A time, energy and security coordination approach

UNIVERSITEIT VAN AMSTERDAM

Benjamin Rouxel, Julius Roeder,
Sebastian Altmeyer and Clemens Grelck

benjamin.rouxel@uva.nl

WATERS, Stuttgart, July 09, 2019

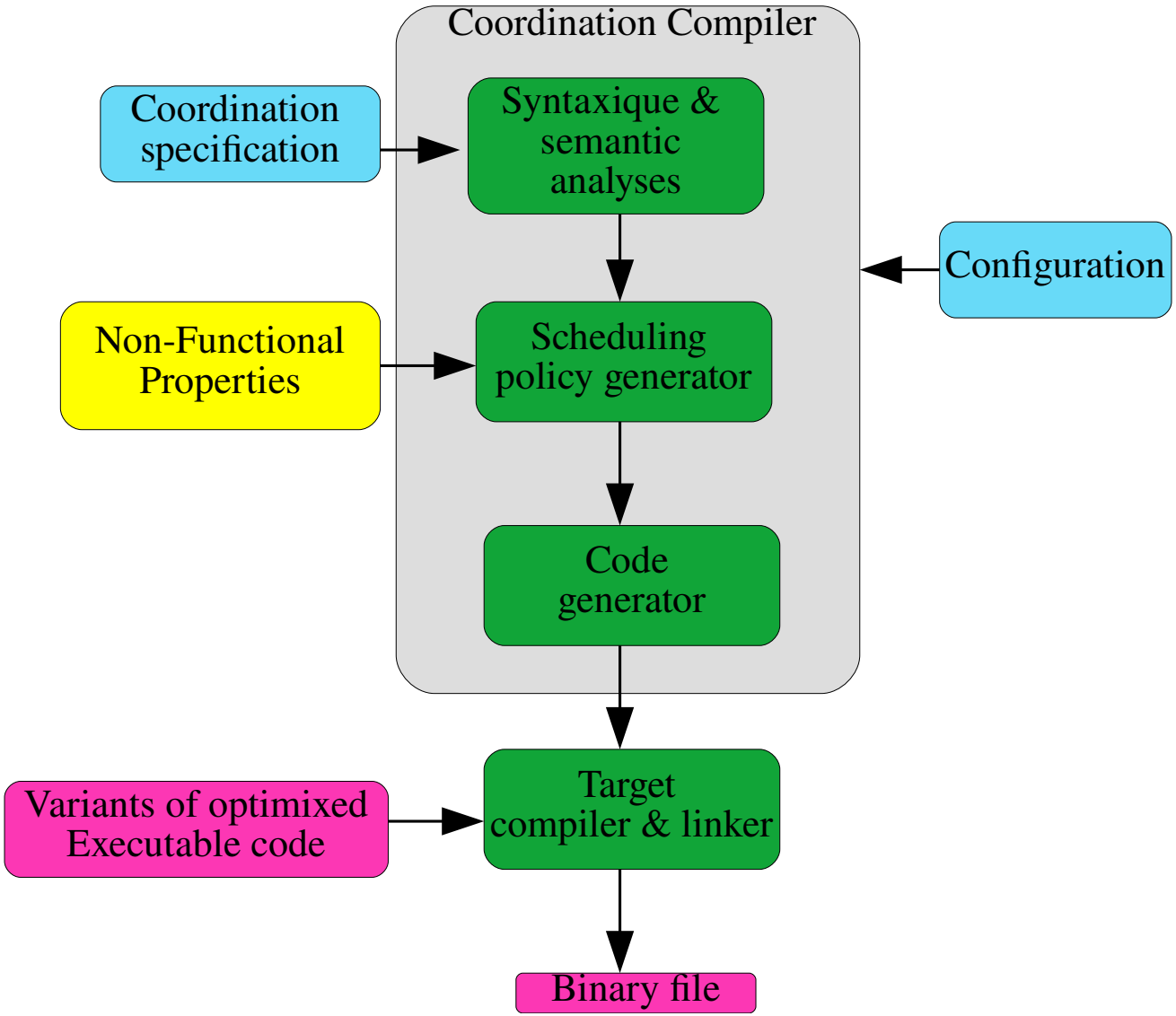
What is coordination ?

Model-Based Engineering (MBE) — Skeleton-Based Programming

> High level programming
> Structural information available

- Increase productivity
- Lower programming error
- Hide low-level parallelism mechanism
- Hide low-level communication mechanism
- Simplify tuning
- Optimisations on time/energy
- Add security mechanism
- Early verification

Coordination Tool-Flow



Coordination Specification

Goal: Describe overall application structure

- Xtext plugin for Eclipse IDE
- ANTLR in background
- CPP Lexer + Parser generation

Non-Functional Properties

Goal: Make scheduling policy generator ETS-aware

- CSV file
- Header:
  - Component Name,
  - Component Version,
  - Variable Name,
  - Line Number,
  - Filename,
  - WCET,
  - WCEngT,
  - AET,
  - AEngT,
  - Security Level,
  - Architecture,
  - Binary,
  - Callable Name,
  - Signature,
  - Confidence

Scheduling premises

Goal: Ensure scheduling strategy is applicable

Verifications

- Edge coherency check
- Deadlock check
- Cycle check

Graph transformations

- Unnecessary tasks/versions removal (security level)
- Multi-rate applications (production $\neq$ consumption)

Offline scheduling policy generation

- Simplified & understandable
- Improved security integration
- Started on integrating with the coordination framework

Code generation

Goal: Generate glue code to execute coordinate components

Target platforms

- Atalis TK1
- Odroid XU4
- Anything that runs Linux

Off-line schedule

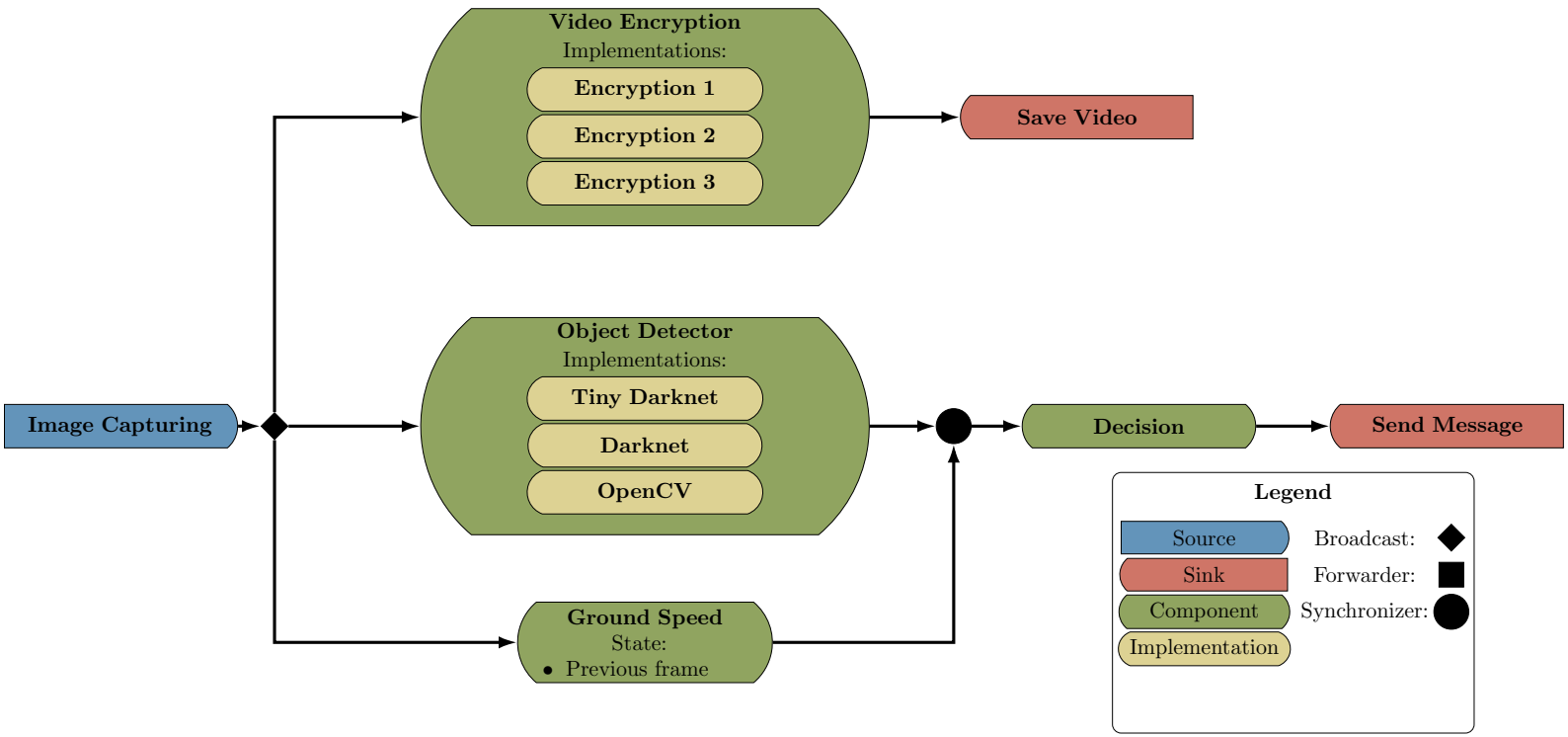
- Work in progress: Linux + Pthread + alarm

On-line schedule

- Ready to test: Linux + Pthread + semaphores
- Work in progress: FreeRTOS, Linux + RT patch

Example: a drone use-case

- Provided by Sky-Watch
- Complex heterogeneous architecture
- Fixed-wing UAV
- Object detection to rescue



Coordination Specification

```
app drone {
  deadline 50Hz
  period 50Hz
  datatypes {
    image_t int[42][42]
    enc_t  long
  }

components {
  component ImageCapture {
    inputs:[]
    outputs:[ (b, 1, image_t) ]
    version v1 {
      targetArch "armv7"
      security [0,100]
    }
  }
  component Encryption {
    inputs:[ (a, 1, image_t) ]
    outputs:[ (b, 1, enc_t) ]
    version AES {security[0,41]}
    version RSA {security[70,100]}
    version SSH3 {security[42,69]}
  }
  component ObjectDetect {
    inputs:[ (a, 1, image_t) ]
    outputs:[ (b, 1, image_t) ]
    version TinyDarknet { security
          [0,100] }
    version Darknet {
      targetArch "armv15"
      security [0,100]
    }
    version OpenCV {
      targetArch "gpu"
      targetArch "armv15"
      security [0,100]
    }
  }
}
edges {
  ImageCapture.b -> Encryption.a
  ImageCapture.b -> ObjectDetect.a
  ImageCapture.b -> GroundSpeed.a
  ObjectDetector.b -> Decision.a
  GroundSpeed.b -> Decision.a
}
}
```

Current & Future works

Coordination specification

- Enable more complex data-flow graph structure

Scheduling

- On-line schedule ETS-aware implementation
- Hardened schedules (timing attacks)
- Multi-graph application scheduling

Code generation

- RTOS

Coordination Specification Language

```
1  Application: 'app' ID '{'
2    'deadline' TIME
3    'period' TIME
4    'datatypes' '{' Datatype* '}'
5    'components' '{' Component+ '}'
6    'edges' '{' Edge* '}'
7  '}';
8  Datatype: ID (CTYPE ('[' INT ']')*
9    | '{' ID (';' ID)* '}');
10 Component: 'component' ID '{'
11   'inputs' ':' '[' Connect* ']'
12   'outputs' ':' '[' Connect* ']'
13   Version+
14 '}';
15 Connect: '(' ID ',' INT ',' ID ')';
16 Version: 'version' ID '{'
17   ('deadline' INT )?
18   ('period' INT)?
19   ('targetArch' STRING)*
20   'security' '[' INT ',' INT ']'
21 '}';
22 Edge: SEdge | DEdge;
23 SEdge: (INT)? CompRef '->' CompRef;
24 DEdge: (INT)? CompRef '->'
25   CompRef  ( '&' CompRef )+;
26 CompRef:ID ('/' ID)? '.' ID ;
```