International Journal on Software Tools for Technology Transfer (2023) 25:785–798
https://doi.org/10.1007/s10009-023-00728-9

FOUNDATIONS FOR MASTERING CHANGE

Special Section: Rigorous Engineering of Collective Adaptive Systems

Comparing perfomance abstractions for collective adaptive systems

Maurizio Murgia[1] · Riccardo Pinciroli[1] · Catia Trubiani[1] · Emilio Tuosto[1]

Accepted: 10 October 2023 / Published online: 2 November 2023
© The Author(s) 2023

Abstract

Non-functional properties of collective adaptive systems (CAS) are of paramount relevance practically in any application. This paper compares two recently proposed approaches to quantitative modelling that exploit different system abstractions: the first is based on generalised stochastic Petri nets, and the second is based on queueing networks. Through a case study involving autonomous robots, we analyse and discuss the relative merits of the approaches. This is done by considering three scenarios which differ on the architecture used to coordinate the distributed components. Our experimental results assess a high accuracy when comparing model-based performance analysis results derived from two different quantitative abstractions for CAS.

Keywords Behavioural specifications · Model-based performance predictions · Queueing Networks · Generalised Stochastic Petri Nets

1 Introduction

The past few years have witnessed the emergence of *collective adaptive systems* (CAS). These systems crop up in many application domains, spanning critical systems, smart cities, systems assisting humans during their working or daily live activities, etc. CAS consist of distributed computational agents acting in a cyber-physical context. Typically, these agents are replicated (namely, they exhibit the same behaviour). However, there are distinguished features telling CAS apart from traditional distributed systems: on the one hand, agents must be *autonomous* and *adaptive,* while, on the other hand, they *collectively* contribute to reach the overall goal of the system (a.k.a. *emergent behaviour*). Remarkably, reconciling autonomous and adaptive behaviour with the overall goal of CAS is challenging. In fact, the emergent behaviour of CAS is the resultant of the contributions of each agent in the system. However, agents are autonomous and try to react to changes in their cyber-physical context which may not be uniform. A paradigmatic example is the use of artificial, autonomous agents in rescue contexts that may put operators lives at stake [4]. An agent should autonomously and quickly adapt its behaviour to changes triggering danger situations in a nearby area. This behaviour should be driven by the changes occurring in the components operational environments as well as the changes in the local computational state of each component, collectively taken. Also, the *global* behaviour of CAS should emerge from the *local* behaviour of its components.

Let us elucidate the points above considering the coordination of a number of robots patrolling some premises to make sure that aid is promptly given to human operators in case of accidents. A plausible local behaviour of each robot can be:

(1) to identify accidents,
(2) to assess the level of gravity of the situation (i.e. to choose an appropriate course of action),
(3) to alert the rescue centre and nearby robots (e.g. divert traffic to let rescue vehicles reach the location of the accident more quickly), and
(4) to ascertain how to respond to alerts from other robots (e.g. if already involved in one accident or on a low battery, a robot may simply forward the alert to other nearby robots).

✉ M. Murgia
maurizio.murgia@gssi.it

R. Pinciroli
riccardo.pinciroli@gssi.it

C. Trubiani
catia.trubiani@gssi.it

E. Tuosto
emilio.tuosto@gssi.it

[1] Gran Sasso Science Institute, L'Aquila, Italy

Note that robots' behaviour depends on the physical environment (tasks (1) to (3)) as well as their local computational state (task (4)).

A possible expected global behaviour is that robots try to maximise the patrolled area while trying to avoid remaining isolated and to minimise the battery consumption. It is worth remarking that the global behaviour of CAS is not typically formalised *explicitly*. As in the robots scenario above, the global behaviour is informally stated and it is expected to emerge from the runtime behaviour of components. For instance, when designing the algorithm for the roaming of robots, one could assume that a robot will not move towards an area where there are already a certain number of robots.

This paper applies behavioural specifications to the quantitative analysis of CAS. In fact, it is often the case that the specification of the global behaviour of CAS involves non-functional properties. In the example above, for instance, one would like to guarantee that there is always a minimum number of robots patrolling certain areas. Using a simple, yet representative, robot scenario, we apply two different approaches to the performance analysis of CAS. Besides showing how to use behavioural specifications to analyse non-functional properties of CAS (emergent) behaviour, this exercise is instrumental for our contribution, which is a study of the relation between two complementary approaches to the performance analysis of CAS recently proposed respectively in [31] and in [18]. The quantitative analysis in [31] is based on generalised stochastic Petri nets, while the one in [18] relies on queuing networks. These approaches differ also on the methodologies for the quantitative modelling they support. The approach in [31] requires that the designer must directly come up with a performance model (rendered as generalised stochastic Petri nets). Instead, in the latter approach [18], the designer does not have to directly develop the queueing network for the quantitative analysis, because it is indeed 'compiled' from the behavioural specification of the CAS. In this sense, the former approach is a *model-based* methodology, while the latter is a *language-based* methodology.

Through the paper we will highlight the main differences between the two methodologies, compare them and discuss their relative merits. More precisely, the paper uses a robot scenario to address the following two research questions:

RQ1 To what extent the approaches in [18] and in [31] support performance-aware design of CAS?
RQ2 How do the features of the approaches in [18] and in [31] compare?

Methodologically, we identify three scenarios which differ among each other in the way components of the system are coordinated to achieve their goals. Each scenario corresponds to a different architecture and it is designed to capture some realistic case study. Then, the quantitative analysis of each scenario is performed using the two approaches. Finally, we analyse the results and draw our conclusions. As we will see, our analysis suggests a hybrid combination hinging on both approaches.

Outline Sect. 2 describes the scenario used in the rest of the paper. We will consider three different architectures (i.e. independent, collaborative and centralised) for this scenario. Section 3 provides the models based on the specification language in [18] for the three architectures. Section 4 shows the performance analysis based on the proposed models of Sect. 3. The comparison between the approach in [31] and the one illustrated in Sect. 3 is discussed in Sect. 5. Final comments, related and future work are in Sect. 6.

This paper is a revised version of [28]. The main difference with respect to [28] is that here we extend our analysis with a new scenario, dubbed *centralised*. Also, we significantly revised Sect. 1 to better motivate our work, and Sect. 2 to improve the presentation of our case study as well as to describe the new centralised scenario. Besides some minor changes to its original, Sect. 3 contains a completely new part (Sect. 3.2). A new format of the messages is introduced to simplify the definition of the predicate ρ_d: the position of the sender is now part of the sent message, while in [28] it was obtained through function pos. The presentation of Sect. 4 has been improved; also, this section has been extended with a new part on the analysis of the centralised architecture (cf. Sect. 4.3). Sections 5 and 6 have also been revised. We extended the former to take into account the analysis of the centralised architecture and the latter to compare our work with other approaches that we could not include in [28].

2 A robot scenario

Through the paper, we will rely on a scenario where robots have to transport some equipment necessary in an emergency from an initial zone to a target location. There are a few paths that robots can take to accomplish their mission. We consider the following options:

- Robots can take a straightforward path.
- Robots can attempt a shorter path.

The first option is always viable, but it is slower than the other. The second option requires robots to go through a sequence of doors that can randomly switch between that states 'open' and 'closed'; for simplicity, we will consider cases with two doors to be open/closed. In all the considered architectures, we design robots so that they take the longer route as soon as they find a door closed. When this happens on the second door, it will take more time for robots to reach the destination than if the alternative route had been taken at the start of the journey. After the delivery, robots return to the initial zone trying to follow the reverse path and the

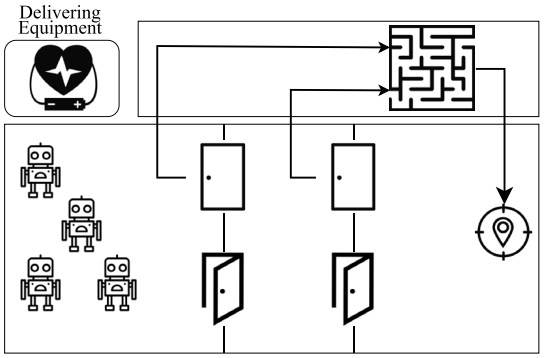

Fig. 1 The analysed scenario. Two doors separate robots from their destination

same constraints apply. A high-level view of the considered scenario is depicted in Fig. 1, which is borrowed from [31].

A clear requirement in this scenario is that the equipment is delivered as soon as possible to tackle the emergency. Designing an optimal, or at least acceptable, strategy for the robots to satisfy such requirement is however not trivial, even in our simple scenario. A *bad* strategy may lead most robots on the slowest path, but a *good* strategy is hard to find due to the fact that it depends on how robots react to environmental changes. Indeed, it is commonly accepted that the performance of a cyber-physical system varies with changes in the physical environment. Moreover, as experimentally measured in [31], dynamic changes in the cyber-physical space and architectural patterns impact on the performance of cyber-physical systems. This type of analysis suggests that in this domain it is useful to factor performance at design time. Following [31], we consider three architectural scenarios:

Independent Robots do not cooperate with each other. In this architecture, robots simply detect the state of doors and behave as described above.

Collaborative Robots behave exactly as above on open doors; instead, on closed doors, they send a message to nearby robots before taking the alternative route. This way, every robot that receives such message can directly follow the alternative route.

Centralised Robots receive from the central coordinator the state of the door they are approaching. If the coordinator (periodically updated by robots that sampled the door state) says that the door is closed, robots immediately follow the alternative route. Otherwise, robots head to the door. If the door is indeed open, robots go through and carry on their mission. Instead, if the door is closed, robots retrace their steps, take the alternative route and update the central coordinator with their findings.

The approach proposed in [31] hinges on Generalised Stochastic Petri Nets (GSPNs) [5] as suitable models of cyber-physical systems. In this paper, we apply such approach by adopting (i) a different modelling language, hinging on behavioural specifications and (ii) relying on queueing networks [20] for performance analysis. The modelling language used here has been advocated in [18] for specifying global behaviour of CAS. As shown in [18], this modelling language has a natural connection with queueing networks, therefore enabling performance analysis of CAS. A key feature of our modelling language is that it abstracts away from the number of agents' instances, i.e.

- an arbitrary number of agents can embody the same role; in our case study, for instance, an unspecified number of devices impersonate the robot role;
- communicating partners address each other through attribute-based communication instead of explicit channels (as in channel-based communication) or their identities (as in the actor model).

Therefore our model allows us to specify complex multiparty scenarios regardless the number of agents' instances.

3 A behavioural specification model

The behavioural specifications in [18] are inspired by AbC, a calculus of *attribute-based communication* [1]. Basically, attribute-based communication is an abstract mechanism for addressing partners of communications. Unlike communication mechanisms based on channels or direct addressing of senders and receivers, attribute-based communication allows one to specify many-to-many communication among dynamically formed groups of senders and receivers. Informally, components expose domain-specific *attributes* used to address communication partners according to *predicate* on such attributes. For instance, the robots and the doors in the scenarios in Sects. 1 and 2, may expose an attribute recording their physical position. This attribute can be used to specify communications among "nearby" robots through a suitable predicate so to determine the communication group as the set of robots satisfying such predicate.

The attribute-based communication mechanism of AbC mechanism is rendered in [18] through *interactions*, which in their most general form are defined as

$$\mathsf{A}_{|}\rho \xrightarrow{e \quad e'} \mathsf{B}_{|}\rho', \tag{1}$$

where A and B are *role names*, ρ and ρ' are logical formulae, e is a tuple of expressions, and e' is a tuple of *patterns*, that is expressions possibly including variables. The intuitive meaning of the interaction in (1) is

> "*any* agent, say A, *satisfying* ρ generates an expression e for *any* agents satisfying $\rho'_{r,d}$, dubbed B, provided that expression e' *matches* e." [18]

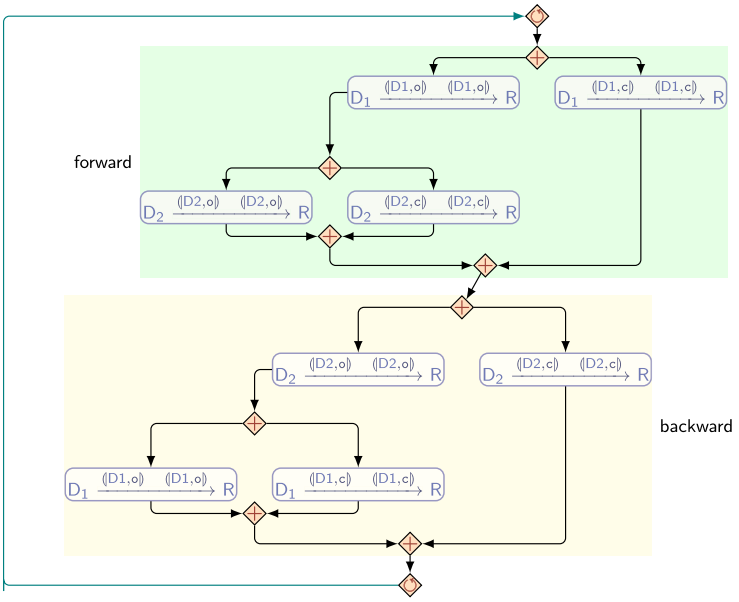

Fig. 2 A model for the independent architecture

The conditions ρ and $\rho'_{r,d}$ predicate over components' attributes. The payload of an output is a tuple of values e to be matched by receivers with the (tuple of) patterns e'; when e and e' match, the effect of the communication is that the variables in e' are instantiated with the corresponding values in e.

A send operation targets components satisfying a given *predicate* on such attributes. Let us illustrate this again using our scenarios. If pos is the position attribute exposed by robots, we can define a predicate ρ as

$$\text{abs}(\text{self.pos} - \text{pos}) < 5\text{mt},$$

for receiving robots. Hence, ρ is satisfied by a receiving robot less than five meters away from a sending robot (i.e. the difference between the position self.pos of the receiver and the one pos of the sender is less than five metres). A robot disregards a message if its position is such that it does not satisfy ρ.

Role names A and B in (1) are pleonastic: they are used just for succinctness and may be omitted for instance writing $\rho \xrightarrow{e \quad e'} \mathsf{B}_{|}\rho'$ or $\rho \xrightarrow{e \quad e'} \rho'$. Also, we abbreviate $\mathsf{A}_{|}\rho$ with A when ρ is a tautology.

Interactions are the basic elements of an algebra of protocols [37] featuring iteration as well as non-deterministic and parallel composition. For the sake of this paper, it is sufficient to think of a protocol as a regular expression in the alphabet of interactions of shape (1). Actually, we will avoid technicalities using the intuitive graphical presentation of this algebra given in [37]. We use *gates* to identify control points[1] of protocols:

- entry and exit points of loops are represented by ◇-gates,
- branching and merging points of a non-deterministic choice are represented by ⊕-gates.

This notation will be further described in the rest of this section which provides information on the architectures of our scenario in terms of the graphical notation sketched above.

3.1 Independent architecture

Figure 2 gives a possible model capturing the independent architecture described in Sect. 2 in the graphical notation of our specification language. The protocol behaviour is rendered by a loop whose body is delimited by the ◇-gates. The model[2] of the body consists of the sequential composition of the behaviour for the forward and the backward journey[3] of robots. Robots try to avoid the longest route. On their forward journey, robots attempt to use the path going through the first door and then the second; likewise, on their backwards journey, they try to use the path going through the second and then the first door.

Interactions among doors and robots do not involve value passing; for instance, robots detect the status of the first door when they pattern match on the tuples $\langle\!|\mathsf{D1},\mathsf{o}|\!\rangle$ and $\langle\!|\mathsf{D1},\mathsf{c}|\!\rangle$ for open and closed doors respectively (and likewise for the second door). Robots detect the status of a door according to the format of the messages they intercept. For instance, on its forward journey, a robot either pattern matches the tuple

[1] We do not consider forking and joining points of parallel composition (represented by |-gates), since this feature is not used in our case study.

[2] The model in Fig. 2 is rather simplistic and we will refine it soon; we use it to introduce our graphical notation.

[3] Highl-ighted by a light green and yellow background, respectively.

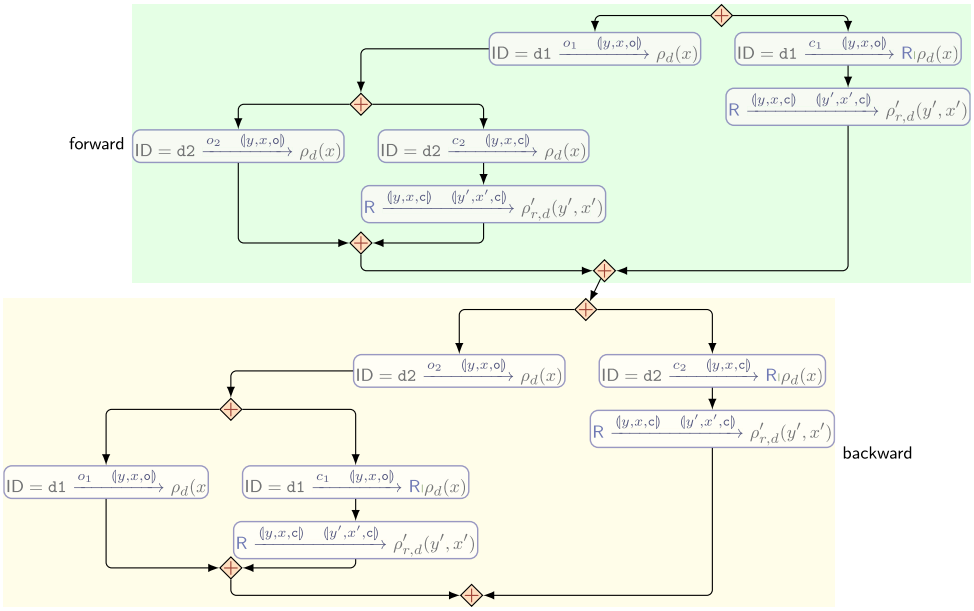

Fig. 3 A model for the collaborative architecture

$\langle\!|\mathsf{D1},\mathsf{o}|\!\rangle$ or the tuple $\langle\!|\mathsf{D1},\mathsf{c}|\!\rangle$ from the first door. This choice is represented in Fig. 2 by ⟡-gate immediately below the topmost ⟡-gate. If the robot receives a $\langle\!|\mathsf{D1},\mathsf{c}|\!\rangle$ tuple from the first door, it continues its journey on the alternative route after which it starts the backward journey. Otherwise, the robot approaches the second door and again goes through if $\langle\!|\mathsf{D2},\mathsf{o}|\!\rangle$ is received, otherwise the robot takes the alternative route. The behaviour on the return journey is similar, depending on the order in which robots approach doors.

Let us now refine this model so to take into account the distance of robots from doors. Indeed, for simplicity, Fig. 2 uses role names D_1, D_2, and R however, this is not very precise. Here we are interested to express that robots detect the status of a door only when they are "close enough" to it. Attribute-based communication can address this issue. To capture the behaviour described above let us assume that robots and doors expose the attribute ID yielding their identity. Then, we can define the conditions

$$\rho_d(x) \equiv abs(\mathsf{self.pos} - x) < d,$$

where d is a parameter of our model setting the maximal distance at which doors and robots communicate. Then we can replace in Fig. 2 the interactions $\mathsf{D}_1 \xrightarrow{\langle\!|\mathsf{D1},\mathsf{o}|\!\rangle \quad \langle\!|\mathsf{D1},\mathsf{o}|\!\rangle} \mathsf{R}$ with

$$\mathsf{ID} = \mathsf{d1} \xrightarrow{\langle\!|\mathsf{self.id},\mathsf{self.pos},\mathsf{o}|\!\rangle \quad \langle\!|y,x,\mathsf{o}|\!\rangle} \rho_d(x) \tag{2}$$

and similarly for the interactions $\mathsf{D}_1 \xrightarrow{\langle\!|\mathsf{D1},\mathsf{c}|\!\rangle \quad \langle\!|\mathsf{D1},\mathsf{c}|\!\rangle} \mathsf{R}$ and those involving D_2. Interaction (2) and the one for the closed status state that the door[4] with ID set to d_1 emits a tuple with their identity and the status. These tuples are intercepted by components whose state satisfy $\rho_d(x)$, where x is the variable instantiated with the position of the sender. Other components would simply disregard those messages. Note that y, substituted with the value d_1, is dummy here; it will be used later.

[4] The fact that identifiers are unique is not built-in in our model; in principle, there could be more doors with the same identifier.

3.2 Collaborative architecture

The collaborative architecture can be obtained by simply extending the independent one with the interactions among robots. A possible solution is given in Fig. 3 where the binary predicate $\rho'_{r,d}$ is defined as

$$\rho'_{r,d}(z,z') \equiv \rho_r(z) \wedge \neg\rho_d(z')$$

and is satisfied by any component within a radius r from component z and to a distance more than d from component z'. Note that r is another parameter of our model setting the radius at which robots communicate with each other. For readability, Fig. 3 shows only the body of the loop and there o_1 and c_1 shorten $\langle\!|\mathsf{D1},p,\mathsf{o}|\!\rangle$ $\langle\!|\mathsf{D1},p,\mathsf{c}|\!\rangle$, respectively, where p is the position of $\mathsf{D1}$ (and likewise for o_2 and c_2).

As in the independent architecture in Sect. 3.1, there is a forward and a backward phase. Remarkably, the modification of the model in Sect. 3.1 is pretty simple. The only difference is that, before continuing their journey on the alternative route, robots communicate to nearby robots the status of the door when they find it closed.

The fact that the adaptation is quite straightforward is due to the features offered by our modelling language. The attributes of components are indeed allowing us to just reuse

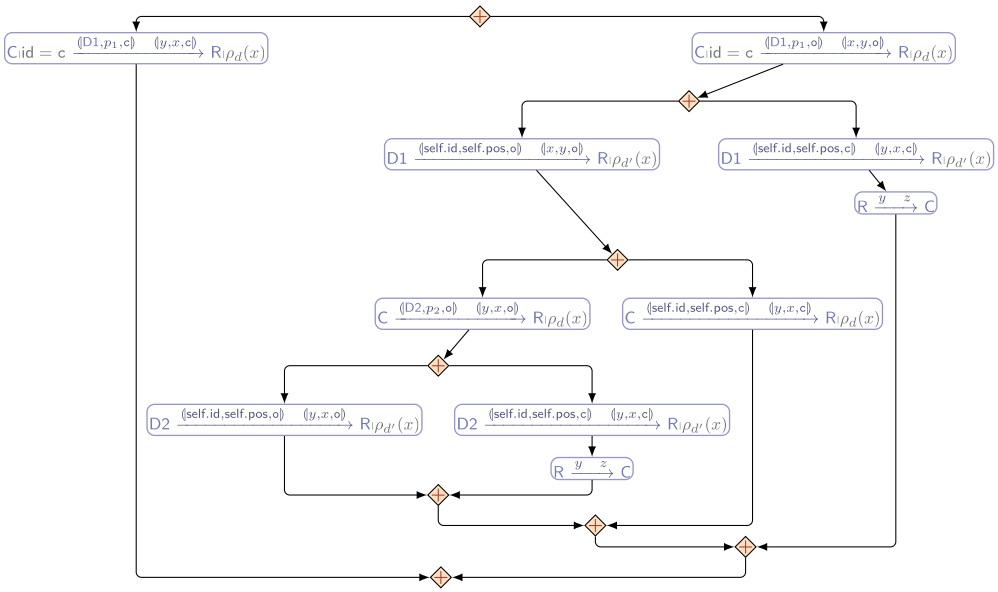

Fig. 4 A model for the centralised architecture. The diagram corresponding to the backward journey is similar and omitted for the sake of space

the condition ρ_d also for coordinating inter-robots interactions. There is the following crucial remark to be made.

The behaviour of robots is to wait for three possible messages: the two sent by the door and one possibly coming from a robot which detected that the door was closed. As a consequence, there might be robots satisfying condition $\rho'_{r,d}(y,x)$ in Fig. 3, that is they are not close enough to the door, but have a nearby robot, say r, aware that the door is closed. These robots should therefore be ready to receive the communication from robot r. Our model accounts for this type of robots, but the graphical notion 'hides' this since it makes explicit only two branches on ⨁-gates. As we will see in Sect. 4, this is a key observation for our performance analysis.

3.3 Centralised architecture

In this architecture, a coordinator C maintains information about the status of doors D1 and D2. The status is communicated to robots approaching a door. Moreover, the coordinator updates its records of a door status when a robot finds the door closed.

The protocol is modelled by the diagram in Fig. 4 where, for the sake of space, we report only the body of the forward journey. When a robot is closer to a door less than a distance d, it receives from C the status of that door (messages $(\!|\mathsf{D1}, p_1, \mathsf{o}|\!)$ and $(\!|\mathsf{D1}, p_1, \mathsf{c}|\!)$). For instance, the topmost interactions in Fig. 4 describe the communication between the coordinator C and the robots R approaching door D1. If D1 is closed, C sends $(\!|\mathsf{D1}, p_1, \mathsf{o}|\!)$ that robots R pattern match with $(\!|y, x, \mathsf{c}|\!)$ when they are within a distance d from D1. This makes robots R to get in variables y and x respectively the identity of the door and its position. If the door is closed, R continues on the longest route; otherwise, R continues towards the door.

When R is away from the door a distance $d' < d$, it can receive the actual status of the door, communicated by the door itself. As before, R decides which route to take depending on such status; however, when approaching a closed door, R updates C informing it that the door was found closed.

4 Quantitative analysis

In [18], we relate our modelling language to Queueing Networks (QNs) [20], a widely used mathematical model to study waiting lines of systems represented as a network of queues [12, 13, 16]. *Customers* and *service centres* are two main elements of QN. Customers represent jobs, requests, or transactions that need to be handled by the system. Service centres are resources that process the customers of the systems. If a service centre is busy (i.e. it is processing a customer), other jobs need to wait in a queue for their turn to be served. Other QN elements are *routers* and *delay stations,* which allow forwarding customers to a specific centre and modelling processing lags that do not require system resources, respectively. In our previous work [18], we provide rules for the automatic generation of QN models from behavioural specifications. The main idea is to transform (i) an interaction into a service centre and (ii) a non-deterministic choice into a router. In the following, we apply this construction to our robot scenario and its architectures.

The QN performance models derived from the behavioural specifications are compared for assessing their validity to GSPN models developed in our recent experience [31]. A GSPN consists of *places* (represented as circles), *tokens* (represented as dots), *transitions* (represented as rectangles), and *arcs* that connect places to transitions and

vice-versa. A transition is enabled (to fire) when all its input places (a.k.a. *preset*) contain enough tokens, i.e. at least as many tokens as specified by the weight of arcs. When a transition fires, the number of tokens specified by the arc weight is removed from input places and new tokens are generated into output places (if any, a.k.a. *postset*). After being enabled, a transition may fire immediately (*immediate transitions*, represented by a thin black rectangle) or after a stochastically distributed time (*timed transitions*, depicted as a thick white rectangle). In this paper, all timed transitions follow an exponential distribution.

In [31], we use GSPNs to show that the performance of cyber-physical systems is affected by architectural patterns and dynamic space changes. Here, we aim (i) to investigate the performance of CAS using QNs and (ii) to compare these results with those obtained by studying the same system with GSPNs. This analysis is carried out using JSIMgraph, i.e. the simulator of Java Modelling Tools (JMT) [8]. The JSIMgraph simulator discards the transient system behaviour, namely, behaviour for which the response of a system (i.e. its output metrics) changes over time. When all performance indices under analysis are within the desired *confidence interval*[5] JSIMgraph stops the simulation. We set to 99% the confidence interval for our experiments.

4.1 Independent architecture

We answer RQ1 by using the approaches in [18, 31] to study the performance of CAS. In this section, we consider the robot scenario with independent architecture.

We obtain the QN in Fig. 5 by applying rules defined in [18] to the forward and backward boxes of the behavioural specification in Fig. 2. For example,

- the first and second non-deterministic choices in the forward box of Fig. 2 become respectively the `D1 status` and `D2 status` router in the *FORWARD* box of Fig. 5;
- interactions on the left branch of each choice in the forward box of Fig. 2 become the `D1 open` and `D2 open` service centres in the *FORWARD* box of Fig. 5;
- similarly, interactions on the right branch of each choice in the forward box of Fig. 2 become the `D1 closed` and `D2 closed` service centres in the *FORWARD* box of Fig. 5.

The GSPN depicted in Fig. 6 is obtained by adapting the performance model proposed in [31] for independent robots to the scenario considered in Sect. 2. A delay centre (i.e. `Robots` in Fig. 5) represents the number of robots in the system as well as the time that each robot waits on average for a new task to be assigned. Our experimental setting considers a fixed number of robots ($N = 100$ in Table 1).

[5] A confidence interval is a range of values that is likely to contain performance parameter with a certain probability.

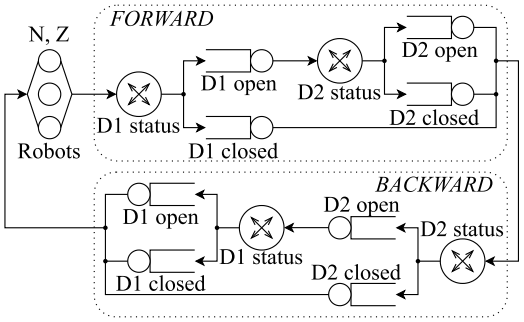

Fig. 5 QN of the independent architecture

Initially, all robots are waiting to receive a task. After an exponentially distributed time, the transition `wait` fires. A task is assigned to one of the waiting robots, a token is removed from the input place of the `wait` transition, and a new token is generated in its output place. Such output place represents robots moving towards the first doors. Every time the transition `D1 reach` fires, a robot reaches the door. If the door is closed, the immediate transition `D1 fail` fires and the robot takes the alternative long route with the timed transition `D1 alt.`, otherwise, the robot goes through the first door with the immediate transition `D1 succ.` and continues its journey towards the second one with the timed transition `D1 straight`. Places and transitions govern the door status (i.e. open or closed). For example, places of the first door are `D1 open` and `D1 closed`, whereas its transitions are `D1 opening` and `D1 closing`. A door is closed or open until the related transition (i.e. for the first door `D1 opening` and `D1 closing`, respectively) fires and changes the door status.

The two performance models are parameterised with numerical values from the literature [40] as shown in Table 1.

Using the QN and GSPN models developed for the scenario under analysis, we can answer RQ2. Specifically, we compare the response time estimated by both models when the probability that each door is open varies. We define the response time as the time taken by each robot to complete the assigned task and return to the initial room. Results of this analysis are plotted in the left histogram of Fig. 7 with their 99% confidence interval. In this figure, the extreme cases of 0.01 and 0.99 probabilities are reported instead of 0 and 1 since the latter ones are not probabilistic by definition, i.e.,such values imply doors are always closed or open. Experimental results show high agreement in the performance predictions. The QN derived from the behavioural specification and the GSPN predicts a comparable response time independently of the considered probability. If the probability of doors being open is high, robots are likely to take the fastest route and the response time is minimum. The longest response time is observed when $0.2 \leq Pr(Door\ is\ Open) \leq 0.3$, i.e. robots may find a door open and the other one closed.

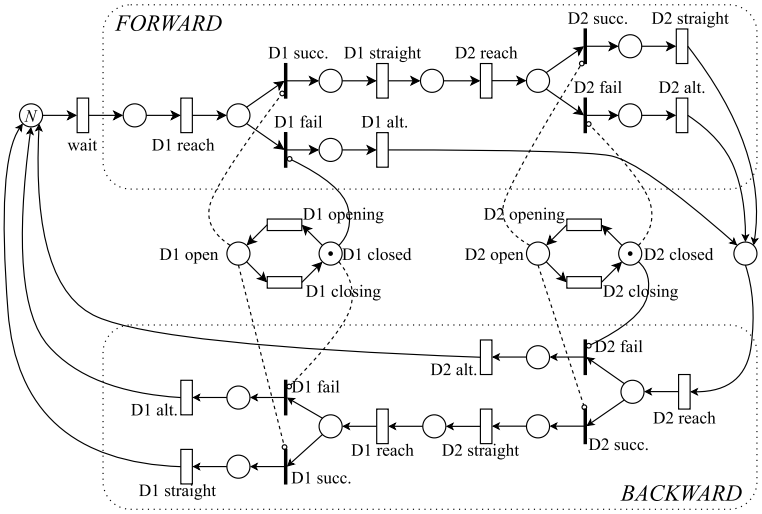

Fig. 6 GSPN of the independent architecture

Table 1 Numerical values used for GSPN and QN models of independent, collaborative, and centralised architectures. *Direction* indicates `Forward` (F) or `Backward` (B). $S_{D* open}$ (QN) is obtained by summing $S_{D* reach}$ and $S_{D* straight}$ (GSPN). $S_{D* closed\,(coord)}$ (QN) is obtained by summing $S_{D* ask}$ and $S_{D* skip}$ (GSPN). Timing parameters are in second

GSPN			QN		
Parameter	Direction	Value	Parameter	Direction	Value
N (num. of robots)		100	N (num. of robots)		100
S_{wait}		10	Z		10
$S_{D* closing} + S_{D* opening}$		60	–	–	–
$S_{D* reach}$	F / B	5	$S_{D* open}$	F / B	10
$S_{D* straight}$	F / B	5			
$S_{D1\,alt.}$	F	45	$S_{D1\,closed}$	F	45
$S_{D2\,alt.}$	F	60	$S_{D2\,closed}$	F	60
$S_{D1\,alt.}$	B	60	$S_{D1\,closed}$	B	60
$S_{D2\,alt.}$	B	45	$S_{D2\,closed}$	B	45
$S_{D* follow}$	F / B	46	$S_{D* msg.}$	F / B	46
$S_{D* send}$	F / B	1	$S_{D* send}$	F / B	1
$S_{D* ask}$	F / B	1	$S_{D* open\,(coord)}$	F / B	1
$S_{D* skip}$	F / B	40	$S_{D* closed\,(coord)}$	F / B	41
$S_{D* update}$	–	60	–	–	–

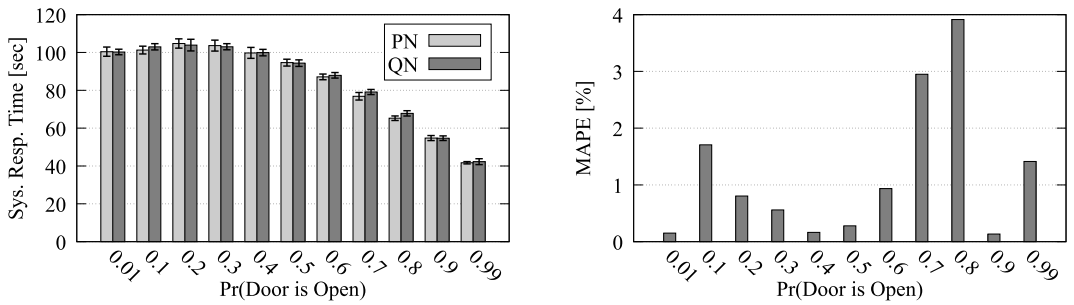

Fig. 7 Independent architecture: System response time (left) and MAPE (right) vs. probability of door being open

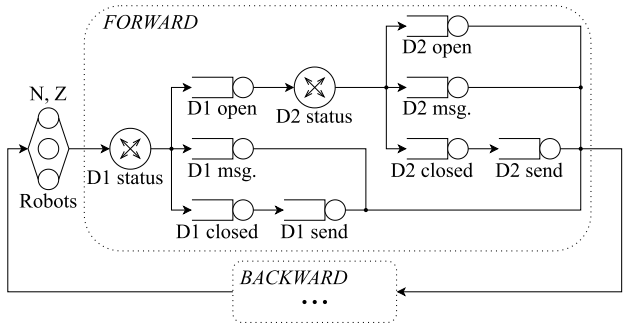

Fig. 8 QN of the collaborative architecture

Indeed, robots spend more time taking the alternative route when they are blocked at the second door, see Table 1. The QN predictions are further assessed in the right histogram of Fig. 7 via mean absolute percentage error (MAPE):

$$\text{MAPE}[\%] = \frac{|\, R_{GSPN} - R_{QN} \,|}{R_{GSPN}} \cdot 100,$$

where R_{GSPN} and R_{QN} are the response times estimated using GSPN and QN, respectively. The MAPE is 1.18% on average and always less than 4%. This is an excellent result when estimating the system response time [35].

4.2 Collaborative architecture

Here, we answer RQ1 and RQ2 considering the robot scenario in Sect. 2 deployed with a collaborative architecture.

Applying rules in [18] to the behavioural specification of Fig. 3 yields the QN depicted in Fig. 8. Now, routers can forward requests to three different branches. Indeed robots may go through an open door, get stuck in front of a closed door, or receive a message from a peer warning them that the next door is closed. The latter case is represented by the `D1 msg.` and `D2 msg.` service centres for the first and second doors, respectively. When robots get a warning message from their peer, they can immediately take the alternative route without spending time approaching the door and checking its status. The warning message needs to be issued by a robot that finds a closed door after having reached it. This is modelled by `D1 send` and `D2 send` service centres positioned after `D1 closed` and `D2 closed`, respectively.

Performance predictions obtained using the QN in Fig. 8 are compared to those observed modelling the scenario under analysis with the GSPN described in [31] and depicted in Fig. 9. These performance models are parameterised with numerical values reported in Table 1, except for probabilities used in the QN routers (i.e. `D1 status` and `D2 status`). This is necessary for a fair comparison of QN and GSPN performance predictions. Indeed, the probability of a robot receiving a message from a peer is conditioned by other system attributes (e.g. the door status, as well as the number, position and velocity of robots). While the GSPN keeps track of dependencies among input parameters via the accurate and complex modelling of the whole environment, the QN performance model leverages only routing probabilities and the time taken to complete actions. Hence, we first analyse the GSPN model of the collaborative architecture given in Fig. 9. This analysis allows us to infer the probabilities for robots to receive a message from a peer under certain system circumstances so to properly parameterise the QN model.

Similarly to the independent architecture, we estimate the response time of the collaborative scenario against the probability of doors being open, using both QN and GSPN models. Hence, we assess the quality of QN predictions by comparing response time to the one estimated using the GSPN model. The response time predicted by both performance models is plotted in Fig. 10 (i.e. left histogram). The QN parameterised as previously described generates predictions that agree with those from the GSPN. The response time of collaborative systems is generally shorter than the one observed with an independent architecture since robots share knowledge about the environment. Note that such shared knowledge might also affect negatively the performance of a CAS. This is the case when doors are open with a high probability, i.e. $Pr(Door\ is\ Open) \geq 0.9$. Indeed, a robot that gets stuck behind a closed door propagates its finding making other agents take the alternative route, even if the probability that the door will soon turn open again is high.

The maximum MAPE observed with the collaborative architecture is smaller (i.e. less than 1%, see the right histogram of Fig. 10) than the one of the independent scenario. Similarly, also the average MAPE (i.e. 0.33%) improves due to probabilities directly derived from the GSPN and plugged into the QN routers. The probability value for which the maximum MAPE is observed varies across the two architectures, i.e. $Pr(Door\ is\ Open) = 0.8$ in the independent case and $Pr(Door\ is\ Open) = 0.2$ in the collaborative one. Different factors can cause this behaviour, e.g. the intrinsic stochasticity of the system as well as the different types of considered architectures.

4.3 Centralised architecture

Here, the approach presented in [18] is used to model and study the performance of a CAS deployed with the centralised architecture described in Sect. 3.3.

The application of the rules listed in [18] to the behavioural specification depicted in Fig. 4 yields the QN shown in Fig. 11. In order to model the behaviour of robots, which depends on the status of the doors communicated by the coordinator, the QN has two extra routers, `D1 coord` and `D2 coord`. If the coordinator communicates that the next door is closed (i.e. `D1 closed (coord)`), the robot takes the alternative and longer path that allows reaching the

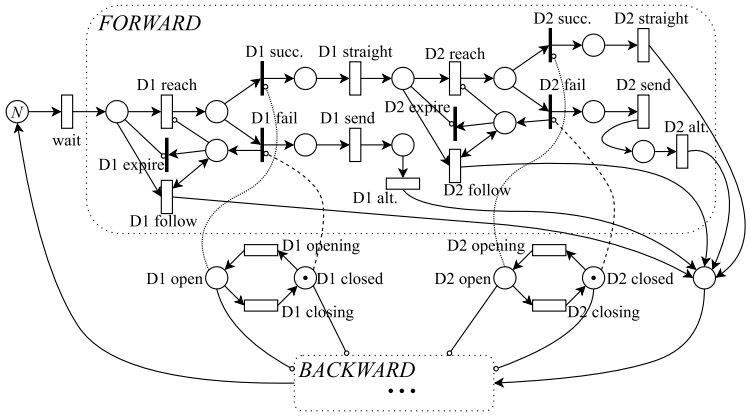

Fig. 9 GSPN of the collaborative architecture

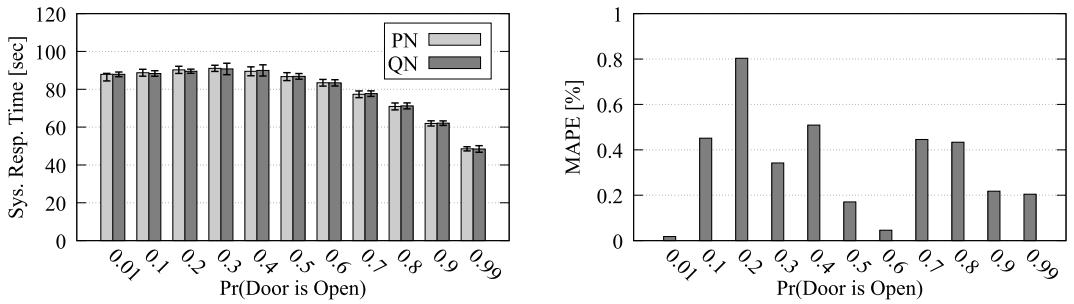

Fig. 10 Collaborative architecture: System response time (left) and MAPE (right) vs. Probability of door being open

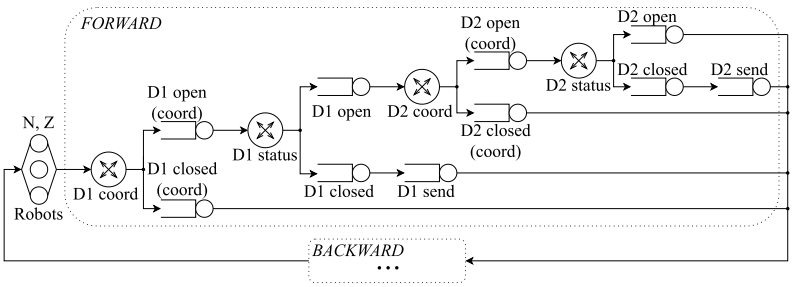

Fig. 11 QN of the centralised architecture

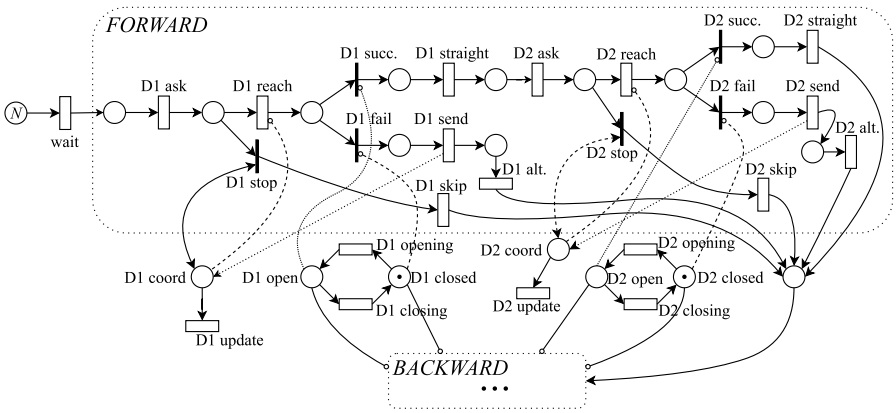

Fig. 12 GSPN of the centralised architecture

destination without finding other obstacles. Otherwise, the robot heads towards the door when the coordinator says that the next door is open (i.e., `D1 open (coord)`). If the door is actually open (i.e. determined by `D1 status`), the robot

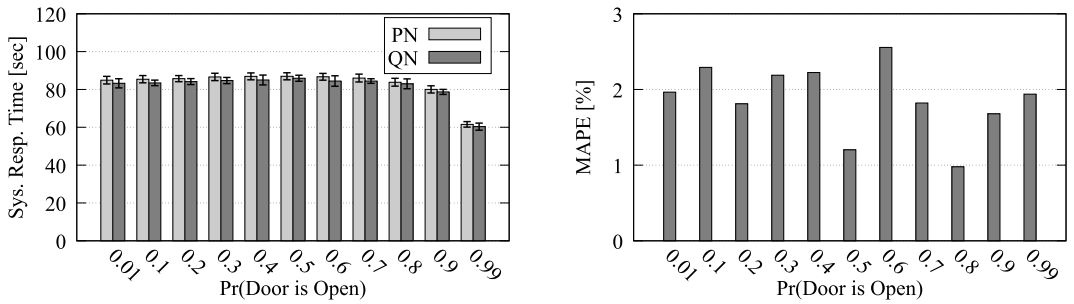

Fig. 13 Centralised architecture: System response time (left) and MAPE (right) vs. probability of door being open

passes through; otherwise, the robot takes the alternative route and communicates its findings to the coordinator (i.e. `D1 send`). The same process is repeated for every door (i.e. *D1* and *D2*) in both directions (i.e. *Forward* and *Backward*).

The performance results obtained by simulating the QN in Fig. 11 are compared to those returned by the GSPN model proposed in [31] and illustrated in Fig. 12. The GSPN and QN models are parameterised as in Table 1. Probabilities used in the QN routers are conditioned on other attributes (e.g., the probability that each door is open and the number of robots in the system) and require further attention to enable a fair comparison between GSPN and QN. Specifically, we parameterise the QN routers with the probabilities observed by running the GSPN model.

The average time taken by a robot to complete its mission and go back to the starting point (i.e. the system response time) is estimated with the two approaches and depicted in Fig. 13 (left). Also in this architecture, QN and GSPN agree on the trend of the system response time, i.e., flat up to *Pr(Door is Open)* = 0.9 when it starts to decrease. Figure 13 depicts the MAPE (right) as well and highlights the accuracy of the QN model with respect to the GSPN one. The small error (i.e. 1.88% on average and less than 3% for all considered probabilities) denotes the ability of the QN to model the same scenarios analysed with the GSPN even when a centralised architecture is considered.

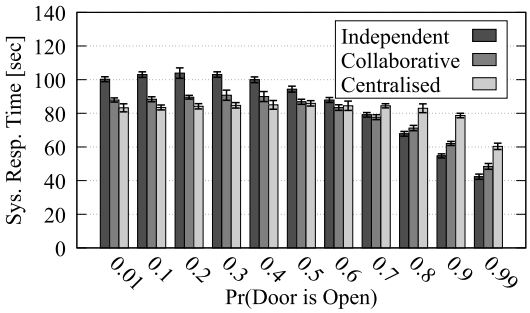

Fig. 14 System response time predicted using QNs to model the three scenarios

5 Discussion

Our analysis confirms that both QNs and GSPNs are suitable for the performance evaluation of CAS.

Our experience shows that there is a trade-off between simplicity and expressiveness in the use of these models (RQ1). The two modelling approaches offer different advantages, which we discuss hereafter. A main advantage of QNs is that they are conceptually simple: performance analysis is based on the probabilities assigned to observable events (e.g. door open). Moreover, QNs can be automatically derived from our behavioural specification of the system. A key observation is that our behavioural specification models introduce a clear separation of concerns: the modelling of the system is orthogonal to its performance analysis that is done by using the derived QNs; hence one just needs to fix the probabilities for the observable events. However, this comes with a cost: it is not usually easy to determine such probabilities.

Instead, the modelling with GSPNs does not require to directly specify probabilities, a clear advantage over QNs. Indeed, with GSPNs one has to simply select a suitable time distribution: this is therefore a more reliable approach compared to QNs. Besides, GSPNs allow for controlling events with a same process; for instance, if a door is open in a direction, it must also be open in the other direction; this cannot be modelled using probabilities only. Overall, GSPNs require more expertise on building the performance model, but its parameterisation includes timing values only, hence they may also be used for monitoring (see, e.g. [29]).

However, GSPNs are more "rigid" than QNs because certain characteristics of the system are hardwired in the model itself. For instance, changing the number of doors robots have to traverse would require a more complex performance model. Moreover, this kind of generalisation will make the size of the model much larger, which can severely affect the performance of the analysis as the state space grows exponentially [5]. This is not the case for QNs derived from our behavioural specification, because they permit to easily abstract

away from the number of system components. On the other hand, GSPNs allow to easily model other types of sophisticate coordination policies. For instance, in the collaborative architecture, it is easy to let the robot first noticing the closed door wait for all nearby robots to take the alternative route before continuing its journey. This is not simple to model with our behavioural specification language or with QNs.

An interesting outcome of our simulations is that the two different model-based performance predictions match, the error is never larger than 4% denoting high agreement between the proposed performance abstractions (RQ2). Figure 14 plots the response time against the probability that doors are open, for the three architectures considered in this paper. These experimental results confirm some expectations for the analysed scenarios. In particular, the system response time is minimal when the probability that doors are open is high; this confirms the observations in [31]. Our experiments highlight also that the independent architecture performs worse than the others when the probability that doors are open is below ~70%. Instead, as this probability increases (from ~80% onwards) the independent architecture outperforms the others. This is due to the fact that for the collaborative and centralised architectures robots can follow the alternative path more promptly avoiding the penalisation of attempting the quicker route and then taking the alternative paths due to a closed door. Moreover, the lack of communication prevents unlucky agents (i.e. robots that find a closed door despite the small chances) from "propagating" their bad luck to their peers. Considering only the response time observed with the collaborative and centralised architectures, experimental results show that the centralised architecture allows saving time when $Pr(Door\ is\ Open) \leq 0.4$. Collaborative and centralised architectures show similar response time if $0.5 \leq Pr(Door\ is\ Open) \leq 0.6$, whereas the collaborative architecture outperforms the centralised one when $Pr(Door\ is\ Open) \geq 0.7$. When the probability that doors are closed is high, the centralised architecture optimize the response time, since the controller has a global knowledge of the environment. As already discussed, this turns into a disadvantage if $Pr(Door\ is\ Open)$ is large, i.e. when the controller's knowledge gets old quickly. In this case, it is convenient leveraging up-to-date information provided by robots that have just sensed the door status.

A problem common to both the QN and GSPN models presented here is that they are not suitable in experimental setting with very high "density" of robots deployed in the system; namely, when the parameter N in Table 1 is much higher than the physical dimension of the operating space. In this case, our QN and GSPN models do not take into account that robots can be significantly slowed down due to the constrained physical space. A possible way to overcome this limitation is to profile the average speed of robots when the population size changes, and parameterise the model accordingly.

6 Conclusions, related & future work

This paper investigates the performance analysis of CAS through the lenses of three different formal notations: (i) an adapted version of a calculus of attribute-based communication (AbC), (ii) Queueing Networks (QNs), and (iii) Generalised Stochastic Petri Nets (GSPNs). We present a case study of autonomous robots for which three architectural scenarios have been elaborated. Experimental results demonstrate the usefulness of our modelling effort that allows to derive performance characteristics of interest. We compare QNs and GSPNs by exploiting the models of the three architectures and observe a high level of agreement on the obtained model-based performance predictions.

Behavioural Abstractions. Coreographic models have been applied to Cyber-Phisical Systems [24, 25], IoT [23] and robotics [27]. These papers focus on verification of correctness properties (e.g. deadlock freedom and session fidelity) and are not concerned with quantitative aspects or performance analysis. Some works in the literature exploit behavioural abstraction for cost analysis of message passing systems. Cost-aware session types [11] are multiparty session types annotated with size types and local cost, and can estimate upper-bounds of the execution time of participants of a protocol. Temporal session types [14] extend session types with temporal primitives that can be exploited to reason about quantitative properties like the response time. A parallel line of research studies timed session types [7, 9, 10], that is session types annotated with timed constraint in the style of timed automata [3]. They have been used for verification of timed distributed programmes by means of static type checking [9, 10] and runtime monitoring [6, 29]. Despite the presence of timed constraints, which makes timed session types appealing for performance analysis, they have never been applied in such setting. Session types have been extended with probabilities [17] for verification of concurrent probabilistic programmes, which is potentially useful for the CAS analysis. A common limitation of these approaches is that they do not easily permit to define the number of agents' instances embodying a specific role in the system specification. Our behavioural model instead allows it, as explained in the final remark of Sect. 2, hence it is suitable for performance analysis that indeed requires such a system workload information.

Quantitative Abstractions. Rigorous engineering of collective adaptive systems calls for quantitative approaches since there is need of designing and managing the coordination activities [15]. A recent survey on verification tools for CAS formation [19] outlines that current analysis techniques are still immature to deal with possible changes in decision-making. This is the reason why quantitative approaches that keep track of behavioural alternatives and their impact on system performance, as we do in this paper, can be of high

relevance for CAS. There exist approaches providing quantitative abstractions for CAS, for instance, Vandin at al. [38] make use of Ordinary Differential Equations (ODEs) to express large-scale systems that are analyzed through bisimulations. Unlike our approach, the analysis presented in [38] has the limitation that is not reliable when the number of agents is rather low, since ODEs provide better accuracy with a large population size. Probabilistic behaviour of agents is investigated by Loreti et al. [26], who also introduce the language CARMA along with a simulation environment to provide quantitative information on CAS. This interesting line of work has currently a limited scalability. Performance properties of CAS are discussed by Viroli et al. [39] with the goal of selecting performance-based optimal configurations while preserving system functionalities. However, this methodology requires the re-deployment of the system and it invalidates the possibility of switching to available alternatives at runtime. Pianini et al. [30] recently proposed a design pattern which partitions system devices into regions and enables internal coordination activities. This is analogous to our collaborative architecture where robots interact with nearby peers.

There are some further approaches that can be considered complementary to ours. For instance, Lee et al. [21] present a language, based on Architecture Analysis & Design Language (AADL), for drones that collaborate in packet delivery, specifically each drone needs to adapt to the motions of the other drones for collision avoidance. Töpfer et al. [36] rely on modelling abstractions that incorporate machine-learning and optimisation heuristics to deal with uncertainty in the environment of CAS. The analysed scenario consists of workers going to a factory that may encounter delays and are replaced by standby workers. Similarly, our scenarios can benefit from specification of uncertainties (e.g. failures of robots and other components [33]) and strategies to rescue items, as recently investigated in [32]. Lion et al. [22] present an operational specification of components as rewrite systems equipped with a Maude specification that is adopted to incrementally analyze the system design. The illustrative application includes two energy sensitive robots roaming on a shared field, and results demonstrate that the introduced coordination prevents livelock behaviour. Summarizing, to the best of our knowledge, our approach differs from the state-of-the-art in the goal of automatically deriving quantitative abstractions from the behavioural specification of CAS. This way, we aim to simplify the derivation of performance indicators of interest, and provide support for a rigorous engineering of CAS.

Future work. Our research agenda includes the investigation of more complex application scenarios, thus to assess the soundness and scalability of our model-based performance analysis. As short-term research direction, we are interested to explore the possibility to automatically deriving the structure of GSPN from our behavioural specifications. We think that this could result beneficial for overcoming some of the drawbacks of GSPN while avoiding the need to determine probabilities. As long-term research direction, we aim to explore the presence of dependencies among input parameters and their impact on the performance analysis of CAS. For instance, in the analysed scenarios, we can consider synchronised behaviour of doors so that they change state in a coordinated way, or the effect of dynamic workloads on the system performance [2, 34]. This fosters some implications in the parameters of the scenarios, e.g. the probability for a robot to find the second door open (after it crossed the first one) relies on its speed and the time when the robot went through the first door.

Acknowledgements The authors thank the anonimous reviewers for their valuable feedback that helped to improve the quality of the paper.

Funding Open access funding provided by Gran Sasso Science Institute – GSSI within the CRUI-CARE Agreement. Research partly supported by the EU H2020 RISE programme under the Marie Skłodowska-Curie grant agreement No 778233. We acknowledge the partial support of MUR project PRIN 20228FT78M *DREAM* (modular software design to reduce uncertainty in ethics-based cyber-physical systems), MUR project PRIN 2017FTXR7S *IT MATTERS* (Methods and Tools for Trustworthy Smart Systems), MUR project PNRR VITALITY (ECS00000041) Spoke 2 ASTRA – Advanced Space Technologies and Research Alliance, and the MUR project PON REACT EU DM 1062/21.

Open Access This article is licensed under a Creative Commons Attribution 4.0 International License, which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons licence, and indicate if changes were made. The images or other third party material in this article are included in the article's Creative Commons licence, unless indicated otherwise in a credit line to the material. If material is not included in the article's Creative Commons licence and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder. To view a copy of this licence, visit http://creativecommons.org/licenses/by/4.0/.

References

1. Abd Alrahman, Y., De Nicola, R., Loreti, M.: A calculus for collective-adaptive systems and its behavioural theory. Inf. Comput. **268**, 104457 (2019)
2. Ali, A., Pinciroli, R., Yan, F., Smirni, E.: It's not a sprint, it's a marathon: stretching multi-resource burstable performance in public clouds. In: International Middleware Conference (Middleware) Industrial Track, pp. 36–42. ACM, New York (2019)
3. Alur, R., Dill, D.L.: A theory of timed automata. Theor. Comput. Sci. **126**(2), 183–235 (1994)
4. Apvrille, L., Tanzi, T., Dugelay, J.-L.: Autonomous drones for assisting rescue services within the context of natural disasters. In: URSI General Assembly and Scientific Symposium (URSI GASS), pp. 1–4 (2014)

5. Balbo, G., Ciardo, G.: On Petri nets in performance and reliability evaluation of discrete event dynamic systems. In: Carl Adam Petri: Ideas, Personality, Impact, pp. 173–185. Springer, Berlin (2019)
6. Bartoletti, M., Cimoli, T., Murgia, M., Podda, A.S., Pompianu, L.: A contract-oriented middleware. In: International Conference on Formal Aspects of Component Software (FACS), vol. 9539, pp. 86–104. Springer, Berlin (2015)
7. Bartoletti, M., Cimoli, T., Murgia, M.: Timed session types. Log. Methods Comput. Sci. **13**(4) (2017)
8. Bertoli, M., Casale, G., Serazzi, G.: JMT: performance engineering tools for system modeling. ACM SIGMETRICS Perform. Eval. Rev. **36**(4), 10–15 (2009)
9. Bocchi, L., Yang, W., Yoshida, N.: Timed multiparty session types. In: International Conference on Concurrency Theory (CONCUR), vol. 8704, pp. 419–434. Springer, Berlin (2014)
10. Bocchi, L., Murgia, M., Vasconcelos, V.T., Yoshida, N.: Asynchronous timed session types – from duality to time-sensitive processes. In: Programming Languages and Systems (ESOP), vol. 11423, pp. 583–610. Springer, Berlin (2019)
11. Castro-Perez, D., Yoshida, N.: CAMP: cost-aware multiparty session protocols. Proc. ACM Program. Lang. **4**(OOPSLA), 155:1–155:30 (2020)
12. Cerotti, D., Gribaudo, M., Piazzolla, P., Pinciroli, R., Serazzi, G.: Multi-class queuing networks models for energy optimization. In: International Conference on Performance Evaluation Methodologies and Tools (VALUETOOLS). EAI (2014)
13. Cerotti, D., Gribaudo, M., Pinciroli, R., Serazzi, G.: Stochastic Analysis of Energy Consumption in Pool Depletion Systems. LNCS, vol. 9629, pp. 25–39. Springer, Berlin (2016)
14. Das, A., Hoffmann, J., Pfenning, F.: Parallel complexity analysis with temporal session types. Proc. ACM Program. Lang. **91**, 1 (2018)
15. De Nicola, R., Jähnichen, S., Wirsing, M.: Rigorous engineering of collective adaptive systems. Int. J. Softw. Tools Technol. Transf. **22**(4), 389–397 (2020)
16. Gribaudo, M., Pinciroli, R., Trivedi, K.S.: Epistemic uncertainty propagation in power models. Electron. Notes Theor. Comput. Sci. **337**, 67–86 (2018)
17. Inverso, O., Melgratti, H.C., Padovani, L., Trubiani, C., Tuosto, E.: Probabilistic analysis of binary sessions. In: International Conference on Concurrency Theory (CONCUR). LIPIcs, vol. 171, pp. 14:1–14:21 (2020)
18. Inverso, O., Trubiani, C., Tuosto, E.: Abstractions for collective adaptive systems. In: International Symposium on Leveraging Applications of Formal Methods, Verification and Validation (ISoLA). Springer, Berlin (2020)
19. Johari, M.H., Jawaddi, S.N.A., Ismail, A.: Survey on formation verification for ensembling collective adaptive system. Adv. Data Comput. Commun. Secur., 219–228 (2022)
20. Lazowska, E., Zahorjan, J., Graham, G.S., Sevcik, K.: Computer System Analysis Using Queueing Network Models. Prentice Hall International, Englewood Cliffs (1984)
21. Lee, J., Bae, K., Ölveczky, P.C.: An extension of hybridsynchaadl and its application to collaborating autonomous uavs. In: Proceedings of International Symposium on Leveraging Applications of Formal Methods, Verification and Validation (ISoLA), pp. 47–64 (2022)
22. Lion, B., Arbab, F., Talcott, C.: A rewriting framework for interacting cyber-physical agents. In: Proceedings of International Symposium on Leveraging Applications of Formal Methods, Verification and Validation (ISoLA), pp. 356–372 (2022)
23. Lopes, L., Martins, F.: A safe-by-design programming language for wireless sensor networks. J. Syst. Archit. **63**, 16–32 (2016)
24. López, H.A., Heussen, K.: Choreographing cyber-physical distributed control systems for the energy sector. In: SAC, pp. 437–443. ACM, New York (2017)
25. López, H., Nielson, F., Nielson, H.: Enforcing availability in failure-aware communicating systems. In: International Conference on Formal Techniques for Distributed Objects, Components, and Systems (FORTE), vol. 9688, pp. 195–211. Springer, Berlin (2016)
26. Loreti, M., Hillston, J.: Modelling and analysis of collective adaptive systems with carma and its tools. In: International School on Formal Methods for the Design of Computer, Communication and Software Systems, pp. 83–119. Springer, Berlin (2016)
27. Majumdar, R., Yoshida, N., Zufferey, D.: Multiparty motion coordination: from choreographies to robotics programs. Proc. ACM Program. Lang. **134**, 1 (2020)
28. Murgia, M., Pinciroli, R., Trubiani, C., Tuosto, E.: On model-based performance analysis of collective adaptive systems. In: LNCS, vol. 13703, pp. 266–282. Springer, Berlin (2022)
29. Neykova, R., Bocchi, L., Yoshida, N.: Timed runtime monitoring for multiparty conversations. Form. Asp. Comput. **29**(5), 877–910 (2017)
30. Pianini, D., Casadei, R., Viroli, M., Natali, A.: Partitioned integration and coordination via the self-organising coordination regions pattern. Future Gener. Comput. Syst. **114**, 44–68 (2021)
31. Pinciroli, R., Trubiani, C.: Model-based performance analysis for architecting cyber-physical dynamic spaces. In: International Conference on Software Architecture (ICSA), pp. 104–114 (2021)
32. Pinciroli, R., Trubiani, C.: Performance analysis of fault-tolerant multi-agent coordination mechanisms. IEEE Trans. Ind. Inform., 1–12 (2023)
33. Pinciroli, R., Trivedi, K.S., Bobbio, A.: Parametric sensitivity and uncertainty propagation in dependability models. In: International Conference on Performance Evaluation Methodologies and Tools (VALUETOOLS). ACM, New York (2016)
34. Pinciroli, R., Ali, A., Yan, F., Smirni, E.: CEDULE+: resource management for burstable cloud instances using predictive analytics. IEEE Trans. Netw. Serv. Manag. **18**(1), 945–957 (2021)
35. Pinciroli, R., Smith, C.U., Trubiani, C.: QN-based Modeling and Analysis of Software Performance Antipatterns for Cyber-Physical Systems. In: International Conference on Performance Engineering (ICPE), pp. 93–104. ACM, New York (2021)
36. Töpfer, M., Abdullah, M., Bureš, T., Hnětynka, P., Kruliš, M.: Ensemble-based modeling abstractions for modern self-optimizing systems. In: Proceedings of International Symposium on Leveraging Applications of Formal Methods, Verification and Validation (ISoLA), pp. 318–334 (2022)
37. Tuosto, E., Guanciale, R.: Semantics of global view of choreographies. J. Log. Algebraic Methods Program. **95**, 17–40 (2018)
38. Vandin, A., Tribastone, M.: Quantitative abstractions for collective adaptive systems. In: International School on Formal Methods for the Design of Computer, Communication and Software Systems, pp. 202–232. Springer, Berlin (2016)
39. Viroli, M., Audrito, G., Beal, J., Damiani, F., Pianini, D.: Engineering resilient collective adaptive systems by self-stabilisation. ACM Trans. Model. Comput. Simul. **28**(2), 1–28 (2018)
40. Weidinger, F., Boysen, N., Briskorn, D.: Storage assignment with rack-moving mobile robots in KIVA warehouses. Transp. Sci. **52**(6), 1479–1495 (2018)

Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.