IPSN 2010
April 12-15
Stockholm, Sweden

Using Meta-Code for Building Task-Specific WSNs

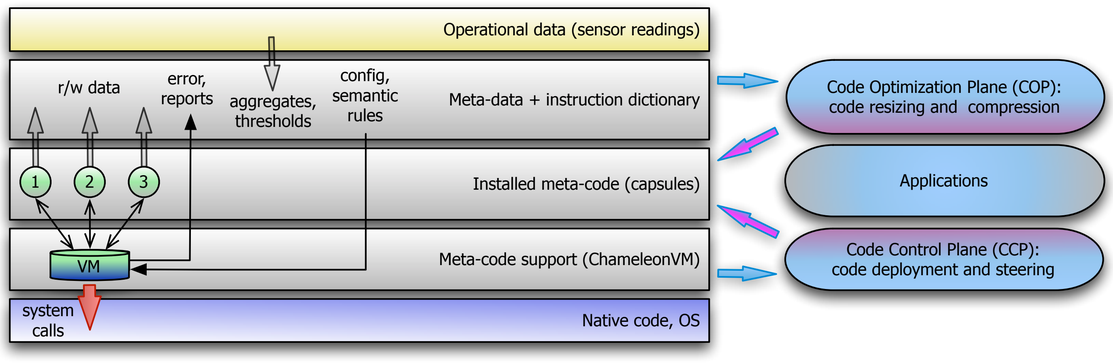
Igor Talzi, University of Basel, Switzerland

Objectives

- Low-level *network* programming model which will allow to
  - build and deploy whole network protocols or separate layers *on-the-fly*,
  - create *task-specific* configurations,
  - and *optimize* system behavior in real time.
- Targets: *system-level network* protocols (time-sync, routing, etc) running on embedded devices (WSN, programmable routers, etc).

What is Meta-Code?

- Platform-independent framework: language (Meta-Lang), embaddable execution environment (ChameleonVM) and front-end support tools.



- Provides: code deployment, execution and *version* control.
- Inspired by: Active Networks and Artificial Chemical Computing (e.g. *fraglets*).

ChameleonVM

- Embeddable, stack-based, type-free.
- Features *dynamically extensible* instruction set.
- Uses system calls to underlying OS (e.g. for ”send” operation).
- Operates with *capsules*.

Capsules

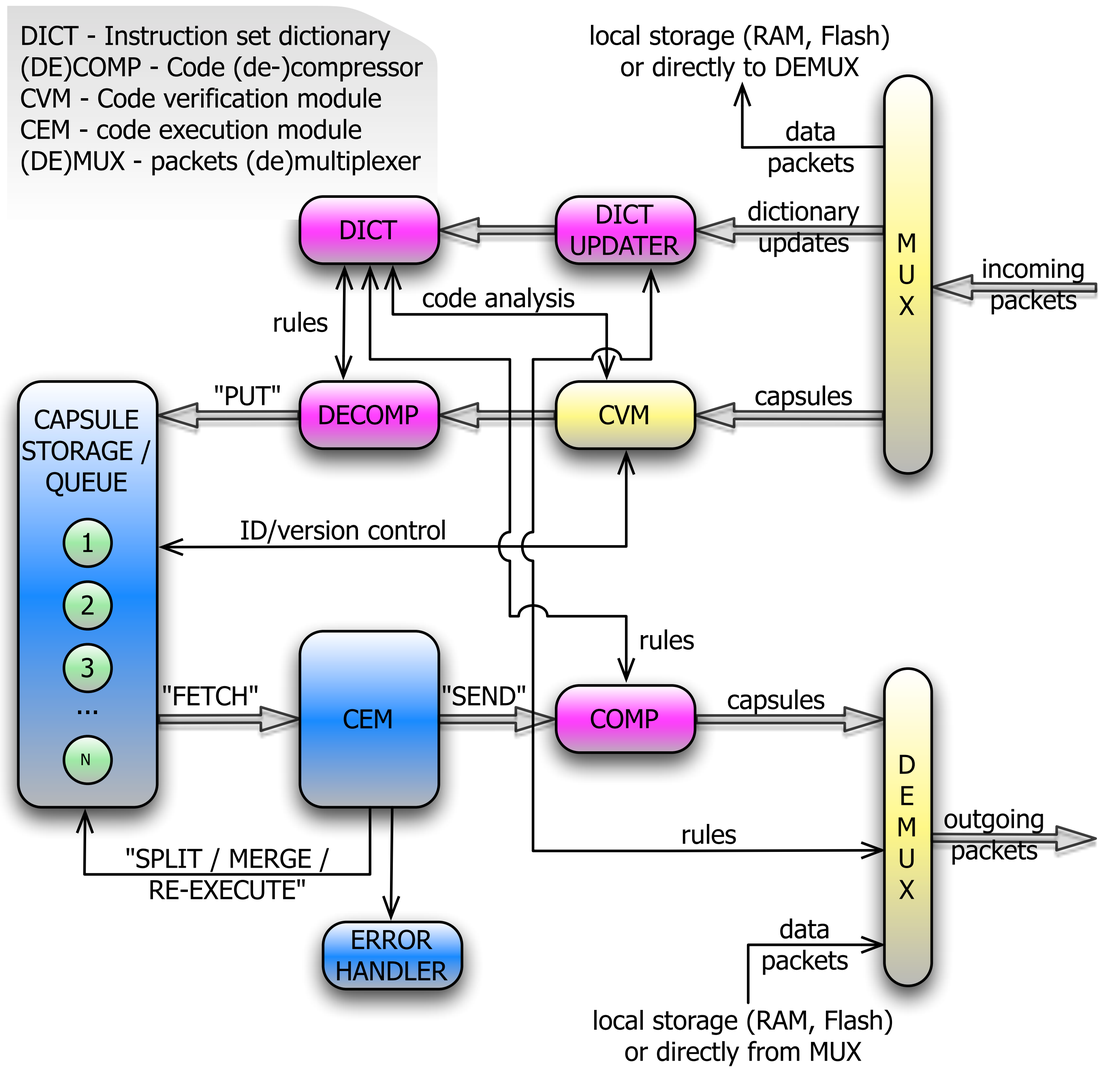
- Can travel through the network and *react* with each other.
- Have a *limited life-time*.
- Can *split* and *merge* with other capsules.
- Executed in a memory-protected environment.
- Code is *versioned*.

Extensible Instruction Set Dictionary

- Resides on each node as a part of ChameleonVM.
- Allows to add new, remove obsolete and change semantics of current instructions.
- Different mapping of the same instruction on different nodes or at different time points on one node (*code polymorphism*).
- Code compression via *dynamic re-encoding*.

Node Architecture

- On-node execution environment is used to propagate, (un)install and run meta-code - no other *pre-deployed* software is needed.



Showcase: Count the Number of Nodes

This self-propagating capsule is executed locally on each node:

```
.sys                  # SYSTEM segment
    AUTOUPDATE 1
    LIFETIME 10s
    ID 0x21
.code.init            # CODE segment "init"
    send ME,ALL       #   broadcast itself
    die TOP           #   clean the code located above
    send ME,S         #   send it up the spanning tree
    die
```

The second capsule is executed locally on the sink node *S*; it calculates all incoming ”counting” capsules:

```
.sys                  # SYSTEM segment
    AUTOUPDATE 1
    LIFETIME 10s
    ID 0x31
.code.cap             # CODE segment "receive capsule"
    push CAP.ID       #   count "marked" capsules only
    jmpeq 0x21,l1
    exit

l1: inc BUFS[0]
    exit
```

Implementation

- Initial tests under TinyOS-2 and ContikiOS.
- User input: capsules, dictionary updates, configuration.
- Examples: spanning tree, route discovery, id-assignment, etc.
- TBD: *self-regulating* network architecture (minimize user input) and autonomous compression scheme.

http://www.fraglets.net/

http://cn.cs.unibas.ch/

Contact:
Igor Talzi, Christian Tschudin
Computer Science Department, University of Basel
Bernoullistrasse 16, CH-4056 Basel, Switzerland
{igor.talzi,christian.tschudin}@unibas.ch

UNI BASEL