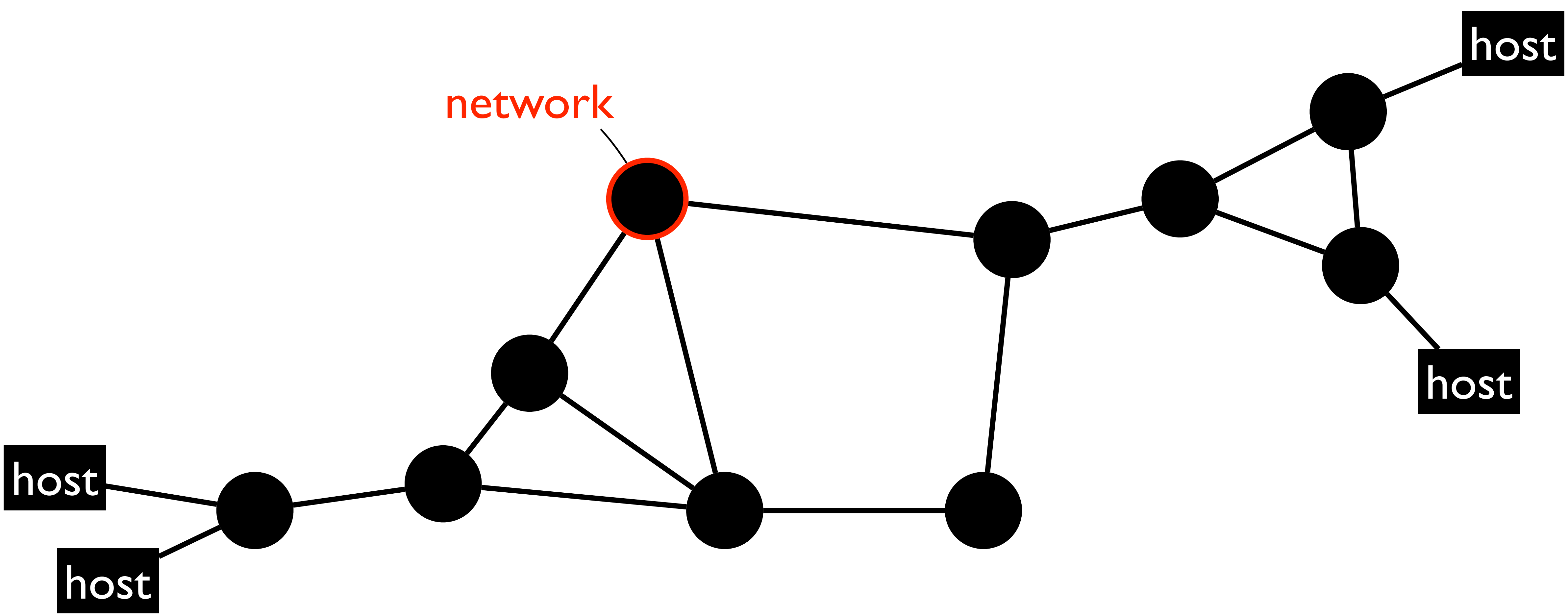
Internet Routing and Non-monotonic Reasoning

Anduo Wang and Zhijia Chen
Temple University

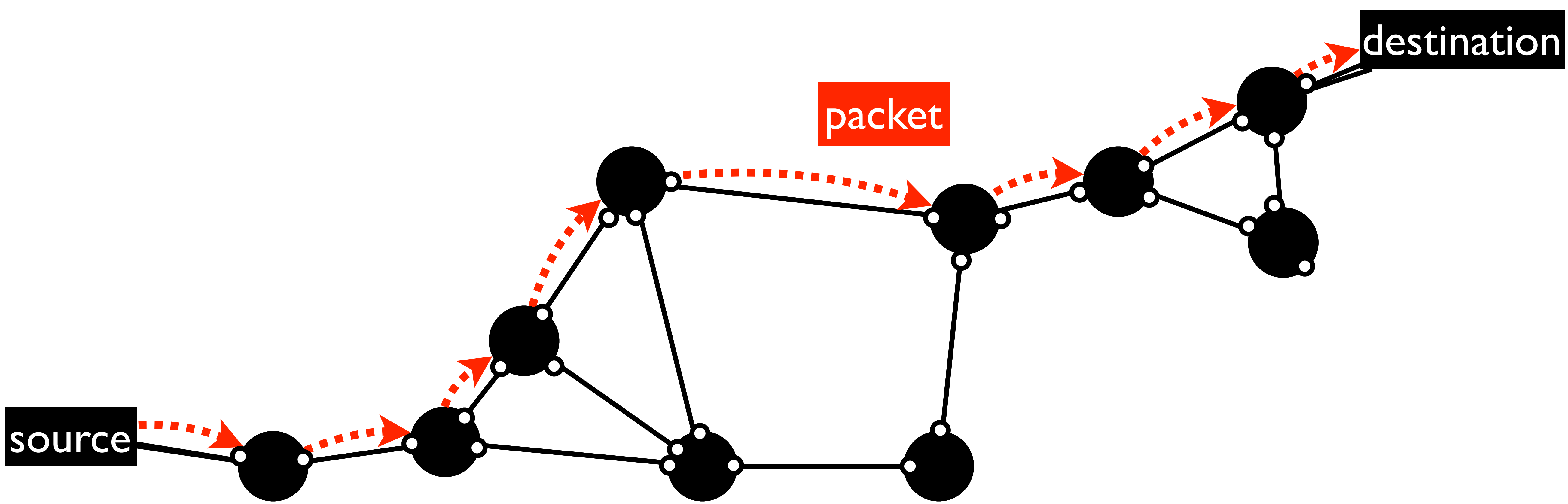
the Internet

a loose federation of networks, also called autonomous systems (ASes)



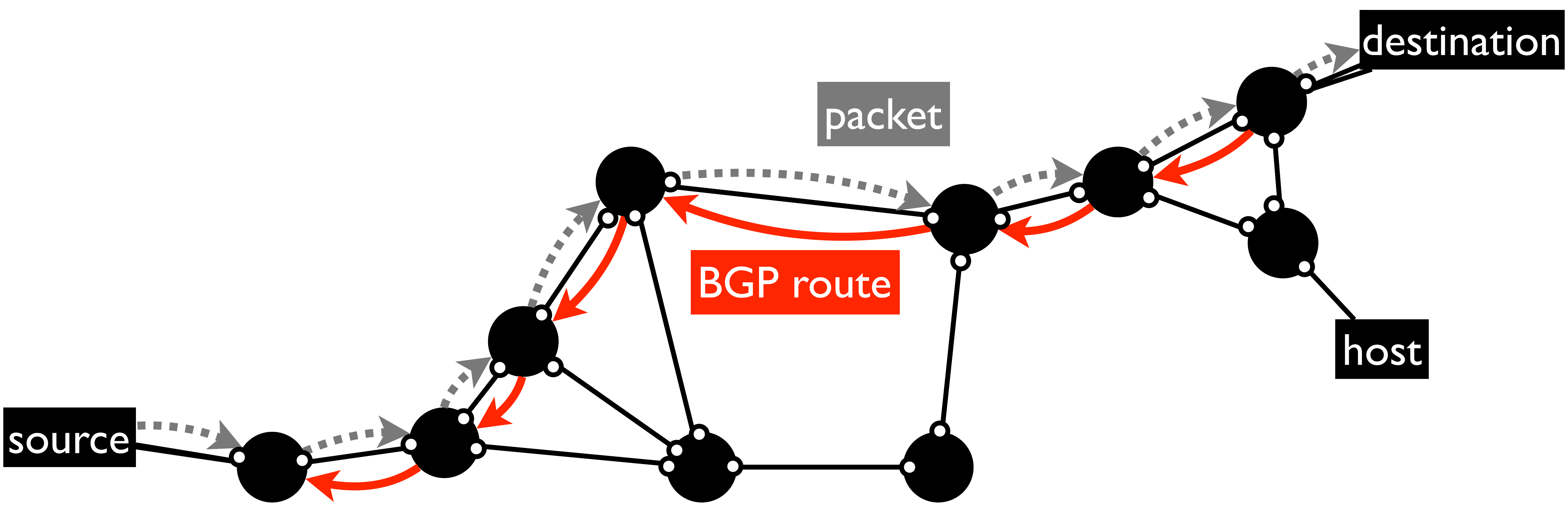
Internet routing

determine a sequence of networks — path — a data packet will traverse in passing from the source to the destination



Internet routing

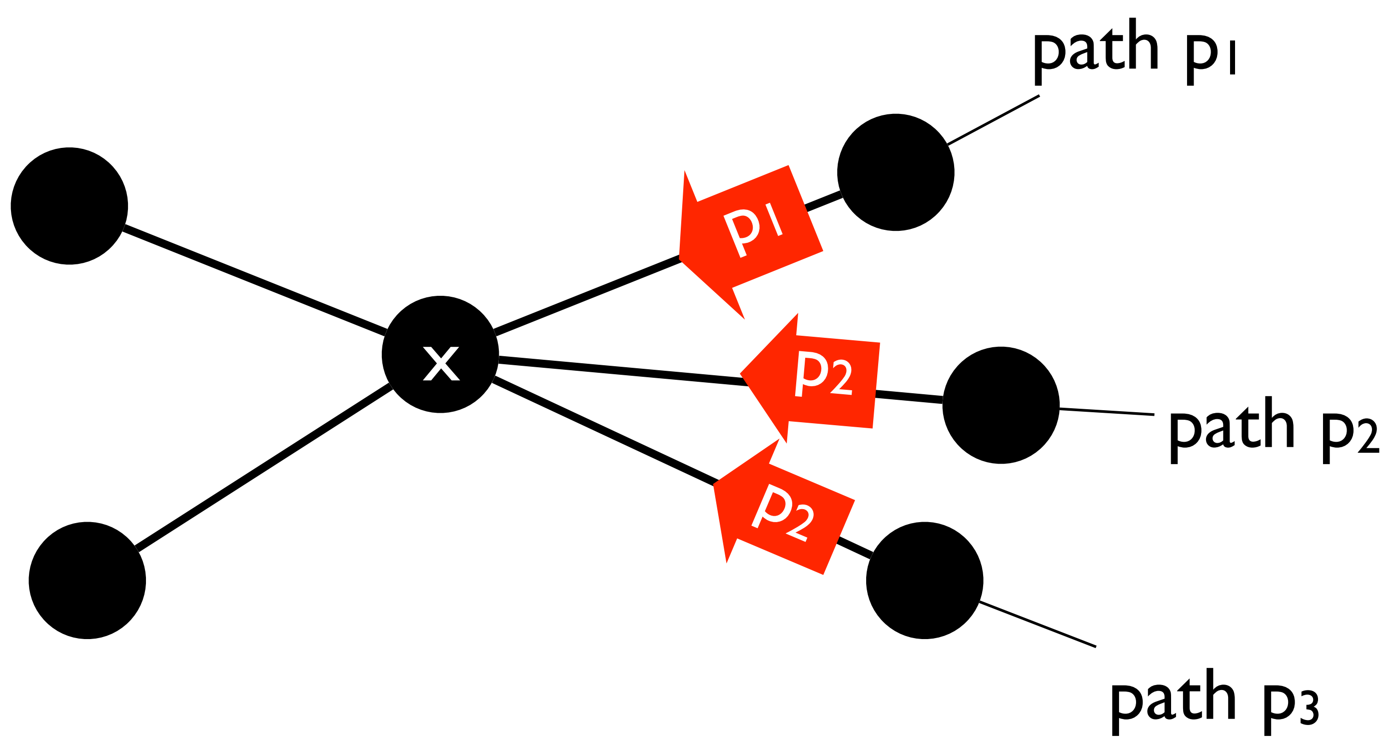
routing path established by distributed algorithm — border gateway protocol (BGP)



distributed route computation

routing path established by distributed algorithm — border gateway protocol (BGP)

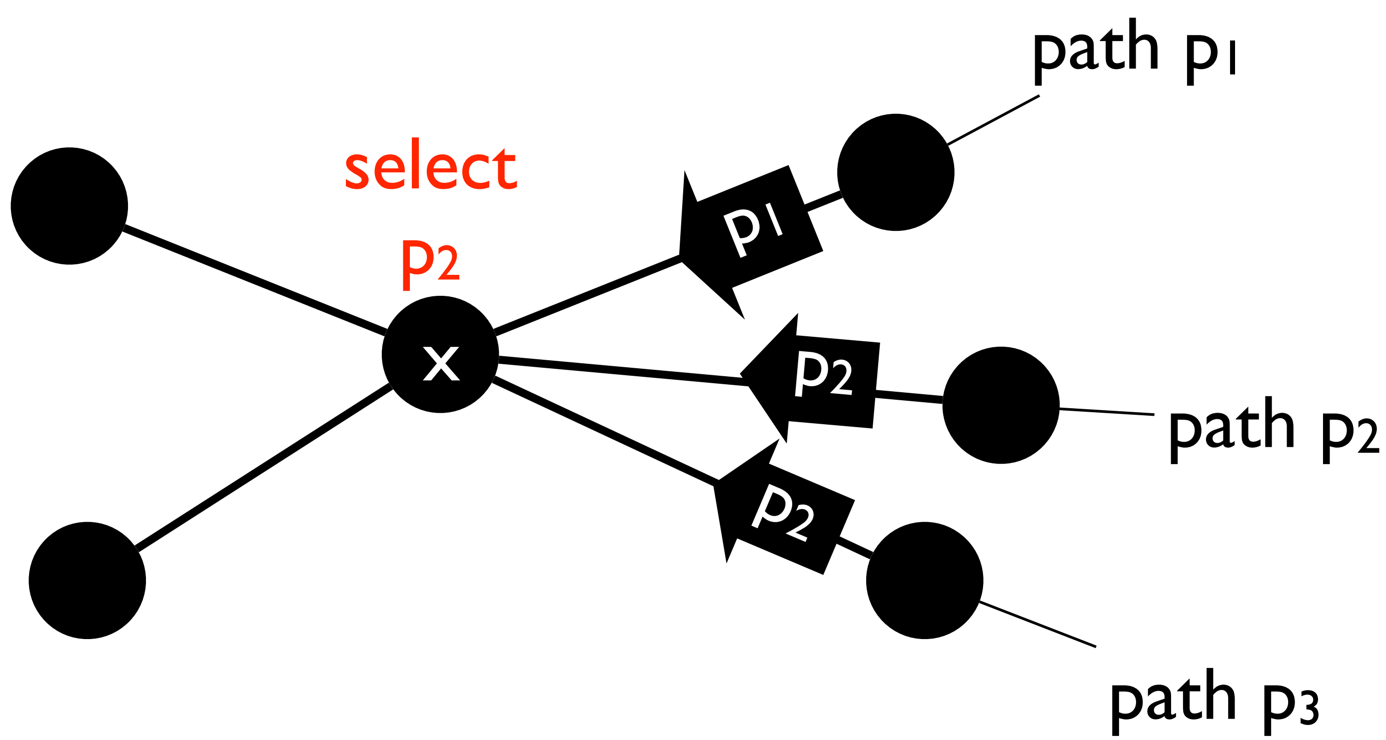
step 1. AS x learns alternative paths (BGP routes) to reach a destination through its neighbors (closer to the destination)



distributed route computation

routing path established by distributed algorithm — border gateway protocol (BGP)

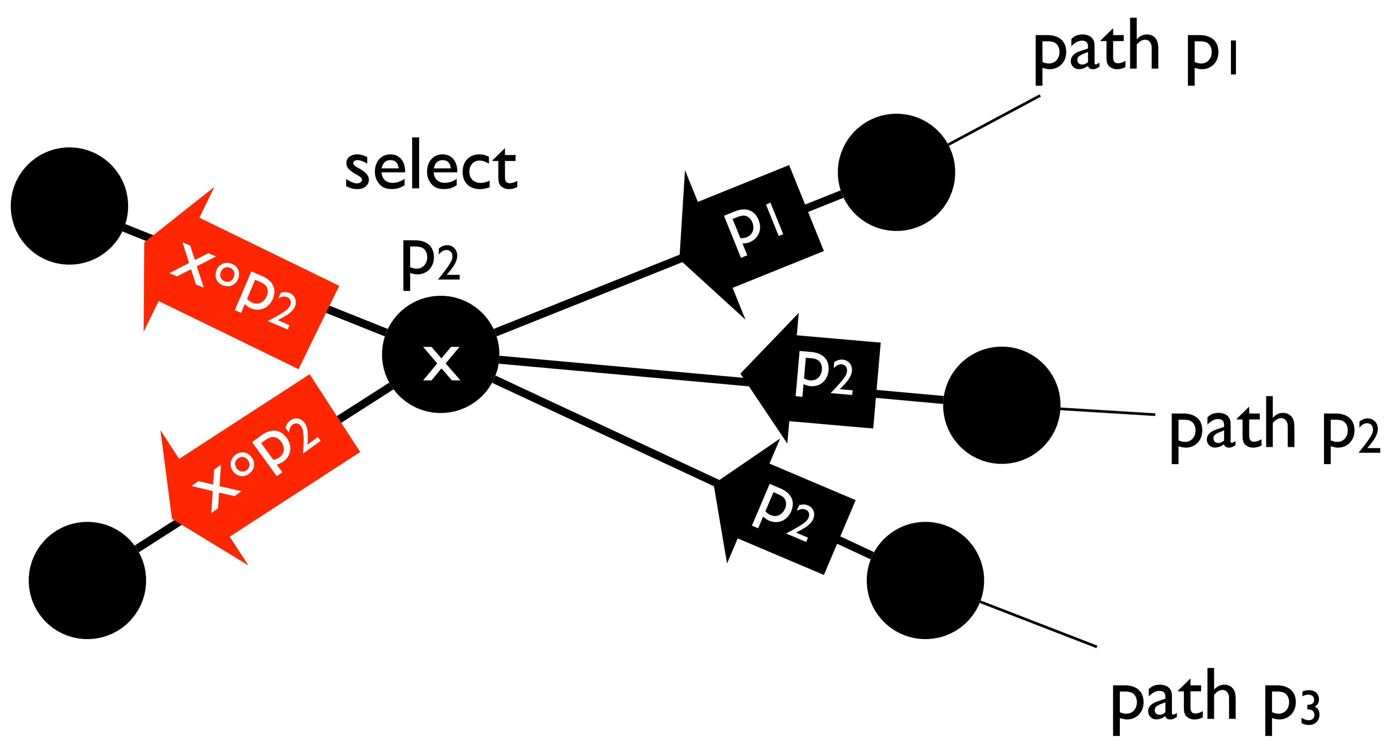
step 2. AS x selects one single best path out of its available paths based on some measure of the paths



distributed route computation

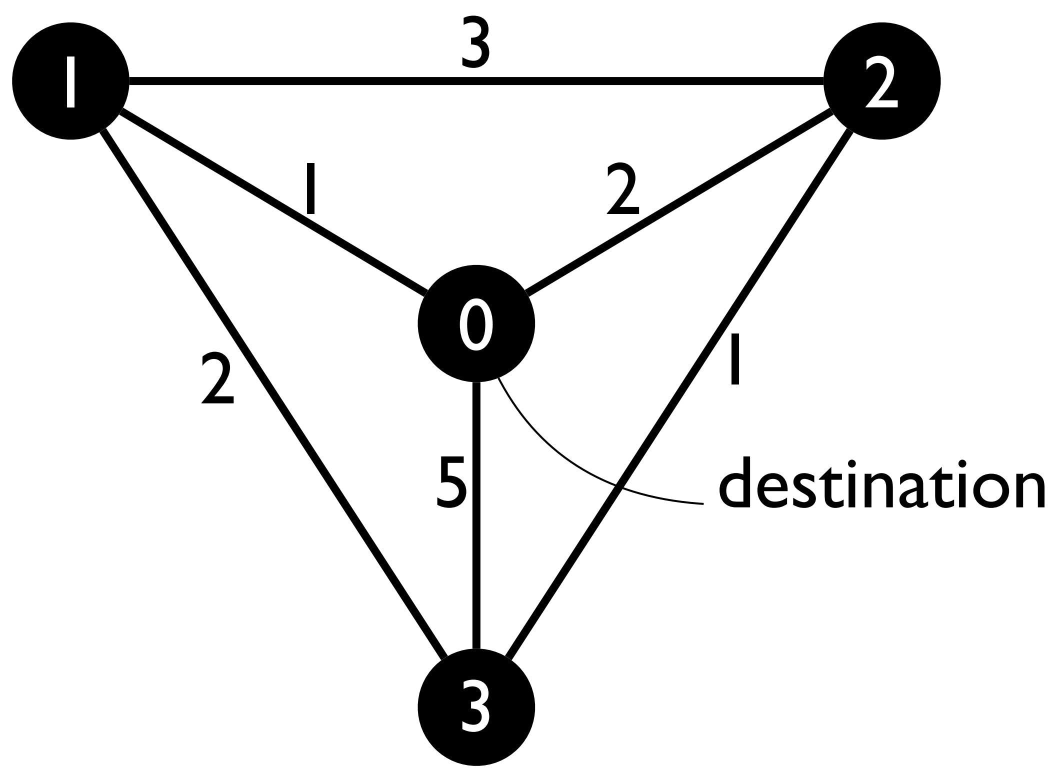
routing path established by distributed algorithm — border gateway protocol (BGP)

step 3. AS x re-advertises its path (by appending itself) to other neighbors (farther away from the destination)



original Internet always selects the shortest paths — measure the paths by distance

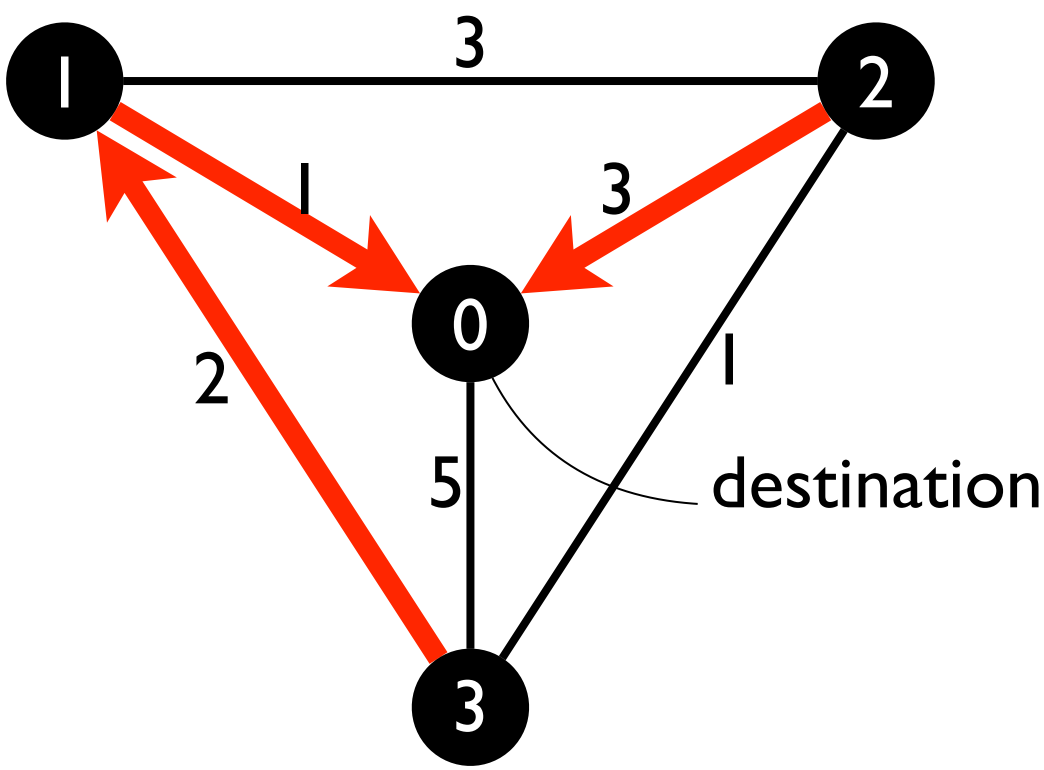
shortest path routing always *converge*



all ASes guaranteed to agree upon some global policy-compliant path(s)

- most preferred possible for all ASes along the path

shortest path routing always *converge*

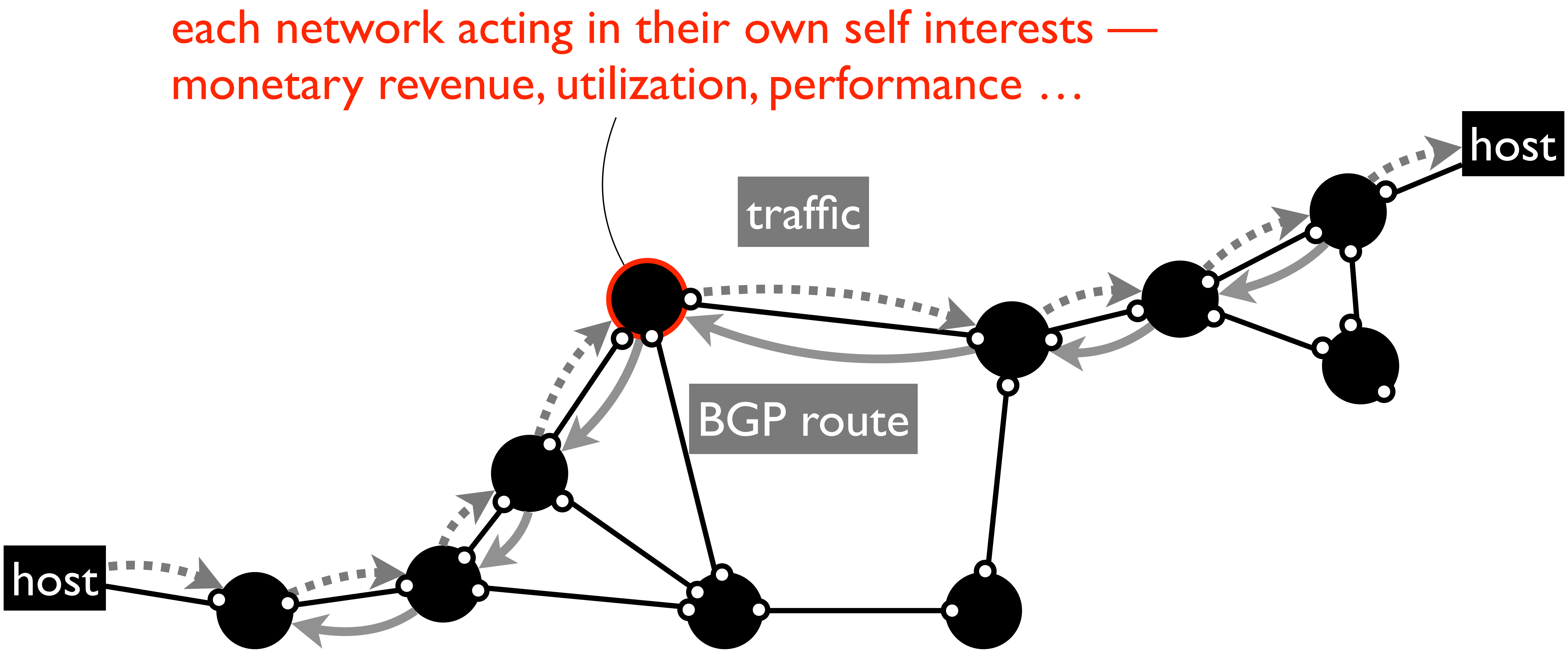


guaranteed to agree upon some global policy-compliant path(s) without coordination

- most preferred possible for all ASes along the path
- form a routing tree

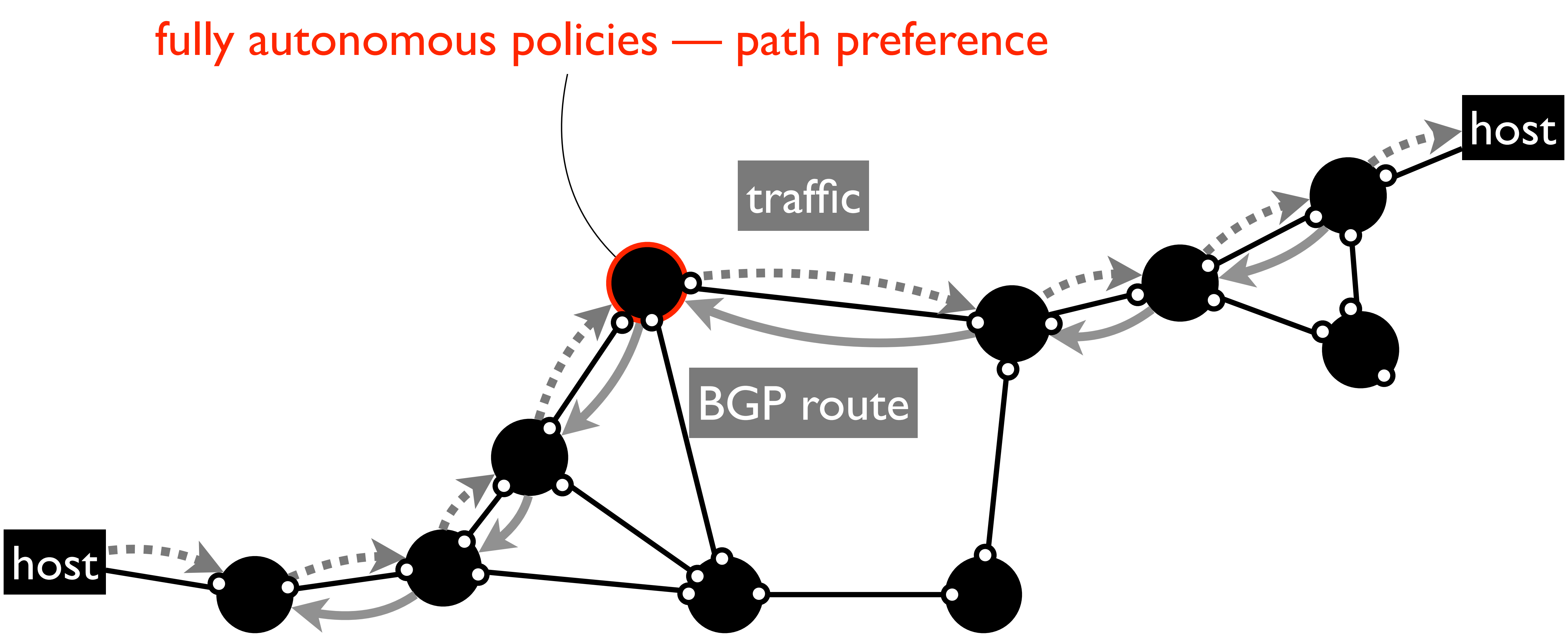
but, Internet routing

far beyond shortest paths computation on a graph

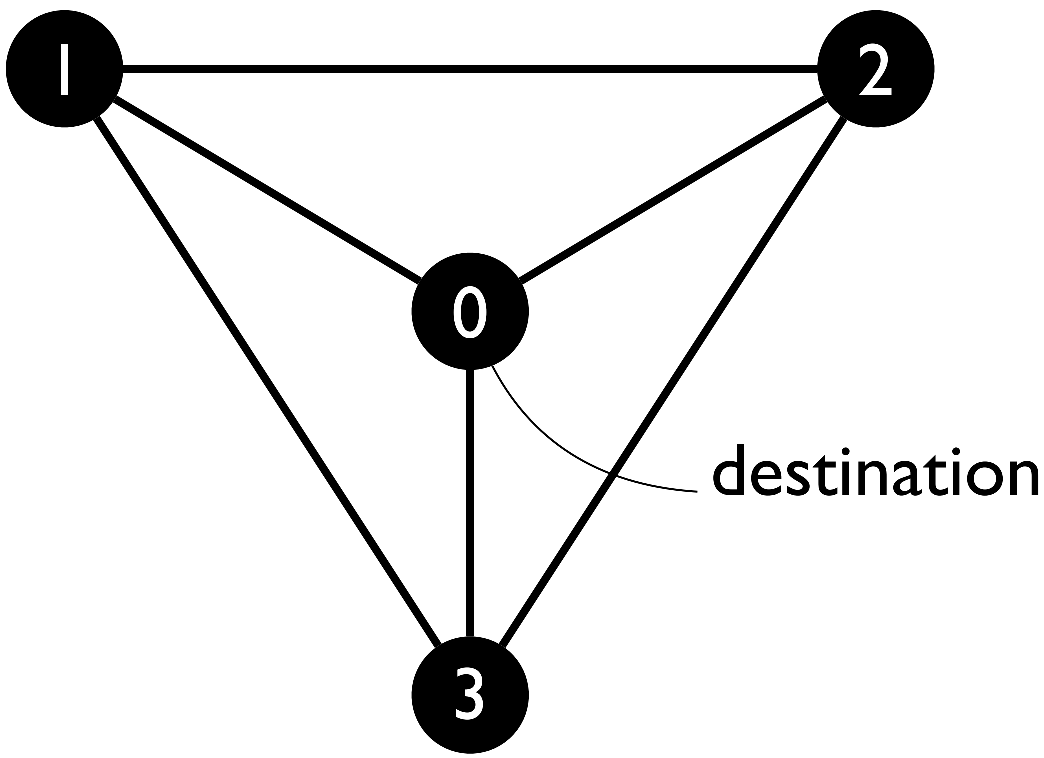


Internet routing

maintaining AS-level path, *modulated by AS policies*

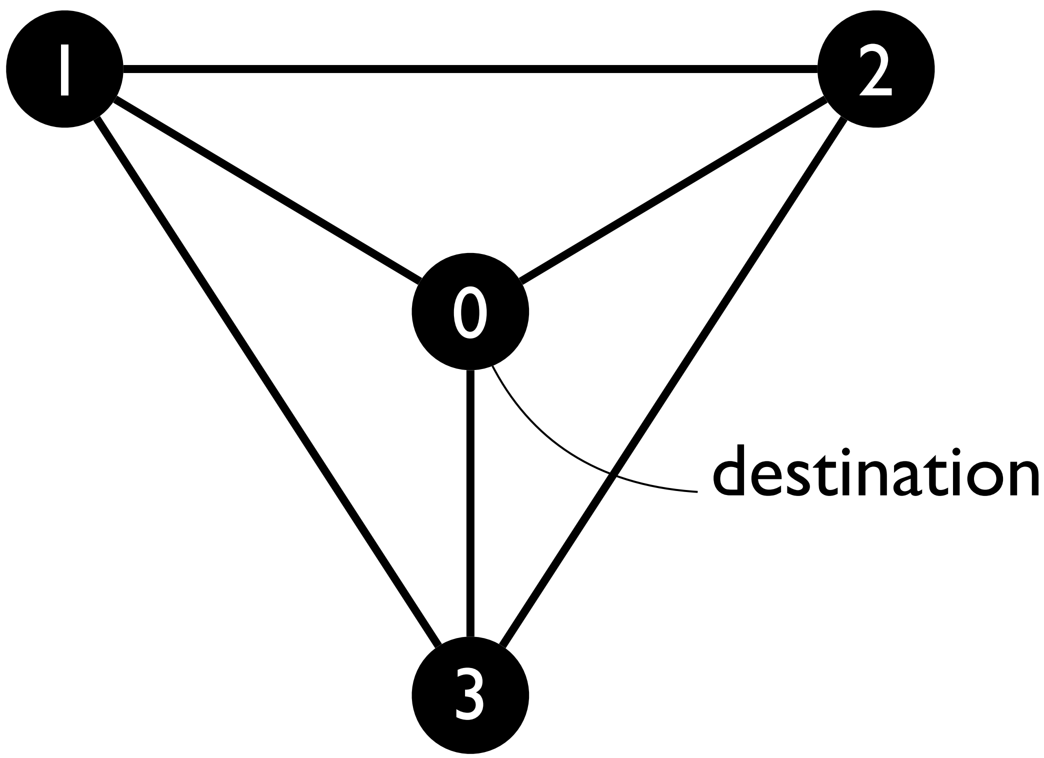


autonomous policy routing can diverge



each ASes — 1,2,3 — prefer their clock-wise neighbor

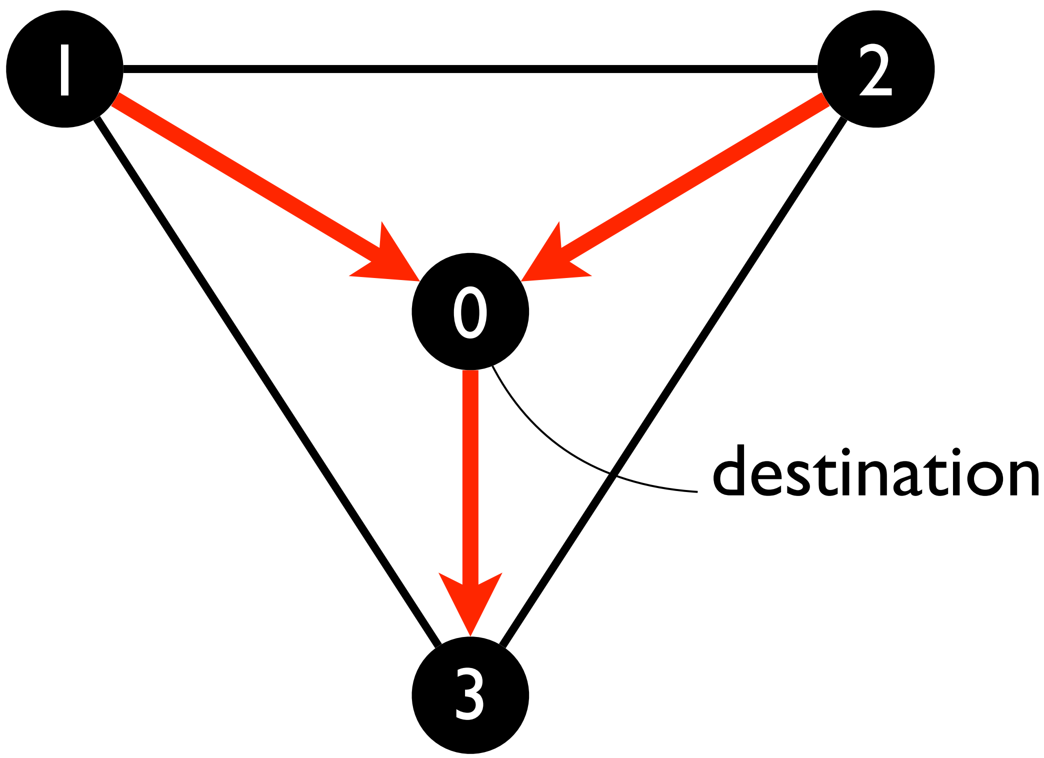
autonomous policy routing can diverge



each ASes — 1,2,3 — prefer their clock-wise neighbor

- cannot be simultaneously satisfied

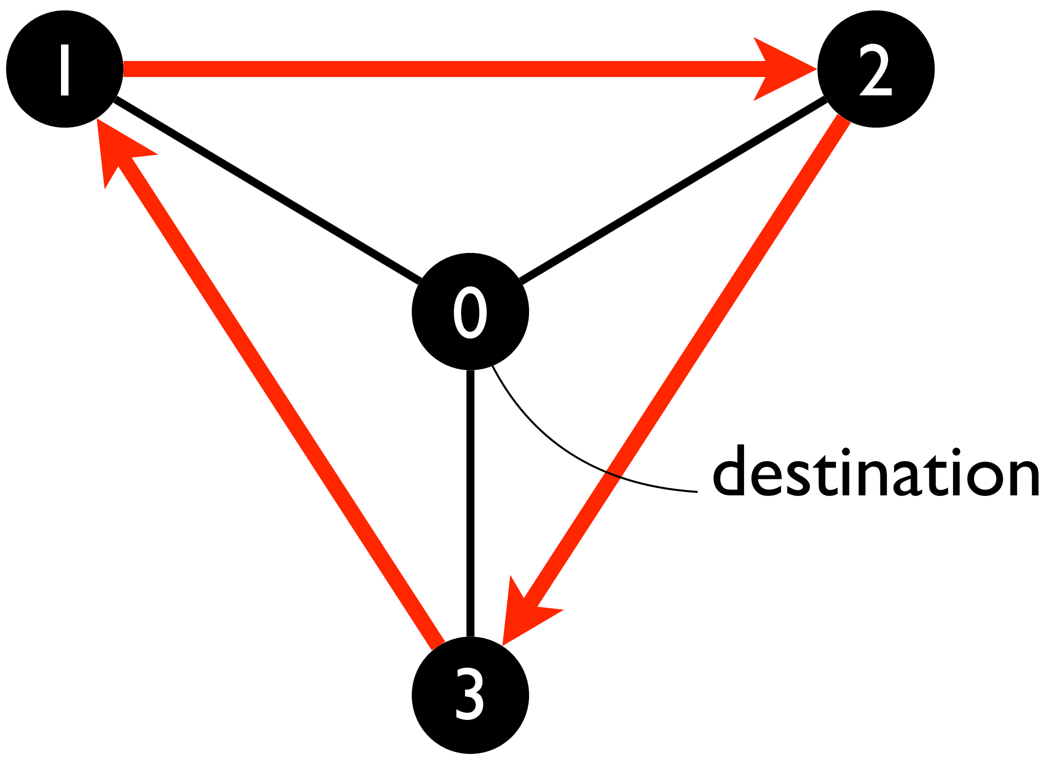
autonomous policy routing can diverge



each ASes — 1,2,3 — prefer their clock-wise neighbor

- cannot be simultaneously satisfied
- lead to permanent oscillation

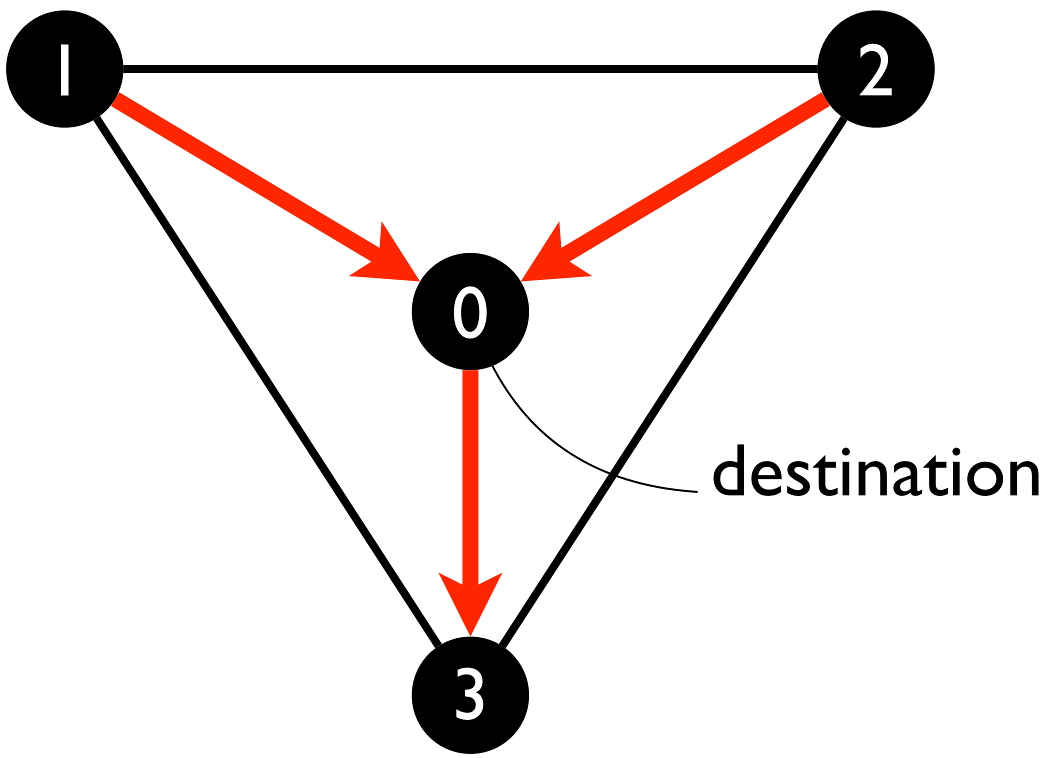
autonomous policy routing can diverge



each ASes — 1,2,3 — prefer their clock-wise neighbor

- cannot be simultaneously satisfied
- lead to permanent oscillation

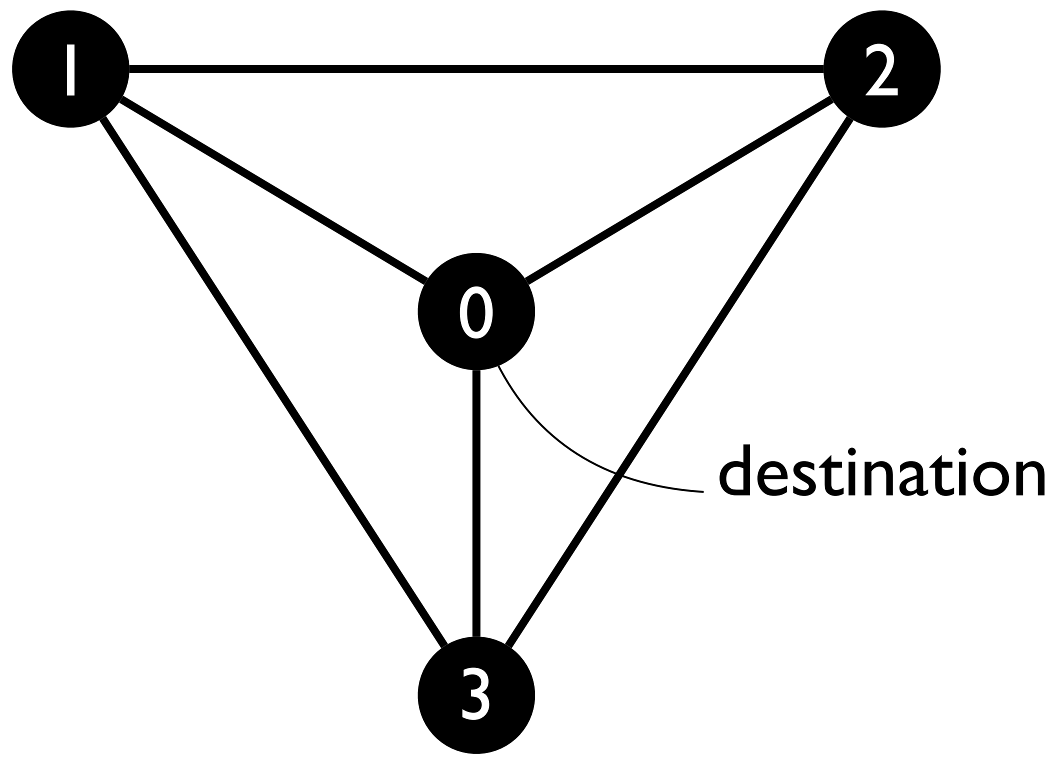
autonomous policy routing can diverge



each ASes — 1,2,3 — prefer their clock-wise neighbor

- cannot be simultaneously satisfied
- lead to permanent oscillation

autonomous policy routing can diverge



each ASes — 1,2,3 — prefer their clock-wise neighbor

- cannot be simultaneously satisfied
- lead to permanent oscillation

sufficient convergence condition

- no circular preference

existing approach: specialized but restricted combinatory structure, expert-guided manual reasoning

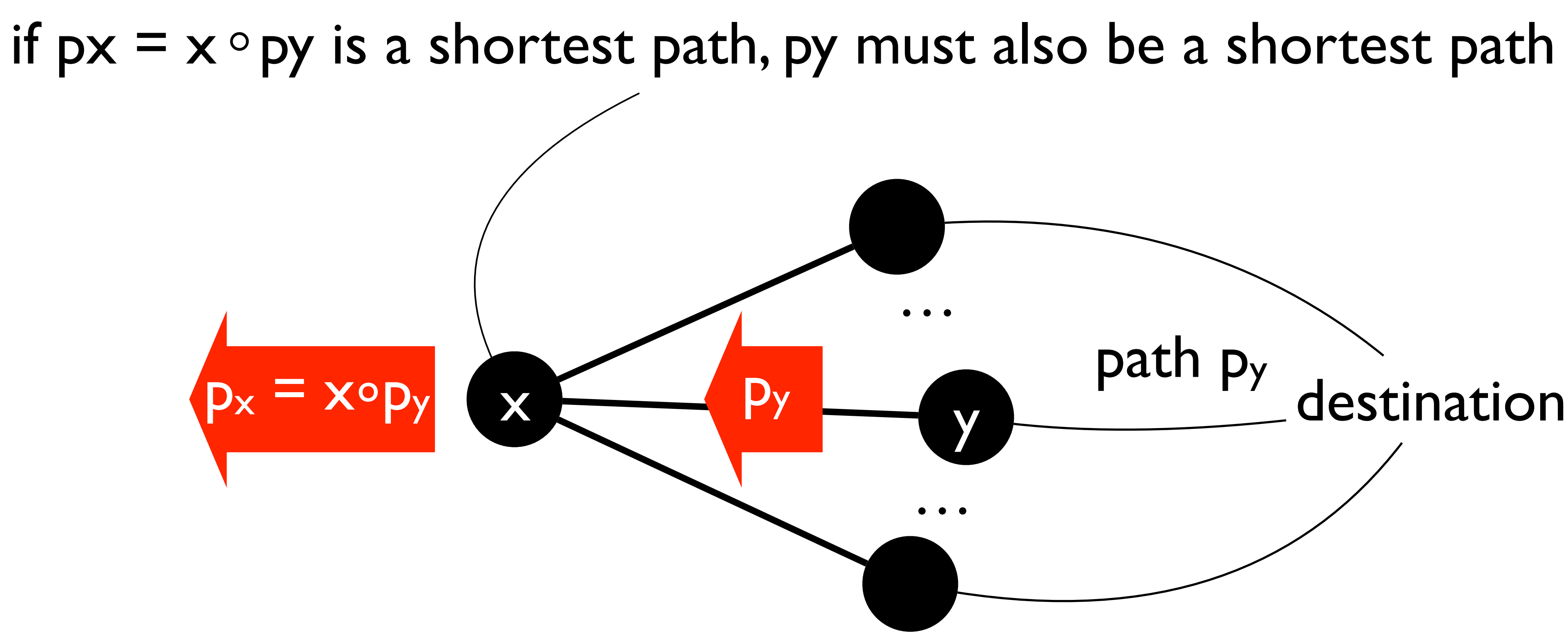
this talk

ASP offers a significantly better solution — accurate characterization, automated detection

shortest path policy revisited

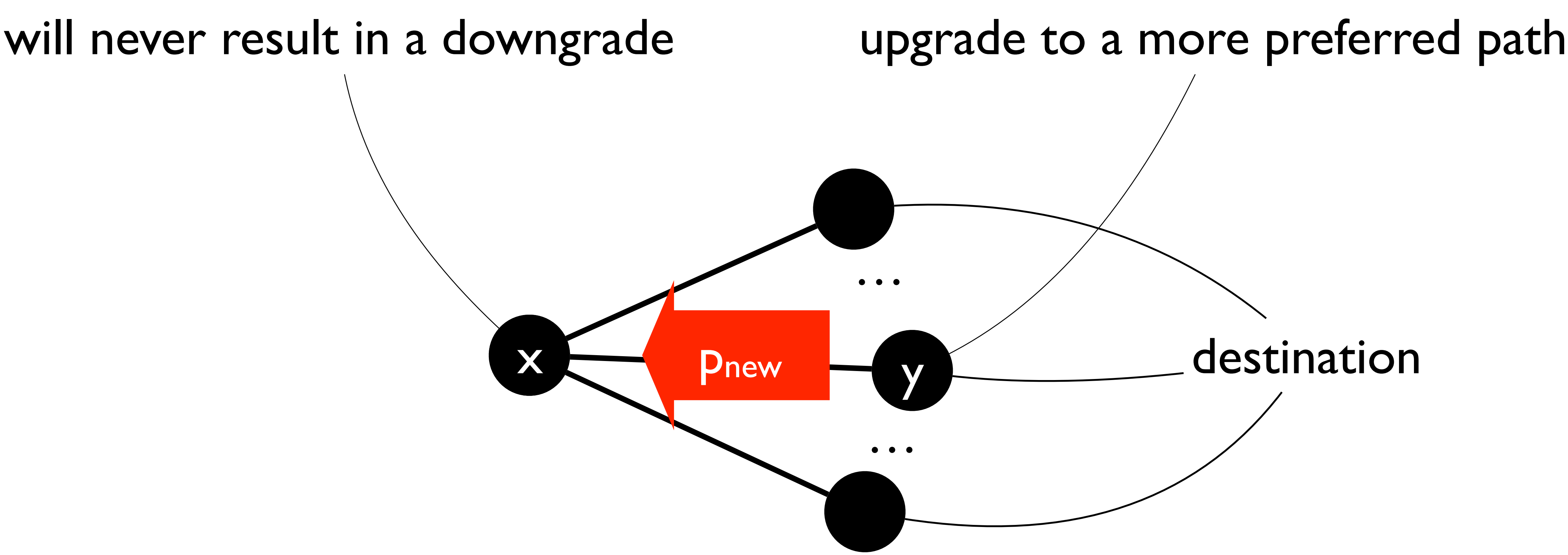
Bellman-Ford equation $sp(x) = \min_y \{x \circ sp(y)\}$

- sp(x) denotes the shortest path from x to a destination of interest

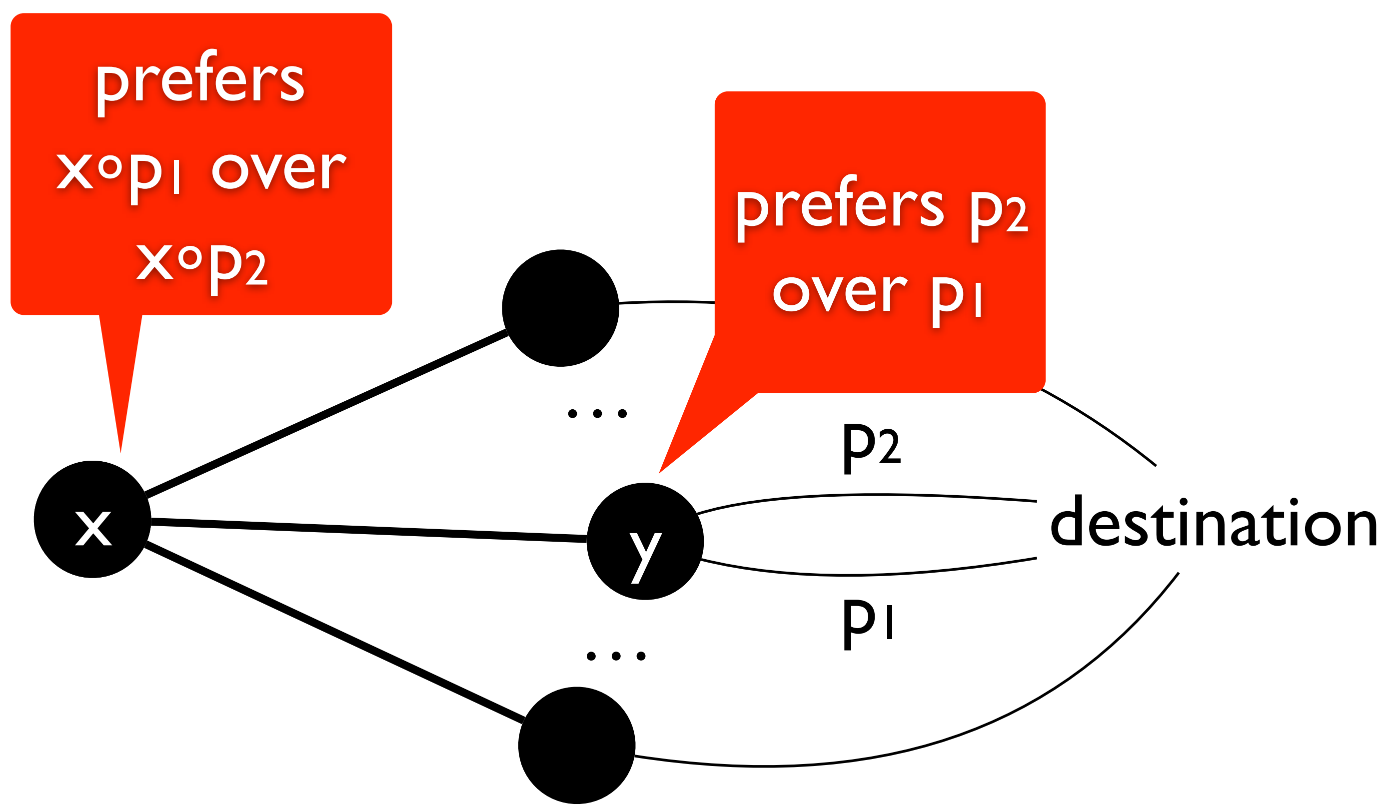


shortest path policy revisited

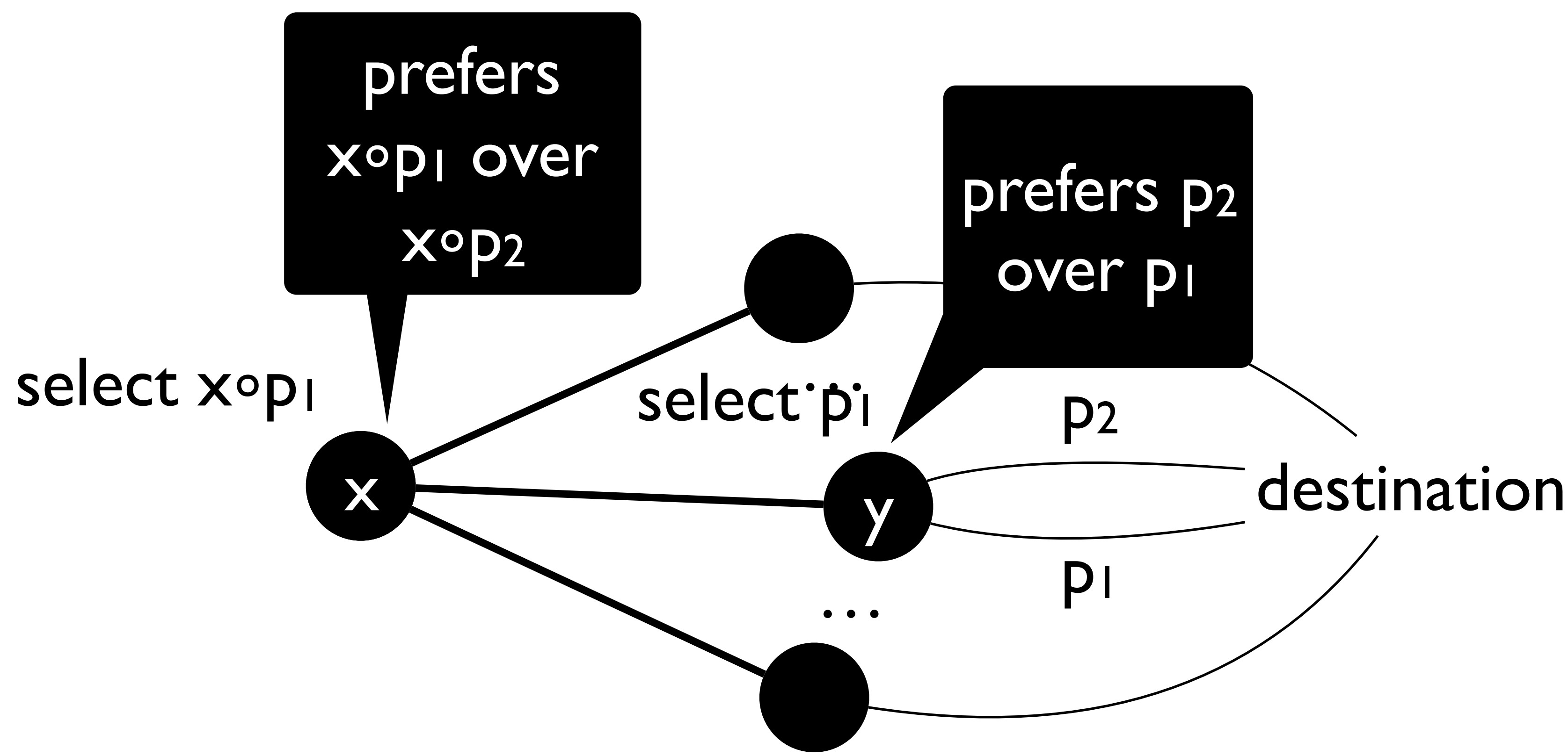
distributed shortest paths computation is *monotonic*



autonomous policies and non-monotonicity

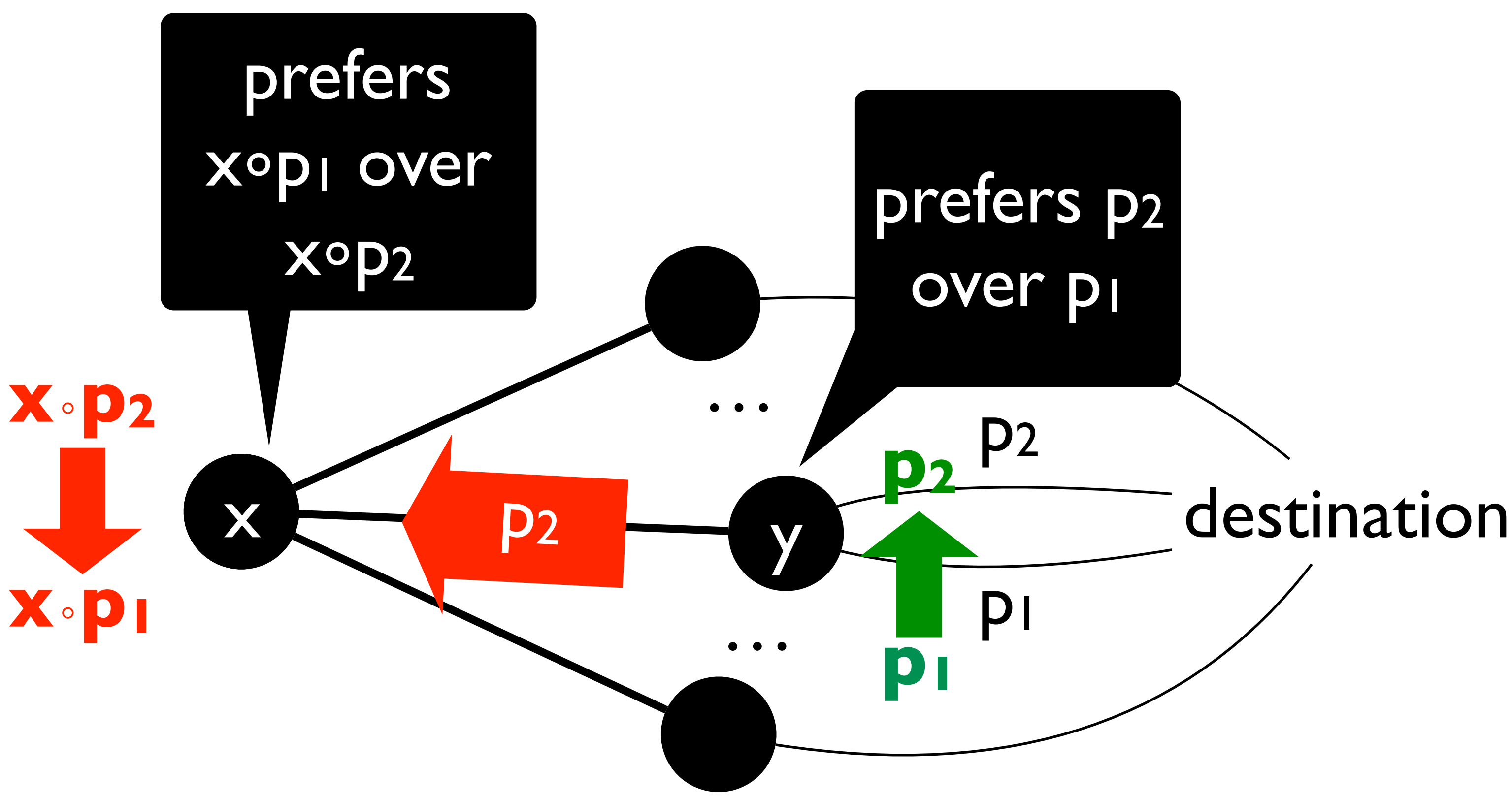


autonomous policies and non-monotonicity

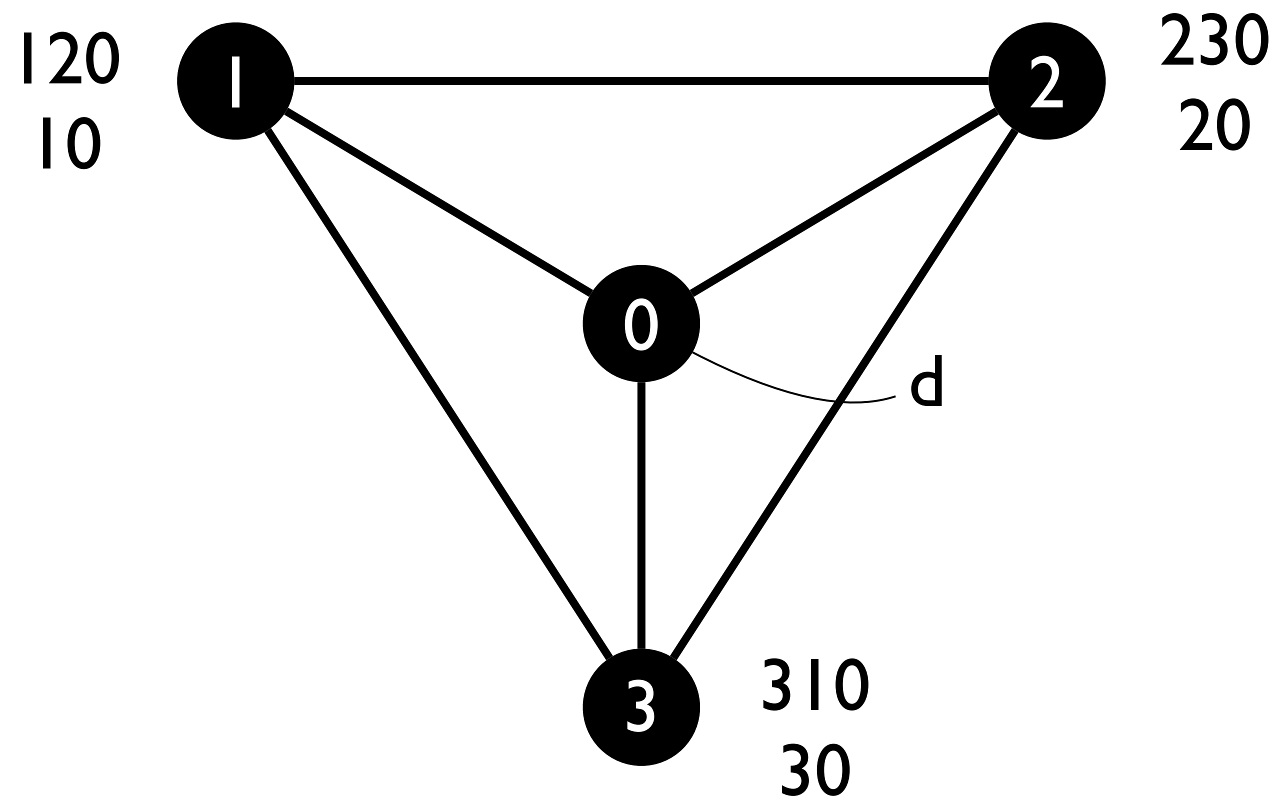


autonomous policies and non-monotonicity

distributed paths computation with autonomous policies can exhibit non-monotonic behavior



non-monotonicity explains oscillation

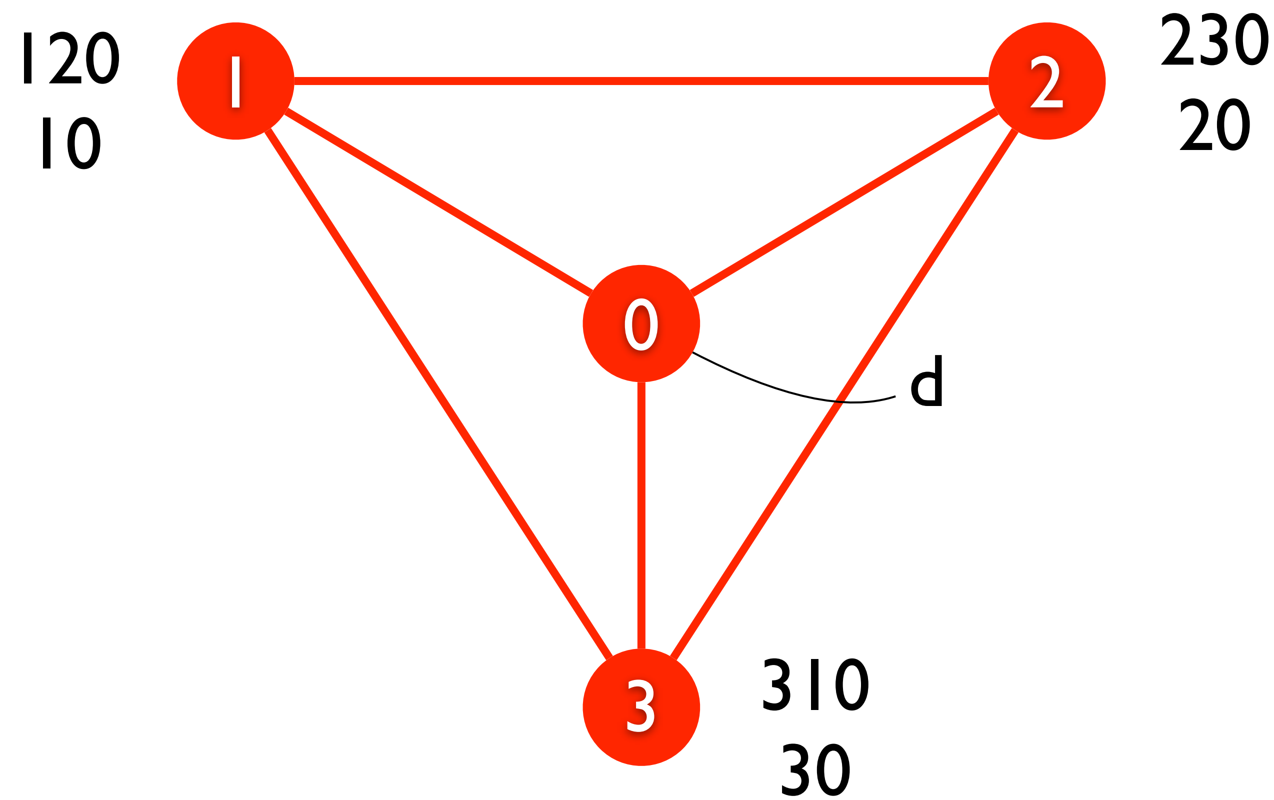


each ASes — 1,2,3 —
prefer their clock-wise neighbor
to reach the destination

modeling policy

- stable path problem (SPP) formalism
- an SPP instance *S*
  - *S = (G,P,R)*
    - *G*: AS graph with a prefixed destination owned by 0
    - *P*: permitted paths at each AS
    - *R*: path ranking function of each AS

non-monotonicity explains oscillation

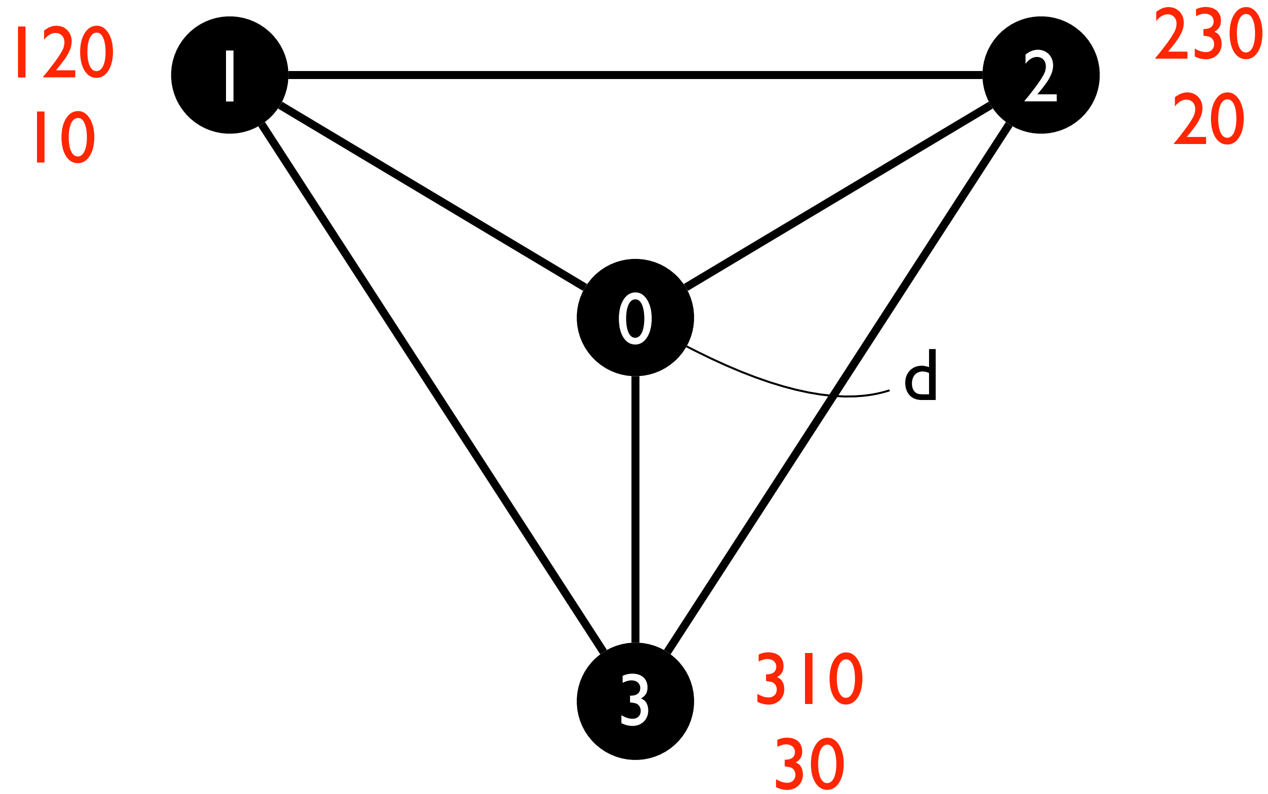


each ASes — 1,2,3 —
prefer their clock-wise neighbor
to reach the destination

modeling policy

- stable path problem (SPP) formalism
- an SPP instance *S*
  - *S* = *(G,P,R)*
    - *G*: AS graph with a prefixed destination owned by 0
    - *P*: permitted paths at each AS
    - *R*: path ranking function of each AS

non-monotonicity explains oscillation

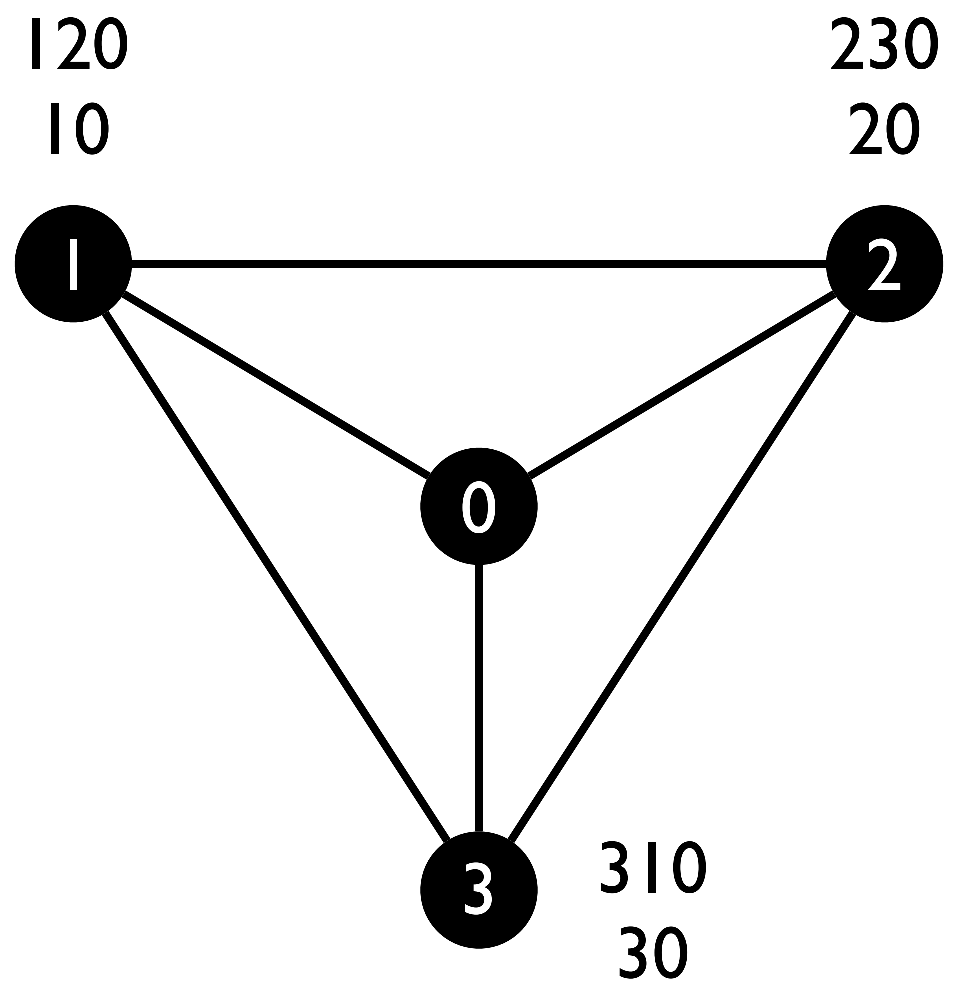


each ASes — 1,2,3 —
prefer their clock-wise neighbor
to reach the destination

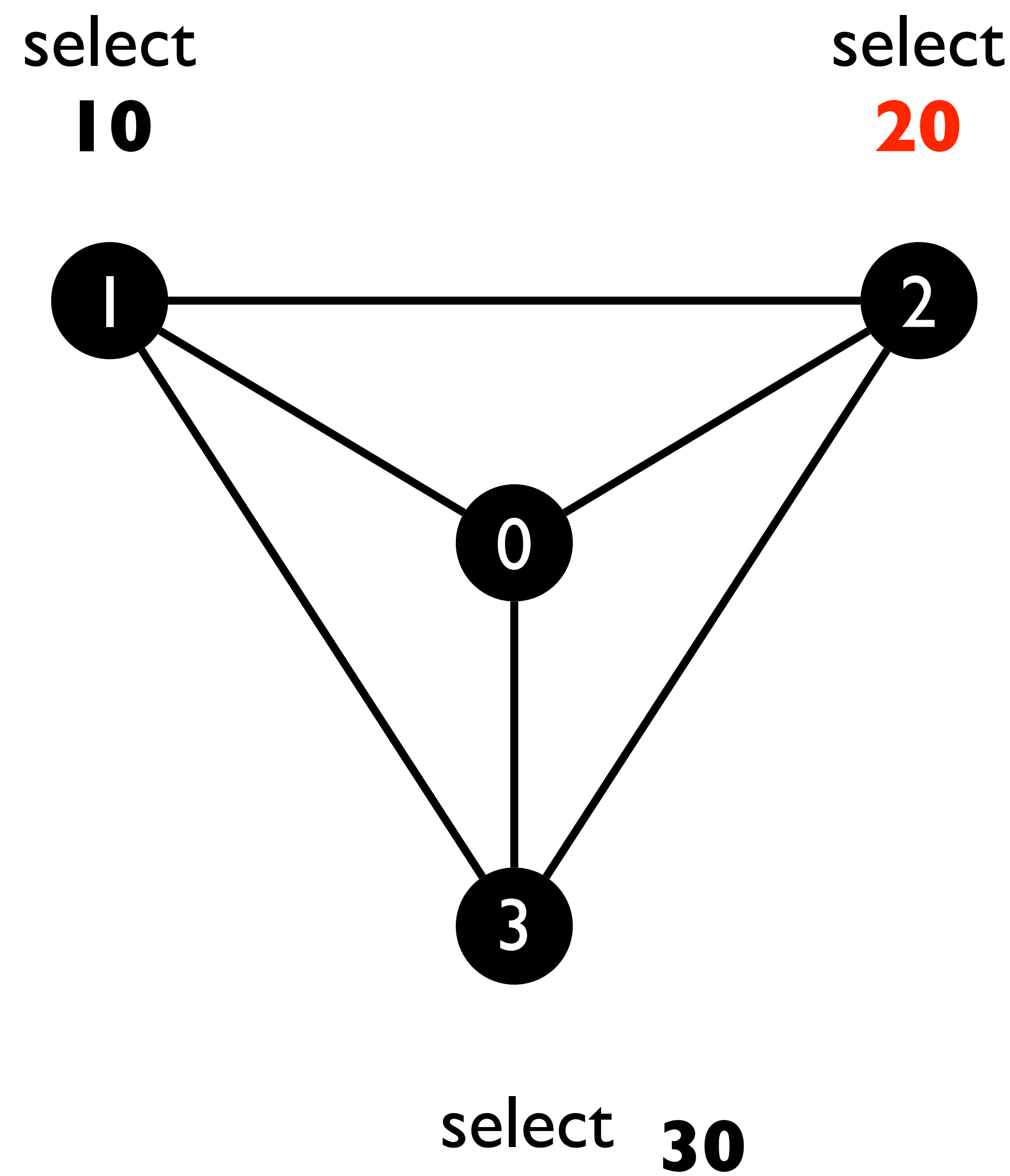
modeling policy

- stable path problem (SPP) formalism
- an SPP instance *S*
  - *S = (G,P,R)*
    - *G*: AS graph with a prefixed destination owned by 0
    - *P*: permitted paths at each AS
    - *R*: path ranking function of each AS

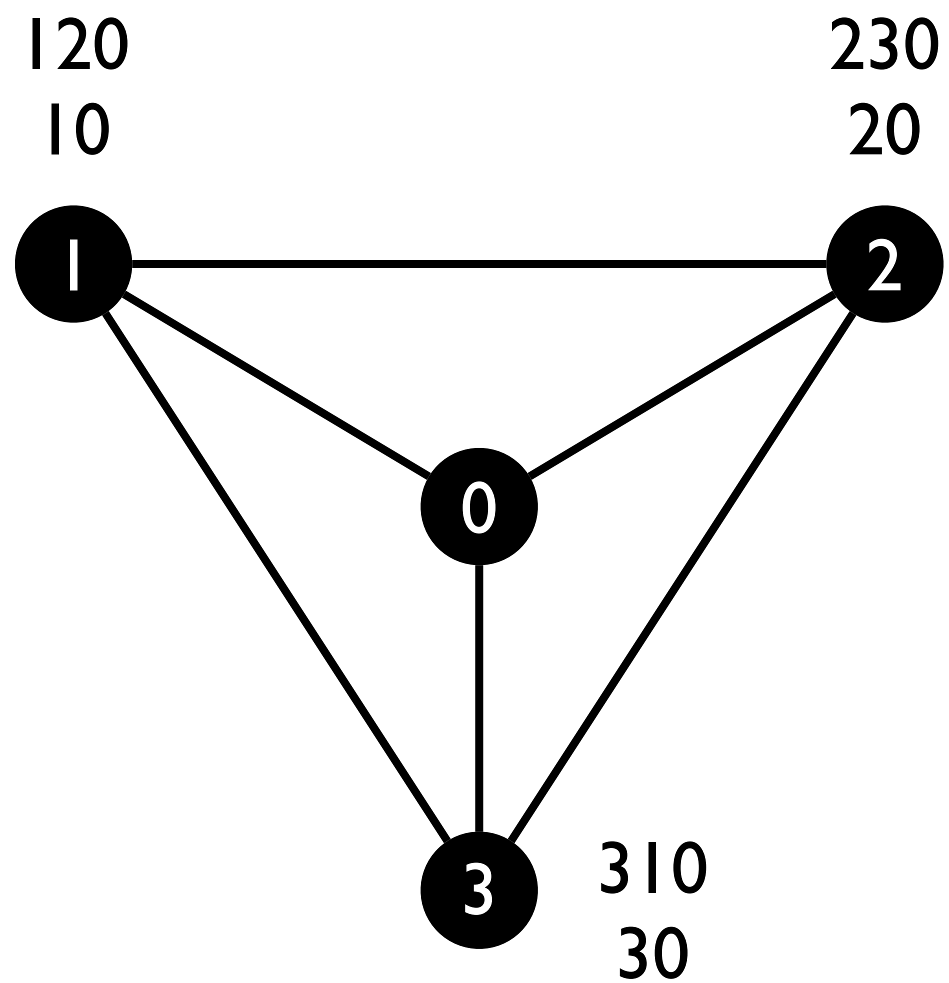
non-monotonicity explains oscillation

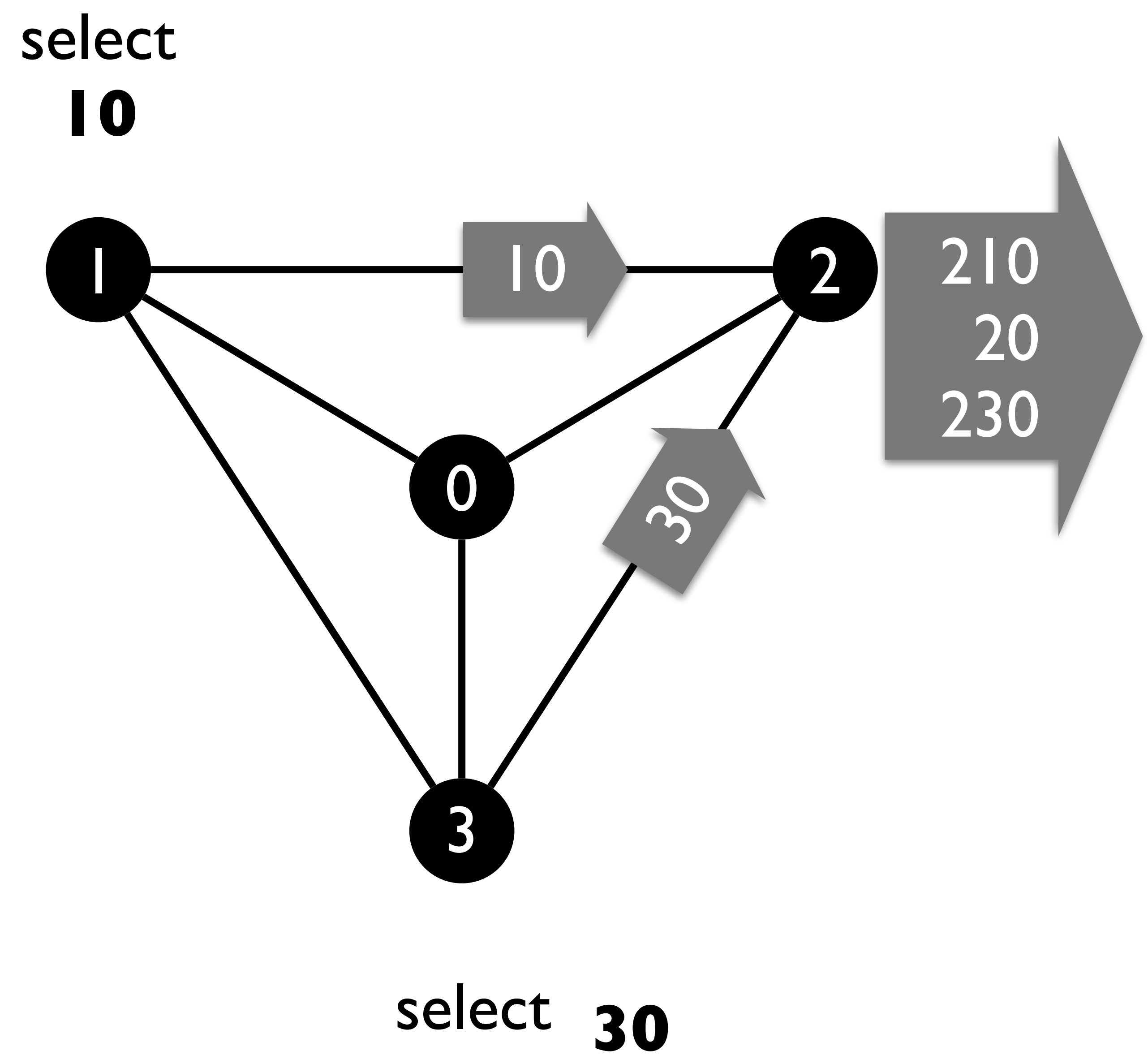


each ASes — 1,2,3 —
prefer their clock-wise neighbor
to reach the destination



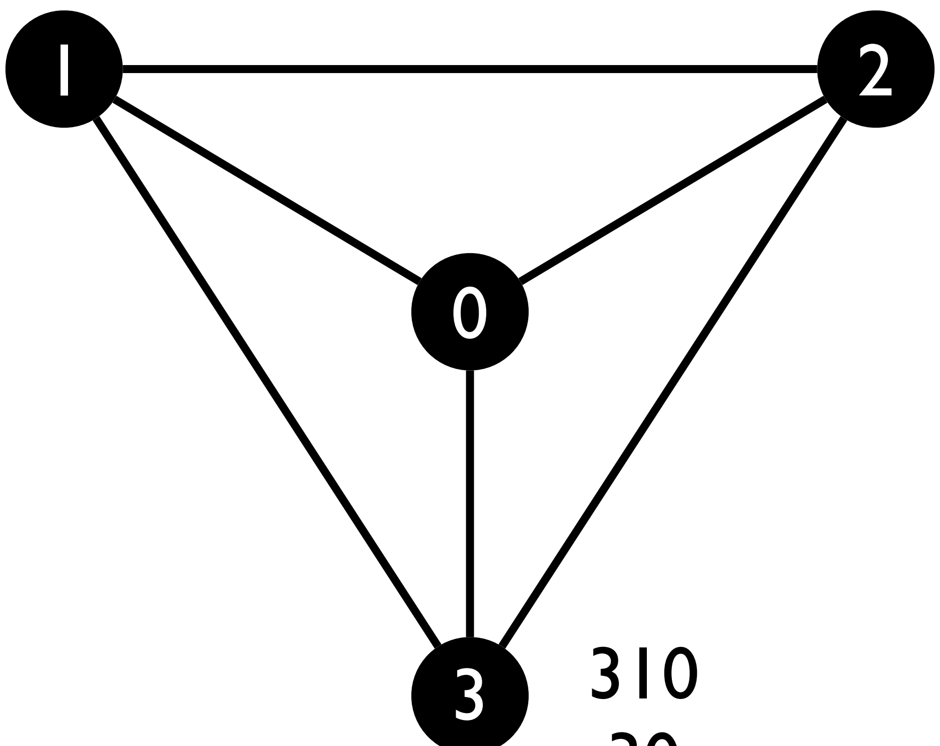
non-monotonicity explains oscillation

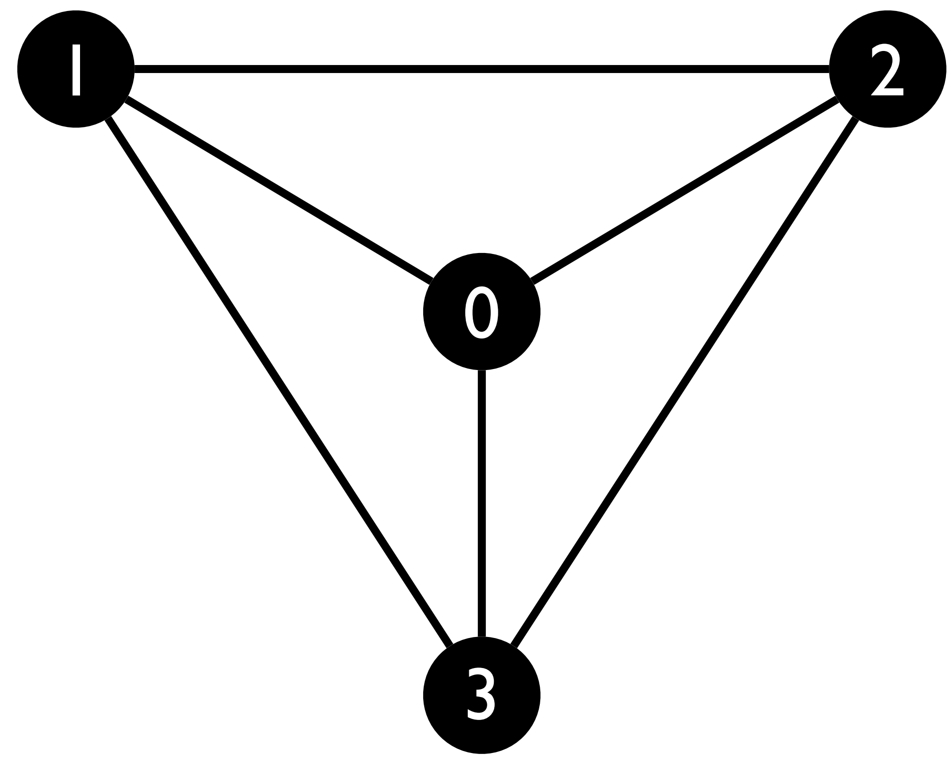




each ASes — 1,2,3 —
prefer their clock-wise neighbor
to reach the destination

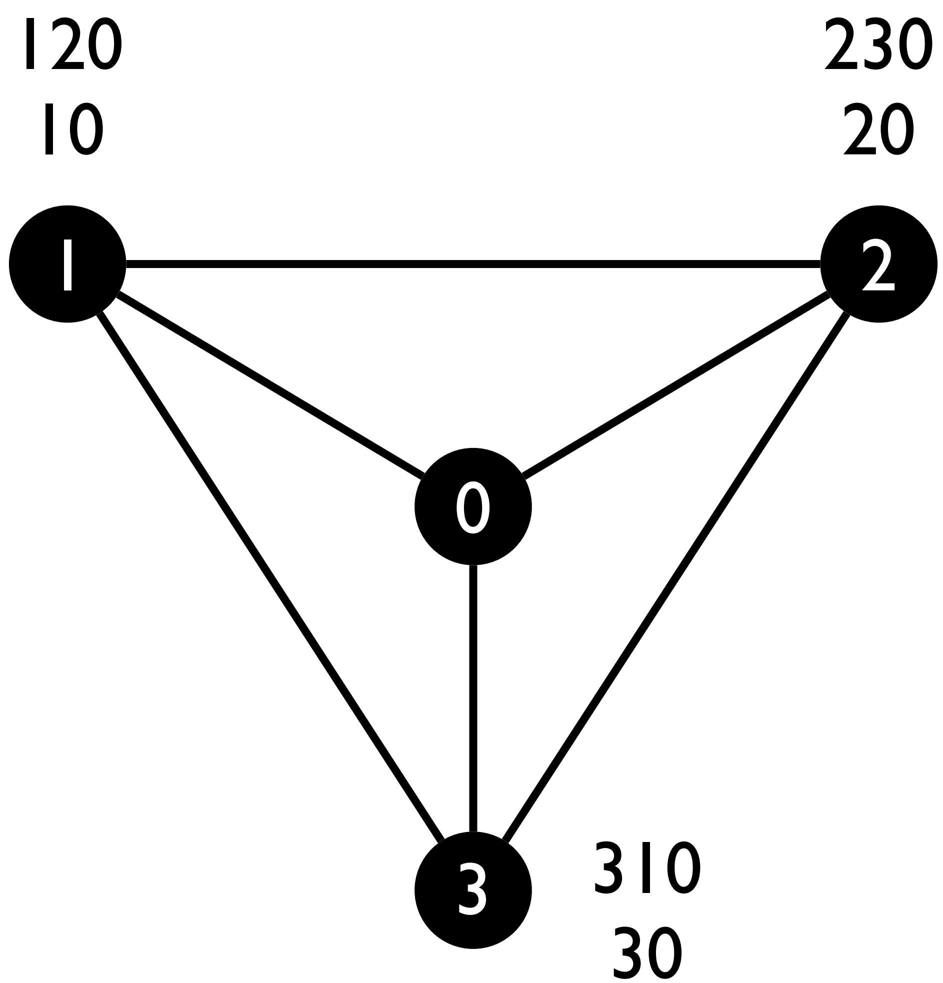
non-monotonicity explains oscillation

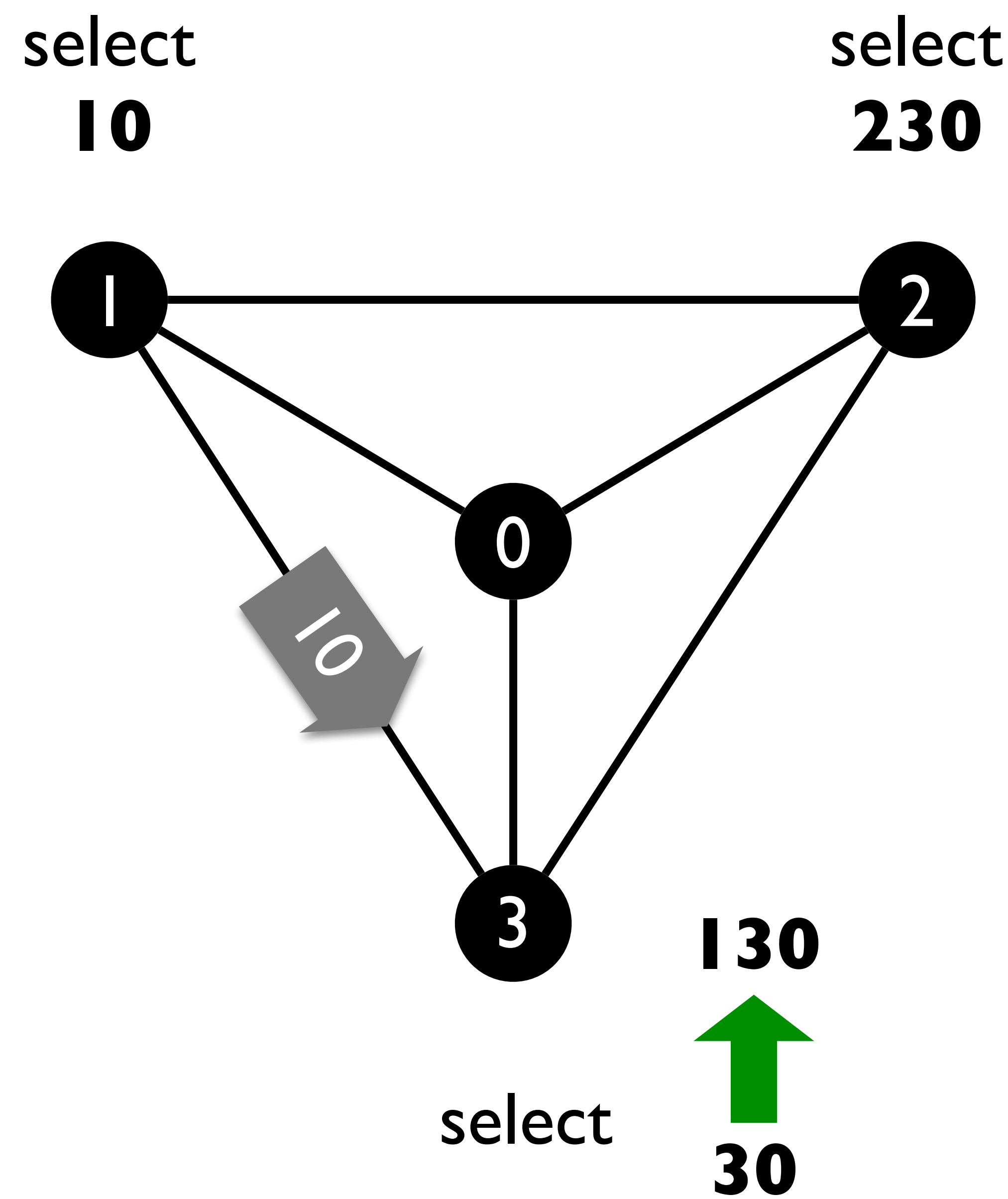




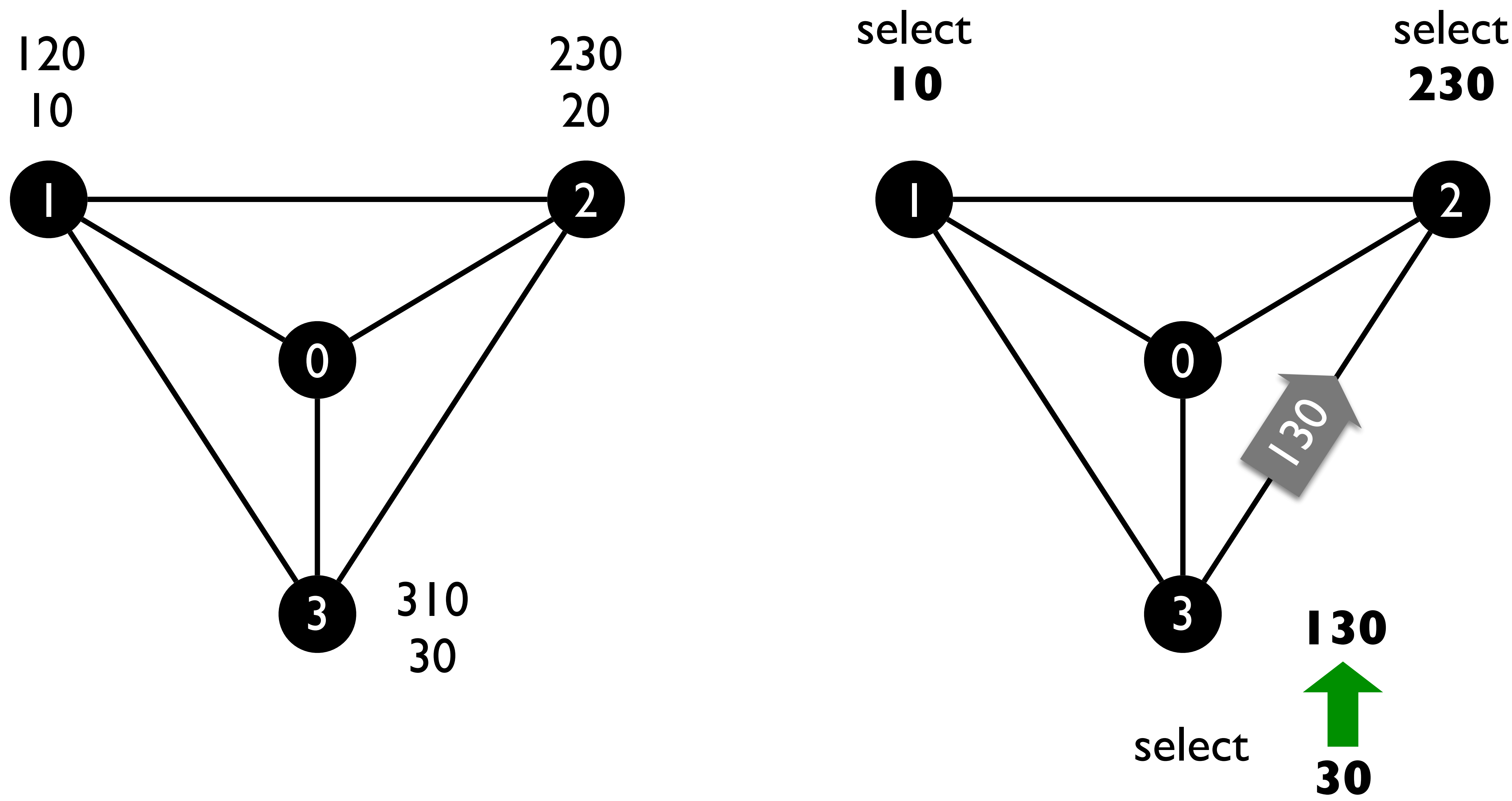
each ASes — 1,2,3 —
prefer their clock-wise neighbor
to reach the destination

non-monotonicity explains oscillation

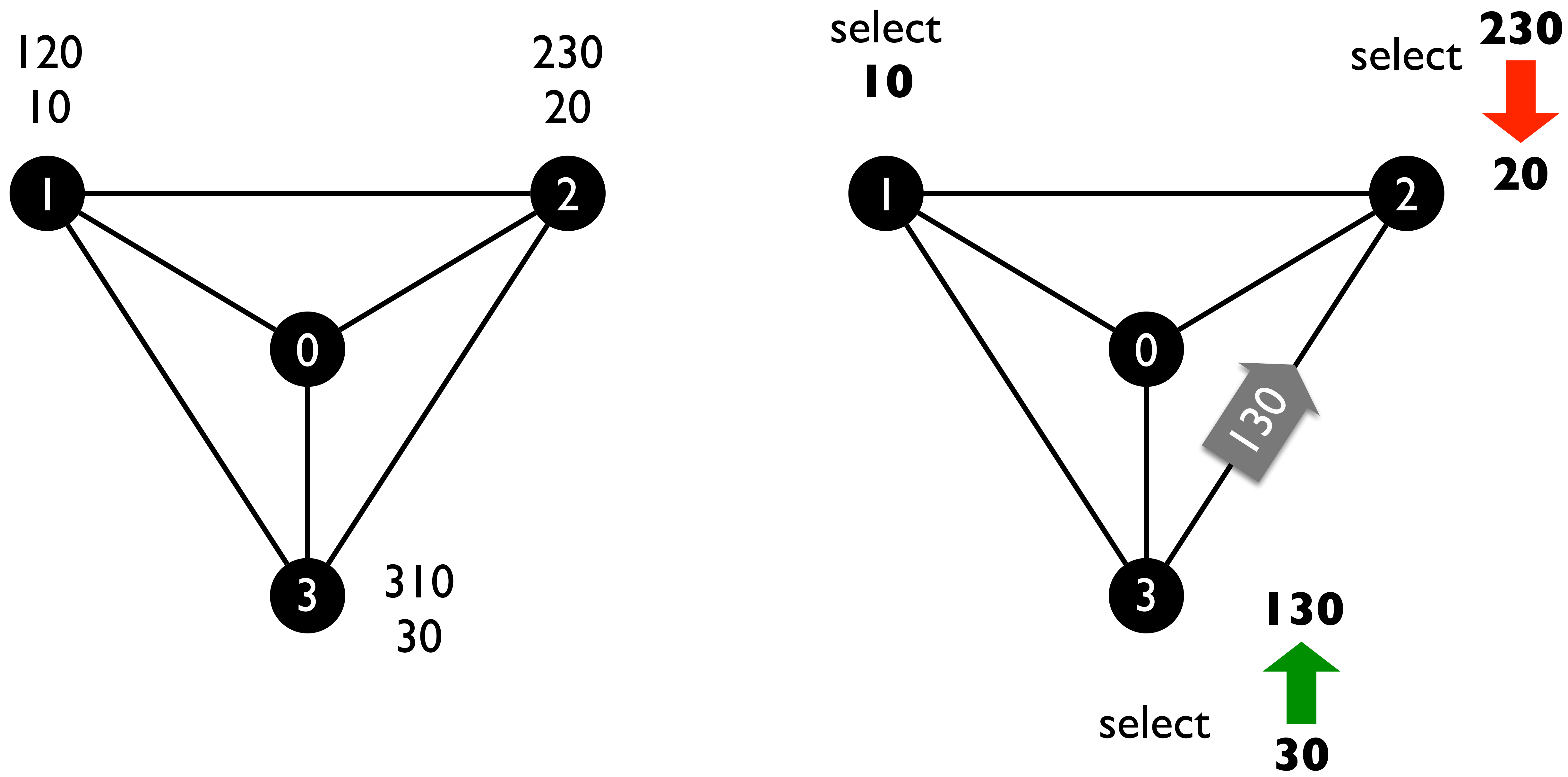




non-monotonicity explains oscillation



non-monotonicity explains oscillation



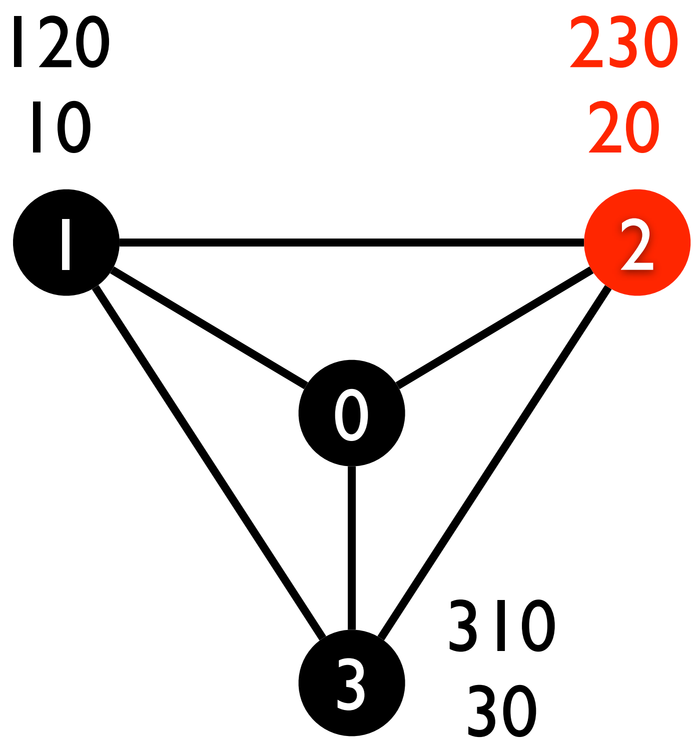
upgrade at AS 3 causes downgrade at AS 2

an ASP formulation

two predicates `r`, `b`, and a tuple variable `p`

- `r(p)` — permitted paths
- `b(p)` — selected paths
- `p` — paths

an ASP formulation



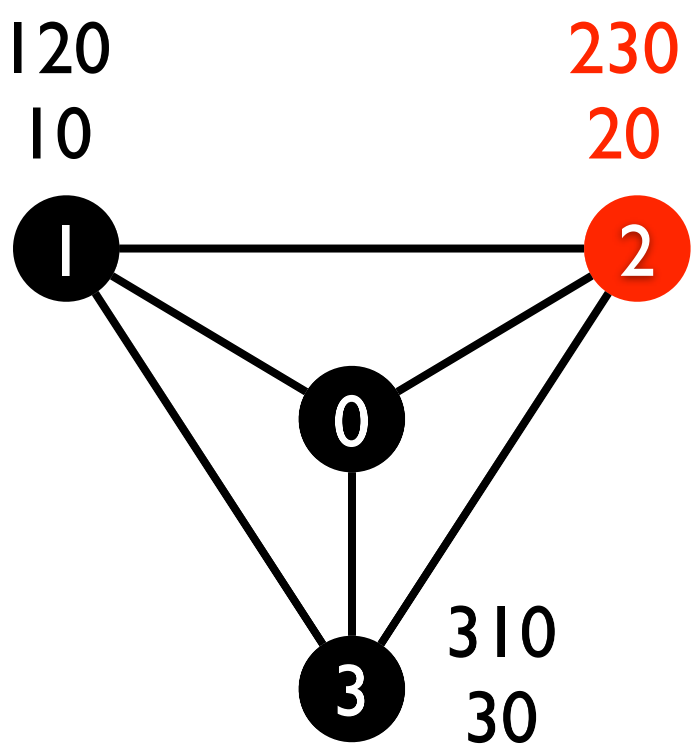
generate permitted paths

- as a direct path the destination
- as learned from a neighbor AS

```
%% direct path as known fact(s)
r((2,0)).

%% an indirect path generated by route announcement from
a neighbor
r((2,3,0)) :- b((3,0)).
```

an ASP formulation



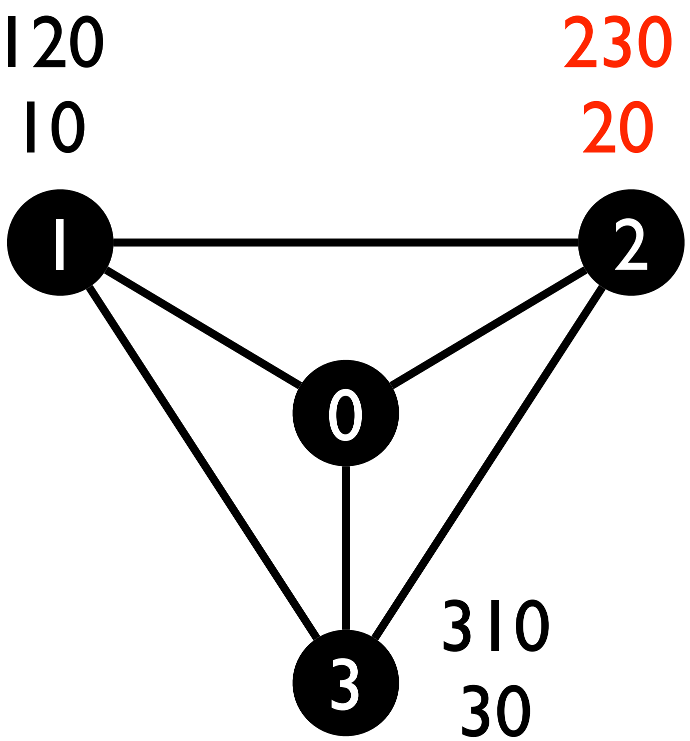
generate selected paths

- as determined by the ranking function
  - for any p_i in a list of paths $p_1, \ldots, p_n$ ranked from the most preferred to the least preferred
  - add a rule

    `b(`p_i`):- r(`p_i`),not r(`p_{i+1}`),…,not r(`p_n`)`

```
%% ranking function of the permitted paths at AS2
b((2,0)) :- r((2,0)), not r((2,3,0)).
b((2,3,0)) :- r((2,3,0)).
```

an ASP formulation

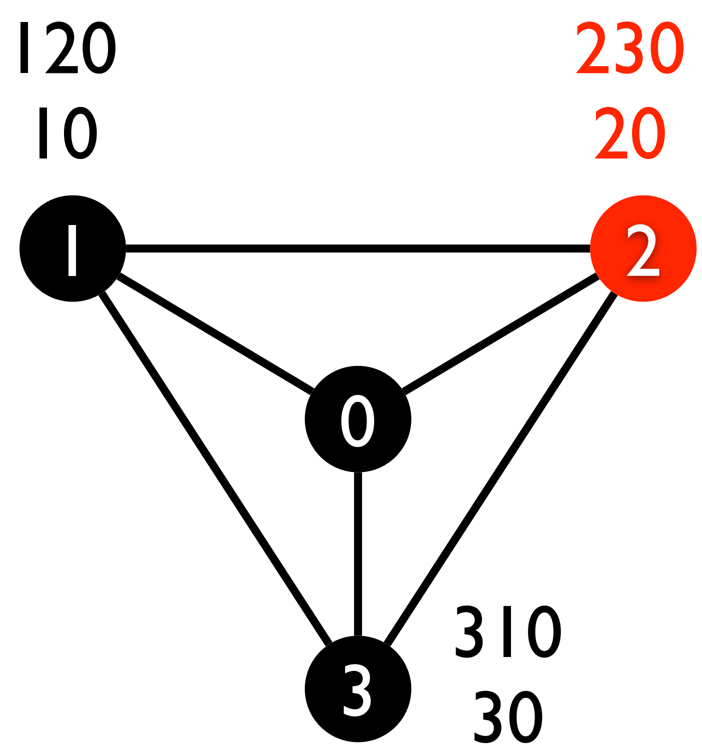


constraint

- only one path is selected as the best
  - for any p_i in the permitted paths $\{p_1, \ldots, p_n\}$, the presence of p_i will prevent the derivation of $p_j, j \neq i$

```
%% only one permitted path can be selected as the best path
:- b((2,1,0)), b((2,0)).
```

ASP formulation and oscillation detection

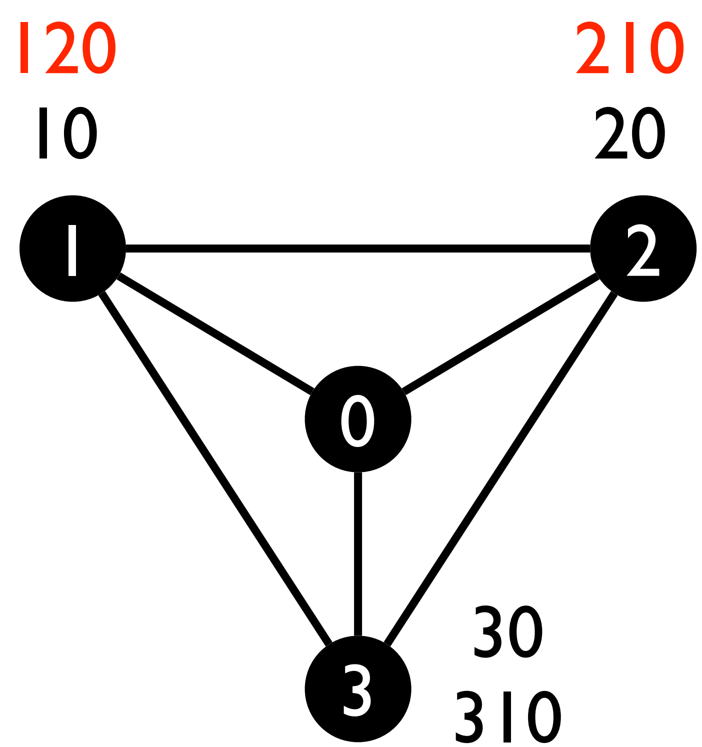



```
%% AS 2
%% direct path as known fact(s)
r((2,0)).
%% an indirect path generated by
route announcement from a neighbor
r((2,3,0)) :- b((3,0)).
%% ranking function of the permitted
paths
b((2,0)) :- r((2,0)),not r((2,3,0)).
b((2,3,0)) :- r((2,3,0)).
%% only one permitted path can be
selected as the best path
:- b((2,1,0)), b((2,0)).

%% AS 3…
%% AS 1…
```

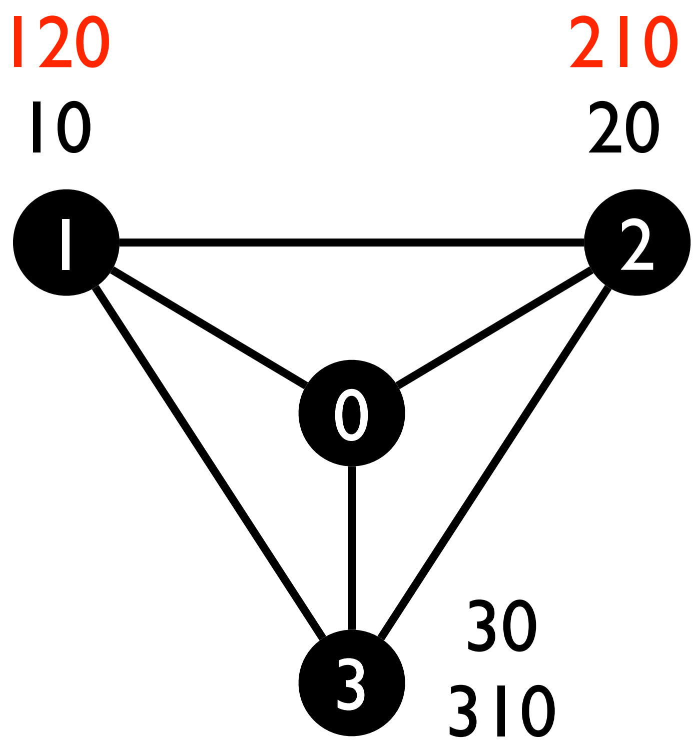
no ASP solution, signaling oscillation

transcient oscillation

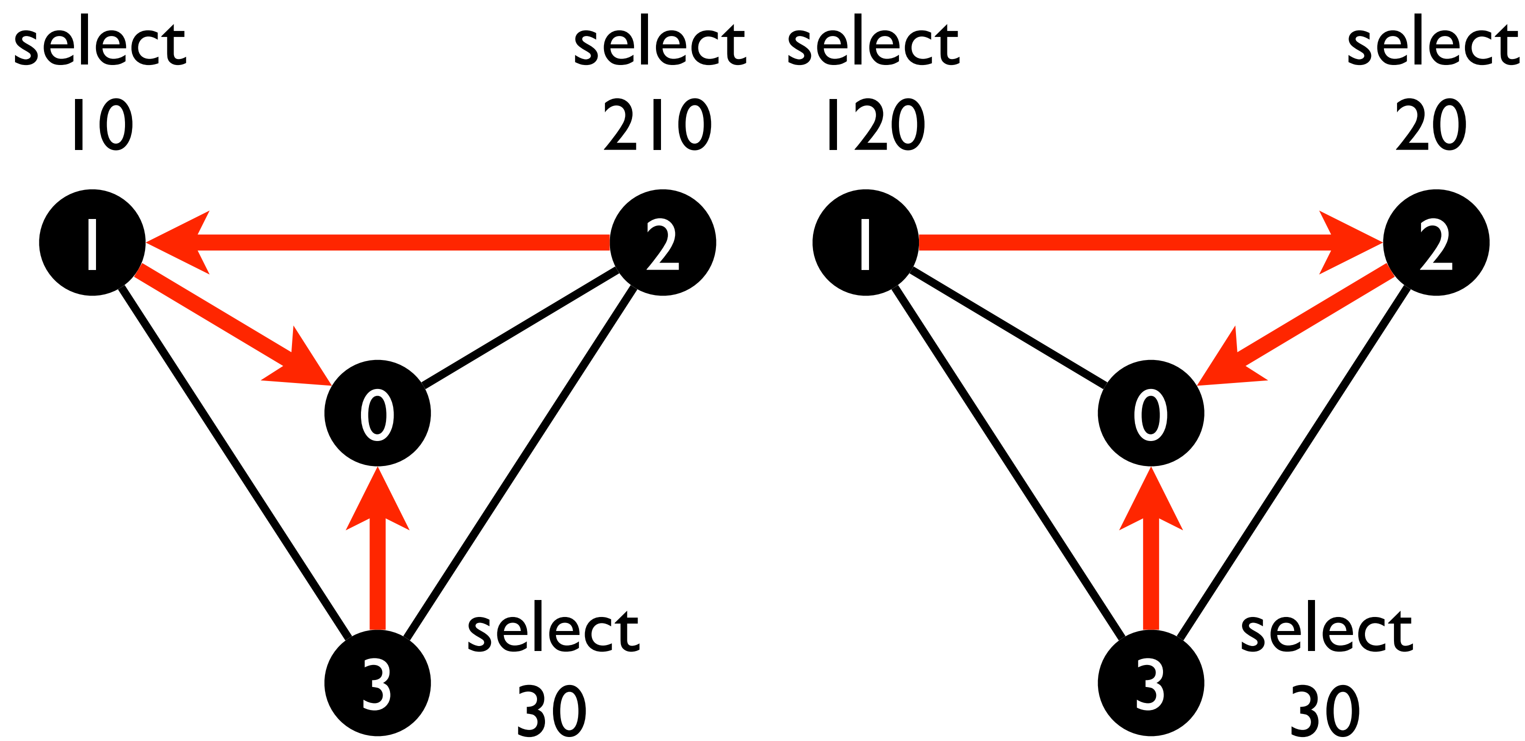


AS 1 and AS 2 each prefer paths through each other, AS 3 prefers a direct path
(cyclic preference)

transcient oscillation



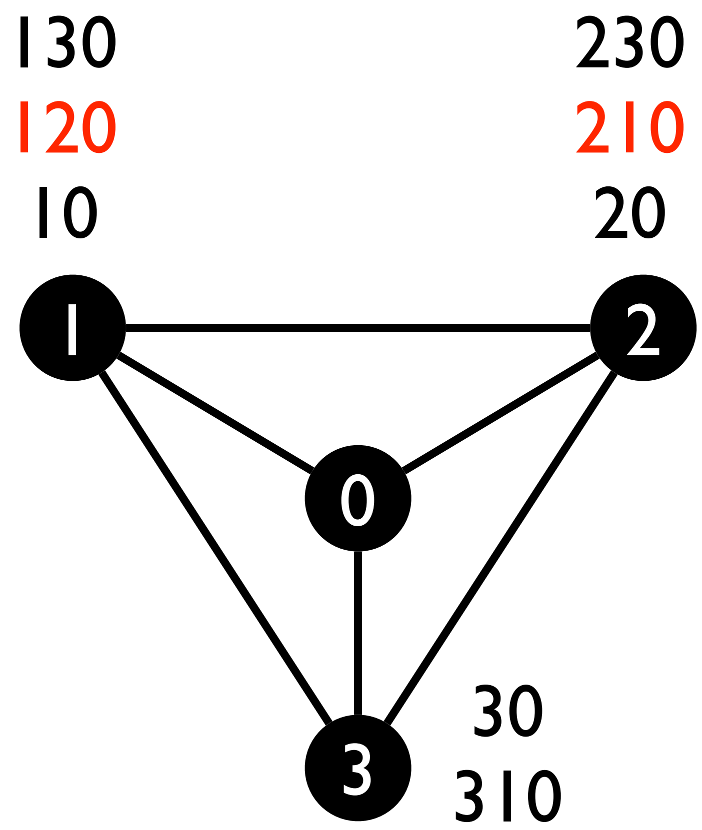
AS 1 and AS 2 each prefer paths through each other, AS 3 prefers a direct path (cyclic preference)



2 (stable) routing trees: depending on the orders of message exchanges, the routing system can converge to either path assignment; some ordering can cause permeant oscillation

2 ASP solutions, capturing the 2 stable rooting trees

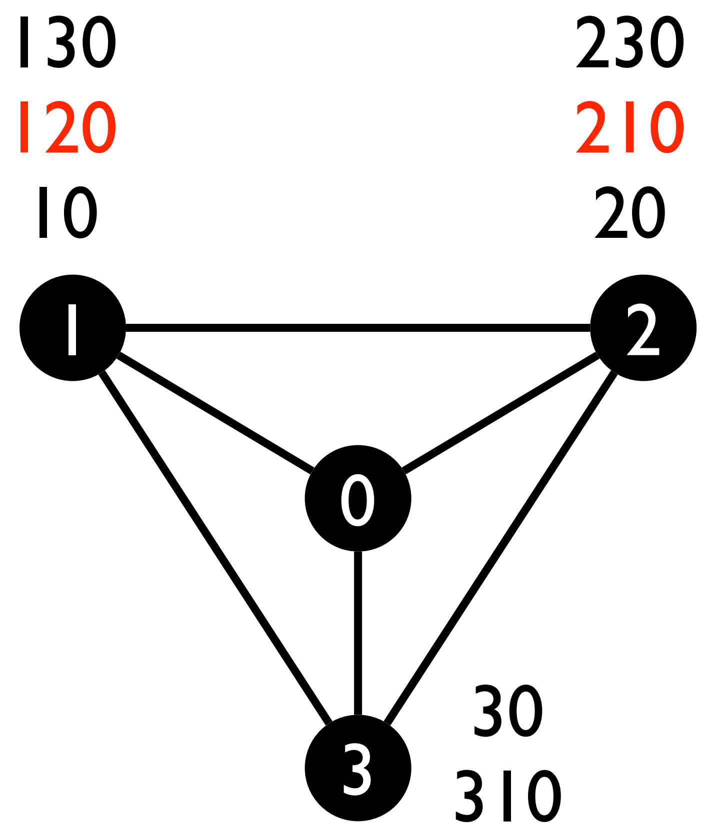
convergence with circular preference



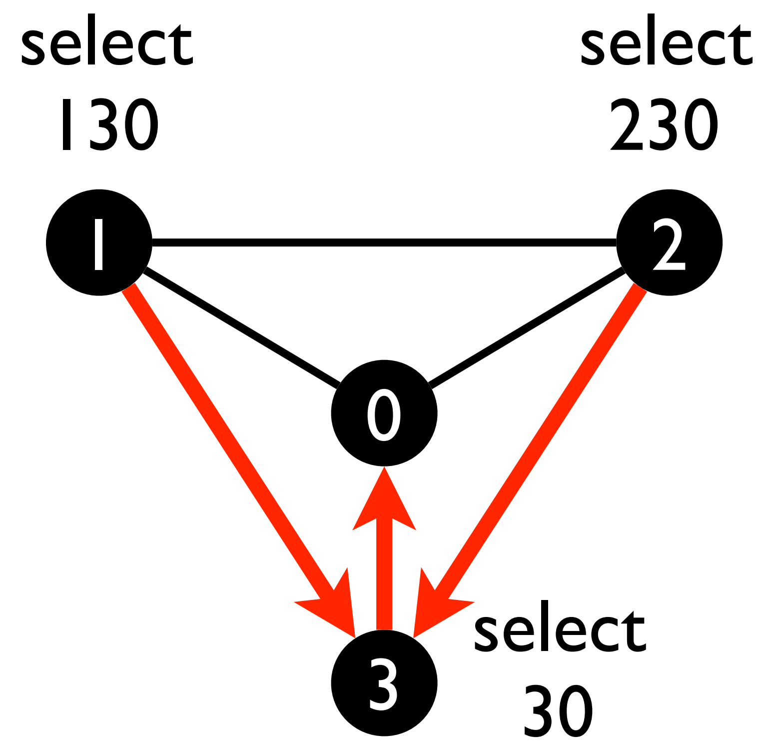
previous approach relying on the circular preference *fails!*

1 ASP solution, verifying convergence

convergence with circular preference



previous approach relying
on the circular preference
fails!



1 (stable) routing tree:
the routing system always converges
on the unique path assignment

the ASP formulation has 1 solution, verifying convergence

main conjecture

# ASP solution	oscillation analysis
0	permanent oscillation
1	convergence
>1	transient oscillation

preliminary evaluation

promising — *correct* and *scalable* with various topologies and policies

- evaluate our ASP formulation with clingo
- 3.4GHz Intel Core i5 16G RAM

preliminary evaluation

promising — *correct* and *scalable* with various topologies and policies

# nodes	# edges	node/edge ratio
1000	5092	0.196
2000	10304	0.194
4000	20766	0.192
6000	31595	0.190
8000	42880	0.187
10000	54954	0.182

topology setup

- randomly generated by GT-ITM tool
  - 2 level hierarchical graph — backbone providers and access networks
  - varying size — convenient for study scalability

preliminary evaluation

promising — *correct* and *scalable* with various topologies and policies

policy setup

- embed in the topology three policy scenarios — permanent oscillation (bad), converging (good), and transient oscilliation (disagree)
  - *good*: all nodes pick shortest paths to the destination
  - *bad*: embed circular path preferences of 3 nodes
  - *disagree*: embed circular path preferences of 2 nodes

preliminary evaluation

promising — *correct* and *scalable* with various topologies and policies

analysis result

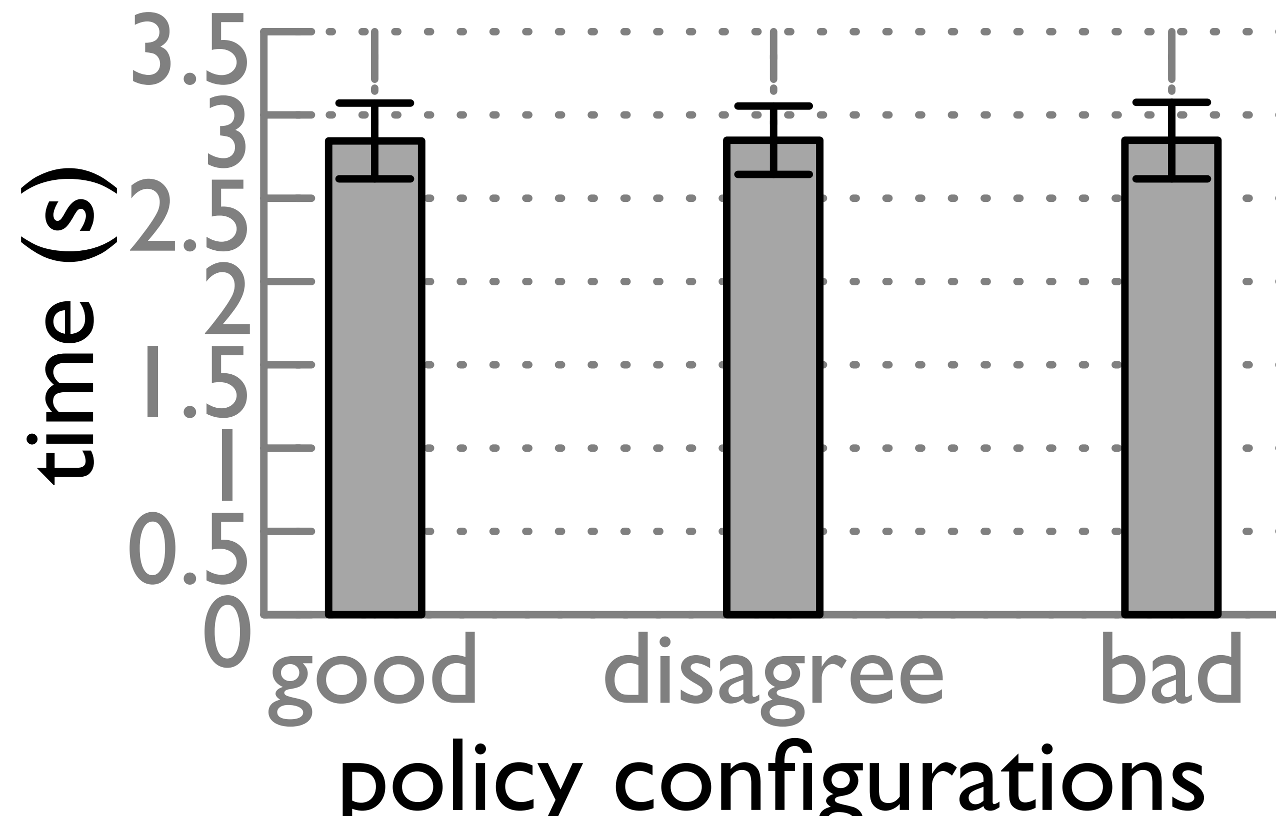
# clingo solutions	policy
1	good
2	disagree
0	bad

preliminary evaluation

promising — *correct* and *scalable* with various topologies and policies

analysis result

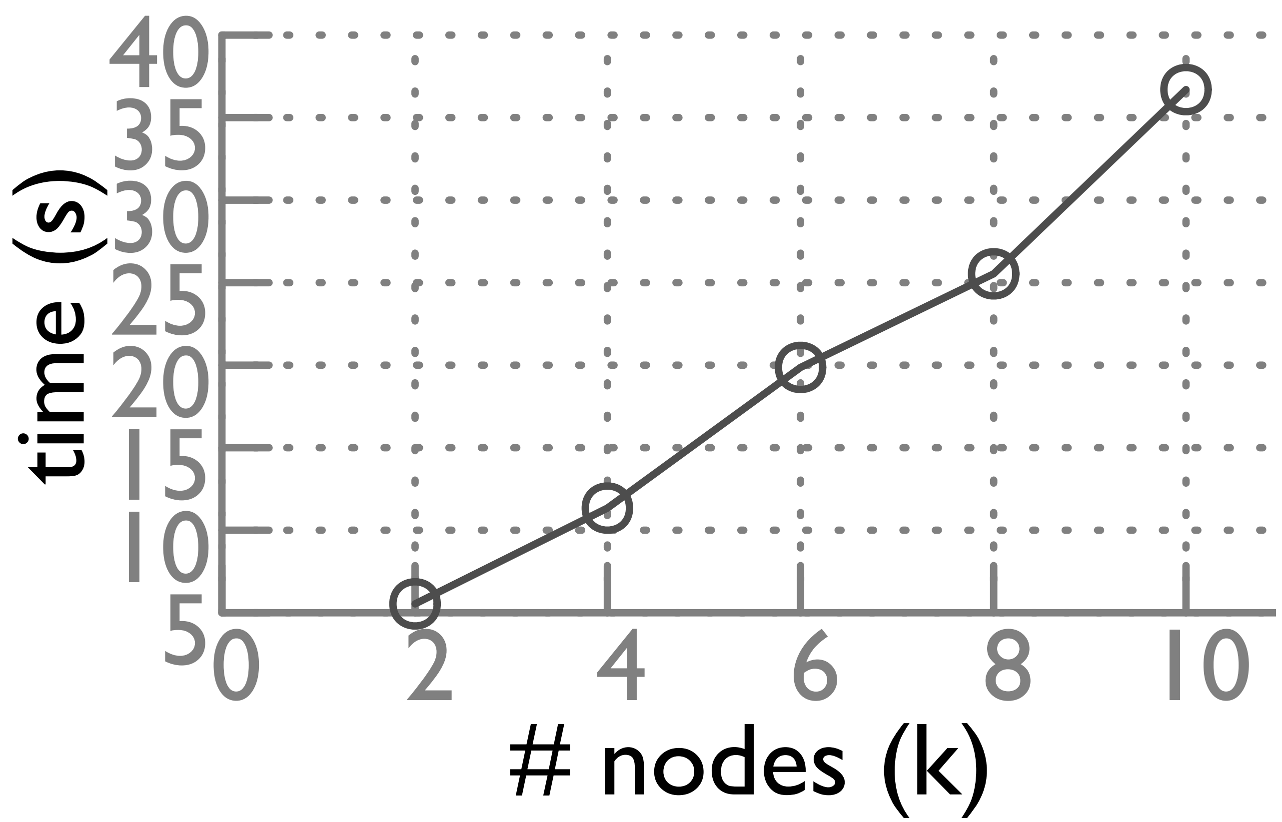
# clingo solutions	policy
1	good
2	disagree
0	bad



(topology has 1k nodes)

preliminary evaluation

promising — *correct* and *scalable* with various topologies and policies



clingo searching time for the disagree policy

- grows linearly from 6.578s to 34.786s

recap

routing oscillation and non-monotonic reasoning

previous effort in the networking community

- manual analysis — identify sufficient but *not* necessary condition
- home made combinatorial construct — circular preferences

recap

routing oscillation and non-monotonic reasoning

previous effort in the networking community

- manual analysis — identify sufficient but *not* necessary condition
- home made combinatorial construct — circular preferences

an ASP approach

- automated with ASP solver
- accurate characterization

# ASP solution	oscillation analysis
0	permanent oscillation
1	convergence
>1	transient oscillation

moving forward

customized ASP

- explain oscillation and recommend fix
  - after detecting oscillations — >1 or 0 solution(s), locate a minimal set of rules responsible for the oscillation?
- optimize routing policies
  - minimize the size of the ASP formulation while preserving the same solution?
- local evaluation
  - determine unique solution by examining policies at a subset of ASes?
- incremental evaluation
  - check whether a small modification can cause a converging system to oscillate?

moving forward

beyond liner-ranking path preferences

- path preferences beyond linear ranking
  - Multi Exit Discriminator (MED)
- verify common policy guideline
  - assume acyclic business relations (customer-provider), certain business relation based policy restrictions suppress oscillation
- analyze abstract policy
  - real-world policy often expressed as comparison of path metrics

beyond routing — *LARGE* scale distributed system

- networking oriented reasoning, tailored for distribution, aggregation, partitioning …
- distributed programming and non-monotonicity

conclusion

the Internet

- understanding the collective behavior of a large distributed system is a long-standing challenge
- state of the art: primitive home made combinatorial tools

ASP — a powerful tool

- a significantly better solution?
- customize ASP for networking?
- new ASP features with new networking problems?

Ravel: database-defined networking ravel-net.org