Sensor Network Application Development
ZIGBEE CONCEPTS 3

Cruise Summerschool
Johannes Kepler University
November 5 - 7, 2007, Linz / Austria
Dipl.-Ing. Andreas Riener
riener@pervasive.jku.at
Overview

Structure of this lesson

ZigBee-Application development

- Communication between Motes and Applications
- Radio Link Distance Limit in TinyViz
- Multihop Sensor-Network with Zigbee / TinyOS
ZigBee-Concepts

Communication between Motes and Applications
Connecting a single Application to the Mote Network

Access to network or serial ports is usually exclusive: only 1 application can connect to the motes.

Network or Serial Link

Single-Hop RF Communication

Multi-Hop RF Communication

Application 1

PC

Node 0

Programming Board (Network or Serial)

Node 1

Node 2

Node 3

Node 4
Connecting Multiple Applications to the Mote Network

- PC
- Application 1
- Application 2
- SerialForwarder (connected to Network or Serial Port)
- Single-Hop RF Communication
- Multi-Hop RF Communication
- Node 0
- Programming Board (Network or Serial)
- Node 1
- Node 2
- Node 3
- Node 4

Application SerialForwarder: Relays all traffic to and from the mote network
Network or Serial Link
Connecting Applications to a TinyViz Simulation

TinyViz automatically opens a SerialForwarder to node 0 in the TOSSIM Simulation.

For applications it makes no difference if the running SerialForwarder is connected to a simulation or actual motes.
Messages from Mote to Application

Send a message from a mote to the PC
If the mote is not connected to the programming board (e.g. it is a wireless, deployed one)
  - Send the message to the mote that is connected to the programming board
    (It's best to use node 0 because TOSSIM always connects node 0 to the SerialForwarder)
  - This node has to forward all incoming messages / send the message to the serial port
    - Basic component is UART
    - Use component GenericComm and target address TOS_UART_ADDR

Receiving a message from the PC
Use component GenericComm
  - No difference if receiving a RF packet
  - If the message is meant for another mote, forward it

Note
Limitation/distinction, because TOSSIM allows only 1 application/node
In common applications, "TOSBASE" is additionally installed on node 0
  - Broadcasts all messages received from the serial port
  - Forwards all incoming messages to the serial port
Communication with Motes in Java Applications

Create a MoteIF object (net.tinyos.message.MoteIF)

- Connects to motes through the serial port, TCP-connections or the SerialForwarder (SerialForwarder by default, if no other target is specified in the environment variable MOTECOM)
- Provides Listeners for incoming packets
- Enables an application to send messages
- Details: <tinyos-dir>/tools/java/net/tinyos/message/MoteIF.java

Requires a message class: Generated by 'mig' from the message header file

```c
typedef struct ExampleMsg {
    uint16_t value;
} ExampleMsg;

enum {
    AM_EXAMPLEMSG = 1
};
```

```java
class ExampleMsg extends net.tinyos.message.Message {
    // ...
    public int get_value();
    public void set_value(int value);
}
```

$ mig java -java-classname=ExampleMsg ExampleMsg.h ExampleMsg -o ExampleMsg.java
Java Example Application

class: Example.java

```
import net.tinyos.message.*;

public class Example implements MessageListener {

    public static void main(String args[]) throws java.io.IOException {
        MoteIF mote = new MoteIF();
        mote.registerListener(new ExampleMsg(), new Example());
        ExampleMsg msg = new ExampleMsg();
        msg.set_value(0);
        mote.send(MoteIF.TOS_BCAST_ADDR, msg);
    }

    public void messageReceived(int dest_addr, Message msg) {
        if (msg instanceof ExampleMsg) {
            ExampleMsg exMsg = (ExampleMsg)msg;
            System.out.println(exMsg.get_value());
        }
    }
}
```

Implement MessageListener
Create MoteIF object and register a new Listener for ExampleMsg
Create and send a new message. Careful: Although we set the broadcast address as target, the message will only be sent to the one connected node. This node has to forward incoming messages
Incoming message event handler
ZigBee-Concepts

Radio Link Distance Limit in TinyViz
TinyViz: Radio Link Distance Limits

- Activate "Radio Model" plugin
- Activate "Auto Update"
- Set to "Fixed Radius" (10.0)
- Arrange the nodes in a suitable manner; Nodes that are in radio range of the selected node are connected with green arrows
ZigBee-Concepts

Multihop Sensor-Network with Zigbee / TinyOS
Multihop-Connections: Why?

Nodes that are too far apart to communicate directly need relay stations in between. To achieve this, every node has to act as a gateway for other nodes.

Sensor request from Server to Node 3:
Server → Node 0 → Node 1 → Node 3

Reply from Node 3 to Server:
Node 3 → Node 1 → Node 0 → Server
MultiHop Sensor-Network: Example

Sensor network with 1 server and many sensor nodes

Nodes are stationary: We assume they never move after system activation

Two message types

- **Sensor messages** are produced by sensor nodes and they are always sent to the server
- **Command messages** originate from the server and their target is either a single node or the whole network (broadcast)
- There are no node-to-node messages
MultiHop Sensor-Network: Example

At system startup, find shortest path to server

- Identify all nodes in radio range (neighbours)
  - Broadcast a "discovery"-packet
  - Nodes that receive this packet and already have a connection to the server return an "I am a potential gateway"-packet, including the length of their path
- Use neighbouring node with shortest path length as gateway (Active Link)
MultiHop Sensor-Network: Example

Transmission of a sensor message from a node to the server

- Sensor nodes send message to its gateway
- The gateway node relays the message to its own gateway
- This continues until the message reaches the server

MultiHop Sensor-Network: Example
MultiHop Sensor-Network: Example

Transmission of a command message from the server to 1 or all sensor nodes

- First the message is sent to Node 0. From there, repeat on every node:
- If the message target is this node or all nodes, execute the command
- If the message target is another node or all nodes, the message is broadcasted
MultiHop Sensor-Network: Example

Problem

- Node 0 broadcasts message – Node 1 and 2 receive message
- Node 1 broadcasts message – Node 0, 2, 3 receive message
- Node 3 executes command, Node 0 and 2 broadcast the command again!
- Infinite loop! We have to find a way to recognize messages we already sent
MultiHop Sensor-Network: Example

Transmission of a command message from the server to 1 or all sensor nodes

- The command message includes a counter-field. The server increases this field for every new command

- First the message is sent to Node 0. From there, repeat on every node:
  - If the stored counter value of the node is equal or higher than the message counter-field, ignore the message

  - If the message target is this node or all nodes, execute the command

  - If the message target is another node or all nodes, the message is broadcasted. The counter-field of the message is stored
MultiHop Sensor-Network: Example

Transmission of a command message from the server and broadcast (execute) to all sensor nodes

Time

0

packet counter = 1

Node X
counter = 0
ACCEPT

Node Y
counter = 0

1

packet counter = 1

Node X
counter = 1
BROADCAST

Node Y
counter = 0
ACCEPT

2

packet counter = 1

Node X
counter = 1
REJECT

Node Y
counter = 1
BROADCAST

C

C

C
Sensor Network Application Development

...for Participants of "KV Spezielle Kapitel aus Pervasive Computing (367.055)"

What to submit?

- ZIP archive for each of the exercise sheets 2 to 4 (exercise 1 is not requested)
  - Use the document templates from www.pervasive.jku.at/temp/SNAD
  - Electronic protocol (.doc, .tex) containing a textual description of your solution idea, a component / architecture overview (see next slide), crucial source code snippets, test cases including screenshots of results from TOSSIM/TinyViz
  - Application directories (*.nc, makefiles, etc.)

Deadline for providing your documents (archive files)
- Wednesday December 5, 2007, Noon (12 AM)

Submit to (exclusively in electronic form, one ZIP for each exercise sheet):
- riener@pervasive.jku.at
Electronic protocol: Component and Application Overview

a) File / Component overview

- `makefile`
- `module: CounterM.nc`
- `configuration: Counter.nc`
- `interface: CounterComm.nc`
- `configuration: CounterCommC.nc`
- `module: CounterCommM.nc`
- `header: CounterMsg.h`

b) Application architecture

- `MAIN`
- `BlinkM`
- `LedsC`
- `TimerC`

Connections:
- `Leds` from `BlinkM`
- `Timer` from `TimerC`
Sensor Network Application Development
ZIGBEE CONCEPTS 3

Cruise Summerschool
Johannes Kepler University
November 5 - 7, 2007, Linz / Austria
Dipl.-Ing. Andreas Riener