

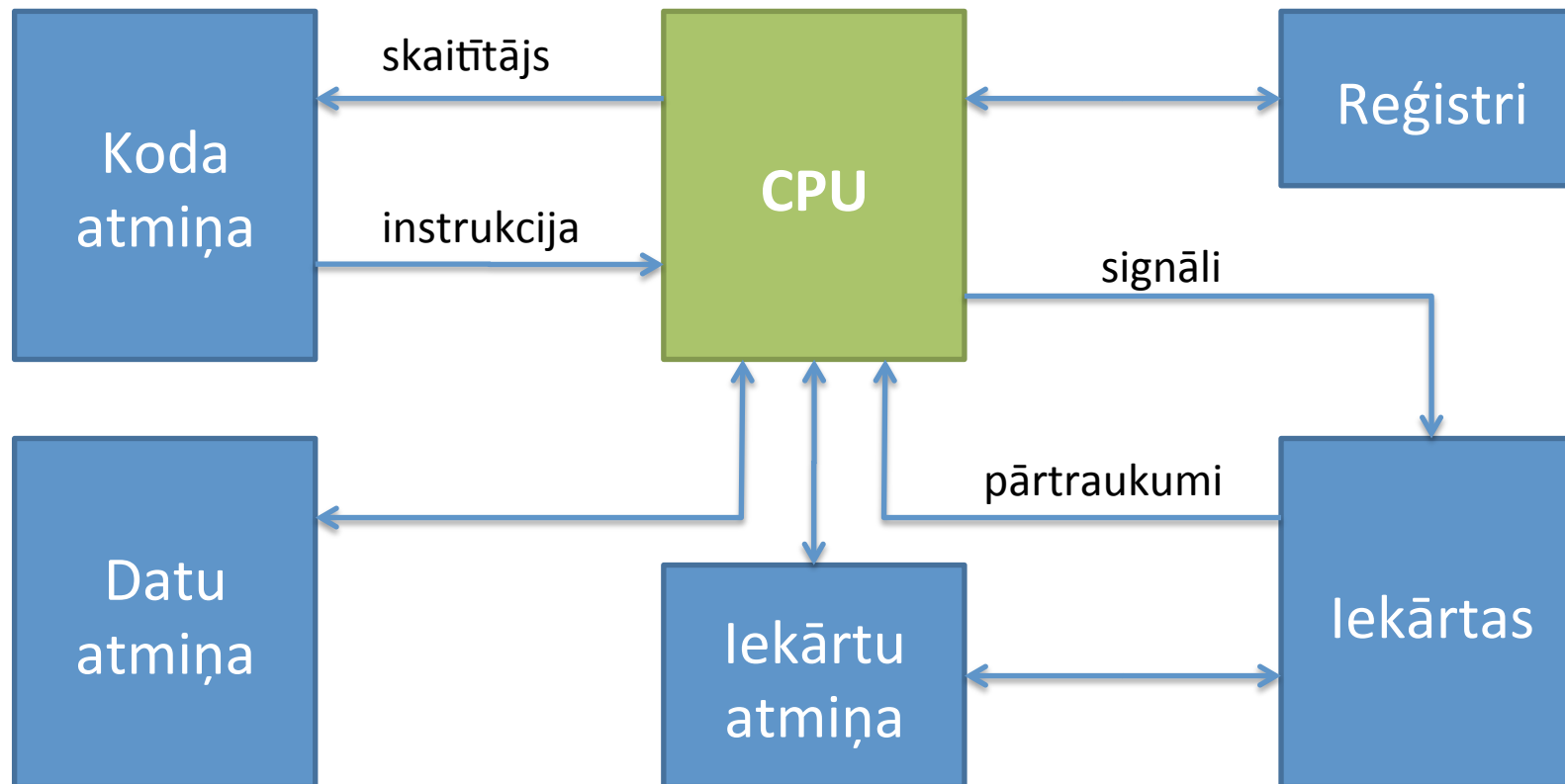
Bezvadu Sensoru Tīkli

Virtuālās mašīnas

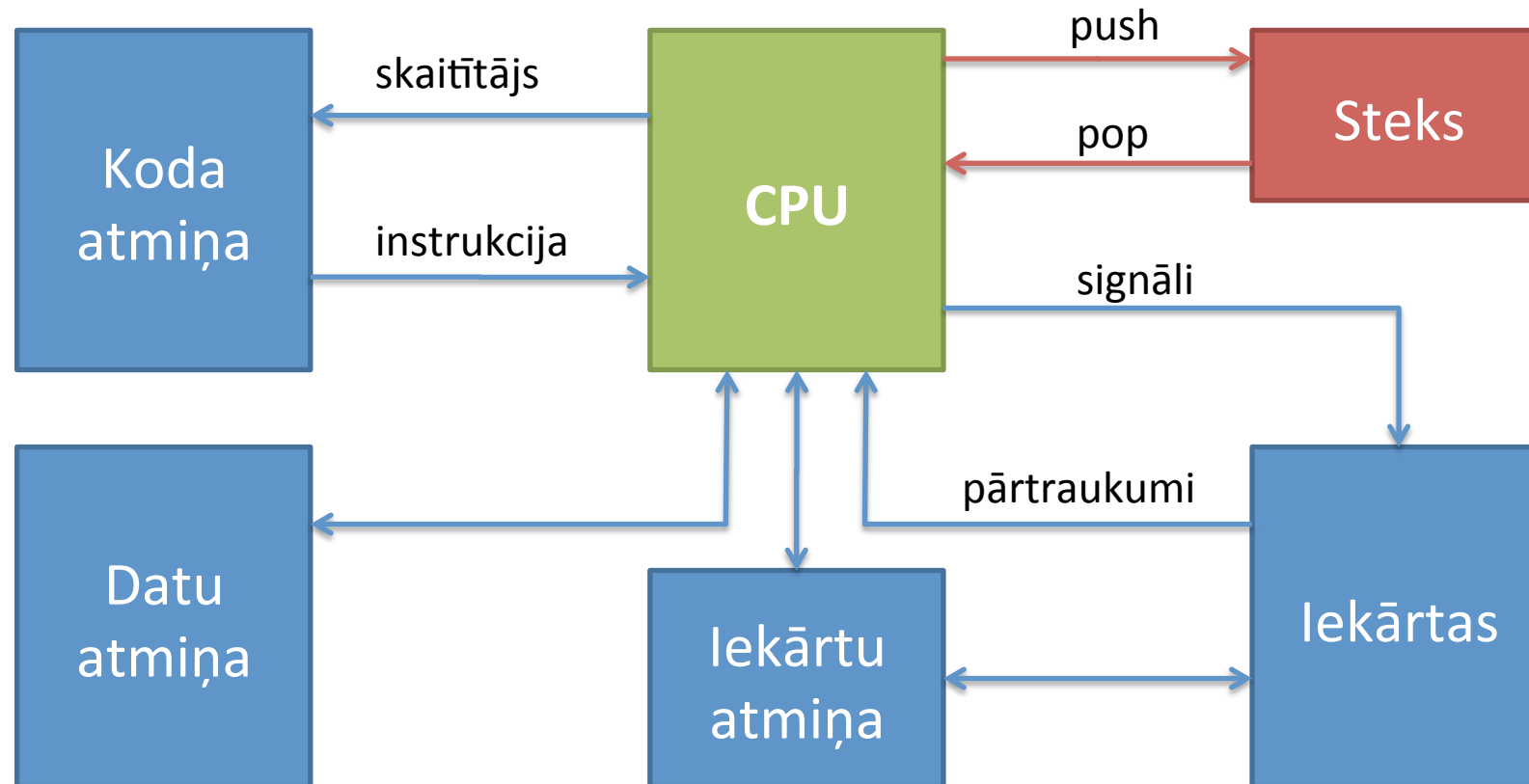
Reinholds Zviedris
Datorikas fakultāte
Latvijas Universitāte
25.11.2015.

Kā ir uzbūvēts dators?

Reģistru mašīna



Steka mašīna



Kā steks aizvieto reģistrus

- Veikt operāciju 2+3
- Rezultātu saglabāt atmiņā ar adresi 100

Reģistru pieeja:

```
set r1, 2
add r1, 3
store r1, [100]
```

Steka pieeja:

```
push 2
push 3
add
store [100]
```

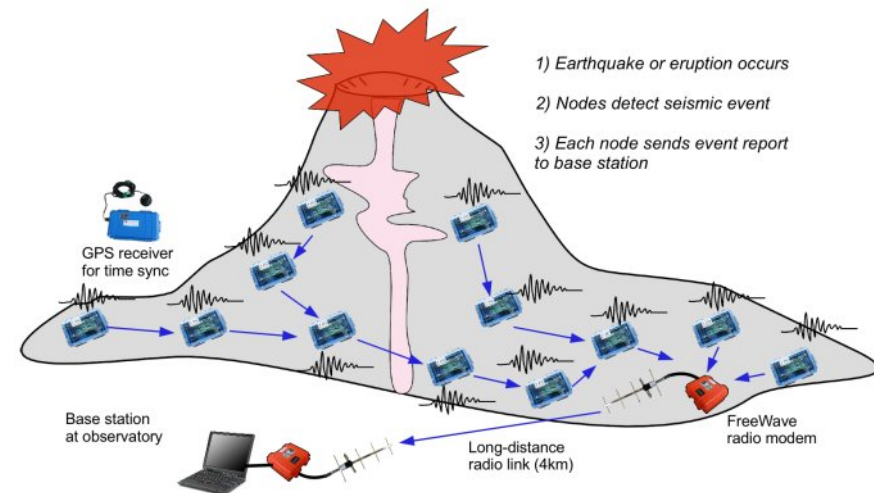
Kas labāks?

- Reģistriem: mazāk instrukciju
- Stekam: īsākas instrukcijas
- Kopsummā steka mašīnām programmas īsākas

Kāds BST labums no VM?

BST motivācija

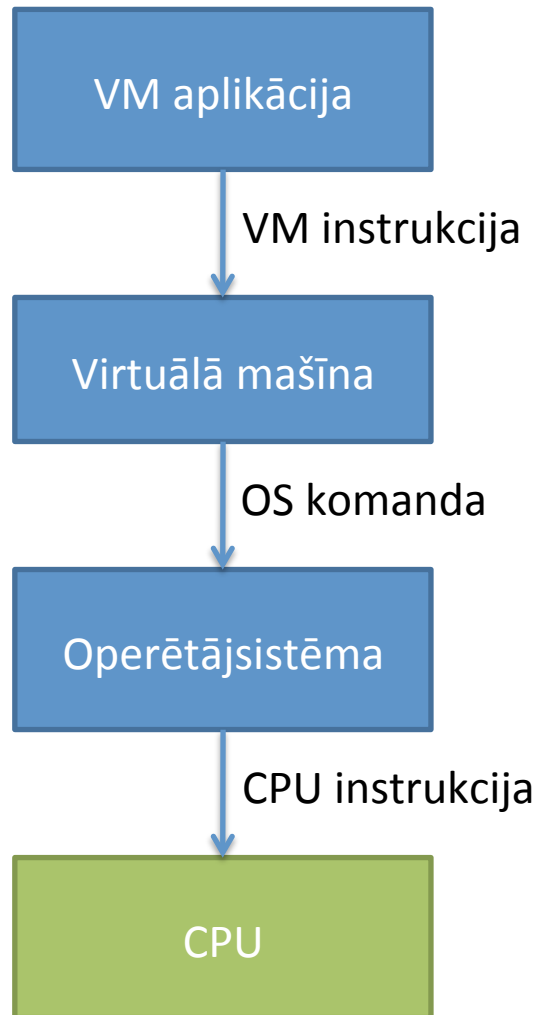
- Motes ir jā(pār)programmē
- Gribas:
 - Ātri
 - Droši
 - Plašas iespējas



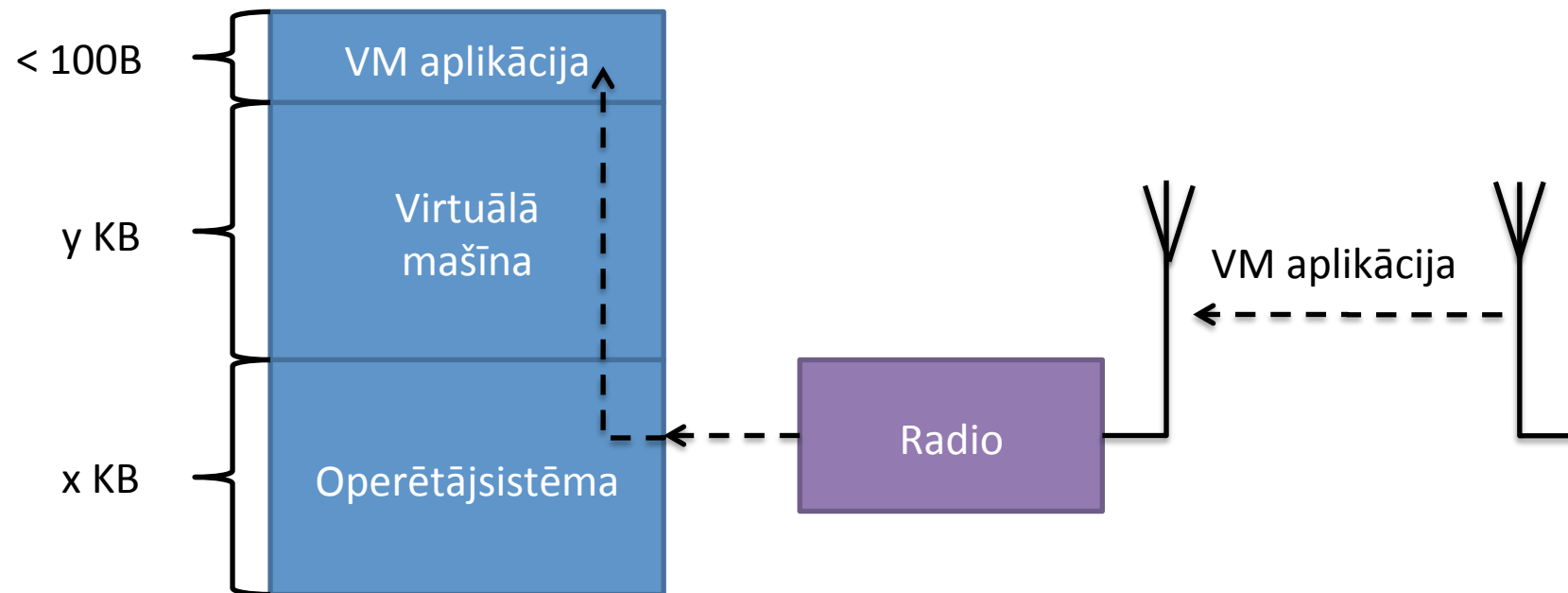
<http://fiji.eecs.harvard.edu/Volcano>

- Risinājums: samazināt koda izmēru

Kā darbojas VM uz motes?



Motes pārprogrammēšana



- VM aplikācija izmēros stipri mazāka par OS un VM kodu

Maté virtuālā mašīna

- UC Berkeley, 2002.g
- Uz komunikāciju orientēta VM sensoru tīkliem
- Mazas programmas
- Ērta programmu “vīrus-veida” izplatīšana



Maté tēja

<http://www.sciencedaily.com/releases/2007/10/071023163949.htm>

- P. Levis and D. Culler,

“Mate: A tiny virtual machine for sensor networks”
- ACM SIGARCH Computer Architecture News,
vol. 30, no. 5, pp. 85–95, 2002.

Mate mērķi

BST programmēšanas sistēmai jābūt:

- Nelielai
- Izteiksmīgai
- Kodolīgai
- Drošai
- Efektīvai
- Pielāgojamai
- Vienkāršai

Sistēmai jābūt:

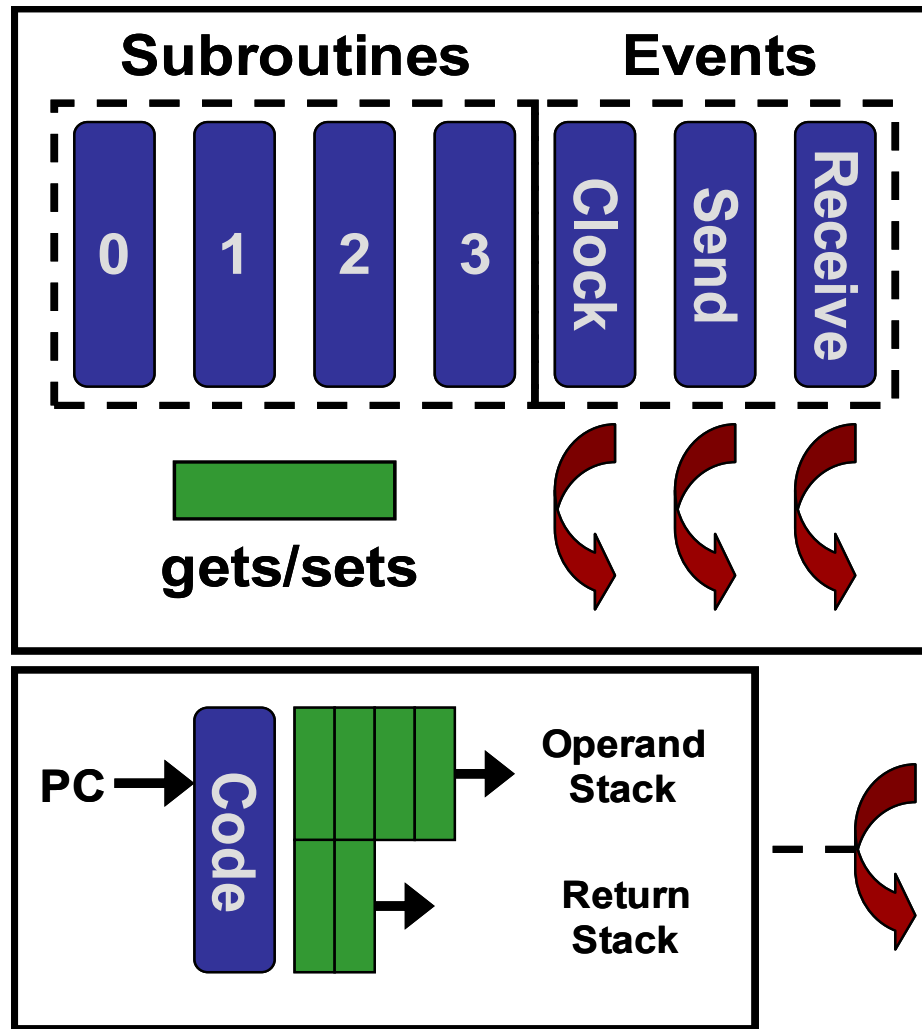
- Nelielai (lai sistēma ietilptu motes atmiņā)
- Vienkāršai (aplikāciju izstrādei)
- Izteiksmīgai (lai var radīt dažādas aplikācijas)
- Kodolīgai (arī aplikācijām jābūt īsām)
 - lai taupītu pārprogrammēšanai vajadzīgo enerģiju
- Drošai (programmas nedrīkst “nograut” moti)
- Efektīvai (jātaupa enerģija)
- Pielāgojamai

Mate instrukciju kopa

basic	00iiiiii	i = instruction
s-class	01iiiixxx	i = instruction, x = argument
x-class	1ixxxxxx	i = instruction, x = argument

- basic: aritmētika, LEDi, halt
- s-class: atmiņa
- x-class: push const, branch if <=

Mate izpildes konteksti



Mate CounterToLeds aplikācija

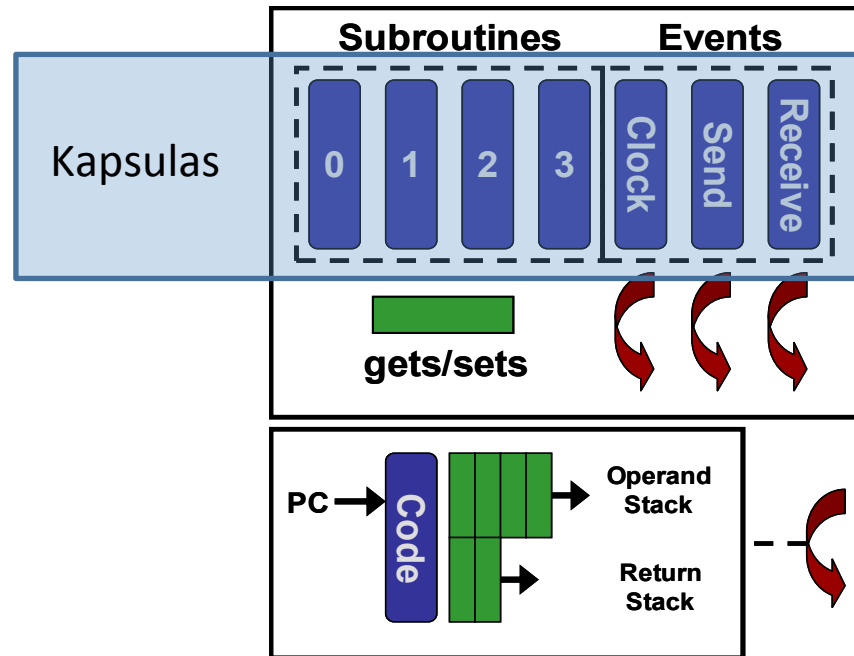
```
pushc 1      # Push one onto operand stack
add          # Add the one to the stored counter
copy        # Copy the new counter value
pushc 7
and          # Take bottom three bits of copy
putled      # Set the LEDs to these three bits
halt
```

Mate SenseAndSend

```
pushc 1    # Light is sensor 1
sense      # Push light reading on stack
pushm     # Push message buffer on stack
clear     # Clear message buffer
add       # Append reading to buffer
send      # Send message using built-in
halt     # ad-hoc routing system
```

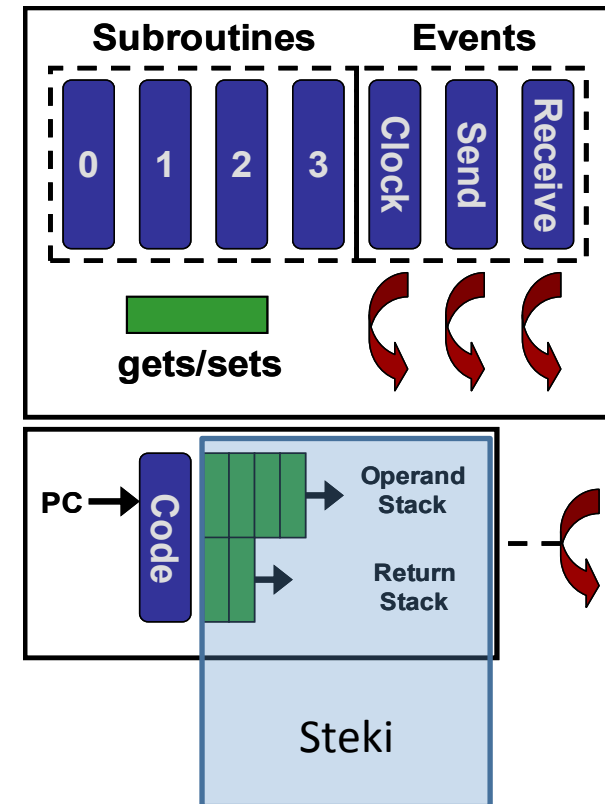
Mate koda kapsulas

- Katra kapsula viena konteksta apstrādei
- Max 24 instrukcijas kapsulā (24B)
- Ja vajag vairāk – apakšprogrammas
- Katrai kapsulai 32bitu versijas numurs



Mate steki

- Katram kontekstam 2 steki:
 - Operandu steks (16B), instrukciju parametri
 - Izsaukumu steks (8B), apakšprogrammām
- Clock konteksts steku saglabā starp izsaukumiem



Mate apakšprogrammas

```
pushc 3 // izsauc apakšprogrammu #3  
call
```

- Atgriešanās adresi ieliek izsaukumu stekā

Mate statistiskais mainīgais

- Tikai viens globālais mainīgais
- Kopīgs visiem kontekstiem
- Instrukcijas:
 - `sets`: vērtību no steka ieraksta mainīgajā
 - `gets`: otrādi

Konteksta kapsulas izpilde

- Iestājas notikums
- Tiek palaista attiecīgā kapsula
- Katra instrukcija kā TinyOS uzdevums
- Konteksti var izpildīties paralēli
- Apstājas pie `halt`

Mate SendOnSenseChange

```
pushc 1    # Push one on the operand stack
sense      # Read sensor 1 (light)
copy       # Copy the sensor reading
gets      # Get previous sent reading

inv        # Invert previous reading
add        # Current - previous sent value
pushc 32
add

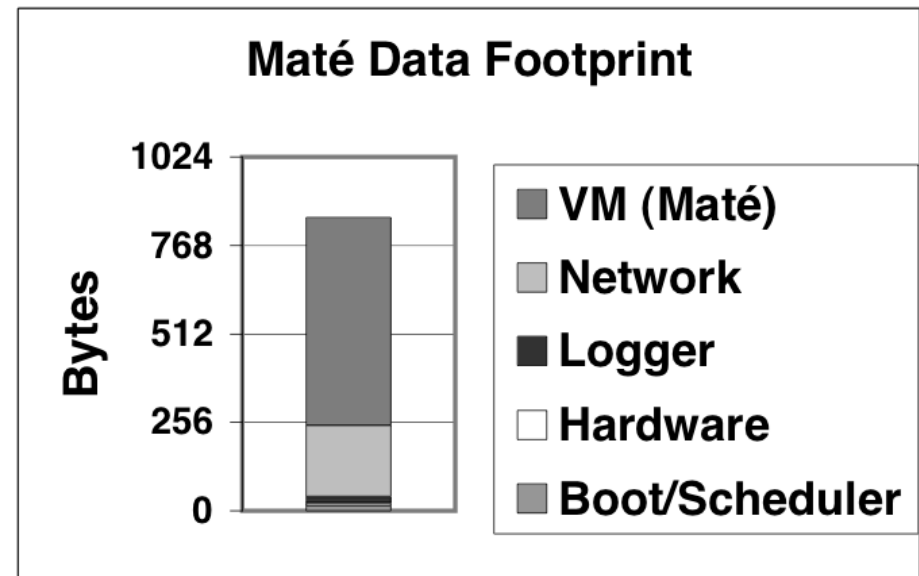
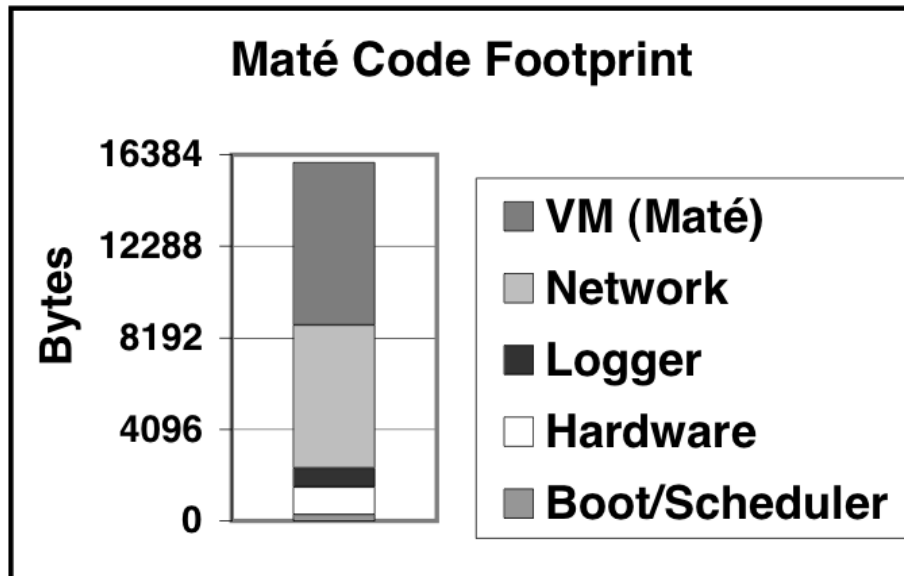
blez 17    # If curr < (prev-32) jump to send
copy       # Copy the sensor reading
inv        # Invert the current
gets      # Get the previous reading

add        # Previous - current
pushc 32
add
blez 17    # If (curr+32) > prev jump to send

halt
copy       # PC 17 -- jump-to point from above
sets      # Set shared var to current reading
pushm     # Push a message onto operand stack

clear     # Clear out the message payload
add       # Add the reading to message payload
send      # Send the message
halt
```


Mate aplikācijas izmēri



Component	Code (bytes)	Data (bytes)
VM (Maté)	7286	603
Network	6410	206
Logger	844	18
Hardware	1232	8
Boot/Scheduler	272	14
Total	16044	849

Mate koda izmēri

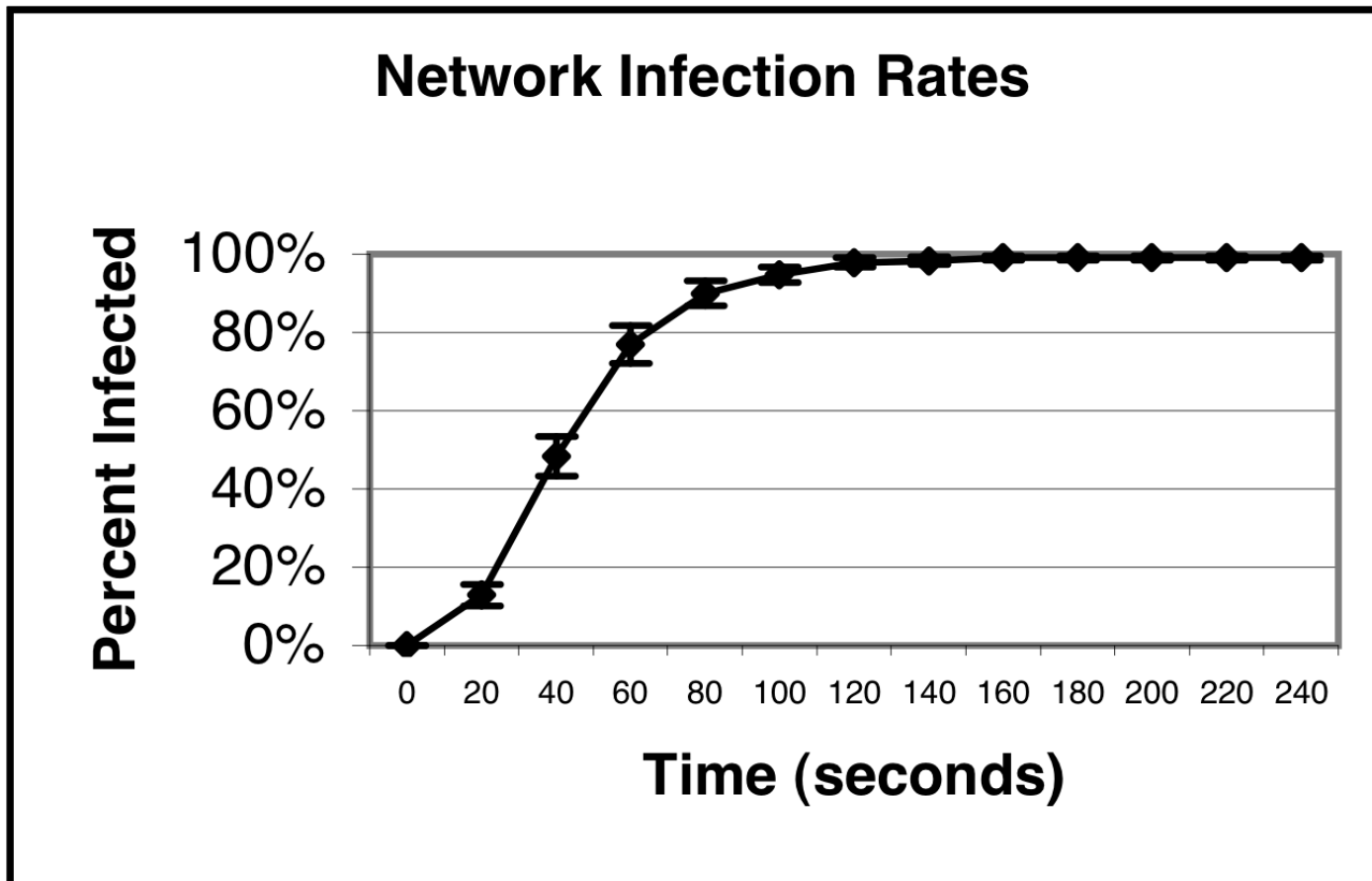
Application	Binary		Maté	
	Size(bytes)	Install Time	Capsules	Instructions
<code>sens_to_rfm</code>	5394	$\approx 79s$	1	6
<code>gdi-comm</code>	7130	$\approx 104s$	1	19
<code>bless-test</code>	7381	$\approx 108s$	7	108

Mate veiktspēja

Operation	Maté Clock Cycles	Native Clock Cycles	Cost
Simple: and	469	14	33.5:1
Downcall: rand	435	45	9.5:1
Quick Split: sense	1342	396	3.4:1
Long Split: sendr	685 + $\approx 20,000$	$\approx 20,000$	1.03:1

- Veiktspēja cieš
- Enerģijas patēriņu tas maz ietekmē

Mate infektologija



42 motes, 3 x 14 režģis

Mate kopsavilkums

- Viena no pirmajām BST VM
- “Tālāk no dzelžiem”
 - nevar izdarīt neko sliktu
 - nav asinhrona izpilde
- Maza izmēra aplikācijas kods (<100B)
- Ērta pārprogrammēšana
- Veiktspēja zema, bet energo efektivitāte no tā daudz necieš

Vaicājumu pieeja:
Kāpēc nesūtīt visu uz bāzes staciju?

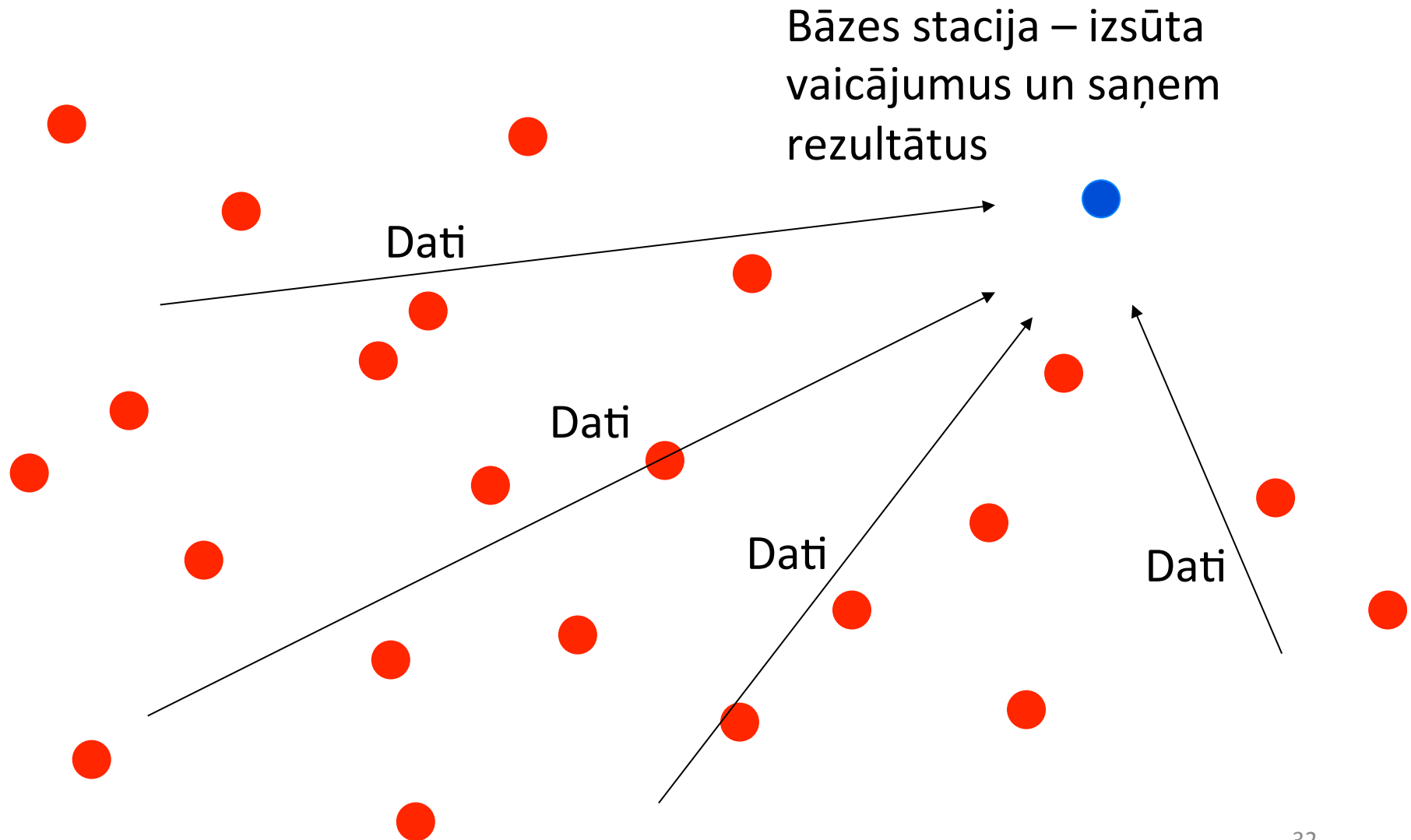


Sensoru tīkla uzdevums

Savākt informāciju:

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

Tipiska vaicājumu pieejas arhitektūra



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 piemēri

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

```
SELECT AVG(volume), room FROM sensors  
WHERE floor = 6  
GROUP BY room  
HAVING AVG(volume) > threshold  
LIFETIME 180 days
```

Datu uzkrāšana mezglos

CREATE

STORAGE POINT recentlight SIZE 8

AS (SELECT nodeid, light FROM sensors

SAMPLE PERIOD 10s)

SELECT COUNT(*)

FROM sensors AS s, recentLight AS rl

WHERE rl.nodeid = s.nodeid

AND s.light < rl.light

SAMPLE PERIOD 10s

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
- QM = Query Machine
 - Query [Engine] + [Virtual] Machine



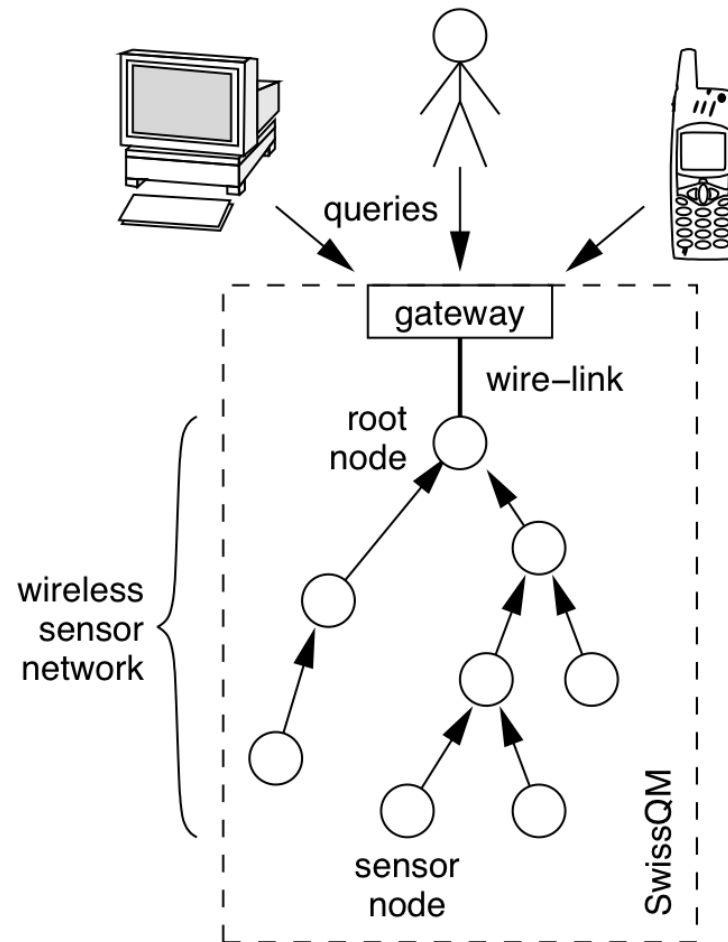
- R. Muller, G. Alonso, and D. Kossmann, “A virtual machine for sensor networks,” in Proceedings of the 2nd ACM SIGOPS/EuroSys European Conference on Computer Systems 2007, pp. 145–158, ACM, 2007.
- R. Muller, G. Alonso, and D. Kossmann, “SwissQM: Next Generation Data Processing in Sensor Networks,” in 3rd Biennial Conference on Innovative Data Systems Research (CIDR) January 7-10, 2007, Asilomar, CA, USA.

Mērķi / prasības

- Sensoru megli atdalīti no [ārējiem] vaicājumu interfeisiem
- Dinamiska, multi-user, multi-programmu BST izstrādes sistēma
- Motes dara tikai to, kas nepieciešams tiešo uzdevumu veikšanai
- Paplašināmība
 - izstrādātāju definētas funkcijas, aplikācijas
 - pašas SwissQM sistēmas paplašināmība

SwissQM pieeja

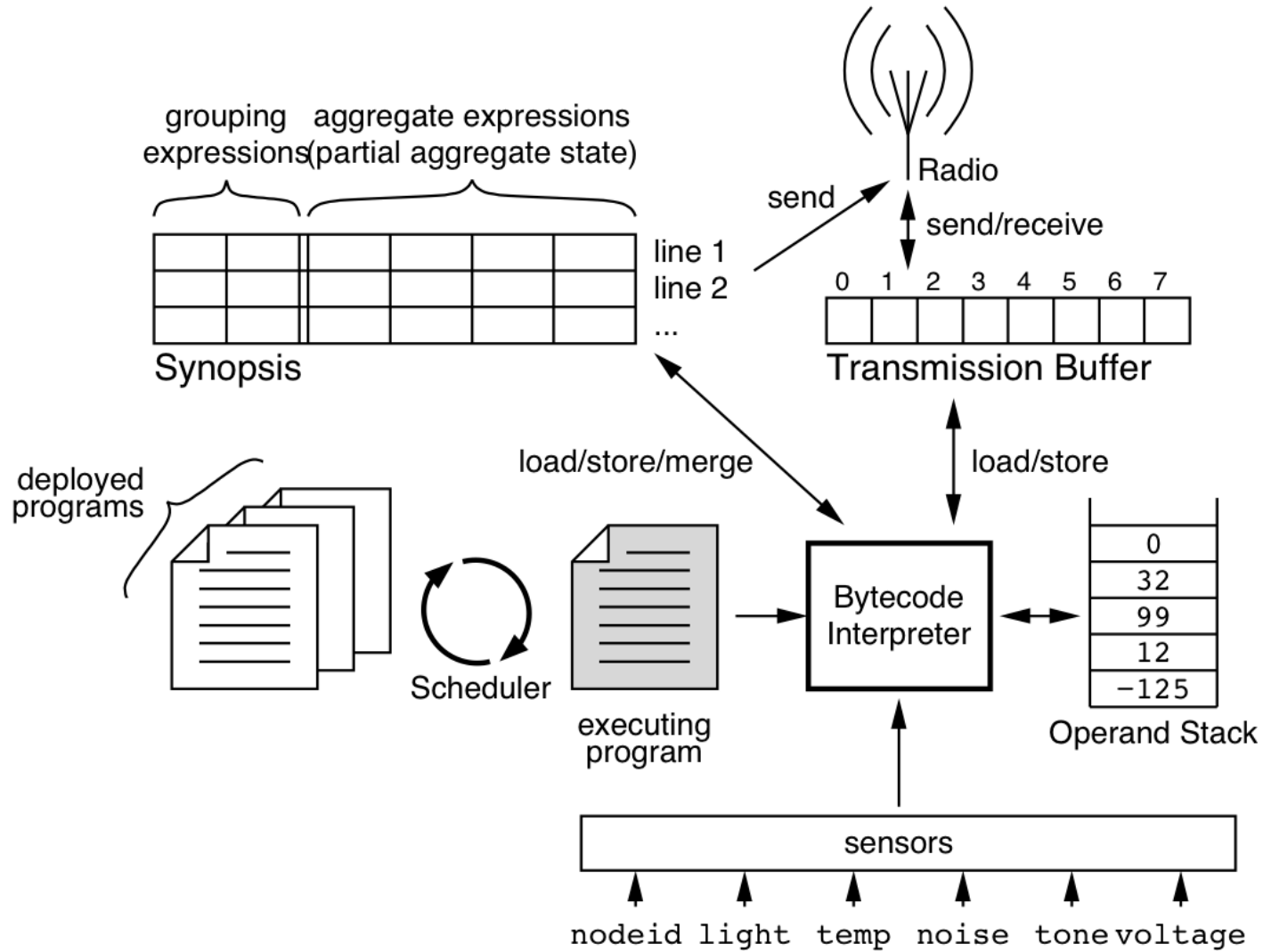
- Gateway kā "gudrais mezgls"
 - vaicājumu pārveidošana mezglu programmās
 - vaicājumu merging
 - multi-query optimization
- Paaugstina efektivitāti
 - motes izpilda tikai to, kas vajadzīgs (sense, aggregate, transmit)



Gateway

- 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

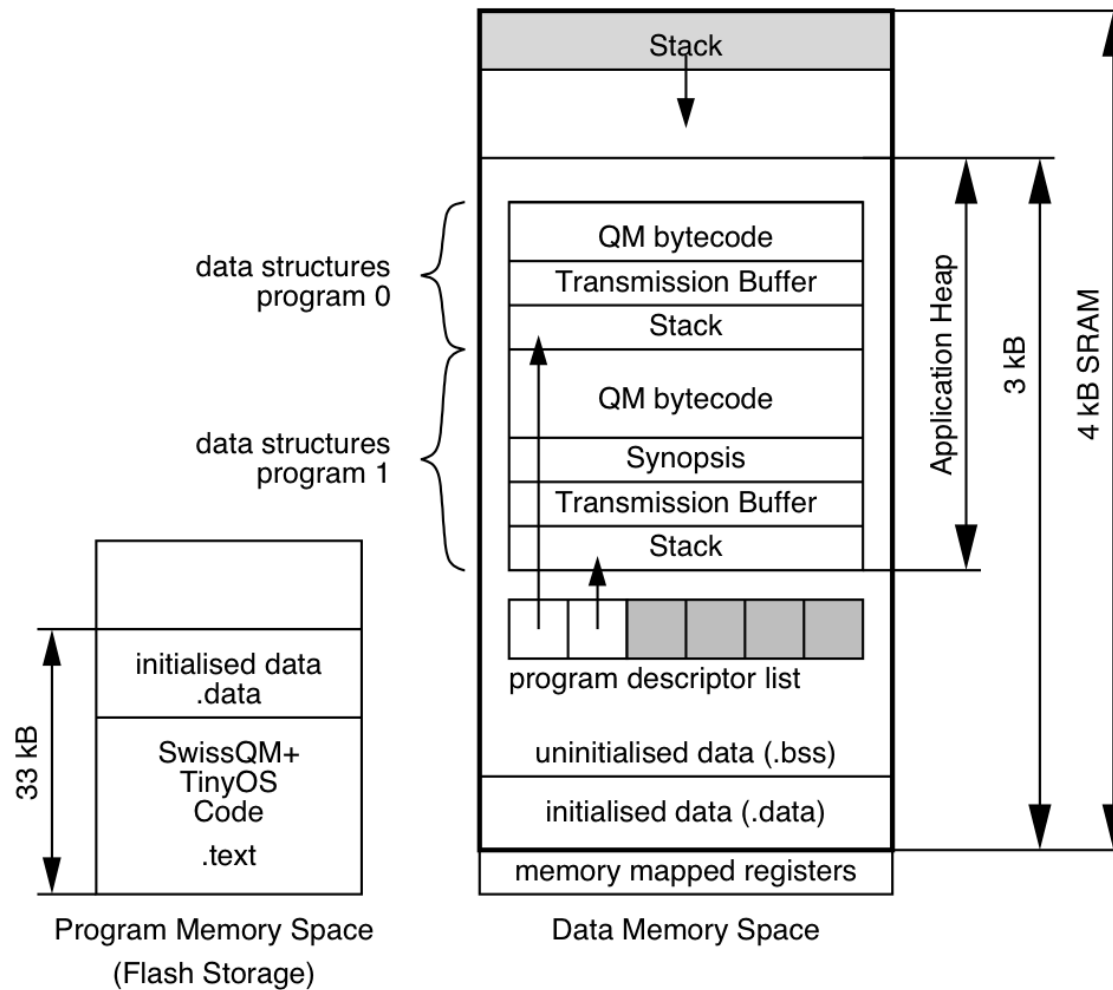
SwissQM arhitektūra



Katrai SwissQM aplikācijai

- Steks (operand stack, results): 16B
- Sūtījumu buferis (transmission buffer): 16B
- Kopsavilkuma buferis (synopsis): 16B
- Viens datu tips: 16bitu integer
- VM byte-kods, sadalīts 3 koda sekcijās:
 - Init
 - Delivery
 - Reception

SwissQM atmiņas izvietojums



SwissQM aplikāciju sekcijas

- **Init:** sākuma uzstādījumi
- **Delivery:** Sense & Send cikls, obligāts
- **Reception:** saņemto ziņu apstrāde (noklusētā izturēšanās – pārsūtīt tālāk)

SwissQM instrukciju kopa

- 59 instrukcijas, 1-3 baitus garas:
 - 37: Java VM analogi
 - Steka operācijas (DUP, SWAP, ...)
 - Artimētika (IADD, ...)
 - Programmas vadība (IF_ICMPLE, ...)
 - 22: darbs ar sensoriem, datu pārraidi un apstrādi:
 - Load / store data in transmission buffer, synopsis
 - Sensing
 - Transmission (SEND_TB, SEND_SY)
 - Aggregation (MERGE)
- Modulāra (var pielikt user-defined instrukcijas)

Sūtījumu buferis

- Ienākošā ziņojuma iegūšanai
- Izejošā ziņojuma būvei
- Pie katras sekcijas izpildes iztīrīts
- Nosūta tikai aizpildīto daļu
- Adresē pēc indeksa
- `istore <x>` un `iload <x>` instrukcijas

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>.


```

.section delivery, "@10s"
    get_temp          # read temperature sensor
    ipushb           60
    if_icmple        end # skip if temp ≤ 60
    get_nodeid       # read node's ID
    istore           0 # store it at pos. 0
    send_tb          # send transmission buffer
end:

.section reception
    send_tb          # forward tuple from child

```

izmērs: 10B, 1 ziņojums

Kopsavilkuma buferis

- Stāvokļa nodošanai starp izsaukumiem
- Agregācijai ar `merge` instrukciju
- Adresē pēc indeksa

- `istore_sy <x>` un `iload_sy <x>`
instrukcijas

SwissQM agregācija

Aggregate	State	Initialiser	Merger	Finaliser
COUNT	c	$c = 1$	$c = c_1 + c_2$	$= c$
MAX	m	$m = expr$	$m = \max(m_1, m_2)$	$= m$
MIN	m	$m = expr$	$m = \min(m_1, m_2)$	$= m$
SUM	s	$s = expr$	$s = s_1 + s_2$	$= s$
AVG	(s, c)	$(expr, 1)$	$(s_1 + s_2, c_1 + c_2)$	$= \frac{s}{c}$
VARIANCE	(s, t, c)	$(expr, expr^2, 1)$	$(s_1 + s_2, t_1 + t_2, c_1 + c_2)$	$= \frac{t}{c} - \frac{s^2}{c^2}$

- Agregācijas funkciju definē stāvoklis un 3 funkcijas (kā TinyDB)
- Finaliser (evaluatoru) rēķina Gateway

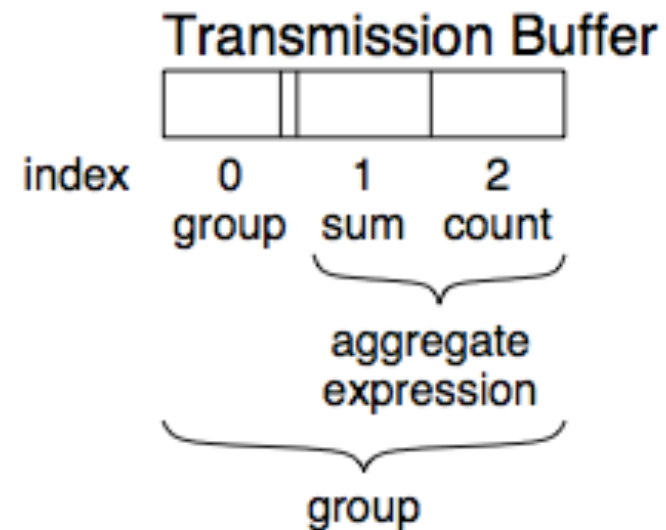
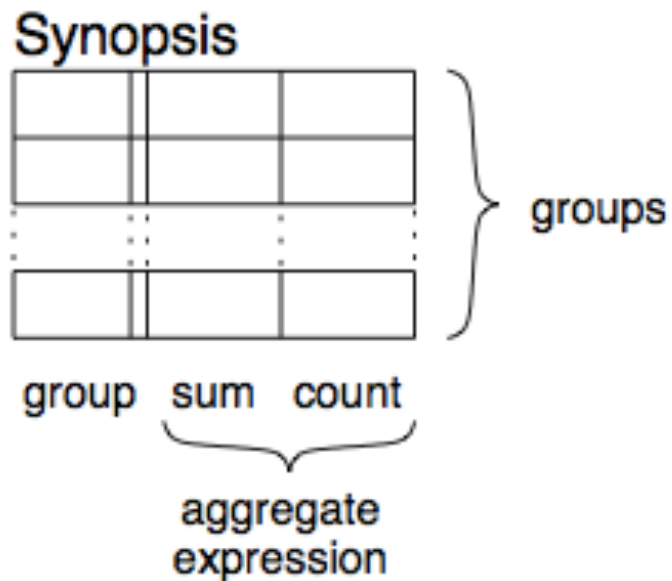
SwissQM merge instrukcija

- Operandi stekā norāda grupētāju un vērtību skaitu
- Datus (agregācijas stāvokli) ņem no kopsavilkuma bufera
 - piem., AVG stāvoklis = <summa, skaits>

merge(n, m, aggop_1, ..., aggop_m) operandi:

- n = grupēšanas izteiksmju skaits
- m = agregācijas izteiksmju skaits
- aggop_1..m = agregācijas operāciju ID

merge datu struktūras (piemērs)

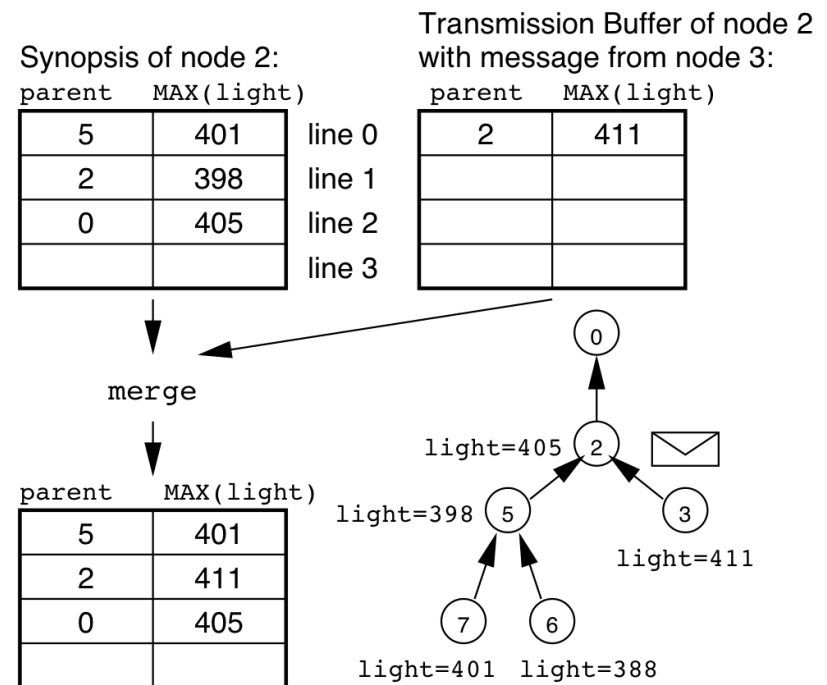


Agregācijas piemērs (1)

```
SELECT parent, MAX(light)
FROM sensors
GROUP BY parent
SAMPLE PERIOD 10s
```

(TinyDB SQL sintakse)

MAX(light) vērtība starp
mezgliem kam ir tas pats
vecāks



Agregācijas piemērs (2)

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

```

.section delivery, "@30s"
    get_light
    ipushw      512
    isub
    ipushb      10
    idiv
    istore      0      # store group expression
    get_temp
    istore      1      # sum := temp
    iconst_1
    istore      2      # count := 1
    ipushb      5      # agg: AVG = 5
    iconst_1    # number of agg expr: 1
    iconst_1    # number of grp expr: 1
    merge
    send_sy

.section reception
    ipushb      5      # agg: AVG = 5
    iconst_1    # number of agg expr: 1
    iconst_1    # number of grp expr: 1
    merge

```

27 bytes
2 messages

Ziņojumu maršrutizācija

- Tiek uzturēts mezglu koks
- Kopīgs duty cycle – bērni sūta pirms vecāku mezgla

Ko darīt, ja *synopsis* dati izaug pārāk lieli?

Ziņojumu maršrutizācija

- Tiek uzturēts mezglu koks
- Kopīgs duty cycle – bērni sūta pirms vecāku mezgla

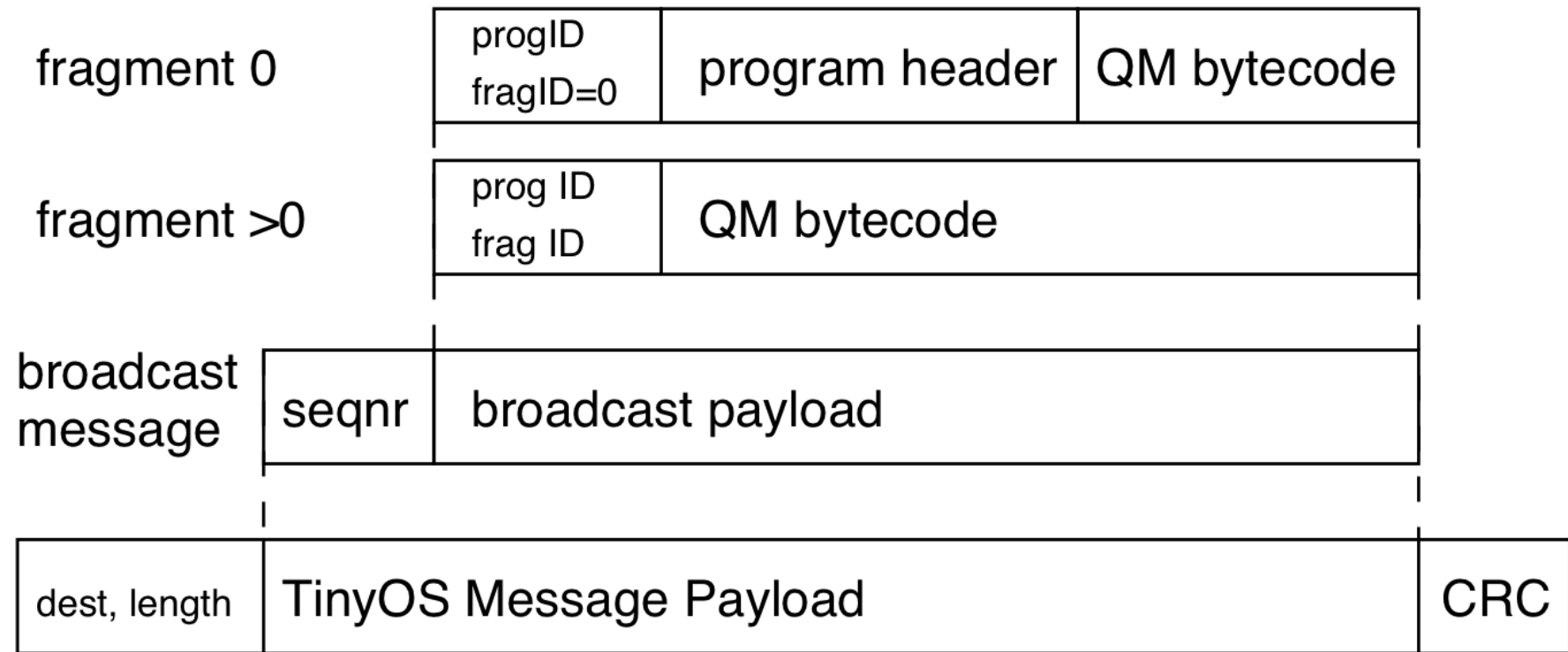
Ko darīt, ja *synopsis* dati izaug pārāk lieli?

- merge operācija pārtrauc agregāciju un forwardē visus saņemtos *synopsis* tālāk
- pārējo agregāciju veiks *gateway*

Pārprogrammēšana

- Gateway izsūta ziņas ar aplikācijas galveni un fragmentiem
- Tiklā tās izpludina, pārsūtot
- Ja kāds fragments iztrūkst, pārprasa vecāka mezglam

Pārprogrammēšanas ziņojumi



TinyOS Message

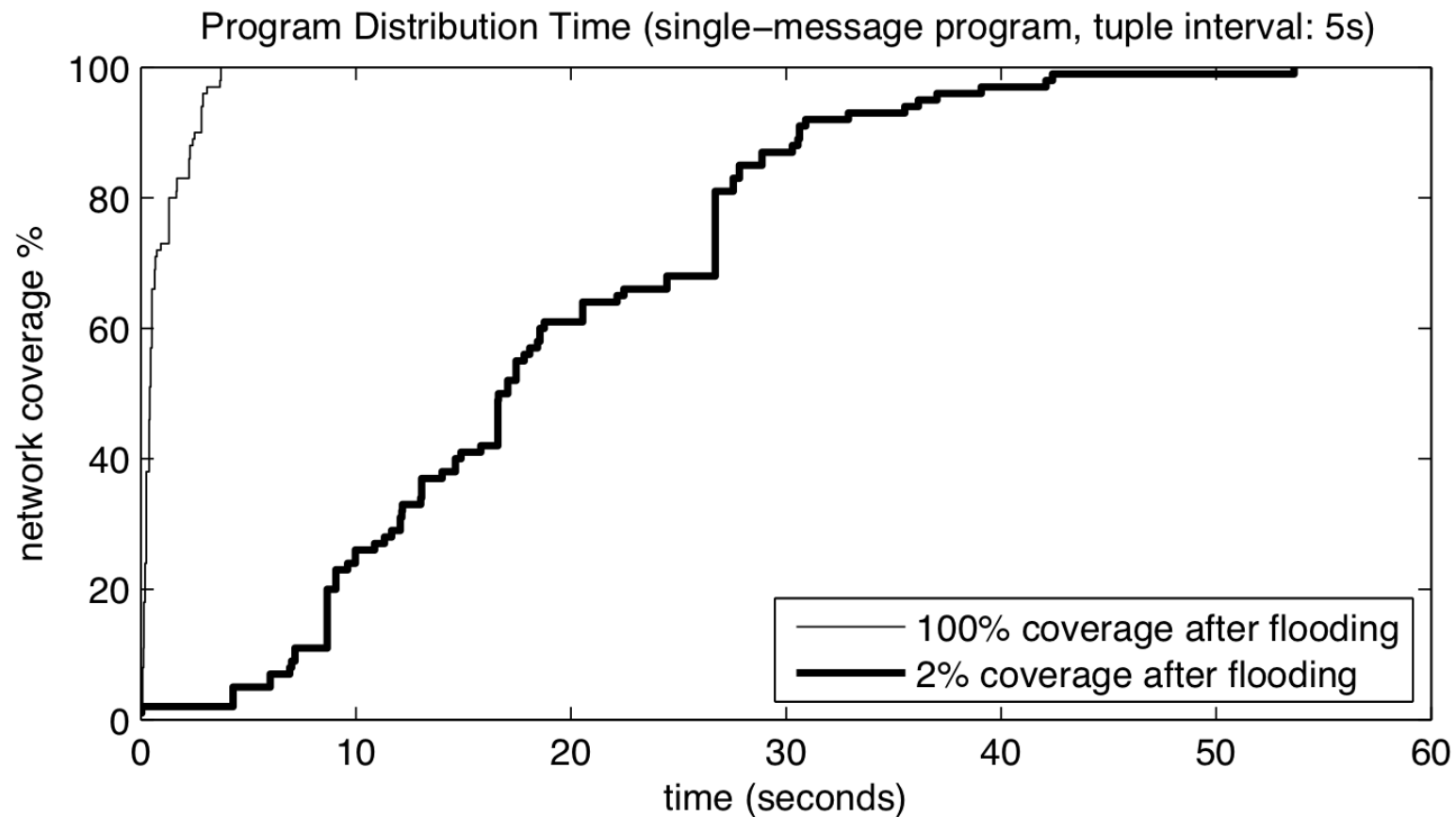
Program header

Programmas metadati (10 baiti):

- [programmas ID, fragmenta ID]
- init, delivery, reception sekciju garumi
- sampling period
- programmas “karodziņi” (flags):
 - vai tiek lietots kopsavilkums (*synopsis*)
 - synopsis dzēšanas režīms (manual vs epoch)
 - ...
- laiks, kad 1. reizi palaist delivery sekciju

Pārprogrammēšanas ātrums

Simulācija: 10x10 mezgli



SwissQM un TinyDB izmēri

Query	TinyDB		SwissQM	
	msgs	bytes	msgs	bytes
SELECT nodeid,light,temp FROM sensors	3	168	1	20
SELECT nodeid,light FROM sensors WHERE temp<512 AND nodeid>50	5	280	2	30
SELECT MAX(light) FROM sensors	2	112	1	22
SELECT parent,MAX(light) FROM sensors GROUP BY parent	3	168	1	25

SwissQM kopsavilkums

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

BST VM turpina dzīvot

A Virtual Machine for Wireless Sensor Networks

Pedro Emanuel Rodrigues Gomes



Departamento de Ciência de Computadores
Faculdade de Ciências da Universidade do Porto
2009

9. eseja

Kādu labumu Jūsu kursa projektam varētu sniegt virtuālās mašīnas izmantošana?

Termiņš: 02.12.2015. 10:00