

PLATFORM FOR ONLINE  
INTEROPERABILITY AND  
PERFORMANCE TEST



# Remote Conformance & Interop Testing

**TPAC2016 – Web of Things IG Meeting – Lisbon**

**22<sup>nd</sup> September 2016**

César Viho & Federico Sismondi

INRIA - France



# F-Interop H2020 Project



- [www.f-interop.eu](http://www.f-interop.eu)
- 1 November 2015 – 31 October 2018
- *develop and provide online interoperability and performance test tools to support emerging technologies from research to standardization and market launch*
- 9 partners



# Goals



1. Describe the F-Interop platform
2. Is this useful for the WoT community?
3. How the WoT community can help?
  - Introduce the F-Interop open call



# Why remote conformance & interop?



## ➤ SDOs

- save time and resources
- running code early
- accelerate standardization process

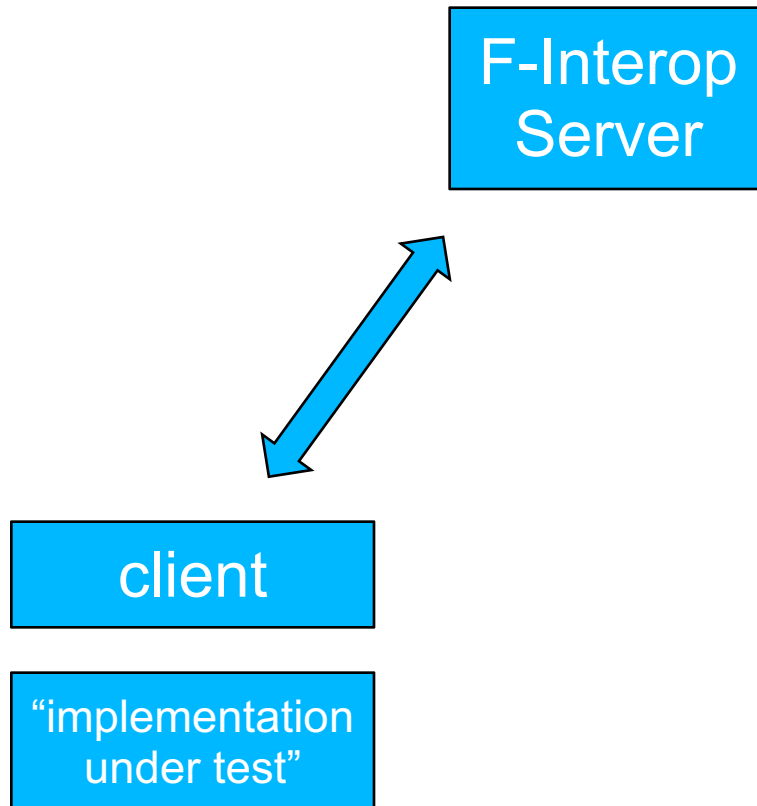
## ➤ SMEs and companies

- interop tests without needing to travel
- lower development cost
- faster development of standards-based products

→ more standards-based products



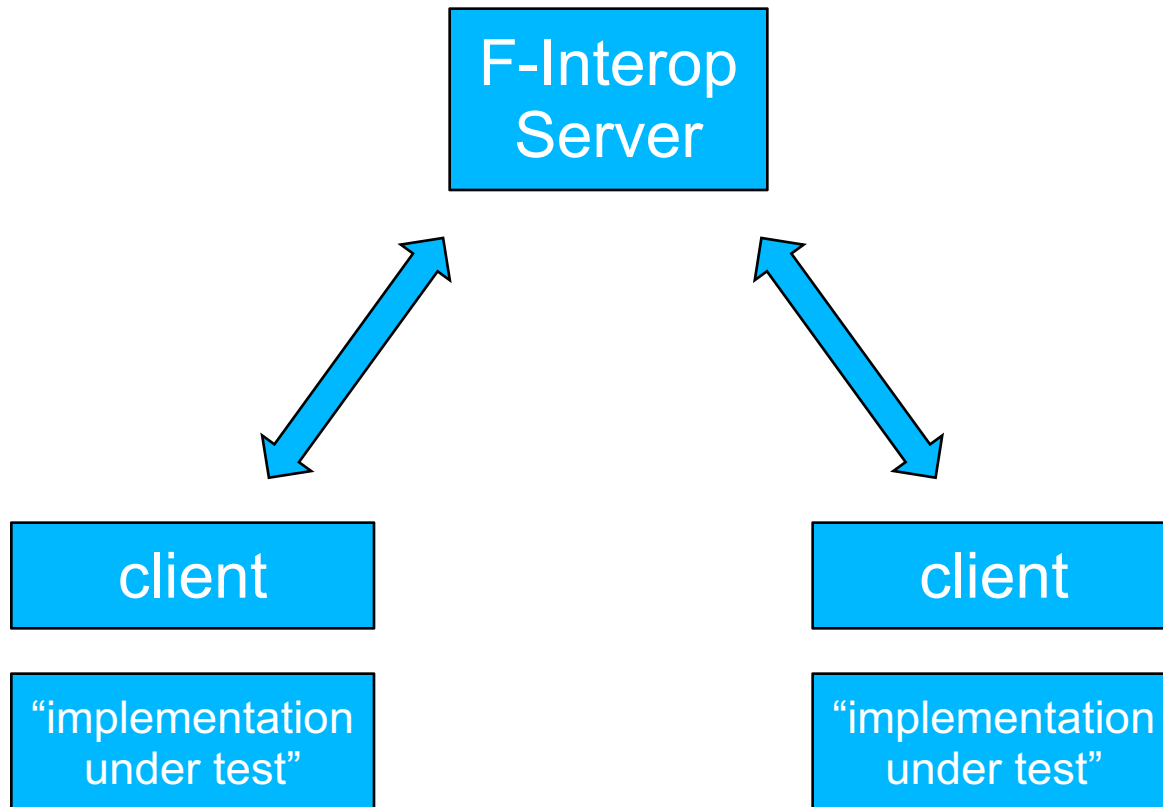
# Core Idea



Conformance Testing



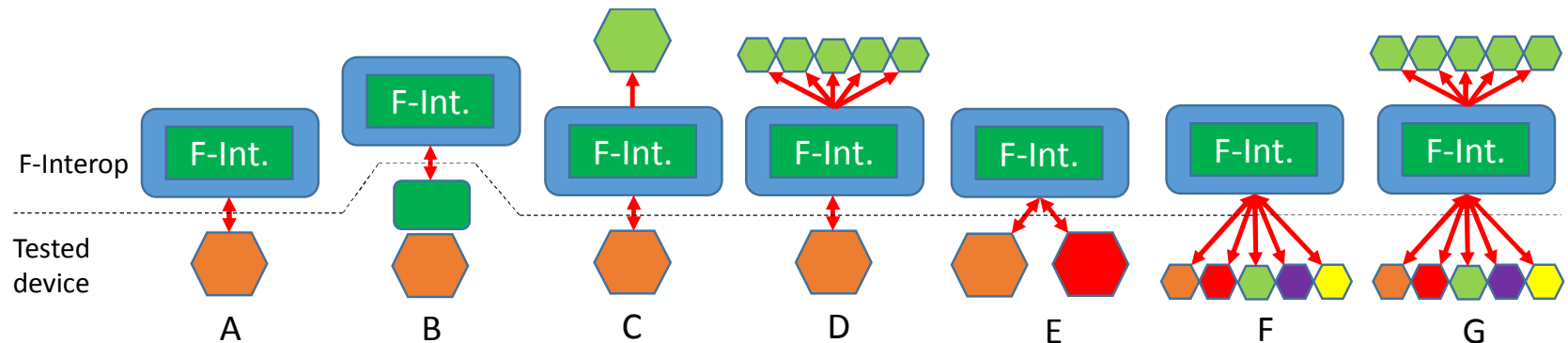
# Core Idea



Interop Testing



# Different Configurations



- A. Tested Device  $\leftrightarrow$  F-Interop test server
- B. Deported test with downloaded resource
- C. Remote interop with 2 participants
- D. Interop against testbed
- E. Local interop
- F. Remote interop with N participants
- G. Remote interop with N participants and testbeds



# Testbeds



32 testbeds, 4755 nodes

- **Fed4FIRE**  
([www.fed4fire.eu/testbeds](http://www.fed4fire.eu/testbeds))
  - 24 testbeds
  - ~1000 nodes
- **OneLab**  
([onelab.eu](http://onelab.eu))
  - Includes 6 IoT-lab deployments (including 2728 IoT nodes)
- **IoT lab**  
([www.iotlab.eu](http://www.iotlab.eu))





# Targeted Standards



- Initially standards of the IoT realm
  - CoAP
  - 6TiSCH
  - 6LoWPAN
- We take, as a starting point, the ETSI plugtests specifications and build an architecture that allows those to be done remotely
- **Contributions/extensions are expected by design**
  - Including:
    - oneM2M
    - **Web of Things (WoT)**





# **CoAP remote online interop testing**

## **A proof of concept**



# Example CoAP Test



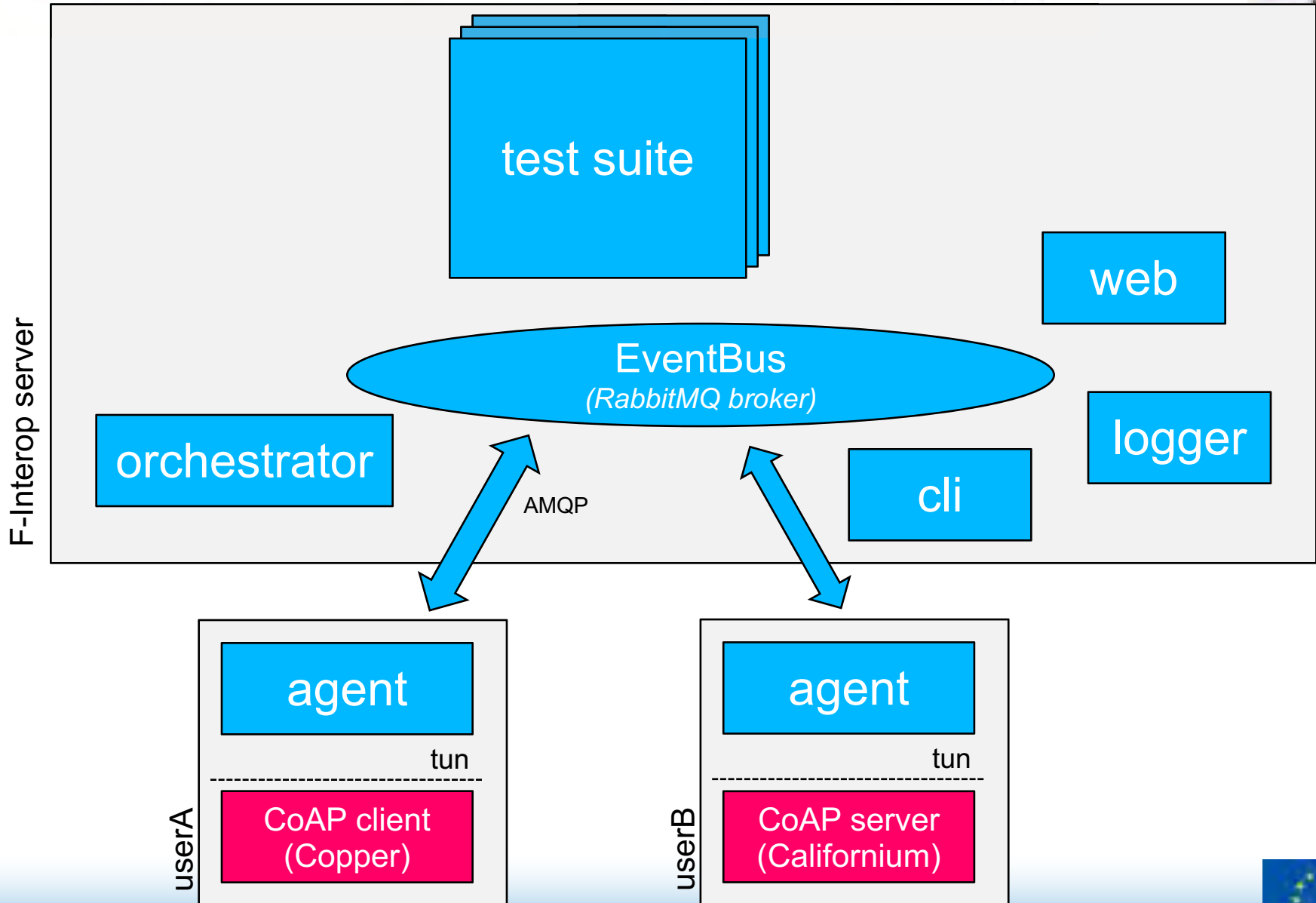
- From ETSI plugtest CoAP#4, IETF89 (London)

Interoperability Test Description

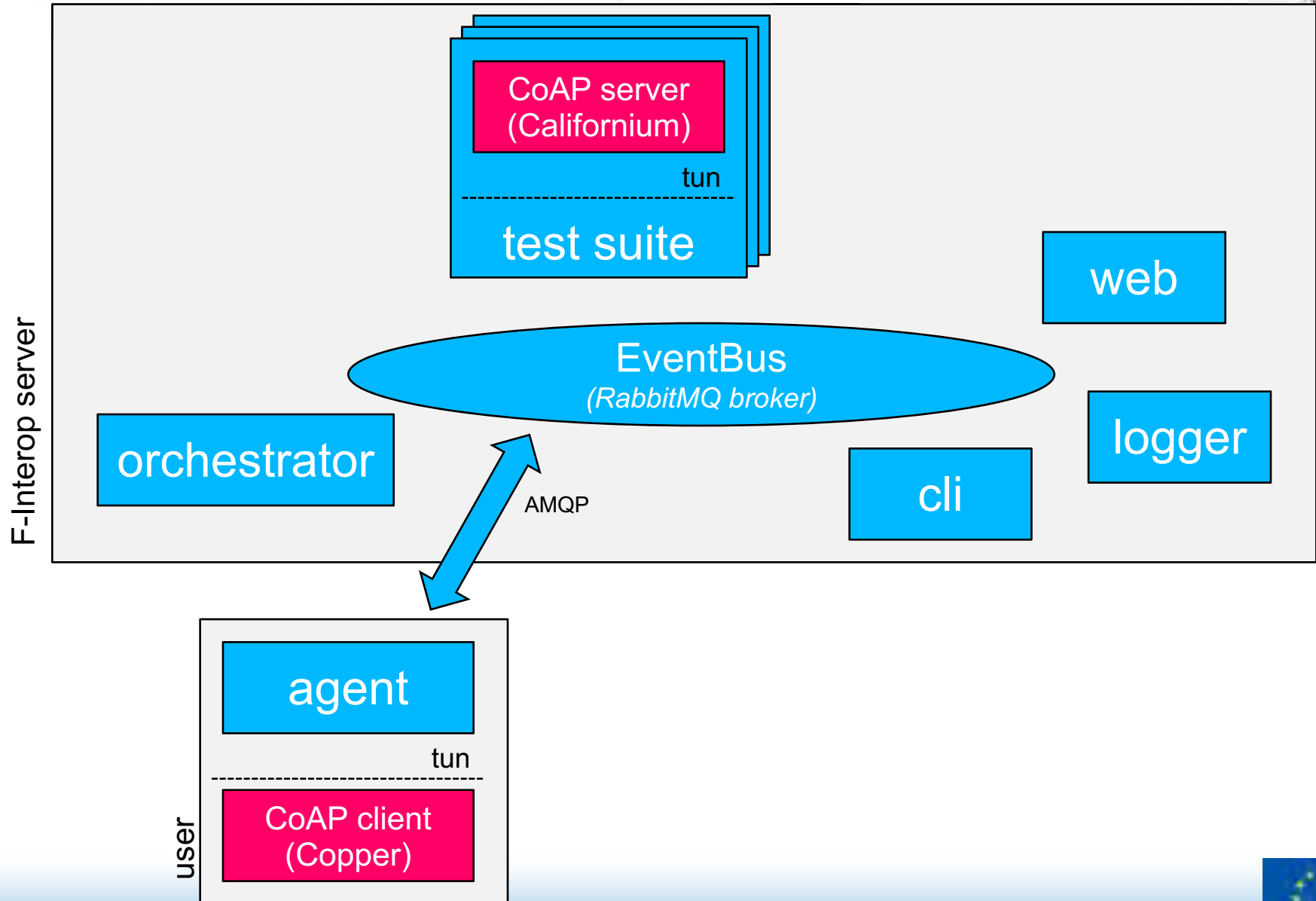
Identifier:	TD_COAP_CORE_03		
Objective:	Perform GET transaction (CON mode)		
Configuration:	CoAP_CFO_BASIC		
References:	[COAP] 5.8.1 1.2 2.1 2.2 3.1		
Pre-test conditions:	Server offers the resource first with resource content is not empty that handles GET with an arbitrary payload		
Test Sequence:	Step	Type	Description
	1	Simulate	Client is requested to send a GET request with <ul style="list-style-type: none"><li>• Type = 0 (CON)</li><li>• Code = 1 (GET)</li></ul>
	2	Check	The request sent by the client contains <ul style="list-style-type: none"><li>• Type=0 and Code=1</li><li>• Client-generated Message ID (= CMD)</li><li>• Client-generated Token (= CTOK)</li><li>• Uri-Path option "test"</li></ul>
	3	Check	Server sends response containing <ul style="list-style-type: none"><li>• Code = 2 (200 OK)</li><li>• Message ID = CMD, Token = CTOK</li><li>• Content-Format option</li><li>• Non-empty Payload</li></ul>
	4	Verify	Client displays the received information



# Base Architecture (CoAP interop)



# Base Architecture (CoAP interop demo)



# Download the Agent



The screenshot shows a web browser window with the URL `finterop.pars.inria.fr`. The page has a dark sidebar on the left and a white main content area on the right. The sidebar contains the text 'F-interop' and 'A platform for Interoperability testing'. Below this, there is a 'Home' section with a list of links: 'Download the agent' (circled in red), 'Documentation', and 'Contact'. The main content area features a heading 'IETF 96 demo' followed by a 'Goals' section with three bullet points and a 'Set up' section with two bullet points. The first bullet point in 'Set up' includes a link to `http://finterop.pars.inria.fr/shinikw/agent/agent.py`.



# Connect to the F-Interop Server



```
# sieben @ sieben-lincs ~ -/opt/interoperability/finterop_agent - git:develop x [1]
$ sudo python -m finterop.agent.agent connect --user bonjour --session bonjour --name client
Password: █
```



# Select and Start the Test Case



A screenshot of a web browser displaying the F-INTEROP interface. The browser's address bar shows "F-interop.pant.inria.fr". The page has a header with the F-INTEROP logo on the left and the Inria logo on the right. The main content area is divided into three columns. The left column, titled "Test cases", contains a list of test case references under the heading "Test case references". The first item is "TD\_COAP\_CORE\_01" with the description "Perform GET transaction (CON mode)". Below it are "TD\_COAP\_CORE\_02" (Perform DELETE transaction (CON mode)) and "TD\_COAP\_CORE\_03" (Perform PUT transaction (CON mode)). The middle column, titled "Console", features a green "Start Test Case" button and a message stating "28 test cases loaded" and "CoAP server URL: coap://[bob]:2/test". The right column, titled "No Frame Selected", contains a "No Frame" message and a "Frame list" section with the text "No test case selected for the moment".





# Send CoAP Packets



A screenshot of a CoAP client interface. The main display shows a response from [bbbb::2]:5683 (RTT: 115ms) with a status of 2.05 Content. The interface includes a top navigation bar with various CoAP methods (Discover, Ping, GET, POST, PUT, DELETE, Observe) and tabs for Payload, Text, Behavior, and Plus. On the left, there is a tree view of resource paths like .well-known, core, large, etc. The central area displays a table of options: Acknowledgment, 2.05 Content, Max-Age, and empty. Below this is a 'Payload (38)' section with filters for Incoming, Rendered, and Outgoing, showing details like Type: 0 (CON), Code: 1 (GET), and MID: 63915. On the right, there are sections for 'Debug Control' (with a Reset button and Token input), 'Request Options' (with an Accept field), 'Content-Format', and 'Observe' (with a use integer input).



# Finish Test Case

A screenshot of a web browser displaying the F-INTEROP interface. The browser's address bar shows "finterop.pam.inria.fr". The page header includes the F-INTEROP logo on the left and the Inria logo on the right. The main content area is divided into three sections: "Test cases", "Console", and "No Frame Selected".  
- The "Test cases" section contains a list of test case references: "TD\_COAP\_CORE\_01 Perform GET transaction (CON mode)", "TD\_COAP\_CORE\_02 Perform DELETE transaction (CON mode)", and "TD\_COAP\_CORE\_03 Perform PUT transaction (CON mode)".  
- The "Console" section shows "20 test cases loaded" and "CoAP server URL: coap://[bbbb-2]/test". A red oval highlights a red button labeled "Finish Test Case" located at the top of this section.  
- The "No Frame Selected" section contains the text "No frame selected for the moment." and a "Frame list" section with the text "No test case selected for the moment."



### Test cases

- TD\_COAP\_CORE\_01 Perform GET transaction (CON mode)
- TD\_COAP\_CORE\_02 Perform DELETE transaction (CON mode)
- TD\_COAP\_CORE\_03 Perform PUT transaction (CON mode)
- TD\_COAP\_CORE\_04 Perform POST transaction (CON mode)
- TD\_COAP\_CORE\_05 Perform GET transaction (NON mode)
- TD\_COAP\_CORE\_06 Perform DELETE transaction (NON mode)
- TD\_COAP\_CORE\_07 Perform PUT transaction (NON mode)
- TD\_COAP\_CORE\_08 Perform POST transaction (NON mode)**
- TD\_COAP\_CORE\_09 Perform GET transaction with separate response (CON mode, no payload)
- TD\_COAP\_CORE\_10 Perform GET transaction containing non-empty token (CON mode)
- TD\_COAP\_CORE\_11 Perform GET transaction containing non-empty token with a separate response (CON mode)
- TD\_COAP\_CORE\_12 Perform GET transaction using empty token (CON mode)
- TD\_COAP\_CORE\_13 Perform GET transaction containing several URI-Path options (CON mode)
- TD\_COAP\_CORE\_14

### Console

TD\_COAP\_CORE\_07

Case the world fail

System failure

4.3

More informations

```

127.0.0.1 | CoAP [NON 127.0.0.1] PUT /test/
msg | match: CoAPType | token: () | tid |
msgLen
CoAPType: CoAPTypeControlFailed()
payload: CoAPPayloadCoAPMessage()
get: request: CoAPMessageControlFailed()
127.0.0.1 | CoAP [NON 127.0.0.1]
Change: + [get] match: CoAPType |
token: AnyStr: () | msgLen: ()

```

This case TD\_COAP\_CORE\_07 failed, press the Finish button when completed

TD\_COAP\_CORE\_08

Case the world pass

System failure

2

More informations

TD\_COAP\_CORE\_09

Case the world pass

System failure

1.3

More informations

TD\_COAP\_CORE\_06

Case the world pass

System failure

3

More informations

### Analyse TC : TD\_COAP\_CORE\_07

Frame n°4

Coap

Version: 1  
Type: 1  
Token length: 2  
Code: 1  
MessageID: 0x0000  
Token: 1/1/0/0  
Options:  
CoAPOptionsPath:  
  Delta: 11  
  Length: 4  
  Value: test

Payload: 0x00

UDP

IPv4

Web socket

Frame list

- 1 [127.0.0.1 -> 127.0.0.1] [CoAP [NON] -> 0x0000]
- 2 [127.0.0.1 -> 127.0.0.1] [CoAP [NON] -> test]
- 3 [127.0.0.1 -> 127.0.0.1] [CoAP [NON] Control Message]
- 4 [127.0.0.1 -> 127.0.0.1] [CoAP [NON] 127.0.0.1] PUT /test/
- 5 [127.0.0.1 -> 127.0.0.1] [CoAP [NON] 127.0.0.1] Change

# Under the Hood: What's a test?



```
testcase:
  testcase_id: TD_COAP_CORE_01_v01
  uri : http://f-interop.paris.inria.fr/tests/TD_COAP_CORE_01_v01
  configuration: CoAP_configuration_BASIC
  objectives: Perform GET transaction(CON mode)
  pre_conditions: Server offers the resource /test with resource content is not empty that handles GET with an arbitrary payload
  references: '[COAP] 3.8.1, 3.2, 3.7, 3.7, 3.3'
  sequence:
    - step_id: 'TD_COAP_CORE_01_v01_step_01'
      type: stimuli
      actor: coap_client
      description:
        - Client is requested to send a GET request with
        - Type = 0(CON)
        - Code = 1(GET)

    - step_id: TD_COAP_CORE_01_v01_step_02
      type: check
      description:
        - The request sent by the client contains
        - Type=0 and Code=1
        - Client-generated Message ID(\u2794 CHID)
        - Client-generated Token(\u2794 CTOK)
        - Uri-Path option "/test"

    - step_id: TD_COAP_CORE_01_v01_step_03
      type: check
      description:
        - Server sends response containing
        - Code = 2.05(Content)
        - Message ID = CHID, Token = CTOK
        - Content-format option
        - Non-empty Payload

    - step_id: TD_COAP_CORE_01_v01_step_04
      type: verify
      actor: coap_client
      description:
        - Client displays the received information
```



# Under the Hood: What's a test?

```
#!/usr/bin/env python3

from ttpROTO.ts_coap.common import CoAPTestcase
from ttpROTO.ts_coap.templates import *

class TD_COAP_CORE_B1 (CoAPTestcase):

    def run (self):

        # match stimuli
        self.match_coap ("client", CoAP (type="con", code="get",
                                         opt = self.uri ("/test")))
        CHID = self.frame.coap["mid"]
        CTOK = self.frame.coap["tok"]

        # match step 2
        self.next()
        if self.match_coap ("server", CoAP (
            code = 2.05,
            mid = CHID,
            tok = CTOK,
            pl = Not(b""),
        )):

            # match step 3
            self.match_coap ("server", CoAP (
                opt = Opt (CoAPOptionContentFormat()),
            ), "fail")
```

# Next Milestones



- July 2016
  - minimal CoAP interop testing (done) -> see demo
- November 2016
  - Functional platform available
  - CoAP CORE interop tests
- March 2017
  - 6TiSCH support, update at IETF98
  - CoAP interop test (advanced version)
- July 2017
  - Use at 6TiSCH/6lo plugtests
  - **minimal WoT interop testing**



# WoT interop test case example

## Properties

Identifier	TC_WOT_BASE_01
Objective	Read Boolean Property
References	<a href="#">3.2.3.1 Property</a> , <a href="#">3.2.4.1 Simple Data</a>
Pre-test conditions	Exposing Thing provides boolean Property
<b>Test sequence</b>	
1. Stimulus	Consuming Thing sends <code>retrieve</code> to Property
2. Check	Consuming Thing sends <ul style="list-style-type: none"><li>- protocol operation bound to <code>retrieve</code></li><li>- no payload</li><li>- to Property URI</li></ul>
3. Check	Exposing Thing sends <ul style="list-style-type: none"><li>- positive response code</li><li>- payload formatted according to TD</li></ul>
4. Verify	Consuming Thing displays read value

Source: <https://github.com/w3c/wot/blob/master/plugfest/2016-beijing/plugfest-test-cases-beijing-2016.md>



# How the WoT community can help?



- **Contributors:**
  - Help us extending F-Interop for interop in WoT context
  - List requirements, identify key priority WoT standards
  - Develop test suites for (new) standards
  - Provide feedback on architecture and choices
- **Users:**
  - Use F-Interop for remote interop events/plugtests







# Open Call



# Open Call Categories



- **New testing tools** to extend capabilities of F-Interop
- **New test descriptions** to test conformance and interoperability of other standards
- **SME F-Interop assessment reports:** SME device Interop tests to test F-Interop platform
- **Plugtest Events:** Third parties selected to conduct 3 remote online plugtest events



# Supported Activities & Budget

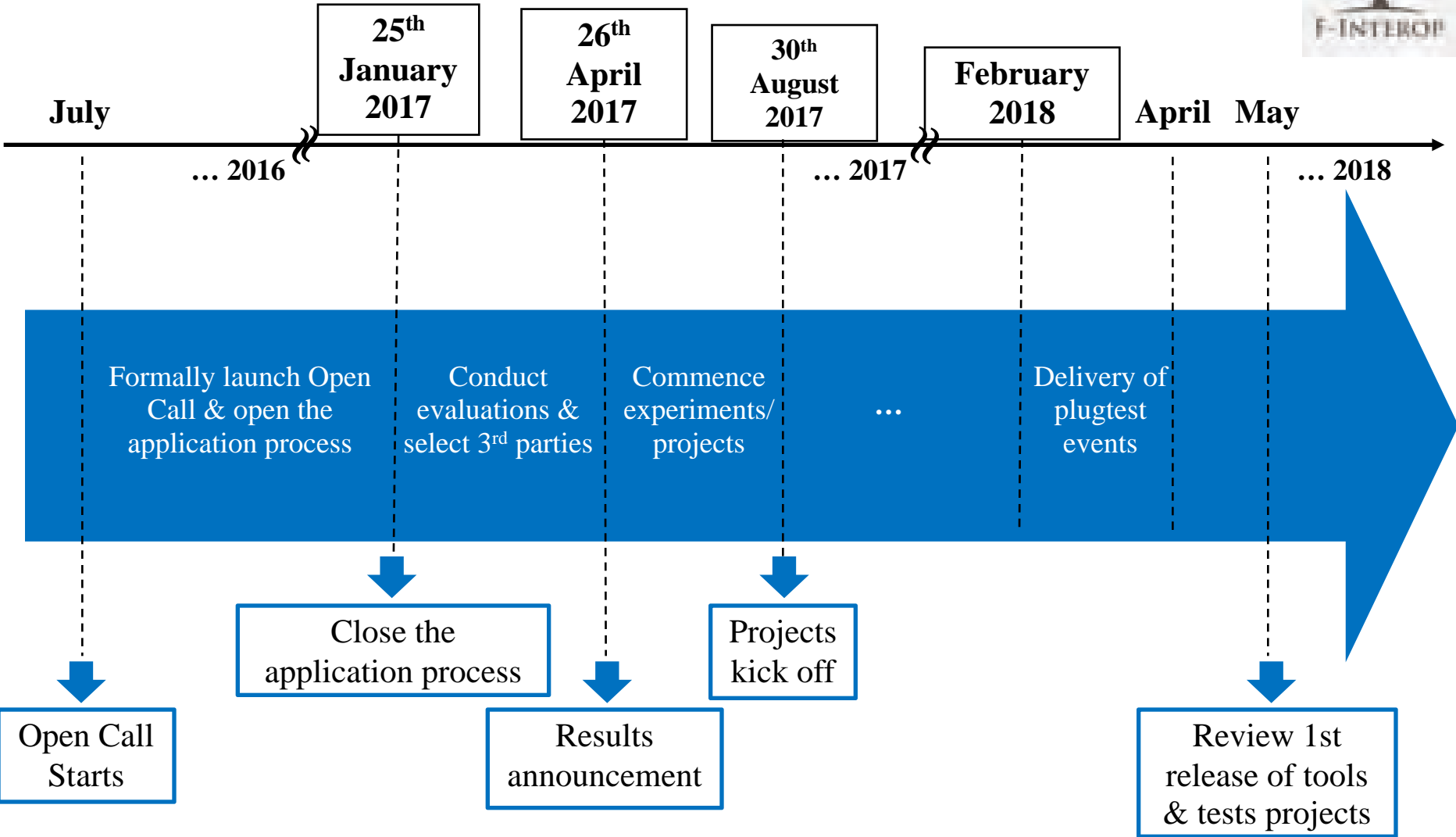


610k for 19 projects

List of Categories	Grants	Award
New F-Interop tools extensions	3	100 000
<b>New interop test descriptions</b>	3	60 000
SME devices F-Interop tests and report	10	10 000
<b>Plugtest Events</b>	3	10 000



# Important Dates



# How to apply?



- Template for the proposal
- Guide for Applicants
- Standard Industrial Experiment Contract
- Open Call Terms and Conditions
- **Submission Portal**

**<http://www.f-interop.eu/index.php/open-call>**





Thank you for your attention

Open-call: <http://www.f-interop.eu/index.php/open-call>

Please, feel free to contact us directly or later via:  
Federico.Sismondi@inria.fr, Cesar.Viho@irisa.fr

