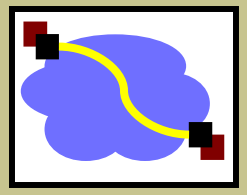


15-441 Computer Networking

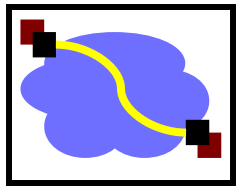
Lecture 9 –

ARP, RARP, BOOTP, DHCP

Layer 2 and 3 Switch Fabric

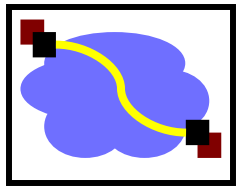
Distributed Agreement

Outline



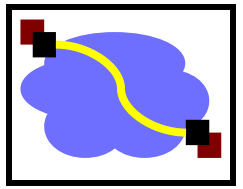
- ARP, RARP, BOOTP, DHCP
- Switch Fabric
- Distributed Agreement

Address Resolution Protocol (ARP) ...and friends



- ARP
  - Used to get Layer 2 address from Layer 3 address, e.g. by router.
  - Request, e.g. “Who has IP address X?” is *broadcast* as IP address is not known.
    - Can also be used as an announcement to periodically update caches
  - Reply is *unicast* to requestor.

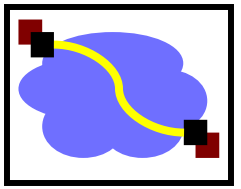
ARP Hacks



ARP is totally unauthenticated

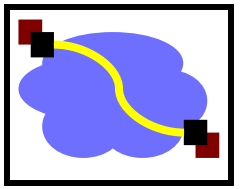
- Bogus message (intentionally) mismapping the router
  - Disconnects a large chunk of the network
- Bogus message directing a particular host to another host with IP Forwarding
  - Misdirect messages and enable IP Forwarding to form man-in- middle
- Flood the network with bogus ARPs
  - Switch tables become full of junk
  - Real messages get flooded to all interfaces
  - Enables sniffing across network segments

Reverse ARP (RARP)



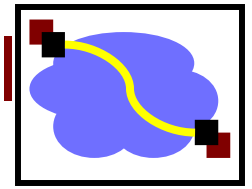
- Network broadcast to request IP address given MAC address
- Server replies via Layer 2 message, providing IP address
- Used as a configuration protocol
- Nominally replaced by BOOTP (and really by DHCP) to eliminate the dependency upon broadcast messages, which limited scope to a single network segment

Boot Protocol (BOOTP)



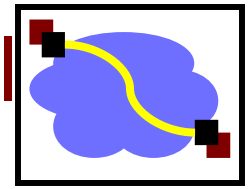
- UDP-based alternative to ARP
- Can carry more than just IP address
- Uses address to unicast via UDP if client knows it
  - Can cross network boundaries
- Sends broadcast message if client does not know IP
  - Not much different than RARP
- Mostly short-lived RARP replacement
  - World quickly moved on to DHCP
  - Still makes occasional appearances for simplicity and flexibility

Dynamic Host Configuration Protocol (DHCP)



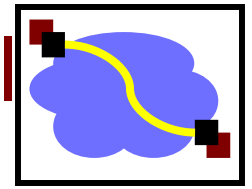
- Configuration information
  - IP Address, subnet, gateway, DNS, time server, etc.
  - IP address can be static or dynamic. Dynamic can prefer stickiness.
  - Different servers can manage different pools, or the same pool if on the same network
- DORA (Discovery, Offer, Request, Acknowledgment)
- “Discovery” Request via UDP broadcast
  - Can include old/expired IP address to request it
  - Relay's can proxy for local client to cross network boundaries for initial request,before IP is assigned
- “Offer” Reply via link-layer unicast
  - IP address of server, so future messages can be unicast
  - Multiple servers can exist, so multiple offers can be received
  - Offered IP address
  - Subnet mask
  - Lease length

Dynamic Host Configuration Protocol (DHCP), *cont.*



- "Request" accepts offer
  - This is **multi**cast. This lets all servers know, so unused offers can be canceled And, is pedantically correct, since the assignment is not made until request received and ACKed.
- "Acknowledgment"
  - Confirms assignment
  - Provides lease duration
  - Provides other configuration information
- Renewals
  - Start requesting at ½ lease duration
  - Sent via UDP **uni**cast to grating server
  - Expires and returned if not renewed

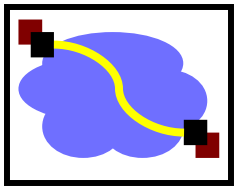
Dynamic Host Configuration Protocol (DHCP), *cont, cont.*



- Things done right
  - Multiple servers for reliability
  - Synchronization is cheap (via broadcast)
  - Unicast, once able, reduces traffic
  - Agents to extend beyond network boundaries
  - **Leases instead of grants**

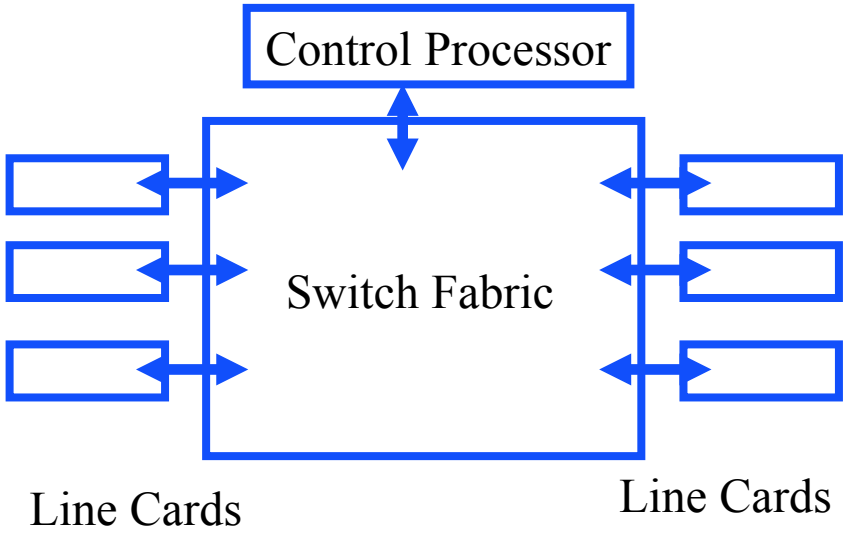
- Lessons not learned: Authentication, anyone?
  - Imposter DHCP servers deny or redirect services
  - Imposter clients gain access
  - DoS by claiming all available IPs

Outline



- ARP, RARP, BOOTP, DHCP
- Switch Fabric
- Distributed Agreement

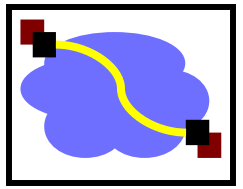
Revisiting Switch Internals: Earlier Picture



- Switches
  - circuit switch
  - Ethernet switch
  - ATM switch
  - IP router

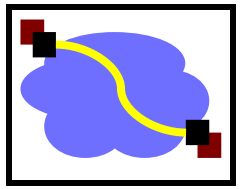
- Switch fabric
  - high capacity interconnect
- Line card
  - address lookup in the data path (forwarding)
- Control Processor
  - load the forwarding table (routing or signaling)

Revisiting Switch Internals Input Port to Output Port?



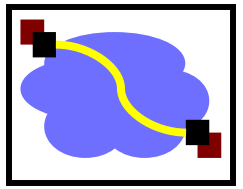
- Ports can be used for input and/or output
- Input and output aren't instantaneous
  - Messages arrive over time and are buffered until complete and verified
- Deciding what to do with a message at an input port takes time
  - More time for complex “routing” than simple “switching”
- It takes machinery to get from an input port to an output port
  - If aspects of this machinery are busy, time may be spent queued/buffered
- The output port might be busy, which may result in queuing/buffering
- (Any queue prior to the input port would need to be at the sender)
- Input and output can operate at different rates, e.g. 1Gbps input ports and 100Gbps output ports, or the reverse.

Revisiting Switch Internals
What is the *Fabric?*



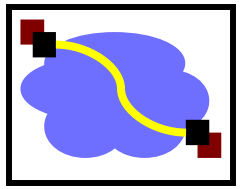
- The *Fabric* is everything involved in connecting a message arriving at an input port with its destination output port.
  - *Queues* provide temporary storage during while waiting
  - *Buffers* provide temporary storage while acted upon
  - A *Switch* or *Bus* provides the connectivity from input to output
  - The *Control Plane* is responsible for complex decision making, e.g. thinking. Essentially, it populates forwarding tables
    - The control play is traditionally involved in layer-3 functions, e.g. maintenance of routing and forwarding tables, multicast, IP options processing, etc.
  - The *Data Plane* is responsible for simple decision making, e.g. table lookups. It basically does the "Look and Go" based on what the control plane set up.
    - The data plane did not traditionally handle layer-3 packet processing, but modern switches can, in situations without complex IP options, handle a variety of layer-2 functions in the data plane.

Revisiting Switch Internals Old School



- Routers and switches were basically computers
  - Global buffers shared by all NICs
  - Layer-2 used a global table of MAC addresses
  - Layer-3 used a global table of destination addresses
  - Simple process
    - Read into global buffer
    - Queue buffer for processing
    - Make decision about output port, and do any necessary updating
    - Queue for output port
    - Drain via port onto network
  - **Memory throughput is limiting**
    - **Copy from line card, to memory**
    - **Copy from memory to line card**
    - **Table lookups**

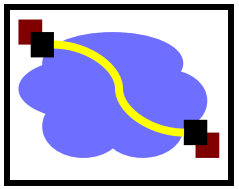
Revisiting Switch Internals Simple Fixes



- Cache tables on line cards
  - Local lookups
  - Only needs to interact with main memory for updates
  - Memory throughput/bus contention no longer an issue
- Memory bypass
  - Use bus to move from line card to line card without buffering in main memory

- Obviously, this only works for the data plane “switching”, whether layer-2 or layer-3
  - Complex decision making still passes through the control plane, which means the CPU, main memory, etc.
- Memory might not be a bottleneck – but the bus still is.

Revisiting Switch Internals
Bye, bye bus

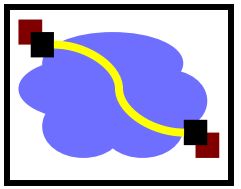


- Since the shared bus is a bottleneck, we need to address it
  - We can make it faster – but scale is not in our favor
  - More ports = more opportunities for parallel input
  - Battling parallelism with speed is rarely going to win at scale.
  - Battling parallelism with parallelism is more likely to win.

- Want a solution with parallelism
  - Traffic shows up at multiple ports at the same time

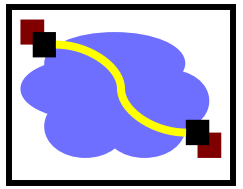
- Queuing is still needed
  - Multiple inputs may need to go to the same output.
  - Outputs may be slower than inputs
  - Architectural artifacts, e.g. head of line blocking

Revisiting Switch Internals How to get parallelism?

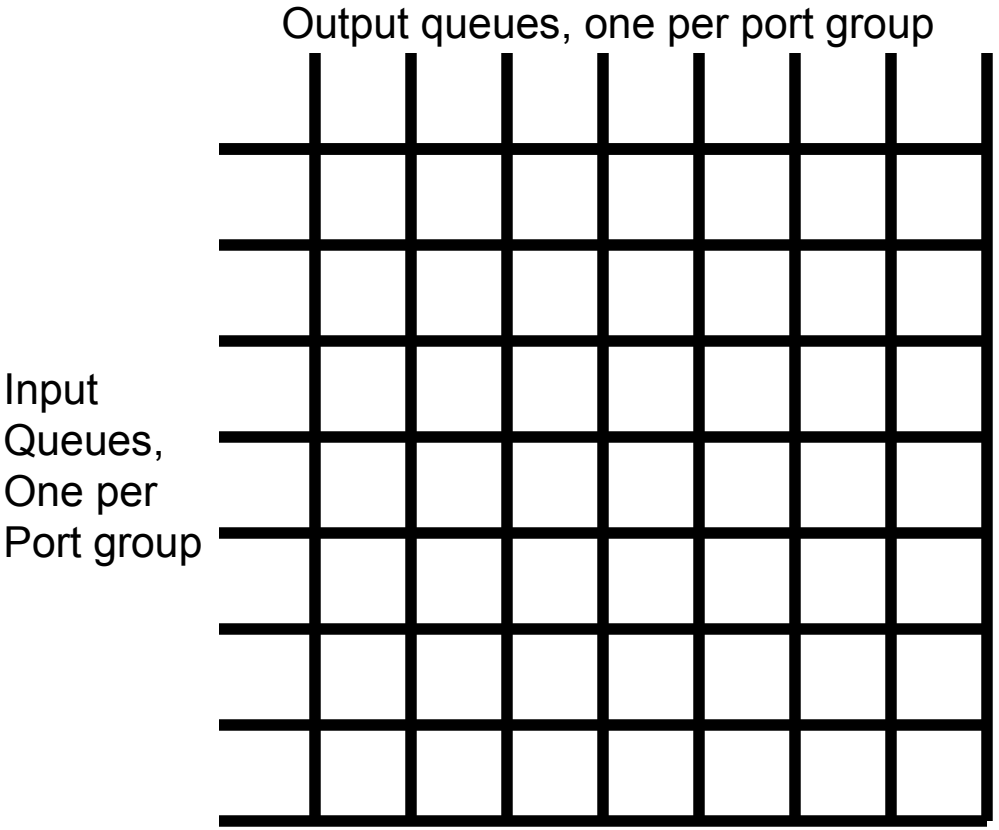


- Replace single bus with parallel switch
  - Classic solution is a crossbar, other architectures exist
- Basic process
  - Work arrives at input ports and is queued
  - Switch is configured
  - Work crosses switch and is queued at output ports to be drained to network as able
  - Repeat
- Open questions
  - How are buffers managed? Where do they live?
  - How do ports map to switch connections/channels?

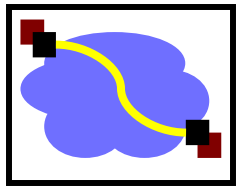
Revisiting Switch Internals Crossbar Example



Although it would be nice to have two crossbar channels/connection per port, one for input and one for output, the n^2 nature makes this expensive. Commonly ports are grouped together, as on a line card, and contend within a group for a channel through the crossbar. In any case, input and output buffering are required.

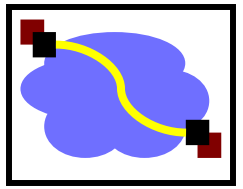


Revisiting Switch Internals
Where to queue?



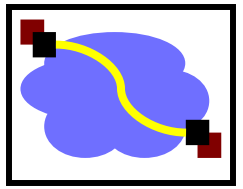
- It would be nice if internal and external capacity, including decision making, were plentiful and queuing were unnecessary
  - {insert reality here}
- Given that we may need to wait for decision making and/or an output port, queuing is necessary.
  - And, beyond that, it can also be generated as an artifact of design
- Queuing strategies trade off scheduling complexity (decision making), memory pressure, and I/O

Revisiting Switch Internals
Queuing models



- Input queued
  - Queue messages at input port until can be copied to the output port's buffer, FIFO
- Output queued
  - Initial placement in shared memory queue, followed by global scheduling and local dispatch
- Combined Input-Output Queued (CIOQ)
  - Input queues + output queues
- CIOQ + Virtual Output Queues (VoQs)
  - Local to each input queue exists a shadow secondary queue for each output port

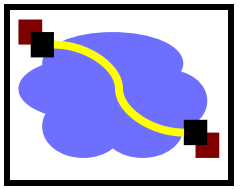
Revisiting Switch Internals
Input queued



- Input queued
  - This is a relatively simple model.
  - Queue messages at input port until can be copied to the output port's
  - buffer

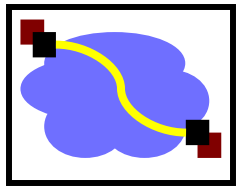
  - The head of the queue is examined and the output port is determined
  - It is written to the output port's buffer, when able
    - The output port may be busy
    - Another input port may be contending for the output port
    - Etc
  - Processing is FIFO, so Head of Line (HoL) blocking can occur, preventing other queued messages from being dispatched to available output ports

Revisiting Switch Internals
Output queued



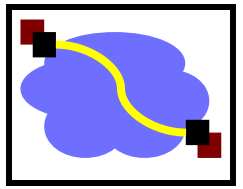
- Output queued, Model Idea
  - The model idea is to buffer at the input port, but to queue directly for the output port
  - This would tie up the input port, preventing the arrival of new packets
    - Until a decision is made
    - And, until contention for the output queue is resolved
    - And, until the copy is complete, but that isn't new

Revisiting Switch Internals
Output queued, *cont.*



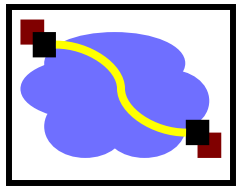
- Output queued, In reality
  - Initial placement in shared memory
  - Global scheduler maps input buffer to output queue(s)
  - Output port scheduler drains its output queue to its output buffer, recycling global memory
- Primary challenges
  - Making the access to global shared memory fast enough to handle all of the inputs
  - Making it fast enough, given contention resulting from concurrent queue access
- Historically, not favored
  - Shared memory is hard
- Future unclear
  - SoC, etc, is making shared memory easier
  - Demand for speed is unrelenting
  - Fewer memory copies

Revisiting Switch Internals
Combined Input-Output Queued (CIOQ)



- Given the challenges of shared memory, separate queues are more common
- Initial Idea: Have Input queues in addition to the Output queues
  - Make decisions on input queue
  - Move to output queue, when able
- Reality: Can't always immediately get to output queue
  - Internal bandwidth limitations
  - Blocking necessitated by contention
- Impact:
  - Head of Line (HOL) blocking, again.

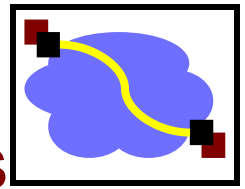
Revisiting Switch Internals
Virtual Output Queues (VoQs)



- Solution: *Virtual Output Queues (VoQs)*
  - Have separate queues for each output port at each input port.
  - No more head of line blocking – each output queues path is separate

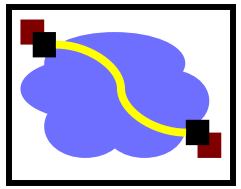
- Nuances
  - Each may be queued twice, once at input and once at output
  - Tons of queues

Revisiting Switch Internals
Lest We Forget: Overarching Considerations



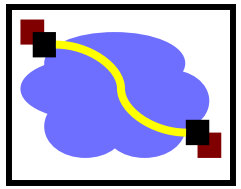
- How to connect
  - Common bus
  - Parallel switch, e.g. crossbar, banyon switch, etc
- Memory architecture
  - Global shared?
  - Per port or port group?
  - Two of above? Three of above?
- Where and how to buffer
  - Input port, output port, both?
  - How to partition? Fixed? Agile?
- Data structure(s) and algorithm(s) for scheduling
  - How to maintain table data structure
  - Algorithm to perform mapping
  - May be distinct global (input side) and local (output side) processes
  - Traffic is not uniform, efficiently selecting from inputs is hard

Revisiting Switch Internals
Random Lingo



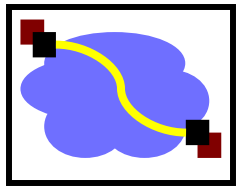
- Arbitration
  - Ensuring that there is not concurrent access to a shared resource (that can't be concurrently used)
  - e.g., Arbitrating access to a VoC
- Scheduling
  - Given a resource and a bunch of work for it, deciding which work should get done when
    - e.g., given a whole bunch of messages we know are destined for an output port, deciding in which order they should be drained

Revisiting Switch Internals Surprise!



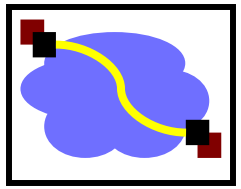
- Energy is
  - A huge concern in switch design and data center management
  - Expensive
  - Not always available
  - Hard to remove from the chassis in the form of transfer and cooling
  - Expensive to remove from chassis in the form of cooling, where possible
- What consumes energy?
  - Fabric is most energy dense
  - But, line cards dominate by shear number.
- Mitigation?
  - Bigger switches to amortize fabric cost.
  - Moore's law still seems to apply to processing ability within switches
    - Outpaces traffic growth in many applications

Outline



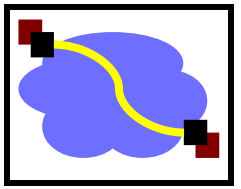
- ARP, RARP, BOOTP, DHCP
- Switch Fabric
- Distributed Agreement

Routing Is A Hard Problem Distributed Agreement



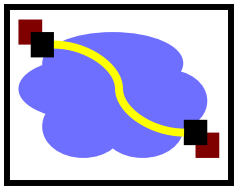
- Maintaining Routing Tables Is Intrinsically a hard problem
  - Distributed Agreement in a dynamic system
- The goal
  - Routers should communicate with each other and maintain a global understanding of the topology of the network, such that they can each, individually, make globally good forwarding decisions.
- No magic
  - Routers must communicate via the same network connections as everyone else
  - These are the same network links that can be too slow, too congested, too lossy, or just plain failed.
  - The state of the network can change at any time, without warning
  - It takes time to communicate

Routing Is A Hard Problem
Convergence



- Convergence versus absolute agreement
  - Once a change happens, it take time to communicate
  - Once a change happens, it take time to figure out what to do about it
  - Participants continue to (otherwise) function, even when, as a result of these changes, agreement is not perfect
  - Participants *converge toward* agreement – whether or not agreement has actually been reached is never known to any participant

Routing Is A Hard Problem Examples



- Unseen Loops
- False Partitioning
- Stale Information Looks New
- Messages Travel Forever