RAGE 2022, July 10th, San Francisco

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

Gabriele Serra and Pietro Fara
Scuola Superiore Sant'Anna, Pisa

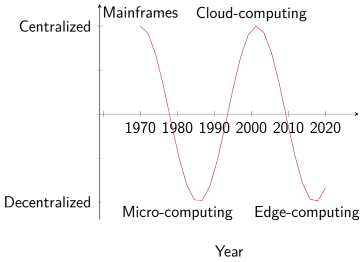
Introduction & context

Advancement in embedded computing

SoCs employed in modern edge devices can be quite powerful:

- They can be host a multi core processor
- They can include hardware accelerators

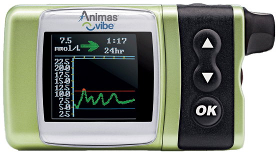
History is Cyclical



Decentralized computing is becoming popular (again):

- Modern SoC allows to perform computations on the device
- Reduces energy and network bandwidth consumption
- Fosters security and privacy preservation

Benefit for several applications



Processing data at the origin is a requirement for some applications:

- Applications that require significant bandwidth
  - i.e. smart surveillance systems
- Applications that works in harsh environments
  - i.e. agriculture systems
- Applications that cannot tolerate latencies
  - i.e. healthcare

Issues

Edge devices are connected systems

Edge computing nodes typically must deliver a result across the network within a **predefined deadline**.

Issues

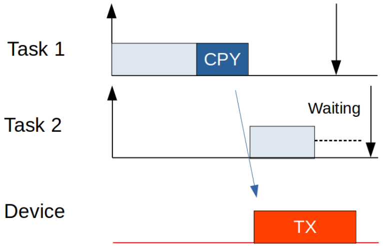
Edge devices are connected systems

Edge computing nodes typically must deliver a result across the network within a **predefined deadline**.

Transmissions over network take time

It is particularly challenging to analyze the time behavior of the system, especially when dealing with transmission queues

Example:



Example: system with 2 tasks, sending packets outside.

The `Task 2` is going to miss its deadline!

Our approach

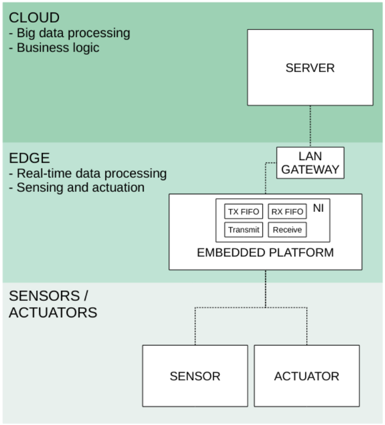
Our work tries to tackle this issue in two steps:

- **analyzing** the latency introduced by the transmission interface
- **modeling** an analytic condition to ensure that introduced latencies are always bounded

System model

System architecture

Architecture considered



The node is connected to a communication network `CN`. It exposes a network interface `NI`.

The `NI` provides an I/O buffer organized as a first-in-first-out (FIFO) queue

Transmission analysis

Latency analysis

The latency introduced by `NI` is studied decomposing it as a sum of four different components:

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

where:

- d_{prop} is **propagation latency**
- d_{trans} is **transmission latency**
- d_{proc} is **processing latency**
- d_{queue} is **queueing latency**

Latency analysis

The latency introduced by `NI` is studied decomposing it as a sum of four different components:

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

where:

- d_{prop} is **propagation latency**
- d_{trans} is **transmission latency**
- d_{proc} is **processing latency**
- d_{queue} is **queueing latency**

The purpose of our methodical analysis is to demonstrate that the latency Δ_{NI} can be bounded.

Queueing analysis

The transmission queue is finite and queueing latency depends on the number of enqueued packets. When the queue is full, a task:

- cannot push another packet into it;
- remains blocked on the send operation.

To avoid **unbounded blocking** the queue must have at least one free slot.

To enforce that condition you have to analyze the quantity of packet send by all tasks. Packets are not arriving regularly but in bursts.

Determine the packet burst

To determine packet burst, we need the number of packets each task send and the job instances that can be executed in the chosen window.

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

in which

- a periodic task τ_i can release at most $\lceil (t + T_i)/T_i \rceil$ jobs;
- each job sends at most M_i packets, each of size b bytes.

The amount of data the tasks can send is also limited by the maximum memory rate which is given by β^{max}.

Determine the packet burst

To determine packet burst, we need the number of packets each task send and the job instances that can be executed in the chosen window.

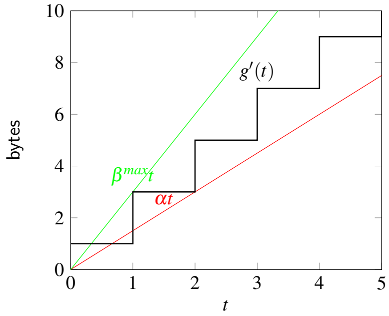
$$g(t) = \min\left\{g'(t),\ \beta^{max} t\right\}$$

in which

- a periodic task τ_i can release at most $\lceil (t + T_i)/T_i \rceil$ jobs;
- each job sends at most M_i packets, each of size b bytes.

The amount of data the tasks can send is also limited by the maximum memory rate which is given by β^{max}.

Determine packet burst



$$g(t) = \min\{g'(t),\ \beta^{max}t\}$$

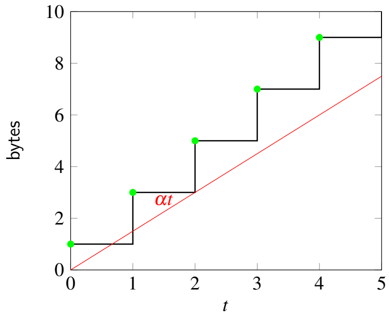
$g(t)$ is number of bytes sent by all tasks.

αt is the number of bytes transmissible by the interface

$$\forall t \geq 0,\ g(t) - \alpha t \leq q^{NI} \cdot b$$

For any instant of time, the distance between $g(t)$ and αt represents the number of bytes in the queue.

Restricting testing set



We need to check only the points in which the function is maximal.

The distance between curves in those points must be less than the size of the queue $q^{NI} \cdot b$

Conclusions & future directions

Contributions

Main contributions of our work:

- 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
- we derived a **model**, an analytic condition and respective formal proofs to ensure that introduced latencies, especially regarding the queueing, are bounded

Future directions

Include a fault model

We are willing to extend this work including a fault model that takes into account the transmission error and error-recovery strategies.

Task timing analysis

Our objective is to provide a response time analysis model for tasks.

Investigate different task model

We may investigate whether a different task scheduling model (e.g. a sporadic sender task) may introduce lower pessimism in the analysis.

Thank you. Questions?