Discovery in SOA-Governed Industrial Middleware with mDNS and DNS-SD

Ahmed Ismail, Wolfgang Kastner
Institute of Computer Aided Automation - Technische Universität Wien
Vienna, Austria
Email: {aismail, k}@auto.tuwien.ac.at

***Abstract*—Research efforts for the Industrial Internet of Things (IIoT) have placed great emphasis on the use of vertical integration as a mainline strategy for the realisation of the fourth industrial revolution (Industrie 4.0). The present paper proposes the use of a distributed Gateway Service Bus (GSB) as a plant-wide and resilient platform for vertical integration, field level reconfiguration, and distributed execution in heterogeneous environments. A basic service set composed of network, discovery, and management services is developed using technologies from P2P networks as cooperative systems and zero configuration networking to achieve a survivable distributed GSB. For evaluation, the resulting implementation is executed on 11 virtual machines (VM) and across two subnets in 33 consecutive runs. Results were consistent in showing that the proposed service set is capable of achieving network-wide discovery of published resources in less than 3 seconds.**

I. Introduction

As of recent years, the Internet of Things (IoT) has become a well established research track in a number of academic fields. Within the industrial domain, the IoT has emerged synonymously with a promise for a fourth industrial revolution. Since then, many consortia and research collaborations have invested extensively into investigating the technologies necessary to make this revolution a reality. A predominant theme found in the ensuing research efforts is focused on the definition of methods for the advancement of Cyber-Physical Production Systems (CPPS), which are hybrid systems of computationally controlled physical processes. An architecture for progress has been put forth by the VDI/VDE Society Measurement and Automatic Control (GMA) as the Reference Architecture Model Industrie 4.0 (RAMI 4.0) [1]. A mainline strategy of RAMI 4.0 involves the use of vertical integration to manage system heterogeneity. Such heterogeneity results from the fact that industrial enterprises typically operate using layer-specific protocols that do not necessarily interoperate. This issue is complicated further when considering the long life cycles associated with industrial systems. These lifetimes ensure the presence of legacy protocols and devices in contemporary industrial environments. It is the task of vertical integration to tackle this heterogeneity to provide enterprises with the ability to adopt modern technologies, such as runtime data anomaly detection, which increasingly depend on the easy accessibility of data and devices.

978-1-5090-1314-2/16/$31.00

Typical methods of vertical integration involve the use of gateways for protocol translation or tunnelling routers for protocol encapsulation [2]. The former approach operates by carrying out the data mappings necessary to allow for communication to take place between two distinct protocols [3]. As for the latter, tunnelling, this operates by having the message of one protocol encapsulated in the payload of another and then treating the channel as a transparent communication medium [4]. Unfortunately, both of these methods are considered to be costly and complex engineering efforts. However, as has been outlined in [5] and [6], these costs may be justified if the pervasive nature of gateway deployments is exploited and combined with a service oriented architecture (SOA) to create the counterpart of the enterprise service bus (ESB), that is, a distributed GSB. In so doing, the resulting system, exploiting the modularity and portability of SOA elements, may be an idempotent and resilient platform for the execution of services and the standardisation and semantic-enrichment of plant data for easy accessibility by enterprise-layer applications.

To exemplify the potential of the proposed distributed GSB, Fig. 1 shows the application of such a system over a heterogeneous industrial Ethernet network. To allow for the distributed execution of production processes across heterogeneous cells, GSB devices such as nodes A and B may intercept field level communications and exploit tunnelling or translation techniques on behalf of their respective local devices to allow for the timely execution of the manufacturing process. However, the distributed GSB may also simultaneously map the acquired data to a set of models that standardises and semantically enriches it. The mapped information may then be served from a portal, such as device D of the DMZ layer, to allow business-enhancing applications operating at the plant's enterprise layer to access a single standardised form of plant data. Furthermore, a failure in node D may immediately be remedied given the distributed nature of the GSB. Another node in the DMZ, for example, may replicate the required service from node C to allow the distributed GSB to continue serving semantically-enriched plant data to the enterprise layer. In doing so, the distributed GSB realises a survivable infrastructure for enhanced vertical integration, while also actively participating in desirable and complex technical manufacturing applications such as distributed execution in heterogeneous environments.

To achieve these capabilities, a multitude of services for modelling and communication is required by the distributed

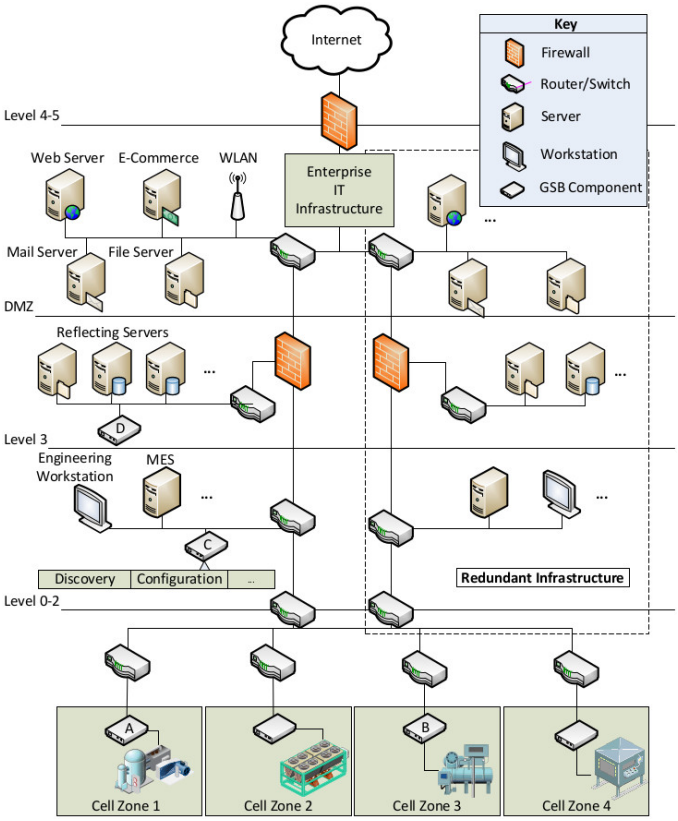

Fig. 1. Deployed Distributed GSB [5]

GSB [7]. Specifically for communication, reliable mechanisms for the coordination of the GSB nodes are a necessary pre-requisite to all functions. Therefore, within the scope of this paper, we focus on the definition of a basic set of services that would allow newly instantiated GSB devices to join a survivable network of distributed GSB nodes and participate in the most basic of functions (e.g. traffic re-laying). Such a set consists of three services; namely, the networking, discovery and management services. The first of these establishes reliable communication mechanisms for the transfer of messages between GSB nodes. The second service is required to allow for the instantaneous detection of service advertisements throughout the enterprise. Finally, the management service is the orchestrator of all executed services on a GSB node. The remainder of this paper is concerned with detailing the development of the basic service set. Consequently, the paper is structured as follows. To begin with, Sections II and III detail the technologies governing the networking and discovery layers of the SOA, respectively. Section IV then outlines the management service and its role in handling the networking and discovery services. Section V proceeds with an experimental evaluation of the basic service set. This is followed by Section VI, which discusses the acquired results, and Section VII, which concludes and gives some recommendations for future work.

II. The Networking Service

It is the purpose of this service to provide autonomous and reliable mechanisms for the coordination of nodes participating in the distributed GSB. Such mechanisms are critical if the functions described in Section I are to be realised. The networking service is based on previous work, namely [5], which establishes arguments in support of using technologies from the domain of P2P networks as cooperative systems for the non-real time (RT) coordination of nodes in the distributed GSB. According to [5], the selection of appropriate measures from this domain would allow the distributed GSB to operate as a survivable network that is capable of service-execution and vertical integration, all without violating the governing architecture of the targeted industrial Ethernet networks. This is as this domain specifically concerns itself with the dynamic creation, dissolution, and merging of independent overlay networks, as well as facilitating the meaningful participation of overlays in cooperative tasks such as intra- and inter-overlay content-sharing and traffic engineering. In such a manner, the distributed GSB network may in fact be composed of several overlays, with message passing between them facilitated or restricted in accordance with the governing constraints of the target systems' networking infrastructure.

To assess the aforementioned hypothesis, [5] converted a P2P protocol, Chimera[1], into a cooperative one. The resulting protocol, as shown in Fig. 2, operated using bootstrap nodes, co-located nodes, fault detection through the use of heartbeat messages, and discovery by iterative peer exchange transmissions. The first of these, bootstrap nodes, initialise newly joining nodes with the information needed by them to be able to communicate with other peers in the network. Co-located nodes, on the other hand, are bootstrap peers that are members of multiple P2P networks. Co-located nodes are therefore both capable of and tasked with relaying traffic between networks on behalf of regular, non-co-located peers. Heartbeat messages for fault detection involve the use of ping messages and acknowledgements to detect and prune non-responsive nodes from the network. Finally, a leaf set is a portion of the routing table that contains information on the nodes of a single overlay. The dissemination of this information uses a mechanism that is inherited from the original implementation of Chimera that requires that every node periodically creates and transmits a packet containing a leaf set to all of the known peers of a different leaf set.

The evaluation of this protocol in [5] was carried out using 50 virtual machines (VM) dispersed between a single 64 bit server and two 32-bit embedded devices. The number of overlays and nodes per overlay were varied and all communication logged. Acquired results highlighted two observable facts. The first is that the optimal network structure should have the number of overlays be double the number of nodes per overlay, or vice versa. The second observation showed that the discovery mechanism of the protocol that is dependent on unicast transmissions of leaf sets is detrimental to the

[1] Available: http://current.cs.ucsb.edu/projects/chimera/

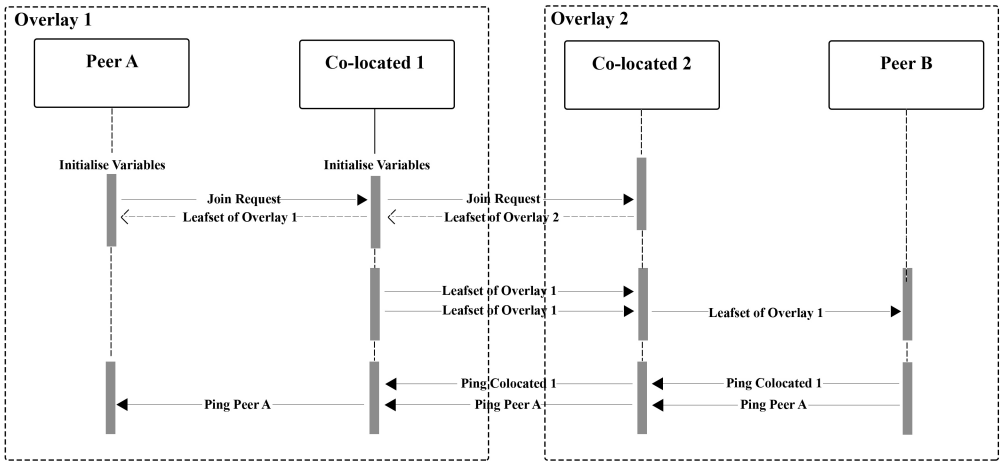

Fig. 2. Mechanisms of Chimera post-modifications [5]

performance of the network. These messages both overloaded the network with traffic, especially the co-located nodes, and were slow to deliver information necessary for the discovery of peers and overlays. For this reason, this paper disables the unicast-based discovery feature of the protocol and outsources discovery to an independent service. It is worth mentioning that another modification done within the scope of this paper is the extension of the protocol implementation with IPv6 capabilities in accordance with the principles of contemporary IoT-capable designs. With the networking layer discussed, the following section will proceed to detail the appropriate features of the discovery service.

III. The Discovery Service

This section is concerned with the selection and implementation of a robust and efficient discovery mechanism for the distributed GSB. It does not aim to provide an exhaustive list of all existing methods for discovery. Rather, we limit our discussion to mechanisms found in other P2P networks as cooperative systems and one from the domain of zero configuration networking.

To begin with the field of cooperative P2P systems, although this subdomain is relatively new, it enjoys several implementations such as [9] and [10], that provide us with a number of possible methods for discovery.

In the case of [9], a framework designed to enable the cooperation of heterogeneous P2P networks is presented. Node discovery within networks is left to the protocols of the respective overlays, while the discovery of co-located nodes is handled by the mechanisms of [9] which involve the use of flooding or a secondary overlay network called the Internet Indirection Infrastructure (i3). The former, flooding, is not an acceptable option as it would only create the same network conditions and strain as occurred in [5]. As for the i3 overlay based method, this is based on the work of [11] and depends on the use of a network of servers, called the i3 network, that behave as discovery and forwarding servers. For example, when a node becomes a cooperative node, it registers itself with an i3 server using a 'trigger' message containing a service identifier. Previously registered nodes periodically check in with the i3 infrastructure by pushing a packet type message containing a service identifier and their network address. The i3 servers forward these messages to other nodes registered with the same or similar identifiers. Once the new node receives this informational packet, it de-registers its trigger from the i3 overlay. Applied to the distributed GSB, this method would require the creation of a secondary infrastructure to support the discovery of distributed GSB nodes. The daunting complexity of this task causes us to look elsewhere for suitable solutions.

As for [10], similar to [9], local discovery is managed by the respective protocols of the member overlays. However, the discovery of co-located nodes, termed synapse nodes in [10], is done using message embedding, active notifications, peer exchange and aggressive discovery mechanisms. The first of these, message embedding, is when a node includes all of the overlays it is connected to in outgoing messages. Any synapse node that is part of the message path decodes this information and updates its tables. Active notifications, on the other hand, is when a synapse node, becoming aware of a transiting message, pro-actively notifies the sender of its connected overlays. As for peer exchange, this involves the iterative transmission of discovery-relevant information. Finally, aggressive discovery is an umbrella term imparted upon the exploitation of specific attributes of the member overlay protocols, such as leaf tables or peer lists, for discovery purposes; hence, [10] offers no specific procedures for the protocol. To evaluate these mechanisms, first off, both message embedding and active notifications are opportunistic discovery methods and do not guarantee the timely discovery of nodes. Likewise, the peer exchange mechanism is similar to the one already in place in [5] and would deliver no added benefit. Lastly, aggressive discovery, being a generic term, lacks any concise definition or specific procedure that would be useful to our implementation.

Unfortunately, other implementations from the field of P2P networks as cooperative systems either operate in a broadcast medium [12], use the mechanisms of the aforementioned

TABLE I
ADVANTAGES OF MDNS [8]

Factor	Explanation
"Opportunistic Caching"	A single multicast response may update all nodes in a network reducing the volume of queries in a network.
Query Suppression	If several machines have the same query, only one device needs to transmit it, and yet, all nodes receive the response.
Passive Failure Detection	If a node observes an unanswered query, this information may be used to prune stale data from the cache.
"Passive Conflict Detection"	Multicast advertisements allow all nodes to promptly detect violations to required unique attributes i.e. peer keys.
Constrained Devices	Multicasting reduces the need for resources for response transmissions. These would otherwise be required to accommodate a list of destination nodes for each response.
Multiple Subnets	If a node receives an advertisement published in a subnet, multicasting guarantees that the advertised services exist on the local link, regardless of the source address.
Robustness	In the case where every node's address, default gateway, subnet mask, and DNS server addresses are incorrectly configured, the use of a multicast address ensures that all peers will still be able to receive advertisement on the local link.

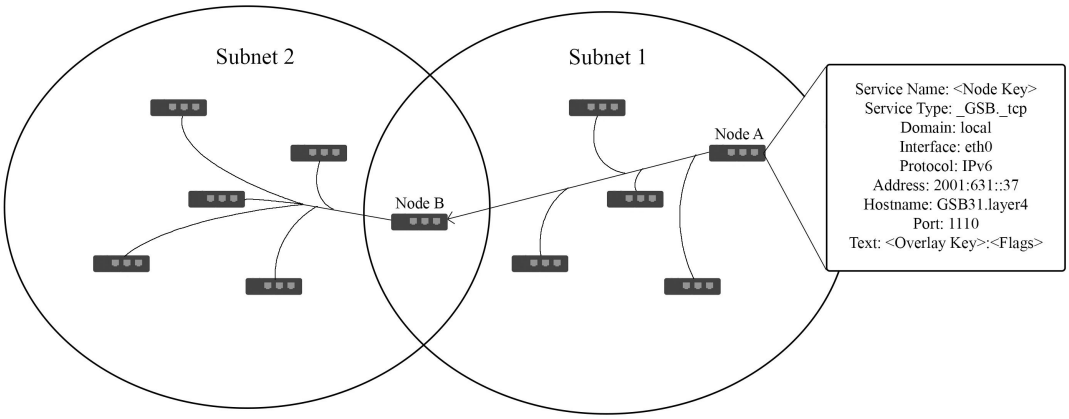

Fig. 3. Transmission of mDNS advertisements in a distributed GSB deployment

protocols [13, 14], do not offer mechanisms for discovery [15], or give incomplete solutions to our application's needs [16]. For these reasons, the coming subsection will look into the application of flexible dedicated service discovery frameworks that may be tailored to our system. Specifically, due to the local nature of the distributed GSB deployment, a multicast-based framework may be used in favour of a unicast-based scheme to reduce traffic. However, out of the various multicast-based discovery frameworks available, we specifically take into account the use of multicast-DNS (mDNS) and DNS Service Discovery (DNS-SD) as a possible solution due to the benefits listed in Table I, and sourced from [8], which guarantee a system with low overhead.

To achieve low discovery times, however, please refer to Fig. 3, which exemplifies the application of the mDNS and DNS-SD framework to our SOA. Here, an example of a service advertisement is created by node A of subnet 1. The advertisement has the service name set to the peer's key, the service type to _GSB._tcp, and the text field is split to contain the overlay key of the node as well as flags to indicate the services or resources locally available for consumption by other nodes. This advertisement is published by node A using the IPv4 and/or IPv6 multicast address. All nodes in subnet 1 with running mDNS discovery services promptly receive the advertisement and update their caches accordingly. To allow the nodes of subnet 2 to also receive this advertisement, and, hence, allow for instantaneous discovery throughout the infrastructure, an mDNS reflector may be executed on a multi-homed device that connects both subnets 1 and 2, which in this case is node B. The mDNS reflector allows node B to replicate an advertisement received from one interface on all of its other interfaces. This replication may also be used to allow for reflection between IPv4 and IPv6. In such a way, the advertisement of node A may be forwarded to subnet 2, allowing all of the GSB nodes of Fig. 3 to receive the advertisement of node A in a timely manner and with minimal effort. It is prudent to mention at this point, however, that the allocation of service advertisement fields such as the service name, or type, in the manner described above is only as such for exemplary and testing purposes. These assignments may change as the SOA of the distributed GSB matures to allow for extensive advertisements as will be discussed later on in Section VI. As for the integration of the aforementioned mDNS and DNS-SD based discovery service into the SOA, this is the topic of discussion in the coming section.

IV. THE MANAGEMENT SERVICE

As is typical of SOA-governed designs, the orchestration of the various services on a single device is a requirement. In this architecture, this feature is handled by the manage-

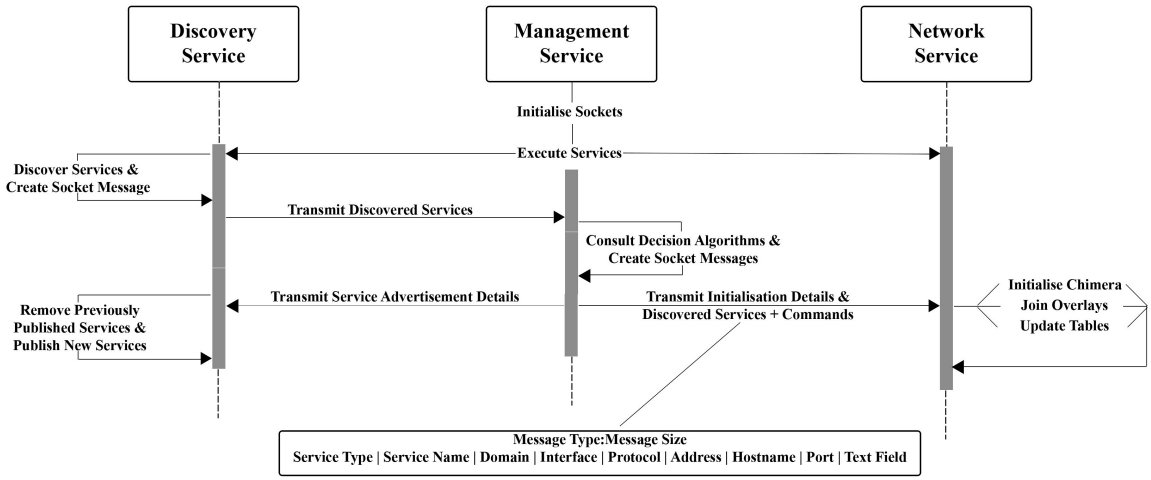

Fig. 4. Interactions in the basic service set

ment service, which is both responsible for the transfer of information between services and occupying the role of the GSB node's decision engine. The former, communication between services, is done using UNIX domain sockets. The management service is a necessary intermediary between these services to allow it to decode received information, and employ it in its decision making process. Currently, the algorithms that would be involved in such a process are outside of the scope of this paper. Instead, the entire system operates in the manner shown in Fig. 4. After the management service is initialised, bi-directional sockets are created and the mDNS and P2P services are executed. After execution, the mDNS service initialises an mDNS discovery thread and collects all advertisements published by other GSB nodes on its connected subnets, packages them in a string that is preceded by the message type and size, as shown in Fig. 4, and forwards the string to the management service via the socket. The management service decodes this information, makes a decision as to what services the mDNS should advertise, and what information to initialise the P2P service with, creates messages identical in format to that shown in Fig. 4 for each service, and forwards them accordingly. Once the information to be published is received by the mDNS service, the message is decoded and the services published. If services are already being advertised by the mDNS service, these are removed and the new ones are published in their stead. As for the P2P service, this, similarly, also decodes the message to receive its key, overlay key, and the port it should run on, which overlays it should join, and which nodes the P2P service should add to or remove from its routing tables. This process repeats indefinitely allowing published services and the status of the P2P service to be updated when required. It is worth mentioning that the management service also executes an mDNS reflector during the initialisation process if it recognises that it is operating on a multi-homed device. With the entire framework detailed, the coming section will proceed with its evaluation.

V. Experiment

The designed framework and the newly adopted discovery mechanism are tested using 11 Debian VMs hosted on a Xen Project (TM) server. These are distributed equally across two different network interfaces, with 5 nodes per interface, and one multi-homed with access to both interfaces. The topology used is shown in Fig. 3. The experiment is executed on all 11 nodes sequentially using a BASH script running in Domain-0 of the server. After execution, the script pauses for 60 seconds during which the behaviour of the networking, routing, messaging, discovery and other application layers and services are logged. Once the timer expires, the script resumes and terminates the framework on all nodes in sequential order before collecting all logs and restarting the experiment. It is important to note that the evaluation of the routing protocol itself is available in [5], the evaluation done here is to determine the speed of discovery within the context of the newly instantiated mechanism and architecture. The experiments were run a total of 33 times to ensure consistency in results.

For all 33 experiments, the discovery times for all nodes is summarised as a percentage distribution in Table II and the mean plotted in Fig. 5. The time taken to discover a service is calculated from the point that the discovering node is initialized or the discovered node's services are published, depending on whichever occurs at a later point on the timeline, till the time that the discovering node receives a copy of the advertisement. As may be seen from Table II, the discovery service implemented allows all nodes to consistently discover any advertised services within 3 seconds from the time of publishing. Having therefore achieved its set goal of timely discovery, the coming section will discuss further opportunities for future work.

VI. Discussion

This section is concerned with discussing possible enhancements for the discovery service as well as its relationship with previous work done in large industrial SOA research projects.

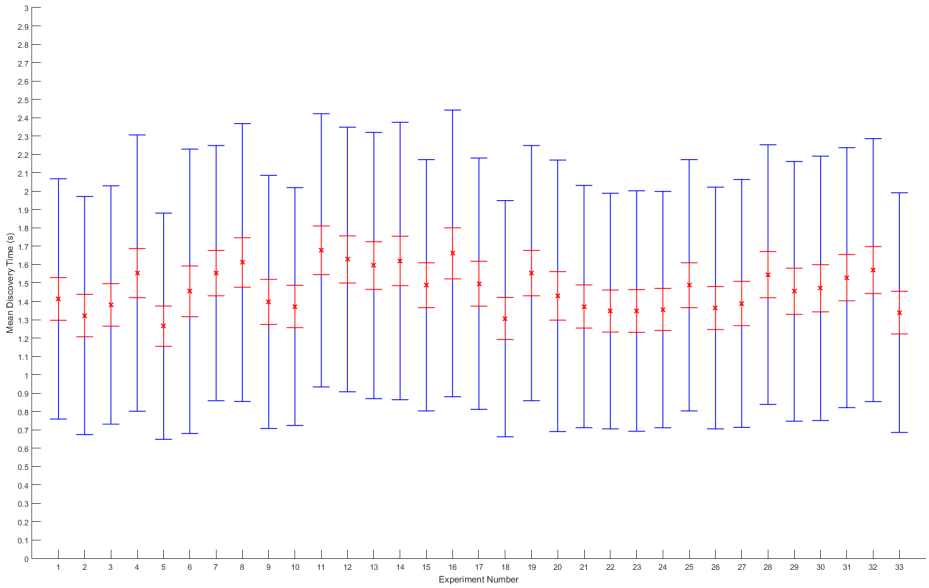

Fig. 5. Mean discovery times with the standard deviation (blue) and standard error of the mean (red) for each experiment

A. Potential Enhancements

Although the system in total achieves its purpose of providing the underlying infrastructure required of a distributed GSB improvements are, as always, possible. We discuss in this section two main concepts that may be applied to enhance the discovery service. Primarily, this involves the application of modifications to the service advertisement change-tracking methods and the reflector implementation.

The first of these, the detection of changes to advertised services, interestingly, also includes the tracking of service advertisement removal. As was previously stated in Section III, in our implementation, a node typically removes its published services before it advertises a new set received from its management service. This removal is done via a multicast message that may be integrated into the GSB as a method for the detection of failures in the network. Effectively, this mechanism may be used to replace the unicast heartbeat-based failure detection methods of Chimera with a multicast system that, again, would provide more timely detections of failure. Such a modification would allow for the integration of more enhanced fault tolerance mechanisms in the distributed GSB.

For example, in this implementation only one reflector may bridge the same set of networks. If more than one is active in the same subnets, a routing loop may occur that would allow for the amplification of multicast messages, and which would effectively amount to a denial of service (DoS) attack on the bridged networks. Backups to the reflector node, however, are necessary, as the failure of a reflector node would be detrimental to the discovery process of the distributed GSB. Naturally, the heartbeat messages mechanism of Chimera may discover the failure of the reflector node; however, its opportunistic nature provides no guarantees in terms of timely detection. The implementation may be modified to incorporate a pinging mechanism exclusive to the reflector nodes, albeit at the cost of increased complexity. Instead, the service removal multicast messaging system that is already available may be used as an indicator for the detection of failed nodes and the timely execution of appropriate measures by backup nodes in order to guarantee a responsive and survivable infrastructure.

A service removal mechanism in the manner described may only be useful, however, in cases where failing nodes have ample time to publish service removal messages. This may not be the case, for example, if a node suffers immediate and total power loss. In this situation, techniques, like the use of non-maskable interrupts and large capacitors in a topology such as in Fig. 6 may be used to give the node enough time to notify the network of its impending failure. In the case of high load nodes, such as servers, UPS systems may be required instead of capacitors. To keep costs down, modules similar to server Intelligent Platform Management Interface (IPMI) subsystems, which would not have high power requirements, may be given the responsibility of monitoring for power outages, and the publishing of service removal messages on behalf of high load nodes. To enact such a comprehensive system, however, requires a complete study of the methods available, costs involved, and mechanisms needed to guarantee timely transmissions of service removal messages in the various node and system failure scenarios possible.

A second possible method for improving the discovery service involves altering the reflector service to impart further reductions in network traffic. In this case, the reflector node, instead of forwarding all received mDNS messages to all other interfaces, may aggregate all of the resources in the received advertisements of one interface and publish them as

TABLE II
PERCENTAGE DISTRIBUTION OF TIME TO DISCOVERY IN THE EXPERIMENTAL ASSESSMENT

	% of services discovered by time T			
Experiment #	T=1s	T=2s	T=3s	Total
1	49.59	50.41	0.00	100.00
2	59.50	39.67	0.83	100.00
3	52.89	47.11	0.00	100.00
4	42.15	51.24	6.61	100.00
5	64.46	35.54	0.00	100.00
6	53.72	38.02	8.26	100.00
7	38.02	59.50	2.48	100.00
8	37.19	55.37	7.44	100.00
9	53.72	43.80	2.48	100.00
10	53.72	46.28	0.00	100.00
11	30.58	61.98	7.44	100.00
12	33.06	61.98	4.96	100.00
13	36.36	58.68	4.96	100.00
14	36.36	56.20	7.44	100.00
15	43.80	54.55	1.65	100.00
16	34.71	55.37	9.92	100.00
17	42.98	55.37	1.65	100.00
18	61.16	38.02	0.83	100.00
19	38.02	59.50	2.48	100.00
20	53.72	40.50	5.79	100.00
21	54.55	44.63	0.83	100.00
22	56.20	43.80	0.00	100.00
23	57.02	42.15	0.83	100.00
24	55.37	44.63	0.00	100.00
25	43.80	54.55	1.65	100.00
26	55.37	43.80	0.83	100.00
27	53.72	44.63	1.65	100.00
28	39.67	57.02	3.31	100.00
29	48.76	47.93	3.31	100.00
30	47.93	47.93	4.13	100.00
31	41.32	55.37	3.31	100.00
32	38.02	57.85	4.13	100.00
33	57.85	41.32	0.83	100.00

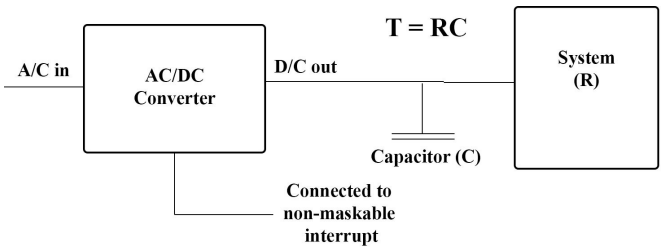

Fig. 6. Mechanism for detecting and delaying the loss of input power

resources local to the reflector node on all other interfaces. Mappings to the actual locations of these resources may then be kept by the reflector node allowing it to route any subsequent attempts to access the advertised resources to their respective locations. Effectively, this would reduce the amount of messages being relayed between networks. However, in order to avoid a scenario where the reflector node, being the only reflector, and hence, aggregator, would suffer the fate of also being the only traffic relaying node between subnets, a method for the distribution of the services to be aggregated across multiple reflector nodes may be required. Alternatively, the reflector service may simply cache responses from one subnet and respond on behalf of those nodes to any polls it receives from its other interfaces thereby reducing the impact of polling on large scale, multi-subnet deployments using a low-effort solution. A feasibility study would be needed at this point in order to determine the overhead cost of such approaches, or modifications thereof, in comparison to that of the currently implemented technique of reflection.

B. On Contemporary Projects

Observing the approaches implemented in large contemporary projects for SO industrial automation, we find that discovery is almost exclusively handled using the Web Services Dynamic Discovery (WS-Discovery) specification. Such projects include the EU FP 7 IMC-AESOP [17], PLANTcockpit [18], and eSONIA [19] research efforts. This specification uses multicast SOAP-over-UDP messages for advertising, searching for, and locating services on a local network. Similar to mDNS, the caching of multicast advertisements to reduce the number of subsequent in-network probes is encouraged. However, a difference lies in the number of possible situations that support caching. For example, while mDNS stipulates that query responses be multicast, WS-Discovery requires unicast responses. This means that in mDNS, if several nodes have the same query, only one node needs to probe for the service and all nodes may benefit from the response. In contrast, with WS-Discovery's unicast responses, this is not possible and indicates that the two specifications may have different impacts on the network performance rates.

Another feature of WS-Discovery that may possibly impact performance lies in its message structuring technologies. The WS-Discovery specification requires the use of XML, while mDNS typically uses DNS-SD. In [20], the authors show that a 'Hello' multicast message predominantly containing addressing information is sized at 1088 bytes, while with mDNS, packet captures from our Xen server showed mDNS query responses with frame sizes in the range of 300 bytes. Remedies are however possible, as [21] and [22] show that extensive savings in message sizes may be incurred if the EXI specification is applied to reduce the XML overhead of WS messages and possibly result in message sizes comparable with mDNS.

Without continuing with a granular comparison of the two specifications, it is already apparent that WS-Discovery, mDNS, and their companion technologies all target the same issues and, at times, using very similar techniques. However, differences do exist that also make it equally true that there is a clear need for a detailed evaluation and comparison of the two systems to create a distinct differentiation between them in terms of their delivered benefits and expected performance.

This is to determine the appropriateness of each as a solution for discovery in industrial environments.

VII. Conclusion

A basic service set for the non-RT coordination of an industrial, SOA-governed and survivable distributed GSB is presented. Composed of network, discovery and management services, the resulting SOA uses sockets for inter-service communication, P2P technologies for inter-node communication, and mDNS and DNS-SD for discovery. The implemented framework is tested using 33 consecutive runs on 11 VMs across two subnets. Results showed that the presented services and their associated mechanisms consistently allowed for the network-wide discovery of published services within a span of 3 seconds.

Further enhancements for the existing services are discussed, and include the use of service removal mechanisms, service aggregation and caching to reduce the impact of the proposed system on channel performance rates. However, based on the acquired results, the architecture is in its current state both stable and suitable for the coordination of the distributed GSB. The presented system may therefore, at this point, progress beyond the basic service set and incorporate advanced services for the acquisition, tunnelling, translation, storage, processing, and distribution of data. In the immediate sense this would involve the application of the distributed GSB using embedded devices to a didactic distributed PLC-based system and the development of modelling services, such as information, and information exchange modelling services, as well as the extension of the SOA with capabilities to support hard real-time tasks, as done in [23], to allow for the complete assimilation of industrial components. The totality of this system would be a comprehensive solution capable of resolving age-old and complex challenges to vertical integration, system reconfiguration, and distributed execution in heterogeneous industrial environments.

Acknowledgement

This paper is supported by TU Wien research funds.

References

[1] P. Adolphs, H. Bedenbender, et al. *Reference Architecture Model Industrie 4.0 (RAMI4.0)*. VDI/VDE Society Measurement and Automatic Control (GMA). July 2015.

[2] T. Sauter. "Linking Factory Floor and the Internet". In: *Industrial Communication Technology Handbook, Second Edition*. Ed. by Richard Zurawski. London: CRC Press, 2014. Chap. 22, pp. 1–24.

[3] H. Derhamy, J. Eliasson, et al. "Translation error handling for multi-protocol SOA systems". In: *IEEE ETFA*. Sept. 2015.

[4] T. Sauter, S. Soucek, et al. "Vertical Integration". In: *Industrial Communication Systems, Second Edition*. Ed. by B. Wilamowski and J. Irwin. London: CRC Press, 2011. Chap. 13, pp. 1–12.

[5] A. Ismail and W. Kastner. "Co-Operative Peer-to-Peer Systems for Industrial Middleware". In: *IEEE WFCS*. May 2016.

[6] A. Ismail and W. Kastner. "A Middleware Architecture for Vertical Integration". In: *IEEE CPPS*. Apr. 2016.

[7] M. García-Valls, I. Rodríguez-López, et al. "Towards a middleware architecture for deterministic reconfiguration of service-based networked applications". In: *IEEE ETFA*. Sept. 2010.

[8] S. Cheshire and M. Krochmal. *RFC 6762: Multicast DNS*. Internet Engineering Task Force (IETF). Feb. 2013.

[9] J. Konishi, N. Wakamiya, et al. "Proposal and Evaluation of a Cooperative Mechanism for Pure P2P File Sharing Networks". In: *BioADIT*. Jan. 2006.

[10] V. Ciancaglini. *From key-based to content-based routing: system interconnection and video streaming applications*. PhD Thesis, Université Nice-Sophia Antipolis, Oct. 2013.

[11] I. Stoica, D. Adkins, et al. "Internet indirection infrastructure". In: *IEEE/ACM Transactions on Networking* 12.2 (Apr. 2004), pp. 205–218.

[12] L. Cheng. "Bridging distributed hash tables in wireless ad-hoc networks". In: *IEEE GLOBECOM*. 2007.

[13] G. Hoang, L. Liquori, et al. "A backward-compatible protocol for inter-routing over heterogeneous overlay networks". In: *ACM SAC*. 2013.

[14] G. Hoang, L. Liquori, et al. "Backward-Compatible Cooperation of Heterogeneous P2P Systems". In: *Distributed Computing and Networking*. Springer, 2014.

[15] A. Datta and K. Aberer. "The challenges of merging two similar structured overlays: A tale of two networks". In: *Lecture notes in computer science* Vol.4124 (2006), p. 7.

[16] M. Kwon and S. Fahmy. "Synergy: an overlay internetworking architecture". In: *IEEE ICCN*. 2005.

[17] A. Colombo, T. Bangemann, et al. *Industrial Cloud-Based Cyber-Physical Systems*. Springer International Publishing, 2014.

[18] S. Iarovyi, J. Garcia, et al. "An approach for OSGi and DPWS interoperability: Bridging enterprise application with shop-floor". In: *IEEE ETFA*. 2013.

[19] B. Zhang, C. Postelnicu, et al. "An open energy consumption-relevant factory automation dataset in the cloud". In: *IEEE INDIN*. July 2012.

[20] L. Durkop, J. Imtiaz, et al. "Service-oriented architecture for the autoconfiguration of real-time Ethernet systems". In: *KommA'12*. Nov. 2012.

[21] F. Johnsen and T. Hafsøe. "Adapting WS-Discovery for use in tactical networks". In: *16th ICCRTS*. June 2011.

[22] R. Kyusakov, J. Eliasson, et al. "Efficient structured data processing for web service enabled shop floor devices". In: *IEEE ISIE*. June 2011.

[23] T. Kothmayr, A. Kemper, et al. "Machine ballets don't need conductors: Towards scheduling-based service choreographies in a real-time SOA for industrial automation". In: *IEEE ETFA*. Sept. 2014.