Service Placement in Ad Hoc Networks

Georg Wittenburg and Jochen Schiller
Department of Mathematics and Computer Science
Freie Universität Berlin
Takustr. 9, 14195 Berlin, Germany
Email: {wittenbu,schiller}@inf.fu-berlin.de

Georg Wittenburg is a Research Assistant and PhD candidate in Computer Science at Freie Universität Berlin. He received his M.Sc. and B.Sc. in 2006 and 2004 from the same university. His research interests include protocols and distributed systems for wireless, ad hoc and sensor networks, in particular the topics of service placement, distributed event detection and simulation accuracy.

Prof. Dr.-Ing. *Jochen H. Schiller* is head of the working group Computer Systems & Telematics at the Institute of Computer Science, Freie Universität Berlin, Germany. Dr. Schiller studied Computer Science at the University of Karlsruhe where he also got his PhD in 1996. His research focus is on wireless, mobile, and embedded devices, communication protocols, operating systems for devices with small footprint, and quality of service aspects in communication systems.

ABSTRACT

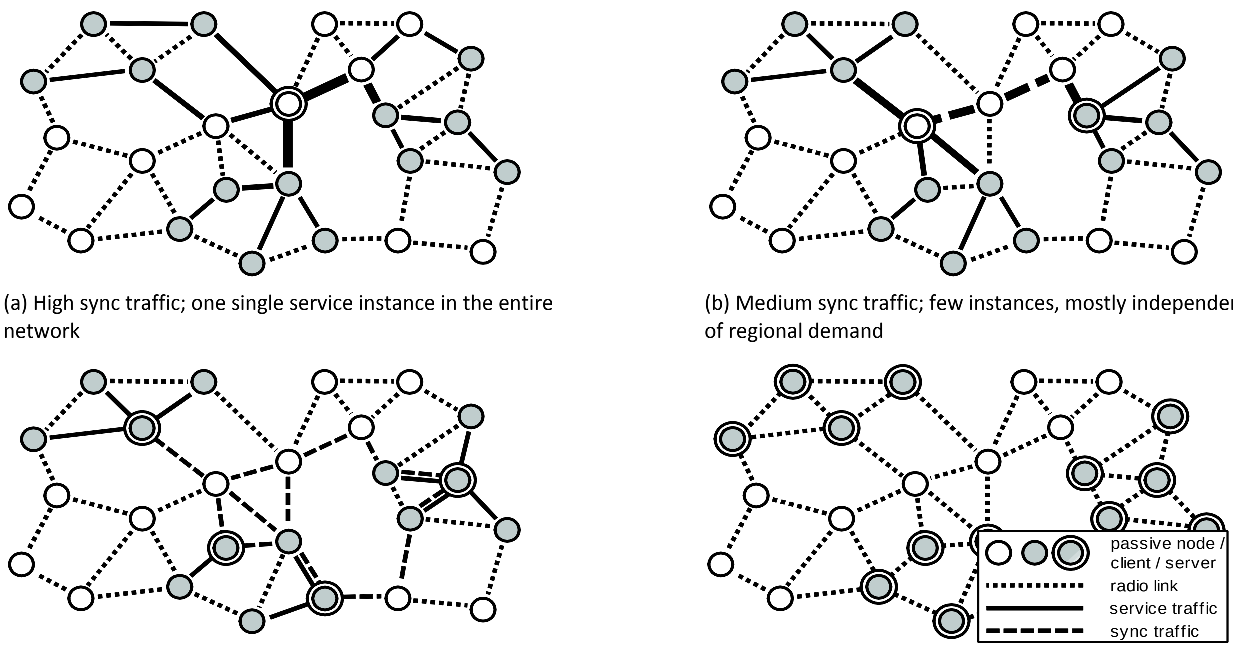
Service placement optimizes the number of instances of a service and their location within an ad hoc network in light of changes in service demand and network topology. The goal is to reduce bandwidth usage and latencies between clients and servers, while improving scalability and availability of the service.
We propose the SP*i* service placement architecture which is unique in that it addresses the interdependencies between service placement, service discovery, and routing. We give an overview of the system, discuss the service model and algorithms with emphasis on the cost of synchronization between service instances, and present simulation-based results that illustrate the benefits of service placement when compared to traditional client/server architecture.

1. INTRODUCTION

Service provisioning in ad hoc networks is challenging given the difficulties of communicating over a wireless channel and the heterogeneity and mobility of devices. In order to optimize the performance of the network as a whole, it is necessary to continuously adapt the logical network topology to both external (e.g., wireless connectivity, mobility, churn) and internal (e.g., communication patterns, service demand) factors. Recent proposals (1) (2) (3) advocate that the nodes in the network should dynamically choose which subset of them should provide application-level services to other nodes. The process of selecting an optimal service configuration, i.e., a set of nodes to host a service in light of a given service demand and network topology, is referred to as *service placement*.
Service placement addresses the questions of *how many* instances of the same service should be available in the network and cooperate to process clients' service requests; *where* these service instances should be placed, i.e., which nodes are best suited for hosting them; and *when* to adapt the current service configuration. The main benefit of this approach is that the service configuration, i.e., the set of nodes to host a service, is adapted automatically at run-time. A good service configuration reduces overall network traffic and latency, and it can also be used to optimize the network performance according to service-specific metrics.
In our prior work (4), we discussed the general applicability of service placement and gave a survey of recent approaches. We found that the service placement problem is either tackled as a byproduct of middleware research or as an application of facility location theory. The interactions between service placement and existing service discovery and routing protocols are widely unexplored, which gives raise to questions regarding optimality, stability, and overhead of current solutions.

2. SERVICE PLACEMENT AND SYNCHRONIZATION

A key aspect in establishing the optimal service configuration is the traffic between service instances. Service instances need to communicate among themselves in order to keep global state and data synchronized. For example, a service instance of a DNS-like service has to propagate a client's update to a record to all other instances. In the following, we make the simplifying assumption that for a service s the synchronization traffic between two service instances can be expressed as a fraction τ_s of the traffic between an instance and its clients. τ_s can either be preconfigured using service-specific expert knowledge, or it can calculated based on the traffic statistics of the service, e.g., by a middleware performing deep packet inspection. The fraction τ_s specifically includes any service-level processing of the service requests, such as aggregation or compression.
Figure 1 illustrates the interdependency between the fraction of synchronization traffic τ_s and the optimal service configuration. If a high volume of synchronization traffic, as it may be the case for a distributed database that provides transactional semantics, the optimal service configuration is to have only a single instance of the service (cf. Fig. 1a). On the other extreme, if no synchronization traffic is required, e.g., for a spell checking service, each client hosts its own service instance (cf. Fig. 1d). For the more interesting cases



(a) High sync traffic; one single service instance in the entire network

(b) Medium sync traffic; few instances, mostly independent of regional demand

(c) Low sync traffic; several instances, strongly related to regional demand

(d) No sync traffic ($\tau_s = 0$); one service instance per client

Fig. 1: Service configurations for different levels of synchronization traffic

of medium to low synchronization traffic (cf. Figs. 1b and 1c), the service configuration needs to be calculated using a cost model based on a (partially known) network topology and the service demand.

For a service s, given the set of nodes N, distances between nodes $d_{m,n}$ with $m, n \in N$, and the set of current hosts of service instances $H_s \subseteq N$, the following parameters are used to calculate the service provisioning cost $c(s)$:

$\delta(s,c)$	Demand for s on client node $c \in N$ in terms of required network traffic
$\tau_s \geq 0$	Fraction of service requests processed by a service instance of s required for synchronization with the other service instances

The cost of providing service s hosted on node $h \in H_s$ to a given client on node c is the product of the distance between h and c and the service demand at the client $d_{h,c}\delta(s,c)$. If there is more than one service instance, i.e., $|H_s| > 1$, then the service discovery process can be assumed to result in the closest (with regard to a metric reflecting the network topology) host being used by each client. Thus, the service host used by client c to satisfy service requests for service s is $\eta(s,c) = \arg\min_{h \in H_s} d_{h,c}$. Thus, the cost of providing service s to a single client c is $d_{\eta(s,c),c}\delta(s,c)$. With the set of clients of a host h for service s given by $C_{s,h} = \{c \in N \mid \eta(s,c) = h\}$, the cost of a service host h can be written as

$$c_{\text{clients}}(s,h) = \sum_{c \in C_{s,h}} d_{\eta(s,c),c}\delta(s,c)$$

As introduced above, service instances may need to synchronize their global state and data. The fraction of the service requests received by one service instance required for the synchronization to another service instance is given by τ_s.

The demand for service s at host h is $\sum_{c \in C_{s,h}} \delta(s,c)$. Hence, the synchronization cost for one service instance of service s hosted on node h is

$$c_{\text{sync}}(s,h) = \tau_s \sum_{h' \in H_s \setminus \{h\}} d_{h,h'} \sum_{c \in C_{s,h}} \delta(s,c)$$

The service provisioning cost for a service s combines the cost incurred by each service instance $h \in H_s$ while serving client requests and while synchronizing global state and data between all service instances:

$$c(s) = \sum_{h \in H_s} c_{\text{clients}}(s,h) + c_{\text{sync}}(s,h)$$

Based on this cost function, the goal of a placement algorithm is to find the optimal service configuration $\widehat{H}_s \subseteq N$ of nodes to host a service s that minimizes the service provisioning cost, i.e., $\widehat{H}_s = \arg\min_{H_s \subseteq N} c(s)$.

3. THE SPI SERVICE PLACEMENT ARCHITECTURE

We propose the SP*i* service placement architecture as a novel approach to service placement in ad hoc networks that optimizes the number and the location of service instances based on usage statistics and a partial network topology derived from routing information. The system only requires minimal knowledge about the service it is tasked with placing in the network, and it is unique in that it explicitly considers the communication between service instances that is required to synchronize shared data. Furthermore, our system implements a cross-layering approach to take advantage of the interdependencies between service placement, service discovery and the routing of service requests to reduce network overhead.

The three main components of the SP*i* architecture are depicted in Figure 2. The *service placement middleware* is

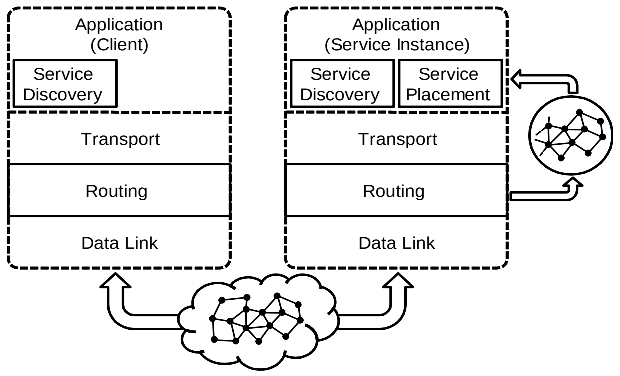


Fig. 2: Components of the SP*i* Service Placement Architecture

active on each node which is currently hosting a service instance. It is tasked with collecting usage statistics of the service and adjusting the service configuration when necessary. The *service discovery component* locates the most suitable service instance to provide a service for a client. Furthermore, it proactively announces the location of new service instances that are created whenever the service configuration is changed. The *routing component* implements an enhanced routing protocol that selectively piggy-backs neighborhood and routing path information on data packets. This information is then aggregated and provided to the service placement middleware on demand.

The service placement middleware operates according to the following steps: Local usage statistics are transmitted to a dynamically-assigned control node. This node calculates the optimal service configuration according to the cost function described in the previous section. The cost metric is based on service demand in terms of used bandwidth and the network topology in terms of hop count and link quality. With this service configuration as input, the control node establishes a set of actions required to migrate from the current to the optimal configuration. Possible actions are the *replication* of a service instance from a current host to a new host, the *migration* of a service instance, and *shutting down* an instance. If the combined cost of these actions in terms of network traffic is less than the difference in cost between the current and the optimal configuration, commands for adapting the configuration are issued to the current service hosts. These nodes then distributively proceed with replicating, migrating, or shutting down individual service instances.

A crucial step in this process is the calculation of the optimal service configuration on the coordinator node. Our algorithm is based on the observation that, provided that clients chose the nearest service instance during service discovery, the mapping of clients to service instances induces a clustering of the network. The goal of the algorithm is thus to calculate the optimal clustering, whose set of cluster heads corresponds to $\widehat{H}_s$. This is achieved by initializing the algorithm with a valid but suboptimal configuration of having one service instance per client, i.e., each client node is its own cluster head, and then iteratively merging the two adjacent clusters (with nodes h and h' as cluster heads) that have the lowest combined cost $\sum_{c \in C_{s,h} \cup C_{s,h'}} \delta(s,c)$. After each step of the iteration, we calculate the service cost $c(s)$ for the current set of cluster heads and retain the service configuration with the lowest cost. This algorithm has a run-time proportional to $|N|^4 \times \deg^2$, where deg is the node degree of the network

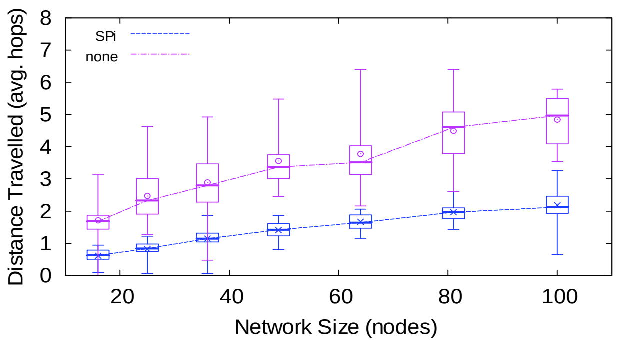


Fig. 3: Hop count of service requests against network size

graph. It terminates for all inputs because the number of clusters decreases with each iteration of the main loop.

4. EVALUATION

In our evaluation, we quantify the advantages of service placement with SP*i* over a traditional approach based on a client/server architecture without service placement. We used version 2.33 of the **ns** network simulator to simulate networks of different sizes (in terms of number of nodes) and under different load scenarios. The nodes were randomly placed in an area whose size was changed depending on the number of nodes in order to achieve a near constant median node degree. The radio link was set to an average transmission radius of 50 m, IEEE 802.11 was used for medium access, and an enhanced version (as detailed above) of DYMO (5) was used for routing. The size of the service as transferred between hosts when a service instance is replicated or migrated was set to 100 kB. The fraction of synchronization traffic τ_s was set to 10%. We observed the network for simulated 15 minutes after an initial 5 minutes for initialization. The initial service host was assigned randomly at the beginning of the simulation. During the initialization phase, half of the nodes began issuing service requests. The results are based on aggregated data from 30 runs of the simulation. Each measurement is plotted as combination of median, mean, first and third interquartile, and minimum and maximum. The medians of related measurements are connected with a dashed line.

As a fundamental effect of service placement, one would expect that the average distance between client nodes and service hosts decreases as a suitable number of service instances is created. We verify this intuition in Figure 3 which shows the distance travelled by service requests against the size of the network. We can observe that service placement with SP*i* reduces the distance that service requests have to cover in order to reach the closest service host. Furthermore, it becomes apparent that the general behavior of the network becomes more predictable as the dependency of the overall performance on the random choice of the initial service host is eliminated.

In Figures 4 to 7, we evaluate the cost of service provisioning and the quality of the service. Figures 4 and 5 plot the overall link layer traffic during the entire simulation against different network sizes and loads. We note that our system with SP*i* service placement consistently, for both increasing network size and load, requires about half the amount of network traffic when compared to a system

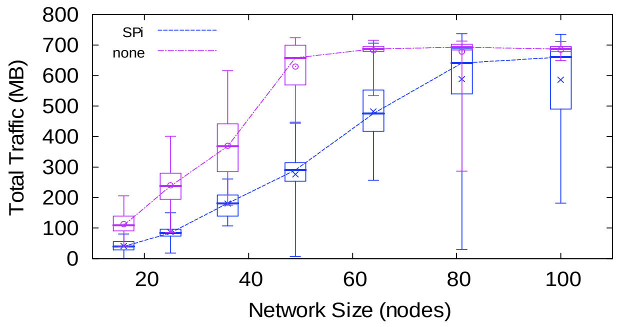


Fig. 4: Overall (link layer) network traffic against network size

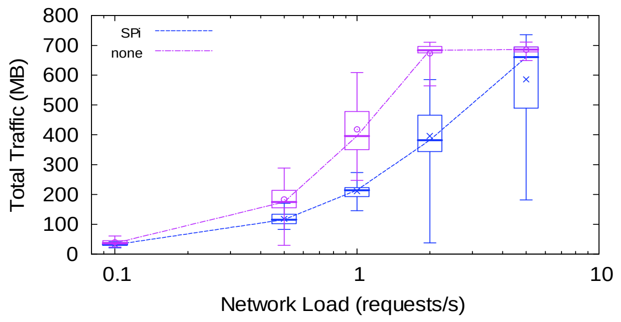


Fig. 5: Overall network (link layer) traffic against network load

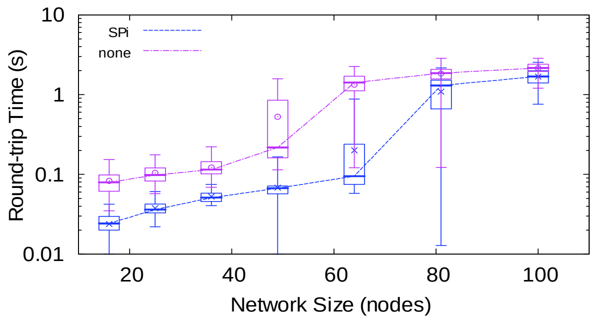


Fig. 6: Round-trip time of service requests against network size

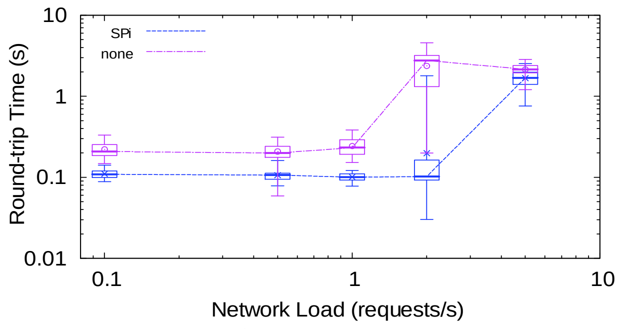


Fig. 7: Round-trip time of service requests against network load

without service placement. As network size or load increase, scenarios without service placement reach a point of saturation at approximately 700 MB, beyond which no additional data can be transferred during the allocated time. As a result, service requests cannot be routed to their destination nodes anymore and the service recall, i.e., the fraction of successful service requests, begins to drop (figures omitted). In comparison, scenarios with SP*i* service placement reach saturation for larger networks or under higher load. In other words, service placement allows deploying more demanding services in larger networks that would otherwise exceed the network capacity.

Figure 6 and 7 plot the round-trip time of service requests for the same scenarios as above. Similarly to the plots in Figures 4 and 5, we observe that our system with SP*i* service placement consistently reduces the round-trip time by about one half. As network size and load increase, the network reaches the limits of its capacity and the round-trip time increases sharply by one order of magnitude. As in the previous part of the evaluation, this transition occurs later for scenarios with SP*i*. Hence, we can conclude that service placement allows sustaining a higher quality of service even in more demanding scenarios.

Combining these two observations, we can sum up our findings in stating that service placement, as implemented in SP*i*, reduces the overall cost of service provisioning (in terms of required network traffic) while at the same time improving the quality of the service (in terms of recall and responsiveness).

5. CONCLUSION

In this paper, we gave a brief introduction to service placement in ad hoc networks with an emphasis on the cost of synchronizing shared data between multiple service instances. We introduced the SP*i* service placement architecture and evaluated the effects of service placement in comparison to a traditional client/server architecture. Our results show that networks with service placement consistently perform better than those without, improving application-level quality metrics while causing less overall network traffic.

Additional to these results, networks employing service placement can also benefit from inherent replication of data which protects against node failures and the effects churn. Scenarios than can potentially draw most benefit from service placement are those in which a strong correlation exists between location of nodes (including group mobility) and demand for a service. Service placement can thus be considered applicable to a wide range of system, ranging from sensor networks and fixed office meshes to vehicular networks.

6. BIBLIOGRAPHY

1. **Herrmann, K.** Self-Organizing Infrastructures for Ambient Services. *PhD thesis.* Berlin, Germany : Technische Universität Berlin, July 2006.

2. **Gossa, J., et al.** Proactive Replica Placement Using Mobility Prediction. *Proc. Workshop on Data Management in Context-Aware Computing (DMCAC '08 / MDM '08).* Beijing, China, April 2008.

3. **Lipphardt, M, et al.** DySSCo - A Protocol for Dynamic Self-organizing Service Coverage. *Proc. Workshop on Self-Organizing Systems (IWSOS '08).* Vienna, Austria, December 2008.

4. **Wittenburg, G and Schiller, J.** A Survey of Current Directions in Service Placement in Mobile Ad-hoc Networks. *Proc. IEEE Conference on Pervasive Computing and Communications (PerCom '08 / PerWare '08).* Hong Kong, March 2008.

5. **Chakeres, I. D. and Perkins, C. E.** Dynamic MANET Ondemand (DYMO) Routing. *IETF Internet-Draft.* March 2009.