Technical Report
KN–2015–DiSy–001

Distributed Systems Laboratory

Versatile and Efficient Multi-Link DNS Service Discovery

Daniel Kaiser, Marcel Waldvogel, Holger Strittmatter
Distributed Systems Laboratory
Department of Computer and Information Science
University of Konstanz – Germany

Oliver Haase
Computer Science Department
University of Applied Sciences Konstanz – Germany

Abstract. DNS Service Discovery over Multicast DNS (DNS-SD/mDNS) is used widely for configurationless service discovery in local networks; due in no small part to the fact that it is based on the well established DNS, and efficient in small networks.
In our research, we address three shortcomings of DNS-SD: (1) adding easy-to-use privacy; (2) making it more efficient, especially for large networks; (3) increasing the versatility by enabling DNS-SD to cross broadcast domains; and (4) providing user control through scopes instead of network boundaries within an organization; while maintaining the deployment simplicity and — where it makes sense — backward compatibility.
In this report, we focus on providing multi-link capabilities and scalable scopes for DNS-SD. We propose DNS-SD over Stateless DNS, a solution that allows configurationless service discovery in arbitrary self-named scopes without multicast by leveraging Stateless DNS and the Raft consensus algorithm.

Table of Contents

1 Introduction

In today's local networks zero configuration service discovery is omnipresent. A widely used zero configuration service discovery solution is DNS Service Discovery [1] over Multicast DNS [2] (DNS-SD/mDNS). It allows users to detect printers and streaming devices, to share data, and to communicate with others in a very convenient way. A particular benefit is that services can be requested and offered using DNS resource records, leveraging the solid and well established DNS; thus all means of offering and requesting DNS records can also be used for service discovery. DNS based service discovery cannot only be used in local networks leveraging mDNS but also – losing the zero configuration property – in the Internet using standard DNS servers.

While the current means of distribution for DNS-SD provide good solutions for the scope of single-link local networks and the global Internet scope[1], local DNS based service discovery in multi-link networks – e.g. used in universities or other institutions – is unsatisfactory. Since multicast packets are not propagated across routers, devices in different subnets cannot exchange service information using mDNS. Even if the routers would propagate the messages, multicast based solutions would not scale. For reasons of efficiency, many institutions deactivate multicast in their WiFi network, further limiting the operational area. Using DNS-SD/DNS would pose an unacceptable configuration overhead and would not scale if every user was allowed to offer services. It would only cover the use-case of the institution offering services to its members.

Demand and motivation for an efficient and convenient service discovery solution compatible with DNS-SD are also expressed in an RFC [3]; this RFC states requirements that should be fulfilled, among them a zero configuration mode of operation for multi-link networks. Preceding the RFC, a petition[2] further expressing popular demand for providing such a service discovery solution has been published.

We propose a technique providing versatile configurationless and low configuration modes of DNS-SD operation for multi-link networks. It works by leveraging Stateless DNS [4] to enter NS resource records in the institution's DNS caching server delegating a special service domain to hosts within the institution's network. These hosts are responsible for the actual service discovery. Stateless DNS uses the existing DNS infrastructure; the only additional entity needed is a simple reflector implementable in a few lines of Perl (see appendix) that can run on any server independent of the current institution.

In section 5 we show the bigger picture integrating DNS-SD over Stateless DNS with our service discovery framework that provides flexible, efficient, user friendly, and privacy preserving DNS-SD supporting scalable scopes.

2 Related Work

Much research has been done in the field of service discovery, especially for ad hoc networks [5]. In this report we focus on related work in the field of DNS based service discovery, because these approaches work with existing infrastructure and

[1] For the use-case of one site offering services to its users.

[2] https://www.change.org/p/from-educause-higher-ed-wireless-networking-admin-group

are either backwards compatible to standard solutions or can at least be easily deployed.

2.1 Scalable Multicast DNS-SD in Low-Power Networks

Since multicast causes significant network load in wireless networks [6], techniques that make DNS-SD/mDNS more scalable – especially in 6LoWPAN [7] networks – have been developed.

Klauck et al. present and analyze their DNS-SD/mDNS implementation for low-power devices in [8] and propose methods to compress DNS messages to make DNS-SD/mDNS more suitable for 6LoWPAN networks in [9].

EADP [10] is a protocol for scalable service distribution in 6LoWPAN Networks; it has been leveraged as means of distribution for DNS-SD in [11] (DNS-SD/EADP).

Since these solutions are designed for 6LoWPAN and also depend on multicast, they do not meet the requirements for larger multi-link home or institutional networks. But they can be incorporated in a DNS-SD framework offering appropriate means of resource record distribution depending on the network and on the capabilities of the client.

2.2 Centralized DNS Related Service Discovery Solutions

There are also DNS related service discovery methods, SkyDNS[3] and Consul[4], using multiple central directory services. For our use-case, they are not suitable because like DNS-SD/DNS they demand setup and maintenance. Nevertheless, they might be the solution of choice for an institution willing to maintain the service directory, because they offer a zero configuration experience for the user, if the directory information is propagated using DHCP.

SkyDNS uses etcd[5] as back end, which is a distributed key-value store also leveraging Raft [12] to maintain consistency. Services are announced by sending the service information with JSON over HTTP to the underlying etcd. Service instances can then be queried using DNS queries.

Consul uses an other synchronization algorithm and is also suitable for distributed computing centers.

3 Requirements

RFC 7558 [3] defines requirements that should be met by a scalable service discovery solution. It summarizes the requirements, desiring "[...] a mechanism [...] that populates the DNS namespace with the appropriate DNS-SD records with less manual administration than is typically needed for a conventional unicast DNS server." Our solution – DNS-SD over Stateless DNS – offers precisely that.

In this report we focus on the requirements for multi-link home and institutional networks. For low-power and lossy networks, we propose to use the solutions mentioned in section 2.1. For single-link home networks the widely used DNS-SD/mDNS is well suited, because it poses no significant impact on

[3] https://github.com/skynetservices/skydns
[4] https://www.consul.io
[5] https://github.com/coreos/etcd

the network load [13]. DNS-SD/DNS or solutions presented in section 2.2 are suitable for global scope service discovery. In section 5 we give a short introduction to our service discovery framework that uses different techniques – among them DNS-SD over Stateless DNS – to cover further requirements stated in RFC 7558 and also protects the user's privacy.

4 Scope Name Server (SNS)

To resolve services without or with minimal configuration in multi-link home or institutional networks where multicast does not work across links we introduce Scope Name Servers (SNS) that act as service information directory. Any host in the network can become SNS following rules explained later in this section. A group of SNSes is responsible for one service discovery scope that in turn is defined by a *service discovery domain*, e.g. `ml.ssdisc.com`[6] for services in the whole current multi-link network or `floor3.buildingB.ssdisc.com` for services that should only be resolvable on the third floor in building B.

To make SNSes available to other hosts in the network, we enter them as authoritative name servers for the service discovery domain in the cache of the current networks' DNS cache leveraging Stateless DNS [4]. In doing so it is possible to integrate the service querying process seamlessly into DNS such that the query process is exactly the same as when using DNS-SD/DNS.

A very important design goal is allowing a zero configuration mode of operation for both querying clients and SNSes including the process of becoming SNS. The questions how to discover and become SNS, how to synchronize a set of SNSes, and how to query for and register service information are addressed in the following subsections.

4.1 Discovering SNSes

Since an SNS can enter and leave a network, we need a very dynamic means of providing SNS information via the DNS. Further, SNSes should only be discovered within the institution they currently stay in. To this end, we use Stateless DNS [4] to enter NS records that delegate a service discovery domain to the SNS in the cache of the local DNS server. This allows to have location dependent SNSes for a single service discovery domain; depending on the network a host is currently discovering in, different resource records can be retrieved using the same global service discovery domain.

4.1.1 Service Discovery Domains The combination of service discovery domain and institution defines a service discovery scope. This brings the advantage to being able to use our service discovery method in multi-link networks that are not even aware of the existence of service discovery domains.

We propose three possibilities for clients to learn about service discovery domains: preset discovery domain, DNS-SD/DNS, and DHCP. The default service discovery domain to use can be preset e.g. to `ssdisc.com` so that clients work without any additional configuration. Further service discovery domains that can be provided via DNS-SD/DNS e.g. on `dom.ssdisc.com`. Clients could

[6] ml → multi-link; ssdisc → scoped service discovery

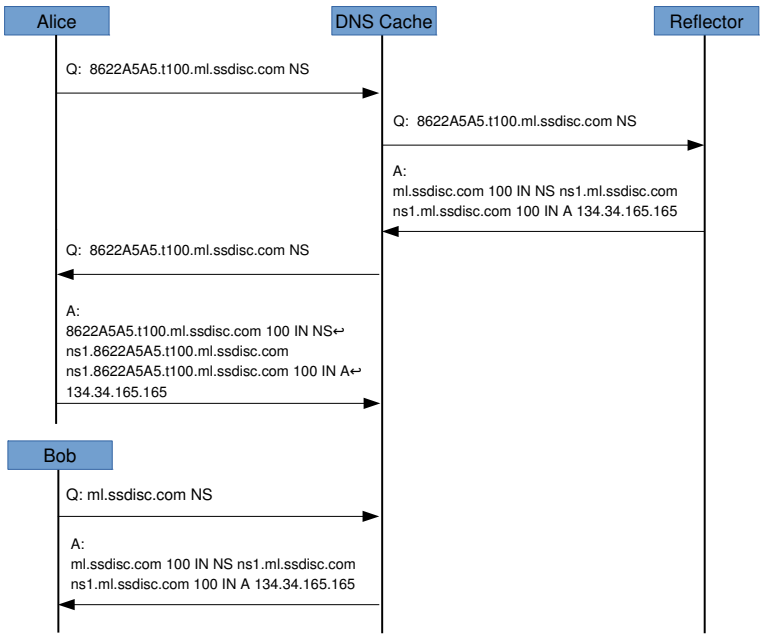

Fig. 1. Using Stateless DNS to provide an NS record for the ml-scope of our example service discovery domain with a TTL of 100 seconds. Alice registers her device as SNS and Bob retrieves the registered information. Thereafter, Bob can interact with the SNS running on Alice's device registering and retrieving service instances.

also receive a service discovery domain from an institution's DHCP server. This would only pose a minor configuration overhead for an institution compared to establishing a centralized service discovery directory, because it only needs one DHCP entry and the reflector script described below.

4.1.2 Providing the NS Records Using Stateless DNS [4] it is possible to enter DNS resource records in the cache of the currently used caching DNS server. For the purpose of making SNSes available to hosts within the same institution – using the same caching DNS server – we only use the method to store NS resource records. This Stateless DNS method is the most stable of all and works with all tested DNS caching server implementations, because it just uses a standard DNS zone delegation.

To make Stateless DNS work, the only additional infrastructure we need is a stateless reflector implemented e.g. in a few lines of Perl (see appendix). It acts as authoritative name server for the parent zone of the service discovery domain.

Figure 1 shows the process of entering an NS resource record in the current DNS cache using Stateless DNS. Alice sends a programming query – which is a valid DNS query – by asking for an `NS` resource record with the label

```
8622A5A5.t100.ml.ssdisc.com IN NS
```

that will be handled by the local DNS server. Because the stateless reflector is authoritative for this query, it will receive the query from the local DNS

server. The reflector then generates the following response using only information contained within this query[7].

```
Question:   8622A5A5.t100.ml.ssdisc.com IN NS
Authority:  ml.ssdisc.com     100 IN NS ns1.ml.ssdisc.com
Additional: ns1.ml.ssdisc.com 100 IN A 134.34.165.165
```

Since it is a delegation to an in-Bailiwick[14] name server the cache will accept the answer if this in-Bailiwick name server – Alice's notebook with the IP address `134.34.165.165` – is able to answer the programming query and NS queries for the service discovery domain.

```
Question:   <LABELS.>ml.ssdisc.com    IN NS
Authority:  ml.ssdisc.com 100 IN NS ns1.<LABELS.>ml.ssdisc.com
Additional: ns1.<LABELS.>ml.ssdisc.com 100 IN A 134.34.165.165
```

The programming query answer's sole purpose is to make the caching name server store the NS entries. The answer for NS queries of the service discovery domain is needed to be able to retrieve the SNSes' IP addresses.
The general form of the programming query is

```
(<hexenc_IPaddr>.){1,4}t<TTL>.<scope>.<sd_domain>
```

allowing to specify up to four name servers in a single programming query. All specified name servers have to be able to answer the programming query and the NS queries for the service discovery domain. Each of them has to return *all* NS entries, which is no problem as the SNSes know each other, to make the caching name server ask the next available name server in the case the first one asked is offline.

4.1.3 Establishing SNSes When entering a network, a host supporting our multi-link DNS-SD technique sends a standard DNS query asking for NS records belonging to the service discovery domain to get a list of SNSes. The query corresponding to our example service discovery domain is

```
ml.ssdisc.com NS
```

In the bootstrap phase there will be no SNS and the host will become the first SNS using the Stateless DNS method explained above (Figure 1). Because these NS entries cannot be overwritten, the TTL should be chosen adaptively. The first SNS should choose a low TTL when establishing its first NS entry, e.g. 10 seconds. To mitigate race conditions arising when other SNSes exist whose TTL just expired in the moment the new host asks to become name server, the new host must ask a second time after a random back off.

When the host gets a list of SNSes, it asks one of them whether it should also join the set of SNSes and if requested joins the set. This concludes discovering the current SNSes and the host can now ask for service information or register service instances as described in section 4.4. Figure 2 illustrates the process of SNS discovery.

It might happen that none of the returned SNSes answers. This is the case when all of them have gone offline without the TTL expiring and with no new

[7] The IP address of the new SNS is transmitted in hexadecimal notation.

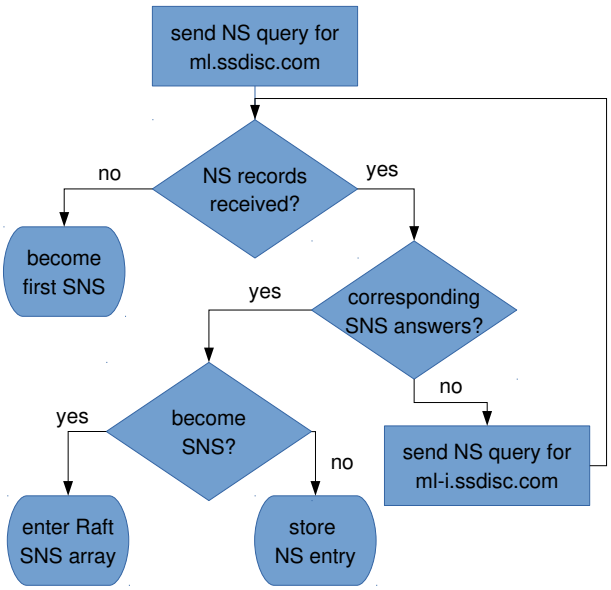

Fig. 2. Process of querying for existing SNSes and becoming SNS.

SNSes joining. Since the discovery domain is blocked in this situation, because Stateless DNS cannot override existing NS records, we need a defined fallback discovery domain and means to restore the standard discovery domain as soon as possible. The fallback discovery domain can be derived from the standard discovery domain by appending -0 to the highest scope defining label; thus the fallback domains for `ml.ssdisc.com` and `floor3.buildingB.ssdisc.com` are `ml-0.ssdisc.com` and `floor3.buildingB-0.ssdisc.com`, respectively. With increasingly low probability, the fallback domain might be blocked in the same way. The fallback of the fallback is defined to be the domain with the corresponding integer incremented by 1. When a discovery domain is blocked in this way, a host has to check whether the next fallback domain has an SNS that answers and if not, become SNS for this fallback domain (see Figure 2). To recover from this situation as soon as possible, a host that has to become name server for a fallback domain sets the TTL of its `NS` resource records to the remaining TTL of the NS records belonging to the standard domain. This makes the standard discovery domain's TTL and all fallback domains's TTLs expire at the same time and the host can register for the standard domain with all fallback domains being unblocked. Because of the adaptive TTL, the chain of backup domains is expected to be small. Further this problem will only occur in the bootstrap phase because the system will stabilize as described later.

4.2 Synchronization of SNS data

To make the system robust, we need several SNSes for a scope. These SNSes have to synchronize so that clients receive consistent information. Further, we want a querying client to be able to abstract away from the existence of several SNSes

that need to synchronize, considering each of them as an equal representer of a black box providing the desired information.

For synchronizing the SNSes we use Raft [12], a simple and efficient consensus protocol, that uses heartbeat messages and randomized timers to elect a leader whose log state is copied to the other participants called followers. After each heartbeat there is consensus[8] about who is leader and what the current log state is. In our case the log state contains information about registered service instances in form of `SRV` and `TXT` resource records. Changes of membership are also expressed and communicated using the log state.

Raft is very efficient with respect to network load as it only uses $2n$ messages per heartbeat, where n is the number of participants. During each heartbeat phase the leader sends a message to each follower, which can

- be empty if there were no log changes
- contain a log change
- contain a confirmation of a log change

and each follower sends a message to the leader which can

- be empty if the leader's message was empty
- contain a configuration of a log change if the leader's message contained a log change.

If the leader fails to send a message within the heartbeat before a follower's timer runs out, the follower becomes a candidate and sends a further type of heartbeat message to each participant, asking to vote it as the new leader. Participants that did not get any other heartbeat message within the current interval will send an answer message voting for the candidate to become the new leader. If the candidate gets consensus it becomes the new leader. All information needed to agree on log changes, leader changes and membership is communicated in these $2n$ message per heartbeat interval.

4.3 Processing Queries

Our goal is to also incorporate all the additional logic necessary for the SNSes within the heartbeat messages described above. Since we want each SNS to be able to process service information updates, we allow a follower to append an update request within the heartbeat answer. The leader receiving this request will then issue the corresponding update request to all followers. This message will also be received by the issuing follower serving as a confirmation.

4.3.1 Deciding on a new SNS The size of the set of current SNSes should be chosen dependent on the number of service instances offered at the moment. The decision whether a querying client should become SNS should be based on a ranking taking the hosts expected time to be stay online into consideration. As of yet, we are still assessing which kind of ranking to use. If there is only one or two other SNSes, the new client should definitely be chosen. The closer the number of SNSes gets to the desired SNS set size, the higher the rank of the new client should be in oder to be accepted. Still there should be the possibility to substitute the new client for an SNS in a full SNS set if the rank is appropriate.

[8] There is a negligible possibility for split votes.

4.3.2 Updating the NS Records When the TTL of the current NS entry in DNS cache runs out, the current leader has to reenter itself and some of the followers with the highest ranking. The longer the current leader is online and the higher the ranking of the other SNS, the longer is the newly chosen TTL. Still the TTL should not exceed a sensible limit. Since the current leader will stay the leader as long as it is online, the system will stabilize. The longer hosts are online the more likely they are to be leader.

4.4 Querying the SNSes

As with standard DNS-SD/DNS a client can ask an SNS for

- a listing of all existing service types,
- a listing of service instances of a service type and
- the resolution of a certain service instance.

The process of querying is independent of the structure the SNSes are organized in. A client can query any of the SNSes retrieved from the DNS cache.

Clients that need to have an up-to-date list of instances of a certain service can request DNS push [15] from the SNSes; this is important e.g. to provide an up-to-date list of online contacts in a chat application.

4.5 Hierarchical SNSes

SNSes can delegate sub zones; e.g. to create sub scopes or to delegate the resolution of certain service types.

5 Integration and Architecture

This section gives a short introduction to our service discovery framework which is enhancing DNS-SD by giving control to the user supporting scalable reachability scopes - using techniques proposed in this report - and scalable privacy. Scalable scopes, as described above, allow users choosing the scope in which the services are offered and requested, e.g. the current link, the current institution, or the whole Internet; our privacy extension [16] allows to selectively offer services to chosen friends, chosen groups, or everyone in a scope.

5.1 An Efficient and User Friendly Service Discovery Framework

Figure 3 classifies service discovery techniques with respect to the reachability scope they are used in and the privacy they offer. To the end of providing such a service discovery framework, we combined existing techniques and found solutions for the yet missing parts. Solutions for the smallest scope of single-link local networks and the biggest scope of the whole Internet are given by DNS-SD/mDNS and DND-SD/DNS. To provide a solution for the gap in between, we developed DNS-SD over Stateless DNS which has been proposed in this report. These techniques combined do not cover the whole desired area as they do not provide privacy. DNS-SD/mDNS publishes private information about services in an unsolicited way [17]. To give users control over the offered and requested services, we do not only provide privacy in a binary way - either it is on or off -

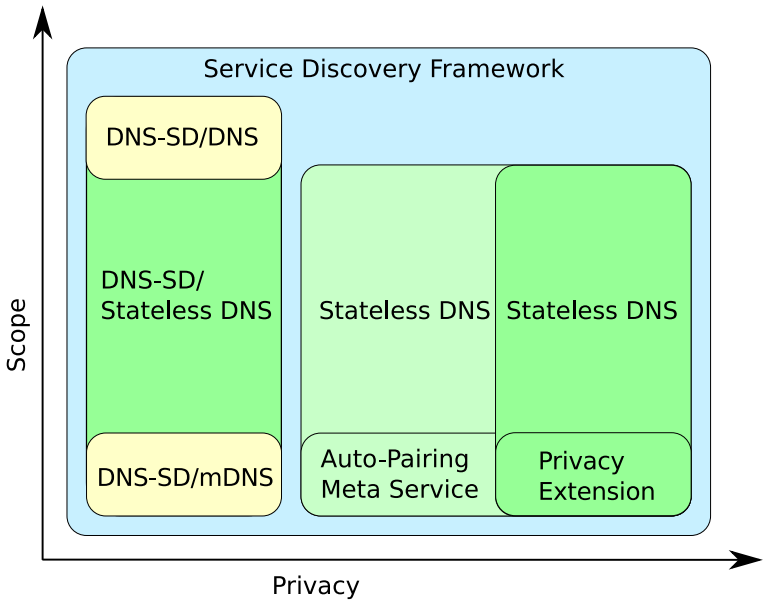

Fig. 3. Service discovery framework supporting scalable privacy and scope. The yellow areas are solved by existing RFCs; we provide solutions for the green areas. For the dark green areas we already have proof-of-concept implementations in form of extensions for the Avahi Zeroconf daemon.

but give the possibility to scale the privacy, i.e. choose a privacy level. This is important because the less privacy is demanded the easier is the necessary process of device pairing. To allow all users in the current network to discover a certain service, we provide a auto-pairing meta service that exchanges pairing information with all devices in the current network. This allows the paired devices not only exchanging service information in the current network but also in all other networks. Pairing at this level of privacy works without *any* configuration. Our privacy extension [16] allows to offer services to chosen friends. While the basic privacy extension, which we implemented as an extension to the Avahi[9] Zeroconf daemon, is limited to single-link networks like DNS-SD/mDNS, augmenting it using Stateless DNS allows scalable scopes within the privacy extension. We also implemented an enhancement of our privacy extension leveraging Stateless DNS to avoid the need of multicast [18].

5.2 User Control

The user interface to control the scope and privacy is provided by our enhanced service browser. Sensible defaults are preset to still allow configurationless service discovery. The enhanced service browser also manages user groups for the privacy extension. It does not matter if group members were paired using the auto-pairing meta service or a user pairing [16]. Users can manage groups, but configurationless group management is also offered: e.g. all friends paired using

[9] http://avahi.org

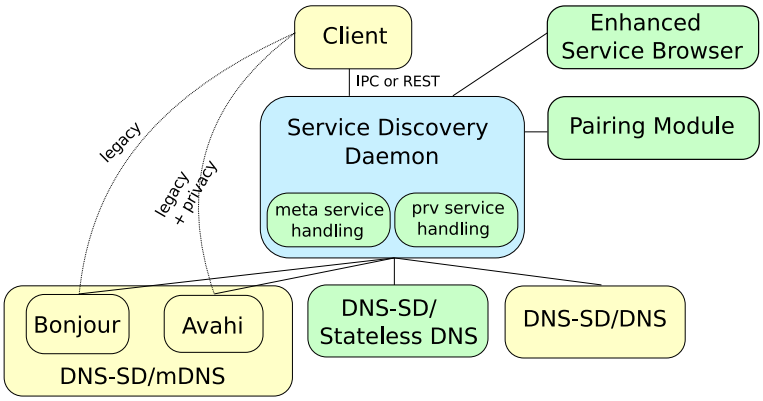

Fig. 4. Service discovery daemon (SDD) architecture. The SDD offers a unified interface to client software and demultiplexes client requests to different means of resource record distribution. Backwards compatibility is granted by the legacy interfaces.

the auto-pairing meta service in a certain network allowing e.g. to automatically get a group of all devices used in the home network.

5.3 Architecture

To ease the integration of alternative ways of service discovery, and to integrate into existing service discovery daemons, we propose a service discovery daemon (SDD) that is responsible to demultiplex client requests to different resolvers. Leveraging the proposed service discovery daemon, client software can use a unified interface for service discovery. Figure 4 illustrates our proposed architecture.

The SDD can be controlled by users via our enhanced service browser that allows to set the scope of discovery and privacy for single service instances, certain service types or all services. The SDD is also connected to a pairing module that handles pairing for privacy preserving service discovery [16].

Based on sensible defaults or decisions made by the user overriding the defaults, the SDD decides – given a client request – which means of resource record distribution has to be used.

Backwards compatibility is provided, because software not supporting the interface to the SDD still works as the interface to existing service discovery daemons has not been changed. Since we provide extensions to Avahi, existing clients can also use the privacy preserving service discovery and offer services in a multi-link scope.

6 Conclusion and Future Work

Multicast DNS Service Discovery over Stateless DNS provides a versatile, convenient and easily deployable means of resource record distribution for scalable DNS Service Discovery. Our proof-of-concept implementation – realized as an extension to the Avahi Zeroconf daemon – already allows to use DNS-SD in our campus WLAN where multicast is disabled. We showed how to integrate DNS-SD over Stateless DNS in our service discovery framework making it part of a

user friendly, efficient service discovery solution supporting both scalable scopes - with the help of the technique proposed in this report - and scalable privacy.

We plan to address further security and privacy problems arising when offering service information in scalable scopes. Further we will evaluate our scope extension with respect to network efficiency using the Omnet++ discrete event simulator[10].

We also plan to integrate DNS-SD hybrid proxy [19] capabilities in the SNSes as soon as the Internet draft becomes an RFC, which is likely to happen soon. This will allow hosts that are not aware of SNSes to use DNS-SD in multi-link networks providing a very elegant way of being backwards compatible. We also plan to adept SkyDNS (see section 2) as distributed directory for our solution.

[10] https://omnetpp.org/

A Reflector Script

```
#!/usr/bin/perl

use strict;
use warnings;
use 5.010;
use Net::DNS::Nameserver;
$|++;

sub reply_handler {
  my ($qname, $qclass, $qtype, $peerhost, $query, $conn) = @_;
  my ($rcode, @ans, @auth, @add, %headermask);

  #-----------------------------------------------------------
  # parse query
  #-----------------------------------------------------------
  $qname = lc($qname); # to lower case
  my (@ips, $ttl, $alias, $qbase);

  my $rx_i  = qr/(\w+(?:\.\w+){0,3})/;
  my $rx_t  = qr/t(\d{1,6})/;
  my $rx_a  = qr/((?:\w|-)+)/;
  my $rx_q  = qr/((?:\w+\.)*\w+)/;
  if($qname =~ /^${rx_i}\.${rx_t}\.${rx_a}\.${rx_q}$/) {
    @ips = split('\.', $1);
    ($ttl, $alias, $qbase) = ($2,$3,$4);
    $rcode  = "NOERROR";
  }
  else{
    $rcode = "NXDOMAIN";
    return ($rcode, \@ans, \@auth, \@add, \%headermask);
  }

  #-----------------------------------------------------------
  # assemble answer
  #-----------------------------------------------------------
  $headermask{aa} = 1; # set authoritative answer flag
  my $alias_domain = $alias . '.' . $qbase;
  my $ns_num = 1;
  foreach my $ip (@ips){
    my $ns_domain    = 'ns' . $ns_num++ . '.' . $alias_domain;
    #convert from hex to dotted notation
    $ip = join '.', unpack "C*", pack "H*", $ip;
    my $rr_ns = new Net::DNS::RR(name => $alias_domain,
      ttl => $ttl,
      class => "IN",
      type => "NS",
      nsdname => $ns_domain);
    push @auth, $rr_ns;
    my $rr_a = new Net::DNS::RR(name => $rr_ns->nsdname,
      ttl => $ttl,
      class => "IN",
      type => "A",
```

```
53          address => $ip);
54        push @add, $rr_a;
55      }
56
57      # return answer
58      return ($rcode, \@ans, \@auth, \@add, \%headermask);
59  }
60
61  # create nameserver object
62  my $ns = new Net::DNS::Nameserver(
63      # LocalAddr    => "127.0.0.1",
64      LocalAddr    => "51.254.124.217",
65      LocalPort    => 53,
66      # LocalPort    => 5300,
67      ReplyHandler => \&reply_handler,
68      # Verbose      => 0
69  ) || die "couldn't create nameserver object\n";
70
71  # start nameserver main loop
72  $ns->main_loop;
```

References

1. S. Cheshire and M. Krochmal, *DNS-Based Service Discovery*, ser. Request for Comments. Internet Engineering Task Force (IETF), 2013, no. 6763. 1
2. ——, *Multicast DNS*, ser. Request for Comments. Internet Engineering Task Force (IETF), 2013, no. 6762. 1
3. K. Lynn, S. Cheshire, M. Blanchet, and D. Migault, *Requirements for Scalable DNS-Based Service Discovery (DNS-SD) / Multicast DNS (mDNS) Extensions*, ser. Request for Comments. Internet Engineering Task Force (IETF), 2015, no. 7558. 1, 3
4. D. Kaiser, M. Fratz, M. Waldvogel, and V. Dietrich, "Stateless DNS," University of Konstanz, Tech. Rep. KN-2014-DiSy-004, Dec 2014. 1, 4, 4.1, 4.1.2
5. C. N. Ververidis and G. C. Polyzos, "Service discovery for mobile ad hoc networks: a survey of issues and techniques," *Communications Surveys & Tutorials, IEEE*, vol. 10, no. 3, pp. 30–45, 2008. [Online]. Available: http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=4625803 2
6. S. Hong, S. Srinivasan, and H. Schulzrinne, "Measurements of multicast service discovery in a campus wireless network," in *Global Telecommunications Conference, 2009. GLOBECOM 2009. IEEE.* IEEE, 2009, pp. 1–6. [Online]. Available: http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=5426121 2.1
7. G. Montenegro, N. Kushalnagar, J. Hui, and D. Culler, "Transmission of ipv6 packets over ieee 802.15. 4 networks," Tech. Rep. RFC 4944, 2007. 2.1
8. R. Klauck and M. Kirsche, "Bonjour contiki: A case study of a dns-based discovery service for the internet of things," in *Ad-hoc, Mobile, and Wireless Networks*. Springer, 2012, pp. 316–329. 2.1
9. ——, "Enhanced dns message compression-optimizing mdns/dns-sd for the use in 6lowpans," in *Pervasive Computing and Communications Workshops (PERCOM Workshops), 2013 IEEE International Conference on.* IEEE, 2013, pp. 596–601. 2.1
10. B. Djamaa, M. Richardson, N. Aouf, and B. Walters, "Towards efficient distributed service discovery in low-power and lossy networks," *Wireless Networks*, vol. 20, no. 8, pp. 2437–2453, 2014. 2.1
11. B. Djamaa and M. Richardson, "Towards scalable dns-based service discovery for the internet of things," in *Ubiquitous Computing and Ambient Intelligence. Personalisation and User Adapted Services.* Springer, 2014, pp. 432–435. 2.1
12. D. Ongaro and J. Ousterhout, "In search of an understandable consensus algorithm," in *Proc. USENIX Annual Technical Conference*, 2014, pp. 305–320. 2.2, 4.2
13. A. Rain, "An analysis of multicast traffic in wireless networks," Master's thesis, University of Konstanz, 2015. 3
14. S. Son and V. Shmatikov, "The hitchhiker's guide to DNS cache poisoning," in *Security and Privacy in Communication Networks.* Springer, 2010, pp. 466–483. [Online]. Available: http://link.springer.com/chapter/10.1007/978-3-642-16161-2_27 4.1.2
15. T. Pusateri and S. Cheshire, *DNS Push Notifications*, ser. Internet Draft. Internet Engineering Task Force (IETF), November 2015, no. 03. 4.4
16. D. Kaiser and M. Waldvogel, "Efficient privacy preserving multicast DNS service discovery," in *Workshop on Privacy-Preserving Cyberspace Safety and Security (IEEE CSS 2014)*, 2014. 5, 5.1, 5.2, 5.3
17. ——, "Adding privacy to multicast DNS service discovery," in *Proceedings of IEEE TrustCom 2014 (IEEE EFINS 2014 Workshop)*, 2014. 5.1
18. D. Kaiser, A. Rain, M. Waldvogel, and H. Strittmatter, "A multicast-avoiding privacy extension for the avahi zeroconf daemon," *Netsys 2015*, 2015. [Online]. Available: https://www.netsys2015.com/wp-content/uploads/NetSys2015_Demo_Kaiser.pdf 5.1

19. S. Cheshire, *Hybrid Unicast/Multicast DNS-Based Service Discovery*, ser. Internet Draft. Internet Engineering Task Force (IETF), November 2015, no. 02. 6