A Brief Introduction to Internet Measurement

Brian Trammell, ETH Zürich

Repeatability and Comparability in Measurement (RCM) Tutorial

ACM SIGCOMM 2018, Budapest, 20 August 2018

mami

measurement and architecture for a middleboxed internet

measurement | architecture | experimentation

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 688421.The opinions expressed and arguments employed reflect only the authors' view. The European Commission is not responsible for any use that may be made of that information.

Supported by the Swiss State Secretariat for Education, Research and Innovation under contract number 15.0268. The opinions expressed and arguments employed herein do not necessarily reflect the official views of the Swiss Government.

Preparation [do this first]

- Tracebox VM
  - install git, vagrant, virtualbox
  - `$ git clone https://github.com/mami-project/vpp-mb -b rcm`
  - `$ cd vagrant && vagrant up`
  - note: minimum RAM ~4G

- PATHspider / PTO VM
  - you need an ssh client, and we need your public key
  - measurement VM is running on DigitalOcean, get your username, password, hostname, and API token from me.

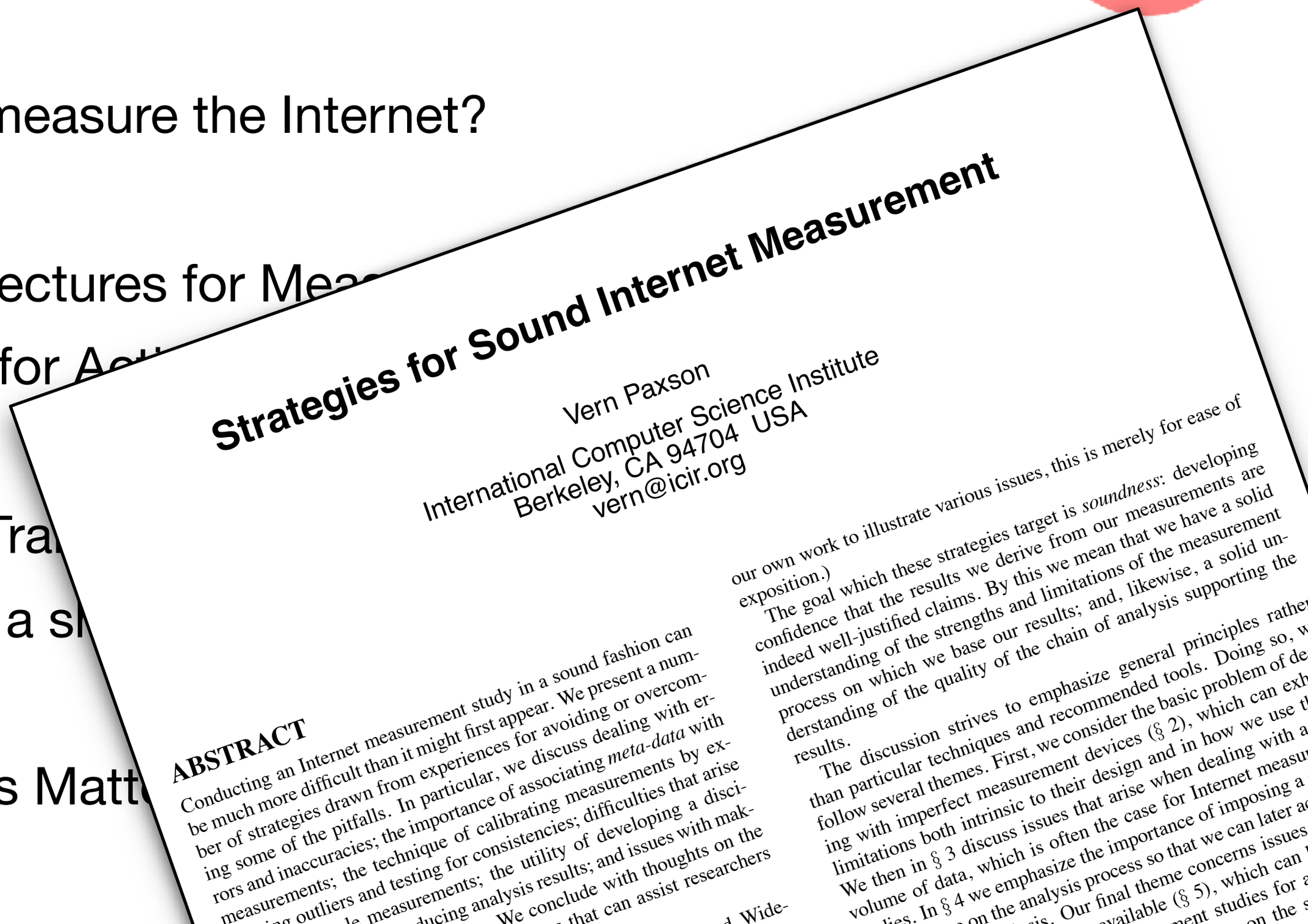
A quick outline

- Why measure the Internet?
- Architectures for Measurement
- Tools for Active Measurement
- Path Transparency Measurement
  - and a short history of ECN
- Details Matter, and a Few Simple Principles

A quick outline

- Why measure the Internet?
- Architectures for Mea
- Tools for Act
- Path Tra
  - and a sl
- Details Matt

mami



Strategies for Sound Internet Measurement

Vern Paxson

International Computer Science Institute
Berkeley, CA 94704 USA
vern@icir.org

ABSTRACT

Conducting an Internet measurement study in a sound fashion can be much more difficult than it might first appear. We present a number of strategies drawn from experiences for avoiding or overcoming some of the pitfalls. In particular, we discuss dealing with errors and inaccuracies; the importance of associating *meta-data* with measurements; the technique of calibrating measurements by exing outliers and testing for consistencies; difficulties that arise e measurements; the utility of developing a discicing analysis results; and issues with makWe conclude with thoughts on the that can assist researchers

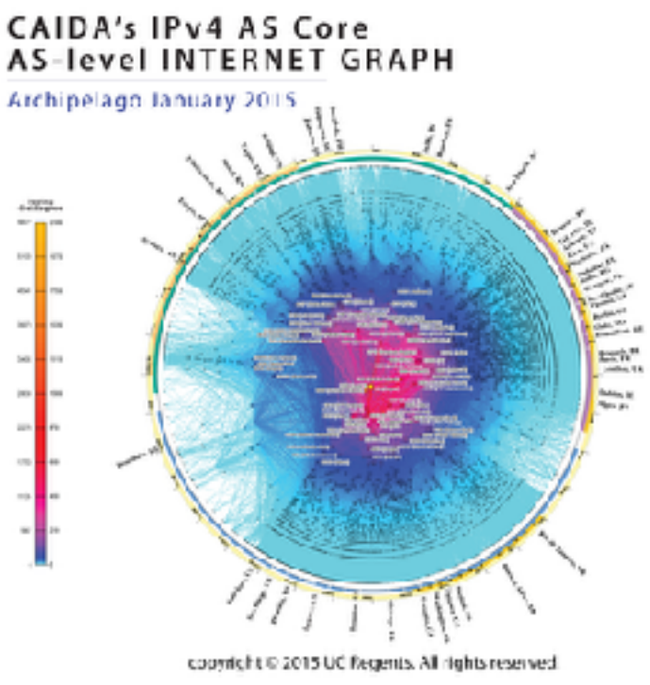
our own work to illustrate various issues, this is merely for ease of exposition.)

The goal which these strategies target is *soundness*: developing confidence that the results we derive from our measurements are indeed well-justified claims. By this we mean that we have a solid understanding of the strengths and limitations of the measurement process on which we base our results; and, likewise, a solid understanding of the quality of the chain of analysis supporting the results.

The discussion strives to emphasize general principles rather than particular techniques and recommended tools. Doing so, w follow several themes. First, we consider the basic problem of deing with imperfect measurement devices (§ 2), which can exh limitations both intrinsic to their design and in how we use t We then in § 3 discuss issues that arise when dealing with a volume of data, which is often the case for Internet measu In § 4 we emphasize the importance of imposing a on the analysis process so that we can later a Our final theme concerns issues available (§ 5), which can ment studies for a on the

Why measure the Internet?

Leonardo Rizzi CC-BY-SA



- **Operations**: keep the Internet working
  - "What's broken?"
  - "Who's attacking me?"
  - "Are things working as expected; if not, why not?"
  - "How should we plan future expansion of our network?"
- **Research:** understand the Internet as a phenomenon in itself.
  - "What does the network look like?"
  - "How will the network look tomorrow?"
  - "Hm, that's interesting, what's that?"
- **Engineering:** support protocol design decisions with data

(Our project focuses primarily on the third, but the techniques we'll explore today are applicable to all three activities)

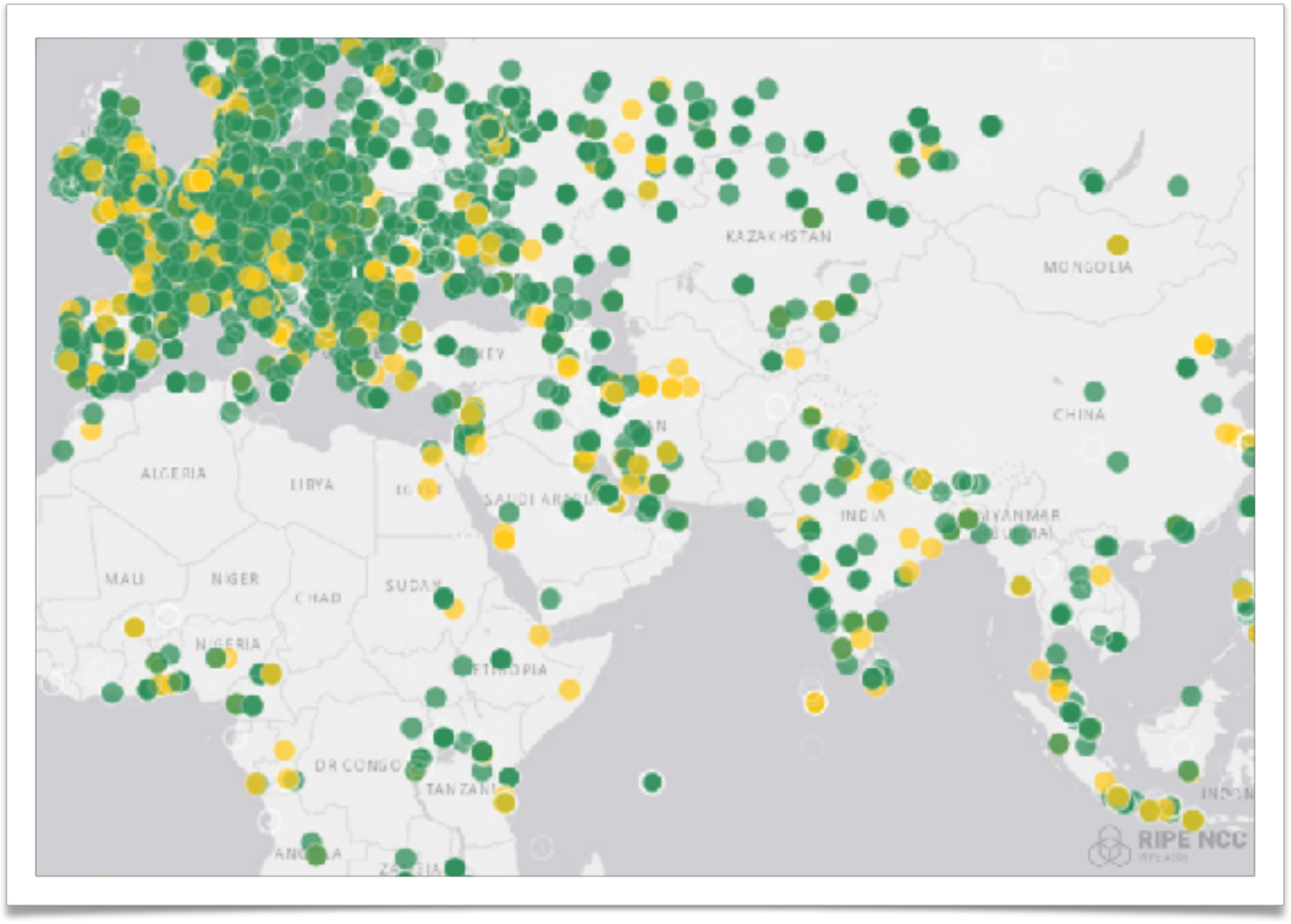
Active versus Passive Measurement

- **Active measurement** uses *dedicated measurement traffic* to induce a measurable reaction from the network and/or far endpoint.
  - Examples: `ping`, `traceroute`, everything RIPE Atlas does
  - Tradeoffs: you can measure what you're aiming at, but (1) you might not be measuring exactly what productive traffic sees and (2) you have to pay the overhead of unproductive measurement traffic.
- **Passive measurement** observes *productive* traffic and draws inferences about the state of the network and endpoints from what it can see.
  - Example: IPFIX, `wireshark`
  - Tradeoffs: what you observe is what you get, but you can only choose paths opportunistically, and you have to be *very* careful not to over-observe (end-user privacy).
- We focus today on active measurement

Context in Active Measurements: tradeoffs in selecting vantage points

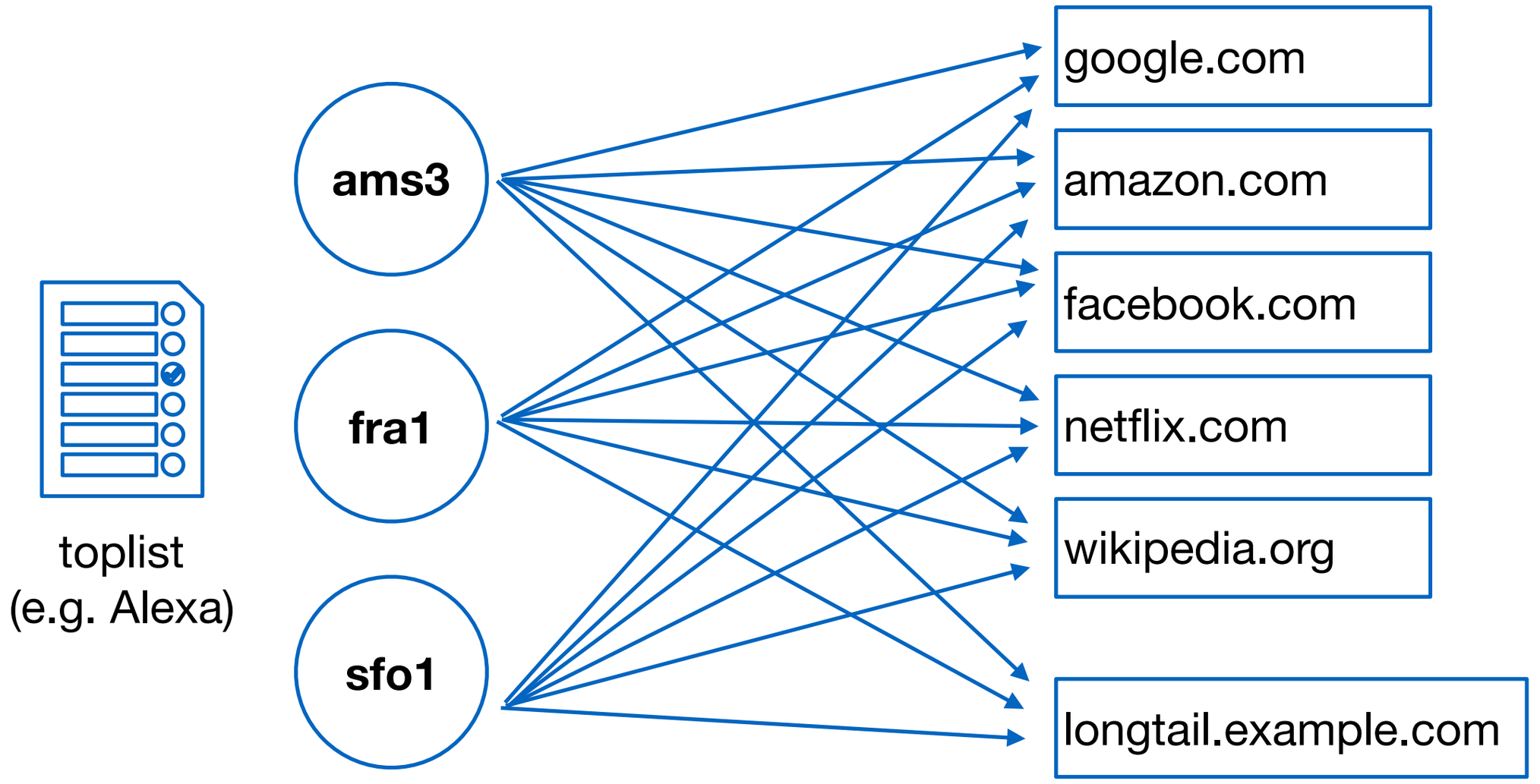
- Where you measure from is also important
- Increasing choice in vantage points:
  - Active measurement networks (e.g. RIPE Atlas); including residential measuremets (e.g. mLab) and mobile testbeds (e.g. MONROE)
  - Mesh of testbed nodes (e.g. Planetlab, Ark)
  - Core endpoints (e.g. DigitalOcean, AWS) toward public/ uncontrolled targets (e.g. Alexa TopN)

Measurement Networks (e.g. RIPE Atlas)



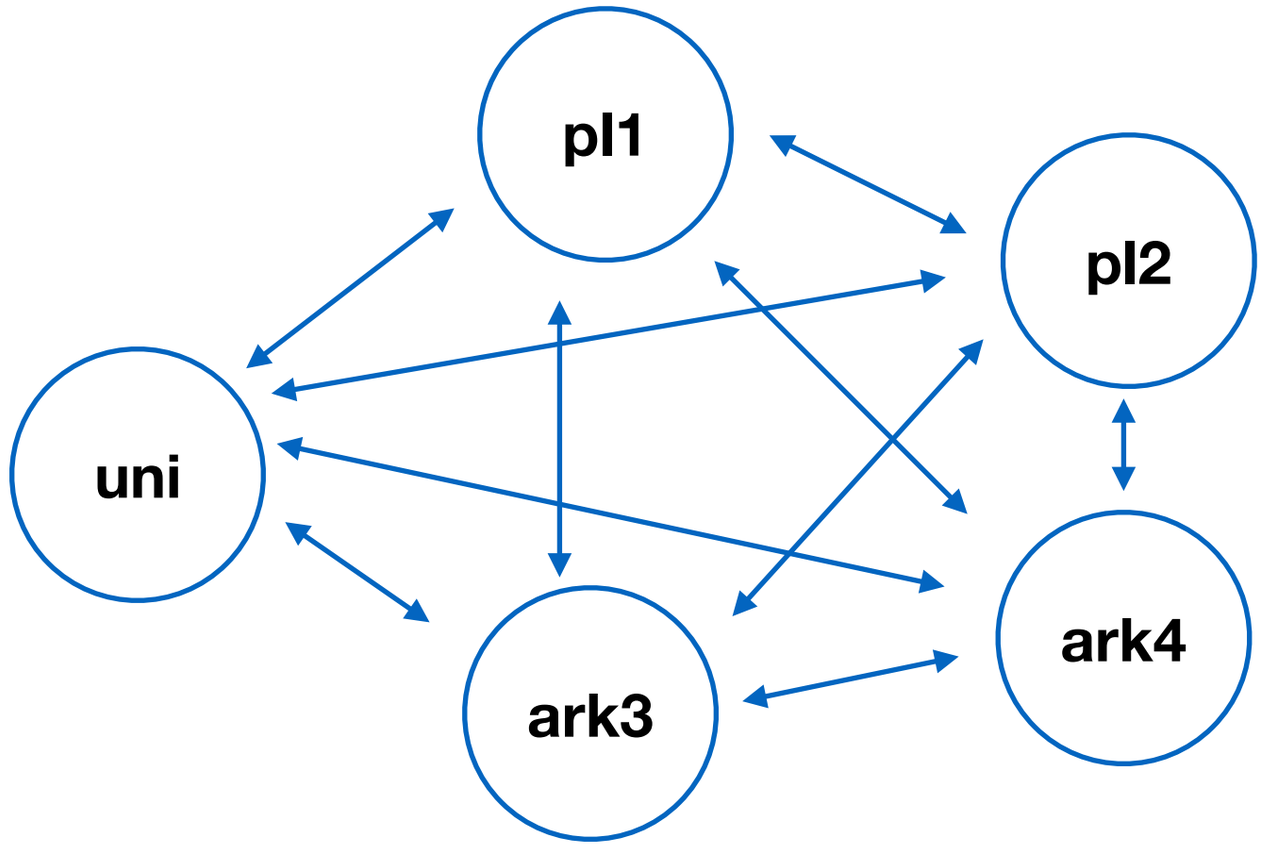
- Many (thousands) of small probes hosted on (often volunteer) networks allow shared access to simple active measurement infrastructure
- Advantages: scale, ease of use, diversity in network access
- Disadvantage: limits on the kinds and scale of measurements that can be done.

Core to Public Measurements



- Generate a list of public targets and measure it from multiple vantage points
- Advantage: the cloud scales nicely
- Disadvantages: one-sided measurement, inference required based on protocol behavior, toplist bias, not useful for access network behavior.

Testbed Mesh Measurements (e.g. PlanetLab)



- Advantage: full control of both endpoints (path isolation and two-way tracing are possible)
- Disadvantage: hard to set up, testbed networks tend to be non-representative.

Crowdsourcing

- Pay people to run small measurement tools from their own Internet connections.
- Advantages: large diversity of access network connectivity.
- Disadvantages: usually non-expert workers, so tooling needs to be packaged to be easy to use in very diverse endpoint environments, bias difficult to quantify.

Tools for active measurements

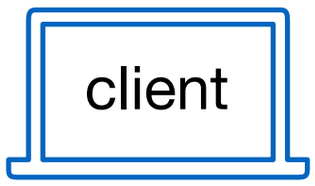
- Community uses a limited set of powerful tools

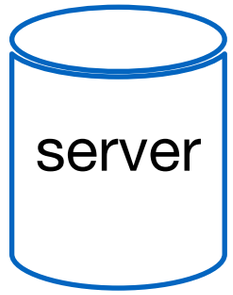
  or

- "let me hack up this quick little script"?

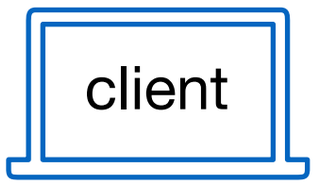
- Using a common set of tools makes measurements more comparable and reproducible.
  - I don't need to get your toolchain up and running to run your measurement on my network
  - When we all use the same tools, we all understand the limitations

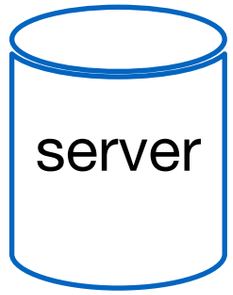
In the beginning, there was `ping`, and it was good.



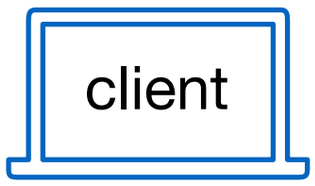


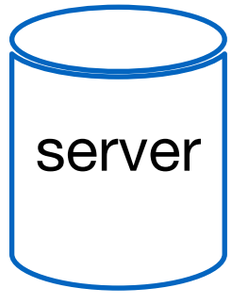
In the beginning, there was `ping`, and it was good.



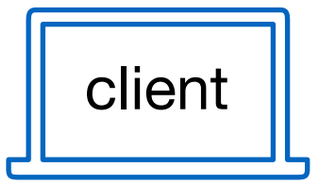


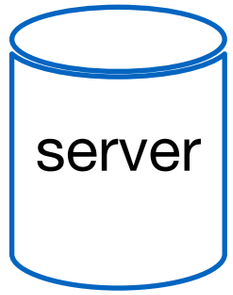
In the beginning, there was `ping`, and it was good.





In the beginning, there was `ping`, and it was good.





In the beginning, there was `ping`, and it was good.

- Sometimes the simplest tools are the easiest to understand.
- Send an ICMP Echo Request, expect a corresponding ICMP Echo Reply.
- To measure* two-way latency: $RTT_{ping} = t_{recv} - t_{send}$
- To measure loss** $RTT_{ping} \cong \infty \rightarrow$ lost

In the beginning, there was `ping`, and it was good.

- Sometimes the simplest tools are the easiest to understand.
- Send an ICMP Echo Request, expect a corresponding ICMP Echo Reply.
- To measure* two-way latency: $RTT_{ping} = t_{recv} - t_{send}$
- To measure loss** $RTT_{ping} \cong \infty \rightarrow$ lost

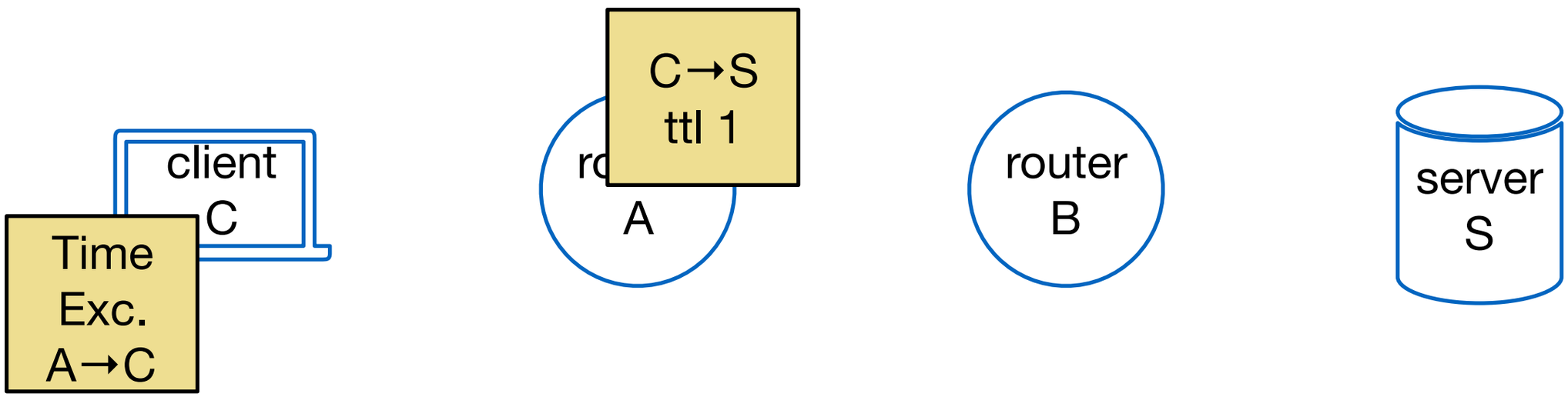
the fine print:

*Might not measure what you want: ICMP traffic may be handled differently from productive traffic; ICMP terminated by the kernel, so doesn't correspond to application status.

**No way to split forward from reverse path: bad for localizing faults.

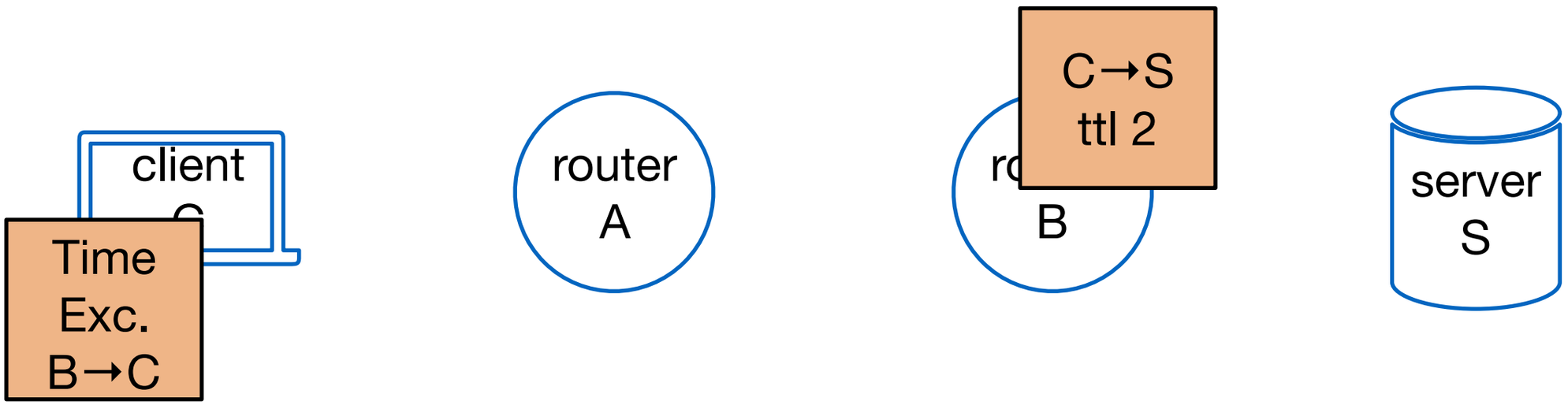
`traceroute`: making educated guesses about where the packets are going

`traceroute`: making educated guesses about where the packets are going



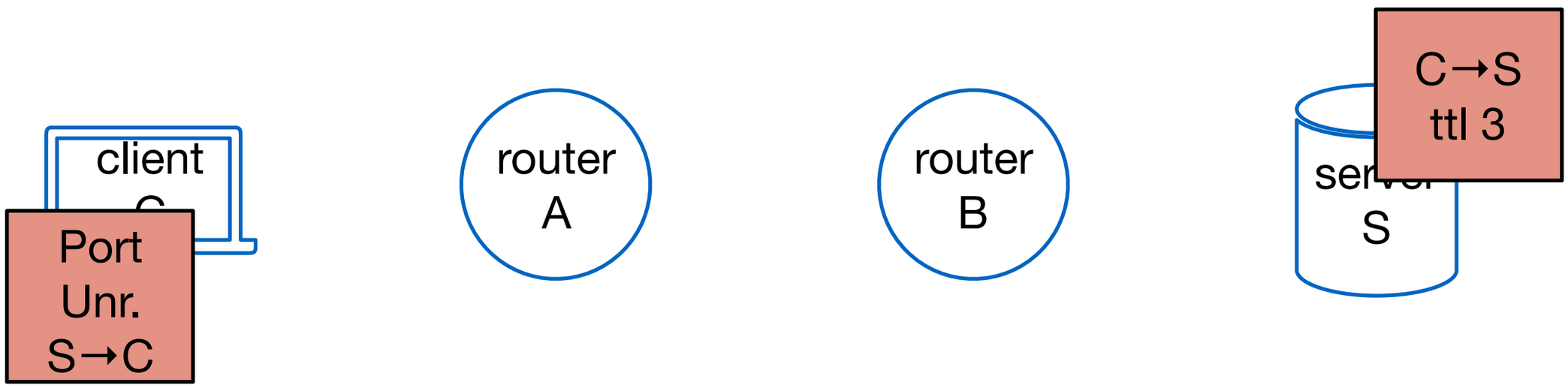
`traceroute`: making educated guesses about where the packets are going

`traceroute`: making educated guesses about where the packets are going



`traceroute`: making educated guesses about where the packets are going

`traceroute`: making educated guesses about where the packets are going



`traceroute`: making educated guesses about where the packets are going

`traceroute`: making educated guesses about where the packets are going

- Send (ICMP, UDP, TCP) packets with a TTL lower than the number of hops between the source and the destination.
  - For TTL *n*, the *n*th hop* away from the source should** send back an ICMP Time Exceeded message.
    - Do this iteratively to make a list*** of hops on the forward path**** with RTTs.
- If this sounds like a hack, that's because it is.
- Tracebox (later) is a refinement on this technique.

`traceroute`: making educated guesses about where the packets are going

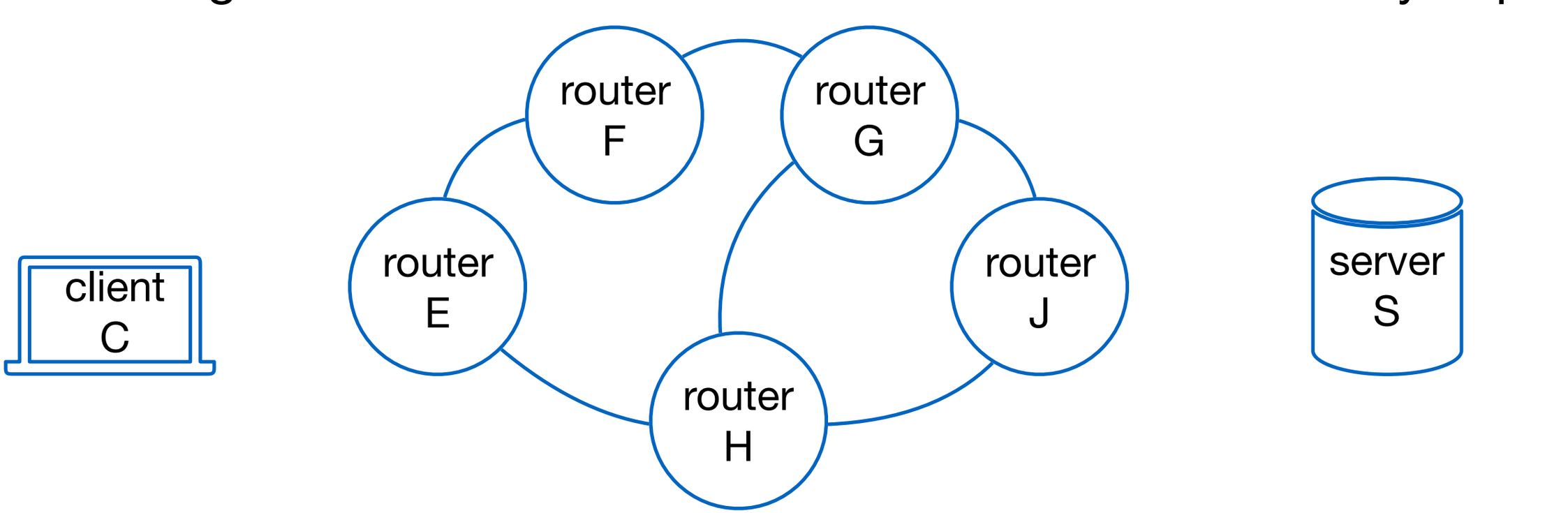
- Send (ICMP, UDP, TCP) packets with a TTL lower than the number of hops between the source and the destination.
  - For TTL *n*, the *n*th hop* away from the source should** send back an ICMP Time Exceeded message.
  - Do this iteratively to make a list*** of hops on the forward path**** with RTTs.
- If this sounds like a hack, that's because it is.
- Tracebox (later) is a refinement on this technique.

the fine print:

* where "hop" means "a router that speaks IP and decrements TTL"; tunnels are invisible
** ICMP is considered by some to be "reconnaissance activity", and therefore blocked at network borders
*** this list may or may not have any relationship to actual connectivity between routers, as a router may use any of its addresses to reply from, and using parts of the flow key to identify different hops will cause traceroute packets to take (wildly) different paths in the presence of ECMP (see Paris Traceroute).
**** no way to split paths: only the forward path is visible (but see various Reverse Traceroute schemes)

Paris `traceroute`: making even better guesses about where the packets are going

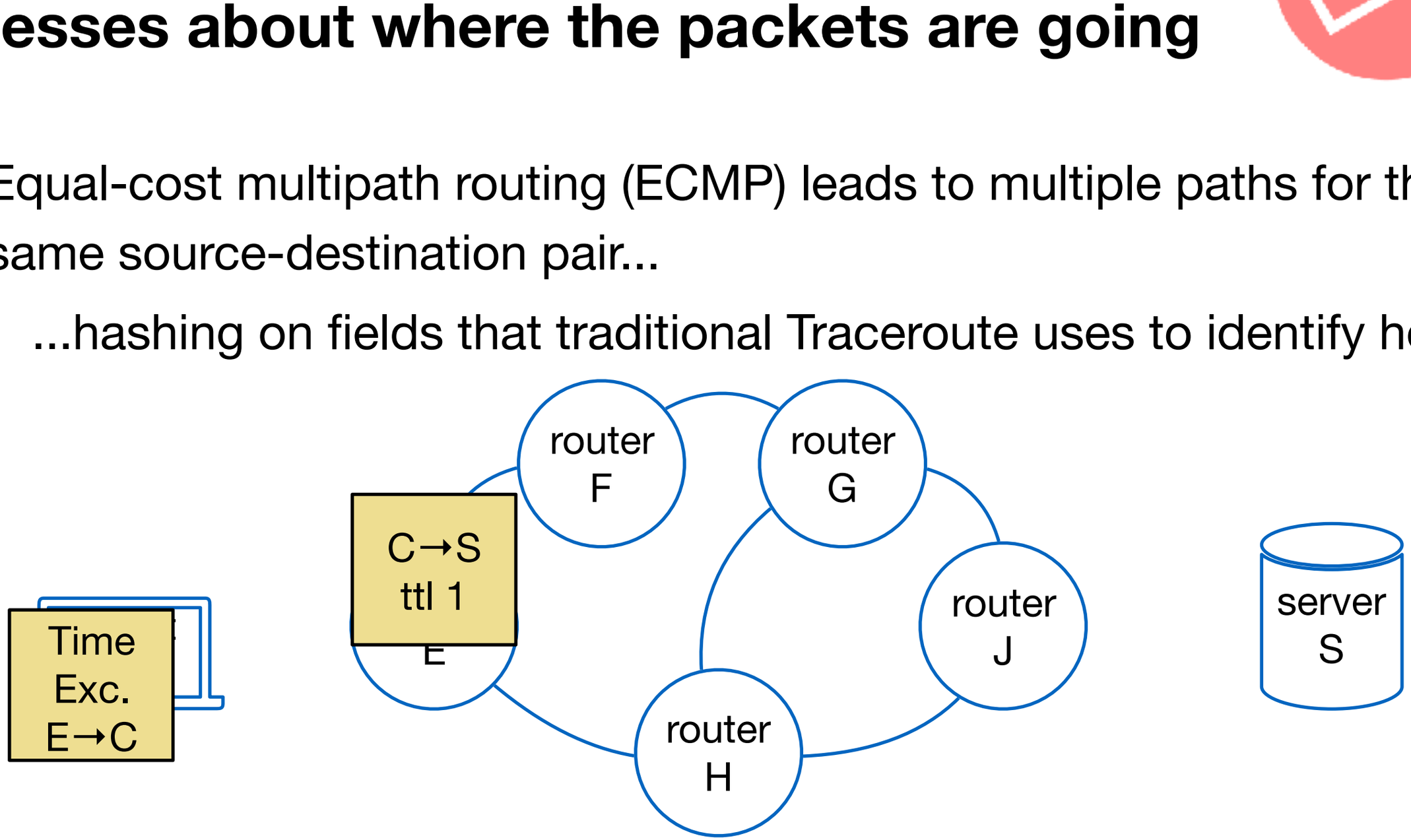
- Equal-cost multipath routing (ECMP) leads to multiple paths for the same source-destination pair...
  - ...hashing on fields that traditional Traceroute uses to identify hops



- Paris traceroute addresses this by holding flow identifiers constant.

Paris `traceroute`: making even better guesses about where the packets are going

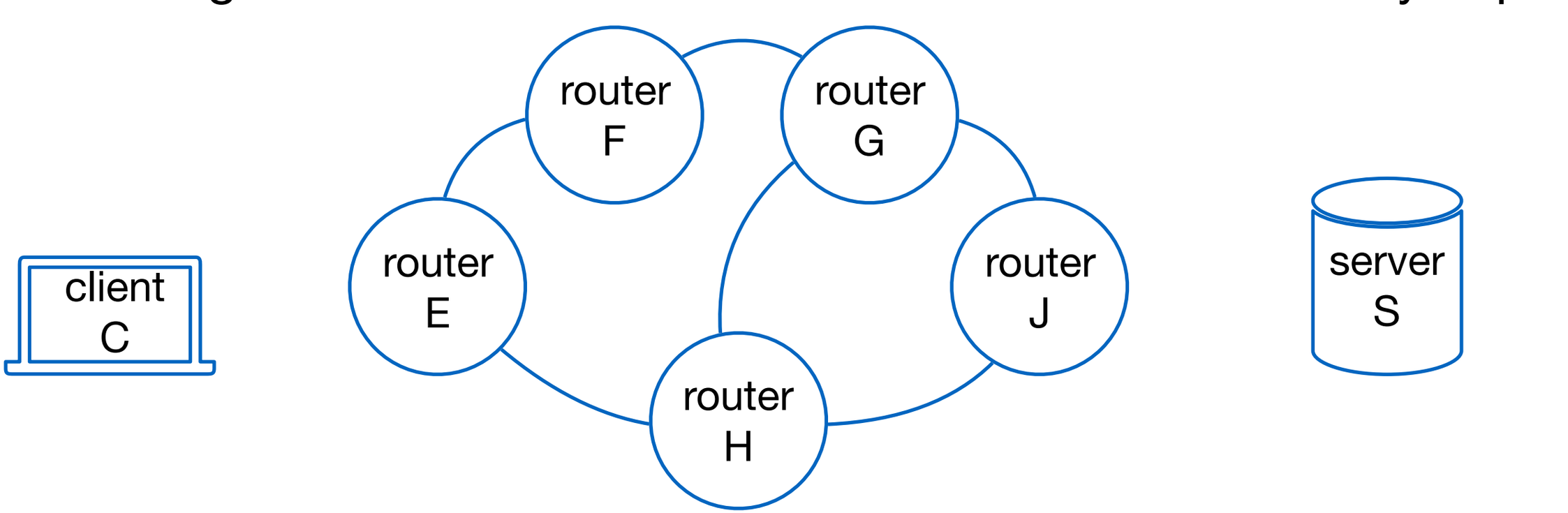
- Equal-cost multipath routing (ECMP) leads to multiple paths for the same source-destination pair...
  - ...hashing on fields that traditional Traceroute uses to identify hops



- Paris traceroute addresses this by holding flow identifiers constant.

Paris `traceroute`: making even better guesses about where the packets are going

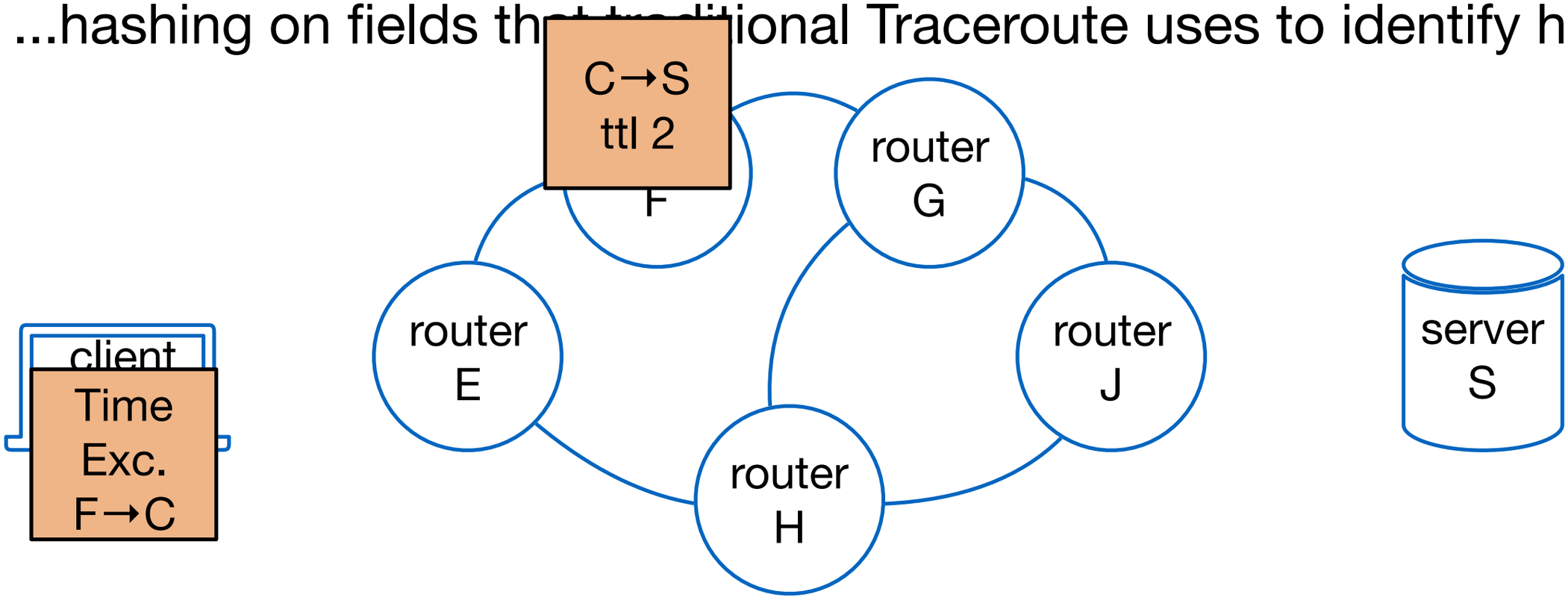
- Equal-cost multipath routing (ECMP) leads to multiple paths for the same source-destination pair...
  - ...hashing on fields that traditional Traceroute uses to identify hops



- Paris traceroute addresses this by holding flow identifiers constant.

Paris `traceroute`: making even better guesses about where the packets are going

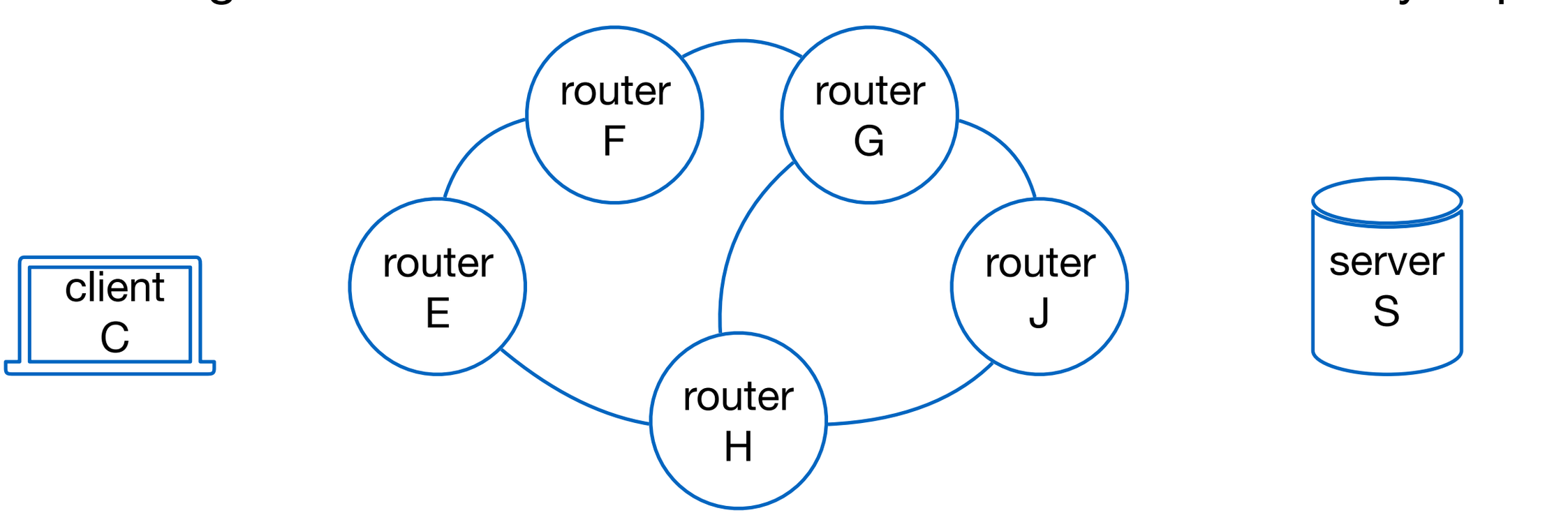
- Equal-cost multipath routing (ECMP) leads to multiple paths for the same source-destination pair...
  - ...hashing on fields that traditional Traceroute uses to identify hops



- Paris traceroute addresses this by holding flow identifiers constant.

Paris `traceroute`: making even better guesses about where the packets are going

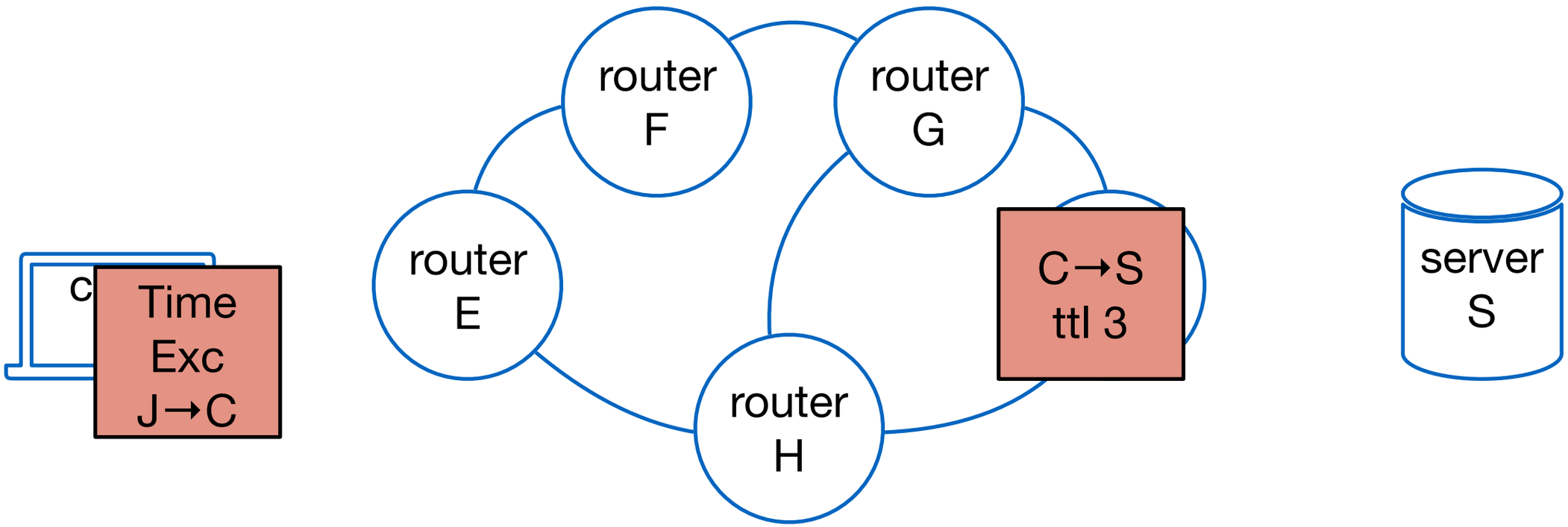
- Equal-cost multipath routing (ECMP) leads to multiple paths for the same source-destination pair...
  - ...hashing on fields that traditional Traceroute uses to identify hops



- Paris traceroute addresses this by holding flow identifiers constant.

Paris `traceroute`: making even better guesses about where the packets are going

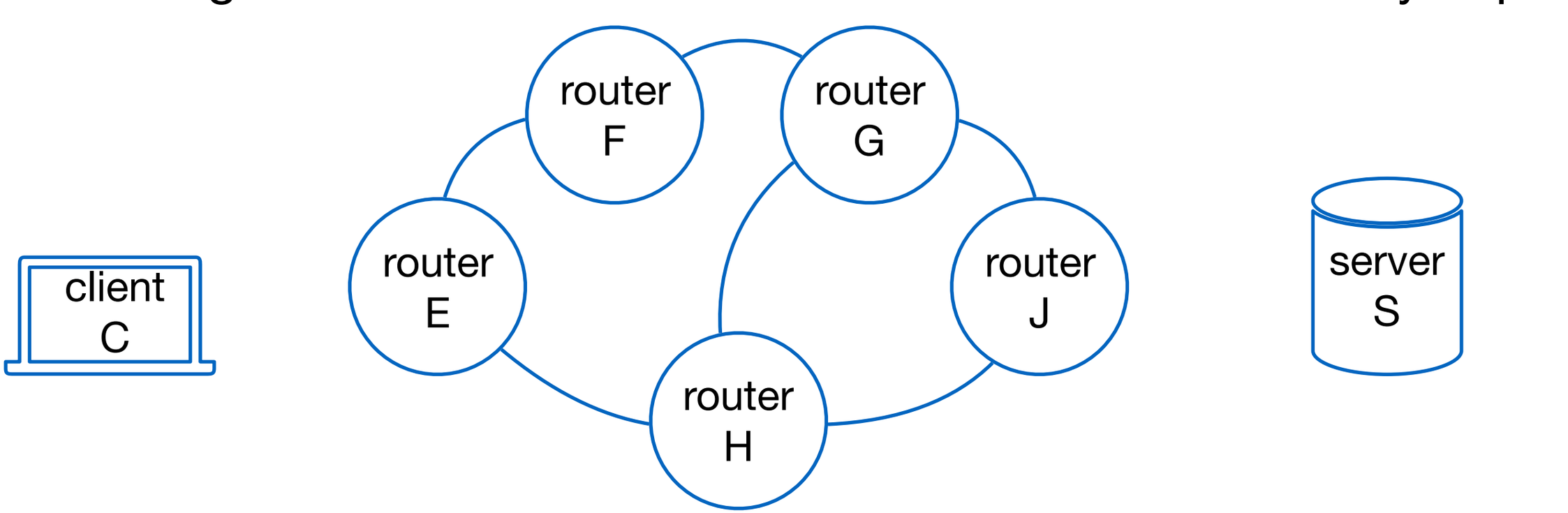
- Equal-cost multipath routing (ECMP) leads to multiple paths for the same source-destination pair...
  - ...hashing on fields that traditional Traceroute uses to identify hops



- Paris traceroute addresses this by holding flow identifiers constant.

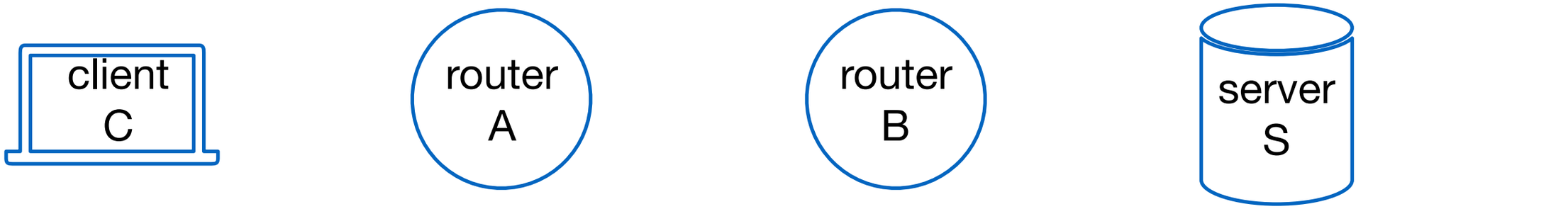
Paris `traceroute`: making even better guesses about where the packets are going

- Equal-cost multipath routing (ECMP) leads to multiple paths for the same source-destination pair...
  - ...hashing on fields that traditional Traceroute uses to identify hops

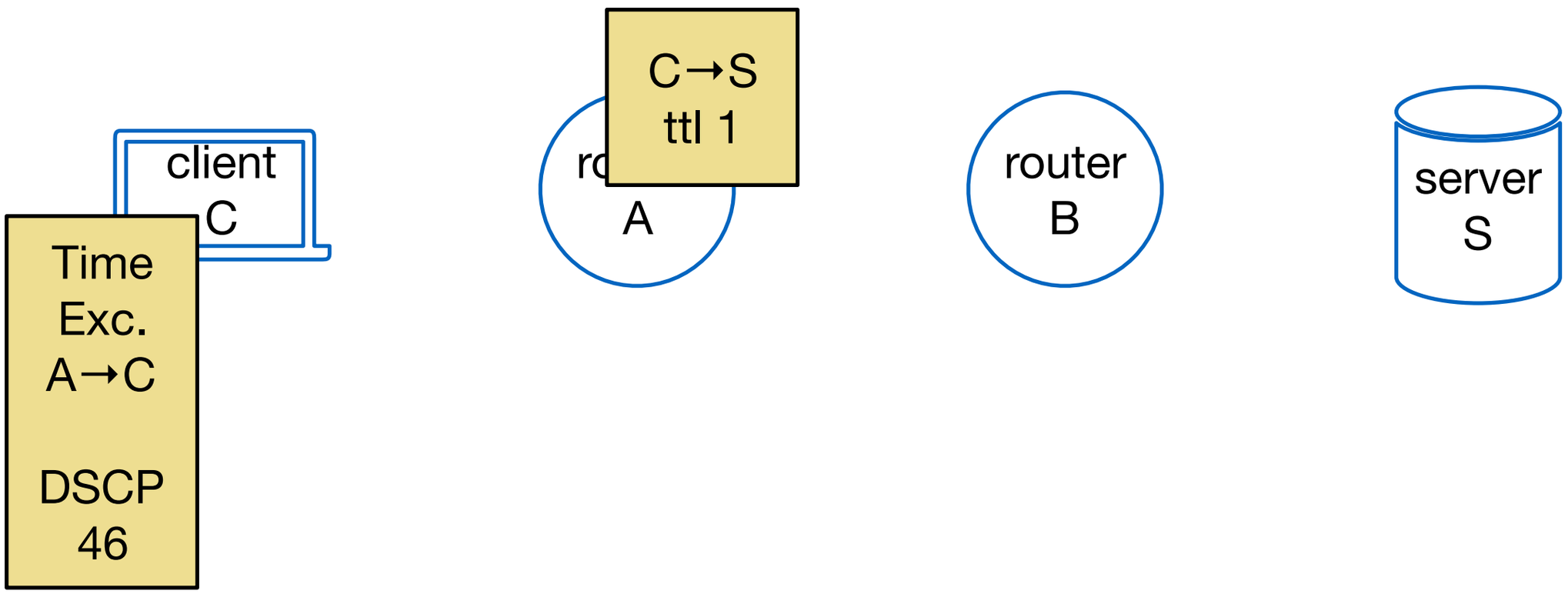


- Paris traceroute addresses this by holding flow identifiers constant.

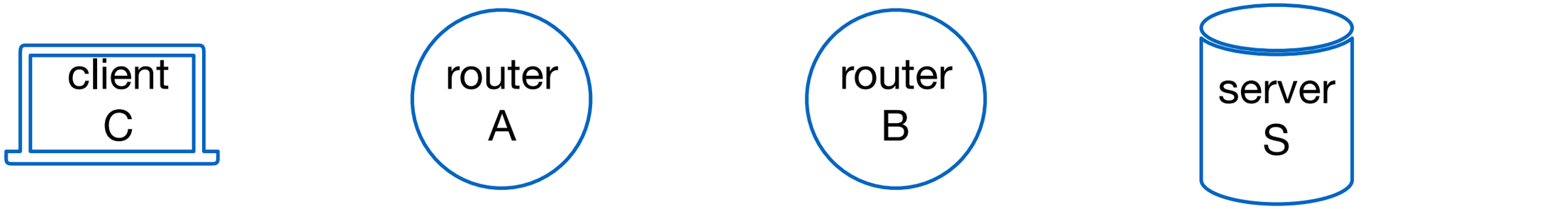
Tracebox: getting the path to tell you how it's messing with your packets



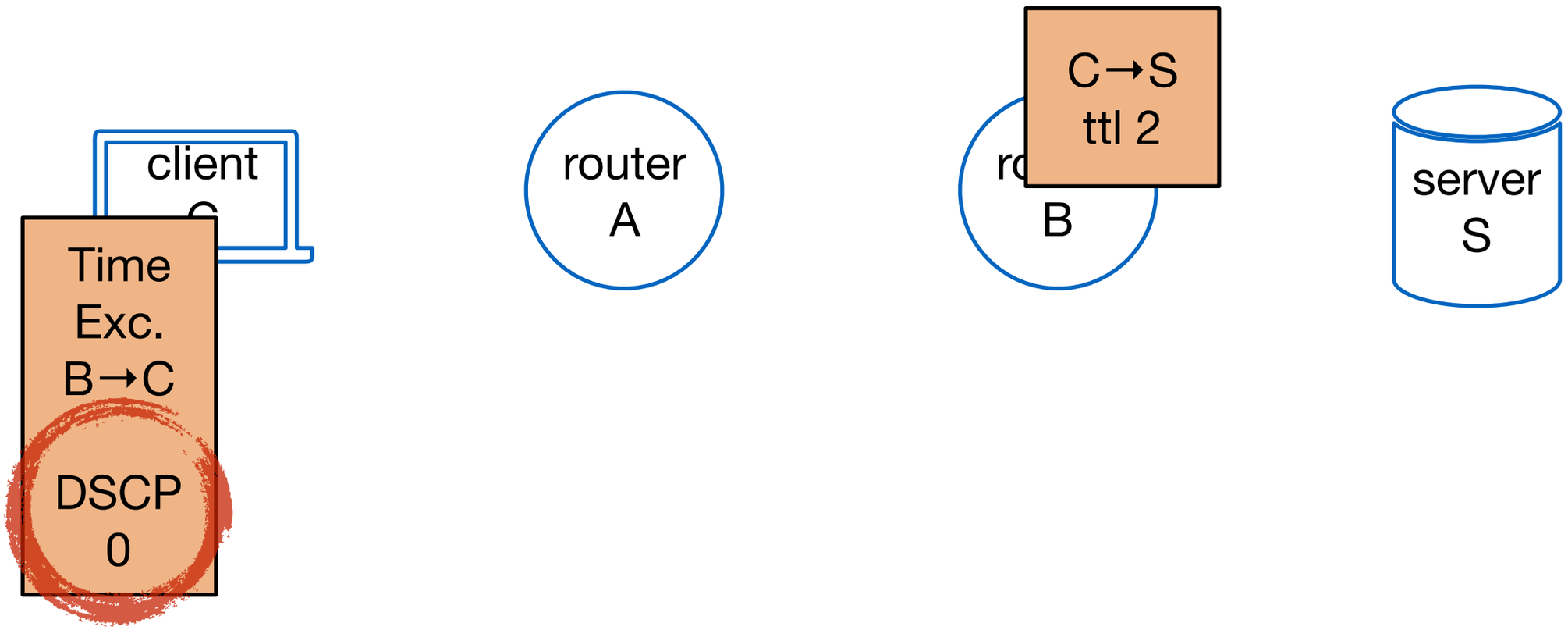
Tracebox: getting the path to tell you how it's messing with your packets



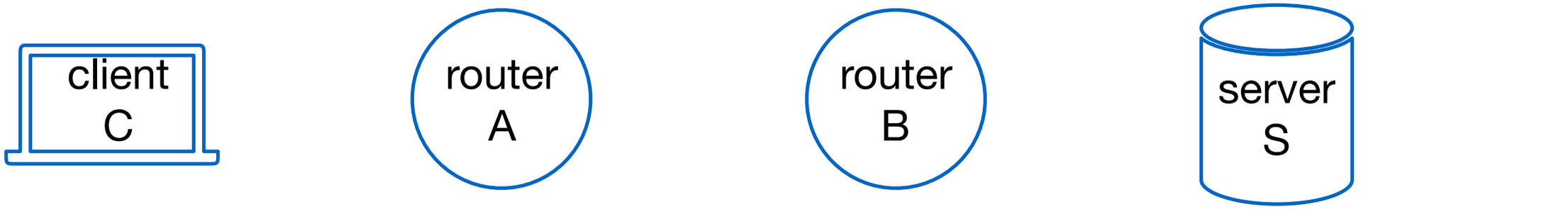
Tracebox: getting the path to tell you how it's messing with your packets



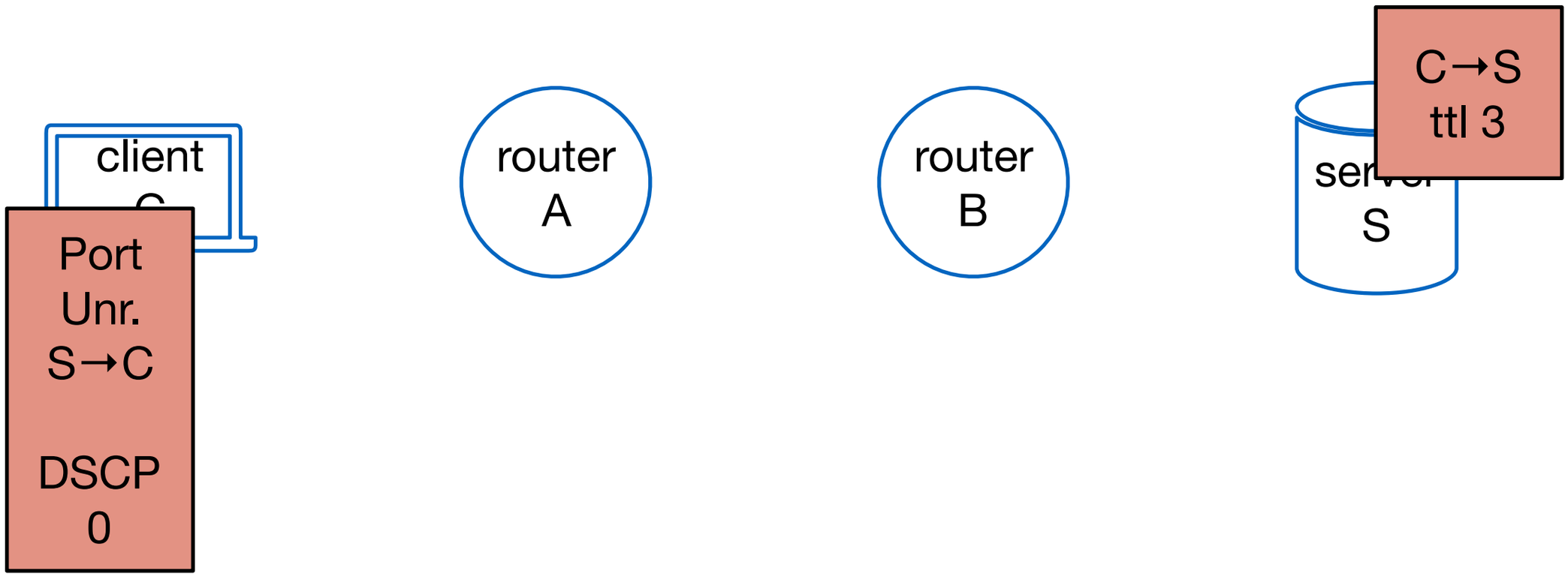
Tracebox: getting the path to tell you how it's messing with your packets



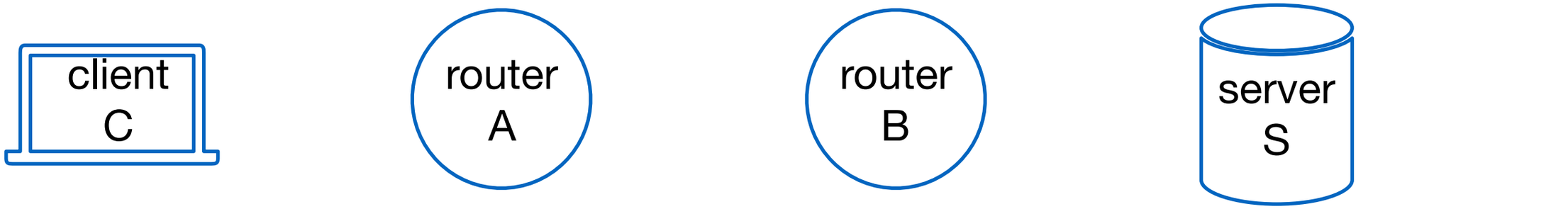
Tracebox: getting the path to tell you how it's messing with your packets



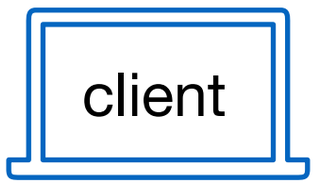
Tracebox: getting the path to tell you how it's messing with your packets



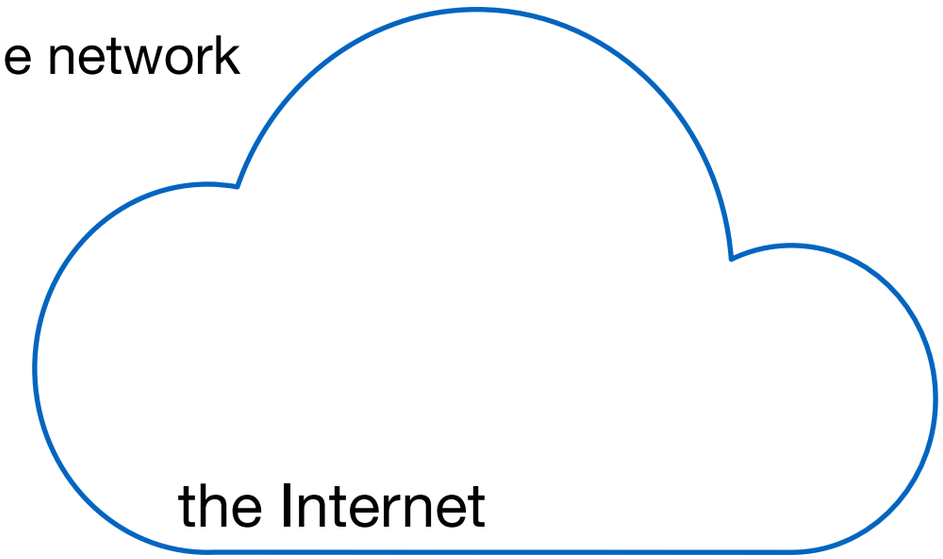
Tracebox: getting the path to tell you how it's messing with your packets



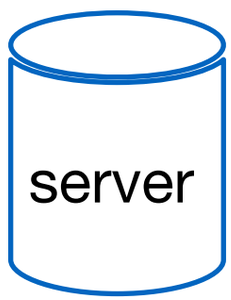
Path transparency measurement (a specific active measurement task)



- The Internet is notionally *transparent:*
  - packets come out the other end of the pipe unchanged.
  - based on the *end-to-end principle*: a maximally capable network made of smart endpoints connected by dumb pipes

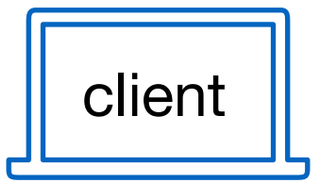


- This is not how things actually are, especially at layer 4:
  - Network address translation
  - Extension and option blocking and stripping
  - TCP ACK/SEQ rewriting
  - etc, etc, etc, etc...

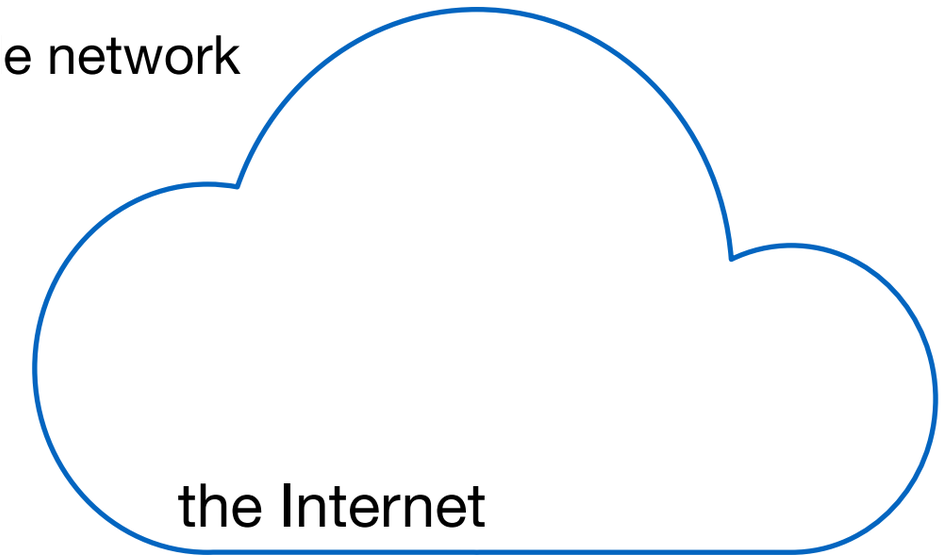


- Designing protocols and protocol extensions that can deal with interference require us to understand the nature and prevalence of different kinds of interference

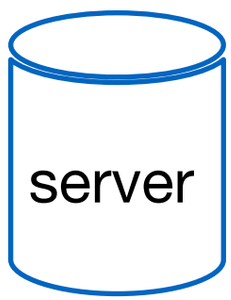
Path transparency measurement (a specific active measurement task)



- The Internet is notionally *transparent:*
  - packets come out the other end of the pipe unchanged.
  - based on the *end-to-end principle*: a maximally capable network made of smart endpoints connected by dumb pipes



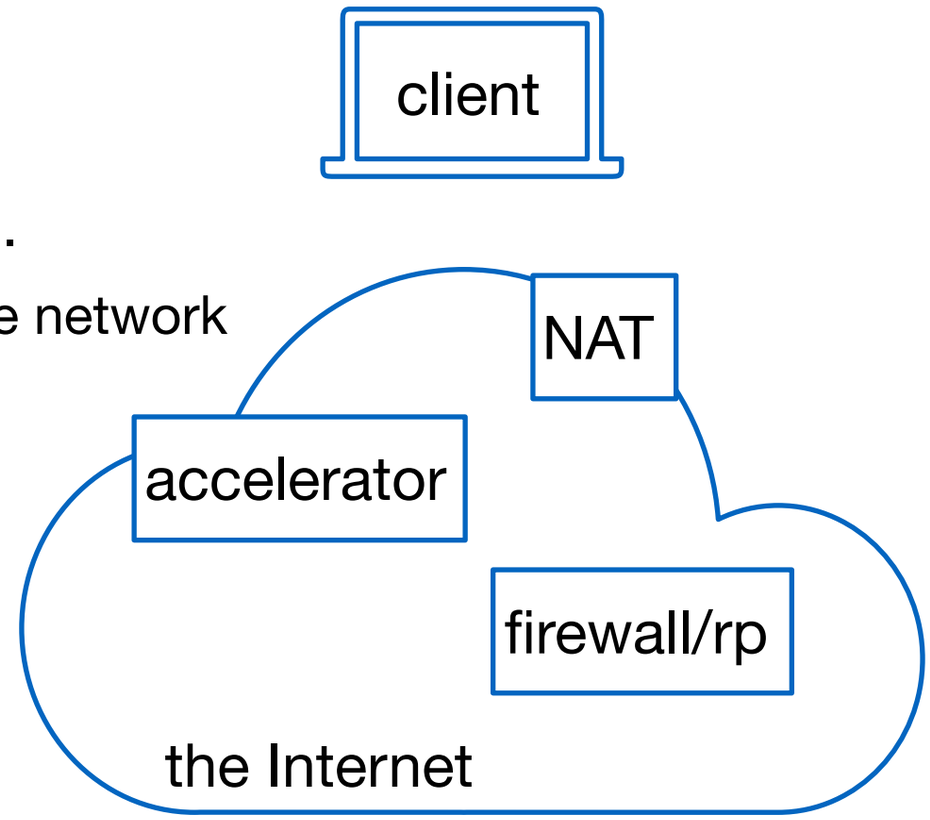
- This is not how things actually are, especially at layer 4:
  - Network address translation
  - Extension and option blocking and stripping
  - TCP ACK/SEQ rewriting
  - etc, etc, etc, etc...



- Designing protocols and protocol extensions that can deal with interference require us to understand the nature and prevalence of different kinds of interference

Path transparency measurement (a specific active measurement task)

- The Internet is notionally *transparent:*
  - packets come out the other end of the pipe unchanged.
  - based on the *end-to-end principle*: a maximally capable network made of smart endpoints connected by dumb pipes

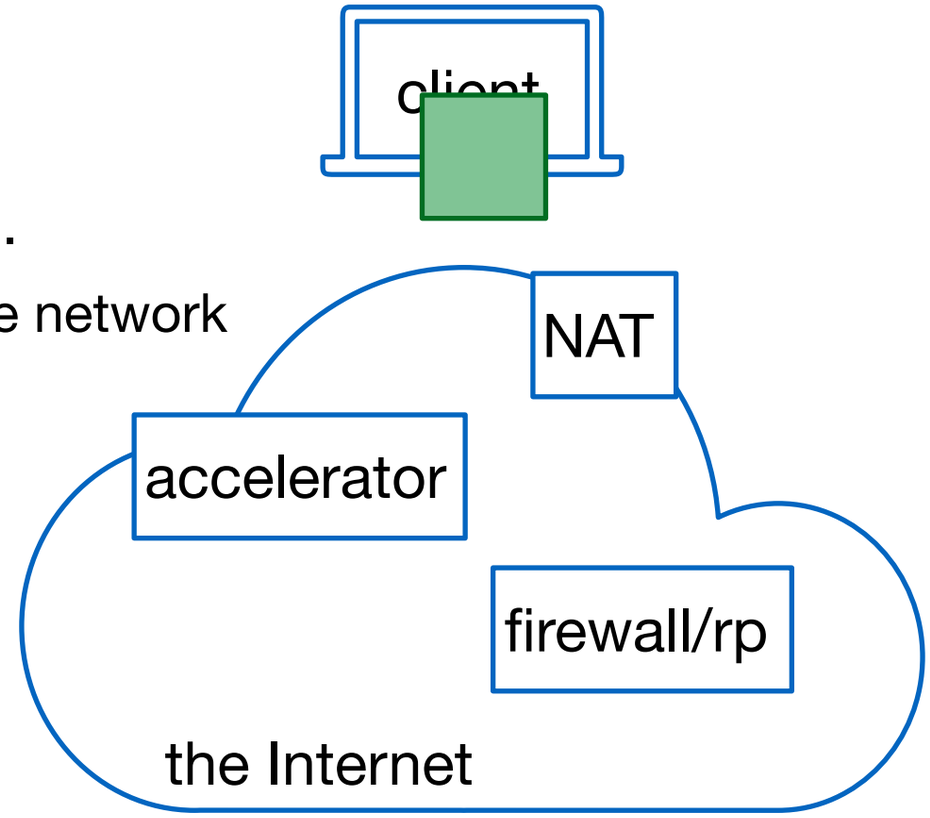


- This is not how things actually are, especially at layer 4:
  - Network address translation
  - Extension and option blocking and stripping
  - TCP ACK/SEQ rewriting
  - etc, etc, etc, etc...

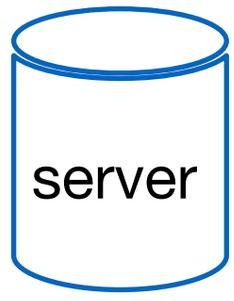
- Designing protocols and protocol extensions that can deal with interference require us to understand the nature and prevalence of different kinds of interference

Path transparency measurement (a specific active measurement task)

- The Internet is notionally *transparent:*
  - packets come out the other end of the pipe unchanged.
  - based on the *end-to-end principle*: a maximally capable network made of smart endpoints connected by dumb pipes



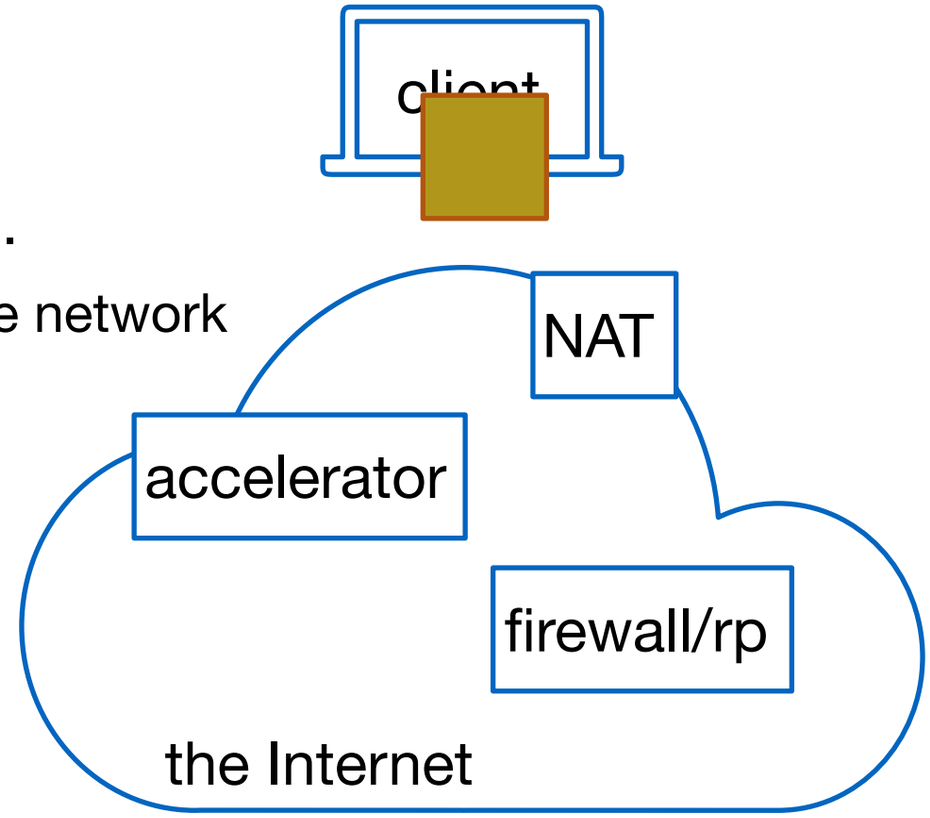
- This is not how things actually are, especially at layer 4:
  - Network address translation
  - Extension and option blocking and stripping
  - TCP ACK/SEQ rewriting
  - etc, etc, etc, etc...



- Designing protocols and protocol extensions that can deal with interference require us to understand the nature and prevalence of different kinds of interference

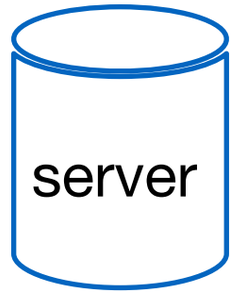
Path transparency measurement (a specific active measurement task)

- The Internet is notionally *transparent:*
  - packets come out the other end of the pipe unchanged.
  - based on the *end-to-end principle*: a maximally capable network made of smart endpoints connected by dumb pipes



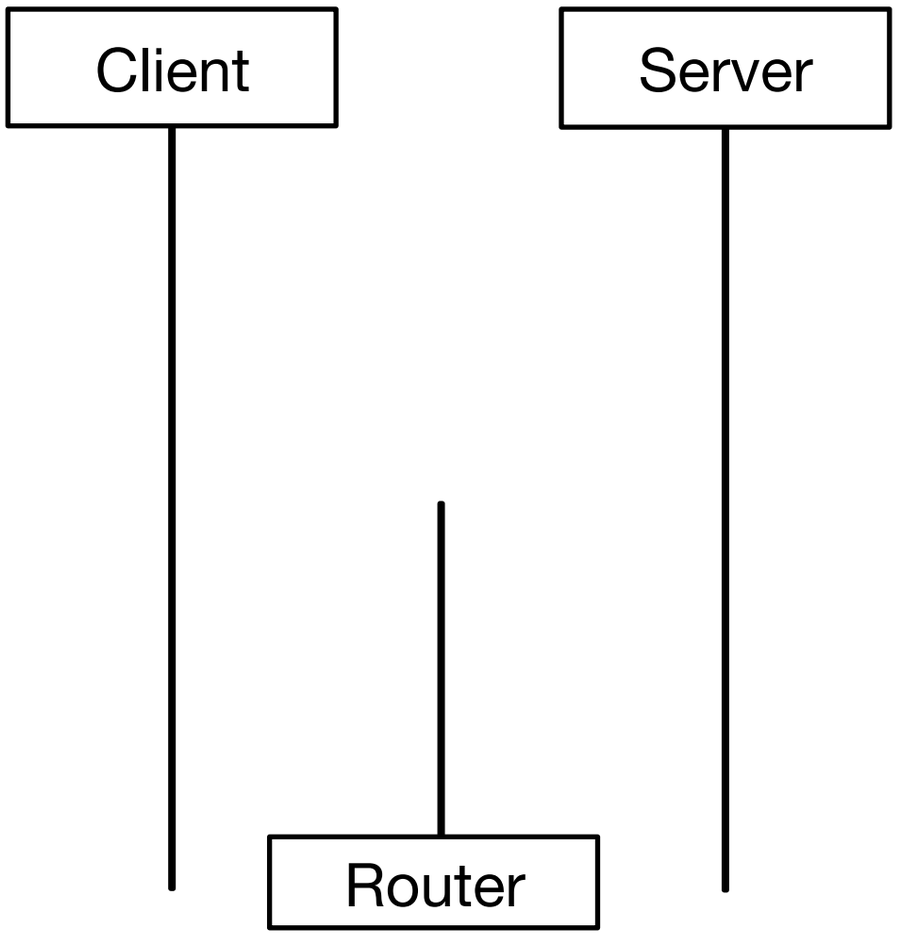
- This is not how things actually are, especially at layer 4:
  - Network address translation
  - Extension and option blocking and stripping
  - TCP ACK/SEQ rewriting
  - etc, etc, etc, etc...

- Designing protocols and protocol extensions that can deal with interference require us to understand the nature and prevalence of different kinds of interference



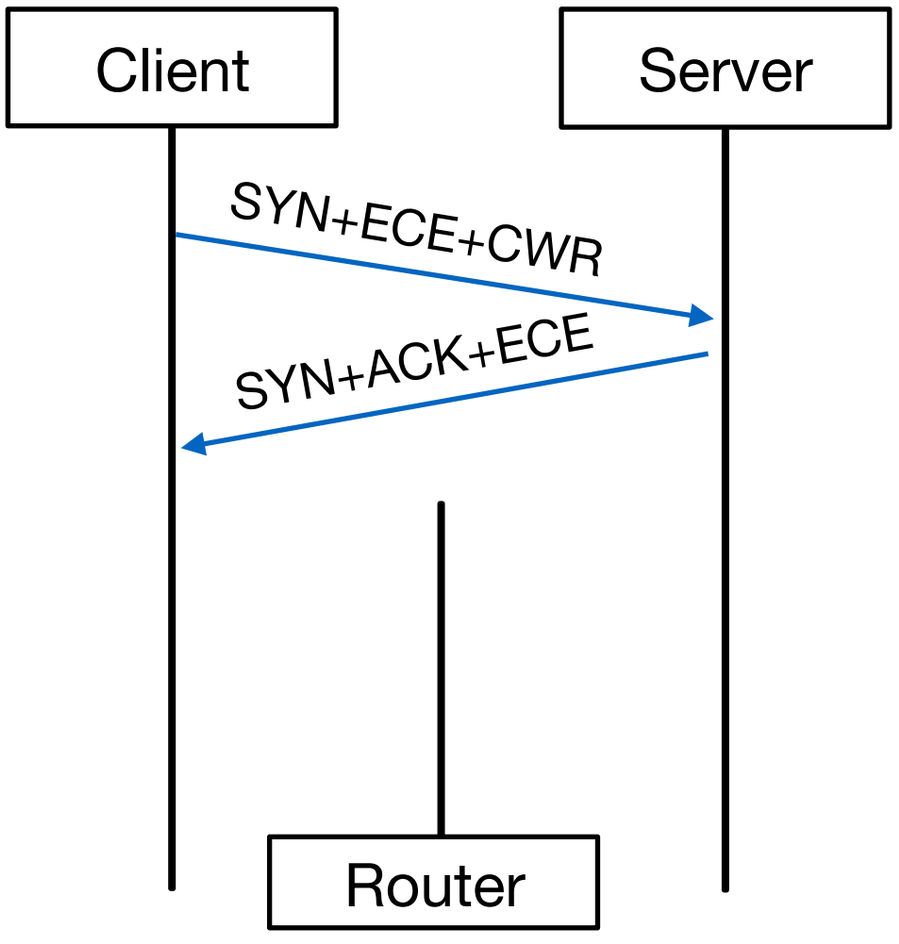
Explicit Congestion Notification (ECN) (a specific "new" feature we'll measure today)

TCP extension allowing routers to signal congestion using bits in the IP header



Explicit Congestion Notification (ECN) (a specific "new" feature we'll measure today)

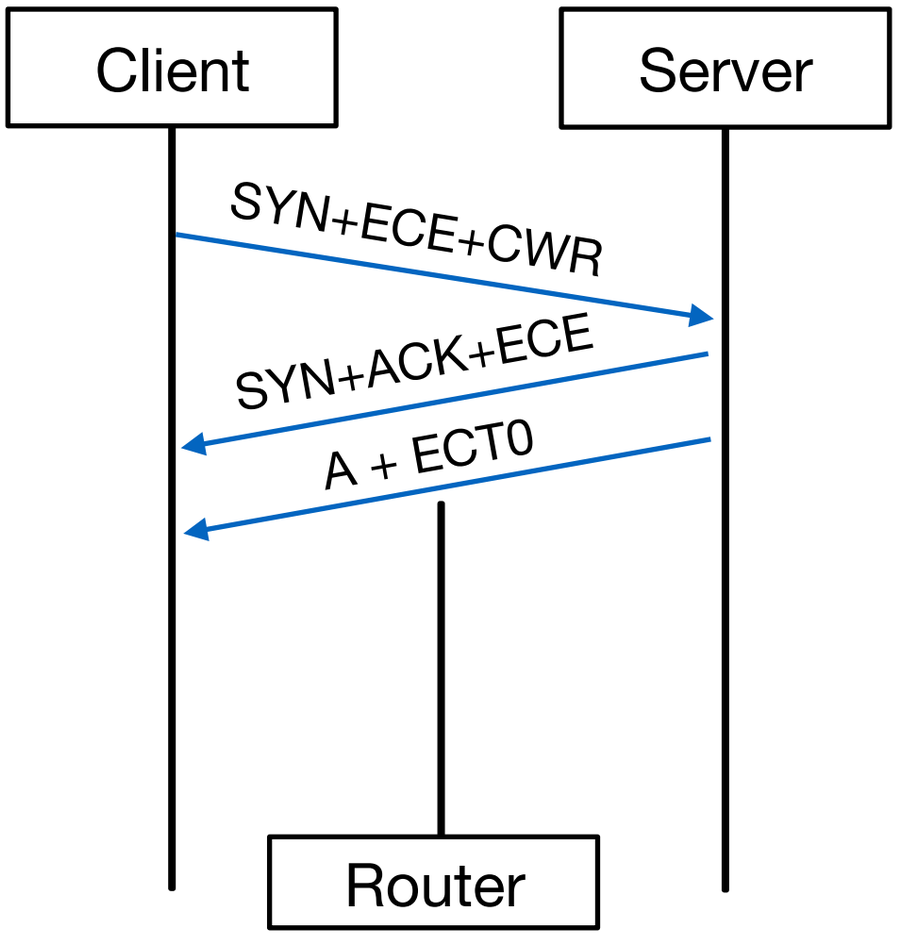
TCP extension allowing routers to signal congestion using bits in the IP header



- Client attempts to negotiate ECN with TCP flags

Explicit Congestion Notification (ECN) (a specific "new" feature we'll measure today)

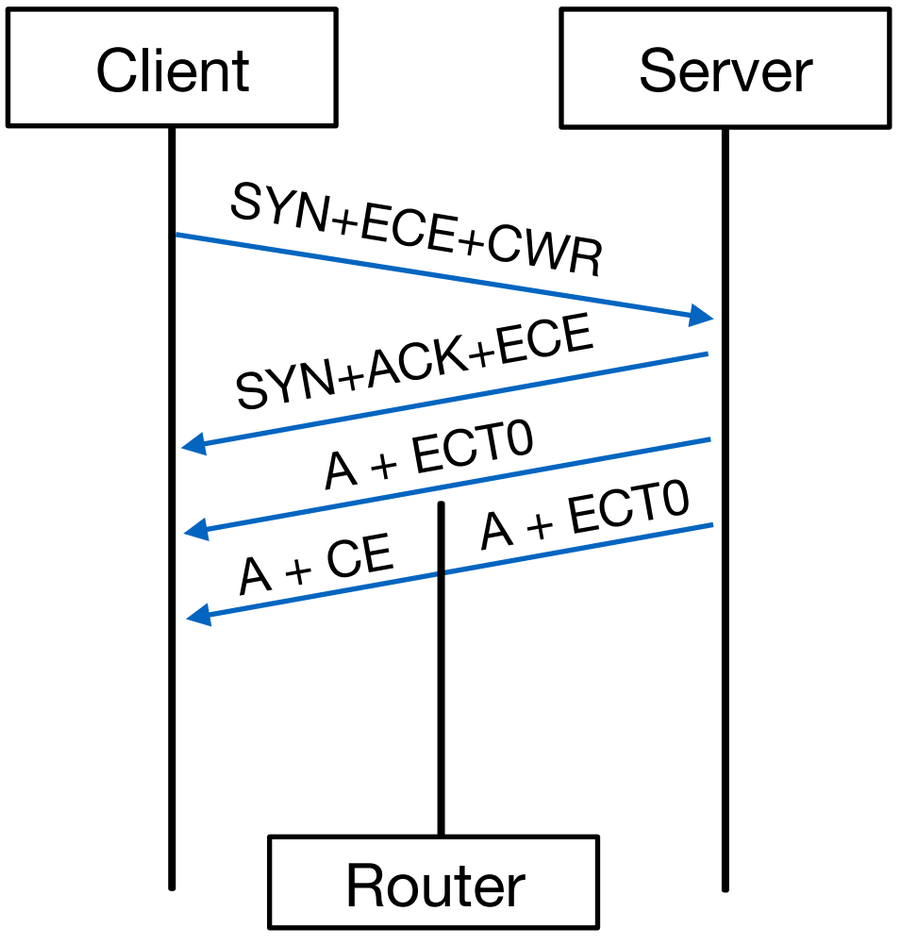
TCP extension allowing routers to signal congestion using bits in the IP header



- Client attempts to negotiate ECN with TCP flags
- Server acknowledges negotiation and begins signaling ECN Capable (ECT0)

Explicit Congestion Notification (ECN) (a specific "new" feature we'll measure today)

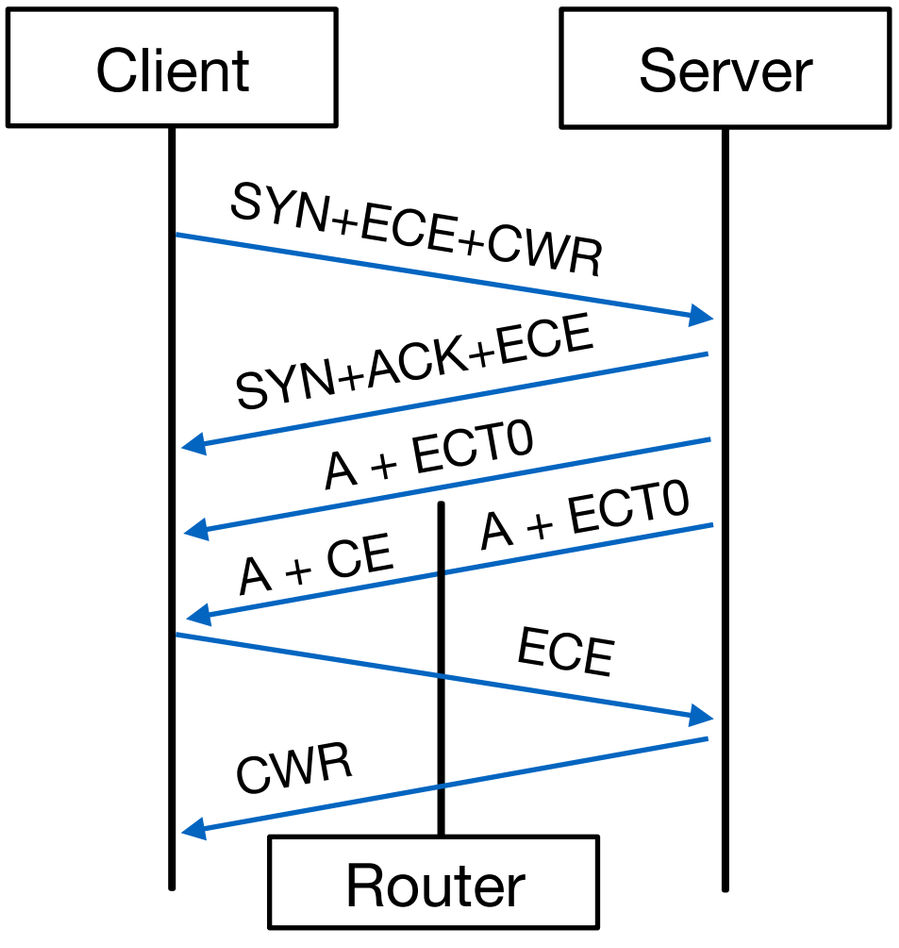
TCP extension allowing routers to signal congestion using bits in the IP header



- Client attempts to negotiate ECN with TCP flags
- Server acknowledges negotiation and begins signaling ECN Capable (ECT0)
- Routers on path can mutate ECT0 to CE to note congestion (instead of tail dropping)

Explicit Congestion Notification (ECN) (a specific "new" feature we'll measure today)

TCP extension allowing routers to signal congestion using bits in the IP header



- Client attempts to negotiate ECN with TCP flags
- Server acknowledges negotiation and begins signaling ECN Capable (ECT0)
- Routers on path can mutate ECT0 to CE to note congestion (instead of tail dropping)
- Client echoes CE, server acks reduction of congestion window

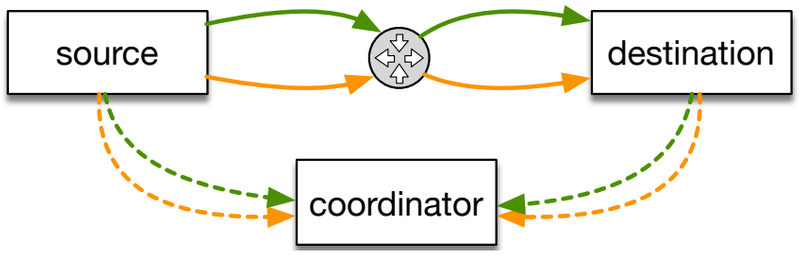
The History of ECN

- Defined in RFC 3168 in 2001
  - Reused two bits from old IP TOS byte for ECT/CE signaling in the IP header
- Deployment was slow due to early problems
  - ECN-marked traffic caused some routers to reboot
- Servers default to negotiate after ~2010, ~80% today.
- Clients default to negotiation recently (iOS, systemd)
- ~2% of Internet paths have some ECN signaling problems
  - Often related to treating ECN as if it were still TOS!

General approaches for finding out what's happening on an Internet path

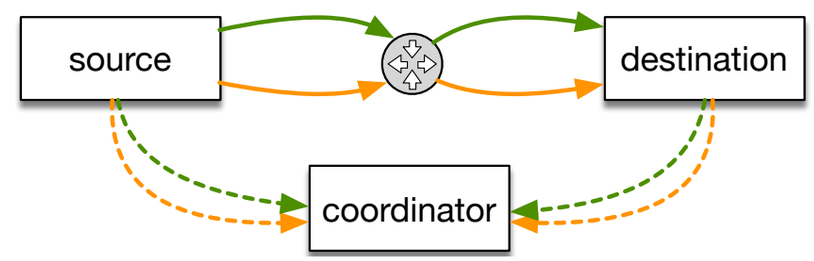
- Throw packets at it and see what happens.
- Ideally: control both endpoints and compare what you send and what you receive.
- More scalable: control one endpoint toward many public endpoints, infer state based on protocol behavior.
- Tomography and inference: do the same measurement from multiple vantage points and compare.

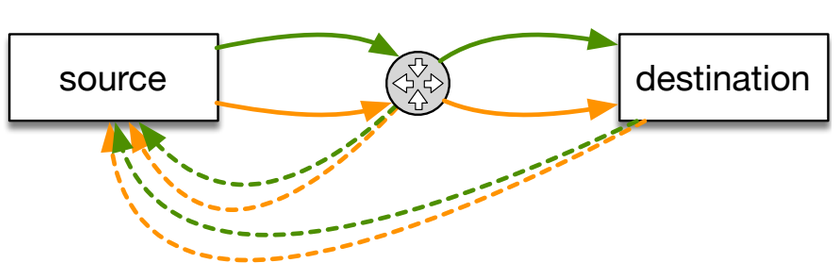
General approaches for finding out what's happening on an Internet path



- Throw packets at it and see what happens.
- Ideally: control both endpoints and compare what you send and what you receive.
- More scalable: control one endpoint toward many public endpoints, infer state based on protocol behavior.
- Tomography and inference: do the same measurement from multiple vantage points and compare.

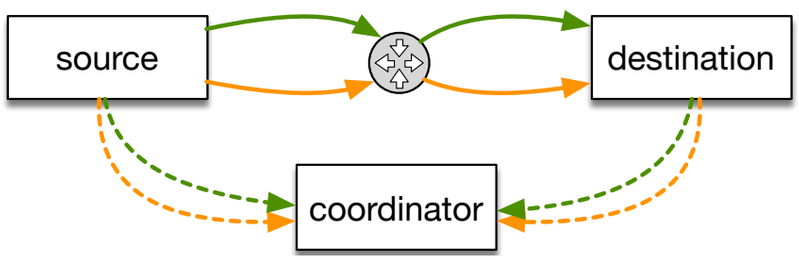
General approaches for finding out what's happening on an Internet path

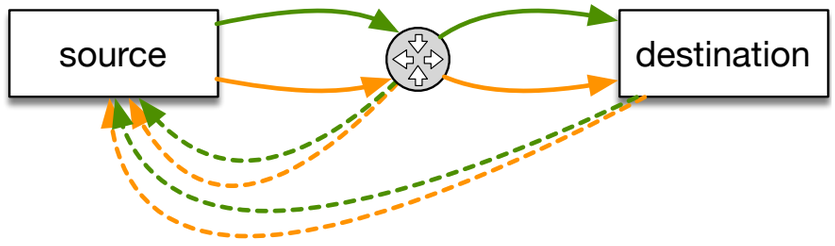


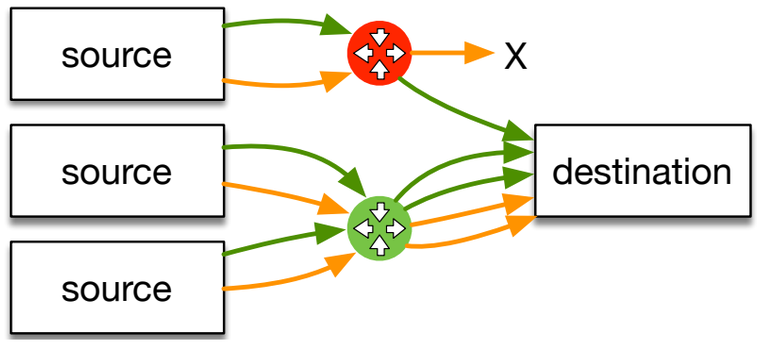


- Throw packets at it and see what happens.
- Ideally: control both endpoints and compare what you send and what you receive.
- More scalable: control one endpoint toward many public endpoints, infer state based on protocol behavior.
- Tomography and inference: do the same measurement from multiple vantage points and compare.

General approaches for finding out what's happening on an Internet path

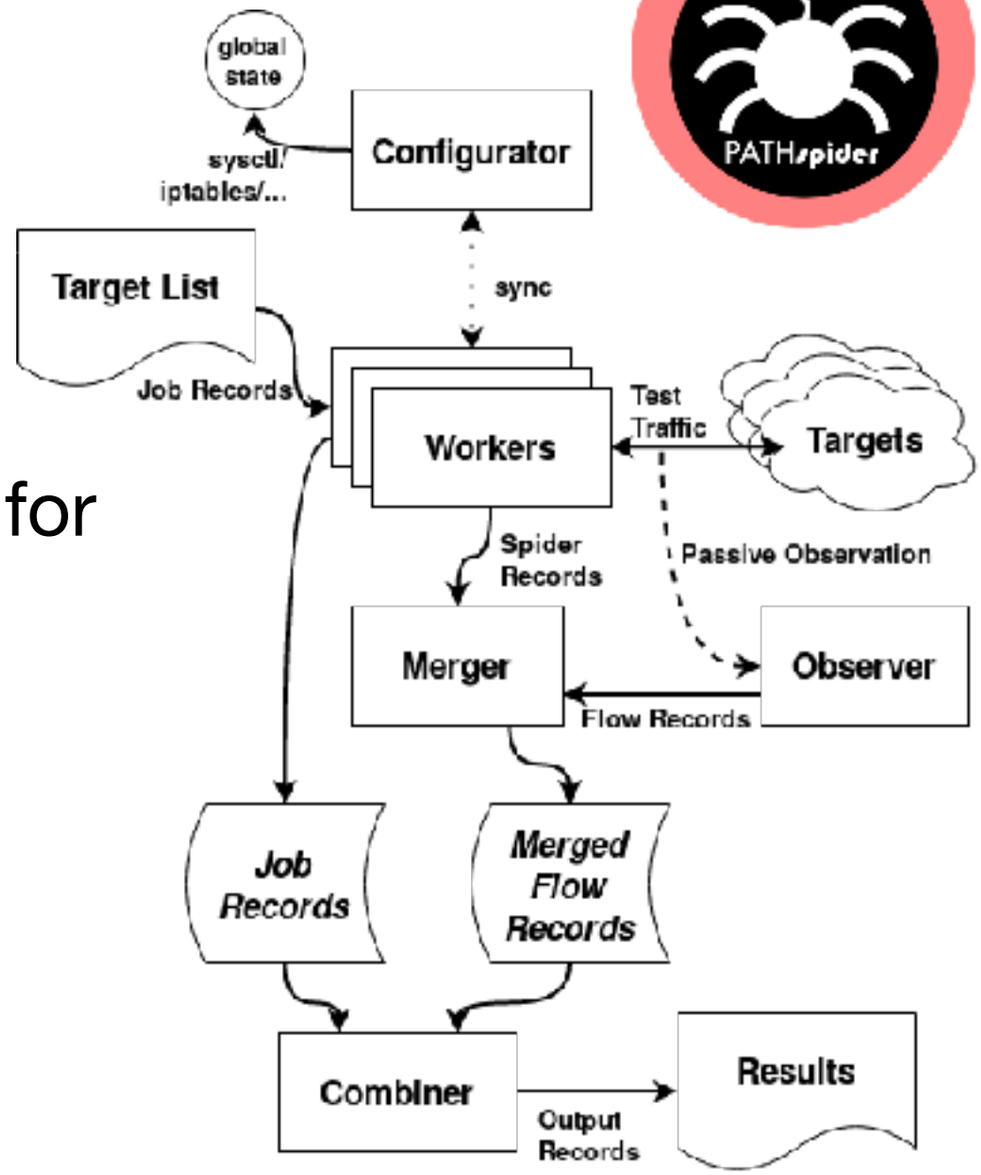






- Throw packets at it and see what happens.
- Ideally: control both endpoints and compare what you send and what you receive.
- More scalable: control one endpoint toward many public endpoints, infer state based on protocol behavior.
- Tomography and inference: do the same measurement from multiple vantage points and compare.

PATHspider 2.0



- Generalized framework for A/B testing
- Plugin-based architecture with plugins for
  - ECN connectivity and negotiation
  - DiffServ Codepoints
  - TCP Fast Open
  - …
- Result outputs **Path Observations**
  - where a certain *condition* (e.g. `ecn.negotiation.succeeded`) has been observed at a certain point of *time* on a certain *path*

Scapy: a toolkit for building anything else you might need

- Packet manipulation tool to create or decode packets header by header, byte by byte.
  - Can be used with ForgeSpider plugin model to PathSpider, as we'll see later.

- Solidly in "hack up a quick little script" territory → be careful with metadata/provenance for these tools

A theme emerges: details matter.

- The details of...
  - what *exactly* a tool does
  - how that tool was configured
  - the context in which a measurement was taken
- are important.

- All tools have limitations.
- Remembering what you did and how is almost as important as trying to do the right thing in the first place.

A few simple principles go a long way.

- Pulling data together into a ***simple schema*** with a small set of metrics/semantics makes comparability possible.
- Measurement ***metadata*** is important: it gives you a way to express what you did and how in a way that you (and others!) can use later.
  - There's a tradeoff between ad-hoc (ease of writing) and standard-structured (ease of use) schemas here.
- ***Provenance*** is also important: knowing where every bit of data and analysis came from allows repeatability
  - Doing this all the way down to commit references / software repository release number for measurement and analysis tools makes reproducibility easier.

Today's tutorial:

- Korian Edeline on Tracebox (with exercises)
  - measurement of middlebox interference on the path
- (Coffee Break)
- Iain Learmonth on PATHspider (with exercises)
  - path transparency measurement end to end
- Brian Trammell on Observatories...
  - collection, preservation, metadata and provenance
- ...and the PTO (with exercises)
  - pulling it all together

Preparation [do this first]

- Tracebox VM
  - install git, vagrant, virtualbox
  - `$ git clone https://github.com/mami-project/vpp-mb -b rcm`
  - `$ cd vagrant && vagrant up`
  - note: minimum RAM ~4G

- PATHspider / PTO VM
  - you need an ssh client, and we need your public key
  - measurement VM is running on DigitalOcean
    - get username, hostname, and API token from me.