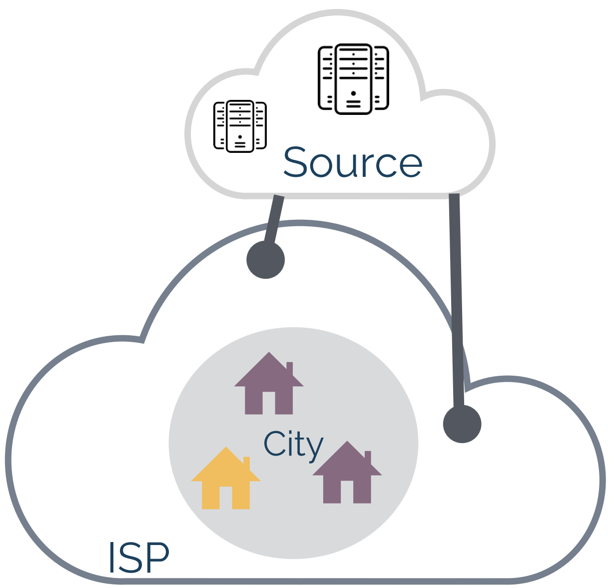
IPD: Detecting Traffic Ingress Points at ISPs

● Stefan Mehner, ▲Helge Reelfs ,

◆ Ingmar Poese, ● Oliver Hohlfeld

●University of Kassel
▲Brandenburg University of Technology
◆BENOCS

“Service degradation for parts of a city”



Question 1: is it the subscriber base causing service degradation ?

Question 2: are overloaded links on the path suddenly causing this ?

Question 3: have sources/upstream providers changed something ?

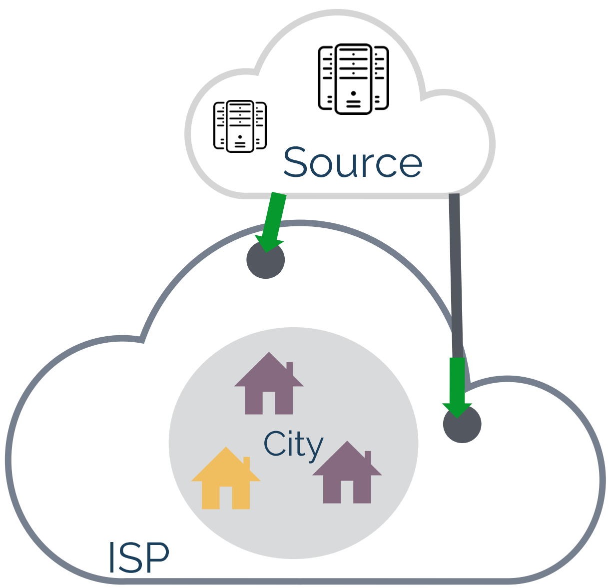
Question ?: or something … else ?

It is relatively easy to find the IP adresses involved in the affected traffic

It is hard to find out where the traffic entered the network

Yet that is more often than not needed for fixing the issue

Why is the ingress point important ?

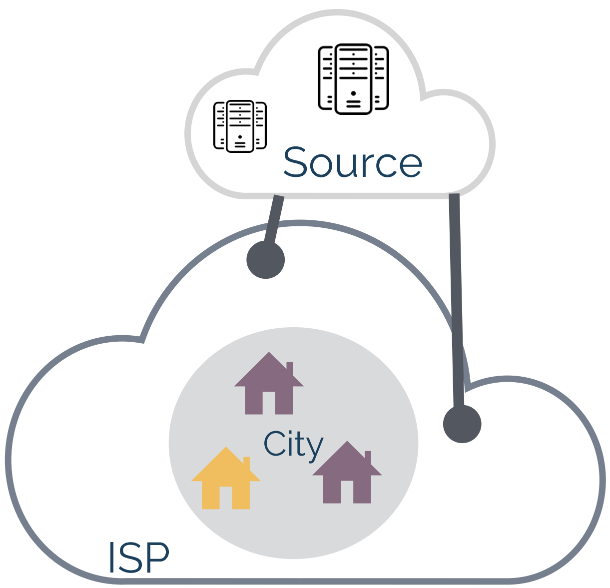


Ingress Point - definition:
The link where traffic enters the network.

Ingress Point is the first time that traffic touches the network.
From there, traffic is only forwarded
Forwarded traffic can be followed

- enables use of IGP/BGP in analysis
- allows reasoning with routing policies/TE
- behavior can be tracked along the path

Design Space



Our solution was designed to:

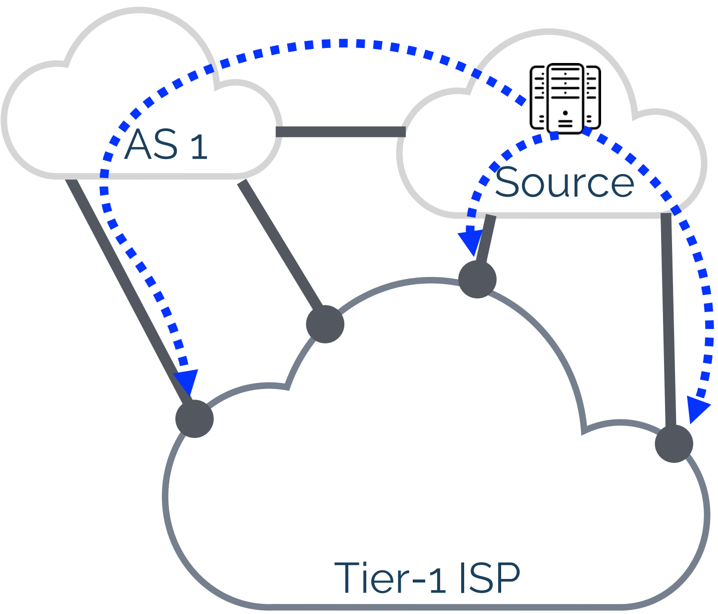
- Track all ingressing traffic
- Adapt automatically to changes
- Keep history for later review/post mortem
- Be light weight in terms of hardware

Limitations:

- Focus only on heavy traffic sources
- No reasoning about unseen sources
- Router based load balancing is unsupported

Ingress Traffic Engineering

Use Case: CDN-ISP traffic steering collaboration



Map subscribers of the ISP optimally to the CDN servers

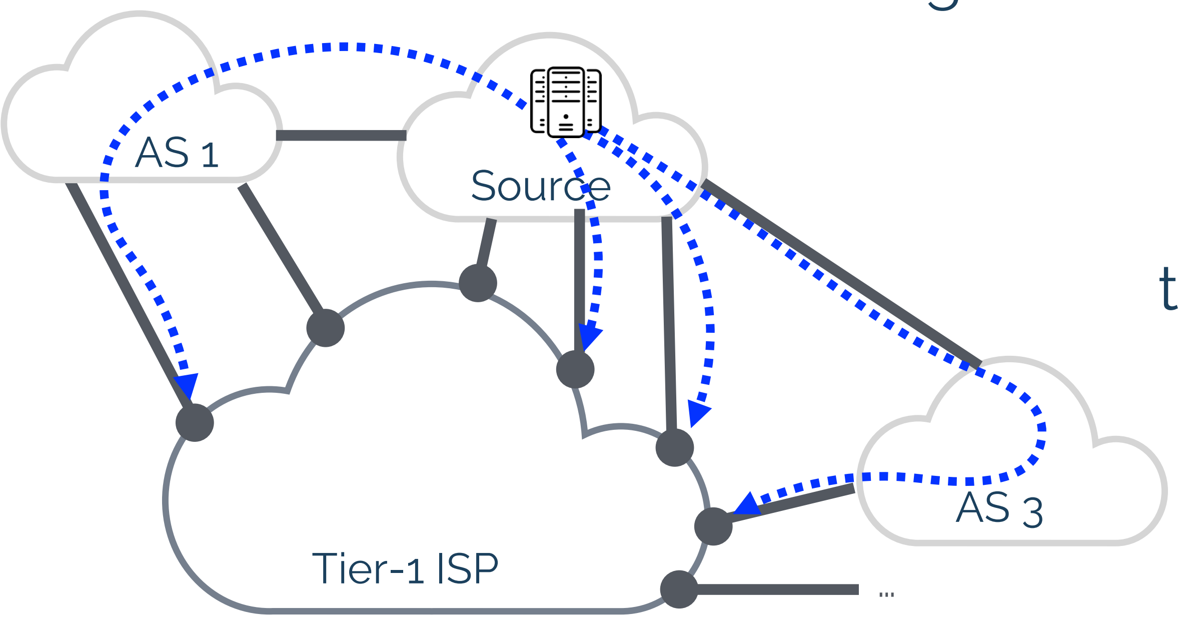
Main challenge: Where does the traffic enter the ISP ?

For mapping sources to subscribers the ingress points are needed

Design Challenges

Scale

Challenge:
monitoring traffic at **all border routers**



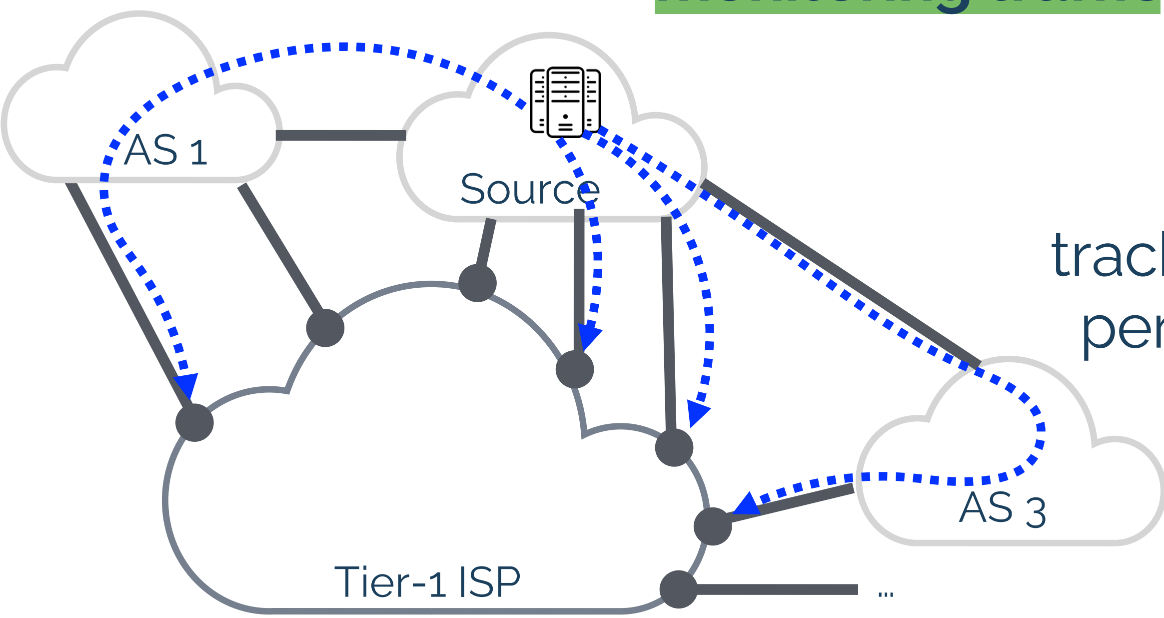
Observed ISP:
thousands of border routers

Design Challenges

Granularity

Challenge:
monitoring traffic at all border routers

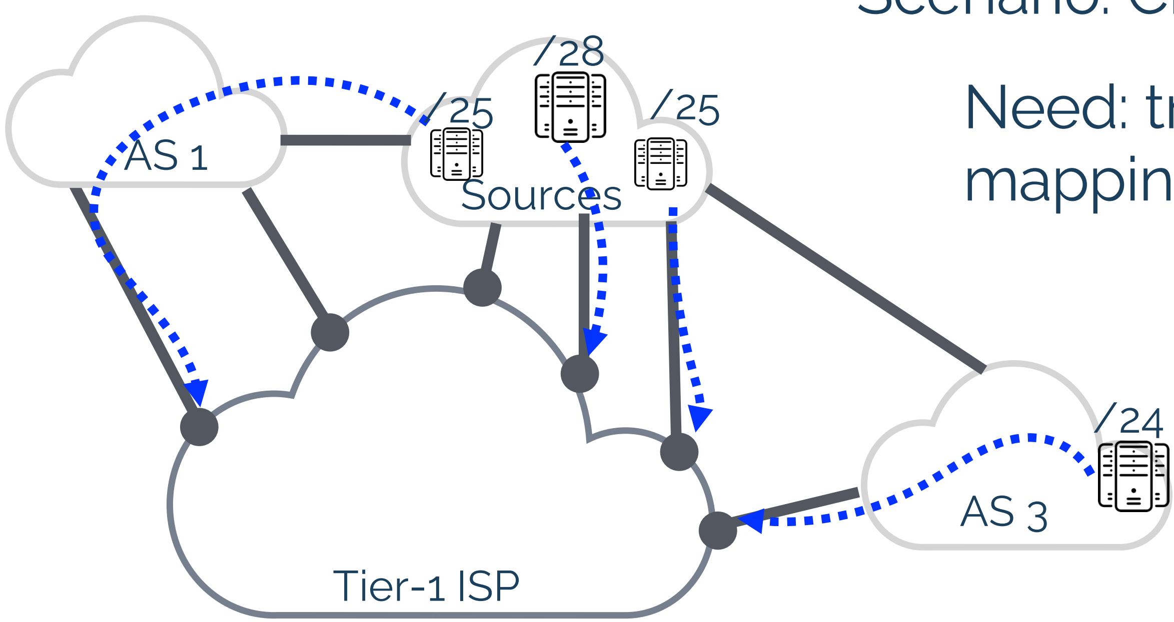
Problem:
tracking ingress points per IP does not scale



Which tracking granularity?

Design Challenges

Traffic-based aggregation of IP ranges



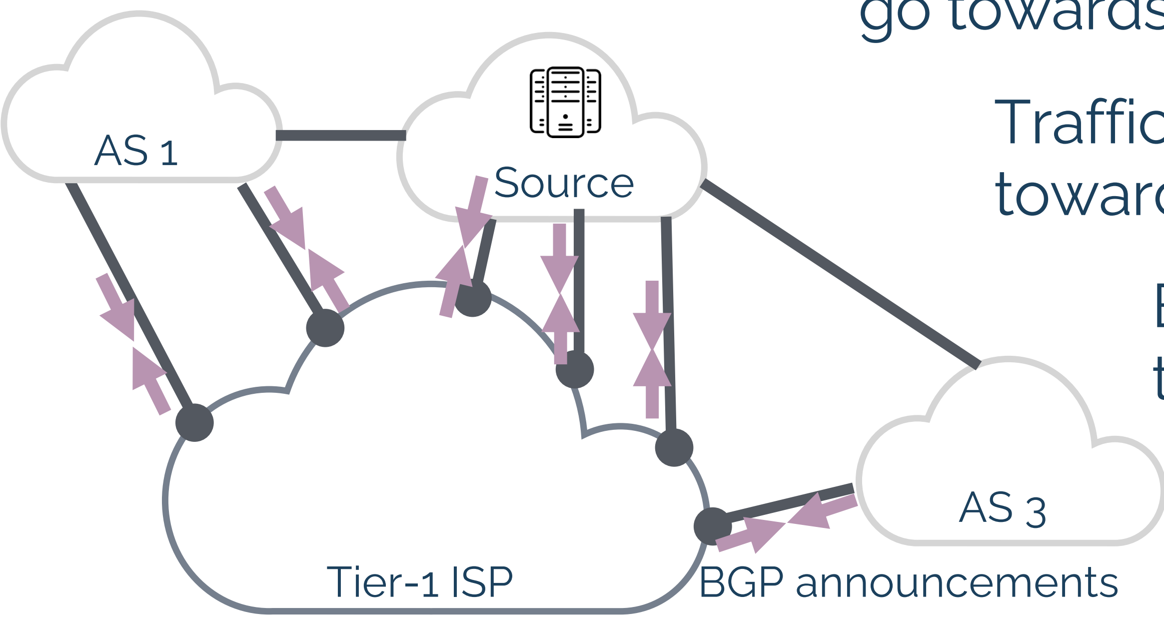
Scenario: CDN traffic steering

Need: tracking granularity at CDN mapping granularity

Requirement: **dynamically** infer aggregation prefixes from traffic

Design Aspects

BGP is not an option!



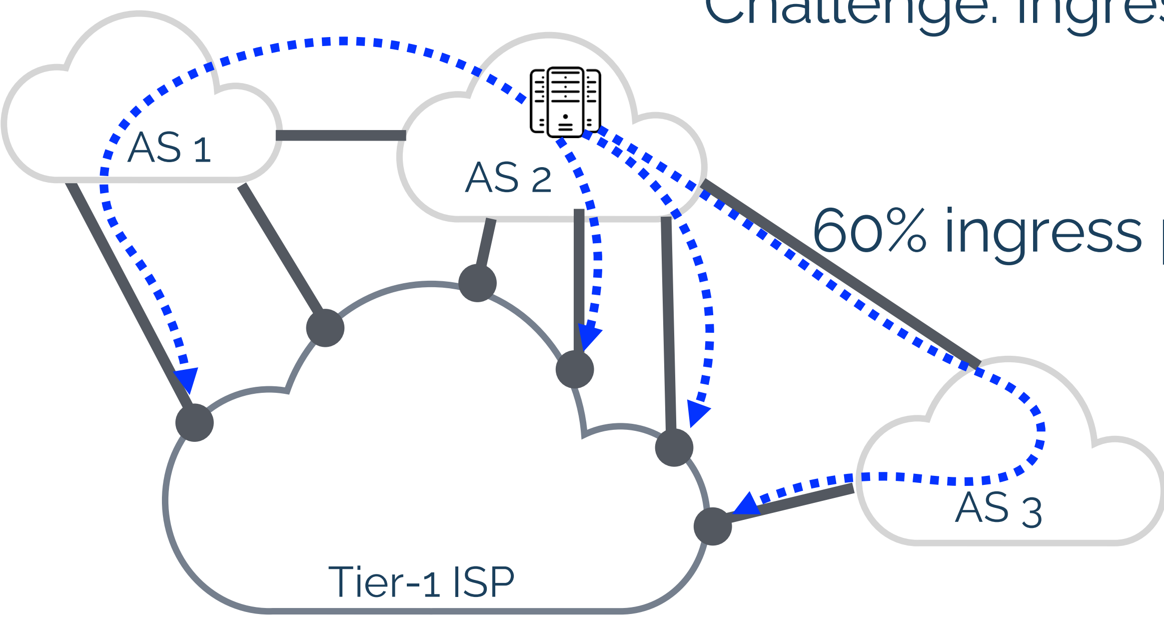
Relevant BGP announcements go towards the traffic source

Traffic flows form the source towards the announcement

BGP announcements point the wrong way !

Design Aspects

Continuous monitoring



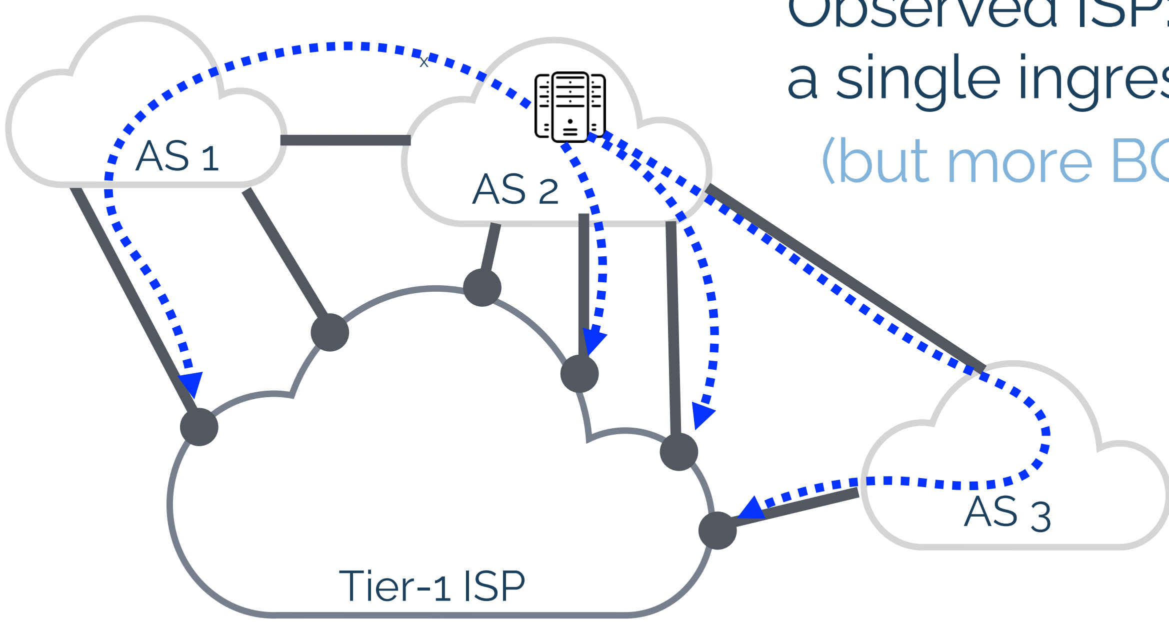
Challenge: ingress points **change often**

Our ISP:
60% ingress points stable for <1 hour

Requirement:
detection within minutes

Design Aspects

Focus on dominant ingress links

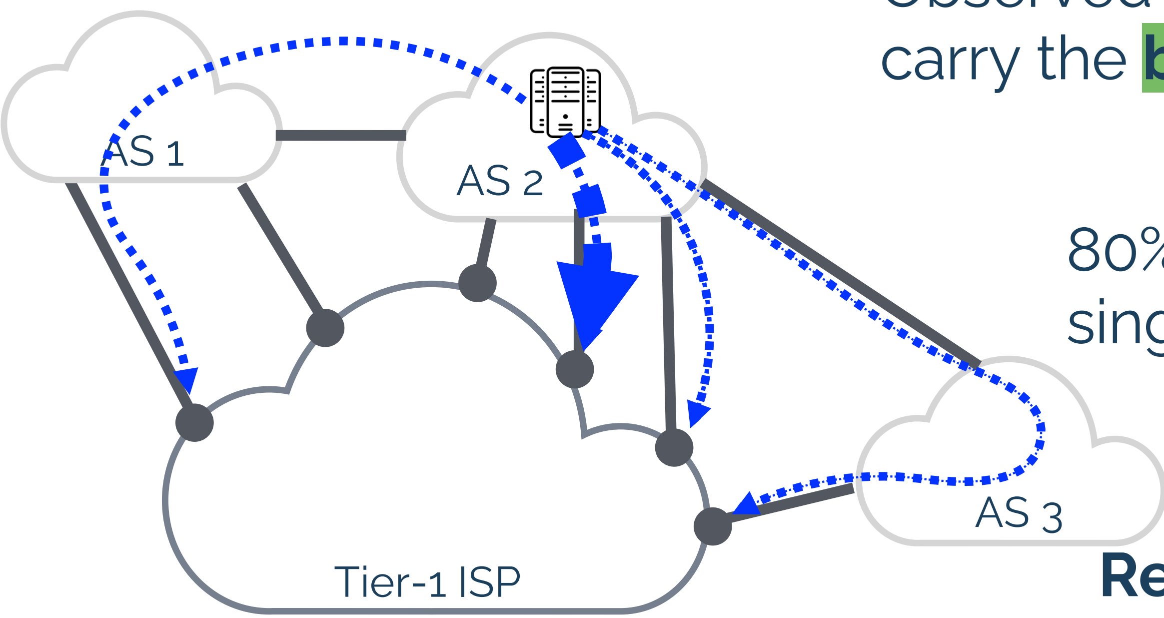


Observed ISP: most prefixes only have a single ingress point

(but more BGP paths exist)

Design Aspects

Focus on high-traffic prefixes



Observed ISP: single ingress points carry the **bulk of traffic**

80% prefixes:
single ingress: 80% traffic

Requirement:
detect dominant ingress point

Ingress Point Detection at ISPs

Requirements:

No-Input: BGP

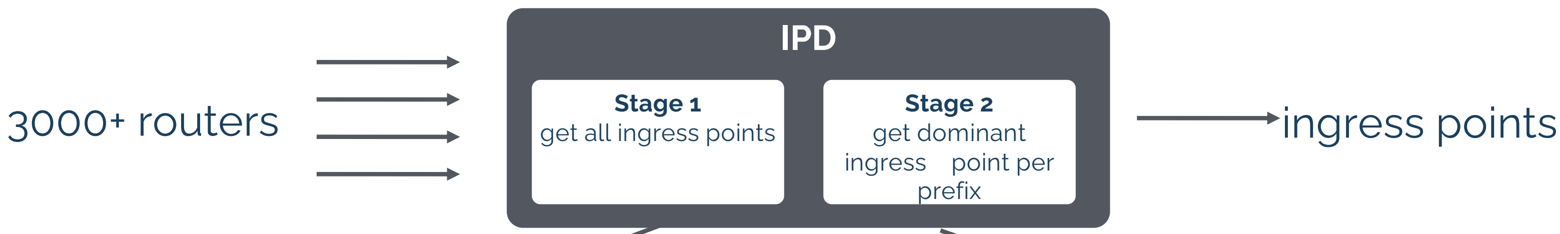
Input: Netflow

Scale:
- 1000+ Netflow sources
- Continuous monitoring
- Dynamic prefix aggregation

Scope:
- All links ingressing traffic
- Source address tracking

Goal:
- Focus on high traffic prefixes
- Dominant ingress point

Ingress Point Detection Algorithm



many CPU cores

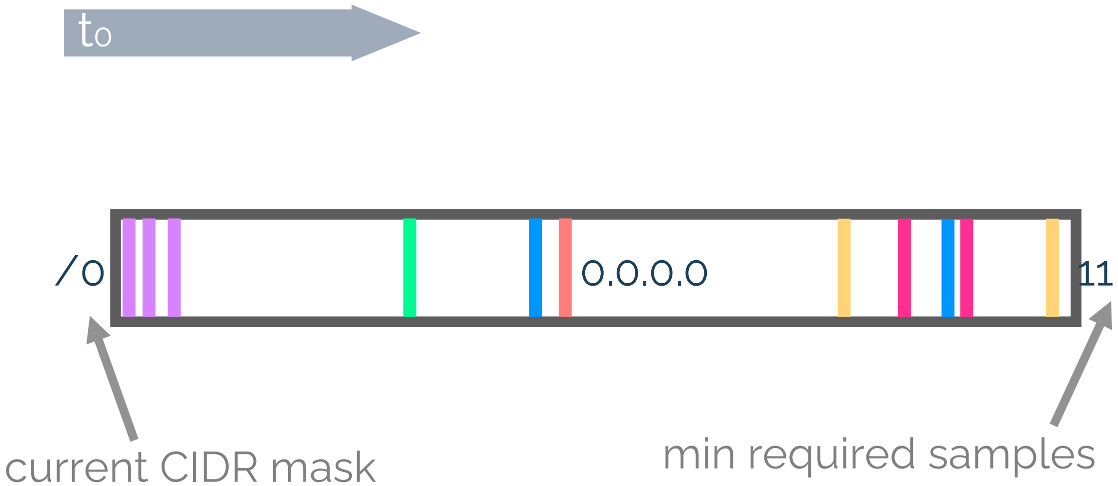
1 CPU core

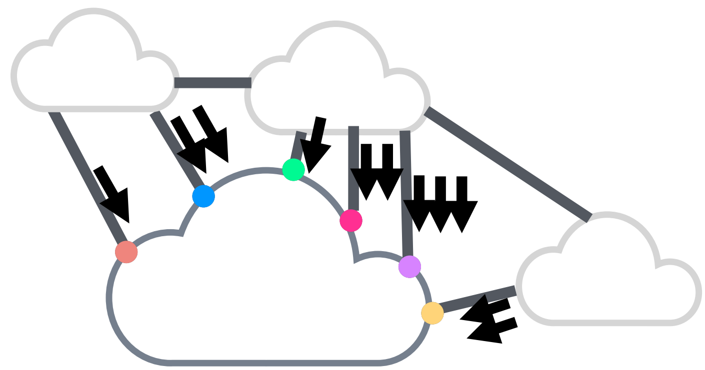
Challenge Stage 1:
massive influx of Netflow

Goal: only collect / ingest / prepare data

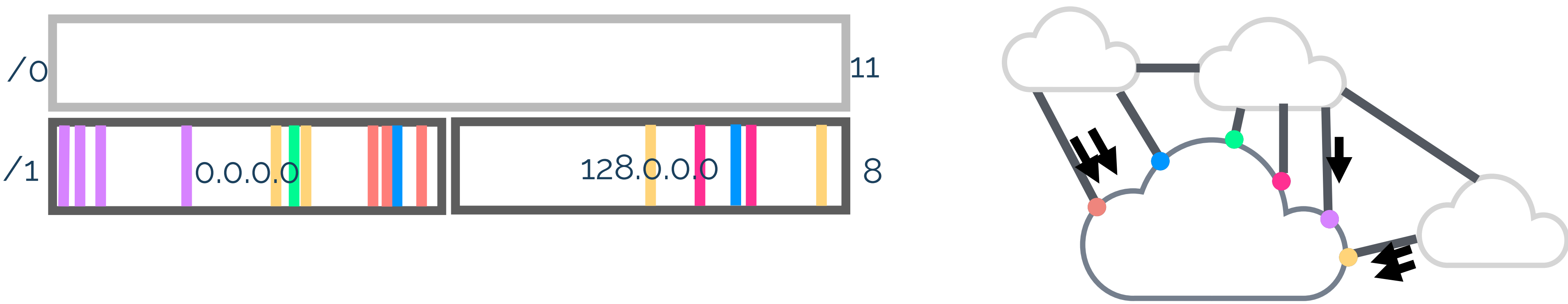
Challenge Stage 2:
classify ranges / find dominant ingress point

Ingress Point Detection Algorithm

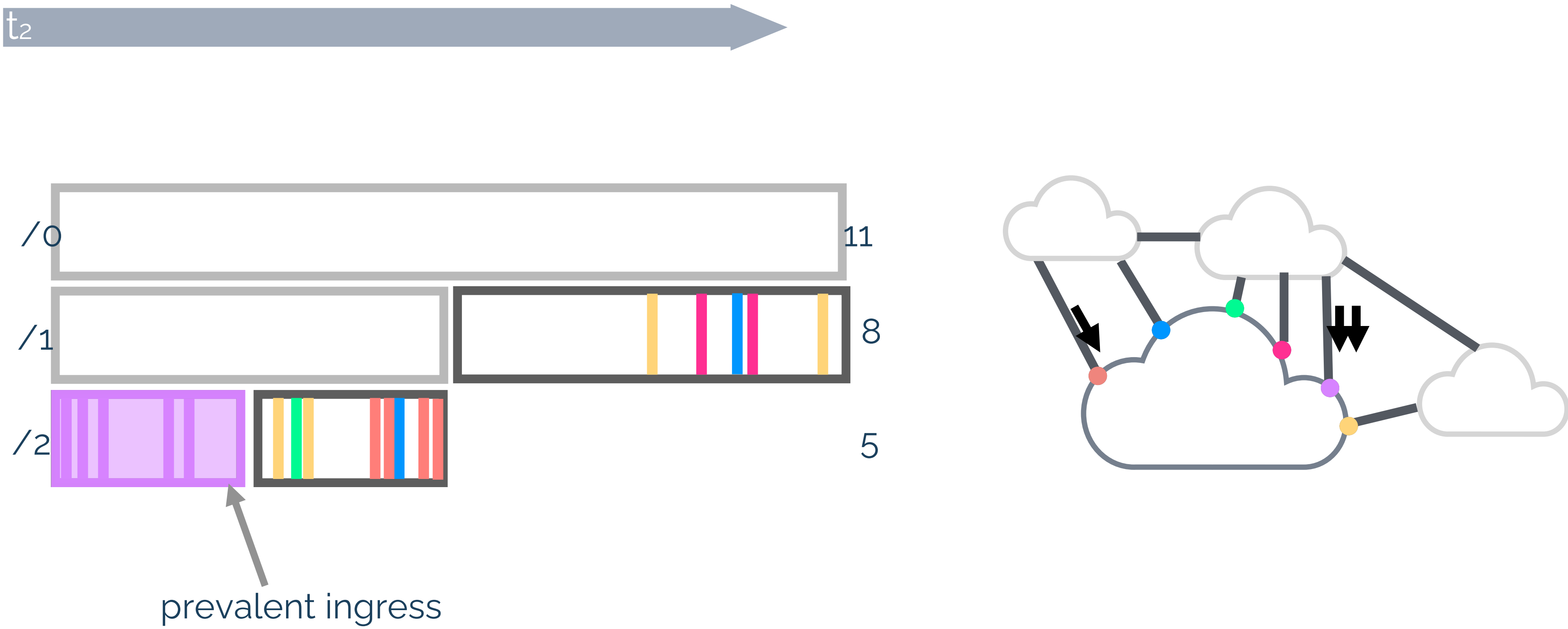




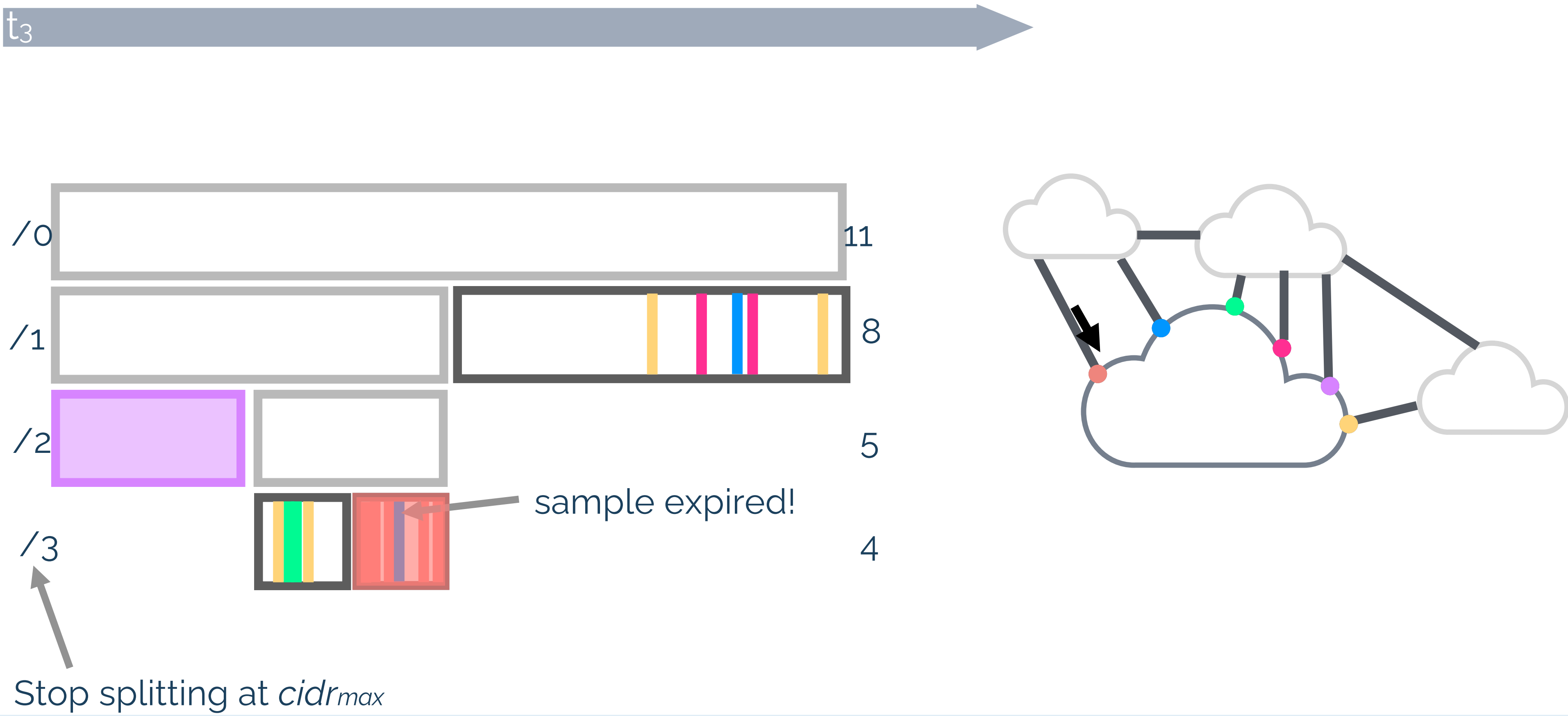
Ingress Point Detection Algorithm



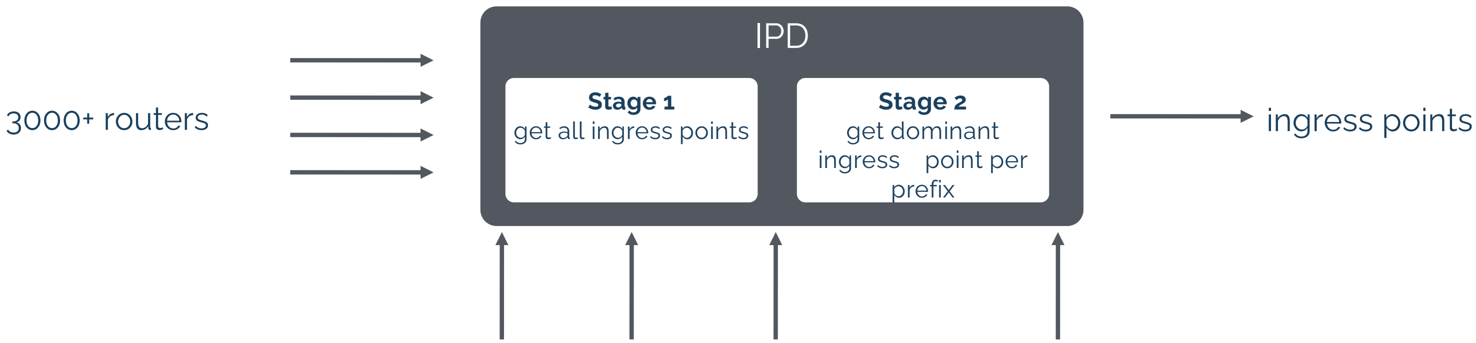
Ingress Point Detection Algorithm



Ingress Point Detection Algorithm



Parameter Study



6 parameter: $cidr_{max}$ … error margin time bucket length

Our default: /28 5% 60 sec

Parameter study with 300+ combinations to infer optimal parametrization

System Requirements

Our IPD has been running algorithm at a Tier-1 ISP for 6 years

Single commodity server for an entire ISP is enough

~30 cores + 120GB RAM in use

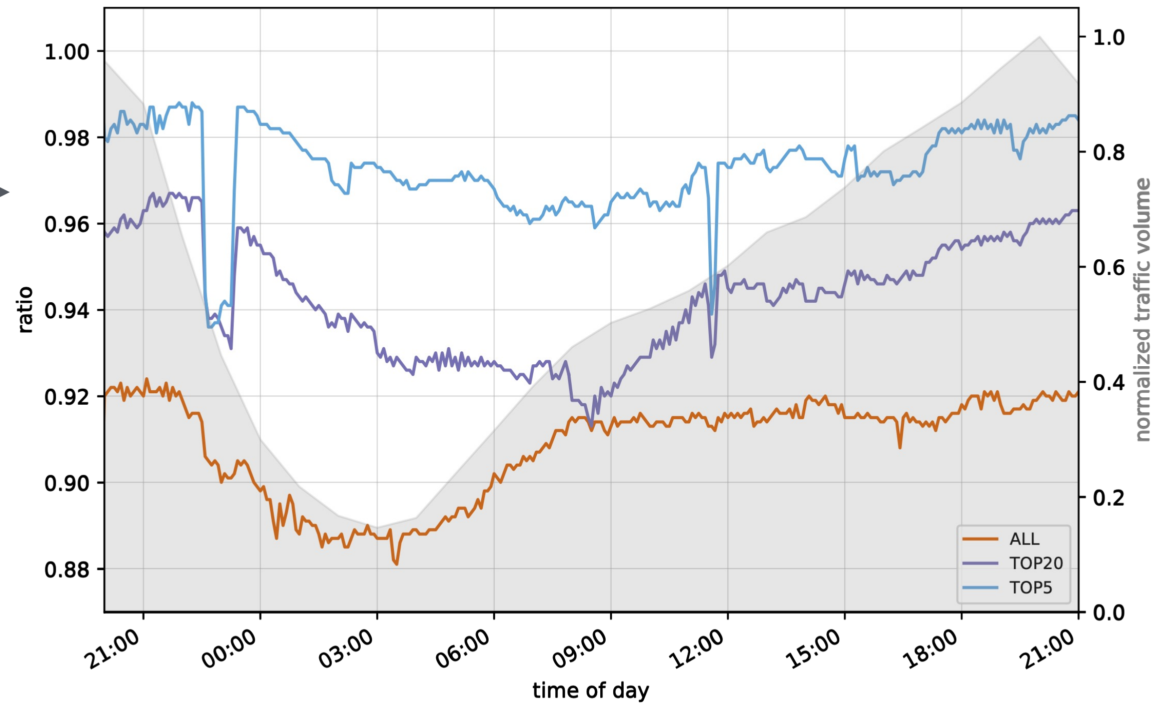
IPD classifications are accurate

Matching IPD results against Netflow

97,4% for top 5 ingress ASes by traffic volume

Reasons for inaccuracies:

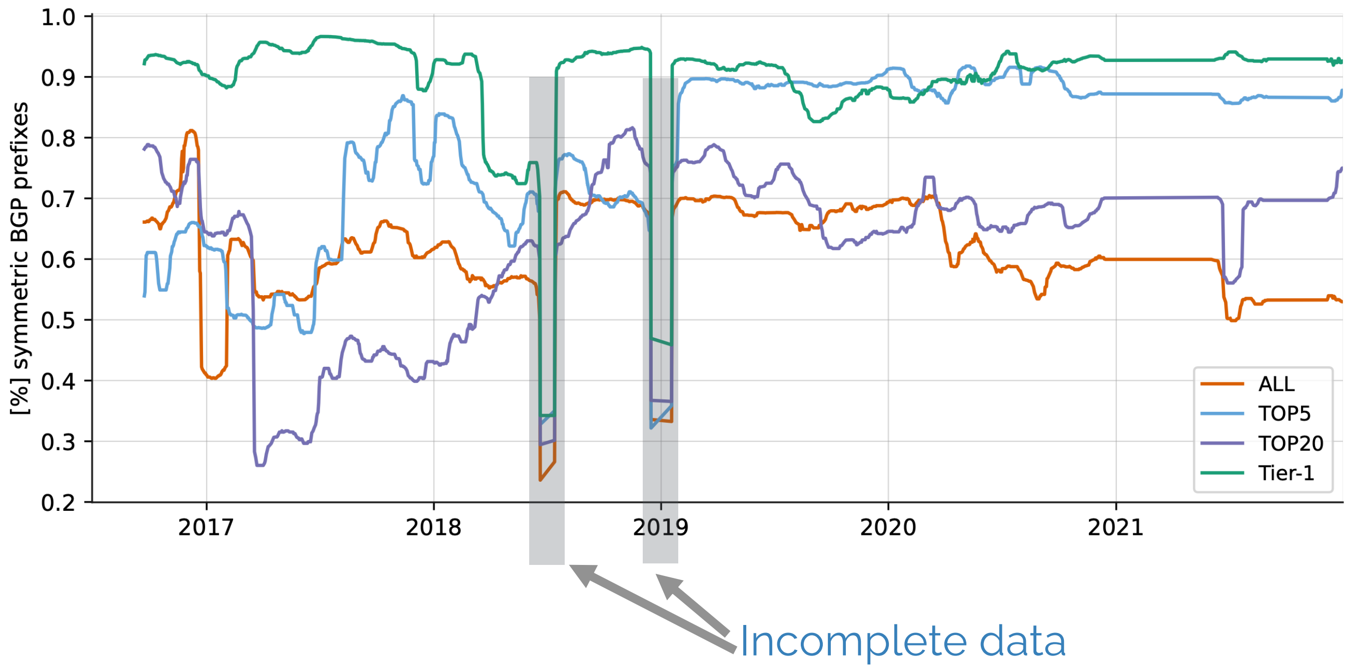
- traffic shifts
- operational changes, …



IPD works well enough in practice for the ISP to build services on top

Don't assume path symmetry

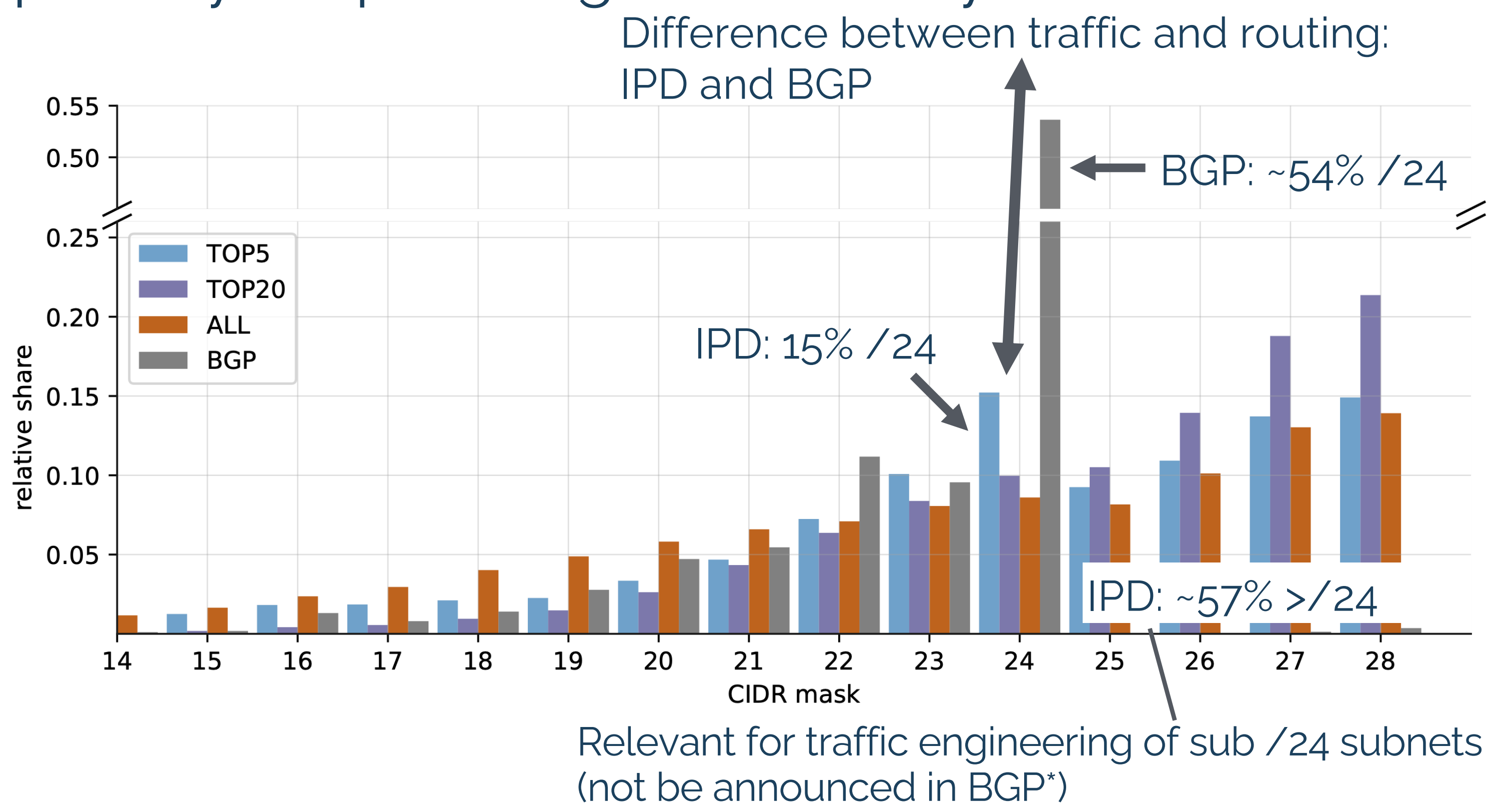
IPD enables first measurement study of path asymmetry at a Tier-1



AS paths can be asymmetrical - BGP cannot be used for IPD

BGP and IPD prefix sizes differ

IPD optimally adapts to ingress traffic dynamics



Operational Experience

IPD enables: Network trouble shooting

CDN-ISP collaboration / ingress traffic engineering

Talking to interconnected ASes about problems

IPD omits: Router-level load balancing (quadratic additional complexity)

IPD is a valuable tool to the tier-1 ISP that is in operation without change for 6 years

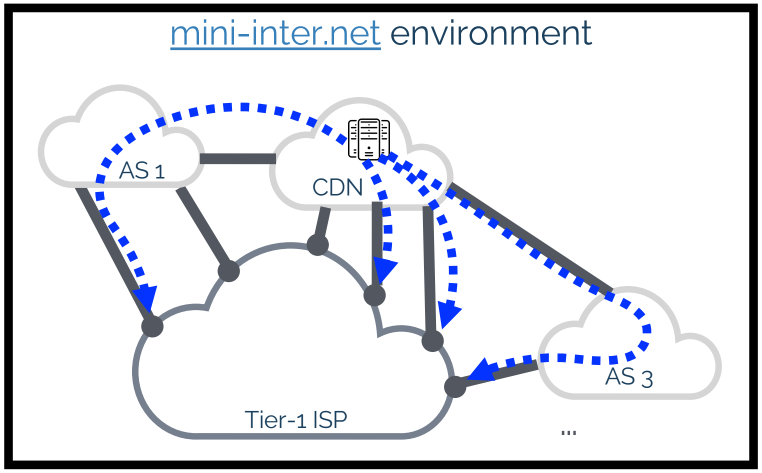
Thank you!

Dr.-Ing. Ingmar Poese

ipoese@benocs.com

BENOCS

Mini-IPD: Experiment with IPD yourself



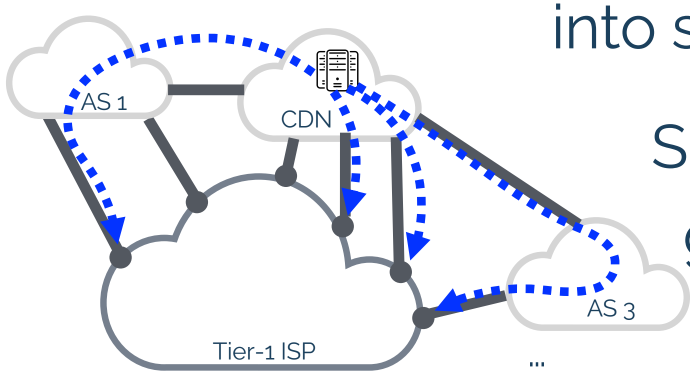
Prototypic IPD implementation:
github.com/smehner1/ipd

IPD in an emulated ISP scenario with CDNs:
github.com/smehner1/mini-ipd

IPD: Ingress Point Detection at ISPs

Stefan Mehner, Helge Reelfs,

Ingmar Poese, Oliver Hohlfeld



IPD infers traffic ingress points

Traffic based partitioning the IP address space into segments sharing the same ingress point

Scales to a tier-1 ISP on a single server

github.com/smehner1/mini-ipd

IPD vs. TIPSY

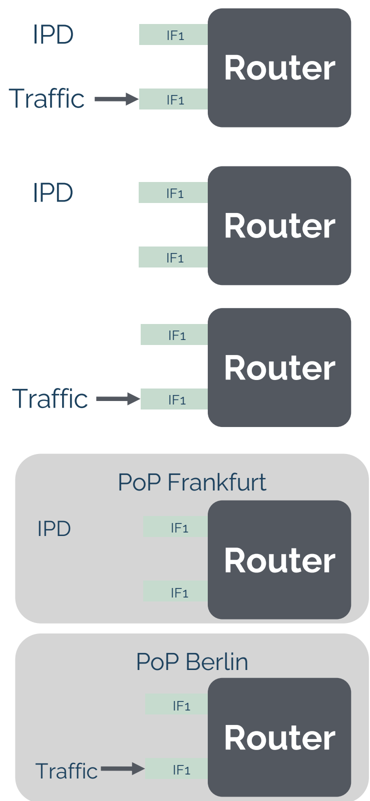
	TIPSY	IPD
Focus	Cloud Provider	ISP
Method	Statistical model of ingress traffic volumes and points Predict effect of shifting traffic by selective BGP withdrawals for prefixes observed in training period	Traffic-based partitioning of the entire IP address space
Granularity	/24	Dynamically up t o a predefined maximum CIDR mask (/28 in operational setting)
Use Case	Congestion Management	Network debugging Joint CDN-ISP traffic steering

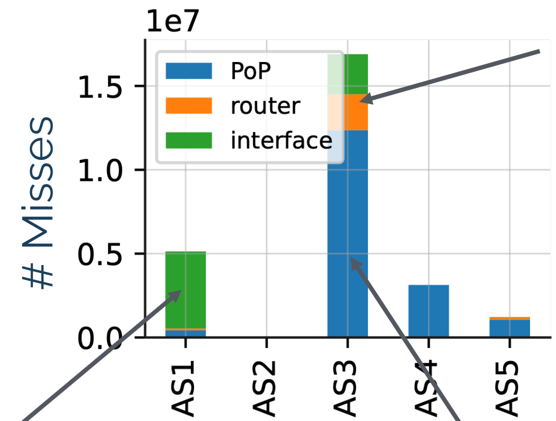
3 types of misses can/do happen

Interface miss

Router miss

PoP miss





AS3: router-level load balancing

AS1: 65% misses: small prefixes (/25 to /27)

IPD identified bundle, misses the other iface

AS3: few prefixes enter via a different country

CDN misalignment

IPD Parameter

Parameter	Default	Meaning
$cidr_{max}$	/28, /48	max. *IPD* prefix length
$n_{cidr}factor$	64, 24	minimal sample factor $n_{cidr} = n_{cidr}factor * \sqrt{2^{(32-s_{cidr})}}$
q	0.95	error margin
t	60	time bucket length
e	120	expiration time
$decay$	1- $\frac{0.9}{(\frac{age}{t})+1}$	factor to reduce outdated *IPD* ranges

Parameter study: 308 combinations

factor	level(s)
t	[60]
e	[120]
q	[0.501, 0.7, 0.8, 0.95, 0.99]
$n_{cidr}factor_4$	[32, 48, 64, 80]
$n_{cidr}factor_6$	[12, 18, 24, 30]
$cidr_{max4}$	[20, 21, 22, 23, 24, 25, 26, 27, 28]
$cidr_{max6}$	[32, 34, 36, 38, 40, 42, 44, 46, 48]

Evaluation of each parameter set against 1 day of Netflow

IPD parameter do not change accuracy, but run-time and resource consumption