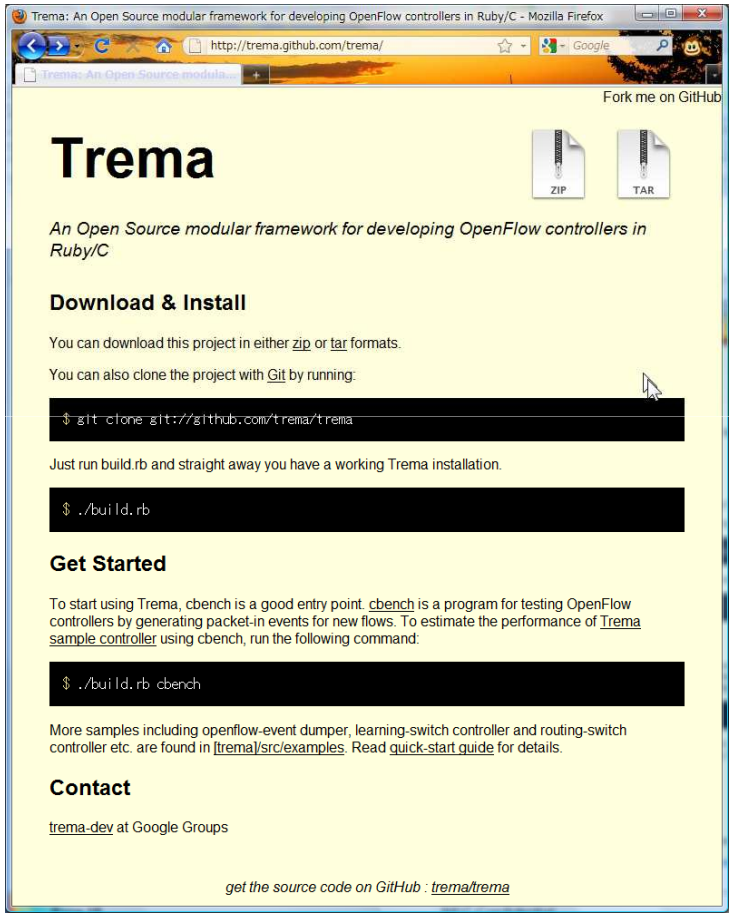
Trema Overview

HIDEyuki Shimonishi

Aug. 4, 2011

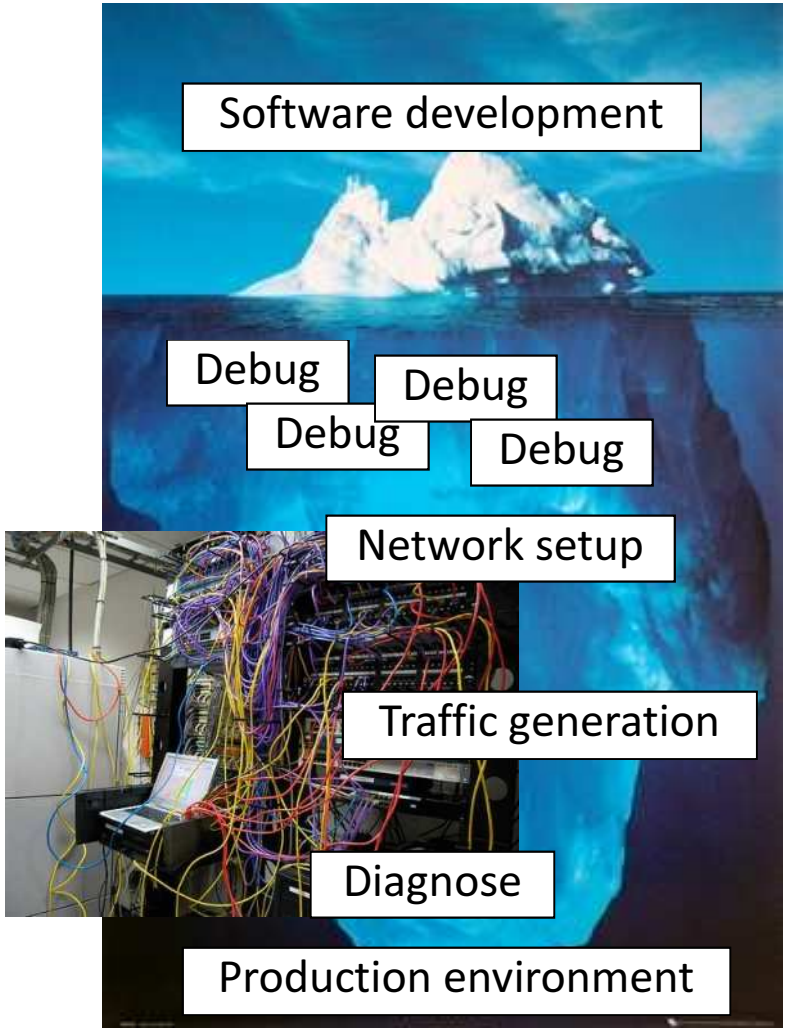
Open Source OpenFlow Controller: Trema

- Free software (GPLv2)
  - Repository https://github.com/trema/
  - ML trema-dev@googlegroups.com
- A software platform for OpenFlow researchers and developers
  - Not a production controller itself
  - Multi-process modular architecture for extensibility
- Integrated developing environment
  - Seamless integration of controller and network environment for testing and debugging
  - TDD (Test Driven Development) framework



Background

OpenFlow iceberg



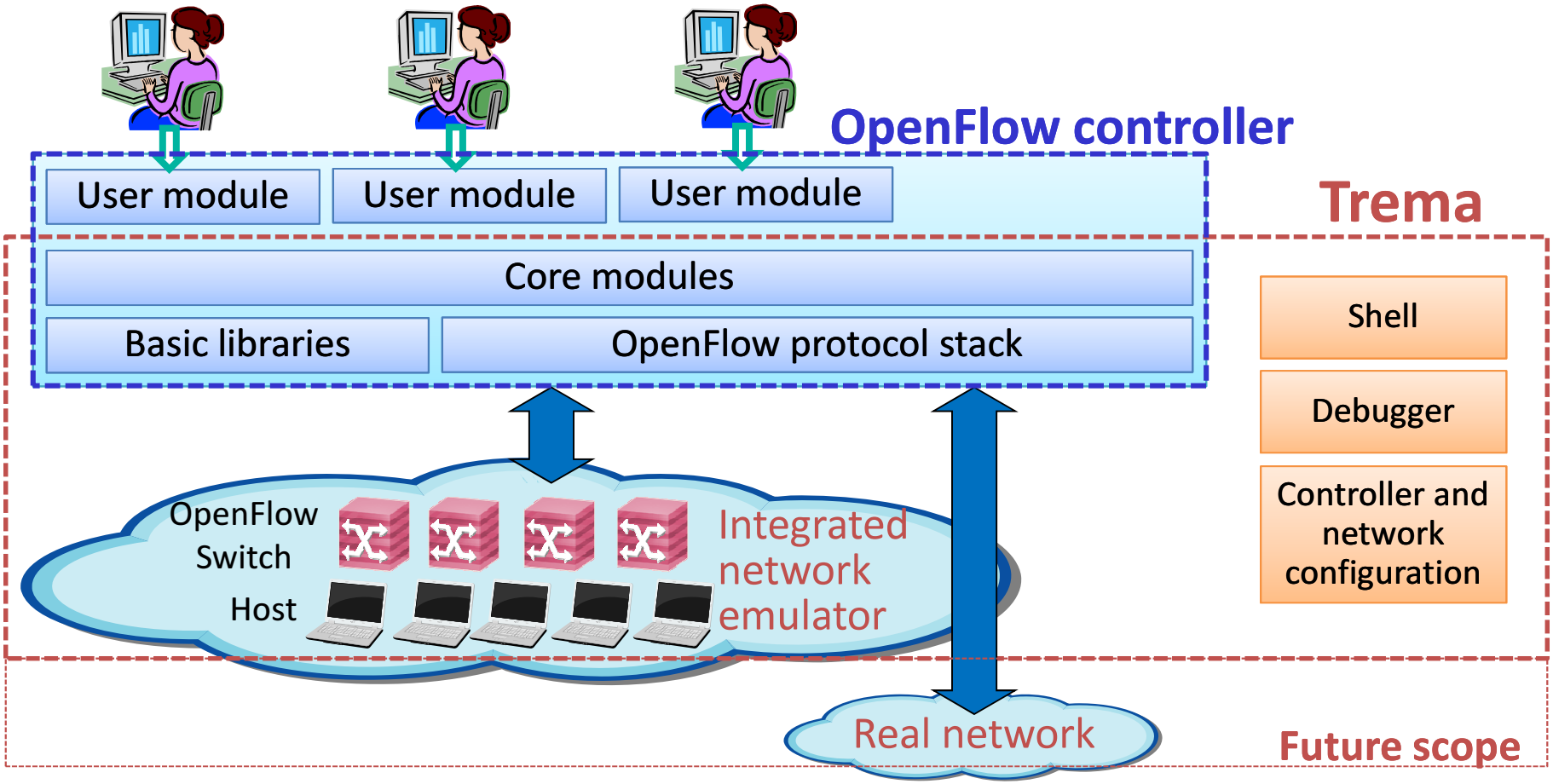
Scope of Trema

Trema is an OpenFlow platform for entire development process (like Rails)

- ***Shorter development cycle***
- ***Reduce labor cost***
- ***More and more research outputs :-)***

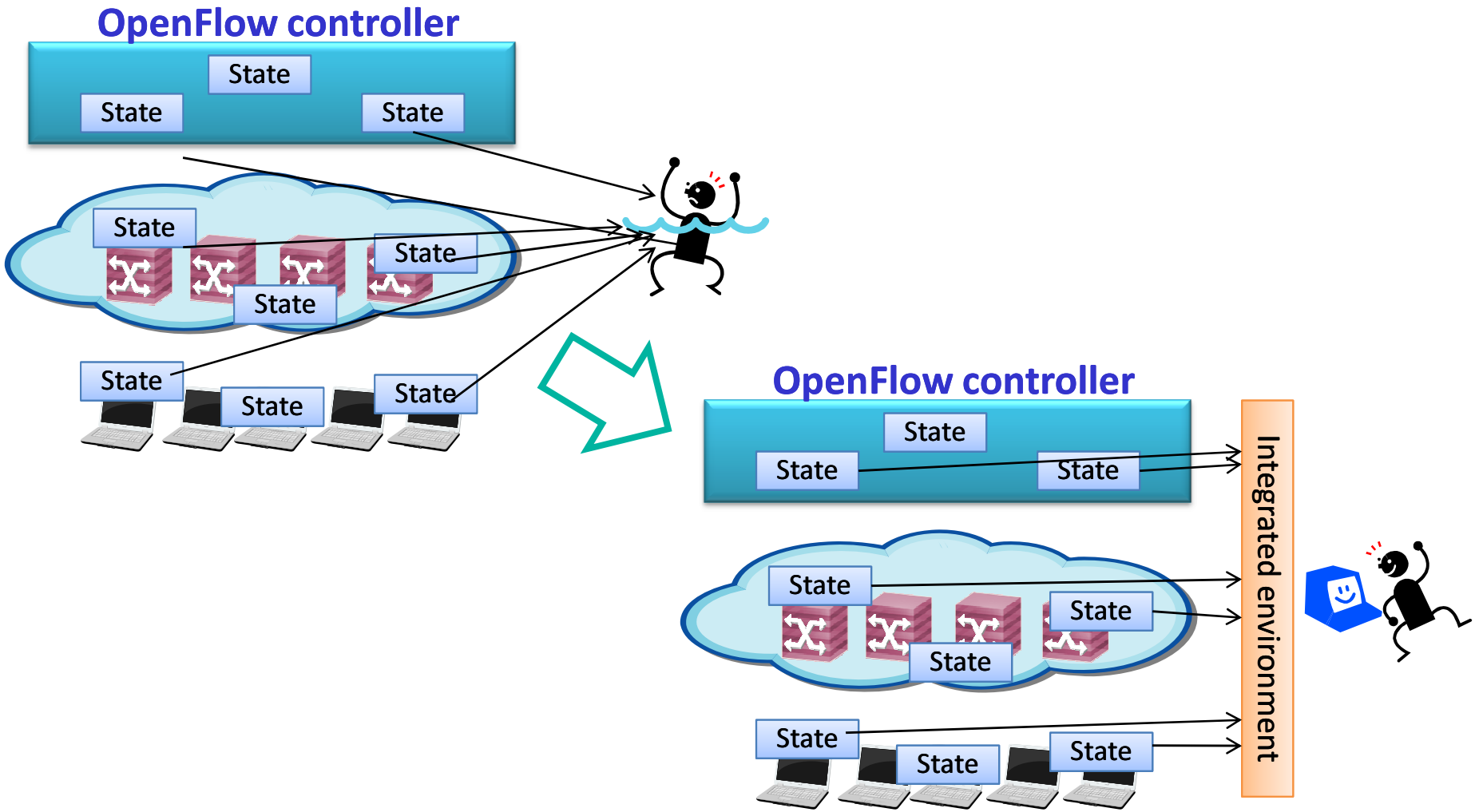
What is “Trema”

- Trema @ https://github.com/trema/trema
- User modules @ https://github.com/trema/apps

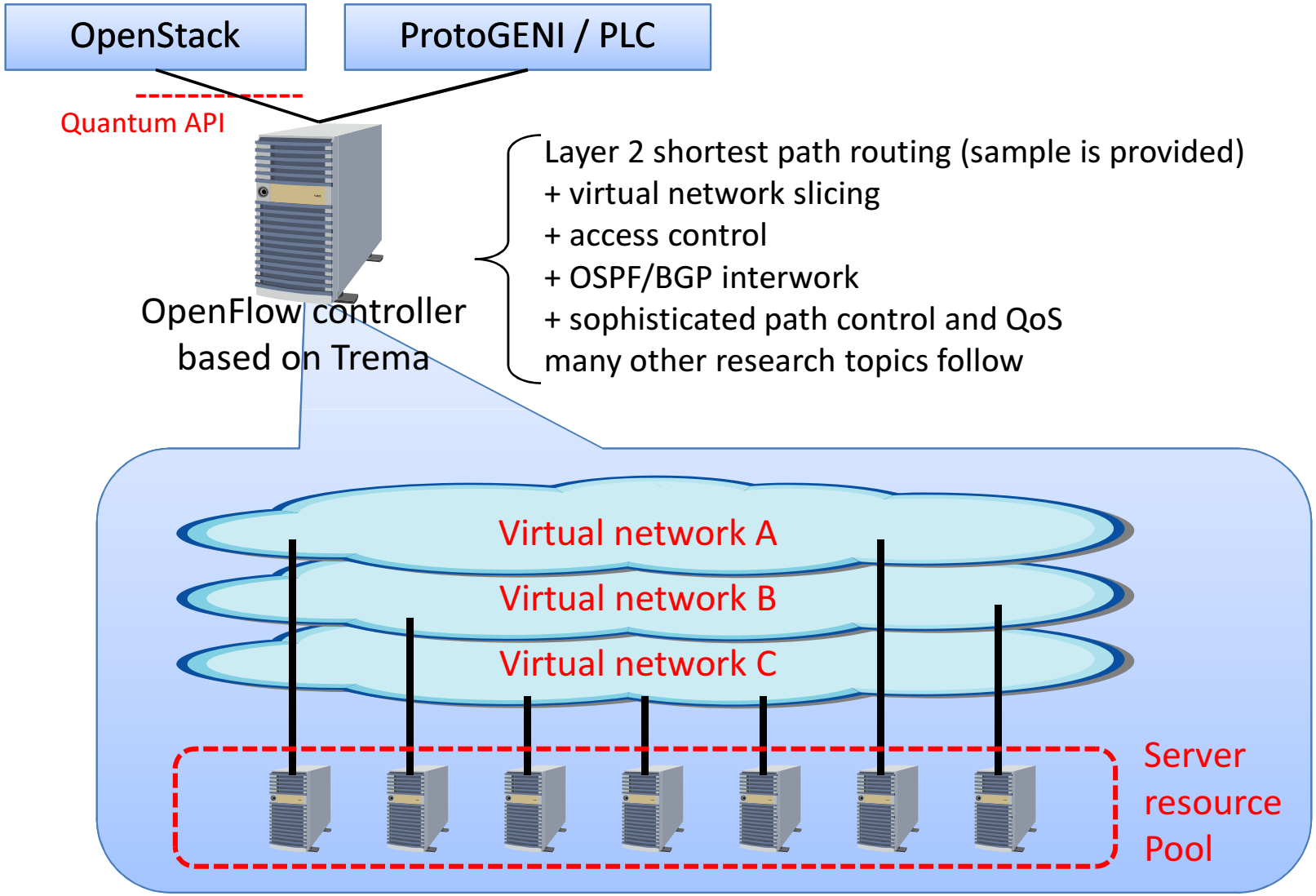


Why integrated ?

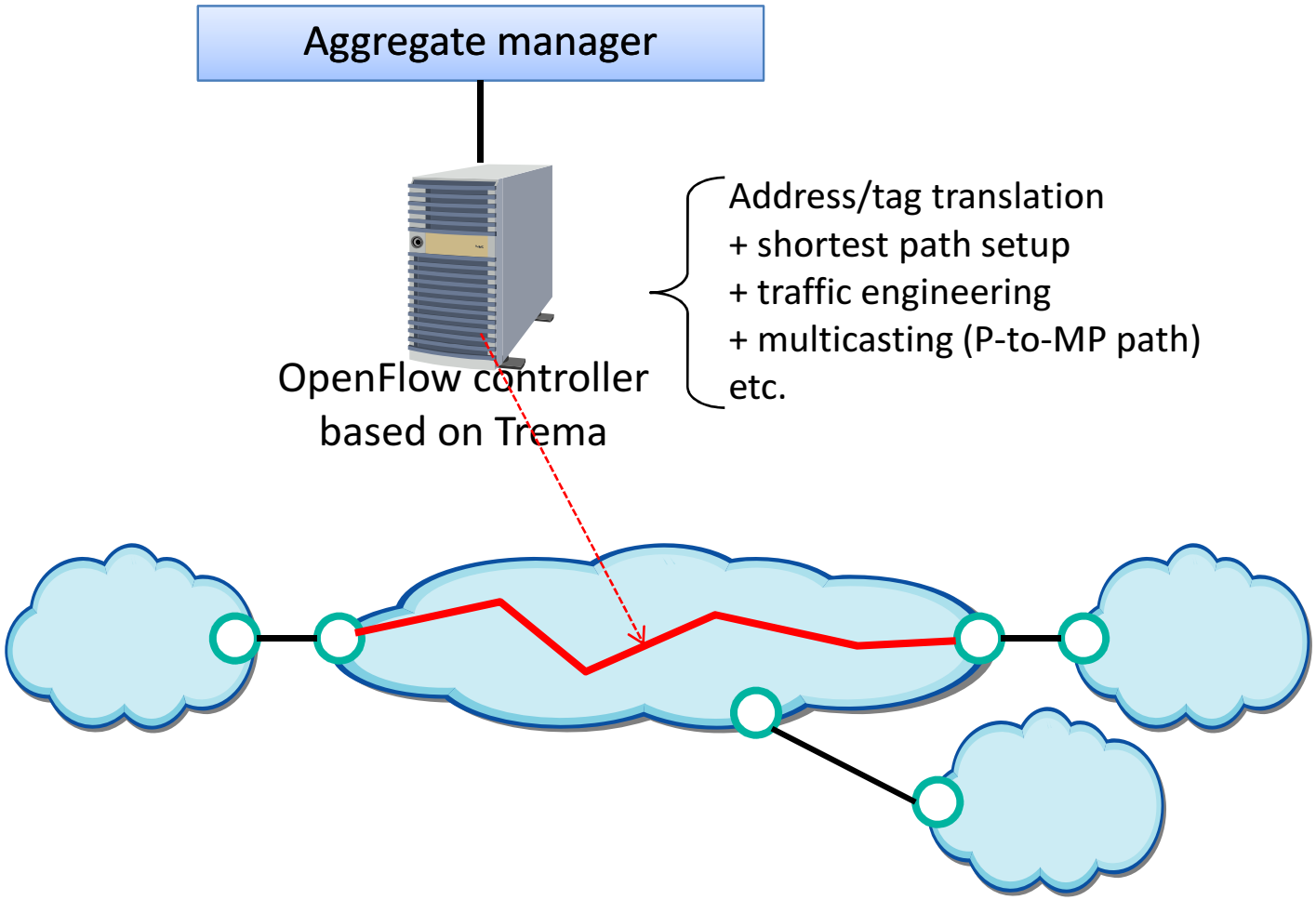
- Network programming is essentially distributed programming



A use case – automatic network slice creation from computing resource slice assignments

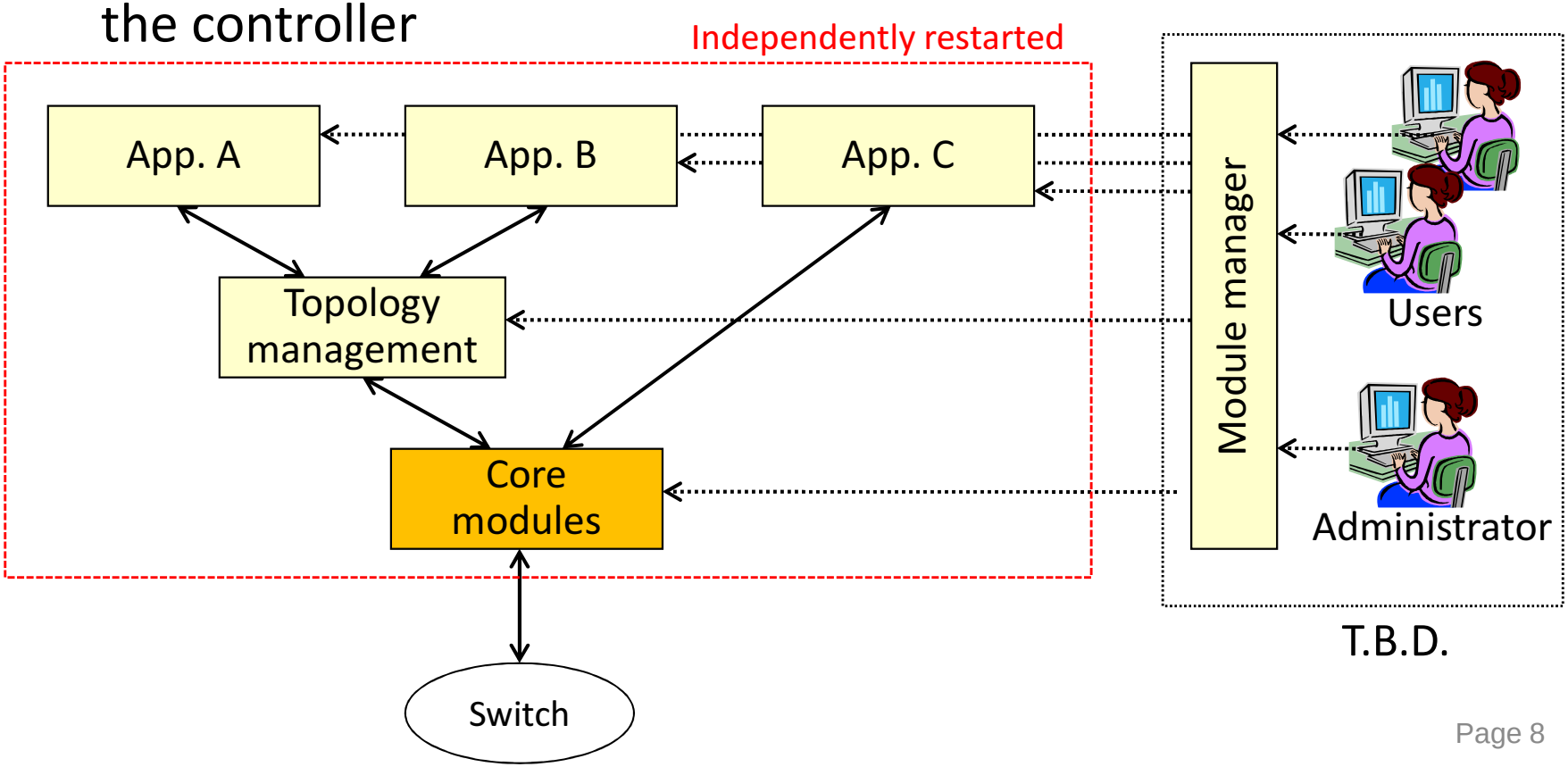


A use case – stitching



A use case – collaboration on a controller

- Protecting a controller from someone installing unstable modules...
  - Isolate modules as independent processes
- Dynamically changing code (without having to stop/start) in the controller



Documents

- Web: http://trema.github.com/trema/
- Wiki: https://github.com/trema/trema/wiki

- Manuals
  - Install: https://raw.github.com/trema/trema/master/INSTALL
  - Quick start: https://github.com/trema/trema/wiki/Quick-Start
  - C API: not available now (3Q/2011 release)
  - Ruby API: http://rubydoc.info/github/trema/trema/master/frames (Q3/2011)

- Tutorial
  - https://github.com/trema/trema/wiki/Trema-tutorial

Current status and work items

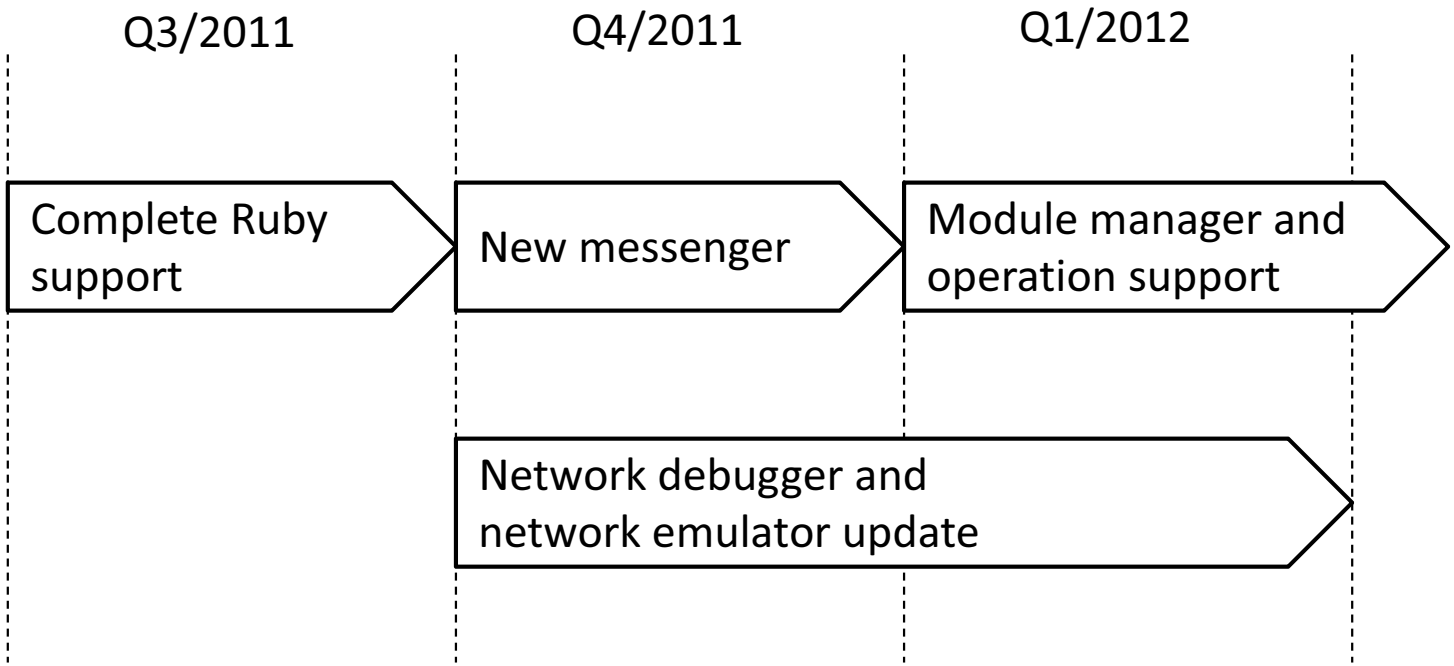
- Core functions
  - Core modules
    - Switch manager, switch daemon, packet_in filter, etc…
  - OpenFlow application interface (OpenFlow 1.0.0 compliant)
    - C APIs (fully compliant with the specification)
    - Ruby bindings (work in progress. planned to be fully supported by Q3/2011)
  - Libraries
    - Basic data structures, packet parser, logging, hash table, linked list, timer, etc…
  - Messenger
    - Point-to-point messaging among processes
    - Group messaging among hosts [T.B.D.]

Current status and work items –cont'd

- Development and test ➔ Tutorial part 3
  - Test driven development framework (Ruby RSpec) [Q3/2011]
  - Automated test environment [available and increasing coverage]
    - automatic build, unit test, acceptance test, etc…
- Test and operation ➔ Tutorial part 2
  - Network DSL and shell [available but updated frequently]
    - Describe and manage network and controller configurations
  - Module manager [T.B.D]
- Debug
  - TremaShark
  - Network debugger [planned]
- Network emulator
  - Pseudo host [available and more functions planned]
  - Virtual switch [available and more functions planned]

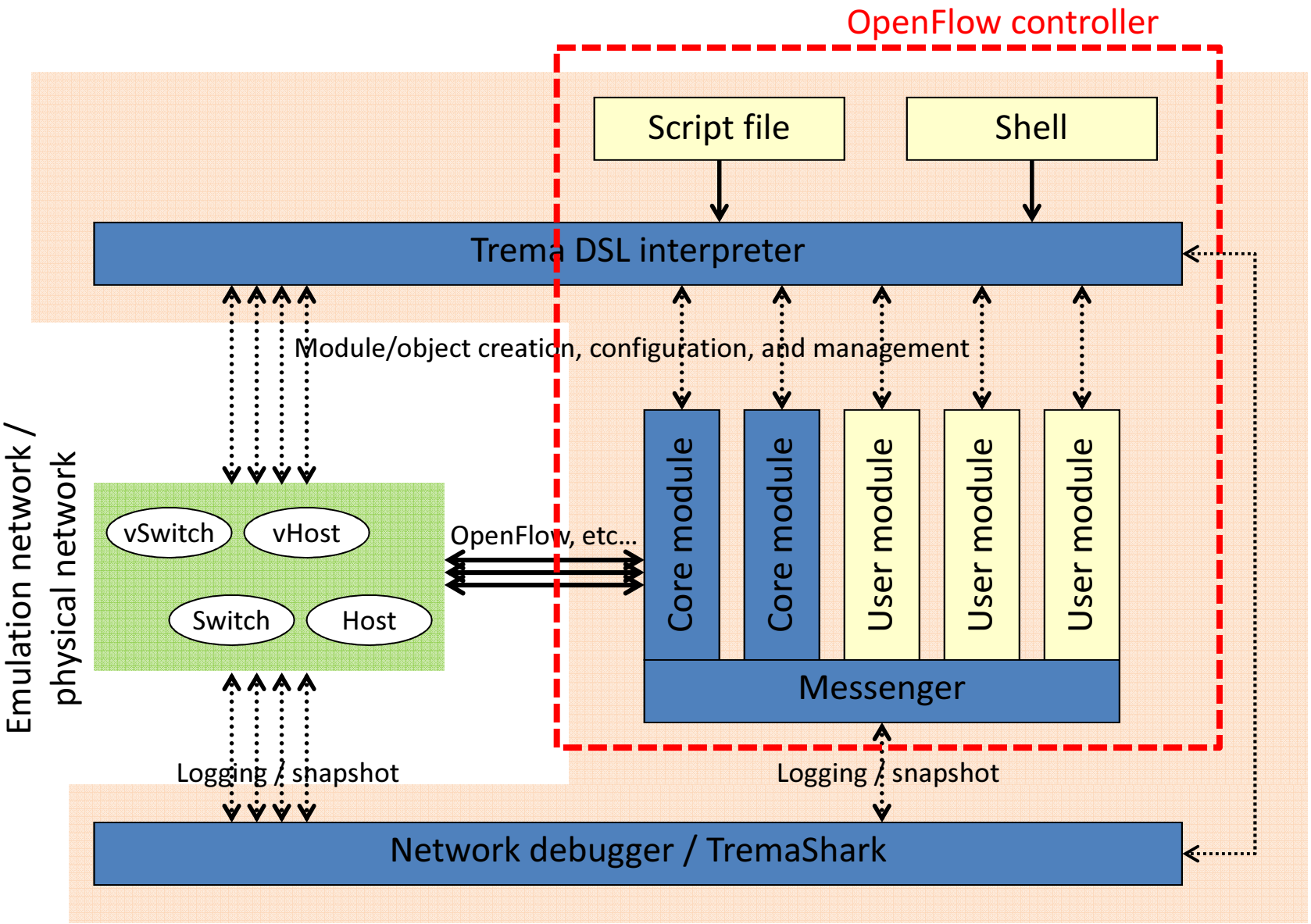
Roadmap

- Q3/2011 : Ruby support fully available
- Q1/2012 : Full set of develop environment available



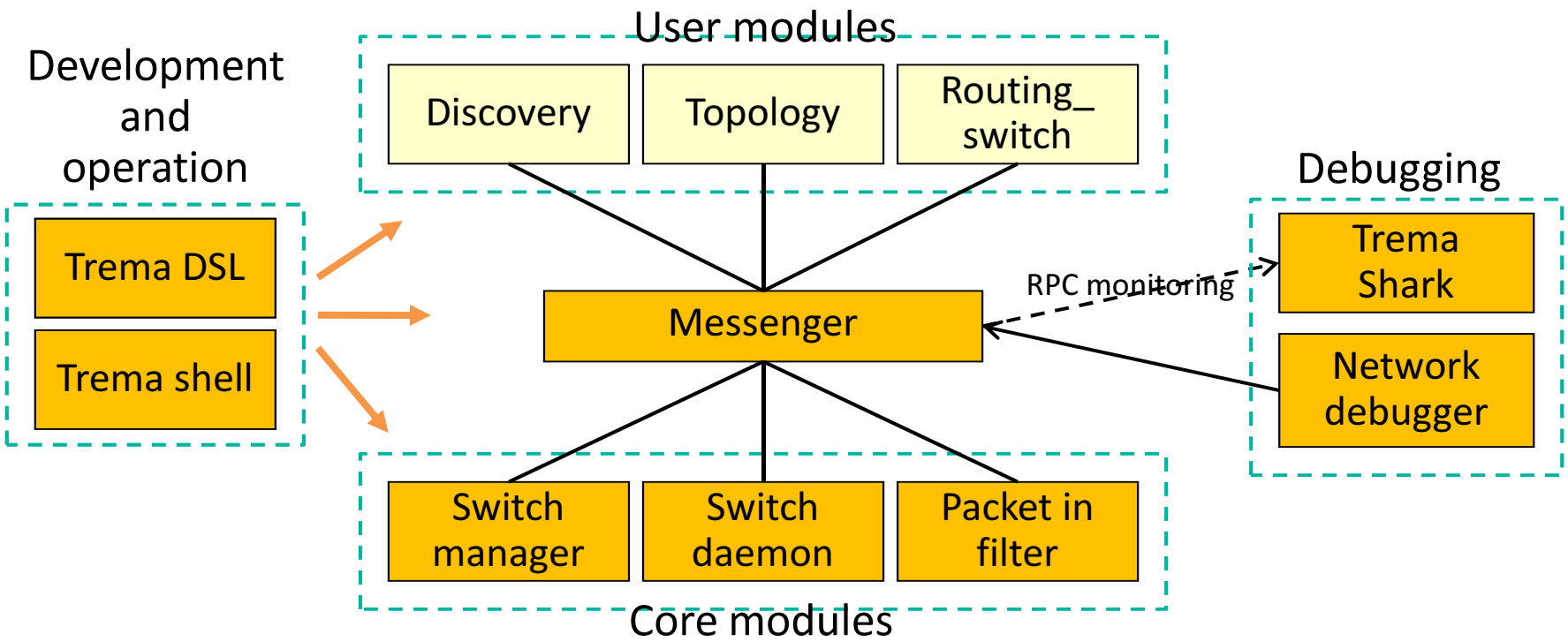
A bit more internal structure

Architecture overview



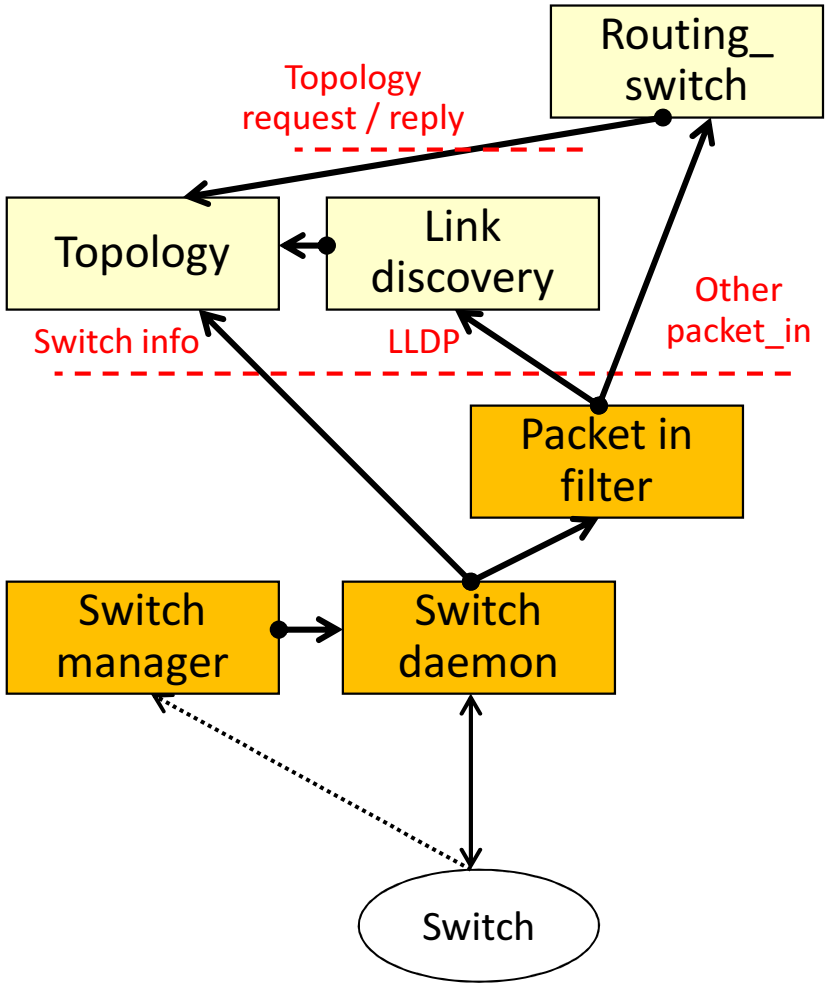
Multi-process model

- Functional modules loosely coupled via messenger
  - User modules, core modules = process
  - Messenger = messaging among modules on different processes/hosts
  - Protecting a controller from someone installing unstable modules...
  - Dynamically changing code (without having to stop) in the controller



Multi-process model – example

- trema/src/examples/routing_switch



Configuration file
- Starting-up application processes
- Module configurations (filters)
- [Configure message routing (if not specified, use default service name)]

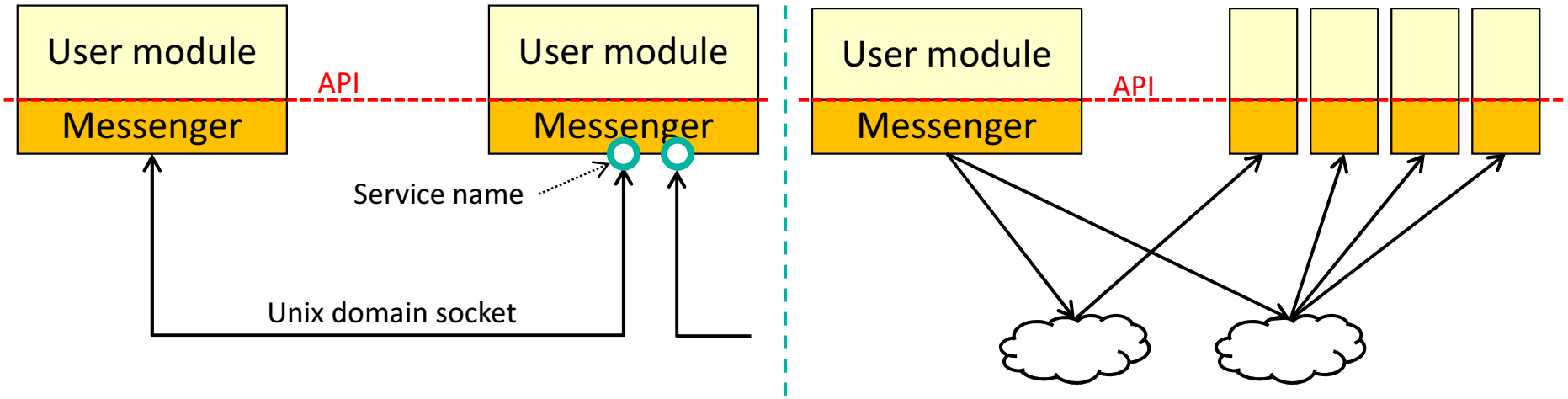
```
app {
  path "./objects/examples/topology/topology"
}
app {
  path "./objects/examples/topology/topology_discovery"
}
app {
  path "./objects/examples/routing_switch/routing_switch"
}

event :port_status => "topology", packet_in =>
"filter", :state_notify => "topology"

filter :lldp => "topology_discovery", :packet_in =>
“routing_switch"
```

Messenger

- Messaging among modules on different processes/hosts
- Messenger API
  - SEND: send_message(service_name, tag, data, data_length)
  - RECEIVE: add_message_received_callback(service_name, call_back_function)



Point-to-point messaging within a host (current)

Pub-sub messaging among multiple hosts (planned)

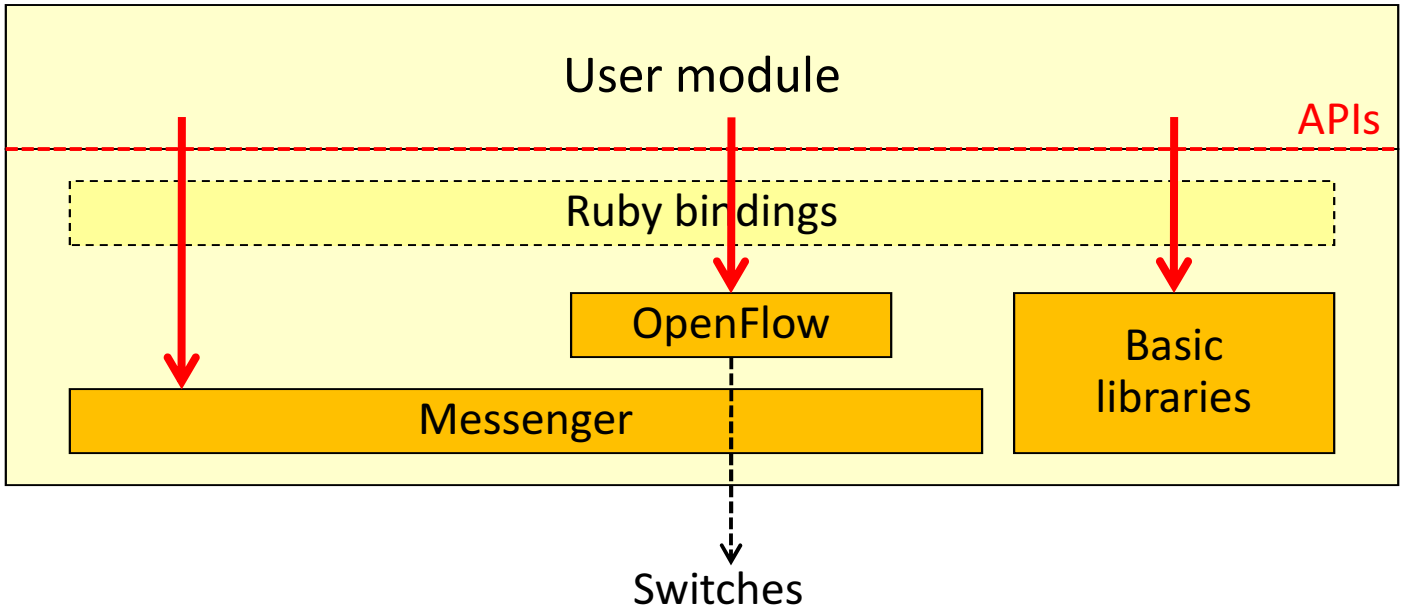
Language support

- User modules can be written in C or Ruby
  - Provides C libraries and Ruby bindings
- Advantages to use Ruby
  - Easy to write
  - Integrated with test framework (sophisticated tests using built-in network emulator)
  - But maybe slower than C…
- Mixture of C and Ruby modules
  - Faster modules in C and slower modules in Ruby
  - Prototype and test in Ruby first, then rewrite to C

Internal APIs

- OpenFlow 1.0.0 (C and Ruby)
- Basic libraries (C and Ruby)
  - packet parser*1, logging, hash table*2, linked list*2, timer, etc.
- Messenger (C and Ruby*1)

*1 ruby in progress
*2 use Ruby built-in



Configurations and external interfaces

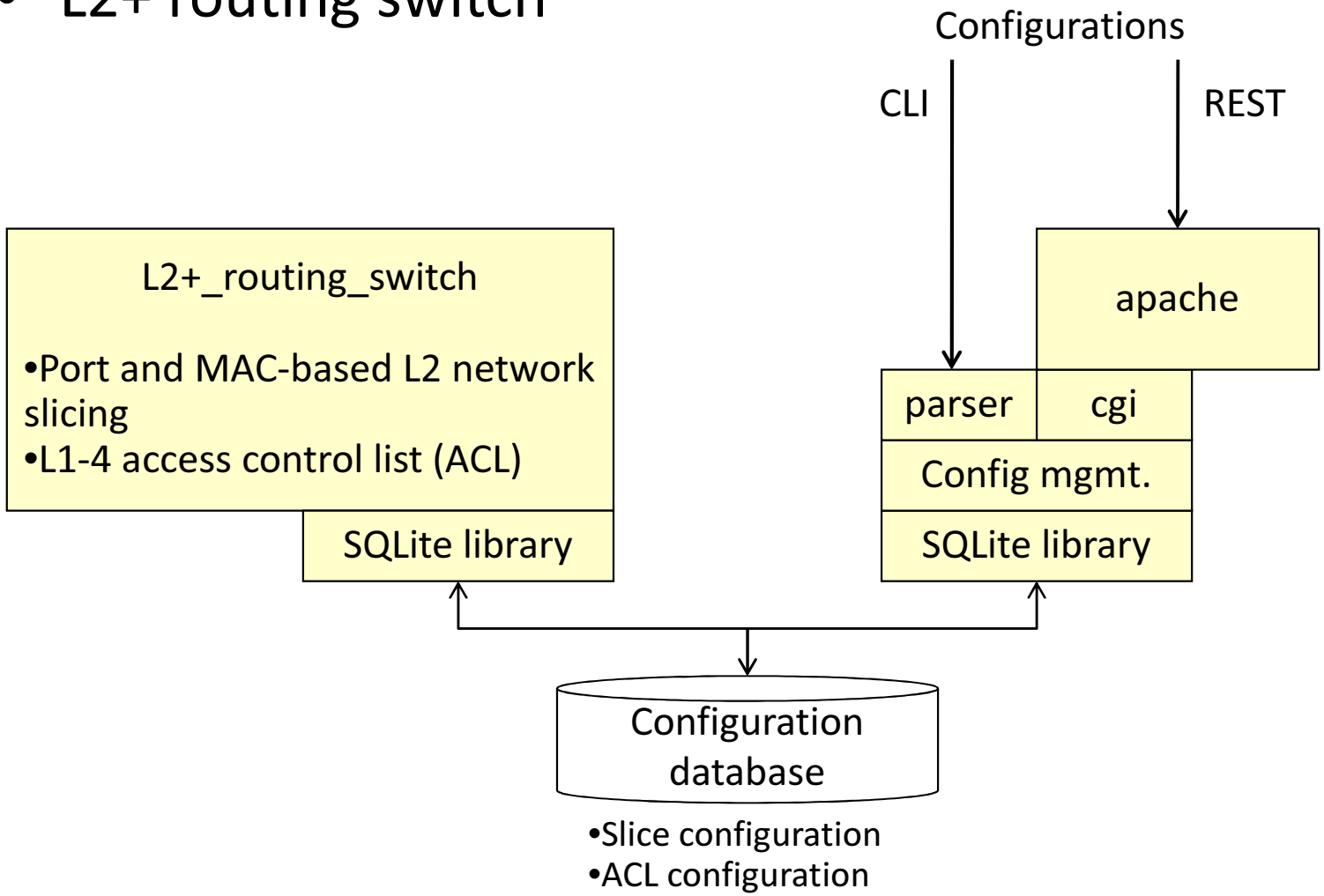
- Command line options
  - Configurations described in configuration file are passed to each module as command line options

- Configurations
  - CLI, REST
- External interfaces
  - REST
- Persistent database
  - SQLite

No official framework is provided yet (see next slide)

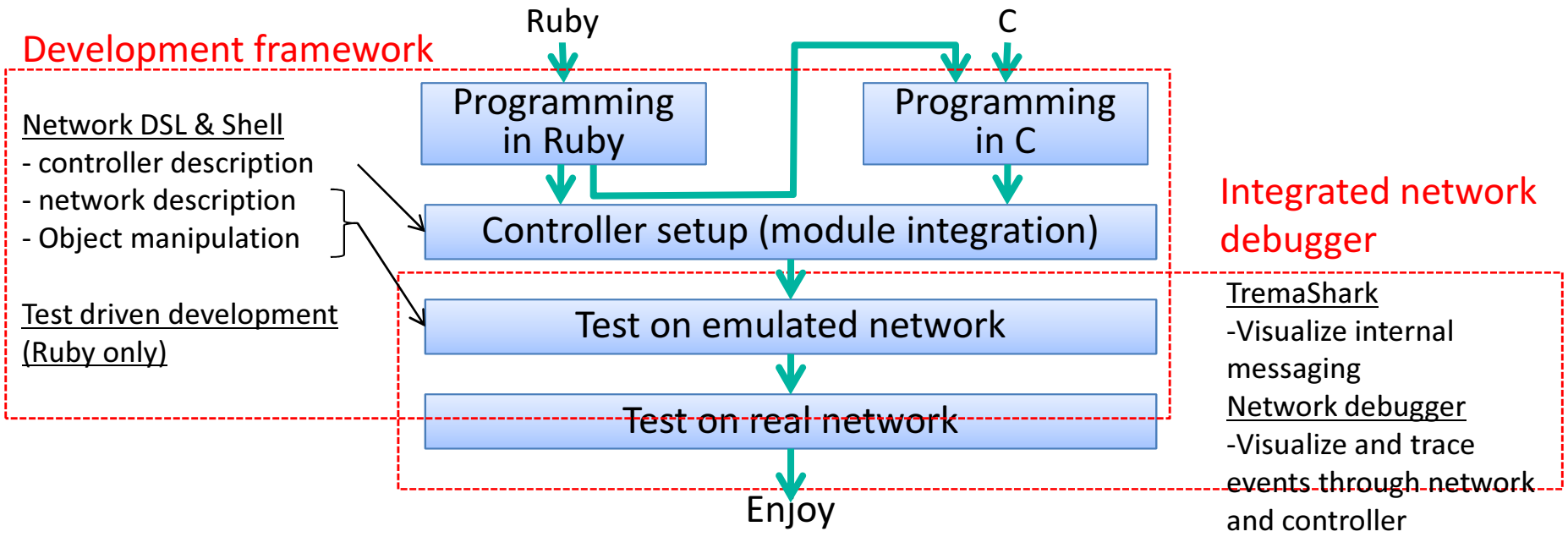
Configurations and external interfaces - example

- L2+ routing switch



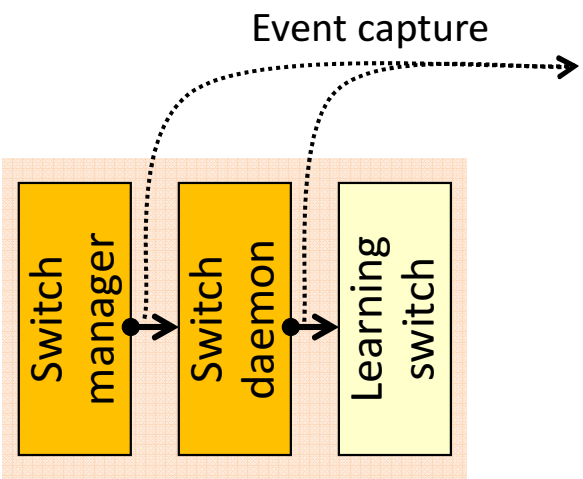
Development, test and debug [work-in-progress]

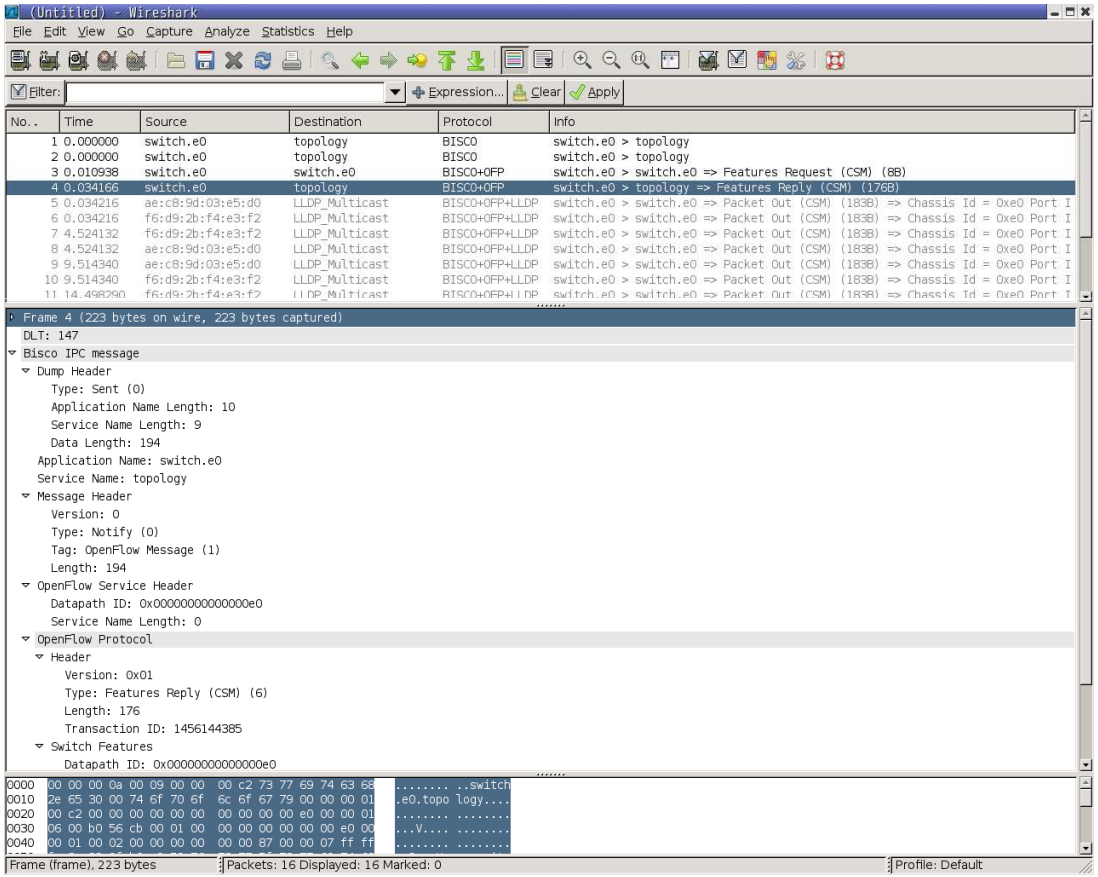
- TDD (Test Driven Development)
  - Specification (test scenario) and implementation are integrated ➔ allows frequent implementation changes
  - Test scenario involves both controller and network, thus, their configurations and operations are integrated (network DSL)
- Integrated network debugger



TremaShark

- Wireshark plug-in for monitoring any IPC events among/on functional modules
  - Messages
  - Secure Channel status
  - Queue status
  - etc…





Directories

```
features/                # test code
spec/                    # test code
unittests/               # test code

src/
    examples/            # sample code
    lib/                 # C libraries
    packet_in_filter/    # core module – packet-in filter
    switch_manager/      # core module – switch manager and switch daemon
    tremashark/          # tremashark

vender/                  # 3rd party software (OVS, oflops, etc.)
ruby/                    # Ruby bindings

build.rb                 # build script
cruise.rb                # CI script for developers
trema                    # Trema command
```

Trema applications as of Jul. 2011

- Sample applications
  - hello_trema, openflow_message, packet_In, switch_Info (C, Ruby)
  - event_dumper (C, Ruby)
  - list_switches (C)
  - repeater_hub (C, Ruby)
  - learning_switch / multi_learning_switch (C, Ruby)
  - routing_switch (C)
  - cbench_switch (C)
  - topology, discovery (C)

- 3rd party applications @ https://github.com/trema/apps/
  - redirectable_routing_switch (C)

Conclusion

- Trema is a software platform for OpenFlow researchers and developers
  - Multi-process modular architecture for extensibility
- Integrated developing environment
  - TDD (Test Driven Development) framework
  - Seamless integration of controller and emulated network for testing/debugging
  - User modules written in C or Ruby
- Contact: trema-dev@googlegroups.com