Managing Real-time Mobile Applications in Reconfigurable Cloud-to-Things Systems

Marco Pettorali, Francesca Righetti, Carlo Vallati, and Giuseppe Anastasi

Dept. of Information Engineering, University of Pisa, Pisa, Italy, {name.surname}@unipi.it

***Abstract*—In future reconfigurable IoT systems, real-time mobile applications characterized by different requirements will coexist, and will be dynamically introduced or removed. This asks for dynamic management mechanisms to ensure the requirements of different real-time mobile applications, even when the system configuration changes over time. In this paper, we propose DJ-NECORA, an online algorithm for the joint allocation of networking and computing resources in the Cloud-to-Things Continuum that is capable of guaranteeing the requirements of real-time applications and efficiently managing possible changes in the system configuration. We evaluate DJ-NECORA through simulation in a realistic scenario. The results show that DJ-NECORA effectively handles application dynamics and, in some scenarios, outperforms offline resource allocation solutions by supporting 14% more nodes.**

***Index Terms*—Cloud-to-Things Continuum, Real-time Apps, Mobility, System Reconfigurability, Dynamic Resource Allocation**

I. Introduction

The rapid proliferation of IoT devices and the growing demand for real-time IoT applications in various domains, such as smart cities, Industry 4.0, and smart healthcare, have driven a shift from Cloud Computing to Edge Computing. Edge nodes can be located at different levels between IoT devices (Things) and the Cloud, giving rise to the so called *Cloud-to-Things Continuum (C2TC)*. Many real-time IoT applications involve Mobile Nodes (MNs), like robots, autonomous vehicles, and wearable devices, to ensure seamless operation of processes.

In future reconfigurable IoT systems, many real-time applications, with different characteristics and QoS requirements, will co-exist. In addition, applications will be dynamically introduced and removed. Guaranteeing the stringent requirements of real-time applications, in the presence of node mobility and heterogeneous resources available in C2TC, is already a challenging task [1]. Facing changes in the system configuration, without service interruptions, adds more complexity to the problem and requires dynamic mechanisms.

In this paper, we propose DJ-NECORA (Dynamic Joint NEtwork and COmputing Resource Allocation), an online algorithm for joint networking and computing resource allocation in C2TC. It ensures real-time application requirements while efficiently handling system reconfiguration. We evaluated DJ-NECORA through simulations considering a specific use case with eight different real-time applications. The results show that DJ-NECORA effectively manages application reconfigurations, ensuring high adaptability. In some scenarios, it outperforms a static optimal approach by supporting up to 14% more MNs.

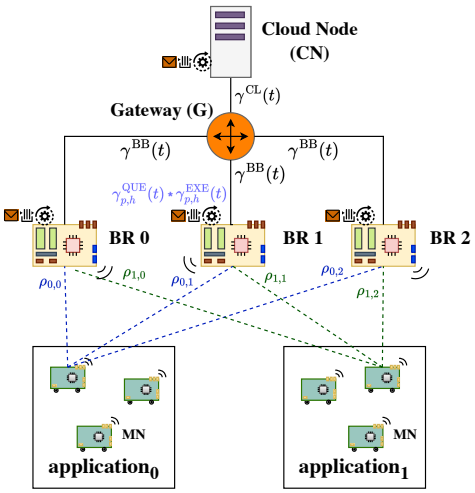

Fig. 1: Reference architecture of the C2TC system

II. System Model

Fig. 1 shows the system architecture, which is distributed across multiple layers and consists of Mobile Nodes (MNs), Edge Nodes (ENs), comprising a set of Border Routers (BRs) and a Gateway (G) enabling communication between BRs, and Cloud Nodes (CNs). We assume that MNs lack computing capabilities; therefore, all application processing is carried out at a BR or CN. MNs move within an *Area of Interest (AoI)*, where they collect and transmit data to BRs via wireless links (*WL*). BRs communicate with G through a wired backbone (*BB*), which in turn connects to CNs via wired links (*CL*). Each application A_p is characterized by: (i) process p to be executed on a host h; (ii) M_p: memory required for p; (iii) message generation period T_p; (iv) maximum tolerable delay d_p; (v) minimum required on-time ratio r_p; (vi) number of MNs m_p associated with the application.

Each network link is modeled using a delay probability distribution $\gamma^{link}_{p,b}(t)$, where $link$ indicates the type of link (*WL*, *BB* or *CL*). Given these distributions, the end-to-end communication delay distribution $\gamma^{\text{COM}}_{p,h}(t)$ for a message generated by an MN and transmitted to a host h, can be computed by convolving the distributions of the links along the path from the MN to the host.

To model the processing time distribution, we introduce the CPU share $\lambda \in [0, 1]$ to represent the CPU fraction allocated to a process on a host, and we model the execution time probability distribution $\gamma^{\text{EXE}}_{p,h}(t, \lambda)$ as a function of λ. In addition, packets can experience queuing delays before being

processed. The queuing time is modeled through probability $\gamma^{\text{QUE}}_{p,h,m}(t)$ [1]. Hence, the total end-to-end delay $\gamma_{p,h}(t,\lambda,m)$ for an application A_p with m MNs, running process p on host h with CPU share λ, is given by

$$\gamma_{p,h}(t,\lambda,m) = \gamma^{COM}_{p,h}(t) * \gamma^{QUE}_{p,h,m}(t) * \gamma^{EXE}_{p,h}(t,\lambda) \quad (1)$$

III. Dynamic Resource Allocation

The DJ-NECORA Algorithm is described in [2]. We provide below some design principles. Each application a_p, has two QoS requirements, namely a *maximum tolerated delay* d_p and a *minimum on-time ratio* r_p, that must be satisfied. To ensure these constraints, we define the function $\texttt{MinCPUShare}(p,h,m)$, which computes the minimum CPU share that host h must allocate to the application to satisfy both constraints. If host h cannot meet the QoS requirements, $\texttt{MinCPUShare}(p,h,m)$ returns a value greater than 1.

When a process requires more CPU resources than a single host can provide, its MNs can be divided into multiple groups, each assigned to a different host. The allocation of MNs among different hosts follows a *splitting policy*. We consider the following three splitting policies: *No Splitting*: MNs of the same process are allocated on a single host; *Lazy Splitting*: MNs of the same process are allocated on a single host until its CPU capacity allows; then, incoming MNs will be assigned to the next available host; *Greedy Splitting*: MNs of the same process are allocated, individually, across multiple hosts. Our algorithm also accounts for memory usage. When process p is assigned to host h, its memory usage increases by M_p. If the amount of requested memory exceeds the currently available memory on h, p will not be allocated on that host.

When allocating a process, multiple hosts may meet its QoS requirements. In such cases, a *selection policy* determines the most suitable host. We consider the following well-known policies: (**i**) *First Fit*; (**ii**) *Next Fit*; (**iii**) *Best Fit*); (**iv**) *Worst Fit*; and (**v**) *Random Fit*.

IV. Simulation Results

We simulated DJ-NECORA in a scenario with a deployment area of 100x200 meters, three walls and an obstacle measuring 50x25 meters located in the bottom-left corner. The area is covered by six BRs. MNs move along predefined linear paths, such as robots following a track. The number and placement of BRs were determined to provide complete coverage of the area using genetic algorithm [1].

Fig. 2 compares static vs. dynamic allocation with different selection policies. J-NECORA assumes global system knowledge and provides an optimal allocation of MNs to cloud/edge nodes. DJ-NECORA allocates slightly fewer MNs than J-NECORA, due to the lack of a global view. The difference is minimal, showing the effectiveness of DJ-NECORA.

Fig. 3 considers a situation where MNs join dynamically (we use Lazy Splitting in these experiments, as it proved to be more efficient [2]). Initially, only a fraction of MNs is allocated, while the rest arrive randomly over time. The initial fraction I varies from 0% (no MN initially allocated) to 100% (all MNs allocated from the start). The initial fraction

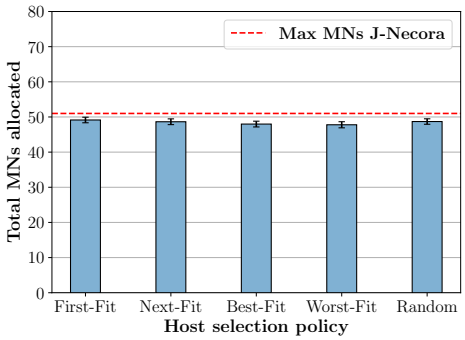

Fig. 2: Total MNs allocated by DJ-NECORA and J-NECORA

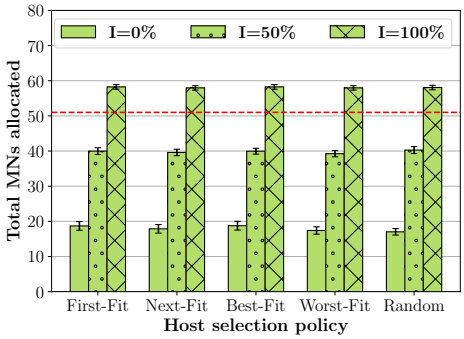

Fig. 3: Impact of the initial fraction of MNs allocated to processes on the total number of MNs allocated. Lazy Splitting

significantly influences the total number of MNs accommodated. When $I = 0\%$, DJ-NECORA allocates fewer MNs (around 18) compared to J-NECORA (51), due to the lack of a global view and the higher likelihood of multiple splits caused by the individual arrival of MNs. When I increases, DJ-NECORA is able to allocate more MNs, gradually approaching J-NECORA when $I = 50\%$ (around 40) and even surpassing J-NECORA when $I = 100\%$ (around 58 MNs, i.e.,$\sim$ 14%). This highlights the effectiveness of the splitting approach. By distributing MNs across multiple hosts, DJ-NECORA overcomes the constraints of J-NECORA, resulting in more efficient resource allocation.

V. Conclusions

In this paper, we have presented DJ-NECORA an algorithm for the dynamic allocation of resources in C2TC, designed for real-time mobile applications, in reconfigurable systems, where new applications are added or removed at runtime and MNs may join or leave dynamically. The results show that DJ-NECORA can achieve better performance than a static approach. As future work, we plan to evaluate DJ-NECORA in scenarios with different mobility patterns and across different wireless technologies (e.g., 5G, WiFi, and LoRaWAN).

References

[1] M. Pettorali, F. Righetti, C. Vallati, S. K. Das, and G. Anastasi, "J-necora: A framework for optimal resource allocation in cloud-edge-things continuum for industrial applications with mobile nodes," *IEEE Internet of Things Journal*, 2025.

[2] ——, "Dynamic Resource Allocation in Cloud-to-Things Continuum for Real-Time IoT Applications," in *IEEE International Workshop on Smart Service Systems (SMARTSYS 2025)*, Cork, Ireland, June 2025.