

Bezvadu Sensoru Tīkli

Programmēšanas abstrakcijas

Reinholds Zviedris
Datorikas fakultāte
Latvijas Universitāte
09.11.2016.

KĀ SENSORU TĪKLU
PROGRAMMĒŠANU
PADARĪT VIENKĀRŠĀKU

Sensoru tīklu galvenais uzdevums:

Savākt informāciju!
(Un darīt to ilgi!)

Informācijas savākšana:

- Jēlo datu interpretācija
- Informācijas apstrāde, agregācija
- Rezultātu nogādāšana līdz apstrādes vietai

Kāpēc nesūtīt visu uz bāzes staciju?



- Komunikācija tērē enerģiju
- Radio kapacitāte ir ierobežota

Galvenās pieejas

- Programmēt katru mezglu veidu atsevišķi
- Makro programmēšana
- Aģentu bāzētas pieejas
- Vaicājumu (query) bāzētas pieejas

Makro programmēšana

- Ļauj programmēt visu tīklu kopumā
- Kompilators globālo programmu nokompilē individuāliem mezgliem

Makro programmēšanas priekšrocības un trūkumi

- + Ļauj domāt par visu tīklu kopumā
- Šāda paradigmas maiņa nav vienkārša
- Sarežģīta kompilatoru būve

Makro programmēšanas piemērs: Pleaides

- Specifiska valoda, kas kompilējas uz TinyOS programmām

```

1: #include "pleiades.h"
2: boolean nodelocal isfree=TRUE;
3: nodeset nodelocal neighbors;
4: node nodelocal neighborIter;

5: void reserve(pos dst) {
6:   boolean reserved=FALSE;
7:   node nodeIter,reservedNode=NULL;
8:   node n=closest_node(dst);
9:   nodeset loose nToExamine=add_node(n, empty_nodeset());
10:  nodeset loose nExamined=empty_nodeset();

11:  if(isfree@n) {
12:    reserved=TRUE; reservedNode=n;
13:    isfree@n=FALSE;
14:    return;
15:  }

16:  while(!reserved && !empty(nToExamine)){
17:    cfor(nodeIter=get_first(nToExamine);nodeIter!=NULL;
        nodeIter = get_next(nToExamine)){
18:      neighbors@nodeIter=get_neighbors(nodeIter);
19:      for(neighborIter@nodeIter=get_first(neighbors@nodeIter);
        neighborIter@nodeIter!=NULL;
        neighborIter@nodeIter=get_next(neighbors@nodeIter)){
20:        if(!member(neighborIter@nodeIter,nExamined))
21:          add_node(neighborIter@nodeIter,nToExamine);
22:      }
23:      if(isfree@nodeIter){
24:        if(!reserved){
25:          reserved=TRUE; reservedNode=nodeIter;
26:          isfree@nodeIter=FALSE;
27:          break;
28:        }
29:      }
30:      remove_node(nodeIter,nToExamine);
31:      add_node(nodeIter,nExamined);
32:    }
33:  }
34:}

```

Ko dara šī
programma?

```

1: #include "pleiades.h"
2: boolean nodelocal isfree=TRUE;
3: nodeset nodelocal neighbors;
4: node nodelocal neighborIter;

5: void reserve(pos dst) {
6:   boolean reserved=FALSE;
7:   node nodeIter,reservedNode=NULL;
8:   node n=closest_node(dst);
9:   nodeset loose nToExamine=add_node(n, empty_nodeset());
10:  nodeset loose nExamined=empty_nodeset();

11:  if(isfree@n) {
12:    reserved=TRUE; reservedNode=n;
13:    isfree@n=FALSE;
14:    return;
15:  }

16:  while(!reserved && !empty(nToExamine)){
17:    cfor(nodeIter=get_first(nToExamine);nodeIter!=NULL;
        nodeIter = get_next(nToExamine)){
18:      neighbors@nodeIter=get_neighbors(nodeIter);
19:      for(neighborIter@nodeIter=get_first(neighbors@nodeIter);
        neighborIter@nodeIter!=NULL;
        neighborIter@nodeIter=get_next(neighbors@nodeIter)){
20:        if(!member(neighborIter@nodeIter,nExamined))
21:          add_node(neighborIter@nodeIter,nToExamine);
22:      }
23:      if(isfree@nodeIter){
24:        if(!reserved){
25:          reserved=TRUE; reservedNode=nodeIter;
26:          isfree@nodeIter=FALSE;
27:          break;
28:        }
29:      }
30:      remove_node(nodeIter,nToExamine);
31:      add_node(nodeIter,nExamined);
32:    }
33:  }
34:}

```

Atbilde:
autostāvvietas
lietotne, kas
realizēta
Pleiades

Aģentu bāzētas programmas

- Kas ir aģents?

Aģentu bāzētas programmas

- Aģents ir programma, kas ceļo starp mezgliem
- Nevis dati nāk pie programmas, bet programma iet pie datiem
- Aģenti autonomi pārvietojas
- Vienlaikus tīklā var darboties vairāki aģenti
- Kaut kas ļoti eksotisks

Aģentu priekšrocības

- Ļoti plašas iespējas
- Iespējami mazs datu pārraides apjoms
- Spēcīga decentralizācija, apstrāde seko datiem
- Spēja dinamiski reaģēt uz izmaiņām tīklā

Aģentu trūkumi

- Sarežģīti modelēt tīkla komponentes un to sadarbību
- Grūti paredzēt izmaiņas, uz kurām jāreaģē
- Sarežģīta testēšana un atklūdošana

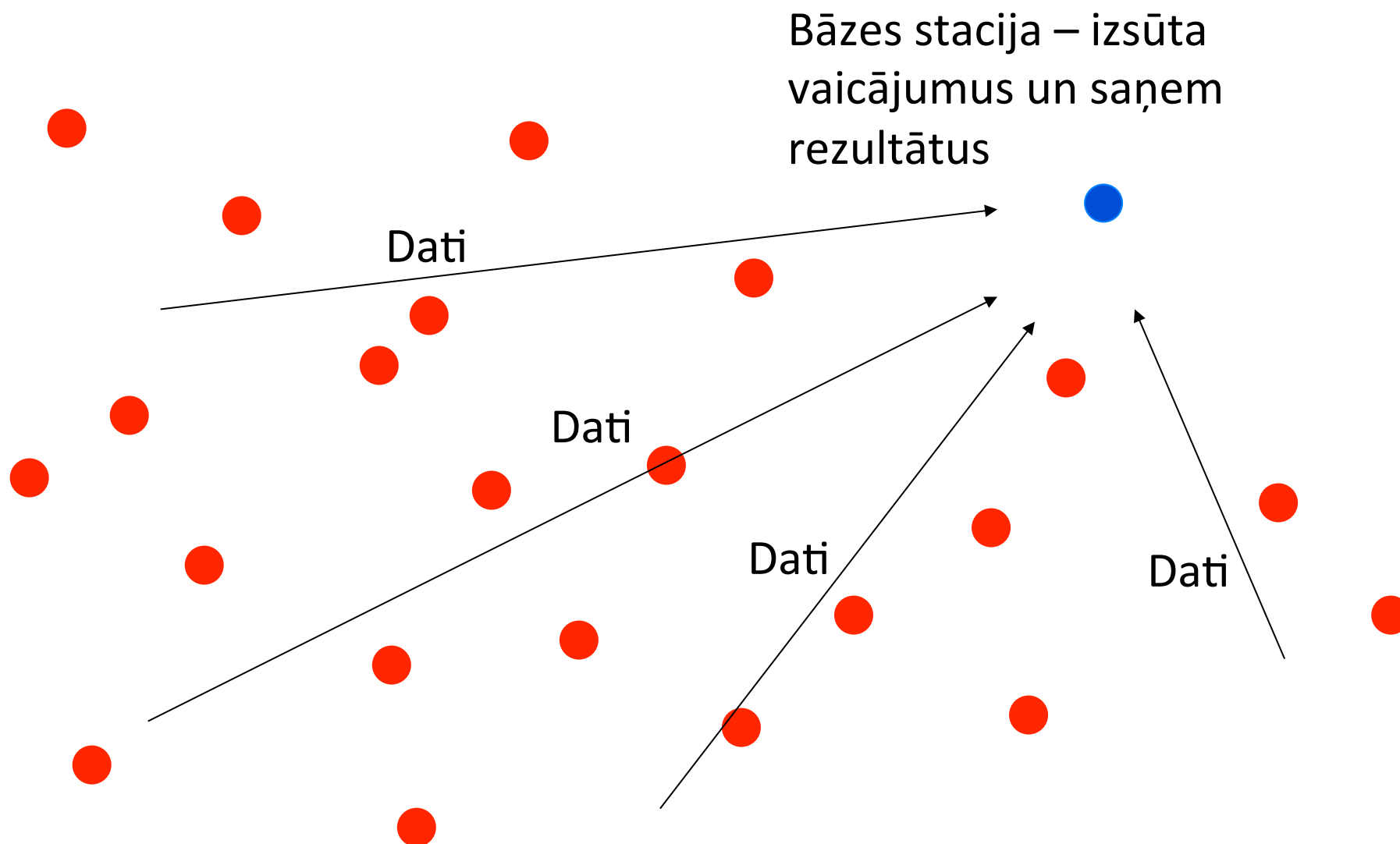
Vaicājumu bāzēta
BST
programmēšana

Vaicājumu bāzētas pieejas

- Tīklā iesūta vaicājumu, atpakaļ nāk atbilde
- Parasti centralizēti risinājumi: bāzes stacija vaicā, tīkls atbild
- Vispopulārākā pieeja – vienkārša realizācija

Piemēri: TinyDB, SwissQM, ...

Tipiska vaicājumu pieejas arhitektūra



Pieejas pamatmērķis

- Sensoru tīkls kā dalīta (*distributed*) datu bāze
- Problēmas:
 - Ierobežota atmiņa
 - Nepieciešams kopīgs *duty cycle*
 - Mezgli parasti nav uzticami
 - Dati var būt trokšņaini
 - Parasti datiem svarīgs arī laiks un vieta

Vaicājumu pieejas priekšrocības

- Vienkārša realizācija
- Abstrahēšanās no tīkla topoloģijas

Vaicājumu pieejas trūkumi

- Pārsvarā triviāla datu apstrāde
- Vaicājums parasti vai nu konkrētam mezglam, vai visam tīklam
- (Sarežģīti apstrādāt telpas un laika atribūtus)

Vaicājumu pieejas piemērs: TinyDB

- SQL-tipa dalītā datu bāze sensoru tīkliem
 - Darbojas kā slānis virs TinyOS
 - Visi sensoru mezgli kā daļa no tabulas
 - Katrs sensoru lasījums kā viens ieraksts
 - Vaicājumus izsūta un atbildes savāc bāzes stacija
-
- Sīkāk rakstā: S. Madden, M. Franklin, J. Hellerstein, and W. Hong, “TinyDB: an acquisitional query processing system for sensor networks,” ACM Transactions on Database Systems (TODS), vol. 30, no. 1, pp. 122–173, 2005.

TinyDB vaicājuma piemērs

```
SELECT nodeid, light, temp  
FROM sensors  
SAMPLE PERIOD 1s FOR 10s
```

Lasa gaismu un temperatūru ik sekundi, 10
sekunžu garumā

TinyDB agregācija

- Datu agregāciju veic pēc iespējas tuvu datu avotam, notiek automātiski

```
SELECT AVG(volume), room FROM sensors  
WHERE floor = 6  
GROUP BY room  
HAVING AVG(volume) > threshold  
SAMPLE PERIOD 30s
```

TinyDB kopsavilkums

- Dalītā datu bāze sensoru tīkliem
- Slānis virs TinyOS [1.x]
- Piedāvā vienkāršu saskarni, efektīvu agregāciju

Trūkumi:

- Paplašināmība
- Iebūvēta vaicājumu valoda ierobežo
 - nevar veidot patvaļīgas programmas

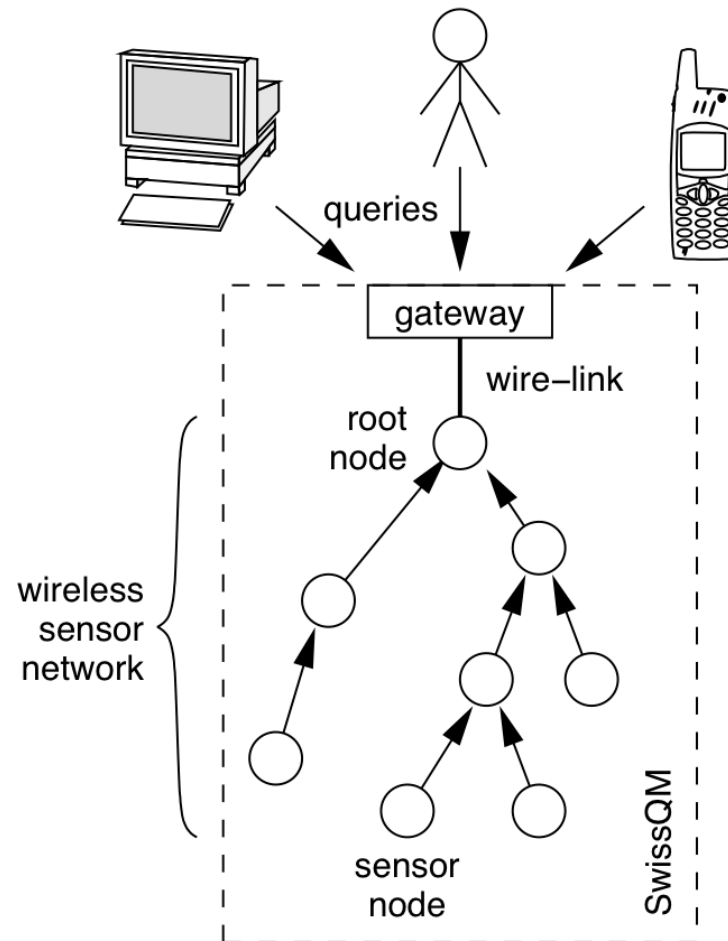
SwissQM virtuālā mašīna

- ETH Zurich, 2007.g
- Izteiksmīgāka par TinyDB
- Kompaktāka par Mate VM
- QM = Query Machine
 - Query [Engine] + [Virtual] Machine



SwissQM pieeja

- Vārteja (*gateway*) kā "gudrais mezgls"
 - vaicājumu pārveidošana mezglu programmās
 - vaicājumu apvienošana (*merging*)
 - daudzu vaicājumu apvienošana (*multi-query optimization*)
- Paaugstina efektivitāti
 - motes izpilda tikai to, kas vajadzīgs (*sense, aggregate, transmit*)



Vārteja

- Daudz plašāka funkcionalitāte kā pierasts
 - *multi-query optimization and merging*
- Var “klausīt” dažādām vaicājumu valodām
 - SQL, XQuery, web services (t.sk. vienlaikus)
- Paplašināms
 - var realizēt dažādu papildus funkcionalitāti
- Ģenerē *byte-code*, ko izpildīt meglu VM

XQuery piemērs

Atgriezt NodeID motēm, kam temp > 60 :

```
for $n in xt:sample($sensors, 10s)//node
  where $n/temp gt 60
  return $n/nodeid
```

vaicājums XML “dokumentam” ar <node/> elementiem, kas satur <nodeid> un <temp>.

Agregācijas piemērs (SQL)

Atrast vidējo temperatūras vērtību nodēm, kam ir līdzīgi gaismas sensora rādījumi:

```
SELECT (light-512) / 10, AVG(temp)
FROM sensors GROUP BY (light-512) / 10
SAMPLE PERIOD 30s
```

vaicājums satur uz motēm izpildāmus aprēķinus

– TinyDB ir ļoti minimālas iespējas to veikt

SwissQM kopsavilkums

- Vienkāršāk programmējams kā Mate VM
- Plašākas iespējas par TinyDB
 - Dažāda veida interfeisi (XQuery, SQL, ...)
 - Multi-user, multi-programming
- Adaptējama / pielāgojama sistēma

MansOS

- Izstrādāta LU un EDI
- Paredzēta bezvadu sensoru tīkliem un resursierobežotām iegultajām iekārtām
- Mērķauditorija ir programmētāji ar C un UNIX programmēšanas pieredzi

MansOS pamatprincipi

- Viegli lietojama
- Portējama

Viegli lietojama BST OS

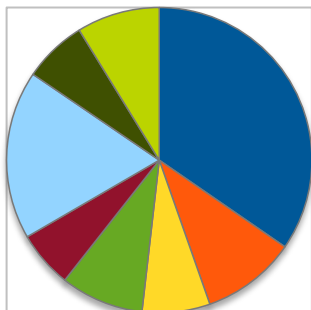
- Sistēmprogrammētājiem ar UNIX un C pieredzi
- Piemēram, TinyOS prasa specifiskas valodas (nesC) un programmēšanas paradigmas (event-based execution) apguvi
- Contiki prasa vismaz *protothreads* abstrakcijas apguvi
- MansOS pārsvarā var programmēt līdzīgi kā “klasiskas” UNIX sistēmas
- Uzinstalēt TinyOS ir netriviāli
- MansOS piedāvā *all included* binārās distribūcijas

Portējama BST OS

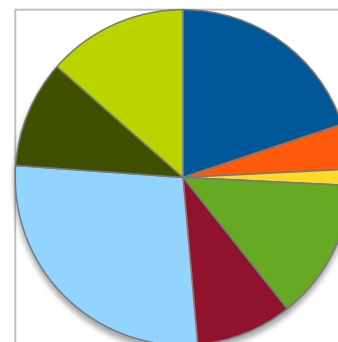
- Aparatūra bieži ir specifiska (vienam) lietojumam
- Daudz eksistējošo platformu, maz standartu
- MansOS darbojas gan uz MSP430, gan uz AVR arhitektūrām
 - t.sk. uz *Tmote Sky*, *Atmega (Arduino)*, *Zolertia Z1*, *AdvanticSYS XM1000*, *TI MSP430 Launchpad* u.c. platformām

MansOS pirmkoda sadalījums

•Komponentu proporcijas:



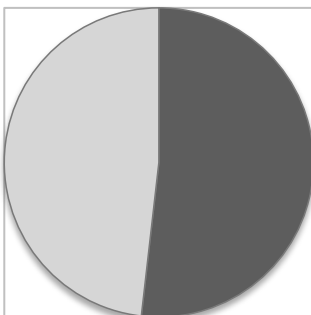
•a) all



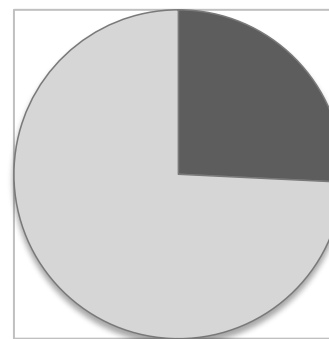
•b) TelosB

- Chip-specific code
- Architecture-specific code
- Platform-specific code
- Interface layer code
- Kernel code
- Network protocol code
- File system code
- Library code

•Aparatūrneatkarīgā koda proporcijas:



•a) all



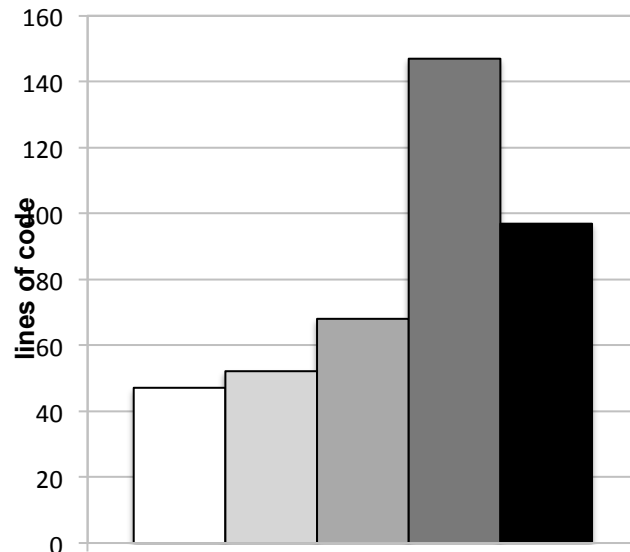
•b) TelosB

- Device-dependent code
- Device-independent code

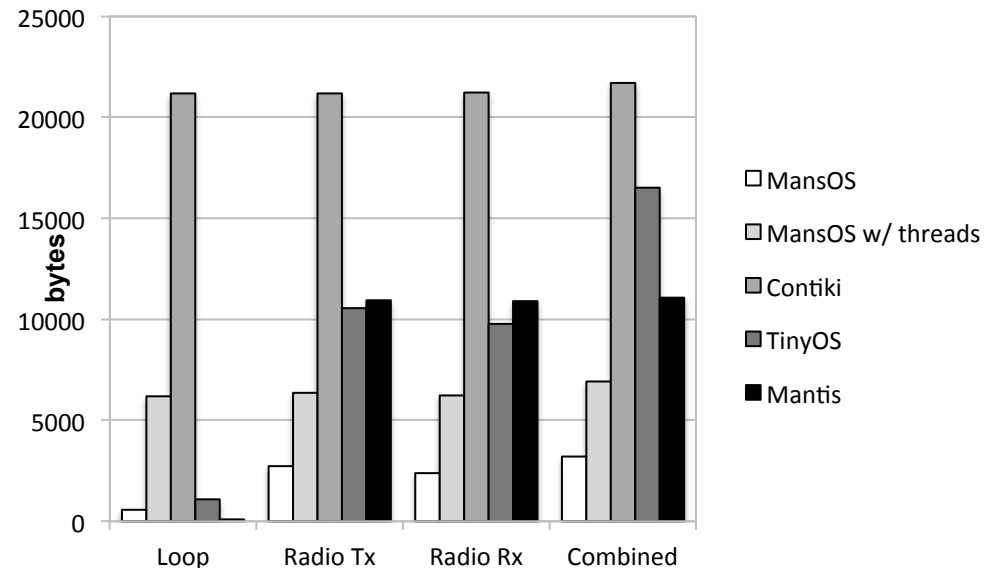
MansOS priekšrocības

- Īsāks lietotņu pirmkods
- Mazāks lietotņu binārais kods
- Vienkārši, bet robusti pavedieni (*threads*)

• *Lietotnes koda rindiņu skaits*



• *Četru dažādu lietotņu binārā koda izmērs*



MansOS funkcionalitāte

- Analogo un digitālo I/O portu apstrāde
- Digitālie piekļuves protokoli (SPI, I²C, u.c.)
- Zema enerģijas patēriņa režīmi
- Radio komunikācija un tīkla steks
- Iespējams IPv6 atbalsts
- Vairākas aparatūras platformas
- *Run-time* pārvaldība
- *Over The Air* pārprogrammēšana

MansOS pavedieni

- Vieglāk programmēt!
- Koda izkārtojums labāk atspoguļo programmas izpildīšanos
- Var izveidot bezgalīgos ciklus, bet sistēma turpinās strādāt...

Pavedienu pamatidejas

- Pavedienu ir maz (bieži pietiek ar diviem)
- Pavedieni ir savstarpēji “draudzīgi”
- Tiek darbināti “pa riņķi” (*round robin scheduling*)
- Kodola pavedienam vienmēr priekšroka
- Nianse - ***sleep()*** ieliek sistēmu zema enerģijas patēriņa režīmā ***tikai*** tad, ja citi pavedieni neaktīvi!

Papildus informācija par MansOS

- A. Elsts, G. Strazdins, A. Vihrov, and L. Selavo, “Design and Implementation of MansOS: a Wireless Sensor Network Operating System,” Scientific Papers, University of Latvia, Volume 787, pp. 79-105, 2012.
- <https://github.com/IECS/MansOS>

SEAL valoda

- SEAL – **S**ensor **A**pplication Development Language
- Domēnspecifiska valoda BST lietojumu izstrādei
- Balstīta uz MansOS
- Komplektēta ar vizuālo programmēšanas vidi (IDE)

NesC kods (fragments):

```
#include "Timer.h"
#include "RadioSenseToLeds.h"

module RadioSenseToLedsC @safe(){
  uses {
    interface Leds;
    interface Boot;
    interface Receive;
    interface AMSend;
    interface Timer<TMilli> as MilliTTimer0;
    interface Timer<TMilli> as MilliTTimer1;
    interface Packet;
    interface Read<uint16_t>;
    interface SplitControl as RadioControl;
  }
}

implementation {
  message_t packet;
  bool locked = FALSE;

  event void Boot.booted() {
    call RadioControl.start();
  }

  event void RadioControl.startDone(error_t err) {
    if (err == SUCCESS) {
      call MilliTTimer0.startPeriodic(2000);
      call MilliTTimer1.startPeriodic(6000);
    }
  }
  event void RadioControl.stopDone(error_t err) {}

  event void MilliTTimer0.fired() {
  }

  event void MilliTTimer1.fired() {
  }

  event void Read.readDone(error_t result, uint16_t data) {
    if (locked) {
      return;
    }
    else {
      radio_send_msg_t* msg;
    }
  }
}
```

C kods:

```
#include "stdmansos.h"
#include <hil/alarm.h>
#include <string.h>

#define SENSOR_READ_INTERVAL 3000 // milliseconds

// our timer
Alarm_t timer;

//-----
// Timer callback
//-----
void onTimer(void *param)
{
  // read light sensor value
  uint16_t light = readLight();
  PRINTF("light = %u\n", light);

  // send the value read to radio
  radioSend(&light, sizeof(light));

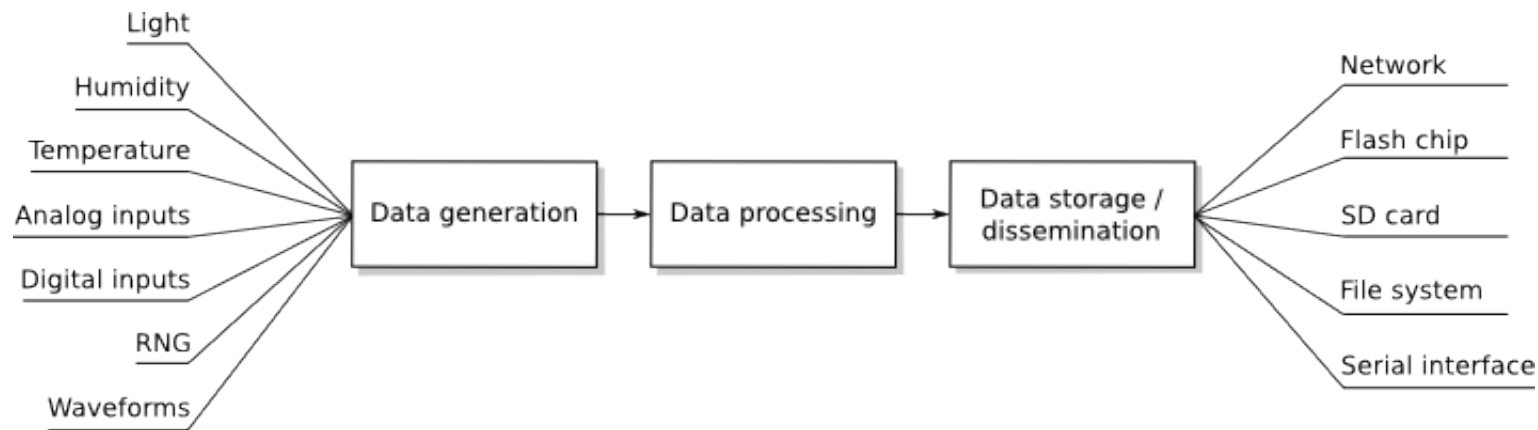
  // blink LED
  toggleRedLed();
}

//-----
// Entry point for the application
//-----
void appMain(void)
{
  // initialize and schedule the alarm
  alarmInitAndRegister(&timer, onTimer, SENSOR_READ_INTERVAL, true);
  // set radio packet receive callback
  radioSetReceiveHandle(onRadioInterrupt);
  // turn radio listening on
  radioOn();
  // our job here is complete
  return;
}
```

SEAL kods:

```
read Light, period 2s;
read Humidity, period 2s;
output Radio;
```

Datu plūsma BST motē

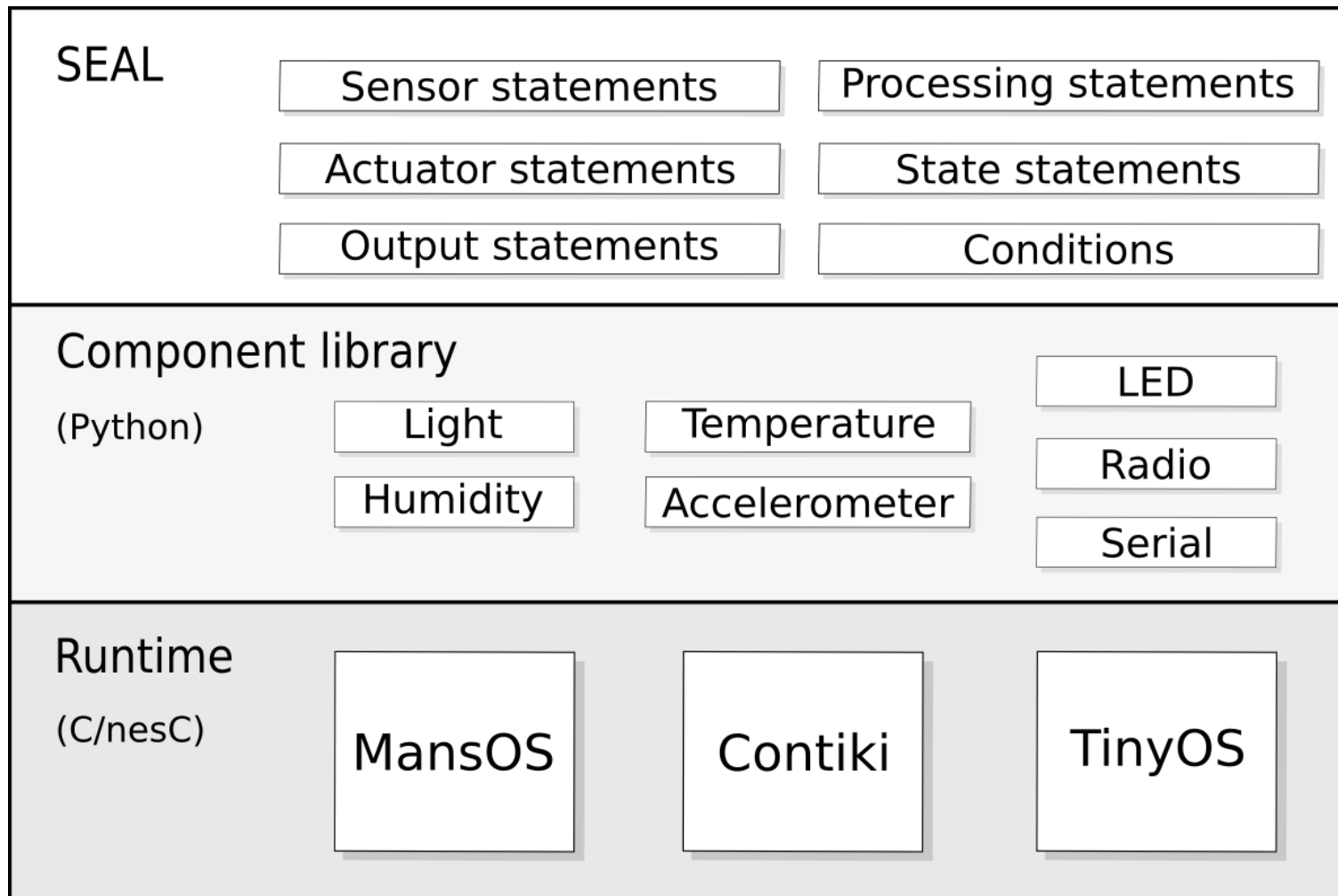


- Datus **ievāc** ar sensoriem un *pseidosensoriem*
- Datus **apstrādā** programmas loģika
- Datus **izvada** tīklā, atmiņā, seriālajā saskarnē u.c.

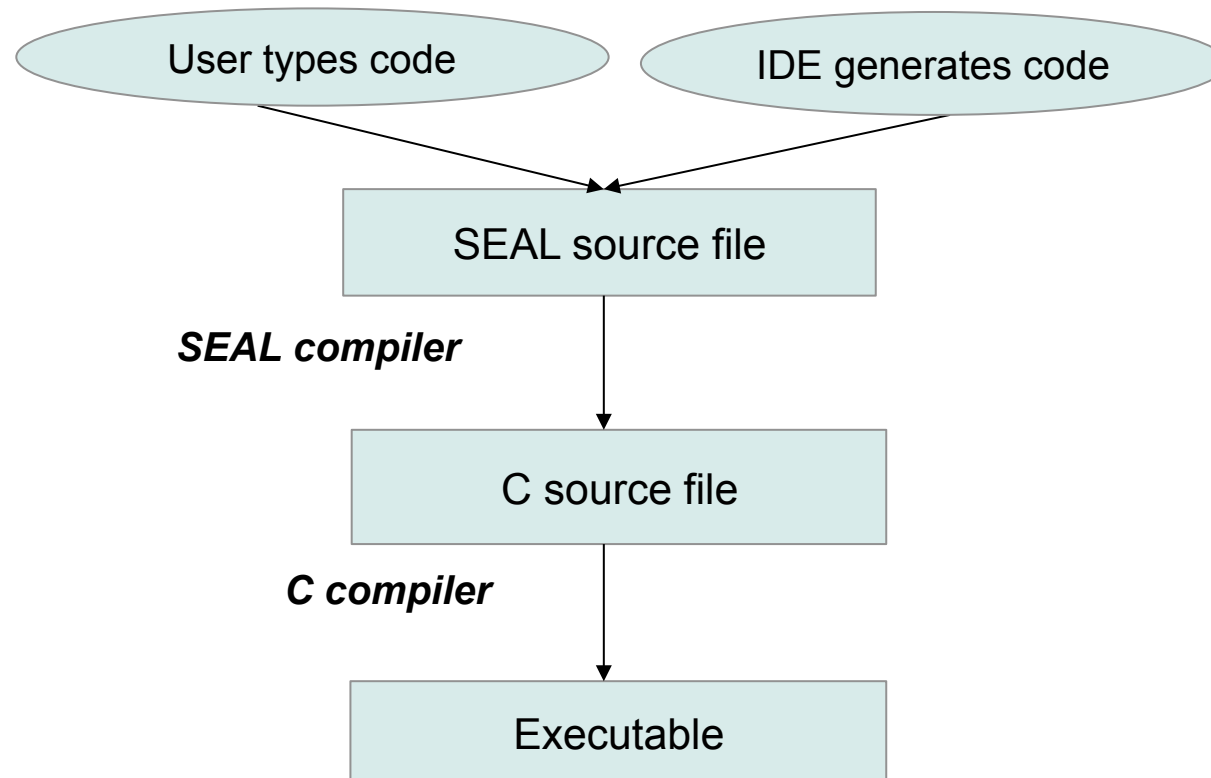
SEAL arhitektūra

- Deklaratīva ar dažiem imperatīviem elementiem
- Kompakta un dabīgi lasāma sintakse
- Divi līmeņi: sintakse & komponentu bibliotēka, kā arī “runtime support” no OS

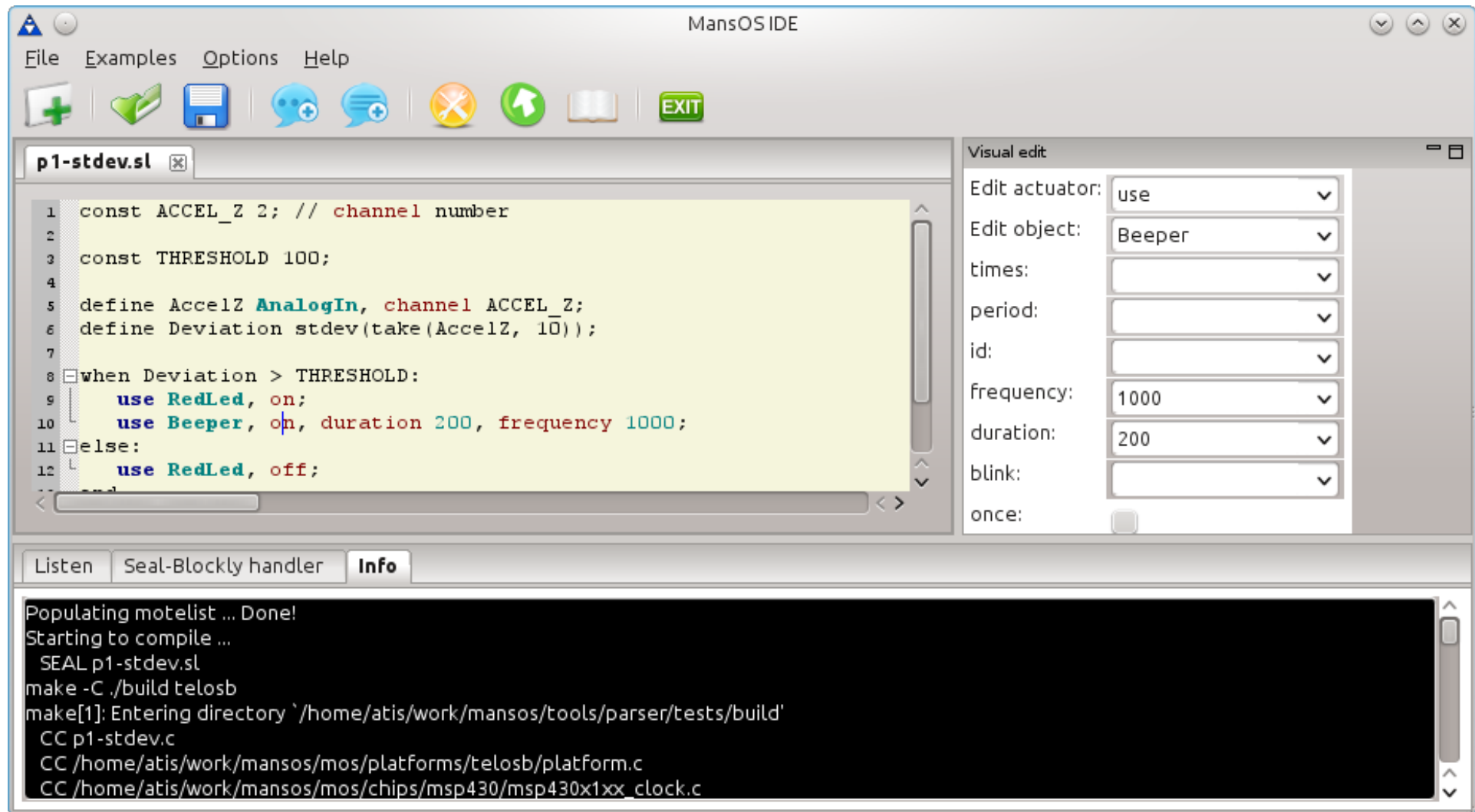
SEAL konceptuālā arhitektūra



SEAL izstrādes process



SEAL grafiskā vide



SEAL programmēšana no tīmekļa

The screenshot displays the 'Seal - Blockly Playground' web application. The browser window title is 'Seal - Blockly Playground – rekonq' and the address bar shows the file path: `file:///home/atis/work/seal-blockly/blockly/seal/playground-seal.html`.

The interface is divided into three main sections:

- Left Panel (Tools):** Contains buttons for 'Export to XML', 'Import from XML', and 'Generate SEAL'. Below these buttons is a text area displaying the generated SEAL code:

```
Use RedLed, period 1000ms;  
Use BlueLed, period 2000ms;  
Use GreenLed, period 4000ms;
```

At the bottom of this panel is a 'Try your app:' section with an 'Upload SEAL code' button.
- Center Panel (Seal):** A vertical stack of blocks representing the program. The blocks are:
 - `use RedLed`
 - `read ADC`
 - `output Radio`
 - A connector block (two white squares).
 - `period 1000 ms`
 - `baudrate 9600`
 - `value 0`
- Right Panel (Scratchpad):** A workspace for additional blocks. It contains three `use` blocks with their respective periods:
 - `use RedLed` with `period 1000 ms`
 - `use BlueLed` with `period 2000 ms`
 - `use GreenLed` with `period 4000 ms` (highlighted with a yellow border)A trash can icon is located at the bottom right of this panel.

SEAL programmas piemērs

- Ko dara šī programma?

```
read Temperature, period 10s;  
output Network;  
when Temperature > 40C:  
    use RedLed, on;  
end
```

9. eseja

Atrast un aprakstīt (ar vismaz vienu koda piemēru) vienu BST operētājsistēmu. Aprakstiet, kādiem lietojumiem tā būtu piemērota.

Termiņš: 16.11.2016. 10:00