A Platform for Measurement Iteration and Automation

Brian Trammell, CSG, ETH Zurich

First NMRG Workshop on Large Scale Network Measurements
Zurich, Switzerland, 14 October 2013

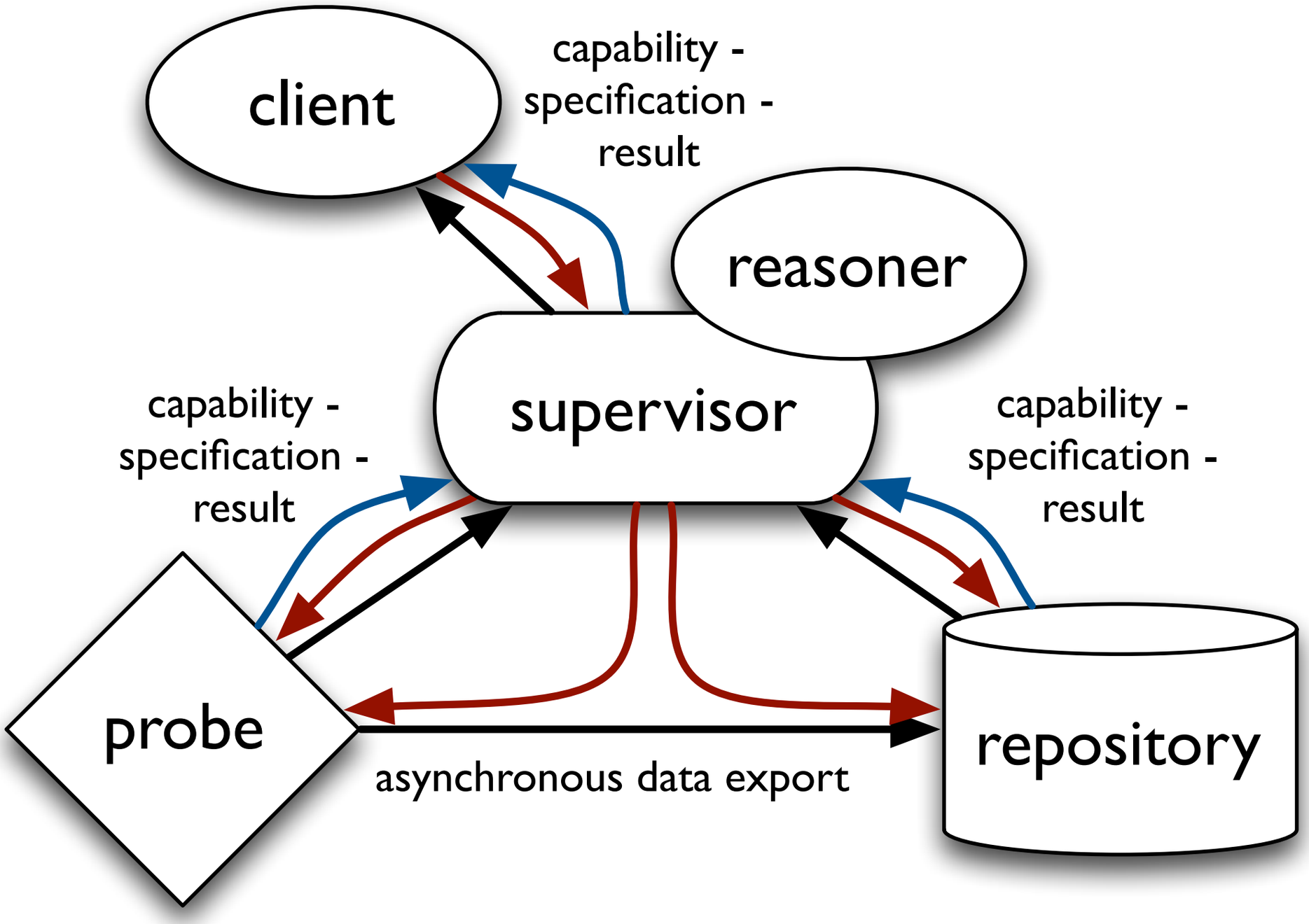
What is mPlane?

- 3-year EU FP7-funded research project, consortium of 16.
- *Goal*: build a measurement platform for intra- and inter-domain network performance troubleshooting support.
  - Support automated and automation-assisted iterative measurement for root cause analysis.
  - Research new techniques in passive and active network measurement and data analysis relevant to performance.
- *Insight*: three years is not long enough to build and integrate a bunch of working measurement tools.
- *Plan*: Leverage existing tools through a simple, easy-to-implement, "standard" interface that covers applicable metrics.

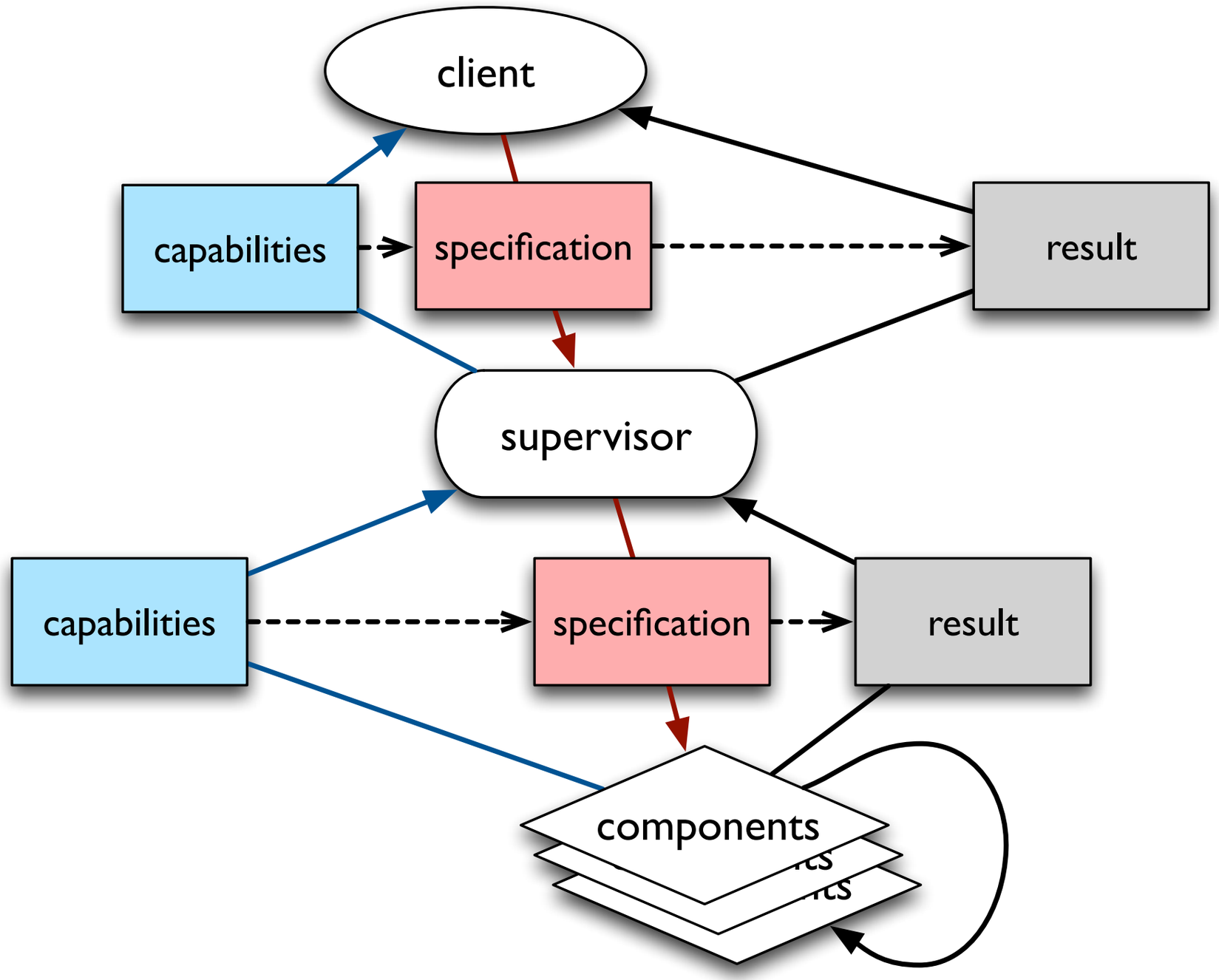
Everything's a component

- A *component* implements the mPlane control interfaces:
  - can advertise its *capabilities*,
  - accepts measurement *specifications*,
  - provides *results* (or *receipts* therefor), and
    - may participate in brokered asynchronous data export.
- *Clients* direct components to perform measurements via interfaces
- *Supervisor*: component + client
  - maps higher-level to lower-level specifications,
  - consolidates results from lower-level components.
- *Reasoner*: client supporting automated measurement iteration.

General Architecture



Statements


client
capabilities
specification
result
supervisor
capabilities
specification
result
components

Capabilities

- What can a component do?
  - Produce measurements (of a given type, directly or via a given protocol)
  - Consume measurements (of a given type, via a given protocol)
  - Other stuff (free-form, matched by name)
- Capabilities have *parameters*.
  - Must be given in a specification to use the capability
  - Parameters have constraints (i.e., acceptable values)

Specifications, Results, and Receipts

- A specification is an order to a component to perform a measurement or analysis.
  - Essentially a “filled-in” capability.
- A result may be returned immediately...
  - in the same format as the specification, with all parameters intact.
- ...or later by presenting a receipt.
- Specifications for *asynchronous export* coordinate the exchange of data among components.

Iterative measurement

- Iterative measurement uses the results of one measurement to influence inputs or choice of a subsequent measurement.
  - ...natural pattern in “drill-down” during troubleshooting
- Reasoner automates iteration by learning which subsequent measurements are most likely to result in a determination of cause.

Flexible Data Model and Transport

- Statement (capability, specification, result) data model defined separate from serialization format.
  - Reference implementation: JSON
  - Examples (for readability): YAML
- Multiple app-layer protocols for moving statements among components, supporting both push and pull for each statement type
  - Default: HTTP over TLS w/mutual auth
  - Easier key management: raw messages over SSH

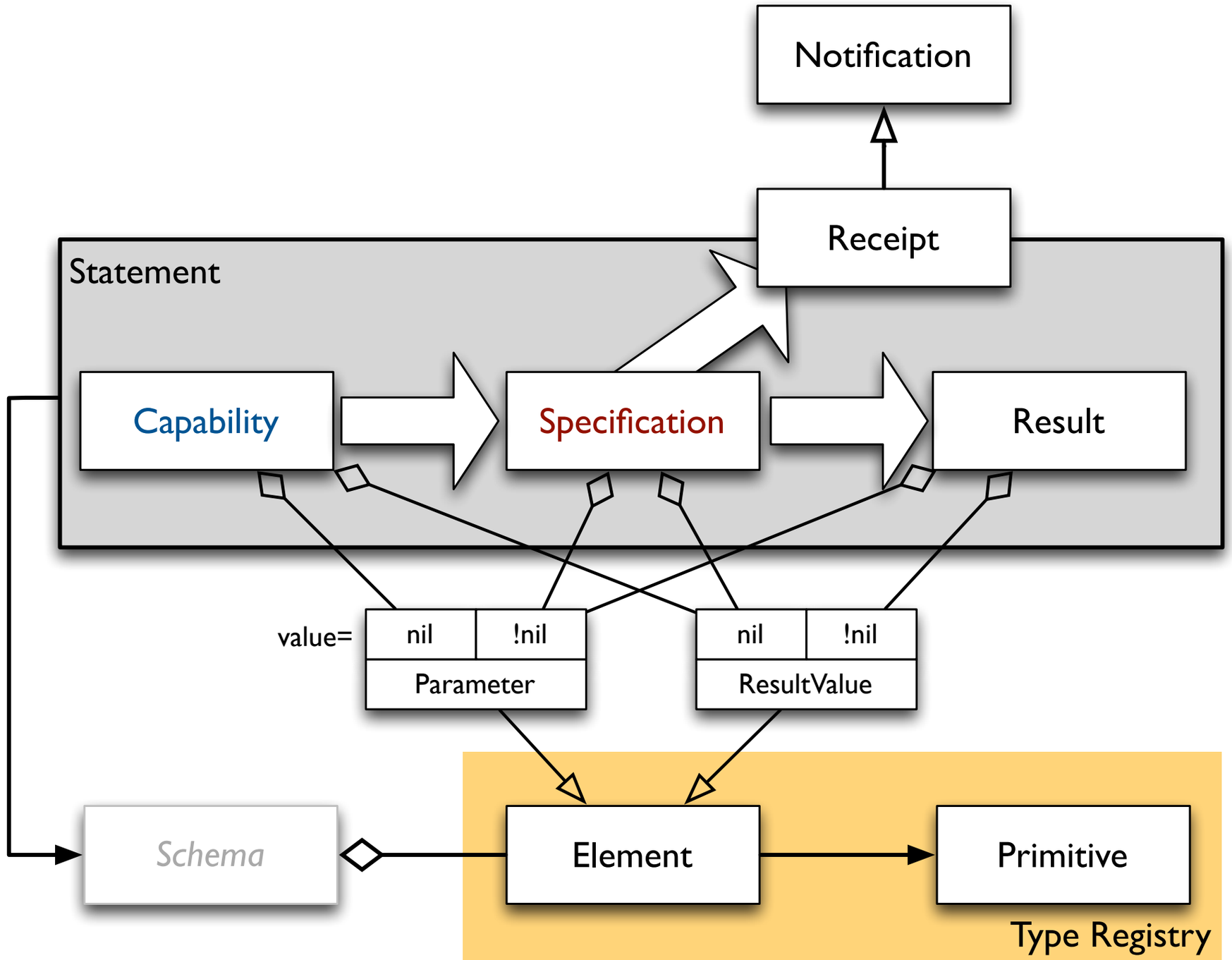
Yay! We've (re-)invented middleware!

- Less ugly than the W3C Web Services stack, but basically just measurement-aware RESTful RPC with timing, delay tolerance, asynchronicity.
- How does this get us any closer to measurement interoperation?

Types

- Network measurement produces rows in databases.
- Analysis munges rows into other rows.
- The measurement or analysis performed is completely described by the schema...
  - ...if you've designed the schema right.
- Operations of common measurement tools can be fully described by the data types involved.

Information Model



Type Interoperability

- Schema = table, template
- Element = column, info element
- Primitive = storage representation
- Two schemas are compatible if one is a subset of the other.
- Interoperability becomes a matter of ensuring elements have equivalent meanings.

Type Registry

- Structured namespace of Elements
  - [value].[modifiers].[units].[function]:[primitive]
    - e.g. delay.twoway.icmp.ms.mean: natural
- Mappings to IPFIX Information Elements when appropriate.
- Current registry covers network flow, common active measurements, and QoS use cases.

Example: ping

- capability: measure
  parameters:
    start.ms: now...+inf
    end.ms: now...+inf
    source.ip4: 10.2.3.4
    destination.ip4: *
    period.s: 1...60
  results:
    - delay.twoway.icmp.ms.min
    - delay.twoway.icmp.ms.mean
    - delay.twoway.icmp.ms.max

Example: ping

- specification: measure
  parameters:
    start.ms: 2013-09-13 11:30:00
    end.ms: 2013-09-13 11:31:00
    source.ip4: 10.2.3.4
    destination.ip4: 10.4.5.6
    period.s: 1
  results:
    - delay.twoway.icmp.ms.min
    - delay.twoway.icmp.ms.mean
    - delay.twoway.icmp.ms.max

Example: ping

- results: measure
  parameters:
    start.ms: 2013-09-13 11:30:01.045
    end.ms: 2013-09-13 11:31:01.044
    source.ip4: 10.2.3.4
    destination.ip4: 10.4.5.6
    period.s: 1
  results:
    - delay.twoway.icmp.ms.min
    - delay.twoway.icmp.ms.mean
    - delay.twoway.icmp.ms.max
  resultvalues:
    - - 41
      - 47
      - 53

Example: traceroute6

- capability: measure
  link: mplane-https://supervisor.example.com/traceroute
  parameters:
    start.ms: now...+inf
    end.ms: now...+inf
    source.ip6: 2001:618:1:102::2
    destination.ip6: *
    hops.ip6.max: 1...255
    delay.twoway.udp.ms.count: 1...3
  results:
    - intermediate.ip6
    - hops.ip6
    - delay.twoway.udp.ms

Example: traceroute6

- specification: measure
  parameters:
    start.ms: now
    end.ms: now
    source.ip6: 2001:618:1:102::2
    destination.ip6: 2001:470:26:9c2::3
    hops.ip6.max: 32
    delay.twoway.udp.ms.count: 1
  results:
    - intermediate.ip6
    - hops.ip6
    - delay.twoway.udp.ms

Example: traceroute6

- results: measure
  parameters:
    start.ms: now
    end.ms: now
    source.ip6: 2001:618:1:102::2
    destination.ip6: 2001:470:26:9c2::3
    hops.ip6.max: 32
    delay.twoway.udp.ms.count: 1
  results:
    - intermediate.ip6
    - hops.ip6
    - delay.twoway.udp.ms
  resultvalues:
    - - 2001:618:ffff:1::1036:1
      - 1
      - 4
    - - 2001:618:ffff:1::1035:2
      - 2
      - 7
    - - 2001:7f8:24::aa
      - 3
      - 9

comparing mPlane and RIPE Atlas

- RIPE Atlas: ~4k small active hardware probes provide traceroute, ping, DNS.
  - Protocol + implementation + instantiation
- Centralized set of RIPE-operated controllers running Atlas-specific control and reporting protocols.
  - REST API for data access.
  - Control subject to credit availability.
- Atlas' tests are mostly covered by the mPlane reference implementation.
  - Interop experiment: mPlane interface proxy to Atlas API, allow retrieval of results by Atlas participants.