Contiki COOJA Crash Course

Thiemo Voigt
Swedish Institute of Computer Science
Networked Embedded Systems Group
thiemo@sics.se

SWEDISH INSTITUTE OF COMPUTER SCIENCE SICS

Outline

- Contiki Overview presentation (about 15 min)
- Contiki hands-on with cool Contiki features
  - Some basics
  - Rime stack
  - COOJA – timeline
  - MSPSim
  - Shell & Coffee file system
  - Larger exercise: neighbour discovery

Sorry, no IP, but you get the basis for doing IP with Contiki

Acknowledgements: The Contiki Project

- Core group
  - Adam Dunkels (leader), Oliver Schmidt, Fredrik Österlind, Niclas Finne, Joakim Eriksson, Nicolas Tsiftes, Takahide Matsutsuka
- Developers
  - Zhitao He, Simon Barner, Simon Berg
  - 5+ incoming
- Contributors
  - Thiemo Voigt, Björn Grönvall, Tony Nordström, Matthias Bergvall, Groepaz, Ullrich von Bassewitz, Lawrence Chitty, Fabio Fumi. Matthias Domin, Christian Groessler, Anders Carlsson, Mikael Backlund, James Dessart, Chris Morse, …

.se WISENET CNS VINNOVA

CONET COOPERATING OBJECTS NETWORK OF EXCELLENCE

ITEA 2 INFORMATION TECHNOLOGY FOR EUROPEAN ADVANCEMENT

GINSENG

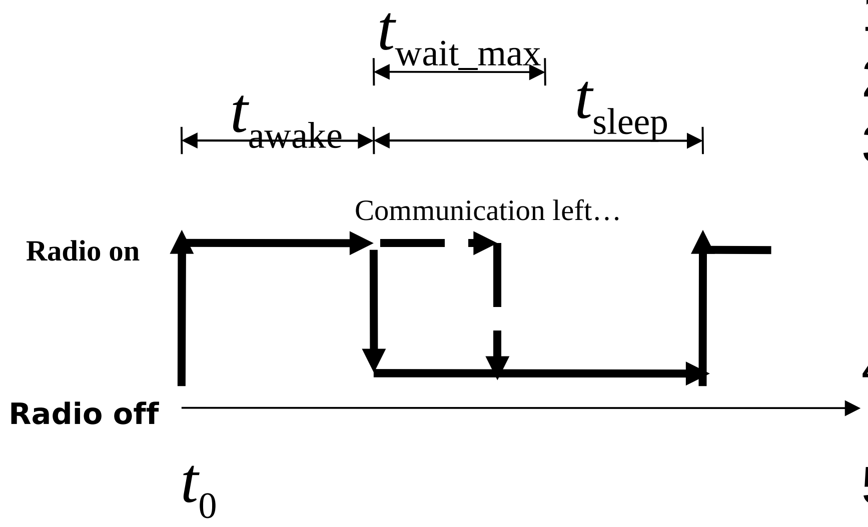
Contiki Overview

- Contiki – a dynamic operating system for networked embedded systems
  - Main author and project leader: Adam Dunkels
- Small memory footprint
  - Event-driven kernel, multiple threading models on top
- Designed for portability
  - Many platforms (ESB, Tmote Sky etc.), several CPUs in CVS code
- Used in both academia and industry
  - Cisco and Atmel have joined Contiki project

Contiki Pioneering features:

- TCP/IP for low power wireless (EWSN 2004, SenSys 2007, IPSN tutorial 2009...)
  - 2005: 6lowpan
  - IPv6 ready since October 2008
- Dynamic loading (Emnets 2004)
  - 2005: SOS, Mantis, TinyOS, …
- Threads on top of event-driven system (Emnets 2004)
  - 2006: TinyThread, …
- Protothreads (SenSys 2006), power profiling (Emnets 2007), Coffee file system (IPSN 2009)

Protothreads – Simplifying Event-driven Programming



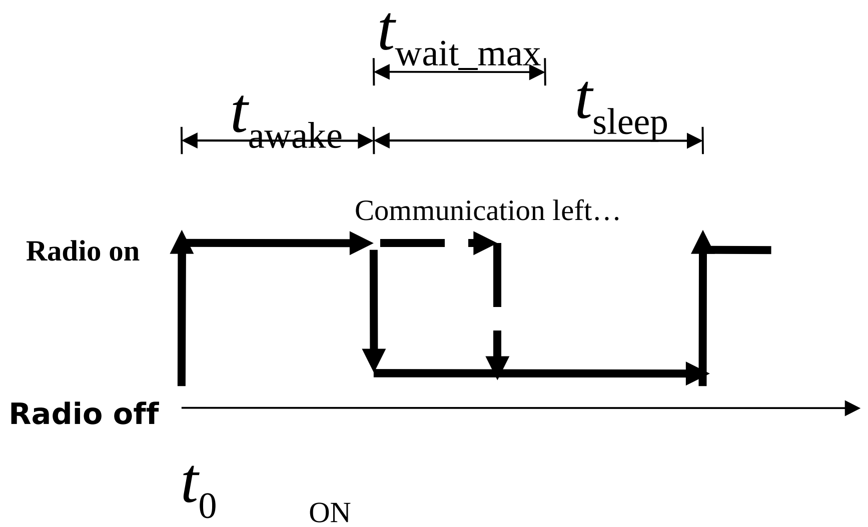
1. Turn radio on.
2. Wait until *t* = *t*_0 + *t*_awake.
3. If communication has not completed, wait until it has completed or *t* = *t*_0 + *t*_awake + *t*_wait_max.
4. Turn the radio off. Wait until *t* = *t*_0 + *t*_awake + *t*_sleep.
5. Repeat from step 1.

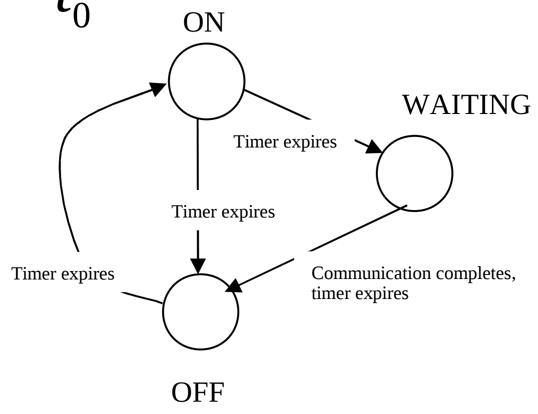
No blocking wait!

Problem: with events, we cannot implement this as a five-step program!

SWEDISH INSTITUTE OF COMPUTER SCIENCE SICS

Event-driven state machine implementation: messy

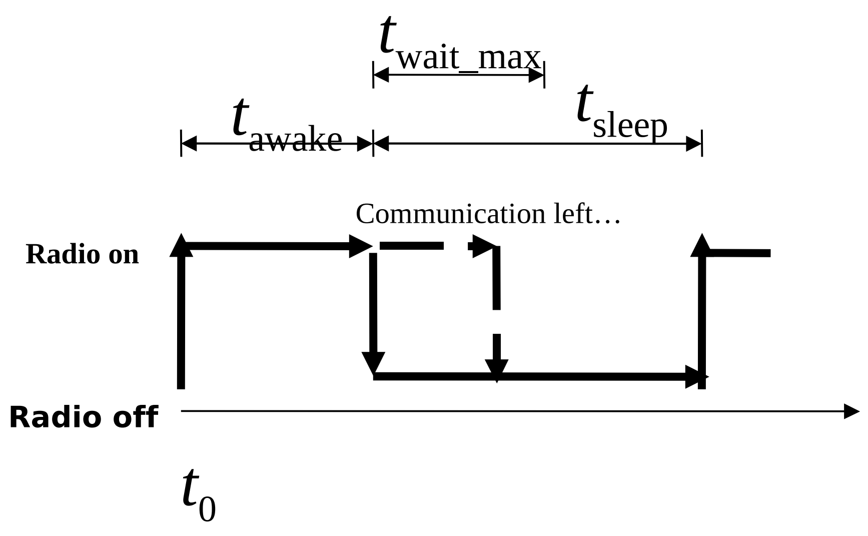




```
enum {ON, WAITING, OFF} state;

void eventhandler() {
  if(state == ON) {
    if(expired(timer)) {
      timer = t_sleep;
      if(!comm_complete()) {
        state = WAITING;
        wait_timer = t_wait_max;
      } else {
        radio_off();
        state = OFF;
      }
    }
  } else if(state == WAITING) {
    if(comm_complete() ||
       expired(wait_timer)) {
      state = OFF;
      radio_off();
    }
  } else if(state == OFF) {
    if(expired(timer)) {
      radio_on();
      state = ON;
      timer = t_awake;
    }
  }
}
```

Protothreads-based implementation



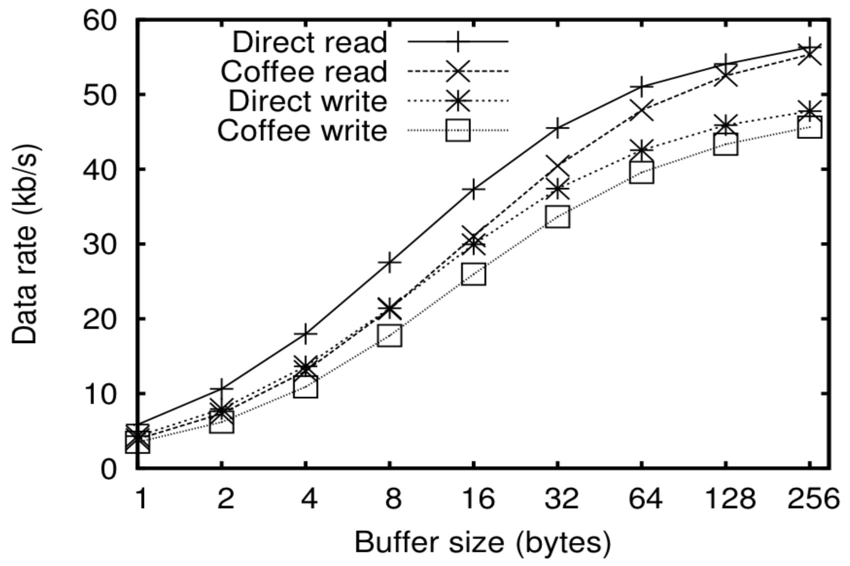
```
int protothread(struct pt *pt) {
  PT_BEGIN(pt);
  while(1) {
    radio_on();
    timer = t_awake;
    PT_WAIT_UNTIL(pt, expired(timer));
    timer = t_sleep;
    if(!comm_complete()) {
      wait_timer = t_wait_max;
      PT_WAIT_UNTIL(pt, comm_complete()
                   || expired(wait_timer));
    }
    radio off();
    PT_WAIT_UNTIL(pt, expired(timer));
  }
  PT_END(pt);
}
```

- Code uses structured programming (**if** and **while**), mechanisms evident from code

  → Prototthreads make Contiki code **nice**

The Coffee file system [IPSN 2009]

- Flash-based file system
- open(), read(), seek(), write(), close()
- Very lightweight
  - 5 kb ROM, < 0.5 kb RAM
- Very fast
  - More than 92% of raw flash throughput



SWEDISH INSTITUTE OF COMPUTER SCIENCE SICS

Interactive shell

- Network debugging, performance tuning
- Leverage UNIX-style pipelines
- Network commands
- Direct serial connection, or over Telnet/TCP
- A generic interface for higher level applications
  - Automated interaction, scripting

Protocol stacks

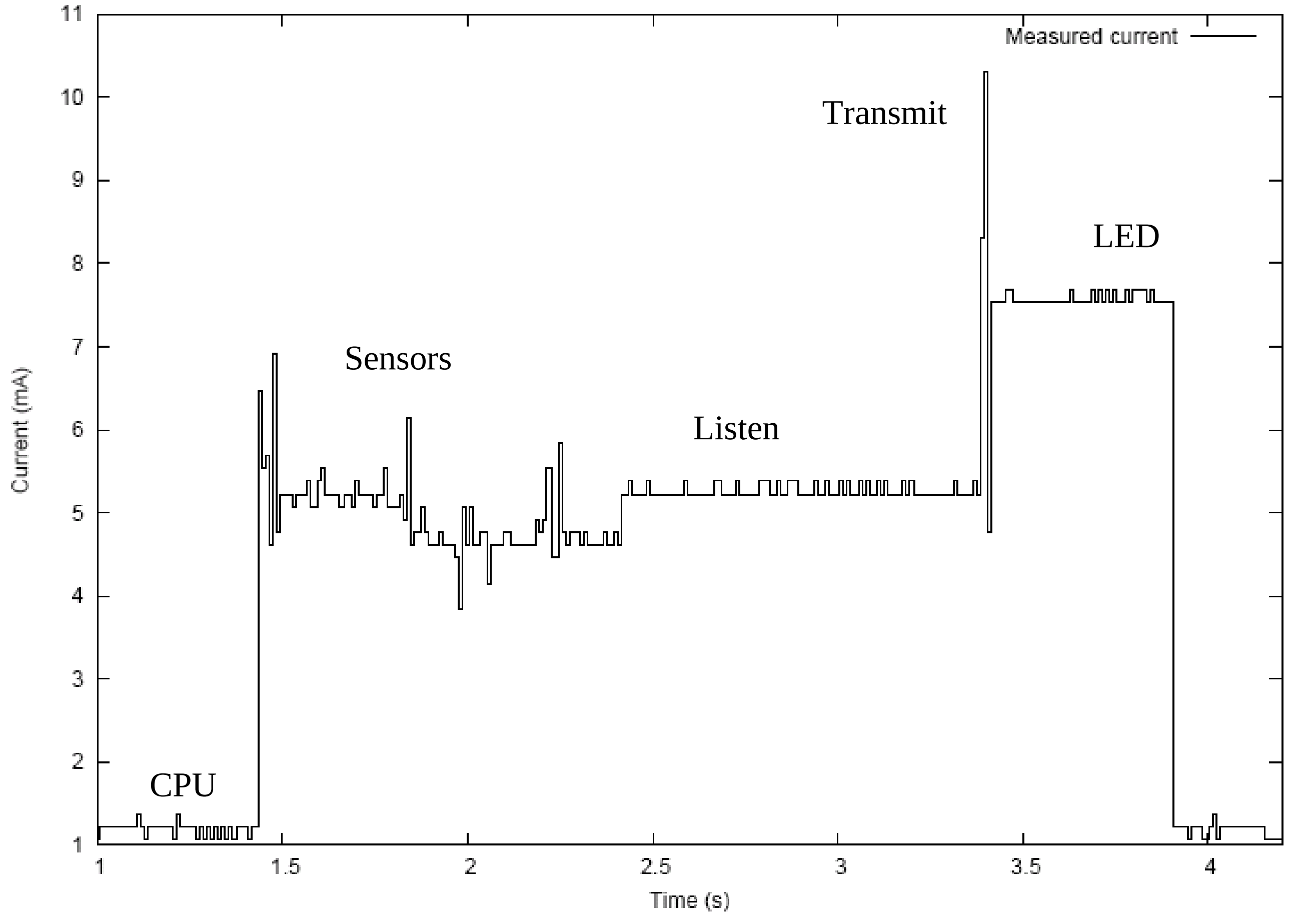
Protocol stacks in Contiki:

- uIP: world's smallest, fully compliant TCP/IP stack
  - Both Ipv4 and IPv6
- Rime stack: protocol stack consisting of small layers built on top of each other
  - From single-hop broadcast to reliable multi-hop flooding
- MAC layers in Contiki:
  - Power-saving X-MAC (SenSys 2006)
  - NULLMAC
  - Low power probing (IPSN 2008)
  - Simple TDMA MACs

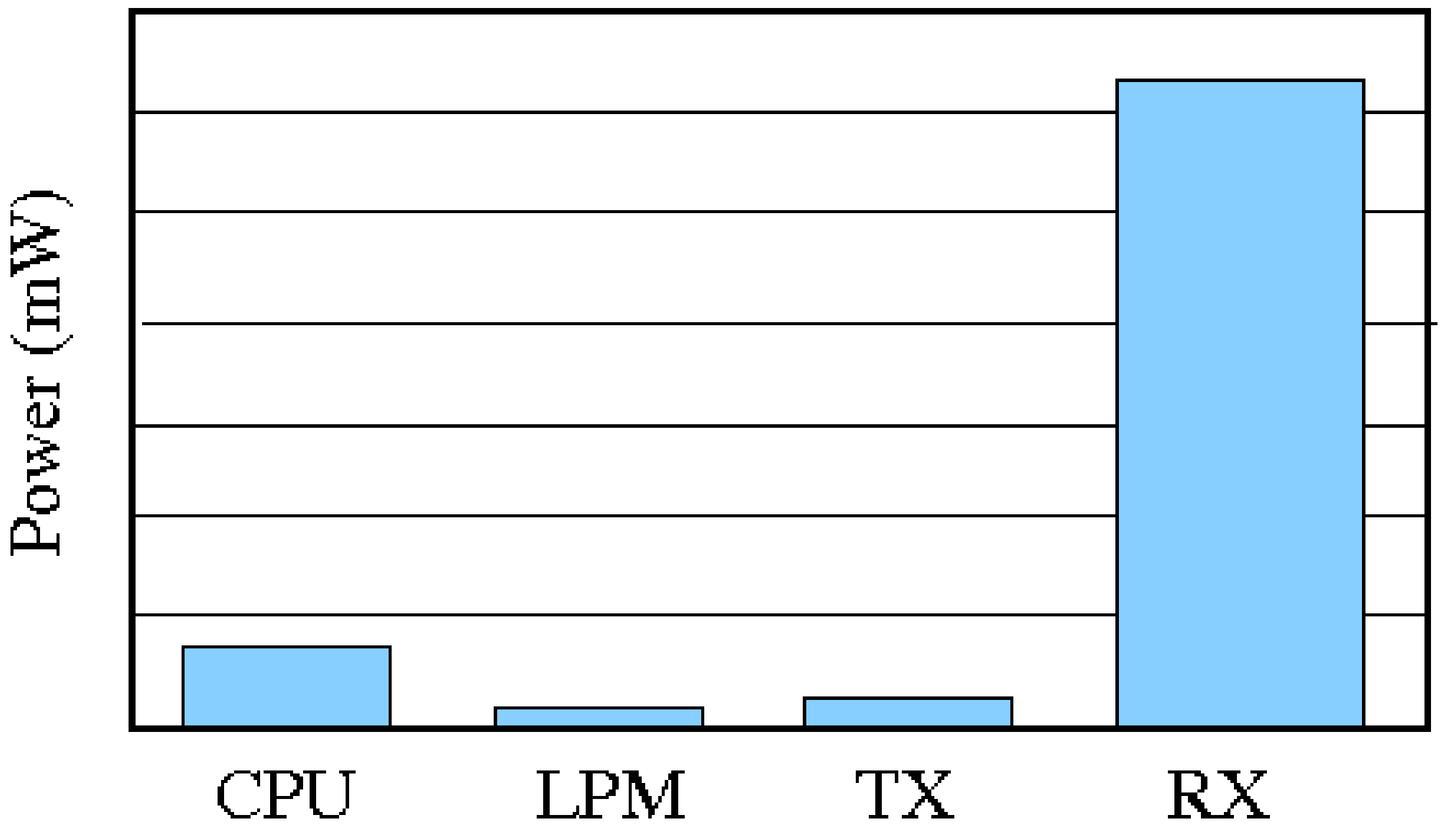
Power Profiling [EmNets 2007]

- Software-based
  - Zero-cost hardware
  - Zero-effort deployment
  - Just add some lines in source code
- Good accuracy, low overhead
- Enables network-scale energy profiling
- Enables energy-aware mechanisms
- Recent Contiki feature:
  - Per packet power profiling

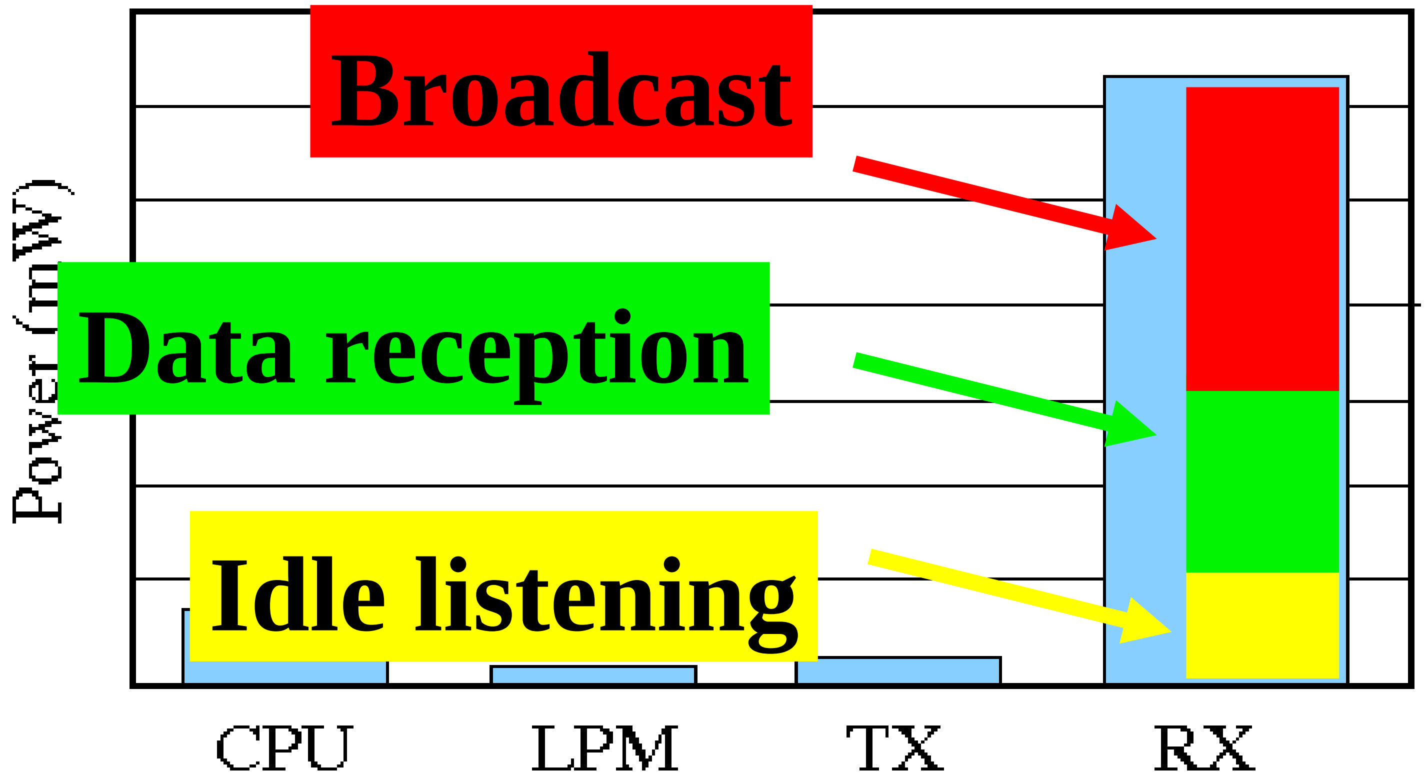
Linear current draw



Per-component Power Profiling



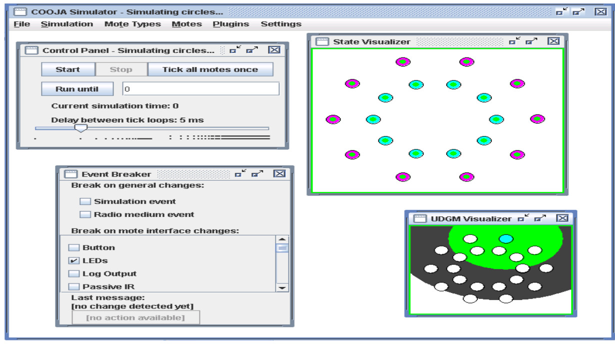
New: Per-packet Power Profiling



Simulators

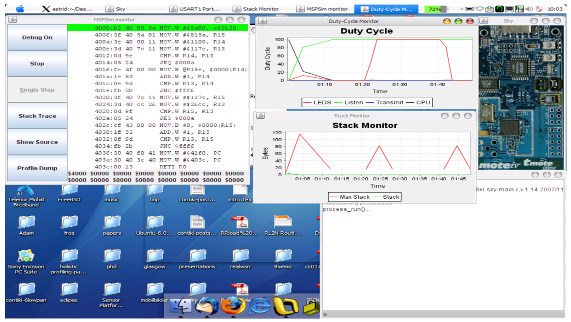
COOJA: extensible Java-based simulator

- Cross-level: Contiki nodes (deployable code), Java nodes, emulated MSP430 nodes



MSPSim: sensor node emulator for MSP430-based nodes:

- Tmote Sky, ESB
- Enables cycle counting, debugging, power profiling etc.
- Integrated into COOJA or standalone
- COOJA/MSPSim enables also interopability testing for MSP-based platforms (e.g. IPv6 interop testing)
  - New: MicaZ



Hands-on

You should have or should do now:

- Install vmplayer
- instant Contiki 2.3 and drivers
- If your keyboard is strange: System->Preferences->Keyboard->Layout... then start new terminal
- Turn off beeping: System->Preferences->Sound...

What you need to do (to use Tmote Skys):

- cd tools/sky and do
  ```
  chmod u+x tmote-bsl-linux
  ```
- To test: go to tools/cooja and type “ant run”

Contiki Hands-On: Hello World

```
/* Declare the process */
PROCESS(hello_world_process, “Hello world”);
/* Make the process start when the module is loaded */
AUTOSTART_PROCESSES(&hello_world_process);

/* Define the process code */
PROCESS_THREAD(hello_world_process, ev, data) {
  PROCESS_BEGIN();              /* Must always come first */
  printf(“Hello, world!\n”);    /* Initialization code goes
here */
  while(1) {                    /* Loop for ever */
    PROCESS_WAIT_EVENT();       /* Wait for something to
happen */
  }
  PROCESS_END();                /* Must always come last */
}
```

Hello World on Tmote Sky

To download Hello World on a Tmote, place Tmote in a USB and it will appear in the top of instant Contiki as “Future Technologies Device”. Click on name to connect it to Instant Contiki.

The do:

```
cd examples/hello-world
make TARGET=sky hello-world.upload
```

When the compilation is finished, the uploading procedure starts (LEDS blink like crazy).

You can see the output of the program by logging into the node

```
make login TARGET=sky
```

Press the reboot button to see some output

Building Hello World

- cd examples/hello-world
- make TARGET=native hello-world.native
  start with ./hello-world.native
- make TARGET=sky
  - Builds monolithic system image for sky
- make TARGET=sky hello-world.upload
  - Build and upload system image for sky
- make TARGET=esb hello-world.upload
  - Build and upload image for ESB
- Make TARGET=sky hello-world.mspsim
  - Build image and starts MSPSim with it, cool!!

Contiki directories

- contiki/core
  - System source code; includes (among others)
    - net: rime, macs etc;
    - sys: processes
- contiki/examples
  - Lots of nice examples, write your apps here
- contiki/apps
  - System apps (telnet, shell, deluge)
- contiki/platform
  - Platform-specific code:
    - platform/sky/contiki-sky-main.c
    - platform/sky/contiki-conf.h

Contiki directories

- contiki/cpu

- CPU-specific code: one subdirectory per CPU

- contiki/tools
    - e.g. cooja, start with “ant run”
    - tools/sky contains serialdump and other useful stuff

Timers in Contiki

- struct timer
  - Passive timer, only keeps track of its expiration time
- struct etimer
  - Active timer, sends an event when it expires
- struct ctimer
  - Active timer, calls a function when it expires
- struct rtimer
  - Real-time timer, calls a function at an exact time

Events and Processes

PROCESS_WAIT_EVENT();

Waits for an event to be posted to the process

PROCESS_WAIT_EVENT_UNTIL(condition c);

Waits for an event to be posted to the process, with an extra condition. Often used: wait until timer has expired

PROCESS_WAIT_EVENT_UNTIL(etimer_expired(&timer));

PROCESS_POST(...) and PROCESS_POST_SYNCH(..)

Post (a)synchronous event to a process.

The other process usually waits with PROCESS_WAIT_EVENT_UNTIL(ev == EVENTNAME);

New events can just be defined e.g. #define EVENT_EX 1

First Task: Hello World in MSPSim: Section 4.1

```
cd examples/hello-world
make TARGET=sky hello-world.mspsim
```

Should start MSPSim.

You can:

- look at output
- Press buttons. Does anything happen?

Next two slides in wrong order in printed version, sorry

Task: Extend Hello World (on MSPSim, Section 4.2)

To wait for an event and when event happens:

- toggle with leds and print something

```
cd examples/hello-world
make TARGET=sky hello-world.mspsim
```

Code parts on next-next slides

Extend Hello World to toggle LEDS and print something in MSPSim

```
/* Declare the process */
PROCESS(hello_world_process, “Hello world”);
/* Make the process start when the module is loaded */
AUTOSTART_PROCESSES(&hello_world_process);

/* Define the process code */
PROCESS_THREAD(hello_world_process, ev, data) {
  PROCESS_BEGIN();                 /* Must always come first */
  printf(“Hello, world!\n”);  /* Initialization code goes
here */
  while(1) {                       /* Loop for ever */
    PROCESS_WAIT_EVENT();          /* Wait for something to
happen */
    leds_toggle(LEDS_ALL);   /* must include “dev/leds.h”*/
   /* add a printf here */
  }
  PROCESS_END();                   /* Must always come last */
}
```

SWEDISH INSTITUTE OF COMPUTER SCIENCE SICS

Exercise: Let Hello World print every four seconds or when button is pressed

Can be done in MSPSim (recommended) or on Tmote Sky:

```
make TARGET=sky hello-world.mspsim
make TARGET=sky hello-world.upload
```

Etimer code that needs to be added at the right places:

```
static struct etimer et;
etimer_set(&et, CLOCK_SECOND*4);
PROCESS_WAIT_EVENT_UNTIL(etimer_expired(&et));
```

See examples/sky for button stuff (cooler: toggle LEDS when button is pressed) or next slide

Button Press Code Fragement

```
#include "contiki.h"
#include "dev/button-sensor.h"
#include "dev/leds.h"
#include <stdio.h> /* For printf() */

  button_sensor.activate();

    PROCESS_WAIT_EVENT_UNTIL(etimer_expired(&et) || ev ==
     sensors_event && data == &button_sensor);
    if (ev == sensors_event) {
      printf("button pressed");
      leds_toggle(LEDS_ALL);
    }
```

SWEDISH INSTITUTE OF COMPUTER SCIENCE SICS

Next task: count how often program prints (a protothread issue)

Add variable that counts how often the program has printed “Hello World”

Do you see the expected result?

If you feel like, download your new version on the node and check if it works

Protothread Limitations

- Automatic variables not stored across a blocking wait
  - Compiler does produce a warning (not always)
  - Workaround: use static local variables instead
- Constraints on the use of switch() constructs in programs
  - No warning produced by the compiler
  - Workaround: don’t use switches

- Note: Contiki processes are protothreads

What we have learned

- How a simple Contiki program looks
- Etimers
- The nice WAIT_UNTIL
- Use static for variables
- How to run autostart and run programs in MSPSim (useful for single-node programs)

COOJA

- cd tools/cooja
- Type “ant run”
- … see hand-outs/follow demo on projector
- Important: choice of node type
  - Sky (emulated tmote skys): more accurate (includes MAC layer), probably better support, slower
  - Contiki: scales better, MAC is simulated
- Simulations can be saved and reloaded
- As example, choose examples/rime/example-abc.c

SWEDISH INSTITUTE OF COMPUTER SCIENCE SICS

Networking with the Rime Stack

COOJA has two stacks: IP and Rime

For the IP tutorial (outdated), see www.sics.se/contiki

The Rime stack is a layered protocol stack. Depending on the type of functionality you want, call the functions at the right layer.

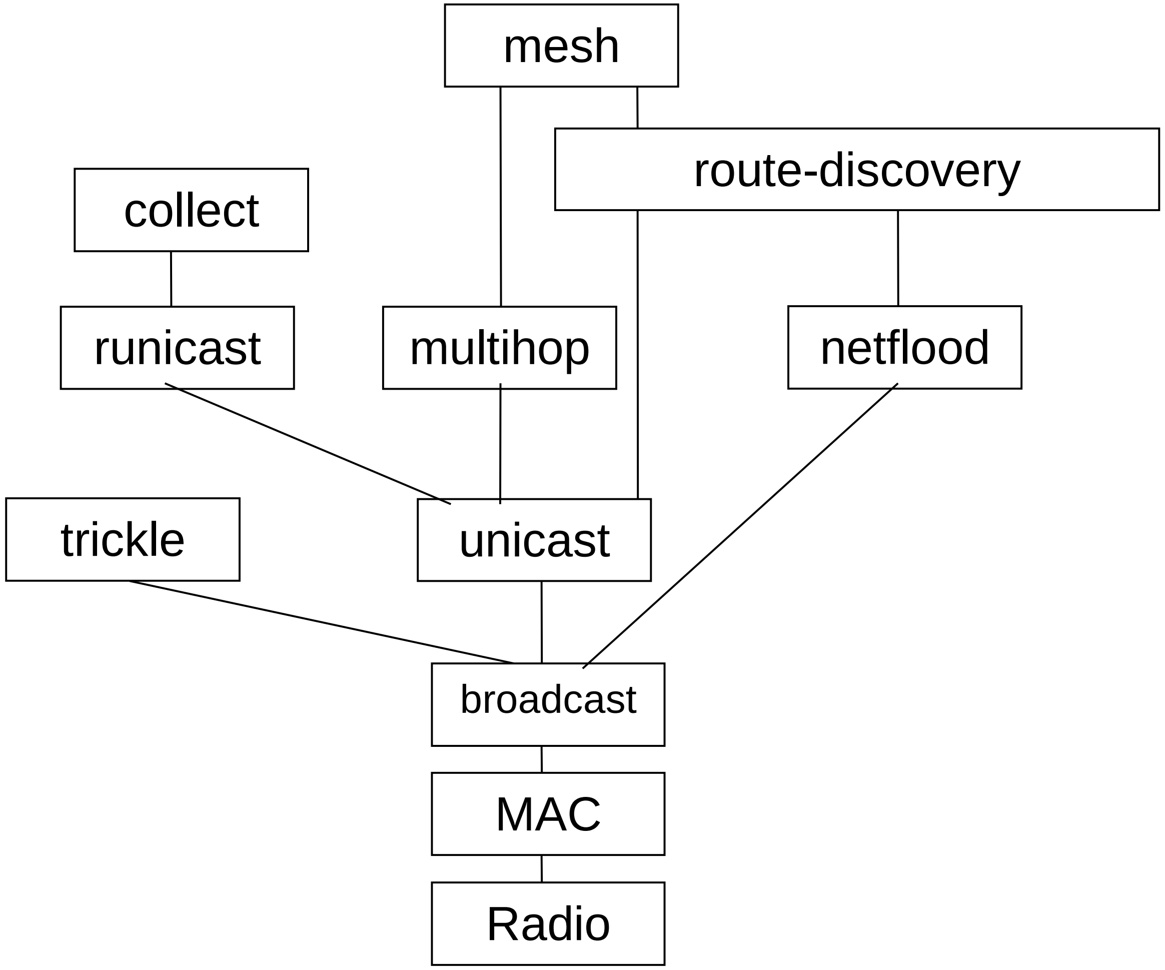
Code in core/net/rime

Have a look at examples/rime:

You will find abc (anonymous broadcast), unicast, runicast and other stuff

SWEDISH INSTITUTE OF COMPUTER SCIENCE SICS

Rime map (partial)



Rime Channels

All communication in Rime is identified by a 16-bit channel (like a port in IP)

Communicating nodes must agree on what modules to use on a certain channel

Example

unicast <-> unicast on channel 155

netflood <-> netflood on channel 130

Channel numbers < 128 are reserved by the system

Used by the shell, other system apps

Rime Programming Model

- Callbacks
  - Called when packet arrives, times out, error condition, …
- Connections must be opened before use
  - Arguments: module structure, channel, callbacks

Networking with the Rime Stack

```
static void
abc_recv(struct abc_conn *c)
{
  printf("abc message received '%s'\n", (char *)packetbuf_dataptr());
}
static const struct abc_callbacks abc_call = {abc_recv};
static struct abc_conn abc;
/*---------------------------------------------------------------------------*/
PROCESS_THREAD(example_abc_process, ev, data)
{
  PROCESS_EXITHANDLER(abc_close(&abc);)
  PROCESS_BEGIN();

  abc_open(&abc, 128, &abc_call);    // rime channel nr: think of it as  IP port
  while(1) {
    ...
    packetbuf_copyfrom("Hello", 6);      // there is one buffer (memory constraints)
    abc_send(&abc);
    printf("abc message sent\n");
  }
  PROCESS_END();
}
```

SWEDISH INSTITUTE OF COMPUTER SCIENCE SICS

Exercise: Networking with Rime Broadcast on Tmote Sky (Sect. 6.2)

1. change MAC layer to NULLMAC:

Seems like the most safe option (see Section 6.1) is to modify platform/sky/contiki-conf.h

MAC_DRIVER for NULLMAC is `nullmac_driver,` for lpp is `lpp_driver`

2. go to examples/rime/examples-abc.c:

- Modify the code to send your name
- Compile and download on tmote sky

See if your message shows up on projector

SWEDISH INSTITUTE OF COMPUTER SCIENCE SICS

Networking with Rime Unicast in COOJA (Sect. 6.3)

Start COOJA (cd tools/cooja and type "ant run")

and create two sky nodes that run

example-unicast (we use unicast because X-MAC broadcast is expensive)

Nodes need to be in communication range, check with the right "visualiser skin" (UDGM) if nodes can communicate

(check the other "skins", useful)

Simulation can be started and stopped with Ctrl-S or from the control panel. Reloaded with Ctrl-R

Hint: COOJA assignes node Ids starting from 1. Therefore, example-unicast.c needs to be slightly modified

Change MAC layer

By default, Contiki on Sky uses X-MAC.

Now change to NULLMAC and then LPP. Can you identify using the time line how these MAC work? Note that Contiki during boot-up prints out the MAC layer (very useful)

You can add compile options in Makefile (does not work for COOJA):

CFLAGS = -DWITH_NULLMAC=1

Or: make DEFINES=MAC_CONF_DRIVER=lpp_driver when compiling (not for COOJA)

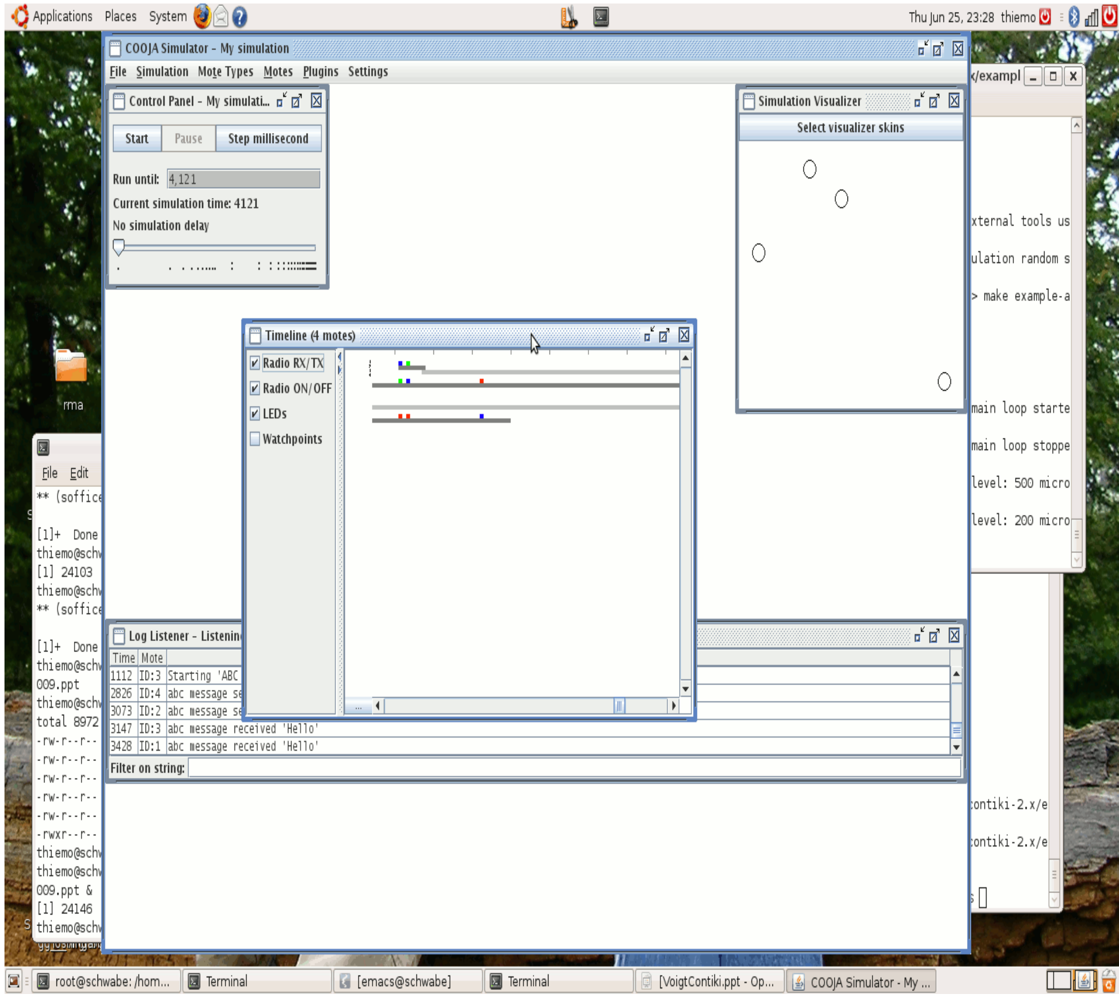
Or set MAC layer driver in contiki-conf.h in platforms/sky, MAC_DRIVER for lpp is `lpp_driver`, for NULLMAC `nullmac_driver`

New cool feature: Timeline in COOJA

<u>ONOFF</u>
No color: radio off
Grey: radio on.

<u>RXTX</u>
Green: received packet
Blue: packet sent
Red: interfered radio (collisions etc).

By moving mouse above you can get explanations.
Right-click to zoom.



```
Run until: 4,121
Current simulation time: 4121
No simulation delay
```

Log Listener - Listening

```
Time  Mote
1112  ID:3  Starting 'ABC
2826  ID:4  abc message se
3073  ID:2  abc message se
3147  ID:3  abc message received 'Hello'
3428  ID:1  abc message received 'Hello'
Filter on string:
```

Power consumption

You could use energest for obtaining per-component power consumption in Contiki, only a few lines need to be added (next slide)

Or use MSPSim (built-in in COOJA)

Right-click on one node, open mote interface and choose MSP Cli, type help, type duty 1 CC2420

What can you identify?

Comment: easier with unicast and by comparing the results seen with NULLMAC and with a low power MAC layer.

Measure Power Consumption with Energest

For this, only a few lines need to be added:

```
PROCESS_BEGIN();
  static struct etimer et;
  static unsigned long rx_start_time;
 ...
...
  rx_start_time = energest_type_time(ENERGEST_TYPE_LISTEN);
  lpm_start_time = energest_type_time(ENERGEST_TYPE_LPM);
  cpu_start_time = energest_type_time(ENERGEST_TYPE_CPU);
  tx_start_time = energest_type_time(ENERGEST_TYPE_TRANSMIT);
..
    printf("energy listen %lu tx %lu cpu %lu lpm %lu\n",
   energest_type_time(ENERGEST_TYPE_LISTEN) - rx_start_time,          // in while loop
   energest_type_time(ENERGEST_TYPE_TRANSMIT) - tx_start_time,
   energest_type_time(ENERGEST_TYPE_CPU) - cpu_start_time,
   energest_type_time(ENERGEST_TYPE_LPM) - lpm_start_time);


  PROCESS_END();
}
```

Measure Power Consumption with Energest

Now we have the times a component was on.
This can be converted to power consumption in the following way (here for RX):

Power (mW) = (rxend- rxstart)*20mA*3V/4096/runtime (seconds)

The 20mA are pre-measured (or from data sheet). 3V is the Tmote operational voltage (approximated). 4096 is ticks per second. If you do not divide by runtime you get the energy consumption during runtime.

What we have learnt

- Rime networking
- COOJA and the timeline
- How to choose MAC layer in Contiki
  - There are NULLMAC, X-MAC and LPP
- How to measure power consumption

Shell

Go to examples/sky-shell
make TARGET=sky sky-shell.mspsim
or
make TARGET=sky sky-shell.upload
then
make TARGET=sky login

Type help to get an overview of the available commands (also see handout):
reboot, ps, blink 10, sense
Use pipe: sense | senseconv

Use write, plug out mote from USB, and put it in again (try read)

Shell

Shell is quite memory-intensive. You can add other shell programs in the sky-shell-program by calling their init functions.

Add e.g. `shell_power_init()`

In my Contiki version, the memory is not sufficient

Shell

Also cool, test (with X-MAC or LPP in MSPSim):

```
mac 0
mac 1
```

Watch the Duty-Cycle Monitor..

Advanced Debugging with COOJA's TimeLine using Watchpoints

“Normal” Breakpoints:

Right-click on a node in the visualizer plugin. Choose “Open mote plugin->MSP code watcher”. Select a source file, scroll to source code line and right-click “do add breakpoint”.

Run the simulation: it will stop at the breakpoint.

Watchpoint: a breakpoint that is displayed in the TimeLine but the simulation does not stop.

To convert the breakpoint into a watchpoint find the hidden pane in the left of the window. Open this pane by clicking on the left hand side of the window and drag the pane open. This pane lists all breakpoints. To turn the breakpoint into a watchpoint, deselect the "Stop" checkbox.

Next, chose a name and a color for the watchpoint. Do this by clicking in the empty "Info" field of the watchpoint. This opens a window in which one can enter a name and chose a color.

Finally, enable watchpoints in the TimeLine by selecting the "Watchpoints" checkbox. Start the simulation. The watchpoint for the selected node should now be visible as a tick in the TimeLine.

Watchpoint exercise: Check Output Transmission power

Go to cc2420 driver in core/dev/cc2420.c

Find the right lines in `cc2420_send` (search for set_tx_power) and set one or two watchpoints

Update examples/rime/example_abc.c

```
packetbuf_set_attr(PACKETBUF_ATTR_RADIO_TXPOWER,txpower);
```

With 1<= txpower <= 32

Now do two things:

1. Check if you really modify the output power using a watchpoint
2. If the above works, check if you can reach the sink node, so that your name is displayed on the projector.

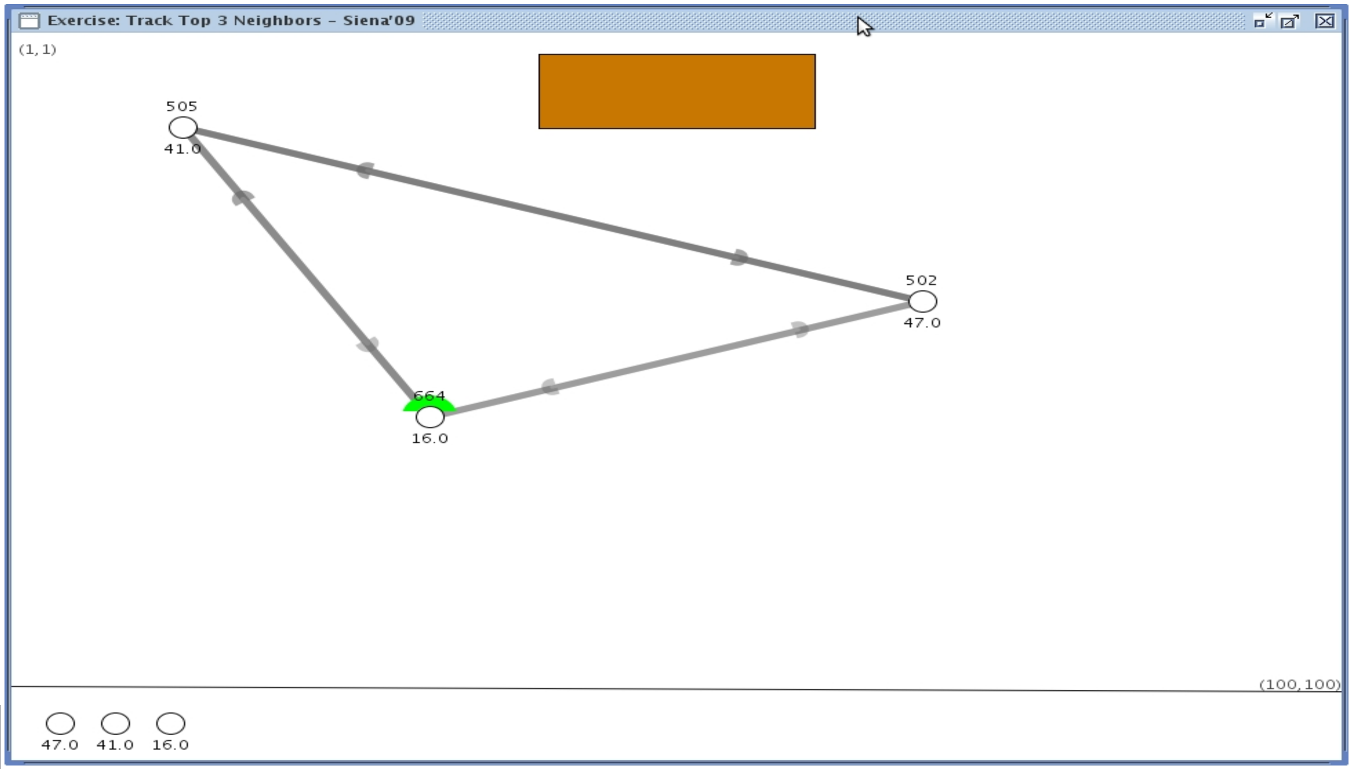
Optional: extend the program to increase transmission power with one when the button is pressed.

Exercise for Last Block

Neighbour discovery: see handout.

Step 1: pos in ex-light.c and report-top3.c

Try COOJA quick-start, see Section 7.7



More stuff

Look at Section 7.7 for COOJA quickstart to help you explore some of the programs:

```
cd examples/contiki-crash
make ex-light.csc TARGET=cooja
```

Same for other csc files

When you are done, you can improve your program:

- Moving average for RSSI values
- Measure power consumption
- ...

SWEDISH INSTITUTE OF COMPUTER SCIENCE SICS

Thank you!

More info on www.sics.se/contiki
with lots of material (papers, tutorials, mailing list) etc.

Thanks to Adam Dunkels for many of the slides
Thanks to Fredrik Österlind for sharing all his material
Thanks to Carlo Boano and Shahid Raza for assistance during the course
Thanks to Zhitao and Anna for sending motes
Thanks to Kay for inviting me

CONET COOPERATING OBJECTS NETWORK OF EXCELLENCE

SWEDISH INSTITUTE OF COMPUTER SCIENCE SICS