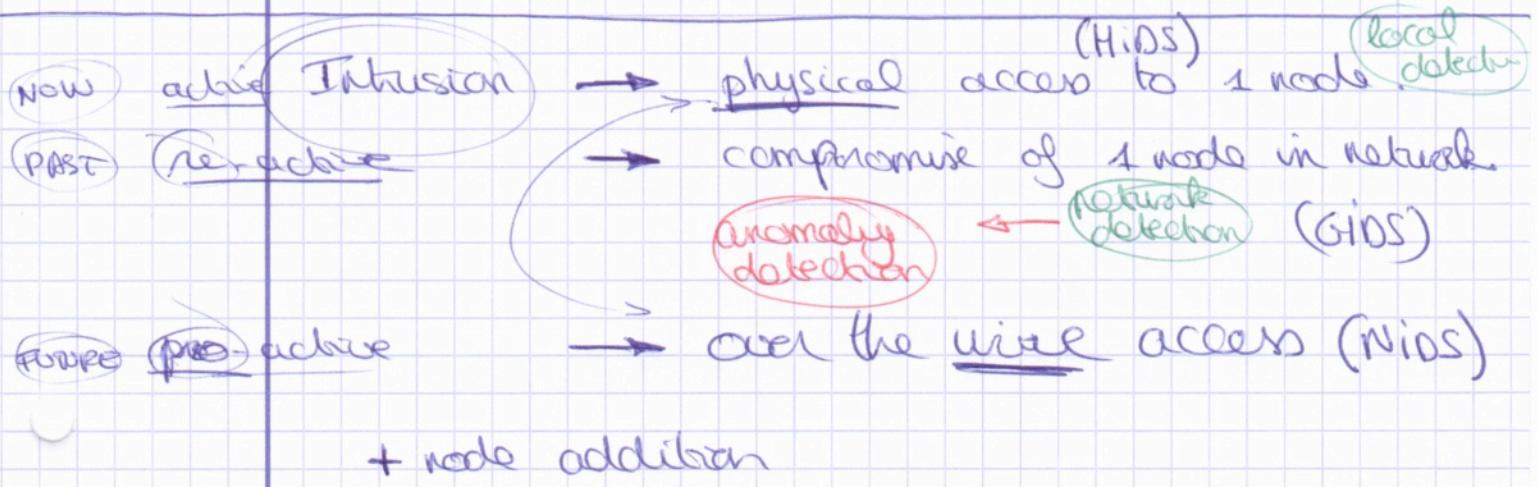
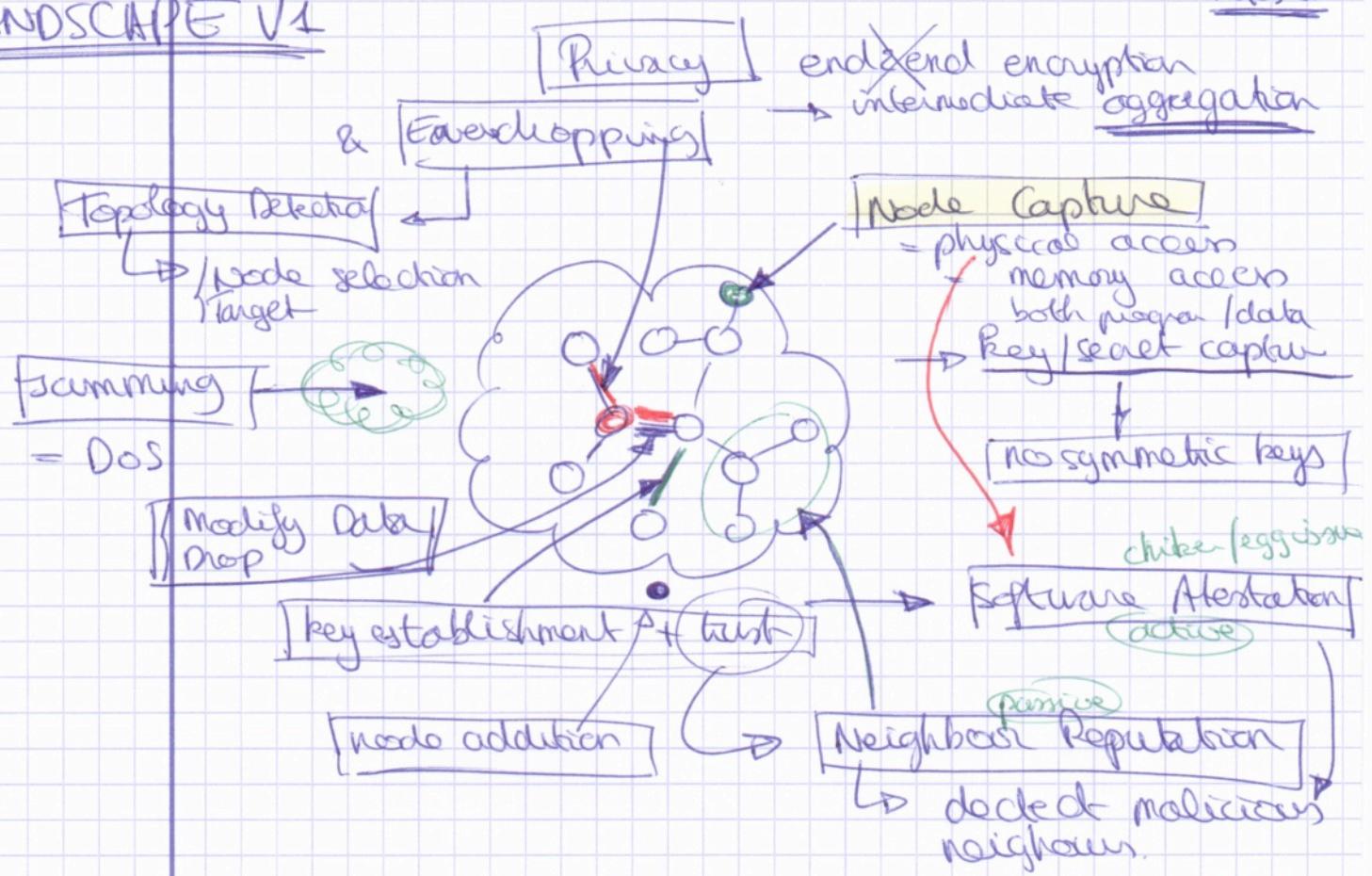
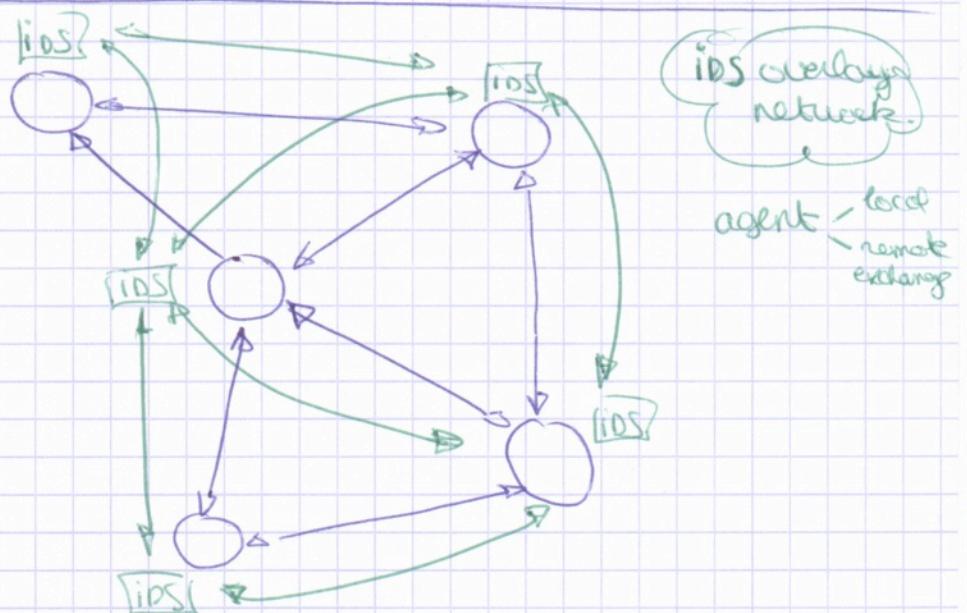


LANDSCAPE V1

notes 1



! intelligent agent
! host agents

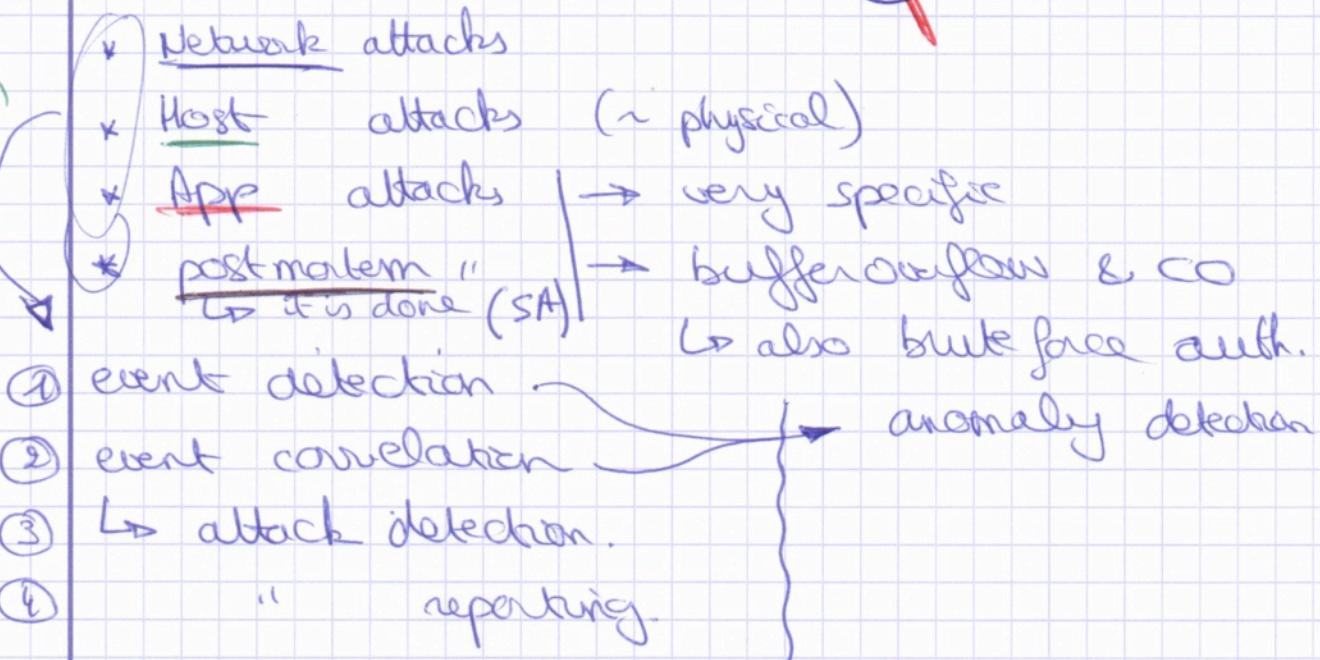


CONCEPT V1

Notes E

FN

~~Accept all kinds~~



Node → Network Group → Server

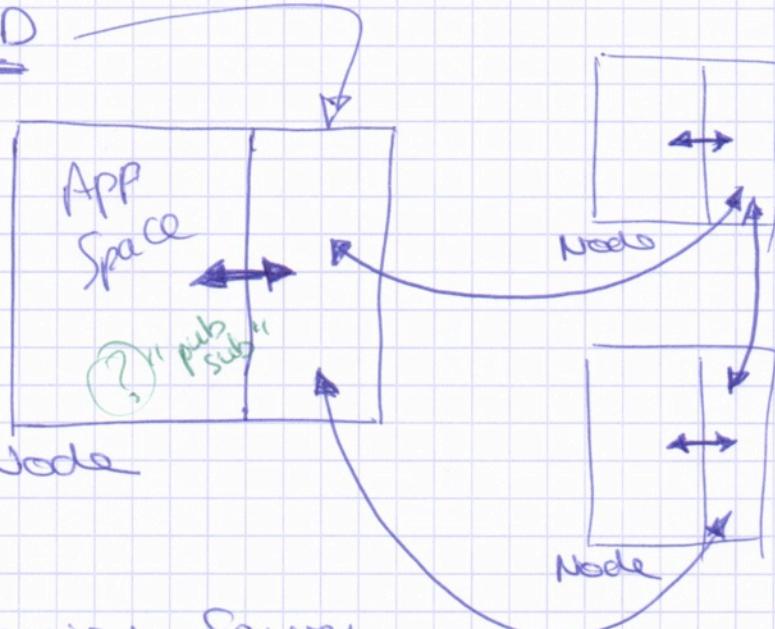
LoC-ID

Loosely Coupled
Intrusion Detection

⇒

≈ overlay networks

all communication
are events.



FIN
(small)!! ↪ rebirth
- comm
- auth
- pubsub
- correlate
- support all

+ Supervised Server

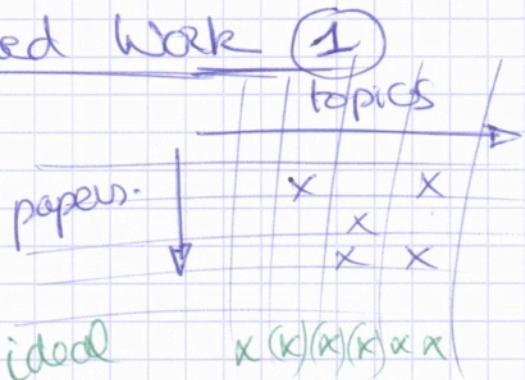
↳ origin of "rules"
policy

↳ pushes through network

→ dynamic / group specific

(?) ↳ real-time resp? ↪ to augment level

↳ divide network sensing
→ special nodes

Naam Related Work 1Thesis structuur

Voorwoord / Dankwoord,

Samenvatting

Inleiding.

- Draadloze sensornetwerken
- Toepassingen ← VB !!
- Probleemstelling
- Doelstelling
- Verloop / structuur tekst

Achtergrond.

- ... → landscape, "nodes", network, contiki, (cosi)
- Generated onderzoek
 - = } major slices

Probleemstelling ← Scenario's

Bestaande Technologieën

2

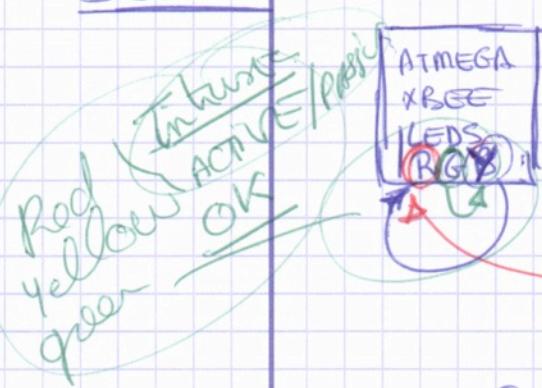
Architectuur

Implementatie

Discussie

Beduidt

1
Vaste technologieën
of platform

Demo

+ topology + supervisor

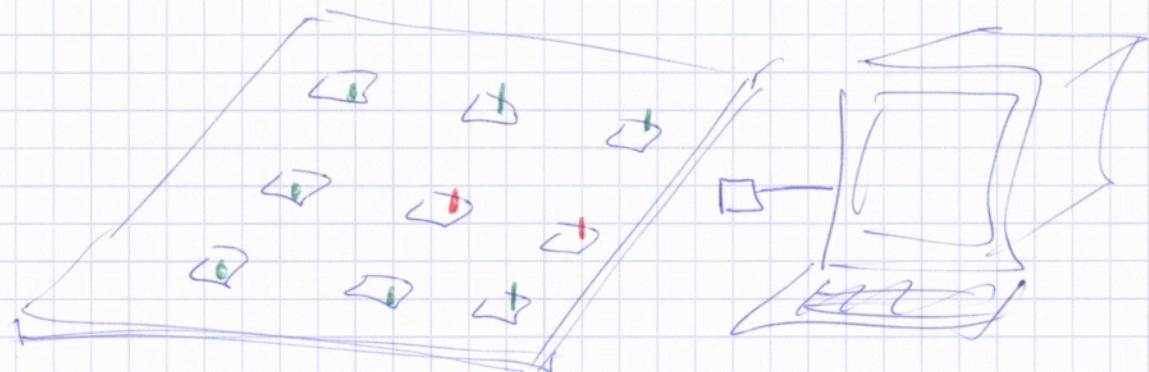
"simulate" 3 attackers $\leq \frac{1}{3}$ 3 types

met implementatie return node app?

mind app?

→ with external interface
→ proximity? → motion detector

↳ case "try not to be seen"



Matrix related work

	topic	item						
✓ support for detection in accu								
✓ takes physical								
✓ papers	X	X			X			
	X	.	.		X	-		

Features

- end 2 end solution
- light weight
- non-intrusive (pub sub, event, ...)

Notes 5

report

node - GW - sever
group

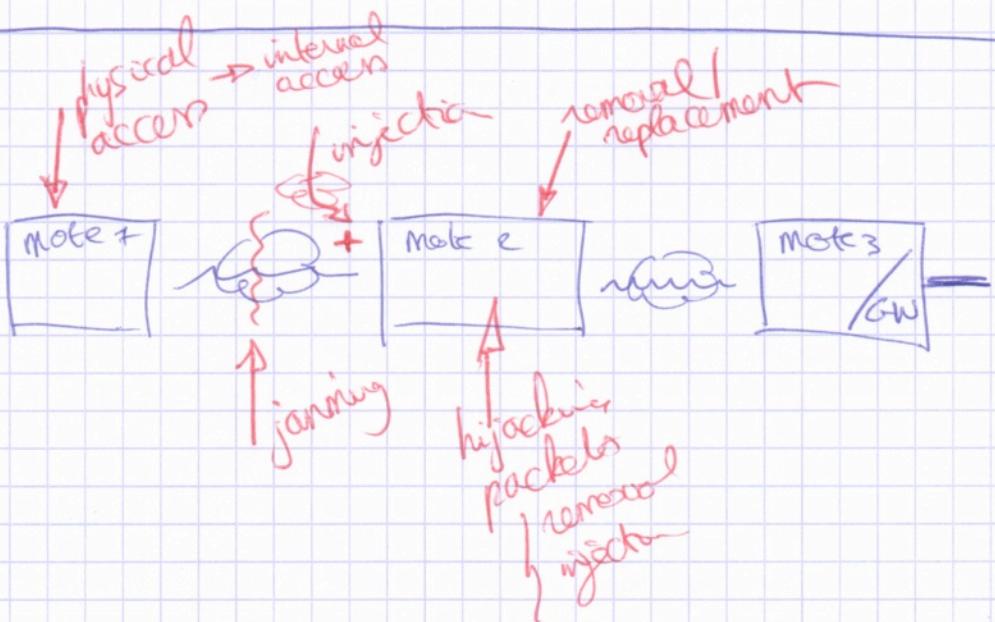
configure/install

Reasons

- not in use - not in scope
↳ no offering

↳ Detection while "sleeping" ! ...

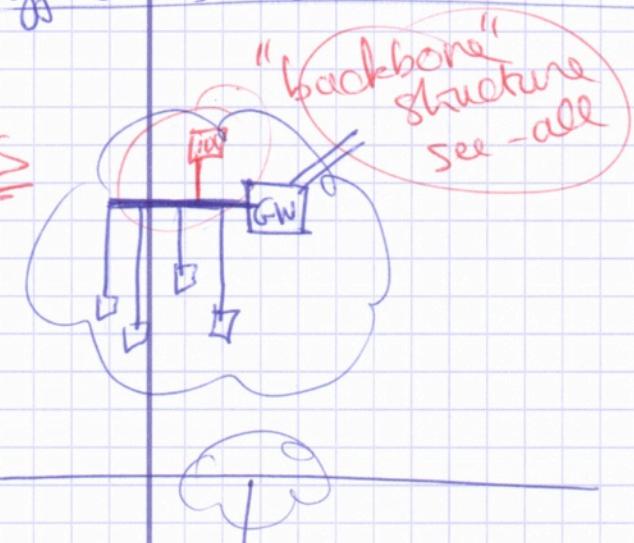
thesis NOT about detection = too large specific related work



Differences "classic" IDS vs NSN-IDS

Notes 6

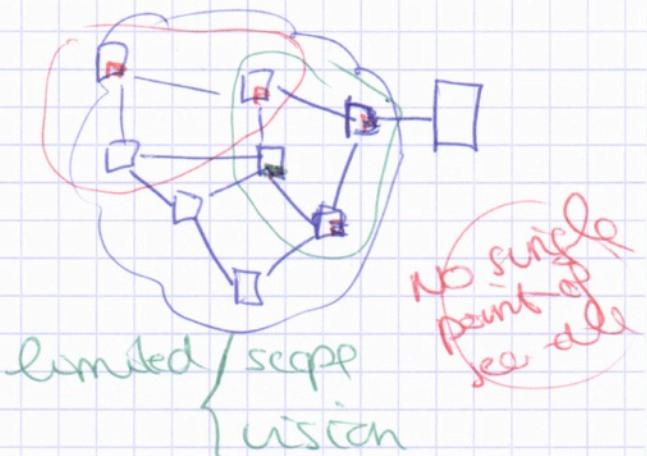
NIDS



HIDS



VS

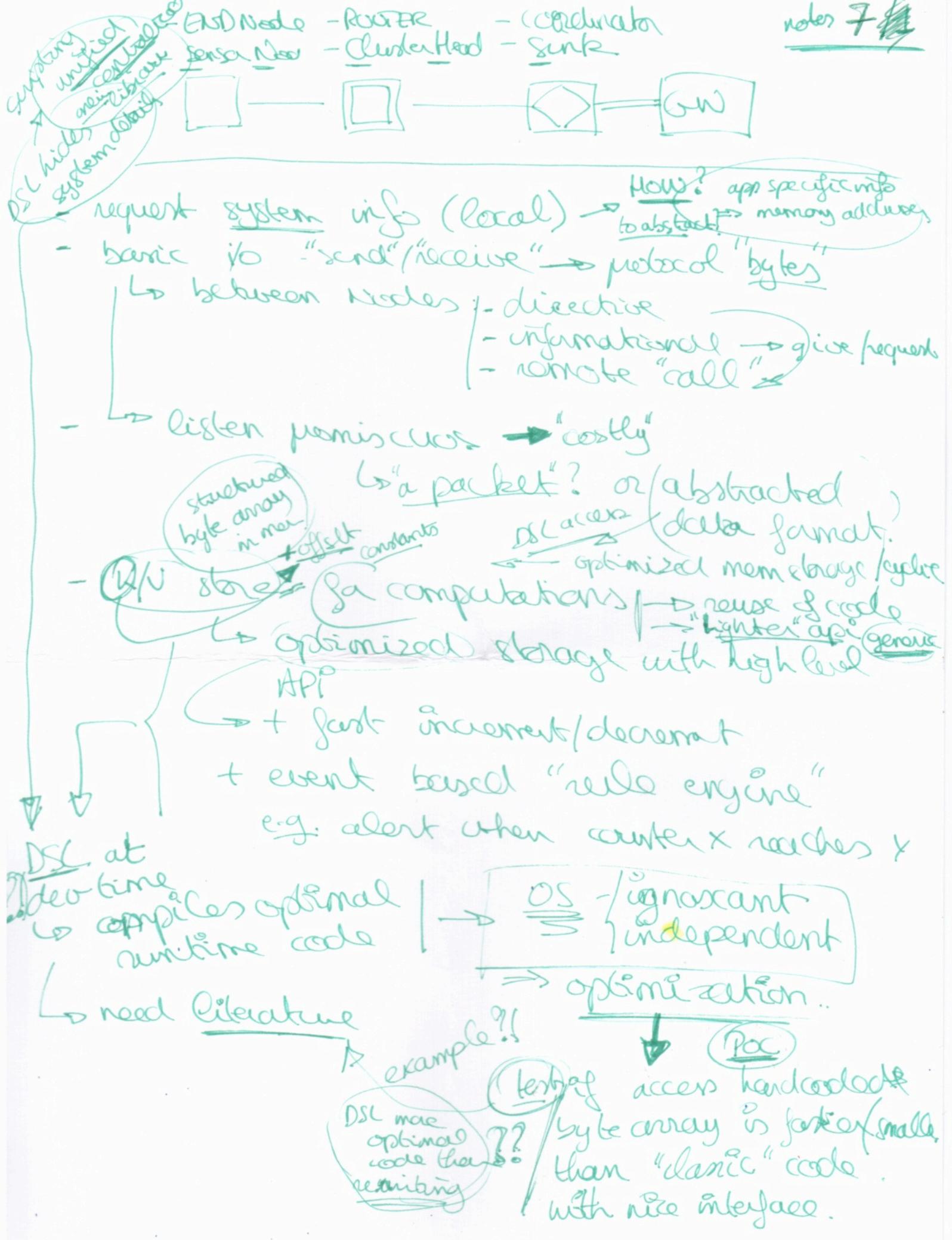


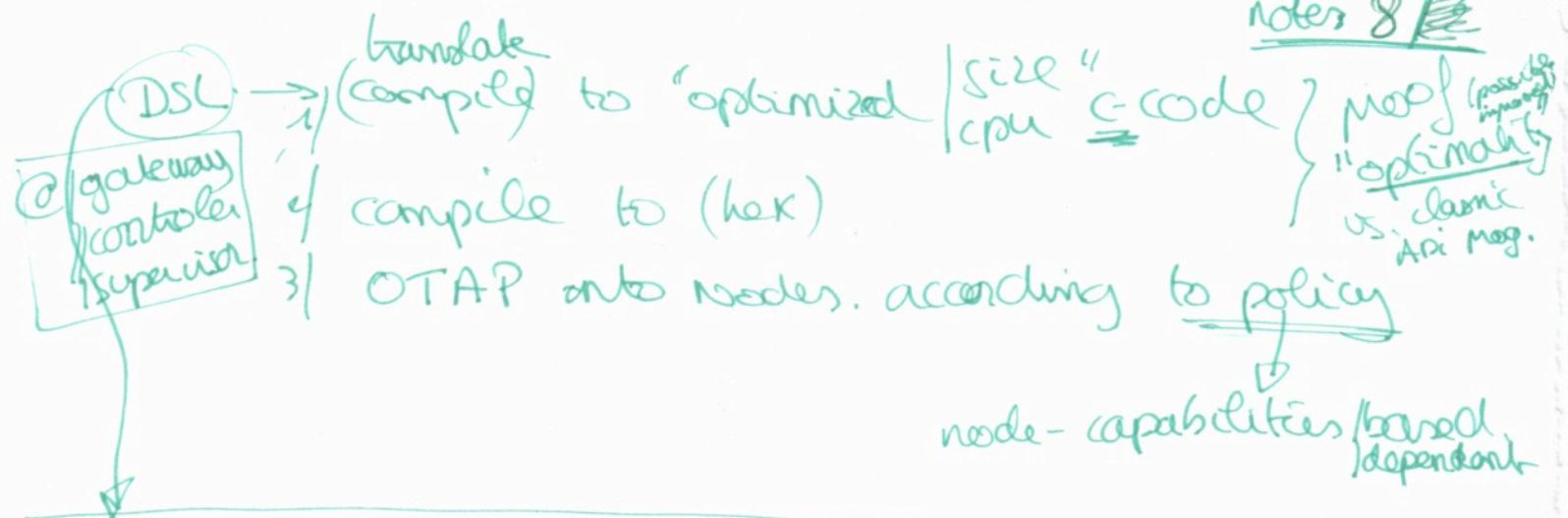
→ no single instance possible

signatures
large list of application specific events

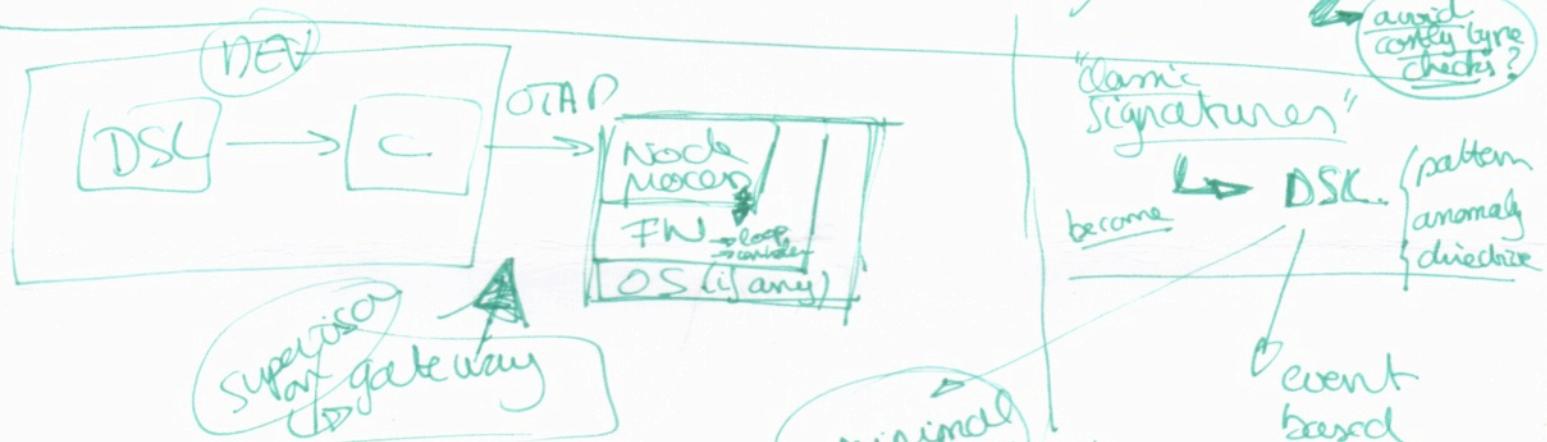
operating system "calls"
application specific

→ ~~files focus~~
~~change~~





optimize multi rules (in classic code this would be a
no-go \Rightarrow imperative language..).



~~DSL~~ (Cisco)
internet
deutsch = ISDN SL
specification
language
:-)

minimal
≤ 10 cond stackups

no loops

1) context ++

2) when context > 10

return
from local

condition
assignment (operator)
event trigger definition

set/get info
local system
remote
complaint

node {shouldn't know about others} → "might not be there"
↳ unintended messaging
↳ inhibited ↳ implicit

~~C~~ check if possible to implement algo's that do access specific other nodes' info.

- DSC
- Code generation
- FW + API → very low-level technical.
- Contiki (+ LooCiD)
- ↳ "or non"

how
possibly
justify

LooCiD FW

Loosely Coupled
Intrusion Detection FW

① event driven? DSL

~~too many~~

(too many)
no DSC
→ just most that
is possible

thesis "scope"

subset possibilities

↳ (just enough) to move overall
possibilities

based on
"detection"

literature

- 1) detection WSN
- 2) detection classic.
- 3) DSL → code generation

Demo

/ shows classic implementation

X 2-3 detections

versus (generated) low-level API + FW

shows / working

+ code / mem / CPU reduction

attach selection

→ "visibility" ① "Sybil" ?

⇒

② "sink hole" ?

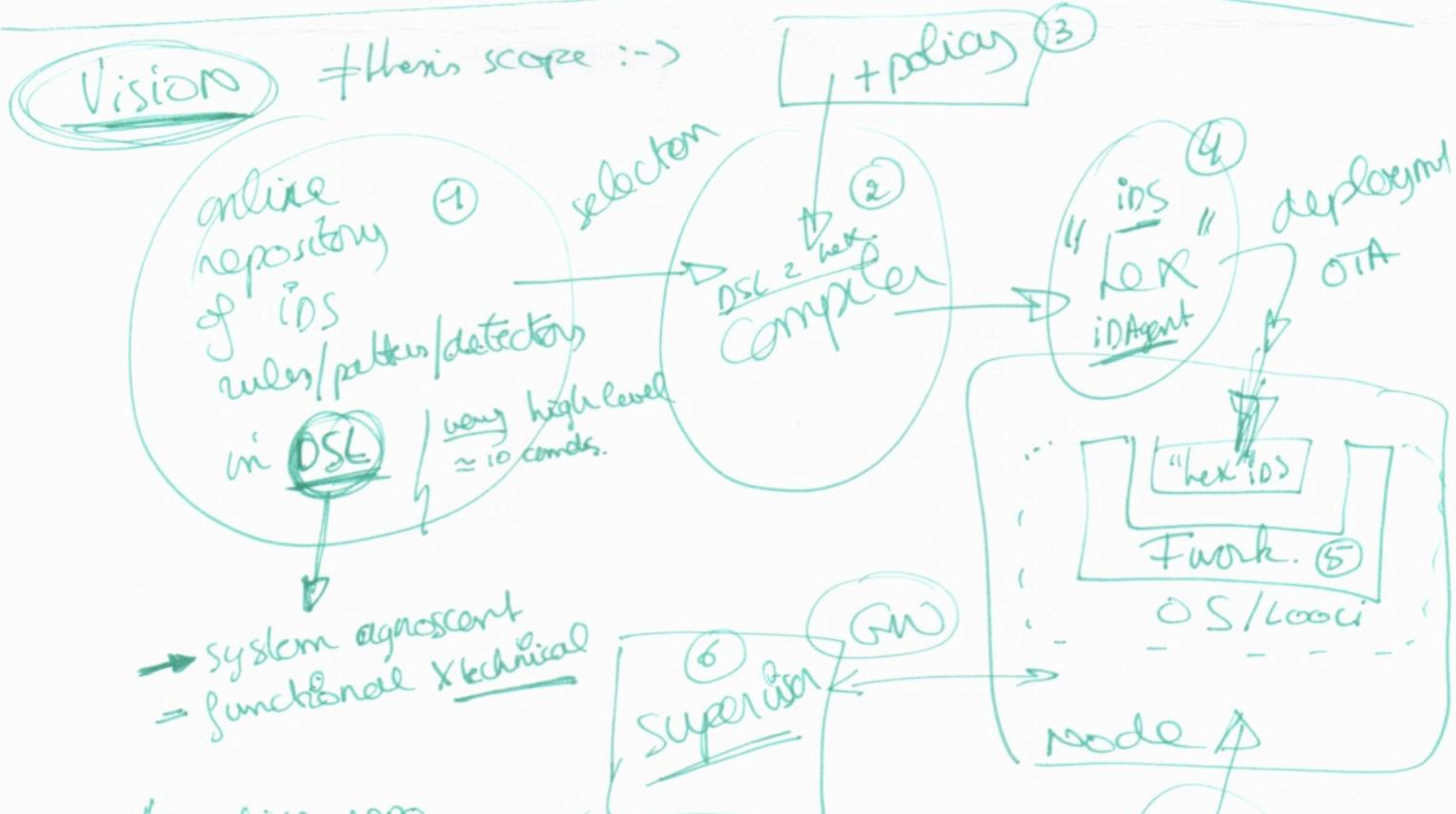
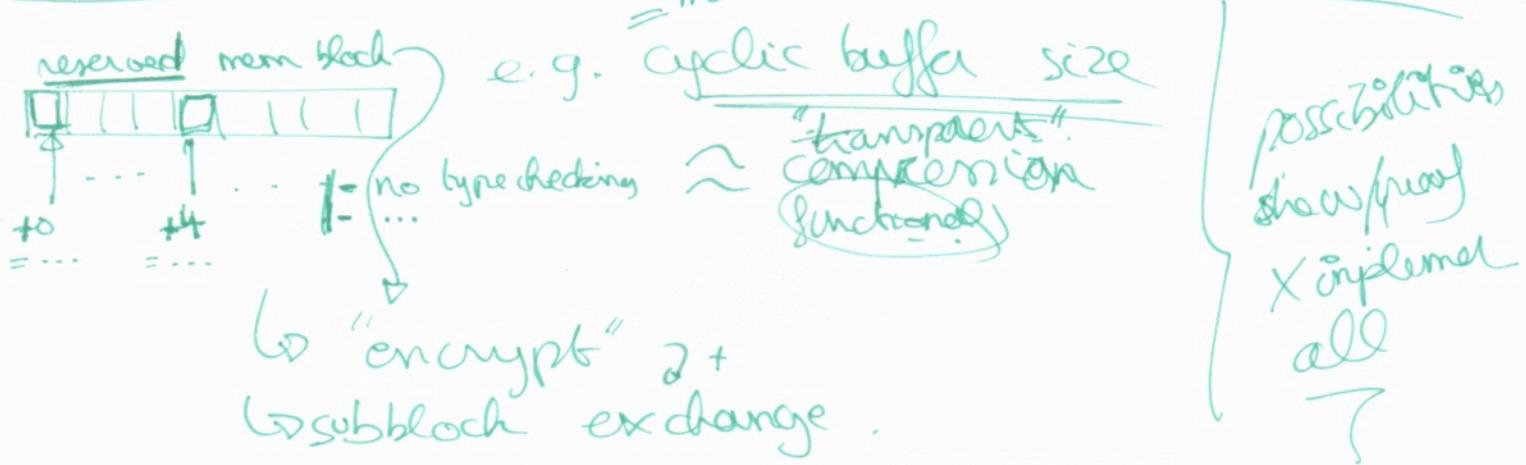
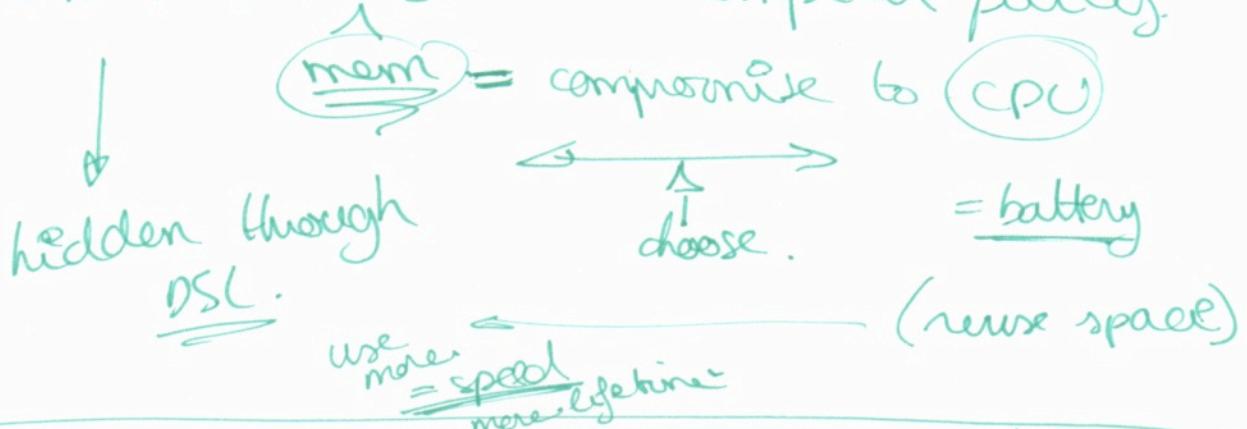
la mama

③ "node capture" ?

anomalie

notes 10 ~~10~~

! limit max size = compiler policy.



1. online repos
 2. DSL compiler
 3. policy δ
 4. low level API implementation.
 5. supported API framework
 6. supervision tools

+ Scope of Hardware → ATMELA1284P + Contiki + LoRa

① RFSN

POC-1

Notes 11

$$T_{ij} = E(R_{ij})$$



$$R_{ij} = (R_{ij})_D + (R_{ij})_{iD}$$

$$\overline{(R_{ij})_D} = f(D_{ij}, (R_{ij})_0)$$

$$(R_{ij})_{ID} = (R_{ij})_{IB} + w_{ik} * \overset{||}{g(R_{ik})}$$

$\Rightarrow \underline{\text{BRSN}}$

$$R_{ij} = \frac{P(D_{ij} | R_{ij}) * R_{ij}}{\sum P(D_{ij} | R_{ij}) * R_{ij}}$$

$$R_{ij} = \text{Beta}(\alpha_i + 1, \beta_i + 1)$$

Gamma

$$\Gamma(x) = (x-1)!$$

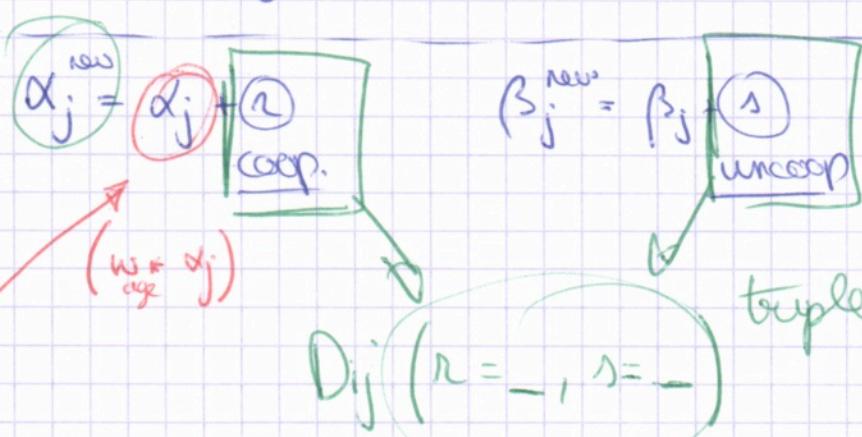
Beta dist

$$P(x) = f(x, \alpha, \beta) = \frac{\Gamma(\alpha + \beta)}{\Gamma(\alpha)\Gamma(\beta)} x^{\alpha-1} (1-x)^{\beta-1}$$

$$X \sim \text{Beta}(\alpha, \beta)$$

$$T_{ij} = E(R_{ij}) = E(\text{Beta}(\alpha_{j+1}, \beta_{j+1}))$$

$$= \frac{\alpha_j + 1}{\alpha_j + \beta_j + 2}$$

UpdateAging2nd hand info

! independent Rep Infra.
e.g. $(R_{ij})_D$

$$\alpha_j^{\text{new}} = \alpha_j + \frac{\beta_j^{\text{new}}}{\beta_j}$$

$$\alpha_j^{\text{new}} = \frac{2\alpha_k}{(\beta_k + 2)(\alpha_j^k + \beta_j^k + 2) + 2\alpha_k}$$

repulation of k
weight

$(R_{ij})_D \rightarrow RTD_i$
refresh (=0)
after broadcast

propagation
cooperative
 RTD_i TH

indirect
observation
through node
about j

only propagate nodes in RTD_i^c and RTD_j^c

Simulation (1,0) (0,1) $\xrightarrow{\text{OK NOT}}$ Forwarding notes 13

accuracy = 0,98

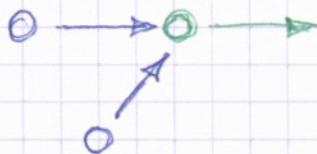
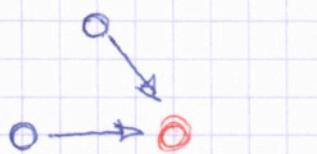
$$TH_{\text{SHI}} = 0,9$$

Needs info on packets send by other nodes

→ on <packet> <trigger-cb> (+ promiscuous mode)

data storage for other nodes

→ variables, dynamic $\sim \#$ neighbour nodes
math functions ↳ max?



POC 2 Cooperative iIDS (paper 2)

Model

→ proofs for iDP + implementation for real.

↳ if impossible \Rightarrow not realistic.
in theory. $\begin{cases} \text{possible in reality} \\ = \text{weaker} \end{cases}$

$$S = \{s_1, s_2, \dots, s_n\} \quad \text{sensors.}$$

$N(s)$

Neighbours \Leftarrow static + symm.
 \hookrightarrow 2-hop.

t nodes attacked \Rightarrow Byzantine failure.

↳ Byz. Agreement protocol.

$|t=1| \rightarrow t > 1$ is hard \Rightarrow

source(s) $\Leftrightarrow s = \text{attacker/captured node}$

honest(s) $\Leftrightarrow \neg \text{source}(s)$

Alert Module $\xrightarrow{\text{alerted node}} D(s) \xrightarrow{\text{suspected sensors}} \text{detected set}$

$|D(s)| = 1 \rightarrow \text{attacker identified}$

honest node

$\exists s' \in D(s) : \text{source}(s')$

$s' = \text{attack} \rightarrow \delta \text{ delay } \exists s \in D(s) : \text{source}(s)$

$\forall s \in S : \text{honest} \Rightarrow D(s) \subseteq N(s)$

iDP

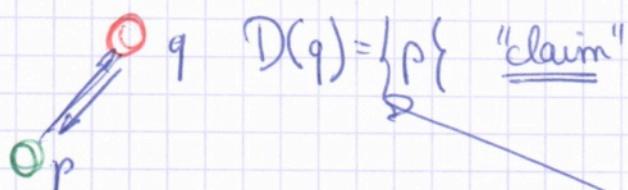
$\forall s, s' \in S : \text{honest}(s) \wedge \text{expose}_{s'}(s')$

correctness

$\Rightarrow A(s) \wedge \text{source}(s')$

termination

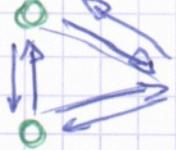
attack \Rightarrow after T all honest nodes have attacked in exposed.



$D(p) = \{q\} \rightarrow$ knows it's honest \Rightarrow discard

$\Rightarrow \text{expose}_p(q)$

$$D(2) = \{p, q\}$$



notes 15

$$D(q) = \{p, r\}$$

p

$$D(p) = \{r, q\}$$

not solvable

necessary

(what is) sufficient

$$AN(s) = \{t \mid A(t) \wedge t \in N(s)\}$$

$$\tilde{AN}(p, q) = AN(p) \setminus \{q\} \quad (= \text{valuable to } q)$$

iBC

(pos)

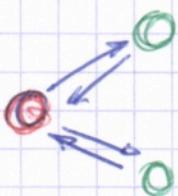
(not necessary)

NC

and

NC₁ all neighbours of attacker are alerted

NC₂ & or more suspects by majority
⇒ honest suspect nodes have non-alerted neighbours



Byzantine

att

$$n > 3t + 1$$

$$t+1 \quad n > 4$$

Algorithm

1) init phase a) preload one way key chain length l

(K_0, \dots, K_{l-1}) unique per node

short short short

b) discover all neighbours
c) 2-hop neighbour hood TTL=2
L-table

d) announce K_0 to 1-hop

- 1) honest/alerted
neighbours of attacker
share views
- 2) agree on some id.
- 3) expose it

2) voting phase

+ timer

$$M_0(s) = D(s)$$

$$\text{MAC}_{K_j}(M_0(s))$$

3) publish key phase

newest committed key notes 16

$$R_{j-1} = \text{share}(k_j)$$

if check is work

else check authority voter

4) exposing the attacker phase

5) external ring reinforcement phase

$$\text{NC holds } P = \{ p_1, \dots, p_k \}$$

non-alerted neighbours of alerted node

→ request by alerted region with P

→ reply with "favor" for $e \in P$

honest nodes have majority
⇒ non alerted neighbours

Needs

[storage keys (\rightarrow flash)]

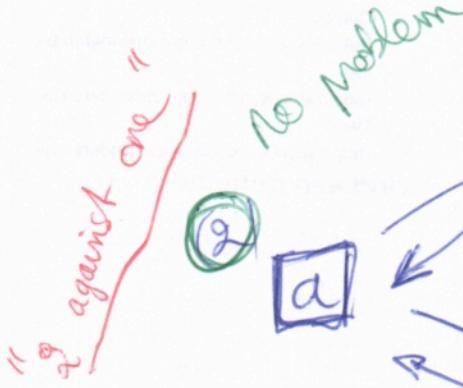
storage key / neighbour

) send packet with TIL
respond to ↑

iDC calculation

timer callback.

SMA1 function



(1)
b

a

(1)
c

IDC

Notes 17

$$\begin{aligned}\sim \bar{AN}(a, b) &= \{c\} = \sim \bar{AN}(b, a) = \{c\} \\ \sim \bar{AN}(c, b) &= \{a\} = \sim \bar{AN}(b, c) = \{a\}\end{aligned}$$

$$\begin{aligned}\sim \bar{AN}(a, c) &= \{b\} = \sim \bar{AN}(c, a) = \{b\} \\ \sim \bar{AN}(b, c) &= \{a\} = \sim \bar{AN}(c, b) = \{a\}\end{aligned}$$

$$D(a) = AN(a) = \{b, c\}$$

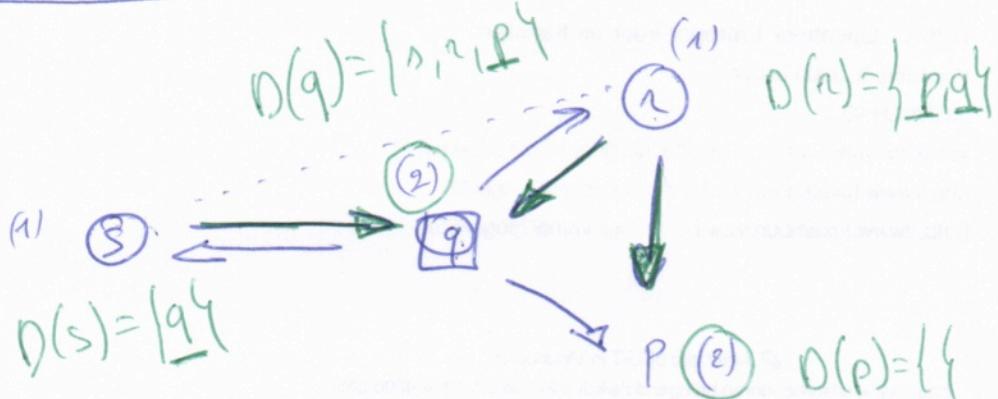
$$D(b) = \{a\} \quad \bar{AN}(b) = \{\textcircled{2} c\}$$

$$D(c) = \{a\} \quad \bar{AN}(c) = \{\textcircled{2} b\}$$

NC

NC₁ ok

NC₂ holds want geen 2 of meer
 \Rightarrow



$$AN(s) = \{q\}$$

$$AN(q) = \{s, r, \cancel{p}\}$$

$$AN(r) = \{q, \cancel{p}\}$$

$$AN(p) = \{q, r\}$$

$$\begin{aligned}\sim \bar{AN}(q, r) &= \{s\} \\ \sim \bar{AN}(r, q) &= \{\cancel{s}\}\end{aligned}$$

OK

ioc holds.
 \Rightarrow p of r is available

$$\begin{aligned}\sim \bar{AN}(p, r) &= \{\cancel{s}\} \\ \sim \bar{AN}(r, p) &= \{q\}\end{aligned}$$

OK

Stel 1 en 2 neighbours

$$\Rightarrow AN(s) = \{r, q\}$$

$$AN(r) = \{s, q, \cancel{p}\}$$

$$AN(q) = \{s, r, \cancel{p}\}$$

$$\bar{AN}(s, q) = \{\cancel{p}\}$$

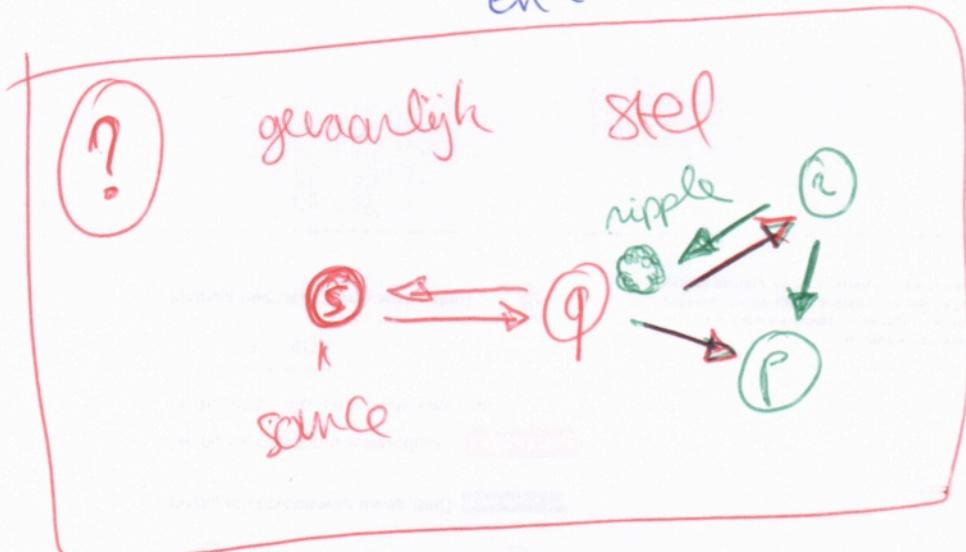
$$\sim \bar{AN}(q, s) = \{\cancel{p}\}$$

een van de twee is al beschikbaar
x IDC

~~Not 1: $\text{source}(a) \wedge \text{agent}(a)$~~ $\Rightarrow A(a)$

~~Not 2: $\exists s_1 s_2 \dots$~~

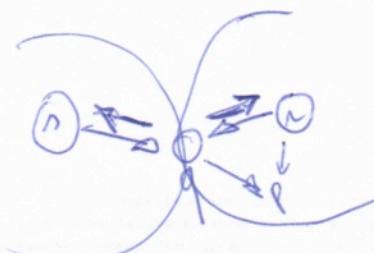
$$\begin{aligned}
 D(s) &= \{q\} \\
 D(q) &= \{s, r, p\} \quad \text{not neighbour} \quad \begin{array}{l} \text{stel } s \text{ honest} \\ \Rightarrow \text{source } \in D(s) \end{array} \quad \begin{array}{l} \text{se} \\ D(s) \cap D(a) \\ = \{q\} \end{array} \\
 D(r) &= \{q, p\} \quad \begin{array}{l} \text{stel } q \text{ honest} \\ \Rightarrow \dots \in D(a) = \{p\} \end{array} \\
 &\quad \text{intersection.} \quad \text{en } r \text{ neighbour.} \\
 &\quad \text{en } t = 1.
 \end{aligned}$$



$$\begin{aligned}
 AN(s) &= \{q\} \\
 AN(r) &= \{q\}
 \end{aligned}$$

stel $s = \text{attach}$

$$\begin{aligned}
 &\cancel{s \in D(r)} \\
 &q \in D(s) \\
 &\quad \in D(r)
 \end{aligned}$$



Software Attestation

① short comings.

base station performs SA

- * Tamper-resistant HW
COST!!
- * challenge-response

① Rootkit based on ROP Hash

② code comprehension.

Adversary + situation

get full control

2) during SA : not full

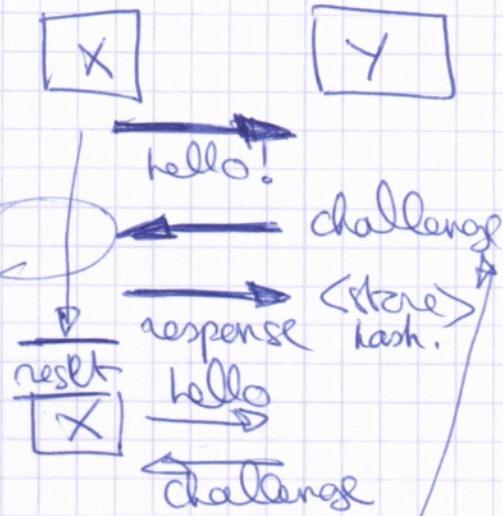
3) injection + "buffer overflow" (softwar vulnerability)

4) no network interaction

5) X HW change

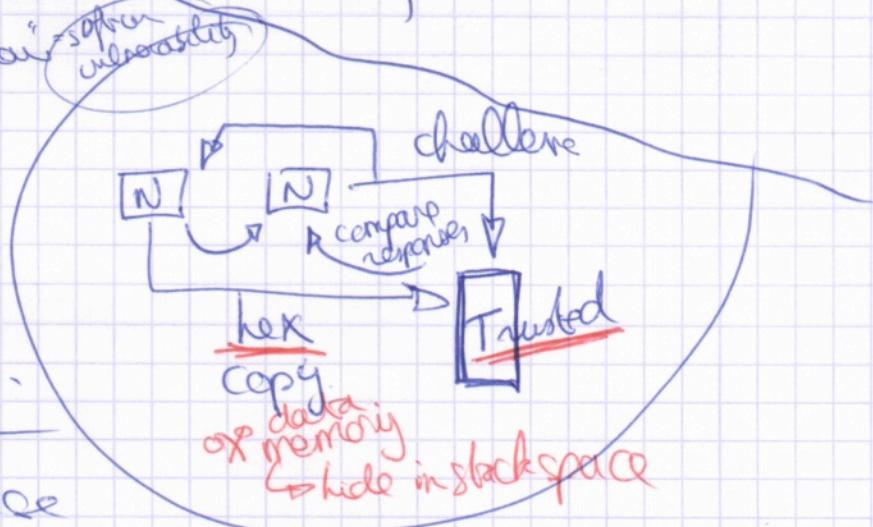
6) verifier knows HWS + config.

stel :-)



go

- 1) extract challenge
- 2) pre compute



challenge / response

$H(\cdot)$ + nonce

pseudo-random \rightarrow selection of mem words.
to seed from verifier

SWATT
software
Attestation

→ timing
redirection
↳ delays

limited to program memory
not data / external .

rootkit

Fill empty program memory

↳ compression
only program memory

~~Self-modifying code based att.~~

→ difficult.

for us

ICE

indisputable code execution

→ SAKF

↳ program counter needed → ↗ RIP

Attestation on attestation

modified on different mem. location.

Rootkit

- modify attestation code

not timing

- jump to hide function

complete needed

- ROP to "replace"

bootloader

TinyOS

bootloader

Compression Attack

room for

memory shadowing

compress + on-the-fly decompression
program mem.

Time-based \leftrightarrow SWATT

↳ avoid using redirection
= introduces delay



e.g. alteration 23 cycles
+ redirection 3 cycles =

13%

can be noticed

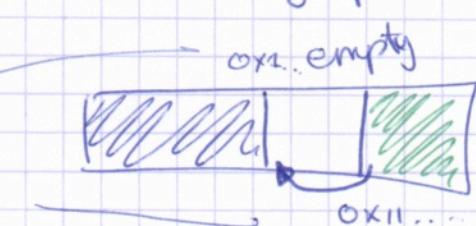
1) empty mem redirecting using bitflip

0xFF

0x1xxxx...x
0x11xx...x

= 2 instructions

= 7,4% = 43% faster than



Data memory alteration

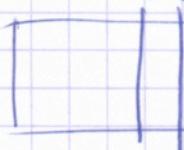
→ data memory is unpredictable



ice

displacement \Rightarrow MSB of PC.

malicious ice



0x1100

0001

original ice

altered program.

0x9100

10001...

ICE

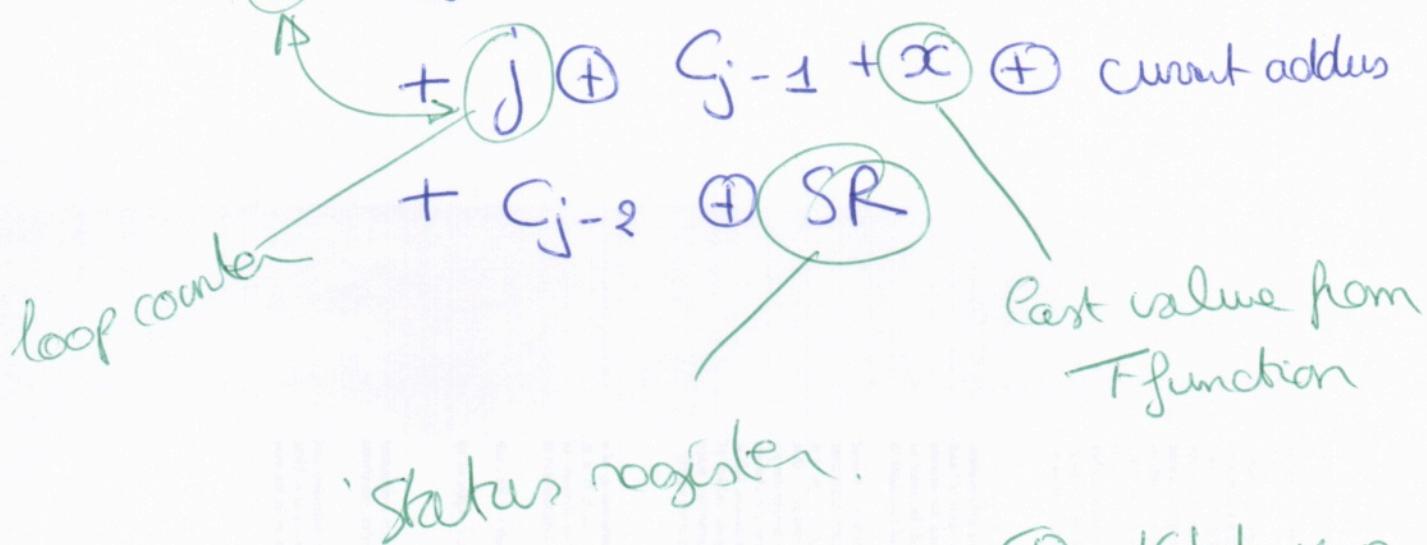
T function \rightarrow random permutations of mem. locations.

Mem loc \rightarrow 160 bit checksum C of 10 16bit registers, C_j

$$C = [C_0, \dots, C_9]$$

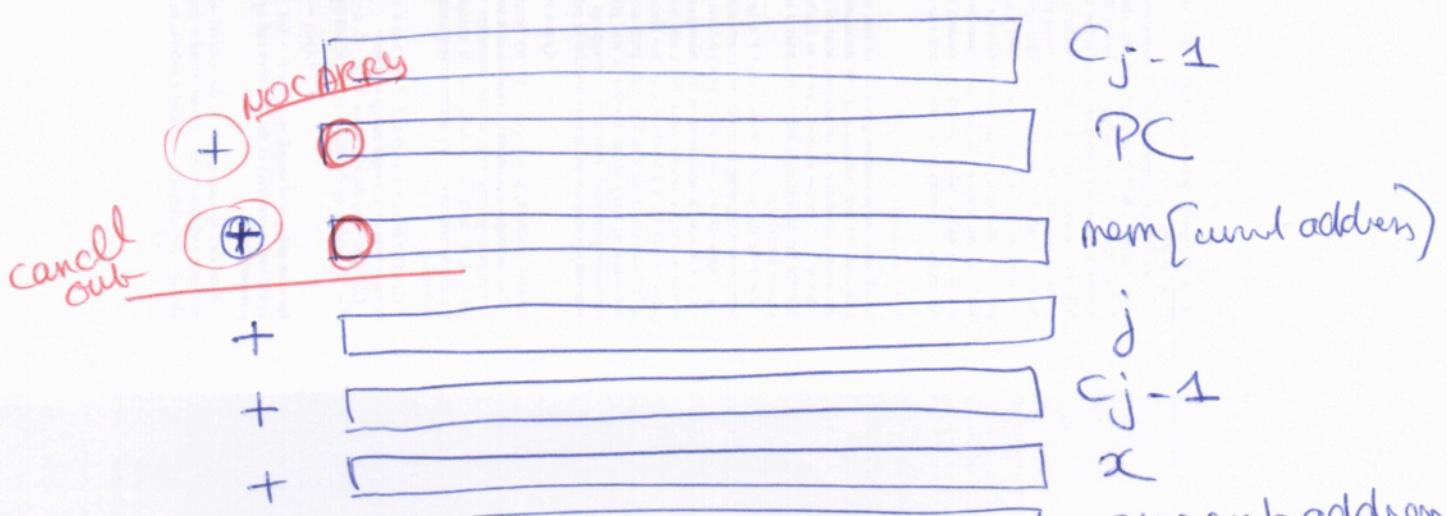
program counter.

$$C_j = C_{j-1} + PC \oplus \text{mem[current_address]}$$



\oplus = 16 bit XOR

$+$ = 16 bit SUM (no carry)



aschenbrenner
Categories

in/out-sider attack
mote/laptop class

bleib

Goals
(of attack)

- manipulate data
- eavesdrop (EE)
- ↳ ROM/RAM dump
- drop data.
- network access.
- Integrity
- Confidentiality
- Availability

threat analysis

Methods

- rushing
- DOS

Situation

- resource constraint (energy)
- physical access (node capture)
- in-network process (no end-to-end encryption)

valores



Attacks

prepare

- * jamming
- * Sinkhole → false ETX count → attack data and dead
- * wormhole → fast redundant connection
cross network = replay attack
- * Sybil
- * selective forwarding

Goals

eavesdropping
traffic Analysis

✓

Disruption

Hijacking

Physical attacks

(blibat)

WSN Properties

- 1) Aggregation → Tree structured routing
- 2) Tolerable failure
- 3) In network processing
- 4)
- 5) Sensor = router
- 6) Phased transmission periods.

Security goals

auth., access control, confidentiality, integrity, authentication, anonymity, non-repudiation, freshness, availability; resilience

Security in WSN

~~perig~~

High level

1) Secure group mgmt

2) iOS

in case of failure

3) Secure data aggregation

Fundamental issues

1) node capture

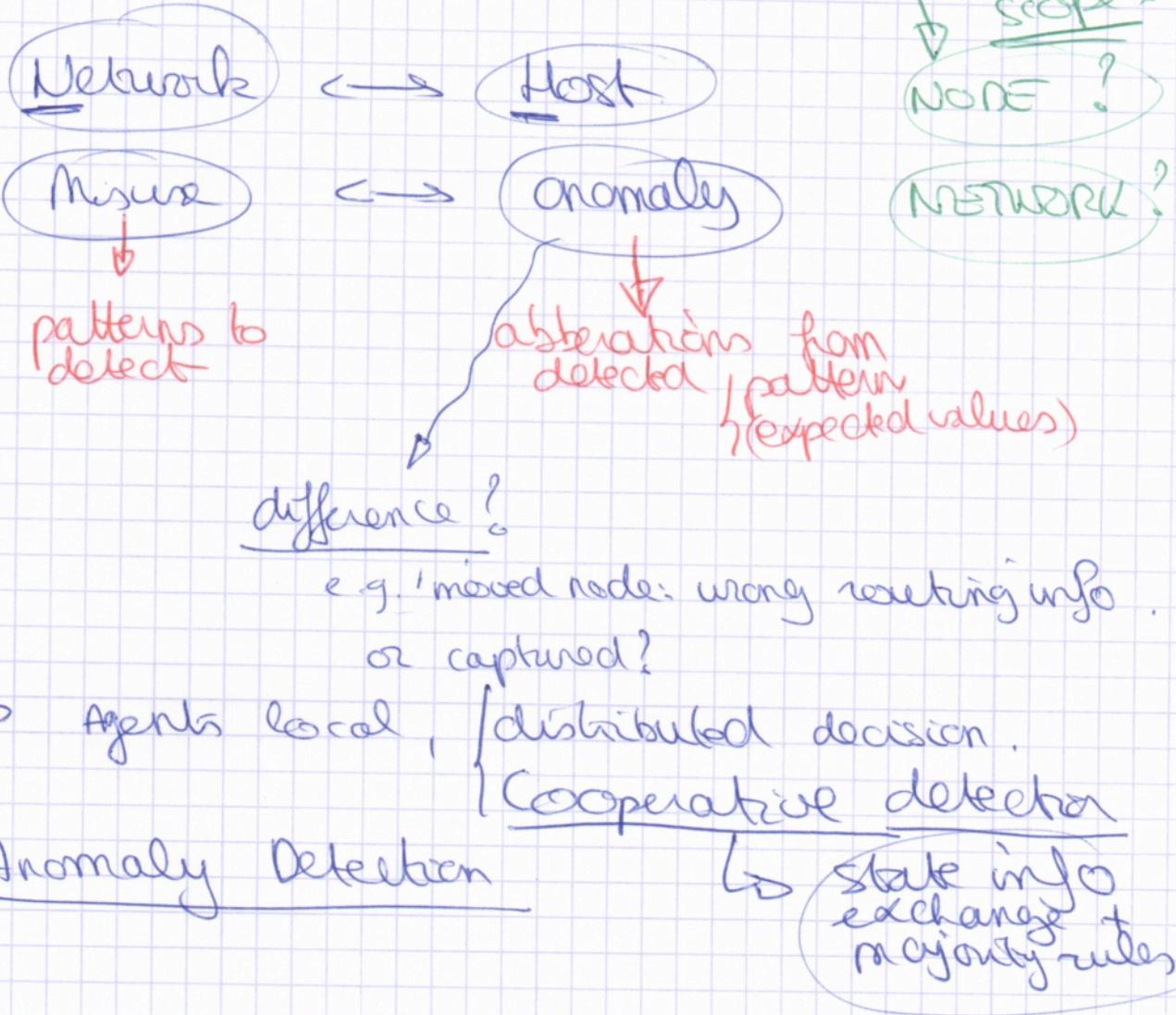
2) physical tampering

3) DDoS

IDS

= "Capturing audit data and reasoning about the evidence in the data to determine whether the system is under attack".

2 hang 2009
intranet



Anomaly Detection

→ definition in conclusion

- IDS arch should be
- distributed / cooperative
- statistical anomaly
- cross layer NW

aschenbuch

heartbeat → node removal

↳ interrupted working

anomaly → requires "central"↳ distributed all nodes
| neighbours.movement → HW : e.g. | accelerometer,
| PIRanomalyevent

Carrier Sense Time

~~OKR~~ ~~CST~~ ^{dt.:} →

→ jamming detection.

agency

$$t_n = (1-\alpha) t_0 + \alpha s$$

anomaly thresholdalarm if $s > t_0 + w$ dd sample
std dev. overall
samplesOTAP

: reprogram

| protocol + ECC

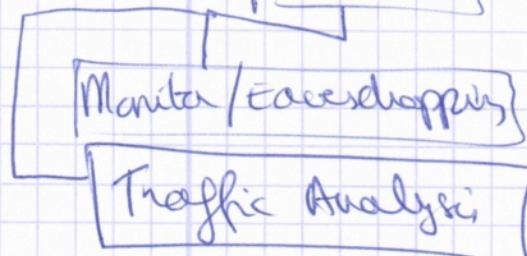
Tiny ECC

padmarathi 2008

Attack

notes 27

Passive \leftrightarrow Privacy



Active

Routing

GOAL
EFFECT

DOS

x attack
x attack

- 1) selective forwarder, greyhole
- 2) sinkhole blackhole
- 3) sybil.
- 4) wormhole
- 5) hello flood

1) jamming
2) tampering

phy

3) collision
4) exhaustiveness
5) unfairness

link
MAC

6) neglect

7) greed

8) homing

Network

9) misdirection

10) blackholes

11) flooding { kommt

12) desync.

Node

Message

- 1) capture
- 2) malfunction
- 3) outage
- 4) physical
- 5) replication.

1) corruption/alteration

information gathering

notes 28

Traffic analysis

sniffing, jamming

Pассив

monitor eavesdrop

Актив

= change

no physical

physical access

= node capture / move / remove

reprogram node

OTAP

Soft vulnerability

access node memory

or

"hard" buffer overflow + code to send.

Normal physical access
e.g. RS232

JTAG, ...

insert node

physical logical

capture network key info

network access

introduce hardware

sybil = not in itself, but base for actual attacks
= ability to create nodes

wormhole

→ no actual participation, just mirror

sinkhole

participate

a) laptop

b)

c)

d)

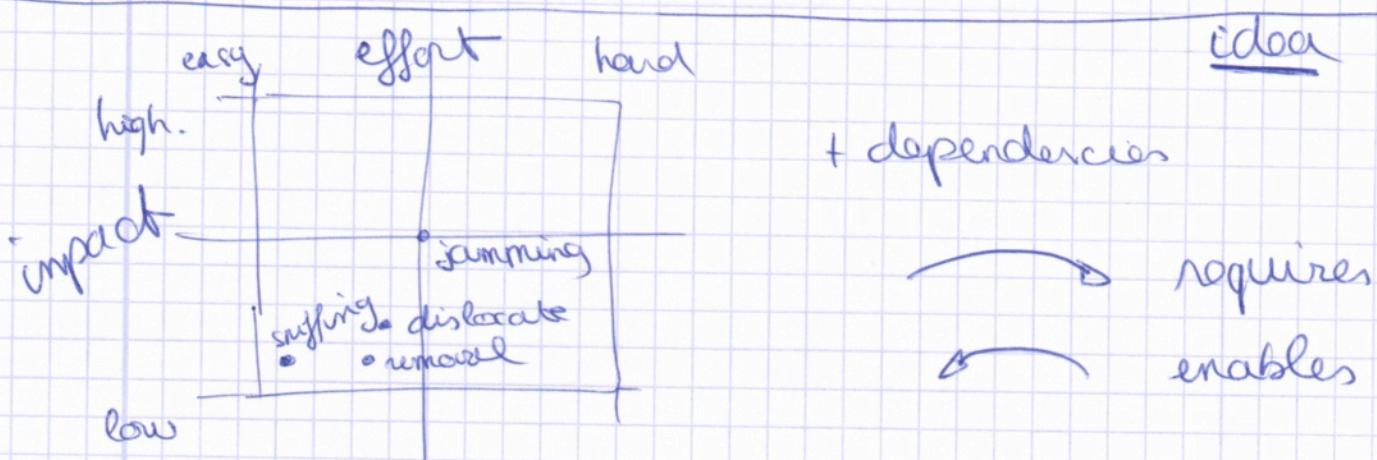
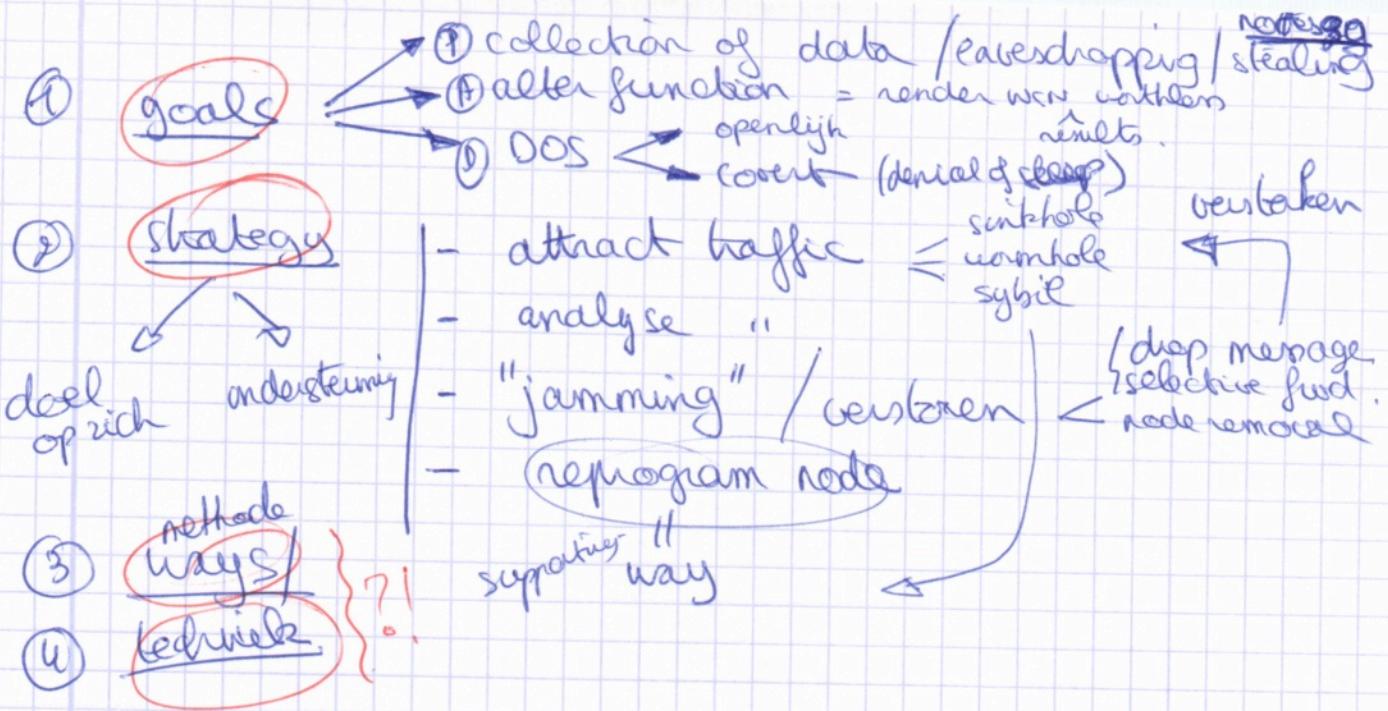
rushing

attract traffic

- ignore delays / timeouts
- for control packets

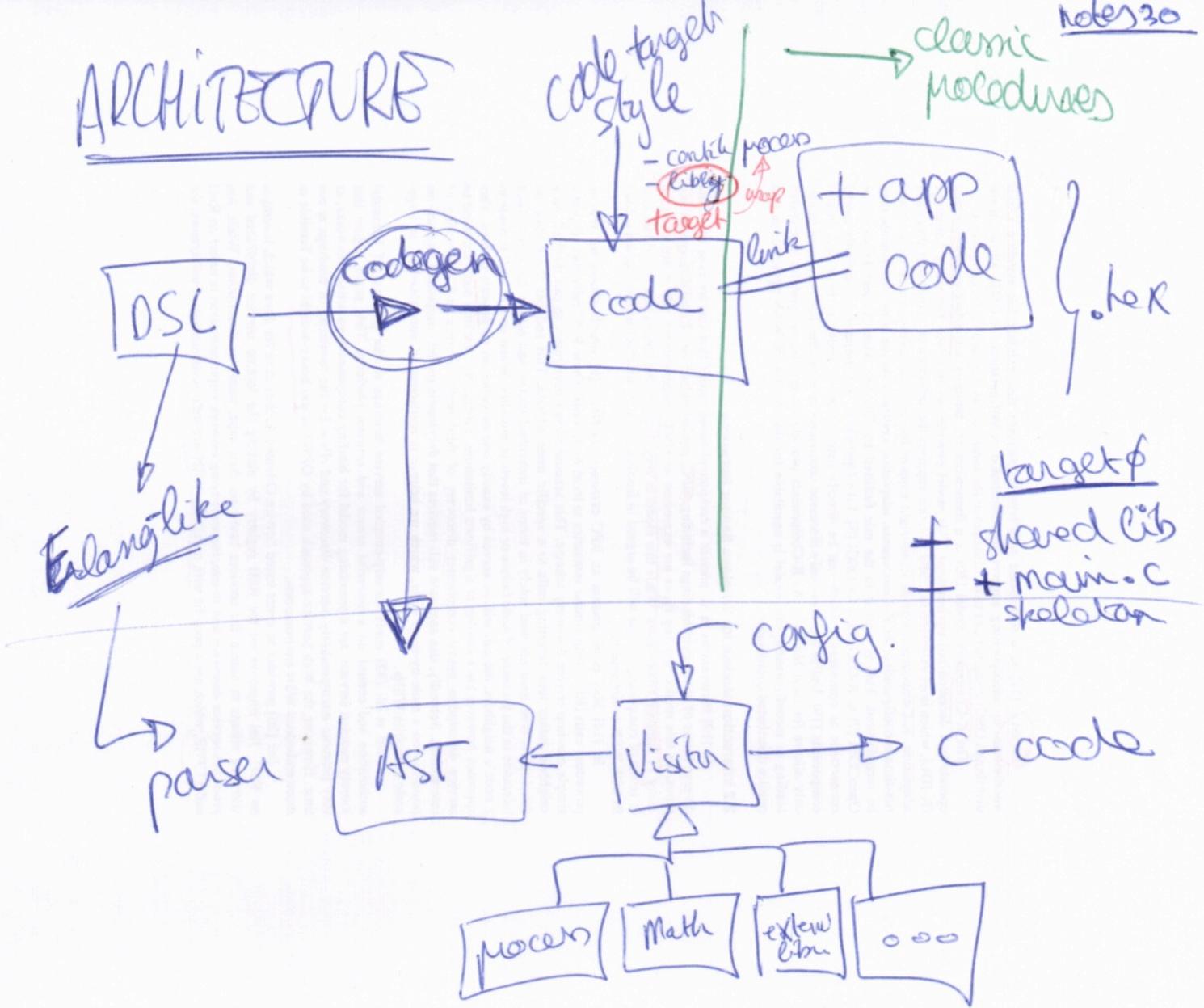
denial-of-service

replication → base for e.g. sybil



goal.	strategy	methods	technique
attack			

ARCHITECTURE



ATTACKS~~corruption~~

Sinkhole
 Wormhole
 Sybil
 selective forwarding
 (↳ cooperative ("anomaly"))
 hello flood
Node dislocate }
 remove }
 insert }
 replication }

~~access memory~~
 reprogram as "anomaly" / (SA)

cooperative decisions → generic FW?

"HIDS" sanity checks (combo ICE-style)

Malfunction!?

~~sniffing~~
 jamming → anomaly
 rushing → anomaly
 Denial-of-Sleep → anomaly
 → more accelerometer (HW)

ERLANG (-style) DSL

Notes 32

communication

receive loop () →

Receive

{ onAny, Pid } →

% actions

{

{ →

% actions.

after 5000 →

% timeout action

end.

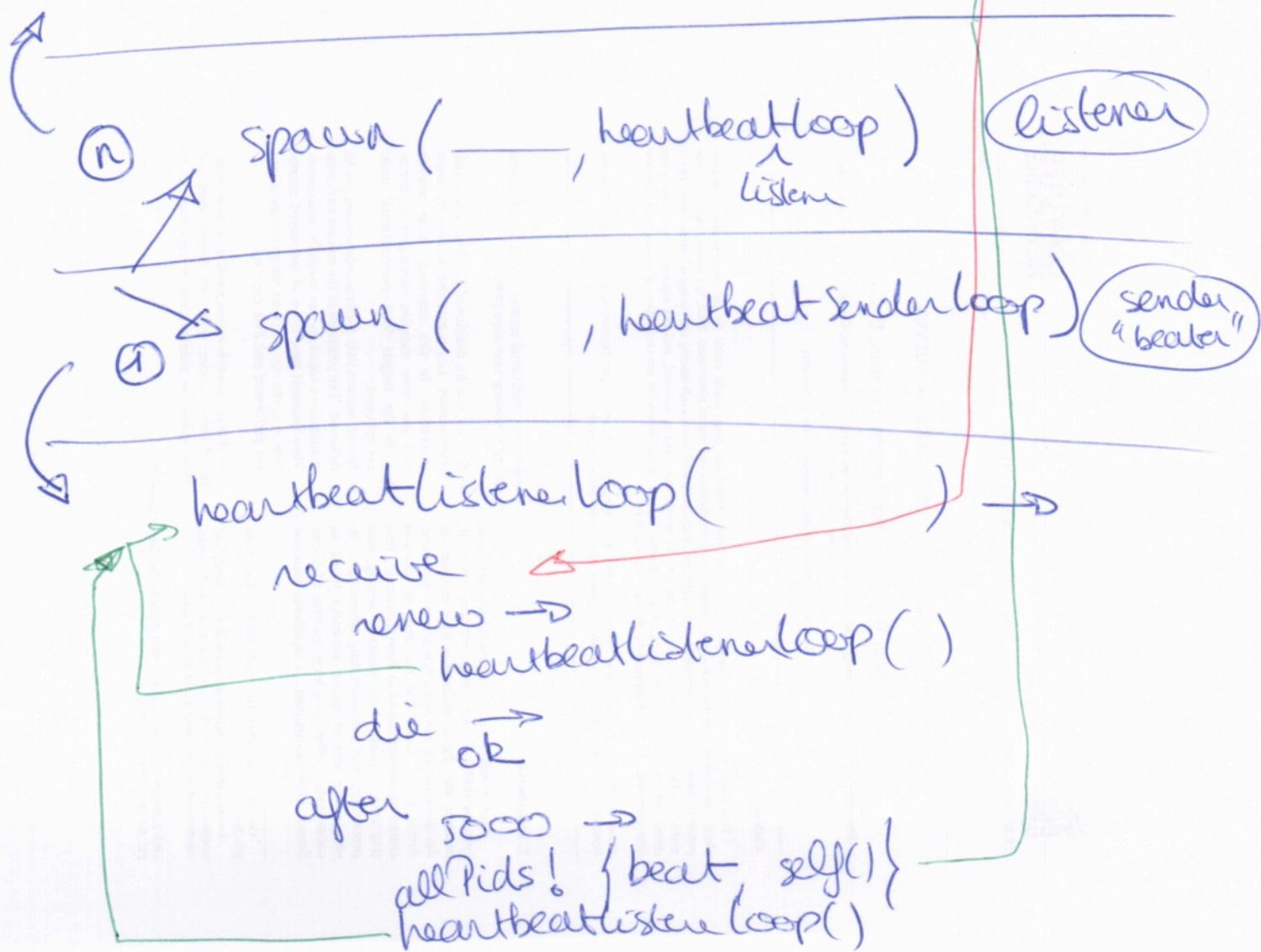
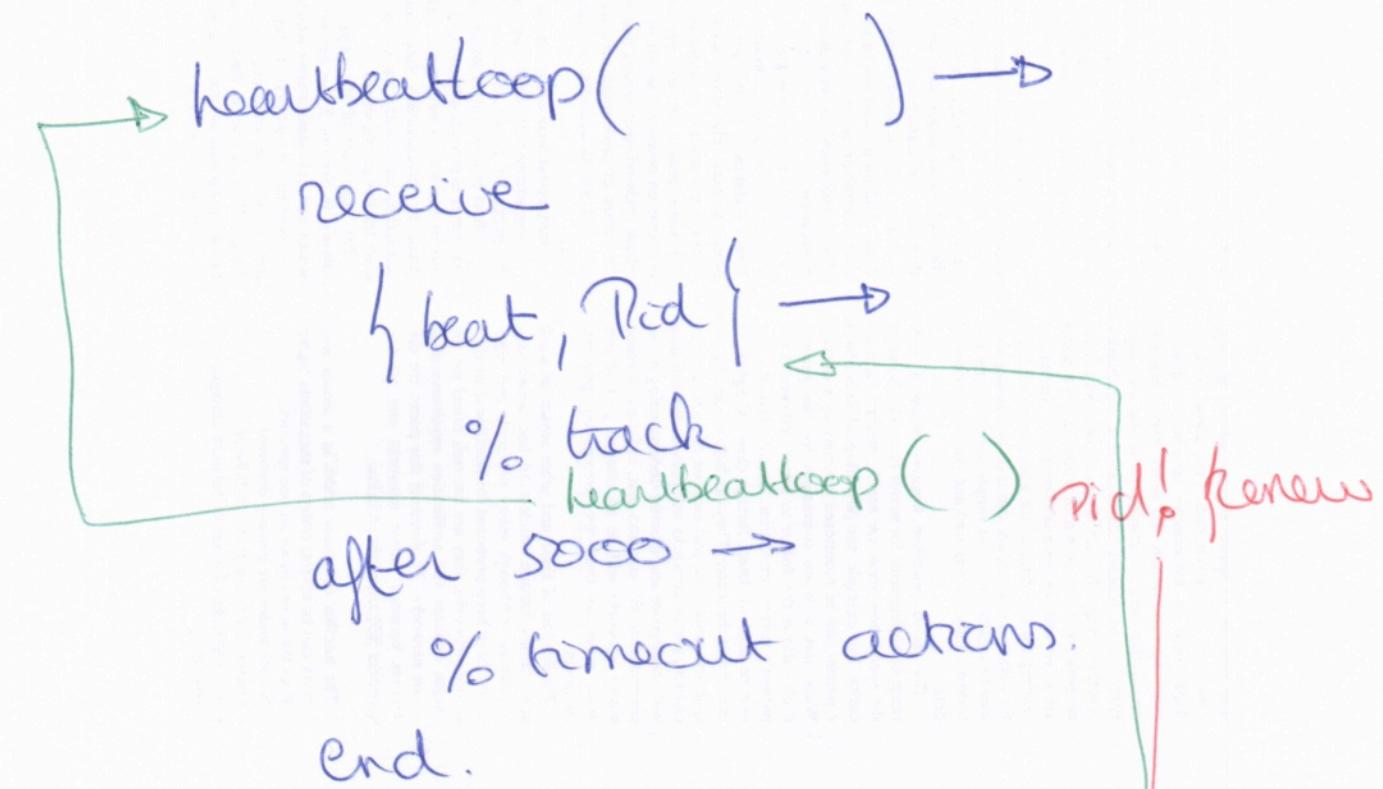


Pid ! { onAny, self() }

process

Spawn (module , function)

example heartbeat



example heartbeat "pseudo code"

Strategy event driven

↳ event table

process events → "near" realtime
 (very near:-))

{ event "wait for heartbeat from Pcd , timeout, epoch"

event "heartbeat from Pcd , time , epoch"
 ↳ serial synchronisation

→ process stack style → pop, merge, ...

↳ VM

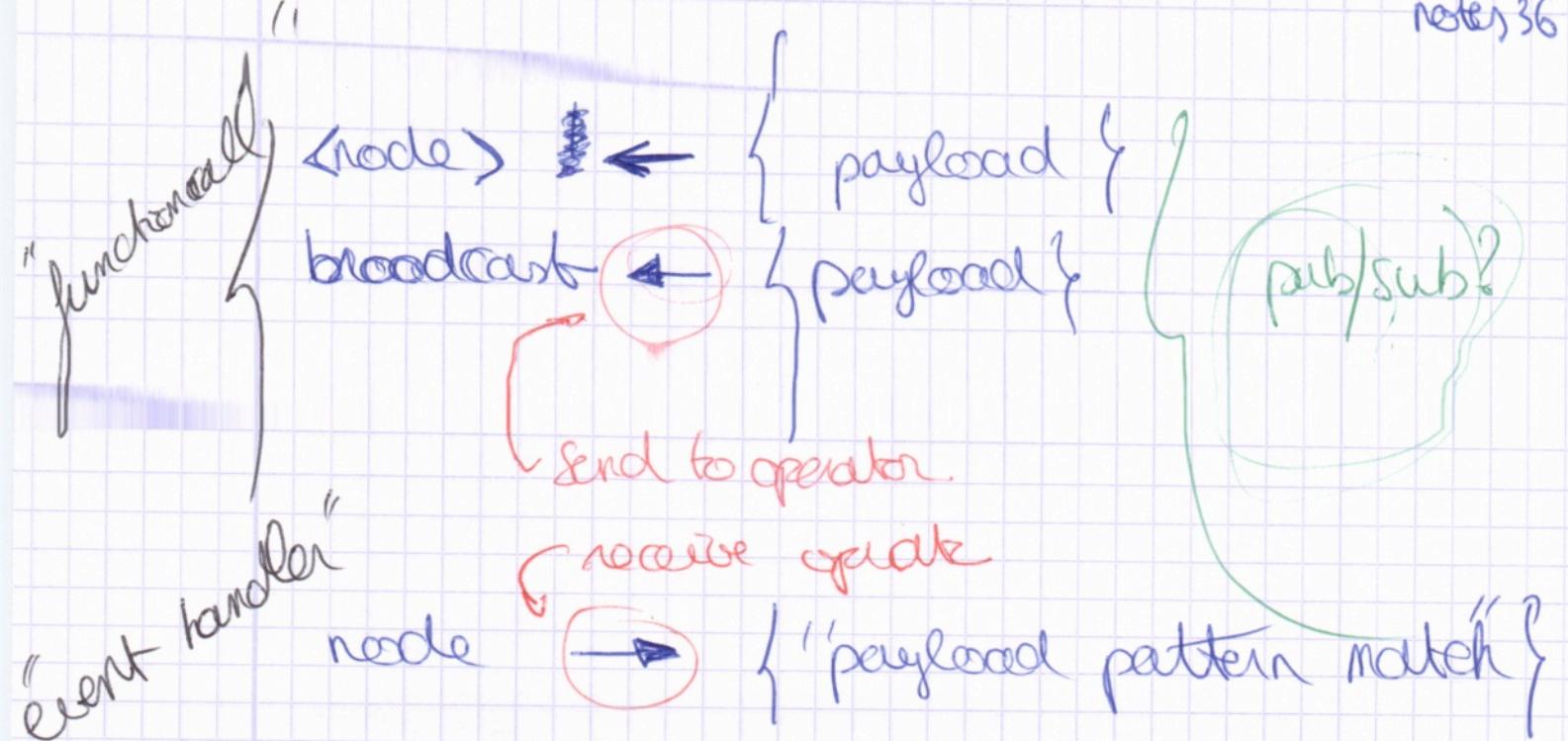
modular
 allow for expansion
 Use **Nodes**
 "variable" with module of funcs
 type?
 nodes.prototype = { key : type, value }
 app specific info
 node info
 extends "basenode"
 ↳ includes - unique id
 build-in hash type
 also [] ext.
 Function to "handle" "process"
 nodes
 visitor
 is valid? principle
 how deal with other nodes?
 handle/condition of event?
 lookup?
 ↳ no loops
 validate
 interval
 too restrictive
 scheduling?
 hash (impl. struct)
 nodes.on - receive (from, payload) {
 nodes[from].property = payload.property
 basic logic available
 math: +, -, *, /, sin, cos, ...
 boolean: and, or, not
 cond.: if then else
 switch case
 reuse C

other modules:
 "possible"

sensors
 actuators

crypto (e.g. sha1, ...)
 memory → e.g. for SA

time (e.g. time.now())



→ ? "event raising" raise()
 "for exception handling" :-)

eg. <do . . . >

what's different after 5000
 few functions?
 few calls?

raise("not ok")
 → handle 'not ok' { . . . }

→ expliciteren van asynchroniteit

reading yet
 repetitive

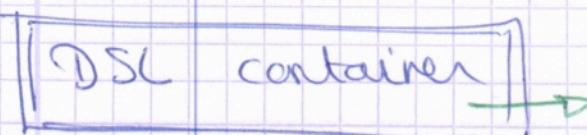
Time ref point
 "after prev start"
 => . . .

send-something(10);
 receive
 type = Monicous, from = , payload = {
 after 20000:
 ("whoops did not send" => nodes.forwardfails+)

"synchronous send style"
 × → ?

✓ immutable

send(node, payload), after(send) = {
 receive . . .
 after . . . })



implement
DSL to structure event-driven
and code-executing DSL's
by getting \hookrightarrow code generation

const

use "DSL"

extend "DSL"

with $\{ K : T = V \}$ \hookrightarrow typed hash

when "DSL scope" <function> do <function>

<name>
function(...) { ... }

with "DSL scope" do <function>

@ annotation

every (time interval)

language constructs

if (<bool-exp>) {<stmts>} else {<stmts>}

memberaccess <object>. <member>

method call <object>. <method>(<args>)

assignment <var> = <value>

+ implicit declaration on first use?!

+ type inferences?!

math + - * / % ()

| push

Lang.

case < variable >
contains

list → methods.

notes 38

? why not simply reuse if()...

```
if(<var>.contains) { ... }  
else if(var.contains) { }  
else { }
```

syntactic sugar?

TODO

? incomplete boolean expressions. ↗

↳ matching

? indicating
• < now() , ->

don't care

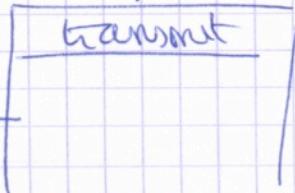
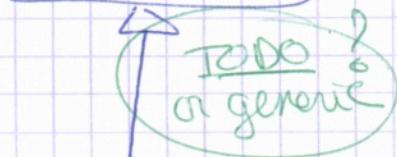
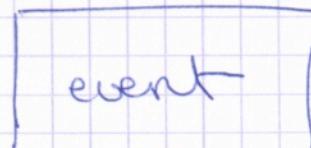
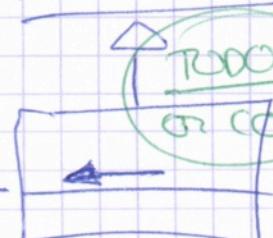
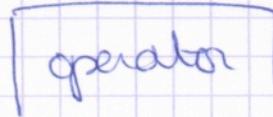
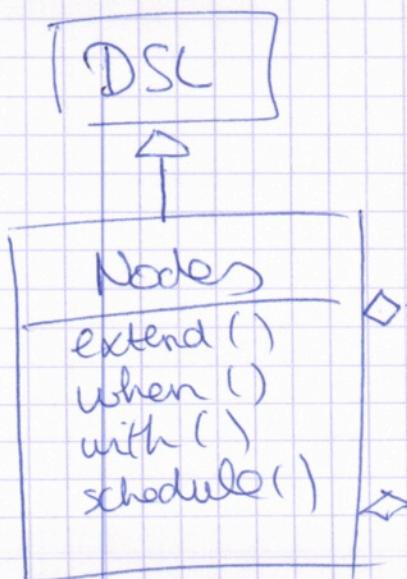
" " ≈ . == .

idem ≡ any.

function(rub) { return true {

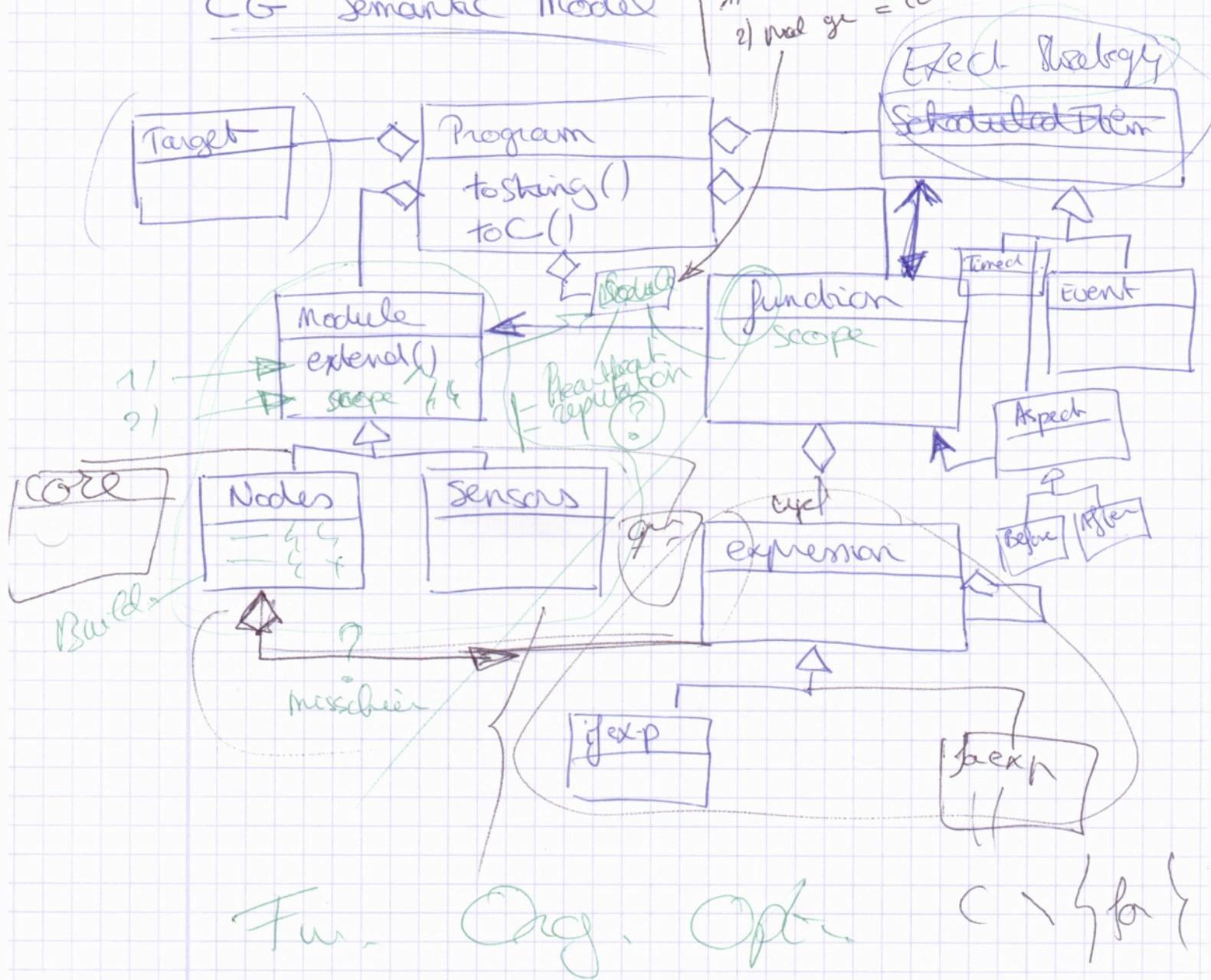
syntactic sugar?
closure?

function(value) { value < now()



CG Semantic Model

1) scope gen = first cond
2) mod gen = cond
notes 33



Fur. Org. Opt.

C \{fa}

200 → API f.

100 → DSC function pool

S. c

When → after / before ?

↳ event ?
≈ after .

↳ aspect oriented.