Exploring Computing at the Edge

A Multi-Interface System Architecture Enabled Mobile Device Cloud

Balasubramanian, Venkatraman; Aloqaily, Moavad; Zaman, Faisal; Jararweh, Yaser

DOI
10.1109/CloudNet.2018.8549296

Publication date
2018

Document Version
Final published version

Published in
Proceedings of the 2018 IEEE 7th International Conference on Cloud Networking, CloudNet 2018

Exploring Computing at the Edge: A Multi-interface System Architecture Enabled Mobile Device Cloud

Venkatraman Balasubramanian[‡], Moayad Aloqaily[⋆], Faisal Zaman[†], and Yaser Jararweh[σ]
[‡]Delft University of Technology(TUD), Netherlands.
[⋆]Australian College of Kuwait, Kuwait, Kuwait
[†]Ciena Networks, Ottawa, ON, Canada.
[σ]Duquesne University, Pittsburgh, PA, USA
E-mails: [‡]V.balasubramanian@tudelft.nl, [⋆]M.Aloqaily@ack.edu.kw, [†]Fzaman@ciena.com, [σ]Jararwehy@duq.edu

***Abstract*—Today, mobile applications advancements have overcome limited device capabilities by offloading to costly public cloud. As the edge computing paradigm began to take precedence, a mobile device cloud (MDC) formed at the edge based on idle intra-device resources emerged. This is a result of a customized user-centric composition service request for a time-bound application. Herein, devices volunteer their intra-device resources for producing a compute environment in turn satisfying the needs of the consumer. Now, with the growth of device technology and the available interfaces for accessing multiple radio technologies, a new transport layer protocol called Multi-path TCP was introduced in literature. This protocol enables multiple sub-flows to join for transmitting data simultaneously. However, in scenarios like formation of device clouds, there are issues pertaining to sub-flows that are involved in a device cloud composition. One such issue is the management of sub-flow buffer. As each of these sub-flows have their own respective buffering and characteristic delays, it leads to sub-optimal performance in term of buffer occupancy. Thereby, degrading the quality of the device cloud composition. To this end, we propose an OS side architecture that plays a crucial role in managing the traffic coming from different flows. We model an agent that works conservatively satisfying Kleinrock's law and show a proof of concept experiment**

***Index Terms*—Mobile Edge Computing, Mobile Cloud, Internet of Things, composition, QoS, Sensor Network.**

I. Introduction

MObile device usage has increased manifolds with the IoT revolution [1]. Nevertheless, there are mobile applications such as real-time gaming, face recognition, music OCR and other such computation intensive applications which have become difficult to process locally in the device. These applications overcome the device limited issues for application processing and offload computations to the cloud. Cloud services have limitations pertaining to the access, infrastructure costs, high round trip times, to name a few. In many services applications are solely dependent on the time and place where-in it needs to be executed [2], [3], [4]. Such place-bound activities demand a separate set of requirements which must be addressed closest to the user. This computing environment formed "on-the-fly" with the co-operation between the devices is known as Mobile Device Cloud (MDC) [4] An MDC is a collection of individual devices with seemingly low computational capabilities but collaborate with one-another in the vicinity together forming a low-cost yet resource rich computation entity. This computational environment is spread over ecosystems such as wireless local area networks (WLANs) and Mobile Ad-hoc networks (MANETs) where nodes cooperatively maintain the network. These are predominantly considered as shown in [4], [5] where users can form a wireless network on the spot. Typically, in these cloud environments two factors are visible *viz.* consumer and provider nodes are mobile and service composition changes dynamically based on the resources (nodes) available. This work considers offloading in such environments where individual devices can behave as a service providing node or a service requesting node. We illustrate one such scenario below.

978-1-5386-6831-3/18/$31.00 © 2018 IEEE

A. Scenario

In a stadium where attendees are gathered to see music artists perform, it is a common site that not all the attendees know the lyrics of a particular song. Various applications like musicOCR convert audio to lyrics and assists the attendees in such situations. However, requesting cloud support would lead to higher round trip times adding to application processing etc. that would render the application to be useless. In such cases a mobile device cloud composition comes of use. As shown in Figure 1 and [4] , a request beacon is sent to an access point for discovering nearby devices (known devices of 2-3 friends) who volunteer their resources (R_1, R_2, etc) for MDC formation. Once the request is received, the AP will assist in finding the resources. The ownership of the resources are with individuals who are readily present in the vicinity and are ready to volunteer their services. The diverse collection of resources (natively stored, and are called virtual resources) are composed into a usable device cloud infrastructure via service APIs . A key-enabler here is the protocol called Multipath TCP (MPTCP) [6]. However, due to unsatisfactory device queuing mechanisms and heavy message passing there is drop in quality of the MDC and most times also leads to disruption. Thus, the problems we are targeting are as follows

- Congestion is a problem in such environments as the amount of control messages and request submission

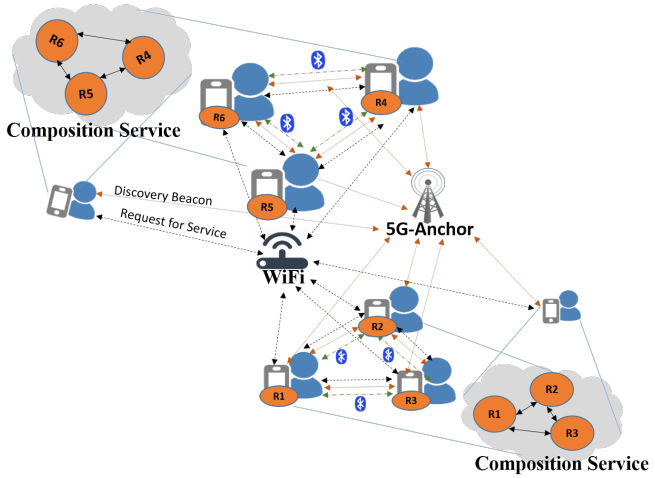


Fig. 1: Scenario Overview

messages increases per-device/request. This leads to poor service times because of heavy on-device buffering which ends up in bloating thereby affecting the MDC negatively.

In view of the above problem, this paper makes the following contributions,

- We propose an architecture that addresses the congestion problem with a transport agent that imposes an application dependent policy for delay specific composition requests. This works in the kernel space with radio and buffer filling information as inputs.
- We introduce a balking technique that allows low latency queuing in mobile device cloud services. To this end, we propose a mathematical model of a *Balking Threshold* (ϵ) a threshold policy that leads to low on-device buffering and a faster request serving scheme, which is integral for a time-bound Mobile Device Cloud application. We show how mean utilization remains constant (β) satisfying Kleinrock's law and provide a sensitivity analysis of the algorithm by rigorous simulated measurements.

The rest of the paper is organized as follows, the related work Section II delineate the novelty of the proposed solution with a brief discussion on the most recent related work. In Section III, the system design is presented with elaboration on the different components included in the solution. Section IV shows performance evaluation and Section V provides concluding remarks.

II. Related Work

In [3], Mtibaa et al. propose computational offloading among mobile devices. Authors show how leveraging the nearby idle computational resources can save execution time and energy consumed. Likewise, in [7] Shi et al. propose Serendipity, an MDC specific profiling and offloading technique that is a continuation of cirrus cloud. A similar vein of research in cloud computing is the Ad-hoc Cloud Computing paradigm. Forming computing environments "on-the-fly" is still the approach followed here. The ad hoc cloud architecture proposed by [2], contains a volunteering set of ad hoc devices who participate for the processing of tasks submitted by cloud users over an ad hoc cloud server. They use reliable ad hoc devices to form small clouds for different applications within an organization to host virtual ad hoc cloud.

In concert with these approaches, in [4] authors propose Mobile Ad-hoc Clouds, that has gained attention with the rise in the number of devices per-person per-household. Unlike all of the above strategies, that are either a base-station controlled D2D techniques or traditional TCP techniques, we propose a Multipath-TCP (MPTCP) enabled device cloud that brings the major benefit of maximizing throughput reaping the multiple paths available between two end-points.

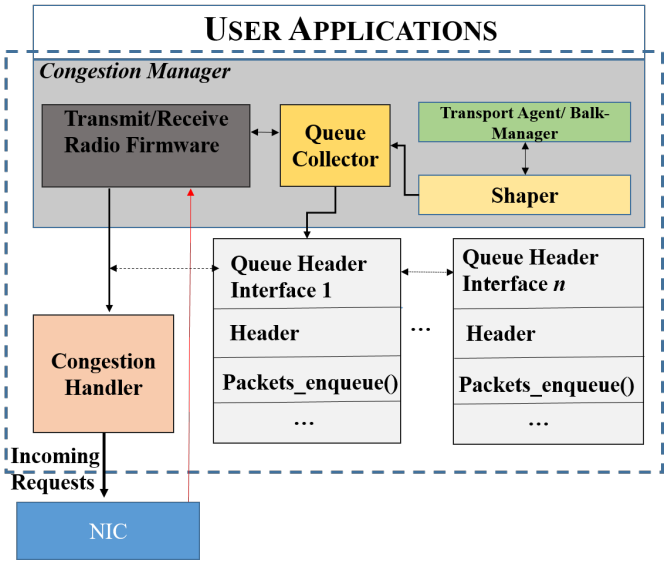


Fig. 2: Congestion Manager Design

III. System Design

As shown in Figure 2 our approach is to incorporate queue management techniques considering multiple-interface technologies. Various transport layer solutions such as [8], [9] work across flows, but do not provide any buffer control inside the device. Assuming, new wireless technologies begin to show up such as 5G and beyond, IoT tech etc., it would be difficult to incorporate on-device solution every-time a new technology is developed. Thus, we propose our design called Congestion Manager that operates in the Kernel space with inputs from the radio transmissions and queuing information.

The main characteristics that we maintain at the Congestion Manager is of a congestion window, a packet transmission rate and composition specific balking value.

A. Congestion Window, Arrival and Service Times

As observed in most congestion control algorithms [8] a sending rate is always specified before the Radio firmware or Tx/Rx begins or becomes active. This is the congestion window ($cwnd$), which is defined as the maximum unacknowledged packets in transit. We consider Poisson arrival λ/unit-time and an exponential service μ/unit-time at the device. Now, the arrivals are such that, the queues have to maintain a considerably low service latency and should offload

packets if it goes beyond a certain point. Our goal here is not to see *how many requests can sit* inside a queue but to demonstrate *how fast the requests* can be serviced. So, we ensure reduced buffering and faster service periods. This is done by the agent that acts as the conduit between the Queue collector and the Shaper.

B. Congestion Handler and the Balking Agent

Congestion Handler keeps the statistics of how fast the packets are arriving and provides this information to the queue collector. Balking Agent does the job of shaping traffic based on value that puts an upper bound on all the queues based on the interface conditions and queuing discipline. Traditionally, Balking is the refusal of an element (queuing parameter) to join a queue [10]. We exploit this technique to an extent where this threshold restricts the amount of requests indirectly in one queue making all the queues participate without waiting for buffering. Consider a balking agent pruning the queue at memory location τ. At time t, for a Poisson arrival of request λ being served by a device μ, the queue size Q. As the queue size is reduced to $Q-\tau$, it is the point where the arrivals move to the next available fast moving queue. Typically, an offloading process works on a priority basis, however, we explore a novel mechanism of balking based request offloading between interfaces to maintain service continuity and reduce network congestion. In doing so, we investigate the probability of *maintaining the service* guarantees at reasonable latency. Although, the difference between the two techniques is minimal, priority techniques do not let the incoming packets *not join* a certain queue. Through balking we achieve this. Consider a memoryless system with a probability of n requests at a time $(t+h)$ where a small event time period of h determines the requests being serviced, if there are $(n-1)$ requests at time t during h, when one request arrived with no completed service requests. In such a case we define, a balking probability ϵ= n/N, where n is the requests in the system including the request being serviced and N is the requests who are going to join the queue. We consider that the serviced requests continues to grow. The mean balking rate (υ) from [10] is taken as,

$$\frac{\upsilon}{\lambda} = \sum_{n=1}^{n=N} \epsilon p_n \tag{1}$$

Our objective is to minimize Z the packet drop x related due to balking

$$Z = min\{x, \upsilon\} \tag{2}$$

such that, a buffer b_i for interface i with link conditions estimated with w_i we have pruning done at τ, represented as $\tau \leq b_i w_i$. Further, the new arrival after the first balk, would be reduced as ($\lambda - \upsilon$)=δ. We avoid the steps showing the steady state probability where all the time dependence is negated.

$$p_{n-1}(t)\delta + p_{n+1}(t)\mu = p_n(t)(\delta h + \mu) \tag{3}$$

In such a situation when a system does not have any requests queued we evaluate for $p_0(t+h)$ as

$$p_0(t+h) = p_0(t)(1-\delta h) + p_1(t)(1-\delta h)\mu \tag{4}$$

As, the probability of no service $(1-\mu h)$ is 1, effectively we maintain

$$\frac{\delta}{\mu} \leq 1 \tag{5}$$

The expected number of requests waiting to be served in the system L_s is reduced as

$$L_s = Q + \delta/\mu \tag{6}$$

From little's law we can also know Q=L_q based on the associated time and length of the system

$$L_q = \delta W_q \tag{7}$$

where, W_q is the average time spent in the queue. Hence, we have the new arrival as, $\frac{\mu p_1 + \upsilon p_0}{p_0}$

Now, for establishing work conservation with λ from above equation, we have,

$$\rho_i = \lambda_i x_i \tag{8}$$

where x_i is the mean service time, λ_i is the average arrival rate. By Kleinrock's conservation law, we have for an average queue delay q_i

$$\sum_{i=0}^{i=n} \rho_i q_i = \beta \tag{9}$$

This is the mean utilization of the link to the i^{th} connection. $\forall\ i \in \{0..n\}$ flows and β is a constant.

Significance- As the queuing time per device decreases, the packet's served by a work conserving scheduler takes short time. To keep mean utilization ($\rho_i = \lambda_i x_i$) constant for the system, let's say it comes at the expense of increase in delay in a device with low resources to spare (as a composition can have processing/storage resources of different varieties) then the delay is bound to increase at such a serving unit. That is, Kleinrock's law for a queuing delay q_i states that, $\sum_{i=0}^{i=n} \rho_i q_i = \beta$ and $\forall\ i \in \{0..n\}$ flows and β is a constant.Essentially, utilization is conserved even in such a scenario. In order to maintain brevity, we show the complete proof in our future work. The functional interaction is shown in Figure 3.

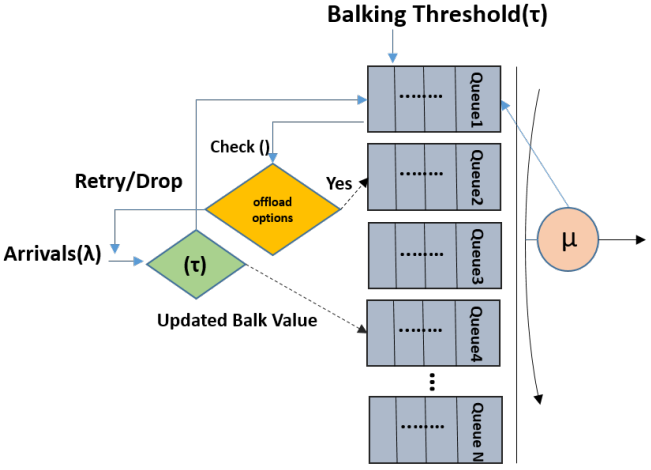
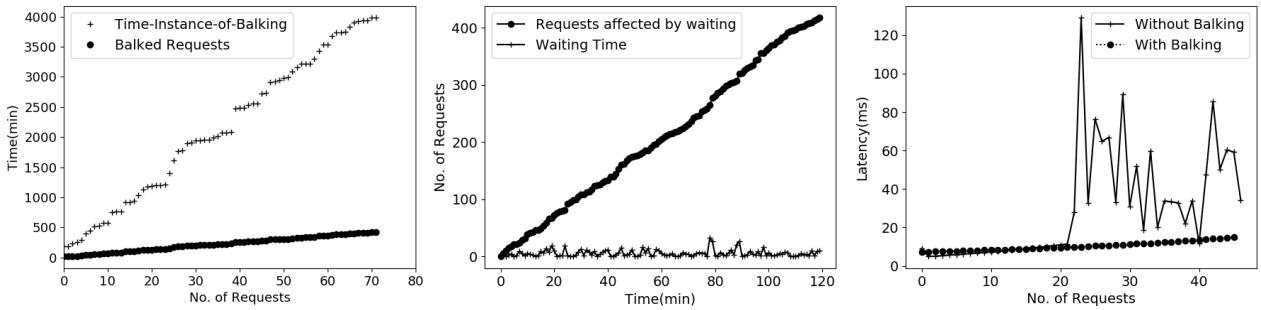


Fig. 3: Functional Interaction



a: Effect of Balking b: Requests offloaded c: Latency Reduction

Fig. 4: Performance Evaluation

IV. Performance Analysis

In this section we perform the sensitivity analysis of our algorithm.Owing to lack of libraries in traditional simulators suitable for simulating such a diverse set of scenarios, we developed our simulator in Python. We run trials for 4000 mins each and average the measurements. The Balking threshold is drawn from exponential distribution and is assigned to each job. Jobs are served in the order of arrival and a max of two jobs are allowed to remain in the system. Each job has an alloted timer values distributed exponentially with τ. On reaching τ the job is reallocated to another interface queue. We call this event as Balking. Figure 4a shows the chunks of requests which are balked at respective balking instances. Figure 4b shows the offloaded requests at the time of balking. We avoid link estimation factors as found in [11] owing to space limitations. The balking algorithm is run on a software controller over a Linux implementation of MPTCP that further enables, the resulting delay characteristics are shown in Figure 4c. Implementation level approximations were made to enable delay characteristics more suitable to our application. Over 50 sample balked requests were considered. Owing to the movements and change in link conditions the fluctuations are evident. However, with the balking threshold in the device the congestion manager can restrict requests. In non-balking scenarios, there were more fluctuations observed that can be attributed to the constant packet-in messages to the controller without a pre-defined threshold monitoring.

V. Conclusion and Future Work

We propose a Multipath-TCP enabled Mobile Device Cloud system architecture for time-bound applications. Offloading computation in a one-hop device cloud is beneficial in terms of cost and average service time. We target a specific problem of excessive on device buffering that leads to bloating in case of densely crowded environments.

To do this end, we adapt a balking technique that enables a better queue management between interface queues, so that the overall queuing time reduces which in-turn control buffer occupancy from the OS kernel. We demonstrate the use and effectiveness of the architectural deployments through simulation measurements and sensitivity analysis. Our model works better in terms of providing low end-end delay between a set of consumer and providers. In the future, we plan to investigate decision making process within the queuing time for offloading between multiple composition services and resource allocation [12], in addition to renting cloud services as observed in [13].[1]

References

[1] F. Anjomshoa, M. Aloqaily, B. Kantarci, M. Erol-Kantarci, and S. Schuckers, "Social behaviometrics for personalized devices in the internet of things era," *IEEE Access*, vol. 5, pp. 12 199–12 213, 2017.

[2] G. A. McGilvary, A. Barker, and M. Atkinson, "Ad hoc cloud computing," in *2015 IEEE 8th International Conference on Cloud Computing*, June 2015, pp. 1063–1068.

[3] A. Mtibaa, K. A. Harras, and A. Fahim, "Towards computational offloading in mobile device clouds," in *2013 IEEE 5th International Conference on Cloud Computing Technology and Science*, vol. 1, Dec 2013, pp. 331–338.

[4] V. Balasubramanian and A. Karmouch, "An infrastructure as a service for mobile ad-hoc cloud," in *2017 IEEE 7th Annual Computing and Communication Workshop and Conference (CCWC)*, Jan 2017, pp. 1–7.

[5] S. A. Abid, M. Othman, and N. Shah, "A survey on dht-based routing for large-scale mobile ad hoc networks," *ACM Comput. Surv.*, vol. 47, no. 2, pp. 20:1–20:46, Aug. 2014. [Online]. Available: http://doi.acm.org/10.1145/2632296

[6] C. Paasch, G. Detal, F. Duchene, C. Raiciu, and O. Bonaventure, "Exploring mobile/wifi handover with multipath tcp," in *Proceedings of the 2012 ACM SIGCOMM Workshop on Cellular Networks: Operations, Challenges, and Future Design*, ser. CellNet '12. New York, NY, USA: ACM, 2012, pp. 31–36. [Online]. Available: http://doi.acm.org/10.1145/2342468.2342476

[7] C. Shi, M. H. Ammar, E. W. Zegura, and M. Naik, "Computing in cirrus clouds: The challenge of intermittent connectivity," in *Proceedings of the First Edition of the MCC Workshop on Mobile Cloud Computing*, ser. MCC '12. New York, NY, USA: ACM, 2012, pp. 23–28.

[8] K. Nichols and V. Jacobson, "Controlling queue delay," *Queue*, vol. 10, no. 5, pp. 20:20–20:34, May 2012. [Online]. Available: http://doi.acm.org/10.1145/2208917.2209336

[9] "Van jacobson et al." http://ee.lbl.gov/papers/congavoid.pdf.

[10] J. C. J. Ancker and A. V. Gafarian, "Some queuing problems with balking and reneging. i," *Operations Research*, vol. 11, no. 1, pp. 88–100, 1963. [Online]. Available: https://doi.org/10.1287/opre.11.1.88

[11] M. Kulin, C. Fortuna, E. De Poorter, D. Deschrijver, and I. Moerman, "Data-driven design of intelligent wireless networks: An overview and tutorial," *Sensors*, vol. 16, no. 6, 2016.

[12] B. Venkatraman, F. A. Zaman, and A. Karmouch, "Optimization of device selection in a mobile ad-hoc cloud based on composition score," in *2017 2nd International Conference on Communication Systems, Computing and IT Applications (CSCITA)*, April 2017, pp. 257–262.

[13] B. Brik, N. Lagraa, N. Tamani, A. Lakas, and Y. Ghamri-Doudane, "Renting out cloud services in mobile vehicular cloud," *IEEE Transactions on Vehicular Technology*, 2018.

[1‡] was working in TUD during the culmination of this work