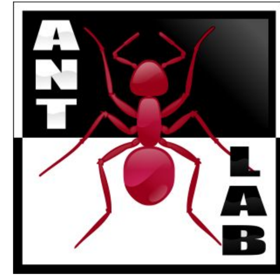
Politecnico di Milano

Advanced Network Technologies Laboratory

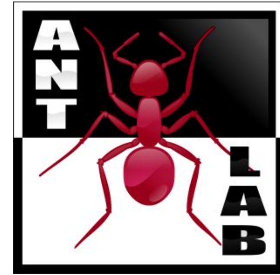


Internet of Things

Contiki and Cooja

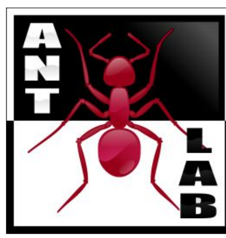
Politecnico di Milano

Advanced Network Technologies Laboratory



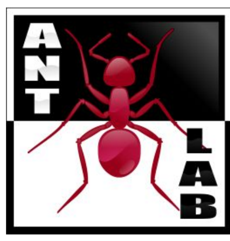
The Contiki Operating System

Contiki



- Contiki features:
  - Open-source
  - Highly portable
  - Multi-tasking
  - Memory-efficient
- Typical configuration
  - 2 KB RAM, 40 KB ROM
- Main key mechanisms widely adopted in the industry
  - TCP/IP like communication (uIP / uIPv6)
  - Low-power communication stack (Rime)

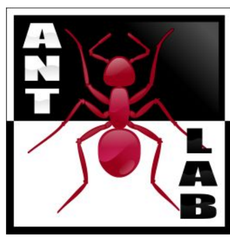
Other Features



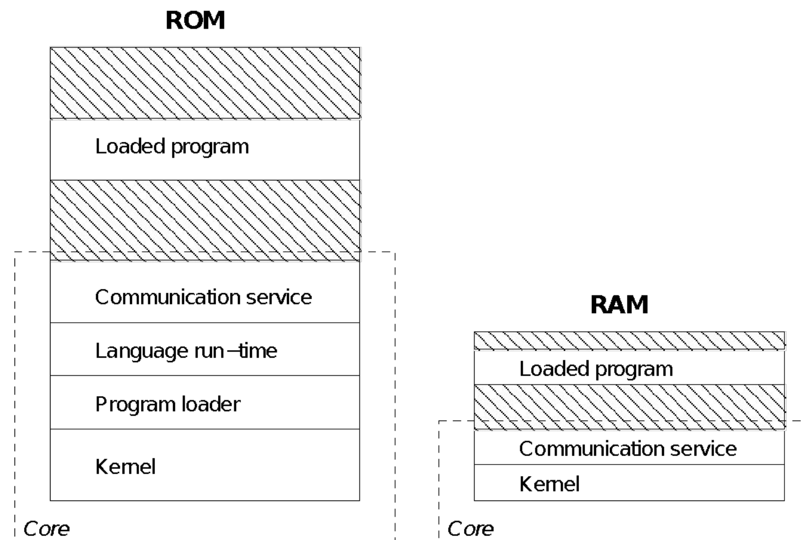
- Other important features of Contiki:
  - Dynamic loading / re-programming
  - Multi-threading implemented as an application library (linked optionally if the application explicitly requires it)

- Cons:
  - Documentation is rather scarce
  - Reading source code is often required to fully understand functionalities

System overview



- A running Contiki System consists of
  - Kernel (event-driven, no HAL)
  - Libraries
  - Program loader (loading, memory allocation, initialization)
  - Processes (services and applications)



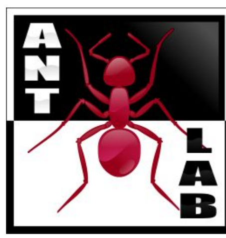
Contiki services

- A service is a process that can be used by other processes (shared library)
  - Communication protocol stack
  - Sensor device drivers
  - Data handling algorithms
- Services can be replaced!

Contiki libraries

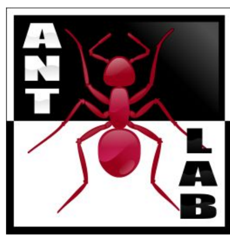
- The kernel provides only the most basic features.
- The rest of the system is implemented as system libraries (static or dynamic)
- Often used libraries are placed in the Contiki core (e.g., memcpy)
- Rarely-used libraries must be included manually (e.g., atoi)

Multi-threading and protothreads



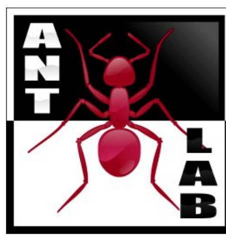
- Contiki is an event-driven kernel, but provides support for multi-threading and a thread-like construct called protothread
- Differently from threads, protothreads do not require a separate stack (memory saving!).
- Contiki processes: a single protothread
  - Declaration: #define PROCESS (name, strname)
  - Start of a process: #define PROCESS_BEGIN()
  - End of a process: #define PROCESS_END()
  - Body of a process: #define PROCESS_THREAD(name, ev, data)
- A process always starts with PROCESS_BEGIN() and ends with PROCESS_END()

Useful macros



- AUTOSTART_PROCESS(&process)
- PROCESS_WAIT_EVENT()
  - Blocks the current process until an event is received
- PROCESS_WAIT_EVENT_UNTIL(c)
  - Similar to the previous one, but an extra condition must be true to continue

Code example



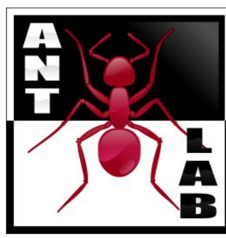
```
PROCESS(hello_world_process,”Hello World”);
AUTOSTART_PROCESS(&hello_world_process);

PROCESS_THREAD(hello_world_process,ev,data){
    PROCESS_BEGIN();
        printf(“Hello world!\n”);
        while(1){
            PROCESS_WAIT_EVENT();
        }
    PROCESS_END();
}
```

Example: Blink application

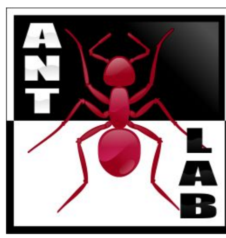
- Two processes:
  - hello_world_process: starts a timer and prints “Hello world” each time
  - blink_process: starts a timer, increments a counter and toggles leds accordingly
- Let’s simulate it with Cooja!

Cooja



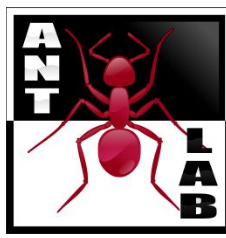
- Cooja is a network simulator for Contiki
- Nice features:
  - Allows to simulate an heterogeneous network
  - Different type of sensor nodes
  - Different applications at the same time
  - Nice graphical interface
  - Save / load simulation

Why Protothreads?



- Easier than managing a state-machine
- Code is generally shorter and uses structured programming
- Logic mechanisms are evident from the code

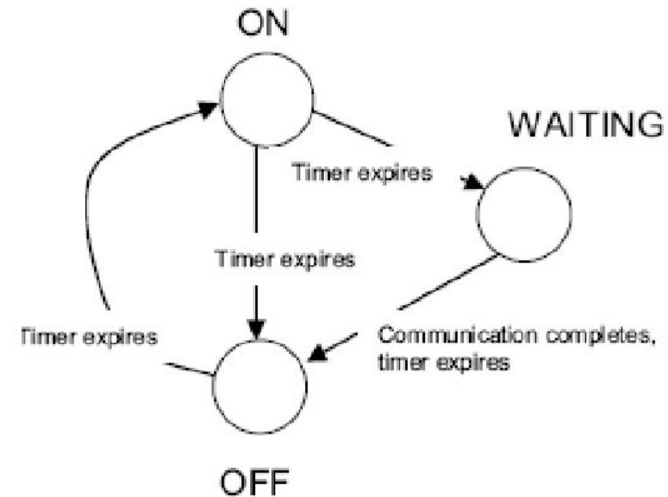
Programming example



- TinyOS

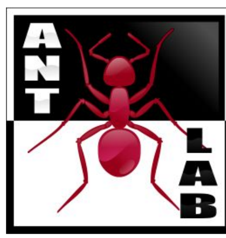
```
enum {ON, WAITING, OFF} state;

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;
    }
  }
}
```



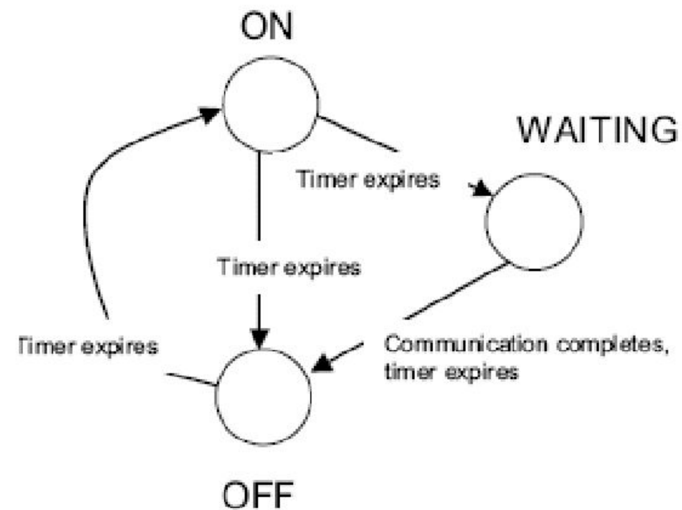
Source: T. Voigt, Contiki – a Dynamic Operating System for Networked Embedded Systems

Programming example



- Contiki

```
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);
}
```

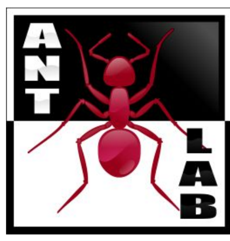


Source: T. Voigt, Contiki – a Dynamic Operating System for Networked Embedded Systems

Contiki communication: Rime

- Rime: custom lightweight networking stack
- Used when IP overhead is too prohibitive
- Provided primitives:
  - Single-hop broadcast / unicast
  - Multi-hop unicast
  - Network-flooding
  - Data collection

Rime example: broadcast



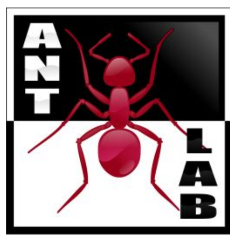
- Broadcast connection:
  - static struct broadcast_conn broadcast;
- Open the connection:
  - **broadcast_open(&broadcast, 129, &broadcast_call);**
  - First parameter is the connection name
  - Second parameter is the channel on which the connection will operate (different from wireless channel)
  - Third parameter is a function pointer to a function that will be called when a packet has been received
- Set up the callback
  - static const struct broadcast_callbacks broadcast_call = {broadcast_recv}
  - static void broadcast_recv(struct broadcast_conn *c, const rimeaddr_t *from){do something when a msg is received}

Rime example: broadcast

- Main process

```
PROCESS_THREAD(example_broadcast_process,ev,data)
static struct etimer et;
PROCESS_BEGIN();
//open the broadcast connection
broadcast_open(…);
while(1){
   etimer_set(&et,CLOCK_SECOND*4);
   PROCESS_WAIT_EVENT_UNTIL(etimer_expired(&et));
   packetbuf_copyfrom("HELLO",5) //copy into packet
   broadcast_send(&broadcast);
 }
PROCESS_END();
```

Other examples



- Rime provides also unicast, reliable unicast and multi hop (mesh) communication
- For more complex applications, other communication stacks may be directly used!
  - uIP (TCP/IP)

Contiki and NodeRED

- Cooja can be attached to NodeRED using the serial monitor
- Data coming from a WSN can be thus used as starting point for high-level, web-based applications
- Let’s see an example…