Network Security and Measurement

- Data Plane Measurements -

Prof. Dr. Thomas Schmidt

http://inet.haw-hamburg.de | t.schmidt@haw-hamburg.de

Agenda

How to obtain data plane measurements?

Passive measurements:

- Traffic classification
- Monitoring Flows
- IPFIX (IP Flow Information Export)

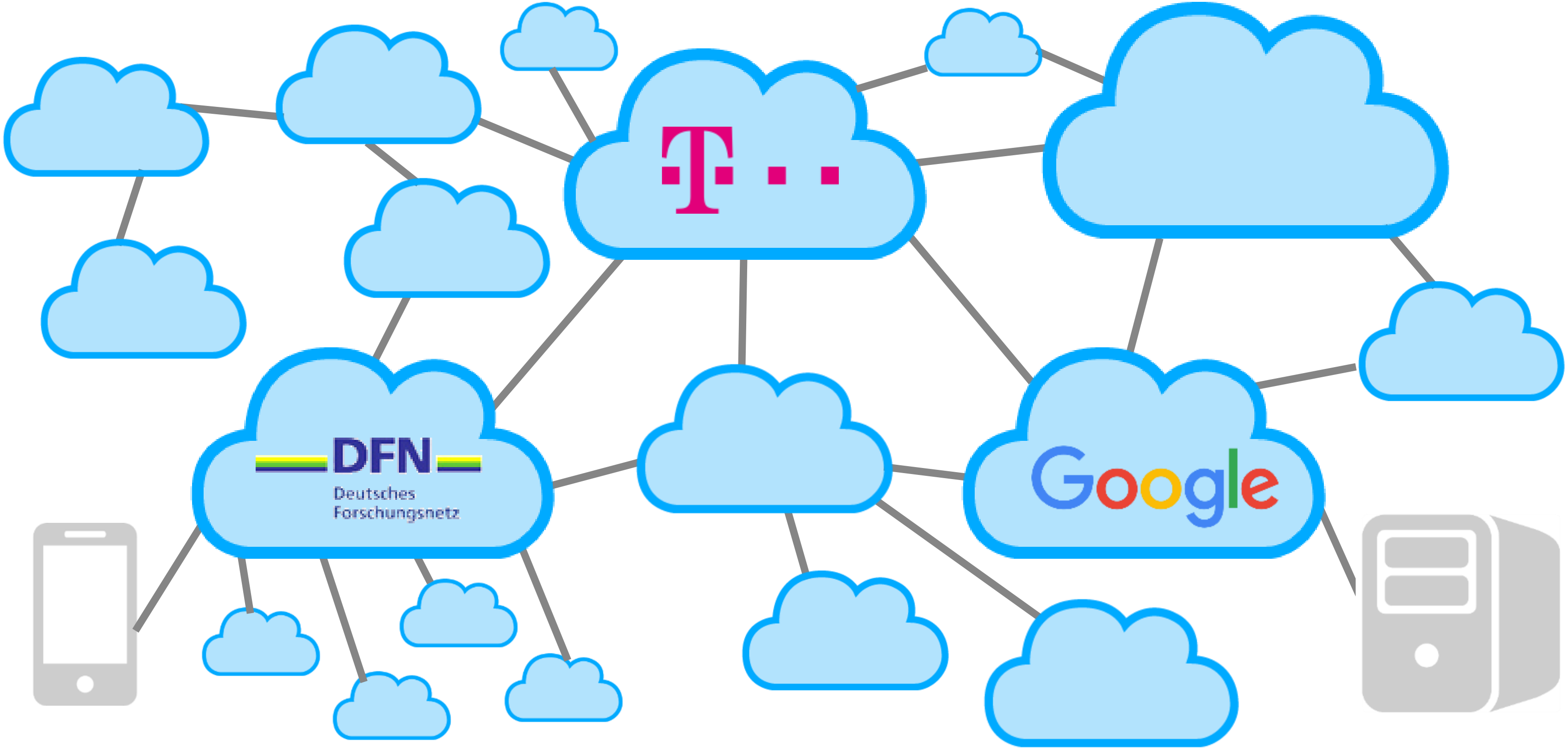
Active measurements:

- Challenges and good practice
- Traceroute measurements are not trivial
- Active measurement infrastructures

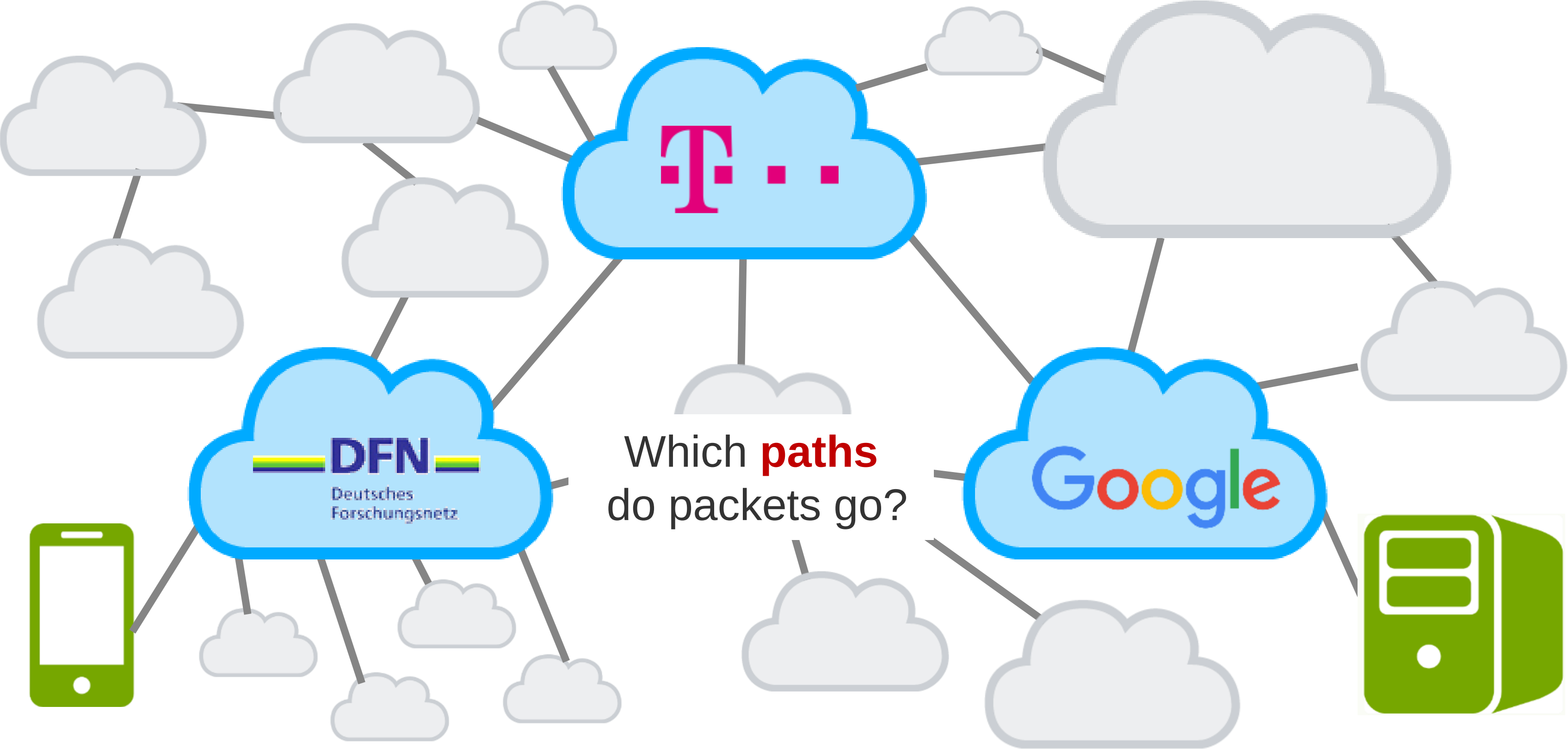
Technical Challenge

MEASURING THE DATA PLANE

From control to data plane

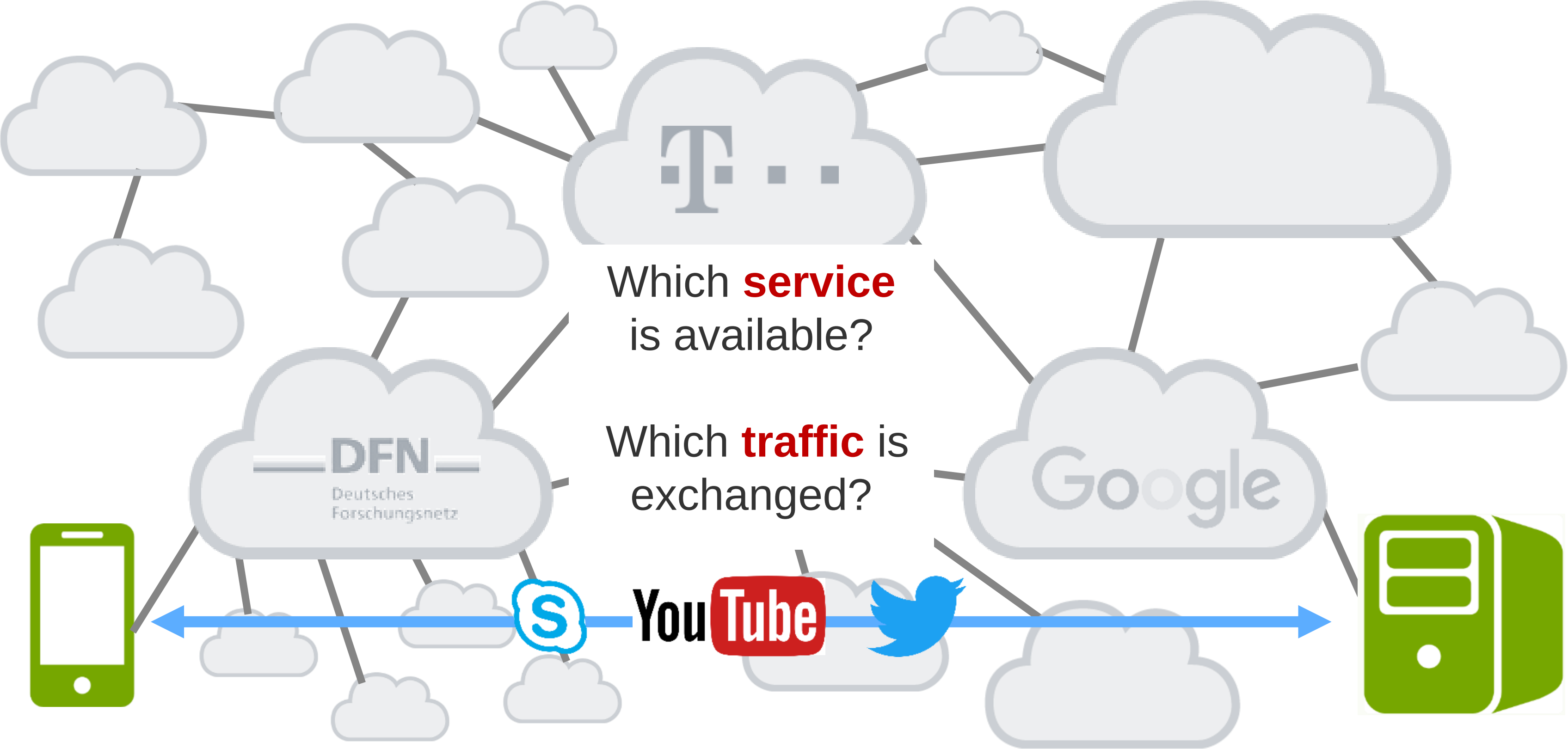


From control to data plane



Which **paths** do packets go?

From control to data plane



Which **service** is available?

Which **traffic** is exchanged?

Why should we measure the data plane?

Protocol deployment

Network provisioning

Security

…

How to measure the data plane?

	Active	Passive
Examples	Ping, traceroute, scanning, …	Traffic monitoring, log files, …

Listen and Record

PASSIVE DATA PLANE MEASUREMENTS

Passive data measurement introduces two questions

How to select traffic?

Sampling vs. full capture

How to classify the captured traffic?

Port-based vs. application payload

Full packet captures are not always achievable

Privacy requirements

Scalability challenges

Select only a subset of data, either in terms of packets or packet headers.

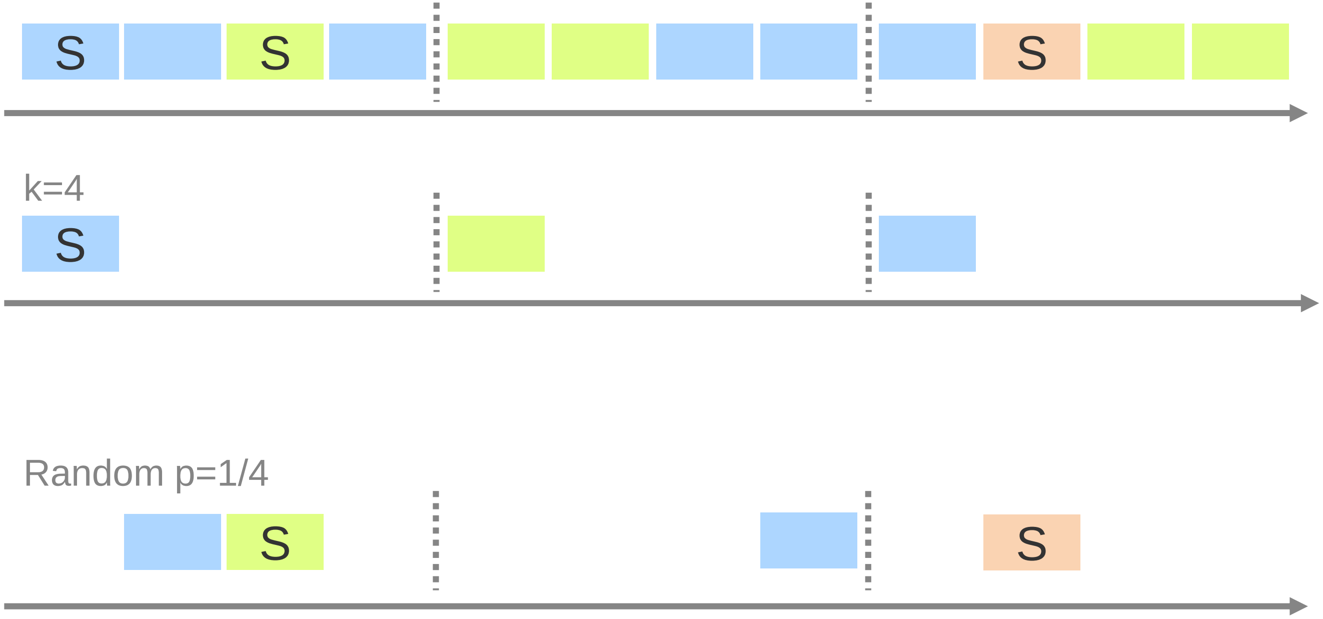
Filtering

“Filtering is the deterministic selection of packets based on the Packet Content, the treatment of the packet at the Observation Point, or deterministic functions of these occurring in the Selection State.” [RFC 5475]

Sampling

“Sampling is targeted at the selection of a representative subset of packets. The subset is used to infer knowledge about the whole set of observed packets without processing them all. The selection can depend on packet position, and/or on Packet Content, and/or on (pseudo) random decisions.” [RFC 5475]

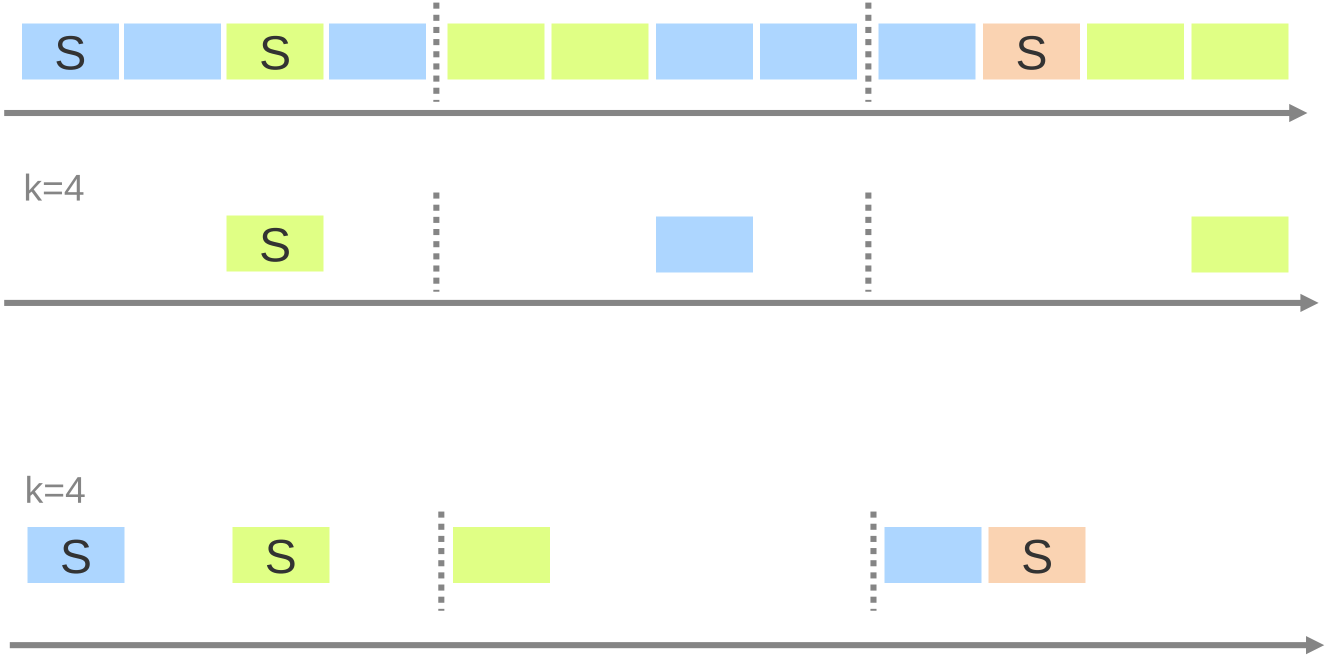
Two basic sampling policies



Systematic sampling
Deterministic selection of every 1-out-of-k elements

Random sampling
Probabilistic selection of elements

Composite sampling strategies



Stratified sampling

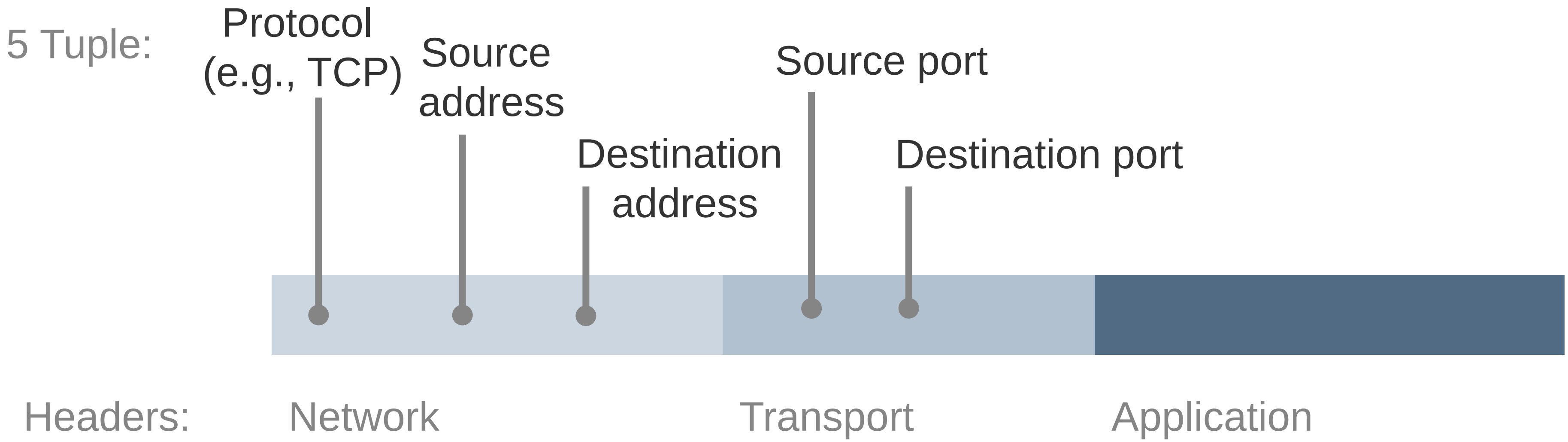
Leverage a priori information and group k consecutive elements, select one randomly within the group

Systematic SYN sampling

Filter all SYN packet and sample k packets

Sampling can be applied on a **per packet** base or **per flow** base.

A flow is typically defined by a 5 tuple



5 Tuple:

Protocol (e.g., TCP)

Source address

Destination address

Source port

Destination port

Headers:

Network

Transport

Application

Packet sampling: Example

Packet sampling uses randomness in the sampling process to prevent synchronization with any periodic patterns in the traffic.

Consider a link with 1,000,000 packets.

You sample 2,500 packets uniformly randomly (sampling rate 0,25%).

1,000 of the sampled packets belong to voice traffic.

How many of the 1M packets are most likely voice packets?

Packet sampling: Example

Packet sampling uses randomness in the sampling process to prevent synchronization with any periodic patterns in the traffic.

Consider a link with 1,000,000 packets.

You sample 2,500 packets uniformly randomly (sampling rate 0,25%).

1,000 of the sampled packets belong to voice traffic.

How many of the 1M packets are most likely voice packets?

400,000 packets, or 40% (1,000/2,500 = 0,4).

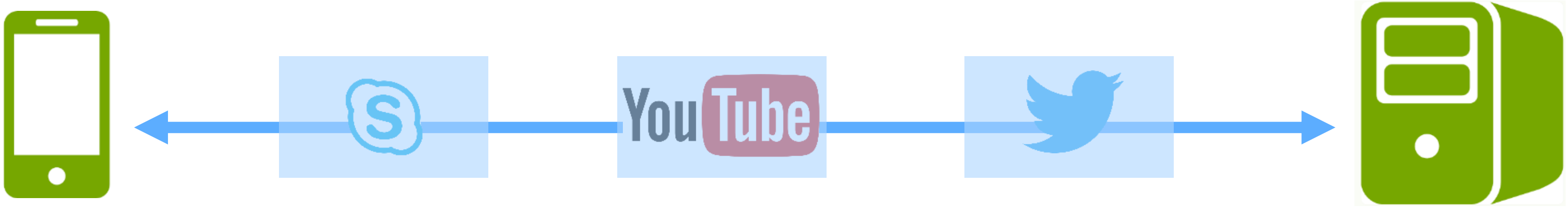
Sampling error

Measurement accuracy does not depend on the number of packets but on the number of samples.

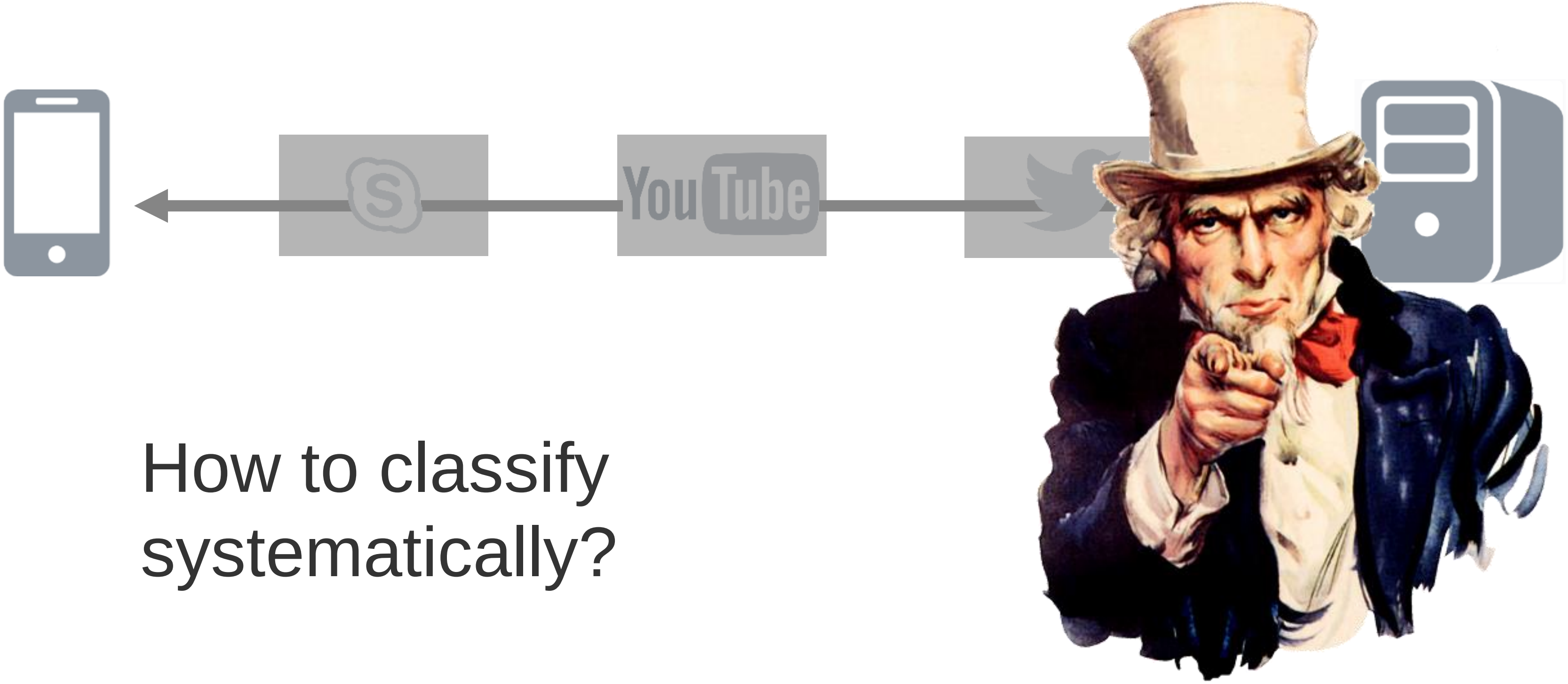
Accuracy can be improved by (i) increasing the sampling rate or (ii) or look at the data over longer time.

TRAFFIC CLASSIFICATION

Which packet belongs to which application?



Which packet belongs to which application?



How to classify systematically?

Traffic classification approaches

Port-based

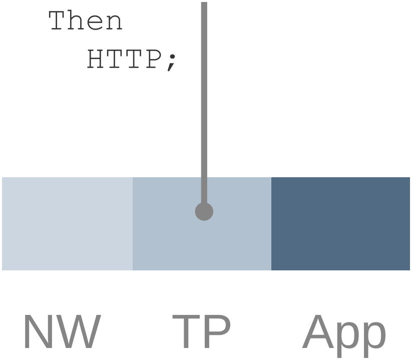
Payload-based

Host behavior-based

Flow feature-based

Port-based traffic classification

```
If
  TCP/SRC or TCP/DST == 80
Then
  HTTP;
```



Assumption

Many applications run on fixed ports

Advantage

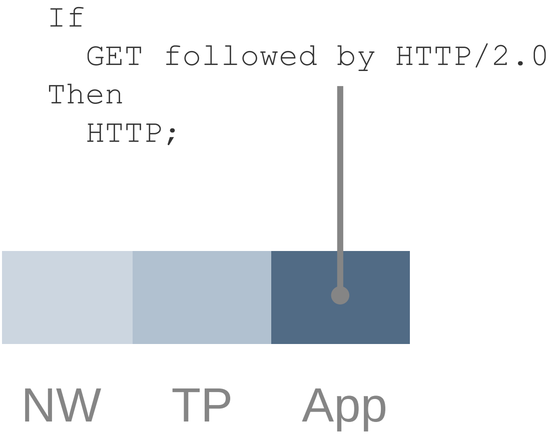
Simple and fast

Drawback

Assumption holds only in some scenarios
P2P apps use random ports, apps use well-known ports to obfuscate traffic etc.

High probability of misclassification

Payload-based traffic classification (or DPI)



```
If
  GET followed by HTTP/2.0
Then
  HTTP;
```

Assumption

Application layer protocol known

Advantage

Very accurate

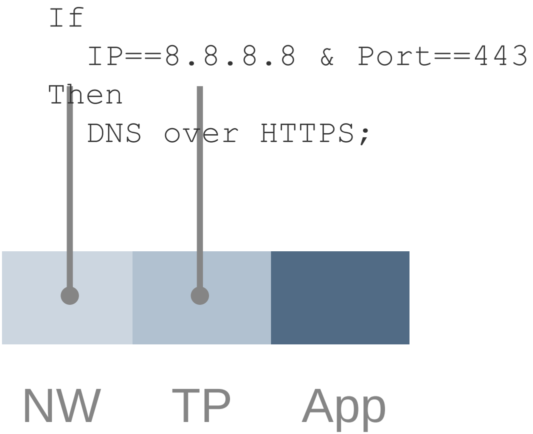
Drawback

Signatures available only for common protocols

Challenging when traffic is encrypted

Usually needs first packet(s) of handshake

Host behavior-based traffic classification



```
If
  IP==8.8.8.8 & Port==443
Then
  DNS over HTTPS;
```

Assumption

Network interaction and host context represent the protocol

Advantage

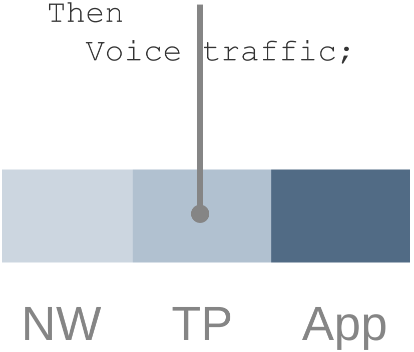
Works well for P2P applications and encrypted traffic

Drawback

Complex profiles needed

Flow feature-based traffic classification (or DPI)

```
If
  <# of packets/s> = 50
Then
  Voice traffic;
```



Assumption

Flow properties (average packet frequency, size etc.) describe application

Advantage

Flexible

Drawback

Needs per flow characteristics

Metrics to assess the performance of classification approaches (2)

Precision

Ratio of True Positives over the sum of True Positives and False Positives or the percentage of flows that are properly attributed to a given application

Recall

Ratio of True Positives over the sum of True Positives and False Negatives or the percentage of flows in an application class that are correctly identified

Metrics to assess the performance of classification approaches (2)

Example

Input
4 packets

Output
2 packets correctly identified,
1 packet incorrectly identified

Precision: 2/3

Recall: 2/4

Precision

Ratio of True Positives over the sum of True Positives and False Positives or the percentage of flows that are properly attributed to a given application

Recall

Ratio of True Positives over the sum of True Positives and False Negatives or the percentage of flows in an application class that are correctly identified

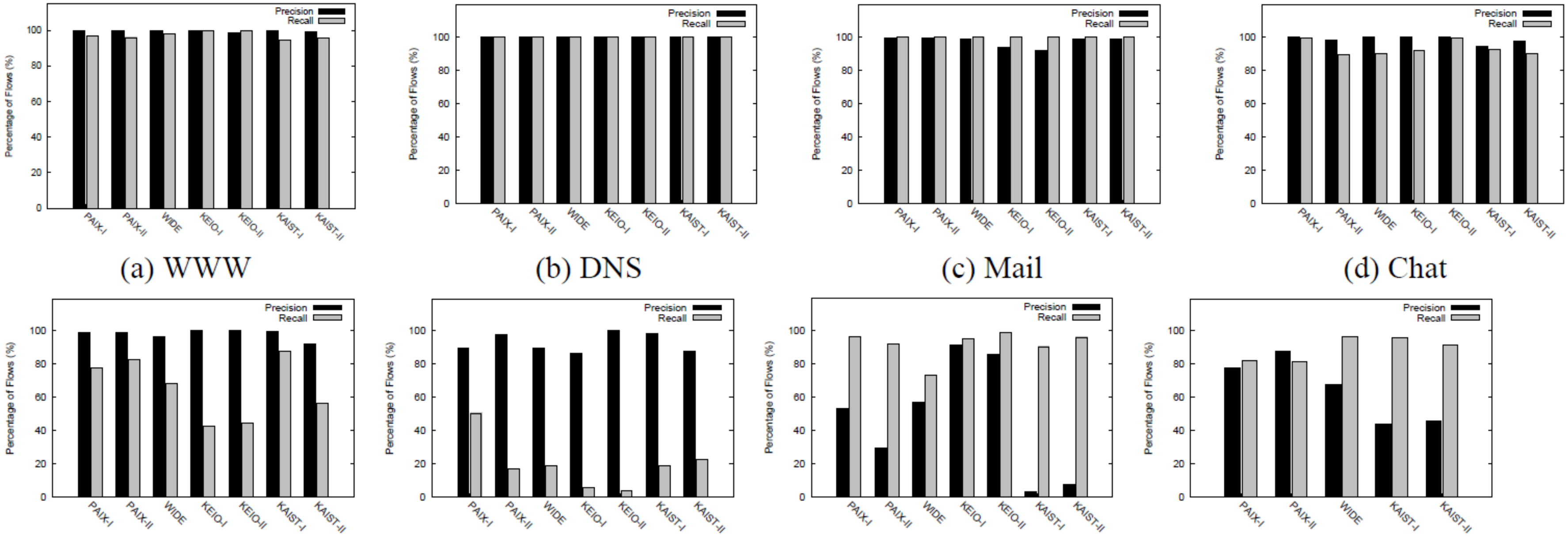
Comparison of different classification schemes

Based on seven (complete) packet traces from different sources from 2004 and 2006.

Details see: Kim et al.: “Internet Traffic Classification Demystified: Myths, Caveats, and the Best Practices,” Proc. of ACM CoNEXT 2008.

We will not focus on flow feature-based machine learning.

Port-based classification



(a) WWW

(b) DNS

(c) Mail

(d) Chat

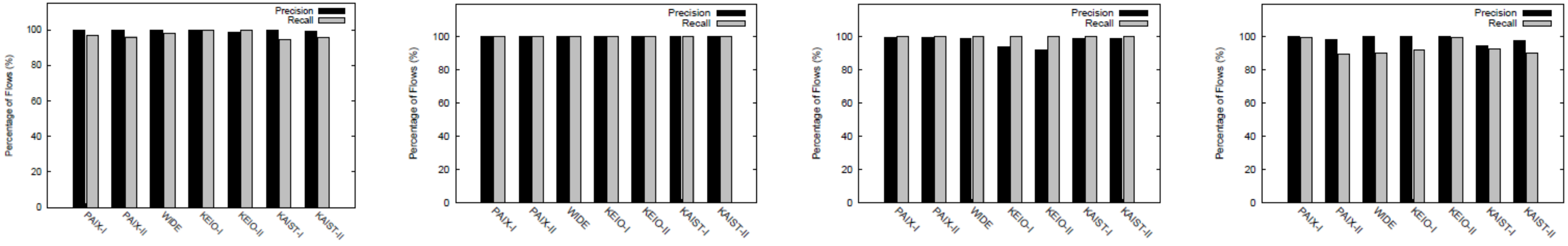
(e) FTP

(f) P2P

(g) Streaming

(h) Game

Port-based classification



(a) WWW (b) DNS (c) Mail (d) Chat

(1) High precision of a port-based classifier implies that its default ports are seldom used by other applications

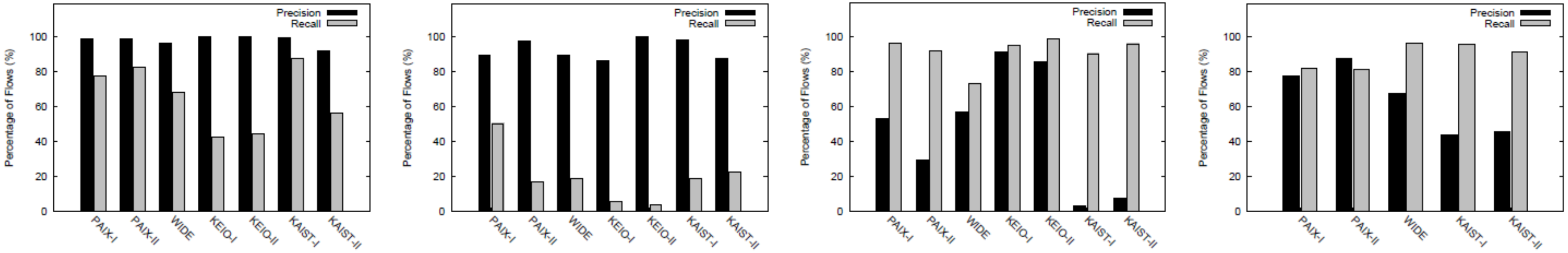
(2) High recall implies that corresponding application mostly uses its default ports.

Port-based classification

Port-based classification fails to yield accurate classification results

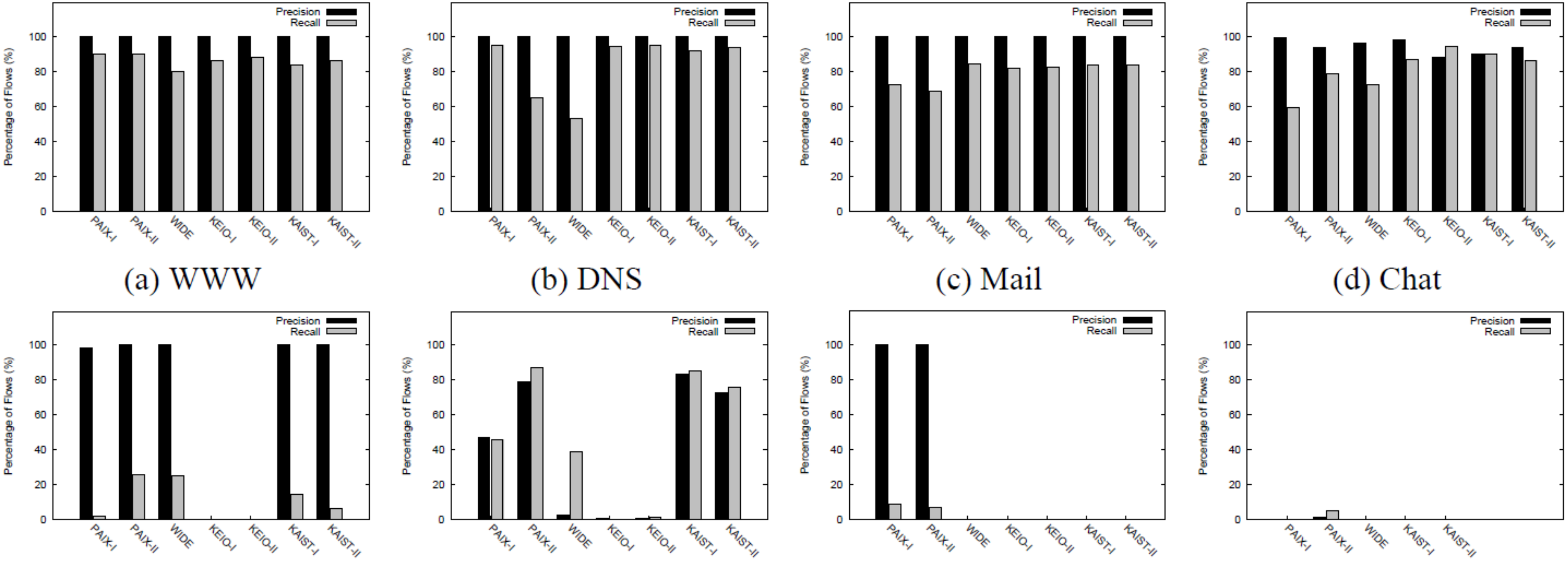
(1) When applications use ephemeral ports

(2) When default ports coincide with port masquerading



(e) FTP (f) P2P (g) Streaming (h) Game

Host behavior-based classification



(a) WWW

(b) DNS

(c) Mail

(d) Chat

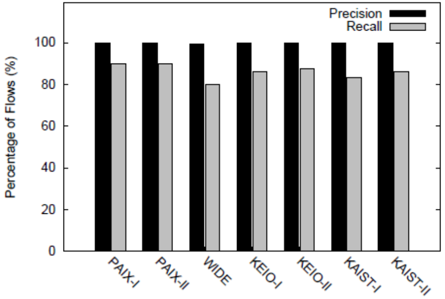
(e) FTP

(f) P2P

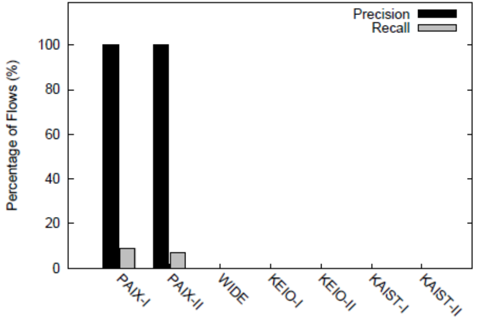
(g) Streaming

(h) Game

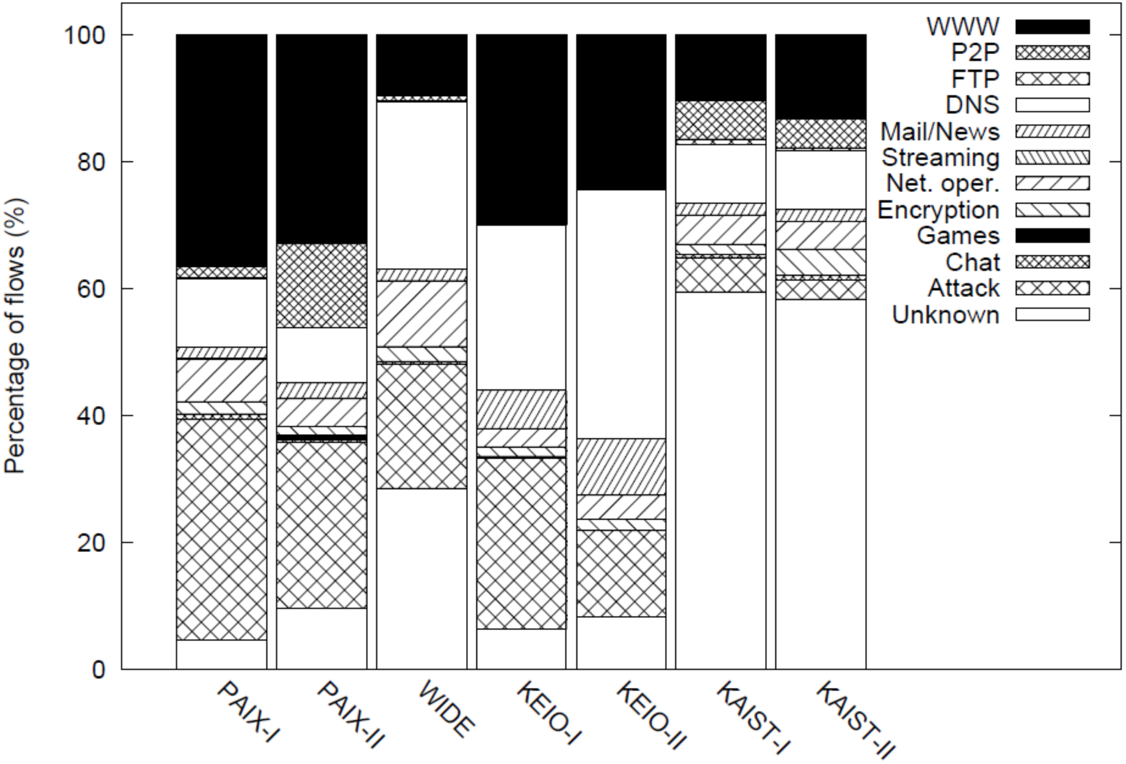
Flow-based Classification



(a) WWW



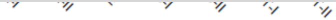
(g) Streaming



Successful classification needs (1) fine tuning and (2) traffic needs to include enough behavioral information about each host.

Best place to use such classification approach: border link of a single-homed edge network

Backbone links are *not* suitable because where (1) only a small portion of behavioral information is collectable of each host and (2) often one direction of traffic flow is missed



(g) Streaming

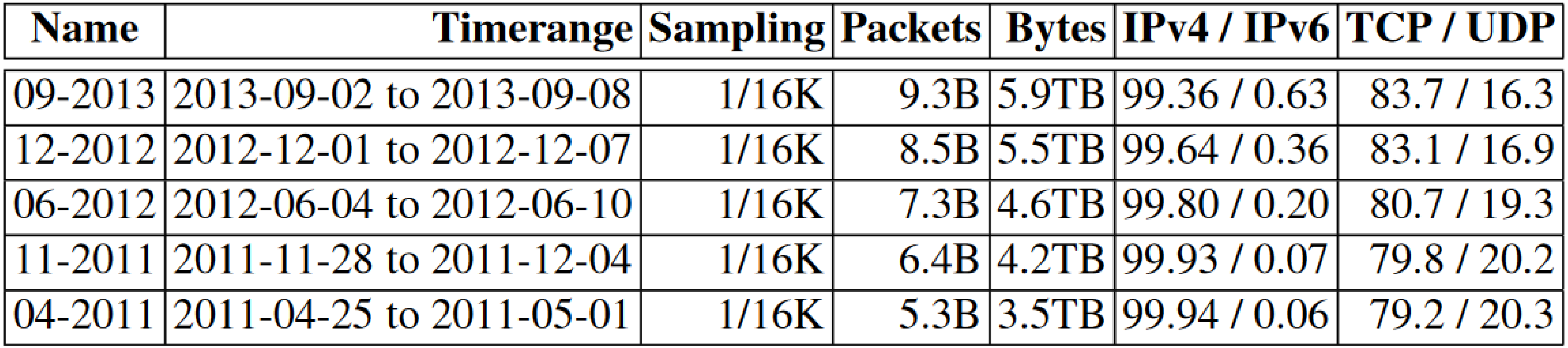
Now, we change the observation perspective and data collection approach.

Observation point: Large European IXP

Data collection: Random packet sampling, data from 2011 – 2013

More details: Richter et al.: “Distilling the Internet’s Application Mix from Packet-Sampled Traffic,” Proc. of PAM 2013.

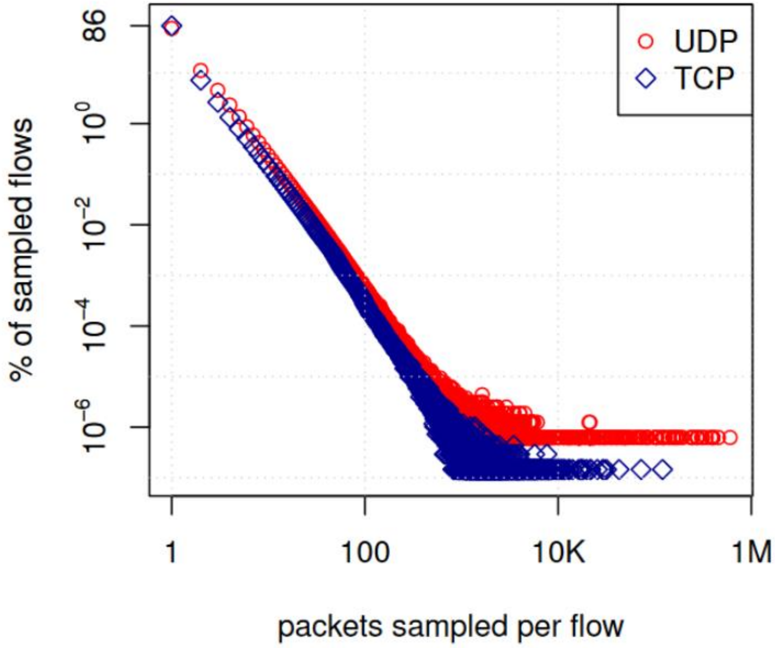
Dataset characteristics



Name	Timerange	Sampling	Packets	Bytes	IPv4 / IPv6	TCP / UDP
09-2013	2013-09-02 to 2013-09-08	1/16K	9.3B	5.9TB	99.36 / 0.63	83.7 / 16.3
12-2012	2012-12-01 to 2012-12-07	1/16K	8.5B	5.5TB	99.64 / 0.36	83.1 / 16.9
06-2012	2012-06-04 to 2012-06-10	1/16K	7.3B	4.6TB	99.80 / 0.20	80.7 / 19.3
11-2011	2011-11-28 to 2011-12-04	1/16K	6.4B	4.2TB	99.93 / 0.07	79.8 / 20.2
04-2011	2011-04-25 to 2011-05-01	1/16K	5.3B	3.5TB	99.94 / 0.06	79.2 / 20.3

Dataset characteristics

86% of sampled TCP flows: only one packet samples



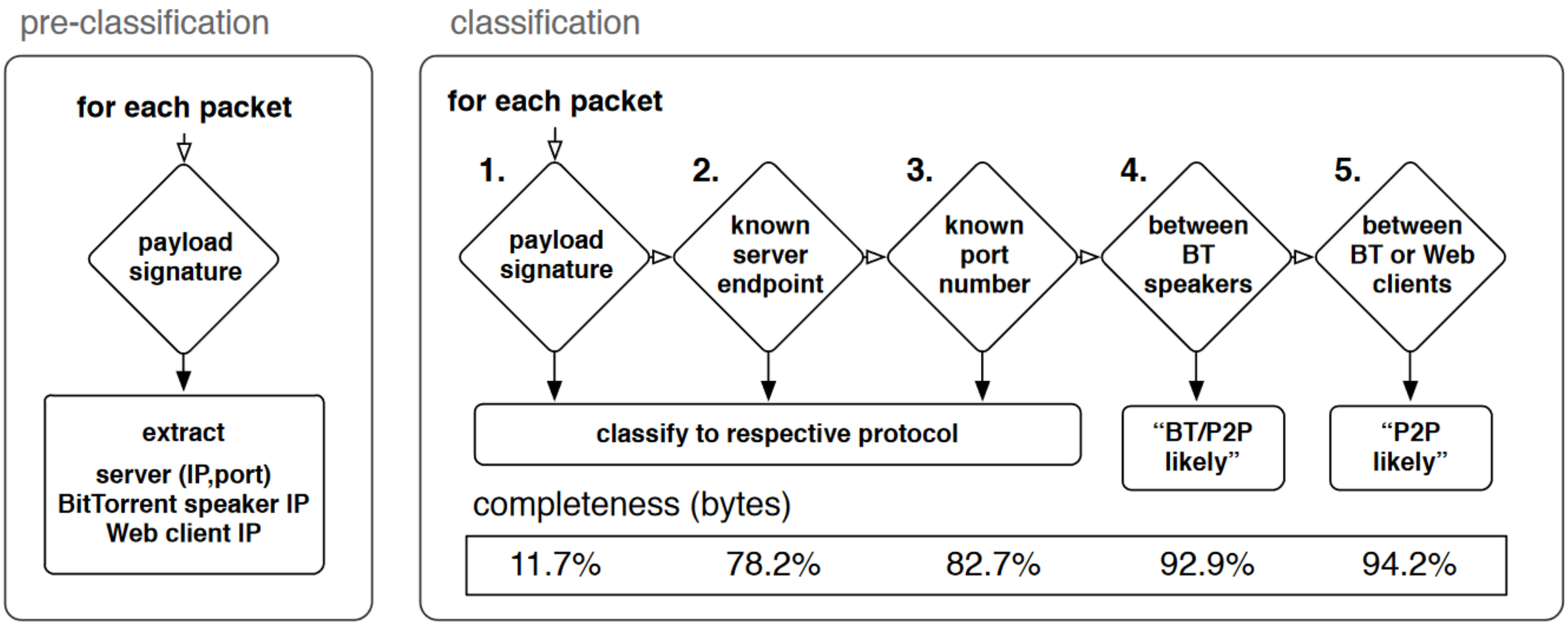
(a) Samples per flow (1200s timeout).

Sampling limits

Only limited amount of payload was captured (details depend on IP and TCP options)

Flow feature-based approaches not applicable

Classification pipeline



pre-classification

for each packet

payload signature

extract

server (IP,port)
BitTorrent speaker IP
Web client IP

classification

for each packet

1. payload signature

2. known server endpoint

3. known port number

4. between BT speakers

5. between BT or Web clients

classify to respective protocol

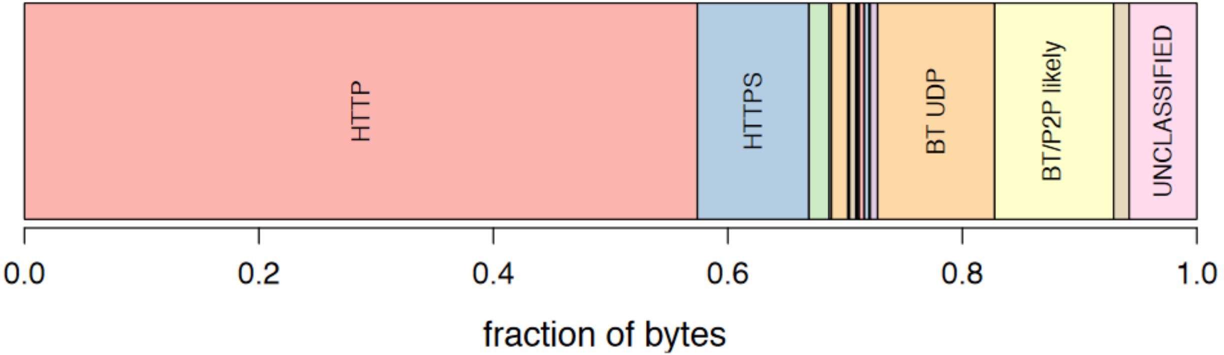
"BT/P2P likely"

"P2P likely"

completeness (bytes)

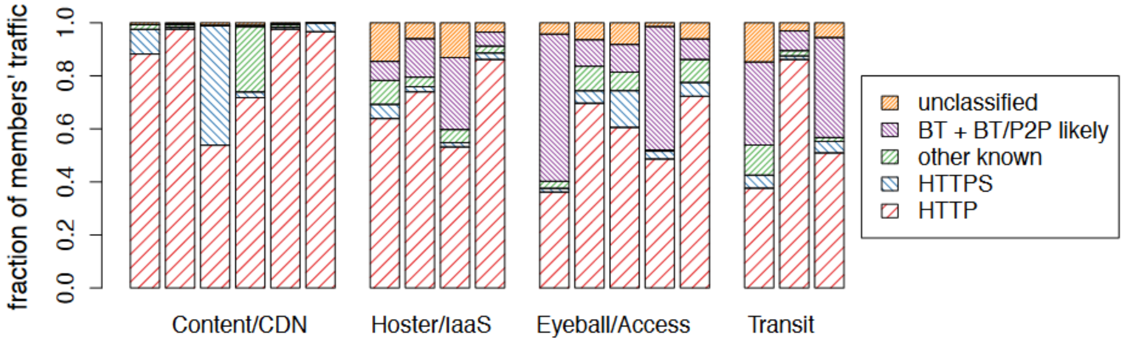
1.	2.	3.	4.	5.
11.7%	78.2%	82.7%	92.9%	94.2%

Application mix: Aggregate



- HTTP(S) dominates ~67%
- other applications (e.g., RTMP, mail, news) ~6%
- BitTorrent/BT/P2P likely ~22%
- unclassified ~5%

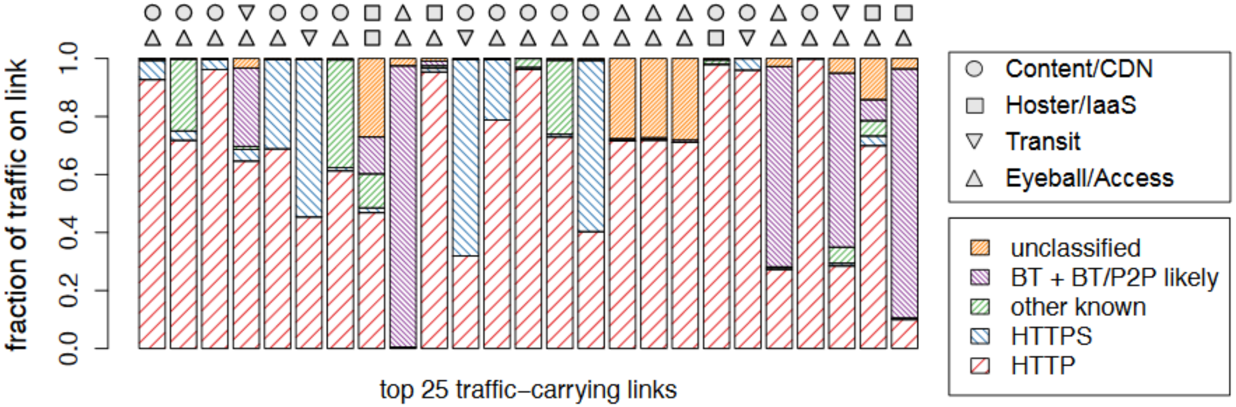
Application mix: Per network type



- Content/CDN almost 100% HTTP
- HTTPS increase driven by only a few networks
- P2P not only between Eyeballs! Hoster/IaaS too!

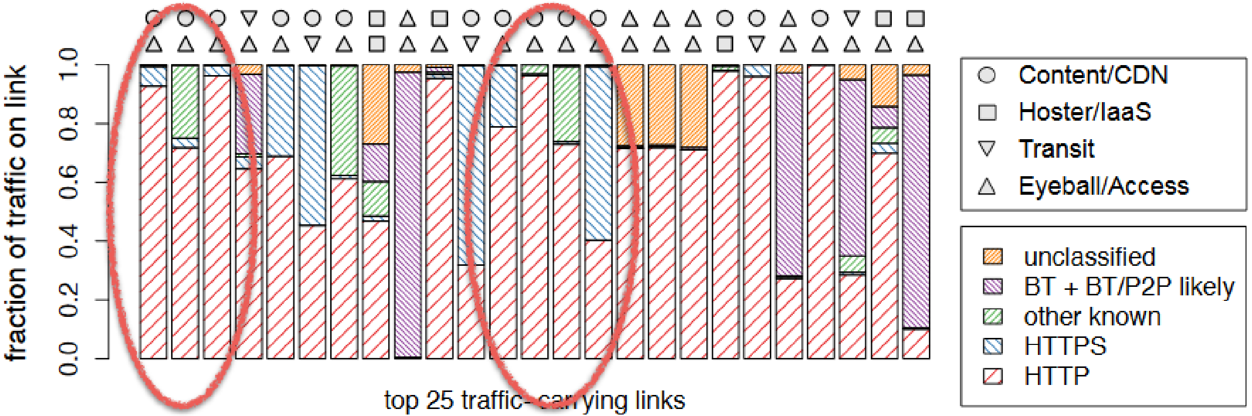
Dissecting per network shows a different appmix!

Application mix: Per link



- Aggregate mix by no means representative of single link
- Many links just have one dominant protocol
- The business type of the ASes gives hints on app mix

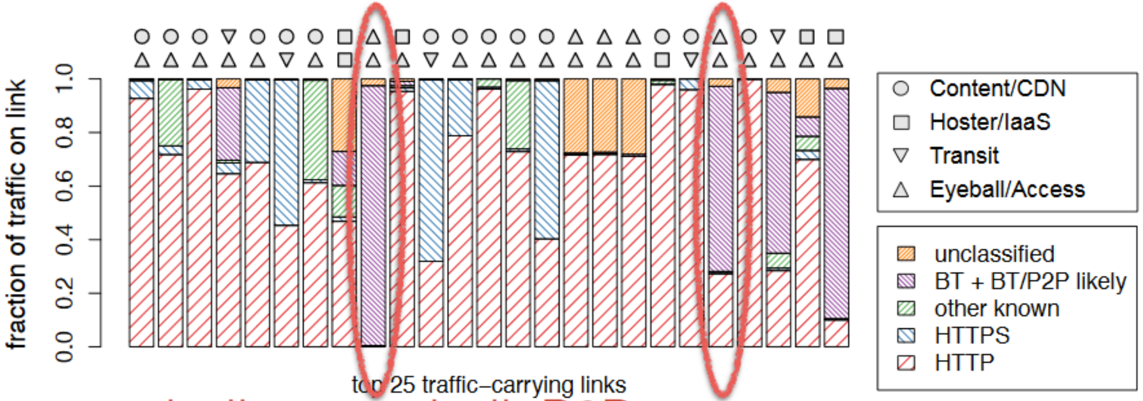
Application mix: Per link (content – eyeball)



content <> eyeball: HTTP

- Aggregate mix by no means representative of single link
- Many links just have one dominant protocol
- The business type of the ASes gives hints on app mix

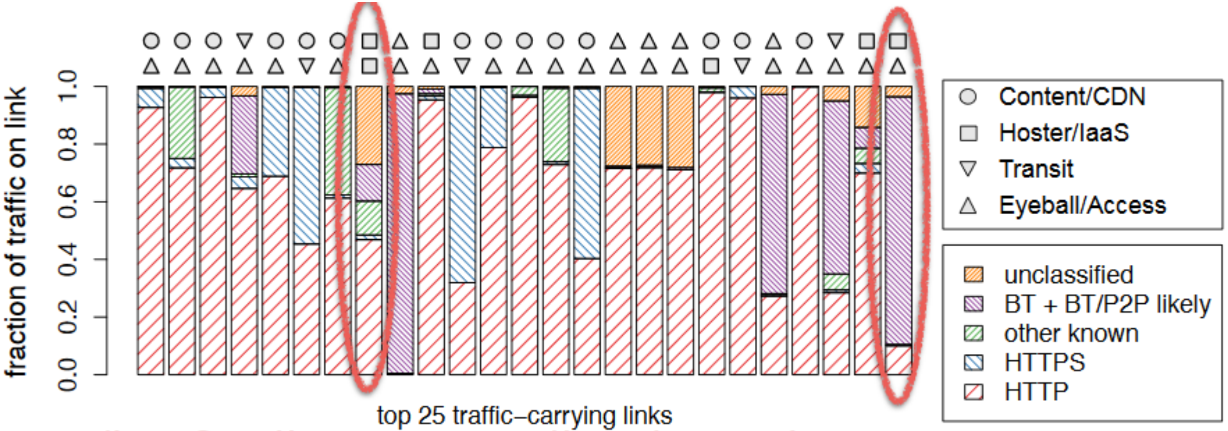
Application mix: Per link (eyeball – eyeball)



eyeball <> eyeball: P2P

- Aggregate mix by no means representative of single link
- Many links just have one dominant protocol
- The business type of the ASes gives hints on app mix

Application mix: Per link (hoster/IaaS)



hoster/IaaS: diverse application mix

- Aggregate mix by no means representative of single link
- Many links just have one dominant protocol
- The business type of the ASes gives hints on app mix

Insights

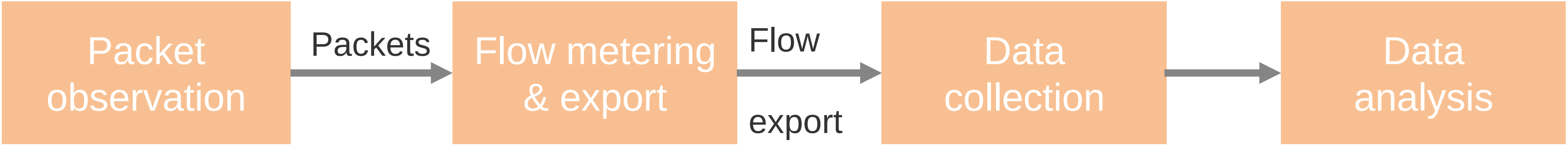
A stateful approach can overcome limitations of random packet sampling

Dissecting network types reveals different application mix

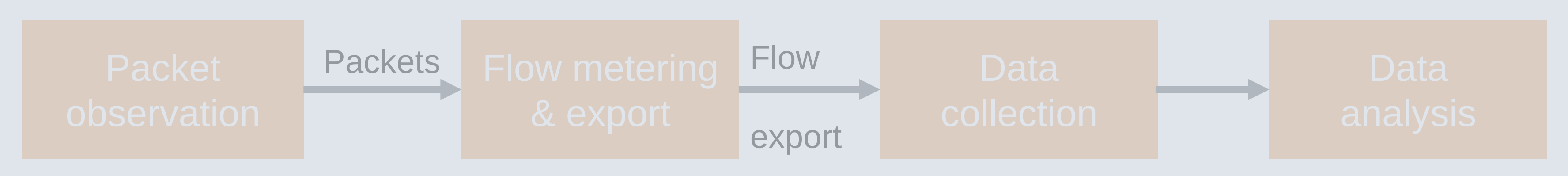
Measuring Packets in Context

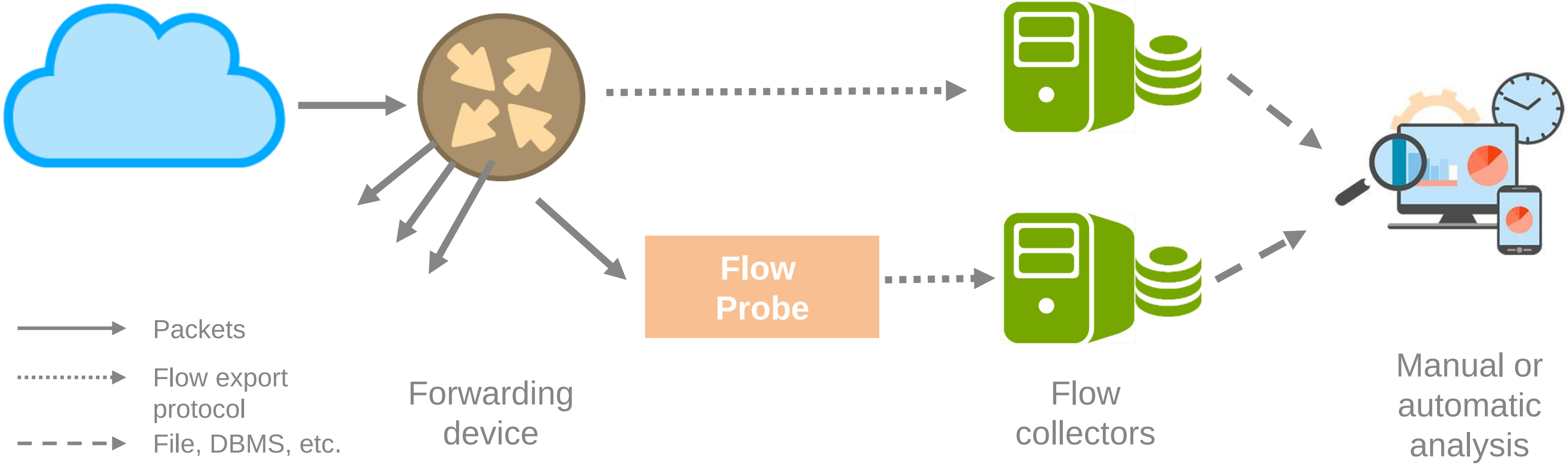
MONITORING FLOWS

Typical flow monitoring setups



Typical flow monitoring setups





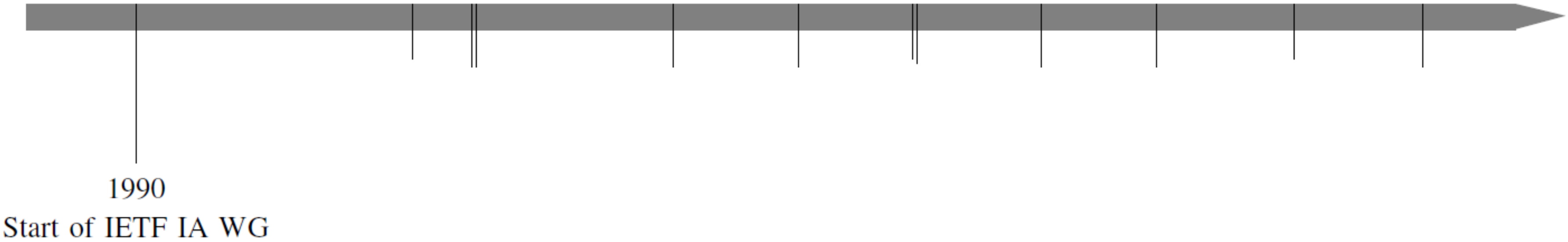
Requirements

Vendor independent

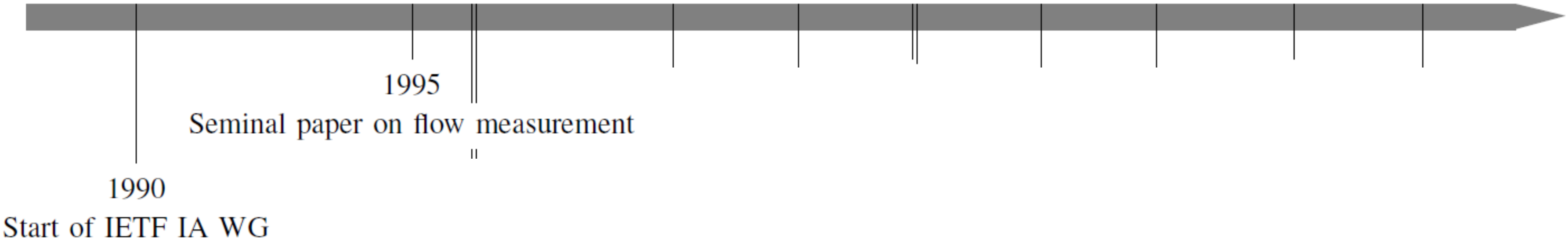
Support different deployments

Handle large data

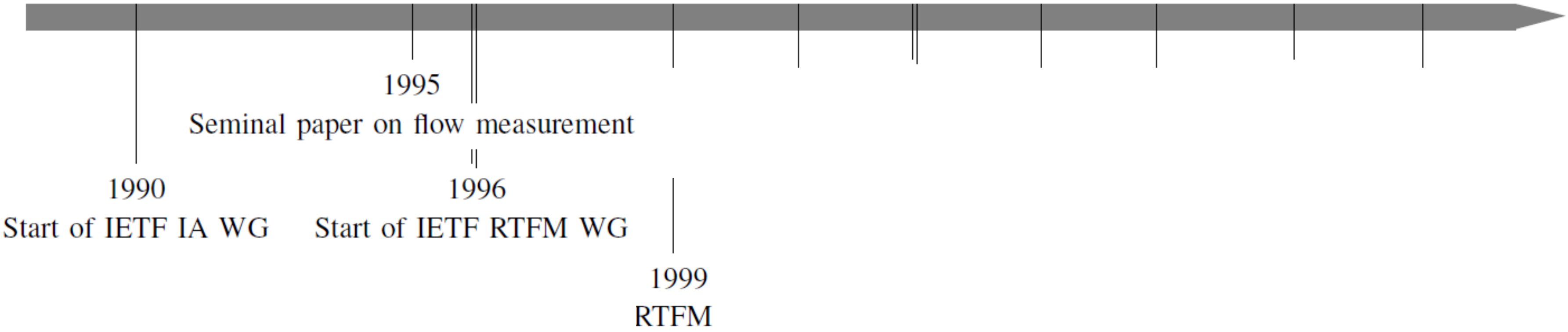
Evolution of flow export technologies and protocols



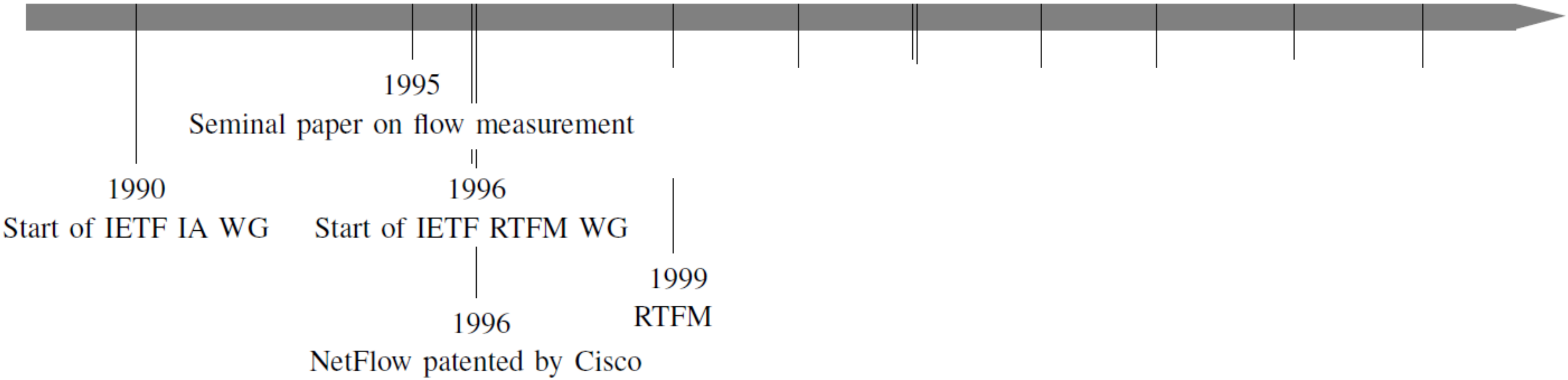
Evolution of flow export technologies and protocols



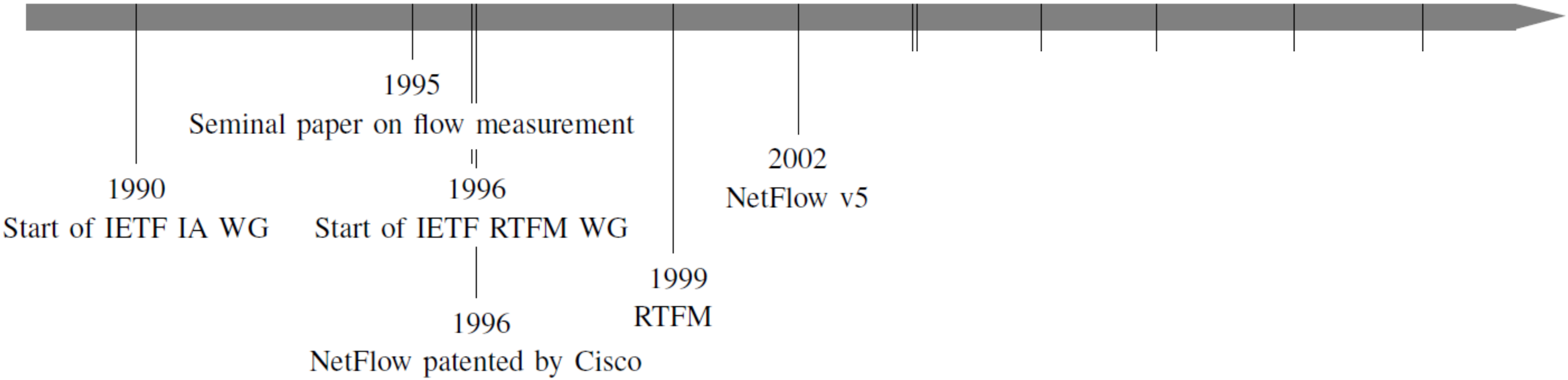
Evolution of flow export technologies and protocols



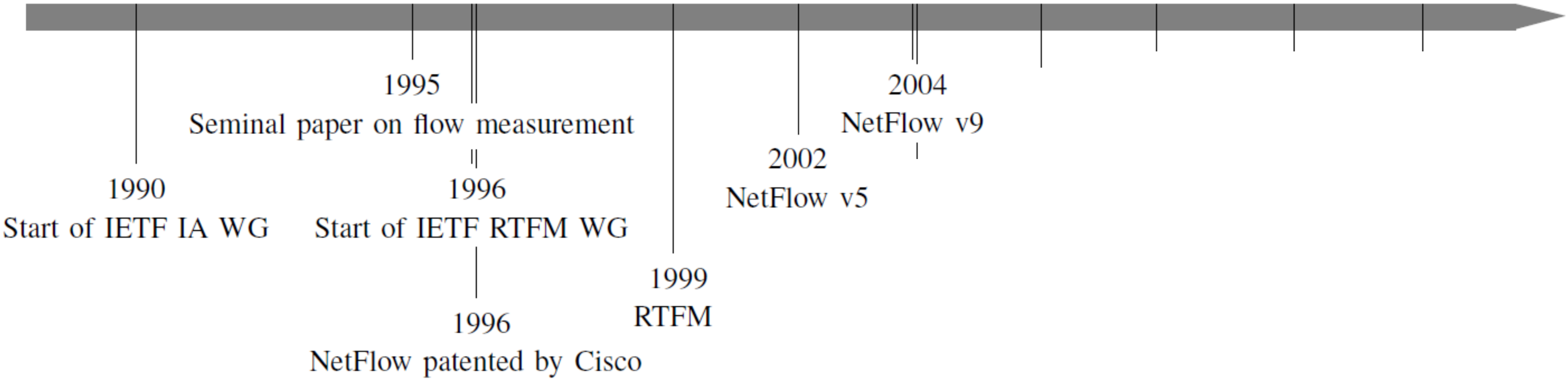
Evolution of flow export technologies and protocols



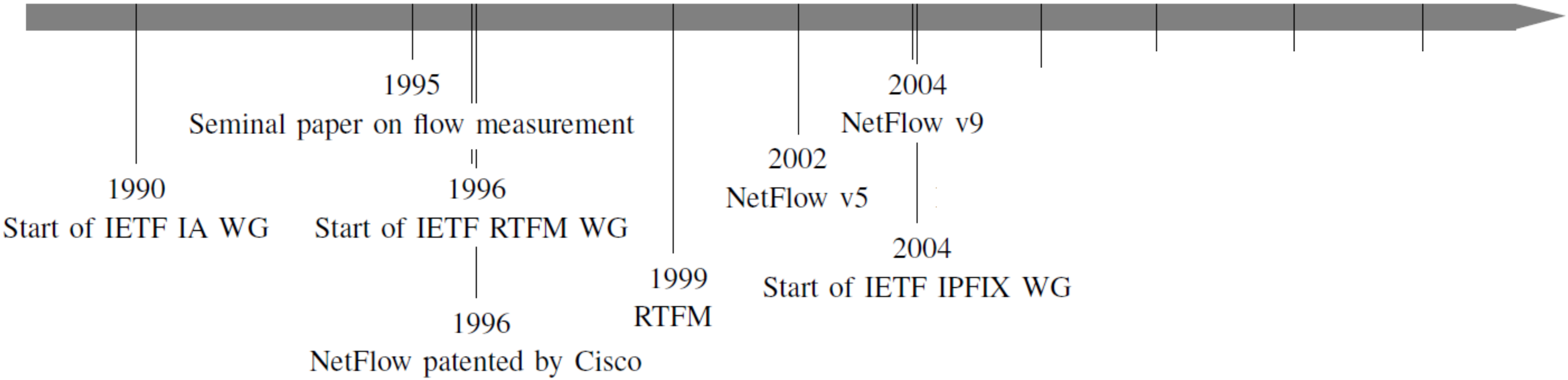
Evolution of flow export technologies and protocols



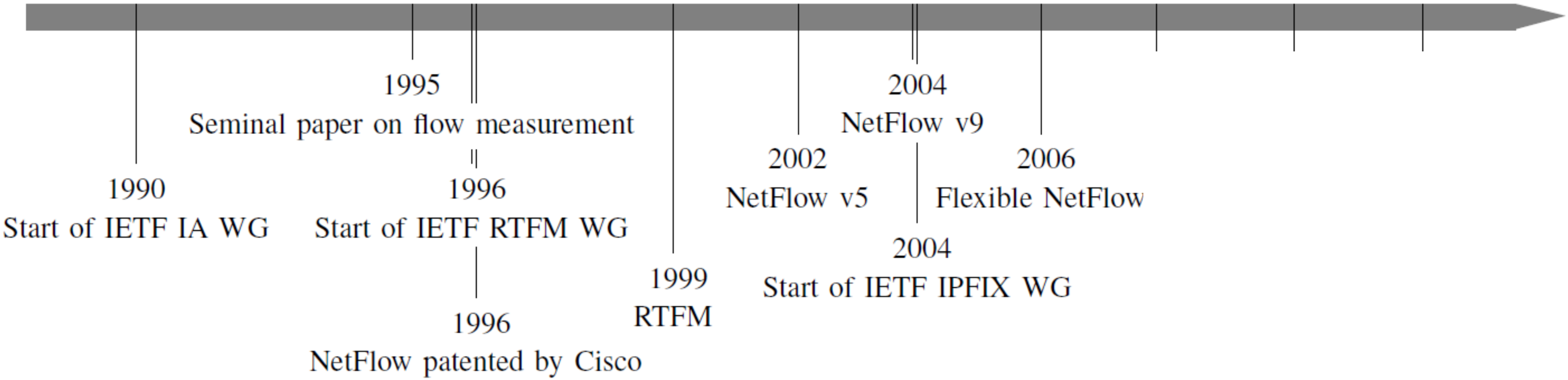
Evolution of flow export technologies and protocols



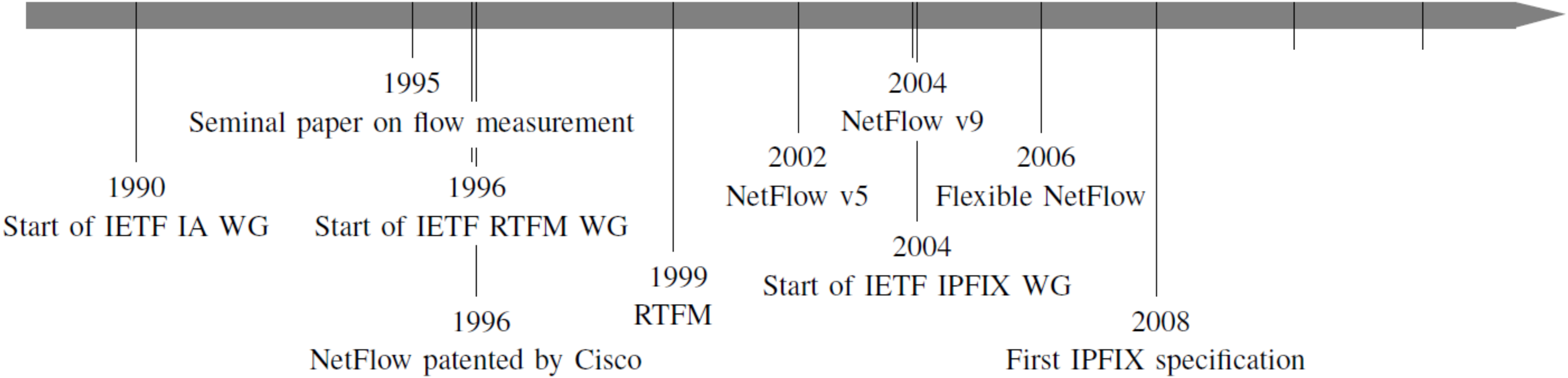
Evolution of flow export technologies and protocols



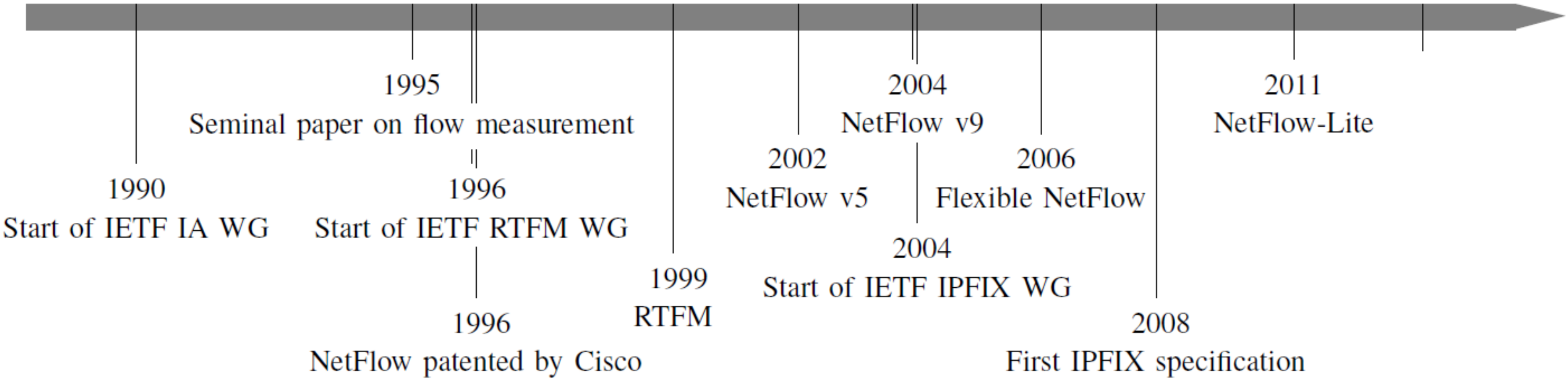
Evolution of flow export technologies and protocols



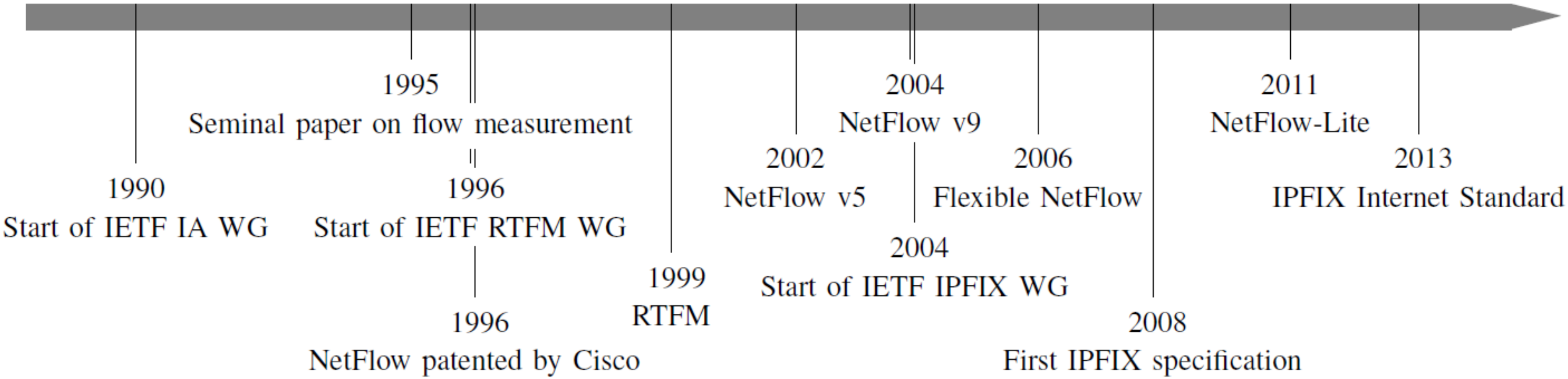
Evolution of flow export technologies and protocols



Evolution of flow export technologies and protocols



Evolution of flow export technologies and protocols



Related but not the same: sFlow

Industry standard

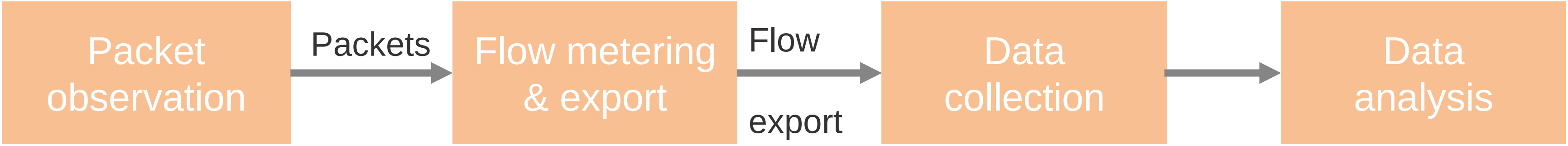
Integrated into many packet forwarding devices

Samples packets and interface counters

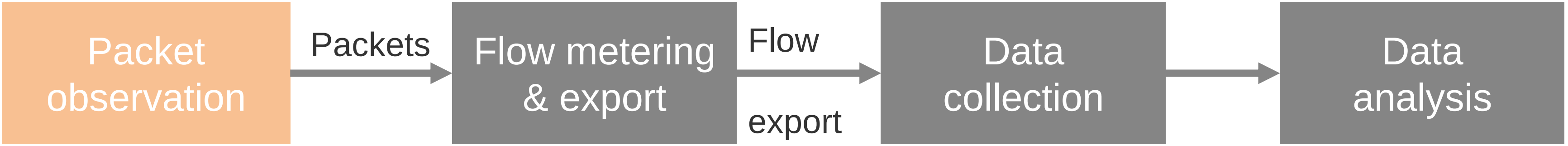
Architectural similar to NetFlow and IPFIX but it is packet-oriented

Closer related to packet sampling techniques

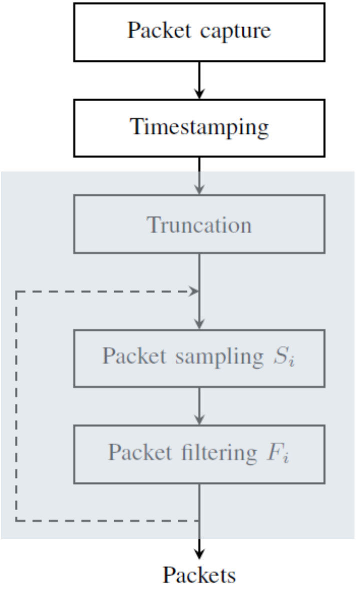
Typical flow monitoring setups



Typical flow monitoring setups



Packet observation

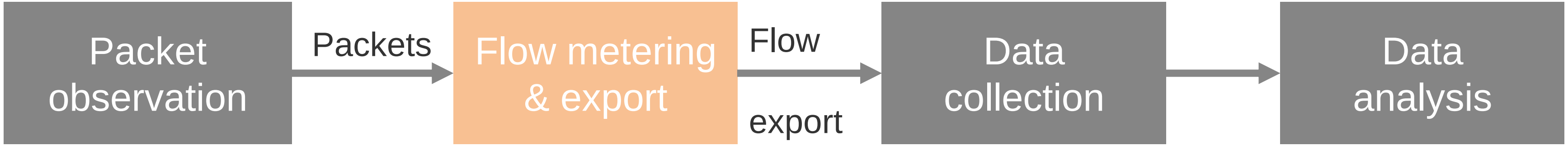


Truncation selects only those bytes that fit into a preconfigured snapshot length

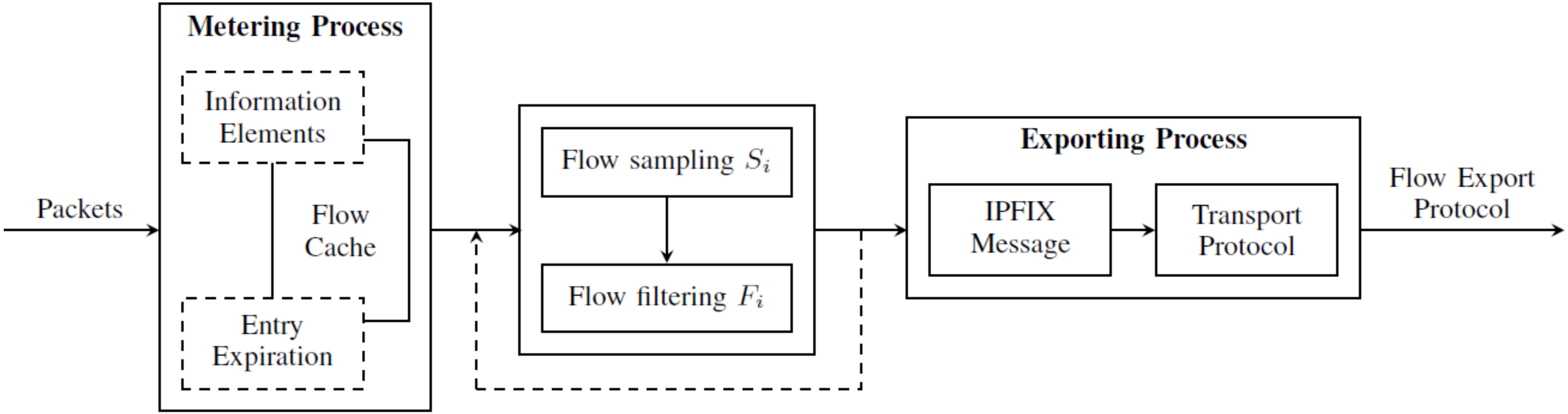
Traffic capture can be implemented in in-line mode or mirroring mode

Software tools, e.g., libpcap
Network stacks are made for general-purpose networking, leading to suboptimal performance; improvements available (e.g., PF_RING)

Typical flow monitoring setups



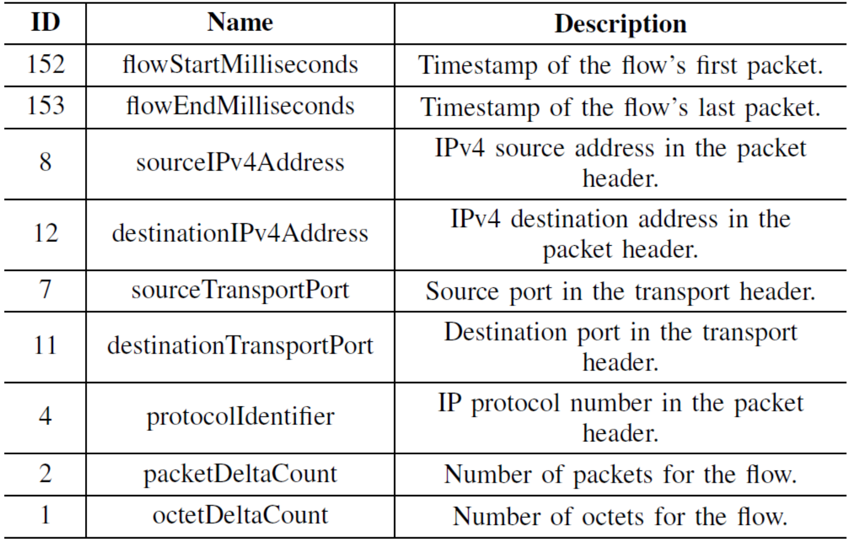
Flow metering and export



Current Standard

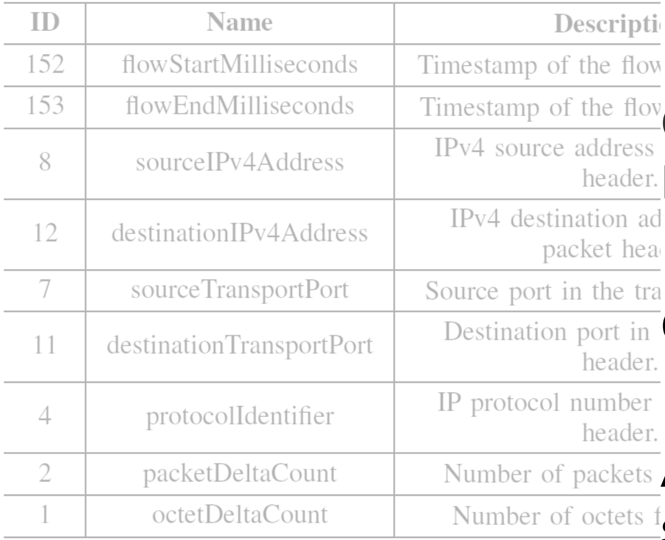
IP FLOW INFORMATION EXPORT (IPFIX)

Information Elements (IE) describe the exported data in IPFIX



ID	Name	Description
152	flowStartMilliseconds	Timestamp of the flow's first packet.
153	flowEndMilliseconds	Timestamp of the flow's last packet.
8	sourceIPv4Address	IPv4 source address in the packet header.
12	destinationIPv4Address	IPv4 destination address in the packet header.
7	sourceTransportPort	Source port in the transport header.
11	destinationTransportPort	Destination port in the transport header.
4	protocolIdentifier	IP protocol number in the packet header.
2	packetDeltaCount	Number of packets for the flow.
1	octetDeltaCount	Number of octets for the flow.

Information Elements (IE) describe the exported data in IPFIX



ID	Name	Descripti
152	flowStartMilliseconds	Timestamp of the flow
153	flowEndMilliseconds	Timestamp of the flov
8	sourceIPv4Address	IPv4 source address header.
12	destinationIPv4Address	IPv4 destination ad packet hea
7	sourceTransportPort	Source port in the tra
11	destinationTransportPort	Destination port in header.
4	protocolIdentifier	IP protocol number header.
2	packetDeltaCount	Number of packets
1	octetDeltaCount	Number of octets f

Maintained by IANA

Enterprise-specific IEs possible

Can be defined for any layer

But common focus on network and transport layer

Configuration of metering process not standardized

Allows for templates, variable-length encoding, and structured data

Flow Caches store information about active network traffic flows

Entries are composed of IEs

Flow key defines whether a packet defines a new flow or not

Flow caches may differ in cache layout

Cope with IE flexibility

… type

e.g., immediate caches, permanent cache

… and size

Cache entries usually require expiration timers

Cache entries are maintained in the flow cache until the corresponding flows are considered terminated

Active timeout, flow has been active for a specified period of time (120s – 30 min); cache entries are not removed but counters are reset

Idle timeout, no packets belonging to a flow have been observed (15s – 5 min)

Resource constraints, special heuristics

Natural expiration, TCP packet with a FIN or RST flag; depends on the exporter implementation

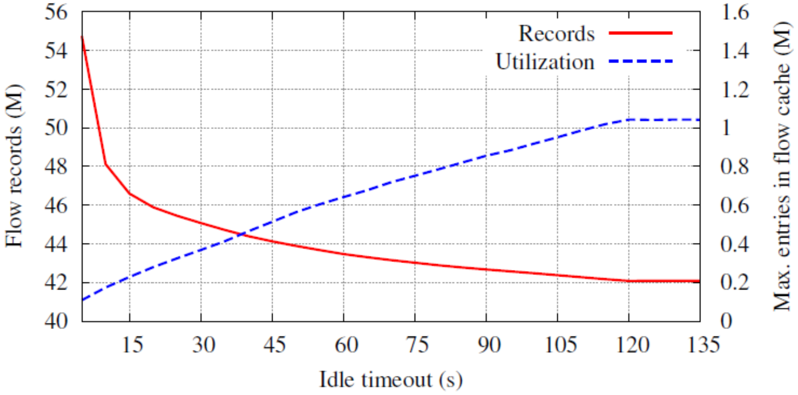
Idle and active timeout have impact on total # of recorded and exported flows

Longer timeout values result in higher aggregation of packets into flow records

Pros: Reduces load on flow collector

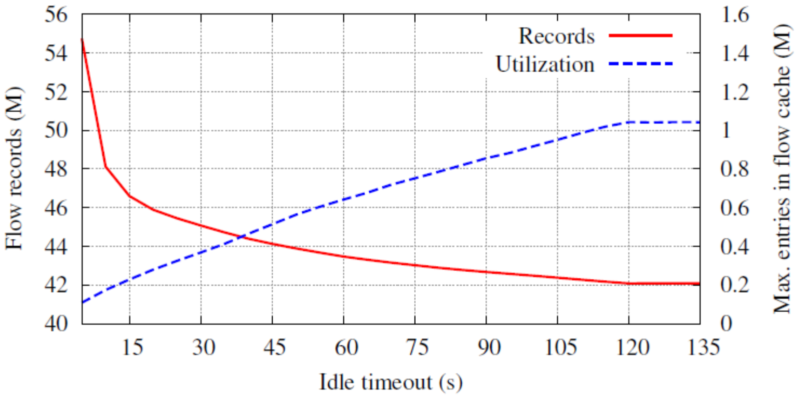
Cons: takes longer before a flow becomes visible in the data analysis

Experimental evaluation

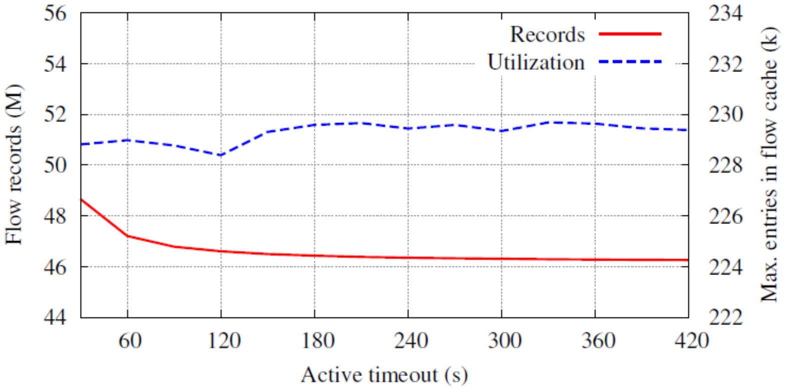


(a) Varying idle timeout values, active timeout = 120 seconds

Experimental evaluation



(a) Varying idle timeout values, active timeout = 120 seconds



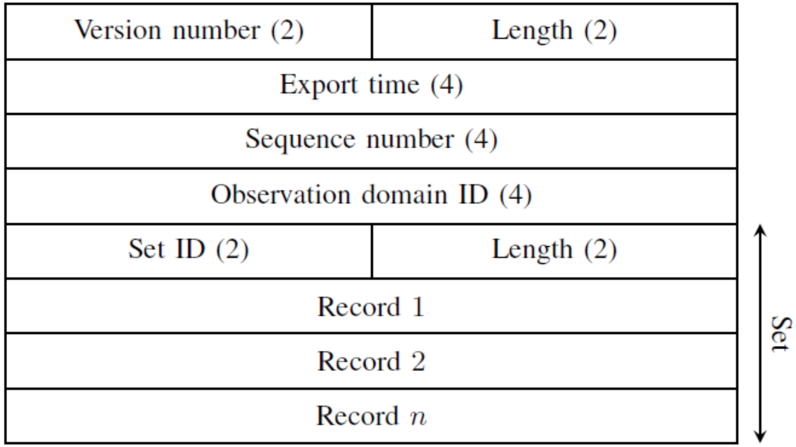
(b) Varying active timeout values, idle timeout = 15 seconds

IPFIX messages [RFC 7011]

Template Set describes the layout of Data Records

Data Set carries exported Data Records (i.e., flow records)

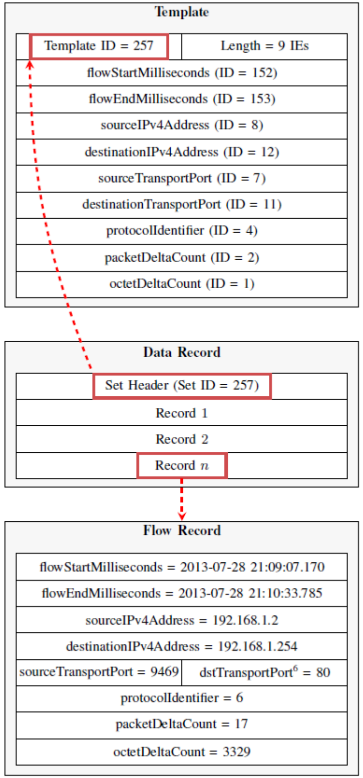
Options Template Set includes meta-data



Version number (2)	Length (2)
Export time (4)	
Sequence number (4)	
Observation domain ID (4)	
Set ID (2)	Length (2)
Record 1	
Record 2	
Record n	

(simplified)

IPFIX messages [RFC 7011]



Template

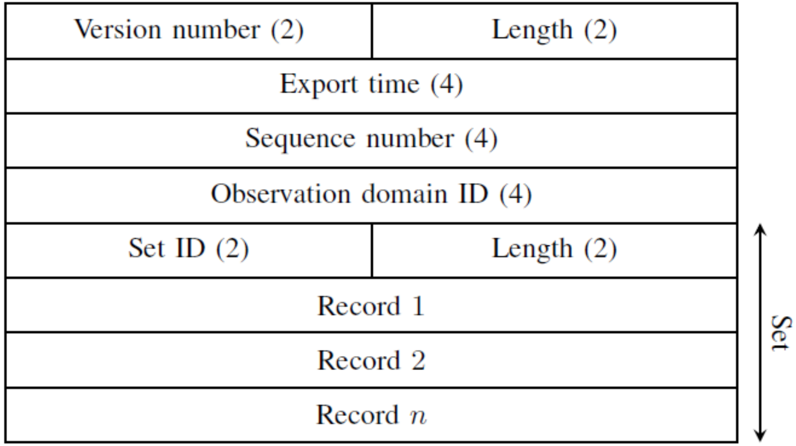
Template ID = 257	Length = 9 IEs
flowStartMilliseconds (ID = 152)	
flowEndMilliseconds (ID = 153)	
sourceIPv4Address (ID = 8)	
destinationIPv4Address (ID = 12)	
sourceTransportPort (ID = 7)	
destinationTransportPort (ID = 11)	
protocolIdentifier (ID = 4)	
packetDeltaCount (ID = 2)	
octetDeltaCount (ID = 1)	

Data Record

Set Header (Set ID = 257)
Record 1
Record 2
Record n

Flow Record

flowStartMilliseconds = 2013-07-28 21:09:07.170	
flowEndMilliseconds = 2013-07-28 21:10:33.785	
sourceIPv4Address = 192.168.1.2	
destinationIPv4Address = 192.168.1.254	
sourceTransportPort = 9469	dstTransportPort[6] = 80
protocolIdentifier = 6	
packetDeltaCount = 17	
octetDeltaCount = 3329	



Version number (2)	Length (2)
Export time (4)	
Sequence number (4)	
Observation domain ID (4)	
Set ID (2)	Length (2)
Record 1	
Record 2	
Record n	

(simplified)

Which transport protocol to export flows?

Problems:

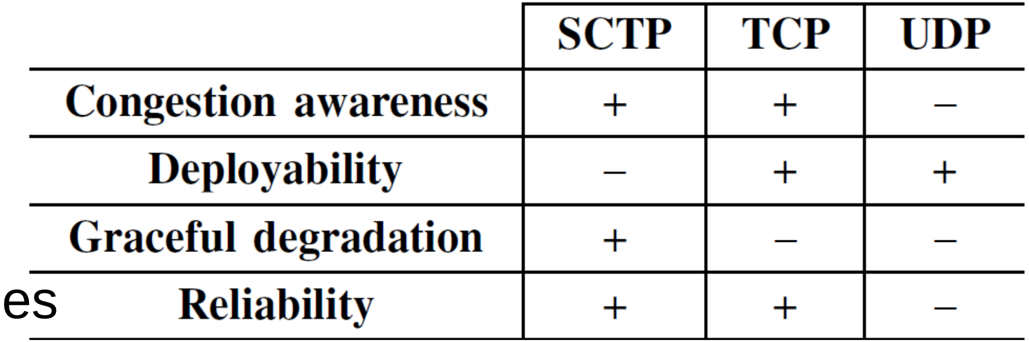
TCP - head-of-line blocking

UDP – unreliable, lack of congestion control

SCTP – missing deployment

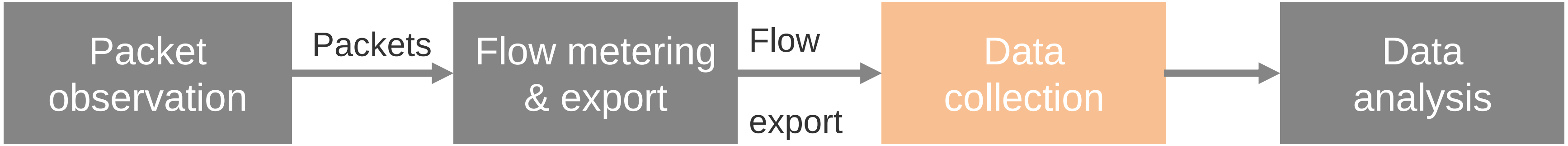
Potentials of SCTP:

- message oriented w/ boundaries
- multiple streams per connectio

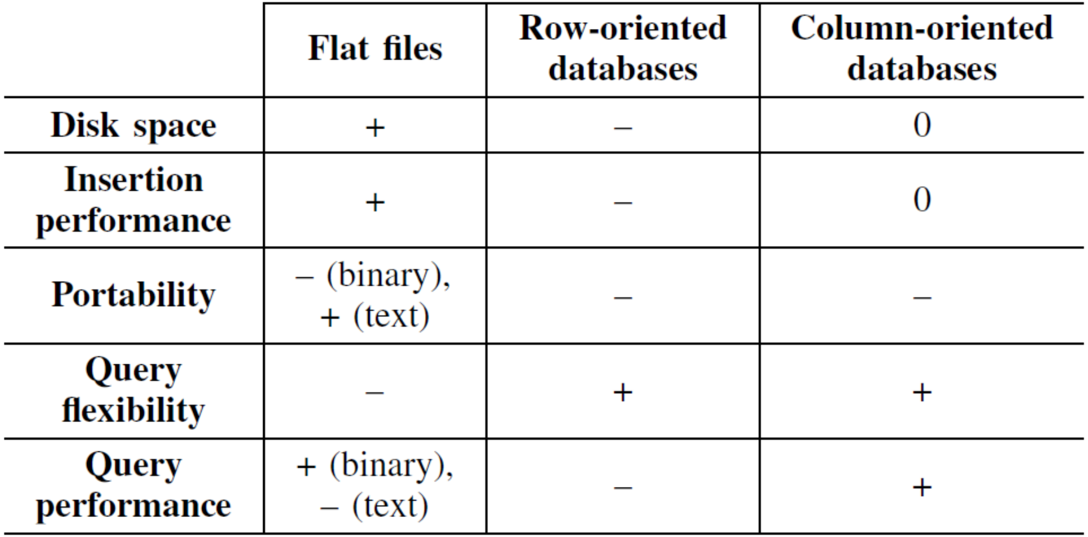


	SCTP	TCP	UDP
Congestion awareness	+	+	–
Deployability	–	+	+
Graceful degradation	+	–	–
Reliability	+	+	–

Typical flow monitoring setups



Storage formats



	Flat files	Row-oriented databases	Column-oriented databases
Disk space	+	–	0
Insertion performance	+	–	0
Portability	– (binary), + (text)	–	–
Query flexibility	–	+	+
Query performance	+ (binary), – (text)	–	+

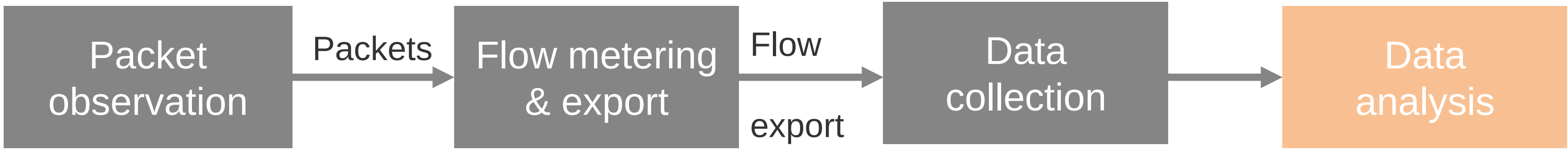
Data anonymization

Even though flow data include no or very limited payload, individuals can be identified and tracked

Anonymization technique depends on the use case

Complete random, prefix-preserving, prefix anonymized

Typical flow monitoring setups



Example: Threat detection SSH

Frequently-used target of dictionary attacks

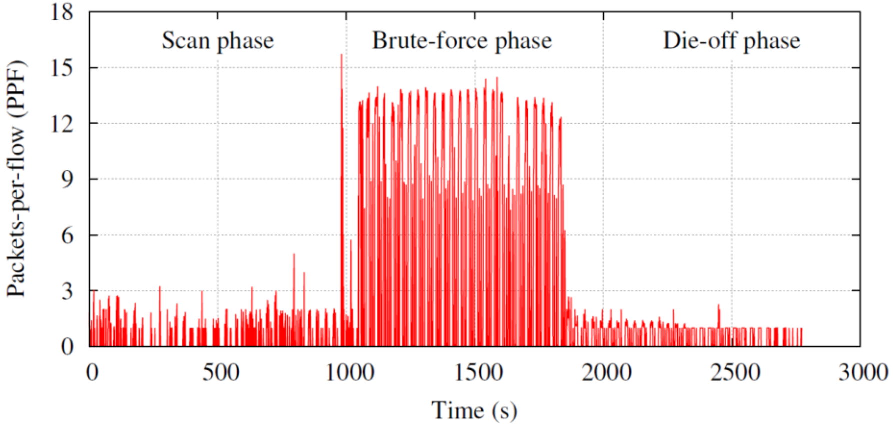
How would you detect those attacks, albeit SSH is encrypted?

Example: Threat detection SSH

Many credentials are tested subsequently

SSH daemons close connections after a fixed number of login attempts
Consequently: Many TCP connections with similar size in terms of packets

Example: Threat detection SSH



Many credentials are tested subsequently

SSH daemons close connections after a fixed number of login attempts

Consequently: Many TCP connections with similar size in terms of packets

Example: Performance monitoring

Two approaches:

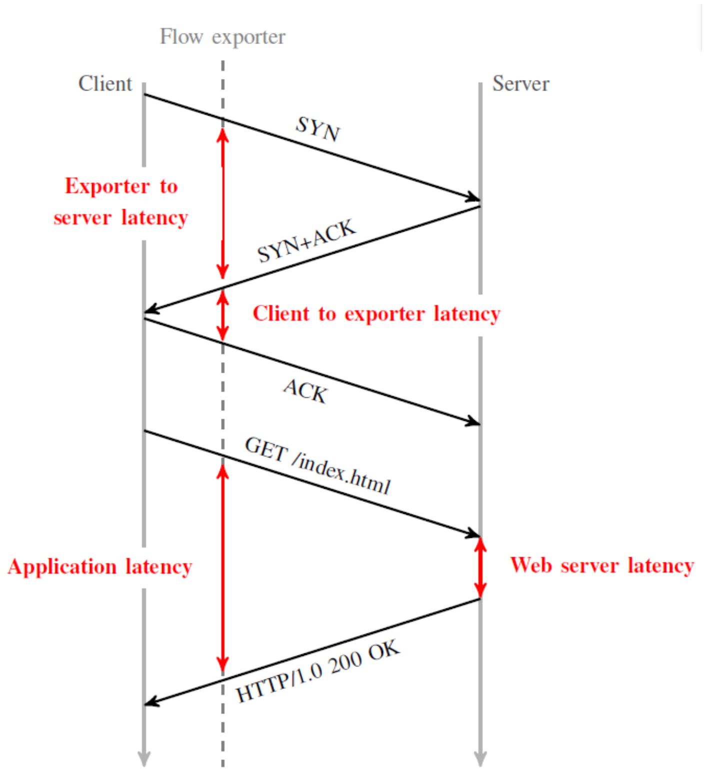
Post processing of information elements

- no customization is needed at flow exporter or collector but limited insights for high-level performance metrics

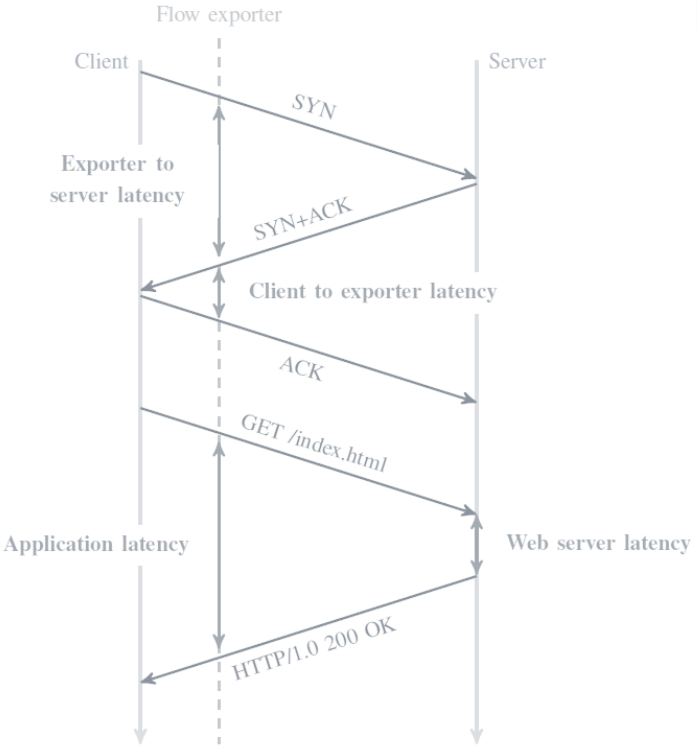
Inline processing of measurement data

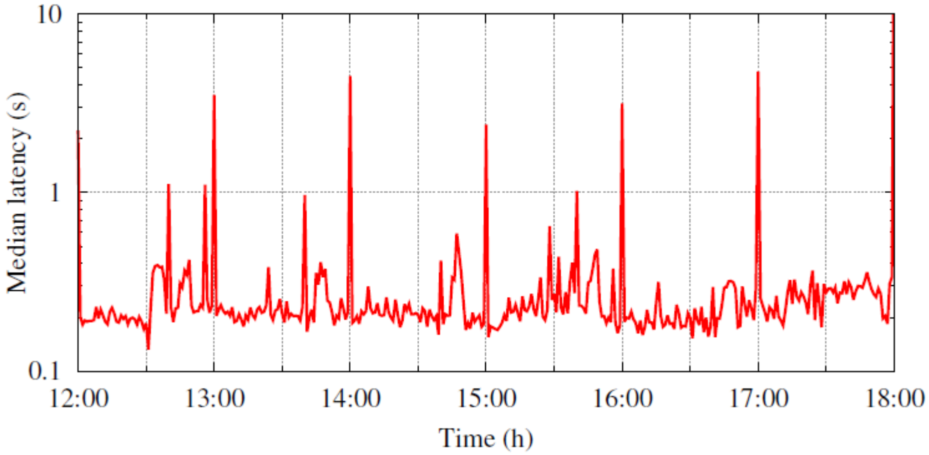
- extension or modification of flow exporters is required

Example: Performance monitoring

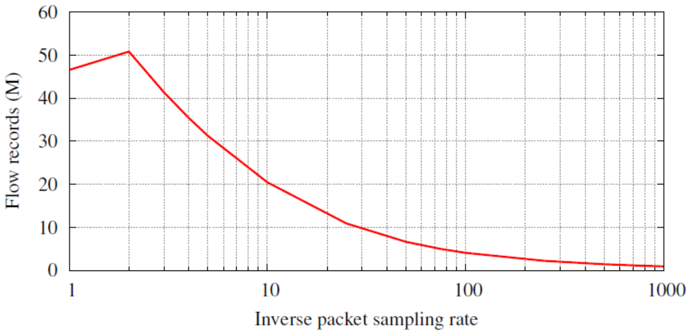


Example: Performance monitoring

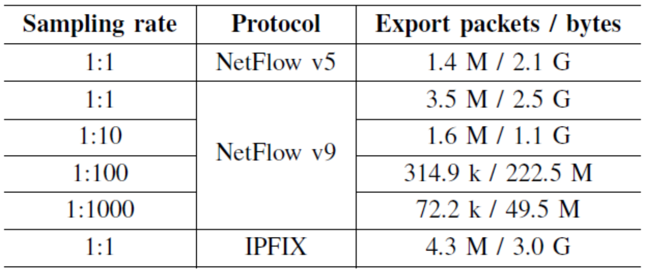




Common pitfalls



Export Volumes for the UT Dataset (2.1 TB)



Sampling rate	Protocol	Export packets / bytes
1:1	NetFlow v5	1.4 M / 2.1 G
1:1	NetFlow v9	3.5 M / 2.5 G
1:10		1.6 M / 1.1 G
1:100		314.9 k / 222.5 M
1:1000		72.2 k / 49.5 M
1:1	IPFIX	4.3 M / 3.0 G

Flow Exporter overload

Flow cache may exceed limits (check loss statistics, adapt timeouts, apply packet sampling)

Transport overhead

Flow Collector overload

Flow data artifacts (timing, data loss, inaccuracies)

Literature

R. Hofstede *et al*., "Flow Monitoring Explained: From Packet Capture to Data Analysis With NetFlow and IPFIX," in *IEEE Communications Surveys & Tutorials*, vol. 16, no. 4, pp. 2037-2064, 2014. https://dx.doi.org/10.1109/COMST.2014.2321898

Flow Monitoring Explained: From Packet Capture to Data Analysis with NetFlow and IPFIX

Rick Hofstede, Pavel Čeleda, Brian Trammell, Idilio Drago, Ramin Sadre, Anna Sperotto and Aiko Pras

***Abstract*—Flow monitoring has become a prevalent method for monitoring traffic in high-speed networks. By focusing on the analysis of flows, rather than individual packets, it is often said to be more scalable than traditional packet-based traffic analysis. Flow monitoring embraces the complete chain of packet observation, flow export using protocols such as NetFlow and IPFIX, data collection, and data analysis. In contrast to what is often assumed, all stages of flow monitoring are closely intertwined. Each of these stages therefore has to be thoroughly understood, before being able to perform sound flow measurements. Otherwise, flow data artifacts and data loss can be the consequence, potentially without being observed.**

This paper is the first of its kind to provide an integrated tutorial on all stages of a flow monitoring setup. As shown throughout this paper, flow monitoring has evolved from the early nineties into a powerful tool, and additional functionality will certainly be added in the future. We show, for example, how the previously opposing approaches of Deep Packet Inspection and flow monitoring have been united into novel monitoring approaches.

***Index Terms*—Flow export, network monitoring, Internet measurements, NetFlow, IPFIX**

I. Introduction

NETWORK monitoring approaches have been proposed and developed throughout the years, each of them serving a different purpose. They can generally be classified into two categories: active and passive. Active approaches, such as implemented by tools like Ping and Traceroute, inject traffic into a network to perform different types of measurements. Passive approaches observe existing traffic as it passes by a measurement point and therefore observe traffic generated by users. One passive monitoring approach is packet capture. This method generally provides most insight into the network traffic, as complete packets can be captured and further analyzed. However, in high-speed networks with line rates of up to 100 Gbps, packet capture requires expensive hardware and substantial infrastructure for storage and analysis.

Another passive network monitoring approach that is more scalable for use in high-speed networks is flow export, in which packets are aggregated into flows and exported for storage and analysis. A flow is defined in [1] as "a set of IP packets passing an observation point in the network during a certain time interval, such that all packets belonging to a particular flow have a set of common properties". These common properties may include packet header fields, such as source and destination IP addresses and port numbers, packet contents, and meta-information. Initial works on flow export date back to the nineties and became the basis for modern protocols, such as NetFlow and IP Flow Information eXport (IPFIX) [2].

In addition to their suitability for use in high-speed networks, flow export protocols and technologies provide several other advantages compared to regular packet capture. First, they are widely deployed, mainly due to their integration into high-end packet forwarding devices, such as routers, switches and firewalls. For example, a recent survey among both commercial and research network operators has shown that 70% of the participants have devices that support flow export [3]. As such, no additional capturing devices are needed, which makes flow monitoring less costly than regular packet capture. Second, flow export is well understood, since it is widely used for security analysis, capacity planning, accounting, and profiling, among others. It is also frequently used to comply to data retention laws. For example, communication providers in Europe are enforced to retain connection data, such as provided by flow export, for a period of between six months and two years "for the purpose of the investigation, detection and prosecution of serious crime" [4], [5]. Third, significant data reduction can be achieved – in the order of 1/2000 of the original volume, as shown in this paper – since packets are aggregated after they have been captured. Fourth, flow export is usually less privacy-sensitive than packet export, since traditionally only packet headers are considered. However, since researchers, vendors and standardization organizations are working on the inclusion of application information in flow data, the advantage of performing flow export in terms of privacy is fading.

Despite the fact that flow export, as compared to packet-level alternatives, significantly reduces the amount of data to be analyzed, the size of flow data repositories can still easily exceed tens of terabytes. This high volume, combined with the

Rick Hofstede, Anna Sperotto and Aiko Pras are with the University of Twente, Centre for Telematics and Information Technology (CTIT), P.O. Box 217, 7500 AE Enschede, The Netherlands (email: {r.j.hofstede, a.sperotto, a.pras}@utwente.nl).

Pavel Čeleda is with the Masaryk University, Institute of Computer Science, Botanická 68a, 602 00 Brno, Czech Republic (email: celeda@ics.muni.cz).

Brian Trammell is with ETH Zürich, Communication Systems Group, Gloriastrasse 35, 8092 Zürich, Switzerland (email: trammell@tik.ee.ethz.ch).

Idilio Drago is with the Politecnico di Torino, Department of Electronics and Telecommunications, Corso Duca Degli Abruzzi 24, 10129, Torino, Italy (email: idilio.drago@polito.it).

Ramin Sadre is with the Aalborg University, Department of Computer Science, Distributed and Embedded Systems, Selma Lagerlöfs Vej 300, 9220 Aalborg, Denmark (email: rsadre@cs.aau.dk).

Probing It Ourselves

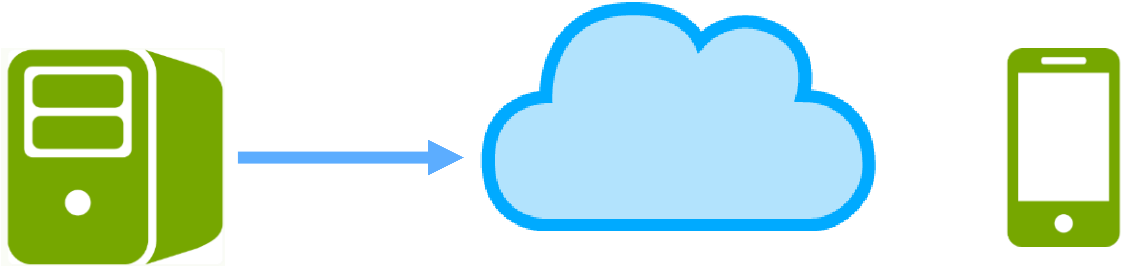
ACTIVE MEASUREMENTS

How to measure the data plane?

	Active	Passive
Examples	Ping, traceroute, scanning, …	Traffic monitoring, log files, …

Active measurements on the data plane send packets from end host(s) to other host(s).

It involves the network, transport, and usually the application layer.



Typical examples for active measurements

Internet delay analysis (round trip time)

Deployment of application layer services

DNS ecosystem

Web ecosystem

Certificate ecosystem

+++

Challenges

Coverage
Which sources and which destinations do you select to prevent a bias?

Performance
Sending many packets takes time, may challenge system resources etc.

Ethics
Easier to inject packets on the data plane compared to control plane, easier to introduce unintended effects

Protection
Depending on the measurement objective, source IP addresses should be whitelisted

Good practices

Add `Whois` entries for measurement prefixes

Add reverse DNS entries for source IP addresses

Create a web page that explains your project and lists a point of contact

If something goes wrong, operators want to know what is going on & who is responsible ;)

Expand the set of measurement probes

Building a dedicated distributed measurement infrastructure, which involves the deployment of specific hardware probes

Recruit users to run software probes

Two simple examples and what might go wrong

Ping

Send ICMP echo requests, wait for ICMP reply

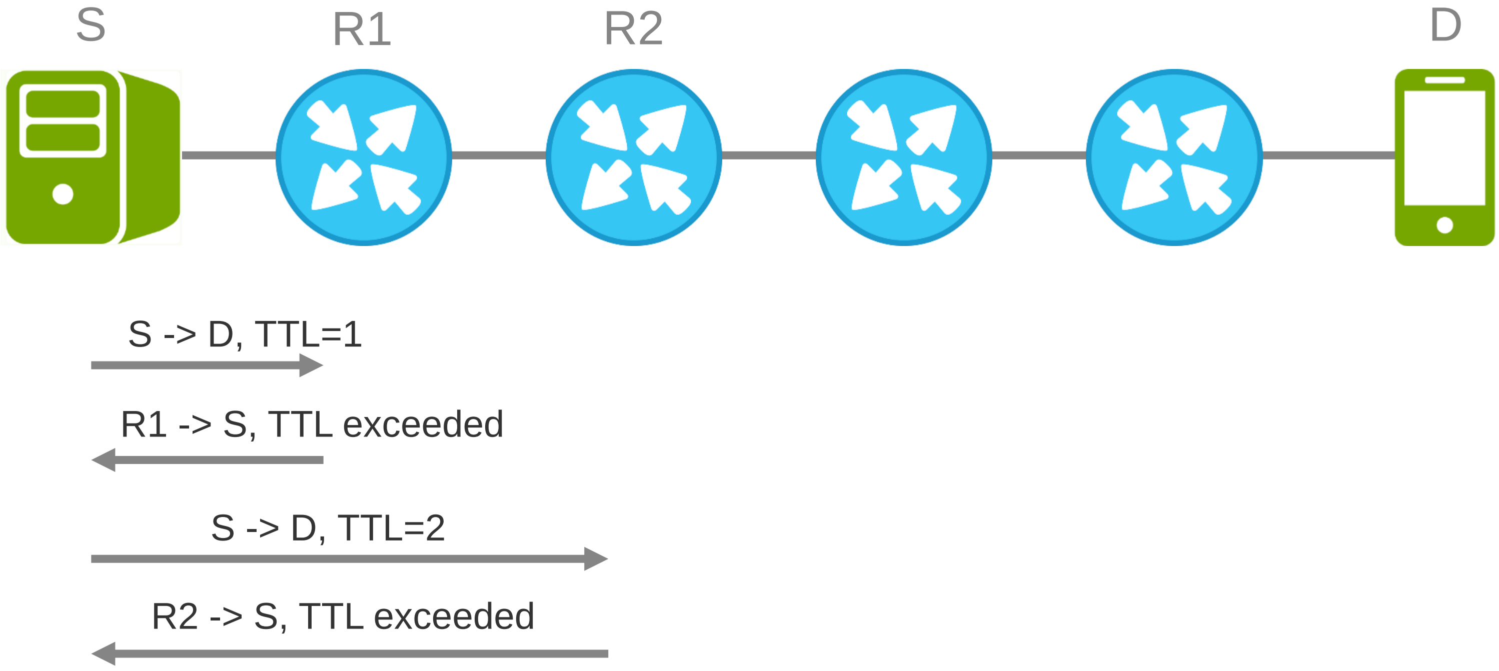
You measure the reachability of an end host, do you?

Traceroute

Probes the IP path

Keeps very few states

Traceroute: Principle approach



S -> D, TTL=1

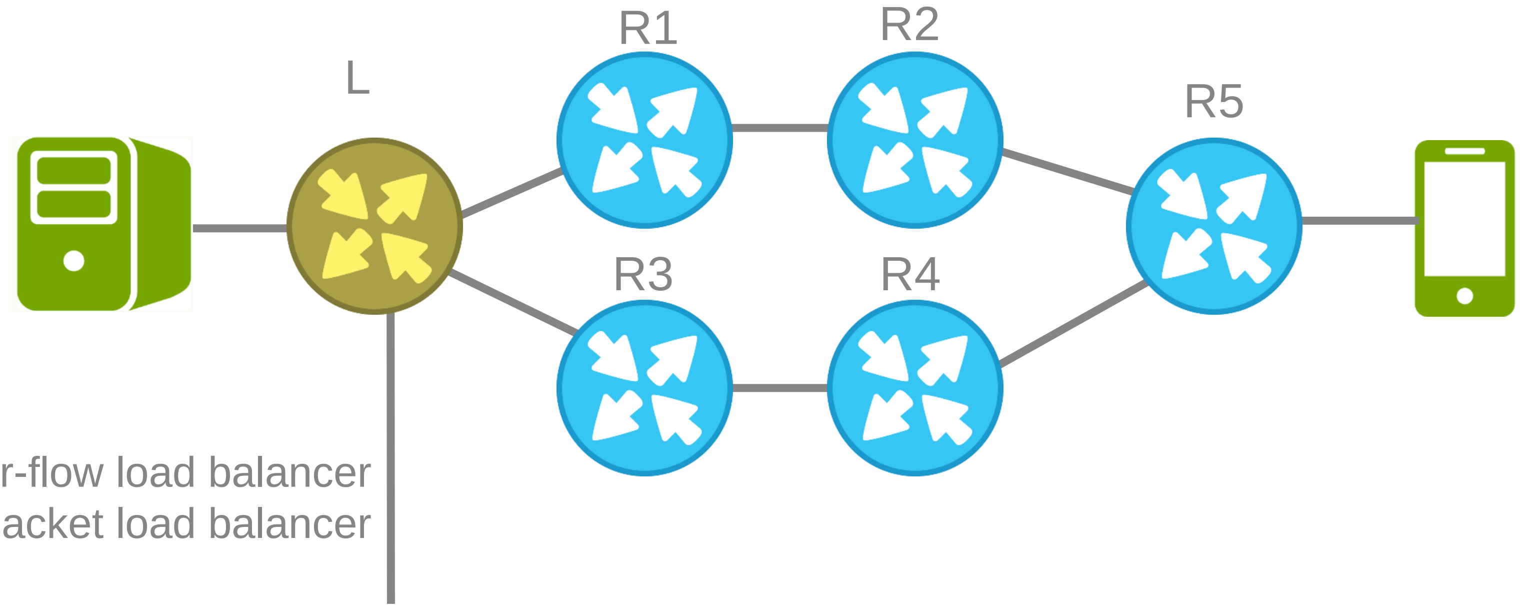
R1 -> S, TTL exceeded

S -> D, TTL=2

R2 -> S, TTL exceeded

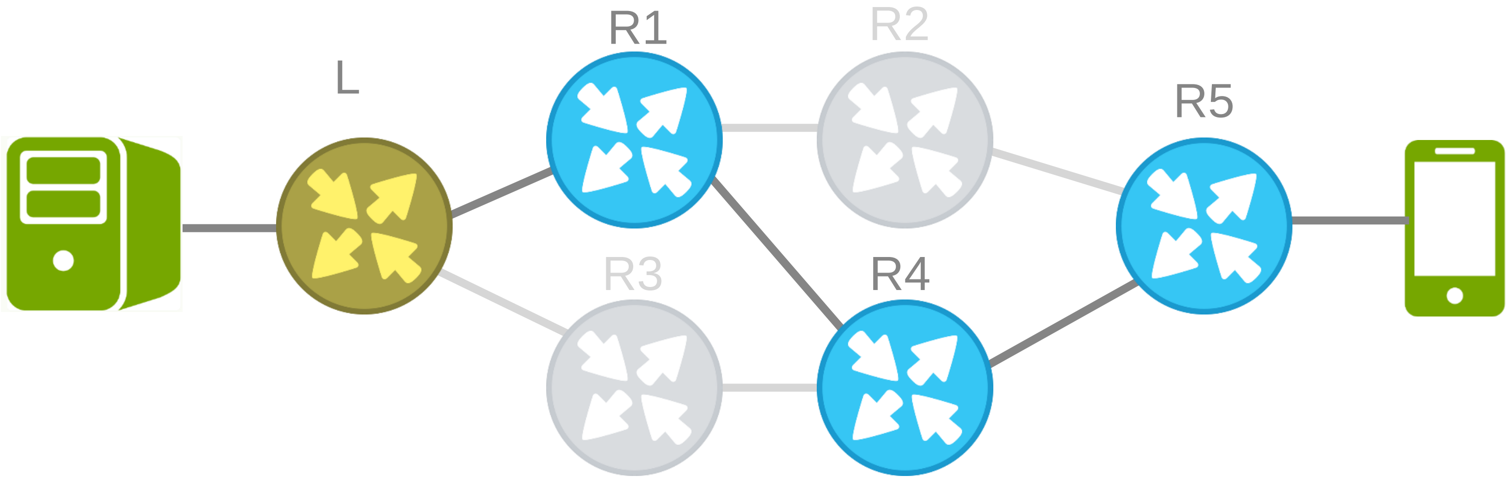
…

The problem of load balancers



Per-flow load balancer
Per-packet load balancer

The problem of load balancers



Missing nodes and links

False links

The core problem

Traceroute changes header fields

UDP traceroute: varies destination port

ICMP traceroute: varies sequence number

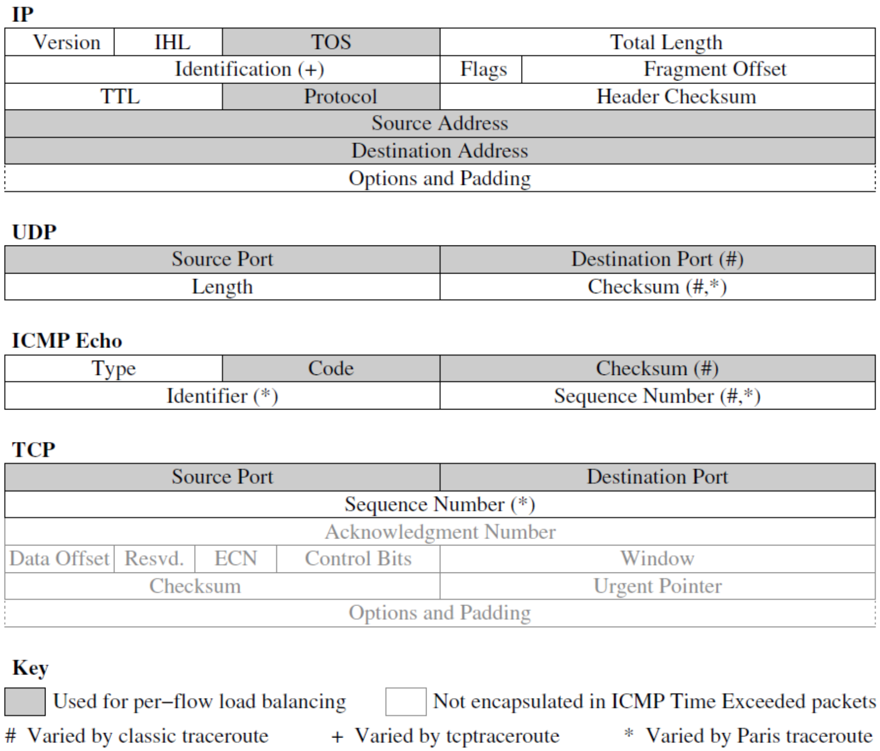
Many load balancers identify flows based on the first four octets of the transport header

Checksums cover even ‘back’ fields

The core problem & solution

Paris traceroute controls probe packet headers to overcome per-flow load balancing

Maintaining header fields is challenging because traceroute still needs to match request and reply



IP

Version	IHL	TOS	Total Length
Identification (+)		Flags	Fragment Offset
TTL	Protocol	Header Checksum	
Source Address			
Destination Address			
Options and Padding			

UDP

Source Port	Destination Port (#)
Length	Checksum (#,*)

ICMP Echo

Type	Code	Checksum (#)
Identifier (*)	Sequence Number (#,*)	

TCP

Source Port	Destination Port			
Sequence Number (*)				
Acknowledgment Number				
Data Offset	Resvd.	ECN	Control Bits	Window
Checksum	Urgent Pointer			
Options and Padding				

Key

▨ Used for per–flow load balancing □ Not encapsulated in ICMP Time Exceeded packets

# Varied by classic traceroute + Varied by tcptraceroute * Varied by Paris traceroute

Based on common header fields you can gain more information to discover anomalies

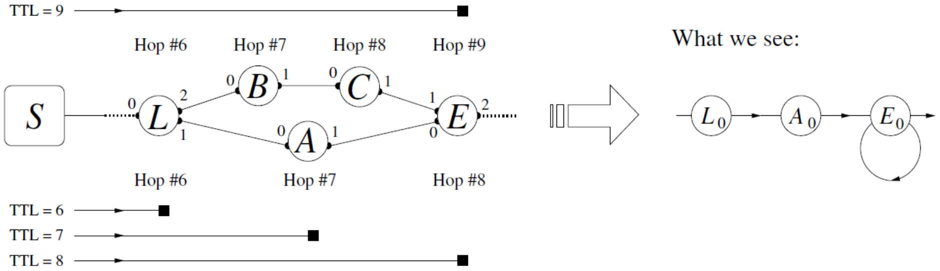
Probe TTL is in the encapsulated IP header echoed in ICMP Time Exceeded message and should be 1

Response TTL is the TTL in the IP header of the Time Exceeded msg. and should reflect the length of the return path

IP ID field set by the router and incremented for each packet send, helps for de-aliasing

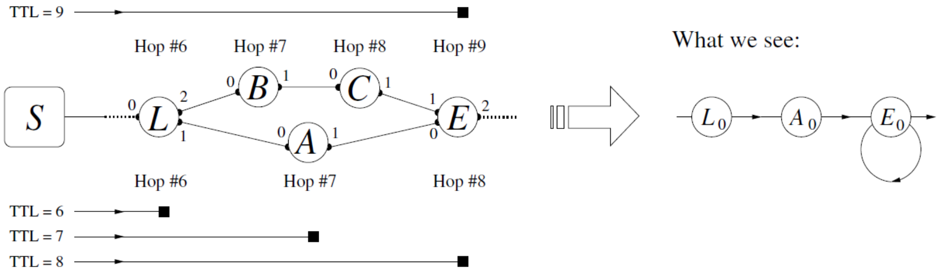
Anomalies in traceroute: Loops

Loop because of load balancing

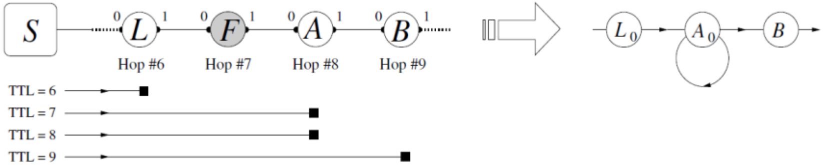


Anomalies in traceroute: Loops

Loop because of load balancing

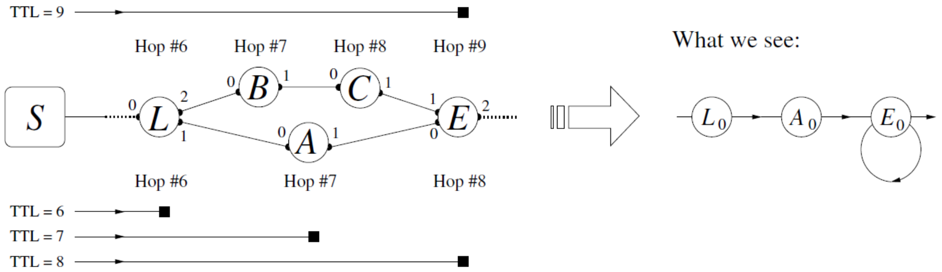


Loop because of zero-TTL forwarding

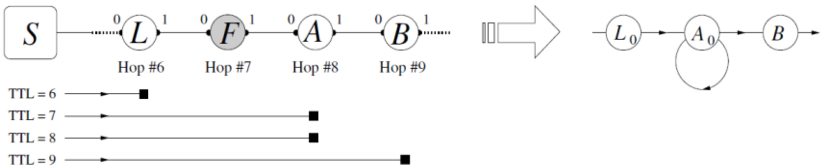


Anomalies in traceroute: Loops

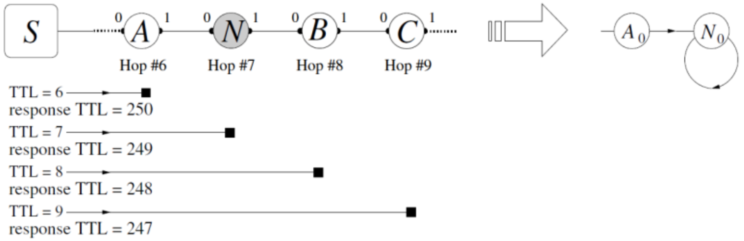
Loop because of load balancing



Loop because of zero-TTL forwarding



Loop because of address rewriting



Anomalies in traceroute: Loops

Destination unreachable messages needs special consideration

Anomalies in traceroute: Loops

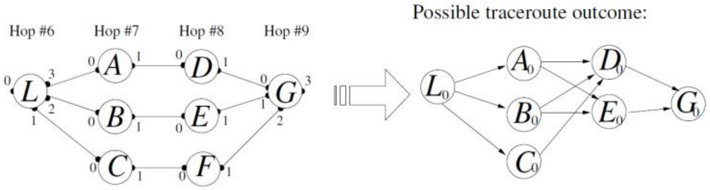
One month measurement study in 2006, to 5,000 randomly chosen nodes

Numbers to give you some idea

5% of the measured routes contained at least one loop

Loops because of load balancing: ~84%

Anomalies in classic traceroute: Cycles and Diamonds



Cycles

Load balancing and unreachability messages may lead to observed cycles, similar to loops

Diamonds

Arises only when multiple probes per hop are sent

Main cause: load balancing

Further challenges in traceroute

Routing path asymmetry

Routing policies, default routes, etc.

IP aliasing

How to distinguish multiple interfaces of the same router?

Literature

Brice Augustin, Xavier Cuvellier, Benjamin Orgogozo, Fabien Viger, Timur Friedman, Matthieu Latapy, Clémence Magnien, and Renata Teixeira. Avoiding traceroute anomalies with Paris traceroute. In *Proceedings of the 6th ACM SIGCOMM conference on Internet measurement* (IMC '06). ACM, New York, NY, USA, 153-158. http://dx.doi.org/10.1145/1177080.1177100

Avoiding traceroute anomalies with Paris traceroute

Brice Augustin*, Xavier Cuvellier*, Benjamin Orgogozo†, Fabien Viger†,
Timur Friedman*, Matthieu Latapy†, Clémence Magnien‡, and Renata Teixeira*

* Université Pierre et Marie Curie – CNRS, Laboratoire LIP6
† Université Denis Diderot – CNRS, Laboratoire LIAFA
‡ Ecole Polytechnique – CNRS, Laboratoire CREA

ABSTRACT

Traceroute is widely used, from the diagnosis of network problems to the assemblage of internet maps. However, there are a few serious problems with this tool, in particular due to the presence of load balancing routers in the network. This paper describes a number of anomalies that arise in nearly all traceroute-based measurements. We categorize them as "loops", "cycles", and "diamonds". We provide a new publicly-available traceroute, called *Paris traceroute*, which controls packet header contents to obtain a more precise picture of the actual routes that packets follow. This new tool allows us to find conclusive explanations for some of the anomalies, and to suggest possible causes for others.

Categories and Subject Descriptors: C.2.3 [Computer Communication Networks]: Network Operations

General Terms: Measurement.

Keywords: traceroute, load balancing.

1. INTRODUCTION

Jacobson's *traceroute* [1] is one of the most widely used network measurement tools. It reports an IP address for each network-layer device along the path from a source to a destination host in an IP network. Network operators and researchers rely on traceroute to diagnose network problems and to infer properties of IP networks, such as the topology of the internet.

This paper describes how traceroute fails in the presence of routers that employ load balancing on packet header fields. The failures lead to incorrect route inferences that may mislead operators during problem diagnosis and result in erroneous internet maps. We provide a new publicly-available traceroute, called *Paris traceroute* [1], which controls packet header contents to obtain a more precise picture of the actual routes that packets follow.

This paper highlights the problems of using classic traceroute for route inference by examining a number of topology artifacts that arise in traceroute-based measurements. We show that, using measurements from a single source tracing toward multiple destinations, one may observe anomalies that we categorize as "loops", "cycles", and "diamonds". We explain how many instances of these anomalies result from load balancing routers, and disappear when one uses Paris traceroute. We explain most other instances using additional information provided by Paris traceroute. Finally, we suggest possible causes for the remaining instances.

2. BUILDING A BETTER TRACEROUTE

This section first describes the deficiencies of the classic traceroute in the face of load balancing. Then we present our new traceroute, Paris traceroute, which avoids some of these deficiencies, notably the ones induced by per-flow load balancing.

2.1 Traceroute and load balancing

Network administrators employ load balancing to enhance reliability and increase resource utilization. They do so through the intra-domain routing protocols OSPF [2] and IS-IS [3] that support *equal cost multipath*. An operator of a multi-homed stub network can also use load balancing to select which of its internet service providers will receive which packets [4].

Routers can spread their traffic across multiple equal-cost paths using a per-packet, per-flow, or per-destination policy [5, 6]. In *per-flow load balancing*, packet header information ascribes each packet to a flow, and the router forwards all packets belonging to a same flow to the same interface. A natural flow identifier is the classic *five-tuple* of fields from the IP header and either the TCP or UDP headers: Source Address, Destination Address, Protocol, Source Port, and Destination Port. We found through our experiments that routers use various combinations of these fields, as well as three other fields: the IP Type of Service (TOS), and the ICMP Code and Checksum fields. We leave an exhaustive study of which header fields serve for load balancing, and in precisely which ways, to future work.

Per-flow load balancing ensures that packets from the same flow are delivered in order. *Per-packet load balancing* makes no attempt to keep packets from the same flow together, and focuses purely on maintaining an even load. *Per-destination load balancing* could be seen as a coarse form of per-flow load balancing, as it directs packets based upon the destination IP address. But, as it disregards source in-

[1] Paris traceroute is free, open-source software, available from `http://www.paris-traceroute.net/`.

IMC'06, October 25–27, 2006, Rio de Janeiro, Brazil.
Copyright 2006 ACM 1-59593-561-4/06/0010 ...$5.00.

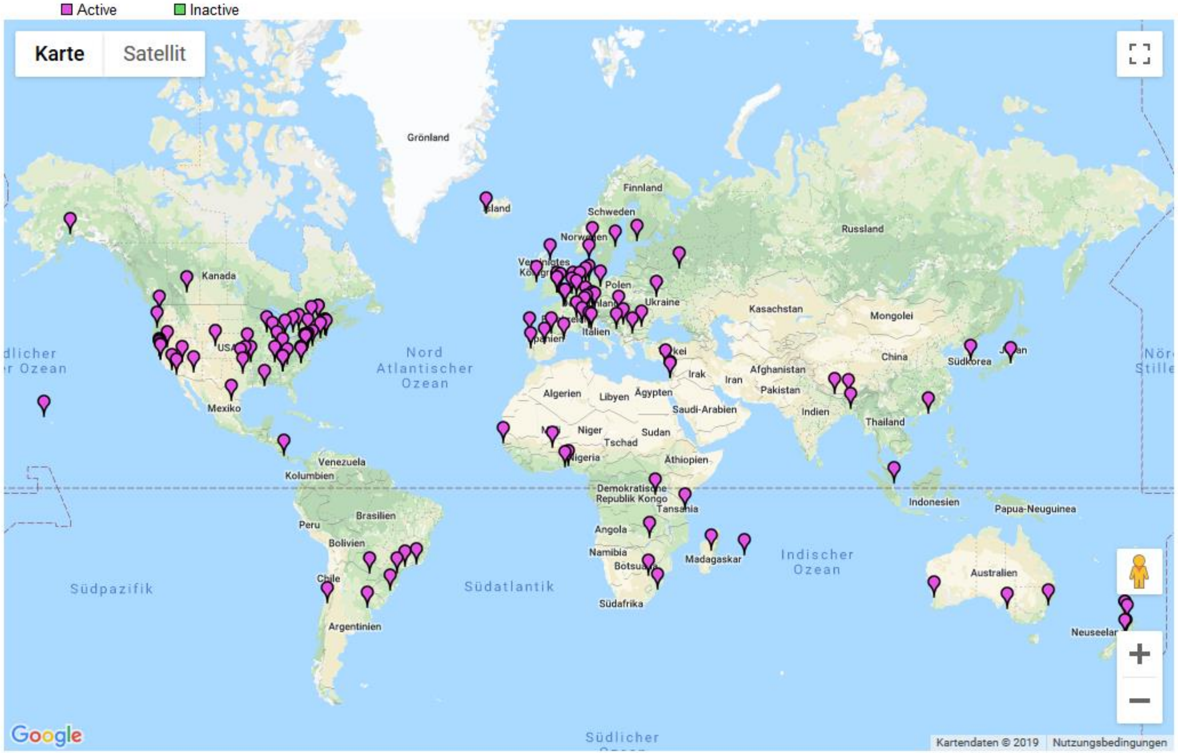
What Researchers Do for Us

COMMON MEASUREMENT INFRASTRUCTURES

CAIDA Archipelago (Ark, http://www.caida.org/projects/ark/)

Dedicated nodes that perform traceroutes and other measurements

Results are public

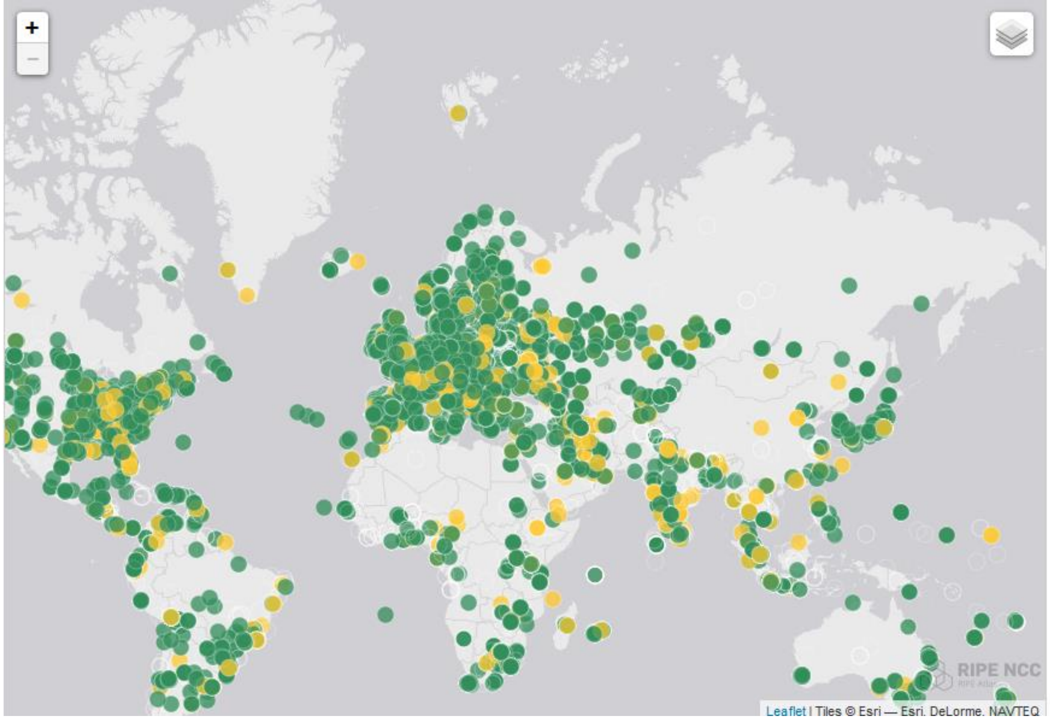


RIPE Atlas

Dedicated nodes common measurements

Credit-based system to perform own measurements

Results are public



RIPE Atlas in numbers

- 10,000 probes and 400 anchors connected worldwide
- 5.6% IPv4 ASes and 9% IPv6 ASes covered 181 countries covered
- 7,000 measurements per second

Most popular RIPE Atlas features

- Six types of measurements: ping, traceroute, DNS, SSL/TLS, NTP and HTTP (to anchors)
- APIs to start measurements and get results
- Powerful and informative visualisations: “Time Travel”, LatencyMON, DomainMON, TraceMon
- CLI tools
- Streaming data for real-time results
- Roadmap shows what’s completed and coming

Ethics design decisions (1)

- Active measurements only
  - probes do not observe user traffic
- Low barrier to entry
  - gratis probes, funded by LIRs and sponsors
- Hosted by volunteers
  - informed consent (accepting T&C)
  - personal data never revealed
- Data, API, source code, tools: free and open
- Measurements sets limited

Ethics design decisions (2)

- No bandwidth measurements
  - Other platforms provide that service
- HTTP measurements only towards RIPE Atlas anchors
  - Otherwise it would rely on hosts' bandwidth
  - And might put volunteer at risk