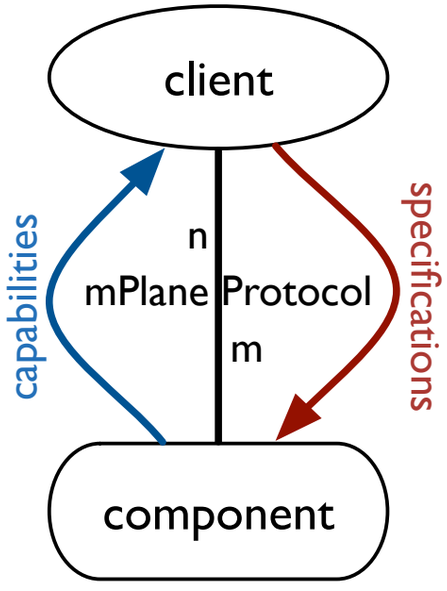
Internet Path Transparency Research and the mPlane Platform

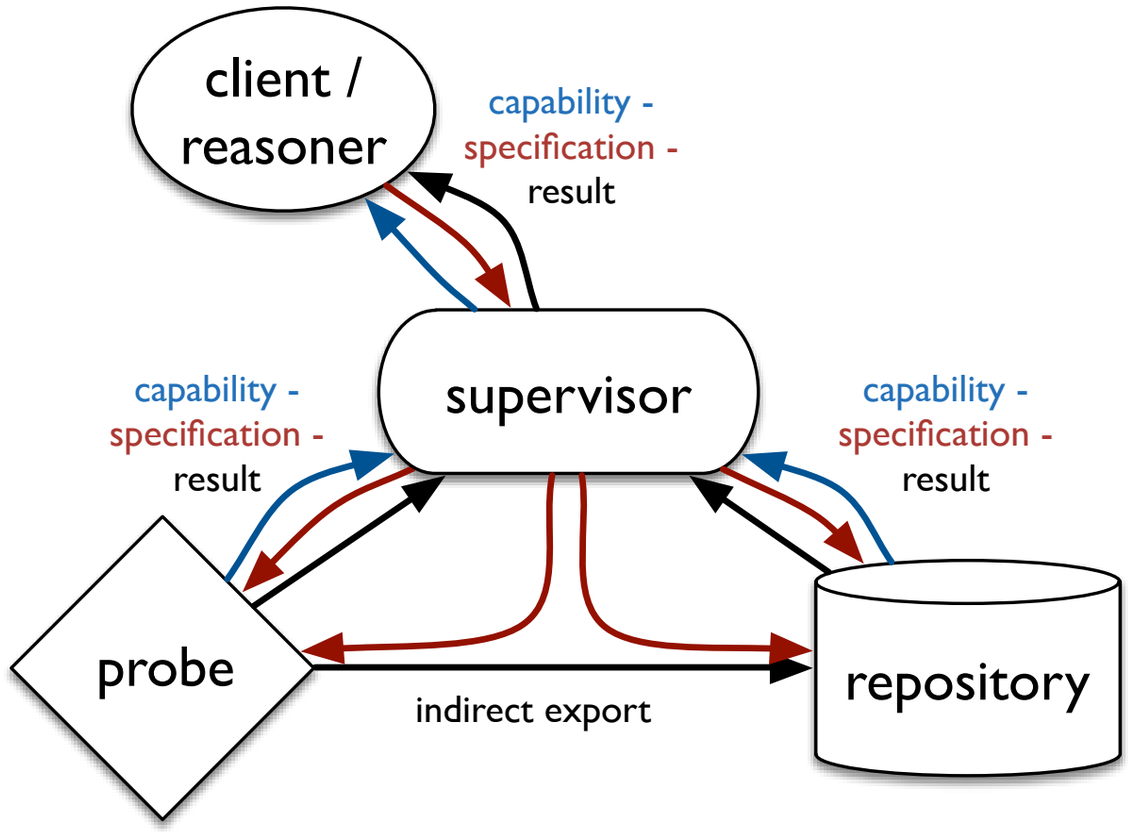
Brian Trammell, ETH Zürich
(with Mirja Kühlewind and Elio Gubser, ETH Zürich)
RIPE Measurement and Tools Working Group
RIPE 71 Bucharest, 18 November 2015

mPlane

(in one slide)

- Self-descriptive, error-tolerant RPC protocol connecting *clients* with *components* to cooperatively perform network measurements and analysis using heterogeneous tools.
- Measurements and analyses described using *capabilities* containing measurement *schemas* defined in terms of a registry of *elements*.
- Schema defines the measurement to perform.
- Weak imperativeness: responsibility follows the message flow.
- *Supervisors* knit larger infrastructures of heterogeneous components together.

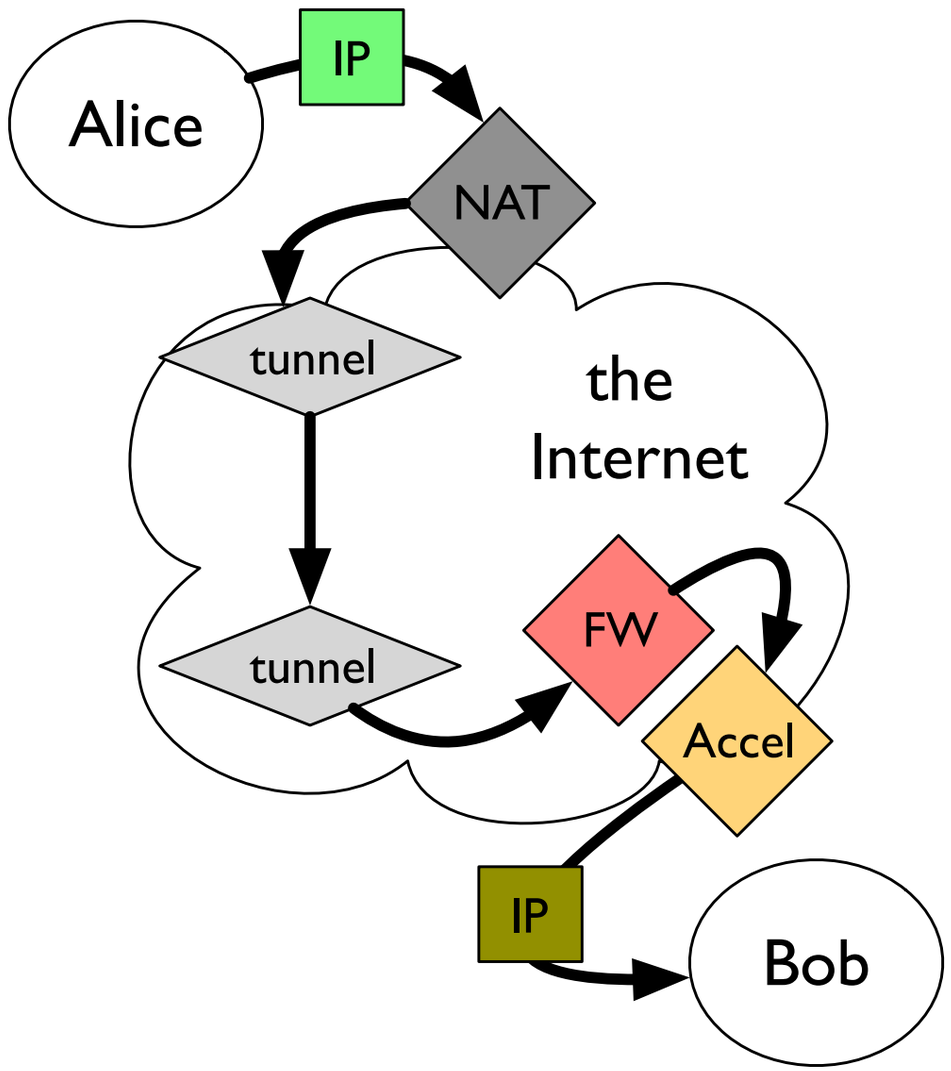




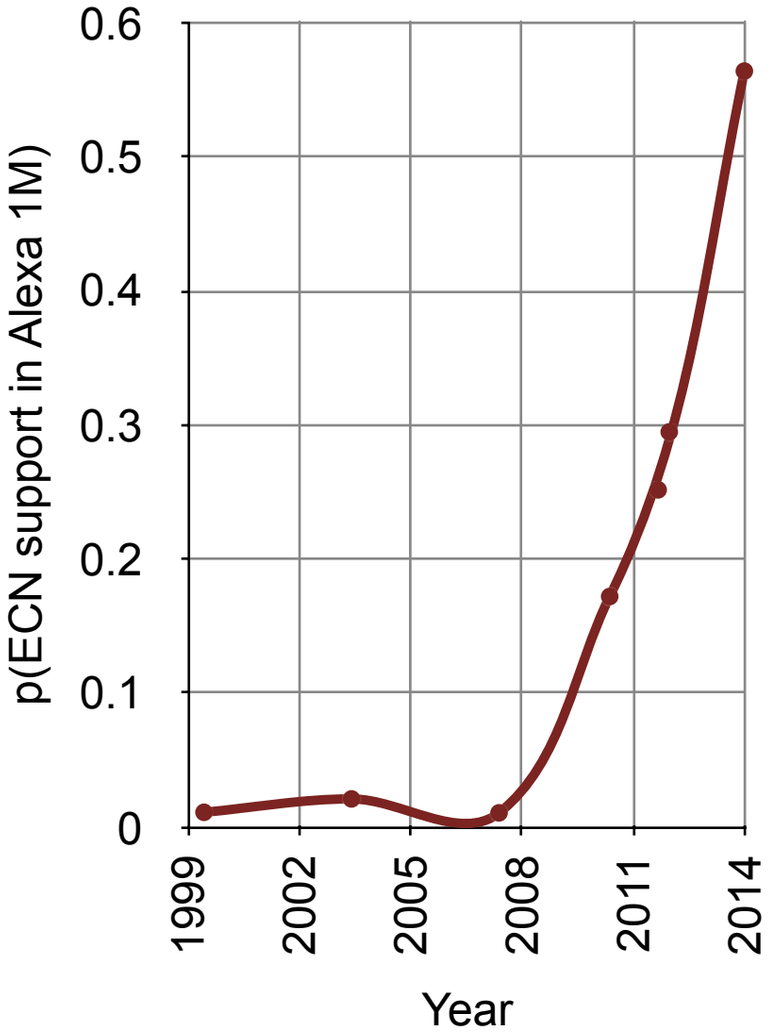
path transparency

(in one slide)

- The Internet is not end-to-end...
  - some of this is policy, but a lot of it is accident
  - deployment of new protocols over IP, transport extensions difficult or impossible
- ...but some paths are worse than others.
  - Goal: data on "how bad" and "where" to guide future protocol design
  - In operations: another tool for troubleshooting connectivity dependency for unusual traffic



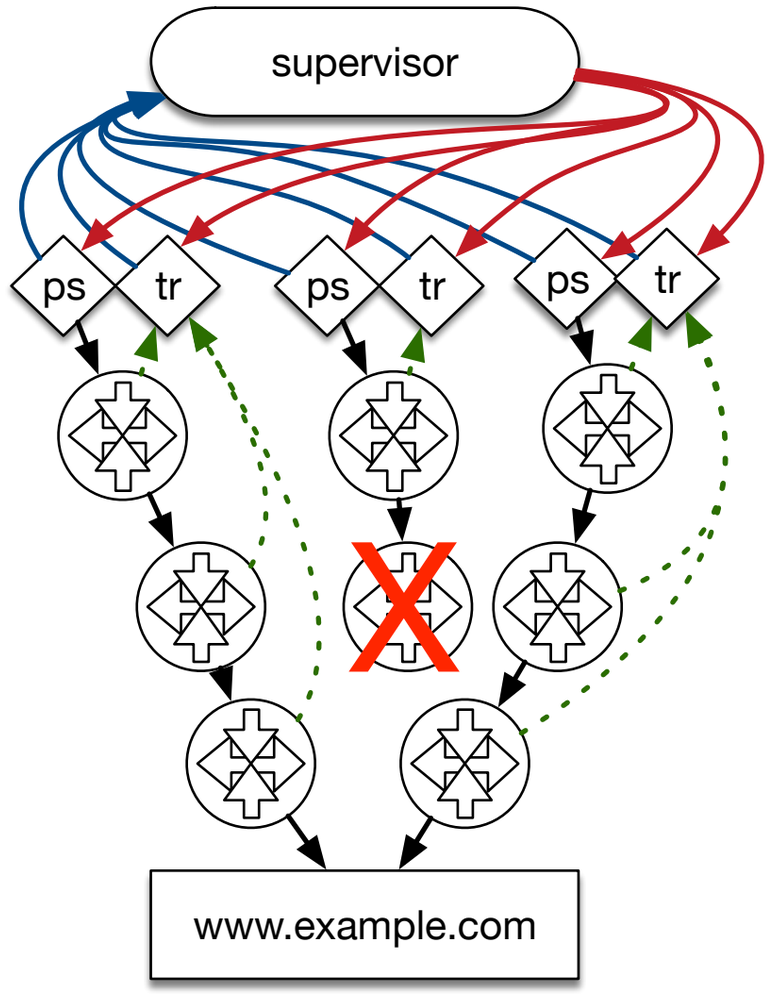
use case: ECN



- replaces loss as a congestion control signal
- Deployment in 00's failed
  - Router reboots, broken middleboxes, use of TOS
  - Linux kernel defaults have since led to server-side deployment
- Question: is it safe to turn on by default in client OSs?

applying mPlane to path transparency

- mPlane-based `pathspider` tool connects to set of targets with feature enabled and disabled.
- `pathspiders` at multiple vantage points find path dependency.
- Triggers `tracebox` to localize impairment.
- mPlane enabled easy integration.



path transparency on the web

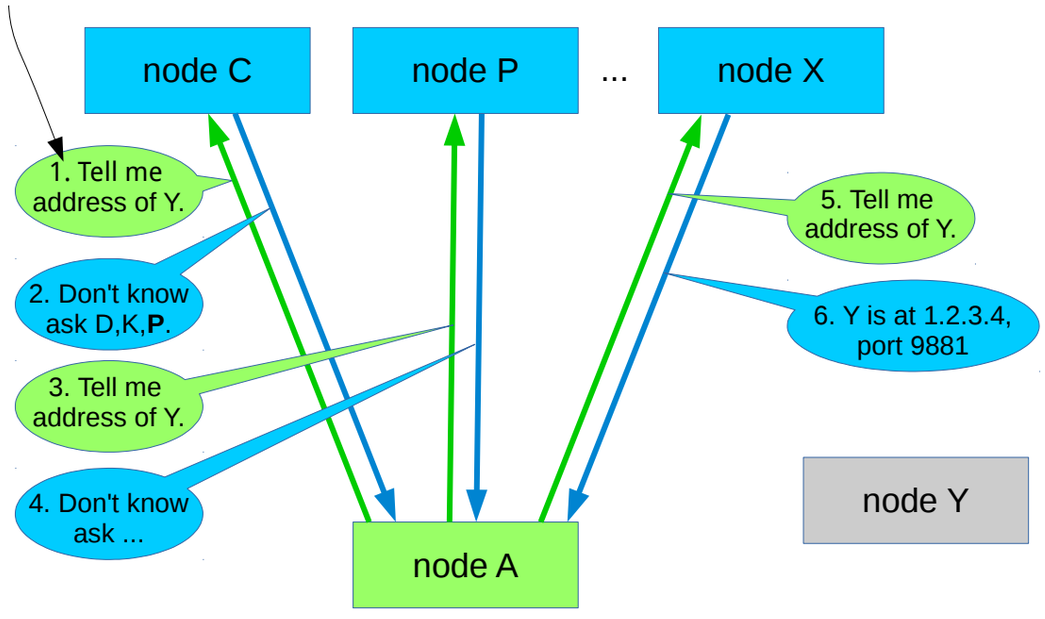
Result[1] *(Trammell et al, PAM 2015)*

- 0.5% of Alexa 1M refuse to connect with ECN, but fallback saves these. 0.04% of sites may have more interesting impairments.
- **Safe to use client defaults to drive ECN deployment.**
- Apple currently testing this result in developer betas of iOS and OSX[2]
  - terrestrial Internet basically OK, mobile Internet less OK
  - impairments primarily access-network dependent: runtime per-connection fallback is a valid strategy.

[1] B. Trammell et. al. “Enabling Internet-Wide Deployment of ECN”, Passive and Active Measurement, March 2015, Brooklyn.
[2] T. Pauly, “Results from wide testing of ECN”, IRTF HOPSRG at IETF 94, November 2015, Yokohama.

path transparency beyond the web

- What about client machines?
- **Ideal approach:** lots of probes on client networks (Atlas, Netalyzr, Chrome, etc.)
- **Stopgap approach:** harvest client IPs from BitTorrent DHT
  - Simply replace the web resolver component with a DHT resolver component.
- **Results**: of 687k machines, 0.21% unambiguously have a problem when ECN negotiated: comparable to web.



I’m sold, tell me more

- mPlane SDK: https://github.com/fp7mplane/protocol-ri
  - Or just `$ pip3 install mplane-sdk`
    - (Note: pathspider needs the `sdk-multival` branch)
- ECN-Spider/pathspider: https://github.com/britram/pathtools
  - ECN now.
    A/B testing of arbitrary protocol features soon.

Backup

mPlane in detail:

Capability

- Each component advertises its willingness to perform ECN measurements in a capability
- Results are intermediate: for each target, return connectivity and ECN TCP/IP flags information both with and without ECN enabled.

```
{
    "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"
    ]
}
```

mPlane in detail:

Capability Schema

- 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.

```
{
    "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"
    ]
}
```

mPlane in detail:

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 to the base registry.

```
{
    "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"
    ]
}
```

mPlane in detail:

Specification

- Specification completely defines the measurement to be performed
- Controller sends a list of targets to each vantage point.
- Vantage point returns a single *result* per specification.

```
{
    "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"
    ]
}
```

path transparency beyond mPlane

- mPlane vision: heterogeneous measurement tools declare their capabilities in terms of a common vocabulary of elements, and you get plug and play measurements.
- messy realities of path transparency measurement:
  - lots of data (e.g. application/OS error logs) available that may not fit into a clean observation schema
  - we need to get more detailed about the definition of path transparency elements in the registry in order to ensure comparability
- next step: a path transparency observatory

future work:
a path transparency observatory

- What we really want is a way to analyze path transparency source data in detail and in aggregate…
  - … from multiple sources of varying resolution
  - … where we may not have much detail at all about the paths (full traceroute, IP pairs, prefix pairs, AS pairs, pseudonymous IDs…)
    - … but where we can verify that the same test was used for the same transparency condition
    - … and (as a bonus) such that we can repeat studies in time and space (yay, science!)
- Insight: transparency conditions can be completely and repeatably described in terms of cleverly expressed patterns of packets.
- Integration: define mPlane elements for tests in terms of these patterns.
- Observatory to be build in the framework of the upcoming (2016-2018) H2020 MAMI project; watch this space!