Wide-Area Routing:
The Devil is in the Configuration

Nick Feamster
M.I.T. Computer Science and Artificial Intelligence Laboratory
feamster@lcs.mit.edu

BGP Configuration Determines Its Behavior

- Route injection, redistribution, aggregation
- Import and export route maps
- Access control lists, filtering
- AS Path prepending
- Communities
- Next-hop settings
- Route flap damping
- Timer settings

BGP is a distributed program.

We need practical verification techniques.

Today: Stimulus-response Reasoning

"What happens if I tweak this import policy?"
"Let's just readjust this IGP weight..."
"New customer attachment point? Some cut-and-paste will fix that!"

Some time later, some "strange behavior" appears.
(OOPS! Revert.)

- This is a terrible "programming environment".
  - Configuration is ad hoc and painful.
  - Wastes operator time.
  - Suboptimal performance, angry customers.

Better: High-level Reasoning

- **Verify** the behavior of a particular configuration.
  - Check "correctness properties".
  - Check that the configuration conforms to intended behavior.

More than a band-aid fix.
Useful for any router configuration language.

- **Specify** configuration based on intended behavior.
  - Configuring low-level mechanisms is error-prone.
  - Specifying high-level intended behavior makes sense.

Higher Level Reasoning about "Correctness"

- **Validity:** Does it advertise invalid routes?
  - Bogus route injection, persistent forwarding loops, etc.
- **Visibility:** Does every valid path have a route?
  - Session resets, missing sessions, damped routes, etc.
- **Safety:** Will it converge to a unique, stable answer?
  - Policy-induced oscillation
- **Determinism:** Answer depend on orderings, etc.?
  - Irrelevant route alternatives can affect outcomes.
- **Information-flow control:** Expose information?
  - Accidental route leaks to neighbors, etc.

Key Challenge: Specification

- Three types of constraints to express.
  - Pattern-based: artifacts of today’s configuation languages
  - Control-flow: interaction with routing at lower "scopes" (e.g., IGP)
  - Information-flow: interaction with other participants in the same "scope" (i.e., other ASes)

We are developing a tool that checks these types of constraints.

- High-level configuration depends on specification.

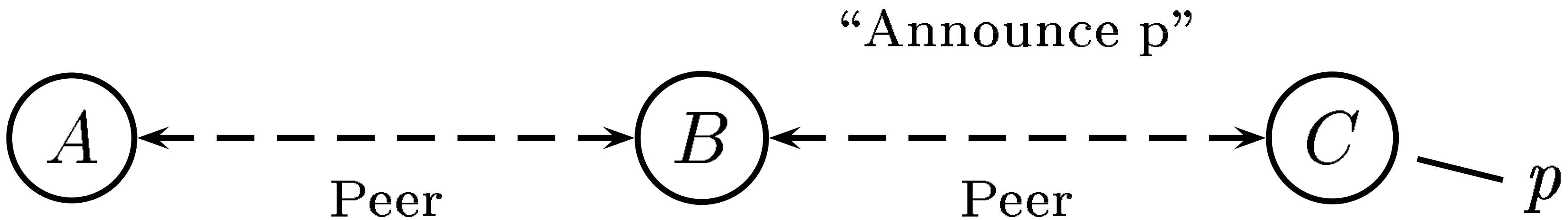
Verification also requires a specification of intent, which can inspire configuration language design.

Intent-Based Configuration: Verification is a Necessary First Step

Nick Feamster and Hari Balakrishnan
M.I.T. Computer Science and Artificial Intelligence Laboratory
{feamster,hari}@lcs.mit.edu

Example: Information-flow Control

Simple rule: don't advertise routes from one peer to other peers.

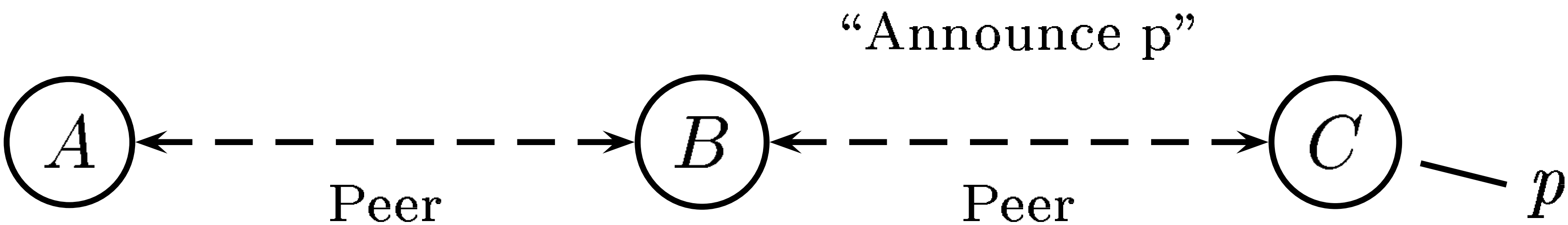


Other Information-flow Control Examples

Goal: Verify that route advertisements conform to intended information-flow policy.

- Partial peering
- Controlling prefix propagation
  - Bogons
  - "No Export" prefixes
- Conditional advertisements
- Signalling (e.g., with communities)

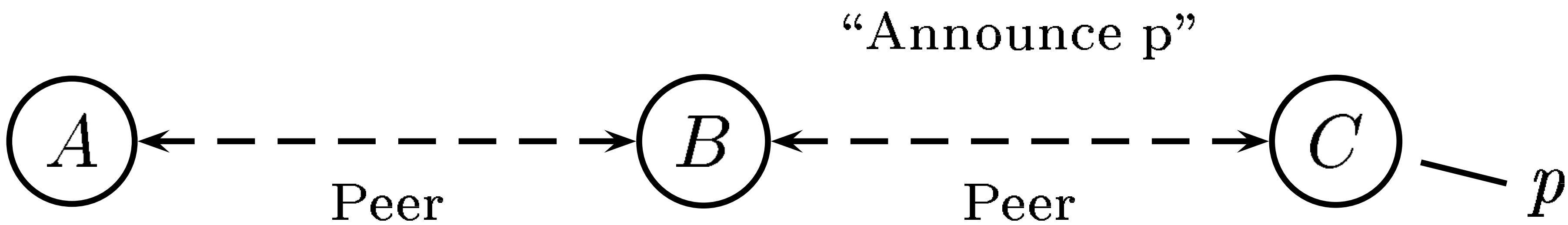
Where are we?



Bad: Import/export route maps, ACLs, communities, etc.

```
neighbor 10.0.0.1 route-map IMPORT-A in
neighbor 10.0.0.1 route-map EXPORT-A out
neighbor 192.168.0.1 route-map IMPORT-C in
neighbor 192.168.0.1 route-map EXPORT-C out
ip community-list 1 permit 0:1000
route-map IMPORT-C permit 10
  set community 0:1000
!
route-map EXPORT-A permit 10
  match community 1
!
```

Where should we be?



Better: Lattice model.

Towards High-level Configuration Languages

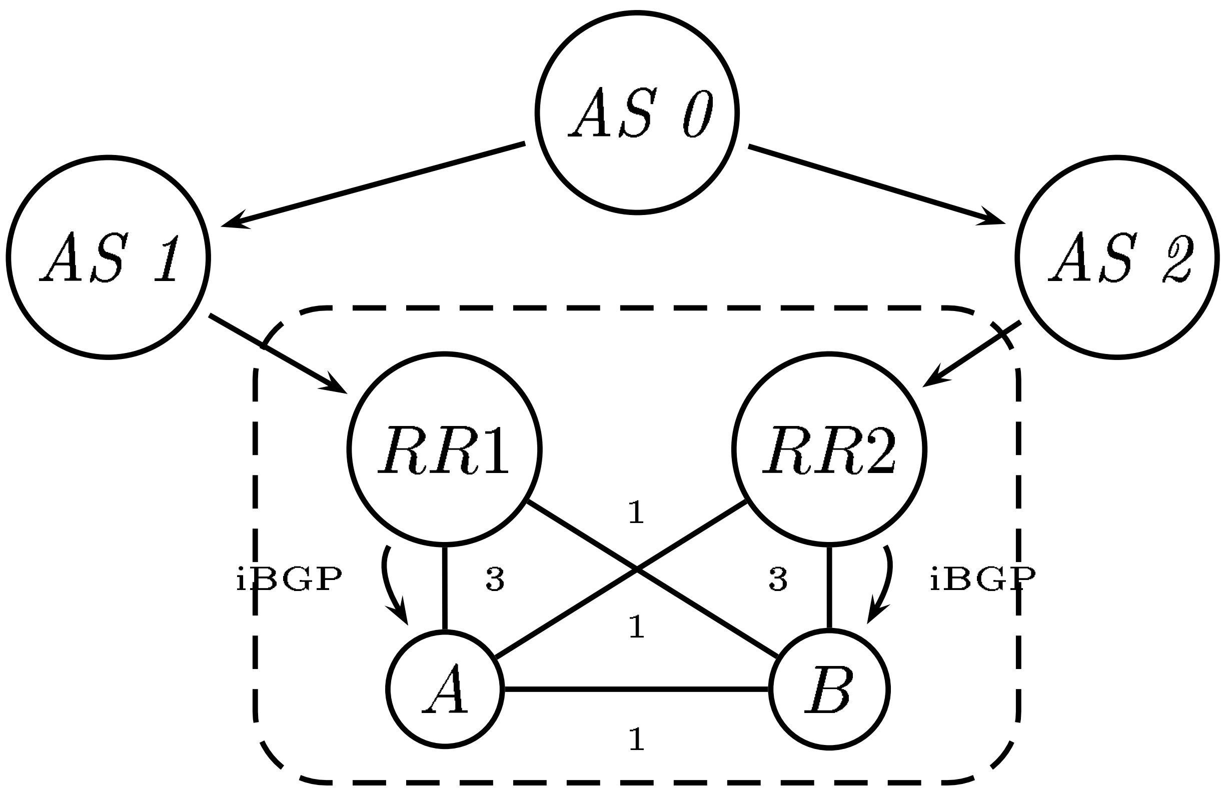
- How to specify the information flow lattice?
  - Must be intuitive.
  - Must express varying levels of detail (i.e., AS-level, session-level, prefix-level, etc.)
  - Must express positive requirements, too.
- Expressing intended behavior will improve routing.
  - **Verification:** check existing configurations against intent.
  - **Synthesis:** generate configurations according to intent.

Beyond Static Rule Checking

- Statistical inference to reduce manual pain. ("Beliefs")
  - 100 routers, 99 have ACLs configured to deny prefix 192.168.0.0/16
  - All eBGP sessions to an AS but one have the same import/export policies.
- Capturing dynamic effects. ("Sandbox")
  - Property violations that appear due to timing, message orderings, failures, etc.
- Avoiding low-level silliness. ("Synthesis")
  - Configuration should be specified at the *intent* level, not at the mechanism level.

Example: Validity

Problem: Persistent forwarding loops due to interactions between iBGP and IGP

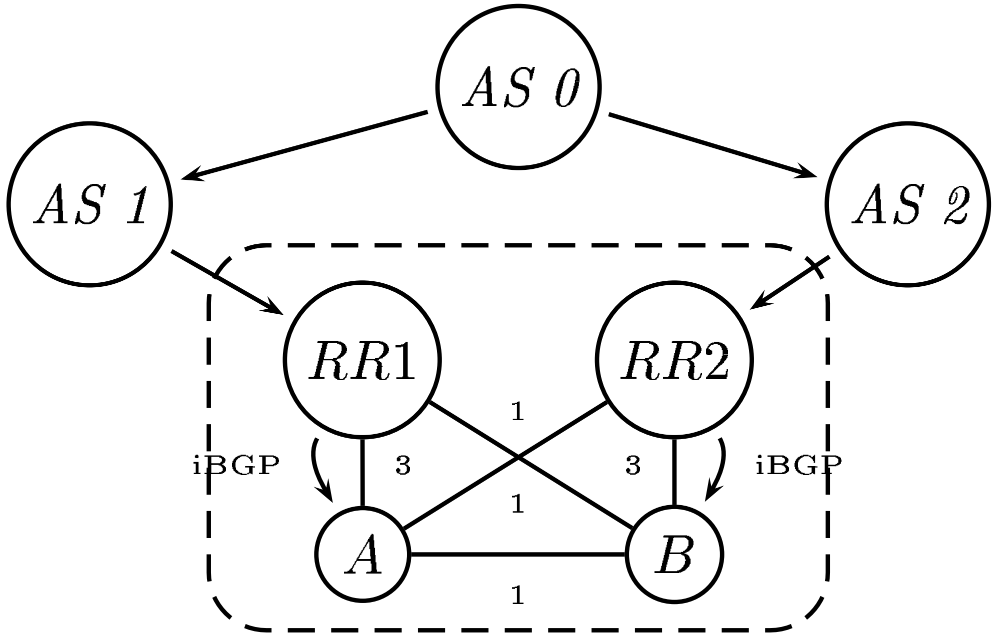


Other Validity Examples

Goal: Verify that advertised routes correspond to valid paths, except where explicitly intended otherwise.

- Accepting/re-advertising bogus or invalid prefixes
- Aggregation
- Next-hop misconfiguration
- eBGP-multihop issues

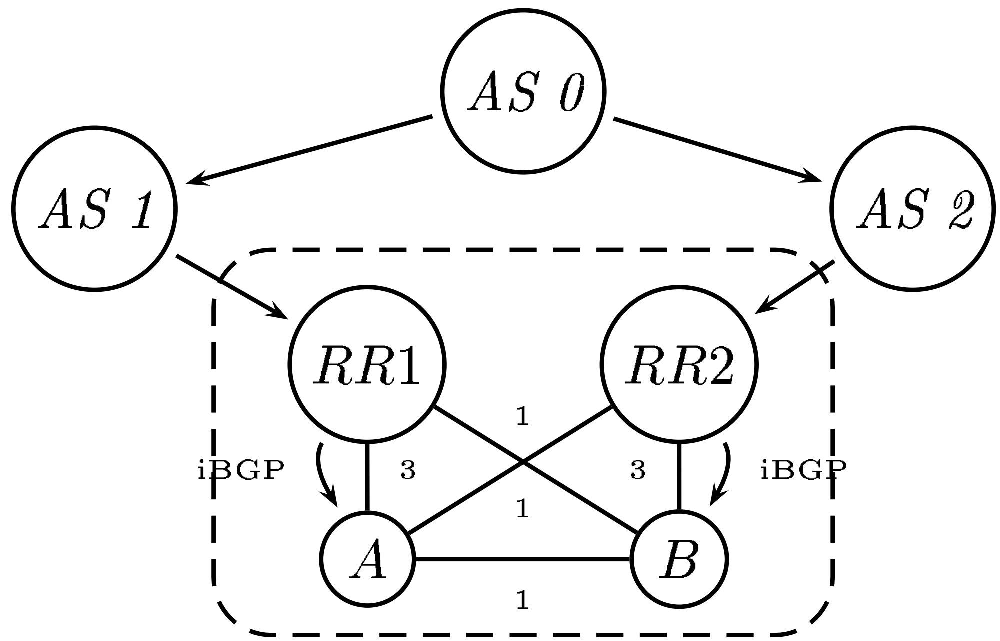
Where are we?



Bad: Ad-hoc heuristics, guidelines for low-level config

```
interface POS1/0
  ip address 10.0.0.1
  ip ospf 10
  ...
!
router bgp 3
  neighbor 10.0.0.2 remote-as 3
  ...
!
```

Where should we be?



Better: Control-flow model.

- Does every IGP hop along the path to the BGP next hop agree on a next-hop?

  (Hamiltonian cycles...)