The University of Kansas

KU INFORMATION & TELECOMMUNICATION TECHNOLOGY CENTER
The University of Kansas

Technical Report

An Open System Transportation Security Sensor Network: Field Trial Experiences

Daniel T. Fokum, Victor S. Frost, Daniel DePardo,
Martin Kuehnhausen, Angela N. Oguna,
Leon S. Searl, Edward Komp, Matthew Zeets,
Daniel D. Deavours, Joseph B. Evans,
and Gary J. Minden

ITTC-FY2010-TR-41420-21

March 2010

Project Sponsor:
Oak Ridge National Laboratory
Award Number 4000043403

Abstract

Cargo shipments are subject to hijack, theft, or tampering. Furthermore, cargo shipments are at risk of being used to transport contraband, potentially resulting in fines to shippers. The Transportation Security Sensor Network (TSSN), which is based on open software systems and Service Oriented Architecture (SOA) principles, has been developed to mitigate these risks. Using commercial off-the-shelf (COTS) hardware, the TSSN is able to detect events and report those relevant to appropriate decision makers. However, field testing is required to validate the system architecture as well as to determine if the system can provide timely event notification. Field experiments were conducted to assess the TSSN's suitability for monitoring rail-borne cargo. Log files were collected from these experiments and postprocessed. We present the TSSN architecture and results of field experiments, including the time taken to report events using the TSSN as well as on the interaction between various components of the TSSN. These results show that the TSSN architecture can be used to monitor rail-borne cargo.

Contents

LIST OF FIGURES

LIST OF TABLES

I. INTRODUCTION

IN 2006 the FBI estimated that cargo theft cost the US economy between 15 and 30 billion dollars per year [1]. Cargo theft affects originators, shippers, and receivers as follows: originators and receivers need a reliable supply chain to deliver goods in a timely and cost-effective manner. Shippers hold liability and insurance costs for shipments, which are proportional to the rate of theft. Finally, receivers are impacted by out-of-stock and scheduling issues due to cargo theft. Most non-bulk cargo travels in shipping containers. Container transport is characterized by complex interactions between shipping companies, industries, and liability regimes [2]. Deficiencies in the container transport chain expose the system to attacks such as the commandeering of a legitimate trading identity to ship an illegitimate or dangerous consignment, hijack, or the theft of goods. Insufficiencies in these areas can be overcome by creating secure trade lanes, or trusted corridors, especially at intermodal points, for example, at rail/truck transitions. Research and development is underway to realize the vision of trusted corridors.

The work described here focuses on: advanced communications, networking, and information technology applied to creating trusted corridors. The objective of the research is to provide the basis needed to improve the efficiency and security of trade lanes by combining real-time tracking and associated sensor information with shipment information. One crucial research question that must be answered in order to attain this objective is how to create open technologies that will allow continuous monitoring of containers by integrating communications networks, sensors as well as trade and logistics data. This integration must occur within an environment composed of multiple enterprises, owners, and infrastructure operators.

To achieve improved efficiency and security of trade lanes, we have developed a Transportation Security Sensor Network (TSSN) architecture, which uses Service Oriented Architecture (SOA) [3] principles, for monitoring the integrity of rail-borne cargo shipments. The TSSN is an open system where different components can be provided by different vendors. The TSSN is composed of a Trade Data Exchange (TDE) [4], Virtual Network Operations Center (VNOC), and Mobile Rail Network (MRN). The functions of each of these components are discussed in greater detail in Section II. The TSSN detects events, integrates the event type from the train in the field with logistics information, and then reports those events that are important to decision makers using networks with commercial links. Ideally, decision makers would be notified of events within 15 minutes [5] so that they can take effective action. For the TSSN to be deployed, we need to validate its architecture and understand the timeliness of the system response. Work by Arsanjani *et al.* [6] and Saiedian and Mulkey [7] shows that service oriented architectures introduce overhead. As a result, we want to determine whether an SOA-based system such as the TSSN provides timely event notification. TSSN event notification is also impacted by unpredictable packet latency on commercial networks and the use of of e-mail and/or Short Message Service (SMS) [8] for event notification. Thus, we have designed and implemented hardware and software needed to realize a prototype of the TSSN and carried out experiments [9] to characterize the system, particularly the end-to-end time between event occurrence and decision maker notification using SMS or e-mail as well as the impact of SOA overhead.

In this paper we provide a high-level description of the TSSN architecture and document two field experiments that were carried out to demonstrate that sensors and software based on an open service-oriented architecture can be used to monitor cargo in motion. Our experiments focussed on determining the time from event occurrence to decision maker notification as well as testing functionality between the component services of the prototype TSSN. Our experimental results show that decision makers can be notified of events on the train in a timely manner using the prototype TSSN architecture. The rest of this paper is laid out as follows: In Section II we describe the TSSN architecture including the components and the prototype hardware implementation. Section III discusses experiments conducted to assess the suitability of the TSSN system for cargo monitoring. In Section IV we discuss the framework used to postprocess the log files from our experiments. Section V presents empirical results showing the interaction between various components of the TSSN. Refinements to the TSSN given our field trial experiences are discussed in Section VI. In Section VII we present related research on monitoring trains and securing shipping containers in motion. Section VIII provides concluding remarks.

II. System Architecture

To achieve the vision of a trusted corridor we have designed and implemented a prototype of the Transportation Security Sensor Network (TSSN) architecture. The detailed architecture of the TSSN, including system extensibility, is found in [10], whereas this section gives an overview of the TSSN. The architectural details discussed here are important in understanding the experiments and results presented in Sections III and V, respectively.

The SOA and web services used in the TSSN enable the integration of different systems from multiple participating partners. Moreover, the use of SOA and web services enable data to be entered once and used many times. Using commercial off-the-shelf (COTS) hardware and networks, as well as an open systems approach, the TSSN is able to detect events and report those relevant to shippers and other decision makers as alarms. Furthermore, the TSSN supports multiple methods for notifying decision makers of system events.

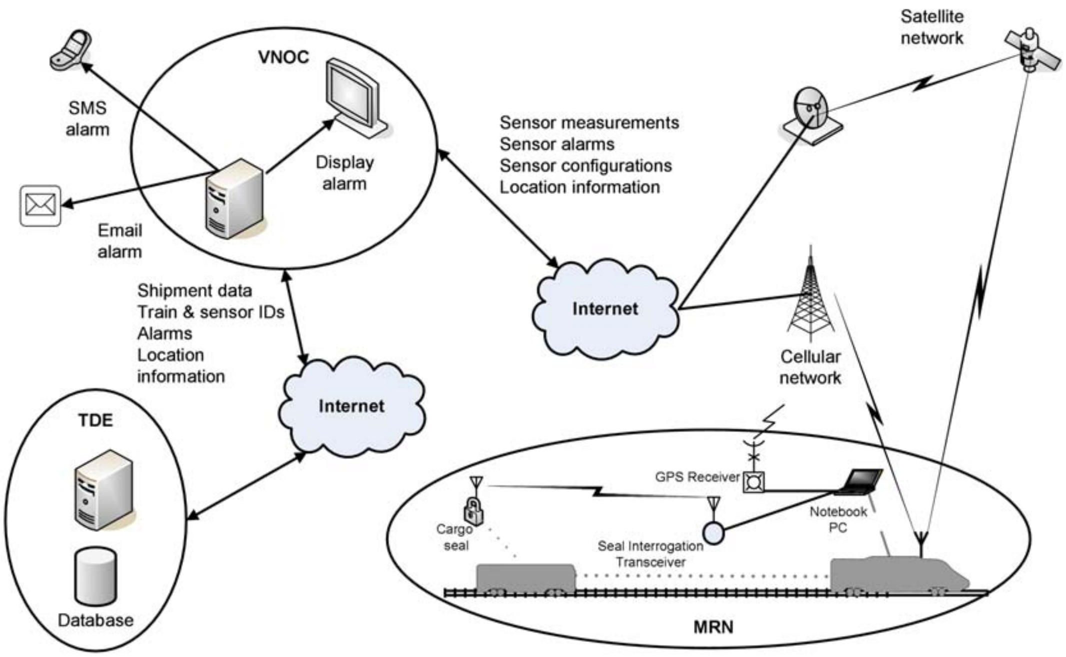

Fig. 1. Transportation Security Sensor Network (TSSN) Architecture

The TSSN uses web service specification standards—such as Web Services Description Language (WSDL 2.0) [11], Simple Object Access Protocol (SOAP 1.2) [12], WS-Addressing [13], WS-Security [14], and WS-Eventing [15]—which are implemented through Apache Axis2 [16] and associated modules. These standards are used to exchange structured information between a web service and client. The use of SOAP allows the deployment of platform-independent interfaces and thus a heterogeneous network of web service platforms. On the other hand, since SOAP and web services are based on XML, which is verbose, there is communication and processing overhead related to SOAP messages.

The TSSN supports wireless and satellite communication technologies such as HSDPA (High-Speed Downlink Packet Access) [17] and Iridium [18]. The TSSN uses the Hypertext Transfer Protocol (HTTP) for message transport over wired and wireless links. Finally, the TSSN prototype uses sensors and readers from Hi-G-Tek [19]. There is also a need to gather log files to enable system debugging as well as to capture metrics that can be used to evaluate system performance. Logging is currently done at the MRN, VNOC, and TDE using Apache log4j [20].

The TSSN system is composed of three major geographically distributed components: the Trade Data Exchange (TDE), Virtual Network Operations Center (VNOC), and Mobile Rail Network (MRN), as shown in Fig. 1. Wired links are used between the TDE and the VNOC, while MRN to VNOC communications are done using networks with commercial wireless link components. The TDE, VNOC, and MRN are examined in greater detail in the following subsections.

A. Trade Data Exchange

The Trade Data Exchange (TDE) contains shipping data and it interconnects commercial, regulatory and security stakeholders. The TDE is based on a “technology-neutral, standards-based, service-oriented architecture” [4]. The TDE is hosted on a server with a wired connection to the Internet. The TDE is geographically separated from the VNOC and responds to queries from the VNOC. The TDE also stores event messages sent by the VNOC. Finally, the TDE sends commands to start and stop monitoring at the MRN as well as to get the train’s current location.

In addition to the functions mentioned above, the TDE monitors the progress of shipment and other logistics information. The TDE captures commercial and clearance data including: the shipping list, bill of lading, commercial invoice, Certificate of Origin (for example, NAFTA Letter), and shipper’s export declaration. It also validates and verifies data to ensure accuracy, consistency, and completeness. The TDE monitors the progress of the documentation and notifies responsible parties when errors or incompleteness pose the threat of delaying a shipment. The TDE forwards notification to the customs broker to request verification of the trade origination documents. The customs broker accesses the TDE via the same portal to review and verify the trade documentation. Finally, the TDE allows for collaboration between participating shippers, third-party logistics providers, carriers and customs brokers to define and document business requirements and risk assessment requirements. Real-time cargo sensing capability is provided to the TDE via the TSSN. Data from the TDE is combined with event data from the MRN to provide the decision maker complete information concerning the alarm, e.g., cargo information, location, and nature of the event.

B. Virtual Network Operations Center

The Virtual Network Operations Center is the management facility of the TSSN [10] and it is also the shipper’s interface to the TDE. The VNOC can be the central decision and connection point for multiple MRNs. The VNOC consists of services that receive and process alarms from the MRN as well as services that notify decision makers of events. Fig. 2 summarizes the VNOC and its components.

The functions of the VNOC include: forwarding commands from a client to the MRN to start and stop sensor monitoring as well as to get the MRN’s current location, receiving *MRN_Alarms* from the MRN, obtaining event-associated cargo information from the Trade Data Exchange (TDE) in real time,

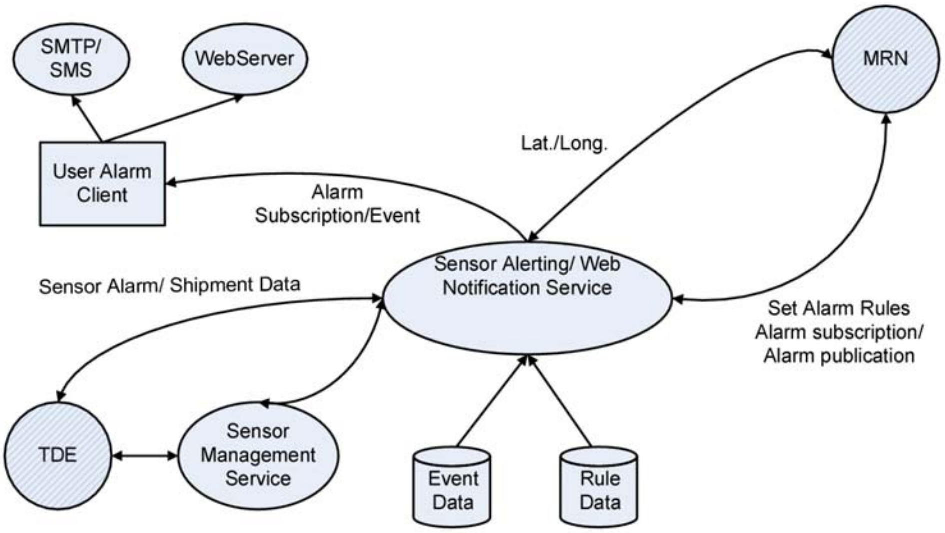

Fig. 2. Virtual Network Operations Center Architecture

and combining cargo information obtained from the TDE with an *MRN_Alarm* to form a *VNOC_Alarm* message that is sent (by SMS and/or e-mail to decision makers as shown in Figs. 9 and 10. A key role of the VNOC is getting the the right alarm information to the right personnel in a timely manner and also to prevent personnel from being overwhelmed by event messages. An AlarmProcessor service in the VNOC makes decisions, using rules, on which *MRN_Alarms* are forwarded to decision makers. For example, a low battery alarm is sent to technical staff, while an unexpected open/close event is sent to system security personnel. These decisions are made using a complex event processor, Esper [21], which takes into account shipping information as well as data (for example, geographical location) from current and past *MRN_Alarms*.

C. Mobile Rail Network

The MRN subsystem is located on the train and it consists of hardware and software. The prototype hardware and software architecture is described below.

1) Mobile Rail Network Hardware: The MRN subsystem hardware consists of a set of wireless shipping container security seals and a TSSN collector node. The collector node is composed of two major sections: an electronics suite mounted in the locomotive cab and a remote antenna assembly that is magnetically attached to the exterior of the locomotive. Fig. 3 summarizes the key components of the TSSN collector node.

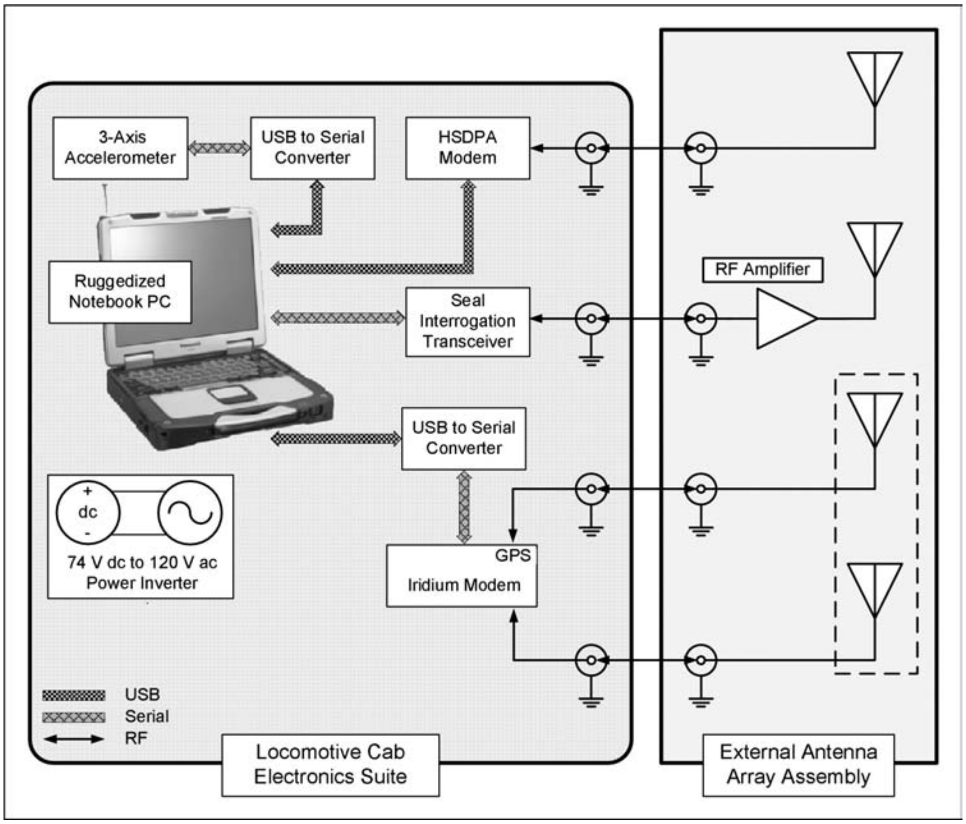

Fig. 3. TSSN Collector Node Hardware Configuration

The electronics suite contains a power inverter, a security seal interrogation transceiver, a computing platform, wireless data modems, a three-axis accelerometer, and a GPS receiver. The antenna assembly consists of three communications antennas, a GPS receiver antenna, and a bidirectional RF amplifier. Coaxial cables connect electronics suite devices to corresponding antennas.

Container physical security is monitored using a system that was originally designed for tanker truck security [19]. Container security is monitored with active and battery-powered container seals (sensors) equipped with flexible wire lanyards that are threaded through container keeper bar lock hasps as shown in Fig. 4. These seals had no support for multihop communications. The TSSN is designed to monitor and report security seal events including seal opened, seal closed, tampered seal, seal armed, seal missing, and low battery warnings. The seal interrogation transceiver (SIT) communicates with the container seals over a wireless network while the interrogation transceiver communicates with a notebook computer via a serial data connection. In order to conserve energy the container seals are asleep most of the time [22]. About every three seconds the seals listen for commands from the interrogation transceiver; however, the

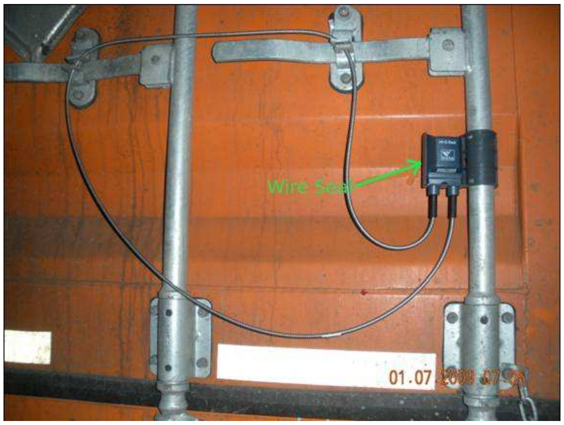

Fig. 4. Container Seal

frequency at which the seals listen for commands is configurable. If the sensors are instructed to listen for commands more frequently then their battery lifetimes are reduced, whereas longer intervals between interrogations result in longer battery lifetimes [22].

Communication between the MRN and the VNOC is accomplished using a HSDPA cellular data modem. An Iridium satellite modem is also available and is intended for use in remote locations that lack cellular network coverage. The Iridium modem is a combination unit that includes a GPS receiver, which is used to provide the MRN with position information.

2) Mobile Rail Network Software: The protoype MRN software was implemented using a service-oriented architecture approach. The software consists of a SensorNode service, an AlarmProcessor service, and a Communications service. The SensorNode service finds and monitors sensors that have been assigned to its control. The SensorNode service manages several sensor software plug-ins, for example, a seal interrogation transceiver plug-in and a GPS device plug-in, that do all the work on behalf of the SensorNode service. During typical operation each container seal listens for interrogation command signals at regular intervals from the interrogation transceiver. In the event of a seal being opened, closed, or tampered with, the seal immediately transmits a message to the SensorNode service running on the Collector Node. The message contains the seal event, a unique seal ID, and event time. The SensorNode service passes the seal message as an *Alert* message to the service that has subscribed for this information.

The AlarmProcessor service determines messages from the SensorNode service that require transmission to the VNOC. Alarm messages include the seal event, event time, seal ID, and train's GPS location. The Communications service uses either HSDPA or Iridium for reporting events via the Internet to the

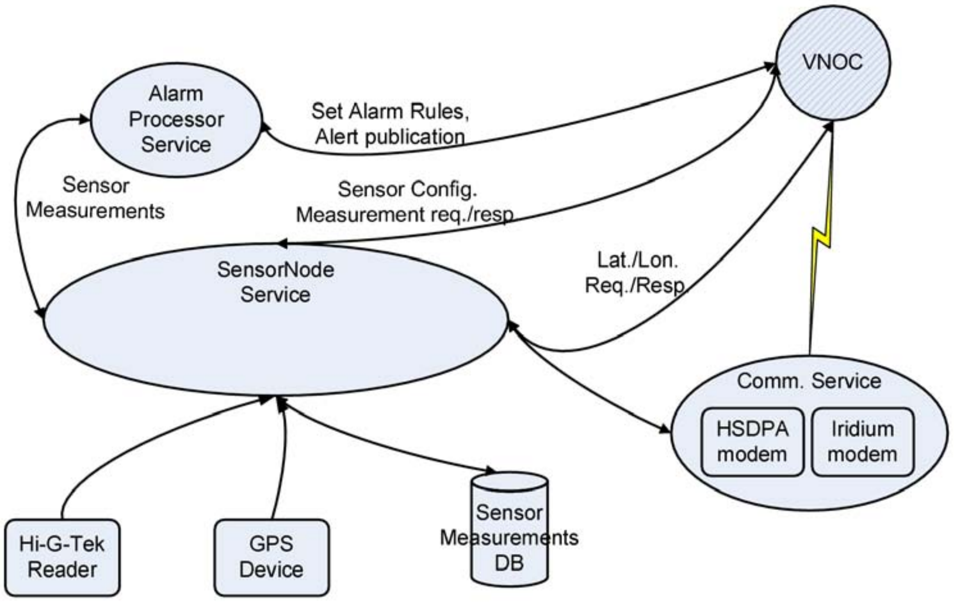

Fig. 5. Mobile Rail Network Collector Node Architecture

VNOC. Fig. 5 shows the key software functions of the MRN.

III. Experiments

This section presents two experiments—a road test and the short-haul rail trial— conducted to assess the suitability of the TSSN architecture for cargo monitoring as well as to collect data that would be used to guide the design of future cargo monitoring systems. It is non-trivial to carry out experiments on moving freight trains; furthermore, as part of this effort we were limited to one chance to carry out experiments from a train. As a result, the TSSN architecture was tested in several static and some mobile tests, including the road test with trucks and the short-haul rail trial. In this section we present the experimental objectives, configuration, data collected during the tests, and issues that were encountered during the tests. The overarching goals of these experiments were to:

- Demonstrate the concept of using sensors, communications, and a service oriented architecture to monitor cargo in motion using the TSSN architecture.
- Determine the time from event occurrence to decision maker notification in a real field experiment.
- Verify proper operation of the prototype TSSN in a field environment. Proper operation means all messages were transmitted, received, and processed as expected and decision makers received all correct notification.

Thus, the following items were within scope of our experiments: the stability and timely performance of the communications protocols between TSSN component services, whereas the following items were out of scope: overall system robustness, whole train monitoring, energy consumption of the sensors, comprehensive security[1] issues, such as message spoofing, and decision maker response time given that event notification had been delivered.

A. Road Test with Trucks

The first experiment was conducted with two pickup trucks on local roads to validate the system operation and to determine if correct information is reported by the TSSN collector node, including valid GPS coordinates. One of the pickup trucks used in the test had the locomotive cab electronics suite in the truck bed, while both trucks had seals in their truck cabins so that seal open and close events could be emulated and reported. The VNOC was located in Lawrence, Kansas while the TDE was located in Overland Park, Kansas. The trucks were driven for approximately 1.5 hours over a 90 km route that began and ended in Lawrence, Kansas. The experiment route covered suburban and rural roads as well as state highways. During the experiment the seals were opened and closed at selected intersections along the test route that were easily identifiable on Google Maps [23].

Fig. 6 shows a trace of our route and the events overlaid on a Google map. The diamonds indicate an open event, tear drops a close event, circles with slashes across indicate a GPS lost signal, tacks indicate where the GPS signal was regained, a triangle with an exclamation sign indicates where HSDPA connectivity was lost, and the arrow indicates where HSDPA connectivity was regained.

B. Short-haul Rail Trial

Another experiment was carried out using a freight train travelling from an intermodal facility to a rail yard. Our objectives in this experiment were the following:

- To determine the performance of the prototype TSSN architecture when detecting events on intermodal containers in a real rail environment.
- To investigate if decision makers could be informed of events in a timely manner using SMS messages and e-mails.
- To collect data that will be used in a model to investigate system trade-offs and the design of communications systems and networks for monitoring rail-borne cargo.
- Evaluate the overall system performance to guide the future development of the TSSN architecture.

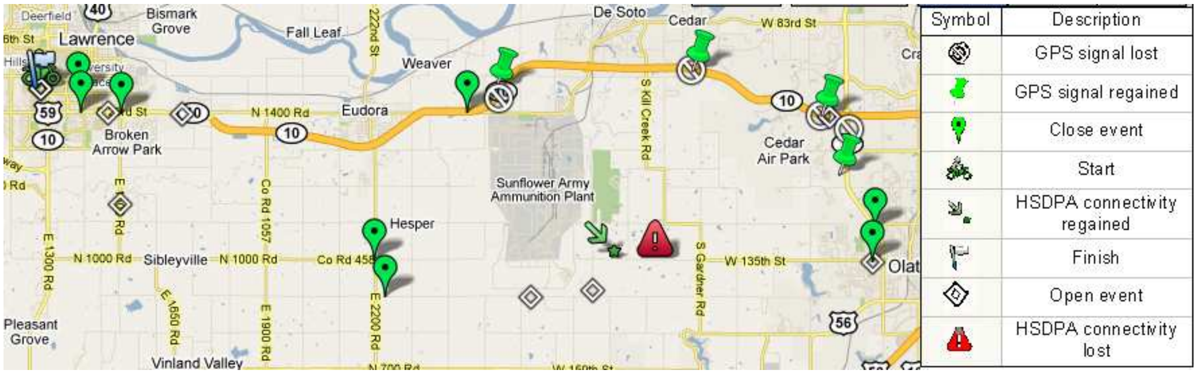

Fig. 6. Map of Road Test with Event Annotations

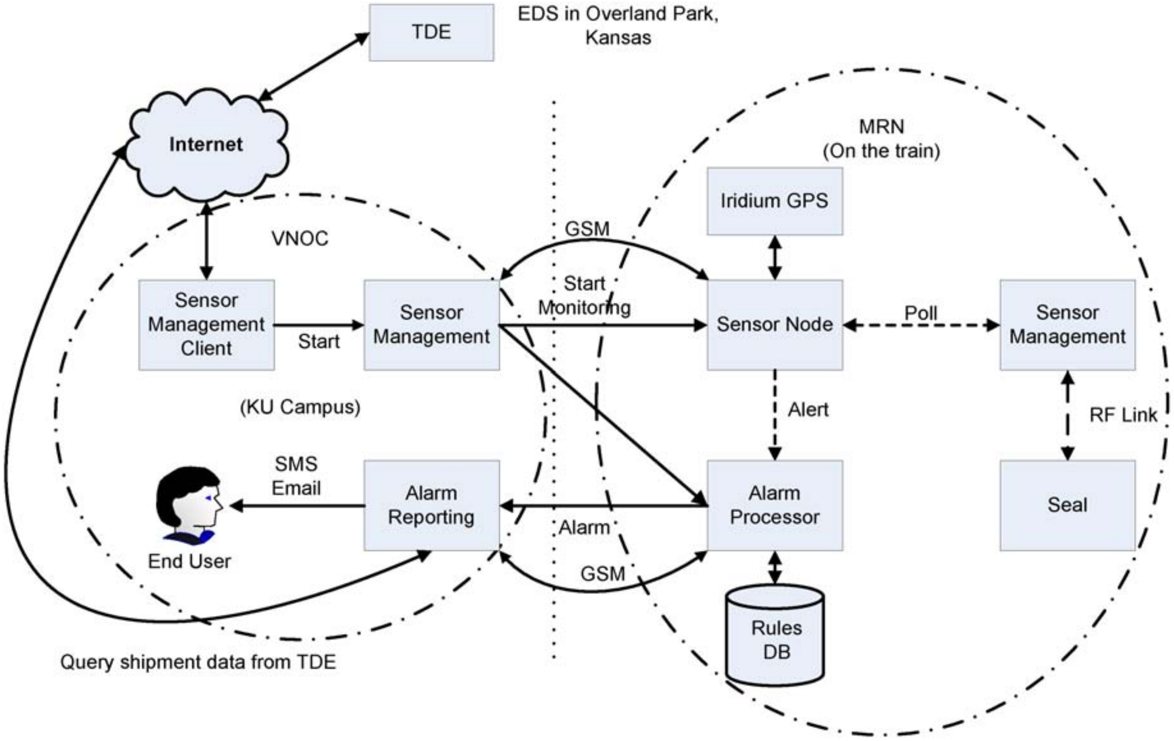

Fig. 7. Logical Short-haul Rail Trial Configuration

Fig. 7 shows the logical system configuration used in the short-haul rail trial. In this experiment the VNOC was located in Lawrence, Kansas, the TDE was located in Overland Park, Kansas, and the MRN was placed on the train. Within the MRN, the TSSN collector node was placed in a locomotive and

[1] Comprehensive security issues are being addressed in the next version of the prototype.

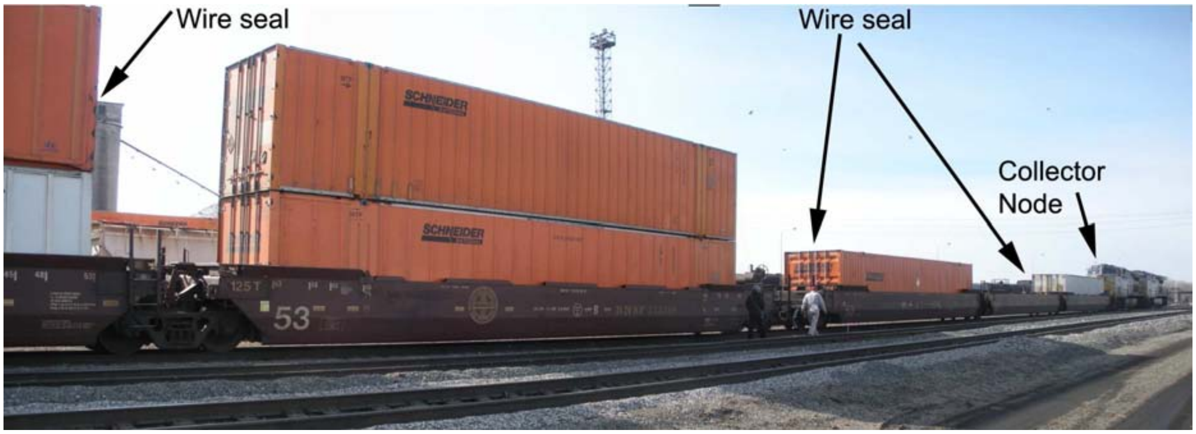

Fig. 8. Collector Node and Sensor Deployment During Short-haul Rail Trial

used to monitor five seals. All communications between the MRN and the VNOC were passed through a Virtual Private Network (VPN) for message security. Prior to the start of the experiment prototype logistics data was added to the TDE to facilitate testing.

During the short-haul trial the train traveled for approximately five hours over a 35 km (22 miles) route. The route, which traversed both rural and urban areas, was relatively flat with a total elevation change of about 100 m. Fig. 8 shows a picture of the train used in the short-haul rail trial along with the arrangement of the sensors (wire seals). As shown in Fig. 8, the short-haul trial train was composed of well-cars with a mixture of empty cars, cars with a single container, or cars with double-stacked containers. Since we were demonstrating a proof of concept and the sensors in use for this test were commercial-off-the-shelf devices with no support for multihop communications, three sensors were placed on containers on three of the five railcars nearest to the locomotive so that they could be within radio range of the seal interrogation transceiver. One sensor was placed on the front of the locomotive while the fifth sensor was kept in the locomotive and manually opened and closed while the train was in motion to create events.

During the experiment the VNOC reported events to decision makers using e-mail and SMS messages. The e-mail messages also include a link to Google Maps, so that the exact location of the incident could be visualized. Fig. 9 shows the content of one of the e-mail messages that was sent to the decision makers and Fig. 10 presents an example of an SMS message.

In Figs. 9 and 10, the sensor ID, latitude and longitude data, and event type come from the MRN, while the shipment data comes from the TDE. The VNOC combines these pieces of information into an

```
NOC_AlarmReportingService:
 Date-Time: 2009.01.07 07:12:17 CST /
    2009.01.07 13:12:17 UTC
 Lat/Lon: 38.83858/-94.56186,
          Quality: Good
 http://maps.google.com/maps?q=38.83858,-94.56186
 TrainId=ShrtHaull
 Severity: Security
 Type: SensorLimitReached
 Message: SensorType=Seal
           SensorID=IAHA01054190
           Event=Open Msg=
 NOC Host: laredo.ittc.ku.edu

Shipment Data:
 Car Pos: 3
 Equipment Id: EDS 10970
 BIC Code: ITTC054190
 STCC: 2643137
```

Fig. 9. E-mail Message Received During Short-haul Trial

```
NOC_Alarm:
Time:2009.01.07 07:12:49 CST
GPS:38.83860/-94.56186
Trn:ShrtHaull
Sev:Security
Type:SensorLimitReached
Msg:Sensortype=Seal SensorID=IAHA01054190
    Event=Close
```

Fig. 10. SMS Message Received During Short-haul Trial

e-mail message that also includes a link to Google Maps, so that the exact location of the incident can be visualized.

During the test the interrogation transceiver lost communication with the seals for a brief period along the route while the train was stationary and then regained communications once the train started moving.

We believe that this loss of communications was due to electromagnetic interference. However, further investigations are needed to validate this claim.

The short-haul rail trial was a success as all seal events were detected and reported to decision makers using both e-mail and SMS messages. Extensive log files were collected during the test and they were postprocessed to obtain data on TSSN system performance. The results from postprocessing, which are reviewed in Section V, show that the prototype system functioned as expected.

Following this experiment, analysis of event logs obtained from the MRN, VNOC, and TDE revealed that there was a significant amount of clock drift on the TSSN Collector Node during this relatively short trial. The time recorded at the VNOC for receipt of a message, in some cases, was earlier than the time recorded at the TSSN Collector Node for sending the message. Since time at the VNOC is controlled by a Network Time Protocol (NTP) [24] server, we conclude that the clock drift is occurring on the TSSN Collector Node. The clock drift problem was resolved in the next version of the TSSN by using a high performance GPS receiver to get high quality local time. Pulse per second (PPS) output from the GPS receiver was used as an input to the NTP server running on the TSSN collector node. It should be noted that in spite of the clock drift in the TSSN collector node we were able to correct for it in our data analysis by assuming that data from different parts of the TSSN is independent, e.g., the time taken to break a seal and generate an alert message is independent of the time taken to transfer a message from the MRN to the VNOC. As a result we can measure elapsed time in different epochs separately and characterize performance of the TSSN prototype.

IV. Postprocessing of Experimental Data

In this section we discuss the framework for postprocessing the results of our experiments. During the short-haul rail trial we recorded events in log files at the geographically distributed VNOC, MRN, and TDE. These log files contained data on message sizes, timestamps, event type, message type (incoming/outgoing) amongst other data elements. Our objective was to postprocess these files to evaluate the performance of the prototype TSSN.

Postprocessing of log files was accomplished using a Java library (LogParser) that was developed in-house. First, the library read in all available information in each log file including time, message size, from and to addresses, as well as the original SOAP message. Information from the MRN, VNOC, and TDE log files in this experiment was combined into a single collection of log entries. We expect that every message transmitted in the TSSN should result in at least two log entries—a transmit log entry (at the originating entity) and a received log entry (at the receiving entity). The LogParser library identified

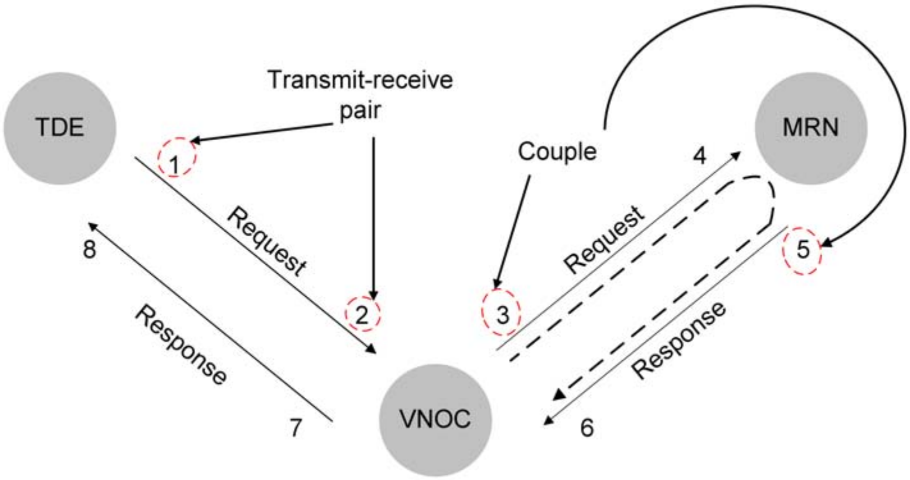

Fig. 11. LogParser Framework Showing Message Couples and Transmit-receive Pairs

log entries as:

- Transmit-receive pairs, that is, the outgoing and incoming log entries with the same SOAP WS-Addressing [13].
- Couples, that is, SOAP request-response message pairs.

Fig. 11 shows the relationship between log entry couples and transmit-receive pairs. Suppose the TDE sends a message to the VNOC requesting the current MRN location. The circled “1” and “2” in Fig. 11 denote the log entries representing message transmission from the TDE and receipt of this same message at the VNOC. Much of the communication between client/server is based on a request-response model. As a result, there are two related messages which contain additional information to establish their relationship:

1) REQUEST: from client to server asking for something; and
2) RESPONSE: from server back to the client with the response.

Log entry couples are marked by the records for the outgoing request and response messages. Consequently, the circled “3” and “5” in Fig. 11 constitute the log entry couple for the VNOC forwarding the location request message to the MRN and the MRN’s origination of a response respectively. Using the receive pairs for records “3” and “5”, we can also identify entries “4” and “6.”

With this framework, programs were written against the log entry collection to extract the number of messages sent by each service, request-response time for messages, processing time at either the MRN, VNOC, or TDE, the time that messages were carried by the network, and message sizes. Additional information, such as, latitude, longitude, sensor IDs, and event timestamps, is extracted from the SOAP message using XPath expressions. XML Path language (XPath) is used for extracting information from XML by using path expressions that traverse the XML tree. Since SOAP is XML and the elements that we use, e.g., *Alerts*, *MRN_Alarms*, and *VNOC_Alarms*, are also XML, the use of XPath is appropriate. XPath also provides "basic facilities for manipulation of strings, numbers and booleans" [25].

V. Results

This section discusses the results of the experiments presented in Section III. Most of the results shown here are based on the short-haul rail trial because we had more data to analyze. The results presented here are selected to test claims that:

- All messages between component services of the TSSN were transmitted, received, and processed as expected.
- Decision makers can be notified of events on the train in a timely manner.

The rest of this section is laid out as follows: Sections V-A and V-B present results on message counts for the road test and short-haul rail trial respectively. These results test the claim that all messages between component services of the TSSN are transmitted, received, and processed as expected. The rest of the results are based on the short-haul rail trial. Sections V-D–V-E study different portions of the time from event occurrence to decision maker notification to verify the claim that the TSSN can notify decision makers of events in a timely manner. Probability distributions are used in Section V-F to determine the likelihood of timely decision maker notification.

Note that due to significant clock drift in the TSSN collector node, we can only present an estimate of the time taken for an event report to travel from the MRN to the VNOC. However, observed time values can be directly used for other TSSN component interactions. These results show how the aggregate time from event detection to decision maker notification is distributed among the various services and communication links in the TSSN. With this information we will be able to guide system refinements to further reduce the overall time. Suppose that T_n indicates when log entry n is made, then we can compute the following metrics:

- **Service request processing time.** This is the time between when a service receives a request and when a response message is composed. Using Fig. 11, this time is: $T_5 - T_4$.
- **Request-response time.** This is the time taken to get a response from a remote service, including the processing time. Using Fig. 11, this time is: $T_6 - T_3$.

- **Network time.** This is the time taken to get a response from a remote service, excluding the processing time. Using Fig. 11, this time is computed as: $T_6 - T_3 - (T_5 - T_4)$.

Our time analysis in Section V-G examines request-response messages from VNOC → MRN → VNOC, TDE → VNOC → TDE, and VNOC → TDE → VNOC.

The last objective of the short-haul rail trial was to collect data that will be used in a model [26] to support the future design of systems for monitoring rail-borne cargo and determine trade-offs. Message sizes are one component of this model. As a result, Section V-H presents a table summarizing the message size statistics between different components of the TSSN. It should be noted that message sizes can be computed *a priori*; however, the distribution of these messages cannot be determined beforehand.

A. Road Test: Message Counts

The primary goal of the road test was to validate TSSN prototype operation and to determine if correct information is reported by the TSSN collector node, including valid GPS coordinates. During the road test a manual record was made of all seal events and this written record was compared with the information generated from the TSSN. This comparison revealed that all open and close events were propagated correctly. During the approximately 1.5 hours long road test 76 messages (72 *Alarms*, 2 *StartMonitorSensors*, and 2 *StopMonitorSensors* commands) were exchanged on the VNOC to MRN link and these messages corresponded with the events that were recorded in the experiment log. Based on analysis of these messages we conclude that the system operated as expected. In addition, the experiment revealed that the TSSN was able to recover from a dropped HSDPA connection. However, it is worth noting that the seal interrogation transceiver was unable to read the sensors when the trucks were over 400 m apart on a hilly road. Based on the road test we conclude that the TSSN prototype worked as expected in a mobile scenario; we were able to combine sensor data from the MRN in a moving vehicle with shipment information obtained from the TDE to generate e-mail messages that were sent to distributed decision makers. Results from the road test showed that the TSSN prototype was ready for evaluation in a real rail environment.

B. Short-haul Trial: Message Counts

One objective of our postprocessing was to determine if messages were being passed correctly between the TSSN components. During the short-haul trial 203 messages (2 *StartMonitorSensors*, 2 *StopMonitorSensors*, 4 *SensorNodeStatus*, and 4 *SetMonitoringState* commands, 30 *getLocation* queries, 30 *Location* responses, and 131 *MRN_Alarms*.) were passed over the VNOC to MRN link. Full details on the messages

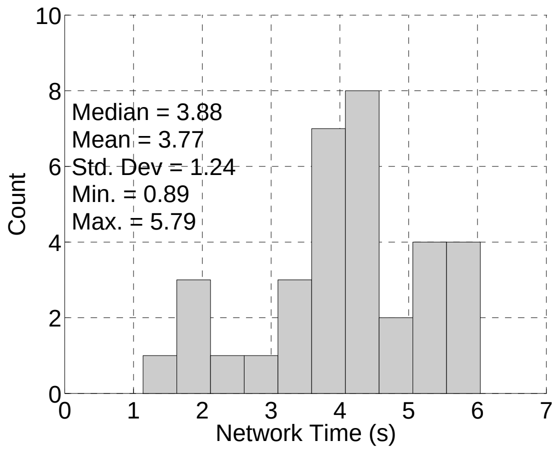

Fig. 12. Network Times from VNOC → MRN → VNOC

exchanged are found in [27]. All of the *MRN_Alarms* received by the VNOC AlarmProcessor met the necessary rules so that they could be forwarded to decision makers as SMS and/or e-mail messages. The test users who were designated to receive all event notifications from the TSSN received 131 e-mail messages each.

C. Network Time from VNOC to MRN to VNOC

The network time statistics from VNOC to MRN to VNOC allow us to draw conclusions on the time taken to transfer request and response messages from the VNOC to the MRN and *vice versa*. These statistics also allow us to gain insight into the one-way network delay from the TSSN collector node on the train to the VNOC in Lawrence, Kansas—a delay that is one component of sending an *MRN_Alarm* message—which indicates an event at a sensor—from the MRN to the VNOC. Due to clock drift in the TSSN collector node, we are unable to obtain statistics on the one-way network delay from MRN → VNOC. However, it is reasonable to assume that the MRN ↔ VNOC links are symmetric thus, the average one-way delay from the MRN to the VNOC is approximately 1.89 s.

D. Elapsed Time from Alert Generation to AlarmReporting Service

Demonstrating that the elapsed time from alert generation to the AlarmReporting service is of the order of several seconds can help establish the value of the TSSN. Fig. 13 shows the messages involved in notifying a decision maker of an event at a seal. This subsection deals with epochs 2, 3, and 4. Exact values can be computed for the time taken to propagate *Alert* and *VNOC_Alarm* messages, while we can

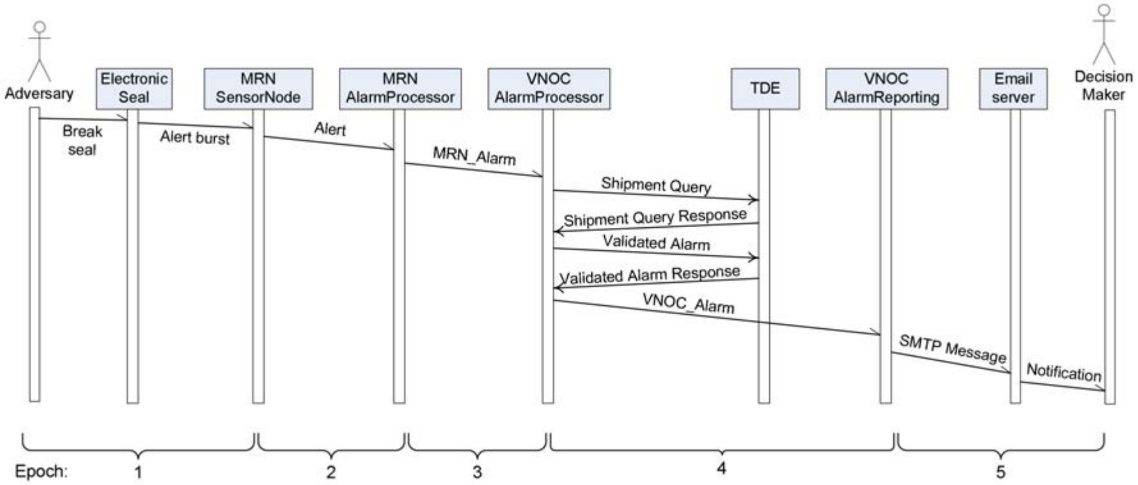

Fig. 13. Sequence Diagram with Messages Involved in Decision Maker Notification

TABLE I

SUMMARY OF TIME STATISTICS FOR DECISION MAKER NOTIFICATION

Epoch	Description	Min./s	Max./s	Mean/s	Median/s	Std. Dev./s
1	Event occurrence to *Alert* generation	0.81	8.75	2.70	2.13	1.86
2	*Alert* generation to MRN AlarmProcessor service	0.01	0.08	0.02	0.01	0.01
3	One-way delay from MRN AlarmProcessor to VNOC AlarmProcessor	0.45	2.90	1.89	1.94	0.62
4	*MRN_Alarm* arrival at VNOC AlarmProcessor to AlarmReporting service	0.01	3.01	0.17	0.05	0.32
5	Elapsed time from VNOC AlarmReporting service to mobile phone	5.2	58.7	11.9	9.8	7.4

use the 1.89 s estimate from the previous subsection as a reasonable value for the time taken to transfer a *MRN_Alarm* message from the MRN to the VNOC.

By analyzing the log files we see that on average it takes about 2 s for messages to get from the MRN SensorNode service to the VNOC AlarmReporting service. Thus, we conclude that the time taken to process events in the TSSN is not an impediment to timely notification of decision makers.

E. End-to-end Time from Event Occurrence to Decision Maker Notification

The primary performance metric for prototype TSSN performance is the time between event occurrence until a decision maker is notified using an SMS message. The components of the end-to-end time include epochs 1–5 in Fig. 13.

To obtain an understanding of the end-to-end system time as well as overcome any clock errors in the MRN subsystem, we set up a laboratory experiment to determine the elapsed time between event occurrence and the TSSN's generation of the related event alert. In this experiment, a stopwatch was started when a seal was either broken or closed; when the MRN SensorNode service generated an *Alert* message the stopwatch was stopped. From Table I we see that the longest observed time in epoch 1 is about 8.8 s, while the mean is about 2.7 s.

Since the commercial wireless networks used for decision maker notification are outside TSSN control, a second laboratory experiment was carried out to determine the elapsed time in epoch 5. In this experiment a client program was written to send messages to the VNOC AlarmReporting service. A stopwatch was started when the VNOC sent an alarm to a decision maker and the stopwatch was stopped when the decision maker's phone received an SMS message. This experiment was repeated for four different carriers resulting in the data shown in row 5 of Table I.

From Table I we see that even though SMS was not designed as a real-time system, it provides excellent notification for this application, since most of our messages were delivered within one minute. Combining all of these results, we see that in these experiments the longest observed end-to-end system time was just over one minute[2] to notify decision makers of events. Most of this time is spent delivering an SMS message to the decision maker, so we conclude that the TSSN provides a mechanism for timely notification of decision makers.

F. Modeling of Decision Maker Notification Time

In this section we determine the likelihood of timely event notification. To determine the likelihood of timely event notification a probabilistic model is needed for the time epochs shown in Fig. 13. The observed histograms for each epoch visually resembled a Gamma distribution. Thus, in this analysis we assume the times in each epoch followed a Gamma probability density function. While the number of observations (less than 130) was insufficient to statistically validate this assumption this postulate allows us to probabistically determine if the TSSN prototype can provide event notification within 15 minutes [5] as required. The parameters for the distributions are estimated from the collected data and shown in Table II, where $\hat{\alpha}$ and $\hat{\theta}$ represent the shape and scale parameters of the associated Gamma random

[2] This time is broken out as follows: in the longest observed times in our experiments it took approximately 8.8 s between event occurrence and the TSSN generating an alert; 2) it took approximately 4.91 s for an alert message to go through the TSSN until notification was sent to decision makers; and 3) it took up to 58.7 s to deliver an SMS message to decision makers.

TABLE II

ESTIMATED GAMMA DISTRIBUTION PARAMETERS FOR TIME TAKEN BETWEEN SEAL EVENTS AND DECISION MAKER NOTIFICATION

Epoch	Symbol	Description	$\hat{\alpha}$	$\hat{\theta}$
1	E_1	Event occurrence to *Alert* generation	4.01	0.60
2 + 4	$E_{2,4}$	*Alert* generation to MRN AlarmProcessor and *MRN_Alarm* arrival at VNOC AlarmProcessor to AlarmReporting service	1.13	0.13
3	E_3	One-way delay from MRN AlarmProcessor to VNOC AlarmProcessor	13.95	0.14
5	E_5	Elapsed time from VNOC AlarmReporting service to mobile phone	10.44	1.00

variable. Let τ, which is composed of each of the epochs presented in Sections V-C–V-E, represent the total time taken from event occurrence on the train to decision maker notification on a mobile phone. Then $\tau = E_1 + E_{2,4} + E_3 + E_5$ and we use the results from [28] to show that $\Pr\left[\tau \leq 240 \text{ sec}\right] = 99.9$ %. These results indicate that the prototype TSSN can notify decision makers in a timely manner with very high probability.

G. Timing Analysis of Other TSSN Interactions

Table III summarizes request/response, processing, and network time statistics for interaction between various TSSN components. The statistics on VNOC → MRN → VNOC interaction allow us to draw conclusions on request-response and processing times for certain (Start or stop monitoring at the MRN and get current MRN location.) VNOC commands. TDE → VNOC → TDE interaction statistics give us insight into the time taken to initiate and process commands to start or stop monitoring at the MRN, get the MRN's current location, or to process the setAlarmSecure command. The VNOC forwards these commands to the MRN and returns the MRN response to the TDE. To the TDE, all the elapsed time from when the VNOC receives a message from the TDE until the VNOC sends a response is processing time at the VNOC, even though part of that time is spent forwarding a response to the MRN and waiting for a response. Finally, the statistics on VNOC → TDE → VNOC interactions allow us to draw conclusions on request-response, processing, and network times for the TDE to store alarm messages and execute shipment queries. Both of these actions are carried out when the VNOC AlarmProcessor service is about to send an alarm to the VNOC AlarmReporting service. Note that there are no results for the MRN to VNOC to MRN interaction. This is for two reasons: first, clock drift in the MRN prevents us from

TABLE III

SUMMARY OF TIME STATISTICS FOR OTHER TSSN INTERACTIONS

Description	Min./s	Max./s	Mean/s	Median/s	Std. Dev./s
Request-response times from VNOC → MRN → VNOC	0.90	10.96	4.39	3.95	2.40
Network times from VNOC → MRN → VNOC	0.89	5.79	3.77	3.88	1.24
Processing times from VNOC → MRN → VNOC	0.00	5.21	0.61	0.01	1.69
Request-response times from TDE → VNOC → TDE	0.34	11.03	4.29	3.94	2.51
Network times from TDE → VNOC → TDE	0.00	4.00	0.14	0.04	0.64
Processing times from TDE → VNOC → TDE	0.29	10.98	4.15	3.85	2.45
Request-response times from VNOC → TDE → VNOC	0.02	0.41	0.12	0.07	0.11
Network times from VNOC → TDE → VNOC	0.01	0.08	0.05	0.07	0.02
Processing times from VNOC → TDE → VNOC	0.01	0.38	0.07	0.01	0.10

TABLE IV

SUMMARY OF MESSAGE SIZE STATISTICS

Description	Min./bytes	Max./bytes	Mean/bytes	Median/bytes	Std. Dev./bytes
TDE → VNOC	846	1278	874.7	848	96.8
VNOC → TDE	968	975	971.5	971	2.6
VNOC → MRN	650	1036	690.8	650	101.5
MRN → VNOC	799	1560	1419.2	1536	237.1

computing a one-way network delay. Secondly, the MRN only generates response messages. As expected there are no request messages originating from the MRN that could be used in a log entry couple to calculate request-response or processing times.

H. Message Sizes

Table IV summarizes the message size statistics for all the messages exchanged in the TSSN. Message size data are needed for a model [26] that is under development to determine system trade-offs as well as optimal or near-optimal sensor locations when using a rail-borne cargo monitoring system. The cost of transmitting a message from the train to an operations center is one component of this model. This transmission cost, in turn, depends on the average message length transmitted from the train and the frequency at which these messages are generated.

VI. Refinements Based on Experimental Results

This section proposes refinements to the TSSN based on experimental results. Recall from Section III-B that we have corrected the clock drift problem by using a high performance GPS receiver to get high quality local time on the TSSN collector node. In addition, postprocessing of the log files also indicated that a unique identifier—perhaps composed of a timestamp and counter—is needed in the *Alert*, *MRN_Alarm*, and *NOC_Alarm* messages to trace an *Alert* message through the TSSN. This identifier can also be used in the future to locate *MRN_Alarm* messages that need to be retransmitted to the VNOC following a loss of connectivity. Finally, the identifier can be used to mark previously processed messages so that the VNOC does not process the same message more than once.

Additional TSSN enhancements include:

- Redesigning the MRN hardware so that the TSSN collector node has redundant backhaul communication capabilities, for example, multiple satellite and cellular modems each with a different provider.
- Creating a comprehensive security framework for the TSSN. Ongoing research is addressing this issue [29].
- Enhancing sensor capabilities so that sensors can engage in multihop communications to enable whole-train monitoring.

The desired result of the research presented here is a standards-based open environment for cargo monitoring with low entry barriers to enable broader access by stakeholders while showing a path to commercialization.

VII. Related Work

In this section we provide a brief overview of related research to monitor trains and to secure cargo in motion. In 2005 Edwards *et al.* [30] presented a prototype system to monitor and control various sensors and actuators on a freight train. The prototype uses a Controller Area Network (CAN) bus to collect data from the sensors. The data is then coupled with GPS information and reported to a web server via a CDMA-based transmitter. Edwards *et al.* [30] argue that "on board sensing of mechanical defects enables car owners to track defects and proactively schedule maintenance" at a time and location that makes economic sense.

The Transf-ID system [31], proposed in 2009, uses radio frequency identification (RFID) tags and a service-oriented architecture to track cargo, railcars, and frequently serviced parts. The authors of [31] argue that use of the Transf-ID system improves rail freight safety since part maintenance schedules are now based on actual use.

In 2007 Lauf and Sauff [32] proposed a security protocol for transmitting information from sensors within a shipping container to a trusted third party. Such a protocol permits tracing liability for cargo theft and/or damage while minimizing the risk that shipping containers can be used for terrorism or shipment of contraband. The protocol was deployed successfully to test hardware; however, additional research is needed to create tamper-resistant units for monitoring container security [32]. Also in 2007, Ruiz-Garcia *et al.* [33] argued that the technology already exists to develop a monitoring system for refrigerated containers. They added that sensor readings and GPS information can be combined to track a shipping container through different stages of the supply chain.

The review of related work presented in this section shows that others have monitored train equipment using a service-oriented architecture as well as developed security protocols for communicating with sensors inside shipping containers. To the best of our knowledge, the TSSN is the first effort that uses sensors and an open service-oriented architecture to monitor freight in motion.

VIII. Conclusion

In this paper we have presented results from field trials of the prototype TSSN (Transportation Security Sensor Network). The TSSN is an open system where different vendors can supply different components of the system. Within the TSSN framework we have successfully combined sensor and shipment information to provide event notification to distributed decision makers. This paper has shown results documenting the interactions between the different components of the TSSN. Based on our experiments and evaluations with the prototype the TSSN architecture is viable for monitoring rail-borne cargo. We have successfully demonstrated that alert messages can be sent from a moving train to the VNOC and combined with cargo information that is forwarded to geographically distributed decision makers using either SMS or e-mail. Furthermore, based on the experiments reported here, we are able to detect events and notify decision makers in just over one minute. Thus, we conclude that the TSSN architecture provides a mechanism for timely notification of decision makers. However, additional development and testing is needed before the TSSN architecture can be deployed in production systems.

Acknowledgments

The authors would like to thank A. Francis reading and commenting on previous versions of this paper. The authors wish to acknowledge Kansas City Southern Railway for their participation in the short-haul rail trial. We would also like to acknowledge the support of EDS, an HP company, and Kansas City SmartPort, Inc. our partners on this project. Finally, L. Sackman of EDS, an HP company, assisted with the short-haul rail trial.

References

[1] Federal Bureau of Investigation. (2006, July 21) Cargo Theft's High Cost. Headline. Federal Bureau of Investigation. [Online]. Available: http://www.fbi.gov/page2/july06/cargo_theft072106.htm

[2] European Conference of Ministers of Transport, *Container Transport Security Across Modes*. Paris, France: Organisation for Economic Co-operation and Development, 2005.

[3] OASIS. (2006, Oct 12) Reference Model for Service Oriented Architecture 1.0. OASIS Standard. [Online]. Available: http://docs.oasis-open.org/soa-rm/v1.0/soa-rm.pdf

[4] KC SmartPort. (2008, Nov 10) Trade Data Exchange—Nothing short of a logistics revolution. Digital magazine. [Online]. Available: http://www.joc-digital.com/joc/20081110/?pg=29

[5] Kansas City Southern Railroad, Private communication, 2007.

[6] A. Arsanjani *et al.*, "Web Services: Promises and Compromises," *Queue, ACM*, vol. 1, no. 1, pp. 48–58, Mar 2003.

[7] H. Saiedian and S. Mulkey, "Performance Evaluation of Eventing Web Services in Real-time Applications," *Communications Magazine, IEEE*, vol. 46, no. 3, pp. 106–111, Mar 2008.

[8] J. Brown *et al.*, "SMS: The Short Message Service," *Computer*, vol. 40, no. 12, pp. 106–110, Dec. 2007.

[9] D. T. Fokum *et al.*, "Experiences from a Transportation Security Sensor Network Field Trial," in *Proc. 3rd IEEE Workshop Enabling the Future Service-Oriented Internet: Towards Socially-Aware Networks (EFSOI 2009)*. Honolulu, HI: IEEE, Dec. 2009, pp. 1–6.

[10] M. Kuehnhausen, "Service Oriented Architecture for Monitoring Cargo in Motion Along Trusted Corridors," Master's thesis, University of Kansas, July 2009.

[11] R. Chinnici *et al.* (2007, June) Web Services Description Language (WSDL) Version 2.0 Part 1: Core Language. W3C Recommendation. W3C. [Online]. Available: http://www.w3.org/TR/wsdl20/

[12] M. Gudgin *et al.* (2007, Apr) SOAP Version 1.2 Part 1: Messaging Framework (Second Edition). Member submission. W3C. [Online]. Available: http://www.w3.org/TR/soap12-part1/

[13] D. Box *et al.* (2004, Aug 10) Web Services Addressing (WS-Addressing). Member submission. W3C. [Online]. Available: http://www.w3.org/Submission/ws-addressing/

[14] OASIS. (2004, March) Web Services Security: SOAP Message Security 1.0. OASIS Standard. OASIS. [Online]. Available: http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-soap-message-security-1.0.pdf

[15] D. Box *et al.* (2006, Mar) Web Services Eventing (WS-Eventing). Member Submission. W3C. [Online]. Available: http://www.w3.org/Submission/WS-Eventing/

[16] The Apache Software Foundation. (2008, Aug 24) Apache Axis2. Project documentation. The Apache Software Foundation. [Online]. Available: http://ws.apache.org/axis2/

[17] D. Mulvey, "HSPA," *Communications Engineer*, vol. 5, no. 1, pp. 38–41, February-March 2007.

[18] C. E. Fossa *et al.*, "An overview of the IRIDIUM (R) low Earth orbit (LEO) satellite system," in *Proc. IEEE 1998 National Aerospace and Electronics Conference, (NAECON 1998)*, Dayton, OH, USA, Jul 1998, pp. 152–159.

[19] Hi-G-Tek. (2009, Mar 17) Hi-G-Tek—Company. Corporate website. Hi-G-Tek. [Online]. Available: http://www.higtek.com/

[20] The Apache Software Foundation. (2007, Sep 1) Apache log4j. Project documentation. The Apache Software Foundation. [Online]. Available: http://logging.apache.org/log4j/

[21] EsperTech. (2009, Feb 11) Esper – Complex Event Processing. Project documentation. EsperTech. [Online]. Available: http://esper.codehaus.org/

[22] *DataReader and DataSeal : User's Manual*, UM4710, Hi-G-Tek Ltd., 2001, ver. A5.

[23] Google. (2009, May 6) Google Maps. Web mapping service. [Online]. Available: http://maps.google.com

[24] D. L. Mills, "Internet Time Synchronization: the Network Time Protocol," *Communications, IEEE Transactions on*, vol. 39, no. 10, pp. 1482–1493, Oct 1991.

[25] J. Clark and S. DeRose. (1999, Nov 16) XML Path Language (XPath). W3C Recommendation. W3C. [Online]. Available: http://www.w3.org/TR/xpath

[26] D. T. Fokum, "Optimal Communications Systems and Network Design for Cargo Monitoring," To appear in Proc. Tenth Workshop Mobile Computing Systems and Applications (HOTMOBILE 2009). Santa Cruz, CA: ACM Press, Feb 2009.

[27] D. T. Fokum *et al.*, "Experiences from a Transportation Security Sensor Network Field Trial," University of Kansas, Lawrence, KS, ITTC Tech. Rep. ITTC-FY2009-TR-41420-11, June 2009.

[28] S. Nadarajah, "A Review of Results on Sums of Random Variables," *Acta Applicandae Mathematicae: An International Survey Journal on Applying Mathematics and Mathematical Applications*, vol. 103, no. 2, pp. 131–140, Sep 2008.

[29] E. Komp *et al.*, "Implementing Web Services: Conflicts Between Security Features and Publish/Subscribe Communication Protocols," University of Kansas, Lawrence, KS, ITTC Tech. Rep. ITTC-FY2010-TR-41420-19, Feb. 2010.

[30] M. C. Edwards *et al.*, "Improving Freight Rail Safety with on-board Monitoring and Control Systems," in *Proceedings of the 2005 ASME/IEEE Joint Rail Conference*, Pueblo, CO, USA, Mar 2005, pp. 117–122.

[31] J. Fernandez *et al.*, "Transf-ID: Automatic ID and Data Capture for Rail Freight Asset Management," *Internet Computing, IEEE*, vol. 13, no. 1, pp. 22–30, Jan.-Feb. 2009.

[32] J. Ove Lauf and H. Sauff, "Secure Lightweight Tunnel for Monitoring Transport Containers," in *Third Int'l Conf. Security and Privacy in Communications Networks (SecureComm 2007)*, Nice, France, Sep 2007, pp. 484–493.

[33] L. Ruiz-Garcia *et al.*, "Review. Monitoring the intermodal, refrigerated transport of fruit using sensor networks," *Spanish Journal of Agricultural Research*, vol. 5, no. 2, pp. 142–156, 2007.