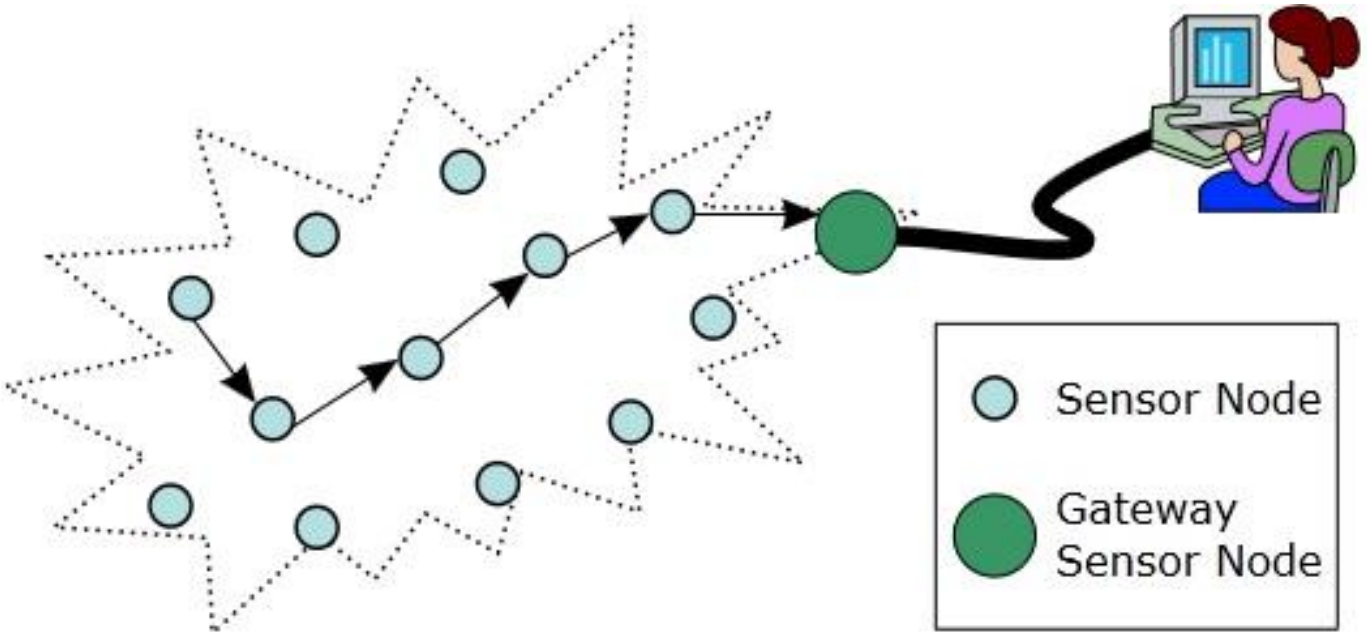
Sieci sensorowe – zastosowania i narzędzia dla programistów

Karol Strzelecki

Wprowadzenie



TinyOS - założenia

- Odzwierciedlenie natury sensorków
- Skalowalność i adaptowalność
- Współbieżność

Mote Type	WeC	rene2	rene2	dot	mica
					
Date	9/99	10/00	6/01	8/01	2/02
Microcontroller					
Type	AT90LS8535		ATMega163		ATMega103
Prog. mem. (KB)	8		16		128
RAM (KB)	0.5		1		4
Communication					
Radio	RFM TR1000				
Rate (Kbps)	10	10	10	10	10/40
Modulation type	OOK				OOK/ASK

TinyOS – główne cechy

- Architektura komponentowa
- Współbieżność zdarzeń (ang. *event*) i zadań (ang. *task*)
- Operacje ‘*split-phase*’

nesC

- Język dla sensorków
- Rozszerzenie C
- Odzwierciedla architekturę TinyOS’a
- Statyczna alokacja pamięci
- Kompilowany do C

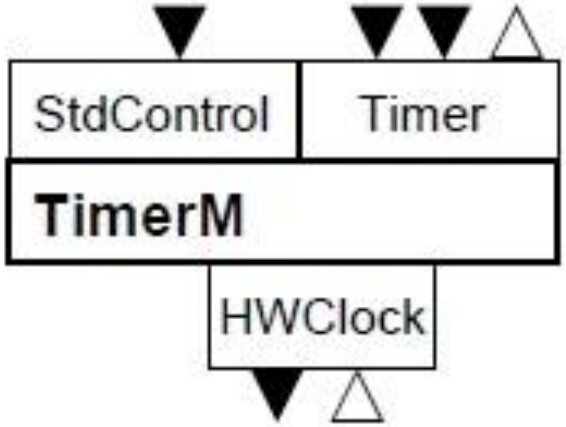
nesC – architektura komponentowa

- Program do kolekcja komponentów
- Komponent to moduł lub konfiguracja
- Moduł to implementacja interfejsów
- A konfiguracja to wiązanie interfejsów z modułami

nesC - interfejsy

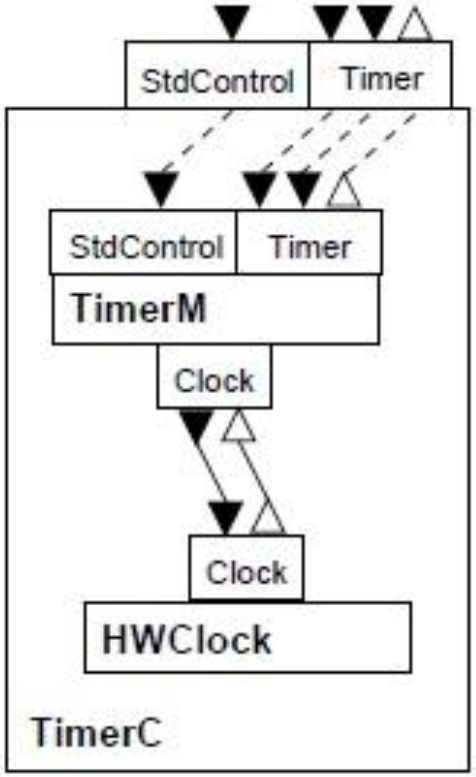
```
interface Timer {
    command result_t start(char type, uint32_t
        interval);
    command result_t stop();
    event result_t fired();
}
```

nesC - moduł




```
module TimerM {
  provides {
    interface StdControl;
    interface Timer;
  }
  uses interface Clock as Clk;
} ...
```

nesC - konfiguracja




```
configuration TimerC {
  provides {
    interface StdControl;
    interface Timer;
  }
}
implementation {
  components TimerM, HWClock;

  StdControl = TimerM.StdControl;
  Timer = TimerM.Timer;

  TimerM.Clk -> HWClock.Clock;
}
```

nesC – zadania (*tasks*)

- Odkładanie obliczeń na później
- Definiuje się z nową naklejką…

```
task void doSth(){
    //...
}
```

- A odkłada tak:

```
post task doSth();
```

TinyOS – model wykonania

- Prosty scheduler:

```
while(jest_task_w_kolejce){
    wyrzuć_task_z_kolejki();
    wykonaj_task();
}
```

- Rozmiar kolejki znany podczas kompilacji
- Przerwania

nesC - współbieżność

- 2 naklejki na procedury: kod synchroniczny, kod asynchroniczny
- Synchroniczny – osiągalny tylko z tasków
- Asynchroniczny – osiągalny z jakiejś procedury obsługi przerwania
- Potencjalne wyścigi między kodem synchronicznym, a asynchronicznym

nesC – wsparcie dla problemów współbieżności

- Sekcje *atomic*
- Kompilator łatwo wykrywa zmienne, o które może być konflikt

Ewaluacja

Wykorzystanie modelu komponentowego TinyOS'a

Application	Modules	OS Modules (% of full OS)	Lines	OS Lines (% of full OS)
Surge	31	27 (25%)	2860	2160 (14%)
Maté	35	28 (25%)	4736	2524 (17%)
TinyDB	65	38 (35%)	11681	4160 (28%)

Ewaluacja cd.

Aplikacje o wysokim poziomie współbieżności

Application	Task count	Event count	% interrupt code
Surge	7	221	64%
Maté	13	317	41%
TinyDB	18	722	57%

Kompilator wykrył ok. 150 potencjalnych bugów, z czego ok. 50 stanowiły ‘*false-positive*’

Optymalizacje

Użycie funkcji *inline,* eliminacja martwego kodu, propagacje stałej

App	Code size		Code reduction	Data size	CPU reduction
	inlined	*noninlined*			
Surge	14794	16984	12%	1188	15%
Maté	25040	27458	9%	1710	34%
TinyDB	64910	71724	10%	2894	30%

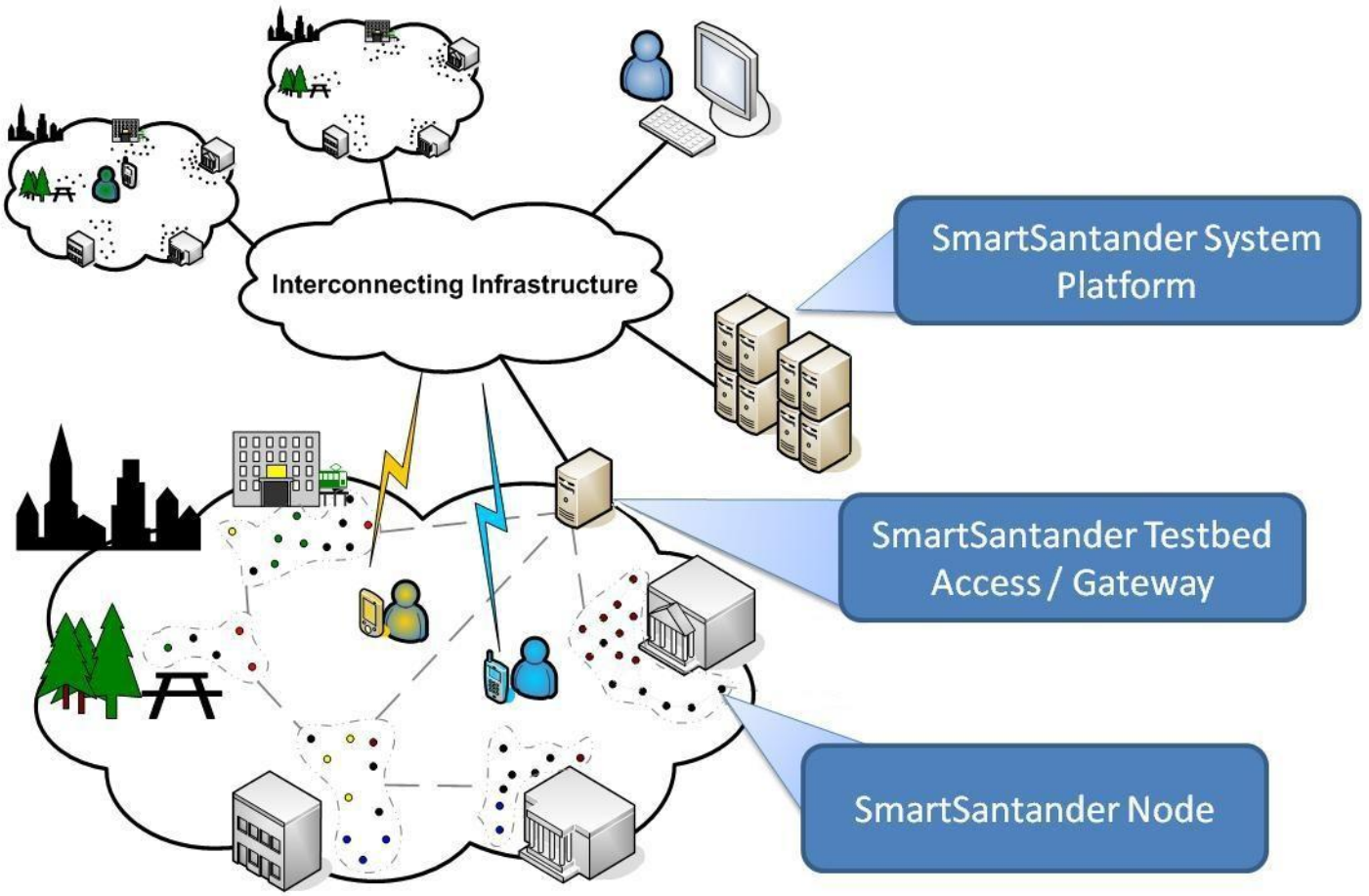
Nasz wkład

- Framework do *unit test'ów*
- Analizator pokrycia kodu

Smart Santander

- Duża (w założeniach 20k sensorów), eksperymentalna sieć
- ‘Realne’ warunki – rozrzucona w mieście
- Nie tylko testy – nastawienie również na promocję oraz próba stworzenia spójnej architektury

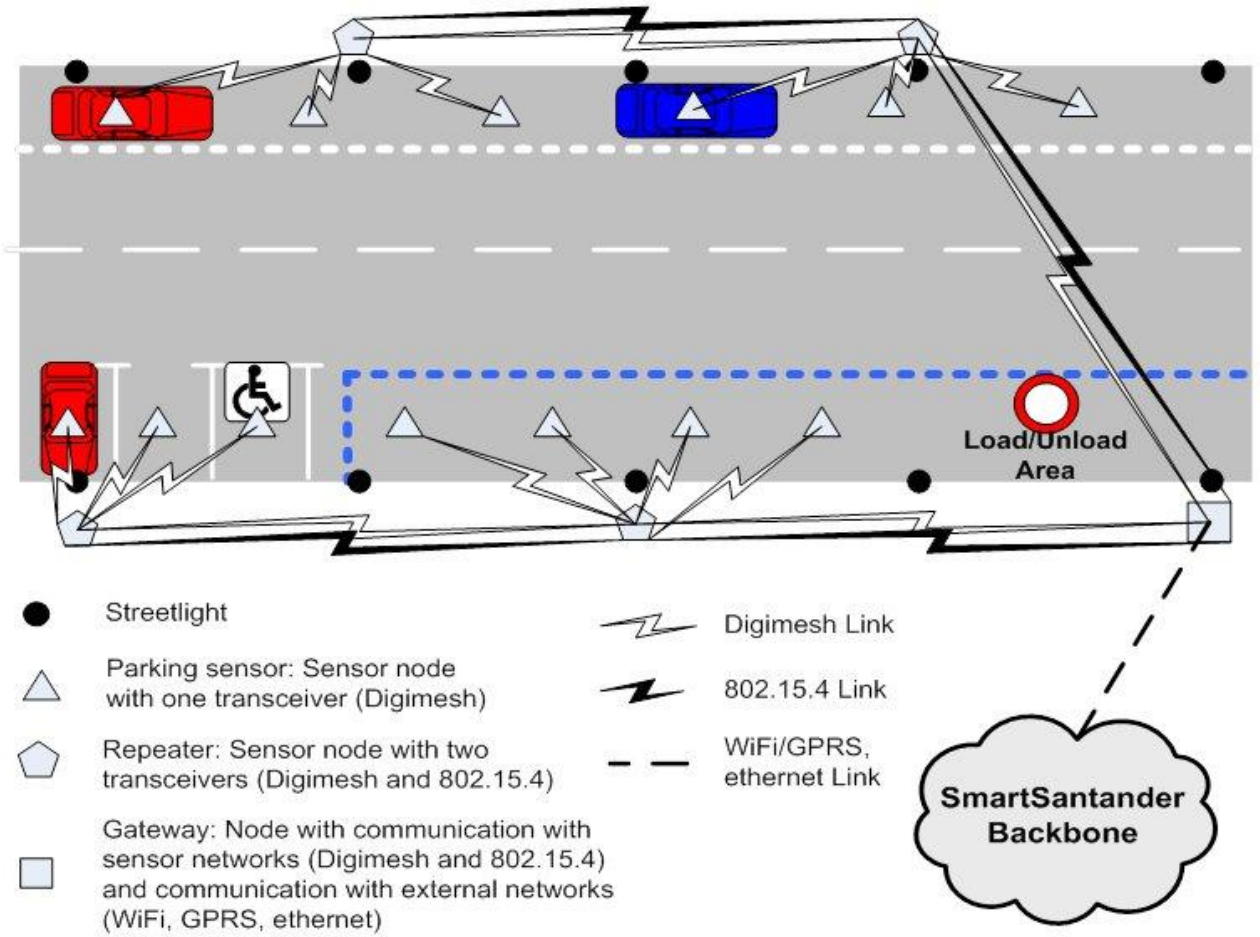
(Bardzo) wysokopoziomowy schemat

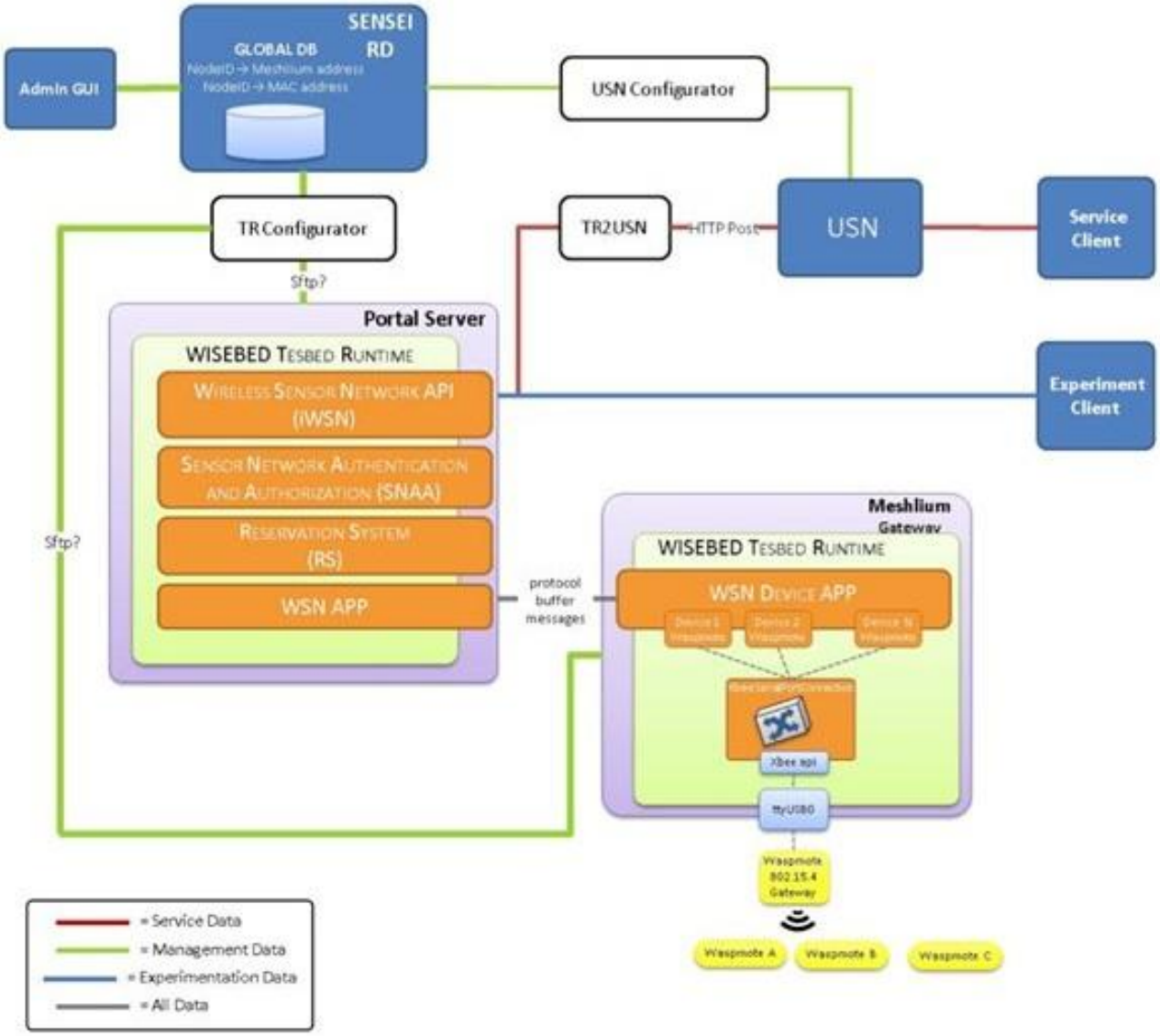


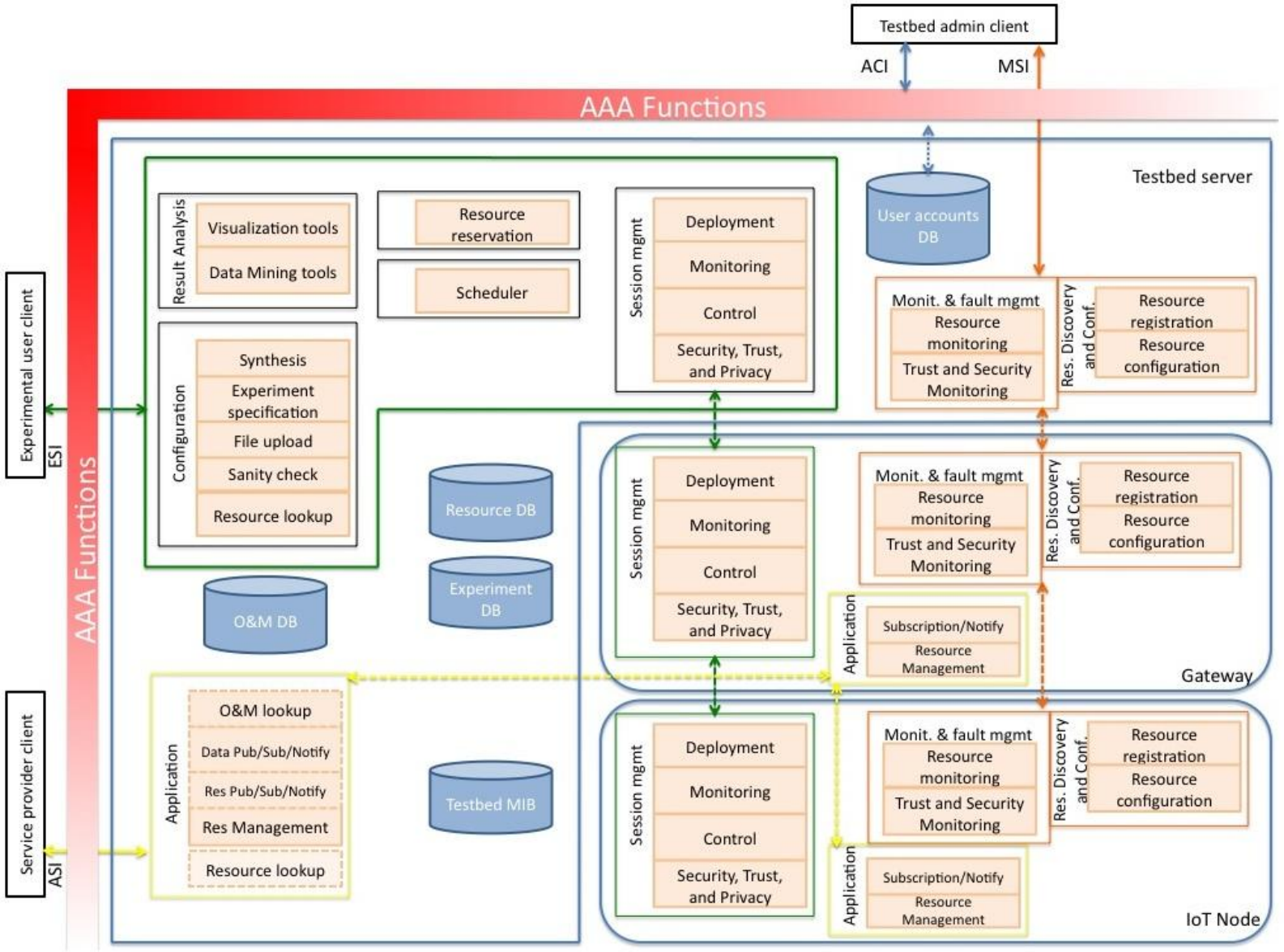
Testbed

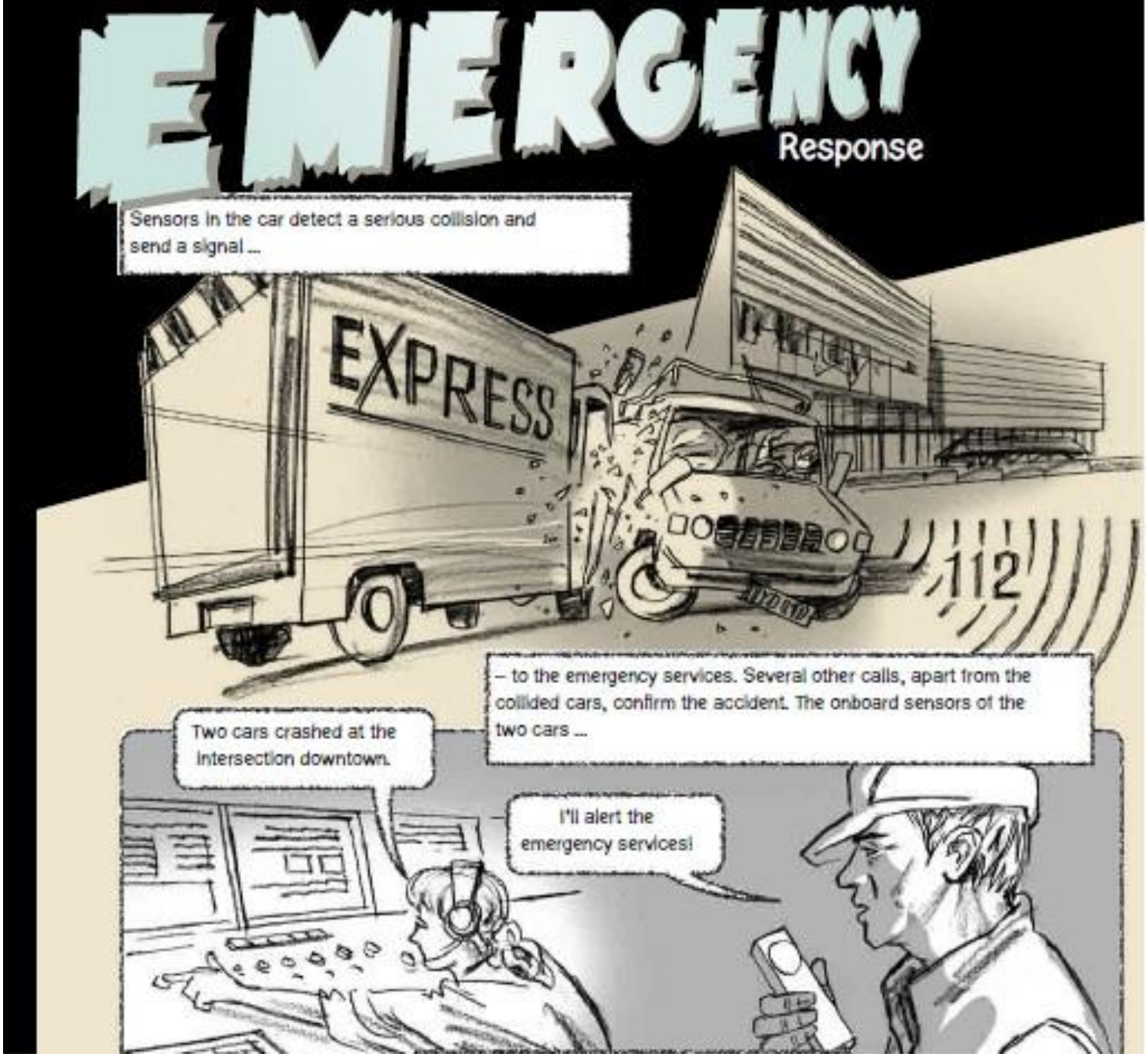
- Obecnie ok. 2k sensorów
- 3-poziomowa architektura:
  - Sensor
  - *Reapeter*
  - *Gateway*

Testbed cd.


Load/Unload Area
Streetlight
Parking sensor: Sensor node with one transceiver (Digimesh)
Repeater: Sensor node with two transceivers (Digimesh and 802.15.4)
Gateway: Node with communication with sensor networks (Digimesh and 802.15.4) and communication with external networks (WiFi, GPRS, ethernet)
Digimesh Link
802.15.4 Link
WiFi/GPRS, ethernet Link
SmartSantander Backbone


SENSEI RD
GLOBAL DB
NodeID -> Meshlium address
NodeID -> MAC address
Admin GUI
USN Configurator
TR Configurator
TR2USN
HTTP Post
USN
Service Client
Sftp?
Portal Server
WISEBED TESBED RUNTIME
WIRELESS SENSOR NETWORK API (iWSN)
SENSOR NETWORK AUTHENTICATION AND AUTHORIZATION (SNAA)
RESERVATION SYSTEM (RS)
WSN APP
Experiment Client
Sftp?
protocol buffer messages
Meshlium Gateway
WISEBED TESBED RUNTIME
WSN DEVICE APP
Xbee api
ttyUSB0
Waspmote 802.15.4 Gateway
Waspmote A
Waspmote B
Waspmote C
= Service Data
= Management Data
= Experimentation Data
= All Data


Testbed admin client
ACI
MSI
AAA Functions
AAA Functions
Testbed server
Result Analysis
Visualization tools
Data Mining tools
Resource reservation
Scheduler
Configuration
Synthesis
Experiment specification
File upload
Sanity check
Resource lookup
Session mgmt
Deployment
Monitoring
Control
Security, Trust, and Privacy
User accounts DB
Monit. & fault mgmt
Resource monitoring
Trust and Security Monitoring
Res. Discovery and Conf.
Resource registration
Resource configuration
Experimental user client
ESI
Resource DB
Experiment DB
O&M DB
Gateway
Application
Subscription/Notify
Resource Management
Service provider client
ASI
O&M lookup
Data Pub/Sub/Notify
Res Pub/Sub/Notify
Res Management
Resource lookup
Testbed MIB
IoT Node


EMERGENCY
Response
Sensors in the car detect a serious collision and send a signal ...
EXPRESS
112
– to the emergency services. Several other calls, apart from the collided cars, confirm the accident. The onboard sensors of the two cars ...
Two cars crashed at the intersection downtown.
I'll alert the emergency services!

Dziękuję za uwagę