

Dear Author,

Here are the proofs of your article.

- You can submit your corrections **online**, via **e-mail** or by **fax**.
- For **online** submission please insert your corrections in the online correction form. Always indicate the line number to which the correction refers.
- You can also insert your corrections in the proof PDF and **email** the annotated PDF.
- For fax submission, please ensure that your corrections are clearly legible. Use a fine black pen and write the correction in the margin, not too close to the edge of the page.
- Remember to note the **journal title**, **article number**, and **your name** when sending your response via e-mail or fax.
- **Check** the metadata sheet to make sure that the header information, especially author names and the corresponding affiliations are correctly shown.
- **Check** the questions that may have arisen during copy editing and insert your answers/corrections.
- **Check** that the text is complete and that all figures, tables and their legends are included. Also check the accuracy of special characters, equations, and electronic supplementary material if applicable. If necessary refer to the *Edited manuscript*.
- The publication of inaccurate data such as dosages and units can have serious consequences. Please take particular care that all such details are correct.
- Please **do not** make changes that involve only matters of style. We have generally introduced forms that follow the journal's style. Substantial changes in content, e.g., new results, corrected values, title and authorship are not allowed without the approval of the responsible editor. In such a case, please contact the Editorial Office and return his/her consent together with the proof.
- If we do not receive your corrections **within 48 hours**, we will send you a reminder.
- Your article will be published **Online First** approximately one week after receipt of your corrected proofs. This is the **official first publication** citable with the DOI. **Further changes are, therefore, not possible.**
- The **printed version** will follow in a forthcoming issue.

Please note

After online publication, subscribers (personal/institutional) to this journal will have access to the complete article via the DOI using the URL: [http://dx.doi.org/\[DOI\]](http://dx.doi.org/[DOI]).

If you would like to know when your article has been published online, take advantage of our free alert service. For registration and further information go to: <http://www.link.springer.com>.

Due to the electronic nature of the procedure, the manuscript and the original figures will only be returned to you on special request. When you return your corrections, please inform us if you would like to have these documents returned.

Metadata of the article that will be visualized in OnlineFirst

Please note: Images will appear in color online but will be printed in black and white.

ArticleTitle	Inter-Cloud Communication Through Gatekeepers to Support IoT Service Interaction in the Arrowhead Framework	
Article Sub-Title		
Article CopyRight	Springer Science+Business Media New York (This will be the copyright line in the final PDF)	
Journal Name	Wireless Personal Communications	
Corresponding Author	Family Name	Varga
	Particle	
	Given Name	Pál
	Suffix	
	Division	Department of Telecommunications and Media Informatics
	Organization	Budapest University of Technology and Economics
	Address	2 Magyar Tudósok krt., Budapest, 1117, Hungary
	Phone	
	Fax	
	Email	pvara@tmit.bme.hu
	URL	
	ORCID	
Author	Family Name	Hegedűs
	Particle	
	Given Name	Csaba
	Suffix	
	Division	Telecommunication Division
	Organization	AITIA International Inc.
	Address	48-50 Csetz János str., Budapest, 1039, Hungary
	Phone	
	Fax	
	Email	hegeduscs@aitia.ai
	URL	
	ORCID	
Schedule	Received	
	Revised	
	Accepted	
Abstract	Although we live in the era of Internet of Things (IoT), many issues on system-interoperability are still not tackled at large. There are limitations for ever-growing IoT realms, especially in resource-constrained domains, where low latency is crucial, and local processing power is a scarce resource. Service Oriented Architectures (SOA) provide functional and configuration flexibility in closed communication environments, where security and service-related orchestration issues are controlled within the local network. For automation systems, these SOA-based networks can have core services, such as Service Registry, Orchestration, Authorization, and so on. A set of such core services are defined, implemented and made available through the Arrowhead Framework. Since the Core Services are distributed resources	

available for all systems that wish to consume them, these networked services can be considered a cloud. As one cloud cannot serve for all, there is a need for such automation system clouds to interact with each other: use the services of one from another. This paper presents a solution for providing inter-Cloud servicing capabilities in the Arrowhead framework by introducing a gatekeeper concept. The main idea is to extend the service discovery functionality outside the boundaries of a single cloud, and solve the security and orchestration issues in a way that fits into the general Arrowhead concept. This paper also introduces the methodology of creating secure connections between service consumers and providers situated in different clouds.

Keywords (separated by '-') Internet of Things - Local Automation Clouds - Orchestration - Service discovery - System of Systems

Footnote Information



Inter-Cloud Communication Through Gatekeepers to Support IoT Service Interaction in the Arrowhead Framework

Pál Varga¹ · Csaba Hegedűs²

© Springer Science+Business Media New York 2017

Abstract Although we live in the era of Internet of Things (IoT), many issues on system-interoperability are still not tackled at large. There are limitations for ever-growing IoT realms, especially in resource-constrained domains, where low latency is crucial, and local processing power is a scarce resource. Service Oriented Architectures (SOA) provide functional and configuration flexibility in closed communication environments, where security and service-related orchestration issues are controlled within the local network. For automation systems, these SOA-based networks can have core services, such as Service Registry, Orchestration, Authorization, and so on. A set of such core services are defined, implemented and made available through the Arrowhead Framework. Since the Core Services are distributed resources available for all systems that wish to consume them, these networked services can be considered a cloud. As one cloud cannot serve for all, there is a need for such automation system clouds to interact with each other: use the services of one from another. This paper presents a solution for providing inter-Cloud servicing capabilities in the Arrowhead framework by introducing a gatekeeper concept. The main idea is to extend the service discovery functionality outside the boundaries of a single cloud, and solve the security and orchestration issues in a way that fits into the general Arrowhead concept. This paper also introduces the methodology of creating secure connections between service consumers and providers situated in different clouds.

Keywords Internet of Things · Local Automation Clouds · Orchestration · Service discovery · System of Systems

✉ Pál Varga
pvarga@tmit.bme.hu

Csaba Hegedűs
hegeduses@aitia.ai

¹ Department of Telecommunications and Media Informatics, Budapest University of Technology and Economics, 2 Magyar Tudósok krt., Budapest 1117, Hungary

² Telecommunication Division, AITIA International Inc., 48-50 Csetz János str., Budapest 1039, Hungary



1 Introduction

The purpose of this work is to introduce the Gatekeeper as a supporting Core System for handling interactions between Local Automation Clouds of the Arrowhead Framework. There is a joint collaboration project within the EU, called Arrowhead, with the grand challenge of enabling interoperability between systems that are natively based on different technologies. This enabler includes design and development methodologies, implemented Core Systems and Services, and a documentation structure to support the common understanding of the experts from various fields. It targets five business domains: production (processing and manufacturing), smart buildings and infrastructures, electro mobility, energy production and the virtual markets of energy.

The integration of the projected, very large collaborative automation systems requires that the participating systems can dynamically exchange information during run-time. This raises issues—among others—of service discovery, orchestration and security. For the use case scenarios of very large System-of-Systems—i.e. whole cities—the present solutions do not scale favorable. Thus Service Oriented Architecture (SOA) approaches have been investigated to address large scale automation systems [7, 19–21].

These Arrowhead Framework compliant System-of-Systems are called Local Automation Clouds—or since its short version has gained popularity already: Local Clouds. By definition, one Local Cloud consists of various Systems and their Services and are governed through the use of mandatory (and optional) Core Services provided by the locally involved Core Systems—the manner of the realization of the Framework is depicted in Fig. 1.

2 The Arrowhead Framework and Related Work

The objective of the Arrowhead Framework is to efficiently support the development, deployment and operation of interconnected, cooperative systems based on the SOA approach. The building elements of the Framework are *Systems* that provide and consume *Services*, and cooperate as a larger *Systems of Systems* (SoS) called *Local Clouds*. There are three mandatory Core Services that are required to create a minimal working SOA Local Cloud based on the Arrowhead Framework. These Services are namely the Service Discovery (from the group of Information Assurance services), Authorization (Infrastructure services) and Orchestration (System Management services). Other—supporting—Core Services have been also created to further enhance the capabilities of Local Clouds; their usage is not mandatory in the System-of-Systems, although they provide useful features. The utilization of such supporting Core Services can ease the development and deployment of these System-of-Systems, since they offer features that are often provided by self-developed modules anyway. These supporting Core Services include service offerings from

- the Plant Description System [5],
- the Configuration System [6],
- the System Registry System [9],
- the Event Handler System [31],
- the QoS Manager System [2, 14],
- the Historian System [9],
- the Translation System [11, 12], and
- the Gatekeeper System.

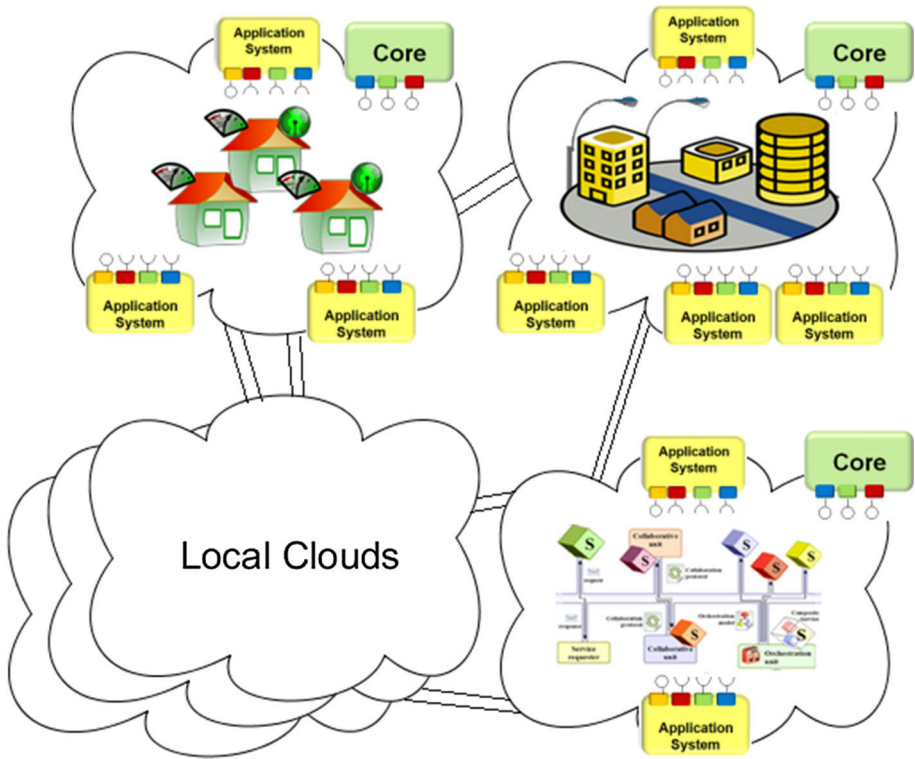


Fig. 1 Interconnected Local Automation Clouds

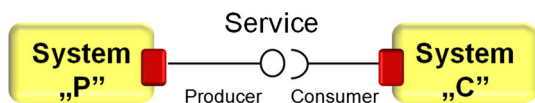
Further, brief descriptions of the Core Systems can be found in [9]. The implementation of the mandatory Core Systems and their services, together with some of the supporting Core Systems and their Services can be found in the Arrowhead Framework wiki [1]. An introduction to the main principles of Arrowhead are presented in [31].

2.1 Service Oriented Architecture Within the Arrowhead Framework

2.1.1 Services

A Service within the Arrowhead Framework is the exchange of information between a provider System and a consumer System, as seen in Fig. 2. In a service, capabilities are grouped together if they share the same context [13]. As an example, a service in a REST [15] architecture can be constituted by all interfaces which are related to the same REST resource (meaning an information source). In the context of Arrowhead, a resource might be a temperature sensor or a power consumption reading from a power meter. The same

Fig. 2 Services provided and consumed by systems



approach also applies to the other technologies being used in Arrowhead, such as COAP [30], XMPP [28], MQTT [17], or OPC-UA [24].

The Arrowhead Framework also implies the usage of a number of service orientation principles derived from high level objectives and properties. By definition, a Service can be realized by an arbitrary number of providers and consumers, therefore insuring its *reusability*. An Arrowhead Service might also support different kinds of *non-functional requirements* such as security, real-time constraints or different levels of reliability. Systems should also be capable of sequentially providing and consuming different types of Services.

Finally, in relation to Service development and maintenance, a Service has its own group of appointed stakeholders with an interest in the Service by being responsible for its governance (definition, development, deployment and maintenance). However, Services are not limited to a specific type (or instance) of a System and might not even share the same stakeholders with any System.

2.1.2 Systems

A System is what is providing and/or consuming Services (see Fig. 2). A System can be the Service Provider of one or more Services and—at the same time—the Service Consumer of one or more other Services. It normally includes software executing on (often constrained) hardware. It might support different protocol implementations of Services associated with it. A System can be a user interface display, a complete HVAC system within a house, but it can also be a small temperature sensor that complies with the Arrowhead Framework.

2.1.3 System-of-Systems

When Arrowhead compliant Systems collaborate, they become System-of-Systems (SoS), see Fig. 3. There are five main characteristics that differentiates these SoS realizations from other very large and complex, but monolithic systems [4, 22], namely

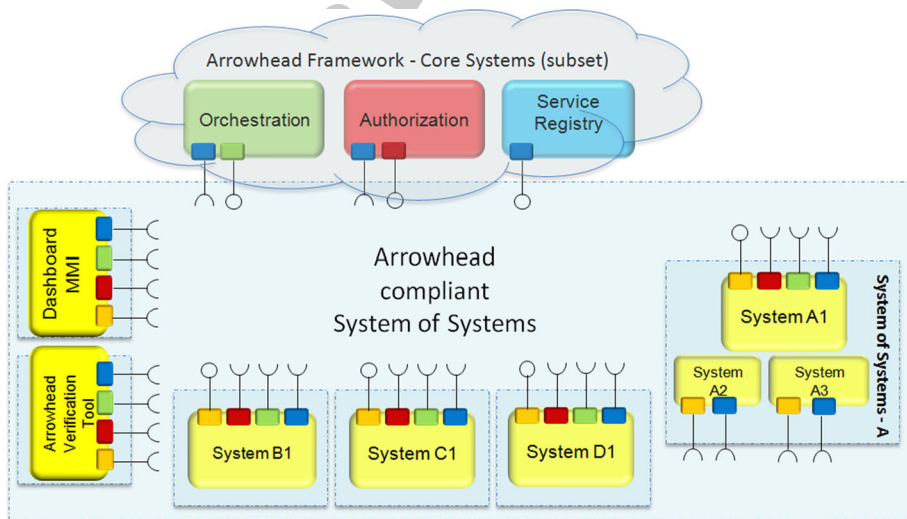


Fig. 3 Arrowhead core components support the System-of-Systems—an example



- the operational and
- management independence of its Systems,
- evolutionary development,
- emergent behaviour, and
- geographic distribution.

Since these System-of-Systems should also collaborate, the Arrowhead Framework becomes a natural enabler of further, complex solutions.

As an example, Systems A1, A2 and A3 are grouped together to create *System-of-Systems "A"*. This SoS then can be bundled together with other Systems (such as B1, C1 and D1, also depicted in Fig. 3), since they are Arrowhead compliant as well. The Arrowhead Framework provides Core Systems, such as *Orchestration*, *Authorization* and *Service Registry*, which help on establishing the most adequate and secure connections between the players. All these Application Systems are consumers of the Services of these Core Systems, which then forms this Arrowhead Local Cloud. The *Dashboard MMI* (Man–Machine Interface) allows interaction with this SoS; whereas the *Arrowhead Verification Tool* provides means for testing Arrowhead compliancy.

2.2 Requirement Against Gateways and Gatekeepers in Various Domains

Gateways and gatekeepers serve various purposes in the different communication domains. They stand at the border of a subnetwork, a service area, or an operator. Their purpose is to provide means of connection and data exchange between the internal and external worlds. There could be very specific requirements against gateway functionalities, but in general, the following challenges can be dedicated to gateways:

- protocol and data format conversion or translation,
- addressing translation,
- access control with content filtering,
- security and AAA features,
- in- and outbound link control and their resource management.

These are not always solved by the gateways themselves; some methods of inter-domain communications handle these issues through specialized protocols, or have designated nodes to handle some of the above mentioned functionalities. A brief summary of gateway requirement and solution approaches within various domains are is given in Table 1. General interoperability issues and their handling are covered in [18], where the authors raise the performance and QoS issues regarding gateways as well. There are further, less general requirements and solutions listed for IPv6-based Internet of Things solutions in [32], for ad hoc vehicular military systems in [8], and for IT services in [10, 23]. In the following Sections, we will heavily depend on these requirements when designing the Gatekeeper System and describe its functionality.

3 Orchestration of Services

In order to find matching Providers for Service Consumers, there have to be some kinds of organizing rules. Depending on the various scenarios and environment, the manifestation of these matchmaking rules can be varied. They can range from purely static, hardwired options through pre-defined, stored rule-based matchmaking, to those completely dynamic

Table 1 Requirements for various gateways—or gatekeepers—and their solution approaches within different domains

	Protocol and data conversion	Address translation, addressing issues	Access control, content filtering	Security and AAA	Link and resource management
Management internet services	–	Network Address Translation (NAT)	Firewall, proxy	Secure VPNs; separate AAA servers	Queue management; Qos solutions (IntServ, DiffServ); content delivery networks (CDNs)
VoIP service	Signaling conversion; dial tone regeneration; media codec conversion	ISDN GT ↔ SIP, H323 addresses	Central access control; or Private Branch Exchange (PBX)	Separate AAA servers	Separate management for outgoing lines to PSTN, PLMN
Cellular mobile networks	–	Multihoming of NW nodes (transparent redundancy); roaming addresses for endpoints	Access control by Home Subscriber Server (HSS/HLR)	Ciphred communication; separate nodes for various AAA functions	Handled through NW layer protocols
IT cloud solutions	IT cloud technology driver (e.g. Azure—amazon conversion)	Public address and NAT management (exposure issues); injecting routing info to partner cloud	Cloud Networking Gateway	Secure VPNs between clouds	Virtualization; grid resource management

and autonomous methods. Although Service Orchestration is not in the focus of this paper, its general principles must be discussed briefly, since it has serious effects on Inter-cloud collaboration, as well.

Note, that within the Arrowhead Framework the result of the Orchestration process is that the Service Consumer gets connectivity information towards a matching Service Provider (or in special cases: Providers). The connection then has to be established by the Consumer and Provider themselves.

3.1 Static, Hardwired Options

This type of matchmaking may not even worth to be called Orchestration, but indisputably the simplest form of matching Service Consumers with Providers. When the Service Consumer wakes up and requests a matching Provider, the options are static, and limited. The only factor that must be taken account is whether the first (primary) Provider

of that Service in the list is connectable and available—if not, then the secondary; etc. This version can be viable for environments with energy- and processing capability constraints.

3.2 Rule-Based Matchmaking

^{AQ3}In a more flexible and computationally capable environment there can be Orchestration rules that consist of many factors, and are subject to change in engineering time. Examples of such rule parameters can be based on pre-advertised metadata of providers: latency, protocol matching (no need for translator), geographical domain, etc.—beside actual availability. If these rules are stored in an easy- and fast-to-fetch manner, then the method is highly feasible for applications with low-latency orchestration constrains. Manipulation of the rules, however, must not interfere with the running SoS's runtime expectations: the rule database always has to be free from (even temporary) inconsistencies, even during rule-reconfiguration (Fig. 4).

Figure 5 depicts a scenario when the Service Consumer is orchestrated to the best matching Service Provider (1), but further matches (2), (3), and (4) are also provided, in case the primary setup fails.

3.2.1 Orchestration to Support Dynamic and Autonomous SoS

In this scenario, the Orchestrator fetches and utilizes actual information about potential Service Providers. Based on this dynamically updated information, it decides which is the actually best matching Service Provider—not by its data sheet, but by its actual status (on resource availability, on network connection capabilities, etc.). During this process the Orchestrator exchanges information with the other Core Systems and makes a decision based on the responses, as Fig. 5 suggests. This orchestration process consists of the following:

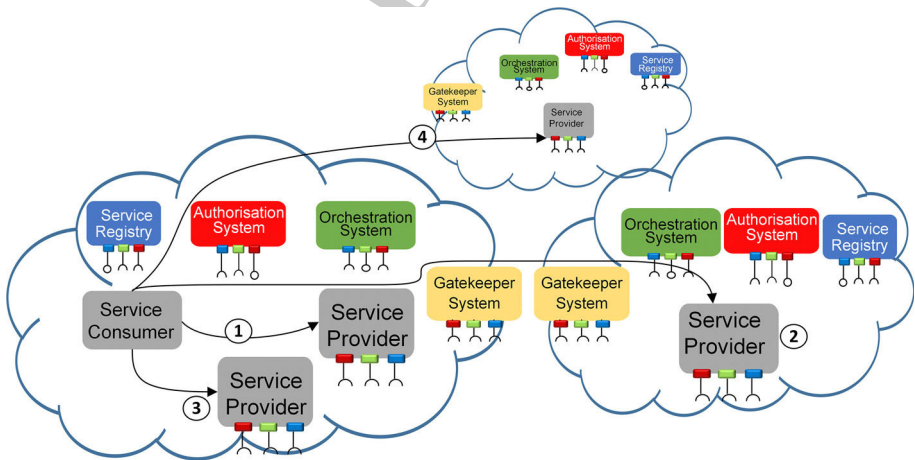


Fig. 4 Rule-based orchestration with best-matching results

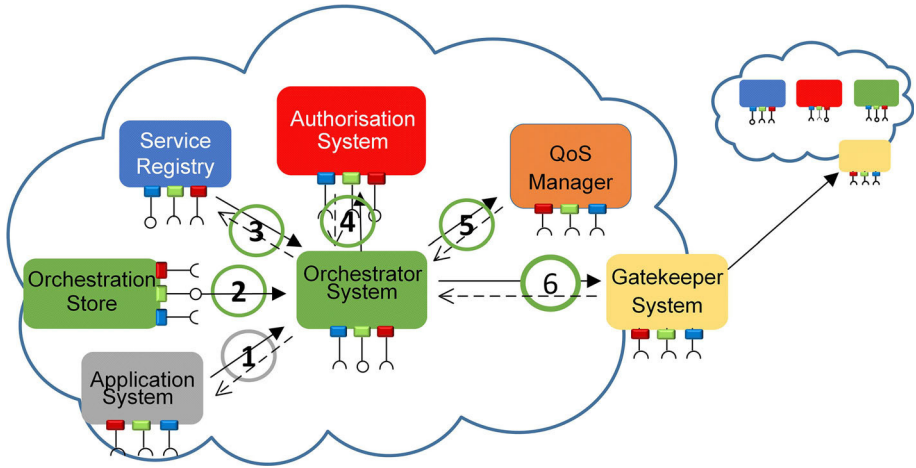


Fig. 5 The orchestration process

1. Checking whether there is a hardwired orchestration rule for the requester System in the Orchestration Store;
2. Fetching a list of suitable on-line Service Providers from the Service Registry that offer the Service at hand;
3. Extending the Service discovery to other Clouds using the Gatekeeper if necessary;
4. Filtering this list of possible Service Providers based on the authorization status of the Service Consumer;
5. Further filtering this list to find out which Service Provider(s) can satisfy the QoS expectations;
6. Choosing one Service Provider from this filtered list (final matchmaking);
7. Then making the QoS resource reservations for this optimal (Provider–Service–Consumer) combination;
8. Responding to the requester System with an Orchestration Form.

If inter-Cloud servicing is set up and enabled, then the Orchestrator can also look for adequate Service Providers outside of its Local Cloud. This is the main topic of the current paper, and is going to be addressed in the coming chapters.

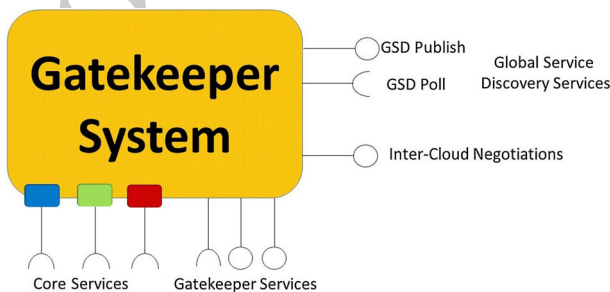


Fig. 6 The proposed gateway service module

4 Gatekeeper Services for Inter-Cloud Collaboration

In this section, we introduce the Gatekeeper as a Core System concept—as shown in Fig. 6—for providing inter-Cloud servicing capabilities in the Arrowhead Framework. All inter-Cloud preparations are to be handled by these Gatekeepers until the System (that requested the Service) enters the new, “Partnering Cloud”, in order to create the desired provider-consumer connection. However, these modules are not involved in the data path as they operate on the control plane, see in Sect. 4.5. In the followings we will discuss the processes and the issues involved and need to be addressed in this architecture.

4.1 General Requirements

There is a valid need for inter-Cloud servicing, as one single Local Cloud cannot serve for all demands; and it is not supposed to, either. Here are some examples (use cases) when inter-Cloud relations are necessary or simply better than handling requests at a local level:

- Servicing is not possible: no service provider locally.
- Service provider is not available: is in out-of-order status or registered, but not found at the time.
- Servicing is currently not possible: no free resources or the QoS expectations cannot be met locally.
- Servicing within the home cloud is not optimal (e.g. geographically a different cloud is closer or better).

One of the major philosophical questions here is about data ownership, whether the command over servicing belongs to the System (e.g. it can choose its partners) or to the managing central entities of the Local Clouds (and these Core Systems pair up Systems for servicing). As various intelligent decision-making processes (e.g. resource management) are implied here, in our primary scenario we assume that

- data ownership belongs to the Local Clouds (but this is not a necessity of the Gatekeeper Services concept),
- one identity of a Local Cloud is requesting a service that cannot be fulfilled within that Local Cloud and therefore
- it might be viable to look for clients in other Local Clouds.

This Core System will provide the following functionality (see Fig. 7):

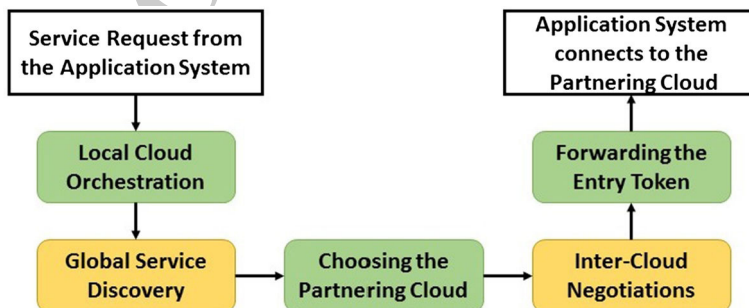


Fig. 7 Creating inter-Cloud consumer-provider relationship methodology



1. *Global Service Discovery* Finding other Arrowhead Local Clouds with suitable providers for the requested purposes.
2. *Inter-Cloud Negotiations* authentication, identity verification of the chosen Partnering Cloud, authorizing transactions and establishing secure connections by managing the dialogs (“negotiations”) between the Clouds for establishing the inter-Cloud provider-consumer relationship.

4.2 Security Issues

Interactions in a closed Arrowhead Local Cloud are secured by the central entities (Core Systems). As one Local Cloud might be a whole cyberphysical system (e.g. smart cars), we face serious security concerns when Service procurement goes outside the boundaries of the Cloud. One Cloud has to be absolutely certain that it can completely trust its Partnering Cloud when redirecting its System there. For this reason, AAA will play a key role. Since a central *Authentication Service* (e.g. Certificate Agency) might not be available in an ad hoc and volatile inter-Cloud environment, these issues might require new approaches like decentralized certificate management [3] or trust management [29]: as trust can also be built on previous experience or the partner simply being a trustworthy contact of my contacts. In this concept, *authorization* and *accounting* (the remaining A-s in AAA) can still be handled on the Local Cloud level with the proper Core Systems, see Sect. 4.5.

Regarding general security issues, the Arrowhead Framework offers methods to follow and measures to keep [9]. Among the many issues, one of the first is that service consumer authentication and service consumption authorisation has to be established. There are two main approaches: certificate (X.509) and ticket (Radius) based solutions [25, 26]. In summary, the certificate based solution provides stronger protection but is computationally heavy, whereas the ticket based solution is better adopted to resource constrained Local Clouds. The integrated tailoring of these methods can be found in [27].

4.3 Global Service Discovery

In this section, we provide approaches that would extend the Service Discovery functionality between Arrowhead Clouds. In our proposition, the Gatekeeper Systems have the ability to look for resources outside the boundaries of their Clouds. They are capable of searching for specific types of Services that are available somewhere else (in different Arrowhead Clouds) and provide the Local Cloud with options for redirection. There is *no* authentication or identity checking at this phase, but tracking the trustworthiness of the “advertisers” might be viable in some cases.

There are several approaches to this tracking of available service locations. These approaches are suitable for different environments and requirements depending on the regularity of inter-Cloud interactions, the volatility of service availability in the different Clouds and the flexibility required from the collaborating Arrowhead System-of-Systems (Fig. 8).

4.3.1 Strict Rules on Possible Gatekeeper Connections

The most basic approach is hardwiring—and manually configuring—information about a certain set of other Gatekeepers into each Gatekeeper where they can turn to, see Fig. 9.

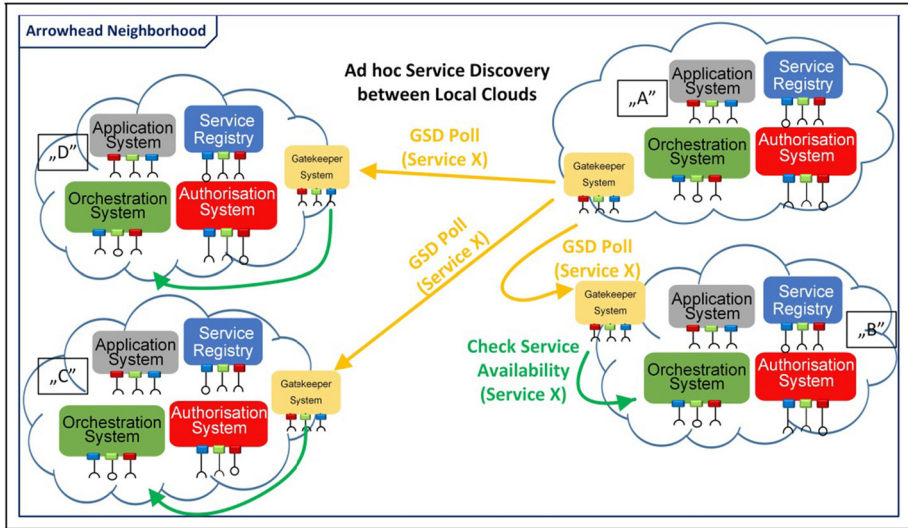


Fig. 8 Service discovery in a hardwired inter-Cloud scenario

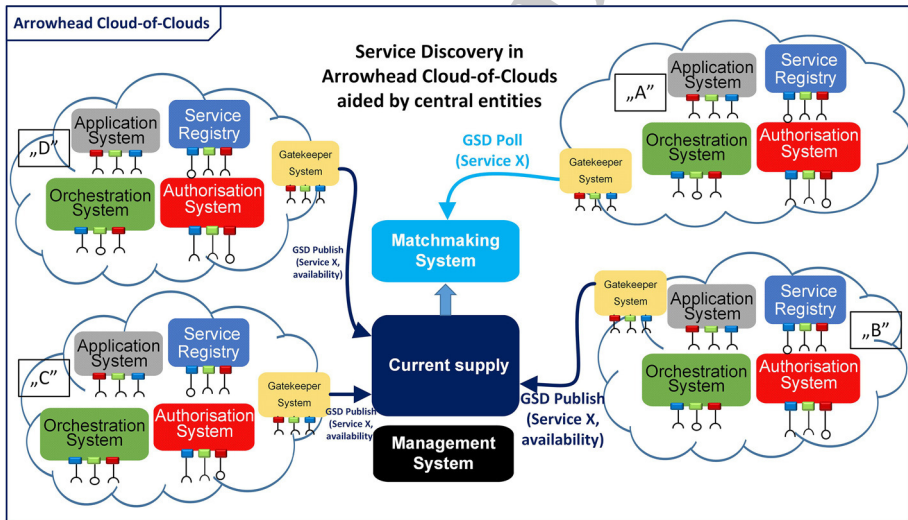


Fig. 9 The Cloud-of-Clouds concept

This can be based on the Service type, time of day, geographical requirements (choosing a physically close partner), etc. This concept brings several advantages and limitations as well. In this scenario, Cloud operators have strict oversight of their Arrowhead Clouds and can easily log the interactivity, set up billing and access control between operators (fits business purposes). It is also highly secure: no need for authentication, identity control or trust management in any phase. However, there is the need for manual edition of configuration and hence there is absolutely no option for scaling automatically.



284 4.3.2 Ad Hoc Neighborhoods

285 The second approach is still based on per-transaction polling of neighbors, but letting the
286 Gatekeepers automatically detect and poll their neighborhood. The ad hoc peer-to-peer
287 detection of neighbors can provide adaptivity for volatile environments (e.g. moving cars
288 looking for charging stations along the road). However, it will also pose a high adminis-
289 trative overhead (for Gatekeeper components) and cause scaling issues. This methodology
290 will also present a limited list of available resources to the Local Cloud (just from the
291 current contacts). There are several problems to solve here, such as the tracking, trust
292 management and authentication of a rapidly changing neighborhood.

293 4.3.3 Cloud-of-Clouds

294 The third approach is pointing towards the creation of a dedicated inter-Cloud system, where
295 the “demand and supply” of inter-Cloud requests can meet. The main purpose of this is to take
296 off the computational overhead from the Arrowhead Cloud Gatekeepers and local Core Sys-
297 tems in a stock-market like environment (see Fig. 9). This centralized entity could provide
298 global matchmaking for the participants of the Cloud-of-Clouds and could also centralize
299 resource allocation and trust management. This can be achieved by tracking the reliability of the
300 parties and providing the requester Clouds with only the globally optimal Partnering Cloud.

301 4.4 Inter-Cloud Negotiations

302 After the requesting Cloud decides for a Partnering Cloud, the Gatekeepers of these Clouds
303 will have to perform negotiations to settle the different aspects of the transaction. At the
304 end of this phase, the requesting System gets a token that will authenticate and navigate
305 [27] it in the Partnering Cloud (using the local Service Discovery data). During this
306 process, the following issues are to be handled:

- 307 1. protocol negotiations (Arrowhead Framework version and protocol matching),
- 308 2. mutual authentication and identity checking of the parties,
- 309 3. actual admission control (authorization and resource allocation for the transaction in
310 the Clouds),
- 311 4. establishing a secure connection between the Gatekeepers, therefore the Arrowhead
312 Clouds (creating the data path),
- 313 5. sharing the local Service Discovery data of the Partnering Cloud (addressing the
314 Partnering Cloud from the outside),
- 315 6. injecting temporary authorization information into the Authorization System in the
316 Partnering Cloud (access control of the “foreigner” System),
- 317 7. the Partnering Cloud Gatekeeper issues a token for the requester with temporary entry
318 and shares it with the requesting Gatekeeper,
- 319 8. the requesting Local Cloud forwards this token to the requesting System,
- 320 9. the System connects to the Provider using the Partnering Cloud’s Core Systems.

321 4.5 Requirements for Local Core Systems and Services

322 The inter-Cloud servicing methodology requires new (earlier considered as optional)
323 functionalities [16] to be designed and implemented within the Core Systems of the
324 Arrowhead Framework. These new functionalities include the following, among others.



- *Authorization* inter-Cloud transactional rights management (access rights from and towards external Clouds);
- *Authentication* identity verification of other Local Clouds and their Systems;
- *Orchestration* intelligent resource management and Service matchmaking capabilities based on the current local and inter-Cloud situation in a scalable but robust manner;
- *Event and Error Handling* for supporting the inter-Cloud activity (e.g. logging, status reporting) the hosting cloud Orchestrator could instruct the foreigner System to contact the Core Services from both Clouds for these purposes. This way everyone will be notified properly of the happenings;
- *QoS Management* the Global Service Discovery response can (and in the dynamic case: must) also take into account the requested QoS parameters;
- *Security System* the creation of the *data path* between the Clouds require specialized Systems that handle all the network configurations in relation to the transaction. On one hand, this is a QoS issue, but more importantly, it is indeed, a Security issue.

5 A Case Study

In an average situation the sequence depicted in Fig. 10 would lead to System 1 (from Arrowhead Cloud A) connecting to Arrowhead Cloud B for getting Service X from a System in that SoS. This sequence consists of the following logical steps:

1. Normal service orchestration procedure begins (with the help of Cloud A's Core Service components).
2. In case the request cannot be fulfilled locally, the Orchestrator A will turn to the Gatekeeper A and initiate a Global Service Discovery (see 4.3).
3. Gatekeeper A provides a list to the local Orchestrator on Arrowhead Clouds where Service X is currently available.
4. Cloud A will decide where it will redirect System 1 for Service X (in this case: Cloud B).
5. If there is a suitable Arrowhead Cloud, where Service X is optimally available, Cloud A will check for the requesting s (System 1s) AAA status and rights (e.g. if it has right to use outside resources).
6. Gateway A will start negotiating with Gateway B for access to Cloud B (Inter-Cloud Negotiations) as in Fig. 11. Cloud B will also prepare for System 1's entry.
7. When access to Cloud B has been granted for System 1 (from Cloud A), Gateway B will share the Cloud B access data (e.g. Service Discovery address) and send the transaction token via Gatekeeper A. This token will enable System 1 to access Cloud B.
8. The local cloud of System 1 (Cloud A) will orchestrate these information back to System 1 and redirect it to discover the Services in Cloud B.
9. With the help of the token, System 1 will enter Cloud B to perform Service X using the normal orchestration process of Cloud B.

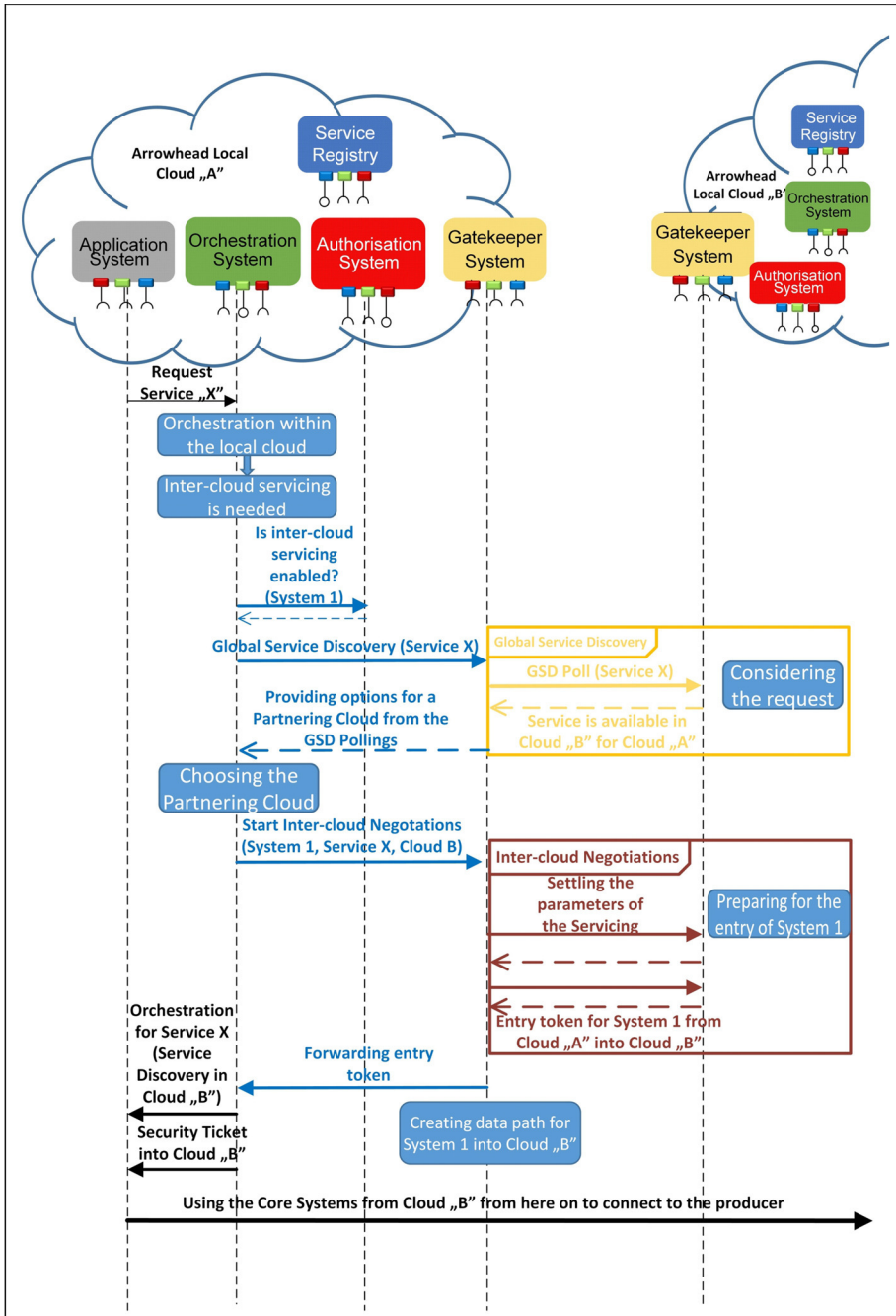


Fig. 10 An example inter-Cloud servicing sequence diagram

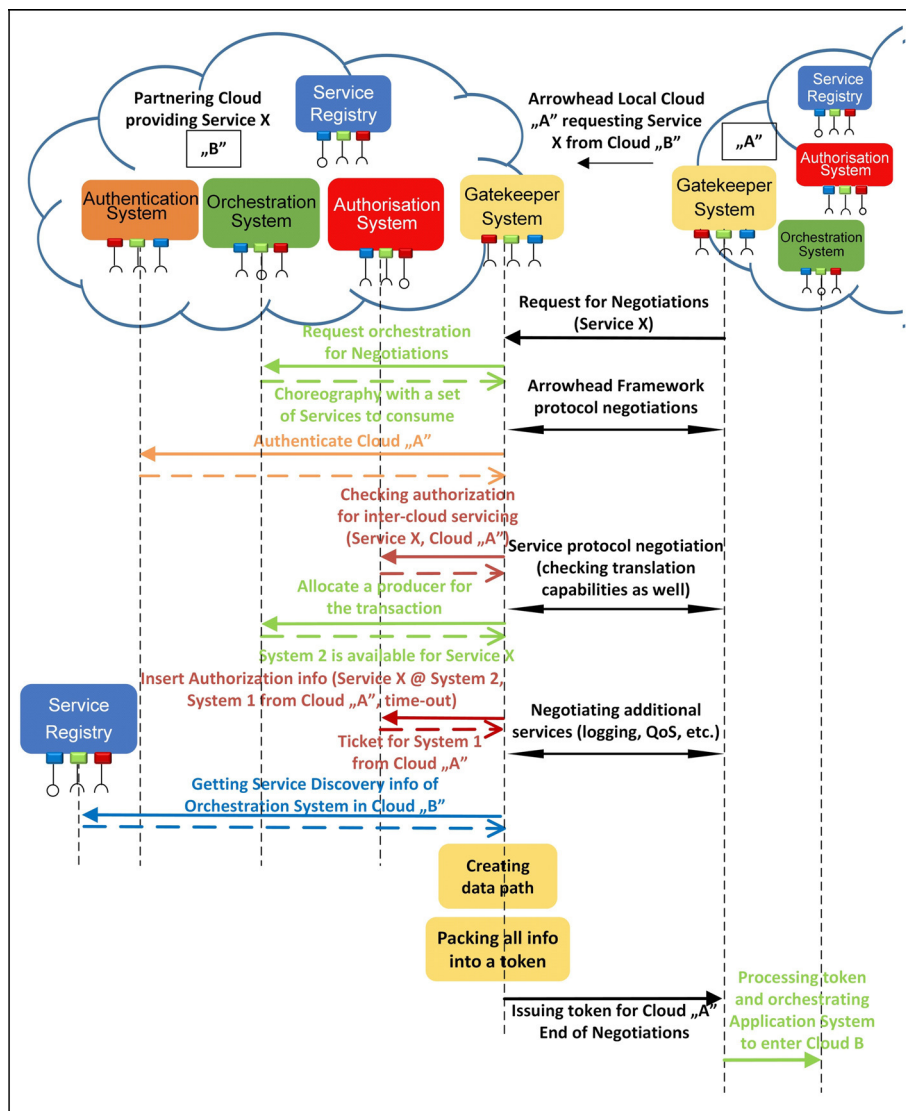


Fig. 11 An example Inter-Cloud Negotiations process

6 Summary

The Arrowhead Framework provides means for Systems to interconnect in a service oriented manner: service discovery, loose coupling and late binding are the fundamental requirements here. Moreover, common design methods and documentation structure helps the engineers and programmers to build their systems Arrowhead compatible; enhancing them with the capability to interwork with other such systems, or Arrowhead-capable Local Clouds.



Beside those Core Systems that are mandatory in every Local Cloud (Service Discovery, Authorization and Orchestration), there are various supporting Core Services (provided by supporting Core Systems), as well.

In order to achieve greater potential, some Service Consumers may accept the possibility to connect to Service Providers outside their own Local Cloud. This need opened a large set of questions related to inter-Cloud service interactions. The current paper aimed to answer these issues; with implemented, working use cases as proof of concepts. The inter-Cloud concept and all its details are now part of the Arrowhead Framework.

The concept of Gatekeeper as a new Core System introduced in this paper solves many of the arising issues, including security, service discovery, orchestration, and other requirements set by the Local Clouds. We described methods for the interworking with the help of such Gatekeepers, and presented a case study of a service consumer System connecting to a service consumer from a different Arrowhead Cloud.

Acknowledgements This work is supported by the EU ARTEMIS JU funding, within Project ARTEMIS/0001/2012, JU Grant No. 332987 (ARROWHEAD).

References

1. Arrowhead framework wiki—The main entry point to documentation and code (2016). <https://forge.soia4d.org/plugins/mediawiki/wiki/arrowhead-f>.
2. Albano, M., Garibay-Martínez, R., & Ferreira, L. L. (2015). Architecture to support quality of service in arrowhead systems. In *Proceedings of INFORUM 2015*, Covilhã, Portugal
3. Aslam, B., & Zou, C. (2009). Distributed certificate and application architecture for VANETs. In *IEEE Military Communications Conference (MILCOM)* (pp. 1–7).
4. Blomstedt, F., Lino Ferreira, L., Klisics, M., Chrysoulas, C., Martinez de Soria, I., Morin, B., et al. (2014). The arrowhead approach for SOA application development and documentation. In *IEEE IECON*.
5. Carlsson, O., Hegedus, C., Delsing, J., & Varga, P. (2016). Organizing IoT systems-of-systems from standardized engineering data. In *IEEE IECON*.
6. Carlsson, O., Pereira, P. P., Delsing, J., Eliasson, J., Ahmad, B., Harrison, R., et al. (2016). Configuration service in cloud based automation systems. In *IEEE IECON*.
7. Colombo, A. W., Bangemann, T., Karnouskos, S., Delsing, J., Stluka, P., Harrison, R., et al. (2013). *Industrial cloud-based cyber-physical systems—The IMC-AESOP approach*. Berlin: Springer.
8. DaSilva, L. A., Midkiff, S. F., Park, J. S., Hadjichristofi, G. C., & Davis, N. J. (2004). Network mobility and protocol interoperability in ad hoc networks. *IEEE Communications Magazine*, 42(11), 88–96.
9. Delsing, J. (Ed.). (2016). *IoT automation—Arrowhead framework*. Boca Raton: CRC Press.
10. Demchenko, Y., Ngo, C., de Laat, C., & Lee, C. (2014). Federated access control in heterogeneous Intercloud environment: Basic models and architecture patterns. In *IEEE international conference on cloud engineering (IC2E)* (pp. 439–445).
11. Derhamy, H., Eliasson, J., & Delsing, J. (2016). IoT interoperability-on-demand and low latency transparent multi-protocol translator. *Transactions on Services Computing* (submitted).
12. Derhamy, H., Eliasson, J., Delsing, J., Pereira, P. P., & Varga, P. (2015). Translation error handling for multi-protocol SOA systems. In *20th IEEE Conference on Emerging Technologies & Factory Automation, ETFA 2015* (pp. 1–8), Luxembourg, September 8–11, 2015.
13. Erl, T. (2007). *SOA principles of service design (The Prentice Hall service-oriented computing series from Thomas Erl)*. Upper Saddle River, NJ: Prentice Hall PTR.
14. Ferreira, L. L., Albano, M., & Delsing, J. (2016). QoS-as-a-service in the local cloud. In *Proceedings of SOCNE 2016, in conjunction with ETFA 2016*, Berlin, Germany.
15. Fielding, R. T. (2000). *Architectural styles and the design of network-based software architectures*. Ph.D. thesis, University of California, Irvine.
16. Hegedus, C., Kozma, D., Soos, G., & Varga, P. (2016). Enhancements of the arrowhead framework to refine inter-cloud service interactions. In *IEEE IECON*.
17. ISO/IEC. (2014). *Message queuing telemetry transport (MQTT) version 3.1.1*.



18. Kalyanasundaram, P., & Sethi, A. S. (1994). Interoperability issues in heterogeneous network management. *Journal of Network and Systems Management*, 2(2), 169–193.
19. Karnouskos, S., Colombo, A. W., Jammes, F., Delsing, J., & Bangemann, T. (2010). Towards an architecture for service-oriented process monitoring and control. In *36th annual conference of the IEEE Industrial Electronics Society (IECON-2010)*, Phoenix, AZ.
20. Kaur, N., McLeod, C., Jain, A., Harrison, R., Ahmad, B., Colombo, A., et al. (2013). Design and simulation of a SOA-based system of systems for automation in the residential sector. IEEE.
21. Kyusakov, R., Eliasson, J., Delsing, J., van Deventer, J., & Gustafsson, J. (2013). Integration of wireless sensor and actuator nodes with it infrastructure using service-oriented architecture. *IEEE Transactions on Industrial Informatics*, 9(1), 43–51.
22. Maier, M. W. (1998). Architecting principles for systems-of-systems. *Systems Engineering*, 1(4), 267–284.
23. Mechtri, M., & Zeghlache, D. (2013). Inter-Cloud networking gateway architecture. In *IEEE international conference on computing technology and science (CloudCom)* (Vol. 2, pp. 188–194).
24. OPC Foundation. (2010). *OPC unified architecture specification*.
25. Pereira, P. P., Eliasson, J., & Delsing, J.: An authentication and access control framework for CoAP-based Internet of Things (pp. 5293–5299). IEEE (2015). doi:10.1109/IECON.2014.7049308.
26. Pereira, P. P., Eliasson, J., & Delsing, J. (2016). Efficient framework for industrial IoT. *IEEE Internet of Things Journal*.
27. Plosz, S., Hegedus, C., & Varga, P. (2016). Advanced security considerations in the Arrowhead framework. *Computer safety, reliability, and security, Lecture notes in computer science* (Vol. 9923, pp. 234–245). Springer.
28. Saint-Andre, P. (2011). *Extensible messaging and presence protocol (xmpp): Core*.
29. Shand, B., Dimmock, N., & Bacon, J. (2003). Trust for ubiquitous, transparent collaboration. In *IEEE international conference on pervasive computing and communications (PERCOM)* (pp. 153–160).
30. Shelby, Z., Hartke, K., & Bormann, C. (2014). *The constrained application protocol (CoAP)*.
31. Varga, P., Blomstedt, F., Lino Ferreira, L., Eliasson, J., Johansson, M., Delsing, J., et al. (2016). Making system of systems interoperable—The core components of the arrowhead technology framework. *Journal of Network and Computer Applications*.
32. Ziegler, S., & Crettaz, C. (2014). IPv6 as a global addressing scheme and integrator for the Internet of Things and the Cloud. In *Advanced Information Networking and Applications Workshop (WAINA)* (pp. 797–802).



Pál Varga is currently associate professor at the Department of Telecommunications and Media Informatics (TMIT) of Budapest University of Technology and Economics (BME), Hungary. His main research interests include traffic analysis and modeling, Deep Packet Inspection, hardware acceleration, network performance measurements, network monitoring and management, fault localization, traffic classification, end-to-end QoS, QoE and SLA issues, 100 Gigabit Ethernet, Service Oriented Architectures, Internet of Things, system interoperability. In the European Research Project Artemis Arrowhead, he is the leader of the central technical work package that deals with the definition and design of the Interoperability and Integrability Framework. This later got its name simplified to Arrowhead Framework.



Csaba Hegedűs is research and development engineer at AITIA International Inc. He received his electrical engineering B.Sc. from BME Hungary and currently doing his M.Sc. studies on the same field, specializing in telecommunications. His research interests include the communication issues of Cyber-Physical Systems and Internet of Things architectures. At AITIA International Inc. he is a main contributor and developer on the ARTEMIS MANTIS and Arrowhead projects. He was the main contributor to the latest generation advances of the Arrowhead Framework.

UNCORRECTED PROOF

Journal : 11277
Article : 4137



Author Query Form

Please ensure you fill out your response to the queries raised below and return this form along with your corrections

Dear Author

During the process of typesetting your article, the following queries have arisen. Please check your typeset proof carefully against the queries listed below and mark the necessary changes either directly on the proof/online grid or in the ‘Author’s response’ area provided below

Query	Details Required	Author’s Response
AQ1	Please check and confirm the given name and family name of author ‘Csaba Hegedűs’.	
AQ2	Please confirm the corresponding author is correctly identified and amend if necessary.	
AQ3	Please note that Figs. 1–3 and 5–12 have been renumbered as Figs. 1–3 and 4–11, respectively, and Fig. 4 has been processed as Table 1. Please confirm.	
AQ4	Please check and confirm the inserted citation of Figs. 4 and 8 is correct. If not, please suggest an alternative citation. Please note that figures should be cited in sequential order in the text.	
AQ5	Please update Refs. [11, 26, 28, 30, 31] with complete details.	