UCLouvain

icteam

ip networking lab
ucl, louvain-la-neuve, belgium

Some Routing Challenges for Future Networks

Olivier Bonaventure

HPSR'21 SARNet workshop
June 2021

https://pluginized-protocols.org

Agenda

- **Some Routing challenges**
- Extending Internet protocols
- xBGP a paradigm shift for routing protocols

Some challenges for future routing protocols

- Scalability
  - Intradomain routing protocols have scalability concerns
    - OSPF and IS-IS tend to be very chatty in large networks
    - Cloud operators opt for BGP in datacenters
    - New protocols are being pushed within IETF
  - Interdomain routing tables continue to grow

BGP Table Data

Report last updated at Wed, 9 Jun 2021 17:16:57 GMT

IPv4 BGP Reports

AS131072	APNIC R&D	884985
AS6447	Route-Views.Oregon-ix.net	1031664

IPv4 Route-Views

IPv6 BGP Reports

AS131072	APNIC R&D	129343
AS6447	Route-Views.Oregon-ix.net	130738

Some challenges for future routing protocols

- Security
  - Intradomain routing security
    - might need more than simply router authentication as currently done
    - Future intradomain routing protocols should associate certificate to routers and force them to authenticate the advertisements that they send
  - Interdomain routing security
    - BGP hijacks are frequent and RPKI gets slowly deployed
    - Future interdomain security will need more than prefix origini authentication

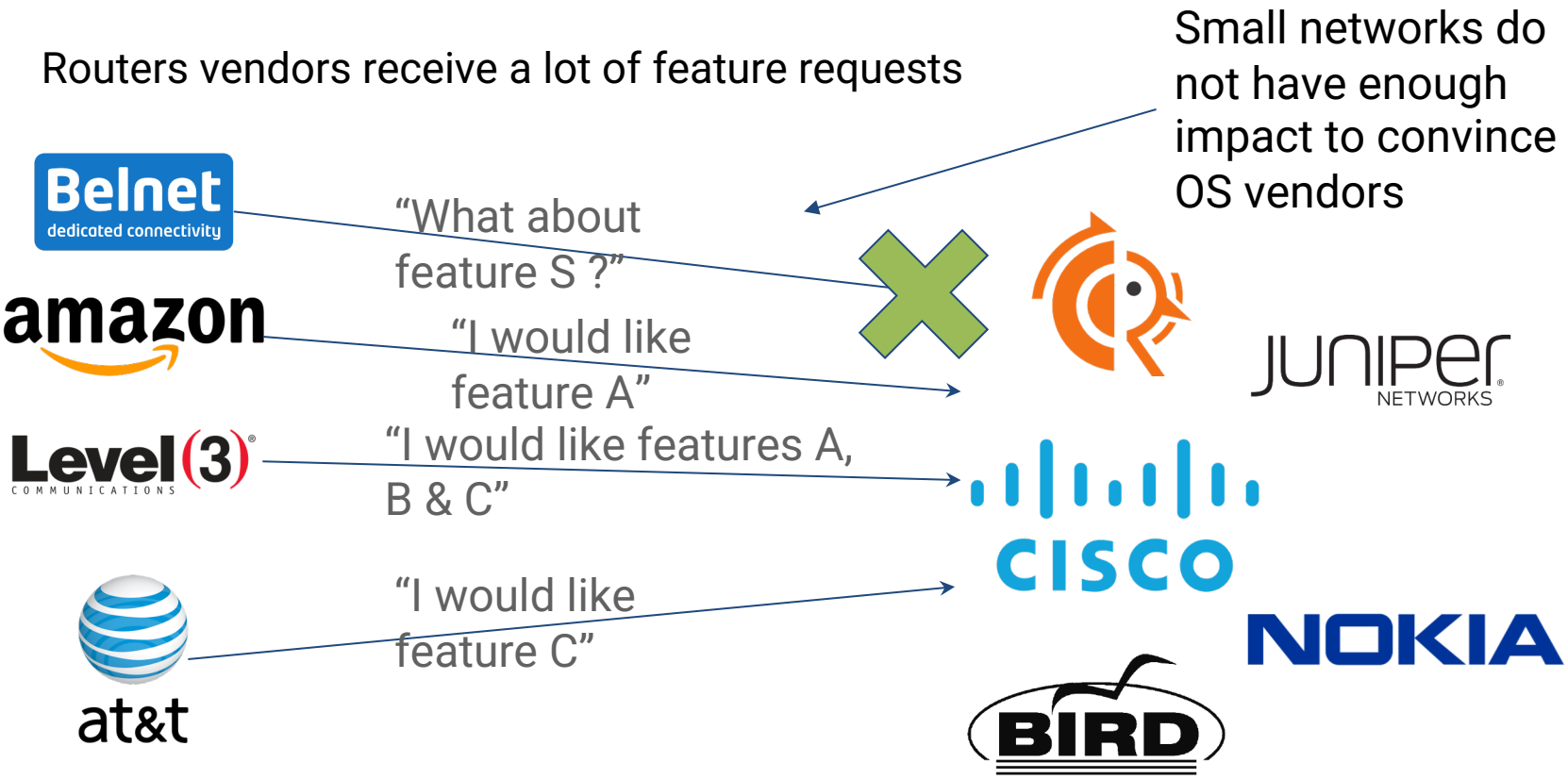
Some challenges for future routing protocols

- Fast convergence ?
  - An important requirement in the past and routing protocols have been heavily tuned
  - Nowadays, we have various fast-reroute techniques (MPLS, LFA, Segment Routing) that allow to cope with link failures for a short period of time
- Loop free convergence
  - When topology changes, routing updates should not cause any transient loop that would cause packet losses

Some challenges for future routing protocols

- Traffic enginneering
  - Intradomain traffic engineering
    - Routing protocols build routing tables, but network operators need to direct flows to mitigate congestion and optimize performance
    - Segment Routing brings opportunities to more closely couple routing and traffic engineering
  - Interdomain traffic enginnering
    - Using smaller prefixes causes scalability problems
    - Need to reconsider this problem with IPv6 but allocating several prefixes to each AS/host and rely on multipath transport protocols to select the most suitable path

How can routers address all needs ?



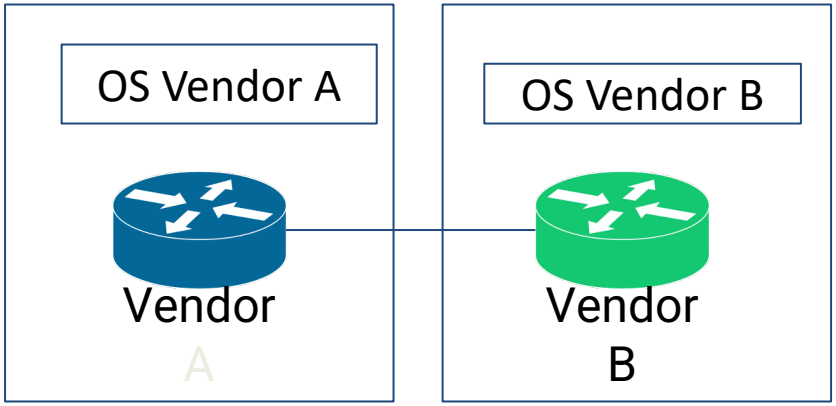
Standardization is important

- Large networks are multi-vendor and standardization provides interoparability even if configuration languages differ

```
routing-options {
  router-id 1.1.1.1;
  autonomous-system 65001;
}

protocols {
  bgp {
    group Session-to-R1 {
      type external;
      neighbor 1.1.1.2 {
      peer-as 65002;
    }
  }
}
```

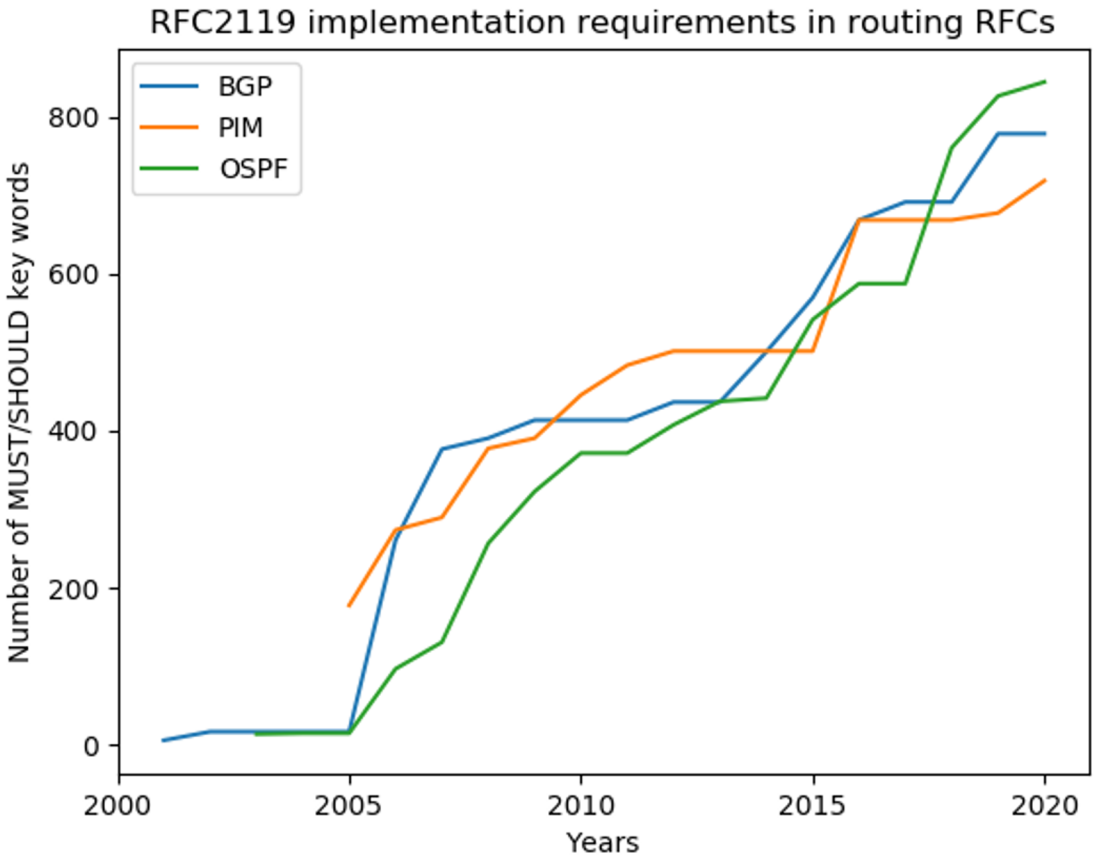
Simple Juniper configuration file



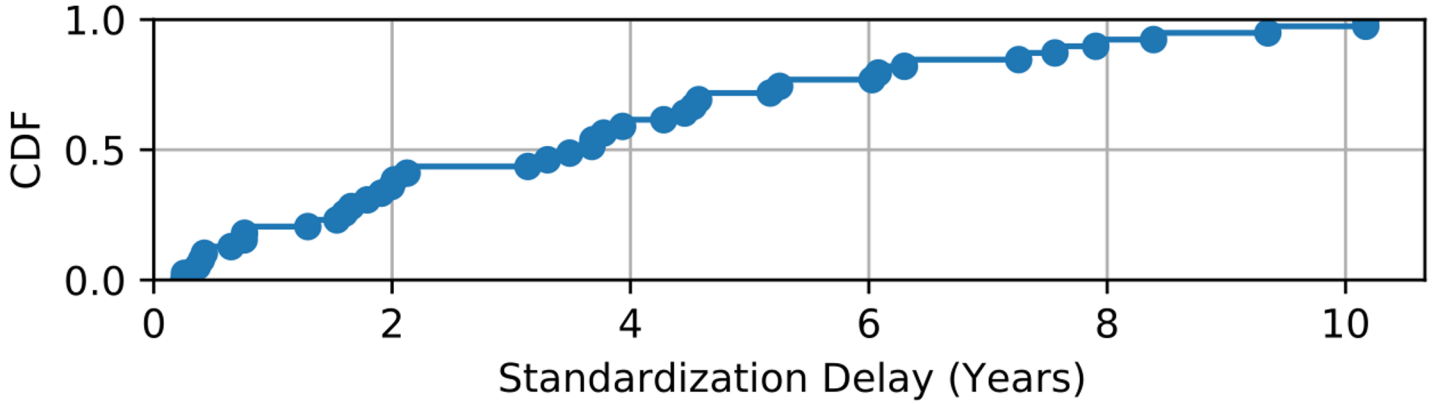

```
router bgp 65001
  bgp router-id 1.1.1.1
  neighbor 1.1.1.2 remote-as 65002
```

Simple Cisco configuration file

Standardized routing protocols evolve



… but slow



- Time required to finalize a BGP extension
  1. IETF discussions (3.5 years in average for BGP)
  2. Implementation on the vendor OS
  3. Update routers of networks

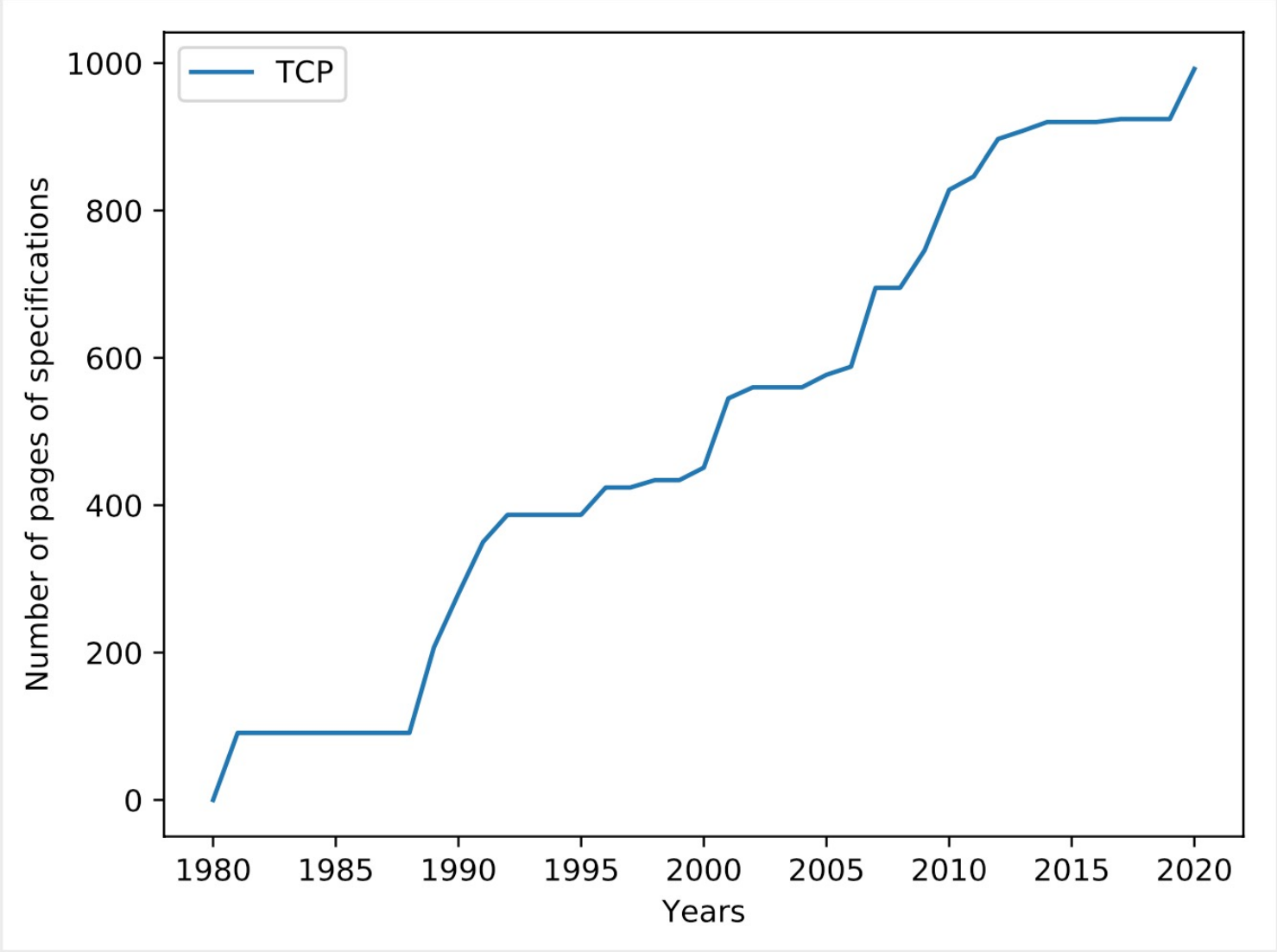
Towards a paradigm shift

- We need to take into account the **evolution** of the routing protocols and their implementations in thedesign
  - Protocol syntax is usually extensible with TLVs or other techniques like ASN.1, JSON, CBOR, ...
  - Unfortunately, protocol implementations are not always designed to be easily extensible

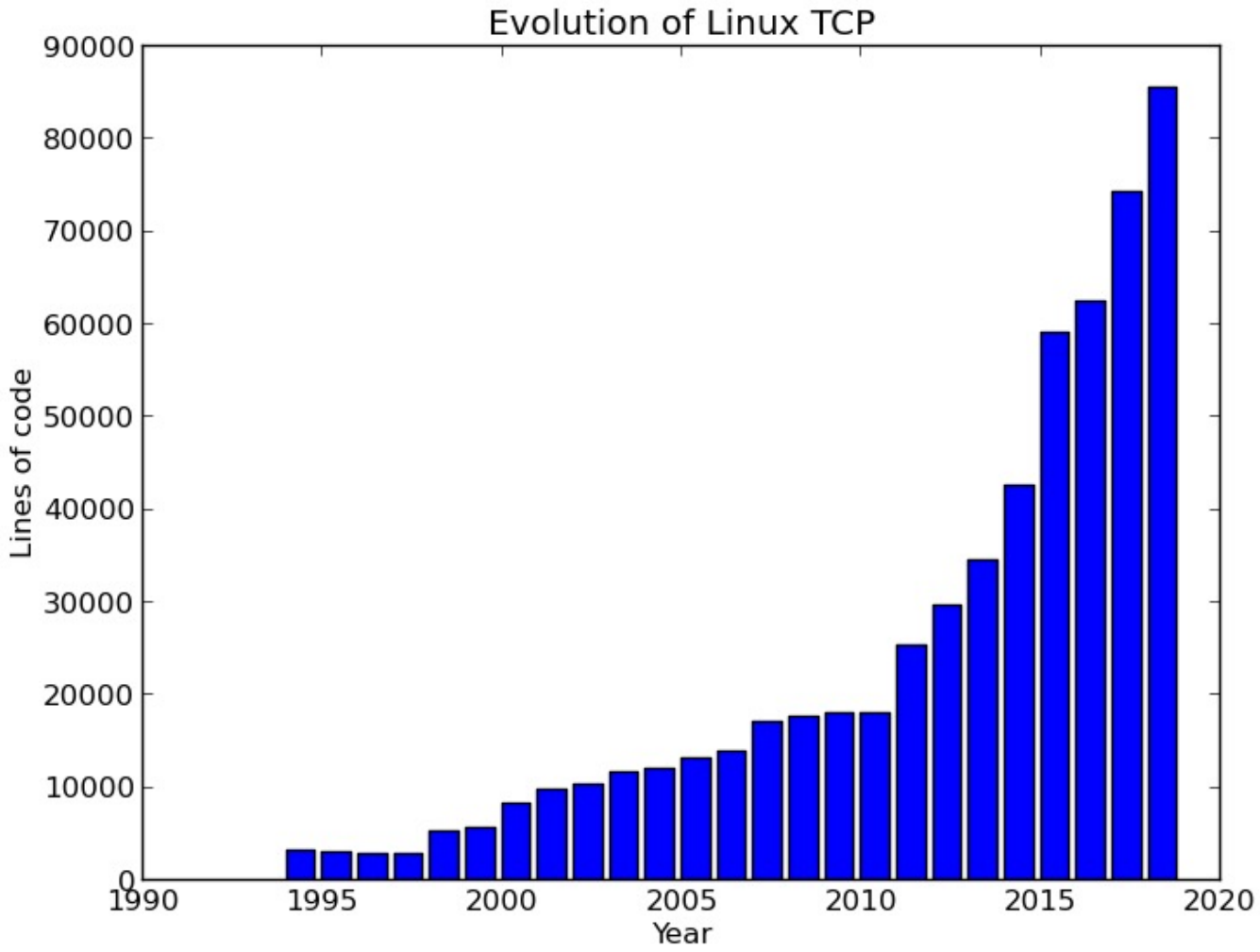
Agenda

- Some Routing challenges
- **Extending Internet protocols**
- xBGP a paradigm shift for routing protocols

Evolution of TCP RFCs



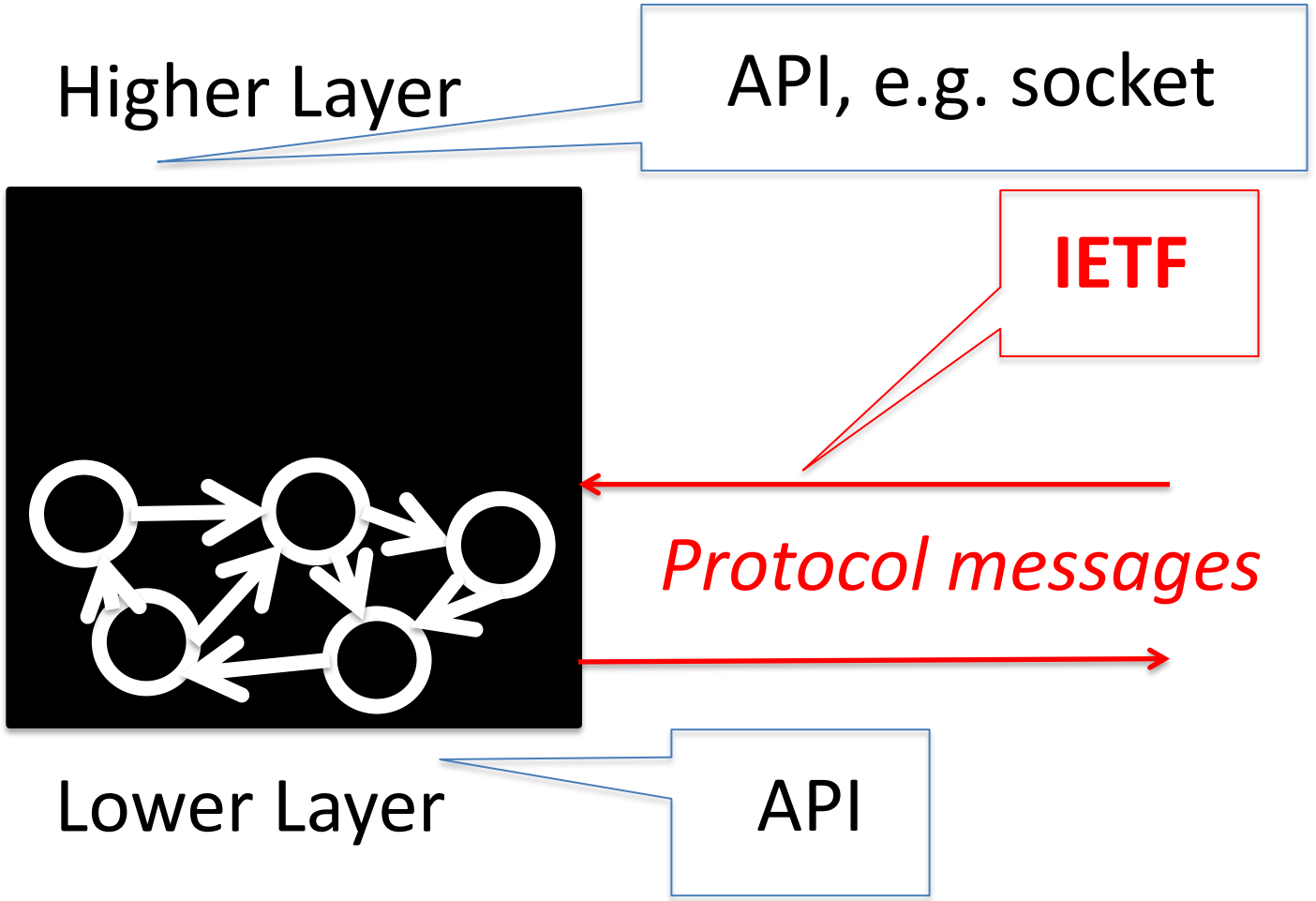
TCP's complexity increases



But deploying TCP extensions remains very difficult

- 20th century extensions took more than a decade to be widely deployed
  - TCP Window Scale
  - TCP Timestamp
    - Still not supported by Microsoft Windows
  - TCP Selective Acknowledgements
  - Explicit Congestion Notification
- Multipath TCP is being deployed, but getting it everywhere will require lots of effort

Today's implementations are **black boxes**

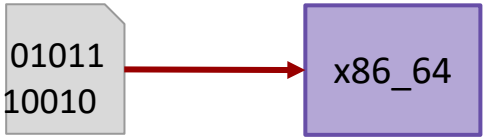


Tuning such an implementation

- Implementations typically expose a few configuration knobs
  - Socket options to enable/disable a given feature
  - Socket options to set some limit (e.g. window)
  - Sysctl variables for system-wide tuning
  - Linux modules provide some flexibility
    - Congestion control as loadable modules
    - Path managers in Linux Multipath TCP

eBPF

- Lightweight virtual machine, in Linux kernel since 2014
  - RISC instruction set (~100)
    - ALU, memory and branch purposes



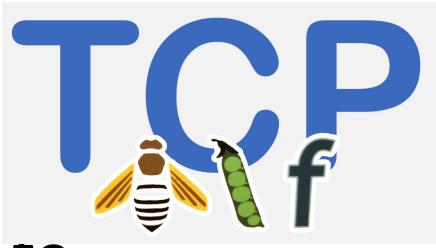
- Bytecode recompiled to native architecture
- Verifier
  - Checks absence of loops, stack usage, …
- Dedicated, isolated stack memory
  - But no persistence
- Use cases
  - Monitoring, SECCOMP, …

In-protocol debugging with eBPF

- eBPF probes can be attached at specific places in the TCP stack to observe unusual events
  - Retransmission of SYN packet
  - Reception of out-of-order packets
  - Peak in measured round-trip-time
  - Application too slow to recv data from kernel
  - …
- Daemon collects stats and sends them via IPFIX

O. Tilmans, O. Bonaventure, COP2: Continuously Observing Protocol Performance, Feb. 2019, arxiv 1902.04280v1

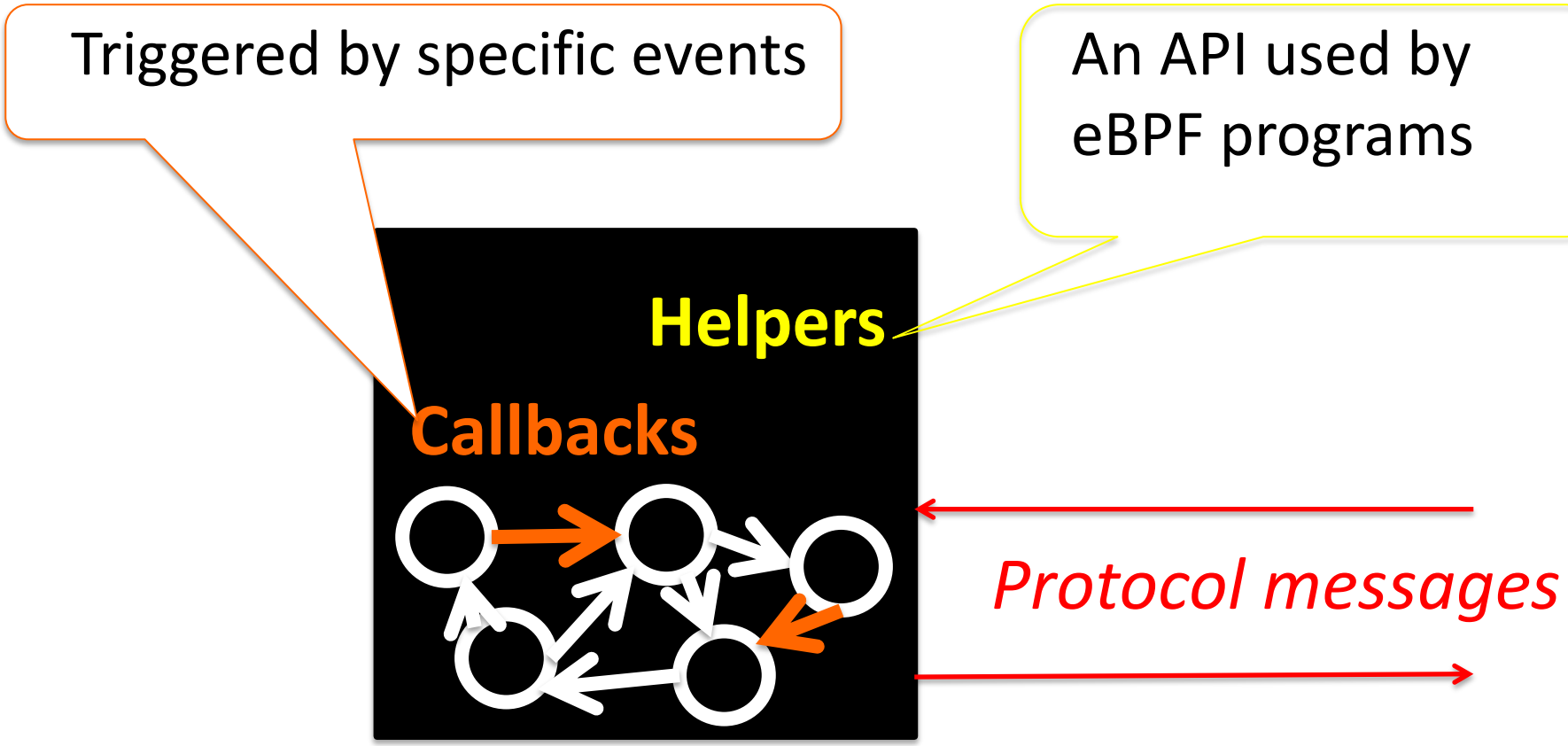
TCP can be made more extensible



- Starting point
  - Lawrence Brakmo's TCP-BPF patches
  - Adds various hooks inside the TCP stack to
    - Callbacks
      - BPF_SOCK_OPS_TCP_CONNECT_CB, BPF_SOCK_OPS_ACTIVE_ESTABLISHED_CB, BPF_SOCK_OPS_PASSIVE_ESTABLISHED_CB
    - Access to socket options
      - BPF_SETSOCKOPT and BPF_GETSOCKOPT
    - Read and write TCP state variables (rtt, cwnd, …)
  - Main use case is to **configure** TCP parameters on a per connection basis

Brakmo, L. (2017). TCP-BPF: Programmatically tuning TCP behavior through BPF. *NetDev 2.2*.

What does TCP-BPF brings ?



Extending TCP and MPTCP using eBPF

Computer Communications

Volume 162, 1 October 2020, Pages 118-138

Beyond socket options: Towards fully extensible Linux transport stacks

Viet-Hoang Tran ✉, Olivier Bonaventure ✉

Show more

Share Cite

https://doi.org/10.1016/j.comcom.2020.07.036

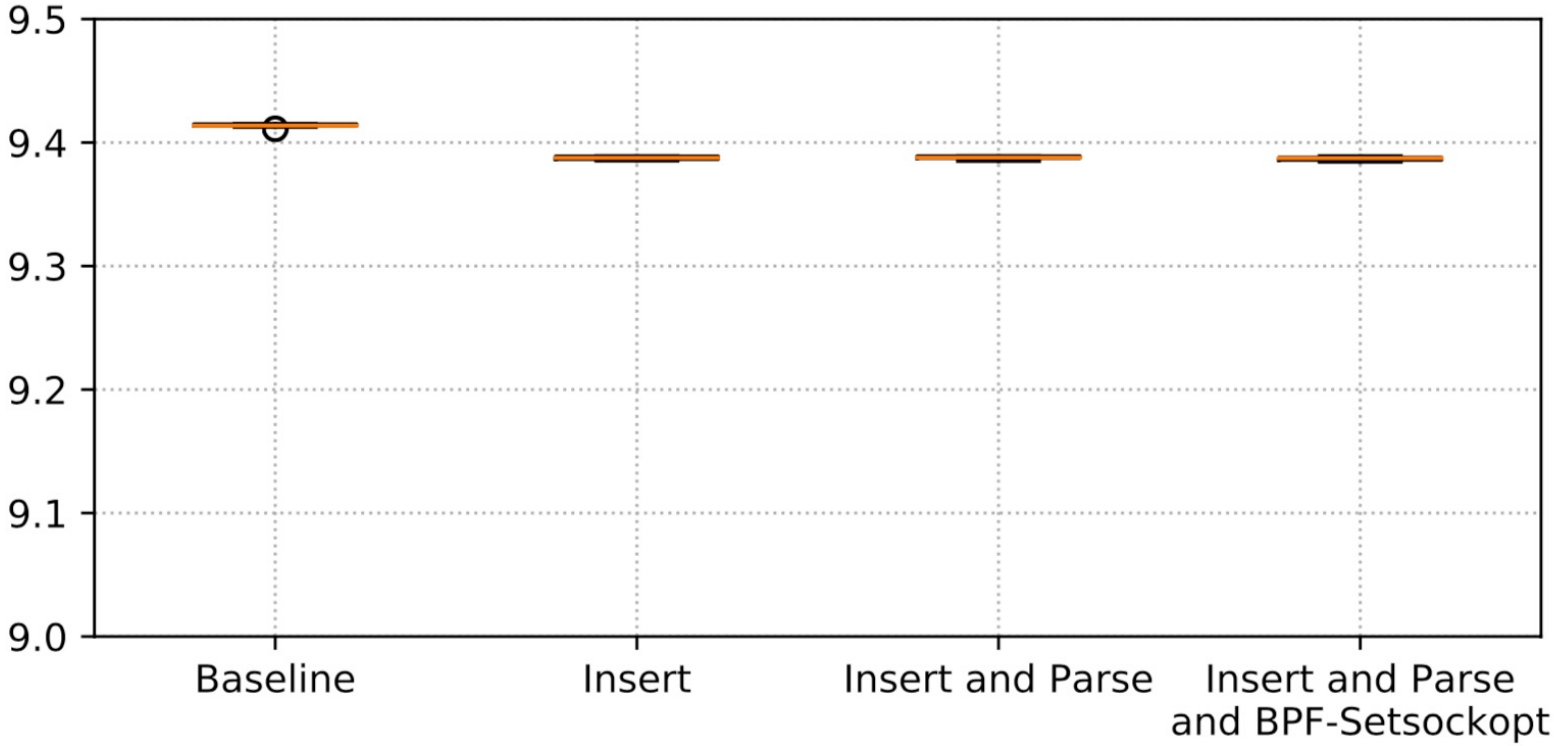
Get rights and content

User Timeout TCP option

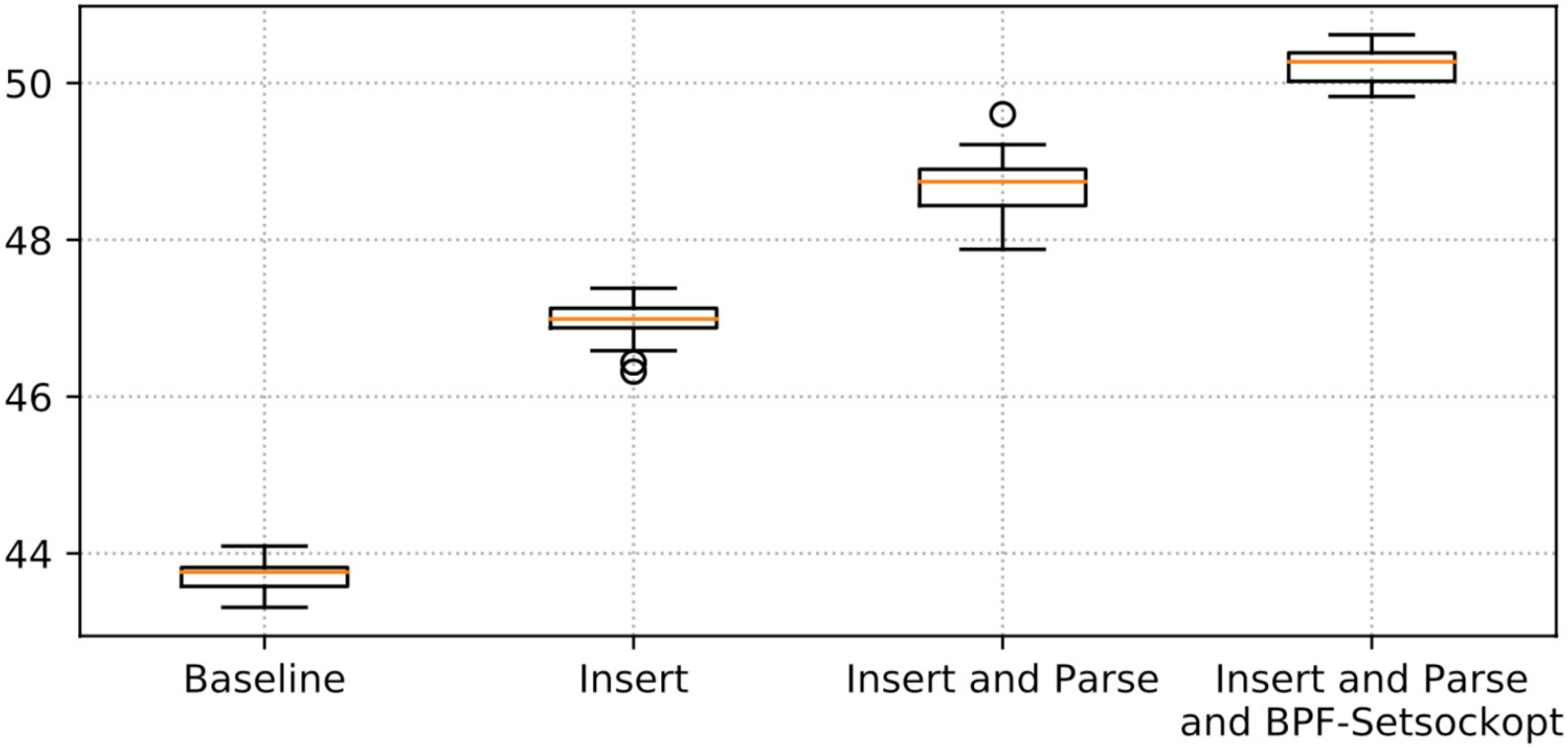
- Defined in RFC5483 but not supported by Linux
- Sender side changes
  - Add eBPF hooks in *tcp_transmit_skb* and *tcp_options_write*
  - eBPF code controls the transmission of this option
- Receiver side changes
  - Add eBPF hook to *tcp_parse_options*
  - eBPF code interprets the received option and adjusts TCP state

Facebook produced a related patch that is now integrated in the mainline Linux kernel
Recent versions allow to implement congestion control entirely using eBPF

Minor performance impact



CPU utilisation on receiver



Agenda

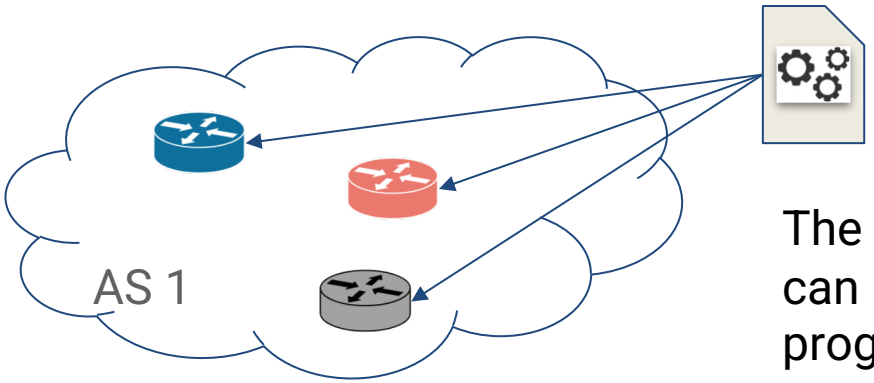
- Some Routing challenges
- Extending Internet protocols
- **xBGP a paradigm shift for routing protocols**

The xBGP vision

Each xBGP compliant router exposes a simple API that allows to dynamically extend the protocol with platform-independent code that we call **plugins**.

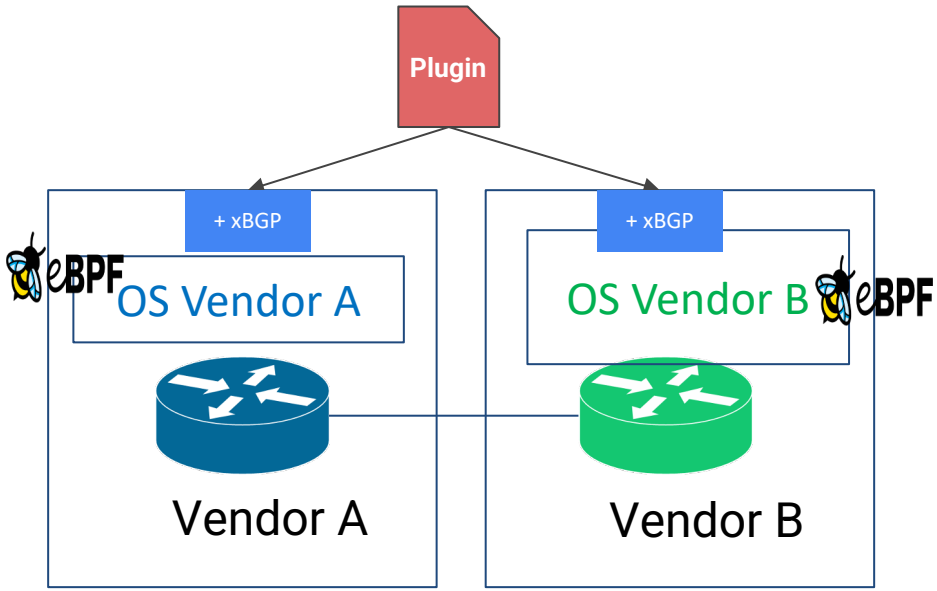
Network operators can program their routers directly using plugins.

xBGP routers expose a common API

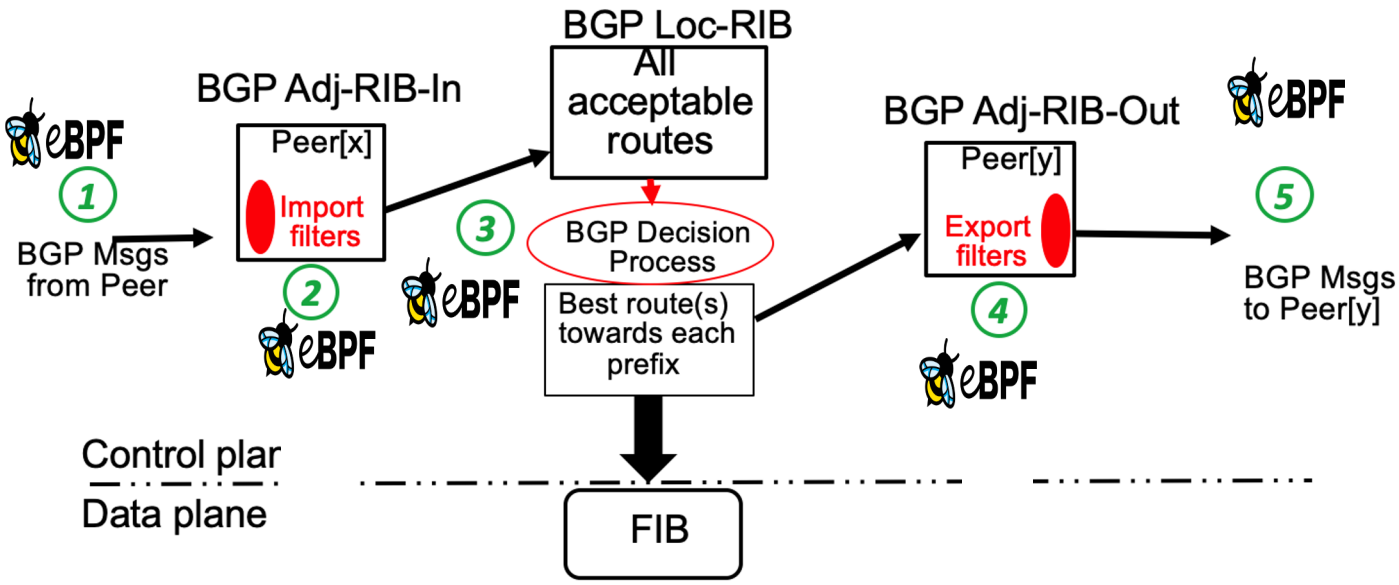


The network operator can inject plugins that program all its routers

xBGP implementations



A vendor neutral xBGP API



RFC 4271 BGP Workflow

How do we support BGP extensions ?

IDR Working Group
Internet-Draft
Intended status: Standards Track
Expires: December 2, 2016

R. Raszuk, Ed.
Bloomberg LP
R. White
Ericsson
J. Dong
Huawei Technologies
May 31, 2016

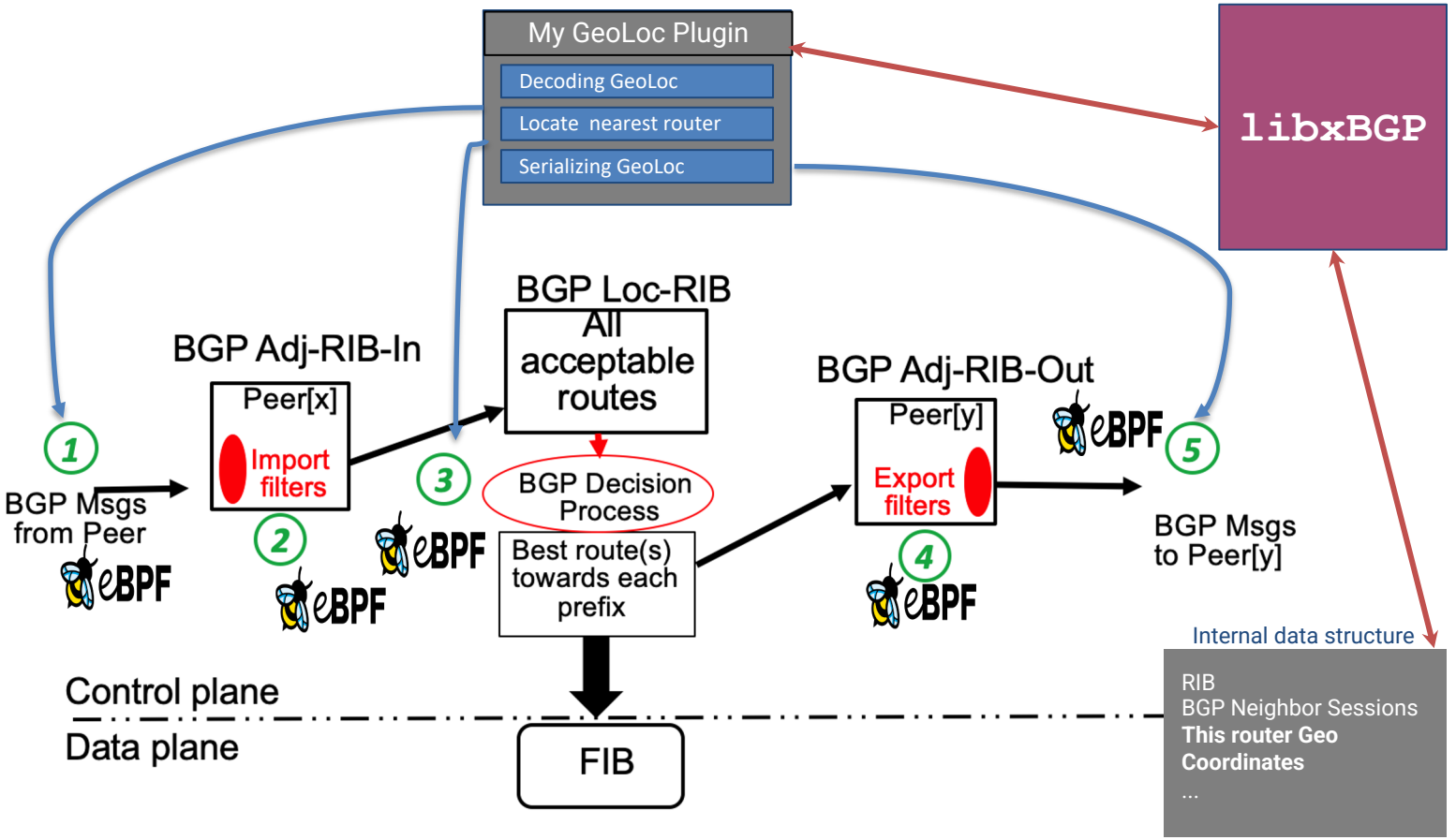
BGP Path Record Attribute
draft-raszuk-idr-bgp-pr-05

Abstract

The BGP protocol contains number of built in mechanisms which records information about the routers which have processed a specific piece of reachability information (are chosen by the protocol. and ORIGINATOR_ID attributes permanent routing loops are particular destination. How other useful information alo through which reachability i helpful to the operator in o the BGP control plane.

Table of Contents

A geoloc plugin



Implementation effort to add xBGP

- xBGP requires a little adaptation on the host BGP implementation
- We have adapted both FRRouting and BIRD to be xBGP compliant

	FRRouting (LoC)	BIRD Routing (LoC)
Modification to the codebase	30	10
Insertion Points	73	66
Plugin API	624	415
`libxbgp`	3004 + dependencies	
User Space eBPF VM	2776	

Other use cases

- Re-implementation of route reflectors (295 LoC)
- More Expressive filters
  - Route Origin Validation (126 LoC)
  - Valley Free path check for datacenters (81 LoC)
- GeoLoc attribute (261 LoC)

T. Wirtgen, Q. De Coninck, L. Vanbever, R. Bush, O. Bonaventure, *xBGP: When You Can't Wait for the IETF and Vendors*, Hotnets'20, Nov. 2020
See https://www.pluginized-protocols.org/xbgp for running source code

Monitoring : count the number of ASes contained in AS_PATH

```
uint64_t count_as_path(args_t *args) {
   unsigned int as_number = 0, segment_length;
   unsigned int *attribute_code = get_arg(ARG_CODE);
   unsigned int *as_path_len = get_arg(ARG_LENGTH);
   unsigned char *as_path = get_arg(ARG_DATA);

   if (!as_path || !as_path_len || !attribute_code) {
      // unable to fetch data from host implementation
      return EXIT_FAILURE;
   } else if (*attribute_code != AS_PATH_ATTR_ID) {
      return EXIT_FAILURE;
   }
   // core part of the plugin
   while (i < *as_path_len) {
      segment_length = as_path[i + 1];
      as_number += segment_length;
      i += (segment_length * 4) + 2;;
   }
   // log the message. If it fails, returns an error code
   if (log_msg(L_INFO "as_count:%d\n", LOG_UINT(as_number)) != 0) {
      return EXIT_FAILURE;
   }
   return EXIT_SUCCESS;
}
```

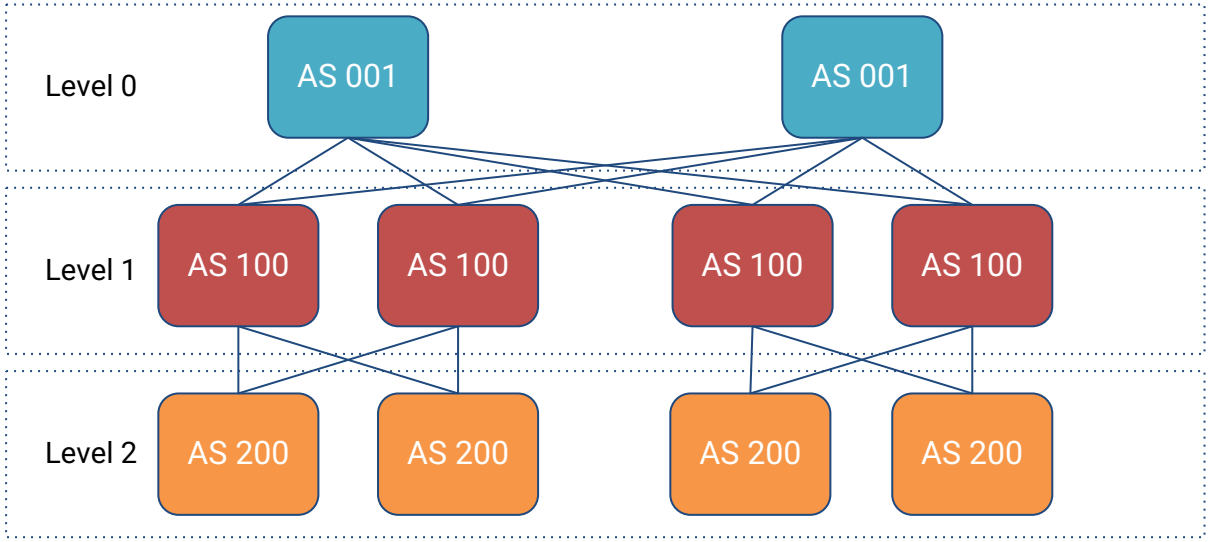
Retrieve data from the host implementation

Parse the AS-PATH attribute

Send to the logger (syslog, stdout, file, etc.)

BGP in Datacenters
Valley Free path check

RFC7938 Use of BGP for Routing in Large-Scale Data Centers



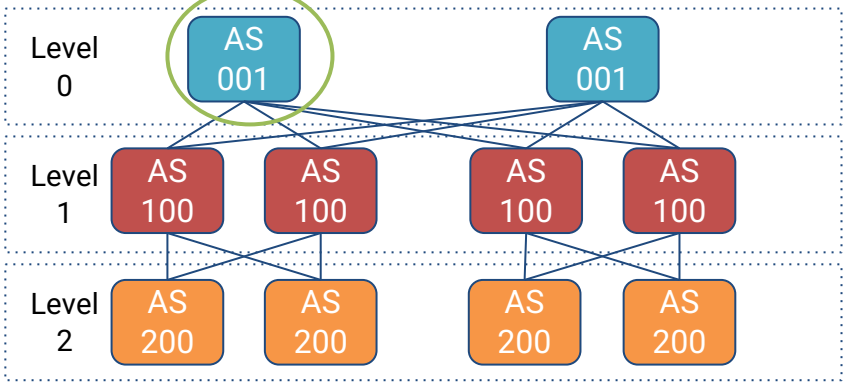
Valley Free path check

RFC7938 Use of BGP for Routing in Large-Scale Data Centers

MyRouterCli > show ip bgp

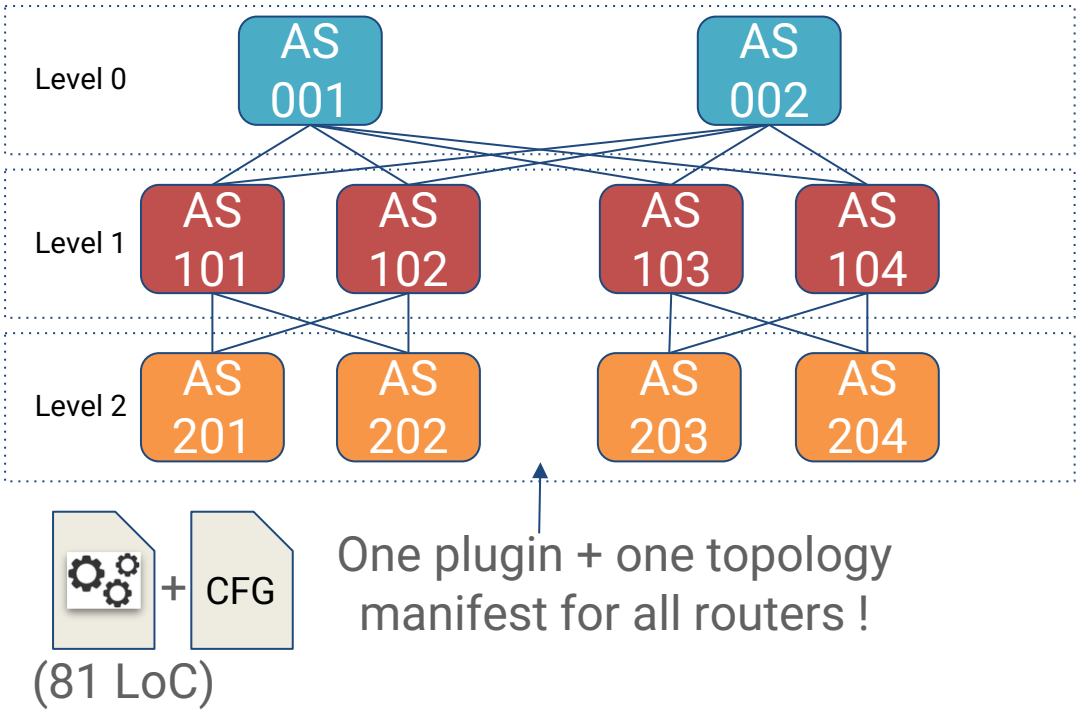
BGP Routing table information for VRF default
Router identifier 192.168.254.5, local AS number 1

Network	Next Hop	Metric	LocPref	Weight	Path
* >Ec 192.168.10.0/24	192.168.255.20	0	100	0	100 200 i
* ec 192.168.10.0/24	192.168.255.4	0	100	0	100 200 i
* >Ec 192.168.254.3/32	192.168.255.4	1	100	0	100 200 i
* ec 192.168.254.3/32	192.168.255.20	0	100	0	100 200 i
* >Ec 192.168.254.4/32	192.168.255.20	0	100	0	100 200 i



Where are these routes sourced from ?

Valley Free path check with xBGP



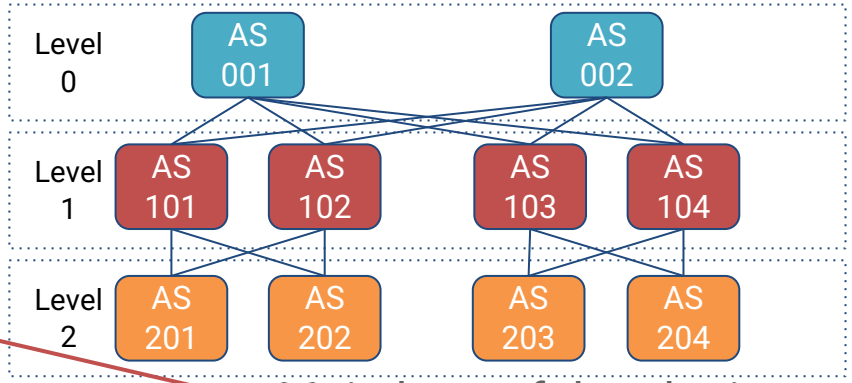
Valley Free path check with xBGP

```
uint64_t valley_free_check(args_t *args UNUSED) {
 /* variable declaration omitted  */
 attr = get_attr_from_code(AS_PATH_ATTR_CODE);
 peer = get_src_peer_info();
 if (!attr || !peer) return FAIL;

 my_as = peer->local_bgp_session->as;
 as_path = attr->data;
 as_path_len = attr->len;

 while (i < as_path_len) {
  i++; /* omit segment type  */
  segment_length = as_path[i++];
  for (j = 0; j < segment_length - 1; j++) {
   curr_as = get_u32(as_path + i);
   i += 4;
   if (!valley_check(next_as, curr_as)) return PLUGIN_FILTER_REJECT;
  }
 }
 next();
 return FAIL;
}
```

Retrieve data from the host implementation



Main loop of the plugin

The route is rejected if such a pair exists

Conclusions …

- Don’t forget that successful protocols will be extended
  - Applies to both standardized and proprietary
- Pluginized routing protocols are a paradigm shift
  - Consider each protocol as a kind of microkernel that exposes a set of functions which can be used by network operators
- Thanks to plugizined routing protocols, (small and large) network operators will be able to fulfill their own needs without having to convince vendors or the IETF

… next steps

- How to redesign routing protocols to completely leverage plugins ?
  - A simple base protocol that provides a clean API
    - Similar to microkernels, offload more complex or less frequently used functions to plugins
  - Interoperable independent implementations
    - The same plugin should work on different implementations
  - Tools and techniques to validate plugins
    - Verified plugins
  - A more efficient virtual machine
    - Webassembly, improved eBPF, other ?

For more information

- Recent papers
  - Tran, Viet-Hoang, and Olivier Bonaventure. *Beyond socket options: making the Linux TCP stack truly extensible*, Computer Communications, 2020
  - Q. Deconinck et al. *Pluginizing QUIC*, SIGCOMM'19, 2019
  - T. Wirtgen et al., *xBGP: When You Can't Wait for the IETF and Vendors*, Hotnets'20, Nov. 2020

- Blog posts and source code
  - https://www.pluginized-protocols.org