An Infrastructure for Cooperative Software Verification

Nico Weise

LMU Munich, Germany

LMU LUDWIG-MAXIMILIANS-UNIVERSITÄT MÜNCHEN

SoSy-Lab Software Systems

Motivation

- Cooperation is known to improve verification performance
  - But: cooperation in single-machine environments can not be scaled
- Multi-machine environments are known to improve efficiency
  - But: Non-cooperating setup do take full advantage of that potential
- Implementing multi-machine communication for each possible use case is not practical!

Motivation

- Cooperating actors
  - use different algorithms,
  - use different formalisms,
  - are written in different languages,
  - ...

Writing custom exchange formats for every use case is not pratical!

New Artifact Exchange Format

- Structured with YAML.
- Files contain an array of *entries*.
- *Entry-type* determines schema of the entry.

Example: Loop Invariants

```yaml
- entry_type: loop_invariant
  metadata:
    # ...
  location:
    # ...
  loop_invariant:
    # ...
```

Example: Loop Invariants (metadata)

```
format_version: 0.1
uuid: 91023a0f-9f45-4385-88c4-1152ade45537
creation_time: 2021-05-05T15:18:43+02:00
producer:
  name: CPAchecker
  version: 2.0.1-svn
  configuration: svcomp21--04-kInduction
  description: ...
  command_line: ...
task:
  input_files:
    - multivar_1-1.c
  input_file_hashes:
    multivar_1-1.c: 511f45a...
  specification: CHECK( ... )
  data_model: ILP32
  language: C
```

Example: Loop Invariants (`location`)

```
file_name: multivar_1-1.c
file_hash: 511f45a...
line: 22
column: 0
function: main
```

Example: Loop Invariants (`loop_invariant`)

```
string: (x >= 1024U) && (x <= 4294967295U) && (y == x)
type: assertion
format: C
```

Example: Loop Invariant Certificates

```
- entry_type: loop_invariant_certificate
  metadata:
    # ...
  target:
    # ...
  certification:
    # ...
```

Example: Loop Invariant Certificates (metadata)

```
format_version: 0.1
uuid: 91023a0f-9f45-4385-88c4-1152ade45537
creation_time: 2021-05-05T15:18:43+02:00
producer:
  name: CPAchecker
  version: 2.0.1-svn
  configuration: svcomp21--04-kInduction
  description: ...
  command_line: ...
```

Example: Loop Invariant Certificates (target)

```
uuid: 91023a0f-9f45-4385-88c4-1152ade45537
type: loop-invariant
file_hash: 622g56b...
```

Example: Loop Invariant Certificates (certification)

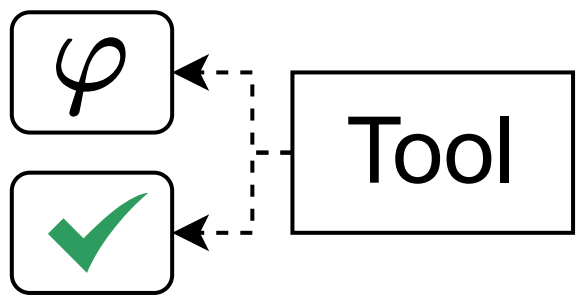
```
string: confirmed
type: verdict
format: confirmed | rejected
```

Communication Middleware

Goal: Support communication of off-the-shelf tools distributed across multiple machines.

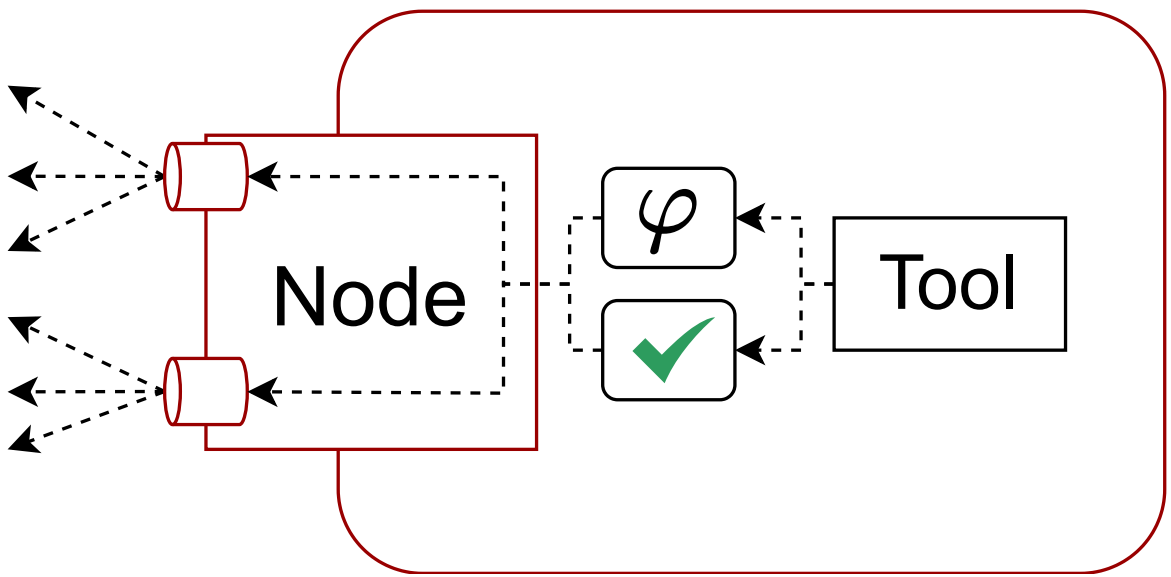
Communication Middleware - Tools

Tools are seen as artifact producers/consumers (c.f. CoVeriTeam).



Communication Middleware - Nodes

- Instruments the actual verification tool.
- Forwards artifacts from tool to (typed) communication endpoints and vice versa.
- Optional processing of artifacts.

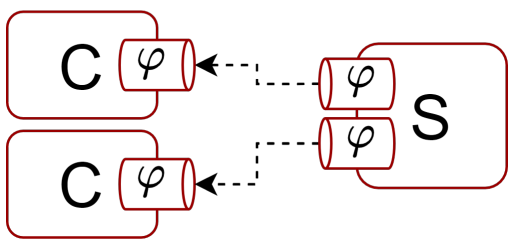


Communication Middleware - Communication

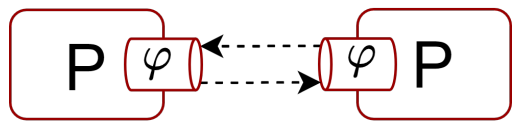
Communication endpoints are specified with

- the type of artifact that they transport (invariants, verdicts, certifications, ...) and
- the messaging pattern:

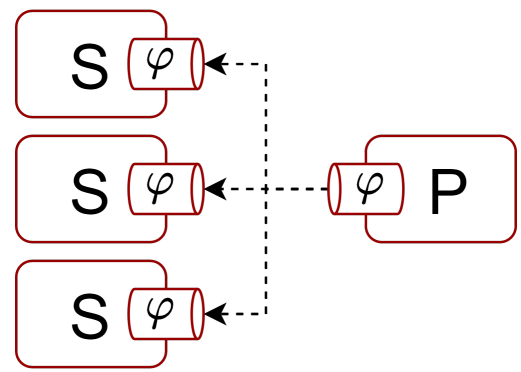
Client-Server



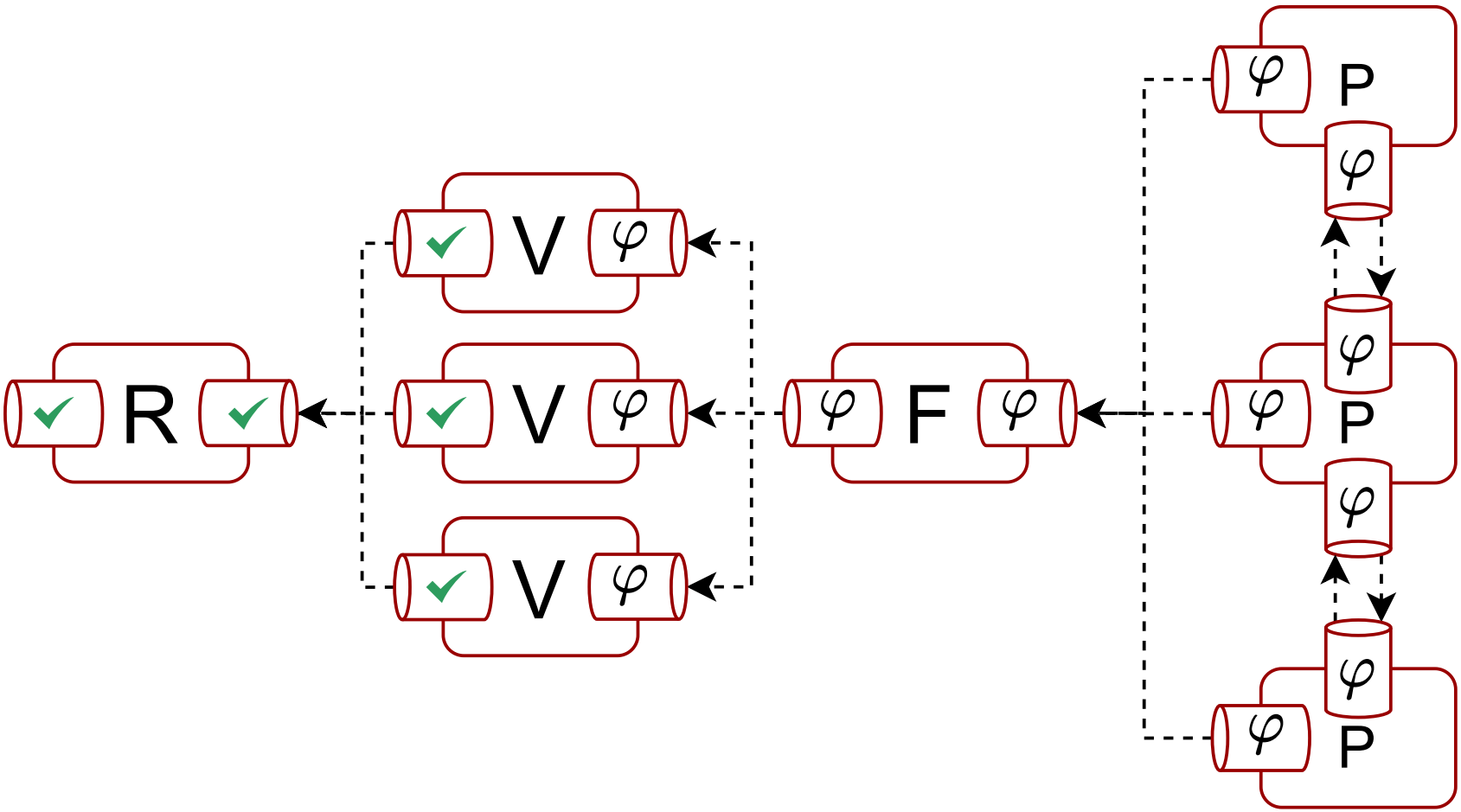
Bidirectional



Publish-Subscribe

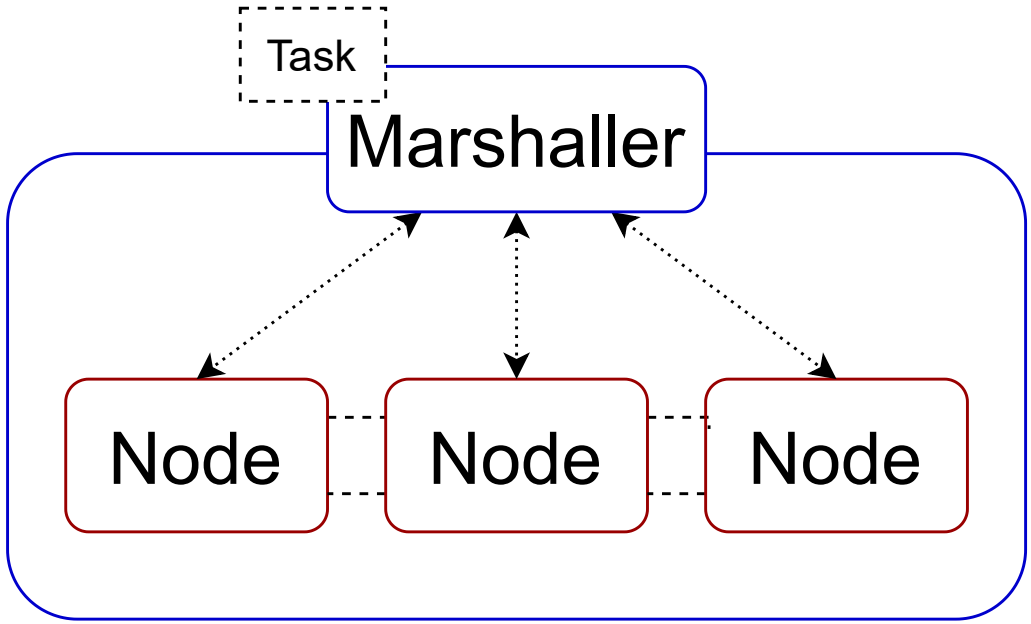


Communication Middleware - Example



Communication Middleware - Mode of Operation

- Setup exists for the duration of one verification run.
- *Marshaller* controls the setup.



Communication Middleware - vCloud Integration

1. Empty nodes register at *controller*.
2. Controller designates a marshaller for the next scheduled verification tasks.
3. Controller sends configurations, marshaller address, and task to nodes.