

# CSE 461

## HTTP / DNS / CDNs (/ Privacy)

---

### Last Time ...

---

- The Transport Layer
- Focus
  - How does TCP share bandwidth?
- Topics
  - AIMD
  - Slow Start
  - Fast Retransmit / Fast Recovery

Application
Presentation
Session
<b>Transport</b>
Network
Data Link
Physical

## This Lecture

---

- HTTP and the Web (but not HTML)
- DNS
- Content Distribution Networks (CDNs)

Application
Presentation
Session
Transport
Network
Data Link
Physical

3

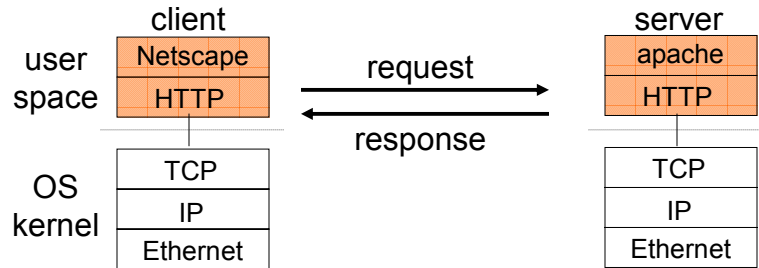
## Part 1: The Web

---

4

## Web Protocol Stacks

---



- To view the URL <http://server/page.html> the client makes a TCP connection to port 80 of the server, by it's IP address, sends the HTTP request, receives the HTML for page.html as the response, repeats the process for inline images, and displays it.

5

## HTTP/HTML

---

- Original goal of the web: file sharing / viewing
- Big idea: display of content should be up to viewer, not author
  - Data might be astronomical observations
  - Client may or may not be able to run a graphical visualizer
  - Data of many different types: plain text, formatted text, tables of values, images, etc.
  - (Compare this idea with WYSIWIG, e.g., Word)
- **HTTP** (Hyper Text Transfer Protocol) is the protocol used to communicate between the browser and the server
- **HTML** (Hyper Text Markup Language) is the language for marking format marks in text
  - `<p>` means "start a new paragraph"
  - `<b>xxx</b>` means "display xxx in bold"
  - `<i>italicize me</i>`
  - Etc.

6

# HTTP

---

- HTTP was originally intended to support file fetch
- HTTP is a request-response protocol
  - Client ⇒ Server: Give me file `index.html`
  - Server ⇒ Client: Here it is...
- Like every other protocol we've seen, it defines a header, which is used to communicate meta-data between the client and the server
  - Client header: operation, URL, HTTP version, etc.
    - `GET /education/courses/cse461/06au/index.html HTTP/1.0`
  - Server header: result code, file metadata
    - `HTTP/1.0 200`

7

# HTTP Protocol Headers

---

- HTTP runs on top of TCP, which runs on top of IP
  - So?
- HTTP headers are very flexible
  - Expressed as text (ASCII)
  - Keyword delimited fields, not position (mostly)
    - First line IS position sensitive
    - As always, need framing
      - A blank line (CrLf) separates header from payload

8

## Example Exchange

---

- GET /index.html HTTP/1.1  
User-Agent: Mozilla/5.0 (Windows; U; Windows NT 5.1; en-US; rv:1.8.0.8) Gecko/20061025 Firefox/1.5.0.8  
Host: www.w3.org  
Accept: text/plain, text/html
- HTTP/1.1 200 OK  
Date: Thu, 30 Nov 2006 09:55:58 GMT  
Server: Apache/1.3.37 (Unix) mod\_pubcookie/3.3.2b  
mod\_ssl/2.8.28 OpenSSL/0.9.8a  
Transfer-Encoding: chunked  
Content-Type: text/html; charset=iso-8859-1  
Content-Language: en 48a  
  
48a  
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN"  
"http://www.w3.org/TR/REC-html40/loose.dtd">  
<html>

9

## Payload is typed (MIME types)

---

- Multi-part Internet Mail Exchange (MIME) types
  - text/plain
  - text/html
  - application/msword
  - application/octet-stream
  - img/jpeg
  - audio/mpeg
  - video/quicktime
  - multipart/encrypted

10

## multipart/encrypted

---

```
Content-Type: multipart/encrypted; protocol="TYPE/STYPE";  
boundary="Encrypted Boundary"
```

```
--Encrypted Boundary  
Content-Type: TYPE/STYPE
```

```
CONTROL INFORMATION for protocol "TYPE/STYPE" would be  
here
```

```
--Encrypted Boundary  
Content-Type: application/octet-stream
```

```
Content-Type: text/plain; charset="us-ascii"
```

11

## HTTP/HTML Recap

---

- Original idea of HTTP/HTML was to support fetch and display of remote data
  - Everyone is a publisher, everyone is their audience
- Original vision was that object being fetched is a file
  - Modified to allow *dynamic content*: pages generated on-the-fly
    - Example: Amazon pages
  - “cgi” (common gateway interface) means a program that is run because of an page fetch request given to a web server
  - “server side include” is a cruder form of server-side dynamic content
  - Applets / Javascript are dynamic content, but running on client side
  - Many more modifications to the original idea have evolved over time

12

## Improving Performance #1: Caching

- Caches can be (and are) deployed in three places:
  - The client
    - The browser keeps a cache of recently viewed pages
      - Avoid network latency/overhead
  - The server
    - Avoid disk IO
  - The network
    - Proxy caches intercept HTTP requests and return cached page
    - Typically deployed between local network(s) and ISP
      - E.g., a UW proxy cache
- Both client and server can influence if / how long a page is cached
  - Server: cacheable/non-cacheable, expiration time
  - Client: browser settings for how often to check for page changed

13

## How Effective Is Caching?

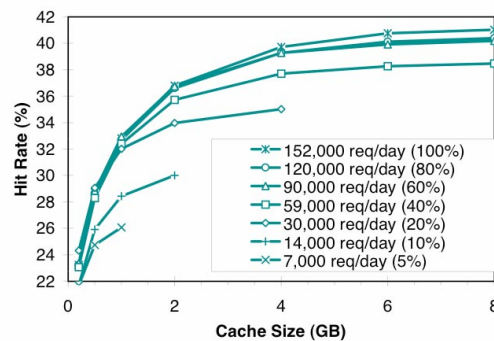


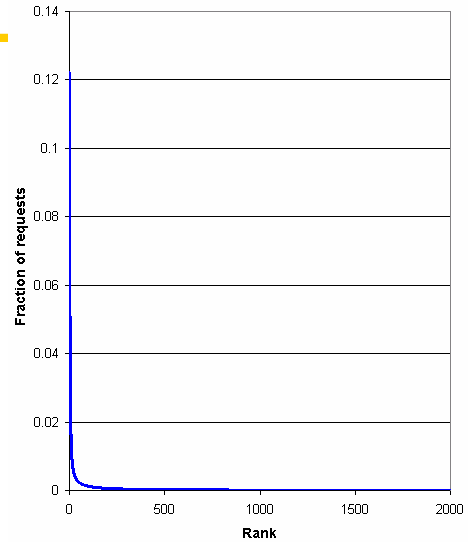
Figure 3: Cache hit rate for KOR as a function of cache size for a range of request rates.

14

## Why?

---

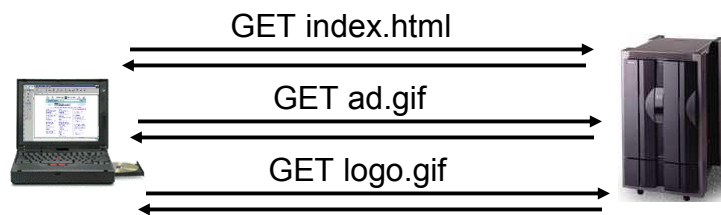
- Page popularity follows Zipf's law
  - $N^{\text{th}}$  most popular page is accessed with frequency proportional to  $1/N$
  - "Heavy tail" of this distribution means there are a lot of pages not visited by many clients



15

## Improving Performance #2: HTTP Protocol Tweak

---



16



# HTTP Request/Response in Action

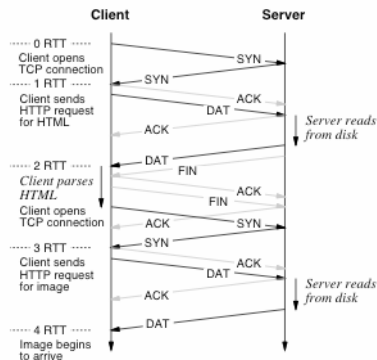


Figure 3-1: Packet exchanges and round-trip times for HTTP

- Problem is that:
  - Web pages are made up of many files
    - Most are very small (< 10k)
  - files are mapped to connections

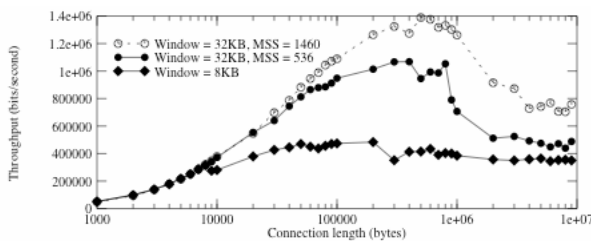
### For each file

- Setup/Tear-down
  - Time-Wait table bloat
- 2RTT "first byte" latency
- Slow Start+ AIMD Congestion Avoidance

The goals of HTTP and TCP protocols are not aligned.

- Implications

# TCP Behavior for Short Connections Over Slow Networks

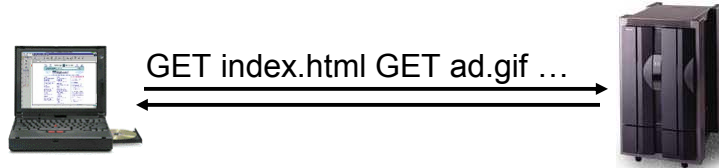


RTT=70ms

Figure 3-2: Throughput vs. connection length, RTT = 70 msec

Figure 3-2 shows that, in the remote case, using a TCP connection to transfer only 2 Kbytes results in a throughput less than 10% of best-case value. Even a 20 Kbyte transfer achieves only about 50% of the throughput available with a reasonable window size. This reduced throughput translates into increased latency for document retrieval. The figure also shows that, for this 70 msec RTT, use of too small a window size limits the throughput no matter how many bytes are transferred.

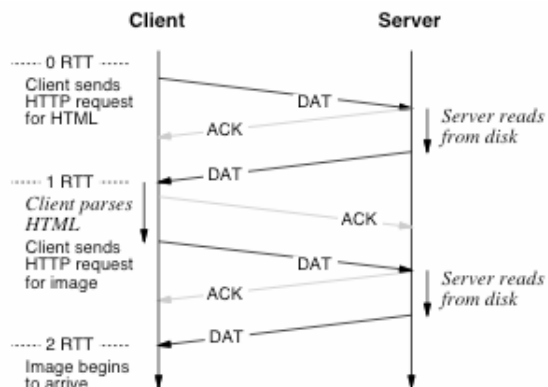
## HTTP 1.1: Persistent Connections



- Bright Idea: Use one TCP connection for multiple page downloads (or just HTTP methods)
- Q: What are the advantages?
- Q: What are the disadvantages?
  - *Application layer multiplexing*

19

## HTTP/1.1



20

## Effect of Persistent HTTP

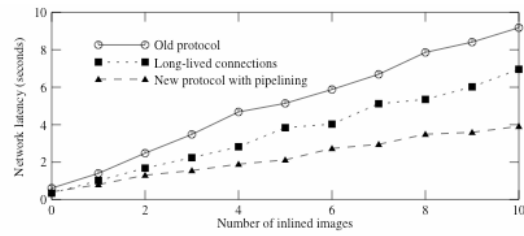


Figure 6-1: Latencies for a remote server, image size = 2544 bytes

Image size=2544

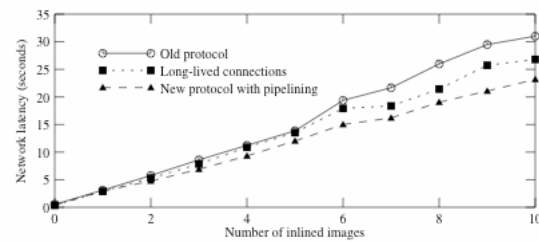


Figure 6-2: Latencies for a remote server, image size = 45566 bytes

Image size=45566

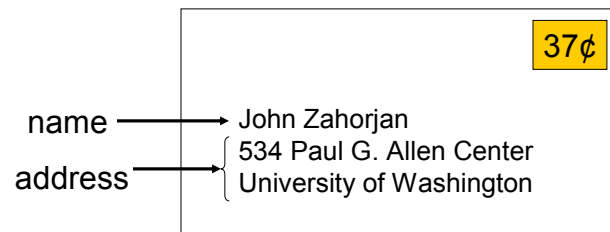
21

## Part 2: The Domain Name System

22

## Names and Addresses

---



- Names are identifiers for objects/services (high level)
- Addresses are locators for objects/services (low level)
- Binding is the process of associating a name with an address
- Resolution is the process of looking up an address given a name
- But, addresses are really lower-level names; many levels used

23

## Internet Hostnames

---

- Hostnames are human-readable identifiers for end-systems based on an administrative hierarchy
  - peshastin.cs.washington.edu is my desktop machine
- IP addresses are a fixed-length binary encoding for end-systems based on their position in the network
  - 128.208.2.93 is peshastin's IP address
- Original name resolution: HOSTS.TXT
- Current name resolution: Domain Name System (DNS)

24

## Domain Name System (DNS)

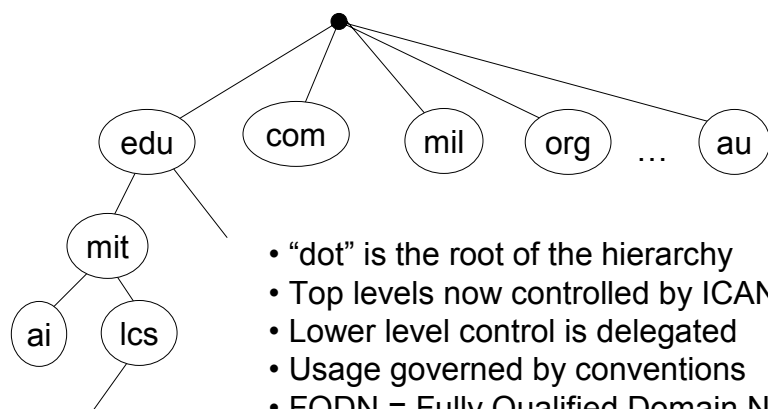
---

- Designed in the mid '80s
- Namespace is hierarchical
  - Decentralized administration
    - e.g., \*.cs.washington.edu managed by CSE
- Name service (DNS) is distributed
  - Allows much better scaling of data structures
    - e.g., peshastin.cs.washington.edu
- Resolution is by query/response
  - With replicated servers for redundancy
  - With heavy use of caching for performance

25

## DNS Hierarchy

---



- “dot” is the root of the hierarchy
- Top levels now controlled by ICANN
- Lower level control is delegated
- Usage governed by conventions
- FQDN = Fully Qualified Domain Name

26

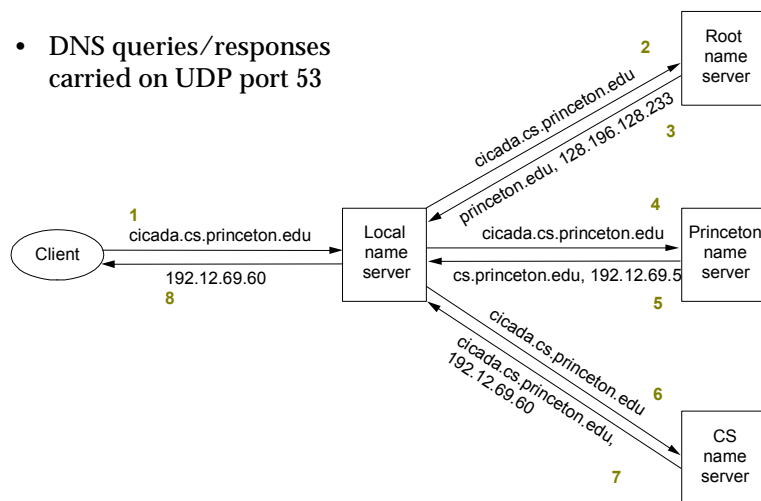
## DNS Distribution

- Data managed by zones that contain resource records
  - Zone is a complete description of a portion of the namespace
  - e.g., all hosts and addresses for machines in washington.edu with pointers to subdomains like cs.washington.edu
- One or more nameservers manage each zone
  - Zone transfers performed between nameservers for consistency
  - Multiple nameservers provide redundancy
- Client resolvers query nameservers for specified records
  - Multiple messages may be exchanged per DNS lookup to navigate the name hierarchy (coming soon)

27

## DNS Lookups/Resolution

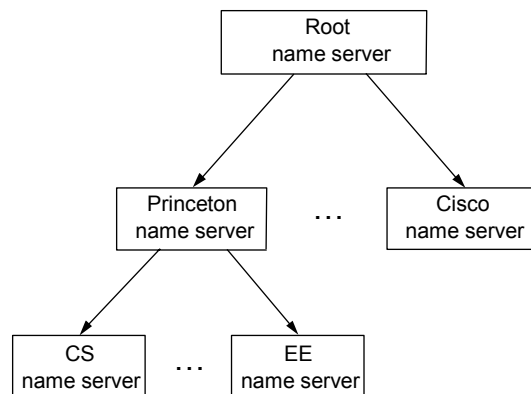
- DNS queries/responses carried on UDP port 53



28

## Hierarchy of Nameservers

---



29

## DNS Bootstrapping

---

- Need to know IP addresses of root servers before we can make any queries
- Addresses for 13 root servers ([a-m].root-servers.net) handled via initial configuration (named.ca file)
- To avoid overloading roots, heavy use of caching (at all levels)
  - Like web pages, cache entries are timed out
  - DNS also caches negative entries!
    - Mistyped names are the most frequent single request to root servers.

30

## Part 3: Content Distribution Networks

---

31

## CDNs

---

- Imagine someone on campus fetches  
[www.nytimes.com/2006/12/04/technology/04adcol.html](http://www.nytimes.com/2006/12/04/technology/04adcol.html)
- It benefits both the UW and the NYT if that page is cached at the UW
  - UW pays less for bandwidth to request page
  - NYT pays less for bandwidth/servers to deliver the page
- But
  - NYT can't rely on everyone being behind an effective cache, and..
  - NYT wants to be able to customize its content
- Basic idea of CDNs:
  - Push content to servers operating near clients
    - E.g., Akamai

32



## Customization (Part 1)

---

- Client may want to customize her view of NYT
  - Show me articles about Chicago at the top of the page
  - Show me weather for Chicago
  - Etc.
- The client's customization options need to be stored somewhere
  - Straightforward: NYT servers keep track
  - Better: push the customization information to the client, and let the browser supply it with each request
- This is the motivation for browser *cookies*

33

## Back to CDNs

---

- Uniform Resource Locators (URLs) begin with a host name
  - `www.nytimes.com/2006/12/04/technology/04adcol.html`
    - Host is `www.nytimes.com`
    - Requested file is `/2006/12/04/technology/04adcol.html`
- Web pages embed links to other pages
  - Example:  
``
- Key idea: use DNS to direct fetch of content (like images) to CDN servers
  - `graphics10.nytimes.com` is a CDN server, not one on the NYT local network

34

## DNS & CDNs

---

```
$ dig www.nytimes.com
;; ANSWER SECTION:
www.nytimes.com.      294    IN     A      199.239.137.245
www.nytimes.com.      294    IN     A      199.239.136.200
www.nytimes.com.      294    IN     A      199.239.136.245
www.nytimes.com.      294    IN     A      199.239.137.200

$ dig graphics10.nytimes.com
;; ANSWER SECTION:
graphics10.nytimes.com. 300    IN     CNAME  p.mii.instacontent.net.
p.mii.instacontent.net. 600    IN     A      192.147.176.114
```

35

## Future Evolution of the DNS

---

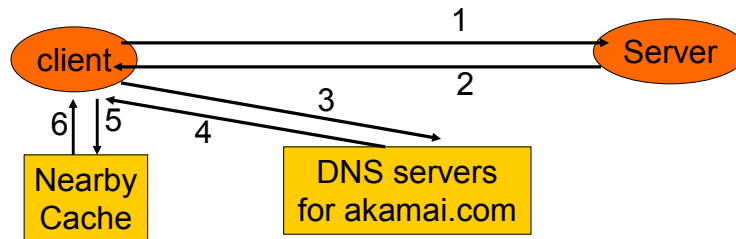
- Design constrains us in two major ways that are increasingly less appropriate
- Static host to IP mapping
  - What about mobility (Mobile IP) and dynamic address assignment (DHCP)
- Location-insensitive queries
  - What if I don't care what server a Web page comes from, as long as it's the right page?
  - e.g., a yahoo page might be replicated

36

## Akamai

---

- Use the DNS to effect selection of a nearby Web cache



- Leverage separation of static/dynamic content
- Beware DNS caching

37

## Part 4: Privacy

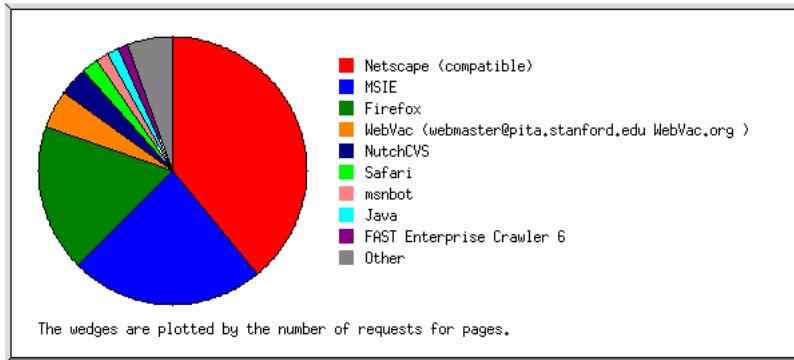
---

- What is your browser supplying to the server on each request?
  - Your IP address
  - Cookies
- What are servers recording?
- What if many different servers shared information about your habits?
- What if a CDN company fronted for many, many companies?

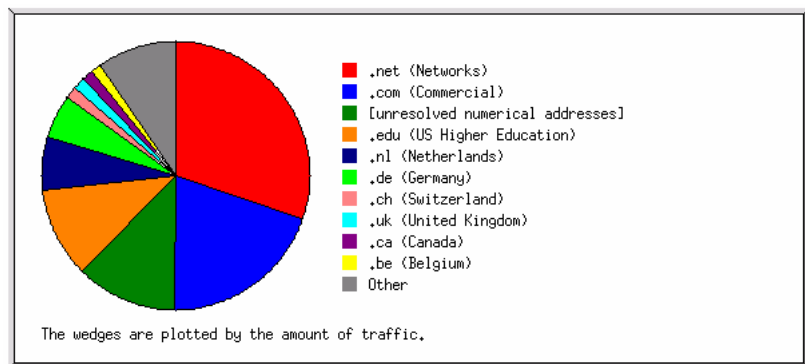
38

## Some statistics from www.cs.washington.edu

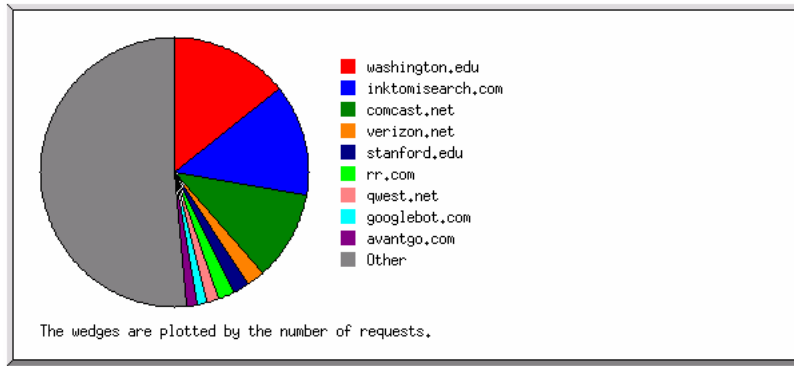
---



39



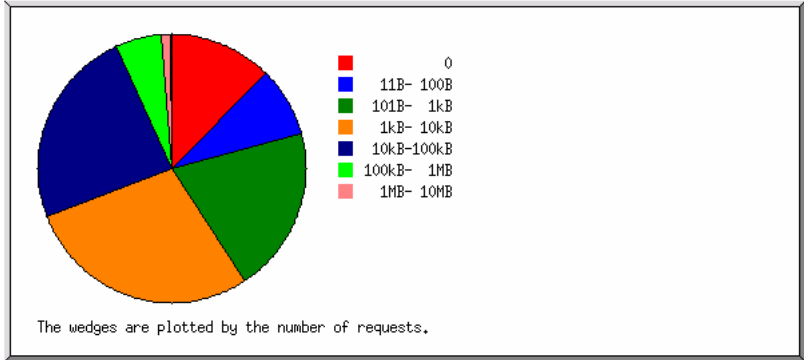
40



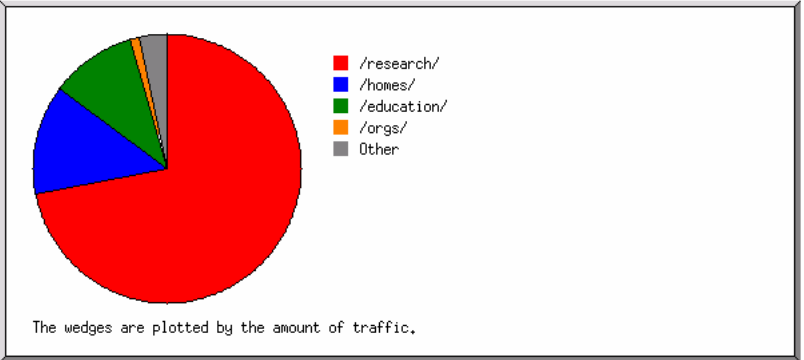
41

#reqs	%bytes	host
22006	0.13%	adamsjq.cs.washington.edu
20159	0.09%	wb14.stanford.edu
12263	0.28%	159.226.43.54
11173	0.01%	c-24-22-189-188.hsd1.wa.comcast.net
9884	2.11%	crawl-66-249-66-193.googlebot.com
9368	1.77%	crawl-66-249-66-135.googlebot.com
8734	0.02%	wb15.stanford.edu
7825	0.58%	70.42.51.20
4567	0.46%	tide531.microsoft.com
4392	0.01%	mangrove.cs.washington.edu
4375	0.21%	livebot-65-55-212-51.search.live.com
3577	0.23%	58.38.152.64
2878	0.01%	219.131.223.235
2576	0.04%	mtlxpqak-1177743779.sdsl.bell.ca
2531	0.05%	218.22.21.23
2526		natcrawlbloc03.net.m1.fi.net

42



43



44