No-more-unbounded-blocking queues: bounding transmission latencies in real-time edge computing

Gabriele Serra
Scuola Superiore Sant'Anna
Pisa, Italy
gabriele.serra@santannapisa.it

Pietro Fara
Scuola Superiore Sant'Anna
Pisa, Italy
pietro.fara@santannapisa.it

***Abstract*—Nowadays, with advancements in computing power and energy efficiency, embedded system platforms are becoming able to provide services that, previously, required the cloud to be served. Accordingly, the edge computing paradigm is becoming increasingly popular, as it allows, among other advantages, to foster security and privacy preservation by processing data at the origin. On the other hand, these systems demand predictability across the edge-to-cloud continuation. Regardless of the communication link used by the edge node, tasks send out data employing a transmission queue; for system-designer, analyzing the time behavior of a task becomes challenging when each task has to wait a variable amount of time to send a packet. This work presents a model to analyze the different sources of latency introduced when dealing with a communication interface. The mentioned model ensures that the data traffic does not exceed the transmission queue limit to avoid unbounded blocking on task execution.**

I. Introduction

Embedded systems are every day more pervasive in human-beings lives: they are used in any kind of environment, ranging from agriculture to transports; they fulfill evermore functions, of which some are related to the safety of environments or people. Further, with technology advancements, embedded system platforms are becoming more powerful and they are able to provide services that, until few years ago, required server to be accomplished. Therefore, the edge computing paradigm is becoming increasingly popular. Processing data closer to its origin drastically reduces the amount of data sent to the cloud and, therefore, facilitates real-time computation, reduces energy consumption (together with carbon footprint) and help preserving data privacy. However, when the functions performed by the edge-computing system need to interact with the external environment, the predictability across the edge-to-cloud continuum is a key requirement. In the meantime, the use of system resources must be kept efficient.

Especially when dealing with connected systems, the edge computing nodes are required to deliver a result within a predefined deadline; the result typically consist in a chunk of data to be transmitted across the network. Regardless the communication mechanism used by the node, commonly the sender task places the packet in a transmission queue. When the queue reaches the maximum queue size, the task must wait an unbounded amount of time to transmit the packet. Furthermore, the device transmission protocol takes some time to be performed. Indeed, as a consequence, the communication device introduces latencies between the edge node and the cloud. This makes particularly challenging to analyze the time behavior of the system. While it is not possible to bound the delay introduced by a complex network such as the global internet, it is interesting to analyze the type of latencies introduced by the transmission device and how these latencies must be accounted when analyzing the timing behavior of the edge node.

Contribution. In summary, this work makes the following contributions:

- It presents a detailed analysis of the latencies that a packet can experience during the transmission phase in an edge node
- It derives an analysis that prevent device transmission queue to be full, in order to bound the time required for a task to send data through the communication peripheral

Paper structure. The remainder of this paper is organized as follows. Section II reviews the related work. Section III presents the system by considering task model and communication assumptions. Section IV analyzes queueing effects and delays in inter-replica communications and Section V concludes the paper.

II. Related Work

In the last few years, edge computing has become more popular due to the increase in performance and the decrease in the size of microcontrollers and devices. A particular architecture, called Mobile Edge Computing (MEC), is gaining more attention from researchers, especially for the massive diffusion of Internet-of-things (IoT) devices. An overview of this new architecture [8] was presented in 2016. Based on MEC, many studies have been done to improve the overall performance of edge-to-cloud applications in terms of latency. Ren et al. [7] formulated an optimization problem to find the better solution for splitting the entire computation of a single task into two different parts: the first one to be executed at the edge and the second one on the cloud. Another work on latency optimization based on data compression was presented in 2018 [6]: they analyzed three different computation models: local compression, edge-cloud compression and partial compression offloading. In the last model, they formulated an optimization problem to find the optimal solution in terms of latency. Latency may also increase because a lot of computation is

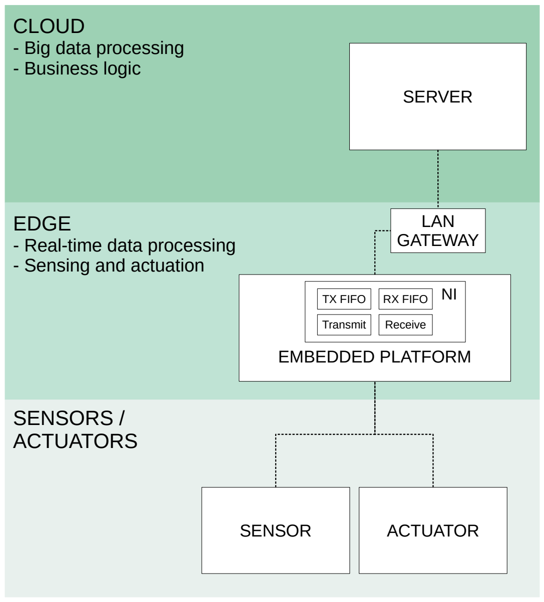

Fig. 1. An overview of the system architecture.

needed to provide services on mobile applications. In [4], Jia et al. proposed a heuristic offloading method for computation-intensive applications in MEC, taking into account the cloud service latencies and computation time and improving it with load-balancing techniques. In 2017, LAVEA platform [10] was designed to offload computation between clients and edge nodes and to let nearby edge nodes collaborate to provide low-latency video analytics at places closer to the users. Latency is also depending on how transmission queues are managed. In [1], Aamir et al. proposed MANET: a new scheme to improve packet queues management in terms of packet loss ratio. Another queue management mechanism, based on the division of the main queue into two sub-queues, has been developed in [9].

III. System model

This work focuses on analyzing delays introduced during data transmission from an edge node. As multi-core interference is outside the scope of this work, the edge node consists of an embedded system platform with a single processor. A set of n periodic tasks $\{\tau_1, \ldots, \tau_n\}$ is executed on the mentioned processor. Tasks are scheduled through a fixed-priority preemptive scheduling algorithm and activated all at the same time without any initial offset. Hence, each periodic task τ_i, is characterized by a worst-case execution time C_i, a release period T_i, and a relative deadline $D_i \leq T_i$. We denote with H the hyper-period among all periods of tasks τ_i.

The node is connected to a communication network CN. To be as general as possible, this work does not take into account a specific network link. The edge node exposes, on the CN, an interface capable of sending data and receiving it. From now on, we will denote the exposed interfaces of the node as the node-interface NI. Data transmission via the CN occurs by acting on a few memory-mapped device registers. An overview of the architecture of the system considered by our model is shown in Figure 1.

The NI provides an output (and respectively, an input) buffer organized as a first-in-first-out (FIFO) queue of q^{NI} elements, each sized b bytes. Accessing such registers consists of performing several writing and reading operations on the memory-mapped device registers. The minimum read/write rate to access such registers is indicated with β (bytes per time unit), while the maximum rate is denoted by β^{max}. Furthermore, the NI guarantees a minimum transmission rate on the link that, in this model, is indicated by α (bytes per time unit).

To send data out from NI, the content of a packet must be copied from the task memory to the device queue. Memory write times are generally shorter than read times; however, we denote with γ minimum read/write rate to access memory. The model considers a matching rate for read/write operations as it does not particularly affect the results of this work.

When the NI transmission queue is full, if a task wants to send data must wait until one of the slots in the queue becomes empty. For the sake of our model, we are not interested in analyzing how the NI operates when receiving data.

To summarize, when a task has to send x bytes out, the NI needs **(i)** at most x/γ time units to read the data from the task memory, **(ii)** at least x/β^{max} time units and at most x/β time units to copy the mentioned data into the NI queue and **(iii)** at most x/α time units to transmit such data into the communication link.

A task τ_i exchanges M_i data packets with a fixed size of b bytes. Note that not all the periodic tasks may produce data that must be sent over the network; hence M_i can also be null.

Data transmission is performed through the NI in a mutual-exclusive way. Thus, each task may have to acquire and release a lock before and after transmitting each packet. Our model adopts the immediate priority ceiling (IPC) locking protocol to avoid priority inversion phenomena. In the case in which all M_i are different from zero, NI is equivalent to a resource shared by all tasks under the IPC protocol. Therefore, the critical sections due to the NI access are practically non-preemptive.

We indicate the average amount of bytes transmitted through the NI as $\overline{B}(t)$.

IV. Transmission analysis

This section will analyze the type of latencies introduced by the NI, deriving a condition to ensure that the number of packets sent through the NI peripheral does not exceed the queue limit to avoid introducing unbounded blocking.

A. Latency modeling

Definition 1: Δ_{NI} indicates the amount of time that elapses since a packet is stored in the transmission queue by the sender task to the time the packet can be considered sent out and thus available for the subsequent node of the network (that, commonly, is the network edge router).

In the following subsection, the latency introduced by NI is studied employing a model commonly used to account for delays in routers network [2] [5]. Under this model, Δ_{NI} can be decomposed as a sum of four different components: propagation latency d_{prop}, transmission latency d_{trans}, processing latency d_{proc}, queuing latency d_{queue}.

Therefore, the total latency due to the NI packet transmission can be computed as:

$$\Delta_{NI} = d_{prop} + d_{trans} + d_{proc} + d_{queue}$$

The purpose of this work is not to provide an estimation of the latency Δ_{NI} but to demonstrate that, adopting our methodical analysis, that latency can be bounded. In order to show that for Δ_{NI} exists an upper bound Δ_{NI}^{max}, we have to analyze every single component. In the following subsections, we will investigate all the sources of latency characterizing each one.

1) Propagation latency: Once bits are pushed in the CN link, they need to propagate to the other end. Therefore, the propagation latency depends on the channel length and the signal propagation speed for the given medium. Commonly, propagation latency can be computed as the distance between devices divided by channel propagation speed. The signal propagation speed is affected by multiple factors. Although some specific wireless networks (e.g., wireless acoustic networks) may have a propagation latency up to several milliseconds, in the majority of cases, if the node is connected by means of a network cable or a wireless network, the latency due to the propagation speed lies in the order of nanoseconds. Accordingly, in the following sections, we will neglect the propagation latency.

2) Transmission latency: The transmission latency is an amount that represents the time required by the device to transmit a chunk of bytes into the link and depends on the device bandwidth. It is computed as the number of bytes to be pushed in the link over the transmission bandwidth. Given that each task exchanges data packets with a fixed size of b bytes, we may indicate d_{trans} as: $d_{trans} = \frac{b}{\alpha} = \sigma_d$ with σ_d being a constant factor to account for the transmission time demanded by each packet.

3) Processing latency: The processing latency represents the time required to read and write device registers and performing register shifts. It is fixed for each packet and does not depend on the packet length. Therefore we denote $d_{proc} = \sigma_o$ with σ_o being the maximum data-size-independent processing overhead.

4) Queueing latency: The queueing latency represents the amount of time each packet must wait in the queue to be transmitted in the link. When the queue is empty, and no other packet is being transmitted, the queueing latency will be null. Otherwise, a packet's queueing latency depends on the number of currently enqueued packets that wait for transmission.

Assuming that each packet transmission costs the equivalent of the transmission latency d_{trans} plus the processing latency d_{proc} , the $N-th$ packet in queue will wait $(N-1)\cdot(d_{trans} + d_{proc})$. Therefore, we can compute the maximum queueing latency as: $d_{queue} = (q^{\text{NI}} - 1) \cdot (\sigma_d + \sigma_o)$.

Provided all the mentioned latencies definition, the upper bound Δ_{NI}^{max} can be computed as: $\Delta_{NI}^{max} = q^{\text{NI}} \cdot (\sigma_o + \sigma_d)$

B. Queueing analysis

In our transmission model, we consider the NI queue as a finite queue. Therefore, when the queue is full, a task cannot push another packet into it, remaining blocked on the send operation. A task must perform the send operation on a queue with available free slots to avoid unbounded blocking.

Let's assume that the queue can host an infinite number of packets for the sake of proving the following lemma. Recalling the notation used above, that indicates the transmission bandwidth with α and the average amount of bytes transmitted through the NI as $\overline{B}(t)$, we define the transmission interface utilization average $\overline{U}^{\text{NI}}(t)$ as $\overline{U}^{\text{NI}}(t) = \frac{\overline{B}(t)}{\alpha}$

Lemma 1: The latency component caused by queuing d_{queue} can be bounded during the entire system service only if

$$U^{\text{NI}} = \lim_{t\to H} \overline{U}^{\text{NI}}(t) \le 1$$

Proof.

Being α a constant, if $\lim_{t\to H} \frac{\overline{B}(t)}{\alpha} > 1$, it means $\lim_{t\to H} \overline{B}(t) > \alpha$, therefore the average rate at which bytes arrives at the queue exceeds the rate at which the bytes can be transmitted by the NI device. Consequently, the queue will tend to grow with no bound, making the queuing latency infinite.

□

In our subsequent timing analysis, we assume as a necessary condition, that U^{NI} must be less or equal than 1. Given that all tasks τ_i are periodic, U^{NI} can be computed as $U^{\text{NI}} = \sum_{i=1}^{n}(M_i b)/T_i$. When the condition mentioned above holds, the nature of $\overline{U}^{\text{NI}}(t)$ impacts the latency. When packets arrive every $\frac{b}{\alpha}$, then each packet finds the queue empty. However, packets can arrive simultaneously, in bursts. Hence, U^{NI} cannot be used to characterize the queuing latency fully. We must be sure that the actual packet burst does not overpass the available queue slots.

We begin by bounding the amount of data sent within arbitrary time windows.

Lemma 2: In any time window of length t, the tasks can provide in the NI queue at most $g(t)$ bytes of data, where

$$g(t) = \min\left\{\sum_{i=1}^{n}\left\lceil\frac{t+T_i}{T_i}\right\rceil M_i b,\ \beta^{max} t\right\}. \qquad (1)$$

Proof.

A periodic task τ_i, in any time window of length t, can release at most $\lceil (t+T_i)/T_i \rceil$ jobs (e.g., see [3]). Each job of the tasks sends at most M_i packets, each of size b bytes. Hence the first term in the minimum of Eq. (1). Note that the amount of data the tasks can send within a time window is also limited by the maximum rate with which the NI queue can be filled, which is given by β^{max}. The lemma follows. □

The above lemma can then be used to derive a safe condition under which the NI queue is never full.

Lemma 3: No task can find the NI queue full if

$$\forall t > 0, \quad g(t) - \alpha t \leq q^{\text{NI}} \cdot b. \tag{2}$$

Proof.
Assume by contradiction that at a certain time instant t_1 a task finds a NI queue full. Let $t_0 < t_1$ be the latest time at which the NI queue has been empty and let $t = t_1 - t_0$. It holds that $(t_0, t_1]$ is an interval of length t in which the NI has always been busy with packets to transmit. Let $x(t)$ be the amount of bytes issued by the tasks to be provided in the NI queue in $(t_0, t_1]$. Note that during this interval the NI must have sent at least αt bytes: hence, if the queue is full at time t_1 it holds that $x(t) - \alpha t > q^{\text{NI}} \cdot b$.

By Lemma 2, in any time window of length t the cumulative amount of bytes provided in the NI queue is bounded by $g(t)$. Hence, $g(t) \geq x(t)$, which implies $g(t) - \alpha t > q^{\text{NI}} \cdot b$. This contradicts Eq. (2). Hence the lemma follows. □

Note that Lemma 3 does not consist in a practical test as any possible value of t shall be checked. This issue is solved below by limiting the test to a finite number of check-points.

Lemma 4: Lemma 3 holds also if $\forall t \in \Phi, \quad g(t) - \alpha t \leq q^{\text{NI}} \cdot b$, where

$$\Phi = \bigcup_{i=1}^{n} \{kT_i + \epsilon \leq t^*, k = 0, 1, 2, \ldots\} \cup \{\psi\} \tag{3}$$

with

$$t^* = \frac{2\sum_{i=1}^{n} M_i b}{\alpha - \sum_{i=1}^{n} \frac{M_i b}{T_i}} \tag{4}$$

$$\psi = \left\{ t \leq t^* \quad | \quad \sum_{i=1}^{n} \left\lceil \frac{t + T_i}{T_i} \right\rceil M_i b = \beta^{max} t \right\} \tag{5}$$

and $\epsilon > 0$ arbitrarily small.

Proof.
We prove the lemma by showing that function $g(t) - \alpha t$ can be maximal only for values $t \in \Phi$. First note that the minimum of two functions is upper bounded by the upper bound of one of the two functions. Let's denote with $G(t) = \sum_{i=1}^{n} \left(\frac{t+T_i}{T_i} + 1 \right) M_i b$, hence $g(t) \leq G(t)$. Denoting $m = \sum_{i=1}^{n} \frac{M_i b}{T_i}$ and $q = 2\sum_{i=1}^{n} M_i b$ we can write $G(t) = mt + q$.

Note that both $G(t)$ and αt are two lines with slope $U^{\text{NI}} = \sum_{i=1}^{n} (M_i b)/T_i$ and α, respectively. Recall that $\alpha > U^{\text{NI}}$ (see Lemma 1). Therefore $G(t)$ and αt intersect and, from their intersection on, we have $g(t) \leq G(t) \leq \alpha t$ and hence also $g(t) - \alpha t \leq 0$. The intersection occurs for the value t^* such that $G(t^*) = \alpha t^*$ and can be computed by solving the latter equality with respect to t^*,

$$mt^\star + q = \alpha t^\star, \quad t^\star = \frac{2\sum_{i=1}^{n} M_j b}{\alpha - \sum_{i=1}^{n} \frac{M_i b}{T_i}}$$

Hence getting the expression at the Eq. (4). Therefore, for values of $t > t^*$ function $g(t) - \alpha t$ cannot be maximal.

If $g(t) = \sum_{i=1}^{n} \left\lceil \frac{t+T_i}{T_i} \right\rceil M_i b$ note that function $g(t) - \alpha t$ can be maximal only for those values of t that correspond to a step of the ceiling term of $g(t)$. The values are of the form $t = kT_i + \epsilon$ with k being a non-negative integer and $\epsilon > 0$ arbitrarily small. Conversely, if $g(t) = \beta^{max} t$, being both the latter function and αt monotonic increasing, function $g(t) - \alpha t$ can be maximal only for those values of t for which at $t' = t + \epsilon$ (when $\alpha \leq \beta^{max}$) or $t' = t - \epsilon$ (when $\alpha > \beta^{max}$), with $\epsilon > 0$ arbitrarily small, it holds $g(t') \neq \beta^{max} t$. These values of t must be an intersection between the two components that define $g(t)$, which are those of the set ψ. Lemma follows. □

Lemma 4 provides a practical test to ensure that no task can find the NI queue full. Furthermore, Δ_{NI}^{max} provides a transmission bound to be used in the timing characterization of the system.

V. Conclusion and future work

In this article, we presented an analysis on the different type of latencies that can be introduced by the communication interface when a task want to send out data packets. Furthermore, we derived a model, an analytic condition and respective formal proofs to ensure that introduced latencies, especially regarding the queueing, are bounded. In the future, we are willing to extend this work including a fault model that takes into account the transmission error and error-recovery strategies also providing a response time analysis model for tasks. Further, we may investigate whether a different task scheduling model (e.g. a sporadic sender task) may introduce lower pessimism in the analysis.

References

[1] Muhammad Aamir and Mustafa A Zaidi. A buffer management scheme for packet queues in manet. *Tsinghua Science and Technology*, 18(6):543–553, 2013.

[2] Dimitri Bertsekas and Robert Gallager. *Data Networks (2nd Ed.)*. Prentice-Hall, Inc., USA, 1992.

[3] B. Brandenburg. Scheduling and locking in multiprocessor real-time operating systems. In *Ph.D. dissertation, The University of North Carolina at Chapel Hill*, 2011.

[4] Mike Jia, Jiannong Cao, and Lei Yang. Heuristic offloading of concurrent tasks for computation-intensive applications in mobile cloud computing. In *2014 IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, pages 352–357, 2014.

[5] J.F. Kurose and K.W. Ross. *Computer Networking: A Top-Down Approach*. Pearson Education, Limited, 2010.

[6] Jinke Ren, Guanding Yu, Yunlong Cai, and Yinghui He. Latency optimization for resource allocation in mobile-edge computation offloading. *IEEE Transactions on Wireless Communications*, 17(8):5506–5519, 2018.

[7] Jinke Ren, Guanding Yu, Yinghui He, and Geoffrey Ye Li. Collaborative cloud and edge computing for latency minimization. *IEEE Transactions on Vehicular Technology*, 68(5):5031–5044, 2019.

[8] Dario Sabella, Alessandro Vaillant, Pekka Kuure, Uwe Rauschenbach, and Fabio Giust. Mobile-edge computing architecture: The role of mec in the internet of things. *IEEE Consumer Electronics Magazine*, 5(4):84–91, 2016.

[9] Lak Sad. Parallelising reception and transmission in queues of secondary users. *International Journal of Electrical and Computer Engineering*, 9(4):3221, 2019.

[10] Shanhe Yi, Zijiang Hao, Qingyang Zhang, Quan Zhang, Weisong Shi, and Qun Li. Lavea: Latency-aware video analytics on edge computing platform. In *Proceedings of the Second ACM/IEEE Symposium on Edge Computing*, pages 1–13, 2017.