Abstraction-Driven Network Verification and Design (a personal odyssey)

Geoffrey Xie
Naval Postgraduate School
xie@nps.edu

Microsoft Research
Faculty Summit
2015
July 8-9, 2015

It started in 2004

- A sabbatical at CMU
  - Joined a collaborative project with AT&T Labs
  - Goal: To reverse engineer the routing designs of 100s of production networks and find ways to detect errors early and minimize outages due to routing loops and blackholes
  - We were given only router configuration files

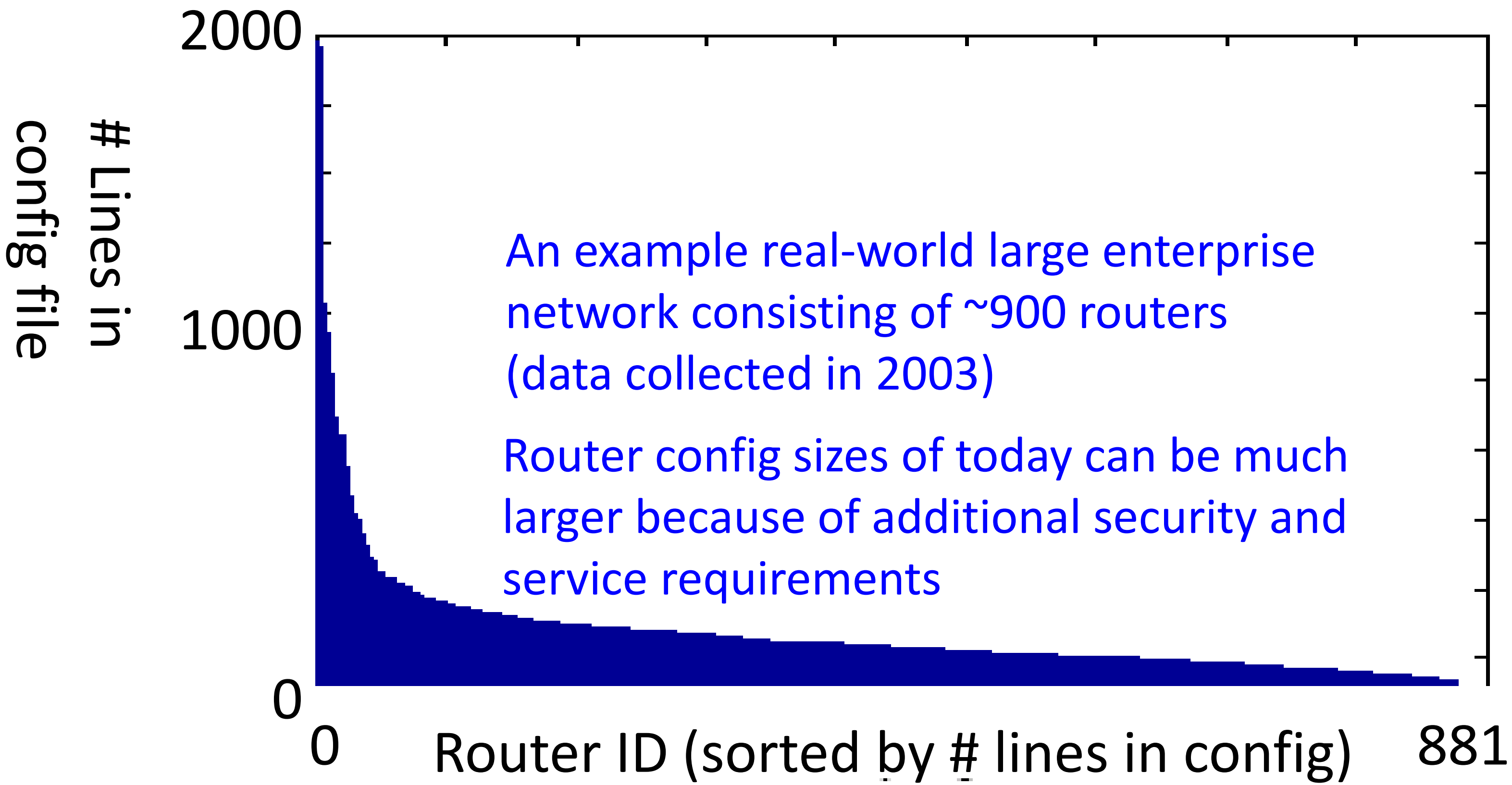
Excerpts of a Router Config

```
interface Ethernet0
  ip address 6.2.5.14 255.255.255.128
interface Serial1/0.5 point-to-point
  ip address 6.2.2.85 255.255.255.252
  ip access-group 143 in
  frame-relay interface-dlci 28

router ospf 64
  redistribute connected subnets
  redistribute bgp 64780 metric 1 subnets
  network 66.251.75.128 0.0.0.127 area 0
router bgp  64780
  redistribute ospf 64 match route-map 8aTzlvBrbaW
  neighbor 66.253.160.68 remote-as  12762
  neighbor 66.253.160.68 distribute-list 4 in
…
```

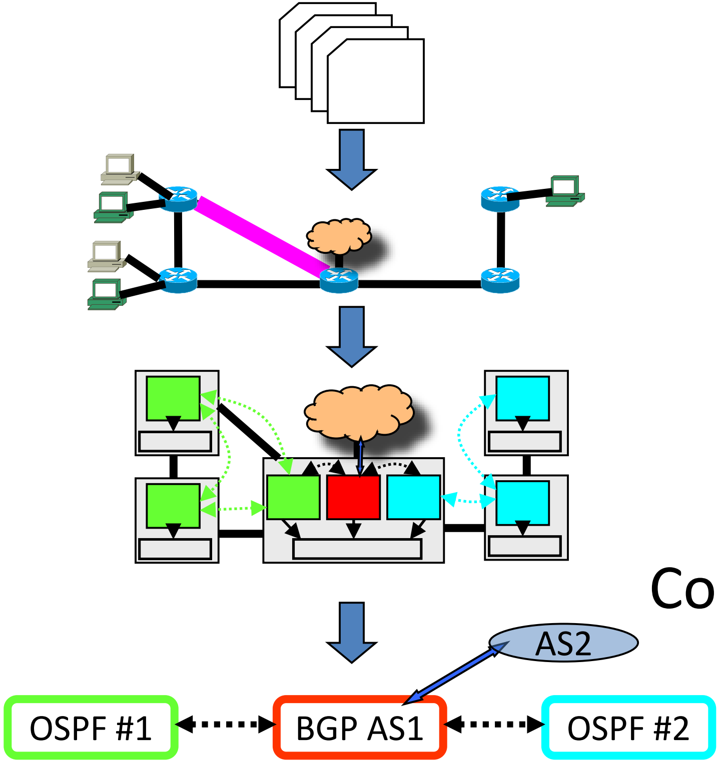
```
…
access-list 143 deny 1.1.0.0/16
access-list 143 permit any
route-map 8aTzlvBrbaW deny 10
  match ip address 4
route-map 8aTzlvBrbaW permit 20
  match ip address 7
ip route 10.2.2.1/16 10.2.1.7
…
```

Lots of Configuration Files



A Reverse-Engineering Methodology

[Maltz et al, Sigcomm'04]



Configuration files

Find links

Construct logical IP Topology

Find adjacent routing processes

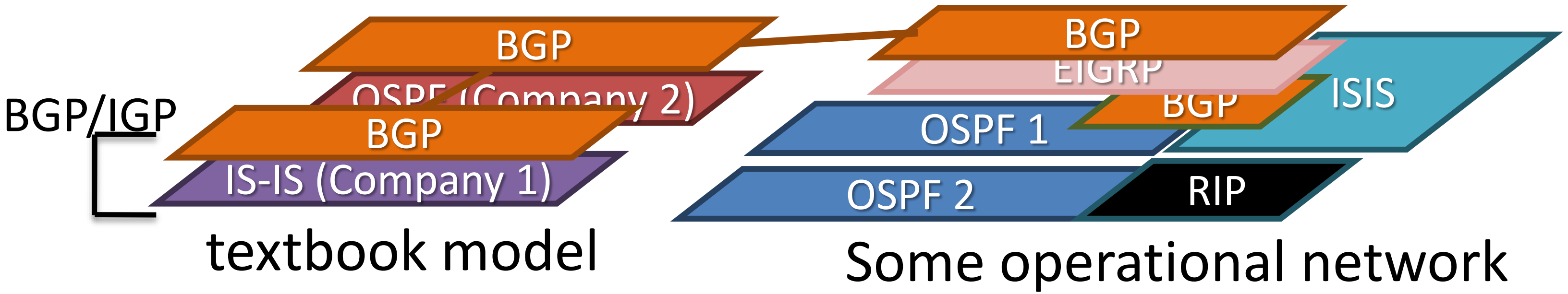
Construct Routing Process Graph

Condense adjacent routing processes into

Routing Instances

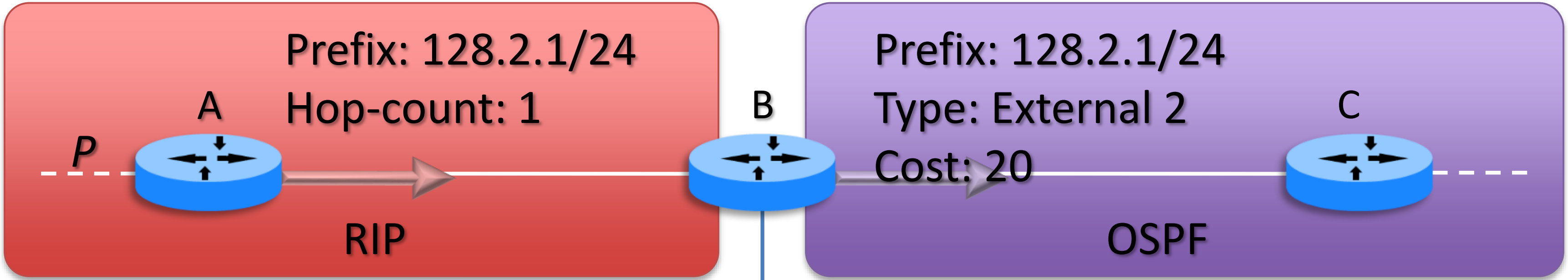
This abstraction is key to a scalable solution

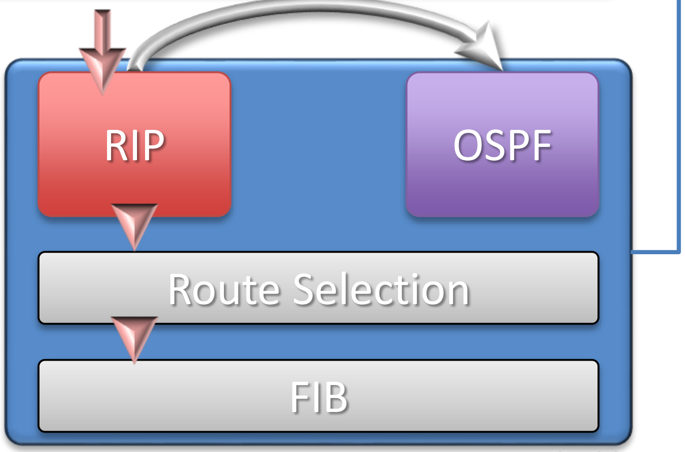
Diverse Deigns beyond Textbook Model



A network can have many routing instances and their interaction will impact routing safety

Route Redistribution

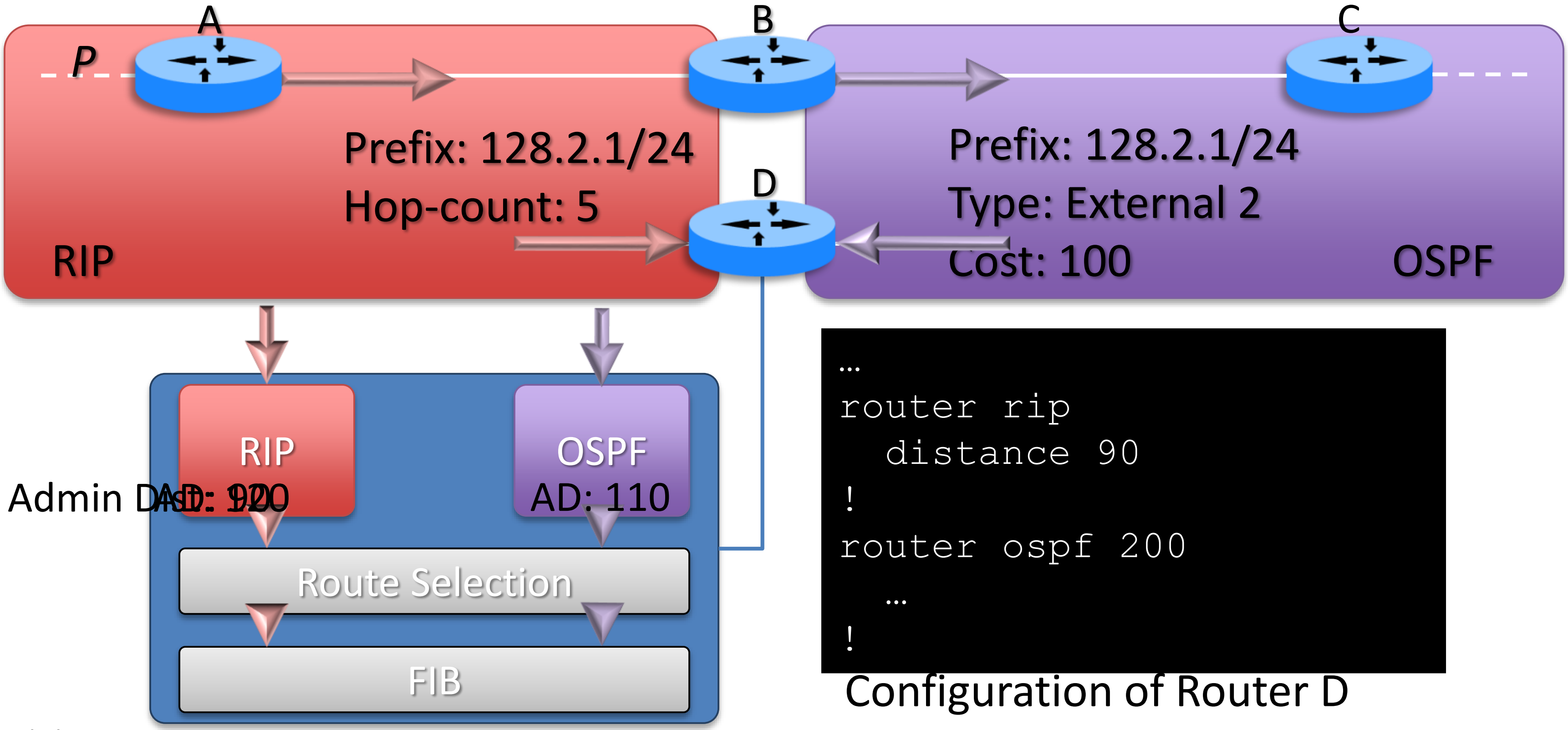





```
…
router rip
  …
!
router ospf 200
  redistribute rip metric 20
!
```

Configuration of Router B

Route Selection



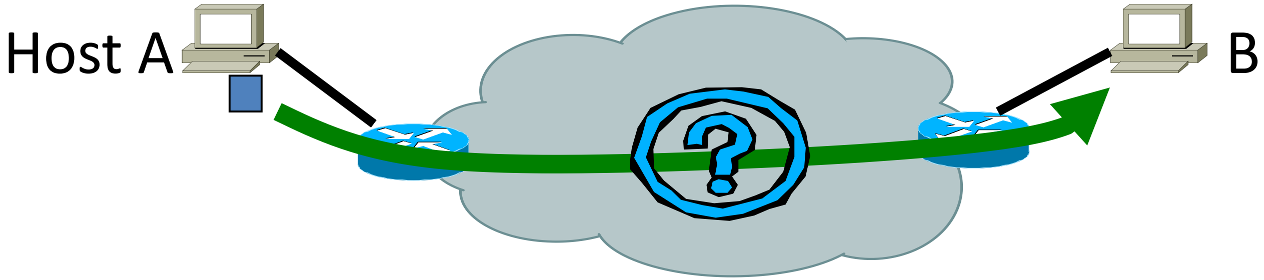

```
…
router rip
  distance 90
!
router ospf 200
  …
!
```

Configuration of Router D

Impact on Routing Theory & Practice

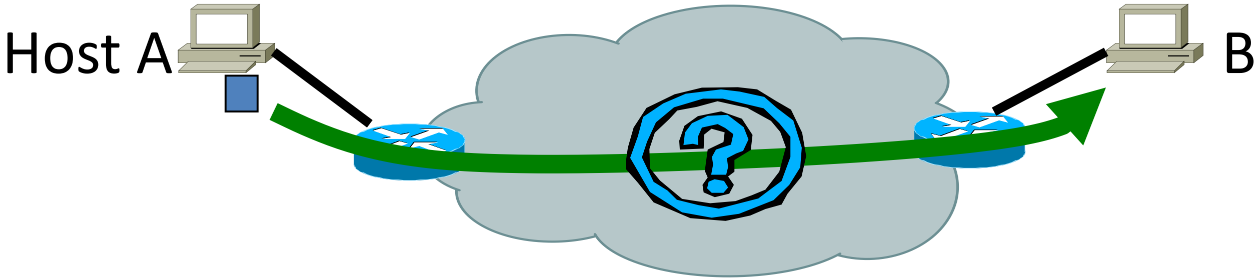
- Understanding current design and ensuring safety
  [Le et al, ICNP'07, Sigcomm'08, CoNext'10] [Benson et al,, IMC'09, NSDI'09] [Alim and Griffin, CoNext'11] [Sun et al, CoNext'12], etc.
- Clean slate design of route redistribution
  [Le et al, Sigcomm'10]
- Routing Reconfiguration
  [Vanbever et al, Sigcomm'12] [Vissicchio et al, Infocom'14] etc.
- Co-existence of multiple control planes (including SDN)
  [Volpano et al, HotSDN'14] [Vissicchio et al, Infocom'15] etc.

Next goal: Predicting Reachability



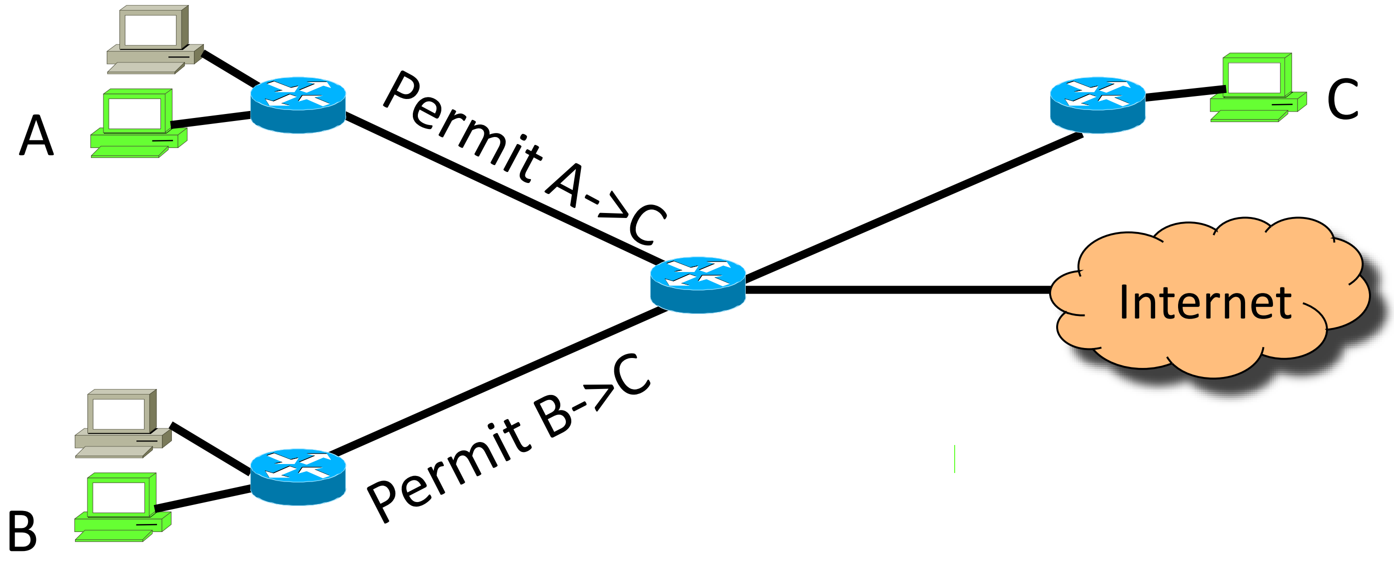
- Reachability depends not just on topology
  - Routing protocols, packet filters, and middleboxes
- Predicting reachability is key to network security and resilience

State of the Art at the time



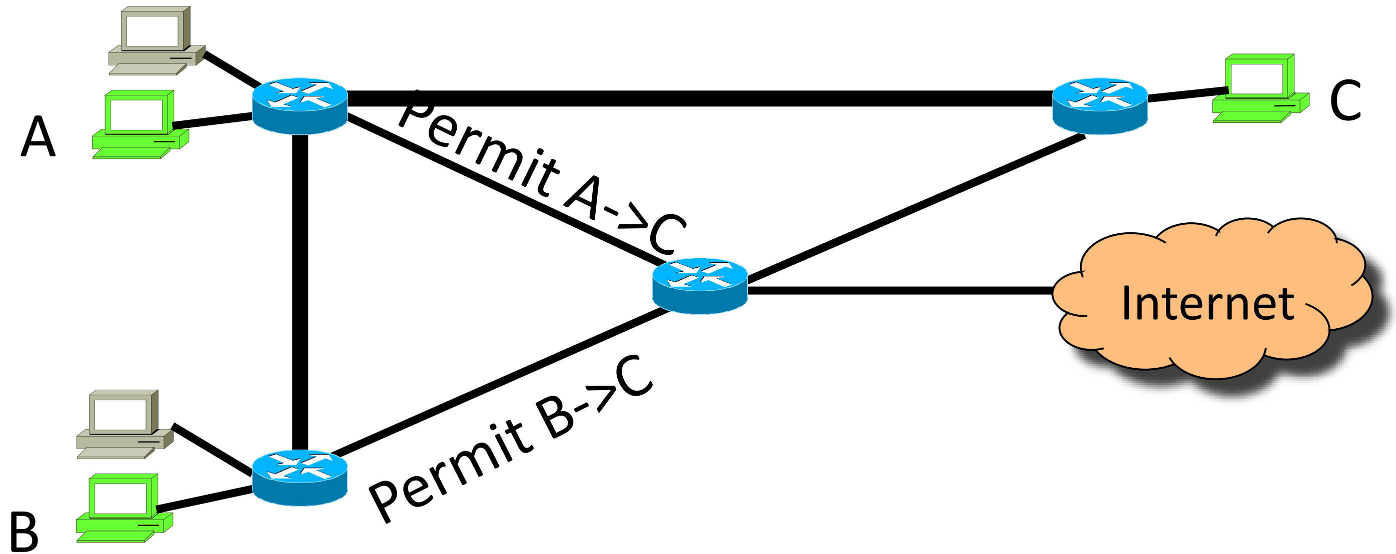
- Build the network and try it
- Dynamic probing (`ping` and `traceroute`) used to troubleshoot reachability problems

Reachability Example



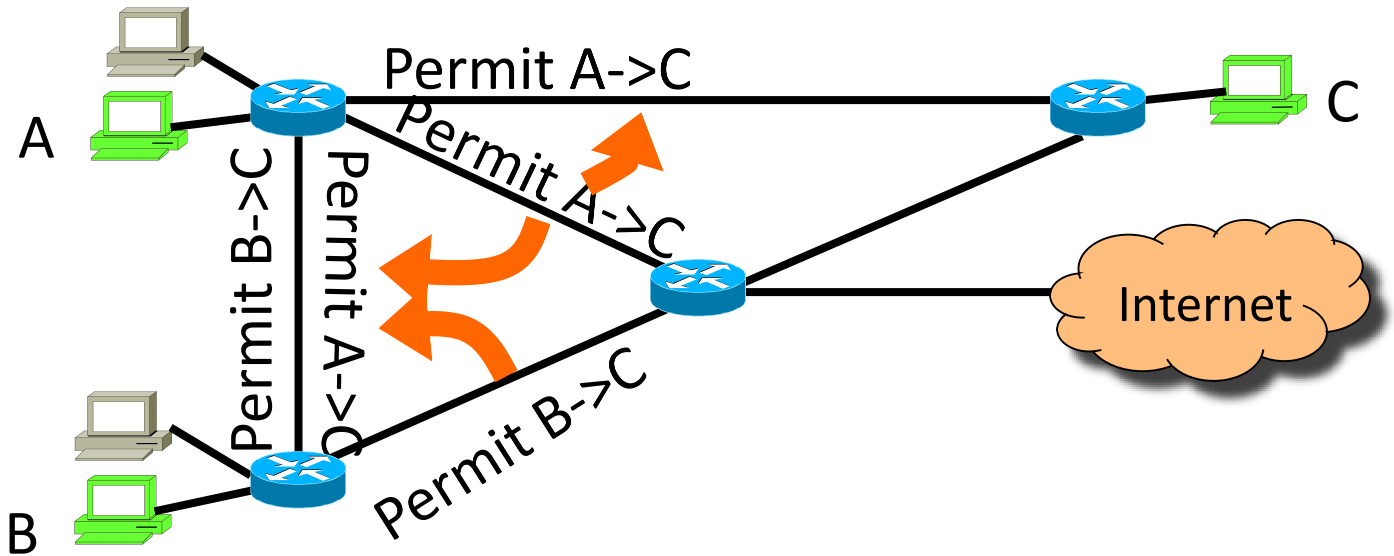
- Enterprise with two remote offices
- Only A&B should be able to talk to server C

Reachability Example



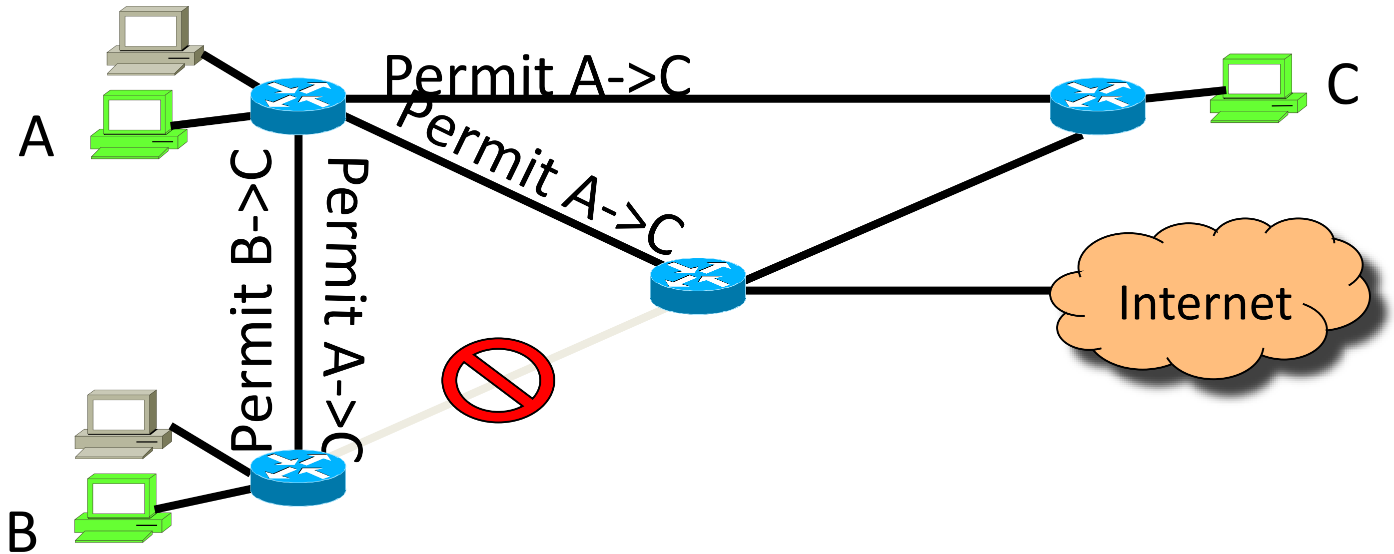
- Network designers add two links for robustness
- Configure routing protocols to use new links in failure

Reachability Example

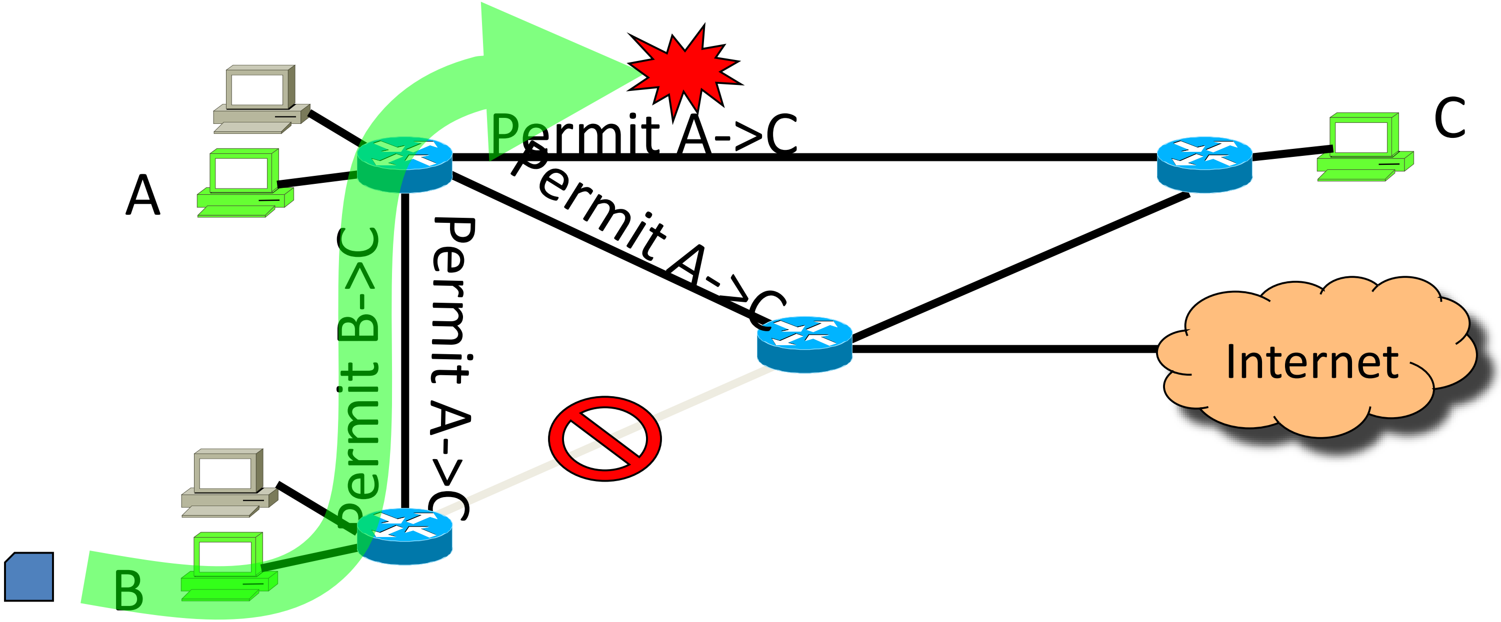


- Designers apply packet filters to new links

Reachability Example



Reachability Example



- Packets from B->C dropped!
- Testing under normal conditions won't find this error!

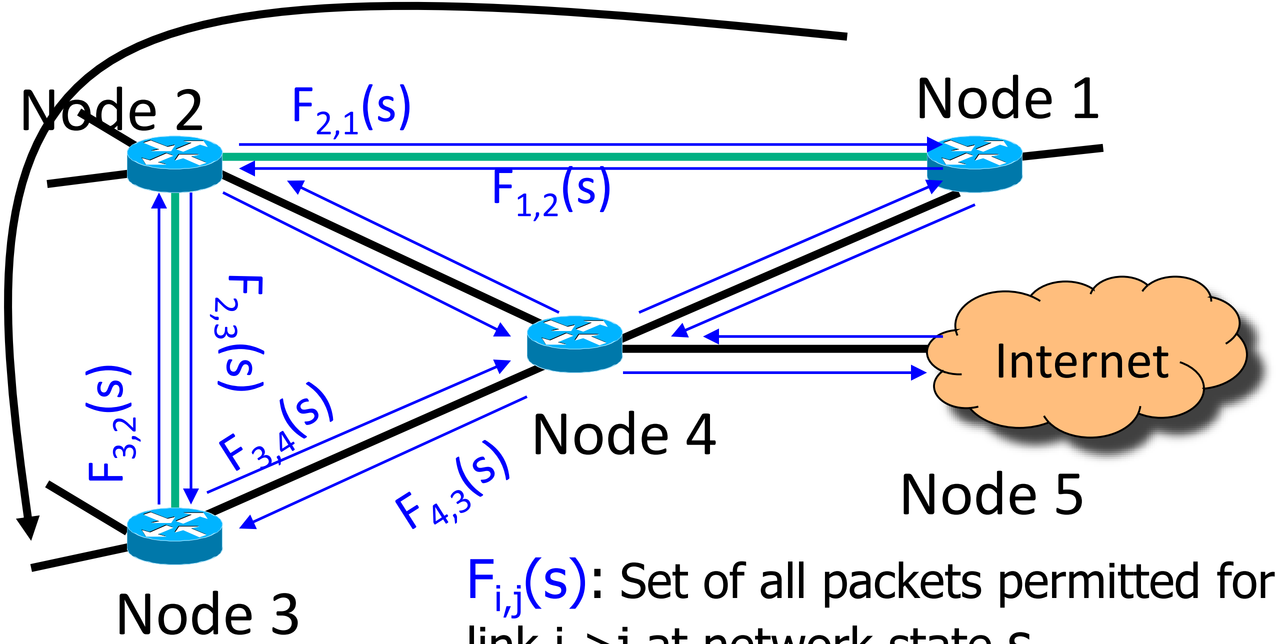
The Reachability Set abstraction

[Xie et al, Infocom'05]

Set of all packets permitted from one node to another

- Model packet filters naturally
  e.g., "Permit A->C" rule defined on link from node *u* to *v*:
  $F_{u,v}$ = {packet p | p.src_addr = A, p.dst_addr = C}
- Effect of routing protocols added as dynamic destination address based packet filters
  - when network is in forwarding state *s*,
    $F_{u,v}(s) = F_{u,v} \cap$ {all packets *u* would forward to *v* at *s*}
- Packet transformation as generalized inverse function that maps a set of packets to another set of packets

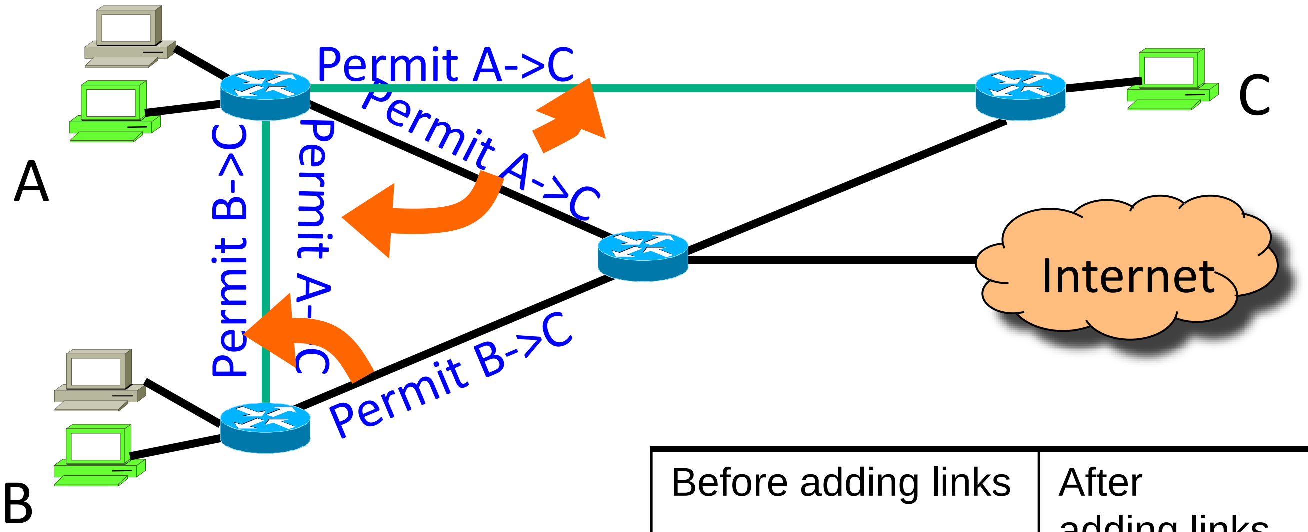
Reachability Analysis Graph



$F_{i,j}(s)$: Set of all packets permitted for link i->j at network state s

Reachable Set over directed path 1->2->3 = $F_{1,2}(s) \cap F_{2,3}(s)$

Let’s revisit that reachability example



	Before adding links	After adding links
$RS^{L}_{B->C}$	Permit B->C	(Permit B->C) ∩ (Permit A->C) = ∅ !

Recent Advances in Static Network Analysis

- Boolean satisfiability formulation [Mai et al, Sigcomm’11]
- Header space analysis [Kazemian et al, Sigcomm’12]
- Fast algorithms [Yang and Lam, ICNP’13]

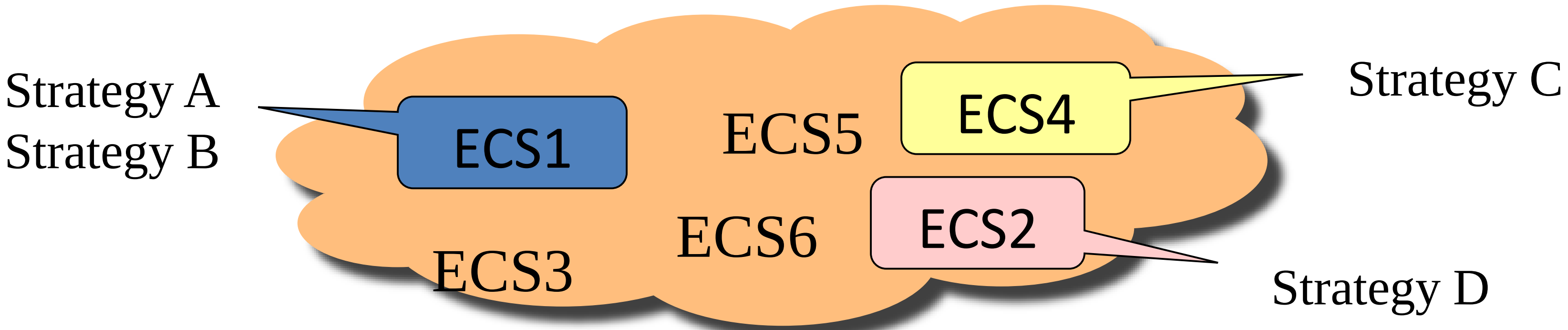
Next goal: Design automation

[Sung et al CoNext'10]

1. Abstract network-wide requirements of a design task
   - Correctness criteria for reachability control modeled as a Reachability Matrix: each cell **RS*(i,j)*** defines precisely the required reachability set from (virtual) subnet *i* to *j*
2. Formulate optimization problems
   - Incorporate resource feasibility constraints (e.g., router capacity for processing packet filter rules)
   - Model explicitly operator strategies (e.g., to deploy a minimum number of filter rules)
3. Solve formulated problems
   - Obtain new packet filter placement algorithm

Automated Packet Filter Placement

- Intuition:
  - To achieve **RS(*i, j*)**, same filters must be replicated in an edge-cut-set (ECS) between gateways of the subnets
    - Correctness guaranteed
  - Variety of heuristics possible based on design strategy which chooses particular ECS (minimizing total # of filters, balancing processing load, etc.)



All (i,j) edge-cut-sets

Some Related Design Efforts

- Integrated design methodology
  [Sun and Xie, CoNext'13]
- Optimizing the "one big switch" abstraction
  [Kang et al, CoNext'13]
- Placement of middleboxes and NFVs
  [Anwer et al, SOSR'15]

Conclusion

- A huge semantic gap exists between network service objectives and actions of individual protocols and nodes.
  - Software defined networking (SDN) doesn't reduce service objectives, while introducing a new type of nodes and diverse control apps
- Developing higher level abstractions may be key to containing this "curse of many knobs".
  - E.g., Separation of correctness and performance concerns as in traditional computer programming?