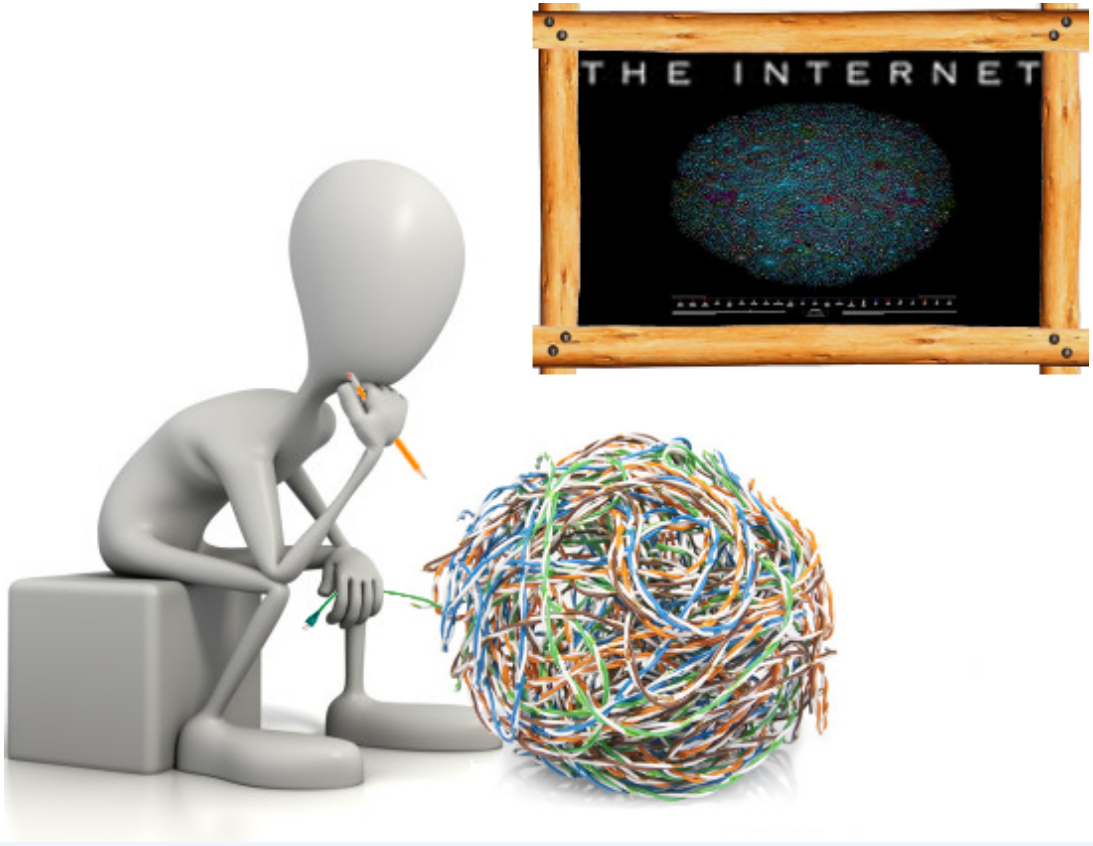


BGP, route collectors & co.

Alessandro Improta
alessandro.improta@iit.cnr.it

Luca Sani
luca.sani@iit.cnr.it

Consiglio Nazionale delle Ricerche
Istituto di Informatica e Telematica

About us

Alessandro Improta

Alessandro Improta is a researcher at the Institute for Informatics and Telematics of the National Research Council of Italy in Pisa since February 2013, where he is part of the Internet Measurements & Design research group. He received his B.Sc. and M.Sc. in Computer Engineering from the University of Pisa, Italy, in 2006 and 2009 respectively and his Ph.D. degree in in Information Engineering from the University of Pisa, in 2013. In 2013 he co-founded the Isolario project at IIT-CNR.
http://www.iit.cnr.it/alessandro.improta

Luca Sani

Luca Sani is a researcher at the Institute for Informatics and Telematics of the National Research Council of Italy in Pisa since March 2014, where he is part of the Internet Measurements & Design research group. He received his B.Sc. and M.Sc. in Computer Engineering from the University of Pisa, respectively in 2008 and 2010 and his Ph.D. degree in Computer Science and Engineering from the IMT Institute for Advanced Studies, Lucca in 2014. In 2013 he co-founded the Isolario project at IIT-CNR.
http://www.iit.cnr.it/luca.sani

Outline

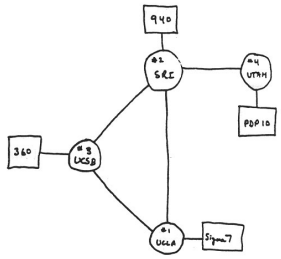
1. Introduction
2. The BGP protocol
3. BGP security issues
4. The Isolario project: a do-ut-des approach to tackle incompleteness
5. ICE: an Interactive Collector Engine
6. BGP Scanner: Isolario MRT-BGP data reader

"It is a capital mistake to theorize before you have all the evidence. It biases the judgment"
(sir A.C. Doyle)

Let's start

First... a little bit of history

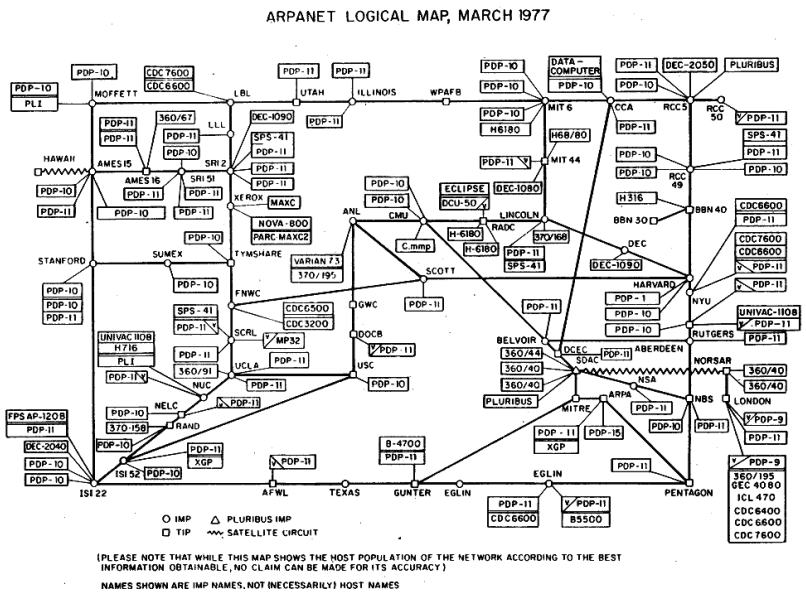
- 1969 - ARPANET
- 1985 - NSFNET
- 1995 - Commercial Internet



THE ARPA NETWORK

DEC 1969

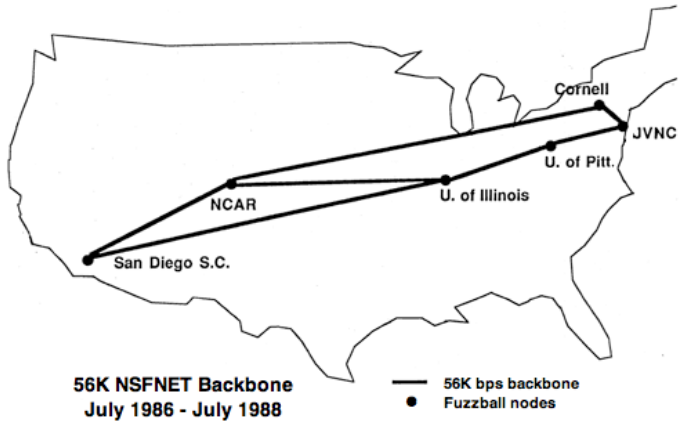
4 NODES



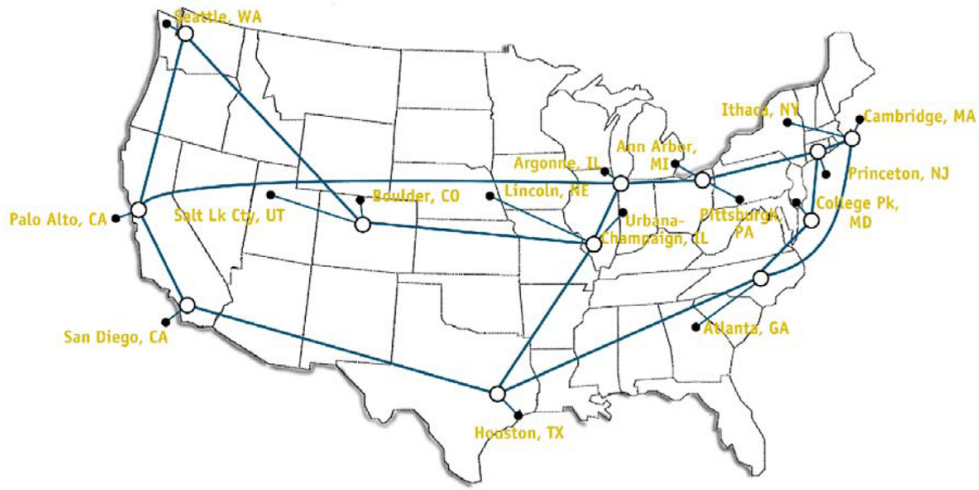
ARPANET was one of the first network to implement TCP/IP (1983)

First... a little bit of history

- 1969 - ARPANET
- 1985 - NSFNET
- 1995 - Commercial Internet



NSFNET T3 Network 1992



NSFNET use from for-profit organizations was acceptable when it was in support of open research and education

What about today?

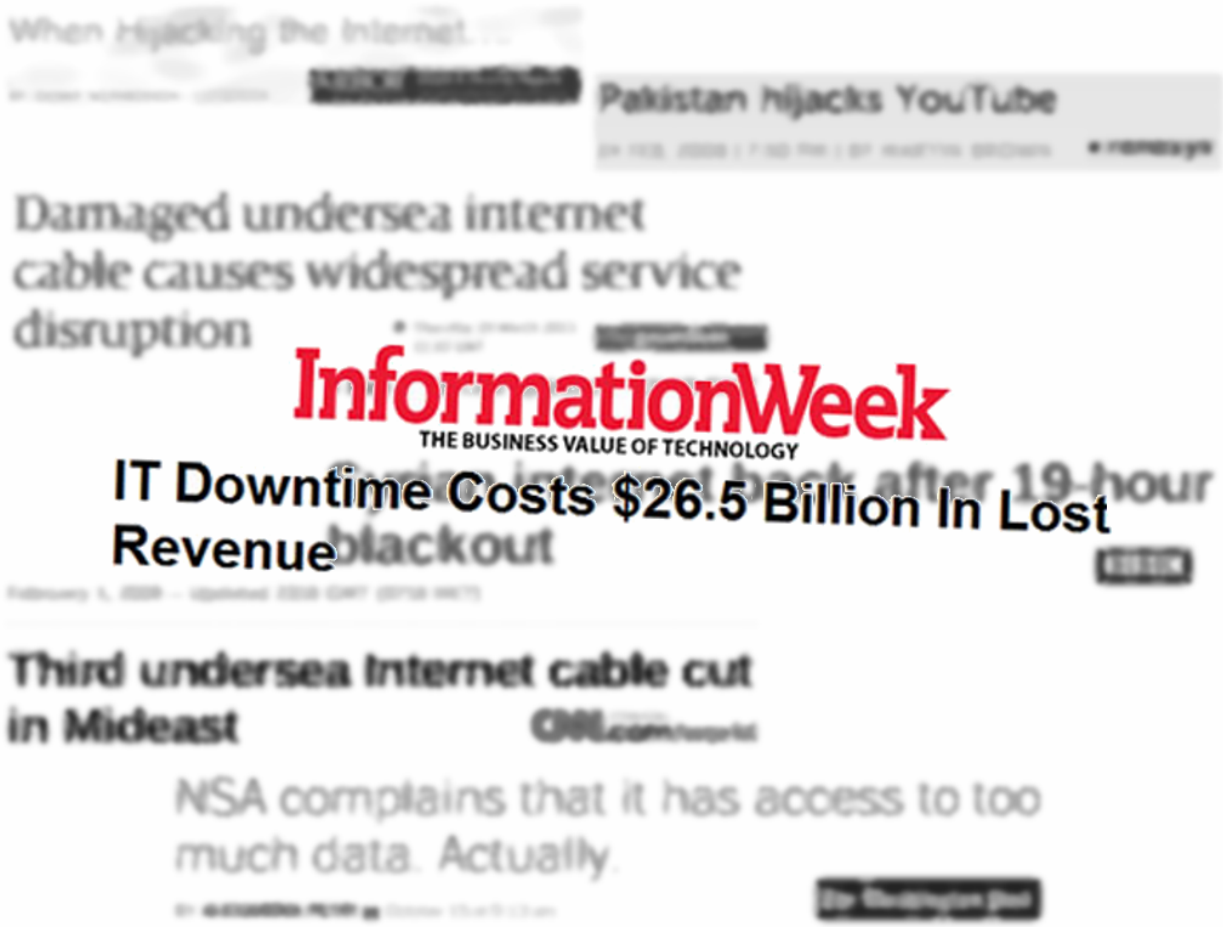
- 1969 - ARPANET
- 1985 - NSFNET
- 1995 - Commercial Internet

From then, its real structure became hidden, as well as its potential structural weaknesses

Why is it important to reveal the Internet structure?

- **To understand how packets are routed in the Internet**
  - Identify routes involving non-national ISPs
  - Identify the importance of each AS in the ecosystem
  - Understand the effects of catastrophic events (or malicious attacks)
- **To create economy-based models of the global Internet growth**
  - Study the effectiveness of p2p connections
  - Build more realistic topology generators to simulate the Internet
- **To properly select peers and diversify upstream providers based on their connectivity**
  - Increase network robustness
  - Select data centers for server replicas
  - ...

Why is it important to reveal the Internet structure?



... plan an optimal inter-domain network configuration to maintain an **acceptable level of service** in case of malicious or unintentional faults

The Internet AS level ecosystem

"An AS is a connected group of one or more IP prefixes run by one or more network operators which has a **single** *and* **clearly defined** *routing policy"*. [RFC 1930]

Each AS is identified by a unique number (AS number - ASN)

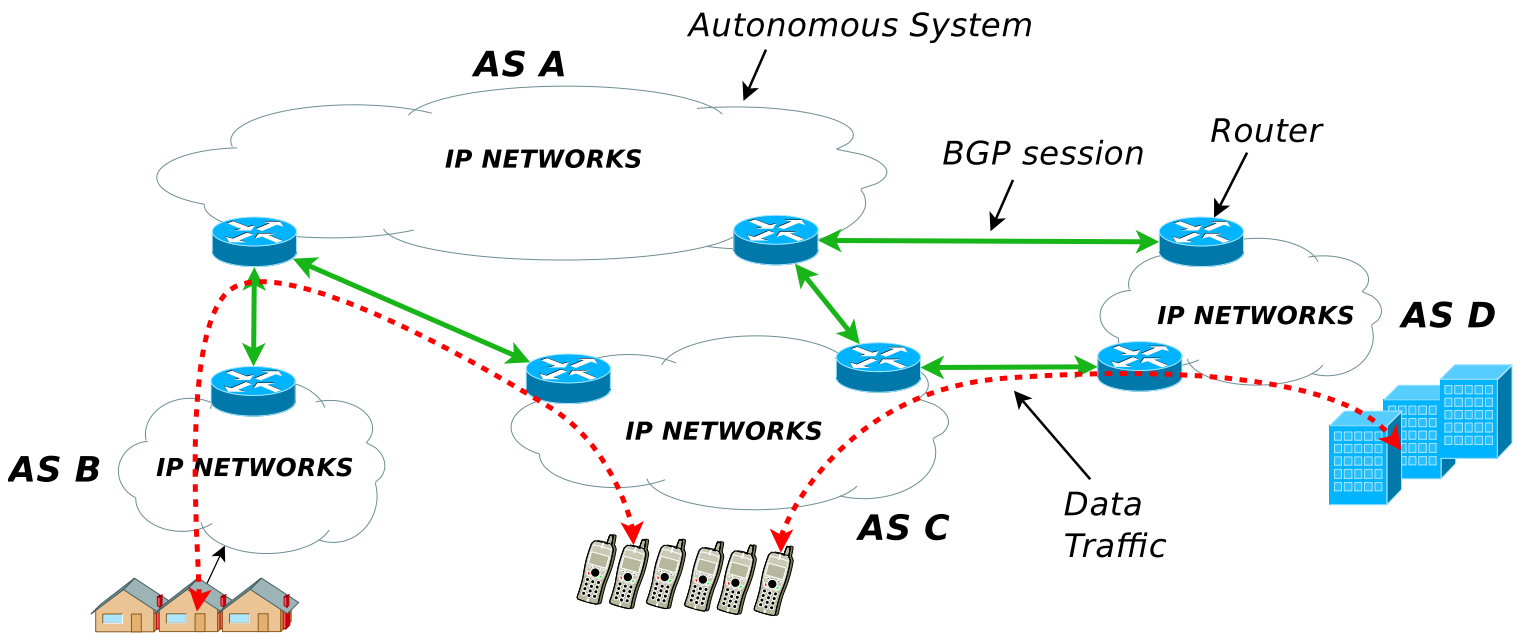
Example of ASes (about 60,000 up to date)

- **AS 3269** Telecom Italia
- **AS 8978** The Holy See - Vatican City State - Secretariat of State Department of Telecommunications
- **AS 12145** Colorado State University
- **AS 15169** Google
- **AS 21115** Nestlé Italia
- **AS 38474** Australian Antarctic Division Federal Government Administration and Scientific Research into Antarctica and the Southern Ocean
- **AS 54115** Facebook

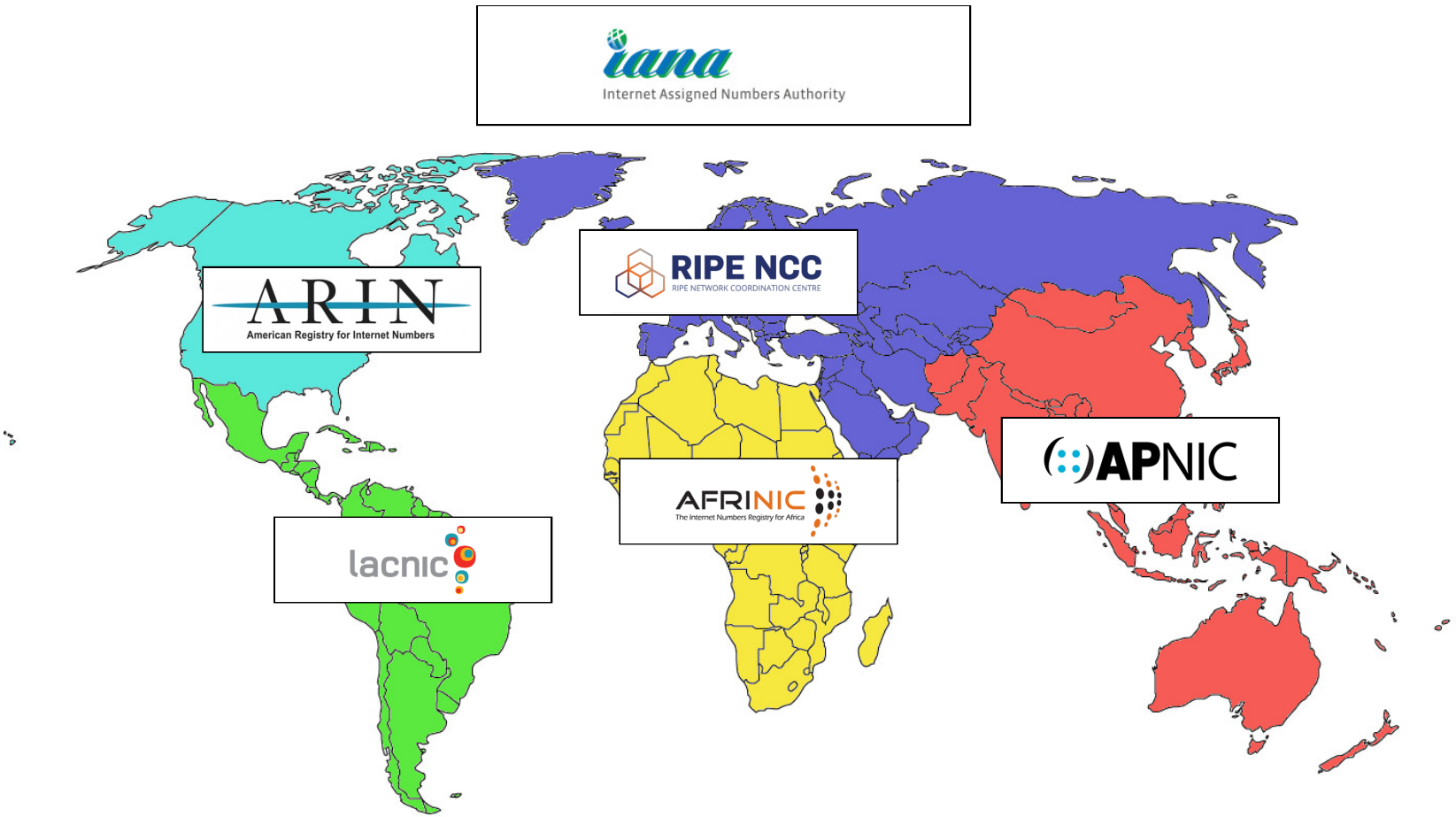
The AS-level abstraction

AS-level

- No matter about what happens inside each AS
- Inter-AS (inter-domain) routing
- Traffic crosses routes built thanks to the **Border Gateway Protocol** (BGP)

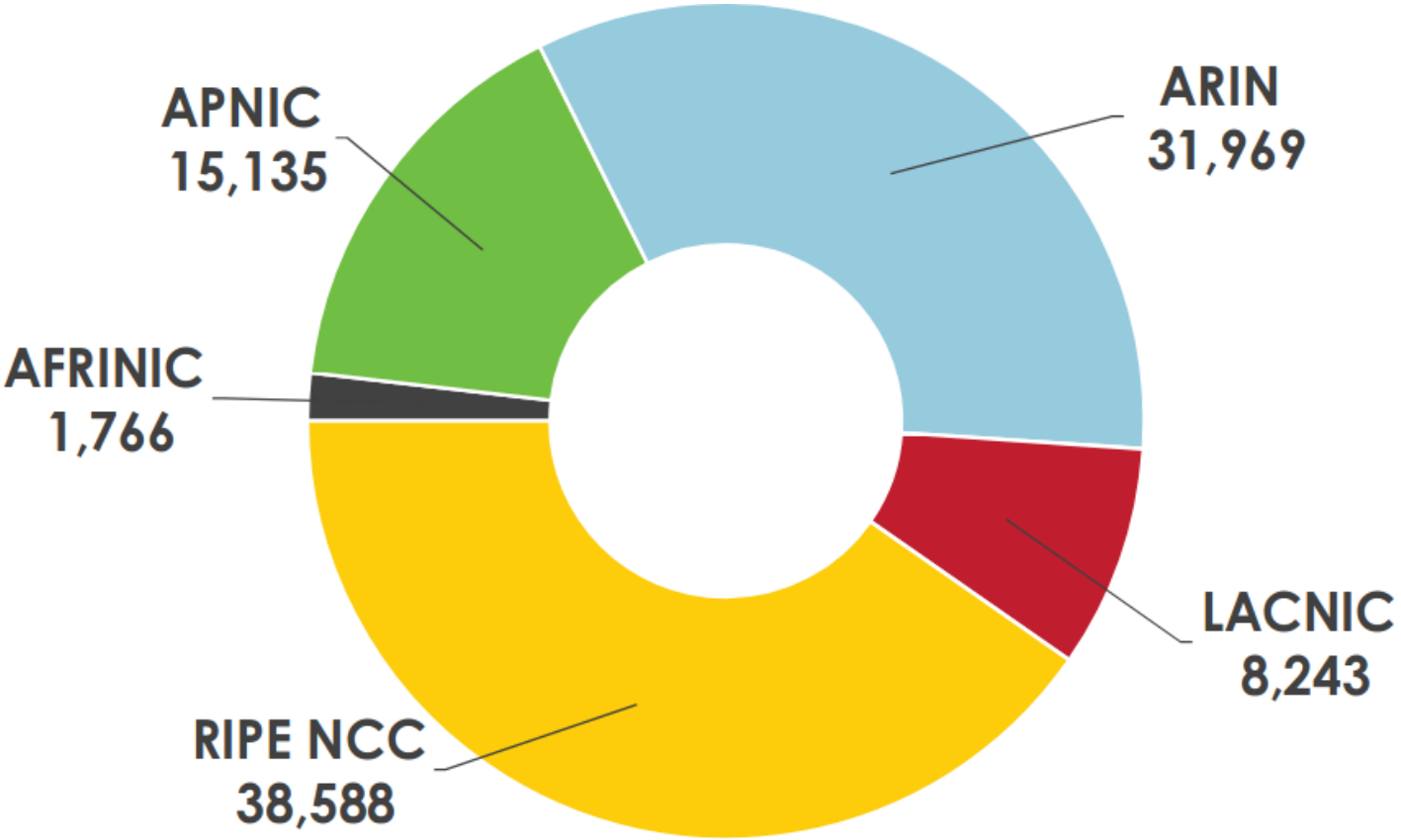


The Regional Internet Registries (RIR)



1. The Internet Assigned Numbers Authority (IANA) oversees global IP address allocation and AS number allocation
2. IANA delegates Internet resources to the five Regional Internet Registries (RIR)
3. RIR follow their regional policies to delegate resources to Local Internet Registries (LIR)
4. LIR (e.g. ISPs, enterprises, academic institutions) assigns parts of the IP block to its own customers

ASN distribution



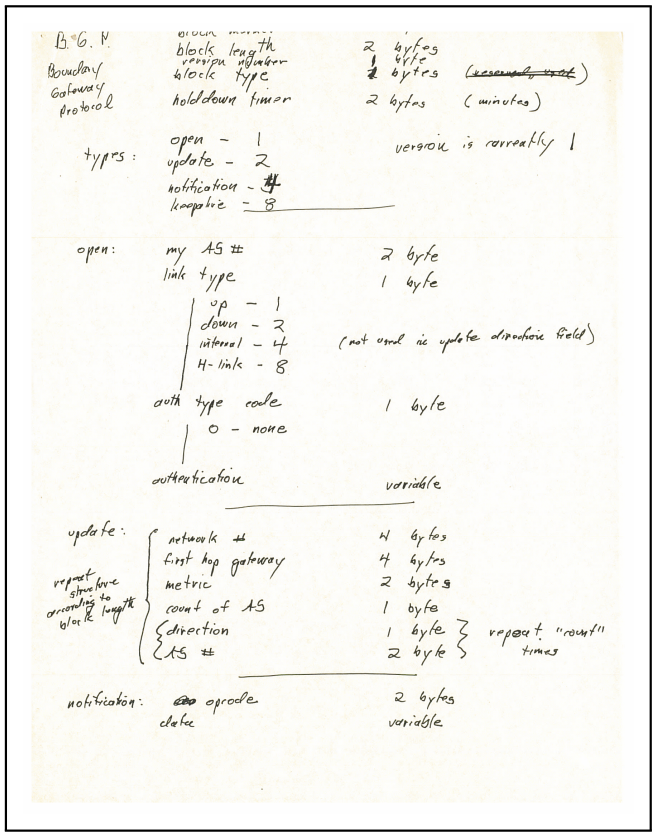
As of March 2018
https://www.nro.net/statistics

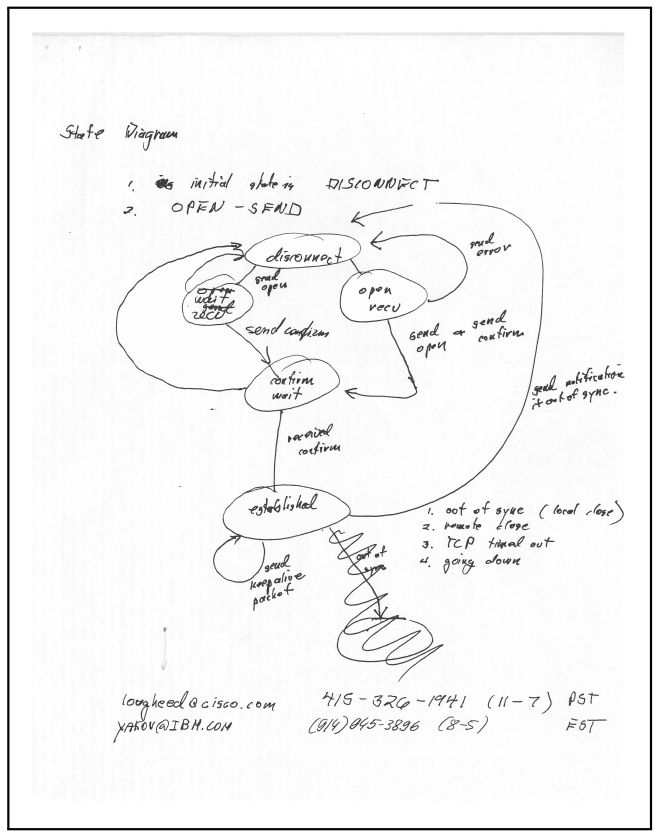
The BGP protocol

The Border Gateway Protocol (BGP)

The two-napkin protocol

- Conceived by Kirk Lougheed (Cisco) and Yakov Rekhter (IBM) on two napkins during lunch at a conference (IETF meeting)
- Intended to be a quick fix: it was 1989

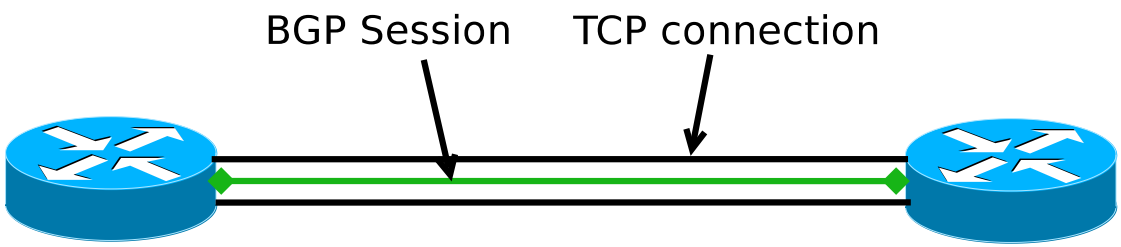




The BGP protocol (RFC 4271)

BGP session

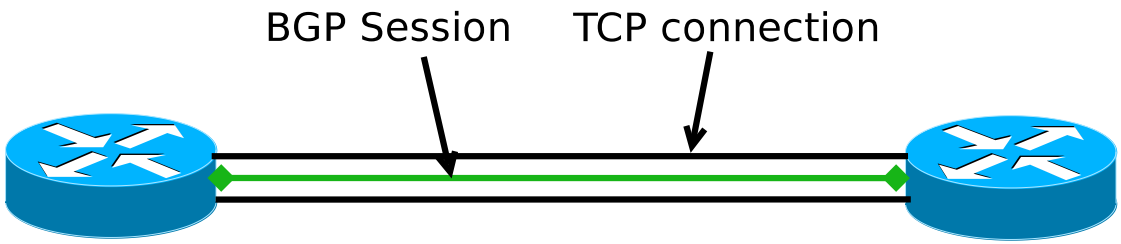
- ASes agree to exchange routing information by establishing BGP sessions each other
- The session runs over TCP (standard port 179), that is, BGP messages are encapsulated into TCP segments



The BGP protocol (RFC 4271)

BGP session

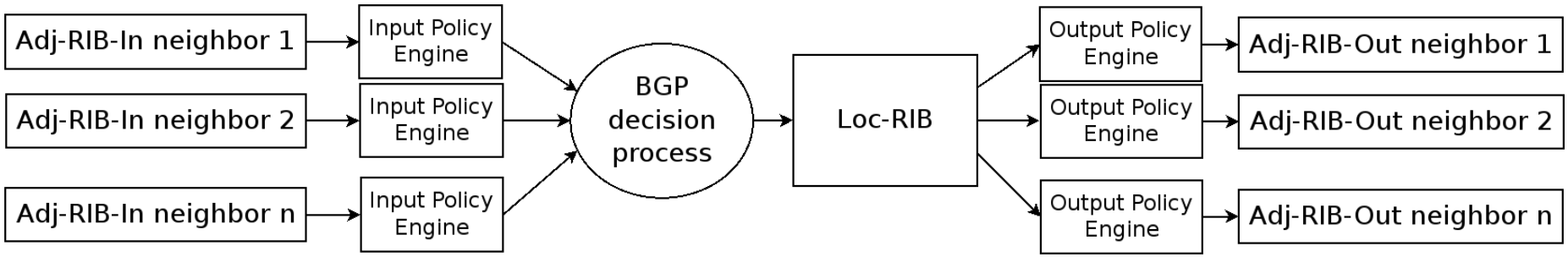
- ASes agree to exchange routing information by establishing BGP sessions each other
- The session runs over TCP (standard port 179), that is, BGP messages are encapsulated into TCP segments



BGP messages

- `BGP_OPEN`: session handshake
- `BGP_KEEPALIVE`: session maintainement
- `BGP_NOTIFICATION`: error reporting
- `BGP_UPDATE`: routing information

The BGP routing process (simplified)

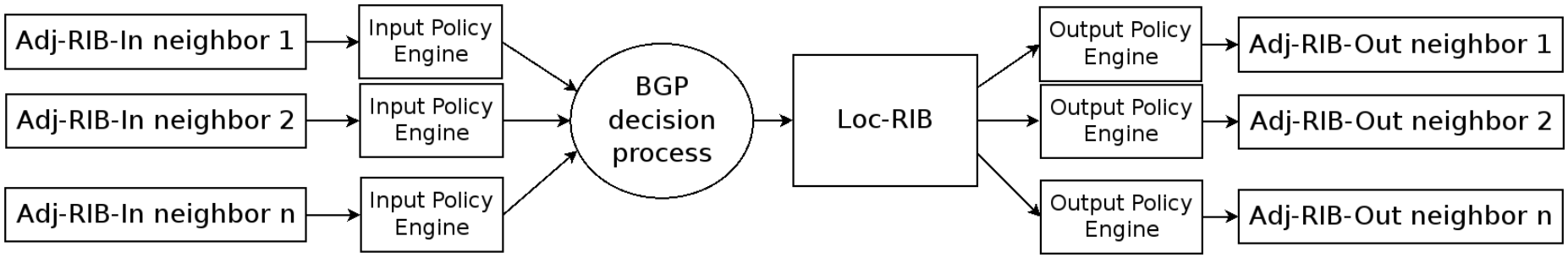


BGP routing process

The BGP routing table consists of three distinct parts:

- Adj-Routing Information Base (RIB)-In
- Loc-RIB
- Adj-RIB-Out

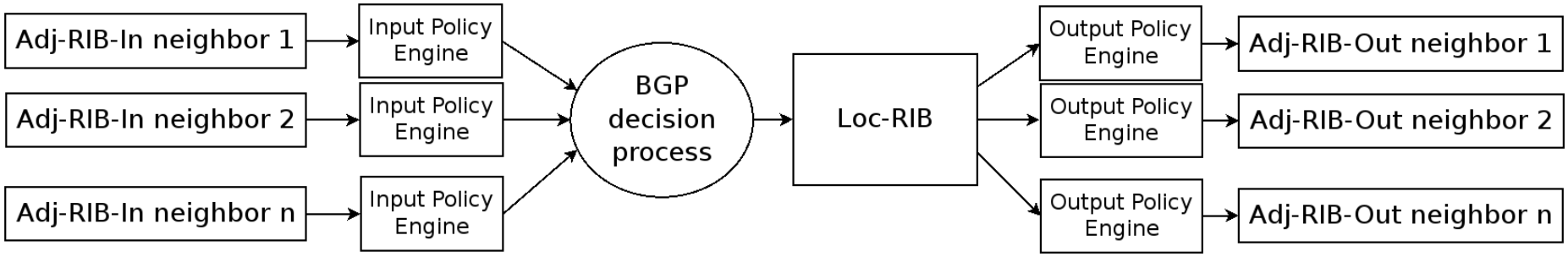
The BGP routing process (simplified)



Adj-RIB-In

- Logically associated to each neighbor
- Stores routes learned from the neighbor via `BGP_UPDATE` messages
- Its content can (and should) be filtered

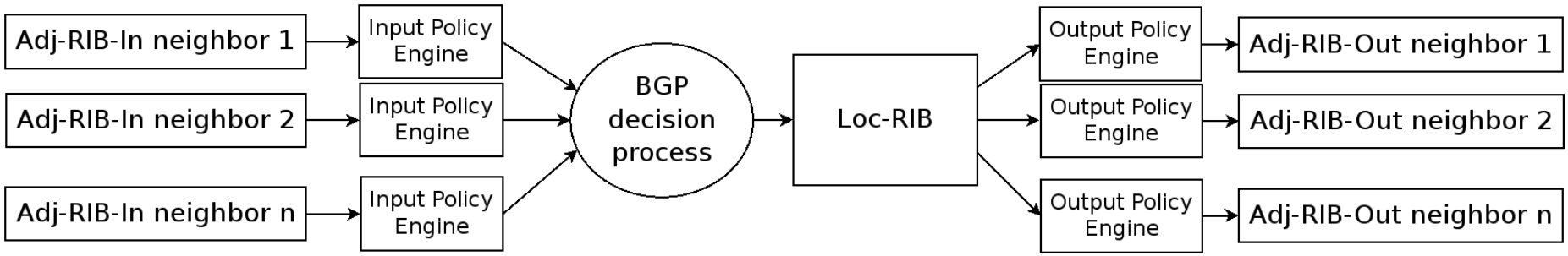
The BGP routing process (simplified)



Loc-RIB

- Contains only the best routes selected via BGP decision process
- Used for presenting routes to the IP routing table

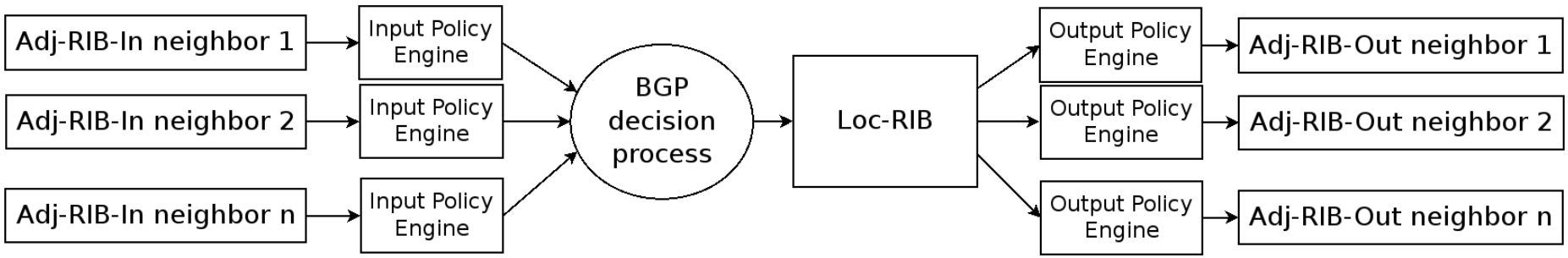
The BGP routing process (simplified)



Adj-RIB-Out

- Logically associated to each neighbor
- Stores routes chosen to be advertised to the peer

The BGP routing process (simplified)



Summarizing...

1. A BGP speaker receives routes via neighbors via `BGP_UPDATE` messages
2. Depending on filters, some of these routes will make it into the Loc-RIB
3. Routes are manipulated and advertised via `BGP_UPDATE` messages to neighbors

The BGP routing process (simplified)

Validity

- Packet must be not malformed
  - Invalid attributes
  - Missing mandatory attributes
  - Sum of attribute lengths exceed packet size
- No looped AS paths

Reachability

- Route must be valid and reachable
  - Next hop must be reachable, or
  - Route reachable through itself

Specificity

- Prefer routes more specific
  - Prefer 10.0.0.0/24 over 10.0.0.0/8

BGP route

Route

A unit of information that pairs a set of destinations with the attributes of a path to those destinations [RFC 4271]

What does that mean?

A route is a way toward one or more destinations (like in real life). The attributes indicate characteristics of the route itself:

- the sequence ASes that traverses to reach the destination `[AS_PATH]`
- form where the route was learned `[ORIGIN]`
- how much that route is good `[LOCAL_PREF]`
- ...

```
>> show feeder exact 90.147.84.10 192.65.131.0/24

192.65.131.0/24 90.147.84.10|137 2598|i|137:1000 137:3025|2019-02-28 17:12:15
Total number of prefixes: 1
>>
```

Commercial agreements

BGP is an extremely flexible protocol that allows to implement both technical and commercial constraints

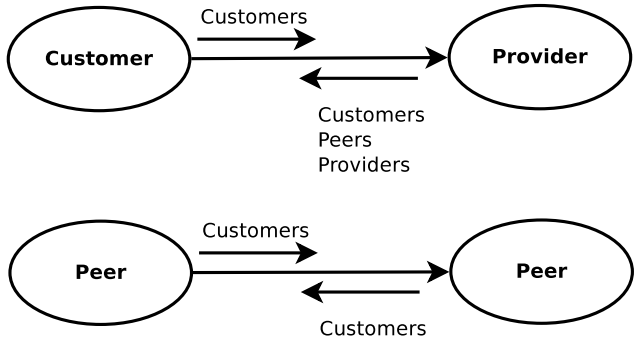
BGP connectivity does not imply IP reachability

Two ASes that establish a BGP session **agree** on which routes they exchange each other each other

Typical agreements (economic relationships)

- Provider-to-customer
- Peer-to-peer

Commercial agreements



Provider-to-customer

The **provider** announces to the customer the routes to reach all the Internet destinations

The **customer** announces to the provider the routes to reach its own networks and its customers networks (if any)

Typically this is established in meet-me-rooms

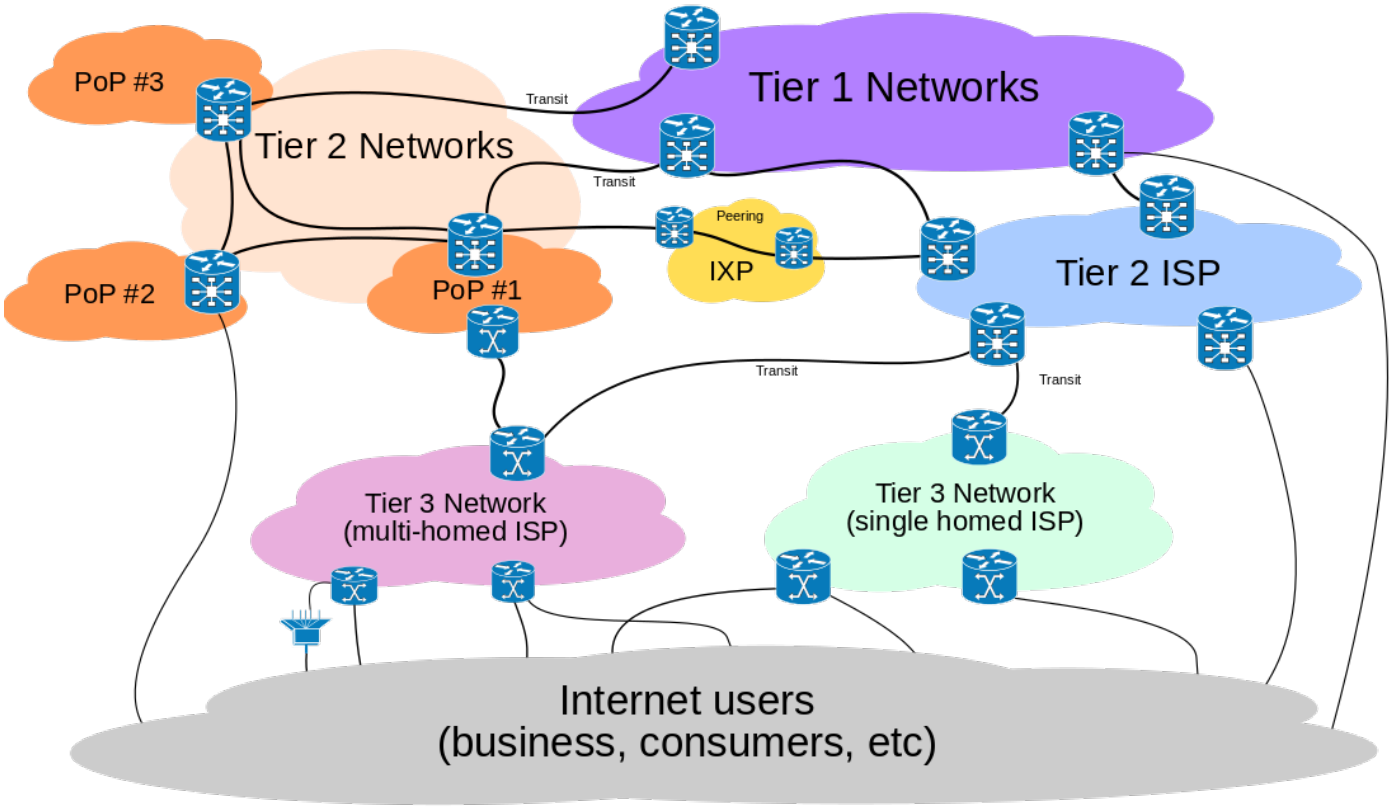
Peer-to-peer

Each **peer** announces to the other the routes to reach its own networks and its customers networks (if any)

Sessions established on IXPs are called *public* peering

Sessions established on meet-me-rooms are called *private* peering

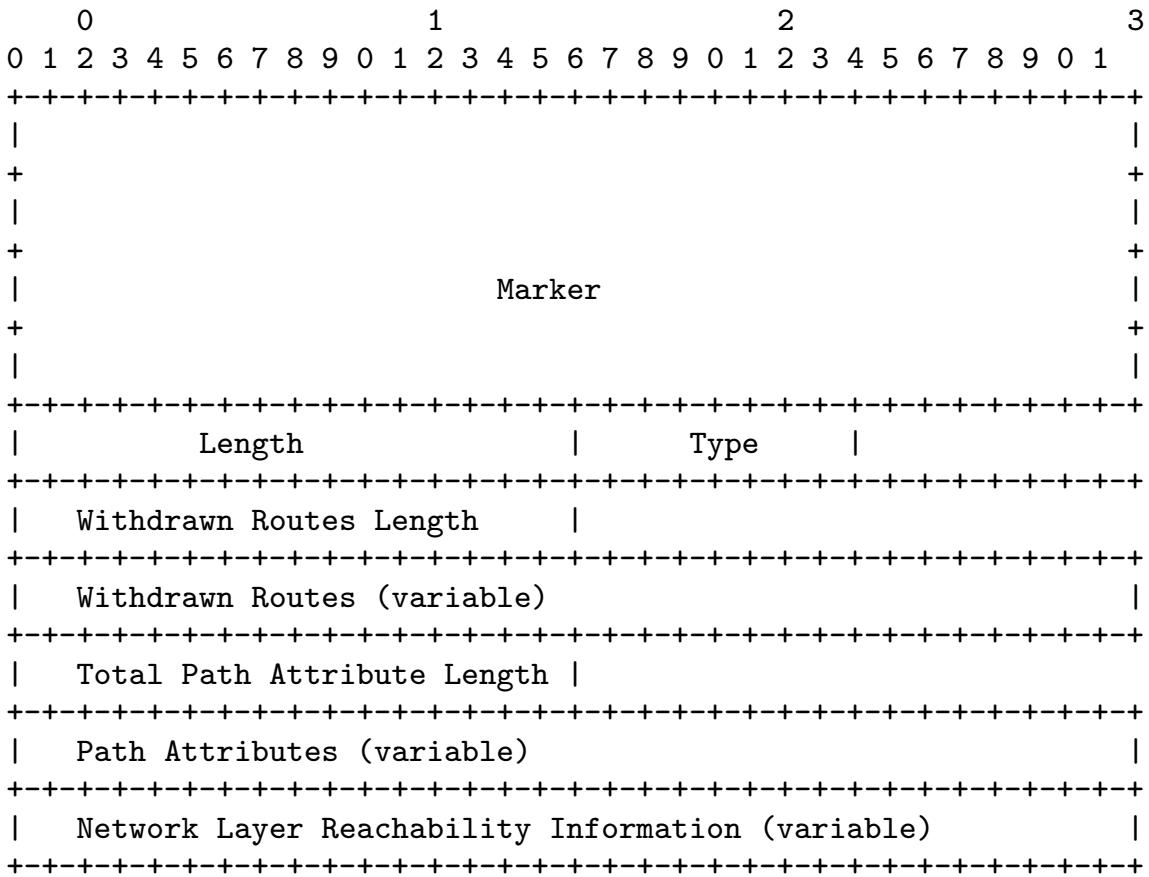
Internet hierarchy



Tier-1 ASes

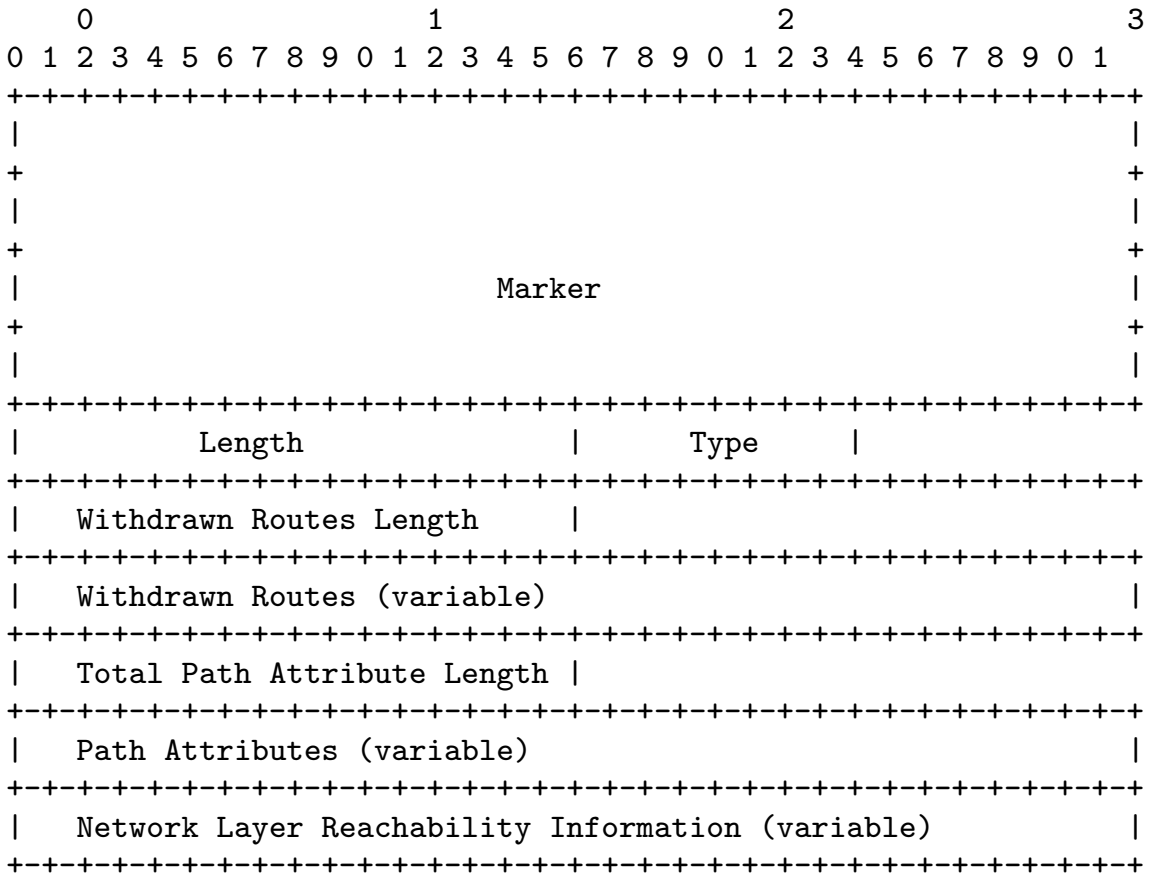
- At the top of the hierarchy there are Tier-1s, i.e. ASes without providers (they are about a dozen)
- Tier-1 ASes are big telcos (e.g. Level3, AT&T, NTT, ...)

BGP_UPDATE message



Routing updates contain all necessary information that BGP uses to construct a loop-free picture of the network

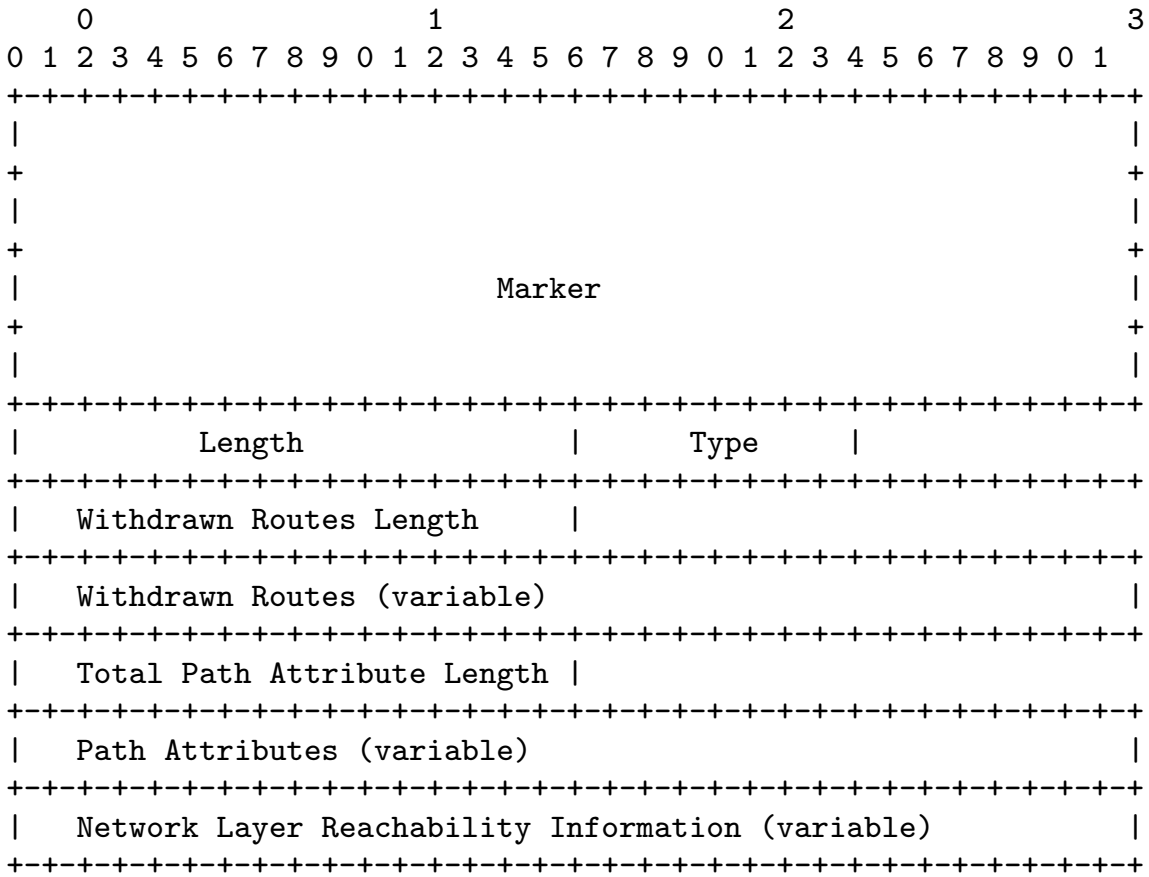
BGP_UPDATE message



NLRI

The list of destinations about which BGP is trying to inform its neighbors

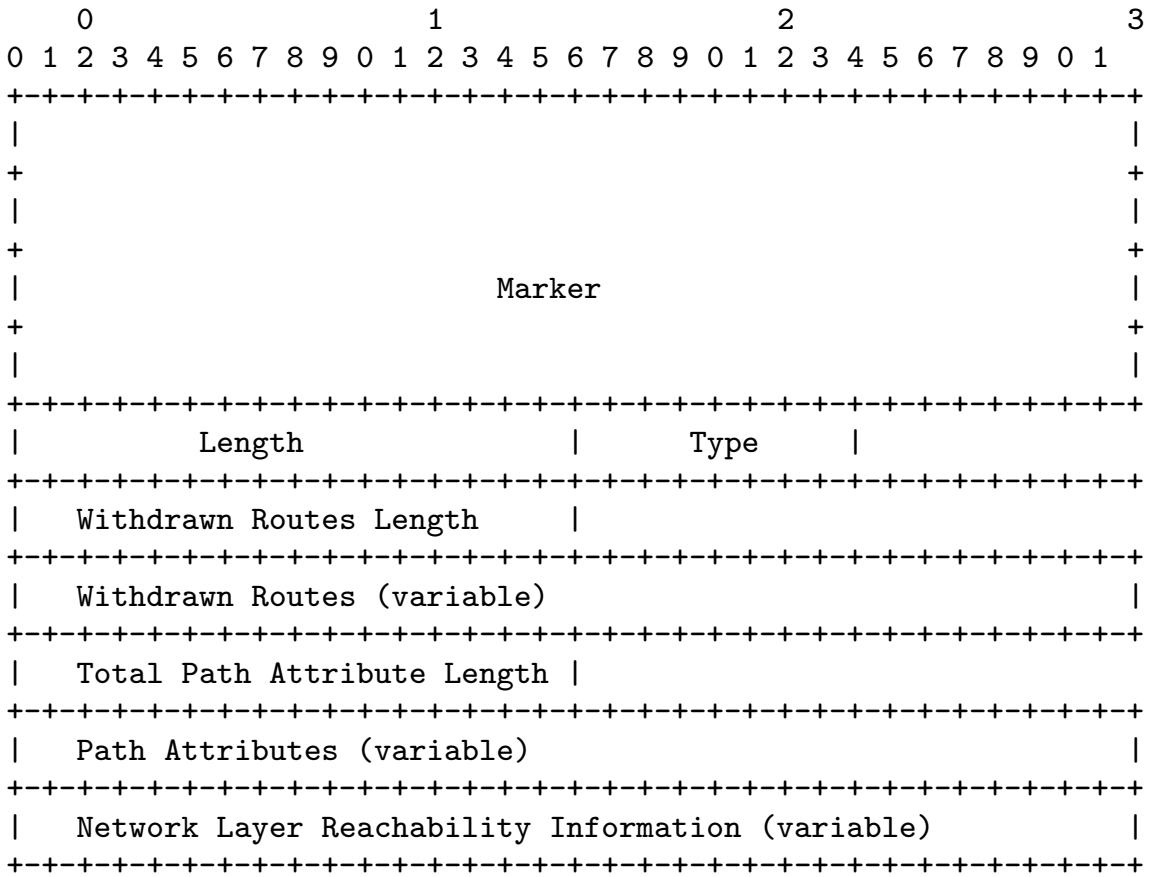
BGP_UPDATE message



Path attributes

The list of parameters used to keep track of NLRI-specific information

BGP_UPDATE message



Withdrawn routes

The list of destinations that are not feasible or no longer in service

BGP attributes

Value	Name	RFC
1	ORIGIN	4271
2	AS_PATH	4271
3	NEXT_HOP	4271
4	MULTI_EXIT_DISC	4271
5	LOCAL_PREF	4271
6	ATOMIC_AGGREGATE	4271
7	AGGREGATOR	4271
8	COMMUNITY	1997
14	MP_REACH_NLRI	4760
15	MP_UNREACH_NLRI	4760
16	EXTENDED_COMMUNITIES	4360
17	AS4_PATH	6793
18	AS4_AGGREGATOR	6793
32	LARGE_COMMUNITY	8092

Attribute classes

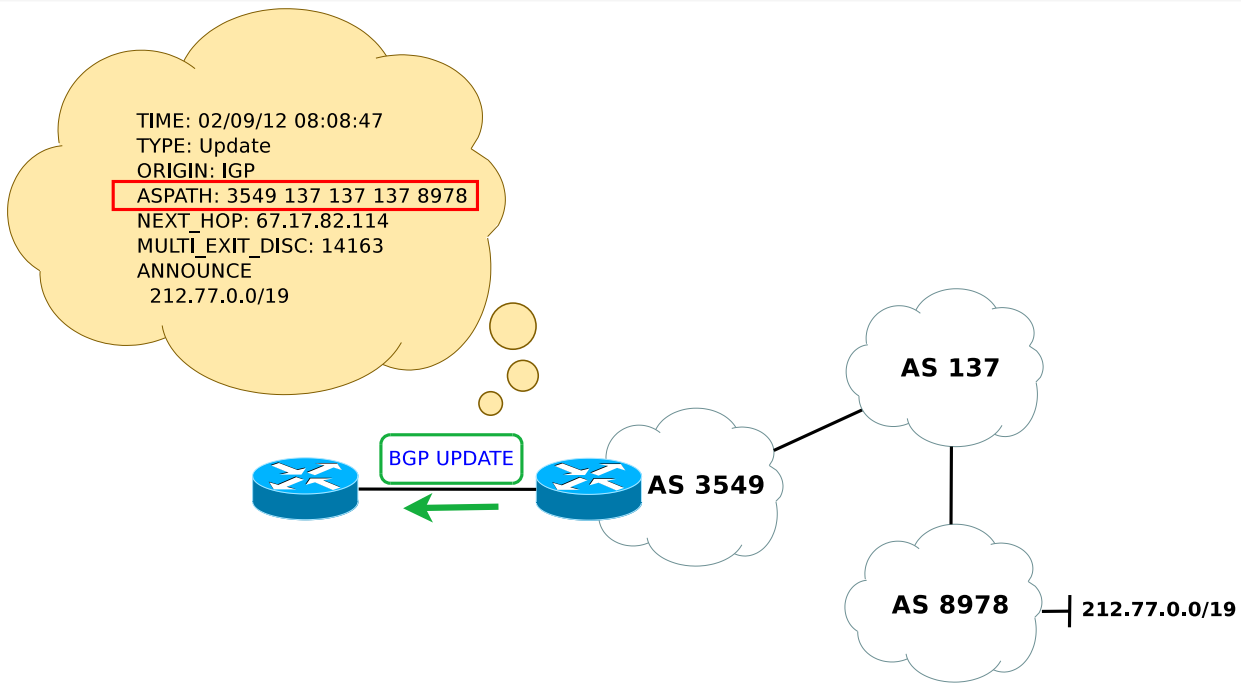
Used in the BGP filtering and route decision process. Can be:

- **Well-known mandatory** - Must be present if a NLRI is announced
- **Well-known discretionary** - Could be present if a NLRI is announced
- **Optional transitive** - If not recognized, must be propagated to neighbors
- **Optional non-transitive** - If not recognized, must **not** be propagated to neighbors

AS_PATH attribute

AS_PATH

- Contains a sequence of AS numbers that represent the path a route has traversed
- When sending routes to neighbors, the AS that originates the route adds its own AS number
- Thereafter, each AS that receives the route and passes it on to other neighbors will prepend its AS number to the list

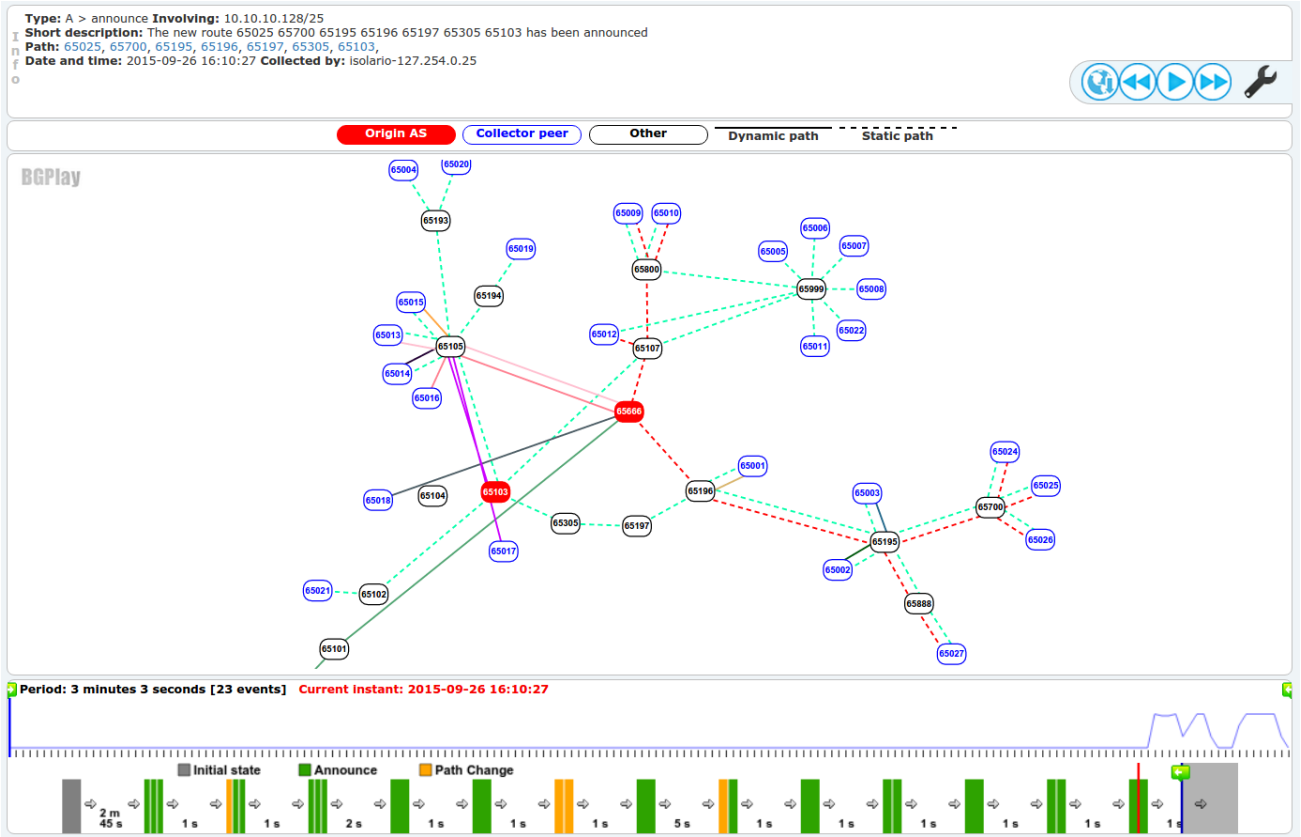


BGP decision process

How the best path algorithm works

1. Prefer the path with the highest `WEIGHT` (Cisco only)
2. Prefer the path with the highest `LOCAL_PREF`
3. Prefer the path that was locally originated (Cisco only)
4. Prefer the path with the shortest `AS_PATH`
5. Prefer the path that was locally originated (Juniper only)
6. Prefer the path with the lowest origin type
7. Prefer the path with the lowest multi-exit discriminator (`MED`)
8. Prefer eBGP over iBGP paths
9. Prefer the path with the lowest IGP metric to the BGP next hop
10. Prefer the route that comes from the BGP router with the lowest router ID
11. Prefer the path that comes from the lowest neighbor address

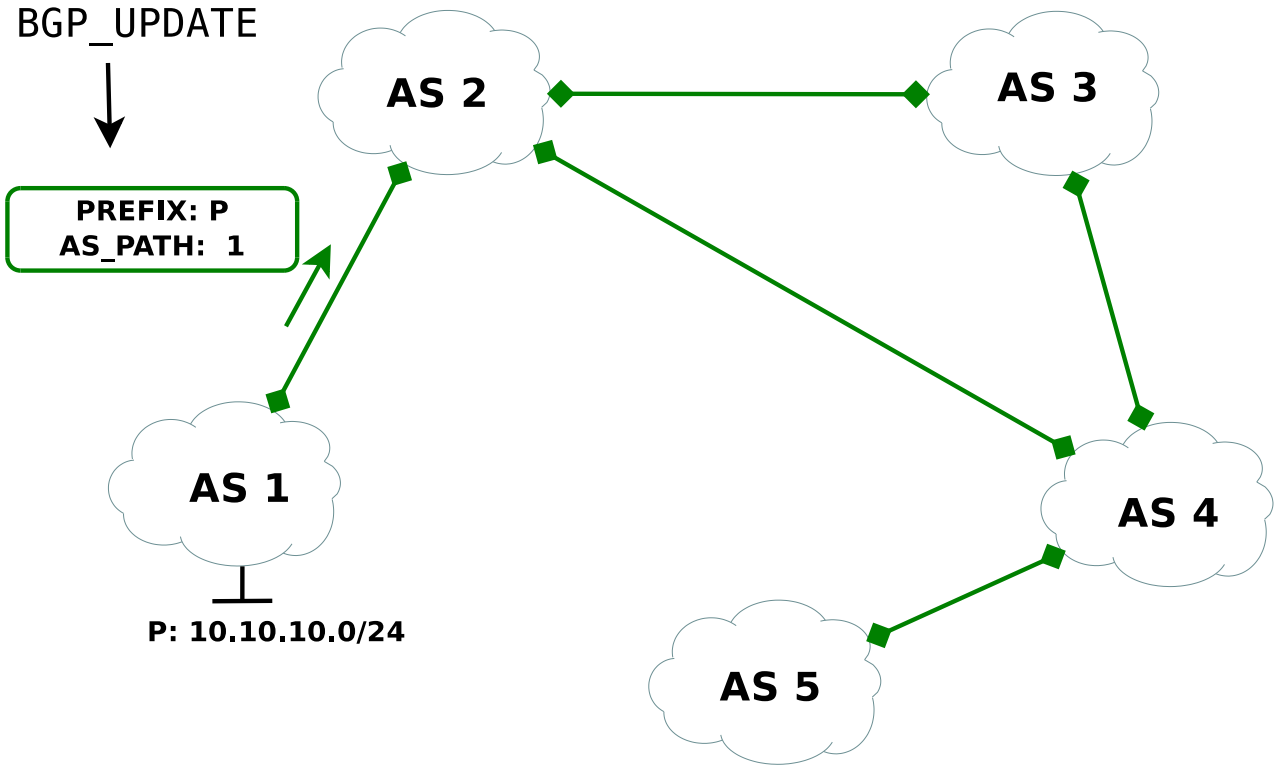
BGP routing events



Routing events

- Route announcement
- Route replacement
- Route withdrawn

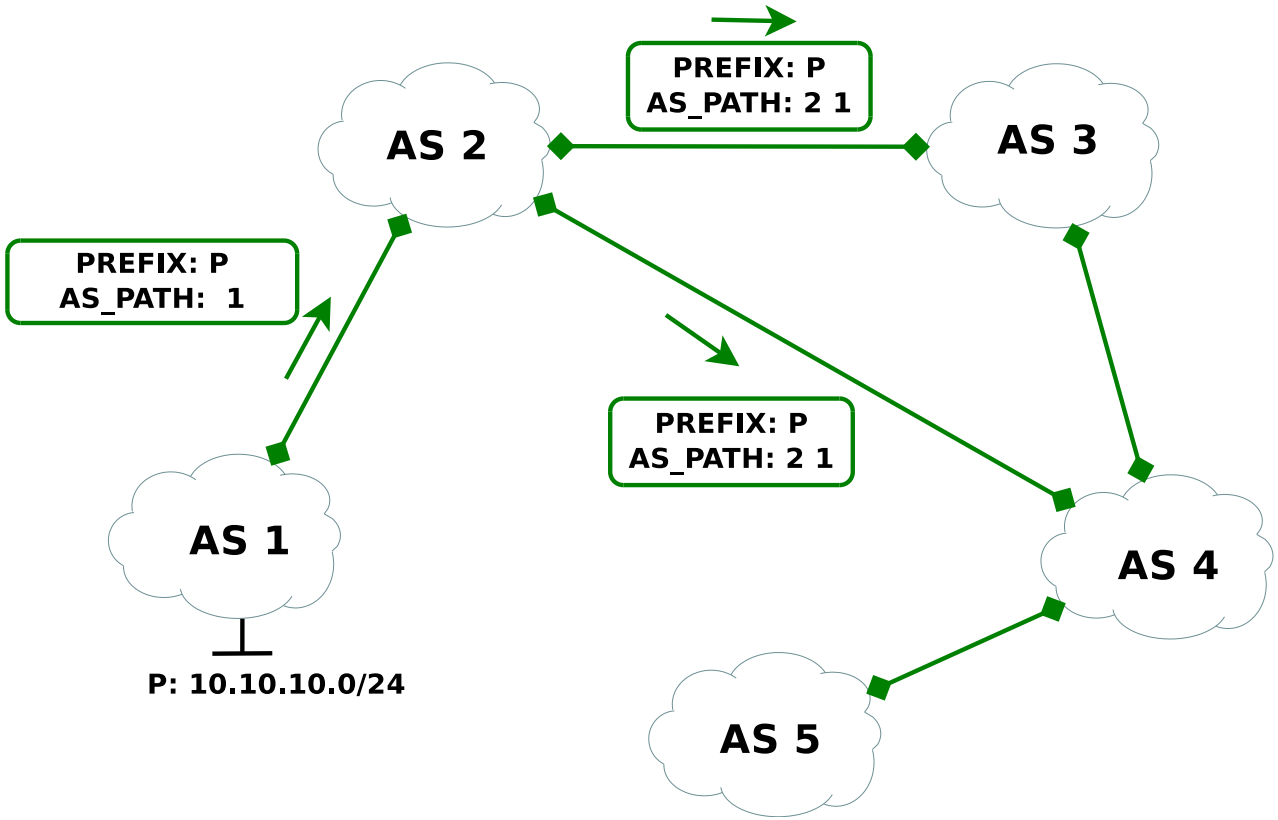
Route announcement



Route announcement

Each AS announces to its BGP neighbors the prefixes that owns. This is the first step to make the prefixes globally reachable.

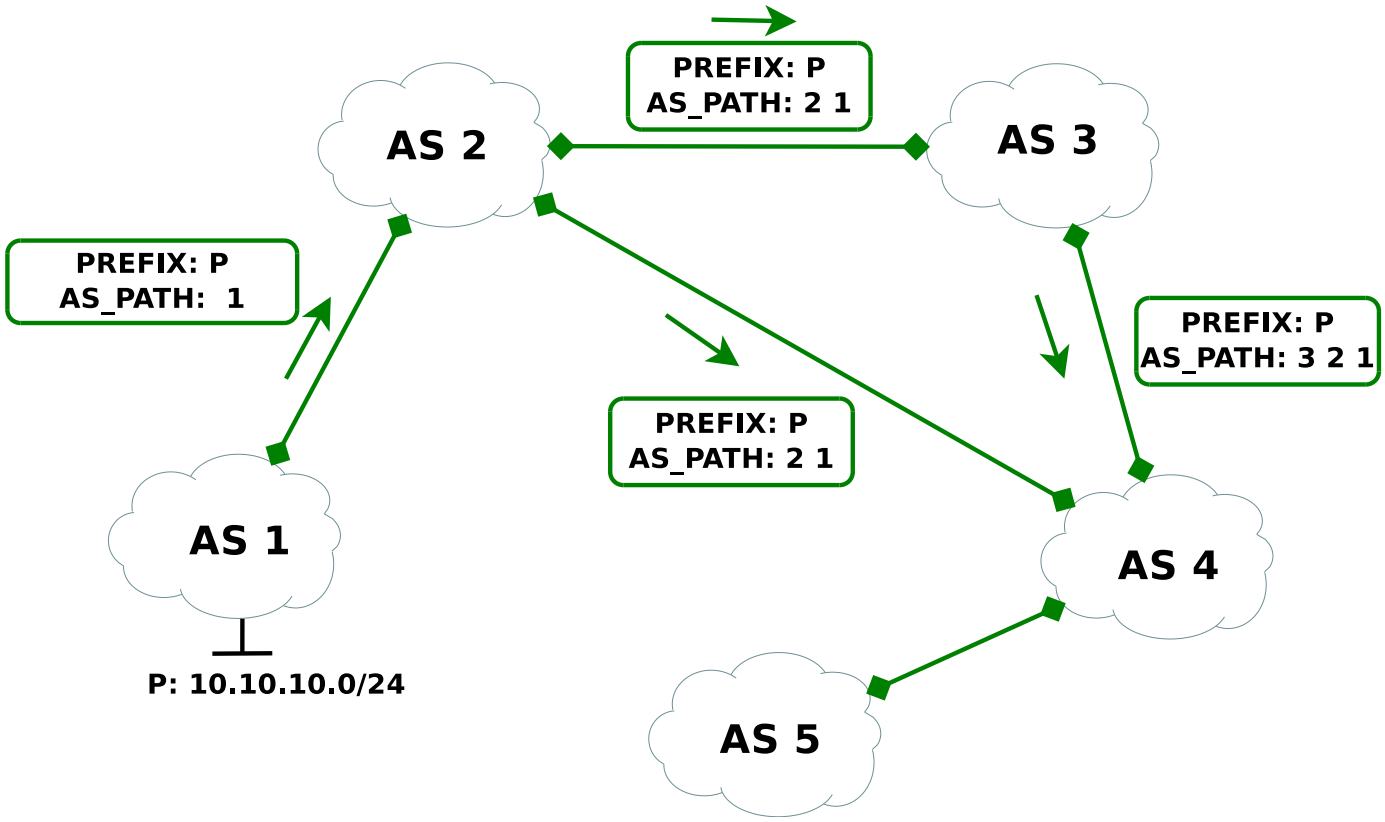
Route announcement



Route announcement

Each AS receiving an `UPDATE` propagates the information to its BGP neighbors, prepending to the `AS_PATH` its ASN

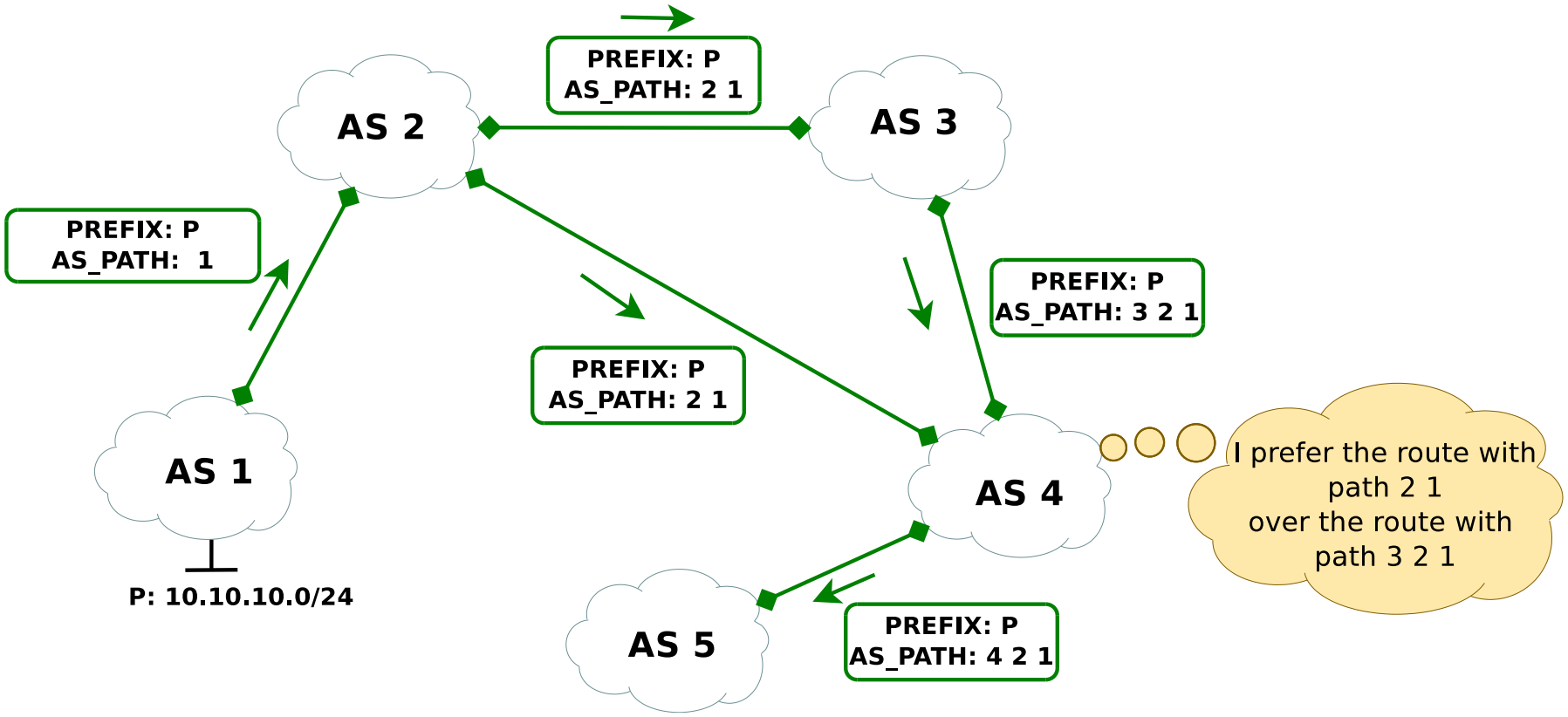
Route announcement



Route announcement

Each AS receiving a route propagates the information to its BGP neighbors, prepending to the `AS_PATH` its ASN

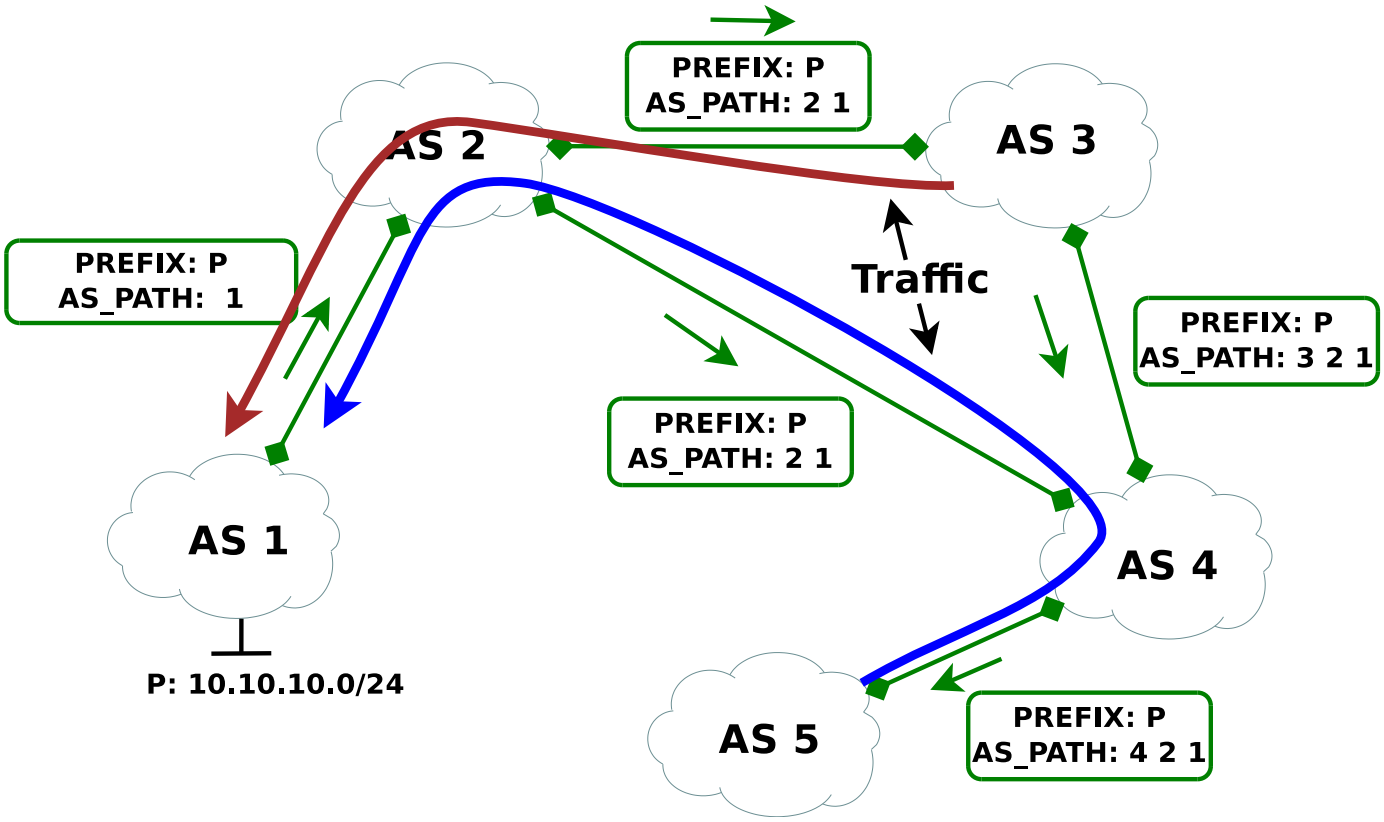
Route announcement



Route announcement

When an AS receives multiple routes to reach a destination, it must apply the **BGP decision process** to choose the **best route**

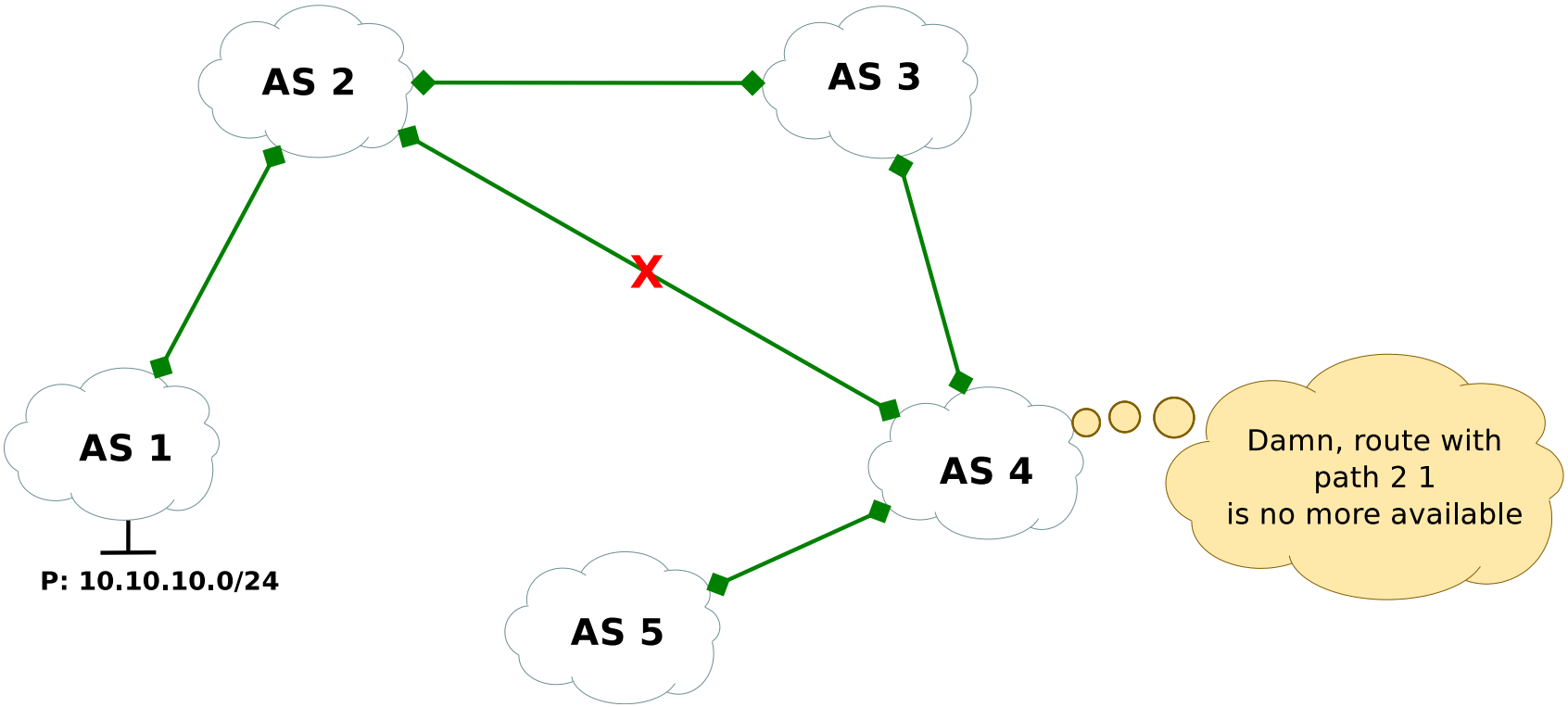
Route announcement



Route announcement

When an AS *A* announces a route to another AS B (towards P), this means that A is **willing** to transit traffic coming from B (directed to P)

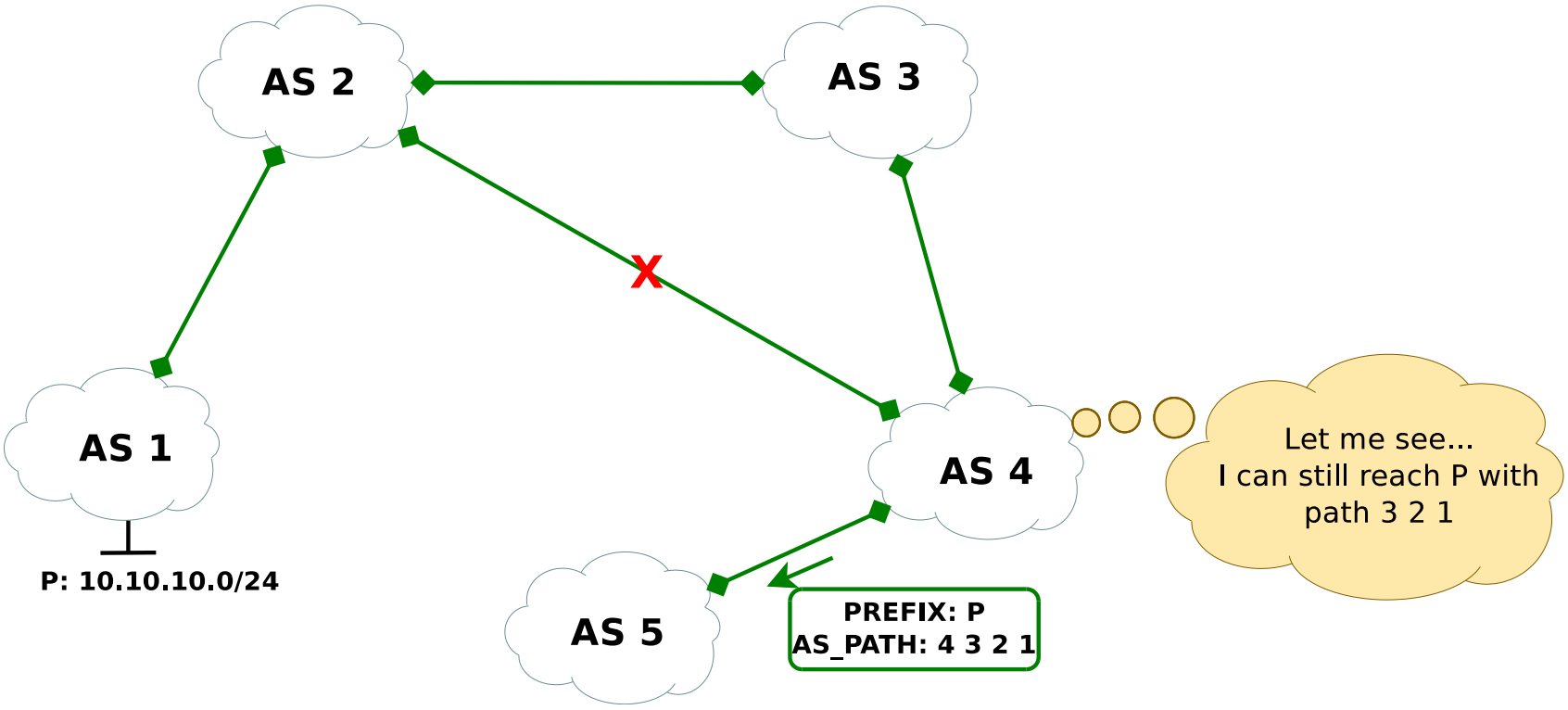
Route replacement



Route replacement

Whenever a link between two ASes fails the BGP session is shut down and every prefix is implicitly withdrawn.

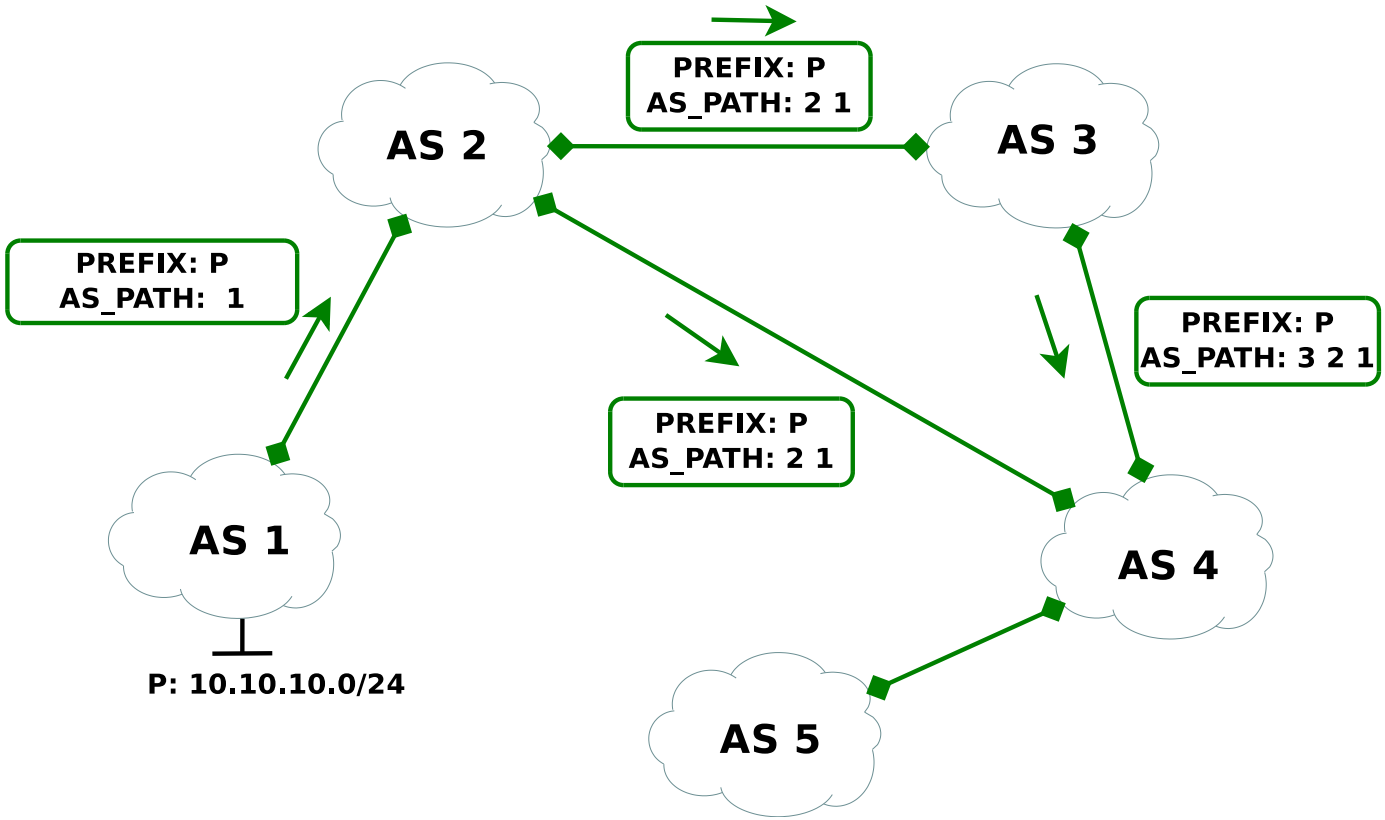
Route replacement



Route replacement

As soon as a change in routing is perceived, the BGP decision process tries to find an alternative way to reach the involved prefixes

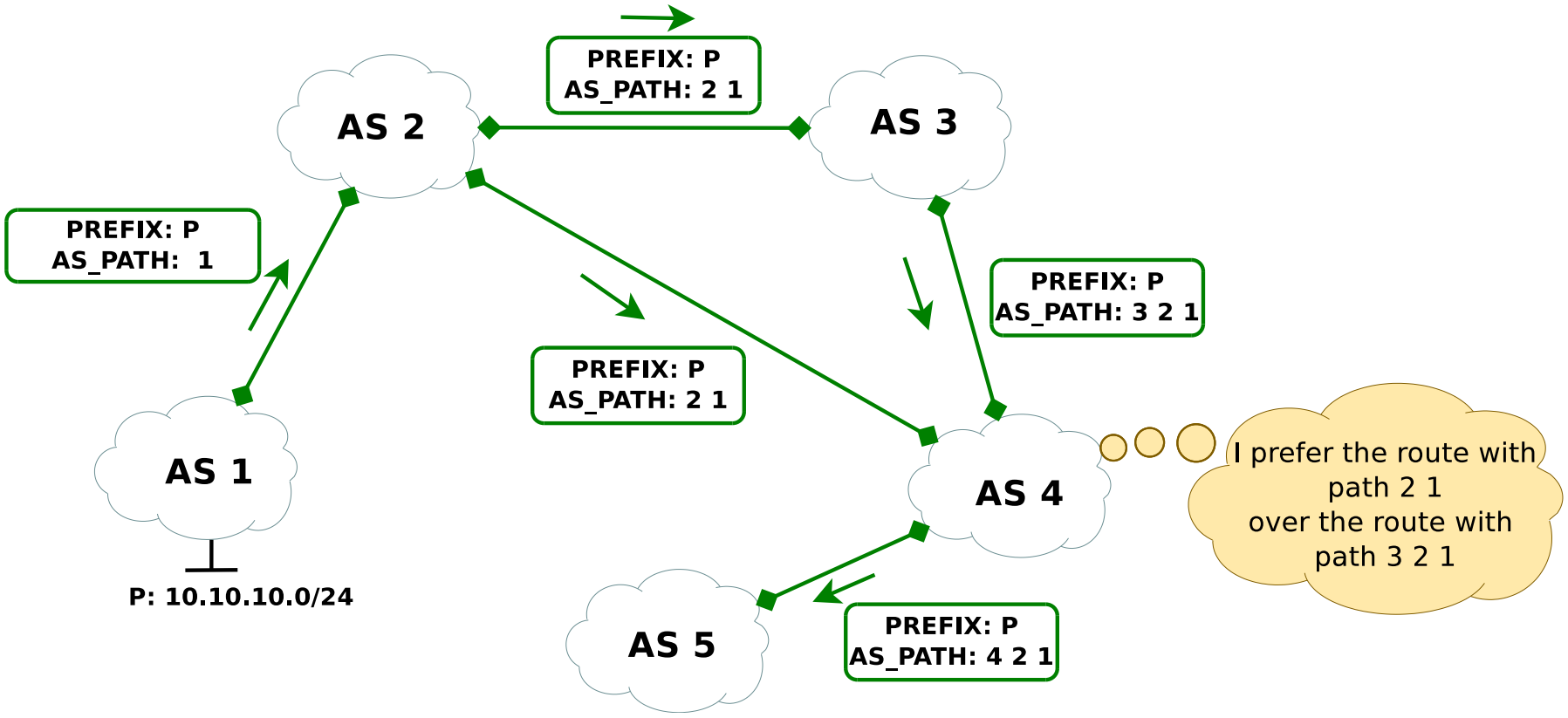
Route replacement



Route replacement

Link failures are not the only reason for route replacements

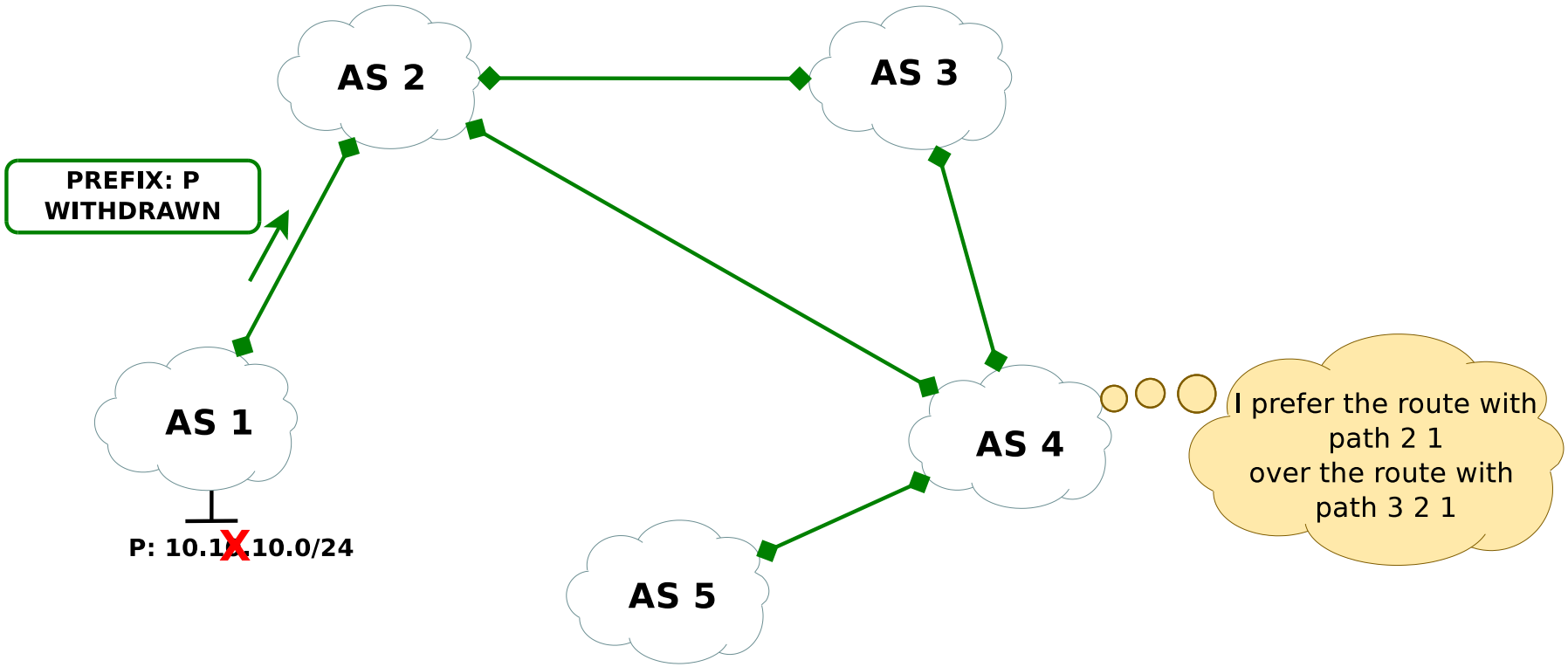
Route replacement



Route replacement

A route is replaced whenever a route with better characteristics shows up

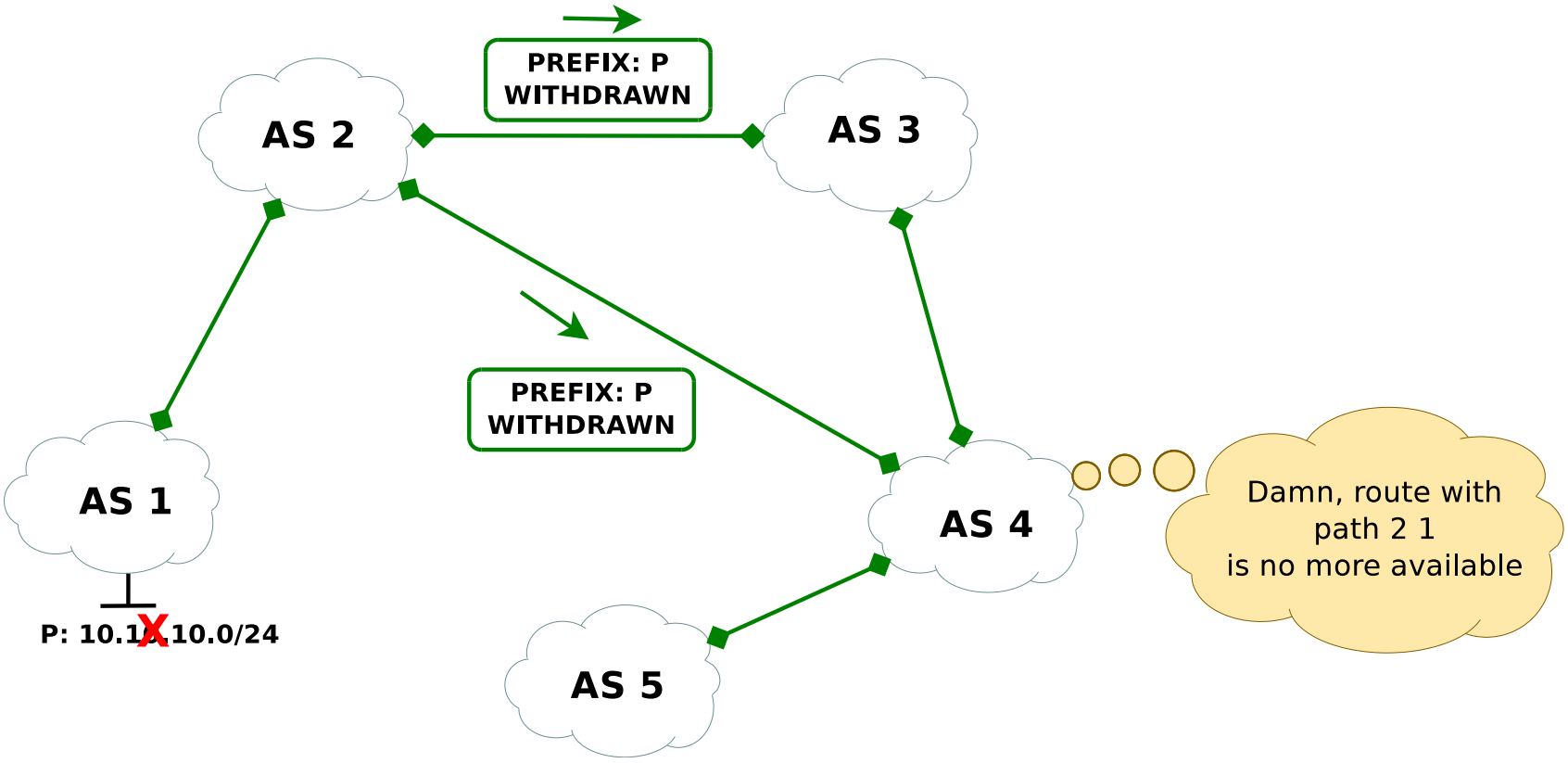
Route withdrawn



Route withdrawn

Whenever a prefix is no more reachable, an `UPDATE` is propagated to its BGP neighbors announcing its withdrawal from the Internet

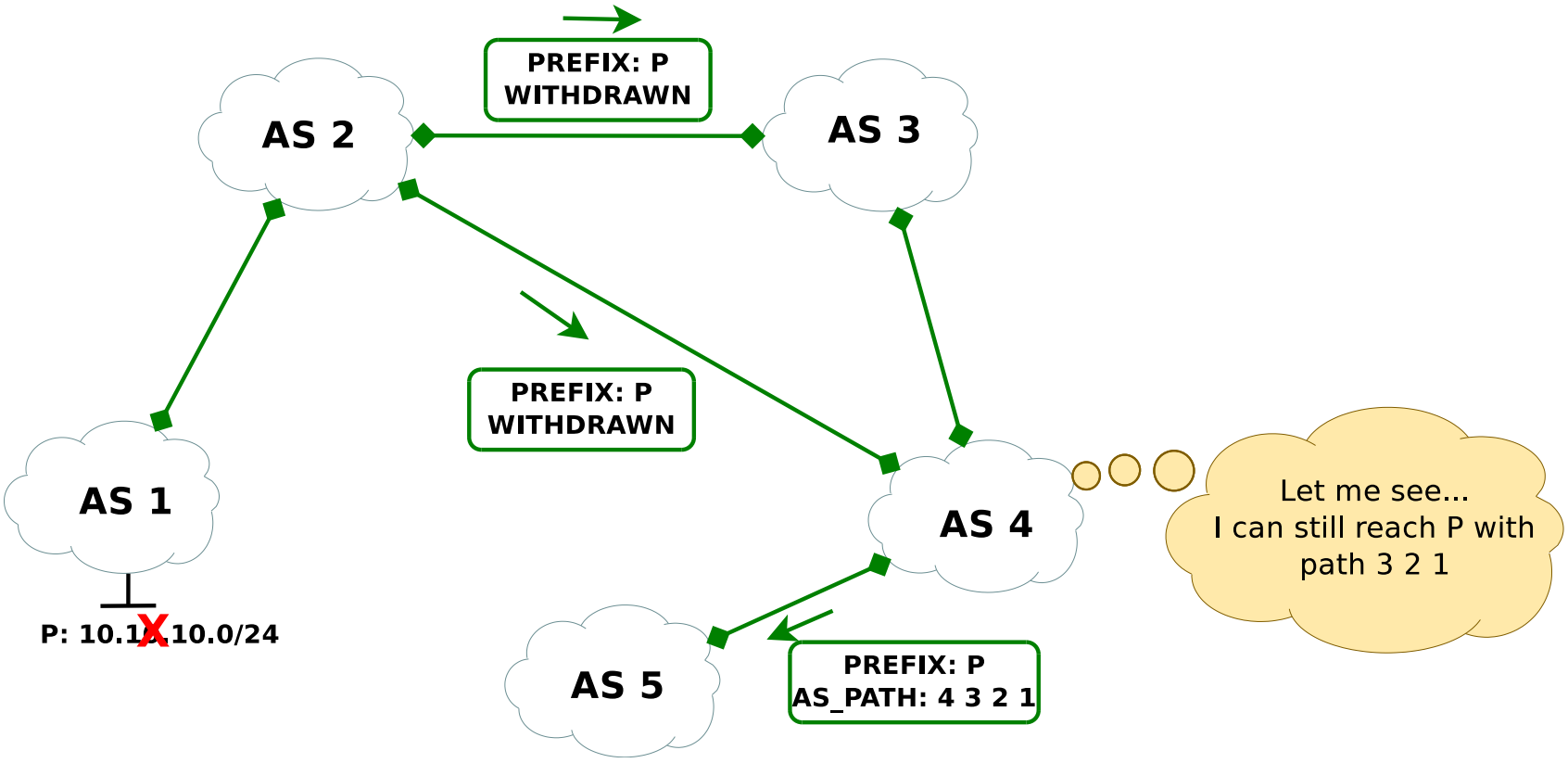
Route withdrawn



Route withdrawn

Each AS receiving an `UPDATE` propagates the information to its BGP neighbors

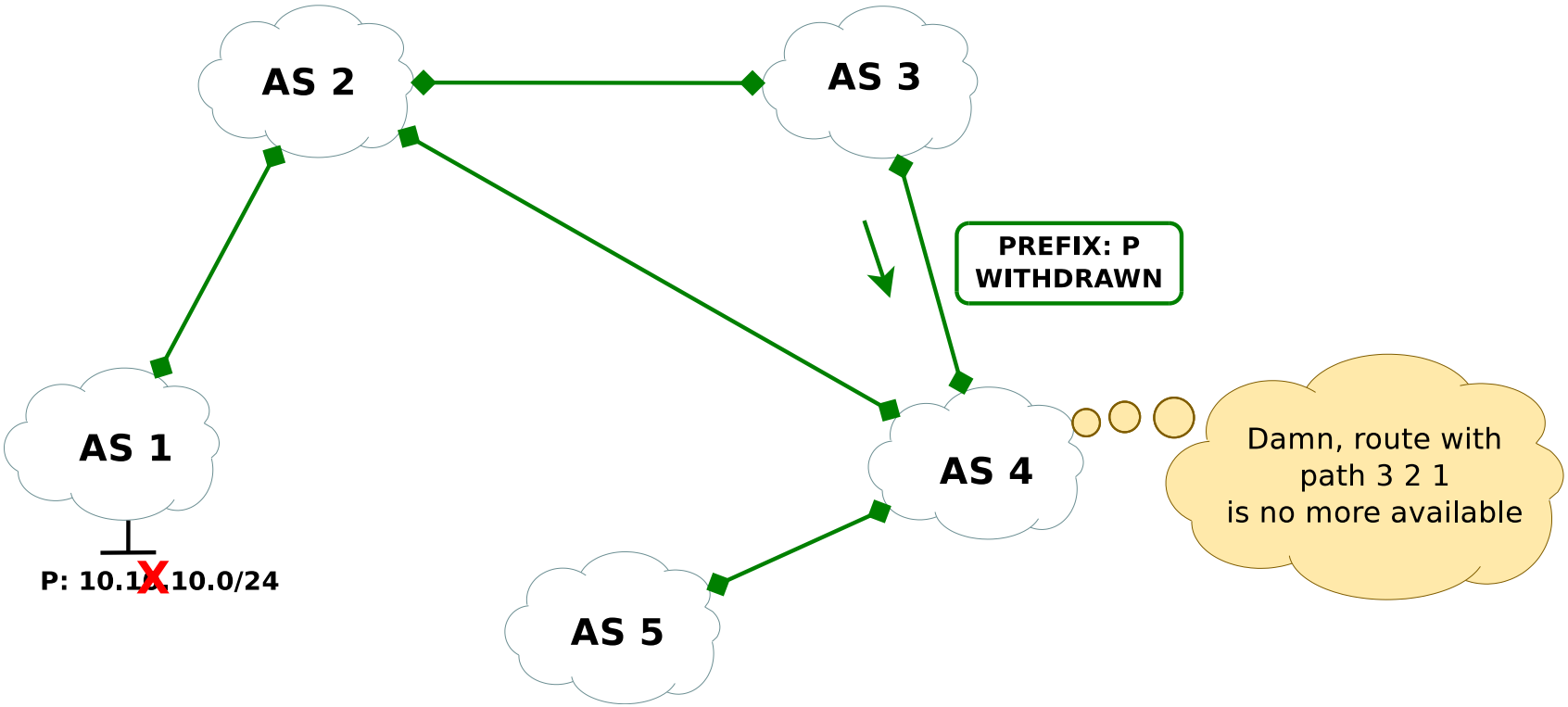
Route withdrawn



Route withdrawn

Again, route is replaced whenever a route with better characteristics shows up, and an unusable AS path shows up

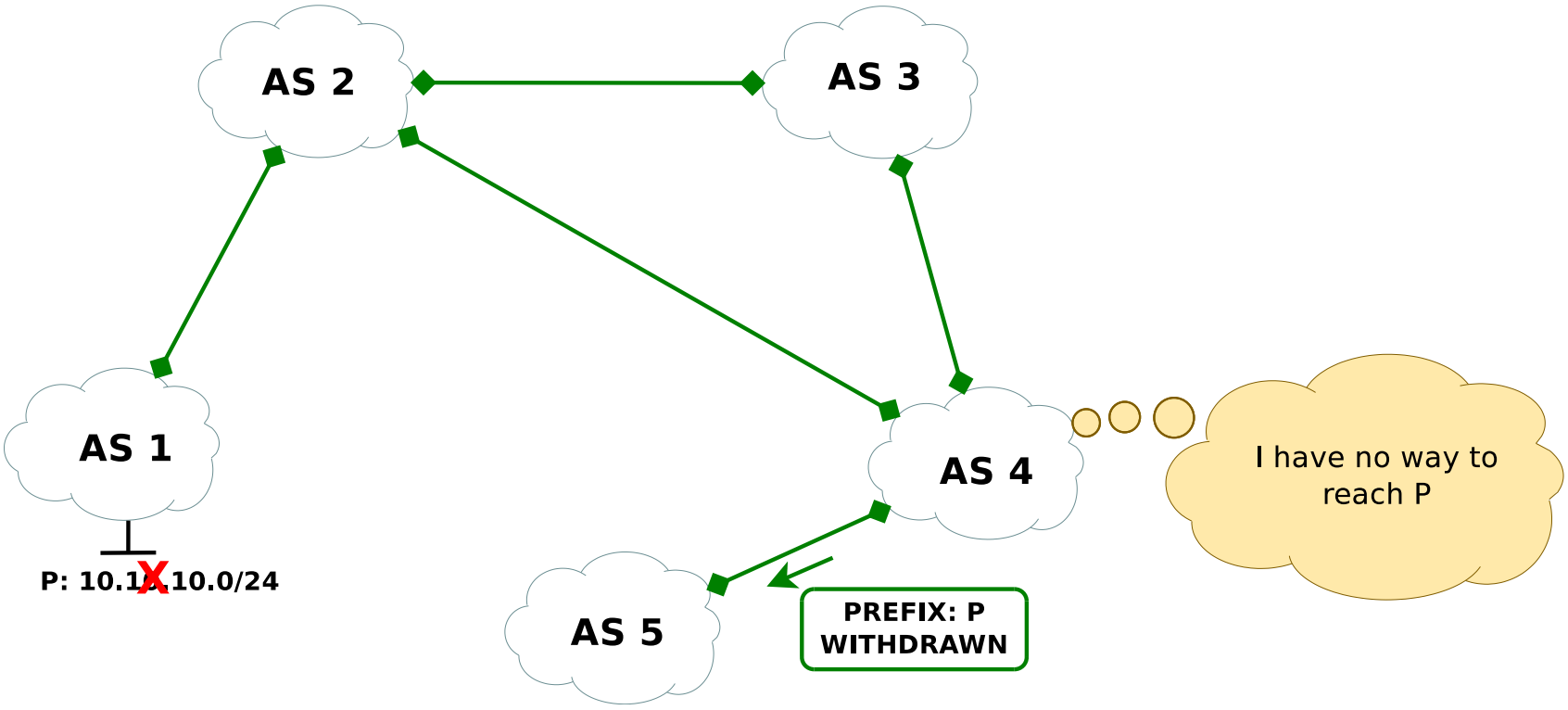
Route withdrawn



Route withdrawn

Until every neighbor announces the withdrawn of the given prefix

Route withdrawn



Route withdrawn

Then, the prefix is declared as withdrawn, and an `UPDATE` is propagated

BGP security issues

BGP security

- BGP it is not designed to provide any security guarantee
- BGP messages are neither authenticated nor encrypted
- Prone to misconfigurations and attacks

BGP problems

BGP routing anomalies

- Prefix hijack
- Route leak

Those problems can be either caused intentionally (**attacks**) or unintentionally (**misconfigurations**).

Prefix hijack

Prefix hijack

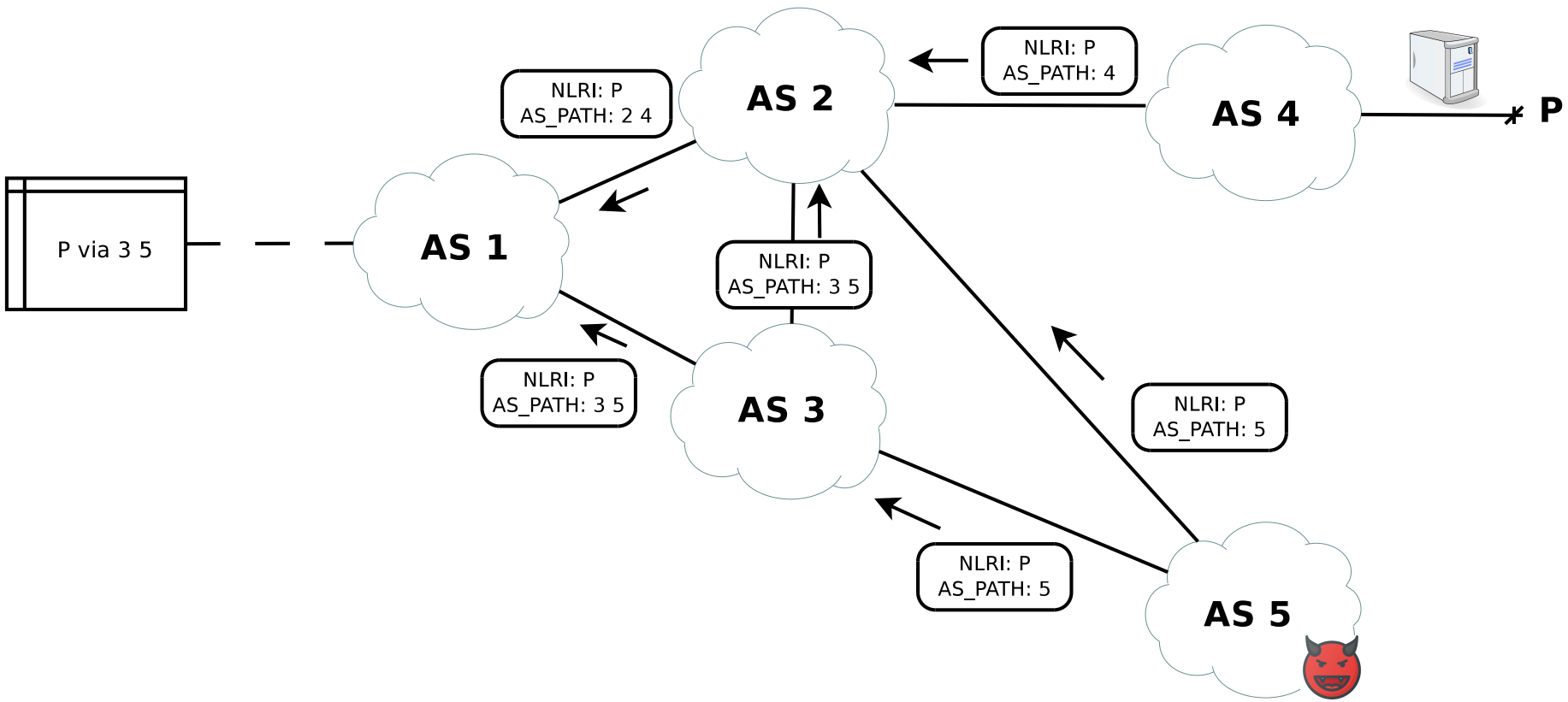
A prefix hijack occurs when a prefix is **originated** by an AS that does not own that prefix.

- This is possible because `BGP_UPDATE` messages can be forged ad-hoc
- The legitimate and fake announcement co-exist
- Depending on the hijack type and subject a portion or even the whole Internet can be affected

Many variants...

Exact prefix hijack

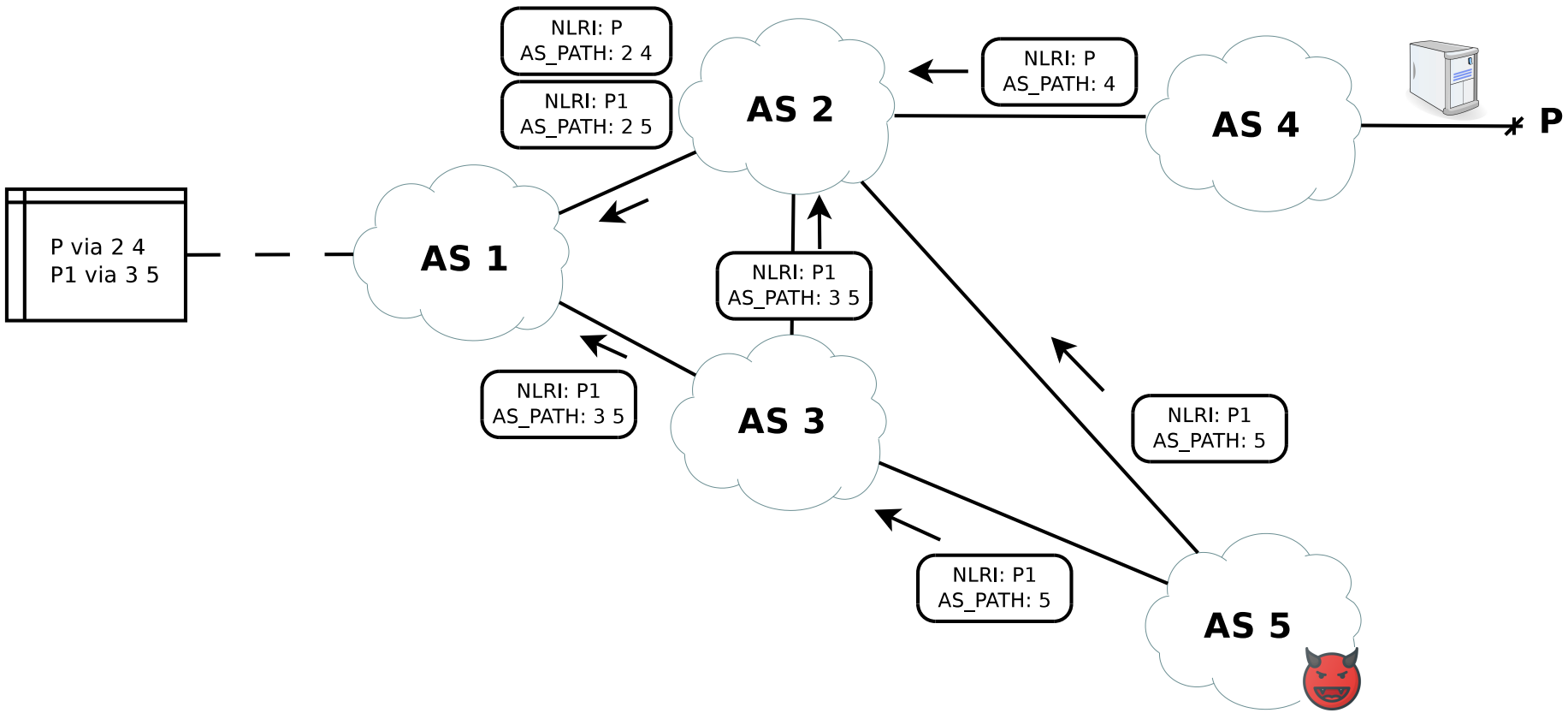
The attacker originates a prefix owned by another AS. Other ASes **may or may not** believe to the attacker.



Many variants...

Subprefix hijack

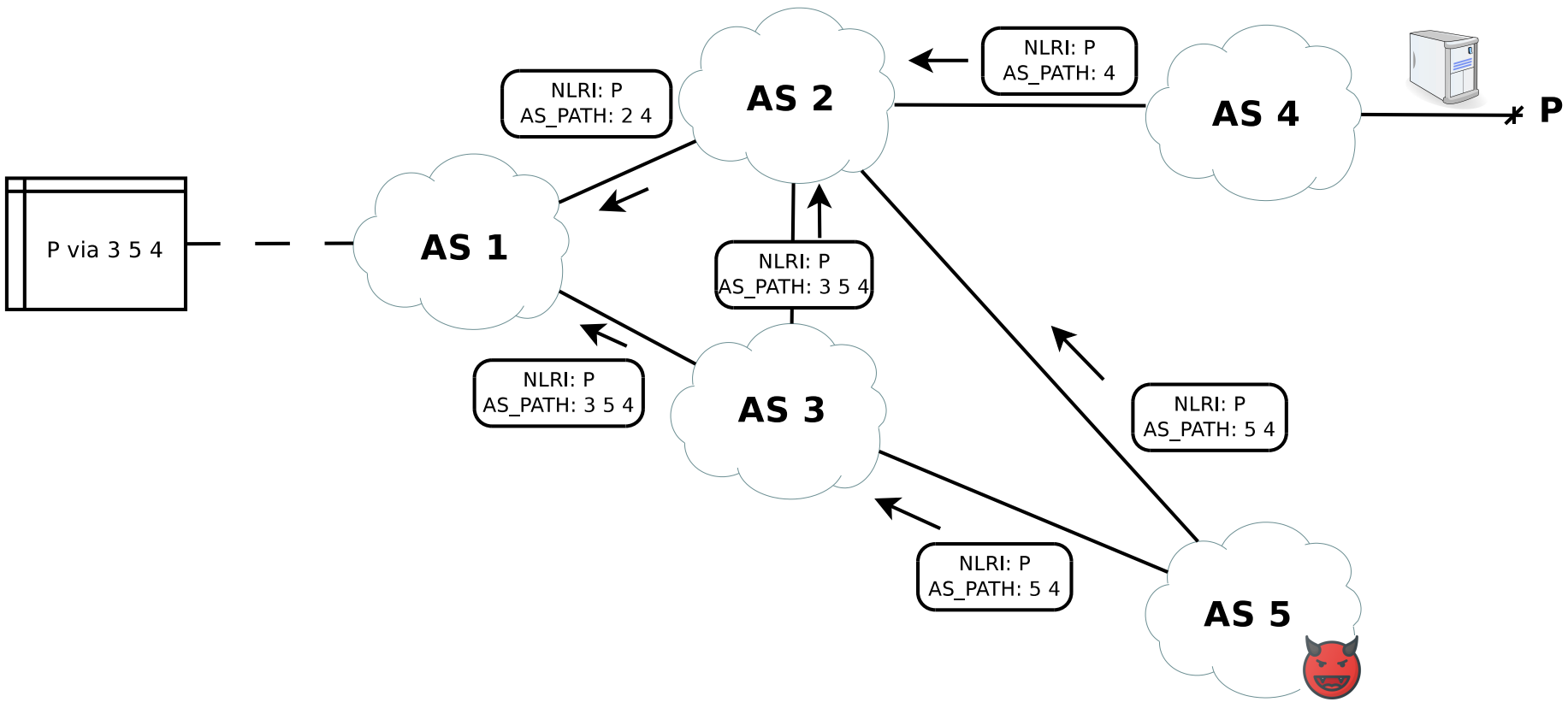
The attacker originates a subnet of a given prefix. All the other ASes will accept the route!



Many variants...

AS path forgery

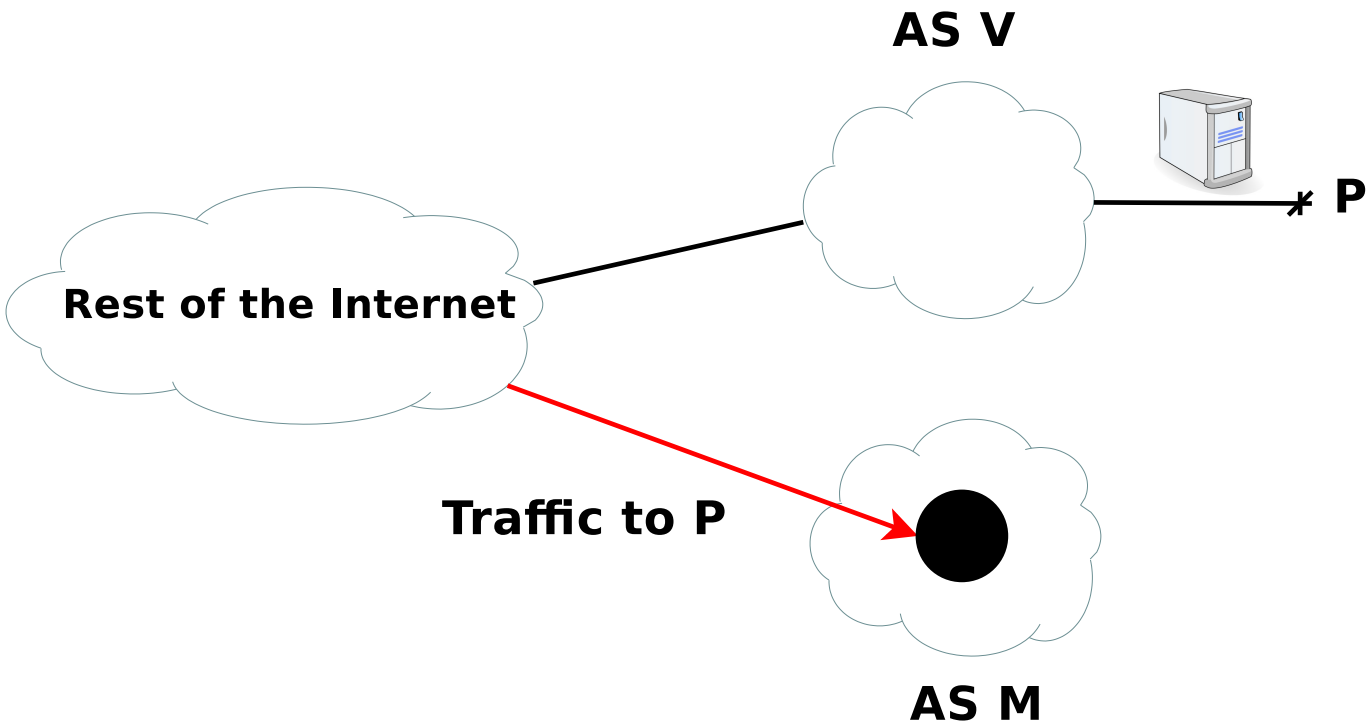
The attacker forges the `AS_PATH` to make more difficult the detection of the hijack



Prefix hijack: consequences

Blackholing (DoS)

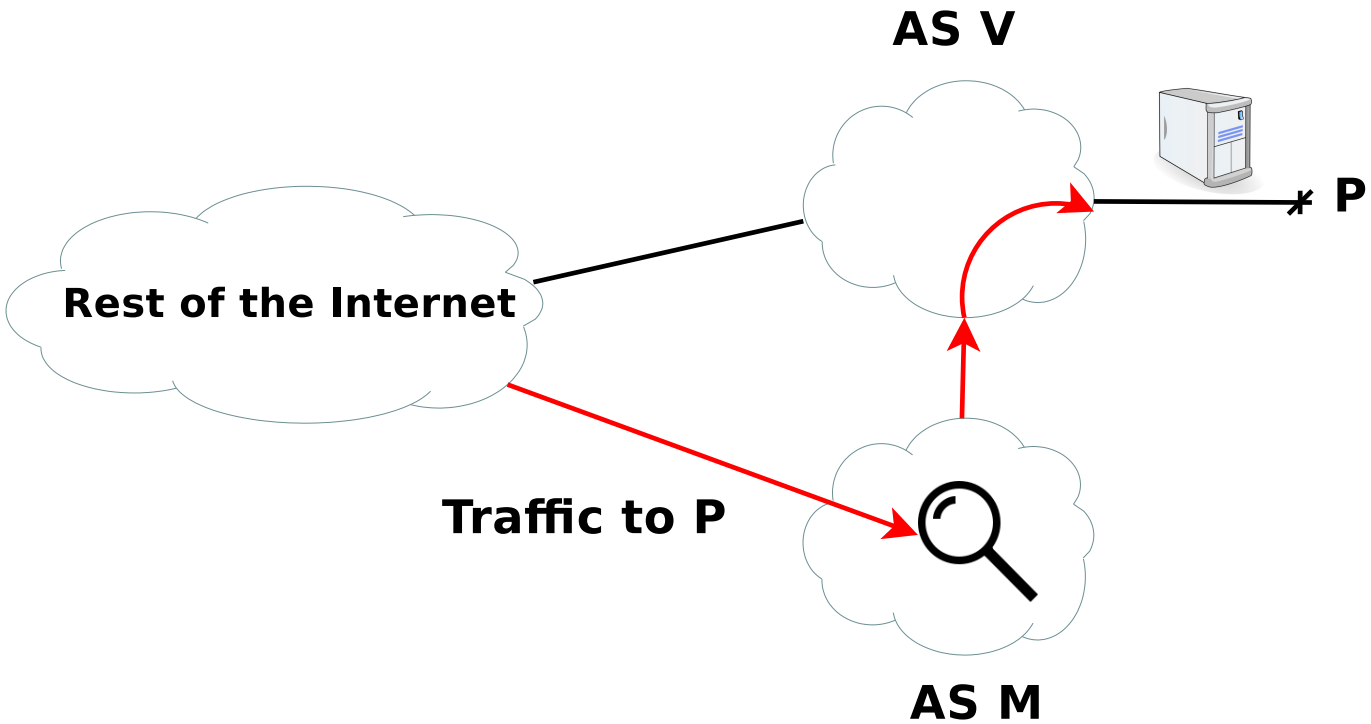
The service hosted on the hijacked prefix



Prefix hijack: consequences

Interception

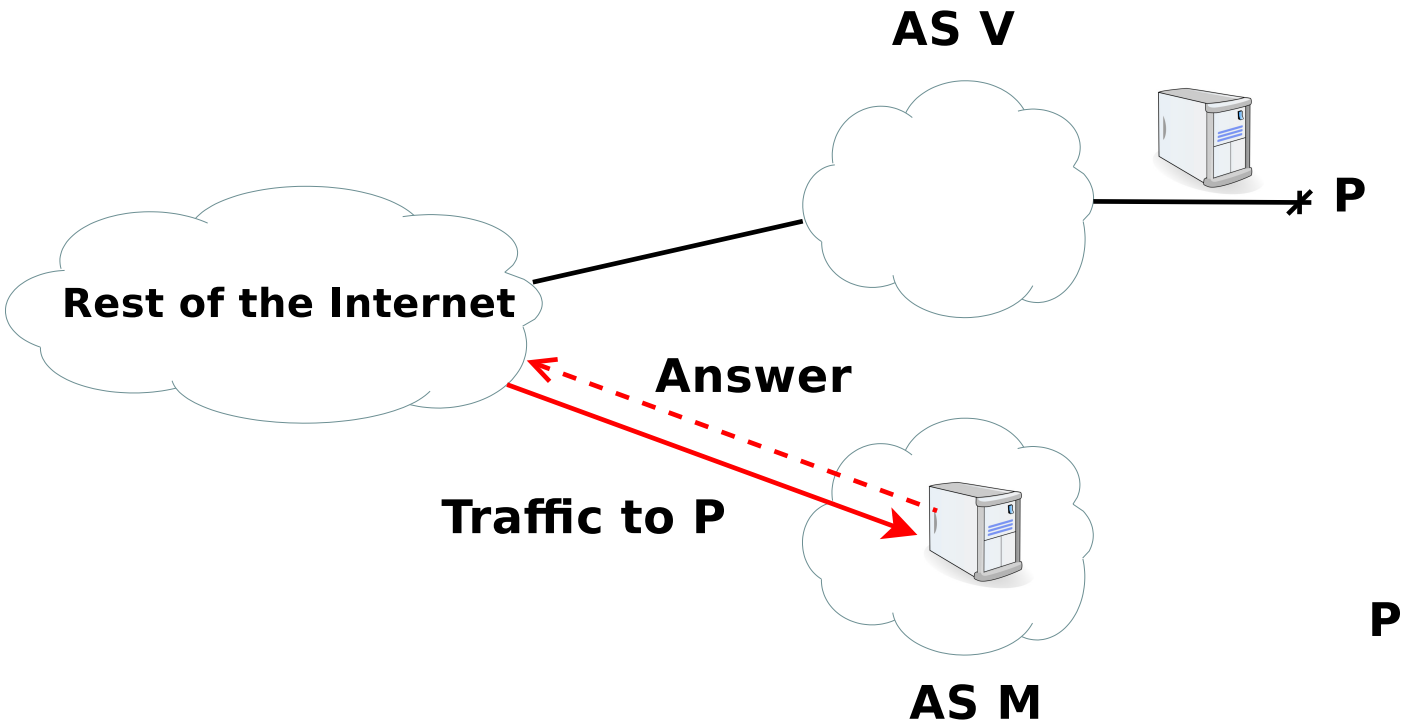
The attacker intercept the traffic toward the hijacked prefix



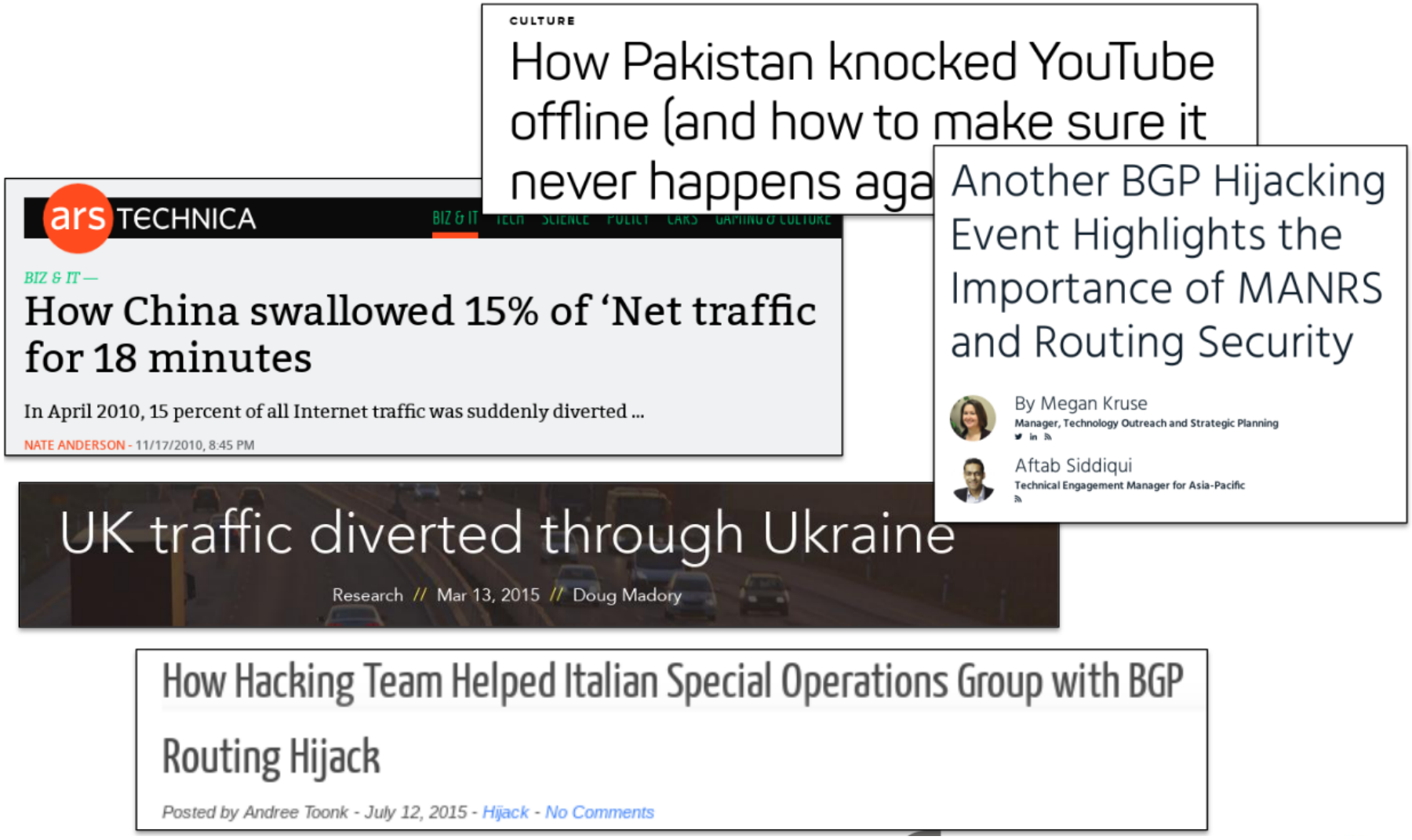
Prefix hijack: consequences

Spoofing

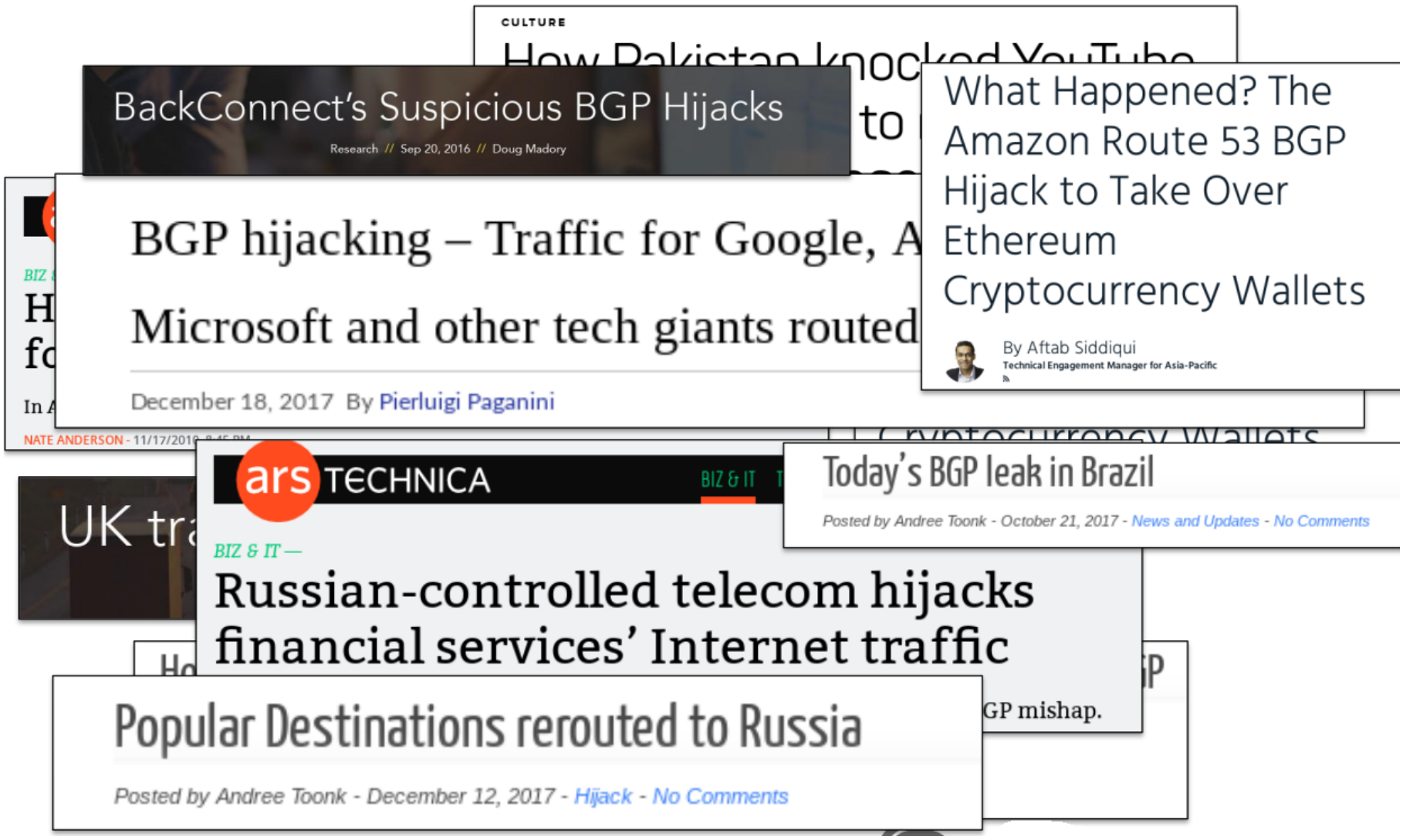
The attacker impersonates the service hosted on the hijacked prefix



Hijacks happens frequently



Hijacks happens frequently



Notable examples

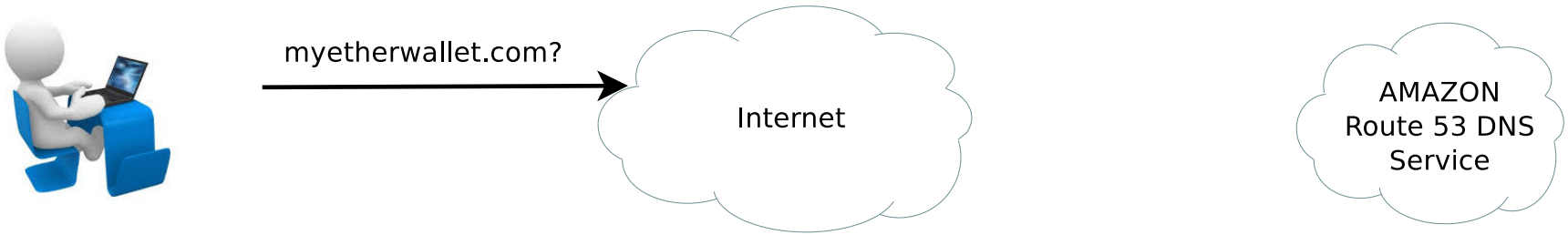
Year	Description
1997	AS 7007 incident
2004	TTNet in Turkey hijacks the Internet
2005	Google Outage
2006	Con-Edison hijacks big chunk of the Internet
2008	Brazilian ISP hijacks the Internet
2008	Pakistan Telecom hijacks YouTube
2010	A Chinese ISP hijacks the Internet
2013	Hacking Team helps Italian Military Policy to hunt a criminal
2014	Canadian ISP used to redirect data from ISPs
2015	BGP Hijack out of India
2016	China Telecom hijacks routes from Canada to South Korean government sites
2017	Iranian pornography censorship
2017	Google, Facebook, Apple, ... destinations rerouted to Russia
2018	MyEtherWallet hijack
2018	Iran Telecom Company hijacks ten prefixes of Telegram Messenger

Notable examples

Year	Description
1997	AS 7007 incident
2004	TTNet in Turkey hijacks the Internet
2005	Google Outage
2006	Con-Edison hijacks big chunk of the Internet
2008	Brazilian ISP hijacks the Internet
2008	Pakistan Telecom hijacks YouTube
2010	A Chinese ISP hijacks the Internet
2013	Hacking Team helps Italian Military Policy to hunt a criminal
2014	Canadian ISP used to redirect data from ISPs
2015	BGP Hijack out of India
2016	China Telecom hijacks routes from Canada to South Korean government sites
2017	Iranian pornography censorship
2017	Google, Facebook, Apple, ... destinations rerouted to Russia
2018	**MyEtherWallet hijack**
2018	Iran Telecom Company hijacks ten prefixes of Telegram Messenger

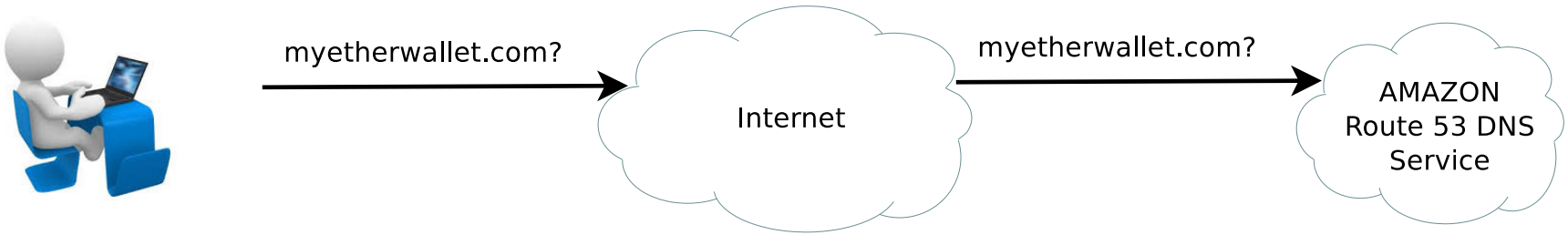
MyEtherWallet hijack

- MyEtherWallet uses Amazon's Route 53 DNS service



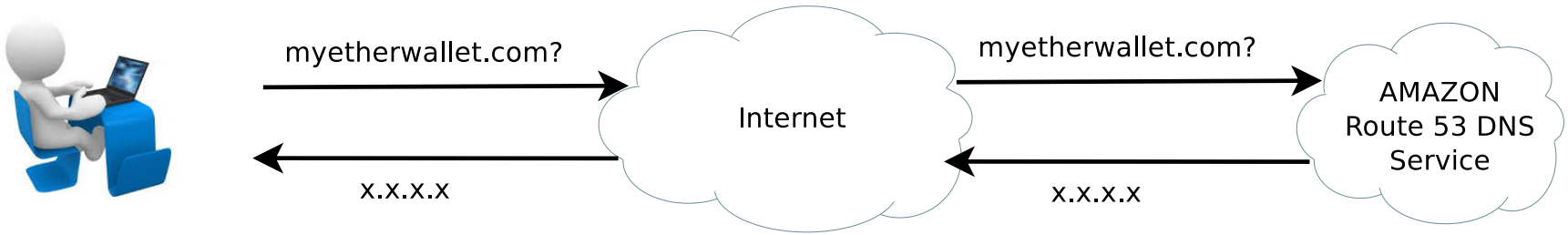
MyEtherWallet hijack

- DNS resolution for `myetherwallet.com` reaches an Amazon server running the DNS service



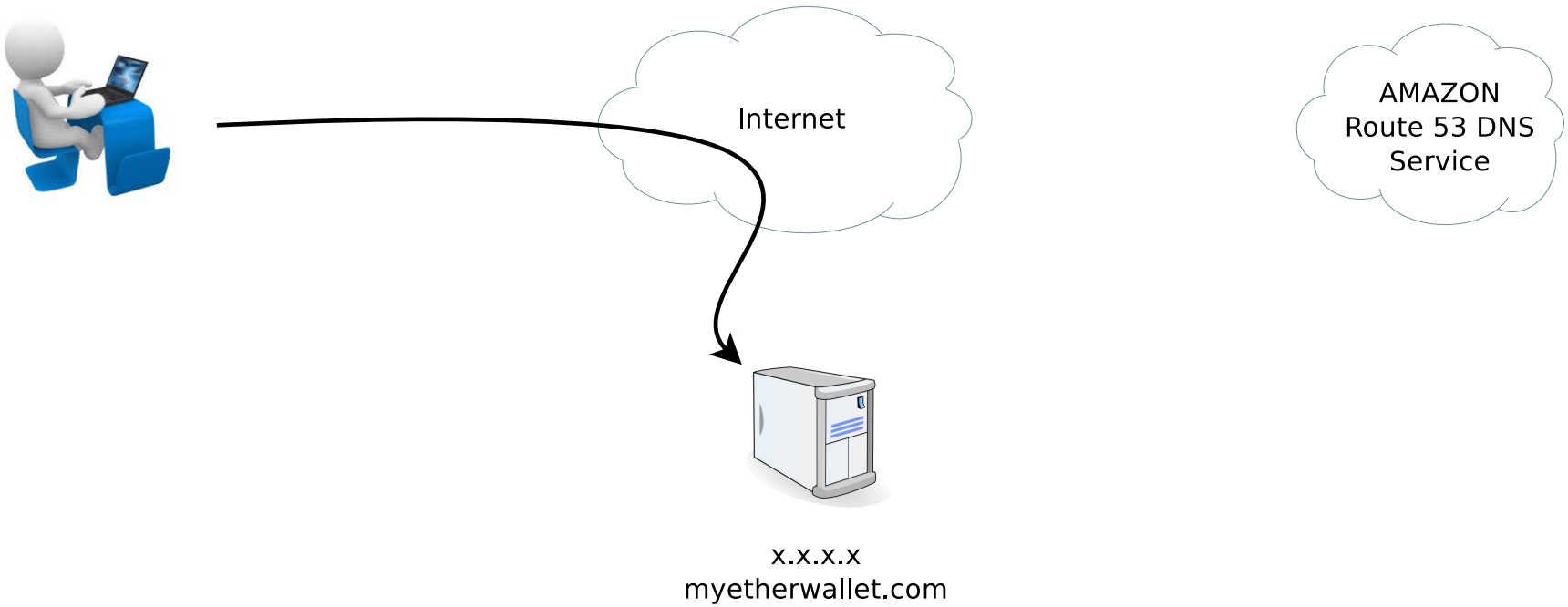
MyEtherWallet hijack

- DNS resolution for `myetherwallet.com` reaches an Amazon server running the DNS service



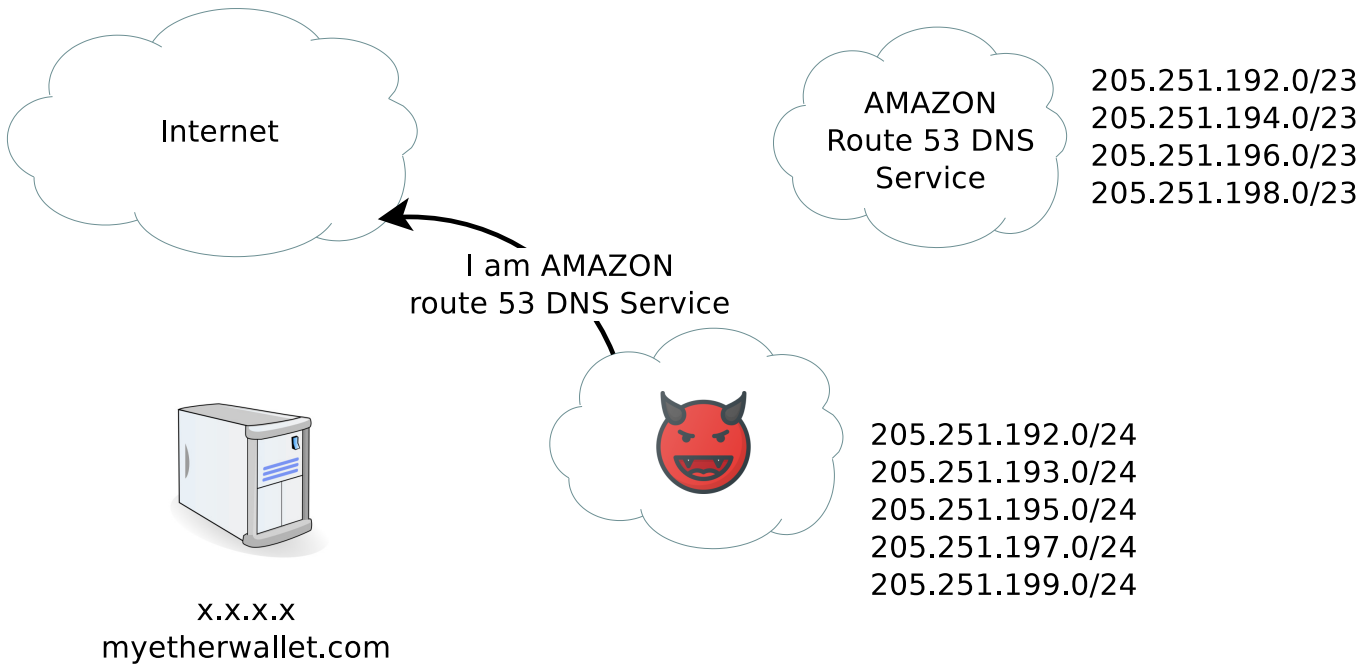
MyEtherWallet hijack

- The user then browse the legitim server hosting `myetherwallet.com`



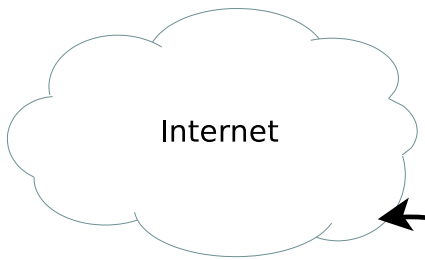
MyEtherWallet hijack

- The attacker hijacked the subnet hosting the Amazon DNS service!



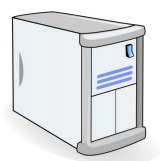
MyEtherWallet hijack

- This was a subprefix hijack, so most of the Internet believed to it!

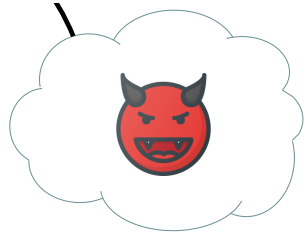


205.251.192.0/23
205.251.194.0/23
205.251.196.0/23
205.251.198.0/23

I am AMAZON
route 53 DNS Service



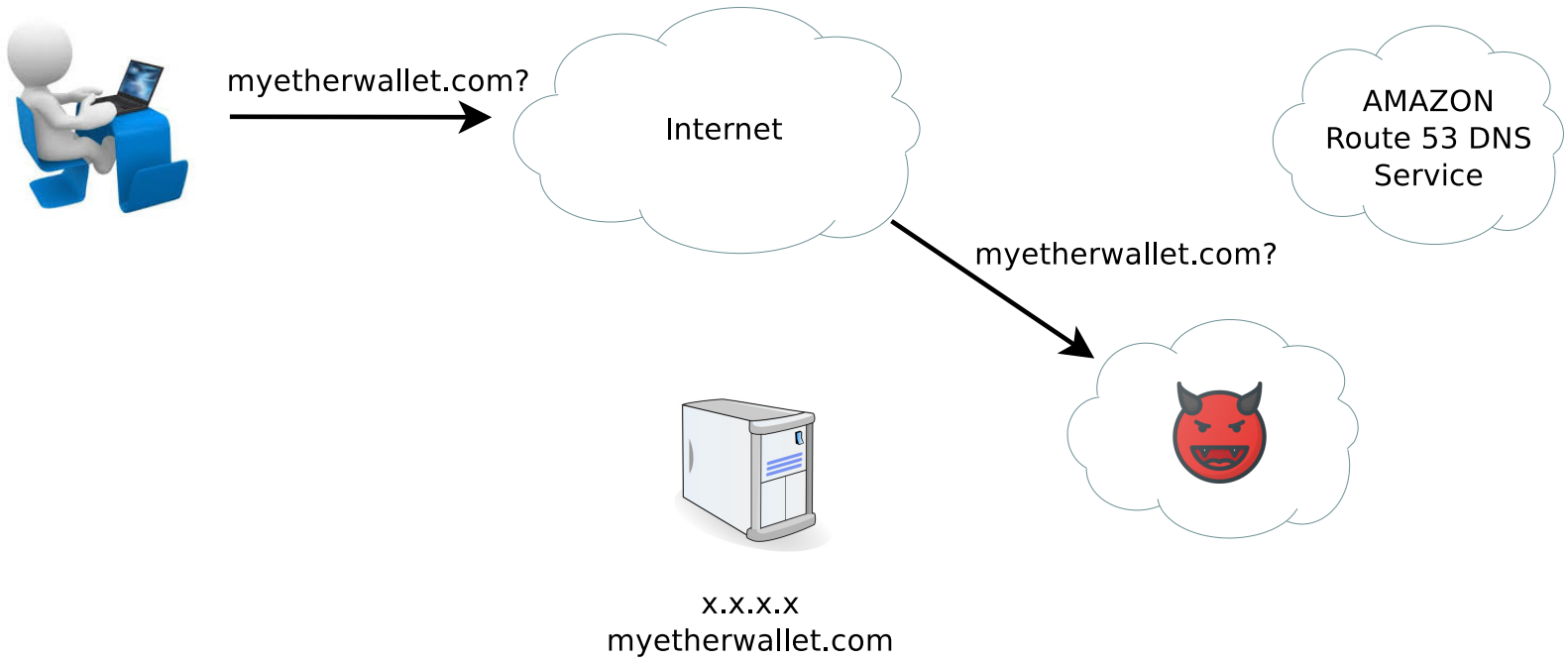
x.x.x.x
myetherwallet.com



205.251.192.0/24
205.251.193.0/24
205.251.195.0/24
205.251.197.0/24
205.251.199.0/24

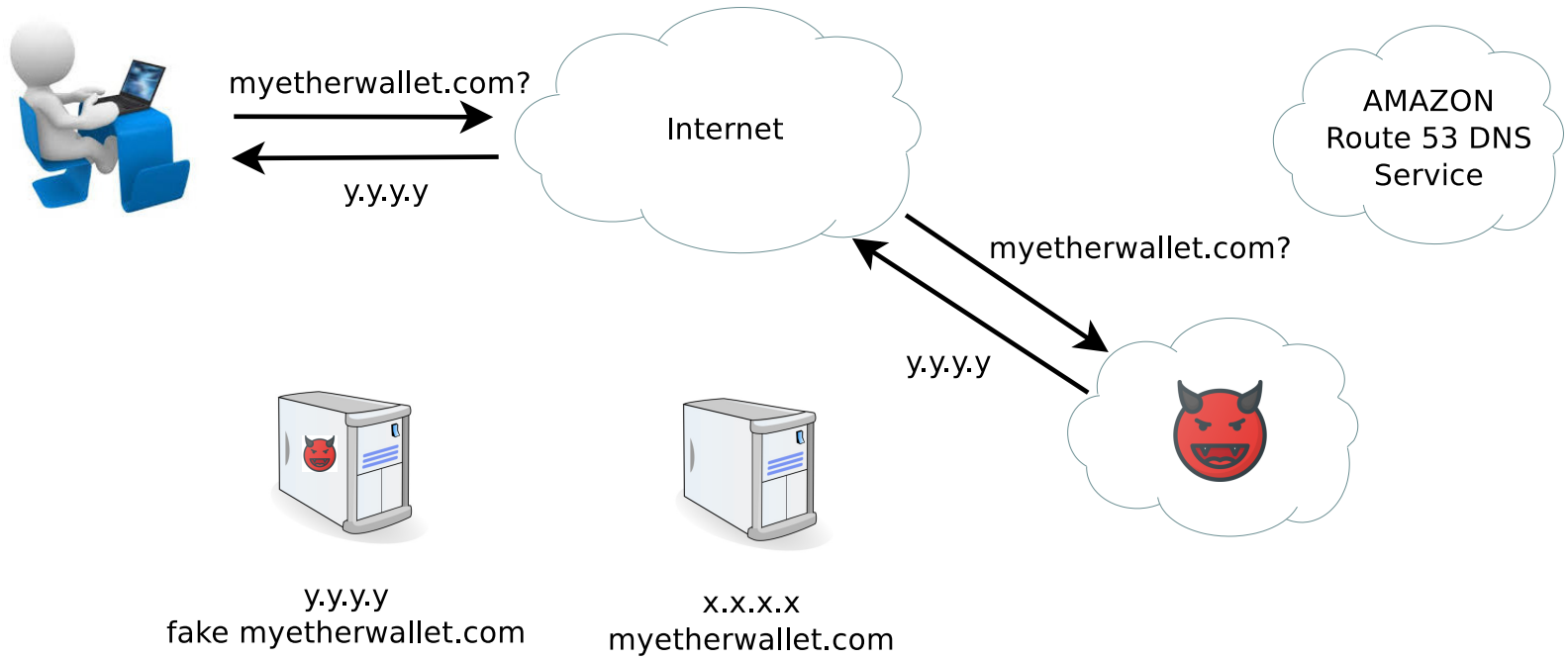
MyEtherWallet hijack

- Further requests to resolve `myetherwallet.com` were reaching the attacker network, where a server faking a DNS service was ready to answer



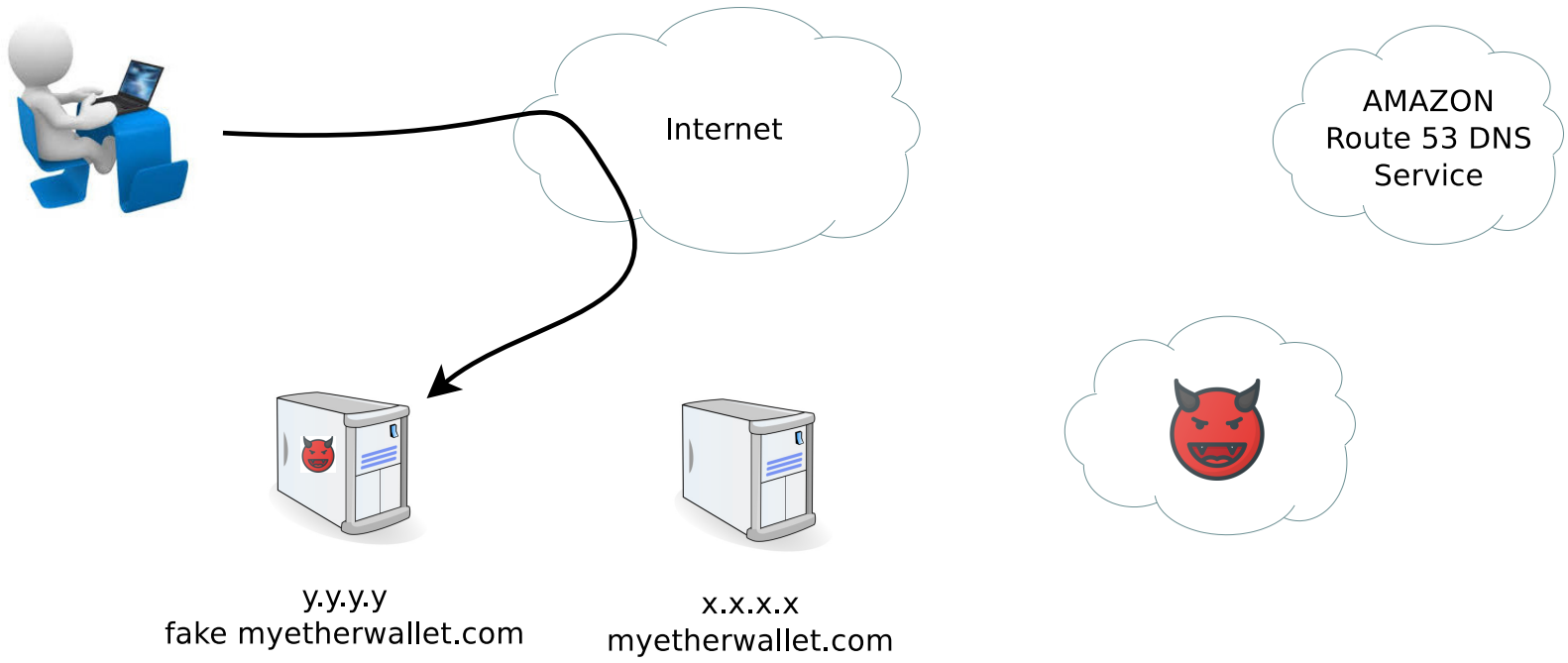
MyEtherWallet hijack

- The attacker returned an IP address in his control



MyEtherWallet hijack

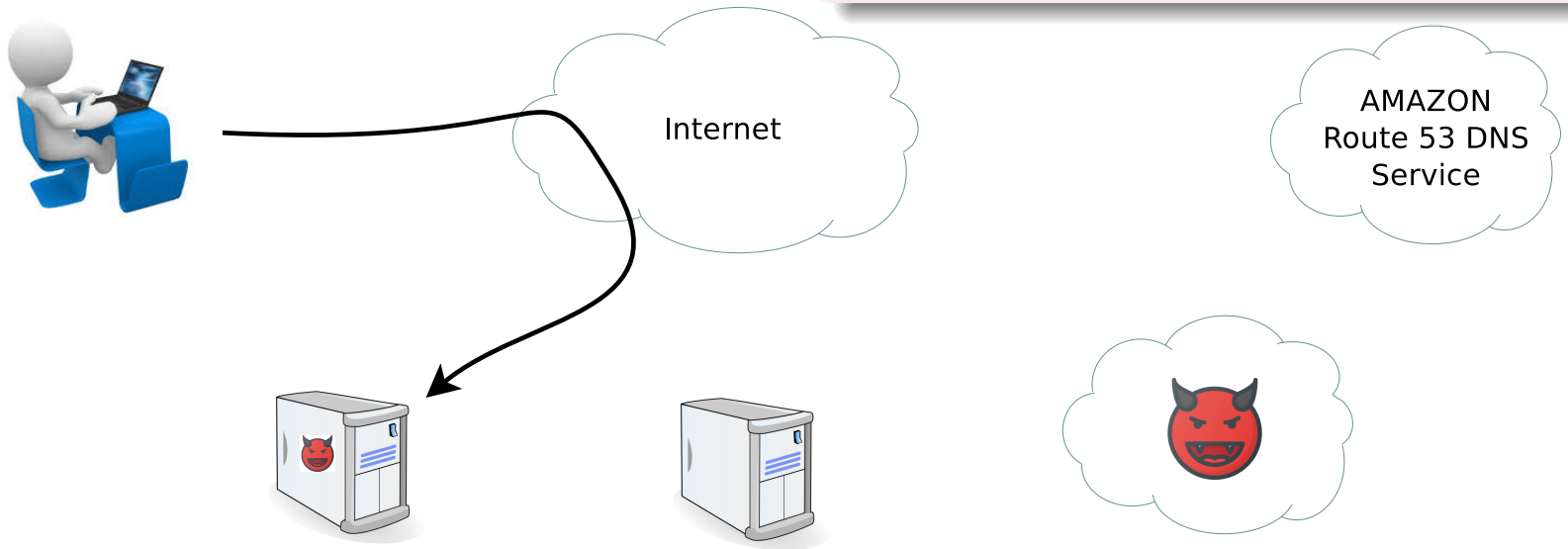
- Thus users where actually browsing on a website faking myetherwallet.com (phishing)



MyEtherWallet hijack

- Thus users where actually browsing on a website faking myetherwallet.com (phishing)

The attacker was using a self-signed TLS certificate (but that's another story!)

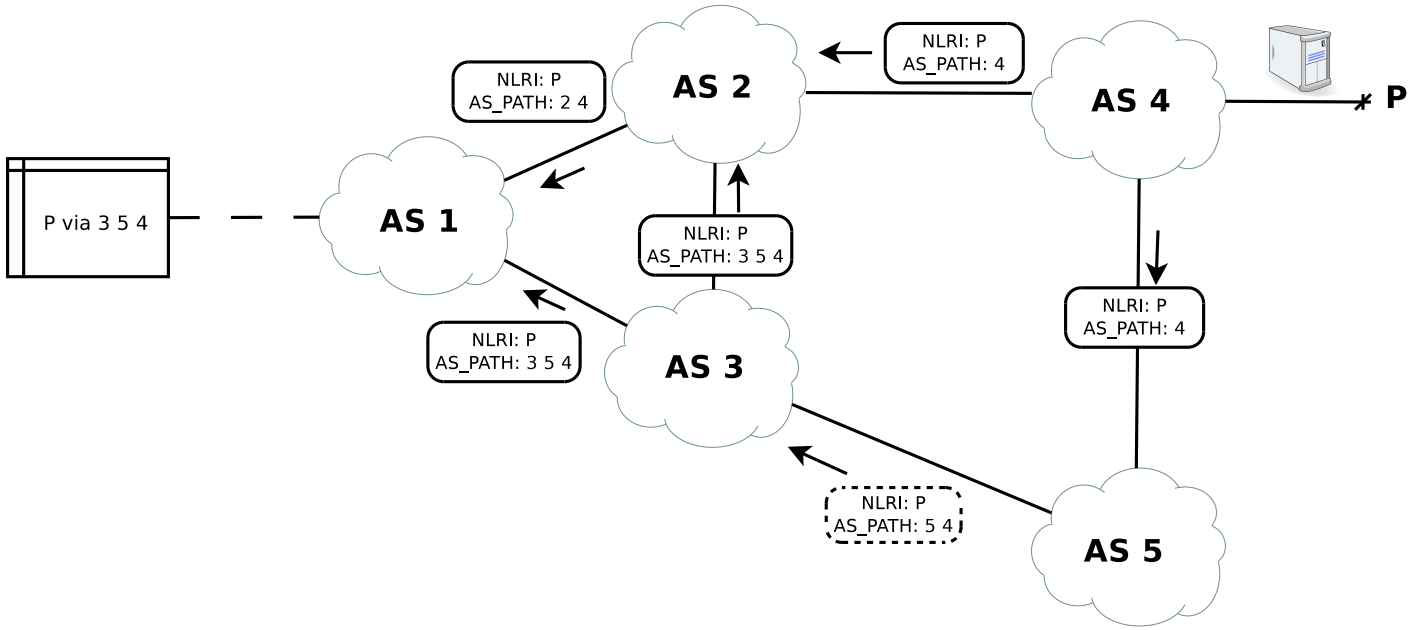


"And even though every part of my body told me not to try and log in, I did."
https://www.reddit.com/r/MyEtherWallet/comments/8ek0jj/think_i_got_scammedphishedhacked/

Route leak

Route leak

Propagation of routing announcement(s) beyond their intended scope, usually caused by misconfigurations in inbound/outbound BGP filters setup at the edge of an AS



Example

An AS propagates to a provider a route learnt from another provider

Route leaks happens often too

Internet boffins take aim at BGP route leaks

Routers should know their place

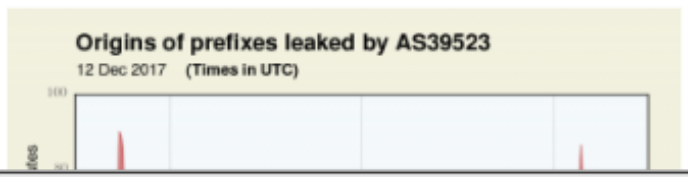
By Richard Chirgwin 19 Jun 2017 at 03:57

Large BGP Leak by Google Disrupts Internet in Japan

Research // Aug 28, 2017 // Doug Madory

DISRUPTIONS, INTERNET INTELLIGENCE | December 22, 2017

Recent Russian Routing Leak was Largely Preventable



By: Doug Madory
Director of Internet Analysis

Leaking Routes

5 Mar 2012 in **Routing**, **Security** by Geoff Huston

Its happened again.

Massive route leak causes Internet slowdown

Posted by Andree Toonk - June 12, 2015 - BGP instability - No Comments

LILY HAY NEWMAN SECURITY 11.06.17 05:00 PM

HOW A TINY ERROR SHUT OFF THE INTERNET FOR PARTS OF THE US

What can be done?

Prevention

- MANRS (Mutually Agreed Norm for Routing Security)

Detection

- Passive measurement (BGP data)
- Active measurement (ping, traceroute like tools)

Prevention - MANRS

Global initiative supported by the Internet Society (ISOC)

Best practices proposed to AS admins

- **Coordination:** Facilitating global operational communication and coordination between network operators
- **Validation:** Facilitating validation of routing information on a global scale
- **Filtering:** Preventing propagation of incorrect routing information
- **Anti-spoofing:** Preventing traffic with spoofed source IP addresses

Detection - Ping

```
$ ping 146.48.78.1
PING 146.48.78.1 (146.48.78.1) 56(84) bytes of data.
64 bytes from 146.48.78.1: icmp_seq=1 ttl=64 time=0.499 ms
64 bytes from 146.48.78.1: icmp_seq=2 ttl=64 time=0.495 ms
64 bytes from 146.48.78.1: icmp_seq=3 ttl=64 time=0.484 ms
64 bytes from 146.48.78.1: icmp_seq=4 ttl=64 time=0.429 ms
64 bytes from 146.48.78.1: icmp_seq=5 ttl=64 time=0.500 ms
64 bytes from 146.48.78.1: icmp_seq=6 ttl=64 time=0.433 ms
64 bytes from 146.48.78.1: icmp_seq=7 ttl=64 time=0.437 ms
64 bytes from 146.48.78.1: icmp_seq=8 ttl=64 time=0.431 ms
64 bytes from 146.48.78.1: icmp_seq=9 ttl=64 time=2.48 ms
64 bytes from 146.48.78.1: icmp_seq=10 ttl=64 time=0.447 ms
64 bytes from 146.48.78.1: icmp_seq=11 ttl=64 time=0.492 ms
64 bytes from 146.48.78.1: icmp_seq=12 ttl=64 time=0.448 ms
64 bytes from 146.48.78.1: icmp_seq=13 ttl=64 time=0.512 ms
64 bytes from 146.48.78.1: icmp_seq=14 ttl=64 time=0.505 ms
```

Mike Muuss, 1983

Ping is a network diagnostic tool which shows the Round Trip Time (RTT) for messages sent from the originating host to a destination and back

Anomalous TTL values can be an hint of a routing problem

Detection - Traceroute

```
$ traceroute espn.go.com
traceroute to espn.go.com (199.181.133.61), 30 hops max, 60 byte packets
 1  146.48.78.254 (146.48.78.254)  0.438 ms  0.411 ms  0.393 ms
 2  ru-cnrpi-rx1-pi1.pi1.garr.net (193.206.136.29)  0.488 ms  0.487 ms  0.472 ms
 3  rx1-pi1-rx2-mi2.mi2.garr.net (90.147.80.210)  7.303 ms  7.294 ms  7.235 ms
 4  rx2-mi2-r-mi2.mi2.garr.net (90.147.80.77)  6.896 ms  6.840 ms  6.845 ms
 5  xe-9-1-0.bar1.Milan1.Level3.net (213.242.65.81)  7.004 ms  6.989 ms  6.938 ms
 6  ae-1-6.bar2.LasVegas1.Level3.net (4.69.148.1)  158.790 ms  159.994 ms  159.978 ms
 7  ae-1-6.bar2.LasVegas1.Level3.net (4.69.148.1)  159.963 ms  158.616 ms  159.957 ms
 8  SWITCH-COMM.bar2.LasVegas1.Level3.net (205.129.18.94)  160.116 ms  161.626 ms  159.845 ms
 9  be76.las-core7-1.switchnap.com (66.209.64.105)  160.210 ms  160.137 ms  160.106 ms
10  te0-3-0-1.las-agg7s3-2.switchnap.com (66.209.72.14)  161.052 ms  160.195 ms  159.755 ms
11  * * *
12  * * *
13  * * *
14  199.181.133.61 (199.181.133.61)  160.272 ms  159.100 ms  159.018 ms
```

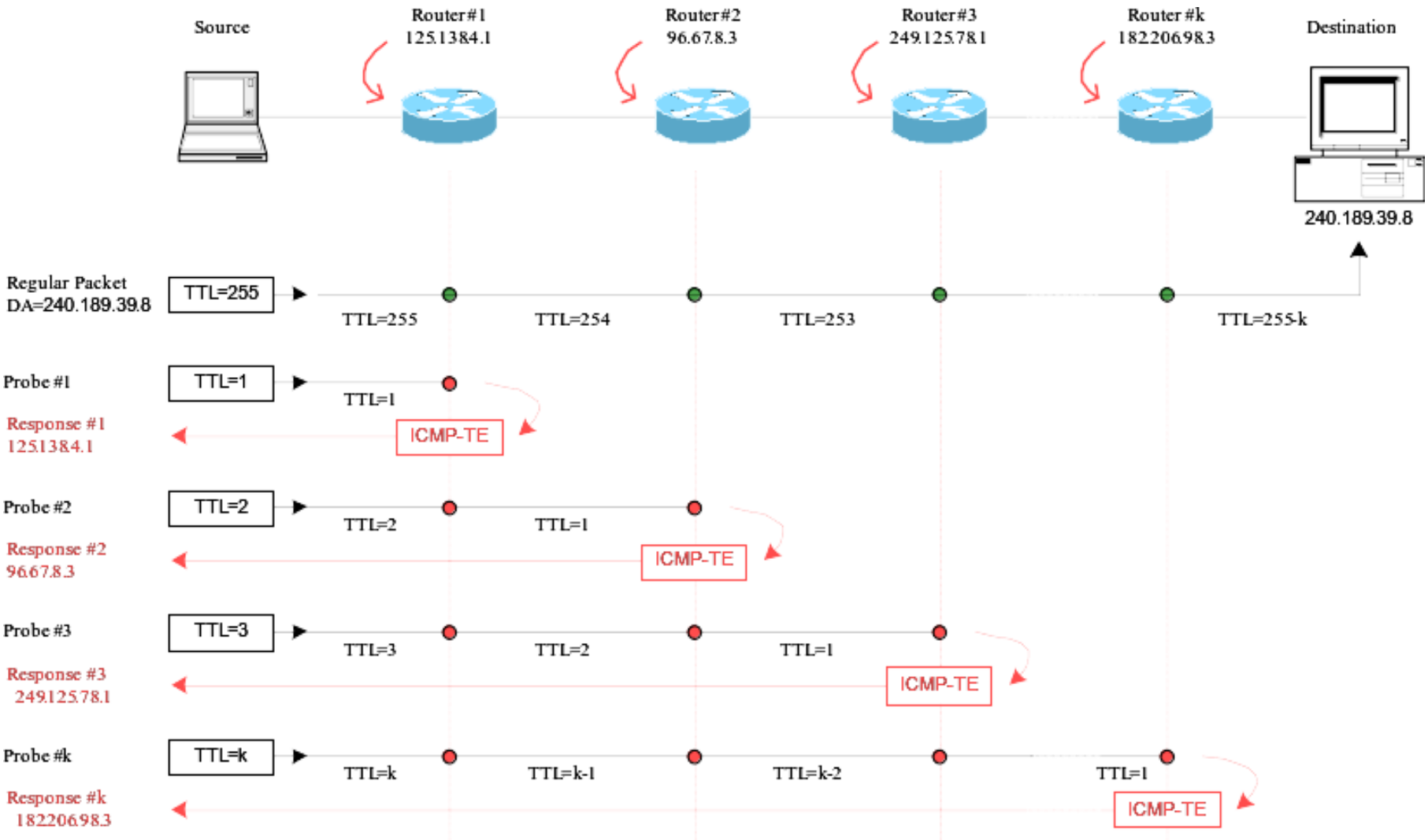
Van Jacobson, 1987

Traceroute is a network diagnostic tool which shows:

- IP path towards a destination
- RTT of the packets towards each node in the path

Anomalous IP hops patterns can highlight a routing problem

Traceroute


Source
Router#1
125.13&4.1
Router#2
96.67.8.3
Router#3
249.125.7&1
Router #k
182.206.98.3
Destination
240.189.39.8
Regular Packet
DA=240.189.39.8
TTL=255
TTL=255
TTL=254
TTL=253
TTL=255-k
Probe #1
TTL=1
TTL=1
Response #1
125.13&4.1
ICMP-TE
Probe #2
TTL=2
TTL=2
TTL=1
Response #2
96.67.8.3
ICMP-TE
Probe #3
TTL=3
TTL=3
TTL=2
TTL=1
Response #3
249.125.78.1
ICMP-TE
Probe #k
TTL=k
TTL=k
TTL=k-1
TTL=k-2
TTL=1
Response #k
182.206.98.3
ICMP-TE

Traceroute monitoring projects

CAIDA Archipelago (Ark)

The Center for Applied Internet Data Analysis (CAIDA) in San Diego, US deploys and maintains a globally distributed measurement platform called Archipelago (Ark) which collects IP level routing data since 2007
http://www.caida.org/projects/ark/

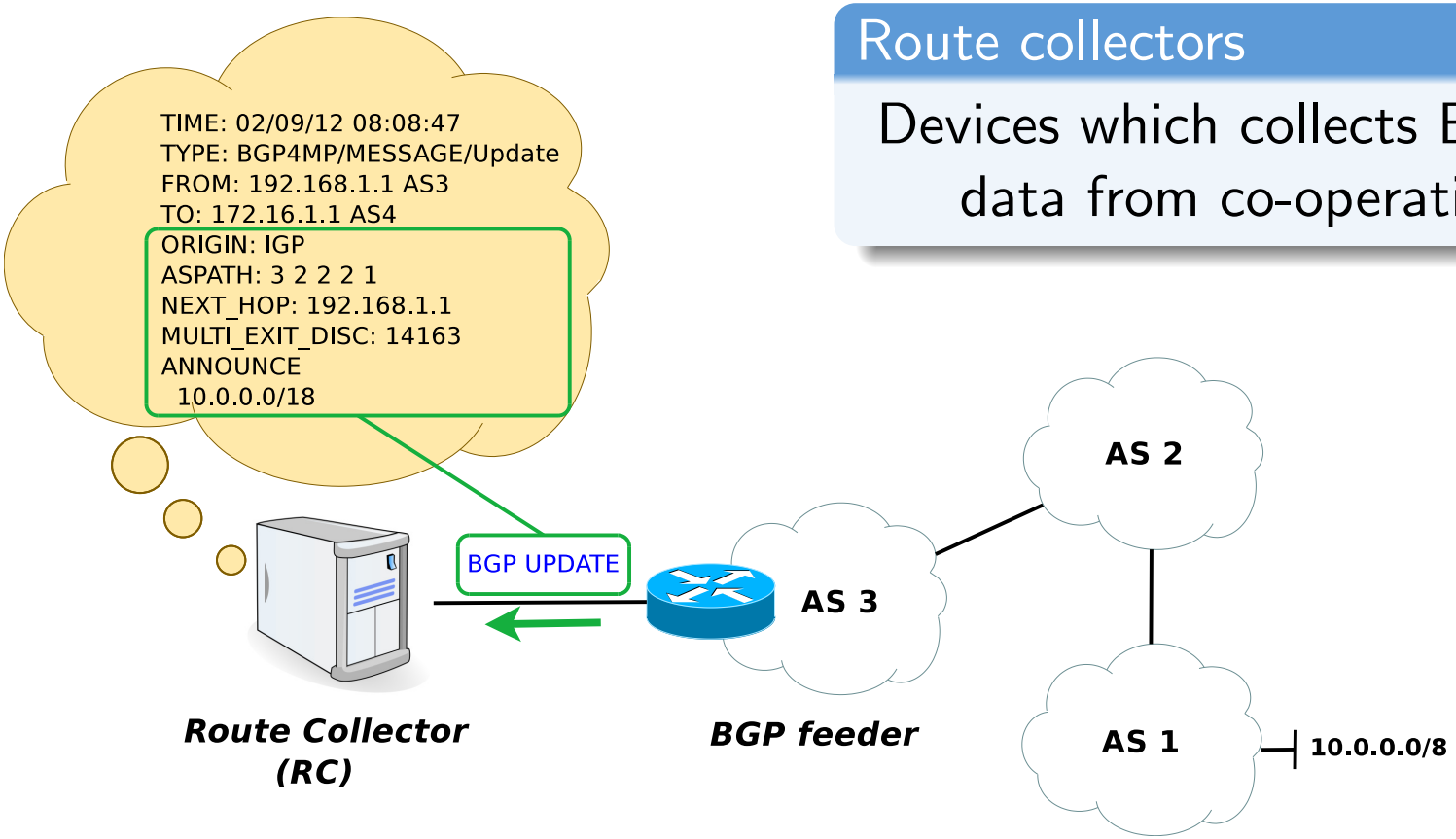
RIPE NCC ATLAS

The RIPE NCC deployed thousands of active probes in the RIPE Atlas network, concentrated in the RIPE NCC's service region of Europe, the Middle East and parts of Central Asia. The RIPE NCC collects the data from this network and provides useful maps and graphs based on the aggregated results. RIPE Atlas users who host a probe can also use the entire RIPE Atlas network to conduct customised measurements that provide valuable data about their own network. No repository is dedicated to scanning the whole IPv4 space, but every measurement made is recorded and publicly available
https://atlas.ripe.net/

Portolan

Portolan is a research project run by the University of Pisa and the Informatics and Telematics Institute of the Italian National Research Council (IIT-CNR). Portolan's aim is to enhance the current knowledge of the Internet structure and build maps of the mobile signal coverage, all with the contribution of volunteers. Data is available since 2013
http://portolanproject.iit.cnr.it/

Detection - BGP route collectors



Route collectors

Devices which collects BGP routing data from co-operating ASes

- Multi-Threaded Routing Toolkit format (RFC 6396)
- Maintains a routing table (RIB) with the best routes received
- Dumps the content of the RIB and received UPDATEs periodically

BGP route collectors

University of Oregon Route Views Project

Route Views was conceived as a tool for Internet operators to obtain real-time information about the global routing system from the perspectives of several different backbones and locations around the Internet. It collects BGP packets in MRT format since 2001
http://www.routeviews.org

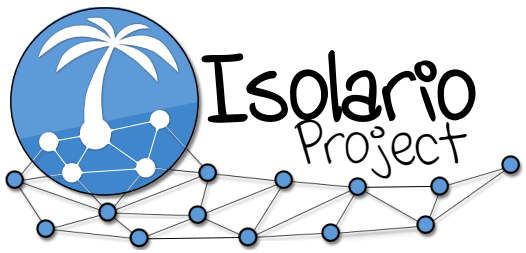
RIPE NCC Routing Information Service (RIS)

The RIPE NCC collects and stores Internet routing data from several locations around the globe, using RIS. It collects BGP packets in MRT format since 1999
https://www.ripe.net/analyse/internet-measurements/routing-information-service-ris

Packet Clearing House (PCH)

PCH is the international organization responsible for providing operational support and security to critical Internet infrastructure, including Internet exchange points and the core of the domain name system. It operates route collectors at more than 100 IXPs around the world and its data is made available in MRT format since 2011
https://www.pch.net/resources/Raw_Routing_Data

PCH
Packet Clearing House



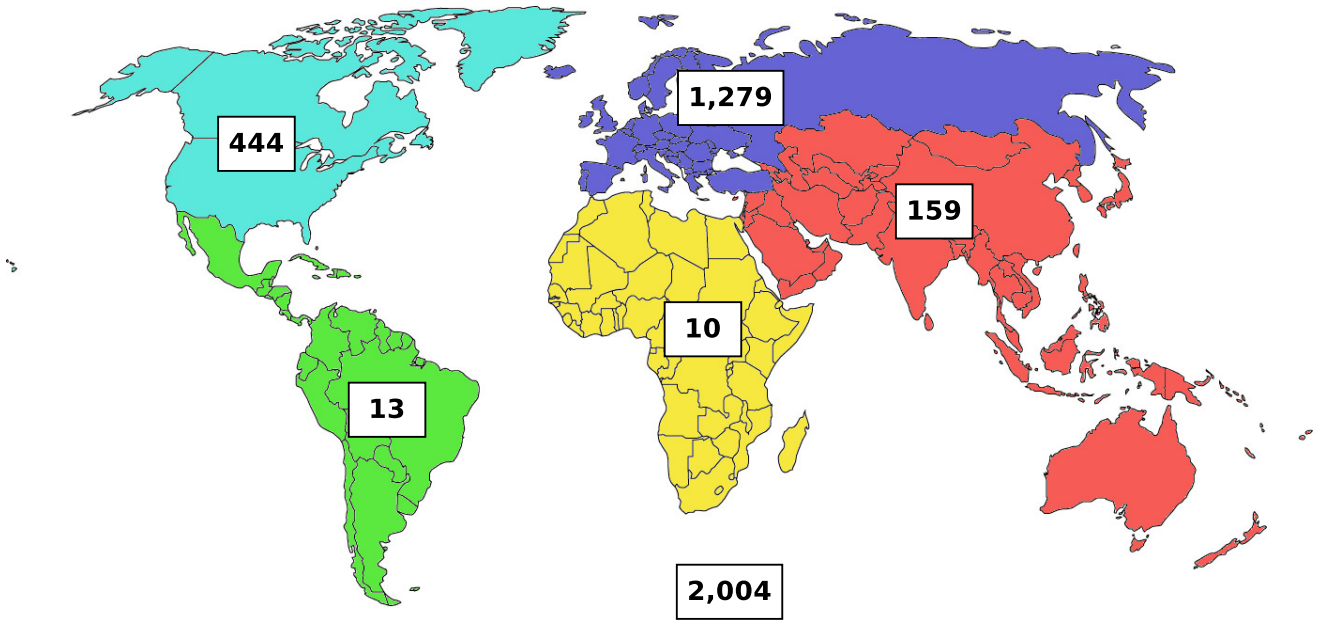
Isolario

Isolario is a route collecting project which provides inter-domain real-time monitoring services to its participants. It collects BGP packets in MRT format since 2013, and supports ADDPATH (RFC 7911) since 2018
https://www.isolario.it

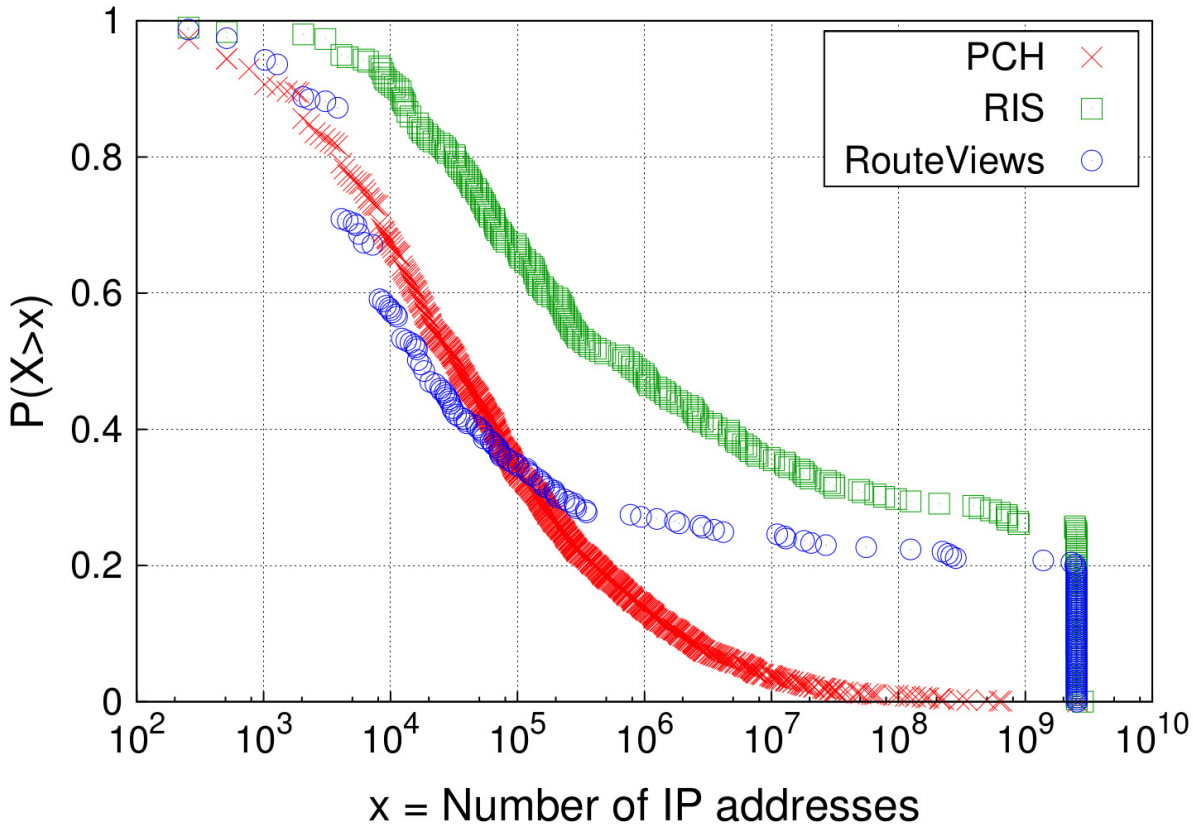
ICE: an Interactive Collector Engine

BGP route collector status (Feb 2012)

	RouteViews	RIS	PCH
N. of RC	10	13	51
N. of feeders	313	299	1,842



Feeder contribution



Only 120 feeders announce to the RCs their full routing table

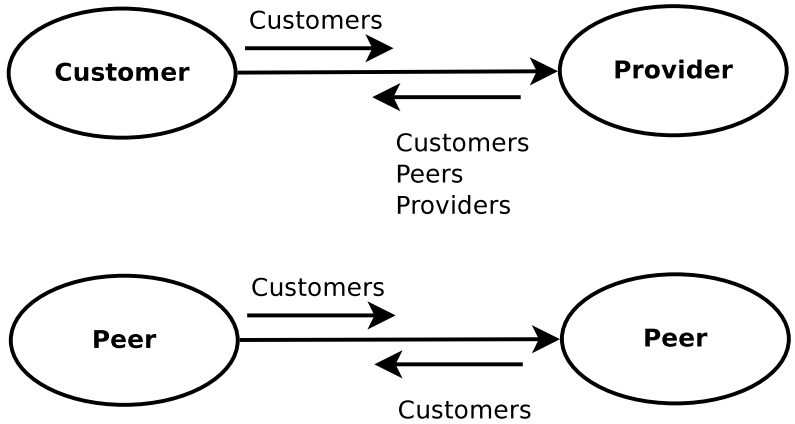
Export policies

BGP connectivity does not imply IP reachability

Two ASes establishes a BGP session and agree for which networks advertise reachability each-other

Typical roles

Provider: an AS that advertises reachability for all Internet networks
Peer: an AS that advertises reachability only for its customer cone networks
Customer: an AS that advertises reachability only for its networks



- PCH establishes only p2p sessions
- Most RCs are placed on IXPs

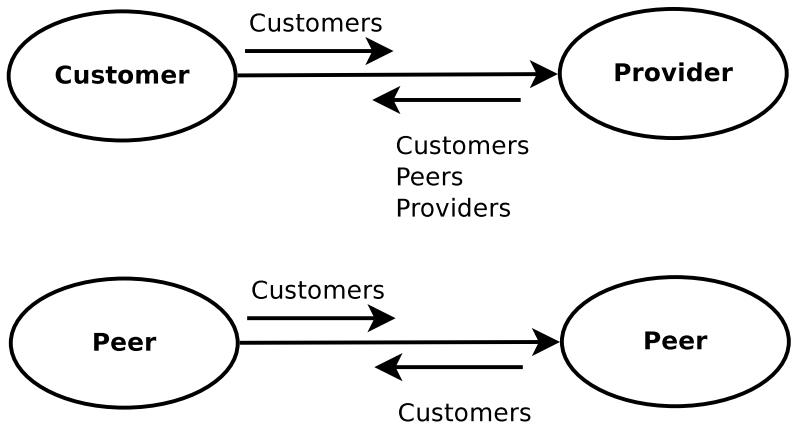
Export policies

BGP connectivity does not imply IP reachability

Two ASes establishes a BGP session and agree for which networks advertise reachability each-other

Typical roles

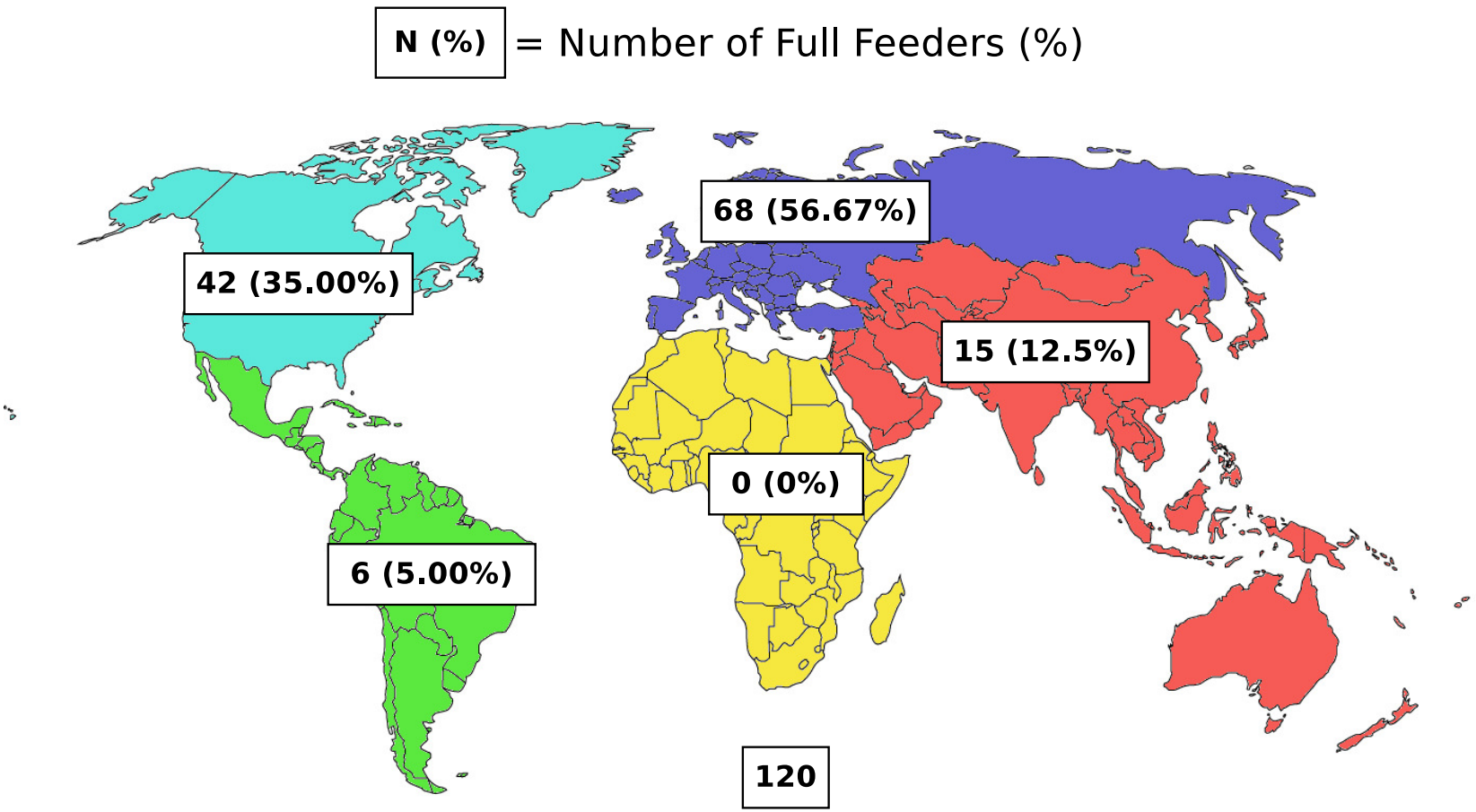
Provider: an AS that advertises reachability for all Internet networks
Peer: an AS that advertises reachability only for its customer cone networks
Customer: an AS that advertises reachability only for its networks



- RCs need to be considered as *customers* by their feeders to gather a full routing table

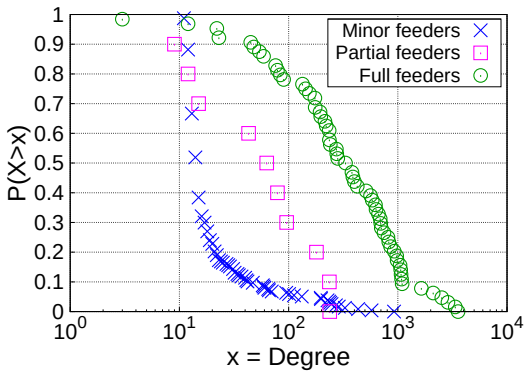
Full feeder geographical distribution

Data collected represent mostly the Internet as viewed from Europe and North America than the real Internet

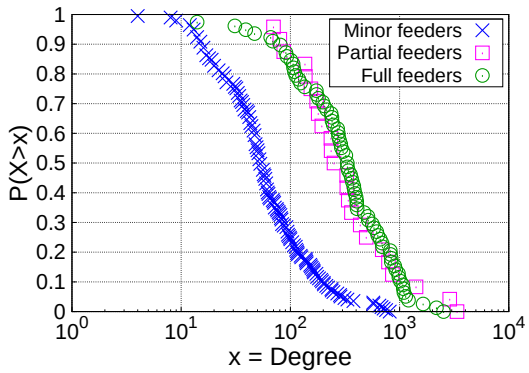


Feeder characterization

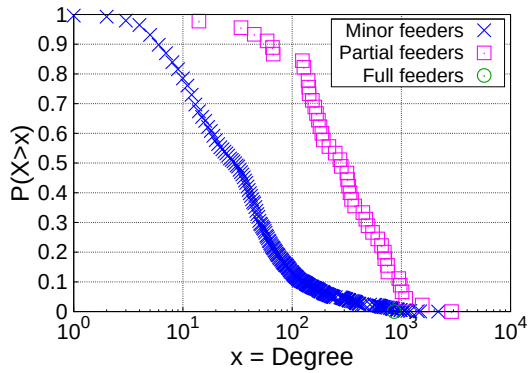
RouteViews



RIS

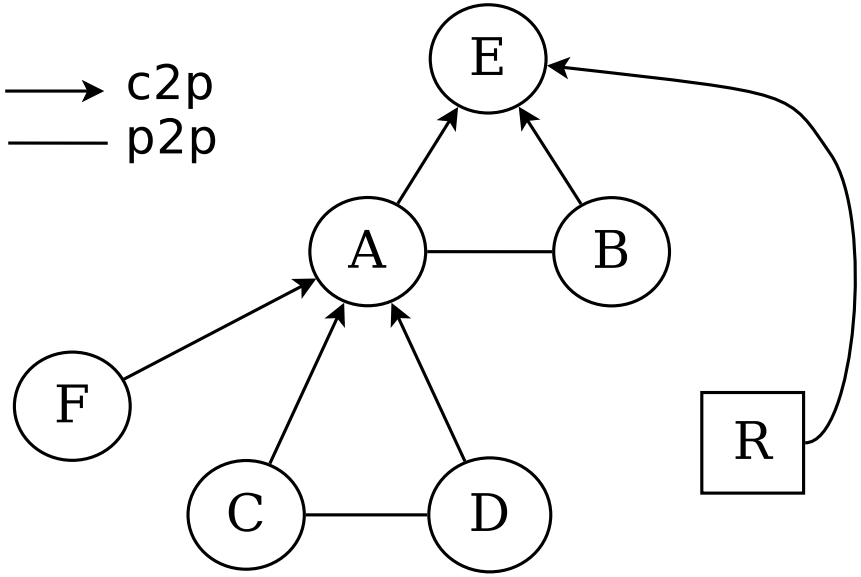


PCH



About 80% of full feeders have a degree higher than 100

A view from the top

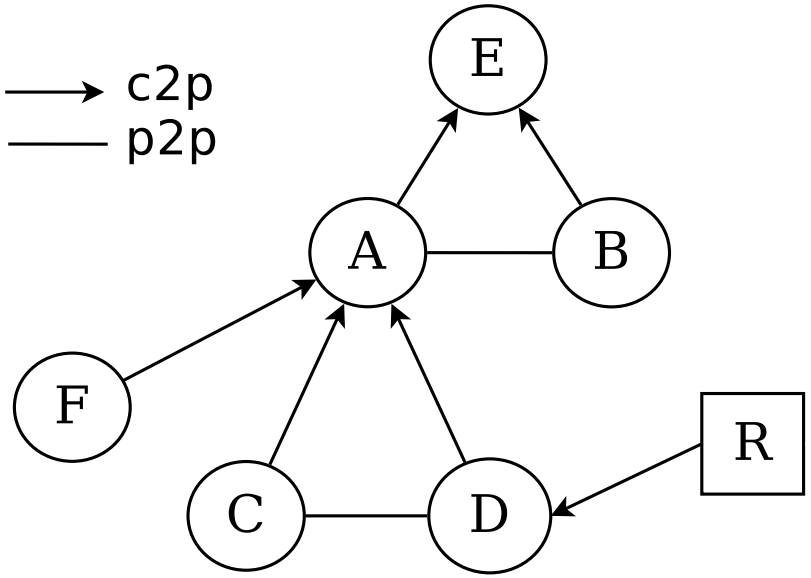


Connections that can be discovered

(A, C) (A, D) (A, E) (A, F) (B, E)

RCs connected to large ISPs will fail to retrieve a large amount of p2p-connectivity

A view from the bottom



Connections that can be discovered

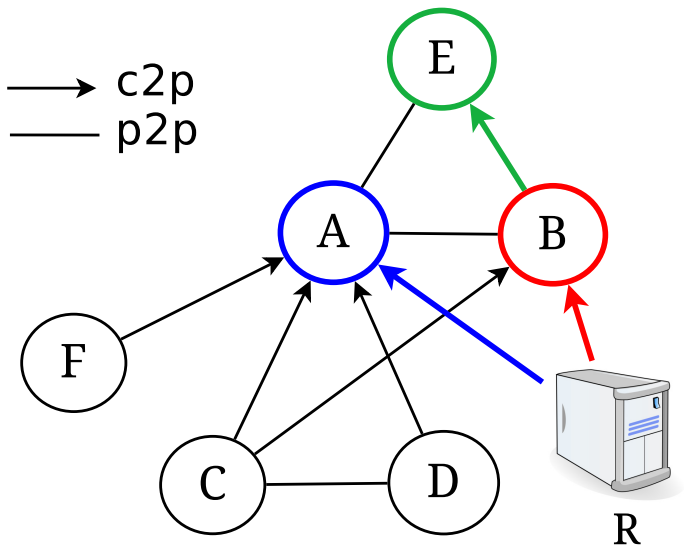
(A, B) (A, C) (A, D) (A, E) (A, F) (B, E) (C, D)

RCs need to be connected to ASes part of the lowest part of the Internet hierarchy to discover the missing p2p connectivity

A new metric: c2p-distance

c2p-distance of AS X from AS Y:

Minimum number of consecutive c2p links that connect X to Y



AS	c2p-distance from R
A	1
B	1
C	-
D	-
E	2
F	-

Farther an AS is from a RC, the greater are the chances to lose AS-level connectivity due to BGP decision processes

Tailored set covering problem

Set Covering

$$\text{Minimize} \quad \left(\sum_{AS_i \in \mathcal{U}} x_{AS_i} \right) \tag{1}$$

subject to

$$\sum_{AS_i : n \in S^{(d)}_{AS_i}} x_{AS_i} \geq 1 \qquad \forall n \in \mathcal{N} \tag{2}$$

$$x_{AS_i} \in \{0, 1\}, \qquad \forall AS_i \in \mathcal{U} \tag{3}$$

Goal rephrased

Select new BGP feeders such that each non-stub AS has a **finite and bounded** c2p-distance from the route collector infrastructure

Tailored set covering problem

Set Covering

$$\text{Minimize} \quad \left(\sum_{AS_i \in \mathcal{U}} x_{AS_i} \right) \tag{1}$$

subject to

$$\sum_{AS_i : n \in S^{(d)}_{AS_i}} x_{AS_i} \geq 1 \qquad \forall n \in \mathcal{N} \tag{2}$$

$$x_{AS_i} \in \{0, 1\}, \qquad \forall AS_i \in \mathcal{U} \tag{3}$$

MSC: NP-complete problem

The scarcity of the matrix allowed us to solve the problem exploiting the concepts of **dominance** and **essentiality**.

[1] E. J. McCluskey, *"Minimization of Boolean Functions"*, Bell System Technical Journal, vol. 35(6), pp. 1417–1444, 1956
[2] W. V. Quine, *"A Way to Simplify Truth Functions"*, The American Mathematical Monthly, vol. 62(9), pp. 627–631, 1955
[3] –, *"On Cores and Prime Implicants of Truth Functions"*, The American Mathematical Monthly, vol. 66(9), pp. 755–760, 1959

Tailored set covering problem

Set Covering

$$\text{Minimize} \quad \left(\sum_{AS_i \in \mathcal{U}} x_{AS_i} \right) \tag{1}$$

subject to

$$\sum_{AS_i : n \in S^{(d)}_{AS_i}} x_{AS_i} \geq 1 \qquad \forall n \in \mathcal{N} \tag{2}$$

$$x_{AS_i} \in \{0, 1\}, \qquad \forall AS_i \in \mathcal{U} \tag{3}$$

Output

- One optimal solution $\mathcal{P}$
- Set of candidates interchangeable with ASes in $\mathcal{P}$

Real world analysis

Distance parameter

- $d_{c2p} = 2$: to obtain the best quality result without the need to establish a connection with every non-stub ASes

Economic topologies

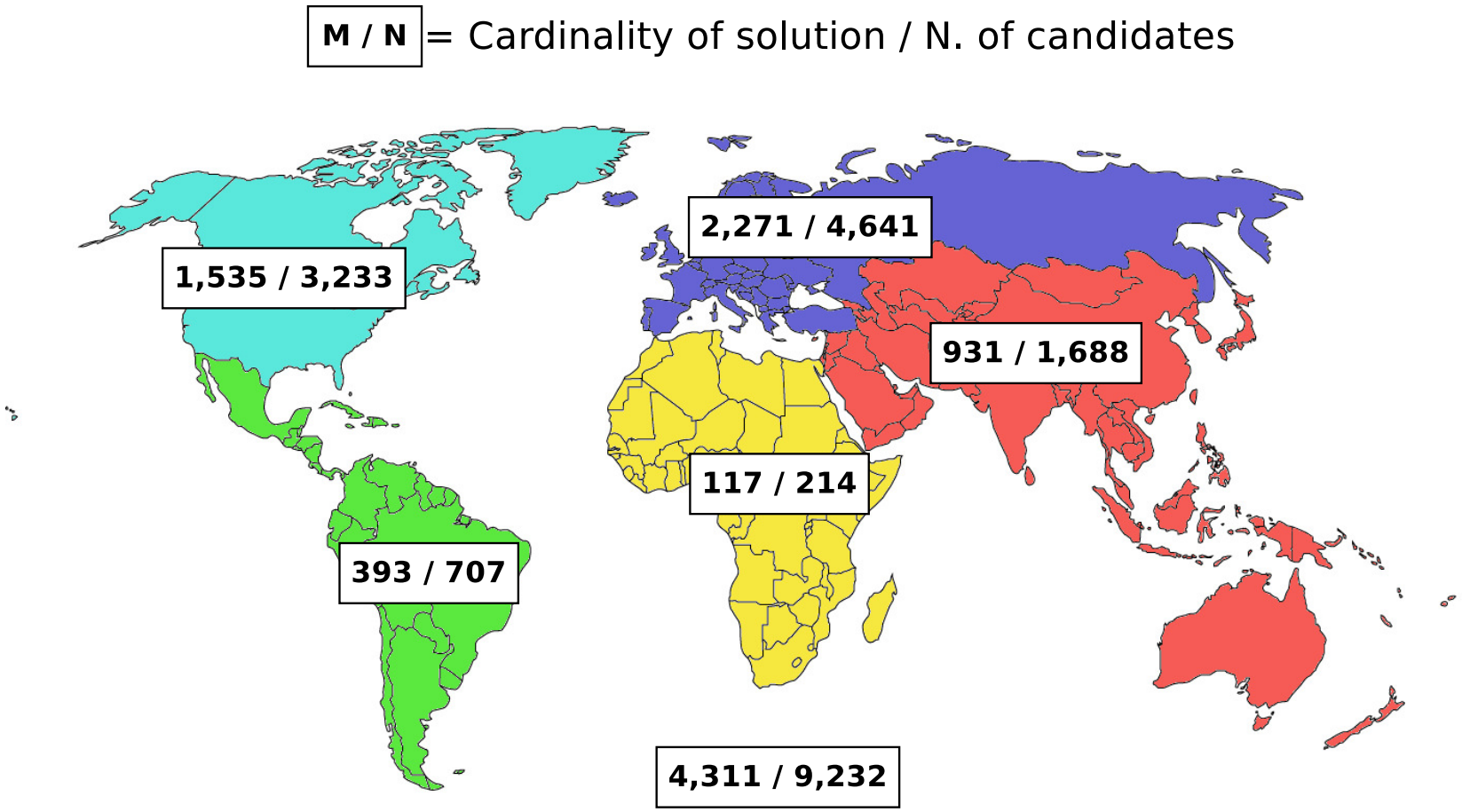
- Global [4]
- Continental [5]

Scenarios

- Empty scenario:
  - current full feeders are ignored
- Full feeders scenario:
  - current full feeders are part of the solution set

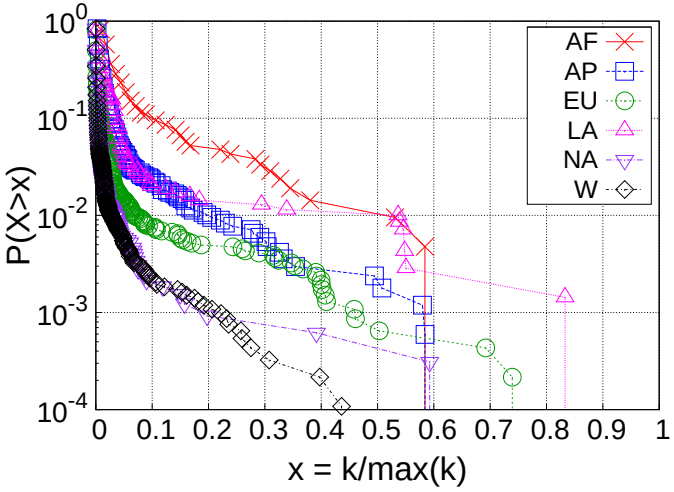
[4] E. Gregori et al., *"BGP and Inter-AS Economic Relationships"*, IFIP Networking 2011, pp. 54-67
[5] E. Gregori et al., *"Inferring Geography from BGP Raw Data"*, IEEE INFOCOM NetSciCom, INFOCOM 2012, pp. 208-213

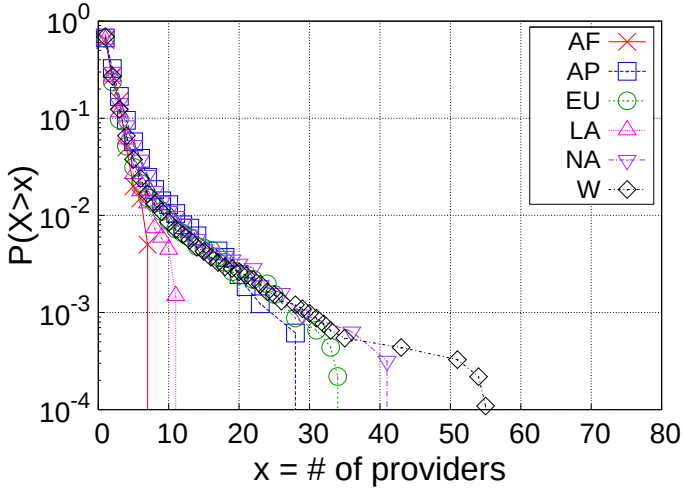
Empty scenario



The number of feeders required heavily outnumbers the current number of (full) feeders

Candidate feeder details

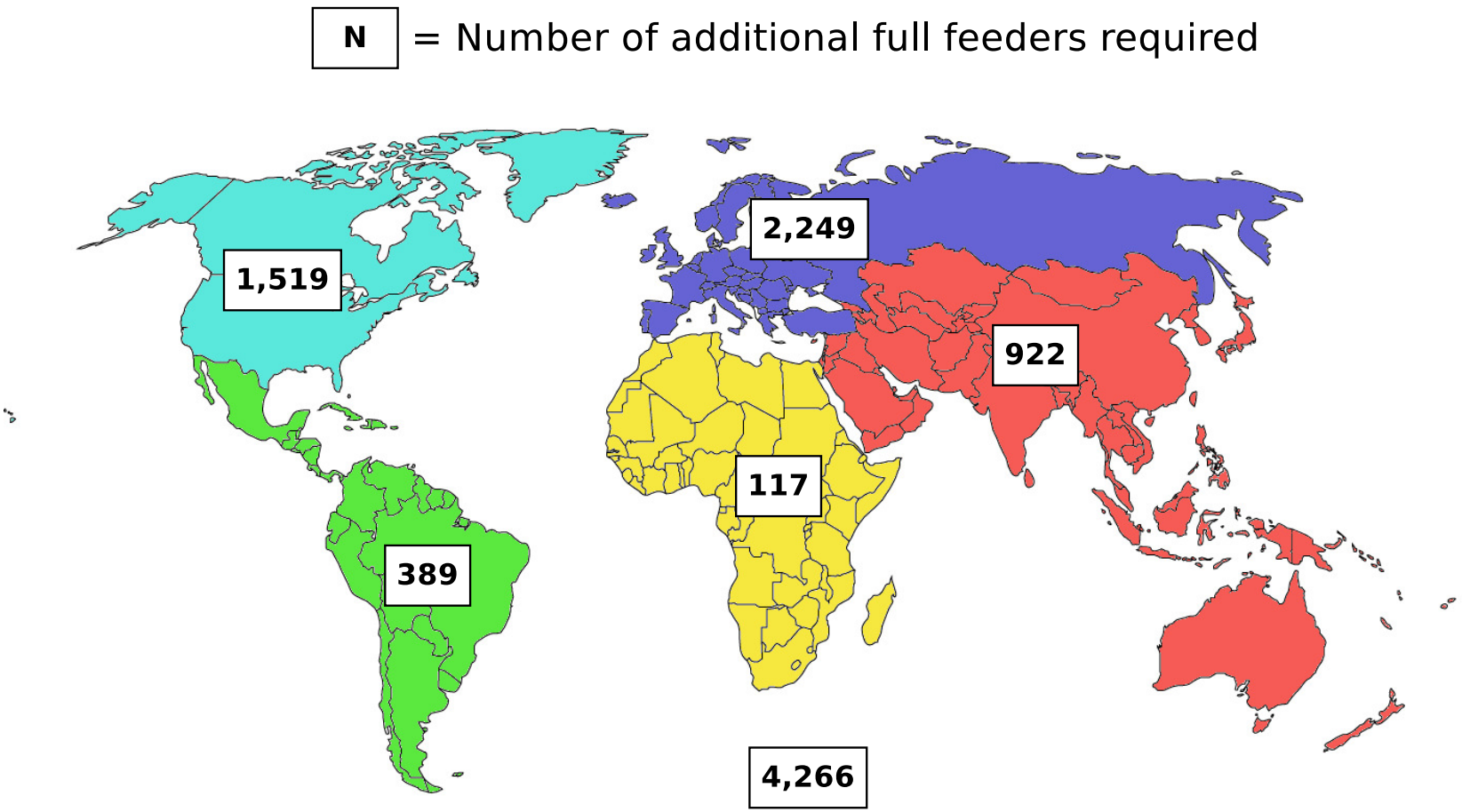




Region	|*Candidates*|	
	On IXPs	*Stubs*
AF	27 (12.79 %)	114 (54.03 %)
AP	472 (28.04 %)	942 (55.97 %)
EU	1,931 (41.60 %)	2,250 (48.48 %)
LA	204 (29.14 %)	394 (56.29 %)
NA	406 (12.55 %)	1,509 (46.67 %)
W	2,944 (31.88 %)	4,221 (45.72 %)

These are exactly the ASes that rarely feed the RCs

Full feeder scenario



The introduction of full feeders in the solution set do not improve much the situation

How to get a ranking of the most useful candidates?

Maximum coverage

$$\text{Maximize} \quad \left(\sum_{AS_j \in \mathcal{N}} y_{AS_j} \right) \tag{4}$$

subject to

$$\sum_{AS_i \in \mathcal{I}} x_{AS_i} \leq k \tag{5}$$

$$\sum_{AS_i \in \mathcal{I} \wedge AS_j \in S_{AS_i}} x_{AS_i} \geq y_{AS_j}, \qquad \forall AS_j \in \mathcal{N} \tag{6}$$

$$y_{AS_j} \in \{0, 1\}, \qquad \forall AS_j \in \mathcal{N} \tag{7}$$

$$x_{AS_i} \in \{0, 1\}, \qquad \forall AS_i \in \mathcal{U} \tag{8}$$

MC: NP-hard problem

Solved thanks to a **greedy** heuristic and the concept of **dominance**

[6] Gregori et al., *"A Novel Methodology to Address the Internet AS-level Data Incompleteness"*, IEEE/ACM Transactions on Networking, pp. 1314-1327, Vol. 23(4), 2015

Conclusions on BGP data available

Conclusions

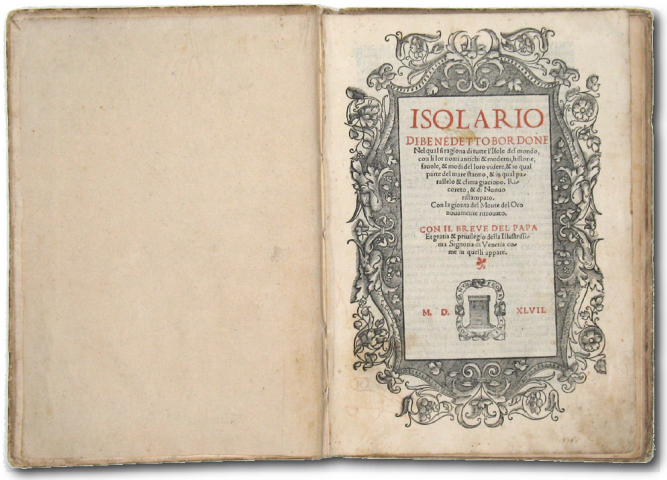
- Several p2p-connectivity is hidden from route collectors
- Several Internet regions are basically uncovered
- IXP role is thus heavily underestimated

- The typical profile of an ideal feeder is a multi-homed stub AS
- Ideal feeders are not connected to route collectors
- Too few feeders are connected to route collectors

Isolario project

Objective: push more ASes to join

The more the ASes, the more the completeness of public BGP data



Isolario - The Book of Islands

"where we discuss about all islands of the world, with their ancient and modern names, histories, tales and way of living..."

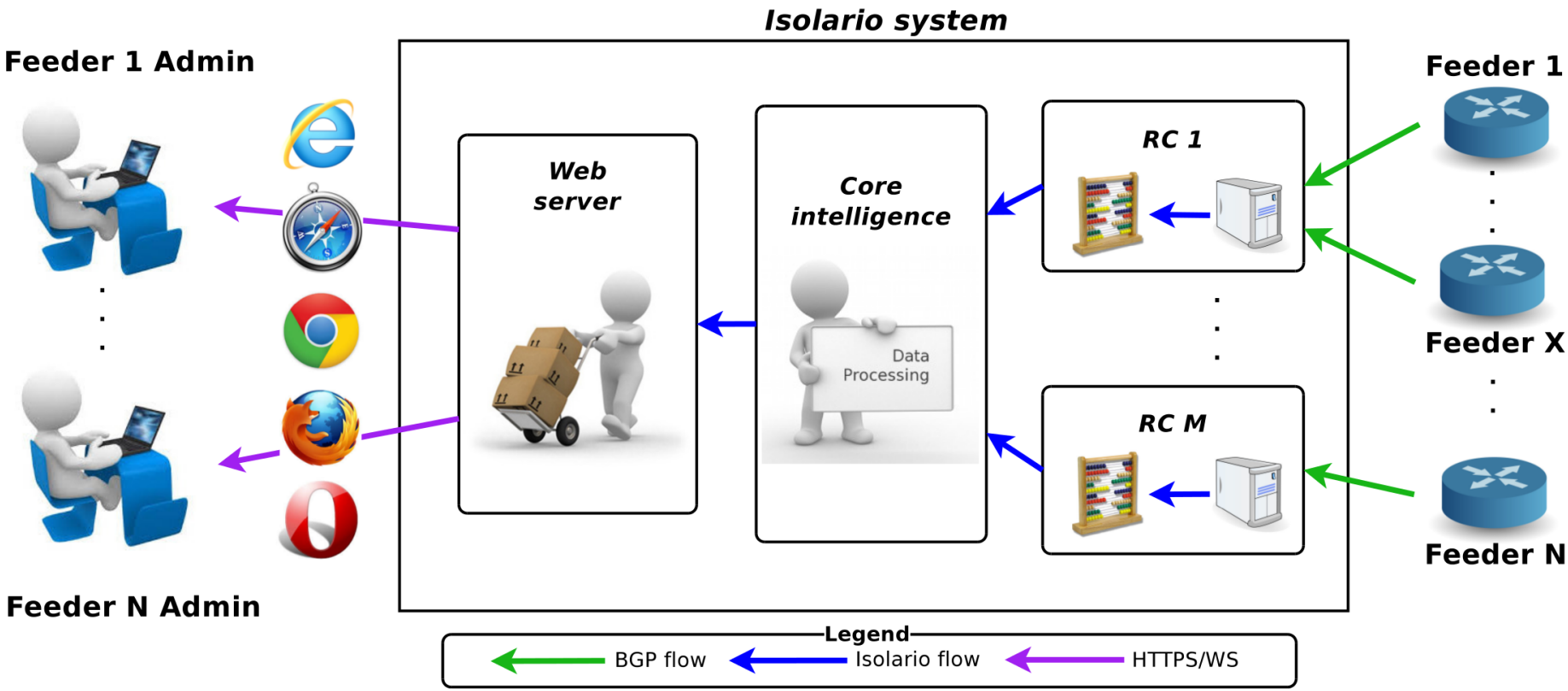
Benedetto Bordone
(Italian cartographer)

Approach: Do-ut-des

- Participants open at least one v4/v6 BGP session with Isolario providing their **full** routing table
- In change, Isolario offers **real-time** and **offline** applications based on the aggregation of every routing information collected

Isolario system overview

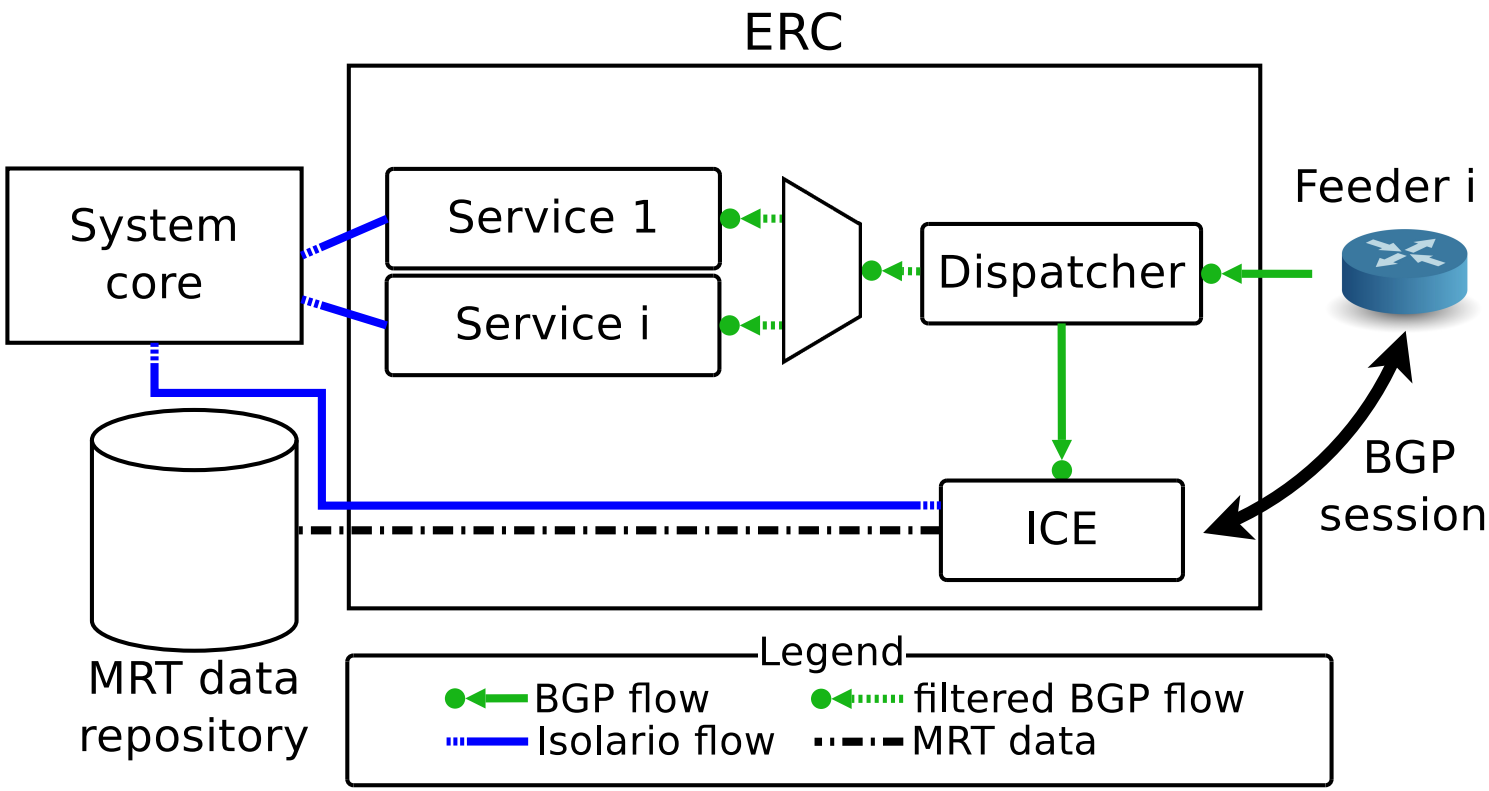
Incoming BGP flows are used as **real-time streams** for services dedicated to participants



Results are provided to users via WebSockets (RFC 6455)

Enhanced BGP route collector

Incoming flows are duplicated as soon as they arrive and feed both the Route Collecting Software (RCS) and service modules



As usual, RCs only collect routing information and **not** user traffic

Isolario free services for feeders

Every feeder has **free** access to a set of services tailored to monitor and analyse BGP data coming into Isolario system

Real-time services

- BGP flow viewer
- Routing table viewer
- Subnet reachability
- Website reachability

Historic services

- Routing table viewer
- Subnet reachability

Diagnostic services

- Alerting system
- Daily report

Feel free to try isolario.it

Username: *guest*

Password: *guest*

Isolario free services for feeders

BGP flow viewer

`https://youtu.be/QynZqNMCyXw`

Subnet reachability

`https://youtu.be/uTrLVv1PoPo`

Alerting system

`https://youtu.be/p_r2pRHK7EI`

Summary: how to use Isolario?

Real-time services

Something is happening

How is my RIB(s) evolving?

How is my reachability affected?

Historic services

Something happened

How was my RIB(s) evolving?

How was my reachability affected?

Alerting System

Something is happening NOW!

Check real-time services!

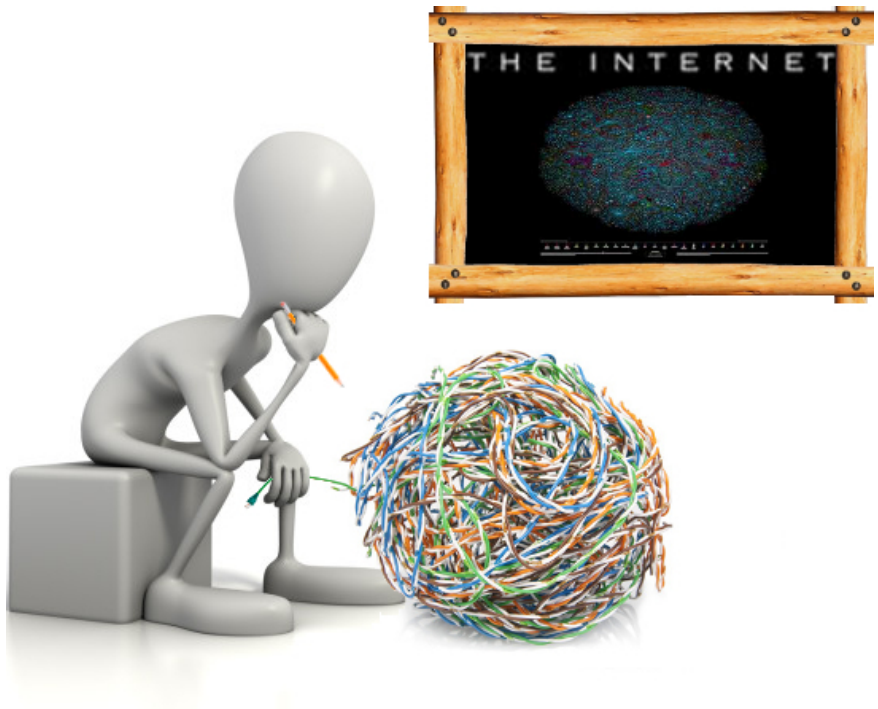
Do something! (if needed)

Daily report

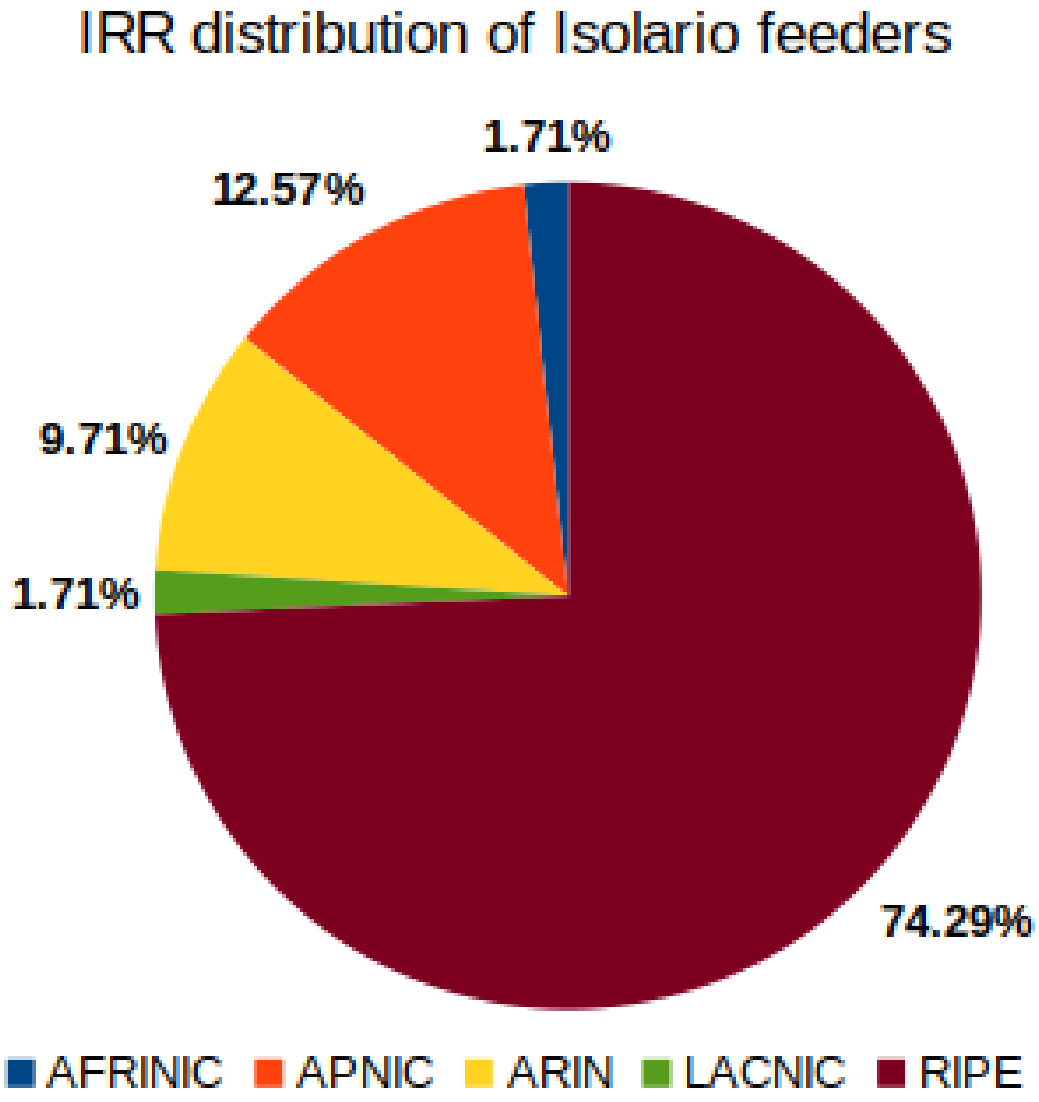
Did something happen yesterday?

Check historic services!

Do something! (if needed)



Isolario numbers (Feb 21st, 2019)



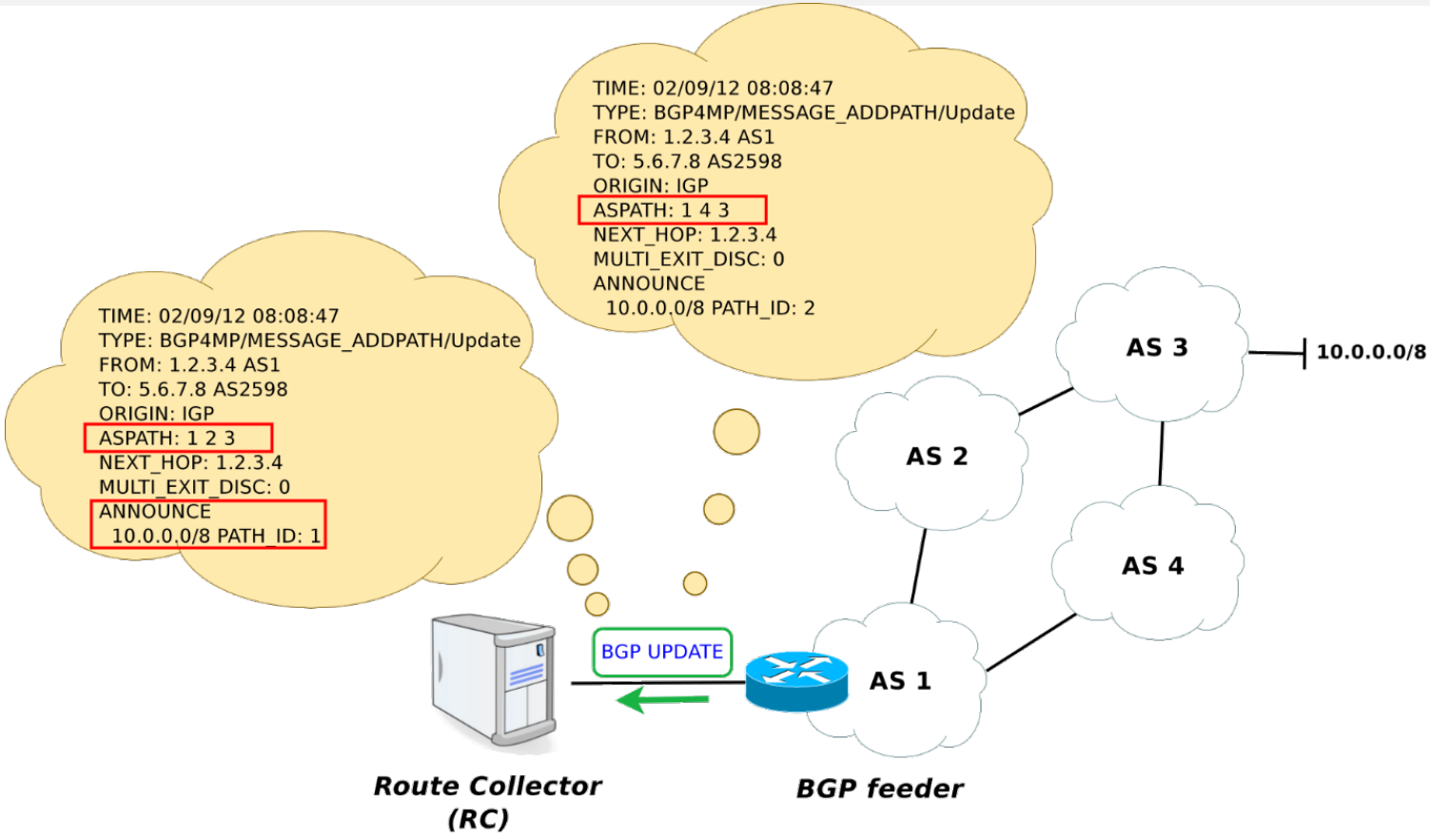
Isolario participants

- **# ASes:** 175
- **# sessions:** 522
  - **# IPv4:** 263
  - **# IPv6:** 259

Isolario full feeders

- **# ASes:** 115
  - **# IPv4:** 90
  - **# IPv6:** 98

Isolario numbers with ADD-PATH (Feb 21st, 2019)



Isolario participants in ADD-PATH

- # **ASes:** 23
- # **sessions configured:** 54
  - # **IPv4:** 25
  - # **IPv6:** 29

Isolario full feeders in ADD-PATH

- # **ASes:** 94
  - # **IPv4:** 80 (**Total:** 151)
  - # **IPv6:** 82 (**Total:** 161)

Data we plan to provide to research community (so far...)

MRT data (RFC 6396, 8050)

- RIB feeder snapshots every 2 hours
- UPDATE collections every 5 minutes

Periodic analyses (daily, weekly, monthly, . . .)

1. AS-level Topologies (Global and Geographic)
2. AS characteristics
3. Feeder contribution
4. Total coverage of RCs

Open source software

1. Interactive Collecting Engine (ICE)
2. BGP Scanner

The routing protocol

Real-time requirements

Is it possible to perform real-time queries on the RC RIB?

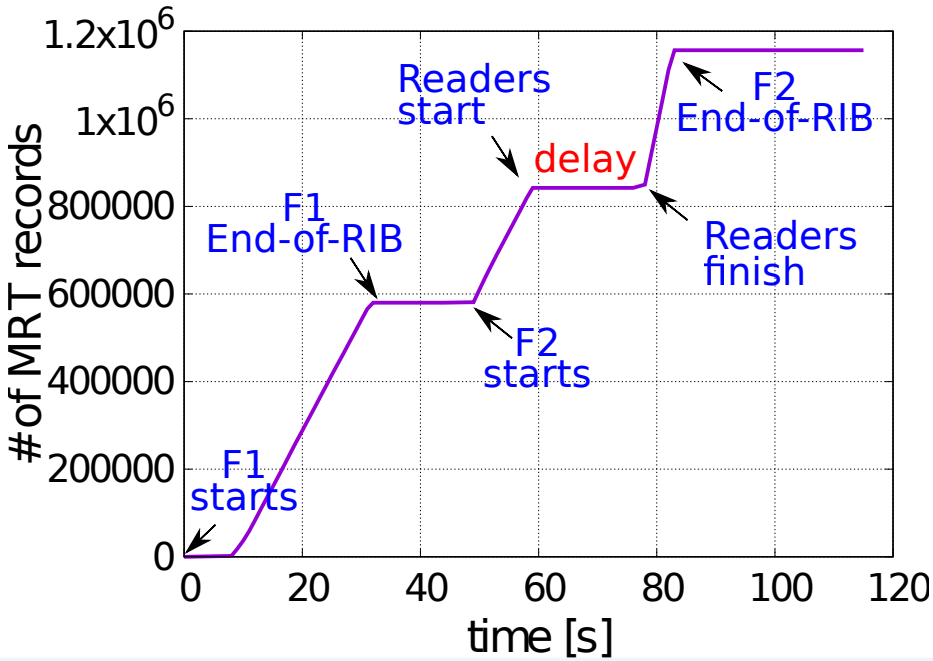
Current situation

Typically RCs run general-purpose routing software, e.g. Quagga

Cons

- The collection process is affected by the queries because most RC software is **single-threaded**
- Overhead in terms of CPU and memory usage due to BGP specs (e.g. BGP decision process)

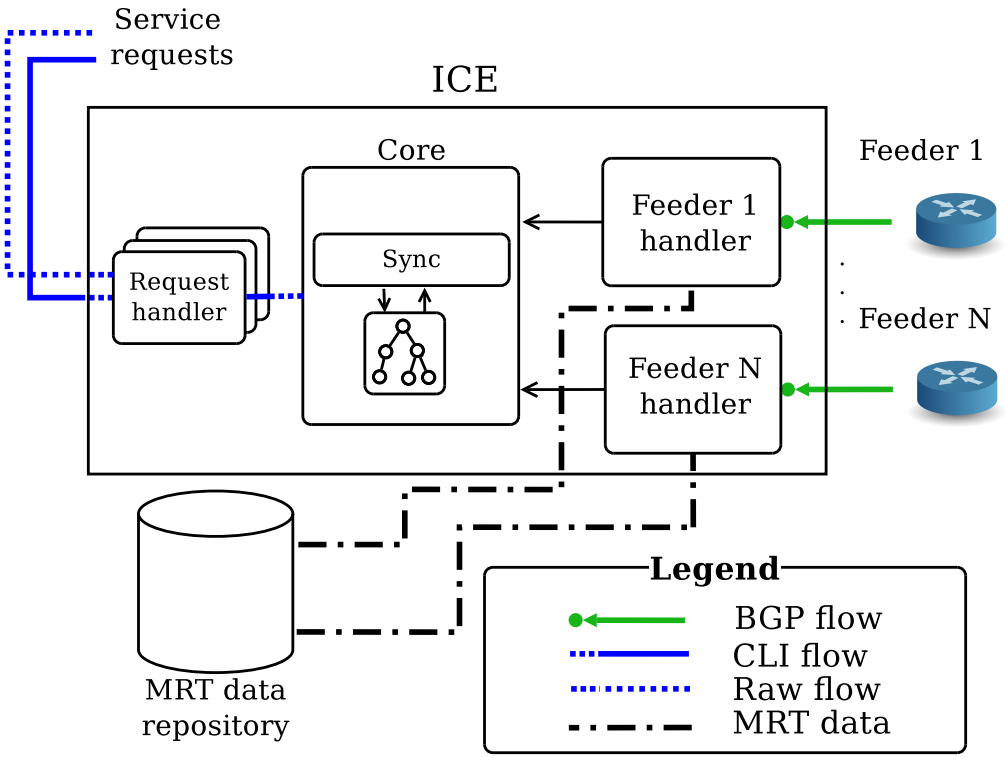
Example with Quagga



- $t = 0$ Feeder F_1 starts a RIB transfer (ends $t = 35$, $\sim$ 580k prefixes)
- $t = 45$ Feeder F_2 starts another RIB transfer
- $t = 60$ **ten** F_1 full table read operations are issued sequentially

- Data collection is delayed of about 20 seconds
- After the read requests, packets arrives with higher rate

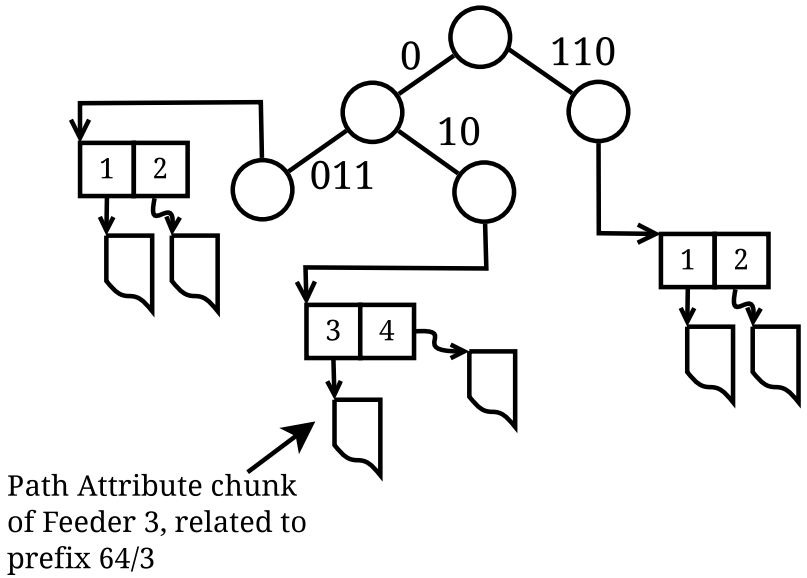
ICE: an Interactive Collector Engine



- Each BGP session is handled by a dedicated set of threads
- Each service request is handled by a dedicated thread
- Readers and writers sync on the RIB according to the classic readers-writers paradigm

[6] E. Gregori et al., *"ICE: A memory-efficient BGP route collecting engine"*, Computer Networks, pp. 162-173, vol. 126, 2017

RIB implementation: Patricia Trie



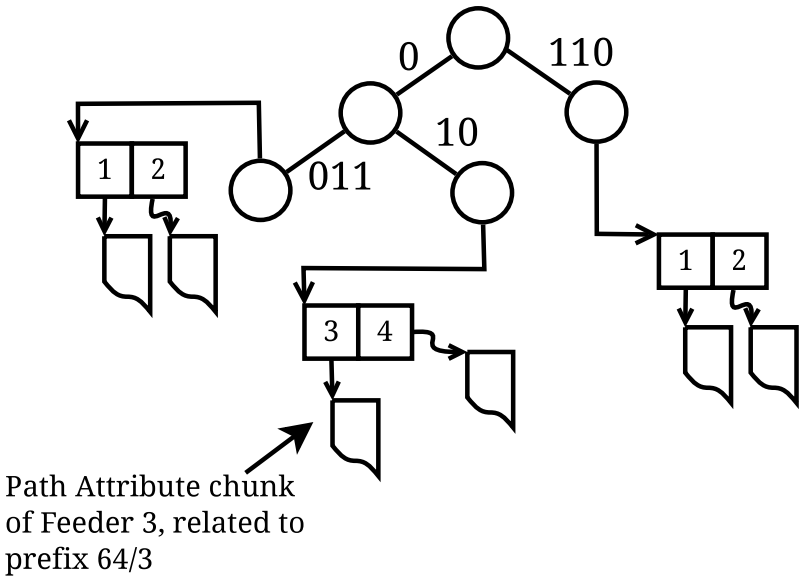
- Each node represents a subnet
- Each subnet has associated a set of path attributes, one for each feeder that announced the subnet

Readers and Writers

- Writer: feeder thread
- Reader: request thread

Writers/Readers can *W_lock*/*R_lock* both the whole trie **and** single nodes

RIB implementation: Patricia Trie



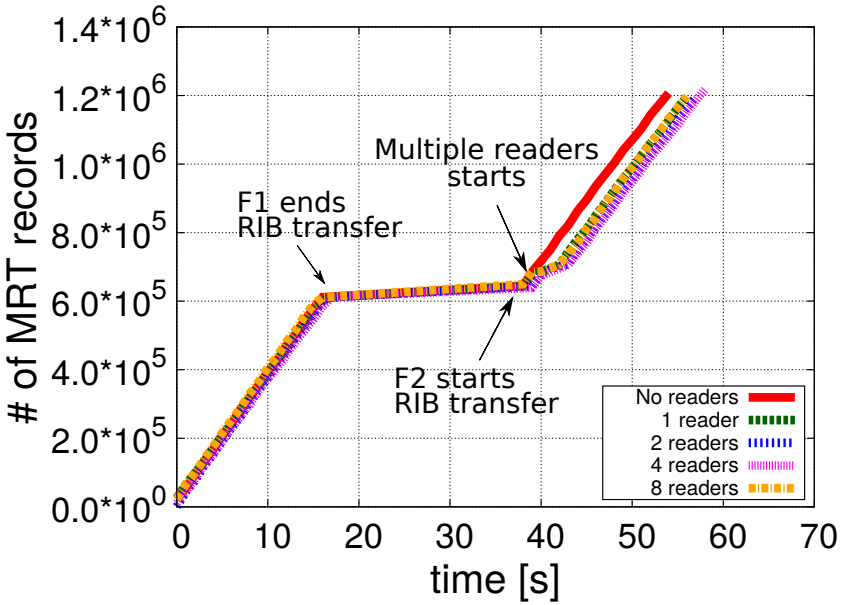
Writer

1. Checks if the node is present (**R_lock** RIB)
2. If not, inserts new node (**W_lock** RIB)
3. Inserts/updates the path attribute (**W_lock** node)

Reader

1. Checks if the node is present (**R_lock** RIB)
2. If yes, reads the node (**R_lock** node)

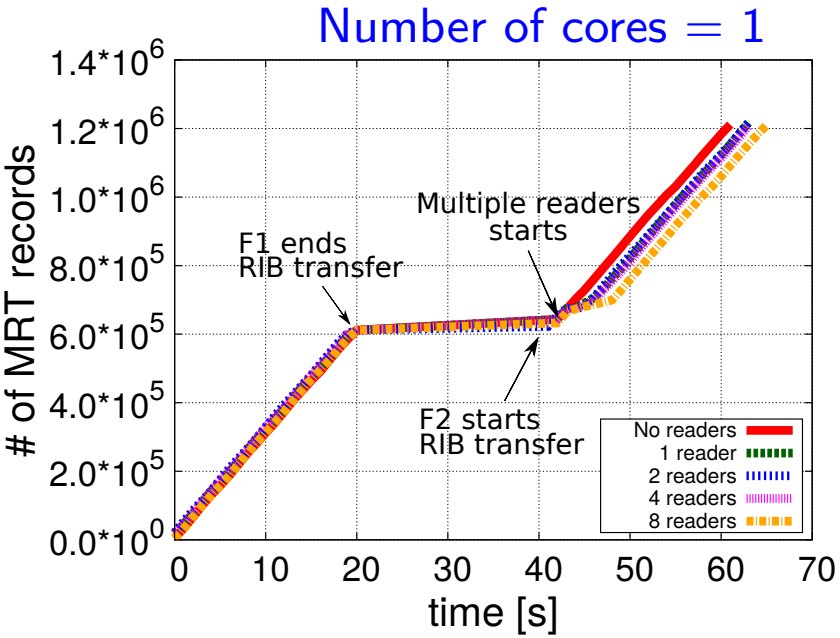
Test: delay in storing BGP packets

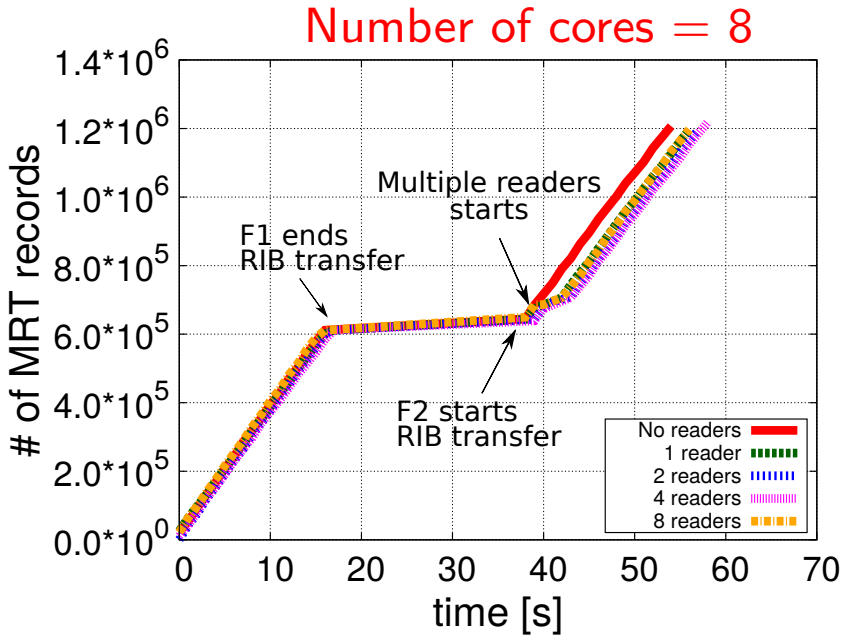


- $t = 0$ Feeder F_1 starts a RIB transfer
- $t \sim 40$ Feeder F_2 starts a RIB transfer
- $t \sim 41$ Multiple F_1 full table read operations are issued **simultaneously**

Thanks to the scheduler activity (and the multi-threaded design) ICE is able to write incoming packets while readers are reading

Test: delay in storing BGP packets





- $t = 0$ Feeder F_1 starts a RIB transfer
- $t \sim 40$ Feeder F_2 starts a RIB transfer
- $t \sim 41$ Multiple F_1 full table read operations are issued **simultaneously**

Thanks to the scheduler activity (and the multi-threaded design) ICE is able to write incoming packets while readers are reading

Test: delay in reading the RIB

# of readers	Before F_2 RIB transfer				During F_2 RIB transfer			
	1 core		8 cores		1 core		8 cores	
	μ	σ	μ	σ	μ	σ	μ	σ
1	2.12	0.07	2.09	0.07	2.21	0.07	2.18	0.08
2	2.15	0.10	2.14	0.10	2.19	0.09	2.16	0.08
4	2.32	0.15	2.20	0.11	2.28	0.13	2.17	0.11
8	3.66	0.06	2.15	0.14	3.73	0.09	2.13	0.15

- Similar to the previous test, but from the reader side
- How much time a reader takes to retrieve the full routing table of F_1 before and **during** F_2 table transfer?
- μ and σ are respectively the average time and the standard deviation

The time the second set of readers takes to retrieve F_1 table is very close to the time taken by the first set, confirming that multiple readers can proceeed in parallel with F_2

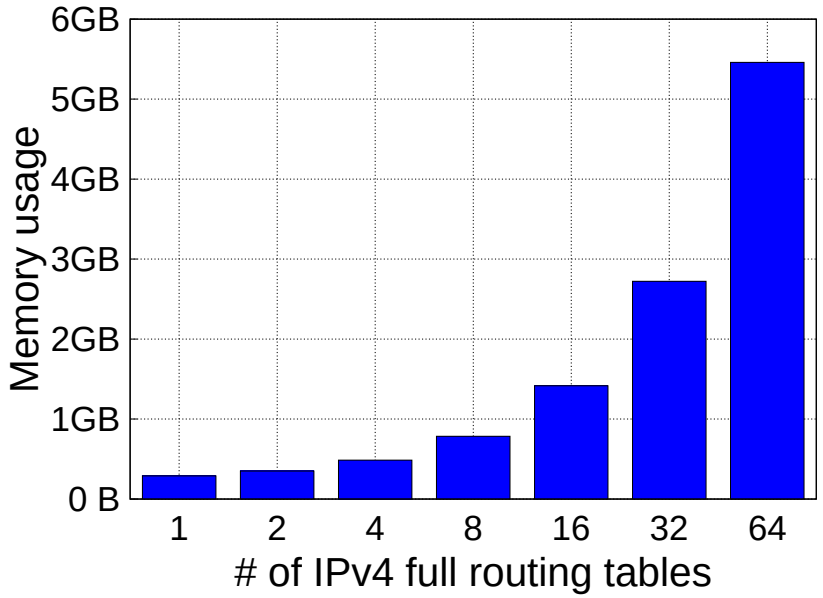
Test: delay in reading the RIB

# of readers	Before F_2 RIB transfer				During F_2 RIB transfer			
	1 core		8 cores		1 core		8 cores	
	μ	σ	μ	σ	μ	σ	μ	σ
1	2.12	0.07	2.09	0.07	2.21	0.07	2.18	0.08
2	2.15	0.10	2.14	0.10	2.19	0.09	2.16	0.08
4	2.32	0.15	2.20	0.11	2.28	0.13	2.17	0.11
8	**3.66**	0.06	2.15	0.14	**3.73**	0.09	2.13	0.15

- Similar to the previous test, but from the reader side
- How much time a reader takes to retrieve the full routing table of F_1 before and **during** F_2 table transfer?
- μ and σ are respectively the average time and the standard deviation

The time the second set of readers takes to retrieve F_1 table is very close to the time taken by the first set, confirming that multiple readers can proceeed in parallel with F_2

What about memory?



Memory consumption

- ICE uses ~ 82.4MB per feeder, ~100 feeders on a standard machine
- This means that in scenarios where the feeders are near to a thousand (e.g. route collecting, route servers) at least ten machines are needed

ICE: Future directions

New functionalities

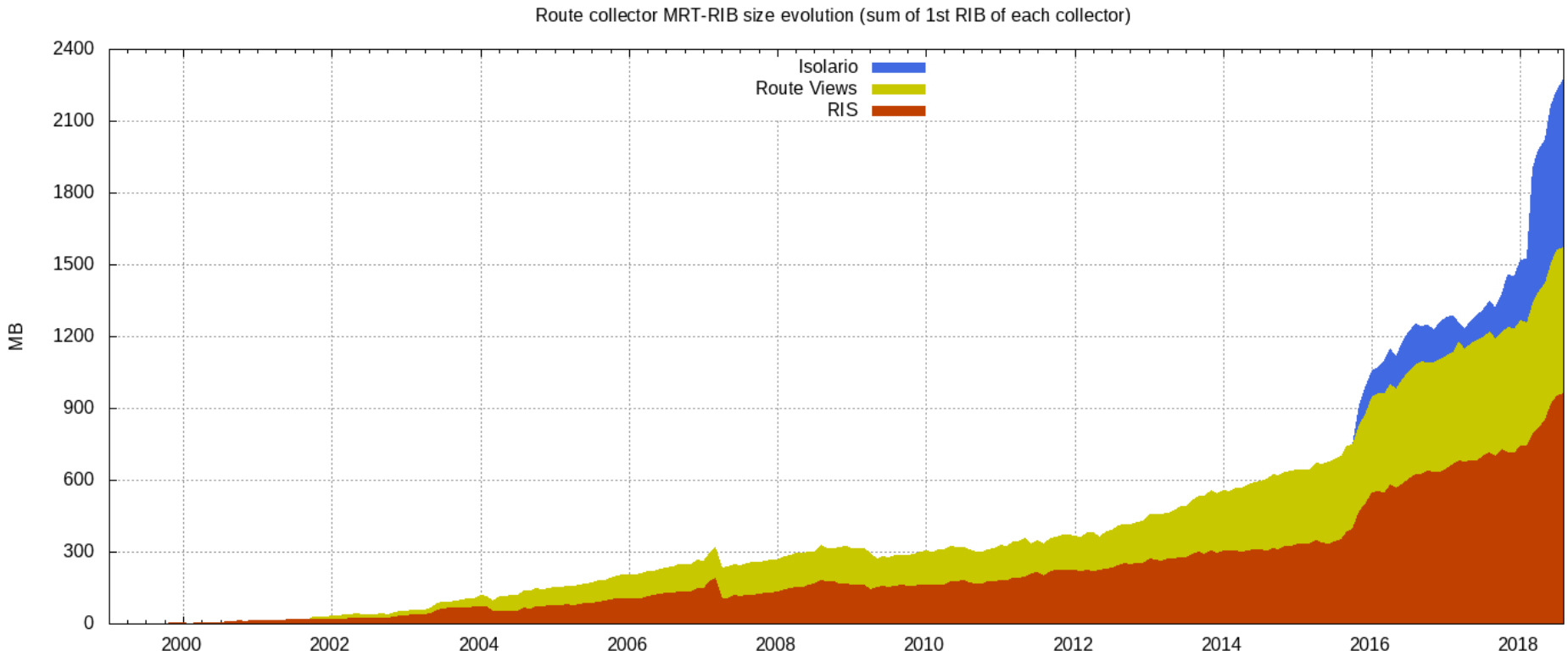
- New filtering capabilities
- Standard configuration file

Memory consumption

- Rewrite core code in C
- Improve efficiency in terms of memory consumption
- Reduce threads usage

BGP Scanner: Isolario MRT-BGP data reader

MRT data is getting bigger and bigger...



What is the problem?

- Tools available are either slow, outdated or miss ADD-PATH handling
- Usually no way to filter packets

Tools available to parse MRT data

Several languages: C, C++, Python, Perl, Java, OCaml

Tool	Lang	RIB	updates	IPv6	ADD PATH	Last updated
bgpdump	C	✓	✓	✓	✓	2018-03-02
bgpdump2	C	✓	x	✓	x	2016-08-10
bgpparser	C++	✓	✓	✓	x	2015-04-11
bgpreader	C	✓	✓	✓	x	2018-07-13
mabo	OCaml	✓	✓	✓	x	2017-06-26
mrtparse	Python	✓	✓	✓	x	2018-06-27
PyBGPdump	Python	✓	✓	x	x	2007-01-15
Java-MRT	Java	✓	✓	✓	x	2013-02-09
zebra-dump-parser	Perl	✓	✓	✓	x	2016-11-07

Solution: `goto c`

> *The fact is, that is* **exactly** *the kinds of things that C excels at. Not just as a language, but as a required* **mentality**. *One of the great strengths of C is that it doesn't make you think of your program as anything high-level.*
>
> Linus

C language benefits

- May be easily wrapped (C++, Python, Lua)
- Close to "metal"

Not only ANSI C: C99

- Allows dynamic allocation on stack ($\rightarrow$ **zero-copy**)
- Improves code readability

Solution: `goto c`

> *The fact is, that is* **exactly** *the kinds of things that C excels at. Not just as a language, but as a required* **mentality**. *One of the great strengths of C is that it doesn't make you think of your program as anything high-level.*
>
> Linus

C language benefits

- May be easily wrapped (C++, Python, Lua)
- Close to "metal"

Not only ANSI C: C99

- Allows dynamic allocation on stack ($\rightarrow$ **zero-copy**)
- Improves code readability

Solution: `goto` c

> *The fact is, that is* **exactly** *the kinds of things that C excels at. Not just as a language, but as a required* **mentality**. *One of the great strengths of C is that it doesn't make you think of your program as anything high-level.*
>
> Linus

C language benefits

- May be easily wrapped (C++, Python, Lua)
- Close to "metal"

Not only ANSI C: C11

- Allows optimization for multithreading
- Thread local/atomic variables

BGP Scanner: Isolario MRT-BGP data reader

Don't worry!

You do not have to use C

MRT-BGP library

- A highly optimized low-level reusable library
- Optimized to achieve high throughput
- Multi-thread friendly
- Memory friendly

The real star of the show: BGP Scanner tool

- Comes with all the benefits of the low-level C library
- Good old `grep` friendly output
- Can be piped to other tools
- Supports `gz`, `bz2`, `xz` and raw
- Powerful filtering features

Wait... what?

Filtering

- Peer IP, Peer AS
- Subnets, supernets, related, exacts
- Peer Index
- AS path regexp and loop detection
- Can be configured with template files and/or directly by command line

Example

```
bgpscanner -s 192.65.0.0/16 -p "174 137" rib.20180701.1400.bz2

=|192.65.131.0/24|7018 174 137 137 137 2598|12.0.1.63|i|||7018:5000 7018:37232|12.0.1.63 7018|1530186440|1
=|192.65.131.0/24|6539 577 174 137 137 137 2598|216.18.31.102|i||||216.18.31.102 6539|1529912797|1
=|192.65.131.0/24|701 174 137 137 137 2598|137.39.3.55|i||||137.39.3.55 701|1529397335|1
=|192.65.131.0/24|3741 174 137 137 137 2598|168.209.255.56|i||||168.209.255.56 3741|1529593095|1
=|192.65.131.0/24|11686 174 137 137 137 2598|96.4.0.55|i||||96.4.0.55 11686|1530338943|1

...
```

Subnets of `192.65.0.0/16` crossing `174 137` link

It can do a lot more...

```
Available options:
-a <feeder AS>
        Print only entries coming from the given feeder AS
-A <file>
        Print only entries coming from the feeder ASes contained in file
-d
        Dump packet filter bytecode to stderr (debug option)
-e <subnet>
        Print only entries containing the exact given subnet of interest
-E <file>
        Print only entries containing the exact subnets of interest contained in file
-f
        Print only every feeder IP in the RIB provided
-i <feeder IP>
        Print only entries coming from a given feeder IP
-I <file>
        Print only entries coming from the feeder IP contained in file
-l
        Print only entries with a loop in its AS PATH
-L
        Print only entries without a loop in its AS PATH
-o <file>
        Define the output file to store information (defaults to stdout)
-p <path expression>
        Print only entries which AS PATH matches the expression
-P <path expression>
        Print only entries which AS PATH does not match the expression
-r <subnet>
        Print only entries containing subnets related to the given subnet of interest
-R <file>
        Print only entries containing subnets related to the subnets of interest contained in file
-s <subnet>
        Print only entries containing subnets included to the given subnet of interest
-S <file>
```

Benchmarks: BGP data evolution scenario

Test machine

- Intel(R) Core(TM) i7-4790K 4.00GHz
- RAM 16GB
- Samsung SSD 850 EVO 500GB
- Debian Stretch

Data sources

- Route Views `route-views6`
- RIS `rrc00`
- Isolario `Korriban`

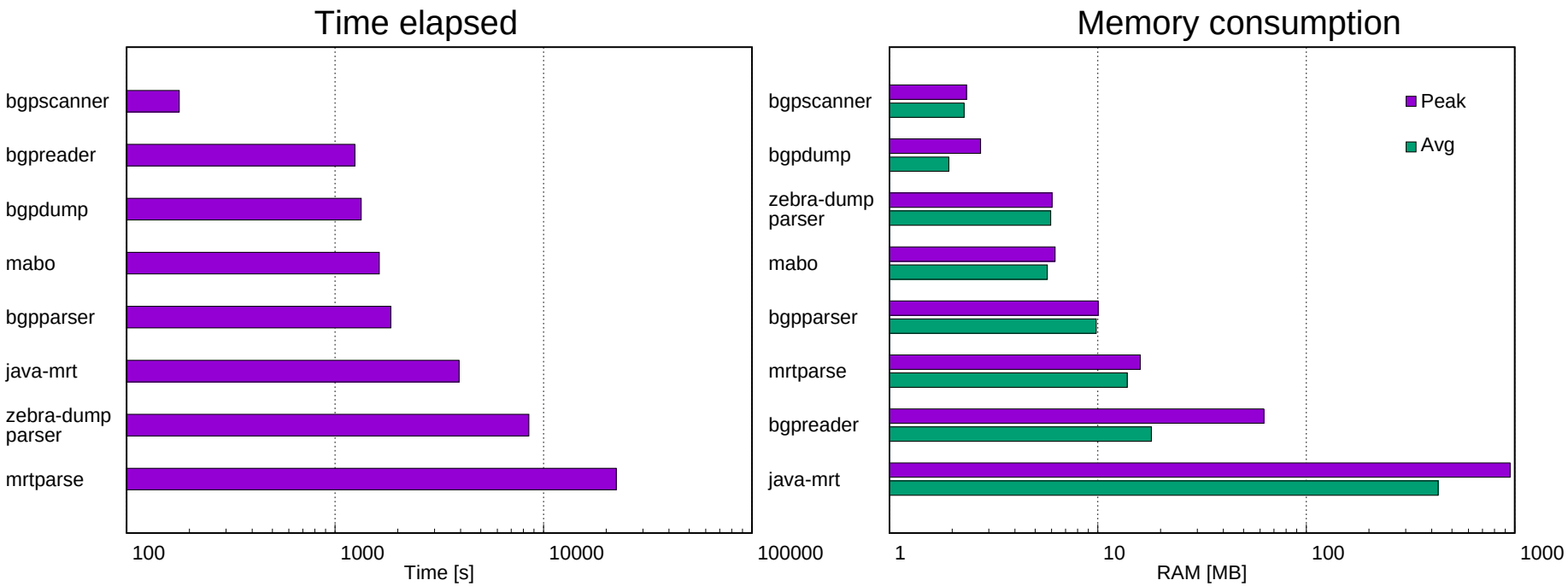
Benchmark phases ($\forall$ collectors)

1. Download first RIB of July, 2018
2. Download all updates of July, 2018
3. Decompress
4. Run 10 times each MRT tool
5. Compute average results of runs for each metric

Data is decompressed to eliminate decompression algorithm overhead

Route Views `route-views6` collector

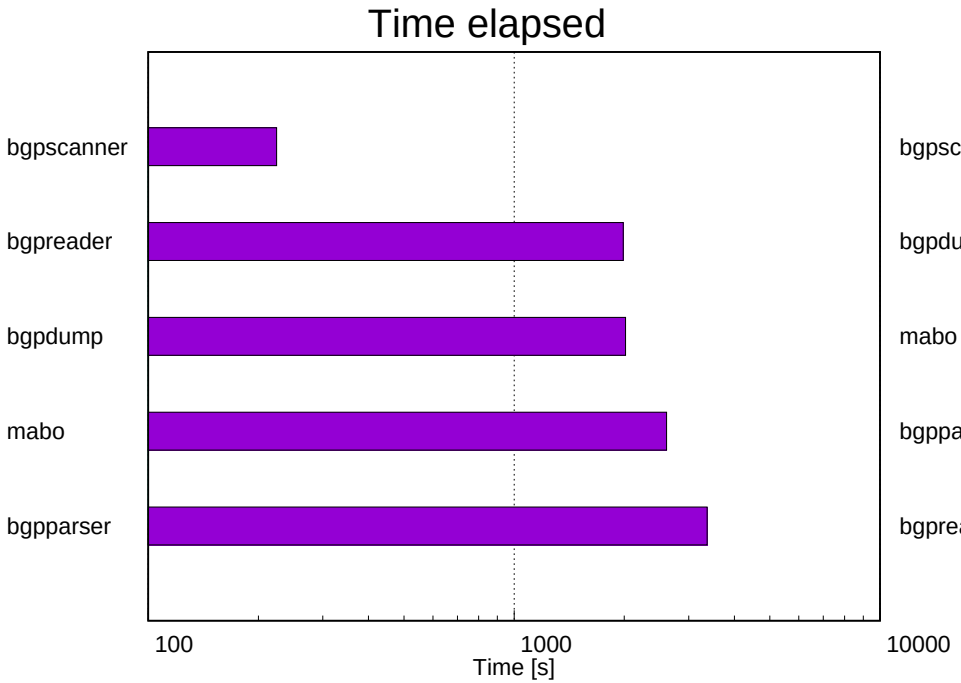
RIB size	$\sum$ UPDATE size	μ UPDATE size	# files
99MB	25.65GB	8.82MB	2977

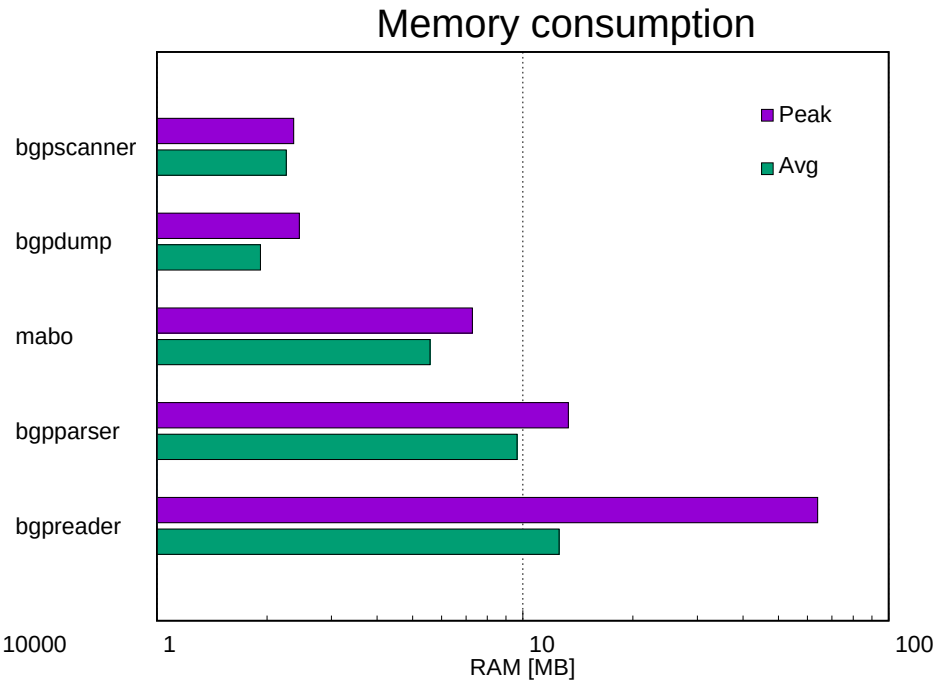


Only IPv6 feeders
(26 sessions, 24 full tables)

RIS `rrc00` collector

RIB size	$\sum$ UPDATE size	μ UPDATE size	# files
1.1GB	33.6GB	3.85MB	8930



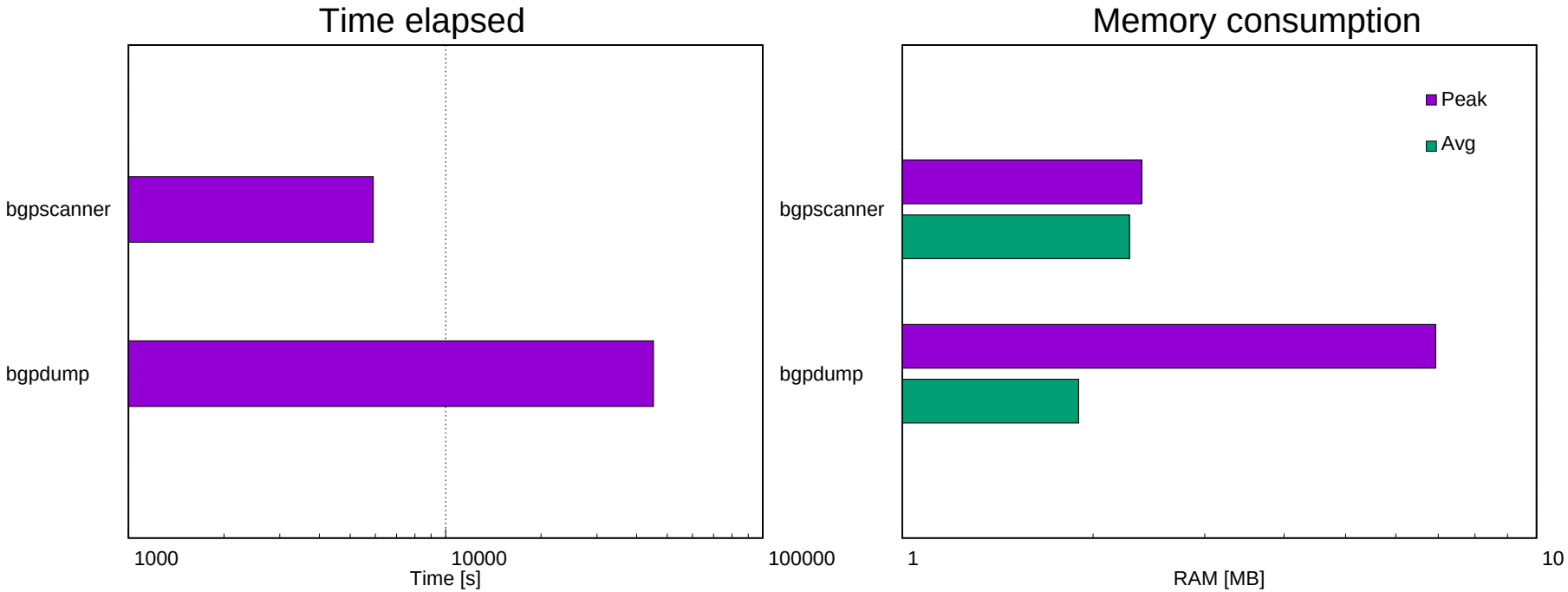


IPv4 + IPv6 feeders (39 sessions)

- 22 IPv4 sessions (21 full tables)
- 17 IPv6 sessions (14 full tables)

Isolario `Korriban` collector

RIB size	$\sum$ UPDATE size	μ UPDATE size	# files
5.7GB	810.64GB	92.97MB	8930



IPv4 + IPv6 feeders with ADDPATH

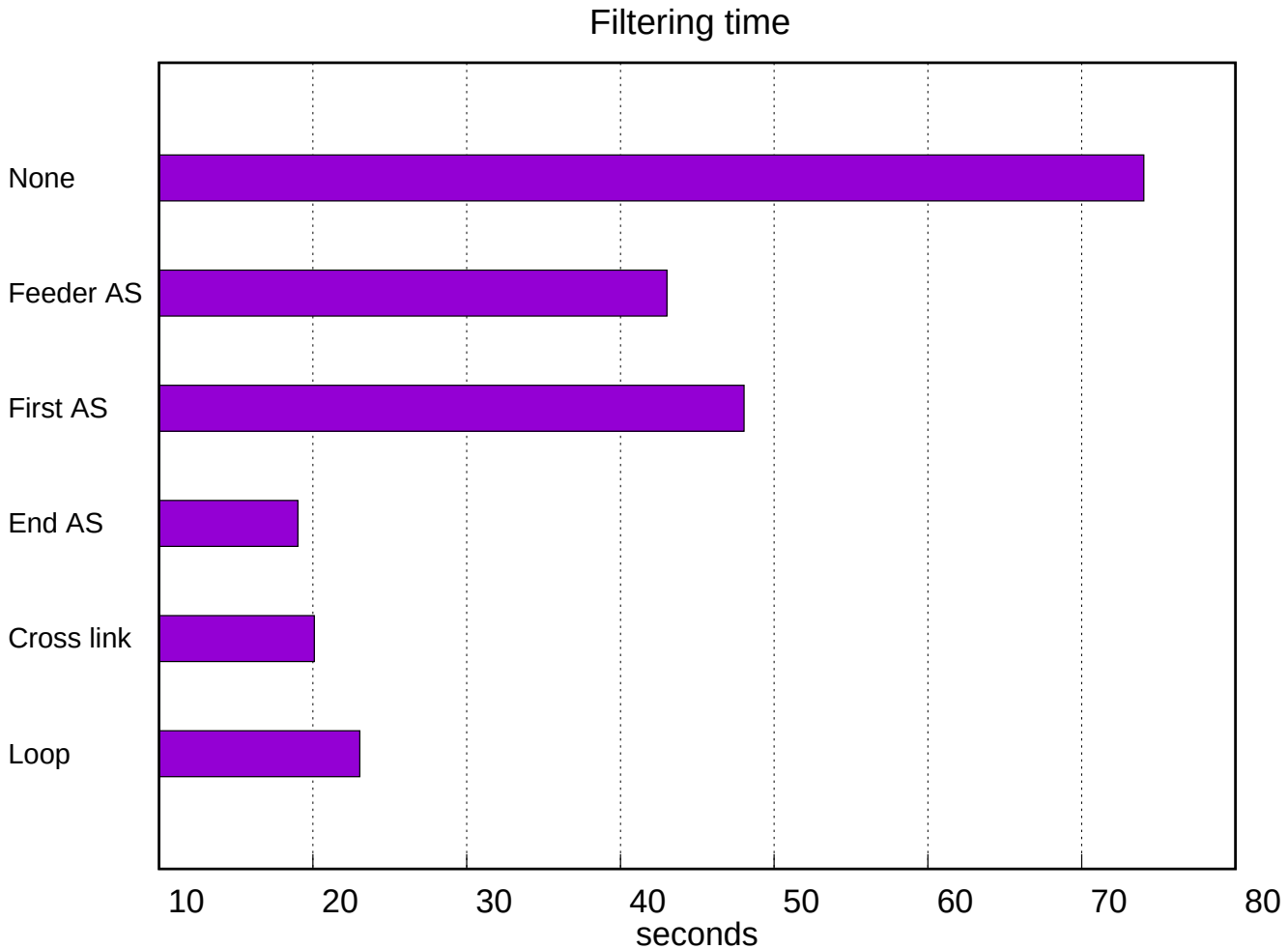
- 512 IPv4 sessions (112 full tables)
- 407 IPv6 sessions (126 full tables)

Thanks to NLNOG RING for providing 68 IPv4 and 69 IPv6 full tables!

Filtering benchmark

Data source

- Last RIB of July 2018 of `Korriban` collector
- 475MB (7.8GB uncompressed)



A possible IXP use case

Assumption

- No IXP peering LAN should be announced on the Internet
- Thus, no route collector should see any IXP peering LAN

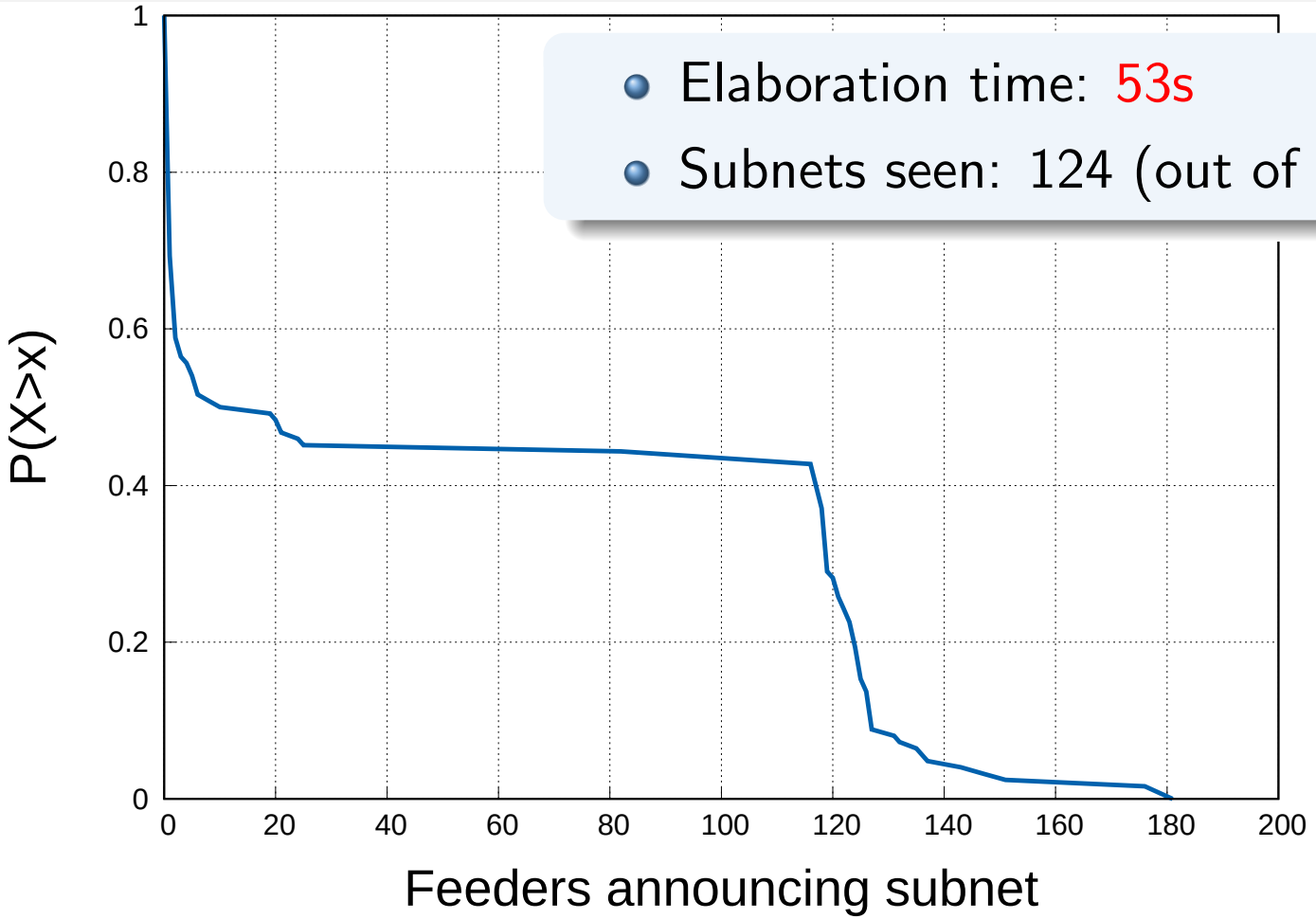
Procedure

1. Extract peering LANs via PeeringDB APIs
2. `bgpscanner -E <Peering LAN file> <RIB>`
3. Analyse the results (e.g. to check for particular LANs of interest)

Data

- Monitor: `Korriban`
- File: `rib.20190212.0000`

A possible IXP use case: results



- Elaboration time: 53s
- Subnets seen: 124 (out of 1176)

Peering LANs could be announced on purpose (for any reason)

Bgpscanner can help in identifying those which are not

How to install BGP Scanner?

BGP Scanner is open-source

- BSD license
- Available on `www.isolario.it` → Tools
- Source code on `https://gitlab.com/Isolario`

Install procedure (Source Tarball)

1. Download bgpscanner-`[version]`.tar.gz
2. `./configure && make && make install`

Install procedure (Debian package archives)

1. Download libisocore1_`[version]`.deb
2. Download bgpscanner_`[version]`.deb
3. dpkg -i *.deb

Thank you for your attention

Any question?

alessandro.improta@iit.cnr.it
luca.sani@iit.cnr.it

https://www.isolario.it