P2P Networks and Software-Defined Networking

CS640, 2015-04-21

Announcements

- Assignment #4 due tonight at 11pm

Outline

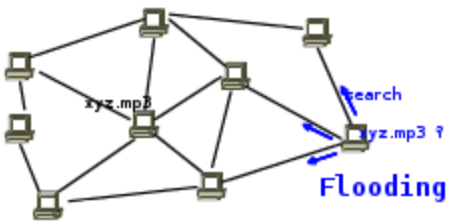
- Peer-to-peer Networks
- Limitations of traditional networks
- Software-defined networking
- OpenFlow

Content Distribution Networks

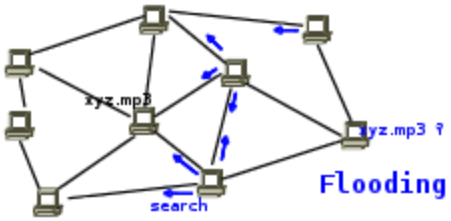
- Who uses CDNs?
    - Netflix
        - Login page & movie selection pages come from a VM in EC2 -- easter egg! bottom of Netflix page contains an ID for the VM in EC2 that provided page
        - Movie is delivered from CDN nodes
    - Pinterest -- study this in assignment #4

Peer-to-Peer Networks

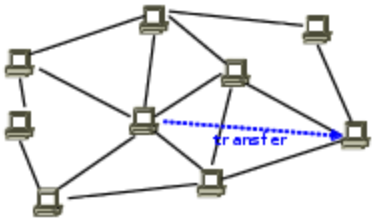
- Problem with CDNs: someone must build the infrastructure and others must pay to use it
- What if hosts wanting content could also serve content?
- Peer-to-peer initially became well known for its role in illegal music sharing -- e.g., Napster and KaZaA
- Other peer-to-peer services tried to encourage legitimate file sharing -- e.g., Gnutella and BitTorrent
    - Obviously, there is still illegal file sharing
- ***How do we locate content?*
    - Flood a query -- Gnutella
    - Lookup in a central directory -- BitTorrent uses a .torrent file with information about a tracker that knows all peers who are requesting or providing the file
- ***How do we deliver content?*
    - From a single peer -- Gnutella
    - From multiple peers -- BitTorrent
- Simple peer-to-peer network construction and data retrieval -- Gnutella
    - Peer join
        - Connect to a peer it knows
        - That peer shares the IP address of its neighbors
        - New peer connects to those neighbors
    - Locate content
        - Send query (with name of file, query ID, and TTL) to neighbor peers



        - Neighbor stores query ID, decrements TTL, and floods query to its neighbors



        - If peer sees a query with the same ID, drop it to avoid forwarding loops
        - If peer has the content, send reply back to the neighbor that forwarded the query
        - Other peers lookup query ID in reply to know where to send the reply
        - Connect directly to peer with the content



- Getting data from multiple peers -- BitTorrent
    - Chunking
        - Important for large files
        - Divide file into many chunks
        - Replicate chunks on different peers
        - Peers “exchange” chunks with each other
            - Peer is both uploading and downloading chunks
            - If a peer is uploading at a slower rate, it is also forced to download at a slower rate -- provides some fairness
        - Peer request chunks from other peers in a random order
        - Peer can (hopefully) assemble the entire file
    - Tracker
        - Infrastructure node that keeps track of peers uploading/downloading parts of a file
        - Swarm -- set of active peers for a given piece of content
        - Peers register with tracker (determined by torrent file) when they want to up/download
        - Tracker provides a random set of peers to contact to attempt to get chunks
        - Can request more peers from tracker if no provided peers have the chunks you need

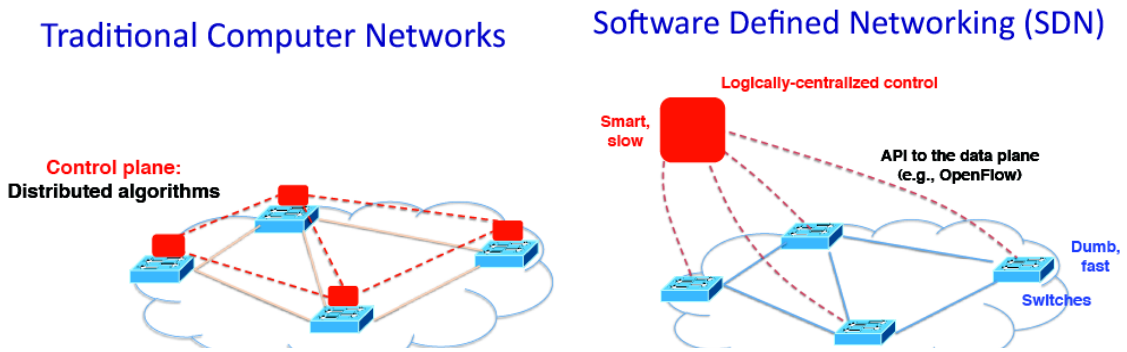
Limitations of Traditional Networks

- What are the two planes that exist within a switch (or router)?
    - Data plane -- forwards packets based on routing table
    - Control plane -- establishes routing table using routing algorithms
- Routing tables are established using distributed algorithms
    - Decisions are based on local view -- DV and BGP decisions depend on decisions of neighbors
    - ***What problems does this introduce?*
        - Different nodes may have a different view -- due to updates that have not yet propagated
        - Decisions may not be optimal -- e.g., BGP finds **a** path; may not be **an optimal** path
        - Debugging is difficult

- Network-wide policy is implemented using distributed configs
    - Config files for each switch/router in vendor-specific config language (e.g., Cisco IOS)
    - Configs define: routing algorithms to use, link weights, access control lists (ACLs), etc.
    - ***What problems does this introduce?*
        - Need to re-write configs when you change vendors
        - Configs can easily become inconsistent with each other, and with an operator's goals
- Policies are largely defined over subnets
    - Layer 3 routing table has entries for individual subnets
    - Layer 2 MAC table has entries for individual hosts
    - ***What problems does this introduce?*
        - Cannot easily customize routing policies for different apps -- e.g., video streaming needs low jitter path, SSH needs low latency path, and file transfer needs high throughput path
- Software stack on switches are closed
    - Vendors write the software; we are limited to using the capabilities exposed by the switch
    - Takes a long time for vendors to adopt and implement new protocols and features
- Summary of problems
    - Distributed routing decisions -> difficult to debug; maybe suboptimal decisions
    - Distributed policy -> prone to human errors
    - Policies defined over subnets -> cannot customize forwarding per application
    - Closed software stack -> limited operator flexibility

Software Defined Networking (SDN)

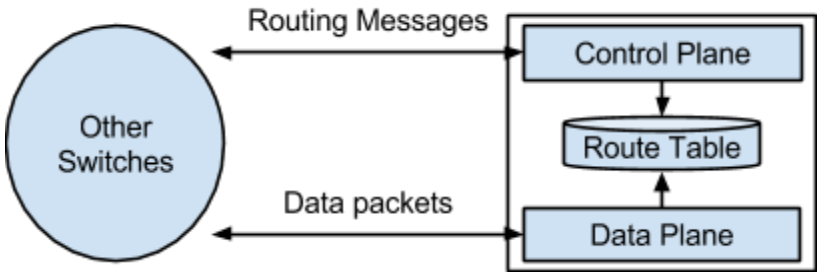
- Remove control plane from switches and run centralized controller
    - Data plane operates as before, but controller populates routing tables on switches
    - Controller interacts with switches using a standardized vendor-independent API
    - Network operators write custom control applications that run atop the controller and implement the desired forwarding policies -- i.e., determine which forwarding entries to install in each switch
- Pictorial view of traditional network versus SDN



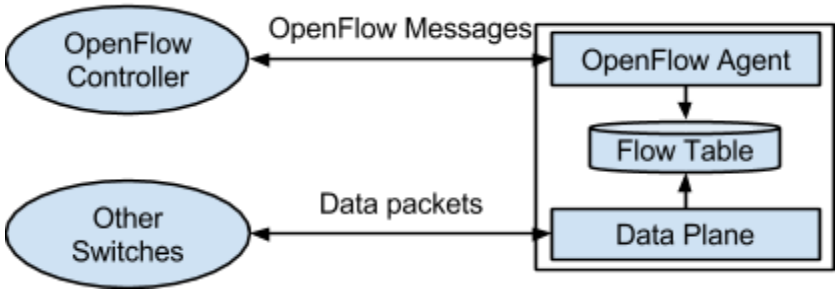
- Benefits
    - Forwarding decisions are centralized
        - Less prone to error; easier to debug -- in theory, at least
        - Decisions can be optimized -- controller has network-wide view
    - Forwarding decisions are flexible
        - Can implement fine-grained policies for individual applications
        - Not restricted to specific protocols/features implemented by vendors

OpenFlow

- Prevailing SDN standard
- Traditional switch vs. OpenFlow switch
    - Traditional switch



    - OpenFlow switch



- Traditional route table vs. Flow Table
    - Traditional route table

Subnet	**Mask**	**Gateway**	**Interface**

        - Only match on IP address in L3 router or MAC address in L2 switch
        - Only action is forward out interface (and drop if we have ACLs)
        - Entries are added/removed by the control plane based on routing algorithms
    - Flow table

Src MAC	**Dst MAC**	**Ether Type**	**Src IP**	**Dst IP**	**Proto**	**Src Port**	**Dst Port**	**Actions**

        - Match on “any” fields in link, network, and transport layer headers
            - Originally matched on 10 different fields; now match on any bytes in the headers
        - Also match based on physical switch port on which packet arrived
        - Actions
            - Forward to port(s)
            - Drop
            - Send to controller
            - Re-write header field(s)
        - Entry in the flow table is called a **rule**
        - Entries are added/removed by the controller
        - Rules can have timeouts
            - Soft -- switch removes rule if no packet matches the rule for some period of time
            - Hard -- switch removes rule after some time has passed since rule was added
        - Rules can have priorities
        - Switch also maintains statistics (packet and byte counters) for each rule

- Example flow table entries

Switching

Switch Port	MAC src	MAC dst	Eth type	VLAN ID	IP Src	IP Dst	IP Prot	TCP sport	TCP dport	Action
*	*	00:1f:..	*	*	*	*	*	*	*	port6

Flow Switching

Switch Port	MAC src	MAC dst	Eth type	VLAN ID	IP Src	IP Dst	IP Prot	TCP sport	TCP dport	Action
port3	00:20..	00:1f..	0800	vlan1	1.2.3.4	5.6.7.8	4	17264	80	port6

Firewall

Switch Port	MAC src	MAC dst	Eth type	VLAN ID	IP Src	IP Dst	IP Prot	TCP sport	TCP dport	Action
*	*	*	*	*	*	*	*	*	22	drop