Review: Clark's Paper + the Final Exam

CS168

Sylvia Ratnasamy

Fall 2024

Outline

- **Discuss Clark'88: The Design Philosophy of the DARPA Internet Protocols**
- **Review for the final exam**

Clark'88 Context

- David D. Clark (MIT): Chief Protocol Architect for the Internet from 1981-89

- At the time of writing …
  - 1 year after Cisco's 1st product, IETF started
  - Number of hosts reaches 10,000
  - NSFNET backbone 1 year old; 1.5Mb/s

General impressions

- What did you think of the paper?
  - Important? Surprising? Snooze?

- My take: important because it provides context and clarity
- Plus a few tips for success
  - Be clear about your goals
  - Have a concrete/grounded use-case
    - E.g., "… give users on the packet radio network access to the machines on the ARPANET."
  - Learn by building, iterating ("engineering attitude")
    - E.g., separation of TCP/IP
  - Attention to detail
    - E.g., discussion of `End of Letter' flag

DDC'88

Goal 0: An "effective" technique for multiplexed utilization of existing interconnected networks

Goal 1: Communication must continue despite loss of networks or gateways

Goal 2: Must support multiple types of communication service

Goal 3: Must accommodate a variety of networks [underneath]

Goal 4: Must permit distributed management of its resources

Goal 5: Must be cost effective

Goal 6: Must permit host attachment with a low level of effort

Goal 7: The resources used in the Internet architecture must be accountable

Goal 0: An effective technique for multiplexed utilization of existing interconnected networks

- **Multiplexing (sharing)**
  - Shared use of a single communication infrastructure
- **Existing networks (interconnection)**
  - Tries to define an “easy” set of requirements for the underlying networks to support as many as possible

- How: different networks connected by packet switched, store-and-forward routers/gateways

Goal 1: Internet communication must continue despite loss of networks or gateways.

"Entities should be able to continue communicating without having to reestablish the high level state of their conversation"

"The architecture [should] mask any transient failure."

Leads to:

1. "Fate-sharing"

"Fate Sharing"

- Basic idea:
  - Communication shouldn't be disrupted by the failure of a particular router/node in the network; two endpoints should be able to communicate if <u>some</u> path exists between them
  - Leads to the decision to keep communication state (e.g., which pkts have been transmitted) at the endpoints, not in routers
  - Now, if state is lost it's because the endpoint failed so it doesn't matter! I.e., an endpoint and its state "share the same fate"

Goal 1: Internet communication must continue despite loss of networks or gateways.

"Entities should be able to continue communicating without having to reestablish the high level state of their conversation"

"The architecture [should] mask any transient failure."

Leads to:

1. "Fate-sharing"
2. Stateless packet switches → "datagrams"

Goal 2: Support multiple types of service

"Different services distinguished by differing requirements for such things as speed, latency, and reliability"

Leads to: separation of TCP from IP

Goal 3: Support varieties of networks

"[Very important that the Internet] be able to incorporate and utilize a wide variety of network technologies"

Leads to: a minimum set of assumptions about the function the network will provide

1. "network can transport a packet"
2. "of reasonable size"
3. "delivered with reasonable reliability"

Other goals

Goal 4: The Internet architecture must permit distributed management of its resources

Q. Does it accomplish this?

Goal 5: The Internet architecture must be cost effective.

Q. Is it cost effective?

Goal 6: Low cost of attachment

Goal 7: The resources… must be accountable

Q. What does this mean?

Q. What would such a network look like?

Other (prescient) observations in the paper

"The most important change in the Internet…will probably be the development of a new generation of tools for management of resources…"

Recall: Rob's lecture and discussion of SDN, OpenConfig, etc.

"The relationship between architecture and performance is an extremely challenging one…"

- the goal of the architecture was […] to permit variability

Recall: Nandita's lectures on Google's efforts to tame tail latency in datacenter contexts

"There may be a better building block than the datagram …"

- identify a sequence of packets -- "flow"

Recall: Rob's lecture on OpenFlow

Outline

- Discuss Clark'88: The Design Philosophy of the DARPA Internet Protocols

- **Review for the final exam**

Coverage

- **Exam will emphasize material in lectures 15+** (i.e., not covered by midterm)
- **But lectures 1-14 are still in scope**
  - **Familiarity with key concepts from those lectures is required for later material**
- **Material from our upcoming guest lecture (lecture 27) will only be lightly tested at a conceptual level, in the form of true-false, simple multiple-choice**

Style

- Similar to the midterm and practice material

This Review

- Walk through what we expect you to know
  - Summarize – not explain -- key concepts and details

- If I didn't cover it, doesn't mean you don't need to know it
  - But if I covered it today, you should know it!

- Use this slide deck as a check list

Outline

- Review:
  - Wireless and Cellular (lectures 24, 25)
  - Host Networking (lectures 22, 23)
  - Datacenters and SDN (lectures 15, 16, 17)
  - How the pieces fit (lectures 18-21)
    - Not explicitly reviewing individual details of HTTP, DNS, Ethernet, DHCP, etc.
  - Pre-midterm material: concepts you should be familiar with

- Will go as far as time permits; entire slide deck will be available

Lectures 25: Cellular

- Mobility introduces some fundamental new challenges
  - Discovery, authentication, seamlessness, accounting
- Cellular infrastructure is composed of:
  - Radio-Access Network (RAN) and Core
  - RAN: antennas, radio transceivers, radio controller that assigns tower's radic resources to each user
  - Core: implements various control and data functions related to mobility
    - Mobility Manager, cellular DB, Radio GW, Packet GW
- Relevant identifiers: IMSI, IP addresses, IMEI, MSISDN
- 4 core operations: Discovery, attach, handovers, roaming
  - Know how each works to the level of detail discussed in lecture

Lectures 24: Intro to Wireless

- **Slides 1-28: only expect you to understand this slide at a high level** **(no equations or details)**

Wired vs Wireless: Some Crucial Differences

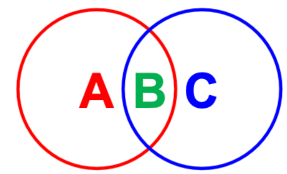
- Wireless is a fundamentally shared medium
  - Wired is not
- Wireless signals attenuate significantly with distance
  - Wired signals do not
- Wireless environments can change rapidly
  - Wired environments do not
- Wireless packet collisions are hard to detect
  - Wired packets collisions are not

CS 168, UC Berkeley: 6

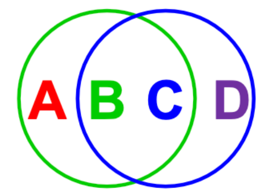
Lectures 24: Intro to Wireless

- **Slides 1-28: only expect you to understand this slide at a high level (no equations or details)**
- **Slides 28-49: do understand different media access approaches and their tradeoffs**
- **Slides 50+: will not be tested**

Lectures 24: Media Access



- **CSMA: listen and don't transmit if someone else is**
  - **Suffers from hidden and exposed terminals**
- **RTS/CTS: request-to-send / clear-to-send**
  - **Solves the hidden terminal**
  - **Problem: only partially helps with exposed terminals**
  - **Problem: RTS collisions (A and C simultaneously send an RTS to B)**



- **Hence, additional techniques (MACA/MACAW)**

For the exam:

- You don't need to know the specific rules of a particular protocol (MACA, MACAW, etc)
- Instead, we'll give you a protocol and you should be prepared to analyze its

Outline

- Review:
  - Wireless and Cellular (lectures 24, 25)
  - Host Networking (lectures 22, 23)
  - Datacenters and SDN (lectures 15, 16, 17)
  - How the pieces fit (lectures 18-21)
    - Not explicitly reviewing individual details of HTTP, DNS, Ethernet, DHCP, etc.
  - Pre-midterm material: concepts you should be familiar with

- Will go as far as time permits; entire slide deck will be available

Lectures 22-23: Host Networking (in Datacenters)

- **Host networking refers to the functions we implement at the host to support the abstraction of the network as a fast, reliable, secure, ordered byte stream**
- **Functions:**
  - **Loss recovery, congestion control, flow control (our TCP lectures)**
  - **+ load balancing, traffic shaping, QoS (know these)**
  - **+ BW allocation, security (only FYI, out of scope for exam)**
- **These functions address the new requirements that arise with DC workloads**
  - **Performance: high BW and low latency**
  - **Ease of development**
  - **CPU efficiency**
- **Traditional OS-based host networking makes it difficult to meet the above requirements**
  - **Kernel development is slow and painful**
  - **Memory copies between userspace and kernel hurts performance**
  - **CPU resources are consumed to implement the above functions**

Lecture 22: OS bypass and NIC offloads

- **Solution#1: "OS bypass" stacks → implement host networking functions in userspace**
  - **Addresses the problem of memory copies and kernel development but still consumes CPU resources**
- **Solution#2: NIC offload → run host networking functions on the NIC, freeing up CPU resources**
  - **Improvement in CPU efficiency depends on what functions we can offload**
  - **Three phases (and degrees) of NIC offload**
    - **Phase 1: simple stateless functions:**
      - **Examples: checksum, segmentation, tx/rx queue selection**
      - **Understand the what and why of these examples but not expected to know the "how" in any detail**
    - **Phase 2: simple stateful functions:**
      - **Example: match-action table to implement the network virtualization concept (introduced in Rob's SDN lecture)**

Lecture 22: RDMA

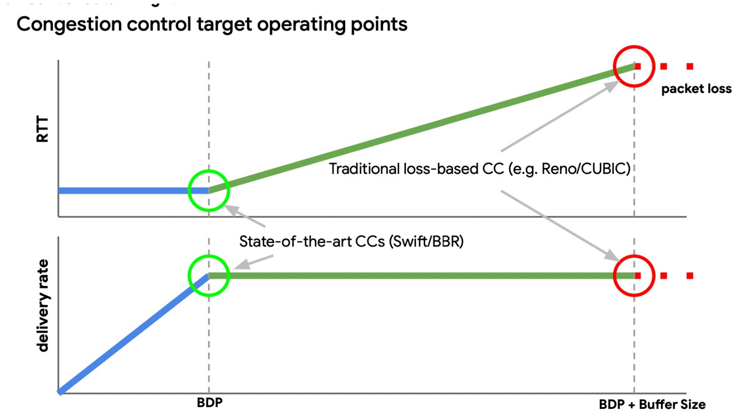
- RDMA → a new abstraction whereby a host A can efficiently read from (write to) the memory at a remote host B
- Efficient → consuming minimal CPU cycles at host A and host B

- How?
  - NICs directly read from (write to) host memory
  - NICs and network responsible for all aspects of host networking including data transmission, reliability, CC, etc.
  - CPU just initiates the transfer then gets out of the way

- Implemented via "queue-pair" abstraction on an RDMA NIC
  - CPU writes send/receive work queue elements (WQEs) to the NIC's queue-pair; WQEs point to memory buffers
  - NIC notifies CPU via completion queue elements (CQE)

Lectures 23: Advanced Host Networking Functions

- **We looked at how 3 advanced host networking functions are implemented in (Google) datacenters**
  - Delay-based CC (Swift), Protective Load-balancing (PLB), Traffic Shaping using timing wheels
- **Understand the problem each of the above i**
  - Swift → loss-based CC leads to high packet delay



Lectures 23: Advanced Host Networking Functions

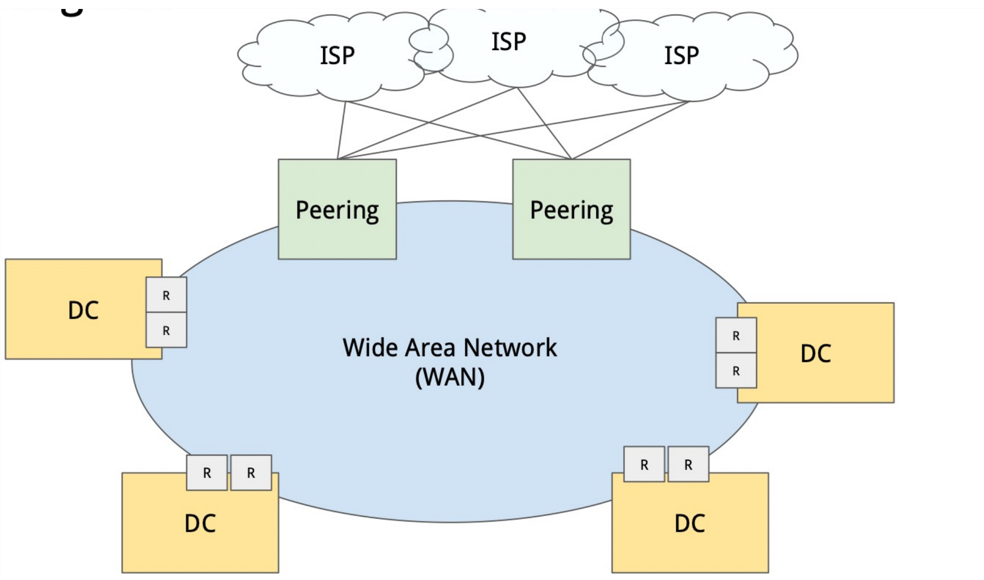
- **We looked at how 3 advanced host networking functions are implemented in (Google) datacenters**
  - Delay-based CC (Swift), Protective Load-balancing (PLB), Traffic Shaping using timing wheels
- **Understand the problem each of the above is solving**
  - Swift → loss-based CC leads to high packet delay
  - PLB → load-balancing with ECMP-based hashing is still imperfect
  - Timing wheels → need traffic shaping but implementing it with per-flow queues doesn't scale
- **Understand the essential idea behind each solution**
  - Swift → AIMD based on packet delay (out of scope: implementation details of Swift)
  - PLB → on congestion, change the "flow label" field in the IP header (changes the fields being hashed)

Outline

- Review:
    - Wireless and Cellular (lectures 24, 25)
    - Host Networking (lectures 22, 23)
    - Datacenters and SDN (lectures 15, 16, 17)
    - How the pieces fit (lectures 18-21)
        - Not explicitly reviewing individual details of HTTP, DNS, Ethernet, DHCP, etc.
    - Pre-midterm material: concepts you should be familiar with
- Will go as far as time permits; entire slide deck will be available

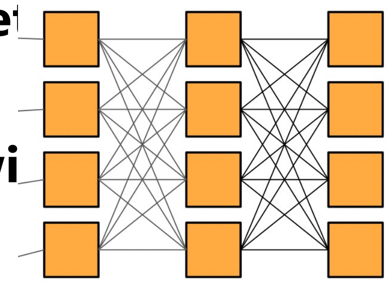
Lectures 15-16: Datacenters

- **DCs in the big picture**



Lectures 15-16: Datacenters

- DCs in the big picture
- Anatomy of a datacenter: servers in racks, top-of-rack switches, interconnected by a DC network ("fabric")
- Know how DC networks are different: homogeneous, single admin control, performance is top priority
- Large volume of "east-west" traffic means we need high bisection bandwidth networks
- Challenge: how do we build a high bisection bandwidth net le and cost-effective manner?
- Solution: Clos networks – interconnect smaller/cheaper swi bisection BW network



Lectures 15: CC in Datacenters

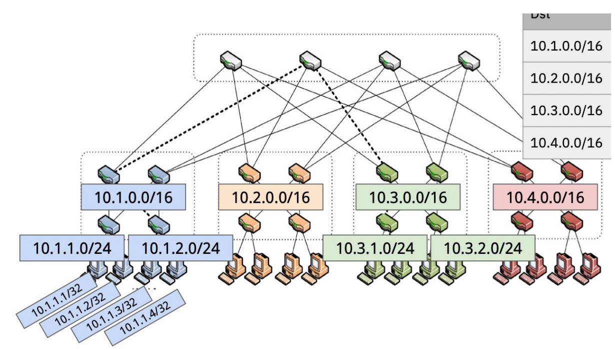
- In datacenters, queueing delay matters (because propagation delay is now in microseconds)
- Problem: TCP deliberately fills queues, leading to undesirably high queueing delay
- We've seen two different approaches for how current datacenters fix this problem
- #1 DCTCP: routers mark congestion using "Explicit Congestion Notification (ECN)" bit in header
  - Sender reduces CWND on seeing the ECN bit set (*vs.* waiting for packet loss)
- #2 Swift: delay-based CC (Nandita's lecture)
- Both #1 and #2 were relatively easy/incremental changes to switches and endhosts

Know the general approach behind DCTCP, Swift, pFabric; not expected to know design details, nor perf graphs

Lectures 16: Routing in Datacenters

Fully understand all this!

- Problem: fully utilizing the high bisection BW available →need multi-path routing
- Goal: use all the paths of equal cost between a source and destination
- Solution: Equal-Cost Multi Path (ECMP) forwarding
  - Every router maintains next-hop information for all paths of equal cost (vs. picking one based on tie-breaking)
  - Picks the next-hop along which to forward a packet by hashing flow-related fields in the packet's header
  - Implication: all packets in a flow follow the same path but different flows follow different paths
- ECMP is about how we do forwarding given multiple paths; we need to extend routing protocols to discover those multiple paths
  - We covered simple DV/LS extensions for this



Lectures 16: Routing in Datacenters

Fully understand all this!

- **Topology-aware addressing enables scalable routing between server *hosts***
- **However, in datacenters, we run many VMs on a host and apps require connectivity between *VMs***
- **Problem: VM addresses assigned by user (vs. operator) + VMs can be migrated → can't assume topo-aware addressing**
- **Solution: separate the problem of connectivity between physical hosts ("underlay") and VMs ("overlay")**
  - **Underlay connectivity: established by routing protocols as before (topology-aware addressing, etc)**
  - **Overlay connectivity: <u>encapsulation</u>**
- **Encapsulation: IP packet from VM1 to VM2 is carried as the payload of an IP packet from hosts H1 to H2**
  - **Implication: underlay only sees packets to/from H1 and H2; where VM1 runs on physical host H1, and VM2 on H2**
  - **Adding (removing) the underlay headers is done in a virtual switch that runs on host H1 (H2)**
- **One last problem: multi-tenancy means VMs from diff tenants might pick the same overlay**

Lectures 17: SDN

Understand the "why" of SDN

- **We've talked about the network's data and control plane, but there's also a *management plane***
  - **E.g., needed to configure router link costs, read telemetry counters, etc.**
  - **The management plane was much neglected until ~mid 2000s**
- **Fixing the management plane was challenging because operators couldn't innovate with router internals**
- **Solution: decouple management, control, and data planes in a router with open APIs between planes**

Understand the SDN arch and the role of OpenFlow

  - **Enables flexibility in *who* implements the control and management planes and flexibility in *where* we run these planes**
- **Early SDN proposal:**
  - **OpenFlow as the API between data and control planes**
  - **Data plane remains largely unchanged: implemented in routers by router vendors**

Know the "what and why" of use-cases; only the general approach to "how"

Rob's slides on SDN applied to the data plane and management plane are out of scope (slides 70+)

a

centralized controller (vs. in routers)

Outline

- Review:
  - Wireless and Cellular (lectures 24, 25)
  - Host Networking (lectures 22, 23)
  - Datacenters and SDN (lectures 15, 16, 17)
  - How the pieces fit (lectures 18-21)
    - Not explicitly reviewing individual details of HTTP, DNS, Ethernet, DHCP, etc.
  - Material covered by the midterm

- Will go as far as time permits; entire slide deck will be available

L2 and L3 have separate addressing

- **MAC (L2) addresses**
  - Hard-coded ("burned in") by device manufacturer
  - Not aggregation-friendly
  - Portable, and can stay the same as the host moves (*topology independent*)
  - Used to get packet between interfaces on the same L2 link/network

- **IP (L3) addresses**
  - Assigned by network operators; configured or learned dynamically (DHCP)
  - Hierarchical structure and allocation allows aggregation
  - Not portable and depends on where the host is attached (*topology dependent*)
  - Used to get the packet to the destination IP "subnet"

Bootstrap and discovery

- A host A is “born” knowing only its MAC address
- Must discover some information before it can communicate with a remote host B
- What is my (A’s) IP address?
  - DHCP
- What is B’s IP address?
  - DNS
- What is B’s MAC address? (if B is local)
  - ARP
- What is my first-hop router’s IP address (needed if B is remote)
  - DHCP
- What is my first-hop router’s MAC address?
  - ARP

ARP and DHCP

- **Discovery protocols**
  - ARP → Address Resolution Protocol
  - DHCP → Dynamic Host Configuration Protocol
  - Confined to the host's local L2 network
  - Rely on <u>broadcast</u> capability (as most discovery protocols)

- **ARP**
  - Initiating host broadcasts query: "*Who has IP address* w.x.y.z"?
  - Host with w.x.y.z responds (unicast): "*I am* w.x.y.z *and my MAC address is* a1:b2:c3:d4:e5:f6"

- **DHCP**
  - Used by a host to learn (bootstrap itself) about its L3 context
  - Discovers its IP address, netmask, IP address of first-hop router, IP address of

Key ideas in both ARP and DHCP

- **Broadcasting**: can use broadcast to make contact
  - Scalable because of limited size

- **Caching**: remember results for a while
  - Store the information you learn to reduce overhead
  - Associate a time-to-live field with the information
  - … and either refresh or discard the information

DNS: Quick review

- Why we need it? Convert names to IP addresses

- Design based on three intertwined hierarchies
  - **Naming structure**: names are hiearchical (cs.berkeley.edu)
  - **Management**: hierarchy of authority over names
  - **Infrastructure**: hierarchy of DNS servers

- Names are "resolved" by starting at the root and querying down the hierarchy

- Availability / scalability / performance: via partitioning, replication, caching

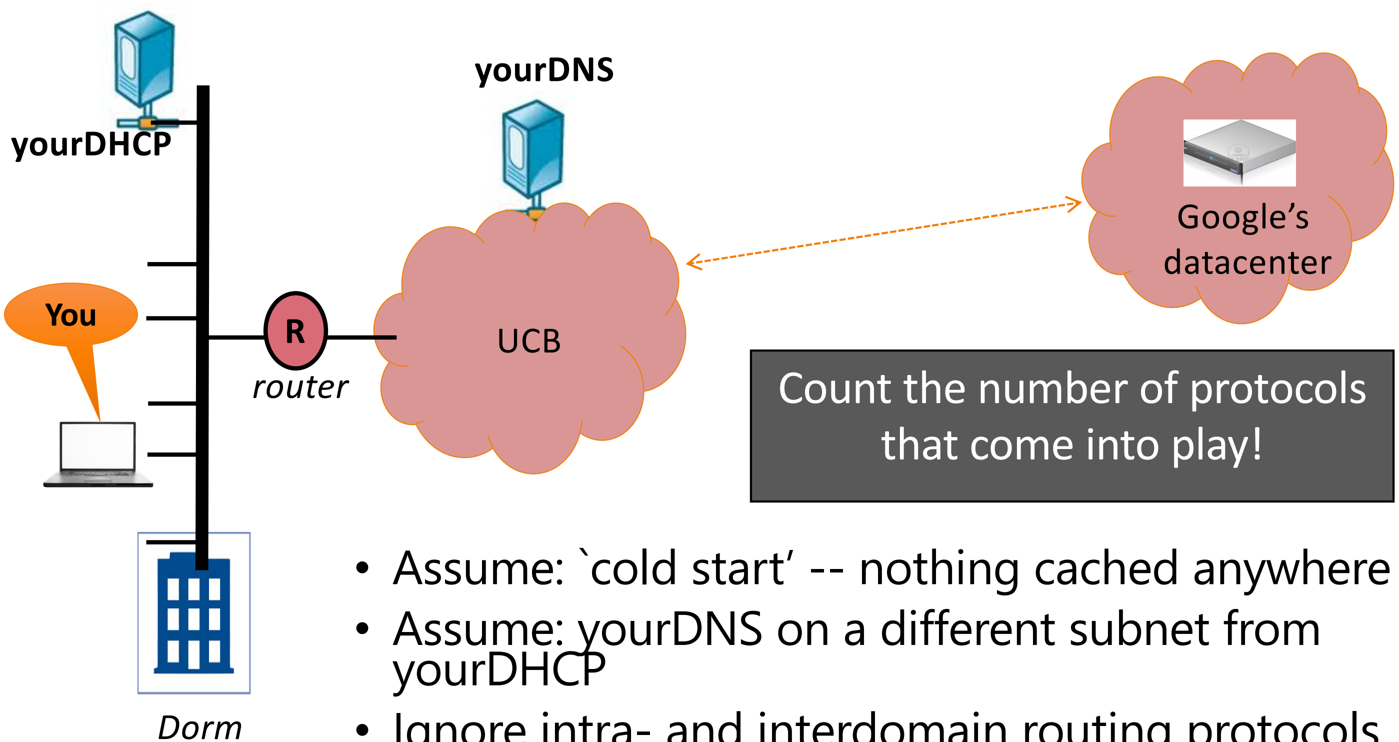
HTTP / Web: Quick Review

- Essential components:
  - **HTM**L: content with links
  - **URL**: reference to content (lot going on in a URL! – protocol, name, location, resource, parameters…)
  - **Infrastructure**: Client browsers and Web servers
  - **HTTP**: protocol used to fetch content from servers

- Availability, scalability, performance
  - Caching: at browser and forward/reverse proxy servers controlled by HTTP caching directives
  - CDNs: 3rd-party entity that replicates/caches/serves your content using their infrastructure
  - TCP optimizations: concurrent, persistent, pipelined connections amortize

Putting the pieces together

Walk through the steps required to download www.google.com/index.html from your laptop

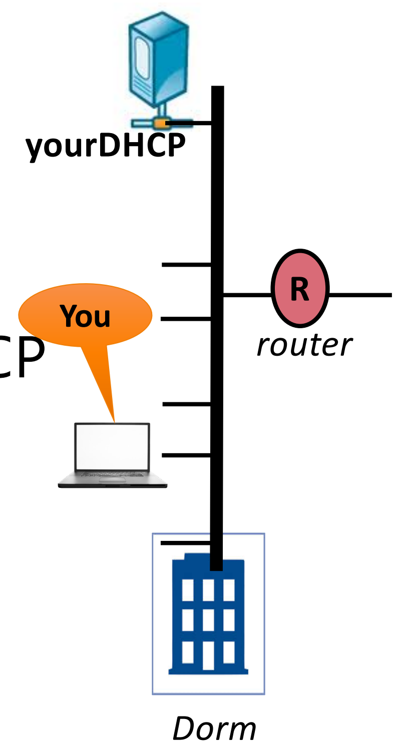


Count the number of protocols that come into play!

- Assume: `cold start' -- nothing cached anywhere
- Assume: yourDNS on a different subnet from yourDHCP
- Ignore intra- and interdomain routing protocols

Step 1: Self discovery

- You use DHCP to discover bootstrap parameters
  - your IP addr (u.u.u.u)
  - your DNS server's IP (u.dns.ip.addr)
  - R's IP address (r.r.r.r)
  - ..



- Exchange between you and yourDHCP

Ethernet	IP	UDP	DHCP

- Protocol count = 4

Next…

- You are ready to contact www.google.com

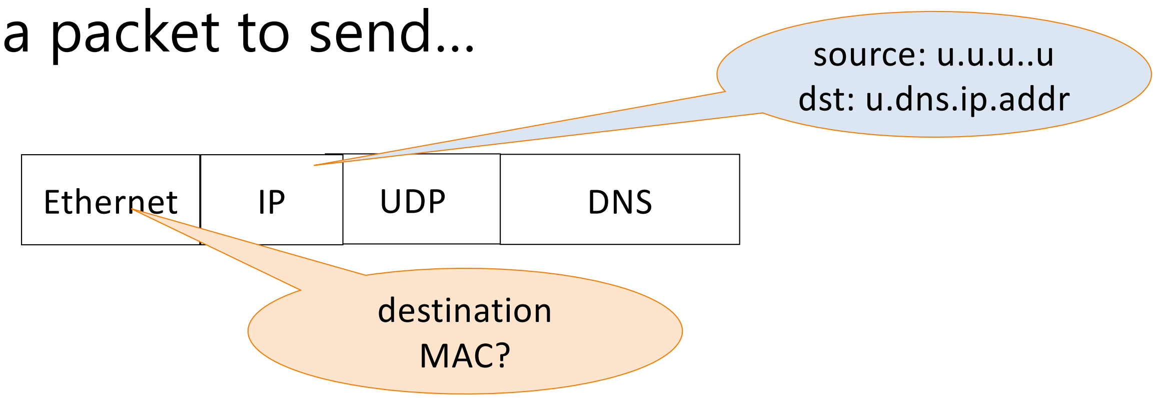
    → need an IP address for www.google.com

    → need to ask google's DNS server

    → need to ask my DNS server to ask google's DNS…

    → I know my DNS server's IP addr is u.dns.ip.addr

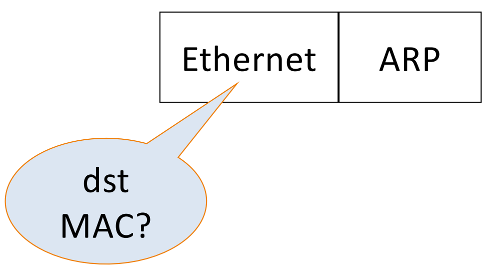
    → create a packet to send…



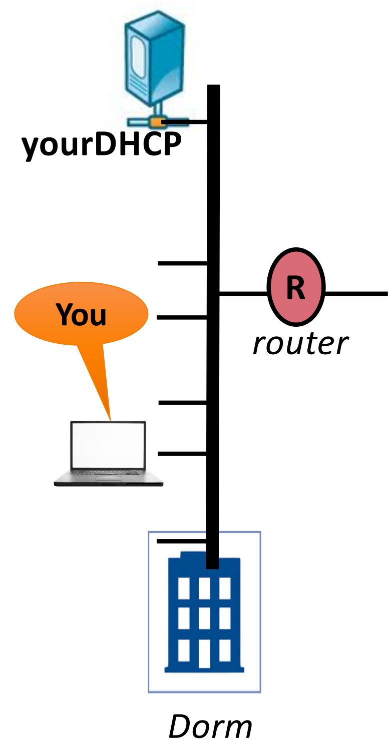
Step 2: Getting out the door

- You use ARP to discover the MAC address of R

- Exchange between you and R

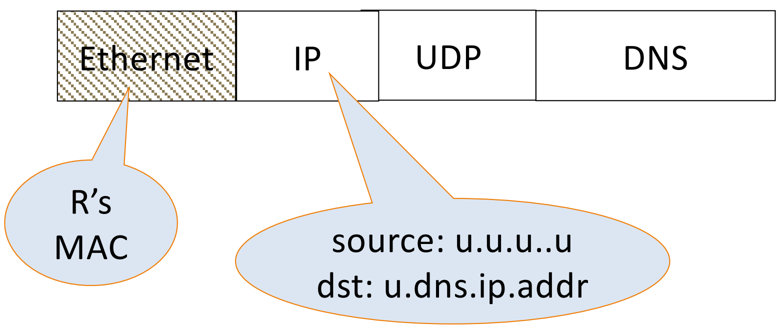


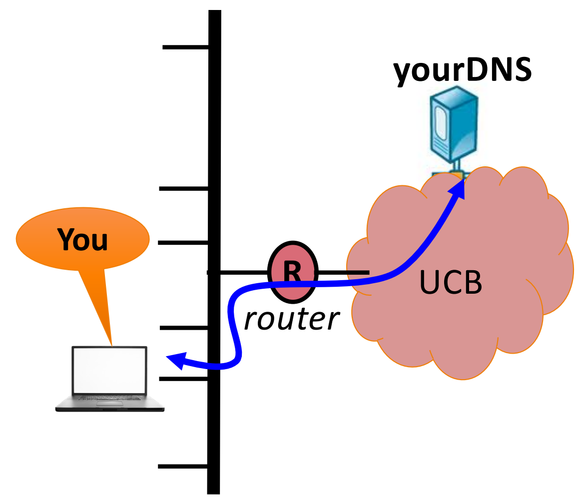
- Protocol count = 5



Step 3: Send a DNS request

- Exchange between you and yourDNS
- Now ready to send that packet

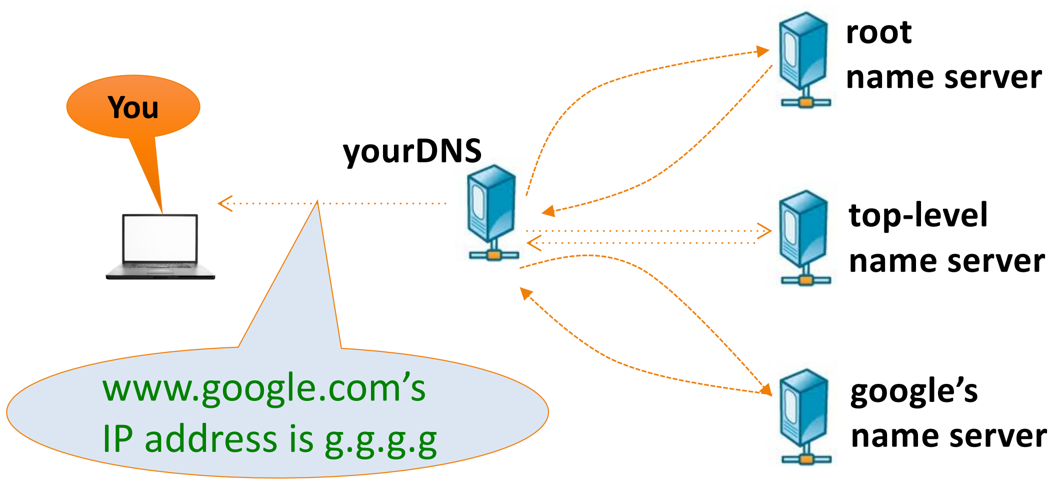




- Protocol count = 6

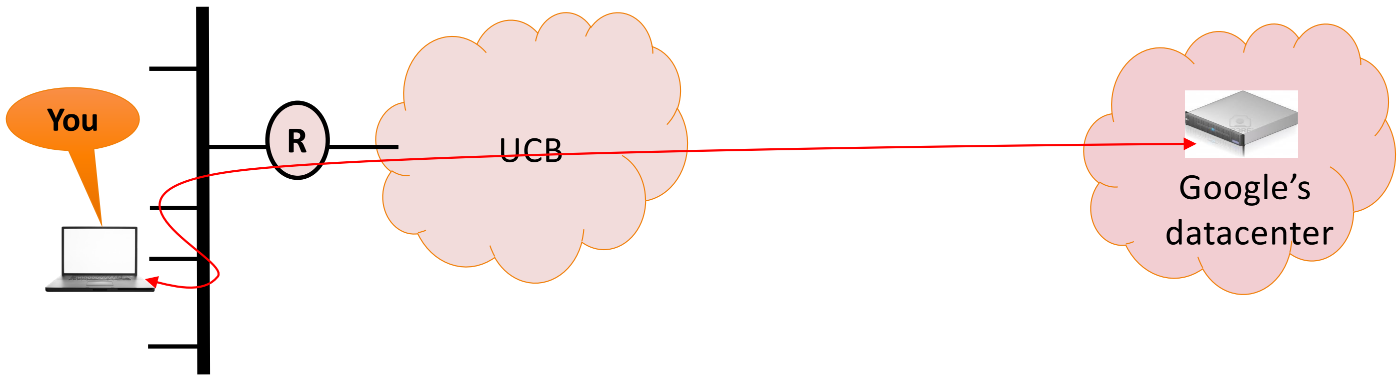
Step 4: yourDNS does its thing

- yourDNS resolves www.google.com

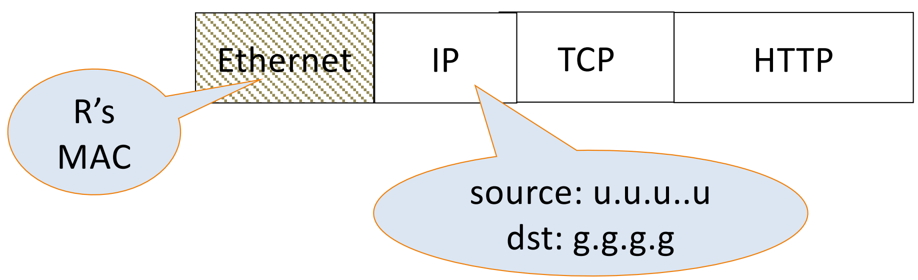


- Protocol count = 6

Step 5: Getting the content (at last)



- Exchange between you and google's server at g.g.g.g



- Protocol count = 8

Recap: Name discovery/resolution

- MAC addresses?
  - my own: hardcoded
  - others: ARP (given IP address)

- IP addresses?
  - my own: DHCP
  - others: DNS (given domain name)
    - how do I bootstrap DNS communication? (DHCP)

- Domain names?
  - search engines

Outline

- Review:
  - Wireless and Cellular (lectures 24, 25)
  - Host Networking (lectures 22, 23)
  - Datacenters and SDN (lectures 15, 16, 17)
  - How the pieces fit (lectures 18-21)
    - Not explicitly reviewing individual details of HTTP, DNS, Ethernet, DHCP, etc.
  - Material covered by the midterm

- Will go as far as time permits; entire slide deck will be available

Material covered by the midterm

- tl;dr: pre-midterm material will be more lightly tested than post midterm
- You should definitely know concepts that are necessary building blocks for the material we've covered since the midterm
- Following slides just elaborate on this …

Lectures 1-4: Overview and Architecture

- Packet delay and link characteristics
- Sharing network bandwidth: best-effort vs. reservations
- Notion of layering and layers in the Internet architecture (L1-L7)
- What layers are implemented where & the end-to-end argument
- Protocols, packet headers, header encap/decap, life of a packet

Lectures 5-8: Intra-domain Routing

- General concepts in routing
  - Control vs. data plane
  - Routing vs. forwarding
  - Neighbors, route advertisements, forwarding tables, next-hop
  - Link weights and least-cost paths
  - Deadends, loops, convergence

Lectures 5-8: Intra-domain Routing

Remember the general idea behind different routing approaches:

- DV: I tells my neighbors about my lowest-cost distance to every destination
- LS: I tell everyone about my immediate links/neighbors
- Know that DV/LS typically operate at L3

Lectures 5-8: IP Routers and IP Addressing

- **IP addresses (CIDR):** hierarchical allocation, prefixes, masks
- **IP routers:** overall architecture: control proc. vs. linecards, fast path vs. slow path

- **IP Forwarding**: based on longest-prefix match (LPM)

- **IP header:** you should be familiar with key concepts relevant to the IP header
  - Why we have checksums, fragmentation, protocol field

Lectures 9-10: Inter-domain Routing

- **Concepts you should know**
  - Autonomous systems (domains)
  - Providers and their biz. relationships (customer-provider vs. peering)
  - Hierarchical addressing
  - That inter-domain routing operates on address prefixes
  - That hierarchical addressing enables scalability in inter-domain routing
  - That inter-domain route selection is driven by policy

Lectures 11-12: Reliability

- **Know the building blocks of reliable protocols**
  - Checksums
  - Cumulative ACKs, duplicate ACKs
  - Timeouts
  - Retransmissions
  - Sequence numbers
  - Sliding windows

Lectures 11-12: TCP, UDP

- **Know the TCP abstraction**
  - **Reliable, in-order bytestream**
  - **Concepts: connection, connection state, 3-way handshake connection setup/teardown**
  - **Understand TCP's role in the overall arch (L4, implemented at end hosts)**
  - **TCP functionality: mux/demux, reliability, flow control, congestion control**
- **Also, UDP abstraction (best-effort packet delivery) and how it differs from TCP**

Lectures 13-14: Congestion Control

- CC used to allocate network BW
- Goals: low packet loss/delay, high like utilization, fair sharing
- Design space ideas that you should recall at a high level
  - Dynamic adjustment vs. reservations
  - Host-based vs. router-assisted
- Know the general approach TCP follows:
  - host-based, with dynamic adjustment based on AIMD, loss as signal, etc.
  - Pros/cons of the above approach
- Trends and implications revealed by the TCP throughput equation

Thanks & Good luck!