An Aggregating Discovery Service for the EPCglobal Network

Jürgen Müller, Jan Oberst, Stefan Wehmeyer, Jonas Witt, Alexander Zeier, and Hasso Plattner

Hasso Plattner Institute for IT Systems Engineering
Prof.-Dr.-Helmert-Str. 2-3, 14482 Potsdam, Germany
{juergen.mueller | alexander.zeier | office-plattner}@hpi.uni-potsdam.de,
{jan.oberst | stefan.wehmeyer | jonas.witt}@student.hpi.uni-potsdam.de

Abstract

Although the achievements of EPCglobal standardization efforts in the last years are substantial and the diffusion of the EPCglobal network continues, a service to access item-level information stored at potentially unknown supply chain partners is still missing. Without this ‘Discovery Service’ the EPCglobal network cannot be used with unknown business partners. In this contribution we present a novel Discovery Service approach. In contrast to other Discovery Service approaches we use the Discovery Service to aggregate the data of interest. This eliminates problems other approaches cannot solve, e.g. assuring data ownership, low access control maintenance effort, and minimal complexity for the information requester. We present a requirements model, describe the implementation of our Discovery Service approach, and compare it to existing Discovery Service approaches. Finally we show how to overcome scalability challenges to implement our Discovery Service approach in a real-world environment.

1. Introduction

Strategies for enhancing supply chain performance often require data sharing between supply chain partners to remove inefficiencies such as out-of-shelf situations, incorrect inventory levels, and incorrect demand and sales forecasts [18]. With Radio Frequency Identification (RFID) and data matrix technology in combination with the Electronic Product Code (EPC) it becomes possible to use unique identifiers for each individual object [8]. RFID furthermore allows the automation of collecting events and therefore lowers costs for enhancing information granularity about an object traveling through the supply chain [17].

The EPCglobal consortium defines standards for how to process and share EPC data [3]. During or after manufacturing of an item an EPC is assigned to it. The EPC is read by readers at participating supply chain partners. These read events are processed by a middleware [6, 16] and stored locally at each supply chain partner using an EPC Information Services repository (EPCIS) which implements the EPC Information Services standard [4].

Having this item-level information at hand, new traceability applications could emerge which are based on sharing data to create business value or which aim at fulfilling legal requirements [17, p. 17 et seq.]. If the URLs of the EPCISs were all known in advance, this is already possible today. But nowadays complex and fast-changing supply chains often limit the visibility of upstream and downstream supply chain partners to those one has direct interaction with. Therefore a service has to be established which identifies relevant but possibly unknown information holders for a specific EPC respecting the access rights defined by the information holder. This service is referred to as ‘Discovery Service’ and describes an application level protocol to collect the information stored in federated EPCISs. Unlike on the Web a top-down search engine approach using spiders or crawling is not possible because the item-level information is business critical and not exposed to the public [17].

A Discovery Service is the intermediary of the EPCglobal network like search engines are intermediaries for the World Wide Web. It simplifies the information exchange of supply chain partners and ensures supply chain partners are able to find other parties that have been in possession of a certain product. With this information and the appropriate access rights, the company can then, e.g., retrieve all RFID read events that occurred. In contrast to search engines, a Discovery Service has to ensure privacy of information publisher and information requester. Given the complexity of today’s supply chains, EPCglobal Network applications like track and trace only become possible with a Discovery Service [14, 19].

Following Hevner’s methodology for conducting design science in information systems research [10] we explicitly mention the artifacts presented in this paper which are:
2. Related Discovery Service Approaches

In this section we summarize existing Discovery Service approaches and research in related disciplines. Two distinct models of gathering information about an item traveling through the supply chain exist. The first model is the ‘Registry Model’ that uses a central Discovery Service to relate specific EPCs to related information holders. The second one is the ‘E-Pedigree Model’. In this model a chain of custody document is created at the manufacturer and enriched at each step in the supply chain [5].

2.1. Beier et al.: Discovery Services

In [1] Beier et al. present a first implementation of a Discovery Service, which can be summarized as Directory Look-up approach. In their paper, they analyze the appropriateness of the Object Name Service [7] and come to the result that this approach is improper for building Discovery Services. The developed Directory Look-up approach works as follows: real-world items attached with an EPC travel through the supply chain. At each company the item passes, the EPC is read and a read event is stored in the companies’ EPCIS. For each EPC that is stored in an EPCIS for the first time, the Discovery Service is notified and stores the EPC, the URL of the submitting EPCIS, a timestamp, the certificate of the submitter, and a visibility flag in its repository. The Discovery Service can then be queried with an EPC of interest. It replies with a list of relevant EPCIS URLs. Finally the requester can query all relevant EPCISs by himself and aggregate the respective information. The underlying assumptions are that all participants of the EPC network are authorized by EPCglobal and equipped with a certificate by a trusted third party.

According to Beier et al. [1], access to a company’s EPCIS should be implemented role-based and policy-based with cell-level data disclosure control. At the Discovery Service level, row-level data access control should be enforced and, using the visibility flag, the owner of the data decides whether the record is shared among all authorized participants of the EPC network or access is restricted to companies having information about the same EPC. To achieve EPCIS address confidentiality, Beier et al. propose the usage of EPCIS proxy servers by storing not the real but the proxies URL at Discovery Service level.

2.2. Huang et al.: A Distributed ePedigree Architecture

Huang et al. follow the ‘E-Pedigree’ model of collecting information about an item [11]. The E-Pedigree document contains a list of all EPCs of the goods being shipped. The manufacturer initially creates and encrypts it and passes it to the next supply chain partner which adds his data, electronically signs the document and passes it to the next supply chain partner etc. The authors propose a distributed architecture for efficiently retrieving the first supply chain partner (for forward tracing) or the last supply chain partner (for reverse tracing) using the General Manager Number stored in an EPC [8] in combination with the Object Naming Service (ONS) [7] which is able to resolve the General Manager Number to the respective EPCIS. The authors propose a distributed ONS architecture based on distributed hash tables. The retrieved EPCIS address is used to traverse the supply chain from one end to discover information holders for a specific EPC of interest. This approach of following the chain is called ‘Daisy Chain’ approach. The authors claim the solution to be scalable for item-level product tagging because their approach does not rely on a central component in the EPCglobal network. To implement their approach, they rely on re-writable tags.

2.3. BRIDGE project: High-level design for Discovery Services

BRIDGE is an acronym for Building Radio frequency IDentification for the Global Environment. The objective of this EU-funded project is to “research, develop and implement tools to enable the deployment of EPCglobal applications in Europe” [20].

In the report [20] the authors propose eight Discovery Service approaches, evaluate them, and finally judge four as promising candidates for large scale Discovery Services. It is important to understand that EPCISs can serve two different types of queries: one-off queries and standing queries. One-off queries are performed by a client once and no further
communication between client and EPCIS is planned. Standing queries are subscriptions which can be time-controlled using a query schedule (e.g., a client wants to be informed every hour) or trigger-controlled (e.g., a client wants to be informed if new information about an EPC of interest is available) [4].

We will now briefly describe the four candidates identified by the authors. The first candidate is called Directory-of-Resources and equals the Directory Lookup-up approach by Beier et al. [1]. The second candidate is called Notification-of-Resources and works as the Directory-of-Resources except that a client shows interest about certain information by creating a subscription at the Discovery Service. Once an EPCIS notifies the Discovery Service about an EPC which matches the criteria defined in the subscription, the Discovery Service informs the client that the respective EPCIS is in possession of information related to the subscription. The third candidate is called Notification-of-Clients: EPCISs notify the Discovery Service for each new EPC they own information about. Once a client shows interest in an EPC the Discovery Server informs all relevant EPCISs. The EPCISs send an availability notification to the client, which then queries the respective EPCISs. The last candidate identified by the authors is called Query Propagation and acts like Notification-of-Clients except that the information is sent to the client by the EPCISs immediately without the availability notification. The authors summarize the first two candidates as Directory Service approach and the last two as Query Relay approach.

2.4. Kürschner et al.: Discovery Service Design in the EPCglobal Network

In their related work, Kürschner et al. [13] describe that the concepts of the Domain Name Service and the Service Location Protocol are not appropriate to solve the Discovery Service problem. The authors present the Directory Look-up approach by Beier et al. and criticize that the EPCIS address of companies having information about the EPC of interest might be revealed if there are no access control policies in place. Otherwise, if these policies were established, the maintenance effort and complexity would rise because fine-grained access rights would have to be defined at Discovery Service level and policies would have to be synchronized between companies’ EPCISs and the Discovery Service. To fulfill the requirement of low access right maintenance effort, they present the Query Relay approach developed in [20] in detail. The idea is to use the Discovery Service as a relay by forwarding the respective client queries to all relevant information holders. The EPCISs reply directly to the requester. Therefore the EPCIS address is not revealed to the requester if the respective company decides not to reply to the query at all. Finally, both presented approaches are discussed and evaluated.

2.5. BRIDGE project: Working prototype of serial-level lookup service

The report [9] is a follow-up of the previously described BRIDGE report [20]. The authors describe the implementation of the Directory Service approach and discuss several design aspects: For efficient data storage the Lightweight Directory Access Protocol (LDAP) is used; interfaces of the Discovery Service adhere to the EPCIS standard [4]; an integration module within EPCISs works as a sniffer for read events that are relevant for being published to the Discovery Service and security is implemented using X.509 certificates to build a public key infrastructure between EPCISs and Discovery Service. The authors again present the Query Relay approach already described in Section 2.4.

2.6. Synthesis

Having analyzed the related Discovery Service approaches we can identify the two models ‘Registry’ and ‘E-Pedigree’. In the ‘Registry Model’ we see the approaches of ‘Directory Service’ (DS) and ‘Query Relay’ (QR). The ‘E-Pedigree Model’ uses the approach ‘Daisy Chain’ (DC). Table 1 summarizes the related work.

<table>
<thead>
<tr>
<th>Table 1. Related Discovery Service Approaches</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Registry</strong></td>
</tr>
<tr>
<td><strong>Beier et al. (2.1)</strong></td>
</tr>
<tr>
<td><strong>Huang et al. (2.2)</strong></td>
</tr>
<tr>
<td><strong>Bridge project (2.3)</strong></td>
</tr>
<tr>
<td><strong>Kürschner et al. (2.4)</strong></td>
</tr>
<tr>
<td><strong>Bridge project (2.5)</strong></td>
</tr>
</tbody>
</table>
3. Requirements for a Discovery Service for the EPCglobal Network

In this section, we will state the requirements for large-scale Discovery Services for the EPCglobal network and structure them compliant to the ISO/IEC 9126 standard [12]. We analyzed requirements stated in Section 2, results of requirements engineering conducted within the BRIDGE project [21] and requirements stated by the Internet Engineering Task Force [17]. We limit the selection of requirements to the subset of all meaningful requirements that exemplifies the differences of the Discovery Service approaches.

The ISO/IEC 9126 standard describes software engineering product quality. It divides product quality in three parts: quality in use, external quality, and internal quality [12]. Quality in use considers the quality requirements by the user when the software is used. External quality is the set of software product characteristics from an external view when the software is executed. Internal quality describes the rigor related to software design and construction of the software product. As proposed in ISO/IEC 9126-1 [12] this classification can be used for structuring requirements. Given the fact that we want to analyze Discovery Service designs and not their implementation, we will only incorporate requirements for the system in use and external requirements.

Requirements for the system in use are placed into the non-exclusive categories effectiveness (E), productivity (P), safety (S), and satisfaction (Sa) [12]. A list of requirements for the Discovery Service in use can be found in Table 2.

<table>
<thead>
<tr>
<th>ID</th>
<th>Short Description</th>
<th>E</th>
<th>P</th>
<th>S</th>
<th>Sa</th>
</tr>
</thead>
<tbody>
<tr>
<td>U₁</td>
<td>Low client complexity [1], [16, p. 30 et seq.]</td>
<td>•</td>
<td>•</td>
<td>–</td>
<td>–</td>
</tr>
<tr>
<td>U₂</td>
<td>Client shall be in full control of query [13, p. 23 et seq.], [16, Section B, p. 29]</td>
<td>–</td>
<td>–</td>
<td>•</td>
<td>–</td>
</tr>
<tr>
<td>U₃</td>
<td>Low response latency [16, p. 4], [17, p. 50]</td>
<td>–</td>
<td>–</td>
<td>•</td>
<td>–</td>
</tr>
<tr>
<td>U₄</td>
<td>Complete and correct query result [13, p. 22], [16, p. 4]</td>
<td>•</td>
<td>–</td>
<td>•</td>
<td>–</td>
</tr>
<tr>
<td>U₅</td>
<td>Data ownership for information holder [13, p. 22], [14, p. 7], [16, p. 4]</td>
<td>–</td>
<td>–</td>
<td>•</td>
<td>–</td>
</tr>
</tbody>
</table>

External requirements are defined by the market or derived from requirements of the system in use. Furthermore, legal requirements are considered in external requirements. External requirements are placed into six non-exclusive categories. These are functionality (F), reliability (R), usability (U), efficiency (E), maintainability (M), and portability (P) [12]. We list external requirements on the Discovery Service in Table 3.

<table>
<thead>
<tr>
<th>ID</th>
<th>Short Description</th>
<th>F</th>
<th>R</th>
<th>U</th>
<th>E</th>
<th>M</th>
<th>P</th>
</tr>
</thead>
<tbody>
<tr>
<td>E₁</td>
<td>Synchronous response to client [17, p. 50]</td>
<td>–</td>
<td>–</td>
<td>•</td>
<td>•</td>
<td>–</td>
<td>–</td>
</tr>
<tr>
<td>E₂</td>
<td>Manageable access control [13, p. 23], [14, p. 23]</td>
<td>–</td>
<td>–</td>
<td>•</td>
<td>•</td>
<td>•</td>
<td>–</td>
</tr>
<tr>
<td>E₃</td>
<td>Using existing infrastructure [16, Section B, p. 34 et seq.]</td>
<td>•</td>
<td>–</td>
<td>•</td>
<td>–</td>
<td>–</td>
<td>–</td>
</tr>
<tr>
<td>E₄</td>
<td>Highly scalable design [1, 11, 13, p. 24], [16, pp. 35]</td>
<td>–</td>
<td>–</td>
<td>•</td>
<td>•</td>
<td>–</td>
<td>–</td>
</tr>
</tbody>
</table>

4. Aggregating Discovery Service

The purpose of the Aggregating Discovery Service (ADS) is to forward client queries to relevant EPCISs, aggregate their responses and synchronously respond to the client request. This reduces client complexity, brings low response latency, delivers complete and correct information for the requester, ensures data ownership for the information holder, avoids the need for fine-grained access control replicated at Discovery Service level and guarantees confidentiality of clients and information holders.

4.1. Design

The ADS is a centralized service which offers two interfaces (see Figure 1). Supply chain partners inform the ADS about relevant EPC read events through the notify interface. The query interface is used to gather information about an EPC of interest from the EPCglobal network. The ADS links EPCs to supply chain partners which can provide detailed information about those EPCs. Certificates are used to provide authentication as proposed in [1, 9].
4.1.1. Notify Interface. The notify interface is used to inform the ADS about read events to be shared within the EPCglobal network. The ADS receives the EPCIS URL of the submitting partner and one or more EPCs that have been handled by this entity. Submitting multiple EPCs at once allow the client to batch notify requests and improves performance by lowering connection overhead. We propose a simple, XML-based format for this message to be submitted via HTTP POST. The ADS maintains an association between submitting EPCISs and submitted EPCs. This allows the ADS to determine all EPCISs that hold more information about an EPC.

4.1.2. Query Interface. The query relay provides an EPCIS-equivalent query interface [4] as proposed in [9]. In addition to a full query, the client also can identify relevant EPCISs using a resource query [20]. For both types of queries the execution is as depicted in Figure 2: The ADS waits for an incoming client query (1.) and parses the query to extract relevant EPCs (2.). The ADS then queries its internal database to look up the URLs of EPCISs which are relevant for this query (3.) and forwards the original query to those EPCISs (4.). After subresponses returned from the EPCISs (5.), they are parsed and the ObjectEvents are extracted and combined (6.). The aggregated result is then returned to the client (7.). Effectively, this means the ADS acts as a proxy that is transparent to the client.

4.2. Discussion of Different Discovery Service Approaches

The fulfillment of the requirements stated in Section 3 and the referenced literature in that section is substantial for a well-designed Discovery Service architecture that can easily be integrated in the EPCglobal network. Following a summarization of the fulfillment of selected requirements in Table 4 we analyze the Directory Service (DS) approach, Query Relay (QR) approach, Daisy Chain (DC) approach and Aggregating Discovery Service (ADS) approach in detail with regards to the requirements stated in Tables 2 and 3 that we repeat for readers convenience:
• Low client complexity (U₁)
• Client shall be in full control of query (U₄)
• Low response latency (U₃)
• Complete and correct query result (U₅)
• Data ownership for information holder (U₆)
• Synchronous response to client (E₁)
• Manageable access control (E₂)
• Using existing infrastructure (E₃)
• Highly scalable design (E₄)

Table 4. Fulfillment of Selected Requirements by Different Discovery Service Approaches

<table>
<thead>
<tr>
<th>Selected Requirements</th>
<th>U₁</th>
<th>U₂</th>
<th>U₃</th>
<th>U₄</th>
<th>U₅</th>
<th>E₁</th>
<th>E₂</th>
<th>E₃</th>
<th>E₄</th>
</tr>
</thead>
<tbody>
<tr>
<td>DS</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>*</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>QR</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>*</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>DC</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>ADS</td>
<td>*</td>
<td>-</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
</tr>
</tbody>
</table>

4.2.1. Directory Service (DS) Approach. Following the Directory Service approach the response of a client query is a list of EPCIS URLs. The client then queries the respective EPCISs by himself. This adds complexity to the client application because the client has to implement necessary parallelization of requests and the appropriate aggregate the EPCISs responses (U₁) [16, Section B, p. 26]. The client is in complete control of which EPCISs he queries (U₄). Complexity at Discovery Service level is low and therefore response latency should be low (U₃). The result of the Discovery Service following the Directory Service approach is correct but not complete (U₆). It only contains pointers to EPCISs, not the information the client needs. Information holders are not in full control of their EPCIS URL, which can be seen as critical information (U₅).

The Directory Service approach proposes a synchronous response to the client query (E₁). Fine-grained access control information is duplicated from EPCISs to the Discovery Service level to assure that EPCIS URLs are not revealed to untrusted parties. This induces further complexity because large numbers of records containing access control information have to be stored and evaluated for each client query. This data furthermore requires continuous synchronization with the respective EPCIS (E₂). The Directory Service approach uses the existing infrastructure (E₃). The overall design can be judged as scalable (E₄).

4.2.2. Query Relay (QR) Approach. In the Query Relay approach the client must be able to receive data from multiple previously unknown sources without knowing the exact number of responses. This results in the need for a complex software design (E₅) that has to handle multiple incoming connections for a single request. Furthermore the client has to aggregate the EPCIS responses by himself. Given the fact that the client query is forwarded to respective EPCISs the client is not in full control of his query (U₅). He cannot cancel the request or deny that his query is forwarded to a specific EPCIS which might be a competitor’s EPCIS. A client querying a Discovery Service implemented according to the Query Relay approach is not aware of any information related to his query. EPCISs might have slow response times, deny a response to his query or be temporarily not available. This leads to a situation that the client has to wait for a substantial amount of time to be sure that each EPCIS that replies to his query had the chance to do so. Therefore the client has to wait until a timeout is reached. This stands in contrast to a low response time (U₃) and even indicating a number of relevant EPCISs to the client as proposed in [16, Section B, p. 30] does not solve the solution and would break information holders’ confidentiality. The result of a client query is complete and correct (U₄) if and only if the client waits long enough to assure that no more replies are still underway. The client has no indication if EPCISs are temporarily unavailable. The data ownership of the information holder is ensured given the fact that he can decide which queries to reply to (U₅).

The communication between Client and EPCISs is implemented asynchronously (E₅). Access control information is manageable in this approach because no information is directly returned to the client (E₆). The request-response-flow does not comply with standard Web Service design (E₃). Accepting unsolicited incoming data from unknown sources might expose the client to a variety of security vulnerabilities and will likely cause problems with firewalls [16, Section B, page 34]. Furthermore the Query Relay approach has to extend the EPCIS standard [4] because the clients’ address has to be included in the query to facilitate the direct response of EPCISs. The design of the Query Relay is scalable (E₄) because simple database lookups at Discovery Service level are sufficient.

4.2.3. Daisy Chain (DC) Approach. Huang et al. [11] do not specify the concrete request response model for the Daisy Chain approach. The client could receive a combined result containing information from all relevant EPCISs or each EPCIS could reply directly like in the Query Relay approach. We assume that the information is collected and then synchronously returned to the client (E₅). Therefore client complexity is low (U₁). The client query is forwarded to the relevant EPCISs and therefore the client is not in full control of the query (U₅). Traversing all EPCISs in the
supply chain for a given EPC cannot be parallelized and therefore leads to high latency because each EPCIS has to be queried sequentially (U₃). The distributed hash table used by Huang et al. to overcome the problem of broken links from one supply chain partner to another might generate the same key for different EPCs. Therefore the result might not be correct (U₄).

Data ownership for the information holder is assured in this approach (E₁). The proposed Daisy Chain approach relies on rewritable RFID tags. Data matrices are not rewritable and cannot be used with this approach (E₂). Given the fact that no central instance is needed, the access control does not have to be managed (E₃) and the approach is scalable (E₄).

4.2.4. Aggregating Discovery Service (ADS) Approach. The ADS approach shifts the complexity (U₁) of query parallelization and the aggregation of EPCIS responses from client to server and acts like an EPCIS by creating a view of the relevant information for the client.

Once the ADS receives a query it immediately forwards it to all relevant EPCISs and the client is no longer in control of the query (U₂). If role-based access control is enforced at EPCIS level this should not be an issue because the client identity is not revealed to the information holder. Furthermore the client could define a blacklist or white list to further improve client confidentiality like proposed in [16, Section B, p. 32]. The response latency is low having in mind that a complete result set is returned (U₃). The ADS efficiently queries the relevant EPCISs in parallel and aggregates the subresults. Because a Discovery Service provider is trustworthy, EPCISs which have relevant information but decline to share it [16, Section B, p. 32] can send a ‘deny’ message to the ADS without being revealed. The result of the ADS response is complete and correct while respecting clients’ access rights (U₄). If EPCISs are temporarily not available the ADS indicates this to the client.

The ADS ensures full data ownership of the information holder (U₅), responds synchronously (E₁), uses existing infrastructure, and leaves access control information with the supply chain partners thus reducing access control complexity (E₂).

According to [16, Section B, p. 32 et seq.] each Discovery Service approach should be scalable in terms of state information that has to be managed and many simultaneous client communications. Despite the fact that we have to maintain more state information than other approaches and deal with a substantial number of simultaneous clients we claim that the ADS is scalable. To show the scalability in a real world environment we discuss the scalability in the next section.

4.3. Scalability

The ADS provides additional functionality which requires more computing power than the ‘Directory Service’, ‘Query Relay’ or ‘Daisy Chain’ approach. Like stated before, the ADS has to wait for all responses of the subqueries, thus maintaining a connection’s state for the request-response cycle with the client. In this section we show that it is possible to implement a scalable Aggregating Discovery Service.

We exemplify the potential load for a Discovery Service in the U.S. pharmaceutical supply chain by a back-of-the-envelope calculation. Following the supply chain network model of Williams et al. [23] a Discovery Service has to deal with 1,000 notifications per second at peak times and 200 queries per second in average. We assume the worst-case scenario that supply chain partners conduct a query for each item they notify as indicated by [21]. The ADS therefore has to deal with 1,000 queries per second at peak times and 200 queries per second in average. As the authors state a supply chain does not exceed 15 partners.

4.3.1. Load Balancing and Data Partitioning. Distributing all incoming notification messages and client queries to many self-contained application servers allows HTTP services, including the ADS, to scale very well. HTTP load balancing can be performed in both hardware and software for very high connection speeds. Additional servers can be added at any time allowing the system to grow in size.

HTTP reverse proxy servers balance incoming HTTP queries but operate at a higher level in the network stack. They accept incoming HTTP connections and are able to act based on queried URL or even arguments in the HTTP request. Implementations like the event-driven nginx² can help lower the CPU load on application server machines by mapping requests to a specific EPC to one specific server. Each server is then responsible for a range of EPCs, implementing partitioning at the application server and database tier.

Client queries always refer to one or more EPCs. No single database query will ever need to JOIN any data with rows for other EPCs. Database queries will only perform index lookups for EPCs and return the corresponding EPCIS URLs. This allows the database to scale horizontally by partitioning all data by EPC. Every database server is then responsible for serve requests for a range of EPCs. The database lookup only consists of small queries requiring basic database functionality. There is no need for complex locking mechanisms because all data has to be stored persistently (no tuples will be deleted or updated) and there is no need for synchronizing database partitions.
This even allows the usage of very basic database systems optimized for a high throughput of short queries.

4.3.2. Open Connections. As depicted above, it is assumed that the Discovery Server has to handle about 1,000 client queries per second. For a supply chain with \( n = 12 \) enterprises in average this results in \( n \times 1 = 6 \) relevant EPCISs in average. This results in 6,000 subqueries per second. Assuming a query to an EPCIS is replied to in one second on average the system has to hold about 6,000 connections simultaneously.

We tested how many connections one commodity PC is able to hold. To simulate a real-world scenario we requested 30,000 random Websites we gathered by querying a search engine with random keywords. All DNS resolving was done before starting the test at a limited upstream speed of 1 Mbit/s and 2.4 GHz CPU speed. Using asynchronous I/O processing we were able to have a single-threaded Python script sustain 3,000 connections (1,100 active) while using 22 to 24% CPU. A low number of commodity-level servers can easily handle the total amount of connections.

4.3.3. Bandwidth. In the basic ‘Query Relay’ architecture, the queried EPCISs reply directly to the client. In comparison, the ADS is the single response endpoint for all sub-queries. The ADS has to be able to cope with 1,000 incoming client requests at peak times like described before. The ADS has to send 6,000 subrequests and to receive 6,000 subresponses, all with a size at or below one kilobyte. Returning 1,000 aggregated responses adds another 6,000 kilobytes every second. At this point the server receives 56 Mbit/s and sends 96 Mbit/s, which is perfectly feasible using available Internet connections.

4.3.4. XML Handling. All replies sent back from EPCISs to the ADS use the XML format standardized by EPCglobal. It wraps all ObjectEvents in a single EventList [4]. XML parsers optimized for high throughput provide efficient functionality for aggregating these XML responses. SAX or Pull Parsers have proven their efficiency in SOAP environments where a large number of small XML queries have to be processed [2]. While receiving the XML data stream from a responding EPCIS every parsed tag inside the EventList can instantly be created on the output stream that will be sent back to the client. This eliminates the need to add further buffers for XML objects and reduces XML rendering time.

5. Summary and Future Work

In this contribution we introduced the relevance of a ‘Discovery Service’ in the EPCglobal network to identify potentially unknown supply chain partners, i.e., to enable data sharing to generate business value or comply with legal requirements. We then presented related work and selected requirements structured by the ISO/IEC 9126 standard. The comparison of our Aggregating Discovery Service with existing Discovery Service approaches (see Table 4) highlights the advantages of our approach. It rectifies the problems of other approaches and in Section 4.2.4 we also showed how to adapt the ADS to enable the client to be in full control of his query. Since our approach is based on a centralized component the question of scalability arises. The arguments provided in Section 4.3 show that our solution can be scaled in multiple ways and can be implemented in a large-scale real-world environment.

We see much future work on Discovery Services and beyond. It might become the fact that multiple Discovery Services will be in place for distinct areas of application. The fast moving consumer goods supply chain then would use a different Discovery Service than the pharmaceutical supply chain. This leads to challenges like finding the right Discovery Service for a particular request [17]. Given the fact that supply chains are not completely isolated, e.g., pharmaceuticals are sold at a retailer that also sells fast moving consumer goods. This leads to a situation where multiple Discovery Services are involved to completely fulfill the request. These Discovery Services probably have to inform each other in a peer-to-peer manner while respecting privacy concerns.

We also see future work in an in-depth analysis in the aforementioned discovery service approaches with regards to the mentioned requirements (Tables 2 and 3) and technical aspects, e.g., scalability, network utilization, database utilization, and performance. For this, a standardized benchmark and a sophisticated test infrastructure have to be developed to assure reliable and reproducibility. First steps towards a benchmark have been taken in [15] but this work has to be extended towards a standardized benchmark.

Privacy and security play a very important role as the resistance with regards to the ONS shows. Therefore, sophisticated and efficient security methods have to be discussed and implemented for each of the Discovery Services described in this paper. As a matter of fact, this adds overhead to the EPCglobal network.

Another problem we realized in our projects while implementing the Aggregating Discovery Service is how to efficiently implement the notification messages. In our solution the middleware processing a read event uses a rule engine to decide if the EPCs of the current read event is of relevance for the Discovery Service. A more sophisticated and non-proprietary approach would be a subscription-based notification. The Discovery Service thereby subscribes to events at the EPCISs and gets notified by the EPCIS if relevant
information is available. To implement a subscription-based notification efficiently, the Discovery Service would have to define detailed attributes it is interested in. This is work to extend the EPCIS standard.

Finally, the presented ADS has to be able to track changes of aggregation and containment, which is future work.

References


