RFID Event Data Processing: An Architecture for Storing and Searching

Matthieu-P. Schapranow, Jürgen Müller, Alexander Zeier and Hasso Plattner

Hasso Plattner Institute, Enterprise Platform and Integration Concepts
August–Bebel–Str. 88, 14482 Potsdam, Germany

Abstract. The pharmaceutical industry suffers from increasing counterfeit rates. U.S. federal restrictions force manufacturers to guarantee product authenticity. RFID technology can be used as foundation for an integer and counterfeit-resistant pharmaceutical supply chain. We present an architecture optimized for storing and searching of pharmaceutical RFID data and share results of our main-memory-based prototype.

1 The Pharmaceutical Supply Chain

The European pharmaceutical industry hit the headlines with operation MEDI-FAKE announcing 34 million detected fake drugs in only two months [1]. The European Commission reports an increase of 118 % for pharmaceutical counterfeits detected at borders in 2008 compared to 2007. The pharmaceutical product category is the third largest product category in terms of quantities of intercepted articles besides the categories CDs/DVDs and cigarettes [2]. Counterfeited goods are a risk for customers and for suppliers, because their efforts is neither tested nor validated and the customer may suffer from various kinds of medical complications. In 2004, it was estimated that more than 500 billion USD were traded in counterfeits, i.e. 7 % of the world trade in the same period [3]. It is argued, that this equals an increase of 150 billion USD compared to 2001 [4]. During the same period, the worldwide merchandise trade increased only by approx. 50 billion USD. This example highlights the risks of counterfeits and the need for product protection. A high level of supply chain integrity is the basis for reliable product tracking to support counterfeit detection. The European pharmaceutical supply chain consists of approx. 2,200 pharmaceutical producers, 50 thousand wholesalers, and more than 140 thousand retailers [5]. Every supply chain participant stores events in a local event repository for any handled product. Reliable product tracking and tracing across the entire supply chain can be implemented with RFID [6], but this technique is not designed to be immunized against threats, such as cloning, spoofing or eavesdropping [7]. The pharmaceutical industry has to identify specific solutions addressing the product's authenticity while data integrity is ensured to shield the global pharmaceutical supply chain against intrusion of counterfeits. Enhancing existing supply chains by introducing RFID tags comes with various prevailing advantages [8], e.g. in contrast to barcodes, RFID tags can be read without establishing a direct line of sight, multiple

Schapranow M., MÃijller J., Zeier A. and Plattner H.
RFID Event Data Processing: An Architecture for Storing and Searching.
DOI: 10.5220/0003017100130022
In *Proceedings of the 4th International Workshop on RFID Technology - Concepts, Applications, Challenges (ICEIS 2010)*, page
ISBN: 978-989-8425-11-9

tags can be read simultaneously, and they can cope with dirty environments [9]. Building an industry-specific solution on top of an existing communication infrastructure is non-trivial, because the communication medium suffers from the amount of event data exchanged for anti-counterfeiting [10]. We propose an architecture which addresses the following challenges:

- small event data (approx. 100-200 bytes) to be stored in local repositories,
- hundreds of thousands of events captured hourly at partner sites,
- queries which need to be forwarded via a global communication infrastructure to decentralized EPC Information Services (EPCIS) repositories, and
- queries to be processed in sub-second time, because the authenticity of hundreds of products is checked simultaneously, e.g. during inbound processing.

The rest of the paper is structured as follows: In Sect. 2 our work is placed in context of related work. Architecture components are outlined in Sect. 3, while Sect. 4 explains their interaction. Sect. 5 describes benchmarking setup, results, and shares our measurement evaluation. Our paper concludes in Sect. 6.

2 Related Work

Fast storing and performing predefined search queries on RFID data is a current trend in research and various works address this topic. Acquiring real-world events by readers to map them to virtual product histories is the task of enterprise RFID middleware systems [11]. They support either one of both categories of RFID applications: history-oriented tracking and real-time monitoring. The latter requires sub-second response times, e.g. for incoming goods processing. Our Java prototype is optimized for real-time monitoring by leveraging very short response times for EPC-based queries. Product flow and temporal event queries are expected to be major queries in RFID event analysis [12]. Existing data stream management systems build on enhancements of query languages, e.g. SQL. Language enhancements for temporal queries exist to reduce complexity of temporal RFID exploration [13]. Special query templates exist for storing and retrieving flow and temporal data of a certain product [14]. They build on the separation of tag, path, and time data. All data is stored in separate Data Base (DB) tables stored in a centralized repository, but a decentralized application prototype was also proved to work [15].

Our architecture approach uses decentralized event stores, which can be uniquely identified by URLs using the Object Name Services [16]. It builds on a tree-based data structure comparable to the one used by the Domain Name Service (DNS). The DNS architecture supports distribution of responsibilities across hierarchy levels via dedicated network information centers [17]. Although a centralized instance reduces costs for maintenance, it introduces a single point of failure, e.g. when the system has to be placed offline. This requires redundant hardware and data to overcome extreme situations which doubles infrastructure costs at least. We combine the ideas of an EPC Discovery Service [18] and a DNS architecture. Queries are forwarded to decentralized EPCIS repositories to prevent load peaks that paralyze centralized architecture components. The huge amount of RFID data makes any kind of architecture fragile in terms of system load. Various researchers focus on data reduction, e.g. filtering duplicate and redundant RFID events which can be performed by enterprise middleware systems to reduce the total data load [19]. The cuboid architecture offers an implementation for partitioning RFID data [20], which can be enhanced to implement static load balancing policies. Once a product reaches the end consumer, associated events can be moved to slower storages to establish data retention management and to increase the system performance [21].

3 Architecture Components

The architecture components depicted in Fig. 1 are described in the following focusing on throughput effectiveness. To satisfy incoming queries, all EPC-related events need to be aggregated which involves the following steps:

– determination of involved EPCISs,
– query propagation to involved EPCISs,
– parallel and distributed processing of query at remote EPCISs,
– return of partial query results,
– aggregation of partial query results, and
– return of final query result to querying participant.

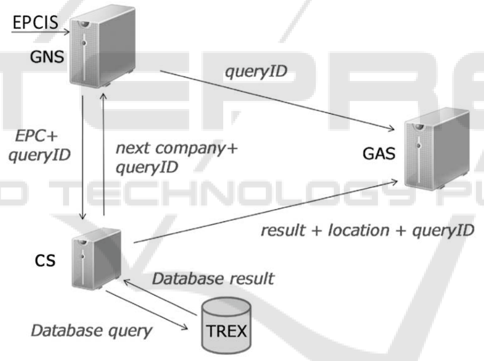

Fig. 1. Architecture: Infrastructure Components and Data Flow.

Queries are only forwarded to EPCISs that are involved in the supply chain path of a certain product. All EPCIS repositories offer interfaces to access their stored data in a unified way, regardless of internal format and vendor. Data unification is not part of the given architecture. The outlined algorithm is chosen to reduce processing time by enabling partial processing comparable to MapReduce [22]. By limiting the recipients for a query, the overall network load is kept minimal and the risk for congestion by flooding the communication network with RFID data is prevented [23].

The *Global Name Server* (GNS) is the central component of our architecture. Its implementation follows the concept of the DNS root servers [17]. The GNS receives an incoming query containing the product's EPC. Firstly, the Global Aggregation Server

(GAS) is notified and secondly, the incoming query is forwarded to servers at lower hierarchy levels, i.e. Company Servers (CSs) which have access to a single EPCIS repository or multiple EPCISs repositories of a company or department. Spanning a tree on top of the CSs hierarchy helps to identify the correct CS for a certain EPC in logarithmic time, i.e. O-complexity $O(\log n)$, instead of performing a long sequential scan on a list of servers with $O(n)$. Our GNS implementation differs from a DNS root server, because query results from servers at lower hierarchy levels are received to enable processing of the next query. If one EPCIS repository declines information, e.g. because of insufficient access rights, the requester is informed. Access rights are implemented and managed by each EPCIS repository separately to guarantee data protection. We consider EPCIS repositories as standardized products, e.g. the open source RFID platform fosstrak[1]. Their design is not discussed due to page limitations.

The *GAS* is responsible for collecting partial query results generated by the CSs, it is queried by the GNS and receives partial query results by CSs. To save a list of known companies and to store partial query results received from the CS, the GAS allocates memory for a temporary session to store the query [18]. It can perform additional plausibility checks for anti-counterfeiting, but this is not an integrated part of our architecture as we encourage counterfeit detection to be implemented by dedicated service providers.

The *CS* has direct access to the EPCIS repositories of the company or department which contains event data. We decided to use the SAP TREX Main Memory (MM) DB to store events. Instead of loading all attributes, the columnar DB layout enables loading of affected attributes, e.g. only EPC and timestamp to answer the following query:

”When did the product with the given EPC leave your company?”

The results are forwarded to the GAS. Each supply chain participant knows details about predecessors and successors of a certain product. This information is combined with the data received from the EPCIS and returned to the GNS. The GNS restarts its algorithms and forwards the initial query to the next participant in the supply chain. Starting from the last participant traversing back to the producer the same steps are performed to improve processing time. The last participant in the supply chain is defined by the querying party. If the querying party is an intermediate and the product was already passed to the next participants the chain has to be reconstructed by the querying party only in one-way traversing to the end of the chain. By executing both queries in parallel, processing time can be divided by two if the querying party holds the product. If the communication network is segmented at a single site or partially unable, this approach can be used to reconstruct the chain around the unresponsive participant, because the query would stop from both ends at the same participant. If multiple participants are unresponsive or multiple paths are unavailable, the outage cannot be bridged and an incomplete result is indicated.

SAP TREX is a columnar MMDB supporting queries to be performed directly on compressed data in MM [24]. Using a MMDB benefits in instant responses when querying a specific EPC. We simulated event data for the pharmaceutical supply chain. Each event consists of the following attributes: `EPC`, `readpoint`, `time`, `action`, `disposi-`

[1] http://www.fosstrak.org/

Table 1. Test Data Scenarios.

	# Events	Size on Disk	Size in RAM
Scenario I	≈1.1 M	≈200 MB	≈20 MB
Scenario II	≈6.3 M	≈1150 MB	≈120 MB

`tion`, `bizlocation`, `bizstep`. The EPC is stored as serialized global trade identification number and the locations (`readpoint` and `bizlocation`) are stored as serialized global location number [25]. The remaining attributes are used for anti-counterfeiting by service providers.

Our simulated data is characterized in Tab. 1. Scenario I consists of 1.1 M events, consuming 200 MB on disk and 20 MB of RAM once the data is loaded and compressed by SAP TREX. Scenario II is 6-times larger and consists of 6.3 M events, 1150 MB on disk and 120 MB in RAM. Both scenarios were compressed 10:1 by SAP TREX when loaded into MM. We identified the mean size of a single event with circa 182 byte uncompressed resp. 18 byte compressed, which helps to estimate storage demands for the global pharmaceutical supply chain.

The *Service Provider* performs plausibility checks for anti-counterfeiting with the product's virtual product history. All supply chain participants have to agree on common access rights for this instance to process any request. If a counterfeit is identified, the service provider returns $counterfeit$ and the product is removed from the supply chain. If the virtual product history is valid, $authentic$ will be returned. If the result of the counterfeit detection is inconclusive, e.g. during network partitioning, $unknown$ will be returned to indicate the need for further processing. We define a function $service_{counterfeit}$ in Equation 1 describing the service provider which performs checks with the help of a given epc_i. It returns either one of the results authentic a, counterfeit c, or unknown u.

$$service_{counterfeit} : epc_i \mapsto (epc_i, \{a, c, u\}) \tag{1}$$

4 Performance Aspects

A product query is created and sent to the GNS, which registers the request and generates a unique `query ID`. With the help of the ID, the GAS is contacted and creates a temporary session to hold responses. The query results received from the CS are assigned by the GAS to the corresponding session. The EPC consists of an encoded company prefix, an encoded product reference, and a serial number [26]. The GNS contacts the producer of a certain product. The initial query and the `query ID` are forwarded to the CS. The CS performs two DB lookups: the first returns the supply chain successor the product was passed to, the second performs the existence check for events associated with the given EPC. Results of the first lookup are used for the following next-hop algorithm. This separation improves response time, because looking up events associated with a certain EPC takes longer time than determining the next participant in the supply chain. The next CS can be queried autonomously to start the next lookup; the algorithm continues until no further successors exist. The GAS receives a notification for anti-counterfeiting checks consisting of:

1. the `query ID` for mapping the query result to the corresponding query,
2. the `company ID` of the current CS, and
3. an indicator for the EPC's existence in the local event store of the company.

4.1 Network

During the development, we switched the communication protocol from Transmission Control Protocol (TCP) to User Datagram Protocol (UDP). This was initiated to handle numerous messages which are exchanged between GNS, CSs, and GAS. We determined that a significant amount of time was spent for TCP's three-way handshake [27]. We assume modern communication infrastructures to be reliable, i.e. lost packets during query retrieval are retransmitted after timeouts. If duplicated results are received, they are distinguished by the GAS using the `company ID` and `query ID`. If expected responses are not received, the return value will be marked to be incomplete. This way, network segmentation and missing data can be identified without creating false-positives.

4.2 Sequence of Company Actions

Besides the selected communication protocol we figured out that the sequence of performed DB queries has an impact on the response time. We decided to implement two different action settings as depicted in Fig. 2. Both action settings differ in the operation of the CS. We focused on the CS because this component is involved multiple times during a trace for a single product. Our architecture builds on two DB queries which are performed by the CS. Action setting I in Fig. 2 shows, that both queries are performed simultaneously in step 5. Once both results are returned, the CS sends one part to the GNS and the other to the GAS. Fig. 2 outlines for action setting II that the first query retrieves details about the successor of a product. In contrast to action setting I, the GNS is now informed immediately in step 4 and in step 5 the second query is performed by the CS while the GNS can proceed traversing the trace. The result of the second query is sent asynchronously to the GAS in step 6.

5 Timing Results

We share insights about our 4 implementation combinations consisting of 2 communication protocols and 2 action settings in the following.

SAP TREX DB was configured to run in a non-distributed environment. It was installed on a single IBM Blade HS 21XM Type 7995 running SuSE Linux Enterprise 10 SP 2 with the 64-bit kernel 2.6.16.60-0.21-smp. It was equipped with two CPUs Intel Xeon E5450 CPUs, each running at 3.0 GHz clock speed consisting of four logical cores with 32 kB L1 cache, 12 MB L2 cache. The system was equipped with eight 4 GB DIMM DDR modules to form a total of 32 GB RAM. We measured the impact of the DB size on the response time.

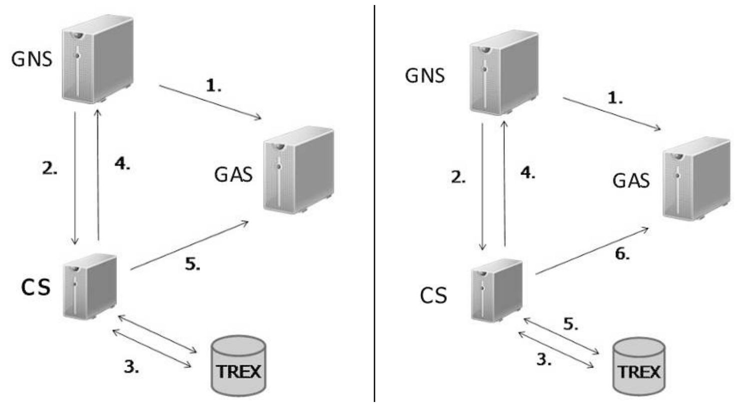

Fig. 2. Action Setting I (left) and Action Setting II (right).

Our results showed, that the next company query returns in avg. in 11 ms and the result query in avg. in 10 ms. These timing results are scenario-independent as shown in Fig. 3 (b) for bars TCP 1 and UDP 1. This underlines the instant response time which is achieved by the SAP TREX MMDB in the given enterprise scenario. The RAM size in the blade systems exceeds the amount of used simulated data, i.e. the entire data sets of both scenarios fit completely in MM and can be accessed with MM access time.

The Network was benchmarked by installing GNS and GAS on workstations running Microsoft Windows 7. They were equipped with a single CPU, two cores running at 2.6 GHz clock speed, 4 GB RAM, and Gigabit network interface cards. All systems were connected through an unmanaged Gigabit network switch. The network throughput of our prototype is addressed by measuring the connection time, i.e. the time needed to transfer messages between peers. The connection time for the communication protocols TCP and UDP are compared.

Our measurements showed that each data transfer using TCP needs 6 ms in avg. whereas UDP is approx. 6-times faster with an average of 1 ms. With TCP one company needs 41 ms in avg. to complete query processing. Fig. 3 (a) on the left shows the cumulated results which consist of the connection time between GNS and CS (6 ms), processing the DB queries for the next participant (11 ms) and the query for detecting the EPC in the local EPCIS (10 ms), evaluating the next company (2 ms), and two connection times: to the GNS (6 ms) and to the GAS (6 ms). Switching our prototype to use UDP as communication protocol reduced the connection time to avg. 1 ms. We were not confronted with unordered or lost packets, which might occur in more complex routed networks. The complete processing time when using UDP is reduced to an average of 26 ms as depicted in Fig. 3 (a) on the right.

Architecture Configurations were implemented as a configurable CS to switch between action settings. Action setting I uses two subsequent DB queries. From the GNS's point of view using action setting I showed to consume 35 ms in avg. with communication protocol TCP and 25 ms with communication protocol UDP as shown in Fig. 3 (b)

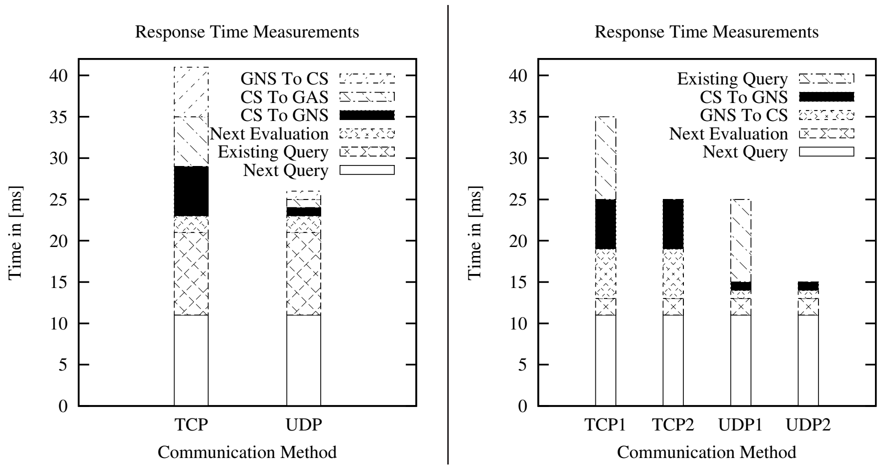

Fig. 3. Comparison of Communication Methods (a) and Configurations (b).

for TCP1 and UDP1. These response timings occur, because result transmission to the GAS is postponed until both DB results are available. By varying query processing in action setting II, it is possible to reduce processing time at the CS by avg. 10 ms. Once the next company is determined, it is immediately sent to the GNS. The GNS receives the result without the need to wait for the second DB query to complete. Using the communication protocol TCP in this case results in avg. 25 ms processing time. The communication protocol UDP consumes 40 % less processing time with 15 ms in avg. as depicted in Fig. 3 (b) for UDP2. Our timing results show, that the GNS is able to process one CS within 15 ms in avg. depending on the architecture configuration. By switching to action setting II using UDP, we were able to reduce processing time by approx. 60 % compared to action setting I using TCP.

6 Conclusions

In the given paper, we shared our insights gathered by the implementation of a Java prototype designed for storing and searching of RFID event data. Our work is motivated by the need for counterfeit detection in the pharmaceutical industry, because of an increasing number of fake drug cases. Our proposed architecture consists of GNS, GAS and numerous decentralized CSs connected to EPCICs. For storing RFID event data we used the SAP TREX MMDB to leverage instant response times for real-time monitoring. To perform centralized anti-counterfeiting, we query all participants involved in the product lifecycle for a certain product to return event details. A linked list of companies is traversed in two directions to save time and to reconstruct single network segmentations. The amount of messages traveling through the communication network is minimized by querying only involved supply chain participants instead of performing broadcasts to all participants.

We outlined different architecture configurations, gave details about timing results and characterized the optimal setup for our prototype. Using action setting II with communication protocol UDP enables us to contact each involved participant in the product's lifecycle in avg. 15 ms which is 60 % faster than action setting I with communication protocol TCP. We observed that our simulated RFID events consume in avg. 18 bytes when stored by SAP TREX compressed in memory. For the European pharmaceutical supply chain, we assume 1 producer, 4 wholesalers, 1 retailer, and 1 end consumer. With approx. 15 billion prescriptions per year and 9 read events – 2 per wholesaler and 1 per retailer – an annual amount of approx. 2.3 TB of compressed event data needs to be stored by all decentralized SAP TREX instances.

References

1. IP Crime Group: IP Crime Report (2008)
2. European Commission Taxation and Customs Union: Report on EU Customs Enforcement of IP Rights (2009)
3. ICC Policy Statement: The Fight against Piracy and Counterfeiting of Intellectual Property. http://www.iccwbo.org/home/intellectual_property/fight_against_piracy.pdfMar. 8, 2010 (2003)
4. Staake, T., Thiesse, F., Fleisch, E.: Extending the EPC Network: The Potential of RFID in Anti-Counterfeiting. In: Proc. of the ACM Symposium on Applied Computing, New York, NY, USA, ACM (2005) 1607–1612
5. Müller, J., Pöpke, C., Urbat, M., Zeier, A., Plattner, H.: A Simulation of the Pharmaceutical Supply Chain to Provide Realistic Test Data. In: Proc. of the Internattional Conference on Advances in Syst. Simulation. (2009)
6. Bundesverband Informationswirtschaft, Telekommunikation und neue Medien: White Paper RFID Technologie, Systeme und Anwendungen (2005)
7. Schapranow, M. P., Müller, J., Zeier, A., Plattner, H.: Security Aspects in Vulnerable RFID-Aided Supply Chains. In: Proc. of the 5th European Workshop on RFID Syst. and Technologies. (2009)
8. Stiehler, A., Wichmann, T.: RFID im Pharma- und Gesundheitssektor. Vision und Realität RFID-basierter Netzwerke für Medikamente. Berlecon Report (2005)
9. White, G., et al.: A Comparison of Barcode and RFID Technologies in Practice. Journal of Inf., Inf. Technology, and Organizations 2 (2007)
10. Müller, J., Schapranow, M. P., Helmich, M., Enderlein, S., Zeier, A.: RFID Middleware as a Service – Enabling Small and Medium-sized Enterprises to Participate in the EPC Network. In: Proc. of the 16th Int'l Conference on Ind. Eng. and Eng. Mgmt. Volume 2. (2009) 2040–2043
11. Wang, F., Liu, S., Liu, P., Bai, Y.: Bridging Physical and Virtual Worlds: Complex Event Processing for RFID Data Streams. In: Advances in Database Technology. Volume 3896/2006., Springer Berlin / Heidelberg (2006) 588–607
12. Fazzinga, B., Flesca, S., Masciari, E., Furfaro, F.: Efficient and effective RFID data warehousing. In: Proc. of the Int'l Database Eng. Applications Symposium, New York, NY, USA, ACM (2009) 251–258
13. Bai, Y., et al.: RFID Data Processing with a Data Stream Query Language. In: Proc. of the 23rd Int'l Conference on Data Eng. (2007) 1184–1193
14. Lee, C. H., Chung, C.W.: Efficient Storage Scheme and Query Processing for Supply Chain Management using RFID. In: Proc. of the Int'l Conference on Management of Data, New York, NY, USA, ACM (2008) 291–302

15. Cheung, A., Kailing, K., Schönauer, S.: Theseos: A Query Engine for Traceability across Sovereign, Distributed RFID Databases. In: ICDE, IEEE (2007) 1495–1496
16. Chawathe, S. S., Krishnamurthy, V., Ramachandran, S., Sarma, S.: Managing RFID Data. In: Proc. of the 30th Conference on Very Large Data Bases, Toronto, Canada, VLDB Endowment (2004) 1189–1195
17. Mockapetris, P., Dunlap, K. J.: Development of the Domain Name System. SIGCOMM Comput. Commun. Rev. 25 (1995) 112–122
18. Müller, J., Oberst, J., Wehrmeyer, S., Witt, J., Zeier, A.: An Aggregating Discovery Service for the EPCglobal Network. In: Proc. of the 43th Hawai'i Conference on Syst. Sci., Koloa, Hawaii, USA (2010)
19. Bai, Y., Wang, F., Liu, P.: Efficiently Filtering RFID Data Streams. In: CleanDB, Seoul, Korea (2006)
20. Gonzalez, H., Han, J., Li, X., Klabjan, D.: Warehousing and Analyzing Massive RFID Data Sets. In: Proc. of the 22nd Int'l Conference on Data Eng., Washington, DC, USA (2006) 83
21. Wang, F., Liu, P.: Temporal Mgmt. of RFID data. In: Proc. of the 31st Int'l Conference on Very Large Data Bases, Norway, VLDB Endowment (2005)
22. Dean, J., Ghemawat, S.: MapReduce: Simplified Data Processing on Large Clusters. Commun. ACM 51 (2008) 107–113
23. Stallings, W.: Data and Computer Communications. 8 edn. Prentice Hall (2007)
24. Schaffner, J., Bog, A., Krüger, J., Zeier, A.: A Hybrid Row-Column OLTP Database Architecture for Operational Reporting. In: Business Intelligence for the Real Time Enterprise, Auckland, New Zealand (2008)
25. EPCGlobal: EPCIS Standard 1.0.1. http://www.epcglobalinc.org/standards/epcis/epcis_1_0_1-standard-20070921.pdfMar. 8, 2010 (2007)
26. EPCGlobal: Tag Data Standard 1.4. http://www.epcglobalinc.org/standards/tds/tds_1_4-standard-20080611.pdfMar. 8, 2010 (2008)
27. Miller, P.: TCP/IP - The Ultimate Protocol Guide - Data Delivery and Routing. Volume 1. Universal-Publishers (2009)