PERVASIVE STRUCTURAL HEALTH MONITORING BASED ON THE SNMP PROTOCOL

Hassan Zeineddine
American University in Dubai, UAE
hzeineddine@aud.edu

ABSTRACT

Computer-based structural health monitoring (SHM) systems are deployed in complex civil structures for continual sanity checks and early warning purposes. Skyscrapers, overextended bridges, oil refineries and pipelines are examples of civil structures that need to be continually monitored for corrosions, decays, stress and temperature change. Integrating multi-vendor SHM systems, which are technologically different, in one global monitoring system is essential for geographically spread organizations. The integration task requires a dedicated network, leased public telecommunications lines, and a protocol stack to interconnect the dispersed SHM systems. In this paper, we propose a new architecture, based on the well-established and ubiquitous Simple Network Management Protocol (SNMP), to interconnect various SHM systems with a global monitoring system through the Internet. We describe the essential components and the basic structure of the corresponding management information base (MIB). Adopting an Internet-based standard yields economical and operational benefits, but with some security drawbacks. We shall also highlight the envisioned benefits and discuss means to mitigate the security concerns.

Keywords: Structural health, global monitoring, architecture, SNMP, MIB.

1 INTRODUCTION

Structural Health Monitoring (SHM) System refers to the process of identifying damages in civil and mechanical structures [1-4]. Damage is indentified if the structure's state is outside the scope of what is considered to be the set of normal conditions. For example, a traffic light system can be in one of three normal states (yellow, red, and green). Transitioning from one state to another during normal operations must follow the finite state machine shown in Fig. 1(a). The normal state's transition process is to shift from Green to Yellow, then to Red, and back to Green.

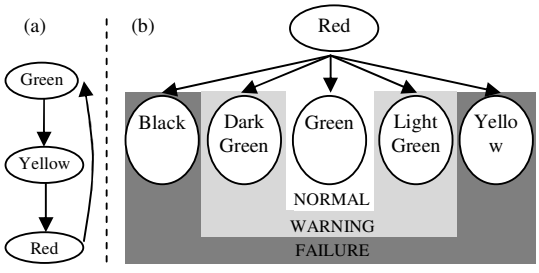

Figure 1: State transition diagram and damage severity map of a traffic light system

Transitioning from one state to another without respecting the specified state machine is considered system damage. The severity of damage defines whether it is a failure or just a warning behavior. Shifting from Red to Dark-Green instead of Green might not be a failure; however, it should be a warning behavior that requires immediate attention. An SHM system evaluates a structure's status according to a damage severity map as described in the traffic light system diagram of Fig. 1(b).

In the early days, structural health monitoring was achieved through manual sanity checks and non destructive validation procedures applied regularly to various key parts of the system [5]. With the advent of computer technology, offline measurements were collected using electronic and radioactive testing gears. Measurements results were manually ported to computer systems for further analysis. With the incorporation of network technologies in SHM systems, online measurements from testing devices and sensors were directly transmitted to a central monitoring system [6]. In general, most of the current SHM systems are composed of the following technical elements as shown in Fig. 2:

- Sensor devices
- Central monitoring device (CMD)
- Wired or wireless data transmission media, linking sensors to the CMD
- Embedded software packages.

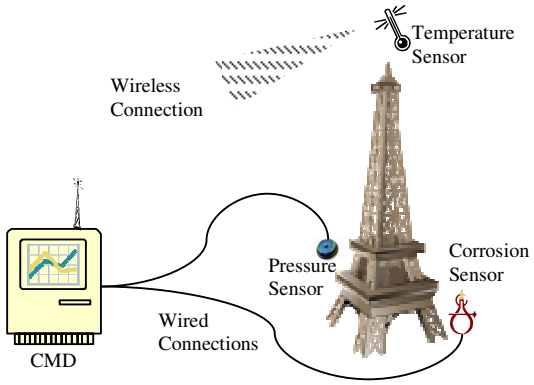

Figure 2: Structural health monitoring system

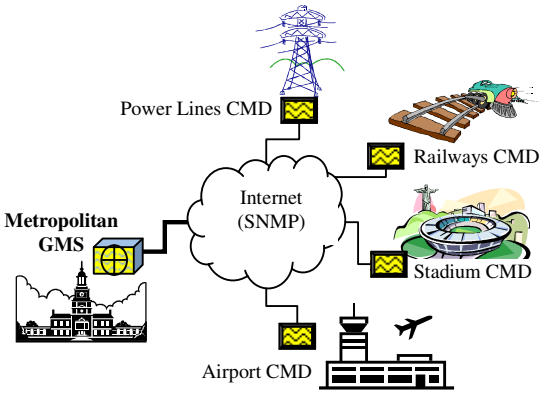

Figure 3: Global structural health monitoring sys.

A sensor device continually measures one or more structural health parameters such as heat, pressure or corrosion, and sends data over to the CMD via wired or wireless links [7]. The embedded software in the CMD manages the collected data, analyzes it to identify warnings and failures, reports behavioral trends through statistical figures and charts, and notifies interested parties about important changes [8]. A personal computer accesses the CMD through a direct link or a local network. Internet access is also possible if the CMD has a web server and allows for TCP/IP connections. Having all these important features, current CMDs from different vendors are yet not inter-operable. Although some CMDs allow the integration of multi-vendor sensors and measurement devices, they cannot exchange data or work with independent third party analysis tools. The main reason behind this limitation is the lack of standardization efforts in the field of SHM systems. A universal interface for different SHM systems (made by different vendors) is crucial for the construction of a global SHM (GSHM) system, which integrates several CMDs across various locations. For example, multi-national or municipal organizations should be able to monitor their civil structures from a central server regardless of the adopted CMDs and sensors, as described in Fig. 3. A global earthquake monitoring system should be able to seamlessly poll data from various geographical monitoring systems, aggregate the collected information, and perform statistical analysis on the data sets. The global monitoring station (GMS) should not, by any means, be a backup system duplicating the contents of the dispersed CMDs. Instead, the GMS initiates synchronous updates to retrieve predetermined parameters on demand; and, it also receives asynchronous updates for notifications on certain key events.

In this paper, we propose a standard architecture to facilitate the integration of multi-vendors SHM. Instead of designing and defining a new protocol for that purpose, we aim to leverage the existing Simple Network Management Protocol (SNMP). Deploying the SNMP protocol in a CMD complements the set of existing user interfaces with no major alterations. SNMP is used to manage equipments across a network regardless of the equipments makers. A computer-based network management station should be able to access SNMP-enabled CMDs through a local area network or the Internet. If Internet communication is involved, stringent security measures must be in place to protect management data.

2 UNDERLYING TECHNOLOGY

Before describing the SNMP protocol, we briefly explain the internal architecture and processing scheme of a typical CMD [9, 10]. The basic architecture is composed of several stacked system layers as shown in Figure 4. The lowest layer is the Sensor Interface Layer, which includes device drivers and essential software adaptors to interface with the deployed sensors. Moving up through the stack, the Data Normalization Layer formats and normalizes the collected data to match with the adopted database schema. The Database Management Layer saves normalized data in corresponding tables. The heart and soul of the monitoring system is the Statistical Layer, where data is analyzed and assessed with respect to predefined state machines. This layer queries the database to analyze information and indentify possible damages. The database might contain special table to store analysis results. The User Interface Layer communicates with users through various interface types such as Command Line (CLI), Graphical (GUI), and Web (HTTP). This top layer interacts with the database or analysis layer to fulfill user initiated requests and configurations. Finally, a Control Layer should be perpendicular to all other layers allowing users to configure, troubleshoot and update all system components as needed.

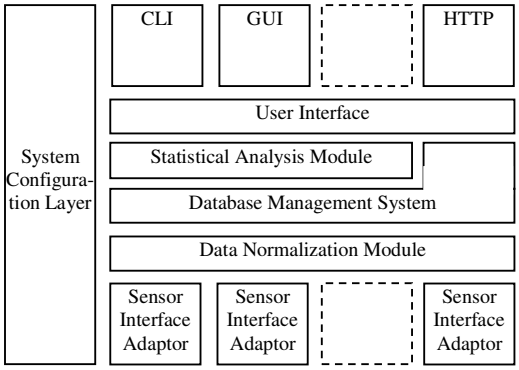

Figure 4: Central monitoring device architecture

What is SNMP? SNMP is a well established protocol that has gone through several revisions since the early 90s [11, 12]. SNMP version 3 (SNMPv3) is the latest revision and most secure [13]. SNMP transactions can be summarized in three basic functions: getting management parameters (SNMP Get), setting parameters (SNMP Set), and issuing change notifications (SNMP Trap). A management parameter holds a value reflecting the status of a managed device. The exchanged parameters' names, data types, access rights and values are carried in a Protocol Data Unit (PDU). The access right could be read-only, read-write, or write-only. For example, the temperature of a critical component in a structure can be represented with a read-only parameter. It can be read using an SNMP Get from any SNMP-enabled network management platform regardless of its vendor. In addition, an SNMP Trap can be configured to issue notifications about any change in values that match certain predefined criteria.

An SNMP system is mainly based on client-server architecture. An SNMP manager acting as client sends SNMP requests to one or more SNMP agents. Playing the server role, an agent responds according to the manager's request. An agent can also notify the manager on certain events based on predefined criteria. The management information at the SNMP agent is organized in a tree data structure, specified by a Management Information Base (MIB) [14]. A standard syntax, known as Abstract Syntax Notation One (ASN.1) [15], is used to define MIBs. All management parameter identifiers in a MIB are based on a hierarchical namespace that contains Object Identifiers (OID). An OID is globally unique and represents one management parameter. Normally, the top portion of the OID identifies a common standard and a particular organization; and, the lower part identifies a management parameter of an equipment or device. For example, if *s.o.d.t* is an OID, the numbers *s, o, d,* and *t* would respectively represent a standard body, an organization, a device, and a temperature parameter. All MIB object identifiers are managed by the Internet Assigned Numbers Authority (IANA) according to the corresponding guidelines in [16].

3 PROPOSED ARCHITECTURE

In order to exchange SNMP messages with an SHM system, we embed an SNMP agent in the corresponding CMD, as shown in Figure 5. If embedding an agent requires a relatively long development life cycle, a proxy agent running outside the box can be a feasible interim solution. In both cases, the agent interacts with the CMD database and statistical modules. A local area network (LAN) or the internet connects the agent with an SNMP manager, embedded in the GMS. Synchronous communication from manager to agent runs in the form of SNMP Get requests, which read various SHM parameters. On the other hand, asynchronous communication in the opposite direction occurs in the form of SNMP Traps notifying the SNMP manager about certain events at the agent side. We define event notification messages in the corresponding MIB module. After compiling the MIB, we implement notification criteria and conditions in the SNMP agent. Since the GMS only monitors and does not update CMD parameters, we disable the SNMP Set functionality. Preventing the GMS from conducting remote updates is invaluable for security reasons.

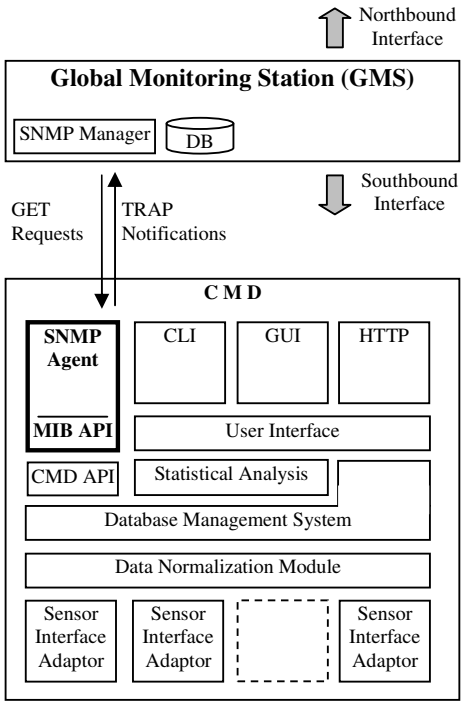

Figure 5: Proposed GHMS architecture

Although the proposed framework requires further standardization efforts, we discuss the basic components based on a laboratory prototype. The

first step in our approach is to construct the basic SHM MIB, which defines the monitored parameters and essential notifications. Second, we describe the SNMP agent. Last, we discuss the SNMP manager's functionalities, and end-to-end message flow.

3.1 Management Information Base

To define the parameters that should be monitored through the SNMP platform, we need to specify and implement the corresponding MIBs at each participating CMD. In addition, the same MIBs must be specified at the GMS. The adopted MIB definitions form the interface constraints between the CMD and the corresponding GMS. Nothing beyond the defined set of parameters can be monitored. We propose a basic SHM MIB which can be extended as deemed necessary by researchers and CMD manufacturers. A CMD can be equipped with the basic MIB definitions or manufacturer specific definitions. A basic MIB defines SHM parameters such as corrosion rate, strain, pressure, vibration, temperature, humidity, crack and tilt. An SHM manufacturer can define its proprietary set of parameters after acquiring its own OID from IANA. Regardless of the measurement methods and employed sensors across various CMDs, the adopted measurement units should comply with a universal standard. A unit transformation process at the SNMP agent is essential to match the adopted standard units at the GMS. For example, corrosion might be measured using different methods; however, the corrosion rate can be universally described in the mpy unit (mils per year), reflecting the loss rate in millimeters per year. A bridge vibration can also be measured in Hertz (Hz), quantifying the bridge reaction to wind and traffic flow. Continual strain on a given object causes structural deformation. The microStrain unit (μStrain) represents the deformation of one part per million. The universal unit of pressure KiloPascal (kPa) can universally represent soil settlement and foundation heave. Other metrics such as structural objects temperature, humidity, crack, and tilt must all be reflected in appropriate universal units. An SHM standard organization shall govern the universal set of measurement units, and take ownership of the corresponding MIB.

To support the basic metrics in an SHM system, we propose to register an SHM specific MIB module under the mib-2 SNMP module. The module number must be reserved through a registration process defined in the ITU recommendation X.660 [17]. In our laboratory emulation, we rely on an experimental MIB under the *iso.org.dod.internet.experimental* (*1.3.6.1.3*) module. The absolute OID of the final SHM MIB will be *iso.org.dod.inernet.mgmt.mib-2.shm* (*1.3.6.1.2.1.shm*) as shown in Fig. 6. All parameters defined under the SHM MIB are prefixed with the string "shm". For example, a table representing a list of corrosion measurements at various spots is labeled *shmCorrosionRateTable*; the corresponding relative OID is *.shm.shmCorrosionRateTable* (*.shm.2*).

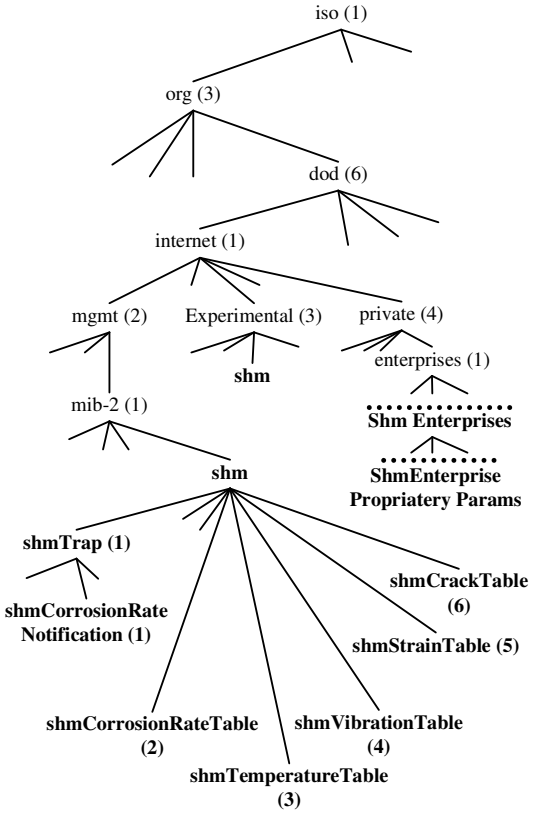

Figure 6: Object identifiers hierarchy in the SHM management information base

Fig. 7 depicts a segment of the SHM MIB highlighting the essential ASN.1 scripts for defining the *shmCorrosionRateTable*. The table's entries hold essential corrosion attributes as defined in the *shmCorrosionRateEntry* object type. In the script below, only two basic attributes are defined, *shmCorrosionRateObjectLabel* representing the monitored object's unique label, and *shmCorrosionRateValue* reflecting the corresponding corrosion rate value. In principle, capturing corrosion rate's information might require more than just two attributes. Each monitored object is assigned a globally unique identifier, an OID branching out of the *shmEnterprise* node on the MIB tree. The identifier includes a unique client code and an object label. For instance, *1.3.6.1.4.1.9999.455.7* uniquely represents the 7^{th} monitored object at customer site no. *455*, which is managed by SHM vendor no. *9999*. The relative OID *.9999.455.7* is enough to indentify that object since the absolute OID can be implied. The corrosion rate value is of type *CorrosionRateType*, a data type defined in the MIB to represent mils per year values. Similar definitions

```
shmTrap OBJECT IDENTIFIER ::= { shm 1 }


CorrosionRateType ::= TEXTUAL-CONVENTION
   DISPLAY-HINT "d"
   STATUS            current
   DESCRIPTION
         "A data type representing an object's
         corrosion rate in mpy (Mils per Year)"
   SYNTAX            Integer32


MonitoredObjectLabel ::= TEXTUAL-CONVENTION
          STATUS      current
          DESCRIPTION
                "A data type representing the unique
                label of  a  monitored object."
          SYNTAX      OBJECT IDENTIFIER


shmCorrosionRateTable OBJECT-TYPE
   SYNTAX  SEQUENCE OF ShmCorrosionRateEntry
   MAX-ACCESS    not-accessible
   STATUS            current
   DESCRIPTION
      "table of corrosion measurements"
   ::= { shm  2 }


shmCorrosionRateEntry OBJECT-TYPE
   SYNTAX            hmCorrosionRateEntry
   MAX-ACCESS     not-accessible
   STATUS            current
   DESCRIPTION
      "An entry in the shmCorrosionTable"
   INDEX   { shmCorrosionRateObjectLabel}
   ::= { shmCorrosionRateTable 1 }


shmCorrosionRateEntry ::= SEQUENCE {
   shmCorrosionRateObjectLabel MonitoredObjectLabel,
   shmCorrosionRateValue CorrosionRateType,

   -- . . . . other attributes . . . . .
}


shmCorrosionRateObjectLabel   OBJECT-TYPE
   SYNTAX            MonitoredObjectLabel
   ACCESS            read-only
   STATUS            current
   DESCRIPTION
      "Monitored object's unique label"
   ::= { shmCorrosionRateEntry 1 }


shmCorrosionRateValue   OBJECT-TYPE
   SYNTAX            CorrosionRateType
   ACCESS            read-only
   STATUS            current
   DESCRIPTION
      "the rate of corrosion on a given object,
        measured in  mpy (Mils per Year)"
   ::= { shmCorrosionRateEntry 2 }


shmCorrosionRateNotification NOTIFICATION-TYPE
   STATUS current
   OBJECTS { shmCorrosionRateObjectLabel,
                shmCorrosionRateValue }
   DESCRIPTION "corrosion rate notification"
   ::= { shmTrap 1 }
```

Figure 7: Segment of the SHM MIB definitions

are made for all other tables such as, *shmStrainTable*, *shmCrackTable*, and *shmVibrationTable*.

Besides the defined MIB parameters, we specify the asynchronous notification types that are sent from the CMD to the corresponding GMS. All notification types are described under the MIB node *shm.shmTrap*. For example, the corrosion rate notification, whose relative OID is *shm.shmTrap.shmCorrosionRateNotification* or *shm.1.1*, is used to notify the GMS about key corrosion events. A notification type specifies the notification message layout. The layout could include one or more MIB scalars. A notification is fired based on predetermined criteria, which are defined at the agent side as discussed in the next section.

3.2 SNMP Agent

After parsing the MIB and generating the corresponding Application Programming Interfaces (APIs), we link the MIB API functions with their corresponding CMD functions. At this point, we implement all formatting and unit transformation code. Since all SHM MIB parameters are read-only, we don't implement the corresponding API Set functions. We link only the API Get functions to the corresponding CMD procedures. For example, the API for retrieving the corrosion rate of a monitored object from the corresponding corrosion rate table is GetCorrRateValue(*ShmCorrosionRateEntry cre*). As shown in the pseudo code of Fig. 8, the API is a wrapper around internal CMD functions, which query the internal database and covert results to a universal unit. In addition to the APIs work, we update the SNMP agent's event handler to send in SNMP notifications based on predefined criteria. For example, if the corrosion rate rises above a given threshold, a shmCorrosionRateNotification trap must be fired through the SNMP network to the subscribed SNMP managers. After configuring and linking all APIs, we compile and embed the SNMP agent in the corresponding CMD. The newly deployed interface provides seamless connection with the GMS SNMP manager. Each CMD must be assigned an IP address and have port 161 open for SNMP communications.

```
GetCorrRateValue(ShmCorrosionRateEntry shmCR)
{

      ShmCorrosionRateEntry          entry          =
(ShmCorrosionRateEntry) shmCR;

      Vector                         m_oid          =
toVector(enry.shmCorrosionRateObjectLabel);

      Integer  cRate = DbGetCorrosionRate(m_oid);
      Integer univCorRate = ConvertCorRate(cRate)
      return univCorRate;
}
```

Figure 8: API Get function in the SNMP agent

3.3 SNMP Manager

At the GMS, we should load standard and vendor-specific MIBs in the SNMP manager. All communications with remote SNMP agents flow through a network interface and port 161. In addition, we must configure the IP addresses of all participating CMDs. On the manager's southbound interface, special scripts retrieve information from the CMDs using standard SNMP commands, at various time intervals depending on the information criticality. Besides the usual SNMP Get command, frequent SNMP Walk commands are essential to go through the entire MIB structure and reconcile the GMS database with its CMD counterparts. In addition, SNMP GetTable command is useful for retrieving all rows in a given table, such as the corrosion rates of all monitored objects at a given CMD. On the Northbound side, the GMS provides different interface types such as, GUI, CLI, HTTP, SMS, and Email. Between the southbound and northbound interface layers, a local database serves as a repository of collected information from all CMDs. In addition, a trap handler captures all SNMP notifications, updates the database accordingly, and informs stakeholders via email or SMS messages.

3.4 System Security

Accessing information through an internet based protocol, like SNMP, entails several security issues. SNMP could be a fertile environment for hackers if not deployed with stringent security measures. The list below summarizes some of the security threats in an SNMP based system:

- Adversarial information update
- Unauthorized access to private data
- Unauthorized SNMP traffic Interception, and content tampering
- Denial of service attack

With respect to our framework, adversarial modification of information is not an issue since all MIB objects have read-only access right. Unauthorized data access and traffic interception are eliminated by relying on the third version of SNMP, which greatly enhances security. Version 3 guarantees confidentiality by encrypting the entire SNMP message payload. Other message integrity features ensure that exchanged payloads between the GMS and CMD are safe against tampering attempts along the route. In addition, a message source is verified through strict authentication mechanisms. Besides SMPv3's strong security features, setting stringent firewall rules restrict access of SNMP information to only the authorized GMS. In addition, a VPN with the highest level of data encryption safeguards the system against illegal traffic interception. An intrusion detection system also helps in minimizing risks of illegal intrusions and denial of service attacks. These basic IT countermeasures are typically deployed in most communication systems that are connected to the Internet. Thus, securing the proposed SNMP-based global structural health monitoring system is easily achieved by leveraging the existing security infrastructure.

3.5 Message Sequence

The GMS is updated via synchronous messages and asynchronous updates as depicted in the message sequence diagram of Fig. 9. Information retrieval requests are either user or system driven. Users interested in learning about the status of some particular structures issue information retrieval requests through various user interfaces, i.e. GUI, CLI, or automated scripts. In addition, the GMS system can internally initiate similar requests to update its local database. Regardless of the information requests' source, all requests are passed to the SNMP manager via internal API calls. These calls trigger appropriate SNMP Get requests towards the SNMP Agent at the corresponding CMD. The agent retrieves the requested attributes from the CMD Database using SQL statements. Note that the CMD Database is continually and asynchronously updated by the Data Normalization Module, which collects and normalizes sensor-generated data. When the SNMP agent receives the requested SQL attributes from the CMD database management system, it sends an SNMP response to the SNMP manager through the underlying network. The manager passes the retrieved information To the GMS' interface layer via the corresponding API parameters. Finally, the GMS informs the involved user through the adopted user interface, and updates the local database.

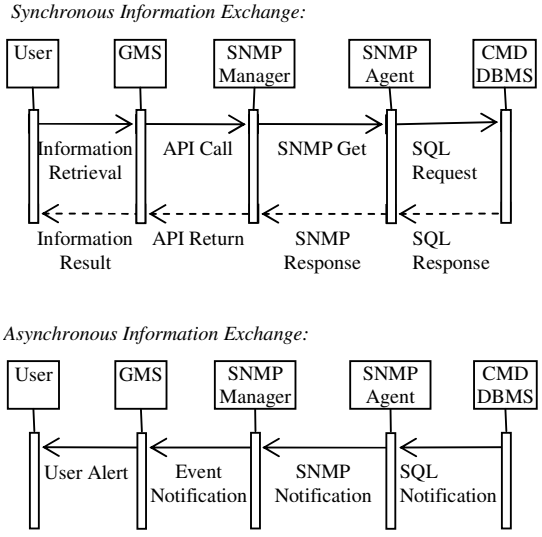

Figure 9: Message sequences in a global SHM

At some points in time, users need to be alerted about critical events pertaining to the monitored structures, such as the corrosion rate of a steal beam going above an acceptable threshold. Thus, an asynchronous update mechanism is required to handle internally triggered notification messages about critical events. These events are triggered at the database management system, where an SQL notification mechanism must be implemented. Microsoft SQL server offers ready-made notification functionalities, saving the hassle of coding a notification system. The SQL notification system watches a predetermined set of attributes, and reports any change in values beyond the corresponding acceptable ranges. The SNMP agent's event handler at the CMD captures all SQL notifications, and fires corresponding SNMP notifications to the SNMP manager at the GMS. The SNMP manager listens for, and processes SNMP notifications through a trap handler. Consequently, the handler calls appropriate GMS interface functions to alert users about the occurring events. Alerts can be in the form of graphical updates, emails and SMS messages.

4 FEASIBILITY DISCUSSIONS

We discuss the feasibility of our proposed global structural health monitoring system in various contexts, operational, technical, legal/political, human factors and economical [18]. Operationally, organizations with geographically dispersed structures appreciate the GSHM's functionalities, given the challenge of interoperability among SHM systems built around different technologies. Oil refineries, metropolitan governments, large ports, and etc. do not need to rely on a single SHM vendor and technology in order to achieve interoperability. SNMP is a ubiquitous open standard which provides seamless system integration. Simply, with a LAN or Internet connection these organizations are able to have an end-to-end view of their key structures and take appropriate decisions from a central station. In addition, the individual SHM systems' initial operation mode is not disturbed since the proposed global system does not alter the existing process; rather, we just plug in an additional interface (SNMP agent). Since the GMS is purely a monitoring station with read-only access rights according to the proposed architecture, conflicts of authority between the GMS and local CMDs are negligible. Based on the above facts, the proposed architecture seamlessly fit in the existing infrastructure and enhances the overall global monitoring operation. Furthermore, collecting distributed information in a central repository offers better insights on behavioral trends and events correlations across various sites.

The proposed architecture relies on existing and mature technology, such as IP and SNMP. SNMP is an established international standard, which is widely adopted by all network equipment makers. Defining the MIBs and developing SNMP agents or proxy agents for existing CMDs are the basic steps to build a global monitoring system that is interoperable with multivendor CMDs. Relying on an established standard and widespread technology offers the benefit of easily finding skilled resources ready to be on board with minimal training. Skilled labor is essential to develop, maintain and interact with the system.

The system would need to operate within some legal and political constraints. Vendors and customers' resistance to embrace the new architecture due to privacy and security concerns might derail the system development. However, relying on SNMPv3 and using read-only transactions mitigate some of these concerns. In addition, leveraging the existing network security infrastructure helps in keeping many of the security threats at bay.

On the economy scale, relying on existing network infrastructures to integrate distributed SHM systems is relatively a cheap investment compared to the valuable gain in knowledge. The emerging data repository at the GMS produces a priceless data warehouse for data mining activities and analysis, such as tracking trends and identifying patterns. A crack in one structure can be correlated to an abnormal soil movement few miles away. More interesting real-time correlations can be made during regional natural disasters such as storms and earthquakes, which could help in optimizing search and rescue efforts and overall disaster management. Real-time and off-line event correlations would definitely enrich research activities and save the economy some avoidable losses. In addition, a global monitoring system in a geographically dispersed organization requires less monitoring resources than several distributed local SHM systems. Furthermore, organizations might opt to outsource the monitoring task to a third party by linking their SHM systems to the third party's GMS through the Internet. The resulting business model, related to outsourcing, leads to SHM budget's optimization and lucrative business opportunities for investors.

5 CONCLUSION

The integration of disconnected and technologically diverse structural health monitoring (SHM) systems in a global SHM (GSHM) is addressed in this paper. The proposed architecture relies on the well established Simple Network Management protocol (SNMP). Adopting the mature SNMP infrastructure saves the hassle of designing corresponding data management and messaging schemes. The bulk of the work is to define a standard management information base (MIB), compiling it in SNMP agents, and linking agents to SHM databases

via appropriate Application Programming Interfaces (APIs). On the SNMP manager side, interfacing with different agents in dispersed SHM systems can be achieved from a simple MIB browser or a sophisticated graphical user interface. Clearly, the proposed architecture is highly feasible from a technical and human factors perspective. It fits seamlessly within the existing SHM's operation without major alterations to the existing process. Legally and politically, relying on SNMPv3 and read-only access of SHM attributes reduces security concerns, and hence stakeholders' resistance to adopt the new architecture. On the economical side, the collected information at the GMS forms a data warehouse for further priceless data mining activities. In addition, the deployed SNMP infrastructure opens the door for possible outsourcing of several SHM functions leading to budget optimization and new business opportunities.

Finally, the future plan is to propose an Internet Request for Comment (RFC) document defining the basic SHM MIB. The draft shall list common SHM attributes and basic notifications. In addition, the MIB will be compiled in an open-source SNMP agent to be a focal point for further research.

6 REFERENCES

[1] C. R. Farrar, K, and Worden: An introduction to structural health monitoring, Philosophical Transaction Of Royal Society 365, pp. 303-315 (2007).

[2] P. C. Chang, A. Flatau, and S. C. Liu: Review paper - health monitoring of civil infrastructure, Structural Health Monitoring, Vol. 2, pp. 257-267 (2003).

[3] H. Sohn, C. R. Farrar, N. F. Hunter, and K. Worden, Structural health monitoring using statistical pattern recognition techniques, Journal of Dynamic Systems -T. ASME, Vol. 123, pp. 706-711 (2001).

[4] A. Elgamal, J.P. Conte, S. Masri, M. Fraser, T. Fountain, A. Gupta, M. Trivedi, and M. El Zarki: Health monitoring framework for bridges and civil infrastructure, Proceedings of the 4th International Workshop on Structural Health Monitoring, pp. 123-130 (2003).

[5] C. Hellier: Handbook of Nondestructive Evaluation, McGraw-Hill. (2003).

[6] K. Worden, and J. M. Dulieu-Barton: An overview of intelligent fault detection in systems and structures, International Journal of Structural Health Monitoring Vol. 3, pp. 85–98 (2004).

[7] B. F. pencer, M. E. Ruiz-Sandoval, and N. Kurata, Smart sensing technology: opportunities and challenges, Structural Control Health Monitoring, Vol. 11, pp. 349-368 (2004).

[8] D. Inaudi, B. Glisic, and L. Manetti: Integrated Systems for Structural Health Monitoring, Structural Health Monitoring, pp. 1-12 (2009).

[9] Y. Gao, B. F. Spencer: Structural Health Monitoring Strategies for Smart Sensor Networks, Newmark Structural Engineering Laboratory Report Series, Vol. 011, University of Illinois at Urbana-Champaign (2008).

[10] T. Nagayama, and B. F. Spencer: Structural Health Monitoring Using Smart Sensors, Newmark Structural Engineering Laboratory Report Series, Vol. 001 (2007).

[11] J. Case, M. Fedor, M. Schoffsall, and J. Davin: A Simple Network Management Protocol (SNMP), Network Working Group RFC 1157 (1990).

[12] J. Case, K. McCloghrie, M. Rose, and S. Waldbusser: Introduction to Community-based SNMPv2, Network Working Group RFC 1901, (1996).

[13] D. R. Mauro and K. J. Schmidt: Essential SNMP, O'Reilly & Associates, (2001).

[14] K. McCloghrie, and M. Rose: Management Information Base for Network Management of TCP/IP-based internets – MIB-II, Network Working Group RFC 1213, (1991).

[15] O. Dubuisson, and P. Fouquart: ASN.1 – Communication Between Heterogenous Systems, Elsevier-Morgan-Kaufman, (2000).

[16] T. Narten and H. Alvestrand: Guidelines for Writing an Iana Considerations Section in RFCs, Network Working Group RFC 5226, (2008)

[17] ITU recommendation X.660: Information technology - Open systems interconnection - Procedures for the operation of OSI registration authorities: General procedures and top arcs of the ASN.1 object identifier tree, (2004)

[18] R. Overton: Feasibility Studies Made Simple, Martin Books, (2007).