centralized control —opportunities and challenges

5590: software defined networking

anduo wang, Temple University

TTLMAN 402, R 17:30-20:00

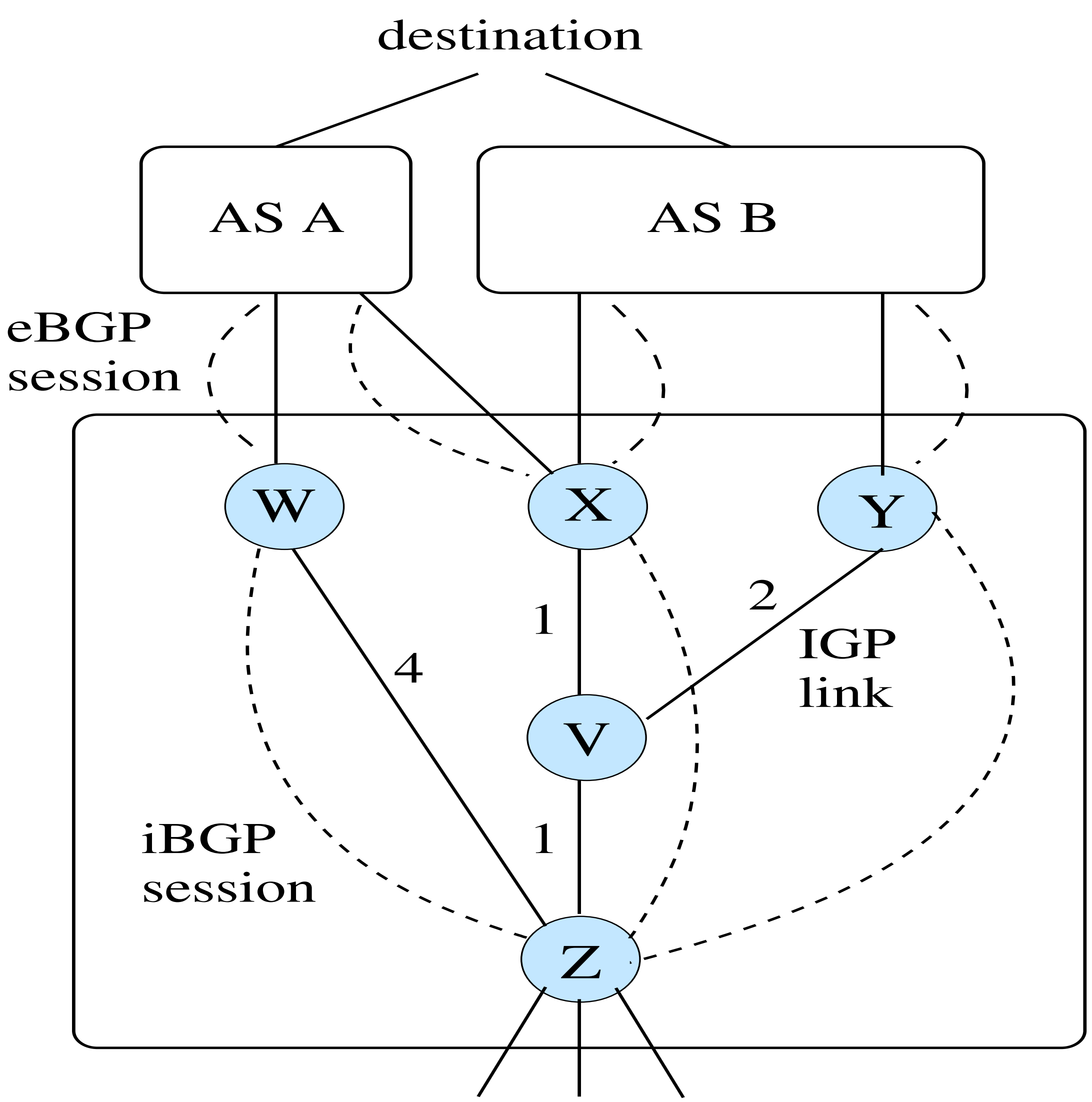
some materials in this slide are based on lectures by

Jennifer Rexford https://www.cs.princeton.edu/courses/archive/fall13/cos597E/

Nick Feamster http://noise.gatech.edu/classes/cs8803sdn/fall2014/

RCP

BGP background



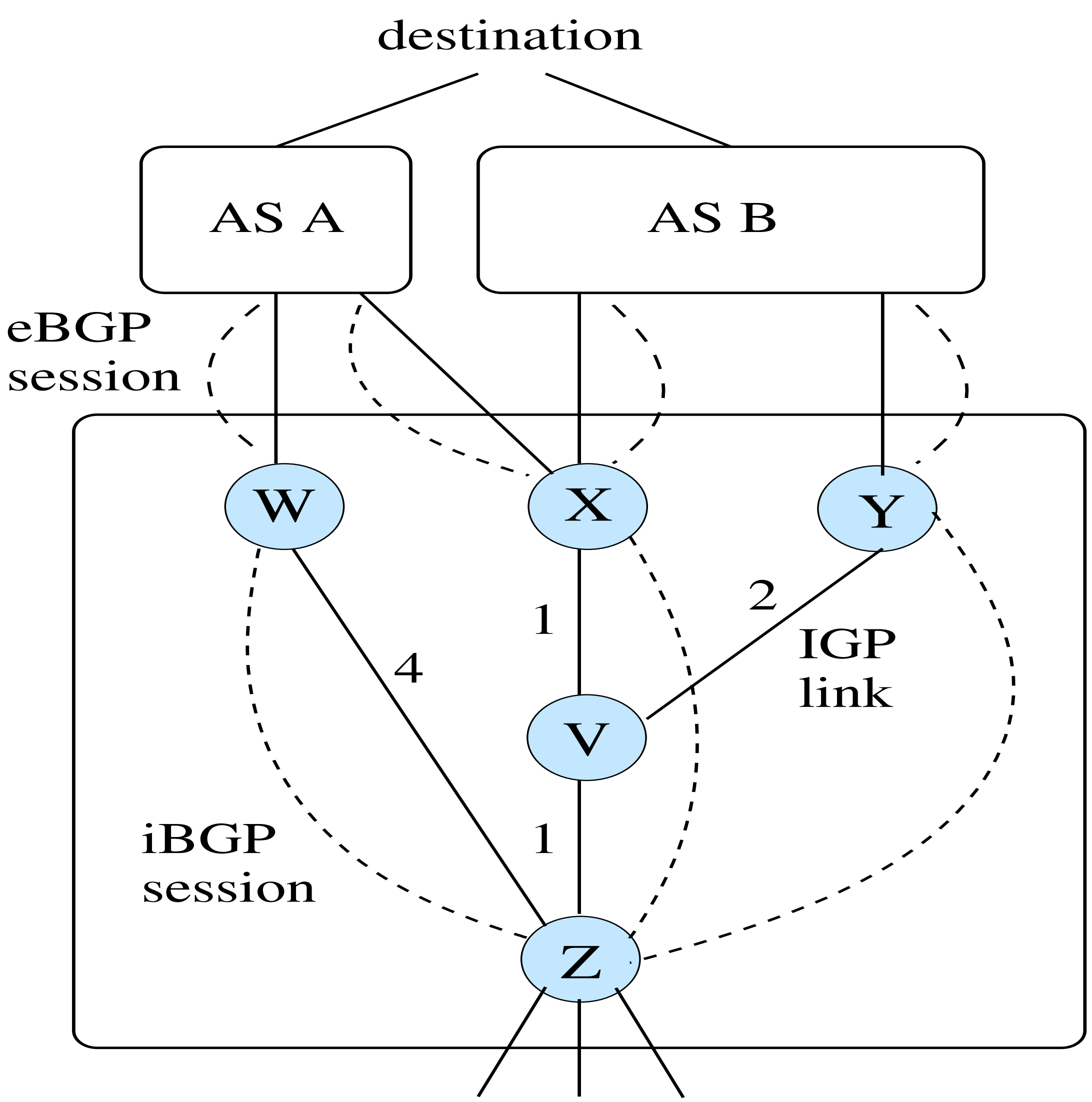
BGP

- de-facto inter-domain (inter-AS) routing protocol

functionality partitioned across routing protocols

- eBGP
- iBGP
- IGP

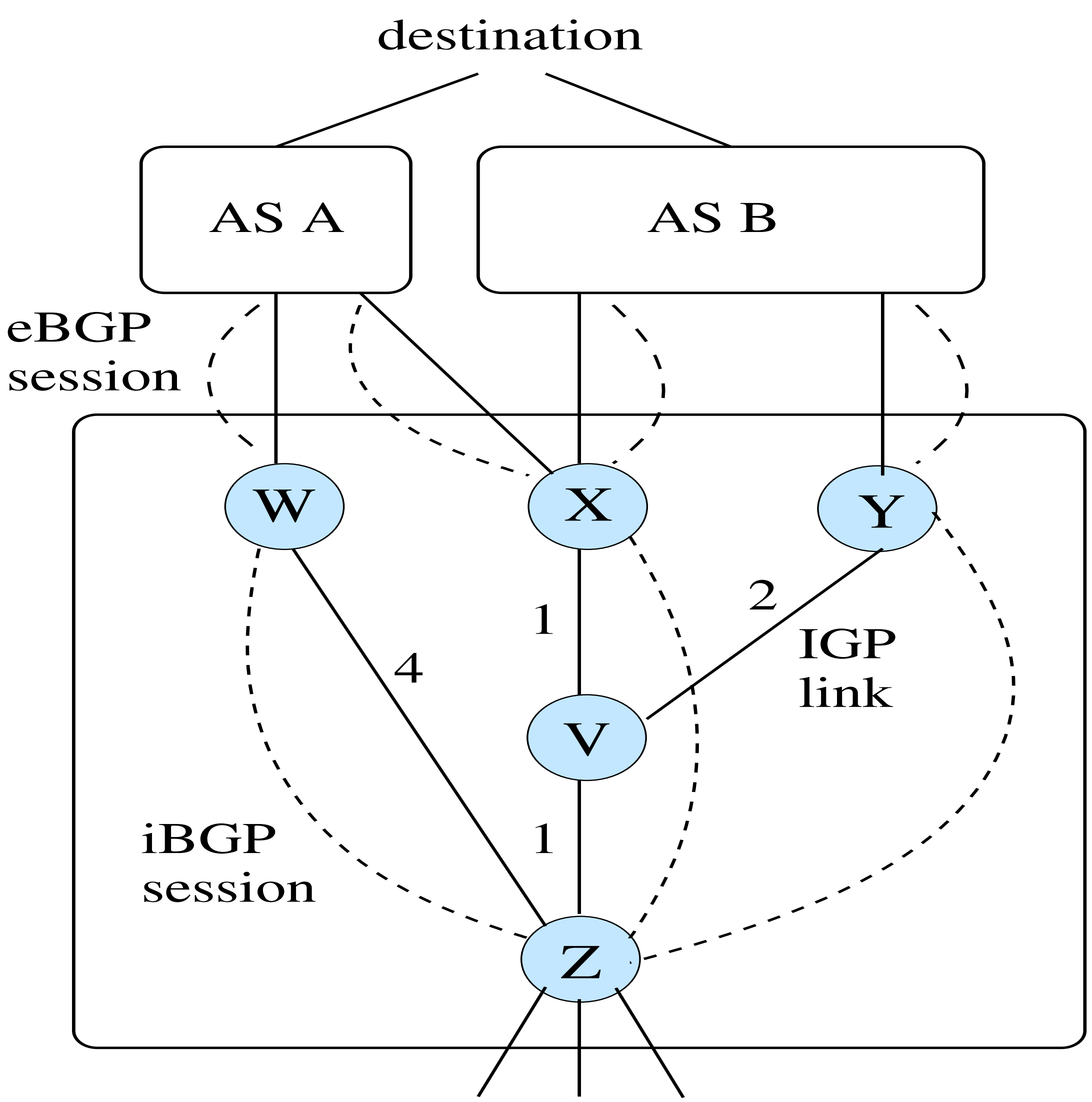
BGP background



BGP route-selection

1. highest local preference
2. lowest AS path length
3. lowest origin type
4. lowest MED (with next hop)
5. eBGP-learned over iBGP-learned
6. lowest path cost to egress
7. lower router ID

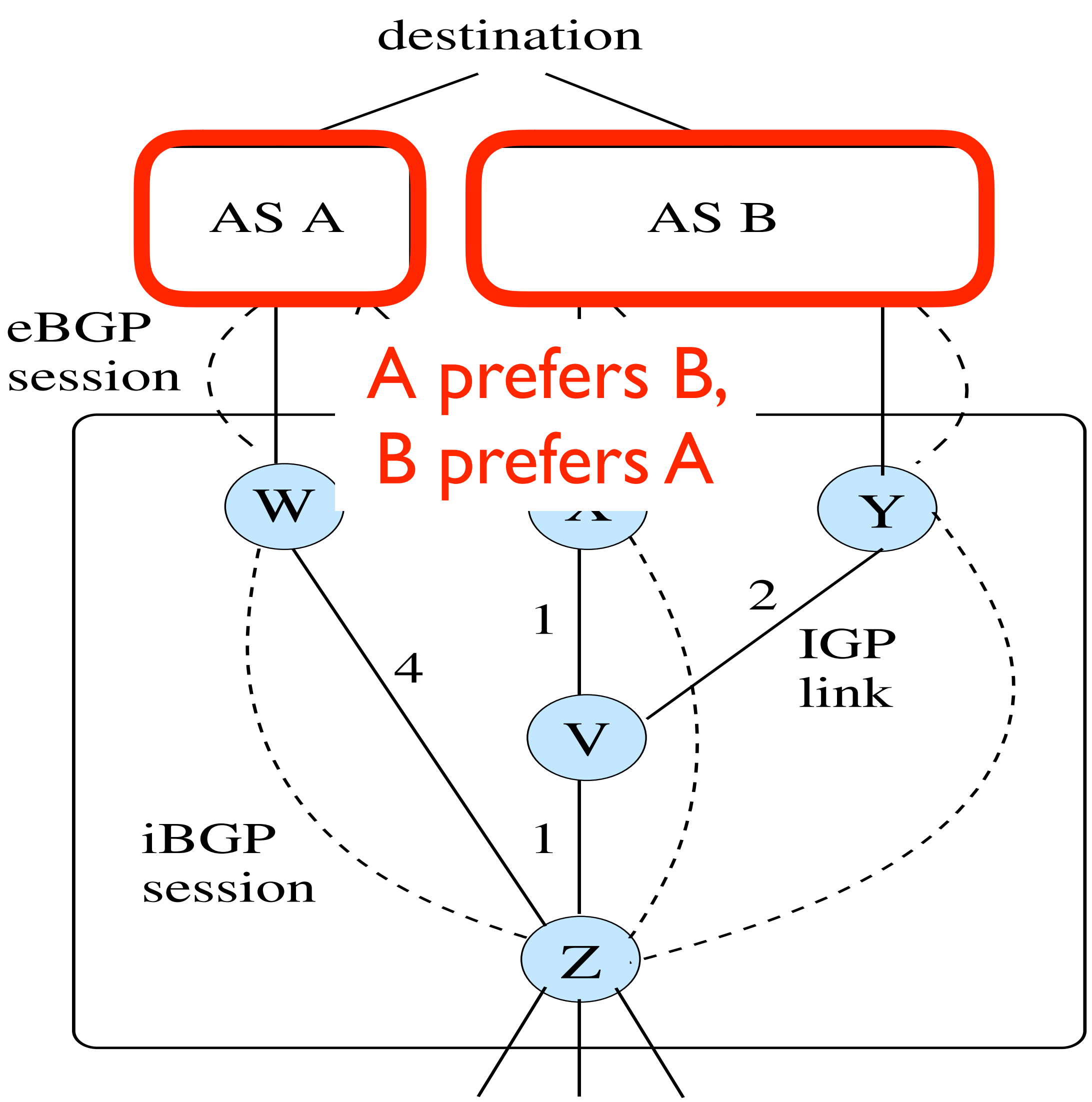
BGP: shortest path routing



BGP route-selection

1. highest local preference
2. lowest AS path length
3. lowest origin type
4. lowest MED (with next hop)
5. eBGP-learned over iBGP-learned
6. lowest path cost to egress
7. lower router ID

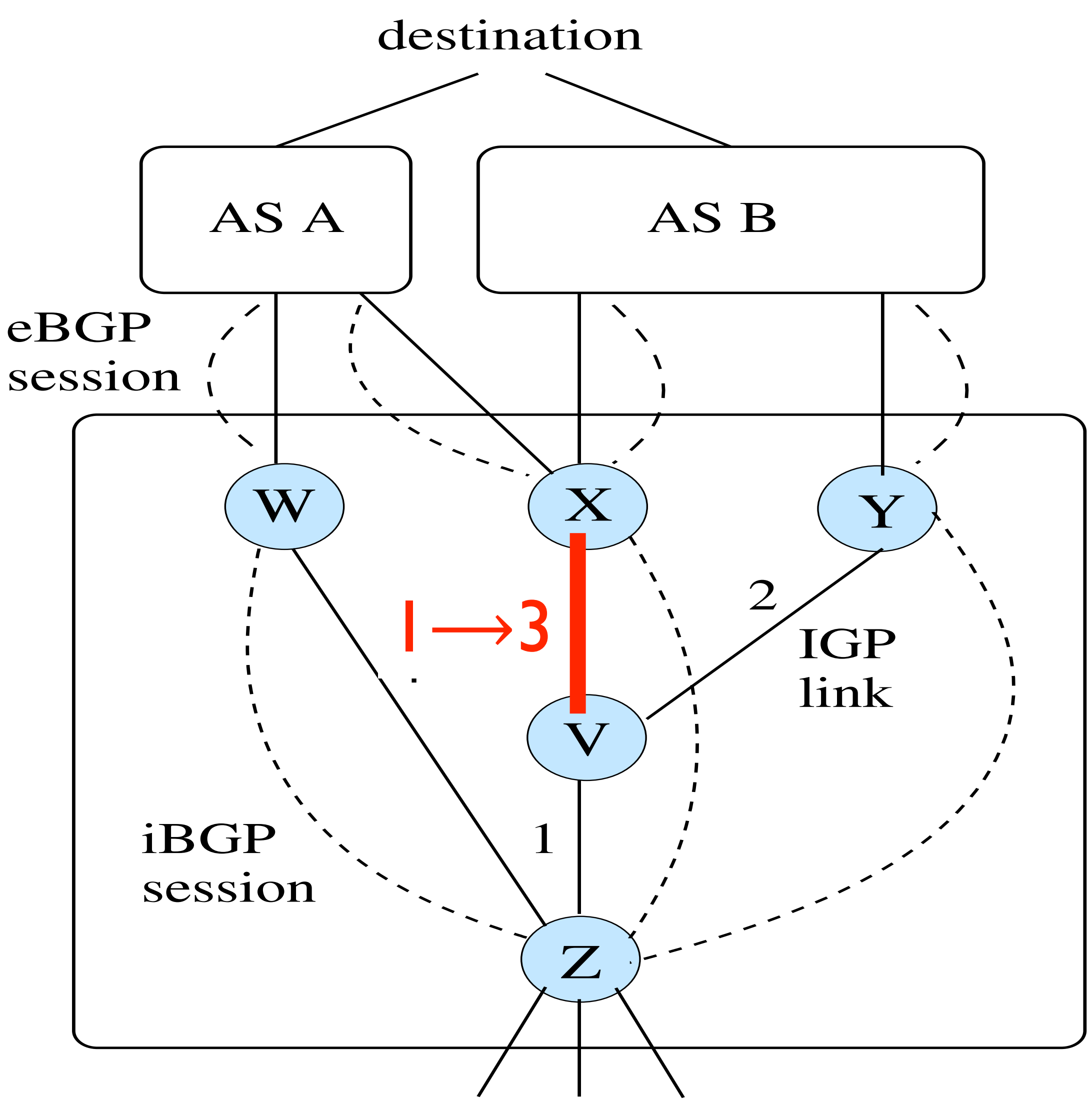
BGP problem: oscillation



BGP route-selection

1. highest local preference
2. lowest AS path length
3. lowest origin type
4. lowest MED (with next hop)
5. eBGP-learned over iBGP-learned
6. lowest path cost to egress
7. lower router ID

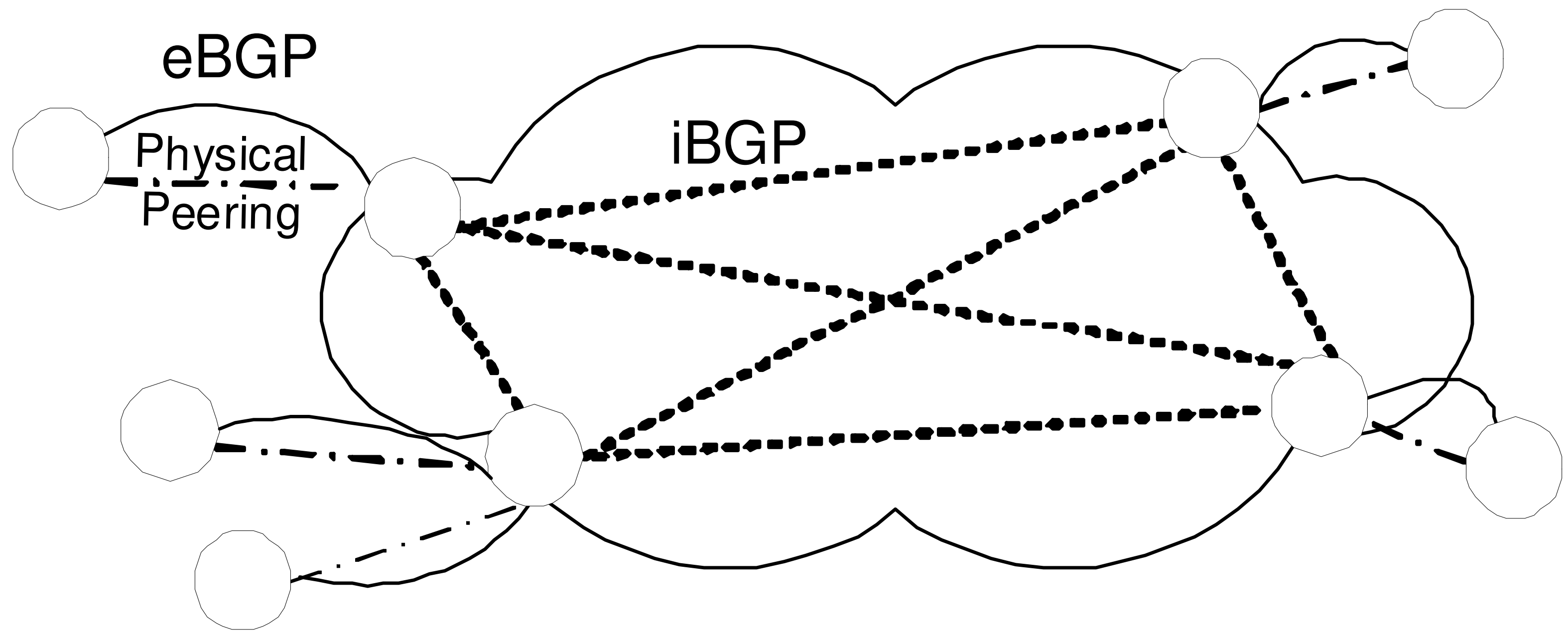
BGP problem: hot-potato

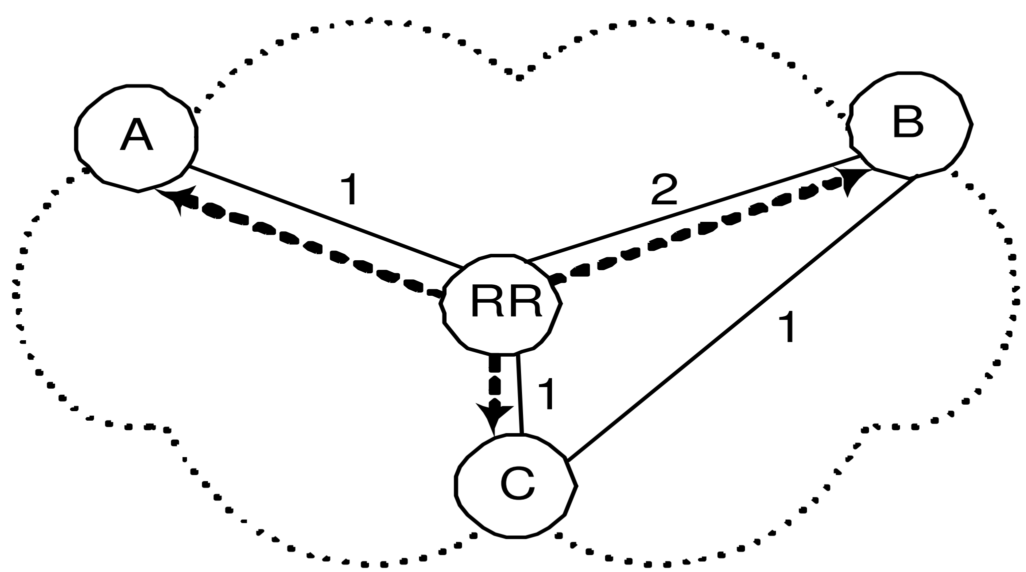


BGP route-selection

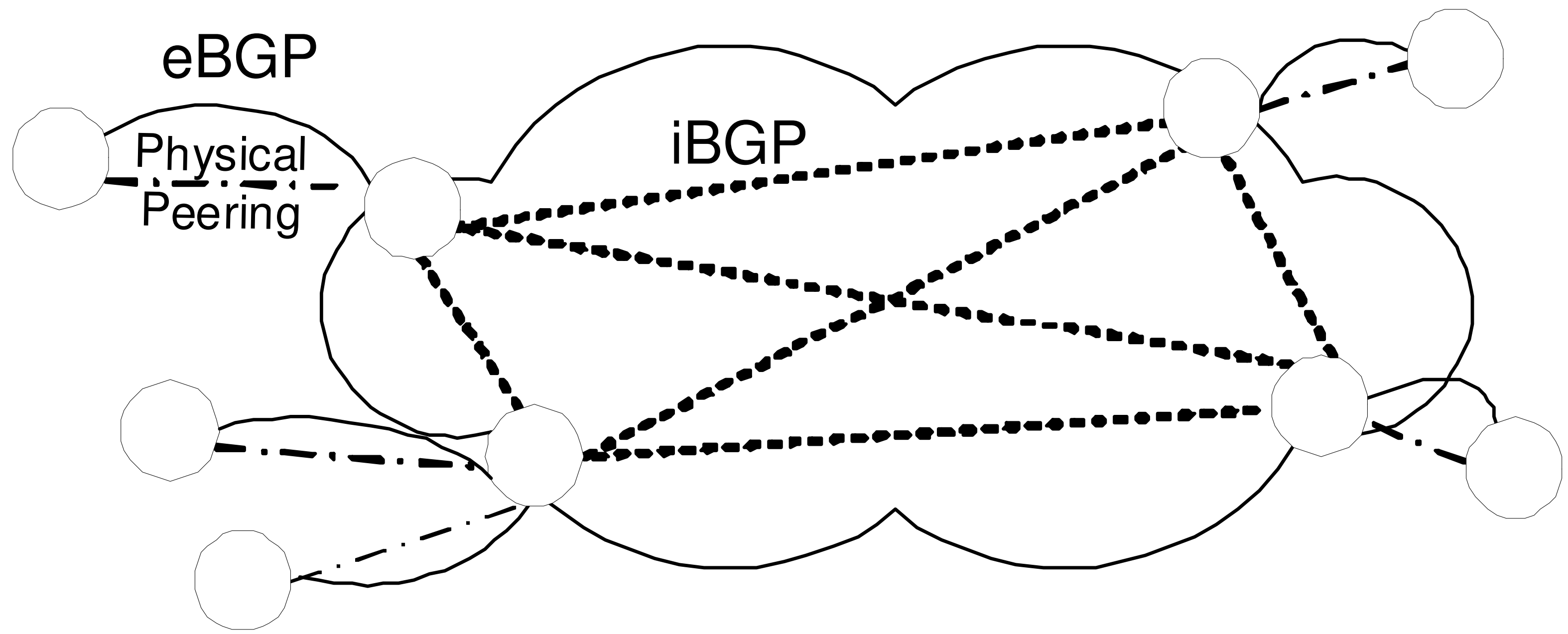
1. highest local preference
2. lowest AS path length
3. lowest origin type
4. lowest MED (with next hop)
5. eBGP-learned over iBGP-learned
6. lowest path cost to egress (hot-potato, early-exit)
7. lower router ID

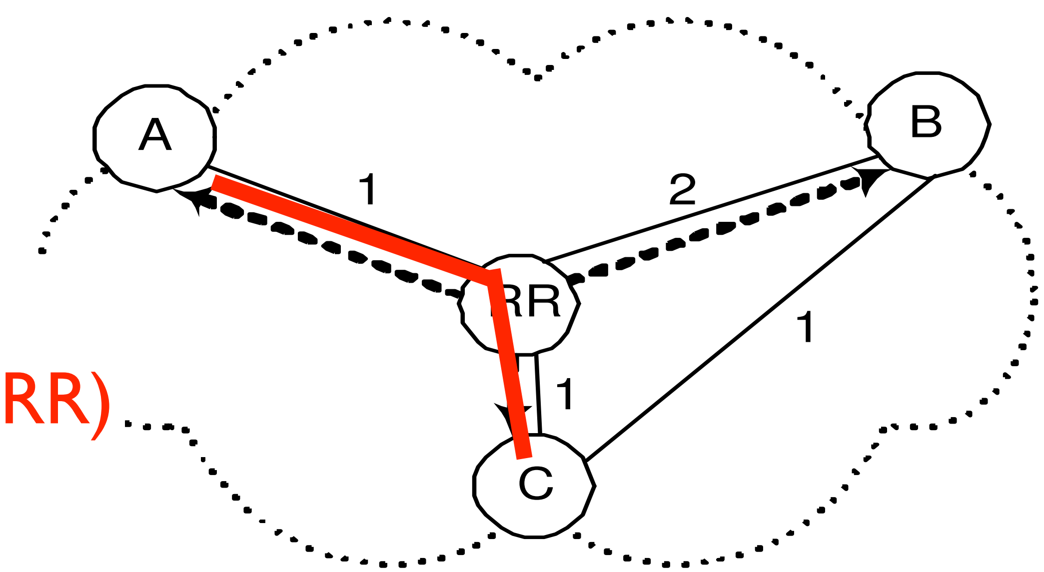
BGP problem: RR ≠ full-mesh



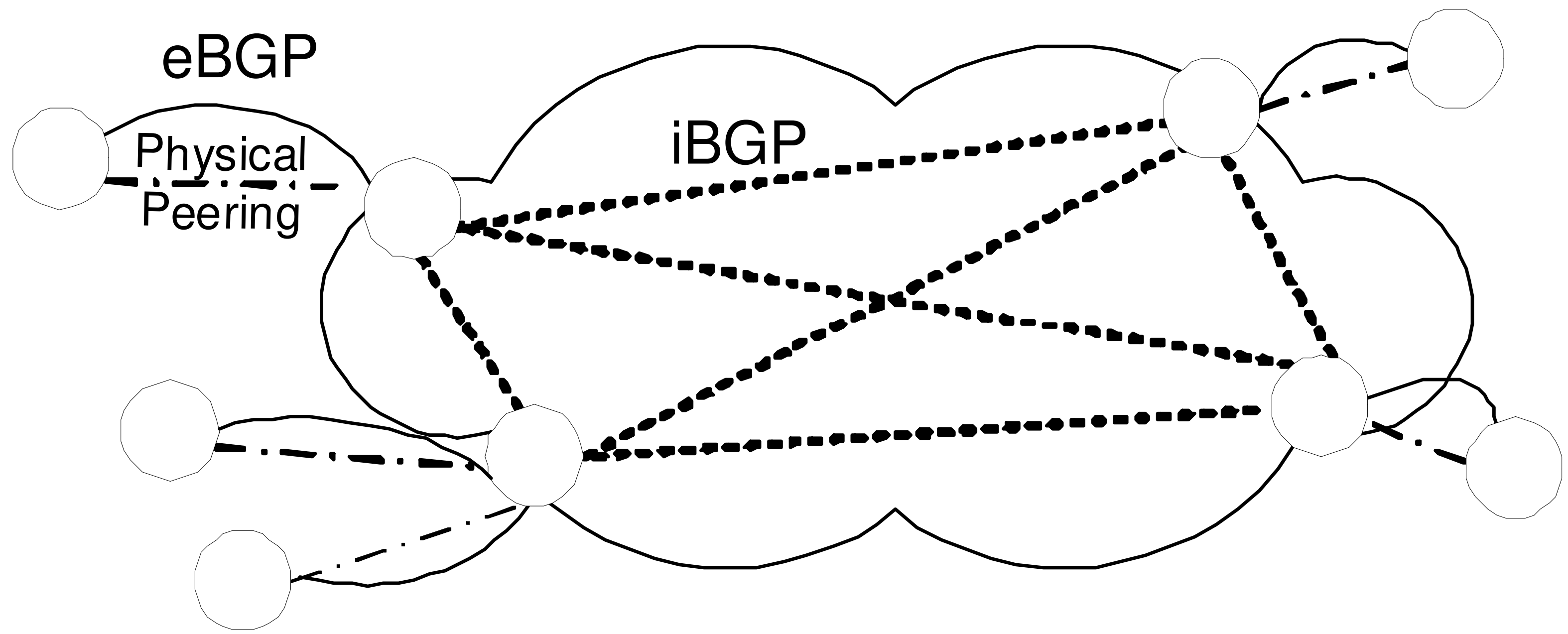


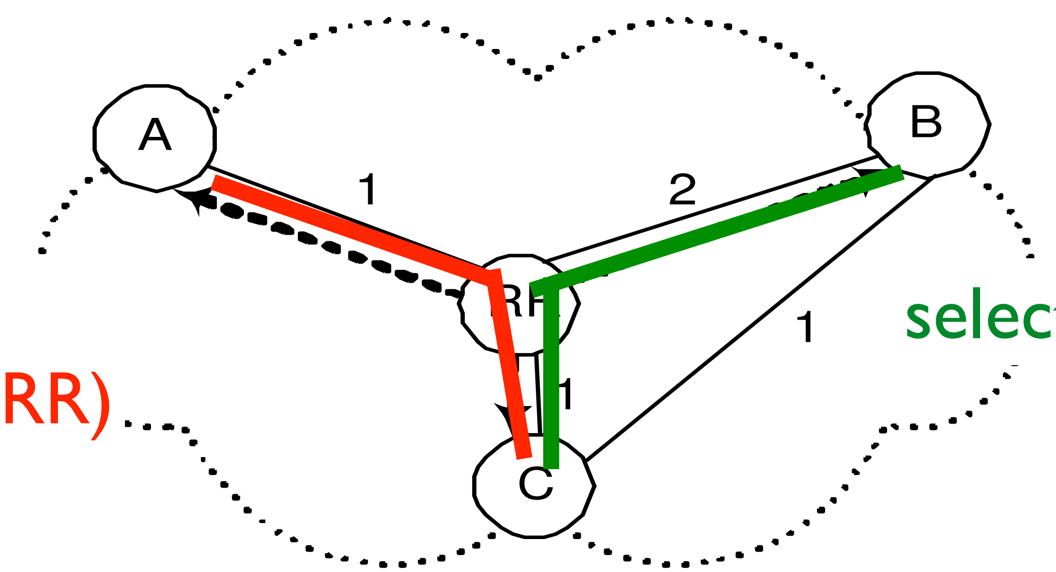
BGP problem: RR ≠ full-mesh





BGP problem: RR ≠ full-mesh





selected by
router reflector (RR)

selected by full mesh

recap: BGP problems

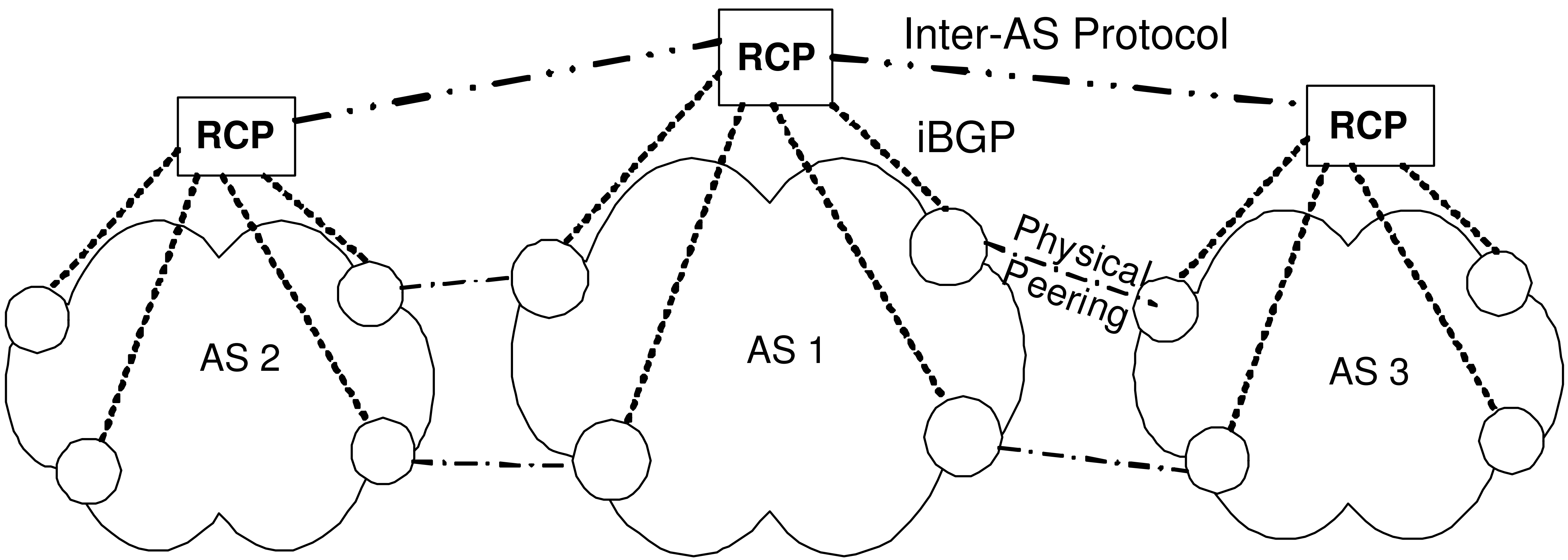
BGP is broken

- converges slowly, sometimes not at all
- routing loops
- misconfigured frequently
- traffic engineering is hard

fixing BGP is hard

- incremental fixes: even more complex
- deployment of new inter-domain protocol almost impossible

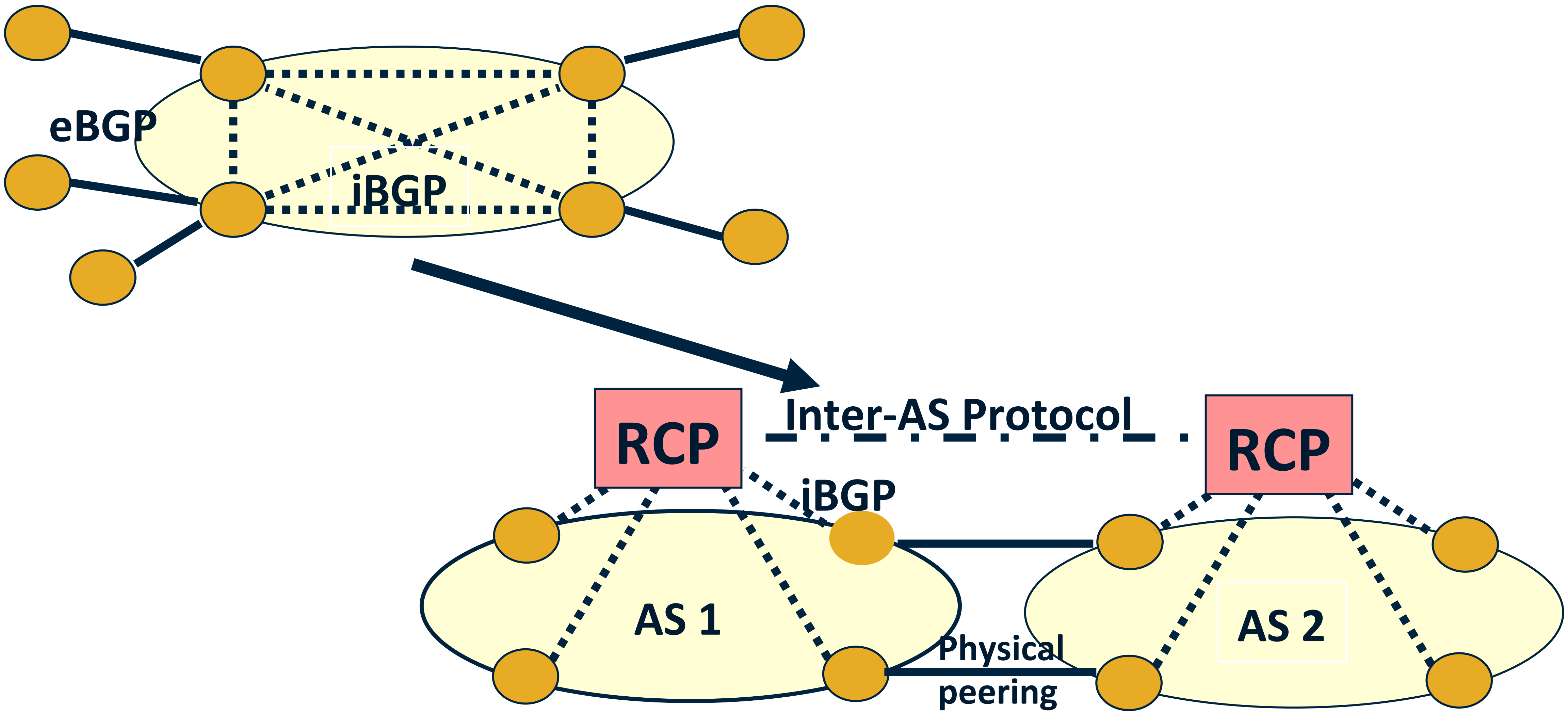
solution: RCP



use centralized controller to customize control

- controller computes routes on behalf of routers
- uses existing routing protocol for control traffic

RCP

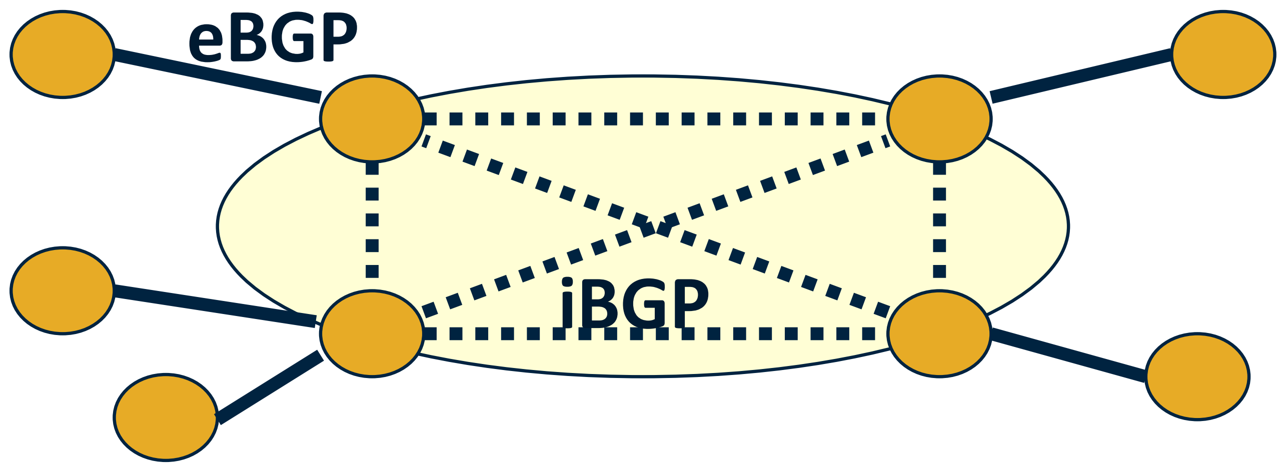


3 phases to achieve

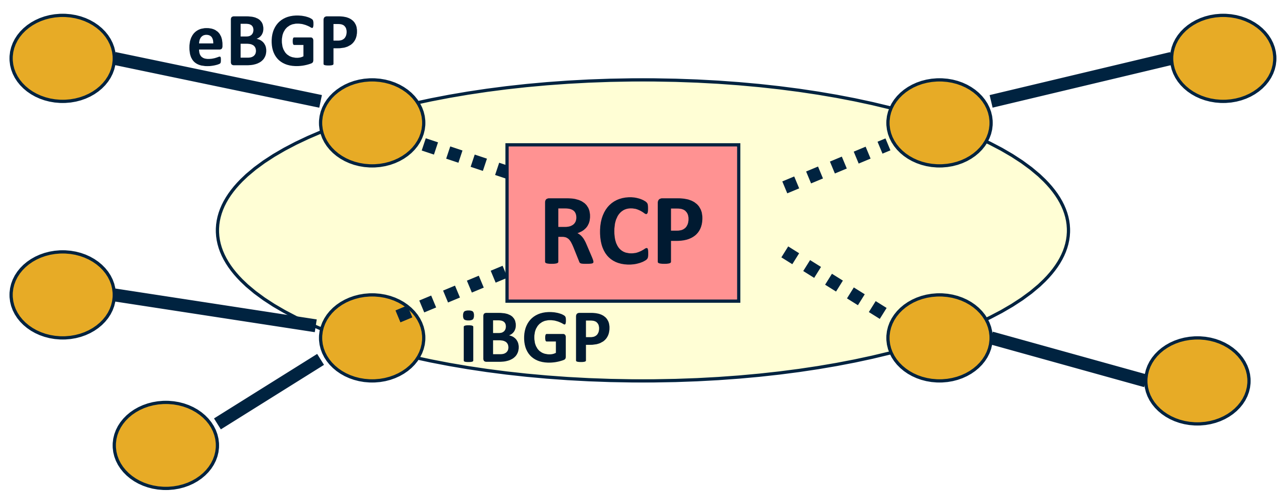
- backward compatibility, deployment incentives

phase 1: control protocol interactions

Before: conventional iBGP



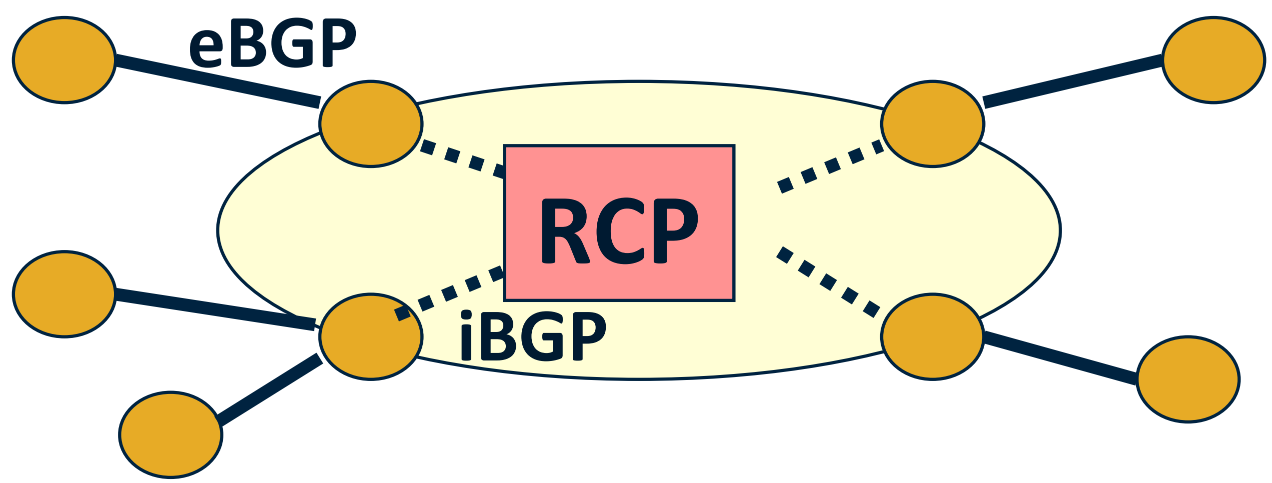
After: RCP gets "best" iBGP routes (and IGP topology)



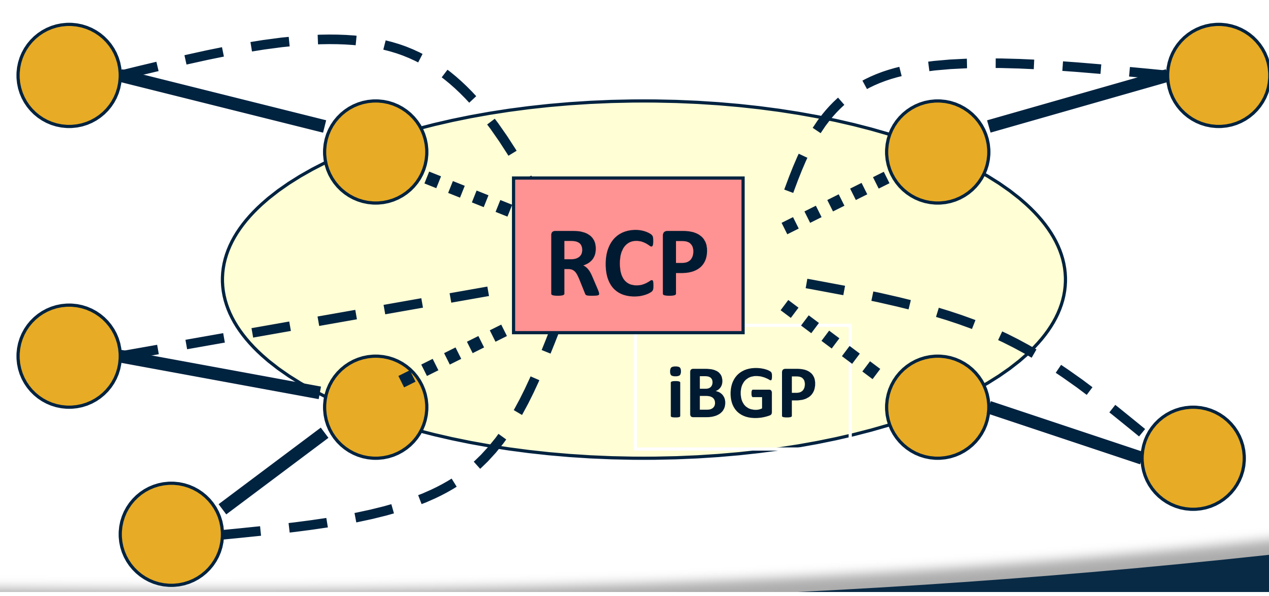
only one AS has to change

phase 2: AS-wide policy

Before: RCP gets “best” iBGP routes (and IGP topology)

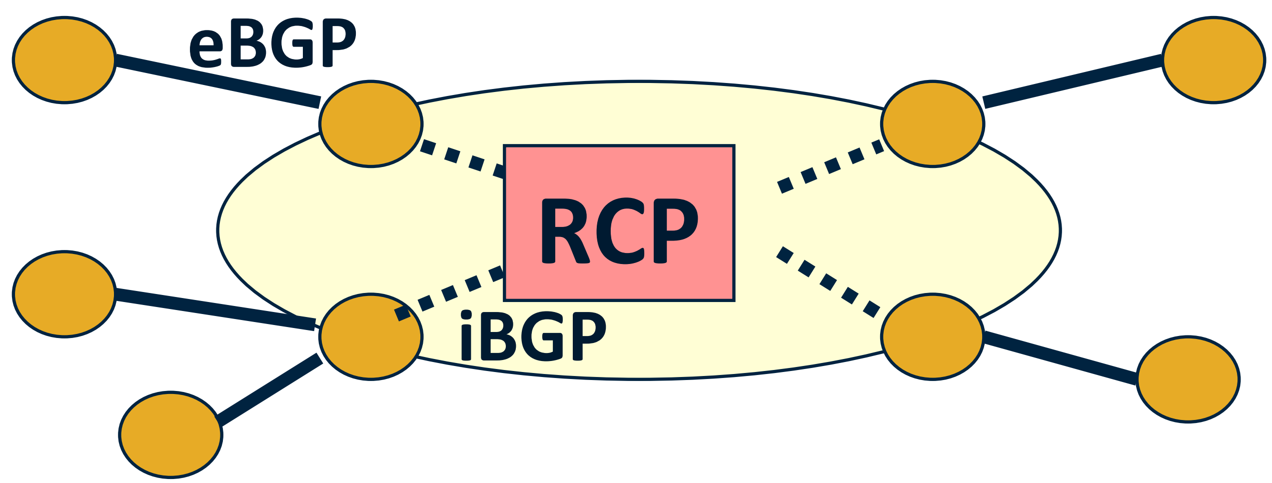


After: RCP gets all eBGP routes from neighbors

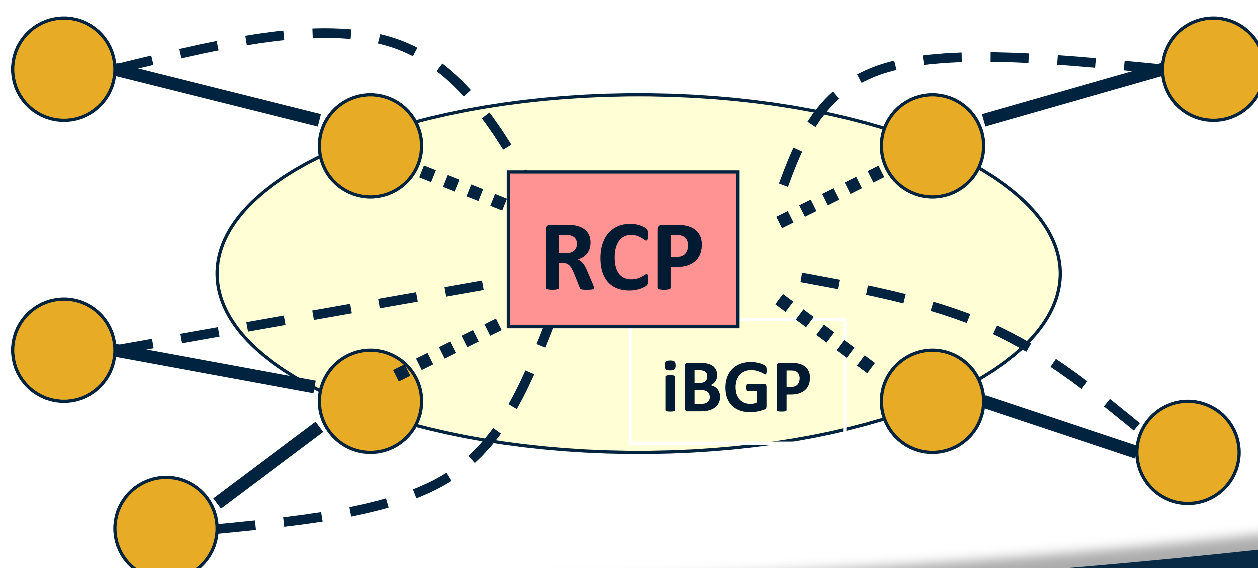


phase 2: AS-wide policy

Before: RCP gets "best" iBGP routes (and IGP topology)

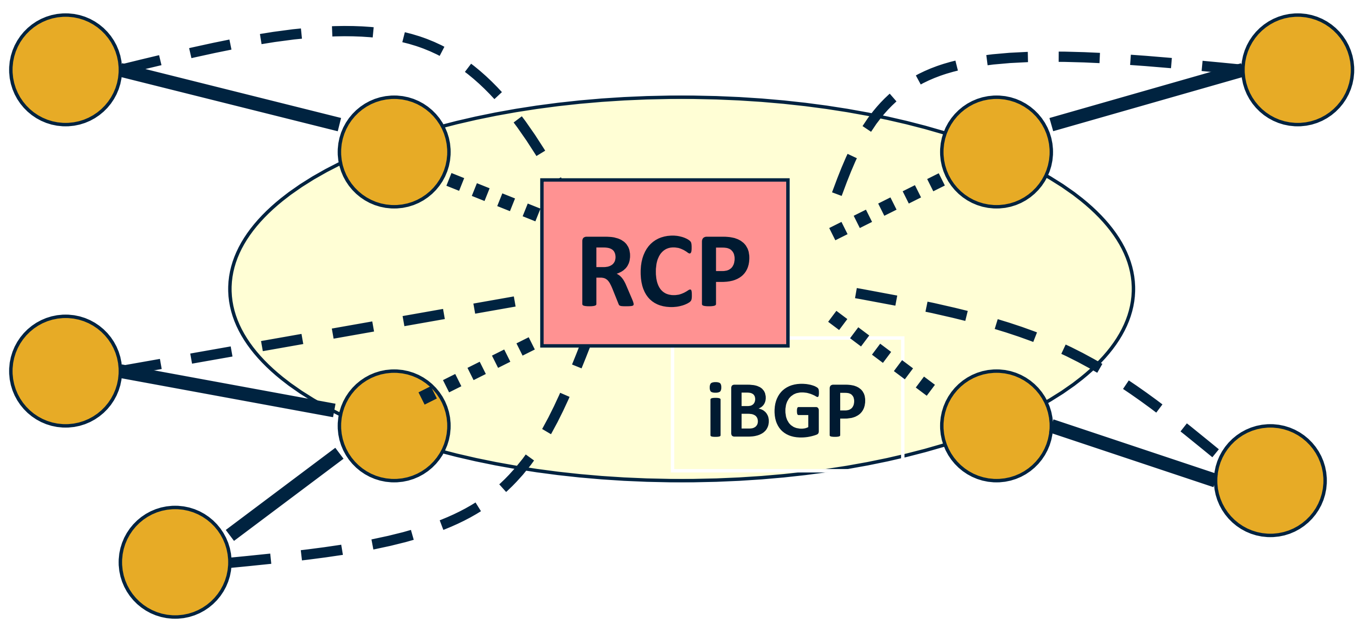


After: RCP gets all eBGP routes from neighbors

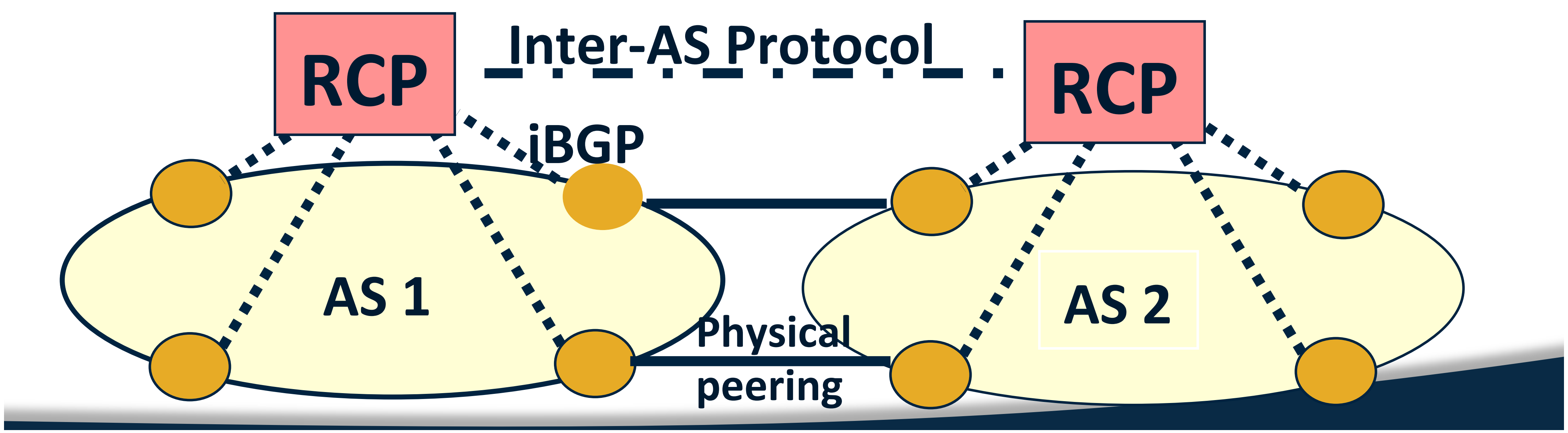


phase 3: all ASes have RCP

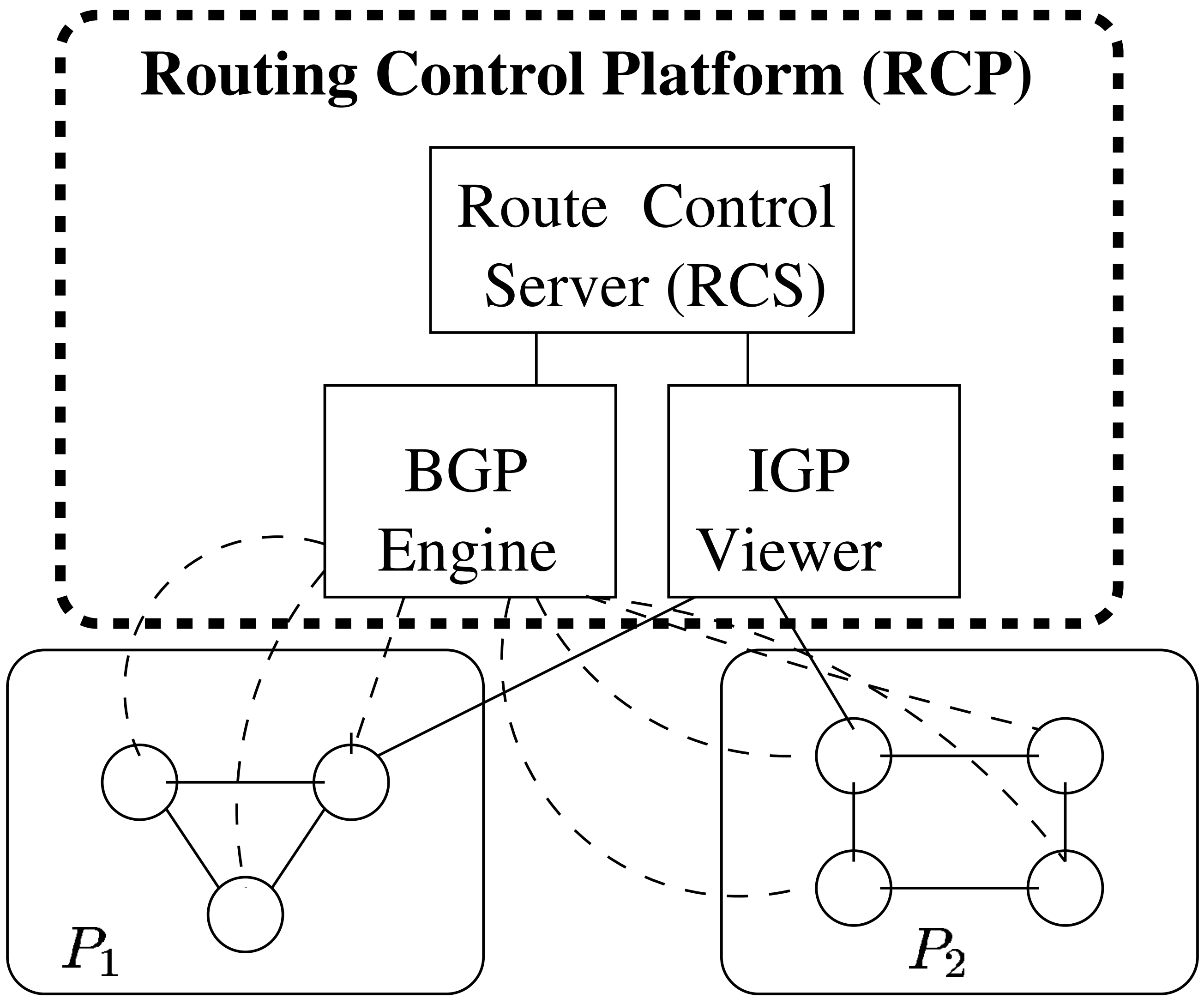
Before: RCP gets all eBGP routes from neighbo



After: ASes exchange routes via RCP



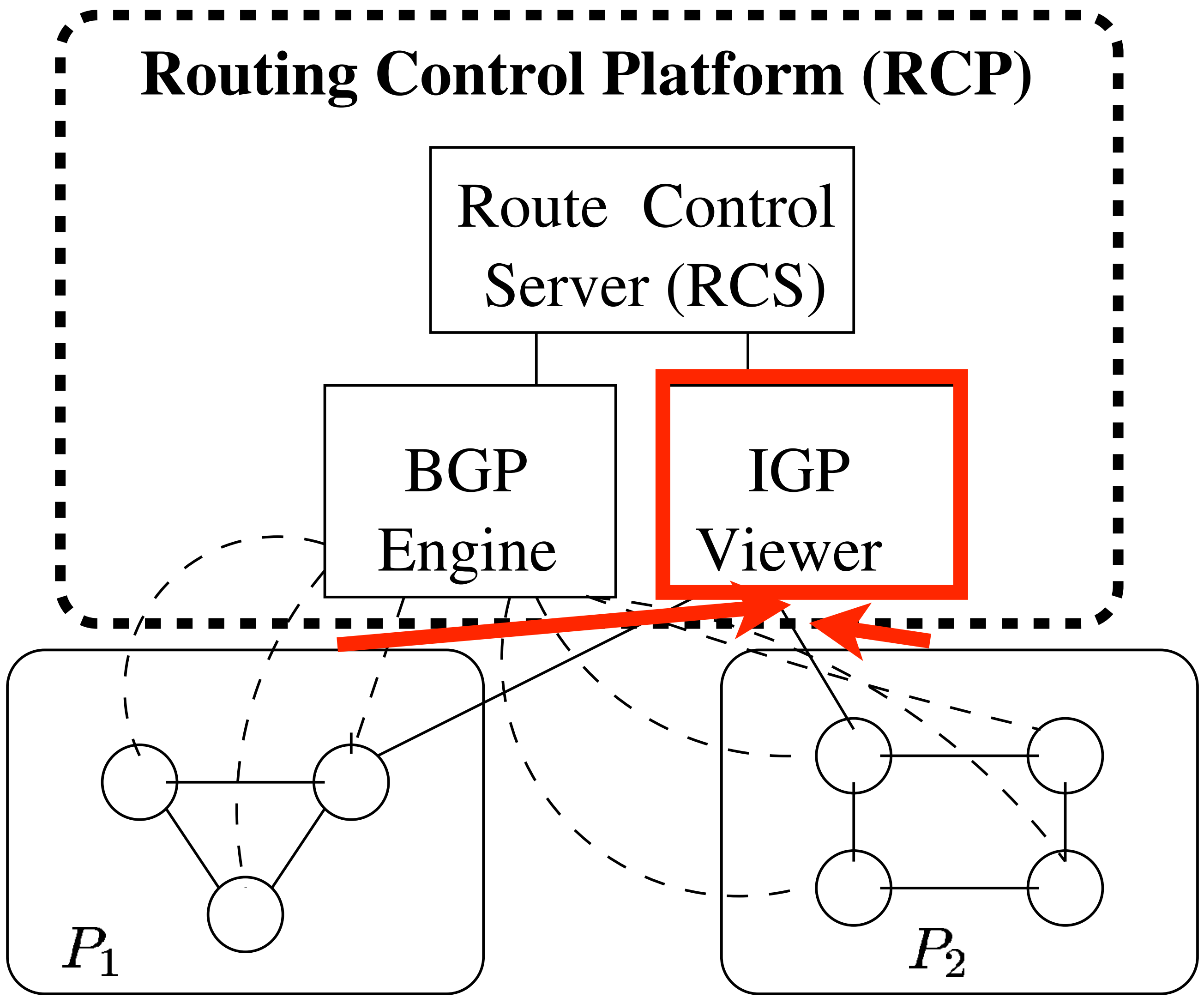
RCP architecture (phase 1)



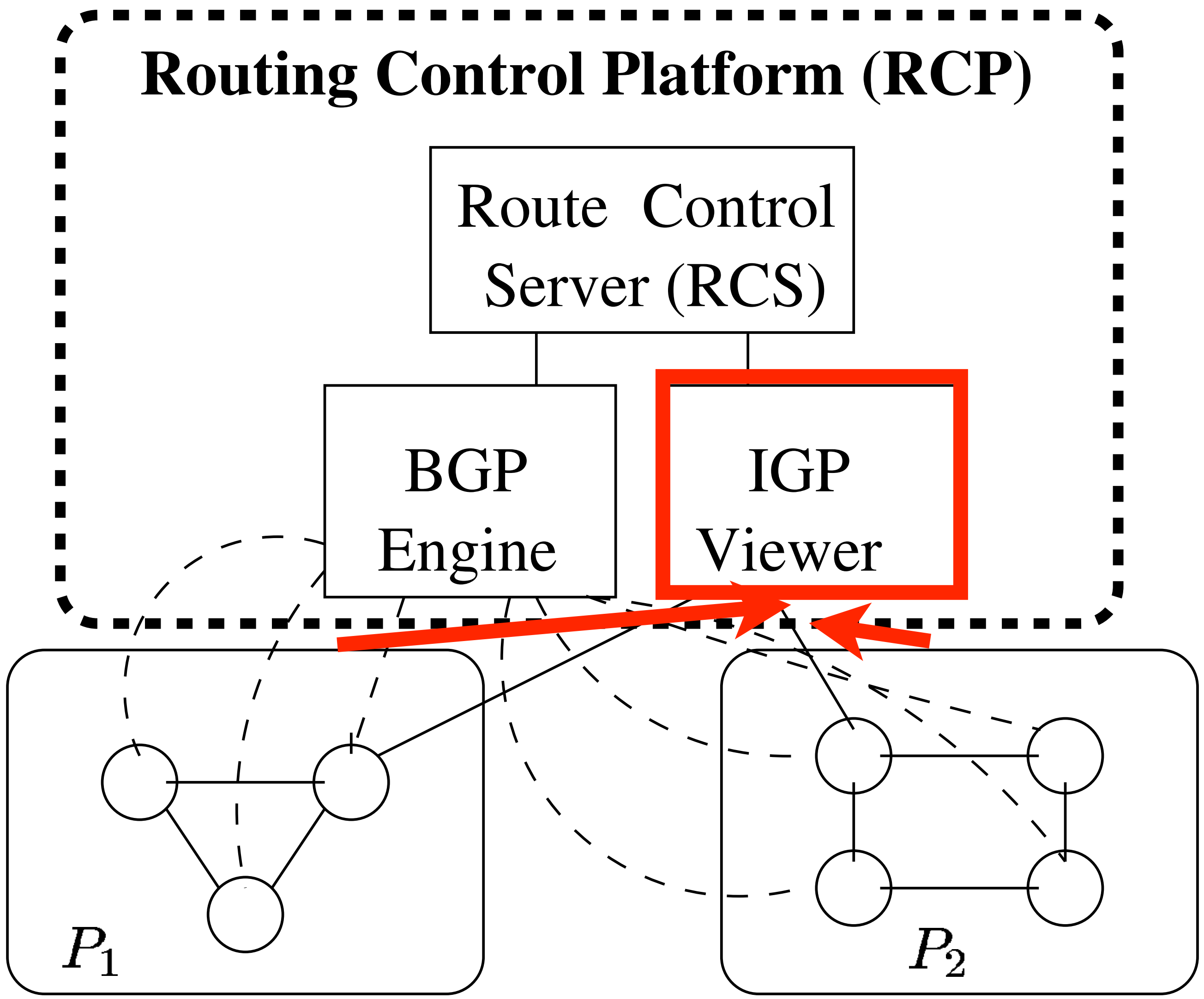
P1, P2

- IGP partitions

RCP architecture



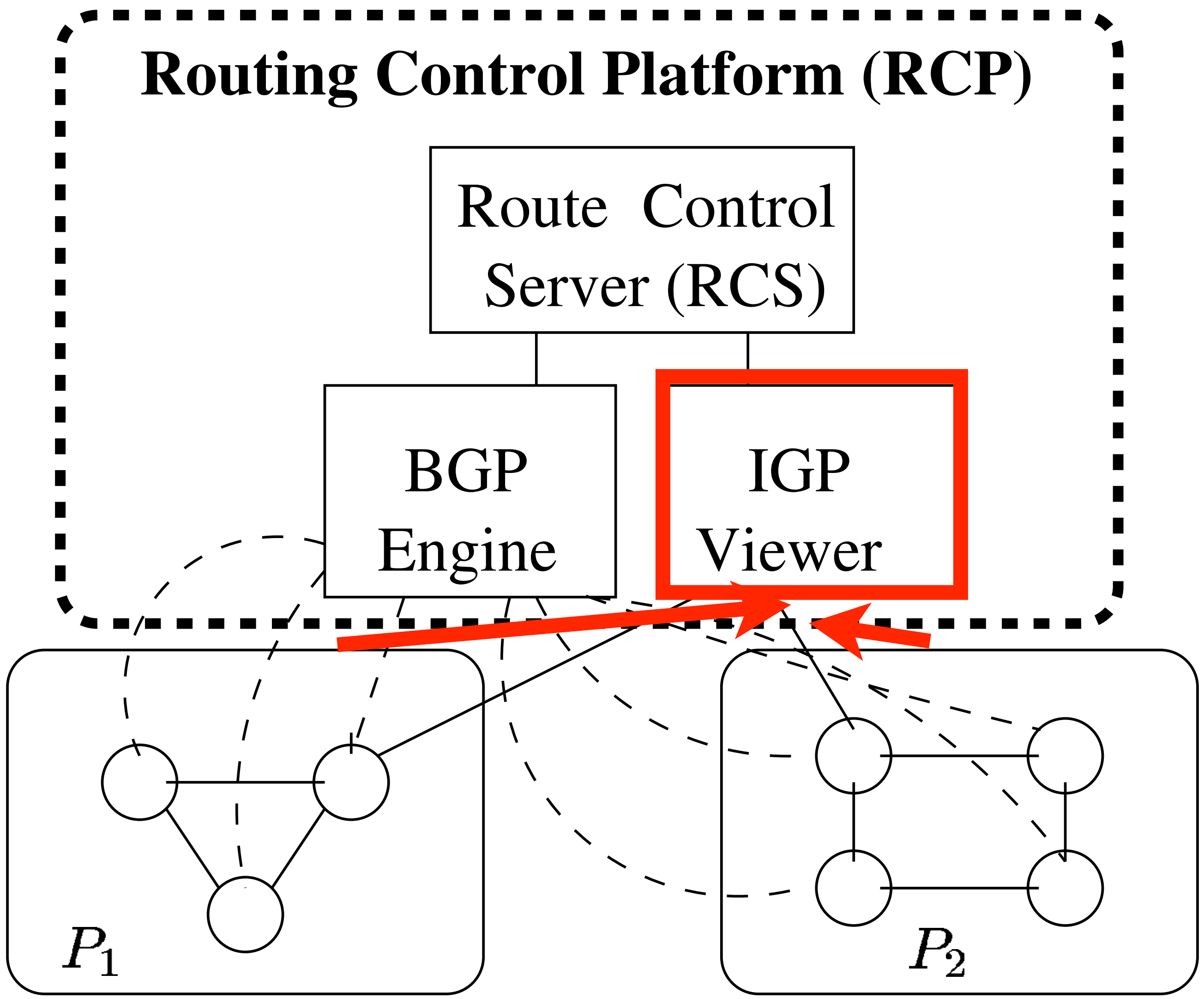
RCP architecture



IGP viewer

- maintains IGP topology
- computes pairwise shortest paths with AS

RCP architecture



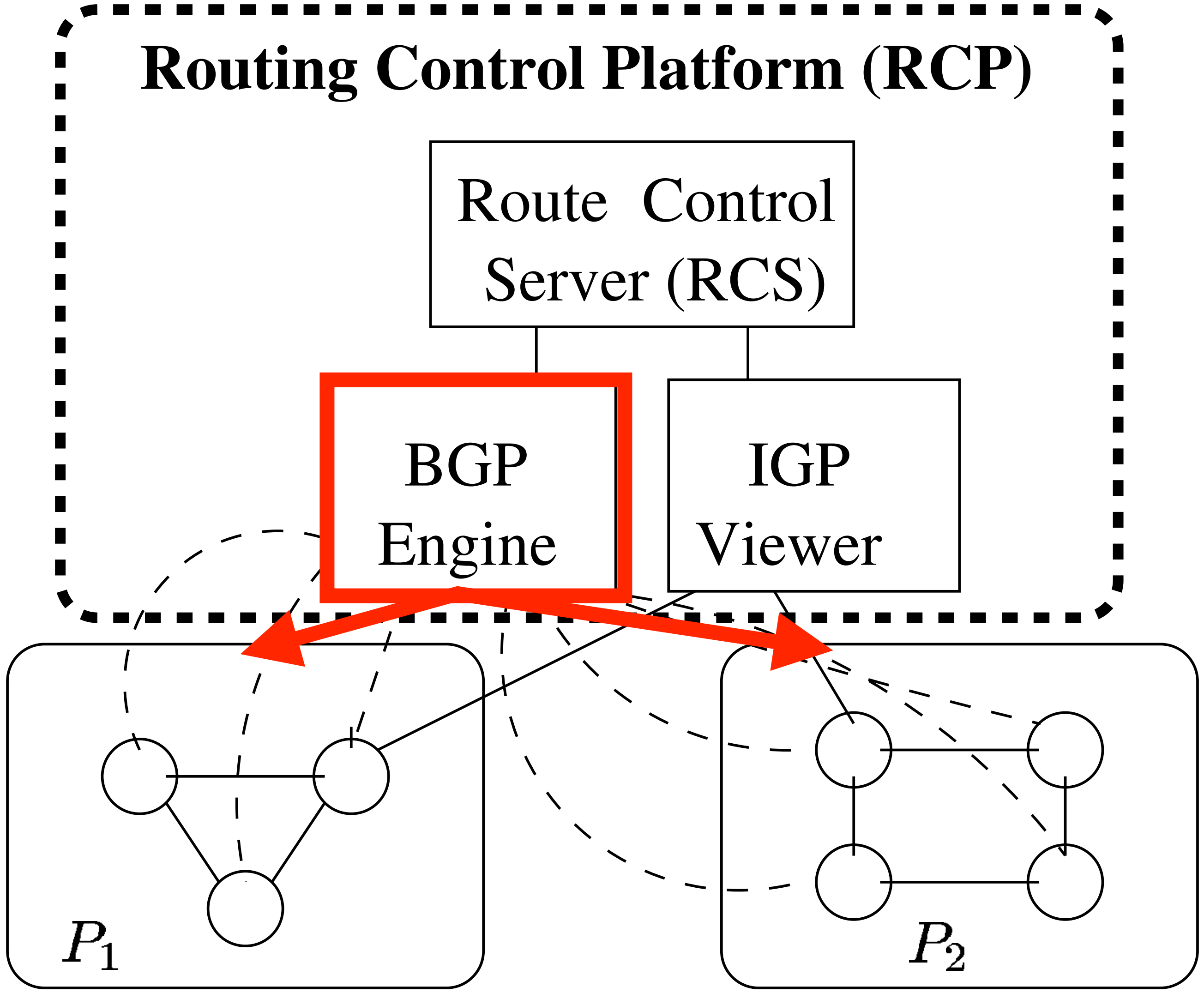
IGP viewer

- maintains IGP topology
- computes pairwise shortest paths with AS

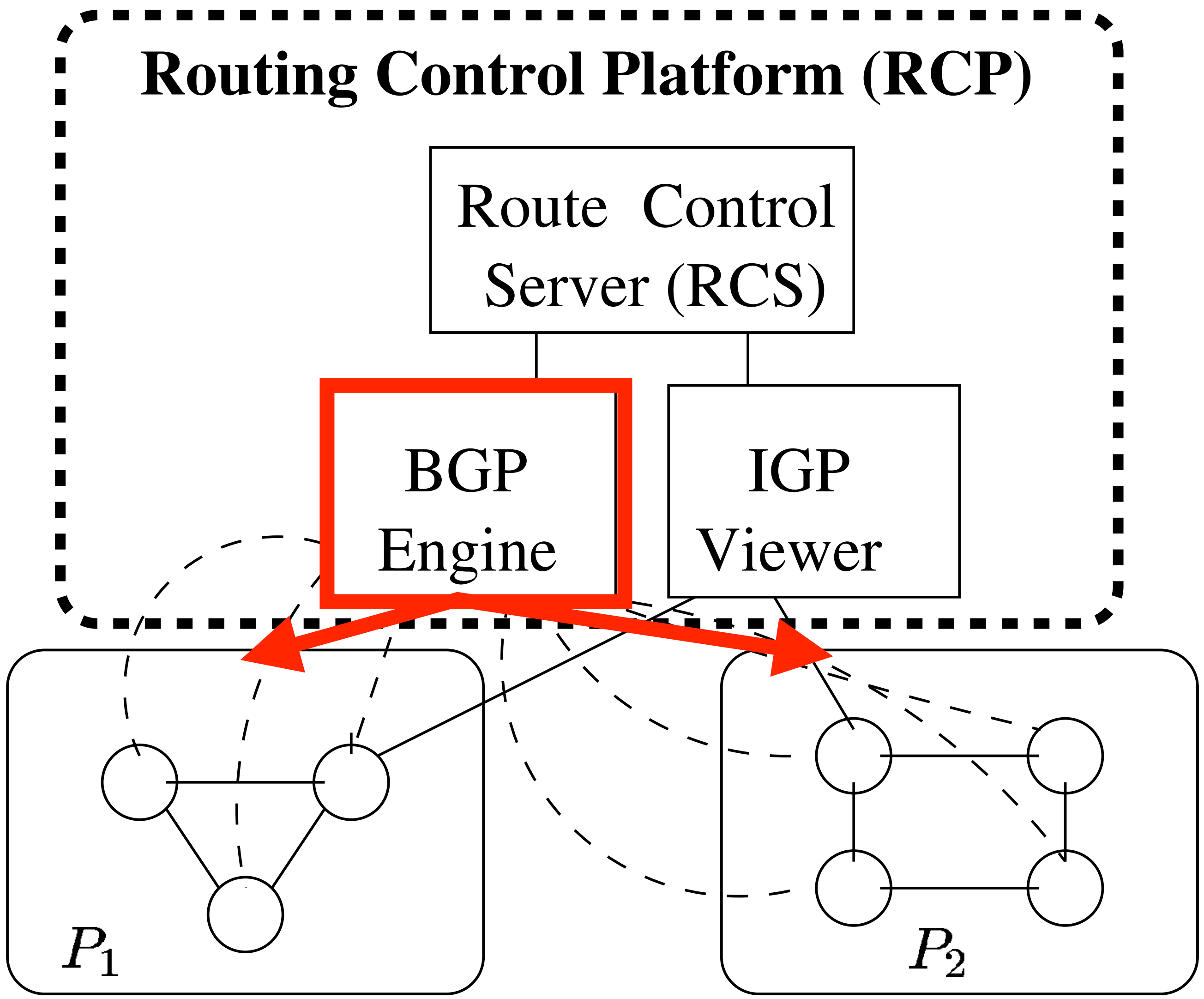
benefit: scalability

- cluster routers
- reduce # independent route computation

RCP architecture



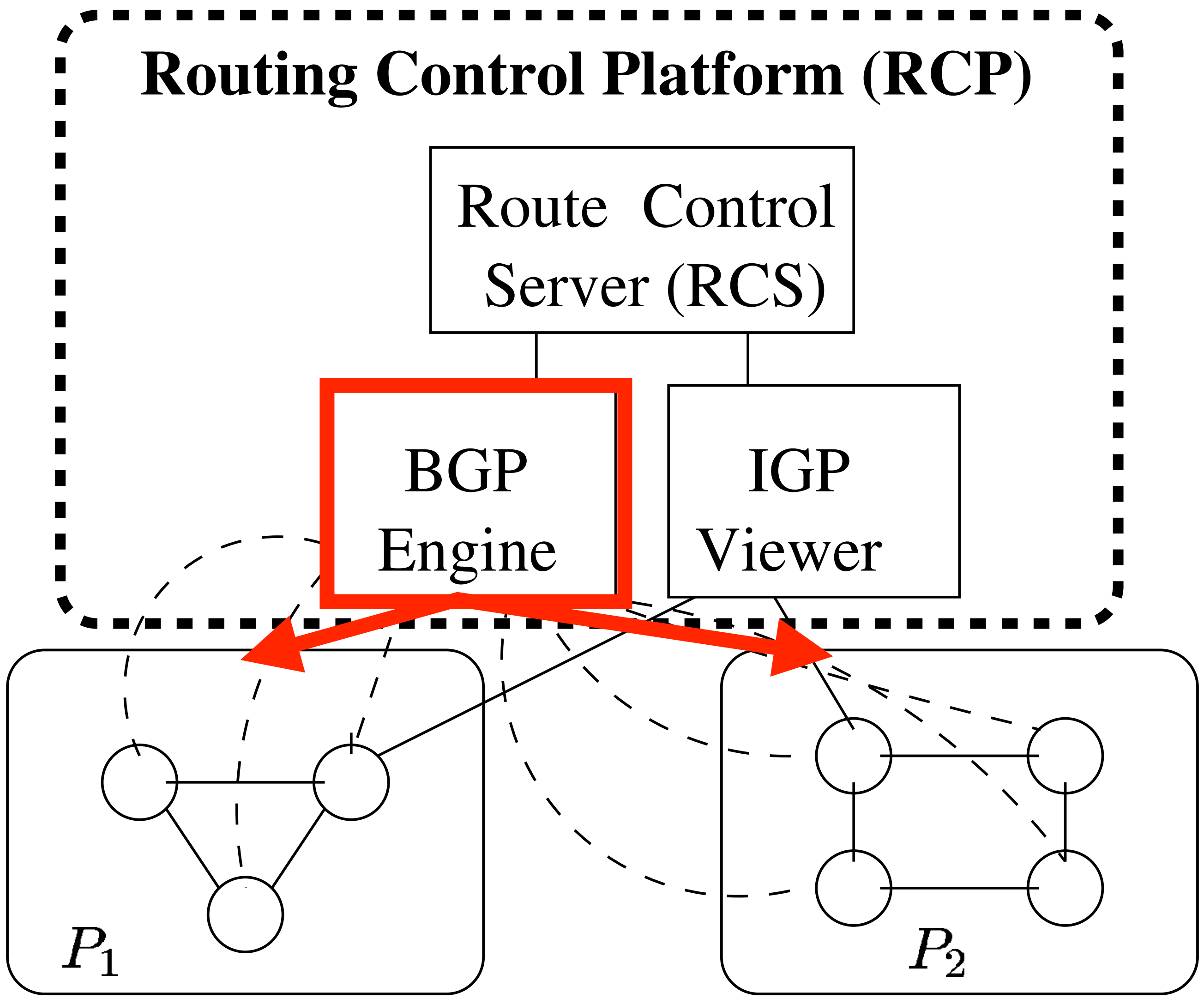
RCP architecture



BGP engine

- communicates RCS decision to routers via iBGP

RCP architecture



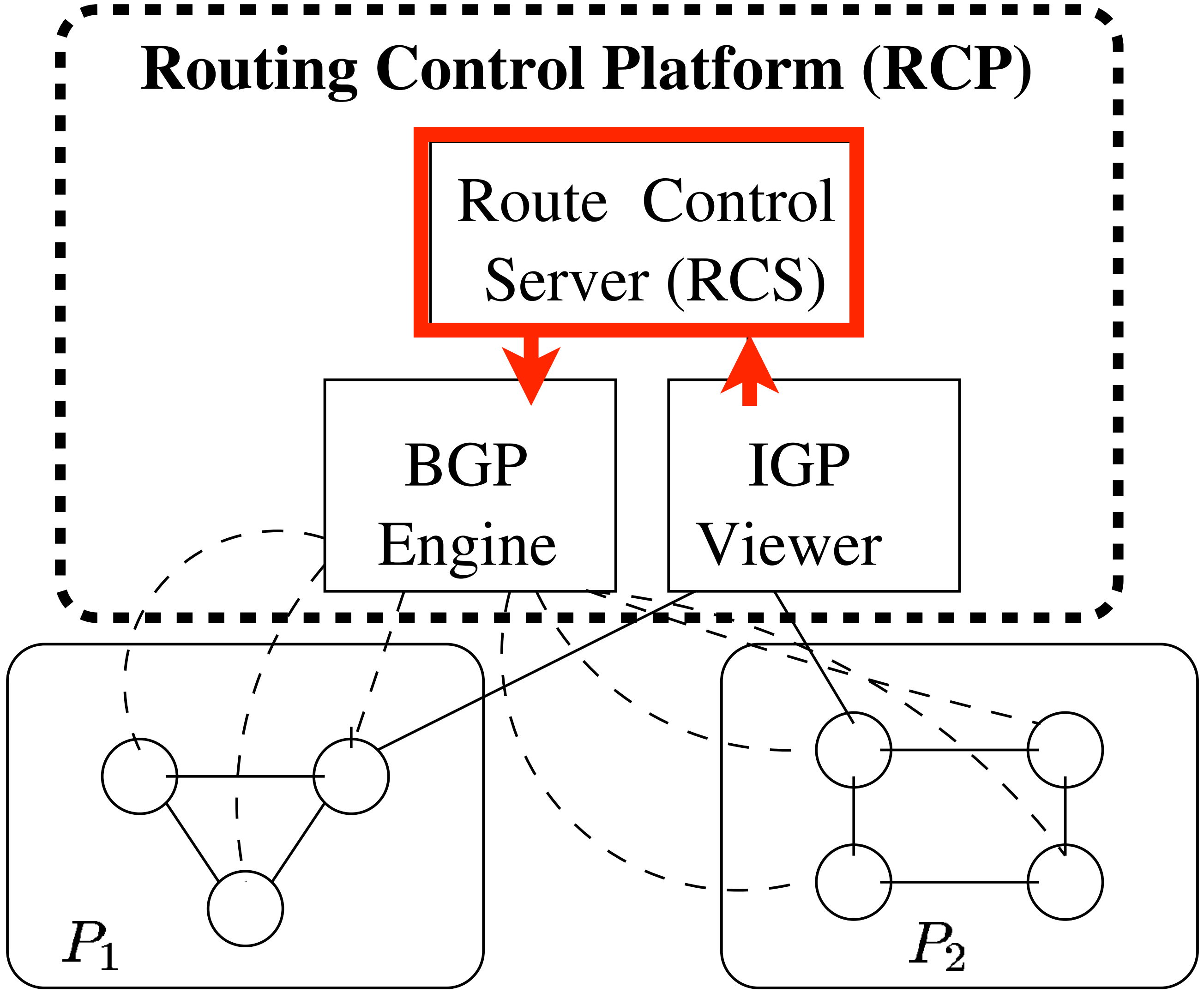
BGP engine

- communicates RCS decision to routers via iBGP

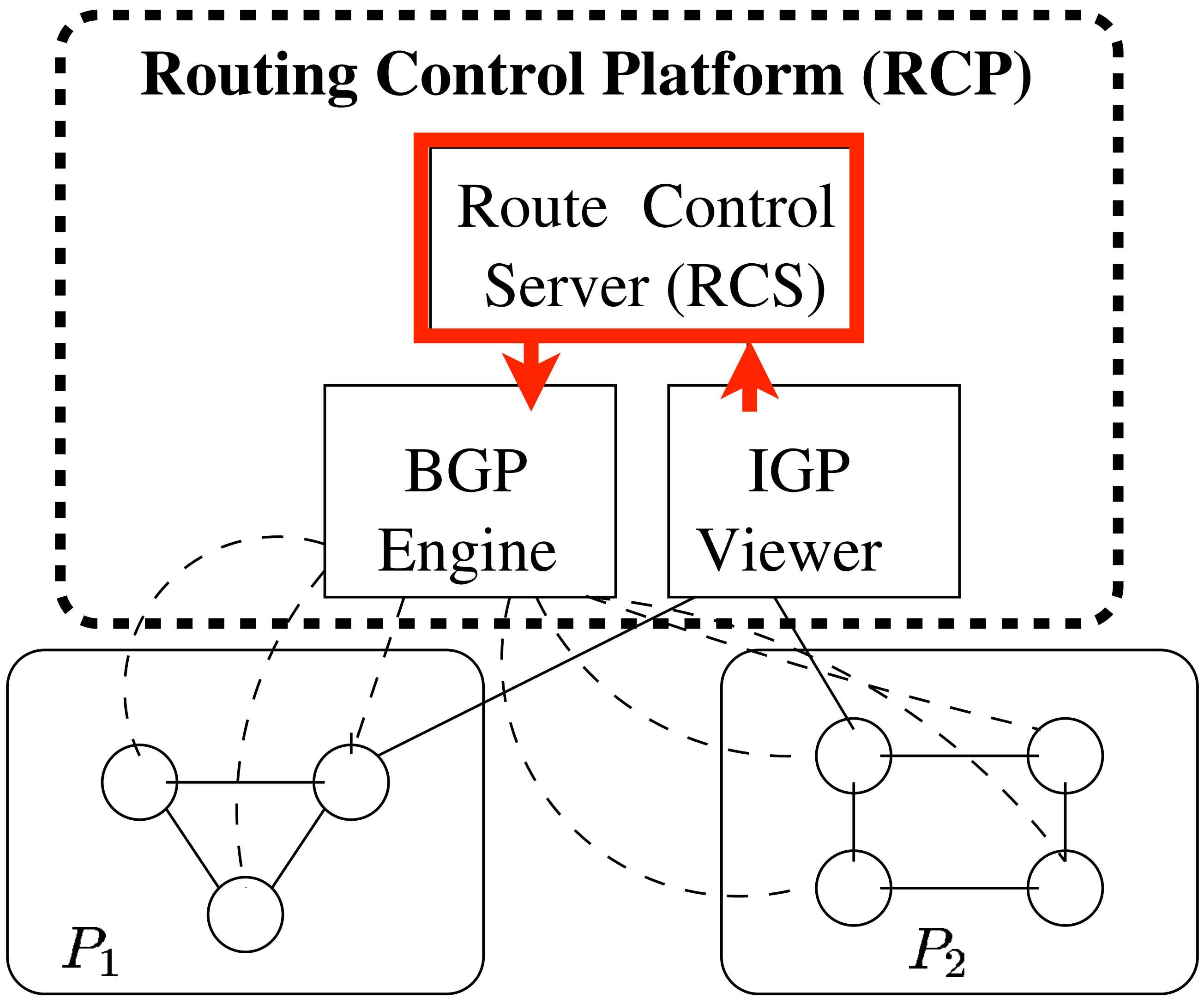
benefit

- backward-compatibility

RCP architecture



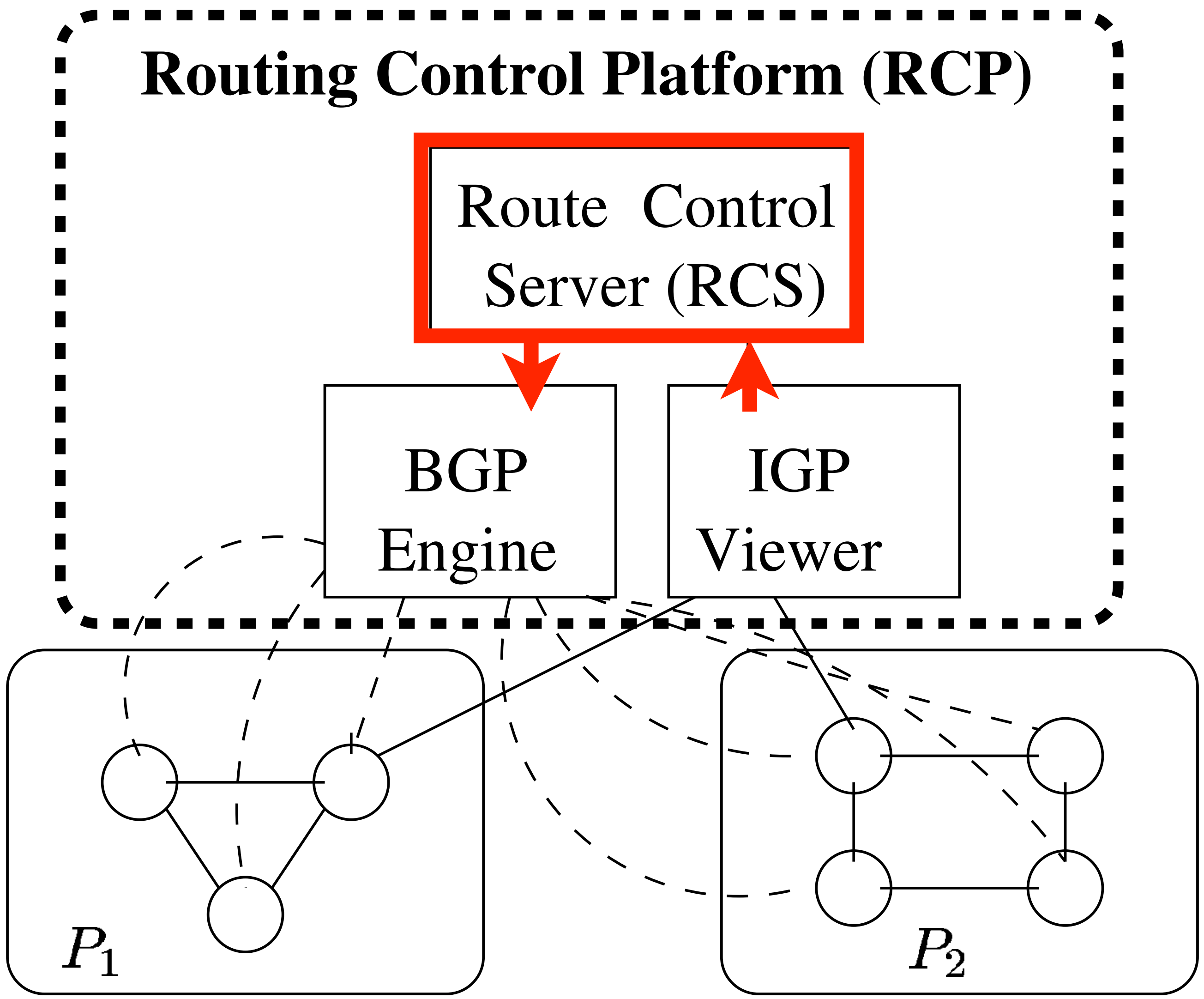
RCP architecture



RCS

- computes BGP route assignments
- obtain topology from IGP
- disseminate decision via BGP engine

RCP architecture



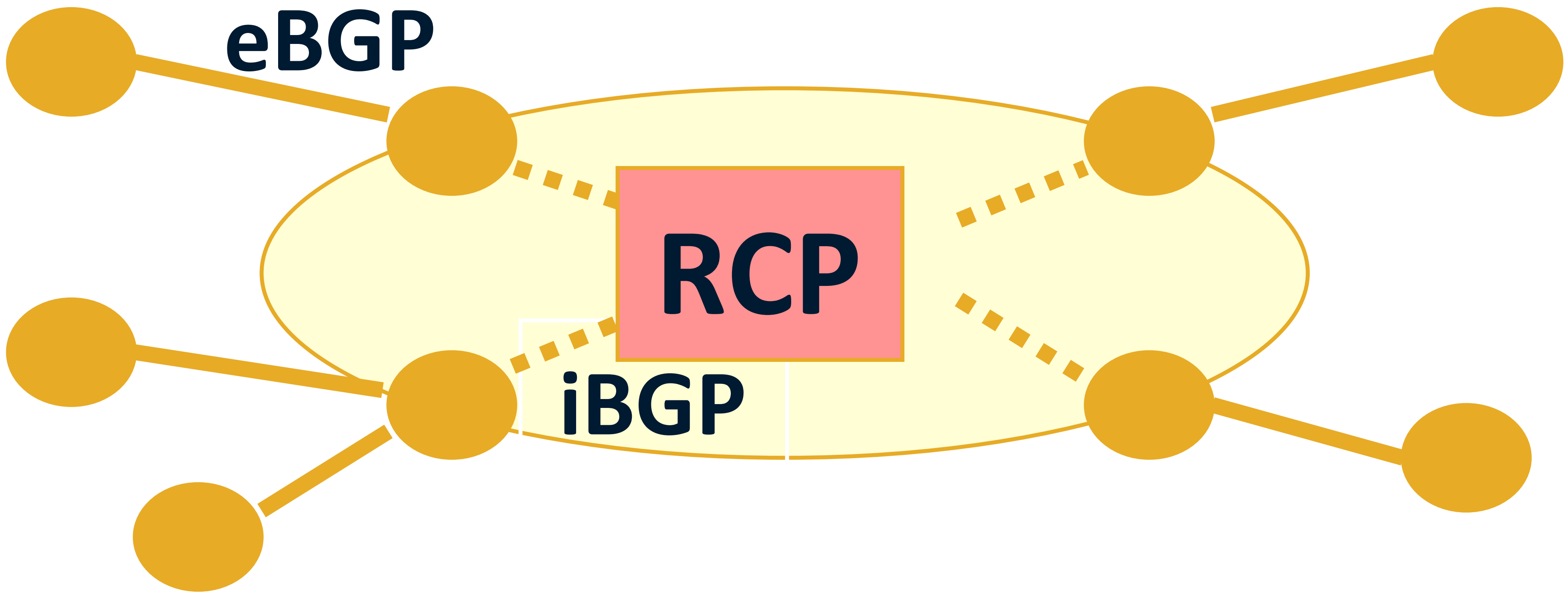
RCS

- computes BGP route assignments
- obtain topology from IGP
- disseminate decision via BGP engine

benefit

- correctness
  - route validity
  - path visibility

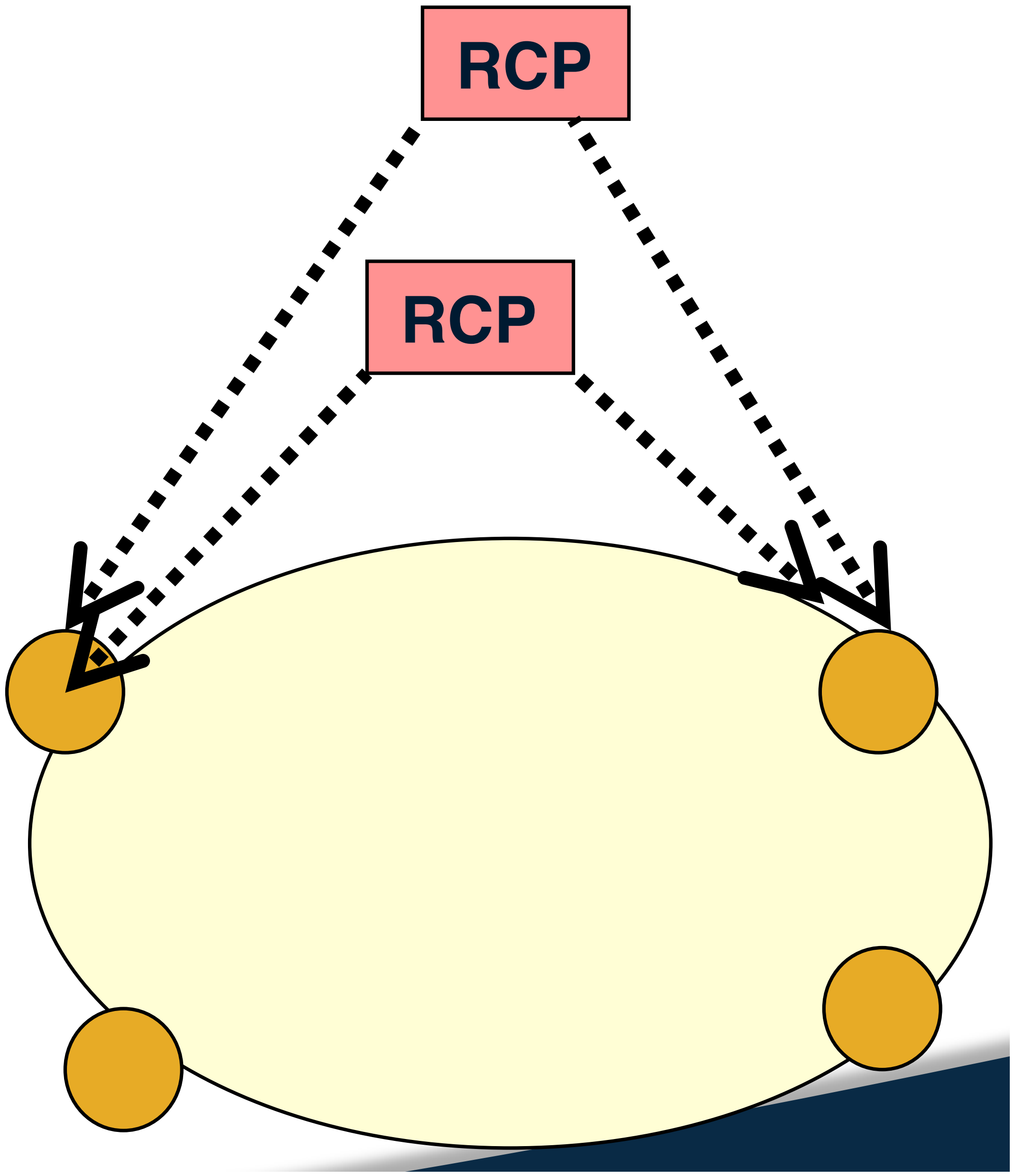
scalability, efficiency, and reliability



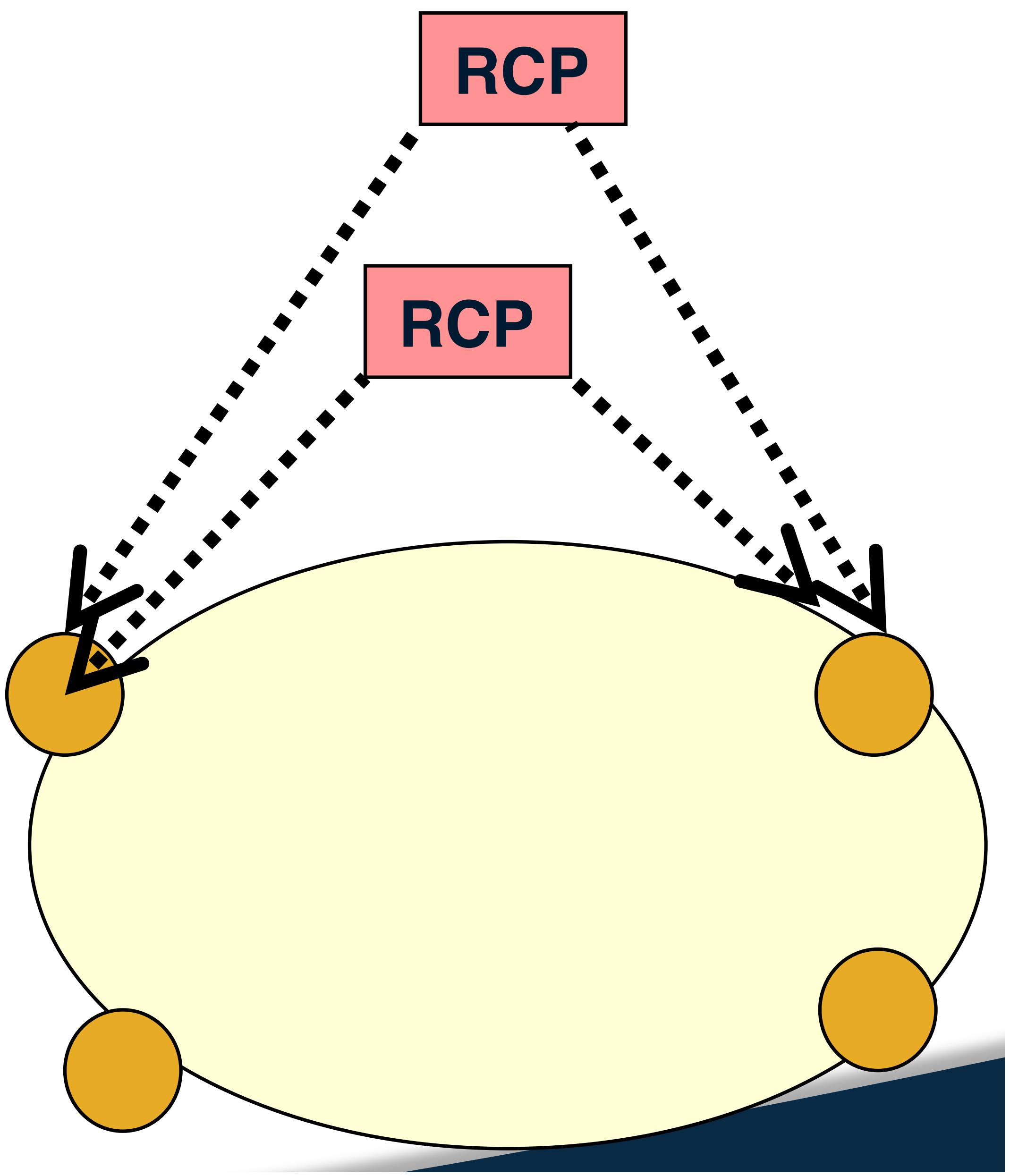
requirements

- many routers (500-1000)
- many destination prefixes (150,000-200,000)
- converge quickly

reliability



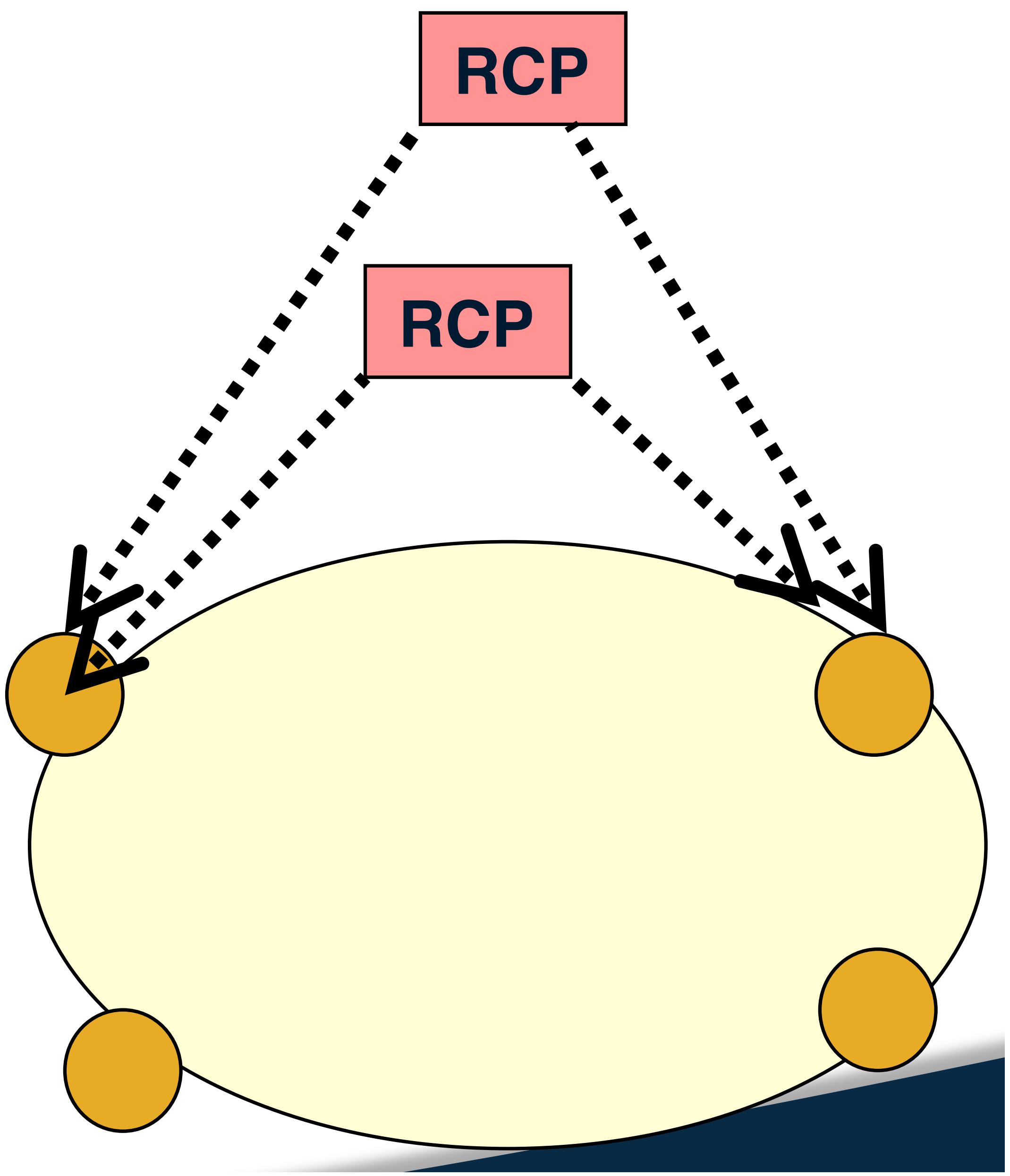
reliability



replicate RCP

- multiple identical servers

reliability



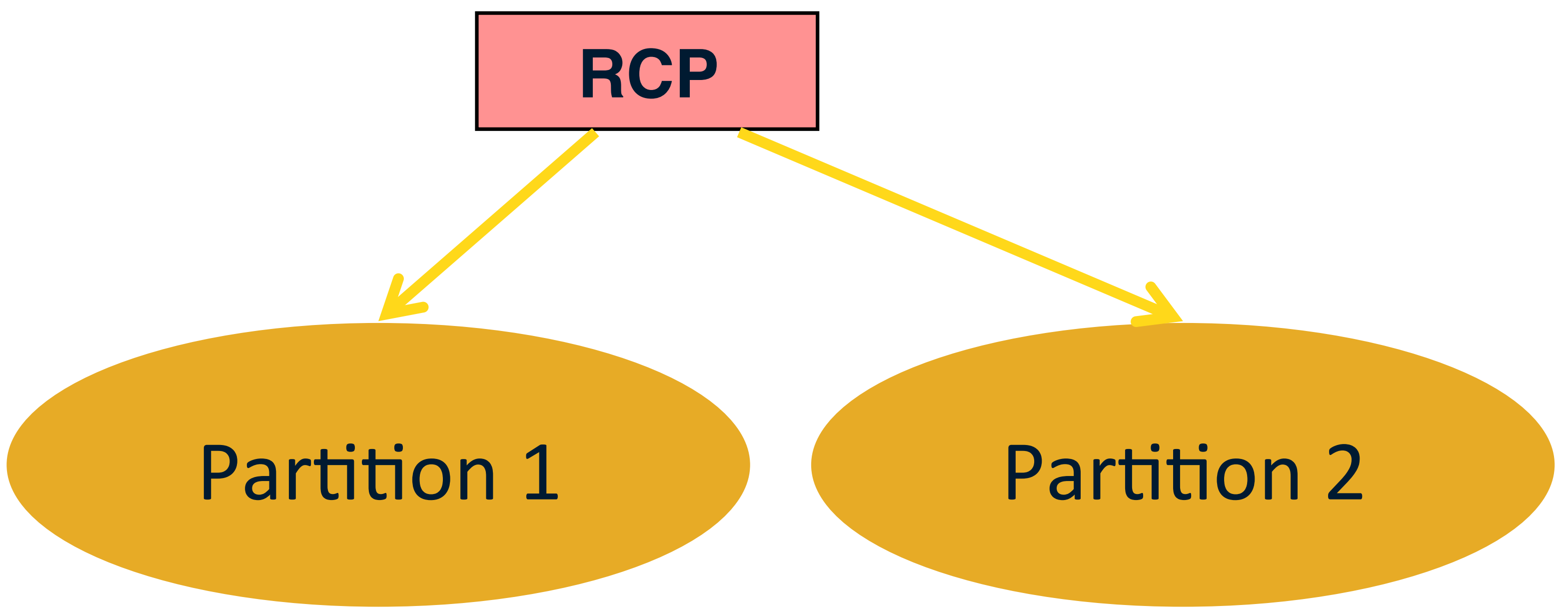
replicate RCP

- multiple identical servers

independent replicas

- each receives same information, running the same routing algorithm
- *NO* need for a consistency protocol if both replicas always see the same information

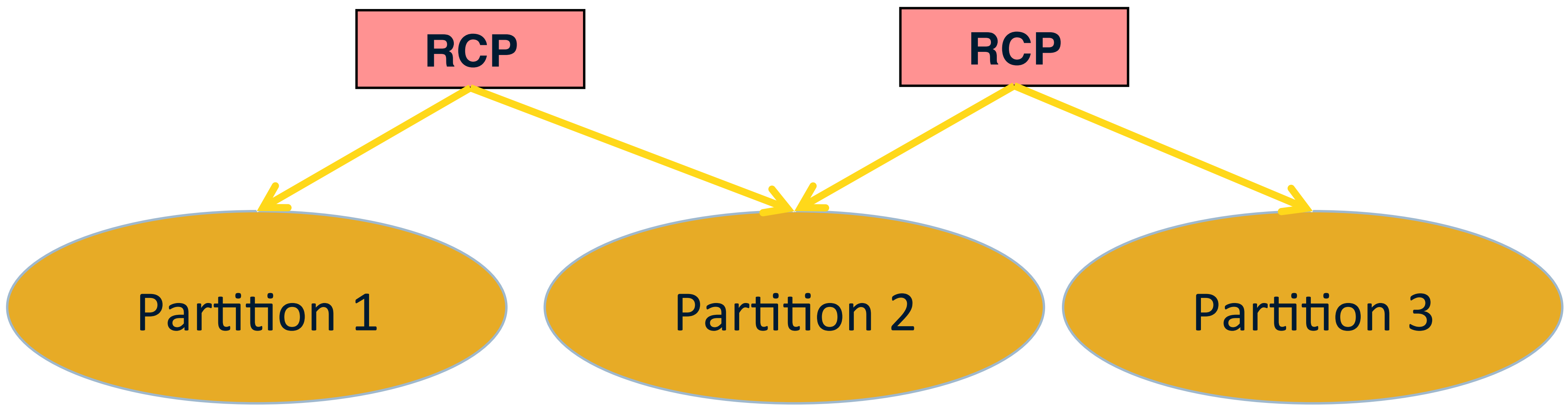
single RCP under partition



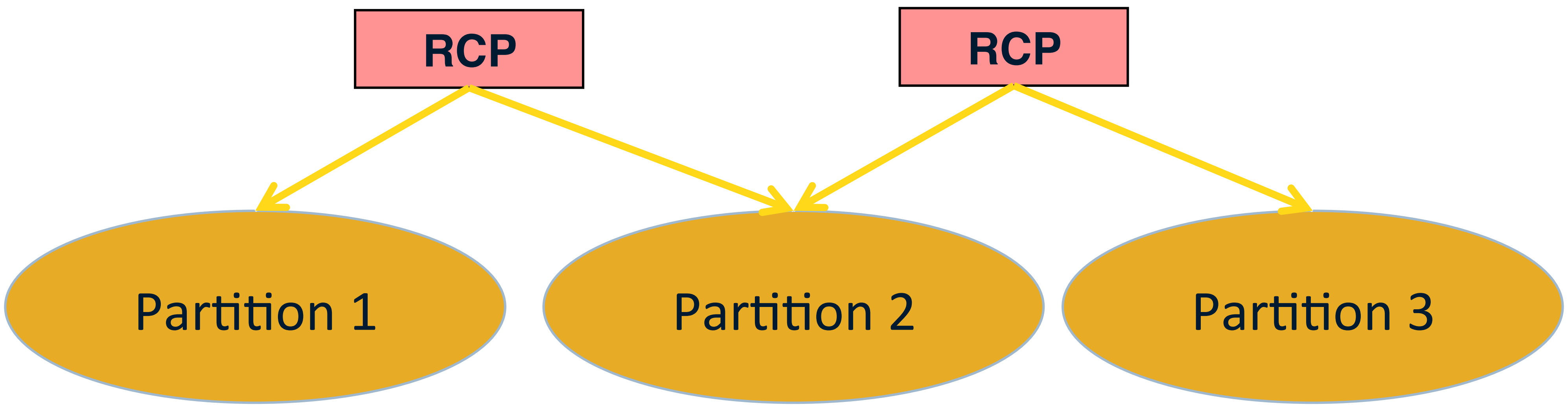
only use state from routers' partition to assign BGP route

- route validity: ensure next-hop is reachable

multiple RCPs under partition



multiple RCPs under partition



RCPs receive same state from each reachable partition

- path visibility: each partition connects to at least one RCP

evaluation

evaluation

goal: determine the feasible operation conditions

evaluation

goal: determine the feasible operation conditions

- performance of RCP as a function of the number or prefixes, routes, and number of routers

evaluation

goal: determine the feasible operation conditions

- performance of RCP as a function of the number or prefixes, routes, and number of routers

methodology

evaluation

goal: determine the feasible operation conditions

- performance of RCP as a function of the number or prefixes, routes, and number of routers

methodology

- use a single BGP and OSPF data set (from a Tier-1 ISP) to "simulate" multiple network sizes

evaluation

goal: determine the feasible operation conditions

- performance of RCP as a function of the number or prefixes, routes, and number of routers

methodology

- use a single BGP and OSPF data set (from a Tier-1 ISP) to "simulate" multiple network sizes
  - timestamped BGP updates, BGP table dumps, timed OSPF link updates

evaluation

goal: determine the feasible operation conditions

- performance of RCP as a function of the number or prefixes, routes, and number of routers

methodology

- use a single BGP and OSPF data set (from a Tier-1 ISP) to "simulate" multiple network sizes
  - timestamped BGP updates, BGP table dumps, timed OSPF link updates
- varying network size

evaluation

goal: determine the feasible operation conditions

- performance of RCP as a function of the number or prefixes, routes, and number of routers

methodology

- use a single BGP and OSPF data set (from a Tier-1 ISP) to "simulate" multiple network sizes
  - timestamped BGP updates, BGP table dumps, timed OSPF link updates
- varying network size
  - selectively filter data set

key metrics

measure decision (route selection) time + memory usage

- white box
- blackbox no queuing
- blackbox real-time

processing time

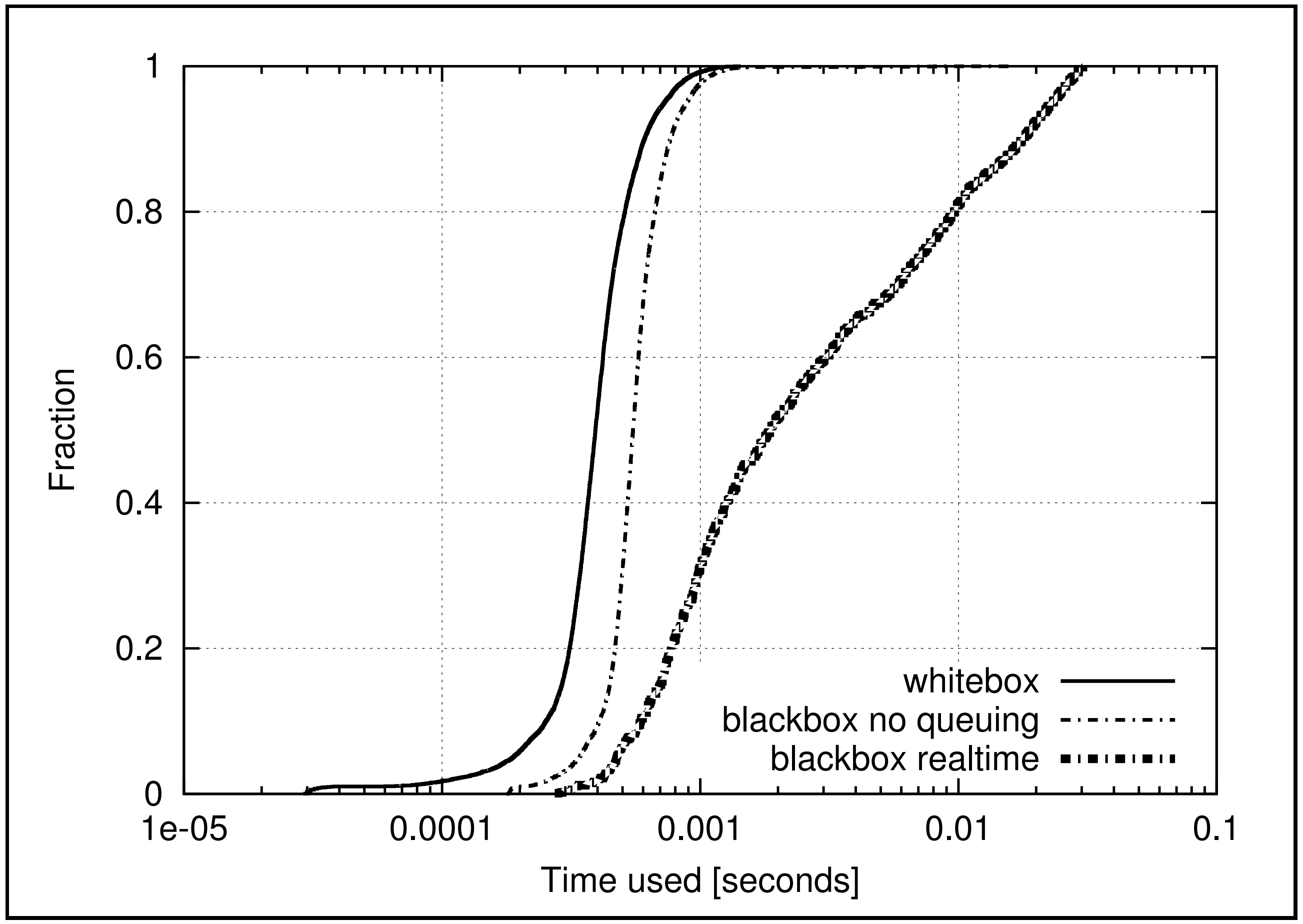


Figure 9: *Decision time, BGP updates*: RCS route selection time for whitebox testing (instrumented RCS), blackbox testing no queuing (single BGP announcements sent to RCS at a time), blackbox testing real-time (BGP announcements sent to RCS in real-time)

discussion: why results on the two blackbox results differ?

memory usage

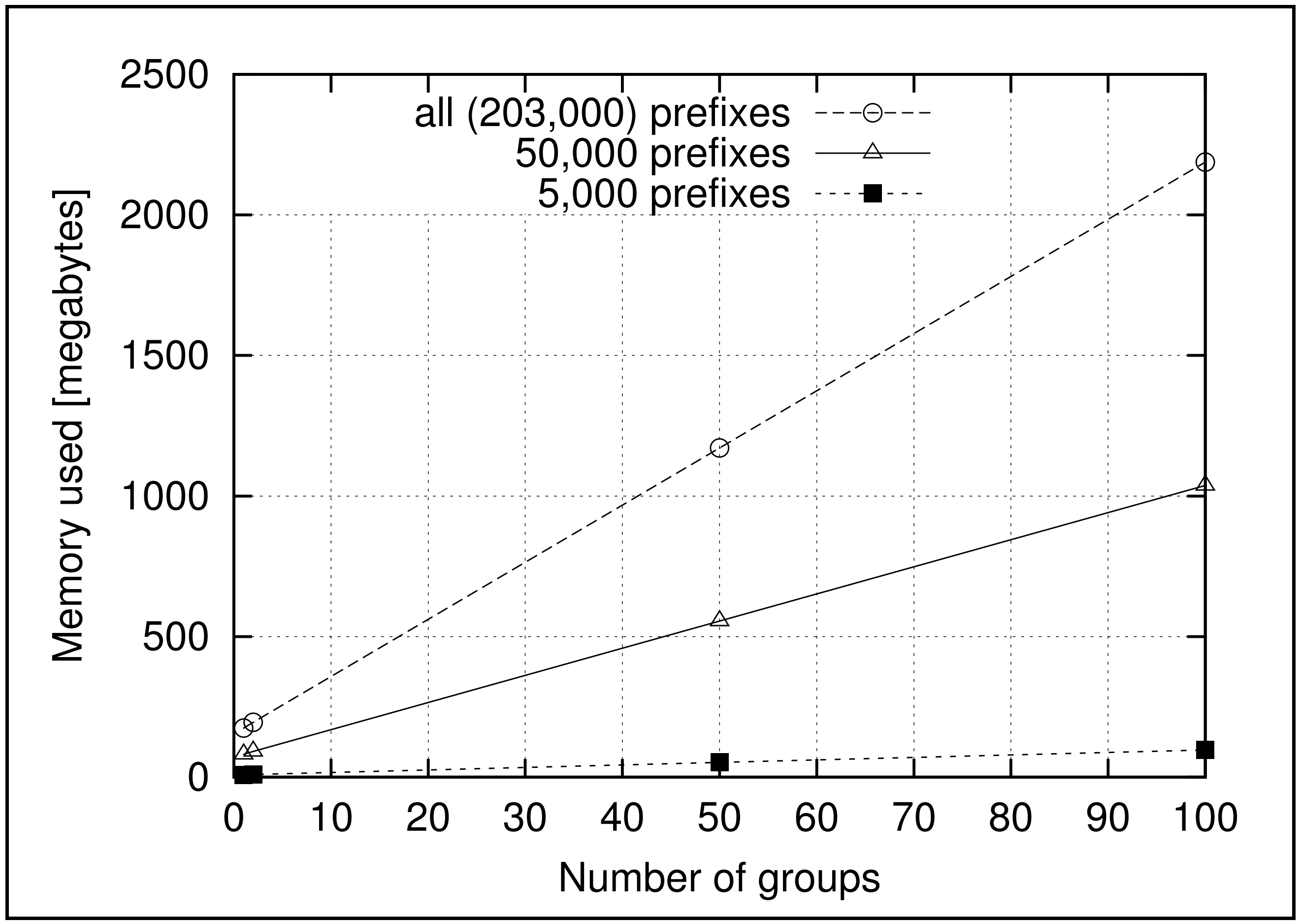


Figure 8: *Memory*: Memory used for varying numbers of prefixes.

memory requirement as a function of group size and for different number of prefixes

- modest increase as # of group grows

three continual challenges

three continual challenges

scalability

- large topology, huge volume of events, flow initiations

three continual challenges

scalability

- large topology, huge volume of events, flow initiations

reliability

- handle equipment (and other) failover gracefully

three continual challenges

scalability

- large topology, huge volume of events, flow initiations

reliability

- handle equipment (and other) failover gracefully

performance

- low control-plane latency

NOX, Onix

challenges

performance

- low control-plane latency

scalability

- large topology, huge volume of events, flow initiations

reliability

- handle equipment (and other) failover gracefully

opportunities

simplicity

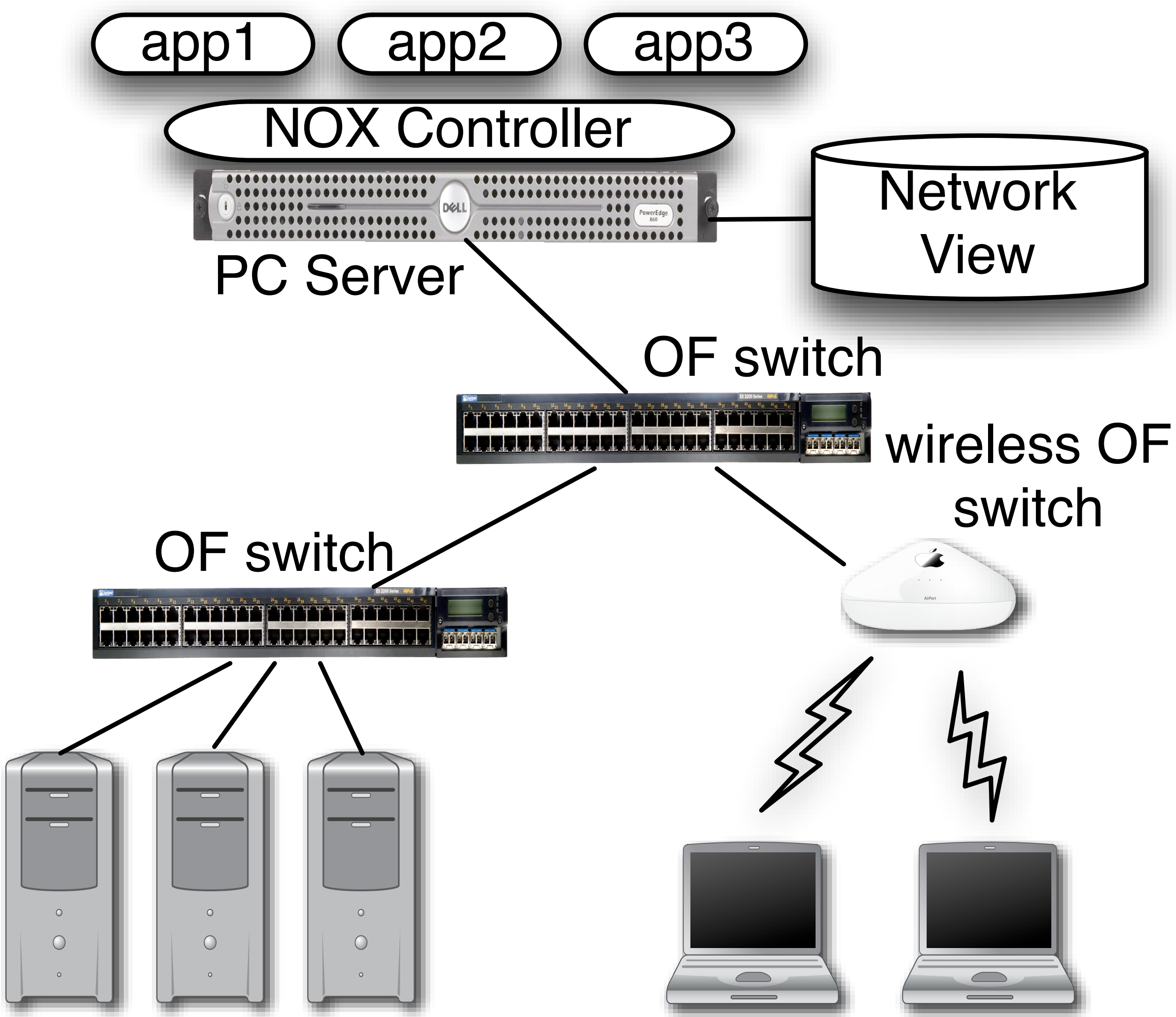
- use centralized controller to customize control

generality

- wide range of applications with diverse requirements

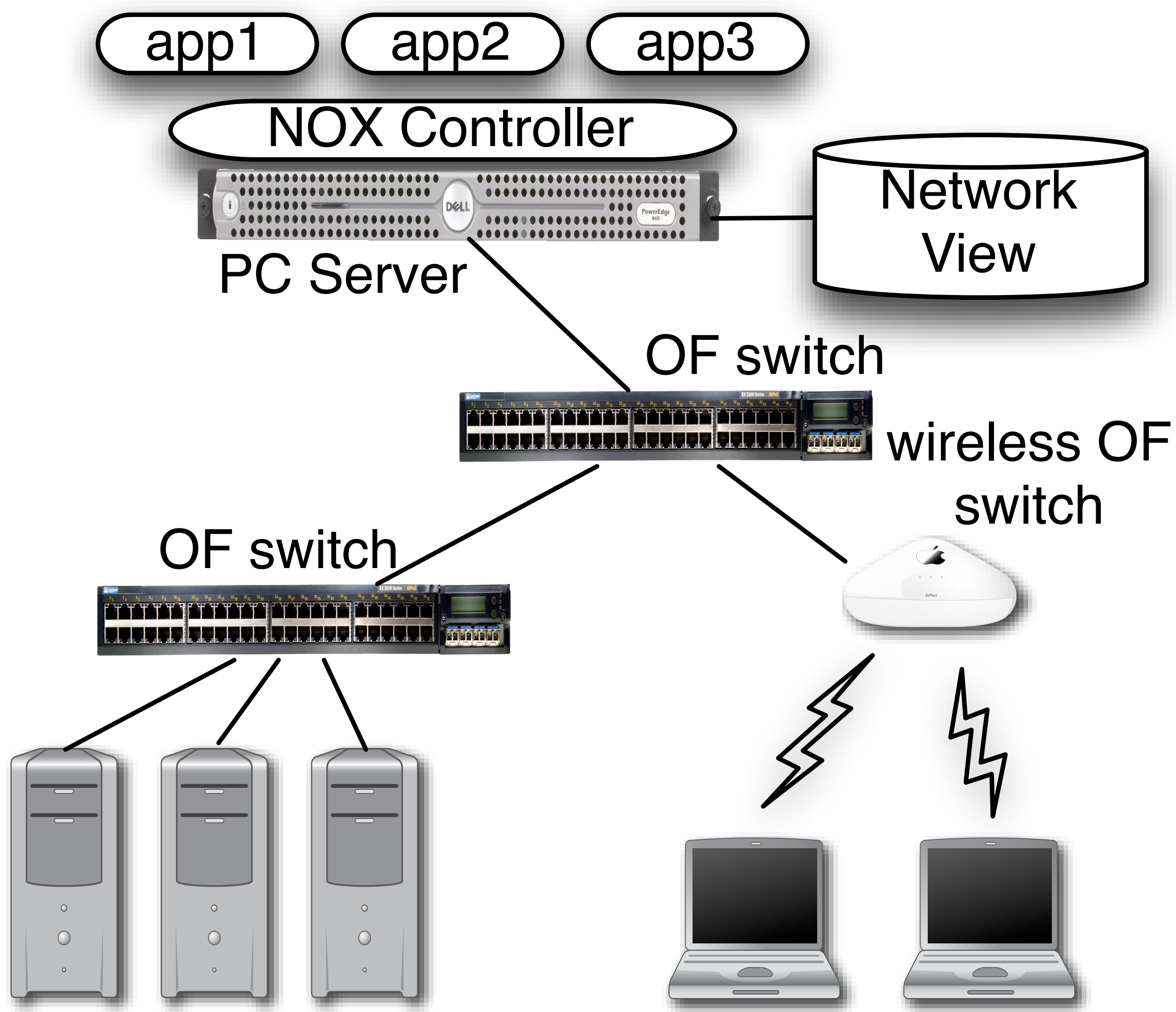
NOX

overview



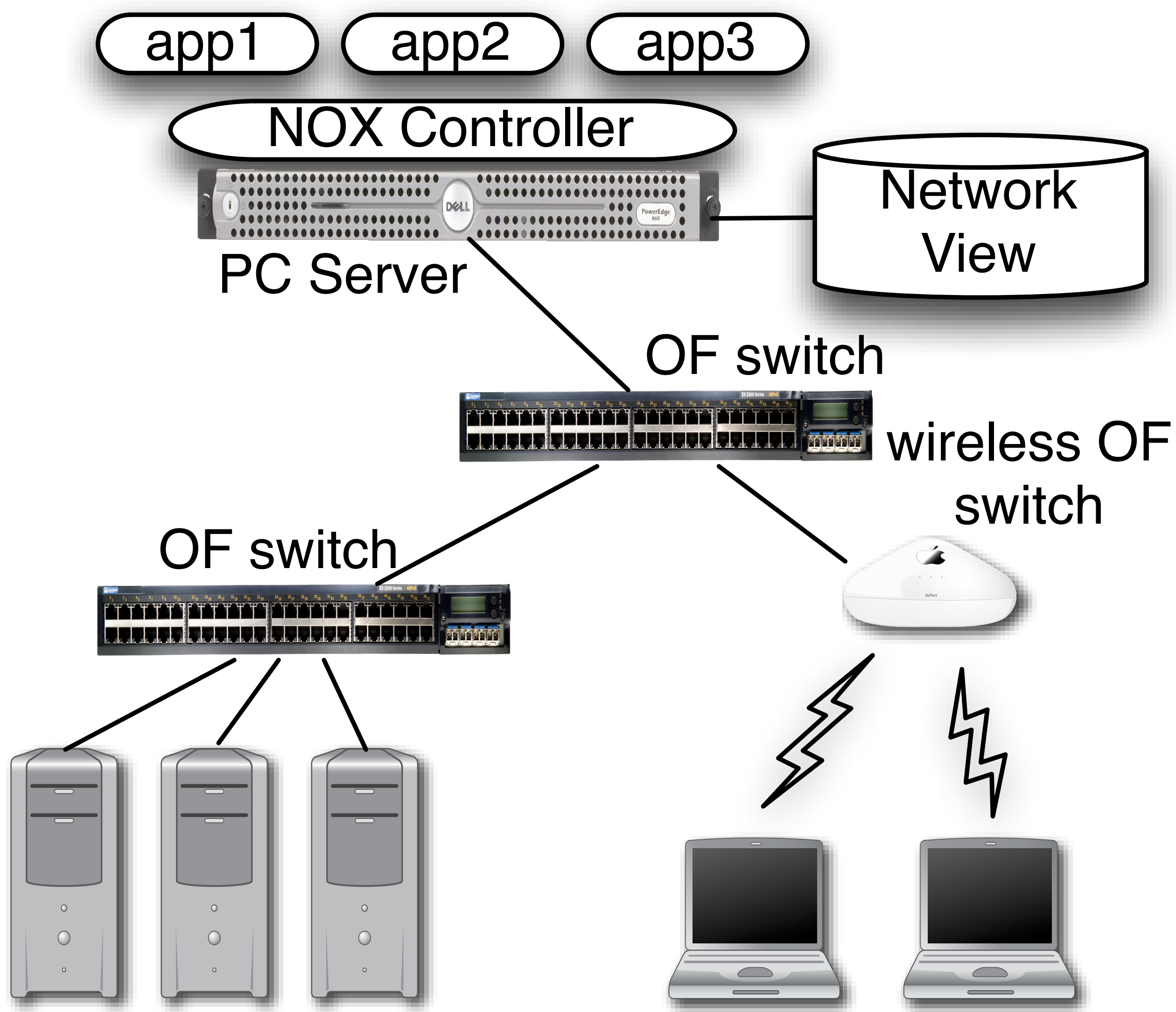
"simple switches enslaved to a logically centralized decision element"

overview



app1
app2
app3
NOX Controller
Network View
PC Server
OF switch
wireless OF switch
OF switch

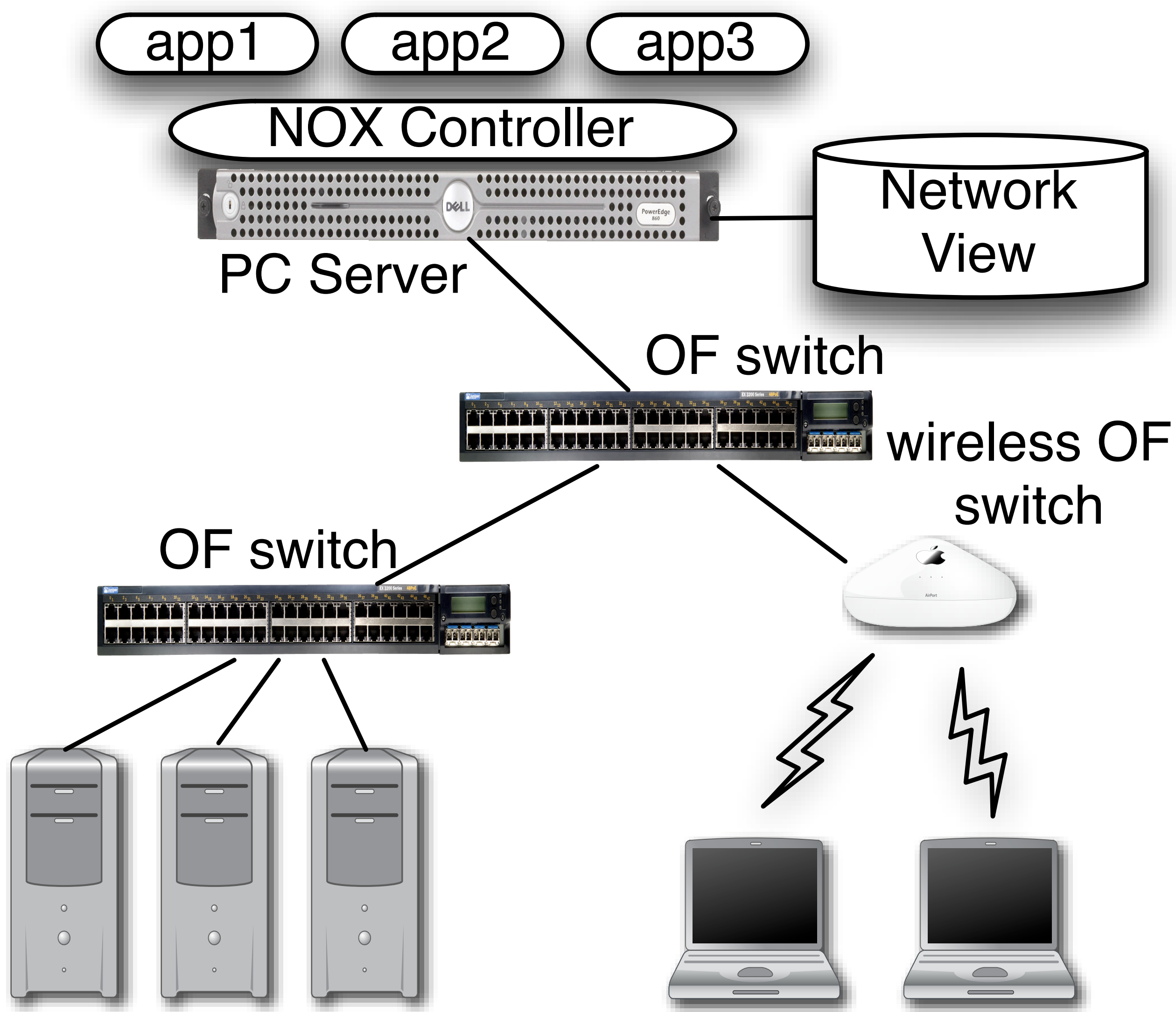
overview



network view

- topology
- locations of network elements
  - users, hosts, services
- *NOT* include traffic

overview



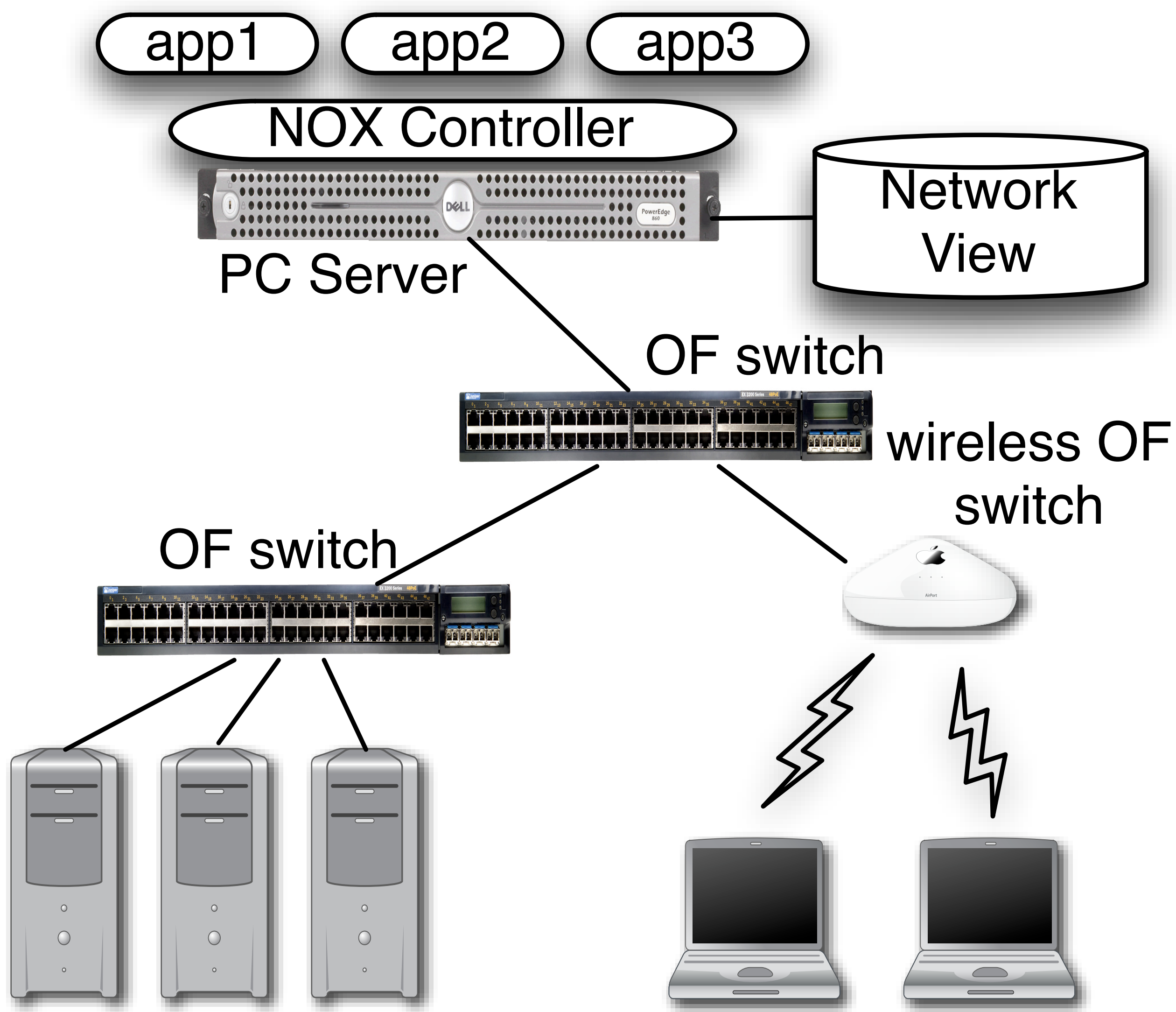
network view

- topology
- locations of network elements
  - users, hosts, services
- *NOT* include traffic

granularity

- flow based

overview



network view

- topology
- locations of network elements
  - users, hosts, services
- *NOT* include traffic

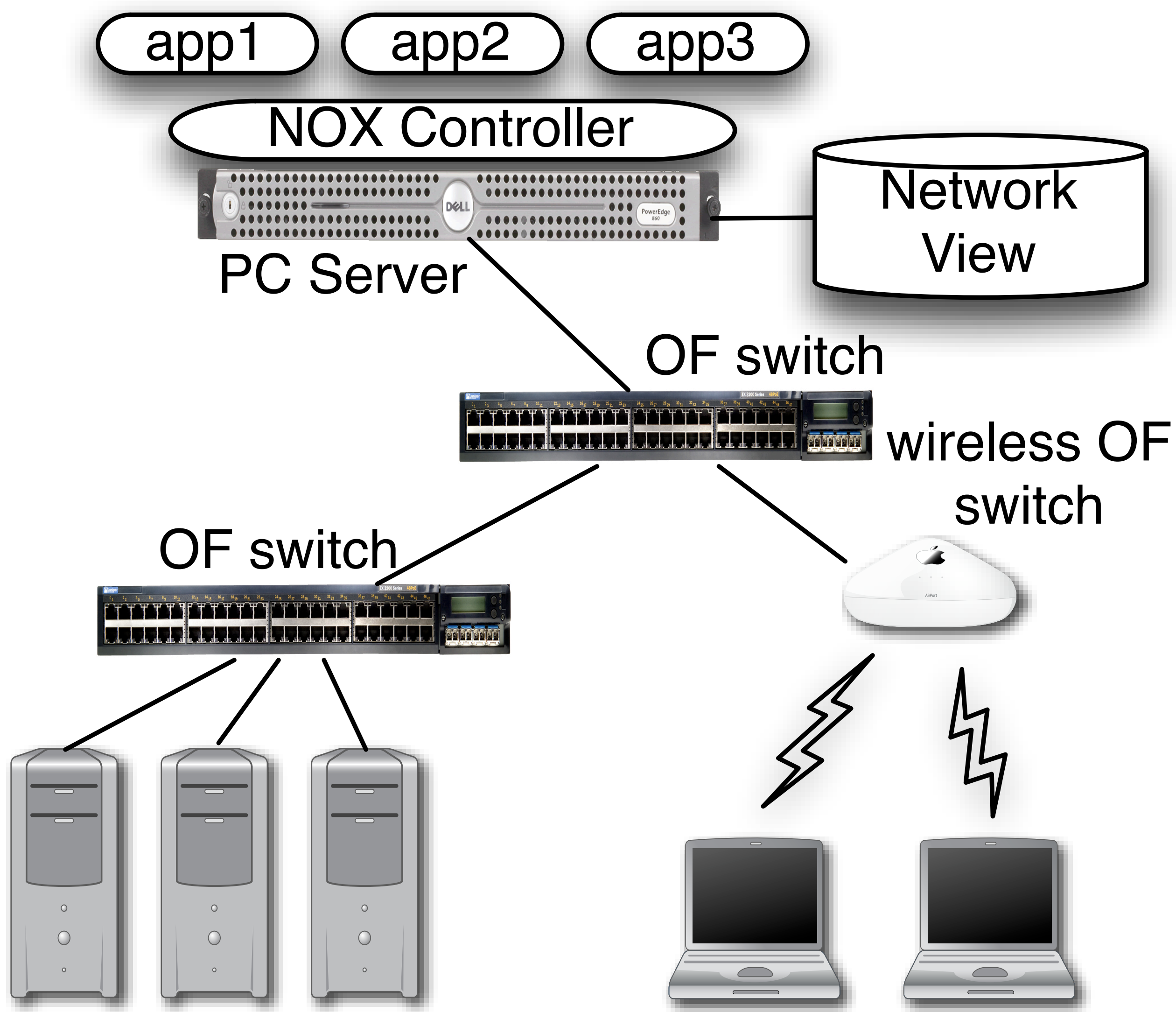
granularity

- flow based

switch abstraction

- OpenFlow
  - flow table
    <pattern: counter, actions>

overview



NOX extends Ethane

- scaling to large systems
- allowing general programmatic control

programmatic interface

events

- event handler, executed in order of its priority
  - applications register event handler

network view and namespace

- maintained by “base” applications
  - user, host authentication
- enables topology independent management applications

control

- exert through OpenFlow

programmatic interface — limitation

discussion

scalability

differing timescales and consistency requirements

	packet arrival	**flow initiation**	**changes in network view**
timescales	millions per second (10 gbps link)	one or more orders of magnitude less	tens of events per second

scalability

differing timescales and consistency requirements

	packet arrival	flow initiation	changes in network view
timescales	millions per second (10 gbps link)	one or more orders of magnitude less	tens of events per second
consistency	local storage (switches, controller instances)		global, consistency across controller instances

scalability

differing timescales and consistency requirements

	packet arrival	flow initiation	changes in network view
timescales	millions per second (10 gbps link)	one or more orders of magnitude less	tens of events per second
consistency	local storage (switches, controller instances)		global, consistency across controller instances

NOX can use parallelism

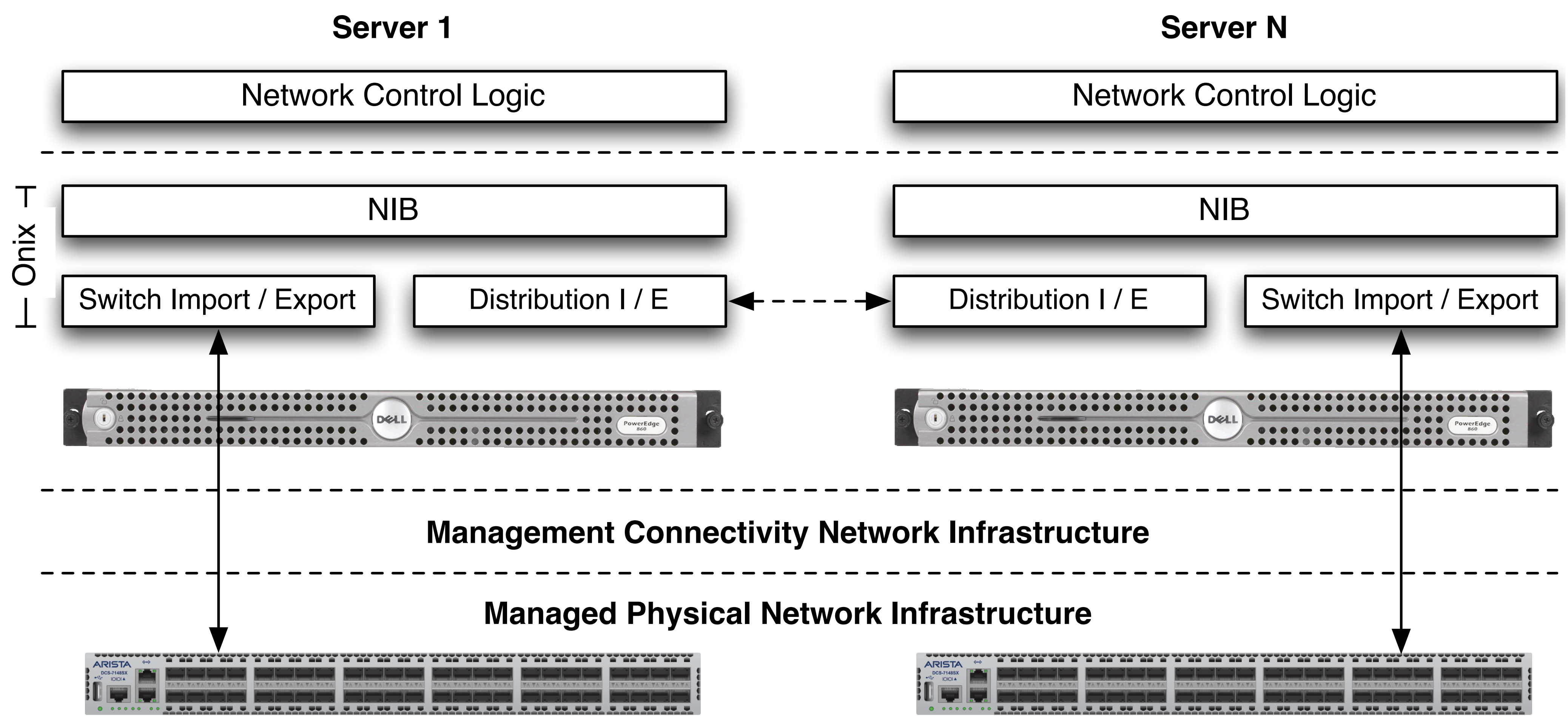
ONIX

ONIX

extends Ethane, NOX, RCP by

- far more general API
  - WAN, public cloud, data-center
- flexible distribution primitives
  - retaining performance/scalability trade-offs
  - without re-inventing distribution mechanism

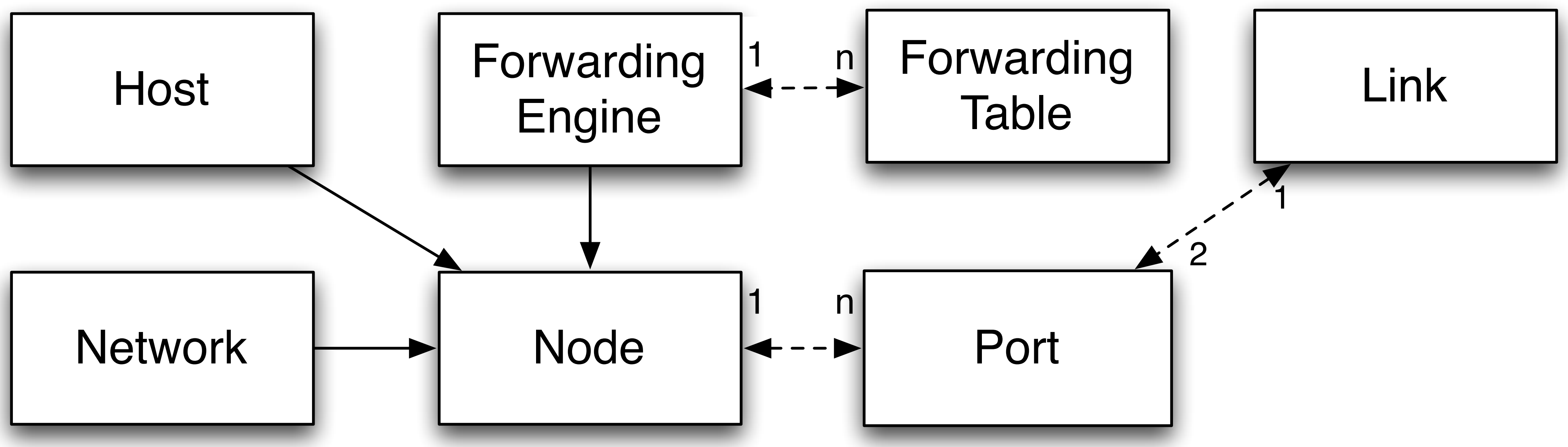
ONIX overview



Onix

- exposes unified view, disseminates network state view to other instances

ONIX API: NIB



NIB (network information base)

- apps (asynchronous) read, write, register notifications of changes
- Onix provides replication distribution
- apps provide conflict resolution, dictates consistency

scalability

scalability

goal

- *NIB not exhaust memory, events not saturate CPU*

scalability

goal

- *NIB not exhaust memory, events not saturate CPU*

mechanisms

- partitioning
  - each ONIX instance handles a subset of network (workload)
- aggregation
  - an ONIX instance exposes a subset of the NIB as a single logical entry to other instances
- consistency & durability
  - durability, strong consistency $\longleftarrow$ transactional database
  - volatile state $\longleftarrow$ memory based one-hop DHT

reliability

network element and link failures

- control logic steers traffic around the failures

reliability

network element and link failures

- control logic steers traffic around the failures

ONIX failures

- running instances detect failed node and take over
- multiple instances simultaneously manage each network element

reliability

network element and link failures

- control logic steers traffic around the failures

ONIX failures

- running instances detect failed node and take over
- multiple instances simultaneously manage each network element

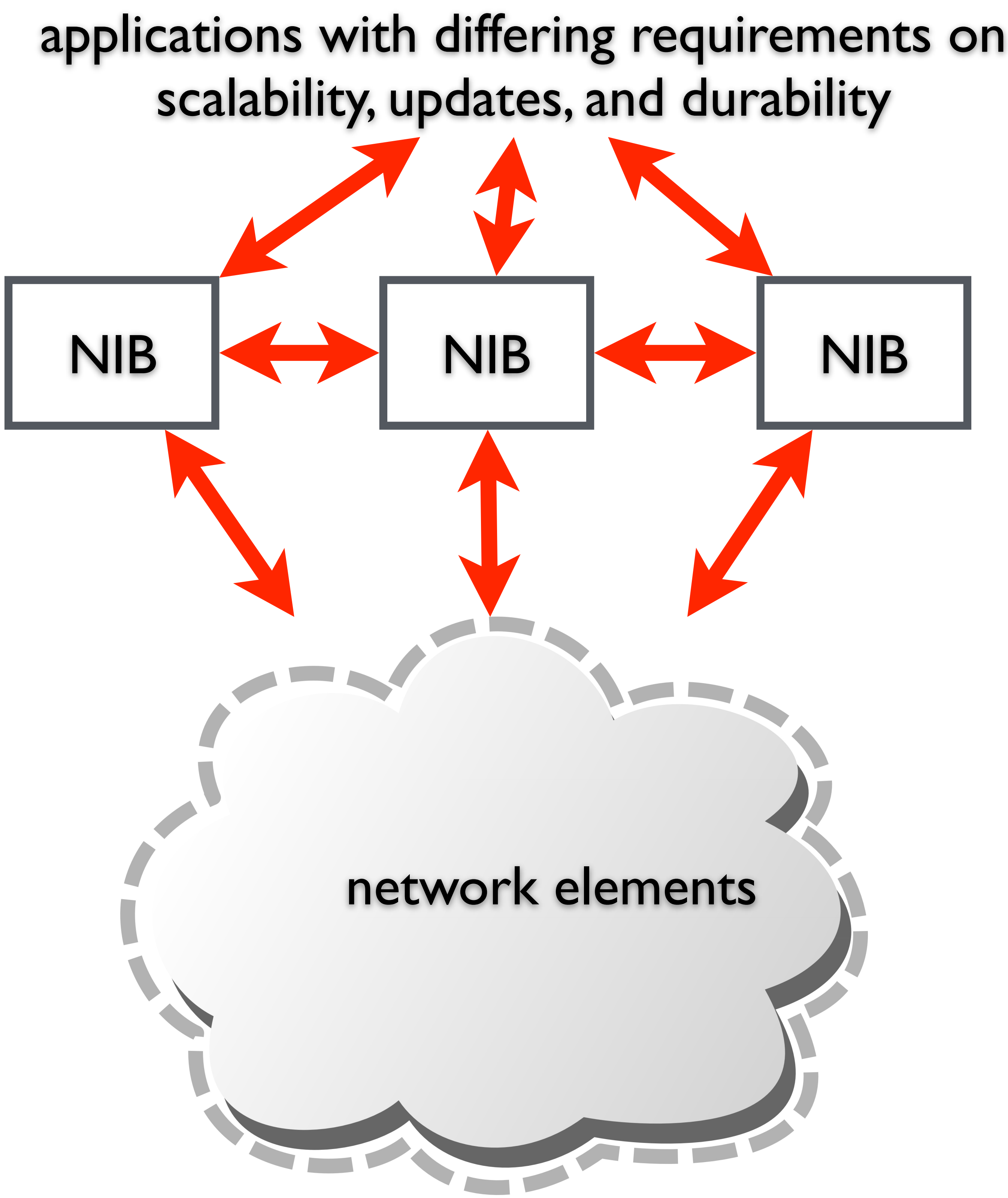
connectivity failures

- use the management network for control traffic, isolating from forwarding plane disruption

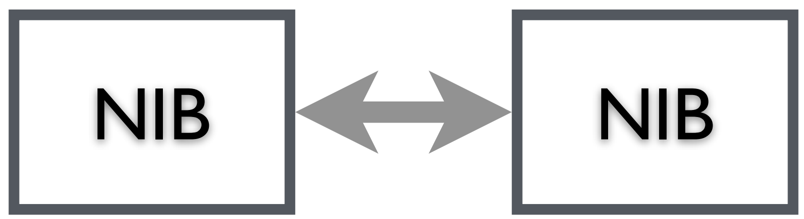
scalability & reliability

enabling mechanism: distributing NIB

distributing the NIB



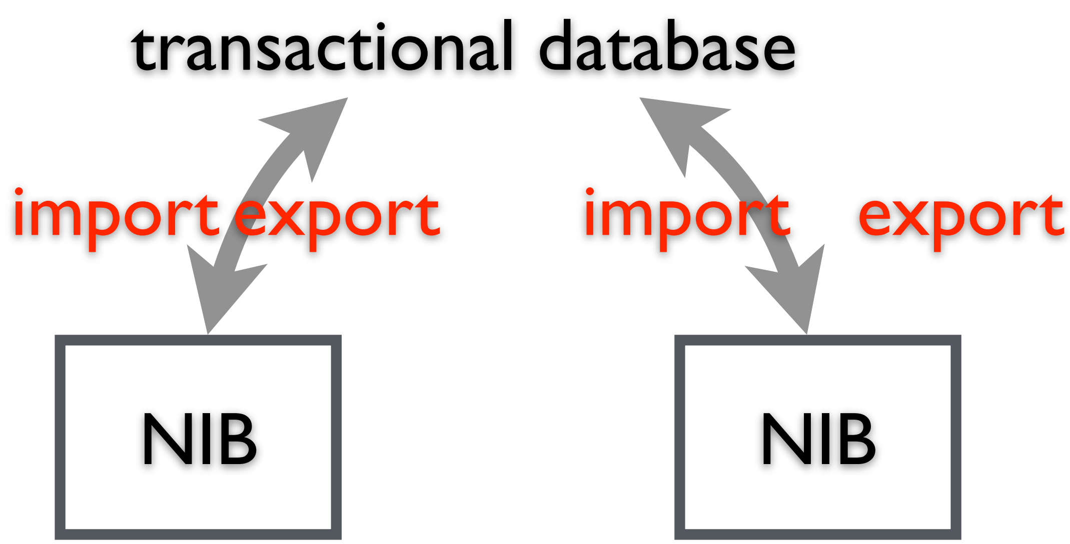
distributing the NIB



between ONIX instances

- transactional database
- DHT and soft-state trigger

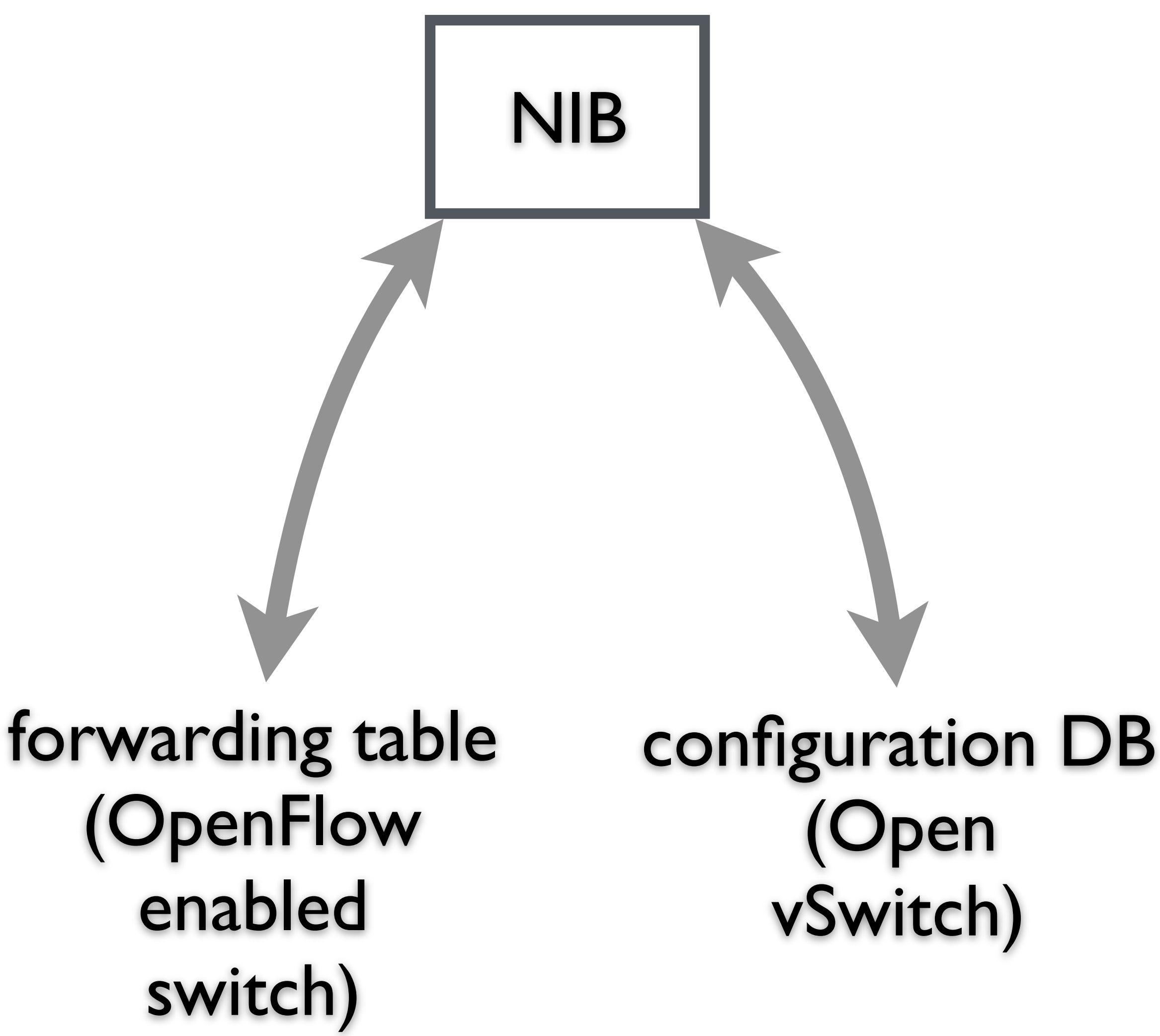
distributing the NIB



between ONIX instances

- transactional database
- DHT and soft-state trigger

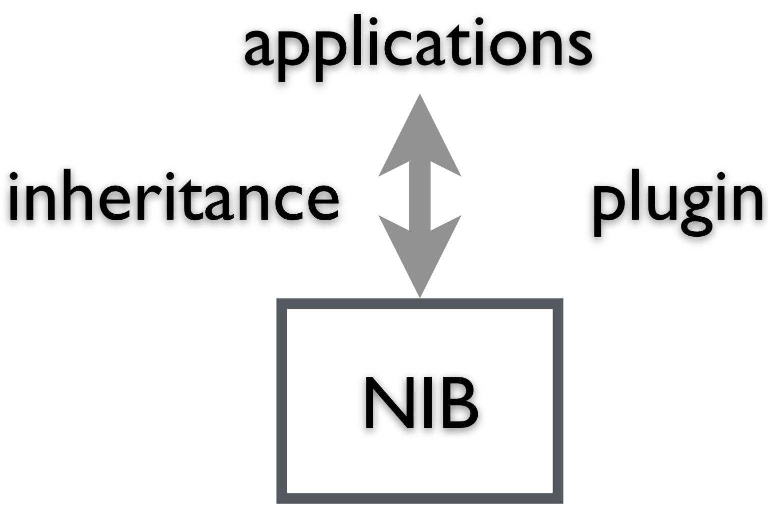
distributing the NIB



between ONIX and network

- OpenFlow
- SQL

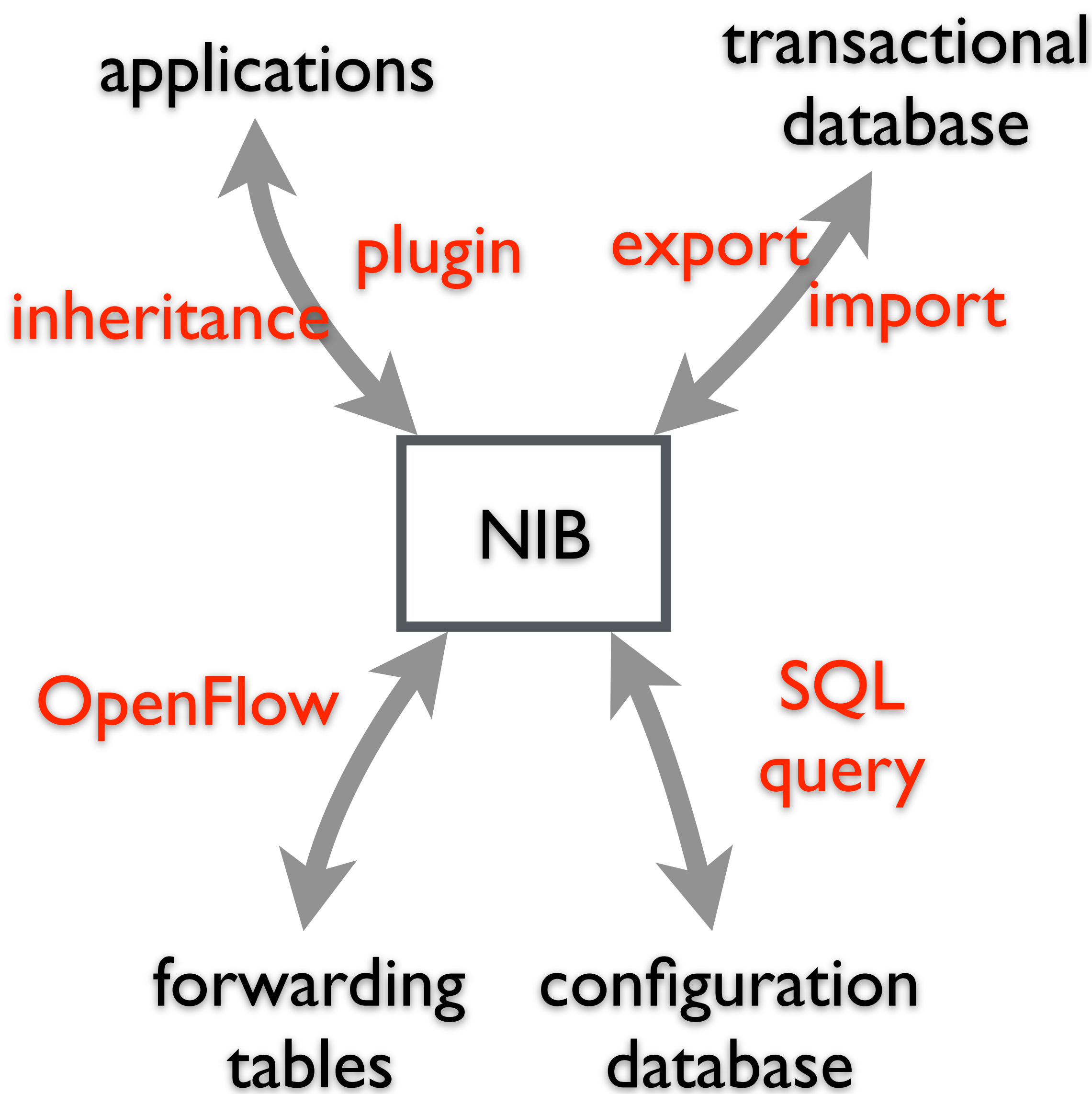
distributing the NIB



application-dependent conflict resolution

- by inheritance
  - applications inherit referential inconsistency detection
- by plugins
  - applications pass to import/export modules implement inconsistency resolution logic

summary: distributing the NIB



NIB

- the central integration point
- multiple data sources
  - ONIX instances
  - applications
  - network elements

evaluation

goals:

- Onix's performance as a general platform
- end-to-end performance of an Onix app

scalability benchmarks

- single-node: NIB throughput, memory, bandwidth
- multi-node: DHT/database throughput

reliability benchmarks

- failures: link, switch, Onix
- perceived application test: in the face of switch hosting tunnel fails, how quickly an application re-creates new tunnels

evaluation

goals:

- Onix's performance as a general platform
- end-to-end performance of an Onix app

scalability benchmarks

- single-node: NIB throughput, memory, bandwidth
- multi-node: DHT/database throughput

reliability benchmarks

- failures: link, switch, Onix
- perceived application test: in the face of switch hosting tunnel fails, how quickly an application re-creates new tunnels

evaluation — NIB throughput

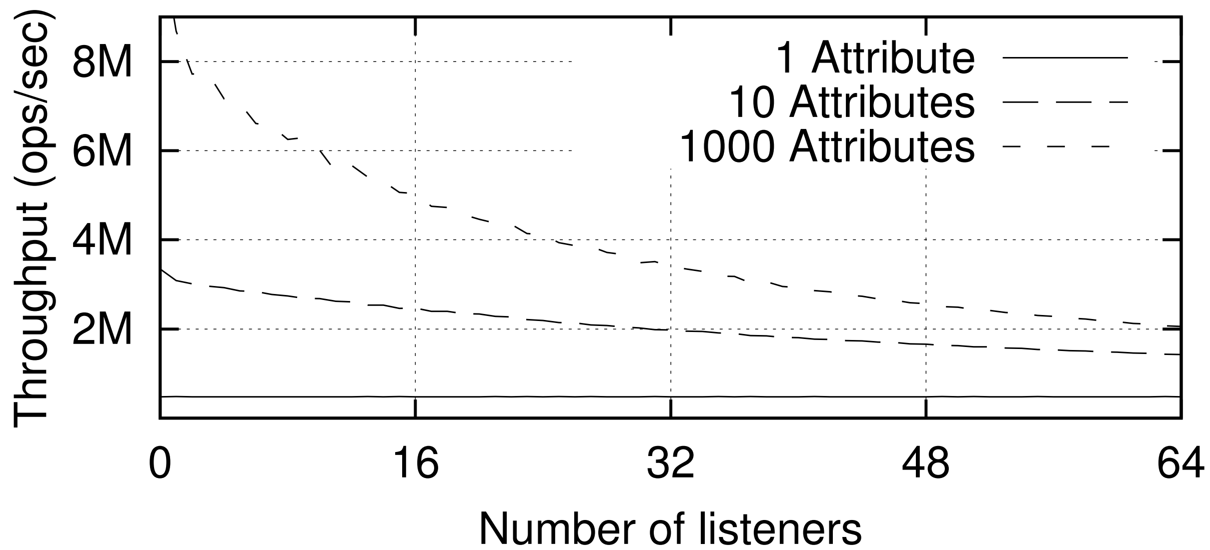


Figure 3: Attribute modification throughput as the number of listeners attached to the NIB increases.

repeatedly acquire access to the NIB

- immediately release access
- not act notifications from the NIB about changes
- acquiring exclusive access translates a context switch

effective throughput

- varying network size and attributes
- increases as number of attributes increases

evaluation — NIB memory

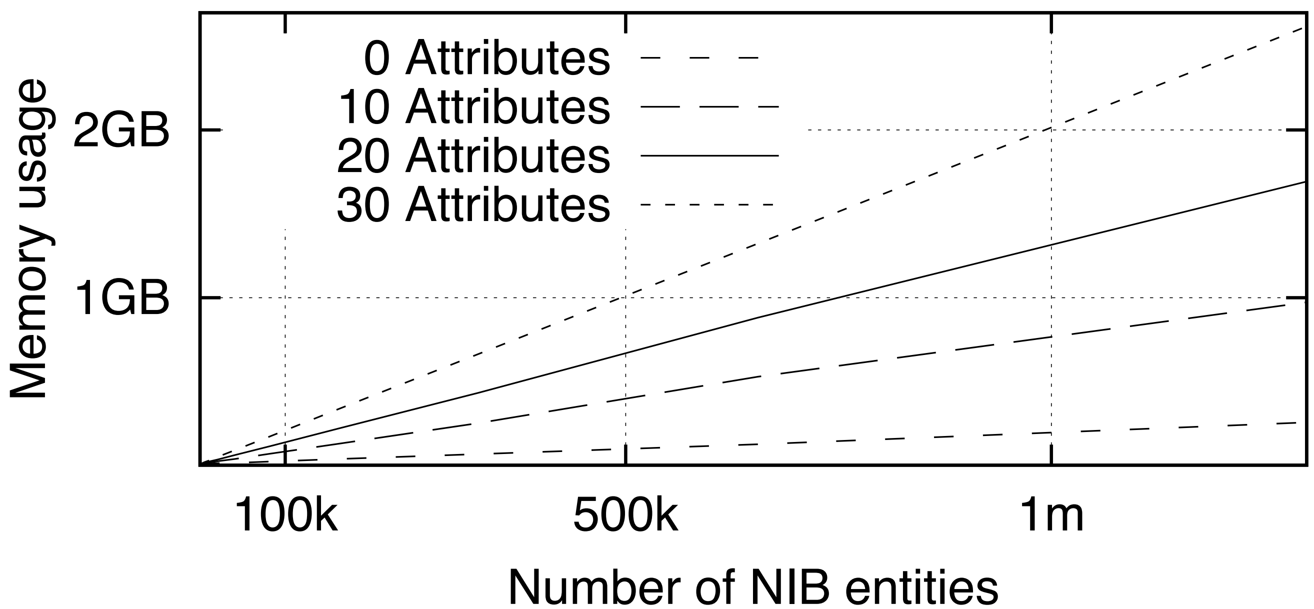


Figure 4: Memory usage as the number of NIB entities increases.

varying network size and number of attributes (per NIB)

evaluation — NIB memory

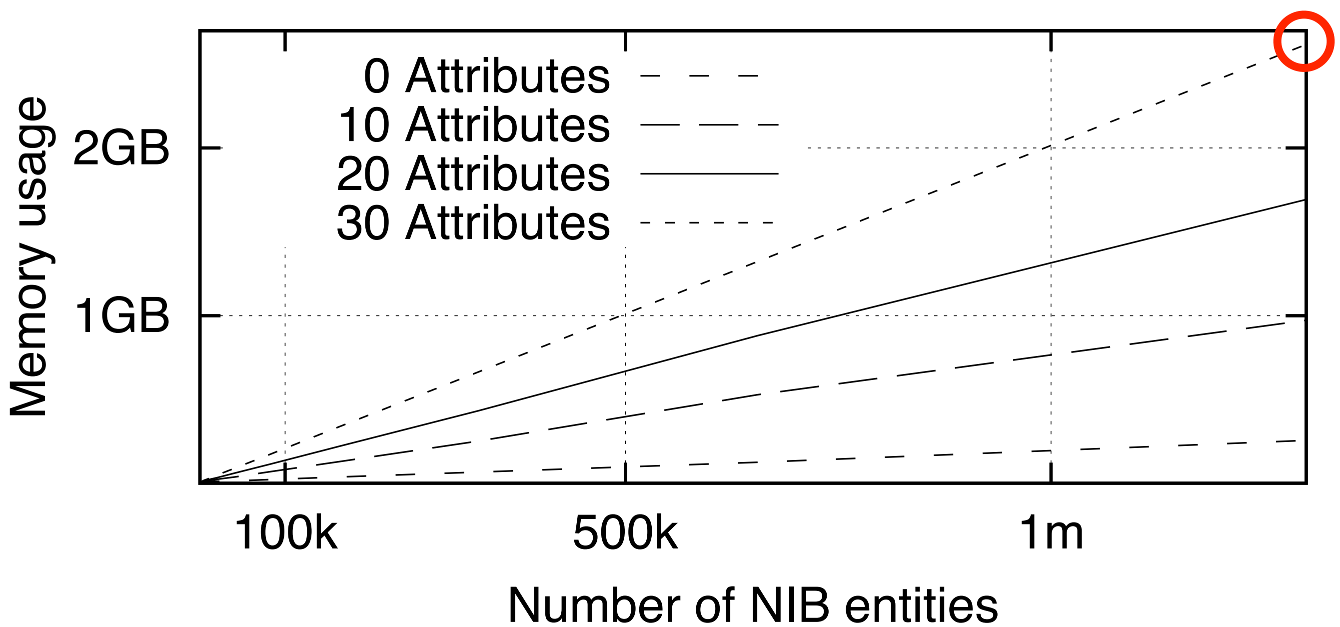


Figure 4: Memory usage as the number of NIB entities increases.

varying network size and number of attributes (per NIB)

- a single Onix instance can comfortably handle millions of entities

evaluation — bandwidth

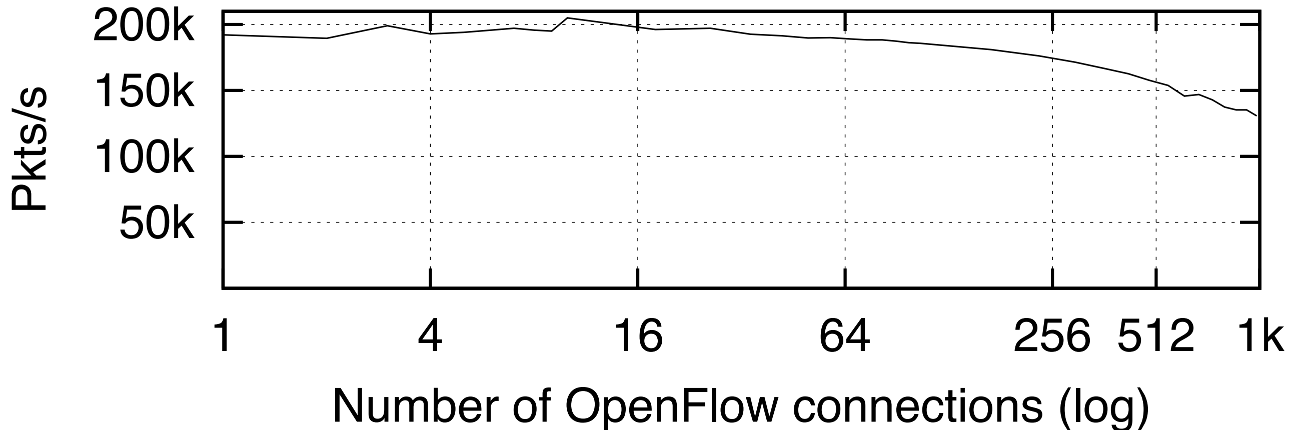


Figure 5: Number of 64-byte packets forwarded per second by a single Onix node, as the # of switch connections increases.

Onix (app) sends forwarding decision to a random switch

- benchmarks the performance of the OpenFlow stack

evaluation — bandwidth

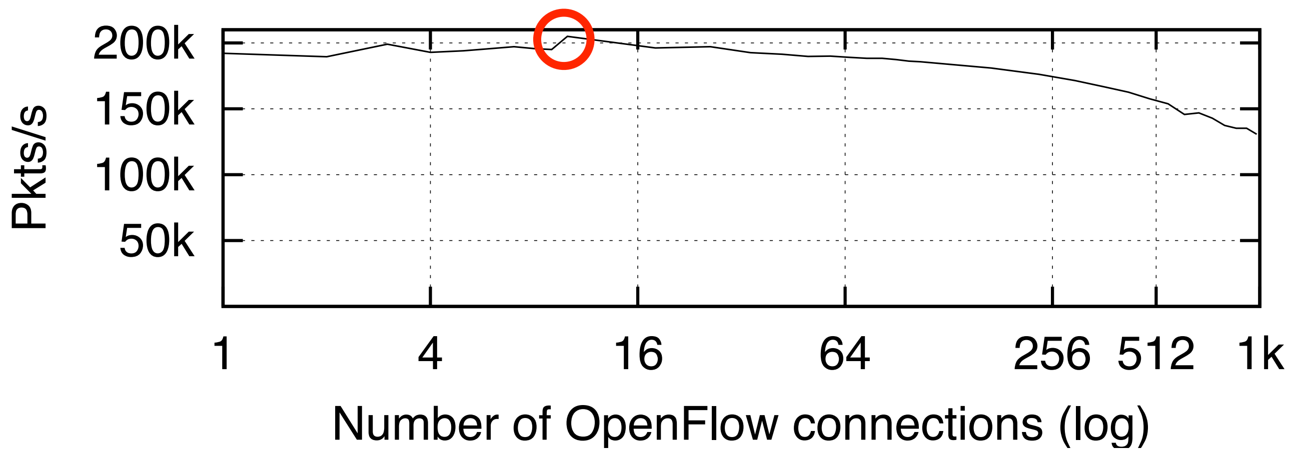


Figure 5: Number of 64-byte packets forwarded per second by a single Onix node, as the # of switch connections increases.

Onix (app) sends forwarding decision to a random switch

- benchmarks the performance of the OpenFlow stack
- bumps due to OS scheduling the controller process over multiple CPU cores

recap: opportunities and challenges

performance

- low control-plane latency

scalability

- large topology, huge volume of events, flow initiations

reliability

- handle equipment (and other) failover gracefully

simplicity

- use centralized controller to customize control

generality

- wide range of applications with diverse requirements

summary

opportunities

	Ethane	RCP	NOX	ONIX
simplicity	✓	✓	✓	✓
API			✓	✓

challenges

	Ethane	RCP	NOX	ONIX
scalability		✓		✓
reliability		✓		✓
performanc		✓	✓	✓

discussion

NOX

- inter-application coordination and isolation

Onix

- a single "application" addressing several issues
- the control platform is not designed for multiple apps to control the network simultaneously