4/19/22

**Routing**

Routing: control plane (processor)

* Doesn’t have to change table that often, once it has the best path it can send a bunch of messages
* Computing paths with the packets will follow
* Routers talking amongst themselves
* Individual router creating forwarding table
* Figuring out topology of network installs forwarding table

Forwarding: data plane (line card)

* Operating on every packet directing to an outgoing link
* Using forwarding table

Routing happens on layer 3 internet layering model

Computing the shortest paths – assuming you already know the network topology

* Path selection model
  + Destination based
  + Load insensitive(e.g. Static link wights)
  + Min hop count or sum of link weights
  + Example using round trip time or delay time or congestion(a router drops a lot of packets) or bandwidth or cost for the ISP

Link State: Dijkstra’s algo (“cheapest way to get somewhere”)

* He created semaphores
* Each node computes shortest paths to other nodes, most the time cost for companies to send it somewhere
* How do we know what the costs are?
  + Manually – doesn’t really work
  + Broadcast it out (advertise it-tell everyone what the cost is) or multicast
  + In link state it will only broadcast when something changes i.e. a link goes up or down
* Used in OSPF and IS-IS
* OSPF doesn’t scale because every router is storing the shortest paths from Dijkstra’s
* Adv: fast to react to changes
  + Nodes communicate whenever there is a change
* O(# of edges)
* Trusts a peers info, but does computation its self

Static and dynamic routing (a lot is static in the world)

4/26/22

**Finishing routing**

Ip, netmask, default gateway needed for???

Distance vector: bellman-ford algo

* Used in RIP and EIGRP
* Define distances at each node x, and update distances based on neighbors
* Initially each node in the network only knows how to get to its neighbors, then updates distances based on neighbors knowledge
* Sends updates periodically or routing decision change
* Adv: less info and lower computational overhead
  + Each node doesn’t have to have a view of the entire network unlike link-state
* O(# of neighbors \* # of nodes)
* Trusts a peers routing computation

Similarities of link state and distance vector routing

* Both compute shortest path routing
* Commonly used **inside** an organization
* RIP and OSPF are mostly used as intra-domain protocols
* E.g. a small business uses RIP and AT&T uses OSPF

Path-Vector Routing

* Link state is problematic
  + Topology info is flooded(high bandwidth and storage overhead)
  + High processing overhead
  + Typically only used in OSPF and IS-IS
* Distance vector is on the right track
  + Adv: hides details of topology of network, nodes only determine next hop
  + Disadv: minimizes some notion of total distance
  + Idea: extend notion of distance vector
* Path-vector
  + Extends distance vector routing by supporting flexible routing policies
  + Key idea: advertise the entire path
  + Used in BGP
  + Nodes can easily detect a loop
  + Nodes can simply discard a path with a loop
  + Each node can apply local policies
  + Path selection: what path to use
  + Path export: what path to advertise

End to end signaling

* Used in MPLS with RSVP

Conclusions

* Path vector routing
  + Share entire path, not distance: faster convergence
  + More flexibility in selecting paths
* Distance vector routing
  + Pro: less info and computation than link state
  + Cons: slower convergence
* Different goals/ metrics if inter-domain or intra-domain

4/28/22

**Routing convergence**

Topology changes

* Planned: maintenance on node, shut down node energy saving, traffic engineering: changing routing configuration for something like money
* Unplanned failures: fiber cut, faulty equipment, power outage, software bugs

Detecting topology changes

* Beaconing: periodic “hello” in both directions and detect a failure after a few missed hellos like an acknowledgment
* Performance tradeoffs: detection delay, overhead on link bandwidth and cpu, likelihood of false detection

Link-state routing (OSPF) convergence

* Control plane: all the node have the same info
* Data plane: all nodes forward packets in a consistent way

Transient disruptions

* Detection delay: a node does not detect a failed link immediately and forwards data packets into a “blackhole”. Depends on timeout for detecting lost hellos
* Inconsistent link state database: some routers know about failure before others, these inconsistent paths cause transient forwarding loops

Convergence delay

* Sources and other on this slide

Reducing convergence delay

* Faster detection: smaller hello timers, better link layer tech
* Faster control plane: flooding immediately, sending routing messages with high priority
* Faster computation: faster processors, and incremental computation
* Faster forwarding table update: data structures supporting incremental updates

Slow convergence in distance vector routing

Distance vector: link cost changes

* Link cost decreases and recovery
  + Node updates the distance table
  + Of cost changes in least cost path, notify neighbors
* Link cost increases and failures
  + Bad news travels slowly
  + “count to infinity” problem (iterating by one in example because of the 1 weight)

Distance vector: poison reverse

* Make the wight infinite instead of incrementing like above

**Final Studying**

* CDN
* Hashing
* Middle boxes
* Routing
* Convergence

CDN (Content Distribution Networks)

* A single server could be easily overloaded from a flash crowd
* A single server has a single point of failure
* Far away from most people
* Proxy caches (Web caching)
  + Forward proxy: Keeps data closer to client so they don’t have to go all the way to the server.
    - Explicit proxy requires configuration
    - Implicit proxy service provider deploys it and intercepts web packets on path
  + Reverse proxy: close to server. Either run by server or third party CDN.

(A) Forward (B) Reverse (C) Both (D) Neither

* Reactively replicates popular content(C)
* Reduces origin server costs(C)
* Reduces client ISP costs(A)
* Intelligent load balancing between origin servers(B)
* Offload form submissions(POSTs) and user auth(D)
* Content reassembly or transcoding on behalf of origin(C)
* Smaller round-trip times to clients(C)
* Maintain persistent connections to avoid TCP setup delay (handshake, slow start) (C)
  + Limits of web caching:
    - A lot of content isn’t cacheable: dynamic data, CGI scripts, cookies, SSL, analytics
    - Stale data or overhead of refreshing cached data
* CDN’s
  + Proactive content replication: content provider contracts with cdn
  + CDN replicates content on many servers
  + Updates the replicas when content changes
  + Sever selection policy requires continues monitoring of liveness, load, and performance
  + Akamai is a CDN
  + BGP (border gateway protocol) routing is the protocol underlying the global routing system of the internet
  + Mapping system of CDN
  + Adapting to failurs
  + DASH (streaming multimedia): dynamic, adaptive streaming over http
    - Server divides video into chunks and encoded at different rates
    - Client chooses max coding rate
    - “intelligence” at client: client determines when what where
  + POP: point of presence. Near but not within access networks

Hashing in networks

* Main goals: lower cost, deterministic uniformity (load balanced)
* Equal cost multipath routing switches network load balancing
* Per – flow statistics in switches (QoS, IDS)
* Caching in cooperative CDN’s and P2P file sharing
* Data partitioning in distributed storage services
* Strategies: modulo, consistent, bloom filters
* Uses of hashing:
  + ECMP (equal cost multipath routing)
    - Splits traffic over multiple paths for load balancing
    - Why not round robin: reordering, different RTT per path, different MTU(max transfer unit) per path
    - Path selection: # of buckets = # of outgoing links
    - Data centers networks are multi-rooted
  + Network load balancing
    - Goal split requests evening over k servers
    - 3 approaches:
      * Load balancer terminates tcp
      * Virtual ip / dedicated ip
        + NAT approach: replace virtual ip with servers actual ip
        + Direct server return (DSR)
    - DSR
      * Server binds to both virtual and dedicated ip
      * Load balancer just replaces MAC addr

Middle boxes and tunneling

* NAT(Network address translation)
* Firewalls
* LAN appliances
* Overlay networks: “on top” of internet
  + Tunnels between host computers
  + Provide better control flexibility, QoS (quality of service), Isolation
* underlay tunnels: “below” IP route
  + Across routers within AS
  + Provide better control flexibility, QoS (quality of service), Isolation
* Network layer principles
  + Globally unique identifiers: each node has a unique, fixed IP address and reachable from everyone and everywhere
  + Simple packet forwarding: network nodes simply forward packets rather than modifying or filtering them
  + Netwrok reality: ip address change ad hosts move, other stuff
* Middleboses are intermediaries
  + Interposed in between communicating parties often without either knowing
  + Lots of uses
* NAT