# routing with NLNETLABS

SURF Internet Measurements and Analysis Workshop 5 April 2023

Luuk Hendriks & Jasper den Hertog

# WHAT DO WE DO?

# CORE INTERNET

Development

Open Source Software

Standards (IETF)

Research

Governance related advisory

# OPEN SOURCE SOFTWARE

soon-to-be super fast authoritative DNS resolver

NSD

RPKI validator ("Relying Party Software")

ROUTINATOR

unbound

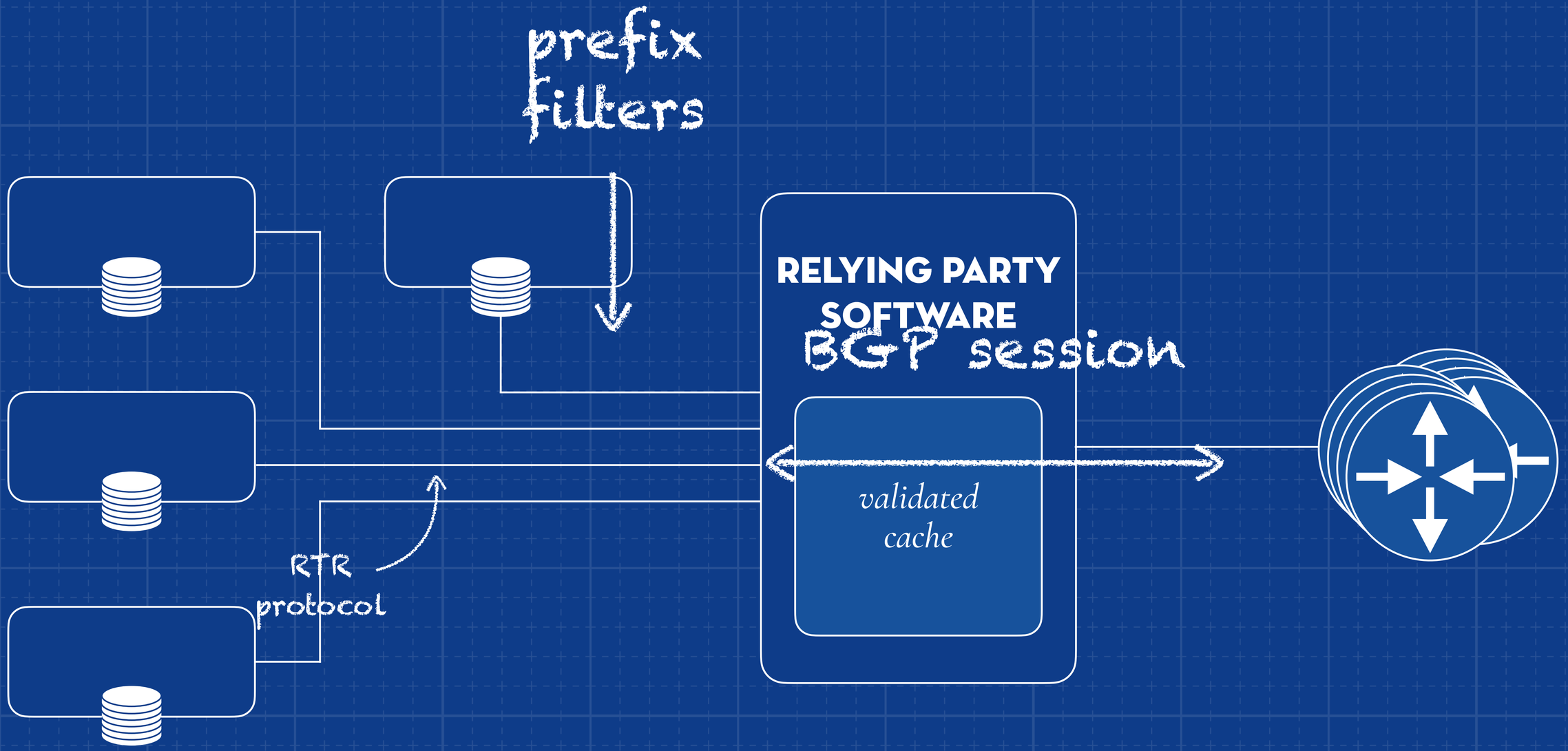
multi-purpose DNS resolver

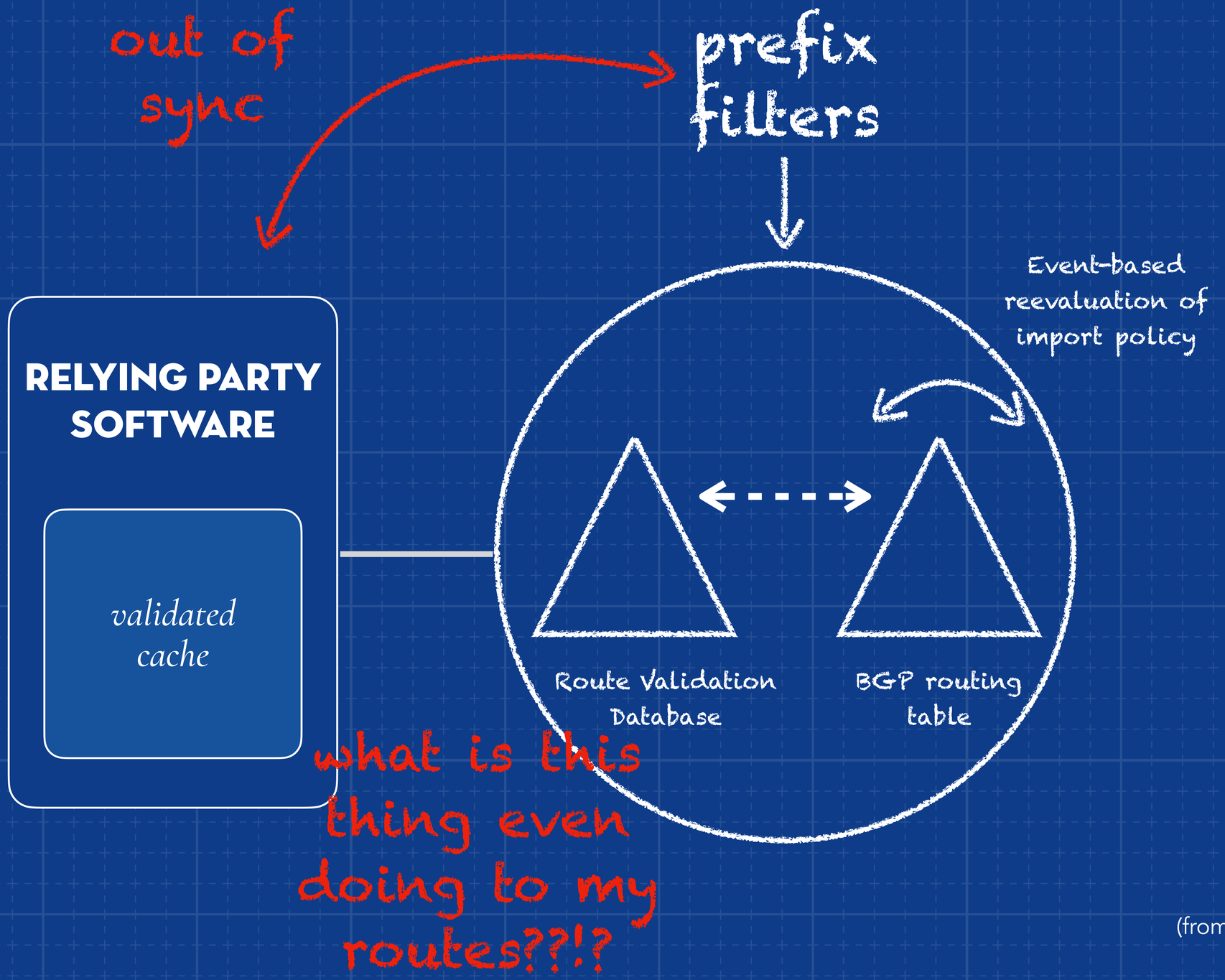
Krill

all singing and dancing delegated RPKI

*memory-safety*
*strongly, statically typed*
*raw performance (no garbage collection)*
*composability (dependencies, modules)*

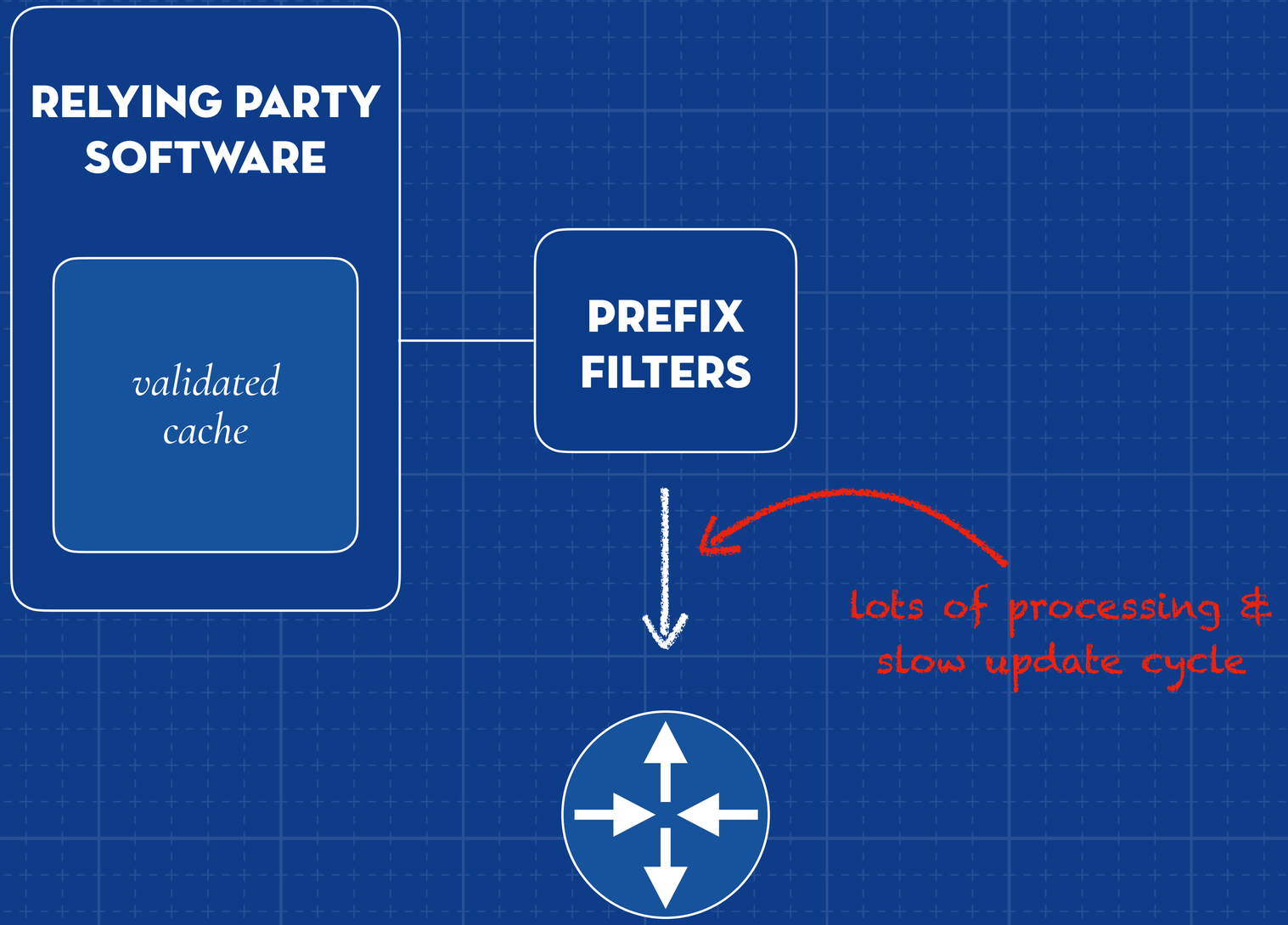
# ROUTING OBSERVABILITY


prefix filters
RELYING PARTY SOFTWARE
BGP session
validated cache
RTR protocol

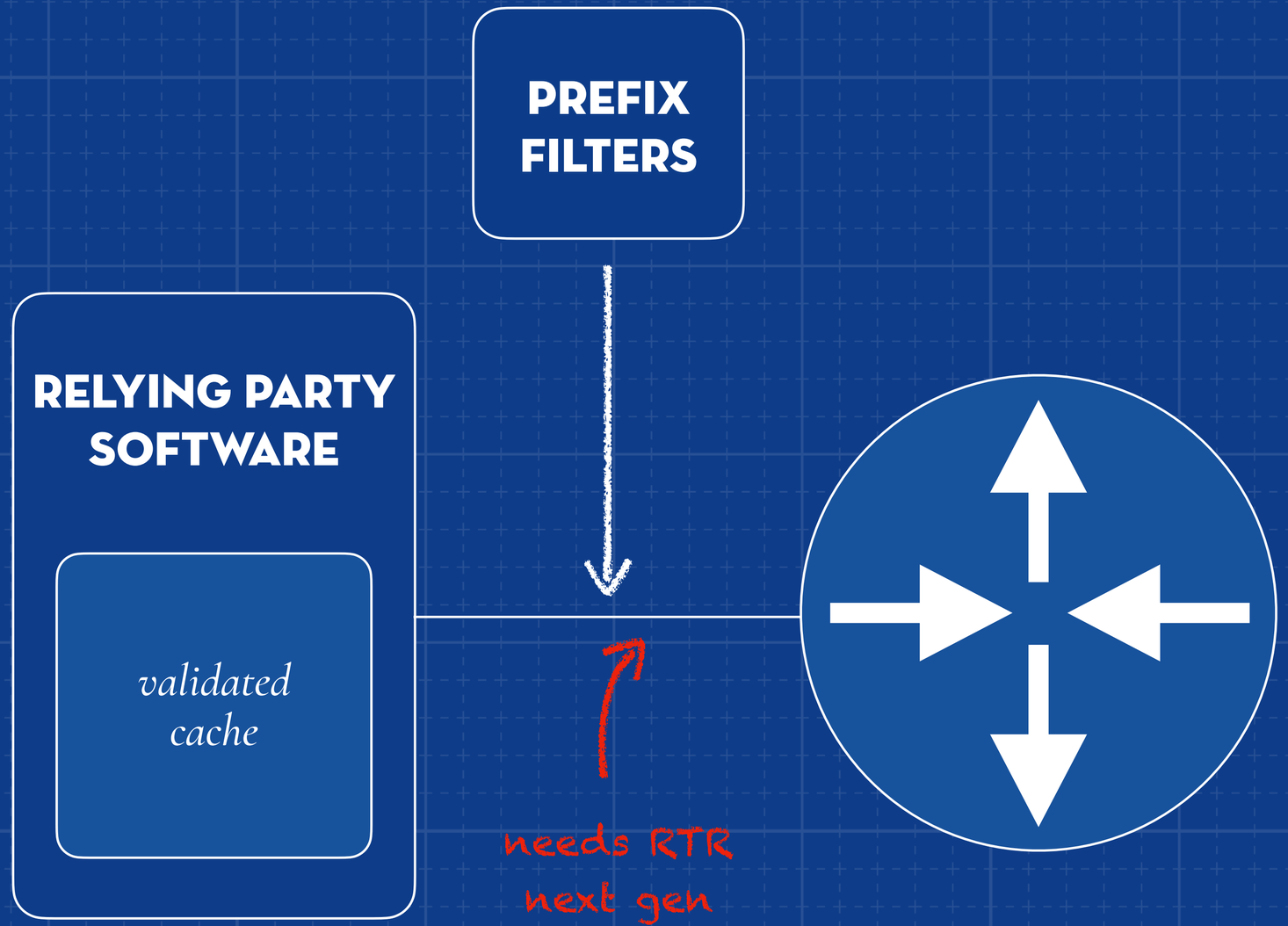



(from the juniper docs)

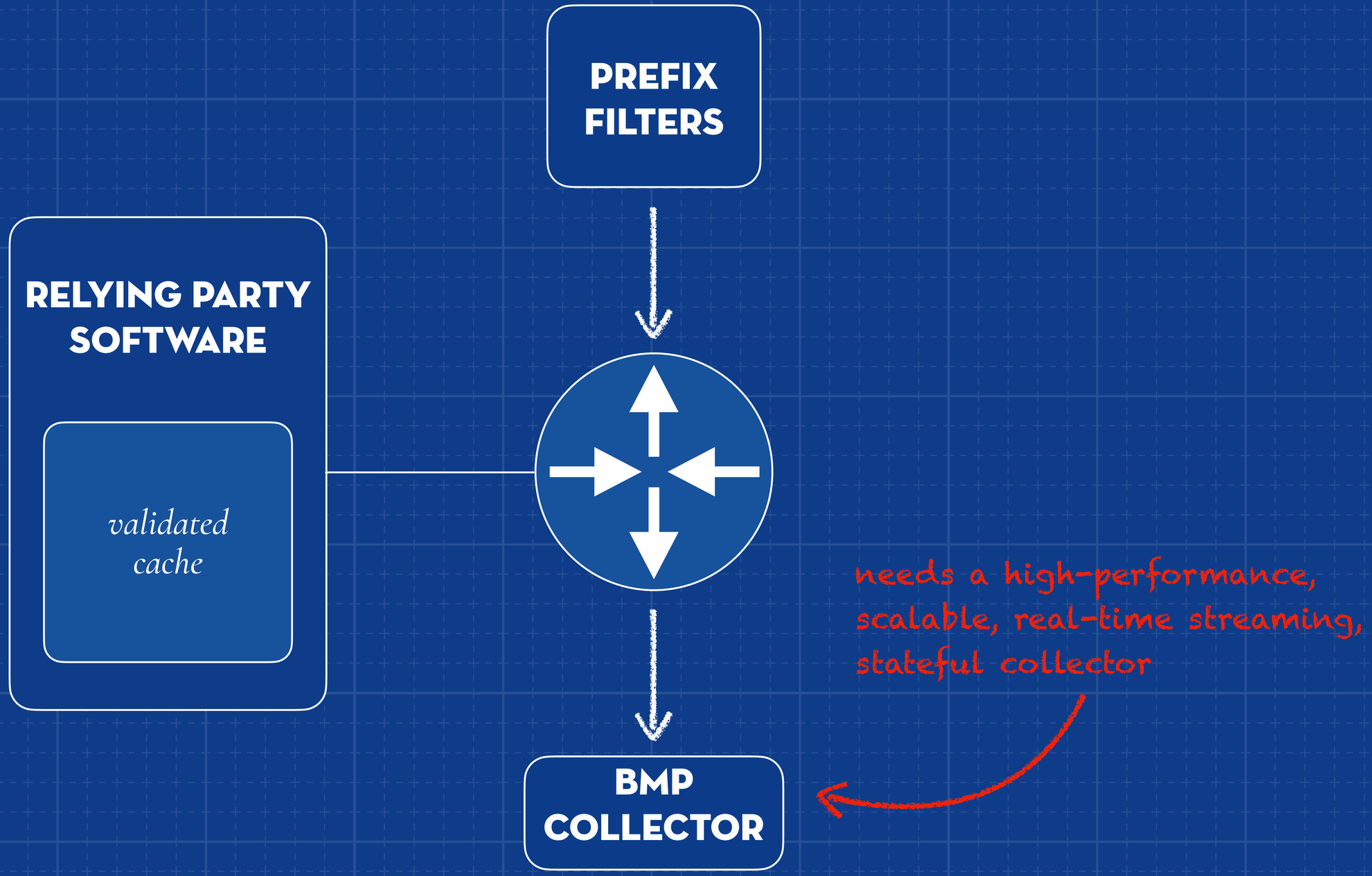
# A SOLUTION

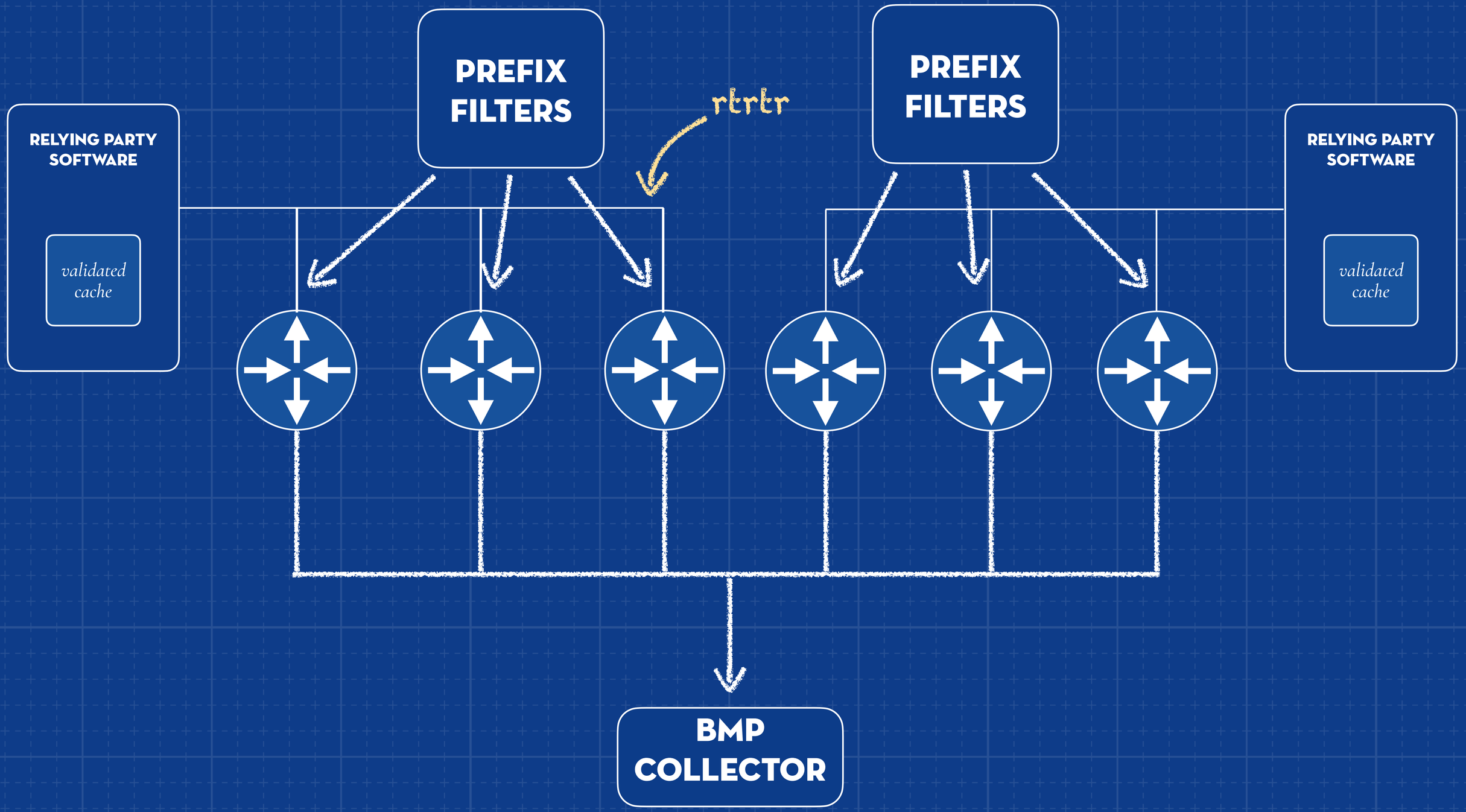

RELYING PARTY SOFTWARE
validated cache
PREFIX FILTERS
Lots of processing & slow update cycle

# A BETTER SOLUTION


PREFIX FILTERS
RELYING PARTY SOFTWARE
validated cache
needs RTR next gen

# ANOTHER SOLUTION


PREFIX FILTERS
RELYING PARTY SOFTWARE
validated cache
BMP COLLECTOR
needs a high-performance, scalable, real-time streaming, stateful collector


PREFIX FILTERS
rtrtr
PREFIX FILTERS
RELYING PARTY SOFTWARE
validated cache
RELYING PARTY SOFTWARE
validated cache
BMP COLLECTOR

# ANATOMY OF A COLLECTOR

# COMPOSABLE RUNTIMES

combine into single runtime

multiple runtimes connected by RTR-inspired protocol (codename “rotoro”)

builds on our RTRTR library

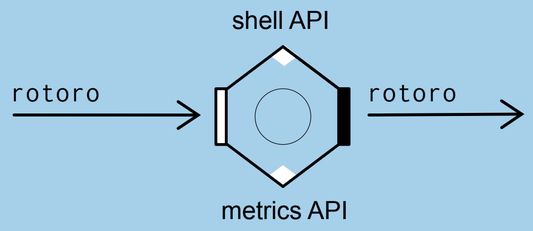
not only BMP, also BGP

many output formats, streams, files, pub/sub

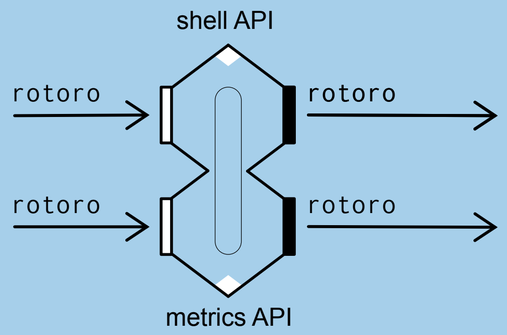
## RUNTIMES

### STORAGE

single input

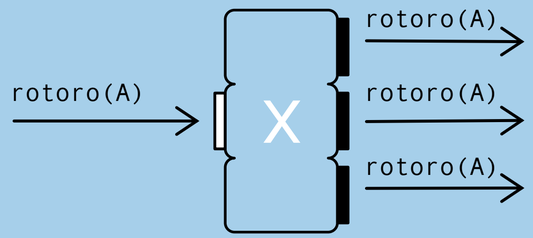



multiple input

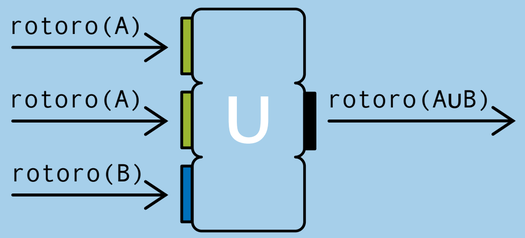



### CHANNEL OPERATORS

broadcast

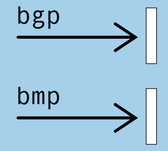



merge

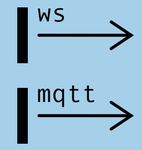



## TRANSFORMERS

input transformers




output transformers

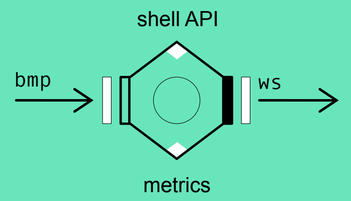


# TOWARDS ROUTE SERVICES

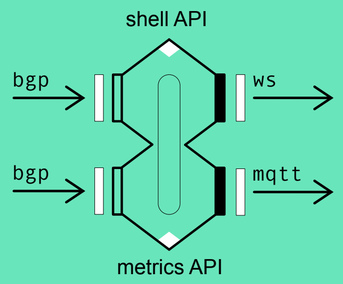
route collector
route monitor
(offline looking glass)
route server
route reflector

## EXAMPLE CONFIGURATIONS

single runtime/single transformed output

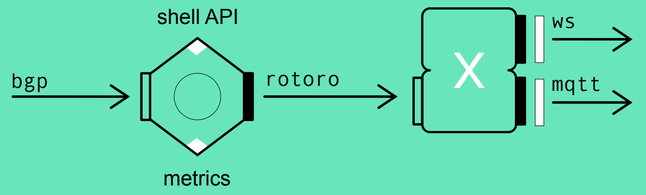



single runtime/multiple input/two transformers

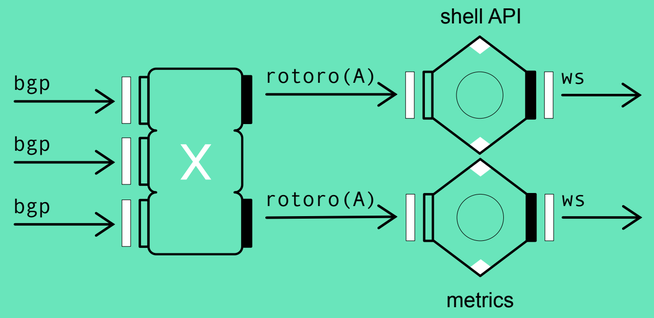



two runtimes

single input/two transformed outputs




redundant ingress



# SINGLE-THREADED PREFIX STORE

STORE

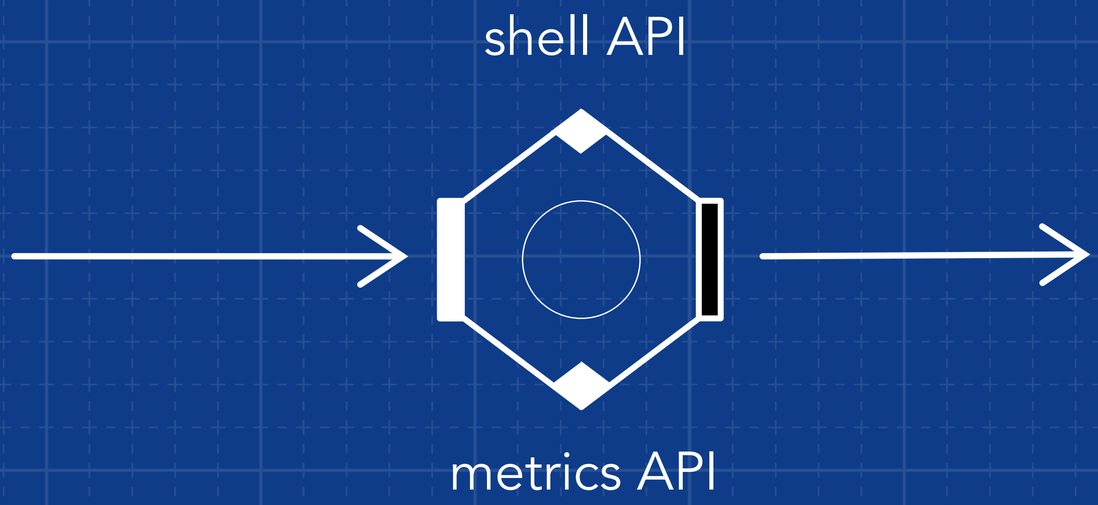
(prefix, meta-data)
pairs with hierarchy

INPUTS

single input/
single type

OUTPUTS

single output/
single type




can be a stand-alone runtime

# MULTI-THREADED PREFIX STORE

STORE

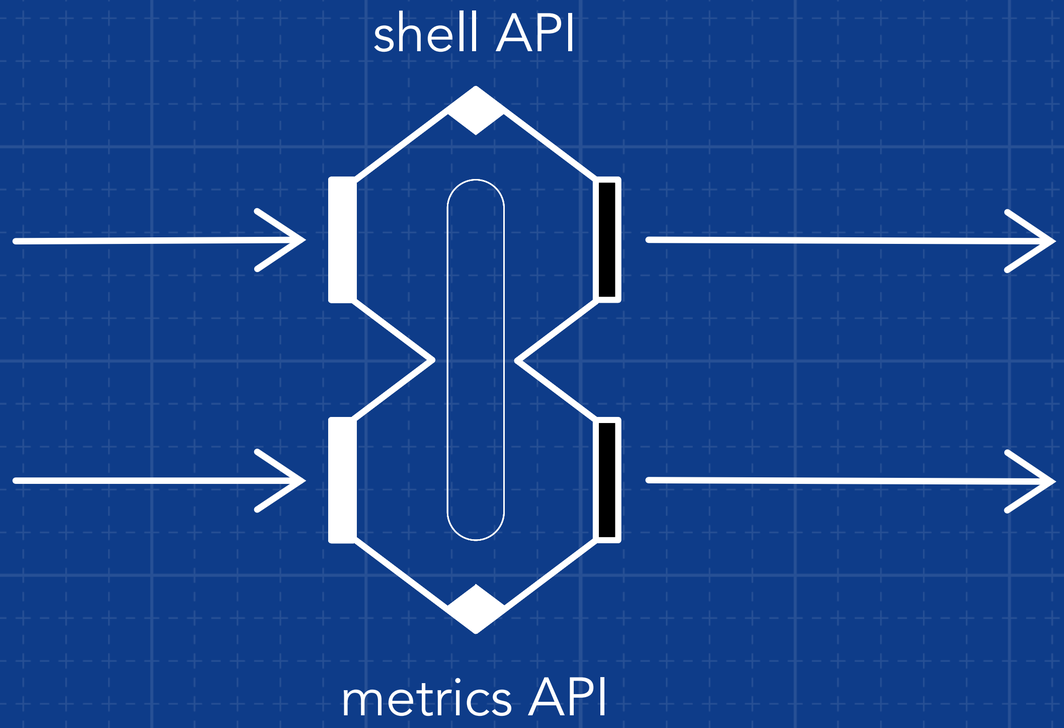
(prefix, meta-data)
pairs with hierarchy

INPUTS

multiple input/
single type

OUTPUTS

multiple output/
single type




can be a stand-alone runtime

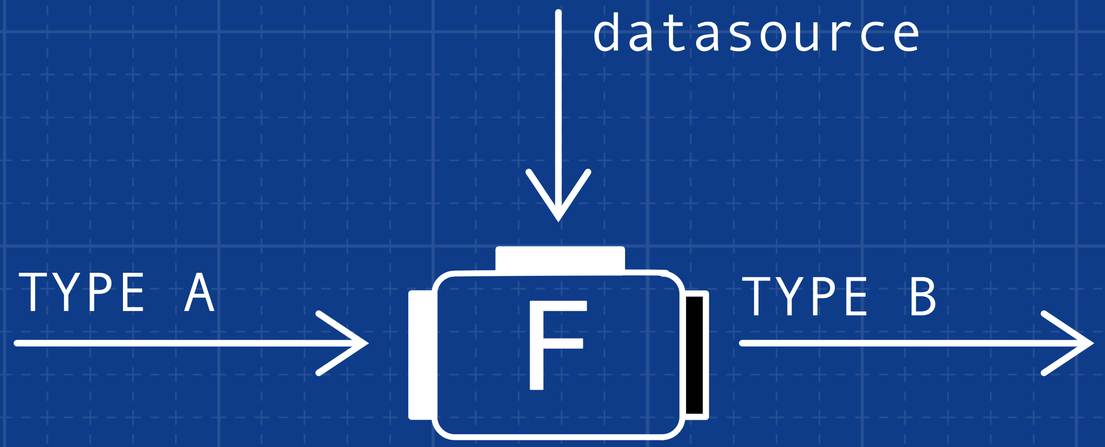
# PROGRAMMABLE FILTERMAP - 1

INPUTS

a single typed data stream

OUTPUTS

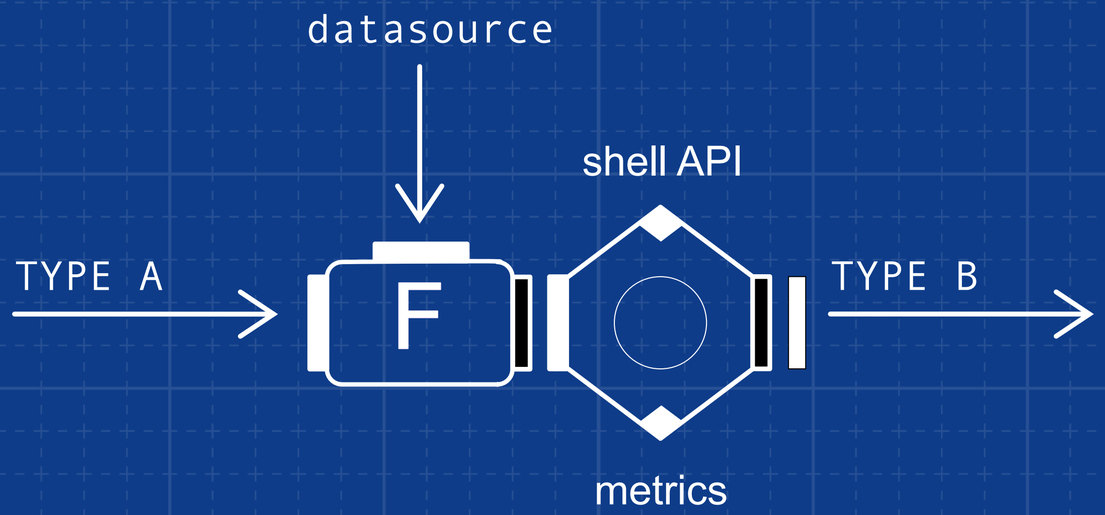
a single typed data stream




lives in a filter stack in front of a RIB

# PROGRAMMABLE FILTERMAP - 2

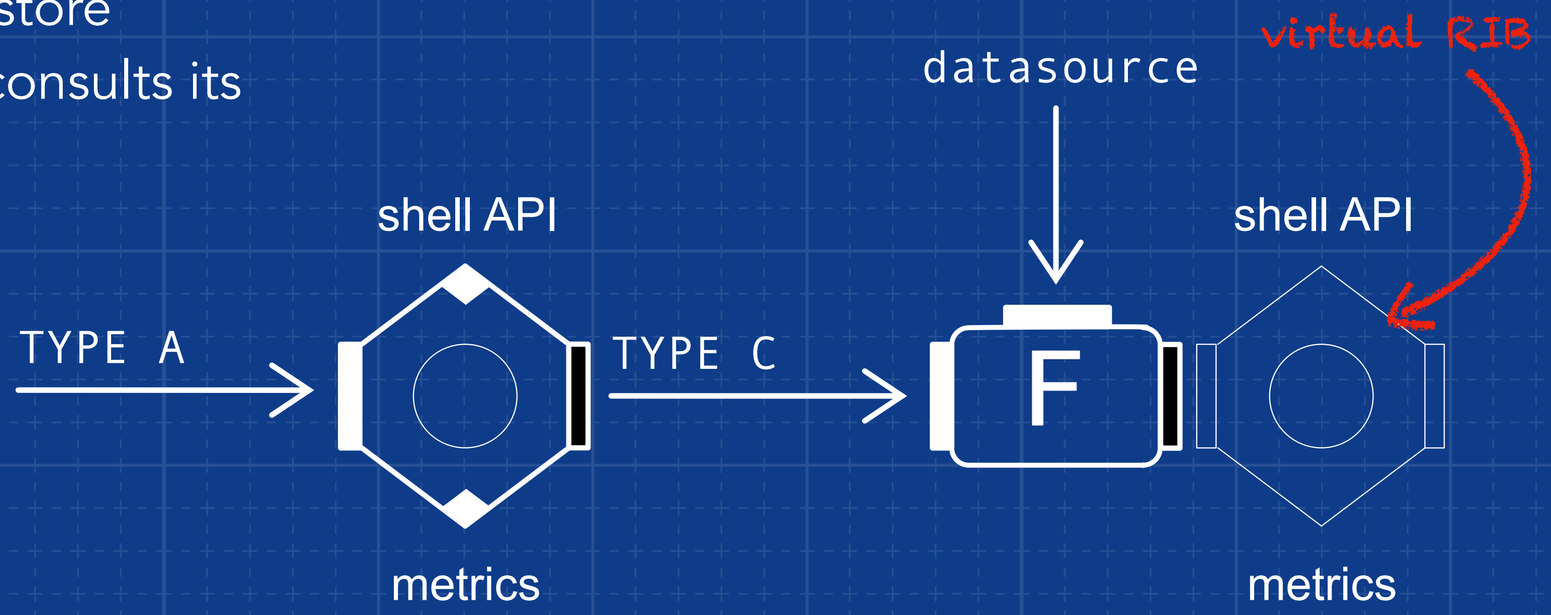
PROCESSING
- ROTO filter/config language
- accept/reject
- transform input stream
- realtime per packet processing




lives in a filter stack in front of a RIB

# VIRTUAL RIB

same functionality as a normal RIB, but doesn't actually store anything, instead it consults its western RIB




lives in a filter stack in front of a RIB

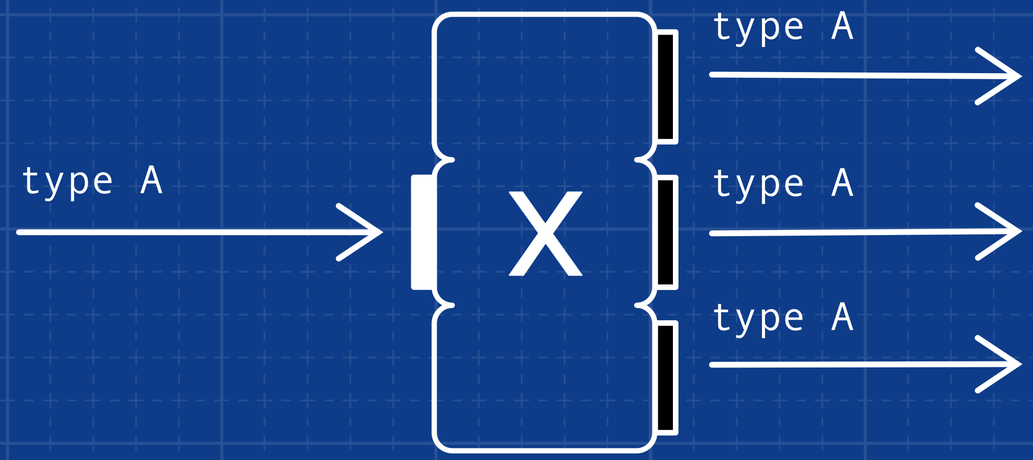
# BROADCAST STREAM

INPUTS

single input/
single type

OUTPUTS

multiple outputs/
single type




can be a stand-alone runtime

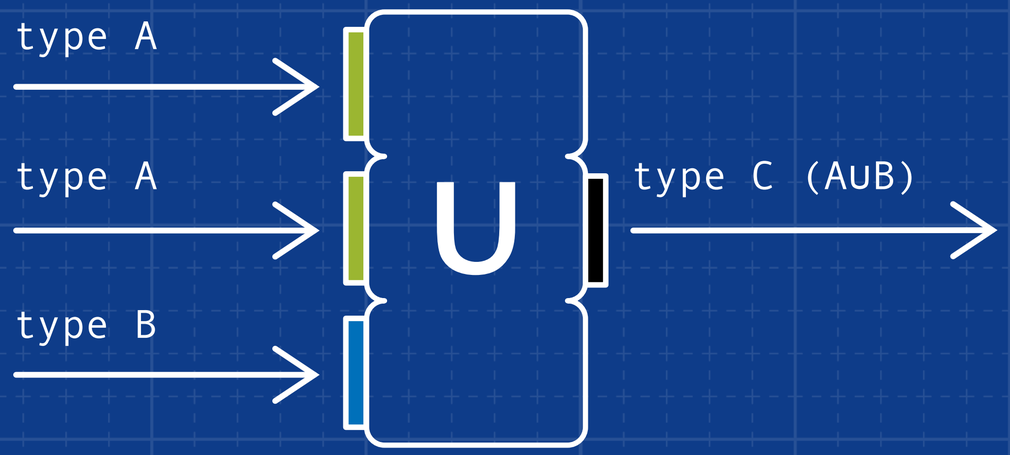
# MERGE STREAMS

INPUTS

multiple inputs/
multiple types

OUTPUTS

single output/
single type




can be a stand-alone runtime

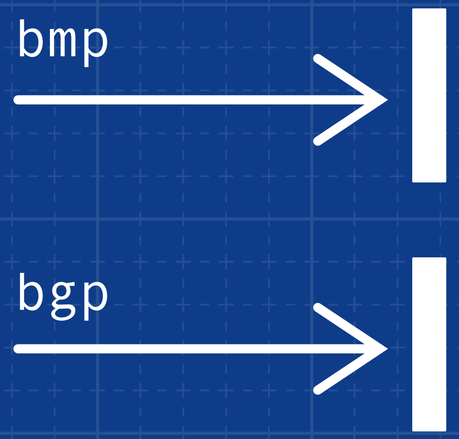
# CONNECTORS: INPUTS

INPUTS

single input/
single types

OUTPUTS

single output/
single type




attached to a runtime

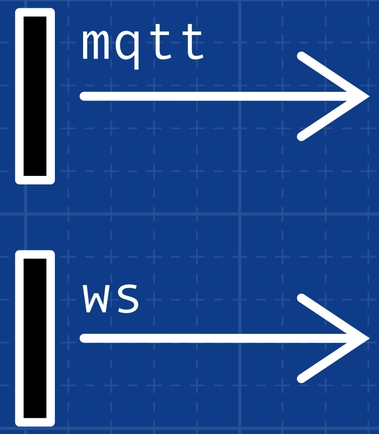
# CONNECTORS: OUTPUTS

INPUTS

single input/
single type

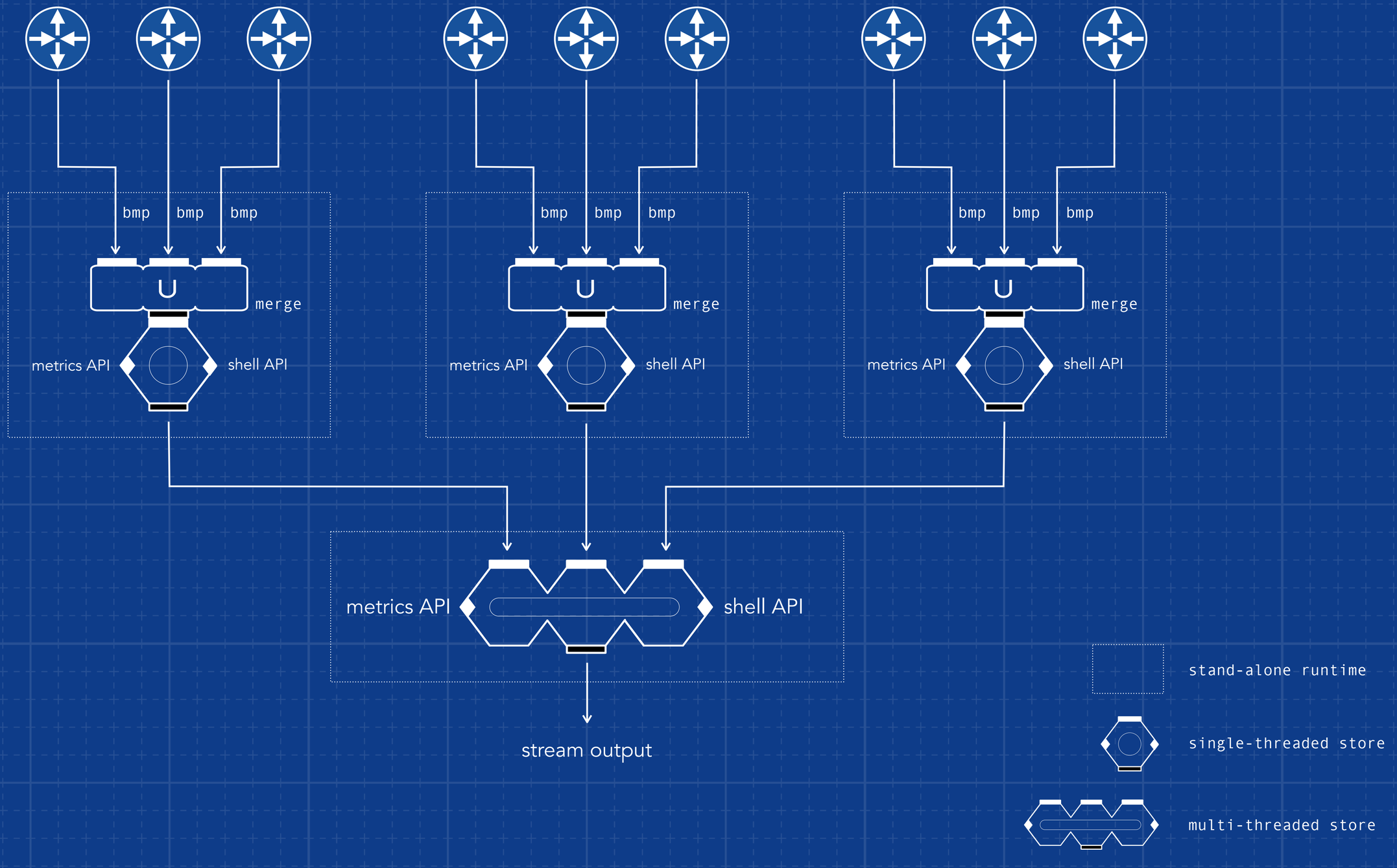
OUTPUTS

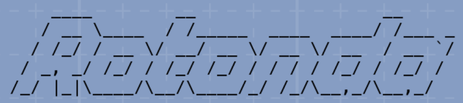
single output/
single type

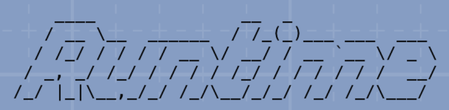



attached to a runtime

# WORK IN PROGRESS


bmp
bmp
bmp
U
merge
metrics API
shell API
bmp
bmp
bmp
U
merge
metrics API
shell API
bmp
bmp
bmp
U
merge
metrics API
shell API
metrics API
shell API
stream output
stand-alone runtime
single-threaded store
multi-threaded store





UNIT BmpRouters: PushEventSource

(from external sources) ⟹

PushBmp | session state

UNIT Api: RequestEventSource

(from end-user) ⟹

InPreQuery

InRovQuery

CustomerPreRovQuery

UNIT RpkiStream: PushEventSource

(from external sources) ⟹

RpkiStream

UNIT Api: RequestEventSource

(from end-user) ⟹

RovQuery

RovInQuery

CustomersPostRovQuery

UNIT RibInPre: RIB

FilterStack

PrePolicyFilter

Write (RibInPre)

Read

UNIT RibRov: RIB

FilterStack

RovFilter

WriteStore

Read

UNIT RibCustomerPreRov: VirtualRIB

FilterStack

CustomerFilter

UNIT RibRov: VirtualRIB

FilterStack

RovFilter

Read (RibRov)

UNIT RibInPre: VirtualRIB

FilterStack

PrePolicyFilter

Read (RibInPre)

UNIT RibInPost: VirtualRIB

FilterStack

PostPolicyFilter

UNIT RibCustomersPostRov: VirtualRIB

FilterStack

CustomerFilter

UNIT BgpUpdate: EmitEvent

⟹ (to other rotonda runtime over rotoro)

UNIT EmitRequestResult

⟹ (to end-user API)

```
GlobalEventRegistry

PushBMP: PushEventSource {
    source: BmpRouters,
}

RpkiStream: PushEventSource {
    source: RpkiStream,

}

Api: RequestEventSource {
    query: QueryType
}
```

```
GlobalFilterRegistry

[
    PrePolicyFilter,
    PostPolicyFilter,
    RovFilter,
    CustomerFilter
]
```

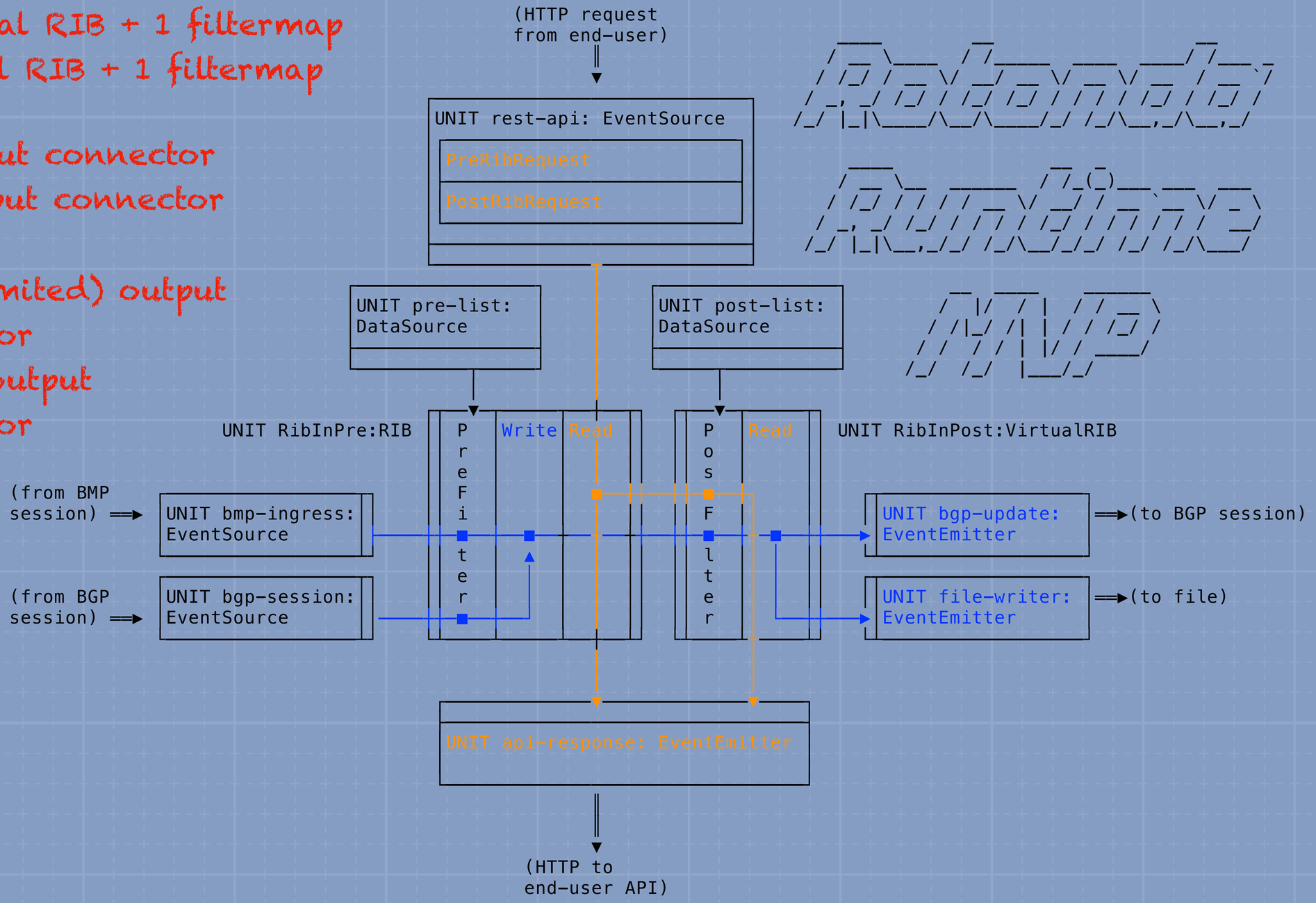
```
GlobalStoreRegistry

[
    RibInPre: Store,
    RibRov: Store,
    RibInPost: VirtualStore,
    RibCustomerPreRov: VirtualStore,
    RibCustomerPostRov: VirtualStore
]
```

- 1 physical RIB + 1 filtermap
- 1 virtual RIB + 1 filtermap

- BMP input connector
- BGP input connector

- BGP (limited) output connector
- to file output connector

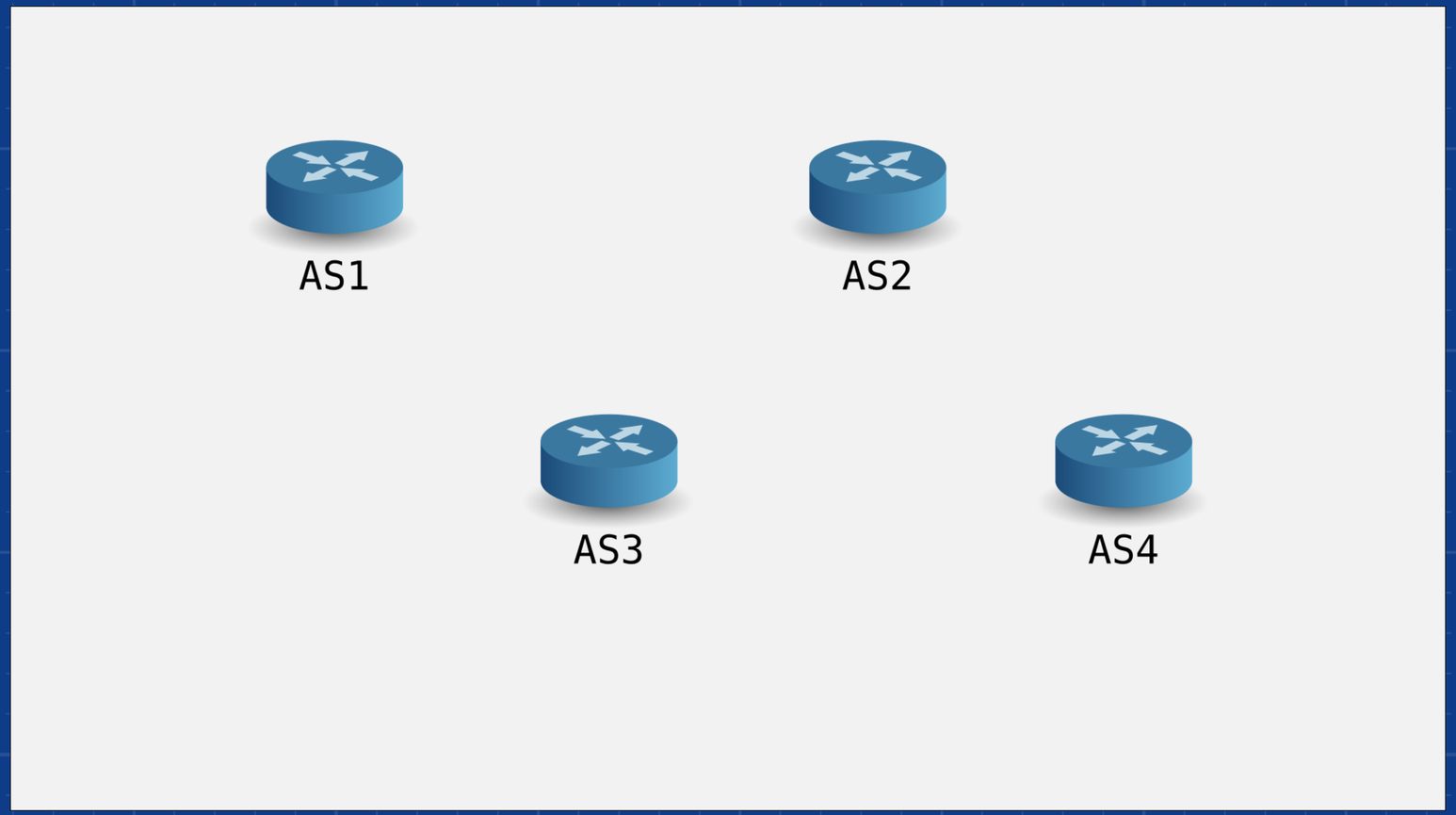


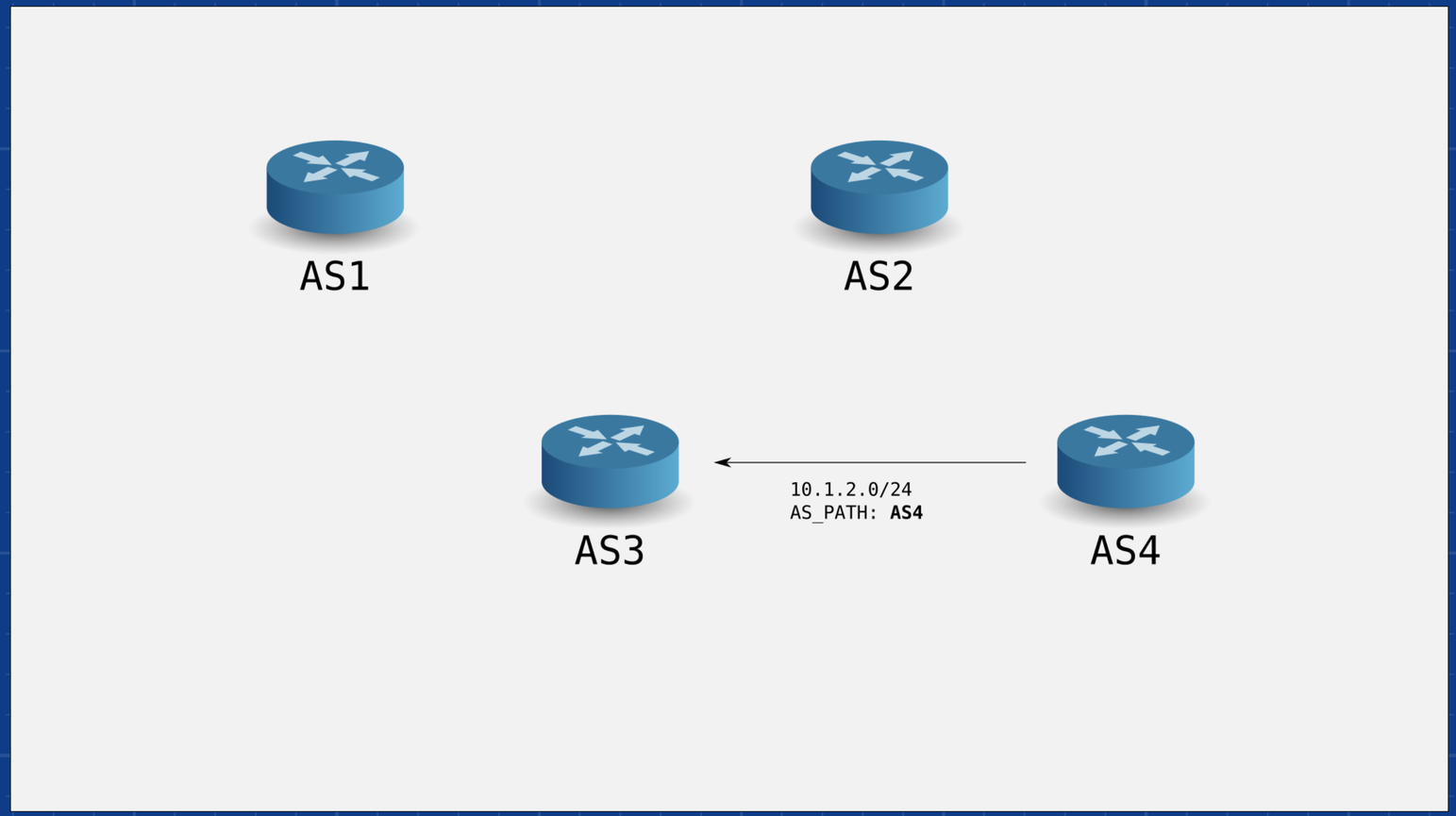
# THANK YOU

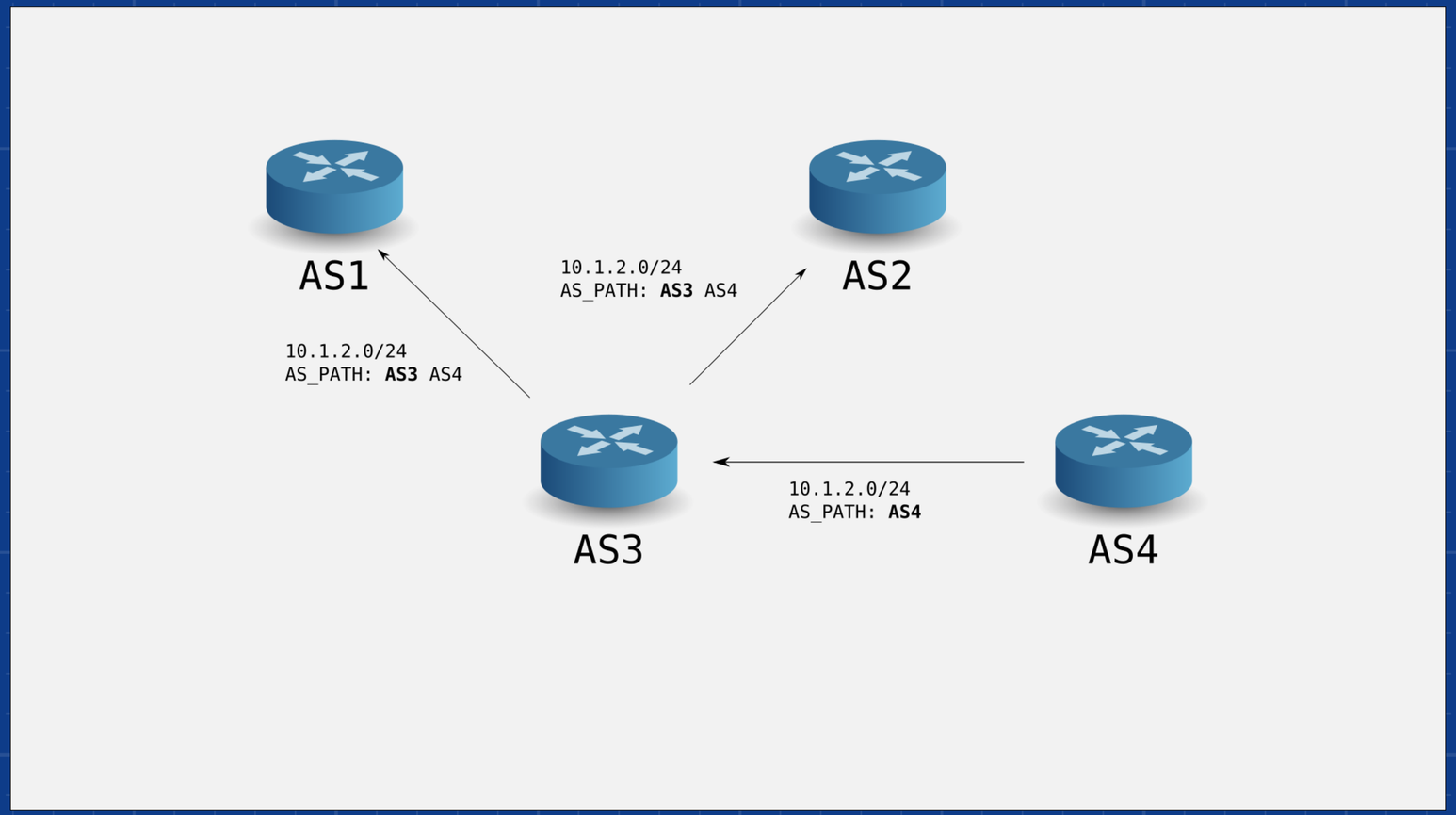
# BMP AS DATA INGRESS

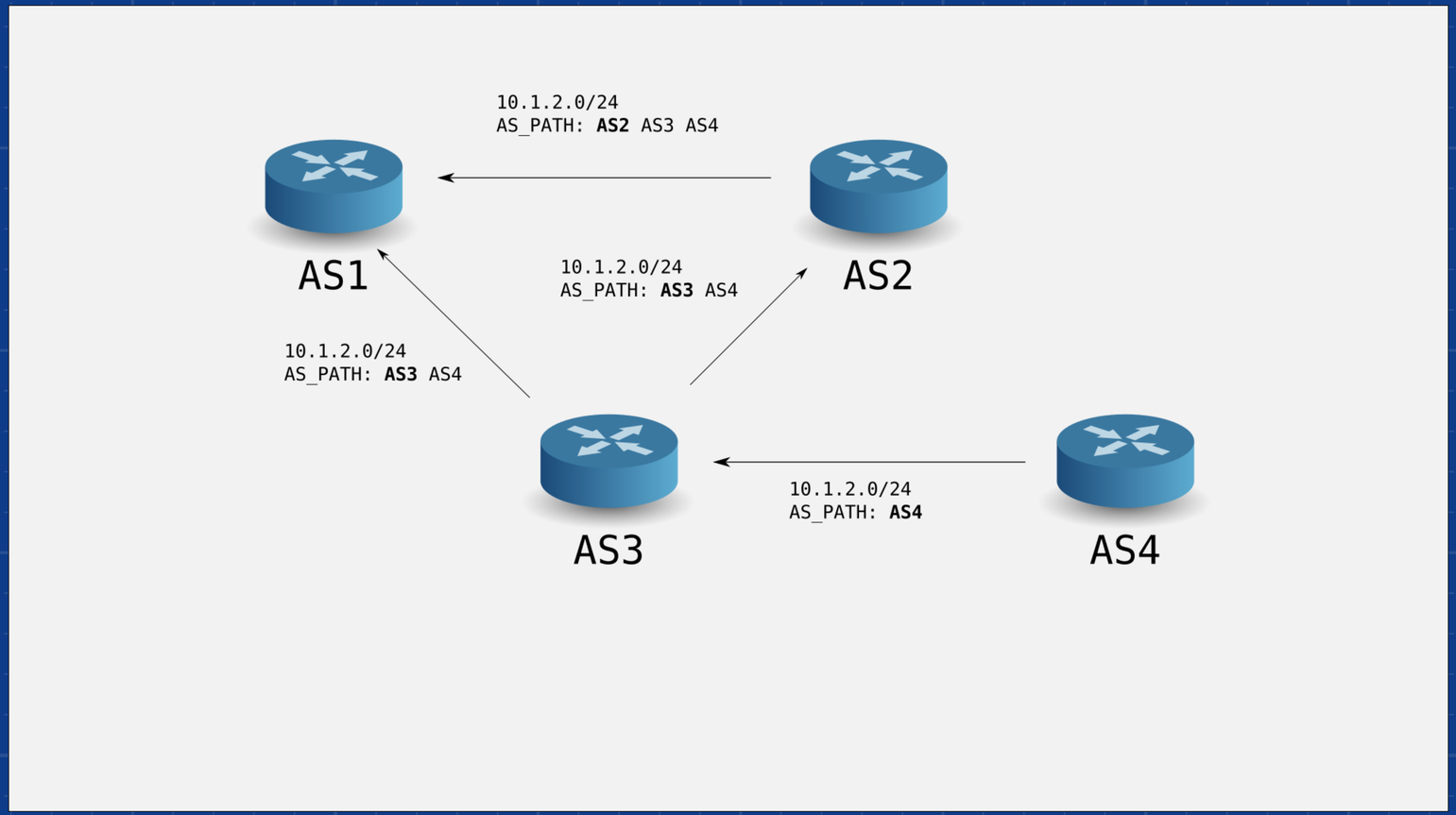
*"BMP provides access to the Adj-RIB-In of a peer on an ongoing basis (...).*
*From a high level, BMP can be thought of as the result of multiplexing together the messages received on the various monitored BGP sessions."*

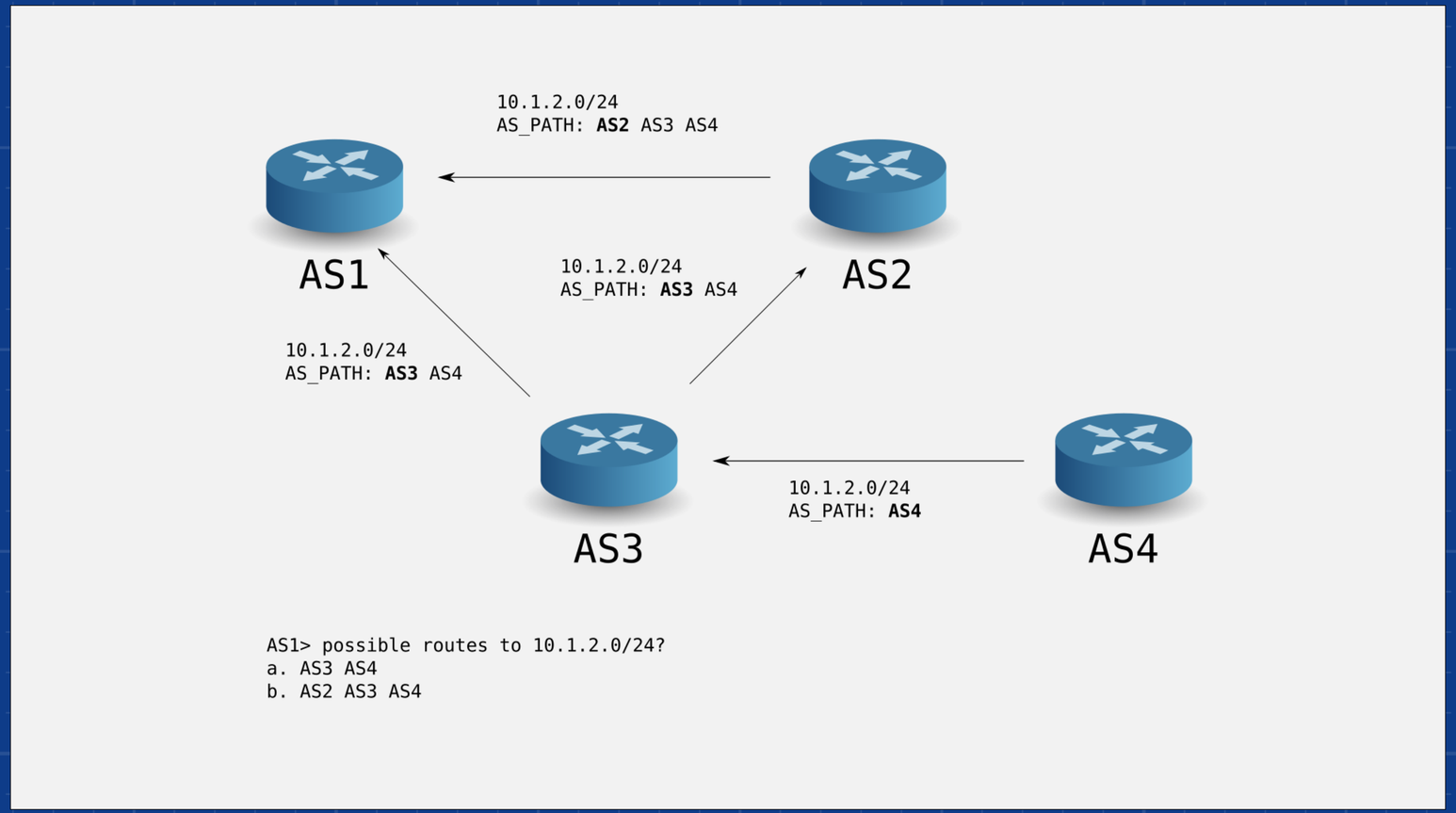
from "BMP: BGP Monitoring Protocol" (RFC 7854)


AS1
AS2
AS3
AS4

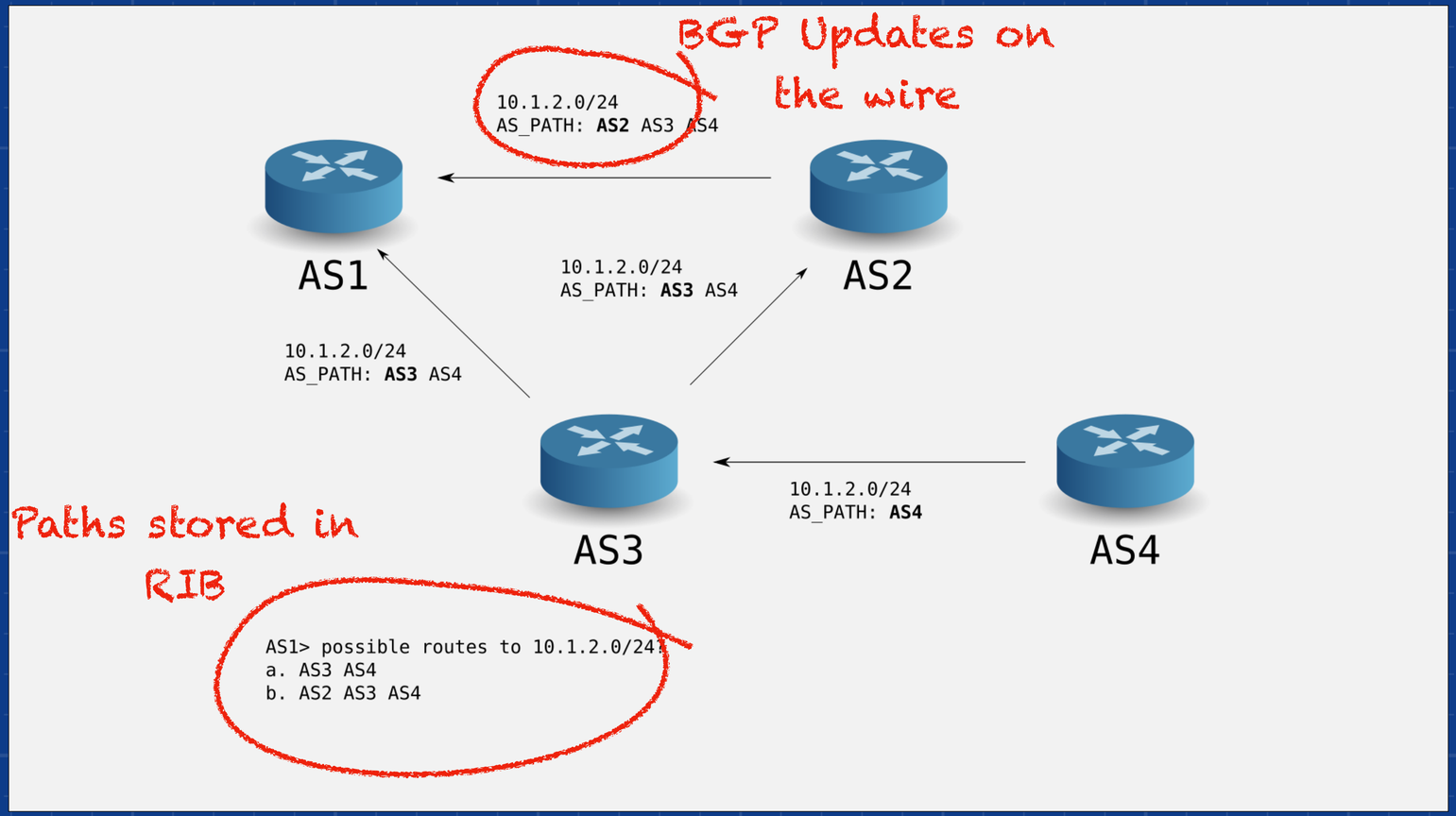

AS1
AS2
AS3
AS4
10.1.2.0/24
AS_PATH: AS4


AS1
AS2
AS3
AS4
10.1.2.0/24
AS_PATH: AS3 AS4
10.1.2.0/24
AS_PATH: AS3 AS4
10.1.2.0/24
AS_PATH: AS4


10.1.2.0/24
AS_PATH: **AS2** AS3 AS4
AS1
AS2
10.1.2.0/24
AS_PATH: **AS3** AS4
10.1.2.0/24
AS_PATH: **AS3** AS4
AS3
10.1.2.0/24
AS_PATH: **AS4**
AS4




AS1> possible routes to 10.1.2.0/24?
a. AS3 AS4
b. AS2 AS3 AS4


BGP Updates on the wire
10.1.2.0/24
AS_PATH: AS2 AS3 AS4
AS1
AS2
10.1.2.0/24
AS_PATH: AS3 AS4
10.1.2.0/24
AS_PATH: AS3 AS4
AS3
10.1.2.0/24
AS_PATH: AS4
AS4
Paths stored in RIB
AS1> possible routes to 10.1.2.0/24?
a. AS3 AS4
b. AS2 AS3 AS4

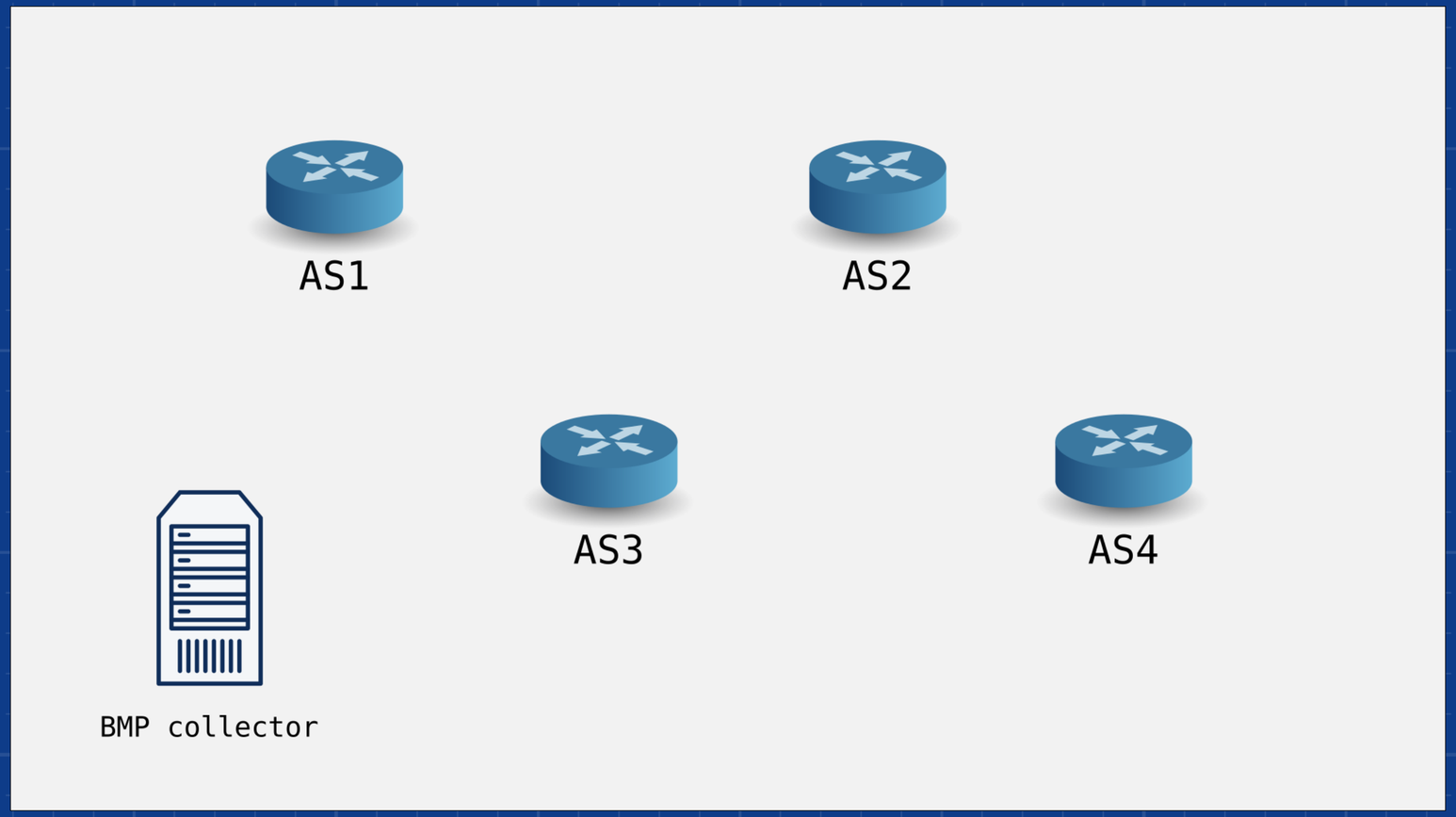
# GETTING DATA FROM YOUR ROUTERS

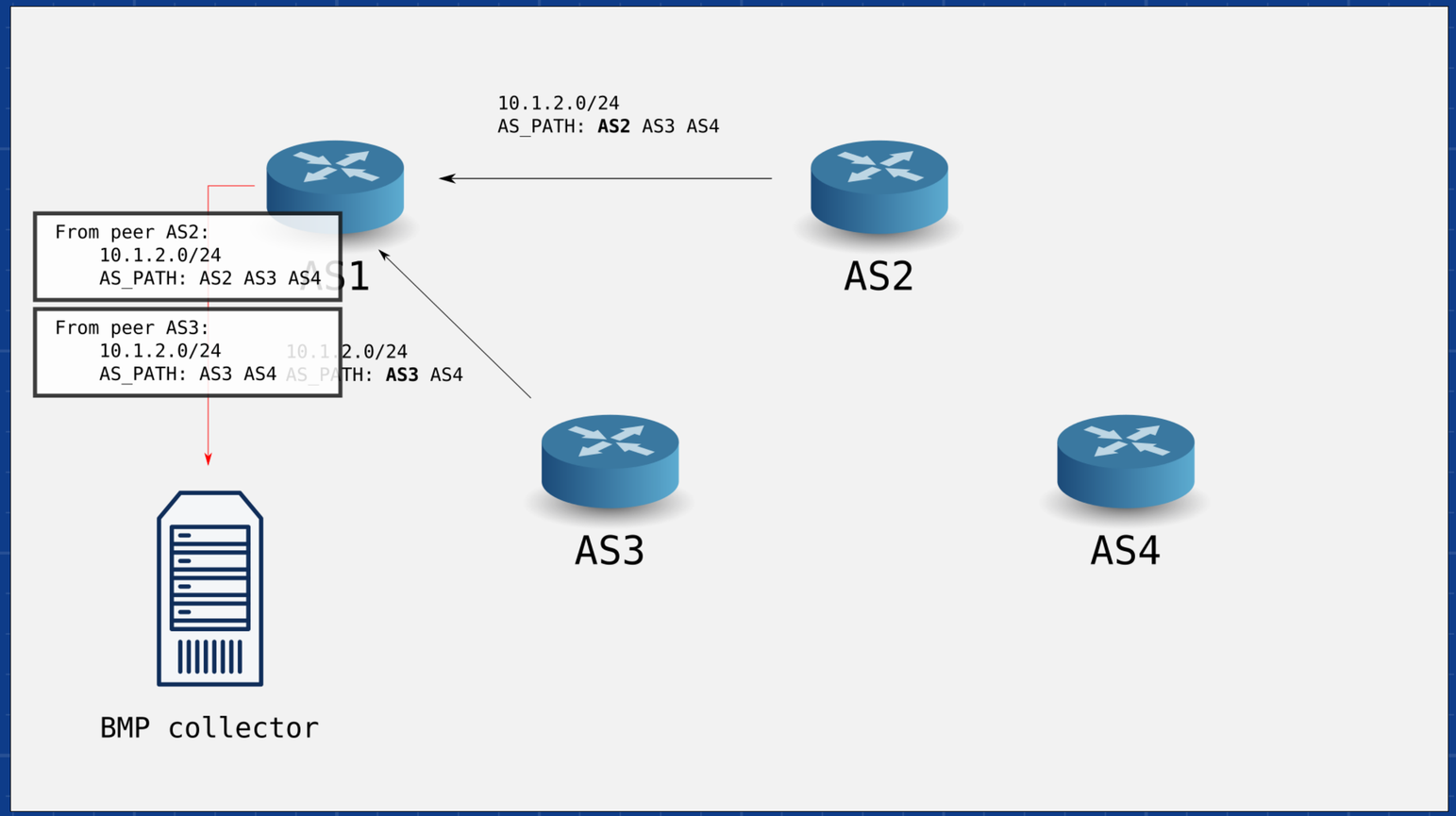
# PRE BMP

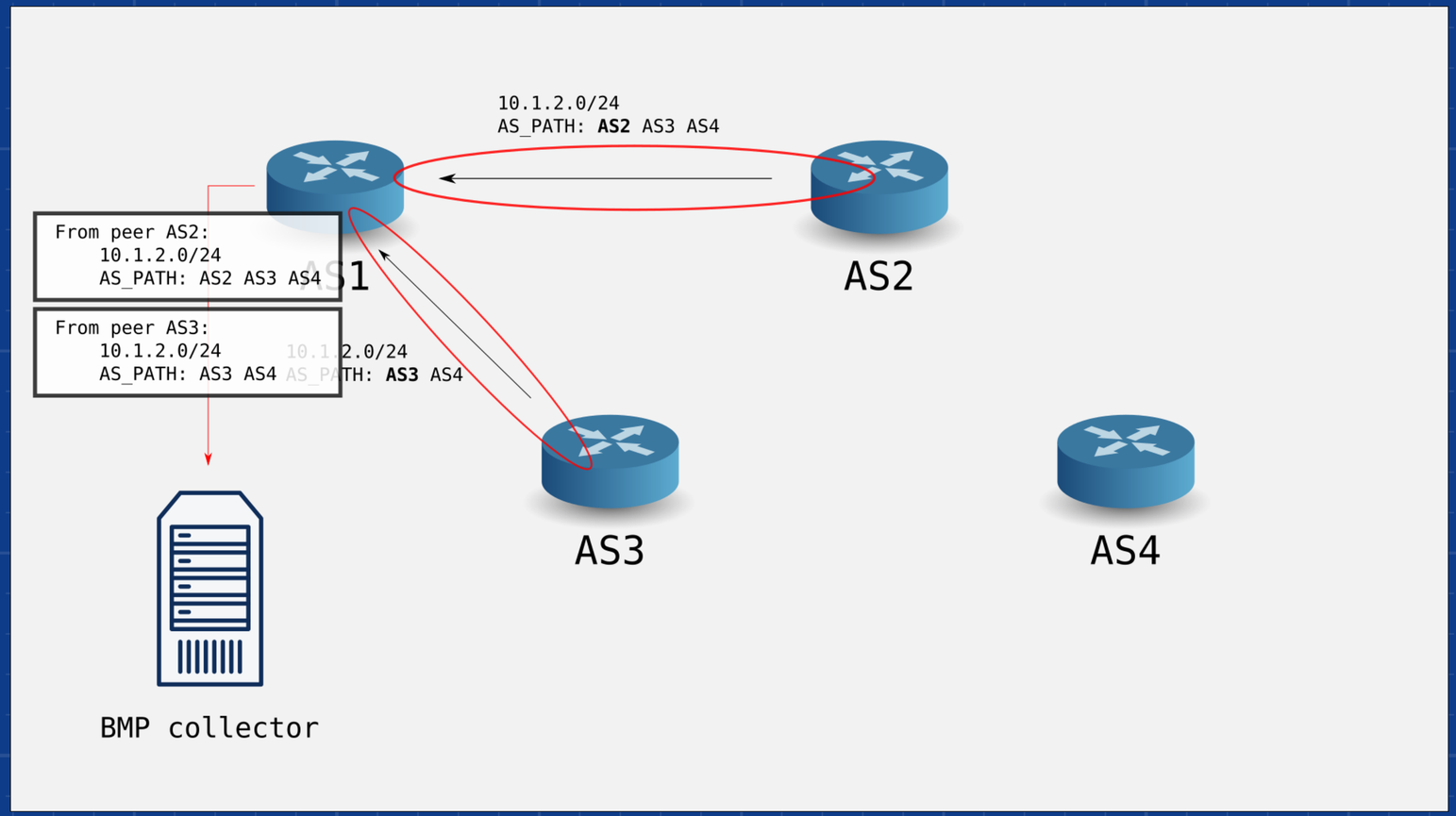
- Scraping the CLI
- Configure a peering session with the router, receive all routes via BGP

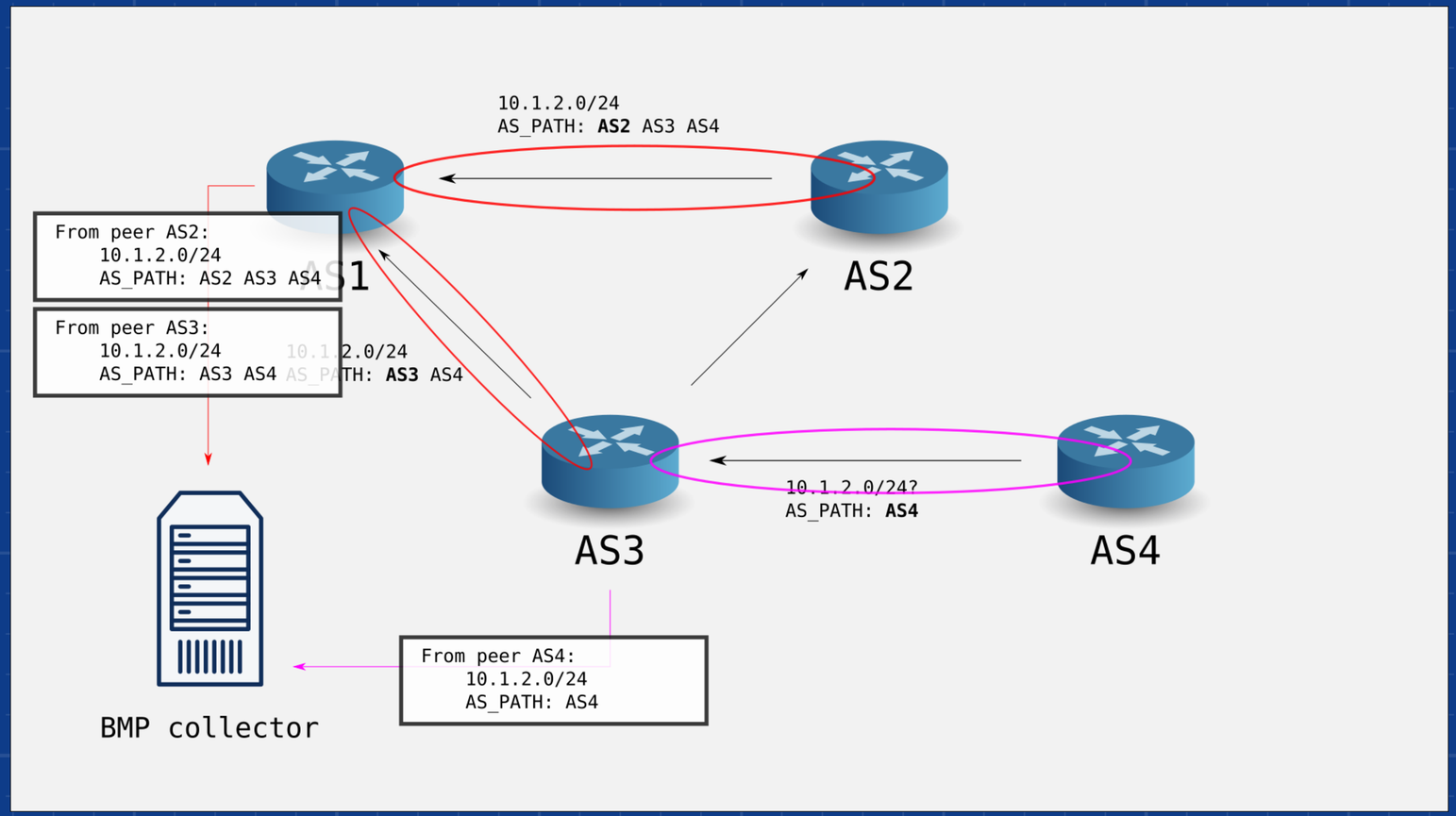
# WITH BMP

- Uni-directional stream of data over a single TCP connection from router to BMP station (‘Collector’)
- Re-using BGP wire-formats,  'encapsulating' BGP UPDATEs with Per-Peer Headers
- Additional information available: Statistics Reports with various counters, session related configuration


AS1
AS2
AS3
AS4
BMP collector


10.1.2.0/24
AS_PATH: **AS2** AS3 AS4
From peer AS2:
10.1.2.0/24
AS_PATH: AS2 AS3 AS4
From peer AS3:
10.1.2.0/24
AS_PATH: AS3 AS4
10.1.2.0/24
AS_PATH: **AS3** AS4
AS1
AS2
AS3
AS4
BMP collector


10.1.2.0/24
AS_PATH: AS2 AS3 AS4
From peer AS2:
10.1.2.0/24
AS_PATH: AS2 AS3 AS4
From peer AS3:
10.1.2.0/24
AS_PATH: AS3 AS4
10.1.2.0/24
AS_PATH: AS3 AS4
AS1
AS2
AS3
AS4
BMP collector


10.1.2.0/24
AS_PATH: AS2 AS3 AS4
From peer AS2:
10.1.2.0/24
AS_PATH: AS2 AS3 AS4
From peer AS3:
10.1.2.0/24
AS_PATH: AS3 AS4
10.1.2.0/24
AS_PATH: AS3 AS4
10.1.2.0/24?
AS_PATH: AS4
AS1
AS2
AS3
AS4
From peer AS4:
10.1.2.0/24
AS_PATH: AS4
BMP collector

# WHAT HAPPENED BEFORE "STEADY STATE"?

# WITH BMP

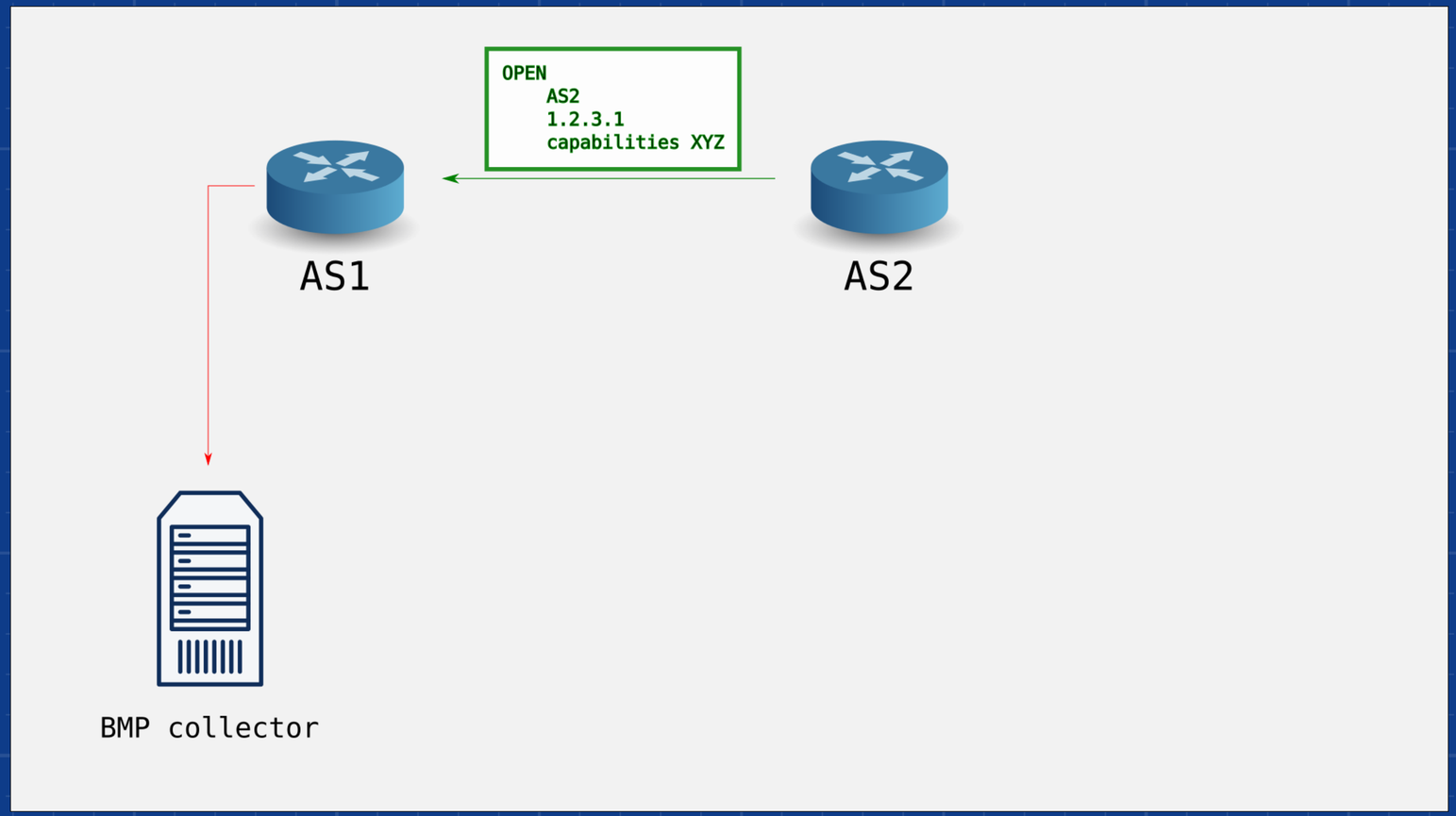
- What about the BGP UPDATEs exchanged before our BMP session was established?
- What happens when a new BGP session between our monitored router and a peer is established?

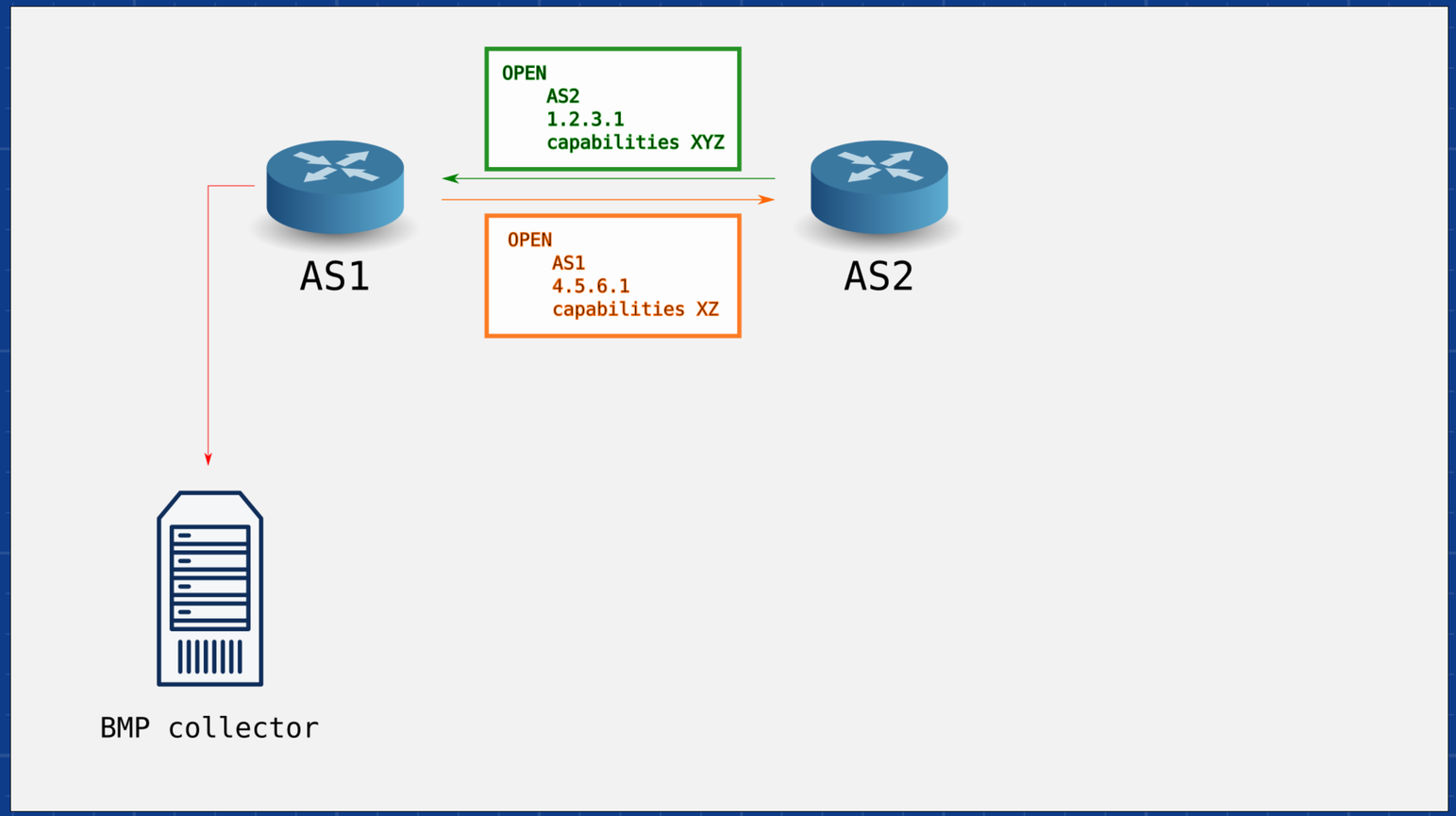
# WITH BMP

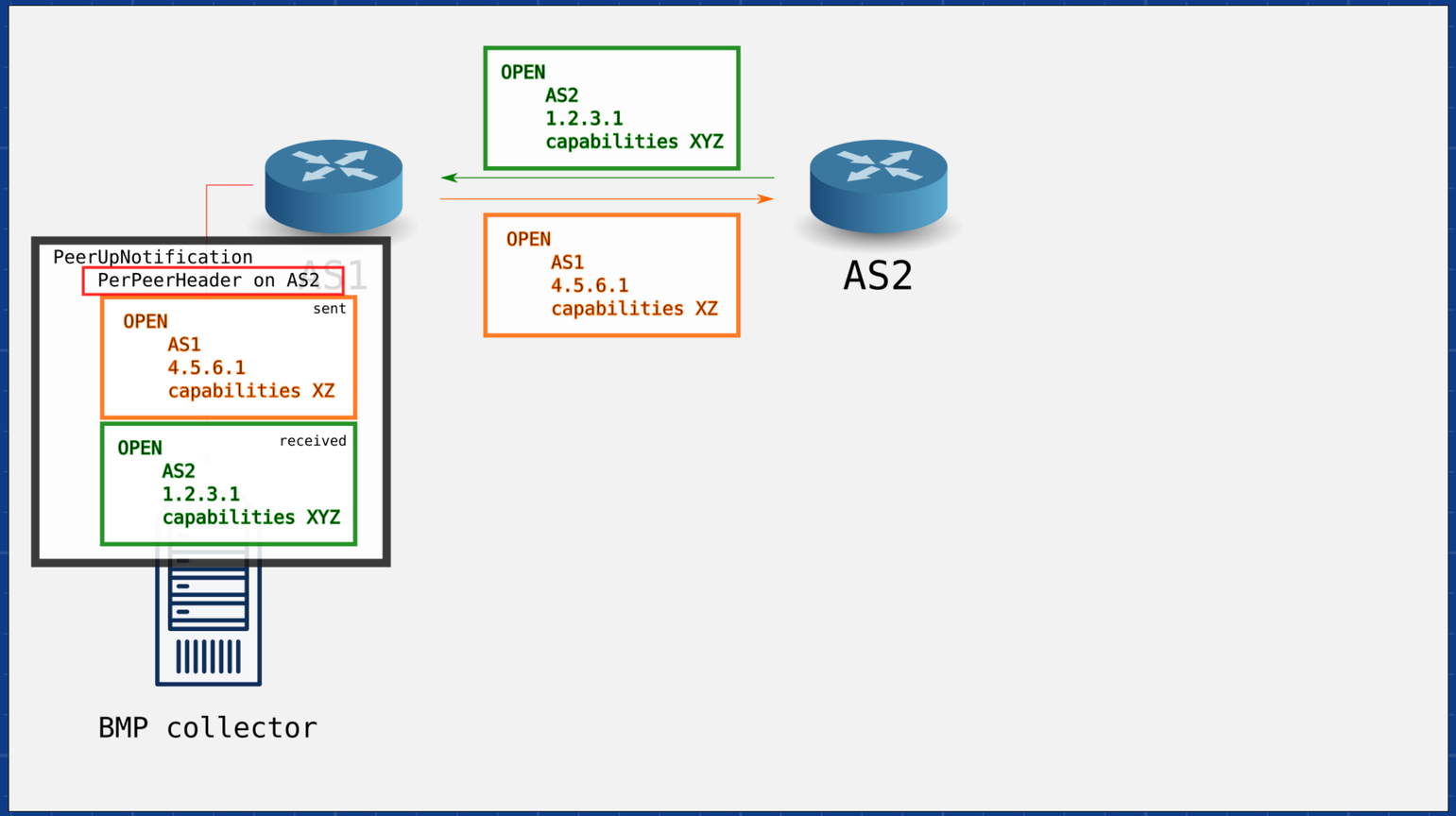
All contents of the monitored RIB are exported to the collector using the same BMP messages as when a new BGP sessions comes up:

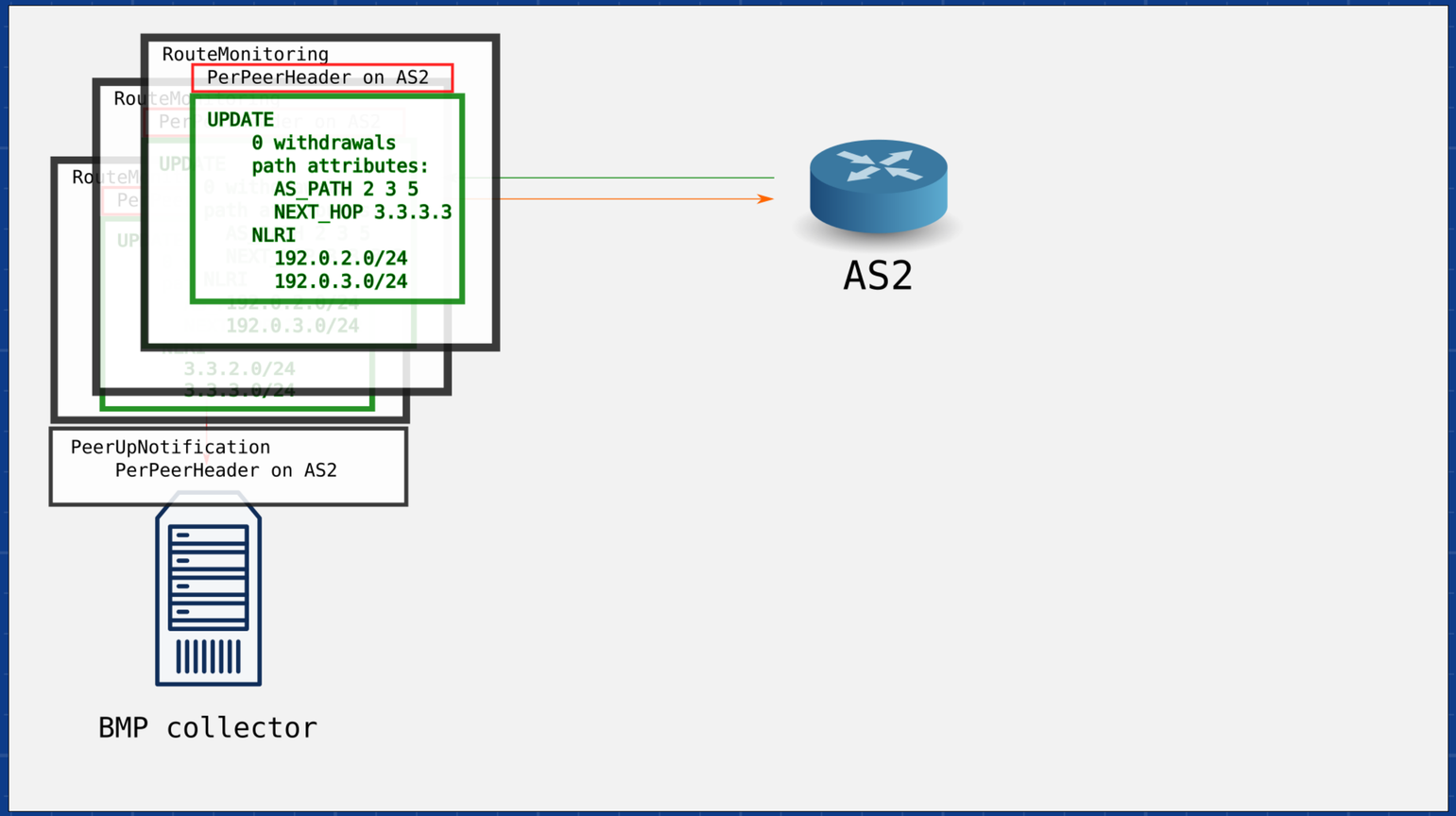
```
PeerUpNotification,
[RouteMonitoringMessage, ..],
RM-with-EoR,
steady-state
```

AS1
AS2
BMP collector


OPEN
AS2
1.2.3.1
capabilities XYZ
AS1
AS2
BMP collector


OPEN
AS2
1.2.3.1
capabilities XYZ
OPEN
AS1
4.5.6.1
capabilities XZ
AS1
AS2
BMP collector


OPEN
AS2
1.2.3.1
capabilities XYZ
OPEN
AS1
4.5.6.1
capabilities XZ
AS2
PeerUpNotification
PerPeerHeader on AS2
sent
OPEN
AS1
4.5.6.1
capabilities XZ
received
OPEN
AS2
1.2.3.1
capabilities XYZ
BMP collector


RouteMonitoring
PerPeerHeader on AS2
UPDATE
0 withdrawals
path attributes:
AS_PATH 2 3 5
NEXT_HOP 3.3.3.3
NLRI
192.0.2.0/24
192.0.3.0/24
PeerUpNotification
PerPeerHeader on AS2
AS2
BMP collector

# OUR EXPERIENCES SO FAR

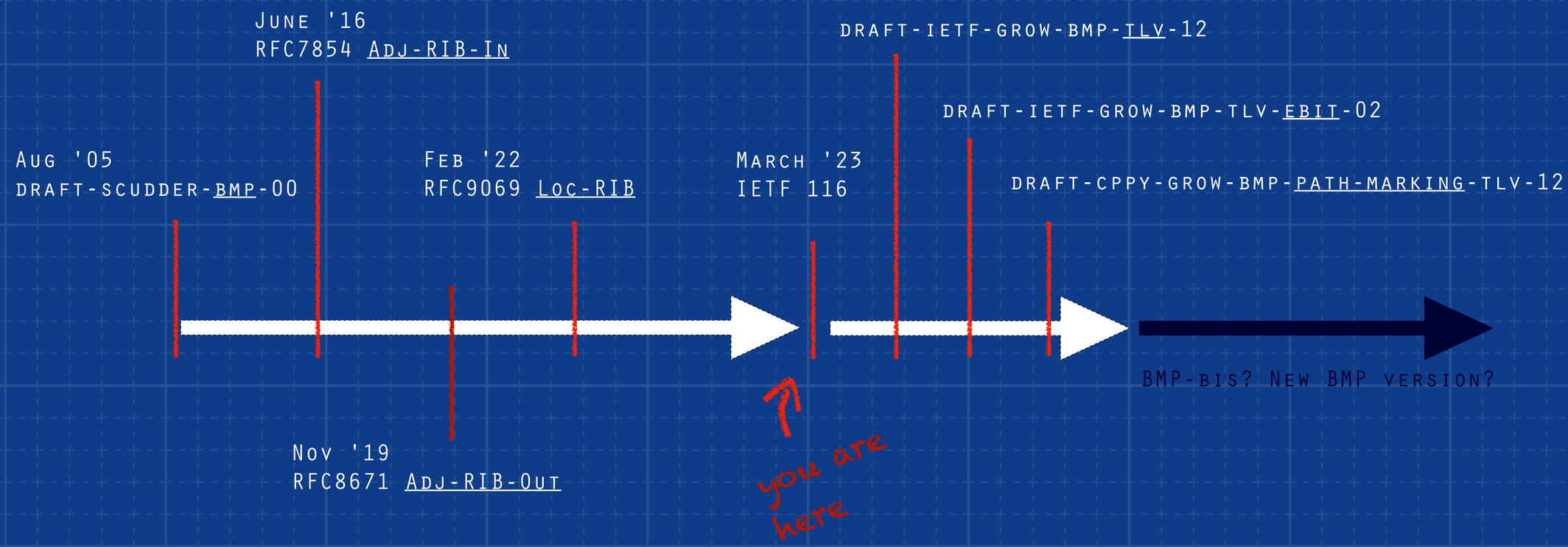
# THE GOOD STUFF

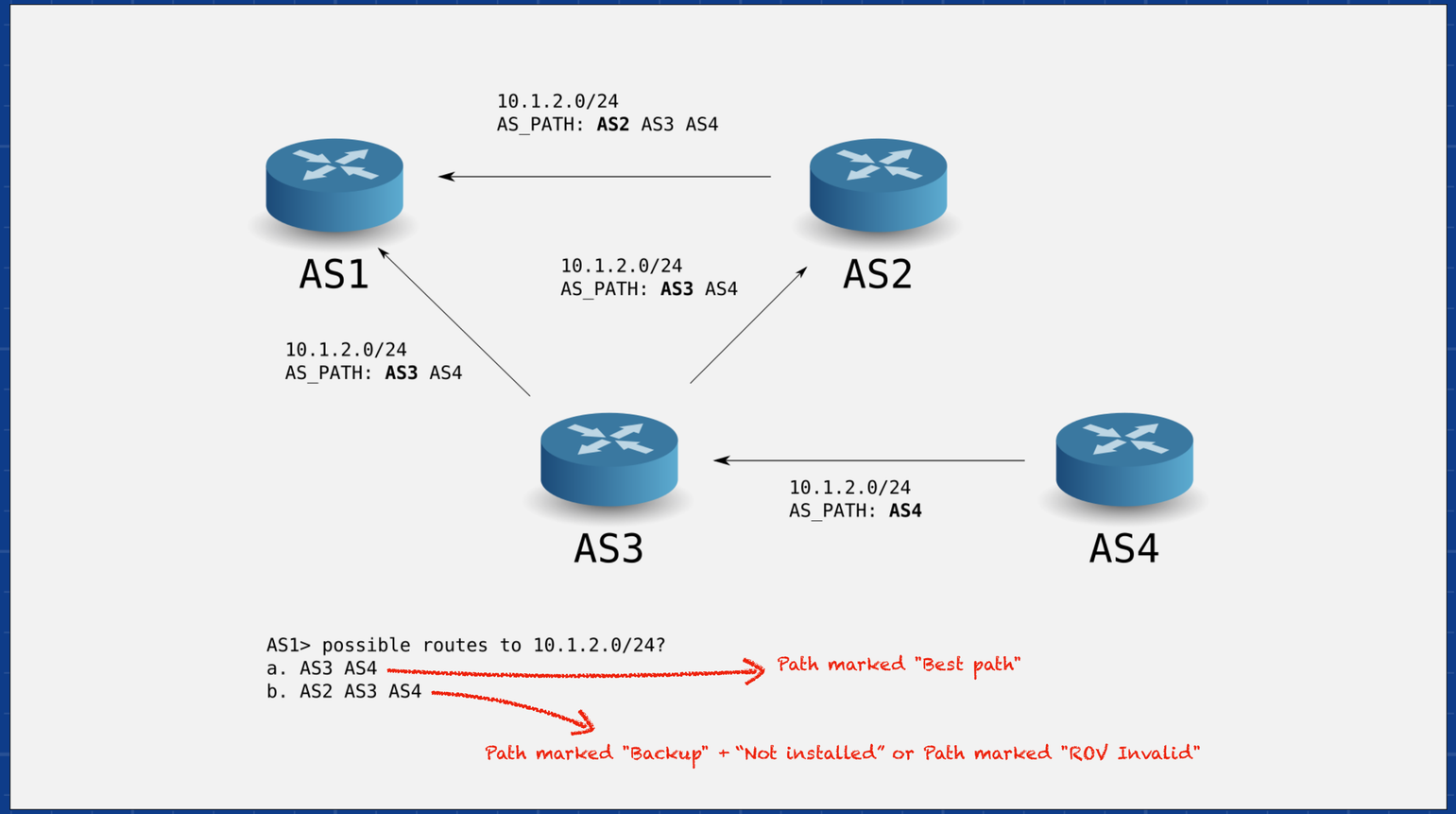
- Monitoring ~250 production routers from a Tier 1 provider, three different vendors, it works.
- Easy (re-)synchronisation of monitored RIBs without tears
- Re-use of BGP wire-formats speeds up development

# THE COULD-BE-NICER STUFF

- Not all parsing can be done stateless, information from PeerUpNotifications needed to process RouteMonitoring Messages. (inherited from BGP)
- EoR signalling is messy and incomplete, or even non-existent in some vendors
- Inconsistencies in (headers in) BMP messages, and textually in standards documents

# EVOLUTION OF BMP STANDARD DOCUMENTS




10.1.2.0/24
AS_PATH: AS2 AS3 AS4
AS1
AS2
10.1.2.0/24
AS_PATH: AS3 AS4
10.1.2.0/24
AS_PATH: AS3 AS4
AS3
10.1.2.0/24
AS_PATH: AS4
AS4
AS1> possible routes to 10.1.2.0/24?
a. AS3 AS4
b. AS2 AS3 AS4
Path marked "Best path"
Path marked "Backup" + "Not installed" or Path marked "ROV Invalid"

# THANK YOU