**CSC3002F Notes – Network Layer**

**16 March 2022**

Services and Protocols

* Transport segment from sending to receiving host
  + Sender encapsulates segments into datagrams; passes to link layer
  + Receiver delivers segments to transport layer protocol
* Network layer protocols are in every Internet device (hosts and routers – Transport layer only runs on hosts)
* Routers: examine header fields in all IP datagrams passing through it. Moves datagrams from input ports to output ports to transfer datagrams along end-to-end path
* Two key network-layer functions:
* Forwarding (router-local action): move packet from a router’s input link to appropriate router output link.

Takes place at very short timescales, thus typically implemented in hardware.

* Routing (network-wide process): determine end-to-end paths taken by packets from source to destination.

Takes place over much longer timescales than forwarding and often implemented in software.

* Provides “best effort” service. No guarantees on:
  + Successful datagram delivery to destination
  + Timing or order of delivery
  + Bandwidth available to end-to-end flow
* Data Plane:
  + Local, per-router function
  + Determines how datagram arriving on router input port is forwarded to router output port
* Control Plane:
  + Network-wide logic
  + Determines how datagram is routed among routers along end-to-end path from source host to destination host
  + Two control plane approaches:
    - Traditional routing algorithms: implemented in routers. Individual routing algorithm components in each and every router interact in the control plane.
    - Software-defined networking (SDN): implemented in remote servers. Remote controller (on server) computes, installs forwarding tables in routers
* Destination-based forwarding:
  + Each host has an address; each packet has the IP address of destination in its header.
  + Router looks at IP address and decides where the packet must go.
  + Use a range of IP addresses in the forwarding table. All the packets with destination IP address in this range go to the same output link. There are too many IP addresses to store all of them.
  + Longest prefix matching: when looking for a forwarding table entry for given destination address, use the **longest** address prefix that matches the destination address
* Longest prefix match:
  + When looking for forwarding table entry for given destination address, use **longest** address prefix that matches destination address
* IP (Internet Protocol) Datagram format: **(see slides for diagram)**
  + Version (so it knows how to interpret packet); header length (useful because size of header is variable); type of service (generally not used on the general internet, but can, for example, indicate priority of particular packets); length (total datagram length)
  + 16-bit identifier; flags; fragment offset [if a packet is too big for some routers, it may be fragmented into smaller packets; then this row of the datagram is used for fragmentation and reassembly]
  + Time to live (maximum hops remaining); upper layer (are we dealing with UCP or TCP); header checksum
  + Source IP address [32-bit]
  + Destination IP address [32-bit]
  + Options (if any)
  + Payload data (variable length, typically a UDP or TCP segment)

[overhead: 20 bytes of TCP + 20 bytes of IP = 40 bytes + application layer overhead for TCP and IP]

IP addressing

* IP address: 32-bit identifier associated with each host or router interface
* Interface: connection between host/router and physical link
  + Router’s typically have multiple interfaces
  + Host typically has one or two interfaces
* Subnets: device interfaces that can physically reach each other without passing through an intervening router
* IP addresses have structure:
  + Subnet part: devices in same subnet have common high order bits
  + Host part: remaining low order bits
* Recipe for defining subnets:
  + Detach each interface from its host or router, creating “islands” of isolated networks; each isolated network is a subnet
* Routers may have multiple interfaces, all with different IP addresses.
* Classless InterDomain Routing (CITD):
  + Subnet portion of address of arbitrary length
  + Address format: a.b.c.d/x, where x is number of bits in subnet portion of address
  + An organization is typically assigned a block of continuous addresses, i.e. a range of addresses with the same common prefix (size, in bits, of the prefix is specified by x)
* How does a **host** get each IP address within its network?
  + Hard coded by sysadmin in config
  + Dynamic Host Configuration Protocol (DHCP): dynamically get address from a server
    - Can renew its lease on address in use
    - Allows reuse of addresses (only holds address while connected/on)
    - Support for mobile users who join/leave network
* DHCP client-server scenario:

1. Client: broadcast – is there a DHCP server out there? [DHCP discover]
2. DHCP Server: broadcast – I’m a DHCP server. Here’s an IP address you can use. Has to be a broadcast because the client doesn’t have an IP address yet. [DHCP offer]

* Two steps above can be skipped if a client remembers and wishes to reuse a previously allocated network address.

1. Client: broadcast – OK. I would like to use this IP address [DHCP request]
2. DHCP Server: broadcast (because there is a chance that the client hasn’t completed its configuration yet) – OK. You’ve got that IP address. [DHCP ACK]

* All these messages are broadcasts, but they contain a specific transaction ID (this ensures that no two clients are allocated the same IP address)
* Hierarchical addressing:
  + Allows efficient advertisement of routing information
  + **(See slides/textbook for diagram & explanation)**
* How does an ISP get block of address?
  + ICANN (Internet Corporation for Assigned Names and Numbers): allocates IP addresses, through 5 regional registries (RRs)
* Are there enough 32-bit IP addresses? No.
  + ICANN allocated last chunk of IPv4 to RRs in 2011
  + IPv6 has 128-bit address space
  + NAT helps IPv4 address space exhaustion
* Network Address Translation (NAT):
  + All devices in local network share just one IPv4 address as far as the outside world is concerned
  + All devices in local network have 32-bit addresses in a “private” IP address space (10/8, 172.16/12. 192.168/16 prefixes) that can only be used in local network
  + All datagrams leaving the local network have the *same* source NAT IP address (the IP address of the local router), but *different* source port numbers
  + Advantages:
    - Just one IP address needed from provider ISP for all devices
    - Can change addresses of host in local network without notifying outside world
    - Can change ISP without changing addresses of devices in local network
    - Security: devices inside local net not directly addressable or visible to the outside world
* NAT implementation:
  + Outgoing datagrams: replace (source IP address, source port number) of every outgoing datagram to (NAT IP address, new port number)
  + Remember (in NAT translation table) every (source IP address, port number) to (NAT IP address, new port number) translation pair
  + Incoming datagrams: replace (NAT IP address, new port number) in destination fields of every incoming datagram with corresponding (source IP address, port number) stored in NAT table
* NAT has been controversial: (but NAT is here to stay – extensively used in home and institutional networks, 4G/5G cellular networks)
  + Routers “should” only process up to layer 3 (port number should not be handled by any of these layers)
  + Shortage of IP addresses should be solved by IPv6
  + Violates end-to-end argument (port number manipulation by network-layer device)
  + NAT traversal: what if client wants to connect to server behind NAT?
* IPv6: Motivation:
  + Initial motivation: 32-bit IPv4 address space would be completely allocated
  + Speed processing/forwarding: 40-byte fixed length header (fixed length speeds up processing)
  + Enable different network-layer treatment of “flows”
* IPv6 Datagram Format
  + ver; pri (identify priority among datagrams in flow); flow label (identify datagrams in same “flow”)
  + payload len; next hdr (“next header” – allows us to append another header to indicate we have some extra information that we want to be used in processing); hop limit
  + source address (128 bits)
  + destination address (128 bits)
  + payload
* Items not in IPv4 (but in IPv6):
  + No checksum (to speed up processing at routers)
  + No fragmentation/reassembly (source must determine maximum packet size and send just that)
  + No options (use next hdr instead)
* IPv6 Addressing Rules:
  + 128 bits long: four times as long as IPv4
  + 2^128 addresses
  + Colon hexadecimal notation
    - Addresses are written using 32 hexadecimal digits
    - Digits are arranged into 8 groups of four to improve readability
    - Groups are separated by colons
  + Note: DNS plays an important role in the IPv6 world
    - Manual typing of IPv6 addresses is not an easy thing
    - Some zero suppression rules are allowed to lighten this task at least a little
* IPv6 Prefixes:
  + The prefix is the address that indicates the bits that have fixed values or are the bits of the subnet prefix
  + Prefixes for IPv6 subnets/routes/address ranges are expressed in the same way as CIDR notation for IPv4
* Transition from IPv4 to IPv6:
  + Not all routers can be upgraded simultaneously. (How will network operate with mixed IPv4 and IPv6 routers?)
  + Tunneling: IPv6 datagram carried as payload (data) inside IPv4 datagram among IPv4 routers (packet within a packet)
    - Tunneling used extensively in other contexts (4G/5G)
    - Lots of overhead due to duplicate headers
    - **(See slides/textbook for diagram showing how datagram is passed from IPv6 router to IPv4 router)**
* Generalized Forwarding:
  + Reminder: each router contains a forwarding table
  + Reminder: “match plus action” – match bits in arriving packet and take an action
  + Reminder: destination-based forwarding – forward based on destination IP address
  + Generalized forwarding:
    - Many header fields can determine action
    - Many actions are possible (drop/copy/modify/log packet)
  + Flow table abstraction (instead of forwarding table):
    - Flow: defined by header field values (in link/network/transport layer fields)
    - Match: pattern values in packet header fields
    - Actions: for matched packet – drop/forward/modify matched packet or send matched packet to controller
    - Priority: disambiguate overlapping patterns
    - Counters: number of bytes and number of packets
* OpenFlow (examples):
  + Destination-based forwarding – IP datagrams destined to IP address 51.6.0.8 should be forwarded to router output port 6
  + Firewall – block (do not forward) all datagrams destined to TCP port 22 (ssh port number); block all datagrams sent by host 128.119.1.1
* OpenFlow abstraction:
  + Router:
    - Match: longest destination IP prefix
    - Action: forward out a link
  + Firewall:
    - Match: IP addresses and TCP/UDP port numbers
    - Action: permit/deny
  + Switch:
    - Match: destination MAC address
    - Action: forward/flood
  + NAT:
    - Match: IP address and port
    - Action: rewrite address and port

Routing Algorithms (Control Plane):

* Note: cost of path from x to y is the same as the cost from y to x
* Routing algorithm classification:
  + Global vs Decentralized:
    - Global: all routers have complete topology (knowledge of the network), link cost info, and how to reach every other router
    - Decentralized: each router knows physically-connected neighbors and link cost to neighbors (iterative process of computation – exchange of info with neighbors)
  + Static vs dynamic:
    - Static: routes change slowly over time
    - Dynamic: routes change more quickly (periodic update in response to link cost changes)
* Notation:
  + c(x, y) = link cost from node x to y (=infinity if not direct neighbors)
  + D(v) = current value of cost of path from source to destination v
  + p(v) = predecessor node along path from source to v
  + N’ = set of nodes whose least cost path is definitively known
* Dijkstra’s Algorithm (a link-state routing algorithm):
  + Net topology and link costs known to all nodes (accomplished via link state broadcast; all nodes have the same knowledge)
  + Computes least cost paths from one node (source) to all other nodes. This gives the forwarding table for that node.
  + Iterative: after k iterations, the least cost path to k destinations is known
  + **(See slides/textbook for example)**
* Bellman-Ford (Distance vector algorithm):
  + Key idea: from time to time each node sends its own distance vector estimate to neighbors. When x receives new DV estimate from neighbor, it updates its own DV using BF-equation:

Dx(y) = cost of least-cost path from x to y = minv {c(x,v) + dv(y)}

(The minimum is taken over all neighbors v of x)

* + **(See slides/textbook for example)**
* Hierarchical Routing:
  + Our routing study thus far has been an idealization: all routers identical, network “flat”. This is not true in practice.
  + Scale:
    - With 600 million destinations we can’t store all destinations in routing tables.
    - Routing table exchange would swamp links.
  + Solution: organize the internet into **administrative zones**:
    - Each network admin manages routing inside that particular zone (example of a zone is an ISP).
  + Routers are aggregated into regions called “autonomous systems” (AS)
  + Routers in the same AS run the same routing protocol
    - “intra-AS” routing protocol
    - Routers in different AS can run different intra-AS routing protocol
  + Gateway router:
    - Routers that link an AS to the other ASes
    - Only the gateway router/s are visible to external routers
* Interconnected ASes:
  + - Forwarding table is configured by both intra-AS and inter-AS routing algorithm
    - Intra-AS sets entries for internal destinations
    - Inter-AS & intra-AS sets entries for external destinations
  + Inter-AS Tasks:
    - Suppose we have 3 ASes. Suppose a router in AS1 receives a datagram destined outside of AS1. Router should forward packet to gateway, but which one?
      * AS1 must learn which destinations are reachable through AS2 and AS3, then propagate this information on to all routers in AS1
    - Hot potato routing: send packet towards closest of two routers
    - **(see slides/textbook for example)**
  + Intra-AS Routing (also known as “Interior Gateway Protocols (IGO)”):
    - Most common intra-AS routing protocols:
      * Routing Information Protocol (RIP)
      * Open Shortest Path First (OSPF)
      * Interior Gateway Routing Protocol (IGRP)
* OSPF:
  + “Open” – publicly available
  + Uses link state algorithm (topology map at each node; route computation using Dijkstra)
  + OSPF advertisement carries one entry per neighbor
  + Advertisements flooded to *entire* AS (carried in OSPF messages directly over IP)
* Hierarchical OSPF:
  + Large networks broken into smaller routing zones
  + Each area has an area border router that is connected to the backbone (area border routers are part of the backbone AND smaller area)
  + Two-level hierarchy: local area & backbone
    - Link-state advertisements only in area
    - Each node has detailed area topology
  + Area border routers: summarize distances to nets in own area; advertise other area border routers
  + Backbone routers: run OSPF routing limited to backbone
  + Boundary routers: connect to other ASes
* BGP (Border Gateway Protocol):
  + Provides each AS a means to:
    - Obtain subnet reachability information from neighboring ASes (eBGP)
    - Propagate reachability information to all AS-internal routers (iBGP)
    - Determine good routes to other networks based on reachability information and policy
* Allows subnet to advertise its existence to the rest of the Internet
* BGP Basics:
  + BGP Session – two BGP routers (“peers”) exchange BGP messages (eg. prefix announcements, link updates)
* BGP Routing Policy: routers can be **provider** networks or **customer** networks. Some routers do not advertise to other routers if they would be routing to non-customers (i.e. only advertise routes to/from its customers; they waste resources and get nothing in return for routing non-customers)
* Why do we use different Intra-/Inter-AS routing?
  + Policy:
    - Inter-AS – admin wants control over how its traffic is routed and who routes through the network
    - Intra-AS – single admin, so no policy decisions needed
  + Scale:
    - Hierarchical routing saves table size and reduces update traffic
  + Performance:
    - Intra-AS – can focus on performance
    - Inter-AS – policy may dominate performance