Co-Operative Peer-to-Peer Systems for Industrial Middleware

Ahmed Ismail, Wolfgang Kastner
Institute of Computer Aided Automation - Vienna University of Technology
Vienna, Austria
Email: {aismail, k}@auto.tuwien.ac.at
Telephone: +43 1 58801-18311

***Abstract*—With the ever-increasing emphasis on the importance of vertical integration for the fourth industrial revolution (Industrie 4.0), we contend that the largest potential lies in the replacement of single-purpose gateway-ing solutions with distributed gateway service bus (GSB) technologies. Within this paper, we primarily focus on the aspect of routing between variously grouped GSB components. We base our study on the premise that the most applicable technologies for such a scenario would be those of inter-system routing protocols from the domain of peer-to-peer (P2P) cooperative systems networking. We therefore summarize the basic principles governing such protocols and, using them, present a study that observes the conversion of a single system P2P networking protocol into an inter-system routing protocol. The final protocol is evaluated extensively using 50 Xen-based virtual machines deployed on 32-bit embedded devices and a 64-bit server.**

I. Introduction

Since their conception, Internet of Things (IoT) research tracks have become well-established academic fields in the particular branches of home automation, transportation, and energy sectors. As of recently, the IoT movement has set its sights on production systems as well, springing forth a vast array of collaborative efforts that aim to convert physical production systems into Cyber-Physical Production Systems (CPPS). The CPPS domain, also dubbed the Industrial Internet of Things (IIoT), envisions systems of computationally controlled physical elements, which is to usher forth the latest wave of disruptive technologies presenting new economic and technological opportunities for entire nations and markets.

However, at this point in time, enabling technologies are as of yet not fully present to allow for the complete integration of physical systems with the Internet. To tackle these limitations, the VDI/VDE Society Measurement and Automatic Control has put forth a Reference Architecture Model Industrie 4.0 (RAMI 4.0) describing the use of vertical, horizontal and lifecycle integration for the strategic advancement of industrial sectors towards the IIoT [1].

Focusing specifically on vertical integration, this cornerstone aims to facilitate the conversion of physical production systems into CPPS by providing solutions that overcome the connectivity limitations associated with currently deployed and future industrial automation systems. That is, due to the long life cycles of industrial systems, it is typical of control environments to operate using a mixture of legacy and non-interoperable protocols and devices. Ultimately, therefore, it is the objective of vertical integration schemes and technologies to manage such heterogeneity in favour of connectivity [2].

An established method for the management of system heterogeneity is through the use of gateways. Within the context of vertical integration, such gateways would perform the protocol translations or mappings necessary for the cross-layer communication of data required by modern IT techniques. However, we contend that gateways may be enhanced beyond being stand-alone single-purpose translation and mapping devices by having them actively participate in said IT techniques. A method for achieving this enhancement would be to have the gateways operate as components of a cooperative distributed system that is governed by a service oriented architecture (SOA). In such a system each gateway may be capable of hosting a myriad of intelligent services; for example, offering processing power, storage space or even high-powered autonomous self-X capabilities. In such a manner, the system would collectively be far more powerful than any one of its components and, as a distributed GSB, it would effectively be the industrial counterpart of the enterprise service bus (ESB) [3].

To exemplify the potential of the GSB, consider the setting depicted in Fig. 1. In this scenario, GSB components are used for distributed execution in a typical Ethernet-based plant composed of multiple manufacturing cells. Assume that each manufacturing cell zone operates using different Ethernet protocols and that all of the cells' respective mechatronic components are configurable using the IEC 61499 standard. As IEC 61499 typically dictates the use of clearly defined interfaces, each GSB node responsible for a set of mechatronic devices may create virtual device representations of all of the foreign objects that the local cell depends on. For example, the GSB node at manufacturing cell 1 would create representations of the devices of cell 3 that cell 1 requires for execution, and vice versa. Any services required to do so could be acquired from a distributed service repository that operates as a thin layer over all of the GSBs. For example, In Fig. 1, Layer 0's GSB 'A' may negotiate the replication of the required 'Configuration' service from layer 3's GSB 'C'. Once all of the appropriate services are acquired and executed, each GSB node may then perform wire-speed translations to the various device protocols involved in the executing process. Finally, the resulting data is transferred to the foreign GSB node

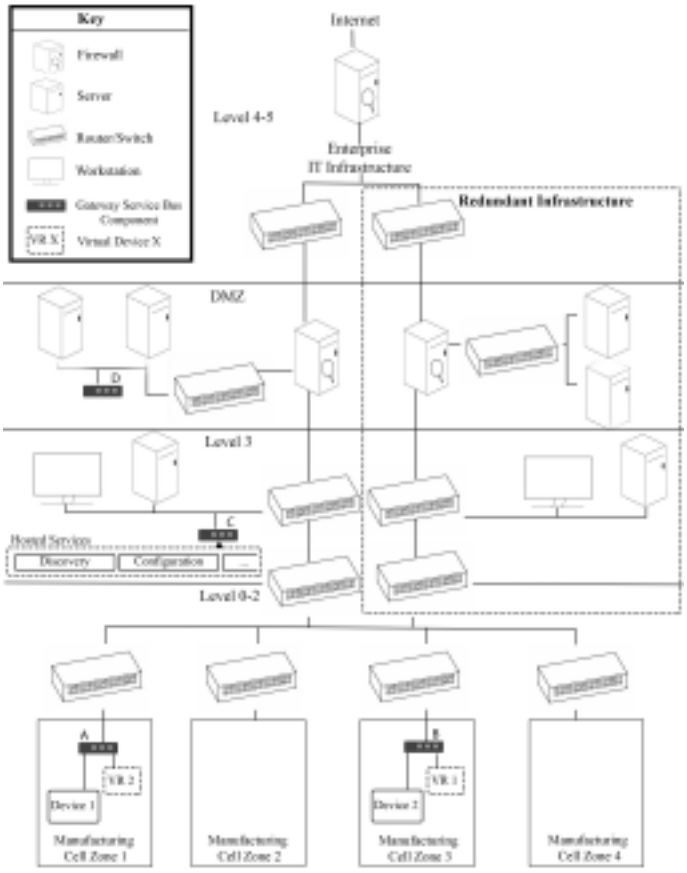

Fig. 1. The deployment of a distributed GSB over a typical Ethernet-based industrial enterprise network [4]

and subsequently to the destination device. In so doing, the distributed GSB has allowed for distributed execution and for the simplified reconfiguration of heterogeneous manufacturing cells.

The scenario does not end here, however, as more benefits may be extracted from the distributed GSB. To elaborate, as the GSB nodes are immediately intercepting traffic between the devices, the acquired data may be transferred to other GSB nodes with enhanced processing capabilities. Such nodes may then cooperatively analyse the data, for example, to detect anomalies in production. The processed data can then be served in real-time to engineering workstations, or, from the DMZ layer, to further functions running at the enterprise layer.

In total, the aforementioned scenario prominently demonstrates how multiple business-enhancing services are possible once gateways are elevated from being single-purpose, standalone devices, to operating as parts of a distributed, SOA-governed GSB.

However, achieving the aforementioned distributed GSB first requires the establishment of reliable coordination mechanisms between the various gateway devices. These mechanisms need to be designed with special consideration to the infrastructure of industrial enterprises. This is as industrial networking infrastructure is typically engineered based on numerous standards and binding legal constraints that may not be violated for the sake of connectivity [5]. Generally, these constraints translate to a network design that follows a hierarchical and layered architecture, as is shown in Fig. 1, with strict controls applied to communication flows between said layers, while intra-layer communication is permitted to flow freely [4]. Superimposed upon the inter-GSB communication protocol, this means that the protocol should be able to mirror and maintain fidelity to such an architecture, while also having the flexibility to autonomously adapt to changing system requirements. The resulting network would therefore effectively be an 'overlay network'; a network that constructs itself upon existing physical infrastructure [6].

It is the concern of this paper to tackle this first stage of designing an industrial SOA by defining reliable coordination mechanisms for an overlay network of GSB devices. To achieve this, the paper is structured as follows. Section II discusses the potential and the shortcomings of various possible networking solutions from the domain of overlay networks and justifies the selection of P2P networks as cooperative systems as a suitable solution for the task at hand. Using the principles of the selected subdomain, Section III exemplifies the design of a GSB routing protocol by detailing the process of creating a cooperative systems P2P protocol. Section IV tests the derived protocol; here, both the experimental procedure applied and the test results attained are detailed and discussed. Finally, Section V draws the necessary conclusions from the study and ends by presenting recommendations for future work.

II. Related Work

A. An Appropriate Networking Domain

Of the various subclasses of overlay networking that exist, one of the most well-developed and popular is that of P2P networks. Briefly, P2P networks are ones in which a number of devices are connected together and share resources using direct exchange in a manner that is resilient against failures and transient population sizes, all without the use of a central manager [7]. In order to do so, P2P networks extend on the definition of overlay networks using a number of well-defined features. Such attributes, listed in Table I are elements that match perfectly with the requirements of a SOA-based GSB for industrial systems. For example, features such as services management and scheduling would be necessary for an efficient SOA, while others such as resiliency and fault tolerance enhance the finalized distributed GSB's suitability for industrial networks [8].

Of the various types of P2P networks possible, the one that is decidedly the most suitable for the task at hand is that of P2P networks as cooperative systems. This is as this class of networks concerns itself with the bridging or merging together of existing P2P networks to allow for expanded systems of networks, inter-system content-sharing, and inter-system traffic engineering [9]. Using the concepts of this domain, GSB devices may therefore be capable of organizing themselves into systems of cooperative and distributed GSBs. Each layer of the enterprise, for example, may comprise of one or more complete and self-contained overlay networks.

TABLE I
THE TYPICAL ARCHITECTURE OF P2P NETWORKS [10]

Layer	Properties
Application-level	Applications
	Tools
	Services
Services-specific	Meta-data
	Services management
	Services scheduling
	Services messaging
Features management	Security management
	Resource management
	Reliability and fault resiliency
Overlay nodes management	Routing and location lookup
	Resources discovery
Network communications	Network

Using the concepts of cooperative systems, new systems may be dynamically formed, as necessary, to bridge together GSB nodes from the different overlays of the enterprise to offer new services, or to execute new functions and processes. Consequently, by allowing for the dynamic reconfiguration of systems, this subdomain may therefore be used to facilitate or restrict message and content transfer between overlays and enterprise layers dynamically, and possible autonomously, in accordance with the enterprise's changing requirements. Such a subdomain therefore allows the proposed distributed GSB to establish pervasive enterprise-wide communication channels without violating or compromising the binding constraints of the industrial architecture.

B. The Principles of Cooperative Systems

Establishing reliable routing mechanisms for a distributed GSB operating using the principles of P2P networks as co-operative systems directly translates to the need for a reliable inter-system routing policy. Unfortunately, work in this domain is scant; it is however possible to derive the fact that, for enhanced reliability, solutions have converged towards the use of either co-located nodes or gateways. The difference between the two strategies, as defined in [11], lies in the fact that co-located nodes participate in the routing process, while gateways only maintain pointers towards specific nodes in different overlays. Works that representatively exemplify these principles are [12] for the gateway-ing concepts and [13] for the co-located nodes method.

In the case of [12], inter-system routing between fundamentally different classes of P2P networks is achieved by having a subset of nodes in an overlay network act as gateways, and then having a second group of nodes, known as the gateway pointers, keep track of these gateways. Nodes are designated gateways if they exist in two networks, physically or otherwise. To traverse across these differing networks, queries go from the origin node to a gateway pointer, then the gateway, and finally to the external network. Unfortunately, the system reported in [12] was not evaluated using simulations or experimentation.

As for [13], a protocol based on co-located nodes named Babelchord is presented. Here, Babelchord is defined as a protocol for bridging together different Chord networks; a type of P2P networks that assign and uses m-bit length identifiers to map peers on to a circle modulo 2^m, known as a chord ring, for routing and other purposes. The mechanisms of Babelchord rely on the use of Synapse nodes, which are peers that belong to multiple rings, or floors, at once. Any query that passes through a Synapse node is forwarded to all of the floors that the node is a part of thereby achieving inter-system behaviour. However, as queries are not guaranteed to reach a Synapse node, this form of routing is defined as "opportunistic routing" [11].

Although Babelchord was extended further through multiple studies, the most relevant study to our current needs is [11]. This is as [11] shows that Babelchord's opportunistic routing benefits greatly once gateway pointers are included and used. Specifically, [11] extends the Synapse protocol with such pointers, creating the Synapse 2.0 Framework, and shows that in so doing so they have allowed for the definition of routing policies that ultimately result in a more flexible system.

Based on the results of the aforementioned studies, it appears that regardless of whether an exclusively gateway-based or co-located-nodes-based method is used, the basic components of an optimal inter-system routing protocol is that of nodes, gateways and gateway pointers. The first step in developing an inter-system routing protocol should therefore be instilling the modifications necessary to allow for the inclusion of these three elements.

To exemplify these principles, Section III will detail their usage in developing a functional cooperative systems P2P protocol. However, due to the limited scope of the study and the large number of existing and mature P2P solutions based on proven methods, an existing single-system solutions, Chimera, is used as a starting point for the development process. The selection of Chimera as the illustrative protocol to undergo the conversion process is mainly due to it being a hybrid implementation of two well established P2P protocols; these being the Pastry and Tapestry protocols. Its hybrid nature means that it has inherited a favourable number of properties and constituent elements that it directly shares with several other largely popular P2P protocols. This lends credibility to the applicability of the methods of Section III for the conversion of other existing P2P protocols into cooperative systems ones as well.

III. SYSTEM DESIGN

This section details the design process in two parts. The first subsection describes the Chimera protocol in its original form, while the second delves into the conversion process that transforms Chimera into an inter-system routing protocol.

A. Base Protocol & Modifications

This subsection will explain the primary elements of Chimera in order to establish its baseline behaviour. These

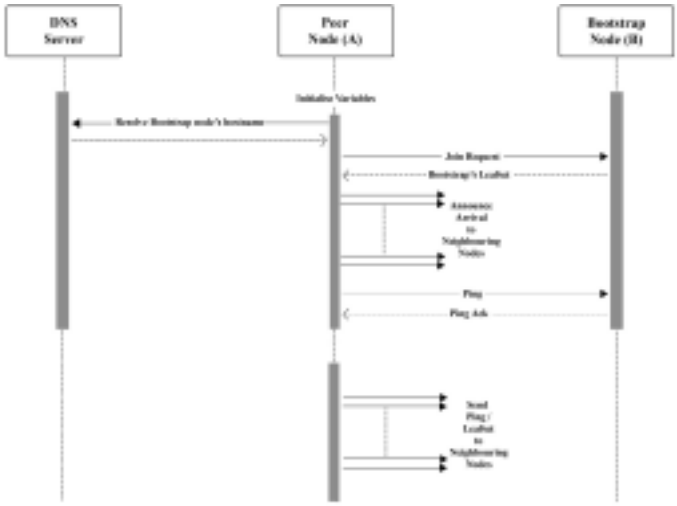

Fig. 2. The operational mechanisms of the vanilla Chimera P2P protocol

elements may be summed up as being bootstrap nodes, prefix-based routing, fault detection through the use of heartbeat messages, neighbour sets and leaf sets. Due to a lack of publications on Chimera, the descriptions provided are inferred from a library implementation in C that was developed at the University of California, Santa Barbara[1].

To begin with, the definition of each of the aforementioned elements of Chimera is as follows. Firstly, a bootstrap node is a member of a P2P network that supplies newly joining nodes with the initial configurations required for them to successfully join the network. Next, prefix-based routing is a form of message routing that uses the unique identifiers, or keys, of nodes in routing. In prefix-based routing, the next hop selected for a message is the one that matches the prefix of the destination with a digit extra than the current hop. As for fault detection through pings, or heartbeat messages, this mechanism relies on the successful acknowledgement of pings between nodes to surmise that other peers are functional. A node's neighbour set, on the other hand, consists of information on a number of peers that are closest to it in terms of proximity. This set is not typically used for routing, but is meant to be used as a source of locality information. Finally, the leaf set of a peer, L, is composed of the nodes that are $|L/2|$ numerically closest larger, and $|L/2|$ numerically smaller keys, as compared to the peer's own key. Unlike the neighbour set, the leaf set is in fact used for message routing [10, 14].

Together, the aforementioned mechanisms operate as follows. Initially, Chimera must be supplied with the details of a bootstrap node or be initiated as the bootstrap node. If it is initiated as the bootstrap node, then it simply waits for join requests. If it is supplied with the host name, port and key (a unique bit string identifier) of a bootstrap node, then, as shown in Fig. 2, the joining node, A, transmits a join request containing its host name, port, and key to the bootstrap node, B, and waits indefinitely for a response. If node B accepts the join request, node B sends node A its leaf set, which is processed by Node A and subsequently used to send an update message to each of the members of the leaf set announcing its arrival.

Once the join process is complete, the network is then maintained through the use of heartbeat messages and what is termed as piggyback messages. To elaborate, every pre-set length of time a node is expected to ping every other node in its leaf set. If the pinged node responds with an acknowledgement within a pre-assigned grace period, the node is acknowledged as a functional one. If, however, the node does not respond in time and its success average is found to be below an acceptable threshold, the node is pruned out of the routing table. As for the piggyback messages, these are communicated after every third ping. This message type contains the entire leaf set of the node, and is communicated to every member of its leaf set in order to disseminate routing table updates.

Further details of the Chimera implementation relate to its external and internal host lookup algorithms. For external host lookups Chimera is wholly dependent on DNS lookups for host resolution. The resolved IPs of hosts are not stored, and lookups are performed before every transmission. The remaining information related to peers, such as host names, keys, and ports, to name but a few, are stored using the Red-Black binary tree library. As such, these binary trees are used for subsequent lookups of information on hosts, as well as for other implementation-specific behaviour relevant to the proper functioning of the program itself.

All of the aforementioned communication occurs solely through the use of UDP packets. The structure given to said UDP packets is shown in Fig. 3. Of the header elements listed, the seqNum and source key fields are not put to use by the original authors of Chimera. That is, the message header struct includes these fields, however, they are not included or used in any transmitted or received packets.

Some of the limitations of Chimera include the fact that support was not extended for TCP communication at all. Furthermore, aside from the provision of UDP-based message transferring mechanisms, no functions were defined for resource lookups, file transfers, or any other cooperative behaviour typical of P2P networks.

B. Modifications

As the purpose of modifying Chimera is to instil the necessary measures to allow for inter-system routing, the first requirement is therefore a method by which overlays may be differentiated from one another. To do so, we mirror the same identification method used to differentiate nodes from each other. That is, as each node is given a unique key, each overlay is also assigned a key. Although a simple change, this modification echoes throughout the routing, messaging, and higher level functions and layers.

Starting with the messages themselves; a message must now be clearly labelled with its destination overlay. Consequently the message header is modified to include this key as well. Furthermore, we also enable the source key and sequence number header fields and add a source overlay key field as

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

Fig. 3. The structure of a UDP Packet in the original Chimera implementation (above), and post-modifications (below).

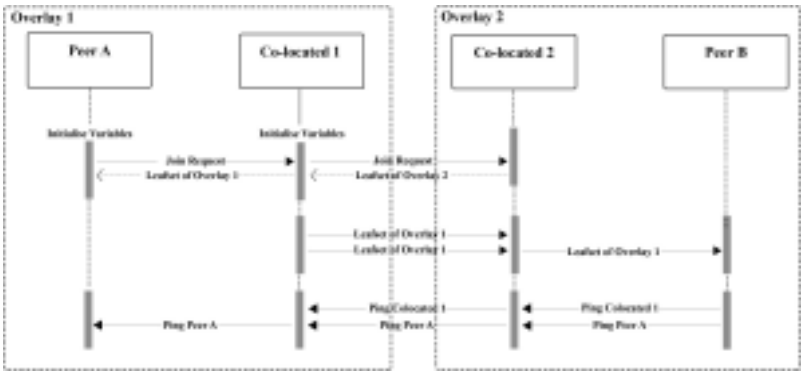

Fig. 4. The operational mechanisms of Chimera after modifications.

well. These three fields are included for reasons that will be clear further on in this subsection. The resulting message header is shown in Fig. 3.

As for the routing process, naturally, the first step is to define a third node type aside from the peer and bootstrap roles that allows a node to operate as a gateway. This role is denoted as the 'co-located' node type and is typically considered to be fixed. This is as, in an industrial setting, nodes that would be responsible for cross-layer communication will need to be granted the proper access rights to do so. This may mean that such gateways may need several of various types of communication interfaces, additional wiring, multi-homing, or the configuration of firewalls and other devices to allow for the traversal of gateway-specific traffic. Since only a number of GSB devices may be afforded the configurations or hardware necessary to operate under these constraints, the P2P protocol must therefore be able to accommodate the use of these nodes as fixed gateways.

Currently, the role of a co-located node is designed to be occupied exclusively by bootstrap nodes. That is, a co-located node is always a bootstrap node, but a bootstrap node does not always have to be a co-located node. As all nodes must know the bootstrap node to join an overlay, and Chimera operates by using fixed bootstrap nodes, by doubling the role of a bootstrap node as a gateway node, all of the members of each overlay become de facto gateway pointers. This reduces the complexity of the protocol by eliminating the need for gateway pointers selection and gateway discovery processes.

With all of the aforementioned modifications in mind, the message routing process is, naturally, modified as well. The first thing to note is that, due to the geographic proximity of peers, prefix-based routing is abandoned. Instead, if a message is destined to a node in the same overlay as the sender, then the message is sent directly towards the destination. However, if the message is sent to a node in a different overlay, then, as shown in Fig. 4, the message is routed via the co-located node with the destination key and destination overlay key clearly labelled. If the bootstrap node contains a listing of a co-located node in the destination overlay, which in this case would be a node that has joined its overlay from the destination overlay or vice versa, then the bootstrap node forwards the message to the co-located node. In turn, the co-located node then forwards the messages towards its final destination. If no co-located node in the destination overlay is available, then the bootstrap node transmits the message to a co-located node from a randomly selected overlay in the hopes that it may have a path to the destination overlay. If the bootstrap node is not aware of any active co-located nodes, then the message is dropped.

As previously mentioned, the message header is modified to actively use the message layer sequence number, source key and source overlay key. The source key and overlay key are used for two purposes. The first is to instil a loop-breaking mechanism which was not originally present in Chimera. That is, we prevent the next hop of a message from being the previous hop to avoid routing loops. The second purpose is to simplify the experimental analysis process. This is as, by using the source key, overlay key, and sequence number, we are able to trace back messages from the destination to the original source allowing us to calculate metrics such as the number of hops travelled per message. Beyond the experimental value of including these three fields, they may also serve to enhance the security of the protocol. This was their intended purpose in the original implementation of Chimera, but, like vanilla Chimera, they are currently still not used for security purposes.

Further routing-related changes involve the modification of the purpose of leaf sets and neighbour sets. As prefix-based routing is not used, leaf sets are therefore instead used as overlay sets. That is, each leaf set maintained by a node in fact corresponds to an overlay. Consequently, as shown in Fig. 4, the pinging and piggyback messages have also been modified

to accommodate for this change. The pinging function now operates by having a node select a leaf set at random before pinging all of the nodes in that overlay. As for the piggyback messages, like the pinging function, the destination leaf set of the piggyback message is chosen at random. However, the contents of the piggyback message is always information on the nodes in the transmitting node's own overlay. Lastly, the second type of sets, neighbour sets, are at present, not put to use; however, as they may later on prove to be useful for the execution of other functions, such as load distribution and data replication, for example, they are kept as vestiges within the current implementation.

The final modifications done to Chimera are related to increasing its efficiency and flexibility. For the former, we eliminate Chimera's dependence on DNS-based host name resolution and instead directly store and share information on host IPs through piggyback messages. As for the latter, further flexibility, this is achieved through two measures. The first measure of these is the modification of all the functions responsible for network communication to allow both 32 bit and 64 bit platforms to execute Chimera. This is to allow the GSB to use computing resources from both ends of the spectrum of available devices. The second modification is the removal of Chimera's dependence on the Red-Black Tree library. Instead, a generic hash table is instilled that may be easily extended to use a Red-Black Tree, a NedTrie or any other digital searching mechanism that developers may desire, based on their respective requirements.

With the behaviour of the protocol outlined, the next section delves into the experimental procedure implemented and the results collected in order to evaluate the designed inter-system routing mechanisms.

IV. Experimental Results

To reiterate on previous points discussed, the protocol presented is designed to allow for the reliable inter-system routing of messages between clusters of GSB nodes. This section is therefore concerned with testing the protocol's abilities and limitations at inter-system routing. These tests are performed while observing the degree of variation in the protocol's behaviour as the number of clusters and nodes per cluster are modified to establish a relationship between the two.

A. Results

The experiment itself is carried out on a Xen Project (TM) server hosting 48 Debian virtual machines (VMs). Two further VMs running Ubuntu are deployed separately on two 32-bit CubieTruck ARM boards that were also extended using the Xen platform. The physical network configuration therefore consists of two CubieTrucks connected to a single network port on the server using an unmanaged switch. The port is then bridged to 48 virtual interfaces, with each interface belonging to one of the 48 VMs.

As for the topologies deployed, the number of clusters and nodes per cluster are varied to establish a relationship between the two variables. As such, a fixed number of nodes, 50 in all, are divided into 1, 2, 5, 10 and 25 clusters with each cluster consisting of 50, 25, 10, 5, and 2 nodes, respectively. Clusters are connected linearly via their bootstrap nodes as shown in Fig. 5.

The configuration and execution of these topologies is carried out by A BASH script running on the server's dom0. The script transfers the Chimera application with the necessary bootstrapping configuration files to all 50 nodes before each experiment. Once the transfers are complete, the same script executes all of the instances sequentially, with a gap of 2 seconds between each execution. This gap is in place to give each instance enough time to start up before it may begin receiving pre-configured messages, such as join requests, from any subsequently executed nodes. It is important to note as well that, as a precaution, the Chimera instances are always run under the GNU debugger (GDB). Finally, each experiment is run for an hour; during this time, each node collects and records information related to the behaviour of its network, routing, messaging and application layers. Once the experiment timer expires, the same BASH script terminates Chimera on all 50 nodes, retrieves and archives all of the relevant logs, and prepares the node for the next experimental run.

The acquired logs are analysed in Matlab to evaluate the performance of the protocol. For each node, the total number of messages sent, acknowledged, received, and rerouted are calculated. Messages are also traced backwards, from destination to origin, to compute the number of hops travelled by every unique message transmitted during the experiment. The results are aggregated by cluster and summarized in the forms of means, medians, and 95^{th} percentiles. Since each of these forms of descriptive statistics may extensively vary from one node to the next, the maximum, minimum, and median values of each is also computed to provide an objective representation of the results. All of the aforementioned results are shown in Tables II, III, IV, and V. It is important to note at this point that the case of having 1 cluster serves the purpose of orienting the reader with the baseline behaviour of the protocol as no inter-system routing is possible in this scenario.

What is immediately apparent from Tables II-V is that the number of messages transmitted and received follows a crude bell-curve shape. It appears that the maximum number is achieved at the mid range, where the number of clusters is half the number of nodes per cluster; although the case of having 10 clusters, i.e. the nodes per cluster is half the number of clusters, trails closely behind. It appears that these higher recorded numbers of messages is due to two properties related to these clusters' configurations. The first is that there are enough nodes per cluster to allow for a healthy number of intra-routed messages. Second, the number of clusters is manageable enough to allow for inter-cluster discovery within the period of one hour. This is apparent from Table V, where the 5 cluster and 10 cluster cases are the only ones with means, medians and 95^{th} percentiles reflecting the dominance of multi-hopped messages in their experiment runs.

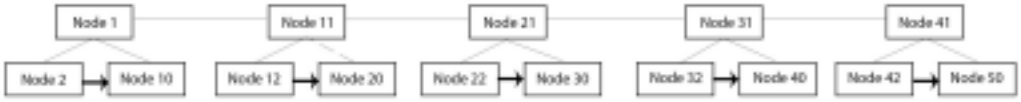

Fig. 5. The structure of a 5 cluster network.

TABLE II
THE NUMBER OF MESSAGES SENT PER NODE

	Messages Sent								
	Mean			Median			95th Percentile		
# of Clusters	Min	Med	Max	Min	Med	Max	Min	Med	Max
1	2180	2180	2208	2166	2185	2174	2295	2312	2833
2	1848	1859	1876	1669	1683	1820	2349	2465	3033
5	1438	1642	2226	1222	1424	1844	3241	4243	5440
10	1368	1396	1777	1304	1368	1466	2047	2557	5294
25	621	698	750	502	724	812	1018	1109	1153

TABLE III
THE NUMBER OF ACKNOWLEDGEMENTS RECEIVED PER NODE

	Acknowledgements Received								
	Mean			Median			95th Percentile		
# of Clusters	Min	Med	Max	Min	Med	Max	Min	Med	Max
1	2080	2121	2126	2079	2097	2129	2202	2207	2688
2	1669	1701	1737	1477	1556	1642	2068	2367	2643
5	1378	1583	2182	1165	1384	1802	3108	4201	5313
10	1344	1372	1740	1274	1345	1433	2039	2488	5213
25	611	692	740	493	723	809	1005	1092	1133

TABLE IV
THE NUMBER OF ACKNOWLEDGEMENTS SENT PER NODE

	Acknowledgements Sent								
	Mean			Median			95th Percentile		
# of Clusters	Min	Med	Max	Min	Med	Max	Min	Med	Max
1	2093	2126	2132	2100	2137	2138	2276	2290	2349
2	1673	1707	1741	755	995	1042	853	1105	1162
5	1384	1588	2186	392	436	462	9897	11757	16699
10	1348	1374	1433	274	286	341	5775	6386	7186
25	613	692	741	370	378	474	1292	1462	1515

TABLE V
THE NUMBER OF HOPS PER MESSAGE

	Number of Hops								
	Mean			Median			95th Percentile		
# of Clusters	Min	Med	Max	Min	Med	Max	Min	Med	Max
1	1	1	1	1	1	1	1	1	1
2	1	1	1	1	1	1	1	1	1
5	1	1	2	1	1	2	2	2	2
10	1	2	2	1	2	2	2	2	3
25	2	2	2	2	2	2	2	2	2

Contrasting with these results is the highly granular case of having 25 clusters of 2 nodes each. Here, the number of transmissions are markedly lower than has been observed in the other experiments. This is because each leaf set is so small in size that a ping run on a leaf set results in only a few message transmissions before the source node rests and waits for the initiation of the next ping run. Although this means that the overhead of pings on the network is lower, it also means that in a fixed set of nodes, the discovery of faulty peers, and the subsequent pruning of their information, is also slow. Furthermore, as is apparent from the mean, median and 95th percentile of the number of hops travelled by the messages of this experiment, the inter-cluster discovery process also suffers with this large number of small clusters. As shown in Table V, the value at 25 clusters is consistently 2 hops, meaning that, within the time span of an hour, it was rare for a cluster to be able to discover clusters beyond its immediate neighbours. This is also a reason for the lower total transmissions observed for this configuration as messages did not have to travel far to reach their final destinations.

A last observation is to do with the discrepancies seen between the number of messages sent and the numbers acknowledged. Although the protocol performed formidably, none of the experiments achieved a perfect record. This is due to two possible explanations. The first is that once the hour-long experiment is complete, the termination of all instances is not, unfortunately, instantaneous. This leads to the loss of acknowledgements, as they are either not sent or not received by a terminated application. Furthermore, at certain points multiple nodes try to communicate with the same one, for example, a co-located node. This led to collisions and the loss of messages, or acknowledgements. This is especially true for the cases of 2, 5, and 10 clusters, where co-located nodes were consistently under exceptionally heavy loads.

B. Discussion

Based on the results of the previous subsection, it is safe to conclude that having the number of nodes be double the number of clusters, or vice versa, is the best configuration to have for the processes in place. Unfortunately, even with this configuration, the discovery process is considerably slow. The procedure of having to send hefty piggyback messages containing the information of all of the nodes of a leaf set, in every sense, requires replacement with a quicker and more robust solution.

The inter-system pings, although unnecessary for the maintenance of whole networks, was introduced to simulate the inter-system routing of messages. The protocol and implementation is agile and stable enough to predominantly allow their messages to reach their destinations without fail, and consistently using the shortest path possible. Although this would allow designers to set conservative TTLs to messages, it comes at the cost of bootstrap nodes being considerably loaded with the task of re-routing messages throughout its operational lifetime. This brings to question the matter of whether such nodes would be able to participate in any other tasks required of a GSB node beyond routing. A further observation is that, naturally, due to these high loads, having star-shaped clusters is completely out of the question and should not be pursued. What ought to be considered, however, are questions of redundancy and load balancing, both to deal with and to offset the inevitable failures that would result from the heavy routing loads that co-located nodes must deal with. Finally, another desirable consideration is that of having mechanisms in place to allow for a mesh configuration between the co-

located nodes themselves. This is in favour to the linear routing currently in place in the hopes of reducing their routing loads as well as message latencies.

Finally, it would not be fair to conclude without discussing the fact that this experiment did not run without its own number of limitations. The gaps between initiation and termination naturally mean that all experiments did not run for an exact hour to the second; however, the effect of this limitation has been noted in the previous subsection. Other limitations include the fact that the experiment was only run for an hour each time, and it would be interesting to observe, for example the steady state behaviour of the bootstrap node in the 2 or 25 cluster network after several hours, once wider knowledge of further clusters is established, and inter-cluster routing plays a much more dominant role.

V. Conclusion

An inter-system routing protocol is built based on the constraints of developing for an industrial, SOA-governed, distributed GSB. The system was subsequently deployed on 50 VMs on 64-bit and 32-bit platforms. Extensive testing was performed in order to determine the performance of the protocol and the optimal configurations for the organization of Chimera peers. The number of clusters and their respective sizes were varied and a number of metrics were collected, analysed and compared.

Based on the results attained, it is reasonable to conclude that the protocol is stable and meets its constraints. As such, the protocol may, in its current form, be integrated into a distributed GSB to participate in the coordination of advanced IIoT services. Such services would typically be expected to tackle age-old problems associated with production and industrial systems. Two formidable examples given in Section I include the use of data analytics for the detection of anomalies during production, and the use of virtual device representations for managing system heterogeneity.

However, it is apparent from the constituencies of this P2P approach that provisioned services may go beyond the resolution of existing problems. For example, there is no limitation that would prevent the evolution of this approach towards further integration with the Internet. Effectively, this would allow the GSB to incorporate advanced services that rely heavily on the Internet. Such services may include the offshoring of functions to cloud-based systems for reduced operational costs, or the enhancement of plant safety through the employment of the event web; a form of event modelling which combines information from the Web with additional sources to infer states and events. Such potential makes it prudent to claim that, although the protocol is quite capable as it is, further research could result in core mechanisms that extend it beyond its current practical limitations.

Acknowledgement

This paper is supported by TU Wien research funds.

References

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

[2] B.M. Wilamowski and J.D. Irwin. *Industrial Communication Systems*. The Industrial Electronics Handbook. Taylor & Francis, 2011.

[3] P. Gaj, A. Malinowski, et al. "Guest editorial: Distributed data processing in industrial applications". In: *IEEE Transactions on Industrial Informatics* 11.3 (June 2015), pp. 737–740.

[4] Paul Didier, Fernando Macias, et al. *Converged Plantwide Ethernet (CPwE) Design and Implementation Guide*. Cisco Systems and Rockwell Automation. 2011.

[5] J. Zerbst, S. Zimmermann, et al. "Towards an Adapted Classification Methodology for Graded Security Approaches in EPU Architectures". In: *Proceedings of the International Council on Large Electric Systems (CIGRE)*. Apr. 2013.

[6] Z. Ou. "Structured peer-to-peer networks: Hierarchical architecture and performance evaluation." PhD thesis. University, 2010.

[7] Stephanos Androutsellis-Theotokis and Diomidis Spinellis. "A survey of peer-to-peer content distribution technologies". In: *ACM Computing Surveys (CSUR)* 36.4 (2004), pp. 335–371.

[8] V. Altmann, J. Skodzik, et al. "A DHT-Based Scalable Approach for Device and Service Discovery". In: *12th IEEE International Conference on Embedded and Ubiquitous Computing (EUC)*. Aug. 2014, pp. 97–103.

[9] Hoang Giang Ngo. "From inter-connecting P2P overlays to co-operating P2P systems". PhD thesis. Université Nice Sophia Antipolis; Hanoi University of sciences, 2013.

[10] Eng Keong Lua, Jon Crowcroft, et al. "A survey and comparison of peer-to-peer overlay network schemes". In: *Communications Surveys & Tutorials, IEEE* 7.2 (2005), pp. 72–93.

[11] Vincenzo Ciancaglini. "From key-based to content-based routing: system interconnection and video streaming applications". PhD thesis. Université de Nice - Sophia Antipolis, 2013.

[12] Lawrence Cheng. "Bridging distributed hash tables in wireless ad-hoc networks". In: IEEE, 2007, pp. 5159–5163.

[13] L. Liquori, C. Tedeschi, et al. "Babelchord: a social tower of DHT-based overlay networks". In: *IEEE Symposium on Computers and Communications (ISCC)*. July 2009, pp. 307–312.

[14] Antony Rowstron and Peter Druschel. "Pastry: Scalable, decentralized object location, and routing for large-scale peer-to-peer systems". In: *Middleware 2001*. Springer, 2001, pp. 329–350.