A Survey of Current Directions in Service Placement in Mobile Ad-hoc Networks

Georg Wittenburg
wittenbu@inf.fu-berlin.de

Jochen Schiller
schiller@inf.fu-berlin.de

Department of Mathematics and Computer Science
Freie Universität Berlin
Takustr. 9, 14195 Berlin, Germany

Abstract

Service placement deals with the problem of selecting which node in a network is most suitable for hosting a service that responds to queries from other nodes. Optimally placing services reduces network traffic and improves connectivity between clients and servers. Service placement algorithms may thus be regarded as an interesting building block for research into service-oriented middleware. Recently, new approaches to address the service placement problem in the field of ad-hoc networking have been proposed. This paper surveys, classifies and evaluates ten representative approaches, thereby providing a summary of the state of the art in service placement.

1 Introduction

In Mobile Ad-hoc Networks (MANETs), the classical distinction between clients and servers as far as physical devices are concerned is replaced by the paradigm of cooperation between a large set of potentially heterogeneous devices. Given the effects of communication over a wireless channel and mobility of the devices, the physical topology of any MANET is in a constant state of flux. In order to optimize the performance of the network as a whole, it is therefore necessary to continuously adapt the logical network topology to both external (e.g. connectivity, mobility, churn) and internal (e.g. communication patterns) factors.

In the past decade, the focal point of research has been to optimize the routing of packets between nodes of a MANET. Taking a more application-centric view on MANETs, cooperation between devices is structured by application-level clients and servers, i.e. certain nodes request services provided by other nodes. Assuming that the heterogeneity of the MANET is limited, i.e. the majority of the nodes have roughly similar amounts of computational resources available to them, the question arises in how far the performance of the network can be increased by intelligently selecting which nodes are to host a particular service. The process of identifying the appropriate nodes in a MANET to act as servers is referred to as the *service placement problem*. For a typical use case, consider a directory service that is hosted on a node whose position in the network is optimal in terms of overall routing hops to clients of the service. If the node that hosts the service moves away from its current position, a service placement algorithm can determine when to migrate the service and which node is best suited to be the new host.

The contributions of this paper are to survey recent approaches to service placement in MANETs (Section 3), to classify and evaluate these approaches (Section 4), and to summarize the state of the art (Section 5). In order to establish a common ground, we begin in Section 2 by stating the service placement problem and key research questions, clarifying terms, and covering the theoretical background.

2 The service placement problem

The problem of service placement in MANETs can be stated as follows: Given a physical network topology and service demands of client nodes, adapt the number and location of services in the network over time such that the service demands are met at a minimal cost. The cost function may include metrics such as network traffic, energy expenditure or service-dependant Quality of Service (QoS) metrics (availability, latency, etc.).

The key questions to be answered as part of solving the service placement problem are thus as follows:

- *Where* in the network are services or service instances of a distributed service to be placed?
- *How many* service instances are required for cost optimal operation?

- *When* should the current configuration of services and service instances be adapted?
- *How* are services and their state to be transfered from one node to another?

The fourth question has been covered in other research areas, especially in the field of mobile agent systems [13], and will not be discussed in this paper.

2.1 Clarification of terms

A **service** is a software component executed on one or several nodes of the network. It replies to the service requests it receives from client nodes through a well-defined interface. The content, structure and sequence of service requests and replies are specific to the function of the service and the protocol that describes the interaction between client and server nodes. Several different services may be active in a network simultaneously.

This intentionally broad view on services may be narrowed down according to the following properties:

- *Node-specific vs. node-independent:* A service may be tied to a specific node, e.g. providing access to special-purpose hardware, or it may work independently of the node it is located on, e.g. as a directory service or a cluster head for hierarchical routing. Obviously, service placement is only applicable to node-independent services.
- *Centralized vs. distributed:* The semantics of the service may require for it to run centralized on one node or distributed in the form of an adaptable number of identical service instances (see below) spread over multiple nodes.
- *Composite vs. monolithic:* The modular structure of a composite service allows for it to be split into multiple interdependent subservices that may run on different nodes. In contrast, monolithic services are indivisible.
- *Message-based vs. streaming:* For some services, e.g. a query to a directory, the data provided by the service may fit into a single message. Alternatively, the request may be served by a continuous stream of data, e.g. when delivering multi-media content.

If the same service component is executed on several nodes, these components are referred to as **service instances**. The idea is that a service may be distributed over the network by spawning new instances depending on demand. Service instances may have to exchange information to coordinate among themselves or synchronize shared data. In case of a centralized service, the terms service and service instance may be used interchangeably.

The **service demand** is the set of scenario-dependant service requests of a client node over a period of time. Meeting the service demands of all client nodes is the primary goal of the network as a whole. Metrics that quantify the success of a service placement algorithm need to take into account in how far this goal has been reached.

The **service provision cost (SPC)** is the abstract cost incurred while service requests are served over time. In the context of MANETs, it is commonly identified with the network traffic. Common metrics are both packets and bytes transfered over the wireless network interface.

The **control traffic overhead (CTO)** is the additional control information about the state of the network that a service placement algorithm may require nodes to exchange. The magnitude of this overhead depends on the approach to service placement and its implementation. Any increase in CTO reduces the benefits achieved by the reduction of SPC as defined above. The CTO is thus to be kept at a minimum and to be weighted carefully against its potential to reduce SPC.

2.2 Theoretical background

The service placement problem is closely related to facility location theory from the area of operations research. Two problems from facility location theory are applicable to the service placement problem: the k-median problem and the facility location problem. Both of them work on a set of facilities F and a set of clients C. For every pair $i, j \in F \cup C$, there is a cost $c_{i,j} \geq 0$. The solution to both problems is to find a subset of open facilities $S \subseteq F$ to serve every client $j \in C$ by a facility $\sigma(j) \in S$, but under different constraints:

- *k-Median Problem:* Given a fixed number of facilities $k < |F|$, minimize the total service cost $\sum_{j \in C} c_{\sigma(j),j}$.
- *Facility Location Problem:* Given a cost $f_i \geq 0$ for opening facility $i \in F$, minimize the sum of the total facility cost and the total service cost $\sum_{i \in S} f_i + \sum_{j \in C} c_{\sigma(j),j}$.

Both problems are NP-hard. They are called *metric*, if $c_{i,j}$ is symmetric and satisfies the triangle inequality. Both problems can be stated in their *capacitated* or *uncapacitated* form, meaning that the resources available at the facilities are either limited or not. Traditional solutions to both problems are not easily applicable in the context of MANETs, because the algorithms are centralized and require global knowledge about facilities, clients and costs. Recently however, distributed approaches are increasingly being explored [4, 7, 8, 11].

3 State of the Art

In the following subsections, we will now proceed to survey representative approaches to service placement. Our main focus are algorithmic aspects, protocol mechanisms and the evaluation conducted by the authors.

3.1 Placement of a centralized service based on network topology

The REDMAN middleware, as described by Bellavista et al. [2], aims at supporting resource replication in dense MANETs. While the placement of replicas relies on a simple heuristic, the placement of the controlling entity, the *replication manager*, considers local network topology.

The process of selecting a node to host the replication manager is started once a node detects that it has become part of a dense network region, i.e. the number of nodes in the single-hop neighborhood exceeds a threshold value, and no replication manager is present. This node then starts a broadcast-based query that establishes in which topological direction the most distant node of the dense MANET is located. Once this direction is known, a node at a one-hop distance in that direction is requested to perform a similar query. Heuristics are used to end this process near the topological center of the dense MANET, and the last node in the series of queries becomes the replication manager.

The REDMAN middleware is evaluated using the network simulator `ns-2`. The results indicate that both the number of messages incurred by the process of selecting the replication manager and the number of required iterations scale well with the number of nodes. Placement inaccuracy as measured by hop count to the optimal node is below one hop on average, but increases with mobility.

3.2 Iterative migration of a centralized service based on service demand

Oikonomou and Stavrakakis [12] propose a policy for placing a single service in a MANET. Similar to hill climbing algorithms, their approach is to iteratively migrate the service from its current host node to the neighboring node through which traffic accounting for more than half of the SPC is routed. If no such node exists, the service remains at its current location. This policy has the advantage that it incurs no CTO and service placement restarts in case service demands or network topology change.

The authors prove analytically that under the assumption of a tree-like network topology their algorithm eventually reaches the globally optimal position. As a consequence, the results are best applicable to MANETs that employ a routing algorithm that organizes nodes into minimal spanning trees.

3.3 Iterative migration and placement of multiple centralized services

The MagnetOS project as described by Liu et al. [10] implements a distributed operating system for MANETs that makes the entire network appear as a single virtual machine. Software components, e.g. Java objects, are dynamically migrated across the network to minimize SPC, which in this case is incurred through remote method invocation.

The algorithmic approach is to discretize time into epochs. At the end of each epoch each of the software components is relocated according to one of five strategies:

- *LinkPull:* Move the software component one hop in direction of the highest SPC
- *PeerPull:* Place the software component on the host that caused the highest SPC
- *NetCluster:* Place the software component on a random node in the one-hop cluster whose nodes caused the highest SPC
- *TopoCenter(1):* Place the software component on a node that, based on local knowledge about the network topology, minimizes the sum of migration cost and the expected future SPC
- *TopoCenter(Multi):* As above, but with all nodes sharing additional information on network topology

The service placement algorithms used in MagnetOS are evaluated using the `sns` network simulator with an energy model validated against a testbed of 16 laptops communicating over IEEE 802.11b. Simulations of three different scenarios show that all algorithms perform similarly well in extending the lifetime of the system through reduction of SPC. TopoCenter(1) is most successful with an increase by a factor of 2.5.

3.4 Placement of a distributed service based on node mobility

As part of the NonStop project, Wang and Li [9] address the problem of service availability in the light of network partitions due to node mobility. Their approach is to group nodes by their velocity vectors and to predict the event of such a group moving out of the radio range of another group, thus partitioning the network. If a single node provides a service to both mobility groups, a new service instance is created on one node in the mobility group that would be left without access to the service otherwise.

The algorithm to establish which node in the departing group should host the new service instance is run centrally on the node that is currently hosting the service. Information about the locations and velocities of the departing

nodes is piggybacked on service requests. Based on this information, the current host selects the node to which the data required to provide the service can be transferred before the network partition occurs. Redundant service instances in the same mobility group shutdown once they detect the availability of a service instance with a higher unique identification number.

The results of simulations indicate full service coverage can be achieved for MANETs whose nodes have correlated mobility patterns. If nodes move randomly, the coverage drops slightly but remains between 80% and 90%. This is in part due to the decreased likelyhood of network partitions.

3.5 Placement of a distributed service based on service coverage

Sailhan and Issarny [14] propose an architecture for service discovery in MANETs built around a homogeneous deployment of cooperating service directories. The idea is to minimize global service discovery overhead by handling service discovery requests locally at a nearby directory.

In order to achieve a homogeneous placement of directories, any node without access to a directory broadcasts a query for available resources and network topology information to nodes within its n-hop neighborhood. Queried nodes reply either with the requested information or with a message stating that they are unable to host a directory. Using this data, the initiating node selects a node for hosting a new directory. The main selection criteria is the expected coverage of the new directory in terms of number of neighbors and number of other directories in the vicinity. The node that best matches these criteria is then notified of the decision and initializes a new service directory by requesting data from existing directories.

The proposed architecture is evaluated using the network simulator `ns-2`. The results show that the overall network traffic is minimal for a neighborhood size of two hops. For high request rates, traffic is lower than the one incurred by flooding-based approaches without service directories. Further, service discovery latency is shown not to be affected by increased mobility.

3.6 Placement of a distributed service by limiting scope

Laoutaris et al. [8] present a distributed algorithm to solve the Uncapacitated k-Median (UKM) and Uncapacitated Facility Location (UFL) problems. The intuition behind their approach is to compensate for the lack of global knowledge by limiting the scope of the problems to the n-hop neighborhood of the nodes currently hosting a service instance. For this region, exact knowledge of network topology and service demands is assumed to be available. Service demand of the remaining nodes is taken into account by mapping it to the outer nodes of the neighborhood. Since the approach considers multiple service instances, there may be multiple overlapping neighborhoods, in which case they need to be merged and considered collectively. These steps are applied iteratively to all service instances until the set of relocated services instances is empty.

In their evaluation, the authors use synthetic network graphs as well as real Internet topologies at the level of autonomous systems to compare their distributed approach to a centralized algorithm. The results show that the SPCs incurred by the placements found by each of the algorithms closely track each other for different numbers of service instances and network sizes. Further, smaller neighborhoods require more iterations of the algorithm, thus increasing the CTO.

3.7 Placement of a distributed service by iterative negotiation

Frank and Römer [4] propose a distributed algorithm for solving the UFL problem that operates by iteratively negotiating the optimal assignment of services between clients and service instances. Based on preexisting knowledge about which nodes *may* host service instances, each of these nodes gathers information about the local topology, calculates the resulting SPC for each potential client in its one-hop neighborhood, and sends this data to the potential client nodes. Client nodes may receive this information from more than one potential service instance, but only reply to the instance that advertises the lowest cost. This process is repeated until potential service instances and clients agree on a mapping. Multi-hop placement is achieved by exponentially increasing the neighborhood size in successive iterations of this procedure.

The authors prove the equivalence of their distributed algorithm and an existing centralized algorithm. Simulations illustrate that the algorithm finds a placement with costs close to the optimal costs independently of the size of the network and performs better than an approach based on minimum dominating sets. Quickly increasing the neighborhood size in the outer, multi-hop loop of the algorithm results in less overall iterations of the inner loop, independent of the network density. The findings are further supported by an experiment with a testbed of 13 nodes in which the gap between real and optimal cost never exceeded 10% in a 24 hour run.

3.8 Placement of a distributed service by distributed linear programming

Moscibroda and Wattenhofer [11] describe a distributed algorithm to solve the UFL problem and study the trade-

off between communication overhead and quality of the approximation. Based on the assumption that each potential service host is able to exchange data with each client (and vice versa) in each communication round of the algorithm, their approach is to describe facility location as a fractional linear program which can be approximated within a constant number of rounds. Distributed randomized rounding is then used to map the fractional results of the previous step back to the original UFL problem.

In their analysis, the authors prove that the quality of the approximation achieved by their algorithm is bounded by the given number of communication rounds used during the calculation process.

3.9 Placement of a distributed service by majority voting

Krivitski et al. [7] propose a different approach to solving the UFL problem in a distributed manner. Their algorithm is based on a large number of local majority votes that nodes use to agree on the next step of a hill climbing algorithm. Hill climbing is a heuristic that has been shown to achieve a constant factor approximation to the globally optimal solution for the UFL problem. CTO is kept low by avoiding majority votes on alternative steps that would not reduce the cost as compared to a currently known best step candidate.

Based on simulations with three different topologies, the authors show that the solution converges towards the globally optimal solution and scales well in terms of CTO for different network sizes.

3.10 Placement of a distributed service by centralized selection

Furuta et al. [5] apply facility location theory to the problem of selecting nodes to act as cluster heads for hierarchical routing. Following the general approach of the LEACH-C clustering protocol [6], they propose to run a centralized selection algorithm on the base station of the network. The major innovation in their approach is that they formulate the problem as an UFL problem (as opposed to an UKM problem in LEACH-C) to allow for a variable number of cluster heads even if candidate nodes are low on energy.

The evaluation of the algorithm is conducted using computational experiments and focuses on longevity of the network. The results indicate that the UFL-based approach is able to maintain a working structure of clusters for a longer period of time and to significantly prolong the overall lifetime of the network. These improvements are attributed to the increased flexibility when considering nodes as potential cluster heads.

Table 1. Classification of approaches.

Supported type of service **Main protocol mechanism**	Centralized service	Distributed service
Passive monitoring	[12] (Sec. 3.2)	
Piggybacking	[10] (Sec. 3.3)	[5] (Sec. 3.10) [1] [9] (Sec. 3.4) [2]
Limited broadcasts		[14] (Sec. 3.5) [3]
Iterative limited broadcasts	[2] (Sec. 3.1)	[4] (Sec. 3.7) [7] (Sec. 3.9) [8] (Sec. 3.6)
Iterative unlimited broadcasts		[11] (Sec. 3.8)

[1] Service placement algorithm is run centrally on base station
[2] Places service instance on any node within mobility group
[3] Only supports placement of one service instance per neighborhood

The approaches presented so far focus on the placement of monolithic services, while other approaches specifically target composite services. Subservices of composite services are commonly organized in a tree-like structure which is then mapped to the network topology. Placement algorithms follow different paradigms, e.g. iterative migration [3], limited flooding [1] and divide & conquer with iterative refinements [15]. We omit a discussion of these approaches due to space limitations.

4 Classification and evaluation

When comparing the different approaches presented in the previous section, the most interesting aspects are the scope of the proposed solutions and their performance in terms of communication efficiency. We classify the approaches in scope by whether they support the placement of distributed or merely of centralized services. To compare their efficiency, we further classify them according to their main protocol mechanism. The idea is that the protocol mechanism relates directly to the CTO incurred while performing service placement, and is thus a good way to qualitatively compare the efficiency. The following protocol mechanisms are commonly employed:

- *Passive monitoring:* Information is extracted from service-related network traffic.
- *Piggybacking:* Information is sent together with service-related network traffic.
- *Limited broadcasts:* Information is sent to all nodes in a bounded, n-hop neighborhood.
- *Iterative limited broadcasts:* As above, but the process is repeated several times.
- *Iterative unlimited broadcasts:* Information is repeatedly sent to all nodes in the network.

Following this classification, a comparison of all approaches discussed in this paper is shown in Table 1. We observe that placing a single centralized service can be achieved by passive monitoring alone. In order to support placing the service instances of a distributed service, iterative limited broadcasts are required by all approaches, with the exemption of those that only consider special cases of service placement.

Further, it is interesting to note that service placement in MANETs is either tackled as a byproduct of middleware research or as an application of facility location theory. The middleware-related approaches [2, 9, 10, 14] commonly employ heuristics based on information gathered from nodes in the neighborhood of the node that is currently hosting the service. Some heuristics are tailored to the specific application, e.g. coverage [14], topology [2] or group mobility [9], and are thus not applicable to general-purpose service placement. None of these algorithms supports truly distributed services.

The approaches with a background in facility location theory [4, 5, 7, 8, 11, 12] either solve the UFL problem with traditional algorithms on a central node after collecting the necessary information from the network [5, 8] or use distributed iterative approximations [4, 7, 11]. Except for [12], all of them support adapting the number of service instances to the current service demand.

None of the approaches discussed in this paper considers placement of disjoint services that compete for available bandwidth, service traffic patterns (message-based vs. streaming) or synchronization traffic between service instances. It will be interesting to see whether the theoretical approaches can be adapted to take these application-centric factors into account.

5 Conclusion and future directions

In this paper, we have summarized, classified and evaluated ten recently published approaches to service placement in MANETs. With respect to the questions raised in Section 2, it can be concluded that while current state-of-the-art research provides answers to the first two questions (*where* and *how many*), the third question (*when*) has not been addressed in depth as of this writing. As the processes of migrating services and duplicating service instances are costly operations in terms of network traffic, the importance of the decision when to adjust the current placement is not to be underestimated.

Looking at the bigger picture, the interactions between service placement and existing service discovery and routing protocols are also widely unexplored. A middleware that intelligently combines these building blocks has the potential to operate more efficiently than one that considers them separately.

References

[1] Z. Abrams and J. Liu. Greedy is Good: On Service Tree Placement for In-Network Stream Processing. In *26th IEEE Intl. Conf. on Distributed Computing Systems*, Lisboa, Portugal, July 2006.

[2] P. Bellavista, A. Corradi, and E. Magistretti. REDMAN: An Optimistic Replication Middleware for Read-only Resources in Dense MANETs. *Journal on Pervasive and Mobile Computing*, 1(3):279–310, Aug. 2005.

[3] B. J. Bonfils and P. Bonnet. Adaptive and Decentralized Operator Placement for In-Network Query Processing. In *2nd Intl. Workshop on Information Processing in Sensor Networks*, Palo Alto, USA, Apr. 2003.

[4] C. Frank and K. Römer. Distributed Facility Location Algorithms for Flexible Configuration of Wireless Sensor Networks. In *3rd IEEE Intl. Conf. on Distributed Computing in Sensor Systems*, Santa Fe, USA, June 2007.

[5] T. Furuta, M. Sasaki, F. Ishizaki, A. Suzuki, and H. Miyazawa. A New Clustering Algorithm Using Facility Location Theory for Wireless Sensor Networks. Technical Report NANZAN-TR-2006-04, Nanzan Academic Society, Mar. 2007.

[6] W. B. Heinzelman, A. P. Chandrakasan, and H. Balakrishnan. An Application-Specific Protocol Architecture for Wireless Microsensor. *IEEE Trans. on Wireless Networking*, Oct. 2002.

[7] D. Krivitski, A. Schuster, and R. Wolff. A Local Facility Location Algorithm for Large-Scale Distributed Systems. *Journal of Grid Computing*, 2006.

[8] N. Laoutaris, G. Smaragdakis, K. Oikonomou, I. Stavrakakis, and A. Bestavros. Distributed Placement of Service Facilities in Large-Scale Networks. In *26th Annual IEEE Conf. on Computer Communications*, Anchorage, USA, May 2007.

[9] 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*, 21(10):1627–1641, Dec. 2003.

[10] H. Liu, T. Roeder, K. Walsh, R. Barr, and E. G. Sirer. Design and Implementation of a Single System Image Operating System for Ad Hoc Networks. In *3rd Intl. Conf. on Mobile Systems, Applications, and Services*, Seattle, USA, June 2005.

[11] T. Moscibroda and R. Wattenhofer. Facility Location: Distributed Approximation. In *24th ACM Symp. on the Principles of Distributed Computing*, Las Vegas, USA, July 2005.

[12] K. Oikonomou and I. Stavrakakis. Scalable Service Migration: The Tree Topology Case. In *5th Annual Mediterranean Ad Hoc Networking Workshop*, Lipari, Italy, June 2006.

[13] G. P. Picco. Mobile Agents: An Introduction. *Journal of Microprocessors and Microsystems*, 25(2):65–74, Apr. 2001.

[14] F. Sailhan and V. Issarny. Scalable Service Discovery for MANET. In *3rd IEEE Intl. Conf. on Pervasive Computing and Communications*, Kauai, USA, Mar. 2005.

[15] B. Seshasayee, R. Nathuji, and K. Schwan. Energy-Aware Mobile Service Overlays: Cooperative Dynamic Power Management in Distributed Mobile Systems. In *4th Conf. on Autonomic Computing*, Jacksonville, USA, June 2007.