On the Identification Device Management and Data Capture via WinRFID Edge-Server

Xiaoyong Su, Chi-Cheng Chu, B.S. Prabhu, and Rajit Gadh, *Member, IEEE*

***Abstract*— In this paper, we present a layered architecture and approach for creating a high performance edge server called Reader Coordinator (RC) for managing variety of automatic identification hardware such as RFID (Radio Frequency IDentification) reader, RFID printer, Sensor, Barcode scanner, or a controllable device (e.g. Programmable Logic Controller). A unique hardware abstraction approach is used to provide device extensibility and manageability. Rule based device configuration and control which allows the real-time changing of system behavior is presented. Various data encoding schemes used to accommodate data captured by a device and used as the input of rule based data filtration and aggregation are discussed. We discuss a mechanism of building self-governed automation processing in the data capture process. Performance is measured by using virtual reader that generates high volume of data at high speed. The measurement shows the RC is capable of handling such data.**

Keywork: RFID, Network, AIDC, Data Capture, System, Extensibility, Scalability

I. INTRODUCTION

Automatic identification technologies such as Radio Frequency IDentification (RFID) [1] [2] and Barcode are utilized by enterprise applications such as Enterprise Resource Planning (ERP), Supply Chain Management (SCM) system, Warehouse Management System (WMS), or, Manufacturing Execution System (MES) to improve productivity and reduce operation cost., Identification devices including RFID Readers, RFID Printers, Barcode Scanners, and other Controllable Devices such as Programmable Logic Controller (PLC) are installed at each transaction location such as assembly line, packaging station, loading point (e.g. port), or, dock door. These devices are used to prepare identification data (e.g. write data into RFID tag, print barcode label) and capture identification data (e.g. read data from an RFID tag or scan barcode number on a barcode label).

Manuscript received May 21, 2007. This work was supported by Wireless Internet for the Mobile Enterprise Consortium (WINMEC).

X. Su was with the school of engineer, University of California, Los Angeles. He is now with Impinj Inc. Seattle, WA, 98103 USA (phone: 425-773-9295; e-mail: xiaoyong.su@impinj.com)

C. C. Chu is with the School of Engineer, University of California, Los Angeles, CA 90095 USA (e-mail:cchu@wireless.ucla.edu).

B. S. Prabhu is with the School of Engineering, University of California, Los Angeles, CA 90095 USA (e-mail:bsp@wireless.ucla.edu).

R. Gadh is with the School of Engineering, University of California, Los Angeles, CA 90095 USA (e-mail:gadh@wireless.ucla.edu).

Because of the diversity of devices within an enterprise and within each enterprise application, managing and coordinating these devices represents a challenge [3] . Also, because of this diversity of devices, there are challenges in developing enterprise applications.

First, each enterprise application has to understand the device protocol to connect to the device and encode/decode data to/from the device. If the application needs to interact with multiple devices, a multitude of protocols have to be understood in the application. Since one of the requirements of an enterprise application is to assist the business operation of the enterprise, the performance would be compromised if there a large numbers of low level protocols have to be processed.

Second, there exists a challenge due to the cost of hardware and physical constraints. If there are N applications, and if on average each application needs M devices for data capture, then a total of M*N devices are needed. If K (a subset of N) applications need to capture data at a dock door, each application needs one device, a total of K devices have to be installed at the dock door. This could be difficult because of limited space and in addition, RFID readers could interfere with each other.

Third, to reduce hardware infrastructure cost, an identification device is usually shared by multiple applications. Since applications are individually operated, managing access conflict becomes critical especially in the case where each application operates on a device over an extended period.

Because of these challenges, it is necessary to create a software component that sits between the hardware and the applications and manages and coordinates the devices to reduce the complexity of hardware infrastructure. With this software layer, the enterprise application does not need to interact with the device, but with the software component. It sends data to the software component and receives data captured by this software component without consideration to the kind of devices it is interacting with. This software component is defined as the edge server. Further the edge servers are inter-connected to form an identification network, which in turn is able to set-up devices, control the devices and capture data from RFID tags or barcode attached on business objects such as raw material, products, equipments, shipments and personnel.

There are several crucial issues to be addressed when deploying a RFID enabled edge server. These issues are discussed in the following three sub-sections.

System performance and scalability

RFID introduces a fundamentally new identification data capture approach as compared to existing barcode technology. It changes the methodology of capturing and processing identification data. First, RFID captures data at a substantially higher rate than bar code – typically 10-100 times [4] . Second, the larger regions of visibility of an RFID reader result in simultaneous reads of large amount of data in a given time period. For example, a case containing 100 tagged items may be read in less than one second using a UHF RFID reader that has the ability to read up to 1000 reads/second [4]. The edge server must have the capability of handling such high volume of data at high speeds.

Device manageability and system extensibility

As new RFID reader models are added into the network, new communication interfaces, new instruction sets for controlling devices, and new features are introduced. In this situation, a new device driver needs to be developed and integrated into the edge server and the edge server needs to be recompiled. This approach makes maintenance of a new reader model in the edge server very tedious. Sometimes, the entire system (both software and hardware) has to be shut down for upgrading. The adoption of new devices into the edge server without significantly modifying the edge server is an important research topic.

Interference of surrounding environment and material

The RFID reading range and reading reliability are highly dependent on the environment and the object to be tagged. Experiments performed [5] [6] [7] and prior research [8] [9] have shown that the reading reliability of a tag is significantly affected in the presence of metal, liquid, and interference with other RF resource [10]. Multiple readers or antennae are required to enhance readability. Parameters such as RF power and sensitivity need to be adjusted to optimize reading results. Device configuration and control on the fly is needed. Sometimes, data capture devices (such as RFID reader) and controllable device (such as PLC used to control conveyor) need to be coordinated to obtain better reading.

To address these issues in an RFID network, we focus on the development of a unique architecture for the edge server, the Reader Coordinator (RC) which is an essential component of WinRFID [11] [12], and, the overall RFID platform. Salient features of RC include high system performance, flexible device manageability and extensibility, the capability of adapting to the environment and the tagged object, and the ability to coordinate with other automation devices. Performance of the RC is evaluated by using virtual readers which generate high data rates in a simulated RFID network.

II. Related Work

In most of the RFID systems, similar functionalities of the RC are addressed by a software layer called middleware [13] [14]. Examples of middlewares and edge servers are discussed in the following paragraph.

LogicAlloy's ALE middleware server [15] is an open source application based on EPCglobal standards. ALE Server allows the clients "subscribe" to RFID data that ALE Server collects. The communication is done using standardized Web-services via "SOAP" messages. Omnitrol's Edge Service Appliance [16] is edge networking appliance. It is Linux-based software architecture which uses XML and SOAP for data and event representation to support interoperability. It supports data collection from a variety of integrated event sensors such as temperature, humidity, pressure, vibration, etc as well as GPS for location. Accada EPC Network Prototyping Platform [17] is an open source RFID prototyping platform that implements the EPC Network specifications. The platform includes the Filtering and Collection Project which implement the Filtering and Collection role in the EPC Network and to develop the appropriate tools that facilitate communication with the Filtering and Collection instance.

Our proposed RC provides a rapid, platform independent publication development environment. It supports a variety of readers/tags from different hardware vendors (Frequencies from LF, HF to UHF, Protocols from EPC, ISO to ICode, Tiris and other vendor-specified protocols). It uses Web-Service based interface which supports third-party hardware (readers, sensors) plug-ins to provide intelligent data capturing, smoothing, filtering, routing and aggregation.

III. Edge Server - Reader Coordinator (RC)

As the edge server of the RFID network, the RC performs the following tasks:

(i) Managing devices - The management functions include adding, deleting or grouping devices. When a new RFID reader is installed, a reader object is created in the RC which maps to the reader. If the reader is removed, then the reader object in RC is deleted automatically.
(ii) Configuring devices - Runtime parameters such as RF power level, reading sensitivity, communication interface settings and tag protocols, determine the device operation. They should be set when the device is initialized or can be changed dynamically during usage.
(iii) Controlling devices - An RFID reader is controlled via a set of pre-defined commands. For example, EPC Gen2 compliant reader has mandatory commands such as "Read", "Write", "Kill", and "Lock" etc [18]. Applications should be able to access the device, send command /receive response to/from device.
(iv) Capturing identification data – Once the devices are set up and controlled by the RC, each device should automatically capture data from RFID tags or barcodes in the reading zone based on the pre-defined rules (defines how the device should behave). The data should be filtered

(to reduce the duplicate reading), aggregated (grouped) by the RC and stored in a local cache which can be used by other applications.

A layered architecture, as shown in Figure 1, is utilized so that the above mentioned tasks can be fulfilled. Three layers that cover device abstraction, device logic definitions, data definitions, data filtering and aggregation, and system information presentation are defined as follows:

(i) Hardware Layer - The lowest layer is the hardware abstraction layer. It contains the communication interfaces (TCP/IP [19] , USB [20] and RS232/RS485 [21]) and hardware protocols (RFID reader protocols, Barcode scanner protocols, Printer languages, General device command/event scheme). This layer is primarily for device manageability and extensibility and it does so by defining a device description scheme.

(ii) Processing Layer - The layer in the middle is the information processing layer that generates device control commands and filtering, parsing the data gathered from RFID tags, Barcode labels, and other data sources based on pre-defined rules. This layer performs device scalability, system performance, and capability to have cooperation between the devices.

(iii) Representation Layer - The upper-most layer is the information representation layer that includes system parameters, device parameters, device configuration, data cache, and applications. In this layer, the information is persistently stored in an exchangeable manner by using Extensible Markup Language (XML) [22] based format. They can be loaded into the system later on or exchanged with other applications.

In the following sections, we discuss the design of each layer and explain how this architecture facilitates the creation of RFID-enabled applications.

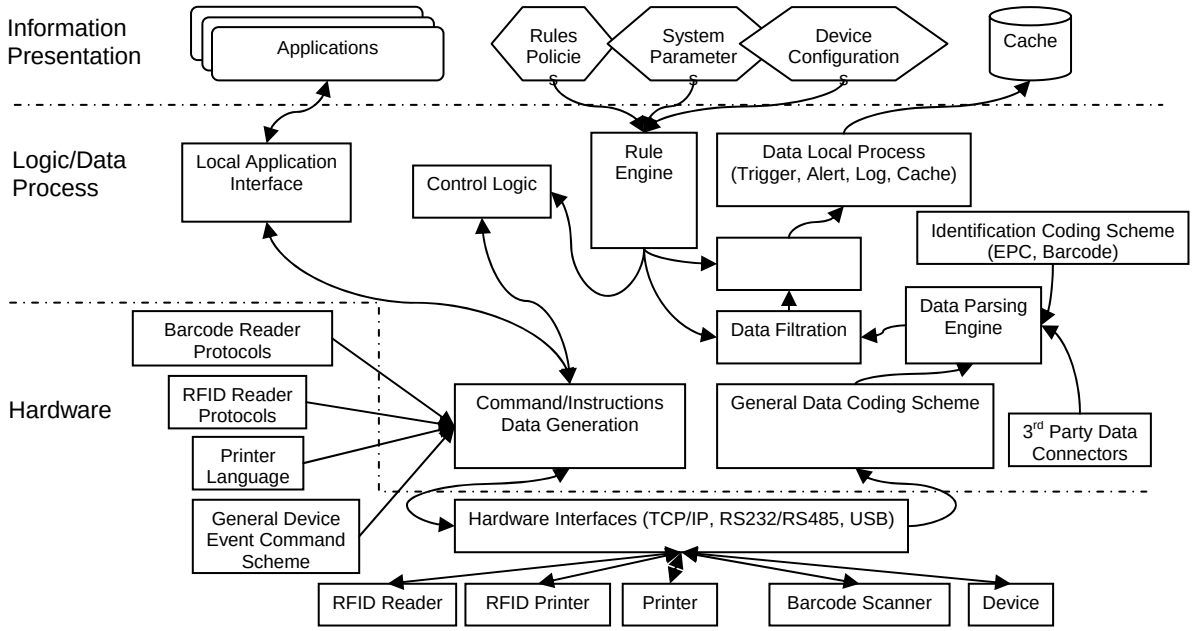

Figure 1 Architecture and Functional Stacks of Reader Coordinator

IV. HARDWARE LAYER

Device manageability and system extensibility is achieved by abstracting hardware and device configuration parameters. Our approach is fairly unique in that all identification devices are placed into one of two abstractions: 1) Communication interface. Identification devices are normally not standalone. They have to be connected to a network or a host computer for functioning through cables or wireless connections. The communication interface is a connection between the identification device and the computer. 2) Device protocols. The device protocol defines how to generate commands and encode/decode data captured by the devices.

The abstraction is typically implemented through a device driver, which contains communication interfaces and device protocols. In our approach, the communication interfaces and device protocols are separated. The advantage of separating the two is that many identification hardware vendors define a common protocol for a series of devices while the interfaces on the devices are different. By this architecture, a single driver is sufficient for all devices from a vendor.

A. Communication Interfaces

The identification device usually has one or more interfaces which connect the device to the network or host-computer. Commands/instruction sets/data are sent to or received from the communication interface.

Although there are various types of connections available, the most popular interfaces are TCP/IP (including Ethernet and Wireless Connections), RS232/RS485, and USB. We provide a

unique method that allows accomplishing transparency of the communication interfaces via a base interface CommInterface is defined in WinRFID. Now, the other interfaces can be automatically derived from it. This interface relationship is shown in Figure 2.

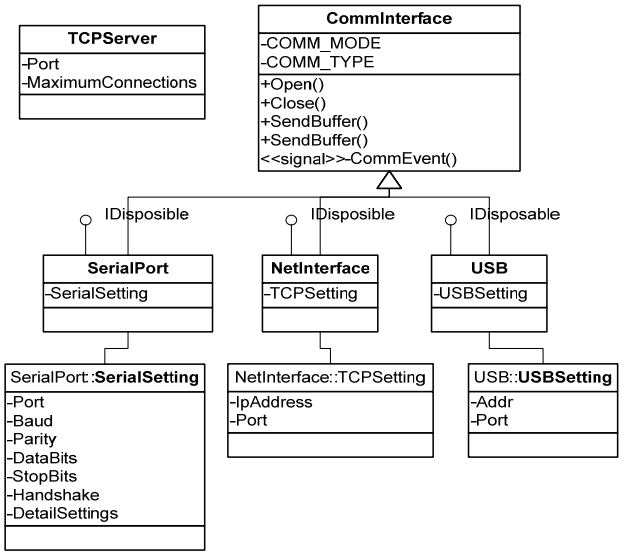

Figure 2 Interfaces abstraction in WinRFID framework

In the CommInterface class, general operations such as Open, Close and SendBuffer, and attributes such as Communicate Mode (Synchronized or Asynchronized), Communication Type (Interface type), and On_Receive Event are defined to describe the common function and attributes of the communication interfaces.

B. Control Device Behavior

1) Device protocol

Usually, application talks to device based on protocol (a set of pre-defined commands that controls device). For example, there are two types of protocols that RFID reader used to communicate with RFID tag. The two protocols are: (i) Protocols that deal with reader and tag communication and information exchange, and, (ii) protocols that deal with organizing host-to-reader command and parsing of data received from the reader. We refer to the first category as RFID air-interface protocols and the second as RFID reader protocols. The RFID air-interface protocols (EPC [18][23], ISO14443 [24] /ISO15693 [25] /ISO18000 [26], etc.) are normally standardized and the RFID reader protocols are proprietary.

The standardized protocols are defined as Interfaces (It is a programming term, only operations are defined in an Interface) in WinRFID. It is mandatory to derive the base device class and optional to derive a protocol interface since the reader may not support all the protocols. All the devices have the same operation procedures so that the integration of a reader in an enterprise application becomes straightforward. This unique reader implementation mechanism in our approach provides simple manageability and configurability of the WinRFID network.

RFID readers, RFID printers, Barcode scanners, Sensors, and Controllable devices that are initialized and controlled by Reader Coordinator are defined as In-Bound devices. To conduct field operations in an enterprise, mobile readers such as forklift-mounted readers are often used. Since these mobile readers are not in one location, it is difficult for I.T. staff to manage them. We define these devices as Out-Bound devices. In-Bound device are directly controlled by RC. Data captured by the mobile devices are pushed to the RC through Web Services [27] [28] or socket connection.

2) Controlling the device behavior

The behavior of a device is controlled by a set of parameters. The parameters are abstracted both on the generic settings of device and the unique features of the individual device. Generic settings include identity, communication interface settings, description, and running schedule etc. Unique features of an individual device are set as system runtime parameters. The parameters can be represented in an unique open-architecture XML interface that enables device control and information exchange in devise enterprise application environment.

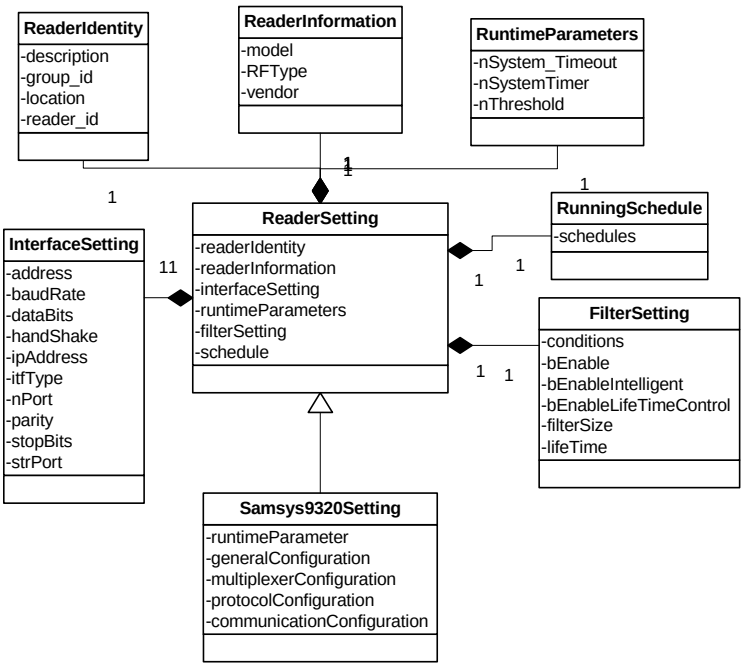

Figure 3 Reader parameters class diagram example

For example, Figure 3 shows the classes used for representing parameters of Samsys 9320 UHF reader. An instance of Samsys 9320 reader parameters is represented into XML code in Appendix A. As may be observed, the reader parameters have five components: Reader Identity, Interface Type, Reader Information, Runtime Parameters, Filter Settings and Running

Schedule. The Reader Identity contains the information of reader id, group id, description and location. The Reader Information contains the information of vendor, model, and radio frequency type. The Interface Type contains the information of interface type and parameters. The Runtime Parameters are different for each reader. The Filter Setting contains the filtration conditions. The filtration conditions could be nested. The Running Schedule is defined for automatically controlling the reader.

V. Logic and Information Processing Layer

The logic/information processing layer is unique in that it simultaneously performs multiple functions including the following - generates device control instructions, receives the raw identification data from the devices, parses the raw data into structured data by following data coding schemes defined in WinRFID, triggers events based on the system conditions and users-defined rules, and caches the structured identification information to proper destinations. In this layer, the logic/information processing relies on coding schemes and a flexible and dynamically modifiable set of rules that are user defined.

The coding schemes defined in this layer include General Data Coding Scheme (GDCS), Identification Coding Scheme (ICS), and General Identification Device Naming Scheme (GDIS). GDCS defines several data structures which are used to carry information collected from devices. ICS is used to decode and encode id numbers. It includes EPC and barcoding scheme, GDIS is used to define the naming mechanism for the identification device.

The user defined rules include event rule and logic control. The event rule is used to generate user defined event. Data captured by device is compared with the user defined rules, based on which events are generated and sent to appropriate applications. Logic control is used to coordinate the operation of identification devices.

A. General Data Coding Schemes

The data captured from an object must contain "location of capture", "time of capture" and "identification number" to identify a business event. For this reason, five types of basic data formats are defined in WinRFID. They are ID, TAG, GENERAL_DATA, EVENT, and EXCEPTIONS. ID, TAG and GENERAL_DATA are used to carry identification data such as EPC ID, sensor data, and barcode number etc. EVENT is defined to accommodate various events triggered by the system and device running status. EXCEPTIONS is used to represent the exception generated by data collection devices such as RFID readers and Barcode scanners. The EXCEPTION is generated when a predefined condition or threshold is reached.

B. Identification Coding Scheme

General data coding scheme defined in Section 4.1 is used to represent different type of data. Each data type contains an identification number. This number is used to identify and track a unique business object. Various coding schemes which define structures of the unique number are utilized to create a unique hierarchic number system. It manages the number assignment and helps trading partners to understand the number. For example, the number "90258580B6464EA4" represents a 64-bit SGTIN number defined by EPCglobal [36], an organization that defines RFID standards and tag data standards. The SGTIN number consists of six fields:

(i) bit 0 to bit 1 is Header which indicates a 64-bit SGTIN number.
(ii) bit 2 to bit 4 is Filter Value which is used for identifying the tag types.
(iii) bit 5 to bit 18 is Company Prefix Index which is used to look up a Company Prefix registered to EPCglobal.
(iv) bit 19 to bit 38 is Item Reference which is used identify the type of an item.
(v) bit 39 to bit 63 is Serial Number which is an unique number of the item in same type.

Based on the SGTIN coding scheme, the number "90258580B6464EA4" can be decoded into: Header = "2", Filter Value = "2", Company Prefix Index = "300", Item Reference = "180315", Serial Number = "4607652".

Since EPC coding schemes are standard coding schemes for RFID and barcode symbologies are standard coding schemes for barcode system, both are implemented in WinRFID to support different coding schemes used in industry.

C. Identification Data Coding Scheme

EPCglobal specifies a series of standards for RFID in the supply chain. In addition to the architecture of the EPCglobal Framework and the EPC reader Air-Interfaces, the EPC tag coding scheme (EPC Data Structure) [36] is also an important part of the global standards. The coding scheme is used to prepare and to parse the identification number that it typically corresponds to the RFID tag ID.

The EPCglobal tag data standard defines various coding schemes that in turn support diverse application in supply chain. They are variants of coding schemes defined by EAN.UCC. The coding schemes include Serialized Global Trade Identification Number (SGTIN), Serial Shipping Container Code (SSCC), Global Returnable Asset Identifier (GRAI), Global Individual Asset Identifier (GIAI), Serialized Global Location Number (SGLN), and Global Individual Asset Identifier (GIAI). SGTIN is used to identify individual product. SSCC is used to identify cartons, pallets and containers. SGLN is used to identify physical locations, function entities and legal entities. GRAI is used to identify a physical entity (such as pallet, barrel, rail car, trailer, etc.) as in returnable asset. GIAI is used to identify and track individual assets.

Besides the EAN.UCC compatible coding schemes, EPCglobal Tag Data Standard also defines two additional types, which are General Identifier (GID-96) and DOD Tag Data Structure. GID is independent of any existing identity specification. It is specified by EPCglobal on the purpose of supporting wide

range of identification requirements. DOD Tag Data Structure is defined to accommodate the information that required by Department of Defense RFID applications.

D. General device identification scheme

Besides the identification number carried by barcode or RFID tag, identification data also contains the location. A unique identification number indicating the logical or physical location of a device is generated and assigned to each device in the WinRFID network. The device id is required so that the device in the network is searchable and manageable. Even for Out-bound devices, a unique id is required by requesting from the RC.

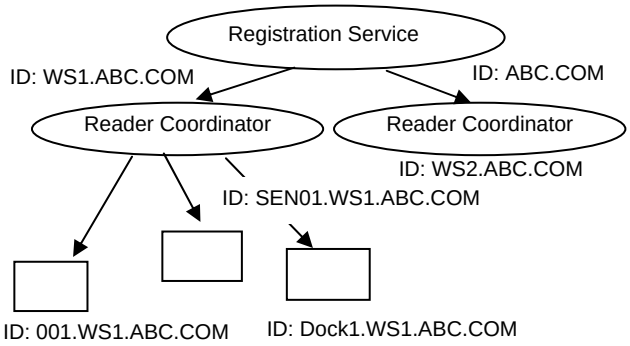

Figure 4 General device identification schemes

In the WinRFID network, the device identification number is in the form of a novel hierarchic structure and uses the same scheme that the Domain Name Service (DNS) uses. As shown in Figure 4, a registration service resides at a layer above all components (both software and hardware). The identity of the registration service is assigned as the enterprise domain name. Once a RC is registered with the registration service, the RC is assigned a name with the department information added to the head of the identity of the registration service. The RC assigns an I.D. using the same method when a device is added into it. Thus, in the WinRFID network, this novel method provides a unique identity for each device and a hierarchic naming structure is maintained.

E. Data Filtering

RFID data captured by an RFID reader is often duplicated because an RFID tag can be read multiple times in the reading zone of a reader. To remove the duplicate (and redundant) data, it has to be filtered. Meanwhile, irrelevant data should also be filtered before being sent to the application. For example, RFID readers may capture all the tags in the same frequency band. If a pallet contains 10 cases uses UHF EPC tags, the reader would read the tags both on the case as well as the pallet. However, a warehouse management software requires only the pallet data to be read, and, without more intelligent filtration (such as that based on a value in the tag data) it would be difficult to distinguish the pallet tag and case tag. Data filtration based on the content (e.g value of each field of the identification number) of the RFID data is required.

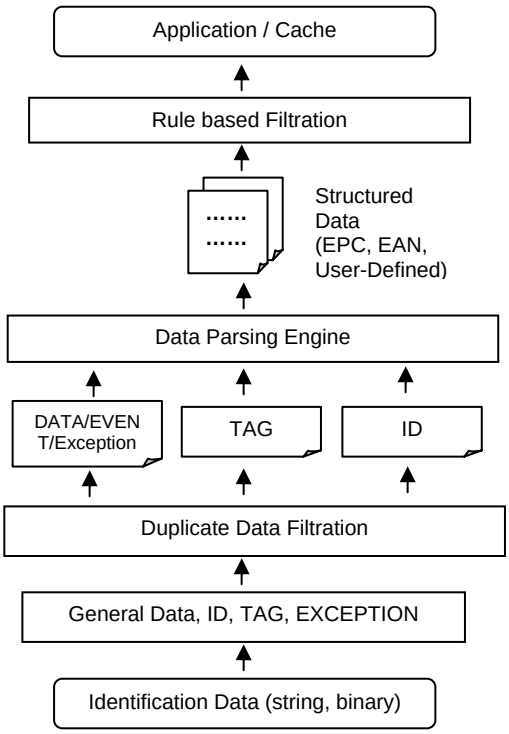

Figure 5 Data Processing Flow

A fundamentally new bi-level filtration algorithm is utilized in the current research. Figure 5 shows the two levels of data filtration in RC. In general, the data captured by a reader would be string or byte array. This data format is usually defined by the device vendor. This data is formatted into the general coding scheme. The first level filtration eliminates duplicatee data. The second level filtration is rule based filtration in which a user defines filtration conditions which are subsequently applied to the data read by the data parsing engine. Data that satisfy the filtration condition is sent to the application or local cache, otherwise it is discarded.

F. Event Generation

One of the unique aspects of the RFID network is that unlike typical I.P. networks, this network traffic is largely composed of events triggering actions. Event generation in WinRFID is triggered by user-defined rules. Accumulated identification data and system run-time status could trigger events (such as system status, alerts, notification, etc.). A trigger is a set of conditions/thresholds (e.g. maximum allowed reader temperature). The condition is a composite of single condition or multiple sub-conditions. The system status or accumulated identification data is compared with the condition. The event is raised if the condition is satisfied or the threshold is reached. The events are used for event-driven processes, which is what most business operations typically are. Based on the roles (system, device, policy, data, user and client application) existing in WinRFID, we define six distinct and unique types of events: EVENT_SYSTEM, EVENT_DEVICE,

EVENT_POLICY, EVENT_DATA, EVENT_USER, and EVENT_APPLICATION.

- EVENT_SYSTEM is defined to accommodate event triggered by WinRFID components such as Reader Coordinator and Data Collector etc. It indicates the current system status.
- EVENT_DEVICE is defined to accommodate status of devices such as RFID reader, Barcode scanner, RFID printer, Sensor and other in-bound devices.
- EVENT_POLICY is defined to accommodate event which indicates the violation of the pre-set system runtime parameters.
- EVENT_DATA is defined to accommodate event triggered by identification data capture.
- EVENT_USER is defined to accommodate event triggered by the rules defined by user.
- EVENT_APPLICATION is defined to accommodate event triggered by the subscription conditions of the subscriber (client application).

G. Control Logic

Control logic definition is a part of logic and information processing layer in RC. In the RC that contains control logic, the controlled devices form an autonomous system. As mentioned earlier, systems based on RFID can be highly or fully automated. The data captured may be used to change how, when, and where the readers read the tags. There are two types of control logic: (i) Devices-pairing control which involves event-command interactions of two devices. The event generated by one device is used as condition to trigger the action of another device. In the RC, a set of rules that have the device-pairing relations in the form of hash-tables is utilized. (ii) Pre-defined device control logic. A set of pre-defined system behaviors (such as start and stop, sensitivity adjust, and RF power etc.) are assigned to the device when the device is initialized and added into the reader coordinator. The behaviors could also be changed at run-time. The pre-defined system behaviors are part of the device configuration while the devices-pairing relations are part of the Reader Coordinator configuration. Once the devices-pairing is defined, it resides in the Reader Coordinator persistently as long as both devices are operational. The event raise and triggers are controlled by the rules.

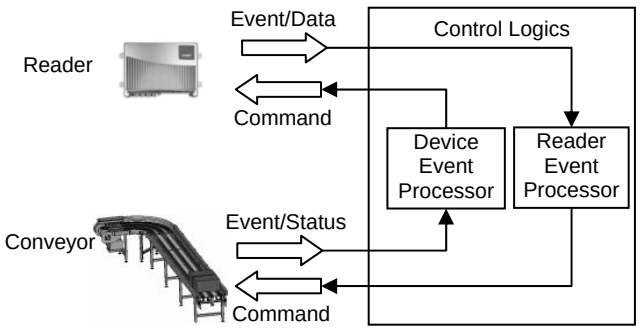

Figure 6 Devices-pairing control logics example

Figure 6 shows the devices-pairing control rules. The output (Event/Data) from the reader is used as input to set the conditions that trigger the action of the conveyor. The output (Event/Status) of the conveyor could also be used to set a condition to trigger the action of the reader.

VI. Information Representation Layer

The information representation layer is the layer that presents the identification data and devices to the end application. Rules, system runtime parameters and device configurations are represented in the XML format. The information persistently resides on the machine while the RC is running. All the system parameters can be restored if there is a system failure of the computer which hosts RC, and thus, the WinRFID network can be recovered. A high performance message queue is used to store the data. Next we will discuss data cache and system parameters.

A. Data Cache

In certain applications the number of readers used within a network is large and data could come in very short intervals. For example, it has been reported that seven terabyte data would be generated by the Wal-Mart RFID deployment every day [42]. To keep persistency and high performance, a local data cache is used to store the high volume of identification data. Rational database is being used in various data-central applications. However, it is incapable of storing the RFID data captured at high speed (e. g. 20 readers at speed of 100 tags/second generate 2000 tags/second data). This is because the performance of INSERT (insert a data to database) rational database is far below the requirement of RFID data capture. For example, using SQLExecDirect to insert data one at a time into a database without using cache, the performance of the INSERT is 54 records per second [37]. By using bulk write (insert multiple data at a time), the performance is improved (e.g. if 1000 records are insert at a time, the insert speed is 98 records per second). Alternatively, message queue is a high speed data cache (24137 messages per second) [38] which is being gradually used in enterprise applications to achieve performance. The filtered and structured data is stored in a local message queue. As the message queue itself is a middleware that provides the enterprise application integration capability, the data stored in the local cache could be directly used by the enterprise application. This is to provide a buffer between the data capture module and data processing module in an environment where high volume of data is capture at high speed.

B. Device Integration

In an enterprise, real-time interaction between the application and device often becomes necessary. For example, a real-time status monitoring system may require direct access to the device. In this research, a novel tightly-coupled integration mechanism is provided that allows applications to directly operate the device remotely. This is accomplished by making the readers and devices as Remoting Objects [39]. A

distributed identification data capture infrastructure can be created because of the remote access feature of the Remoting Object.

VII. Performance Evaluation

An RFID network needs to capture RFID data without errors and in real-time. We conducted the performance evaluation on the RC by measuring the CPU Usage, Missed Reading Rate (MRR) (the number of tags missed) by varying the numbers of readers and tag reading rates (tags read per second). Virtual readers (software that generates identification numbers) are used to perform the high volume tag reads at high speed. The RC was tested on a Dell Precision 650 workstation with 2.0 GHz Xeon Processor and 1.0 GB RAM. The operating system was Windows Server 2003. The client was Dell Latitude X1 laptop that runs Windows XP with 512MB RAM.

Figure 7 shows the CPU Usage of the RC measured as a function of indicating system responsiveness (the capability of responding to a request from client). Higher CPU Usage implies that the system is less responsive. Based typical RFID device throughput, three unique-tag rates (50 tags/second, 100 tags/second and 200 tags/second) are measured. The graph shows that the CPU Usage increases as the number of readers and the unique-tag rate increases. As the number of controlled readers increases to 200, the CPU Usage is over 80%. At this rate, the system response is slow. Because the computation pushes the maximum usage of the CPU, the CPU usage increases at a slower rate. Thus, the missed data increased (as shown in Figure 7). We conclude that to ensure a responsive system, the number of readers controlled by RC should be optimized. If more readers need to be managed, more RCs should be installed.

MRR is measured by counting the difference between the number of tags generated by virtual readers and that stored in the locale cache. Figure 8 shows the MRR versus the number of readers by varying the tag rate. As indicated by the diagram, at 100 tags/second and 50 tags/second, even with 200 readers in the RC, the MRR is still zero. However, as the tag rate increases to 200 tags/second, there is a positive and increasing reading error for number of readers greater than 80. For 200 readers at read rate of 200 tags per second, the missed reading rate is about 0.05%. 200 unique tags captured in a second is still a rare requirement for typical industrial applications, but we expect that to change rapidly as industry starts to adopt RFID for scalable applications. Ideally one would like to capture all tags without errors. To avoid any reading error, the number of readers should be controlled for each RC.

VIII. Summary and Discussion

The RC is the edge server of the WinRFID AIDC system. It provides the ability to manage devices, and capture identification data. The identification data is filtered and aggregated before being sent to the local cache for further processing. There are two benefits of using an edge server in AIDC infrastructure: (i) The network traffic is reduced, and, (ii) the computing burden of the end application is reduced. The end result is that the enterprise application developer can effectively develop business logic processes via high level constructs and be relieved of the burden of writing low-level device operations and Data filtering/processing.

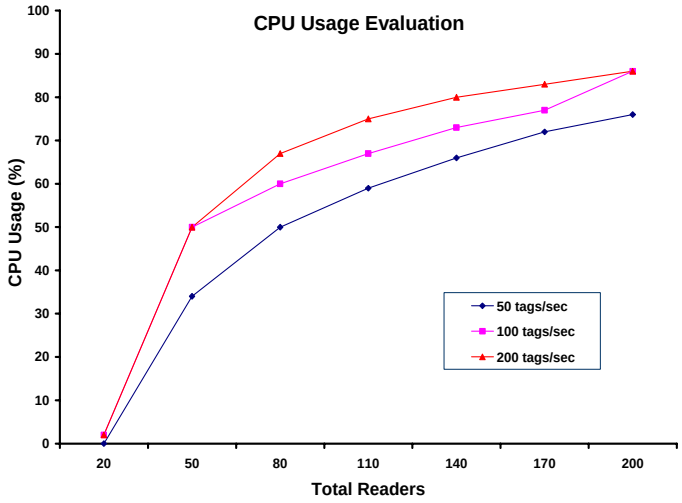

Figure 7 CPU usage

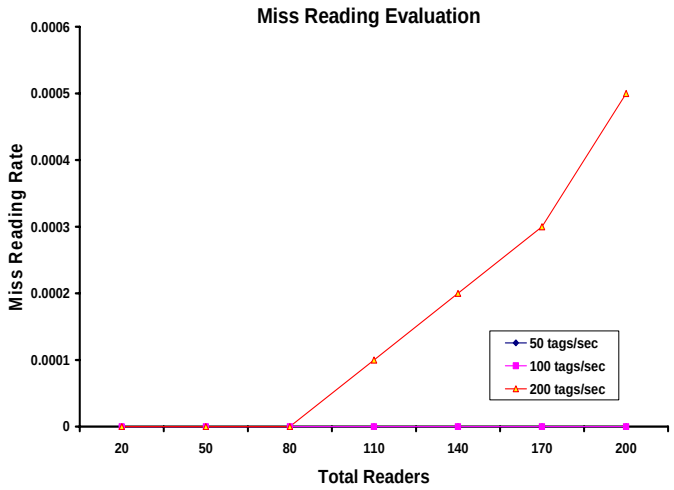

Figure 8 Missed reading rate

Because of the diversity of the hardware, it is challenging to manage various devices while achieving accuracy in the data capture process. The goal of the RC is to achieve a robust, scalable, efficient, flexible and high performance AIDC system. A unique layered structure is used in the RC, whereby our system is based on three layers: (i) the hardware layer, (ii) the information and logic process layer, and (iii) information presentation layer. Novel methodologies of abstracting device, processing logic and data are a key contribution in this research.

To achieve manageability and extensibility of the RC, the communication interfaces are separated from the device. Thus, single device driver can be used to drive multiple devices that have the same protocol but different communication interfaces. A unique abstraction uses XML based device description

allowing the user to define rules to control the behavior of the device. A control logic definition allows users to define the interaction between the data capture device and other controllable device such as PLC.

Various functional modules in the RC are based on industry standards including communication protocols, coding schemes and information exchange formats or methods. General data formats are delicately defined to accommodate various data formats that are captured by the identification devices. A hierarchical device naming mechanism is proposed and implemented in the RC. It provides an effective way to identify a device in the WinRFID network.

High performance data capture is achieved by data filtration and using message queues [40] for local cache. More intelligent and complex business related processing on the data is moved to other modules to reduce the load on the edge nodes, hence improving the system performance.

We evaluated the performance by measuring the CPU Usage and Missed Reading Rate. The results show that the RC works in a stable manner under high load. The results are extremely useful for planning of a real AIDC system by assigning proper number of readers to each RC. However, missed readings are observed in the presence of higher tag read rates – which is to be expected in the absence of sufficient computing power.

Currently, to integrate a reader into the RC, a driver still needs to be developed by following the abstraction method of the reader. In the future, a more general device description language based on XML could be defined. Once a new device is available for potential addition to the existing AIDC infrastructure, only the device description file is needed to be created according to specification thus eliminating the need to create a new driver. It simplifies the deployment process and improves the extensibility.

Another future research area is self-governing control of the reader network. Only simple logic that uses the output from reader or from device to control the device behavior or reader behavior has been developed. In the future, more advance control could be developed to fulfill more complex control. Due to the constraints of resources, we use a virtual reader to simulate a large number of RFID readers. The behavior of the system however provides only a first order simulation, but may not be able to reflect the constraints of a real AIDC environment.

APPENDIX A

```
<?xml version="1.0" encoding="utf-8"?>
<ReaderSetting>
  <ReaderIdentity ReaderID="1.warehouse1.abc.com "
GroupID="" Description="" Location="" />
  <ReaderInformation Vendor="SAMSYS" Model="9320"
RFType="UHF" />
  <Interface Type="RS232" Port="COM2" Baud="9600"
DataBits="8" Parity="none" StopBits="1"
HandShake="none" />
  <RunTimeParameters CommandTimeOut="1000"
SystemTimer="300000" Threshold="100">
    <General RFContinueMode="0" RFOnIdle="1"
SerialOutputMode="1" />
    <AntennaConfiguration EnableHopping="0"
EnableFailureMessage="1" InventoryRound="0">
      <Antenna No="1" Enable="1" />
      <Antenna No="2" Enable="0" />
      <Antenna No="3" Enable="0" />
      <Antenna No="4" Enable="0" />
    </AntennaConfiguration>
    <ProtocolConfiguration EPC1="0" EPC0="0"
ISO180006A="0" ISO180006B="1" EM="0" />
  </RunTimeParameters>
  <FilterSetting Enable="1" Size="1000"
EnableLifeTimeControl="0" LifeTime="100"
EnableIntelligent="0">
    <Conditions LogicalOperation="OR">
      <Conditions LogicalOperation="OR">
        <Condition Operation="NotEqual"
ClassName="SGTIN" FieldName="item_reference"
Value="991200" />
        <Conditions LogicalOperation="AND">
          <Condition Operation="NotEqual"
ClassName="SSCC" FieldName="serial_number"
Value="10002" />
          <Condition Operation="Greater"
ClassName="GRAI" FieldName="serial_number"
Value="11273773" />
        </Conditions>
      </Conditions>
    </Conditions>
  </FilterSetting>
  <RunningSchedule Enable="1">
    <Schedule Day="Sunday" Start="06:00:00"
Stop="18:00:00" />
    <Schedule Day="Monday" Start="06:00:00"
Stop="18:00:00" />
  </RunningSchedule>
</ReaderSetting>
```

REFERENCES

[1] B. Eckfeldt (2005) What does RFID Do for the Consumer? Communication of the ACM. Volume 48, No. 9, Sep. 2005, pp. 77-79.

[2] Borriello, Gaetano (2005). RFID: Tagging the World. Guest Editorial to RFID Special Issue. Communications of the ACM, September 2005, Vol. 48, No. 9, pp. 34-37.

[3] Ohkubo, M., K. Suzuki and S. Kinoshita (2005) RFID Privacy Issues and Technical Challenges. Communication of the ACM. Volume 48, No. 9, Sep. 2005, pp. 66-71.

[4] Andy Winans, “Sinnk or Swim? As you dive into the depths of RFID, be sure to avoid drowning in data”, *RFID Journal*, Dec. 12, 2005, http://www.rfidjournal.com/article/articleview/2009/1/82/

[5] Tadej Semenic, RFID Radio Study, *RFID System Course project report, WINMEC, UCLA*, Fall, 2005.

[6] Anish Shah, Hiral Patel, Tag Performance with Samsys 9320, *RFID System Course Project, WINMEC, UCLA*, Fall, 2004.

[7] Seung Ryong Han, Jim Shaughhessy, Comprehensive Reader and Tag Test (AWID), R*FID System Course Project, WINMEC, UCLA*, Fall, 2004.
[8] Juels, A., "RFID Security and Privacy: A Research Survey," *IEEE Journal on Selected Areas in Communications*, vol. 24, no. 2, pp. 381-394, Feb. 2006.
[9] John Goff, "Dude, Where's My Printer?", CFO Magazine, September 01, 2004. http://www.cfo.com/article.cfm/3127088?f=home_featured
[10] Rajit Gadh, B. S. Prabhu, "Radio Frequency Identification of Katrina Hurricane Victims", *IEEE Signal Processing*, March 2006.
[11] B.S. Prabhu, Xiaoyong Su, Charlie Qiu, Harish Ramamurthy, Peter Chu, Rajit Gadh, "WinRFID – Middleware for Distributed RFID Infrastructure", *International Workshop on Radio Frequency Identification (RFID) and Wireless Sensors*, Indian Institute of Technology, Kanpur, India, November 11-13, 2005
[12] Xiaoyong Su, Chi-Cheng Chu, B.S. Prabhu, Rajit Gadh, "On the Utilization and Integration of RFID data into Enterprise Information Systems via WinRFID", in the proceeding of *ASME 27th Computers and Information in Engineering Conference (CIE)*, September 4-7, 2007.
[13] Carvalho, H.S., Murphy, A.L, Heinzelman, W.B., and Coelho, C.J.N., 2003, "Network-Based Distributed Systems Middleware", Proc. of Intl. Middleware Conference, June 16-20, Rio de Janeiro, Brazil, 13-20.
[14] Colyer, A., Blair, G., and Rashid, A., 2003, Managing Complexity in Middleware. Proc. Of Second AOSD Workshop on Aspects, Components, and Patterns for Infrastructure Software (ACP4IS), Boston, Massachusetts, 21-26.
[15] LogicAlloy's ALE server feature, http://www.logicalloy.com/index.cfm
[16] Omnitrol's Edge Service Appliance, http://www.omnitrol.com/products-features.cfm
[17] Accada EPC network Prototyping platform, http://accada.org/
[18] EPCGlobal, EPC Redio-Frequency Identity Protocols: Class-1 Generation-2 UHF RFID Protocol for Communications at 860 MHz – 960 MHz, Version 1.0.9, January 2005.
[19] Joseph Davis, TCP/IP Fundamental for Microsoft Windows, Microsoft Corporation, May 21, 2006
[20] USB.org, USB 2.0 Specification, http://www.usb.org/developers/docs/
[21] RS485.com, Quick reference for RS485, RS422, RS232 and RS423, http://www.rs485.com/rs485spec.html
[22] Xml.com: XML From the inside out: http://www.xml.com/
[23] EPCGlobal, Reader Protocol Standard, Version 1.1, Ratified Standard, June 21 2006,
[24] Texas Instruments, Series 4000 Reader ISO 14443 Library Reference Guide, First Edition, October 2003, http://www.ti.com/rfid/docs/manuals/refmanuals/RF-MGR-MNMN-14443-refGuide.pdf
[25] Texas Instruments, Series 4000 Reader ISO 15693 Library Reference Guide, Second Edition, July 2004, http://www.ti.com/rfid/docs/manuals/refmanuals/RF-MGR-MNMN-15693-refGuide.pdf
[26] ISO/IEC JTC 1/SC 31 WG 4, Information Technology – Radio frequency identification for item management – Air interface, Part 1-7, Documents for review and comment, http://www.autoid.org/SC31/sc_31_wg4.htm
[27] David Booth, Hugo Haas, Francis McCabe, Eric Newcomer, Michael Champion, Chris Ferris, David Orchard, Web Service Architecture, W3C Working Group Note 11 February 2004, http://www.w3.org/TR/ws-arch/
[28] Erik Christensen, Francisco Curbera, Greg Meredith, Sanjiva Weerawarana, Web Servicces Description Language, W3C Note 15 March 2001, http://www.w3.org/TR/wsdl
[29] SAMSys, Reference Guide for SAMSys UHF Readers: Comprehensive Heuristic Unified Messaging Protocol, Version 6.0
[30] Applied Wireless ID (AWID), *MPR Serial Communication Protocol*, November, 2004
[31] FEIG, System Manual – ID ISC.MR/PR/PRH100/101-A
[32] Texas Instruments, *HF Reader System Series 6000: S6500/S6550 Configuration and Host Protocol Reference Guide*, August 2001
[33] Texas Instruments, *HF Reader System Series 6000: S6350 Midrange Reader Module RI-STU-TRDC-02 Reference Guide*, September 2002
[34] Texas Instruments, Series 2000 Reader System: ASCII Protocol Reference Guide, May, 2000
[35] Symbol technology, *XR400 Interface Control Guide, Revision A*, October, 2005
[36] EPCglobal, Inc. EPC Tag Data Standard Version 1.1 rev 1.27. May 2005.
[37] Bill Wilkins, "Tips for improving INSERT performance in DB2", *Universal Database, IBM*, 11, Mar 2004,
[38] Ingo Rammer, Richard Turner, "System Messaging Performance", Web Service Technical Articles, http://msdn2.microsoft.com/en-us/library/ms996476.aspx
[39] K. Sasikumar, .Net Remoting, C-Sharpcorner, Sep. 2, 2004, http://www.c-sharpcorner.com/Code/2004/Sept/NetRemoting.asp
[40] Microsoft Corporation, Queues Overview, MSDN technique document, http://msdn2.microsoft.com/en-us/library/ms733789.aspx
[41] Printronix, SL5000r MP and MP2 RFID Smart Label Printers: RFID Labeling Reference Manual.
[42] IBM Business Consulting Services, Global Data Synchronization, http://www-935.ibm.com/services/us/imc/pdf/g510-3990-global-data-synchronization.pdf