The An Binh Nguyen, Christian Meurisch, Stefan Niemczyk, Doreen Böhnstedt, Kurt Geihs, Max Mühlhäuser and Ralf Steinmetz. *Adaptive Task-Oriented Message Template for In-Network Processing*. In Proc. International Conference on Networked Systems (NetSyS), p.1 - 8, IEEE 2017. ISBN 978-1-5090-4394-1

Adaptive Task-Oriented Message Template for In-Network Processing

The An Binh Nguyen*, Christian Meurisch†, Stefan Niemczyk‡, Doreen Böhnstedt*,
Kurt Geihs‡, Max Mühlhäuser†, Ralf Steinmetz*
*Multimedia Communications Lab, TU Darmstadt, Rundeturmstr. 10, 64283 Darmstadt, Germany
Email: {the.an.binh.nguyen,doreen.boehnstedt,ralf.steinmetz}@kom.tu-darmstadt.de
†Telecooperation Lab, TU Darmstadt, Hochschulstr. 10, 64289 Darmstadt, Germany
Email: {meurisch,max}@tk.tu-darmstadt.de
‡Distributed Systems, University of Kassel, Wilhelmshöher Allee 73, 34121 Kassel, Germany
Email: {niemczyk,geihs}@uni-kassel.de

***Abstract*—In disaster situations or on emergency terrains, Internet and Cloud access may be restricted; it may still be important to process complex resource-intensive tasks and to acquire distributed information for emergency response, using ad-hoc networks among, e.g., first responder mobile devices. Corresponding approaches towards coordination, resource utilization, and interoperability are still challenging. This paper introduces the concept of adaptive task-oriented message templates (ATMT) as a basis for overcoming these issues and for enabling cooperative in-network processing without additional synchronization overhead for mobile devices. An ATMT serves as a self-encapsulated message containing the operation chains that need to be executed as well as the required data. In order to address heterogeneity and interoperability issues, we integrate a lightweight ontology. Depending on the current utilization, devices can autonomously decide whether to participate in the network or not. We evaluate our approach in an indoor testbed with 8 wireless mesh nodes. The results confirm that our approach efficiently supports cooperation among heterogeneous devices towards utilizing available in-network resources while reducing network traffic.**

I. Introduction

In recent years, the development and the proliferation of networking mobile devices have enabled the vision of Internet of Things (IoT) [1]. The IoT paradigm leverages various heterogeneous devices with specialized sensors to capture their environments as well as powerful hardware which enables processing of the sensor data locally [2]. This supports the development of new application types, e.g., smart cities. Due to the limited battery lifetime of these mobile devices, many IoT research applications rely on a centralized approach, where a cloud platform acts as central management or computational offloading unit [3]. However, in emergency situations such as disaster scenarios with limited or no access to the cloud due to impaired infrastructures, other techniques for decentralized communication and computational analysis of sensor data are required to support emergency response operations [4].

In the literature, several techniques on collecting and processing data directly in the network (termed *in-network processing*) have been proposed, especially in the research field of *wireless sensor network* (WSN) [5]. However, in-network processing for WSN focuses on resource-constrained sensor motes and only considers simple aggregation operations such as max, min, or average. In contrast, the available resources (e.g., smartphones, robots, or home routers) in emergency scenarios enable the in-network processing of complex multi-stage operations such as voice or image processing. State of the art works like [6] or [7] show how mobile devices are able to create an ad-hoc network and efficiently exchange data. However, none of the related works addresses delay-tolerant in-network processing of complex multi-stage data analysis in mobile ad-hoc mesh networks.

In this paper, we present an adaptive task-oriented message template (termed *ATMT*) as well as an entire set of possible operations and associated roles to address the issues of emergency response. The message template supports delay-tolerant in-network processing and allows participatory mobile devices to autonomously cooperate for accomplishing complex multi-stage tasks. We further integrate the *ICE ontology* (Information processing self-Configuration and Exchange) [8] to ensure interoperability of heterogeneous devices.

The major contributions of this paper are threefold:

- We propose an *adaptive task-oriented message template* that allows devices to dynamically participate in the delay-tolerant in-network processing. Our message template is self-encapsulated, logically separated and hierarchically constructed using direct acyclic graph representation. We also provide an entire set of graph operations to efficiently handle or modify messages as well as a system model with associated roles.
- To enable interoperability of heterogeneous devices, we present and integrate the *ICE ontology*.
- We provide a *reference implementation* supporting the proposed operations and associated roles, which is evaluated in a small testbed with 8 mesh nodes. Our results show the feasibility of our message template and its applicability in mobile ad-hoc mesh networks.

The remainder of this paper is organized as follows. First, we give an overview of the related work. Second, we describe the design of our proposed message template and the system model. To enable interoperability for heterogeneous participa-

tory devices, we briefly elaborate on our used ontology. The paper closes with evaluation of the reference implementation, result discussion and conclusion.

II. Related Work

In this section we discuss the related work in the relevant research disciplines, namely *in-network processing*, *service discovery*, and *service composition* in ad-hoc mesh networks.

A. In-network Processing

To distribute or offload the computational workload of resource-limited mobile nodes and to reduce network traffic, *in-network processing* has been introduced. For instance, in wireless sensor networks (WSN), data collected by different sensors can be aggregated within the network using simple functions such as filter, min, max, or average to reduce the amount of redundant information [5]. Other research works in WSN address network, data format or application representation issues. Most of the existing WSN works regarding network focus on tree-based routing, or on clustering to enhance the network topology for aggregation. However, these approaches are designed to work within a stable network and the performance decreases in dynamic mobile networks [5]. Only a few works target data formats and application representation for in-network processing [9]. Under the emergence of IoT devices, the abstraction on the application layer for WSN in-network processing gained attention, e.g., *T-RES* [10]. T-RES utilizes the Constraint Application Protocol (CoAP) to support dynamic reconfiguration for WSN in-network processing. Tasks are modeled as resources on each node. The REST-like resource requests of CoAP support to model an in-network processing chain in form of input, processing function and output destination. Wang et al. introduce *CS-Man* [3], which utilizes Named Data Network (NDN) to manage service deployment: services are represented in a hierarchical dashed name structure, and service providers have to actively advertise the services to a service manager. However, T-RES and CS-Man are designed to work in rather stable networks.

B. Service Discovery

Service discovery comprises several aspects such as service description, discovery protocol and architecture [11]. Common service description formats (e.g., WSDL[1]) can only be used in stable networks. In scenarios where the connection among devices is intermittent, the service description should be lightweight to cope with the frequently changing environment. *DEAPspace* [12] - a framework targeting service discovery for an 1-hop ad-hoc network - uses a hierarchically structured service description including name, attribute, input, output and service details. In contrast, *Chitchat* [7] provides a context representation suite based on Bloom filters to further reduce the message size and to facilitate the context exchange between devices. Despite the reduction of representation size, *Chitchat* introduces false positives, which is not desired in terms of task allocation. Discovery architectures often rely on distributed services directory and require that the service providers actively advertise their services [13]. However, such approaches are infeasible to apply in a highly dynamic environment, where the providers and their offered services change frequently.

[1]https://www.w3.org/TR/wsdl (accessed: 2016-12-01)

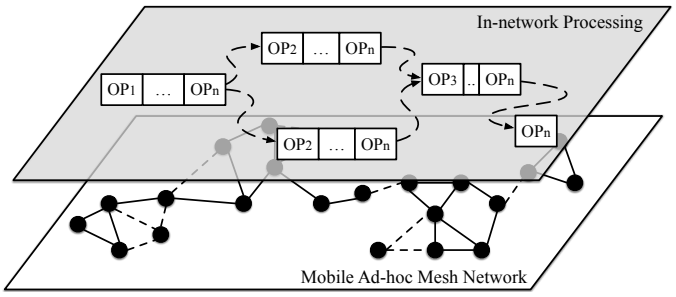


Fig. 1: Mobile ad-hoc mesh network consisting of heterogeneous (mobile) devices that piecewise process the task-oriented messages propagating through the network

C. Service Composition

A *service composition* is an aggregation of services collectively composed to execute specific tasks. In mobile ad-hoc networks, most research on services composition focuses on designing algorithms and protocols to solve the graph mapping problem (e.g., [14], [15]). In [14], the authors follow a demand-based approach for service composition, i.e., the services are dynamically activated at runtime. This approach requires synchronization to obtain the service overlay. In [15], the authors propose the construction of a goal-oriented service composition workflow. The service composite is built by traversing the service overlay graph. Following an auction-based approach, the initiator coordinates the allocation of individual services in the composite workflow. However, such an approach is not suitable for a dynamic environment.

III. Adaptive Task-oriented Message Template

In this section, we first elaborate on two scenarios to show the need for our proposed message template. Afterwards, we analyze the requirements and describe the design of the template and the system model that enables autonomous participation for in-network processing.

A. Application Scenarios

Figure 1 shows the network scenario targeted by our proposed message template. The heterogeneous mobile devices form an ad-hoc mesh network to exchange and process the task-oriented messages, which propagate through the network.

In emergency situations, the communication between devices and to remote services is limited. Nevertheless, the devices can form a decentralized ad-hoc mesh network to recover and maintain communication [6]. However, this type of network makes the coordination of rescue operations more difficult. Instead of transferring all data to a central server, the processing should be performed in a decentralized manner. Therefore, operation chains describing the required processing steps can be defined by the authorities, such as fire services.

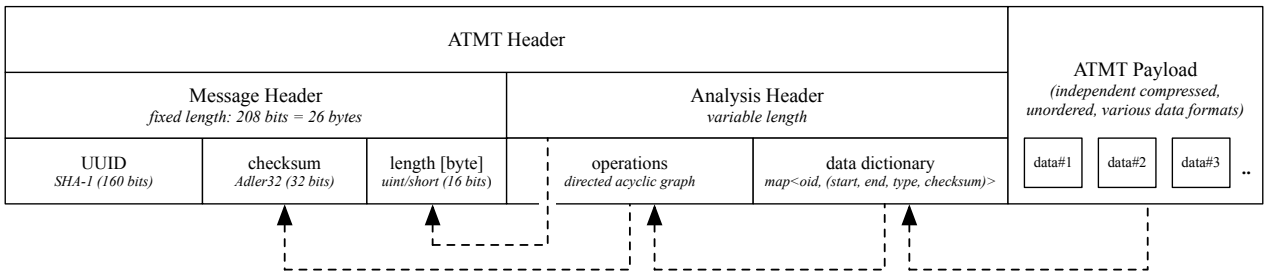


Fig. 2: Adaptive Task-oriented Message Template (ATMT) for in-network processing

For instance, victim detection based on thermal images can be defined as the following operation chain: *edge detection* → *object detection* → *victim detection*. The thermal image is provided by a rescue robot equipped with an infrared camera; the resulting image and the operation chain are combined to a message and forwarded to the network. This enables the processing by participatory devices within the network. The first operation *edge detection* extracts the required features for *object detection*. Afterwards, *object detection* is used to extract objects. The last operation, i.e., *victim detection* applies heuristics to determine if an object is a victim. The fire services will then receive only the final result. Since the operations required for the example chain are complex, not all devices are able to perform these. The propagation of an operation chain through the network allows to find capable devices, which are able to execute specific operations; thus, contribute to the processing chain or to the overall analysis.

Another example is crowd density estimation based on audio processing techniques (e.g., [16]). Crowd density estimation is an important task in crowd monitoring, which is often used to plan an evacuation in emergency situations. The operation chain for this example looks as follows: *speech detection* → *feature extraction* → *speakers counting*. The audio data as input for this chain are provided by a participatory device. The operation chain can be executed by the same device or by other devices in the network. The operation *speech detection* uses pitch estimation to filter noise in the background. The *feature extraction* operation determines feature vectors from the extracted speech data. The last operation - *speakers counting* - uses a forward clustering algorithm to count the number of speakers based on the feature vectors. Each operation requires devices with specific capabilities or high computational resources, which could be rare in emergency situations. Again, the propagation of the operation chain through the network helps to deal with this issue.

The examples highlight two crucial aspects of in-network processing in extreme situations such as emergency response: first, the discovery of resources in decentralized ad-hoc networks and, second, the exploitation of the discovered resources.

B. Requirement Analysis

Our work addresses open issues of decentralized in-network processing by defining a task-oriented message template. The template allows devices to cooperate with each other in order to execute a complex operation chain. We consider an IoT landscape, where heterogeneous devices are connected within a dynamic ad-hoc network. This network permits the devices to directly exchange messages. In the following, we describe requirements for a task-oriented message template:

- *Communication:* due to resource restrictions of participatory devices, the communication and coordination overhead should be as small as possible.
- *Mobility:* due to the high mobility of the devices, joining the network and contributing to the processing chain in an autonomous way should be easy.
- *Heterogeneity:* due to heterogeneity of participatory devices ranging from sensor motes over smartphones to powerful stationary cloudlets, the approach should address and exploit the capabilities of these devices.
- *Computation:* due to the battery limitations of mobile devices, the computational overhead or the resource usage should be as low as possible.

In summary, a task-oriented message should be adaptive, self-descriptive, and computing- and communication-efficient.

C. Task-oriented Message Construction

Meeting the above defined requirements, we design the task-oriented message template (namely *ATMT*) for mobile ad-hoc networks. A task-oriented message enables the cooperation of networking devices without additional coordination overhead.

Figure 2 shows the ATMT structure to construct a single self-encapsulated message consisting of a *header* and a *payload* part. The header cuts into two parts: the *message header* with a fixed length of 208 bits and the *analysis header* with a variable length depending on the amount of required operations. The message header consists of the 160-bit universal unique identifier (*UUID*) using SHA1[2], the operation *checksum* using Adler-32[3] algorithm, and the *length* of the variable analysis header. With the message UUID, participatory devices are able to identify, track, and distinguish received messages using fast bitwise XOR comparison without central coordination. Messages with different identifiers are treated separately. Otherwise, the checksum, which represents the current analysis state of the message (similar to a revision

[2]https://tools.ietf.org/html/rfc3174 (accessed: 2016-12-01)
[3]https://tools.ietf.org/html/rfc1950 (accessed: 2016-12-01)

TABLE I: Definition of common operations (termed *CRUD*)

Abbr.	Operation	Description
C	**(C)**reate	Create a new message with sensor data attached
R	**(R)**ead	Parse the message and stream its content
U	**(U)**pdate	Modify the message, i.e., its operation graph and content (cf. Table II)
D	**(D)**elete	Delete the message from the network

TABLE II: Definition of update operations (termed *MODS*)

Abbr.	Update op.	Description
U_M	**(M)**erge	Consolidate two messages by merging two operation graphs and their attached data
U_O	**(O)**perate	Execute an operation on the message's data
U_D	**(D)**elegate	Add a new operation to the operation's graph
U_S	**(S)**plit	Divide the message into two messages

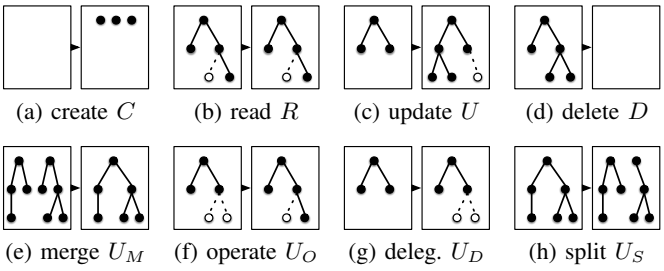
number), is checked. If the checksums of both messages are equal, only one message needs to be considered since the analysis states are also equal. It is important to note that the analysis state only depends on the already executed and the open operations. In other words, the comparison does not consider the attached data since performing the same operation twice can result in slightly different results, which makes a checksum or fuzzy checksum comparison unfeasible. The last field of the message header is the 16-bit *length* field that specifies the length (max. 65,535 bytes) of the analysis header.

The *analysis header* consists of the required *operations* and a *data directory*. The *operations* field represents a *directed acyclic graph* (termed *operation graph*) of various operations or operation chains that are required to be executed on the attached data. Since the operation graph is an essential component of our proposed template, we give a detailed description in the next section. The *data dictionary* maps the graph operations, which is within the message uniquely identifiable by the *oid* attribute, onto the data. This mapping includes the start and the end point of the data within the *payload*, the type and the checksum of the data.

The *payload* contains the raw data as well as the evaluated data. Different data are separated for individual access, unordered for high flexibility, independent compressed for reducing transmission overhead, and can have various data formats to cover a broad range of analysis use cases.

D. Operation Graph

Resource-intensive tasks can be completed either by a single operation or by a chain of dependent operations (e.g., victim detection). For this reason, we model the operations and their execution order as a *directed acyclic graph* [17]. The operation graph $G_O = (V_O, E_O)$ consists of the set of vertices V_O, where each vertex represents an operation, and the set of edges E_O, where each direct edge represents the dependent order between two vertices. The resulting graph is topologically sorted and, thus, it can be easily traversed to search for open operations that a device can execute. Table I lists the available *CRUD* operations, namely *create* (C), *read* (R), *update* (U), and *delete* (D), that a device can perform and, thus, modify the operation graph or the message. The update operations can



(a) create C (b) read R (c) update U (d) delete D

(e) merge U_M (f) operate U_O (g) deleg. U_D (h) split U_S

Fig. 3: Scheme representation of graph-based operations

be further characterized by four *MODS* operations, namely *merge* (U_M), *operate* (U_O), *delegate* (U_D), and *split* (U_S) (cf. Tab. II).

We now explain the defined graph-based operations illustrated in Figure 3. A device can *create* a new message with sensor data attached and publish it to the network. The raw sensor data are represented as root node in the operation graph (cf. Fig. 3a). Another device can then *read* or parse this message and its operation graph without modifications to access its content (cf. Fig. 3b). To modify the message, a device needs to perform an *update* operation (cf. Fig. 3c): *merge* consolidates the operation graphs from two messages with different executed operations and attaches the required data to the message (cf. Fig. 3e). While *split* divides one operation graph and the required attached data of a message into two separate messages (cf. Fig. 3h). Devices with knowledge about how to analyze the data can *delegate* operations to the network by adding required operations to the operation graph (cf. Fig. 3g, white circles). A device can process or *operate* on the data by performing a defined operation from the operation graph and attaching the result to the payload (cf. Fig. 3f, transition from white to black circles). Last but not least, a device can *delete* the message from the network by not forwarding it (cf. Fig. 3d). The described graph-based operations define an entire set of how devices can autonomously and piecewise process data in the network.

E. System Model for In-Network Processing

Here, we introduce the system model where our task-oriented message template is used as a substrate for cooperation among the devices. The envisioned model does not assume any specific requirements about the network or the devices. The devices cooperate with each other in a self-organizing manner. This is essential in ad-hoc intermittent networks such as given in disaster scenarios. To organize the in-network processing, we introduce four *SOFD* roles, namely *sensor* (r_s), *operator* (r_o), *forwarder* (r_f), and *delegator* (r_d) (cf. Tab. III).

We now explain the dynamic roles and its associated message abilities. A *sensor* node, i.e., this node takes the role r_s, has the ability of sensing which inherits the operations of creating a new message and attaching collected information or raw sensor data. *Forwarder* nodes - this is the basic role r_f of each node - have networking capabilities to receive

TABLE III: Definition of dynamic roles and the associated message abilities of participatory nodes (termed *SOFD*)

Abbr.	Role	Abilities	Description
r_s	**(S)**ensor	Sensing = {C}	Create messages with raw sensor data attached
r_o	**(O)**perator	Processing={U_O}	Execute operations on the message's data and update the result
r_f	**(F)**orwarder	Networking = {R}	Read and forward the message to its neighbors without modifications
r_d	**(D)**elegator	Controlling = {U_M,U_D,U_S,D}	Adjust the message, e.g., merge, delegate, split, delete

and forward messages to their neighbors. This includes the parsing and reading of the message without modifying it for effectively handling duplicated messages or messages with the same processing or revision state. During the propagation of messages, *operator* nodes are allowed to analyze the data by performing *operate* on the operation graph. For that, an operator node executes open operations from the operation graph which this node is able to execute. The result of the operations are attached to the message or added to the payload. The used (raw) data required for the executed operations are then removed when no other open operations require these data. Thus, the message's size is successively reduced - assuming that the size of higher-level information is smaller than the size of raw sensor data - to further reduce the network traffic. Since the network can be highly dynamic, a decentralized coordination is important. For that, we introduce the *delegator* role r_d. Nodes taking this role (e.g., rescuers in disaster scenarios) know which analysis are required to perform on these data and adjust the messages to delegate the processing into the network. That implies that delegator nodes control and manage the in-network processing by performing *merge*, *split*, *delegate*, or *delete* on the message's operation graph.

In our system model, each node can autonomously decide to dynamically take multiple roles based on its available capabilities and current utilization. Such decisions can be optimized using heuristic and local optimization algorithms [18]. For security and trust issues of some scenarios, roles can also be assigned in a static way to the participatory devices. With the defined roles, our model allows a self-organizing and distributed in-network processing in mobile ad-hoc mesh networks with loosely connected devices.

IV. Ontology for In-Network Processing

The cooperation of mobile devices like smartphones, autonomous mobile robots, and stationary servers results in a heterogeneous landscape for in-network processing: devices differ in their operating system, system architecture, programming language, representation of information, tasks and goals. New joining devices may also provide or use unknown operations and representations. To overcome cooperation issues of heterogeneous networks, we incorporate a hierarchical ontology (termed *ICE Ontology* [8]) in our message template to establish a common understanding and to enable interoperability. The ICE Ontology, which is an extended version of the *MUSIC Context Ontology* [19], provides a modeling vocabulary.

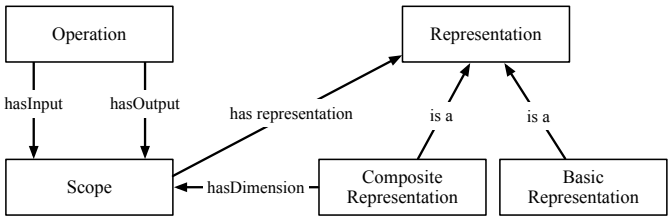


Fig. 4: Classes provided by the top-level *ICE Ontology*

Figure 4 gives an overview of the ontology classes. The structure of representations is modeled by the two classes *Scope* and *Representation*. Scopes describe properties of entities, such as position, color, or state. Representations define how the information are represented and are associated to scopes by the *hasRepresentation* object property. For example, a position can be represented as WGS84 coordinates or as an address composed of street, house number, and zip code. Representations are divided in *Basic-* and *Composite-Representations*. Basic representations cover single valued data types, such as integer, double, or string. Composite representations describe a representation composed of multiple dimensions. Each dimension is again a scope associated by the *hasDimension* object property. This results in a hierarchical definition of representations, where existing representations can be used as building blocks. *Operation* describes computational processing of information in defined representations. An operation can have multiple outputs with different scopes, which is modeled by the *hasOutput* object property. Each output has one fixed defined representation. In contrast, each input of an operation can have a set of possible representations, which is represented by the *hasInput* object property.

Using the described ontology, our task-oriented message template benefits from several aspects: each node in the heterogeneous network can derive required knowledge for processing the data from the ontology model, e.g., identifying similar representations or operations, adding additional definitions at runtime and connecting ontologies by so-called *bridge ontologies*. However, the overhead of ontologies is one of the major drawbacks. For that, only few nodes require to operate on the ontology itself. All other nodes (e.g., sensor, operator, or forwarder nodes) only need to use the *Internationalized Resource Identifiers* (IRI) of supported operations and representations. If most parts are known at the design time, which is the case in some application domains like service robotic apps, the ontology model can be replaced by a simpler taxonomy to further save network traffic, i.e., representations and operations can be identified by integer values instead of their IRIs.

V. Evaluation

In this section, we evaluate our proposed message template. We developed a *reference implementation* in Java, which is portable to several platforms. In the following, we first evaluate

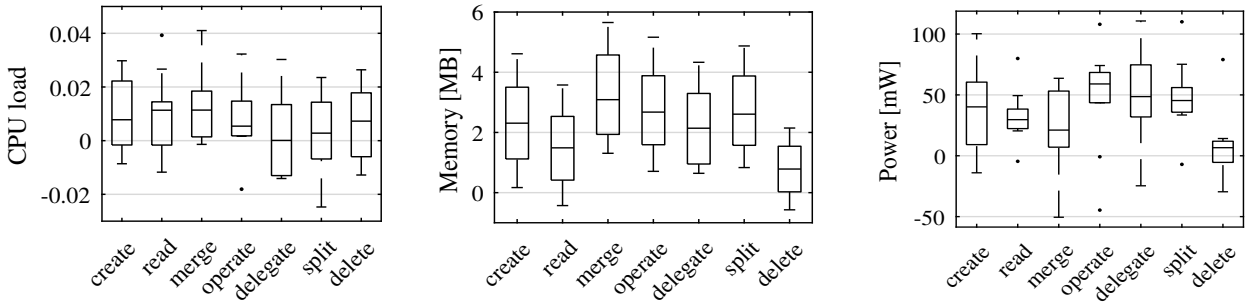


Fig. 5: Message overhead of graph operations for Nexus 5: (a) CPU usage, (b) memory usage, and (c) power consumption

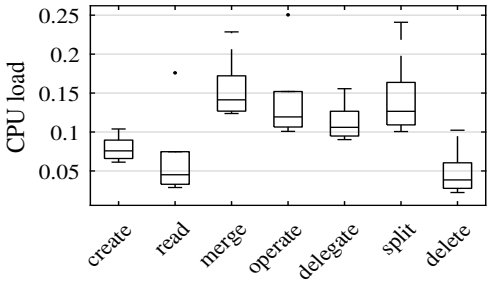


Fig. 6: Message overhead of graph operations for Raspberry PI3: CPU load

the resource usage on different devices and platforms for handling our task-oriented messages. Second, we assess the feasibility of our approach in an indoor testbed with wireless mesh nodes.

A. Message Overhead: Resource Usage

First, we measure the resource usage (CPU, memory, and energy) of particular graph operations to assess the overhead for processing our message template.

1) Experimental setup: We evaluate on two platforms: (a) *LG Nexus 5*, and (b) *Raspberry PI3*. The Nexus 5 smartphone is equipped with 2.26 GHz quad-core 64-bit ARMv8 processor (Qualcomm Snapdragon S800), and 2 GB memory. The operating system is updated to Android 7.0, namely Nougat. The Nexus 5 is equipped with a 2,300 mAh Lithium Polymer (LiPo) battery by default. The smartphone includes all electronics required for measuring the battery voltage and the current consumption of the device. The integrated MAX170485 fuel-gauge chip4 provides high-accuracy voltage measurements and battery level estimation. The resolution of 1.25 mV with an error of 7.5 mV enables to measure differences between graph operations. The Raspberry PI3 Model B is equipped with 1.2 GHz 64-bit quad-core ARMv8 processor, and 1 GB memory. The operating system is Raspbian 4.4.

Program profiling is an obvious approach for optimizing and comparing systems. Thus, an implemented lightweight runtime profiler[4] (i.e., an app running in the background) measures three metrics for our benchmarks: (i) CPU load, (ii) memory usage, and (iii) energy consumption on the mobile device. We choose a sampling interval of 300 ms for CPU and memory monitoring, and a sampling rate of 50 ms - a good, empirical determined balance between accuracy and CPU load - for energy measurements.

For measuring, we disabled all unnecessary background services and switched off the display of the mobile device. We first run a 30-second baseline measurement to get the default average resource usage of the operating system and our profiler tool. Then, we execute each graph operation multiple times on messages containing an operation graph with $|V| = 5$ and measure the above metrics for 10 seconds. We repeat this measurement run 10 times. Finally, the resulting values of these are averaged and cleaned by subtracting the baseline values to reduce measurement errors.

2) Results: Figure 5 shows the resource usage for the particular graph operations on the Nexus 5 smartphone. The CPU load of the quad-core processor, which reflects the CPU queue length and is a better performance metric than the utilization [20], is vanishingly low (< 0.04) for all graph operations (cf. Fig. 5a). Compared to the measured baseline values of the Android system, no visible differences can be seen. We repeated the same experiment on the PI3, which has a slower CPU and less memory than the Nexus 5 (Fig. 6). We see that the *update* operations (especially *merge* and *split*) need the most computational resources. All in all, the CPU load on the Raspberry PI3 is higher than on the Nexus 5 due to the hardware differences, but is still very low (< 0.25).

Contrary to the CPU load, the memory usage can be assigned exactly to an app. The usage of each operation is lower than 6 MB for the given setup (cf. Fig. 5b). We see that the *create* and *update* operations use more memory than the *read* or the *delete* operations. The energy consumption is also very low ($< 100mW$) considering that the most complex graph operation (*merge*) requires less than 500 ms (cf. Fig. 5c).

In comparison to the resource usage for sensing [21], communication [22], and processing [2], the resource usage for handling our proposed message template is nearly negligible.

B. Feasibility: Load tests

Second, we conduct load tests in an indoor testbed with wireless mesh nodes to assess the feasibility of our approach.

[4]https://github.com/Telecooperation/profiler-android (access.: 2016-12-01)

Fig. 7: (a) Venue floor plan and deployment map of the indoor testbed consisting of 8 wireless mesh nodes, (b) snapshot of the deployment, and (c) the wireless mesh node and its constituent components.

1) Experimental setup: The indoor testbed contains 8 wireless mesh nodes deployed at different locations on the same floor of our building. Figure 7a shows the venue plan and the deployment map. Each mesh node is installed on the wall or on a truss directly below the ceiling (cf. Fig. 7b), and consists of a Netgear GS108Tv2 router, a DeLock Splitter, an APU2 board, and a Raspberry PI3 Model B (cf. Fig. 7c). For our evaluation, we only use two components of the mesh node: the PI3 as computing unit and the router as networking device.

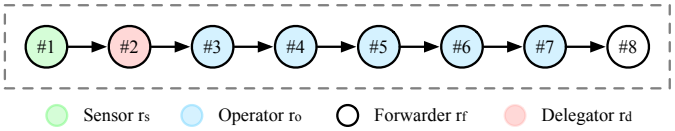
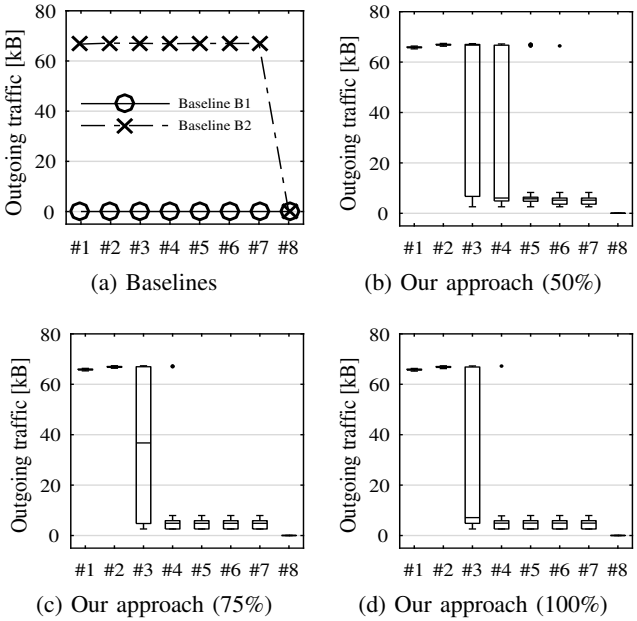


Fig. 8: Test chain with dynamic *operator* role assignments

Figure 8 shows the topology chosen for our experimental setup. We choose a chain, which is a special case of a mesh network, to avoid side effects, to reduce measurement errors, and to get reproducible results. In addition, we assign the *sensor* and *delegator* role statically to the first two nodes. The next five nodes can take the *operator* role depending on their utilizations. The last node collects and consolidates the results. All nodes take at least the *forwarder* role to forward messages. We further synchronize the clock of all nodes via NTP (Network Time Protocol) for measuring purpose. Considering the disaster scenario example, a rescuer (taking the delegator role) gets raw sensor data collected by victims' smartphones (taking the sensor role). While the rescuer has knowledge about the required information, he is not able to analyze all received data with his resource-limited mobile device. Therefore, he delegates the processing into the network. Civilians (taking the operator or forwarder role) can then process the data and forward the result back to the rescuer.

To show the feasibility of such a scenario, we evaluate our approach with load tests and compare the results against two baseline settings: (B1) the sensor node collects and evaluates the raw data; the node only forwards the results to the network (*local processing*). (B2) the sensor node collects and forwards the raw data to the network. Nodes interested in the analysis results have to process the raw data by their own (similar to a *centralized approach*). Our approach aims for finding



(a) Baselines (b) Our approach (50%)

(c) Our approach (75%) (d) Our approach (100%)

Fig. 9: Outgoing network traffic (= sum of message sizes) per node within our test chain using 20 simultaneous messages

a trade-off between these two by processing the data in the network. We conduct small load tests with 5, 10, 15, and 20 simultaneous messages. Based on simple heuristics (50%, 75%, 100% utilization), the operator nodes only act as forwarder. To compare the settings, we measure two metrics: (i) network traffic per node, and (ii) overall completion time. Each setting is repeated 5 times to reduce measurement errors.

2) Results: Figure 9 shows the results of the outgoing network traffic of 20 simultaneous messages for each node. As expected, the network traffic of baseline *B1* is rather low (lower bound) since the first node collects and evaluates the data by its own and only forwards the result (cf. Fig. 9a). In contrast, baseline *B2* has the highest network traffic (upper bound) since the raw data needs to be forwarded through the entire chain. In comparison to the baselines, our approach distributes the computational load among the network (cf. Fig. 9b-d). If the nodes use the 50% utilization heuristic, the first three operator nodes (#3, #4, #5) evaluate most of

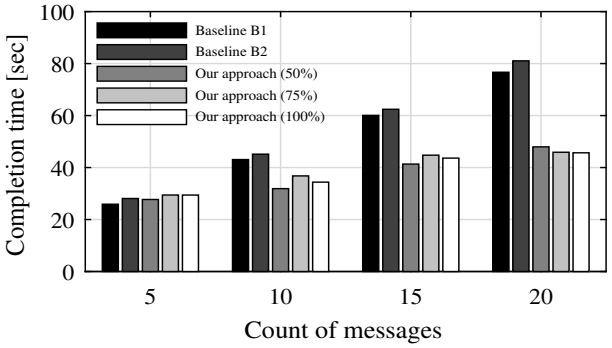


Fig. 10: Completion times of two baselines (B1, B2) and our approach with different utilization thresholds

the messages since the outgoing data is markedly decreased at node #5 (cf. Fig. 9b). Using 75% and 100% utilization heuristics, the first operator node process more messages, thus, the network traffic is decreased earlier (cf. Fig. 9c-d).

In addition to low network traffics, short completion times for analyzing the data are desirable. Figure 10 shows the completion time results for our test settings. In case of low load (i.e., 5 messages), we can see that the completion time is nearly the same for our approaches and the baselines. However, the performance difference gets visible for higher loads: our proposed approach - independent on the utilization heuristic - outperforms the baselines since it balances the work load for a particular node in a self-organizing way.

In conclusion, our self-organizing approach finds a better tradeoff between network traffic and completion time than the baselines by using in-network resources based on their load.

VI. Conclusion

In this paper, we presented an adaptive task-oriented message template (namely *ATMT*) that enables cooperative in-network processing for heterogenous mobile devices in a decentralized ad-hoc network. Utilizing our proposed template, no additional synchronization overhead (e.g., control information) is required since the message contains the operations that need to be executed and the required data. In the first evaluation, we showed that the message overhead and the additional resource usage to handle messages are negligible. We also assessed the feasibility of our approach in an indoor testbed with 8 nodes. Our results show a self-organizing in-network processing approach to reduce network traffic and utilize computational resources in mobile ad-hoc networks.

Acknowledgment

This work has been co-funded by the LOEWE initiative (Hessen, Germany) within the NICER project and by the German Federal Ministry of Education and Research (BMBF) Software Campus project "OppEPM" [01IS12054].

References

[1] J. Gubbi, R. Buyya, S. Marusic, and M. Palaniswami, "Internet of Things (IoT): A Vision, Architectural Elements, and Future Directions," *Future Generation Computer Systems*, vol. 29, no. 7, pp. 1645–1660, 2013.

[2] C. Meurisch, A. Seeliger, B. Schmidt, I. Schweizer, F. Kaup, and M. Mühlhäuser, "Upgrading Wireless Home Routers for Enabling Large-scale Deployment of Cloudlets," in *7th International Conference on Mobile Computing, Applications, and Services (MobiCASE'15)*. Springer, 2015, pp. 12–29.

[3] Q. Wang, B. Lee, N. Murray, and Y. Qiao, "CS-Man: Computation Service Management for IoT In-network Processing," in *27th Irish Signals and Systems Conference (ISSC'16)*. IEEE, 2016, pp. 1–6.

[4] L. Yang, S.-H. Yang, and L. Plotnick, "How the Internet of Things Technology Enhances Emergency Response Operations," *Technological Forecasting and Social Change*, vol. 80, no. 9, pp. 1854–1867, 2013.

[5] A. A. Abbasi and M. Younis, "A Survey on Clustering Algorithms for Wireless Sensor Networks," *Computer Communications*, vol. 30, pp. 2826–2841, 2007.

[6] Z. Lu, G. Cao, and T. La Porta, "Networking Smartphones for Disaster Recovery," in *14th International Conference on Pervasive Computing and Communications (PerCom'16)*. IEEE, 2016, pp. 1–9.

[7] S. Cho and C. Julien, "Chitchat: Navigating Tradeoffs in Device-to-device Context Sharing," in *14th International Conference on Pervasive Computing and Communications (PerCom'16)*. IEEE, 2016, pp. 1–10.

[8] S. Niemczyk and K. Geihs, "Adaptive Run-Time Models for Groups of Autonomous Robots," in *10th International Symposium on Software Engineering for Adaptive and Self-Managing Systems (SEAMS'15)*. IEEE, 2015, pp. 127–133.

[9] E. Fasolo, M. Rossi, J. Widmer, and M. Zorzi, "In-network Aggregation Techniques for Wireless Sensor Networks: A Survey," *IEEE Transactions on Wireless Communications*, vol. 14, pp. 70–87, 2007.

[10] D. Alessandrelli, M. Petraccay, and P. Pagano, "T-RES: Enabling Reconfigurable In-network Processing in IoT-based WSNs," in *9th International Conference on Distributed Computing in Sensor Systems (DCOSS'13)*. IEEE, 2013, pp. 337–344.

[11] A. N. Mian, R. Baldoni, and R. Beraldi, "A Survey of Service Discovery Protocols in Multihop Mobile Ad-hoc Networks," *IEEE Pervasive Computing*, vol. 8, no. 1, pp. 66–74, 2009.

[12] R. Hermann, D. Husemann, M. Moser, M. Nidd, C. Rohner, and A. Schade, "DEAPspace–Transient Ad-hoc Networking of Pervasive Devices," *Computer Networks*, vol. 35, no. 4, pp. 411–428, 2001.

[13] F. Sailhan and V. Issarny, "Scalable Service Discovery for MANET," in *3rd International Conference on Pervasive Computing and Communications (PerCom'05)*. IEEE, 2005, pp. 235–244.

[14] C. Groba and S. Clarke, "Opportunistic Service Composition in Dynamic Ad-hoc Environments," *IEEE Transactions on Services Computing*, vol. 7, pp. 642–653, 2014.

[15] L. Thomas, J. Wilson, G.-C. Roman, and C. Gill, "Achieving Coordination Through Dynamic Construction of Open Workflows," in *10th International Middleware Conference (Middleware'09)*. Springer, 2009, pp. 268–287.

[16] C. Xu, S. Li, G. Liu, Y. Zhang, E. Miluzzo, Y.-F. Chen, J. Li, and B. Firner, "Crowd++: unsupervised Speaker Count with Smartphones," in *15th International Joint Conference on Pervasive and Ubiquitous Computing (UbiComp'13)*, 2013, pp. 43–52.

[17] L. Wang, "Directed Acyclic Graph," in *Encyclopedia of Systems Biology*. Springer, 2013, pp. 574–574.

[18] R. V. Kulkarni and G. K. Venayagamoorthy, "Particle Swarm Optimization in Wireless Sensor Networks: A Brief Survey," *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, vol. 41, no. 2, pp. 262–267, 2011.

[19] R. Reichle, M. Wagner, M. U. Khan, K. Geihs, J. Lorenzo, M. Valla, C. Fra, N. Paspallis, and G. A. Papadopoulos, "A Comprehensive Context Modeling Framework for Pervasive Computing Systems," in *8th International Conference on Distributed Applications and Interoperable Systems (DAIS'08)*. Springer, 2008, pp. 281–295.

[20] D. Ferrari and S. Zhou, "An Empirical Investigation of Load Indices for Load Balancing Applications," DTIC Document, Tech. Rep., 1987.

[21] I. Schweizer, R. Bärtl, B. Schmidt, F. Kaup, and M. Mühlhäuser, "Kraken.me Mobile: The Energy Footprint of Mobile Tracking," in *6th International Conference on Mobile Computing, Applications and Services (MobiCASE'14)*. IEEE, 2014, pp. 82–89.

[22] C. Gross, F. Kaup, D. Stingl, B. Richerzhagen, D. Hausheer, and R. Steinmetz, "EnerSim: An energy consumption model for large-scale overlay simulators," in *38th Annual IEEE Conference on Local Computer Networks (LCN'13)*, 2013, pp. 252–255.