Modeling Responsiveness of Decentralized Service Discovery in Wireless Mesh Networks

Andreas Dittrich[1], Björn Lichtblau[2], Rafael Rezende[1], and Miroslaw Malek[1]

[1] Advanced Learning and Research Institute (ALaRI)
Università della Svizzera italiana, Lugano, Switzerland
{andreas.dittrich,ribeiror,malekm}@usi.ch
[2] Humboldt-Universität zu Berlin, Berlin, Germany
lichtbla@informatik.hu-berlin.de

Abstract. In service networks, discovery plays a crucial role as a layer where providers can be published and enumerated. This work focuses on the responsiveness of the discovery layer, the probability to operate successfully within a deadline, even in the presence of faults. It proposes a hierarchy of stochastic models for decentralized discovery and uses it to describe the discovery of a single service using three popular protocols. A methodology to use the model hierarchy in wireless mesh networks is introduced. Given a pair requester and provider, a discovery protocol and a deadline, it generates specific model instances and calculates responsiveness. Furthermore, this paper introduces a new metric, the expected responsiveness distance d_{er}, to estimate the maximum distance from a provider where requesters can still discover it with a required responsiveness. Using monitoring data from the DES testbed at Freie Universität Berlin, it is shown how responsiveness and d_{er} of the protocols change depending on the position of nodes and the link qualities in the network.

Keywords: Real-time systems, Responsiveness, Service discovery, Wireless mesh networks, Markov Models, Probabilistic Breadth-First Search

1 Introduction

Service-oriented architecture (SOA) describes a paradigm where services are the building blocks of system design. SOA introduces several principles to support its paradigm. Among them is discoverability, which means that structured data is added to services to be effectively published, discovered and interpreted. Communication of this data is done by *service discovery* (SD). Using SD, service instances can be enumerated and sorted according to functional and non-functional requirements, facilitating autonomous mechanisms like optimization of service compositions or fall-back to correctly operating instances in case of failure.

If discovery fails, a service cannot be available. Comprehensive service dependability evaluation thus needs to consider the discovery process. This is traditionally neglected, however. Since SD is a time-critical operation, one key dependability property is *responsiveness* – the probability to perform some action on time even in the presence of faults [14]. For SD, responsiveness quantifies the probability that a required ratio of present service instances is found within a deadline. Due to the diversity of usage scenarios and the dynamics of modern networks, it is not trivial to predict. This problem is exemplified in unreliable networks with more complex fault behavior, such as self-organized *wireless mesh networks* (WMNs), where the quality of links is constantly changing and heavily affected by external interference, fading effects and multi-path propagation.

This work provides a hierarchy of stochastic models to evaluate responsiveness of decentralized SD in unreliable networks. It provides a methodology to apply these models to WMNs. Because of a high variability of link quality in such networks, the responsiveness is expected to change significantly with the positions of requester and provider. The methodology thus considers the *user-perceived* responsiveness of given communication partners. It estimates packet loss probabilities and transmission time distributions for each link on the communication paths between the partners and generates specific model instances to assess SD responsiveness. This facilitates the evaluation of responsiveness in common SD scenarios, to provide hints on the suitability of current protocols and detect their shortcomings in WMNs. The provided solution is expected to spur future research on service dependability which includes the discovery layer.

The remainder of this paper is structured as follows. After brief background information and related work in Sections 2 and 3, the problem is described in Section 4. A hierarchy of stochastic models to evaluate SD responsiveness is introduced in Section 5, followed by a methodology that uses these models in Section 6. The case study in Section 7 shows the responsiveness of SD in different scenarios. Results are explained and interpreted. Section 8 concludes the work.

2 Service Discovery

SD is realized with three different architectures. In two-party or decentralized architecture, service clients and providing instances communicate directly with each other. In three-party or centralized architecture, this communication is handled by a registry. Hybrid architectures can switch between these two on demand. SD describes service instances with a unique identifier, optionally a service type and other structured information relevant to a service user. Syntax and semantics of this information are known to discovery clients.

Discoverability requires the ability to both publish a service instance and to discover it. All discovery protocols supply these two basic types of operation. A providing service instance can publish its presence either directly to the network via multi- or broadcast or to a registry, also known as registration. Clients use discovery to enumerate providing instances passively, by listening to *publish* messages or actively, through discovery requests with subsequent responses, if providers are available. Responses are either sent directly from providers or from the registry, via uni- or multicast. The use of multicast generally causes higher load on the network than unicast. However, it may suppress requests from other clients by responding proactively and greatly simplifies distributed cache maintenance. In WMNs, efficient flooding poses a great challenge so using multicast should be considered carefully.

In *Internet Protocol* (IP) networks, three different discovery protocols are prevalent: *Service Location Protocol* (SLP), *Simple Service Discovery Protocol* (SSDP) and *Domain Name System based Service Discovery* (DNS-SD). DNS-SD as part of the Zeroconf protocol family is referred to by that name throughout this paper. Especially, SSDP and Zeroconf can be found in a plethora of embedded devices, such as printers, network-attached storage or cameras. The protocols transmit messages using the lightweight *user datagram protocol* (UDP). UDP is an unreliable transport so recovery operations are done by the SD protocols themselves. Fail-stop faults may be classified as regular exhibition of network dynamics and are recovered by goodbye messages. Crash, omission and timing faults are recovered by request retries and timeouts. The number of retries and the time between them vary among the protocols. Zeroconf and SLP specify an initial retry timeout and then double it every period. In SSDP, the requester may choose a timeout in a specified interval for every period. Values for the individual intervals are shown in Table 1. Quantitative analysis of specific properties to justify these strategies, responsiveness in particular, is practically non existent.

Table 1. Service discovery retry intervals for the studied protocols

	$t_{retry}(1)$	$t_{retry}(2)$	$t_{retry}(3)$	$t_{retry}(4)$	$t_{retry}(5)$
Zeroconf	$1s$	$2s$	$4s$	$8s$	$16s$
SLP	$2s$	$4s$	$8s$	$16s$	$32s$
SSDP (min/max)	$1s/5s$	$1s/5s$	$1s/5s$	$1s/5s$	$1s/5s$

Most widely used for IP networking in WMNs is *Optimized Link State Routing* (OLSR). OLSR nodes proactively search for routes and cooperatively create a spanning tree that covers the whole topology. A number of different metropolitan networks, such as in Athens, Berlin and Leipzig successfully employ OLSR.

3 Related Work

An overview of decentralized discovery protocols can be found in [22] and [11]. Dabrowski et al. did an experiment-based analysis of various dependability properties in existing discovery protocols [6,7,8]. Among them, related to responsiveness is *update effectiveness*, the probability to restore a consistent state after failure. They did not consider active SD responsiveness during regular operation. Furthermore, the widespread Zeroconf protocol is not considered, the responsiveness of which has been evaluated in experiments in [10]. The paper at hand aims to provide analytical methods to reproduce the results in [10].

The automatic generation of steady-state service availability models from service descriptions and infrastructure information is presented in [15]. A related approach with state-of-the-art tool support can be found in [9], which has been extended to support instantaneous availability evaluation in [19]. However, none can be easily adapted for responsiveness evaluation and their complexity is prohibitive in highly connected WMNs. A work inspirational to this paper describes dependability analysis using directed acyclic graphs [20]. The paper at hand combines a network topology and a discovery operation in a Markov model that reflects such a graph. A related model has been used for a cost-estimation of automatic network address assignment in Zeroconf [4].

General problems and challenges in WMNs are presented in [1]. There are several approaches to model packet transmission delays at the 802.11 MAC level, e.g. [17,18], which will not be considered due to their complexity. Bianchi [2] developed a Markov model to compute the 802.11 *Distributed Coordination Function* (DCF) saturation throughput. It assumes a known finite number of terminals, ideal channel conditions and all nodes in one collision domain. These assumptions do not hold for the WMNs targeted in this work. Instead of a detailed modeling of low-level MAC operations, we favor an approach that encompasses application layer protocols. Our delay estimation is based on the *expected transmission count* (ETX) metric [5] used by OLSR with packet transmission delays as defined in the 802.11 standard [12], related to, but more efficient than [16].

4 Problem Statement

A methodology is needed to quantify *user-perceived* responsiveness of decentralized SD in WMNs. The user-perceived scope is defined by the position of requester and provider and the time of discovery. The methodology needs to use a stochastic model to evaluate responsiveness and an automated procedure that covers the following steps: (1) Define SD scenario that contains requester and provider, protocol and deadline for the SD operation. (2) Gather monitoring data from the network and prepare that data as input parameters of the model. (3) Instantiate specific models using these parameters and the scenario definition. (4) Evaluate user-perceived responsiveness by solving these model instances.

The methodology should support evaluation of three different variants of SD responsiveness. First, the responsiveness for different requester-provider pairs, second, the average responsiveness of a specific provider for all requesters in the network. Third, a novel metric, the *expected responsiveness distance* should be investigated, to estimate the maximum distance from a provider where requesters are expected to discover it with a required responsiveness (see Definition 1). This work focuses on IP networks and their most common discovery protocols: Zeroconf, SSDP and SLP. Routing is done by the prevalent OLSR protocol.

Definition 1. *Given a* service discovery *deadline t_D with a required responsiveness $R_{req}(t_D)$, a set of service providers S and sets of clients $C_d, d \in \mathbb{N}^+$ with d denoting the minimum hop distance of each client in C_d from all providers in S. Let $R_{avg,d}(t_D)$ be the average responsiveness when discovering S from C_d. The expected responsiveness distance d_{er} is the maximum d where $R_{avg,d}(t_D) \geq R_{req}(t_D)$ and $\forall d' \in \mathbb{N}^+, d' < d : R_{avg,d'}(t_D) \geq R_{req}(t_D)$.*

5 Modeling Service Discovery

When doing SD, the number of requesters and providers may vary. For instance, multiple clients might request a single service. One client could discover all existing providers in the network to choose one meeting best its requirements. We will now focus on decentralized SD by a single client. Any such SD operation can be described with a generic family of Markov models, which includes three types of states: (1) A single state named req_0 defines the beginning of an SD operation, when the initial request has been sent. (2) Two absorbing states *ok* and *error* define the successful or unsuccessful end of an SD operation. (3) A set containing every state between the first two types where not all required responses have been received and the final deadline has not been reached.

The Markov model family is parametric in two parameters: number of retries and required coverage. The maximum number of retries n describes the first dimension of the model family. Beginning from req_0, it defines a chain of retry states $req_i,\ i = 1...n$, that stand for "retry i has been sent". From every retry state the model can transition into *ok* in case a sufficient number of responses was gathered. If not it will transition to the next retry state and eventually from req_n to *error*. The parameter required coverage describes how many responses need to be received before an SD operation is called successful and is the second dimension of the model family. The retry states $req_i,\ i = 1...n$ become a set of states that relate to the success ratio when doing retry i. The size of this set can be arbitrary but for example, if three services need to be found for successful operation, there could be three states $req_{ij},\ j = 0, 1, 2$ for every retry i that stand for "retry i has been sent and j responses have been received so far".

Estimating the transition probabilities within this Markov model family is not trivial. In the following, we propose a hierarchy of stochastic models where the probabilities of these high level discovery models are calculated by low level models based on link quality data measured in the network.

5.1 Service Discovery Model

To demonstrate the model hierarchy, we will now instantiate a specific model for discovery of a single service within a deadline $t_D = 5s$ using the Zeroconf protocol. The number of retries n can be derived by examining the retry strategy of the discovery protocol under analysis (see Table 1). Given the times in seconds $t_{retry}(i), i \in \mathbb{N}^+$ between retries $i-1$ and i with $t_{retry}(0) = 0$. The total time $t_{total}(r)$ after the beginning of a discovery operation when sending retry r is calculated according to Equation 1.

$$t_{total}(0) = 0, \quad t_{total}(r) = \sum_{i=1}^{r} t_{retry}(i) \quad , r \in \mathbb{N}^+ \tag{1}$$

For Zeroconf, $t_{total}(2) < t_D < t_{total}(3)$. So, $n = 2$ retries will be sent. The resulting regular Markov model instance is depicted in Figure 1: In short, retries continue to be sent until a response is received, triggering a transition to the *ok* state. If no response is received after retry n has been sent and before t_D, the operation is considered failed by transitioning to state *error*. So, the discovery operation is successful as soon as the first response packet arrives at the requester. Transitions between the retry states will only happen if no response has been received until the specific retry timeout.

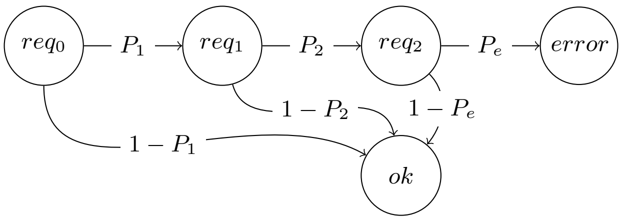


Fig. 1. Markov chain for single service discovery

There are two related probabilities in Figure 1: $P_r, 1 \leq r < n$ is the probability that no discovery response was received between $t_{total}(r) - t_{retry}(r)$ and $t_{total}(r)$. P_e is the probability that no response was received between $t_{total}(n)$ and t_D. Arrival times of responses to a specific request r can be considered as a random variable X_r. Equation 2 describes the cumulative distribution of this variable, the probability that a response to request r has arrived by time t or, the *responsiveness* $R_r(t)$ of a single request-response operation for request r.

$$F_{X_r}(t) = P\{X_r \leq t\} = R_r(t) \tag{2}$$

Knowing this, Equation 3 calculates the probability that a response to request r arrives in a specific time interval.

$$P\{t_x \leq X_r \leq t_y\} = R_r(t_y) - R_r(t_x) \tag{3}$$

Functions $Pr : \mathbb{N}^+ \to [0, 1]$ and $Pe : \mathbb{N}^+ \to [0, 1]$, as defined in Equations 4 and 5, can now calculate P_r and P_e such that $Pr(r) = P_r$ and $Pe(n) = P_e$.

$$Pr(r) = \prod_{i=1}^{r} \left(1 - \frac{R_i(t_{total}(r)) - R_i(t_{total}(r-1))}{1 - R_i(t_{total}(r-1))}\right) \tag{4}$$

$$Pe(n) = \prod_{i=1}^{n} \left(1 - \frac{R_i(t_D) - R_i(t_{total}(n))}{1 - R_i(t_{total}(n))}\right) \tag{5}$$

Since Equation 5 is a special case of Equation 4, only Equation 4 is explained in detail. A retry is forced when no response packet arrived until the retry timeout, so the product multiplies the individual probabilities for non-arrival of responses to each request that has been sent so far. The probability for the response to request i to arrive *within the specific interval* is described by the quotient. The numerator describes the unconditional probability for a response to arrive within the specified interval. But, deducting from the structure of the Markov model, it cannot have arrived before. That condition is given in the denominator. The quotient, thus, gives the probability that a response to request i is received in the specified time interval, provided it has not arrived before. It is then subtracted from 1 to get the probability of non-arrival.

Missing is a specification to calculate the functions $R_r(t)$. One way would be to measure response times of request-response pairs and fit a distribution to them. We provide an analytical solution instead, using a retry operation model.

5.2 Retry Operation Model

When discovering a single service, each retry step relates to a request-response pair, described by the semi-Markov process in Figure 2. In state *Rq*, a request has been sent. When it arrives at the destination, the provider will send a response, triggering a transition to state *Rp*. As soon as this response arrives back at the requester, the model enters state *ok*. If one of the messages gets lost, it will transition to state *error*.

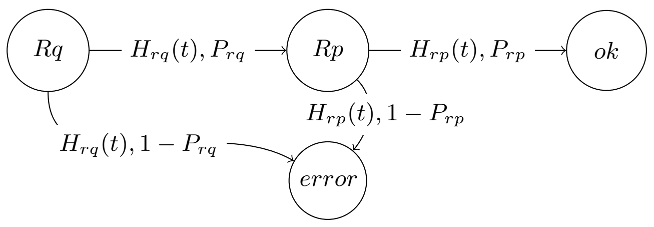


Fig. 2. Semi-Markov chain for a single request-response pair

In case messages arrive (with a probability of P_x), they have a certain distribution of arrival times. This is described by $H_x(t)$, the sojourn time distribution for state x. The cumulative distribution function of time to absorption in state *ok* now calculates $R_r(t)$. Since t is relative to the beginning of the SD operation, $R_r(t)$ is in fact parametrized by the location $t_{total}(r)$, the time at which retry r is being initiated. The retry operation model is independent of a concrete network infrastructure. It has no knowledge of how to calculate the probabilities and transition time distributions. Providing concrete values of P_x and $H_x(t)$ for specific SD pairs on demand is the purpose of the network mapping model.

5.3 Network Mapping Model

Mapping requests and responses to the network under analysis means providing models that calculate P_x and $H_x(t)$ in the retry operation model (see Figure 2) by taking into account the details of the used communication mechanism, unicast or multicast. This mapping is dependent on the concrete network infrastructure. In this case, we provide models for a WMN running OLSR. Different networks could need diverse models, but, provided they estimate P_x and $H_x(t)$, these could be used in the proposed model hierarchy as well.

Unicast Model A unicast message follows the shortest path according to the routing metric. In OLSR, every node periodically calculates that shortest path and saves the next hop for every destination. A unicast message is sent to the next hop node which then decides to forward according to its own next hop information for the destination. We use an algorithm that calculates the unicast path hop by hop based on the next hop information on each node. If no such global information is available, using the shortest path known to the first node remains a viable, albeit less accurate solution.

Since there is only one path with n nodes and $m = n - 1$ links, this can be modeled as a simple semi-Markov chain of n states. Each state $k_i, i = 1...n - 1$ stands for "message forwarded by node i", state k_n means "message arrived at node n". The links between nodes i and $i + 1$ become transitions $k_i \rightarrow k_{i+1}$. Further, there is a transition from $k_i, i = 1...n-1$ to $error$ to account for packet loss. State transition probabilities $P_{k_i,k_{i+1}}$ are calculated from the currently monitored packet transmission probabilities of the link between nodes i and $i+1$ (see Section 6), taking into account that unicasts will be retransmitted up to seven times if not acknowledged by the receiving node. The estimation of sojourn time distributions $H_{k_i,k_{i+1}}(t)$ is described in Section 5.4. The resulting unicast chain is then integrated into the retry operation model in Figure 2 – for a unicast response, for example, by merging states k_i and *Rp*, k_n and *ok* as well as the two error states. The rest of the chain replaces transition *Rp* → *ok*.

Multicast Model In theory, modeling the traversal of a multicast discovery packet should consider all possible paths between source and destination. This redundancy has been taken into account in [9] but finding all paths between two nodes is NP-complete, a prohibitive complexity especially in networks with high connectivity, such as WMNs. Since the vast majority of those paths has a very low probability of traversal and their impact on the responsiveness of the multicast communication would be minor, this work instead uses *probabilistic breadth-first search* (PBFS) [13] to derive an estimation of the multicast path length. In PBFS, node neighbors are only considered if the edge between a node and its neighbor succeeds a random roll against its transmission probability, as monitored by the routing layer (see Section 6). This way, each run of PBFS realistically simulates how a multicast packet would traverse the WMN. PBFS is sampled a sufficient number of times to approximate with which probability the destination node could be reached. This reflects P_x in the retry operation model, for example, P_{rq} for a multicast request. We additionally store the probability for each path length in case of arrival to later estimate the distribution of sojourn time $H_x(t)$ in Section 5.4.

5.4 Transmission Time Distributions

Estimations for sojourn time distributions in the network mapping models are based on the (re-)transmission and potential back-off periods defined in the 802.11 standard [12]. For transmission times over links, we assume the lowest data rate, which is correct for multicasts. Unicasts, however, will transmit at higher data rates if possible, reducing the time for individual retry transmissions. The estimation thus presents an upper bound for the transmission time as dependent on the data rate. The estimation also ignores additional contention due to internal traffic or external interference, which affects the upper limit of transmission times. To account for this, a certain percentage of packets is assumed to arrive *after* the estimated maximum transmission time for both uni- and multicasts. The bounds calculated from these assumptions are fitted to an exponential distribution for the transmission time. For the unicast model, this is done for each transition $k_i \rightarrow k_{i+1}, i = 1...n-1$ and provides $H_{k_i,k_{i+1}}(t)$. In the multicast model, one distribution function is generated for each possible path length given by PBFS. The distributions are then weighted with the corresponding probability for their length and combined in a single function $H_x(t)$.

6 Methodology

To calculate the SD responsiveness for given pairs of requester and provider in a network, the model layers described in Section 5 need to be generated bottom-up using the following steps:

1. Define a scenario which consists of (1) the SD communication partners requester and provider, (2) the discovery protocol and (3) a deadline for the SD operation.
2. Generate low level network mapping models for individual requests and responses between the SD pair requester and provider based on the communication mechanisms of the protocol, uni- or multicast (see Sections 5.3, 5.4).
3. Integrate the network mapping models from Step 2 in the semi-Markov chain for the retry model (see Section 5.2). This chain calculates the responsiveness of an individual retry over time.
4. Calculate the number of retries n based on the defined protocol and deadline. This defines the structure of the high level discovery model (see Section 5.1).
5. Estimate the state transitions probabilities in the discovery model, using Equations 4 and 5. In these equations, $R_n(t)$ is the cumulative probability for absorption at time t in state *ok* in the retry model from Step 3.

The discovery model can then be solved. The steady-state probability of arrival in state *ok* in this model is the probability that an SD operation as specified in the scenario is successful, given the current monitored state of the network. The methodology has been implemented in a Python framework that carries out all necessary steps. More complex stochastic analysis is performed using the SHARPE tool [21].

All monitoring data is gathered on demand from the routing layer. This approach is least invasive and can be used in every network where the routing layer provides the needed data. OLSR nodes use probe messages to measure link qualities for every neighbor. Given the forward delivery ratio d_f and reverse delivery ratio d_r, ETX is defined as the reciprocal of $(d_f \cdot d_r)$. This information allows to construct a complete network graph with edges weighted by their ETX value. In the graph, nodes are annotated with meta information from their local OLSR routing table that includes the next hop for every other reachable node in the network.

7 Case Study

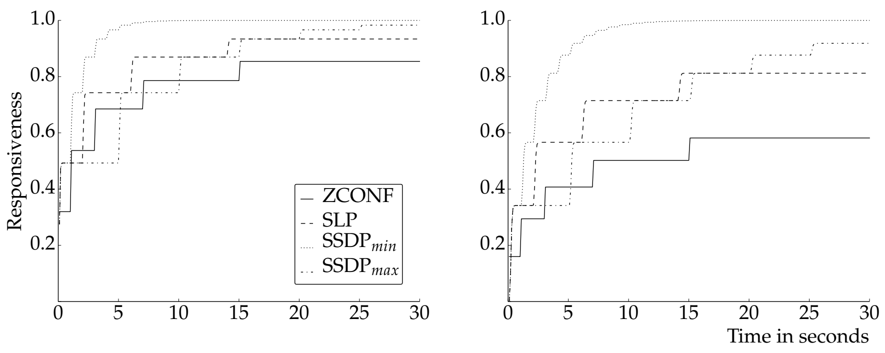
To demonstrate how the proposed methodology can be used to estimate the responsiveness in common use cases of SD, the three protocols explained in Section 2 are now evaluated in three different scenarios using measured data from a real-life WMN, the *distributed embedded systems* (DES) testbed at *Freie Universität Berlin* (FUB). This testbed consists of around 130 wireless nodes that are spread over multiple campus buildings of FUB. Due to space limitations, we refer to a complete description of the testbed in [3]. For the sake of traceability, node identifiers in this text reflect the actual hostnames in the testbed.

In this case study, OLSR was used in version 0.6.5.2. It provides a valid reference for the real world application of the methodology. All monitoring was done by OLSR. Topology data was gathered with OLSR's JSON plug-in and then integrated into the network model using the Python framework. The testbed was configured and data gathered with different transmission power levels to obtain different topologies. Retry intervals of the discovery protocols are set according to the standards as described in Section 2. Since SSDP does not have fixed intervals, it is assessed in two different configurations reflecting the minimum and maximum interval as defined in the standard.

7.1 Scenario 1 – Single Pair Responsiveness

First, the responsiveness of a single pair requester and provider is evaluated over time. In order to investigate also how the responsiveness changes with the distance between nodes, two different pairs were chosen. One pair *(t9-105, t9-154)* is within the main cloud *t9*, a dense and well-connected part of the WMN consisting of 56 nodes (see Figure 3a). The other pair *(t9-105, a3-119)* covers almost the maximum distance in the network (see Figure 3b). In both cases, node *t9-105* is the requester. The results clearly show that as the distance between requester and provider increases, overall responsiveness decreases.

The difference in responsiveness among the protocols is apparent. With increasing deadlines the responsiveness of the Zeroconf protocol is consistently lower compared to the other protocols. This is because Zeroconf uses multicast for both requests and responses. Multicast packets will not be resent seven times before considered lost, so the danger of packet loss is much higher. The positive



(a) Within same cloud, provider *t9-154* (b) Maximum distance, provider *a3-119*

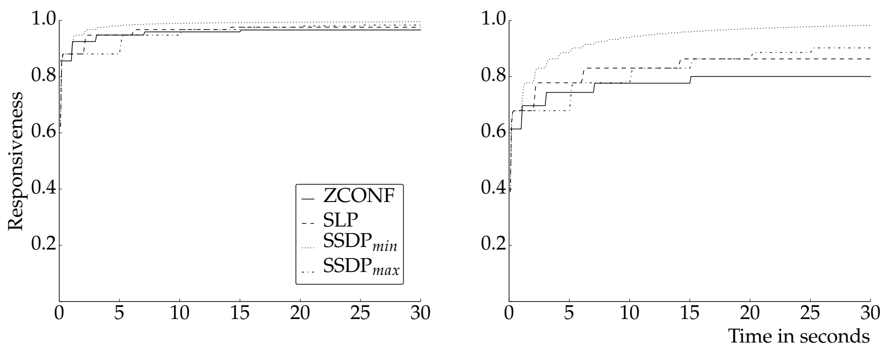
Fig. 3. Discovery responsiveness over time for different providers requested from *t9-105*

effects of multicast responses for multiple communication pairs, as pointed out in Section 2, cannot be considered in this analysis. Also not included are the effects of additional load on the network caused by discovery. Since retries are considered independent events, lower retry intervals will lead to a higher responsiveness. While this assumption can be justified for retry intervals in the order of seconds – discovery packets are only a few bytes in size – it cannot hold for ever-lower intervals. So, although SSDP with a minimum interval ranks consistently best, the increased load might not be in the best interest of the service network as a whole. More in-depth research is needed on that matter. However, it can be deducted that with low deadlines, the chosen retry interval is more relevant for responsiveness than the communication mechanism (i.e., unicast vs. multicast). In general, current SD protocols struggle to achieve a high responsiveness in WMNs, even over short distances.

7.2 Scenario 2 – Average Provider Responsiveness

The second scenario covers the average responsiveness of a single provider over time when requested from an arbitrary client in the network. To demonstrate how the models capture topology changes, this scenario uses data measured in two different topologies that were generated with different radio power settings. The focus lies on provider *t9-154* from Section 7.1, which is well centered within the network so it provides a good reference to see the effects of overall link quality on responsiveness. Figure 4 shows the results.

The main observation is that the average responsiveness when discovering node *t9-154* is quite high due to its prominent, almost optimal position in the network. With high quality radio links, depicted in Figure 4a, all protocols quickly reach a responsiveness of over 90%. Responsiveness is considerably decreased for lower quality wireless connections (see Figure 4b). With deadlines above 15 seconds, there is a consistent ranking of the discovery protocols, with



(a) High radio power (b) Low radio power

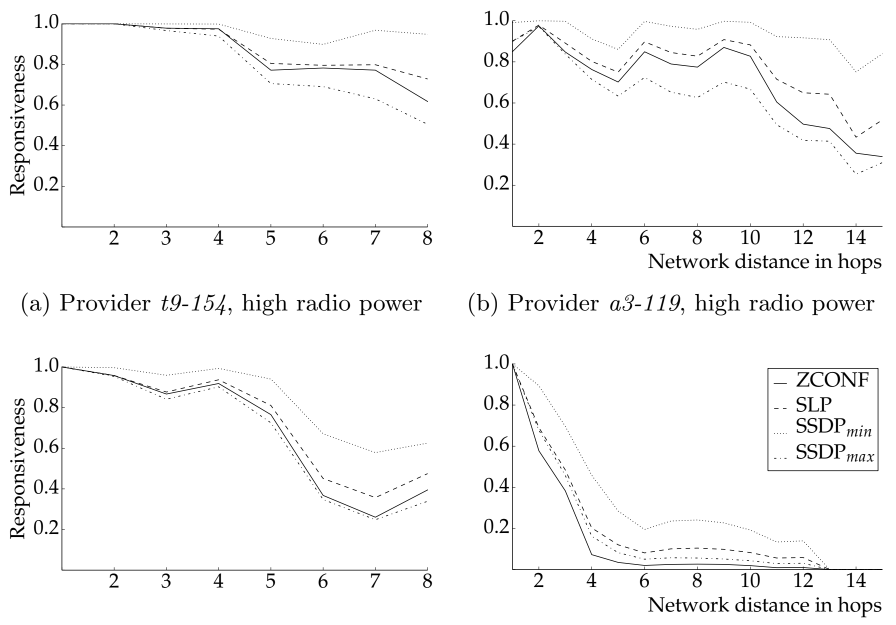
Fig. 4. Average responsiveness over time for provider *t9-154* in different topologies

Zeroconf again having the lowest responsiveness. The ranking is identical for different link qualities, only the overall values are different. Due to different retry strategies of the protocols, however, this behavior is not consistent for lower deadlines. This underlines the findings from the Scenario 1: With deadlines close to the individual retry intervals, the chosen interval is more relevant for responsiveness than the communication mechanism. In summary, it can be said that purely multicast based SD as in Zeroconf is justified when positive effects for multiple communication partners are expected. For single discovery operations among few partners, responding via unicast like in SSDP and SLP provides higher responsiveness because of its more reliable communication mechanism. Among SSDP and SLP, the specific retry strategy until the deadline is the main factor impacting responsiveness.

7.3 Scenario 3 – Expected Responsiveness Distance

The last scenario covers the *expected responsiveness distance* d_{er} from Definition 1. The responsiveness of two different providers, *t9-154* and *a3-119*, is calculated when requested from every client in the network. Then, the responsiveness is averaged for requesters at the same distance of these providers. Again, the used data was measured in two different topologies that were generated with different radio power settings. The discovery deadline is set to five seconds. Results are illustrated in Figure 5.

The ranking among the protocols is not the same as in the previous scenarios. This is due to the chosen, realistically short deadline of five seconds. The retry strategy until this deadline has an important impact and the maximum retry timeout for SSDP simply did not force enough retries to account for lost messages. It can also be recognized in Figure 5d, that badly placed providers risk a very low d_{er} with decreasing link quality. The d_{er} for the different protocols with a required responsiveness $R_{req} = 0.8$ is summarized in Table 2. It should



(a) Provider *t9-154*, high radio power (b) Provider *a3-119*, high radio power

(c) Provider *t9-154*, low radio power (d) Provider *a3-119*, low radio power

Fig. 5. Average responsiveness over number of hops for providers *t9-154* and *a3-119* in two different topologies.

be noted that the maximum d_{er} depends on the eccentricity ϵ of the provider node, the greatest distance from any other node.

As can be seen in Figure 5, the average responsiveness is not always decreasing over distance. This happens because hop count as the chosen distance metric does not necessarily reflect the quality of a path. In fact, longer paths might be of higher quality. The hop distance is, however, an intuitive metric that in this case presents the lower bound for d_{er}. If needed, a more realistic, quality-based distance metric should be used to increase accuracy.

Table 2. Expected responsiveness distance d_{er} of the studied protocols with a deadline of five seconds (ϵ = provider eccentricity, R_{req} = required responsiveness, RPS = radio power setting). A higher d_{er} is generally desired.

Provider	ϵ	R_{req}	RPS	Zeroconf	SLP	SSDP (min)	SSDP (max)
t9-154	8	0.8	high	4	5	8	4
			low	4	5	5	4
a3-119	15	0.8	high	3	4	13	3
			low	1	1	2	1

8 Conclusion

Dependability evaluation of *service discovery* (SD) in dynamic and decentralized networks remains challenging. This work proposes a stochastic model family to evaluate the user-perceived responsiveness of SD, the probability to find providers within a deadline, even in the presence of faults. The family consists of a hierarchy of Markov and semi-Markov processes that are parametrized to allow instantiation for diverse SD scenarios and use current network monitoring data as input. To put the models into use, a methodology has been introduced that works specifically in wireless mesh networks with proactive routing. Upon request, it generates and solves model instances for specific SD scenarios.

Using data from the DES testbed at Freie Universität Berlin, responsiveness was evaluated for the three most prevalent SD protocols in IP networks. First, the responsiveness for different pairs of requester and provider has been compared. Second, the average responsiveness of a single provider, depending on the topology, has been analyzed. Results demonstrate that responsiveness varies dramatically depending on the position of nodes in the network and the overall link quality. The results further indicate that, with short deadlines close to the individual retry intervals, the right retry timing strategy is more important than the communication mechanism. With longer deadlines, using the more reliable unicast instead of multicast consistently improves responsiveness. In either case, the fixed strategies of current SD protocols struggle to achieve a high responsiveness in these dynamic and inherently unreliable networks. Finally, a new metric *expected responsiveness distance* d_{er} has been introduced, estimating the maximum hop distance from a provider at which nodes can discover it with a required responsiveness. To deploy a responsive service with a minimum number of nodes, every requester in the network should be within the d_{er} of at least one provider. The d_{er} of two different providers has been evaluated and the results underline the importance of position when placing service instances.

Future work will include a comprehensive experimental validation of the model, also in centralized and hybrid SD architectures. The model could then be used to develop novel discovery protocols that, for example, support variable retry intervals depending on the state of the network. Finally, the model facilitates comprehensive service availability evaluation that includes also the discovery layer.

References

1. Akyildiz, I.F., Wang, X.: A survey on wireless mesh networks. IEEE Communications Magazine 43(9), S23–S30 (2005)
2. Bianchi, G.: Performance analysis of the IEEE 802.11 distributed coordination function. IEEE Journal on Selected Areas in Communications 18(3), 535–547 (2000)
3. Blywis, B., Günes, M., Juraschek, F., Hahm, O.: Properties and topology of the DES-testbed. Tech. Rep. TR-B-11-02, FU Berlin, Germany (March 2011)

4. Bohnenkamp, H., van der Stok, P., Hermanns, H., Vaandrager, F.: Cost-optimization of the IPv4 zeroconf protocol. In: International Conference on Dependable Systems and Networks. pp. 531–540. IEEE (June 2003)
5. Couto, D.S.J.D., Aguayo, D., Bicket, J., Morris, R.: A high-throughput path metric for multi-hop wireless routing. In: 9th annual international conference on Mobile computing and networking. pp. 134–146. ACM (2003)
6. Dabrowski, C.E., Mills, K.L.: Understanding self-healing in service-discovery systems. In: Workshop on Self-healing systems (WOSS). pp. 15–20. ACM (2002)
7. Dabrowski, C.E., Mills, K.L., Elder, J.: Understanding consistency maintenance in service discovery architectures during communication failure. In: 3rd International Workshop on Software and Performance (WOSP). pp. 168–178. ACM (2002)
8. Dabrowski, C.E., Mills, K.L., Elder, J.: Understanding consistency maintenance in service discovery architectures in response to message loss. In: 4th Annual International Workshop on Active Middleware Services. pp. 51–60 (2002)
9. Dittrich, A., Kaitovic, I., Murillo, C., Rezende, R.: A model for evaluation of user-perceived service properties. In: International Symposium on Parallel Distributed Processing, Workshops and Phd Forum (IPDPSW). pp. 1508–1517. IEEE (May 2013)
10. Dittrich, A., Salfner, F.: Experimental responsiveness evaluation of decentralized service discovery. In: International Symposium on Parallel Distributed Processing, Workshops and Phd Forum (IPDPSW). pp. 1–7. IEEE (April 2010)
11. Edwards, W.K.: Discovery systems in ubiquitous computing. IEEE Pervasive Computing 5(2), 70–77 (2006)
12. IEEE Standards Association: IEEE standard for information technology – telecommunications and information exchange between systems – [. . .]. IEEE Std 802.11-2012 pp. 1–2793 (March 2012)
13. Lichtblau, B., Dittrich, A.: Probabilistic breadth-first search – a method for evaluation of network-wide broadcast protocols. In: International Conference on New Technologies, Mobility and Security (NTMS). IEEE (April 2014), to appear
14. Malek, M.: Responsive systems: A marriage between real time and fault tolerance. In: Cin, M.D., Hohl, W. (eds.) Fault-Tolerant Computing Systems, Informatik-Fachberichte, vol. 283, pp. 1–17. Springer (1991)
15. Milanovic, N., Milic, B.: Automatic generation of service availability models. IEEE Trans. on Services Computing 4(1), 56–69 (2011)
16. Naimi, A.M., Jacquet, P.: One-hop delay estimation in 802.11 ad hoc networks using the OLSR protocol. Tech. Rep. RR-5327, INRIA, Le Chesnay, France (2004)
17. Oliveira, R., Bernardo, L., Pinto, P.: Modelling delay on IEEE 802.11 MAC protocol for unicast and broadcast nonsaturated traffic. In: Wireless Communications and Networking Conference (WCNC). pp. 463–467. IEEE (March 2007)
18. Raptis, P., Vitsas, V., Paparrizos, K.: Packet delay metrics for IEEE 802.11 distributed coordination function. Mobile Networks and Applications 14(6), 772–781 (2009)
19. Rezende, R., Dittrich, A., Malek, M.: User-perceived instantaneous service availability evaluation. In: Pacific Rim International Symposium on Dependable Computing (PRDC). pp. 273–282. IEEE (December 2013)
20. Sahner, R.A., Trivedi, K.S.: Performance and reliability analysis using directed acyclic graphs. IEEE Trans. on Software Engineering SE-13(10), 1105–1114 (1987)
21. Trivedi, K.S.: SHARPE (symbolic hierarchical automated reliability and performance evaluator). Software (February 2010), `http://sharpe.pratt.duke.edu`
22. Zhu, F., W. Mutka, M., M. Ni, L.: Service discovery in pervasive computing environments. IEEE Pervasive Computing 4, 81–90 (2005)