UNIVERSITEIT VAN AMSTERDAM

amsix
amsterdam internet exchange

BGP Parallelization

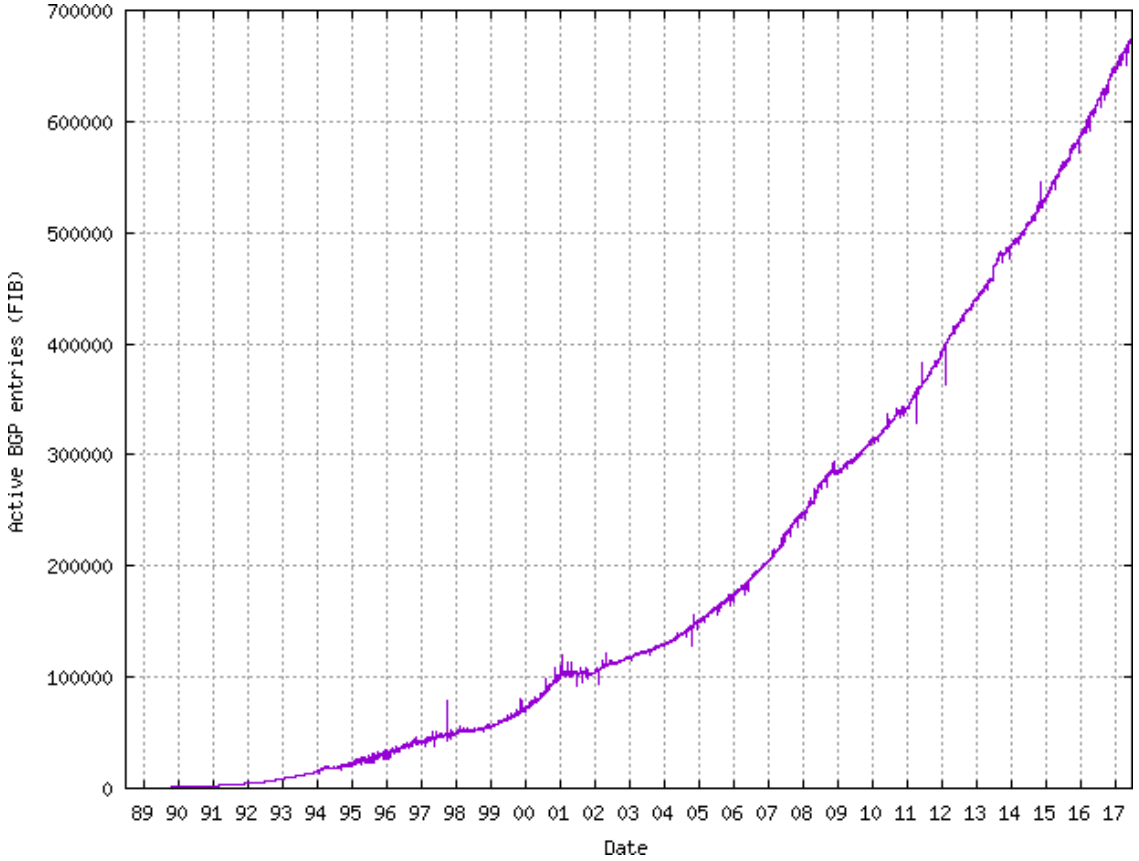
A study into the BGP protocol as well as BGP implementations to improve Route Server scalability.

Jenda Brands
Patrick de Niet

The internet is growing

Active BGP entries in FIB

From cidr-report.org



NETWORKS

More prefixes announced

- De-aggregation of prefixes
- Thus, more prefixes announced
- Currently 673,602 prefixes (03-07-17)

ROUTES

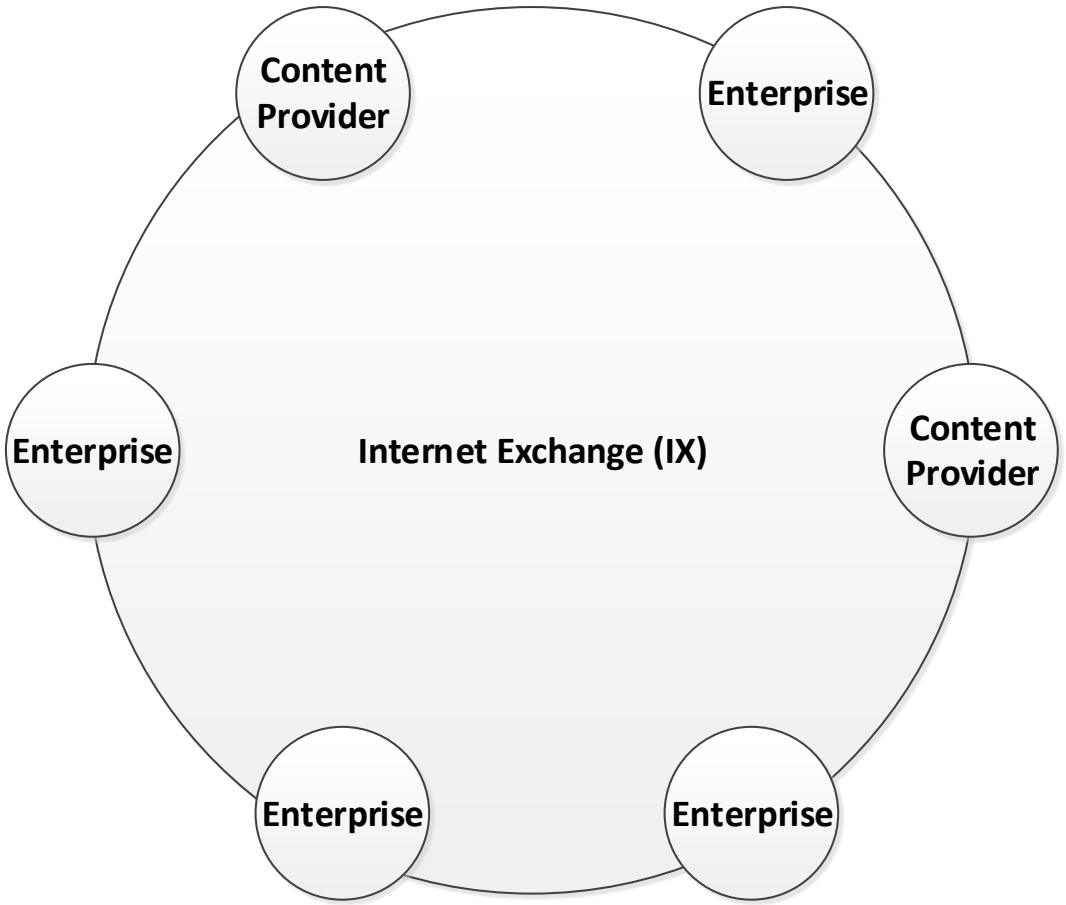
More routes to prefixes announced

- More interconnections are made

Introduction to internet exchanges

Internet Exchange

All nodes in the same layer-2 domain



IX

Benefits of IX

- Flat fee from IX
- Negotiate peering terms with neighbours

PEERING

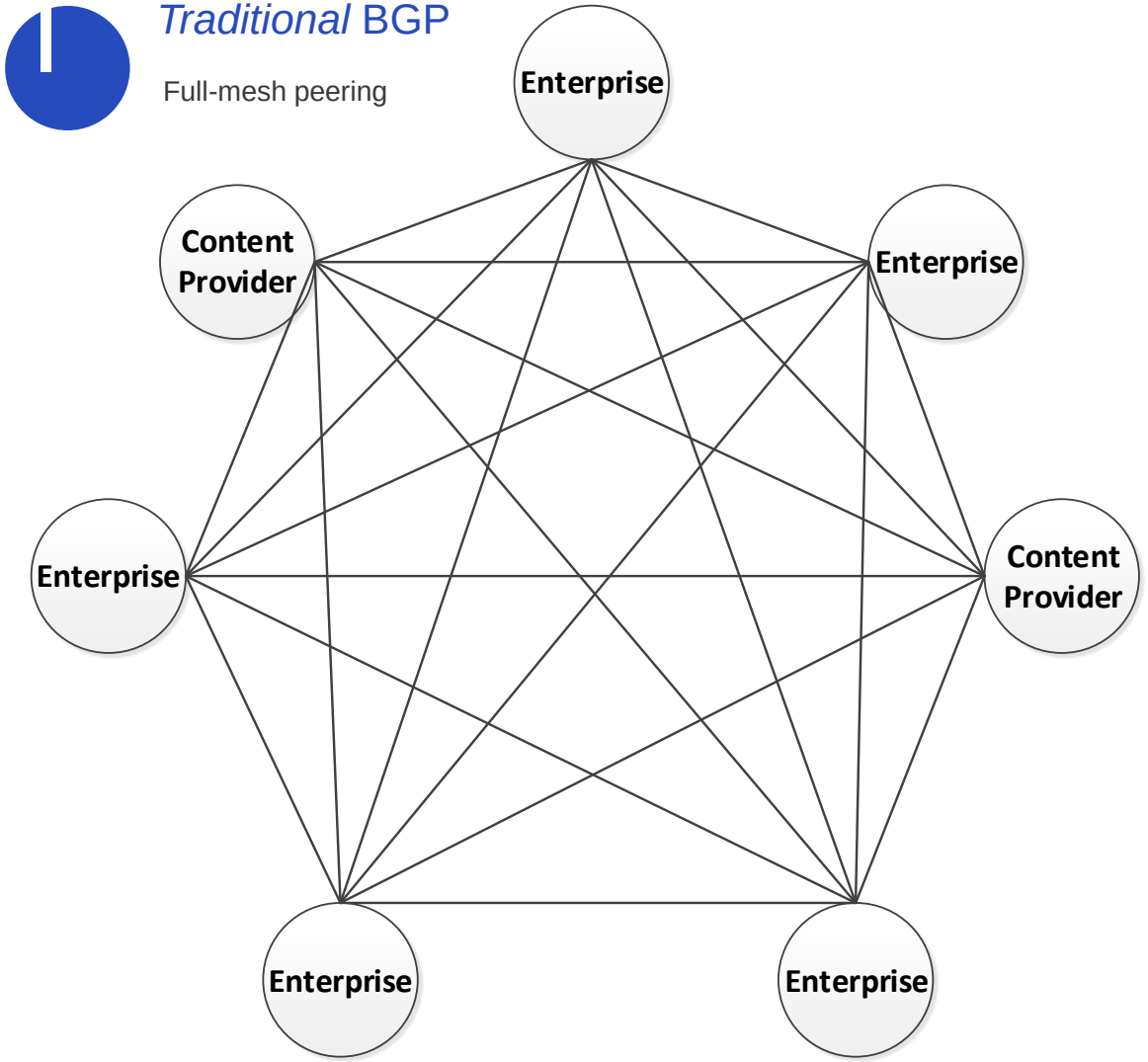
Costs of peering

- Internet Exchanges reduce peering costs and administration

Introduction to route servers

Traditional BGP

Full-mesh peering



BGP

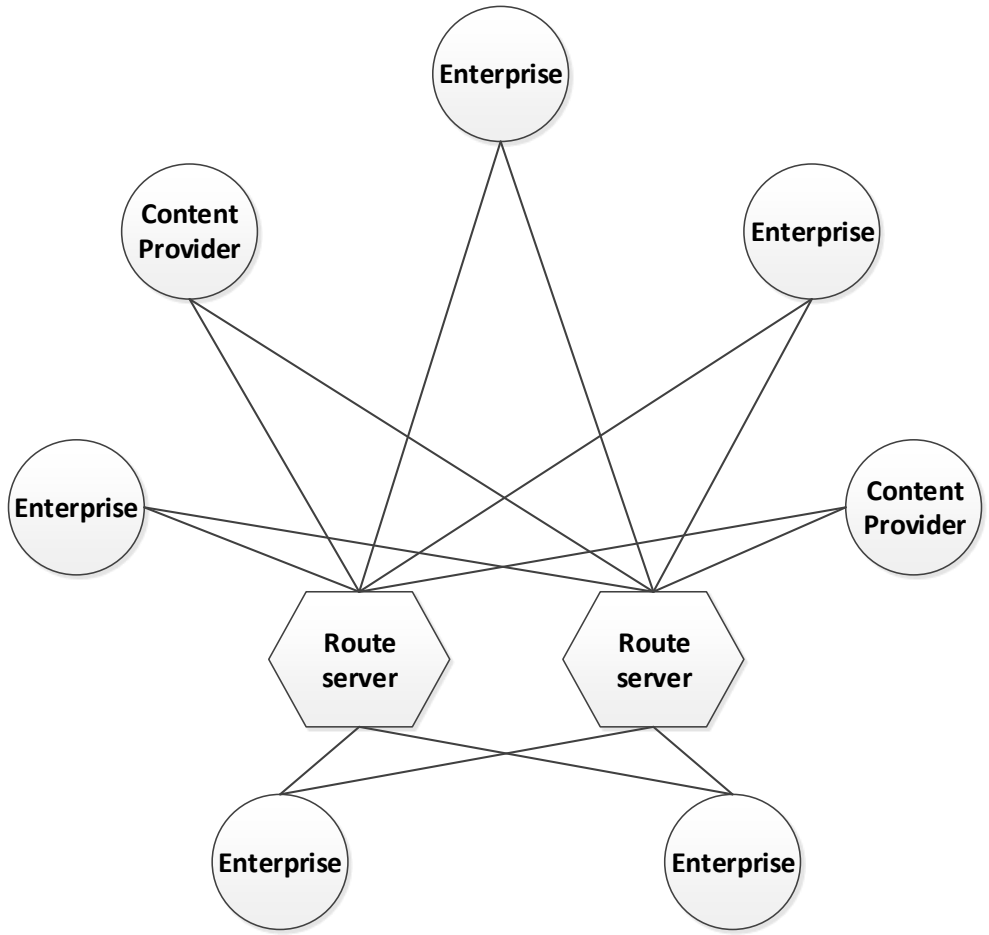
Without Route Server

- 21 Peerings in full-mesh required (*N(N-1)/2*)
- 6 sessions per node
- Same layer 2 network
- Lot of administration/configuration for all peers

Introduction to route servers

Current BGP

Peering with route server



BGP

With route server

- 14 Peerings required (*N2*)
- 2 sessions per node, each route server has 7
- Less administration/configuration needed for peering
- Private peering possible
- Route Server reduces load on clients

Problem

Convergence time

- Maximum CPU usage on route server
- Aged routes on the clients

Problem summary

Route servers are doing the heavy lifting and pushing BGP capabilities

As a result convergence times are increasing

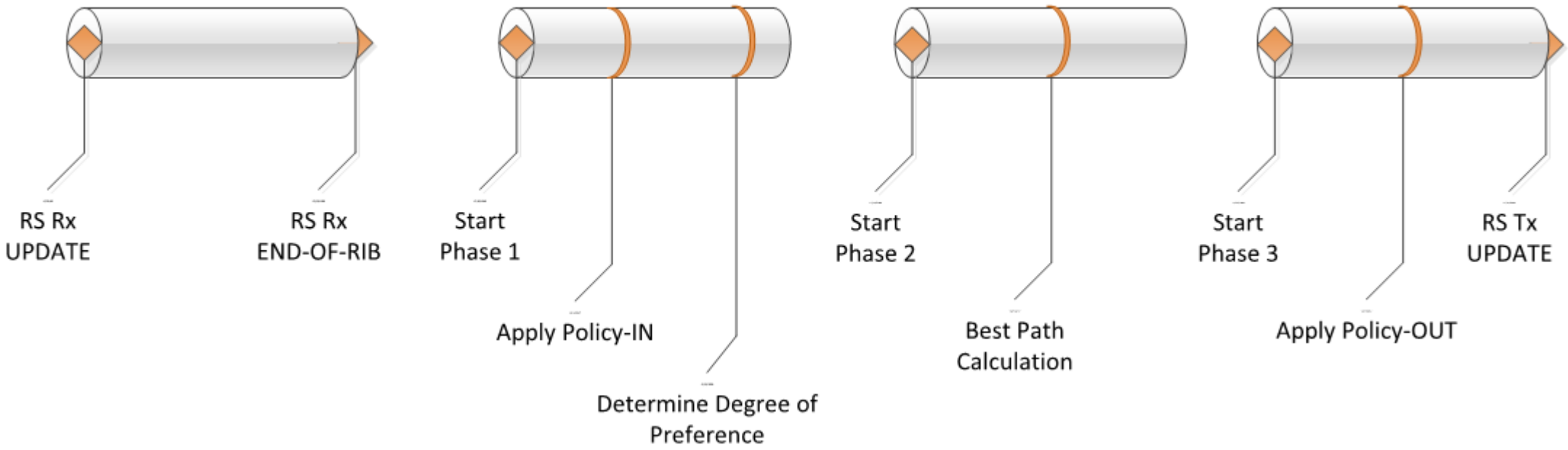
The exact cause of this behaviour within BGP is unidentified

Research question

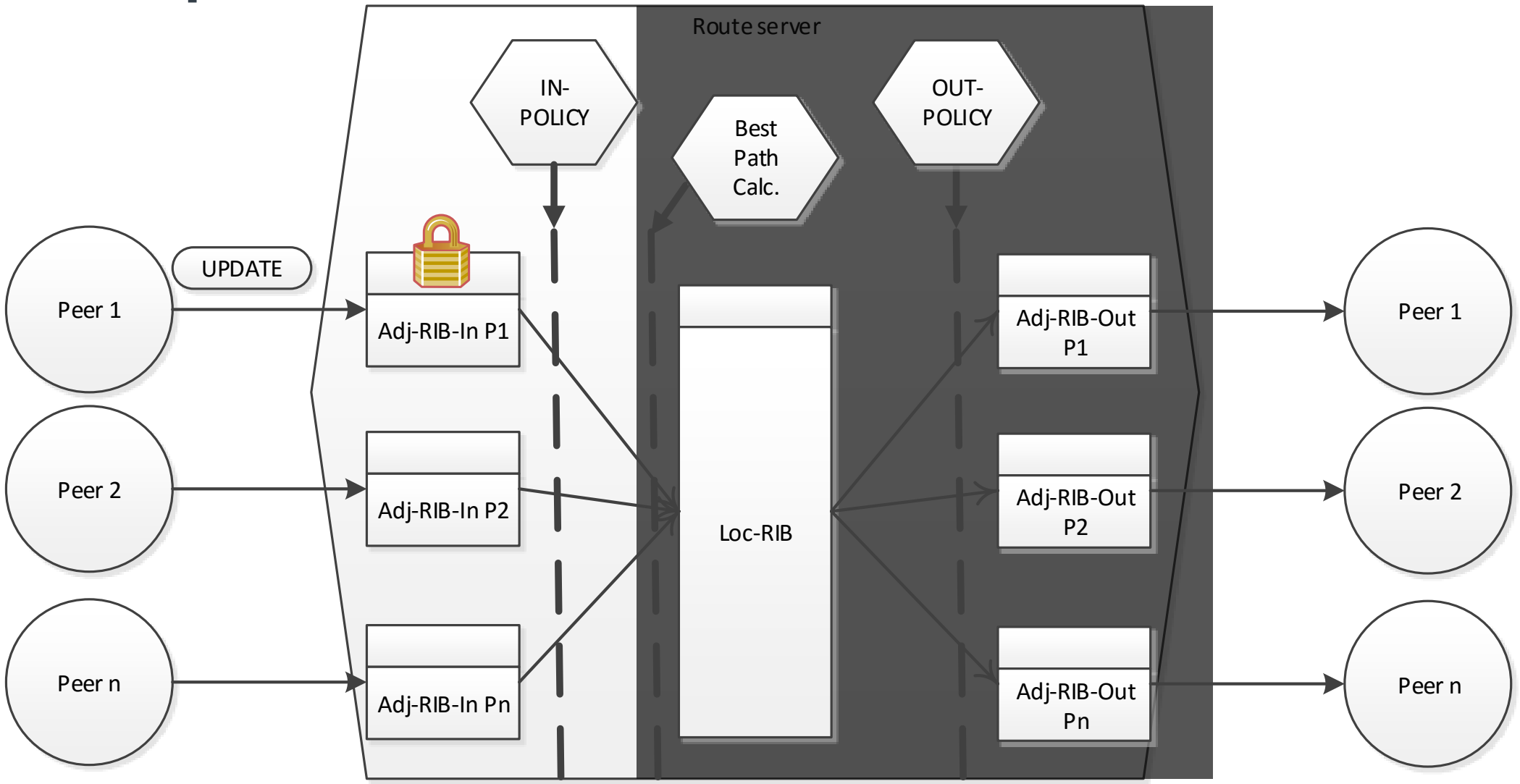
What **improvements** can be made to the **Border Gateway Protocol** (BGP) or its **implementations** to resolve current **CPU bottlenecks** when **processing updates**?

- Why are current BGP implementations (inherently) single-threaded?
- What past work has been done to solve this specific issue?
- What optimizations can be done to resolve this issue?

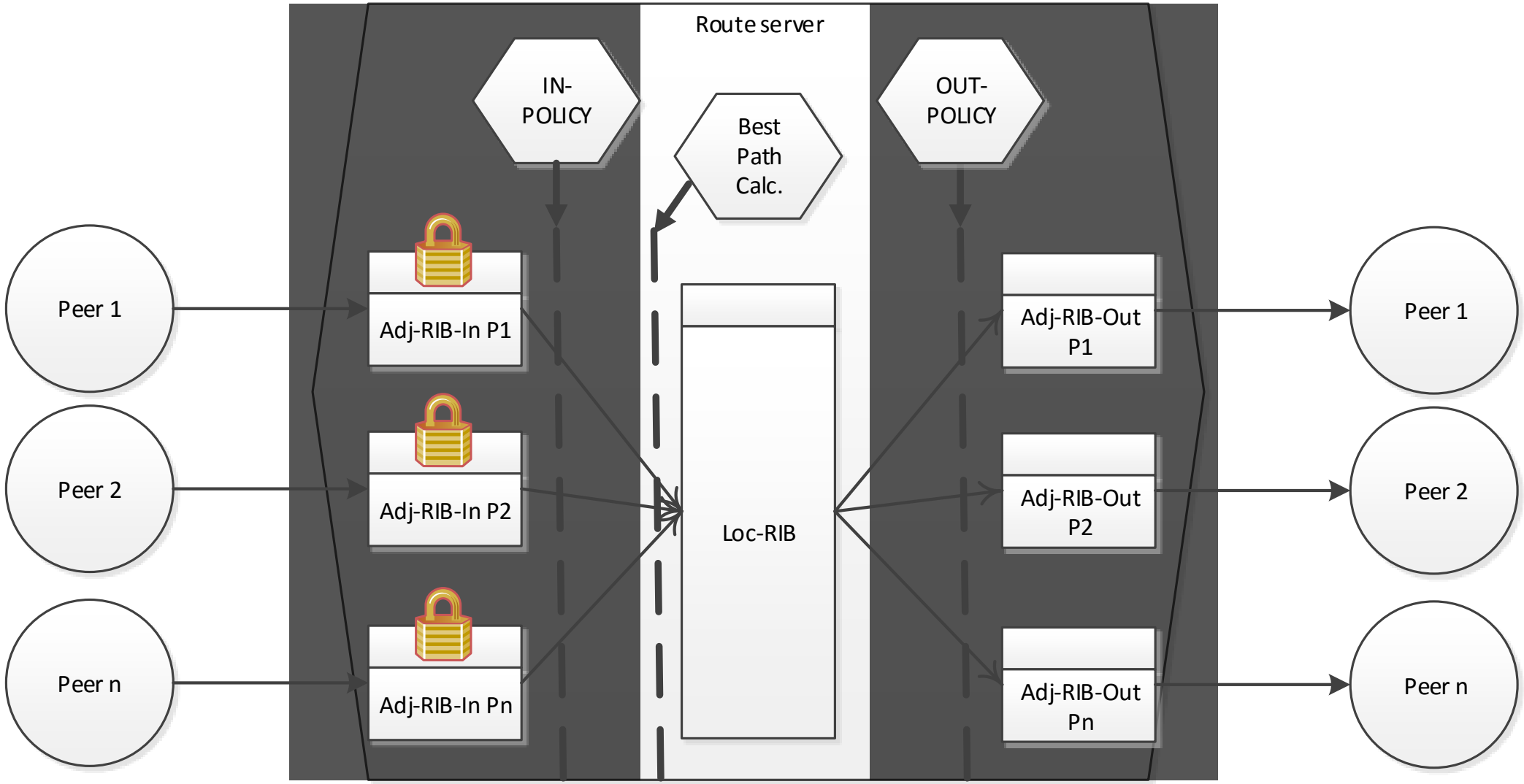
General BGP architecture



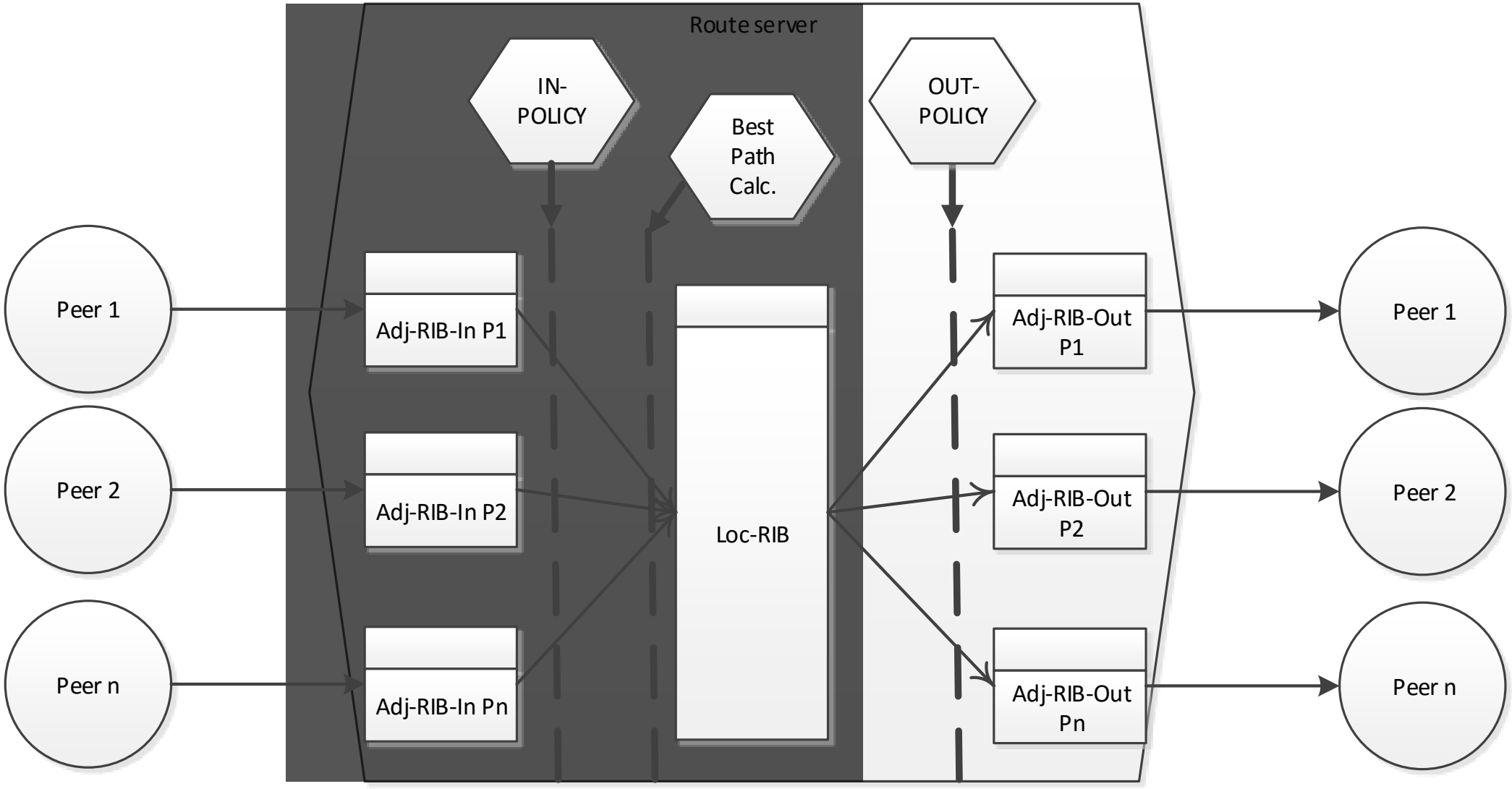
BGP specification (phase 1)



BGP specification (phase 2)



BGP specification (phase 3)

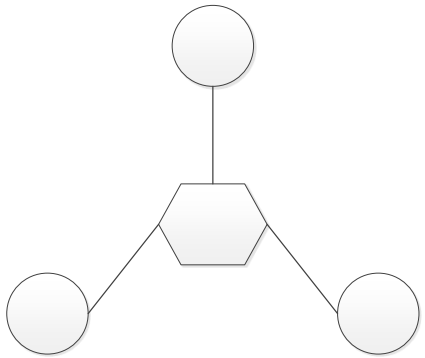


Testing scenarios

SCENARIO 1

THREE to ONE

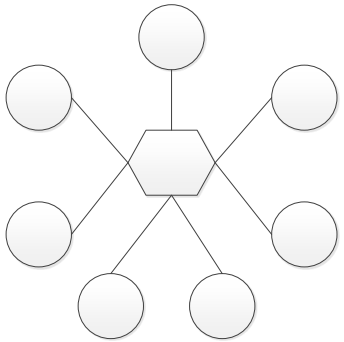
- Three peers
- One route server
- Simulate link-flap



SCENARIO 2

MANY to ONE

- Many peers
- One route server
- Simulate link-flap



SCENARIO 3

REAL WORLD

- Many peers
- One route server
- Overlapping prefixes
- Simulate link-flap

Testing scenarios

SAME

All peers **SAME** prefix

- Peer 1
  - 1.0.0.0/24
  - 1.0.1.0/24
  - 1.0.2.0/24
- Peer 2
  - 1.0.0.0/24
  - 1.0.1.0/24
  - 1.0.2.0/24
- Peer n
  - 1.0.0.0/24
  - 1.0.1.0/24
  - 1.0.2.0/24

UNIQUE

All peers **UNIQUE** prefix

- Peer 1
  - 1.0.0.0/24
  - 1.0.1.0/24
  - 1.0.2.0/24
- Peer 2
  - 1.0.3.0/24
  - 1.0.4.0/24
  - 1.0.5.0/24
- Peer n
  - 1.0.6.0/24
  - 1.0.7.0/24
  - 1.0.8.0/24

REAL-WORLD

REAL WORLD

- Peer 1
  - 1.0.0.0/20
  - 1.0.16.0/20
  - 1.0.32.0/20
- Peer 2
  - 1.0.4.0/23
  - 1.0.6.0/23
  - 1.0.8.0/23
- Peer n
  - 1.0.5.0/24
  - 1.0.7.0/24
  - 1.0.8.0/24

Testbed

ROUTE SERVER

ONE route server

- Intel(R) Xeon(R) CPU E3-1220L V2 @ 2.30GHz (4 cores)
- 7.7GB RAM
- BIRD BGP daemon

PEER SERVERS

EIGHT servers for peers

- Intel(R) Xeon(R) CPU L3426 @ 1.87GHz (8 cores)
- 7.7GB RAM
- Docker used for containers

PEERS

800 peers max

- ExaBGP daemons

Definitions

CONVERGENCE

What defines **CONVERGED** state

- Either
  - Got END-OF-RIB for last peer
  - Stops sending UPDATES

LINK FLAP

All peers **UNIQUE** prefix

- Simulate flapping link
  - Bring link to RS down

METRICS

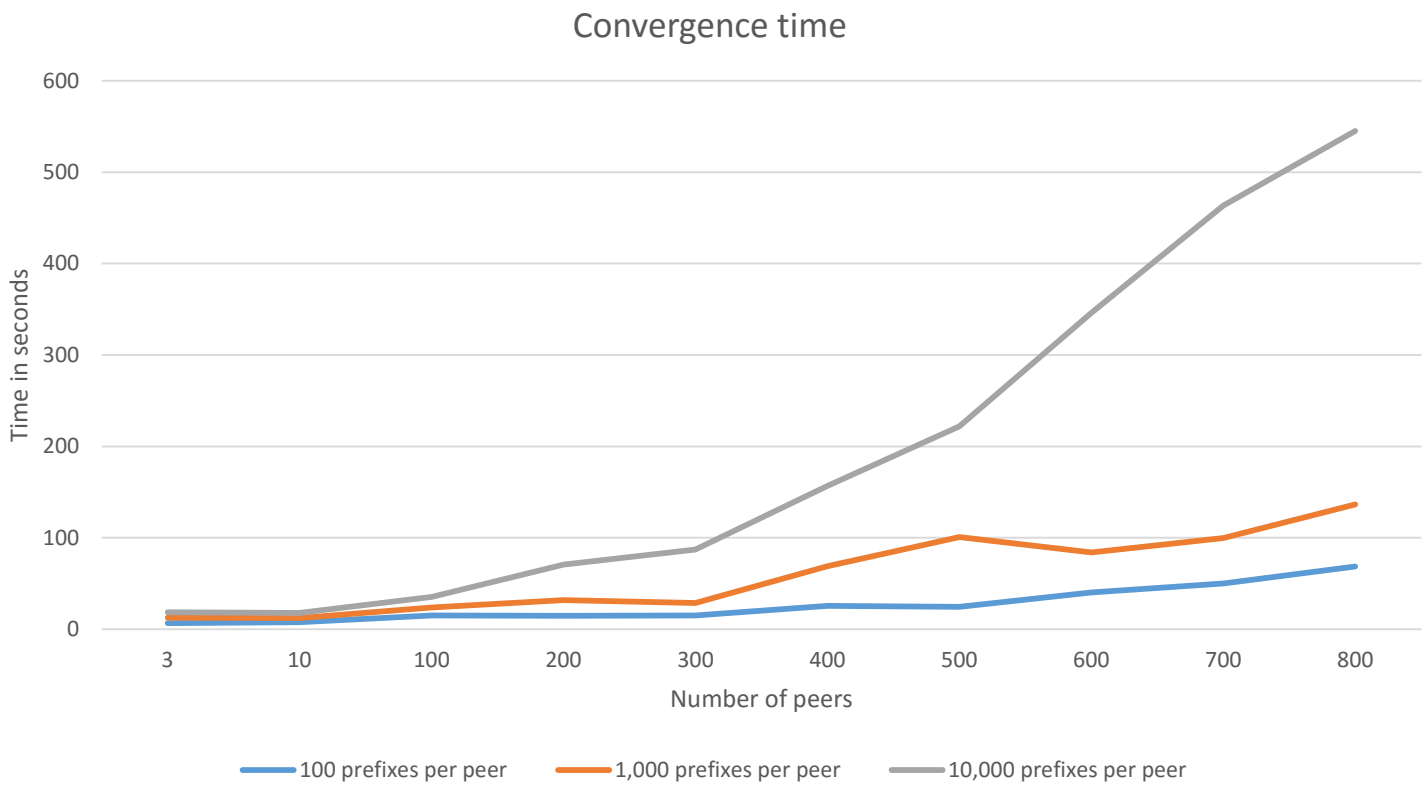
What was **MEASURED**

- CPU Utilization
- Memory Utilization
- Bandwidth

Observations

Convergence time

Convergence time vs number of peers



RESULTS

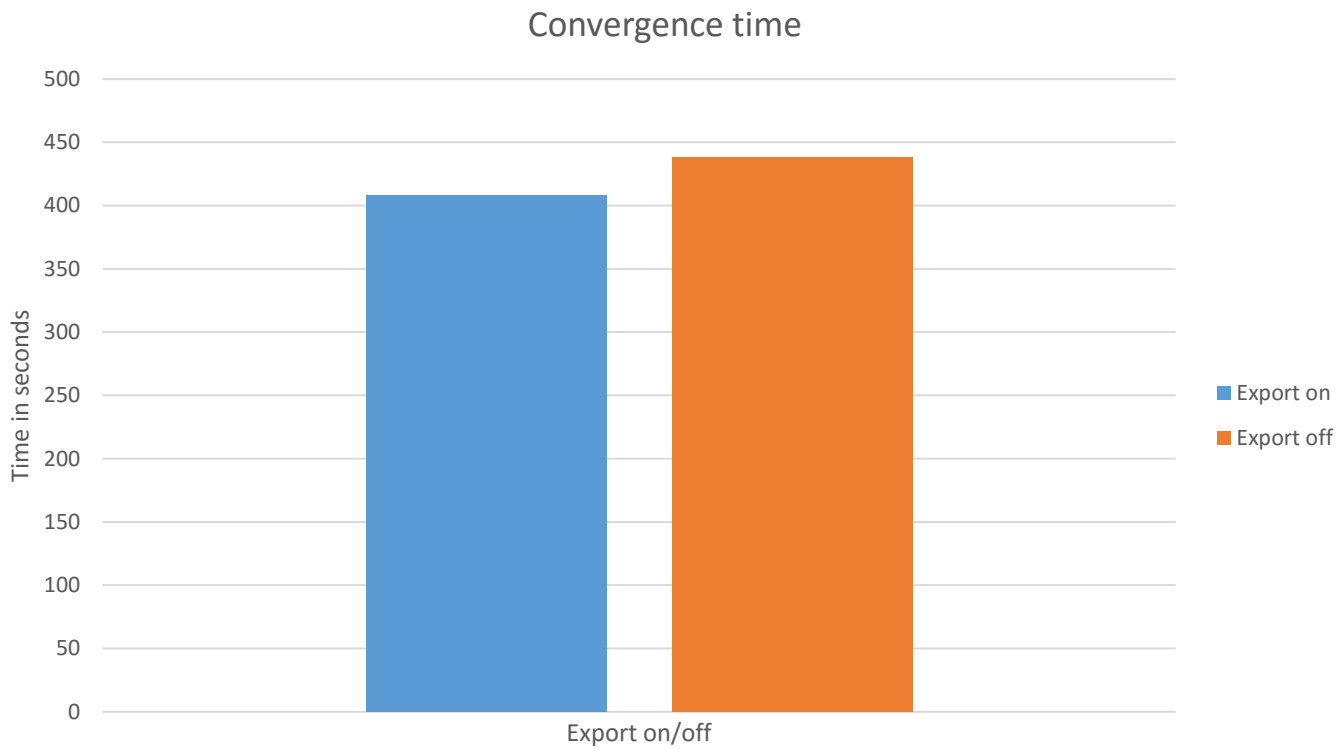
Convergence times

- Lower numbers show lower convergence times
- Higher numbers show increasingly higher times

- 10,000 prefixes with 800 peers significantly higher

Observations

Turning off export of routes



NO EXPORT

Phase 3

- Sending UPDATES disabled
- “export none”

- No significant difference
- Phase 3 (sending UPDATES) can not be the issue

- Unable to conclusively rule out remaining phases

Solutions

PROTOCOL

PROTOCOL improvements

- Snapshot of Adj-RIB-In
- Sorted on prefix

- Calculate hash on peer side
- With OPEN message send hash
- RS compares hash
- If hash is the same no need for full UPDATE

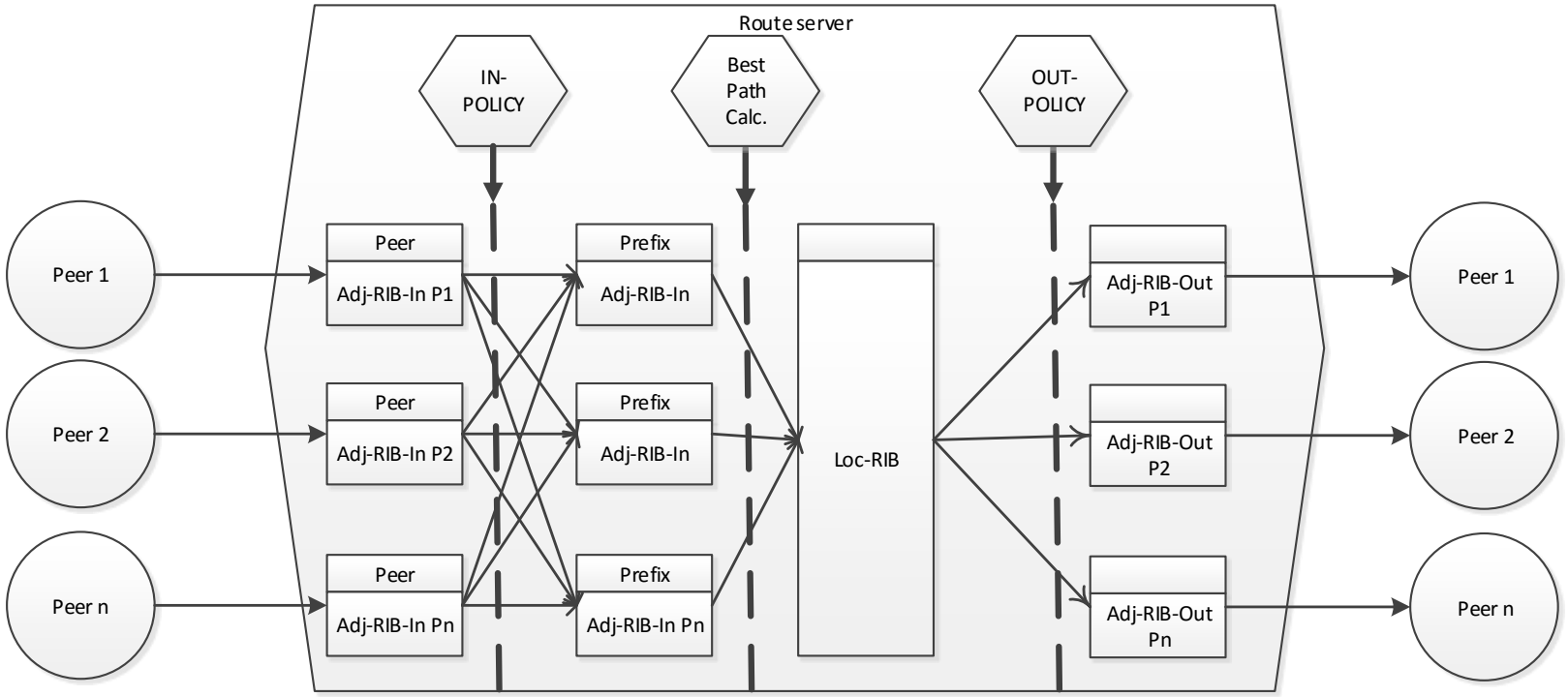
IMPLEMENTATION

IMPLEMENTATION improvements

- Load balance route servers
- Single endpoint for customers
- iBGP for internal convergence
- eBGP for peering

Protocol solution

Protocol modifications



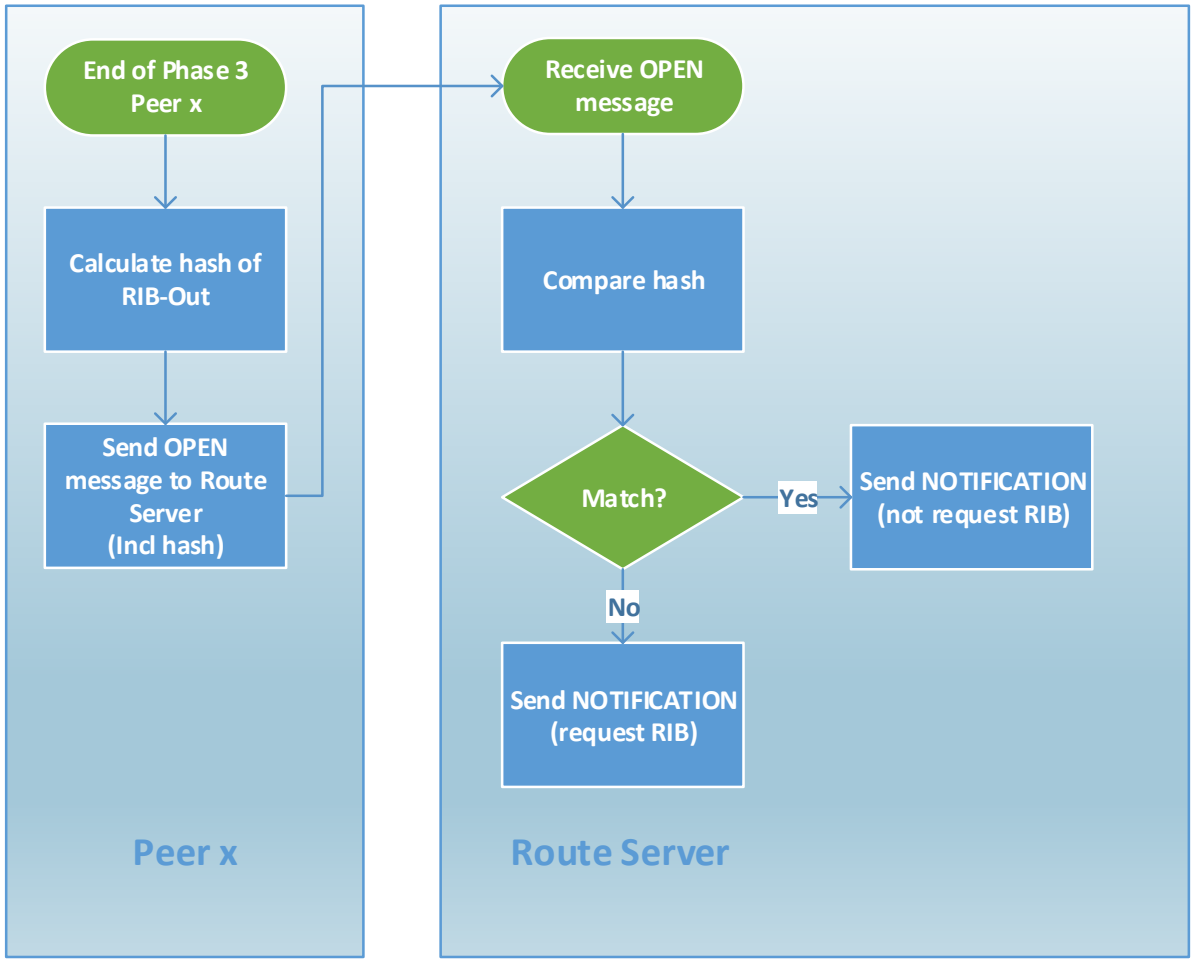
PREFIX BASED

Create prefix based RIB-In

- Create table per prefix
- Add all paths to that prefix
- When starting Phase 2 calculation *only* lock that specific RIB

Protocol solution

Protocol modifications



HASHING

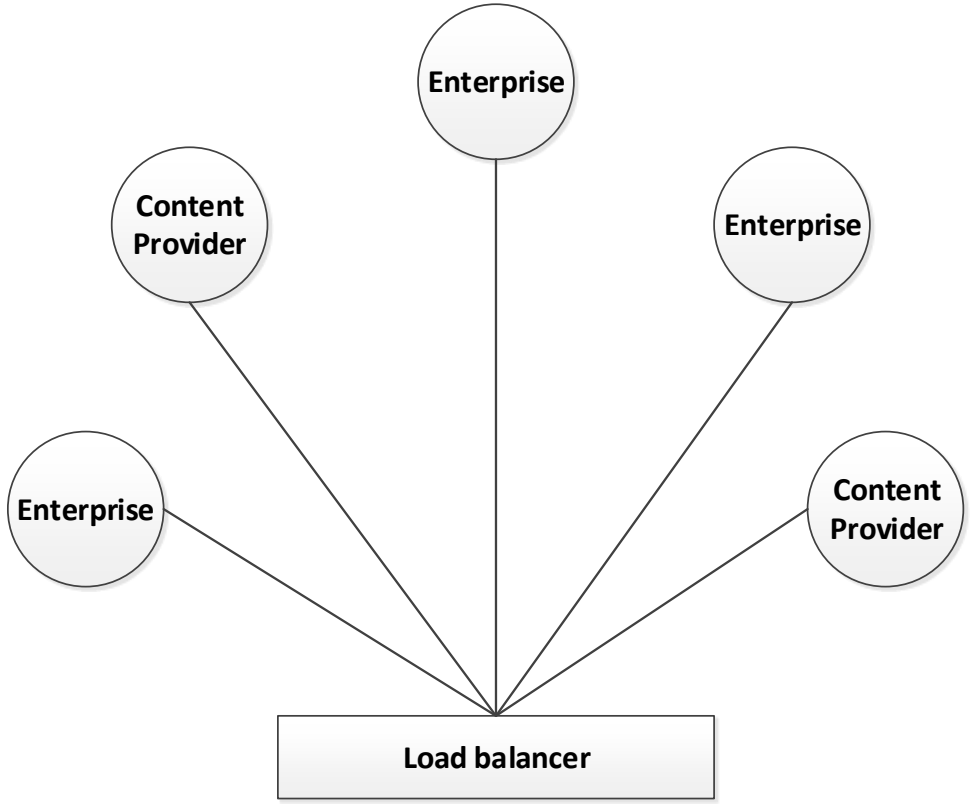
Compare hash before full UPDATE

- Calculate hash of RIB on peer-side
- After link-flap send hash in OPEN message
- RS compares hashes, if match, no need for full UPDATE

Implementation solution

Load balancing

Customers do peering with load-balancer



BEFORE LB

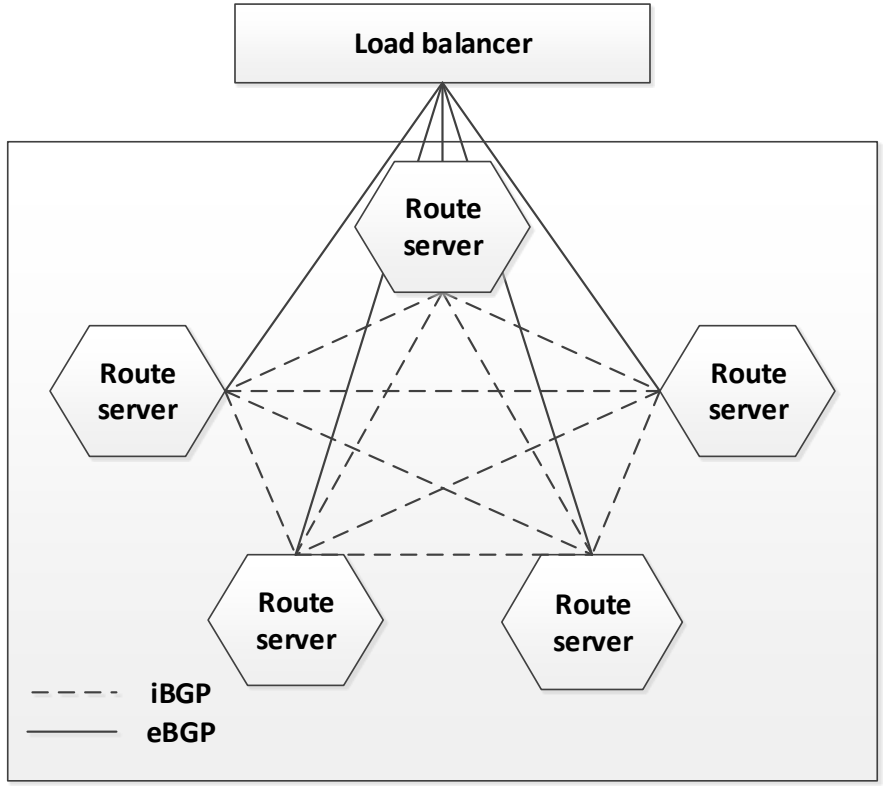
eBGP

- Customers peer through load balancer
- Peer with route server behind load balancer

Implementation solution

Load balancing

Load-balancer balances between route servers



BEHIND LB

iBGP

- iBGP full mesh
- eBGP to load-balancer

Future work

1

Rule out phase 1

Narrow down the problem as much as possible

Good chances phase 1 is also not the issue

2

PoC of hashing mechanism

Set up a proof of concept of the proposed hashing mechanism

1

Benchmarking of code

Go through (open-source) code

Put timestamps, find delaying pieces of code

Narrow down bottleneck

2

PoC of load balancing

Set up a proof of concept with load balancing

Measure convergence time gain

Find any caveats not identified yet

UNIVERSITEIT VAN AMSTERDAM

amsix
amsterdam internet exchange

THANK YOU

Any further questions?

UNIVERSITEIT VAN AMSTERDAM

amsix
amsterdam internet exchange

THANK YOU

Any further questions?

Let’s have a beer