Gossiping Components for Cyber-Physical Systems

Tomas Bures[1,2], Ilias Gerostathopoulos[1], Petr Hnetynka[1], Jaroslav Keznikl[1,2], Michal Kit[1], and Frantisek Plasil[1]

[1]Faculty of Mathematics and Physics, Charles University in Prague, Prague, Czech Republic
[2]Institute of Computer Science, Academy of Sciences of the Czech Republic, Prague, Czech Republic
{bures, iliasg, hnetynka, keznikl, kit, plasil}@d3s.mff.cuni.cz

Abstract. Developing software for dynamic cyber-physical systems (CPS) is a complex task. One has to deal with the dynamicity and unreliability of the physical environment where the software resides in, while, at the same time, provide sufficient levels of dependability and scalability. Although emerging software engineering abstractions, such as dynamic ad-hoc component ensembles, provide a convenient way to structure software for dynamic CPS, they need to be mapped to robust decentralized execution schemes in real-life settings. A particular challenge in this context is the robust distributed data dissemination in dynamic networks. Gossip-based communication stands as a promising solution to this challenge. We argue, that exploitation of application-specific information, software architecture in particular, has a large potential for improving the robustness and performance of gossip-based communication. This paper proposes a synergy between high-level architectural models and low-level communication models to effectively enable application-specific gossiping in component-based systems. The synergy is exemplified on the DEECo component model which is tailored to the needs and specifics of CPS, and evaluated on an emergency coordination case study with realistic network configurations.

Keywords: Component, Ensemble, Gossip, Cyber-Physical Systems, MANET.

1 Introduction

Cyber-physical systems (CPS) are complex networked systems where the interplay of software control with the physical environment has a prominent role. Examples range from intelligent navigation systems (cars that communicate with each other and with street infrastructure to minimize traffic congestion, fuel consumption, etc.) to emergency coordination systems. Modern CPS are inherently distributed on a large scale and consist largely of mobile devices. They are also increasingly depending on software which has actually become their most intricate and extensive constituent [1].

Building software for large-scale software-intensive CPS via systematic software engineering approaches is a notoriously difficult task. This stems from the fact that CPS invalidate most of the assumptions that typically hold in software engineering of general-purpose systems [2]. Whereas the challenges and opportunities of CPS cover

a range of areas, in this paper we focus on the communication requirements of CPS. In CPS, the physical substratum continuously evolves following the movement of mobile devices. Locality of devices directly affects reachability and connectivity. Communication between devices is opportunistic; there are no guarantees regarding the stability and reliability of the established links. The network topology itself is dynamic and often relies on ad-hoc means without any managing infrastructure. Finally, the environments where CPS operate (e.g., road networks, emergency sites) are highly dynamic and inherently unpredictable.

At the same time, CPS have also a number of specifics that can be advantageously exploited, such as the fact that by moving around in the environment, the wireless devices effectively enlarge the physical area where information can be disseminated [3]. Physical locality and location-dependency of data offer also a natural way to partition the system and provide built-in scalability and robustness.

Looking at the state-of-the-art in distributed communication, gossip and epidemic protocols provide an efficient way to address the aforementioned specifics. Gossip protocols cope with node and network failures, are scalable due to their symmetric nature, and can exploit the physical mobility of gossiping nodes [3]. The gossiping paradigm has already been applied with success in both Internet-based systems and wireless mobile ad-hoc networks (MANETs) [4].

The central idea in gossip protocols is the periodic and probabilistic data transmission from a source node to a set of selected peers [4–6]. They typically combine probabilistic forwarding with counter-based, distance-based, and location-based mechanisms. These mechanisms and configuration parameters are, however, only available at the lower level of the software stack, often transparent to the application/architecture layer. While this is reasonable for uniform data dissemination, it becomes problematic when the spread of data depends on the architectural configuration in question.

The problem lies in a significant abstraction gap between gossip protocols and application-level architecture design using component models tailored to CPS.

In this paper, we aim at bridging this gap by incorporating concerns of gossiping into sound software engineering abstractions, which allow for (i) systematic engineering of CPS via gossiping components and (ii) application-specific, scalable, and efficient gossip-based communication. We do so in the context of DEECo [7] – a component model that specifically targets dynamic, ever-changing architectures of CPS by relying on the concepts of autonomous (soft) real-time components, and dynamic ad-hoc component ensembles. Our approach is not limited to DEECo though, since it is based on the generic synergy between a set of high-level architectural abstractions supporting dynamicity and low-level primitives of gossip-based protocols.

The rest of the text is structured as follows. In Section 2, we elaborate on a scenario from an emergency coordination case study that provides the motivation for architecture-based decentralized solution. Section 3 presents our approach and its integration into DEECo, while Section 4 outlines the implementation. Following, Section 5 presents the simulation-based evaluation results. Section 6 discusses key contributions and emerging related challenges. Finally, in Section 7 we survey the related work and in Section 8 we present our conclusions.

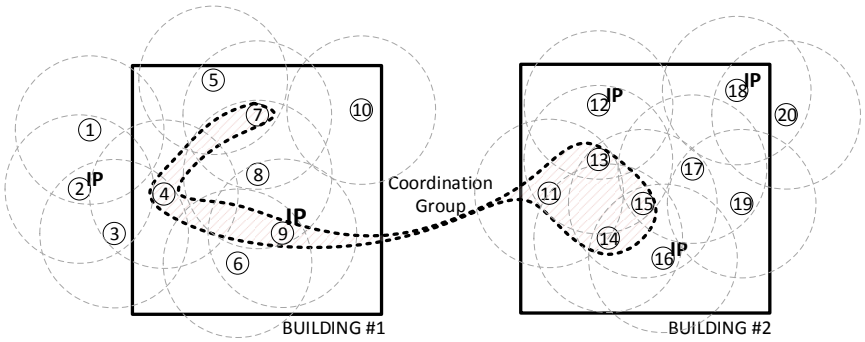

Fig. 1. Motivating scenario: Mobile and stationary nodes cooperate via ad-hoc coordination groups that span within designated boundaries.

2 Motivating Scenario

To illustrate the need for effective mapping of architecture-level concepts to decentralized communication schemes in CPS, we use a scenario taken from a firefighter coordination case study[1], which is a real-world real-scale case study for evaluating distributed adaptive systems.

In the scenario, firefighters belong to tactical groups corresponding to the mission in hand. In case of an emergency, a scouting team composed of a team leader and several team members is initially dispatched to the operation site with the goal to assess the criticality level of the situation in hand, so that appropriate strategic decisions can be taken (e.g., mission escalation, request for additional teams). A strong requirement for the effective cooperation of team members is efficient data dissemination – every member has to be notified in a timely manner about important events and threats (e.g., low oxygen level in a particular room, firefighter in danger because of high temperature level) so that the team can act collaboratively and proactively.

Firefighters are equipped with low-power devices with sensing and actuating capabilities that are integrated into their personal protection equipment (being thus mobile). The devices communicate primarily via wireless mobile ad-hoc network (MANET) protocols (e.g., IEEE 802.15.4); additionally, some devices have IP connectivity. Advantageously, the firefighters may exploit other devices on the fire scene (e.g., on-site access points or devices of other emergency personnel) as network relays to boost their wireless coverage and performance. For illustration, consider an operation site that consists of two buildings (Fig. 1).

Obviously, the key challenges stem from the dynamicity of the whole scenario; in particular, the issues to be addressed include (i) MANET management and efficient use of the communication medium and (ii) seamless inclusion of the related concepts in the high abstraction level employed in the design of the corresponding software architecture.

[1] http://daum.gforge.inria.fr/

```
1.  role TemperatureSensor:
2.    missionID, temperature
3.
4.  role TemperatureAggregator:
5.    missionID, firefightersInDanger, temperatures
6.
7.  component Firefighter13 features TemperatureSensor:
8.    knowledge:
9.      ID = 13, missionID = 1024, position = {50.075306, 14.426948}, oxygenLevel = 90%, temperature = 35.2
10.   process measureTemperature (out temperature):
11.     temperature ← Sensor.read()
12.   scheduling: periodic( 1000ms )
13.   … /* other process definitions */
14. … /* other firefighter definitions */
15.
16. component Leader features TemperatureAggregator:
17.   knowledge:
18.     ID = 2, missionID = 1024, position = {50.075310, 14.426952}, firefightersInDanger = {1,3, …},
19.     temperatures = {{1,30.7}, {2,25.0}, {3,35.2},…}
20.   process findFirefightersInDanger(in temperatures, out firefightersInDanger):
21.     firefightersInDanger ← analyze(temperatures)
22.   scheduling: periodic( 500ms )
23.   … /* other process definitions */
24.
25. ensemble TemperatureUpdate:
26.   coordinator: TemperatureAggregator
27.   member: TemperatureSensor
28.   membership:
29.     member.missionID == coordinator.missionID
30.   knowledge exchange:
31.     coordinator.temperatures ← { (m.ID, m.temperature) | m ∈ members }
32.   scheduling: periodic( 500ms )
```

Fig. 2. Examples of DEECo components and ensembles of the firefighter coordination case study.

2.1 A DEECo-Based Solution

A promising approach for developing software of dynamic CPS is to employ the DEECo component model and its related methods and tools [7].

The design process in DEECo starts with identifying the main system components and dynamic ad-hoc coordination groups – *ensembles* – that the components should establish in order to cooperate for a common goal. In the scenario, ensembles reflect the groups of firefighters exchanging measured data (e.g., temperature, oxygen level) and the groups of officers exchanging strategic information (e.g., mission updates, orders from the chief officer). For illustration, consider the ensemble definition in Fig. 2, lines 25-32. Here, the goal is to enable the members of a firefighting team to propagate information on the measured temperature to the leader of the team so that the leader can determine which firefighters are in danger. In general, an ensemble definition in DEECo contains a condition specifying which components should be considered for membership (lines 28-29), and a function that specifies knowledge exchange among the members (lines 30-31). A particular ensemble (i.e., an instance of an ensemble definition) is identified by its coordinator which features a specific role (line 26). It is instantiated and dissolved by the DEECo runtime environment (Runtime further on),

which periodically (line 32) checks the membership of potential groups of coordinator-members. Within an established ensemble, Runtime periodically performs the knowledge exchange, which transfers data between the coordinator and members.

A component in DEECo is an independent unit of computation and deployment. In the scenario, components correspond to the actors of the system (active firefighter, officer, relay node, etc.). For illustration, consider the two components in Fig. 2. Their state is captured by knowledge (lines 8-9, 17-19) and functionality by processes (lines 10-12, 20-22). Every component features a number of roles, i.e., sets of knowledge fields (lines 1-2, 4-5), which are used as the contract between the component and ensembles. Processes are executed by Runtime in a time- or event-triggered fashion (lines 12, 22). Each process execution consists of atomically reading (a part of) the knowledge of the component, executing the process body, and atomically updating the knowledge with the result.

Note that components in DEECo do not explicitly communicate with each other; their only means of communication is knowledge exchange mediated by the ensembles to which the components belong. A component may belong to a number of ensembles at a time (i.e., ensemble instances may overlap).

2.2 Challenges in DEECo-Based Solution

As shown above, DEECo provides a comprehensive set of concepts at a high level of abstraction, coping with the dynamicity by means of component roles and ensembles. However, mapping the concepts into a scalable and robust DEECo implementation is challenging. The particular challenge lies in how and where to evaluate the membership condition for every possible ensemble. This typically requires reasoning at the system level, exploiting some form of global view over the system state. If this reasoning is encapsulated into a special-purpose entity in Runtime, this entity becomes a bottleneck – single point of failure. In particular, such a centralized solution does not scale when ensembles are to be formed among large numbers of components.

3 Gossiping in Ensembles

In order to mitigate the above issue, we have adopted a fully decentralized and robust approach relying on gossiping for establishing ensembles and performing knowledge exchange. In principle, we replace the network communication layer of DEECo by gossip-based communication and extend the DEECo architectural model (the definition of ensembles in particular) by the concept of a *communication boundary* so as to allow efficient functioning of the underlying gossiping mechanism.

To connect components at the architectural level with their physical deployment, we define *node* as a hardware/software platform where a number of DEECo components are deployed (hosted in an instance of Runtime). Nodes communicate with each other via their network interfaces depending on the available networking infrastructure. Thus, component communication is constrained by the available networking infrastructure

between the nodes the components are deployed on. Inspired by the motivating scenario, we focus on combinations of IP-based networks (wireless and wired) and MANET networks (which allow only for short range broadcast communication). As a product of distributed communication among nodes, each node obtains copies – *replicas* – of the knowledge of components hosted on (some of) the other nodes.

The main principles of our approach to gossip-based ensemble creation and knowledge exchange can be characterized by the following points:

1. A node has its own awareness of ensemble instances existing in the system, specifically of those that include the components deployed on the node. This awareness is based on evaluating the membership with respect to the current knowledge of local components and replicas of other components.
2. Based on the awareness obtained in (1), a node performs only knowledge exchange that results in updating the knowledge of the local components using, again, the current knowledge of the local components and replicas of others.
3. A node proactively disseminates component knowledge, so that every other node has the replicas necessary for realization of (1) and (2).

The following describes the individual elements of our approach in more detail – points 1 and 2 are explained in Section 3.1, while point 3 is elaborated in Sections 3.2 and 3.3.

3.1 Decentralized Evaluation of Ensemble Membership/Knowledge Exchange

Instead of forming ensembles by looking at a snapshot of the whole system (which would imply that a global view on the system has to be available), we take a node-centric approach. Every node periodically iterates over all known ensemble definitions and checks whether a local component can act as a member or coordinator in an instance of the ensemble definition, given its replicas. For each such ensemble instance, it performs the corresponding knowledge exchange, which results in updating the local components' knowledge (but not the replicas).

As an example, consider an instance of the TemperatureUpdate ensemble (Fig. 2) evaluated on the site of the coordinator. In this case, the knowledge exchange results into updating the coordinator's field temperatures.

Note that a consequence of this technique is that degradation of system performance when no connectivity is available (e.g., due to appearance/disappearance/mobility of nodes) is gradual: each Runtime effectively operates on the locally available replicas until they become too outdated to rely upon. Here, we count on one of the specifics of CPS, namely on the fact that the values of most magnitudes in CPS (e.g., temperature in Fig. 2) evolve gradually according to physical laws [8]. Practically this means that a belief which is not too old may still be at least partly relevant. Another consequence is that, due to belief outdatedness causing belief inaccuracy, it is possible for a component to behave as if it were in ensemble with a coordinator, which is not aware of it (and vice-versa). These consequences are further analyzed in Section 6.2.

3.2 Asynchronous Knowledge Dissemination via Gossip

The decentralized solution presented in Section 3.1, requires that each node possesses all the necessary replicas from the components that can potentially participate in ensembles with its local components. We enable this by asynchronous gossip-based knowledge dissemination between all the components of a DEECo application.

The main idea is that every node periodically publishes the knowledge of its local components on the network. For MANETs, this translates to periodic broadcast within the wireless range of the node. For IP networks, it translates to periodic sending to randomly selected nodes. Upon reception of a component's knowledge, a node probabilistically decides whether to retransmit the received knowledge. The nodes that perform such re-transmission then act as relays. Here, we rely on the probabilistic convergence of gossip protocols [9], which ensures that every node will eventually receive the knowledge of every component in a bounded number of steps. The nodes that dynamically appear in the system join the publication and re-transmission of knowledge automatically.

Note that this dissemination scheme dictates that all nodes potentially perform the retransmission, not only the ones that are interested in the disseminated knowledge (i.e., nodes hosting components that could be members of the ensemble which the disseminated knowledge relates to).

3.3 Bounding the Gossip

Although the aforementioned gossip-based knowledge dissemination successfully propagates the knowledge of all nodes to all nodes, it raises performance issues. Specifically, if a DEECo application is considered as a ubiquitous ecosystem in a real environment, the application is potentially boundless w.r.t. network reachability. In such a system, unlimited gossiping is not a viable option. Advantageously, in contrary to the assumption of traditional gossip protocols discussed above, not every node is interested in all the data being disseminated by all the components. Thus, certain application-specific bounds should be established for knowledge dissemination.

For this purpose, we define for each ensemble its *communication group* as the set of nodes to which the ensemble's knowledge dissemination is limited. This set consists of all the nodes where components forming the ensemble are hosted and all the relays necessary for knowledge propagation. Relying on the fact that data is disseminated via gradual flooding, we define a *communication boundary* as the predicate determining the limits of a particular communication group w.r.t. network topology. The relays not satisfying the communication boundary will not participate in the dissemination. In a way, a communication group forms a dynamic, architecture-specific network overlay for knowledge dissemination.

Naturally, a communication boundary includes all the nodes "potentially interested" in the disseminated replicas, while excluding as many of the other nodes as possible. Thus, a communication boundary forms a conservative approximation of the ensemble membership. For example, given the pervasive application from Fig. 1, the communication boundary for the ensemble definition in Fig. 2 can be formulated as follows:

"*For every mission, include all components within all the areas in which the participants of the mission operate.*"

In this example, the communication boundary reflects the fact that all components satisfying the membership condition of the ensemble, i.e., those participating on the same mission, operate in one of the predefined areas. Note however, that the communication boundary predicate is generic w.r.t. a particular mission – it determines a number of different communication groups (thus approximating a number of different ensemble instances), namely a distinct group per distinct mission.

To achieve its desired functionality, a relay has to evaluate a communication boundary much more efficiently than membership condition, preferably using exclusively locally-available information. Thus, we specify communication boundary as a predicate over the local knowledge of the relay and the particular knowledge being disseminated.

Since "communication group" is an application-specific concept relating to application architecture (namely to ensemble membership), we capture it by extending the ensemble definition with a definition of the communication boundary. In addition, we extend the existing concept of "role" to be applicable also at the level of nodes – we say that a node supports a role if one of the components (representative) deployed on the node has structurally-matching knowledge (structural matching enables designing open-ended architectures).

Technically, a communication boundary is defined by a set of predicates (lines 13-16 in Fig. 3). Each of these predicates, given a relay role and a replica role, determines whether a node that has a representative matching the relay role meets the communication boundary for a replica that matches the replica role. Formally, the communication boundary is a conjunction of these predicates (having the form of implications). A relay role has to be either the coordinator or member role.

As an example, in Fig. 3 we show a revised version of the ensemble definition from Fig. 2. Specifically, given a replica corresponding to the member role (TemperatureSensor), the communication boundary includes all relay nodes featuring the TemperatureRelay role, which are in one of the mission areas specified by the replica. This is captured on lines 13-14, which semantically form an implication: the line 13 forms the antecedent (i.e., "if the relay has the role TemperatureRelay and the replica corresponds to the member's role"), while line 14 forms the conclusion. Note, that we have extended the TemperatureSensor role and the knowledge of all the related components to provide the information about mission areas. Similarly, on lines 15-16 the predicate prevents any relaying of replicas matching the coordinator role (as there is no knowledge exchange towards the member). This can be illustrated on Fig. 1 as follows. Provided that all nodes feature the TemperatureRelay role and given that the node 6 participates in a mission that is different to the mission of 9 and localized only to the building #1, then this communication boundary prevents 9 disseminating knowledge of 6 to building #2, as well as 3 from disseminating knowledge of 4. On the other hand, 9, as well as any node in building #1, will disseminate the knowledge of 6 within the building #1. Moreover, 9 will disseminate knowledge of #4 and #7 also to the building #2 via IP.

This part of specification of communication boundary aligns well with the knowledge dissemination in MANETs, where the set of potential recipients is limited by their geographical locality. On the other hand, in large networks that enable routing

```
1.  role TemperatureRelay:
2.    position
3.
4.  role TemperatureSensor:
5.    missionID, missionAreas, temperature
6.
7.  ensemble TemperatureUpdate:
8.    coordinator: TemperatureAggregator
9.    member: TemperatureSensor
10.   membership:
11.     member.missionID == coordinator.missionID
12.   boundary:
13.     case relay: TemperatureRelay, replica: roleOf(member):
14.         ∃area ∈ replica.missionAreas: isInArea(relay.position, area)
15.     case relay: any, replica: roleOf(coordinator):
16.         false
17.     ip-registry: 10.10.16.35, 10.10.16.112
```

Fig. 3. Example of a communication boundary definition in DEECo.

based on global addressing, such as IP networks, a necessary performance optimization is to disseminate replicas only to recipients which themselves meet the communication boundary (rather than blindly pollute the entire IP network). To do this, given a replica, a sender has to be able to (at least partially) assess the validity of the communication boundary with respect to the recipient.

To address this issue, we assume that well-known registries exist providing a relay node the information which other IP-based nodes are part of a communication group (given a particular replica). To avoid unnecessary centralization, such a registry is ensemble specific. The registry either provides statically-defined recipients (well-known relay nodes) or evaluates the communication boundary with respect to a recipient. In the latter case, the potential recipient relay nodes provide the registry with the required relay knowledge. Syntactically, the communication boundary definition contains a set of IP addresses identifying the registries that are specific to the corresponding ensemble (line 17 in Fig. 3). Note that due to the nature of gossip, we do not require all the registries in a given ensemble specification to contain the same information.

3.4 Gossip-based Semantics

To allow for formal analysis of functional and timing properties and precise simulations, as for instance given in Section 4, we have formalized the computational model described in the previous section in terms of operational semantics, which also acts as a thorough, detailed description of the computational model. Technically, based on our previous work [10] we represent the semantics via a state transition system generated by a set of inference rules. Additionally, considering (soft) real-time properties of CPS, the formalization allows only transition traces that are admissible with respect to real-time periodic scheduling of the system processes, ensemble knowledge exchange, and (gossip-based) knowledge dissemination. In a way, these restrictions impose a fairness constraint on the transition traces. Due to space constraints, we refer the interested reader to the technical report [11] for a description of the semantics.

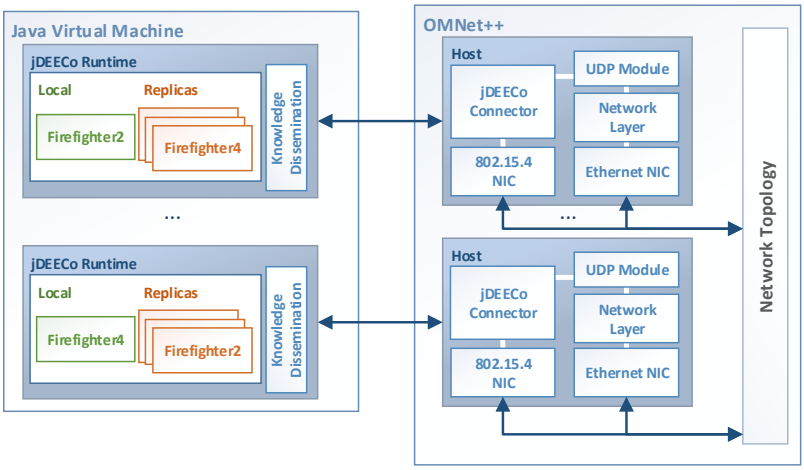

Fig. 4. jDEECo Runtime Framework – OMNet++ integration overview.

4 Implementation

We have implemented[2] the proposed approach by extending the current implementation of jDEECo (a Java implementation of DEECo Runtime). Specifically, we have added support for the concept of communication boundary and the gossip-based knowledge dissemination and ensemble evaluation presented in Section 3. Since these concepts are closely connected to the network layer, we have also integrated jDEECo with the OMNet++ simulation framework[3] that provides an appropriate abstraction for the network infrastructure, enabling precise discrete-time simulations (Fig. 4).

From the perspective of the OSI (Open Systems Interconnection) model [12], our implementation glues together the application layer given by jDEECo Runtime (along with the deployed components and ensembles) with the underlying layers implemented in OMNet++ (Fig. 4). An instance of jDEECo Runtime reflects a single unit of network deployment (e.g., a mobile device). Apart from managing components, scheduling of component processes' execution and ensemble evaluations, jDEECo Runtime automates knowledge management, including network communication needed for knowledge replica dissemination. Each jDEECo Runtime continuously advertises the knowledge of the locally deployed components and, additionally, acts as a relay.

At the network layer, each jDEECo Runtime is bound to its OMNet++ counterpart (namely OMNet host), with which it communicates via JNI (Java Native Interface) calls. Every OMNet host is equipped with two kinds of Network Interface Cards (NICs): one for MANET-based wireless (IEEE 802.15.4) and one for IP-based (Ethernet) communication. Direct communication is implemented via UDP on top of the Ethernet NIC, while MANET-oriented broadcast communication is performed via the wireless NIC. For implementation, we relied on two extensions of OMNet++:

[2] https://github.com/d3scomp/JDEECo
[3] http://omnetpp.org/

the MiXiM plugin delivering a detailed model of the 802.15.4 protocol and the INET framework implementing the whole Ethernet stack.

Each jDEECo Runtime gossips knowledge replicas obtained from the network. We specifically distinguish two cases: gossiping via MANET and direct gossiping. In the case of MANET gossiping, a jDEECo Runtime calculates a probabilistic rebroadcast delay relying on RSSI (Radio Signal Strength Indicator); in case of direct gossiping the data is retransmitted to a random set of peers using a fixed delay. To prevent network overload, the rebroadcast is aborted in case a newer replica is received from another peer. Additionally, MANET gossiping is aborted if the same replica comes from the MANET NIC. The delay and aborting mechanism of MANET gossip is based on the counter-based algorithm proposed in [6].

5 Evaluation

In this section, we show that our gossip-based ensemble evaluation is practically feasible by providing measurements that answer the following fundamental questions: (1) how the gossip-based ensemble evaluation scales with respect to the number of nodes in the system, and (2) how the communication boundary improves the scalability. Specifically, we do it by simulation and measurements of the motivating scenario model.

Building on the implementation outlined in Sections 2 and 3, the evaluated scenario consists of several deployed firefighter teams that partially overlap in terms of radio signal coverage. Each team uses the other teams' members as relays for knowledge dissemination in the overlapping areas to ensure the necessary wireless coverage. The objective of this scenario is to illustrate the performance gain of employing communication boundary, which limits data sharing strictly to the overlapping regions. Note that the communication boundary being used (Fig. 3) allows any node that monitors its position, such as a device of other emergency personnel, to be equally included into the scenario and act as a relay; for brevity we include only firefighters. The scenario combines MANET-based gossiping (with evenly distributed nodes in the area) and direct gossiping realized by Ethernet-enabled nodes (a small fraction of the nodes).

The scenario is affected a large number of factors, such as network density, size of the overlapping regions, wireless communication range, gossip protocol configuration, etc. Therefore, we have simulated our system under a variety of configurations; however, due the space limits, this paper presents results for configurations varying in the number of overlapping teams (thus also in the total number of nodes), while maintaining a fixed node density (close to the highest density safely manageable by the implemented MANET gossip protocol, as evaluated by Williams and Camp in [4]). The detailed information on the configuration parameters, which were set to match the realistic case described in Section 2 as close as possible, as well as the simulation results for various set-ups, can be found on the DEECo project website[4].

The results presented in Fig. 5 show the leader-member end-to-end communication time in a firefighting team (in particular, the time it takes a leader node to learn that a

[4] http://d3s.mff.cuni.cz/projects/components_and_services/deeco/simulations

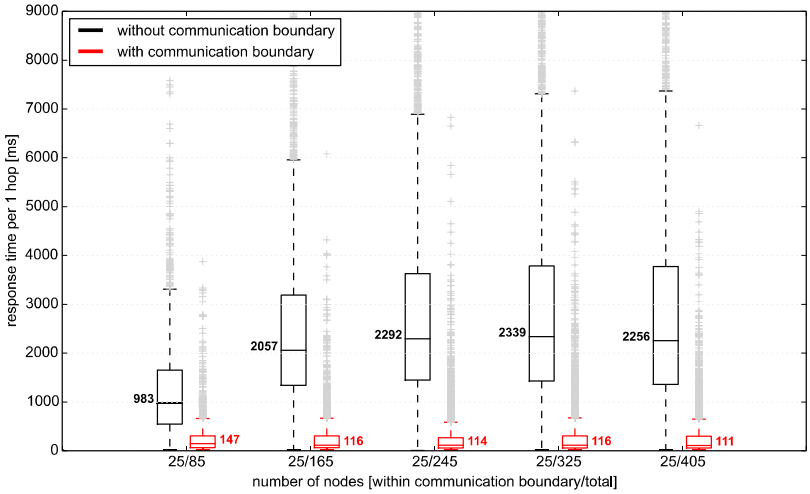

Fig. 5. Time for discovering a team Member in danger by a corresponding Leader.

member of its team is in danger, normalized by the hop distance between the two nodes). Specifically, we compare the cases with and without communication boundary. Not using communication boundary results into propagation of a team's data across all nodes; this causes global degradation of end-to-end communication performance (corresponding to the performance limitations of the implemented gossip protocol). On the other hand, communication boundary localizes the team's data dissemination and prevents the communication channels from overloading, which results in stable performance (as long as the dynamic communication boundary does not grow). Specifically, the communication boundary reduces the utilization of the shared communication medium by preventing "outside" data from penetrating deeper (than necessary) into the team's area. This reduces the overhead of the communication medium; the freed capacity can be now utilized to handle dissemination of the team's data.

6 Discussion

In this section we review the key contributions of our approach and discuss the main related challenges that stem from the decentralized decisions on ensemble membership and gossip-based communication.

6.1 Key Contributions

Integrating the DEECo concept of ensemble with gossip-based communication enables for efficiently dealing with scenarios where system architecture is open-ended and changes continuously; e.g., systems with high mobility of components or largely unreliable communication links. To this end, the autonomicity of DEECo components and best-effort style of communication provided by the gossip-based implementation of ensemble knowledge exchange deliver means for assuring high infrastructural resilience.

Although, due to the dynamic nature of ensembles, the gossip-based implementation of knowledge exchange requires disseminating knowledge to all the potential members, possibly requiring all nodes to act as relays, communication boundary provides means to accurately reduce the dissemination to only those nodes, which are actually needed considering the application-logic point of view. Moreover, as the knowledge dissemination governed by the communication boundary exploits the contextual information available at the application level in the form of component knowledge (current position, temperature etc.), the possible set of relay nodes may change dynamically according to data being disseminated and the state of the relay nodes, as opposed to generic indicators for limiting communication, such as timestamps and hop count.

Consequently, by accurately preventing data from flowing to irrelevant parts of the system, the proposed communication boundary mechanism considerably improves the utilization of the shared communication medium within the MANET network. The gain in communication performance depends on how accurate estimate of a membership the relevant communication boundary is.

6.2 Related Challenges

Belief inaccuracy in asynchronous knowledge dissemination. The belief a component has about the knowledge of another component is essentially always outdated. This outdatedness is mainly rooted in (i) network infrastructure performance (e.g., bandwidth, packet delays, medium access rate, etc.) (ii) MANET topology issues (e.g., large hop distance between sender and receiver), and (iii) ineffective tuning of the employed gossip algorithm (e.g., too long (re)transmission period).

The outdateness of belief determines its inaccuracy, i.e. the difference between the value of the belief and the actual value of the knowledge. Depending on the nature of data (i.e., continuous or discrete domain, rate of change), slight incoherence between knowledge and belief might be tolerated or accounted for during design [8]. Advantageously, this is the case with most of CPS where real-world phenomena (e.g., position, oxygen level, velocity) are to be captured.

Split-brain situations in ensembles. Due to the belief outdatedness and isolated membership evaluation by each potential member, situations where different nodes arrive at conflicting conclusions regarding ensembles may arise. This results in a member acting as if it were in an ensemble having a coordinator who is not aware of it (or vice-versa). As an example, consider an ensemble that is formed of the firefighter components (each hosted on a separate node) whose positions lie within a 10-meter perimeter from a leader (coordinator). When a firefighter node steps out of the designated area, the corresponding firefighter component should not be part of the ensemble. The coordinator, however, will only learn about that at the next time its host node receives an up-to-date replica of that component. Until then, it will falsely consider the firefighter component as a legitimate member of its ensemble.

In cases where belief outdatedness and topology dynamicity are not too high these “split-brain” situations are of temporal nature. For deeper analyses, system simulations

(see Section 4) and timing analysis can be used to provide measurements of the distribution of such inconsistencies and their duration.

Gossip implementation. For our experiments we employed a basic version of counter-based gossiping [6] without emphasis on its optimization, as we did not intend to evaluate the gossip protocol per se but rather the practical feasibility of gossiping ensembles and the impact of the communication boundary. One of such optimizations of the communication that we identified as an absolute necessity was stripping down the size of the disseminated replicas. This is especially critical in MANET settings, where the bandwidth is limited and larger replicas (more than approx. 128 bytes) lead to fragmentation. In combination with the CSMA/CA medium access technique and the hidden node problem [13] this leads quickly to network contention.

7 Related work

The solution presented in this paper brings about the convergence of software component models for CPS and gossip-based communication. Although there are some attempts to achieve synergy between the two areas ([14–16]), they are set on a significantly different track than our approach. In [14], the authors propose a conceptual architecture and design framework for gossip. The framework is based on reusable building blocks, where individual protocols are treated as monolithic black boxes. In [15], the authors propose an API for programming gossip-based systems by analyzing the identified recurrent design dimensions of gossip protocols – namely randomness, neighborhood, and communication. Finally, in [16], the authors introduce a component framework GossipKit, which aims at facilitating the development and testing of gossip protocols by relying on reusable and modular gossip abstractions and standard component-based composition techniques. In all of these approaches the focus is on providing an architectural solution for building gossip-based middleware by means of ready-made components/interfaces. We, in contrast, focus on modeling application logic by means of autonomous components which use gossip internally and partially transparently as the primary means of their communication.

Regarding the state of the art in gossip-based communication, different variations of the basic gossiping scheme have been proposed for different application domains and with slightly different semantics ([17–19]). In MANETs gossiping translates to probabilistic broadcasting within the wireless range of each node [3]. Probabilistic forwarding is often combined with some other locally computable mechanism, such as counter-based [6], location-based [20], distance-based [21], energy-based [22], or a combination of these, to further reduce the number of retransmitted messages (with respect to blind flooding). In our work we do not intend to extend or evaluate the state of the art in gossip-based communication, but provide a method for architecting CPS using abstractions that facilitate the efficiency of the gossip by relying on the architecture-level context information.

Regarding component models and architectures supporting distributed dynamic systems such as CPS, different approaches related to self-adapting/self-organizing systems [23, 24], self-managing architectures [25], component-based architectures [26, 27], and

architectural models at runtime [28] have been proposed. The common denominator of these approaches is the fact that they do not support high dynamicity (which does not scale with the ever-changing landscape of CPS) or they do not readily map to decentralized architectures. DEECo, on the other hand, fits better the specifics of CPS by relying on dynamic component grouping and implicit component communication.

8 Conclusions

In this paper, we presented a synergy of software component model abstractions and gossip-based communication primitives as a promising solution for engineering scalable dynamic decentralized cyber-physical systems. Our approach relies on providing architecture-level descriptions that feature communication groups (captured by communication boundaries) and allow us "driving" the gossip efficiently. The presented experiments show that our approach is in principle feasible. Our current and future work involves improving the scalability of our approach by various optimizations of the gossip protocol (e.g., employing location-based algorithms where GPS-enabled devices are required). Another direction is investigating timing constraints on the gossip-based knowledge dissemination and exchange which will supplement the strict real-time constraints already imposed on local component behaviors.

Acknowledgments. This work was partially supported by the EU project ASCENS 257414 and by Charles University institutional funding SVV-2014-260100. The research leading to these results has received funding from the European Union Seventh Framework Programme FP7-PEOPLE-2010-ITN under grant agreement n°264840.

References

1. Beetz, K., Böhm, W.: Challenges in Engineering for Software-Intensive Embedded Systems. Model-Based Engineering of Embedded Systems. pp. 3–14. Springer (2012).
2. Lee, E.A.: Cyber Physical Systems: Design Challenges. Proc. of ISORC'08. pp. 363–369. Orlando, FL, USA (2008).
3. Friedman, R., Gavidia, D., Rodrigues, L., Viana, A.C., Voulgaris, S.: Gossiping on MANETs: the Beauty and the Beast. ACM SIGOPS Oper. Syst. Rev. 41, 67–74 (2007).
4. Williams, B., Camp, T.: Comparison of Broadcasting Techniques for Mobile Ad Hoc Networks. Proc. of MobiHoc'02. pp. 194–205. ACM, Lausanne, Switzerland (2002).
5. Eugster, P.T., Guerraoui, R., Handurukande, S.B., Kouznetsov, P., Kermarrec, A.-M.: Lightweight probabilistic broadcast. ACM TOCS. 21, 341–374 (2003).
6. Tseng, Y.-C., Ni, S.-Y., Chen, Y.-S., Sheu, J.-P.: The Broadcast Storm Problem in a Mobile Ad Hoc Network. Wirel. Networks. 8, 153–167 (2002).
7. Bures, T., Gerostathopoulos, I., Hnetynka, P., Keznikl, J., Kit, M., Plasil, F.: DEECo – an Ensemble-Based Component System. Proc. of CBSE'13. pp. 81–90. ACM, Vancouver, Canada (2013).
8. Ali, R. Al, Bures, T., Gerostathopoulos, I., Keznikl, J., Plasil, F.: Architecture Adaptation Based on Belief Inaccuracy Estimation. To appear in Proc. of WICSA'14 (2014).

9. Drabkin, V., Friedman, R., Kliot, G., Segal, M.: RAPID: Reliable Probabilistic Dissemination in Wireless Ad-Hoc Networks. In Proc. of SRDS'07. pp. 13–22. IEEE, Beijing, China (2007).
10. Barnat, J., Benes, N., Bures, T., Cerna, I., Keznikl, J., Plasil, F.: Towards Verification of Ensemble-Based Component Systems. To appear in Proc. of FACS'13. Springer, Nanchang, China (2013).
11. Bures, T., Gerostathopoulos, I., Hnetynka, P., Keznikl, J., Kit, M., Plasil, F.: Computational Model for Gossiping Components in Cyber-Physical Systems. Charles University in Prague, TR no. D3S-TR-2014-03.
12. OSI: OSI Basic Reference Model: The Basic Model - ISO/IEC 7498-1, http://standards.iso.org.
13. Yoo, J., Kim, C.: On the Hidden Terminal Problem in Multi-rate Ad Hoc Wireless Networks. Information Networking, v. 3391 of LNCS. pp. 479–488. Springer (2005).
14. Rivière, E., Baldoni, R., Li, H., Pereira, J.: Compositional gossip: a conceptual architecture for designing gossip-based applications. ACM SIGOPS Oper. Syst. Rev. 41, 43–50 (2007).
15. Eugster, P., Felber, P., Le Fessant, F.: The "Art" of Programming Gossip-based Systems. ACM SIGOPS Oper. Syst. Rev. 41, 37–42 (2007).
16. Taiani, F., Lin, S., Blair, S.G.: GossipKit: A Unified Component Framework for Gossip. IEEE Trans. Softw. Eng. PP, 1–17 (2013).
17. Branco, M., Leitão, J., Rodrigues, L.: Bounded Gossip: A Gossip Protocol for Large-Scale Datacenters. Proc. of SAC'13. pp. 591–596. ACM, Coimbra, Portugal (2013).
18. Khelil, A., Suri, N.: Gossiping: Adaptive and Reliable Broadcasting in MANETs. Dependable Computing, v. 4746 of LNCS. pp. 123–141. Springer (2007).
19. Kermarrec, A.-M., Van Steen, M.: Gossiping in distributed systems. ACM SIGOPS Oper. Syst. Rev. 41, 2–7 (2007).
20. Karp, B., Kung, H.T.: GPSR: Greedy Perimeter Stateless Routing for Wireless Networks. Proc. of MobiCom'00. pp. 243–254. ACM, Boston, USA (2000).
21. Cartigny, J., Simplot, D.: Border Node Retransmission Based Probabilistic Broadcast Protocols in Ad-Hoc Networks. Proc. of HICSS'03. pp. 303–312. IEEE, Hawaii, USA (2003).
22. Miranda, H., Leggio, S., Rodrigues, L., Raatikainen, K.: A Power-Aware Broadcasting Algorithm. Proc. of PIMRC'06. pp. 1–5. IEEE, Helsinki, Finland (2006).
23. Serugendo, G.D.M., Fitzgerald, J., Romanovsky, A.: MetaSelf – An Architecture and a Development Method for Dependable Self- * Systems. Proc. of SAC'10. pp. 457–461. ACM, Sierre, Switzerland (2010).
24. Liu, H., Parashar, M., Hariri, S.: A Component Based Programming Framework for Autonomic Applications. Proc. of ICAC'04. pp. 10–17 (2004).
25. Kramer, J., Magee, J.: Self-managed systems: an architectural challenge. Proc. of FOSE'07. pp. 259–268. IEEE, Minneapolis, USA (2007).
26. Baresi, L., Guinea, S., Tamburrelli, G.: Towards Decentralized Self-adaptive Component-based Systems. Proc. of SEAMS'08. pp. 57–64. ACM, Leipzig, Germany (2008).
27. Peper, C., Schneider, D.: Component engineering for adaptive ad-hoc systems. Proceedings of SEAMS '08. pp. 49–56. ACM, Leipzig, Germany (2008).
28. Morin, B., Barais, O., Jezequel, J.-M., Fleurey, F., Solberg, A.: Models at Runtime to Support Dynamic Adaptation. Computer (Long. Beach. Calif). 42, 44–51 (2009).