NIMBLE: Networked Identification and Management of Bovine on Large Expanses

6.033 DP #2 B. Horn TR11
Ryan Adams, Shantanu Sinha, Chris Taylor
{iceman, ssinha, taylorc}@mit.edu

Abstract

Herd management on large cattle ranches is a difficult task. Most functions on cattle ranches are done manually. As such, ranches stand to gain significant benefits with the aid of technology. In this paper, we present the design of NIMBLE, Networked Identification and Management of Bovine on Large Expanses. NIMBLE is an RFID based application that automates many functions that cattle ranchers must perform. NIMBLE aids ranchers with cattle identification and management tasks. The application offers several novel services including cattle location tracking, event notifications, and data access in the pasture. NIMBLE addresses the unique constraints of ranches that cover thousands of acres and manage thousands of cattle.

1. Introduction

Large ranches have the difficult task of managing their cattle herds. They must keep track of large amounts of information about their herds, including data on historical feedings and vaccinations, present whereabouts, and general profile information. The problem is especially challenging on large ranches which may contain anywhere from hundreds of acres to several thousands of acres of pastureland. Typical cattle herds can range from anywhere between a few hundred to several thousands.

Current cattle management processes are typically handled manually. Most cattle identification is done using numeric lables and paper records. When a herd of cattle must be vaccinated, a rancher begins by rounding up the herd and driving it into a large holding pen. From the pen, each individual cow is directed into a smaller pen which corresponds to a particular vaccination that must be administered. Cows are mapped into these smaller pens based on their numeric labels. Variants on this method are used for most identification, inventory, and classification tasks. Cow records, which are often stored on paper, are manually updated and transcribed into the filing system used at the ranch.

One can easily imagine the difficulty of using numeric labels and paper records for cattle herd management. When a cow must be identified, the numeric label must be physically read from the body of the cow. While in the pasture, the appropriate records must be located in order to determine the tasks scheduled for a particular cow. Finally, any information recorded in the pasture must be transcribed or filed at the ranch. Clearly, this manual system is prone to error and is an inefficient use of the rancher's time. The difficulties associated with managing large numbers of cows prevent many ranches from scaling to larger herd sizes.

RFID technology can offer ranchers significant productivity improvements. In this paper, we present NIMBLE, Networked Identification and Management of Bovine on Large Expanses. NIMBLE is an RFID-based application that is uniquely suited to the needs of cattle ranchers. While version 1.0 of NIMBLE specifically focuses on the needs of cattle ranchers, the application architecture can be readily extended to the needs of other types of ranchers as well.

NIMBLE uses RFID technology, a sensor network, a central database and a thick-client user interface to automate the maintenance and management of large cattle ranches. NIMBLE's design scales to ranches that cover thousands of acres and maintain thousands of cows.

The use of RFID technology automates the cattle identification process. Passive, electronic tags embedded in the cattle allow RFID readers to quickly and accurately identify cattle. RFID readers are attached to a mesh network of mote computer nodes located throughout the pasture. Each network node allows information about the herd to be sent back to the ranch from the pasture. The central database automates the storage of individual records for each cow. The thick-client user interface ensures that ranchers have access to all of their information while they are in the pasture. Finally, the introduction of RFID technology also provides benefits beyond what was possible with a label-based paper system, such as cattle location tracking, event notification and data access in the pasture.

The rest of this paper is organized as follows. Section 2 describes the assumptions and requirements behind NIMBLE. Section 3 provides a high-level overview of the system model and application architecture. Section 4 explores the major design decisions and trade-offs made in NIMBLE's design. Section 5 discusses the performance, scalability and feasibility characteristics of NIMBLE. Finally, section 6 concludes this paper with a summary of NIMBLE.

2. Design Assumptions & Requirements

NIMBLE's design makes several assumptions about typical large ranches. First, large ranches are typically organized hierarchically into pastures partitioned into subdivisions usually 100-200 acres in size, which we will call major pastures. Each of those subdivisions is subsequently divided into smaller sections, typically 10-20 acres large, which we call minor pastures. A typical large ranch may have hundreds of cows grazing within a major pasture.

The goal of NIMBLE is to automate herd management and identification of large cattle ranches, covering thousands of acres and maintaining thousands of cattle. Specifically, NIMBLE assists ranchers with the following tasks:

- *Location tracking* – present cattle locations should be accurately available to a range of major pastures within a 6 hour period. A 6 hour period provides a reasonable amount of locality in the attempt to find a particular cow.
- *Profile management* – herd information must be available to ranchers whether they are at the ranch or in the pasture, both for input and output.
- *Event notifications* – the application must be able to notify the rancher in response to salient events. Some examples of interesting events include vaccination notifications, feeding notifications, and lost cow notifications, among others.

Since most ranchers are not savvy technology users, the user interface must be extremely simple.

NIMBLE is most likely extensible to the management of other types of livestock. However, the application is designed with the specific requirements of cattle ranches in mind.

Finally, while cost is a significant consideration in the design of the application, completeness and correctness of design prevails when trade-off decisions are made.

3. System Model

NIMBLE consists of three major components: an RFID sensor network, a data repository, and a user interface. A high-level model is illustrated in Figure 1.

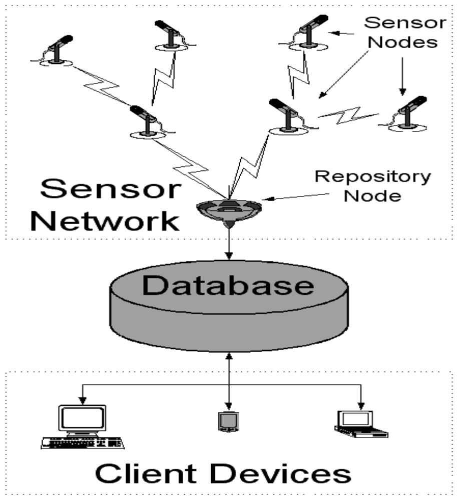

Figure 1 System Architecture

The sensor network tracks the locations of cattle in the pasture. It generates a reasonably current stream of cow-location events. A cow location event represents the sighting of a cow at some location in the pasture. Each cow is implanted with an RFID tag containing a globally unique tag ID.

An array of RFID readers located throughout the pasture reads these tags, generates cow-location events, and sends them to the repository node. The network topology is a mesh of nodes that maintain routes through neighbors to a special node designated to be the repository node. The repository node is configured at deployment time and is directly connected to the database.

The database consists of three components, the data repository, the notification thread and the synchronization module. The data repository stores a set of bindings that map RFID tag IDs to cows. Each binding is associated to relevant data such as a cow's vital statistics: weight, health, vaccination history, identifying marks, genealogy, etc. As cow location events arrive from the sensor network, they are added to the data repository, so that the repository contains a coarse location history for each cow. The notification thread analyzes data obtained from the pasture and generates notifications of interesting events. The synchronization module allows the data repository to be accessible off-line, to allow a user to take data into the pasture.

The user interface will be initially available for laptops and PCs. The client allows users to generate reports about the herd, edit vital statistics, add new animals, and flag dead, lost, or sold animals. While on the network (typically at the ranch), the client interacts with the data repository directly. When offline, the client reads and modifies a local snapshot of the database that is synchronized with the repository when the client returns to the network.

4. System Details

This section covers the details of the major components of NIMBLE. Each subsection contains a discussion of the trade-offs and decisions behind the design of the application.

4.1. Tagging

NIMBLE presents a unique set of requirements for RFID tag technology. This section provides a discussion of these requirements, and presents the advantages of the technology that was selected for NIMBLE.

4.1.1. Tag Requirements

The cattle population in the United State is approximately 100M, with approximately 45M calves born each year [8]. Assuming that this rate is maintained, there will be approximately one billion cows born over the next 25 years. RFID tags that support tag IDs greater than 64 bits clearly provide a sufficient namespace.

Because NIMBLE only determines cattle locations to the granularity of a major pasture, passive RFID tags must offer enough range to cover the entire width of a pasture gate. Typical pasture gates are 7 meters wide. Passive RFID tags are preferable to active tags for several reasons. First, passive tags are cheaper than their active counterparts. Second, typical battery life for an active tag is less than 5 years, which is shorter than the lifespan of a calf-bearing heifer. Once the battery dies, the cow would need to be retagged and its information would need to be updated to match the new tag.

The use of tags with ranges that cover the width of a gate allows large movements of cattle to be tracked by placing readers at the junctions between pastures. Mounting a reader in the middle of a junction is unreasonable, since it would be susceptible to trampling and destruction. Thus, a functional minimum range for the RFID technology is approximately 7 meters.

Fortunately, cows do not typically move very quickly. Furthermore, the large size of cattle will prevent extremely high densities of cattle within proximity of the reader. Thus, data will not need to be sampled at a very high frequency. Cattle typically move no faster than 2-3 miles per hour (3-4 feet per second). Therefore, a reading rate of 1-2 RFID tags per second should be sufficient to capture most cow-location events.

4.1.2. RFID Tags

The range requirement places the largest restriction on the selection of RFID technology. Two types of RFID tags meet NIMBLE's requirements. They operate in two different frequency ranges, UHF (868 to 915MHz) and Microwave (2.45GHz). Standard RFID tags utilize the 13.56MHz frequency range but their ranges do not exceed one meter. Thus, standard tags are not a viable option for NIMBLE. Microwave tags yield the largest distances, but currently available microwave tags are prohibitively expensive and have large power consumption requirements.

The EcoTag, developed by Trolley Scan of South Africa, is an example of a suitable UHF tag. Measurements conducted using passive transponders have demonstrated a range of 9-12 meters. Moreover, power consumption at the reader is less than 0.5 Watts. The EcoTag system can read up to 20 tags per second.

The EcoTag transponder is approximately 80mm long, 27mm wide and credit card thin [6]. This small size allows an EcoTag to be embedded within a conventional cow ear tag, illustrated in Figure 2. Conventional ear tagging technology utilizes tear resistant polyethylene with ultraviolet inhibitors to prevent solar deterioration. As a result, NIMBLE ear tags are waterproof and resistant to damage from the sun. Existing applicator equipment can be used to deploy tags to cattle. Initial tags may be deployed by the same processes that traditional tags are deployed (most likely at the vaccinations or "roundups").

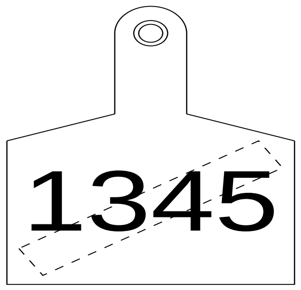

Figure 2 A standard cow ear tag with an embedded RFID tag

The cost of the transponders varies with volume, but a reasonable expected cost is $5 per tag. Though this price is somewhat expensive, the tags are necessary to meet NIMBLE's requirements. We expect the prices to fall as UHF tags become more widespread. EcoTag transponders support 70-bit tag IDs, but for the purposes of NIMBLE, we will only use 64-bits. The EcoTag RFID readers provide an RS-232 interface, allowing them to easily interface to an embedded computer.

4.2. Sensor Network

The sensor network is the mechanism that relays information from the pasture back to the ranch home. This section describes the sensor network in detail. It explores the design of a sensor node and describes the network routing protocol. Finally, this section analyzes the power and price characteristics of sensor nodes.

4.2.1. Node Architecture

A NIMBLE sensor node contains the following components: an RFID reader, a microcomputer, a power supply, and one or more network devices. The microcomputer is a Berkeley mote with an 8Mhz processor and 32K of RAM, and consumes extremely low power. NIMBLE utilizes a solar power/battery combination. The power supply is discussed in section 4.2.8. The network device is a low-bandwidth RF modem, capable of transmitting to nodes 6-7 miles away.

Each RFID reader generates a stream of cow location events that correspond to cows entering a reader's range. The mote archives and timestamps the location event, storing it in a hash table that maps tag IDs to timestamps. If the mote receives multiple location events for a single tag ID, it discards all but the most recent.

Every six hours, the mote creates a message containing the contents of the hash table and sends it over the network to the repository. Once the message is transmitted, the table is cleared to provide space for new event storage. The event table will use all available RAM. Since cows are not very mobile, we do not expect a large number of location events to occur within a 6-hour period. Thus, a mote's hash table should approach the size of RAM very rarely, if ever. Having said that, NIMBLE's message delivery protocol handles this case readily; this protocol is described in greater detail in section 4.2.5.

4.2.2. Node Placement

Most ranches are subdivided into fenced-off major pastures of 100-200 acres each. NIMBLE sensor nodes are stationed at the gates between pastures, so that a cow moving from one major pasture to another must pass a sensor node. Since cows move between pastures periodically through random movement and planned herding, NIMBLE sensor placement provides enough data to locate a cow to the granularity of a major pasture.

Each node is also assigned a node ID, which is registered in the data repository before deployment. This node ID allows the origin of network messages to be determined. Since most pastures are organized in a way that optimizes fence costs, we expect that a 200 acre major pasture will be fenced off as a 3000 feet x 3000 feet enclosure. As a result, the distance between two nodes is likely to be a mile or less; the RF modem provides ample coverage. In fact, each node will most likely be able to communicate several other nodes, which provides additional network redundancy.

4.2.3. Routing Protocol

In order to route messages from a source node to the repository node, the nodes in the sensor network utilize a simple routing protocol.

One node is designated as the repository node at deployment time. The repository node is a sensor node that has message delivery privileges to the data repository. Each sensor node knows of a single route to the repository node, called its path. In fact, the nodes and routes form a tree with the repository node at the root. Routes are advertised "down" the tree (away from the root), and messages are sent "up" the tree (toward the root). The path contains a list of nodes through which a transmitted message will pass with high probability on its way to the repository (in the case of a node failure, a route could change while a message is being transmitted).

4.2.4. Route Discovery

Routes are discovered and updated using advertisements. Every half-hour, each node in the system broadcasts the route from itself to the repository. When a node receives an advertisement, it may choose to replace its route information with the new advertised path through the advertising node.

A node responds to an advertisement as follows. First, it checks if the advertised path contains its own node ID. If its node ID is present, the advertised route is discarded to prevent routing cycles. Second, if the receiving node does not already have a route to the repository node, the advertised route becomes that route. Otherwise, the advertised route is compared with the stored route. If the advertised route is shorter in the number of hops, it is accepted as the new stored route. The advertised route is also accepted if the advertised route begins with the same node as the stored route, allowing effects in the upstream network to be propagated to a node.

The routes generated by this protocol stabilize when each node knows the shortest available route to the repository node. In this case, route length is equivalent to the number of intermediate nodes. This stabilization should occur within 30*d minutes of the most recent node addition/removal, where d is the depth of routing tree. Depending on system capacity measurements, advertisement periods can be adjusted without loss of generality. Route stabilization occurs automatically, without human input.

Failed nodes will cease to advertise routes. As a result, messages from downstream nodes will not reach the repository node. If a downstream node does not hear an advertisement from an established route (i.e. an advertisement such that the first node in the advertised route matches the first node in the path), the node will clear its path. The node will then accept the next new route advertisement it hears.

Because nodes must be registered with the data repository before being deployed, node failures can be detected by the database, which flags them for attention. See the section 4.3.1 for details. The following pseudocode outlines the route discovery process:

```
void init() {
if (myID() == repositoryID)
  path = [];
else
  path = NULL;
}

void advertise() {
  // to be executed every 30 minutes
  if (path != NULL)
    transmit(cons(myID(),path));
}

void receive_route(p) {
  if (member(myID(), p))
    return;
  if (path == NULL ||
      first(p) == first(path) ||
      length(p) < length(path))
    path = p;
}

void onTimer() {
  if (haveNotHeardInLastHour(
        first(path)))
    path = NULL;
  ... other timer code ...
}
```

4.2.5. Message Delivery

NIMBLE messages are data packets containing the following fields: 16-bit node ID, a 32-bit current timestamp, an 8-bit value indicating the number of events contained in the packet, and a list of events. Each event is a 64-bit RFID tag ID followed by a 32-bit time-stamp.

A node delivers a packet by transmitting the packet pre-pended with a header containing a 16-bit destination node ID. The destination is selected by choosing the first node from its path. This node is then responsible for forwarding the packet to the repository.

Packet forwarding is also straightforward. First, the receiving node confirms that it is the intended recipient by comparing the destination field of the packet header to its nodeID. If the values do not match, the packet is discarded. Otherwise, one of two things happens. If the receiving node is the repository node, the packet is delivered to the repository. If not, the receiving node forwards the packet to the first node in its path.

The packet forwarding system also enables the database to detect node failures. Since all nodes are registered with the repository and must send a packet every six hours, any node that has not sent a packet in twelve or eighteen hours has most likely experienced an equipment failure, either in the node itself or in one of the intermediate nodes along its route. These failures are flagged for attention so that the node can be repaired.

The following pseudo-code illustrates NIMBLE's packet delivery protocol:

```
void send(packet) {
  if (path != NULL)
    transmit(first(path), packet);
}

void forward(nexthop, packet) {
  if (myID() != nexthop)
    return;
  if (myID() == repositoryID)
    addToRepository(packet);
  else
    transmit(first(path),packet);
}
```

If a node must send a large number of events within a 6-hour window, sending all events in a single message could potentially overrun the receive buffer of intermediate nodes. To handle this contingency, sensor nodes fragment large messages into a series of smaller independent packets, each containing its own header. Each fragment is sent serially with a delay of 1-2 minutes between each transmission. This approach permits a large number of events to be sent without risking overrunning intermediate node receive-buffers.

4.2.6. Lost Messages

Most RF modems offer some link-level error detection and correction. Wireless links typically exhibit 1% packet corruption rates [1]. Since cow location data is not mission critical data, NIMBLE simply drops corrupted packets. The loss incurred due to packet corruption is tolerable. The alternative approach is to implement an acknowledgment based message protocol that ensures message delivery. However, implementing such a system imposes significant complexity in the transmission protocol and is not worth the cost.

4.2.7. Line of Sight and Reliability

The network architecture discussed above is more than likely sufficient for most NIMBLE deployments. However, NIMBLE RF modems depend on line of sight. Although most ranches are large expanses of flat land, on some ranches, hills and other fixed obstacles could limit the range of node transmissions. Furthermore, the tree topology created by the routing protocol could leave large sub-networks vulnerable to node failures. For instance, a node failure near the root of the tree may prevent a substantial fraction of the sensor network from delivering packets. Thus, we propose two alternative network topologies, both compatible with NIMBLE's network protocol.

The addition of relay stations provides a viable solution to the problems described above. A relay station is a standard sensor node with the RFID reader removed. A relay station is less costly than a sensor node in both the power and price dimensions. Relays execute the same routing protocol as sensor nodes. They are deployed on fence lines or on tall poles in the pasture to bridge gaps between sensor nodes. They can also provide alternative paths for packets in the event of an upstream node failure.

Reducing routing tree depth may sometimes be necessary, particularly on very large ranches. High routing tree depth poses two problems. First, deep nodes face a large number of potential failure points along the message delivery path. Second, nodes near the root could face traffic that exceeds available system resources. One possible solution is to deploy a hierarchical topology, with high powered "supernodes," having a directional antenna that connects directly to the repository or another supernode. Thus, the network topology would consist of clusters of network nodes communicating to the repository through these supernodes.

4.2.8. Power and Price Considerations

The EcoTag reader requires continuous power of 0.5W. Thus, the reader consumes 12 W-hrs of energy, daily. The EcoTag reader costs approximately $100.

A 9600 baud RF modem costs approximately $80. It consumes 150 mA during transmission, 50 mA during reception, and 1 μA when idle, at 5 VDC [2]. A data packet of 1.8 kilobytes requires 0.2 seconds of transmission time, corresponding to 0.042 mW-hrs of energy per transmit, and 0.056 mW-hrs of energy per forward. A conservative upper-bound of twenty packets sent by a node per day, forwarding packets for fifty other nodes, requires a daily energy consumption of 56.84 mW-hrs.

A mote computer consumes approximately 1.75 W-hrs/day, and costs approximately $25 when purchased in volume [3].

NIMBLE nodes are primarily powered by solar panel. We expect that the daily energy consumption of a NIMBLE sensor node to be less than 14 W-hours. To withstand periods of bad weather, a NIMBLE node may operate for up to 10 days without sunlight. A lead-acid battery with a 36 Amp-Hour capacity at 6 Volts costs approximately $40 [4]. A 5-Watt solar panel supplies 40 W-hrs of power on a sunny day and can be purchased for about $70 [5]. A solar panel of this capacity should be able to supply sufficient power to the sensor node and recharge the battery simultaneously. Thus, standard sensor node components should cost approximately $315. Field experience will most likely reduce costs.

4.3. Central Database

The central database serves as a sink for all messages received and processed by the repository node. It must support full querying, including the ability to search on any field, perform joins and updates, and support synchronization with snapshots that have been altered off-line. The database is a very simple object which consists of three components: a data repository, a notification thread and a synchronization module.

The structure of the data in the repository is intentionally defined abstractly. Using abstract data structures allows the implementer to choose the data storage mechanism that provides the most cost-benefit effectiveness (whether it be an RDBMS, XML, or otherwise). As long as the database implementation supports searching, querying, and synchronization, any data storage system will suffice.

4.3.1. Data Repository

The data repository is responsible for maintaining a simple, generic data structure consisting of a set of bindings, each of which is associated to three types of data. Figure 3 illustrates the NIMBLE central data repository data structure.

The bindings are name-value pairs mapping RFID tag IDs to cow identifiers. A cow identifier is the name assigned to a particular cow and is locally unique within the scope of a ranch. This cow identifier is probably the numerical label currently used to tag cattle and is analogous to SKU numbers in inventory management applications.

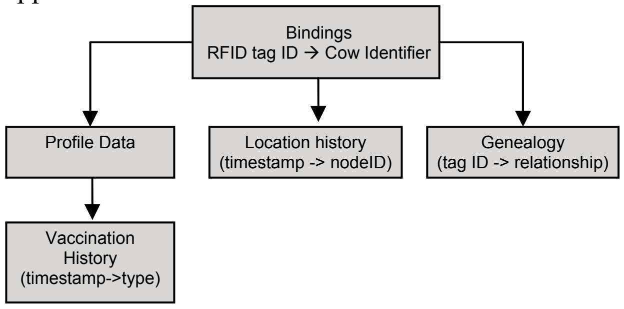
Bindings
RFID tag ID → Cow Identifier
Profile Data
Location history (timestamp -> nodeID)
Genealogy (tag ID -> relationship)
Vaccination History (timestamp->type)

Figure 3 NIMBLE Data Repository Data Structure – each grey box represents a complex data type.

The advantage of using bindings to cow identifiers, rather than directly using RFID tag IDs, is straightforward. Each ranch can implement its own local naming scheme. In actuality, each ranch will most likely have a naming system already in place. Using cow identifiers allows a ranch to easily port their current naming system into NIMBLE, rather than requiring them to convert their records to use RFID tag IDs.

Each cow identifier is associated to three types of data. Because simple bindings are used as the primary data structure in the database, additional types of data can be added, if required in the future. However, we note that the developers of NIMBLE would be the ones expected to extend the data structure of NIMBLE for future versions.

First, a location history table maintains data on where the cow has grazed over the last year. This data allows ranchers to determine effective utilization of the pasture and last known locations of their cattle.

Second, each cow identifier is mapped to profile data. Profile data consists of health information, distinguishing marks, value, origin (i.e. where it was obtained or where it was born) and vaccination history. Vaccination history is non-scalar data consisting of a list of name-value pairs, mapping timestamps to a given vaccination.

Finally, each cow identifier is associated to a genealogical history. Genealogical data is a list of name-value pairs: the name is the RFID tag IDs of a related cow, and the value is the relationship (i.e. mother, father, sibling, etc.). Using RFID tag IDs in this case allows genealogy data to be easily transferred between multiple installations of NIMBLE, a useful feature when a cow is traded between different ranches.

The data repository also maintains meta-data about the NIMBLE sensor network. The data repository maintains a list of node IDs each marked by the timestamp of the last message received from that node. This information can be used to detect dead nodes and other network faults.

No data is ever deleted from the data repository. If a cow dies or is sold, its entry in the database is simply marked 'inactive'. Space may become a concern eventually, but hard drive capacity is cheap enough such that we can ignore this problem. However, eliminating deletes simplifies the design of the synchronization module, described in section 4.3.3.

4.3.2. Notification Thread

The second component of the database is the notification thread. The notification thread periodically scans the database, searching for interesting events. Examples of interesting notifications include the following:

- No recent location information reported for a particular cow identifier (i.e. indication of potential cow death or tag failure)
- Vaccinations are due for a particular cow
- Too many cows reported in a particular perimeter
- Several cows have appeared in a new location without checking out of their old location (an indication of a possible fence break or similar fault)
- Silent node (possible indication of node failure or other network fault)

Each notification is configured to occur on one of two events, a data event or a timer event. A timer event is the expiration of some timer. A data event is the arrival of new data from a source, such as the network or the user. For example, vaccination information is checked daily. However, the broken fence check is performed in response to data events (in this case, when new data arrives from the network).

While we do not expect ranchers to implement new notifications, new notifications can be added easily. They are specified through a text file, which allows NIMBLE developers to create new notifications and quickly update a running system.

4.3.3. Synchronization Module

The NIMBLE database supports the ability to generate snapshots of the database for off-line use and synchronize with snapshots that have been updated off-line. The following section describes how the synchronization module is used by users.

To provide synchronization capabilities, data items are marked with a GUID and a timestamp indicating the last update time. A GUID is a globally unique identifier that can be generated with a negligible probability of collision. A data item is any name-value pair. A snapshot is an exact copy of some part of the repository, possibly in entirety, including the GUID and timestamp data. When a snapshot is synchronized with the central data repository, all data items with new GUID values are added to the data repository. If an existing data item was modified off-line, the synchronization process will alert the user that a conflict was found and ask the user which copy of the data is accurate. In most cases, the data item with the most current timestamp is be the most accurate, though this may not always be the case (e.g. if multiple off-line snapshots are modified in parallel). As a result, only the user can correctly resolve conflicts. Because deletes are disallowed, the synchronization module does not have to handle this case.

4.4. User Interface

The NIMBLE user interface is the rancher's point of entry into the system. The user interface is a simple tool that allows a rancher to analyze the data NIMBLE collects. The user-interface supports two modes of operation, on-line and off-line. In on-line mode, the user interface directly interfaces with the data repository. In off-line mode, the user interface makes a local copy of data repository and interfaces directly to it. Because the data repository could become very large, off-line snapshots will consist of profile and genealogical data and only the most recent location notifications for cattle.

4.4.1. Thick-client Versus Thin-client

Ranchers need to access to herd data while they are out in the pasture. Since most pastures do not have access to the Internet, a thin-client application is not feasible. As a result, the user interface is a thick-client application deployed on a laptop. Future versions of NIMBLE may offer a PDA based user-interface.

The laptop contains a tool that obtains a snapshot of the database for off-line use and synchronizes it to the central repository once the laptop is connected at the ranch.

4.4.2. Transaction Batching

To support full off-line capabilities, the user interface is built to support the batching of transactions. While off-line, the primary use of the user interface is to record profile updates. These profile updates are stored as a series of transactions. Each transaction is formatted as follows:

```
Begin
   <ID>
   <Cow ID>
   <Data Item>
   <Data Value>
   <Timestamp>
End
```

The `ID` entry is the GUID of the data being inserted or modified. The `Cow ID` entry indicates the cow identifier pertinent to the data being inserted. The `data item` entry indicates which type of data is being inserted (i.e. vaccination entry, genealogical entry, or some kind of profile data). The `data value` entry is the actual data value to be stored in the repository. The `timestamp` entry indicates when the transaction entry was created. The transaction format is identical whether used for entering a new cow found in the pasture (e.g. when a calf is born) or when updating prior information for a particular cow.

When an off-line user interface makes a change, the local data repository is updated and a transaction record for the change is stored. These updates are necessary so that the user can accurately query the local database. When the laptop is synchronized with the central data repository, the transactions are replayed into the central data repository and subsequently flushed from the laptop.

One concern is that new data items may be recorded by multiple user interfaces in the pasture. As a result, each data item will create a new GUID, even though the data items represent the same event. For example, suppose a calf is born and multiple off-line interfaces create a new binding for the calf's tag. When those interfaces synchronize with the central data repository, a conflict will occur since one RFID tag ID will be inserted into the repository twice. The user will be notified of this conflict and will be required to resolve it manually by selecting the correct values. However, we expect that ranches will rarely have multiple laptops being used to record the same data, so these conflicts should be rare.

5. Discussion

5.1. System Availability

RFID readers claim to have MTBF value of 50,000 hours [7]. We estimate from empirical experience that a typical ranch will lose one tag per hundred head of cattle per year to predators such as coyotes. The RF network component of the design has a range capability of six to seven miles, exceeding the distance across a typical pasture by a factor of ten. The combination of the large RF modem range and the dynamic routing discovery discussed in section 4.2.4 enables the NIMBLE system to tolerate a large number of node failures.

The largest concern for the database is the loss of data. Many data backup schemes exist as recovery strategies, but are outside the scope of this paper. The system also tolerates database downtime since the thick-client architecture can provide off-line access. Although location information will not be accessible while the database is down, a rancher can continue accessing herd data.

5.1.1. Scalability

Scalability is an important design requirement of NIMBLE. The number of readers required scales with the area of the land and the number of RFID tags scales with the number of cows. Two facts enable NIMBLE to scale to very large ranches. First, NIMBLE provides only coarse grained location information, accurate to within 6 hours. Second, cows are not very mobile. We expect that a cow will generate no more than 2 to 3 location events within a 6 hour window. On a ranch with 10,000 cows, this number of events corresponds to a maximum of 21 bytes/message * 3 messages/cow * 10,000 cows = 610,000 bytes of data per 6 hour window.

5.2. Performance

One scenario presents a potential performance problem for NIMBLE. If an extremely large herd of cattle moves through a gate quickly enough, the reader may not interrogate all the RFID tags. In this case, some location events are inevitably missed. However, readers have ranges on the order of 10-12m and interrogation rates up to 20 tags per second. Since cows typically move at only a few meters per second at peak speeds, the probability of this occurrence is low.

6. Conclusion

In this paper, we have presented the design of NIMBLE, Networked Identification and Management of Bovine on Large Expanses. NIMBLE is expected to scale well for large ranches covering thousands of acres and managing thousands of cattle. We expect that NIMBLE will improve the efficiency of cattle management and hope that the application allows ranches to scale beyond their current systems.

References

[1] Balakrishnan, H. Katz, Randy. Explicit Loss Notification and Wireless Web Performance. Proc IEEE GLOBECOM Global Internet Conf., Sydney, Australia, Nov. 1998.
[2] http://www.maxstream.net/products_standalone.html
[3] Mote Comparisons. http://www-bsac.eecs.berkeley.edu/~shollar/webthesis/formthe139a.html
[4] PlanetBattery.com. http://www.batteryplanet.com
[5] Northern Arizona Wind \& Sun. http://www.solar-electric.com
[6] Trolley scan ltd. http://trolleyscan.co.za/.
[7] Allex iso compatible rfid reader module. http://www.allexboulder. com/Docs/datasheet/OEMModData.PDF.
[8] United states census of agriculture, 1997.