eecs589

ADVANCED COMPUTER NETWORKS

[GSW99] Griffin, Shepherd, and Wilfong, "Policy Disputes in Path-Vector Protocols," *Proc. of Int'l Conf. on Network Protocols '99,* Nov. 1999

[GR01] Gao and Rexford, "Stable Internet Routing without Global Coordination," *IEEE/ACM Trans. on Networking*, 9(6):681-692, Dec. 2001

[GSW02] Griffin, Shepherd, and Wilfong, "The Stable Paths Problem and Interdomain Routing," *IEEE/ACM Trans. on Networking (TON)*, 10(2): 232-243, Apr. 2002

Internet inter-AS Routing: BGP

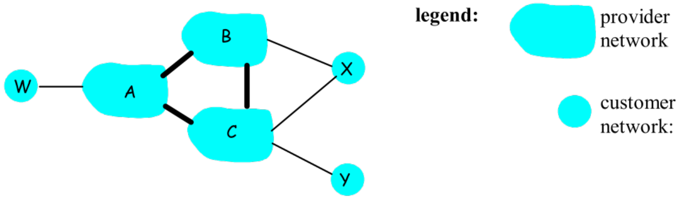
BGP (Border Gateway Protocol), released 07/94, is *de facto* standard for inter-AS routing

BGP provides each AS a means to:

- advertise Address Prefixes (APs) reachability information to neighboring ASs (with eBGP)
- propagate AP-reachability to all AS-internal routers (with iBGP)
- determine "good" routes to APs based on reachability and policy
  - inter-AS routing is policy driven, not load-sensitive, generally not QoS-based

---

BGP Routing Policy Example



A,B,*C* are provider networks

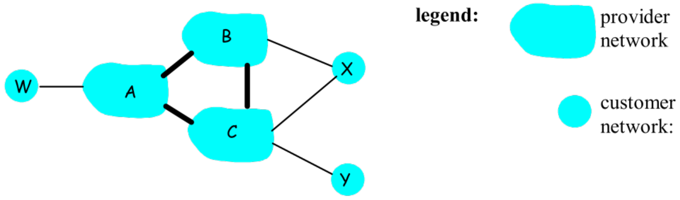
X,W,Y are customers (of provider networks)

X is multi-homed: attached to ≥ 2 networks

X does not want to route from B via X to *C*

.. so X will not advertise to B a route to *C*

BGP Routing Policy Example



A advertises to B the path *A*W

B advertises to X the path B*A*W

B does not advertise to *C* the path B*A*W

- B gets no "revenue" for routing *C*B*A*W since neither W nor *C* are B's customers
- B wants to force *C* to route to W via *A*
- B wants to route only to/from its customers!

Path Attributes & BGP Routes

BGP associates BGP attributes with each AP

Two important attributes:

- `AS_PATH`: the path vector of ASs through which the advertisement for a prefix passed through
- `NEXT_HOP`: the specific router at neighbor AS (there may be multiple exits from current AS to neighbor AS)

Sample BGP routing table entry:

AP	NEXT_HOP	AS_PATH
198.32.163.0/24	202.232.1.8	2497 2914 3582 4600

- address prefix `198.32.163.0/24` is in AS `4600`
- to get there, send to router at address `202.232.1.8`
- the path goes through ASs `2497`, `2914`, `3582`, in order

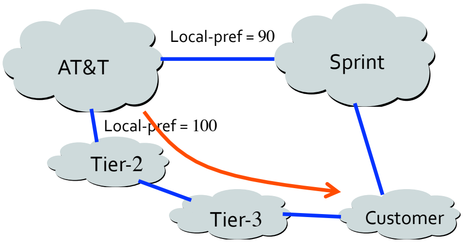
BGP Policy Tools

Export policies: in addition to `AS_PATH`, an AS can set these additional attributes when advertising an AP:

- multiple-exit discriminator (`MED`): an AS can tell a neighbor its preferred ingress point
- community set (`c_set`): an AS can tag certain APs as belonging to the same group, e.g., customer, peer, back-up

Import policies: an AS may learn of more than one routes to some APs

- `local_preference`: an AS can specify its preferred egress point for an AP, e.g., prefer customer over peer



BGP Implicit Policies

Implicit import policies:

- sets `NEXT_HOP` and local preference
- discards some route announcements, to prevent routing loop, configuration mistakes, and attacks
  - discard route if AS already appears in `AS_PATH`
  - discard route if AP advertised by customer is not owned by customer
  - discard customer advertisement that contains other large ISP in its `AS_PATH`

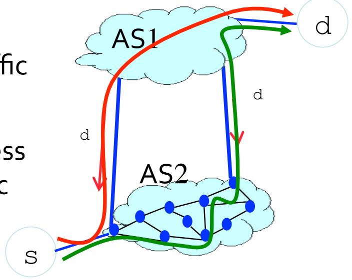
Implicit export policies:

- sets `MED` values
- prepends AS to `AS_PATH`

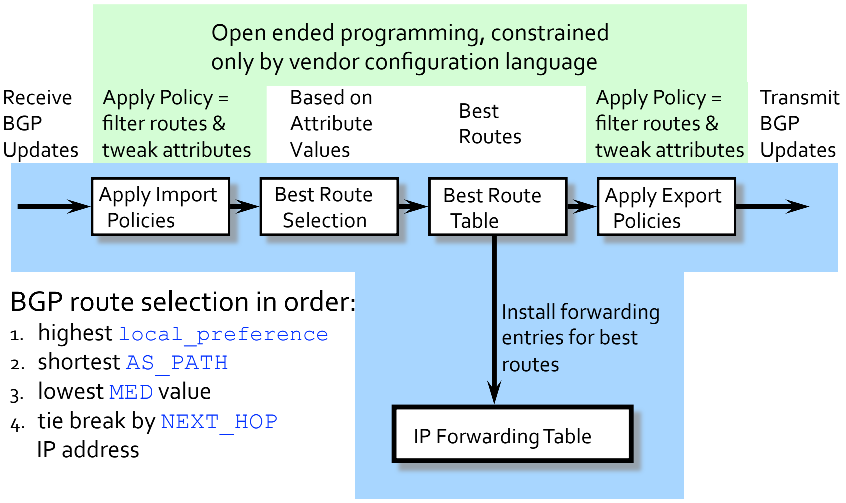
BGP Policy in Play

How an AS sets the attributes of it advertisements influences its neighbors' behavior

- AS prepending: artificially inflate the AS path length (by repeating the AS number in `AS_PATH`) to convince neighbors to use a different AS
- cold-potato routing: AS1 sets `MED` in advertisement for AP `d` to prefer traffic ingress closest to `d`
- hot-potato routing: AS2 prefers egress (`local_preference`) closest to traffic source (ignoring the other AS's `MED`)



BGP Policy: Implementation



BGP route selection in order:

1. highest `local_preference`
2. shortest `AS_PATH`
3. lowest `MED` value
4. tie break by `NEXT_HOP` IP address

[Rexford]

Policy Disputes

BGP allows path choices to be dictated by policy instead of distance metric

Each AS sets its own policy, without any global coordination

Problem: there are unsafe collections of routing policies that can cause BGP to diverge (exchanging BGP routing messages indefinitely)

Griffin, Shepherd, and Wilfong present sufficient conditions on routing policies that guarantee BGP safety [GSW99]

Policy Safety

Steps to ensure a collection of policies is safe:

1. model BGP as Simple Path Vector Protocol (SPVP)
2. check for dispute cycle (or, equivalently, dispute wheel) in an SPVP specification
3. no dispute wheel means an SPVP spec is safe

Simple Path Vector Protocol (SPVP):

- a formal system designed to capture the underlying semantics of BGP
- strips away all but the essentials of BGP, leaving only:
  - permitted paths to a destination
  - the ranking of those paths

SPVP and Solvability

Simple Path Vector Protocol (SPVP) specification:

- each node, representing an AS, has a set of permitted paths to a single destination
- and a ranking *function* that ranks its permitted paths by preference

Solutions to an SPVP specification are routing trees that satisfy certain stability conditions

- static solvability of SPVP is still NP-complete
- but a dynamic evaluation heuristic grows a stable path assignment (a routing tree) in a greedy manner [GSW02]
- the stable states of the dynamic evaluation are solutions to the SPVP specification

Terminology

$G(V, E)$ a network of nodes, $V = \{0, 1, 2, ..., n\}$
Node 0: the origin, a special node that is the destination node to which all other nodes attempt to establish a path

Permitted path (P): a path that has not been filtered out by policy along the way

Ranking function ($\lambda^i(P)$): gives node i's ranking of permitted path P by policy preference; larger $\lambda()$ means higher preference

Terminology and Notation

A path in G, $P = v_k, v_{k-1}, ..., v_1, v_0$, s.t. $\forall\, i > 1, \{v_i, v_{i-1}\} \in E$

PQ: concatenation of P and Q, the last node in P must be the same as the first node in Q

$(u, v_k)P = u, v_k, v_{k-1}, ..., v_0$ (v_k must be the first node in P)

$\varepsilon P = P\varepsilon = P$, ε empty path

$P[v_i, v_j]$: subpath $v_i, v_{i-1}, ..., v_j$ of simple path P

$\mathcal{P}^v$: set of permitted paths from v to the origin

For P_1, $P_2 \in \mathcal{P}^v$, and $\lambda^v(P_1) < \lambda^v(P_2)$, then P_2 is said to be preferred over P_1 (larger $\lambda()$, higher preference)

SPVP

For $\mathcal{P} = \cup_{v \in V} \mathcal{P}^v$, the set of all permitted paths to the origin, and $\Lambda = \{\lambda^v | v \in V - \{0\}\}$, the set of all ranking functions, an SPVP specification is $S = (G, \mathcal{P}, \Lambda)$

Restrictions on Λ and $\mathcal{P}$:

- for each $v \in V$, $\varepsilon \in \mathcal{P}^v$ (it's ok not to have a path)
- for each $v \in V$, $\lambda^v(\varepsilon) = 0$
- If $\lambda^v(P_1) = \lambda^v(P_2)$, then $P_1 = P_2$ or $P_1 = (v, u)P'_1$ and $P_2 = (v, u)P'_2$ (P_1 and P_2 have the same next hop)
- if path $P \in \mathcal{P}^v$, P is a simple path (no repeated nodes)
- if path $P \in \mathcal{P}^v$, and node $w \neq 0$ is in P, then $P[w, 0] \in \mathcal{P}^w$ (consistency: tail of a permitted path must be a permitted path)

Stability and Solvability

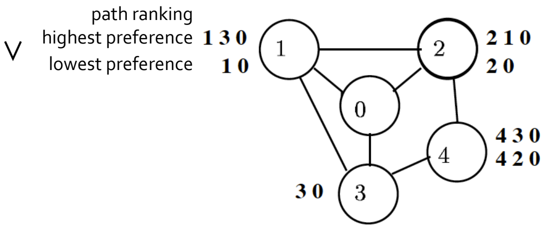
A routing tree $T = (P_1, P_2, ..., P_n)$ is a vector of paths with $P_i \in \mathcal{P}^i$ s.t. the union of these paths is a tree

Node i is stable with respect to T if $\lambda^i((i, j)P_j) \leq \lambda^i(P_i)$ whenever $(i, j)P_j \in \mathcal{P}^i$, i.e., an alternate permitted path is not preferred over current path

T is stable if every node is stable

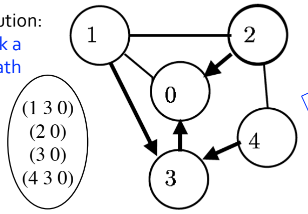
S is solvable if $\exists$ a stable $T \Rightarrow T$ is a solution to S

Example 1: GOOD GADGET



SPVP specification: **GOOD GADGET**

a routing tree/solution:
no node could pick a more preferred path



GOOD GADGET
is safe

Solution to specification: **A routing tree**

Dynamic Evaluation

Now consider collection of permitted paths at all nodes at any one time as a state

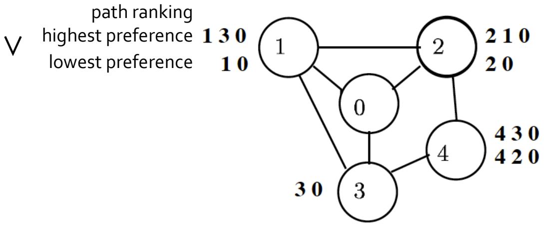
(1 3 0)
(2 0)
(3 0)
(4 3 0)

A state for SPVP S is a vector $s = (P_1, P_2, ..., P_n)$, where $P_i \in \mathcal{P}^i$
- s is not always a tree (could be cyclic)

In dynamic evaluation, $\mathrm{Eval}(S)$, the SPVP moves from one state to another where each "activated" node (a node that must recompute path):
- processes all neighbors' updates
- computes any changes to preferred routes
- and sends updates to its neighbors

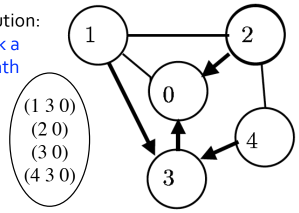
Example 1

GOOD GADGET
is safe



SPVP specification: **GOOD GADGET**

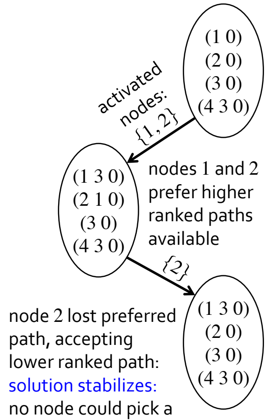
a routing tree/solution:
no node could pick a more preferred path



A routing tree

State transition diagram
or evaluation digraph,
$\mathrm{Eval}(S)$

unstable
initial state



nodes 1 and 2 prefer higher ranked paths available

node 2 lost preferred path, accepting lower ranked path:
solution stabilizes:
no node could pick a more preferred path

stable
state

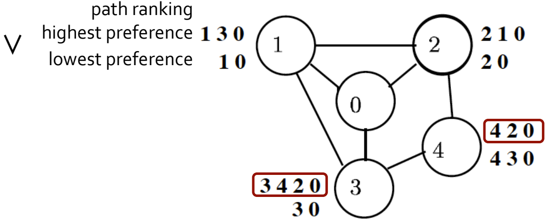
Dispute Cycle

Captures a certain type of circular policy inconsistency

An SPVP specification with no dispute cycle always has a unique solution and is safe
- its dynamic evaluation will always arrive at a stable state

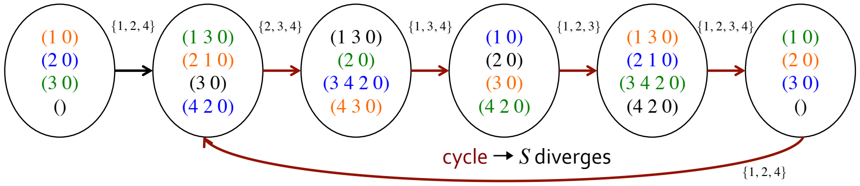
Example 2

BAD GADGET
is not solvable



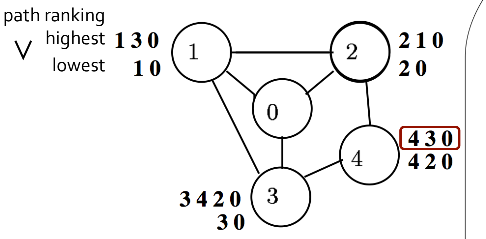
SPVP specification: **BAD GADGET**

Has no solution, dynamic evaluation diverges:



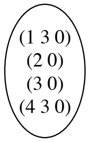
cycle → S diverges

Example 3



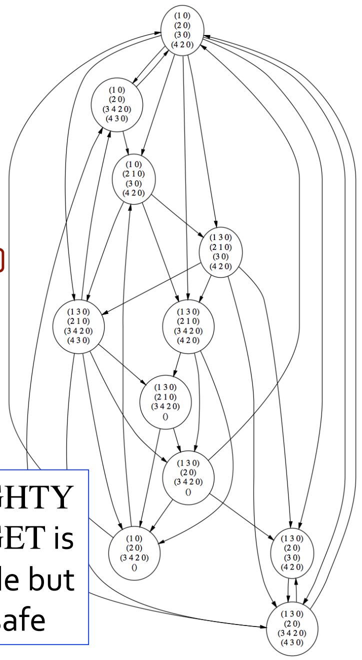
SPVP specification: **NAUGHTY GADGET**

Same unique solution as GOOD GADGET:



But dynamic evaluation diverges:

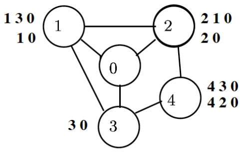
NAUGHTY GADGET is solvable but not safe



Dynamic Evaluation: Formally

Let:

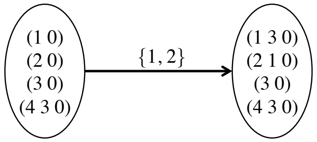
- $A \subseteq V \neq \varnothing$ be the set of nodes that must update paths (activated nodes),
- $s = (P_1, \ldots, P_n)$ be the SPVP state before the updates, and
- $s' = (P'_1, \ldots, P'_n)$ be the SPVP state after nodes in A update their paths



GOOD GADGET

$$P'_i = \begin{cases} P_i \text{ if } i \notin A \text{ (} i\text{'s path doesn't change),} \\ P \in \mathcal{P}^i \text{ s.t. } \lambda^i(P) \text{ is maximal} \end{cases}$$

$s \xrightarrow{A} s'$ denotes this transition



Stable State

A state s is stable if $s \xrightarrow{A} s$ for every A, i.e., no node could pick a better path than its current path

An update sequence σ is a function s.t. $\sigma(t) \subseteq V$, for each $t \geq 0$, i.e., $\sigma(1) = A_1, \sigma(2) = A_2, \ldots, \sigma(t) = A_t$

$\sigma(s_0, t) = s_t$: $s_0 \xrightarrow{A_1} s_1 \xrightarrow{A_2} s_2 \xrightarrow{A_3} \ldots \xrightarrow{A_t} s_t$

Convergence and Safety

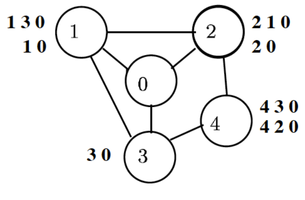
S is said to converge with respect to σ and s_0 if $\exists\, t$ s.t. $\sigma(s_0, t)$ is stable

Otherwise it is said to diverge with respect to σ and s_0

σ is fair if for each node u, $u \in \sigma(t)$ for infinitely many t's (σ makes progress)

S is safe if it converges for every fair σ and every initial state s_0

Dispute Digraph

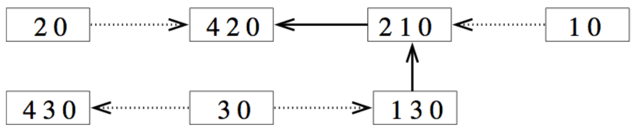


GOOD GADGET

A dispute digraph of S ($\mathcal{DD}(S)$) consists of nodes and arcs where:

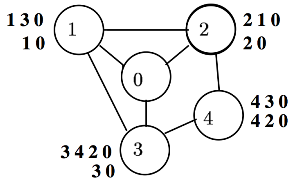
- each node represents a permitted path
- an arc is either a transmission arc or a dispute arc
- transmission arc (-->) : a permitted path at one node allowing another permitted path at another node
- dispute arc ($\longrightarrow$): policy dispute between nodes that disallow a permitted path at one of the nodes

$\mathcal{DD}$(GOOD GADGET):

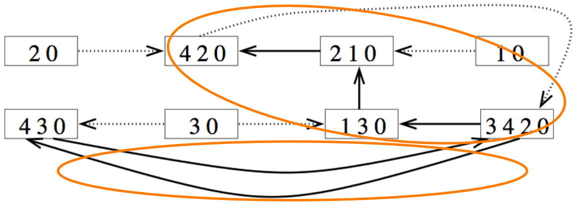


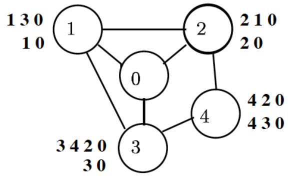
Dispute Cycle

A cycle in the dispute digraph

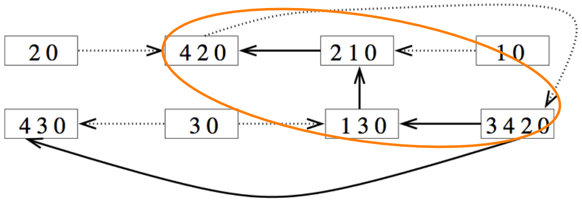


NAUGHTY GADGET





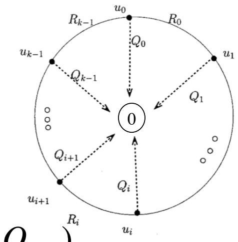
BAD GADGET



Dispute Wheel

Generalization ("long-distance") and formalization of dispute cycle used to prove solvability and safety of SPVP specification

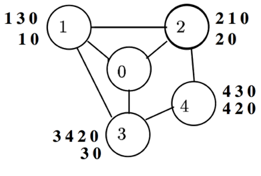
Dispute wheel constructed from a set of nodes where each node u_k has two permitted paths Q_k and $R_k\, Q_{k+1}$ where the path through the neighbor is preferred over the other $\lambda^{u_k}(Q_k) \leq \lambda^{u_k}(R_k\, Q_{k+1})$



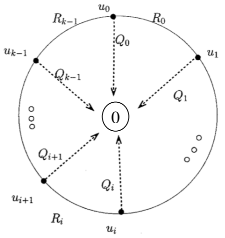
- neighbor in dispute wheel is not necessarily neighbor in actual network, i.e., the path R can have length > 1

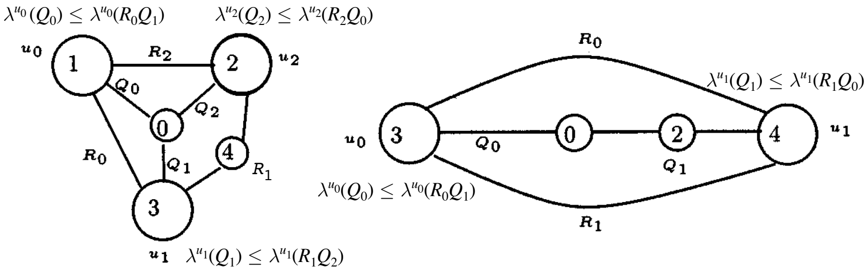
(Non-)existence of dispute wheel is then used to prove solvability and safety of SPVP specification

Example



NAUGHTY GADGET





A dispute wheel of both BAD and NAUGHTY GADGETs

Another dispute wheel of NAUGHTY GADGET

[GSW02]

Theorems

A specification S has a dispute wheel iff $\mathcal{DD}(S)$ contains a cycle

If S has no dispute wheel, S is solvable, i.e., $\exists$ a stable routing tree for S

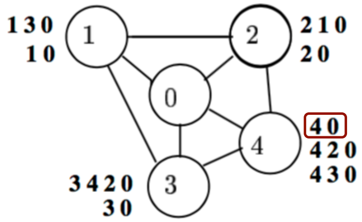
Divergence implies a dispute wheel: if $\exists$ a non-trivial cycle (contains no self-loops) in the evaluation digraph of S, $\mathrm{Eval}(S)$, S contains a dispute wheel

Theorems

Sufficient condition: if S has no dispute wheel, $\mathrm{Eval}(S)$ has no non-trivial cycles, and S is safe

$\neg$(Necessary condition): if S has a dispute wheel, $\mathrm{Eval}(S)$ may or may not contain a cycle

Example: BAD BACKUP has a dispute wheel that is not realizable in the evaluation and is safe



BAD BACKUP

Summary

Authors present sufficient conditions on routing policies that guarantee BGP safety

Dispute cycle captures a circular set of relationships between ranking functions

An SPVP specification with no dispute cycle always has a unique solution and safe

- specification with no dispute cycle is safe
- its dynamic evaluation will always arrive at a stable state (solution to the SPVP specification)

Implication of SPVP

Conjecture: only SPF route selection is provably safe

SPVP: if S is consistent with a coherent cost function, such as SPF, then S has no dispute wheel ⇒ S is safe

However, S being safe doesn't require consistency with a coherent cost function ⇒ route selection can "violate" distance metric and remain safe!

Application of SPVP

Static evaluation of BGP is NP-hard, even of SPVP is NP-complete [GSW99]

How do we ensure BGP convergence?

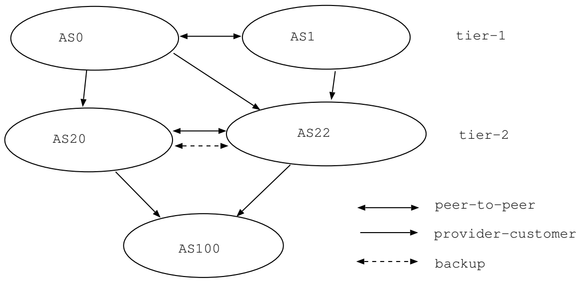
Gao and Rexford propose a set of policy guidelines that

- imposes a partial order on the set of routes to each destination
- does not require global coordination
- exploits the hierarchical structure of the Internet and the commercial relationships between ASs
- conveniently already conforms to common practices
  ⇒ why we haven't seen BGP divergence on the Internet [GR01]

AS Relationships

Commercial relationships between ASs:

- peering: peers agree to exchange traffic for free (settlement free), usually when traffic exchange is balanced (not more than 1:3 ratio)
- customer-provider: customer pays for access
- backup

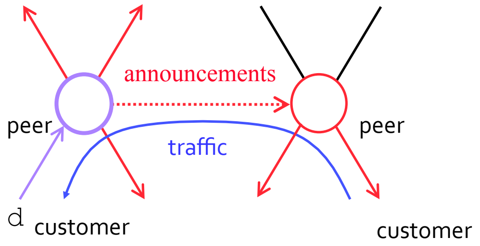


Peering Relationship

Peers exchange traffic of their customers

- AS exports only its customers' APs to a peer
- AS exports a peer's APs only to its customers
- Peers don't advertise APs learned from other peers or providers (no transit)

Traffic to/from the peer and its customers



[Rexford]

Customer-Provider Relationship

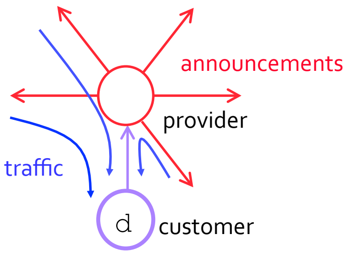
Customer needs to be reachable by everyone

- provider tells all its neighbors how to reach the customer
- prefer-customer over peer in case of multi-homed customer

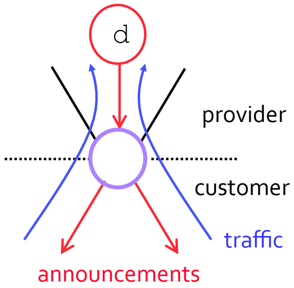
Customer needs to reach everyone

- provider advertises all APs to customer

Traffic to the customer



Traffic from the customer

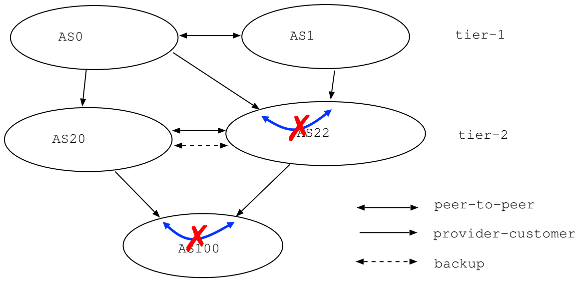


[Rexford]

Valley-Free Routing

Customer does not want to provide transit service

- customer only advertises its own Aps
- not APs from peers nor other providers (in case of multi-homing)



Policy Guidelines

Guideline A: Prefer-Customer

- prefer routing via customer over routing via peer or provider
- results in stable path; prove by induction:
  - Phase 1: activate ASs in customer-to-provider DAG in linear order
  - Phase 1 is stable:
    - customer itself is stable
    - assume stable after k hops to provider
    - $k+1$ hop is stable because its options are stable
  - Phase 2: activate provider-to-customer DAG in linear order
  - Phase 2 is stable:
    - first AS (provider) is stable
    - assume stable after l hops from provider
    - $k+1$ hop is stable because its options are stable

Policy Guidelines

Guideline B:

- allow routing via customer or peer with equal preference, but over routing via provider
- results in stable path if after clustering peers into clusters, the clusters form a DAG
  - prove by induction in two phases similar to Guideline A, but additionally assume activation in linear order of the cluster DAG
  - and note that an AS always prefers a customer route with a shorter AS path to a peer route, ensuring preference for the customer-provider DAG with shorter route

Policy Guidelines

Guideline C:

- use backup link only if there's no customer, peer, or provider link
- requires coordination between ASs
  - to mark backup path using community set
- set all backup paths the same `local_preference` value
- to ensure safety, activate backup paths in shortest path first order

Policy Guidelines

ASs can have different relationship for different APs, guidelines apply per destination AP

During relationship change (customer to peer or customer to provider—unlikely), modify provider's policy configuration first

BGP Routing Policy Loop

Current approach to prevent BGP policy loops:

- ISPs register their policy with Internet Routing Registry (IRR)
- Policy specified in a standard language
- Conflicts can be checked

Problems:

- Policies/relationships must be revealed and updated
- Static checking for convergence is NP-hard
- BGP may not converge under router/link failure or policy changes