IPD: Detecting Traffic Ingress Points at ISPs

Stefan Mehner, Helge Reelfs ,

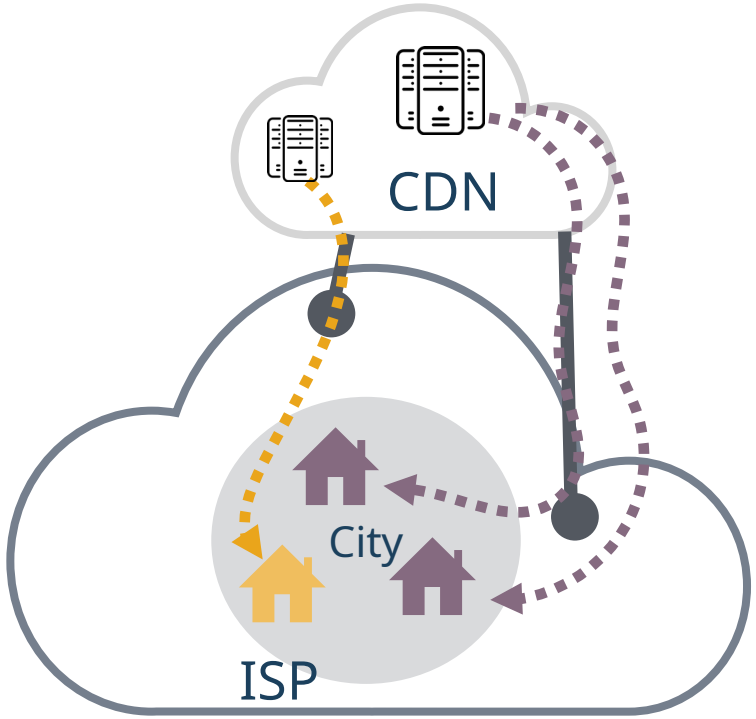
Ingmar Poese, Oliver Hohlfeld

University of Kassel

Brandenburg University of Technology

BENOCS

“Why is *$service* slow for some customers?”



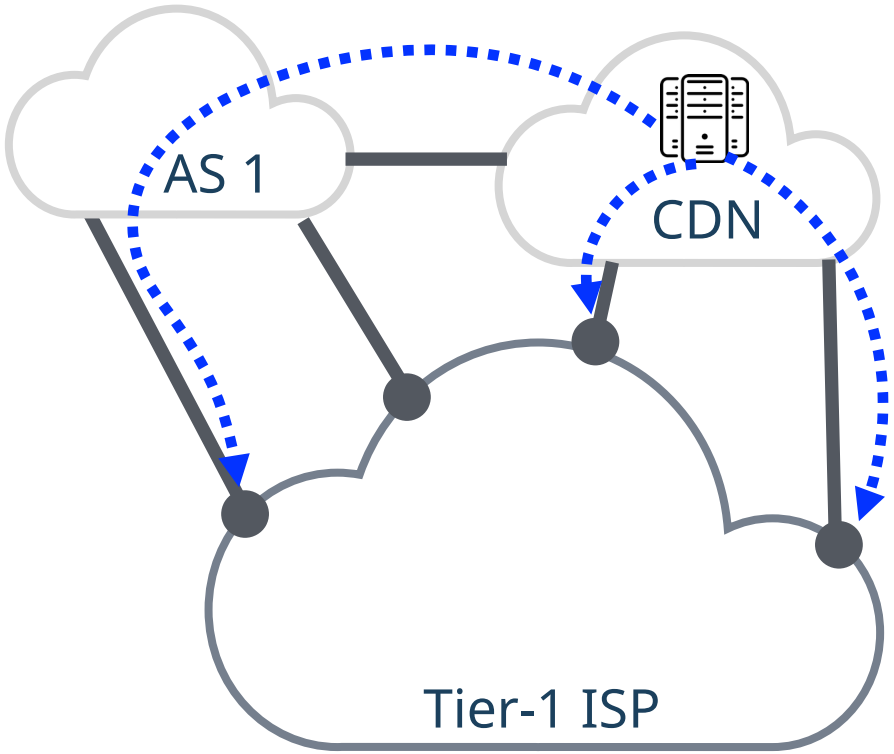
Problem: incorrect CDN mapping of customer prefixes

Easy to find mismatching source IP for subscribers

Hard to find source IP ingress point into ISP

Ingress Traffic Engineering

Use Case: CDN-ISP traffic steering



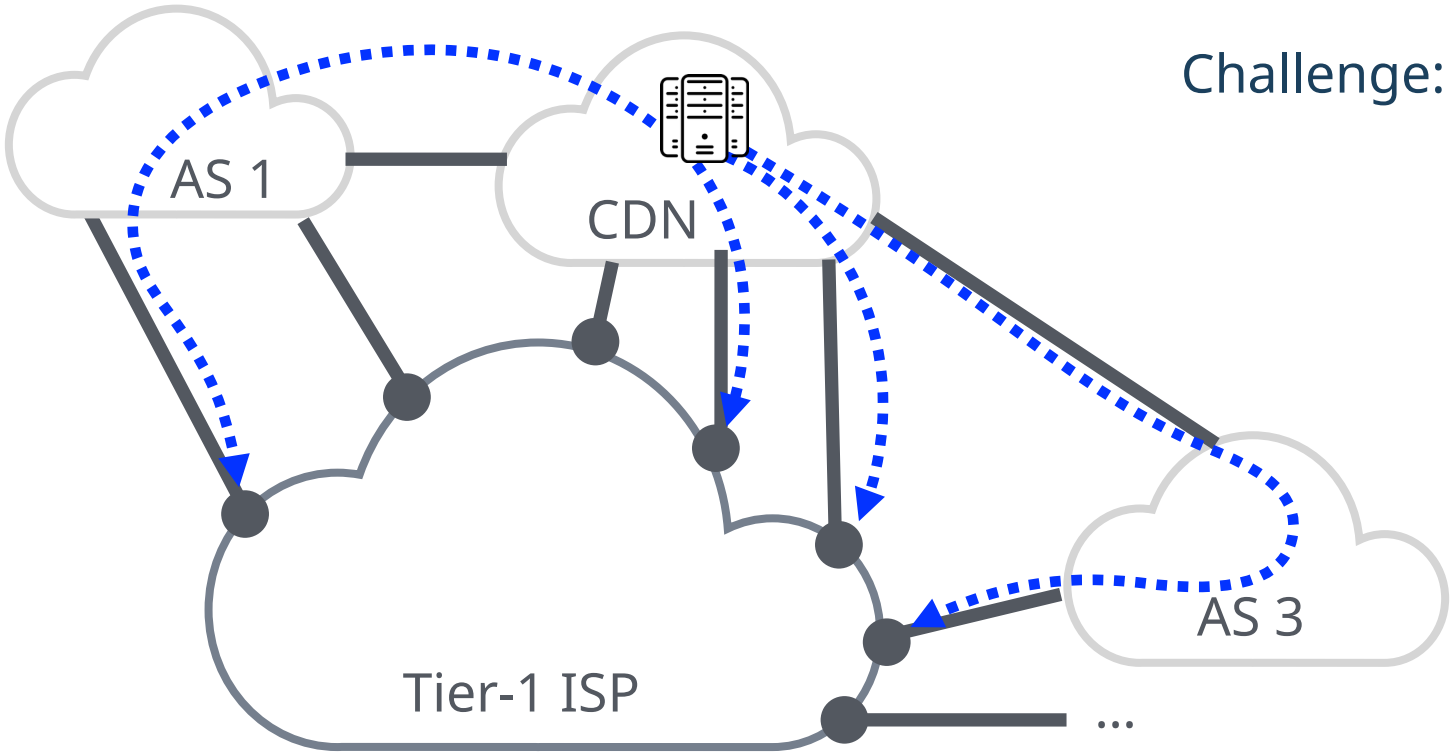
Map Subscribers of the ISP optimally to the CDN servers

Challenge: Where does the traffic enter the ISP ?

Solution: ingress point detection enables detection

Design Challenges

Scale

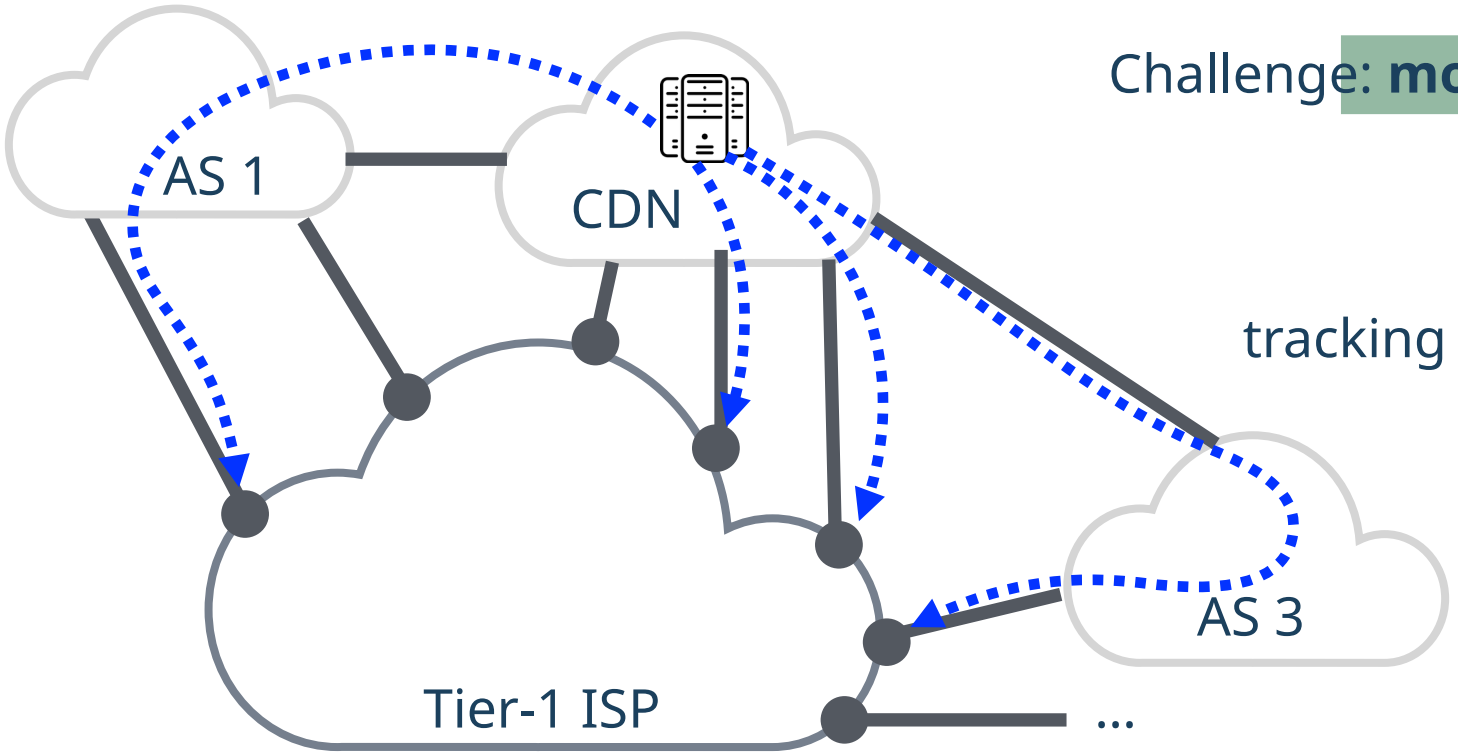


Challenge: Monitor traffic at **all Border Routers**

Observed ISP:
Thousands of Border Routers

Design Challenges

Granularity



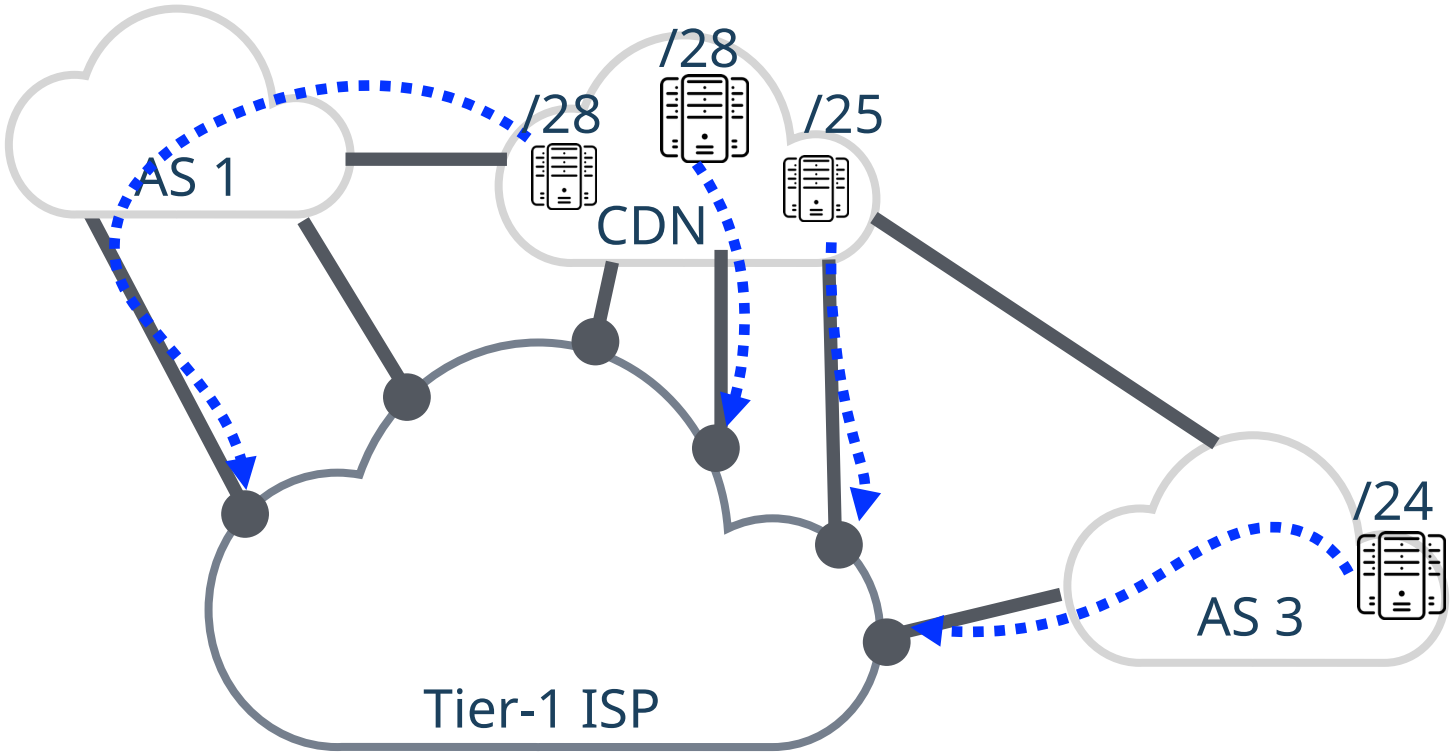
Challenge: **monitoring traffic** at all border routers

Problem:
tracking ingress points per IP does not scale

Which tracking granularity?

Design Challenges

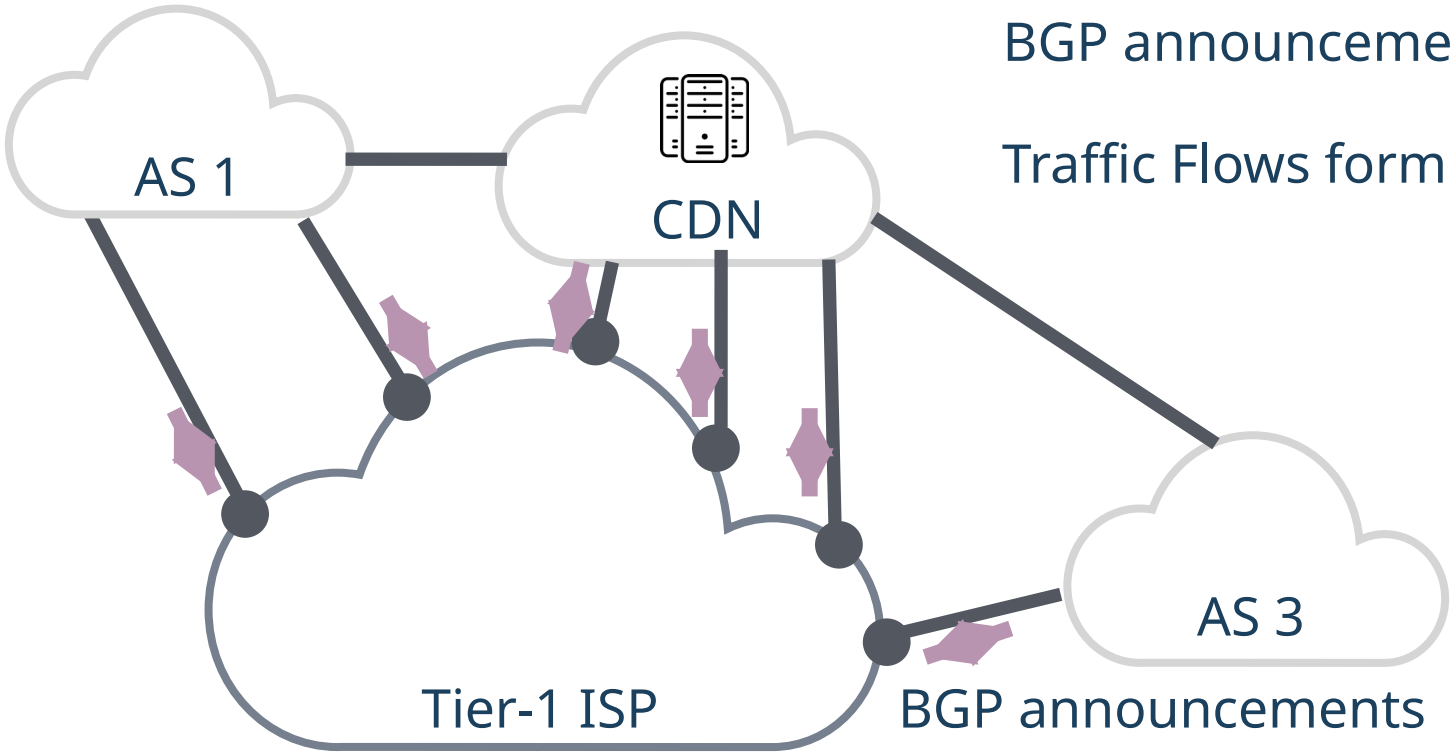
Traffic based aggregation of IP ranges



Need: tracking granularity at CDN mapping granularity

Challenge: dynamically infer aggregation prefixes from traffic

Design Aspects



BGP announcement go towards the traffic source

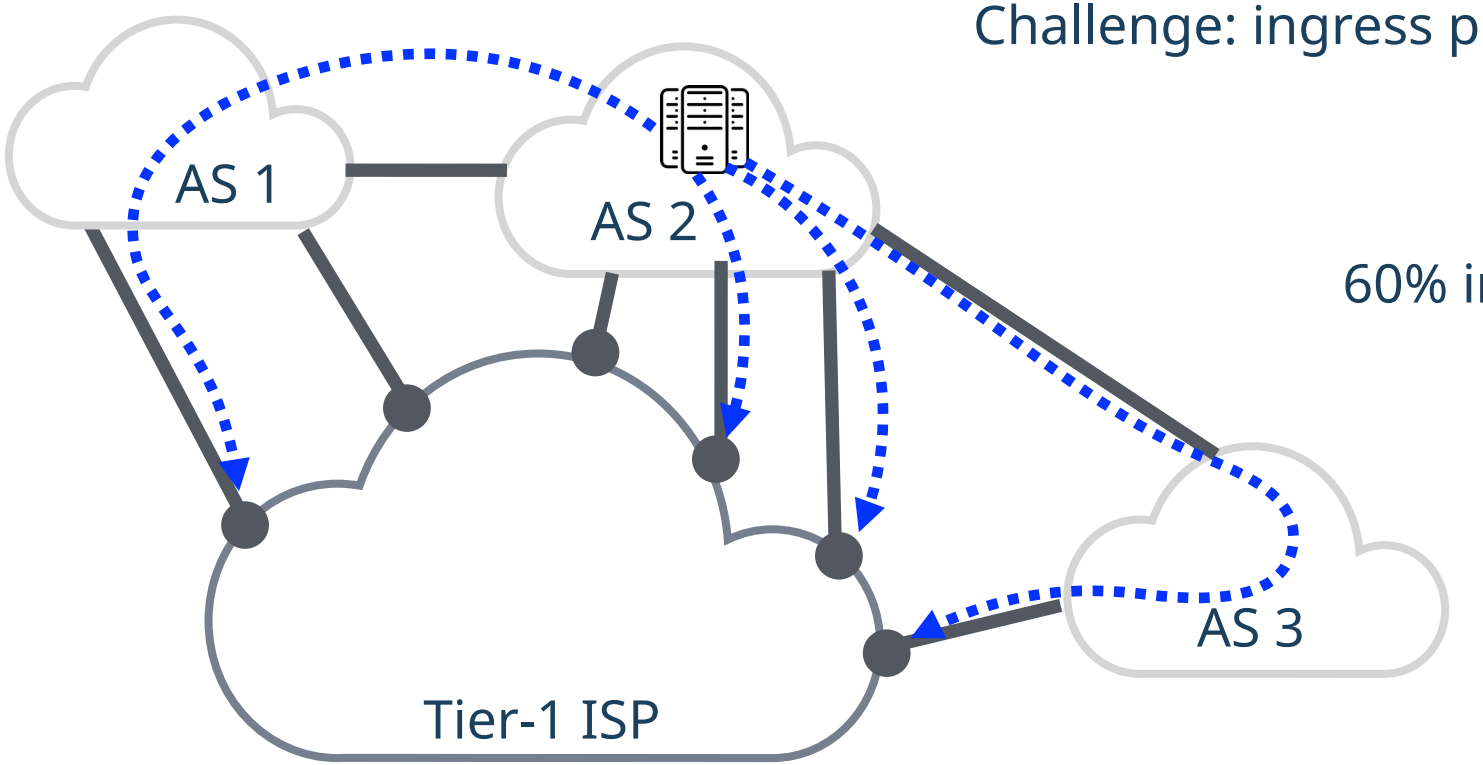
Traffic Flows form the source towards the announcement

Requirement:

- Do not use BGP Prefix sizes

Design Aspects

Continuous monitoring



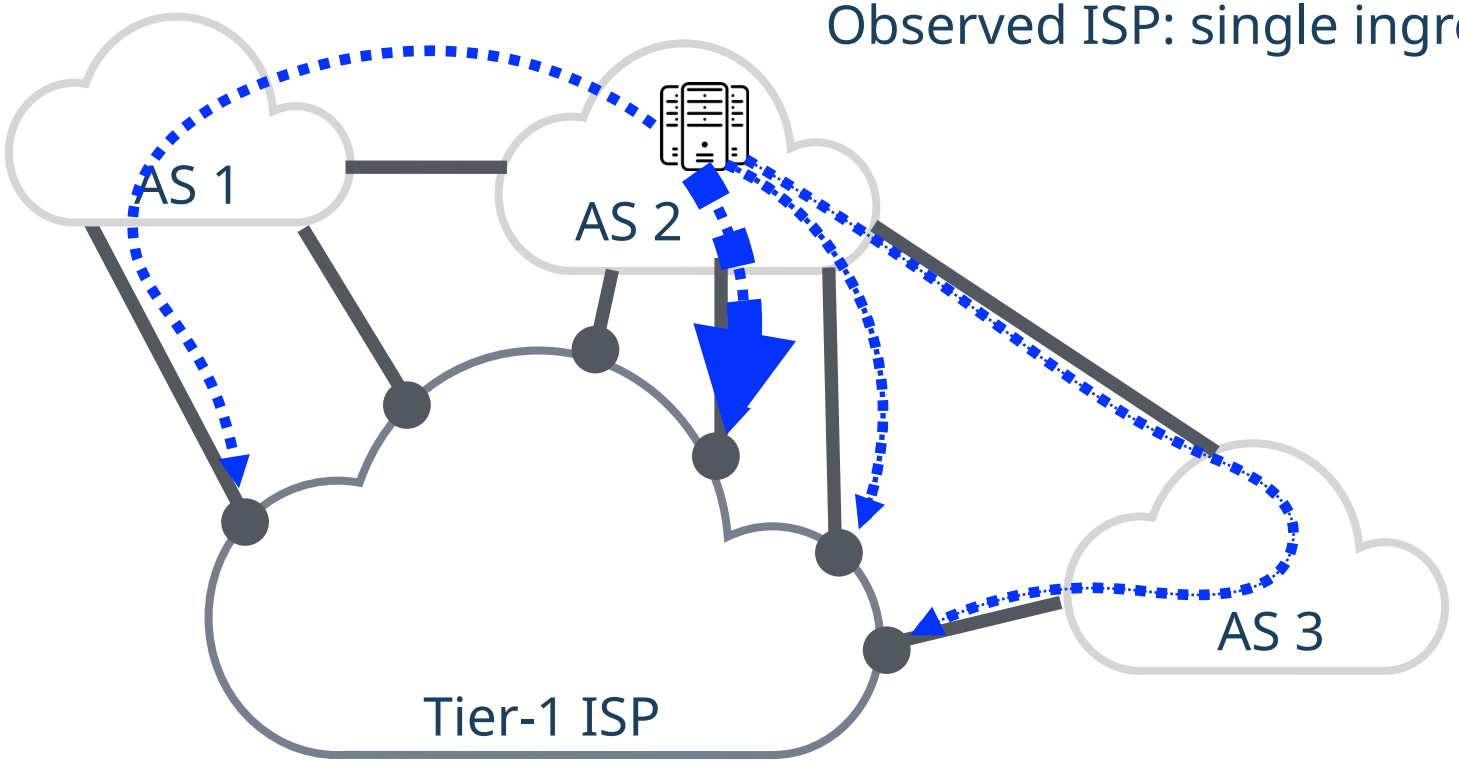
Challenge: ingress points **change often**

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

Requirement:
- detection within Minutes
- fast adaption of changes

Design Aspects

Focus on high-traffic prefixes



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

single ingress: 80% traffic

Requirement

- detect dominant ingress point
- at right prefix size

Ingress Point Detection at ISPs

Requirements:

No-Input: BGP

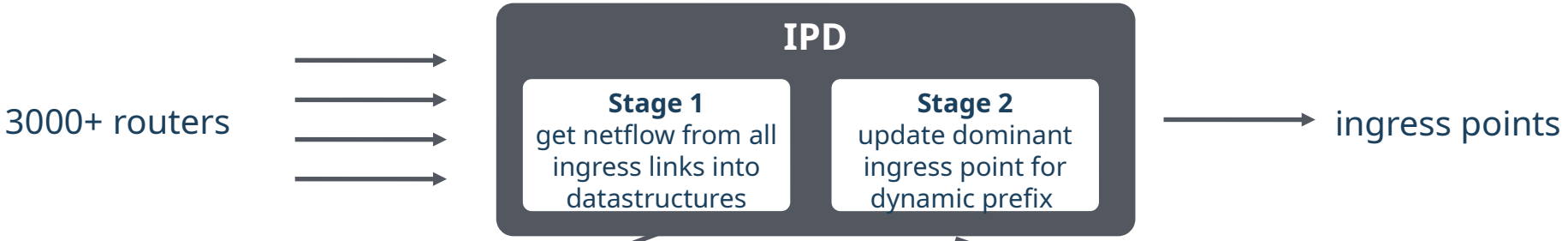
Input: Netflow
Link Classification

Scale: 1000s of routers
Continuous monitoring
Aggregate Ips into dynamic IPD ranges

Scope: All ingress points
Source address tracking

Goal: Focus on high volume prefixes
Find dominant ingress point

Ingress Point Detection Algorithm



many CPU cores

1 CPU core

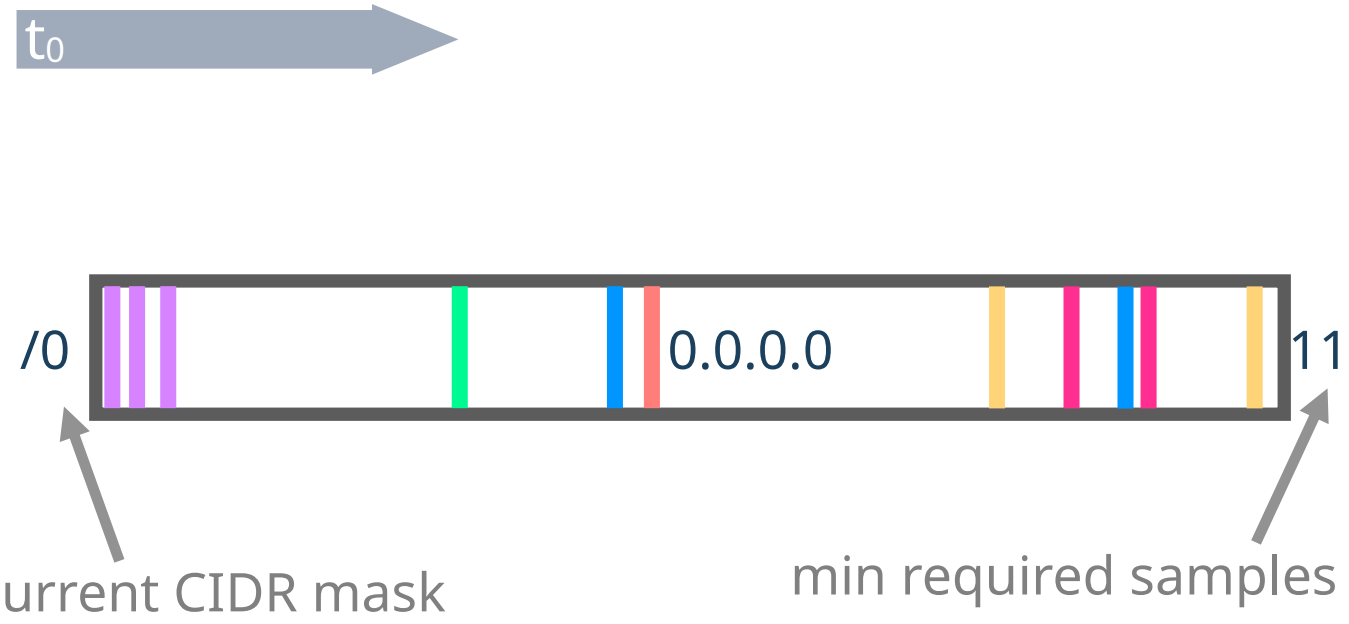
Challenge Stage 1:
massive influx of Netflow

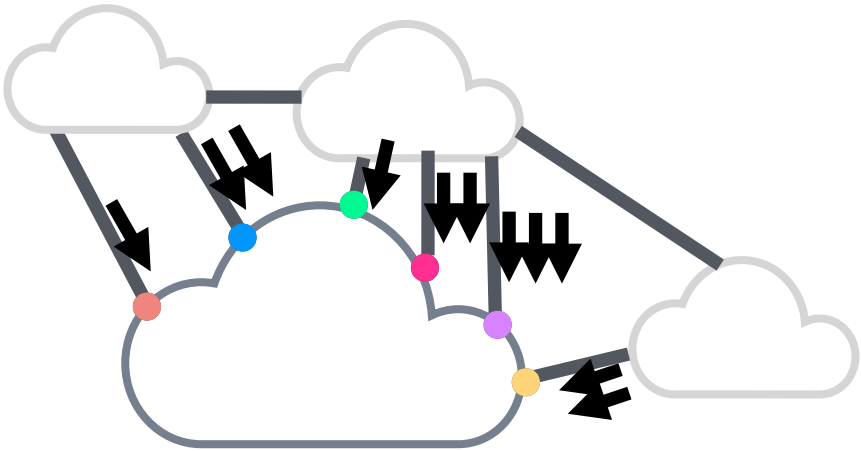
Goal: only collect and prepare incoming data

Challenge Stage 2:
classify ranges

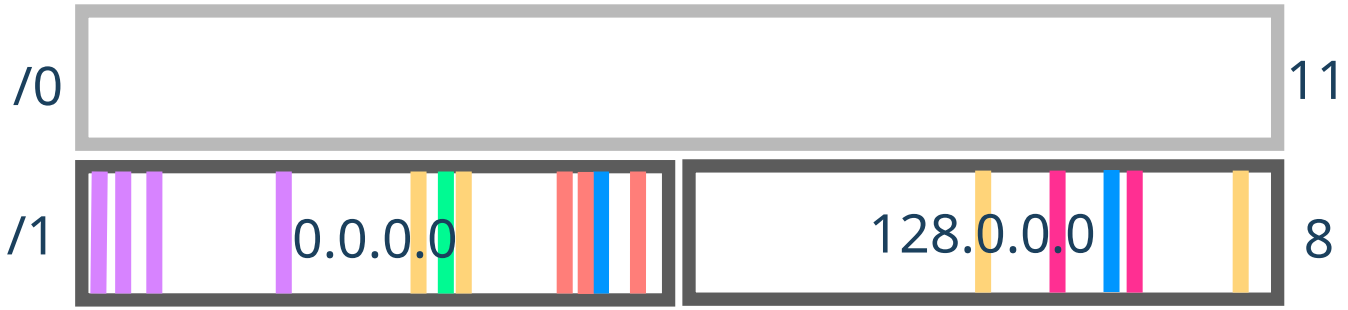
find dominant ingress point

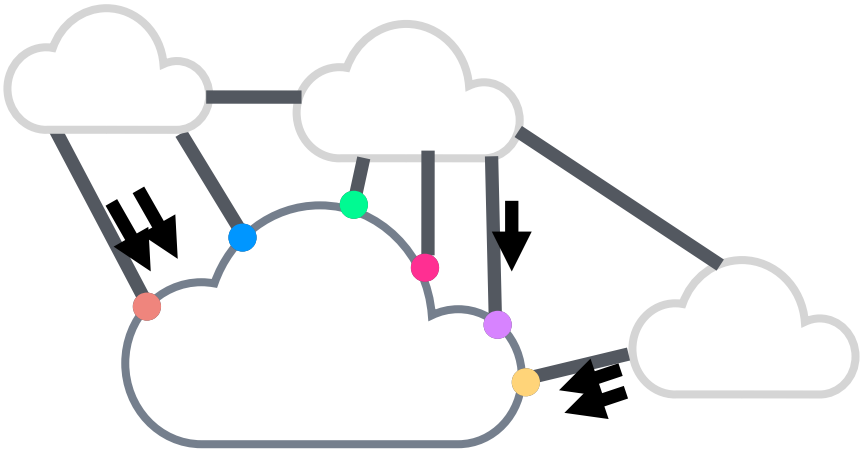
Ingress Point Detection Algorithm



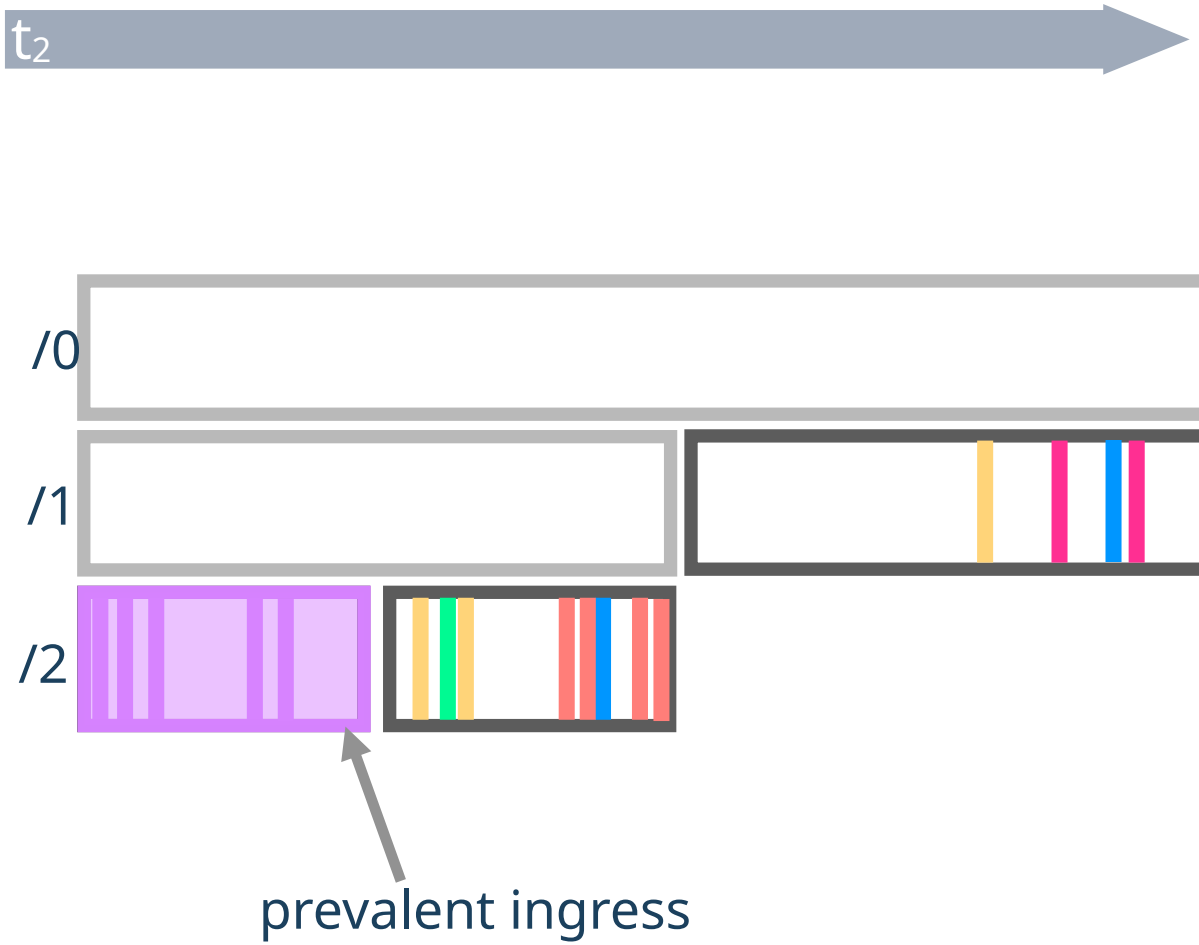


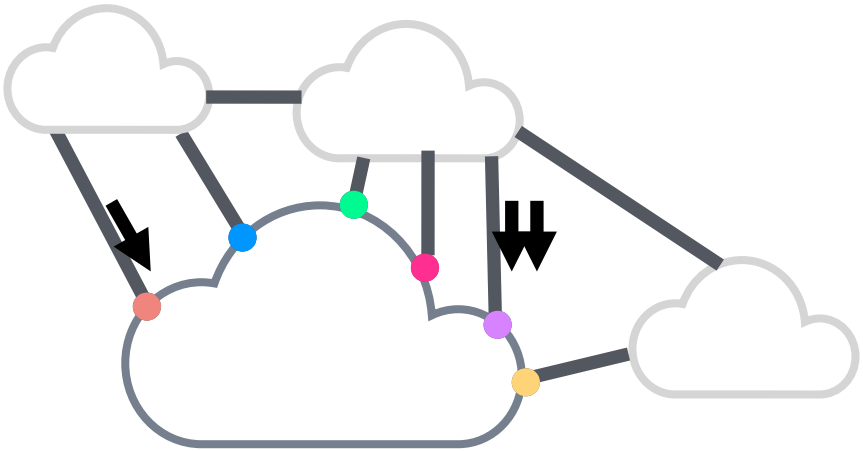
Ingress Point Detection Algorithm



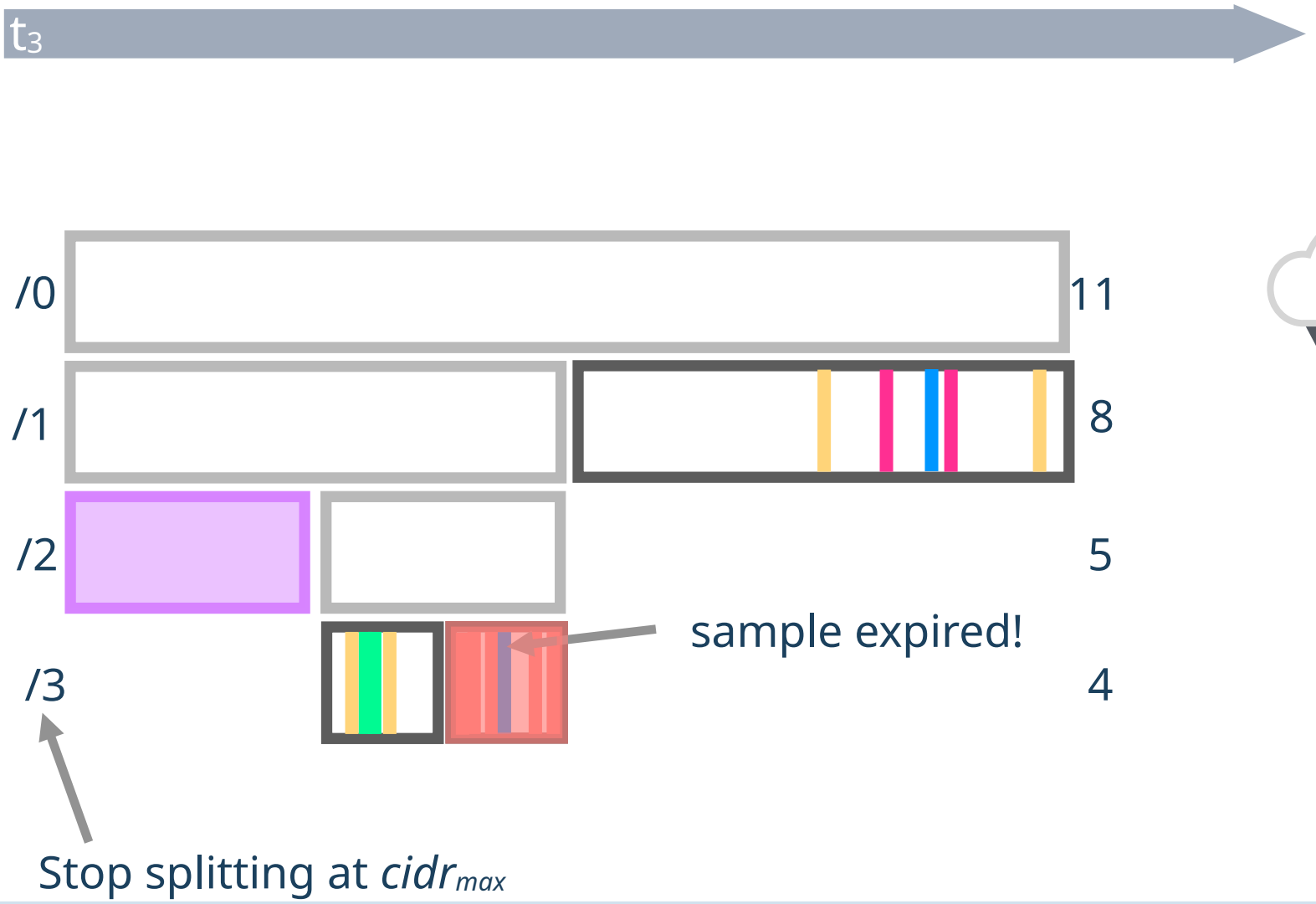


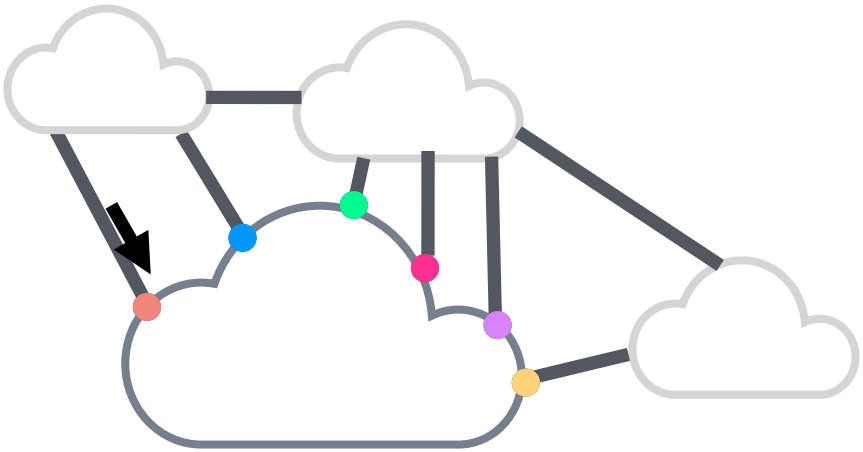
Ingress Point Detection Algorithm



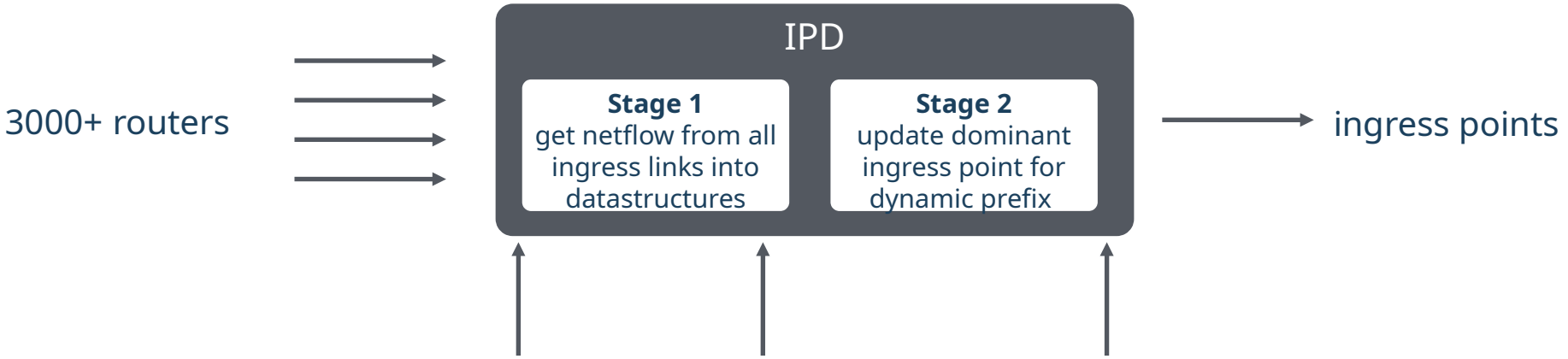


Ingress Point Detection Algorithm





Parameter Study



Example parameters:	$cidr_{max}$	error margin	time bucket length
Our production 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

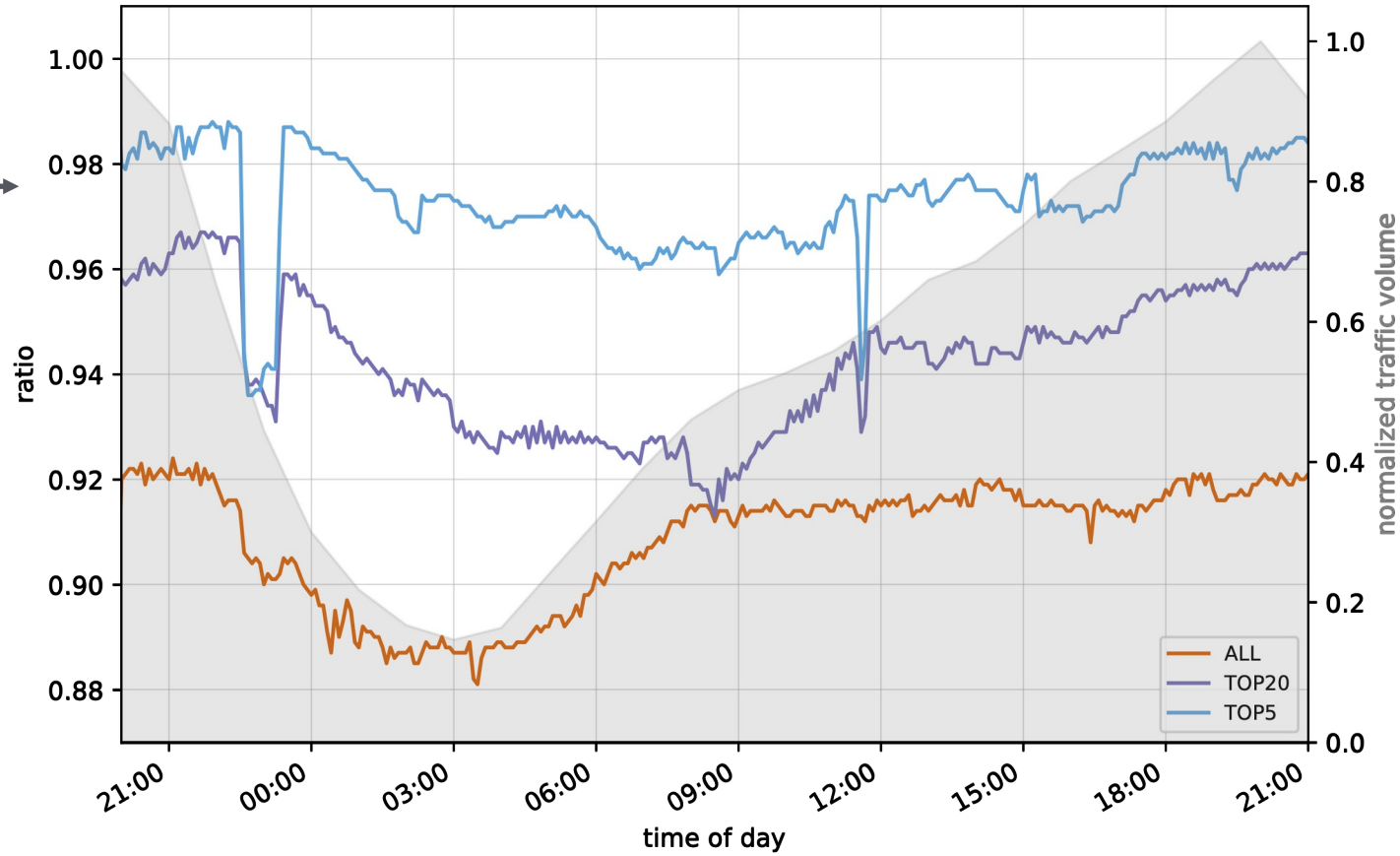
Single commodity server for an entire ISP is enough

~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



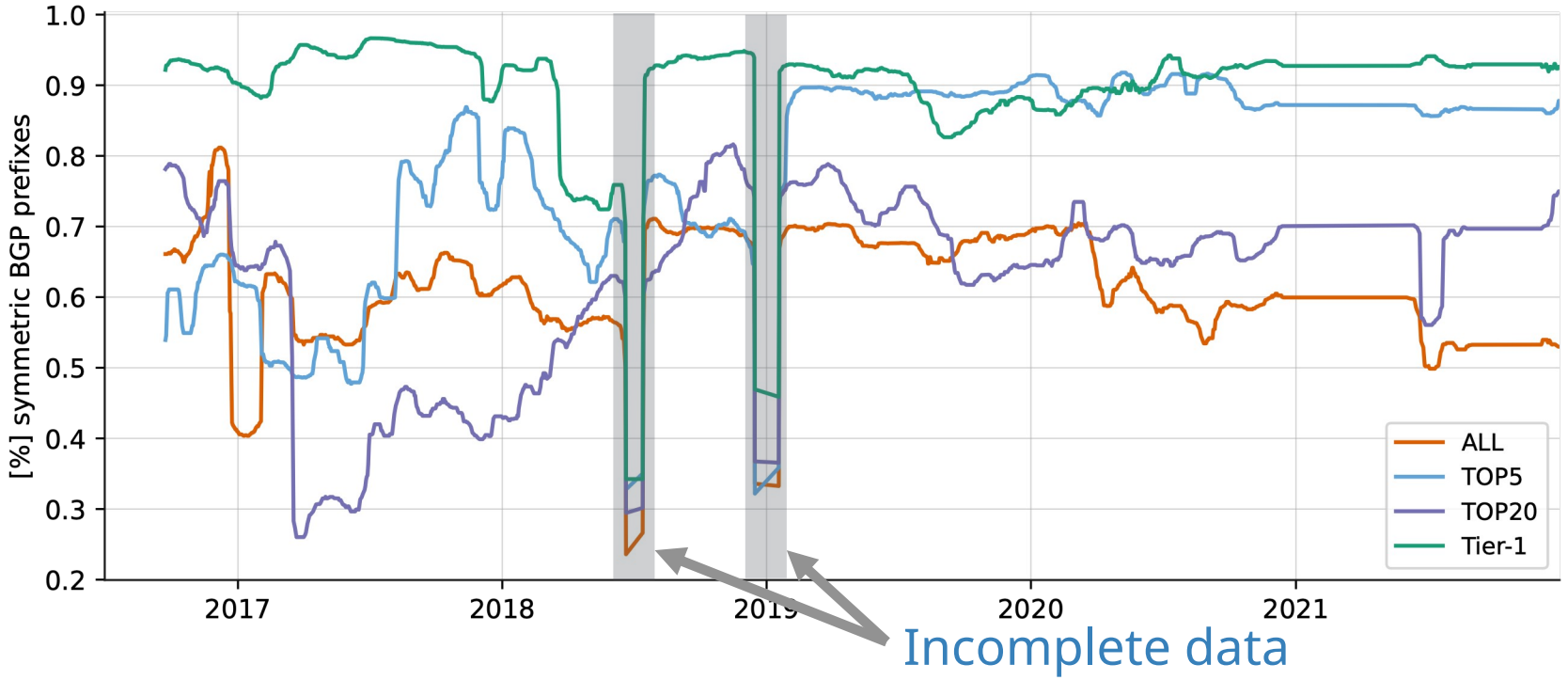
Reasons for inaccuracies:

- traffic shifts
- operational changes
- noise
- Malicious/spoofed packets
- …

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

Don’t assume path symmetry

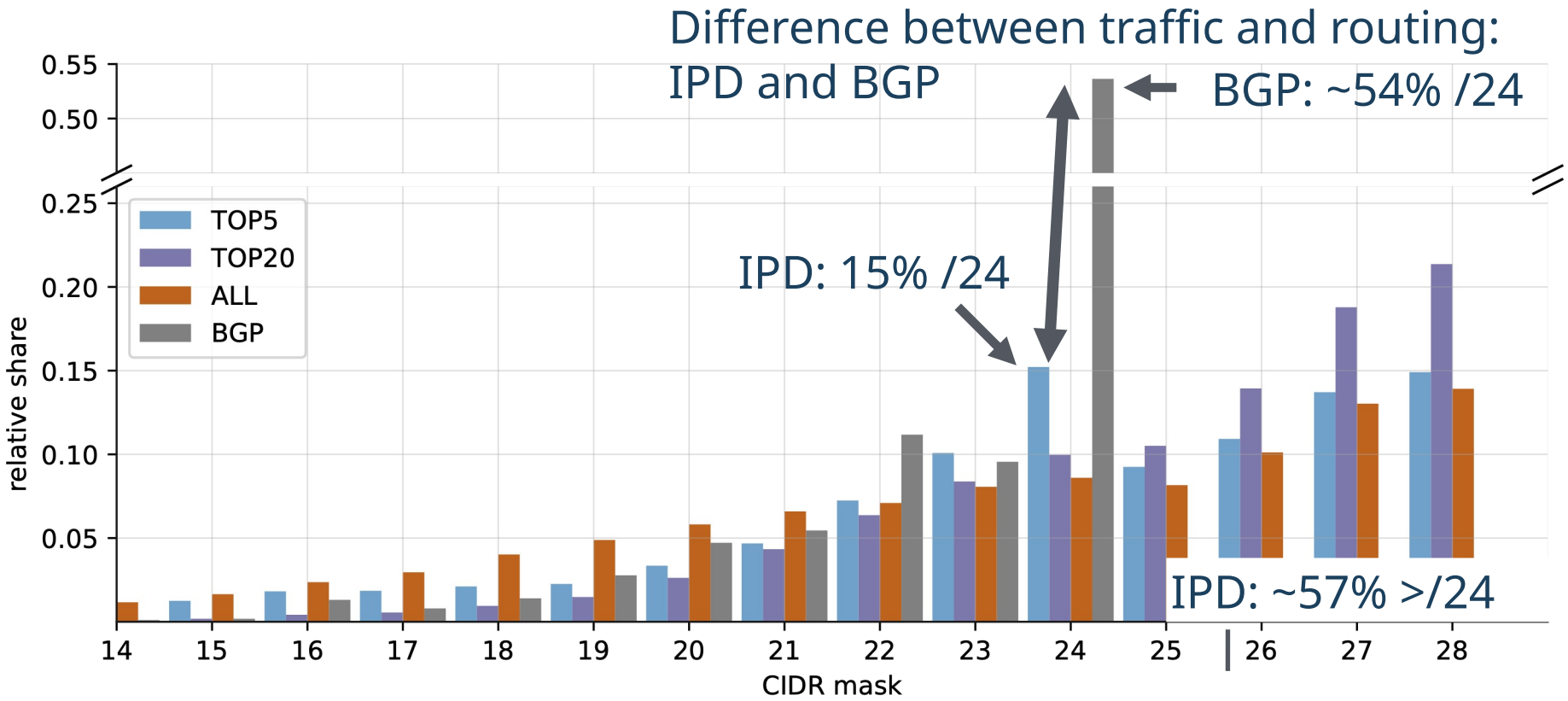
IPD enables first Tier-1 measurement study on path asymmetry



AS path have not been and still are not symmetric

BGP and IPD prefix sizes differ

IPD optimally adapts to ingress traffic dynamics



Operational Experience

IPD enables: Network troubleshooting

CDN-ISP Collaboration / joint ingress traffic engineering

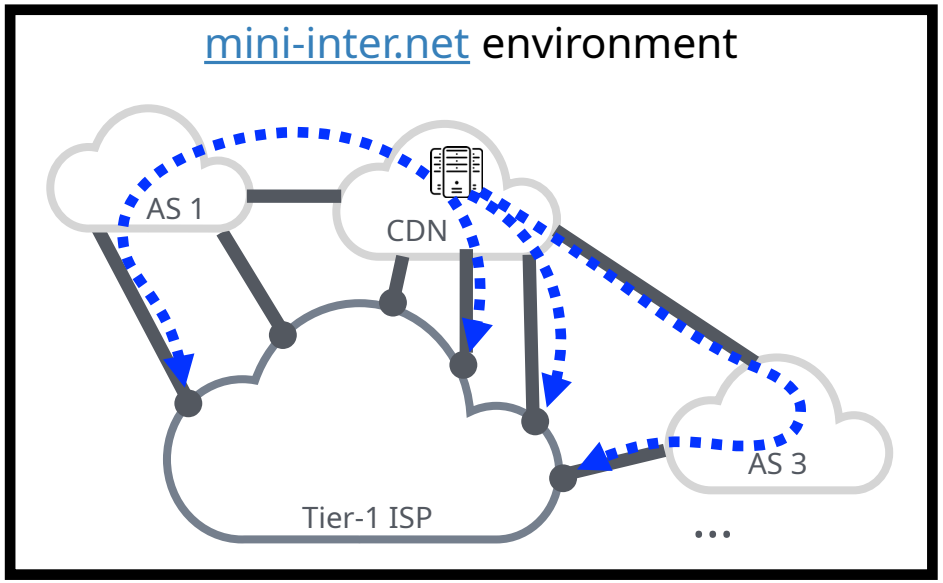
Detecting and watching ingress points in real time

Comparison of BGP to traffic behavior

IPD omits: Router-level traffic load balancing

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

Mini-IPD: Do it yourself IPD Prototype



Prototype IPD implementation:

https://github.com/smehner1/ipd

IPD in an emulated ISP scenario with CDNs:

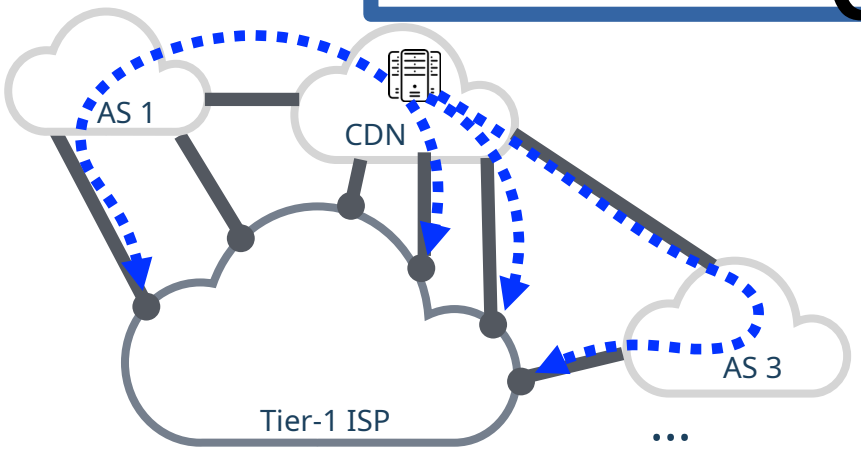
https://github.com/smehner1/mini-ipd

IPD: Ingress Point Detection at ISPs

Stefan Mehner, Helge Reelfs,

Ingmar Poese, Oliver Hohlfeld

Thank you.
Questions ?



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

Scales to a tier-1 ISP on a single server

github.com/smehner1/mini-ipd

Backup

IPD vs. TIPSY

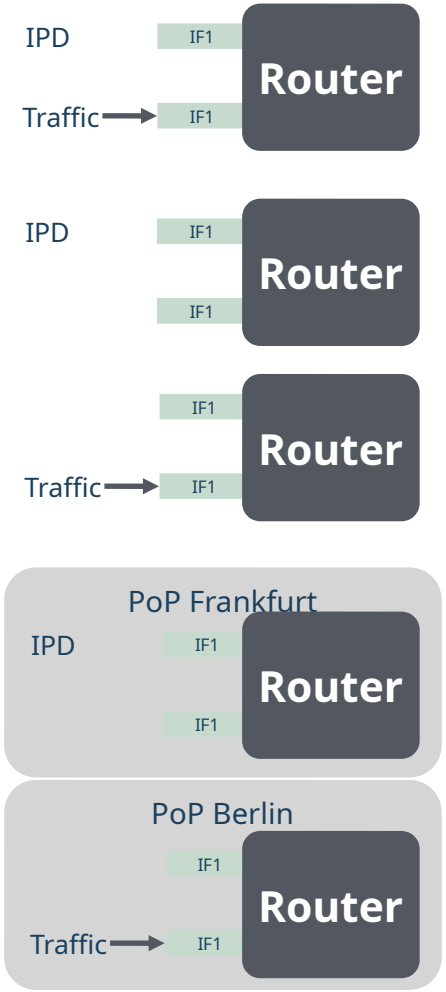
	TIPSY	IPD
Focus	Cloud Provider	ISP
Method	Statistical model of ingress traffic volumes and points Predict effect of shifting traffic by selective BGP withdrawals for prefixes observed in training period	Traffic-based partitioning of the entire IP address space
Granularity	/24	Dynamically up t o a predefined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

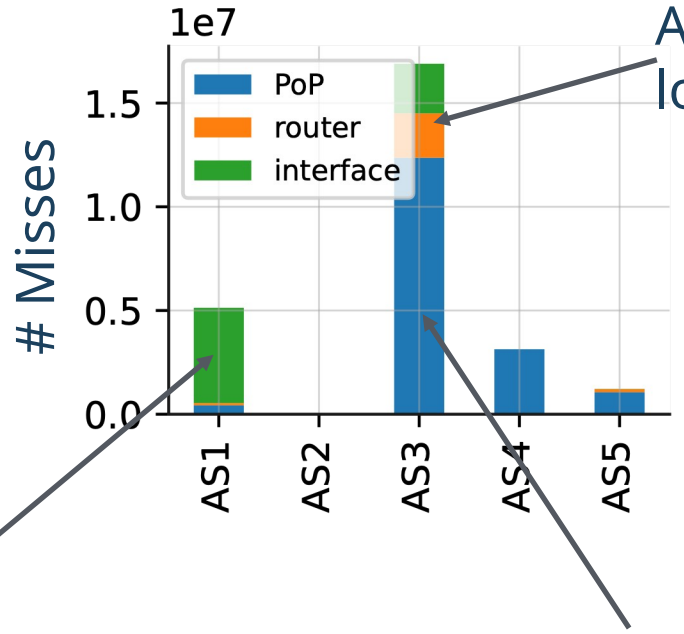
3 types of misses can/do happen

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