Web Services and SOA Standards

Ken Birman

Cornell University. CS5410 Fall 2008.

A story of standards…

- What's a standard?
  - Historically, the industry has advanced in surges
  - First, a major advance occurs, like first web browser
  - Big players jump on board, agree to cooperate to ensure interoperability of their products, which will innovate in terms of the user experience but standardize "internals"
- Today, we're awash in standards
  - But creating a standard isn't any formula for success
  - There are far more *ignored* standards than *adopted* ones

A short history of standards

- Some standards that mattered
  - CORBA: general object-oriented interoperability
  - J2EE: Java runtime environment
  - .NET: Microsoft's distributed computing infrastructure
  - Web Services: the web, but not limited to browsers interacting to web servers.
    - Web services use the same standards
    - But the focus on programs that interact by exchanging documents (web pages) that encode information

(Today) Web Services are “hot”

- This is the basic standard employed in cloud computing systems
  - Internet is at the “bottom” of the stack
  - Then layer on standards used when browsers talk to web servers (HTTP) and to encode those pages (HTML)
  - Web services run *over* HTTP and HTML, but the web pages have their own mandatory encoding, called SOAP. It describes requests and responses on services
- The associated architecture is referred to as a “service oriented architecture” (SOA) and the systems built this way are “service oriented systems” (SOS).

Turtles all the way down…

“ A well-known scientist (some say it was Bertrand Russell) once gave a public lecture on astronomy. He described how the earth orbits around the sun and how the sun, in turn, orbits around the center of a vast collection of stars called our galaxy. At the end of the lecture, a little old lady at the back of the room got up and said: "What you have told us is rubbish. The world is really a flat plate supported on the back of a giant tortoise." The scientist gave a superior smile before replying, "What is the tortoise standing on?" "You're very clever, young man, very clever," said the old lady. "But it's turtles all the way down!"

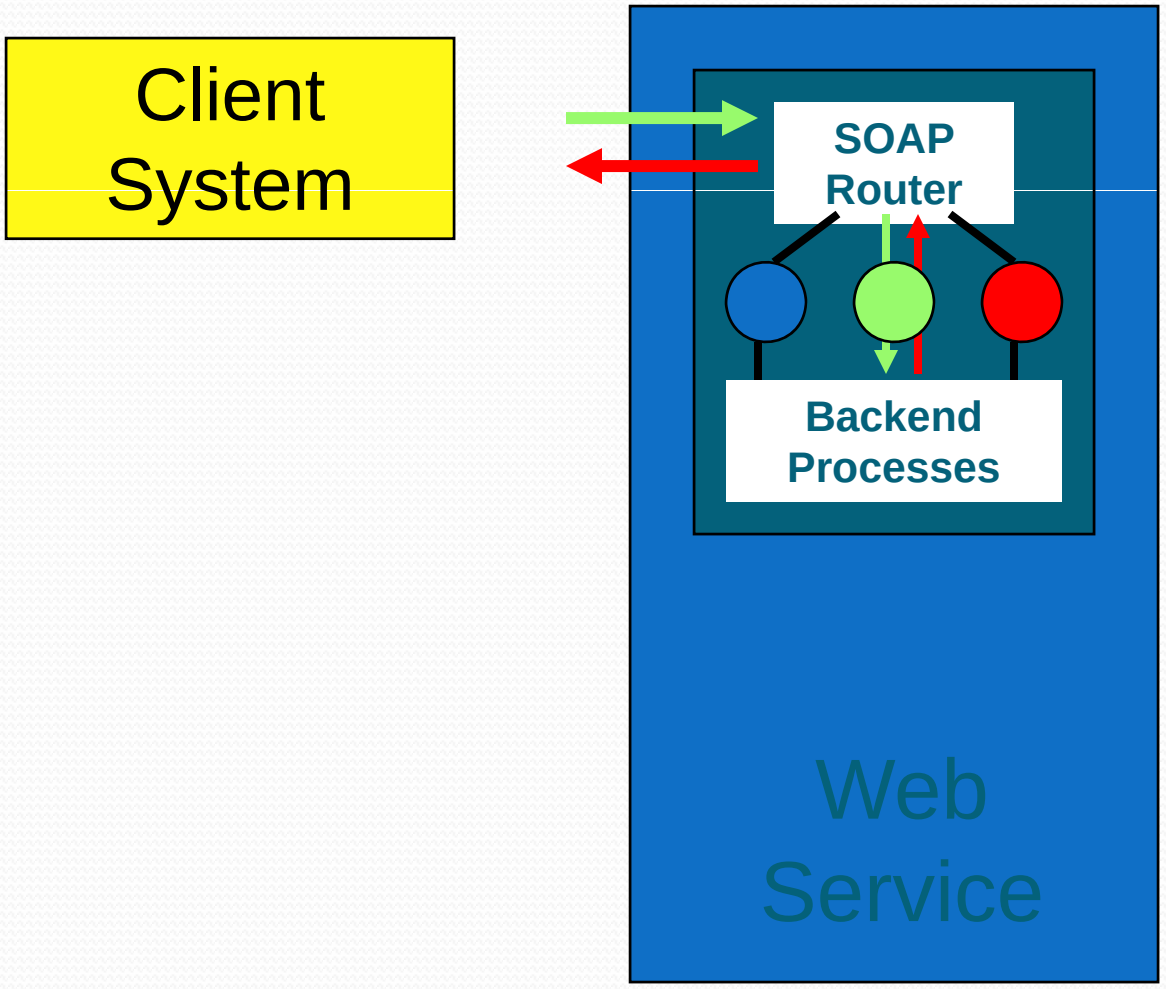
Standards all the way down…

- We're starting to see a second generation of standards layered on the basic web services ones
  - XML on the bottom (web page stuff)
  - Then web services on top of the web page stuff
  - Then, for example, the military "global information grid" (GIG) layered over web services
  - Other emerging standards: financial data centers, medical computing systems, etc
- These generally adopt the underlying standard, then add additional rules for using it for specific purposes

Elements of the standard?

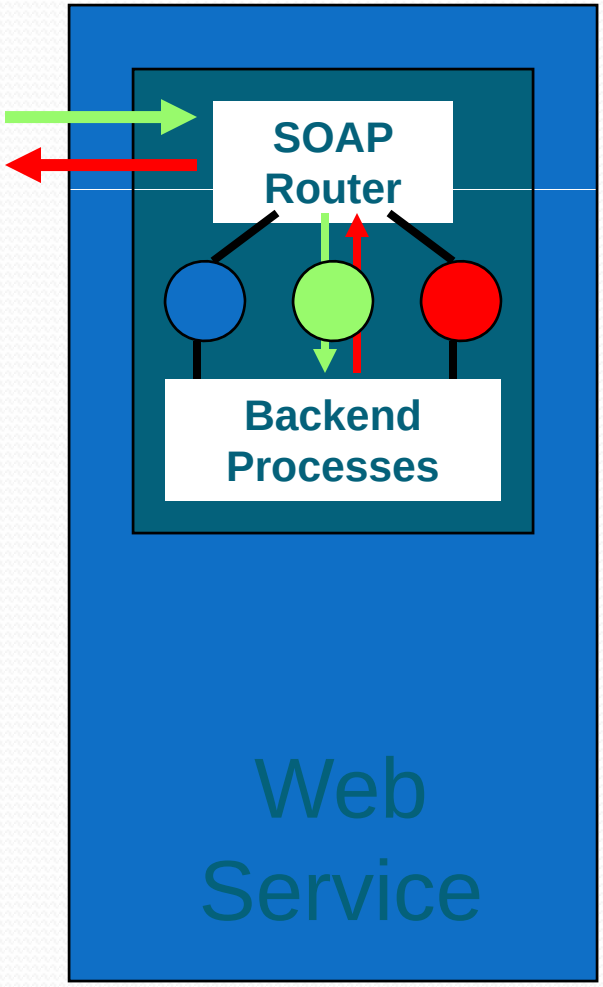
- A collection of documents that spell out the rules
  - There are a *great many* of these documents
  - And like many standards, not all have been widely adopted
- Vendors like Microsoft, BEA, IBM (even Google) have their own platforms implementing parts of these documents; in theory the systems interoperate
- But they also compete, by innovating around the edges

Basic Web Services model



Basic Web Services model

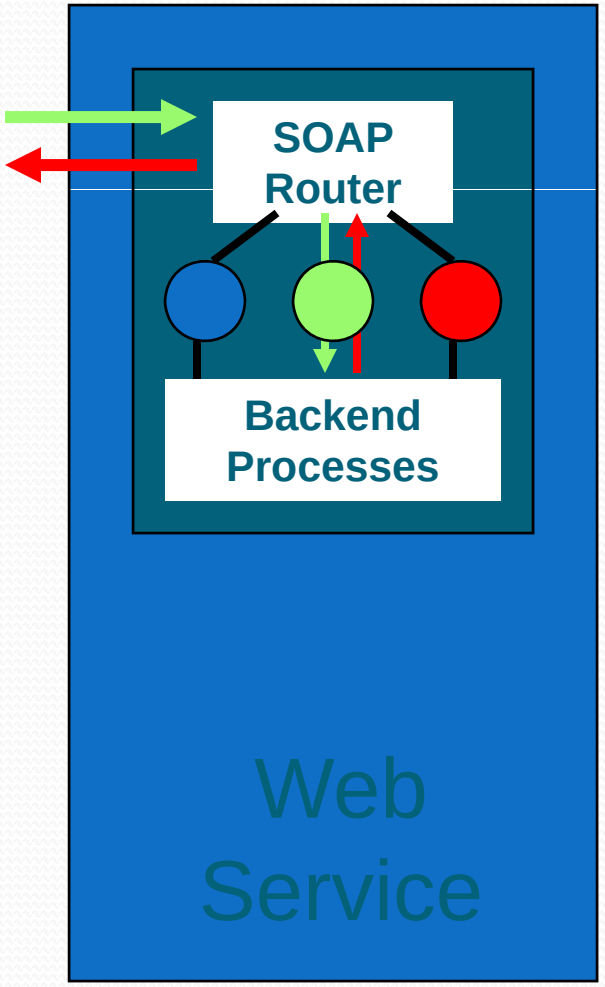
- "Web Services are software components described via WSDL which are capable of being accessed via standard network protocols such as SOAP over HTTP."



Basic Web Services model

- "Web Services are software components described via WSDL which are capable of being accessed via standard network protocols such as SOAP over HTTP."

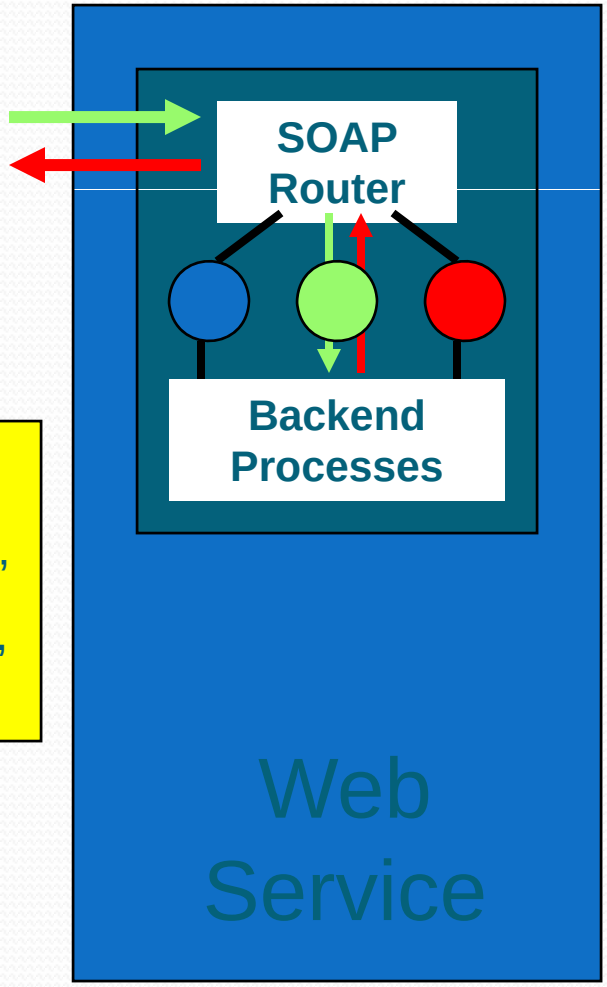
Today, SOAP is the primary standard. SOAP provides rules for encoding the request and its arguments.



Basic Web Services model

- "Web Services are software components described via WSDL which are capable of being accessed via standard network protocols such as SOAP over HTTP."

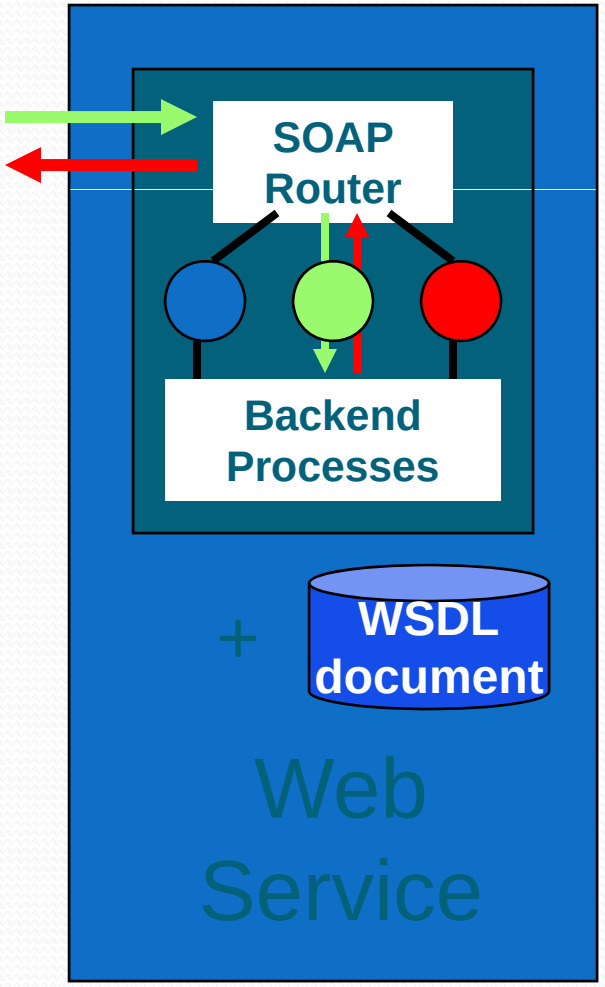
Similarly, the architecture doesn't assume that all access will employ HTTP over TCP. In fact, .NET uses Web Services "internally" even on a single machine. But in that case, communication is over COM



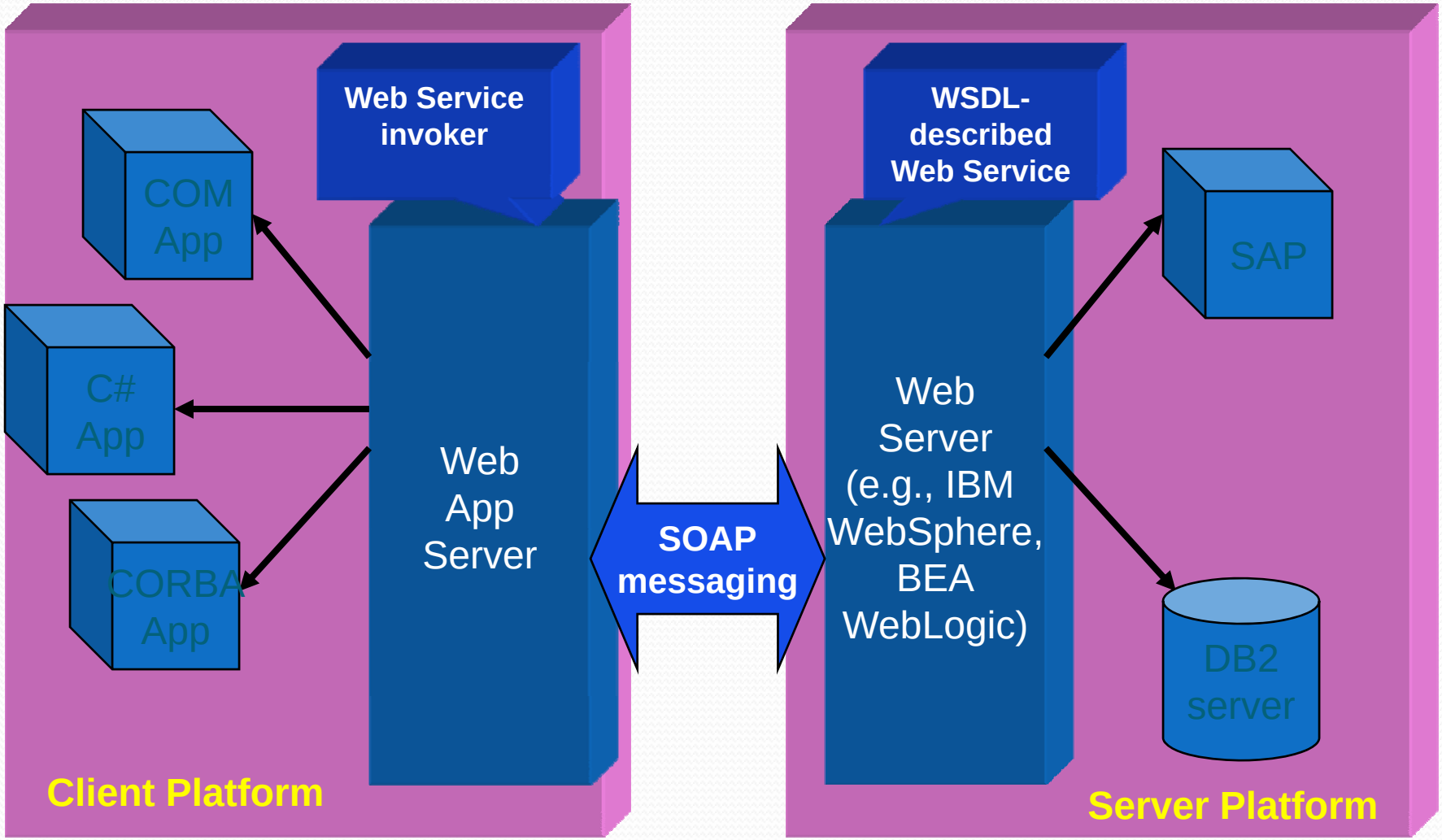
Basic Web Services model

- "Web Services are software components described via WSDL which are capable of being accessed via standard network protocols such as SOAP over HTTP."

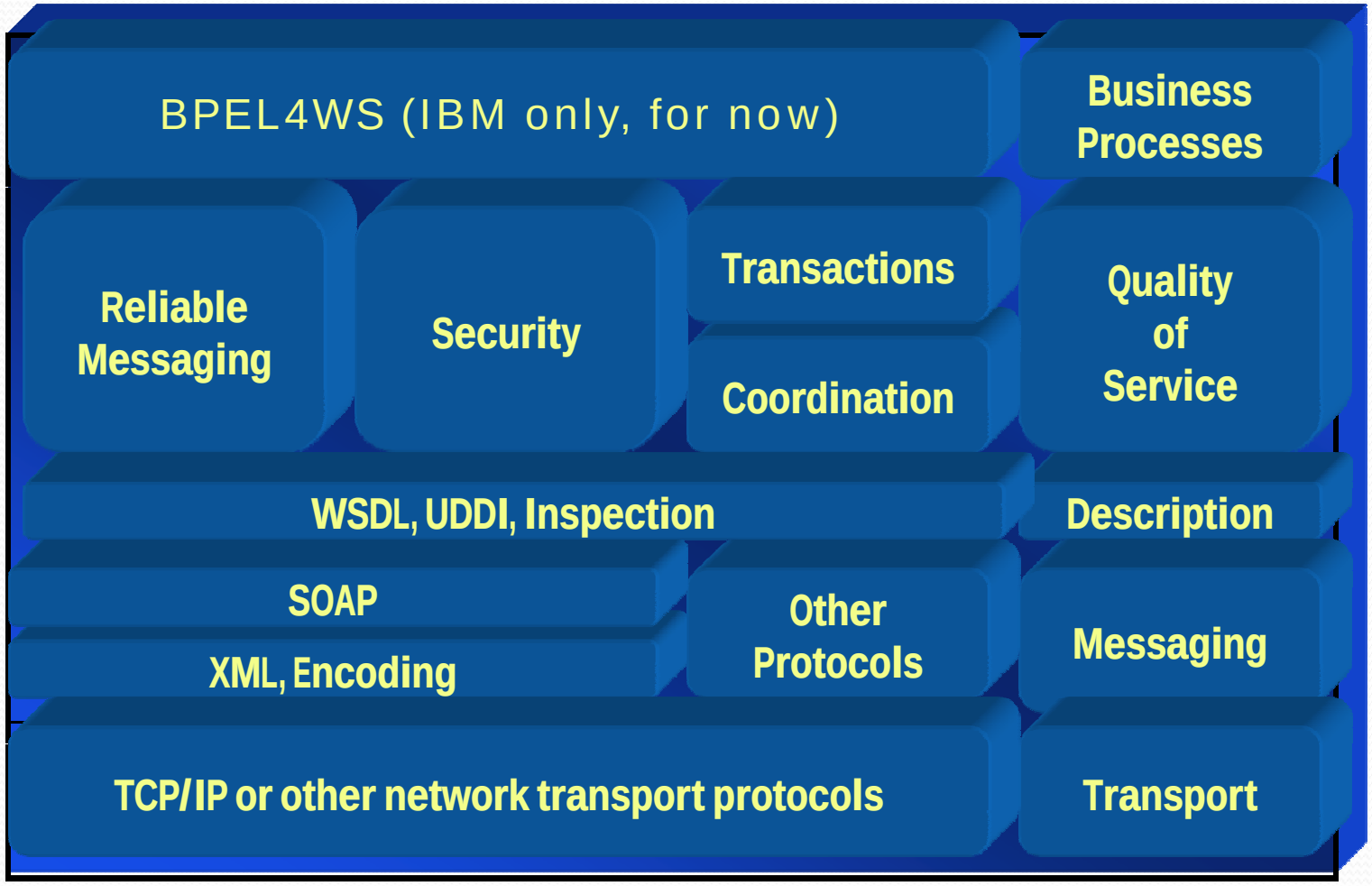
WSDL documents are used to drive object assembly, code generation, and other development tools.



Web Services are often Front Ends



The Web Services “stack”



How Web Services work

- First the client discovers the service.
- Typically, client then binds to the server.
- Next build the SOAP request and send it
  - SOAP router routes the request to the appropriate server(assuming more than one available server)
  - Can do load balancing here.
- Server unpacks the request, handles it, computes result. Result sent back in the reverse direction: from the server to the SOAP router back to the client.

Marshalling Issues

- Data exchanged between client and server needs to be in a platform independent format.
  - "Endian"ness differ between machines.
  - Data alignment issue (16/32/64 bits)
  - Multiple floating point representations.
  - Pointers
  - (Have to support legacy systems too)

Marshalling…

- In Web Services, the format used is XML.
  - In UNICODE, so very verbose.
  - There are other, less general, but more efficient formats.

Comparing with CORBA

- CORBA is an older and very widely adopted standard
  - J2EE mimics it in most ways
  - .NET (Windows) is very similar in style

- Models applications as (big) "objects" that export interfaces (methods you can call, with typed args)
- Then standardizes various tools for managing them
- Also provides for ways of connecting data centers over a WAN protocol of their design (which runs on TCP)

Comparing with CORBA

CORBA

- *Object* centric
  - RPC / remote method invocation with typed interfaces
- Much emphasis on semantics of active objects
- Standardizes most OO infrastructure

Web Services

- *Document* centric
  - Services treated as document processors
  - But can still do RPC…
- Document defines its own needs and services try to carry them out
- Standardizes things documents can express

Remote method invocation

- Also called Remote Procedure Call: Invoke a procedure on a remote machine "just" as you would on the local machine.
- Introduced by Birrell and Nelson in 1985
- Idea: mask distributed computing system using a "transparent" abstraction
  - Looks like normal procedure call
  - Hides all aspects of distributed interaction
  - Supports an easy programming model
- Today, RPC is the core of many distributed systems.
- Can view the WS client server interaction as an RPC.

RPC Optimization

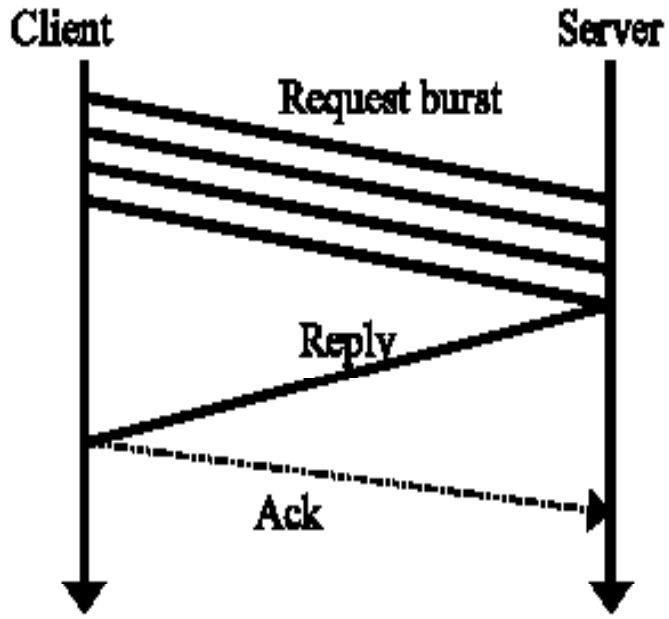


Figure 4.4. RPC using a burst protocol; here the reply is sent soon enough so that an acknowledgement to the burst is not needed.

- Delay sending acks, so that imminent reply itself acts as an ack.
- Don't send acks after each packet.
- Send ack only at the end of transmission of entire RPC request.
- NACK sent when missing sequence number detected

RPC – what can go wrong?

- Network failure, client failure, server failure
- Assuming only network idiosyncrasies for now…
- RPCs use acks to make packet transmission more reliable.
  - If timeout with no ack, resend packet.
  - Leads to the issue of replayed requests.
- Each packet has a sequence number and timestamp embedded to enable detection of duplicates.

What happens when machines could fail too?

- What does a failed request mean?
  - Network failure and/or machine failure!
  - Client that issued request would not know if the server processed the request or not.

How about layering RPC on TCP?

- Web services often (not always) run over TCP
- TCP gives reliable in-order delivery, flow control and congestion control.
  - Reliable: Acknowledgments and retransmissions.
  - In-order: Sequence numbers embedded in each message.
  - Flow Control: Max allowed window size.

TCP…

- Congestion Control: the saw tooth curve
  - Ramp up as long as no timeouts.
    - Slow-start phase – exponential increase (until the slow-start threshold is hit)
    - Congestion Avoidance phase – additive increase
  - Multiplicative Decrease on timeout.

TCP optimizations

- Random Early Detection
- Selective Acknowledgments
- Fast Retransmit/Recovery

Back to RPC on TCP:

- TCP gives reliable communication when both ends and the network connecting them are up.
- So the RPC protocol itself does not need to employ timeouts and retransmission.
  - Simpler RPC implementation.
  - But the failure semantics remain the same (weak)

RPC Semantics

- "Exactly Once"
  - Each request handled exactly once.
  - Impossible to satisfy, in the face of failures.
    - Can't tell whether timeout was because of node failure or communication failure.

RPC Semantics…

- “At most Once”
  - Each request handled at most once.
  - Can be satisfied, assuming synchronized clocks, and using timestamps.
- “At least Once”
  - If client is active indefinitely, the request is eventually processed (maybe more than once)

Most data centers are behind a NAT box

- Overcomes limited size of IPv4 address space
- Role is to translate a large number of internal host addresses (Amazon or Google might have tens of thousands of machines at each data center) into a small number of externally visible ones
- Can also play a load-balancing function

Discovery

- This is the problem of finding the “right” service
  - In our example, we saw one way to do it – with a URL
  - Web Services community favors what they call a URN: Uniform Resource Name
- But the more general approach is to use an intermediary: a discovery service

Example of a repository

Name	Type	Publisher	Toolkit	Language	OS
Web Services Performance and Load Tester	Application	LisaWu		N/A	Cross-Platform
Temperature Service Client	Application	vinuk	Glue	Java	Cross-Platform
Weather Buddy	Application	rdmgh724890	MS .NET	C#	Windows
DreamFactory Client	Application	billappleton	DreamFactory	Javascript	Cross-Platform
Temperature Perl Client	Example Source	gfinke13		Perl	Cross-Platform
Apache SOAP sample source	Example Source	xmethods.net	Apache SOAP	Java	Cross-Platform
ASS 4	Example Source	TVG	SOAPLite	N/A	Cross-Platform
PocketSOAP demo	Example Source	simonfell	PocketSOAP	C++	Windows
easysoap temperature	Example Source	a00	EasySoap++	C++	Windows
Weather Service Client with MS- Visual Basic	Example Source	oglimmer	MS SOAP	Visual Basic	Windows
TemperatureClient	Example Source	jgalyan	MS .NET	C#	Windows

Roles?

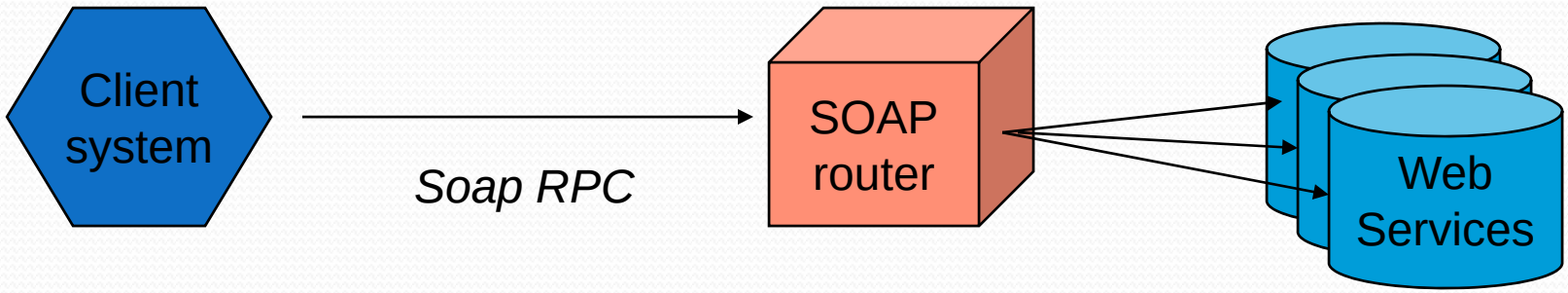
- UDDI is used to write down the information that became a “row” in the repository (“I have a temperature service…”)
- WSDL documents the interfaces and data types used by the service
- But this isn’t the whole story…

Discovery and naming

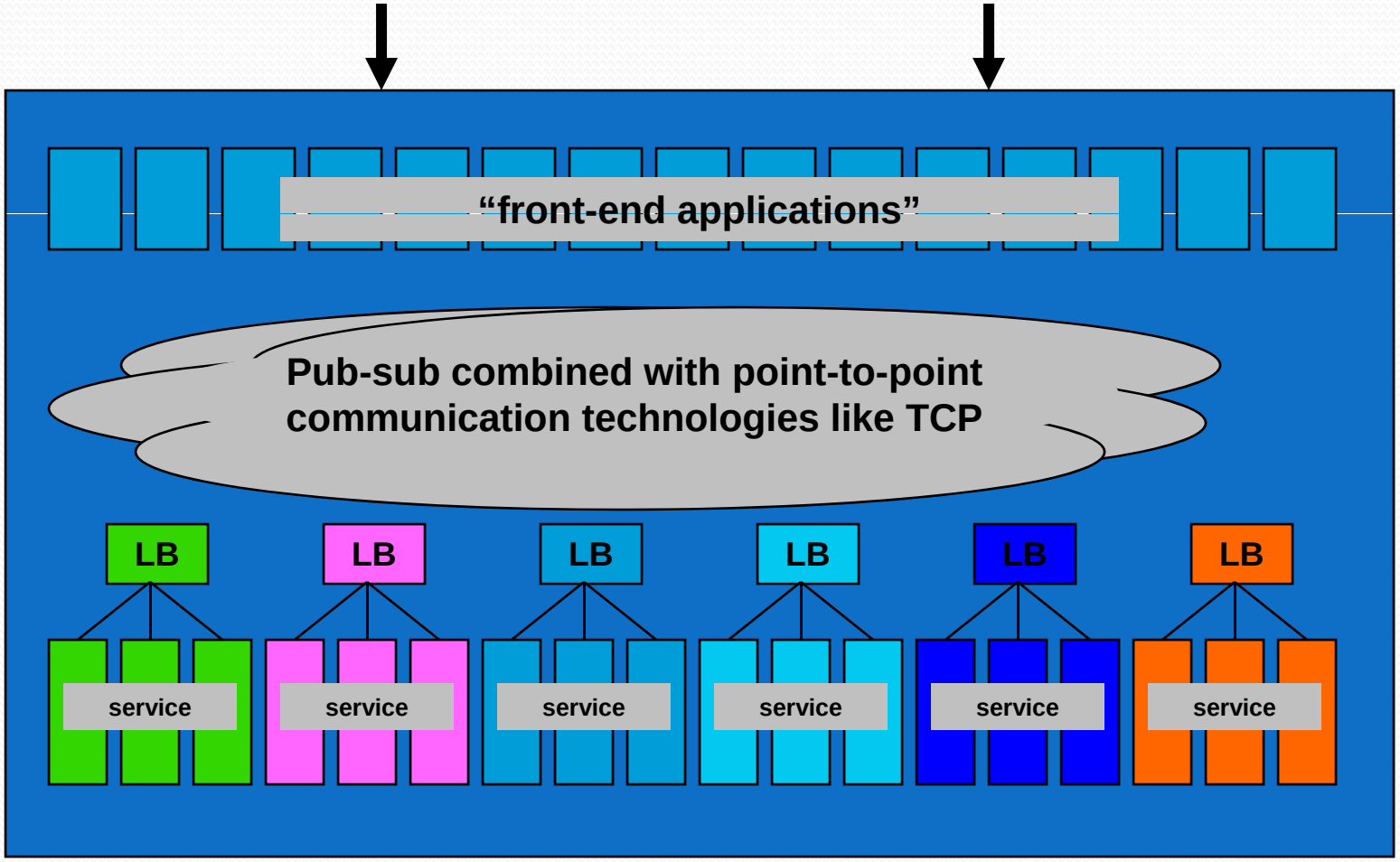
- The topic raises some tough questions
  - Many settings, like the big data centers run by large corporations, have rather standard structure. Can we automate discovery?
  - How to debug if applications might sometimes bind to the wrong service?
  - Delegation and migration are very tricky
  - Should a system automatically launch services on demand?

Client talks to eStuff.com

- One big issue: we're oversimplifying
- We think of remote method invocation and Web Services as a simple chain:



A glimpse inside eStuff.com



In fact things are even more complex….

- Major providers often have multiple centers in different locations

- So: You access "Amazon.com" but
  - Which data center should see your request?
  - When it arrives, which front-end host should handle it?
  - That host will parallelize page construction… using multiple services
  - Those are replicated: which servers will be used?

To illustrate, look at CDNs

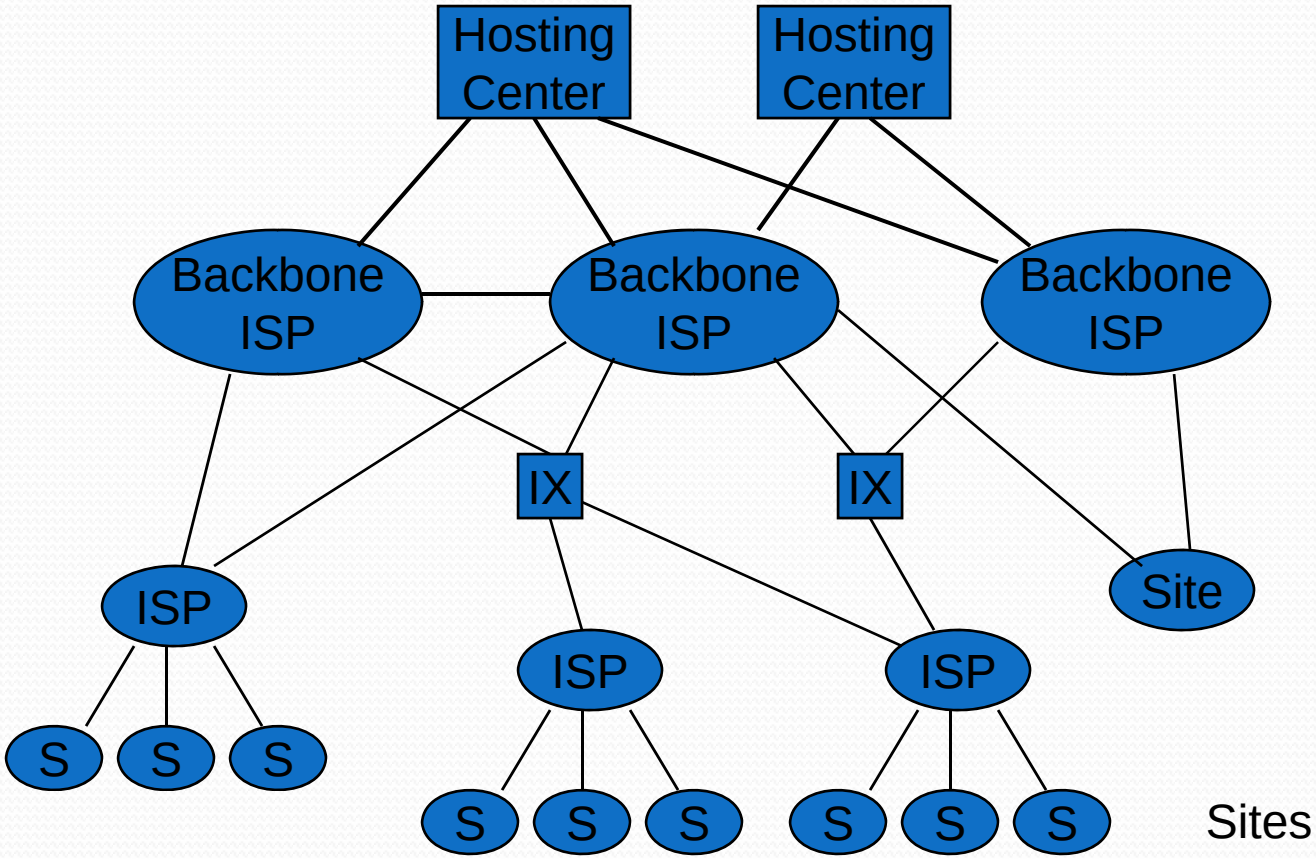
- Content distribution networks serve up videos and other web content
- A simpler case than full-scale web services, but enough to see some of the major mechanisms in action
- Used whenever you access a page with lots of images on it, like the home page at Yahoo! or live.msn.com

Basic event sequence

- Client queries directory to find the service
- Server has several options:
  - Web pages with dynamically created URLs
    - Server can point to different places, by changing host names
    - Content hosting companies remap URLs on the fly. E.g. http://www.akamai.com/www.cs.cornell.edu (reroutes requests for www.cs.cornell.edu to Akamai)
  - Server can control mapping from host to IP addr.
    - Must use short-lived DNS records; overheads are very high!
    - Can also intercept incoming requests and redirect on the fly

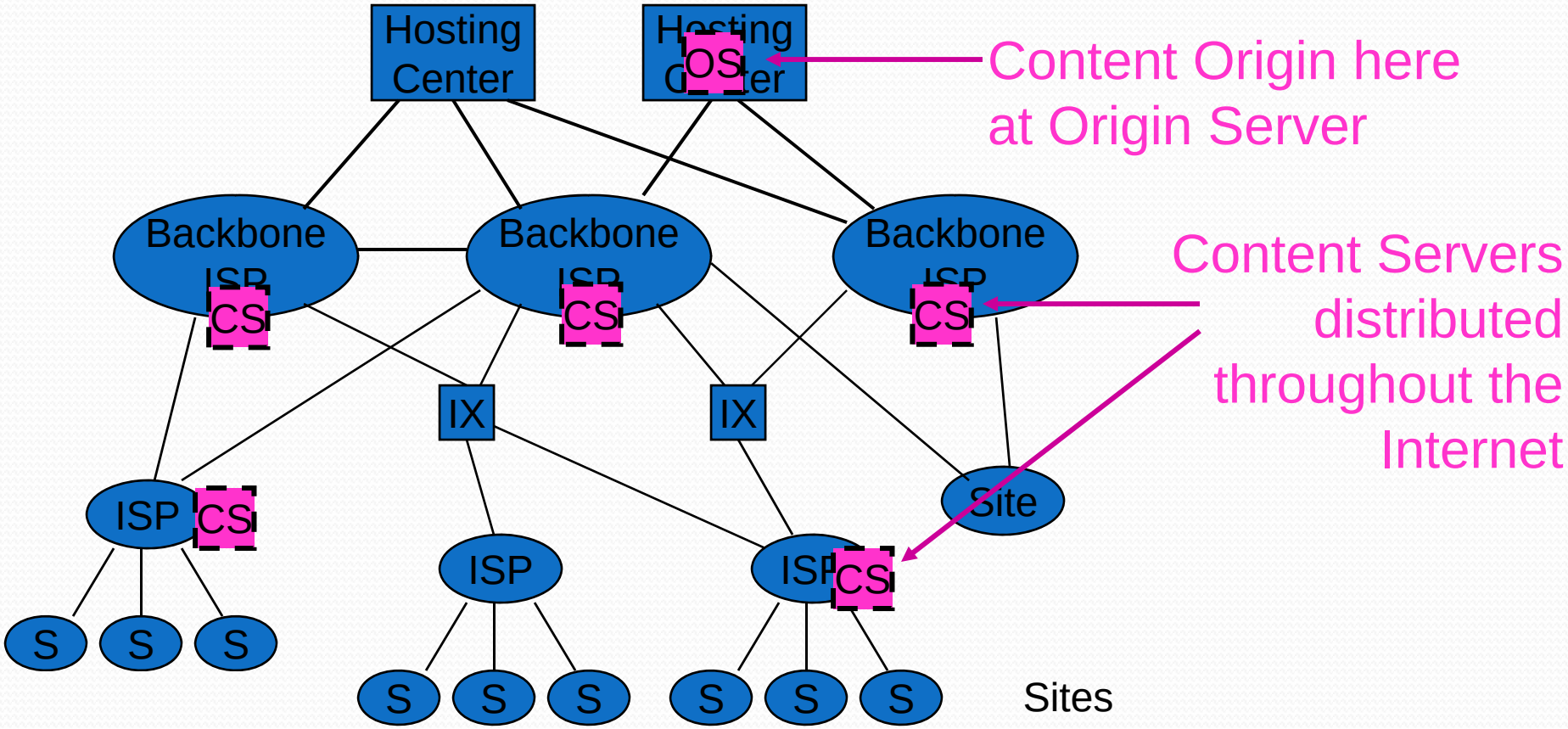
Content Routing Principle

(a.k.a. Content Distribution Network)



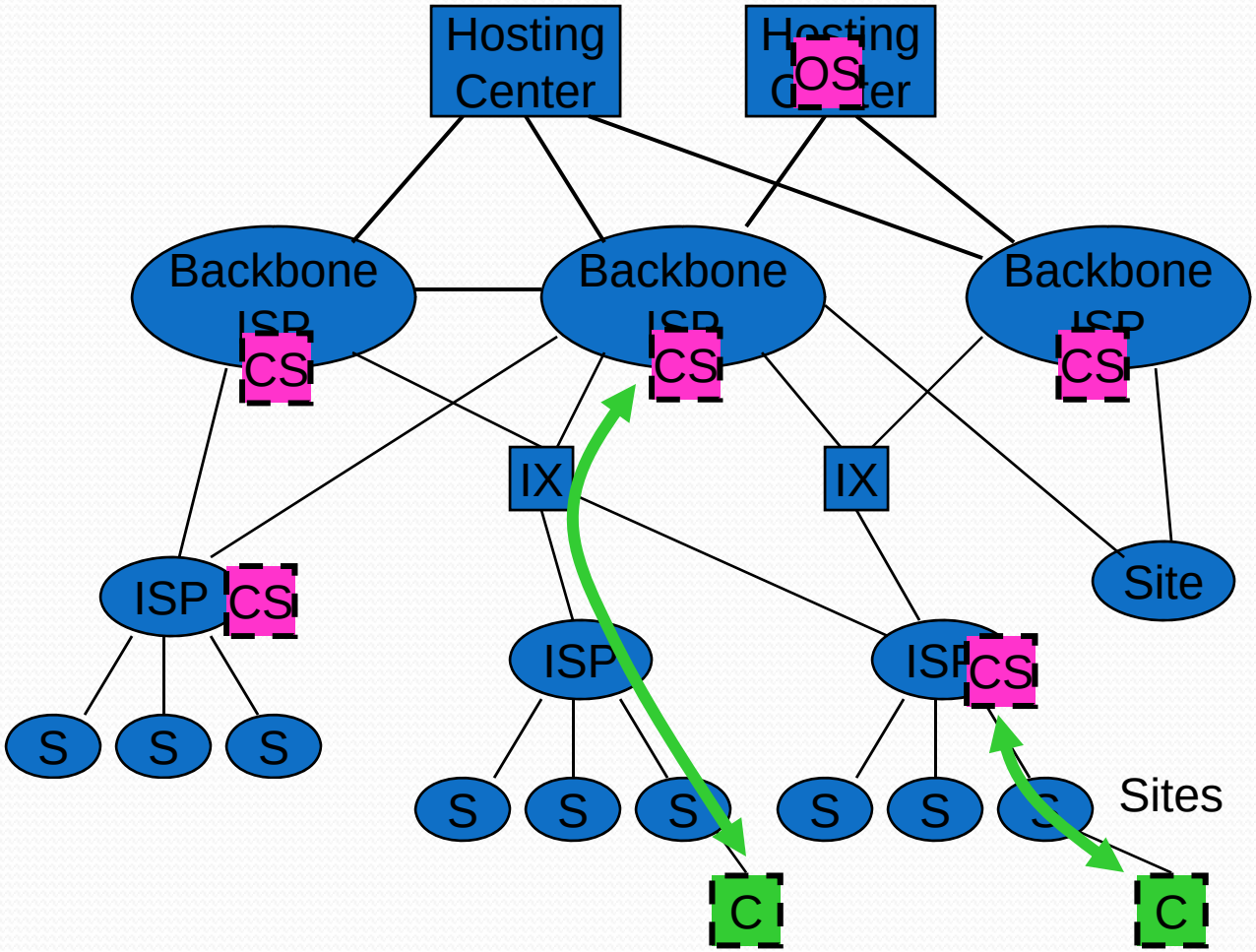
Content Routing Principle

(a.k.a. Content Distribution Network)



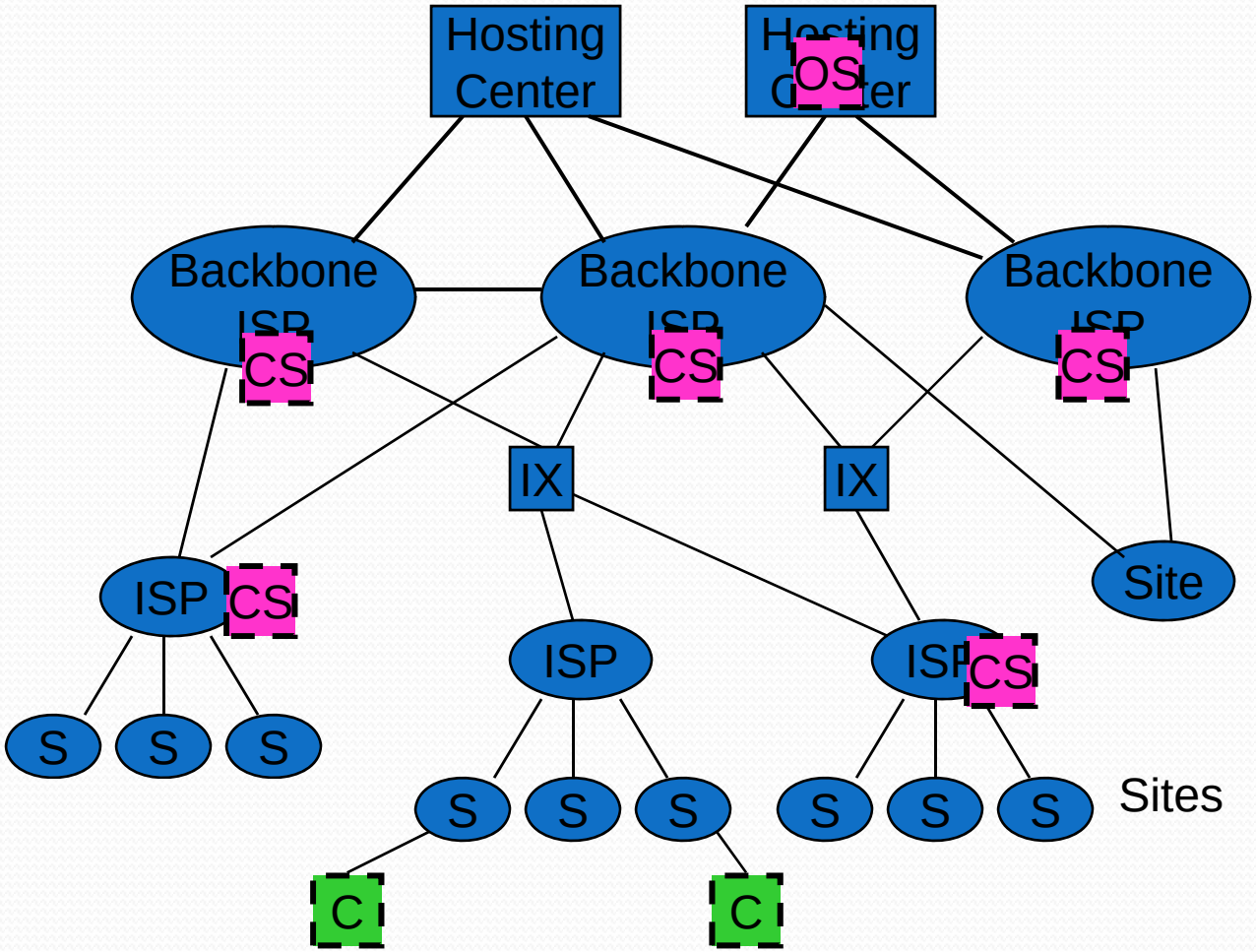
Content Routing Principle

(a.k.a. Content Distribution Network)

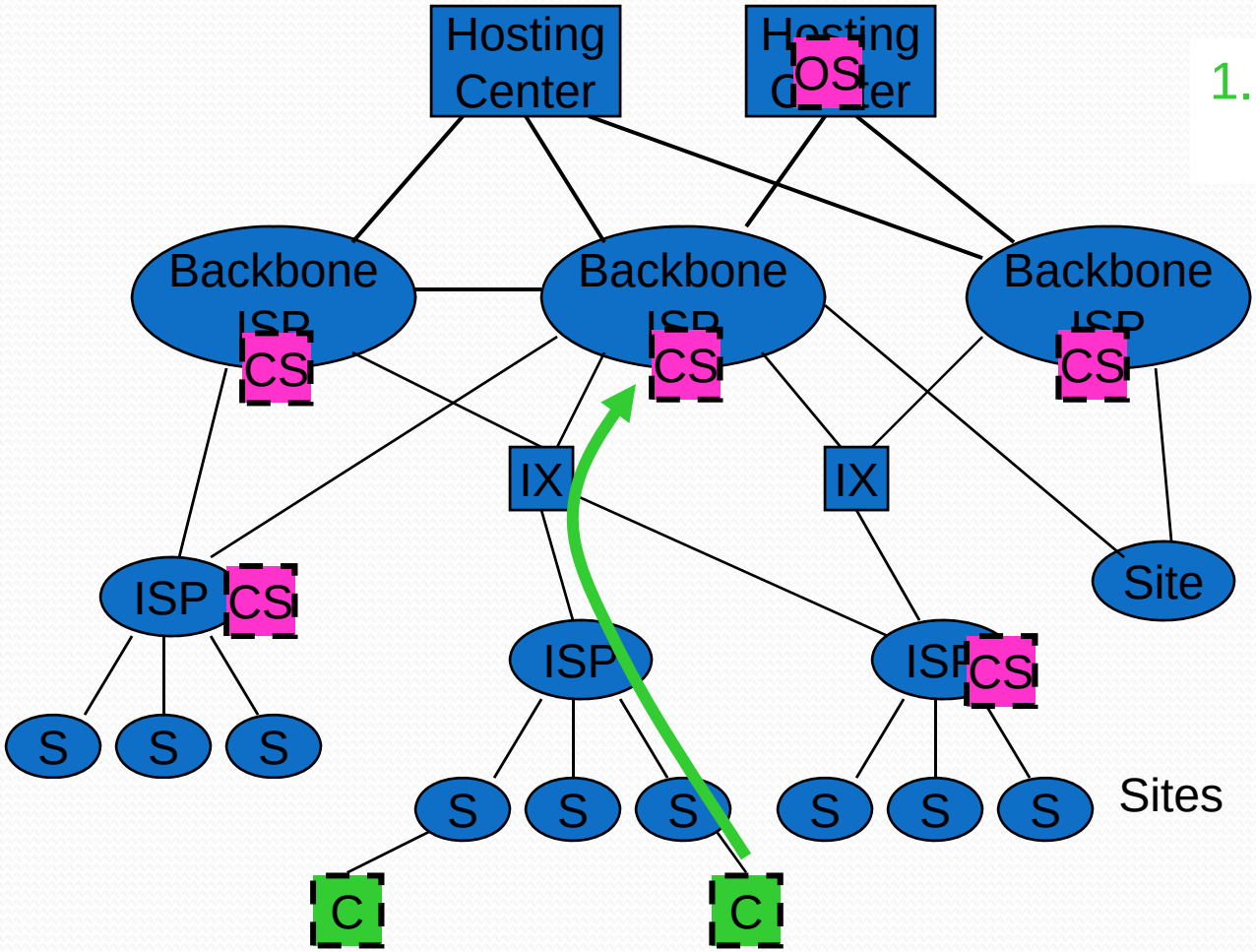


Content is served from content servers nearer to the client

Two basic types of CDN: cached and pushed

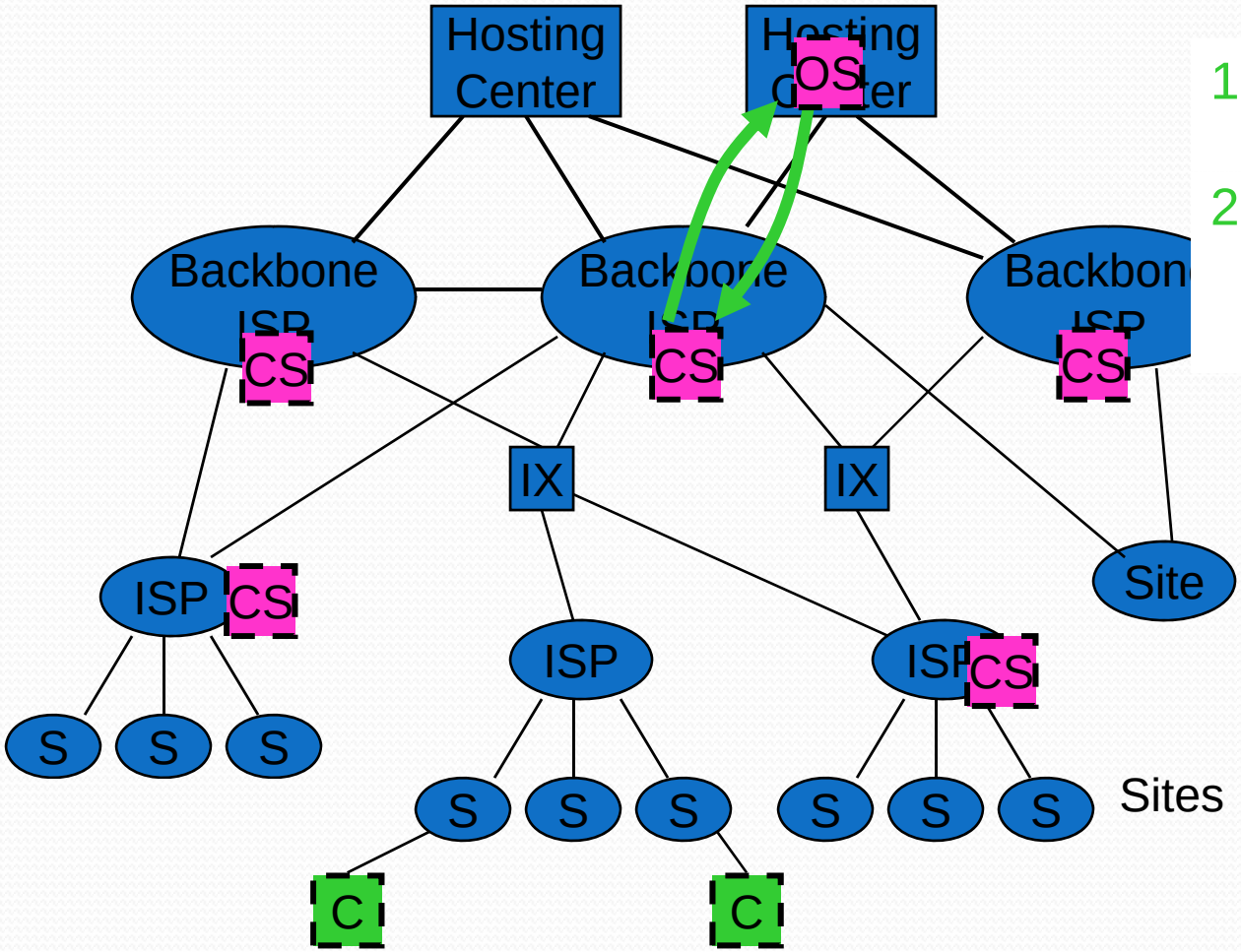


Cached CDN



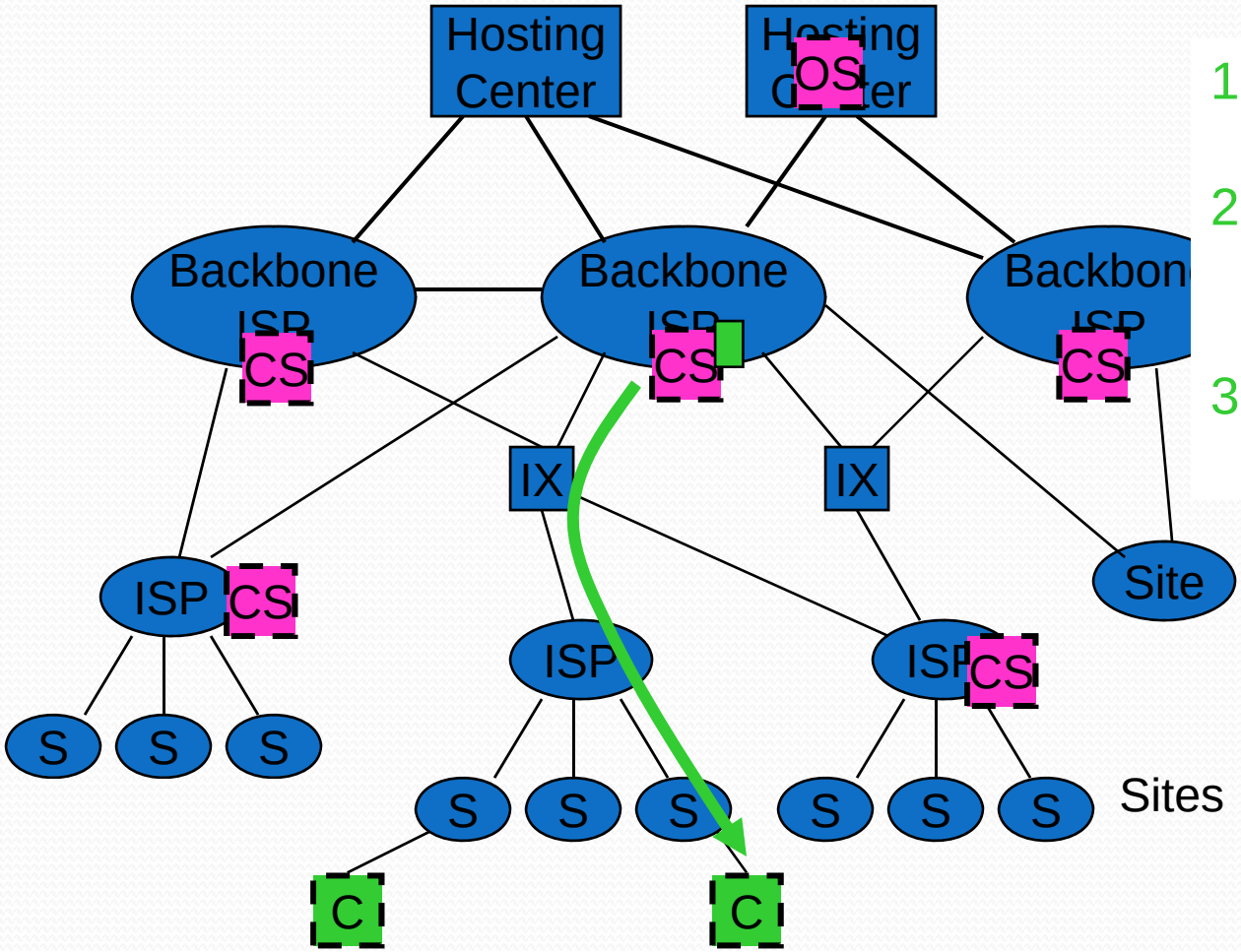
1. Client requests content.

Cached CDN



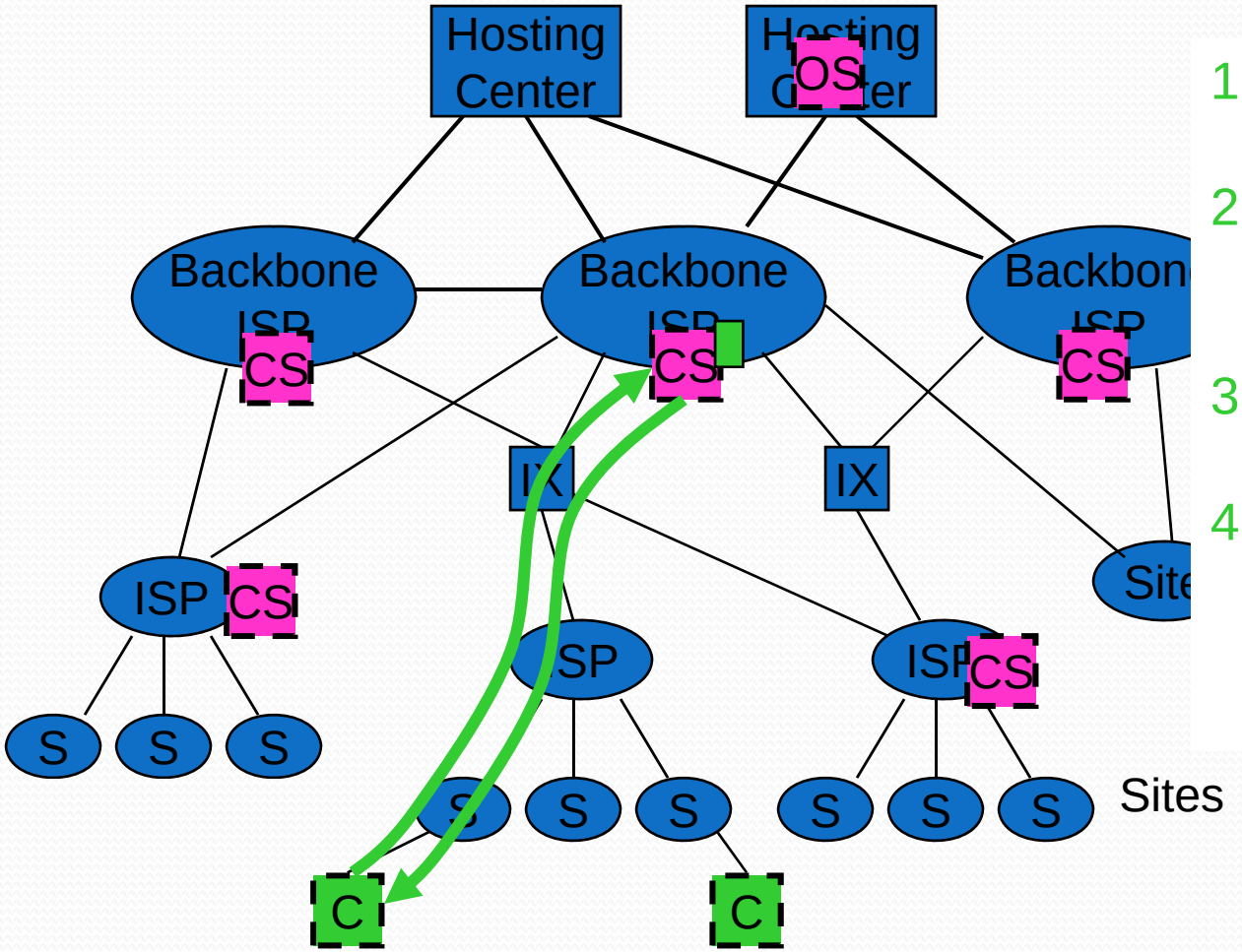
1. Client requests content.
2. CS checks cache, if miss gets content from origin server.

Cached CDN



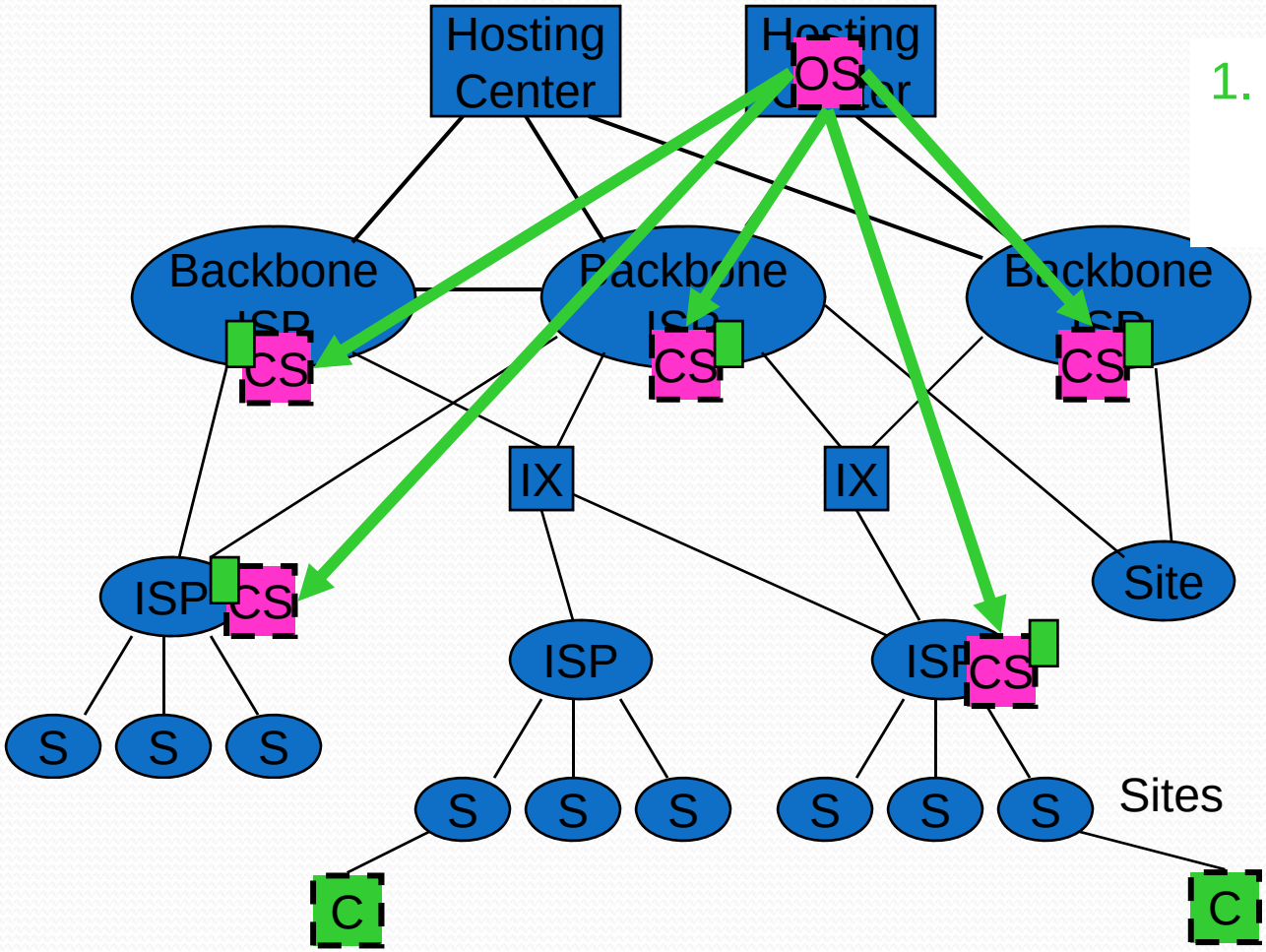
1. Client requests content.
2. CS checks cache, if miss gets content from origin server.
3. CS caches content, delivers to client.

Cached CDN



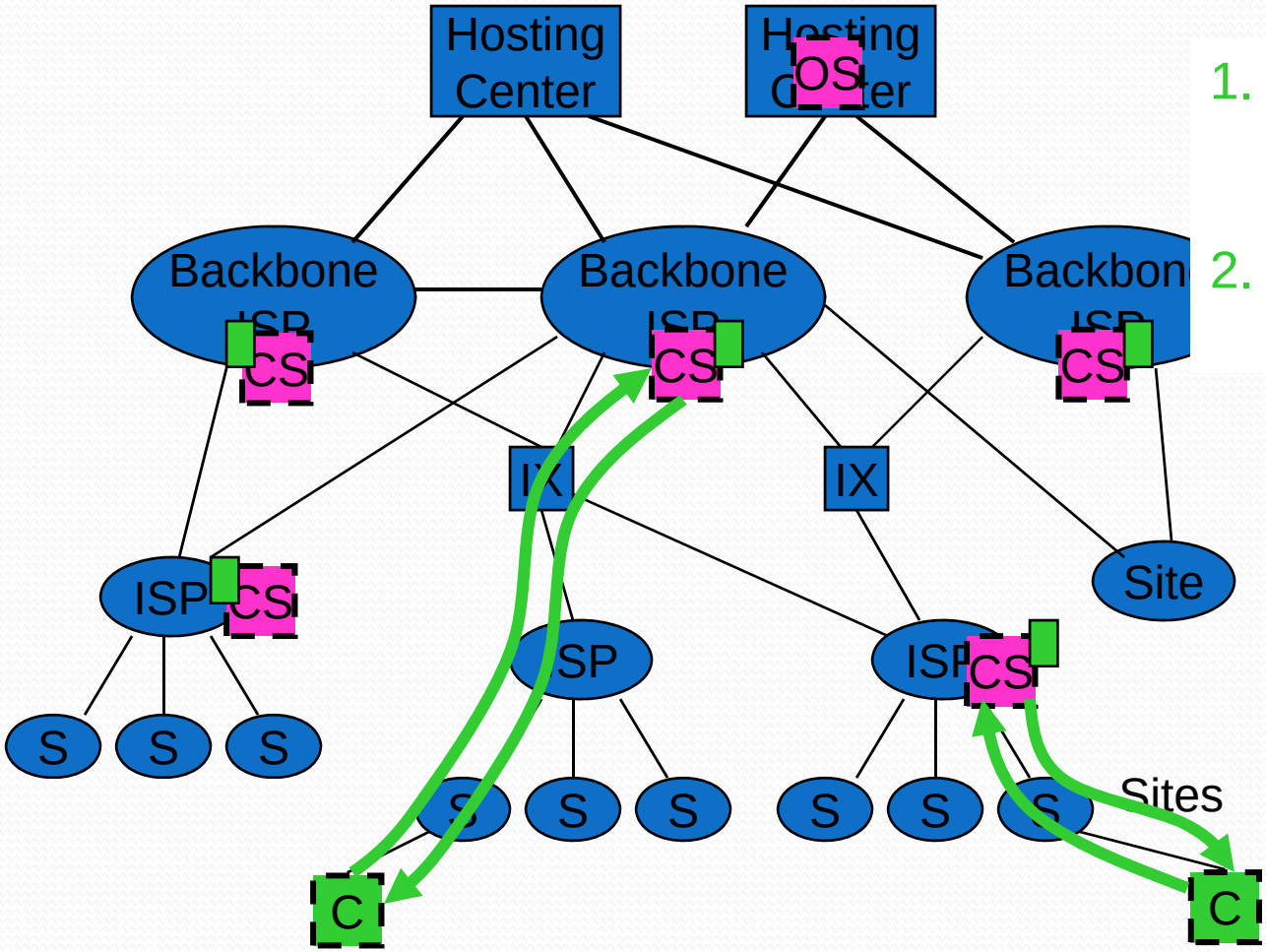
1. Client requests content.
2. CS checks cache, if miss gets content from origin server.
3. CS caches content, delivers to client.
4. Delivers content out of cache on subsequent requests.

Pushed CDN



1. Origin Server pushes content out to all CSs.

Pushed CDN

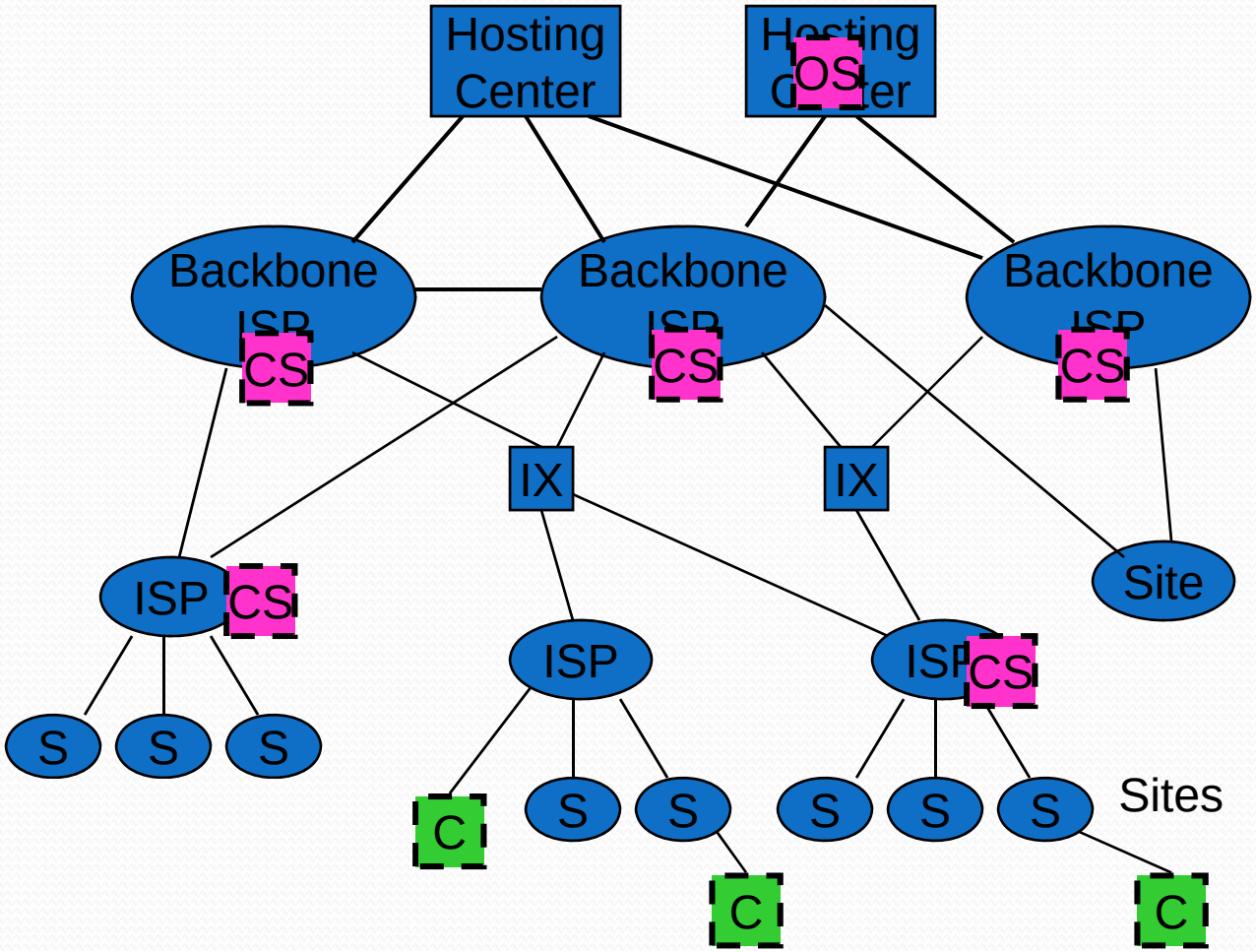


1. Origin Server pushes content out to all CSs.
2. Request served from CSs.

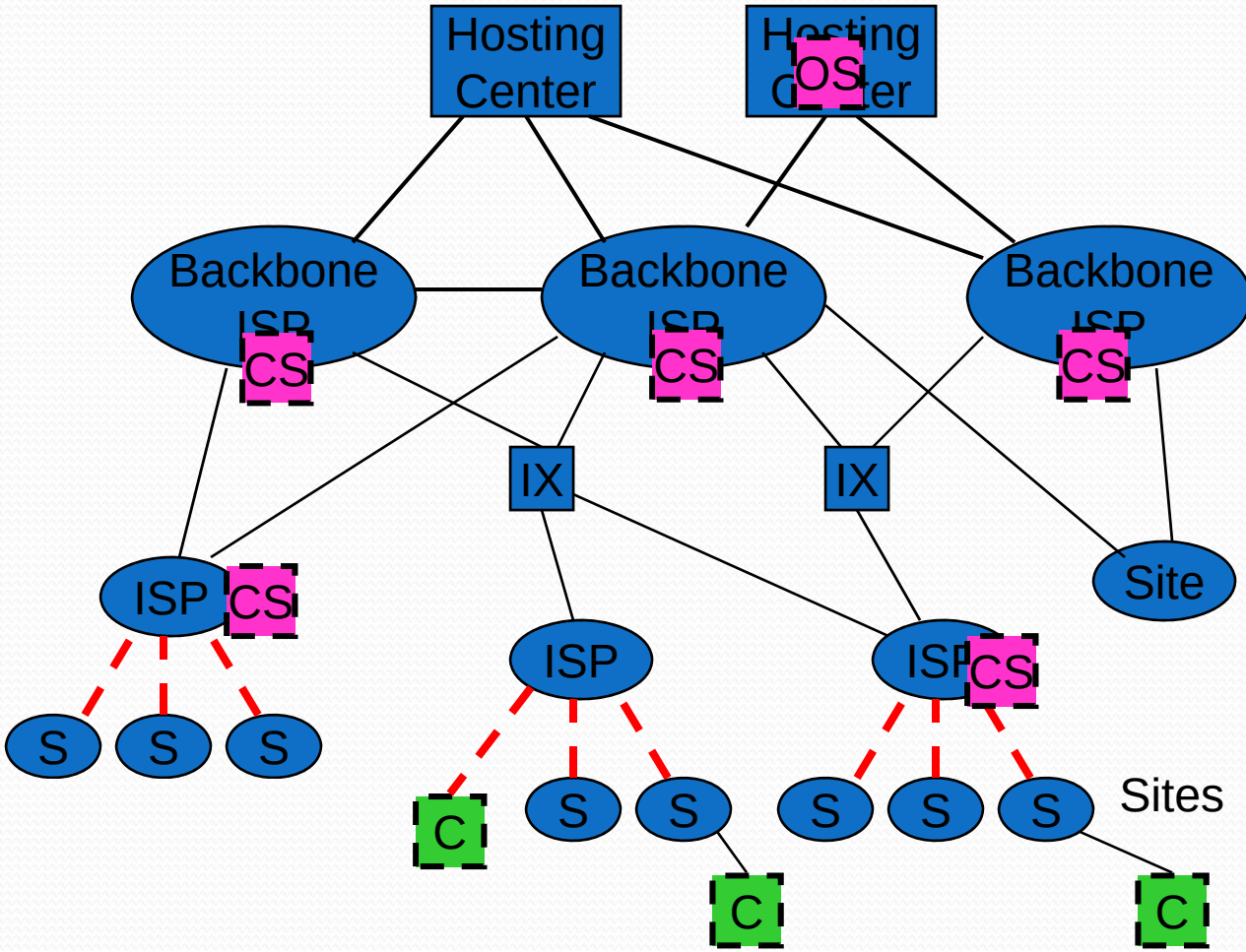
CDN benefits

- Content served closer to client
  - Less latency, better performance
- Load spread over multiple distributed CSs
  - More robust (to ISP failure as well as other failures)
  - Handle flashes better (load spread over ISPs)
  - *But well-connected, replicated Hosting Centers can do this too*

How well do CDNs work?



How well do CDNs work?



Recall that the bottleneck links are at the edges.

Even if CSs are pushed towards the edge, they are still behind the bottleneck link!

Reduced latency can improve TCP performance

- DNS round trip
- TCP handshake (2 round trips)
- Slow-start
  - ~8 round trips to fill DSL pipe
  - total 128K bytes
    - Compare to 56 Kbytes for cnn.com home page
    - Download finished before slow-start completes
- Total 11 round trips
- Coast-to-coast propagation delay is about 15 ms
  - Measured RTT last night was 50ms
    - No difference between west coast and Cornell!
- 30 ms improvement in RTT means 330 ms total improvement
  - Certainly noticeable

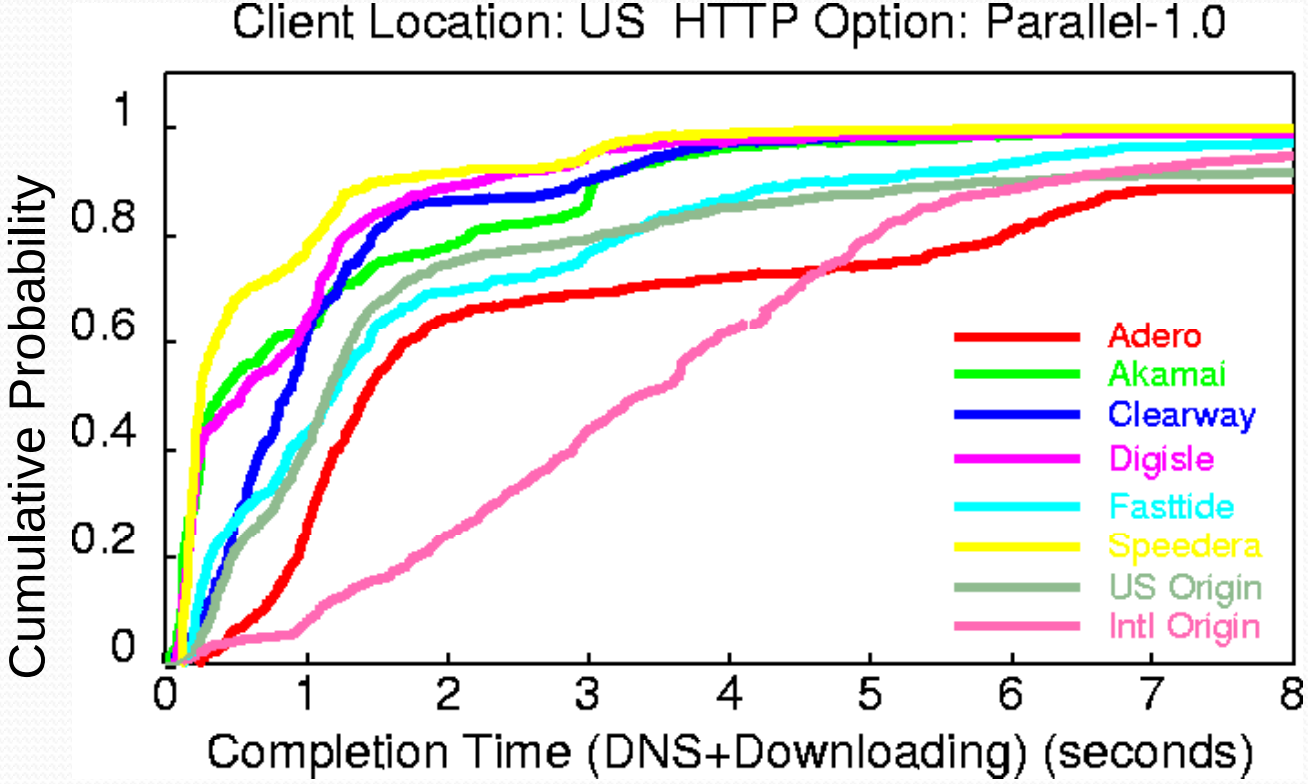
Lets look at a study

- Zhang, Krishnamurthy and Wills
  - AT&T Labs
- Traces taken in Sept. 2000 and Jan. 2001
- Compared CDNs with each other
- Compared CDNs against non-CDN

Methodology

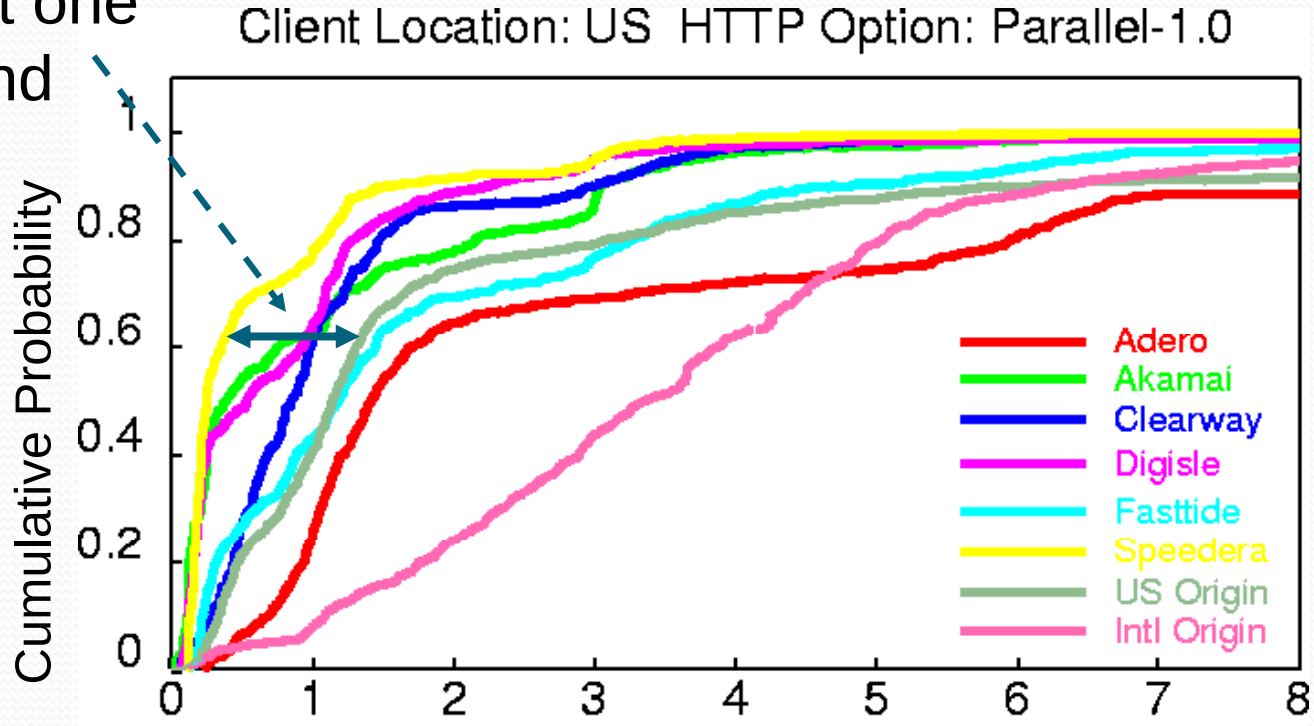
- Selected a bunch of CDNs
  - Akamai, Speedera, Digital Island
    - Note, most of these gone now!
- Selected a number of non-CDN sites for which good performance could be expected
  - U.S. and international origin
  - U.S.: Amazon, Bloomberg, CNN, ESPN, MTV, NASA, Playboy, Sony, Yahoo
- Selected a set of images of comparable size for each CDN and non-CDN site
  - Compare apples to apples
- Downloaded images from 24 NIMI machines

Response Time Results (II) Including DNS Lookup Time



Response Time Results (II) Including DNS Lookup Time

About one second

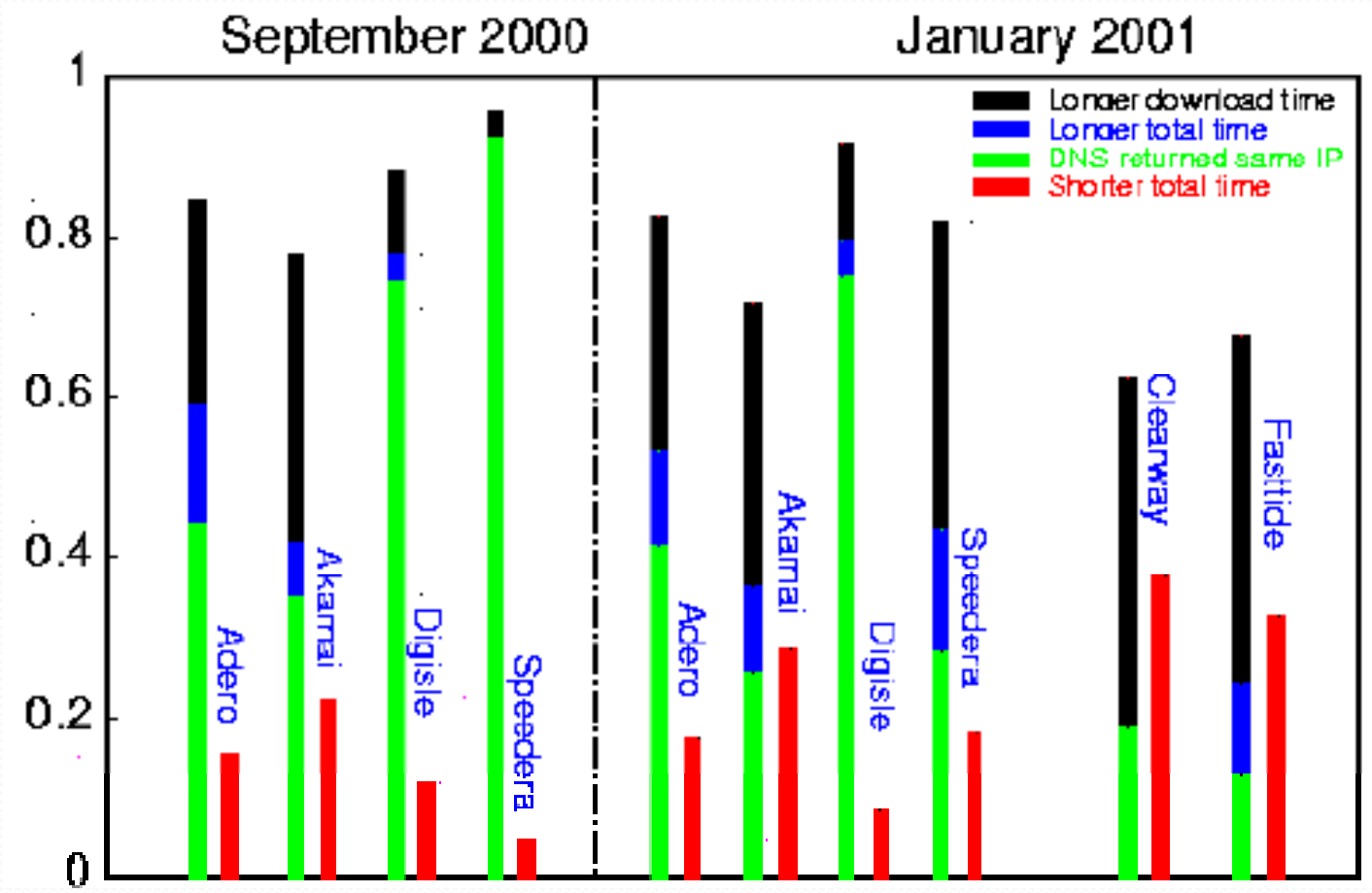


Author conclusion: CDNs generally provide much shorter download time.

CDNs out-performed non-CDNs

- Why is this?
- Lets consider ability to pick good content servers…
- They compared time to download with a fixed IP address versus the IP address dynamically selected by the CDN for each download
  - Recall: short DNS TTLs

Effectiveness of DNS load balancing



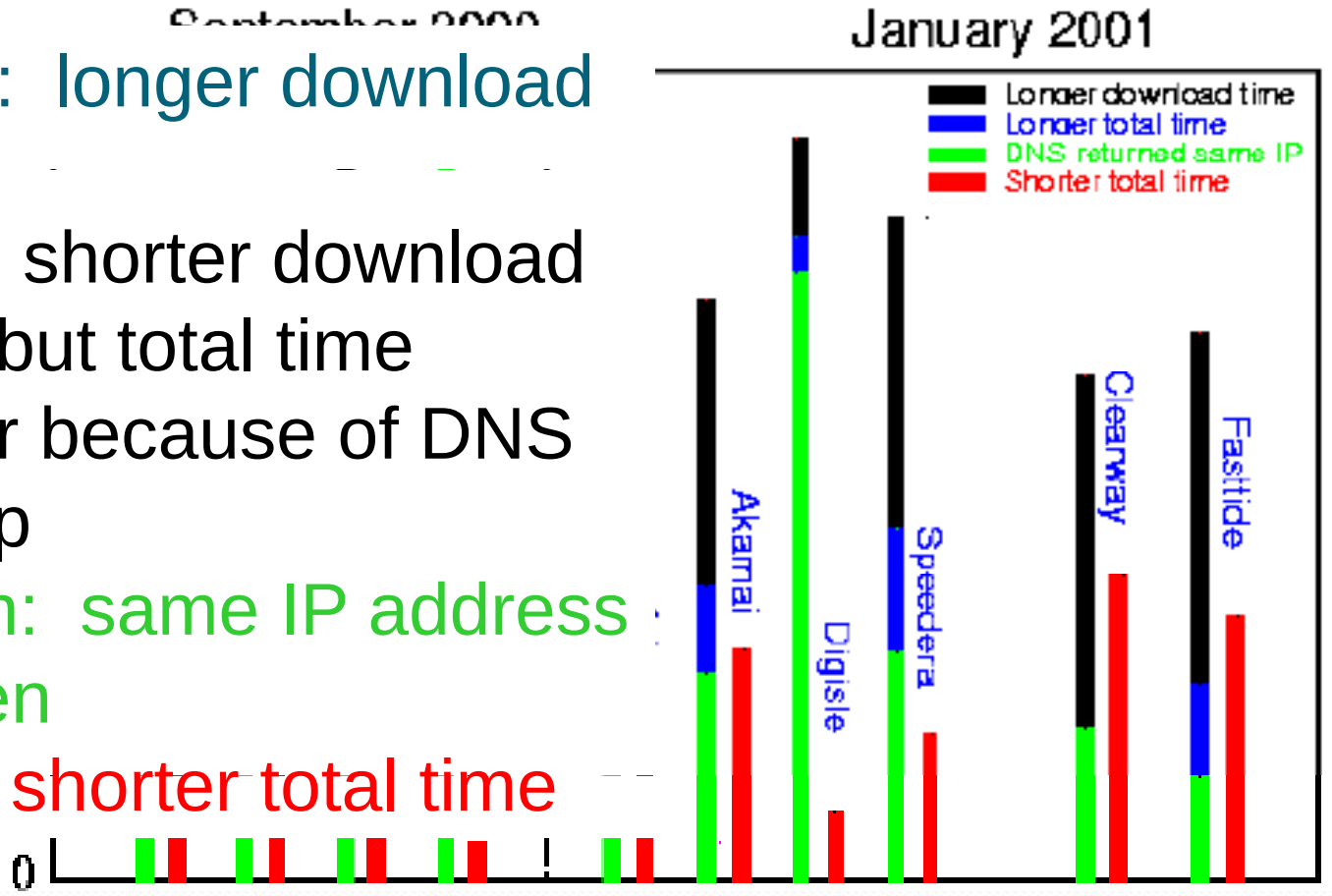
Effectiveness of DNS load balancing

Black: longer download time

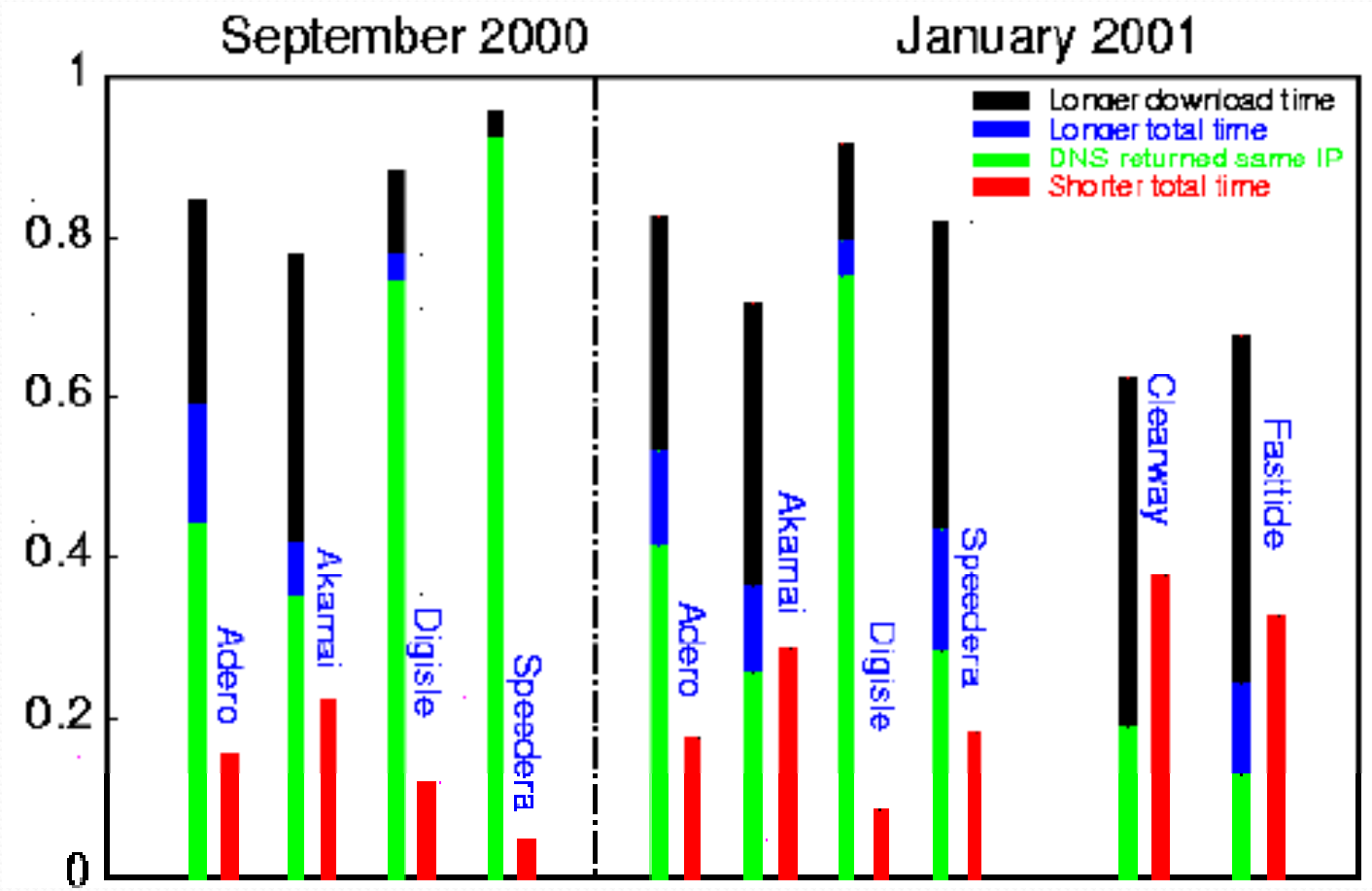
Blue: shorter download time, but total time longer because of DNS lookup

Green: same IP address chosen

Red: shorter total time



DNS load balancing not very effective



Other findings of study

- Each CDN performed best for at least one (NIMI) client
  - Why? Because of proximity?
- The best origin sites were better than the worst CDNs
- CDNs with more servers don't necessarily perform better
  - Note that they don't know load on servers…
- HTTP 1.1 improvements (parallel download, pipelined download) help a lot
  - Even more so for origin (non-CDN) cases
  - Note not all origin sites implement pipelining

Ultimately a frustrating study

- Never actually says *why* CDNs perform better, only that they do
- For all we know, maybe it is because CDNs threw more money at the problem
  - More server capacity and bandwidth relative to load

Back to web services

- We've seen that
  - They embody a lot of standards, for good reasons
  - Talking to Amazon.com is far more complex than just connecting one computer to another: many levels of choices and many services are ultimately involved
  - Even serving relatively static content entails remarkably complex and diverse infrastructure. True services do much more than just hand out copies of files!

Relating to CS5140 themes

- We'll look more closely at some of the major components of today's most successful data centers
- But rather than limiting ourselves to superficial structure, we'll ask how things work on the inside
  - For example, how does Google's Map/Reduce work?
  - It resides on a cluster management platform. How does that work?
  - At the core, locking and synchronization mechanisms. How do these work?

Trustworthy web services

- A preoccupation of many today, and a Cornell specialty
  - Not only do we want this complex infrastructure to work, but we ALSO want it to…
  - … be secure, and protect private data
  - … give correct answers, and maintain availability
  - … be hard to disrupt or attack
  - … defend itself against spoofing, pfishing, etc
  - … be efficient to manage and cost-effective
- Existing platforms don't satisfy these goals!

Next week

- Services found in cloud computing systems and other SOA environments
  - There are lots of ways to build them… some more effective than others
  - Today we looked at standards… but standards don't extend to telling us how to build the services we need
- We'll spend a full lecture on Map/Reduce
  - Recommend that you read the OSDI paper about this platform
  - Map/Reduce will be a focus of assignment one