Protocol Design, Testing, and Diagnosis towards Dependable Wireless Sensor Networks

XIONG, Junjie

CSE, CUHK
Supervisor: Michael R. Lyu, Evangeline F.Y. Young
July 11, 2012
Outline

- Introduction to wireless sensor networks
- Part 1: An Efficient MAC Protocol Design
- Part 2: Reliable Protocol Conformance Testing
- Part 3: Mobility-assisted Diagnosis
- Conclusions
Introduction: Wireless Sensor Networks (WSNs)

- Application-oriented
  - Surveillance
  - Target tracking
- Resource-constrained sensor nodes
  - E.g. Energy unit: two AA batteries
  - E.g. RAM: 8k bytes for Iris
- Capable of self-organizing
- Subjected to dynamic changes
Reference Architecture of A Sensor Node

Protocol stack

- Application Layer Protocol
- Transport Layer Protocol
- Network Layer Protocol
- Media Access Layer Protocol
- Physical Layer Protocol

Operating System: Contiki & TinyOS

Other Related Protocols & Software Support

Hardware
Introduction: Thesis Scope

Towards successful WSN applications: the development, deployment, and maintenance

Software Design

Implement software

System Testing

Deploy application

Application Maintenance

MAC layer protocol

Chapter 2 & 3
Correct protocol implementation

Network layer protocol

Application layer protocol

Chapter 4
Protocol testing

Chapter 5
Protocol diagnosis

Efficiency improvement

No intrusion & cost-efficient

Simulation

Testbed

Program update

...
Part I:  
An Efficient MAC Protocol Design
Background

- Underwater acoustic sensor networks (UWASNs)
  - WSNs deployed in the water
  - Wireless medium: sound

- Difference from terrestrial wireless sensor networks (TWSNs)
  - Longer latency
  - Higher cost
  - Sparser deployment
Background

- An ocean bottom surveillance example of UWASNs
Motivations

- UWASNs VS TWSNs

A DATA transmission in UWASNs with CSMA/CA

A DATA transmission in TWSNs with CSMA/CA
Motivations

Simultaneous data transmissions: collision or not?
Motivations

- Use parallel transmissions
- Throughput and delay performance improvement with a compact schedule

Data transmission between 3 nodes
Scheduling Element

- The scheduling element & scheduling problem in UWASNs is very different from that in TWSNs

DATA1: P -> Q

Node P

Node Q

Node S

One time

Three times

Interference

Data transmission between 3 nodes in UWASNs
A Routing and Application based Scheduling Protocol (RAS)

- RAS components towards compact schedule
  - TDMA based MAC mechanism
  - Utilize static routing & application data direction information
  - Centralized schedule calculation
    - Calculate the traffic of each node
    - Schedule the traffic receptions and transmissions
Congestion Avoidance Algorithm of RAS

Towards better queue utilization and fairness with priority scheduling -> higher priority to nodes with heavier traffic

- **Step 1**: Schedule the BS's data receptions from 1 hop nodes
- **Step 2**: Schedule the data receptions tier by tier: from inner tier to outer tier
- **Step 3**: For data receptions from the same tier, arrange them alternatively
Performance Evaluation

- Simulation settings under NS-3 (network simulator 3)
  - Networks of 6 different sizes: from 9-node to 64-node
  - Nodes are randomly distributed and connected
  - Maximum hop distance range: 1- 7 hops

- In comparison with UW-FLASHR: a distributed TDMA based MAC protocol that utilizes propagation delay to increase throughput
Performance Evaluation

- Schedule length for RAS: scalable

![Graph showing schedule length for RAS and lower bound with varying numbers of nodes. The x-axis represents the number of packets each node generates in a cycle, and the y-axis represents the schedule length with time slots as the unit.]
Performance Evaluation

- Throughput

![Graph showing throughput vs. traffic rate for RAS and UW-FLASHR with different node counts.]
Performance Evaluation

- Average end-to-end delay

![Graph showing average end-to-end delay vs. traffic rate](image-url)
Performance Evaluation

- Average maximum queue length per node

![Graph showing average maximum queue length per node vs traffic rate of each node (kbps)]
Contributions of Part 1

- Design a priority scheduling protocol to provide efficient communications for UWASNs
  - Allow parallel transmissions, and thus improve the throughput and delay performance
  - Mitigate queue overflow and scalable in calculating proper schedules
Part 2:
Reliable Protocol Conformance Testing
Motivations

- Experiences from real deployments show that protocol implementations are prone to software failures
  - A three-day network-outage on a volcano deployment: a bug in the routing protocol Deluge
  - Sporadic packet loss on all GSM nodes in the Swiss Alps deployment: a bug in the GPRS drivers of the BS
- Very expensive and difficult to fix the bugs after deployment
Related work

Current main methods in tackling the software bugs in WSNs

- Simulation: different from real execution (Li & Regehr, 2010; Sasnauskas et al., 2010)
- Testbeds: designed for network performance evaluation rather than for software bug detection
- Large-scale real deployment: expensive

We are the first to use a small number of real sensor nodes to mimic large-scale WSNs and test the protocol implementation against the specification -> RealProct
Challenges

- Sensor node is **difficult to control** than a PC
  - Limited CPU and inconvenient interface
- How to test the protocol with **various** topologies and events with only **a few** real sensor nodes
- Volatile wireless environment will lead to random **packet loss**, and cause problems in testing
RealProct Solutions to the Challenges

- An architecture that enables testing with real sensor nodes
- Topology virtualization and event virtualization
- Dynamic Test Execution
Background

- Protocol conformance testing (PCT) process
  - IUT (Implementation Under Test)
RealProct Architecture

- SUT (System Under Test)

Tester executes test cases
Topology Virtualization

- Use the tester to virtualize a 3-node topology for SUT

Packet 1
Addr1
SUT
Addr2
Packet 2
Addr3
Tester

Content of Packet 1:
Sender address is Addr1.

Content of Packet 2:
Sender address is Addr2.
The sender has a neighbor with Addr3.
Event Virtualization

- Use the tester to create a packet disorder event at the SUT

![Diagram showing packet disorder event]

- Neighboring nodes are connected by dotted lines.
Reason to Use Dynamic Test Execution

Suppose packet loss probability is $L_0$, a test case is executed $n$ times, and it passes $n_1$ times and fails $n_2$ times

- If $n_1 > n_2$, then declare as **pass**, calculate the FN (false negative) probability
- If $n_1 < n_2$, then declare as **fail**, calculate the FP (false positive) probability
Dynamic Test Execution

- To guarantee that the FN and FP error rates are lower than a required value, first calculate the minimum count to execute each test case.
- The actual execution times are dynamic:
  - Repeat the test case execution until its FN and FP error rates are satisfied.
Performance Evaluation

• Equipment: two real TelosB sensor nodes and a PC
  ◦ Tradeoff between simulation and large-scale deployment
  ◦ First to find two new bugs that the developers added into their bugzilla
Performance Evaluation

- Protocols tested in OS Contiki 2.4: μIP TCP/IP protocol and Rime mesh routing protocol for WSNs
- Two new bugs found in μIP TCP/IP and previous bug repetition
  - Bug 1 & 2 – Connect to opened & unopened TCP ports
  - Bug 3 – SYN/ACK packet loss
  - Bug 4 – SYN packet duplication
Performance Evaluation

- Bug 1 (new) – Connect to opened TCP ports
  - Test opened port 0 (within 0 to 65535)

TCP client: 1025  TCP server: 80

- Sending SYN
  - [SYN]seq=0

- SYN/ACK received, sending ACK
  - [ACK]seq=1,ack=1

- SYN received, sending SYN/ACK

ACK received, connection established
Bug 1 – Client (Tester) connects to opened TCP port 0 of Server (SUT)

Bug: Client expects SYN/ACK response while it receives no reply

Test result. Bug: Server does not reply correctly when its port 0 receives SYN.
Contributions of Part 2

- As a protocol testing tool with real sensor nodes, RealProct finds two new bugs, repeats previously detected bugs in the TCP/IP stack of WSNs

- Propose two techniques, topology virtualization and event virtualization, for testing

- Design an algorithm to tackle the inaccuracy problem caused by non-deterministic events in test execution
Part 3: Mobility-assisted Diagnosis
Motivations and Related Work

- **Truth**: despite extensive testing, bugs still sneak into real deployment
  - In-situ diagnosis in real-time failure detection

- **Implant diagnosis agents into each sensor node** (Ramanathan et al., 2005; Liu et al., 2011; Miao et al., 2011)
  - Many already-deployed WSNs are not facilitated with the agents
  - Intrude the WSN application
  - Insert agents at all protocol layers: inefficient

- **Deploy another network to monitor the WSN** (Khan et al., 2007)
  - Inefficient
  - Costly
Overview of Our Solution: MDiag

First to propose a mobility-assisted diagnosis (MDiag) approach to detect failures by patrolling WSNs with smartphones

- Mobile smartphones are increasingly popular
- Not intrude the WSN applications during the patrol
  - smartphones collect and analyze packets sent from sensor nodes
- Able to collect raw packets (contain header information in all protocol layers) of all types, MDiag frees us from inserting agents at all the protocol layers
- On-demand diagnosis without deploying another monitoring network
A Diagnosis Scenario of MDiag
Challenges

- How to determine the abnormal behaviors from the collected various kinds of raw packets
- How to design the patrol method to increase the failure detection rate
Background: Network Architecture

- A WSN with a BS and static sensor nodes deployed for monitoring applications
- The smartphone is able to receive the packets sent from the sensor nodes as long as
  - equipped with the same reception device as the sensor nodes
  - or attached with a sensor node for snooping purpose only
- We discuss the case of using one smartphone to patrol
MDiag Framework

• Three steps

1. Raw packets collected by the smartphone
2. Statistical rules on packet analysis
3. Statistical results on raw packets

Packet decoder

Patrol set of K sensor nodes

Algorithm of patrol set selection

Packet sequence

WSN developers (or specifications)

Problem report

Step 1

Step 2

Step 3
Statistical Rules on Packet Analysis

In the statistical results, the following fields are analyzed by the statistical rules:

- Packet type
- Packet count of each type
- Packet directions
- Neighbor information
- Packet value, e.g., data content of an application data packet
Statistical Rules on Packet Analysis

- Not applicable to analyze a single packet process, e.g., a random TCP packet loss
- Based on the targets of all protocol layers
- In aspect of completeness:
  - More complete than Sympathy (employs only one rule)
  - A subset of the specification-based rules
Coverage-oriented Smartphone Patrol Algorithms

- The patrol approach should try to cover all the sensor nodes in the WSN
- The problem is the patrol set selection rather than the patrol path design
  - The cost during the travel is not considered
Coverage-oriented Smartphone Patrol Algorithms: Naïve Method (NM)

- The smartphone visits all the sensor nodes one by one
  - Long time
  - Low failure detection
Coverage-oriented Smartphone Patrol Algorithms: Greedy Method (GM)

Utilizing the broadcast nature of wireless communications, the smartphone visits several sensor nodes, but is able to cover all the sensor nodes.
Coverage-oriented Smartphone Patrol Algorithms: Greedy Method (GM)

The smartphone always selects to visit the sensor node with the largest degree.

- Degree(v): sensor node v’s neighbor count
- Snooping efficiency (SE) of v: degree(v)
- SE of a patrol set S with K sensor nodes: average of the K sensor nodes’ SE

Aim at improving patrol set snooping efficiency
Not the minimum set cover problem
Part 3: Coverage-oriented Smartphone Patrol Algorithms: Maximum Snooping Efficiency Patrol (MSEP)

- MSEP is better than GM
  - Cover every sensor node
  - Enhance the patrol set snooping efficiency by reducing small degree node selection probability

![Diagram showing comparison between GM and MSEP patrol sets]
Performance Evaluation: Settings

- **Real experiments and emulations**
- An existing data collection application with routing protocol CTP and X-MAC protocol
- Use *real* failures encountered in our experiments and also failures found in the code repositories of OS Contiki
- Besides NM and GM, implement a baseline method called RM-K to compare with MSEP
Performance Evaluation: Permanent Failure Detection

- A rule: **X-MAC** protocol behaviors between a pair of communicating sensor nodes
- Rule is violated: performance *degradation* failure
- **Not** noticeable at the application layer
- **Cannot** be detected by agent approach Sympathy

MDiag collects raw packets of all types!
Performance Evaluation: Permanent Failure Detection

- **Surprising** reason: a ‘printf’ statement in the WSN application program
  - Trigger serial port **interrupts**: consume a lot of CPU **resources**
  - CPU is too busy to handle packet transmissions and receptions
Performance Evaluation: Short-term Failure Detection

Short-term failure: routing fluctuation after reboot

- Routing fluctuation -> using each other to forward data -> bidirectional data exchange -> abnormal case (AC)
- Disobey a rule on routing behaviors
- Lasting time is short-term: patrol approaches matter

I forward data to you

I forward data to you
Performance Evaluation: Short-term Failure Detection

- **Topology**
- **Due to no initialization** of the routing value
  - For the BS: initialized as 0
  - For the other sensor nodes: should be a maximum value (in fact 0)

Reboot at 600s
Performance Evaluation: Short-term Failure Detection

Abnormal case (bidirectional data exchange)

- Short & long
- Frequent & infrequent

R represents a datum in the opposite direction of a datum D.
A frequent AC: DRDRDRDRDR
An infrequent AC: DDDDDR MMM
Performance Evaluation: Short-term Failure Detection

Detection probability of abnormal case 1 (long and frequent)
Performance Evaluation: Short-term Failure Detection

Detection probability of abnormal case 2 (long but infrequent)
Performance Evaluation: Short-term Failure Detection

Detection probability of abnormal case 4, 5, 6, and 7 (short)
Detection probability of all ACs

Performance Evaluation: Short-term Failure Detection

![Graph showing detection probability over working time of a smartphone]
Contributions of Part 3

• Propose a mobility-assisted diagnosis method called MDiag:
  ◦ **Not intrude** the WSNs
  ◦ **More efficient** than deploying another network for diagnosis purpose
  ◦ Able to snoop all kinds of raw packets, it can help **find more failures**

• Design statistical rules to guide the abnormal phenomena determination

• Propose MSEP algorithm to improve the detection rate and reduce the patrol time of MDiag
Thesis Scope Review

Software Design → Implement software → System Testing → Deploy application → Application Maintenance

MAC layer protocol

Network layer protocol

Application layer protocol

Chapter 2 & 3
Correct protocol implementation

Chapter 4
Protocol testing

Chapter 5
Protocol diagnosis

Simulation

Testbed

No intrusion & cost-efficient

Program update

Efficiency improvement

...
Conclusions

- Design a priority scheduling protocol RAS to provide **efficient** communications for UWASNs
- Design a protocol conformance testing tool RealProct with **real** sensor nodes for **correct protocol implementation**
- Propose a protocol diagnosis method MDiag to diagnose the deployed WSNs **efficiently without intrusion**
Thank you!

Q & A
Appendix
Introduction: Sensor Nodes

- Power supply unit
- Sensors
- ADC (analog-to-digital converter)
- Microprocessor
- Radio transceiver
- Storage

![Diagram of a sensor node with components labeled: Sensing element, Processing element, Communication, Sensors, ADC, Microprocessor, Radio, Storage (RAM, Flash Memory, EEPROM), Power supply unit (Batteries).]

Part 1: Motivations

Data transmission between 3 nodes in UWASNs

Node P

DATA2: Q→S

Node Q

Receptions

DATA3: S→Q

Node S

Transmissions

Interferences

Data transmission between 3 nodes in UWASNs
Part 1: Motivations

- Use parallel transmissions
- Throughput and delay performance **improvement** with a **compact schedule**

Data transmission between 3 nodes
Part 1: Scheduling Principles

- At a node, guarantee a DR will not overlap any DT
- At a node, guarantee a DR will not overlap any IR
- At a node, a DT and one or more IR can coexist
- No DR from i-th hop node to (i+1)-th hop node
- At a node, use DR as the scheduling basis rather than DT or IR

**DR**: data reception
**IR**: interference reception
**DT**: data transmission
Part 1: RAS Cycle

RAS cycle
## Part 1: Parameters for Data Transmissions

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data Rate</td>
<td>10 kbps</td>
</tr>
<tr>
<td>Data Packet Size</td>
<td>100 bytes</td>
</tr>
<tr>
<td>Control Packet Size</td>
<td>10 bytes</td>
</tr>
<tr>
<td>Transmission Range (communication range)</td>
<td>1500 m</td>
</tr>
<tr>
<td>Interference Range</td>
<td>3500 m</td>
</tr>
<tr>
<td>Average Distance between Two Nodes</td>
<td>1110 m</td>
</tr>
<tr>
<td>Guard time</td>
<td>20 ms</td>
</tr>
<tr>
<td>Wireless model</td>
<td>TwoRayGround</td>
</tr>
</tbody>
</table>
Part 1: UW-FLASHR

- **UW-FLASHR**
  - is a distributed TDMA based MAC protocol.
  - utilizes propagation delay to increase throughput.
  - employs no energy-saving mechanism.
  - suffers from collisions.
RAS Protocol Design

- Scheduling algorithm formulation
  - Schedule the combination of a DR, a DT and a sequence of IR to the other nodes

\[ Q_{mj} = \sum_w Q_{mjw} \]

\[ E_{mC_{mj}} = \left\{ \begin{array}{l} DT_{C_{mj}}, \\
DR_{C_{mj}}, \\
IR_{SC_{mj}}, \\
\end{array} \right. \]

\[ E_{mC_{mj}} = \left\{ DT_{C_{mj}}, \right. \]

\[ DR_{C_{mj}} = \frac{DIS(m, C_{mj})}{SOUND\_SPEED}, \]

\[ DR_{C_{mj}} = \frac{DIS(m, C_{mj})}{SOUND\_SPEED}, S_{C_{mj}} \]

is the set of nodes which are within the node \( C_{mj} \)'s interference range, and \( m \in S_{C_{mj}}, j = 1, 2, \ldots, K_m. \)

Set \( DR_{C_{mj}} \) to 0

\[ R_{mC_{mj}Q_{mjw}} \]
RAS Protocol Design

- Scheduling algorithm formulation
  - Schedule according to the principles

\[
\begin{align*}
\min \max R_{mc_{mj}Q_{mjw}}, \text{ for } j = 1, 2, \ldots, K_m, \sum_w Q_{mjw} &= Q_{mj} \text{ and } m \in S. \\
\text{s.t. } \begin{cases}
R_{mc_{mj}Q_{mjw}} + E_{mc_{mj}} \cdot DR_{cm_{mj}} > R_{zc_{zj}Q_{zjw}} + E_{zc_{zj}} \cdot DT_{cm_{zj}} + D_{DATA}, \text{ or} \\
R_{mc_{mj}Q_{mjw}} + E_{mc_{mj}} \cdot DR_{cm_{mj}} + D_{DATA} < R_{zc_{zj}Q_{zjw}} + E_{zc_{zj}} \cdot DT_{cm_{zj}}, \\
\text{for } C_{zj} = m, z \in S', \\
R_{mc_{mj}Q_{mjw}} + E_{mc_{mj}} \cdot DR_{cm_{mj}} > R_{zc_{zj}Q_{zjw}} + E_{zc_{zj}} \cdot IR_{m} + D_{DATA}, \text{ or} \\
R_{mc_{mj}Q_{mjw}} + E_{mc_{mj}} \cdot DR_{cm_{mj}} + D_{DATA} < R_{zc_{zj}Q_{zjw}} + E_{zc_{zj}} \cdot IR_{m}, \\
\text{for } z \in S'.
\end{cases}
\end{align*}
\]

\[\begin{align*}
R_{mc_{mj}Q_{mjw}} > R_{mc_{mj}Q_{mjw'}}, \text{ or } R_{mc_{mj}Q_{mjw}} < R_{mc_{mj}Q_{mjw'}}, \text{ for } w \neq w'. \\
R_{mc_{mj}Q_{mjw}} > R_{mc_{mj}Q_{mj'w'}}, \text{ or } R_{mc_{mj}Q_{mjw}} < R_{mc_{mj}Q_{mj'w'}}, \text{ for } j \neq j'. \\
R_{mc_{mj}Q_{mjw}} > R_{mc_{mj}Q_{mj'w'}}, \text{ or } R_{mc_{mj}Q_{mjw}} < R_{mc_{mj}Q_{mj'w'}}, \text{ for } m \neq m'. \\
R_{mc_{mj}Q_{mjw}} + E_{mc_{mj}} \cdot DT_{cm_{mj}} \geq 0. \\
S' = \{z : \text{the set of nodes } z \text{ that have been scheduled}\}
\end{align*}\]
Part 1: Advantages of RAS

- Reduces mutual communications
- Reduces energy consumption
- Avoids collision, increases throughput, and reduces delay and queue overflow probability for each node

Cycle comparison between UW-FLASHR and RAS
Part 1: RAS Protocol at the BS

**Algorithm 1** RAS protocol at the BS

1. Load node position information
2. Calculate distance between any two nodes
3. Calculate all nodes’ hop distance to the BS
4. Calculate the number of data to be transmitted and received at each node
5. CalcSchedule() /* Algorithm 3*/
6. The BS broadcasts the routing table and the schedule to all its children with high power
Part 1: Congestion Avoidance Algorithm

Algorithm 3 CalcSchedule() function at the BS

1: \( Parent = BS; \ hop = 1 \)
2: \( \textbf{while} \ hop \leq \text{maxhop} \ \textbf{do} \)
3: \( \quad \textbf{while} \ Parent \ has \ children \ \textbf{do} \)
4: \( \quad \quad \textbf{while} \ Parent \ has \ data \ to \ receive \ from \ its \ children \ \textbf{do} \)
5: \( \quad \quad \quad \textbf{if} \ Parent \ is \ idle \ in \ the \ Slot \ \textbf{then} \)
6: \( \quad \quad \quad \quad Parent \ searches \ its \ entire \ children \ set \ to \ alternatively \ find \ a \ child \)
7: \( \quad \quad \quad \quad \text{whose \ transmission \ results \ in \ its \ reception \ at \ the \ Slot} \)
8: \( \quad \quad \quad \textbf{if} \ Parent \ finds \ a \ suitable \ child \ \textbf{then} \)
9: \( \quad \quad \quad \quad \text{schedule \ the \ child's \ transmission \ and \ the \ related \ reception \ and} \)
10: \( \quad \quad \quad \quad \text{interference} \)
11: \( \quad \quad \quad \text{break \ searching} \)
12: \( \quad \quad \textbf{end \ if} \)
13: \( \quad \textbf{end \ if} \)
14: \( \quad Parent \ fetches \ the \ next \ Slot \ for \ reception \)
15: \( \textbf{end \ while} \)
16: \( \text{fetch \ the \ next} \ Parent \ \text{to \ schedule \ reception} \)
17: \( hop = hop + 1 \)
18: \( \textbf{end \ while} \)
Performance Evaluation

- Schedule ratio: the lower bound schedule length ($L_2$) divided by the RAS schedule length ($L_1$)
Part 2: RealProct Architecture

SUT (System Under Test)

Point of Control & Observation

Upper Tester

Lower Tester

Diagram showing the architecture with SUT, Lower Tester, Upper Tester, Point of Control & Observation, and various components and layers.
Part 2: Generality of RealProct

RealProct provides a generic framework and universal techniques to keep the testing process the same and easy to follow:

- Design abstract test cases according to protocol specification.
- Translate the abstract cases into executable ones with the virtualization techniques.
- The PC downloads each test case into the tester (a sensor node) in real-time to execute.
- Control the execution times are with the dynamic test execution algorithm.
- Repeat the failed test cases to help debug.
Part 2: Performance Evaluation

- **Bug 2 (new)** – Client (Tester) connects to unopened TCP port 0 of Server (SUT).

Bug: Client expects RST response while it receives no reply.

Test result: Bug: Server does not reply RST when it should do so.
Part 2: Codes that Cause Bugs

```c
1 // Make sure that the TCP port number is not zero.
2 if (BUF -> destport == 0 || BUF -> srcport == 0)
3 {
4     UIP_LOG("tcp: zero port.");
5     goto drop;
6 }
```
Part 2: Repeat Bug – SYN Packet Loss

TCP client: 1025

Sends SYN

TCP server: 80

/SYN/seq=0

SYN received. Server sends SYN/ACK

/[SYN/ACK]seq=0,ack=1

Pretends that SYN/ACK is lost. Client sends SYN again.

/[SYN]seq=0

SYN received, Bug: Server replies ACK

/[ACK]seq=0,ack=1

Should be an SYN/ACK
Part 2: Repeat Bug – SYN Packet Duplication

TCP client: 1025

Sends SYN

TCP server: 80

[SYN]seq=0

SYN received. Server sends SYN/ACK

[SYN/ACK]seq=0,ack=1

ACK received and connection established.

[ACK]seq=1,ack=1

Duplicate SYN received

Bug: Server sends SYN/ACK

[SYN/ACK]seq=0,ack=1

Should be an empty ACK
Part 2: Dynamic Test Execution

1: Calculate $n_{min} = \lceil \log_{L_0} F \rceil$, $n = n_1 = n_2 = 0$
2: while $n \leq n_{min}$ do
3: Execute the test case
4: if Execution result is pass then
5: $n_1++$
6: else
7: $n_2++$
8: end if
9: end while
10: loop
11: if $n_1 > n_2$. then
12: Calculate $P(FN) = \binom{n}{n_1} L_0^{n_1}(1 - L_0)^{n_2}$
13: if $P(FN) \leq E$ then
14: break //end test execution
15: else
16: Execute the test case and increase $n_1$ or $n_2$ according to the result
17: end if
18: else if $n_1 < n_2$, then
19: Calculate $P(FP) = \binom{n}{n_2} L_0^{n_2}(1 - L_0)^{n_1}$
20: if $P(FP) \leq E$ then
21: break //end test execution
22: else
23: Execute the test case and increase $n_1$ or $n_2$ according to the result
24: end if
25: else if $n_1 = n_2$ then
26: Execute the test case and increase $n_1$ or $n_2$ according to the result
27: end if
28: end loop
Part 3: Background - Failure Classification

- **Transient failure**: lasts for a very short period
  - E.g., random packet loss
- **Short-term failure**: lasts only for a longer period
  - E.g., routing failure and link failure
- **Permanent failure**: stays until fixed or for a very long period
  - E.g., node crash and incorrect resource allocation failure
Packet Decoder Input

- Input: raw packets
  - From the radio frequency chip
  - Of various types: e.g., routing packets and application data packets
  - Help find more failures than agent approaches that do not insert agents at all the protocol layers
Packet Decoder Output

- Output: statistical results for the failure report

For sensor node $W_1$,
$W_1$ has neighbors:
A, B, C, ……
$W_1$ has sent out the following types of packets to neighbor A:
XMAC STROBE count: X
XMAC STROBE_ACK count: Y
XMAC DATA count: Z
XMAC DATA_ACK count: U
Routing ANNOUNCEMENT count: V
……
$W_1$ has send out the following types of packets to neighbor B:
……
For sensor node $W_2$,
……
An Example of the Statistical Rules

- For the data gathering application with routing protocol CTP and MAC protocol X-MAC:

<table>
<thead>
<tr>
<th>Layer</th>
<th>Statistical rules</th>
</tr>
</thead>
<tbody>
<tr>
<td>Application layer</td>
<td>Rule 1. For BS, ( Z ), the number of application data sent out, is 0.</td>
</tr>
<tr>
<td>Application layer</td>
<td>Rule 2. For sensor nodes other than BS, ( Z ) is within the application requirement.</td>
</tr>
<tr>
<td>Routing layer</td>
<td>Rule 3. For BS, the legal routing metric is 0.</td>
</tr>
<tr>
<td>Routing layer</td>
<td>Rule 4. For sensor nodes other than BS, routing metric value is legal.</td>
</tr>
<tr>
<td>Routing layer</td>
<td>Rule 5. For sensor nodes other than BS, no bidirectional data exchange exists, i.e., ( Z \times U = 0 ).</td>
</tr>
<tr>
<td>MAC layer</td>
<td>Rule 6. For sensor nodes other than BS, the number of each kind of MAC packets sent is normal, i.e., ( Y \sim U, X \succ \gg Z ).</td>
</tr>
</tbody>
</table>
Coverage-oriented Smartphone Patrol Algorithms: Maximum Snooping Efficiency Patrol (MSEP)

- Cover every sensor node
  - first find i, the sensor nodes with the minimum degree.
- Enhance the patrol set snooping efficiency by reducing small degree node selection probability
  - elect a sensor node j with the largest degree from i’s neighbor set
Part 3: Experiment Settings

  - It allows smartphones to be connected to a sensor node.
Part 3: Performance Evaluation: Short-term Failure Detection

Detection probability of AC 3 (a long and frequent AC)

Patrol set size
NM: 25
GM: 10
MSEP: 7
RM-7: 7
RM-10: 10

Patrol time
NM: 625
GM: 260
MSEP: 180
RM-7: [0,625]
RM-10: [0,625]