Distributed Cluster-Topology Maintenance for Mobile Collaborative Applications

Jan Gäbler and Hartmut König
Computer Network and Communication Systems Group
Brandenburg University of Technology (BTU), Germany
{jan.gaebler, hartmut.koenig}@b-tu.de

***Abstract*—Nowadays, collaborative applications play an increasing role in mobile communications in order to enable cooperation among mobile and/or stationary participants – often using the peer-to-peer (P2P) group communication paradigm. The probability of communication failures increases in mobile environments. As a consequence, frequent leaving and joining of partners can be observed resulting in unstable group topologies. A widely used approach is to divide the group into peer clusters that are fully meshed among each other. In this paper, we propose an application-independent approach for a periodic maintenance of distributed cluster-based group topologies for mobile collaborative applications. The proposed approach also dynamically adapts the overlay structure to changing network conditions. The maintenance strategy can be parametrized by several metrics. It outperforms simple overlay maintenance strategies, such as round-robin, by respecting the resource and computation limitations of mobile devices. The run-time is increased by 10% in the best case compared to LEACH, and by $3-5\%$ on average. We describe the approach and evaluate its performance w.r.t. different metrics.**

***Index Terms*—mobile collaborative applications; clustering algorithm; closed group topology maintenance; P2P cluster-topology optimization; application independent optimization;**

I. Motivation

Collaborative applications play an increasing role in mobile communications, e.g., for audio/video conferences, collaborative writing, and mobile gaming. A new kind of collaborative applications that emerged from the traditional stationary ones – called *mobile collaborative applications* [1] – use *group communication systems* to support the collaboration among mobile as well as stationary participants (peers). The participants can be located in a single network (covering small local areas) or in different networks interconnected via the internet or other backbone networks. A close collaboration between the participants is a key property of collaborative applications. The number of participants typically varies between only a few and up to a hundred. The collaboration takes place in a decentralized closed group, whereby each participant has a comprehensive knowledge of all the others. Effects like churns, missing network coverage, resource limitations of the mobile devices, however, significantly increase the probability of communication failures. As a consequence, the participants frequently leave and join the group, thus leading to unstable group topologies.

There are a number of group communication systems [2] that offer message transfer services based on *virtual synchrony* [3] to ensure a global knowledge among the group members. These services operate without any kind of infrastructure. They are only using peer-to-peer (P2P) techniques [4] to distribute messages within the group. A widely used approach in this context is to divide the group into clusters of peers that are fully meshed, such as in TRANSIS [5], SpovNet [6], or Moversight [7]. Each cluster is managed by a certain peer – the *master* – that is responsible for all the other peers in the cluster which are hence called *slaves*. A good cluster topology is characterized by a reasonable number of peers per cluster, a homogeneous distribution of the peers (w.r.t. peer resources and mobility properties), and well-chosen masters.

In order to increase the stability of the cluster topology and to optimize its structure, a topology maintenance service should be provided by the underlying group communication system. This can be done either in a centralized or in a distributed manner. For mobile collaborative applications, the distributed variant should be preferred to avoid the single point of failure issue and to ensure an equitable use of the peer resources. Note that such a maintenance service is also required in other cluster-based topologies. Such can be found, for instance, in mobile ad hoc networks (MANETs) or in sensor networks.

The design and implementation of a maintenance service is challenging in different ways. (1) The topology has to be optimized periodically based on the resources available to each peer. (2) This optimization has to be performed by each group member literally at the "*same time*". (3) Although performed locally, each peer must preserve the global group knowledge w.r.t. the global state of the supported application.

In this paper, we propose a generic distributed approach of a maintenance service for cluster-based network topologies that is, besides mobile collaborative applications, also applicable to other areas, as mentioned above. It aims at a periodic and dynamic adaptation of the cluster structure to changing network conditions. The maintenance strategy can be parameterized by certain metrics. The remainder of the paper is structured as follows: The subsequent section discusses existing approaches for optimizing cluster-based topologies. In Section III we describe the applied system model followed by a presentation of the proposed maintenance approach in Section IV. Next, in Section V, we describe the integration of the service into the *Moversight* [8] protocol. Section VI deals with some synchronization issues that have to be solved in

DOI 10.1109/LCN.2016.55

order to preserve group consistency. Thereafter, in Section VII, we evaluate the performance of our approach using various metrics. We conclude the paper with some final remarks.

II. Related Work

The communication topology is a crucial property of a distributed system. Its optimization represents a classical design issue. Therefore, we will focus on protocols that manage the placement of peers in a distributed topology – called clustering protocols. They can be classified into (1) centralized, (2) distributed, and (3) hybrid ones. Various distributed clustering protocols have been proposed (see [9] and [10] for comprehensive overviews). Most of these approaches relate to MANETs and sensor networks rather than to mobile collaborative applications.

SONDe [11] represents an example of a class of LAN and WAN protocols, which create clusters merely based on the number of logical hops between the master and the slaves using a constant lower boundary h for the maximum number of hops to the master. Each peer observes its h-hop neighborhood and decides to become a master or a slave. Thus, the composition of the clusters is changing dynamically depending on the nodes neighborhood. The resulting topology is unstable and not well balanced. Since this may result in single points of failure and resource exhaustion of the mobile peers, these protocols are not suitable for mobile collaborative applications.

Protocols like NONSTOP [12] and others [13]–[15] rely on peer properties such as node mobility. NONSTOP addresses foremost aspects of network partitioning and user mobility. Nodes with comparable properties are clustered together. Peer properties are distributed in the neighborhood by piggybacking. Based on partition prediction techniques, the peers estimate the place and time at which the next partition will occur. After that the protocols select a new master in advance that can bridge the predicted partition. The number of peers per cluster and their load vary over time. Therefore, these protocols merely reach a master availability rate of 80% which destroys the group synchronicity. Furthermore, the created topologies are unbalanced and cost more energy compared to an optimized topology.

Sensor network protocols, such as LEACH [16] and [17], use stochastic techniques to build a cluster topology. During each round, LEACH selects P master nodes, where P is a predefined value. To reduce the communication-related energy consumption the distributed selection algorithm uses the current available energy resources per peer as a criterion. Slaves are assigned to the cluster with the closest master and time slots are used for the communication between masters and slaves. Stochastic approaches lower the complexity of the selection algorithm, but the slave assignment to clusters tends to be unstable and leads to highly dynamic topologies that do not guarantee a group consistency as required for mobile collaborative applications.

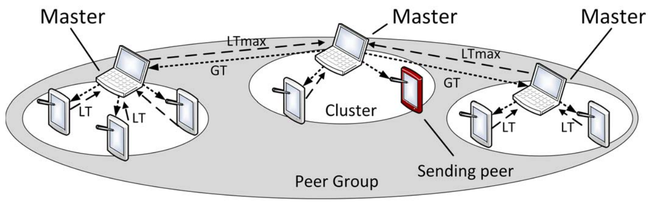


Fig. 1. Message delivery scheme in cluster-based topologies

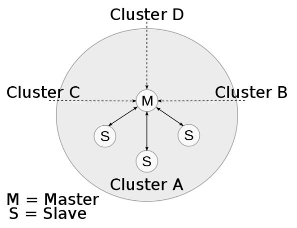
III. System Model

Mobile collaborative applications usually consist of closed groups in which participants enter by invitation. In order to ensure group consistency and to synchronize the application context, these groups often apply the *virtual synchrony* paradigm [18]. Multicast is used for exchanging messages and the groups must handle time constraints, responsiveness, connectivity loss, and mobility.

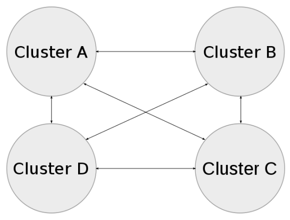
Virtual synchrony models the group composition as a sequence of views ordered by a strict $<$ relation. A *view* comprises the set of peers that currently form the group and each view is referenced by an ID. All peers locally store the global knowledge of the group. This knowledge consists of the group composition including the state, transport address, role and resources of each peer as well as the state of the application, which is derived from the application messages exchanged within the group.

The group communication system ensures updates and constant accuracy of the global knowledge. If accuracy is not ensured, the group disintegrates into partitions and the distribution of messages will fail. Every change in the group composition results in a new view. There are different properties used in view-aware groups [2] of which the most important one is that all messages have to be delivered in a totally ordered manner. A group member receives these messages if and only if it is connected to the group and runs error-free.

Closed P2P groups often use a cluster-based overlay topology (also called P2P-super-peer formation) below the group abstraction layer. The message delivery follows the *abcast* approach of Isis [3] (see Figure 1). A peer first sends a message to its master that distributes it within the local cluster and forwards it to the masters of the other clusters that in turn disseminate it within their cluster. To ensure a totally ordered delivery a *virtual logical time* is applied [19], which – to put it in simple terms – models an event counter for each peer. This is used to bring the messages exchanged in the group in a total order. If a slave receives the message, it acknowledges the reception to its respective master by sending the logical *local reception time* (LT). All masters collect the LTs of their clusters and send the maximum LT to the master of the sending slave, which in turn calculates the logical *global reception time* (GT) of the message. This GT is eventually broadcasted back into the group and used as a reference value to order the messages.



(a) Intra-cluster Level



(b) Inter-cluster Level

Fig. 2. Cluster levels

IV. A Distributed Topology Maintenance Approach

In order to guarantee a stable and optimized cluster topology, a maintenance service is required that is able to handle peer failures and cluster instabilities as well as optimize the topology [17]. A *failure handling* should immediately be triggered, if either a single peer fails or whole parts of the network are no longer reachable. In general, these situations are detected by network failure detectors and usually induce a partial or complete reset of the overlay. The network failure detector is a distributed monitoring service running on each peer within the group. The goal of the *optimization maintenance* is to continuously improve the structure of the overlay, e.g., by using view changes for an adaptive optimization. Clustered topologies can be divided into an intra-cluster and an inter-cluster level [10]. The intra-cluster level (see Figure 2a) only considers the structure of a single cluster, i.e., the contained peers, their roles, and the internal topology of the cluster. The inter-cluster level (see Figure 2b), in contrast, characterizes a cluster by its properties, e.g., the number of peers, the available resources per peer, and other specific relations among the clusters.

The optimization of a clustered overlay is generally a NP-hard problem [20]. Combinatorial approaches are based on the bell-number [21] and often result in a variety of possible solutions. Therefore, it is preferable to optimize the topology regarding dedicated parameters instead of pursuing an optimal topology. A decisive impact on the overlay performance has the number of clusters [14]. If the number of clusters is too small, each master has to maintain a large number of slave connections. Too many clusters, in contrast, impair the inter-cluster communication in fully-meshed cluster overlays and increase the number of connections that have to be monitored by the masters failure detector. The number of connections that a master has to handle affects its resources and the overall performance of the overlay. In addition, overlays consisting only of one or two clusters tend to behave like a star topology, i.e., the master becomes a single point of failure.

A. Optimizing the Number of Clusters

In order to optimize the number of clusters we first analyze the connections of the cluster overlay. The optimal number of clusters C depends on the optimal number of connections in the cluster overlay E for a group of N peers.

The total number of connections in the cluster overlay E is determined by Formula 1. The referenced number of connections among the masters E_m is given by Formula 2, where M depicts the number of masters. Formula 3 defines the number of connections to the slaves.

$$E = E_m + E_s \tag{1}$$

$$E_m = 1/2\ (M^2 - M) \tag{2}$$

$$E_s = N - M \tag{3}$$

To determine the optimal number of clusters C we have to represent E as a function of C, called $e(C)$, as shown in Formula 4.

$$C = M => e(C) = 1/2\ (C^2 - C) + N - C \tag{4}$$

The determination of the first and second derivation of $e(C)$ results in the optimal number of clusters. The result is independent of N with 1.5 clusters. Figure 3 shows the graph of the function $e(C)$ for groups of 4, 9, 16, and 25 peers.

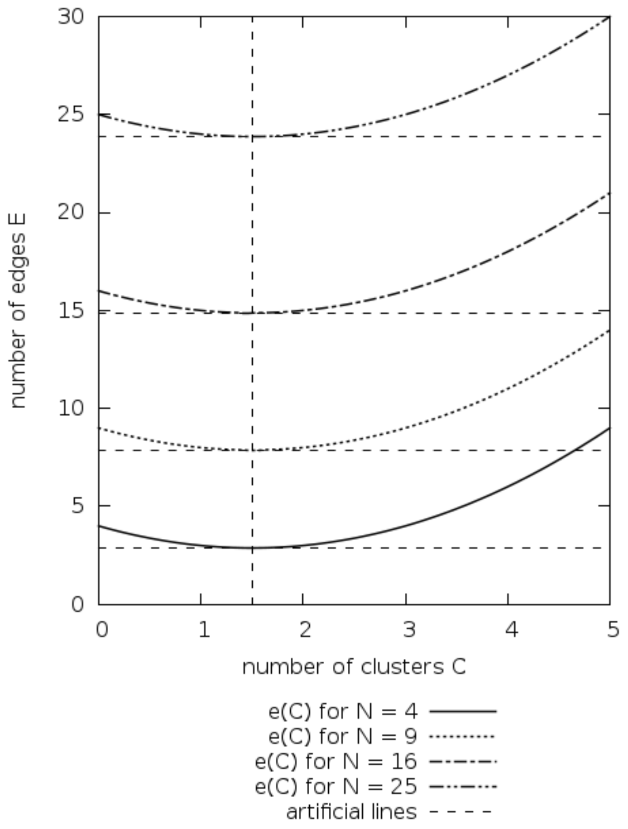


Fig. 3. Number of edges per cluster

As argued above, an overlay with one or two clusters acts like a star topology. In this case, it is not possible to optimize the topology solely based on the number of clusters. Since a master has the largest number of connections to manage, we have to find a trade-off between the desired number of clusters C and the number of edges to be managed by a master.

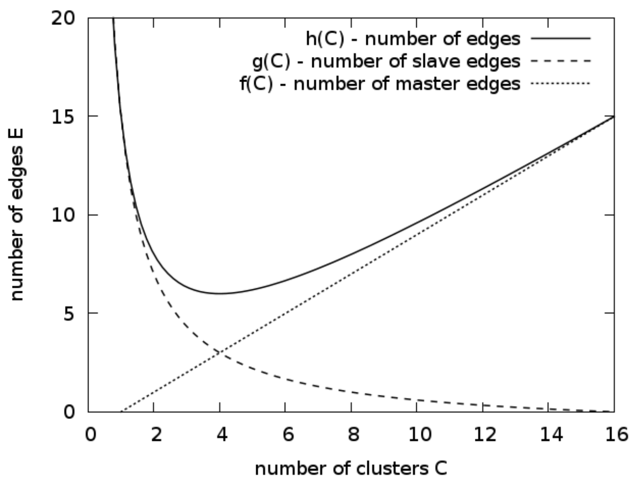


Fig. 4. Master edges for different numbers of clusters in a group of 16 peers.

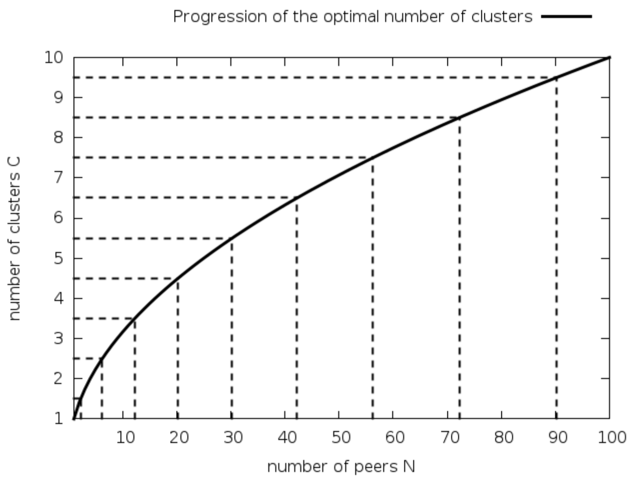


Fig. 5. Example curse of the cluster count for a group of 100.

The number of edges of the master belonging to cluster x is referred to as E_{mx}. It depends on the number of edges to the other masters and the number of slaves in cluster x – referred as S_{cx}. Formula 5 shows this relationship.

$$E_{mx} = \alpha \cdot (M-1) + \beta \cdot S_{cx} \tag{5}$$

Figure 4 depicts the curve of the function E_{mx} for a group of 16 peers. The left branch of the parabola can be approximated by $g(C) = \frac{N}{C} - 1$, and the right one by $f(C) = C-1$. In addition, $g(C)$ describes the average number of slaves per cluster and $f(C)$ the number of edges to other masters.

In Formula 5 the expression $M-1$ can be replaced by $f(C)$ and S_{cx} by $g(C)$. This leads to Formula 6.

$$e_d(C) = \alpha \cdot (C-1) + \beta \cdot (\frac{N}{C} - 1) \tag{6}$$

If we assume that the less connections a master has to manage, the better it is for energy consumption, then the minimum of Formula 6 results in Formula 7. For actual use, the value of C has to be rounded appropriately.

$$C = \pm\sqrt{\beta/\alpha \cdot N} \tag{7}$$

$$C = \sqrt{N}, \text{ for } C \in \mathbb{N} \tag{8}$$

Since there is no difference between an edge connecting two masters and an edge between a master and a slave, we apply $\alpha = \beta$. The resulting optimal number of clusters C is given in Formula 8.

Letting the maximum number of peers in a group being limited to 100 peers, then the maximum number of clusters would be $C_{max} = \sqrt{(}100) = 10$. Figure 5 shows the trend of the number of clusters in a group up to 100 peers. It indicates the optimal number of clusters for a given number of peers.

B. Optimizing the Cluster Heterogeneity

After determining the optimal number of clusters in the overlay, we have to reduce the cluster heterogeneity regarding the cluster properties, e.g., the number of slaves or their resource values. The heterogeneity indicates the deviation of the current situation from a defined reference value [9].

The *resource value* represents the state of a peer regarding a selected set of resources. There are various resources that can be considered for this. The selected subset is mapped onto a normalized value. A simple example is the normalized battery charge of a mobile device. Analogously, it is possible to use the available bandwidth, signal strength, latency to a well-known Internet host, and the devices class (e.g., smart phone, laptop, sensor node) for the calculation. The *reference value* is usually defined as an average or optimal resource value. It is calculated by the system or defined by the application developer. Due to the fluctuating nature of mobile communication, the property values, such as latency, data rate, and signal strength, have to be smoothed over the time before they are used in the resource value calculation. The mapping of various resource properties onto a single value is discussed in [15].

According to the observed deviation, a cluster is marked either as *overloaded*, *loaded*, or *unloaded*. Based on the deviation, the distribution of the selected property for the overlay can be calculated. The result can then be used, for instance, to select the optimal cluster for a joining peer.

The outlined approach is generic and can be used to define a three step iterative optimization algorithm, as shown in Algorithm 1. The algorithm first determines the number of optimization rounds. Then the topology is optimized in forward direction by moving peers from overloaded to unloaded clusters – a step which is called *forward balancing*. If the used optimization property is specific to each peer, a third step – the *backward balancing* – is required to relieve possible overpopulated clusters.

We explain the principle with an example. To balance the

Algorithm 1 balance()

```
1: $max\_rounds \Leftarrow N - C$
2: $balancingForward(max\_rounds)$
3: if property is peer specific then
4:     $balancingBackward(max\_rounds)$
5: end if
```

peer load of each cluster based on the cluster size property, the reference load has to be determined using the load caused by the number of peers per cluster. In addition, we have to define an allowed deviation for this value. To balance the peer load, the load of each cluster has to be determined. The forward balancing is executed as shown in Algorithm 2.

Algorithm 2 forwardBalancing()

```
1: for $i = 0$ to $max\_rounds$ do
2:     if exists overloaded cluster then
3:         $srcC \Leftarrow getMostOverLoadedCluster()$
4:         $dstC \Leftarrow getLeastLoadedCluster()$
5:         if $isBalancingPossible(srcC, dstC)$ is true then
6:             $moving\_peer \Leftarrow srcC.getPeer()$
7:             $memRegister.movePeer(moving\_peer, dstC)$
8:             $calculateClusterLoad()$
9:             $determineReferenceValue()$
10:        end if
11:    end if
12: end for
```

In each round, the most loaded peer from the most overloaded cluster (line 3) is transferred to the least loaded one (line 4). In our example, we randomly select a peer in the source cluster (line 6). If the used optimization property is specific to the peer, we have to choose the most loaded peer w.r.t. the selected property. For example, the current stage of the battery charge is a peer-specific property. In the next step the least loaded peer from the least loaded cluster is moved to the most loaded one (line 7). Afterwards, the load and the reference values of the clusters are recalculated (line 8 and line 9). This procedure is repeated for at most N-1 peers. Now the topology is balanced regarding the number of peers per cluster. If the procedure stops the peer load of every cluster is as close as possible to the given reference value.

C. Optimizing the Master Selection

The previous steps optimize the overlay regarding to the number of clusters and the distribution of the peers over the clusters. The latter are populated in a way that the degree of heterogeneity is minimized regarding the selected peer or cluster properties. Now we have to select in each cluster the most suitable peer as master. In general, a master should possess the necessary resources to cope with the cost of being master and to ensure stable network connectivity. The selection of a master represents, therefore, a protocol strategy, which can again be applied in a centralized or a distributed manner. We consider only the distributed strategy here. There are several ways to do this.

Round-robin methods sequentially select a new master per cluster in each round independently of the peer properties. This strategy is often used in wireless sensor networks, e.g., in [16]. Since mobile collaborative applications are view-aware, each peer can locally apply the strategy without any further communication overhead.

Resource-aware methods select the peer that possesses the ”*best*” resources. They ignore poorly equipped peers that are not suitable for being master. For this, the resource value of each peer must be part of the global knowledge of the group. In addition, also the amount of time a peer has to be master could be taken into account. A fair strategy distributes the burden of being the master equally within the cluster. Last but not least, the connection properties of each peer can be used as a selection criterion.

Auction-based methods [22] use bids offered by the peers. They are difficult to implement because each peer has to send its bid to the group which causes an extra communication overhead. In addition, the fairness of each bid has to be guaranteed. This is a well-known P2P problem. Since a distributed implementation that meets view-aware restrictions is hard to find, auction-based strategies have to mitigate these drawbacks.

V. Integration into Moversight

We have integrated our topology maintenance approach into the *Moversight* protocol [7]. The implementation consists of two parts. The first part enhances the existing membership service of the protocol by a new peer-placing strategy. The second one is a dedicated service that manages the periodic master changes during periods where no peers join or leave.

A. Dynamic Peer Placing

The *dynamic clustering strategy* (DCS) applies the cluster balancing algorithm described in Section IV. The used optimization property is encapsulated in a metric which manages the fair distribution of peers over the clusters. Each time a peer enters or leaves the group, the balancing algorithm is triggered locally by each peer.

In each round, the metric determines the current cluster load and rates the cluster. If balancing is necessary, the most loaded peer is moved. We apply three metrics to handle the cluster heterogeneity as argued in Section IV-B: (1) a *cluster-size metric*, (2) a *resource-value metric*, and (3) a *mixed metric*.

The average number of peers per cluster determines the *cluster-size metric*, as shown in Formula 9. The goal of this metric is the balancing of the number of peers per cluster. The most loaded cluster is the one with the largest number of peers over the average. For each cluster, the metric prefers to select the slave with the smallest ID as the new master, since it represents the most stable member of the cluster. For example, if a cluster consists of peers with IDs 1, 3 and 5, peer 1 is chosen. The *resource-value metric* aims at balancing the average resource value among all clusters. For this purpose, it uses the average of the resource value of each peer (RV_i) as shown in Formula 10.

$$ClusterSize_{avg} = N/C \tag{9}$$

$$ResourceValue_{avg} = 1/N \cdot \sum_{i=1}^{N} R \cdot V_i \tag{10}$$

The *mixed metric* combines the two metrics. It moves peers between clusters based on the cluster-size metrics and selects the master based on the resource value. The concrete relation between the cluster and the node load depends on the chosen metric. If the *cluster size metric* is applied, any node may be selected for move from an overloaded cluster. In case of the *resource value metric*, the most loaded peer (regarding the currently available peer resources) is selected for move. Moving a peer may overload the destination cluster. This is fixed during backward balancing.

B. Role Maintenance Service (RMS)

The DCS strategy balances the overlay whenever a peer joins or leaves the group. If the group composition does not change, the *role maintenance service* (RMS) is responsible to replace "*bad*" masters over the time. RMS is a decentralized service that is applied and synchronized locally. It works without any additional communication overhead and ensures the continuous communication in the group. Each time a peer receives a group message, the service checks whether an optimization is required. If so, the collaboration is interrupted and messages that do not belong to the current view are stored in a next-view buffer. When the last message currently in transmission is delivered, new masters are selected for each cluster using the DCS strategy. To keep the overlay stable no peers are moved across the clusters. As soon as the optimization is finished, the collaboration is re-enabled and the messages from the next-view buffer are processed. To determine the necessity of an overlay optimization, we have defined different *role change metrics* based on the following three criteria: (1) the number of processed messages, (2) the elapsed logical time since the last optimization, and (again) (3) the resource value. The processed message criterion further differentiates (a) the number of messages sent by the master, (b) the number of messages sent by peers of the local cluster, and (c) the total number of messages sent to the group. The logical time criterion counts the logical time steps that are generated by the message transfer of the group. As the logical time correlates with the number of processed messages, the role-change probability depends on the group size or the number of messages which have been disseminated within the group, respectively. The resource criterion monitors the minimum resource value of a master and intervenes if the resource value drops significantly.

VI. Synchronization Issues

Messages exchanged in the collaborating group are associated to a certain view. A maintenance operation may reorder the role and cluster assignment of the peers. Due to network latency, the views of the peers may differ for a small period of time. Thus, it is possible that peers receive messages from older or newer views, respectively. This can destroy the virtual synchrony of the group.

We illustrate this with a group of at least two peers p_1 and p_2 of which p_1 is the master. A message sent by p_1 may trigger a role change. If the global reception time (GT) of this message is delayed for p_2, p_1 may switch to a new view and a new role, whereas p_2 remains in the old view. Immediate additional messages sent by p_1 will never be accepted by p_2 until the peer updates its view. Under poor network conditions, this destroys the group synchrony.

There are two possible solutions to this problem: (1) increasing the timeout values and the number of allowed retransmissions for each transfer, and (2) buffering messages from future views in the meantime. The first approach increases the communication overhead and resource consumption. It also implies the risk of false failure detection assumptions. Therefore, we prefer the second approach. Messages arriving from future views are stored in a next-view buffer. When a peer switches to the new view, this buffer is checked for messages. If there are any, they are handled like normal message transfer.

Another synchronization issue occurs when p_2 sends a message to p_1 in an old view, while p_1 has changed to a new one due to a role change. The message is dropped because it belongs to the past view. This delays the message processing and causes several retransmissions, because p_2 expects an acknowledgement for its message before it is able to update the view. In combination with network latency effects and other group operations, this also destroys the group synchronization. To solve this problem, sending of messages has to be delayed by the local peer when it is currently processing a message that may change the view, until this processing has finished. However, on application layer, there is no blocking issued by the optimization. The optimization results only in a delayed message dissemination.

VII. Performance Evaluation

In order to prove the applicability of our approach we tested the *Moversight* protocol using the OMNeT++ simulator [23]. Our test principle was inspired by [10]. We test the accuracy of our solution and the maximum reachable runtime of the entire group. The group communication protocol is responsible for the accuracy of the information disseminated in the group. If the information accuracy is not given, a dissemination of a message after applying a new topology will fail. Since each peer determines the optimized topology locally, divergent results will destroy the groups virtual synchrony. This results in an impossibility to successfully disseminate messages within an inaccurate topology. Based on this fact, the test proves the accuracy of our solution. Our performance evaluation consisted of 250 test cases. Each test case created a defined group and was executed on a simulated wireless network. During the test, we compared our approach to the one of LEACH, which is – from our perspective – the most relevant clustering protocol usable in the area of pure P2P-networks with closed groups that offer a global knowledge.

The network consisted of up to 20 access points which were interconnected by a wired backbone network consisting of 20 connected routers. The backbone network added a delay between 10ms and 200ms to each transmission. The data rate was uniformly distributed between 100Kbps and 100Mbps. In addition, a random packet loss probability was defined, also uniformly distributed between 1% and 5%. Each access point managed at most five wireless hosts. It was connected to the backbone network via a 100Mbps Ethernet connection.

Each simulated host was connected wirelessly to the network and used the *Moversight* protocol on top of an UDP/IP stack. The network was configured using DHCP. In addition, all hosts used a small battery with a nominal capacity of 10kJ. The wireless part of the network was specified by means of the OMNeT++ INET framework and simulated an IEEE 802.11g network with 54Mbps at 2.4Ghz. The consumption of the wireless interface is characterized by the following settings:

$$\begin{aligned} \text{Off} &= 0.0\text{W} \\ \text{Sleep} &= 0.0495\text{W} \\ \text{Idle} &= 0.6696\text{W} \\ \text{Reception} &= 1.0791\text{W} \\ \text{Transmission} &= 1.7787\text{W} \end{aligned}$$

Each test case distinguished two phases. In the first phase the group was set up with peers that joined and left the group until the desired group size was reached. This simulated a high group dynamic and tested the optimization algorithm for join and leave operations. When the desired group size was reached, the second phase started simulating a stable group without any changes in its composition. If required, the group could select a new master for certain clusters. To test the operation of the role maintenance service (RMS) a peer periodically sent messages to the group. This tested the service execution. The test stopped when any of the group members ran out of energy.

A test case represented a specific collaborating group described by (1) the size of the group and (2) a specific combination of maintenance strategies. (1) The size of the group scaled from ten to one hundred peers, in steps of ten peers. (2) A maintenance strategy X-Y-Z combined each one of three peer placing strategies X, one of six DCS strategy metrics Y, and one of the seven role change metrics Z. Peer placing strategies were: a sequential (X = 0), a parallel (X = 1), and a DCS strategy (X = 2). The DCS strategy metrics comprised the cluster-size (Y = 0), the resource-value (Y = 1), and the mixed peer placing metric (Y = 2). The sequential strategy populated the clusters one by one, the parallel strategy similarly selected clusters in parallel. The applied role change metrics were: no role change at all (Z = 0), total number of messages sent in the group (Z = 1), number of messages sent by a master (Z = 2), number of messages sent by the cluster (Z = 3), the elapsed virtual time (Z = 4), the resource value (Z = 5), and resource value using larger thresholds (Z = 6). A notation of the form Z_0 depicts, for instance, the role change metric with Z = 0, notations like X_0 or Y_0 are to be understood accordingly. In total, 23 different strategy combinations were applied for each desired group size. In conjunction, we evaluated two peer placing strategies of LEACH. The two were used in combination with Z_3 to select the masters and implemented a round-robin based optimization, as discussed in [16]. Our LEACH implementation L-S statically selected the number of masters depending on the group size. For a group up to 20 members, we selected $N/2$ masters, $N/2.5$ masters for groups of up to 60 members, and $N/5$ masters for larger groups. We believe that this is a fair setup to compare our approach with that of LEACH. L-D is a dynamic variant of LEACH that selects its masters like DCS. The master role was changed in round-robin manner within a cluster. In total, we applied 250 test cases. During the tests we measured the simulation run-time. They varied from approximately 5500s up to 8000s.

TABLE I
SIMULATION RESULTS

Run	10	20	30	40	50	60	70	80	90	100
0-0-0	90.1	92.3	93.7	92.3	89.9	93.8	95.2	92.9	94.6	95.7
1-0-0	89.3	92.0	91.2	90.4	89.1	94.9	96.7	93.8	95.5	96.6
2-0-0	89.6	92.0	93.7	91.8	89.6	93.2	94.9	92.6	94.3	95.1
2-0-1	89.9	92.3	93.7	91.8	89.6	93.2	94.9	92.6	94.3	95.1
2-0-2	96.0	92.3	93.7	91.8	89.6	93.2	94.9	92.6	94.3	95.1
2-0-3	89.9	92.3	93.7	91.8	89.6	93.2	94.9	92.6	94.3	95.1
2-0-4	89.9	92.3	93.7	91.8	89.4	93.2	94.9	92.6	94.3	95.1
2-0-5	92.5	95.3	98.5	95.7	94.1	97.9	97.0	90.9	89.5	90.6
2-0-6	92.5	95.3	98.5	96.6	94.9	98.5	96.4	90.9	89.5	90.6
2-1-0	89.6	91.2	95.7	92.1	92.7	98.2	99.7	97.3	98.8	**100**
2-1-1	89.9	91.2	95.7	92.1	92.7	98.2	99.7	97.3	98.8	**100**
2-1-2	89.9	91.2	95.7	92.1	92.7	98.2	99.7	97.3	98.8	**100**
2-1-3	89.9	91.2	95.7	92.1	92.7	98.2	99.7	97.3	98.8	**100**
2-1-4	90.1	92.5	91.8	92.1	92.7	97.9	99.7	97.3	98.8	**100**
2-1-5	**100**	94.2	91.8	90.6	89.1	99.1	**100**	90.3	98.5	92.4
2-1-6	**100**	94.2	94.6	92.1	91.6	98.8	**100**	92.9	99.7	97.5
2-2-0	89.6	92.0	98.0	98.8	93.5	95.2	99.1	90.3	91.9	93.6
2-2-1	95.7	92.5	98.0	94.0	93.5	94.6	99.1	90.3	91.9	93.6
2-2-2	96.5	92.5	98.0	98.8	93.5	95.2	99.1	90.3	91.9	93.6
2-2-3	95.7	92.5	98.0	94.0	93.5	94.6	99.1	90.3	91.9	93.6
2-2-4	93.3	95.8	95.7	92.6	93.5	97.9	99.1	90.3	98.2	93.6
2-2-5	99.2	96.9	97.1	96.8	92.1	**100**	94.6	90.6	**100**	92.1
2-2-6	99.2	**100**	99.4	**100**	92.1	**100**	94.6	**100**	91.6	91.8
L-D	92.0	96.1	**100**	98.8	**100**	99.4	95.8	90.6	89.8	89.7
L-S	91.5	94.2	96.0	95.4	93.8	96.4	95.5	90.9	90.7	93.9

Table I shows the run-time of each test case relative to the longest run for each group size. The best results are marked in bold font. The first column defines the test case with its corresponding combination of strategies – X-Y-Z. The results of the test run for each group size were cumulated depending on the used parameter combination X-Y-Z. Figure 6 shows the resulting maximum run-time and the average run-time of each combination. The relative maximum run-time of each run varied between 89.3% (1-0-0) and 100% (2-1-5), the relative average run-time between 96.92% (2-0-1) and 104% (2-2-6).

The results of DCS were overall the most stable ones. In other words, the maximum run-time of all test cases was close to the average run-time of all test cases for this parameter combination. The mixed DCS metrics performed better than the resource-based DCS metrics. In particular, the results of the combinations 2-1-5 and 2-1-6 were the most stable individual combinations, followed by the combinations 2-0-0, 2-2-5, and 2-2-6. The most unstable strategies were the sequential (0-0-0) and the parallel (1-0-0) peer placing strategies, if used in combination with the role change metric Z_0. The results of the L-D and L-S algorithms were also quite unstable with a difference of 5% and 8%, respectively.

Afterwards, we cumulated the results of the different runs depending on the used peer placing strategy. Figure 7 shows the cumulated results for each strategy, again for the maximum run-time and the average run-time of each peer placing strategy. The abbreviation x-0-x refers to the parallel strategy, x-1-x to the sequential one, x-2-x to DCS, and L-D and L-S to the results of the LEACH variants. The average run-time of the DCS strategy outperforms the sequential and the parallel strategies on average by 2%. The average run-time of L-D and L-S is more or less similar to the one of DCS. The maximum run-time of the DCS strategy is 10% better than that of the sequential strategy, 11% than the parallel one, 5% better than L-D, and 10% better than L-S.

Figure 8 shows the cumulated results of the DCS strategy compared to the ones of L-D and L-S. The average run-time of the mixed DCS metrics was the best and the resource-based was the second best. The average run-time of L-D is similar to the mixed DCS metrics caused by the same cluster creation scheme, and 3% better than the one of L-S.

Figure 9 shows the cumulated results of the RMS metrics. The worst maximum run-time achieved the no-role-change metrics (x-x-0) with 93%, followed by the elapsed-virtual-time metrics (x-x-4) with $95,48\%$. The best maximum run-time reached the resource-value metrics with larger thresholds (Z_6, combination x-y-6) with 100%. The worst average run-time was reached by the RMS metrics total-number-of-sent-messages Z_1 with 76%. L-D had an average run-time of 103% and L-S of 102%. The metrics Z_6 attained the best average performance with 105%.

To sum it up, the performance of the combinations 2-1-5 and 2-1-6 outperformed L-D and L-S by 3-5% on average and 10% in the best case. In addition, we have to take into account that the performance of LEACH depends on the predefined number of clusters. If the number of masters is not suitable for the application use case, its performance may decrease. For this reason, the LEACH improvement L-D is preferable compared to its original implementation, L-S. The performance of the combinations 2-2-5 and 2-2-6 are close to the one of 2-1-5, but it showed better results on average. Thus, we propose DCS in combination with the mixed-peer-placing metrics and a resource-value-based role change as a general solution for clustering and peer placing tasks.

VIII. Conclusion

In this paper, we have proposed a distributed maintenance approach for cluster-based group topologies to support mobile collaborative applications. The approach is generic and can also be applied to other cluster-based topologies, e.g., in mobile ad hoc networks and in sensor networks. The service is activated periodically and consists of two parts: the dynamic clustering strategy (DCS) and the role maintenance service (RMS). DCS is activated when a peer joins or leaves the group in order to dynamically improve the topology. It outperforms simple placing strategies, such as the sequential or parallel peer placing methods. DCS can be used with three metrics, with the resource-based one performing best. RMS is applied in periods without any changes of the group composition to replace overloaded masters. It is based on the DCS strategy and defines six metrics to determine the appropriate time for a role change. The performance evaluation of the *Moversight* implementation showed that the combination of the two strategies provides a stable basis for applications. It increases the run-time in the best case by 10% compared to LEACH, and $3-5\%$ on average. The resulting loads are fairly distributed within the group without any additional communication. DCS itself does not issue any additional traffic, instead the global knowledge is used, already accessible locally by the peer. Optimization decisions are derived by each peer locally from the global knowledge. It ensures small blocking times and a good responsiveness to messages. In detail, from the application perspective, there is no blocking period during the maintenance operation. This enables mobile collaborative applications to progress largely continuously without remarkable interruptions. The proposed metrics can be enhanced by further parameters in the future, e.g., the network bandwidth, the latency of each peer, its CPU load, and its device class. To ensure the quality of the optimization, it is required to periodically refresh the relevant parameters. Furthermore, it could be beneficial to announce this to the group by a lightweight update service instead of using piggybacking. Currently the metrics thresholds are defined hard. An adaptive computation would be advantageous. In addition, new RMS metrics could be created based on a combination of the proposed basic types.

References

[1] J. Gäbler, R. Klauck, M. Pink, and H. König, "uBeeMe; A platform to enable Mobile Collaborative Applications," in *Proc. of the 9th Int. Conf. on Collaborative Computing (CollaborateCom).* Austin, TX, USA: IEEE, 2013, pp. 188–196.

[2] G. V. Chockler, I. Keidar, and R. Vitenberg, "Group communication specifications: A comprehensive study," *ACM Computing Surveys*, vol. 33, no. 4, pp. 427–469, 2001.

[3] K. P. Birman and R. V. Renesse, *Reliable Distributed Computing with the Isis Toolkit.* IEEE Computer Society, 1994.

[4] R. Schollmeier, "A definition of peer-to-peer networking for the classification of peer-to-peer architectures and applications," in *Proc. of the 1th IEEE Int. Conf. on Peer-to-Peer Computing, 2001.* IEEE, 2001, pp. 101–102.

[5] D. Dolev and D. Malki, "The transis approach to high availability cluster communication," *Communications of the ACM*, vol. 39, pp. 64–70, 1996.

[6] O. Waldhorst, C. Blankenhorn, D. Haage, R. Holz, , G. Koch, B. Koldehofe, F. Lampi, C. Mayer, and S. Mies, "Spontaneous Virtual Networks: On the Road towards the Internet's Next Generation," *it - Information Technology*, vol. 50, no. 6, pp. 367–375, Dec. 2008.

[7] J. Gäbler and H. König, "Moversight: An approach to support mobility in collaborative applications," in *Proceedings of the 10th Annual Conference on Wireless On-Demand Network Systems and Services (WONS), 2013.* IEEE Communication Society, March 2013.

[8] ——, "Moversight: a group communication protocol for mobile scenarios," *Telecommunication Systems*, vol. 61, no. 4, pp. 1–22, May 2015.

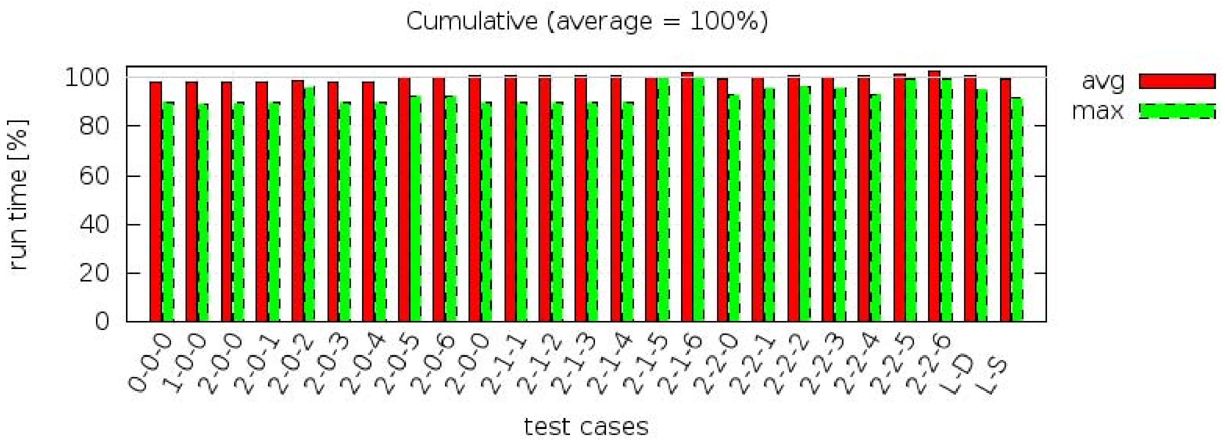


Fig. 6. Maximum and average run-time per test case.

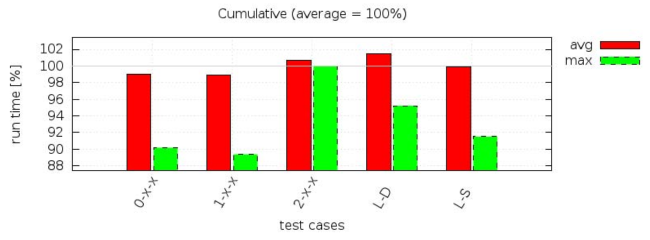


Fig. 7. Cumulated placing strategies results compared to L-D and L-S.

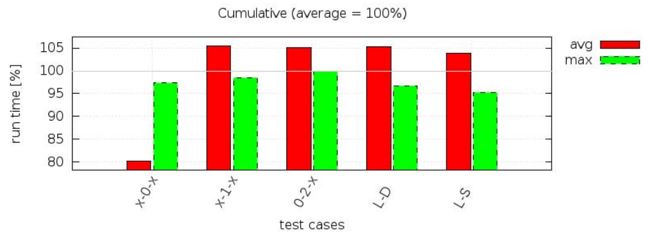


Fig. 8. Cumulated placing strategies metrics compared to L-D and L-S.

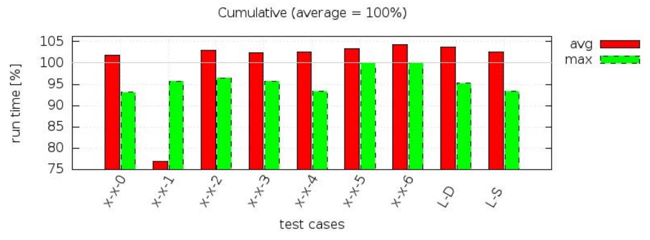


Fig. 9. Cumulated results of the RMS metrics compared to L-D and L-S.

[9] A. Olteanu, F. Pop, C. Dobre, and V. Cristea, "An adaptive scheduling approach in distributed systems," in *Proc. of the IEEE 6th Int. Conf. on Intelligent Computer Communication and Processing.* IEEE, 2010, pp. 435–442.

[10] A. A. Abbasi and M. Younis, "A survey on clustering algorithms for wireless sensor networks," *Computer Communications*, vol. 30, no. 14-15, pp. 2826–2841, Oct 2007.

[11] V. Gramoli, A.-M. Kermarrec, E. L. Merrer, and D. Neveux, "Sonde, a self-organizing object deployment algorithm in large-scale dynamic systems," in *Proc. of the 7th European Dependable Computing Conference (EDCC 2008).* IEEE, 2008, pp. 157–166.

[12] B. Li and K. H. Wang, "Nonstop: Continuous multimedia streaming in wireless ad hoc networks with node mobility," *IEEE Journal on Selected Areas in Communications*, vol. 21, no. 10, pp. 1627–1641, 2003.

[13] C. Fu, R. H. Glitho, and F. Khendek, "Signaling for multimedia conferencing in stand-alone mobile ad hoc networks," *IEEE Trans. on Mobile Computing*, vol. 8, no. 7, pp. 991–1005, 2009.

[14] B. Elbhiri, S. Fkihi, R. Saadane, N. Lasaad, A. Jorio, and D. Aboutajdine, "A new spectral classification for robust clustering in wireless sensor networks," in *Proc. of the 6th Joint IFIP Wireless and Mobile Networking Conference (WMNC 2013)*, April 2013, pp. 1–10.

[15] S. Basagni, "Distributed clustering for ad hoc networks," in *Proc. of the 4th International Symposium on Parallel Architectures, Algorithms, and Networks, 1999. (I-SPAN '99)*, Australia, June 1999, pp. 310–315.

[16] W. B. Heinzelman, A. P. Chandrakasan, and H. Balakrishnan, "An application-specific protocol architecture for wireless microsensor networks," *IEEE Trans. on Wireless Communications*, vol. 1, no. 4, pp. 660–670, 2002.

[17] M. Frincu, N. M. Villegas, D. Petcu, H. A. Müller, and R. Rouvoy, "Self-healing distributed scheduling platform," in *Proc. of the 11th IEEE/ACM Int. Symposium on Cluster, Cloud and Grid Computing (CCGrid)*, Newport Beach, USA, 2011, pp. 225–234.

[18] K. P. Birman, "Building secure and reliable network applications," in *Worldwide Computing and Its Applications*, ser. LNCS. Springer, 1997, vol. 1274, pp. 15–28.

[19] L. Lamport, "Time, clocks, and the ordering of events in a distributed system," *Communications of the ACM*, vol. 21, pp. 558–565, 1978.

[20] P. K. Agarwal and C. M. Procopiuc, "Exact and approximation algorithms for clustering," *Algorithmica*, vol. 33, no. 2, pp. 201–226, 2002.

[21] M. Aigner, "A characterization of the bell numbers," *Discrete Mathematics*, vol. 205, no. 1-3, pp. 207–210, 1999.

[22] N. Mohammed, H. Otrok, L. Wang, M. Debbabi, and P. Bhattacharya, "Mechanism design-based secure leader election model for intrusion detection in manet," *IEEE Trans. Dependable and Secure Computing*, vol. 8, no. 1, pp. 89–103, 2011.

[23] A. V. Rudolf and Hornig, "An Overview of the OMNeT++ Simulation Environment," in *Proc. of the 1st Int. Conf. on Simulation Tools and Techniques for Communications, Networks and Systems & Workshops*, Belgium, 2008, pp. 1–10.