What's Up

- Today: Introduction to Declarative Routing
  - Goals, framework, syntax, semantics, basic networking protocols
- Tuesday: Declarative network overlays
  - Larger protocols in declarative form---DHTs, multicast, etc.

Traditional Routers

- Control Plane typically:
  - Sends and receives beacons for neighborhood detection
  - Advertises local paths to other nodes
  - Computes new paths based on detected links and received advertisements
  - Configures the forwarding plane based on those paths

Router Elements: Control

- Forwarding Plane typically:
  - Receives incoming packets
  - Looks them up in paths given by control plane
  - Sends them back along the link toward the destination
- Some variations in the interaction between the two planes, e.g. in reactive protocols
  - Incoming packet requests trigger control traffic and decision making
New Routing Approaches

- Many areas addressing these problems:
  - Modular and extensible routers
  - Active networks
  - Overlay networks

Modular & Extensible Routers

- Idea: Make it easier to construct & deploy new routers, enable code-sharing
- Still faces deployment challenges
- Not necessarily well suited for rapid or cross-layer design

Active Networks

- Idea: Enable packets to carry routing intelligence with them as general code
- Major security considerations
- Efficiency and reliability concerns
- Still writing programs rather than working with data & intent

Overlay Networks

- Idea: Add new functionality over existing infrastructure via application-layer routing
- Lots of work to develop & get right
- Still writing programs

Declarative Routing

- Basic premise: Replace the control plane with a general query/rule processor
- Express routing protocols via rules manipulating network and application data
- Goal: Enable routing protocol development much like SQL enables rapid database application development
- Goals: Easy deployment & flexibility to run many protocols simultaneously

Declarative Routers
Declarative Routing Logic

- Specify routing logic using a declarative, Datalog-based rule/query language
- DataLog syntax:
  - Conditions are predicates or tables containing basic network state data and knowledge derived as head of rules

Example: Reachability

- On a single host w/ global knowledge, can easily compute reachability in Datalog:
  R1: reachable(S,D) ← link(S,D)
  R2: reachable(S,D) ← link(S,Z), reachable(Z,D)

  "For all nodes S and Z, if there is a link from S to Z AND Z can reach D, then S can reach D."

Network Datalog

- Don’t have global data on a router
  - Have to exchange state
- Introduce operators to indicate data placement on neighboring hosts
  - Restrict to neighbors to limit communication

Path Vector Protocol

- Each node transmits advertisements to its neighbors of all known paths
- Each node receives advertisements, adds itself to paths, and forwards to neighbors

Network Datalog Path Vector

- Input: Link knowledge from router framework (e.g. receiving periodic pings)
- Output: Paths in the network, i.e. results to push to forwarding plane or return to host

Path Vector Execution
Path Vector Execution

R1: path(@S,D,P) = link(@S,D,Path(@S,D,P)),
R2: path(@S,D,P) = link(@S,S), path(@S,D,P), P=(S,D),
Query: path(@S,D,P)

Communication patterns are identical to those in the actual path vector protocol.

All Pairs Best Path

R1: path(@S,D,P,C) = link(@S,D,C), P=(S,D),
R2: path(@S,D,P,C) = link(@S,S), path(@S,D,P,C), C=C,C,
P=(S,D),
R3: bestPathCost(@S,D,C) = path(@S,D,C),
R4: bestPath(@S,D,C) = bestPathCost(@S,D,C), path(@S,D,P,C),
Query: bestPath(@S,D,P,C)

Customizable All Pairs Best Path

R1: path(@S,D,P,C) = link(@S,D,C), P=(S,D),
R2: path(@S,D,P,C) = link(@S,S), path(@S,D,P,C), C=FM(C,C),
P=(S,D),
R3: bestPathCost(@S,D,C) = path(@S,D,C),
R4: bestPath(@S,D,C) = bestPathCost(@S,D,C), path(@S,D,P,C),
Query: bestPath(@S,D,P,C)

Customizing C, AGG and FM: lowest RTT, lowest loss rate, highest capacity, highest

Questions/Concerns

- How is truth maintenance performed?
- Still limits to application layer reasoning, e.g. path planning.
- Performance? How to aggregate control traffic, e.g. path advertisements?
- Late-binding & source routing policies?

Other Interesting Possibilities

- Verification of protocols?
- Compilation of code or even hardware for extreme high-performance?
- Tracing/explanatory answers for diagnosing network behavior
  – “Why are you routing there?”