Optimization of Ad Hoc Device and Service Discovery in Large Scale Networks

Vlado Altmann, Peter Danielis, Jan Skodzik, Frank Golatowski, and Dirk Timmermann
University of Rostock
Institute of Applied Microelectronics and Computer Engineering
18051 Rostock, Germany
Email: vlado.altmann@uni-rostock.de

***Abstract*—Device and service discovery is an essential part of automation networks in order to ensure the interoperability of devices and Plug&Play functionalities. Embedded Web services such as Devices Profile for Web Services are an upcoming technology for building and factory automation. For ad hoc discovery, the WS-Discovery standard is used. This approach avoids single points of failures. However, it shows low scalability due to static response time values. The rising number of devices leads to an increasing network load and especially to an increasing device processing load as a result of the discovery process. In this work, we evaluate the issues of ad hoc service discovery and suggest an optimized algorithm for dynamic data rate reduction in large scale networks. Moreover, the suggested improvements allow reducing peak data rates by more than 50% for constant timing parameters. Using the suggested approach, the ad hoc discovery can be applied in any network size. The desired peak data rates or response times of the system can be significantly reduced and dynamically adjusted according to the desired values. This approach guarantees a scalable ad hoc discovery mechanism and is backward compatible with legacy equipment.**

Keywords—Web services; Simulation; Communication networks; Embedded software

I. INTRODUCTION

In recent years, the significant growth in smart home and smart metering sectors led to an increasing number of installed devices. Moreover, new trends and recommendations in communication technology suggest the use of media independent technologies such as IP and open interfaces such as Web services to connect these devices [1][2]. In order to ease the installation, Plug&Play technologies are desired. The mentioned demands can be fulfilled by the Devices Profile for Web Services (DPWS) [3]. Thereby, WS-Discovery is an essential part of Plug&Play functionality [4]. It allows searching for all or some specific devices or services in a local network. Two types of discovery are supported by this standard: an infrastructure-based and an ad hoc discovery. Infrastructure-based discovery utilizes a service broker – a central instance, which lists all available services in the network. The benefit of this approach is that only one device must be asked for service discovery. However, the service broker represents a single point of failure (SPoF). Thus, if it is out of operation service discovery becomes unavailable. An ad hoc discovery mechanism overcomes this problem. The discovery is sent via multicast and thus every device should send a response. The drawback of this approach is the low scalability of the original WS-Discovery standard due to a static application delay – maximum waiting time before sending a response. The rising number of devices leads to an increasing network load and especially to an increasing device processing load as a result of the discovery process, e.g., in smart metering or building automation networks. In the worst case, the device can be set out of operation due to buffer overflow or some devices will not be discovered due to packet rejection.

In this work, the scalability issues of ad hoc WS-Discovery are addressed. For timing optimization, we analyzed the behavior of WS-Discovery in large scale networks by means of simulations. For this purpose, the Network Simulator 3 (ns-3) was used [5]. As communication medium, switched Ethernet network as common communication technology is assumed. The peak data rates occurring on the searching device were observed dependent on the number of nodes, channel data rate, switch and device capabilities, application delay, and other parameters. The simulation results are presented. Based on the simulation results, we derive a formula for dynamic adjustment of the application delay according to the desired peak data rate. The calculated delay will be injected in the discovery request message and should be used by the devices instead of the static values. Some legacy devices that do not support dynamic application delay can still use the static application delay. Thus, the suggested approach is backward compatible. Dynamic parameters imply the knowledge of a network, i.e., number of connected devices. For this purpose, a network size discovery approach is provided.

For further performance improvements, we apply device aggregation. Thereby, all devices will no longer share the same timing interval but will be subdivided in groups. Each group shares one time slot that represents some part of the total timing interval. The subdivision is done on the searching device based on IP addresses according to the estimation. Group membership constants will be also included in the request. Thereby, each receiving device can calculate its group affiliation. For a more efficient network usage, we provide a mechanism for the group size determination based on simulation results. Legacy equipment that does not support device aggregation can ignore the included values and still use the complete timing interval.

Moreover, Address Resolution Protocol (ARP) [6] traffic volumes can become critical in large scale networks, since the searching device can receive and must process an ARP request from all other devices. In order to avoid this, the hardware address is also included in the discovery request. WS-Discovery uses an unreliable UDP binding. Thus, packet repetitions are required to improve reliability. We suggest a more efficient timing schema for packet repetition in terms of performance without impact on reliability.

The suggested improvements allow reducing peak data rates by more than 50% for constant timing parameters. Thus, the response time of the system can be reduced for the same peak data rates. Briefly summarized the main contributions of this paper regarding WS-Discovery are the following:

- Hardware address injection
- Dynamic application delay
- Node aggregation
- Optimization of packet repetition
- Network size discovery

The remainder of this paper is structured as follows. Section II gives a short overview about related work in the field of device discovery. Section III describes the suggested improvements for WS-Discovery. The network size discovery approach is presented in Section IV followed by the conclusion and future work in Section V.

II. Related Work

Device discovery is an essential feature that allows Plug&Play capabilities for any network device. Two kinds of discovery can be applied: an infrastructure-based discovery and an ad hoc discovery. The drawback of the infrastructure-based discovery is the SPoF in the form of a service broker, also called Universal Description, Discovery and Integration (UDDI). In [7], a distributed structure of the UDDI based on a Distributed Hash Table (DHT) is suggested. Although the UDDI is resilient against failures due to its distributed realization, the devices need to know a service proxy to be able to access the registry since devices are not part of the DHT themselves. Therefore, the service proxy represents a SPoF itself. In [8], the authors suggest a mixed architecture consisting of ad hoc discovery and service proxy. Thereby, the entire network is divided into small subnets. Ad hoc discovery is used for the subnet search. For a cross network search discovery, the proxy is requested. The IP addresses of service proxies are transferred as vendor specific option in the Dynamic Host Configuration Protocol (DHCP). The solution counteracts the SPoF nature of the service broker. However, if the service proxy is not reachable, discovery in affected subnet is not possible. Moreover, the authors do not discuss the scalability of their approach with a rising number of subnets. In contrast, fully distributed ad hoc discovery can achieve maximal reliability. Service discovery optimizations using hybrid discovery mechanism are suggested in [9]. However, the suggested approach is only applicable for wireless sensor grids. In contrast, in this work service discovery for Ethernet-based automation networks is investigated. In [10], the device discovery is realized in the form of a device scan. Single device addresses or addresses with wildcards can be requested for response. This kind of discovery scales due to the limitation of requested address space. However, this solution significantly reduces the total response time of the system. The number of responses can be reduced as well by searching for some specific service type or using semantic search [11]. However, the number of responses and the accruing traffic cannot be predicted. In [12], the lower bound for application delay is suggested as follows:

$$D = S * \frac{G}{R}, \tag{1}$$

where D denotes application delay, S is the message size, G is the group size (number of nodes), and R is data rate. By means of Formula 1, an average data rate on the client can be calculated and will be used as reference in this work. However, in this work we consider peak data rate estimation since it has more significant impact on the client.

Our proposed solution utilizes the benefits of ad hoc discovery. We provide optimizations that allow scalability and reliability of WS-Discovery for all network sizes without impact on total system response time.

III. WS-Discovery Optimizations

WS-Discovery is used to discover devices and services in the local network. In this work, only the ad hoc discovery is considered since it represents a distributed process without SPoF. The drawback of the WS-Discovery is the limited scalability due to the static timing behavior. The request uses SOAP-over-UDP binding with the specific multicast group. The specification dictates to use a random application delay between 0 and 500 ms before sending a response. Moreover, the UDP binding claims to send a response for a second time after 50-250 ms to increase reliability. Since more devices must share the same timing interval to send a response, the data rate increases. Thus, making the application delay customizable would help to adjust the timing interval to the network size. The broad rule mentioned in [12] allows the estimation of the average data rate for the client. However, the most critical part is the peak data rate. During the peak data rate, the client must process a high amount of data. In the worst case, this can lead to buffer overflow, packet discarding, or network congestion. We are considering switched Ethernet network as physical media since it is the most common communication technology nowadays.

A. Hardware Address Injection

For device discovery, *Probe* messages are used. In the *Type* field, a desired device or service type can be specified. It can also be left empty if all services need to be discovered. After the service provider receives the message, it waits the application delay and then tries to send the response. In Ethernet networks, knowledge about the destination Media Access Control (MAC) address is required in order to create a packet. In the case of the new requesting device, the service provider has to learn its MAC address. For this purpose, ARP is utilized. ARP requests are also sent as multicast. The client responds with its own MAC address to the service provider. Afterwards,

the first WS-Discovery response can be created and sent to the client. This procedure is illustrated in Figure 1.

As required by the SOAP-over-UDP binding, a second response must be sent after 50-250 ms in order to achieve higher reliability. Thus, one WS-Discovery request will evoke four times as much packets as devices in the local network in a short period of time. The requesting device must consequently process three times as much packets as devices in the network. In order to improve scalability and relieve the network and the client, we suggest the injection of the hardware address into the WS-Discovery request. Devices can then directly use it for Ethernet packet creation. The hardware address (HWAddr) can be injected in the Probe message as follows:

```
<s12:Body>
  <wsd:Probe>
    <wsd:Types />
    <wsd:HWAddr>01:01:01:01:01:01</wsd:HWAddr>
    …
```

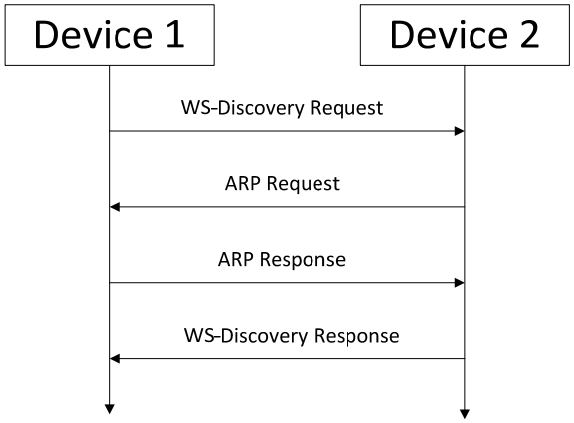

Figure 1: Message flow during service discovery

If legacy equipment cannot understand this new field, it can be ignored, since it does not have any impact on the functionality of the WS-Discovery. The injection of the hardware address results in the reduction of the overall transmitted packet number by a factor of two. By omitting the ARP protocol, the client must only process the WS-Discovery responses.

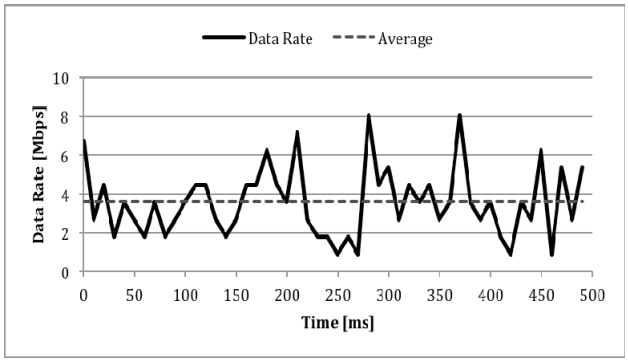

Figure 2: Comparison of actual and average data rate

B. *Dynamic Application Delay*

Ns-3 was used to investigate the network behavior. In the simulation scenario, 100 Mbps Ethernet (100BASE-T) is selected exemplarily as common Ethernet type in automation networks. The WS-Discovery response is set to an average value of 1176 Byte. It corresponds to a response containing two service types. In order to review the traffic load caused by WS-Discovery, we checked the data rate on the client depending on the number of nodes in the network. Thereby, we omitted the repetition of the response to compare the result with Formula 1. The experiment was performed for 200 nodes. As shown in the Figure 2, the data rate can highly deviate from the average value. The peak data rate can be even more than twice as high as the average data rate. Thus, considering the peak data rate is an essential issue in order to protect the client in the worst case.

To estimate the behavior of WS-Discovery, we varied the number of nodes and application delay and measured the peak data rate on the client. For the purpose of accuracy, about 100 simulations with different values were performed. The results are presented in Figure 3.

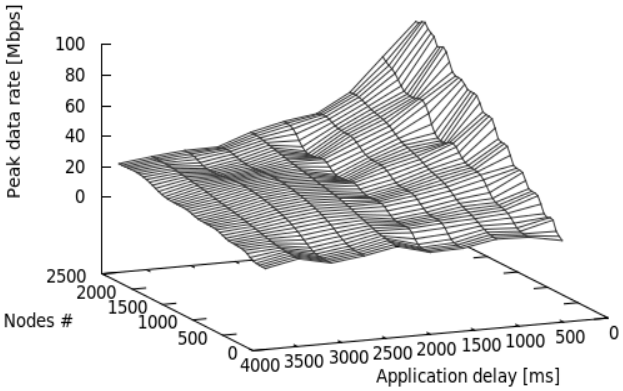

Figure 3: Peak data rate dependency on application delay and number of nodes

The peak data rate is linearly dependent on the number of nodes and represents a power function of application delay. Thus, in order to ensure scalability of WS-Discovery, the application delay must be adjusted in such way that the desired peak date rate is not exceeded. However, setting the application delay to a very pessimistic value (higher value) would lead to a very high response delay of the system. We used the collected simulation results to approximate the behavior of WS-Discovery. For function approximation, the least squares method was used. The resulting formula is provided below:

$$D = \left(\frac{5.25 * G + 407.94}{R}\right)^{1.3}, \qquad (2)$$

where D is the application delay in ms, G is the number of nodes, and R is the desired data rate in Mbps. The approximating function is presented in Figure 4.

Formula 2 provides a tradeoff between peak data rate and system response. Since it bases on estimations, the values for the number of nodes and desired data rate should nevertheless

be chosen pessimistic rather than optimistic. The calculation of the resulting delay should take place on the client side. The desired data rate is manufacturer dependent. The number of nodes is an estimated value. The network administrator can, for example, specify it. If the client does not have any information about the number of network devices, the maximum network size should be chosen. Maximum network size can be determined, e.g., by means of network mask in case of IPv4. For more general network size estimation, we suggest a mechanism in Section IV.

After the delay definition, the client must inject the calculated result into the WS-Discovery request. The *Delay* value in ms can be included into the request as follows:

```
<s12:Body>
  <wsd:Probe>
    <wsd:Types />
    <wsd:Delay>650</wsd:Delay>
    …
```

All receiving stations should take into account the new value for the maximum application delay and select the corresponding value between 0 and *Delay* ms. Some devices, which cannot understand or adjust the application delay can ignore it and still use default values. However, in some cases the client can reject default timing responses if the response timeout is less than 500 ms.

The suggested approach spreads the timing interval and hereby reduces the peak data rate. However, the deviation of the peak date rate from the average is still the same. In order to improve it, a node aggregation approach is suggested.

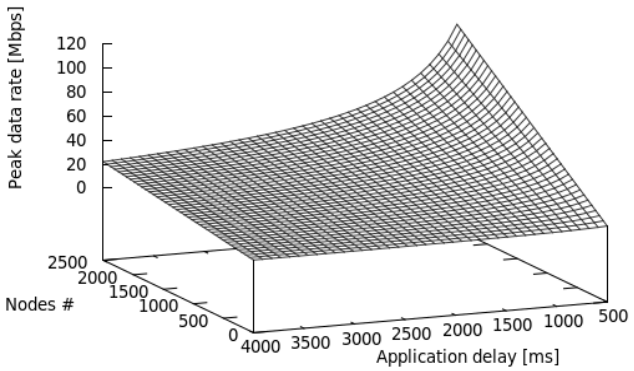

Figure 4: Delay function approximation

C. Node Aggregation

During examination of Figure 2, it is noticeable that the function has many spikes and pits. If it were possible to force some nodes, which are sending during the spikes, to send during the pits then the data rate behavior would be more balanced. In order to achieve this behavior, we divide the total timing interval into slots. Thus, every node is assigned to one time slot. Thereby, several nodes can share the same time slot. Dependent on the nodes assignment, we can get slots with many nodes and slots without nodes. In order to achieve an efficient distribution of the nodes, an IP-based approach is suggested. In the common Ethernet network, a DHCP server is present. When a new station goes online, it requests an IP address from this DHCP server. Usually, the server performs an iterative assignment of the IP addresses starting with the lowest one and proceeds to the highest possible IP address. Thus, it is more likely that the majority of the nodes are assigned to the lowest part of the IP space. Consequently, if the nodes were assigned to the time slots sequentially it would lead to overfilled and empty slots. A more efficient way is to assign IP addresses to one time slot from both IP spaces: full and empty. In this case, time slots would be half full and the assigned nodes can share more time for packet transmission.

The nodes are assigned to the time slots by using modulo operation. The client should decide how many slots are reasonable for the current constellation. In order to determine the optimal number of slots, we provide simulation results for a network with 250 and 1000 nodes and a variable number of slots. Moreover, we assume that the smallest slot period is 1 ms. Since different devices with different capabilities can be connected to the network, WS-Discovery must guarantee that all devices support a timing resolution of 1 ms. According to the smallest slot period, a total timing interval of 500 ms can then be subdivided into maximum 500 slots.

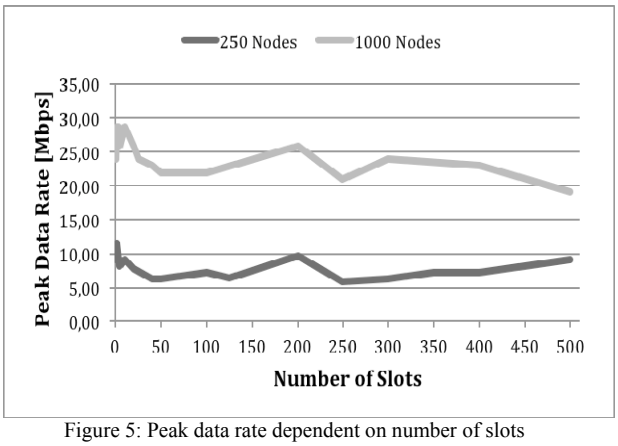

Figure 5: Peak data rate dependent on number of slots

As shown in Figure 5, different numbers of slots produces local maxima and minima. However, one local minimum applies for all number of nodes if the nodes number is larger than the maximum possible number of slots, i.e., when the smallest possible slot size is reached. This local minimum corresponds to the maximum possible number of slots. In the second case, one minimum applies for all number of nodes if the number of nodes is less than the maximum possible number of slots. In other words, the minimum peak data rate occurs if every node has its own slot. In the real network, this is less likely to occur since IP addresses are not distributed exactly sequentially. When some station leaves the network, a gap emerges. However, freed IP addresses can be assigned again to new stations. The suggested approach can nevertheless approximate the desired behavior. According to the gained knowledge, a simple formula for the number of slots estimation is provided:

$$S = S_{\max}, \text{ for } G > S_{\max} \text{ and } S = G, \text{ for } G < S_{\max}, \quad (3)$$

where S is the number of slots, S_{max} is the maximum possible number of slots, and G is the number of nodes (group). The

client can inject the desired number of time slots into WS-Discovery response as follows:

```
<s12:Body>
  <wsd:Probe>
    <wsd:Types />
    <wsd:Slots>250</wsd:Slots>
    …
```

Any device, which wants to send a WS-Discovery response, has to determine its time slot affiliation. For this purpose, it only needs to apply modulo operation to its own IP address according to the number of slots. Afterwards, the random application delay within the associated time slot must be calculated.

D. Optimization of Packet Repetition

In all our previous considerations, we did not consider the repetition of WS-Discovery requests due to the SOAP-over-UDP binding. It is obvious that repeated packets will interfere with delayed packets of the first response from other nodes.

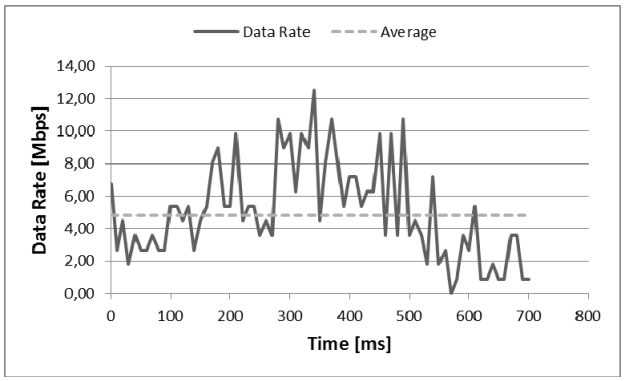

Figure 6: Data rate of WS-Discovery with packet repetition

If we consider the first example (see Figure 2) with packet repetition, even higher peak data rate occur (see Figure 6). We adjusted the average data rate according to the overall timing interval and number of packets sent for comparison purpose.

Since SOAP-over-UDP binding claims packet repetition after 50 – 250 ms after sending the first packet, the repetition packet can interfere with the first response from the other station, which delayed it by more than 50 ms. In order to avoid additional packet interference with repeated responses, a division of the timing interval into first and second response parts is proposed. The repetition part should start right after the end of the first part. All improvements mentioned above can also be applied to the repetition part. For this purpose, this part must be symmetric to the first one. In order to improve reliability and avoid random network errors, the second part should present the full repetition of the primary part. In this case, the temporal distance between the original response and the repetition will be maximal. However, in order to keep the response timing equal, the first response part must be shortened in favor of the second part. Thus, both parts will reach the same peak data rate and allow more balanced packet processing by the client.

If all improvements are applied to the described scenario, an essential reduction of the peak data rate (over 50%) on the client side is achieved (see Figure 7). As we previously mentioned, the theoretical average value cannot be achieved in reality. However, the suggested approach can improve the approximation. If the network devices support node aggregation, a common rule expressed by Formula 1 with more pessimistic values can satisfy the requirements for the peak data rate. Thus, Formula 1 can be applied for optimized WS-Discovery for any number of nodes, message size, and data rate (i.e., 1 Gbps, 10 Gbps Ethernet) that was also approved by simulations.

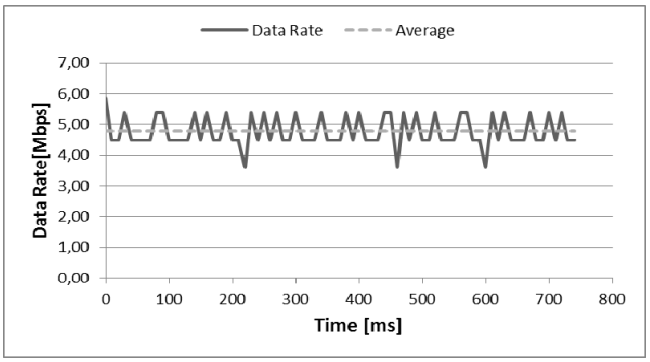

Figure 7: Data rate of WS-Discovery with improvements applied

IV. NETWORK SIZE DISCOVERY

The approach suggested above provides efficient discovery mechanism for any number of nodes. However, one condition must be fulfilled: the network size must be known beforehand or must be provided by a network administrator. This approach can be unfavorable in automation and Plug&Play networks, where direct device access is not possible or not in the interests of the owner. In order to overcome this issue, we suggest an approach for network size discovery. This feature is not originally provided by WS-Discovery. Thus, some specifications are required. In this work, we suggest a mechanism, which can be used to discover the size of the network.

If a new device joins the network and announces itself with the "Hello" message, it is informed by other devices about the network size. All other devices should already know the network size at that moment. Since automation or Plug&Play networks are built up iteratively (some device is connected at first), all devices get to know each other while the network grows. Thus, a network size notification can be sent from any device. The notification is sent as a multicast. This gives two benefits. First, if some notification is detected in the network, all other devices keep quiet. Second, devices can check their stored values for the number of devices. In order to send a notification, a device must gain a sending permission. The probability for sending a notification depends on the number of devices. With an increasing number of devices, the probability decreases. Additionally, the notifications should be sent temporarily separated from each other. Thereby, the sending time is determined by means of the random application delay. If some malfunctioning device would gain a sending permission and send a wrong value, this can lead to network failures (Byzantine failures). To detect and avoid a Byzantine failure, at least 3 messages are required if the network size is larger

than 2 (majority voting). If all 3 messages have different values, more messages from other devices are required.

For a better comprehension of the approach, let us consider a network with 1000 nodes. All devices already know the network size, as they were informed during connection. A new device that is connected to the network sends a "Hello" message to present itself. All other devices automatically increment the number of nodes value when they receive the "Hello" message. In the next step, the new device must be informed by means of a multicast network discovery message about the network size. As previously mentioned, at least 3 responses are required for large networks to avoid a Byzantine failure. For this purpose, all devices calculate a probability of sending a response resulting in 3 overall responses. For 1000 devices, this would correspond to 0.3 %. A simple random number generator is used for this purpose. In the first stage, the waiting time should be randomly set to a value between 0 and 5 ms. If some device gets the permission to send, it must wait the time period mentioned above before sending a network size notification. In the first stage, only 3 messages are expected. If any other device detects 3 network discovery messages, its own message must be dropped. Despite of the sending probability and notification drop, slightly more messages can occur due to the network delay. This behavior guarantees that the network will not be overwhelmed with notification messages. In case of a discrepancy of sent values or if not enough devices gained the sending permission, all devices switch to the next stage. In the next stage the sending probability (the number of sending devices) and the maximal waiting time is doubled. This procedure can be repeated until the correct value is determined. If the number of devices is less than 3, only the first stage is required. Although the mentioned procedure can be repeated until the sending probability reaches 100 %, we expect that network size discovery will be accomplished in very few stages. In most cases, the first stage will be sufficient.

When some device leaves the network, it sends a "Bye" message. All other devices must then decrement the stored network size value. If a device crashes or is unplugged from the network improperly, it will not be able to send a "Bye" message. However, this does not pose a problem for service discovery as traffic load on the requesting device is reduced in this case.

WS-Discovery does not provide a mechanism for handling simultaneous "Hello" messages from a large number of devices. This can happen, e.g., after a blackout if all devices power up at once. Thus, we also suggest using a power up waiting time of up to 1 minute in order to avoid network congestion and device overloading.

V. Conclusion and Future Work

In this work, we provide an approximation formula for peak data rate estimation of WS-Discovery in Ethernet networks. According to the simulation results, we propose optimizations for WS-Discovery to improve its scalability. The proposed approach includes MAC addresses in WS-Discovery requests avoiding large traffic volumes due to the address resolution protocol. Moreover, we suggest dynamic adaptation of the application delay in order to reduce the peak data rate on the client. Furthermore, the suggested additional node aggregation can reduce the data rate independent of the application delay by more than 50 %. The optimization of packet repetition allows avoiding the packet interference between original and repeated WS-Discovery responses. For this purpose, we propose the division of timing interval into two parts for the first and the second response, respectively. In order to properly set the dynamic values, the knowledge of the network size is required. A new approach for network size discovery is proposed to gain this knowledge. All optimizations allow steady packet processing and significantly relieve the client. The suggested approach can be applied for any network size due to its dynamic nature and consequently provides improved scalability for ad hoc WS-Discovery. The suggested optimization is backward compatible with the standard WS-Discovery specification.

Prospectively, WS-Discovery behavior on other media such as powerline or RF will be investigated.

Acknowledgment

We would like to thank the German Federal Institute for Research on Building, Urban Affairs and Spatial Development (BBSR) within the Federal Office for Building and Regional Planning for their support in this project. This work is partially granted by BBSR within the scope of project Zukunft Bau.

References

[1] National Institute of Standards and Technology, "NIST Framework and Roadmap for Smart Grid Interoperability Standards, Release 2.0", NIST Special Publication, 2012.

[2] SMB Smart Grid Strategic Group, "IEC Smart Grid Standardization Roadmap", Edition 1.0, 2010.

[3] OASIS, "Devices profile for web services version 1.1", 2009. [Online]. Available: http://docs.oasis-open.org/ws-dd/dpws/wsdd-dpws-1.1-spec.html.

[4] OASIS, "Web Services Dynamic Discovery (WS-Discovery) Version 1.1", 2009. [Online]. Available: http://docs.oasis-open.org/ws-dd/discovery/1.1/wsdd-discovery-1.1-spec.html.

[5] ns-3, [Online]. Available: http://www.nsnam.org.

[6] D. C. Plummer, "An Ethernet Address Resolution Protocol", RFC 826, 1982. [Online]. Available: http://tools.ietf.org/html/rfc826.

[7] S. Banerjee, S. Basu, S. Garg, S. Garg, S.-J. Lee, P. Mullan, and P. Sharma, "Scalable grid service discovery based on UDDI", The 3rd international workshop on Middleware for grid computing, pp. 1–6, 2005.

[8] S. Poehlsen and C. Werner, "Robust Web Service Discovery in Large Networks", IEEE International Conference on Services Computing, vol. 2, pp. 521-524, July 2008.

[9] R. Moreno-Vozmediano, "A hybrid mechanism for resource/service discovery in ad-hoc grids", Elsevier B.V., 2009.

[10] "Communication systems for and remote reading of meters - Part 3: Dedicated application layer", EN 13757-3, 2011.

[11] Zeng ZhiHao; Hu JiPing; Dong Ting; and Wang Yu, "Semantic Web Service Similarity Ranking Proposal Based on Semantic Space Vector Model", Second International Conference on Intelligent System Design and Engineering Application (ISDEA), pp. 917-920, 2012.

[12] Z. Shelby, K. Hartke, C. Bormann, and B. Frank, "Constrained Application Protocol (CoAP)", Internet Draft, 2012. [Online]. Available: https://datatracker.ietf.org/doc/draft-ietf-core-coap.