ABSTRACT
The emerging area of (smart) Cyber Physical Systems (sCPS) triggers demand for new methods of design, development, and deployment of architecturally dynamic distributed systems. Current approaches (e.g. Component-Based Software Engineering and Agent-Based Development) become insufficient since they fail in addressing challenges specific to sCPS such as mobility, heterogeneous and unreliable deployment infrastructure, and architectural dynamicty. The strong dependence on the underlying communication infrastructure, often combining ad-hoc established links typical for wireless connectivity with more reliable connections of infrastructural networks, requires a novel method to optimize system deployment. In this paper we propose such a method based on the domain knowledge elicited from design level specification. As a proof of concept, we have provided an extension to the DEECo (Dependable Emergent Ensembles of Components) model and validated it on a scenario from the domain of Vehicular Area Networks.

Categories and Subject Descriptors
C.2.4 [Computer-Communication Networks]: Distributed Systems – distributed applications; D.2.10 [Software Engineering]: Design; D.2.11 [Software Engineering]: Software Architectures;

Keywords
Cyber-physical systems; Domain knowledge; Component communication

1. INTRODUCTION
Recent growth in connectivity of electronic devices results in the birth of new kind of distributed systems regarded often as the Internet of Things (IoT) or as (smart) Cyber Physical Systems (sCPS) [1]. There are already multiple examples of such systems, stemming from different usage domains such as assisted living, intelligent transportation, and (mobile and ad-hoc) cloud computation. They are usually composed of autonomous components designed to execute independently in order to support resilience of the system. Nevertheless, components need to communicate with each other, exchanging data that allows them to perform cooperative actions. The way components interact together along with the conditions under which their interaction occurs are usually grasped by an architecture description articulated during the design phase.

After being implemented, components are deployed to physical nodes interconnected by the means of ad-hoc and/or infrastructure networks, each of these requiring a dedicated approach towards data dissemination.

Problem statement: Depending on its scale, deployment of sCPS may involve a large number of physical nodes with different communication infrastructure. To optimize utilization of a network, some information about the communication aspects should be introduced yet at the design level when the application architecture is decided. This, however may violate infrastructural transparency needed to allow for deployment independency. In the end the challenge here is to find a solution that would sustain communication transparency at the architectural level and, at the same time, allow for its optimization at the infrastructure level.

Goal: This paper aims to propose a method targeting the problem above and to show its feasibility on a case study implemented in the existing DEECo (Dependable Emergent Ensembles of Components) [2] component model and framework. Specifically, we address the problem by introducing communication groups based on adding domain-specific knowledge to the architecture; this allows optimization of the network use while preserving the level of abstraction typical for architectural design.

The rest of the paper is structured as follows: Section 2 describes motivating example and overviews the background technology – DEECo. Section 3 introduces the idea of using domain specific knowledge for communication optimization and in Section 4 the benefits of the idea are evaluated. Section 5 discusses potential applications of the communication groups. Section 6 focuses on related work while Section 7 concludes the paper by summarizing its contribution.
2. MOTIVATION AND BACKGROUND

2.1 Road - trains scenario

As a running example, consider a scenario of emergency vehicles forming road-trains (a chain of vehicles heading towards the same destination). The purpose of a road train is to optimize movement (in terms of speed, safety, and traffic disruption) of emergency vehicles towards the site of an accident. We assume that each vehicle is equipped with the necessary hardware enabling both short-range wireless communication (via MANETS) as well as infrastructure-based connectivity (long range, dedicated to emergency services). Vehicles within a single road train communicate in order to maintain proper internal organization of the train and to ensure satisfaction of the safety requirements such as minimal distance between vehicle, maximal speed etc. Furthermore, all vehicles (also across different road trains) exchange information necessary to form a road train, including desired destination, current location etc. The scenario together with two types of data flows is illustrated in Figure 1.

In this scenario, we focus on the organization of vehicles seen as autonomous components that need to communicate globally (to form a road train) and locally (while in a road-train). Whereas the local coordination requires low latency in data exchange, which is achieved by short-range communication, global coordination accounts for optimality in terms of network utilization.

2.2 Background: DEECo

Proposed for development of dynamic CPS, the DEECo (Dependable Emergent Ensembles of Components) component model and its framework was introduced [2] for designing applications of autonomous components and their dynamic ad-hoc groups (ensembles) that the components establish serving for their communication.

A DEECo component is an independent unit of computation and deployment. In the scenario, components correspond to the main actors of the system (i.e. vehicles). The template of these components is specified in Figure 2 by the Vehicle specification. Its state is captured by knowledge (a set of attributes - lines 7-14) and operational functionality by processes (lines 15-20). Every component features a number of roles, i.e., sets of knowledge fields (lines 1-4, 6), which provide contract between the component and ensembles. Processes are executed by the runtime periodically or in a triggered manner. Line 19 demonstrates a specification of periodic execution of the processes with a given time period. Each process execution starts with atomic reading (a part of) of component knowledge, executing the process body, and ends with atomic writing the results back to the knowledge.

In Figure 2, ensembles reflect the two types of communication groups of vehicles - within a road-train and across all the vehicles heading to the same destination. For instance, consider the SameDestination ensemble definition (lines 22-34). The goal here is to propagate information about the vehicle’s desired destination to other vehicles so that they can coordinate movement to form road-trains. The figure illustrates that an ensemble definition in DEECo contains a membership condition specifying which components can join the ensemble (lines 25-28), and a prescription of knowledge exchange between its coordinator and members (lines 29-33). The coordinator and potential members are characterized by specific roles (lines 23-24). An ensemble is instantiated and dissolved by the DEECo runtime framework, which periodically (line 34) checks the membership condition. Whenever the ensemble is formed (i.e. there is a coordinator and at least one member), the runtime framework periodically performs

Figure 2: Examples of DEECo component and ensembles of the road trains scenario

the knowledge exchange by transferring data form the members to the coordinator (and vice versa) as specified by the mapping in the ensemble definition. A component specification may feature multiple roles; consequently, a component may be a member/coordinator of many ensembles at a time.

It should be emphasized that, knowledge exchange, realized by the ensembles to which a particular component belongs, is the only means of inter-component communication.

3. COMMUNICATION EMPLOYING DOMAIN KNOWLEDGE

Tailored for development of sCPS, the DEECo component model allows designing a system at the architecture level without considering aspects related to its actual deployment - component
distribution, communication infrastructure, and even its scaling in terms of the eventual number of component instances. Such an abstraction level simplifies modeling and development of the system, as it allows reasoning about components and ensembles in isolation, a crucial property when dealing with complex systems. Problems arise when it comes to deployment of the system, since there is a gap between the abstraction level of design and runtime infrastructure. This typically implies the need to apply standard generic methods for communication among distributed nodes. In particular in sCPS the efficiency of communication can be substantially improved by employing application domain data to optimize the deployment of the system.

In this section, we present how this can be achieved in DEECo by employing the concept of communication boundary [3], and, as a key contribution, the novel idea of communication groups.

### 3.1 Ad-hoc Networks

DEECo and specifically its Java realization jDEECo [4], supports ad-hoc communication via MANETS. It relies on periodic channel-level broadcasts (rebroadcasts) of component knowledge. In a system, this allows a node not only be aware of the knowledge of the components it hosts but also to learn about knowledge of other (remote) components. This approach is appropriate for MANETS that are not fully reliable and prone to frequent disconnections due to radio interference and mobility of nodes.

The communication protocol in jDEECo is based on bounded gossiping [3], where components’ knowledge rebroadcasting is limited by communication boundaries articulated in ensemble specifications. A communication boundary is employed by a node for deciding whether or not to rebroadcast the component knowledge heard from other nodes. This way, by constraining component knowledge dissemination to a specific geographical area, this mechanism allows better utilization of the communication channel, which in wireless settings comes at a great price.

An example of communication boundary is given in Figure 2 (lines 48-52) when the component knowledge data dissemination is bounded to the nodes of the vehicles participating in the same road train.

The specification of a communication boundary, is given as a predicate formulated on the component knowledge to be rebroadcasted and the knowledge of a rebroadcasting node. This way, the communication boundary reflects only the application domain-specific knowledge known at the architectural level. (Specifically, no information about future deployment is required.) In case of Figure 2, the domain-specific knowledge captures the fact that a road-train is a spatially connected structure and thus it is sufficient to involve only the road-train nodes in the rebroadcasting. As an aside, this is the only enhancement to the original semantics of the ensemble as specified at the architectural level.

### 3.2 Infrastructure Networks

The benefits of the communication boundary are apparent for ad-hoc networks; nevertheless the idea is also applicable when dealing with more reliable communication infrastructure networks (IN for short). In such settings, however, one can do more than just restrict data retransmissions. Having a topology that does not change often (in particular if established links hold for a relatively long time), a routing mechanism can be introduced to provide for optimality with respect to, e.g., bandwidth utilization, latency, and computation balancing.

Therefore, jDEECo utilizes gossiping in case of infrastructure networks [5]. As a data dissemination protocol, gossiping is resilient to communication failures. Nevertheless, depending on the application, standard gossiping may still be costly, especially in terms of the amount of data being transmitted. Specifically, in jDEECo standard gossiping causes that component knowledge is published periodically to all nodes in a system, which does not scale well for large-scale systems.

**Communication groups:** To mitigate the problem of unnecessary data transmission over the network, we propose an extension to ensemble definition by introducing communication group, delineated according to the component knowledge of the coordinator and members of an ensemble. The basic idea is to introduce the groups of components (members) that are related to each other in terms of a specific knowledge value (e.g. having the same value of the destination attribute). Such a group serves as a hint for optimizing deployment in terms of communication efficiency by restricting and localizing the area in which discovery of components to form an ensemble is performed. Defined again at the architecture level via component knowledge specified in roles, orthogonally to the membership, knowledge exchange, and communication boundary, the concept of communication group just enhances the original semantics of ensemble, not modifying the meaning of other DEECo abstractions. For illustration, consider line 4 in Figure 3, indicating that communication groups will be based on the destination value in the coordinator’s knowledge. In this case vehicles going to the same destination (expressed by the membership condition) compose communication groups, each of them corresponding to a specific value of the destination attribute in the coordinator’s knowledge. The situation is visualized in Figure 4, where ensembles of different emergency vehicles trains are heading to distinct locations in Prague 6 and in Prague 4 districts.

**Groupers:** Communication groups are utilized to optimize deployment, where they support the planning of inter-node communication links. For that reason an extension to the jDEECo runtime environment is proposed by introducing the concept of grouper. The basic idea is that a grouper limits the gossiping only to the nodes that host the components belonging to a particular communication group. Thus a grouper employs the communication group specification. Technically a grouper enhances the jDEECo runtime environment in the following way. The environment contains a set of jDEECo runtime instances (Figure 5). Each of them hosts a set of components, the knowledge of which is gossiped around, using the addresses of other nodes stored in its recipient table. In the enhancement, a grouper can also be referenced in the recipient table as illustrated in Figure 5. It is assumed that a grouper (i) is a representative of a communication group(s), (ii) is equipped with all the ensemble definitions in the system, (iii) has access to knowledge of all components needed to evaluate the membership

```
1. ensemble SameDestination:
2.   coordinator: OtherVehiclesAggregator
3.   member: OtherVehiclesAggregator
4. communication group: coordinator.destination.CityDistrict
5. membership:
6.   member.1destination.Address = coordinator.1destination.Address
7.   AND (member.isRoadTrainMember
8.   AND (coordinator.isRoadTrainMember
9. knowledge exchange:
10.   coordinator.1otherVehicles ← (m.id, m.position) | m ∈ members
11.   for(m ∈ members)
12.     m.1otherVehicles ← {t ∈ coordinator.1otherVehicles | T.ID ≠ m.ID}
13.     m.1otherVehicles ← (coordinator.ID, coordinator.position)
14. scheduling: periodic(700ms)
```

**Figure 3:** An example of communication group specification
conditions, and (iv) can modify the recipient tables that contain a reference to it. The basic functionality of a grouper is to continuously monitor the current ensemble memberships of all the components in the group it represents and update the recipient tables accordingly. By modifying the recipient tables, a grouper implicitly routes the component knowledge to the most relevant nodes (i.e., hosting the same ensemble components) in the network. Figure 5 exemplifies the whole idea on the SameDestination ensemble, the communication group of which depends on the destination field in the coordinator’s knowledge. In this case the grouper dedicated to Prague 6 continuously monitors the ensemble membership status of all the components it is aware of. If necessary, it updates the recipient tables of the respective jDEECo runtime instances on the nodes hosting the components in the SameDestination ensemble. The specification of the communication group allows a node to determine the groupers for a given pair of an ensemble type and a component hosted on the node. This way the knowledge of the component is routed only to a limited set of groupers (as opposed to being propagated throughout the whole system). Also, as communication groups are typically geographically or network-wise localized, they lead to a decentralized solution, potentially characterized by low-latency. The decentralization also means that the operation is possible even in case that the infrastructure network gets partitioned into a number of disconnected subnets (i.e., without global internet connectivity).

4. EVALUATION

4.1 Proof of concept

As a proof of concept we have simulated the road trains scenario and conducted several experiments, allowing us to assess the applicability of the method\(^1\). We used the total number of messages exchanged in the system as a metric for expressing communication efficiency. The simulation, conducted with use of MATSim [6], was focused on optimization of emergency vehicles’ routings across realistic road network of the Prague city provided by OpenStreetMap [7]. Firefighter, police, and ambulance vehicles were considered as the emergency vehicle types. The locations of ambulance, police and firefighter bases were set according to their real locations. For simplicity, all non-road objects and several minor roads were removed from the original map which yielded a road network covering the area of approximately 100km\(^2\).

The simulation comprises three groups of experiments: (i) emergency call response by 3 vehicles, (ii) emergency call response by 5 vehicles, and (iii) single large road train (convoy with the right of the way). The groups (i) and (ii) encompass experiments differentiated by number of concurrent emergency calls (1, 2, 3, 5, 10, 15, 20), while (iii) encompasses experiments with several road train sizes (3, 5, 10, 15, 20).

As to (i) and (ii), when an emergency call is issued (e.g., a serious car crash), vehicles are dispatched to the accident site (destination). In the simulation, the emergency vehicles heading to the same destination aim at forming a road-train to make it easier to clear their path in heavy traffic by driving closely behind each other. The emergency vehicles are dispatched from the emergency service bases as close to the destination as possible. Specifically it is assumed that: in (i) one of each emergency vehicle type is sent to every destination, in (ii) two ambulance, two firefighter and one police vehicles are sent to every destination, in (iii) emergency vehicle types are not distinguished.

Once a vehicle is on its way to the accident site, it aims at following another emergency vehicle heading to the same destination. A road train is established, when the distance between two solo vehicles heading the same destination is negligible in a street. A vehicle is allowed to join a road train only when its prolongation of its route to the destination is minor and has the ability to increase its speed temporarily.

In order to show that the results do not depend on particular routing and destination choice, 10 different simulation runs parametrized by destination choice were executed.

4.2 Experiment results and lessons learned

In (i) and (ii) a key result of these sets of experiments is the proof of communication complexity reduction (from quadratic to linear). Recall that this complexity metric is the number of IN messages — in our case those were the IP messages. For the group (i) the number of IP messages was measured (Figure 6); for the group (ii) this

---

\(^1\) Source code of the scenario implementation used in the experiments is available at: [http://github.com/d3scomp/cbse-2015-tutorial](http://github.com/d3scomp/cbse-2015-tutorial)
measurements are in (Figure 8). The reason for not considering MANET messages is that these are local and thus not influencing the infrastructure network load, even though small fraction of these is inherently rebroadcasted in MANET network. From these figures, it follows that when just gossip is applied, the number of IP messages grows quadratically with the number of vehicles. This is caused by the fact that IP messages from a vehicle are sent to all of other vehicles. On the contrary, when groupers are applied the IP messages are sent only to the vehicles sharing a particular destination, the number of IP messages is linear in number of groups while assuming the size of the group is constant. Moreover, here it is also visible that the effect of improvement starts at a minimal number of destinations (such as 3 in Figure 6), since there is an overhead of communication among groupers.

Note that a system that enables message passing between the MANET and IN networks (such as jDEECo originally) needs to be configured in such a way that messages from different communication groups do not leak from one communication group to another, otherwise this would harm the positive effect of communication groups. As an aside, in this simulation this was ensured by preventing rebroadcasting of IP messages by MANET.

Finally, domain specific knowledge can be further exploited by evaluating ensemble membership conditions in groupers. Such a feature would enable distribution of knowledge only to those nodes that host components satisfying a particular ensemble membership condition. In the road trains scenario (Figure 3), a vehicle that is a member of a road train is not a member of an instance of SameDestination any more, thus not being subject to the respective knowledge exchange, since only the “solo” vehicles and road-train leaders need to communicate via IN network. Therefore, thanks to ensemble membership condition evaluation in groupers, it is possible to exclude those vehicles from communication group. The effect of this optimization would be minimal in experiment groups (i) and (ii), since the road-trains considered are relatively short. In order to study this effect, the group (iii) was introduced. From Figure 7 it is clearly visible that introducing groupers evaluating ensemble membership condition further reduces the number of IP messages for larger road trains. (Note that due to a higher variance of results in Figure 7, we use box-plots instead of simple points. Results in Figure 6 and 7 exhibit very low variance; thus we show only the mean values.)

5. DISCUSSION

The idea of the communication groups is applicable in most sCPS, nevertheless it shows its full strength when a combination of both ad-hoc and infrastructure networks takes place. Advantageously, it accounts for infrastructural dynamicity and mobility of network nodes in particular. In the scenario described in Section 2, the communication groups are for simplicity static in the sense that when a vehicle is assigned its destination, it is rather unlikely that this will be modified. However, the approach equally supports situations where a communication group depends on a dynamically changing factor such as current position of a vehicle. For instance, it is possible to specify the group in such a coarse way that the current position of the vehicle is refers just a particular Prague districts. Then, depending on the current position of the vehicle, the jDEECo runtime instance - grouper associations would vary over time, optimizing the network traffic (with respect to data latency and amount of data sent) via the nearest grouper in the district.

While communication boundary constrains the actual communication topology, communication groups provide for (context-aware) routing mechanisms. These two concepts complement each other, and in case of infrastructure networks can be used either separately or in combination, effectively providing for different scenarios of component knowledge dissemination. Communication group offers more flexibility in terms of the possible optimization strategies to be implemented during deployment process. Depending on non-functional requirements, deployment optimizing network traffic in terms of data latency can be achieved. In such cases, it is desirable to bring groupers as close as possible (considering geographical distance) to the relevant nodes. On the other hand, if the main concern is balancing (or optimizing) the utilization of computational resources, then the deployment would be based mainly on their availability. Such flexibility is useful in heterogeneous deployments, where a part of the network is to be latency sensitive and another latency tolerant.

Figure 6: Communication complexity comparison of gossip and groupers; experiment group (i) – 3 vehicles per accident

Figure 7: The effect of introducing groupers evaluating ensemble membership condition; experiment group (iii) – single large road train

Figure 8: Communication complexity comparison of gossip and groupers; experiment group (ii) – 5 vehicles per accident
As an example consider imposing additional requirement to the scenario from Section 2 that intra-train communication should be latency sensitive and the communication between road trains latency tolerant.

6. RELATED WORK

The idea of communication groups relates to Distributed Hash-Tables (DHT) which introduce key-space partitioning [8], [9] and overlay networks [10]. The former assign a range of keys to particular network nodes that take the responsibility for storing the actual value corresponding to a key. The latter store references to other nodes to allow each node to query another node during key lookups. In this context communication groups can be partially interpreted also as a key-value storage problem, where the key is the particular value of the communication group specification given in the ensemble DSL definition, while the value is the set of components forming the ensemble. Other commonalities of communication groups and DHT include: implementation transparency, topological dynamism and redundancy mechanisms (increasing the overall reliability of the system). All in all, DHT may serve as supporting technology for implementing the part of communication group functionality that provides a mapping between a particular group and a set of network nodes. The main difference thus lies in the level of abstraction, where communication groups are primarily an architectural concept, while DHTs belong to middleware.

In terms of benefits of communication group, they go along the same lines as fog computing (also edge computing). Fog computing extends the concept of cloud computing by pushing the data and computation from centralized nodes (data centers) closer to end devices – i.e. to the edge of the network [11], [12]. Similar to cloud, fog provides data, computation power, and networking services more likely in a latency-free manner. Basically, communication group resembles the idea of fog computing of coping with demand for low latency but does it by different means (employing domain knowledge in particular).

With respect to exploiting domain-specific knowledge to improve network utilization communication group resembles context-aware routing protocols [13], a technique used in wireless (mesh) networks or delay-tolerant mobile ad hoc networks. It uses various information from the environment (context) to discover optimal path from the source to a destination or to adapt to changes in network topology. In [14], the authors propose a method building on node mobility and the history of establishing links with other nodes including their location. In a similar vein, geographic routing (or geo-routing) [15] relies on the geographical position of nodes. In addition to classical packet addressing, it also employs indication of the actual geographic position of a target node. The concept of communication groups takes the idea of context-aware routing a step further. Driven by application domain knowledge, it is more flexible with respect to the information being exploited in order to provide for more accurate addressing. Even though context-aware routing is fully distributed, communication groups are dedicated to infrastructure networks (see Section 3.2), where communication links are relatively stable and reliable, making the centralization aspect of groupers not a big issue.

7. CONCLUSION AND FUTURE WORK

In this paper, we have presented a DEECo dedicated method that introduces communication groups exploiting application domain knowledge in order to optimize communication infrastructure utilization. As an extension to the DEECo model, communication groups increase efficiency of sCPS with respect to their deployment. The method relies on providing architecture-level descriptions that define communication groups and allow component knowledge routing according to custom preferences (latency sensitivity, resource utilization). Our current and future work involves adding features such as key partitioning and improved data exchange to groupers. Further, we plan to apply OMNet++ based simulations to obtain measurements reflecting network latency.

8. ACKNOWLEDGMENTS

This work was partially supported by the EU project ASCENS 257414 and by Charles University institutional funding SVV-2015. 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.

9. REFERENCES