IEICE TRANS. INF. & SYST., VOL.E93–D, NO.11 NOVEMBER 2010

PAPER Special Section on Architectures, Protocols, and Applications for the Future Internet

Otedama: A Relocatable RFID Information Repository Architecture

Shigeya SUZUKI[†a)], ***Member***, **Rodney VAN METER**[††], ***Nonmember***, **Osamu NAKAMURA**[††], ***and*** **Jun MURAI**[††], ***Members***

SUMMARY We present a novel RFID middleware architecture, Otedama, which makes use of a unique property of RFID information to improve performance. RFID tags are bound to items. New information related to an RFID tag is generated at the site where the ID exists, and the entity most interested in the history and the item itself is in close proximity to the RFID tag. To exploit this property, we propose a scheme which bundles information related to a specific ID into one object and moves that bundle to a nearby server as the RFID tag moves from place to place. By using this scheme, information is always accessible by querying a system near the physical location of the tag, providing better query performance. Additionally, the volume of records that must be kept by a repository manager is reduced, because the relocation naturally migrates data away as physical objects move. We show the effectiveness of this architecture by analyzing data from a major retailer, finding that information retrieval performance will be six times better, and the cost of search is possibly several times cheaper.

key words: *RFID, middleware, intelligent storage, reactive agent*

1. Introduction

In this paper, we propose a novel information relocation and search scheme for RFID information systems that provides better search performance and resource utilization than currently standardized and proposed schemes.

RFID, an automatic-identification technology using Radio Frequency, has been developed for different application areas [1]. Recently, several different standards have been developed [2] and used in many application areas [3]. Figure 1 depicts a supply chain item flow with contexts captured by an RFID system. An item travels from the factory, through several intermediate sites such as distribution centers (DCs), finally gets to a store, and is sold to a consumer. The RFID tag and related information allows software to fulfill a variety of business functions.

Since around the year 2000, networked RFID systems have received a lot of attention [4]. Such systems use the ID number recorded on a tag as a hint to locate and access information held in networked databases. By limiting the functionality required of the tag, a networked system allows the tag manufacturer to produce tags at very low cost even as the system functionality improves.

Manuscript received March 6, 2010.
Manuscript revised June 20, 2010.

[†]The author is with the Graduate School of Media and Governance, Keio University, Fujisawa-shi, 252–8520 Japan.
[††]The authors are with Faculty of Environment and Information Studies, Keio University, Fujisawa-shi, 252–8520 Japan.
a) E-mail: shigeya@wide.ad.jp

DOI: 10.1587/transinf.E93.D.2922

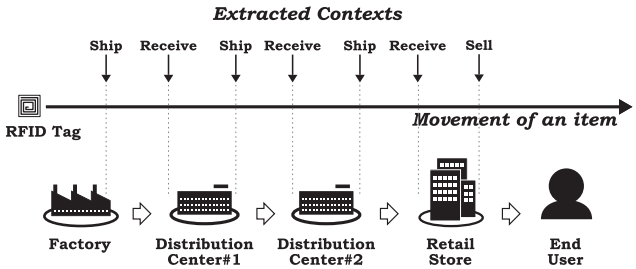

Fig. 1 RFID and contexts in supply chain management.

The information typical RFID equipment provides is the observed ID of an RFID tag [*]. By combining the ID information and some contextual information about the RFID equipment – where it is, and when it read the ID – software can capture meaningful information, such as the time the item with the RFID passed a specific gate. At each site in Fig. 1, deployed equipment will generate raw RFID information. Upon reception of raw RFID information, preconfigured software components will capture contexts and store the contexts to databases, which are accessed by other business software components.

There are several peculiarities of an RFID system:

- because it is designed to detect the movement of the objects among physical locations, the system must be distributed;
- information generated by the equipment is also distributed; and
- to track an item among these locations, we need to integrate information generated among distributed sites.

To handle these issues, RFID middleware provides two functions: a distributed repository which stores the information, and a mechanism to discover the relevant repository locations based on a tag's ID.

The first industry-defined middleware mechanism, Object Naming Services (ONS) [5], stores the information at a single repository which is statically mapped from an ID. That repository will normally be operated by the manufacturer of the item to which the tag is attached. Another mechanism currently in development, Discovery Services (DS) [6], requires each organization in the supply chain to operate a repository, which stores the information gener-

[*]Some RFID tags, also have user memory, but the use of that memory is beyond the scope of this paper.

ated at its sites. A Discovery Service, which locates these repositories based on ID, is provided by one or more service providers, which are similar to Internet search engines.

Both ONS and DS have limitations. ONS expects to use a single repository for a specific ID range, requiring the manufacturer to operate the repository indefinitely, since the optimal data retention duration is unknown. This raises concern about the scalability of the repository. DS expects that there will be multiple repositories operated by possibly different organizations. When a client wants to find information, it must issue requests to multiple servers in different locations, which requires far more computing resources. The DS scheme also cannot define a retention policy.

We propose a novel approach to resolve these issues, making use of the fact that the entity most interested in the RFID information about an item is usually physically located near the item. We use an information placement strategy which relocates or replicates the information related to a specific ID alongside the actual item as it moves between sites. By using this strategy, the number of requests required to find records can be smaller and response latency can be lower. Also, since each repository can monitor the level of interest for specific records bound to an ID, it can decide when to delete unused records. This policy will effectively provide better query performance and better resource utilization.

The contributions of this paper are:

- to point out that the entity most interested in a specific RFID tagged item is often near the physical item, providing a basis for improved automatic retention decisions;
- to propose a possibly faster, resource efficient storage system matched to the information scheme by using this fact; and
- to describe a new scheme that relocates information between sites while implementing access control, with the help of a digital rights management (DRM) mechanism.

This paper is structured as follows: in Sect. 2, we detail the current architecture and its issues, in Sect. 3, we describe our proposal in detail, in Sect. 4, we evaluate the effectiveness of our proposal, then in Sect. 5, we discuss related work. Finally, we wrap up our discussion in Sect. 6.

2. EPCnetwork Architecture and Issues

In this section, we introduce the de-facto standard EPCnetwork architecture and its issues. To clearly state our focus, we first describe the generic components of a network RFID system, then continue on to the specifics of the EPCnetwork system.

2.1 Motivation: Data Visibility

The motivation to use RFID is to improve visibility of the movement of products in a supply chain. Track and Trace ("where is my stuff?" and "where has my stuff been?") is the major application use case of the EPCnetwork system. To achieve this, sites participating in the supply chain will collect information generated by RFID equipment and provide that information to interested parties (clients). In the Track and Trace (TnT) scenario, a client will request information bound to a specific ID, called an EPC (Electric Product Code) [7].

2.2 Components of a Networked RFID System

A networked RFID system comprises the following six components:

Identification Number Scheme The numbering scheme for an ID imprinted on an RFID tag, used as the primary key of a networked RFID system.

RFID Event Capturing RFID is a sensor system which captures information from the physical world. This component, with the help of some additional equipment, captures events generated by communication between an RFID tag and an RFID reader and provides information to software components.

Context Capturing An RFID event – observation of an ID at a specific time on a specific reader – itself does not provide any meaning. By combining RFID events with other information, such as placement of the RFID reader, the system can infer meaningful context such as "item A has moved from room X to room Y."

Repository Once meaningful contexts are captured, they can be stored to a repository for later reference. A repository can be either distributed or centralized. A repository may contain some extra information related to business logic.

Interfaces Captured contexts can be provided to other systems such as business information system through defined interfaces.

Discovery If repositories are distributed among sites, there must be a mechanism for discovering the set of repositories relevant to a given search criteria.

2.3 Data Model of a Networked RFID System

RFID-related information can be divided into three categories:

RFID Event An RFID Event is information directly generated by the RFID equipment with the minimum amount of contextual information (placement of the reader and time of read), e.g., "EPC x.y.z is observed at reader A at time T."

Captured Context Meaningful context is captured from single or multiple RFID Events along with other site specific or contextual information, e.g., "EPC x.y.z passed docking door D," which is captured by adding the information that reader A is at docking door D.

External information In addition to the captured context,

with the help of some business information, the system can determine some business context, e.g., "EPC x.y.z has been shipped (because it passed docking door D)." External information may also include database keys which interface with external business systems.

The minimum information which RFID systems are required to support is RFID Event and Captured Contexts, a combination we call "RFID Information." Note that, in reality, it is impossible to perfectly read RFID tags every time, and the deployment environment can be more complex. Some filtering mechanism with multiple readers is needed, but for brevity of discussion, here we assume perfect read.

2.4 EPCglobal Network Standards

The EPCglobal architecture consists of several standards [†]. The following standards relate to this paper: the Tag Data Standard specifies the Identification Number Scheme; the Low Level Reader Protocol, Reader Protocol and Application Level Event provide the functionality for Event Capturing; the EPC Information Services (EPCIS) [9] is the specification for Repositories; one standard (Object Naming System – ONS [5]) and one developing standard (Discovery Services – DS) for discovering EPCIS repositories are available [6]; and although the EPCglobal architecture standards do not specify how to capture contexts, the EPCglobal event data model is a mixture of the three categories of information we discussed in Sect. 2.3.

2.5 EPCnetwork Repository and Discovery Mechanism

An EPCIS repository is a common component which stores RFID Events (EPC Event information) keyed by ID (EPC), and provides a query interface to EPCnetwork clients. Since EPCIS is distributed among several sites, a discovery mechanism is essential. In this section, we discuss the design of the two currently-available mechanisms, ONS and DS.

2.5.1 Object Naming Service (ONS)

The Object Naming Service service discovery mechanism uses part (or all) of an EPC and service type as keys to resolve a list of service locations. ONS is implemented on top of the Internet Domain Name System [10], and the list of service locations is returned by Domain Name.

Figure 2 shows typical information access flow in an EPCnetwork system using ONS. Initially, a service location (i.e., EPCIS repository) for a range of EPCs (typically assigned to a manufacturer) is preconfigured and stored as a set of DNS records. Then, at each site, a subsystem with data to store finds the service location matching the EPC just read by consulting the ONS[(1)(2) ††], and stores the event information to the server[(3)]. Similarly, to access information, a client at, e.g., retail company headquarters first asks ONS to resolve a repository for the ID[(4)(5)], then asks the repository

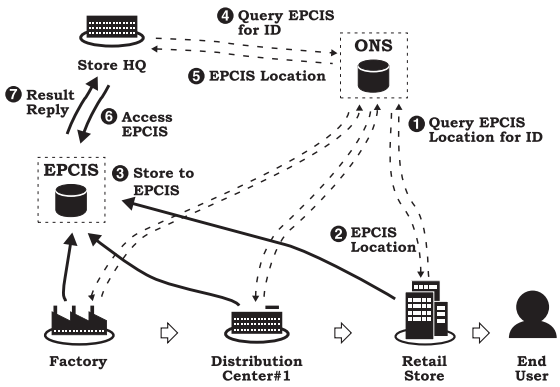

Fig. 2 Access flow of ONS based system.

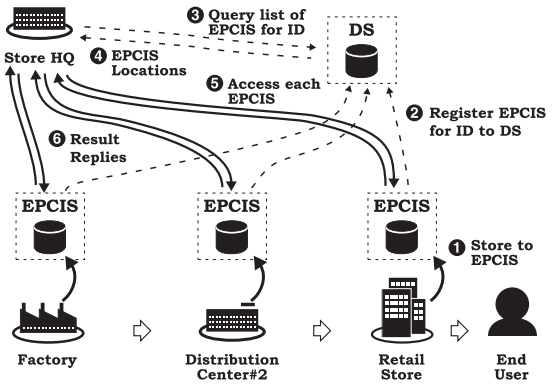

Fig. 3 Access flow of Discovery Services based system.

to retrieve[(6)(7)]. The mapping of EPC with type to service location is 1 : 1, and does not change.

2.5.2 Discovery Services (DS)

Discovery Services is also a discovery mechanism for EPC-keyed information, but has different characteristics, and works much like an Internet search engine. Figure 3 shows typical information access flow in an EPCnetwork system using DS. First, the subsystem that read the EPC stores information in an EPCIS[(1)], which records in the DS that it has information about the given EPC[(2)]. When a client asks the DS about an EPC, the DS will return a list of service locations which may have information[(3)(4)]. The client then will ask each service and combine all of the information as the query result[(5)(6)]. The EPC to service location mapping is $1 : N$, and the several service locations are usually physically distributed among multiple sites.

[†]For brevity, we do not reference each standard individually. All EPCglobal standards are available via [8]. As of this writing, the document on DS is not publicly available from EPCglobal, but work from the same group can be obtained from the EU Bridge Project [6].

[††]Superscript numbers in parentheses represents numbers in black circles in figures.

2.6 Problems with the EPCnetwork Architecture

The distributed nature of information sources in an RFID system forces difficult design decisions, and both ONS and DS exhibit some problems with mismatch between design decisions and actual usage patterns:

ONS: not for item-level use An EPC consists of three fields: manufacturer ID, product ID and serial number [7]. ONS uses the manufacturer ID and product ID to locate the manufacturer's service location – usually a repository. When system integrators use item-level information, since ONS only provides product level service location, servers have to deal with all serialized items for a product. In other words, ONS is not designed to resolve item level information, which needed to use in Track and Trace scenarios. There is concern on scalability for repository server.

DS: Placement policy and search key mismatch Discovery Services provide a way to discover repositories which may have information related to an ID. However, information generated at a site store the information to the location near by the information source, regardless of the observed ID. Client must request each of the candidate servers which DS resolved, to get full set of results. In other words, there is a mismatch between the path of the physical item and how each of repository stores the information according to an ID.

Also, for both architectures, the retention policy is undefined, resulting in storage requirements considerably larger than optimal. These problems cause inefficient use of computing resources and slower responses.

3. Proposal

We propose a novel architecture to overcome the problems discussed in Sect. 2.6 by making use of the collocation of the RFID tag and the interested party, as discussed in Sect. 1. To use this unique property effectively, we propose a relocatable information repository architecture, "Otedama," with the following properties:

Information as bundles A time series of RFID Information keyed by an ID is managed as a bundle.

Shared, coordinated distributed repository Shared, coordinated distributed repositories will exchange bundled information.

Bundles that track RFID tags A bundle is relocated or replicated between repositories. *Push* and *pull* on bundles occur when an ID is observed at a location, causing effective relocation of the information to the proximity of the tag itself.

Automatic Retention Since bundles can move based on the client's interest, suitable retention policies often occur naturally.

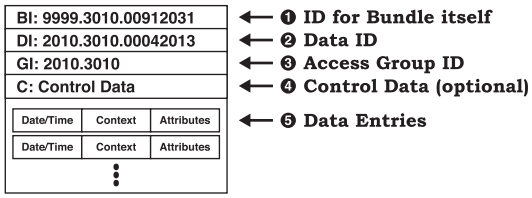

Fig. 4 Example of bundle.

Access control at bundle Access control is enforced at the bundle level using a DRM-like scheme.

Notifications To update external databases, a bundle may have notification control parameters set to send update notification to remote service entities.

Use DS as current location discovery For fallback search, DS is used for lookup current bundle location.

We made the following assumptions when designing this proposal:

- The relationships between sites are business relationships which are generally stable for weeks to years.
- The previous hop neighbor can be found based on a given ID (which includes the manufacturer ID) and knowledge of those business relationships.
- The size of each record is small: a few hundred bytes per records at most, usually less than 100 bytes (see Sect. 3.1).

3.1 Information Model and Bundle

We separate the information handled by Otedama into two types: RFID-related information and external business information. Otedama manages only the former, and relies on external services for the latter by providing notification mechanisms (see Sect. 3.6). All of the information stored in Otedama is keyed by an RFID tag's ID. Otedama treats a series of RFID information as a bundle, which is the unit of relocation. Figure 4 depicts an example of a bundle. Each bundle has an ID for the bundle itself(1), a data series ID which represents the RFID tag ID (EPC)(2), an access group ID used for access control (see Sect. 3.5)(3), control data used for notification and extensions (see Sect. 3.6)(4), and a series of data entries(5). In a bundle, each RFID Event and its Captured contexts are expressed as a single entry consisting of Date/Time, Context and Attributes.

3.2 Bundle Update and Relocation

When a subsystem needs to store an RFID event, it is added to the bundle for the matching ID. Bundles are automatically discovered (see Sect. 3.3) and, if necessary, relocated to the local repository prior to update. If the bundle is not instantiated locally, the system will search for it and pull the bundle to the local repository. Relocation will be initiated when a bundle matches a search request, the item physically arrives at a new location and the ID is observed (pull), or by

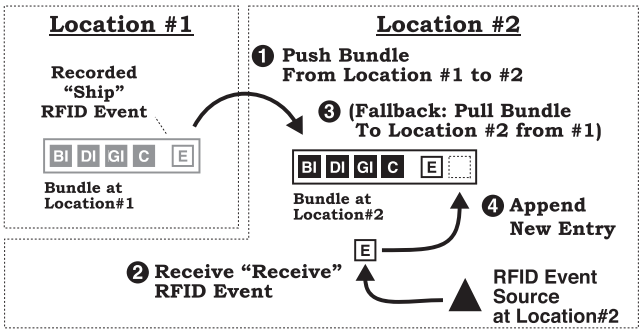

Fig. 5 Example of bundle pull.

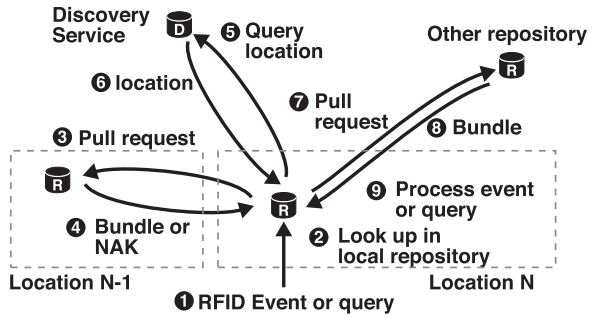

Fig. 6 Otedama search scheme. Steps 3-4 may be skipped if the bundle is already local. Steps 5-8 are executed only when the bundle is not found either locally or in the nearest neighbor in the supply chain.

using knowledge about the destination on shipment (push).

Figure 5 depicts the pull procedure. Location #1 has the bundle for an ID. Usually, Location #1 tries to push the bundle to Location #2(1). When the ID is observed at Location #2, a "Receive" event occurs(2). If the bundle is not found, then, as a fallback scheme, the bundle for the ID is pulled from Location #1(3), and finally, a new entry is added to the bundle(4). Since the shipping side of the system knows about the unique destination location anyway, the cost to determine the destination server will be minimal. Once the pull is complete, the destination system will find the information locally when needed. Since physical movement of an item takes time, push can happen asynchronously, putting the data in place before the actual arrival of the item, eliminating the wait for a synchronous pull. When a bundle moves, to make its location visible to other systems, the repository location is updated at DS. For combined use with Otedama, DS can simply keep the latest location of the bundle and can eliminate records for previous locations. Note that, if DS is used with Otedama, DS is required to return the latest repository location of the bundle to Otedama, and repositories which do not participate in the Otedama scheme must not register with DS. Repositories can voluntarily keep records, but registering these obsolete locations could create confusion to Otedama. Therefore, we prohibit this kind of update. Access control must be enforced accordingly. Due to the problem's nature, when the system tries to pull information, the bundle usually is located at the previous hop in a well-defined chain. Note that, since a bundle is always updated at most at a single location, a merge operation is not necessary. But once that happens, since all of the data is time-series data, it is possible to merge without problem.

3.3 Search

Figure 6 shows Otedama's three-way search mechanism. When a repository receives a request (either a query or a store request following an RFID event)(1), it first tries to locate the matching bundle(s) locally(2). If the bundle is not found, the repository next queries a neighbor(3)(4). In applications such as TnT, this process works well, because a single neighbor is typically defined based on business relationships in the supply chain, as noted in our assumptions above. As a last resort, DS is used to find the latest bundle location(5)(6), then the bundle is pulled from the discovered location(7)(8).

3.4 Automatic Retention

Otedama uses the proximity property to determine the preferred location for a bundle. After a bundle is relocated, it is safe to delete older versions; only the site where the ID was last seen is required to retain the record. However, for redundancy, historical tracking, or other business purposes, it might be desirable to retain older records for a time.

3.5 Access Control

Each bundle's control information and data entries can be encrypted with a symmetric key called the *bundle access key* (BAK), which is shared among entities allowed to access the bundle's contents. The access group ID identifies which key will decrypt the bundle contents. That key is managed using a PGP-like multiple entity key sharing scheme [11].

Each client has its own PKI public/private key pair [12]. When a group is created, a unique secret key is generated. That key is encrypted using the public key of *each* group member, and the set of X.509 subject [12]/encrypted key pairs is stored in the repository as a bundle. To access the data bundle, a client uses the access group ID to find the key bundle, extracts the copy of the BAK encrypted with its own public key, and uses its matching private key to decrypt the BAK. The key bundle is relocated or replicated along with data bundles as necessary.

3.6 Notification

To interact with other systems on the reception of events, Otedama has a notification mechanism. In a bundle, one or more callback service locations may be configured as control data, allowing the system to notify external services with selected data. Using this scheme, rich, purpose-oriented databases can be created at a site, while keeping bundle management simple.

4. Evaluation

In this section, we evaluate Otedama by comparing with two schemes: ONS and vanilla DS. We analyze two simplified procedures, storing a captured context bound to an ID, and retrieving a set of captured contexts based on an ID. We present the cost in terms of:

- the number of requests required for store and retrieve,
- the number of records stored in each site repository, and
- the number of records that the DS must store.

To simplify discussion, we use the parameters listed in Table 1. First, we discuss some of these parameters.

A client-server based service's cost, such as response time, is composed of network cost (communication delays) and computing cost (processing of a request). Although precise modeling is difficult, simplified modeling focusing on some of the factors is possible. For a distributed system, network delay is the major factor in the response time. For a system using databases, performance can be affected by the amount of of data (in bytes or number of records), which will affect the number of accesses to secondary storage, which is the major factor in the database systems' processing cost. Since it is difficult to build an analytic model for throughput independent of an actual system configuration, we primarily focus on response time, which we can discuss independent of the underlying individual servers.

The parameter pairs of A_R - A_L, S_R - S_L, K_R - K_L distinguish remote/local request costs for information access, store and delete. When comparing response time of remote server request and local server request, the major differentiating factor is network delay. Depending on the network distance, the delay may vary by several orders of magnitude. If both of the nodes are communicating using a local area network, the round-trip response time of a packet is less than 0.3 ms even with consumer class gigabit Ethernet switches. In contrast, if the communicating nodes are connected through a wide area network, the round-trip response time is 7 to 8 ms[†]. If the destination is overseas, the delay could be a minimum of 100 ms. We can approximate remote server response time is 20 to 300 times slower than local server. Reportedly, Walmart is importing 70% of its commodities from China [13]. If the factory's servers are in China, A_R, Q_R or S_R can be larger than for US domestic companies' servers, making the effect more dramatic. We also need to estimate the contribution of the request processing to our total latency. A typical RFID repository system's implementation relies on some relational database implementation such as PostgreSQL or MySQL. According to Intel, latest high-end server system's performance is nearly 0.7Million mixed transaction per minute [14]. Clearly, for a simple table indexed by a single ID, which is adequate for our needs, requests on even a much lower performance system will still require far less than one millisecond. In any distributed RFID system, processing time will be negligible compared to long distance network delay. Similar reasoning can be applied to store and delete also. We can conclude that:

$$A_R \gg A_L, \quad S_R \gg S_L, \quad K_R \gg K_L$$

The parameter pair Q_R - Q_N are costs to locate candidate repositories. Q_R is the cost to locate repositories from DS or ONS by specific ID. Q_N is the cost to locate neighbors by specific ID. Service cost for either ONS or DS (both response time and computing) are unknown and difficult to model, or even difficult to measure in a reasonable way. If we likewise assume processing time of less than one millisecond for both ONS and DS, we can ignore the difference between ONS and DS because communication delay is far larger. For this reason, we treat ONS and DS as having the same locate cost, Q_R. Note that, ONS is based on DNS, and DNS data is extensively cached. Since it is too complex to include such factor into this model, we just treated as a remote request. Similar discussion can be applied on DS registration cost, R_D. We treat R_D as same as Q_R. For Q_N, as stated in assumption above, the single target neighbor can be found by product ID. We can use a single, low-frequently update, relatively low-volume table for the neighbor search. The size of the table can be determined from the number of different items available at a large store (Please refer to Table 2), and the record size is the size of a URL of a neighbor server plus the size of an ID. If we allocate 100 bytes for each record, the amount of data of the whole table will be $142{,}000 \times 100 \cong 13.5$ MBytes. With modern system, all of that data can be held in memory and lookup time will be negligible. We can conclude with following expressions:

Table 1 Parameter notation.

Parameter	Description
x	Type of site: Factory: F, DC: D or Store: S
I	Number of items observed through the supply chain
I_x	Number of items observed at site x
L	Number of sites handling an item (supply chain depth)
P_x	Average stock duration at site x (days)
D_x	Average data retention duration at site x (days)
E	Number of events observed per item at each site
A_L, S_L, K_L	Cost to Access(read)/Store/Delete a record at local server
A_R, S_R, K_R	Cost to Access(read)/Store/Delete a record at remote server
Q_R, Q_N	Cost to lookup repository for remote/neighbor
R_D	Cost to register a repository to DS
U_i	Number of items factory i ships over lifetime

Table 2 Walmart supply chain parameters [13], [15].

Parameter	Value	Units
Cases handled in whole company	5.5 billion	cases / year
Number of DC	147	
Percentages of import from China	70%	of commodities by price
Number of products at large Walmart SuperStore in CA	142,000	Products

[†]Sampled using 100 Mbps VDSL in Tokyo metropolitan area.

Table 3 Per-item Performance Cost to Store a Series of E Contexts. The operation may consist of several phases, depending on the scheme. L, P, S and R denotes Locate, Pull, Store and Register. The three Otedama lines are for three separate fallback cases. Phase shows each step's cost, and total shows overall costs.

Scheme	Phase				Total
	L	P	S	R	(L, P cumulative)
ONS	A_R	0	$E \cdot S_R$	0	$A_R + E \cdot S_R$
DS	0	0	$E \cdot S_L$	R_D	$E \cdot S_L + R_D$
Otedama (Typical)	$\cong 0$	0	$E \cdot S_L$	0	$E \cdot S_L$
Otedama (Neighbor)	Q_N	A_R	$E \cdot S_L$	R_D	$Q_N + A_R + E \cdot S_L + R_D$
Otedama (DS)	Q_R	A_R	$E \cdot S_L$	R_D	$Q_N + Q_R + 2A_R + E \cdot S_L + R_D$

$$Q_R \gg Q_N, \quad Q_R \cong R_D$$

In addition to the discussion above, we use the figures in Table 2 [13], [15] to evaluate the system requirements and workload. We use six for the number of contexts captured at a site for each ID (receiving, storage, order-picking, accumulation, sorting and shipping) [16].

4.1 Number of Store Requests

Table 3 shows the expressions for required performance to store E contexts at a site for a single item. These expressions capture the importance of the first request, and the amortization of cost over the set. ONS requires one resolve request, then stores to a single server. DS, in contrast, requires the local store cost plus one-time registration cost for DS. Otedama works in three stages. First, Otedama consults the local repository if it is an Otedama site. The locate cost is negligible since Otedama can detect whether the bundle for the ID to be stored exists when storing the information onto the local repository. In other words, the cost is absorbed into S_L. This is the typical case, since Otedama pushes bundles to the next hop neighbor when it ships the product the ID is referring to. Due to this, Otedama typical does not require R_D, since registration happens on the initial push. If Otedama can't find the record locally, it find the neighbor from local table, then consults it. Otedama requires one pull cost and local store costs when the bundle is found at the immediate neighbor. If not, it must consult the DS, resulting in two more remote calls (Q_R and A_R). Since $S_R \gg S_L$, it is clear that ONS, which relies heavily on remote requests, has big disadvantages.

In the fallback case, Otedama consumes more resources than DS ($Q_N + Q_R + 2A_R$ or $Q_N + 1A_R$ compared to zero A_R). However, because business relationships are usually stable, and bundle usually pushed from previous neighbor, we can expect the typical result most of the time. While non-Otedama participating sites require DS-based search, at the Otedama sites, which participate in the supply chain, the typical case applies except in some failure situations. As long as entire Otedama chain is operating correctly, fallback will not happen. If we expect 99.99% uptime of the system, we expect 0.01% fallback rate.

Table 4 Performance cost to retrieve contexts by ID. The three Otedama lines represent different fallback cases. Requests shows each step's cost, and total shows overall costs.

Scheme	Requests		Total
	Locate	Retrieve	(Cumulative for Otedama)
ONS	Q_R	A_R	$Q_R + 1 \cdot A_R$
DS	Q_R	$L \cdot A_R$	$Q_R + L \cdot A_R$
Otedama (Typical)	A_L		A_L
Otedama (Neighbor)	Q_N	A_R	$Q_N + 1 \cdot A_R + A_L$
Otedama (DS)	Q_R	A_R	$Q_R + Q_N + 2 \cdot A_R + A_L$

Table 5 Number of records at each site by scheme.

Scheme	Number of Records	Note
ONS	$E \cdot U_i \cdot L$	For each factory i
DS	$E \cdot I_x \cdot D_x$	For each site, by type x
Otedama	$E \cdot I_x \cdot P_x$	For each site, by type x

4.2 Number of Retrieve Requests

Table 4 shows the performance impact of retrieving a set of contexts among supply chain sites for a requested ID. For ONS, the single remote repository can be found by a single request, then the records are retrieved from the server. For the vanilla DS system, after retrieving the list of servers from the remote DS server, the client must access at most L remote servers to fetch the complete set of records across the supply chain. Otedama initially consults the local repository. If the bundle is found, the locate phase is done and no pull is necessary. If the bundle is not found in this step, Otedama falls back first to neighbor query, then, DS. If there is no repository nearby, the client first queries DS to find the correct repository, then accesses it to get all of the records as a single bundle. If the client is at an Otedama site, the cost to find a record is either A_L when it is found locally, or an additional Q_N and A_R to fetch from the neighbor site, or additional $Q_R + A_R$ to use DS as last resort.

From the query cost point of view, because vanilla DS requires accessing up to L servers remotely to retrieve the latest information, the system-wide cost is far more than Otedama. As we discussed in Sect. 4.1, during normal, fault-free operation, all Otedama requests will correspond to the typical case, which is very efficient.

4.3 Estimated Number of Records at Repositories

Table 5 shows the expressions for the estimated number of records at each repository for the three schemes. For DS and ONS, there is usually no way to know when to delete the record. Thus, repositories must either 1) keep records for the foreseeable future, or 2) use a preconfigured retention duration (D_x). In contrast, Otedama can delete the records whenever the item itself moves to a different location. For this reason, for Otedama, the duration for a record at each site in the supply chain is the same as the item's stock duration, P_x. Since the preconfigured retention duration D_x for

both ONS and DS is larger than expected stock duration, we can estimate Otedama's advantage over both DS and ONS in required storage volume to be a factor of D_x/P_x. Because supply chain management operations try to minimize P_x at any given time, unless there is a good way to estimate, D_x will be several times larger ($D_x \gg P_x$). For example, if $D_x = 90$ (one quarter) and $P_x = 14$ (2 weeks), the ratio will be more than a factor of six, a considerable penalty for sites with large numbers of records. Systems with larger numbers of records will have higher search costs, though the details are beyond the scope of this paper. Of course, D_x may be managed more carefully, and P_x may increase (e.g., for more durable or slower-selling goods), reducing the advantage of Otedama. However, Otedama's natural flow reduces the number of records, improving site management.

4.4 Estimated Number of Records for DS

Vanilla DS requires at least $L \cdot I$ records to be stored in the DS, because each site retains and manages RFID information separately. On the other hand, Otedama only uses DS to record a pointer to the repository where the latest bundle exists. It requires at most I DS records, which is $1/L$ of the original DS design.

4.5 Estimated Impact on Storage

Now, we evaluate these expressions to see the impact. By using the numbers in Table 2, we can estimate the number of cases to go through a single DC, per year, as

$$I_D = \frac{\text{cases per year}}{\text{number of DCs}} = \frac{5.5 \times 10^9}{147} = 2.8 \times 10^8.$$

At the DC, at least six contexts will be recorded per item. If each context is 128 bytes, we require 768 bytes per item, or 26 GB per year or 6.5 GB per quarter, for the data only, without index. If we estimate average stock duration as 2 weeks, Otedama will require a little less than 1 GB at each DC, compared to 6.5 GB for a quarter-based vanilla DS retention policy. These numbers assume one RFID tag per *cases*. If tags are used at the item level, the volume of records can be ten times larger or more.

Thus, the effect of lowering amount of storage by Otedama is beneficial.

4.6 Retention Policy and ONS/DS

In this section we discuss possible retention policy extensions for ONS/DS compared with the Otedama scheme.

For Otedama, deletion of a bundle will be initiated by the loss of the interest in the information. For example, an actual sales event at a point of sale terminal may trigger final deletion of a bundle. Otedama can make use of this trigger to naturally determine when to delete records. As we discussed in Sect. 4.3, the design of ONS and DS does not introduce the concept of retention. This effectively forces these systems to use some fixed duration costs. But, by making use of the sales trigger, ONS/DS system also are able to reduce records – requesting deletion of records bound to an ID on the point of the sales. Table 6 shows performance cost to delete records, derived from the discussion of retrieval in Sect. 4.1. The component of delete request cost is very similar to retrieval costs, so the evaluation is: For DS, the client should send a request to all L servers on each delete sales trigger on the sales. For ONS, a single server should handle all of the requests. While it is possible to postpone requests and aggregate multiple requests per repository, the cost seems very high compared to Otedama.

Table 6 Performance cost to delete records by an ID.

Scheme	Requests		Total
	Locate	Request	
ONS	Q_R	K_R	$Q_R + 1 \cdot K_R$
DS	Q_R	$L \cdot K_R$	$Q_R + L \cdot K_R$
Otedama (Typical)	K_L		K_L

However, delete operations have different characteristics from store/retrieval operations. Their destructive nature will cause access control issues. An Otedama bundle as a whole always adds records log-style, and whole bundles are deleted at the site when the party holding a bundle loses interest in it. In contrast, if some repository needs to handle a delete request from some other party, the repository needs to enforce access control, since it is a destructive operation. In both ONS and DS, every repository has to handle such access control for every possible requestor, which may be difficult. Also, propagating delete requests to multiple repository will raise reliability concerns. Otedama, in general, simplifies the issues of distributed transactions by keeping the locus of control in a single location near the current instance of the bundle. Nevertheless, extending Otedama using a full distributed transaction mechanism may bring benefits. We leave this for future work.

4.7 Cost of Bundle Push

On the shipment of a product to a destination, a supply chain location knows about the destination location and its information system – the repository. The destination will be determined by some external system outside of Otedama, but once we know the destination site, we can determine the repository of the destination site with help of the external system. When this happens, Otedama will ask DS or other scheme to find the remote repository, using the destination location rather than the product ID as a lookup key. Otedama can push the bundle for the ID to the remote Otedama, prior to the physical arrival of the product. The cost of this query will be repository lookup and one time store:

$$Q_R + S_R$$

When look at the total operational cost, push cost should be considered. But from the response time point of view, since push operates asynchronously and guarantees the local discovery of the information on the product's arrival,

Otedama's push scheme is advantageous.

5. Related Work

Otedama is a kind of RFID middleware. The EPCglobal network specification, the current de-facto standard discussed in Sect. 2.4, evolved from Savant [17], which was proposed by the Auto-ID Center. Since the specification only defines interfaces and an abstract data model, software developers can select their implementation scheme. For example, Fosstrak [18][†] provides a prototyping platform using open source components; SAP implemented their RFID platform [19] integrated with their ERP platform; and Siemens developed an RFID information query model focusing on temporal-based data modeling in Dynamic Relationship ER-Model [20]. Note that, all of the research above on EPC-network focuses on the repository, and does not discuss the distributed data model or their own performance in detail.

The EU Bridge Project developed DS as a possible successor to ONS [6]. They categorized and evaluated eight data discovery mechanisms, and selected "Directory-of-Resources" and "Notification-of-Resources" as their models of choice. The former is the DS design described in Sect. 2.5; the latter provides a notification mechanism by publish-subscribe model coordinated by an intermediary DS.

From server mechanism point of view, and its focus on resource balancing management, our scheme resembles P2P based Contents Delivery Network (CDN) architecture [21], especially some of the work focusing on storage resource management, such as OceanStore [22]. In Sect. 3.6, we mentioned our notification mechanism. Event monitoring style architecture has similarity with regard to RFID event information processing [23], [24] or the notification mechanism. It will be possible to incorporate event monitoring scheme to improve notification mechanism. Also, this mechanisms could be categorized as very simple form of reactive agents [25].

6. Conclusion

We have presented a novel RFID middleware architecture designed to overcome issues with existing systems by making use of the proximity property of RFID information, keeping the information close to the RFID tag (and item), where processing is most likely to occur. Otedama builds on the developing standard Discovery Services by modifying the behavior of the repositories and clients to operate on bundles rather than individual events, and migrating those bundles to track the item location, improving the efficiency of both queries and storage, and providing a more natural framework for currently-undefined issues such as retention policy.

We have evaluated Otedama by comparing it to vanilla DS and to ONS by calculating the expected request workload for store and the volume of records stored in the repositories and Discovery Services. Store performance is somewhat worse for Otedama than DS, while retrieve performance is substantially better – more than a factor of six. Since the application workload will typically execute more retrieves than stores, we can say that Otedama is better than the current developing standard. Searching the repository will also be faster for large systems, because each Otedama repository stores only a fraction of the total records. Otedama places less load on a DS server, as well, because it stores only a single record per item, compared to the several records required in a vanilla DS system. Otedama likewise is expected to provide faster query response time than ONS.

This analysis suggests that Otedama can provide better performance with better resource utilization than DS or ONS. We plan to extend our study on this architecture by developing a working prototype in the near future, with the long-term goal of extending developing RFID standards toward Otedama.

[†]Fosstrak was formerly known as Accada.

References

[1] M.R. Rieback, B. Crispo, and A. Tanenbaum, "The evolution of RFID security," IEEE Pervasive Computing, vol.5, no.1, pp.62–69, 2006.
[2] T. Phillips, T. Karygiannis, and R. Huhn, "Security standards for the RFID market," IEEE Security and Privacy, vol.3, no.6, pp.85–89, 2005.
[3] J. Landt, "The history of RFID," IEEE Potentials, Jan. 2005.
[4] S.E. Sarma, D. Brock, and K. Ashton, "The networked physical world–proposals for engineering the next generation of computing, commerce & automatic identification," MIT Auto-ID Center White Paper, Oct. 2000.
[5] Object Naming Service (ONS) Standard. http://www.epcglobalinc.org/standards/ons (Verified 2010/2/23)
[6] "High level design discovery services," BRIDGE Project, Sept. 2007. http://www.bridge-project.eu/ (Verified on 2009/2/23)
[7] EPC Tag Data Standard (TDS) Standard. http://www.epcglobalinc.org/standards/tds (Verified: 2010/2/23)
[8] EPCglobal Standards. http://www.epcglobalinc.org/standards/ (Verified: 2010/2/23)
[9] EPC Information Services (EPCIS) Standard. http://www.epcglobalinc.org/standards/epcis (Verified: 2010/2/23)
[10] P. Mockapetris, "RFC 1034: Domain names - concepts and facilities," Internet RFCs, Nov. 1987.
[11] P.R. Zimmermann, The Official PGP User's Guide, MIT Press, Cambridge, Mass., 1995.
[12] D. Cooper, S. Santesson, S. Farrell, S. Boeyen, R. Housley, and T. Polk, "RFC 5280: Internet X.509 public key infrastructure certificate and certificate revocation list (CRL) profile," Internet RFCs, May 2008.
[13] "Wal-Mart's China inventory to hit US$18b this year (News paper article)," China Daily, Nov. 2004. http://www.chinadaily.com.cn/english/doc/2004-11/29/content_395728.htm (Verified: 2010/2/23)
[14] Intel, "Server perfomance - Intel processors." http://www.intel.com/performance/server/index.htm (Verified: 2010/6/20)
[15] "Walmart facts." http://walmartstores.com/PressRoom/9689.aspx (Verified: 2010/6/20)
[16] J.P.V. den BERG, "A literature survey on planning and control of warehousing systems," IIE Trans., vol.31, no.8, pp.751–762, Jan. 1999.
[17] "The savant version 0.1," Auto-ID Lab White Paper, pp.1–46, Jan. 2002. MIT-AUTOID-TM-003.

[18] C. Floerkemeier, C. Roduner, and M. Lampe, "RFID application development with the Accada middleware platform," IEEE Syst. J., vol.1, no.2, pp.82–94, Jan. 2007.
[19] C. Bornhövd, T. Lin, S. Haller, and J. Schaper, "Integrating automatic data acquisition with business processes experiences with SAP's Auto-ID infrastructure," Proc. 30th VLDB Conference, Toronto, Canada, Jan. 2004.
[20] F. Wang and P. Liu, "Temporal management of RFID data," Proc. 31st VLDB Conference, Trondheim, Norway, Jan. 2005.
[21] S. Androutsellis-Theotokis and D. Spinellis, "A survey of peer-to-peer content distribution technologies," ACM Comput. Surv., vol.36, no.4, pp.335–371, Jan. 2004.
[22] J. Kubiatowicz, D. Bindel, and Y. Chen, "Oceanstore: An architecture for global-scale persistent storage," ACM SIGARCH Computer Architecture News, vol.28, no.5, pp.190–201, Jan. 2000.
[23] M. Mansouri-Samani and M. Sloman, "GEM: A generalized event monitoring language for distributed systems," Distributed Systems Engineering, Jan. 1997.
[24] J. Bacon, K. Moody, J. Bates, R. Hayton, C. Ma, and A. McNeil, "Generic support for distributed applications," Computer, vol.33, no.3, pp.68–76, Jan. 2000.
[25] H. Nwana, "Software agents: An overview," Knowledge Engineering Review, vol.11, no.3, pp.205–244, Oct. 1996.

Shigeya Suzuki is associate director of research and development at Auto-ID Labs Japan (Keio Research Institute at SFC, 2004). Visiting researcher of USC Information Sciences Institute (2005-2007). Assistant Professor of Keio University, Faculty of Media and Governance (2007). His current recent research focuses are ubiquitous computing and scalability issue around distributed computing. He is currently Ph.D. candidate of Keio University, Graduate School of Media and Governance.

Rodney Van Meter received a B.S. from the California Institute of Technology in 1986, an M.S. from the University of Southern California in 1991, and a Ph.D. from Keio University in 2006. His research interests include storage systems, networking, post-Moore's Law computer architecture, and quantum computing. He is an Assistant Professor of Environment and Information Studies at Keio University's Shonan Fujisawa Campus.

Osamu Nakamura received a B.S. from Keio University in 1982, an M.S. in 1984, and a Ph.D. in engineering in 1993. He became assistant professor in the Faculty of Environment and Information Studies at Keio University's Shonan Fujisawa Campus in 1993, associate professor in 2000, and professor in 2006. He has been Associate Director of the Auto-ID Labs Japan since 2003.

Jun Murai is dean/Professor, Faculty of Environment and Information Studies, Keio University. Founder of WIDE Project. MS for Computer Science from Keio University in 1981, received Ph.D. in Computer Science, Keio University, 1987. Former Vice-President of Keio University from May 2005 to May 2009. He received Funai Achievement Award 2007 from Forum on Information Technology on 2007, Jonathan B.Postel Service Award from the Internet Society on 2005, Personal Award from Minister of Internal Affairs and Communications on 2004, Personal Award from Minister of Economy, Trade and Industry on 2002, Personal Award from Minister for Public Management, Home Affairs, Posts and Telecommunications on 2000, The Best Paper of '90s Award, ISPJ, 2000.