Available online at www.elixirpublishers.com (Elixir International Journal)

Computer Science and Engineering

Elixir Comp. Sci. & Engg. 91 (2016) 38540-38544

ISSN: 2229-712X

Viable intrusion detection on static and dynamic resource allocation on wireless adhoc network

N.Praveena[1,*] and Ch Meher Babu[2]

[1]Information Technology, V.R.Siddhartha Engineering College Vijayawada (A.P), India.

[2]Computer Science & Engg Dept Nist, Vijayawada India.

ARTICLE INFO

Article history:

Received: 23 March 2013;

Received in revised form:

18 February 2016;

Accepted: 23 February 2016;

Keywords

Security,

Denial of Service (DoS),

Wireless Ad Hoc Networks,

Distributed Probing,

Secure Routing Protocols, Simulation

ABSTRACT

Control architecture for resource allocation in satellite networks is proposed, along with the specification of performance indexes and control strategies. The latter, besides being based on information on traffic statistics and network status, rely upon some knowledge of the fading conditions over the satellite network channels. The resource allocation problem consists of the assignment, by a master station, of a total available bandwidth among traffic earth stations in the presence of different traffic types. Traffic stations are assumed to measure continuously their signal fade level, but this information may either be used only locally or also communicated to the master station. According to the information made available on-line to the master station on the level of the fading attenuation of the traffic stations, the assignment can be made static, based on the a priori knowledge of long-term fading statistics, or dynamic, based on the updated measurements. In any case, the decisions can be adapted to slowly time-varying traffic characteristics. At each earth station, two basic traffic types are assumed to be present, namely guaranteed bandwidth, real-time, synchronous data (stream traffic), and best effort traffic (datagram traffic). Numerical results are provided for a specific architecture in the dynamic case, in a real environment, based on the Italian satellite national coverage payload characteristics.

Introduction

The emerging field of wireless sensor networks combines sensing, computation, and communication into a single tiny device. Through advanced mesh networking protocols, these devices form a sea of connectivity that extends the reach of cyberspace out into the physical world. As water flows to fill every room of a submerged ship, the mesh networking connectivity will seek out and exploit any possible communication path by hopping data from node to node in search of its destination. While the capabilities of any single device are minimal, the composition of hundreds of devices offers radical new technological possibilitie

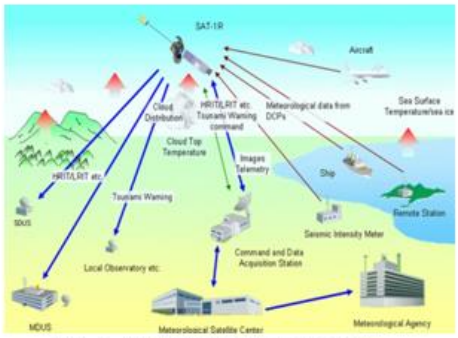

Fig. 1. Ad-hoc with connecting nodes

The most straightforward application of wireless sensor network technology is to monitor remote environments for low frequency data trends. For example, a chemical plant could be easily monitored for leaks by hundreds of sensors that automatically form a wireless interconnection network and immediately report the detection of any chemical leaks. Unlike traditional wired systems, deployment costs would be minimal. Instead of having to deploy thousands of feet of wire routed through protective conduct, installers. Adaptation mechanisms n respond to changes in network topologies or can cause the network to shift between drastically different modes of operation. Current wireless systems only scratch the surface of possibilities emerging from the integration of low-power communication, sensing, energy storage, and computation.

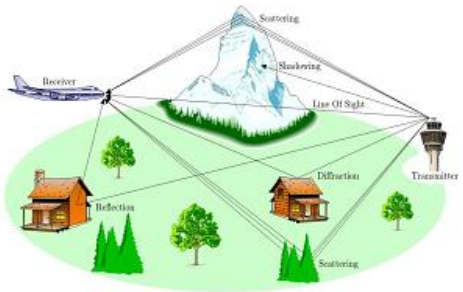

Fig.2. Example of mobility

Unlike traditional wireless devices, wireless sensor nodes do not need to communicate directly with the nearest high-power control tower or base station, but only with their local peers. Instead, of relying on a pre-deployed infrastructure, each individual sensor or actuator becomes part of the overall infrastructure. Peer-to-peer networking protocols provide a mesh-like interconnect to shuttle data between the thousands of tiny embedded devices in a multi-hop fashion. The flexible mesh architectures envisioned dynamically adapt to support introduction of new nodes or expand to cover a larger geographic

Tele:

E-mail address: praveena.4u@gmail.com

region. Additionally, the system can automatically adapt to compensate for node failures.

The vision of mesh networking is based on strength in numbers. Unlike cell phone systems that deny service when too many phones are active in a small area, the interconnection of a wireless sensor network only grows stronger as nodes are added. As long as there is sufficient density, a single network of nodes can grow to cover limitless area.

Architecture

The concept of wireless sensor networks is based on a simple equation: Sensing + CPU + Radio = Thousands of potential applications.

As soon as people understand the capabilities of a wireless sensor network, hundreds of applications spring to mind. It seems like a straightforward combination of modern technology. However, actually combining sensors, radios, and CPU's into an effective wireless sensor network requires a detailed understanding of the both capabilities and limitations of each of the underlying hardware components, as well as a detailed understanding of modern networking technologies and distributed systems theory. Each individual node must be designed to provide the set of primitives necessary to synthesize the interconnected web that will emerge as they are deployed, while meeting strict requirements of size, cost and power consumption. A core challenge is to map the overall system requirements down to individual device capabilities, requirements and actions. To make the wireless sensor network vision a reality, architecture must be developed that synthesizes the envisioned applications out of the underlying hardware capabilities.

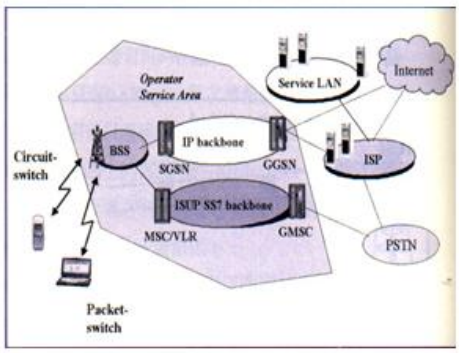

Fig. 3. Example of a selective Random Packet

To develop this system architecture we work from the high level application requirements down through the low-level hardware requirements. In this process we first attempt to understand the set of target applications. To limit the number of applications that we must consider, we focus on a set of application classes that we believe are representative of a large fraction of the potential usage scenarios. We use this set of application classes to explore the system-level requirements that are placed on the overall architecture. From these system-level requirements we can then drill down into the individual node-level requirements. Additionally, we must provide a detailed background into the capabilities of modern hardware.

After we present the raw hardware capabilities, we present a basic wireless sensor node. The Rene node represents a first cut at system architecture, and is used for comparison against the system architectures.

Sensor network application classes

The three application classes we have selected are: environmental data collection, security monitoring, and sensor node tracking. We believe that the majority of wireless sensor network deployments will fall into one of these class templates.

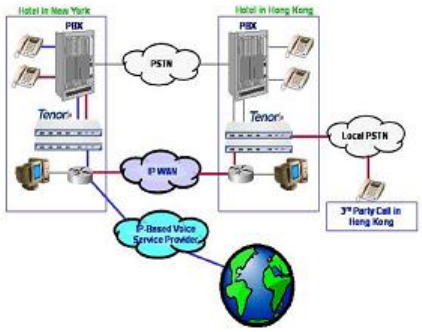

Fig. 4. Example of an Attack Model

Environmental Data Collection

A canonical environmental data collection application is one where a research scientist wants to collect several sensor readings from a set of points in an environment over a period of time in order to detect trends and interdependencies. For the data to be meaningful it would have to be collected at regular intervals and the nodes would remain at known locations

locations.

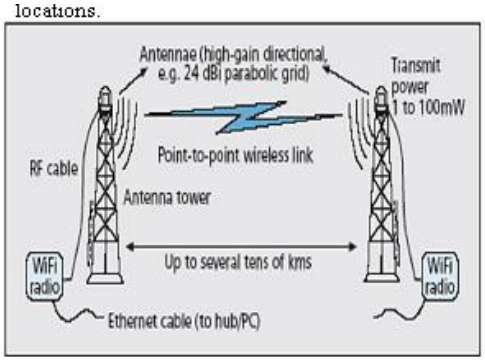

Fig. 5. Example of Monitor identification method

At the network level, the environmental data collection application is characterized by having a large number of nodes continually sensing and transmitting data back to a set of base stations that store the data using traditional methods. These networks generally require very low data rates and extremely long lifetimes. In typical usage scenario, the nodes will be evenly distributed over an outdoor environment. This distance between adjacent nodes will be minimal yet the distance across the entire network will be significant.

The routing strategy can then be used to route data to a central collection points. In environmental monitoring applications, While the time variant nature of RF communication may cause connectivity between two nodes to be intermittent, the overall topology of the network will be relatively stable. Environmental data collection applications typically use tree-based routing topologies where each routing tree is rooted at high-capability nodes that sink data. Data is periodically transmitted from child node to parent node up the tree-structure until it reaches the sink. With tree-based data collection each node is responsible for forwarding the data of all its descendants. The most important characteristics of the environmental monitoring requirements are long lifetime, precise synchronization, low data rates and relatively static topologies. The data transmissions can be delayed inside the network as necessary in order to improve network efficiency.

Security Monitoring

Security monitoring networks are composed of nodes that are placed at fixed locations throughout an environment that

continually monitor one or more sensors to detect an anomaly. A key difference between security monitoring and environmental monitoring is that security networks are not actually collecting any data. Each node has to frequently check the status of its sensors but it only has to transmit a data report when there is a security violation.

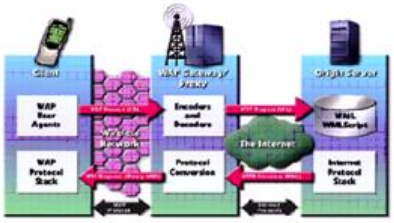

Fig.6. An Example of Filtering fields

If a node were to be disabled or fail, it would represent a security violation that should be reported. For security monitoring applications, the network must be configured so that nodes are responsible for confirming the status of each other.

The accepted norm for security systems today is that each sensor should be checked approximately once per hour. Combined with the ability to evenly distribute the load of checking nodes, the energy cost of performing this check becomes minimal. Once detected, a security violation must be communicated to the base station immediately. The latency of the data communication across the network to the base station has a critical impact on application performance. Users demand that alarm situations be reported within seconds of detection. This means that network nodes must be able to respond quickly to requests from their neighbors to forward data.

In security networks reducing the latency of an alarm transmission is significantly more important than reducing the energy cost of the transmissions. This is because alarm events are expected to be rare. In a fire security system alarms would almost never be signaled. In the event that one does occur a significant amount of energy could be dedicated to the transmission. Reducing the transmission latency leads to higher energy consumption because routing nodes must monitor the radio channel more frequently.

Node tracking scenarios

The system breaks down when objects do not flow from checkpoint to checkpoint. In typical work environments it is impractical to expect objects to be continually passed through checkpoints.

Fig.7. An Example of Filtering fields

With wireless sensor networks, objects can be tracked by simply tagging them with a small sensor node. The sensor node will be tracked as it moves through a field of sensor nodes that are deployed in the environment at known locations. Instead of sensing environmental data, these nodes will be deployed to sense the RF messages of the nodes attached to various objects. A database can be used to record the location of tracked objects relative to the set of nodes at known locations. With this system, it becomes possible to ask where an object is currently, not simply where it was last scanned.

System Evaluation Metrics

One goal of this context is to present an understanding of the tradeoffs that link each axis of this space and an understanding of current capabilities. The architectural improvements and optimizations we present in later chapters are then motivated by increasing the ability to deliver these capabilities and increasing the volume of the capability hypercube.

Problems and challenges

Audit Source Location

Since the audit source is the most important source of information about ongoing activity in the supervised environment, the result of problems here can be devastating for the performance of an IDS. This section covers the main problems with the different audit sources.

Network Packets

The audit source most used today is network packets. When data is picked up before it has reached its destination, as is done when picking up network packets, there is a problem if the data has been encrypted in any malicious behavior hidden within the packet can pass undetected. To successfully monitor a switched network an IDS is either needed on each individual segment or on each of the switches span ports.

Host and Application Log Files

This makes it easy for attackers to evade detection by the IDS. A performance problem can arise if we want to increase the amount of logging.. Intrusion Detection Systems – Technologies, Weaknesses Trends Problems and Challenges.

System and API Calls

The downside of it is that several attacks exploit the same flaw and therefore look the same on this level in the system. Therefore, it is very hard to differentiate between different attacks and to truly know the originating process. Because of this, an IDS that uses system and API calls as its primary source will sometimes generate the same general event for several different attacks. This makes finding and stopping the source of the attack much harder.

IDS Sensor Alerts

The problem is that they all have implemented the communication different from each other. This makes it very hard to manage and supervise several IDSs from different vendors at the same console. In addition, SNMP messages are quite simple in nature and limit the amount and type of information that can be sent.

Detection Method

One problem that all detection methods suffer from is the difficulty of keeping the false positive rate small relative the true positive rate. Even if the IDS has a small false positive rate, say 0.1%, the number of false alarms related to the number of real alarms will be high. The problem is based on a statistical phenomenon called the base-rate fallacy.

Knowledge-based

The main problem with knowledge-based detection methods is that they cannot detect attacks for which no signature exists. Without constant updating of the signatures the IDS will have problems detecting new attacks. Another problem with signatures is that if they are too narrow an attacker could potentially bypass the IDS by changing the attack slightly.

Behaviors-base

This is a very good approach to detecting attacks since it can also detect new attacks, but these methods have some major disadvantages. One big problem is how to define what is normal. Looking at, for example, network packets to detect anomalies

(as the behavior-based part of protocol decode does) the IDS searches for deviations from the protocol specification for the actual protocols analyzed.

Behavior on Detection

When used with proactive/reactive responses, an IDS could easily be fooled into blocking random addresses. Carelessly implementing automated passive alerts could result in the Intrusion Detection Systems – Technologies, Weaknesses and Trends Problems and Challenges security officer being paged in the middle of the night every time someone scanned the ports on the outer firewall.

Passive Alerting

The most common response used in the studied IDSs is notify console. Every time an event triggers this response, the user interface of the IDS is updated with the new alert information. The monitoring security officers review the alerts, True session recording, where the packet payload is recorded as well, requires even more resources.

Reactive Response

If the attack involved installing a listening service, like a telnet daemon, blocking the IP-address of the attack only bought the time it takes for the attacker to hijack a different IP-address and logon to the newly installed service.

Proactive Response

False positives are particularly dangerous in this area. Proactive responses prevent system calls from executing, and false positives could lead to a denial of service situation for a legitimate application or process running at the host. Automatic proactive/reactive responses should only be used when confidentiality and integrity outweighs availability.

Usage Frequency

Most of the systems studied continuously monitor their audit sources. To be able to detect an attack as fast as possible, this is necessary. The problem is that it can be really resource exhausting. A NIDS monitoring a gigabit connection can have a lot of Intrusion Detection Systems – Technologies, Weaknesses and Trends

Proposed mechanism

We described the fundamental concept behind discrete time event-driven simulations; it is easy to implement a toy simulation runtime that can process events. However, the implementation found in a real simulator is considerably more complex due to the need to export this functionality behind a programming interface that is easy to use and that does not significantly aspect performance. The event execution main loop but there is other operations that are fundamental to a discrete time event-driven simulator:

Run: execute every event in order of increasing timestamps until there are no more events or the user triggers Stop.

Stop: set the global stop flag to true to make sure that the Run function returns at the next available opportunity.

Schedule: create a new event, insert it in the global event list, and return a reference to the newly-created event.

Cancel: disable the execution of an event that is present in the global event list.

Cancel is usually considered a constant-time operation that merely sets a disabled flag in the event that is checked by Run before executing that event.

Remove: Remove an event from the global event list to ensure that it is never executed. This operation is usually considered to be of O (n) complexity (where n is the number of events).

Status: return whether an existing event has expired or is still running In practice, there is very little variance across deferent simulators in the way these functions are exported to users except on the way the simulation time is represented and on how simulation events are created and managed. In the these two aspects of the ns-3, Cisco packet tracer simulation runtime library. We first consider the representation of simulation time in section and then discuss how simulation events are created and managed.

The example below shows how this might be done:

Users can derive from the Event base class and provide their own version of the Expire method:

```
Class MyEvent: public Event {
virtual void  Expire (void) {
// do my thing here
} MyEvent * event = new MyEvent ( );
Schedule (Seconds (10. 0), event);
```

Discrete Time Event-Driven Simulations

This protocol requires the sender to deal with three types of events: DataReceived, AckReceived and AckTimeout. Whenever we receive data, we need to send back an ACK:

```
Class DataReceived: public Event {
StopAndWaitArqProtocol *m_src;
StopAndWaitArqProtocol *m_dst;
Data *m_data;
Virtual void Expire (void) {
AckReceived * ack = new AckReceived ( );
ack->m_src = m_dst;
ack->m_dst = m_src;
ack->m data = m_data;
Time delay = . . . ;
Schedule (delay, ack);
```

Whenever we receive an ACK, we need to cancel our ACK timeout and send the next packet present in our transmission queue:

```
class  AckReceived : public Event {
StopAndWaitArqProtocol *m_src;
StopAndWaitArqProtocol *m_dst;
Data *m_data;
Virtual void Expire (void) {
If (m_dst->m_currentData! = m_data) {
// we arrived too late?
} else {
Cancel (m_dst->ackTimeout);
Data _ next = m_dst->m_ txQueue->GetNext ( )
m_dst->Send (next);
```

}f we are not lucky and the ACK is not received before the ACK timeout expires, we need to retransmit our data:

```
class AckTimeout: public Event {
StopAndWaitArqProtocol *m_ sender;
virtual void Expire (void) {
m_sender->Send (m_protocol->m_ sender);
```

}Finally, to send data, we first start the ACK timeout and then create a receive event for the receiver.

```
class StopAndWaitArqProtocol {
friend class AckTimeout;
friend class AckReceived;
friend class DataReceived;
StopAndWaitArqProtocol *m_other;
Void Send (Data*data) {
m_currentData = data;
AckTimeout _ timeout = new AckTimeout ( );
Timeout->m sender = this;
Schedule (Seconds (1) , timeout ) ;
Time delay = . . . ;
DataReceived _ received = new DataReceived ( );
received->m_src = this;
```

```
received->m_dst = m_other ;
Schedule (delay, received);
}
```

The problem with this approach is that we are forced to split the entire protocol implementation in many subclasses which need to manipulate both states that is local to them and that is shared with the protocol instances communicating together

Conclusions and future work

This thesis has presented a system architecture for wireless sensor nodes that is capable of addressing the strict requirements of wireless sensor networks. By utilizing a single shared controller that is augmented by a collection of specialized hardware accelerators, the architecture is able to support flexible, application-specific communications protocols without sacrificing efficiency. This architecture has been validated through the development of three hardware platforms and a software operating system architecture that addresses key issues that arise when building a wireless sensor network device that must meet strict power consumption and size requirements. They include: flexibility, fine-grained concurrency, precise synchronization, and decoupling between RF and data path speed. The platform must be flexible enough to meet the wide range of application requirements that sensor networks are addressing. We have identified core application scenarios that range from environmental data collection, to security networks, to node tracking networks. Each scenario has substantially different communication patters and protocols that must be supported by single hardware architecture.

References:

1. Madden, S., et al., TAG: A Tiny AGgregation Service for Ad-Hoc Sensor Networks. 2002: OSDI.

2. Intanagonwiwat, C., R. Govindan, and D. Estrin, Directed Diffusion: A Scalable and Robust Communication Paradigm for Sensor Networks. 2000: Mobile Computing and Networking.

3.A Multi-Tiered Portable Wireless System for Monitoring Weather Conditions Krihsnamurthy, "The Platforms Enabling Wireless Sensor Networks," *Communications of ACM*, Vol. 47, No. 6, June 2006.

4. C. Hartung, R. Han, C. Seielstad, S. Holbrook, "FireWxNet in Wildland Fire Environments," *Mobisys* 2006

5.K. Lorincz, D. Malan, T. Fulford-Jones, A. Nawoj, A. Clavel, V. Shnayder, M. Welsh and S. Moulton, "Sensor Networks for Emergency Response: Challenges and Opportunities," *IEEE Pervasive Computing*, 2004

6. S. Kumar, V. Raghavan and J. Deng, “Medium Access Control for Ad-Hoc Wireless Networks : a Survey” AdHoc Networks, 4(3). 2006.

7. P. Naik and K. Sivalingam, “ A Survey of MAC Protocols for Sensor Networks, in Wireless Sensor Networks,

8. N.B. Priyantha, H. Balakrishnan, E. Demaine, S. Teller, "Mobile-Assisted Localization in Wireless Sensor Networks

9. B. Sundararaman, U. Buy, A. D. shemkalyani, "Clock synchronization for wireless sensor networks: a survey" Ad Hoc Networks 3 (2005) 281–323, January 2005

10. J. Kulik, W. Henzelman, and H. Balakrishnan, "Negotiation-Based Protocols for Disseminating Information in Wireless Sensor Networks," *Wireless Networks*, Vol. 8, 2002

11. Q. Cao, T. Abdelzaher, T. He and J. Stankovic, "Towards Optimal Sleep Scheduling in Sensor Networks for Rare Events Detection," Proc. ISPN’05, 2005