Complex Services Offloading in Opportunistic Networks

The An Binh Nguyen*, Marius Rettberg-Päplow*, Christian Meurisch†, Tobias Meuser*,
Björn Richerzhagen*, Ralf Steinmetz*
*Multimedia Communiations Lab (KOM), TU Darmstadt, Germany
Email: {firstname.lastname}@kom.tu-darmstadt.de, mberg@hotmail.de
†Telecooperation (TK), TU Darmstadt, Germany
Email: meurisch@tk.tu-darmstadt.de

***Abstract*—Situation awareness is important to plan relief work in emergency response. However, impaired communication and computation infrastructure makes it difficult to acquire and analyze information. Accordingly, complex and resource-intensive information processing can be offloaded through opportunistic ad hoc contact to, e.g., first responder mobile devices, leveraging their idle resources. Ensuring complete service execution without overloading individual devices is a challenging task in such dynamic networks. In this work, we propose handover mechanisms that utilize the current context of individual mobile devices to balance load and achieve complete task execution without requiring a global view on the opportunistic network. We study their scalability and performance by combining them with our unified message template for distributed service processing in the OMNeT++ simulation environment. The evaluation shows that our handover mechanisms increase the success rate significantly and achieve distributed load balancing.**

I. Introduction

In recent years, mobile services become more computation-intensive and more complex, requiring multiple processing stages and highly-specialized hardware. Executing such services on a single device is therefore impractical. To alleviate a device with limited computing capacity, a complex mobile service can be offloaded as a task to a remote cloud for execution. However, offloading to the cloud is not always possible due to overloaded or impaired infrastructures, which can occur, for instance, in emergency situations such as disaster scenarios. Opportunistic offloading [1] has been introduced as an emerging solution for offloading computation. Hereby, the computation tasks can be offloaded to a nearby stationary computing unit such as cloudlet [2], or to an opportunistic network formed by mobile devices [3]. While both approaches share the common idea of leveraging nearby available computing resources, offloading in an opportunistic network provides more flexibility and more advantages in favor of executing complex tasks with multiple processing stages. A complex task can be divided into several subtasks, and distributed to the participating mobile devices, leveraging their idle, and heterogeneous capabilities. Besides ensuring successful execution of the offloaded tasks, balancing services execution among the participating devices is also essential, and beneficial. On the one hand, load balancing relieves overloaded devices, effectively leading to improved overall performance. On the other hand, the energy consumption for executing the offloaded tasks by participating devices can be decreased through load balancing, resulting in (i) longer lifetime of opportunistic networks, which serves as communication medium during critical situations, and (ii) more acceptance of users to contribute their resources. Most approaches dealing with load balancing in mobile systems are based on global knowledge of the network to formulate the load balanced assignment as an optimization problem. In line with the dynamic nature of opportunistic networks, the optimization problem can also be solved in a distributed manner, as proposed in [4]. Still, rapidly changing environments require a flexible and adaptive approach to load balancing that takes changing conditions, resource constraints, heterogeneity of tasks and services, and mobility into account.

In this work, we propose several handover mechanisms for load balancing in complex services offloading based on the currently available context of single devices. To this end, we extend our previous work [5] on a task message template that allows the user to define a task and the services required to accomplish the defined task. Our message template bundles the control information and the corresponding payload data into a single message. This enables mobile devices to decide autonomously whether and how to participate in service processing. We implement our proposed mechanisms within the OMNeT++ simulator [6], allowing for an in-depth evaluation of their performance in terms of load balancing and success rate and their cost in terms of message overhead and latency.

In summary, the contributions of this paper are threefold:

- We propose several load balancing (*LB*) mechanisms for distributing complex tasks across devices in an opportunistic network, optimizing resource utilization and success rate, while minimizing the communication overhead.
- We develop a simulation environment based on OMNeT++, which integrates our earlier work on an adaptive task oriented message template [5].
- We conduct an extensive evaluation of our LB mechanisms within the OMNeT++ simulation environment. We show the overall performance gain, improved fairness, and inherited trade-offs of our proposed LB mechanisms.

ISBN 978-3-903176-08-9

The remainder of this paper is organized as follows. *First*, we discuss related work. *Second*, we give a brief introduction in our adaptive task message template (namely *ATMT*) for distributed in-network processing, and highlight an important open research challenge, namely, *distributed fair load balancing*. *Third*, we present our load balancing handover mechanisms and an in-depth evaluation relying on OMNeT++ simulations, before concluding the paper.

II. Related Work

Offloading computational workload in mobile systems, aiming to reduce network traffic have been studied in several research work. [7] propose a decentralized optimization model for the underlying operator placement problem. This approach, however, does not consider dynamic changes of the environment. Recent research on *Complex Event Processing (CEP)* in the context of vehicular networks has put more attention to adaptive mechanisms; an example is CEP operator migration [8]. Another research direction is edge computing, in which computation tasks are offloaded to nearby computing resources such as cloudet-upgraded router for processing [2]. In the aforementioned work, balancing computational workload is neglected.

Load balancing or fair resource allocation have always been an important research aspect in mobile networks. To analyze fair resource allocation, Fossati et al. [9] propose to extend the Jain's fairness index with a satisfaction factor of users. The problem of resource allocation is modeled through game theory, using their proposed metric. Tham et al. [4] target mobile edge networks and formulate a constrained optimization problem to achieve load balancing. The problem, however, has to be solved by a central entity. Fernando et al. [10] incorporate work stealing concepts in mobile crowd computing, allowing a worker device to take over workload from other devices. The authors focus on the practical implementation using mobile devices and, thus, do not consider work stealing in a large scale setup. Centralized coordination is impractical in an opportunistic network. Consequently, Benchi et al. [11] study the consensus problem in opportunistic networks, which allows each node to make a consent decision upon receiving enough votes from others. Comparable to our work is load balancing for services composition in opportunistic networks. Viswanathan et al. [12] use a time deadline for services composition to formulate an optimization problem, which can be solved by service providers in a distributed manner. The complexity of such optimization formulation is high, thus cannot cope well with the rapid changes of an opportunistic network. In [13], Sadid et al. introduce a hop by hop composition model designed for opportunistic networks, considering load and mobility of the devices. The authors propose to let each service provider decide on the next composition to cope with dynamic changes. Our work differentiates from [13] in that we explicitly incorporate uncertainty factors in our local optimization mechanisms to increase their robustness. Furthermore, we provide a thorough evaluation focusing on the quality of load balancing.

III. Scenario: In-Network Data Analysis in Emergency Situations

A. Scenario Description

To plan relief operations in emergency response situations efficiently, the relief workers need to have situational information. The required raw sensing data can be obtained through built-in sensors on the mobile devices as shown in [14]. Thereafter, these data have to be processed and analyzed to extract valuable information. A concrete example can be found in [15]. In this work, image processing techniques are applied to extract faces of victims through pictures shot by smart phones. To capture the situational overview, a large amount of data might be required. Processing all these data in a single device of the relief worker is inefficient. Two options are possible: (i) offloading the data analysis to cloud servers, (ii) offloading the data analysis to several surrogate devices for distributed processing. The first option is not always possible in case of impaired communication infrastructure, which often occurs in disaster situations. The second option provides a more flexible solution to analyze data, leveraging idle resources available in opportunistic network.

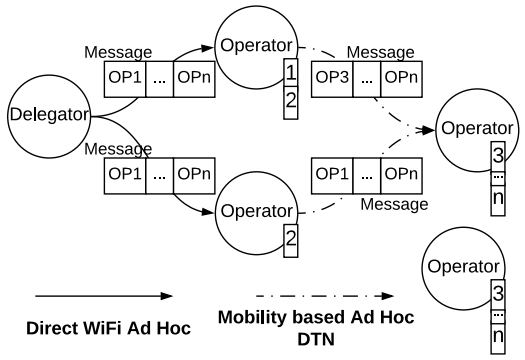


Fig. 1: Abstract system model of disseminating ATMT task messages in opportunistic networks for processing.

To facilitate distributed processing through mobile devices in the elaborated scenario, the *adaptive task-oriented message template* (ATMT) is proposed in [5]. The objective of this message template is to allow users to define an analysis goal and the operations/services required to accomplish this goal. Using the task message template, the data analysis is handed over from one device to the next device, wheres each device can perform one or several operations. Hence, the task is divided and processed in a distributed manner. The workflow of processing an ATMT message is illustrated in Figure 1. In this illustration, a device (called *delegator*) with required domain knowledge of how to process the data analyis, defines n operations ($op_1..op_n$) and disseminate the message into the opportunistic ad hoc networks. Each device participating in the processing (called *operator*) executes the operations provided by this device and hands over the processed message upon opportunistic contact with other devices for further executions.

B. Adaptive Task Message Template

The construction of the ATMT task message template designed in our previous work [5] is illustrated in Figure 2. To facilitate distributed processing in opportunistic networks, one of the objectives of ATMT is to allow the participating devices to cooperate without having to rely on any centralized coordination. Due to this reason, an ATMT message contains both meta-information required for processing and the belonging payload data. The meta-information is stored in the ATMT header, consisting of two parts, i.e., *message header* and *analysis header*. The first part is the fix-sized *message header*, which contains an UUID for identification, a checksum on the status of the processing and the length of the header. By comparing the checksum in the *message header*, a device can check on the current status of the processing and decides to merge, drop or to handover a task, without parsing the whole message content. The *analysis header* composes of an *operations graph* and a *data dictionary*. The *operations graph* is based on an acyclic directed graph, that is used to model the processing goal, the required operations/services, and the processing order. The *data dictionary* in the header maps the operations in the *operations graph* to the respective data pieces in the *ATMT payload*. When an operation is completed by a device, this device can replace the old payload data with the processed result. All in all, the construction of an ATMT message allows each device to make autonomous decision.

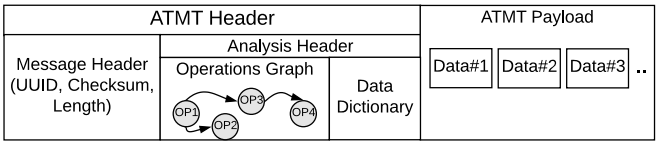


Fig. 2: Construction of ATMT task message template as designed in [5].

A system utilizing ATMT message to perform in-network data analysis as the aforementioned scenario depends on the heterogeneous capabilities of the devices, which can be translated into different roles. Four roles are conceived, i.e., *sensors* for obtaining raw data, *delegators* with the domain knowledge for constructing the *operations graph* which can be understood as a way to coordinate the devices in a distributed manner, *operator* for performing operations/services, and *forwarder* to handover the ATMT messages. As briefly described in the previous section, Figure 1 shows a sample workflow using ATMT concept. *Sensor* devices are omitted in the illustration. A *delegator* device receiving data from the sensors constructs an ATMT message, and hands over this message to its directly connected operators via WiFi ad hoc communication. Each operator processes the ATMT message and executes the operations/services required in the *operations graph* according to its available resource and services. The resulting ATMT messages can be forwarded through *store, carry and forward* concept of opportunistic mobile networks to another operator at later time for further processing. In doing so, the chances for successful execution of a complex analysis task can be increased.

IV. Challenges and Assumptions

Based on the description of the ATMT construction and the in-network data analysis workflow, we can identify several challenges. (i) A centralized coordination with the complete view over the services available in all mobile devices does not exist. Consequently, each device only has a partial view of the network. (ii) The devices considered in this work are highly dynamic and mobile. This requires adaptive mechanisms. (iii) Due to the challenges elaborated in (i) and (ii), the handover of ATMT messages in an uncoordinated way might lead to massive communication overhead and processing redundancies, i.e., workload waste. Optimizing both successful execution of complex ATMT tasks and load balancing under the aforementioned challenges is thus our main target.

With respect to the challenges and the elaborated application scenario, the following assumptions are made:

- Decentralized opportunistic ad hoc network: we focus on complex services offloading and distributed processing in an opportunistic ad hoc network. Thus, we assume that the devices are mobile and they are able to communicate if they are in WiFi range of each other.
- Heterogeneous resource and services: we assume that the participating devices possess different capabilities, i.e., each device has different resource capacity left, can provide different services, perform different operations.
- Cooperative behaviour: we assume that no participating device has malicious intention. To establish a trustworthy distributed processing environment in a mobile system, trust measurement concept such as in [16] can be utilized.
- Location-aware: we assume that each device is able to determine its own location.

V. Handover Mechanisms

We design our handover mechanisms with special focus on load balancing. Our target is to improve the distribution of workload among participating devices in an opportunistic network, taking into account the challenges and assumptions as previously discussed. According to Alakeel [17], we have to consider three main aspects when designing load balancing mechanisms for distributed systems, i.e., *transfer strategy, location strategy, information strategy*. *Transfer strategy* is the decision whether to offload/handover the task, *location strategy* indicates which destinations should the tasks be offloaded to, and *information strategy* refers to the context information which can be used to devise *transfer strategy* and *location strategy*. Accordingly, the *information strategy* is the most important component of handover mechanisms. W.r.t. our scenario, the *information strategy* is limited, since a global view of all devices in an opportunistic network is not possible. Therefore, a device in an opportunistic network can only use either (i) its own context information or (ii) a partial view of the network through information shared by other devices via opportunistic contact. Based on this observation,

we devise three categories for handover mechanisms, i.e., *naive*, *work stealing*, and *local optimization*. A device using *naive* mechanism only requires its own resources utilization as context to make handover decision; while a device using *work stealing* and *local optimization* requires shared context from other devices. The details of each devised mechanisms will be elaborated in the following.

A. Naive

Naive mechanism does not require any sophisticated shared context; the decision is made by single device's context with respect to the resource utilization on this device. To this end, each participating device in our system maintains a queue of ATMT tasks. The size of ATMT tasks queue indicates the total resource, which a device can contribute. Two options are possible for naive handover. (i) Since an ATMT message represents a complex task that requires the execution of several services in a predefined order, the successful completion of a task is not guaranteed in opportunistic network. Consequently, to increase the success rate, a naive node simply contributes all of its resource available in ATMT tasks queue and passes the processed ATMT tasks to all neighbours. This behavior resembles the well-known *epidemic* routing [18]. Hence, the common observed characteristics of *epidemic* routing can also be applied for our naive mechanism; i.e., the success rate is improved by scarifying communication and computation overhead. Due to this reason, a naive mechanism utilizing full resources of participating devices, serves well as the baseline for benchmarking purpose. (ii) It can also be observed that, in dense opportunistic networks, a high number of devices providing similar services can exist. On the one hand, the resource on these devices will be used redundantly, following a greedy naive behavior. On the other hand, the success rate when reducing the size of ATMT tasks queue and rejecting ATMT tasks upon reaching a limit, can be compensated by the high number of participating devices with similar capabilities. In such cases, reducing the size of the ATMT tasks queue and rejecting tasks can decrease the number of redundantly executed operations, while preserving the high success rate and leading to improved load balancing. This intuition will be analyzed later in the evaluation (cf. Section VI). In summary, a naive device in our system will either fully utilize all its available resource, i.e., epidemic flooding of ATMT tasks in the whole network, or a device can intentionally reduce its tasks queue and drop upcoming received tasks.

B. Work Stealing

The term *work stealing* is coined in the context of parallel computing [19]; it refers to the act of an underutilized processor *stealing* threads from over-utilized processor, aiming to relieve over-utilized processors from high workload, thus a better load balancing among processors can be achieved. Fernando et al. [10] incorporates the concept of *work stealing* in the context of *mobile crowd computing*. Our devised *work stealing* strategy extends this idea for a more decentralized dynamic system, i.e., mobile devices in opportunistic network with the ability to act autonomously.

In our system, each operator device is qualified as a work stealer, i.e., if an operator device deems itself to be under-utilized, this device can ask to take over ATMT tasks from the nearby devices. Underutilization is determined based on the current number of ATMT tasks in the tasks queue. If this number is less than a work stealing limit, then an operator device will ask the surrounding operators to handover ATMT tasks. An operator device triggers the work stealing process by sending a work stealing message, indicating the number of ATMT tasks (n_{ws}) that this work stealing operator is willing to accept and the list of its providing operations. In order not to exhaust the maximum resource of the work stealing operator, n_{ws} should not exceed the maximum size of the ATMT tasks queue on the device. Furthermore, to avoid egoistic behavior of the participating operators, when receiving the work stealing message with the indicated capacity n_{ws}, a device is allowed to handover maximum up to n_{ws} tasks, however a minimum number of task n_{keep} should always be kept back in the tasks queue. To decide how many tasks should be handed over to the work stealing operator, three options are conceived: (i) Devices receiving work stealing message try to exploit the maximum capacity indicating by the work stealing operator without any coordination from the work stealing device. (ii) The work stealing device assumes the local coordination and divides the number of allowed ATMT tasks equally for its neighbors. (iii) The work stealing device accepts tasks from its neighbors following *first come first serve* principle. As soon as the maximum threshold is reached, the work stealing device will notify the neighboring devices to stop handing over tasks.

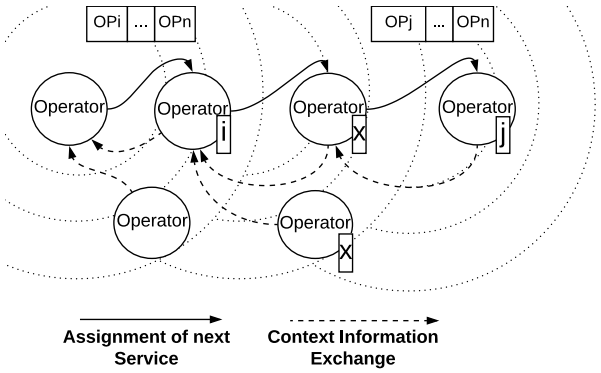


Fig. 3: Illustration of local optimization concept, choosing to the best next handover destination benefiting load balancing.

C. Local Optimization

Local optimization is inspired by the observation of Eager et al. [20], that a simple load adaptation locally in a distributed environment can lead to the overall improved performance of the whole system. Additionally, in the context of services composition in opportunistic network, Sadid et al. [13] show that the overall performance of opportunistic hop by hop composition is comparable to the performance of composition

orchestrated by a centralized entity. Following this line of thought, we devise strategies for tasks handover decision at an operator device in our system, requiring only local knowledge obtained through shared context of the neighboring devices. Our target is to optimize the load sharing among several devices locally by handing over the tasks to the next best destination within a close proximity. The local optimization is done by single devices autonomously, but still in a collaborative manner through shared context. The overall workflow of *local optimization* strategy is illustrated in Figure 3.

In opportunistic networks, the context of devices can be shared either in a reactive or proactive manner. Reactive context sharing is triggered only if a device receives an explicit query asking for its context. However, in a highly dynamic environment, a long time might elapse since the query is sent, until the information comes back to the query initiator. Due to this reason, proactive context sharing seems to be more favorable in opportunistic network. The context information is thus exchanged at any opportunistic contact of two devices in our system. Two devices exchange the summary of the context information about themselves and about the other devices that these two have seen in the past. Through this way, every devices have a snapshot of the shared context information. The context information required for local optimization of load balancing are generated by each device as a list of available operations ($op_i..op_j$), the currently-used capacity (n_u), the current position ($(long, lat)$), moving direction ($\vec{v}$) and a time stamp (t_{info}) when generating context information. When an operator device triggers the local optimization, it checks the current shared context and filters the nodes within a proximity of distance d_{max}, that possess the required operations, as potential destinations for task handover. The potential destinations can be further filtered, omitting devices that have distance around d_{max} and currently move farther away from the initiating device. To choose destinations benefiting the load balancing, we use a cost function covering three aspects for local optimized assignments, which are the currently-used capacity in the tasks queue (n_u), the distance and the uncertainty of the shared context information about operator O, i.e., ($\mu(N_O)$). The cost function is defined as follows:

$$c(N_A, N_O, \#_{OP}) = (w_l * c_l * \#_{OP} + w_d * c_d) * \mu(N_O) \quad (1)$$

in which:

$$\begin{aligned} \mu(N_O) &= 1 + \frac{t_{current} - t_{info}}{t_{keepAlive}} \\ c_l(N_O) &= \frac{n_{max} - n_u}{n_{max}} \\ c_d(N_A, N_O) &= \frac{d(N_A, N_O)}{d_{max}} \end{aligned} \quad (2)$$

In Equation 1, N_A is the node that wants to trigger the handover, to assign some of its tasks to other operator; N_O is a potential destination operator, to which the tasks can be assigned. $\#_{OP}$ is the number of operations that will be handed over. w_l and w_d are weighting factors for cost values of load (c_l) and distance (c_d), respectively. In Equation 2, the uncertainty factor $\mu(N_O)$ is captured using the time elapsed since the context information of operator O are generated until recently. The main cause of the uncertainty is the high dynamic of the network, caused by mobility or by disappearance upon exhaustive utilization of the devices. Consequently, outdated context information, which results in a higher uncertainty factor $\mu(N_O)$, can lead to a negative handover decision, increasing the total cost. The cost for load component in the equation is considered based on the number of currently utilized tasks in the tasks queue and the maximum size of the task queue (n_{max}). The distance component is determined by the ratio between the current distance $d(N_A, N_O)$ from the assigner to the operator and the search radius (d_{max}), as in Equation 2. This is based on the intuition, that the communication overhead for a nearer node is less than that for the farther node; since more hops might be required to reach an operator at larger distance.

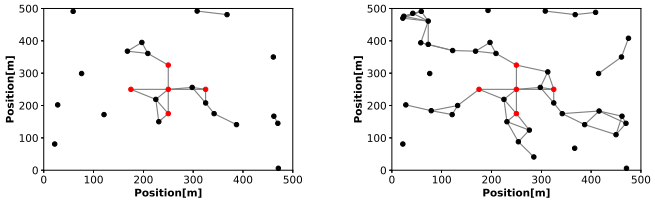
In order to improve load balancing, each device can trigger the local optimization to find the best destination with minimum handover cost for the upcoming operations of an ATMT task. We propose two modes to trigger local optimization to find the best next handover destination, i.e., (i) proactive mode: every time the shared context information are updated, indicating possible better destination for the next handover or (ii) reactive mode: only when a device receives more tasks than the current size of its task queue, indicating over-utilization. Regardless of trigger modes, to ensure effective dissemination of shared context information, every time a device detects a new neighbor, this device exchanges its summarized context information with the new neighbor.

VI. Evaluation

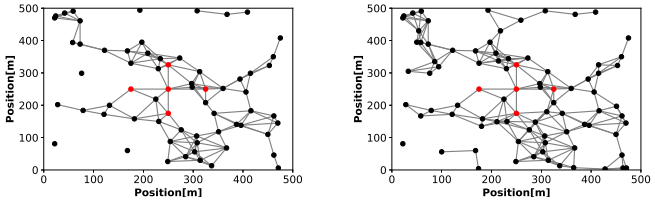
We implement and evaluate the task handover mechanisms as detailed in Section V, using a customized OMNeT++ module compatible with our designed ATMT message [5]. In this section, we first elaborate on the evaluation methodology, the simulation setup, and the evaluation metrics. Next, we study each handover mechanisms independently w.r.t. the evaluation metrics to identify the best performing option within each category. Last, we compare the proposed handover mechanisms against each other and point out the trade-off between the performance and load-balancing metric.

A. Scenario Modelling, Setup and Evaluation Metrics

Since the main target of our evaluation is the analysis of computation balancing, we model a simulation scenario to enable the dissemination of ATMT tasks into an opportunistic network. This network consists of several mobile nodes that move around a $500 \times 500m^2$ simulation area. Two nodes can communicate within $75m$ WiFi range. We abstract from a WiFi ad hoc model to enhance the scalability of the simulation and assume that the congestion will be handled by Link Layer mechanisms [21]. We set up five static nodes, one main delegator and four helper delegators which are connected to the main delegator. The main delegator generates ATMT-tasks and

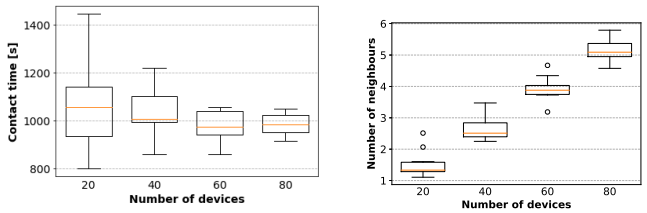


(a) 20 devices (b) 40 devices



(c) 60 devices (d) 80 devices

Fig. 4: Contacts among nodes for varied number of devices.



(a) Contact duration (b) Number of neighbors

Fig. 5: Average contact duration and number of neighboring devices according to the used Levy Walk mobility model.

injects these tasks to the network through the helper delegators. The reason for this particular setup is to allow the initial ATMT tasks to reach more operator nodes even under sparse network as in case of 20 devices (cf. Fig. 4a), aiming solely at generating a similar start configuration in both dense and sparse setups. The performance of the handover mechanisms, which rely on the behavior of the participating nodes during the simulation run, is not affected by this setup. We create two types of task; a simple task which contains between two or three operations, and a complex task which always contains five operations. The delegator nodes are marked in red as shown in Fig. 4. To control the movement of the simulated mobile nodes, we use the Levy Walk mobility model. This decision is based on the fact, that the Levy Walk mobility model is reported in [22] to resemble the human mobility patterns. We generate mobility traces accordingly using BonnMotion [23]. The direct contacts among mobile nodes from the generated traces are illustrated in Fig. 4. Fig. 5 shows the observed characteristics of the generated traces, which suggest a longer, more stable contact duration and an increasing number of direct neighbors with more devices in the network. As such, 20 nodes represent a sparse opportunistic network, while 80 nodes represent a dense opportunistic network. The most important simulation parameters are summarized in Table I.

TABLE I: Simulation Setup

Simulated Area Size	$500 \times 500\, m^2$
Simulation Time	one hour
Number of Nodes	20, 40, 60, 80
WiFi Transmission Range	$75\ m$
Mobility Model	`LevyWalkMobilityModel`
#ATMT-Tasks	100, 1000
Naïve Greedy	full, limited
Work Stealing	full, FCFS, equalized
Local Optimization	proactive, reactive

We repeated each simulation ten times and plotted all obtained results with 90% confidence intervals. The following evaluation metrics were used to analyze the results:

(a) *Success rate* denotes the ratio between the number of successfully completed ATMT tasks that can be delivered back to the main delegator and the total number of tasks.

(b) *Communication overhead* is defined as the total number of ATMT messages that are generated and duplicated by the handover strategies.

(c) *Completion time* is the time elapsed since the main delegator injects tasks into the network, until all processed results come back to the main delegator.

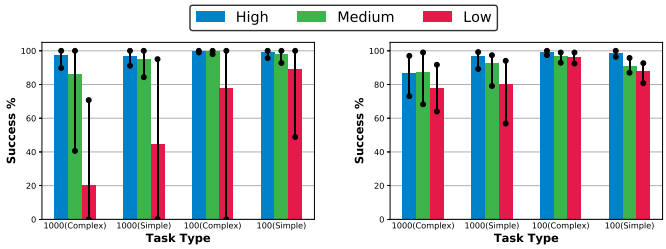
(d) *Jain index* is proposed by Jain et al. in [24] as follows: $JI(x_1, x_2, ..., x_n) = \frac{(\sum_{i=1}^{n} x_i)^2}{n * \sum_{i=1}^{n} x_i^2}$, wheres x_i denotes the resource consumed (in our scenario the number of operations executed) by node i. Jain index with value closer to 1 indicates higher fairness among the resources consumed by all nodes. Thus, Jain index is able to quantify the quality of load balancing mechanisms.

(e) *Redundancy factor* is defined as the ratio between the number of redundantly executed operations and the original number of operations in the network.

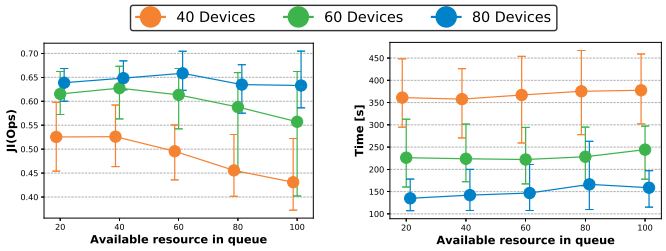
B. Handover Mechanisms Analysis

Naive: The evaluation for naive mechanisms has two objectives: (i) assessment of ATMT tasks dissemination in scarce and dense opportunistic networks and (ii) identification of suitable tasks queue's size which benefits load balancing quality as a baseline for further analysis.

In our simulation, each node possesses a number of predefined services which this node can execute. For evaluation of naive mechanisms, we set up three different classes characterizing the availability of the services on all nodes, i.e., *high, medium, low*. The distribution of the services availability on the nodes in each class follows a normal distribution. The *high* class assigns 50% of the nodes with all 5 available services required for the operations defined in the ATMT task; the *medium* class assign 50% of the nodes with between 2 and 3 available services; and the *low* class assign 50% of the nodes with no services, the majority of the rest are assigned only 1 single service. Fig. 6a and 6b show the dependency of success rate on the availability of the services. Low services availability decreases the success rate, which is visible in case complex tasks are executed in sparse network with only 20

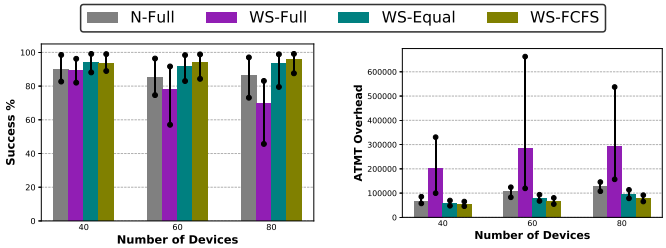


(a) Success rate with 20 nodes (b) Success rate with 80 nodes

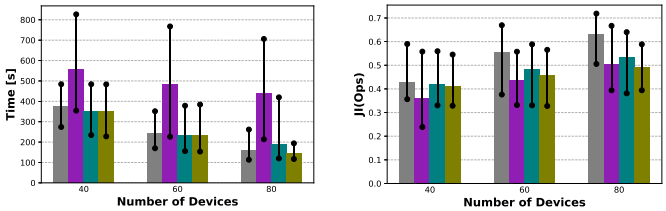


(c) Jain index (d) Completion time

Fig. 6: Analysis of naive handover.

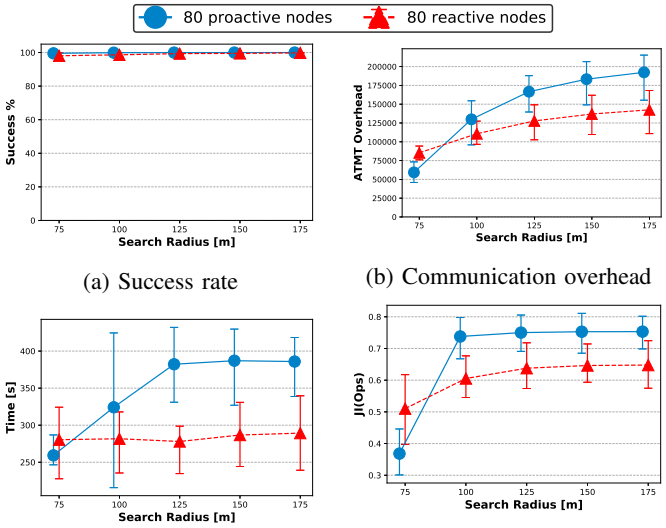


(a) Success rate (b) Communication overhead



(c) Completion time (d) Jain index

Fig. 7: Analysis of work stealing mechanisms.



(a) Success rate (b) Communication overhead

(c) Completion time (d) Jain index over executed operations

Fig. 8: Analysis of different modes for local optimization.

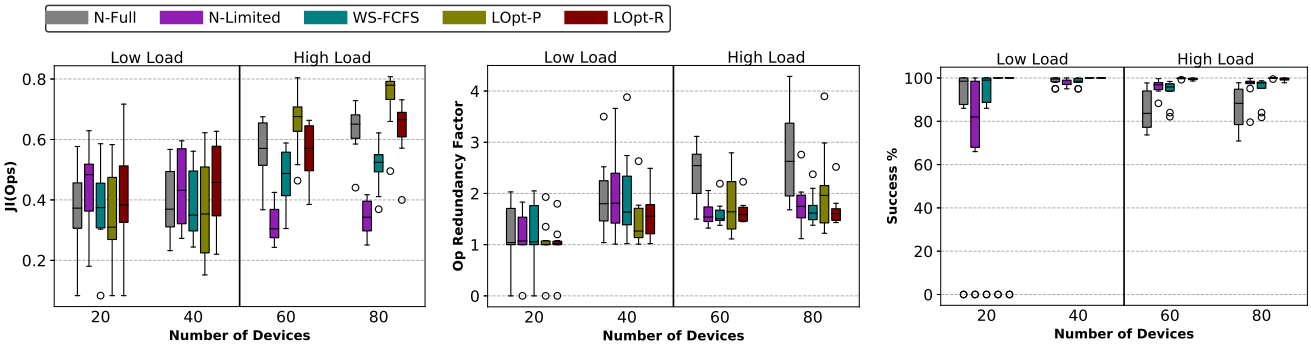
nodes. The success rate in this case only reaches around 20% (cf. Fig. 6a). However, the success rate despite low services availability can be compensated through higher number of devices as we anticipated. Fig 6b shows, that the success rate for complex tasks with low services availability can be improved from 20% (with 20 nodes) to 80% (with 80 nodes).

The effect of tasks queue's size on the performance and the quality of load balancing was examined. Fig. 6c shows slightly better values for Jain index over the executed operations when decreasing task queue's size, compared to the maximum size (100 in our simulation), indicating slightly improved fairness in the system. Shorter queue size also means less resource has to be contributed by the nodes. With respect to the completion time, a shorter tasks queue does not have any negative effect. Rather, the completion time depends on the number of devices, i.e., faster completion time can be achieved with more devices in the network as shown in Fig. 6d. Overall, the analysis of naive handover mechanisms suggests reducing the size of the tasks queue, thus frees resources for participating nodes.

Work Stealing: We compare thee options for work stealing as introduced in Section V-B against naive flooding handover mechanisms (*N-Full*). The 3 options for work stealing are respectively: *WS-Full* which tries to exploit the full capacity of the work stealing node, *WF-Equal* in which the work stealing node divides the accepted capacity equally among neighbors, and *WS-FCFS* which follows first come first serve principle.

It can be observed that greedy behavior when handing over tasks in *WS-Full* decreases the success rate (down to 70% with 80 nodes), while generating even more communication overhead compared to the naive handover *N-Full*. This negative effect is due to the redundant task handovers triggered by the neighbors in *WS-Full*, which the work stealing nodes have to drop at overloaded capacity. On the contrary, the two other work stealing options, *WS-Equal* and *WS-FCFS* slightly improve the success rate and even the completion time in some cases compared to *N-Full* (cf. Fig. 7a, 7c). The reason is, work stealing with *WS-Equal* and *WS-FCFS* can free some resources of the nodes locally; in contrast, naive handover mechanism generates more redundant operations (cf. overall comparisons, Fig. 9b). However, depending on the distribution of the nodes in the area, the chance for a work stealing node and an overloaded node to meet cannot always be guaranteed. Correspondingly, Jain index values obtained through *work stealing* display no major load balancing improvement using *work stealing* concept (cf. Fig. 7d).

Local Optimization: Since the local optimization looks for the best next destination within a search radius to assign the handover, we anticipate the size of this search radius affects



(a) Jain index over executed operations. (b) Redundancy factor. (c) Success rate.

Fig. 9: Comparison of handover mechanisms.N-Full denotes the flooding based naive handover; N-Limited denotes the naive handover with limited task queue; WS-FCFS represents work stealing, using first come first serve; LOpt-P denotes the *proactive local optimization*; LOpt-R denotes the *reactive local optimization.*

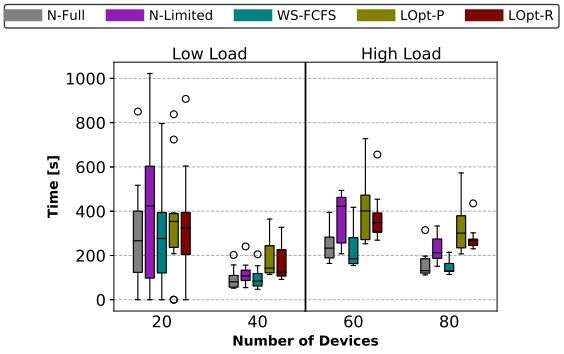


Fig. 10: Completion time of handover mechanisms.

all evaluation metrics. Hence, we vary the size of the search radius and analyze the corresponding influences. The results are shown in Fig.8. We evaluate two modes of the local optimization as introduced in Section V-C, i.e., proactive mode which triggers the local optimized handover upon receiving new shared context, and reactive which triggers the local optimized handover only for overloaded situations.

The success rate for both proactive and reactive modes are high (almost always at 100%) regardless of the size of the search radius. Obviously, larger search radius leads to more communication overhead. *Proactive local optimization* generates more communication overhead compared to *reactive local optimization*; since the context in an opportunistic mobile network tends to change rapidly, leading to more frequent information exchange in proactive mode (cf. Fig. 8b). A longer completion time for *proactive mode* is visible when increasing the size of the search radius, which is the trade-off for obtaining better result for optimization. In contrast, the completion time for *reactive mode* is quite stable, since it only triggers the local optimization at circumstances (cf. Fig. 8c). Fig. 8d shows improved Jain index values with larger search radius. With *proactive mode*, the Jain index value increases from 0.37 with 75 m search radius, up to 0.75 with 125 m search radius. *Reactive mode* increases the Jain index value from 0.5 at 75m, up to 0.65 at 125 m. Increasing the search radius more than 125 m shows no more fairness improvement, suggesting converge quality for load balancing. Hence, the search radius should be restricted in order not to waste communication overhead. Between two modes, *proactive local optimization* yields better quality for load balancing than *reactive mode* at larger search radius. This can be explained by the fact, that *proactive mode* reacts on the context changes of the network, while *reactive mode* waits for an overloaded situation.

C. Handover Mechanisms Comparison

Having analyzed the handover mechanisms individually in Section VI-B, we now compare all mechanisms against each other. To cover the performance indicators for both sparse and dense network situations, we use two setups: (i) a *low load* setup with 100 tasks distributed to 20 or 40 nodes and (ii) a *high load* setup with 1000 tasks distributed to 60 or 80 nodes. Selected results for the comparison regarding the Jain index, redundancy factor, success rate and completion time are presented accordingly in Fig. 9a, 9b, 9c, 10. For a sparse network, the quality for load balancing fluctuates, regardless of handover mechanisms. It is to be expected, since a sparse opportunistic network tends to be partitioned; many nodes are therefore isolated the whole time, providing no way for their resources to be exploited. Evidently, the quality for load balancing can be improved with more nodes in the network. Fig. 9a shows that our proposed *proactive local optimization* can achieve the best Jain index value (around 0.8 in case of 80 nodes), outperforms other handover mechanisms. The quality of load balancing obtained by *reactive local optimization*, despite being less than *proactive local optimization*, is still comparable to flooding based naive handover (both achieve Jain index values at around 0.65 with 80 nodes). *Proactive local optimization* yields the lowest redundancy factor (at avg. 1.5), compared to a very high redundancy factor of N-Full (at avg. 2.5, the worst case up to more than 4) (cf. Fig. 9b).

This confirms that the resources in naive mechanisms are used redundantly, while our proposed local optimization mechanisms help to alleviate this problem. As already discussed in the analysis of *work stealing*, *work stealing* cannot improve the overall load balancing, but can achieve higher success rate compared to naive mechanisms. The result shown in Fig. 9c again confirms this observation. The same result also demonstrates that our proposed local optimization mechanisms not only outperform other mechanisms w.r.t. load balancing, but are also able to outperform others w.r.t. success rate. Moreover, the marginal variances shown in the box plot obtained from the results of both local optimization modes, prove the robustness of the mechanisms, against the rapid changes in dynamic, mobile networks. The improvements achieved by local optimization mechanisms, however, have to take into account longer completion time (cf. Fig. 10).

VII. Conclusion and Future Work

In this paper, we extended the adaptive task-oriented message template (ATMT) defined in our previous work [5] and proposed several handover mechanisms that enable load balancing for distributed processing of complex tasks. Our proposed mechanisms were designed focused mainly on opportunistic networks, thus do not require any centralized coordination. The evaluation results show that we were able to achieve better load balancing through local optimization, leveraging only locally shared context information. Overall, our proposed task message template facilitates distributed coordination and is thus suitable for decentralized, highly dynamic environment.

Several directions are possible as our future work. First, the load balancing mechanisms proposed in this work can be further evaluated using real hardwares, which allows us to determine over-utilized situation in realistic conditions, e.g., based on CPU load or energy consumption level. This will also allow us to incorporate, and consequently study the effect of heterogeneity in terms of hardware configuration, energy consumption when executing a complex operation on distributed load balancing. Second, the handover mechanisms, especially work-stealing can be further augmented by prioritizing tasks, i.e., setting higher handover priority for nearly completed tasks can benefit the success rate, while setting higher priority for computation-intensive tasks will work in favor of load balancing. Third, within the context of information centric ad hoc network (ICN), it is shown that situational data can be collected by mobile devices [14]. Hereby, we want to combine the design of ATMT with data transport phase in ICN to deliver processed high-valuable information to the query initiator.

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] D. Xu, Y. Li, X. Chen, J. Li, P. Hui, S. Chen, and J. Crowcroft, "A Survey of Opportunistic Offloading," *IEEE Communications Surveys & Tutorials*, 2018.

[2] C. Meurisch, J. Gedeon, T. A. B. Nguyen, F. Kaup, and M. Mühlhäuser, "Decision Support for Computational Offloading by Probing Unknown Services," in *IEEE ICCCN*, 2017.

[3] E. Borgia, R. Bruno, M. Conti, D. Mascitti, and A. Passarella, "Mobile edge clouds for Information-Centric IoT services," in *IEEE ISCC*, 2016.

[4] C.-K. Tham and R. Chattopadhyay, "A Load balancing Scheme for Sensing and Analytics on a Mobile Edge Computing Network," in *IEEE WoWMoM*, 2017.

[5] T. A. B. Nguyen, C. Meurisch, S. Niemczyk, D. Böhnstedt, K. Geihs, M. Mühlhäuser, and R. Steinmetz, "Adaptive Task-Oriented Message Template for In-Network Processing," in *IEEE NetSys*, 2017.

[6] A. Varga and R. Hornig, "An Overview of the OMNeT++ Simulation Environment," in *SIMUtools*, 2008.

[7] B. J. Bonfils and P. Bonnet, "Adaptive and Decentralized Operator Placement for In-network Query Processing," *Telecommunication Systems*, vol. 26, no. 2-4, pp. 389–409, 2004.

[8] B. Ottenwälder, B. Koldehofe, K. Rothermel, K. Hong, D. Lillethun, and U. Ramachandran, "Mcep: A mobility-aware Complex Event Processing System," *ACM Transactions on Internet Technology*, vol. 14, no. 1, p. 6, 2014.

[9] F. Fossati, S. Moretti, and S. Secci, "A Mood Value for Fair Resource Allocations," in *IFIP Networking*, 2017.

[10] N. Fernando, S. W. Loke, and W. Rahayu, "Mobile Crowd Computing with Work Stealing," in *IEEE NBiS*, 2012.

[11] A. Benchi, P. Launay, and F. Guidec, "Solving Consensus in Opportunistic Networks," in *ACM ICDCN*, 2015.

[12] H. Viswanathan, E. K. Lee, I. Rodero, and D. Pompili, "Uncertainty-aware Autonomic Resource Provisioning for Mobile Cloud Computing," *IEEE transactions on parallel and distributed systems*, vol. 26, no. 8, pp. 2363–2372, 2015.

[13] U. Sadiq, M. Kumar, A. Passarella, and M. Conti, "Service Composition in Opportunistic Networks: A Load and Mobility aware Solution," *IEEE Transactions on Computers*, vol. 64, 2015.

[14] T. A. B. Nguyen, P. Agnihotri, C. Meurisch, M. Luthra, R. Dwarakanath, J. Blendin, D. Bohnstedt, M. Zink, R. Steinmetz *et al.*, "Efficient Crowd Sensing Task Distribution Through Context-aware NDN-based Geocast," in *IEEE LCN*, 2017.

[15] P. Lampe, L. Baumgärtner, R. Steinmetz, and B. Freisleben, "Smartface: Efficient face detection on smartphones for wireless on-demand emergency networks," in *IEEE ICT*, 2017.

[16] R. Dwarakanath, B. Koldehofe, Y. Bharadwaj, T. A. B. Nguyen, D. Eyers, and R. Steinmetz, "TrustCEP: Adopting a Trust-Based Approach for Distributed Complex Event Processing," in *IEEE MDM*, 2017.

[17] A. M. Alakeel, "A Guide to Dynamic Load Balancing in Distributed Computer Systems," *International Journal of Computer Science and Information Security*, vol. 10, no. 6, pp. 153–160, 2010.

[18] A. Vahdat and D. Becker, "Epidemic Routing for partially connected Ad Hoc Networks," Duke University, Tech. Rep., 2000.

[19] R. D. Blumofe and C. E. Leiserson, "Scheduling Multithreaded Computations by Work Stealing," *Journal of the ACM (JACM)*, vol. 46, no. 5, pp. 720–748, 1999.

[20] D. L. Eager, E. D. Lazowska, and J. Zahorjan, "Adaptive Load Sharing in homogeneous Distributed Systems," *IEEE transactions on software engineering*, no. 5, pp. 662–675, 1986.

[21] C. Lochert, B. Scheuermann, and M. Mauve, "A Survey on Congestion Control for Mobile Ad Hoc Networks," *Wireless communications and mobile computing*, vol. 7, no. 5, pp. 655–676, 2007.

[22] I. Rhee, M. Shin, S. Hong, K. Lee, S. J. Kim, and S. Chong, "On the Levy-Walk Nature of Human Mobility," *IEEE/ACM transactions on networking (TON)*, vol. 19, no. 3, pp. 630–643, 2011.

[23] N. Aschenbruck, R. Ernst, E. Gerhards-Padilla, and M. Schwamborn, "BonnMotion: A Mobility Scenario Generation and Analysis Tool," in *SIMUTools*, 2010.

[24] P. N. D. Bukh and R. Jain, "The Art of Computer Systems Performance Analysis, Techniques for experimental Design, Measurement, Simulation and Modeling," 1992.