Leveraging OPC-UA Discovery by Software-defined Networking and Network Function Virtualization

Dominik Henneke*, Alex Brozmann*, Lukasz Wisniewski*, Jürgen Jasperneite*†,
*inIT — Institute Industrial IT
OWL University of Applied Sciences, 32657 Lemgo, Germany
{firstname.lastname}@hs-owl.de
†Fraunhofer IOSB-INA Application Center Industrial Automation
32657 Lemgo, Germany
{juergen.jasperneite}@iosb-ina.fraunhofer.de

***Abstract*—The discovery process of OPC Unified Architecture (OPC-UA) is well defined but lacks flexibility when implemented in large networks. While the standard provides concepts for local and global discovery services, the core assumption is that all clients and servers are aware of the discovery server(s) in the network. This dependency is a barrier for Plug-and-Play installations and is overcome by self announces via the multicast Domain Name System (mDNS), however, this scales badly with an increasing number of network nodes and deployed servers. We propose a discovery approach based on Software-Defined Networking (SDN) and Network Function Virtualization (NFV), that installs a plug-and-play mechanism for global OPC-UA discovery in modern virtualized networks.**

I. Introduction

The evolution of intelligent and distributed applications in industrial control applications far advanced during the last decades. Fixed control loop wirings were amended by early field-bus networks, superseded by specialized Ethernet-based protocols, and might soon be replaced by standardized protocol stacks in combination with modern communication and modeling protocols such as Time-Sensitive Networking (TSN), OPC Unified Architecture (OPC-UA), or 5G [1]. In this context, topics such as interoperability and auto-configuration get increasing attention, especially since technologies from the Information and Communications Technology (ICT) domain and the usability experiences known from end-user applications find their way into the industrial domain.

OPC-UA, as one example of modern frameworks (that spans different domains such as information modeling, encoding, transport, or discovery), allows the implementation of consumer-producer interactions based on self-describing information models and automated discovery mechanisms. But though the protocol specifications theoretically deliver a complete toolset for self-description and self-configuration of distributed systems, actual implementations need to face the limitations of todays network architectures. Based on these limitations, this paper proposes an automated server discovery that exploits the improvements of future network architectures and supports use cases such as *Condition Monitoring-as-a-Service* or *Simplified Production Commissioning*. Its goal is the definition of a flexible and universal discovery approach to provide information about all OPC-UA servers in a (routed) network without any change in the original OPC-UA standard, i.e. a Discovery-as-a-Service approach.

The remainder of the paper presents the relevant state-of-the-art approaches, and proposes and evaluates a potential solution to the stated problems. It starts by describing related work in the domains of auto-discovery, OPC-UA, and companion technologies in section II. Section III focuses on the problem statement, presents the proposed solution, names potential applications, and reviews details of a prototype implementation. Finally, section IV summarizes the paper and discusses whether the proposed approach is a valuable extension and should be pursued in the future.

II. Related Work

A. Auto-Discovery & OPC Unified Architecture (OPC-UA)

Automatic configuration and discovery in networks are widely used across nearly all applications of distributed systems. Basic uses include DHCP-assigned interface addresses, ARP or DNS resolutions, or application-layer services such as multicast Domain Name System (mDNS) or Apples Bonjour. OPC-UA extends this network focused scope and lets clients find the correct endpoint instance on a server and the matching information in an endpoints information model [2]. The discovery service itself is implemented in dedicated discovery servers in different scopes, namely the Local Discovery Server (LDS) and the Global Discovery Server (GDS). The LDS is used to resolve different server endpoints on a single host and usually shares the network interface with the discovered servers. In contrast, the GDS is a separate entity in the network with its own IP-Address, providing the same functionality as the LDS but spanning different hosts that register at the GDS. Thus it provides information about several servers in a local subnet or an administrative domain. The information about servers, provided by the discovery services, contain a description of the servers features and their endpoints. They represent the root directories of servers as a starting point to browse the information model and other provided services. Fig. 1 shows the LDS discovery procedure.

978-1-5386-1066-4/18/$31.00 ©2018 IEEE

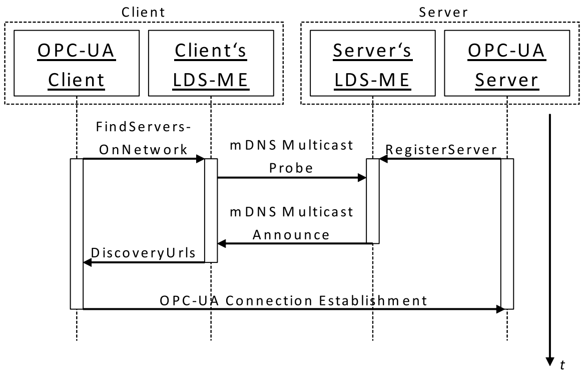


Fig. 1. Procedure of the local discovery process with LDS-ME.

Both the GDS and the LDS depend on the client to know the host to access discovery directories. In cases where the hosts of other (discovery) servers are unknown to a client, the standard specifies the Local Discovery Server with Multicast Extension (LDS-ME). It allows LDSs to announce their address and registered services on the local subnet using mDNS. Due to the design of Multicast messages, this only allows a discovery of servers in a local broadcast domain.

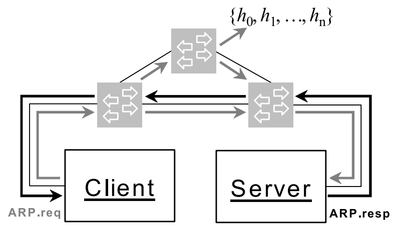
B. Software-Defined Networking (SDN)

SDN converts legacy full-stack network appliances into modular systems that consist of three logical layers: application, control, and data [3]. Forwarding and traffic shaping are no longer decided on individual appliances but are implemented by network wide policies. This increases programmability and agility of network configurations and provides full control over the forwarding, as well as vendor-neutrality by using Open-Source solutions. There are already studies about the use of SDN in industrial automation networks [4], [5]. Additionally, the concept of SDN is also considered in the configuration and management of TSN-based networks [6].

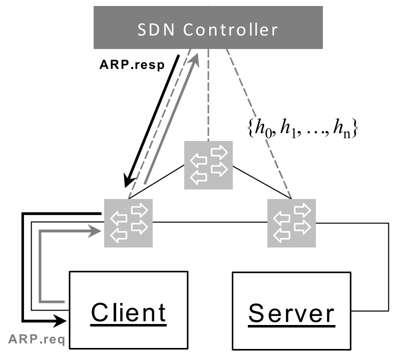
In the discovery context, SDN can transform basic network services, such as the address resolution, from a distributed to a centralized approach. Fig. 2 depicts the differences between a regular ARP resolution and an SDN-based one, as it is implemented in existing SDN-controllers. In Fig. 2a, the resolution is distributed and all messages are broadcasted in the local network to resolve network addresses. In Fig. 2b, the network appliances forward these frames to the SDN controller, which resolves the requested IP-addresses with its global network knowledge. Since the original protocol doesn't need to be changed and the resolution is transparent to the client, this approach works with all existing network devices and reduces the network load. This approach can also potentially be adapted to other discovery services.

C. Network Function Virtualization (NFV)

The main feature of NFV is the decoupling of hardware and software of network functions. Virtual Network Functions (VNFs) are realized by software modules running on virtual computing resources in private or public cloud infrastructures.



(a) Regular ARP resolution



(b) SDN-based ARP resolution

Fig. 2. Comparison of different ARP resolution processes in a network containing a client, a server, and several hosts $h_i, i \in [0..n]$.

This allows the usage of powerful commodity hardware that is much more scalable and flexible compared to traditional hardware-bound network functions. The infrastructure can also be shared among different "network islands" in a distributed network system. The full potential of NFV can be utilized when combined with the SDN approach. This allows to create efficient algorithms, based on highly intensive calculations, that provide high quality VNFs, such as communication scheduling or load balancing, to the network system. These network functions can directly interact with the control plane of SDN in order to adjust the network accordingly. NFV includes not only the VNFs but also the means to manage and orchestrate them, and whenever a new VNF is needed, it can be added to the set of offered functions without the need to install additional hardware components in the local network.

III. SDN POWERED OPC-UA DISCOVERY

OPC-UA connections require shared knowledge between servers and clients in order to interact. They need to (**i**) know each other, (**ii**) know how to reach each other, (**iii**) use a shared communication protocol, (**iv**) have authorized credentials, and (**v**) be able to receive, understand, and interpret the respective address spaces. The following approach focuses on a solution for requirements (**i**) and (**ii**). Requirements (**iii**) and (**v**) are integral parts of the OPC-UA specification or rather of related companion specifications [7] and (**iv**) is viewed as out-of-scope for this evaluation.

A. Problem Statement

Section II described different approaches of OPC-UA to learn about communication and service endpoints of servers, while the approaches share the need to explicitly register at a directory service and a client to explicitly query it. Therefore, both need to know the directories communication endpoint. The LDS is installed at the same device as the client or server and can assumed to be known, but the GDSs endpoint is infrastructure dependent. OPC-UA bypasses this limitation by exploiting the mDNS protocol to discover the addresses of the GDS as well as of LDSs of other servers in a local subnet, however, this doesn't work across broadcast domains and also scales badly [8].

The goal of the following proposal is the definition of a discovery mechanism, that meets the following characteristics:

- Discover servers across administrative domains
- Provide a central directory with a well-known API
- No explicit registrations of servers at a GDS
- No (extra) configuration at the endpoints
- No changes in the specification or implementations
- Less to no communication overhead

B. Solution Approach

The distributed nature of computer networks and the lack of global knowledge calls for auto-discovery extensions to overcome the drawbacks in terms of knowing which services are provided in a network. In order to analyze refinement points in the discovery process, this can be split into four different actions, each executed by different entities:

1) Register resources at a discovery participant [Server]
2) Announce registered resources [LDS]
3) Compile (distributed) resource directories [LDS/GDS]
4) Query the directories [Client/LDS]

Action 1) is required if a server and its discovery participant (i.e. the LDS) are divided. This should not be changed under the assumption that both components are located on the same device and can be accessed on the loopback interface. For action 2), only LDS-ME is relevant in this study. Distributing all information to each LDS or GDS instance results in a large flood of Multicast messages due to the $N : M$ relationship between participants. At this stage, the concepts of SDN and NFV step in to resolve this relationship into an $N : 1 : M$ relationship, whereby a new question arises: Can action 3) be modeled as a VNF-instance and can the SDN-approach forward service-announcements to this instance, in order to maintain a global service directory and to provide a single service endpoint for querying all available services?

Fig. 3 shows the proposed SDN/NFV-based discovery approach that replaces the need for a GDS with explicit server registrations, reduces the number of Multicast messages that would normally flood the network, and is able to discover all services in the SDN-controlled network (including broadcast domains). The SDN-controller installs default flows to network appliances so they forward the announces to the VNF. The newly formed VNF uses the received announcements to maintain a list of all OPC-UA servers in the network. If messages need to be sent by the VNF during the discovery process, these can be instructed via the SDN controller.

In this solution, actions 1) and 2) remain unchanged, i.e. each server endpoint provides a local LDS-ME that announces

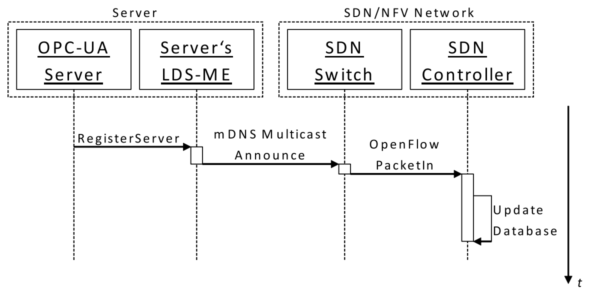


Fig. 3. SDN-based OPC-UA Server Discovery.

its services according to the specification. Action 3) is executed by the newly defined VNF but there still remains an open question for action 4). A simple solution would be to provide a well-defined API at the VNF (e.g. the original GDS interface), implying that the clients are aware of this. The OPC-UA specification supports other service directories such as LDAP, but does not further specify the implementation. This would require non-standard extensions, that were initially excluded in section III-A. The best approach is to stick to existing mDNS procedures, a client would use to discover local services, with the difference that the VNF replies to all requests. This adopts the behavior of the SDN-based ARP resolution.

With the modeling of the discovery service as VNF, the functionality was moved to the network itself and all clients can profit from it without changes.

C. Application Areas

1) Condition Monitoring as-a-Service: In addition to the described discovery process, a user can install a VNF to automatically monitor the production process based on the information from all OPC-UA servers (and devices). If using the companion specifications and standardized information modeling, condition monitoring can be automatically applied to all OPC-UA-powered sensors/actuators (similar to [9]).

2) Network Management: Knowledge about the devices in the network and what services are installed on them is valuable for a network management system [10]. The service directory would add an information source that can help the user to learn more about its network and the services that are run, and lets automated processes propose optimization algorithms.

3) Production Commissioning: The engineering and reconfiguration of (new) production machines needs knowledge about available sensors or actuators [11]. The proposed approach provides a single information source of reachable services, can enable new use cases in this domain, and can be integrated into existing engineering tools.

D. Prototype Implementation

The described proposal has been implemented in a prototype using ONOS[1]) as SDN controller and as VNF-runtime, where the discovery service was developed as application plugin.

[1]https://onosproject.org/

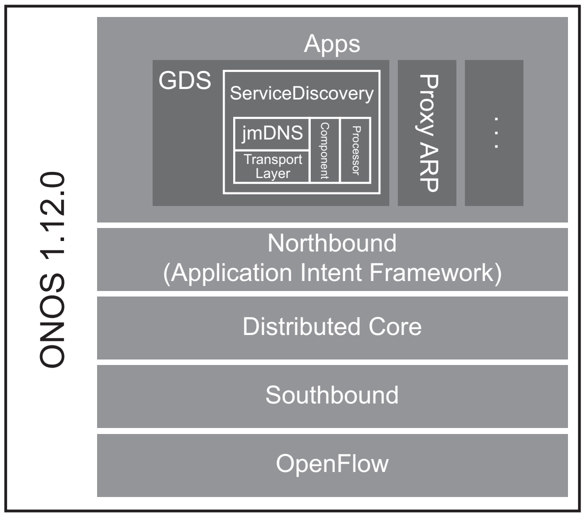


Fig. 4. Architecture of the ONOS implementation.

The jmDNS framework[2] provides the mDNS resolver with an adapted transport layer to use the ONOS framework to send and receive frames. Fig. 4 shows the resulting architecture.

The *Component* is the main entrypoint of the application. It acquires resources and installs the required flows:
Match: $\{eth_{type} = IPv4, protocol = UDP, UDP_{dst} = 5353\}$
Action: $\{OUTPUT = CONTROLLER\}$
The *Processor* processes all packets that are received by the controller, filters the relevant packets (based on multicast IPv4, protocol, port), and pushes it into the custom *Transport Layer*. The *Transport Layer* implements the MulticastSocket interface, hooks up the received packets into the jmDNS library, and also receives send-requests from the library and forwards them via a $PACKET\_OUT$ to the receivers. Finally, a service listener is used to receive the resolved services and updates the local server database.

For the test, an OPC-UA server was setup with open62541[3] and run in Mininet[4]. Multiple virtual hosts each provide an OPC-UA server and advertise their addresses. The ONOS application receives the advertisements and provides a list of discovered devices via its RESTful API. With this setup, the proposal and the implementation could be verified in a test environment. A further validation in a real production environments with commercial OPC-UA servers is still open. By default, the multicast discovery is not enabled in open62541, so it is questioning whether this feature is already available in commercial devices.

IV. Conclusion

This paper introduced an improved discovery approach for OPC-UA applications. By combining the existing discovery approaches with the SDN and NFV architecture, it allows the deployment of a network wide discovery service, that discovers all available servers as long as they implement the standardized LDS-ME approach. The main advantage of this approach is that clients and servers don't need to be aware of a dedicated GDS but rely on the discovery function that is provided directly by the network. The main disadvantage is that the proposed solution requires a deployed SDN and NFV infrastructure. Future developments (also in TSN management) will show whether this can be expected in real networks or not.

Open issues include the possibility to use this approach in production networks (also with the focus on brown-field deployments), the study of permission control and the automated credential management (similar to [12]), as well as a more extensive evaluation of the concept in a real environment with commercial solutions and off-the-shelf OPC-UA servers.

[2]https://github.com/jmdns/jmdns/
[3]https://open62541.org/
[4]http://mininet.org/

Acknowledgment

This work has received funding by the German Federal Ministry of Education and Research ("Deutsches Bundesministerium für Bildung und Forschung – BMBF") as part of the research project "TACNET 4.0 – Tactile Internet".

References

[1] M. Wollschlaeger, T. Sauter, and J. Jasperneite, "The future of industrial communication: Automation networks in the era of the internet of things and industry 4.0," *IEEE Industrial Electronics Magazine*, vol. 11, no. 1, pp. 17–27, March 2017.
[2] OPC Foundation, *OPC Unified Architecture Discovery*, release 1.03 ed., OPC Foundation, July 2015.
[3] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, and J. Turner, "OpenFlow: enabling innovation in campus networks," *SIGCOMM Comput. Commun. Rev.*, vol. 38, no. 2, pp. 69–74, Mar. 2008.
[4] D. Henneke, L. Wisniewski, and J. Jasperneite, "Analysis of realizing a future industrial network by means of software-defined networking (SDN)," in *2016 IEEE World Conference on Factory Communication Systems (WFCS)*, May 2016, pp. 1–4.
[5] M. J. Mytych, D. Henneke, L. Wisniewski, and J. Jasperneite, "Potential of SDN/NFV network management concepts with regards to industrial automation," in *Kommunikation in der Automation – KommA 2017*, Magdeburg, Germany, Nov 2017.
[6] S. Schriegel, C. Pieper, S. Gamper, A. Biendarra, and J. Jasperneite, "Vereinfachtes Ethernet TSN-Implementierungsmodell für Feldgeräte mit zwei Ports," in *Kommunikation in der Automation — KommA 2017*, Magdeburg, Germany, Nov 2017.
[7] F. Pethig, S. Schriegel, A. Maier, J. Otto, S. Windmann, B. Böttcher, O. Niggemann, and J. Jasperneite, *Industrie 4.0 Communication Guideline Based on OPC UA*. Frankfurt, Germany: VDMA Guideline, Publisher: VDMA Verlag GmbH, Editor: Verband Deutscher Maschinen- und Anlagenbau e.V., Sep 2017, vol. ISBN: 978-3-8163-0709-9.
[8] K. Lynn, S. Cheshire, M. Blanchet, and D. Migault, "Requirements for Scalable DNS-Based Service Discovery (DNS-SD) / Multicast DNS (mDNS) Extensions," Internet Requests for Comments, RFC Editor, RFC 7558, July 2015. [Online]. Available: https://www.rfc-editor.org/rfc/rfc7558.txt
[9] A. Bunte, A. Diedrich, and O. Niggemann, "Integrating semantics for diagnosis of manufacturing systems," in *2016 IEEE 21st International Conference on Emerging Technologies and Factory Automation (ETFA)*, Sept 2016, pp. 1–8.
[10] A. Neumann, M. Ehrlich, L. Wisniewski, and J. Jasperneite, "Towards monitoring of hybrid industrial networks," in *2017 IEEE 13th International Workshop on Factory Communication Systems (WFCS)*, May 2017, pp. 1–4.
[11] L. Dürkop, *Automatische Konfiguration von Echtzeit-Ethernet*. Springer Berlin Heidelberg, 2017.
[12] M. Rentschler, H. Trsek, and L. Dürkop, "OPC UA extension for IP auto-configuration in cyber-physical systems," in *2016 IEEE 14th International Conference on Industrial Informatics (INDIN)*, July 2016, pp. 26–31.