Monitoring and Debugging of Deployed Sensor Networks

Matthias Ringwald, Kay Römer, ETH Zürich
Overview

- Wireless Sensor Network (WSN) Deployment
- State of the Art in WSN Debugging
- Non-Intrusive Debugging of WSNs
- Summary & Discussion
Wireless Sensor Networks (WSNs)

Network of many sensor nodes:
- battery powered
- wireless
- environmental sensors

Environmental monitoring
- Great Duck Island
- Glacier
- Volcano
WSN Life Cycle

Simulation, emulation → Lab testbed → Configuration & test → Operation

Deployment

Unexpected problems in deployed networks

- Moisture
- Depleted batteries
- Incorrect sensor readings
- Frequent connectivity changes
- ...

Matthias Ringwald, Kay Römer, Institut for Pervasive Computing
“Unexpected” Problems?

- Reasons
  - Simulation: does not capture all real-world effects
  - Testbed: few nodes, controlled environment
  - Users: Application experts, not system experts

⇒ Problems are inevitable and cannot be avoided

⇒ Get insight into deployed sensor network
State of the Art

Wireless network used for data gathering
- Nucleus: attribute query, event logging [Tolle2005]
- Sympathy: In-network state collection, statistic analysis [Ramanathan2005]

Additional code/serial communication
- Deployment support network (DSN): additional, temporary wireless network as backbone [Beutel2005]

Non-intrusive to deployed sensor node
- TinyOS message center: gateway packet monitor [2003]
Deployment Support Network

More Info:
btnode.ethz.ch

BTnut on BTnode3
- Cooperative Multithreaded OS + POSIX: Signals, Mutex, Semaphore, Device Driver...
- 260 KiBi RAM
- Bluetooth Radio
- Chipcon CC1000
Non-intrusive debugging of WSNs

Mission statement:
“Monitor & debug WSN by non-intrusive observation of radio communication”
Distributed Sniffer

Generic MAC listening

- Frequency, data rate
- Start-Of-Packet symbol
- Packet size

SINGLE NODE SNIFER IMPLEMENTED
NEXT: INTEGRATION IN DSN
Generic Packet Decoding

- Existing Solutions:
  - Top-Down: ASN.1 packet description
  - Bottom-Up: Ethereal network analyzer: decode code written in C

- New: “Attributed C Structs”
  - most WSN apps are developed in C
  - Structs are easy to understand/parsing
  - Extensions allow for encapsulated packets

```c
/** TinyOS message */
struct TOS_Msg {
    uint16_t addr;
    uint8_t  type;
    uint8_t  group;
    uint8_t  length;
    int8_t  data [29 ];
    uint16_t crc;
};
```
Data Analysis

- Topology Visualization: link-layer header info
- Statistics: re-implement Sympathy?
- Rule engine: assertions, failure tests

... eliminate bugs
Summary

• Inevitable problems during WSN deployment
• New tools needed to aid in debugging
• Current approaches alter behavior of nodes
• Our approach is non-intrusive, promising and can be added on the spot
Discussion

• Pro:
  • non-intrusive
  • access to all communication
  • allows to understand/ correct communication issues

• Contra:
  • multiple frequencies, frequency hopping: hard
  • Internal state can only be inferred

• Extension:
  • nodes could send additional/intrusive debugging data

Thanks!
Example Packet Description

```c
struct Packet {
    u_int16 src;
    u_int16 dest;
    u_int8 type;
    u_int8 size;
    u_int8 data[size];
    u_int16 CRC;
};

struct RoutingPacket
    : Packet.data ( type == 1 ) {
    u_int8 nrHops;
    u_int16 nextHop;
};
```