to perform an analysis of the benchmark and is able to predict resource utilization, network characteristics and message latency with high accuracy. Mühl et al. [39] present an approach to stochastic analysis of pub/sub systems employing identity-based hierarchical routing. This work only considers routing table sizes and message rates as metrics. Moreover, in [40], authors study the tradeoffs between performance and QoS in pub/sub systems. Performance evaluation process algebra (PEPA) language is used to express the systems. Authors of the three above efforts, try to tackle the basic functionalities of pub/sub systems and they do not consider subscribers' mobility or events' lifetime. Finally, PEPA is used in [41] to analyze the performance of mobile applications.

Concerning the use of a network of brokers, multiple overlay topologies have been proposed for the case of event based publish-subscribe to aid in reliability, scalability and fault tolerance. Notable among these include Jedi [32], Siena [33], Gryphon [21] and Hermes [18]. The event model may be generated through pattern matching, semantic hierarchy or via object based filtering. In Jedi [32], a hierarchy of overlay servers are used wherein the events are forwarded up to parent nodes from sub trees. Siena [33] uses a more efficient peer-to-peer architecture to forward messages. Gryphon [21] makes use of a link-matching algorithm to partially filter out the most probable directions to which the event will be sent.

In this paper, we generate an analytical solution for publish/subscribe performance using queueing network models (QNMs). By using QNMs, developers have the flexibility to design their systems with the evaluation capability of these models. Moreover, developers can use our models to predict response times by taking into account realistic subscriber

mobility traces. The prediction model can be further applied for system tuning and broker capacity planning. Furthermore, we are agnostic to the type of overlay topology used in the broker network – we model all the queuing stations for end-to-end communication from the publisher to the subscriber.

6. CONCLUSIONS

In this work, we model the intermittent connectivity of mobile users over Publish/Subscribe (pub/sub) middleware. *Queueing Network Models* (QNMs) are employed to provide an analytical solution for response times. We incorporate the "ON/OFF queueing center" into the overlay infrastructure of a mobile pub/sub system to model connections/disconnections of mobile publishers and subscribers. The *ON/OFF queueing center* is modeled as a separate center within QNMs, which can be isolated and analyzed within large queueing networks. We then validate the model using both simulations (including ones relying on real-world workload traces) and traces collected in field conditions (metro network coverage). Our analytical model matches simulation results within 5% deviation, proving the efficacy of our work.

In future, we intend to extend this model by applying time-to-live lifetime periods to each published event. Furthermore, we aim to take into account mobile brokers (e.g., deployed in a mobile devices), which requires the addition of further *ON/OFF queueing centers* to the pub/sub model.

Acknowledgements

This work is partially supported by the associate team ACHOR and the H2020 project CHOReVOLUTION. Furthermore, the authors would like to thank Assistant Prof. Ioannis Moscholios - Univ. of Peloponnese, for his helpful review and Dr. Rachit Agarwal - Inria, for his help in the analysis of the real traces.

7. REFERENCES

[1] Y. Sun *et al.*, "A low-delay, lightweight publish/subscribe architecture for delay-sensitive iot services," in *IEEE ICWS*, Santa Clara, CA, USA, 2013.

[2] H. Liu *et al.*, "Client behavior and feed characteristics of rss, a publish-subscribe system for web micronews," in *ACM SIGCOMM*, Berkeley, CA, USA, 2005.

[3] M. Petrovic *et al.*, "G-topss: fast filtering of graph-based metadata," in *ACM WWW*, Chiba, Japan, 2005.

[4] V. Setty *et al.*, "The hidden pub/sub of spotify:(industry article)," in *ACM DEBS*, Arlington, Texas, USA, 2013.

[5] J. Reumann, "Goops: Pub/sub at google," *Lecture & Personal Communications at EuroSys & CANOE Summer School*, 2009.

[6] "PubNub." http://www.pubnub.com, 2013.

[7] F. Mehmeti and T. Spyropoulos, "Performance analysis of "on-the-spot" mobile data offloading," in *IEEE GLOBECOM*, ATLANTA, GA, USA, 2013.

[8] F. Mehmeti and T. Spyropoulos, "Is it worth to be patient? analysis and optimization of delayed mobile data offloading," in *IEEE INFOCOM*, Toronto, Canada, 2014.

[9] K. Lee, "Mobile data offloading: how much can wifi deliver?," in *ACM, CoNEXT*, Philadelphia, Pennsylvania, 2010.

[10] T. Phung-Duc *et al.*, "A simple algorithm for the rate matrices of level-dependent qbd processes," in *ACM QTNA*, Beijing, China, 2010.

[11] H. Wu and K. Wolter, "Tradeoff analysis for mobile cloud offloading based on an additive energy-performance metric," in *VALUETOOLS*, Bratislava, Slovakia, 2014.

[12] S. Kounev *et al.*, "A methodology for performance modeling of distributed event-based systems," in *IEEE ISORC*, Orlando, FL, USA, 2008.

[13] T. Martinec *et al.*, "Constructing performance model of jms middleware platform," in *ACM/SPEC ICPE*, Dublin, Ireland, 2014.

[14] M. Vernon *et al.*, *A comparison of performance Petri nets and queueing network models*. University of Wisconsin-Madison, CSD, 1986.

[15] E. Lazowska *et al.*, *Quantitative system performance: computer system analysis using queueing network models*. Prentice-Hall, Inc., 1984.

[16] F. Baskett *et al.*, "Open, closed, and mixed networks of queues with different classes of customers," *JACM*, 1975.

[17] R. Baldoni and A. Virgillito, "Distributed event routing in publish/subscribe communication systems: a survey," *DIS, Universita di Roma La Sapienza, Tech. Rep*, 2005.

[18] P. R. Pietzuch, *Hermes: A scalable event-based middleware*. PhD thesis, University of Cambridge Cambridge, UK, 2004.

[19] L. Baresi *et al.*, "On accurate automatic verification of publish-subscribe architectures," in *IEEE ICSE*, Minneapolis, USA, 2007.

[20] "Sun Microsystems. JMS Specifications and Reference Implementation." http://www.oracle.com/technetwork/java/jms/index.html.

[21] R. Strom, "Extending a content based publish-subscribe system with relational subscriptions," tech. rep., Technical report, IBM Research, 2003.

[22] J. Gascon-Samson *et al.*, "Dynamoth: A scalable pub/sub middleware for latency-constrained applications in the cloud," in *IEEE ICDCS*, Columbus, Ohio, USA, 2015.

[23] A. Gaddah, "Extending mobility to publish/subscribe systems using a pro-active caching approach," *Journal of Mobile Information Systems*, 2010.

[24] T. Pongthawornkamol, *Reliability and timeliness analysis of content-based publish/subscribe systems*. PhD thesis, University of Illinois at Urbana-Champaign, 2011.

[25] A. Adya *et al.*, "Thialfi: a client notification service for internet-scale applications," in *ACM SOSP*, Cascais, Portugal, 2011.

[26] G. Bolch *et al.*, *Queueing networks and Markov chains: modeling and performance evaluation with computer science applications*. John Wiley & Sons, 2006.

[27] I. Adan and J. Resing, *Queueing Theory: Ivo Adan and Jacques Resing*. Eindhoven University of Technology, 2005.

[28] W. Chang, "Preemptive priority queues," *Operations research*, 1965.

[29] T. Field, "JINQS: An extensible library for simulating multiclass queueing networks, v1. 0 user guide," August 2006.

[30] G. Bouloukakis *et al.*, "Leveraging cdr datasets for context-rich performance modeling of large-scale mobile pub/sub systems," in *IEEE WiMob*, Abu Dhabi, UAE, 2015.

[31] G. Bajaj *et al.*, "Toward enabling convenient urban transit through mobile crowdsensing," in *IEEE ITSC*, Gran Canaria, Spain, 2015.

[32] G. Cugola *et al.*, "The jedi event-based infrastructure and its application to the development of the opss wfms," *Software Engineering, Transactions on*, 2001.

[33] A. Carzaniga, *Architectures for an event notification service scalable to wide-area networks*. PhD thesis, Politecnico di Milano, 1998.

[34] E. Hyytiä *et al.*, "Optimizing offloading strategies in mobile cloud computing," *Cryptanalyst*, 2013.

[35] G. Latouche and V. Ramaswami, *Introduction to matrix analytic methods in stochastic modeling*. Siam, 1999.

[36] A. Kattepur *et al.*, "Performance modeling of multi-tiered web applications with varying service demands," in *IPDPS Workshops*, Orlando, Florida USA, 2015.

[37] D. Kassa *et al.*, "Analytical models of short-message reliability in mobile wireless networks," in *ACM MSWiM*, Miami, Florida, USA, 2011.

[38] K. Sachs *et al.*, "Performance modeling and analysis of message-oriented event-driven systems," *Software & Systems Modeling*, 2013.

[39] G. Mühl *et al.*, "Stochastic analysis of hierarchical publish/subscribe systems," in *Euro-Par*, 2009.

[40] G. Singh and A. Singh, "Measuring tradeoffs between performance and qos in event based systems," 2015.

[41] N. Arijo *et al.*, "Modular performance modelling for mobile applications," in *ACM ICPE*, Karlsruhe, Germany, 2011.