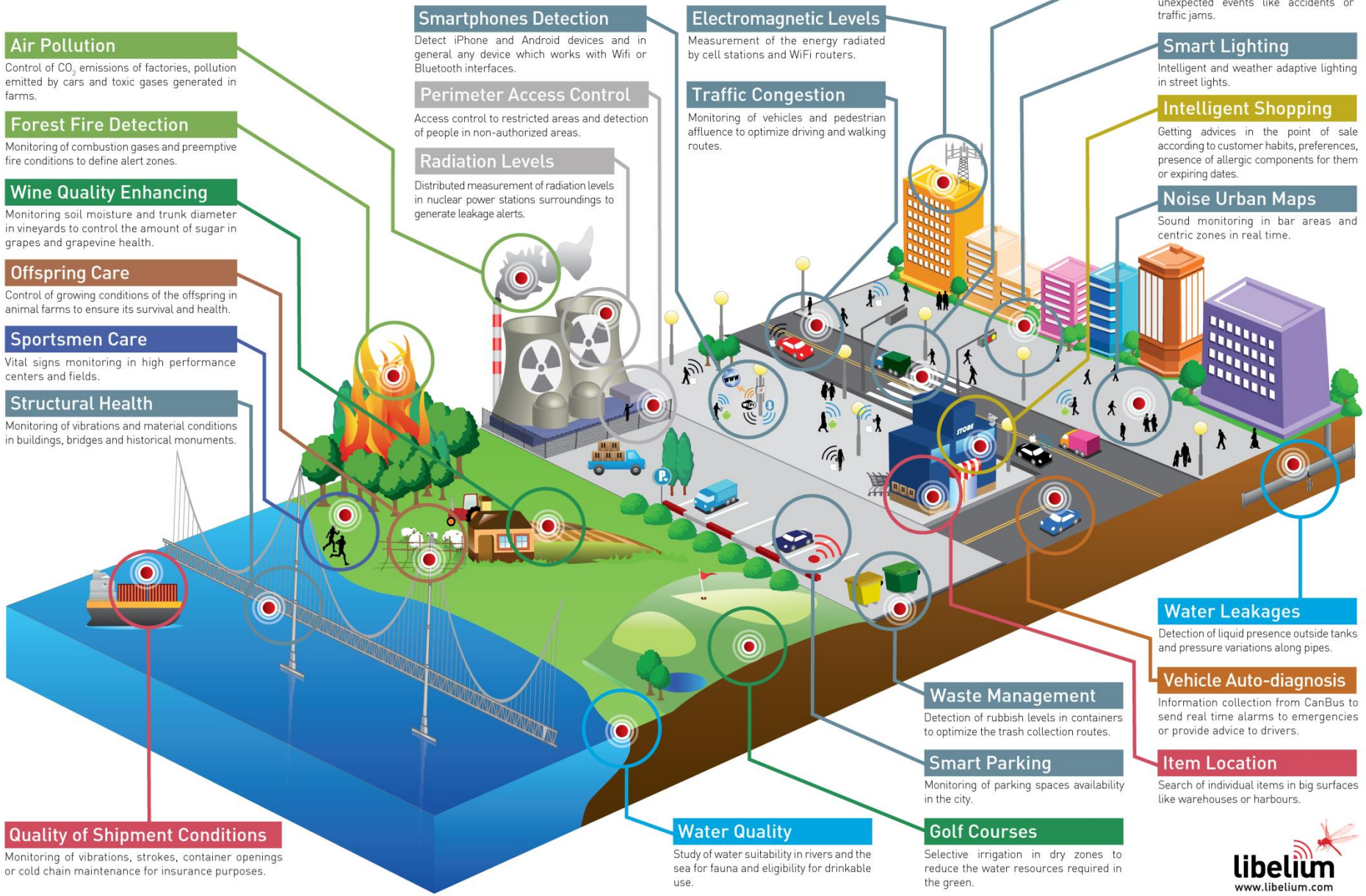
ContikiOS és Cooja

VITMMA09 – Okos város MSc mellékspecializáció

Libelium Smart World



Air Pollution
Control of CO_2 emissions of factories, pollution emitted by cars and toxic gases generated in farms.

Forest Fire Detection
Monitoring of combustion gases and preemptive fire conditions to define alert zones.

Wine Quality Enhancing
Monitoring soil moisture and trunk diameter in vineyards to control the amount of sugar in grapes and grapevine health.

Offspring Care
Control of growing conditions of the offspring in animal farms to ensure its survival and health.

Sportsmen Care
Vital signs monitoring in high performance centers and fields.

Structural Health
Monitoring of vibrations and material conditions in buildings, bridges and historical monuments.

Quality of Shipment Conditions
Monitoring of vibrations, strokes, container openings or cold chain maintenance for insurance purposes.

Smartphones Detection
Detect iPhone and Android devices and in general any device which works with Wifi or Bluetooth interfaces.

Perimeter Access Control
Access control to restricted areas and detection of people in non-authorized areas.

Radiation Levels
Distributed measurement of radiation levels in nuclear power stations surroundings to generate leakage alerts.

Electromagnetic Levels
Measurement of the energy radiated by cell stations and WiFi routers.

Traffic Congestion
Monitoring of vehicles and pedestrian affluence to optimize driving and walking routes.

Water Quality
Study of water suitability in rivers and the sea for fauna and eligibility for drinkable use.

Waste Management
Detection of rubbish levels in containers to optimize the trash collection routes.

Smart Parking
Monitoring of parking spaces availability in the city.

Golf Courses
Selective irrigation in dry zones to reduce the water resources required in the green.

Smart Roads
Warning messages and diversions according to climate conditions and unexpected events like accidents or traffic jams.

Smart Lighting
Intelligent and weather adaptive lighting in street lights.

Intelligent Shopping
Getting advices in the point of sale according to customer habits, preferences, presence of allergic components for them or expiring dates.

Noise Urban Maps
Sound monitoring in bar areas and centric zones in real time.

Water Leakages
Detection of liquid presence outside tanks and pressure variations along pipes.

Vehicle Auto-diagnosis
Information collection from CanBus to send real time alarms to emergencies or provide advice to drivers.

Item Location
Search of individual items in big surfaces like warehouses or harbours.

libelium
www.libelium.com

OKOS VÁROS @ BME TMIT

Contiki

- Nyílt forráskodú operációs rendszer az “Internet of Things” eszközöknek

- A Contiki a kis költségű, kis teljesítményű mikrokontrollerek egymás közötti kommunikációját és Internetre való csatlakozását valósítja meg

Contiki

- Contiki egy nyílt forráskodú operációs rendszer, amely kisméretű, kis teljesítményű mikorkontrollereken fut és lehetővé teszi olyan alkalamzások fejlesztését ezen eszközökre, amelyek a hardvert hatékonyan használják, miközben szabványosított alacsony energiafogyasztású vezeték nélküli kommunikációt biztosít

- A Contiki-t már számos kereskedelmi és nem kereskedelmi rendszerben használják mint például városi zajszennyezetség mérés, városi világítás, smart metering, ipari gépek felügyelete, sugárzás mérés, épitkezési terület felügyelete, riasztó rendszerek, smart home, stb....

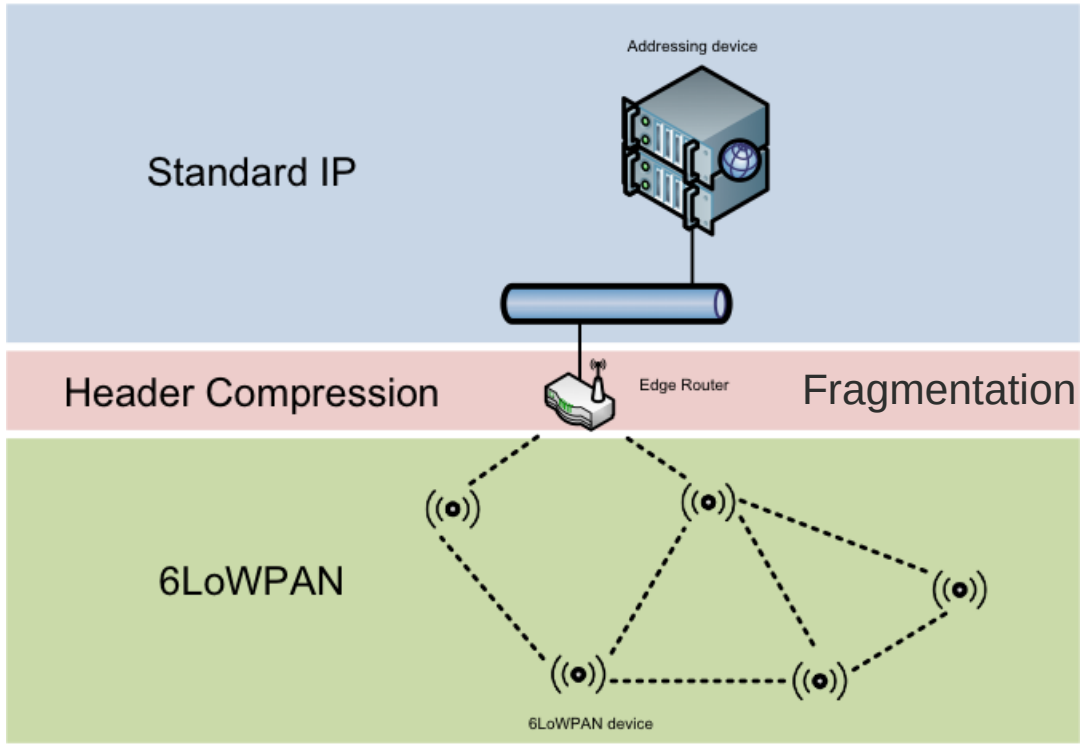
Contiki

Egységes környezetet biztosít a felhasználói alkalmazásonak:

- Kezeli a programszálak konkurens futtatását
- Időzítő funciók
- Memóriamenedzsment
- Kommunikációs funkciók
  - Legújabb szabványok támogatása: 6lowpan, RPL, CoAP, stb..
- Hadver-illesztőprogramok

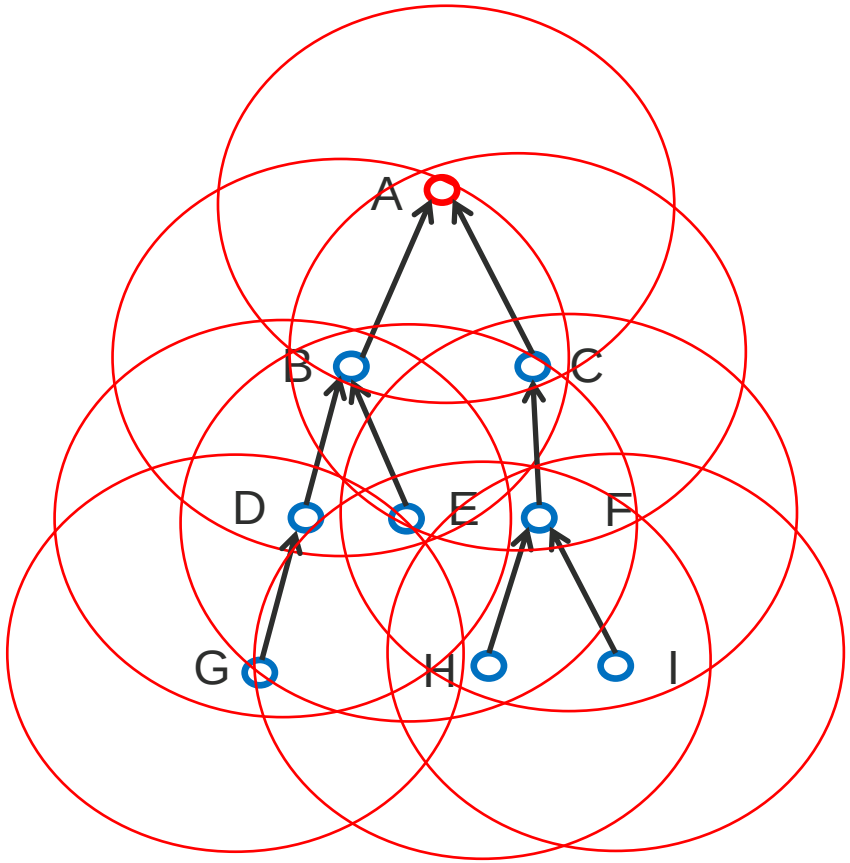
6lowpan

- Csomagméretek átalakítása
  - Fejléc tömörítés
  - Csomag darabolás



RPL - Routing Over Low power and Lossy networks

- Proaktív, „any-to-any” routing protokoll, bár „many to one” használatra van optimalizálva
- Egy körmentes irányított gráfot épít fel egy gyökér elemből
- A routing metrikák folyamatos frissítése
- Az IoT eszközökre van szabva
  - Korltázott erőforrások
  - A rádiós csatornák zajosak, veszteségek, kis adatsbebesség érhetőek el



CoAP Constrained Application Protocol

- Alkalmazás rétegbeli protokoll

- Az erőforrásokhoz és szolgáltatásokhoz egy URL van rendelve, ezt a szolgáltatók egy szerveren teszik közzé
  - GET
  - PUT
  - POST
  - DELETE

  - IoT eszközökre optimalizált

Contiki létrejötte

- Adam Dunkels Ph.D.
  - svéd kutató és vállakozó
  - uIP (TCP/IP stack, kis számú kódsor, kis RAM igény)
    - uIP6 (2008)
  - Contiki OS (2002)
    - A nevet Thor Heyerdahl 1947-es expedíciója ihlette -> Kon-Tiki
  - Protothread
    - proramozási modell -> a többszálón futó és az esemény-vezérelt programozási modelleket ötvözi

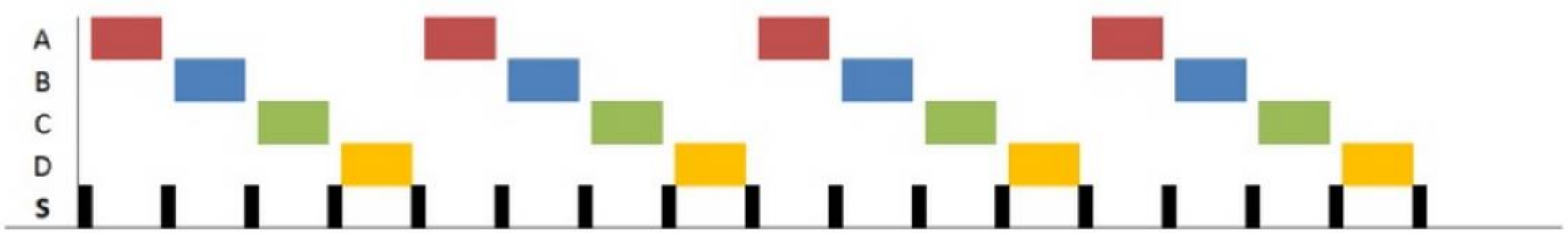
Jelenleg támogatott hardver eszközök

MCU/SoC	Radio	Platforms	Cooja simulation support
RL78	ADF7023	EVAL-ADF7023DB1	-
TI CC2538	Integrated	cc2538dk	-
TI MSP430x	TI CC2420	exp5438, z1	Yes
TI MSP430x	TI CC2520	wismote	Yes
Atmel AVR	Atmel RF230	avr-raven, avr-rcb, avr-zigbit, iris	-
Atmel AVR	TI CC2420	micaz	Yes
Freescale MC1322x	Integrated	redbee-dev, redbee-econotag	-
ST STM32w	Integrated	mb851, mbxxx	-
TI MSP430	TI CC2420	sky, jcreate, sentilla-usb	Yes
TI MSP430	TI CC1020	msb430	-
TI MSP430	RFM TR1001	esb	Yes

MCU/SoC	Radio	Platforms	Cooja simulation support
Atmel Atmega128 RFA1	Integrated	avr-atmega128rfa	-
Microchip pic32mx795f512l	Microchip mrf24j40	seed-eye	-
TI CC2530	Integrated	cc2530dk	-
RC2300/RC2301	Integrated	sensinode	-
6502	-	apple2enh, atari, c128, c64	-
Native	-	native, minimal-net, cooja	Yes
Microchip pic32mx795f512l	Microchip mrf24j40	seed-eye	-
TI CC2530	Integrated	cc2530dk	-
RC2300/RC2301	Integrated	sensinode	-
6502	-	apple2enh, atari, c128, c64	-
Native	-	native, minimal-net, cooja	Yes

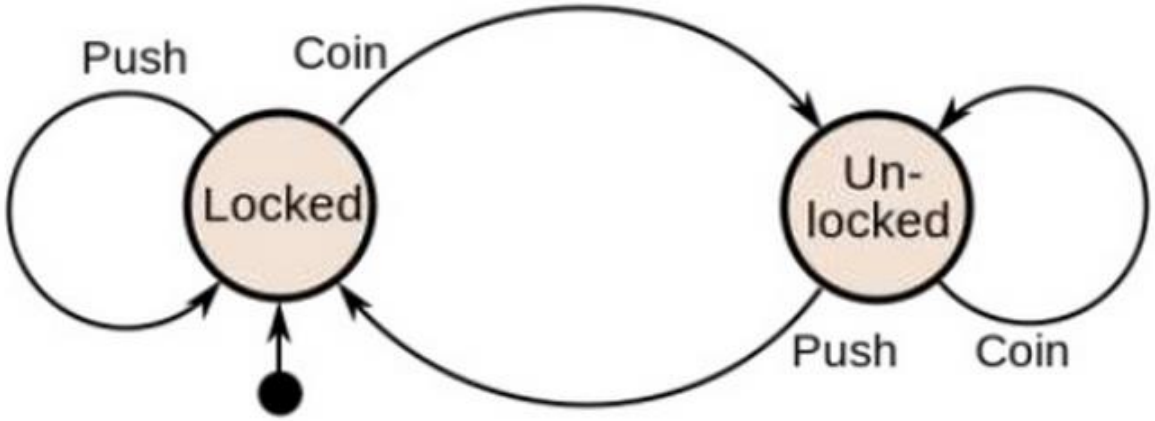
Többszálon futó programozási alapelv

- Minden program szál (thread) külön idő rést kap, amiben futhat
- Ezeket az OS ütemezője szüneteltetheti / folytathatja, ha további futtatás szükséges
- Költséges – RAM, CPU



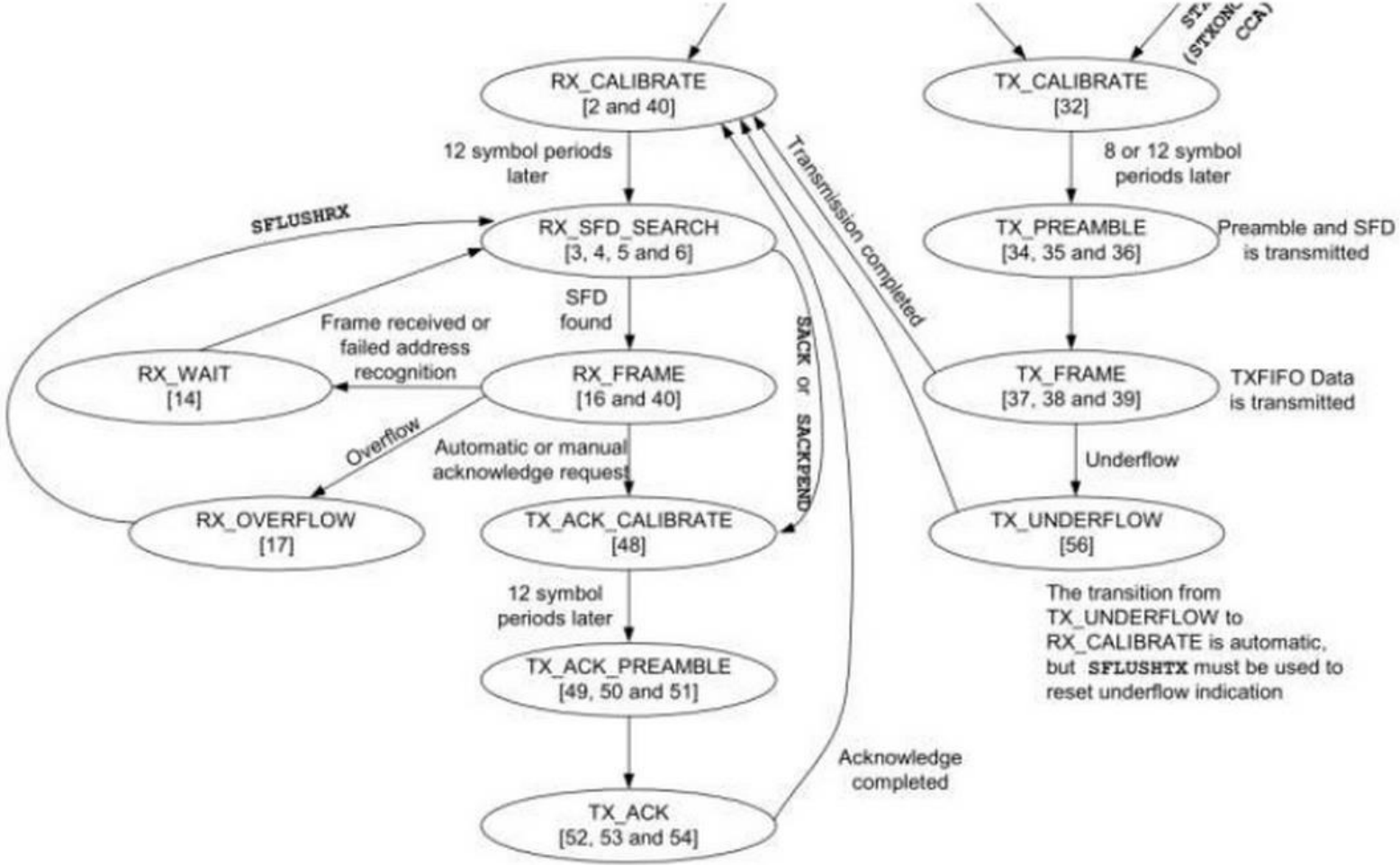
Esemény-vezérelt / állapotgépes programozási alapelv

- Reaktív – az események szabják meg a program futását
  - Timers, hardware interrupts, radio, etc...
- Kevesebb erőforrást igényel
- Nehezebb átlátni és programozni (függvényhívások, állapotváltozók)



Esemény-vezérelt / állapotgépes programozási alapelv

- Állapotok
- Események
- Feltételek
- Állapot átmenetek



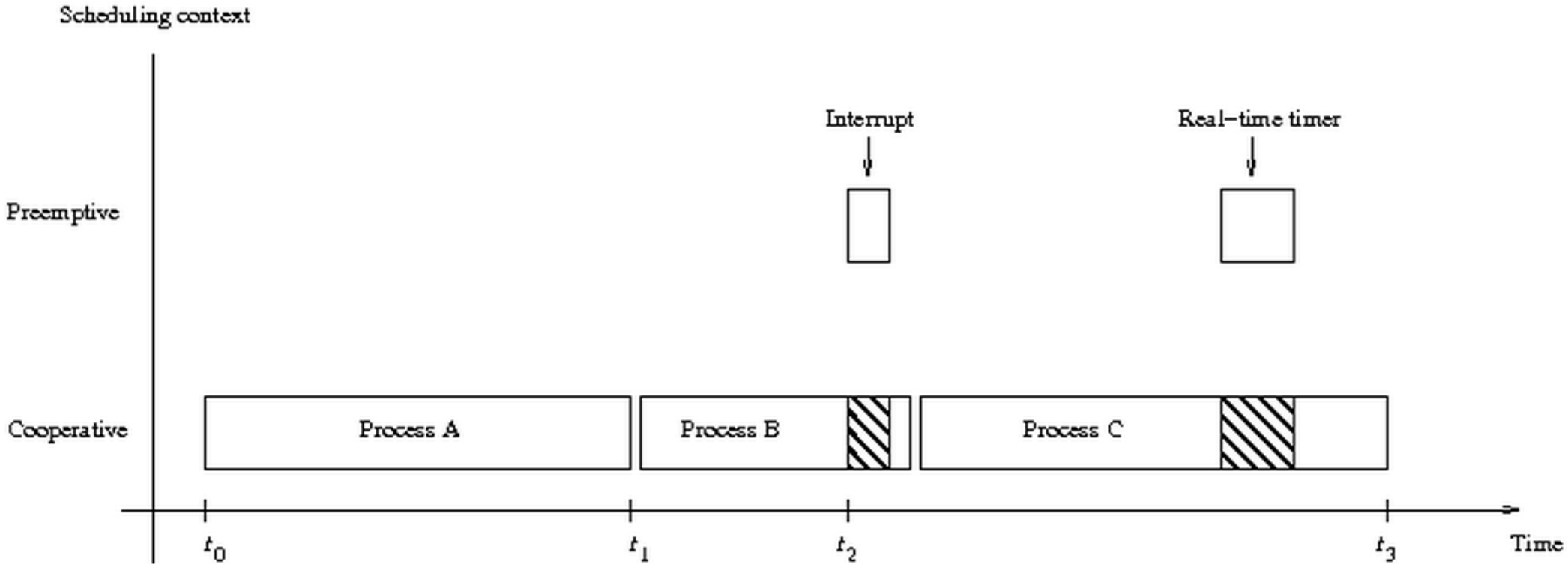
Prothothreads

- Esemény-vezérelt pr. hatékonyabban bánik az erőforrásokkal
- Többszálon futó pr. egyszerűbb programozni
- Lehetséges e az előnyös tulajdonságaikat ötvözni?
- => Prothothread
  - Többszálasítás a memória többlet igénye nélkül
  - Eseményvezérelt, de többszálas programozási technikával
  - “Co-operative” és “Preemptive” ütemezés egyszerre

Contiki kódok végrehajtása

- A Contiki kódok két féle végrahajtási módban futhatnak
  - Cooperative és preemptive
- Cooperative kódrészek szekvenciálisan futnak
  - A “processek” mindig ilyen kontextusban futnak
- Preemptive kódok megszakíthatják a cooperative kódok futását
  - interruptok, eszköz driverek, timers

Contiki kódok végrehajtása



Protothreads

- Programkód szervezési módszer
  - Lehetővé válik más programkód futtatása, ha egy adott programkód várakozó állapotban van
- Ezen programkód szervezés által megoldhatóvá válik, hogy C kódok úgy fussanak, mintha többszálasítva volnának
  - a memóriatöbblet igénye nélkül
- A protothread-k hagyományos C függvények, melyek speciális makró-k közé vannak ágyazva
  - PROCESS_BEGIN();
  - PROCESS_END();

Protothread makrók a Contikiben

```
PROCESS_BEGIN(); // Declares the beginning of a process' protothread.
PROCESS_END(); // Declares the end of a process' protothread.
PROCESS_EXIT(); // Exit the process.
PROCESS_WAIT_EVENT(); // Wait for any event.
PROCESS_WAIT_EVENT_UNTIL(); // Wait for an event, but with a condition.
PROCESS_YIELD(); // Wait for any event, equivalent to PROCESS_WAIT_EVENT().
PROCESS_WAIT_UNTIL(); // Wait for a given condition; may not yield the process.
PROCESS_PAUSE(); // Temporarily yield the process.
```

Contiki processes

```
#include "contiki.h"
/*---------------------------------------------------------*/
PROCESS(hello_world_process, "Hello world process");
AUTOSTART_PROCESSES(&hello_world_process);
/*---------------------------------------------------------*/
PROCESS_THREAD(hello_world_process, ev, data)
{
  PROCESS_BEGIN();

  printf("Hello, world\n");

  PROCESS_END();
}
```

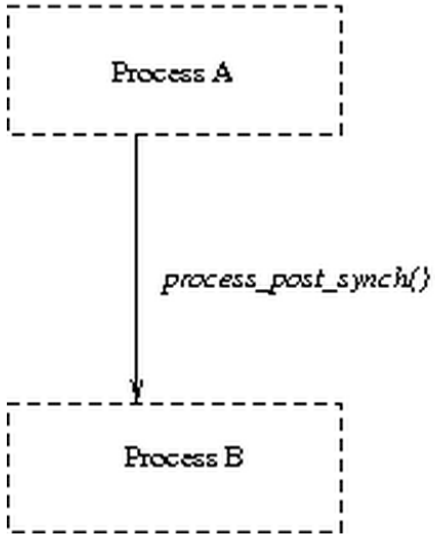
Hogyan várjunk egy eseményre?

```
PROCESS_THREAD(first_process, ev, data)
.. ..
PROCESS_WAIT_EVENT_UNTIL(ev == servo_event);
if(data != NULL) {
  handle_servo((struct servo_data *) data);
}
```

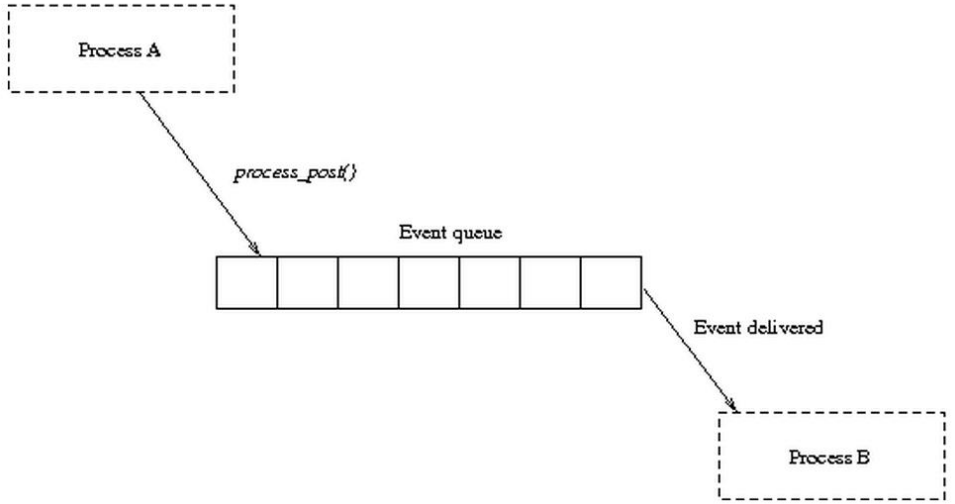
- ev contains the event number
- data is a void* to any data
  - Can be NULL

Szinkron és aszinkron események

Szinkron események



Aszinkron események



A “process” ütemező

- A különböző processek futtatását intézi, amennyiben azok futtatására szükség van

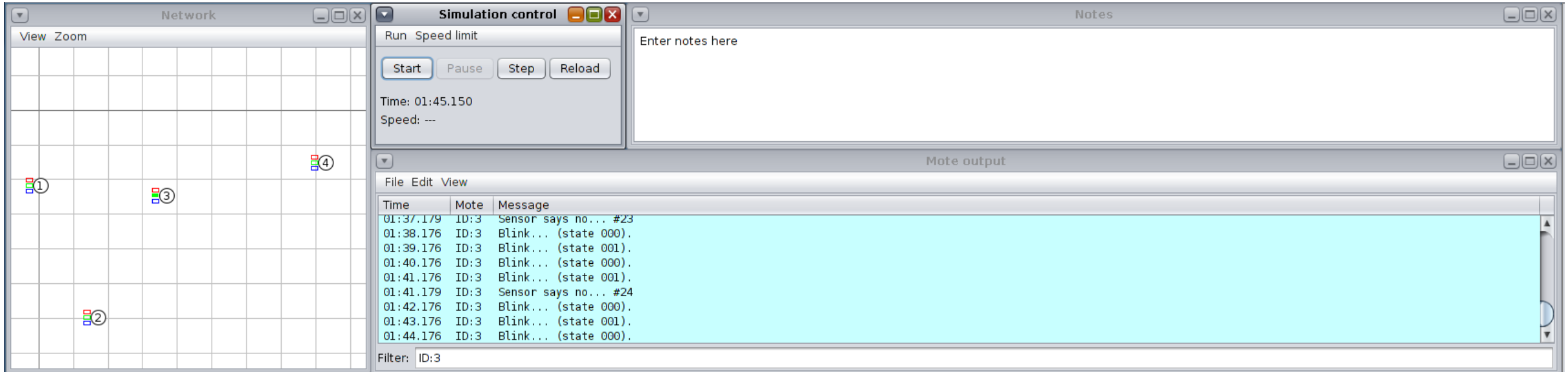
- Process hívás akkor történik,
  - amikor az adott process-t érintő esemény történik
  - egy másik process által kerül meghívásra

- Minden ilyen híváskor a process-nek átadódik
  - az esemény ID
  - adat pointer

Timers

Event Timer	Callback timer	Simple timer
-generates an event when the timer expires a	-call a function when the timer expires	-have to be actively queried to check when they have expired
•etimer_expired() •etimer_reset •etimer_set() •etimer_restart()	•ctimer_expired() •ctimer_reset •ctimer_set() •ctimer_restart()	•timer_expired() •timer_reset •timer_set() •timer_restart()

Etimers (Blink-hello application)

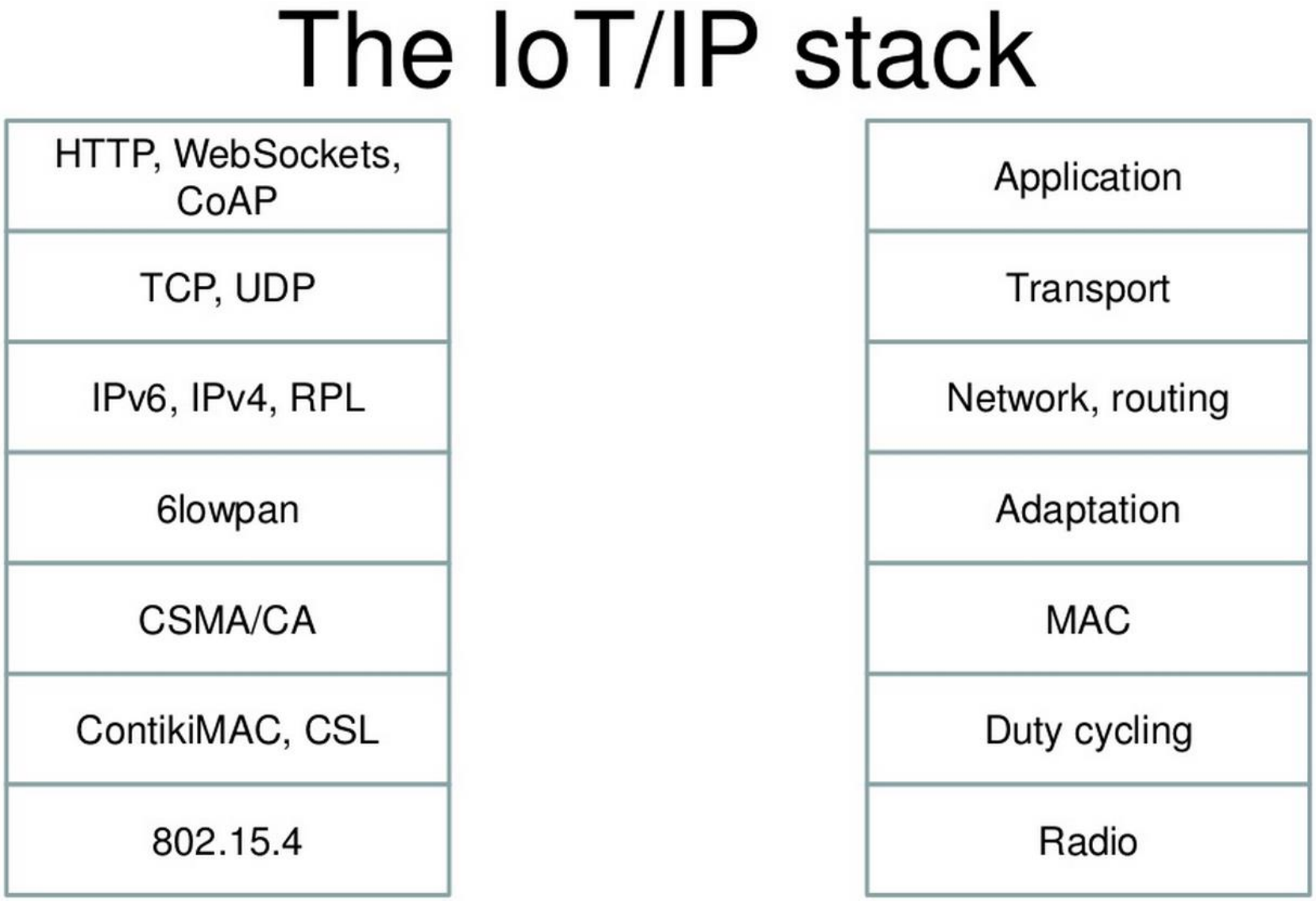


Ctimer

- Print ”Hello, callback” after 1/8 second.

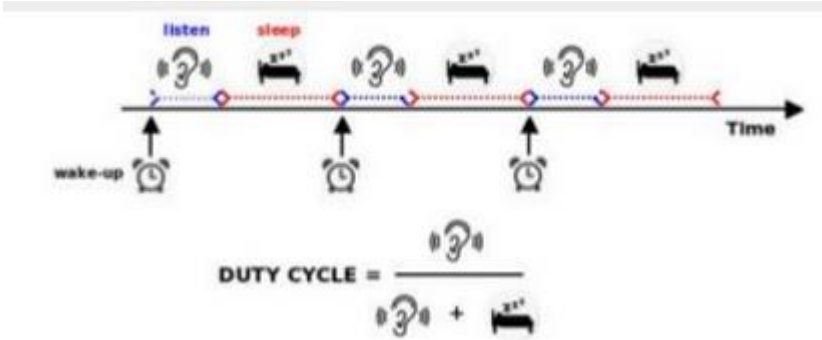
```
static struct ctimer callback_timer;
static void
callback(void *data)
{
  printf("%s\n", (char *) data);
}

void
application(void)
{
  ctimer_set(&callback_timer, CLOCK_SECOND / 8, callback,
      "Hello, callback!");
  return 0;
}
```

ContikiMAC


The IoT/IP stack
HTTP, WebSockets, CoAP
TCP, UDP
IPv6, IPv4, RPL
6lowpan
CSMA/CA
ContikiMAC, CSL
802.15.4
Application
Transport
Network, routing
Adaptation
MAC
Duty cycling
Radio

ContikiMAC

- Radio-Duty Cycling (RDC) protokollok az energiafogyasztást igyekeznek csökkenteni, azzal hogy engedélyezik a node-k számára a rádiós interfészük kikapcsolását



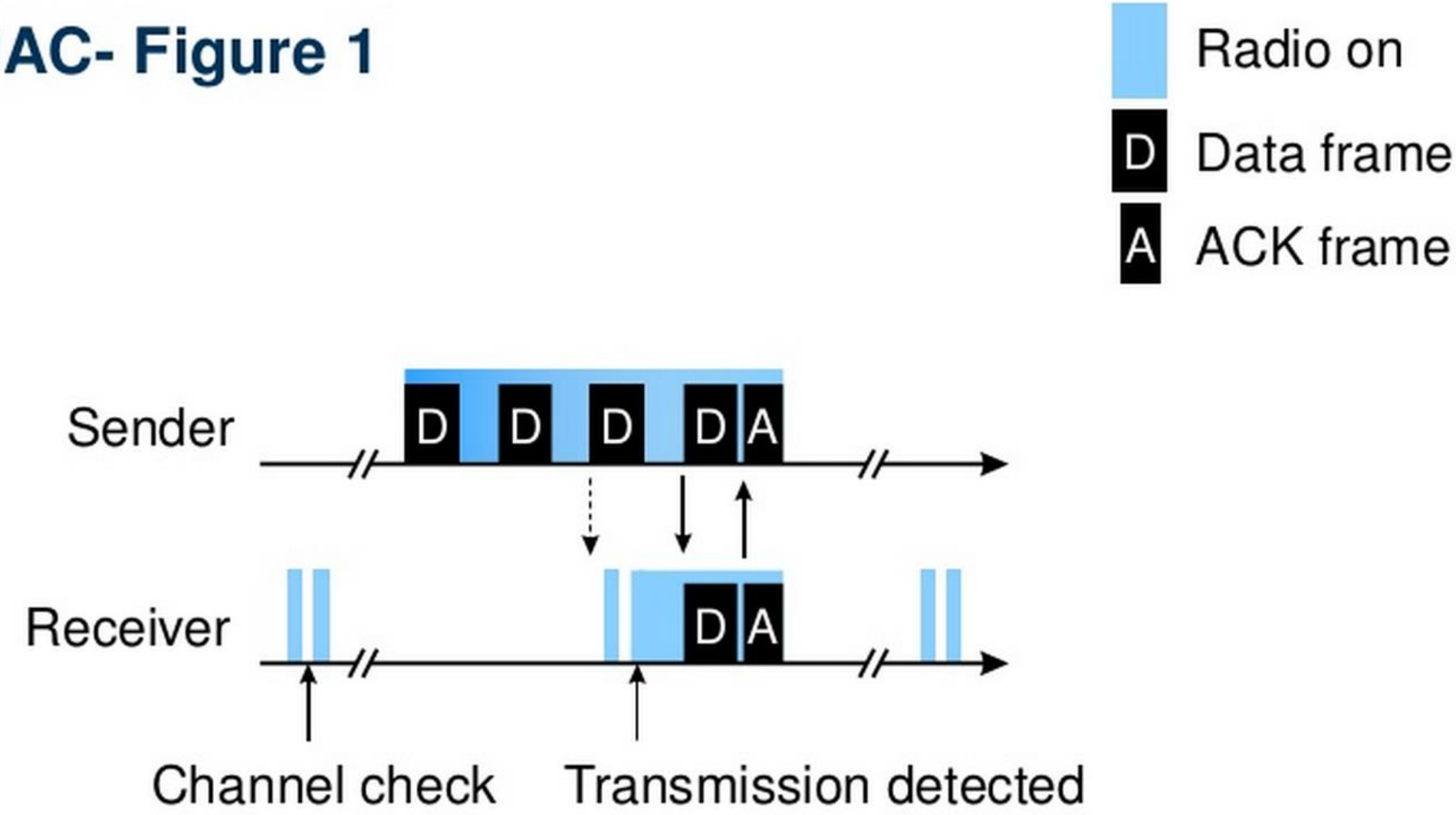
- Amennyiben egy node nem részese semmilyen kommunikációnak kikapcsolja a rádióját

- Minden node periódikusan „belehallgat” a rádiós csatornába, hogy esetlegesen érzékelje hogy van e neki szánt rádiós csomag

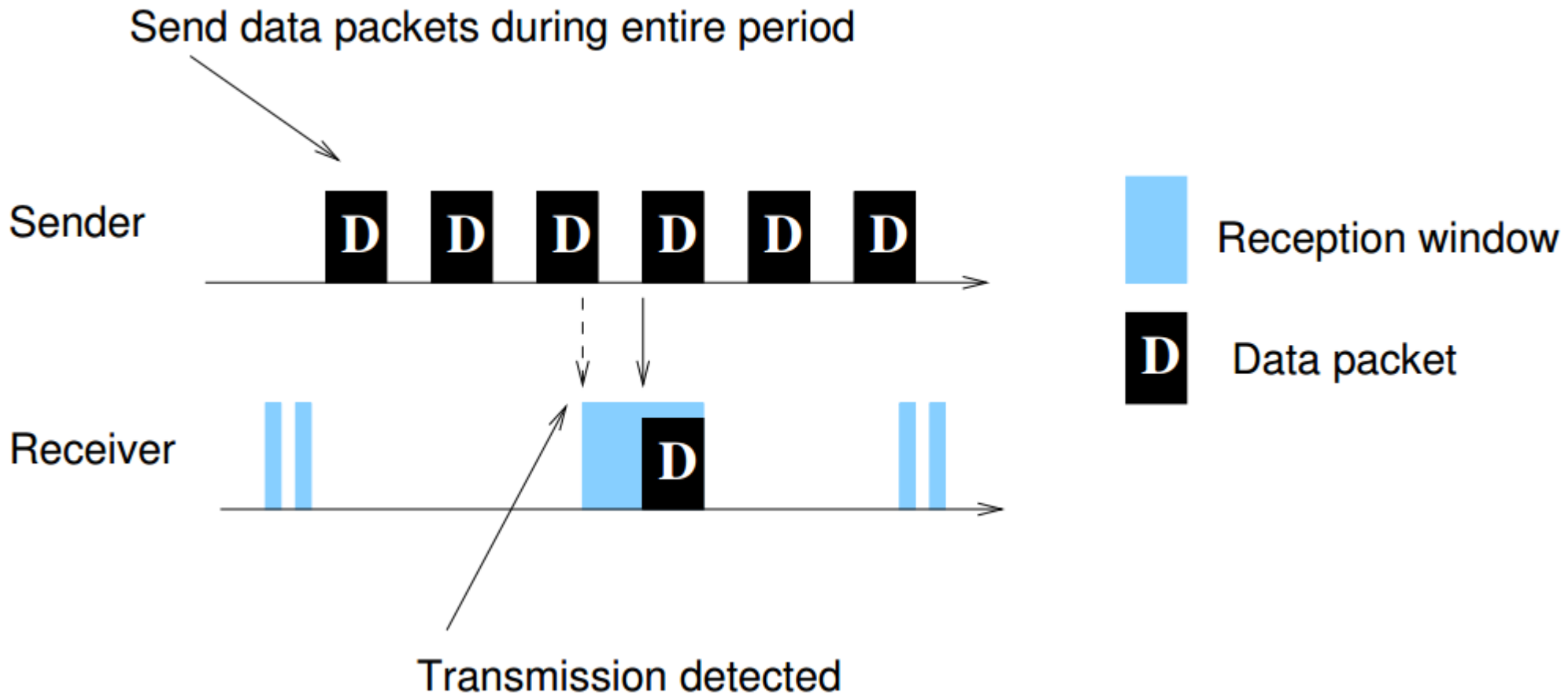
- Amennyiben van neki szánt csomag, a rádiós interfész bekapcsolva marad

ContikiMAC

ContikiMAC- Figure 1



ContikiMAC


Send data packets during entire period
Sender
D
D
D
D
D
D
Receiver
D
Transmission detected
Reception window
D
Data packet

ContikiMAC - Ébrenlét fázistudatosság

- Minden esetben, ha egy node üzenetet küld a szomszédjának, megjegyzi hogy az milyen időpillanatban volt ébren (ACK üzenet)
- Ennek tudatában képes kiszámítani a jövőbeni érbenlétfázisait
- Legközelebbi üzenetküldéskor ehhez a fázishoz igazítja az üzenetküldést
- Ha elhibázza ezt a fázist, akkor folytatja az üzenetcsomag ismételt küldését, mintha először küldene ennek a node-nak