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

XIONG, Junjie

CSE, CUHK

Supervisor: Michael R. Lyu, Evangeline F.Y. Young

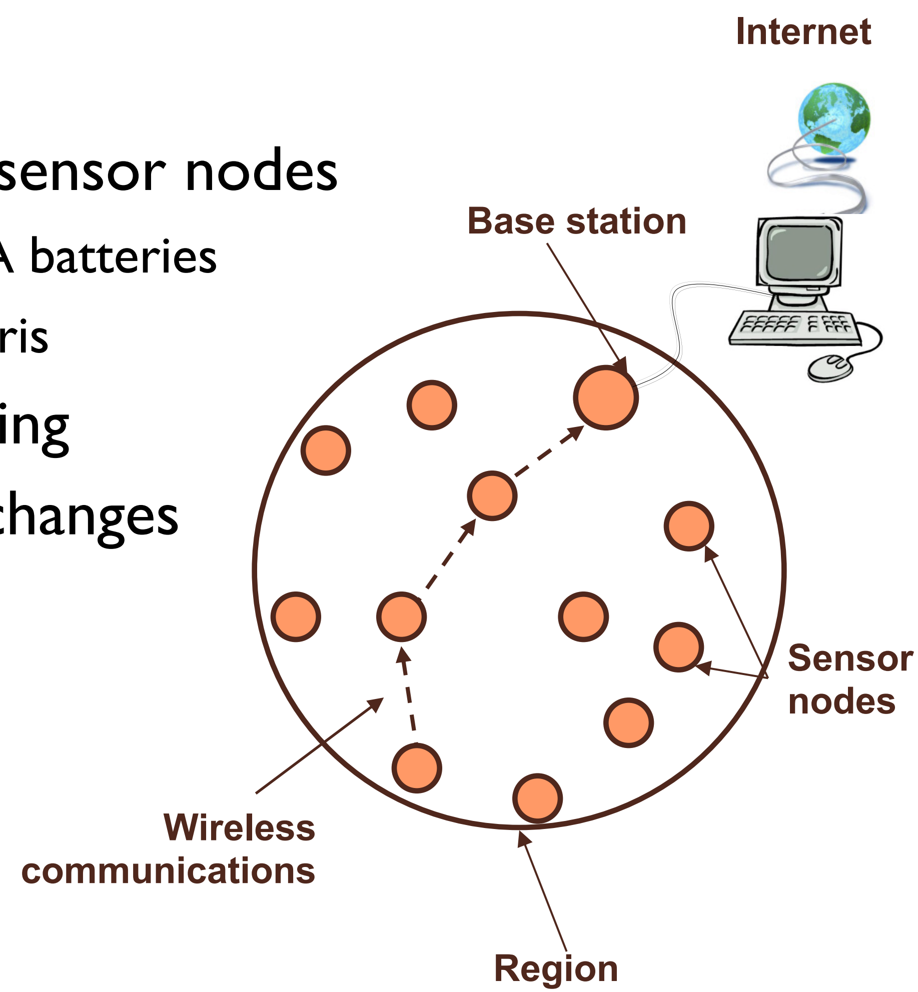
July 11, 2012

Outline

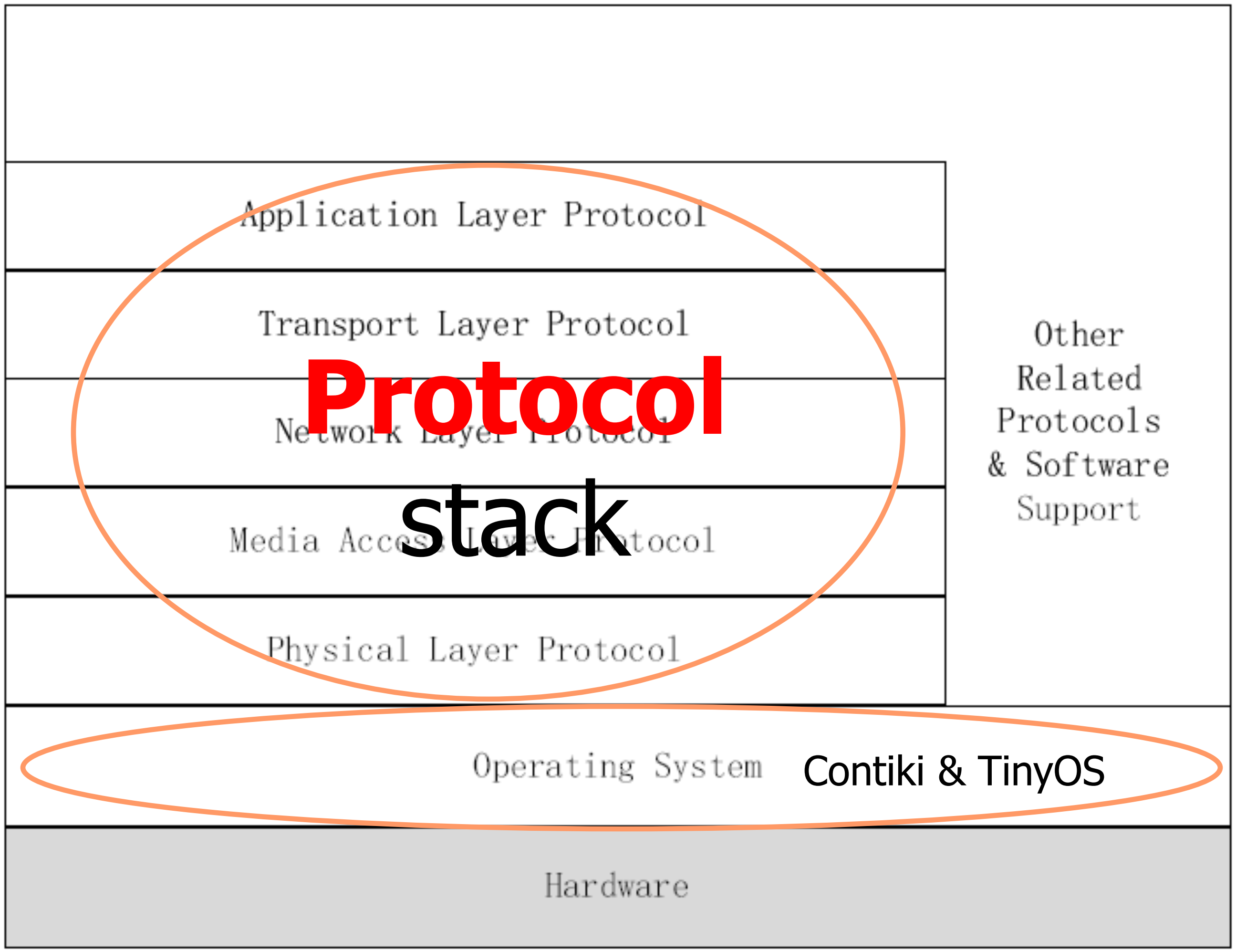
- Introduction to wireless sensor networks
- Part 1: An Efficient MAC Protocol Design
- Part 2: Reliable Protocol Conformance Testing
- Part 3: Mobility-assisted Diagnosis
- Conclusions

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

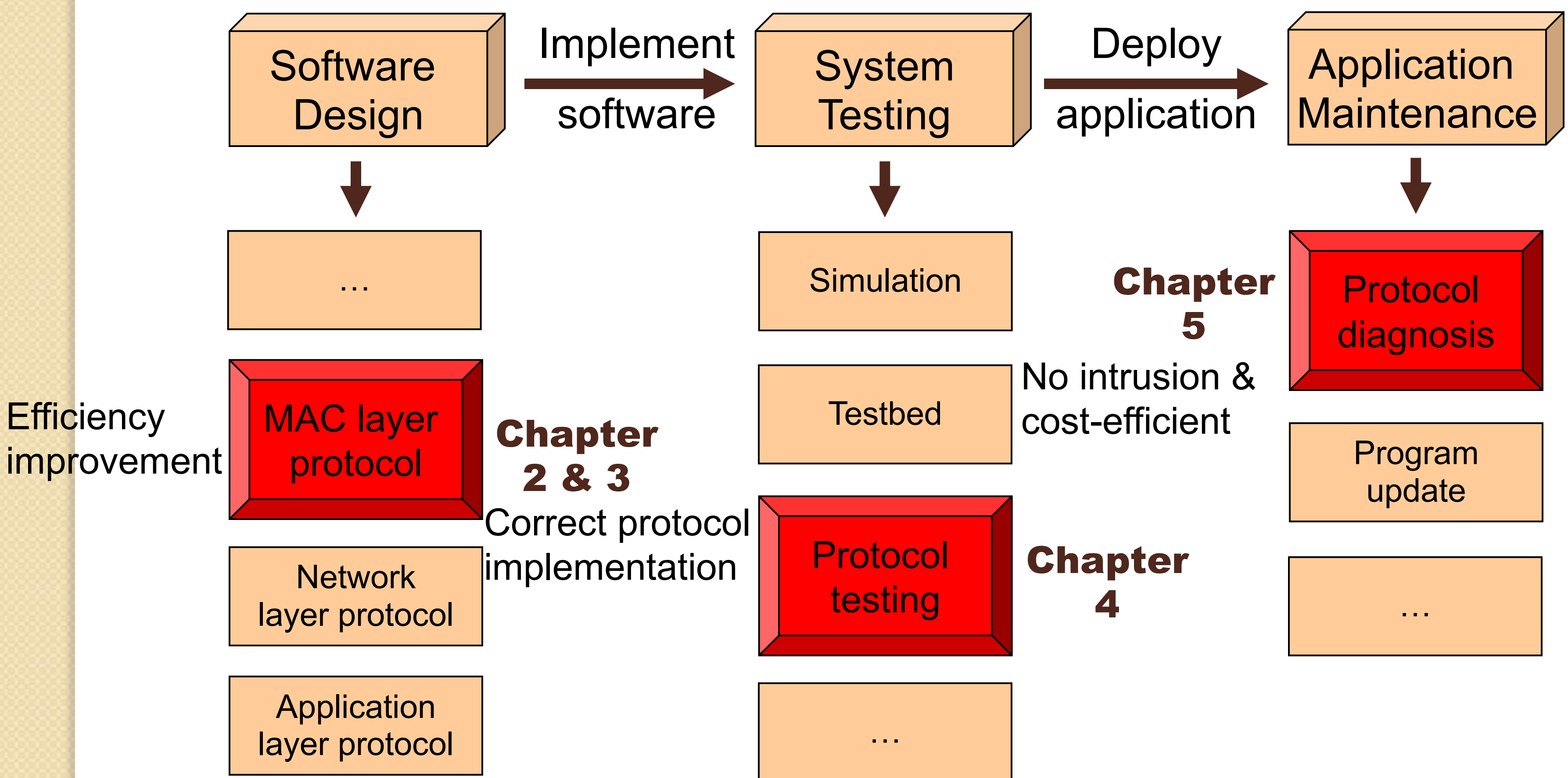


Reference Architecture of A Sensor Node



Introduction: Thesis Scope

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



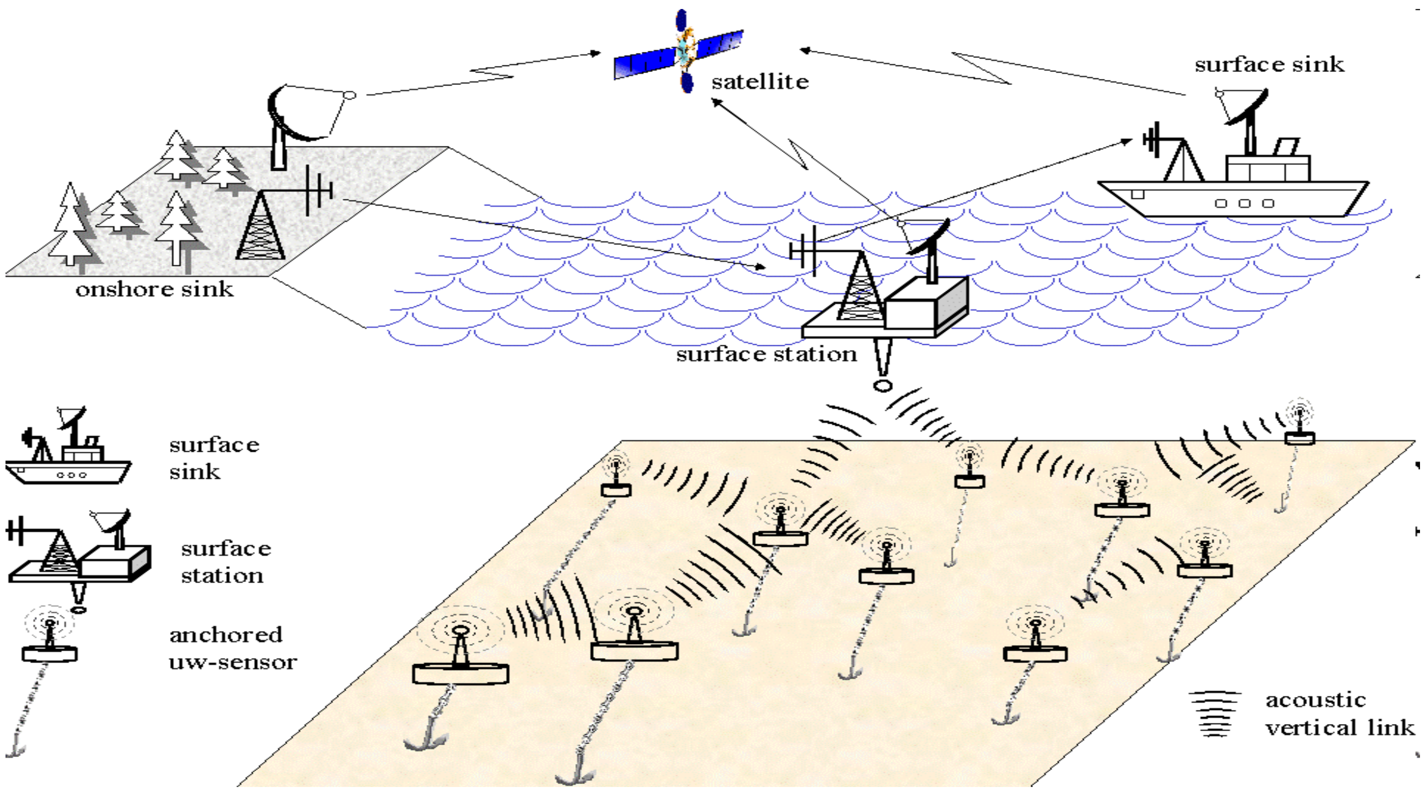
Part 1:
An Efficient MAC Protocol Design

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

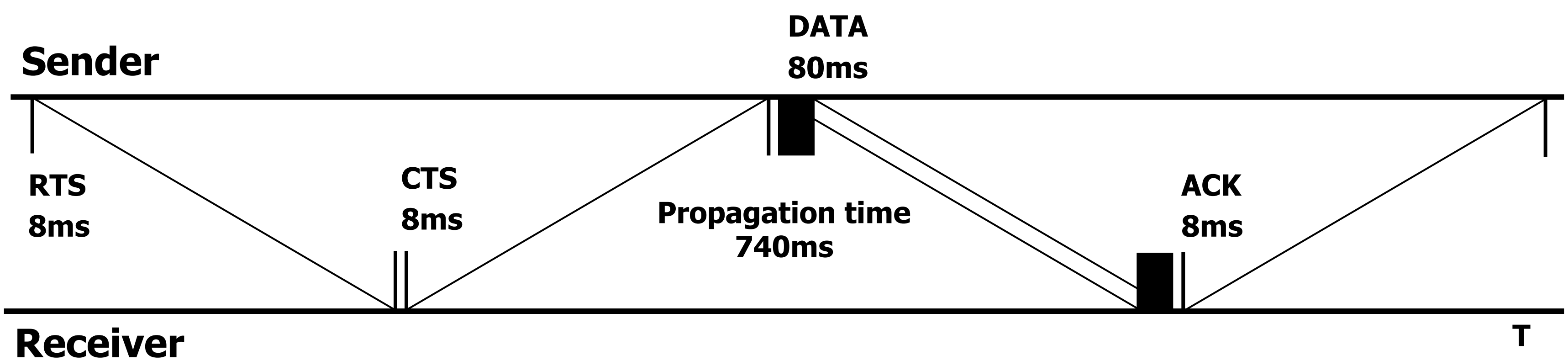
Background

- An ocean bottom surveillance example of UWASNs

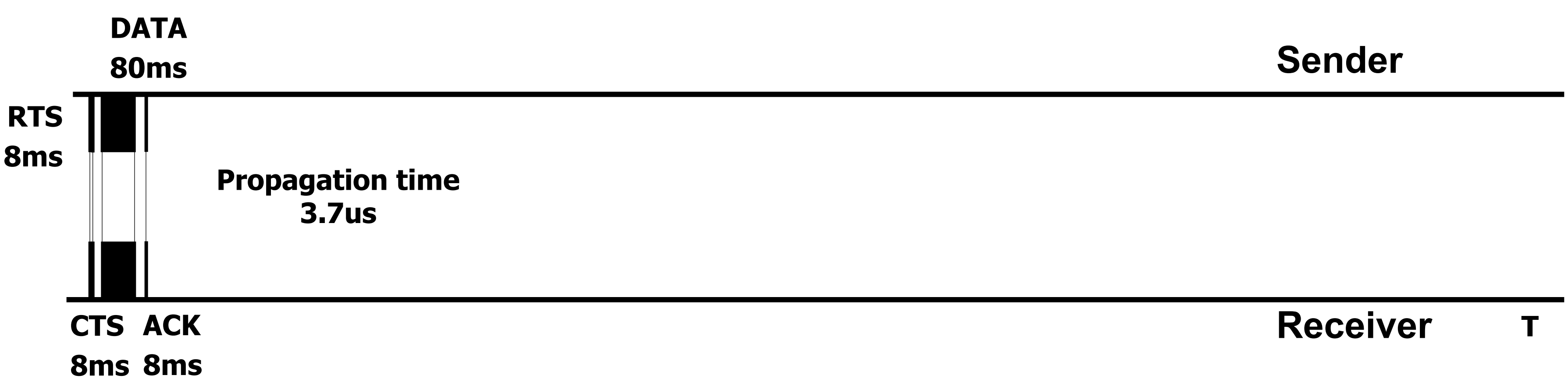


Motivations

- UWASNs VS TWSNs



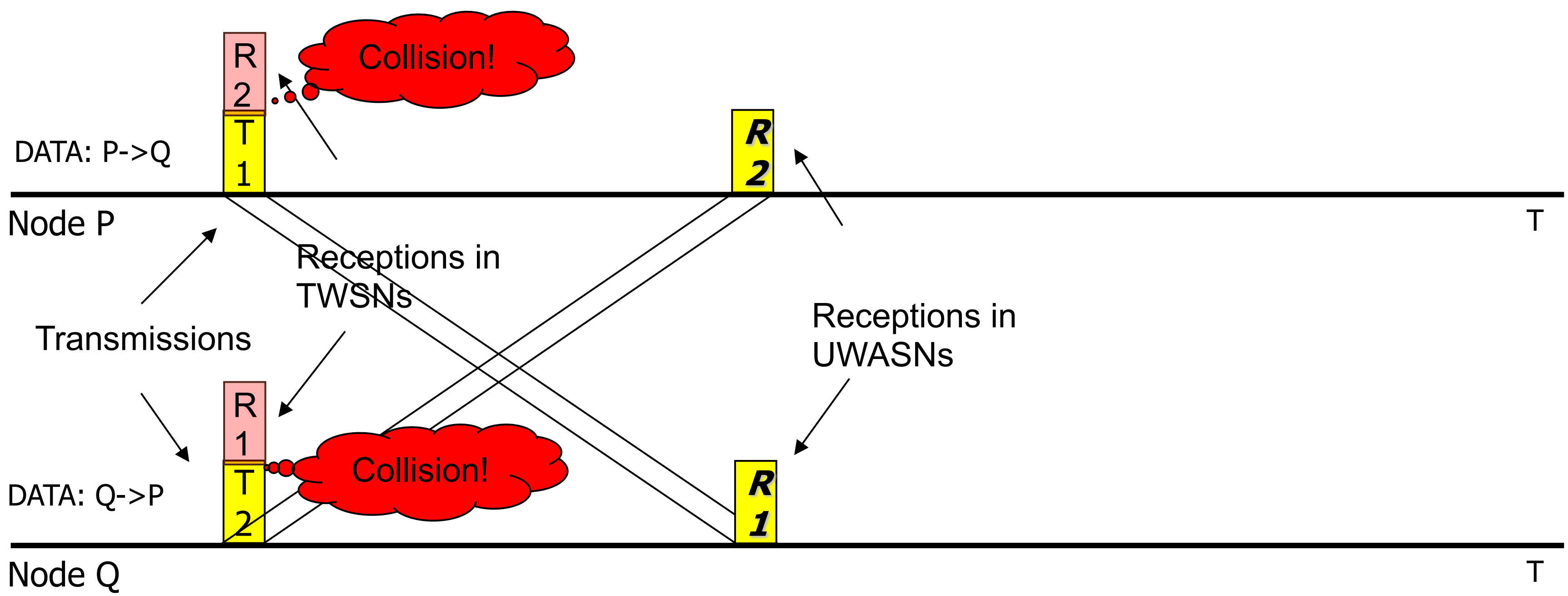
A DATA transmission in UWASNs with CSMA/CA



A DATA transmission in TWSNs with CSMA/CA

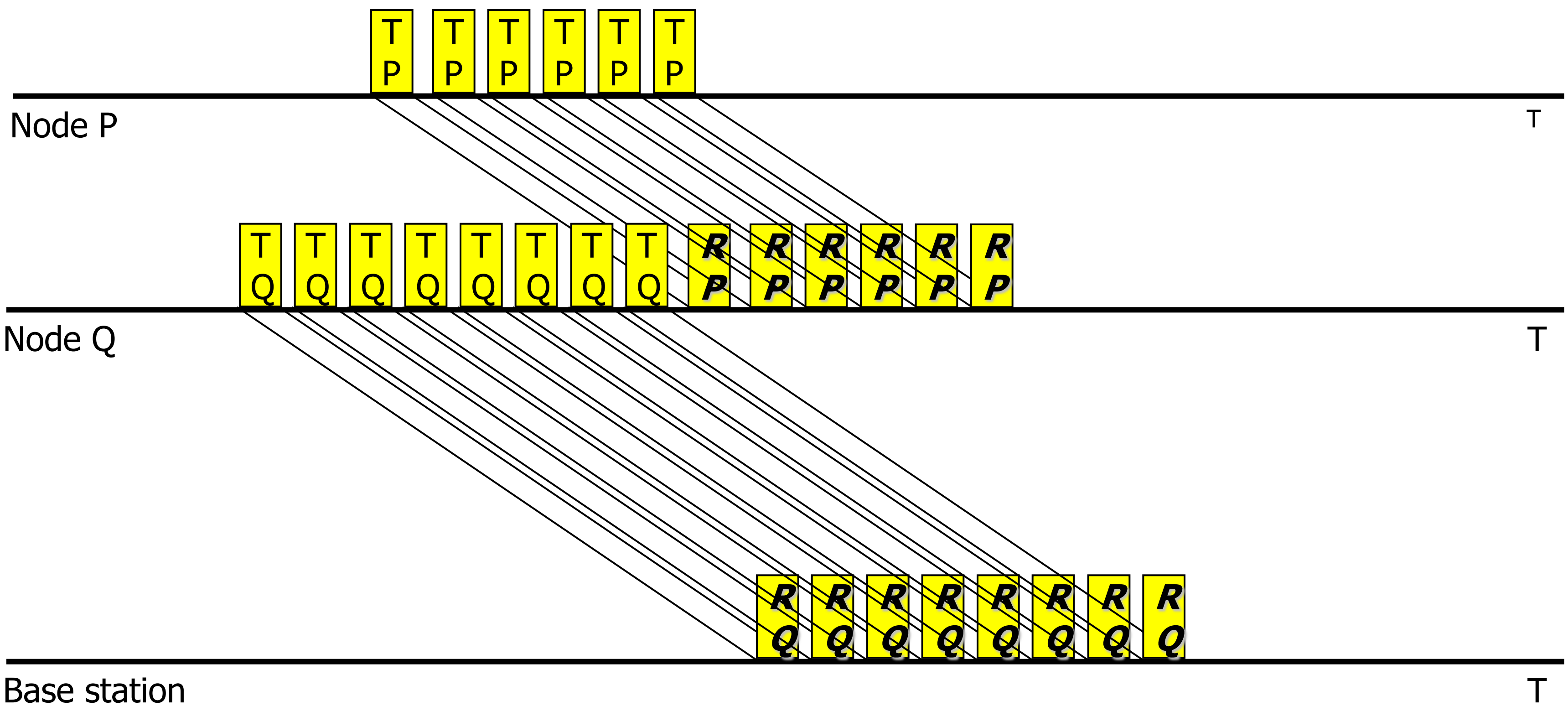
Motivations

- Simultaneous data transmissions: collision or not?



Motivations

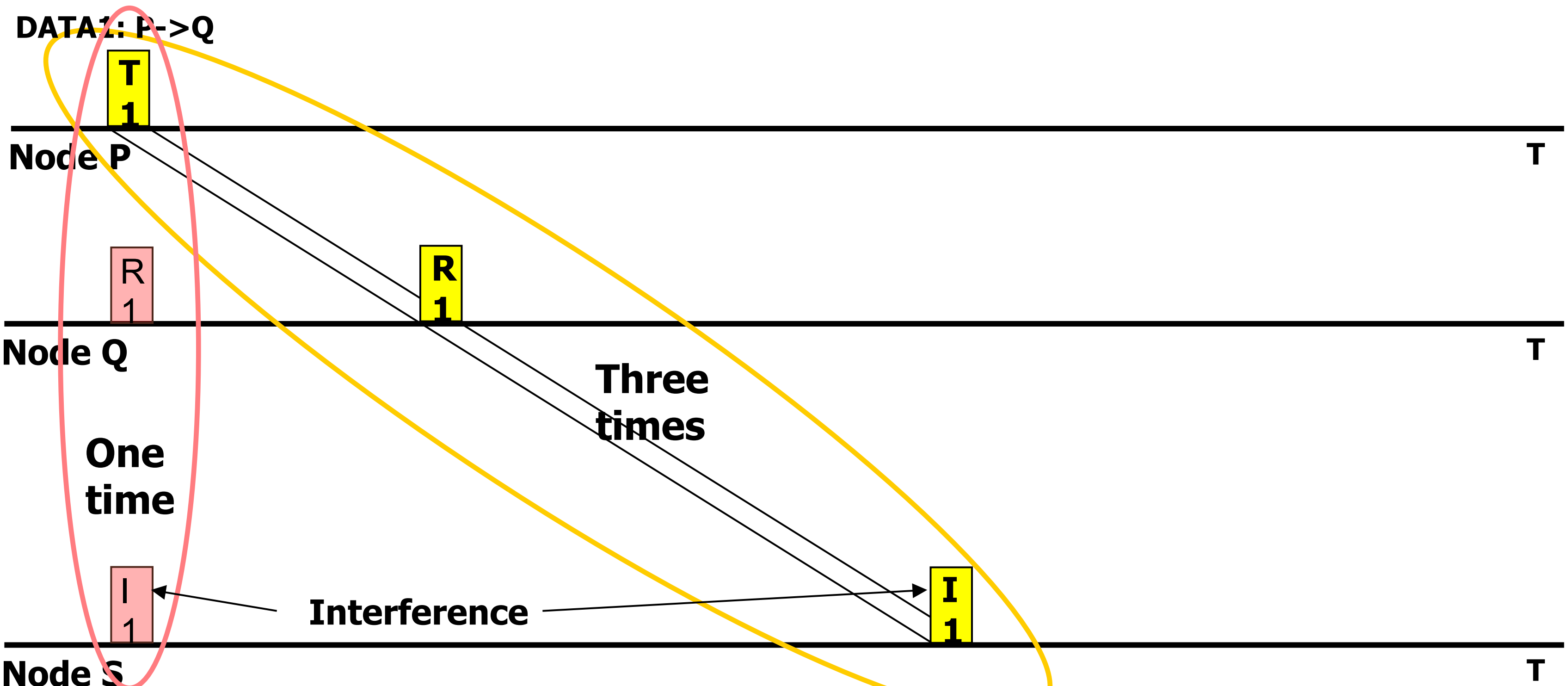
- Use parallel transmissions
- Throughput and delay performance improvement with a compact schedule



Data transmission between 3 nodes

Scheduling Element

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



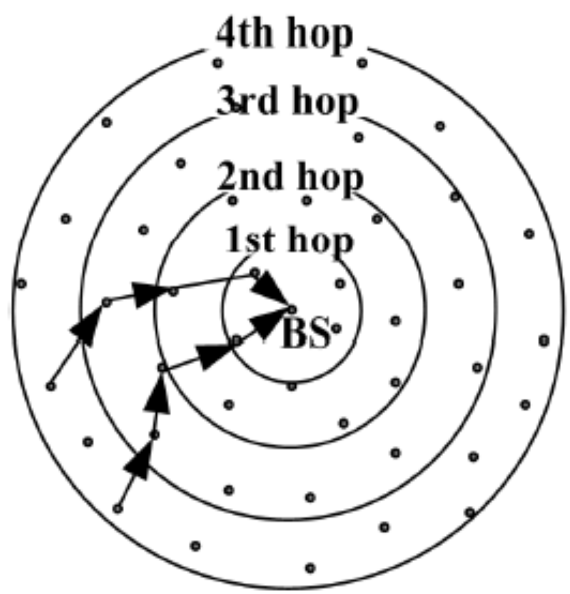
Data transmission between 3 nodes in UWASNs

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
  - Step1: Schedule the BS's data receptions from 1 hop nodes
  - Step2: Schedule the data receptions tier by tier: from inner tier to outer tier
  - Step3: For data receptions from the same tier, arrange them alternatively

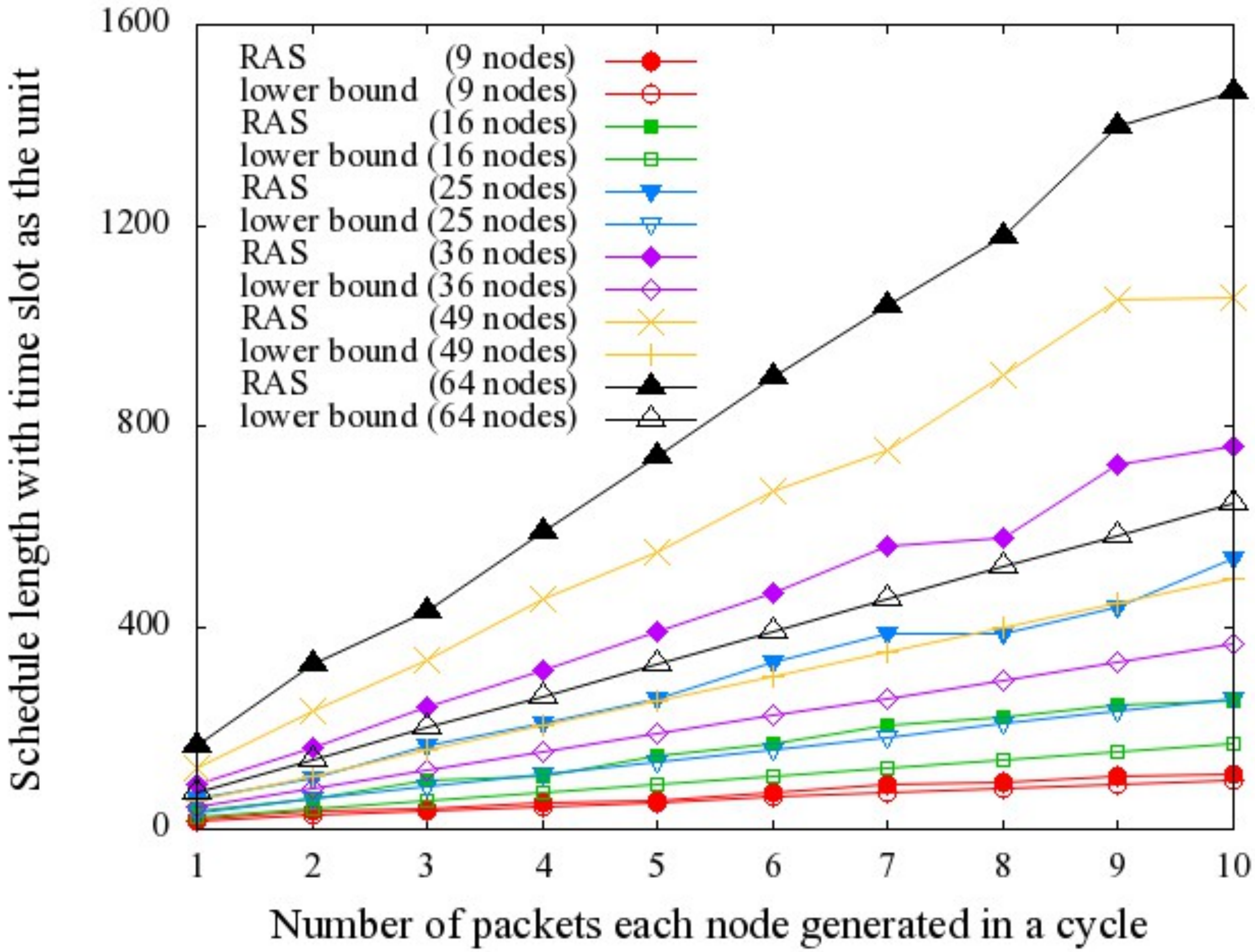


Performance Evaluation

- 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

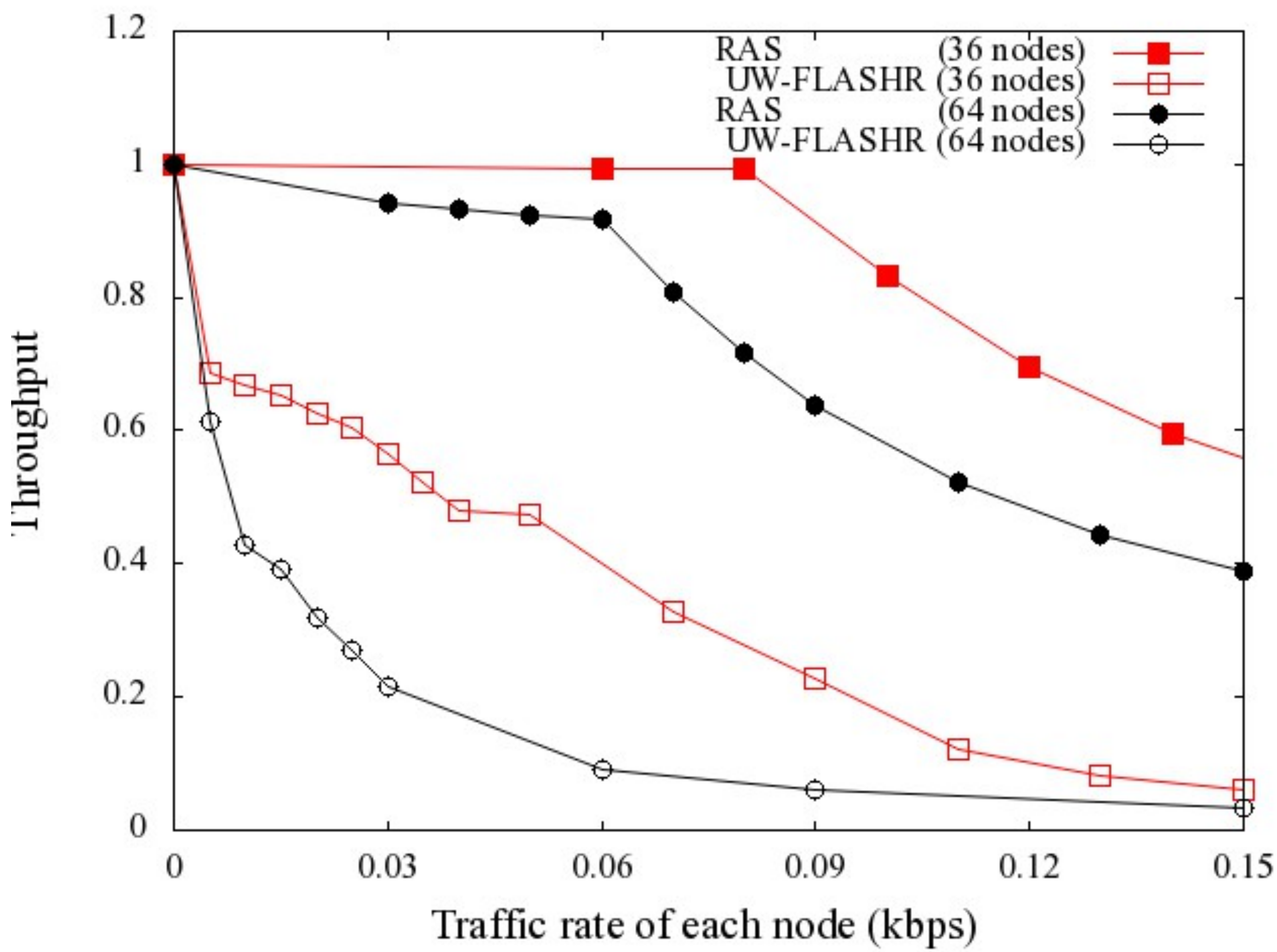
Performance Evaluation

- Schedule length for RAS: scalable



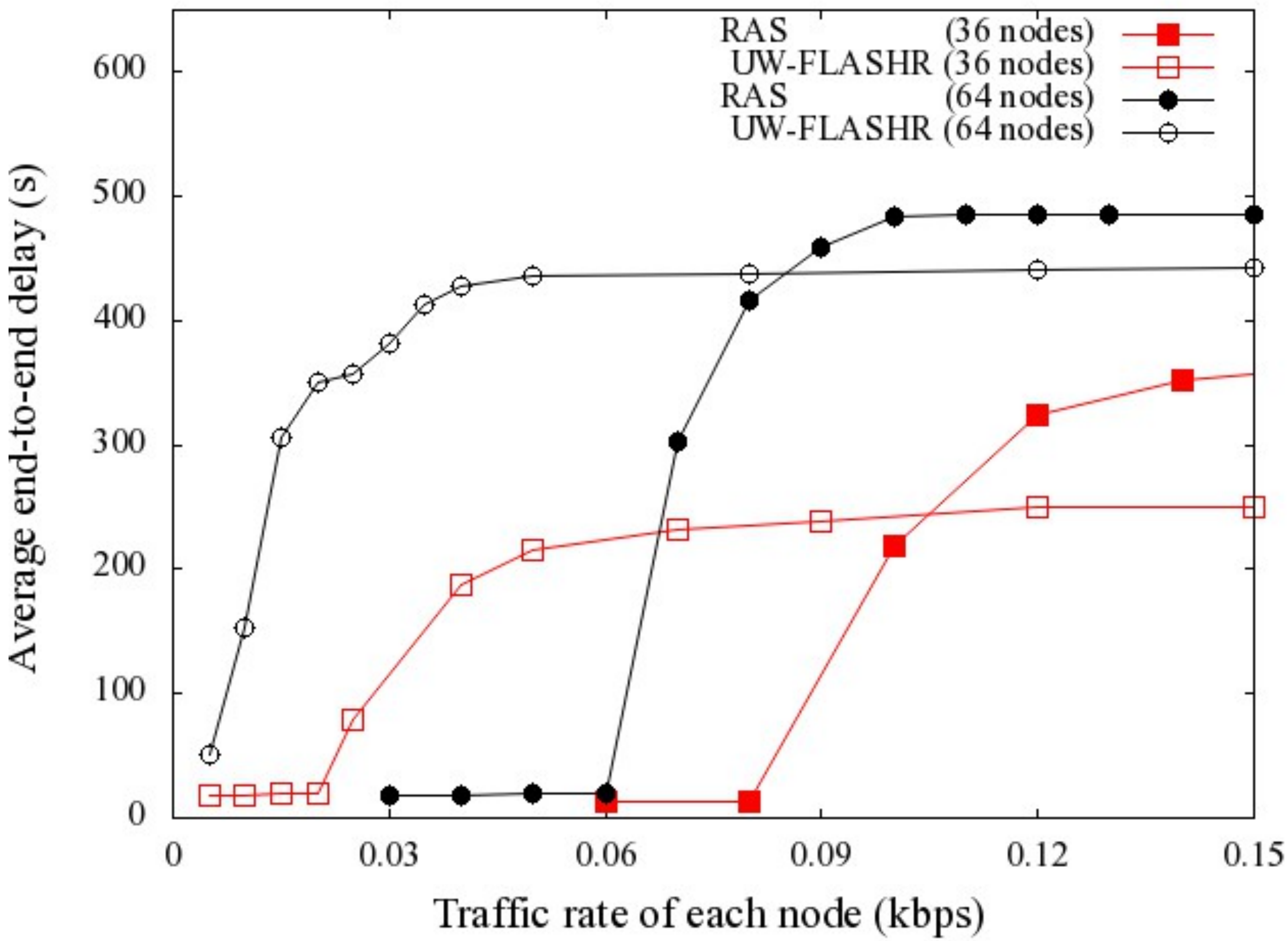
Performance Evaluation

- Throughput



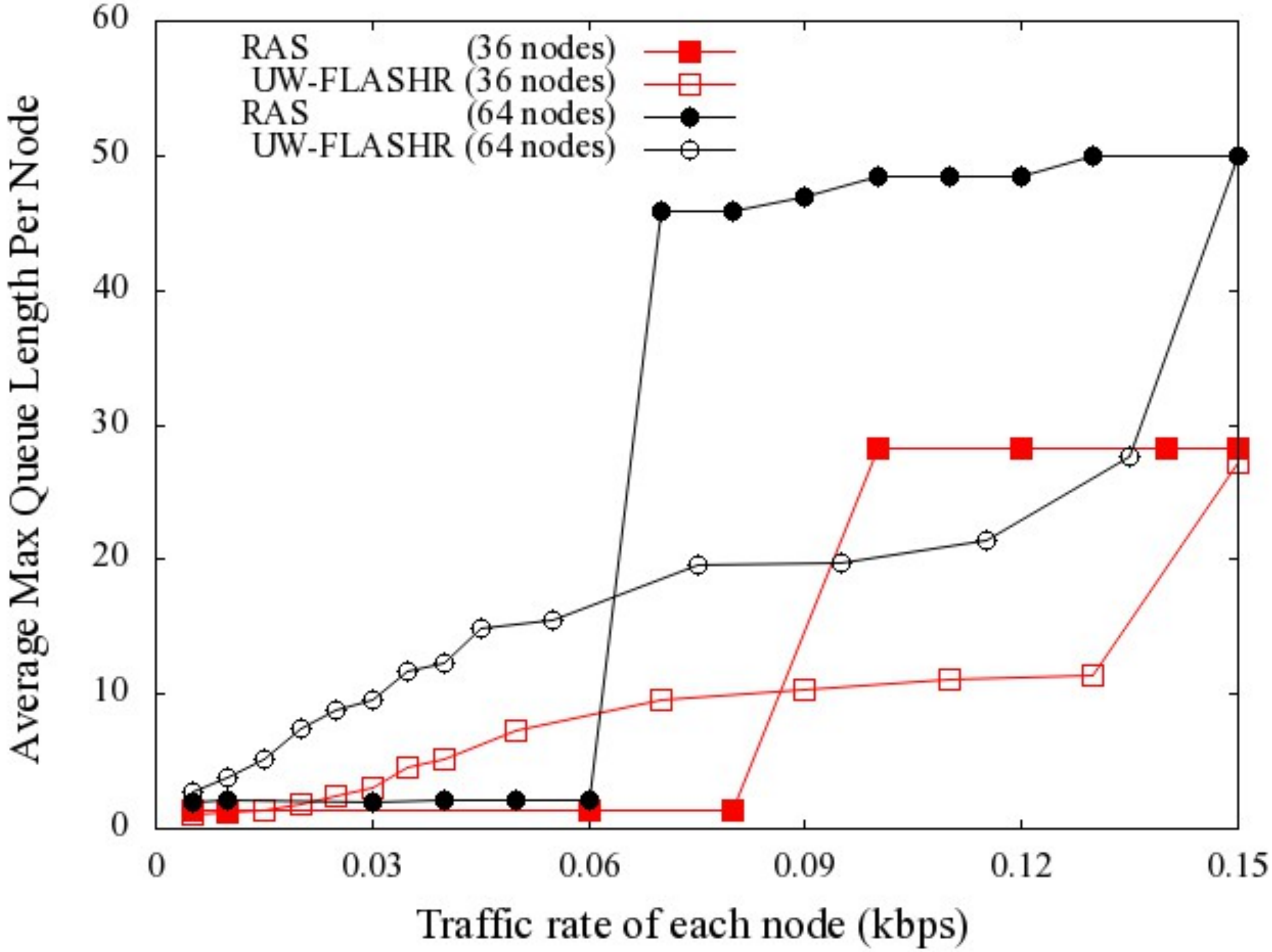
Performance Evaluation

- Average end-to-end delay



Performance Evaluation

- Average maximum queue length per node



Contributions of Part I

- 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

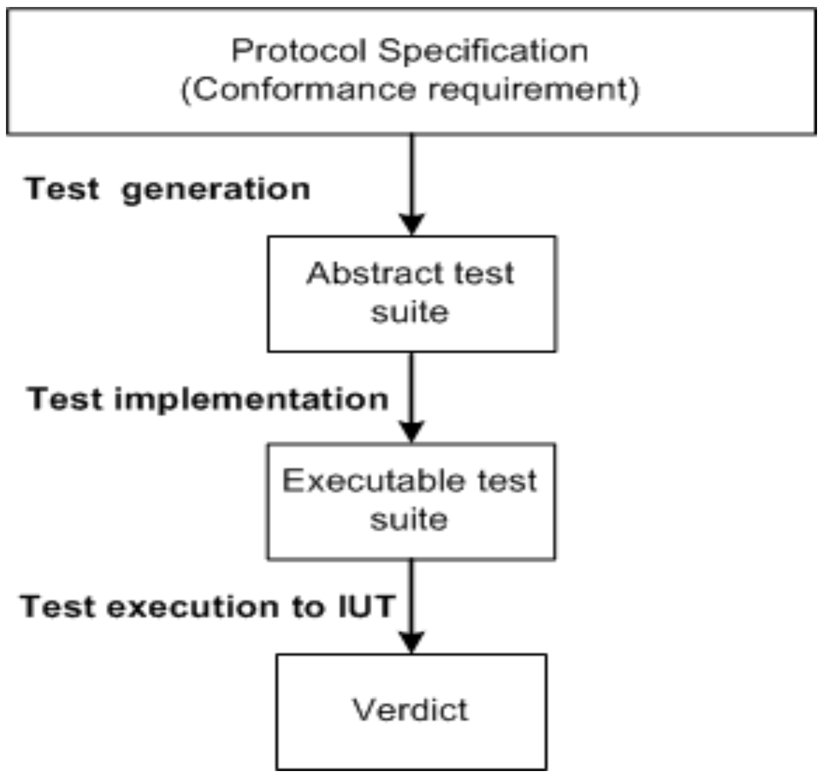
- Sensor node is difficult to control than a PC
  - Limited CPU and inconvenient interface
- How to test the protocol with various topologies and events with only a few real sensor nodes
- Volatile wireless environment will lead to random packet loss, and cause problems in testing

RealProct Solutions to the Challenges

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

Background

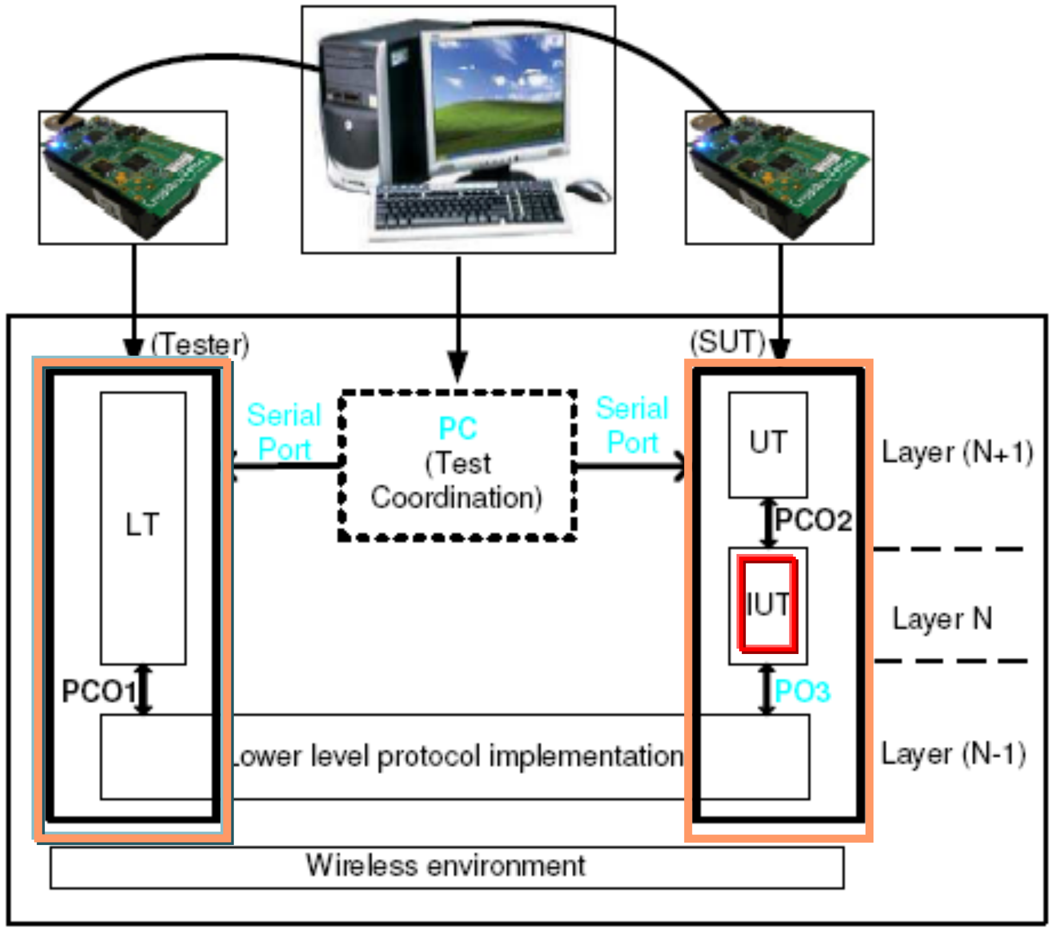
- Protocol conformance testing (PCT) process
  - IUT (Implementation Under Test)



RealProct Architecture

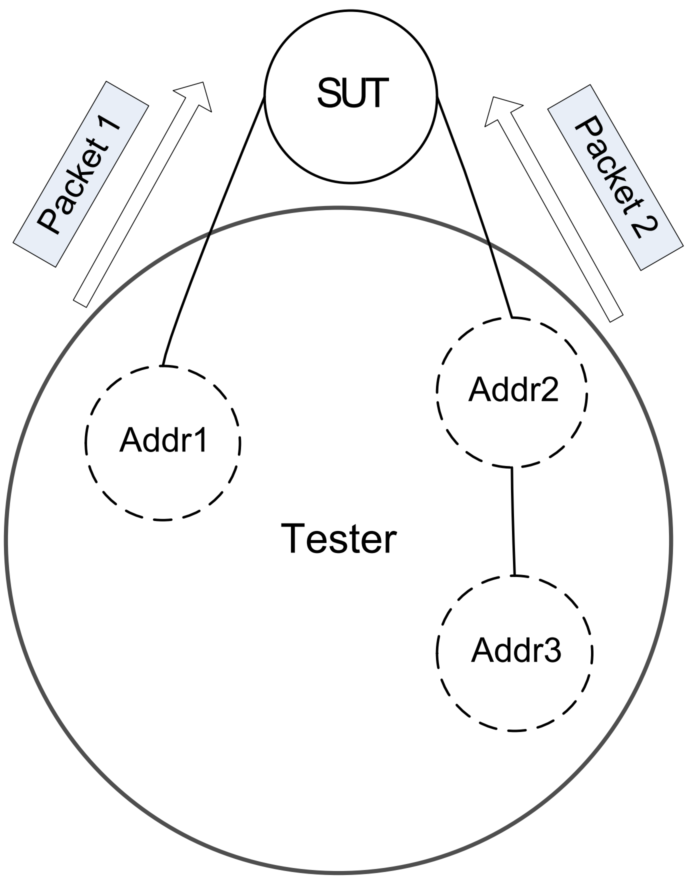
- SUT (System Under Test)

Tester executes test cases



Topology Virtualization

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

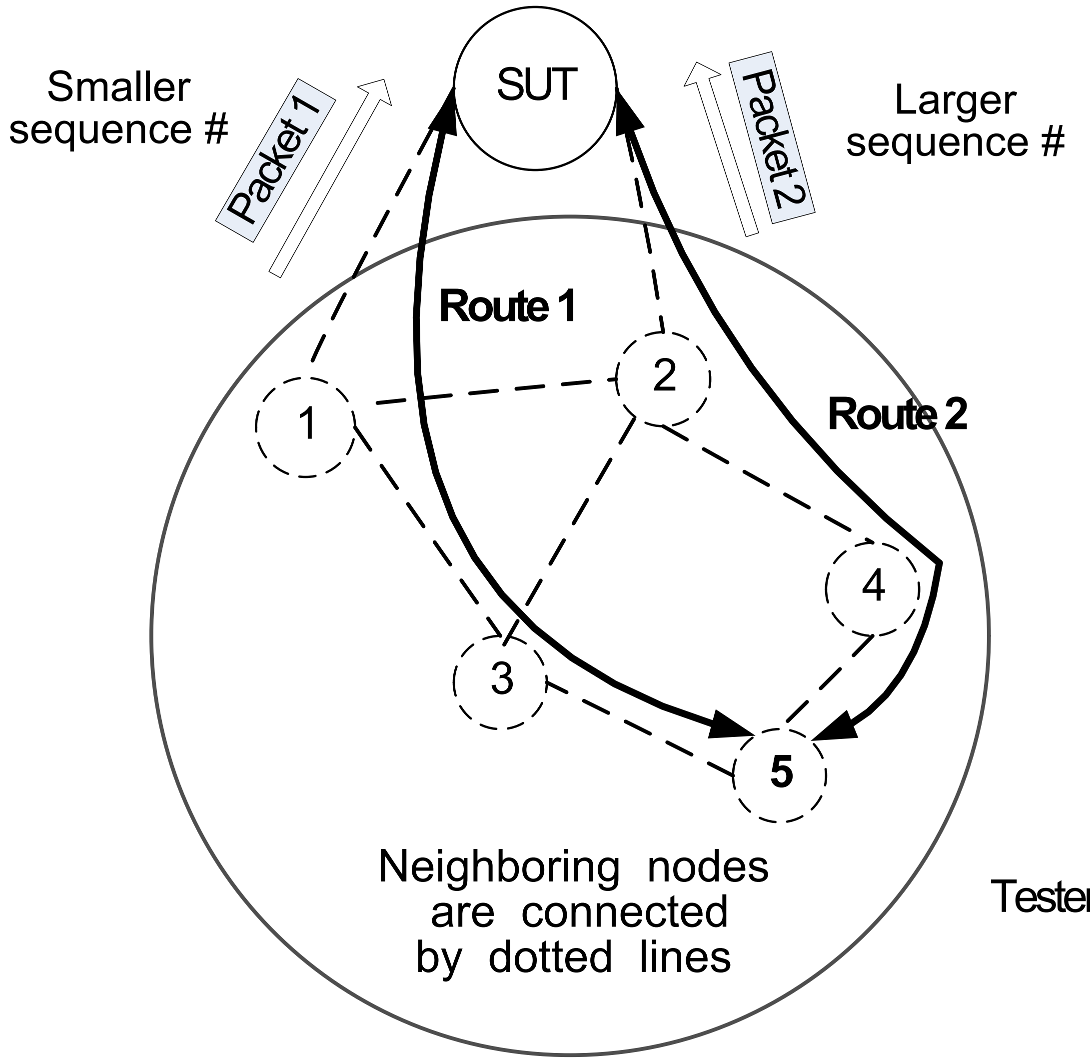


Content of Packet 1:
Sender address is Addr1.

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



Reason to Use Dynamic Test Execution

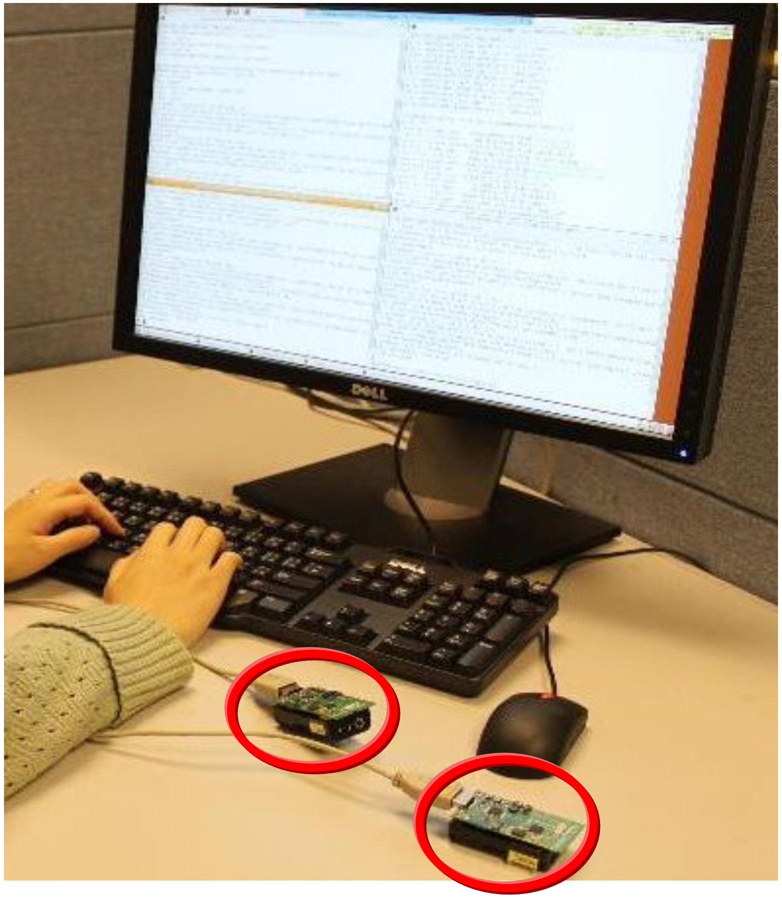
- Suppose packet loss probability is L_0, a test case is executed n times, and it passes n_1 times and fails n_2 times
  - If $n_1 > n_2$, then declare as pass, calculate the FN (false negative) probability
  - If $n_1 < n_2$, then declare as fail, calculate the FP (false positive) probability

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

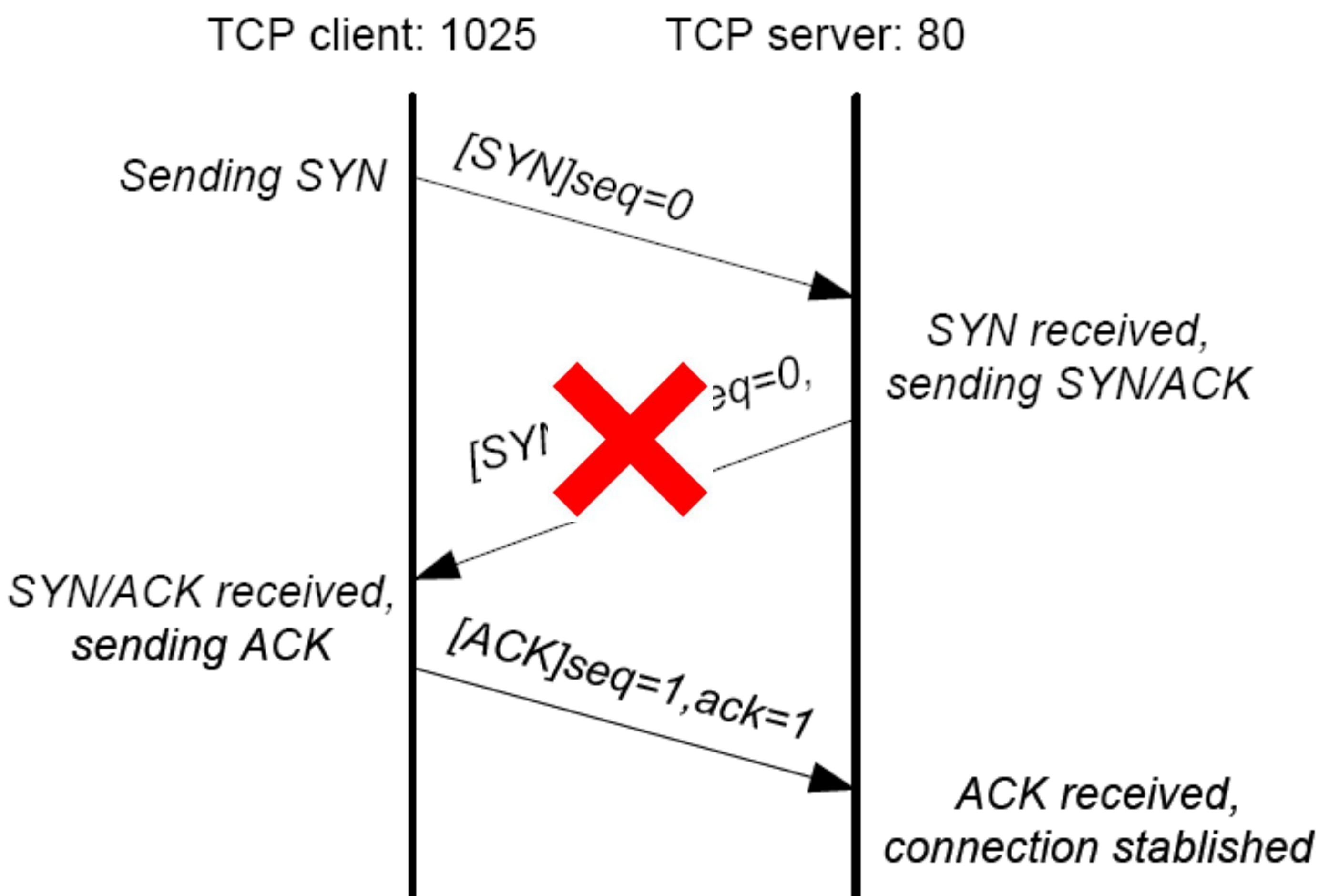


Performance Evaluation

- 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

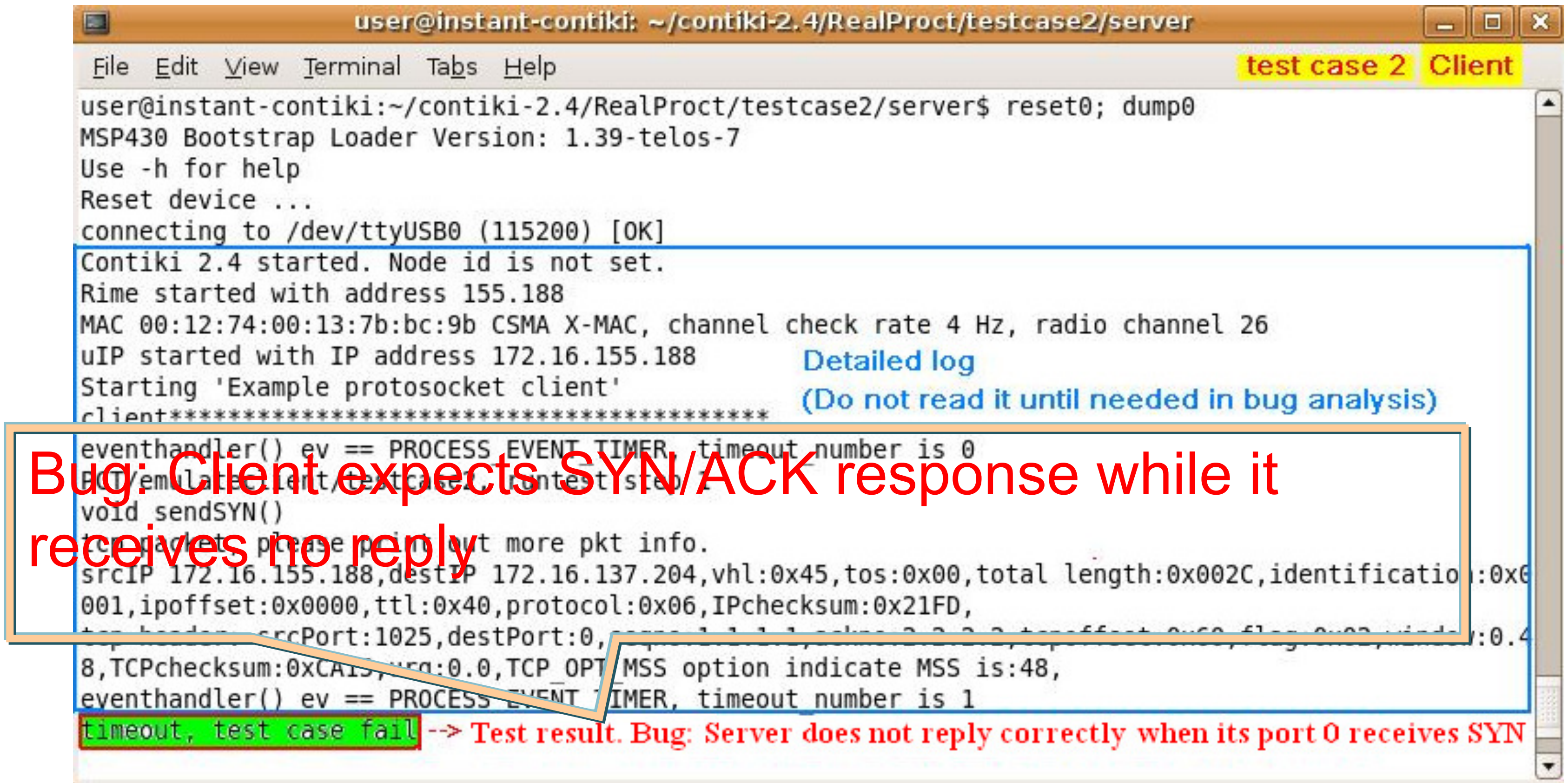
Performance Evaluation

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



Performance Evaluation

- Bug 1 – Client (Tester) connects to opened TCP port 0 of Server (SUT)



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

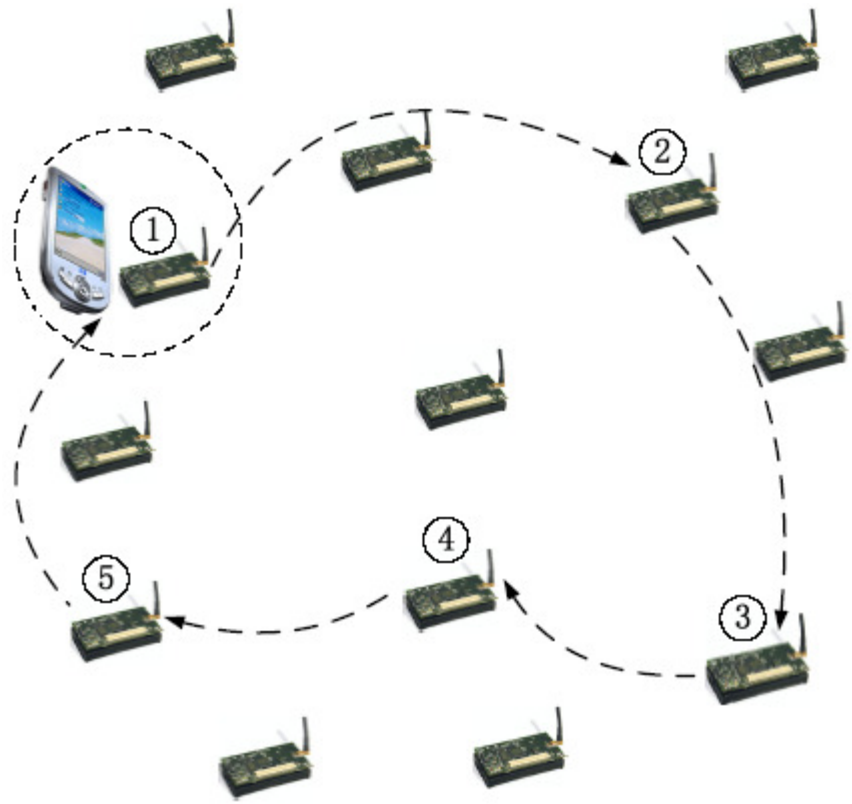
Motivations and Related Work

- 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

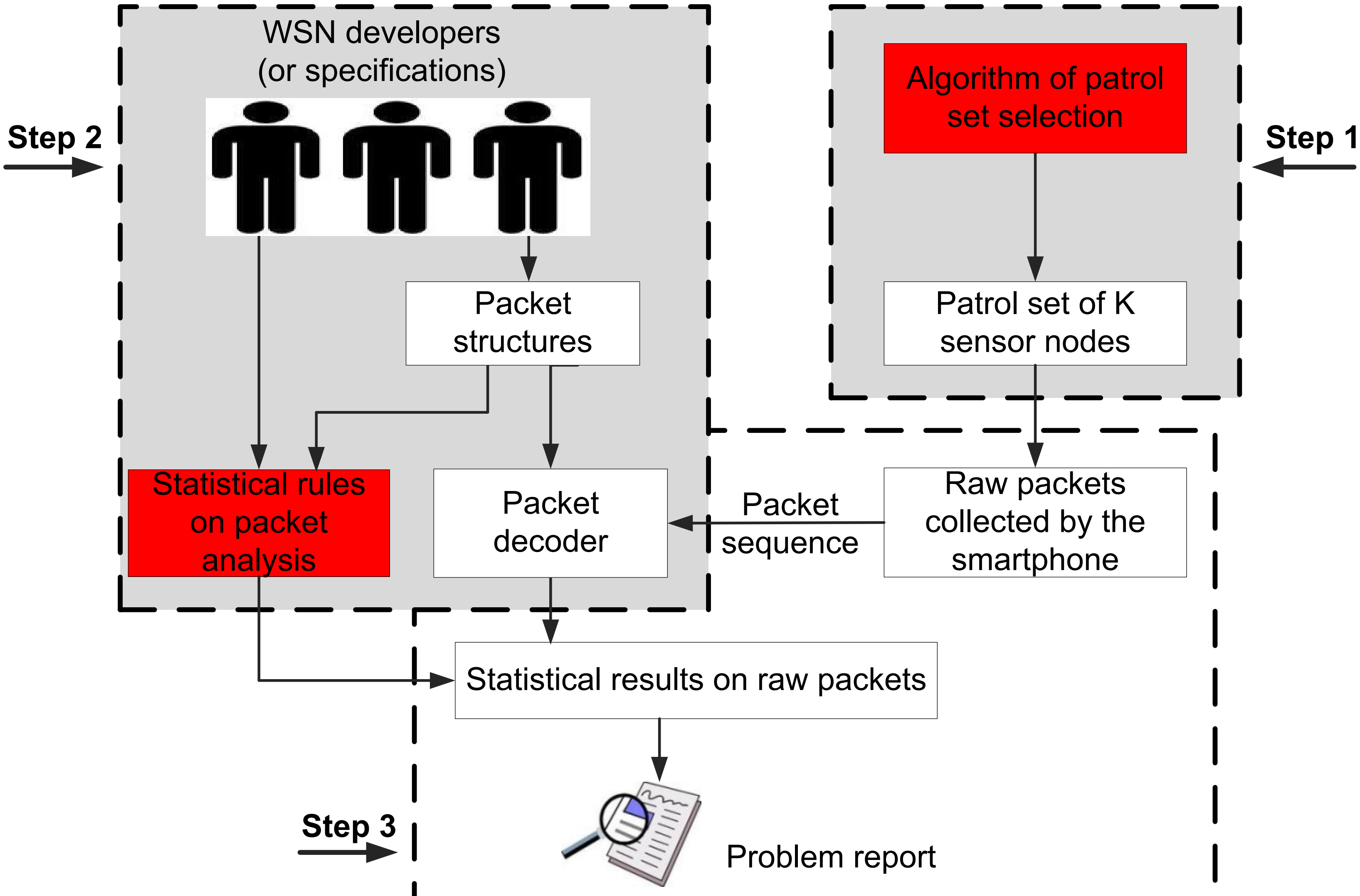
- How to determine the abnormal behaviors from the collected various kinds of raw packets
- How to design the patrol method to increase the failure detection rate

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



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

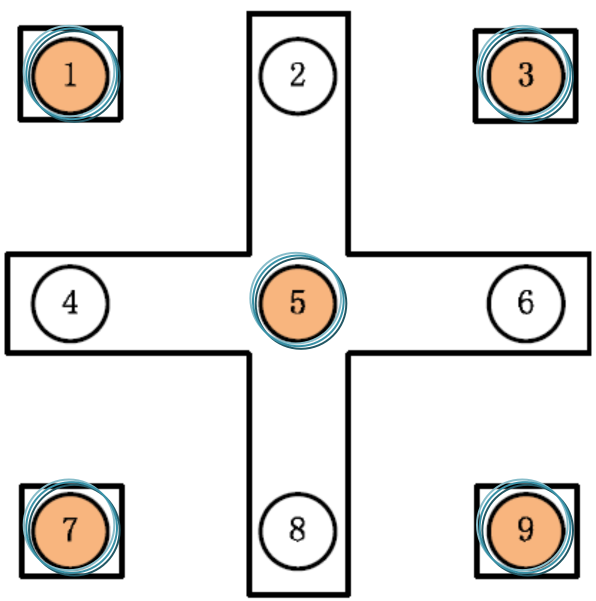
Coverage-oriented Smartphone Patrol Algorithms: Greedy Method (GM)

- 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

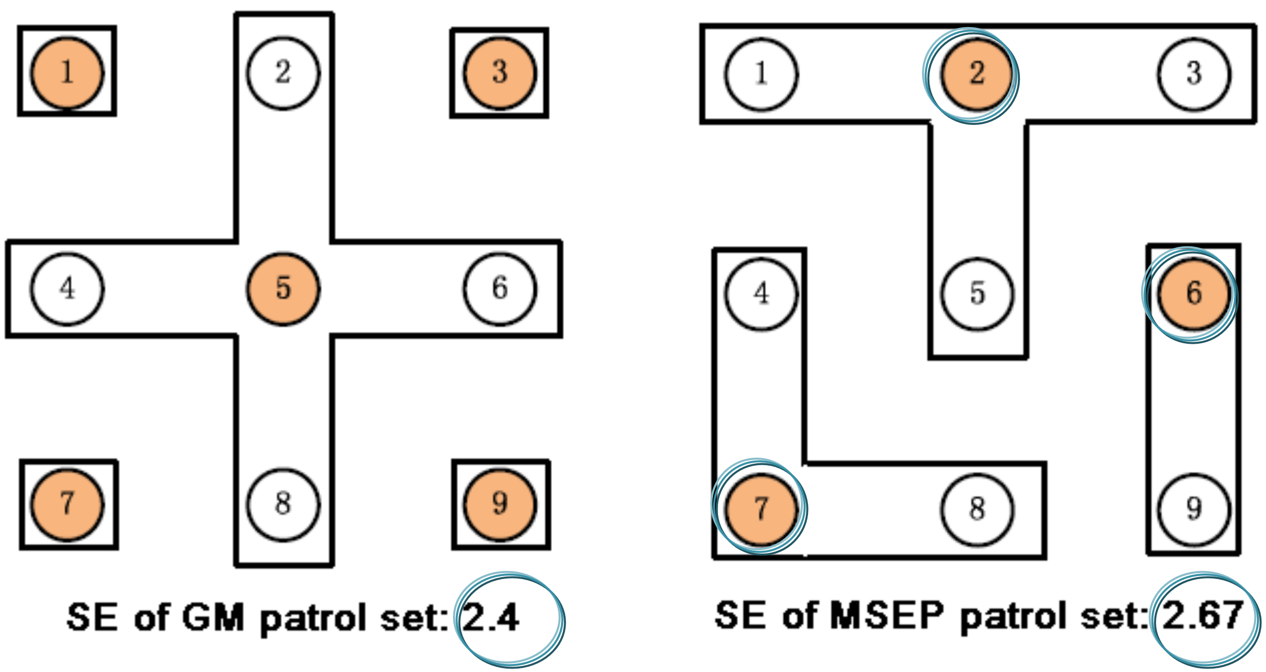


Aim at improving patrol set snooping efficiency

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

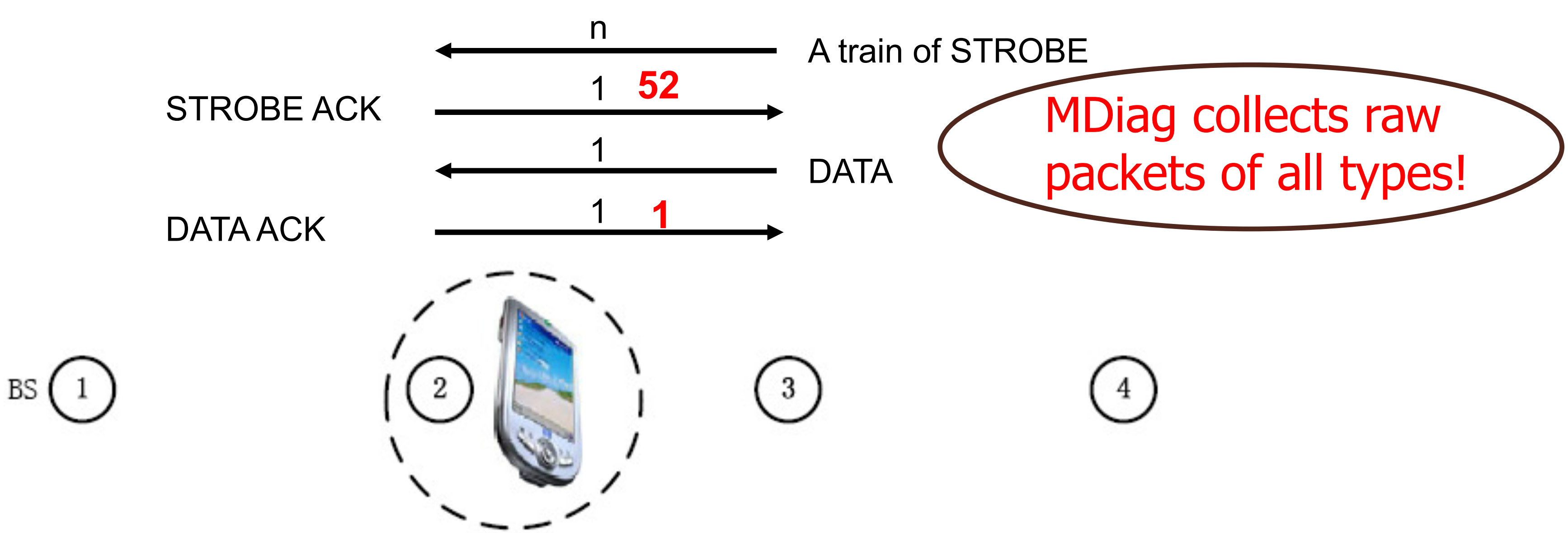


Performance Evaluation: Settings

- 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



Performance Evaluation: Permanent Failure Detection

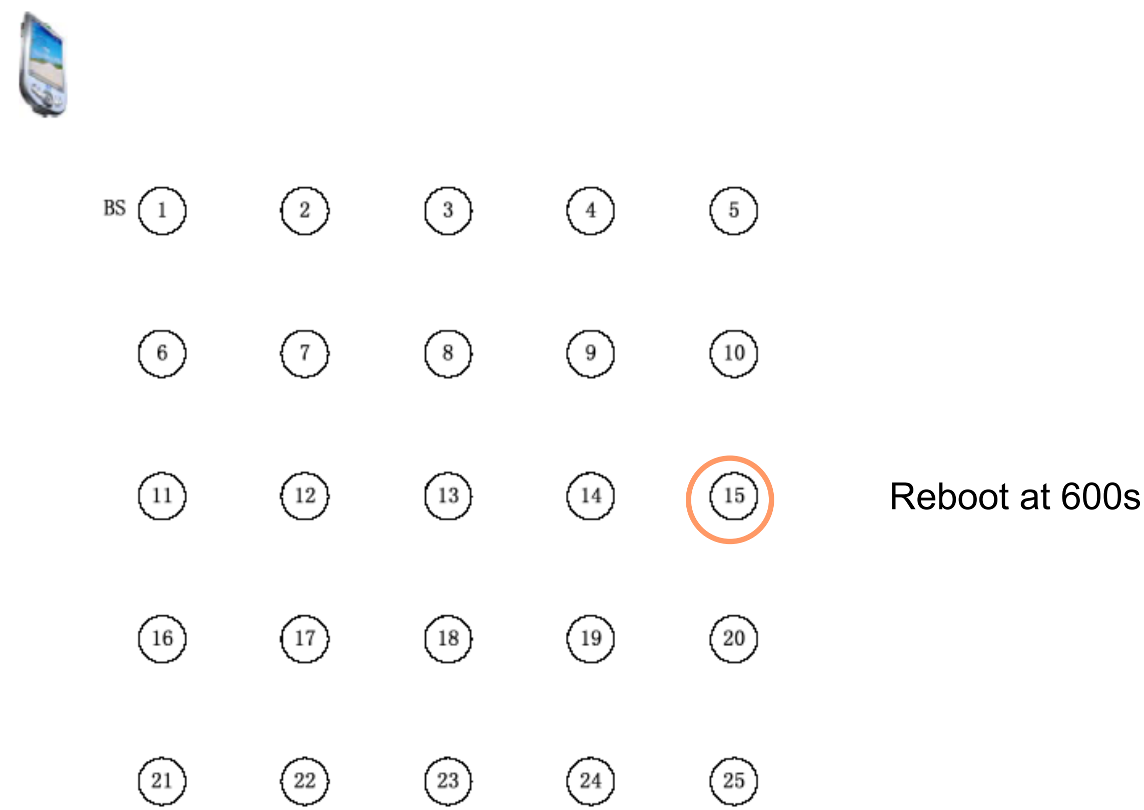
- Surprising reason: a 'printf' statement in the WSN application program
  - Trigger serial port interrupts: consume a lot of CPU resources
  - CPU is too busy to handle packet transmissions and receptions

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

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)



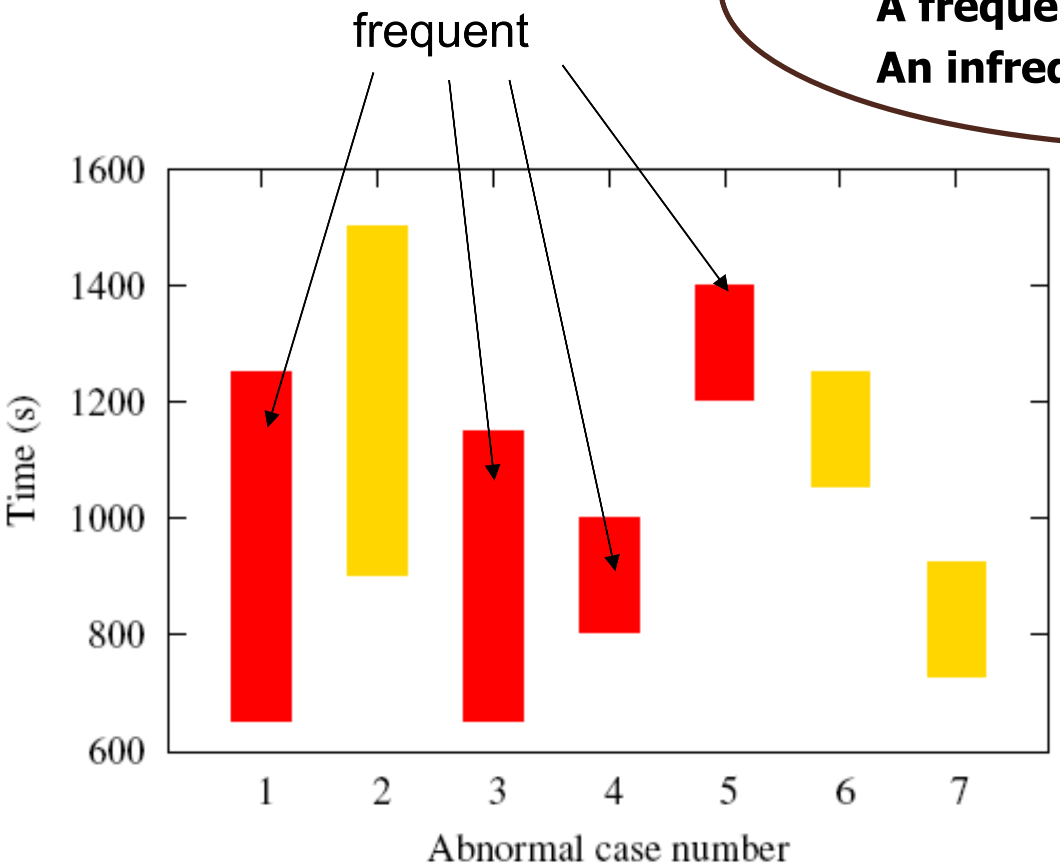
Performance Evaluation: Short-term Failure Detection

- Abnormal case (bidirectional data exchange)
  - Short & long
  - Frequent & infrequent

R represents a datum in the opposite direction of a datum D.

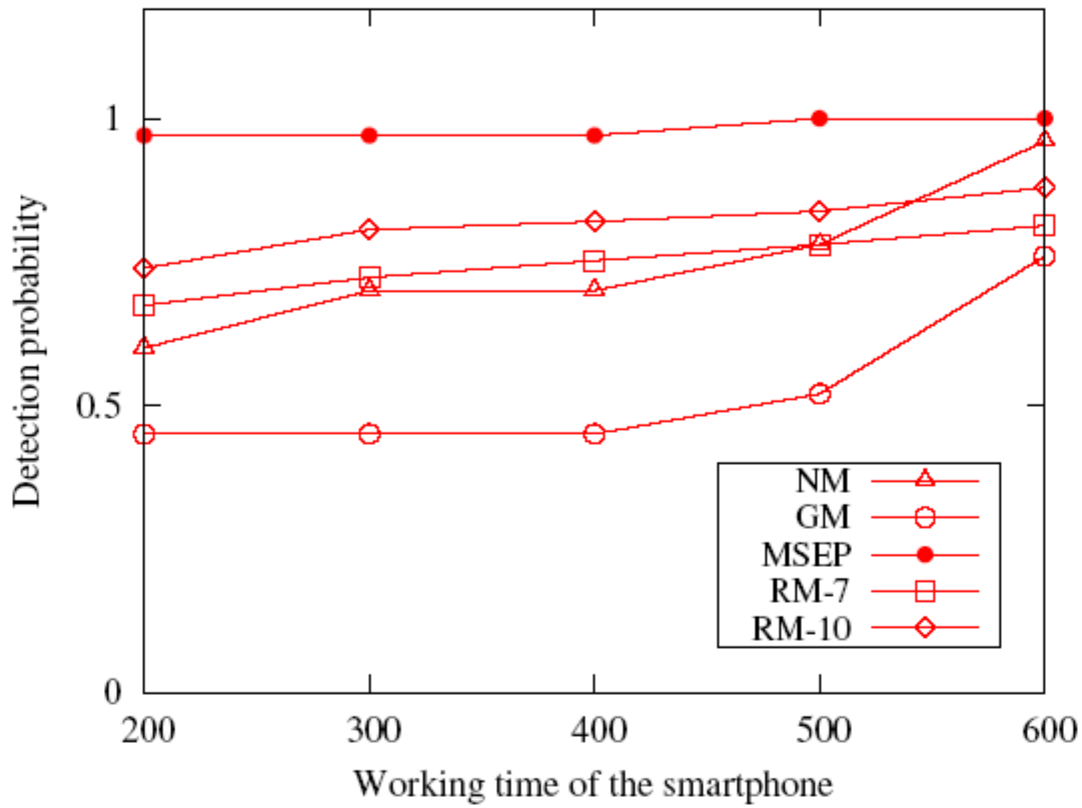
A frequent AC: DRDRDRDRDR

An infrequent AC: DDDDDRRRRR



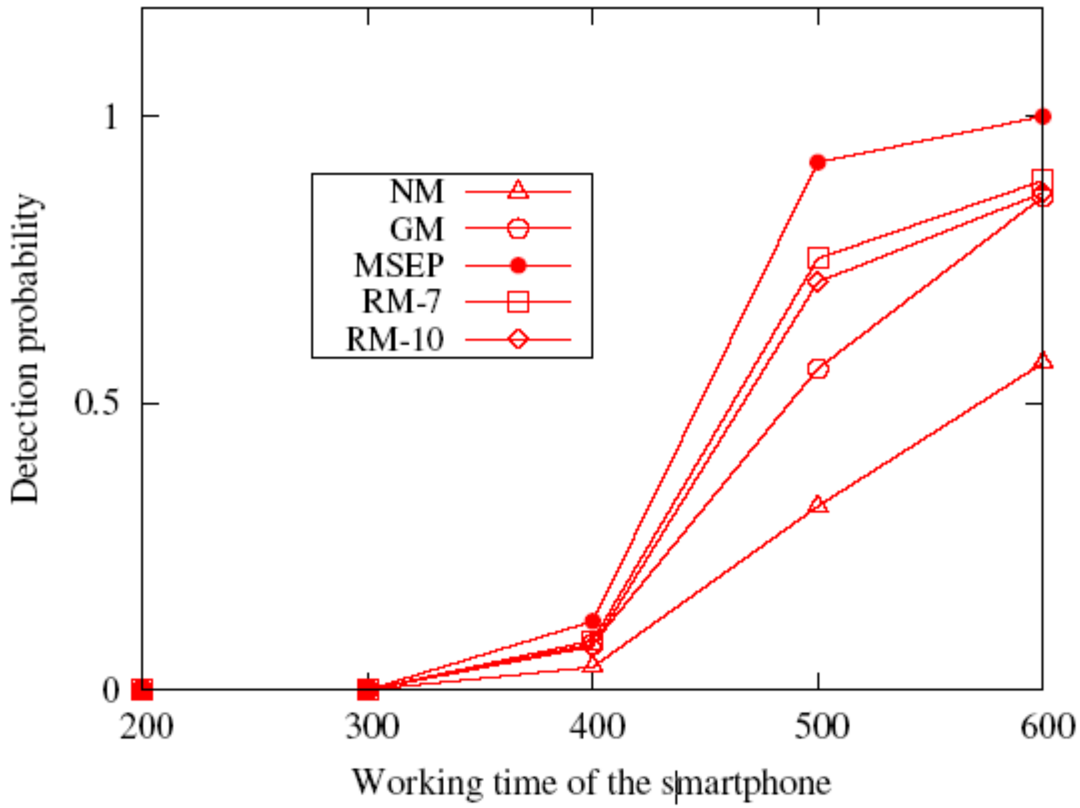
Performance Evaluation: Short-term Failure Detection

- Detection probability of abnormal case 1 (long and frequent)



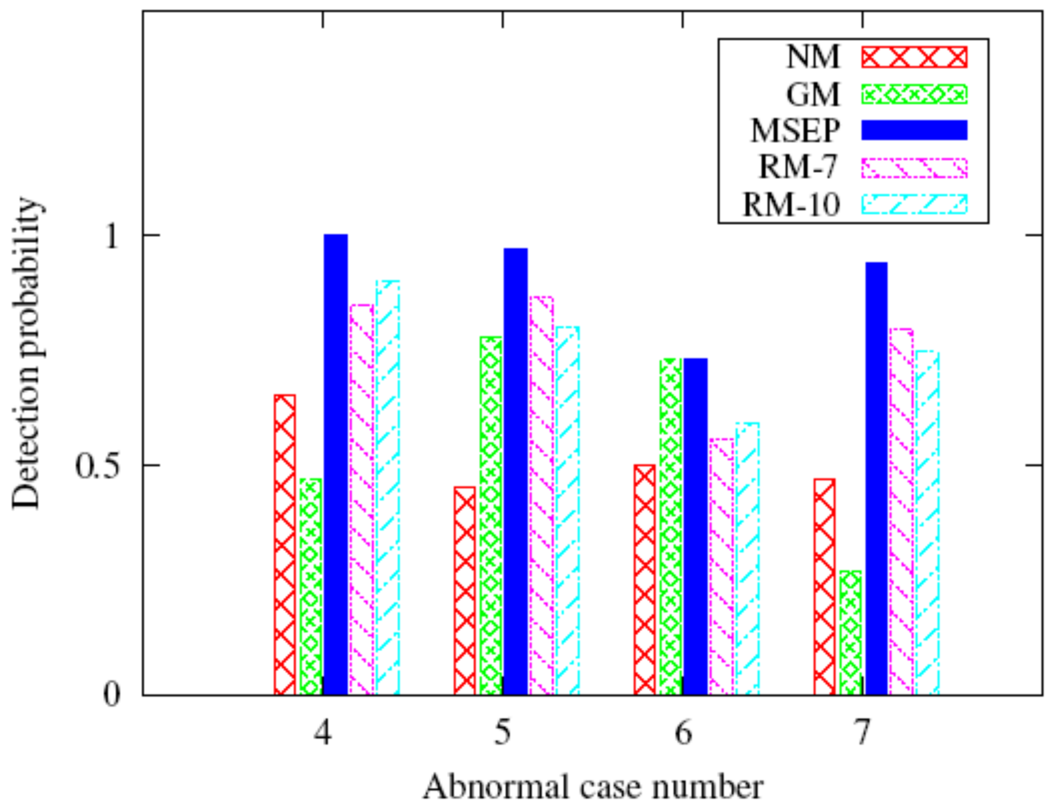
Performance Evaluation: Short-term Failure Detection

- Detection probability of abnormal case 2 (long but infrequent)



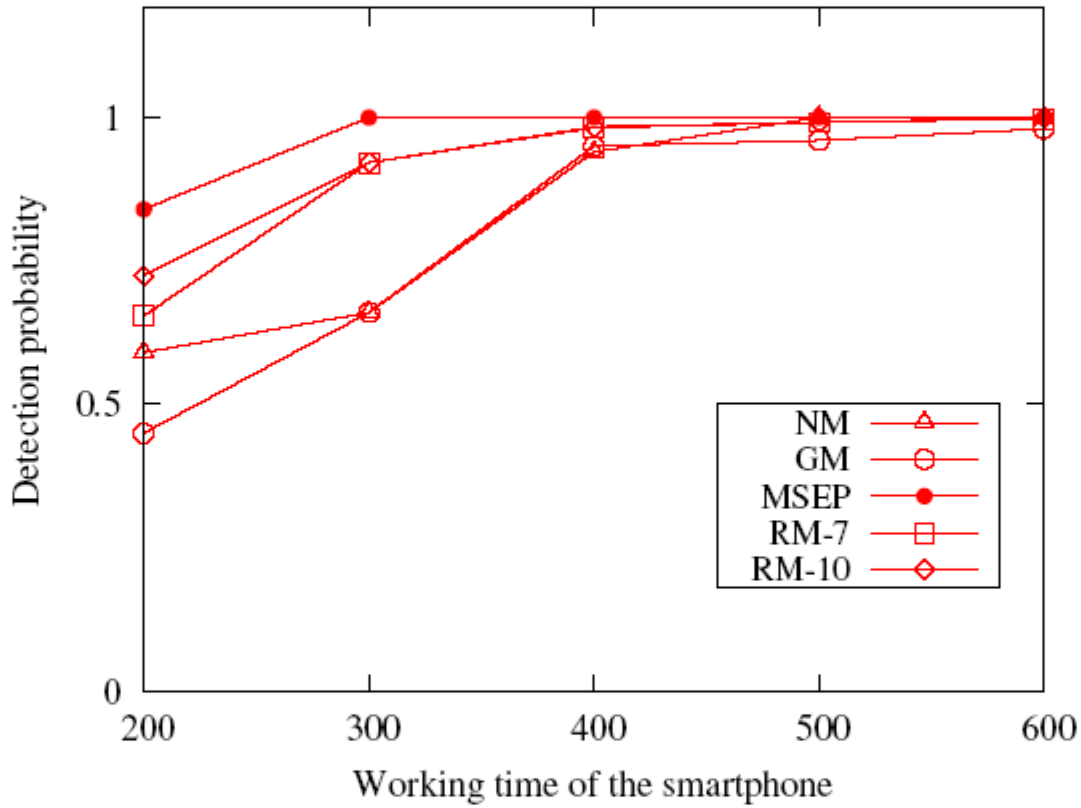
Performance Evaluation: Short-term Failure Detection

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



Performance Evaluation: Short-term Failure Detection

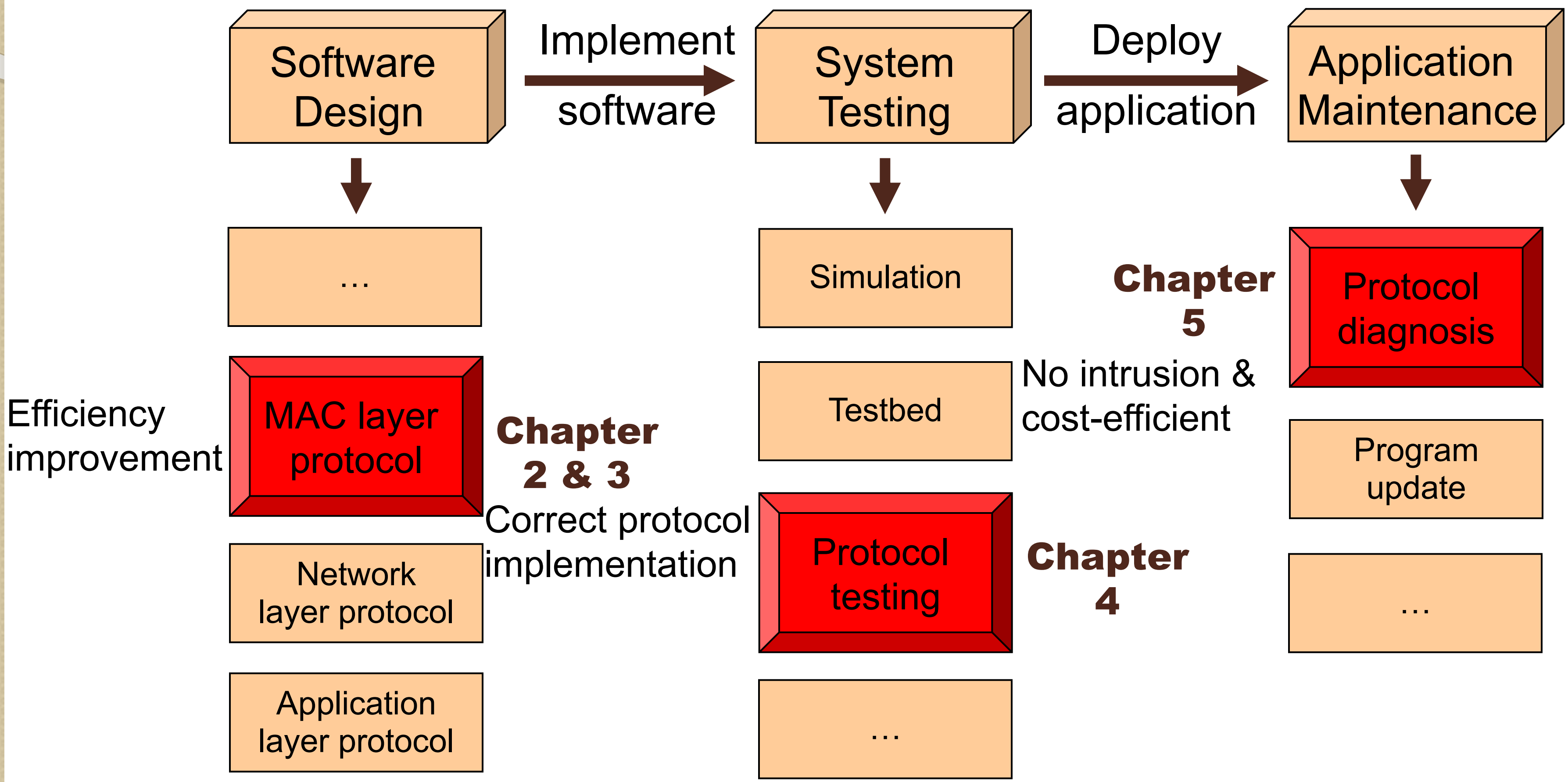
- Detection probability of all ACs



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



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

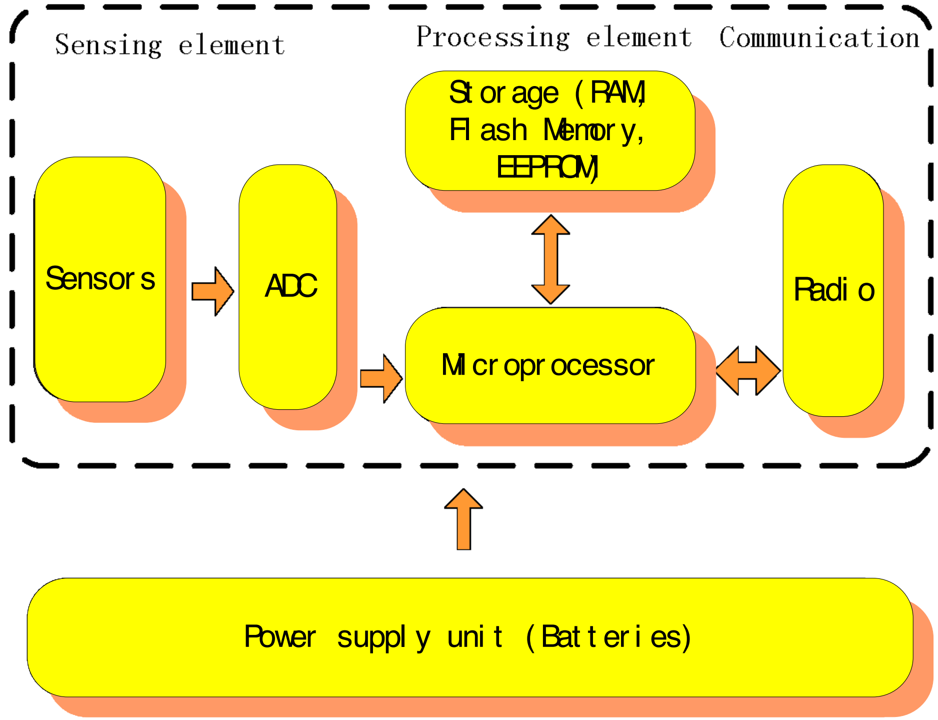
Thank you!

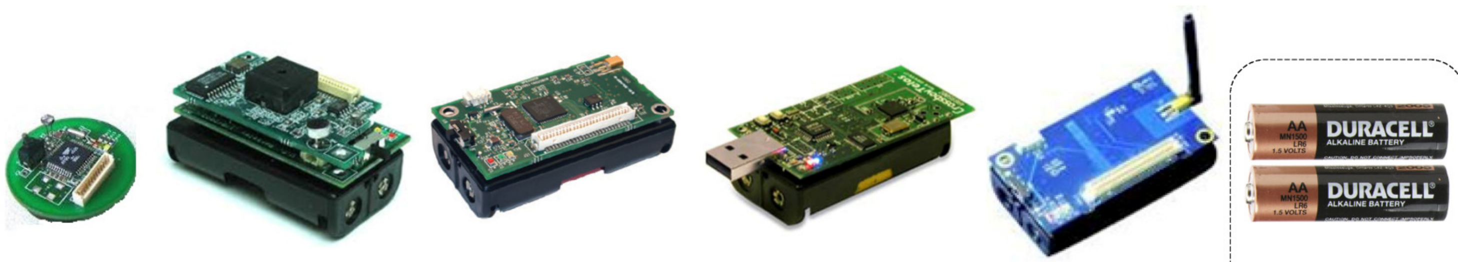
Q & A

Appendix

Introduction: Sensor Nodes

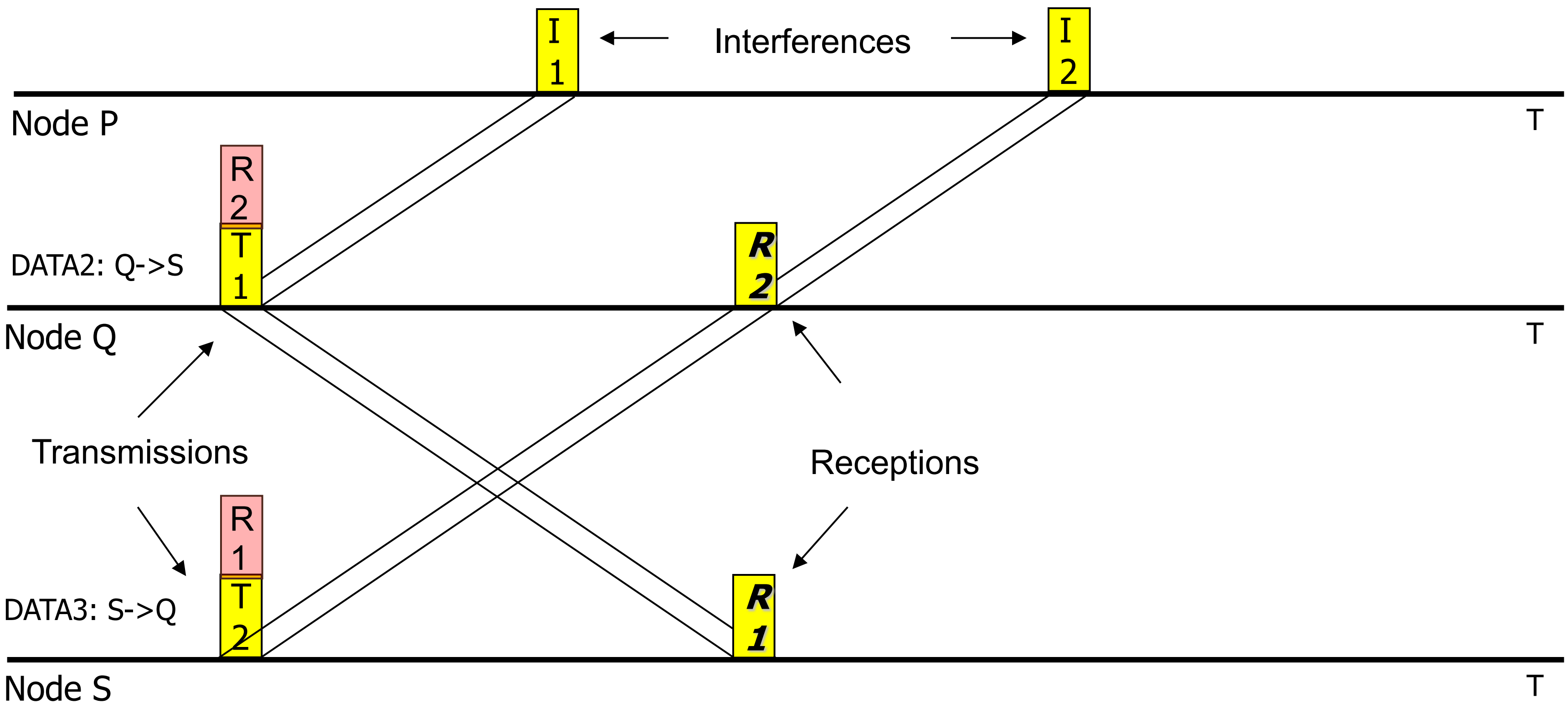
- Power supply unit
- Sensors
- ADC (analog-to -digital converter)
- Microprocessor
- Radio transceiver
- Storage





WeC mote (1999) | Mica (2001) | Mica2 (2003) | TelosB (2007) | Iris (2008) | Two AA batteries

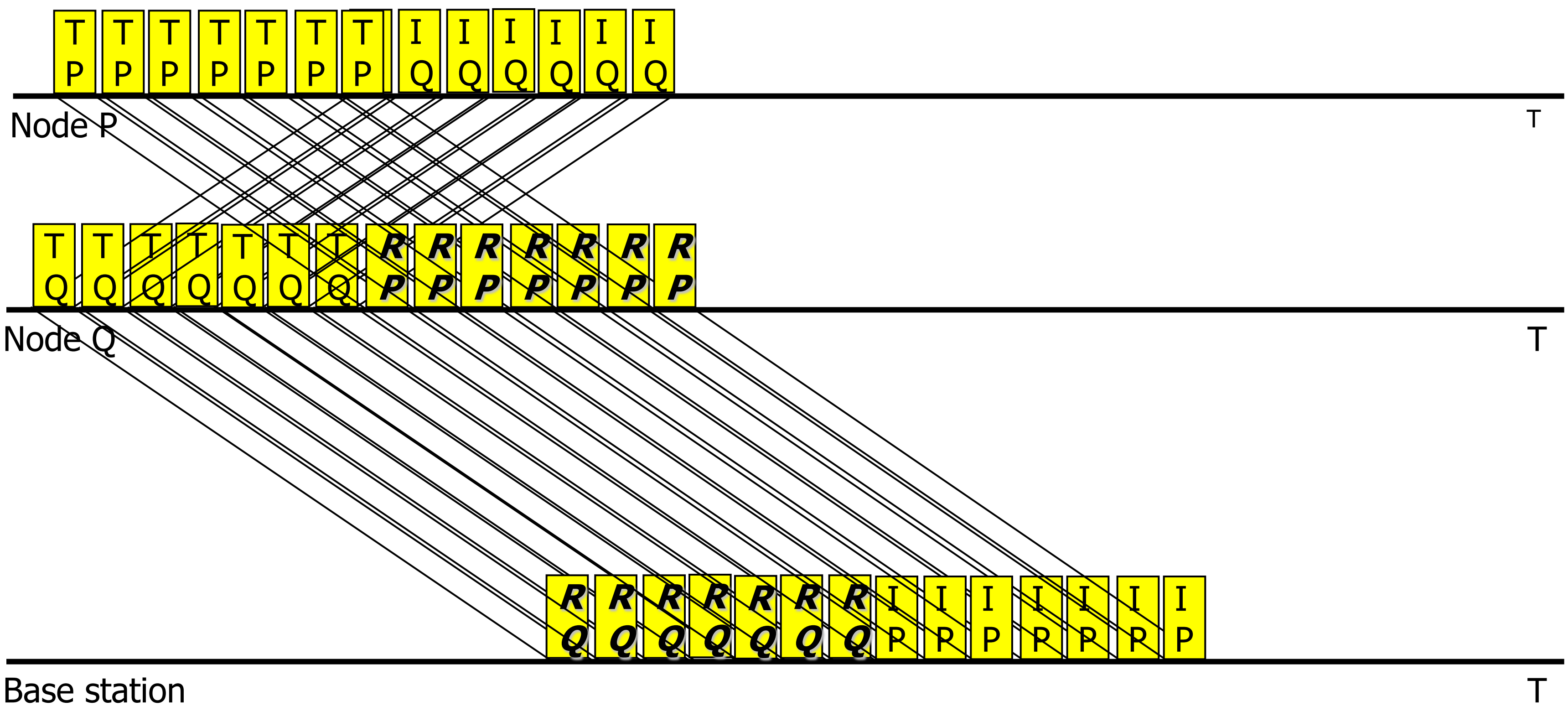
Part 1: Motivations



Data transmission between 3 nodes in UWASNs

Part 1: Motivations

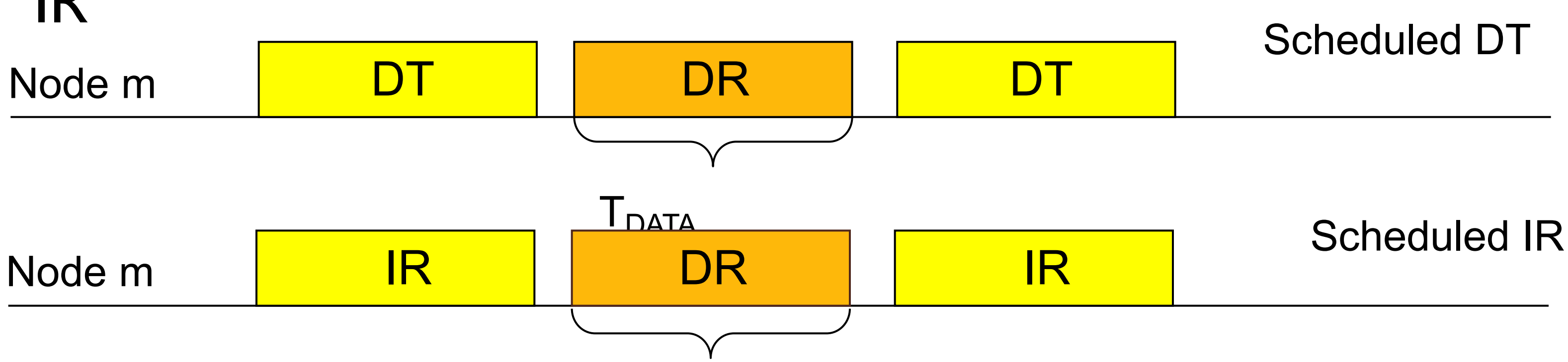
- Use parallel transmissions
- Throughput and delay performance improvement with a compact schedule



Data transmission between 3 nodes

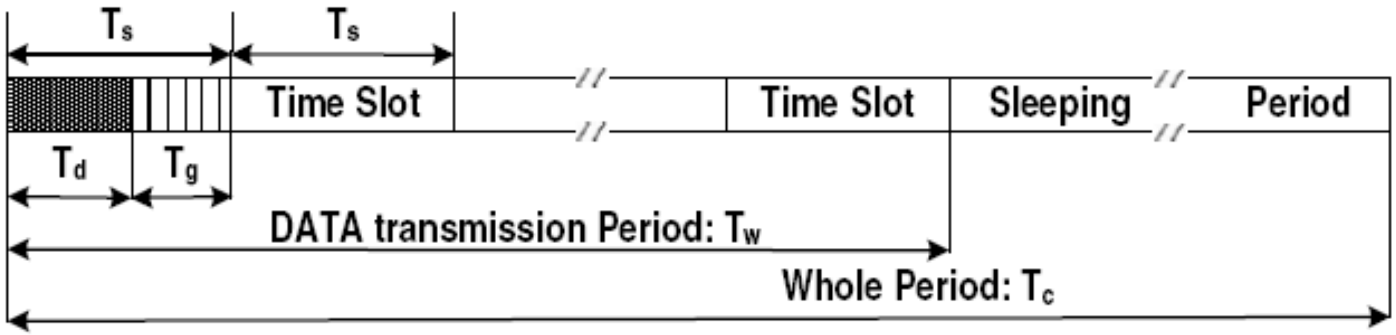
Part 1: Scheduling Principles

- 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



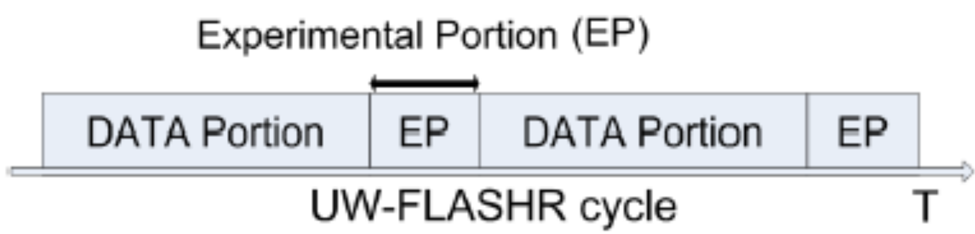
RAS cycle

Part 1: Parameters for Data Transmissions

Parameter	Value
Data Rate	10 kbps
Data Packet Size	100 bytes
Control Packet Size	10 bytes
Transmission Range (communication range)	1500 m
Interference Range	3500 m
Average Distance between Two Nodes	1110 m
Guard time	20 ms
Wireless model	TwoRayGround

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.

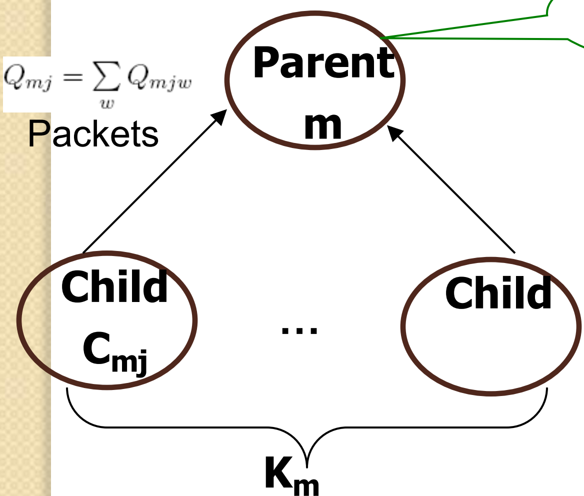


RAS Protocol Design

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

K_m types of elements: $E_{mC_{mj}}$

$$E_{mC_{mj}} = \begin{cases} DT_{C_{mj}}, \\ DR_{C_{mj}}, \\ IR_{SC_{mj}}, \end{cases}$$



$$E_{mC_{mj}} = \begin{cases} DT_{C_{mj}}, & DR_{C_{mj}} - DIS(m, C_{mj})/SOUND\_SPEED, \\ DR_{C_{mj}}, & \\ IR_{SC_{mj}}, & DT_{C_{mj}} + DIS(C_{mj}, S_{C_{mj}})/SOUND\_SPEED, S_{C_{mj}} \text{ is} \\ & \text{the set of nodes which are within the node } C_{mj}\text{'s} \\ & \text{interference range, and } m \in S_{C_{mj}}, j = 1, 2, \cdots, K_m. \end{cases}$$

Set DR_{Cmj} to 0

$R_{mC_{mj}Q_{mjw}}$

RAS Protocol Design

- Scheduling algorithm formulation
  - Schedule according to the principles

$$
\min \max R_{mC_{mj}Q_{mjw}},\ for\ j = 1,2,...K_m, \sum_{w} Q_{mjw} = Q_{mj}\ and\ m \in S.
$$

$$
\text{s.t.} \begin{cases}
R_{mC_{mj}Q_{mjw}} + E_{mC_{mj}}.DR_{C_{mj}} > R_{zC_{zj}Q_{zjw}} + E_{zC_{zj}}.DT_{C_{zj}} + D_{DATA}, or \\
\quad R_{mC_{mj}Q_{mjw}} + E_{mC_{mj}}.DR_{C_{mj}} + D_{DATA} < R_{zC_{zj}Q_{zjw}} + E_{zC_{zj}}.DT_{C_{zj}}, \\
\quad for\ C_{zj} = m, z \in S', & (1) \\
R_{mC_{mj}Q_{mjw}} + E_{mC_{mj}}.DR_{C_{mj}} > R_{zC_{zj}Q_{zjw}} + E_{zC_{zj}}.IR_m + D_{DATA}, or \\
\quad R_{mC_{mj}Q_{mjw}} + E_{mC_{mj}}.DR_{C_{mj}} + D_{DATA} < R_{zC_{zj}Q_{zjw}} + E_{zC_{zj}}.IR_m, \\
\quad for\ z \in S', & (2) \\
R_{mC_{mj}Q_{mjw}} > R_{mC_{mj}Q_{mjw'}}, or\ R_{mC_{mj}Q_{mjw}} < R_{mC_{mj}Q_{mjw'}}, for\ w \neq w', & (3) \\
R_{mC_{mj}Q_{mjw}} > R_{mC_{mj'}Q_{mj'w'}}, or\ R_{mC_{mj}Q_{mjw}} < R_{mC_{mj'}Q_{mj'w'}}, for\ j \neq j', & (4) \\
R_{mC_{mj}Q_{mjw}} > R_{m'C_{m'j'}Q_{m'j'w'}}, or\ R_{mC_{mj}Q_{mjw}} < R_{m'C_{m'j'}Q_{m'j'w'}}, for\ m \neq m', & (5) \\
R_{mC_{mj}Q_{mjw}} + E_{mC_{mj}}.DT_{Cmj} \geq 0. & (6) \\
\quad S' = \{z : \text{the set of nodes z that have been scheduled}\}
\end{cases}
$$

DR : DT

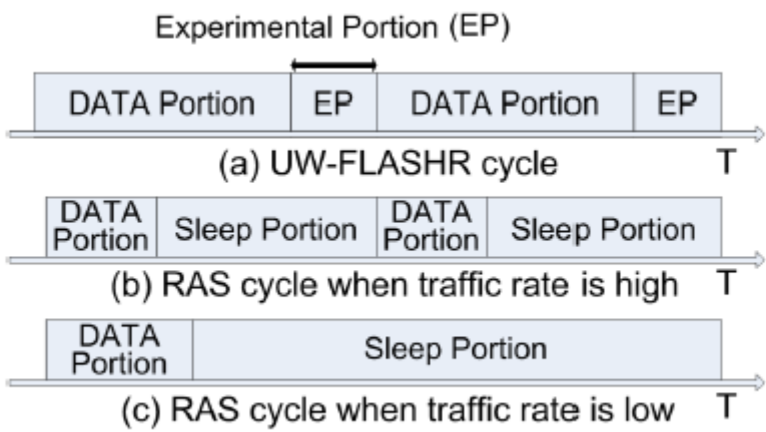
DR : IR

One DR

DT >= 0

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

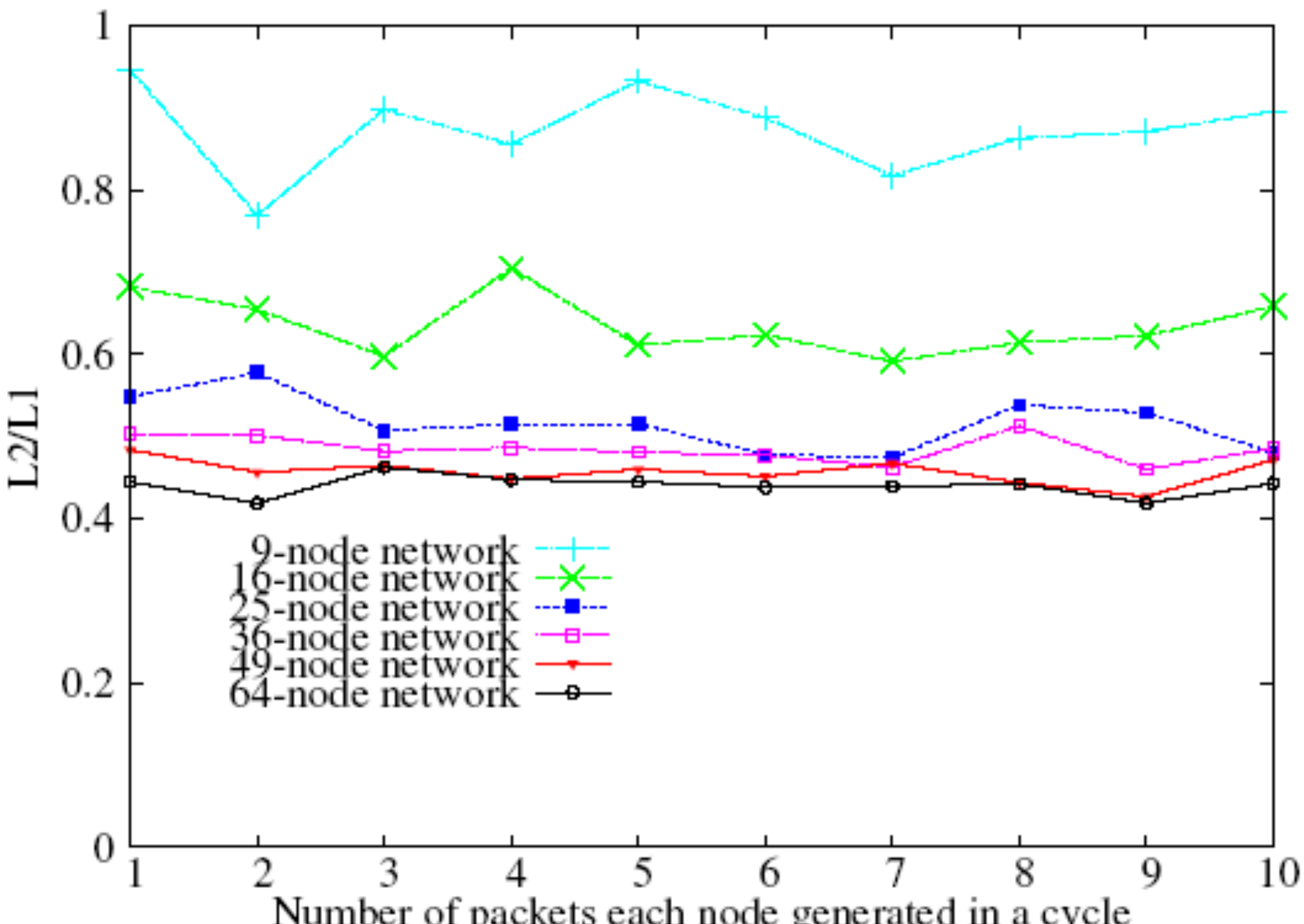
Part 1: Congestion Avoidance Algorithm

Algorithm 3 CalcSchedule() function at the BS

```
Parent = BS; hop = 1
while hop ≤ maxhop do
    while Parent has children do
        while Parent has data to receive from its children do
            if Parent is idle in the Slot then
                Parent searches its entire children set to alternatively find a child
                whose transmission results in its reception at the Slot
                if Parent finds a suitable child then
                    schedule the child's transmission and the related reception and
                    interference
                    break searching
                end if
            end if
            Parent fetches the next Slot for reception
        end while
        fetch the next Parent to schedule reception
    end while
    hop = hop + 1
end while
```

Performance Evaluation

- Schedule ratio: the lower bound schedule length (L_2) divided by the RAS schedule length (L_1)



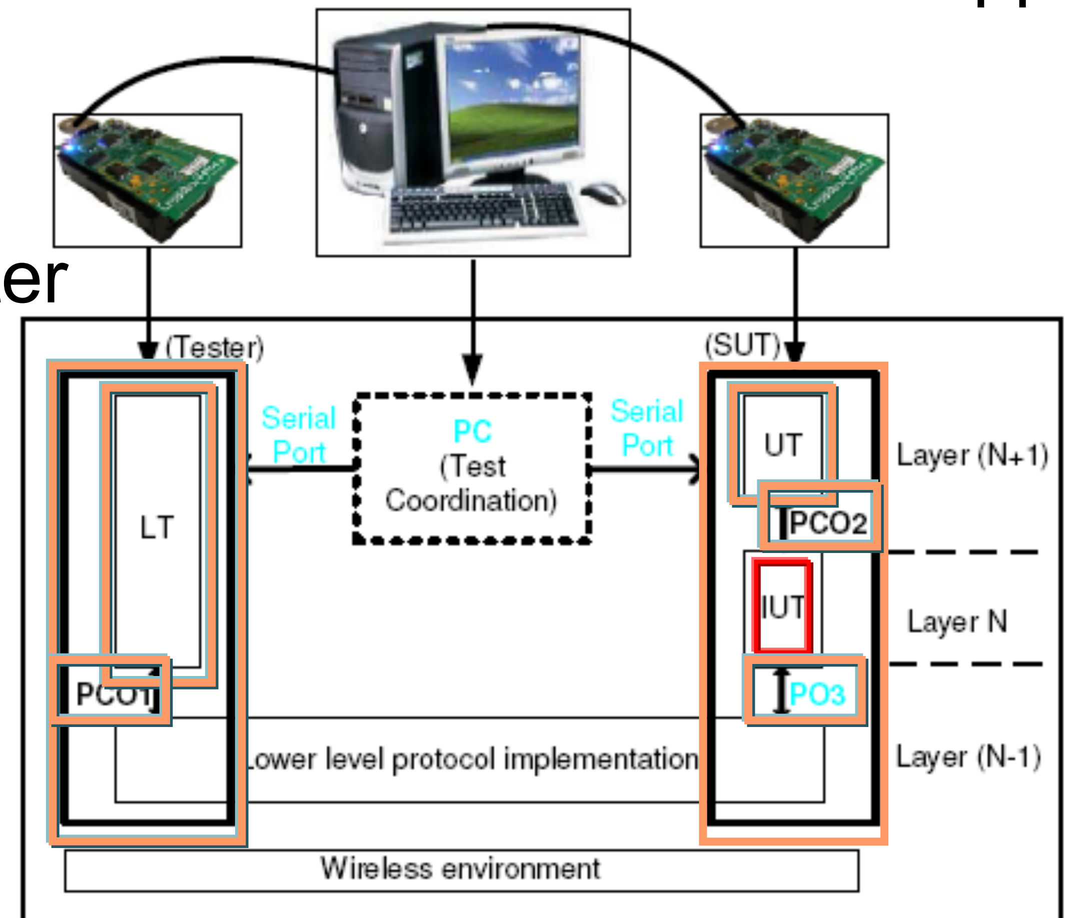
Part 2: RealProct Architecture

- Point of Control & Observation

SUT (System Under Test)

Upper Tester

Lower Tester



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).

user@instant-contiki: ~/contiki-2.4/RealProct/testcase4/testcase

File Edit View Terminal Tabs Help — test case 4 Client

```
user@instant-contiki:~/contiki-2.4/RealProct/testcase4/testcase$ reset0; dump0
MSP430 Bootstrap Loader Version: 1.39-telos-7
Use -h for help
Reset device ...
connecting to /dev/ttyUSB0 (115200) [OK]
Contiki 2.4 started. Node id is not set.
Rime started with address 155.188
MAC 00:12:74:00:13:7b:bc:9b CSMA X-MAC, channel check rate 4 Hz, radio channel 26
uIP started with IP address 172.16.155.188
Starting 'Example protosocket client'
client*****************************************
I am client with IP address 172.16.155.188
eventhandler() ev == PROCESS_EVENT_TIMER, timeout_number is 0
runtest: step 1
void sendSYN()
tcp packet, please print out more pkt info.
srcIP 172.16.155.188,destIP 172.16.137.204,vhl:0x45,tos:0x00,total length:0x002C,identification:0x0
001,ipoffset:0x0000,ttl:0x40,protocol:0x06,IPchecksum:0x21FD,
tcp header: srcPort:1025,destPort:0,seqno:1 1 1 1,ackno:2 2 2 2,tcpoffset:0x60,flag:0x02,window:0 4
8,TCPchecksum:0xCA13,urg:0 0,TCP_OPT_MSS option indicate MSS is:48,
eventhandler() ev == PROCESS_EVENT_TIMER, timeout_number is 1
timeout, test case fail
```

Detailed log
(Do not read it until needed in bug analysis)

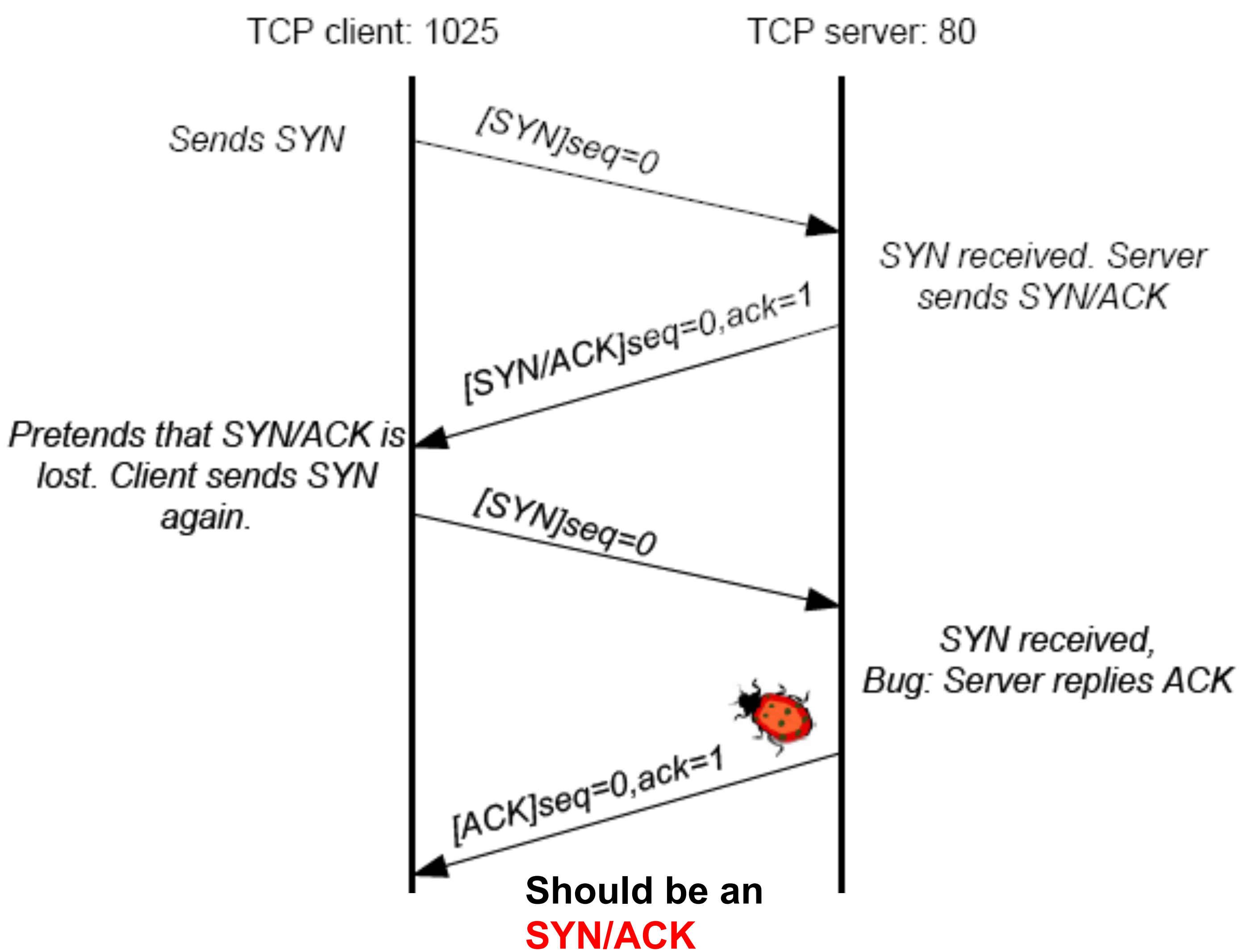
Bug: Client expects RST response while it receives no reply.

--> Test result. Bug: Server does not reply RST when it should do so.

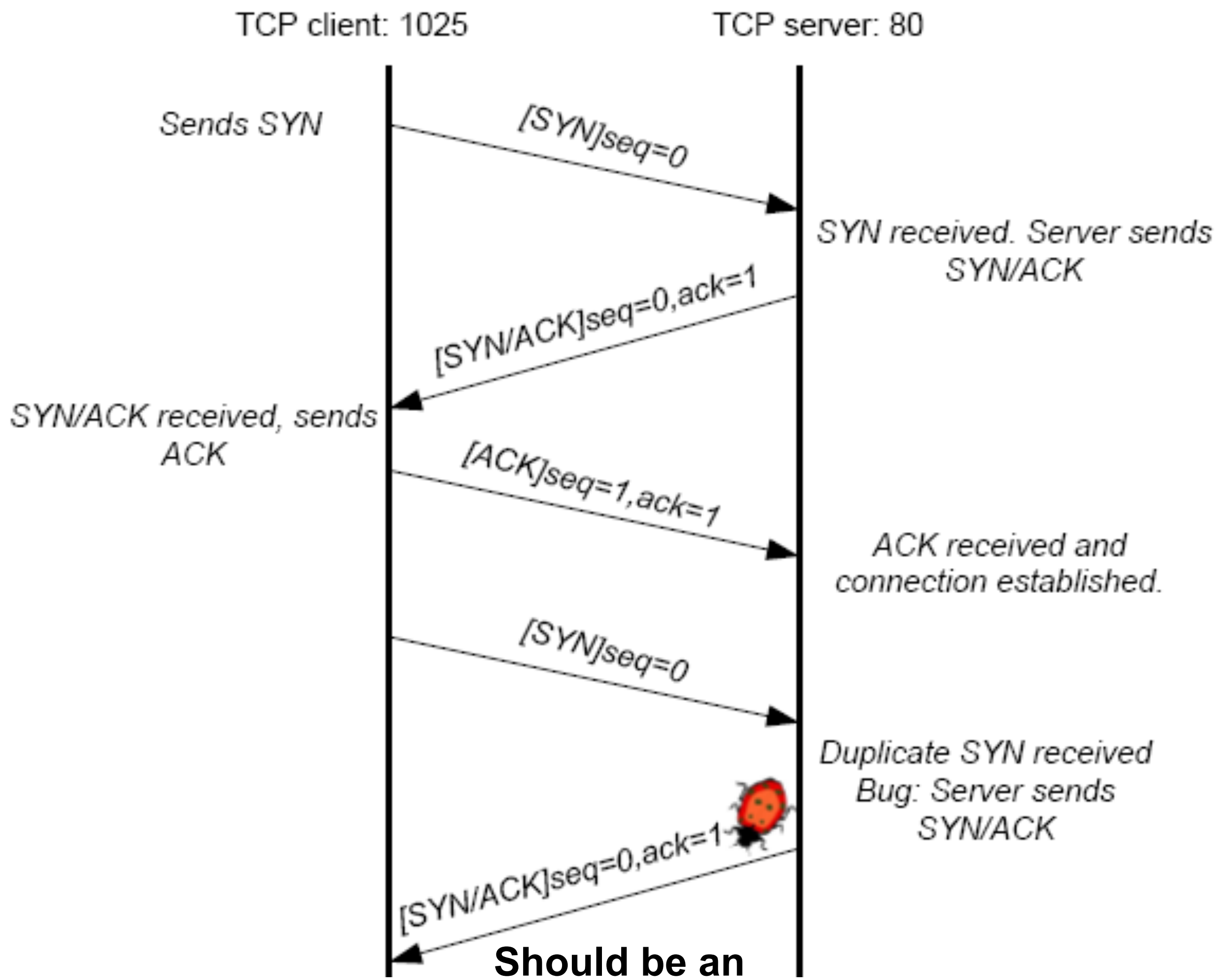
Part 2: Codes that Cause Bugs

```
// Make sure that the TCP port number is not zero.
if(BUF->destport == 0 || BUF->srcport == 0)
{
    UIP_LOG("tcp: zero port.");
    goto drop;
}
```

Part 2: Repeat Bug – SYN Packet Loss



Part 2: Repeat Bug –SYN Packet Duplication



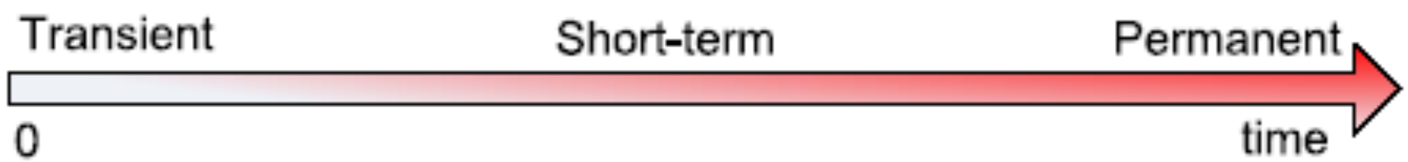
Should be an empty ACK

Part 2: Dynamic Test Execution

Calculate $n_{min} = \lceil \lg^{E}_{L_0} \rceil, n = n_1 = n_2 = 0$
while $n \le n_{min}$ **do**
  Execute the test case
  if Execution result is pass **then**
    n_1++
  else
    n_2++
  end if
end while
loop
  if $n_1 > n_2$. **then**
    Calculate $P(FN) = \binom{n}{n_1} L_0^{n_1} (1 - L_0)^{n_2}$
    if $P(FN) \le E$ **then**
      break //**end test execution**
    else
      Execute the test case and increase n_1 or n_2 according to the result
    end if
  else if $n_1 < n_2$. **then**
    Calculate $P(FP) = \binom{n}{n_1} L_0^{n_2} (1 - L_0)^{n_1}$
    if $P(FP) \le E$ **then**
      break //**end test execution**
    else
      Execute the test case and increase n_1 or n_2 according to the result
    end if
  else if $n_1 = n_2$ **then**
    Execute the test case and increase n_1 or n_2 according to the result
  end if
end loop

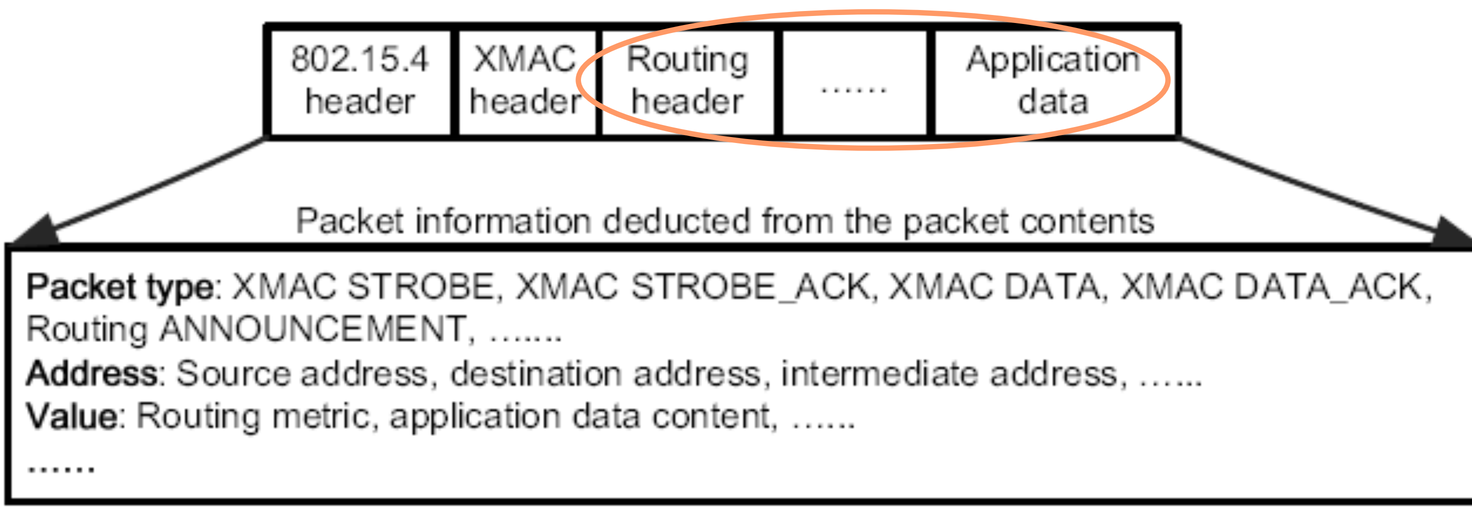
Part 3: Background - Failure Classification

- 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



Raw packet structure

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

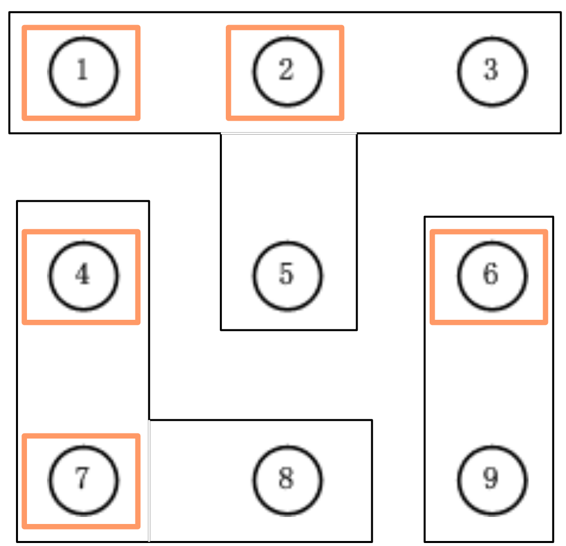
- For the data gathering application with routing protocol CTP and MAC protocol X-MAC:

Statistical rules for a typical WSN application

Layer	Statistical rules
Application layer	Rule 1. For BS, Z, the number of application data sent out, is 0.
Application layer	Rule 2. For sensor nodes other than BS, Z is within the application requirement.
Routing layer	Rule 3. For BS, the legal routing metric is 0.
Routing layer	Rule 4. For sensor nodes other than BS, routing metric value is legal.
Routing layer	Rule 5. For sensor nodes other than BS, no bidirectional data exchange exists, i.e., $Z * U = 0$
MAC layer	Rule 6. For sensor nodes other than BS, the number of each kind of MAC packets sent is normal, i.e., $Y \simeq U, X >$ or $\gg Z$.

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

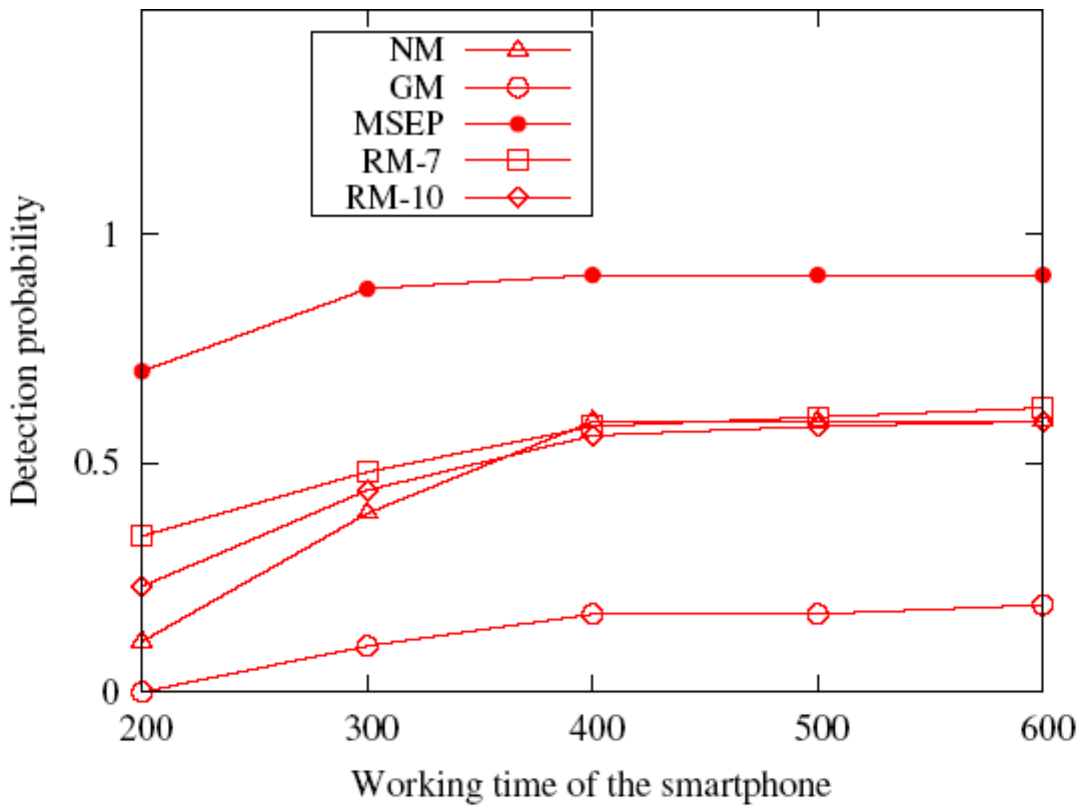


Part 3: Experiment Settings

- Sensei-UU: A Relocatable Sensor Network Testbed (2010)
  - 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]