BGP Analysis and Modeling

Bruno Quoitin
IP Networking Lab
Computer Science and Engineering Dept.
Université catholique de Louvain, Belgium
(**bruno.quoitin@uclouvain.be**)

TOTEM

RÉGION WALLONNE

france telecom R&D

Agenda

- Introduction
- BGP Modeling
- C-BGP
- Modeling an ISP
- Modeling the Internet
- Conclusion

Agenda

- **Introduction**
- BGP Modeling
- C-BGP
- Modeling an ISP
- Modeling the Internet
- Conclusion

Introduction

- **Analyzing and modeling BGP**
  - Goals
    - Better understand how BGP works
    - How it can be controlled
    - How it can be improved
    - Detect faulty behaviors, ...
  - How ?
    - Get some hands on real BGP data
    - Review BGP data formats and analysis tools available
      - Sources of public BGP data
    - Understand BGP modeling approaches
    - Build our own BGP models using C-BGP

Agenda

- Introduction
- **BGP Modeling**
- C-BGP
- Modeling an ISP
- Modeling the Internet
- Conclusion

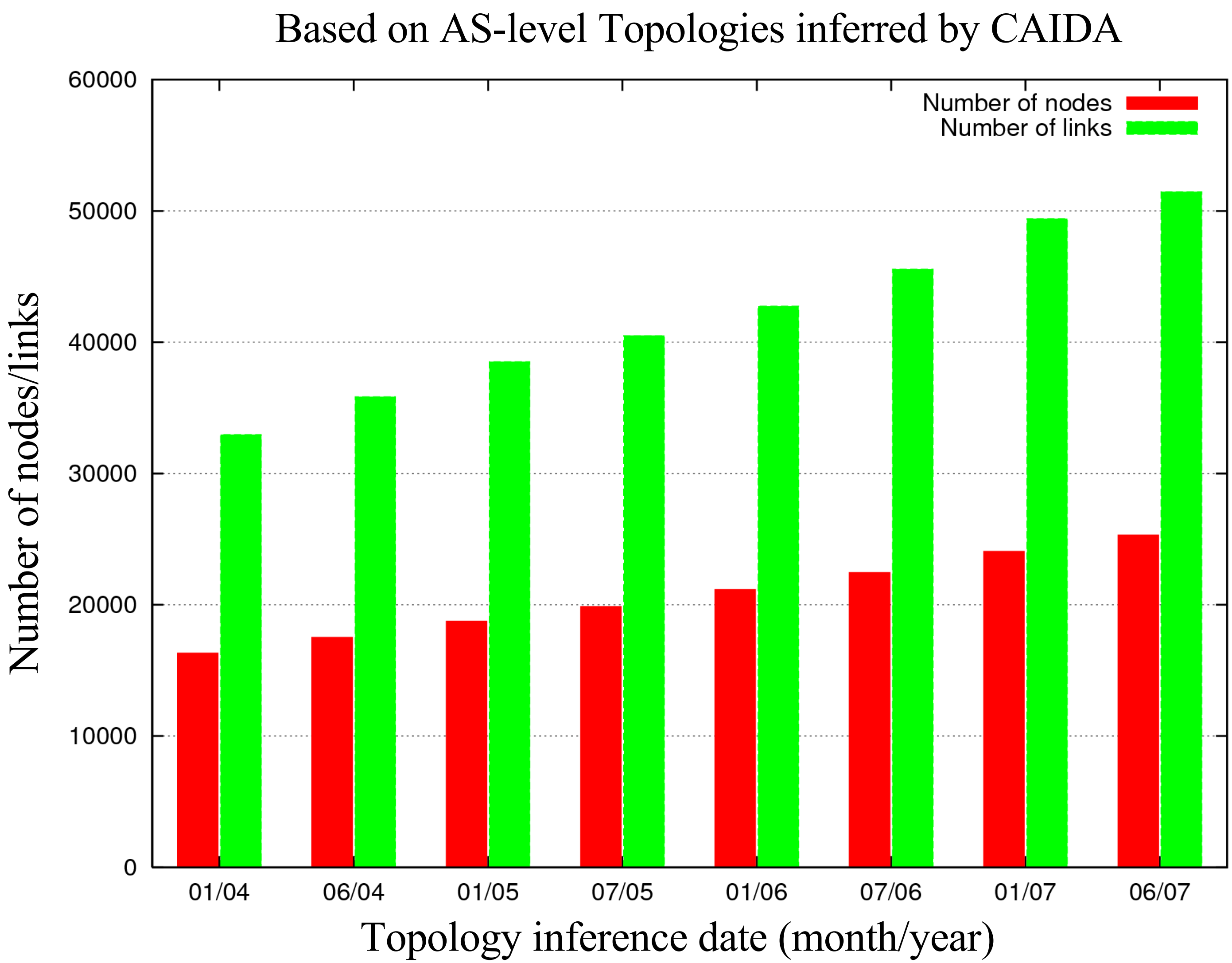
Why Modeling BGP ?

- **For the researcher**
  - To get my PhD degree :-)
  - Study macroscopic behavior of interdomain routing
  - Change BGP operation (decision process, attributes)
  - Not possible to experiment in real Internet (don't disrupt operational Internet)
  - Have only four CISCO 3640 in the lab

For the network operator

  - Understand complex interaction of IGP, BGP and traffic for thousands of destinations in a network composed of hundreds of nodes
  - Predict/evaluate impact of
    - Link/router failures
    - Routing policy changes
    - Peering changes
    - Network configuration (iBGP organization for example)

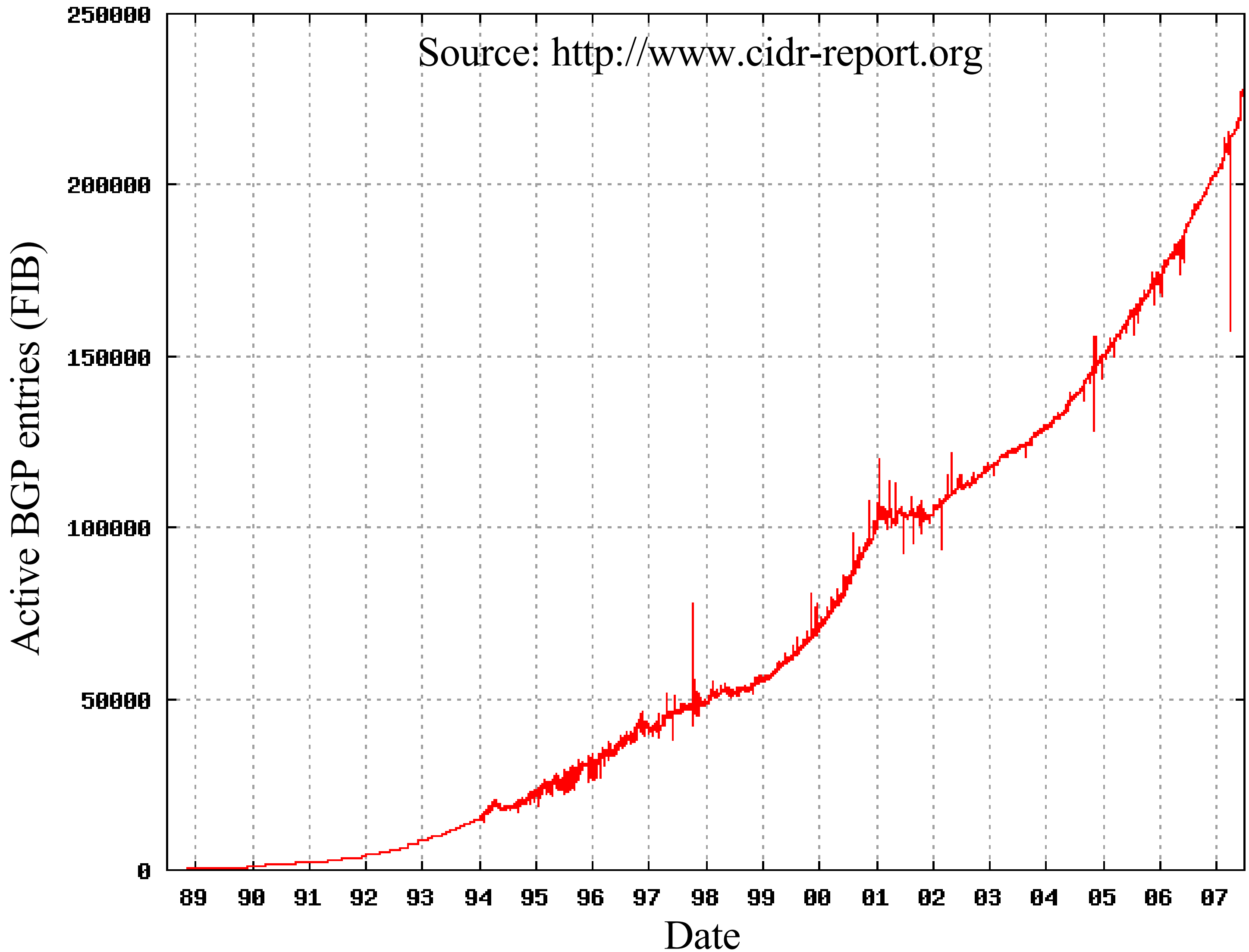
BGP Modeling Challenges



AS-level graph of the Internet:
On the order of 25.000 nodes and 50.000 links

Router-level: several orders of magnitude higher

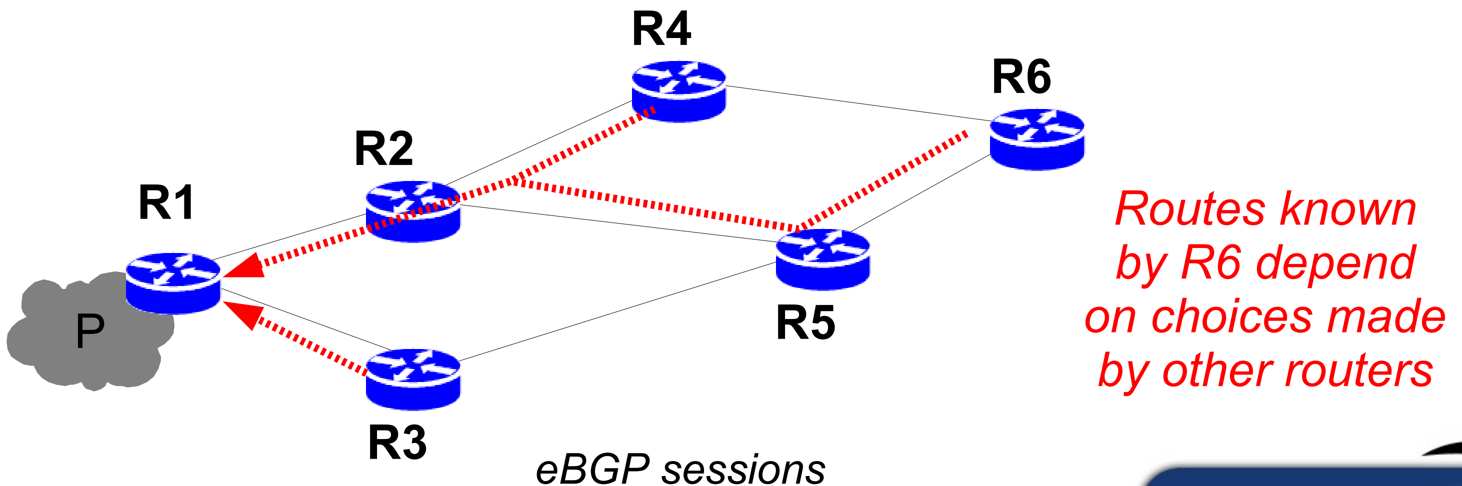
BGP Modeling Challenges



Number of destination:
More than 220,000 prefixes !

BGP Modeling Challenges

- **BGP Operations**
  - Complex by nature
  - Decision process: sequence of rules (~ lexicographic order)
  - Autonomy of Decision (no global optimization)
  - Complex routing filters (policies)
  - Path-vector protocol
    - Interdependency between decision choices
    - Differ from link-state protocols such as OSPF or IS-IS (that can be modeled using shortest-path)



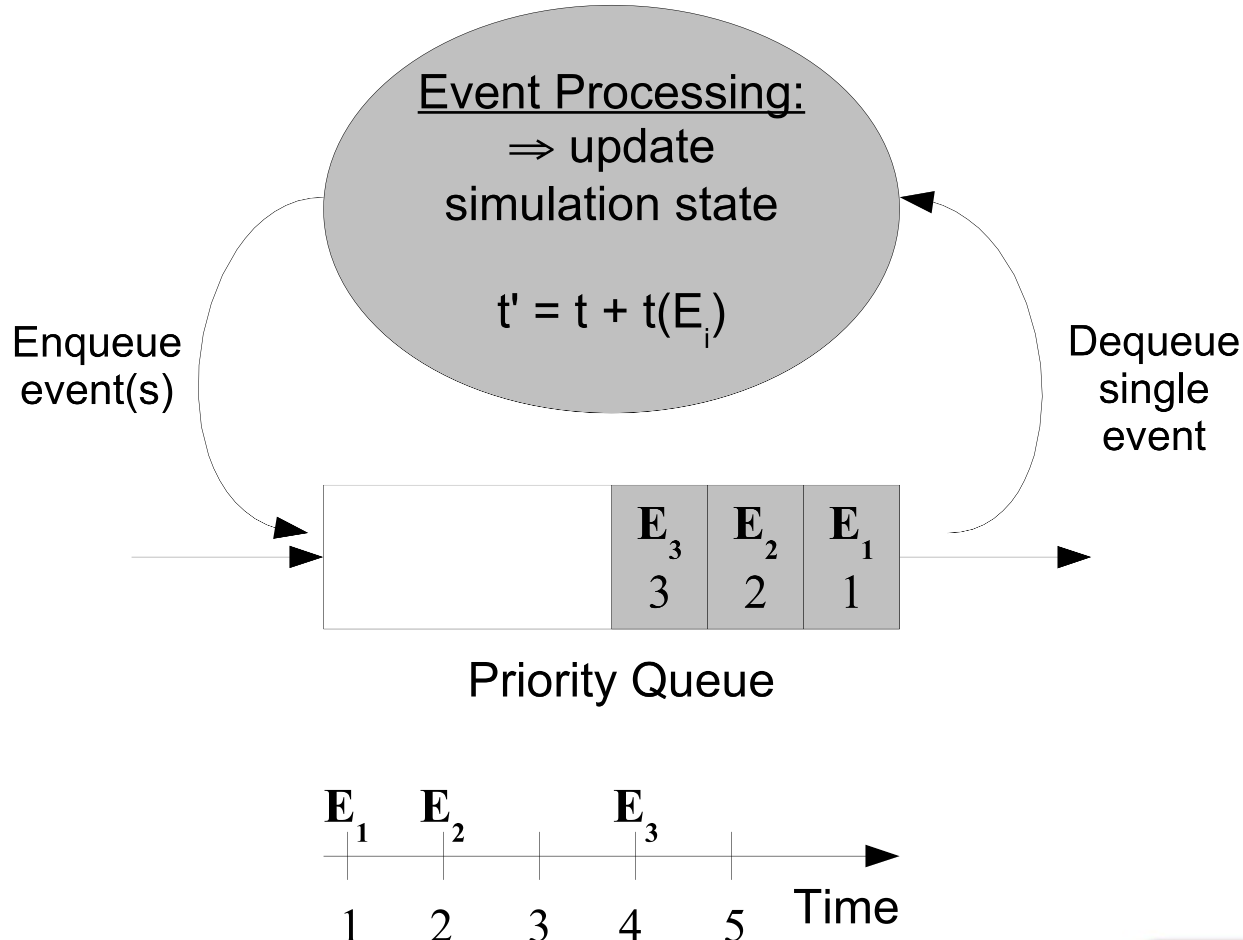
BGP Modeling Tools

- **BGP Daemons**
  - Examples: MRTd, **Zebra**, Quagga, XORP, BIRD, OpenBGPd, ...
  - Real BGP implementations that could almost be used in a production environment
  - Multiple instances can be run on a single workstation using virtualization (e.g. Netkit, VNUML)
  - **Advantages:** detailed BGP implementations
    - Full BGP decision process
    - Versatile filters
    - Mature, well-tested implementations: Zebra, Quagga
  - **Drawbacks:** too detailed
    - Work in real time
    - Every detail is implemented: full protocols states are maintained
    - Young, incomplete implementations: OpenBGPd, BIRD, ...

⇒ **Limited to simulating only a few routers**

BGP Modeling Tools

- **Discrete-Event Simulation (DES)**
  - Priority queue: usually a calendar queue



BGP Modeling Tools

- **Packet-level simulators**
  - Examples: SSFNet, ns-2, J-Sim
  - Rely on *Discrete-Event Simulation* (DES):
    - Allows skipping periods of time when nothing happens.
  - **Advantages:**
    - Faster than real implementations
    - Support other protocols such as TCP, applications, ...
    - Good for simulating protocol dynamics
  - **Drawbacks:**
    - Too detailed for BGP: need a lot of resources to perform large-scale simulations (all protocols state machines are modeled)
    - Partial, sometimes highly experimental implementations. For example, SSFNet's BGP has very simple route filtering expressiveness and does not support full decision process.

⇒ **Still limited to small topologies**

BGP Modeling Tools

- **Steady-state simulation**
  - Example: BGP Emulation (by Nick Feamster)
  - Compute outcome of BGP decision process (steady-state)
    - Solves the interdependency of routing decisions (due to path-vector) without modeling the BGP messages propagation
  - **Advantages:**
    - Faster and less memory-consuming than DES (focuses on BGP only)
  - **Drawbacks:**
    - AT&T proprietary tool (not publicly available)
    - Limited to modeling a single AS
    - Very specific approach

Feamster and Rexford, IEEE/ACM Transactions on Networking, April 2007

BGP Modeling Tools

Tool	Pros	Cons
BGP Daemons *Zebra, Quagga, OpenBGPd, BIRD, ...*	- most complete versatile filters - mature implementation (Zebra)	- work in real time - too detailed **Limit: 10s of routers**
Packet-level simulators *SSFNet, ns2, J-Sim, ...*	- skip time with no event - support for other protocols	- need a lot of resources - partial / experimental implementations **Limit: 100s of routers**
BGP emulator *from N. Feamster*	- solve dependencies bw routing choices without propagation	- proprietary tool - specialized algorithm - limited to a single domain **Limit: not available**

Agenda

- Introduction
- BGP Modeling
- **C-BGP**
- Modeling an ISP
- Modeling the Internet
- Conclusion

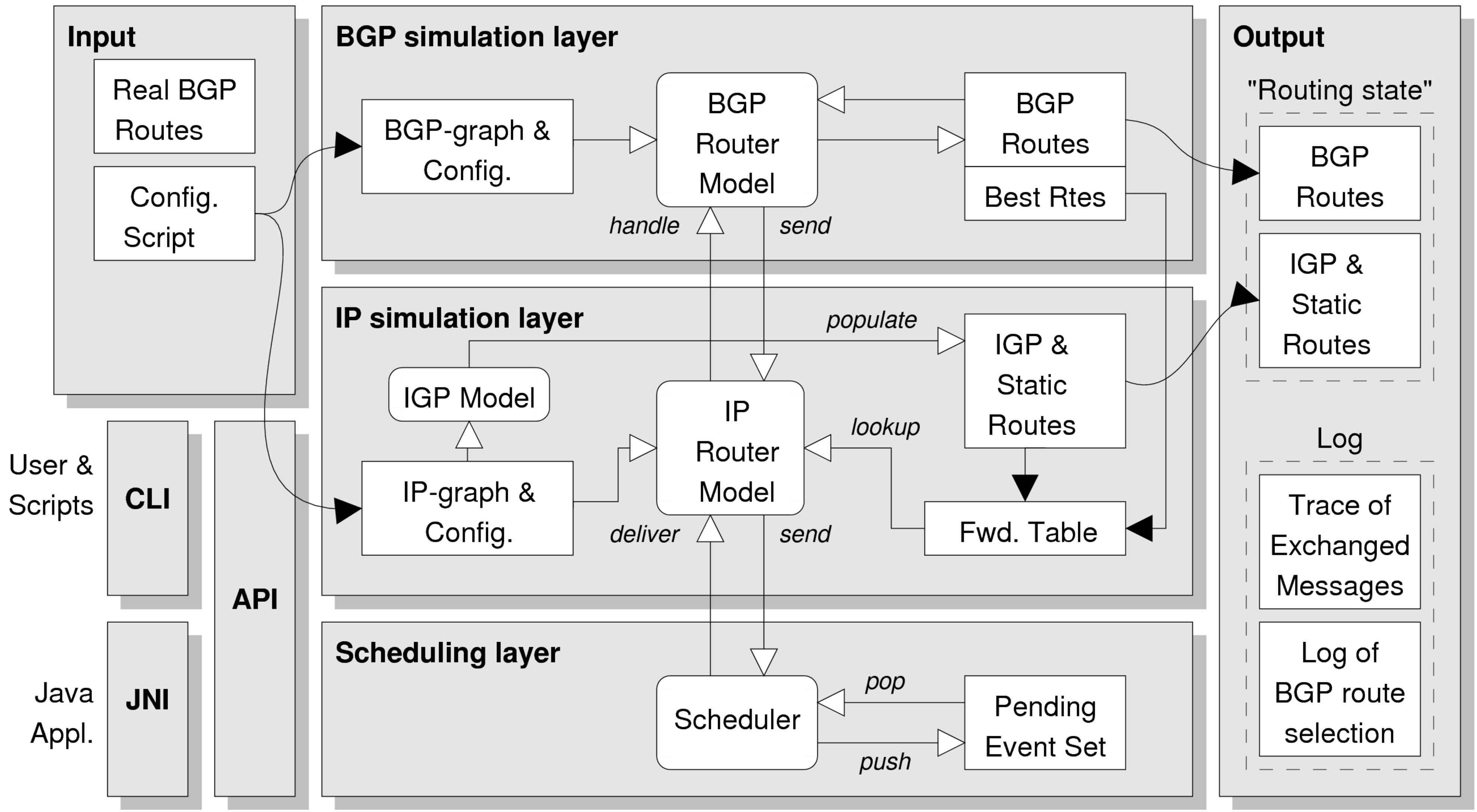
C-BGP Routing Solver

- **Timeless Event Scheduling**
  - Extend approaches of Labovitz (BGP model) and Griffin (SPVP)
  - Router-level
  - IGP routing model
  - iBGP (including Route-reflector hierarchy)
  - Full decision process
  - Versatile route filters, ...
- **Simplifications brought to BGP**
  - Do not model TCP connections
  - Do not model complete BGP's Finite State Machine (FSM)
  - Do not model BGP timers (MRAI, dampening)
  - Static IGP routing model
  - Not “event-driven”

Labovitz et al, ACM SIGCOMM, 2000

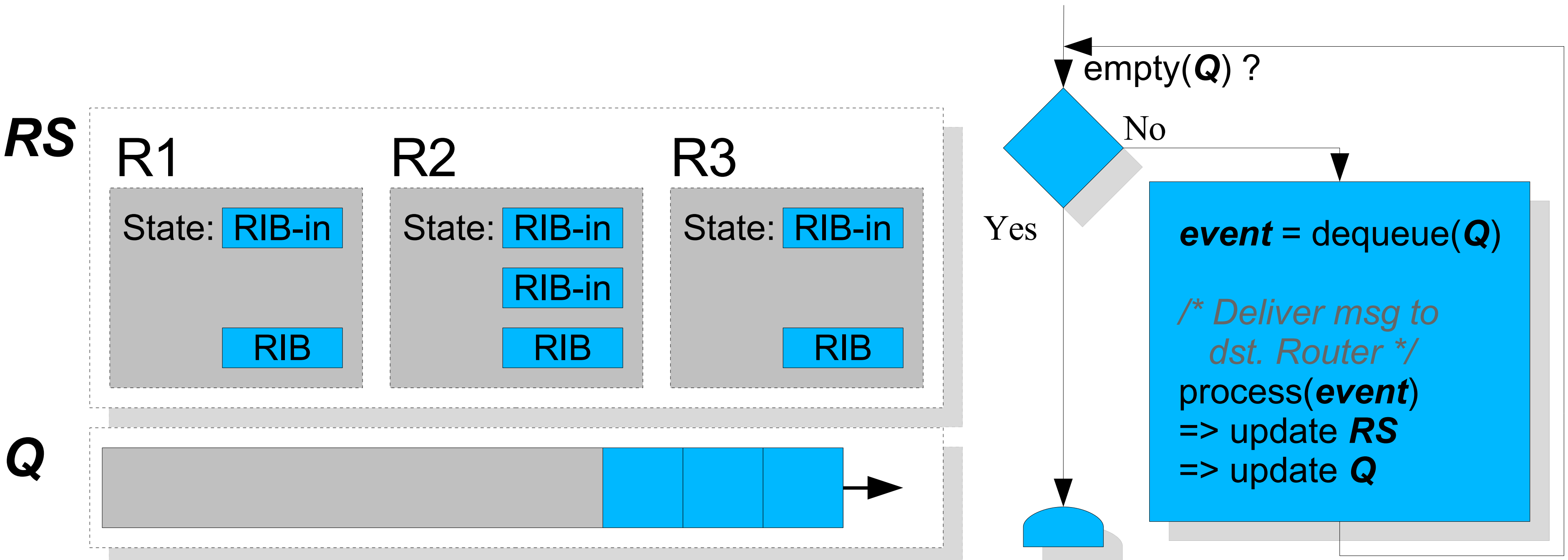
Griffin and Wilfong, IEEE INFOCOM 2000

C-BGP Architecture



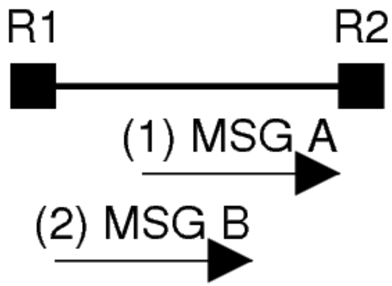
C-BGP Principles

- **Routing State and Pending Event Set**
  - C-BGP Maintains the global protocol state ***(RS, Q)***
  - ***RS***: set of all routers states, i.e. Loc-RIBs and Adj-RIB-ins
  - ***Q***: pending event set
    - Event= propagation of reachability information from Router Ri to router Rj

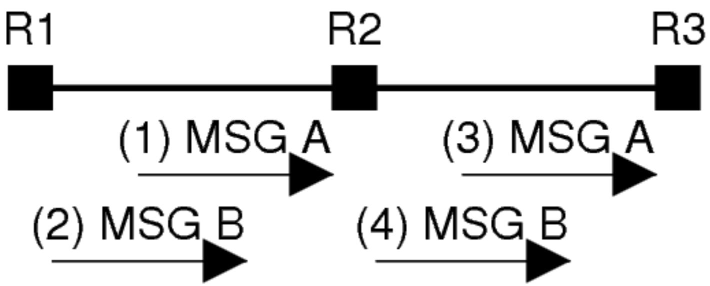


C-BGP Principles

- **Timeless event scheduling**
  - Processing of queue ***Q***
  - Model of a simplified reliable transport protocol
    - preserves the ordering of messages along an edge (A), along a path (B)
    - no need for re-transmission
  - Details of TCP not modeled



(A) along an edge



(B) along a path

C-BGP Principles

Phase 1: compute intradomain routes

$\forall$ *domain D*, $\forall$ *router R* $\in$ *D*,
  compute $SPT_R(D)$

Phase 2: compute interdomain routes

$\forall$ *R* $\in$ *Routers*
  $\forall$ *P* $\in$ *Prefixes(R)*
    $\forall$ *N* $\in$ *Neighbors(R)*
      if (out_filter_accept(*R*, *N*, *rte(P)*))
        push(***Q***, *R*→*rte(P)*→*N*)

```
/* Convergence */
while (Q ≠ Ø)
  (N→r→R)= pop(Q)
  if (in_filter_accept(R, N, r))
    r*= bgp_decision_process(R, r)
    if (changed(r*))
      ∀ N ∈ Neighbors(R),
        if (out_filter_accept(R, N, r*))
          push(Q, R→r*→N)
```

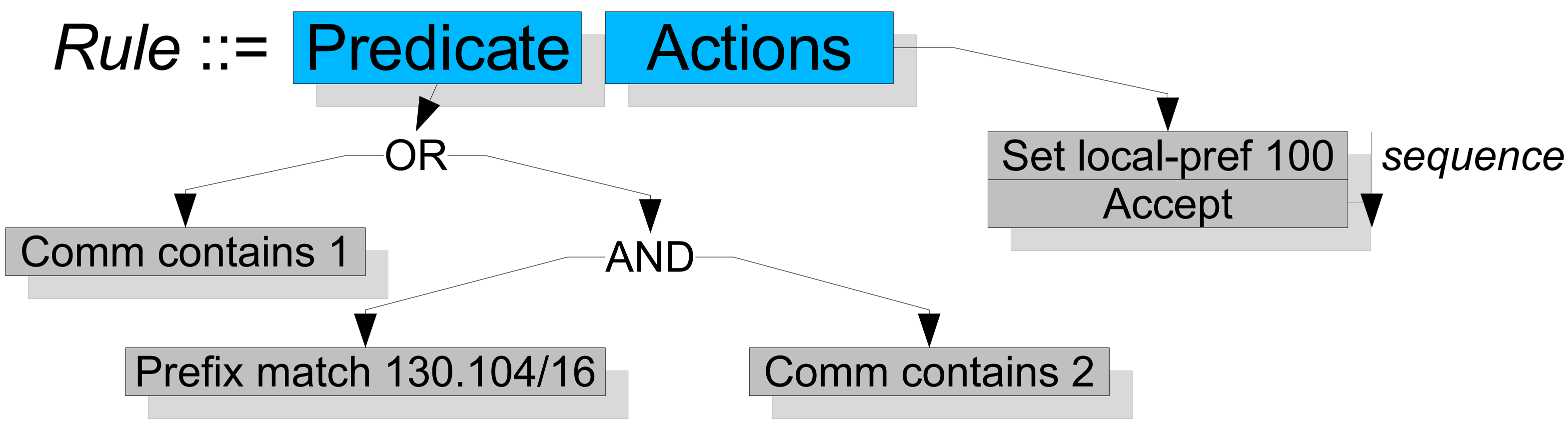
Queue ***Q***
contains events "A→*M*→*B*"
meaning *A* sends route *M* to *B*

Decision Process

1. Ignore if next-hop unreachable
2. Prefer locally originated networks
3. Prefer highest Local-Pref
4. Prefer shortest AS-Path
5. Prefer lowest Origin
6. Prefer lowest MED
7. Prefer eBGP over iBGP
8. Prefer nearest next-hop
9. Prefer lowest Router-ID or Originator-ID
10. Prefer shortest Cluster-ID-List
11. Prefer lowest neighbor address

Route Filtering

- **Route Filtering Processes**
  - Protocol Filters:
    - avoid AS-Path loop, sender-side loop detection (SSLD), avoid Cluster-ID-List loop, route-reflection rules, stateful BGP (using RIB-out)
  - Policy filters:
    - Per session input and output filters
    - Sequence of rules

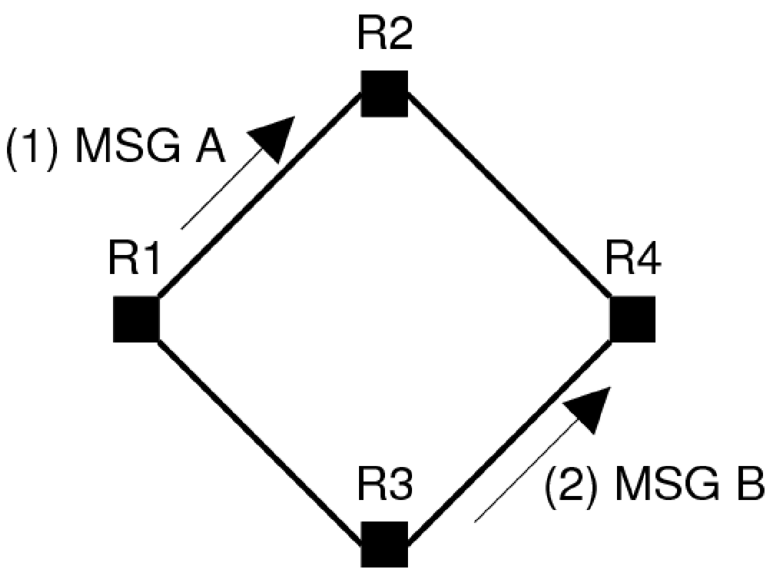


Route Filtering

- **Supported predicates**
  - Match on prefix
  - Match on Communities
  - Match on AS-Path
- **Supported actions**
  - Accept / Deny route
  - Add / Remove Community
  - Set Local-Preference
  - Set Multi-Exit-Discriminator
  - Prepend AS-Path
  - Set Redistribution Community

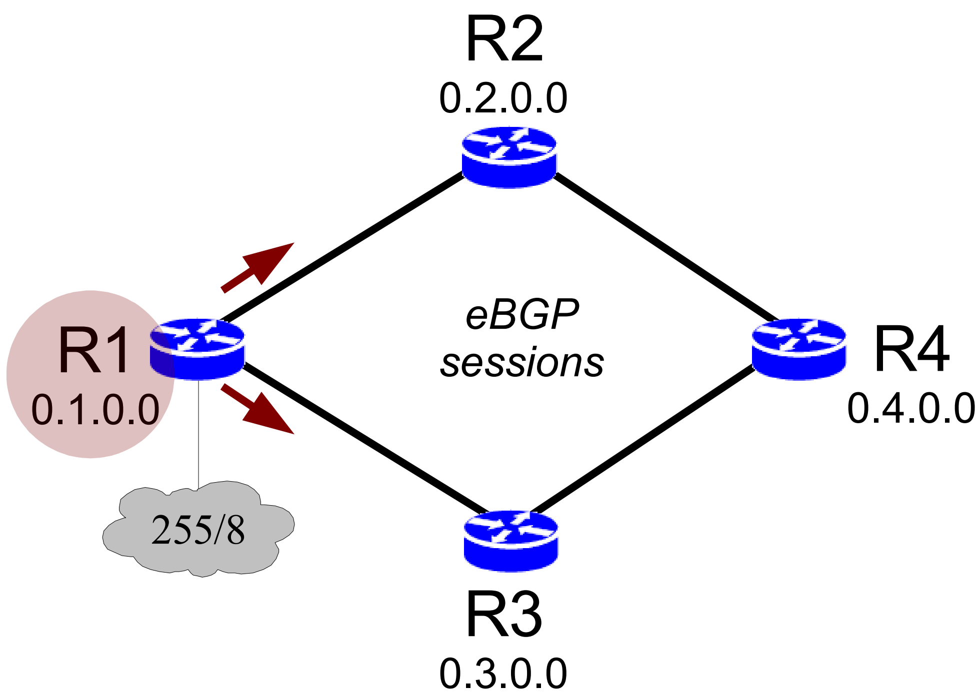
Determinism

- **Determinism**
  - Global Ordering of Messages ~ propagation delay ignored
  - Ordered dissemination of routes to neighbors
  - Deterministic Decision Process
    - No intentional non-determinism, e.g. “prefer oldest route” tie-break



C-BGP Convergence

- **Convergence Example**



Queue:

R1→R2: 255/8, {1}
R1→R3: 255/8, {1}

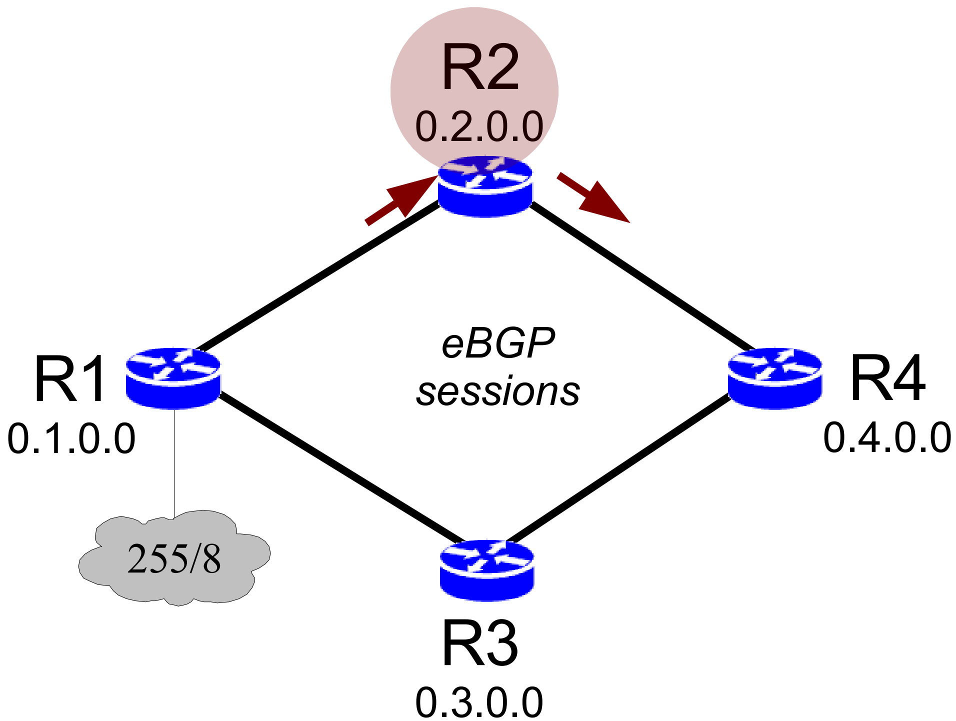
Simulation time: 0

Routing State:

R1:	**255/8, local**
R2:	
R3:	
R4:	

C-BGP Convergence

- **Convergence Example**



Queue:

R1→R3: **255/8, {1}**
R2→R4: 255/8, {2 1}

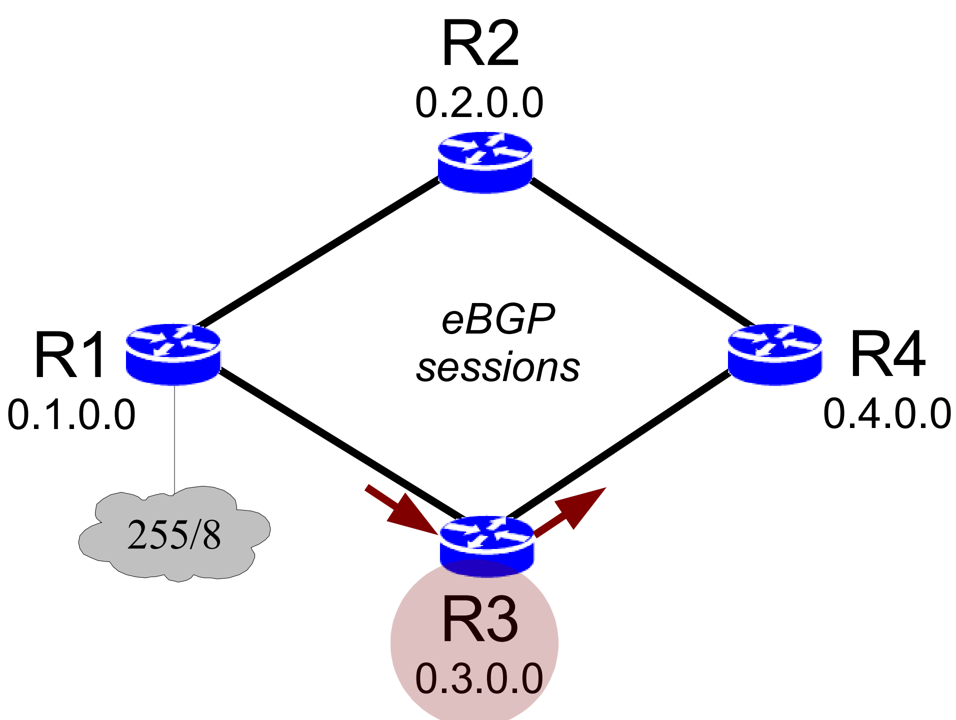
Simulation time: 1

Routing State:

R1:	**255/8, local**
R2:	**255/8, {1}** ADJ-RIB-in(R1): 255/8, {1}
R3:	
R4:	

C-BGP Convergence

- **Convergence Example**



Queue:

R2→R4: **255/8, {2 1}**
R3→R4: **255/8, {3 1}**

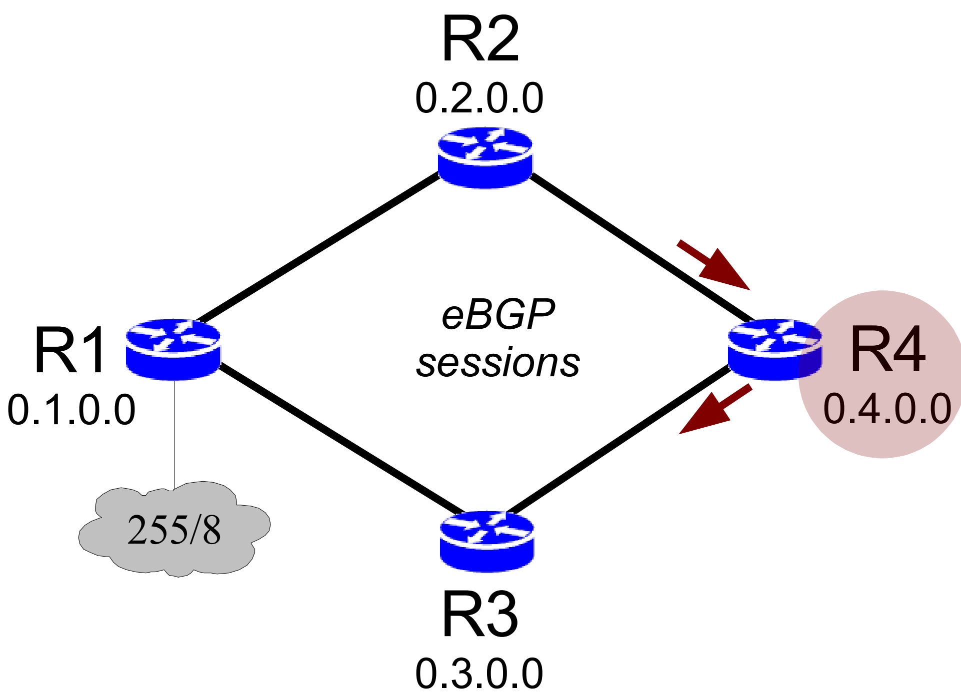
Simulation time: 2

Routing State:

R1:	**255/8, local**
R2:	**255/8, {1}** ADJ-RIB-in(R1): 255/8, {1}
R3:	**255/8, {1}** ADJ-RIB-in(R1): 255/8, {1}
R4:	

C-BGP Convergence

- **Convergence Example**



Queue:

R3→R4: **255/8, {3 1}**
R4→R3: 255/8, {4 2 1}

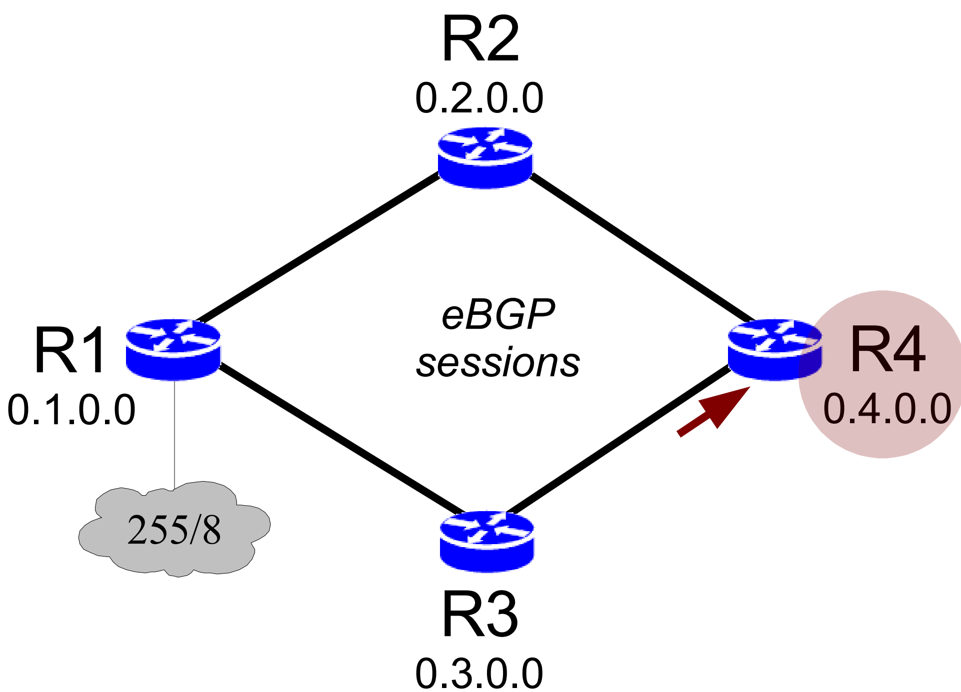
Simulation time: 3

Routing State:

R1:	**255/8, local**
R2:	**255/8, {1}** ADJ-RIB-in(R1): 255/8, {1}
R3:	**255/8, {1}** ADJ-RIB-in(R1): 255/8, {1}
R4:	**255/8, {2 1}** ADJ-RIB-in(R2): 255/8, {2 1}

C-BGP Convergence

- **Convergence Example**



Queue:

R4→R3: **255/8, {4 2 1}**

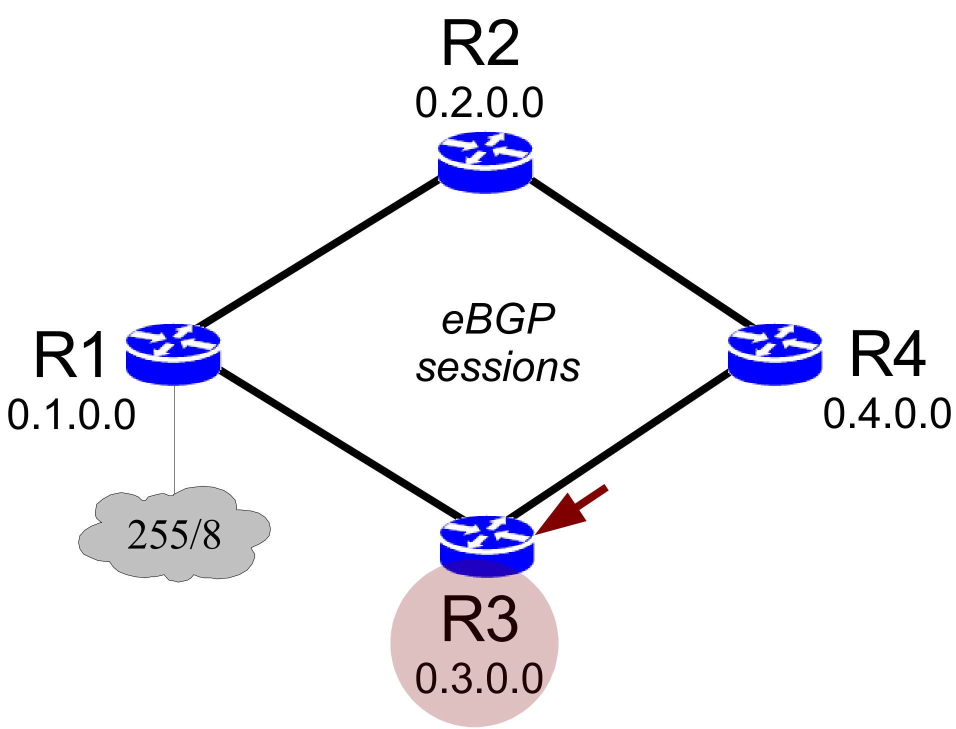
Simulation time: 4

Routing State:

R1:	**255/8, local**
R2:	**255/8, {1}** ADJ-RIB-in(R1): 255/8, {1}
R3:	**255/8, {1}** ADJ-RIB-in(R1): 255/8, {1}
R4:	**255/8, {2 1}** ADJ-RIB-in(R2): 255/8, {2 1} ADJ-RIB-in(R3): 255/8, {3 1}

C-BGP Convergence

- **Convergence Example**



Queue:

Queue is empty: simulation has converged !

Simulation time: 5

Routing State:

R1:	**255/8, local**
R2:	**255/8, {1}** ADJ-RIB-in(R1): 255/8, {1}
R3:	**255/8, {1}** ADJ-RIB-in(R1): 255/8, {1} ADJ-RIB-in(R4): 255/8, {4 2 1}
R4:	**255/8, {2 1}** ADJ-RIB-in(R2): 255/8, {2 1} ADJ-RIB-in(R3): 255/8, {3 1}

C-BGP Convergence

- **Observations based on previous example**
  - Simulation converges when queue is empty.
  - Each router receives its best route first. There is no path exploration during the convergence.

- **Can these results be generalized ?**
  - Does C-BGP always converge ? **NO**
    - If there is a unique solution and it is reachable, C-BGP will always find it
    - If there are multiple solutions, C-BGP might find one or fail to converge**.**
    - If there is no solution, C-BGP will not converge
  - Does C-BGP always compute the BGP outcome without path exploration ? **NO**
    - Can be caused by policies (route filters) or artificial propagation delays

C-BGP Convergence Issues

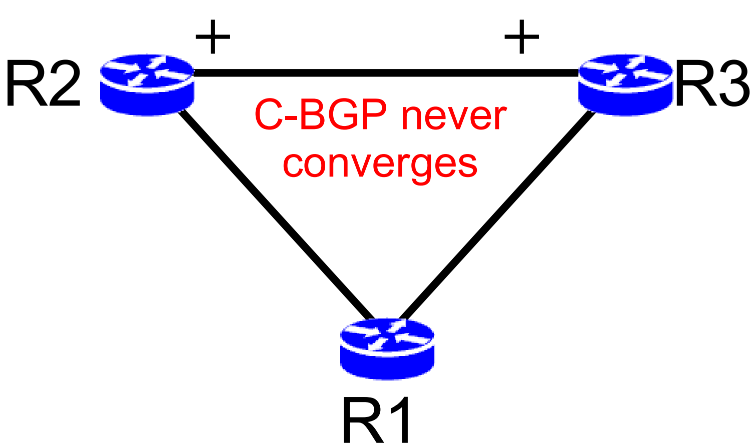
- **DISAGREE**
  - What if multiple solutions exist for a BGP system ?
  - **DISAGREE system** (described by T. Griffin in 1999)

Griffin and Wilfong, ACM SIGCOMM 1999

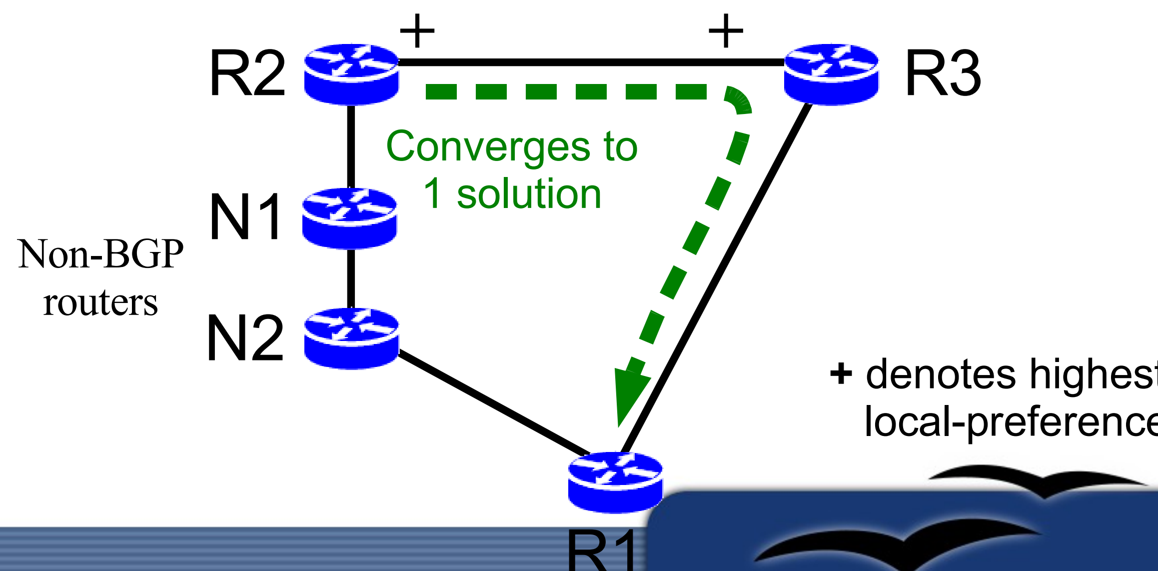
⇒ **C-BGP will not always converge in this case**

Reason: in DISAGREE, messages arrive consecutively at R2 and R3. They keep sending Update/Withdraw to each other and the queue is never empty.

DISAGREE *[GW99]*



DISAGREE2



C-BGP Convergence Issues

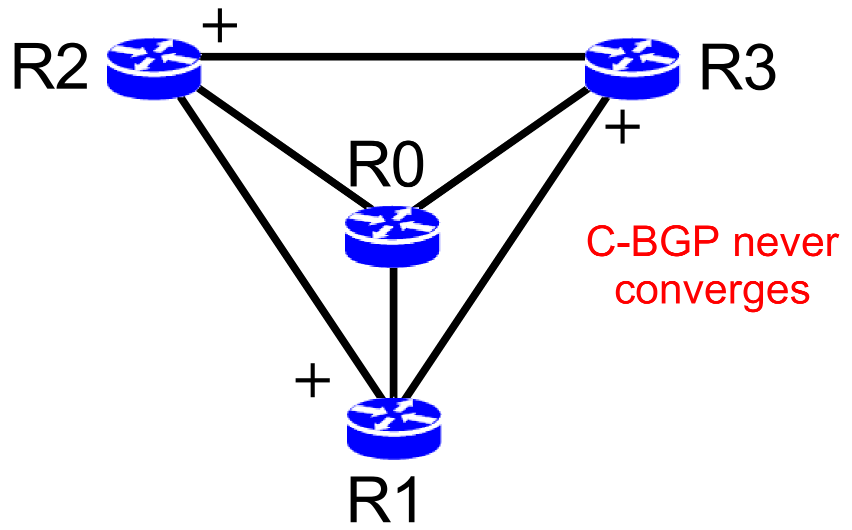
- **BAD-GADGET**
  - What if no solution exists ?
  - **BAD-GADGET system** (by T. Griffin in 1999)

Griffin and Wilfong, ACM SIGCOMM 1999

⇒ **C-BGP will never converge in this case**

<u>Reason:</u> no stable routing state exists, hence new messages are perpetually enqueued/dequeued. The event queue is never empty.

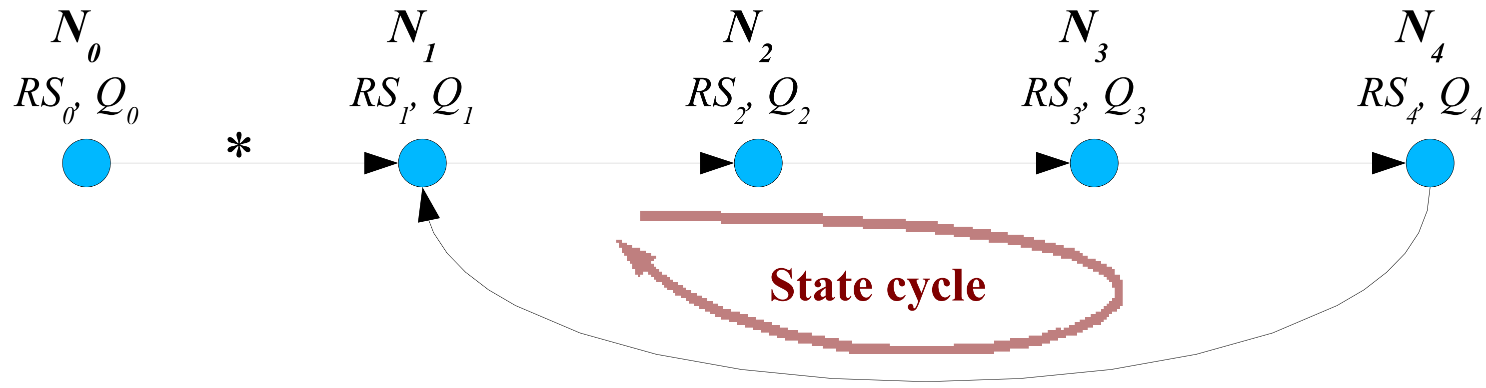
BAD-GADGET *[GW99]*



+ denotes highest local-preference

C-BGP Convergence Issues

- **Detecting non-convergence: state cycles**
  - Methodology:
    - Build path in the state graph, corresponding to current simulation run.
    - Each node N_i contains full simulation state (RS_i, Q_i).
    - Each transition $(j \rightarrow k)$ corresponds to the propagation of a BGP message from j to k.
  - Cycle detection:
    - If latest transition $(j \rightarrow k)$ leads to already traversed state $N_k = N_i$ ($i \leq j < k$).



C-BGP Convergence Issues

State	Queue	RS (1.0.0.1)	RS (2.0.0.1)	RS (3.0.0.1)
0	(1.0.0.1,2.0.0.1) UPDATE 255.0.0.0/8 (1.0.0.1,3.0.0.1) UPDATE 255.0.0.0/8	RIB:- RIB-in(3.0.0.1): RIB-in(2.0.0.1):	RIB: RIB-in(3.0.0.1): RIB-in(1.0.0.1):	RIB: RIB-in(1.0.0.1): RIB-in(2.0.0.1):
1	(1.0.0.1,3.0.0.1) UPDATE 255.0.0.0/8 (2.0.0.1,3.0.0.1) UPDATE 255.0.0.0/8	RIB:- RIB-in(3.0.0.1): RIB-in(2.0.0.1):	RIB:1 RIB-in(3.0.0.1): RIB-in(1.0.0.1):1	RIB: RIB-in(1.0.0.1): RIB-in(2.0.0.1):
2	(2.0.0.1,3.0.0.1) UPDATE 255.0.0.0/8 (3.0.0.1,2.0.0.1) UPDATE 255.0.0.0/8	RIB:- RIB-in(3.0.0.1): RIB-in(2.0.0.1):	RIB:1 RIB-in(3.0.0.1): RIB-in(1.0.0.1):1	RIB:1 RIB-in(1.0.0.1):1 RIB-in(2.0.0.1):
3	(3.0.0.1,2.0.0.1) UPDATE 255.0.0.0/8 (3.0.0.1,2.0.0.1) WITHDRAW 255.0.0.0/8	RIB:- RIB-in(3.0.0.1): RIB-in(2.0.0.1):	RIB:1 RIB-in(3.0.0.1): RIB-in(1.0.0.1):1	RIB:2 1 RIB-in(1.0.0.1):1 RIB-in(2.0.0.1):2 1
4	(3.0.0.1,2.0.0.1) WITHDRAW 255.0.0.0/8 (2.0.0.1,3.0.0.1) WITHDRAW 255.0.0.0/8	RIB:- RIB-in(3.0.0.1): RIB-in(2.0.0.1):	RIB:3 1 RIB-in(3.0.0.1):3 1 RIB-in(1.0.0.1):1	RIB:2 1 RIB-in(1.0.0.1):1 RIB-in(2.0.0.1):2 1
5	(2.0.0.1,3.0.0.1) WITHDRAW 255.0.0.0/8 (2.0.0.1,3.0.0.1) UPDATE 255.0.0.0/8	RIB:- RIB-in(3.0.0.1): RIB-in(2.0.0.1):	RIB:1 RIB-in(3.0.0.1): RIB-in(1.0.0.1):1	RIB:2 1 RIB-in(1.0.0.1):1 RIB-in(2.0.0.1):2 1
6	(2.0.0.1,3.0.0.1) UPDATE 255.0.0.0/8 (3.0.0.1,2.0.0.1) UPDATE 255.0.0.0/8	RIB:- RIB-in(3.0.0.1): RIB-in(2.0.0.1):	RIB:1 RIB-in(3.0.0.1): RIB-in(1.0.0.1):1	RIB:1 RIB-in(1.0.0.1):1 RIB-in(2.0.0.1):

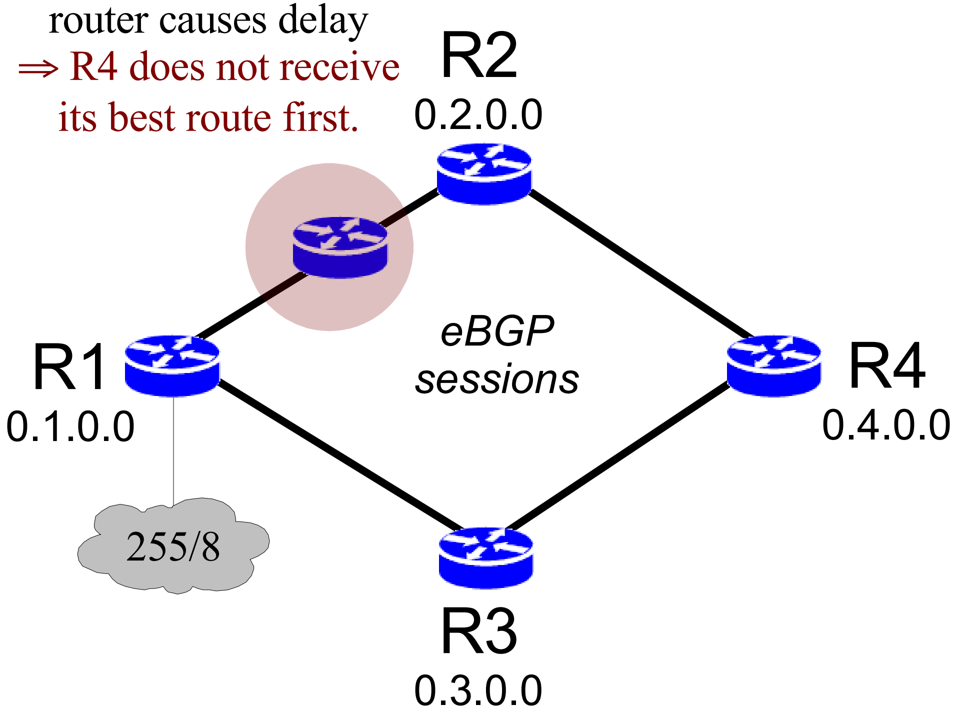
Cycle detected !

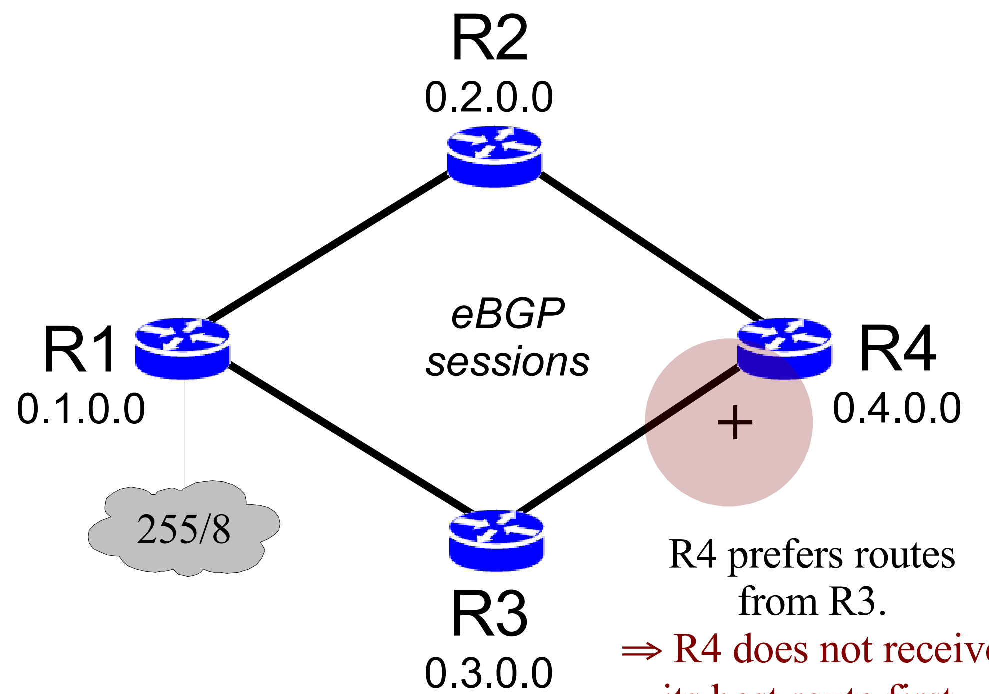
Example based on DISAGREE system

C-BGP Path Exploration

- **Caused by propagation delay ... or routing policies**

Intermediate (non-BGP) router causes delay
⇒ R4 does not receive its best route first.



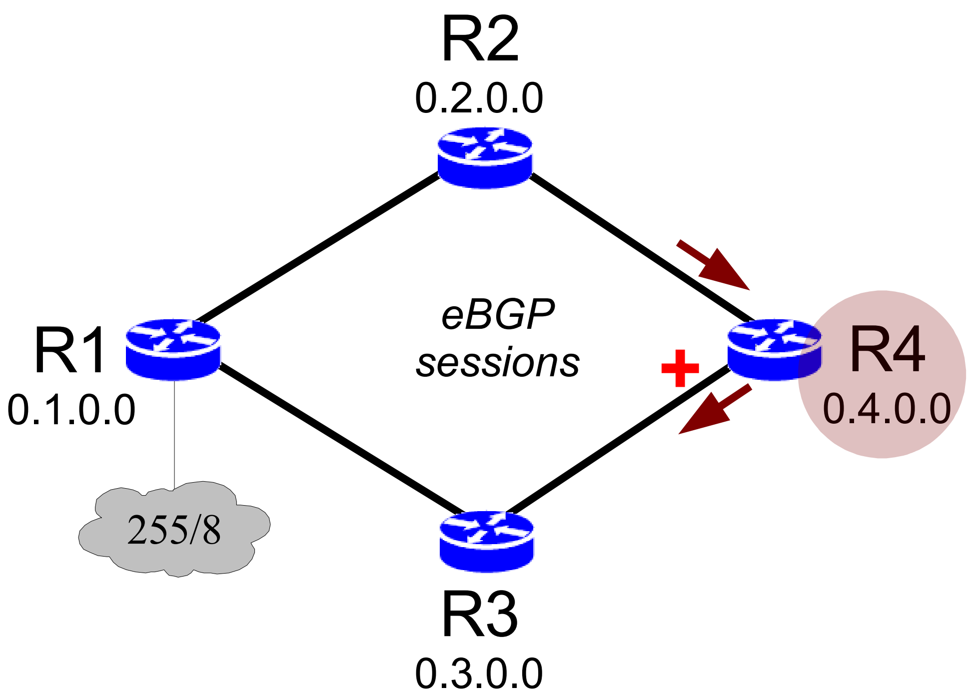


R4 prefers routes from R3.
⇒ R4 does not receive its best route first.

+ denotes highest local-preference

C-BGP Path Exploration

- **Convergence Example**



+ denotes highest local-preference

Queue:

R3→R4: **255/8, {3 1}**
R4→R3: 255/8, {4 2 1}

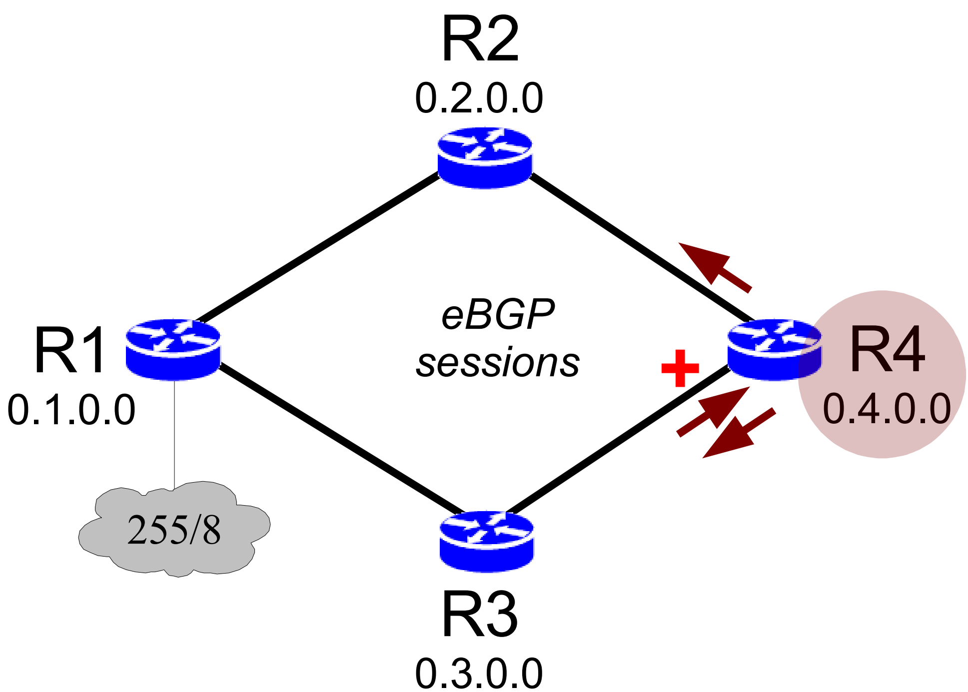
Simulation time: 3

Routing State:

R1:	**255/8, local**
R2:	**255/8, {1}** ADJ-RIB-in(R1): 255/8, {1}
R3:	**255/8, {1}** ADJ-RIB-in(R1): 255/8, {1}
R4:	**255/8, {2 1}** ADJ-RIB-in(R2): 255/8, {2 1}

C-BGP Path Exploration

- **Convergence Example**



Queue:

R4→R3: **255/8, {4 2 1}**
R4→R2: 255/8, {4 3 1}
R4→R3: 255/8, ---

Simulation time: 4

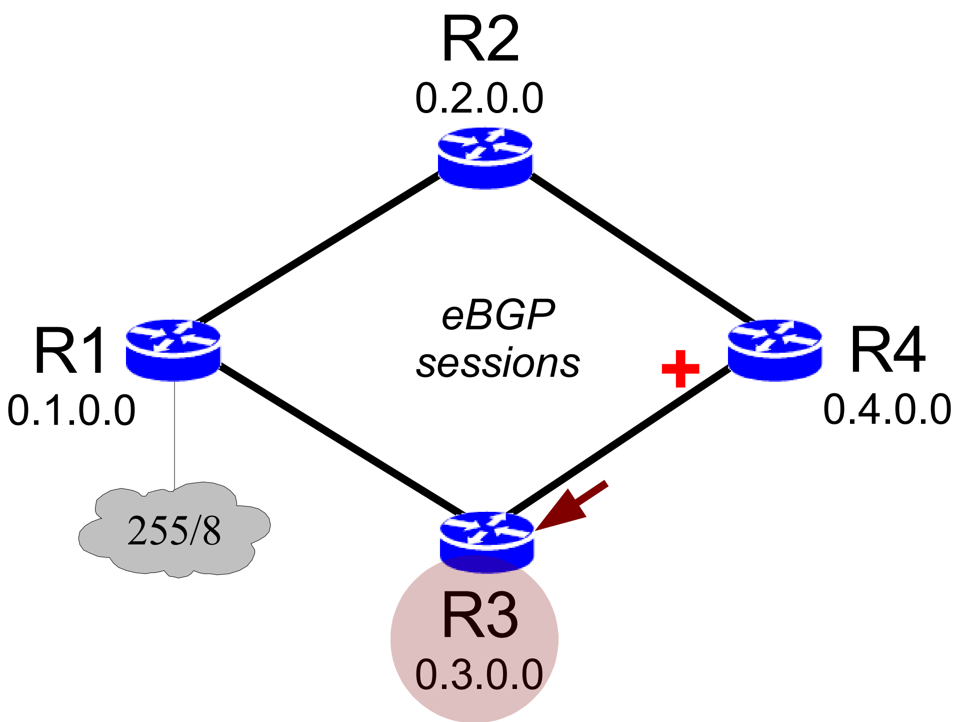
Routing State:

R1:	**255/8, local**
R2:	**255/8, {1}** ADJ-RIB-in(R1): 255/8, {1}
R3:	**255/8, {1}** ADJ-RIB-in(R1): 255/8, {1}
R4:	**255/8, {3 1}** ADJ-RIB-in(R2): 255/8, {2 1} ADJ-RIB-in(R3): 255/8, {3 1}

+ denotes highest local-preference

C-BGP Path Exploration

- **Convergence Example**



+ denotes highest local-preference

Queue:

R4→R2: **255/8, {4 3 1}**
R4→R3: **255/8, ---**

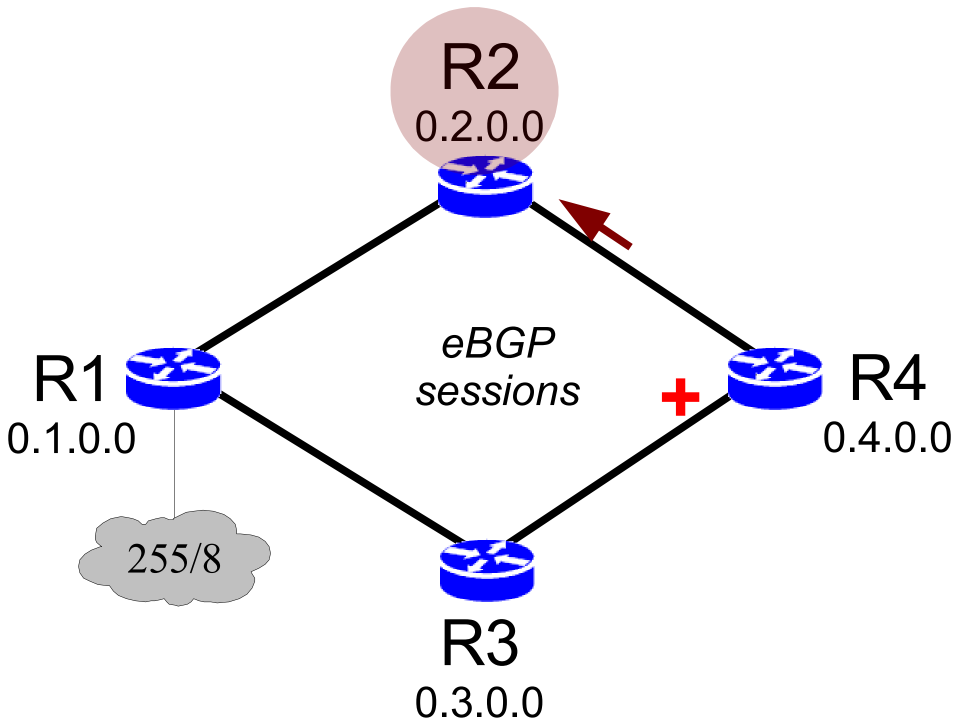
Simulation time: 5

Routing State:

R1:	**255/8, local**
R2:	**255/8, {1}** ADJ-RIB-in(R1): 255/8, {1}
R3:	**255/8, {1}** ADJ-RIB-in(R1): 255/8, {1} ADJ-RIB-in(R4): 255/8, {4 2 1}
R4:	**255/8, {3 1}** ADJ-RIB-in(R2): 255/8, {2 1} ADJ-RIB-in(R3): 255/8, {3 1}

C-BGP Path Exploration

- **Convergence Example**



+ denotes highest local-preference

Queue:

R4→R3: **255/8, ---**

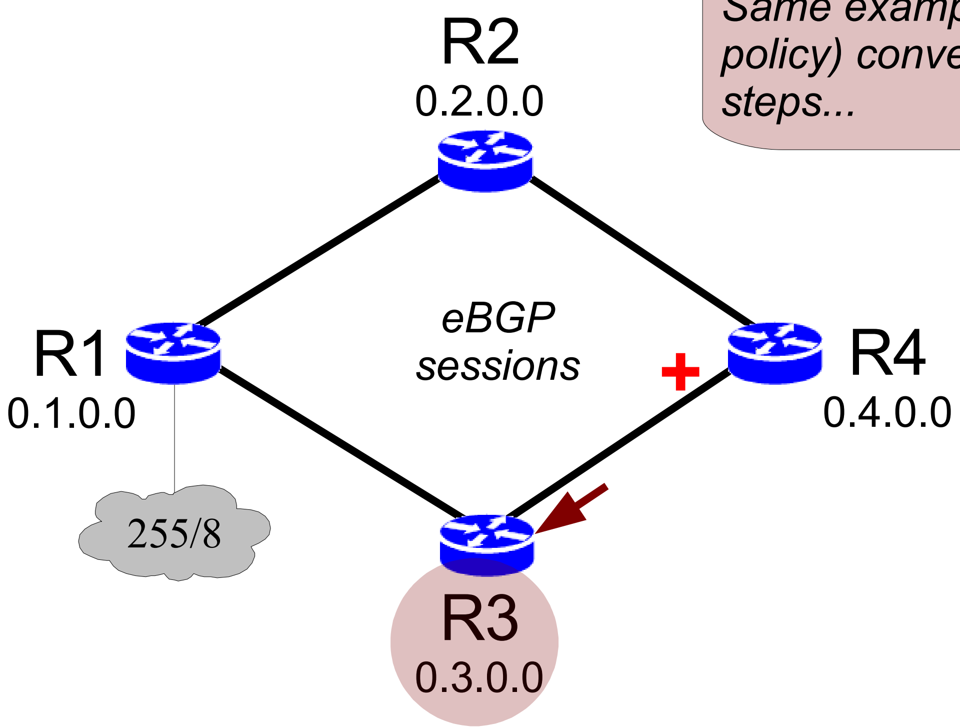
Simulation time: 6

Routing State:

R1:	**255/8, local**
R2:	**255/8, {1}** ADJ-RIB-in(R1): 255/8, {1} ADJ-RIB-in(R4): 255/8, {4 3 1}
R3:	**255/8, {1}** ADJ-RIB-in(R1): 255/8, {1} ADJ-RIB-in(R4): 255/8, {4 2 1}
R4:	**255/8, {3 1}** ADJ-RIB-in(R2): 255/8, {2 1} ADJ-RIB-in(R3): 255/8, {3 1}

C-BGP Path Exploration

- **Convergence Example**



Same example (without policy) converged in 5 steps...

Queue:

Queue is empty: simulation has converged !

Simulation time: 7

Routing State:

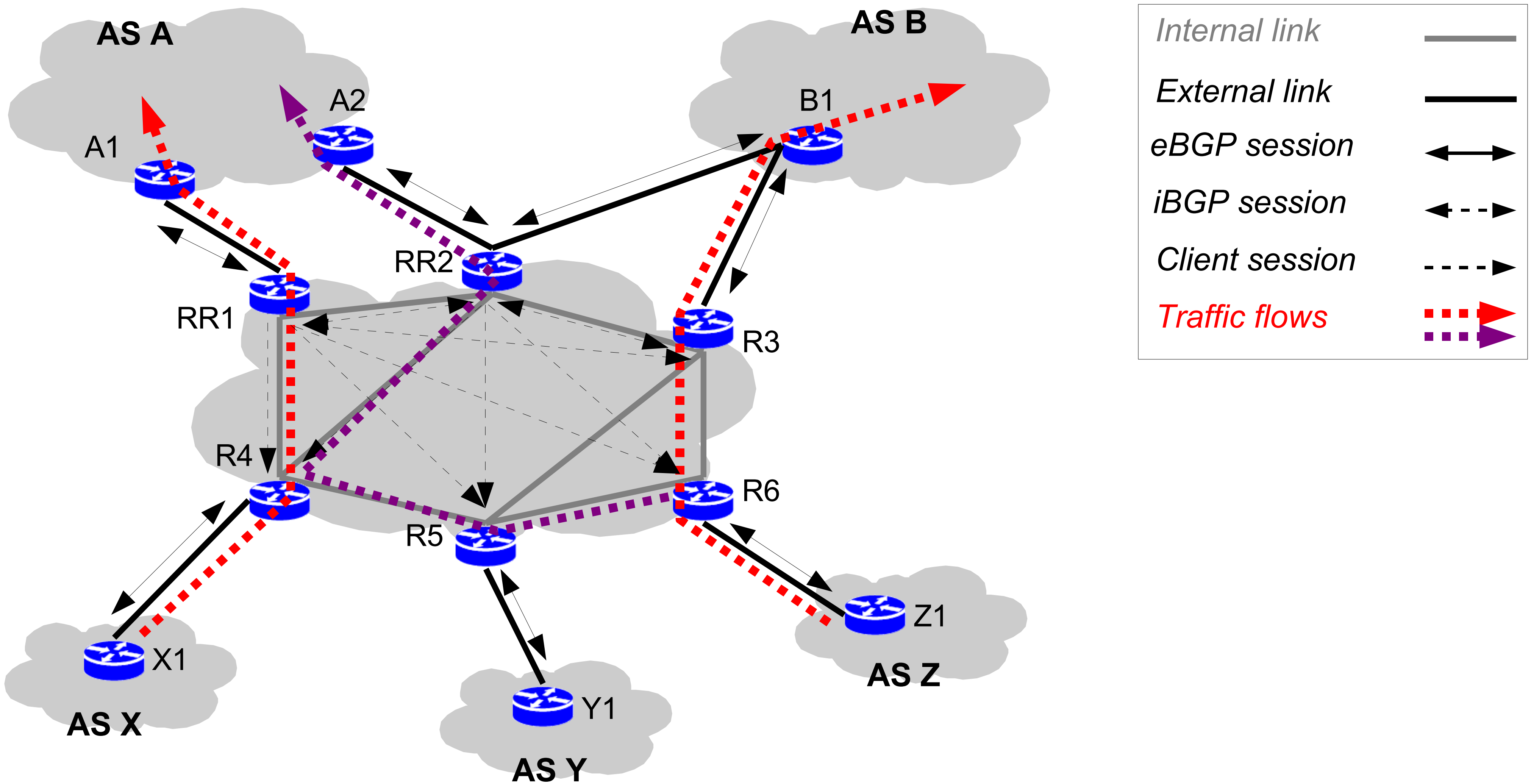
R1:	**255/8, local**
R2:	**255/8, {1}** ADJ-RIB-in(R1): 255/8, {1}
R3:	**255/8, {1}** ADJ-RIB-in(R1): 255/8, {1} ~~ADJ-RIB-in(R4): 255/8, {4 2 1}~~
R4:	**255/8, {2 1}** ADJ-RIB-in(R2): 255/8, {2 1} ADJ-RIB-in(R3): 255/8, {3 1}

+ denotes highest local-preference

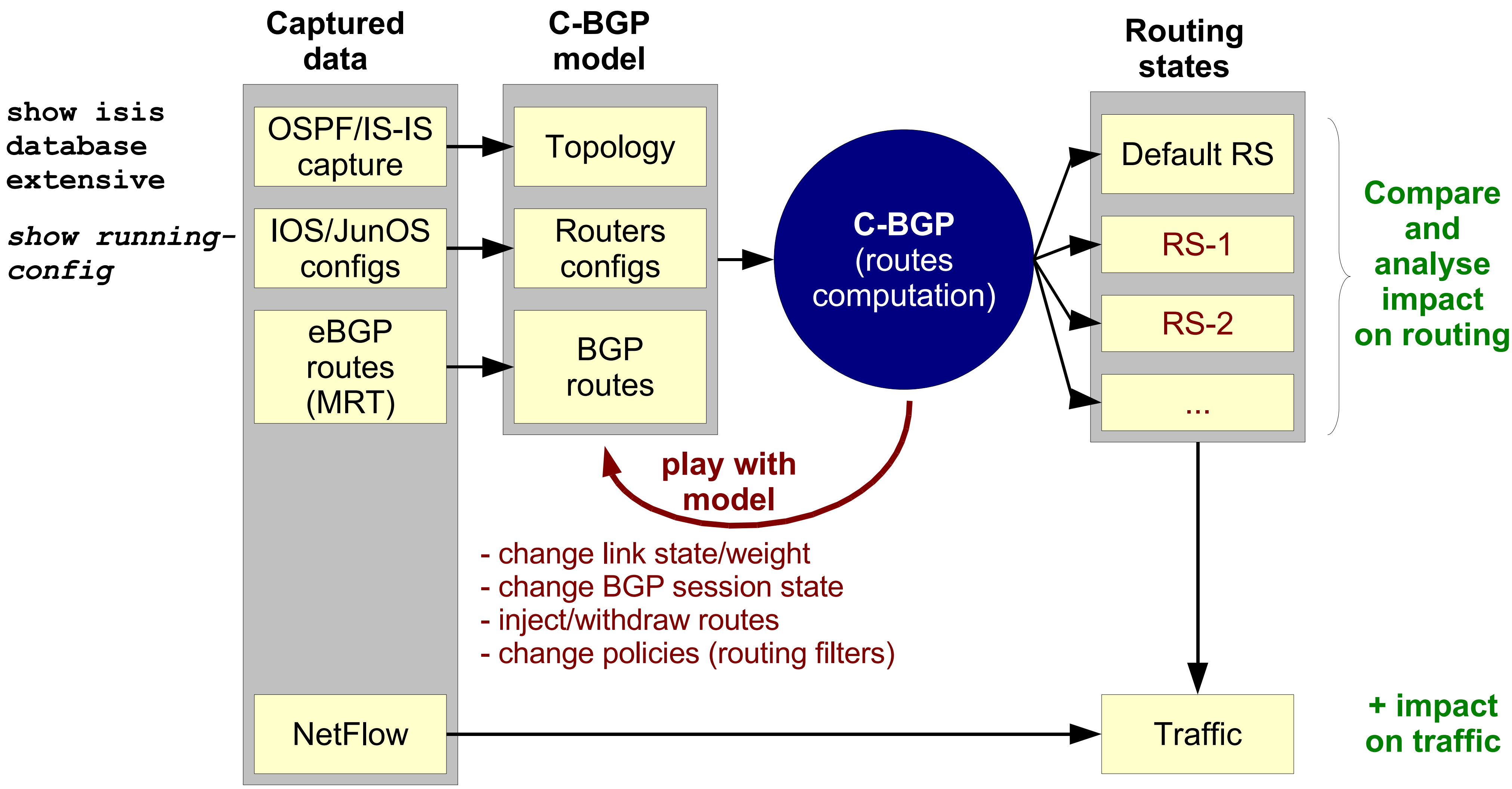
Agenda

- Introduction
- BGP Modeling
- C-BGP
- **Modeling an ISP**
- Modeling the Internet
- Using C-BGP
- Hands on...

Building the model



What-if Scenarios



Issues

- **Build an accurate model**
  - Configuration is distributed among many devices, in various formats
  - Inconsistent router configurations are frequent
  - Need parsers for various data formats
    - IGP database dumps (show isis database), router configurations (IOS, JunOS, XML, etc), BGP RIB dumps (MRT, show ip bgp), BGP message traces, NetFlow traces, ...
  - Obtaining complete BGP view is difficult (many vantage points)

Feamster et al, NSDI 2005

Feldmann and Rexford, IEEE Network, Sept/Oct 2001

- **Utilities to parse / convert network configuration**
  - `Netopeer` *(by liberouter project)*
    - http://www.liberouter.org/netopeer/about.php
  - `BGP-converter` *(by S. Tandel)*
    - http://www.info.ucl.ac.be/~standel/bgp-converter
  - `rcc` *(by N. Feamster)*
    - http://nms.lcs.mit.edu/bgp/rcc/

IS-IS Database

```
Router# show isis database extensive

IS-IS level 1 link-state database:

IS-IS level 2 link-state database:

rt1.net.00-00 Sequence: 0x1e9c3, Checksum: 0xceed, Lifetime: 1166 secs
   IS neighbor: rt2.net.00                    Metric:      400
   IS neighbor: rt3.net.00                    Metric:      400
   IP prefix: 192.168.0.0/30                  Metric:      400 Internal Up
   IP prefix: 192.168.0.4/30                  Metric:      400 Internal Up
   IP prefix: 10.0.0.1/32                     Metric:        0 Internal Up

rt2.net.00-00 Sequence: 0x2a840, Checksum: 0x7b1e, Lifetime: 736 secs
   IS neighbor: rt1.net.00                    Metric:      250
   IS neighbor: rt3.net.00                    Metric:      170
   IP prefix: 192.168.0.0/30                  Metric:      250 Internal Up
   IP prefix: 192.168.0.8/30                  Metric:      170 Internal Up
   IP prefix: 10.0.0.2/32                     Metric:        0 Internal Up

...
```

Source: fake example

IS-IS database (in XML)


```
<vn xmlns:junos="http://xml.juniper.net/junos/5.6R3/junos"
   xmlns:xnm="http://xml.juniper.net/xnm/1.1/xnm"><router name="atla-m5">
<isis-database-information xmlns="http://xml.juniper.net/junos/7.6R2/junos-
   routing" junos:style="extensive">
<isis-database>
  <level>1</level>
    <isis-database-entry>
      <lsp-id>ATLA-m5.00-00</lsp-id>
        <sequence-number>0x67f2</sequence-number>
        <checksum>0xfbeb</checksum>
        <remaining-lifetime>678</remaining-lifetime>
        <isis-prefix xmlns="http://xml.juniper.net/junos/7.6R2/junos-
   routing" junos:style="normal">
          <protocol-name>V6</protocol-name>
          <address-prefix>2001:3c8:e100:1004::/64</address-prefix>
          <metric>63</metric>
          <prefix-flag>External</prefix-flag>
          <prefix-status>down</prefix-status>
        </isis-prefix>
        <isis-prefix xmlns="http://xml.juniper.net/junos/7.6R2/junos-
   routing" junos:style="normal">
          <protocol-name>V6</protocol-name>
          <address-prefix>2001:400:2005:7::/64</address-prefix>
```


Source: Abilene Observatory (http://abilene.internet2.edu/observatory/)

Router configuration

```
...
bgp {
    log-updown;
    group ABILENE {
        type internal;
        local-address 198.32.8.194;
        family inet {
            any;
        }
        family inet-vpn {
            unicast;
        }
        export NEXT-HOP-SELF;
        peer-as 11537;
        neighbor 198.32.8.195 {
            description HSTNng;
        }
        neighbor 198.32.8.197 {
            description KSCYng;
        }
...
```

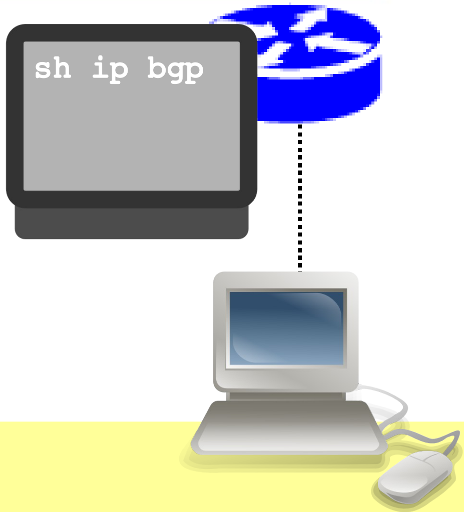
Source: Abilene Observatory

```
...
router bgp 100
 no synchronization
 bgp log-neighbor-changes
 neighbor 10.10.10.1 remote-as 100
 neighbor 10.10.10.1 next-hop-self
 neighbor 10.10.10.1 send-community
   both
 neighbor 10.10.10.4 remote-as 100
 neighbor 10.10.10.4 next-hop-self
 neighbor 10.10.10.4 send-community
   both
 neighbor 20.1.1.18 remote-as 200
 neighbor 20.1.1.18 dmzlink-bw
 neighbor 20.1.1.22 remote-as 200
 neighbor 20.1.1.22 dmzlink-bw
 maximum-paths 6
...
```

Source: CISCO example

BGP Routes

- **CISCO's "show ip bgp"**
  - Snapshot of RIB (subset of attributes)
  - Available on most router (need only telnet/ssh)
  - Not very handy



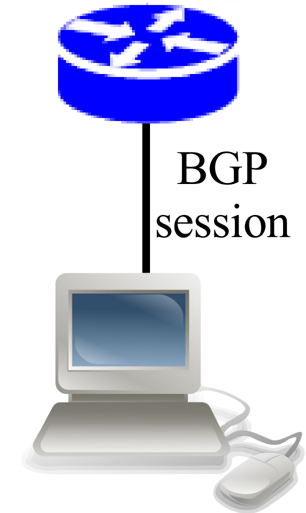

```
route-views.oregon-ix.net> show ip bgp
BGP table version is 321070916, local router ID is 198.32.162.100
Status codes: s suppressed, d damped, h history, * valid, > best, i - internal,
              S Stale
Origin codes: i - IGP, e - EGP, ? - incomplete

   Network          Next Hop            Metric LocPrf Weight Path
*  3.0.0.0          208.51.134.254         225             0 3549 701 703 80 i
*                   193.0.0.56                             0 3333 3356 701 703 80 i
*                   207.172.6.20             5             0 6079 3356 701 703 80 i
*                   194.85.4.55                            0 3277 3216 3549 701 703 80 i
*                   134.222.85.45                          0 286 3549 701 703 80 i
*                   203.62.252.186                         0 1221 4637 703 80 i
...
```

Source: http://www.routeviews.org

BGP Routes

- **MRT (Multithreaded Routing Toolkit)**
  - RIB dumps (snapshot) or BGP messages trace
  - All attributes are recorded
  - Well documented, easy to parse




```
TIME: 2007-6-26 14:27:50
TYPE: MSG_TABLE_DUMP/AFI_IP
VIEW: 0  SEQUENCE: 2
PREFIX: 3.0.0.0/8
STATUS: 1  ORIGINATED: Tue Jun 26 01:41:56 2007
FROM: 62.18.14.252 AS12682
AS_PATH: 12682 1299 701 703 80
NEXT_HOP: 62.18.14.252

TIME: 2007-6-26 14:27:50
TYPE: MSG_TABLE_DUMP/AFI_IP
VIEW: 0  SEQUENCE: 3
PREFIX: 3.0.0.0/8
STATUS: 1  ORIGINATED: Mon Jun 25 14:30:59 2007
FROM: 208.51.134.253 AS3549
...
```

Source: http://www.routeviews.org

route_btoa
  http://mrt.sourceforge.net

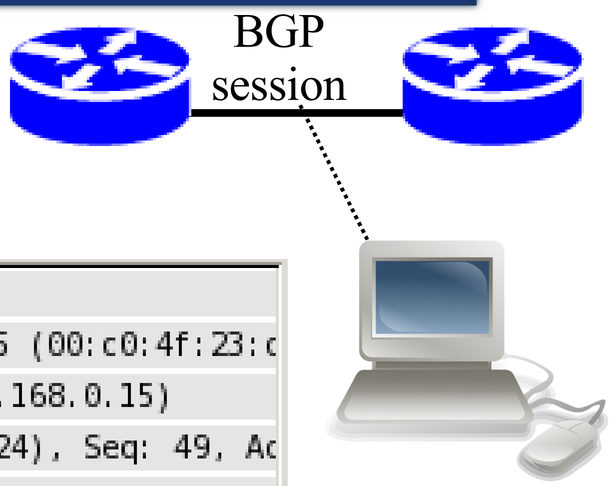
libbgpdump
  http://www.ris.ripe.net/source/libbgpdump-1.4.99.7.tar.gz

zebra-dump-parser
  http://www.linux.it/~md/software/zebra-dump-parser.tgz

BGPAnalysis
  http://gforge.info.ucl.ac.be/projects/bgpprobing

BGP Routes

- **libPCAP (tcpdump, ethereal, wireshark)**
  - Records details such as underlying TCP connection segments

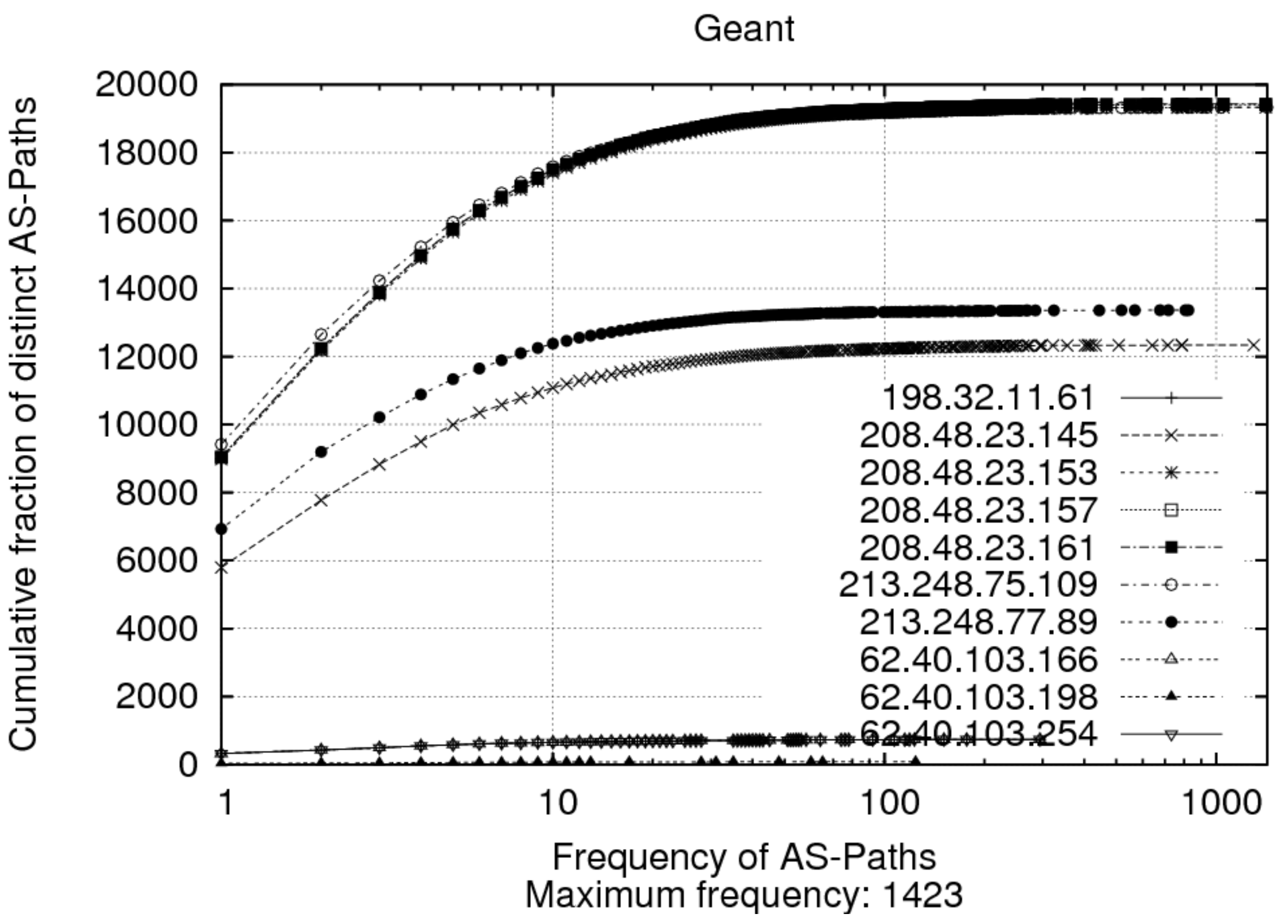



```
▷ Frame 17 (118 bytes on wire, 118 bytes captured)
▷ Ethernet II, Src: Cisco_35:0e:1c (00:00:0c:35:0e:1c), Dst: DellComp_23:c5:95 (00:c0:4f:23:c
▷ Internet Protocol, Src: 192.168.0.33 (192.168.0.33), Dst: 192.168.0.15 (192.168.0.15)
▷ Transmission Control Protocol, Src Port: bgp (179), Dst Port: elatelink (2124), Seq: 49, Ac
▽ Border Gateway Protocol
  ▽ UPDATE Message
      Marker: 16 bytes
      Length: 64 bytes
      Type: UPDATE Message (2)
      Unfeasible routes length: 0 bytes
      Total path attribute length: 39 bytes
    ▽ Path attributes
      ▷ ORIGIN: EGP (4 bytes)
      ▷ AS_PATH: empty (3 bytes)
      ▷ NEXT_HOP: 192.168.0.33 (7 bytes)
      ▷ MULTI_EXIT_DISC: 0 (7 bytes)
      ▷ LOCAL_PREF: 100 (7 bytes)
```

Source: http://www.wireshark.org

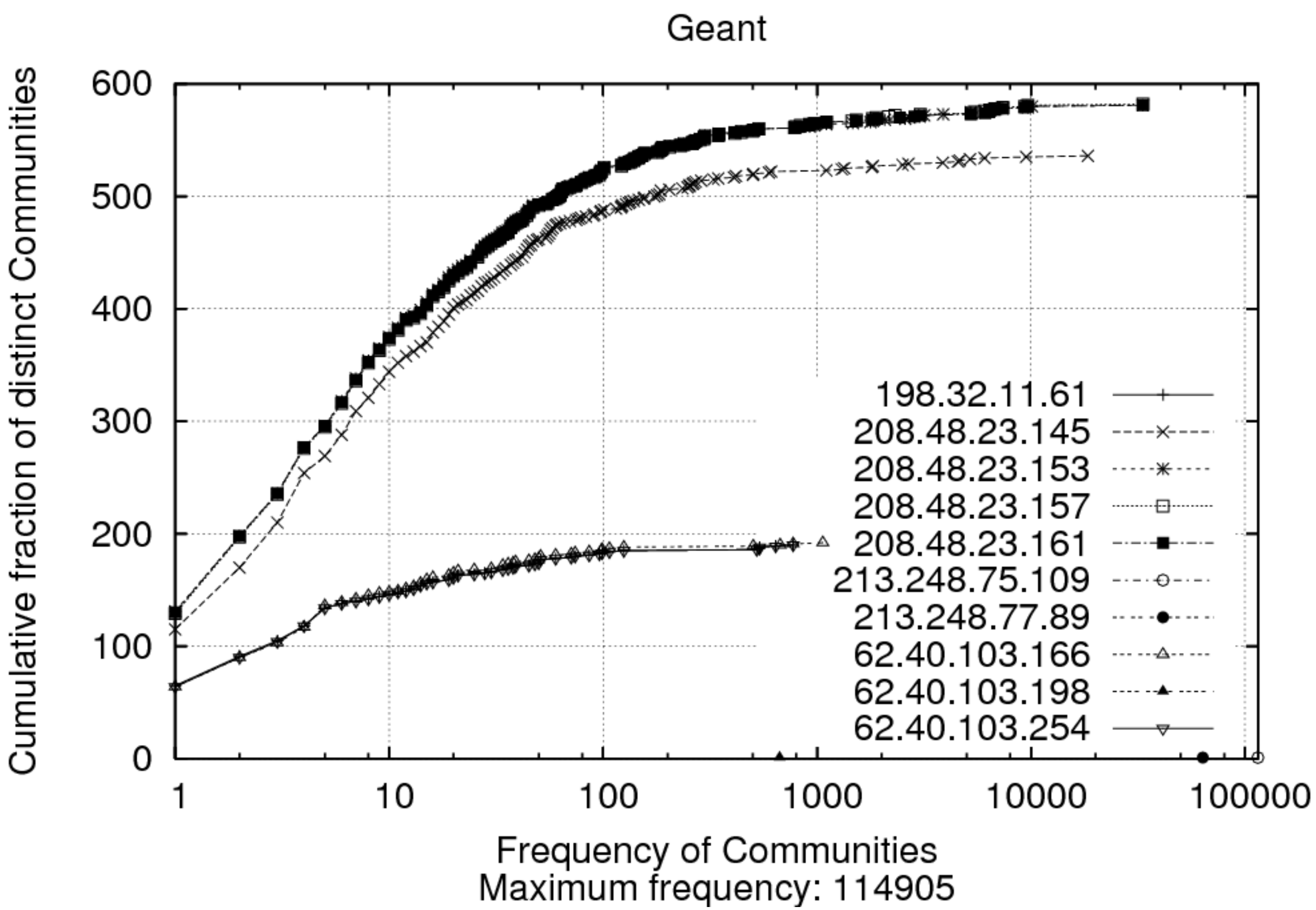
Memory

- **Routing Tables Redundancy**
  - Example: AS-Paths (from GEANT BGP table dump).



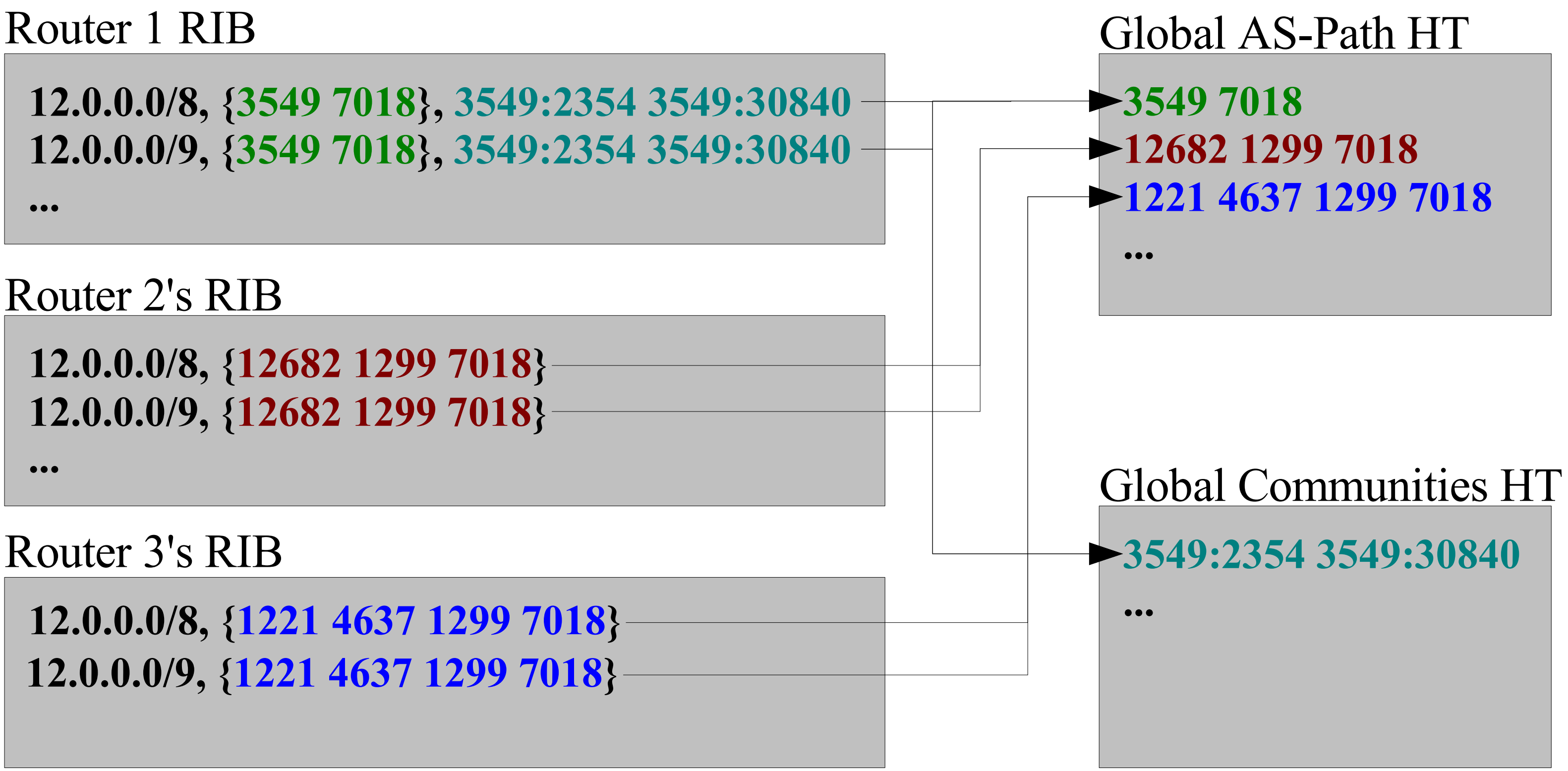
Memory

- **Routing Tables Redundancy**
  - Example: Communities (from GEANT BGP table dump).



Memory

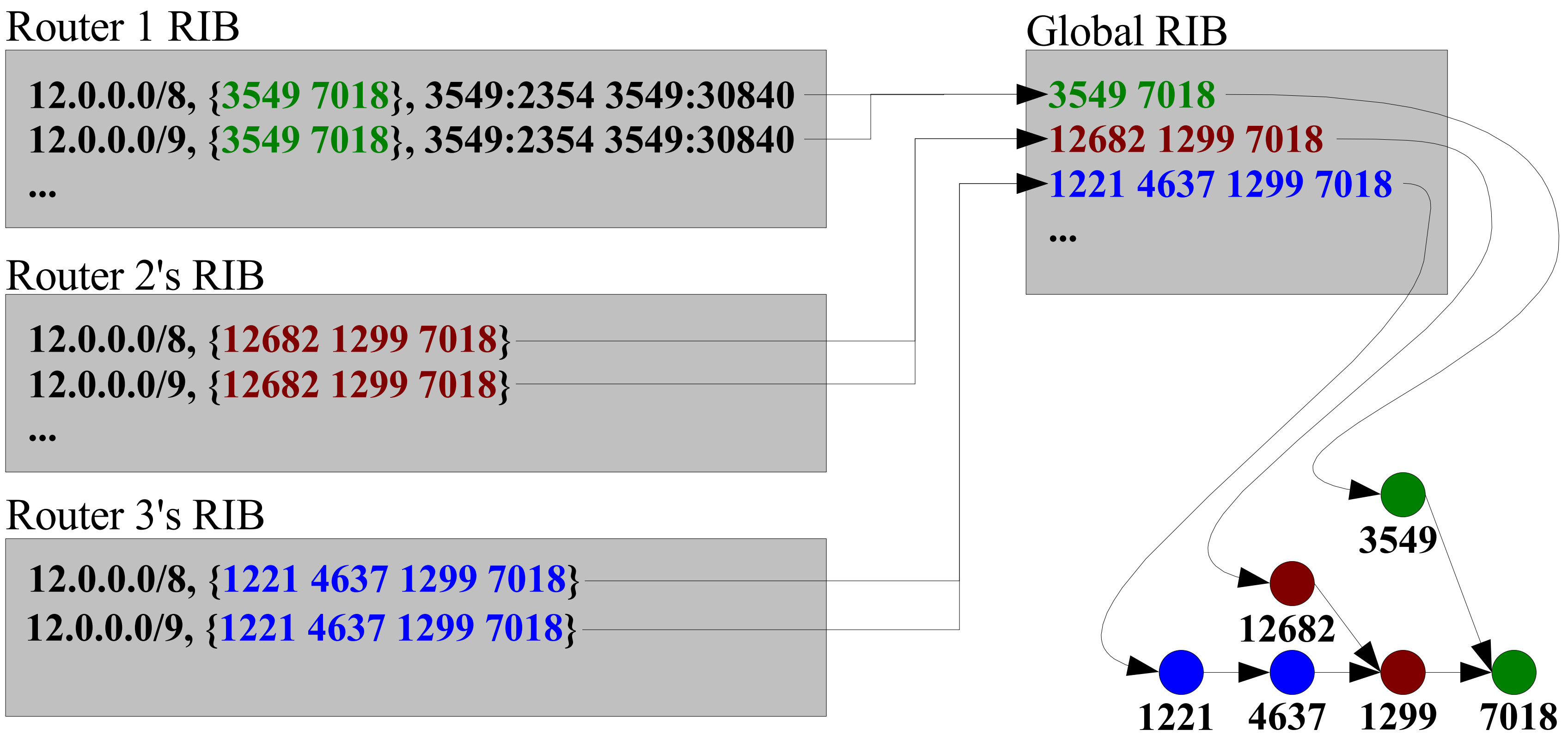
- **Impact on Routing Tables Structure**



Memory

- **Hao and Koppol's proposal: next step ?**

Hao and Koppol,
ACM SIGCOMM CCR,
July 2003



Issues

- **Scalability**
  - Routes towards 1000s of destinations learned at various vantage points
  - Many prefixes learned from
    - same neighbors
    - with same BGP “quality”

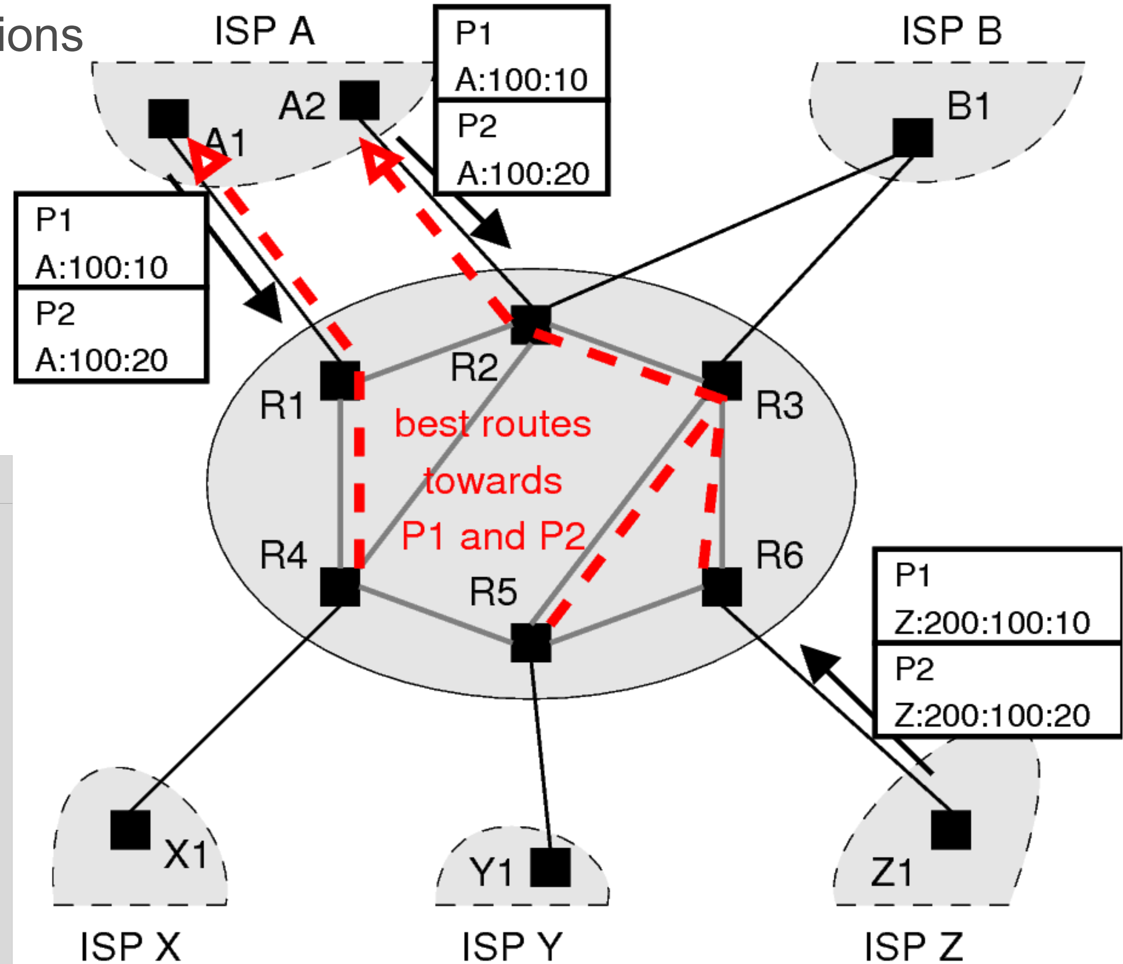
  ⇒ **Prefix Clustering**

Cluster1

Egress-list: R1, R2, R6

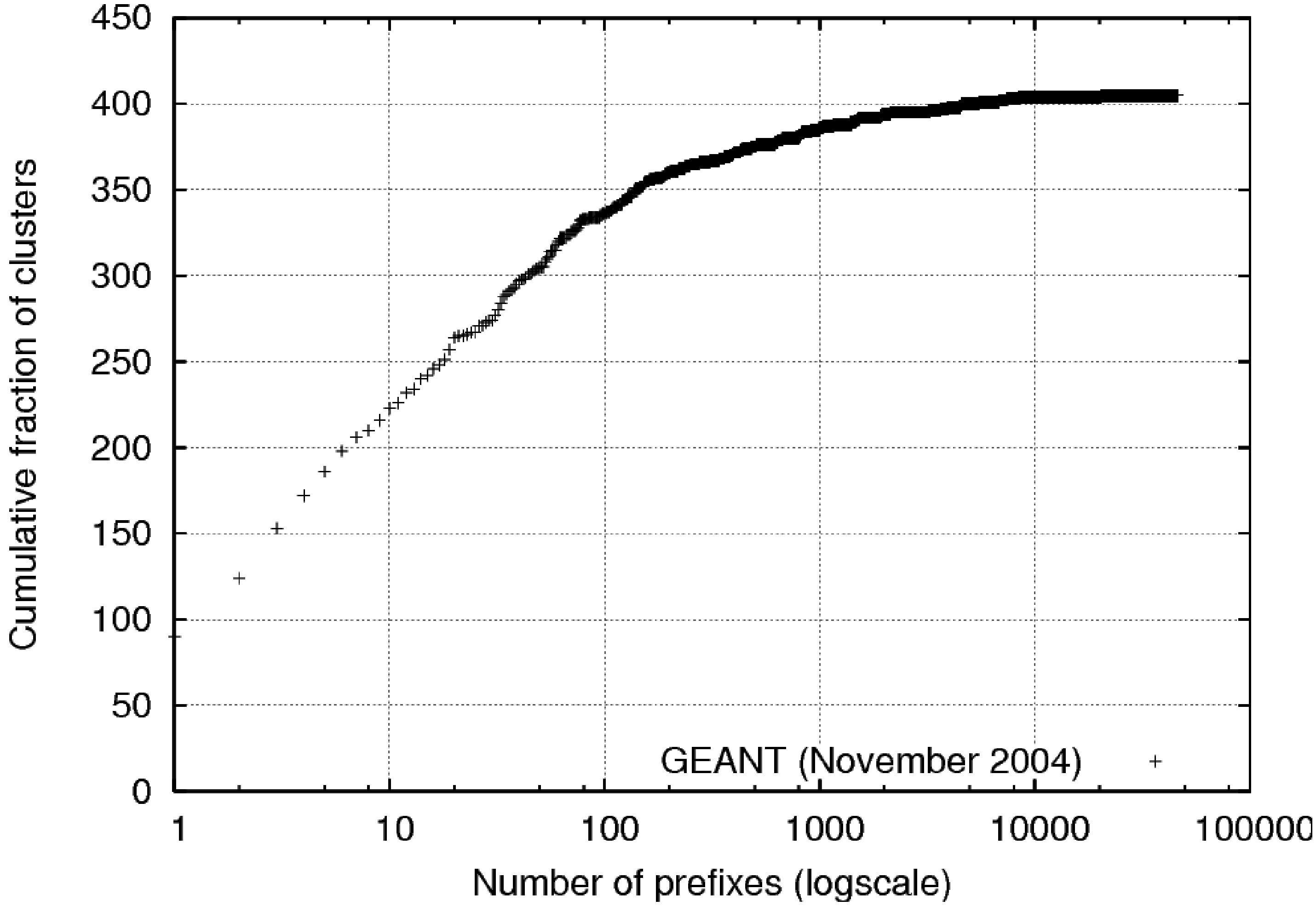
P1
(R1 -> A:100:10),
(R2 -> A:100:10),
(R6 -> Z:200:100:10)

P2
(R1 -> A:100:20),
(R2 -> A:100:20),
(R6 -> Z:200:100:20)



Prefix Clustering

- **105,071 prefixes ⇒ 406 clusters**



Example Research Studies

- **Modeling the Routing of an ISP with C-BGP**
  - Q: What is the impact of link failures and prospective peerings on the traffic matrix ?
  - C-BGP was used to compute routing state after changes in the physical and logical topologies.

Quoitin and Uhlig, IEEE Network, November 2005

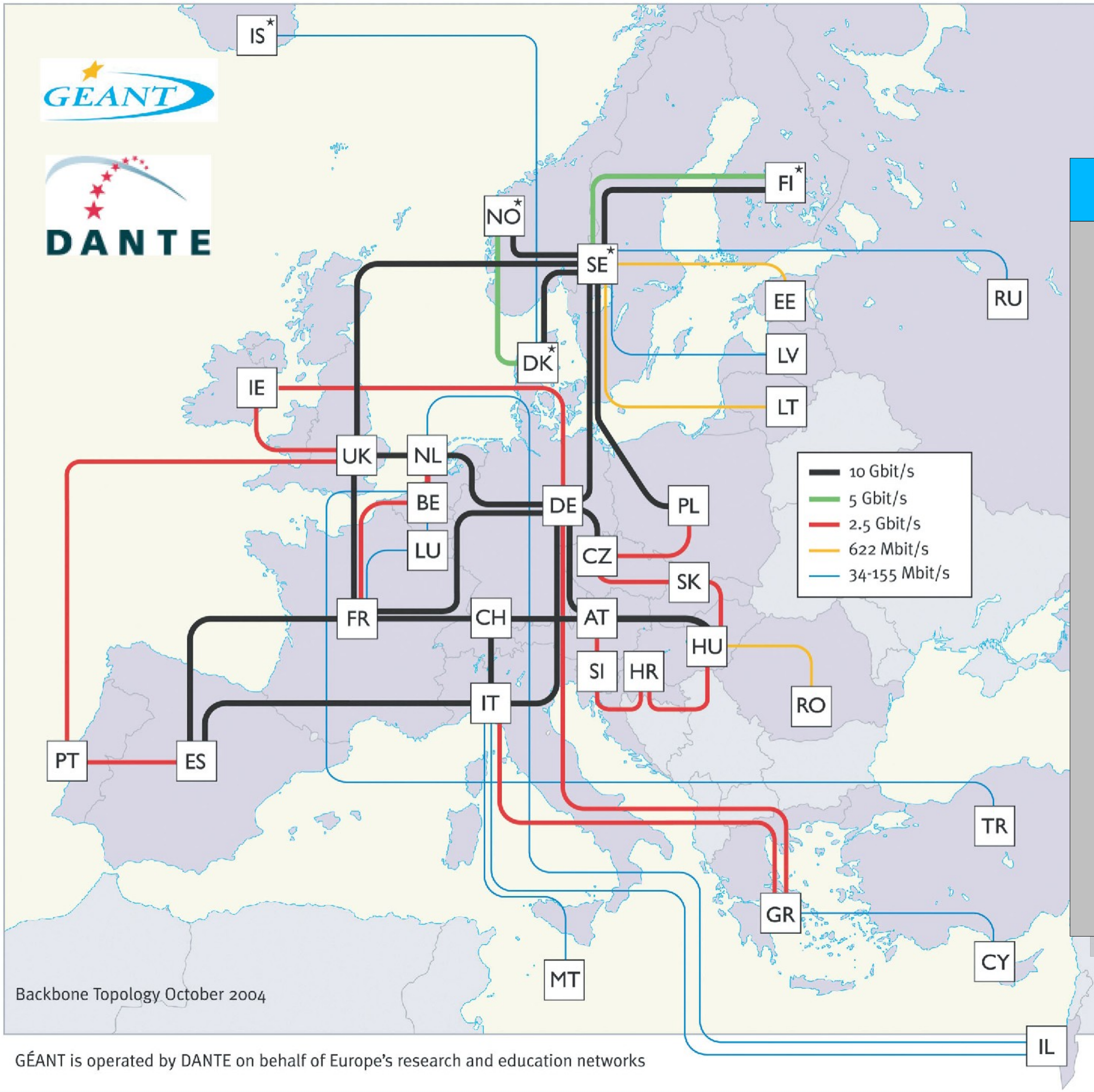
- **Providing public intradomain traffic matrices to the research community**
  - Q: How to obtain an accurate intradomain traffic matrix ?
  - C-BGP was used to compute forwarding paths for traffic transiting through GEANT.

Uhlig et al, ACM SIGCOMM CCR, January 2006

- **The Interaction of IGP Weight Optimization with BGP**
  - Q: Can we ignore BGP routing when performing IGP-optimization based traffic engineering (Fortz & Thorup) ?
  - C-BGP was used to compute routing state with optimized IGP weights.

Cerav-Erbas et al, ICISP, 2006

Case Study: GEANT



Some figures

"Pan-european research and education network"

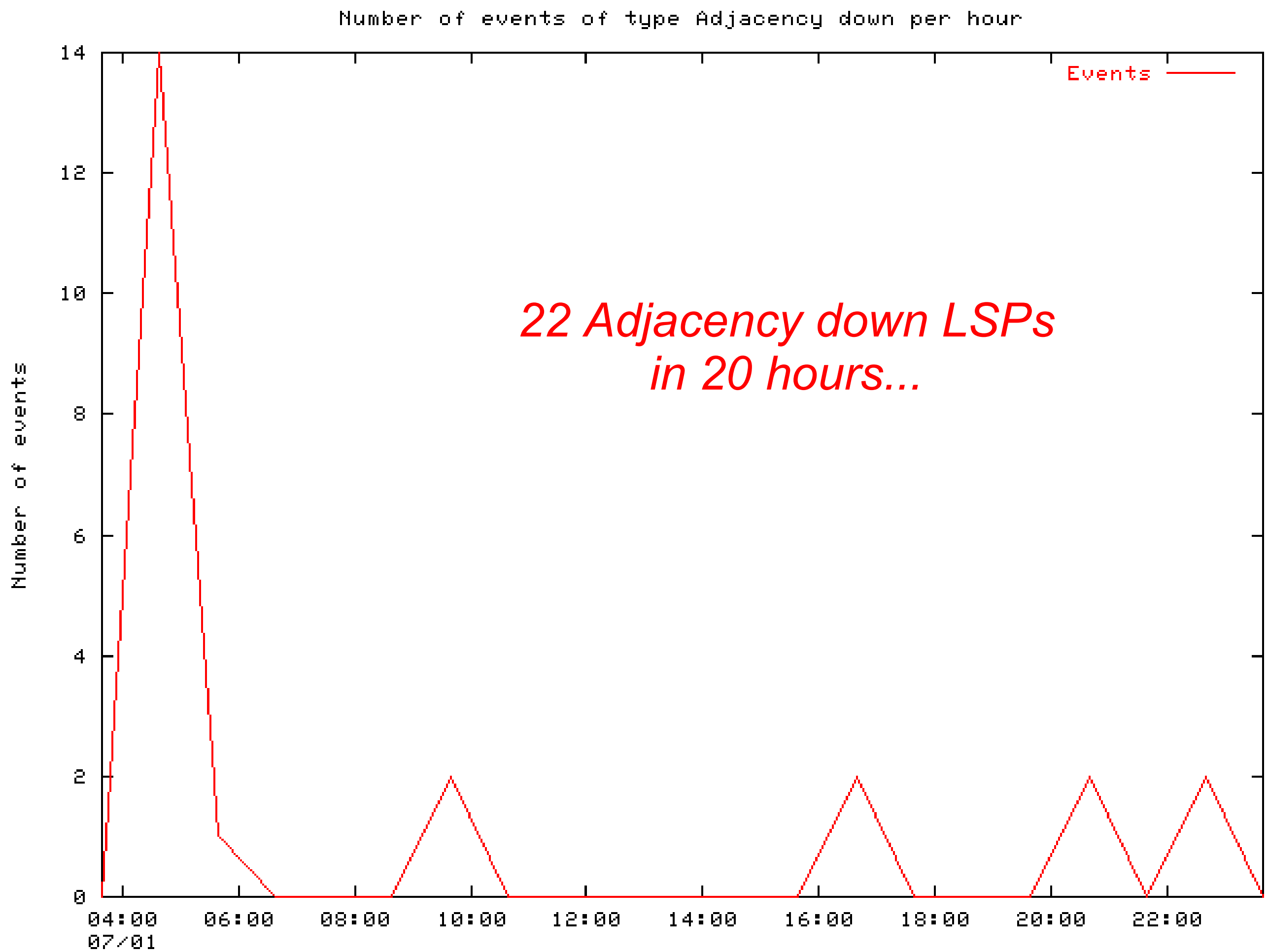
23 nodes
38 routers
53 peers

BGP routing:
  640,897 routes
  (105,071 prefixes)

Traffic:
  Netflow
  1/1000 sampling

Link Failures

- **What if a link or router fails ?**
  - IGP Adjacency changes
    - Failures
    - Maintenance
    - TE / tweaking
  - Routing affected ?
  - Traffic affected ?



Link Failures

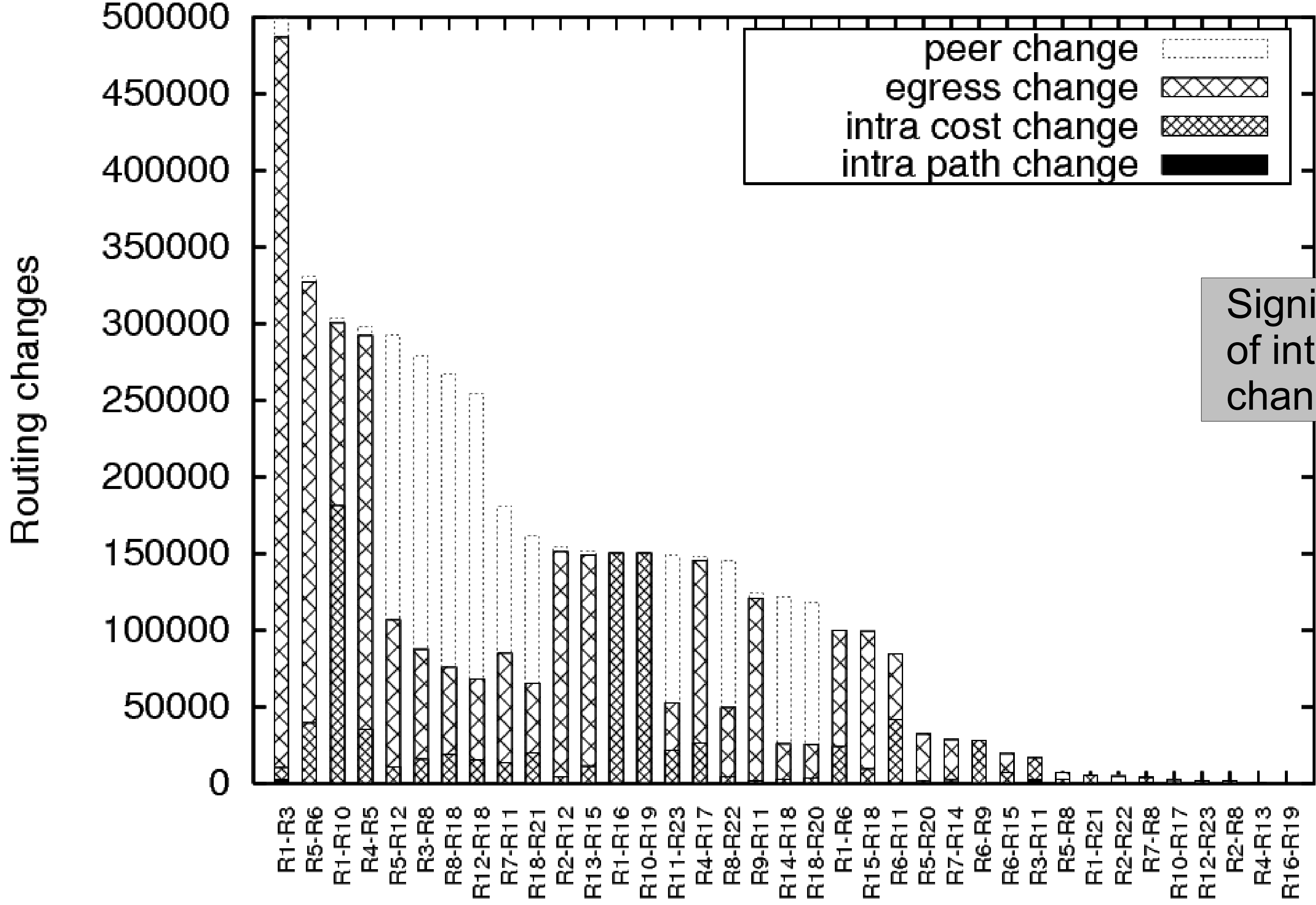
- **Methodology:**
  - Take snapshot of Routing State (default situation)
  - Remove a single link or router, let C-BGP converge
  - Compare Routing State with default
  - Classification of routing changes

Routing change	Class
Prefix not reachable	PREFIX DOWN
Prefix reachable	PREFIX UP
Neighbor AS has changed	PEER CHANGE
Egress router has changed	EGRESS CHANGE
IGP cost has changed	INTRA COST CHANGE
Intradomain path has changed	INTRA PATH CHANGE

BGP changes (first four rows)

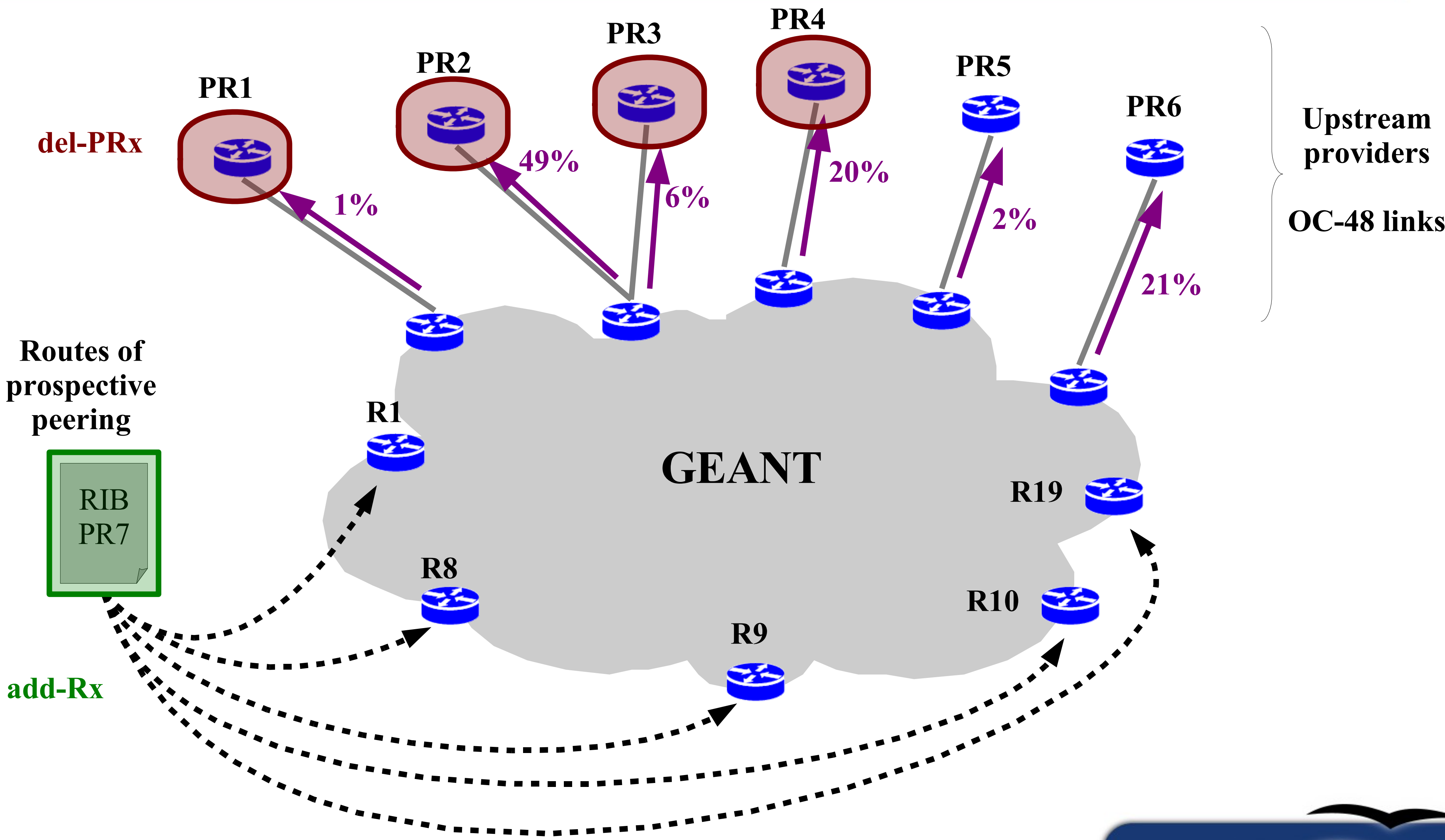
IGP changes (last two rows)

Link Failures

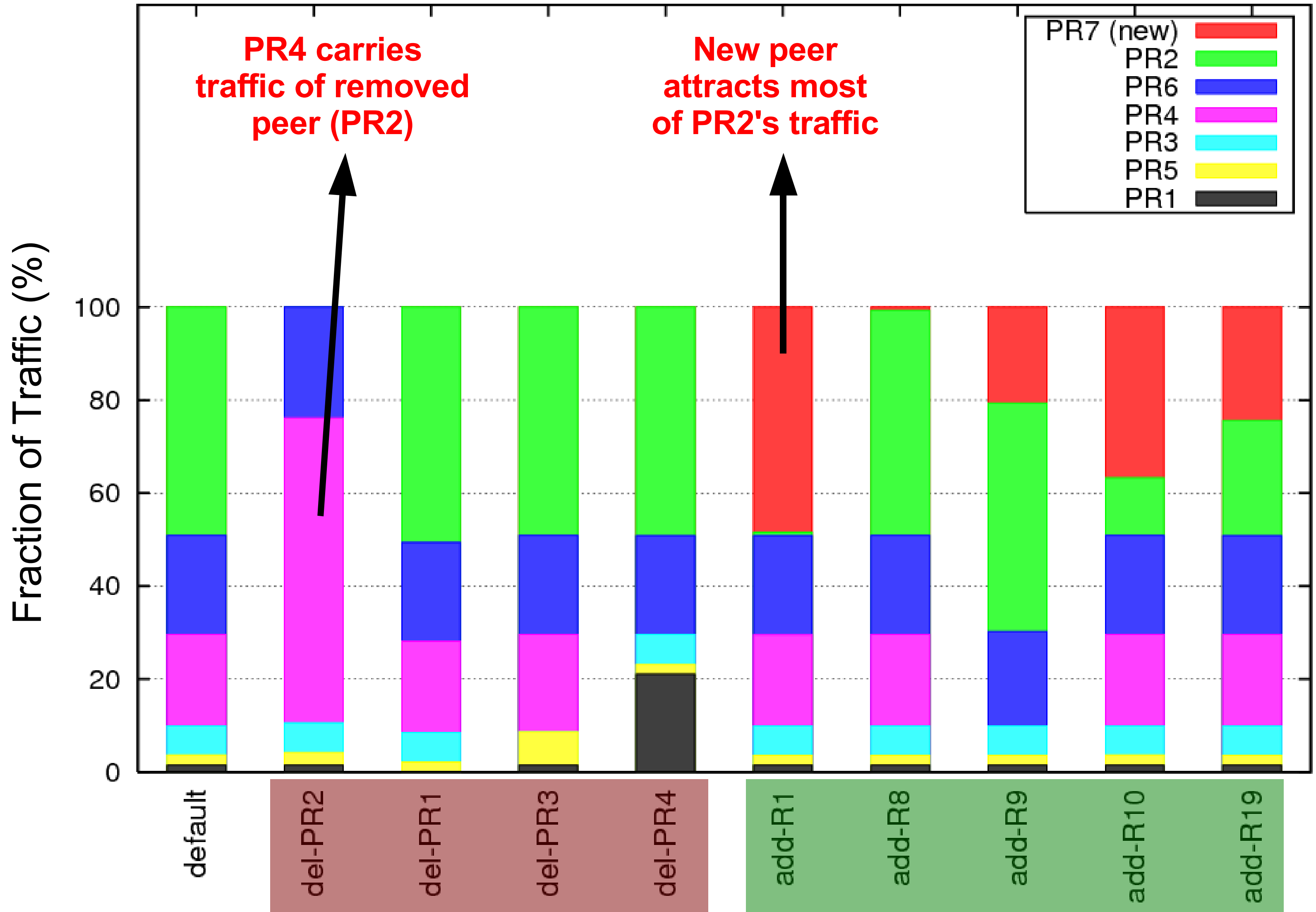


Significant number of interdomain changes...

Peering Changes



Peering Changes

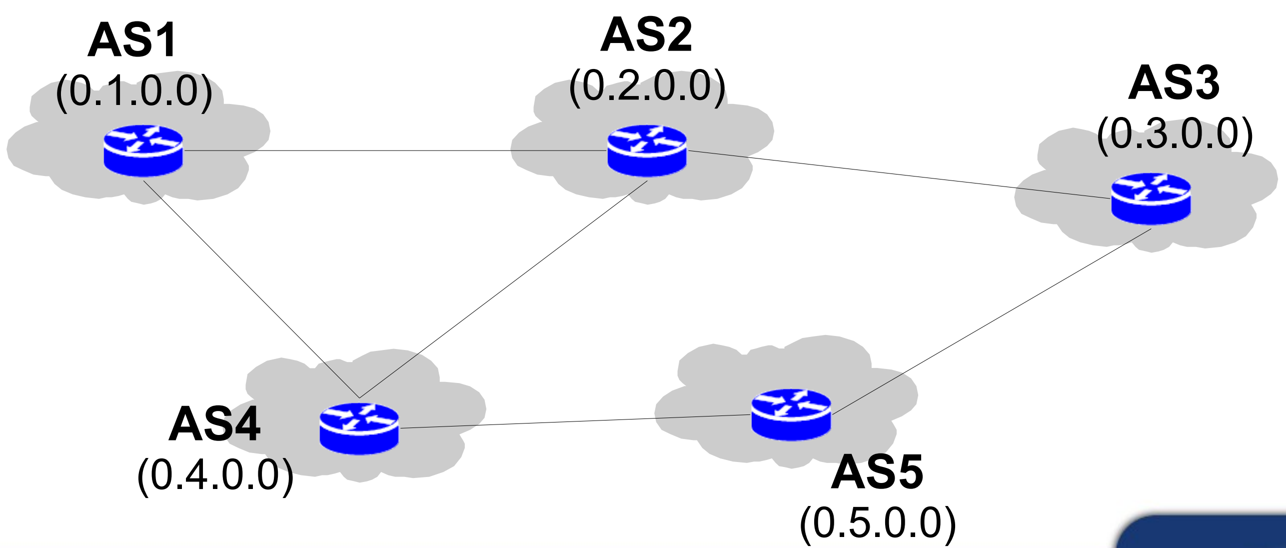


Agenda

- Introduction
- BGP Modeling
- C-BGP
- Modeling an ISP
- → **Modeling the Internet**
- Conclusion

AS-level Topology

- **Model**
  - Nodes = AS, Edges = business relationships
  - Each AS is modeled with a single router
    - Identifier derived from ASN
    - Example: AS7018 → 27.106.0.0
  - Business relationships enforced by routing filters
    - Valley-free property enforced with Communities (selective-export)
    - Business-preference enforced with Local-Preference (Cust > Peer > Prov)



Example Research Studies (1)

- **A performance evaluation of BGP-based traffic engineering**
  - Q: How accurate and deterministic is ASPP for traffic engineering ?
  - C-BGP is used to compute paths with different amounts of ASPP.

Quoitin et al, International Journal of Network Management, May/June 2005

- **Leveraging Network Performances with IPv6 Multihoming and Multiple Provider-Dependant Aggregatable Prefixes**
  - Q: How does the path diversity obtained by using IPv6 PA addresses compare to that obtained with BGP ?
  - C-BGP is used to compute paths between multi-homed stubs with one prefix per provider.

de Launois et al, Computer Networks, June 2006

- **Building an AS-topology model that captures route diversity**
  - Q: How to obtain an AS-level topology with a better path diversity ?
  - C-BGP used to compute paths in a model inferred from real routes observed at more than 1,300 vantage points.

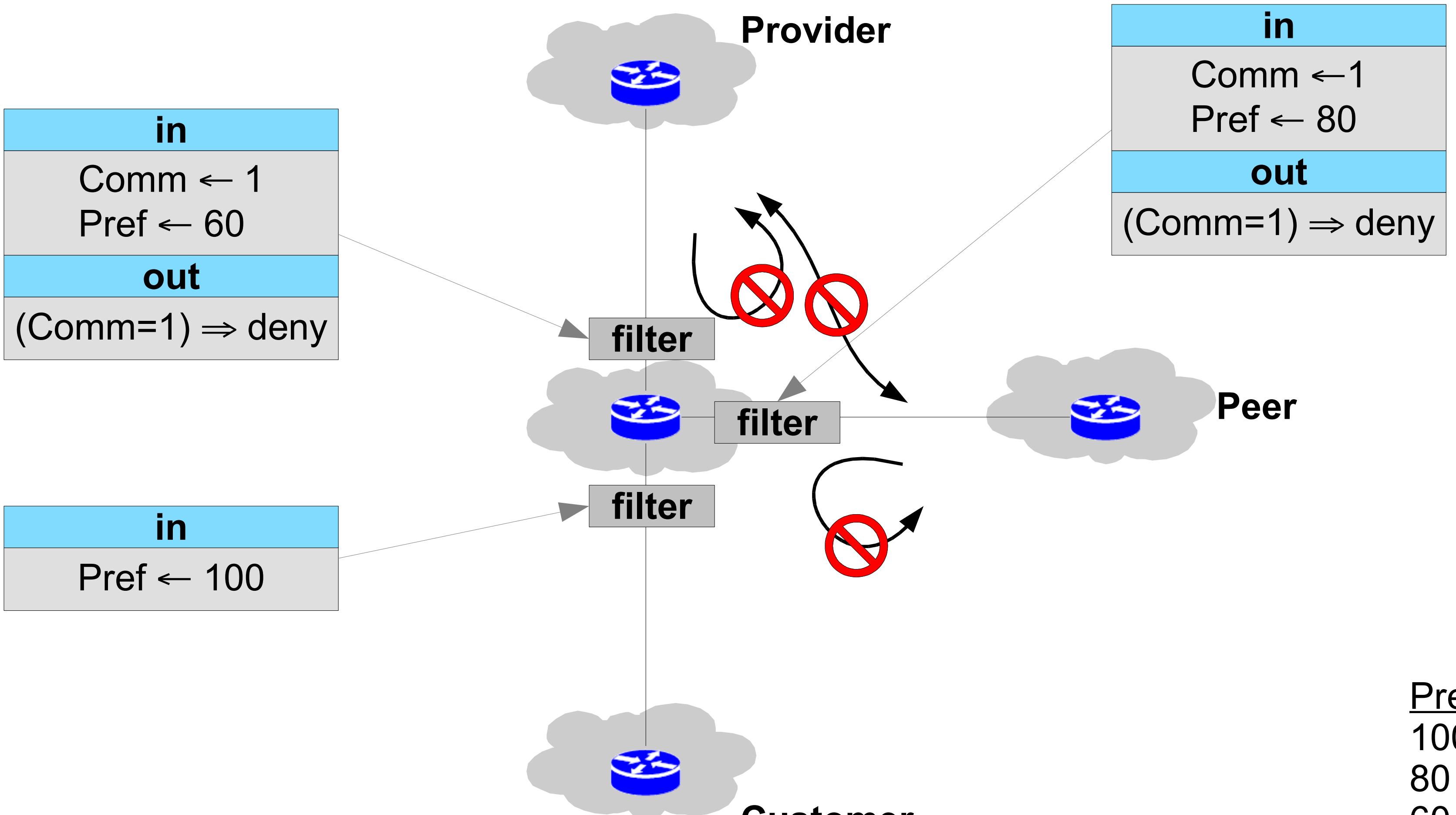
Muehlbauer et al, ACM SIGCOMM, August 2006

Example Research Studies (2)

- **Scaling Global IP Routing with the Core Router-Integrated Overlay**
  - Q: Can we reduce the size of routing tables with tunneling, mapping and virtual prefixes ?
  - C-BGP is used to compute inter-POP paths.

Zhang et al,
ICNP, November 2006

Routing Policies (1)



Preferences:
100 customer
80 peer
60 provider

Routing Policies (2)

```
cbgp> bgp router 27.06.0.0
cbgp-router> peer 11.98.0.0 show filter in
0. any --> append community 0:10
1. any --> set local-pref 80
default. any --> ACCEPT
cbgp-router> peer 11.98.0.0 show filter out
0. (comm contains 1)OR(comm contains 10) --> DENY
1. any --> remove community 1
2. any --> remove community 10
default. any --> ACCEPT
cbgp-router>
```

AS-level Topology Sources

- **Inferred AS-level topologies**
  - Infer AS-level graph and business relationships based on observed BGP paths
    - Subramanian et al
    - di Battista et al
    - CAIDA

Subramanian et al, IEEE INFOCOM 2002

Di Battista et al, IEEE/ACM Transactions on Networking. April 2007.

Dimitropoulos et al ACM SIGCOMM CCR, January 2007

- **Synthetic AS-level topologies**
  - Generate AS-level graph using preferential attachment algorithm
    - BRITE and GT-ITM (unsuitable as they do not assign business relationships to edges).
  - Generate AS-level graph using P-A + strict hierarchy. Links in same level are p2p while links between levels are p2c.
    - GHITLE

De Launois, http://ghitle.info.ucl.ac.be

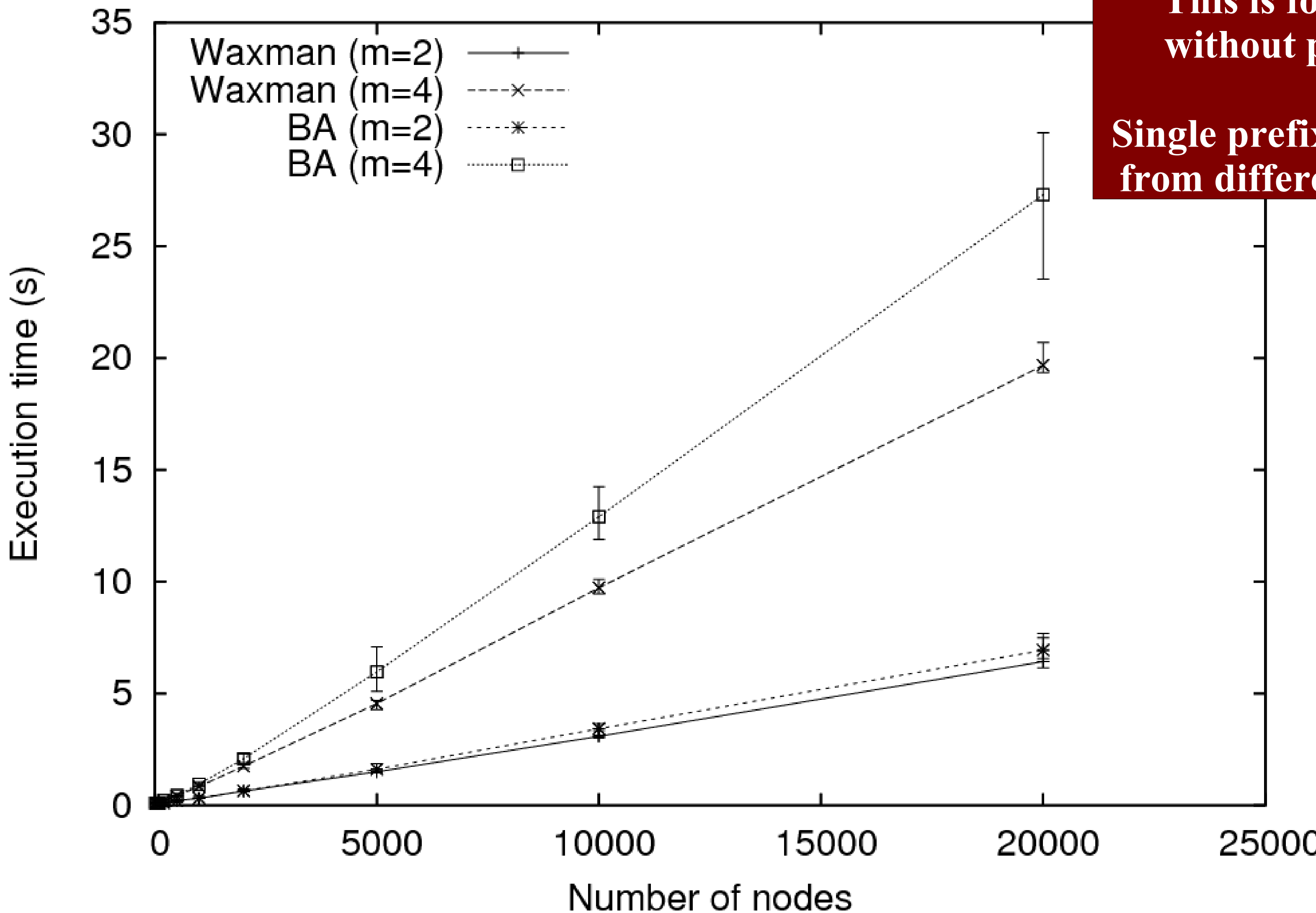
Some inferred AS-level topos contain policy cycles...

Convergence is not guarantee in this case

Challenges

- **Memory Scalability**
  - Topology size: on the order of 20k domains, 50k links
  - One prefix / AS: 20k prefixes
  - 20k nodes * 20k routing entries ≅ 400,000,000 routes to store
    - ⇒ Memory requirement is huge !
- **Time Scalability**
  - Graph structure and policies can cause expensive path exploration
    - ⇒ Simulation time is difficult to predict (can sometimes be looong) !

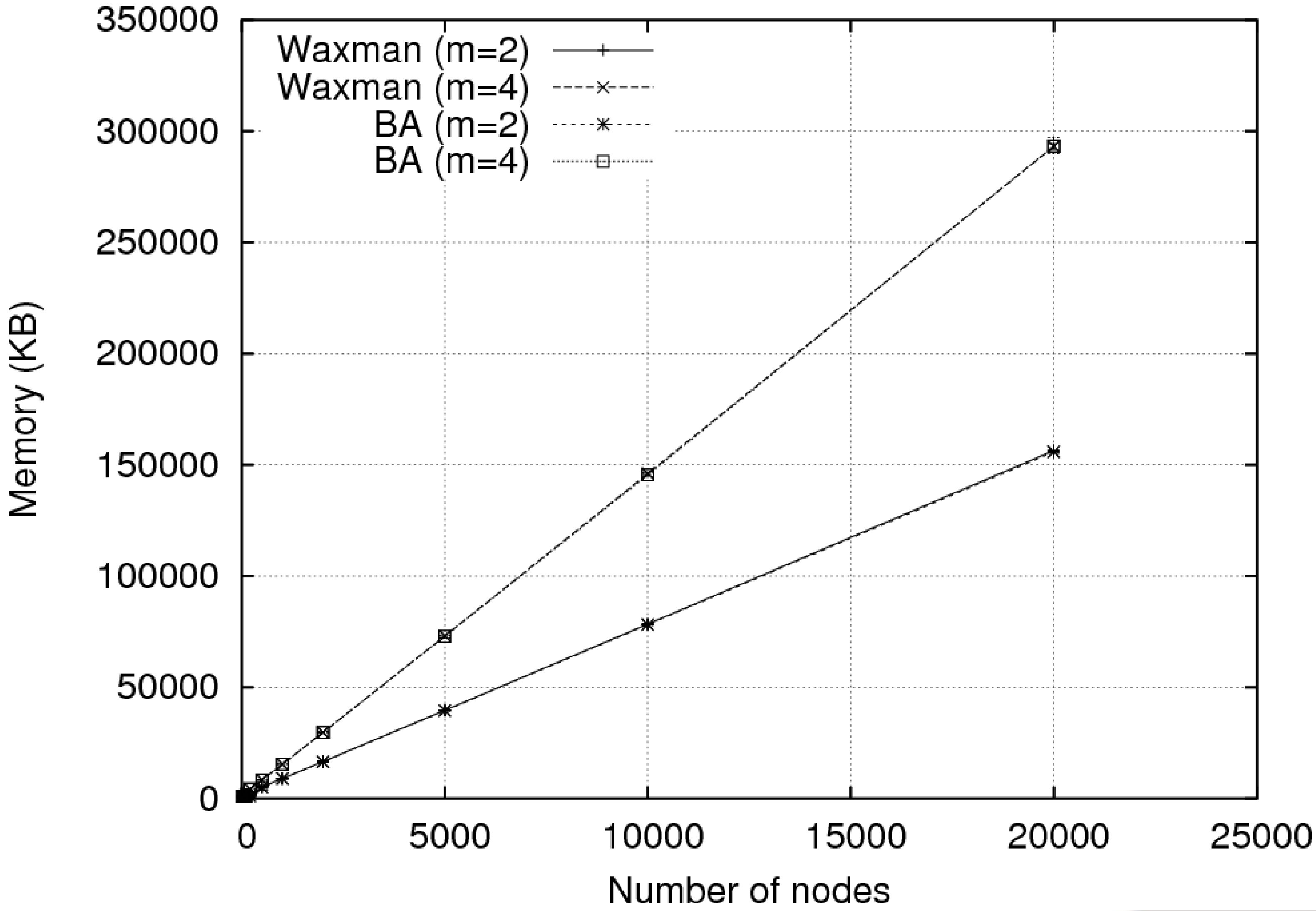
Challenges



This is for graphs without policies...

Single prefix originated from different routers.

Challenges



Waxman (m=2)
Waxman (m=4)
BA (m=2)
BA (m=4)
Memory (KB)
350000
300000
250000
200000
150000
100000
50000
0
0
5000
10000
15000
20000
25000
Number of nodes

Challenges

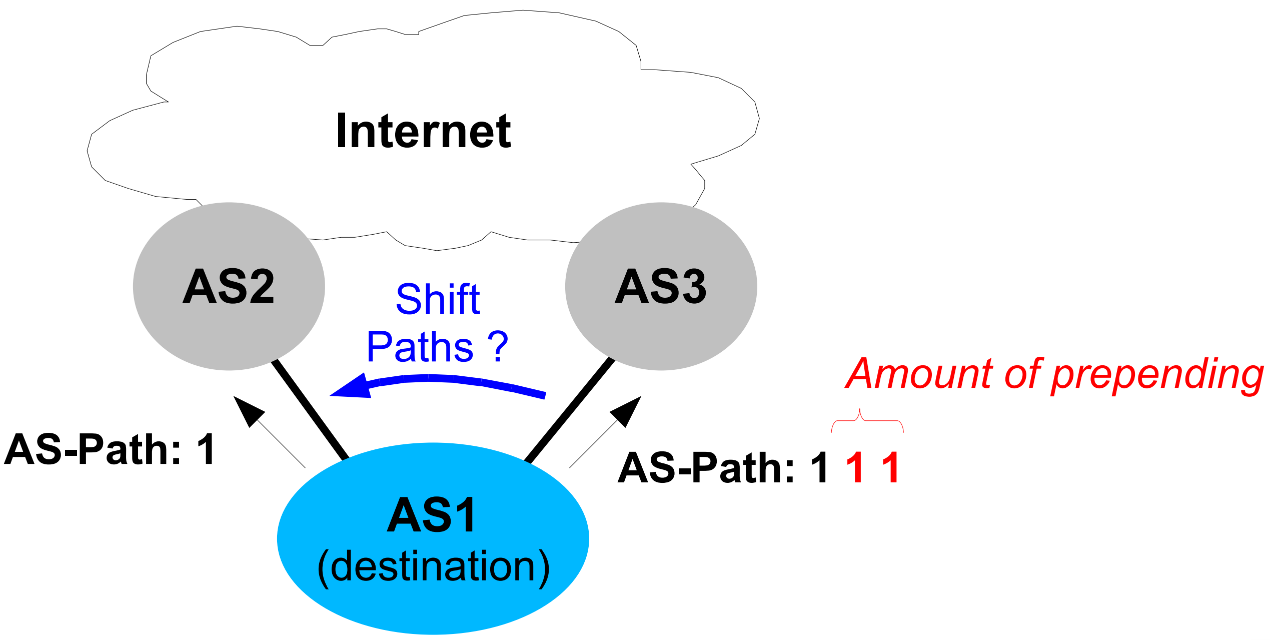
- **Increasing memory scalability**
  - Swap computed routes onto disk
  - Many stubs have identical providers (if policies are equal, their prefixes will be propagated equally)
    - Keep a single instance of equally connected stubs
  - Hao and Koppol's memory reduction technique [to be done]

- **Increasing time scalability**
  - Reduce path exploration due to address assignment. Use intended non-determinism in BGP tie-breaking rule.
  - Perform computation on multiple threads / CPUs [ongoing work]
    - Propagation of two prefixes is independent (“prefix slicing”).

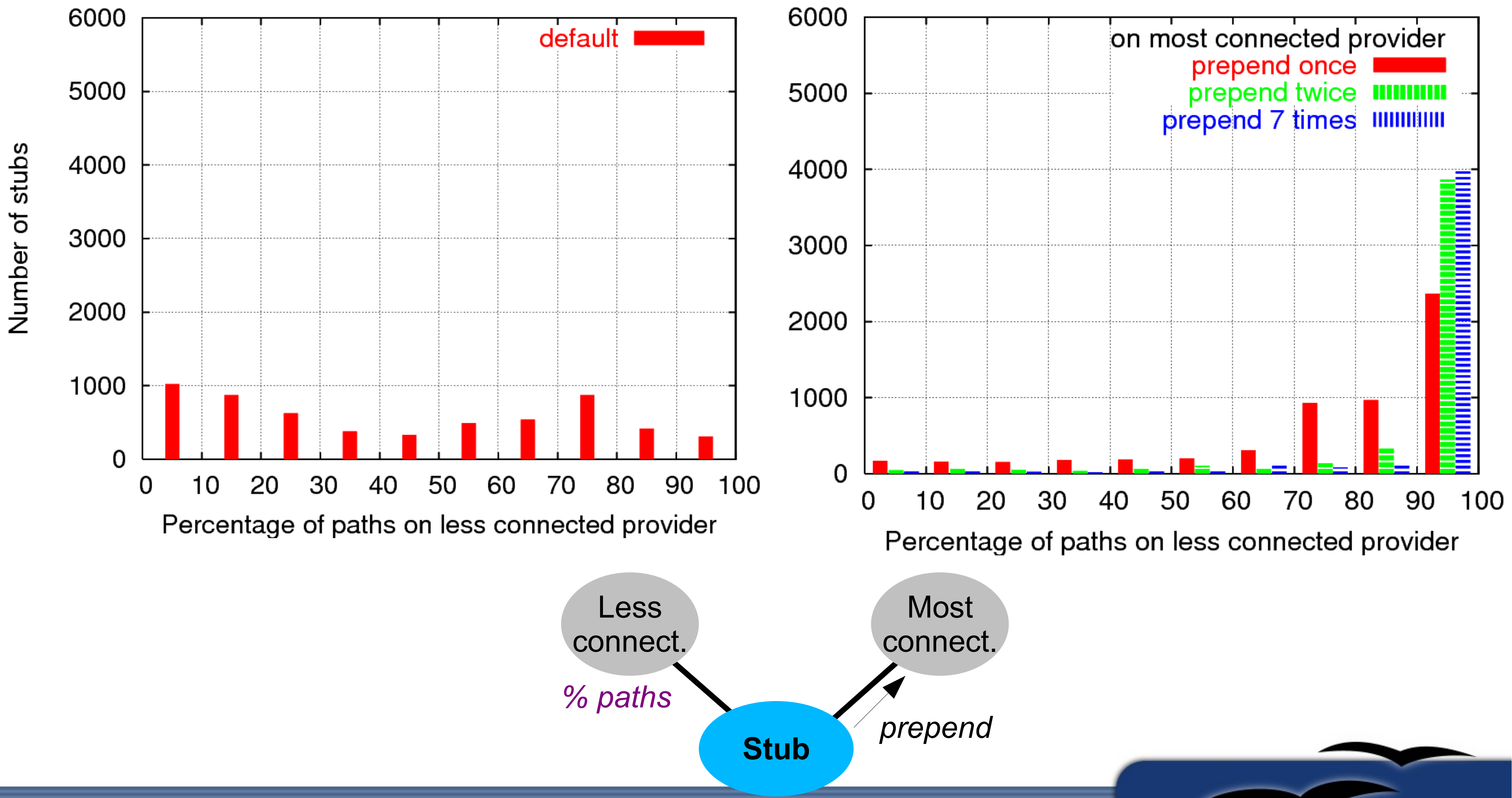
Case Study: ASPP

- **AS-Path Prepending**
  - Make AS-Path artificially longer so that distant routers dislike the route through one access link
  - Widely used technique

    Broido et al, European Transactions on Telecommunications, 2002
  - No large scale performance evaluation



Case Study: ASPP



Agenda

- Introduction
- BGP Modeling
- C-BGP
- Modeling an ISP
- Modeling the Internet
- **Conclusion**

Conclusion
&
Future Research Directions...

Conclusion

- **Future C-BGP Improvements**
  - BGP/MPLS VPNs
  - Scalability
  - IPv6 ?
  - Scalability (again)

- **Research Directions**
  - IBGP organization (RR, liBGP, route servers)
  - Incompatible routing policies (mis-configurations, oscillations)
  - Internet Routing Architecture's Scalability (HLP, LISP)...
  - More realistic synthetic topologies (Internet, router-level topologies)
    - Size: 20k domains with 1-100 routers / domain ⇒ ... !
  - More realistic policies (beyond p2c, p2p and s2s)

Vandenshrieck, RESCOM 2007

Subramanian et al, ACM SIGCOMM 2005

C-BGP
“Mini User's Guide”

Command-Line Interface (CLI)

- **C-BGP command-line interface (CLI)**
  - Script syntax (version 1.4.0)
  - CISCO-like syntax
  - Commands are organized in 3 main classes:
    - **`net`** : network topology setup + static and IGP routing
    - **`bgp`** : BGP routing
    - **`sim`** : simulation management

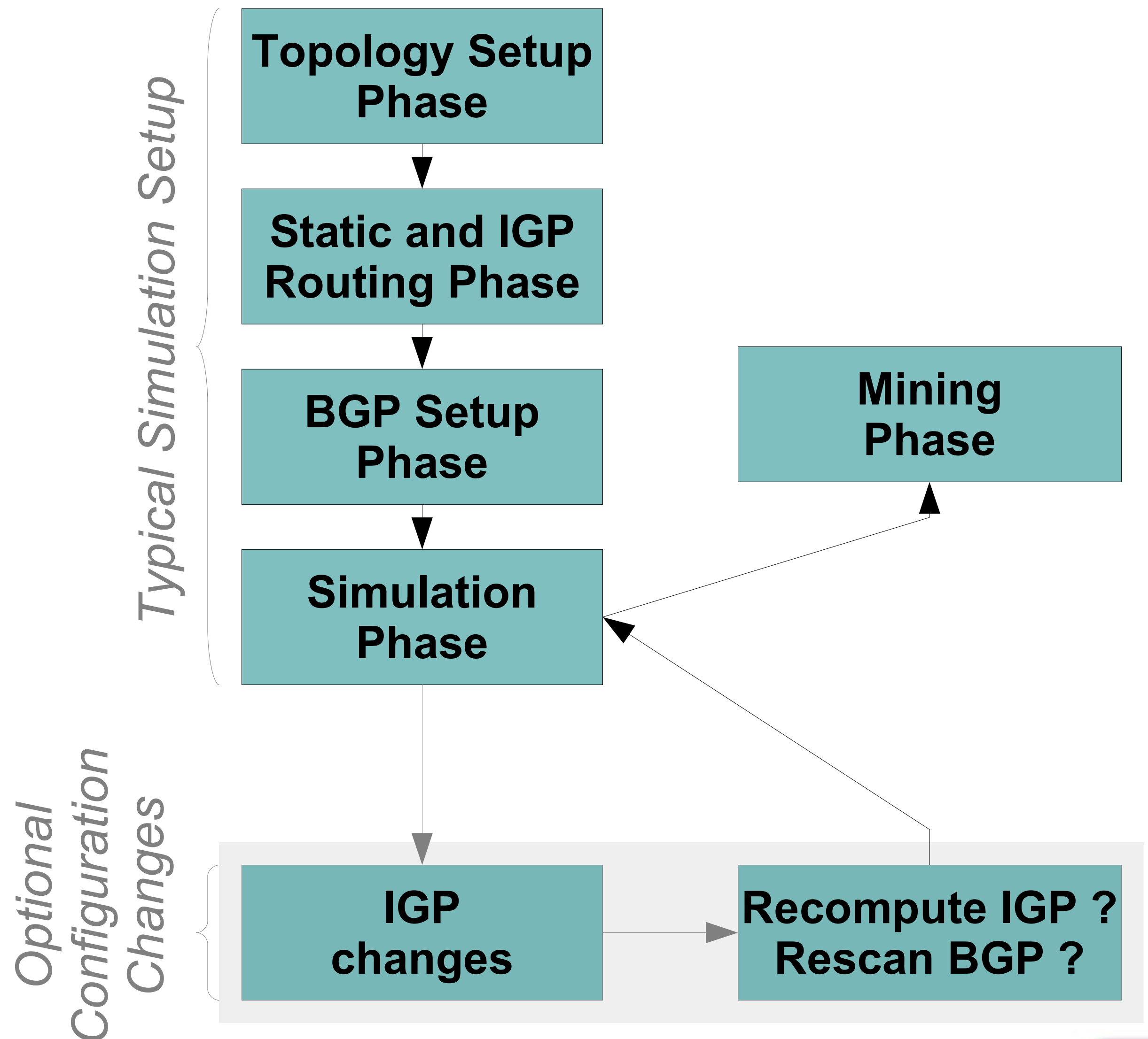
    + set of general purpose commands

```
quoitin@meat:~ $ cbgp -c myscript.cli
...
quoitin@meat:~ $ cbgp -i
cbgp> show version
cbgp version: 1.4.0-rc1
cbgp> net node 1.0.0.0 ping 2.0.0.0
...
```

"Script" mode

"Interactive" mode

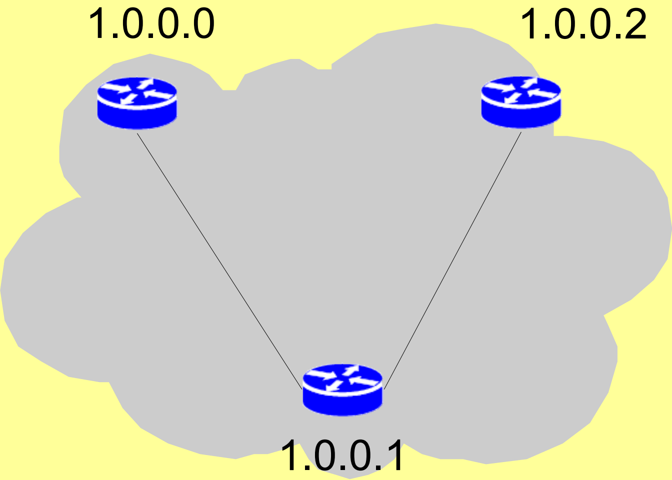
Command-Line Interface (CLI)



Example

Topology

```
net add node 1.0.0.0
net add node 1.0.0.1
net add node 1.0.0.2
net add link 1.0.0.0 1.0.0.1 0
net add link 1.0.0.1 1.0.0.2 0
```

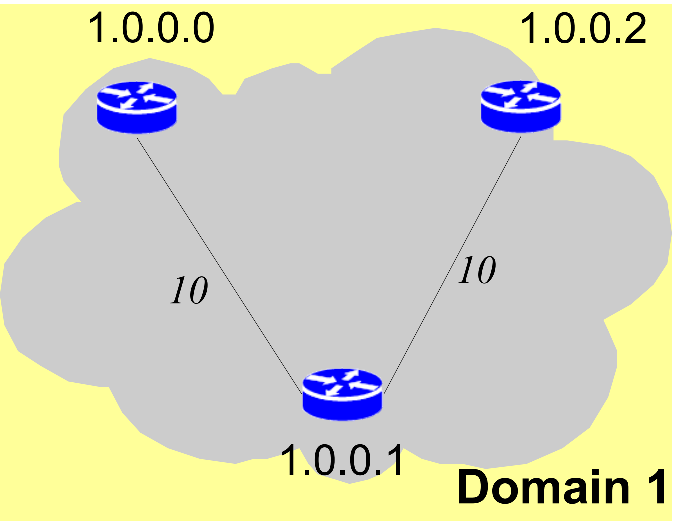


Example

Topology

IGP

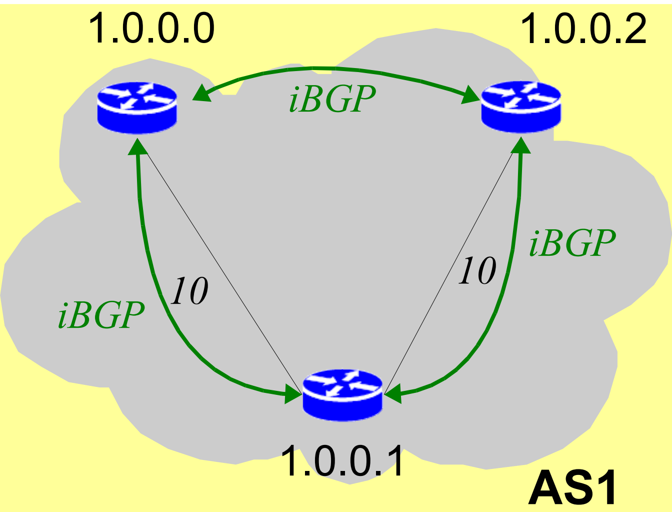
```
net add node 1.0.0.0
net add node 1.0.0.1
net add node 1.0.0.2
net add link 1.0.0.0 1.0.0.1 0
net add link 1.0.0.1 1.0.0.2 0
net add domain 1 igp
net node 1.0.0.0 domain 1
net node 1.0.0.1 domain 1
net node 1.0.0.2 domain 1
net link 1.0.0.0 1.0.0.1 igp-weight --bidir 10
net link 1.0.0.1 1.0.0.2 igp-weight --bidir 10
net domain 1 compute
```



Example

Topology / IGP / BGP

```
net add node 1.0.0.0
net add node 1.0.0.1
net add node 1.0.0.2
net add link 1.0.0.0 1.0.0.1 0
net add link 1.0.0.1 1.0.0.2 0
net add domain 1 igp
net node 1.0.0.0 domain 1
net node 1.0.0.1 domain 1
net node 1.0.0.2 domain 1
net link 1.0.0.0 1.0.0.1 igp-weight --bidir 10
net link 1.0.0.1 1.0.0.2 igp-weight --bidir 10
net domain 1 compute
bgp add router 1 1.0.0.0
bgp add router 1 1.0.0.1
bgp add router 1 1.0.0.2
bgp domain 1 full-mesh
```



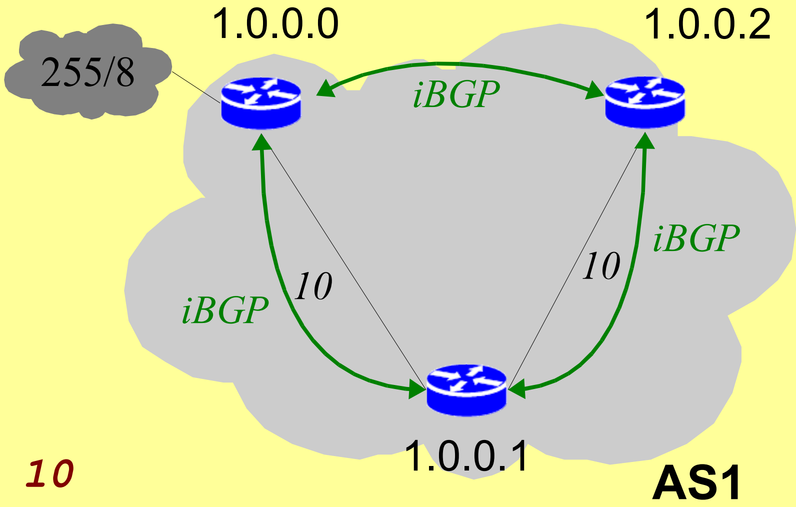
Example

Topology

IGP

BGP

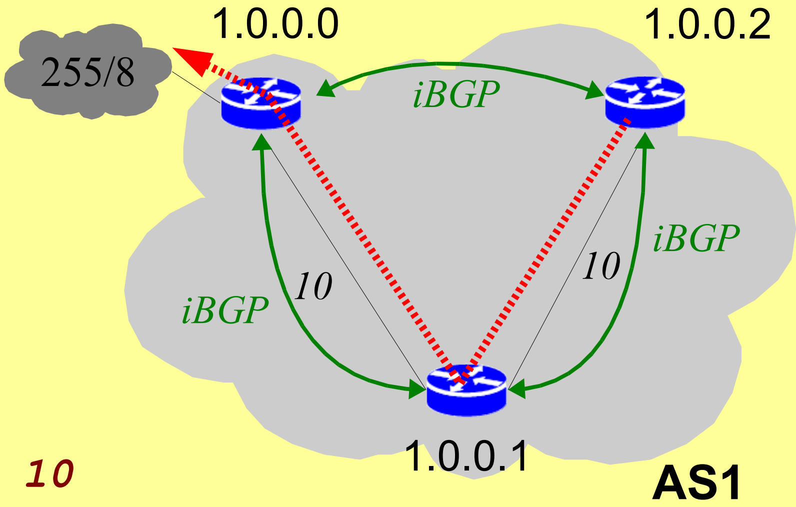
```
net add node 1.0.0.0
net add node 1.0.0.1
net add node 1.0.0.2
net add link 1.0.0.0 1.0.0.1 0
net add link 1.0.0.1 1.0.0.2 0
net add domain 1 igp
net node 1.0.0.0 domain 1
net node 1.0.0.1 domain 1
net node 1.0.0.2 domain 1
net link 1.0.0.0 1.0.0.1 igp-weight --bidir 10
net link 1.0.0.1 1.0.0.2 igp-weight --bidir 10
net domain 1 compute
bgp add router 1 1.0.0.0
bgp add router 1 1.0.0.1
bgp add router 1 1.0.0.2
bgp domain 1 full-mesh
bgp router 1.0.0.0 add network 255/8
sim run
```



Example

Topology / IGP / BGP

```
net add node 1.0.0.0
net add node 1.0.0.1
net add node 1.0.0.2
net add link 1.0.0.0 1.0.0.1 0
net add link 1.0.0.1 1.0.0.2 0
net add domain 1 igp
net node 1.0.0.0 domain 1
net node 1.0.0.1 domain 1
net node 1.0.0.2 domain 1
net link 1.0.0.0 1.0.0.1 igp-weight --bidir 10
net link 1.0.0.1 1.0.0.2 igp-weight --bidir 10
net domain 1 compute
bgp add router 1 1.0.0.0
bgp add router 1 1.0.0.1
bgp add router 1 1.0.0.2
bgp domain 1 full-mesh
bgp router 1.0.0.0 add network 255/8
sim run
net node 1.0.0.2 record-route 255.0.0.0
```




```
1.0.0.2 255.0.0.0      UNREACH 3       1.0.0.2 1.0.0.1 1.0.0.0
```

Miscellaneous

- **`show version`**

  Display C-BGP's version.

- **`include`** *F*

  Execute C-BGP script file ***F***.

- **`print`** *M*

  Print message ***M*** to the console.

Network Topology

- **`net add node`** *`X`*

  Add a new node identified by address ***X***.

- **`net add link`** *`X Y D`*

  Add a link between nodes ***X*** and ***Y*** (with informational delay ***D***).

- **`net add subnet`** *`P T`*

  Add a subnet with prefix ***P*** and type ***T*** (type is **`transit`** or **`stub`**).

- **`net add link`** *`X P D`*

  Add a link from node ***X*** to a subnet ***P*** (with info delay ***D***). Note the network part of ***P*** identifies the interface of ***X*** on the subnet.

- **`net link`** *`X Y / P`* **`up`** *`/`* **`down`**

  Change the status of link between nodes ***X*** and ***Y*** (or subnet ***P***).

Static Routing

- **net node** *X* **route add** *P I G W*

  Add a static route to ***P*** in node ***X***. ***I*** specifies the outgoing interface and ***G*** specifies the gateway. ***W*** is the weight of the route.

- **net node** *X* **route del** *P I G*

  Remove a previously installed static route to ***P***. If there are multiple routes towards ***P***, it might be necessary to identify the route to remove by specifying the outgoing interface ***I*** and the gateway ***G***.

Intradomain Routing

- **`net add domain`** *`D T`*

  Add an IGP domain identified by the positive integer ***D***. The domain's type ***T*** can only be of **`igp`** (the type **`ospf`** is experimental).

- **`net node`** *`X`* **`domain`** *`D`*

  Set node ***X*** in domain ***D***.

- **`net link`** *`X Y`* **`igp-weight [--bidir]`** *`W`*

  Set the link weight of link ***X→Y***. If the option **`--bidir`** is mentioned, the weight is changed in both directions, i.e. Also for link ***Y→X***. Note that the default link weight is "*infinity*" (2^{32}-1) which means they are not taken into account in the SPT computation.

- **`net domain`** *`D`* **`compute`**

  Compute routes in domain ***D***.

Ping and Traceroute

- **`net node`** *X* **`ping`** *Y*

  Send an *ICMP echo-request* from node ***X*** to address ***Y***.

- **`net node`** *X* **`traceroute`** *Y*

  Send *ICMP echo-requests* with increasing TTL values from node ***X*** to address ***Y***.

- **`net node`** *X* **`record-route`** *Y* / *P*

  Record the router-level path from node ***X*** to node ***Y*** or network ***P***. Compared to the above traceroute command, the record-route command does not need the existence of a reverse route to ***X*** to forward the *ICMP echo-replies*.

BGP Setup

- **`bgp add router`** *A R*

  Setup BGP on node ***R*** (which becomes a BGP router). The AS number (ASN) of ***R*** is ***A***.

- **`bgp router`** *R* **`add peer`** *A N*

  Add a neighbor ***N*** to router ***R***. ***N*** is in AS with ASN ***A***. By default, the session is configured to accept all routes in both directions (see the section on filters).

- **`bgp router`** *R* **`peer`** *N* **`up`** / **`down`**

  Change the state of the BGP session between ***R*** and ***N***.

- **`bgp router`** *R* **`peer`** *N* **`next-hop-self`**

  Configure neighbor ***N*** of router ***R*** to set its own address as BGP next-hop.

- **`bgp router`** *R* **`add network`** *P*

  Originate a network with prefix ***P*** from router ***R***.

BGP Filters (1)

- **bgp router** *R* **peer** *N* **filter in** / **out**

  Specify a input / output filter on session between router ***R*** and its neighbor ***N***.

- ...**add-rule** / **insert-rule** *I* / **remove-rule** *I*

  Add a rule to the filter, insert a rule at position ***I*** or remove filter at position ***I***.

- ...**match** "*M*"

  Specify the conditions for applying this filter rule. ***M*** is a predicate expressing the condition.

- ...**action** "*A*"

  Specify the action ***A*** to take when the above filter rule's condition is true.

BGP Filter Predicates

- `any`
  Match any route.
- `community is` *C*
  Match if the route has a community equal to ***C***.
- `nexthop in` *P*
  Match if the route's next-hop in prefix ***P***.
- `nexthop is` *A*
  Match if the route's next-hop equals address ***A***.
- `prefix in` ***P***
  Match if the route's destination prefix is in prefix ***P*.**
- `prefix is` *P*
  Match if the route's destination prefix equals prefix ***P***.
- `path` *E*
  Match if the route's AS-Path matches the regular expression ***E***.

BGP Filter Actions

- **`accept`**

  Accept the route.

- **`deny`**

  Deny the route.

- **`local-pref`** *L*

  Set the route's Local-Preference to value ***L***.

- **`metric`** *M* / **`internal`**

  Set the route's Multi-Exit-Discriminator to value ***M*** or to the IGP weight of the route towards the BGP next-hop if **`internal`** is mentioned.

- **`as-path prepend`** *N*

  Prepend the route's AS-Path ***N*** times.

- **`community add`** *C*

  Add the community value ***C*** to the route's communities list.

- **`community strip`**

  Clear the route's communities list.

BGP Route-reflection

- **`bgp router`** *R* **`peer`** *N* **`rr-client`**

  Specify that the neighbor ***N*** of router ***R*** is a route-reflector client.

- **`bgp router`** *R* **`cluster-id`** *C*

  Set to ***C*** the cluster-ID of router ***R***.

Injecting real BGP data

- **`bgp router`** *R* **`load`** *F*

  Load into router ***R*** the routes from the MRT ASCII file ***F***.

- **`bgp router`** *R* **`peer`** *N* **`virtual`**

  Define peer ***N*** of router ***R*** as a virtual router, i.e. router ***N*** does not really exist in the simulation model.

- **`bgp router`** *R* **`peer`** *N* **`recv`** *M*

  Inject a BGP message (update/withdraw) to router ***R*** as if it was coming from neighbor ***N***. The neighbor ***N*** must be virtual. The message ***M*** must be expressed in MRT ASCII format.

Simulation

- **`sim run`**

  Run the simulation until it has converged.

- **`sim step`** *N*

  Advance by ***N*** steps in the simulation.

- **`sim stop at`** *N*

  Limit the number of steps of the simulation to ***N***.

- **`sim queue show`**

  Show the pending event set's content.

Mining the BGP routing state

- **`bgp router`** *R* **`dp-debug`** *P*

  Show the BGP decision process's steps in router ***R*** for destination prefix ***P***.

- **`bgp router`** *R* **`show rib`** *P* / *A* / **`*`**

  Show the content of the BGP RIB of router ***R*** for destination prefix ***P*** (exact match is used), destination address ***A*** (longest-match is used) or all routes (using “**`*`**”).

- **`bgp router`** *R* **`show rib-in`** *N P* / *A* / **`*`**

  Show the content of the BGP Adj-RIB-in of router ***R*** for neighbor ***N*** and destination prefix ***P*** (exact match is used), destination address ***A*** (longest-match is used) or all routes (using “**`*`**”).

- **`bgp router`** *R* **`record-route`** *P*

  Record the AS-level route from router ***R*** to the origin of destination prefix ***P***.

Updating the Simulation State

- **`net domain`** *`D`* **`compute`**

  Recompute routes in IGP domain ***D***.

- **`bgp router`** *`R`* **`rescan`**

  Scan the router ***R*** for BGP routes whose path to the BGP next-hop has changed in state (up/down) or weight. Rerun the BGP decision process for the impacted destination prefixes.

- **`bgp domain`** *`A`* **`rescan`**

  Perform "**`bgp router`** *`R`* **`rescan`**" for each router ***R*** in AS ***A***.

Loading AS-level Topologies (1)

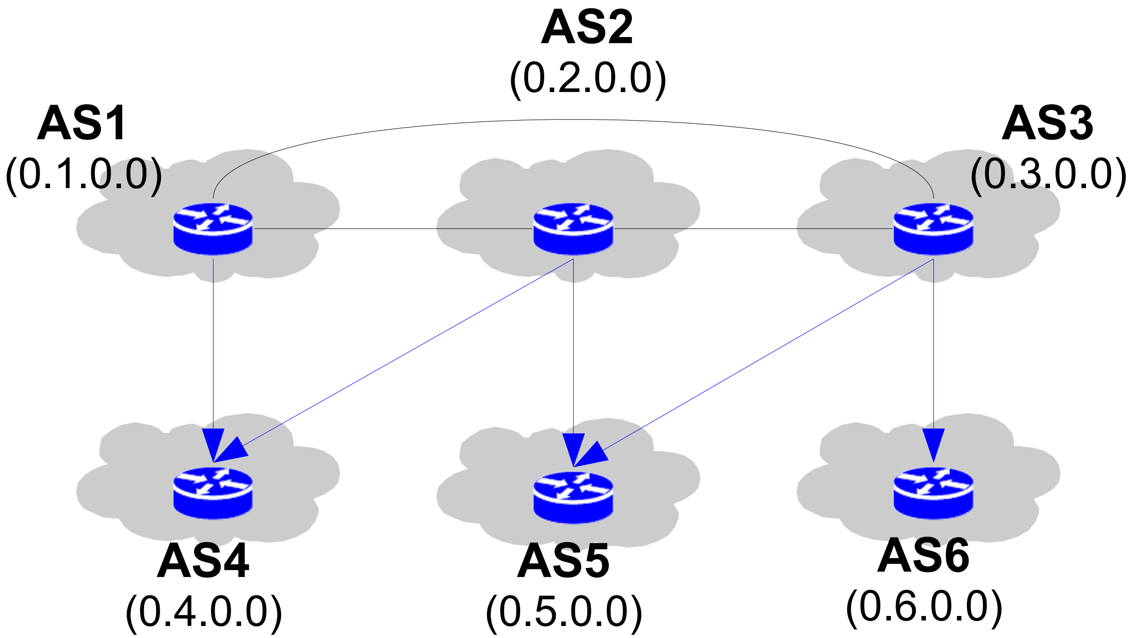
- **`bgp topology load [--format=T] F`**

  Load an AS-level topology from file ***F***. The default file format is “Subramanian”. The **`--format`** option allows to specifiy the **`caida`** format.

-

- **Subramanian's file format**

```
# AS1 AS2 Relation
1       2   0
1       3   0
1       4   1
2       3   0
2       4   1
2       5   1
3       5   1
3       6   1
```



Loading AS-level Topologies (2)

- **`bgp topology filter`** *T*

  Filter the topology from specific nodes/links class. Example for ***T***: **`stubs`** (remove all domains without customers).

- **`bgp topology install`**

  Install the topology (create routers and links).

- **`bgp topology policies`**

  Install policies on BGP sessions according to business relationships.

- **`bgp topology run`**

  Set all BGP sessions as administratively up.

- **`bgp topology record-route [--output=`***F***`]`** *P*

  Record the AS-level route from all Ases to prefix ***P***. Optionally write the output to file ***F***.

Hands on...

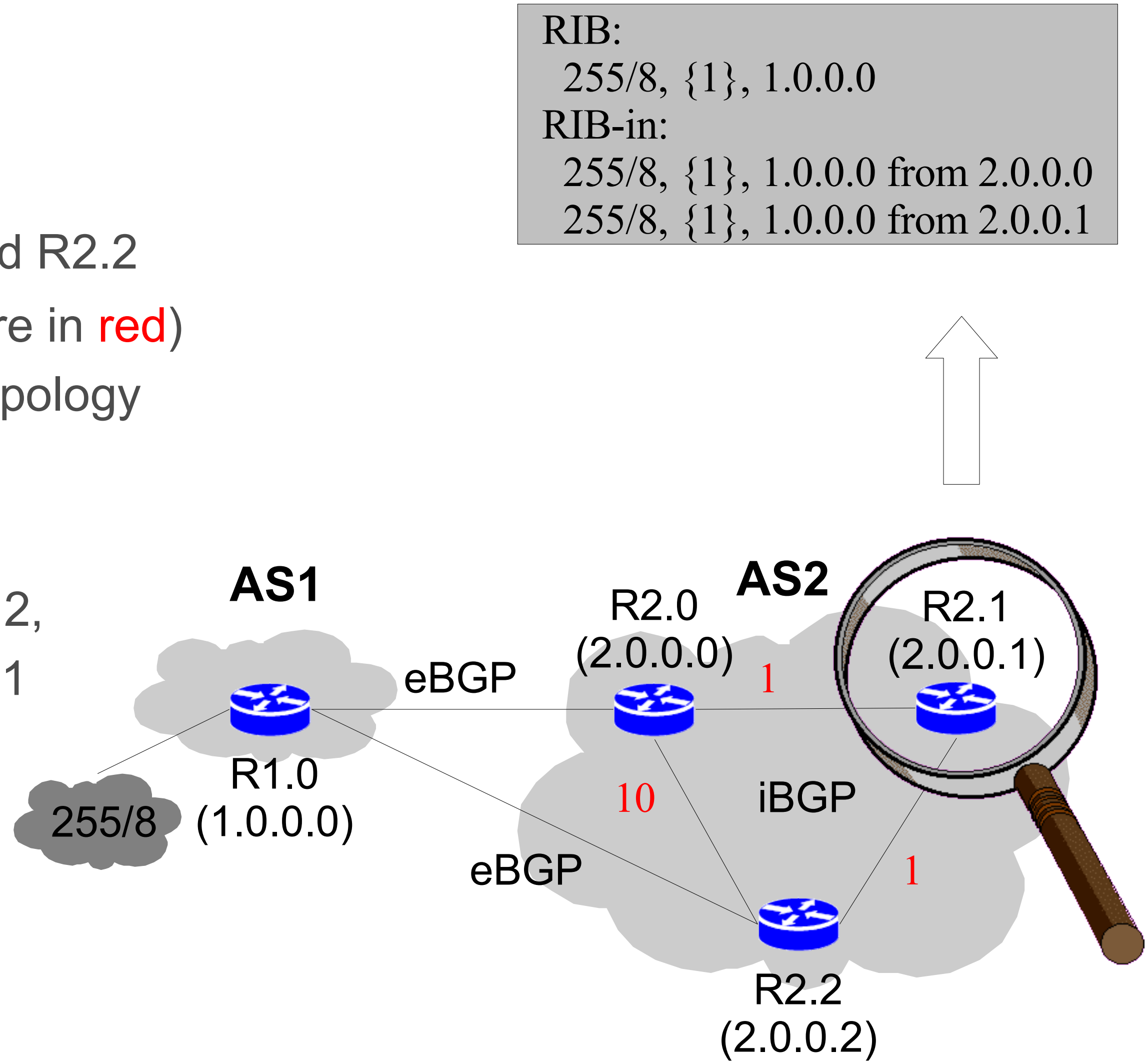
Installation

- **Pre-installed on server**
  - `ssh` to `meat.eecs.jacobs-university.de`
  - Run from `/home/quoitin/local/bin/cbgp`
  - Version 1.4.0-rc1 (thanks for beta-testing)

- **On your own laptop**
  - URL: http://cbgp.info.ucl.ac.be/emanics.php
  - Prerequisites:
    - `libgds`
    - `libpcre`, `libpcre-dev`, `libreadline`, `libreadline-dev`
  - Installation with `./configure` + `make`

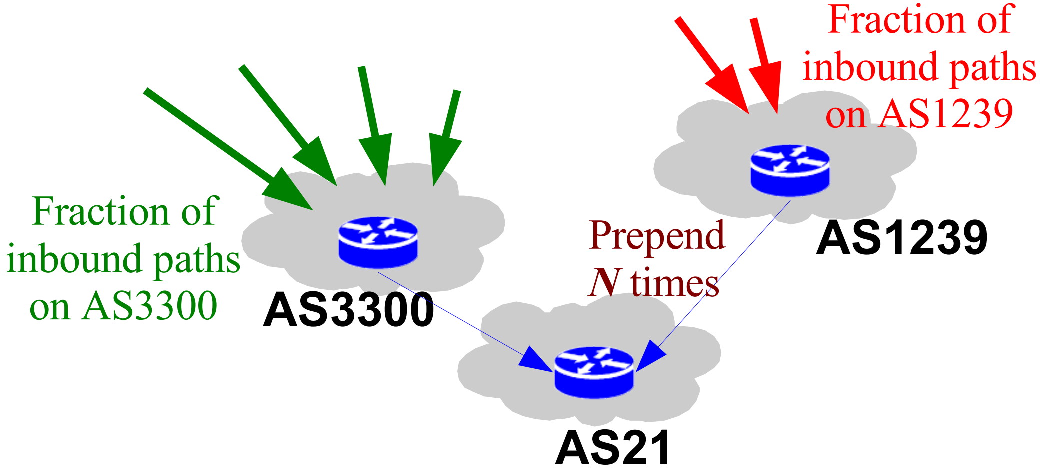
Exercise 1

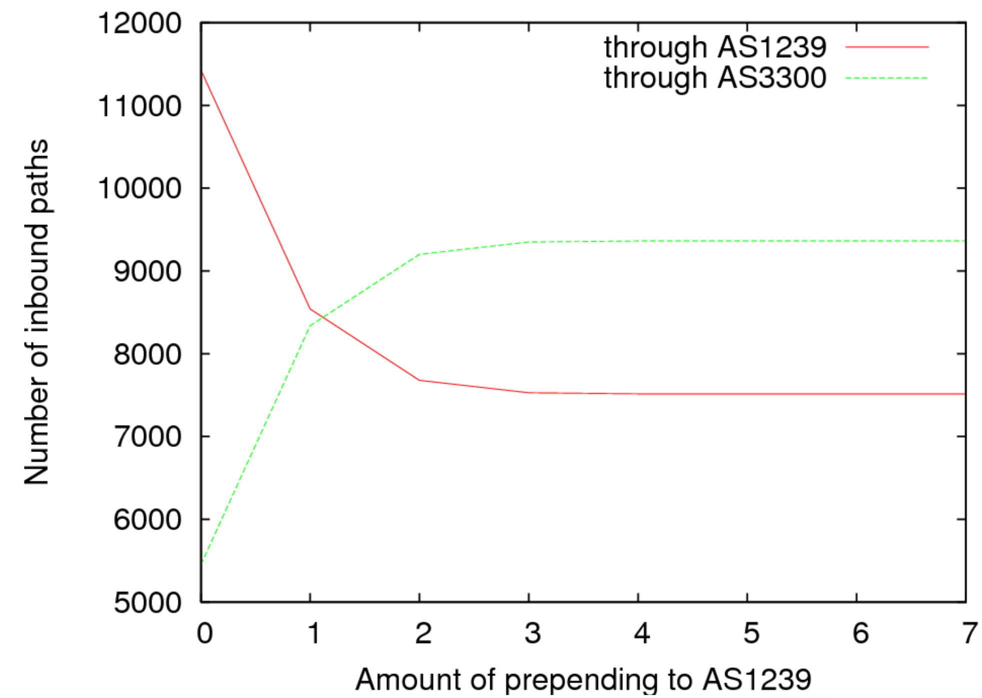
- **Simple setup**
  - AS1 composed of R1.0
    - Originates prefix 255/8
  - AS2 composed of R2.0, R2.1 and R2.2
    - Runs an IGP (link weights are in red)
  - BGP topology equals physical topology
  - Compute routes in
    - (a) default state,
    - (b) after the failure of R1.0-R2.2,
    - (c) after the failure of R2.0-R2.1
  - Look from R2.1



Exercise 2

- **AS-level experiment: AS-Path Prepending**
  - Topology "`subra-2004.topo`"
  - Propagate a prefix from stub AS21 (homed to AS1239 and AS3300)
  - Dump all AS-Paths
  - Change output policy to prepend ***N*** times to AS1239 (***N*** varying from 1 to 7)
  - Dump all AS-Paths (for each ***N***)
  - Compare path shifts (use your favorite scripting language to perform this computation)





Exercise 3

- **ISP model: Abilene**
  - Abilene Observatory
    - http://abilene.internet2.edu/observatory/data-collections.html
  - Simplified topology model: use "`include abilene.cli`"
  - Setup policies