mPlane Architecture and Protocol

Brian Trammell, Architecture (WP1) lead
ETH Zürich
mPlane final workshop
30 November 2015, Heidelberg
mPlane architecture in one slide

- **Components** make measurements available to **clients** via the mPlane protocol.
  - Components can be **probes**, which measure, or **repositories**, which store and analyze.
- These measurements are **completely** defined in terms of **capabilities** advertised by the components.
- Clients send **specifications** to invoke these capabilities.
- Specifications can lead to **results**, or to components sending bulk data to others via **indirect export**.
Probes, Repositories, and Reasoners

- **Probes** are components that can measure something now.
- **Repositories** are components that can answer queries about the past.
- **Reasoners** are clients with learning component for (semi-)automation of measurement workflows.
### Coordination and Federation

- A **supervisor** mediates between clients and components:
  - Measurement aggregation
  - Access control centralization
  - Interdomain federation
- *Not* a measurement controller in the traditional sense due to delegation of responsibility to components.
- Requires *application-specific logic* for control distribution and result collection
Architectural Principles

- **Schema-centric measurement definition**: a measurement is completely described by the parameters it takes and the columns in the results it produces.

- **Weak imperativeness**: capabilities aren’t guarantees, normal exceptions discovered in later analysis, state and responsibility dynamically distributed throughout an infrastructure.

- Component management left out of scope
  - assume components too heterogeneous anyway.
Schema-centric measurement definition

- Traditional RPC:
  
  ping -c 3 -w 5 10.2.3.4
  ping(count, period, dest) => [int]

- Need to register entry points, argument names.

- “Can I compare ping() to webping() to nmap_christmas_tree_warning_very_beta()?”

- Schema-centric:
  
  measure(param(singleton_measurement_count, period, destination_ip4);
            result(delay_oneway_icmp))

- Requires rigorous control over the set of column names, but allows more or less infinite combination (cf. www.iana.org/assignments/ipfix)
Weak imperativeness

- **Failure is inevitable. Embrace it.**
  - Two kinds of failure:
    - Things that are part of what you’re measuring (e.g. variable connectivity on mobile probes)
    - Things that need a forklift to fix.
  - For the second class, you need completely separate infrastructure monitoring anyway.
  - For the first class, export enough metadata to allow analysis as *part of the normal measurement workflow.*
The mPlane Protocol

- Error-tolerant, distributed RPC protocol comprised of an information model (message types and contents), a representation (JSON), and a session protocol (HTTPS)
  - Flexibility in future representation (e.g. CBOR) and session protocols (e.g. WebSockets, SSH).
- Under submission to IETF for standardization (draft-trammell-mplane-protocol)

**Diagram:**

- **what a client wants a component to do**
  - messages sent by clients
  - Specification
  - Interrupt
  - Redemption
  - Withdrawal
  - Indirection
  - Receipt
  - Result

- **what a component says it can do**
  - messages sent by components
  - Capability

- **what the component did**
Capability Composition
Delegation
Query and Iteration
mPlane clients and components organized into domains by:
- which supervisor (if any) they use for coordination and federation
- common issuer of X.509 certificates for all entities in a domain
Capability Example

```json
{
    "capability": "measure",
    "version": 1,
    "registry": "http://mplane.corvid.ch/ecnspider.json",
    "label": "ecnspider-ip4",
    "when": "now ... future",
    "parameters": {
        "source.ip4": "192.0.2.33",
        "destination.ip4": "[\*]"
    },
    "results": [
        "destination.ip4",
        "ecnspider.ecnstate",
        "connectivity.ip",
        "octets.layer5",
        "ecnspider.initflags.fwd",
        "ecnspider.synflags.fwd",
        "ecnspider.unionflags.fwd",
        "ecnspider.initflags.rev",
        "ecnspider.synflags.rev",
        "ecnspider.unionflags.rev"
    ]
}
```

- Case study: path transparency measurement for ECN
- Each component advertises its willingness to perform a specified measurement in a capability
- Capability lists parameters (which the client needs to fill in) and results (which the component will measure)
The verb and set of parameters and results together define the measurement’s schema.

The schema is equivalent to the name of the RPC entry point.
Registry Extensibility

- Each measurement is bound to a registry of elements.
- Registries inherit elements from the base registry.
- Here, ECN-specific elements have been added.

```json
{
    "capability": "measure",
    "version": 1,
    "registry": "http://mplane.corvid.ch/ecnspider.json",
    "label": "ecnspider-ip4",
    "when": "now ... future",
    "parameters": {
        "source.ip4": "192.0.2.33",
        "destination.ip4": "[*]"
    },
    "results": [
        "destination.ip4",
        "ecnspider.ecnstate",
        "connectivity.ip",
        "octets.layer5",
        "ecnspider.initflags.fwd",
        "ecnspider.synflags.fwd",
        "ecnspider.unionflags.fwd",
        "ecnspider.initflags.rev",
        "ecnspider.synflags.rev",
        "ecnspider.unionflags.rev"
    ]
}
```
Specification Example

```
{
    "specification": "measure",
    "version": 1,
    "registry": "http://mplane.corvid.ch/ecnspider.json",
    "label": "ecnspider-ip4",
    "when": "now",
    "token": "d41d8cd98f00b204e9800998ecf8427e",
    "parameters": {
        "source.ip4": "192.0.2.33",
        "destination.ip4": [
            "192.0.2.67",
            "192.0.2.89",
            "192.0.2.123"
        ]
    },
    "results": [
        "destination.ip4",
        "ecnspider.ecnstate",
        "connectivity.ip",
        "octets.layer5",
        "ecnspider.initflags.fwd",
        "ecnspider.synflags.fwd",
        "ecnspider.unionflags.fwd",
        "ecnspider.initflags.rev",
        "ecnspider.synflags.rev",
        "ecnspider.unionflags.rev"
    ]
}
```

- Specification completely defines the measurement to be performed.
- Client sends a list of targets to each component.
- Component will return a single result per specification.
mPlane SDK

- Open-source toolkit for building mPlane clients and components in Python 3
  - $ pip install mplane-sdk
- Current release: feature freeze for today’s demos
- 1.0 release: post-project
  - improved configuration
  - multiple value support

![Diagram of mPlane model and scheduler](image-url)
mPlane SDK
component/client framework

- mplane/component.py provides a framework for building components and proxies for existing components in three easy steps:
  - (1) Implement logic for each activity in run() method in a subclass of mplane.scheduler.Service.
  - (2) Build capabilities to describe the specifications this run() method will accept.
  - (3) Wrap these in a Python module that returns these subclasses via a services() method.

- Common and component-specific configuration via a unified configuration file
How do I get started?

- [https://github.com/fp7mplane/protocol-ri](https://github.com/fp7mplane/protocol-ri)
  - README.md: how to build stuff on top of the SDK
  - doc/protocol-spec.md: protocol specification

- Repository is active
  - master branch stable for demonstration
  - 1.0 release in sdk-rc1.0 branch
  - Something broken? read the docs, then file an issue.