Protocol Design Considerations

Bruno Rijsman

Version 1, Presented at QuTech (Delft) on 22 November 2019

A few true stories…

Designing correct protocols is hard

On 27 October 1980, *all* routers in the entire Internet had to be rebooted *simultaneously* to recover from a protocol design weakness.

http://www.hannaypublications.com/portfolio/wp-content/uploads/2016/02/AppliedComms_A2.pdf

Version numbers to pick newest message

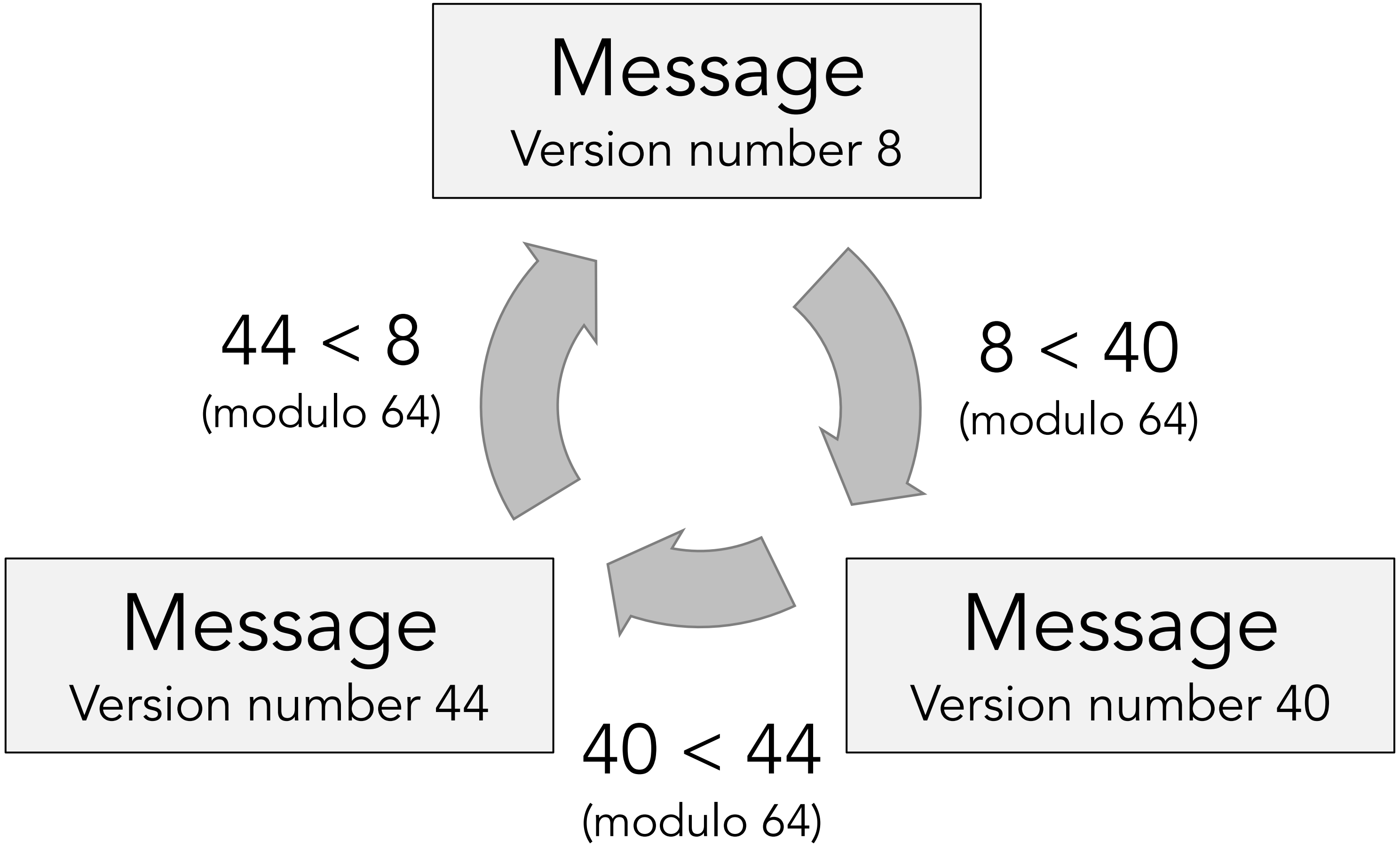
Message Version number 8	<	Message Version number 40	<	Message Version number 44

Oldest version

Newest version

Note: this mechanism is still used today, for example in OSPF and ISIS link state flooding

Modulo 64 comparison creates "version loop"



Version field was unsigned 6-bit integer, so math is modulo 64
Competing message versions kept replacing each other forever, eating up all CPU
The versions were introduced to a double bit error, which was not detected because check-sums were disabled.

We have not quite yet learned our lessons

A similar problem exists in the BGP specification and is observed in the Internet *to this day*.

See BGP persistent route oscillations (RFC3345)

Implementing protocols correctly is hard

A bug in a widely used BGP implementation allowed an attacker to advertise a BGP route that caused many BGP routers *on the other side of the world* to crash.

Sending a specially constructed 4-octet AS-path caused certain routers to crash
CVE-2014-3818 https://securitytracker.com/id/1031009

Protocol security is hard

A BGP vulnerability enabled hackers to rob a bitcoin bank.

https://www.wired.com/2014/08/isp-bitcoin-theft/

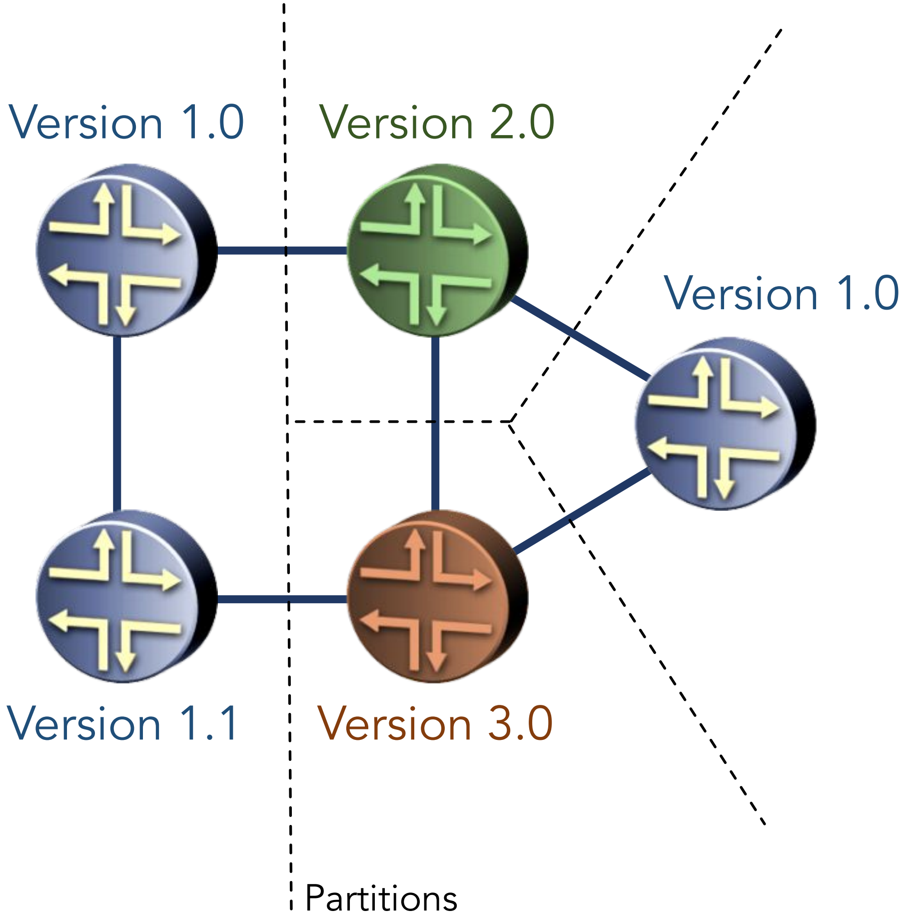
Planning for evolution

Multi-version, multi-feature, multi-vendor networks

Your protocol will evolve over time

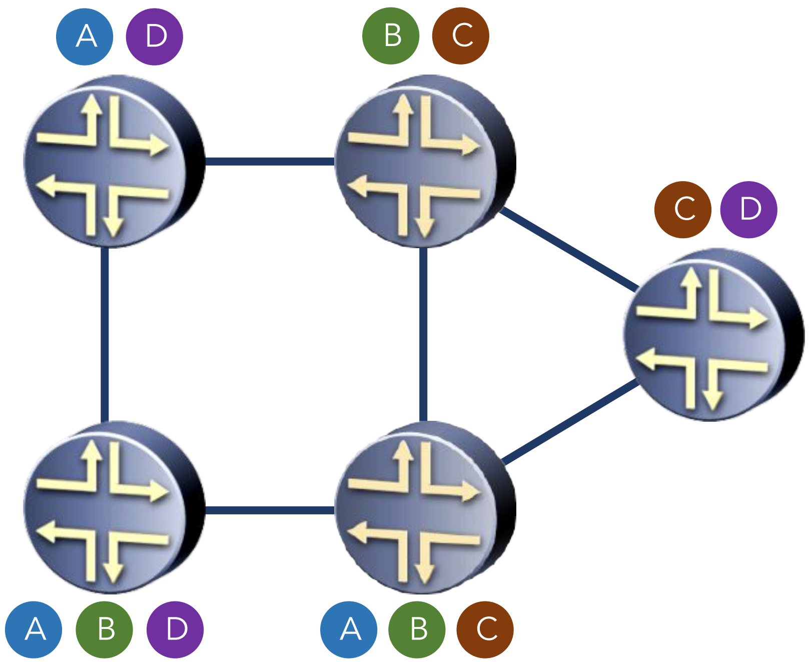
- Typical versioning scheme:
  - Minor version bump = backwards compatible
  - Major version bump = not backwards compatible
  - In practice, this does not work!
- Examples of doing it wrong:
  - IPv4 to IPv6
  - OSPFv2 to OSPFv3
- Examples of doing it right
  - ISIS
  - BGP
  - Evolve features without breaking backwards compatibility
- If you don't design your protocol with the future in mind, every new feature will be a major version bump…

The problem with major version bumps



- It's extremely hard avoiding running different major versions in your network if major bumps happen frequently.
- Temporarily during upgrade. **You cannot upgrade your devices all at once; it takes a lot of time.**
- Or semi-permanently due to operational considerations (e.g. different vendors support different versions of the protocol). **You want to be able to have multiple vendors in the same network.**
- Your network will be partitioned. Having a partitioned network makes finishing the upgrade nearly impossible.

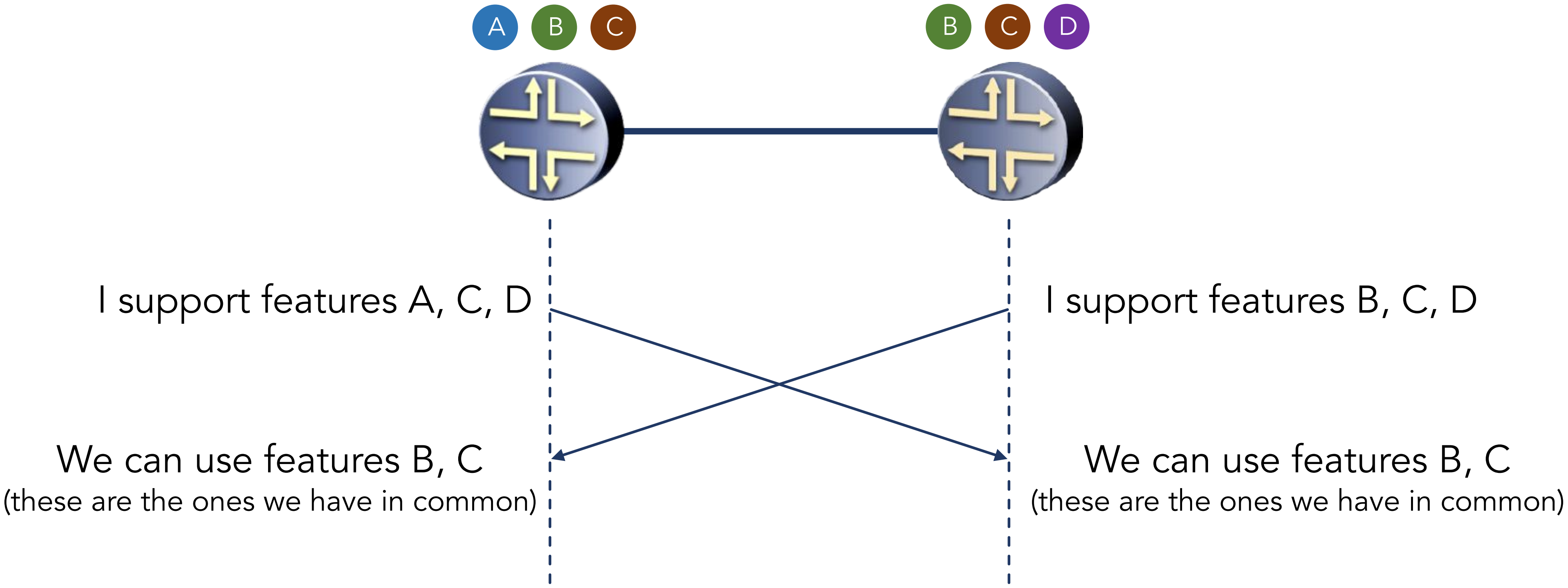
A better approach based on features



Protocol features (aka **capabilities**) A B C D

- Protocol implementation that has feature A must interoperate with protocol implementation that does not have feature A.
- Different vendors and versions of code have different feature sets.
- Don't assume the feature set can be derived from the protocol version. **Allows protocol extensions to evolve and be deployed independently.**
- Don't assume the feature set is the same. Feature set A+B must interoperate with B+C.

Capability announcement ("negotiation")



(Things get a bit more complicated when more than 2 devices are involved, e.g. flooding)

Example: BGPv4 capabilities

Capability Codes

Reference

[RFC5492]

Available Formats

CSV

Range	Registration Procedures	Note
1-63	IETF Review	
64-127	First Come First Served	
128-255	Reserved for Private Use	IANA does not assign

Value	Description	Reference
0	Reserved	[RFC5492]
1	Multiprotocol Extensions for BGP-4	[RFC2858]
2	Route Refresh Capability for BGP-4	[RFC2918]
3	Outbound Route Filtering Capability	[RFC5291]
4	Multiple routes to a destination capability (deprecated)	[RFC8277]
5	Extended Next Hop Encoding	[RFC5549]
6	BGP Extended Message	[RFC8654]
7	BGPsec Capability	[RFC8205]
8	Multiple Labels Capability	[RFC8277]
9	BGP Role (TEMPORARY - registered 2018-03-29, extension registered 2019-03-18, expires 2020-03-29)	[draft-ietf-idr-bgp-open-policy]
10-63	Unassigned	
64	Graceful Restart Capability	[RFC4724]
65	Support for 4-octet AS number capability	[RFC6793]
66	Deprecated (2003-03-06)	
67	Support for Dynamic Capability (capability specific)	[draft-ietf-idr-dynamic-cap]
68	Multisession BGP Capability	[draft-ietf-idr-bgp-multisession]
69	ADD-PATH Capability	[RFC7911]
70	Enhanced Route Refresh Capability	[RFC7313]
71	Long-Lived Graceful Restart (LLGR) Capability	[draft-uttaro-idr-bgp-persistence]
72	Unassigned	
73	FQDN Capability	[draft-walton-bgp-hostname-capability]
74-127	Unassigned	
128-255	Reserved for Private Use	[RFC5492]

Number	Description	Reference	Registration Date
0	Reserved		
1	IP (IP version 4)		
2	IP6 (IP version 6)		
3	NSAP		
4	HDLC (8-bit multidrop)		
5	BBN 1822		
6	802 (includes all 802 media plus Ethernet "canonical format")		
7	E.163		
8	E.164 (SMDS, Frame Relay, ATM)		
9	F.69 (Telex)		
10	X.121 (X.25, Frame Relay)		
11	IPX		
12	Appletalk		
13	Decnet IV		
14	Banyan Vines		
15	E.164 with NSAP format subaddress	[ATM Forum UNI 3.1. October 1995.][Andy_Malis]	
16	DNS (Domain Name System)		
17	Distinguished Name	[Charles_Lynn]	
18	AS Number	[Charles_Lynn]	
19	XTP over IP version 4	[Mike_Saul]	
20	XTP over IP version 6	[Mike_Saul]	
21	XTP native mode XTP	[Mike_Saul]	
22	Fibre Channel World-Wide Port Name	[Mark_Bakke]	
23	Fibre Channel World-Wide Node Name	[Mark_Bakke]	
24	GWID	[Subra_Hegde]	
25	AFI for L2VPN information	[RFC4761][RFC6074]	
26	MPLS-TP Section Endpoint Identifier	[RFC7212]	
27	MPLS-TP LSP Endpoint Identifier	[RFC7212]	
28	MPLS-TP Pseudowire Endpoint Identifier	[RFC7212]	
29	MT IP: Multi-Topology IP version 4	[RFC7307]	
30	MT IPv6: Multi-Topology IP version 6	[RFC7307]	
31-16383	Unassigned		
16384	EIGRP Common Service Family	[Donnie_Savage]	2008-05-13
16385	EIGRP IPv4 Service Family	[Donnie_Savage]	2008-05-13
16386	EIGRP IPv6 Service Family	[Donnie_Savage]	2008-05-13
16387	LISP Canonical Address Format (LCAF)	[David_Meyer]	2009-11-12
16388	BGP-LS	[RFC7752]	2013-03-20
16389	48-bit MAC	[RFC7042]	2013-05-06
16390	64-bit MAC	[RFC7042]	2013-05-06
16391	OUI	[RFC7961]	2013-09-25
16392	MAC/24	[RFC7961]	2013-09-25
16393	MAC/40	[RFC7961]	2013-09-25
16394	IPv6/64	[RFC7961]	2013-09-25
16395	RBridge Port ID	[RFC7961]	2013-09-25
16396	TRILL Nickname	[RFC7455]	2014-09-02
16397	Universally Unique Identifier (UUID)	[Nischal_Sheth]	2019-11-04
16398-65534	Unassigned		
65535	Reserved		

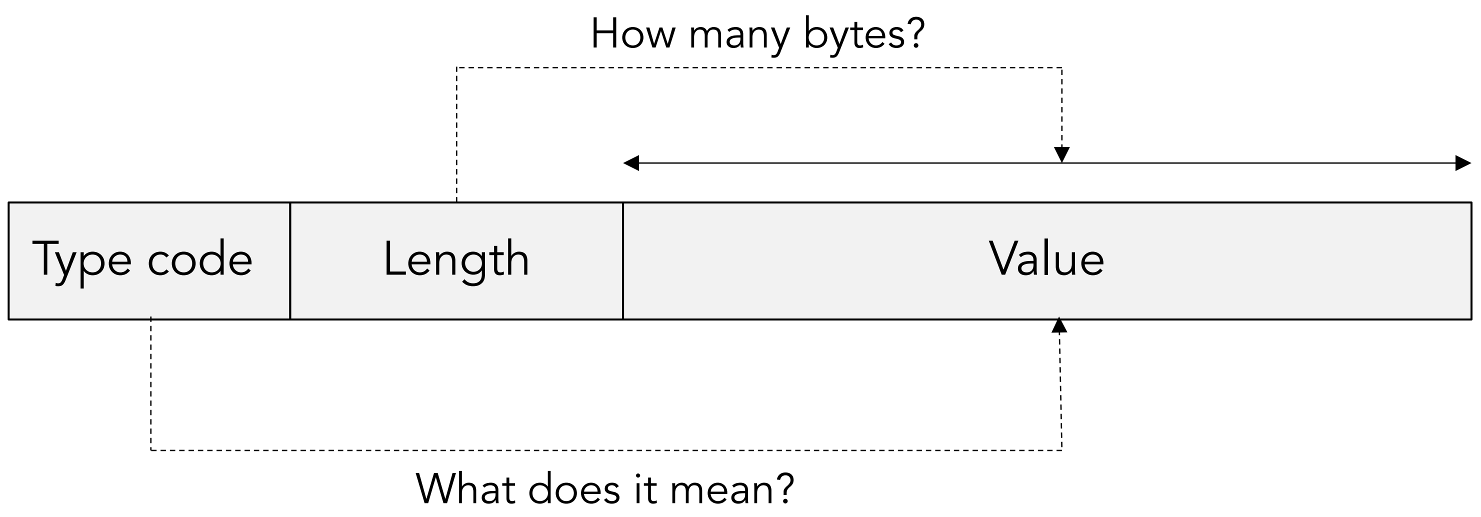
Range	Registration Procedures	Note
1-63	Standards Action	
64-127	First Come First Served	
128-240	Some recognized assignments below, others Reserved	
241-254	Reserved for Private Use	Not to be assigned

Value	Description	Reference
0	Reserved	[RFC4760]
1	Network Layer Reachability Information used for unicast forwarding	[RFC4760]
2	Network Layer Reachability Information used for multicast forwarding	[RFC4760]
3	Reserved	[RFC4760]
4	Network Layer Reachability Information (NLRI) with MPLS Labels	[RFC8277]
5	MCAST-VPN	[RFC6514]
6	Network Layer Reachability Information used for Dynamic Placement of Multi-Segment Pseudowires	[RFC7267]
7	Encapsulation SAFI	[RFC5512]
8	MCAST-VPLS	[RFC7117]
9-63	Unassigned	
64	Tunnel SAFI	[Gargi_Nalawade][draft-nalawade-kapoor-tunnel-safi-01]
65	Virtual Private LAN Service (VPLS)	[RFC4761][RFC6074]
66	BGP MDT SAFI	[RFC6037]
67	BGP 4over6 SAFI	[RFC5747]
68	BGP 6over4 SAFI	[Yong_Cui]
69	Layer-1 VPN auto-discovery information	[RFC5195]
70	BGP EVPNs	[RFC7432]
71	BGP-LS	[RFC7752]
72	BGP-LS-VPN	[RFC7752]
73	SR TE Policy SAFI	[draft-previdi-idr-segment-routing-te-policy]
74	SD-WAN Capabilities	[draft-dunbar-idr-sdwan-port-safi]
75-127	Unassigned	
128	MPLS-labeled VPN address	[RFC4364][RFC8277]
129	Multicast for BGP/MPLS IP Virtual Private Networks (VPNs)	[RFC6513][RFC6514]
130-131	Reserved	[RFC4760]
132	Route Target constrains	[RFC4684]
133	IPv4 dissemination of flow specification rules	[RFC5575]
134	VPNv4 dissemination of flow specification rules	[RFC5575]
135-139	Reserved	[RFC4760]
140	VPN auto-discovery	[draft-ietf-l3vpn-bgpvpn-auto]
141-240	Reserved	[RFC4760]

Type Length Value (TLV) Encoding

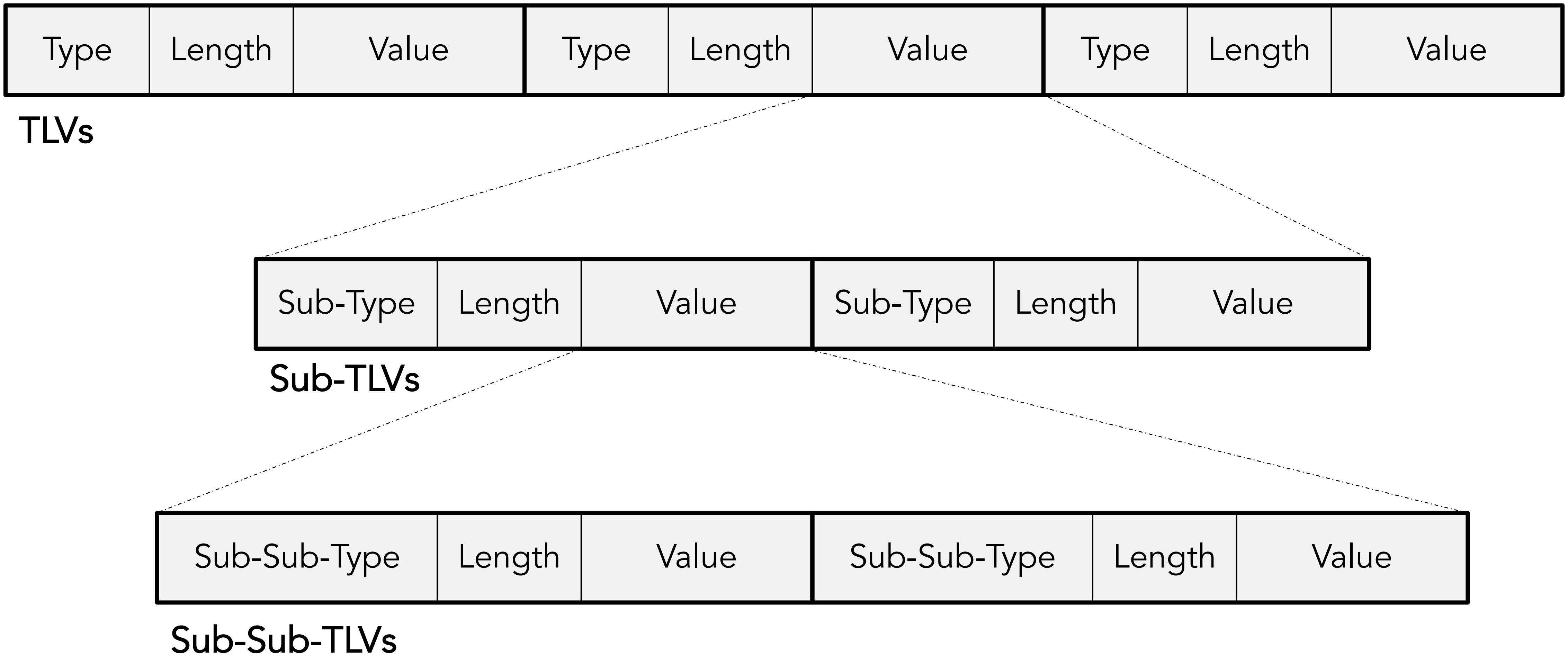
- Makes it easy for old versions of the protocol to interoperate with new versions of the protocol.
- Also avoid bugs in the code by making it easier to write parsers.
- Used in almost all modern protocols (OSPF, ISIS, BGP, RSVP, …)

Type Length Value (TLV) Encoding

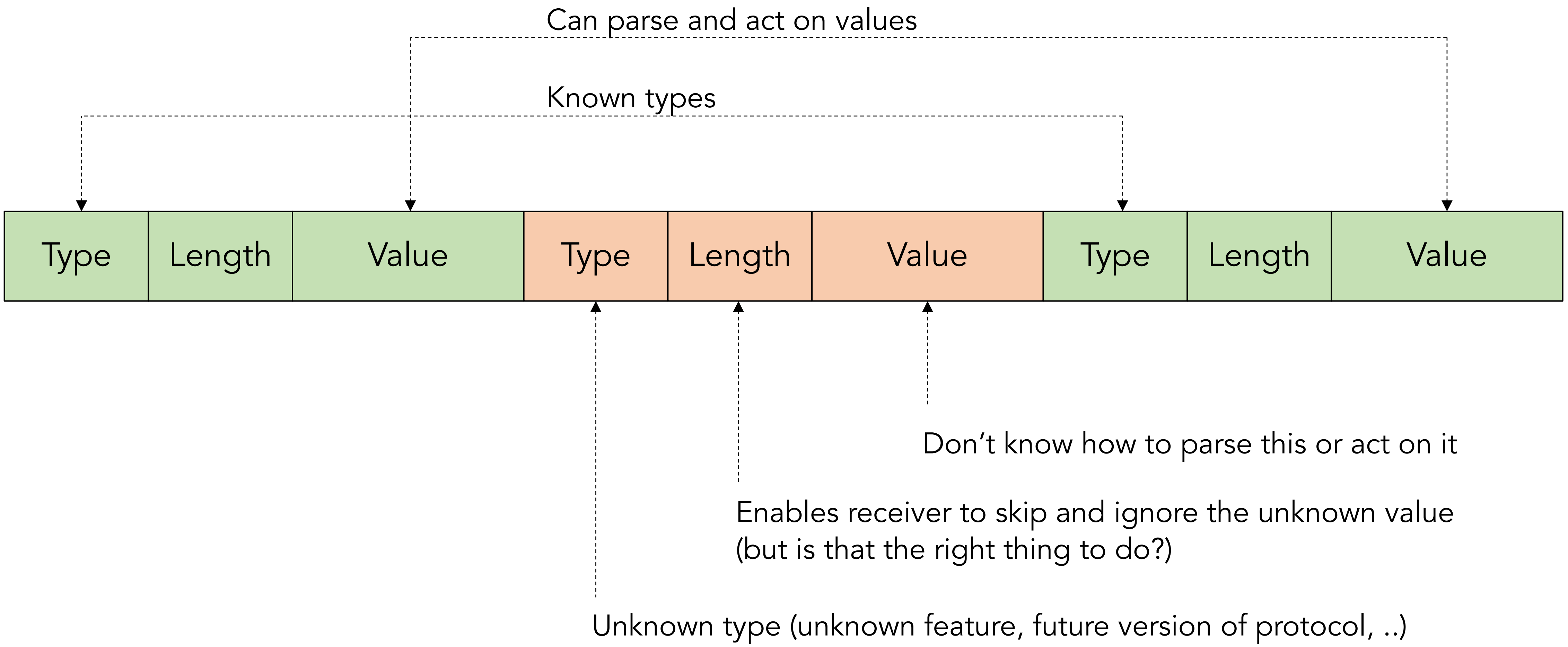


Very easy to parse; less chance of bugs in parses.

Hierarchical TLVs



Easy to skip unknown TLVs



Example: IANA ISIS TLV code point registry

TLV Codepoints Registry

Registration Procedure(s)
Expert Review

Expert(s)
Chris Hopps, Hannes Gredler, Les Ginsberg

Reference
[RFC3563][RFC6233][RFC7356]

Note
IETF SHALL keep JTC1/SC6 informed of TLV codepoint values allocated, and JTC1/SC6 SHALL refer allocation requests arising within JTC1 constituencies to the IANA registry process.

Note
Codepoints greater than 255 can only be used in PDUs designated to support extended TLVs.

Available Formats
CSV

Value	Name	IIH	LSP	SNP	Purge	Status/Reference
0	Reserved					
1	Area Addresses	y	y	n	n	[ISO 10589, "Intermediate System to Intermediate System Intra- Domain Routeing Exchange Protocol for use in Conjunction with the Protocol for Pr Service (ISO 8473)", International Standard 10589: 2002, Second Edition, 2002.]
2	IIS Neighbors	n	y	n	n	[ISO 10589, "Intermediate System to Intermediate System Intra- Domain Routeing Exchange Protocol for use in Conjunction with the Protocol for Pr Service (ISO 8473)", International Standard 10589: 2002, Second Edition, 2002.]
3	ES Neighbors	n	y	n	n	[ISO 10589, "Intermediate System to Intermediate System Intra- Domain Routeing Exchange Protocol for use in Conjunction with the Protocol for Pr Service (ISO 8473)", International Standard 10589: 2002, Second Edition, 2002.]
4	Part. DIS	n	y	n	n	[ISO 10589, "Intermediate System to Intermediate System Intra- Domain Routeing Exchange Protocol for use in Conjunction with the Protocol for Pr Service (ISO 8473)", International Standard 10589: 2002, Second Edition, 2002.]
5	Prefix Neighbors	n	y	n	n	[ISO 10589, "Intermediate System to Intermediate System Intra- Domain Routeing Exchange Protocol for use in Conjunction with the Protocol for Pr Service (ISO 8473)", International Standard 10589: 2002, Second Edition, 2002.]
6	IIS Neighbors	y	n	n	n	[ISO 10589, "Intermediate System to Intermediate System Intra- Domain Routeing Exchange Protocol for use in Conjunction with the Protocol for Pr Service (ISO 8473)", International Standard 10589: 2002, Second Edition, 2002.]
7	Instance Identifier	y	y	y	y	[RFC8202]
8	Padding	y	n	n	n	[ISO 10589, "Intermediate System to Intermediate System Intra- Domain Routeing Exchange Protocol for use in Conjunction with the Protocol for Pr Service (ISO 8473)", International Standard 10589: 2002, Second Edition, 2002.]
9	LSP Entries	n	n	y	n	[ISO 10589, "Intermediate System to Intermediate System Intra- Domain Routeing Exchange Protocol for use in Conjunction with the Protocol for Pr Service (ISO 8473)", International Standard 10589: 2002, Second Edition, 2002.]
10	Authentication	y	y	y	y	[ISO 10589, "Intermediate System to Intermediate System Intra- Domain Routeing Exchange Protocol for use in Conjunction with the Protocol for Pr Service (ISO 8473)", International Standard 10589: 2002, Second Edition, 2002.][RFC6233]
11	ESN TLV	y	n	y	n	[RFC7602]
12	Opt. Checksum	y	n	y	n	[RFC3358]
13	Purge Originator Identification	n	y	n	y	[RFC6232]
14	LSPBufferSize	n	y	n	n	[ISO 10589, "Intermediate System to Intermediate System Intra- Domain Routeing Exchange Protocol for use in Conjunction with the Protocol for Pr Service (ISO 8473)", International Standard 10589: 2002, Second Edition, 2002.]
15	Router-Fingerprint	y	y	n	y	[RFC8196]
16	Reverse Metric	y	n	n	n	[RFC8500]
17	IS-IS Area Node IDs TLV (TEMPORARY - registered 2019-08-08, expires 2020-08-08)	n	y	n	n	[draft-ietf-lsr-dynamic-flooding]
18	IS-IS Flooding Path TLV (TEMPORARY - registered 2019-08-08, expires 2020-08-	n	y	n	n	[draft-ietf-lsr-dynamic-flooding]

How to handle unrecognized TLVs?

- What should you do when you receive a TLV with an unknown type?
- Happens when another device is running a newer version of the protocol.
- Options:
  - Ignore and propagate the TLV
  - Ignore and remove the TLV
  - Declare an error
  - Note: ignoring is possible because the length L is known and the value can be skipped
- The originator of the TLV (who by definition understands it) sets some extra bits in the TLV to instruct the receiver what to do.

Example: Label Distribution Protocol (LDP)

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|U|F|        Type               |            Length             |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                                                               |
|                             Value                             |
~                                                               ~
|                                                               |
|                               +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

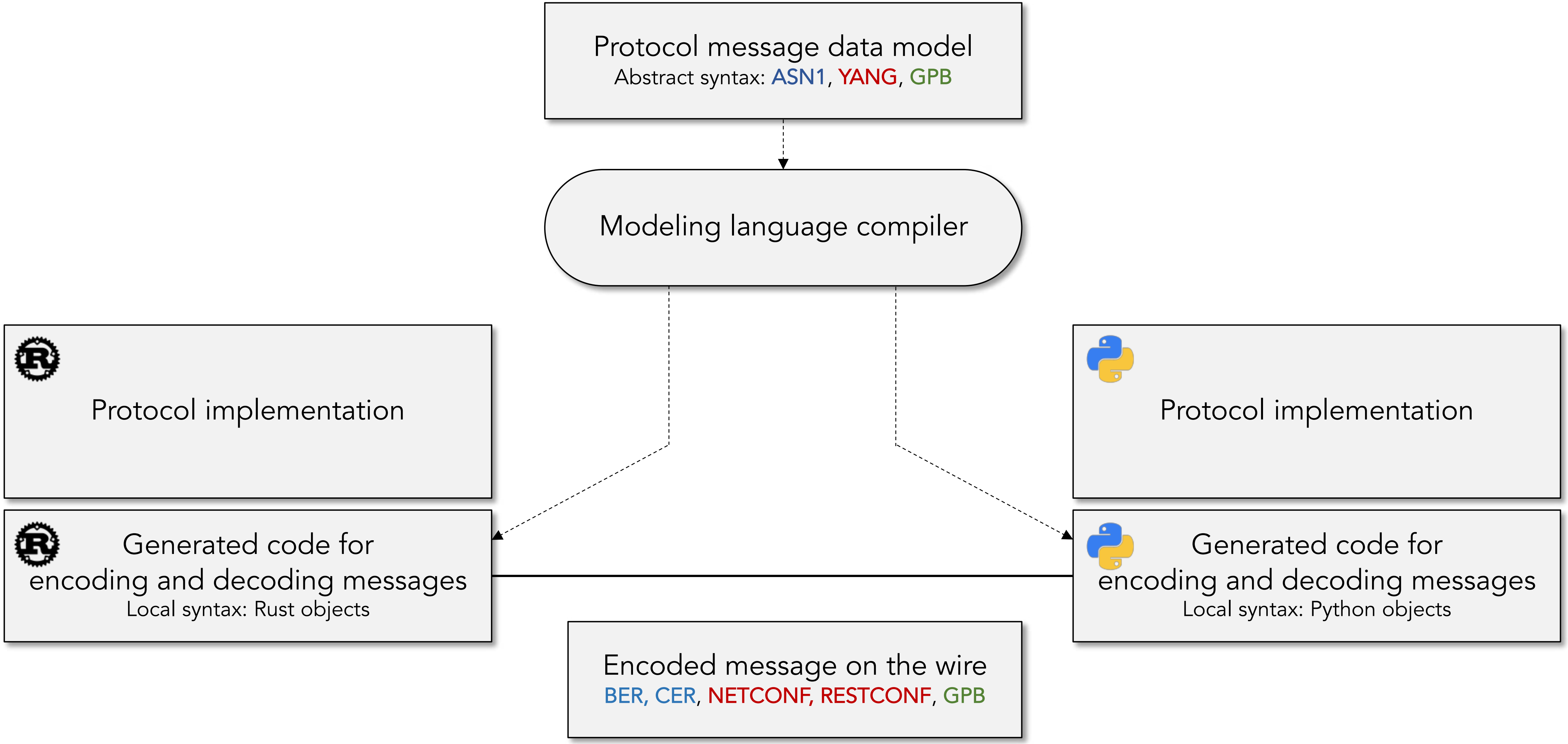
U-bit
   Unknown TLV bit. Upon receipt of an unknown TLV, if U is clear (=0), a notification MUST be returned to the message originator and the entire message MUST be ignored; if U is set (=1), the unknown TLV MUST be silently ignored and the rest of the message processed as if the unknown TLV did not exist. The sections following that define TLVs specify a value for the U-bit.

F-bit
   Forward unknown TLV bit. This bit applies only when the U-bit is set and the LDP message containing the unknown TLV is to be forwarded. If F is clear (=0), the unknown TLV is not forwarded with the containing message; if F is set (=1), the unknown TLV is forwarded with the containing message. The sections following that define TLVs specify a value for the F-bit. By setting both the U- and F-bits, a TLV can be propagated as opaque data through nodes that do not recognize the TLV.

TLV encoding is also used in:

- BGP
- OSPF
- ISIS
- RSVP
- Many more…

Beyond TLVs: formal modeling languages



Examples

Modeling language	Protocols
ASN.1	Simple Network Management Protocol (SNMP)
ASN.1	Light-weight Directory Access Protocol (LDAP)
YANG	Network Configuration Protocol (NETCONF)
Protobuf	GRPC Network Management Interface (gNMI)
Thrift	Routing In Fat Trees (RIFT)

Examples

ASN.1 model for LDAP protocol

```
SearchRequest ::= [APPLICATION 3] SEQUENCE {
     baseObject       LDAPDN,
     scope            ENUMERATED {
          baseObject                 (0),
          singleLevel                (1),
          wholeSubtree               (2),
          ...  },
     derefAliases     ENUMERATED {
          neverDerefAliases          (0),
          derefInSearching           (1),
          derefFindingBaseObj        (2),
          derefAlways                (3) },
     sizeLimit        INTEGER (0 ..  maxInt),
     timeLimit        INTEGER (0 ..  maxInt),
     typesOnly        BOOLEAN,
     filter           Filter,
     attributes       AttributeSelection }
```

YANG model for interface management

```
/*
 * Configuration data nodes
 */

container interfaces {
  description
    "Interface configuration parameters.";

  list interface {
    key "name";

    description
      "The list of configured interfaces on the device.

       The operational state of an interface is available in the
       /interfaces-state/interface list.  If the configuration of a
       system-controlled interface cannot be used by the system
       (e.g., the interface hardware present does not match the
       interface type), then the configuration is not applied to
       the system-controlled interface shown in the
       /interfaces-state/interface list.  If the configuration
       of a user-controlled interface cannot be used by the system,
       the configured interface is not instantiated in the
       /interfaces-state/interface list.";

   leaf name {
      type string;
      description
        "The name of the interface.

         A device MAY restrict the allowed values for this leaf,
         possibly depending on the type of the interface.
```

Thrift model for RIFT protocol

```
union TIEElement {
    /** used in case of enum common.TIETypeType.NodeTIEType */
    1: optional NodeTIEElement            node;
    /** used in case of enum common.TIETypeType.PrefixTIEType */
    2: optional PrefixTIEElement          prefixes;
    /** positive prefixes (always southbound)
        It MUST NOT be advertised within a North TIE and ignored otherwise
    */
    3: optional PrefixTIEElement          positive_disaggregation_prefixes;
    /** transitive, negative prefixes (always southbound) which
        MUST be aggregated and propagated
        according to the specification
        southwards towards lower levels to heal
        pathological upper level partitioning, otherwise
        blackholes may occur in multiplane fabrics.
        It MUST NOT be advertised within a North TIE.
    */
    5: optional PrefixTIEElement          negative_disaggregation_prefixes;
    /** externally reimported prefixes */
    6: optional PrefixTIEElement          external_prefixes;
    /** positive external disaggregated prefixes (always southbound).
        It MUST NOT be advertised within a North TIE and ignored otherwise
    */
    7: optional PrefixTIEElement          positive_external_disaggregation_prefixes;
    /** Key-Value store elements */
    9: optional KeyValueTIEElement        keyvalues;
}

/** TIE packet */
struct TIEPacket {
    1: required TIEHeader  header;
    2: required TIEElement element;
}

/** content of a RIFT packet */
union PacketContent {
    1: optional LIEPacket     lie;
    2: optional TIDEPacket    tide;
    3: optional TIREPacket    tire;
    4: optional TIEPacket     tie;
}
```

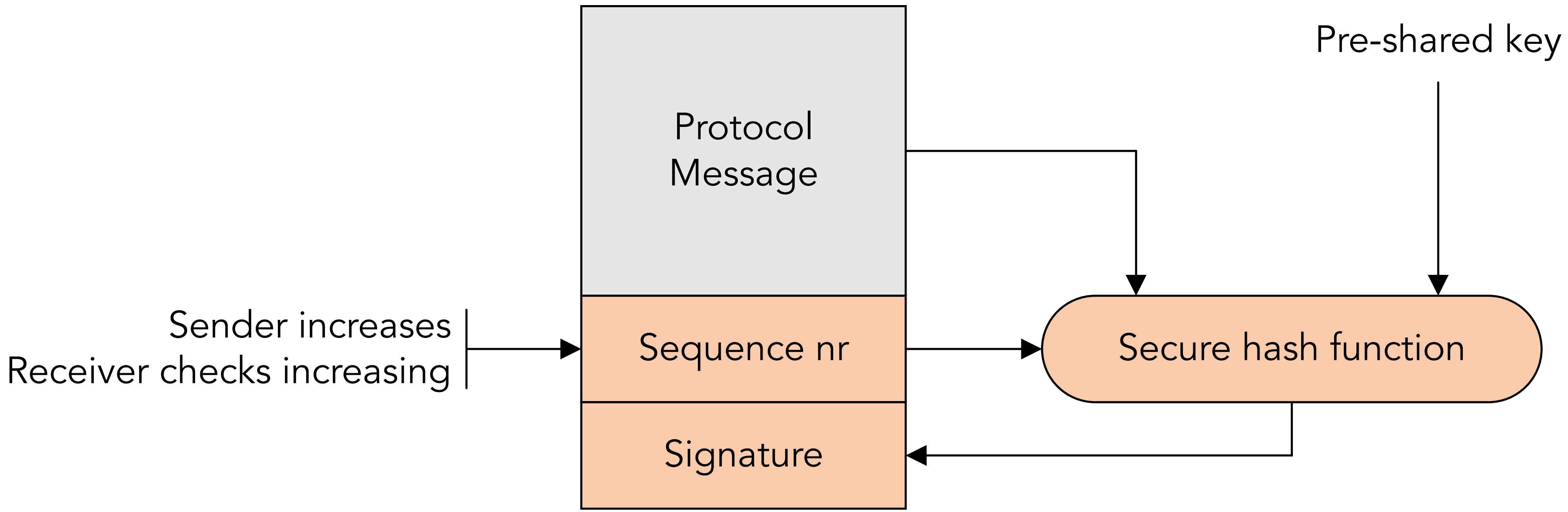
Control protocol security

Control plane security

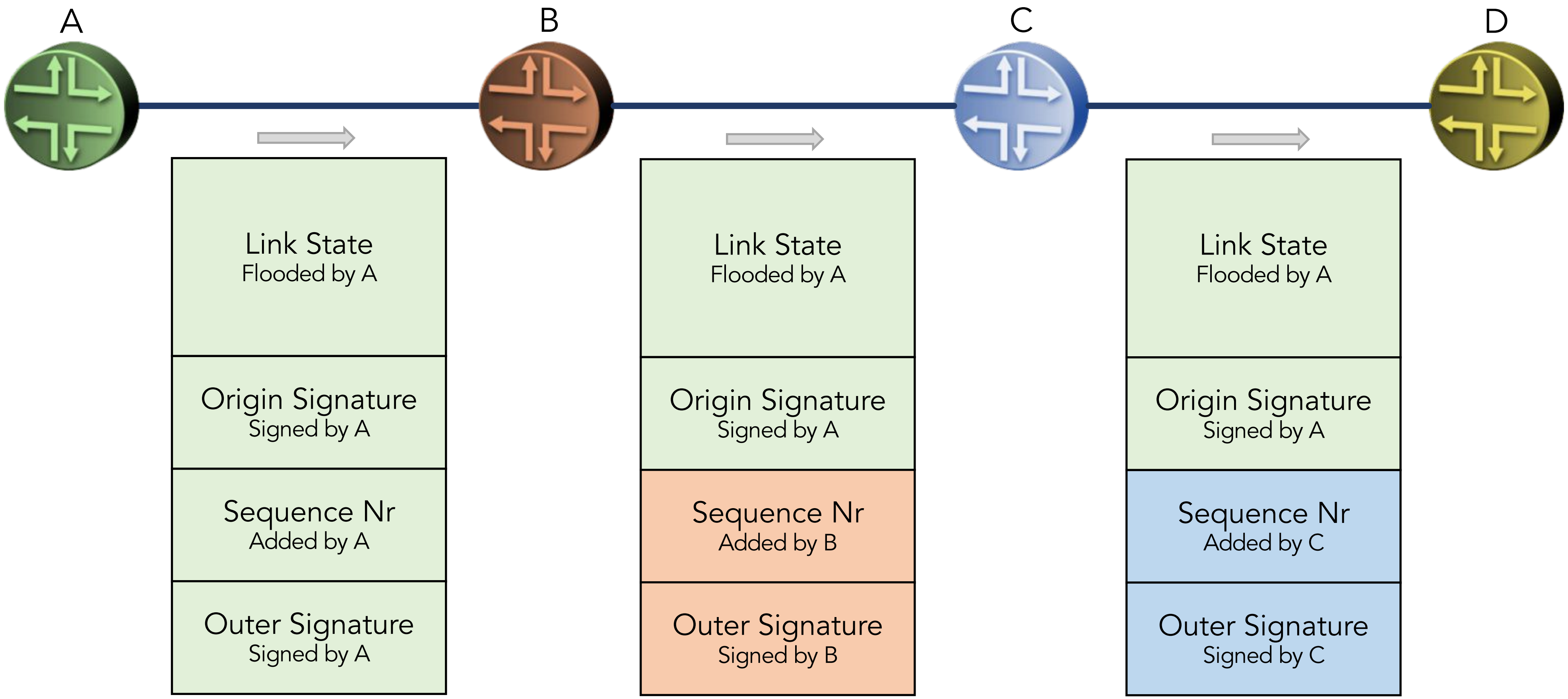
- Authentication
  - Is the router that I am talking to really the router it claims to be?
- Integrity
  - Are the routing messages modified or spoofed by an adversary?
  - Special case: replay attack
- Confidentiality?
  - Encrypt the protocol messages (often **not** a requirement)
- Authorization
  - Is this router allowed to have an adjacency with me?
  - Is this router allowed to advertise this route?
    - If it originates the route, can it prove it owns the prefix?
    - If it propagated the route, can it prove that the advertised path is correct?

Neighbor authentication and integrity

- Pre-shared key configured on each router
- Sign each message with secure hash using pre-shared key
- Add sequence number to prevent replay attacks
- Message often not encrypted

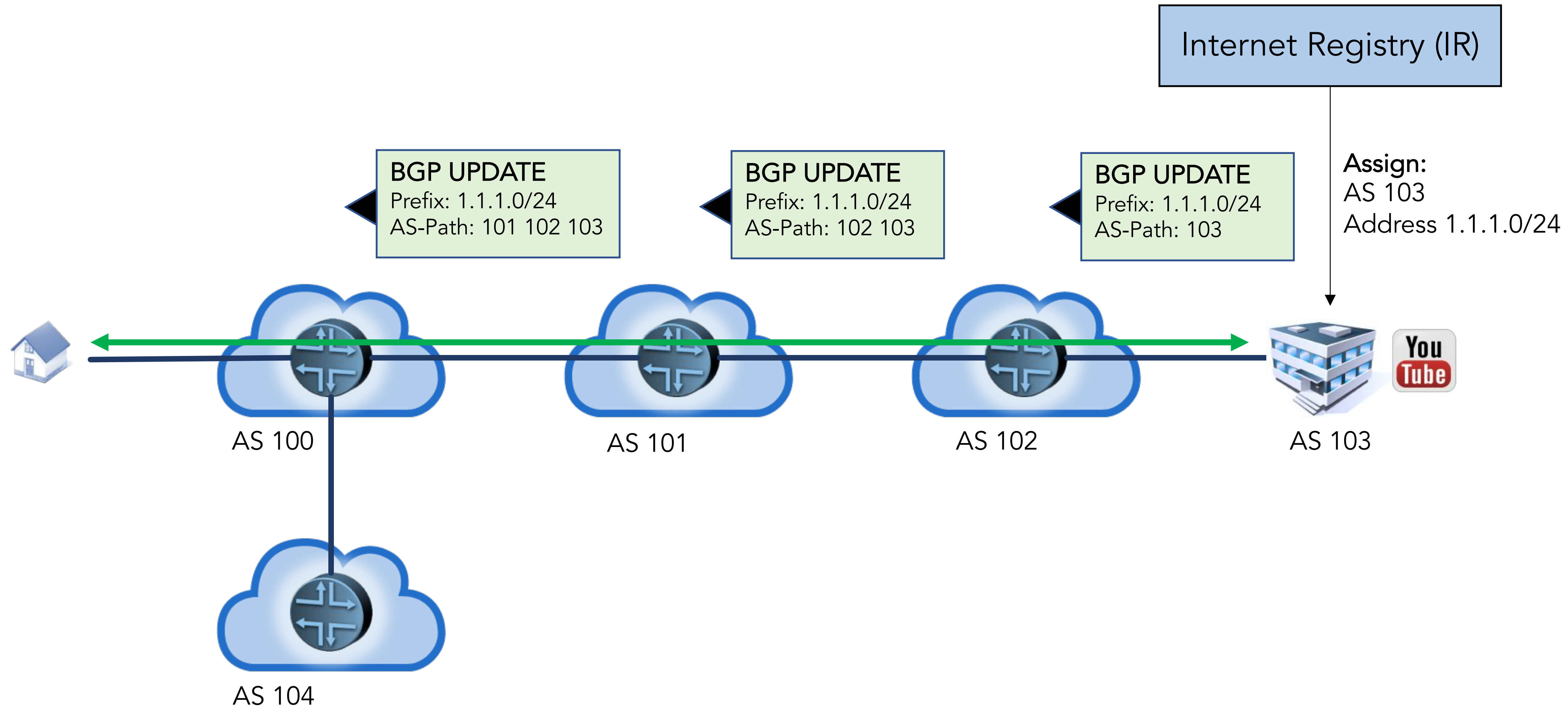


Link state flooding: origin and outer key

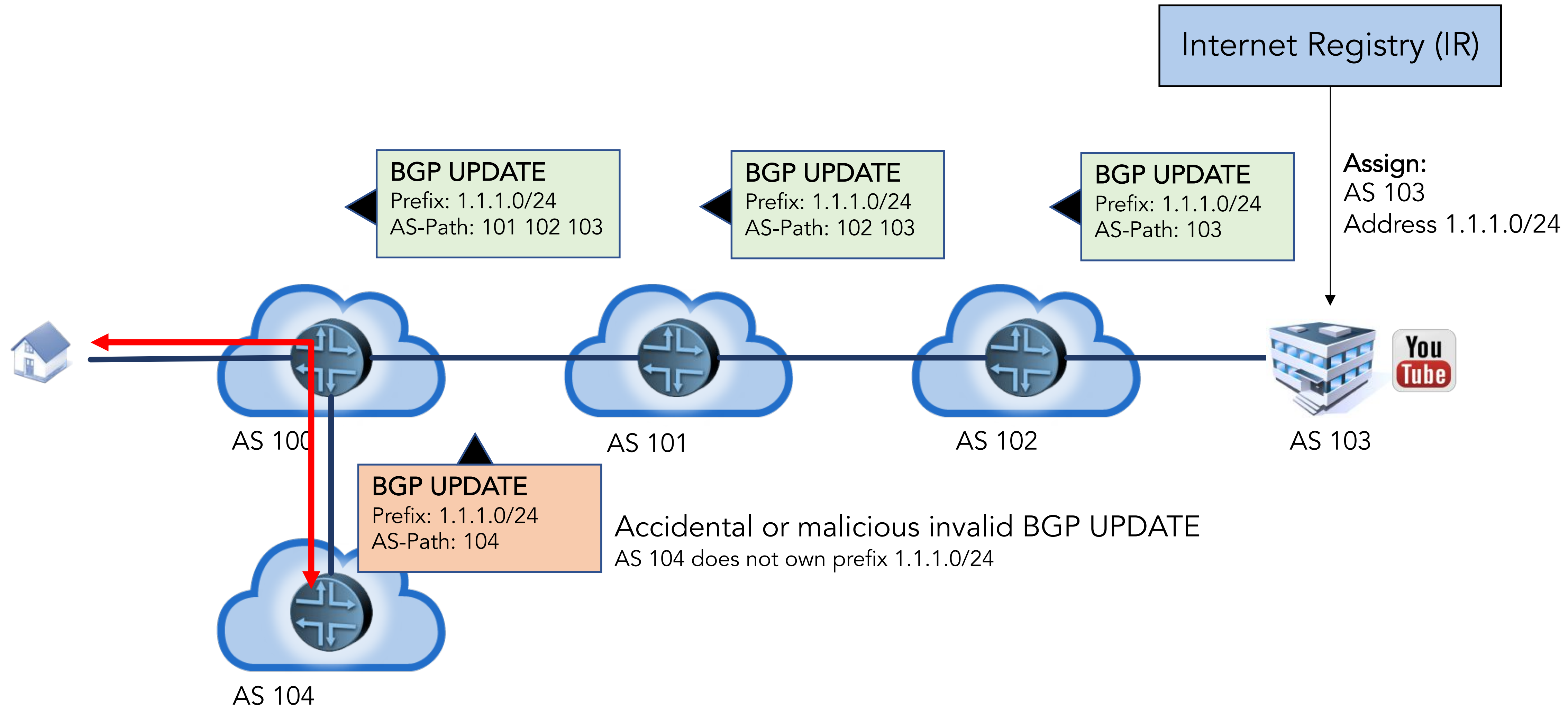


Every router in the networks must have the same pre-shared origin key *(not feasible for IDR)*
Pre-shared outer keys can be different per link, but are the same in practice
Pre-shared key roll-over is very painful; often does not happen in practice

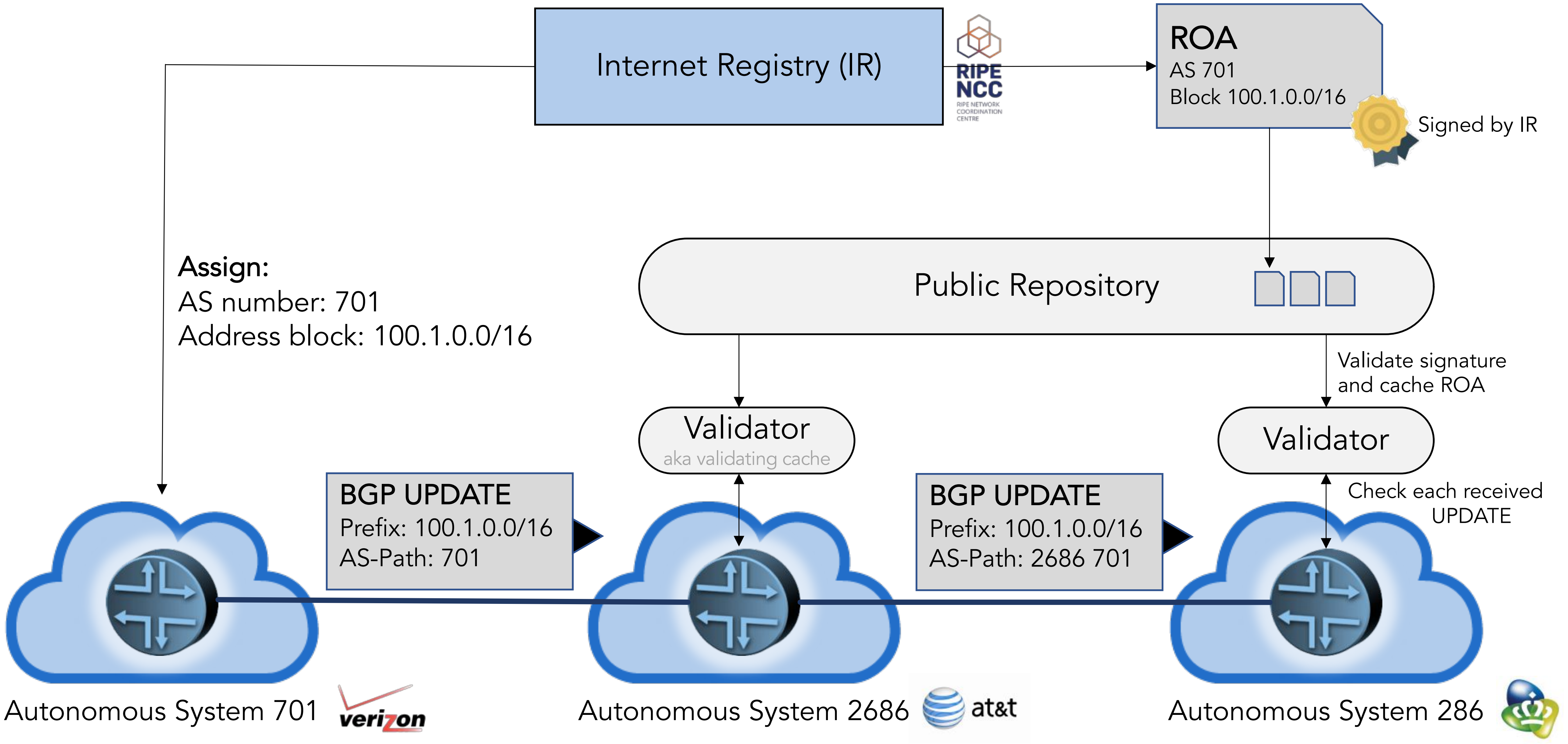
BGP prefix hijacking



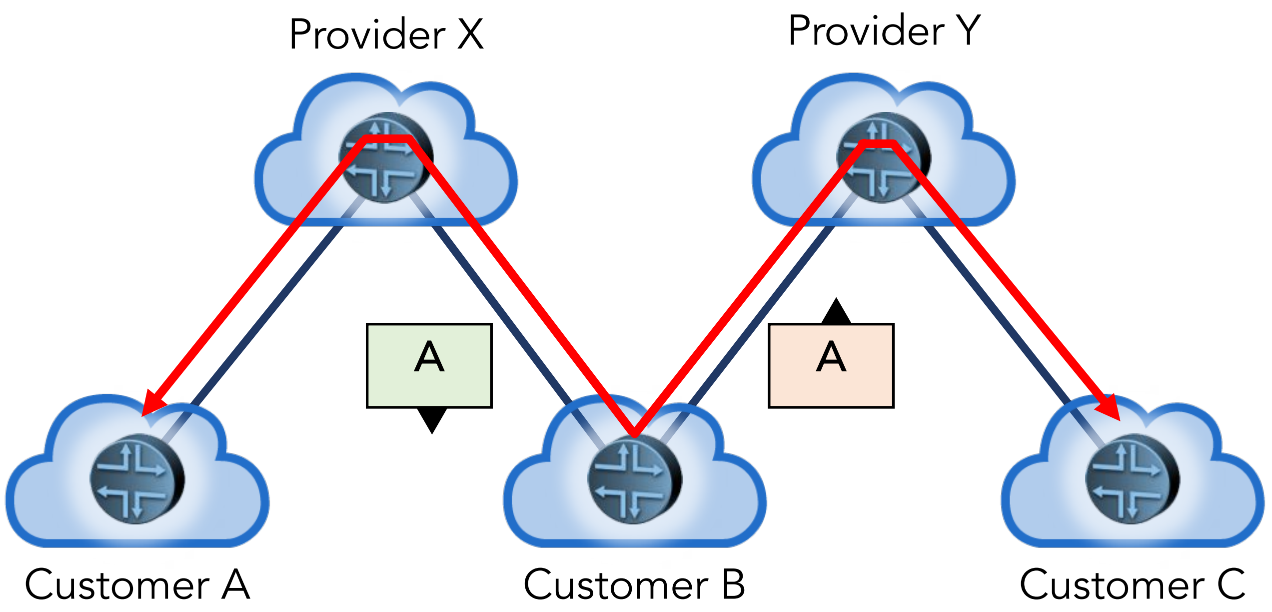
BGP prefix hijacking



BGP route origin authentication (ROA) (Simplified)



Also need BGP path validation



Accidental violation of valley-fee routing

Y should not do transit through customer B

B should not advertise A to Y

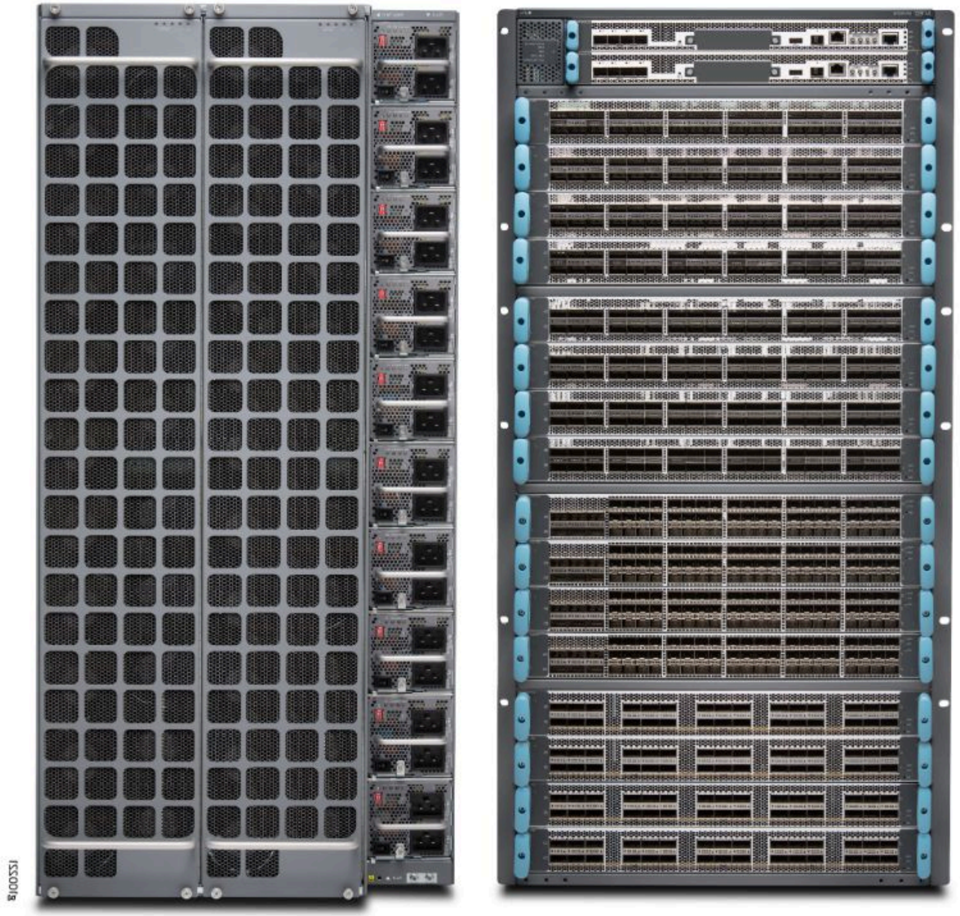
Y should not accept A from B

(There are also malicious flavors of this, e.g. BGP poisoning)

Upgrades, crashes

There is no maintenance window for the Internet

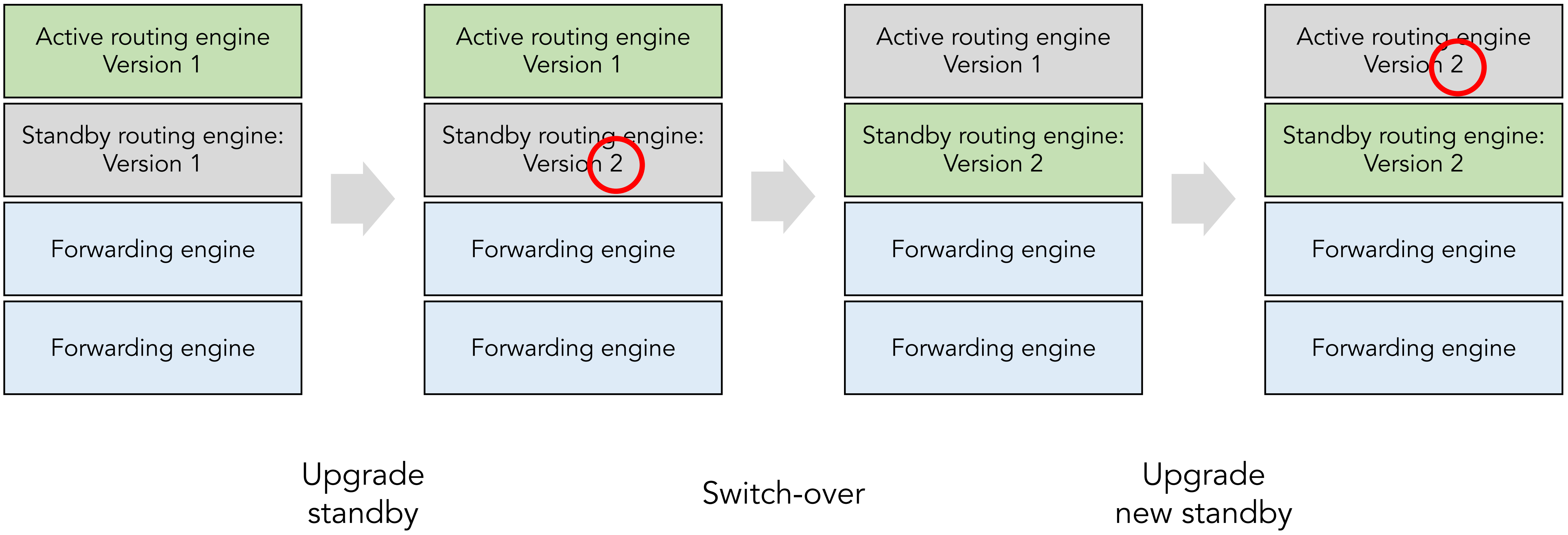
Chassis devices



Redundant control planes
(routing engines)

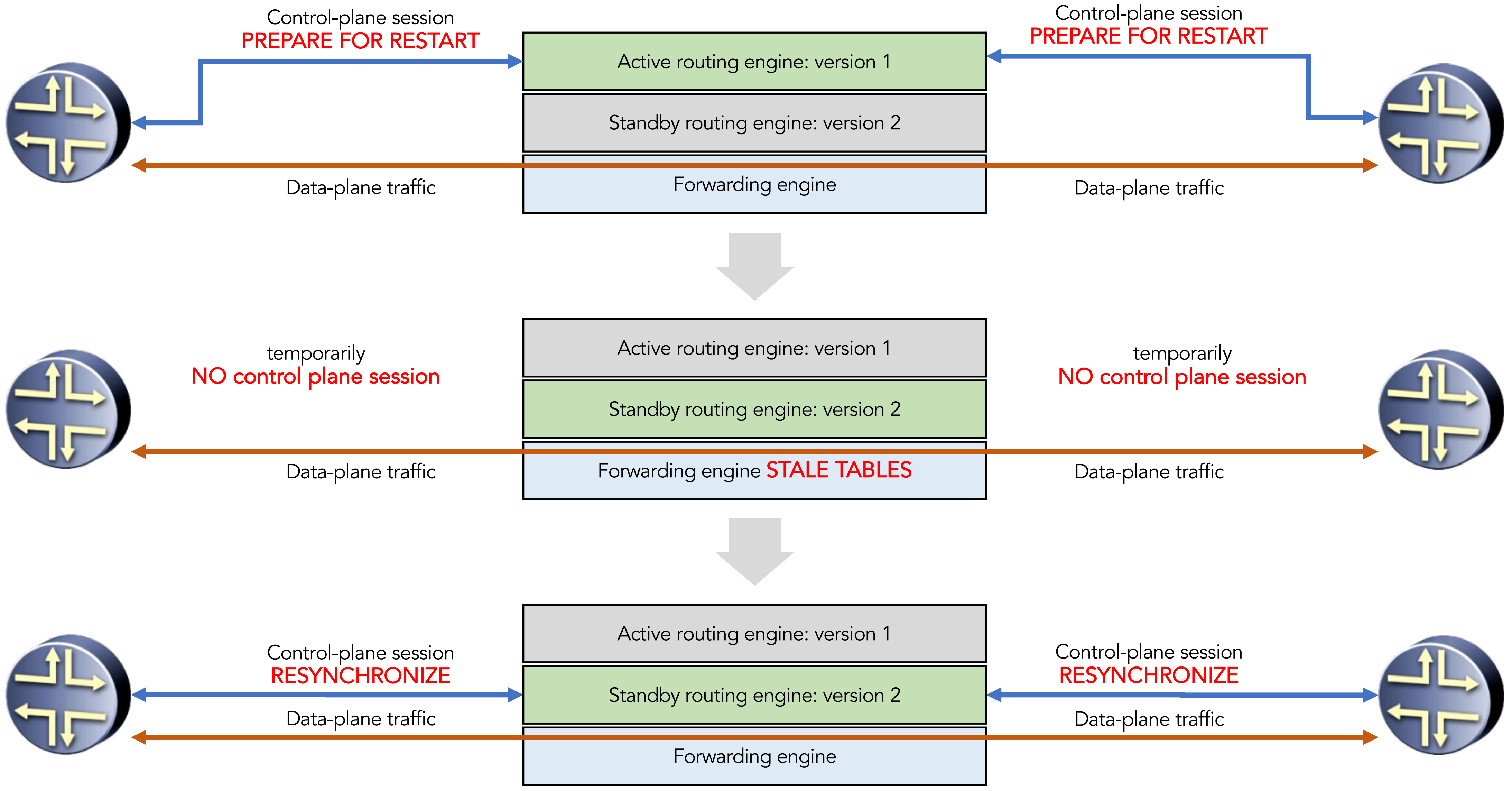
Everything (routing engine, switch fabric, power supplies, fans, …) redundant and in-service replaceable

In-Service Software Upgrade (ISSU)

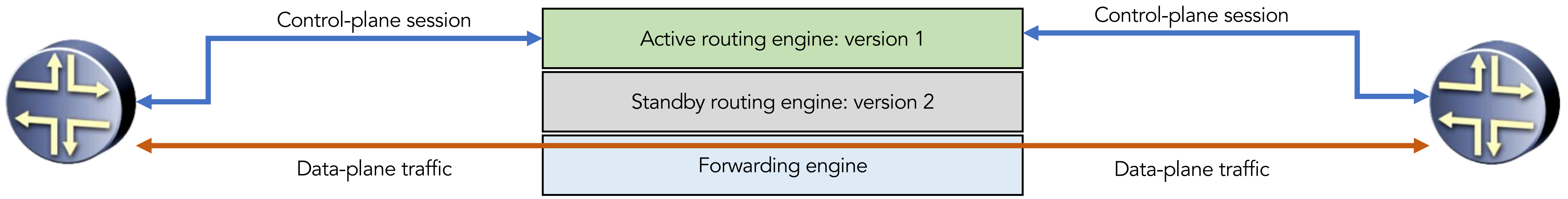


Non-Stop Forwarding (NSF)

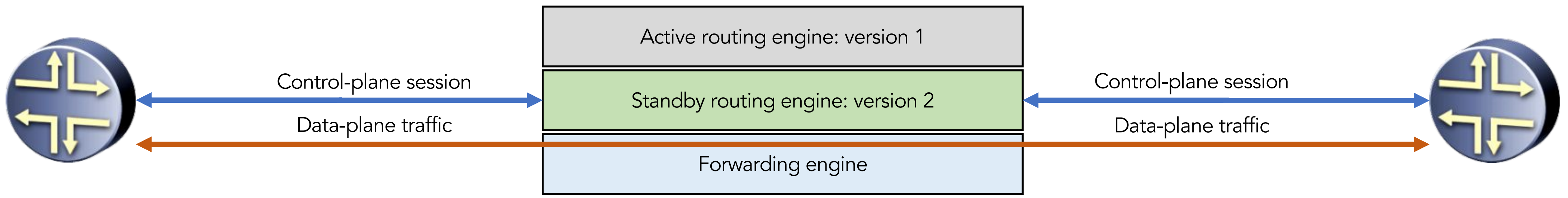
Also known as Graceful Restart (GR)



Non-Stop Routing (NSR)



Switchover a LIVE control plane session (TCP connection)
Incredibly difficult; requires constant synchronization of TCP state
Cause more problems than it solves in real life.
Modern approach: use scale-out instead of scale-up (when possible)



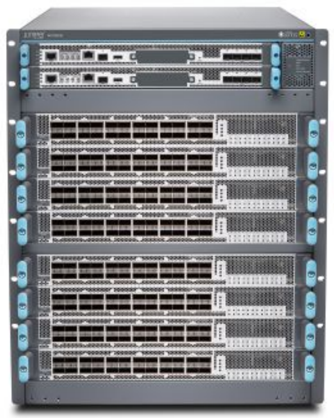
Pizza-box devices

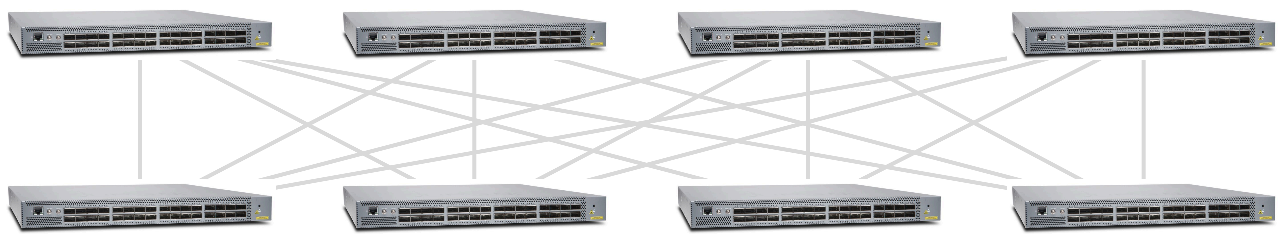
Does not have redundant routing hardware routing engines.

(Sometimes in redundant software routing engines using virtual machines or containers.)

Cheap commodity hardware

Scale-out instead of scale-up





Scale-up

- One large chassis switch
- Internal switch fabric
- If switch fails, all is lost
- NSF or NSR for upgrade
- Everything must be redundant
- Expensive
- Complex

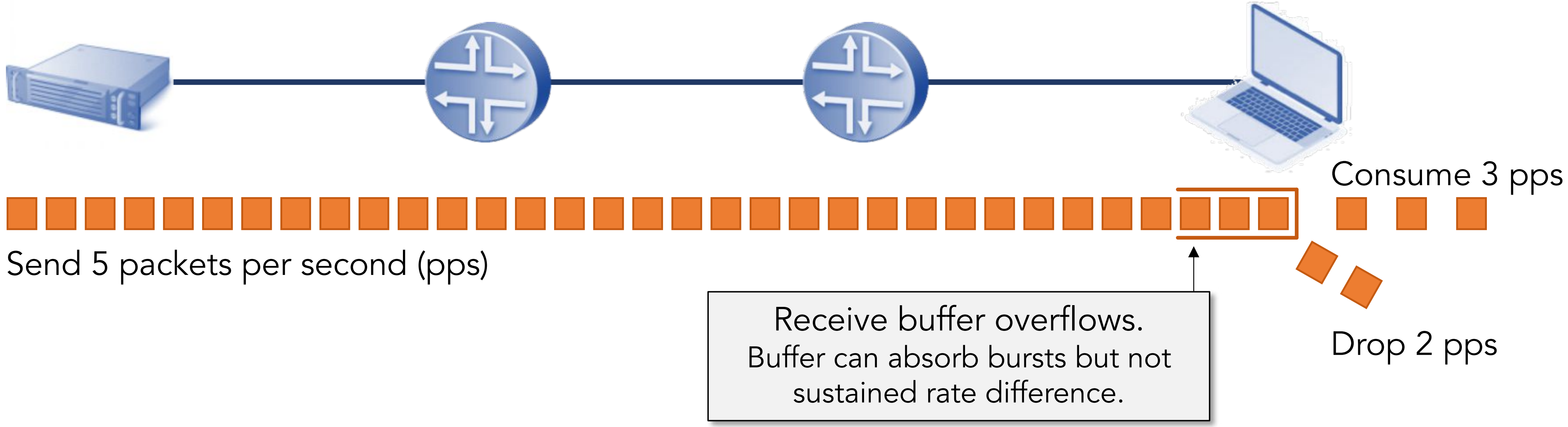
Scale-out

- Multiple small pizza-box switches
- Clos fabric
- If one switch fails, only 1/Nth of capacity lost
- Rolling upgrade
- No redundancy needed inside switch
- Cheap
- Simple

Flow control
Congestion control
Error control
in the **data** plane

Flow control

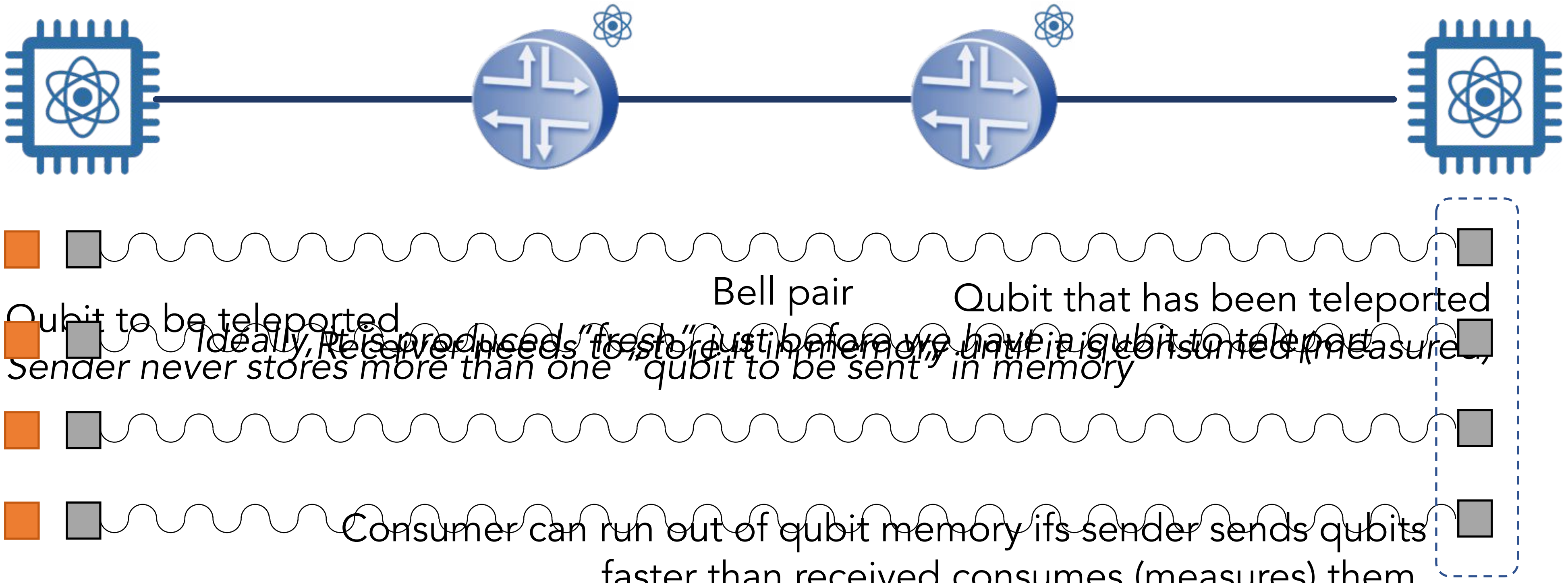
Make sure the sender does not produce data faster than the receiver can consume it.



Note: here it is the receiver that is not keeping up; the network can keep up fine.

Flow control in quantum network

Make sure sender does not produce **qubits** faster than receiver can consume them.

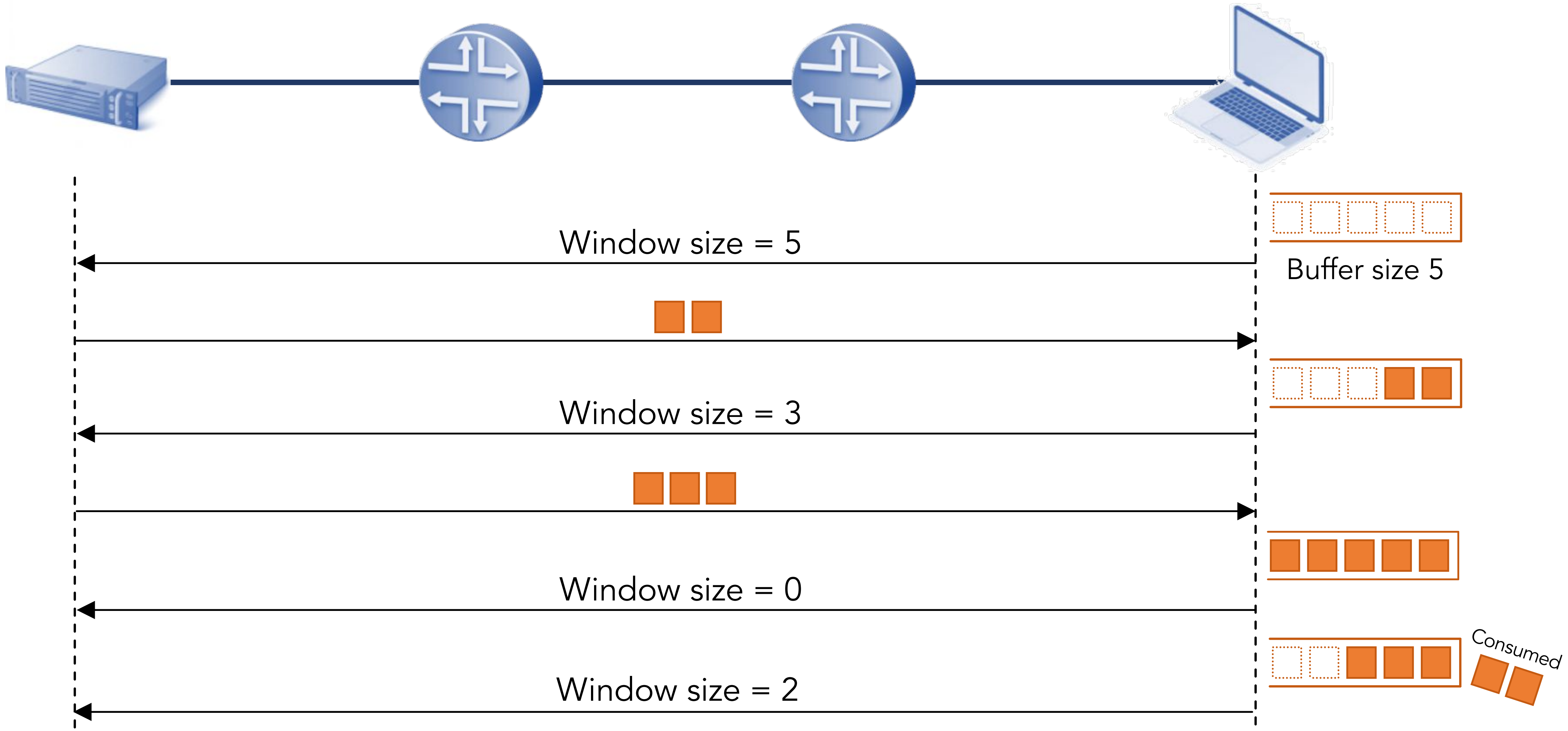


[Animation]

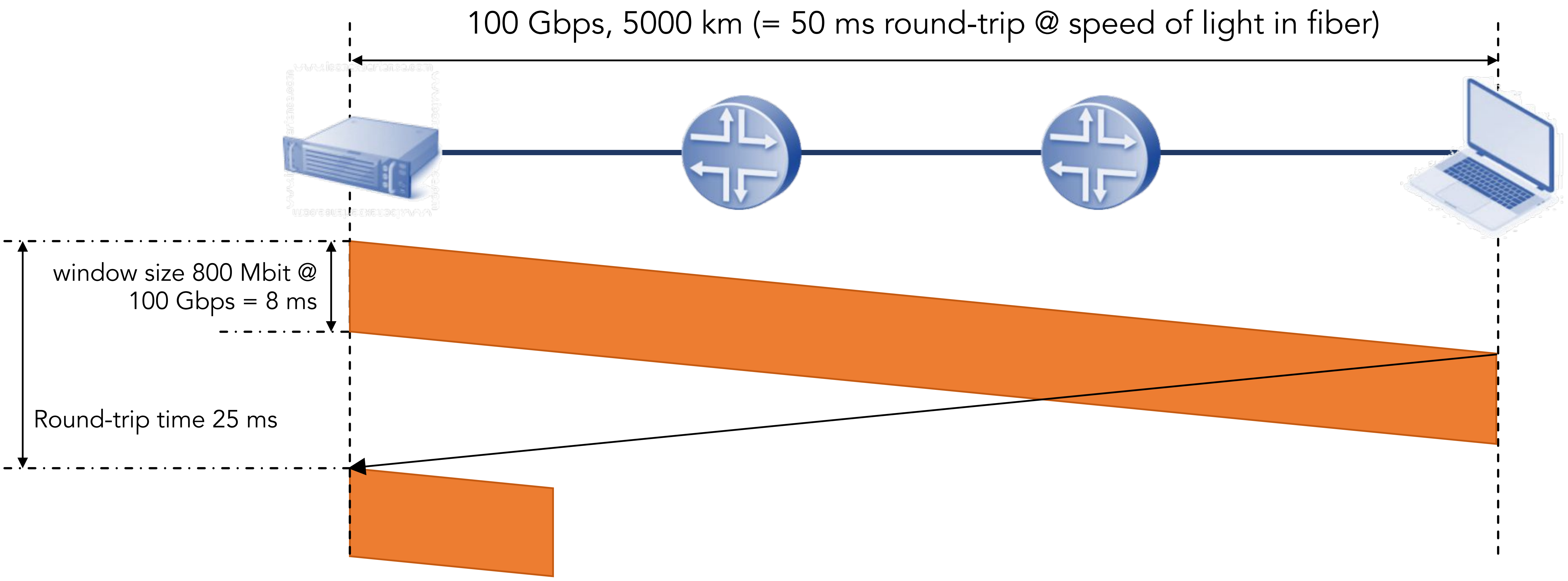
Flow control approaches

- General principle: make the data arrive more slowly
- Approach 1: **Sliding window**
  - Receiver makes the sender slow down by delaying permission to send
  - It is okay to receive data later (delay in-sensitive traffic)
  - Web pages, file transfer, e-mail, …
- Approach 2: **Adaptive encoding**
  - The sender sends less data by switching to more lossy compression
  - It is not okay to receive data later (delay-sensitive traffic)
  - Voice, live video, video on demand, …
- Related approach: **Time slot scheduling**
  - Used in Time Division Multiplexing (TDM) and wireless networks
  - Avoids congestion from happening in the first place
- Sender must be able to back-pressure application

Sliding window



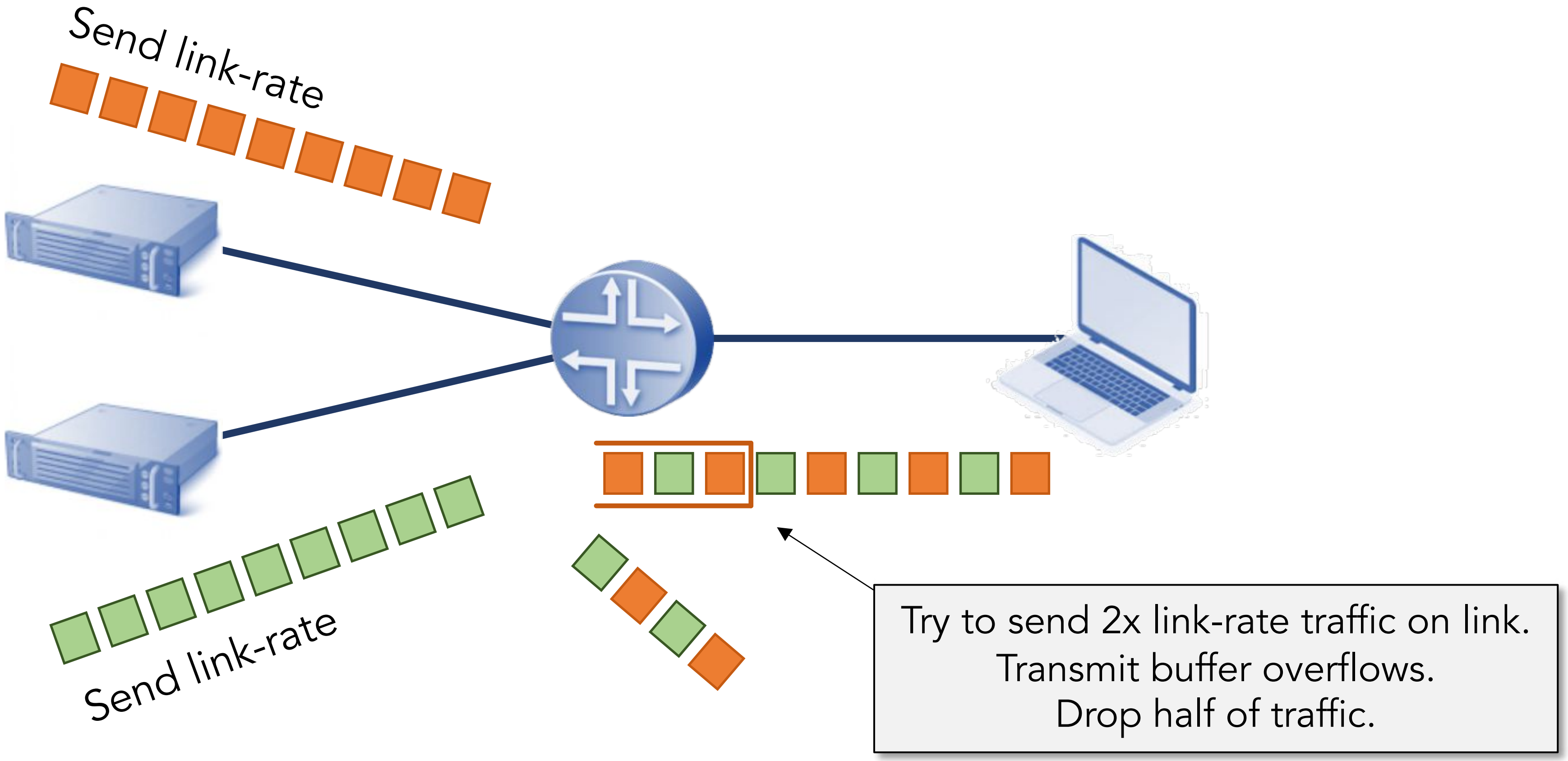
Bandwidth-delay product



Window size must be ≥ bandwidth * round-trip delay to fully utilize available bandwidth.
In this example window size (= receive buffer size) is 100 MB = 800 Mbit
Only 32% of available bandwidth is used.

Congestion

Make sure senders do not produce data faster than network can carry it.



Congestion control

- Congestion **avoidance**: avoid congestion in the first place
- Congestion **mitigation**: deal with congestion when it does occur
- Worst case scenario: congestive **collapse**
  - Effective capacity of the network to carry useful traffic drops to zero
  - Retransmissions due to excessive drops and queueing delays overwhelm network
  - More likely to happen when capacity is low compared to demand (as it will be in early quantum networks!)
  - This actually happened in the early internet: in 1986 the capacity collapsed from 32 Kbps to 40 bps (*not* Kbps)
    https://blog.acolyer.org/2015/05/21/congestion-avoidance-and-control/
    https://tools.ietf.org/html/rfc896

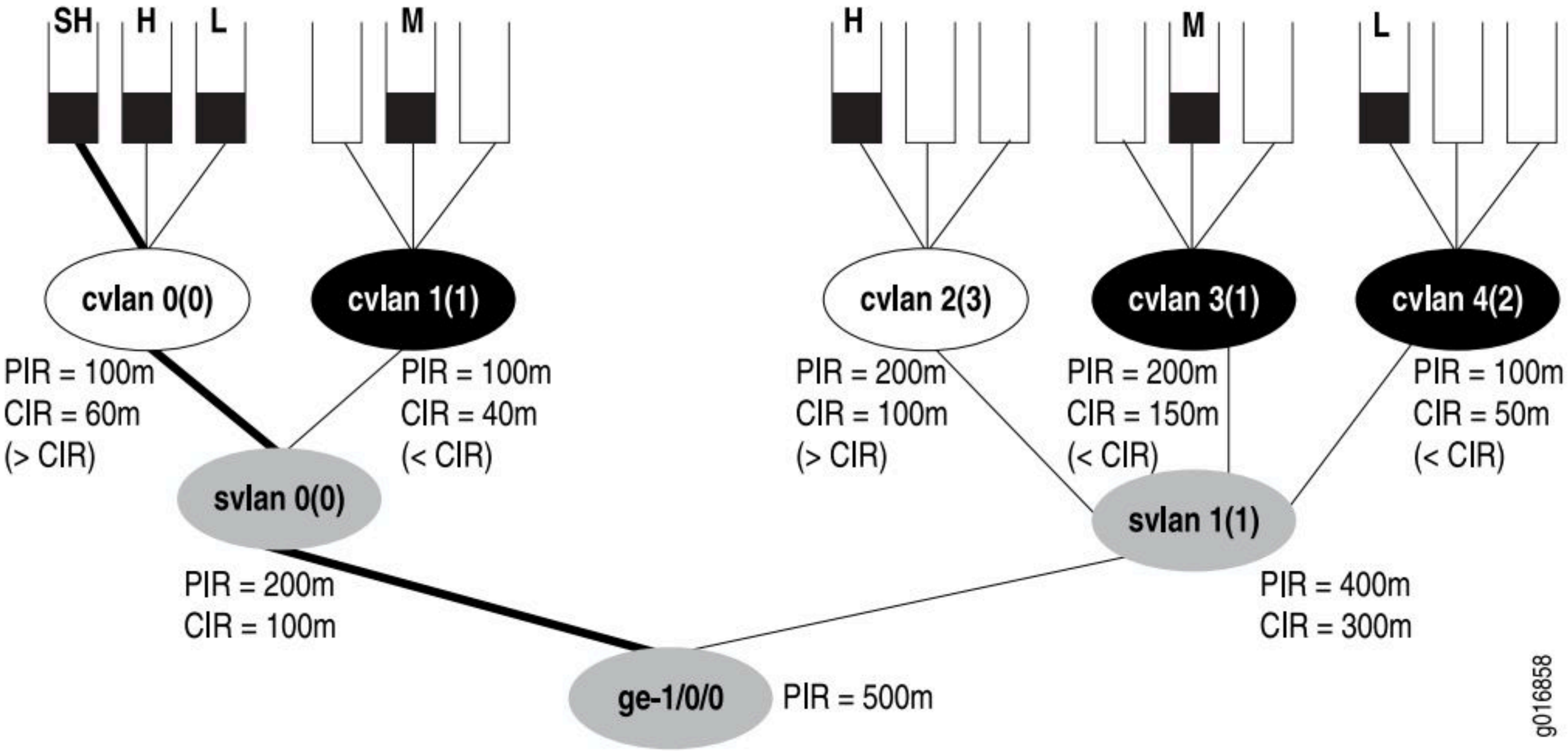
Congestion avoidance

- Congestion avoidance: mechanisms to avoid congestion in the first place
- General principle: send traffic more slowly when congestion occurs
- TCP congestion control
  - TCP tries to estimate maximum rate before congestion occurs
  - TCP sender dynamically congestion window based on presence or absence of drops
  - Extremely difficult to get right: Reno, Vegas, BIC, CUBIC, …. and many more algorithms
  - Assumes everyone plays fair
- Backwards Explicit Congestion Notification (BECN)
  - Router sends message to source to slow down when it observes congestion
- Forward Explicit Congestion Notification (FECN)
  - Set bits in forwarded IP packet: Congestion Encountered (CE)
  - Requires cooperation from the transport layer to slow down

Congestion mitigation

- Active Queue Management (AQM):
  - Mechanism for dealing with congestion when it does occur
  - Queueing only happens if there is contention = congestion
- Decide which packets are most important
  - Classification at the edge
  - Mark classification result into QoS / precedence bits in packet header
- Decide which packet to service next in the send queue
  - Hierarchical schedulers
  - Shapers
- Decide when to start dropping packets and which packet to drop
  - Tail-drop
  - Random Early Drop (RED)
  - Drop out-of-profile packet before in-profile packets

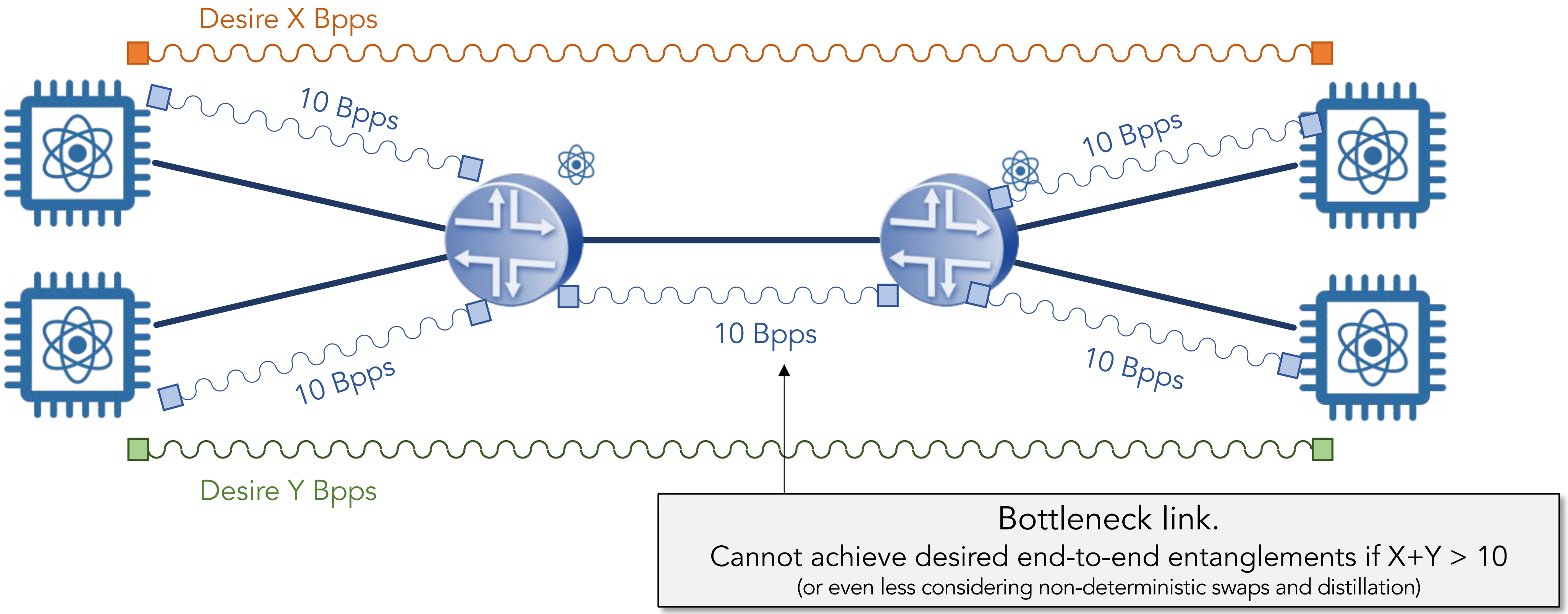
Hierarchical queue scheduler



Source https://www.juniper.net/documentation/en_US/junos/information-products/pathway-pages/cos/config-guide-hierarchical-cos.pdf

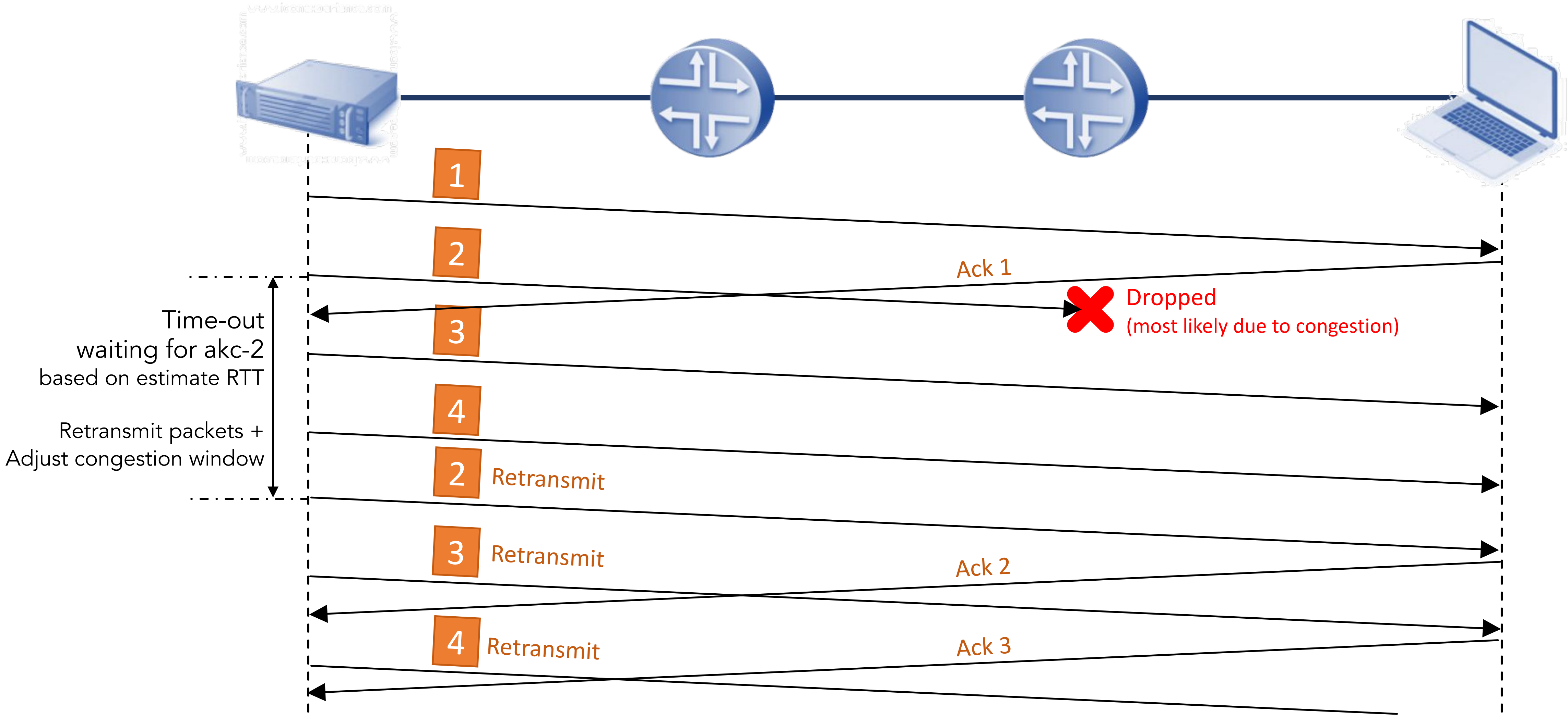
“Congestion control” in quantum networks

End-points cannot get more end-to-end Bell pairs per second (Bpps) than network can produce them based on bottleneck link or router.



Reliable transport: error detection and recovery

This is an example of the go-back-N error recovery



Error detection and recovery mechanisms

- Checksums
  - To detect transmission errors
  - Classical networks are so reliable that the vast majority of drops are due to congestion
- Automatic Repeat Request (ARQ)
  - Receiver sends an acknowledgement (ack) when it has correctly received the data
  - The sender re-transmits the data when it does not receive the ack
  - Not an option for quantum networks (requires that the sender keeps a copy until the data is acknowledged, in case the data needs to be retransmitted).
  - Go-back-N: resend all data starting at the dropped packet
  - Selective repeat: only resend the data that was dropped (requires selective ack)
- Forward Error Correction (FEC)
  - When error rate is too high (noisy links)
  - When retransmission is delay is not acceptable (e.g. long latency satellite links)
  - The only option in quantum networks (quantum error correcting codes)

What is so difficult about this?

- Flow control, congestion control, and error control are separate but closely related concepts
- Most transport protocols don't have clean separation of concerns
- For example, in TCP absence of ACK can mean many things:
  - Flow control: receiver wants us to slow down
  - Error: data packet or ACK was dropped
  - Congestion: data packet or ACK was delayed in queue
- Over time TCP has been enhanced:
  - Selective acknowledgement (SACK)
  - Timestamps

Flow control
Congestion control
Error control
in the **control** plane

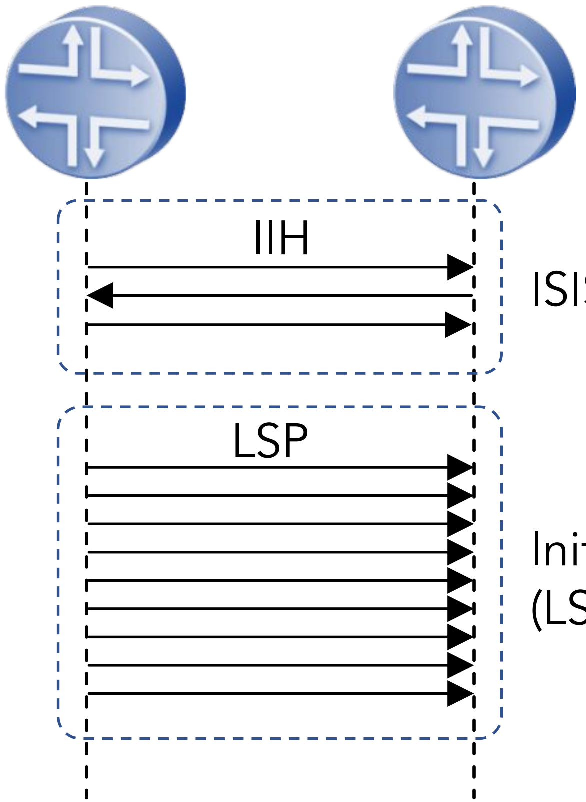
Flow control matters in the control plane

- Routing and signaling protocols exchange large volumes of data
- Sender must not overwhelm receiver
- Flow control approaches in routing and signaling protocols:
  - Fixed pacing
  - Flow control mechanism built into the protocol itself
  - Rely on TCP flow control

Example: ISIS flow control

Initial database synchronization sends large volume of link state packets (LSPs)

Potentially tens of thousands of LSPs



ISIS adjacency comes up

Initial link-state database (LSDB) synchronization

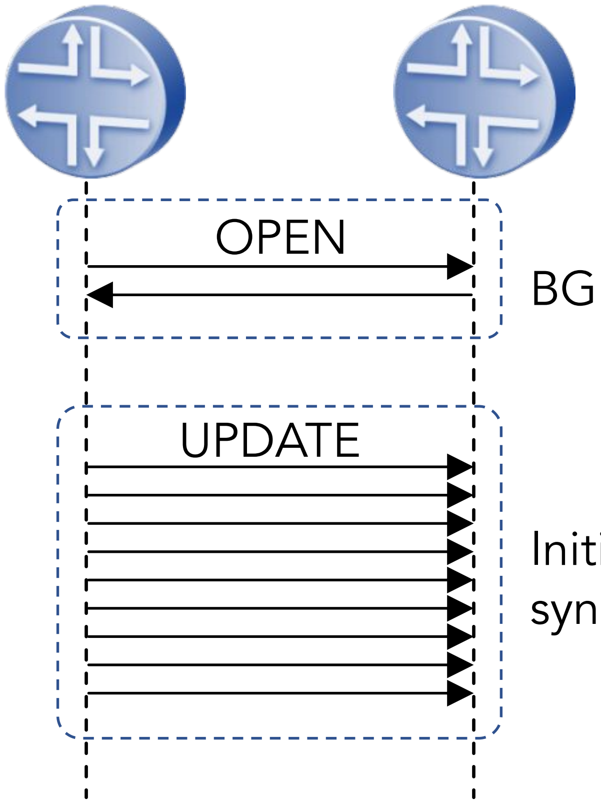
Pacing parameters in standard, e.g. minimumLSPTransmissionInterval

Recent proposals to dynamically control intervals in ISIS protocol.

Example: BGP flow control

Initial database synchronization sends large volume of routes (UPDATEs)

Often well above 1 million routes



BGP session comes up

Initial route table (RIB) database synchronization

BGP relies on TCP for flow-control (and reliability for that matter)

Congestion control matters in the control plane

- Control packets have strict priority over user packets
- Large volume of control packets can still cause congestion
- Congestive collapse of control protocols is not rare
  - Flapping ISIS adjacencies self-reinforcing feedback loop
- Solutions:
  - Prioritize important control plane packets (mainly hellos)
  - Multiple streams (QUIC, SCTP)
  - Off-load liveness detection to separate protocol (BFD)

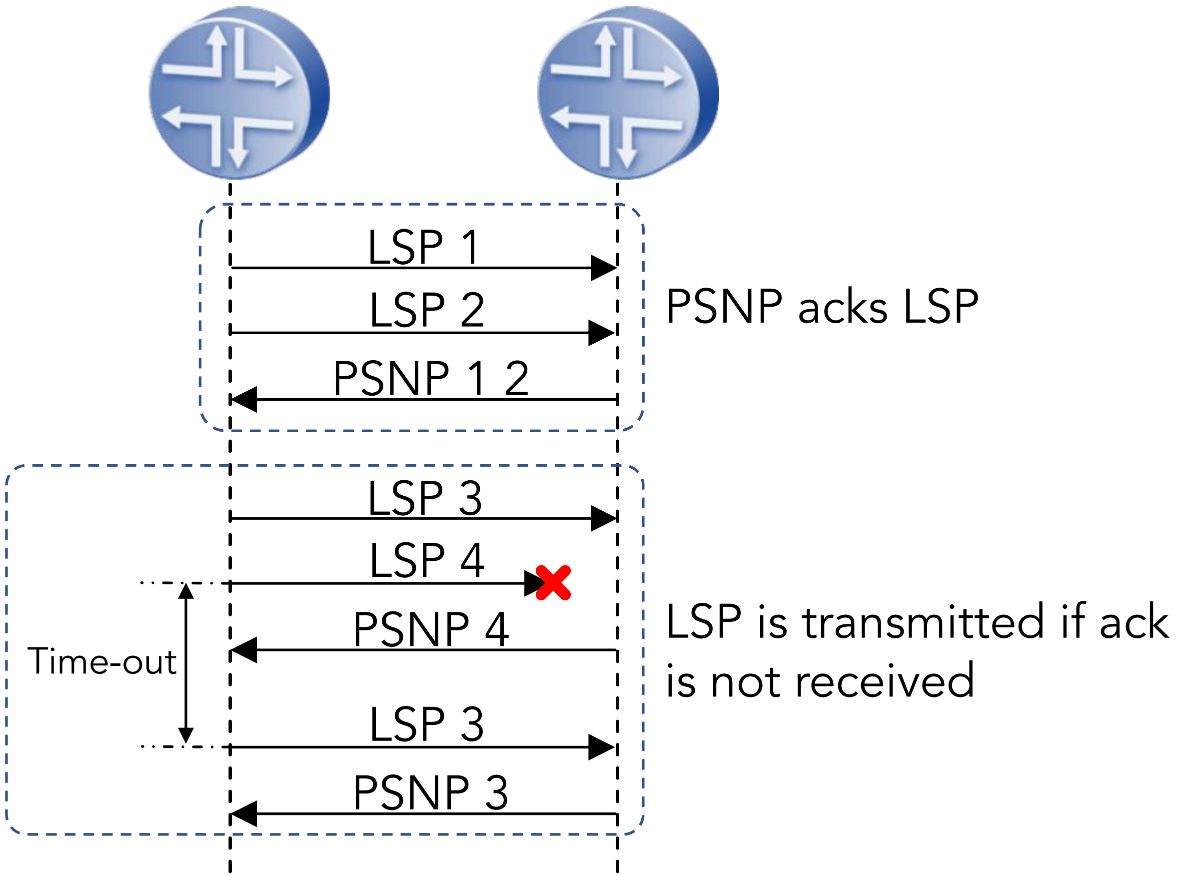
Error control matters in the control plane

- Control plane packets also get dropped
- Need reliability mechanism (error control) in routing and signaling protocols
- Approaches:
  - Separate mechanism built-in to protocol itself: ISIS, OSPF, …
  - Rely on transport (TCP) for reliability: BGP, LDP, …

Example: ISIS error control

Partial Sequence Number PDU (PSNP) is used to ack LSP

There is also a Complete Sequence Number PDU (CSNP) that we don't discuss here



Take-aways for quantum control protocols

- You need to worry about flow control
- You need to worry about congestion control
- You need to worry about error control
- Not just in the data plane, but also in the control plane
- Avoid re-inventing the wheel: rely on layer 4 transport when possible

  This is somewhat contentious; it is my opinion that not everyone agrees with.

Soft state versus hard state

Soft state vs hard state

Sender wants to advertise some data (e.g. a route) to receiver.

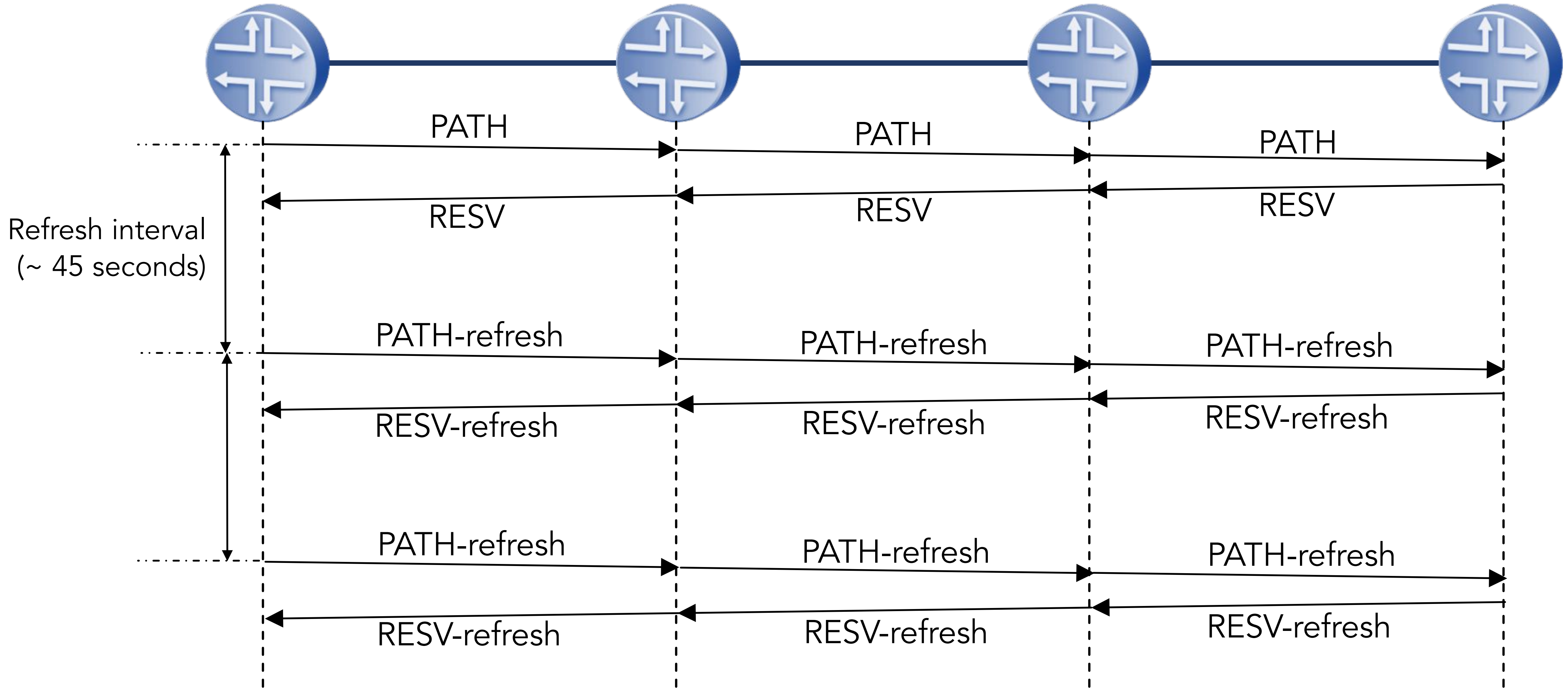
Soft state protocol

- A.k.a. Periodic protocol
- Sender initially sends data.
- Receiver puts data in its database.
- Sender periodically resends data (refresh).
- If receiver doesn't receive data anymore, it removes data from its database after time-out.
- Heavy load due to periodic transmission.
- Fallacy: simpler cleanup when sender crashes.

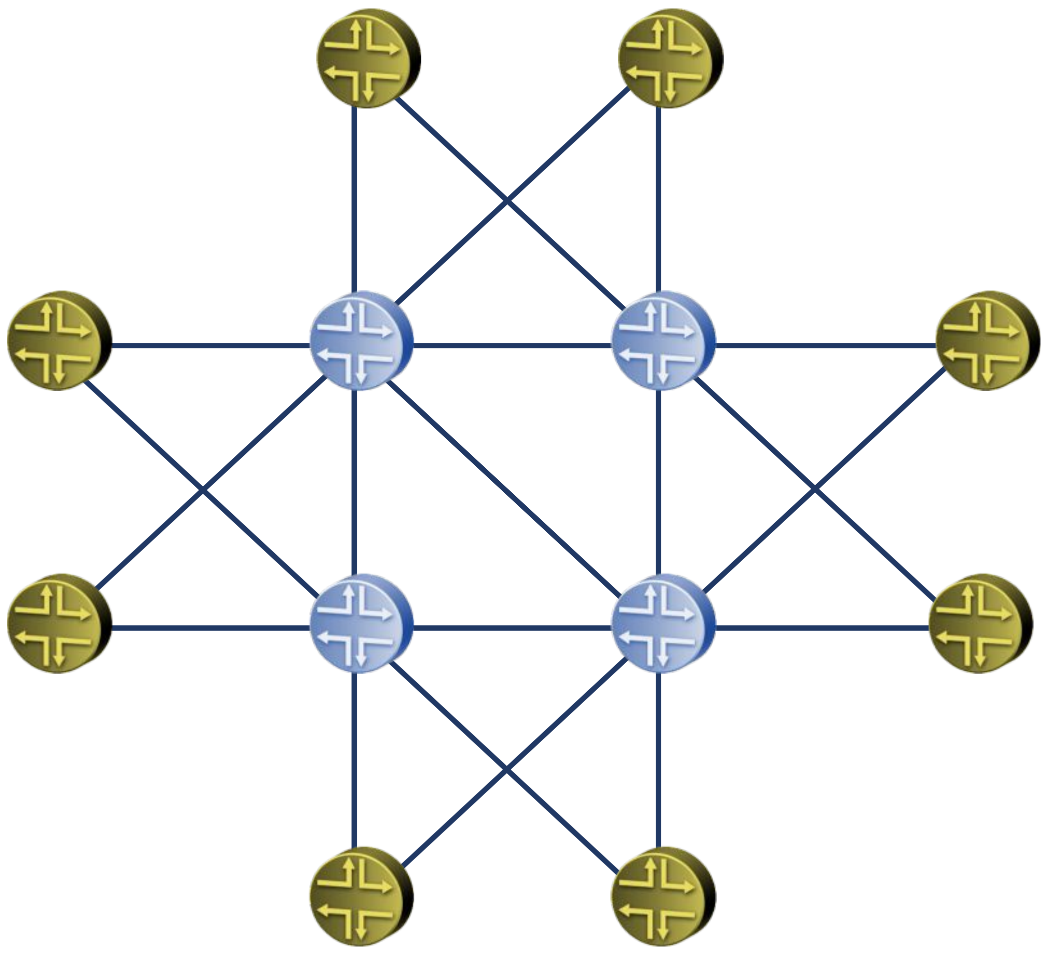
Hard state state protocol

- Runs protocol over reliable transport connection.
- Sender initially sends data.
- Receiver puts data in its database.
- No periodic re-transmissions of data.
- Receiver keep data in database until (a) sender explicitly withdraws data or (b) transport connection is disconnected.
- No traffic if no changes in the database.
- Needs reliable transport connection and keep-alive mechanism (typically TCP)

Example: soft state in RSVP



The problem with soft state



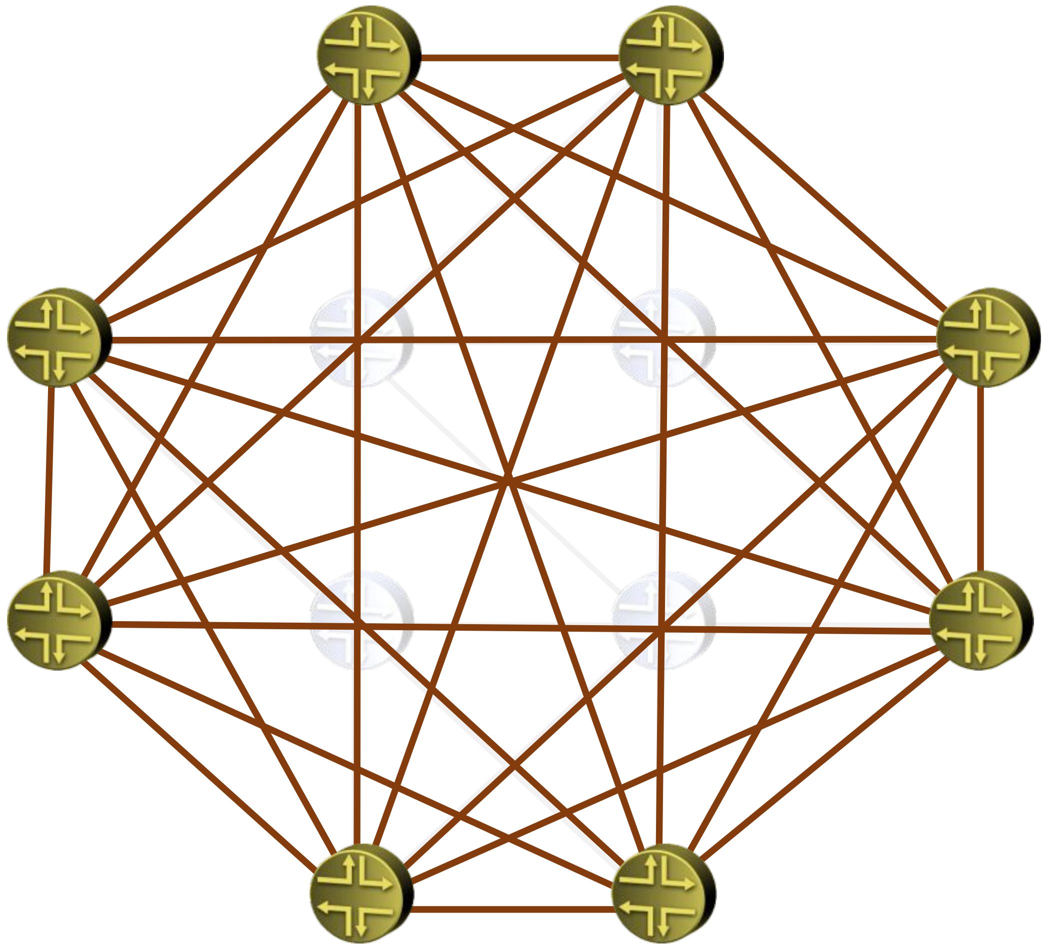
Typical network topology

Core router

Edge router

Dual-homed to core router

The problem with soft state



Typical network topology

Core router

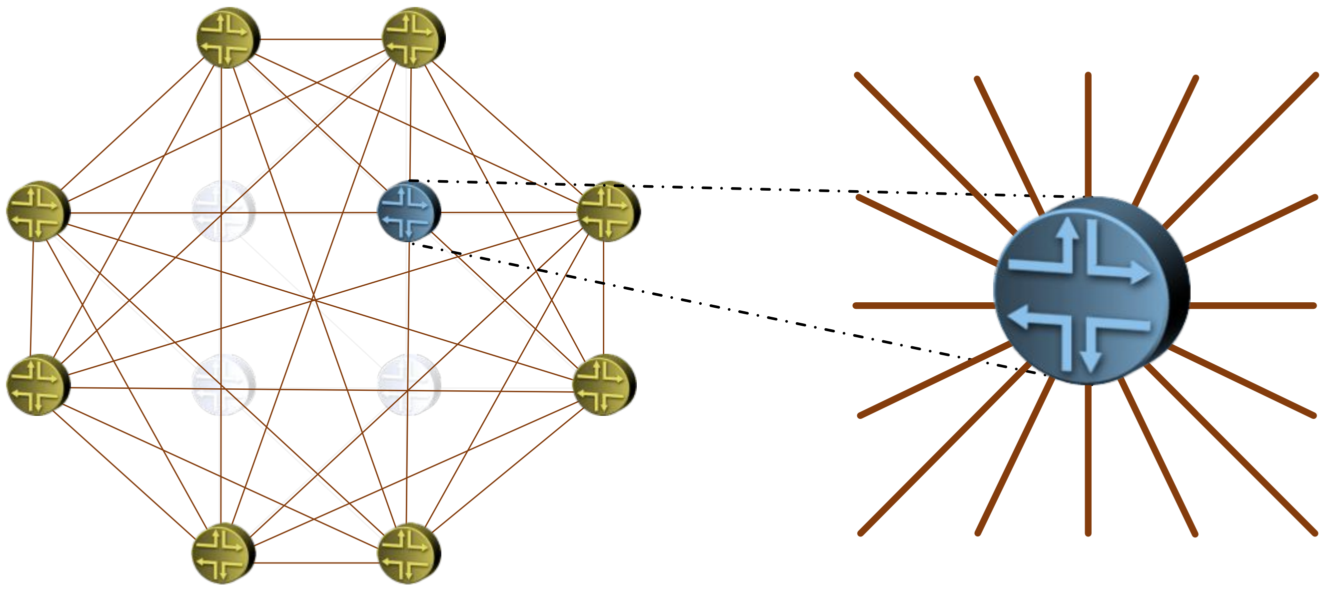
Edge router

Dual-homed to core router

Full mesh of LSPs

From every edge router to every other edge router
Traffic engineered according to bandwidth-demand matrix
This diagram show logical topology of LSPs (not physical topology)

The problem with soft state



Core router

Number of LSPs is $O(N^2)$

When N is number of edge routers

Refresh frequency is $O(N^2)$

Real-life example

Number of edge routers: 500

Number of LSPs: 500*499 = 249,500

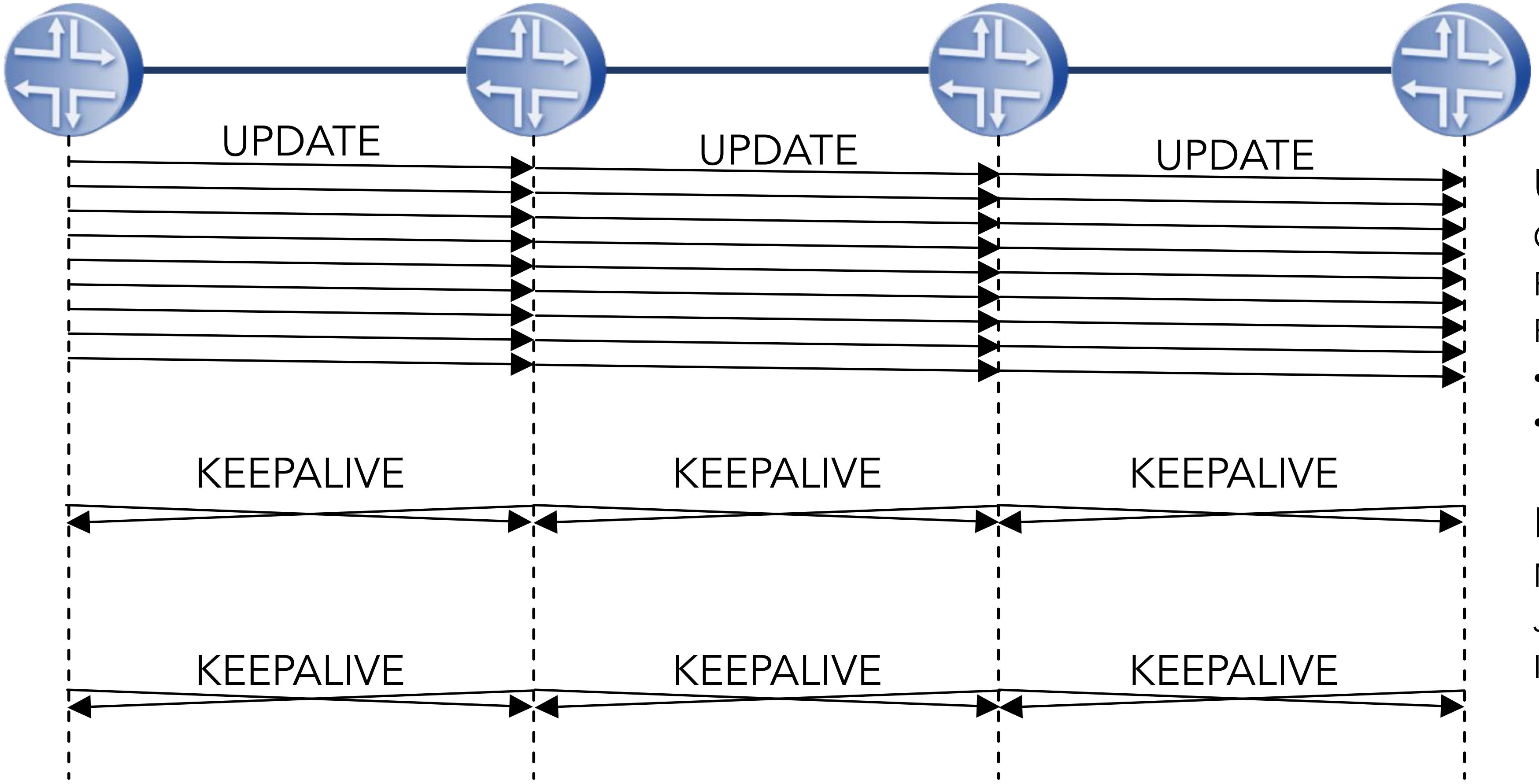
Refresh interval = 45 seconds

Refresh traffic: 11,088 messages/sec

(Simplifying assumption that all LSPs pass through the same core router)

This is not feasible in the control plane

Example: hard state in BGP



UPDATE creates **hard state**

Can be > 1 million routes

Periodic refresh not feasible

Route stays valid until:

- Explicitly WITHDRAW or
- TCP connection breaks

Periodic KEEPALIVE

Needed to check TCP connection

Just a single message

Independent of number of routes

Network meltdown due to

Run-away replication

Packet replication loops

Packet replication

Making multiple copies of a packet when you forward it (e.g. multicast)

+

Forwarding loop

=

Network meltdown

Packet replication loops

Packet replication
Making multiple copies of a packet when you forward it (e.g. multicast)

+

Forwarding loop

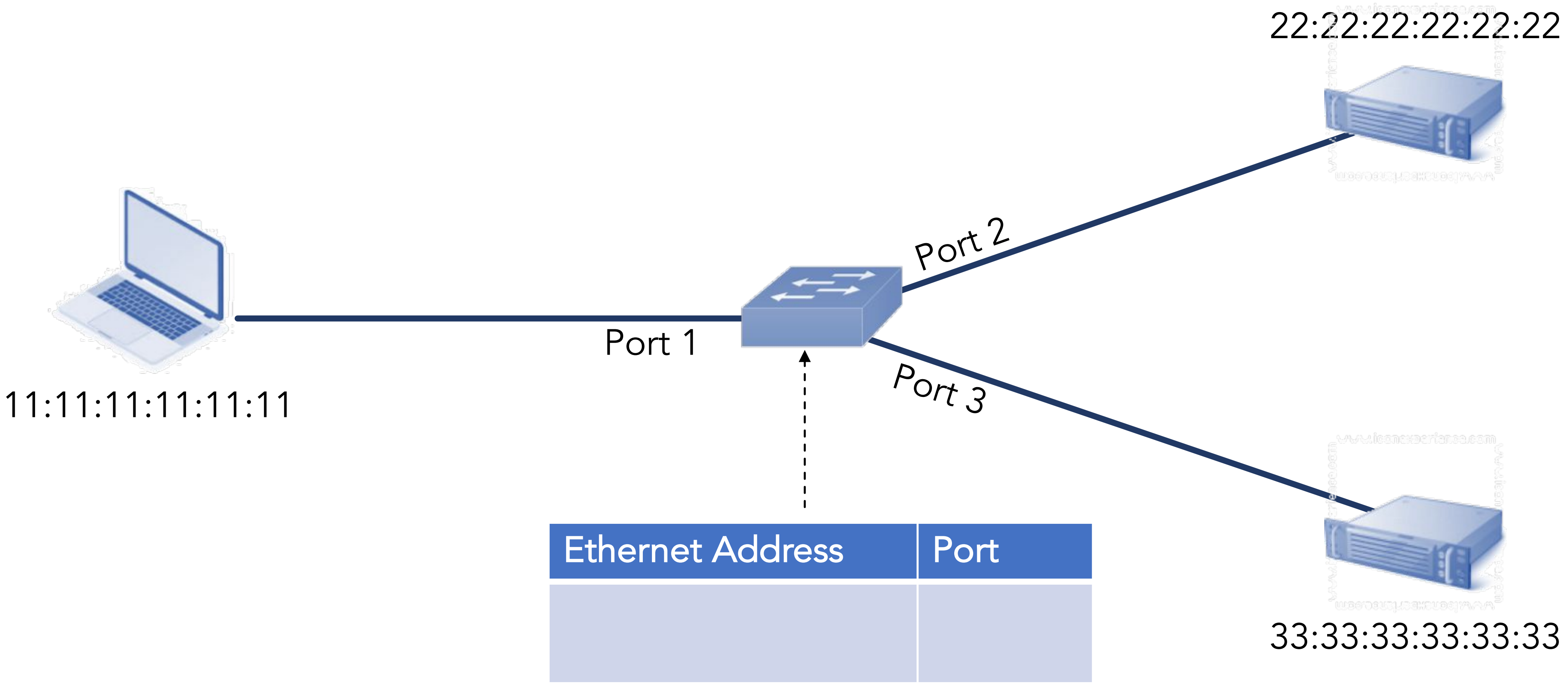
+

No time to live (TTL) mechanism
Packets can loop forever without ever being removed due to TTL expiry

=

Catastrophic network meltdown

Ethernet address learning

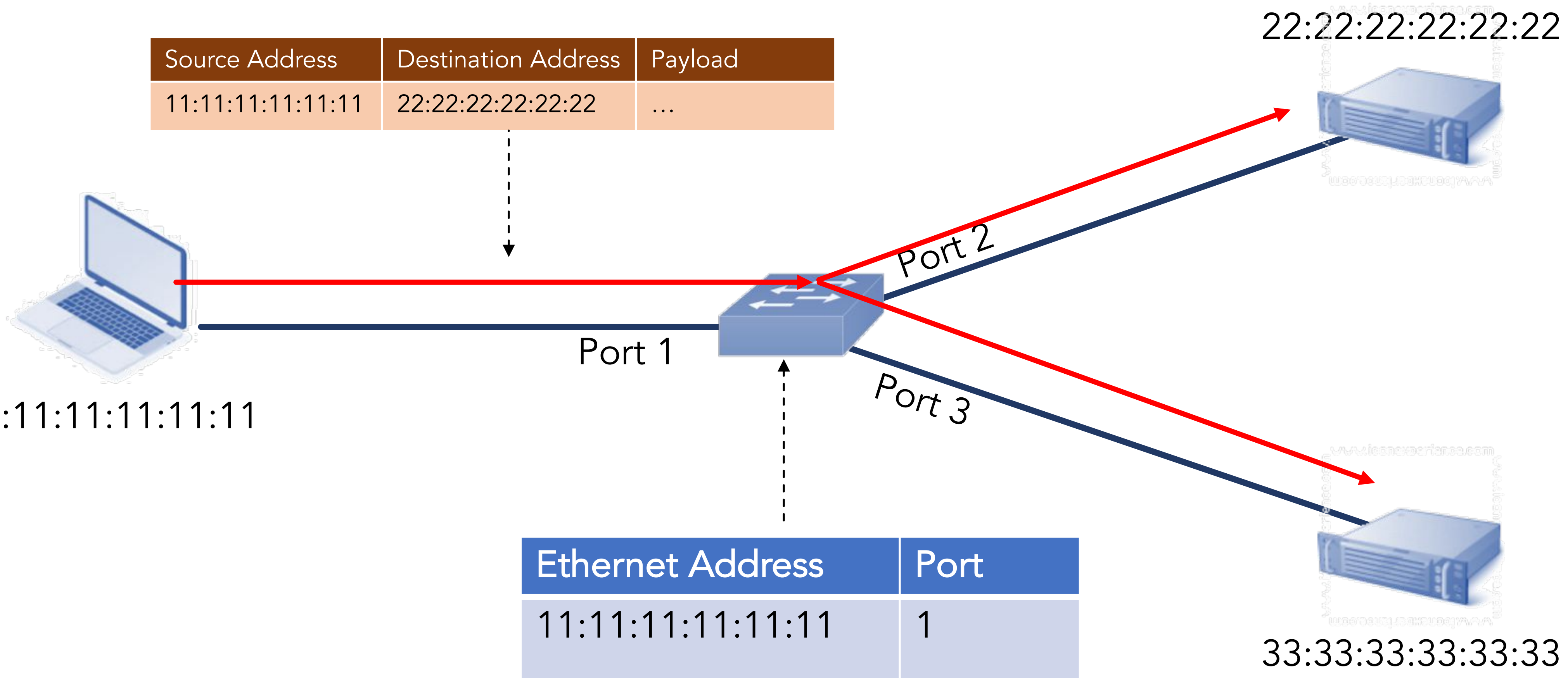


Switch layer 2 forwarding table

Ethernet address learning

Unknown destination address: flood and learn source

Source Address	Destination Address	Payload
11:11:11:11:11:11	22:22:22:22:22:22	…



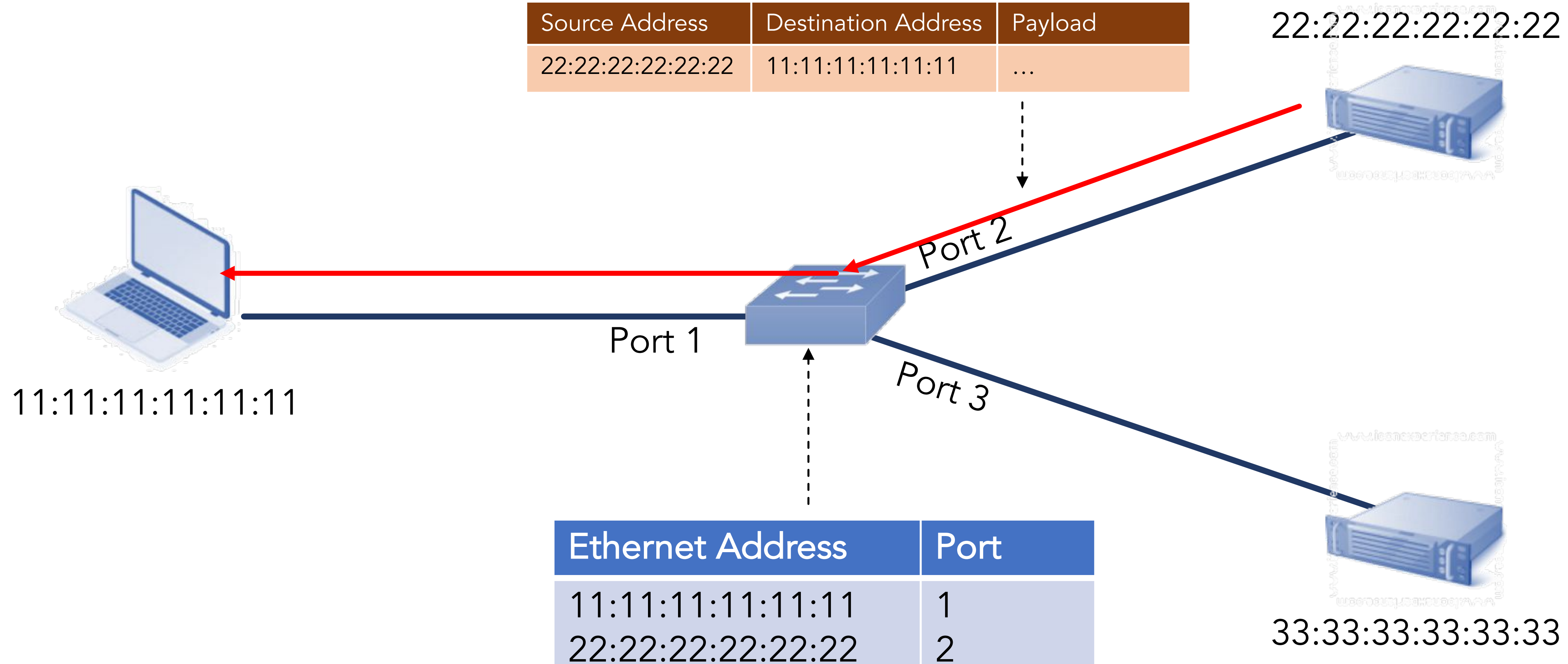
Ethernet Address	Port
11:11:11:11:11:11	1

Switch layer 2 forwarding table

Ethernet address learning

Known destination address: unicast and learn source

Source Address	Destination Address	Payload
22:22:22:22:22:22	11:11:11:11:11:11	…



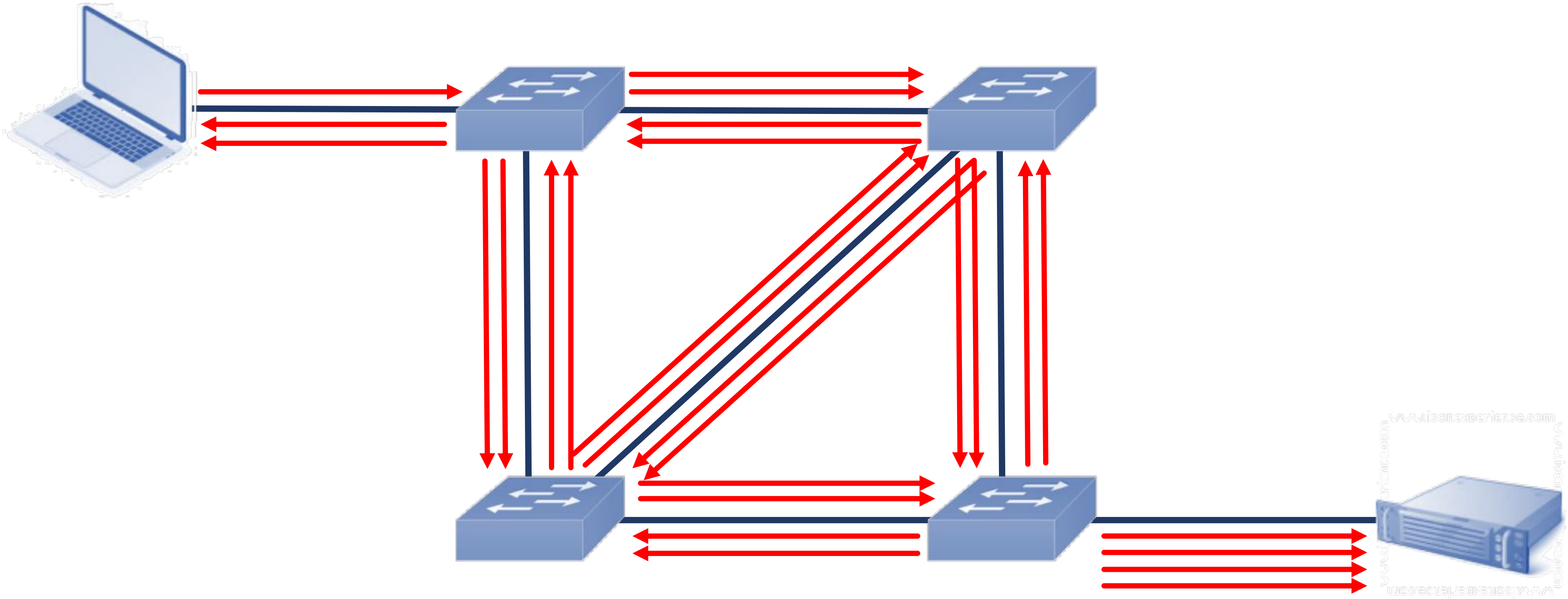
Ethernet Address	Port
11:11:11:11:11:11	1
22:22:22:22:22:22	2

Switch layer 2 forwarding table

Ethernet meltdown due to forwarding loop

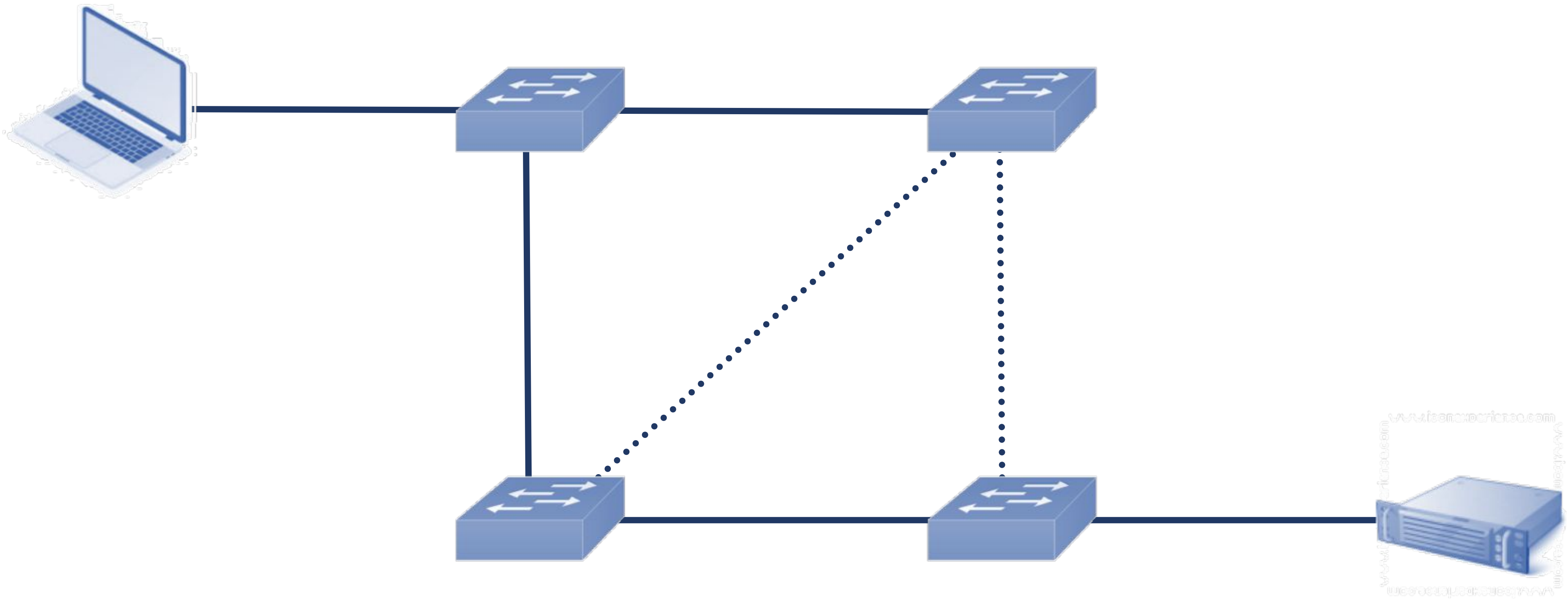
Packet loop around the network at line rate

Number of packets grows exponentially



Spanning Tree Protocol (STP)

STP removes loops by blocking links to form a spanning tree.



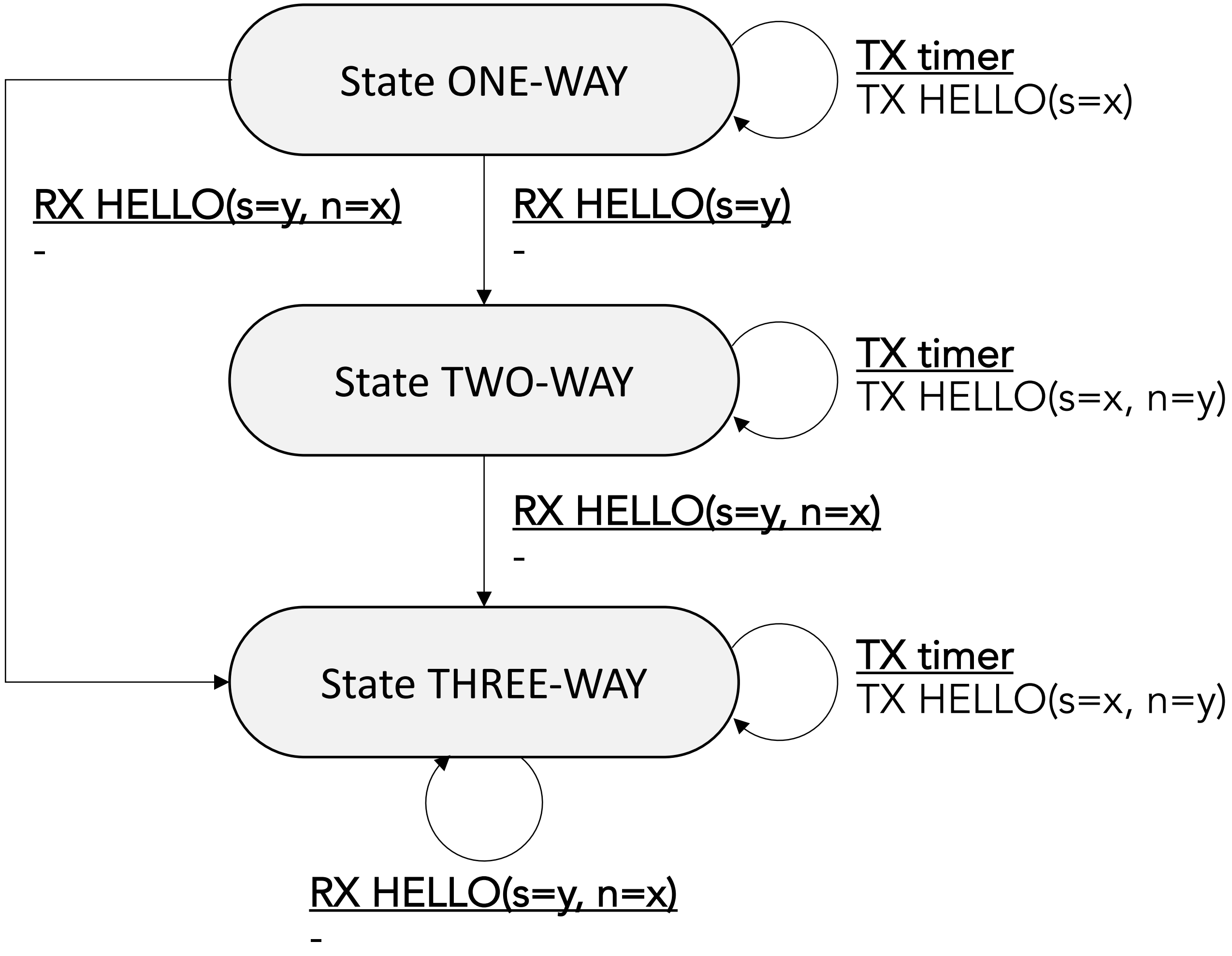
Wastes a lot of link capacity (no multipath)

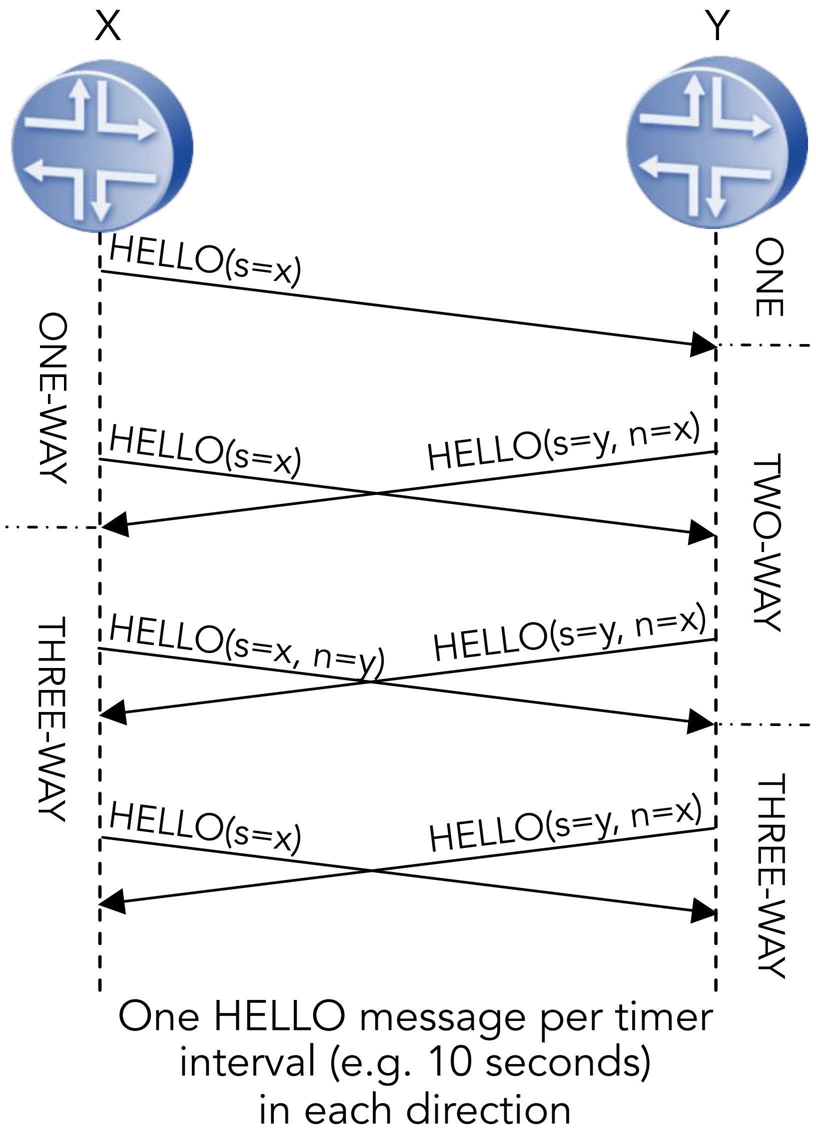
Network melts down if STP doesn't converge perfectly.

Network meltdown due to

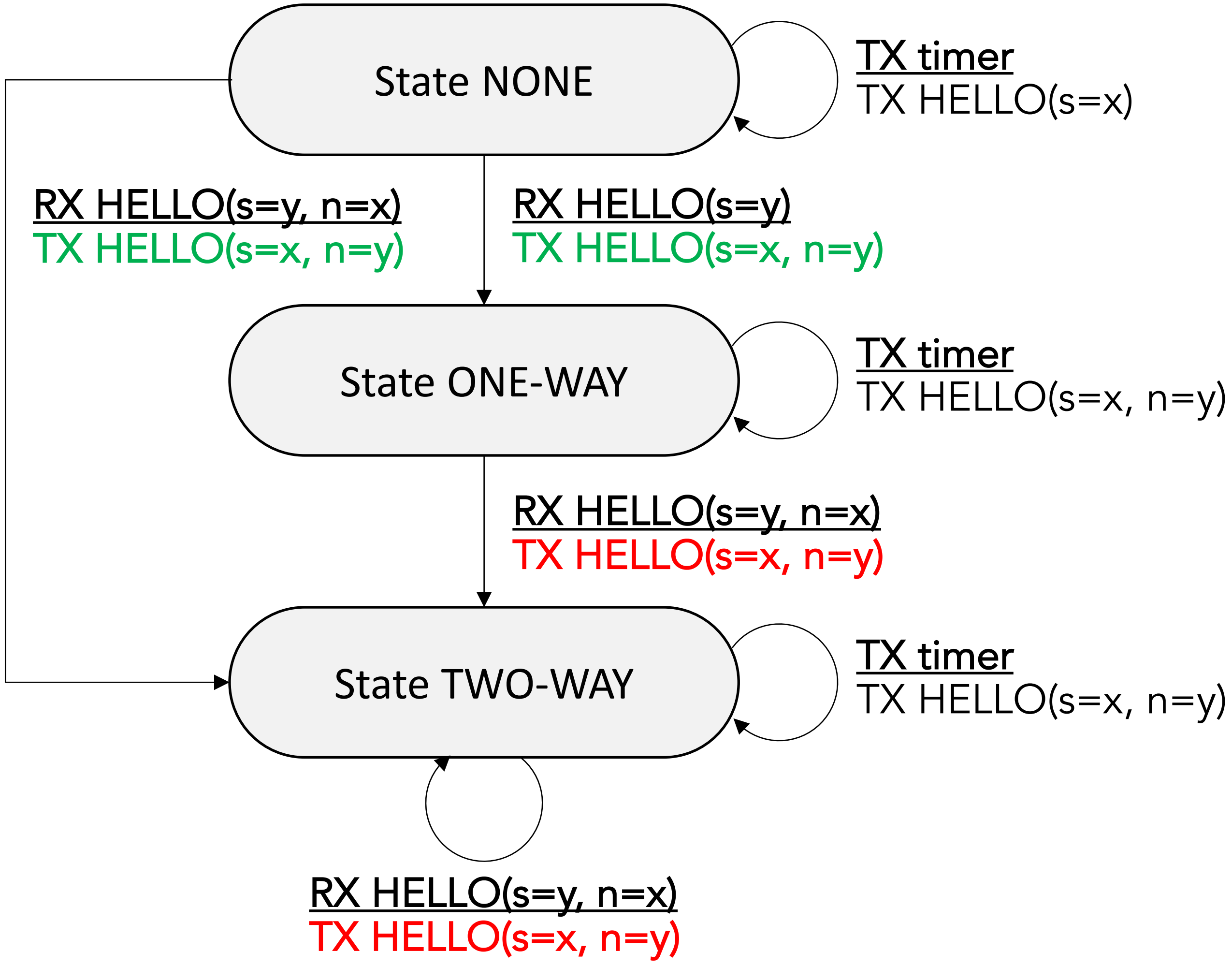
Run-away state machine

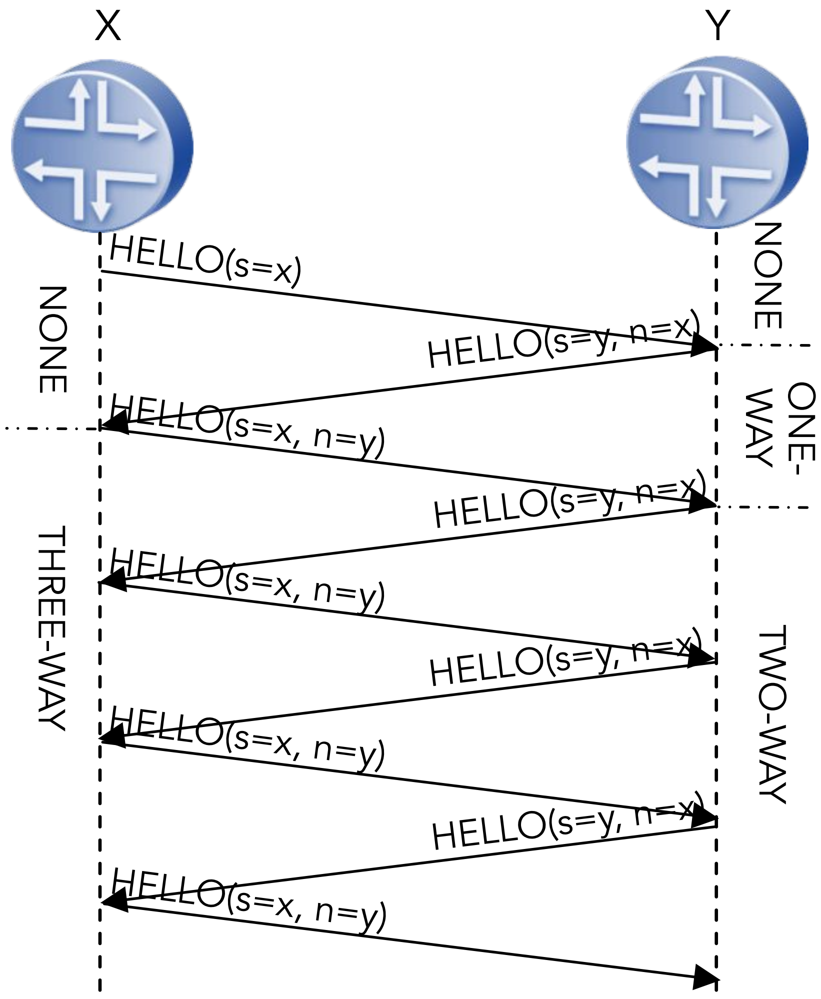
Typical 3-way adjacency finite state machine





Run-away 3-way adjacency FSM





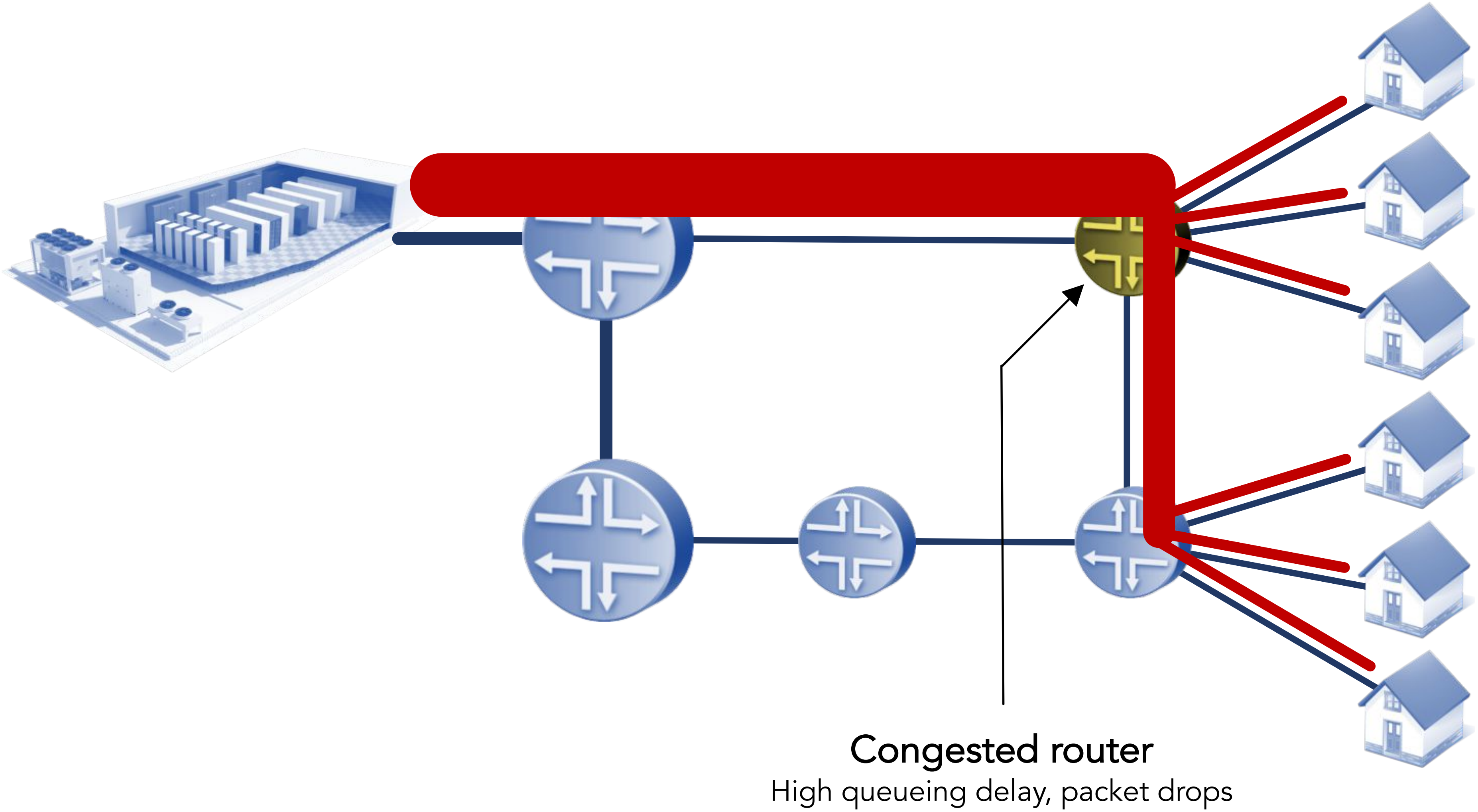
Non-stop line-rate of HELLOs going back and forth

Real life example: early draft of RIFT

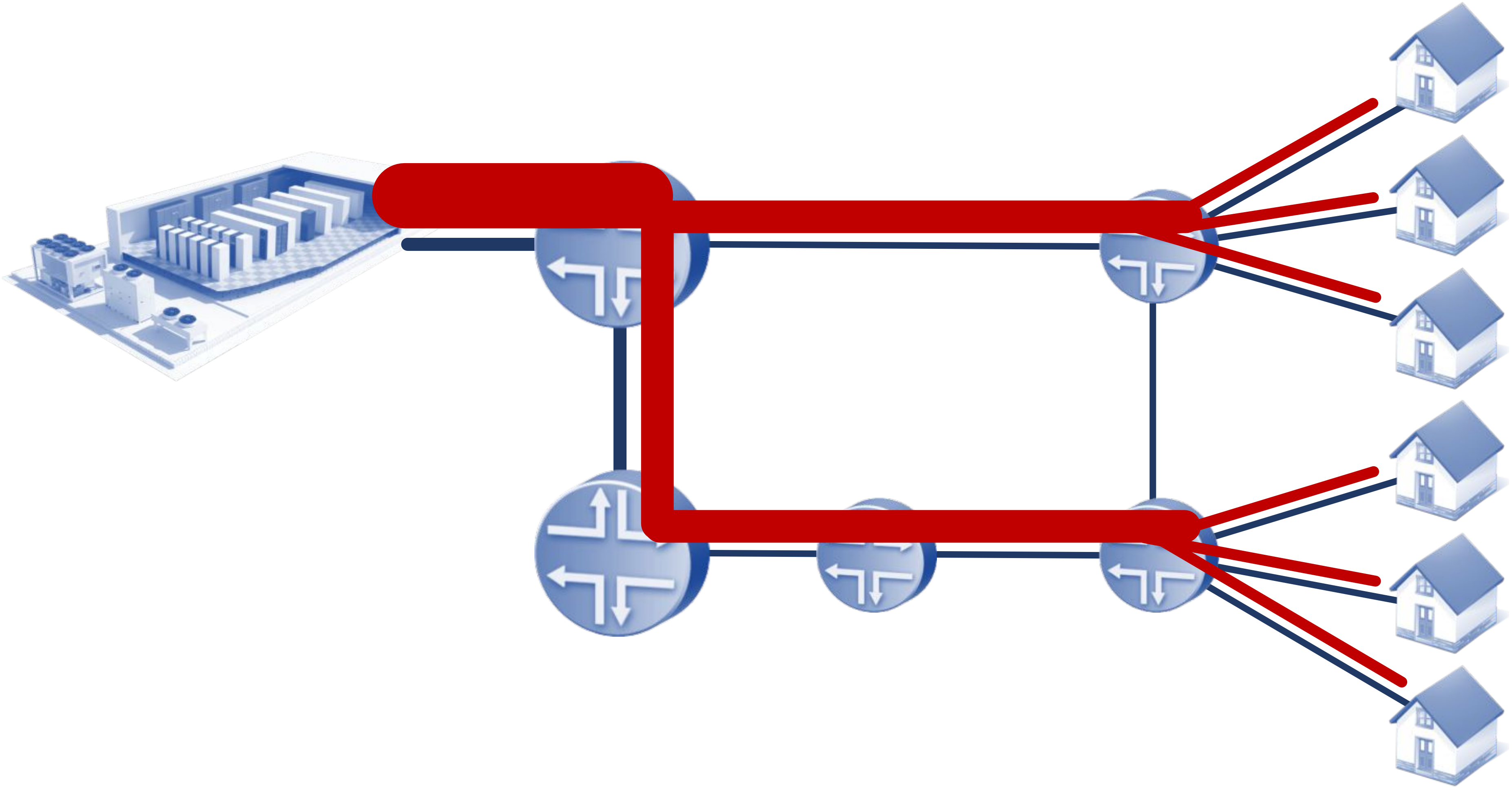
Network oscillation due to

Run-away feedback loop

Shortest path routing, bandwidth metrics

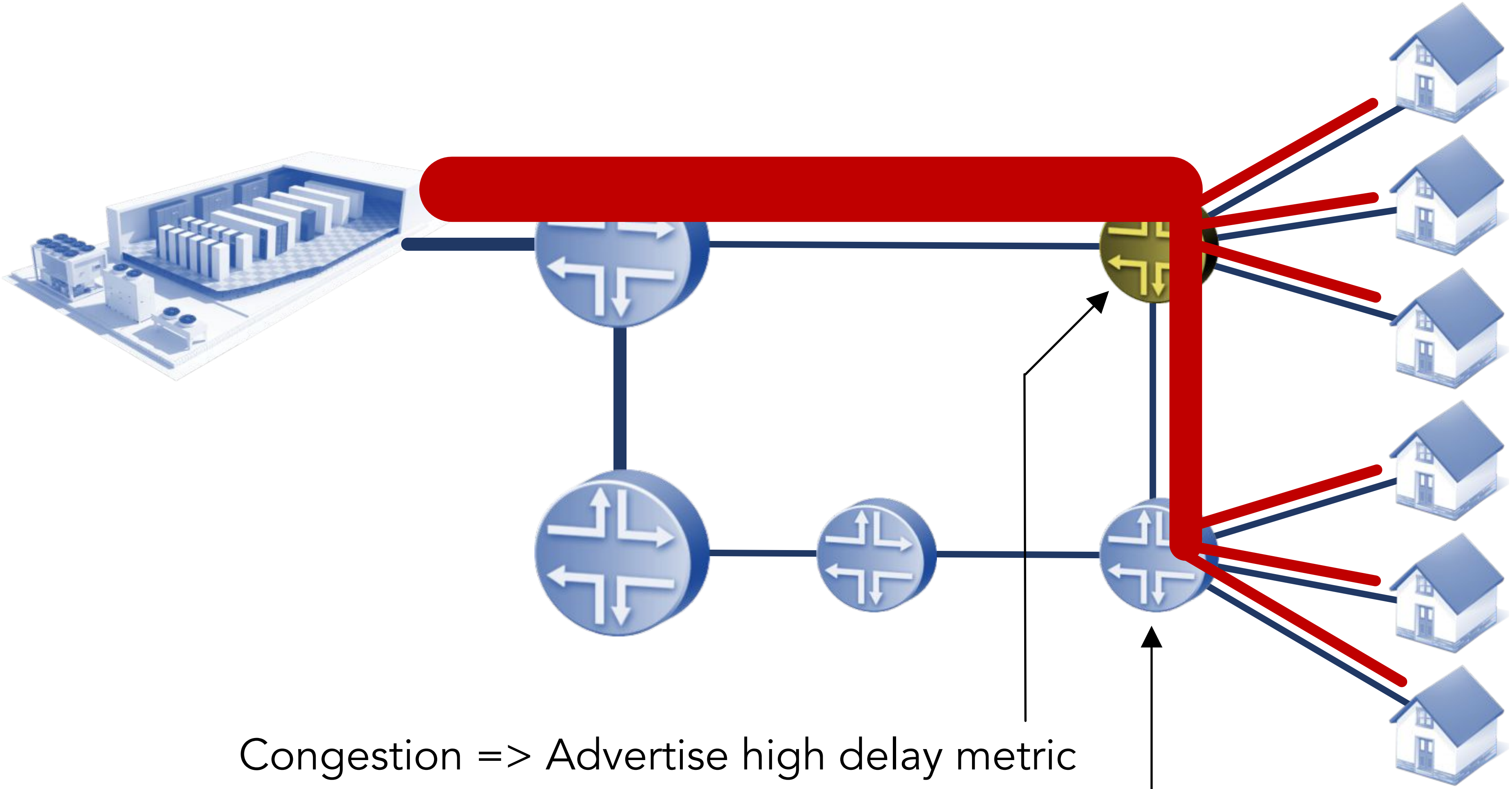


Desired situation: spread the load



This is what traffic engineering would do.

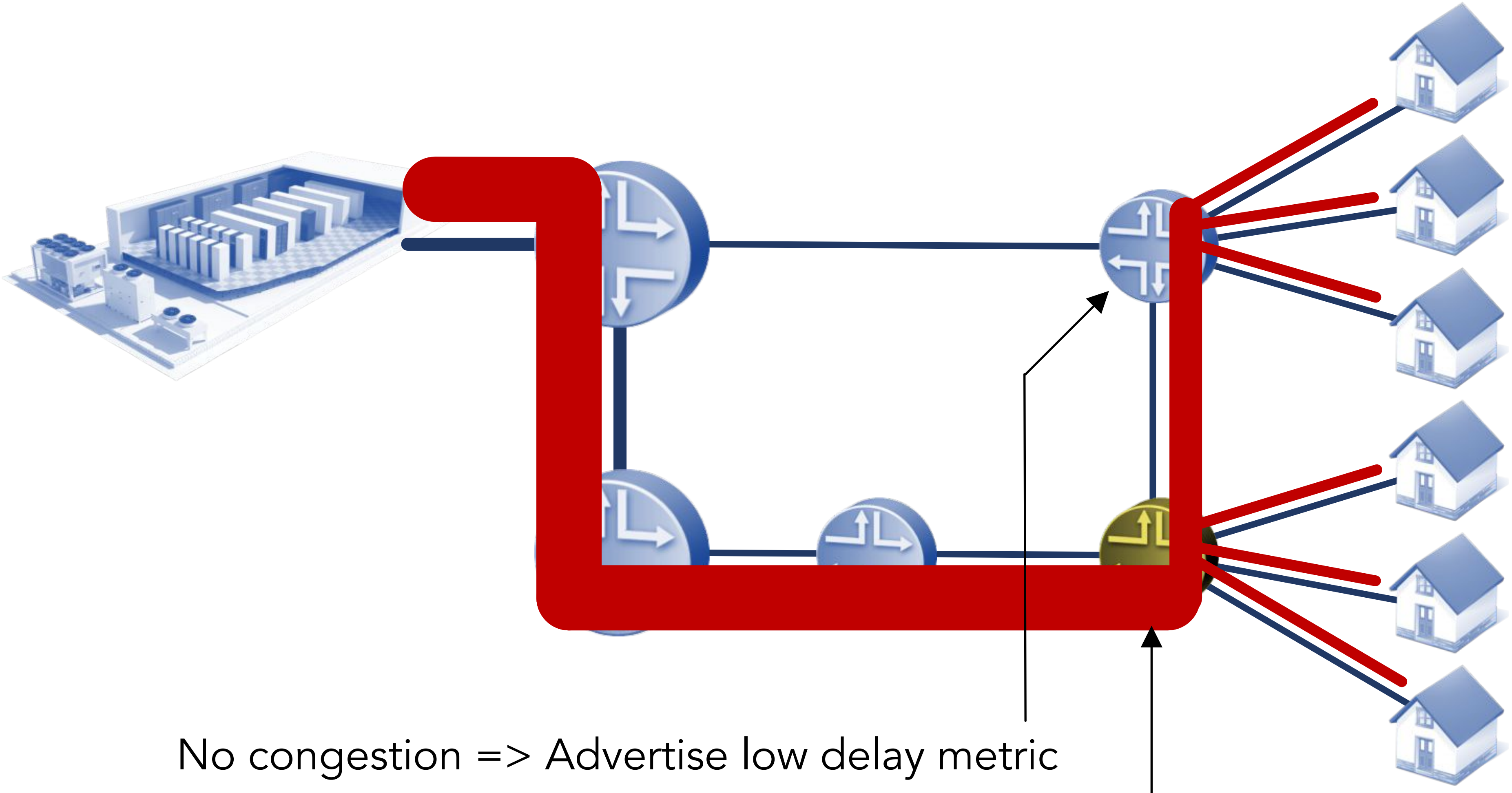
Failed attempts at congestion-sensitive routing



Congestion => Advertise high delay metric

No congestion => Advertise low delay metric

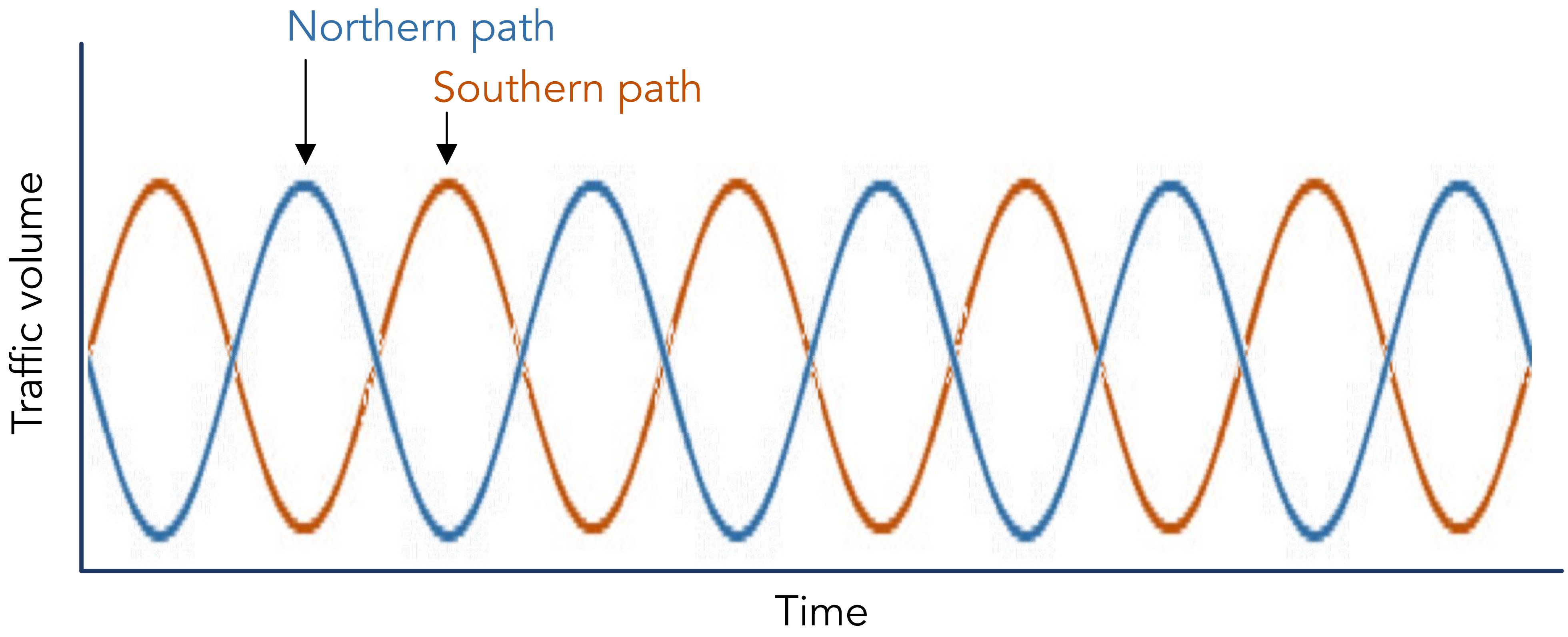
Failed attempts at congestion-sensitive routing



No congestion => Advertise low delay metric

Congestion => Advertise high delay metric

Oscillation due to unstable feedback loop



Delay-based metrics have been attempted many times, and have failed as many times

Take-aways for quantum control protocols

- Make sure your protocols don’t have run-away scenarios
- Beware of run-away replication
- Beware of run-away state machines
- Beware of run-away feedback loops

Thank you.