Optimizing Resource Reuse in the Web of Things

Ruben Gomes 0000-0002-0562-2955 and Noélia Correia 0000-0001-7051-7193
CEOT, Faculty of Science and Technology
University of Algarve
Faro, Portugal
{rdgomes, ncorreia}@ualg.pt

2024 IEEE URUCON | 979-8-3503-5538-3/24/$31.00 ©2024 IEEE | DOI: 10.1109/URUCON63440.2024.10850112

***Abstract*— The Web of Things enables collecting vast amounts of data about the environment and sharing them over the Internet. Its popularity, however, has brought challenges under constrained environments, such as an increase in the number of connected devices, and a consequent additional consumption of resources, leading to the deterioration of communications and application response times. By discovering and reusing semantically equivalent Things among applications, resources may be spared and responsiveness improved. In this article, an optimization model is developed to address this issue, and both fair and unfair goals are used for performance comparison over multiple scenarios, with different levels of network connectivity, Thing equivalence and placement. Results show that the interplay between network connectivity and equivalence ratio is determinant for responsiveness improvement, and both parameters can be used for an adequate planning of which Thing hosts to reuse. Also, ensuring a fair responsiveness among applications imposes fewer reuses, particularly in low connectivity and equivalence conditions, than using unfair criteria, where some applications can benefit from high responsiveness while others don't. When a high equivalence ratio is coupled with a low connectivity degree, reuses tend to compound into fewer hosts, independently of fairness.**

***Index Terms*—Internet of Things, Web of Things, Reusability.**

I. Introduction

The Internet of Things (IoT) encompasses many types of connected devices, both physical and virtual. Physical devices are typically equipped with sensors which allow them to gather and share information about their surroundings autonomously, while virtual devices are software-based representations that simulate the behavior of services or devices. Its impact spans various industries, including manufacturing, transportation, healthcare, and agriculture [1].

The Web of Things (WoT) builds on the IoT by connecting devices over the Internet using standard Web protocols. A key component in the WoT is the Web Thing Model, which provides standardized information about types of physical or virtual devices (Things). Other standards include the WoT Architecture, Thing Description, Scripting API, and Binding Templates [2]. Unlike IoT, which operates at the network layer, WoT works at the application layer [3].

While the rapid increase in the number of Things is expanding applications' scope, there is a growing challenge for managing resource constraints in IoT, since Things are commonly limited in energy, memory and processing power. The encapsulated nature of Things, however, promotes reusability which may be used to reduce resource consumption [4]. This means applications should be designed from the ground up with efficient resource usage in mind, by leveraging existing resources instead of spawning additional processes [5].

Research Question: How can the responsiveness of applications be optimized in constrained environments, such as the WoT, by taking advantage of Things' reusability?

The following sections are organized as follows. Section II introduces the topic of semantic discovery and Thing reuse in the Web of Things. In Section III, the equitability of reuse processes distribution is explained. Our optimization problem is modelled in Section IV, and its results are analyzed in Section V. Lastly, conclusions and future work are detailed in Section VI.

II. Semantic Discovery and Thing Reuse in WoT

The abundance of available services to choose from, poses a challenge for users and applications seeking the right service. On top of that, the diversity of IoT devices and WoT Things with varying capabilities makes resource discovery complex as well. Different devices use various representations for their resources, hindering seamless discovery. Additionally, IoT devices often operate under resource constraints, requiring lightweight solutions [6].

A. Semantic-Based Resource Discovery

By adding meaning to resource descriptions, semantic interoperability enhances their information exchange and machine interpretability, along with resource discovery. This facilitates precise matching between user requests and available resources. By describing resources semantically, the gap between heterogeneous devices and applications is bridged [7].

B. Thing Equivalence and Reuse

Semantic Discovery enables the identification of equivalent Things across different devices. In this work, we use the terms *Thing* and *operationalization* interchangeably, to make it clear that we view equivalent Things as alternative ways of operation that achieve the same goal. By finding these equivalences, we unlock the potential for resource reuse, which reduces network communication overhead, speeds up resource retrieval, and also minimizes redundant resource creation.

Fewer new resources mean less memory usage, reduced power consumption, and improved overall efficiency. As the IoT ecosystem grows, reusing existing resources becomes essential for scalability [8].

This work was supported by Fundação para a Ciência e Tecnologia (FCT) through the Center for Electronics, Optoelectronics, and Telecommunications (CEOT) under projects UIDB/00631/2020 CEOT BASE and UIDP/00631/2020 CEOT PROGRAMÁTICO, and grant UI/BD/152864/2022.

Related work includes using Quality of Service (QoS) parameters as criteria for optimization algorithms to select services in web service compositions [9] [10], although such studies were not performed in the context of WoT, nor did they take fairness into consideration in their models.

III. Equitability of Reuse Processes Distribution

In traditional optimization models, metrics like response time, queue length, throughput, or cost are used, and fairness is often overlooked. While efficiency is important, considering fairness ensures equitable treatment and better overall system performance. More clearly:

1) Unfair Optimization Models: Sometimes, an unfair objective function is used to minimize communication delay. This approach prioritizes overall delay reduction, even if it disproportionately affects specific users or connections. Strategic decisions or severe resource limitations may drive this choice. However, it can lead to unequal treatment and impact user experience. Previous works have attempted a more balanced approach, allowing unfairness to reach a predefined limit [11].

2) Fair Optimization Models: In systems with multiple entities, fairness can be important. Fair optimization models prevent any participant from obtaining resources at the expense of others. These models represent inequality-averse optimization. In this work, we use the MinMax fairness principle, of minimizing the maximum cost in network traffic [12].

IV. Problem Formulation

By formulating the Thing reuse problem as a mixed integer linear programming (MILP) model, we aim to optimize resource usage and communication times.

A. Motivation

In the WoT, reusing resources promotes efficiency and cost-effectiveness, by leveraging existing technologies and designs, which enables cheaper and faster development of new applications. The one-way delay (OWD) is an important metric we consider for QoS of the network traffic, since it reflects metrics such as the uplink and downlink transmission rates (physical capacity), throughput (application transmission), packet loss, availability, and also energy consumption in constrained devices. By minimizing message OWD within or between applications, the responsiveness of the system can be improved. The reuse of Things can reduce OWD by taking advantage of other applications, whose Things may be closer to the current application's controller. If fairness is taken into account then the reuse of components does not disproportionately benefit certain applications, in detriment of others. This consideration avoids the creation of congestion points in the network, by strategically distributing load and ensuring redundancy.

B. Mathematical Formulation

Given a network topology, let us summarize the known information that defines a scenario and the ongoing deployment:

- $\mathcal{C}$ Set of application controllers.
- $\mathcal{O}$ Set of all operationalizations.
- $\mathcal{O}^c$ Set of operationalizations of controller $c \in \mathcal{C}$.
- $r_{o_j}^{o_i}$ One if operationalization o_i is equivalent to o_j, i.e., is able to replace it, where $o_i, o_j \in \mathcal{O}$.
- $C_o^{\mathrm{T},c}$ Transit cost from controller $c \in \mathcal{C}$ to $o \in \mathcal{O}^c$.
- $\mathcal{S}$ Set of hosted operationalization batches, where each may be a single or multiple operationalizations, of a size b. This size is used when accounting for waiting times; $\mathcal{S} = \{0, 1, 2, ..., \frac{|\mathcal{O}|}{b}\}$.
- w^s Average waiting time in the system (i.e., queue and service times), considering a specific arrival rate and service rate, for batch $s \in \mathcal{S}$.

To outline a solvable optimization model, in this case a MILP model that finds the best set of replacement actions (i.e. reusing an operationalization while removing another), the following variables must be defined:

- $\alpha_{o_j}^{o_i}$ One if operationalization o_i has replaced o_j, where o_i, $o_j \in \mathcal{O}$, and $o_j \neq o_i$; zero otherwise.
- ϱ_o One if $o \in \mathcal{O}$ has been removed (due to the reuse of another operationalization); zero otherwise.
- $\delta_{o_j}^{o_i}$ One if the final operationalization replacing o_i is o_j; zero otherwise. The final reused operationalization is at the root of the replacements tree.
- $\lambda_{o_j}^{o_s,o_i}$ One if the replacement $\alpha_{o_j}^{o_i}$ exists as a node in the replacements tree, and that node is in the path from the source operationalization, o_s, to its final operationalization; zero otherwise.
- $\upsilon_{o_j}^{+,s}$ One if the number of reuses of $o_j \in \mathcal{O}$, matches batch $s \in \mathcal{S}$ or is higher; zero otherwise.
- $\upsilon_{o_j}^{-,s}$ One if the number of reuses of $o_j \in \mathcal{O}$, matches batch $s \in \mathcal{S}$ or is lower; zero otherwise.
- $\upsilon_{o_j}^{s}$ One if the average waiting time in the system, w^s, to be considered for $o_j \in \mathcal{O}$ is identified by $s \in \mathcal{S}$; zero otherwise.
- Φ_o^c OWD from controller $c \in \mathcal{C}$ to $o \in \mathcal{O}^c$.
- Φ Highest overall OWD.

1) Objective Function:

– Unfair approach:

$$\text{Minimize} \sum_{\{c \in \mathcal{C}\}} \sum_{\{o \in \mathcal{O}^c\}} \Phi_o^c \tag{1}$$

– Fair approach:

$$\text{Minimize } \Phi \tag{2}$$

The impact of using one criterion or another will be analyzed in Section V where their impact on the use of replacements will be measured.

2) Constraints:

– Find operationalization pairs to perform replacements:

$$\alpha_{o_j}^{o_i} \leq r_{o_j}^{o_i}, \forall o_i, o_j : o_j \neq o_i \in \mathcal{O} \tag{3}$$

$$\sum_{\{o_i \in \mathcal{O} : o_j \neq o_i\}} \alpha_{o_j}^{o_i} \leq 1, \forall o_j \in \mathcal{O} \tag{4}$$

Constraints (3) state that operationalization replacements can only happen for equivalent operationalizations, while Constraints (4) state that an operationalization can only be replaced by a single other operationalization.

– Build replacement tree using flow conservation law to find final reuses:

$$\varrho_{o_j} \geq \alpha_{o_j}^{o_i}, \forall o_i, o_j : o_j \neq o_i \in \mathcal{O} \tag{5}$$

$$\sum_{\{o_j \in \mathcal{O}: o_j \neq o_i\}} \lambda_{o_i}^{o_s,o_j} - \sum_{\{o_j \in \mathcal{O}: o_j \neq o_i\}} \lambda_{o_j}^{o_s,o_i} = \\ = \begin{cases} \varrho_{o_s}, & if\ o_i = o_s \\ -\delta_{o_i}^{o_s}, & otherwise \end{cases}, \forall o_s, o_i \in \mathcal{O} \tag{6}$$

$$\lambda_{o_j}^{o_s,o_i} \leq \alpha_{o_j}^{o_i}, \forall o_s, o_i, o_j : o_j \neq o_i \in \mathcal{O} \tag{7}$$

$$\delta_{o_i}^{o_i} = 1 - \sum_{\{o_d \in \mathcal{O}: o_d \neq o_i\}} \delta_{o_d}^{o_i}, \forall o_i \in \mathcal{O} \tag{8}$$

Constraints (5) determine if an operationalization o_j has been removed due to a replacement; this is used to build the flow conservation law in Constraints (6). Constraints (7) make the flow conservation law follow the replacements tree by using only valid replacements, while Constraints (8) ensure that there is a final placement defined for each operationalization, where either it is replaced due to reuse or it does not move.

– For each operationalization, find the batch $s \in \mathcal{S}$; this quantity impacts the queue average waiting time to be considered:

$$\upsilon_{o_j}^{+,s} \geq \frac{\frac{1}{b}\sum_{\{o_i \in \mathcal{O}\}} \delta_{o_j}^{o_i} - s}{|\mathcal{S}| + 1} + \Theta, \forall o_j \in \mathcal{O}, \forall s \in \mathcal{S} \backslash \{0\} \tag{9}$$

$$\upsilon_{o_j}^{-,s} \geq \frac{s - \frac{1}{b}\sum_{\{o_i \in \mathcal{O}\}} \delta_{o_j}^{o_i}}{|\mathcal{S}| + 1} + \Theta, \forall o_j \in \mathcal{O}, \forall s \in \mathcal{S} \backslash \{0\} \tag{10}$$

where Θ is a relatively small value, e.g. $\frac{1}{|\mathcal{S}|^2}$. These Constraints determine if the batch is higher or equal (9), and lower or equal (10), to s. The exact batch to be considered is determined by the following two constraints:

$$\upsilon_{o_j}^{s} \geq \begin{cases} 1 - \sum_{\{o_i \in \mathcal{O}\}} \delta_{o_j}^{o_i}, & if\ s = 0 \\ \upsilon_{o_j}^{+,s} + \upsilon_{o_j}^{-,s} - 1, & otherwise \end{cases}, \\ , \forall s \in \mathcal{S}, \forall o_j \in \mathcal{O} \tag{11}$$

$$\sum_{\{s \in \mathcal{S}\}} \upsilon_{o_j}^{s} = 1, \forall o_j \in \mathcal{O} \tag{12}$$

– One-way delay (OWD):

$$\Phi_{o_i}^{c} \geq C_{o_d}^{\mathrm{T},c} \times \delta_{o_d}^{o_i} + \sum_{\{s \in \mathcal{S}\}} \upsilon_{o_d}^{s} \times w^{s} - (1 - \delta_{o_d}^{o_i}) \times \Delta, \\ , \forall c \in \mathcal{C}, \forall o_i \in \mathcal{O}^{c}, \forall o_d \in \mathcal{O} \tag{13}$$

where Δ is a big value. Constraints (13) account for both transit time and waiting time in the system, for all operationalizations associated with a controller. Constraints (14) find the highest overall OWD.

$$\Phi \geq \Phi_{o}^{c}, \forall c \in \mathcal{C}, \forall o \in \mathcal{O}^{c} \tag{14}$$

– Non-negativity assignment to variables:

$$\alpha_{o_j}^{o_i}, \varrho_{o}, \delta_{o_j}^{o_i}, \lambda_{o_j}^{o_s,o_i}, \upsilon_{o_j}^{+,s}, \upsilon_{o_j}^{-,s}, \upsilon_{o_i}^{s} \in \{0,1\}; \Phi_{o}^{c}, \Phi \geq 0 \tag{15}$$

TABLE I
SCENARIOS AND PARAMETERS

Scenario	Connectivity	Equivalence
A	0.2	0.1
B	0.2	0.6
C	0.3	0.1
D	0.3	0.6

TABLE II
OPTIMIZATION RESULTS

Scenario	Max OWD (Fair)	Sum OWD (Unfair)	Total Reuses (Fair)	Total Reuses (Unfair)
A	2.69	35.44	0	5
B	1.60	25.57	15	15
C	2.69	35.44	1	6
D	1.10	17.60	15	15

V. ANALYSIS OF RESULTS

To evaluate our model performance, results are gathered by defining a set of scenarios and running them through the MILP model optimization. Then results are compared between the scenarios, with both fair and unfair objective functions.

A. Scenario Setup

Four scenarios are defined (see Table I). Each scenario has different configuration parameters, namely the network connectivity degree[1], and Thing equivalence ratio[2].

Each scenario includes four applications, each with four Things distributed over a network with 100 hosts. Directed links between hosts, and the initial placement of Things are generated randomly. Following [14], two connectivity degrees are used: sparse (20%) and dense (30%); also, two equivalence ratios are used: low (10%) and high (60%) [9].

Scenarios with equal connectivity degree share the same topology among them. Scenarios with equal equivalence ratio use the same set of equivalences. To achieve results' consistency, all scenarios use the same base topology, to which connections are added according to connectivity degree. Using scenarios' datasets, the starting transit costs (time) are obtained for the shortest paths between all controllers and all Things.

The average waiting time in the system, for each operationalization, is calculated according to a single server M/D/1 queue, where $\lambda = 100$ and $\mu = 1$, for all possible batches, as these impact the arrival rate.

Using the information on transit costs, the average waiting time, and Thing equivalences, problem instances are generated and solved by IBM's ILOG CPLEX Interactive Optimizer 22.1.0.0, in order to find their optimal solutions. All simulations were executed on an 8-core Intel i7-6700 CPU @ 3.40 GHz, with 15.5 GiB of RAM.

B. Evaluation

The results for fair and unfair optimization models are shown in Table II and the replacement trees in Figure 1.

[1]Network connectivity degree is the probability of two nodes being connected.

[2]Thing equivalence ratio is the probability of two Things being equivalent, meaning one service can be replaced by another while keeping the remaining components unchanged [13].

1) Highest overall OWD and Sum of OWDs: The same analysis is valid when analyzing both the Highest overall OWD (Fair) and the Sum of OWDs (Unfair) metrics. Scenarios A and C have the highest OWD. According to Table I scenario A has a lower connectivity degree than scenario C, while both have a low equivalence ratio, which conditions the available options for reusability among Things. This limitation makes it difficult to take advantage of greater connectivity to minimize OWD. The best OWD is reached in scenario D where network connectivity and equivalence are both at high level.

2) Total Reuses per scenario (Fair vs Unfair): Scenarios A and C had the fewest reuses under fair criteria, with none in A. While the low count of replacements in these scenarios can be linked to low equivalence, it is also influenced by the enforcement of fairness (any replacement attempt is unable to minimize the highest OWD, given the lack of freedom due to a low equivalence ratio). When comparing the same scenarios to the unfair approach, it's clear that the latter takes more advantage of possible replacements to reach its goal, and thus leads to an increased number of replacements happening in both scenarios. As for scenarios B and D, the maximum number of possible replacements was executed both in fair and unfair approaches. This suggests that as the available options for reuse increase (with more equivalences), both fair and unfair criteria are able to take advantage of available replacements to reach their goal.

3) Fair vs unfair replacement trees per scenario: Analyzing the resulting replacement trees in Figure 1, scenarios with low equivalence ratio as A and C, especially when using a fair objective function, result in smaller trees with few or no replacements. On the other hand, for these scenarios, the use of an unfair approach results not only in trees with more replacements, but also with multiple final active operationalizations.

Scenarios B and D use a single final host operationalization, effectively placing every other operationalization into a single one. Given the high level of equivalence present in these scenarios, the optimization is able to find a node that provides the smallest OWD for all operationalizations and their controllers; it also causes these scenarios to generate the deepest trees, i.e. with more compounded replacements. Between these two scenarios though, B creates the deepest trees. This difference may be caused by the lower connectivity degree in scenario B when compared to D. The fewer alternative paths, and consequently fewer OWD costs to choose from, are likely funneling the replacements in-depth into fewer operationalizations. Thus, connectivity is also an issue to be taken into account.

VI. Conclusions and Future Work

This work set out to demonstrate how to improve applications' responsiveness in the WoT by leveraging modularity and reusability in Things. MILP models with fair and unfair criteria were created, for minimizing messages' OWD, in multiple scenarios with different network connectivity and Thing equivalence. The optimization output was analyzed for OWD behavior and replacement trees.

Results show that the interplay between network connectivity and equivalence ratio is crucial in planning Thing reuses. The fewest replacements occur when using fair criteria in more constrained environments with low equivalence, enabling a more homogeneous response time among applications. Conversely, unfair criteria generally execute more reuses, under any conditions, but unequally improves responsiveness, benefitting only some applications. The best OWD, independently of fairness, is achieved with high connectivity and equivalence levels. This increase in options enables both fair and unfair approaches to use more of the available replacements to minimize the OWD. Replacement trees show that while low equivalence ratio with unfair criteria create trees with multiple final active operationalizations, a high equivalence causes all operationalizations to reuse a single final one, independently of fairness. Low connectivity and high equivalence generate the deepest reuse trees, where the former promotes compounding reuses into fewer hosts.

Since it may not always be possible or convenient to mathematically model every relation between the variables of a problem, machine learning algorithms such as reinforcement learning may be explored in the future, to find solutions using an agent-based trial-and-error approach.

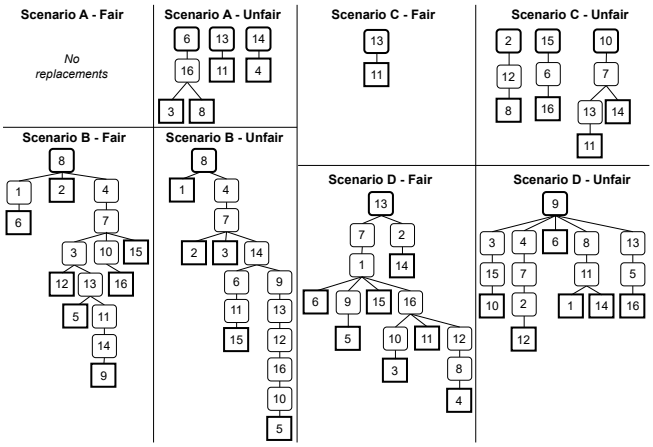

Fig. 1. Reuse trees for fair and unfair approaches. Bold squares: starting operationalizations; Rounded squares: intermediate reuses; Bold rounded squares: final reused operationalizations. Upper operationalizations are reused when replacing lower ones.

References

[1] A. Khanna and S. Kaur, "Internet of things (IoT), applications and challenges: a comprehensive review," *Wireless Personal Communications*, vol. 114, pp. 1687–1762, 2020.

[2] "W3C web of things (WoT) working group." [Online]. Available: https://www.w3.org/WoT/wg/

[3] A. García Mangas and F. J. Suárez Alonso, "Wotpy: A framework for web of things applications," *Computer Communications*, vol. 147, pp. 235–251, 2019.

[4] R. Gomes and N. Correia, "Enhancing dynamism of IoT service composition," in *Doctoral Conference on Computing, Electrical and Industrial Systems*. Springer, 2023, pp. 268–278.

[5] F. Pereira, R. Correia, P. Pinho, S. I. Lopes, and N. B. Carvalho, "Challenges in resource-constrained IoT devices: Energy and communication as critical success factors for future IoT deployment," *Sensors*, vol. 20, no. 22, p. 6420, 2020.

[6] F. Serena, M. Poveda-Villalón, and R. García-Castro, "Semantic discovery in the web of things," in *Current Trends in Web Engineering: ICWE 2017 International Workshops, Liquid Multi-Device Software and EnWoT, practi-O-web, NLPIT, SoWeMine, Rome, Italy, June 5-8, 2017, Revised Selected Papers*. Springer, 2018, pp. 19–31.

[7] P. Gomes *et al.*, "A semantic-based discovery service for the internet of things," *Journal of internet services and applications*, vol. 10, pp. 1–14, 2019.

[8] R. Mecibah, B. Djamaa, A. Yachir, and M. Aissani, "A scalable semantic resource discovery architecture for the internet of things," in *Advances in Computing Systems and Applications: Proceedings of the 3rd Conference on Computing Systems and Applications*. Springer, 2019, pp. 37–47.

[9] S. Chattopadhyay and A. Banerjee, "Qos-aware automatic web service composition with multiple objectives," *ACM Transactions on the Web (TWEB)*, vol. 14, no. 3, pp. 1–38, 2020.

[10] W.-H. Ai, Y.-X. Huang, H. Zhang, and N. Zhou, "Web services composition and optimizing algorithm based on qos," in *2008 4th International Conference on Wireless Communications, Networking and Mobile Computing*. IEEE, 2008, pp. 1–4.

[11] E. Angelelli, V. Morandi, and M. G. Speranza, "Minimizing the total travel time with limited unfairness in traffic networks," *Computers & Operations Research*, vol. 123, p. 105016, 2020.

[12] F. Barsotti and R. Koçer, "Minmax fairness: from rawlsian theory of justice to solution for algorithmic bias," *AI & SOCIETY*, pp. 1–14, 2022.

[13] P. Xiong, Y. Fan, and M. Zhou, "A petri net approach to analysis and composition of web services," *IEEE Transactions on Systems, Man, and Cybernetics-Part A: Systems and Humans*, vol. 40, no. 2, pp. 376–387, 2010.

[14] E. Al-Hawri, N. Correia, and A. Barradas, "Dag-coder: Directed acyclic graph-based network coding for reliable wireless sensor networks," *IEEE Access*, vol. 8, pp. 21 886–21 896, 2020.