Web Traffic Analysis

IEC Workshop 2000

Geoffrey M. Voelker
UCSD CSE

June 30, 2000

Seminar Motivation

- Why do Web traffic analysis?
- If you're interested in
  - Web performance (browser, cache, server, network)
  - Web architecture
  - Web applications and services
- You will perform some kind of Web traffic analysis at some point
  - Necessary for making design choices for your system/application/widget
  - Understanding its behavior
  - Analyzing its performance

Seminar Goals

- Familiarity with Web architectural components
- Types and formats of traces at the various components
- Kinds of analyses the traces enable
  - What can you do with browser, cache, and/or server traces?
- Lessons if you're doing own tracing projects
  - Learned the hard way
- Pointers to software for doing Web traffic analysis
- Lab experience generating requests, examining and processing logs
  - Admittedly brief, but the first place to start

Seminar Overview

- Introduction
- Architecture
  - Browsers, caches, servers, CDNs, etc.
  - HTTP protocol
- Traces and Analyses
  - Using browser, cache, server, and/or network packet traces
- Tools
  - Making requests, log scripts, workload generators
- Lab
  - Tutorial exercises
- Should not take all day (even if indicated by schedule)

An Aside

- Level of material
  - Tried not to make too many assumptions
  - Some material will be familiar already
  - Let me know if I cover material you already know
- Lecturing vs. discussion
  - Prefer discussions
  - Interrupt me with questions or comments at any time
- Coverage of material
  - If you have questions about Web-related topics that I do not cover, again, do not hesitate to ask

Personal Experiences

- How did I get into this area?
- We were interested in cooperative caching algorithms
  - Using multiple distributed caches to make more efficient use of cache resources to increase performance
  - Extensive research on LANs, wanted to move to WANs
- Key necessary characteristic
  - Sharing behavior among groups of clients
- Problem
  - No traces had been taken identifying user subgroups
  - No simultaneous traces available for multiple sites
- So we decided to do our own tracing project…

UW Tracing Project

- Traced all Web traffic crossing UW's border routers
  - Significant user population of 50,000
  - Started at 40 Mbit/s peak, ended at 60-70 Mbit/s
- Approach
  - Passive network monitoring
    - » Monitoring ports from four switches fed into trace machine
  - Traces collected onto disk, analyzed offline
- Novelty was organization information
  - Tagged requests as coming from organizations
    - » CSE, English, Drama, dorms, modem pool, etc.
  - Mechanism for investigating group behaviors
    - » Sharing within, across groups

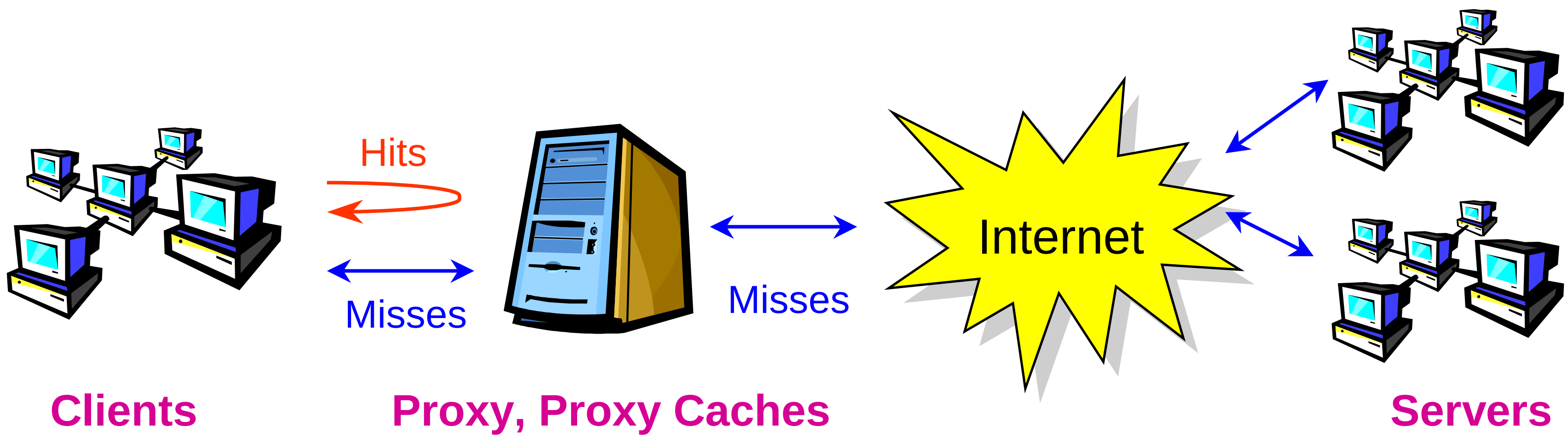
UW Tracing Project

- Requirements
  - Passive monitoring
  - Privacy: Summarize and anonymize all data saved to disk
- We built from scratch
  - Wrote all of our own tracing software
  - Wrote all of our own analysis software
  - Both required significant investments in time
- Learned a number of lessons
  - Did not know what we were getting into
  - Will go into network packet tracing for the Web in detail later
- Initiated into Web traffic analysis…

Part I: Architecture

- Components
  - Browsers, servers, caches, reverse caches, content delivery networks (CDNs), etc.
  - Will assume basic knowledge of Internet/IP infrastructure
  - Stop me if I make too many assumptions, though
- HTTP Protocol
  - Requests and responses
  - Header formats, fields
  - Cache control headers
  - Persistent connections

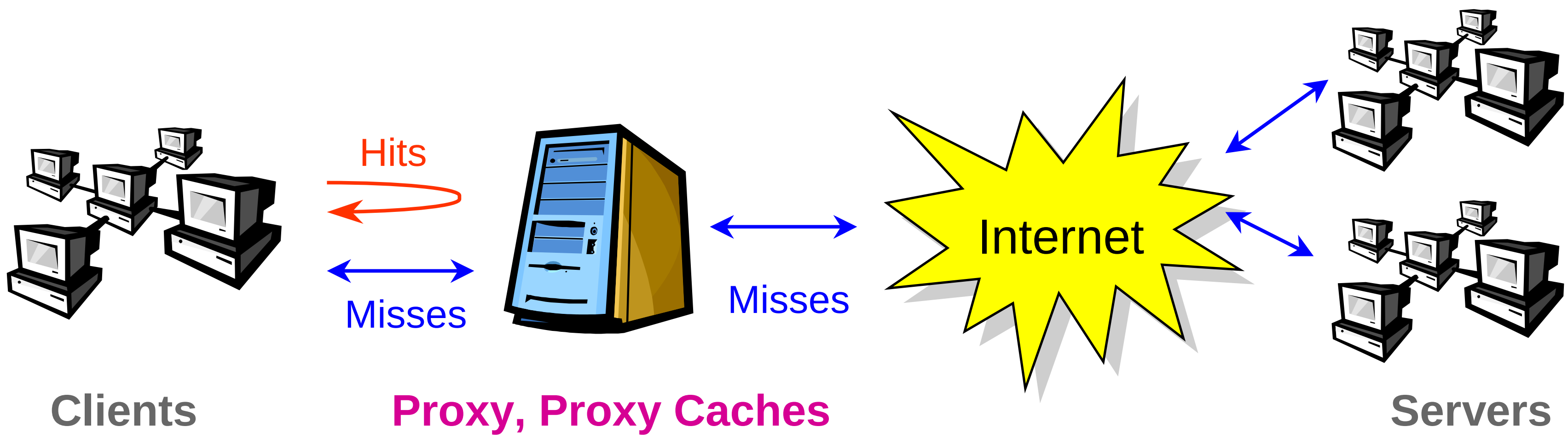
Big Picture


Hits
Misses
Misses
Internet
Clients
Proxy, Proxy Caches
Servers

Clients

- Originate the Web traffic in public Internet
  - Roughly, two classes of clients: browsers and robots
- Browsers operated by users: Netscape, IE, etc.
  - Request rate/workload limited to user
  - Utilize memory and disk caches
    - » Exploit temporal locality of an individual user
    - » Repeated requests to the same object when navigating a site
- Robots, crawlers, other programs
  - Request rate/workload limited by computation, network speed
  - Show up as outliers in analyses
  - Can skew results, need to be aware of them in traces

Proxies


Hits
Misses
Misses
Internet
Clients
Proxy, Proxy Caches
Servers

Proxies (Firewalls, Caches)

- Web proxies serve a client population
  - Often part of the enterprise firewall mechanism
  - Now, almost all are caching proxies
- A proxy cache handles requests on behalf of clients
  - Request sent from browser to cache
  - Cache returns object if stored locally and up to date
    - » Based on URL, ETag, TTL-related fields
  - Otherwise cache forwards request to server
  - If out of date, cache validates object
- Excellent resource on caches
  - Information Resource Caching FAQ by Duane Wessels

    `http://www.ircache.net/Cache/FAQ/ircache-faq.html`

Cache Benefits

- Proxy caches exploit locality among a group of clients
- Caches benefit clients, servers, and network
  - Bandwidth: Reduce network utilization
    - » Original goal of caches
    - » Especially useful in bandwidth-constrained environments (Europe, international links)
  - Latency: Reduce response time
    - » Closer the cache is to the clients, faster the response time
  - Server load: Offload requests onto caches
- Empirically, large caches experience a 50% hit rate, 40% byte hit rate

Explicit/Transparent Caches

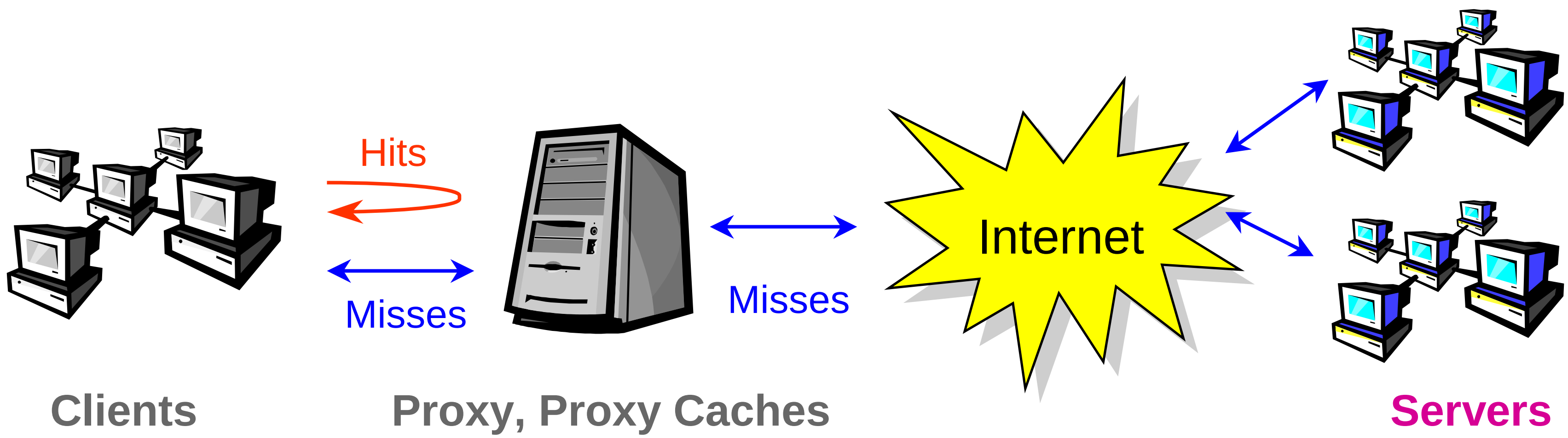
- Explicit: Browsers explicitly configured to use cache
  - Bit of administrative overhead
  - Difficult to enforce use of cache (e.g., university environment)
  - Mechanisms for browsers to do cache discovery (c.f., DHCP)
  - UW has caches, but no one uses them…
- Transparent: Caches monitor network streams, automatically intercept HTTP requests
  - Router vendors entering this market; e.g., Cisco
  - No mechanisms for user to avoid cache

Coordinated Caches

- Cluster caches
  - Cluster of machines that looks like a single logical cache to clients, servers (e.g., Inktomi products)
  - Used to scale cache performance with workload
  - Sometimes also called "cooperative", but different than below
- Cooperative caches
  - Collection of distributed caches across network
  - Used to exploit combined cache contents, resources
    - » More clients you have, the better the sharing, locality
  - NLANR Squid cache hierarchy is best example

    http://ircache.nlanr.net/
  - Increases complexity of system

Servers

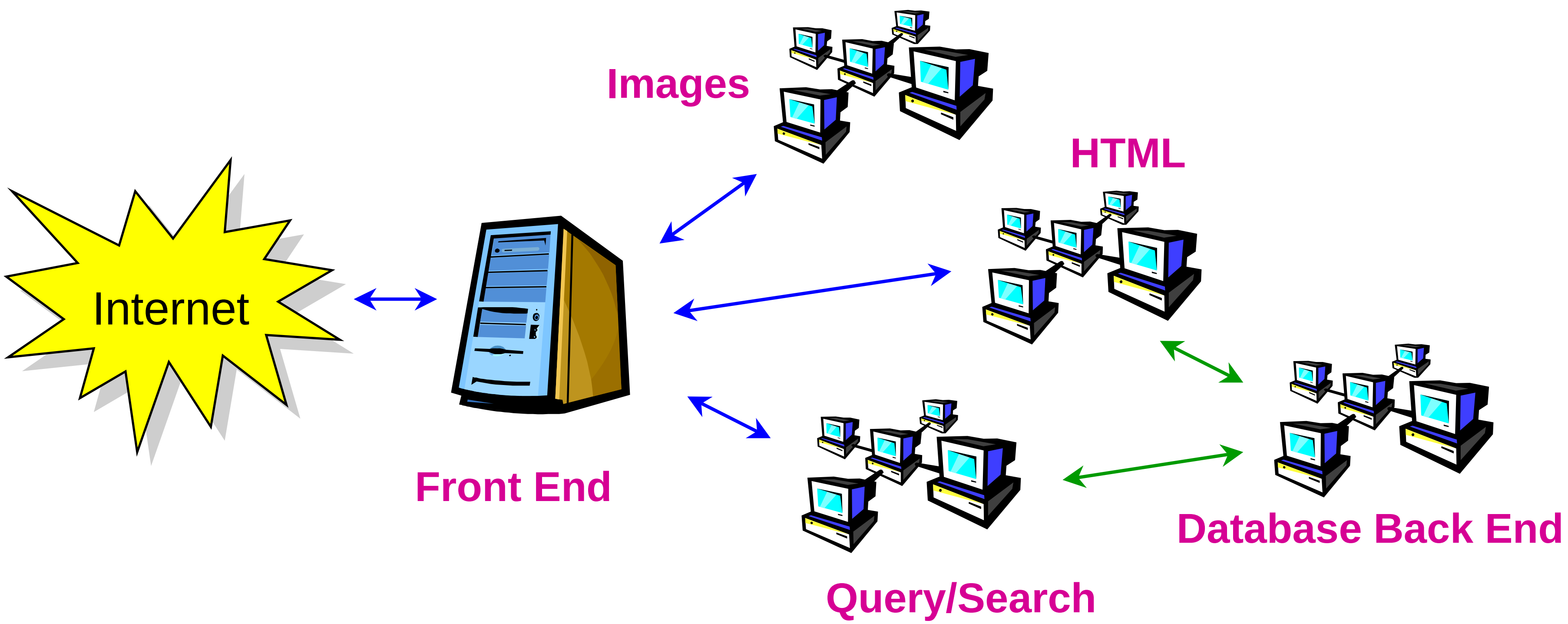


Hits
Misses
Misses
Internet
Clients
Proxy, Proxy Caches
Servers

Servers

- Basic job is straightfoward
  - Connect with client
  - Receive request
  - Parse
  - Locate and return result
- High performance complicates matters
  - Multiple processes, threads to handle connection load
  - Computation: CGI scripts, servlets
  - Caching
- Apache is the most popular open source server
  - http://www.apache.org

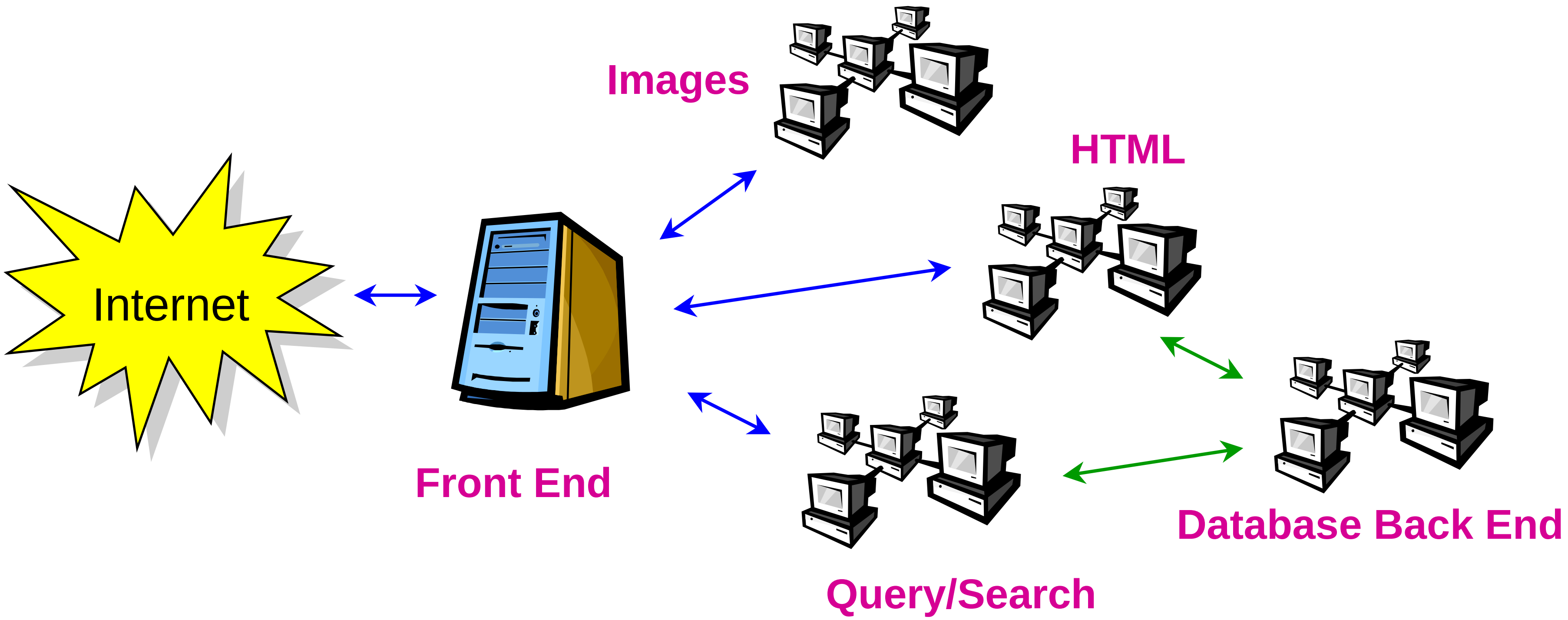
Complex Servers



Complex Servers

- Popular sites have large server farms (e.g., Excite)
  - Clusters of machines
  - Roughly structured in tiers
- Front end
  - Server cache, request router, load balance
- Content servers
  - Sometimes differentiated according to request type (HTML, image, query, search)
- Database back ends
  - Raw data (e.g., sports scores)
  - User configuration information (personalized home pages)

Front End Services



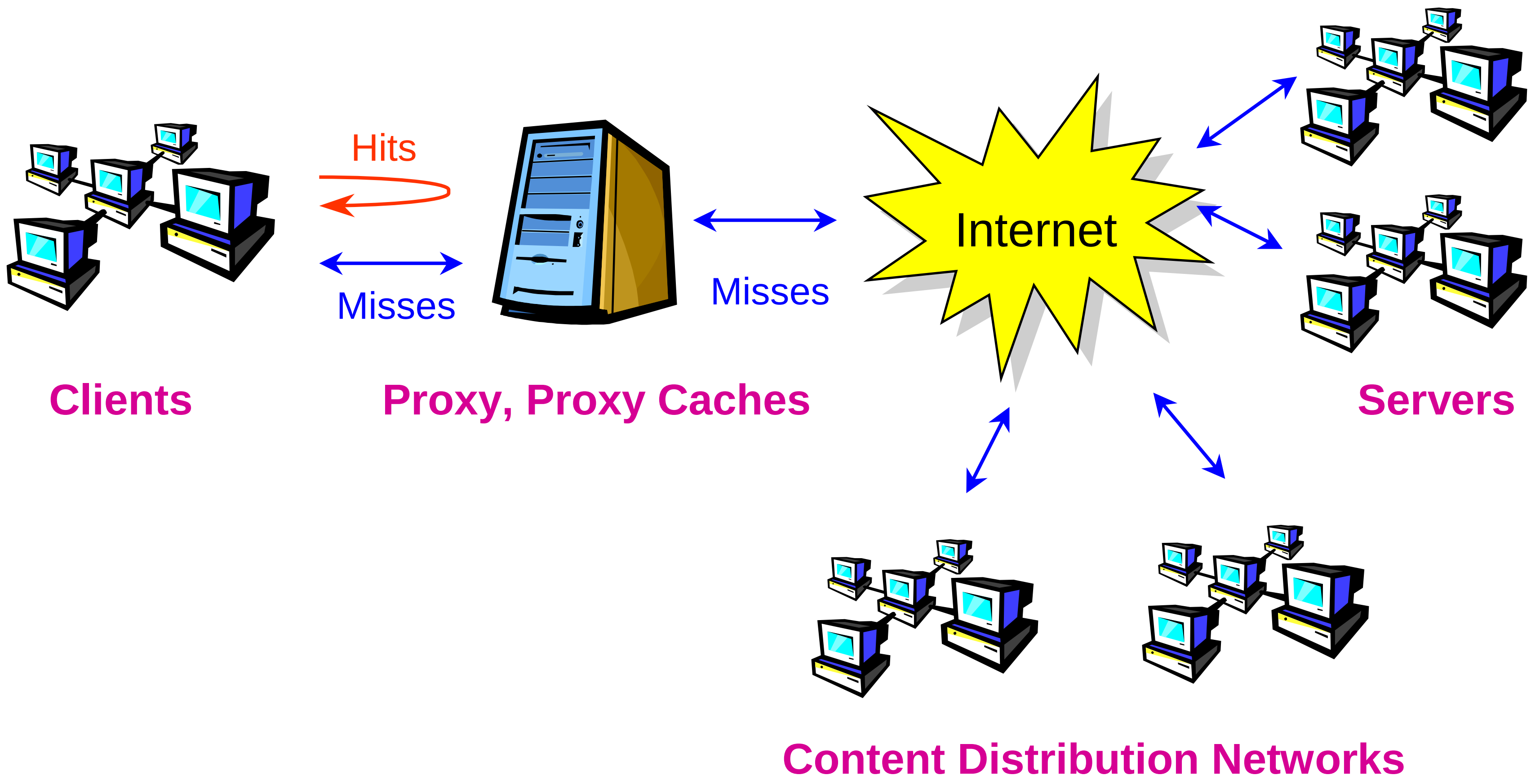
Request Routing

- IP switching (e.g., Cisco)
  - Load balancing (e.g., round robin across set of servers)
  - Alternative to DNS round robin
- Web switching
  - Load balancing
  - Content routing (e.g., HTML, image, query, etc.)
  - Based upon HTTP request (e.g., suffix)
    - » Peeking into application layer

Reverse Caches

- Reverse caches (aka server caches) intercept requests to servers
  - Offload server by exploiting locality of requests to server
    - » Side Q: Why haven't client caches removed this locality?
  - Offload server by caching results of expensive operations
    - » Search queries
  - Route requests to appropriate servers
    - » Differentiated content (HTML, image, query, etc.)
    - » Load balance

Content Distribution Networks



Content Distribution Networks (CDNs)

- CDNs host content on behalf of content providers
  - Content providers usually located in one place in network
  - CDNs have servers distributed throughout network
- CDNs employ their own overlay network
  - Servers at the edges, close to clients
    - » Previous picture is misleading, servers everywhere
  - CDNs redistribute content among servers according to popularity, demand, load, etc.
  - CDNs try to route client requests to “best” content server

Advantages of CDNs

- Advantages
  - Lower latency: Can potentially locate data closer to clients
  - Availability: Data hosted on multiple servers accessible via multiple networks
  - New services: Hit counting, invalidation, dynamic data, etc.
    - » Leverage contractual agreement between provider and CDN
- Hot new area
  - Rush of new companies to carve up the market
    - » Akamai, Sandpiper (Digital Island), etc.
  - Recent International Web Caching Workshop dominated by CDN topics and people

CDN Request Routing

- How does Akamai route requests?
- DNS hacks
  - Embedded URLs rewritten to point to Akamai DNS servers
    - » http://espn.go.com
      - http://a12.g.akamaitech.net/7/12/621/000/espn.go.com/i/h.gif
      - http://a12.g.akamaitech.net/7/12/621/000/espn.go.com/i/vc.gif
      - ....
- Routing decision made when server name resolved
  - Return IP address of "closest" content server
  - Also load balance, availability
    - » "Consistent Hashing" (not clear it is used anymore)
  - Use client IP address, route, BGP tables as input

DNS Hack Implications

- Does not handle top-level HTML page
  - Akamai serves images only (for now)
- Interaction with caches
  - Cache does name resolution for clients
  - Akamai sees the IP address of the cache, not clients Individual download time for images may be smaller
- Increases initial RTTs
  - Not clear how overall downtime affected
  - Further motivation for studying Web performance in terms of pages instead of objects

HTTP Protocol

- Communication protocol among browsers, caches, and servers
  - Application-level protocol (typically on top of TCP/IP)
  - Request/response interaction (RPC-like)
- We will cover
  - Request and response formats
  - Semantics of various header fields
  - Protocol aspects most relevant to tracing and analysis
    - » Persistent connections
    - » Cache control directives

Packet Format

- Packets composed of a header and body
  - In addition to any transport headers (TCP/IP)
  - Syntax and semantics defined by RFCs
    - » HTTP 1.0: http://www.w3.org/Protocols/rfc1945/rfc1945.txt
    - » HTTP 1.1: http://www.w3.org/Protocols/rfc2616/rfc2616.html
- Header
  - Sequence of fields terminated by CRLF CRLF
  - Fields encoded as ASCII strings terminated by CRLF
- Body
  - Presence depends upon request and result
  - Content determined by object type
  - Identified 712 content types in UW trace

Header Fields

- RFCs define supported protocol fields
  - Content-Length, Date, Last-Modified, etc.
  - 47 defined fields in HTTP 1.1 specification
- Additional fields can be used arbitrarily
  - Extensibility mechanism
  - Unknown fields ignored by clients, caches, servers
  - No name space management, though
  - Identified 518 different fields in UW traces

Request Format

- Generic format:

```
Request Line
Headers
Body
```

- Request line:

```
Method Request-URI HTTP-Version
```

  - Method
    - » GET (read), PUT (write), HEAD (attr), DELETE (delete), etc.
  - Request-URI
    - » URI of object
  - HTTP-Version
    - » HTTP/1.0, HTTP/1.1

Request Example

- In Netscape 4.7, the URL:

  `http://localhost:5000/example.html`

- Generates the following request:

```
GET /example.html HTTP/1.0
Connection: Keep-Alive
User-Agent: Mozilla/4.72 [en] (WinNT; I)
Host: localhost:5000
Accept: image/gif, image/x-xbitmap, image/jpeg,
  image/pjpeg, image/png, */*
Accept-Encoding: gzip
Accept-Language: en
Accept-Charset: iso-8859-1,*,utf-8
```

Proxy Requests

- Note that the Request Line specifies the URL path rather than the full URL

  ```
  GET /example.html HTTP/1.0
  ```

- HTTP 1.0 did not require the Host field

  ```
  Host: localhost:5000
  ```

- Without Host field, proxy cannot determine endpoint
- Solution: Proxy Requests, which specify full URL

  ```
  GET http://localhost:5000/example.html HTTP/1.0
  ```

  - Browsers explicitly configured to use caches
  - Also switch to use Proxy Requests
- What about transparent caches?

Response Format

- Generic format:

```
Status Line
Headers
Body
```

- Status line:

```
HTTP-Version Status-Code Reason-Phrase
```

  - HTTP-Version same as Request Line
  - Status-Code
    - » Informational (1xx), success (2xx), redirection (3xx), client error (4xx), server error (5xx)
    - » 200 (OK), 304 (Not Modified), 404 (Not Found), etc.
  - Reason-Phrase
    - » OK, Not Modified, Not Found, etc.

Request Example

`http://espn.go.com:80/` generates:

```
HTTP/1.1 200 OK
Server: Microsoft-IIS/4.0
Cache-Control: max-age=300
Expires: Fri, 30 Jun 2000 00:44:02 GMT
Content-Location: http://espn.go.com/index.html
Set-Cookie: SWID=EE796C81-4E1C-11D4-9ED1-090279A9290;
    path=/; expires=Fri, 30-Jun-2020 00:39:02 GMT;
    domain=.go.com;
Date: Fri, 30 Jun 2000 00:39:02 GMT
Content-Type: text/html
Accept-Ranges: bytes
Last-Modified: Fri, 30 Jun 2000 00:31:45 GMT
ETag: "5c6dfe912ae2bf1:2cc1"
Content-Length: 36812
```

HTTP 1.1

- HTTP 1.1 adds a number of features to HTTP 1.0
- Most prominent (in terms of tracing, analysis)
  - Persistent connections
  - Better support for caches

Connection Management

- HTTP 1.0
  - Creates and closes a connection per request/response
  - Not quite true (ad-hoc Connection: Keep-Alive header)
- Disadvantages
  - Increases overhead for multiple requests to same server
    - » Connection establishment (handshake) for every object
    - » TCP slow-start
- Advantages
  - Easy to implement

Persistent Connections

- HTTP 1.1 introduced persistent connections
  - Clients maintain open connections with caches, servers
  - Send multiple requests across connection
  - Pipeline requests and responses
- Advantages
  - Reduce # of connections and associated delays, memory, and CPU resources
  - Keep TCP congestion window open
- Disadvantages
  - Much more complex than HTTP 1.0 model

Persistent Connection Issues

- Connection management
  - Open connections a finite resource
  - Caches, servers must time-out an open connection
  - Which timeout value to use?
- Responses ordered according to requests (FIFO)
  - Cache implications: Head-of-line blocking

Cache Control Headers

- Need to ensure that cached contents are equivalent to what is stored on server
- Mechanisms in HTTP 1.0 were very ad-hoc
- HTTP 1.1 added mechanisms to enable caches to be more consistent with servers
- Two models
  - Expiration
    - » Reduces requests to servers
  - Validation
    - » Reduces amount of data that has to be transferred
- Directives specified via “Cache-Control” header

Expiration

- Determine if object “age” is older than “freshness”
- Key headers
  - Expires: Expiration time
  - Date: Time server generated response
  - Age: Duration object has been stored in caches
  - Cache-Control: max-age: Object lifetime outside server
- Key times
  - Request time, response time, “now”
- Determine freshness
  - Based upon header values and times
  - If not fresh, need to communicate with server

Validation

- Ideally, only want object contents if it has changed
- Use conditional requests: If-Modified-Since
- Key headers
  - Last-Modified: Timestamp of object on server
  - ETag (entity tag): Signature of object on server
    - » Counter, hash of contents, etc.
- Determining if a cache has the latest value
  - Last-Modified still the same
  - ETag equivalence

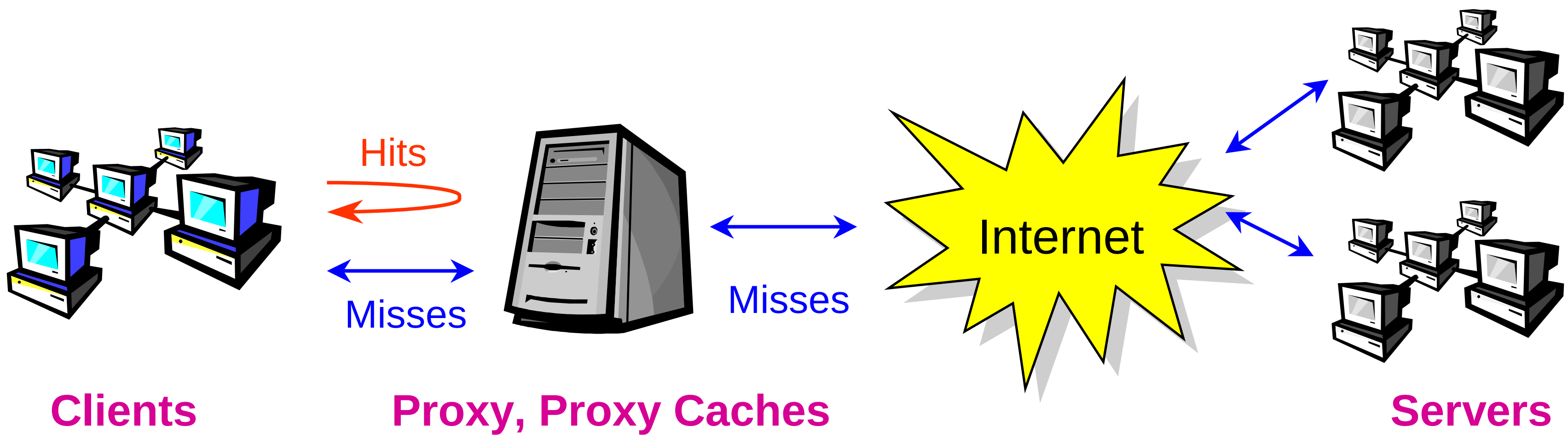
Part I: Summary

- Components
  - Browsers, servers, caches, reverse caches, content delivery networks (CDNs), etc.
- HTTP Protocol
  - Requests and responses
  - Header formats, fields
  - Persistent connections
  - Cache control headers
- Questions?

Part II: Traces and Analyses

- Survey various kinds of traces
  - Browser, proxy cache, server, CDNs, network packet traces
  - Traces at different places see different pieces of the picture
- Discuss
  - Advantages
  - Options, formats
  - Kinds of analyses
  - Limitations
- Lessons learned from UW tracing project
- Trace archives

Browsers


Hits
Misses
Misses
Internet
Clients
Proxy, Proxy Caches
Servers

Browser Traces

- Advantage: Witness all user events
  - In particular, user requests that are browser cache hits
- Options
  - Problem: Not easy
    - » External monitoring of network traffic not enough
  - Reconfiguration
    - » Zero disk/memory caches to force requests on network
  - Modification
    - » Early work at Boston Univ. modified open source Mosaic
  - Instrumentation
    - » Binary rewriting to record key events (not published)

Browser Trace Analyses

- Not many published analyses at browser (at least in systems/network literature)
  - Early Boston Univ. work stands out
- Will discuss unpublished analyses, potential analyses

Overall User Experience

- Complete round trip time for individual objects
- Estimate breakdowns of round trip times
  - Browser, cache, network, server contributions to delays
- Complete round trip time for entire web pages
  - Surprisingly, not much explored yet…

Browser Cache Performance

- Effectiveness of browser caches
  - Does it matter? It can (IE3)
- Replacement policies
  - Of course, disk resources not necessarily scarce
  - More applicable for browsers on PDAs

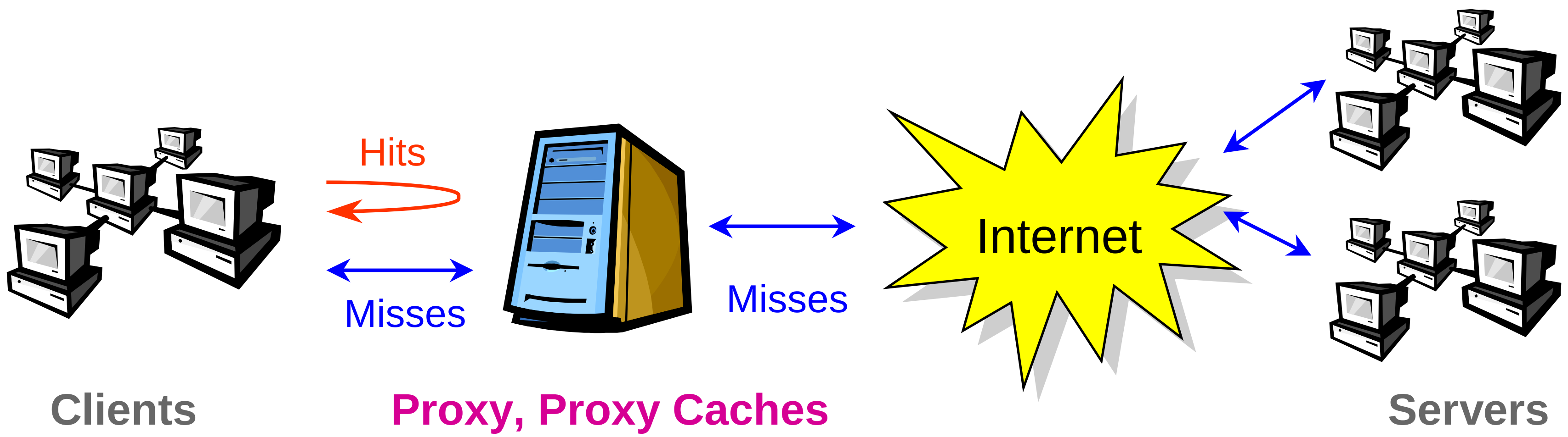
Browser Evaluation

- Browser display/rendering performance
  - E.g., Tables in Netscape can dominate total page time
- User interface studies
  - User navigation
  - Semantics to requests: Links, back, forth, reload

Browser Trace Limitations

- Tracing requires effort
  - No existing logging facility, no standard format
  - Modification
    - » Open source Netscape is a possibility (although there are issues)
  - Instrumentation
    - » Promising, but requires skill and cleverness
- Only witness behavior of a single user
  - To make generalizations, need to study many users

Proxies


Hits
Misses
Misses
Internet
Clients
Proxy, Proxy Caches
Servers

Proxy Cache Traces

- Advantages
  - Cross-section of clients and servers
  - Aggregate client behavior (e.g., sharing, object popularity)
  - Aggregate server usage (e.g., server popularity)
- Events recorded in logs
  - Easy to enable
  - Easy to analyze
    - » Logs record events at high level (e.g., server download time)
- Squid proxy cache logs
  - Open source
  - Most widely available and analyzed

Proxy Cache Trace Formats

- Common Logfile Format
  - Format used by multiple vendors, products
  - Same tools can be used on logs from different products
  - But, least common denominator
  - Postpone to server log discussion
- Squid formats
  - Specific to Squid
  - Record detailed state and behavior of Squid cache
- Vendor formats
  - Cache vendors also have their own formats
  - Not going to cover them

Squid Logs

- Squid records two key logs
  - access.log
    - » Records client accesses
    - » Useful for analyzing access behavior, request and object characteristics
  - store.log
    - » Records cache actions
    - » Useful for simulating cache behavior
- Documentation of log formats
  - `http://www.squid-cache.org/Doc/FAQ/FAQ-6.html`

Squid access.log

- General format:

```
time elapsed remotehost code/status bytes \
method URL rfc931 peerstatus/peerhost type
```

- Example entry:

```
962175640.444 210 69.133.208.39 TCP_MISS/200 367 \
POST http://http.pager.yahoo.com/notify/ - \
DIRECT/204.71.201.128 text/plain
```

access.log Example

time	962175640.444	Timestamp
elapsed	210	Service time
remotehost	69.133.208.39	Client (anon)
code/status	TCP_MISS/200	Cache result
bytes	367	Size
method	POST	Request Method
URL	http://http.pager.yahoo.com/notify/	
rfc931	-	Ident
peerstatus/peerhost	DIRECT/204.71.201.128	Origin Server
type	text/plain	Content-Type

Cache Trace Analyses

- Source of majority of Web traffic studies and analyses
  - Cross-section of clients and servers
- Request and object analyses
  - Basic distributions
    - » Object size, download latency, etc.
  - Parameters and trends well understood
    - » For static docs, at least

Cache Performance

- Hit rate, byte hit rate, bandwidth savings, latency reduction
- Replacement algorithms (in memory, on disk)
  - Popular research topic…
- Invalidation algorithms (explicit update on expiry, delta encoding)
  - Will see more work on this…
- Implementation
  - Interactions with file systems
  - Caches do a lot of lookups, reading, and writing

Client Behavior

- Sharing patterns
  - Temporal, popularity, hot spots
- Session behavior
  - Access sequence on site, across sites
- Effects of scaling client populations
  - Aggregate client behavior

New Cache Services

- Encryption, hit counting
- Prefetching
- “Active” caching
  - Computation environment (c.f., Active Networks)
  - Dynamic data

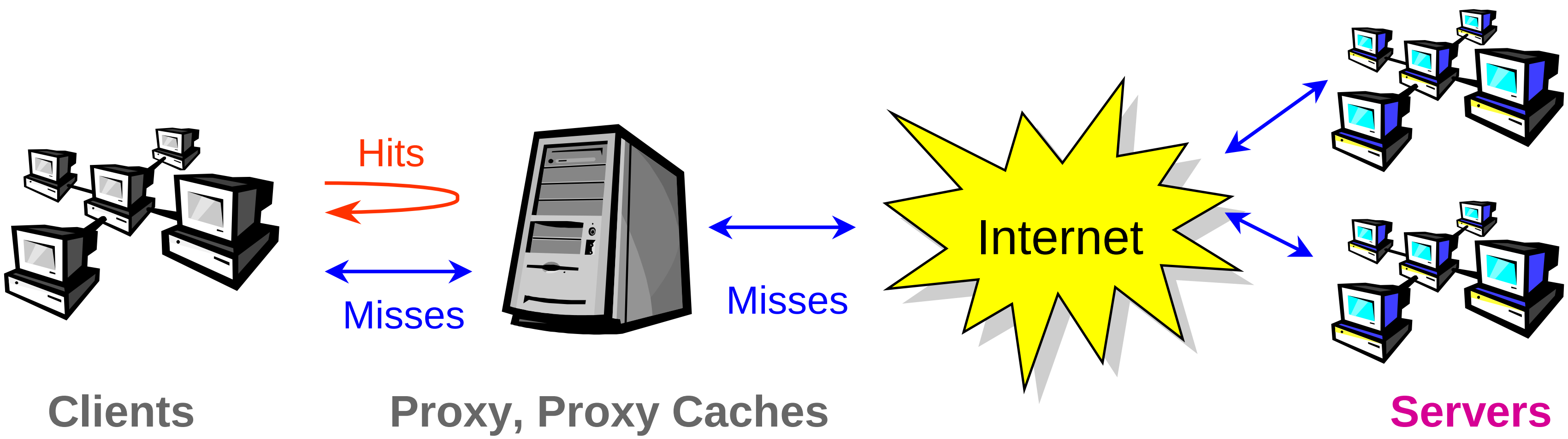
Coordinated Caches

- Cluster caches
  - Request routing (locality vs. load balancing)
  - Resource management (efficient use of memory, disk)
- Cooperative caches
  - Architecture (hierarchy, directory, hash-based, mesh, etc.)
  - Protocol (request routing, updates)
  - Utility (as a function of client population size)
  - Placement

Cache Trace Limitations

- No TCP information
  - Connection establishment, close
  - Delay for opening connection, dropped syns
- Persistent connections
  - Lose persistent connection semantics
  - Log entries not associated with connections

Servers



Hits
Misses
Misses
Internet
Clients
Proxy, Proxy Caches
Servers

Server Traces

- Advantages
  - Global client behavior across entire Internet
  - Object change events
- As with caches, server events recorded in logs
  - Easy to enable, analyze
- Formats
  - Common Logfile Format
  - Convention established by W3C httpd server
  - Supported by all server vendors
  - **http://www.w3.org/Daemon/User/Config/Logging.html#common-logfile-format**

Common Logfile Format

- General format:

```
remotehost rfc931 authuser [date] "request" status
bytes [optional]
```

- Example entry:

```
dt103n5a.san.rr.com - - [30/Jun/2000:00:36:12 -
0700] "GET / HTTP/1.1" 304 - "-"  "Mozilla/4.0
(compatible; MSIE 5.0; Windows 98; DigExt)"
```

Logfile example

remotehost	dt103n5a.san.rr.com	Client
rfc931	-	Ident
authuser	-	Auth Ident
[date]	[30/Jun/2000:00:36:12 -0700]	
"request"	"GET / HTTP/1.1"	Request Line
status	304	Response Status
bytes	-	Size (unknown)
[opt refer]	"-“	Referrer
[opt agent]	"Mozilla/4.0 (compatible; MSIE 5.0; Windows 98; DigExt)“	User-Agent

Server Trace Analyses

- Overall request and object distributions
  - Servers see all their clients
- Useful for modeling, generating synthetic loads
  - Request arrival rates, distributions

Server Performance

- Structure
  - Multiprocessing, multithreading
  - Handling common cases efficiently
- Interactions with OS, file system
  - Locating, stat-ing, reading workload
- Dynamic data
  - Fast execution of server CGI scripts, servlets, etc.
  - Caching dynamic data

Server Front Ends

- Server front ends
  - Cache studies on server caches
  - Request routing, affinity
  - Load balancing algorithms

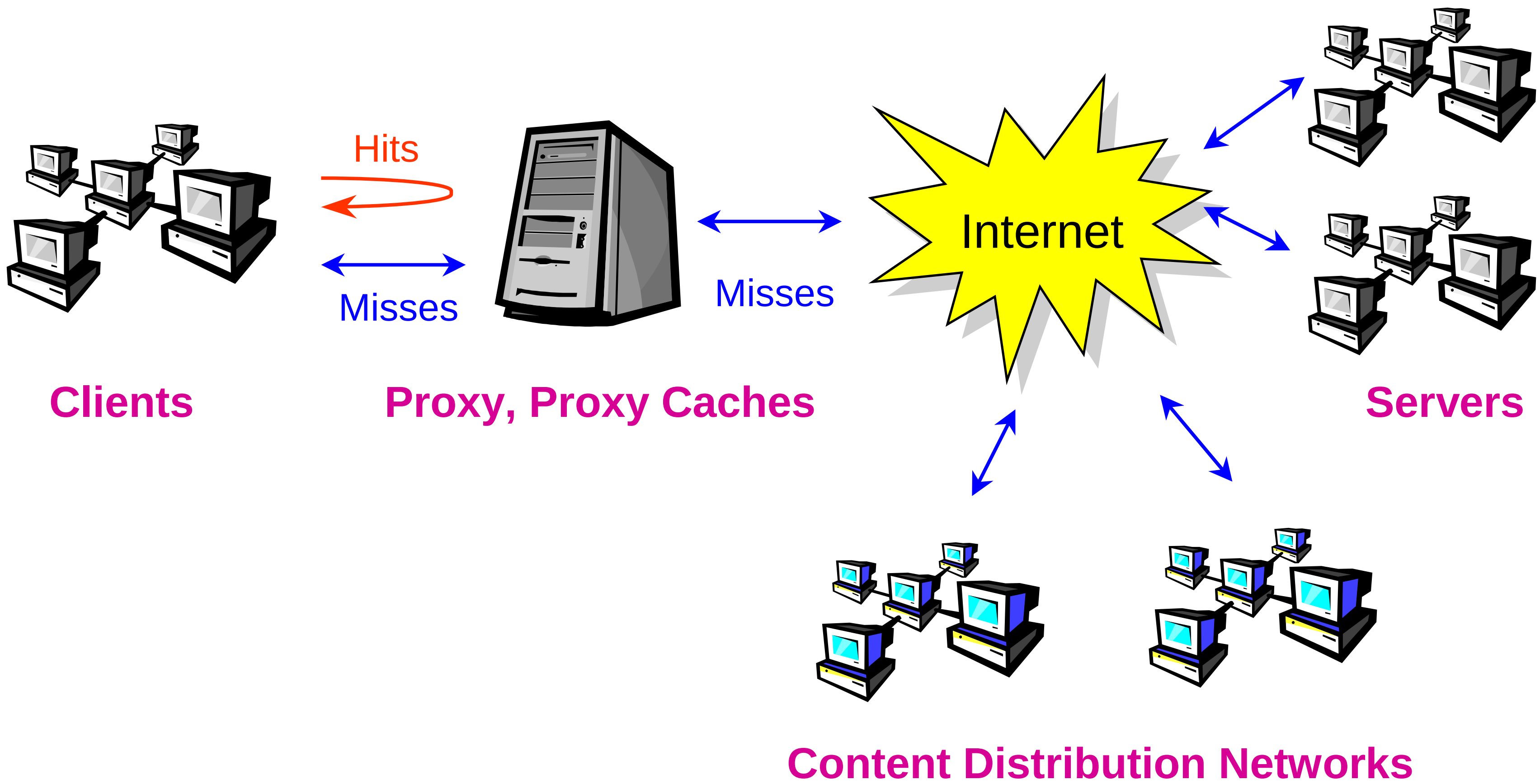
Object Rate of Change

- Potential to see object generation as well as access
- Crucial for numerous mechanisms, analyses
  - Invalidation protocols
  - Prefetching
  - Modeling cache workloads

Server Trace Limitations

- Cannot see client behavior that spans servers
  - No server popularity
  - No session trails
- Caches can mask client behavior
  - Same IP address for all client requests
  - Difficult to disambiguate individual client behavior
  - X-Forwarded-For header can disambiguate

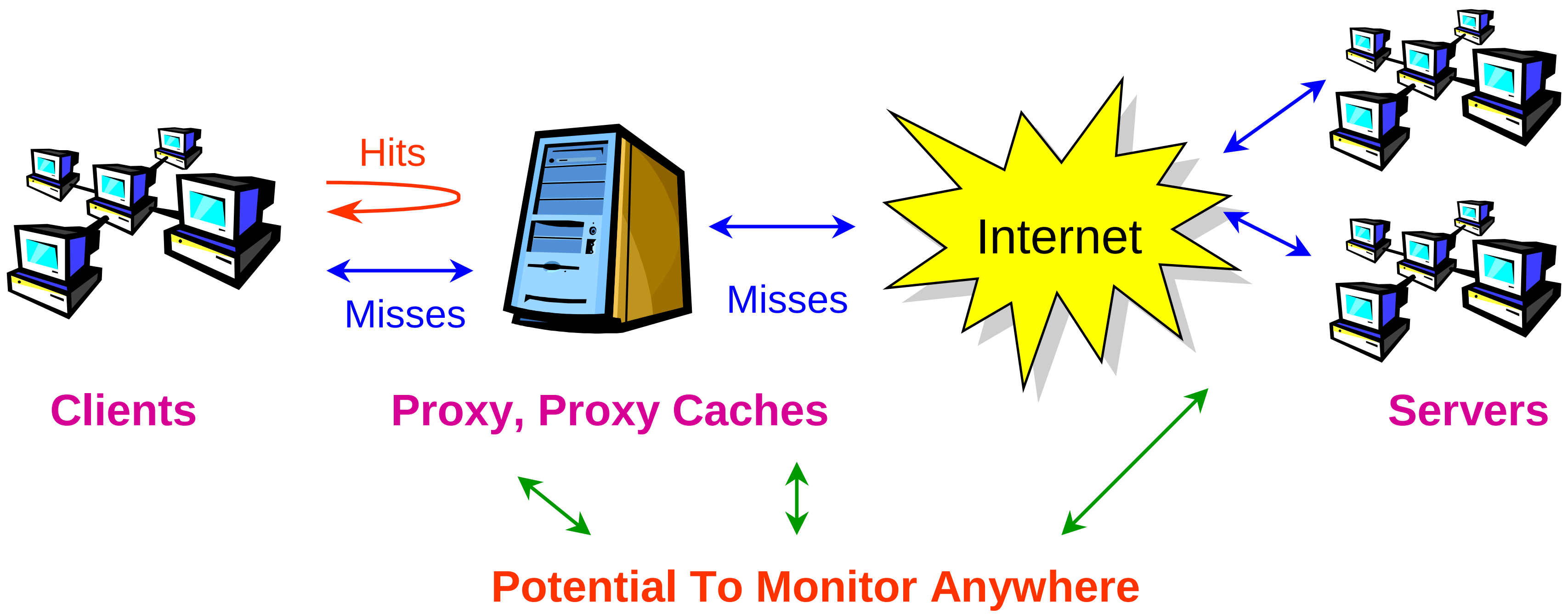
Content Distribution Networks



CDN Traces

- Only available within companies (as far as I know)
  - Trade secret
- Slew of great problems/opportunities, though
  - Server placement
  - Request routing
  - Content redistribution
  - Prefetching
  - Advanced features (metering, invalidation, etc.)

Network Traces



Network Packet Traces

- Advantages
  - Full knowledge of network behavior
  - Nothing is hidden
  - Sometimes the only option you have (e.g., UW, wide-area)
- Passive monitoring
  - Mirrored ports from switches, routers
  - Splitters (OC3/12-mon tools)

Network Trace Approaches

- Full packet dumps
  - Easy to do
  - Run tcpdump, save to file
  - Can do the hard stuff offline
- Summaries derived from packets
  - Requires a lot more software support
    - » Online modeling of TCP connections
    - » HTTP request, response parsing
  - Why do it this way?
    - » Anonymization
    - » Storage

Packet Trace Analyses

- Get to see everything
- But, more of a hassle to deal with such low-level data
- Analysis software usually developed from scratch
  - Opportunity for a general tool here (maybe there is one)
  - At least to recover requests/responses from packets

SYNs and FINs

- Witness TCP SYNs and FINs
  - Connection establishment, termination
- Establishment
  - Delay between SYN and first data packet
    - » Setup time for connection
    - » Potential benefits for persistent connections
  - Dropped SYNs, nasty timeout delays
- Termination
  - Delay between last data packet and FIN (close)
  - Useful for determining timeouts for persistent connections

Complete Protocol Overhead

- Network utilization due to protocol (in addition to data)
  - IP and TCP headers, options
  - ACKs
  - Retransmissions
- AT&T study by Douglis et al.
  - Modem environment
  - Connection establishment significant source of delay
  - Terminated connections significant source of wasted bandwidth, additional delay

Packet Payloads

- Change analysis
  - Has an updated object really changed?
- Delta analysis
  - Has it changed very much?
- Duplication analysis
  - Is the same page (content-wise) accessed via different URLs?

TCP Sequence Numbers

- Content-Length can lie
  - Except within persistent connection
- TCP sequence numbers count bytes, can use them to determine amount of data sent over connection

Persistent Connections

- Utilization
  - Requests/responses per connection
- Timeouts
  - How long should you keep the connection open?

Challenges

- In general, have to reconstruct TCP connection state
  - Need to recover data in TCP stream
  - Fragmentation and reassembly, acks, retransmissions, etc.
  - Huge hassle, especially if done on-line
- HTTP 1.0 Hack
  - Record first segment of connection
  - Capture entire HTTP 1.0 header almost all of the time
  - Useless if you want all the data, too
- Persistent connections
  - Hack: Assume requests/response headers always begin on packet boundaries

Part II: Traces and Analyses

- Survey various kinds of traces
  - Browser, proxy cache, server, CDNs, network packet traces
- Discuss
  - Advantages
  - Options, formats
  - Analyses
  - Limitations
- Lessons learned from UW tracing project
- Trace archives

UW Tracing Lessons

- If you plan to take your own packet traces…
- Expect to iterate tracing and analysis
  - If you do not save all data, you will not save the right data the first time around
- Trace format will change over time
  - How do you write analysis software that adapts to format changes?
  - Our solution was clumsy: CVS tags
- Tracing software stability
  - Has to run for a week or more without interruption…

UW Trace Lessons (2)

- You will spend a lot of time
  - Debugging (of course)
  - Performance tuning
    - » Load always goes up over time…
  - On both tracing software and analysis software
- Scalability
  - How to use separate machines?
  - Time stamp issue
  - How do you synchronize clocks?

UW Analysis Lessons

- Need lots of memory
  - More than tracing server
- Compute bound more than I/O bound
  - Favor faster compression libraries over lower ratio
  - Zlib is incredibly slow…
- Consider dumping all data into a database, data mining system
  - Likely to be worth it in the long term

Privacy Issues

- Scenarios UW required us to address
  - Subpeona of traces and machines (Freedom of Information)
    - » UW is a public university
    - » Already had a self-appointed civilian "watchdog" shutting down Quake servers in the department
  - "Future President" scenario
- Issues
  - MD5 key management
    - » How do you do repeat measurements?
  - Only anonymized data on disk
    - » How do you debug when machine crashes?

Part II: Traces and Analyses

- Survey various kinds of traces
  - Browser, proxy cache, server, CDNs, network packet traces
- Discuss
  - Advantages
  - Options, formats
  - Analyses
  - Limitations
- Lessons learned from UW tracing project
- Trace archives

Popular Sources of Traces

- Client and proxy traces
  - Boston University client traces (six months, 11/94-5/95)
    - » **http://ita.ee.lbl.gov/html/contrib/BU-Web-Client.html**
  - Digital Equipment Corp proxy (weeks)
    - » Very widely used, although very dated (1996)
    - » **ftp://ftp.digital.com/pub/DEC/traces/proxy/webtraces.html**
  - UC Berkeley Dialup (18 days)
    - » **http://ita.ee.lbl.gov/html/contrib/UCB.home-IP-HTTP.html**
  - CA*netII (Canada research network Squid cache logs, 9/99)
    - » **http://ardnoc41.canet2.net/cache/squid/rawlogs/**
  - NLANR (Daily Squid cache hierarchy)
    - » **ftp://ircache.nlanr.net/Traces/**

Server Traces

- Archived at Internet Traffic Archive
  - http://ita.ee.lbl.gov/html/traces.html
- WorldCup98 servers
- University servers
- Government servers
- ISP server

Part II: Summary

- Survey various kinds of traces
  - Browser, proxy cache, server, CDNs, network packet traces
- Discuss
  - Advantages
  - Options, formats
  - Analyses
  - Limitations
- Lessons learned from UW tracing project
- Trace archives

- Questions…?

Part III: Tools

- Generating requests
- Munging cache and server logs
- Cache and server benchmarks, workload generators

Wget

- Useful command-line tool for downloading objects
  - `ftp://gnjilux.cc.fer.hr/pub/unix/util/wget/`

    "GNU Wget is a free network utility to retrieve files from the World Wide Web using HTTP and FTP, the two most widely used Internet protocols. It works non-interactively, thus enabling work in the background, after having logged off."

- Easy way to get headers…

libwww-perl

- Perl library for generating HTTP requests
  - **http://www.ics.uci.edu/pub/websoft/libwww-perl/**
- Useful for writing perl programs that use the Web

"libwww-perl is a library of Perl packages/modules which provides a simple and consistent programming interface to the World Wide Web."

Squid Logs and Common Logfile Scripts

- Squid logs and Common Logfile Format scripts
  - **http://www.squid-cache.org/Scripts/**
- Additional Common Logfile Format (httpd server) tools
  - **http://www.w3.org/Tools/Overview.html#LogStat**

NLANR Scripts

- Scripts used to generate NLANR stats
  - **http://www.squid-cache.org/Scripts/NLANR/**
- Published stats
  - **http://www.ircache.net/Cache/Statistics/**

Web Polygraph

- IRCACHE Proxy performance benchmark
  - **http://polygraph.ircache.net/**

  "Our ambition is to develop and support a de facto benchmarking standard for the Web caching industry."

Wisconsin Proxy Benchmark

- Workload generator for proxies
  - `http://www.cs.wisc.edu/~cao/wpb1.0.html`

Rice Server Traffic Generator

- Targets peak loads to exceed server capacity
  - **http://www.cs.rice.edu/CS/Systems/Web-measurement/sources.html**

Part III: Summary

- Many different tools out there
- Now on to the lab…

Blank

This slide intentionally left blank