Sizing Buffers of IoT Edge Routers

Jamal Ahmad Khan
Department of Computer Science,
Virginia Tech
Blacksburg, VA
jamal93@vt.edu

Muhammad Shahzad
Department of Computer Science,
North Carolina State University
Raleigh, NC
mshahza@ncsu.edu

Ali R. Butt
Department of Computer Science,
Virginia Tech
Blacksburg, VA
butta@vt.edu

ABSTRACT

In typical IoT systems, sensors and actuators are connected to small embedded computers, called IoT devices, and the IoT devices are connected to one or more appropriate cloud services over the internet through an edge access router. A very important design aspect of an IoT edge router is the size of the output packet buffer of its interface that connects to the access link. Selecting an appropriate size for this buffer is crucial because it directly impacts two key performance metrics: 1) access link utilization and 2) latency. In this paper, we calculate the size of the output buffer that ensures that the access link stays highly utilized and at the same time, significantly lowers the average latency experienced by the packets. To calculate this buffer size, we theoretically model the average TCP congestion window size of all IoT devices while eliminating three key assumptions of prior art that do not hold true for IoT TCP traffic, as we will demonstrate through a measurement study. We show that for IoT traffic, buffer size calculated by our method results in 50% lower queuing delay compared to the state of the art schemes while achieving similar access link utilization and loss-rate.

CCS CONCEPTS

• **Networks → Routers**;

KEYWORDS

IoT, Edge Routers, Buffers

ACM Reference Format:
Jamal Ahmad Khan, Muhammad Shahzad, and Ali R. Butt. 2018. Sizing Buffers of IoT Edge Routers. In *EdgeSys'18: 1st International Workshop on Edge Systems, Analytics and Networking , June 10–15, 2018, Munich, Germany.* ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/3213344.3213354

1 INTRODUCTION

In typical IoT systems, IoT devices obtain measurements from sensors and send them to appropriate cloud services over the internet using a suitable application layer protocol such as CoAP, MQTT, XMPP *etc.* While some IoT application layer protocols, such as CoAP, use UDP, several others, such as MQTT and XMPP, use TCP as their transport layer protocol. Our focus in this paper are the IoT systems that use TCP.

EdgeSys'18, June 10–15, 2018, Munich, Germany

ACM ISBN 978-1-4503-5837-8/18/06…$15.00
https://doi.org/10.1145/3213344.3213354

The IoT devices access the internet through an edge access router. Due to several differences between conventional and IoT networks, such as in terms of traffic type, medium access protocols, and device heterogeneity, vendors have already started developing edge routers tailored for IoT, such as Cisco's 829 router [1] and Norton Core Router [2].

A key design aspect of an edge router is the size of the output packet buffer of its interface that connects to the access link, *i.e.*, the internet. Selecting an appropriate size of the output buffer is crucial because it directly impacts two key performance metrics of the IoT system: 1) access link utilization (*i.e.*, the percentage of time the link is under use at full capacity), and 2) latency (*i.e.*, the time a packet takes to go from sender to receiver). *If the buffer is under-provisioned*, *i.e.*, the size is too small, the access link utilization decreases because TCP in IoT devices takes an intermittent pause on experiencing packet losses and buffer drains quickly, which mean that the access link stays idle until the devices resume sending packets. *If the buffer is over-provisioned*, the latency experienced by the packets increases because during an intermittent increase in the number of packets arriving at the edge router, the packets that arrive are trying to enter when the buffer already has a large number of packets. Hence, they experience large queueing delays. The lower access link utilization and higher queuing delays can lead to decrease in the throughput and/or increase in the power consumption of the IoT devices.

Increasing the access link utilization is important because IoT systems lease access link bandwidths from internet service providers and this bandwidth is expensive, especially for systems deployed in remote locations, such as $1,200 per sensor for a cellular-based soil moisture measuring system [4]. Reducing the latency is important for real-time IoT systems, such as smart power grids: a small increase in latency may cause violations of service level agreements and result in significant financial and functional losses [16].

In this paper, our objective is to calculate the size of the output buffer of IoT edge routers that ensures that the access link utilization stays high and at the same time, the average latency experienced by the packets stays low.

2 LIMITATIONS OF PRIOR WORK

Although researchers have previously proposed methods to calculate optimal buffer sizes for routers in *conventional networks*, the assumptions that the prior works make about the TCP traffic and the network characteristics, do not hold true in IoT systems. Therefore, the problem of buffer sizing must be revisited for IoT. Next, we describe the three important assumptions, either or all of which are made by prior schemes, and explain why they do not hold true in IoT systems.

The first assumption is that the TCP congestion window size of *each* sender follows a *uniform* sawtooth time-series [5, 9, 13, 14]. The sawtooth shape results from the famous additive-increase & multiplicative-decrease congestion avoidance method of TCP. A *uniform* sawtooth time-series is a sawtooth time-series wherein any given crest or trough has the same value as any other crest or trough, respectively. For the sawtooth time-series of a TCP flow to be uniform, that flow must never experience more than one packet drop at a time, and the packet drop should happen exactly when the congestion window size attains the same value as its value at the time of the previous most recent packet drop. While we observed from our experiments (described in Section 3) that the congestion window sizes of IoT devices make sawtooth time-series, the sawtooth time-series are never uniform. The reason being that the TCP flows mostly experience variable number of consecutive packet drops and packet drops do not always occur at a fixed value of the congestion window size. This assumption leads to under-provisioning of the buffer in some schemes [9] and over-provisioning in others [5, 15].

The second assumption is that the TCP flows of the majority of the devices are *long flows*, *i.e.*, the flows are persistent and always stay in congestion avoidance mode [5, 9, 13, 14]. This isn't true for IoT systems because IoT devices are often resource constrained and thus, terminate their TCP connections and sleep soon after transmitting the current data. Hence, the flows are *short flows*, *i.e.*, they are either non-persistent, or in slow start mode, or both. Consequently, the TCP traffic generated by a typical IoT system contains a good mix of both long and short flows. It is imperative to remove this assumption as well because it leads to the over-provisioning of the output buffer because by treating short flows as long flows, the amount of total traffic theoretically expected to arrive at the edge router is greater than the amount of traffic that actually arrives.

The third assumption is that the traffic arriving at the router can be smoothed by asking the senders to pace their TCP traffic, *i.e.*, instead of sending all packets allowed by the congestion window in a single burst, space the packets out over an entire round trip time (RTT) [6–8, 10]. While smoother non-bursty traffic resulting from TCP pacing reduces the buffer size required to keep the access link utilization high, and thus reduces the latency, unfortunately, TCP pacing is not suitable for the energy-constrained IoT systems because it requires the IoT devices to stay awake the entire time. This requirement is contrary to one of the primary goals in the design of energy-constrained IoT systems, viz., keep the IoT devices asleep for as long as possible by performing sensing/actuation and data transmission/reception as quickly as possible. A caveat worth mentioning here is that when any TCP traffic reaches the internet core, it automatically gets paced on the core links due to the significantly larger bandwidths of the core links compared to the access links. Consequently, the TCP traffic arriving at the core routers is already well paced [8]. However, the difference in the bandwidth of the access link, and the bandwidth of the channel between the IoT devices and the IoT edge router, is small and hence, such automatic pacing does not occur.

3 EXPLORATORY STUDY

In this section, we present our observations from two sets of experiments that show that the first two assumptions of prior work, described in Section 2, indeed do not hold in IoT traffic. We do not study the third assumption as it is not an assumption about the traffic characteristics due to TCP's congestion control mechanism, rather it is about traffic properties due to the way the senders schedule packets.

Assumption 1 – Uniform Sawtooth: We took 15 Raspberry Pi 3s and installed Raspbian on them with a modified Linux kernel (ver. 4.9). We placed a `printk` statement in the `net/ipv4/tcp_input.c` to log the value of the `snd_cwnd` variable, *i.e.*, the congestion window size. We physically connected these 15 Raspberry Pi 3s to a server through a NETGEAR FS750T2 switch for which all ports were configured to operate full duplex at 10Mbps, as shown in Figure 1. We deployed an application layer process on each Raspberry Pi that initiates a TCP connection with a broker server and pumps data into that connection using MQTT protocol.

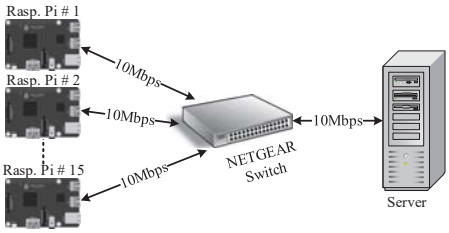


Figure 1: The setup used to study assumption 1 of prior art

Figure 2 plots the TCP congestion windows over time for three randomly chosen Raspberry Pis. We observe from this figure that although all three time-series follow a sawtooth pattern, the sawtooth is not uniform due to the variable number of packet drops experienced by the TCP flows at different time instants. This is contrary to the assumption made by several prior schemes, such as [5, 9, 13, 14].

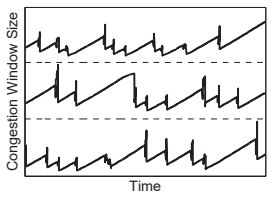


Figure 2: Congestion windows of 3 Raspberry Pis

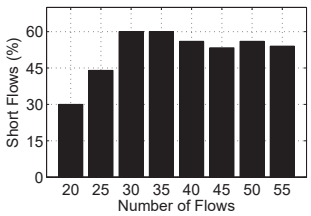


Figure 3: Percentage of short flows in IoT traffic

Assumption 2 – Negligible Short Flows: To study the validity of the second assumption, we performed NS-3 [3] simulations. We could not use Raspberry Pis in this case because the number of IoT devices needed to study this assumption is rather large. We simulated the same topology as shown in Figure 1 with varying number of IoT devices from 10 to 55 using access link bandwidth of 40Mbps, one way link-latency between IoT devices and server of 20.02 ms, and MTU of 1500 bytes. We programmed the application

layer process on each simulated IoT device to generate packets with exponentially distributed inter-arrival times such that the aggregate rate at which the data arrived at the switch was equal to the access link bandwidth. Each IoT device generated a single flow at a time. Note that the IoT devices, especially the ones that are constrained by energy, go to sleep mode and terminate their TCP connections if their corresponding application layer processes do not provide data within a certain time-window. We kept this time fixed at twice the inter-arrival time of packets. Any flow that terminates while it is still in TCP's slow-start mode is considered a short flow; all other flows are considered long flows. We ran our simulator multiple times using a different number of devices during each run and measured how many flows generated by the IoT devices turned out to be short flows. Figure 3 shows a bar chart of the percentage of short flows observed. We observe from this figure that this percentage is large for any number of IoT devices, which is contrary to the assumption made by several prior schemes, such as [5, 9].

On a final note, we emphasize that while the the preliminary measurement study presented above was done using NS-3 simulations and Raspberry Pis in lab environment, the traffic that we used is representative of the traffic generated by real IoT devices. In future, we plan to conduct a similar measurement study using traces from real IoT deployments.

4 BUFFER SIZE CALCULATION

Most IoT devices connect to their corresponding destination cloud service, as shown in Figure 4. Note that there are two access links, one that connects the IoT edge router to the internet and the other that connects the data center hosting the cloud service to the internet. "Today, the core of the internet is over-provisioned with high speed links that experience little congestion" [12]. Thus, it is the access link that is the bottleneck, and it needs carefully sized output buffer at the IoT edge router. In this section, we derive an expression to calculate the minimum size of the output buffer that keeps the access link highly utilized.

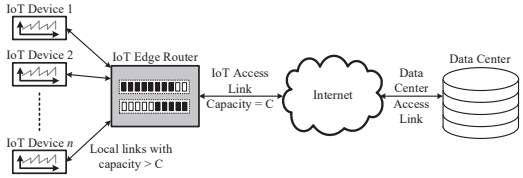


Figure 4: Block diagram of a typical IoT implementation

Fraction of Flows with a Given Number of Dropped Packets: Let n represent the number of flows that are sending packets to the IoT edge router during a given congestion event. Let $\mathcal{P}_T$ be the random variable that represents the total number of packets that arrive at the router during the congestion event out of which, the router drops l packets. Let f_i represent the i^{th} flow, where $1 \leq i \leq n$, and $\mathcal{P}_{f_i}$ be the random variable that represents the number of packets of this i^{th} flow that arrive at the router during the congestion event. Let $\mathcal{D}_{f_i}$ be the random variable for the number of packets of flow f_i that the router drops during the congestion event. The random variable $\mathcal{D}_{f_i}$ follows a binomial distribution $\mathcal{D}_{f_i} \sim \text{Binom}(l, \mathcal{P}_{f_i}/\mathcal{P}_T)$.

In typical IoT deployments, all devices perform the same sensing/actuation tasks and are also often deployed by the same vendor. Consequently, the sizes of their flows follow identical distributions. This implies that the distribution of the number of packets of any given flow arriving at the router during the given congestion event is the same across all flows, *i.e.*, $\forall i, j \in [1, n], \mathcal{P}_{f_i} = \mathcal{P}_{f_j}$, which in turn means that $\forall i, j \in [1, n], \mathcal{D}_{f_i} = \mathcal{D}_{f_j}$ during that congestion event. Therefore, we represent the number of packets of any given flow among the n flows arriving at the router during the congestion event with random variable $\mathcal{P}$ and the number of packets that the router drops of any given flow among the n flows during the congestion event with random variable $\mathcal{D}$. Following from the distribution of $\mathcal{D}_{f_i}$, the random variable $\mathcal{D}$ also follows a binomial distribution $\mathcal{D} \sim \text{Binom}(l, \mathcal{P}/\mathcal{P}_T)$. As the total number of packets that arrive at the router during the congestion event is the sum of the number of packets of all flows that arrive at the router during the congestion event, $\mathcal{P}_T = \sum_{i=1}^{n} \mathcal{P} = n\mathcal{P}$. Consequently, we can represent the distribution of $\mathcal{D}$ as $\mathcal{D} \sim \text{Binom}(l, 1/n)$.

Let $\mathcal{I}_d$ be the indicator random variable for any given flow whose value is 1 if the given flow experiences a drop of exactly d packets. Let $\mathcal{N}_d$ be the random variable that represents the fraction of all flows that experience a drop of exactly d packets. Thus, $\mathcal{N}_d = \sum_{i=1}^{n} \mathcal{I}_d/n = \mathcal{I}_d$. As $\mathcal{I}_d$ is a bernoulli random variable, $E[\mathcal{I}_d] = Pr\{\mathcal{I}_d = 1\} = Pr\{\mathcal{D} = d\}$. Applying the expectation operator on the expression of $\mathcal{N}_d$ and substituting the expression for the distribution of $\mathcal{D}$ in it, we get the following equation to calculate the expected fraction of all flows that experience a drop of exactly d packets during a congestion event, where $0 \leq d \leq l$.

$$E[N_d] = \binom{l}{d} \left(\frac{1}{n}\right)^d \left(1 - \frac{1}{n}\right)^{l-d} \tag{1}$$

Average Congestion Window Size after a Congestion Event: Let W_b represent the average size of the congestion window across all n flows right before the congestion event started. Similarly, let $W_a^{(\xi)}$ represent the average size of the congestion window across all n flows ξ RTTs after the start of the congestion event. During the congestion event, on average, the fraction of flows that experience a drop of d packets will be $E[\mathcal{N}_d]$. As a TCP Reno sender reduces its congestion window size by 50% on detecting each packet drop and as a TCP Reno sender detects only one packet drop per RTT, all flows that experience a drop of d packets will reduce their window sizes by a factor of 2^d after d RTTs since the start of the congestion event. Note that, on average, $E[\mathcal{N}_0]$ fraction of all n flows will not experience any packet drops, and will therefore increase their window sizes. If a flow that experiences no packet drop during the congestion event is in slow start phase, it will double its window size within one RTT. Similarly, if a flow that experiences no packet drop during the congestion event is in congestion avoidance phase, it will increase its window size by the maximum segment size (MSS) within one RTT. Let α represent the expected fraction of flows among the n TCP flows that are long flows, *i.e.*, are in congestion avoidance phase. Thus, the fraction of short flows, *i.e.*, the flows in the slow start phase, is $1 - \alpha$. Hence, the average congestion window size across all flows approximately ξ RTTs after congestion event's start is:

$$W_a^{(\xi)} = \left(\alpha(W_b + \xi\text{MSS}) + (1-\alpha)2^{\xi-1}W_b\right)E[\mathcal{N}_0] + \sum_{d=1}^{\xi}\frac{W_b}{2^d}E[\mathcal{N}_d] \quad (2)$$

The term $\left(\alpha(W_b + \xi\text{MSS}) + (1-\alpha)2^{\xi-1}W_b\right) \times E[\mathcal{N}_0]$ captures the increase in the congestion window size contributed by the flows that did not experience packet drops. It explicitly takes into account the contributions from both long flows and short flows separately by using the terms $\alpha(W_b + \xi\text{MSS})$ and $(1-\alpha)2^{\xi-1}W_b$, respectively. The term $\sum_{d=1}^{\xi}\frac{W_b}{2^d}E[\mathcal{N}_d]$ captures the decrease in the average congestion window size contributed by the flows that experienced packet drops. It also takes the contributions from both long flows and short flows into account because on experiencing a packet loss, both types of flows reduce the sizes of their congestion windows by 50% every RTT.

Output Buffer Size: Let C represent the capacity of the bottleneck access link. Thus, the maximum aggregate bandwidth achievable by all IoT devices is C. Let T represent the average RTT experienced by the packets minus any queueing delay in the router's output buffer. Immediately before the congestion event starts, the buffer must be full (otherwise, the congestion event would not have occurred). Furthermore, at this time instant, the total amount of data sent by all IoT devices that is still unacknowledged is nW_b. This unacknowledged data is either residing in the buffer of the router or accommodated by the delay bandwidth product CT. Let B represent the size of the output buffer. Thus,

$$nW_b = B + CT \quad \Rightarrow \quad W_b = (B + CT)/n \quad (3)$$

Just under one RTT since the start of the congestion event, the TCP flow of any IoT device whose packets were dropped either times-out or receives triple duplicate ACKs, and thus detects a packet drop. On detecting the drop, TCP halves the congestion window size of that flow. As the window size limits the number of unacknowledged packets in the network, the flow is allowed to have a larger number of unacknowledged packets, before a packet drop compared to after the drop is detected. Thus, the flow has more unacknowledged packets than it is currently allowed, and it must pause while it waits for the ACKs for those packets. As our objective is to keep the access link fully utilized, the output buffer must not go empty while some flows are paused. As we are interested in the minimum size for the output buffer, we consider the case when new packets start to arrive at the same moment when the last byte is drained from the buffer. As the sending rate of a TCP flow is equal to the ratio of its window size to the RTT its packets experience, the average sending rate of all n TCP flows ξ RTTs after the congestion event is $nW_a^{(\xi)}/T$. To keep the access link fully utilized, at the moment the last byte drains out of the buffer, the smallest value of this average sending rate $nW_a^{(\xi)}/T$ must equal the bandwidth of the access link C. The expected value of the number of RTTs at which the smallest value of this average occurs is $\xi = \sum_{d=0}^{l} dE[\mathcal{N}_d] = \frac{l}{n}$. This is intuitive because after l/n RTTs since the congestion event, on average, half the flows that had stopped increasing their amounts of outstanding packets have recovered from their respective losses and have started to increase their congestion windows. Thus,

$$nW_a^{(l/n)}/T = C \quad \Rightarrow \quad W_a^{(l/n)} = CT/n \quad (4)$$

Substituting the values of W_b and $W_a^{(l/n)}$ from Eqs. (3) and (4), respectively, into Eq. (4) and solving for B, we get the following closed form solution to calculate the buffer size B.

$$B = \frac{CT\left[1 - \Psi - E[\mathcal{N}_0]\left[(1-\alpha)2^{\frac{l}{n}} - \alpha\right]\right] - \alpha l\text{MSS}E[\mathcal{N}_0]}{\Psi + E[\mathcal{N}_0]\left[(1-\alpha)2^{\frac{l}{n}} - \alpha\right]} \quad (5)$$

where $\Psi = \sum_{d=1}^{l/n}\frac{\binom{l}{d}\left(\frac{1}{n}\right)^d\left(1-\frac{1}{n}\right)^{l-d}}{2^d}$ and $E[\mathcal{N}_0] = \left(1-\frac{1}{n}\right)^l$.

We have used Reno as a case study and the study can be extended to use different congestion control algorithms. The change required will be in the modeling of the change of congestion window in equation . In particular the growth and reduction characteristics of the congestion window would change. The rest of the changes should easily follow.

Parameter Selection: To calculate the value of buffer size B using Eq. (5), we need the values of the following six parameters: 1) C: bandwidth of the access link, 2) n: number of flows , 3) MSS: maximum segment size, 4) T: average round trip time minus any queuing delays, 5) α: percentage of long flows among the n TCP flows, and 6) l: number of packets dropped by the edge router during a congestion event. The value of C is a constant in any given IoT system and the value of n depends on the number of IoT devices and the number of flows each IoT device generates at any given time. Both these values are provided by the network administrator. The value of MSS is defined by TCP. The value of T is dictated by the internet and can be measured by simply timing pings over a few days. The values of α and l, in this work, have been empirically calculated. In our future work, we plan to develop theoretical models to calculate these two parameters.

5 EXPERIMENTAL EVALUATION

While no prior work exists on calculating buffer sizes for IoT edge routers, we still compare our buffer sizing scheme with [5], [9], and [11]. For all of the experiments, we used the topology shown in Figure 1 for three minutes of simulated time on NS-3 with different number of IoT devices using access link bandwidth of 40Mbps, MTU of 1500 bytes, one way latency from the IoT devices to the edge router of 20μs, and one way latency from the edge router to the server of 20ms. The application layer process on each simulated IoT device generated a single flow whose packets had exponentially distributed inter-arrival times, where the rate R at which the process sent packets to the TCP layer was different in different experiments. To produce traffic of γ Mbps on the physical link from an IoT device to the edge router, the application layer process sends messages of size MTU − 40 bytes to the TCP layer at the rate of $R = \lceil x/\text{MTU}\rceil$. The 40 bytes are to accommodate TCP/IP headers.

Buffer Size vs. Link Utilization and Latency: To empirically study how the buffer size affects the access link utilization and the latency experienced by the packets, we ran our NS-3 simulations using $n = 40$ devices, where each device produced traffic of $\gamma = 1$Mbps. Thus, the aggregate traffic was approximately 40Mbps, which is equal to the bandwidth of the access link. We ran our simulations multiple times, where in each simulation, we used a different buffer size in the range $[2 \times \text{MTU}, 2 \times CT/\sqrt{n}]$, where $CT/\sqrt{n}$ is the buffer size proposed in [5].

Figure 5 plots the average utilization of the access link for different buffer sizes when all flows are long and also when 25% of the flows are short. We observe that as the buffer size increases, the link utilization increases because the buffer holds more data and thus goes empty less frequently when some IoT devices pause after experiencing a packet drop. One might conclude that to keep the link utilization high, one could simply over-provision the output buffer. This would be problematic, especially for the latency sensitive applications, because the increase in buffer size increases the queuing delay experienced by the packets, as shown in Figure 6.

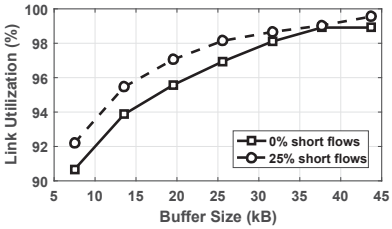


Figure 5: Size vs. Utilization

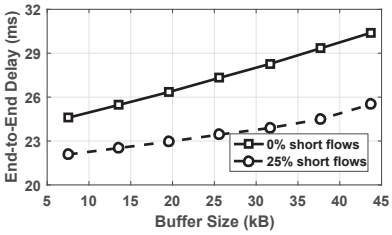


Figure 6: Size vs. Delay

Comparison of Buffer Sizes: Next, we compare the buffer size calculated by our scheme using Eq. (5) with the buffer sizes calculated by [5],[9] and [11]. Figures 7 and 8 plot the buffer sizes for number of flows ranging from $n = 10$ to 100 for two different IoT traffics, one containing 0% short flows and the other containing 25% short flows, respectively. We observe that as the percentage of short flows changes, the buffer sizes calculated by [5] and [9] change because both [5] and [9] calculate the buffer size based on only the number of long flows in the traffic. While the size calculated by [11] does not change as the percentage of short flows changes because it calculates the size based the total number of flows, irrespective of whether they are long or short.

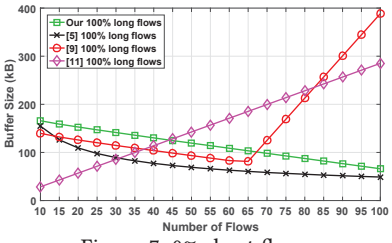


Figure 7: 0% short flows

We make three more observations from Figures 7 and 8. First, as the number of IoT devices increases, the buffer sizes calculated by Eq. (5) as well as by [5] and [9] decrease. This is because with the larger number of IoT devices, when TCP flows of some devices pause after experiencing packet drops, there is a higher probability that at any given time instant, TCP flows of some other IoT devices will be sending data to the IoT edge router and keeping the link

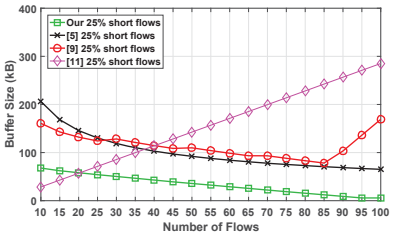


Figure 8: 25% short flows

utilized. Note that after a certain value of the number of flows, [9] starts increasing the buffer size. This is because after a certain number of flows, it shifts the goal from maximizing link utilization to minimizing loss-rate.

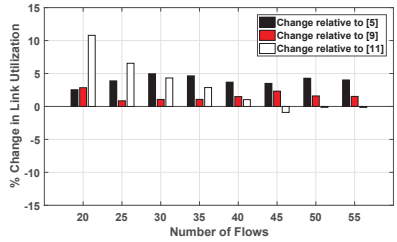
Second, as the percentage of short flows increases, the buffer size calculated by our scheme decreases because when a short flow experiences a packet drop, due to its smaller congestion window size, it pauses for a shorter duration compared to when a long flow experiences a packet drop. Note that in conventional networks, the duration of pause depends on both RTT and congestion window size. However, in the case of IoT deployments, the RTTs of all devices are very similar, and therefore, the duration of pause is primarily dictated by congestion window size.

Third, the buffer sizes calculated by [5], [9] and [11] in the presence of short flows are larger compared to the buffer sizes calculated by Eq. (5), because [5] and [9] consider only the number of long flows and thus, *implicitly* assume that all flows will pause for longer durations, which leads to over-provisioning the buffer. Hence, prior schemes over provision the buffer sizes by several times in comparison to ours which implies that they can keep the access link utilization higher, but at the cost of a significantly larger latency.

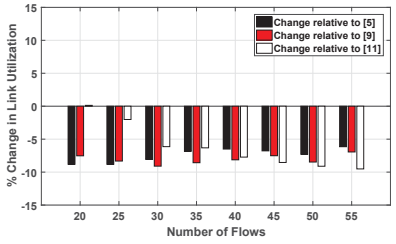
Comparison of Link Utilization, Latency, Loss, & Goodput: For comparison of link utilization, latency, packet loss rate, and goodput among the schemes, we performed NS-3 simulations where each IoT device produced traffic of $\gamma = \frac{40}{n}$Mbps on the physical link, where n represents the total number of flows. Hence the link was congested, regardless of the number of flows. We varied the number of IoT devices from 20 to 55 and generated two different sets of IoT traffics, one containing no short flows and the other containing 25% short flows. The later set of IoT traffic is more important because IoT devices generate more short flows.

Figure 9(a) plots the percentage change in the average link utilization of our scheme compared to prior schemes for traffic that contained no short flows (only long flows). We observe that the link utilizations resulting from the buffer sizes calculated by [5], [9] and [11] are slightly lower compared to our scheme because the buffer sizes calculated by our scheme are slightly larger than prior schemes. Figure 9(b) plots the percentage change in the average link utilization for traffic that contained 25% short flows. As prior schemes significantly over estimate the buffer sizes in the presence of short flows, the link utilizations for them are slightly higher compared to our scheme.

While prior schemes achieve slightly higher link utilization for IoT traffic that contained 25% short flows, this higher utilization comes at the cost of significantly higher latency. Figures 10(a) and 10(b), which plot the percentage change in average queueing delay experienced by the packets, highlight this fact. We observe that for the traffic with 25% short flows, which is more representative

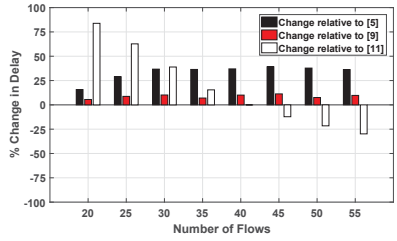


(a) 0% short flows

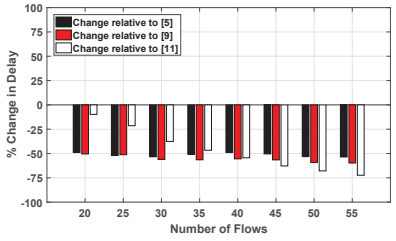


(b) 25% short flows

Figure 9: Change in link utilization by our scheme



(a) 0% short flows



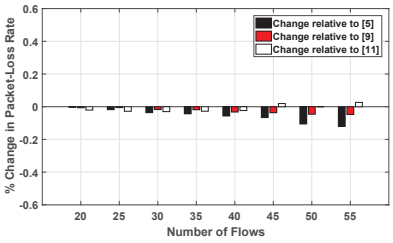
(b) 25% short flows

Figure 10: Change in in delay by our scheme

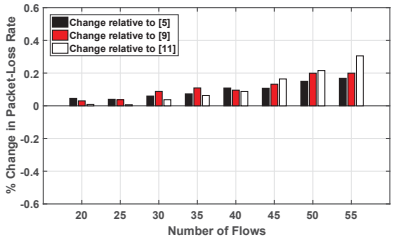
of IoT, the buffer sizes calculated by prior schemes increase the queuing delay by an average of 50% compared to our scheme. Such large queueing delays deteriorate the performance of realtime and streaming IoT applications. Finally if we look at the subfigures in Figures 11 , which plot the change in the packet loss rate, we see that the results are almost the same. Thus, we conclude that our scheme reduces the buffer size significantly, which leads to 50% lower queueing delay for IoT traffic with negligible impact on the remaining 2 performance metrics, *i.e.*, link utilization, and loss rate.

6 CONCLUSION

In this paper, we have presented a theoretical method to calculate the size of the output buffer for IoT edge routers. We have identified three key assumptions of prior art and shown through both real world experiments and NS-3 simulations that they do not hold true



(a) 0% short flows



(b) 25% short flows

Figure 11: Change in loss-rate by our scheme

in IoT traffic. Our results show that for IoT traffic, [5], [9] and [11] over-estimate the buffer size by at least 2×, due to which, flows experience over 50% higher queuing delay compared to when using the buffer size calculated by our scheme. In our future work, we plan to extend our measurement study and evaluations using traces from real-world IoT deployments, and update our scheme based on any important observations we make.

ACKNOWLEDGMENTS

This work is sponsored in part by the NSF under the grants: CNS 1565314, CNS 1405697, CNS 1615411, and CNS 1616273. The authors would also like to thank the anonymous reviewers for their valuable feedback and suggestions.

REFERENCES

[1] Cisco 829 Industrial Integrated Services Routers. http://www.cisco.com/c/en/us/products/routers/829-industrial-router/index.html.
[2] Norton Core Router. https://us.norton.com/core.
[3] NS-3: a discrete-event network simulator. https://www.nsnam.org/.
[4] Remote control: LoRa offers a cheaper link to the IoT. http://www.reuters.com/article/us-tech-communications-lora-idUSKCN10E2TE.
[5] Appenzeller *et al.* 2004. Sizing Router Buffers. *SIGCOMM CCR* 281–292.
[6] Beheshti *et al.* 2008. Experimental Study of Router Buffer Sizing. In *IMC*. 197–210.
[7] Beheshti *et al.* 2008. Obtaining High Throughput in Networks with Tiny Buffers. In *Workshop on QoS*. 65–69.
[8] Beheshti *et al.* 2006. Buffer sizing in all-optical packet switches. In *Optical Fiber Communication Conference*. 3 pp.–.
[9] Dhamdhere *et al.* 2005. Buffer sizing for congested Internet links. In *Joint Conference of the IEEE Comp. and Comm. Societies*.1072–1083.
[10] Enachescu *et al.* 2005. Part III: Routers with Very Small Buffers. *SIGCOMM CCR* 35, 3, 83–90.
[11] Gorinsky *et al.* 2007. Simulation Perspectives on Link Buffer Sizing. *SIMULATION* 83, 3 (2007), 245–257.
[12] Kurose *et al.* 2016. *Computer networking: a top-down approach*. Vol. 7.
[13] R. Morris. 1997. TCP Behavior with Many Flows. In *ICNP*. 205–211.
[14] R. Morris. 2000. Scalable TCP congestion control. In *INFOCOM*.
[15] Villamizar *et al.* 1994. High Performance TCP in ANSNET. *SIGCOMM CCR* 24, 5, 45–60.
[16] Wan *et al.* 2013. Power-aware cloud computing infrastructure for latency-sensitive internet-of-things services. In *UKSim*. 617–621.