CS 172: INFORMATION RETRIEVAL

Web Crawler (Chapter 3)

Outline

- **Intro stuff (upto slide 35)**
  - **HTTP Requests**
  - **Server-Client**

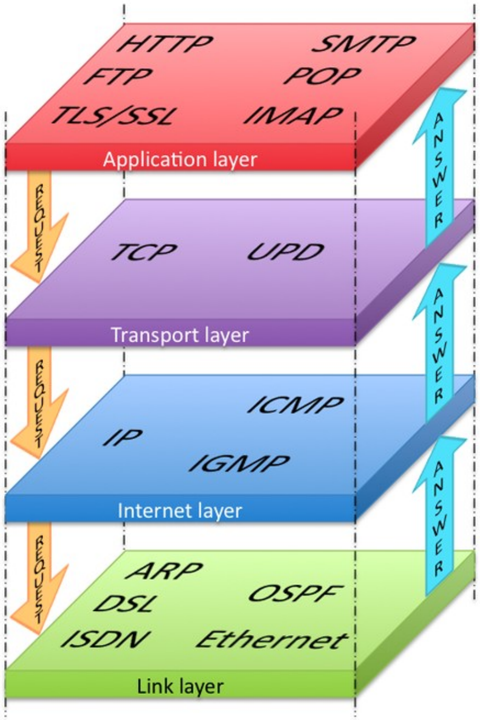
- Web Crawlers (Chapter 3)
  - How to build a web crawler
  - Distributed crawling
  - BigTable
  - Duplicate detection

TCP/IP

- To promote the growth and unification of the disparate networks a suite of protocols was invented to unify the networks together.

- By 1981, new networks built in the US began to adopt the **TCP/IP (Transmission Control Protocol / Internet Protocol)** communication model, while older networks were transitioned over to it.

Layered Architecture



Higher protocols that allow applications to interact with the transport layer

Ensures transmissions arrive in order and without error

Establishes connection, routing, and addressing

Responsible for physical transmission of raw bits

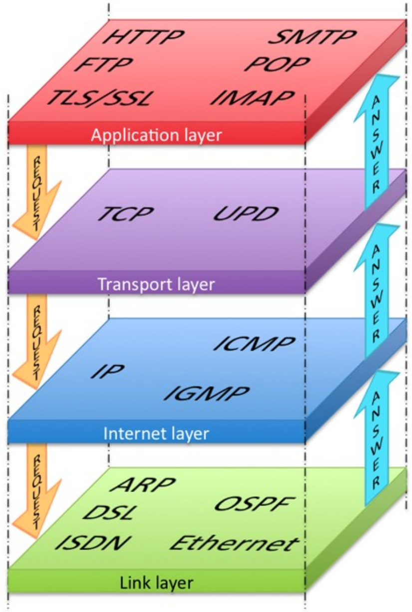
Link Layer

- The **link layer** is the lowest layer, responsible for both the physical transmission across media (wires, wireless) and establishing logical links.

- It handles issues like packet creation, transmission, reception and error detection, collisions, line sharing and more.

Internet Layer

- The internet layer (sometimes also called the IP Layer) routes packets between communication partners across networks.



Internet Protocol (IP)

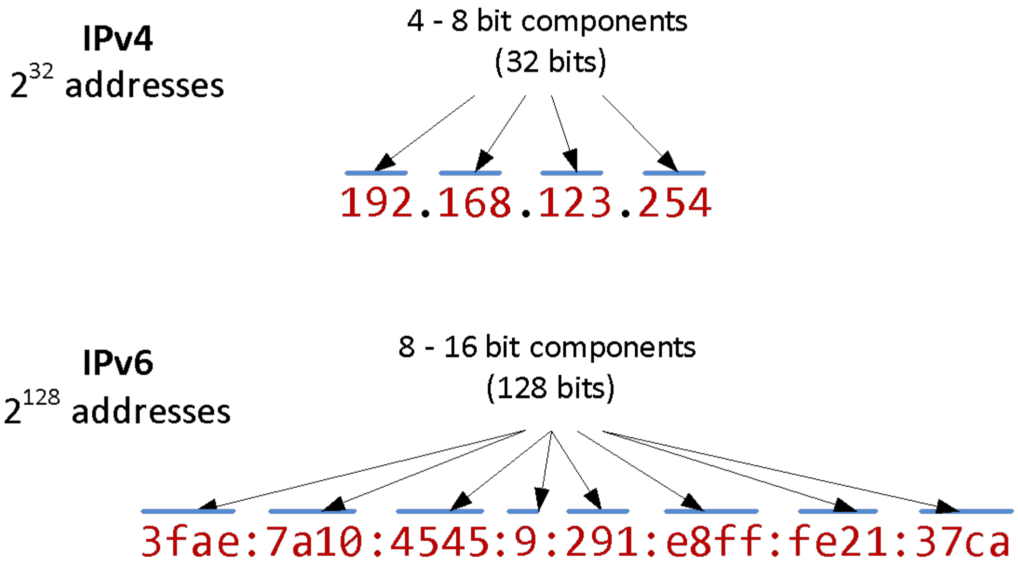
- The Internet uses the Internet Protocol (IP) addresses to identify destinations on the Internet.

- Every device connected to the Internet has an IP address, which is a numeric code that is meant to uniquely identify it.

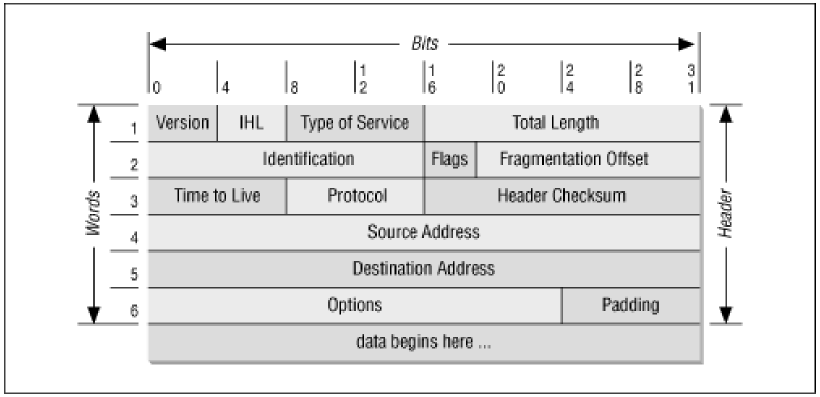
- Your IP address will generally be assigned to you by your Internet Service Provider (ISP).
  - In organizations, large and small, purchasing extra IP addresses from the ISP is not cost effective.
  - In a local network, computers can share a single IP address between them.
  - The router, which has a static address, can delegate dynamic IP address to the devices connected to it.

IP Addresses

- **IPv4** addresses are the IP addresses from the original TCP/IP protocol.
  - In IPv4, 12 numbers are used (implemented as four 8-bit integers), written with a dot between each integer.
  - Since an unsigned 8-bit integer's maximum value is 255, four integers together can encode approximately 4.2 billion unique IP addresses.
- **IPv6** is newer version of IP addresses.
  - It uses eight **16-bit** integers for 2^{128} unique addresses, over a billion billion times the number in IPv4.
  - These 16-bit integers are normally written in hexadecimal, due to their longer length.



Transport Layer



- The **transport layer** ensures transmissions arrive, in order, and without error.

- This is accomplished through a few mechanisms.
  - First, the data is broken into packets formatting according to the **Transmission Control Protocol** (**TCP**).
  - Secondly, each packet is acknowledged back to the sender so in the event of a lost packet, the transmitter will realize a packet has been lost since no ACK arrived for that packet.
  - That packet is retransmitted, and although out of order, is reordered at the destination.

Application Layer

- With the **application layer**, we are the level of protocols familiar to most web developers.

- Application layer protocols implement process-to-process communication and are at a higher level of abstraction in comparison to the low-level packet and IP addresses protocols in the layers below it.

- Examples: HTTP, SSH, FTP, DNS, POP, SMTP.

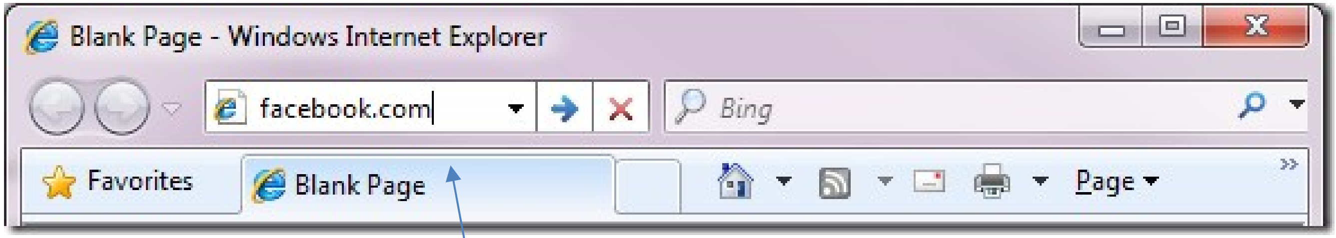
What happens when you navigate to a URL?

- We will concentrate on the application layer and the communication that is made between the client and server when a URL is requested.

- Lets take a look at the sequence of events that take place when you visit a URL (like facebook.com)

1 – You enter a URL into a browser

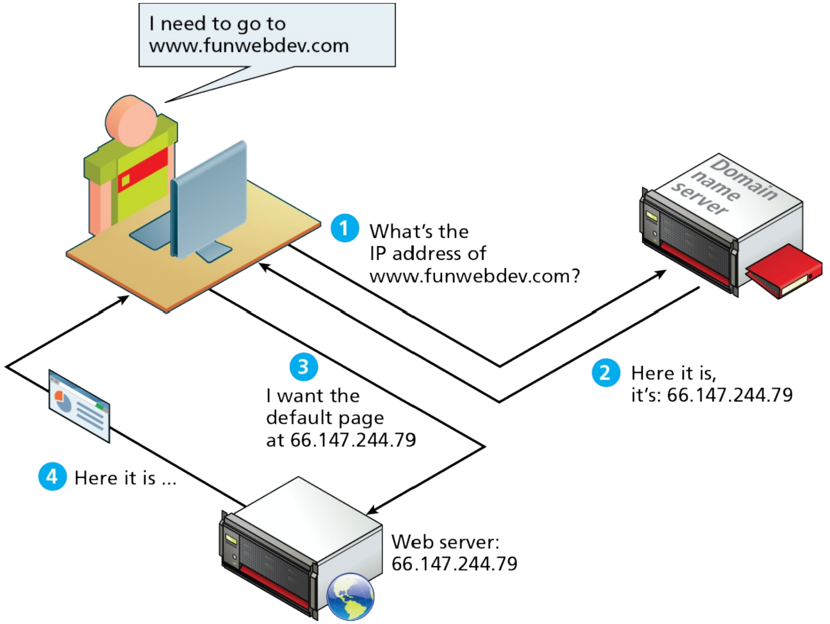
- It all starts with you typing a URL in the browser



Notice, I typed http://facebook.com instead of http://www.facebook.com

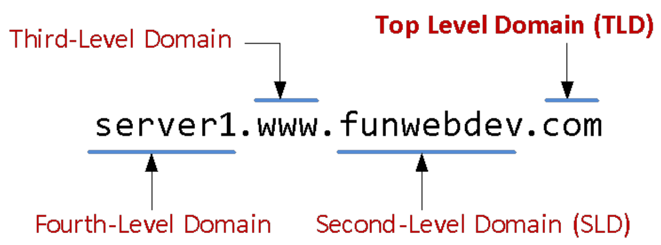
2 – Browser looks up IP via DNS server

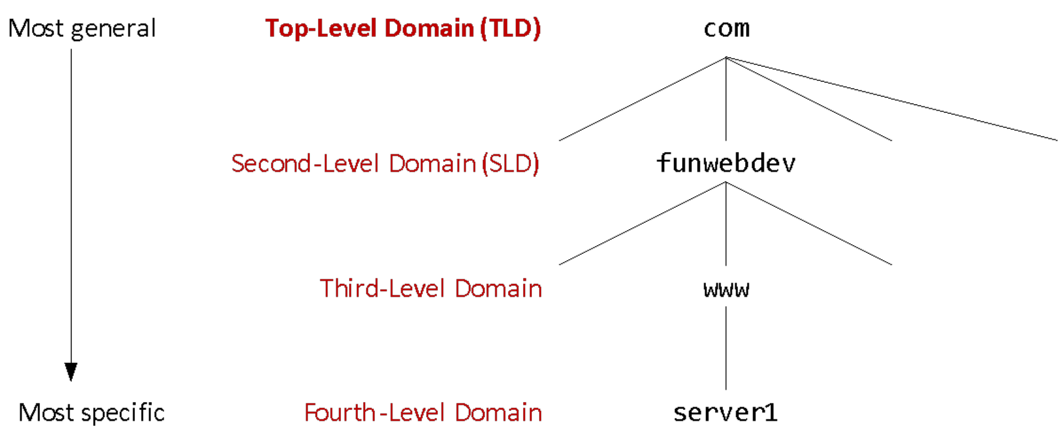
- As elegant as IP addresses may be, we do not enjoy having to recall long strings of numbers.
- Instead of IP addresses, we use the Domain Name System (DNS) to lookup the the IP address associated with a given domain.
- Also, by separating IP from domain name, a site can move to a different location without changing its name.



Domain Levels

TLD
Unrestricted: .com, .net, .org, .info
Sponsored: .gov, .mil, .edu
Country Codes: .us,.ca, .uk, .au





DNS Address Resolution

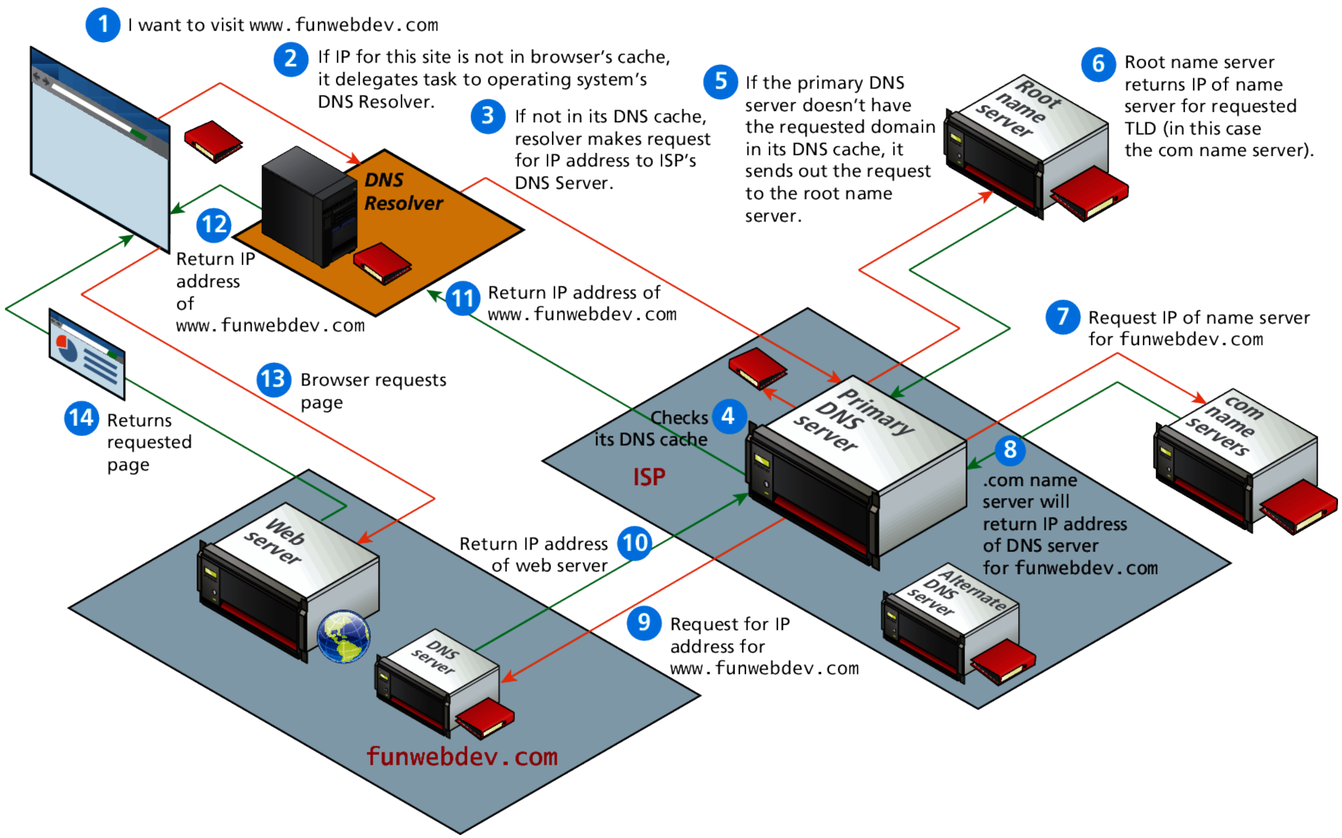
- So, IP lookup (referred to as **address resolution**) is not as simple as was shown in the previous slide.

- The DNS is a distributed database system (no longer centralized) of name servers.
  - From your perspective, its like a phonebook, mapping a unique name to a number.

- The address resolution process goes through the following steps:

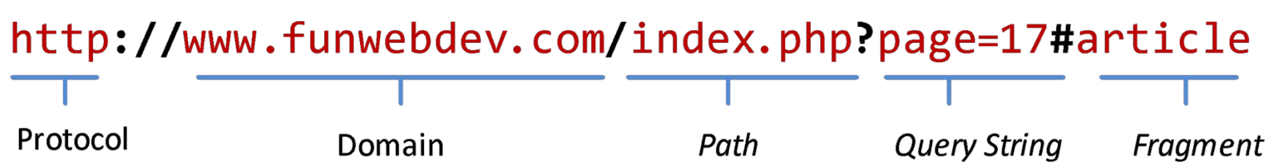
  - **Browser cache** – The browser caches DNS records for some time.
  - **OS cache** – If the browser cache does not contain the desired record, the browser makes a system call to get the data from the OS ... if available.
  - **Router cache** – The request continues on to your router, which typically has its own DNS cache.
  - **ISP DNS cache** – The next place checked is the cache ISP's DNS server.
  - **Recursive search** – Your ISP's DNS server begins a recursive search, from the root nameserver, through the .com top-level nameserver, to Facebook's nameserver. Normally, the DNS server will have names of the .com nameservers in cache, and so a hit to the root nameserver will not be necessary

Domain name address resolution process



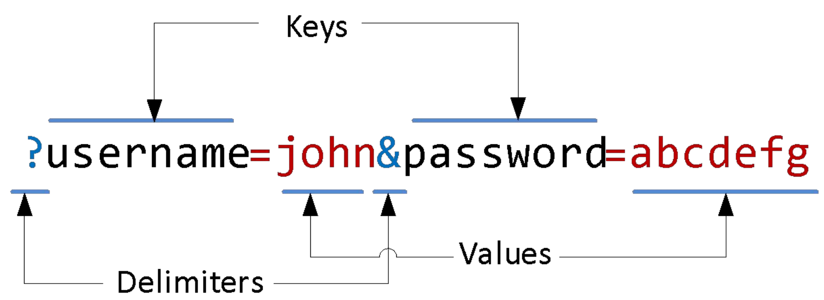
URL Components

- In order to allow clients to request particular resources from the server, a naming mechanism is required so that the client knows how to ask the server for the file.
- For the web that naming mechanism is the **Uniform Resource Locator** (**URL**).



Query String

- Query strings will be covered in depth when we learn more about HTML forms and server-side programming.
- They are the way of passing information such as user form input from the client to the server.
- In URL's they are encoded as key-value pairs delimited by “&” symbols and preceded by the “?” symbol.



EX: https://www.yelp.com/search?find_desc=Restaurants&find_loc=Boston

3. The browser sends a HTTP request to the web server

- So now the browser knows the IP address of the server, the next step is to issue a HTTP GET request.

- You can be pretty sure that Facebook's homepage will not be served from the browser cache because dynamic pages expire either very quickly or immediately (expiry date set to past).

- So, the browser will send this request to the Facebook server:

```
GET http://facebook.com/ HTTP/1.1
Accept: application/x-ms-application, image/jpeg, application/xaml+xml, [...]
User-Agent: Mozilla/4.0 (compatible; MSIE 8.0; Windows NT 6.1; WOW64; [...]
Accept-Encoding: gzip, deflate
Connection: Keep-Alive
Host: facebook.com
Cookie: datr=1265876274-[...]; locale=en_US; lsd=WW[...]; c_user=2101[...]
```

- HTTP establishes a TCP connection on port 80 (by default)

HTTP Request Headers

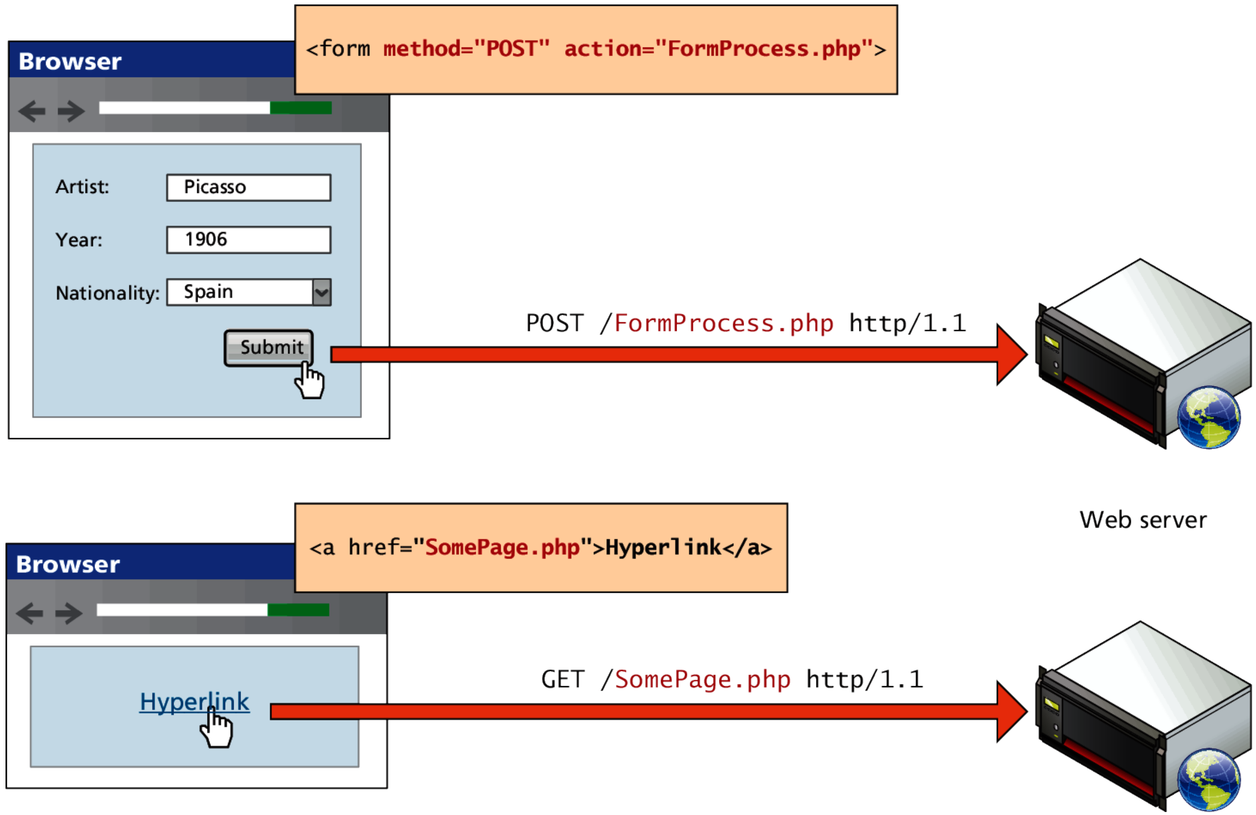
- Headers are sent in the request from the client and received by the server.
- These headers include data about the client machine that can be used by a developer for analytical reasons or personalization.
  - **Host:** requests for different domains can arrive at the same IP, the host tells the server which
  - **User-Agent:** specifying OS and browser used by the client
  - **Accept:** tells the server what kind of media types the client can receive
  - **Accept-Encoding:** specify what types of modifications can be done to the data before transmission
  - **Connection:** specifies whether the server should keep the connection open, or close it after the response.
  - **Cache-Control:** allows the client to control caching mechanisms.

Request Methods

- HTTP protocol defines several different types of requests, each with a different intent.
  - GET, POST, HEAD (and a few seldom used requests such as PUT, DELETE, CONNECT, TRACE, OPTIONS)

- GET request – request for a resource located at a specified URL
- POST request – Used to transmit data to the server using an HTML form
- HEAD request – similar to GET, but only includes the header of a page (not the content) … used by search engines to see if a page needs to be reindexed.

GET versus POST requests



4. The server responds with a permanent redirect

- So, once the client sends a GET request, the server will respond.
- This is the response that the Facebook server sent back to the browser request:

```
HTTP/1.1 301 Moved Permanently
Cache-Control: private, no-store, no-cache, must-revalidate, post-check=0,
      pre-check=0
Expires: Sat, 01 Jan 2000 00:00:00 GMT
Location: http://www.facebook.com/
P3P: CP="DSP LAW"
Pragma: no-cache
Set-Cookie: made_write_conn=deleted; expires=Thu, 12-Feb-2009 05:09:50 GMT;
      path=/; domain=.facebook.com; httponly
Content-Type: text/html; charset=utf-8
X-Cnection: close
Date: Fri, Wed, 18 Jan 2017 21:16:53 GMT
Content-Length: 0
```

- The server responded with a 301 Moved Permanently response to tell the browser to go to “http://www.facebook.com/” instead of “http://facebook.com/”.

5. The browser follows the redirect

- The browser now knows that http://www.facebook.com/ is the correct URL to go to, and so it sends out another GET request:

```
GET http://www.facebook.com/ HTTP/1.1
Accept:   application/x-ms-application,   image/jpeg,
application/xaml+xml, [...]
Accept-Language: en-US
User-Agent:   Mozilla/4.0   (compatible;   MSIE   8.0;
Windows NT 6.1; WOW64; [...]
Accept-Encoding: gzip, deflate
Connection: Keep-Alive
Cookie: lsd=XW[...]; c_user=21[...]; x-referer=[...]
Host: www.facebook.com
```

- The meaning of the headers is the same as for the first request.

6. The server ‘handles’ the request

- The server will receive the GET request, process it, and send back a response.

- This may seem like a straightforward task, but in fact there is a lot of interesting stuff that happens here – even on a simple site, let alone on a massively scalable site like Facebook.

7. The server sends back a HTML response

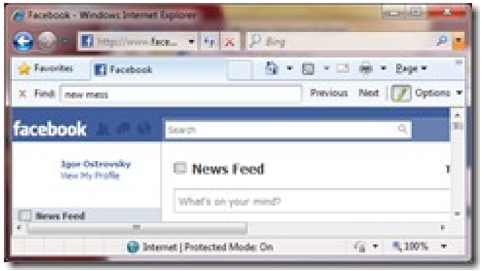
- Here is the response that the server generated and sent back:

```
HTTP/1.1 200 OK
Cache-Control: private, no-store, no-cache, must-
revalidate, post-check=0,
    pre-check=0
Expires: Sat, 01 Jan 2019 00:00:00 GMT
P3P: CP="DSP LAW"
Pragma: no-cache
Content-Encoding: gzip
Content-Type: text/html; charset=utf-8
X-Cnection: close
Transfer-Encoding: chunked
Date: Fri, 12 Feb 2019 09:05:55 GMT

2b3
��������T�n�@����[...]
```

8. The browser begins rendering the HTML

- Even before the browser has received the entire HTML document, it begins rendering the website:



9. The browser sends requests for objects embedded in HTML

- As the browser renders the HTML, it will notice tags that require fetching of other URLs. The browser will send a GET request to retrieve each of these files.

- Here are a few URLs that my visit to facebook.com retrieved:
  - **Images**
  - **CSS style sheets**
  - **JavaScript files**

Each of these URLs will go through process a similar to what the HTML page went through. So, the browser will look up the domain name in DNS, send a request to the URL, follow redirects, etc.

However, static files – unlike dynamic pages – allow the browser to cache them.

10. The browser sends further asynchronous (AJAX) requests

- In the spirit of Web 2.0, the client continues to communicate with the server even after the page is rendered.
  - For example, Facebook chat will continue to update the list of your logged in friends as they come and go.
  - To update the list of your logged-in friends, the JavaScript executing in your browser has to send an asynchronous request to the server.
  - The asynchronous request is a programmatically constructed GET or POST request that goes to a special URL.
  - In the Facebook example, the client sends a POST request to http://www.facebook.com/ajax/chat/buddy_list.php to fetch the list of your friends who are online.

- This pattern is sometimes referred to as “AJAX”, which stands for “Asynchronous JavaScript And XML”.

Outline

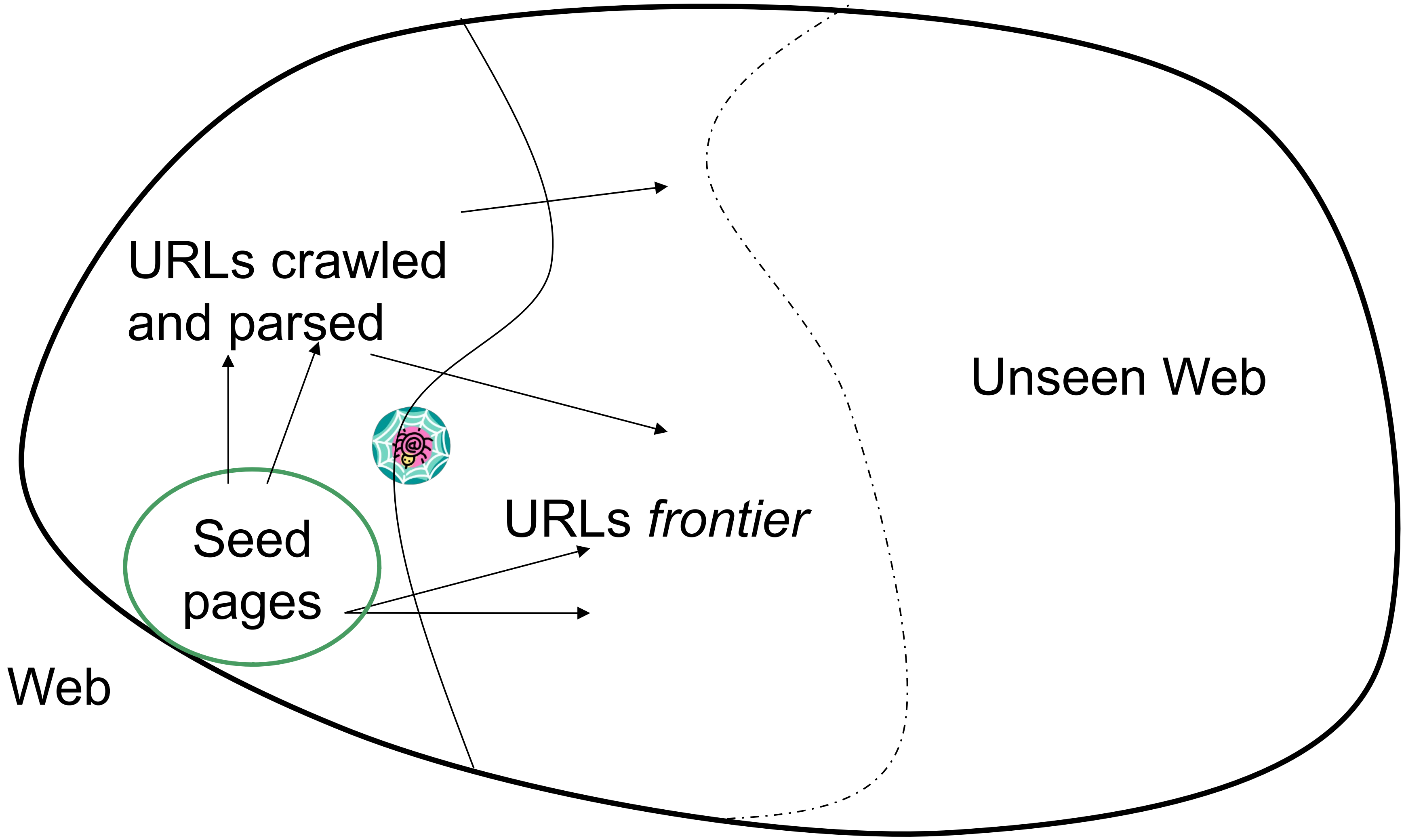
- Crawling
  - BigTable
  - Duplicate detection

- Link Analysis
  - Page Rank
  - MapReduce
  - Page Rank with MapReduce

Basic crawler operation

- Begin with known “seed” URLs
- Fetch and parse them
  - Extract URLs they point to
  - Place the extracted URLs on a queue
- Fetch each URL on the queue and repeat

Crawling picture



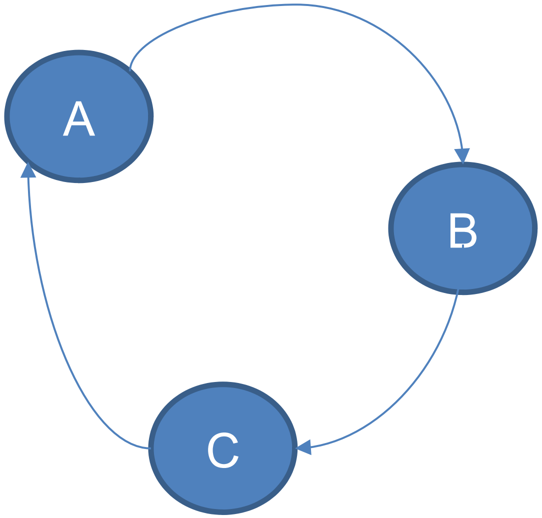
Simple picture – complications

- Web crawling isn’t feasible with one machine
  - All of the above steps distributed
- Malicious pages
  - Spam pages
  - Spider traps
- Even non-malicious pages pose challenges
  - Latency/bandwidth to remote servers vary
  - Webmasters’ stipulations
    - How “deep” should you crawl a site’s URL hierarchy?
- Site mirrors and duplicate pages
- Politeness – don’t hit a server too often

What any crawler must do

- Be Polite: Respect implicit and explicit politeness considerations
  - Only crawl allowed pages
  - Respect robots.txt (more on this shortly)

- Be Robust: Be immune to spider traps and other malicious behavior from web servers



What any crawler should do

- Be capable of distributed operation: designed to run on multiple distributed machines
- Be scalable: designed to increase the crawl rate by adding more machines
- Performance/efficiency: permit full use of available processing and network resources
- Fetch pages of "higher quality" first
- Continuous operation: Continue fetching fresh copies of a previously fetched page

Explicit and implicit politeness

- What is URL frontier?
  - Can include multiple pages from the same host
  - Must avoid trying to fetch them all at the same time

- Explicit politeness: specifications from webmasters on what portions of site can be crawled
  - robots.txt

- Implicit politeness: even with no specification, avoid hitting any site too often

What is Robots.txt ?

- Protocol for giving spiders (“robots”) limited access to a website, originally from 1994
  - www.robotstxt.org/wc/norobots.html

- Website announces its request on what can(not) be crawled
  - For a server, create a file /robots.txt
  - This file specifies access restrictions

Robots.txt example

- No robot should visit any URL starting with "/yoursite/temp/", except the robot called “searchengine":

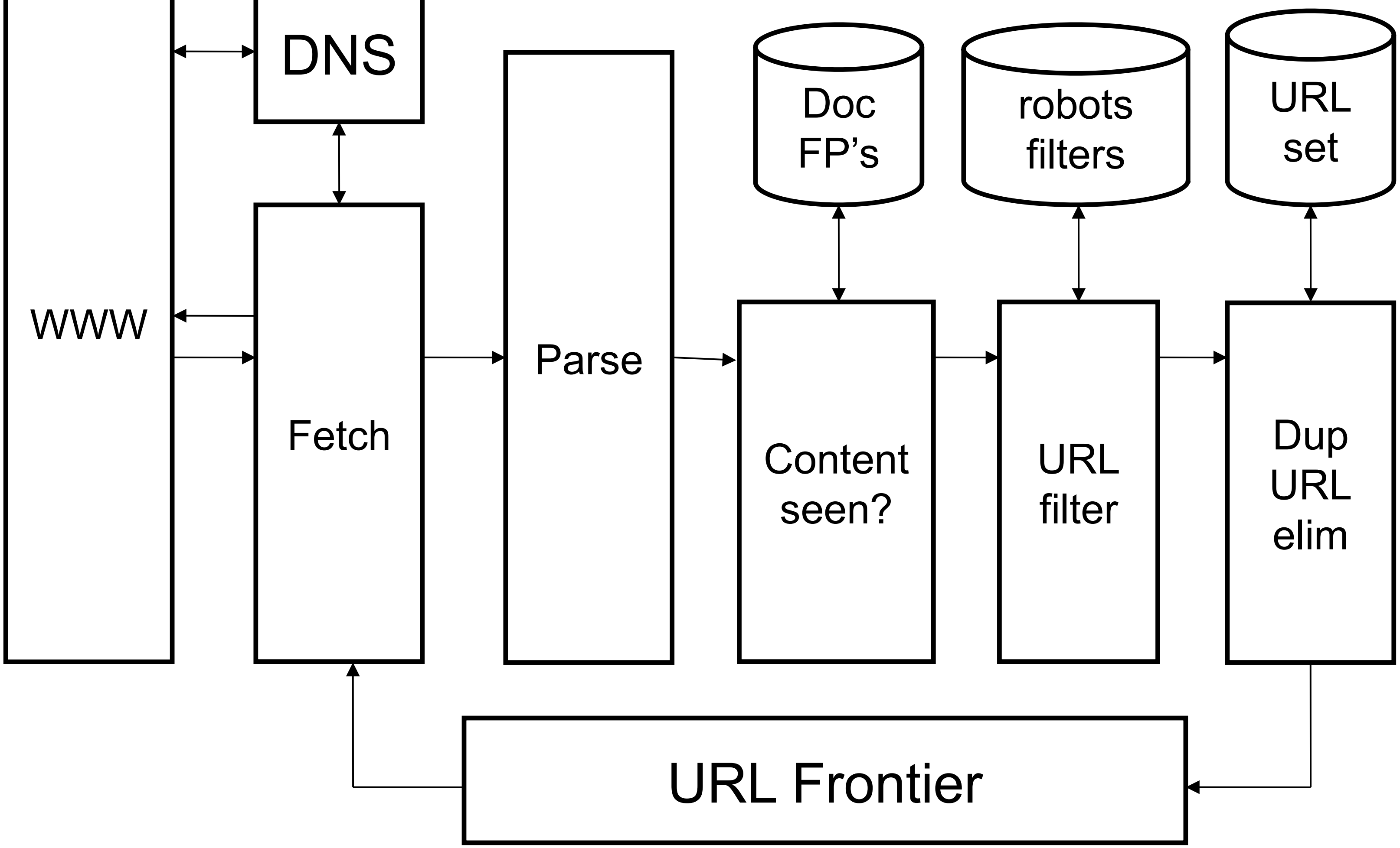
```
User-agent: *
Disallow: /yoursite/temp/

User-agent: searchengine
Disallow:
```

Processing steps in crawling

- Pick a URL from the frontier
- Fetch the document at the URL
- Parse the URL
  - Extract links from it to other docs (URLs)
- Check if URL has content already seen
  - If not, add to indexes
- For each extracted URL
  - Ensure it passes certain URL filter tests
  - Check if it is already in the frontier (duplicate URL elimination)

Basic crawl architecture



Parsing: URL normalization

- When a fetched document is parsed, some of the extracted links are *relative* URLs

- E.g., http://en.wikipedia.org/wiki/Main_Page has a relative link to /wiki/Wikipedia:General_disclaimer which is the same as the absolute URL http://en.wikipedia.org/wiki/Wikipedia:General_disclaimer

- During parsing, must normalize (expand) such relative URLs

Content seen?

- Duplication is widespread on the web
- If the page just fetched is already in the index, do not further process it
- This is verified using document fingerprints or shingles
  - Second part of this topic

Distributing the crawler

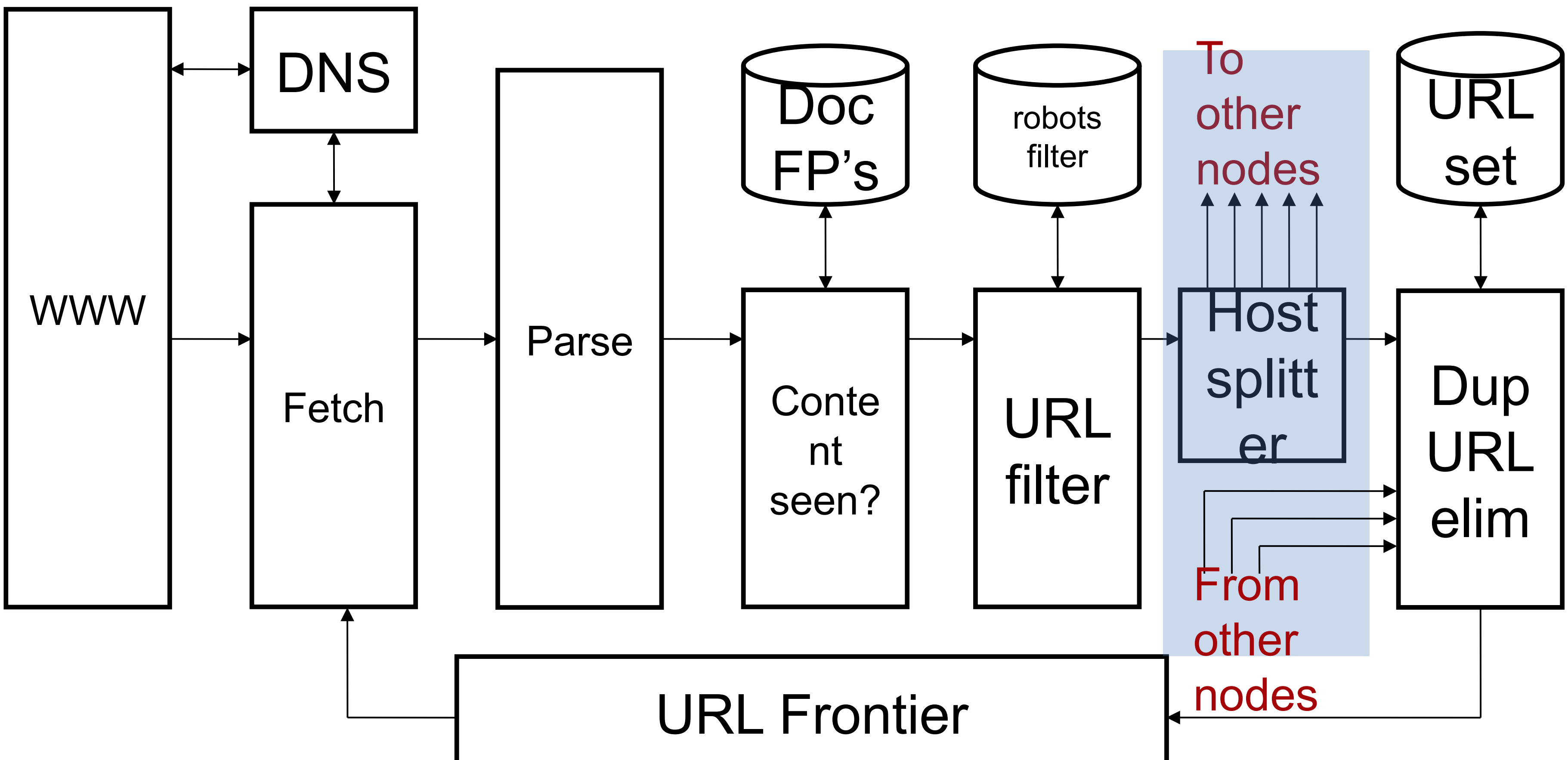
- Run multiple crawl threads, under different processes – potentially at different nodes
  - Geographically distributed nodes

- Partition hosts being crawled into nodes/machines
  - Hash used for partition

- How do these nodes communicate and share URLs?

Communication between nodes

- Output of the URL filter at each node is sent to the Dup URL Eliminator of the appropriate node



URL frontier: two main considerations

- Politeness: do not hit a web server too frequently
- Freshness: crawl some pages more often than others
  - E.g., pages (such as News sites) whose content changes often

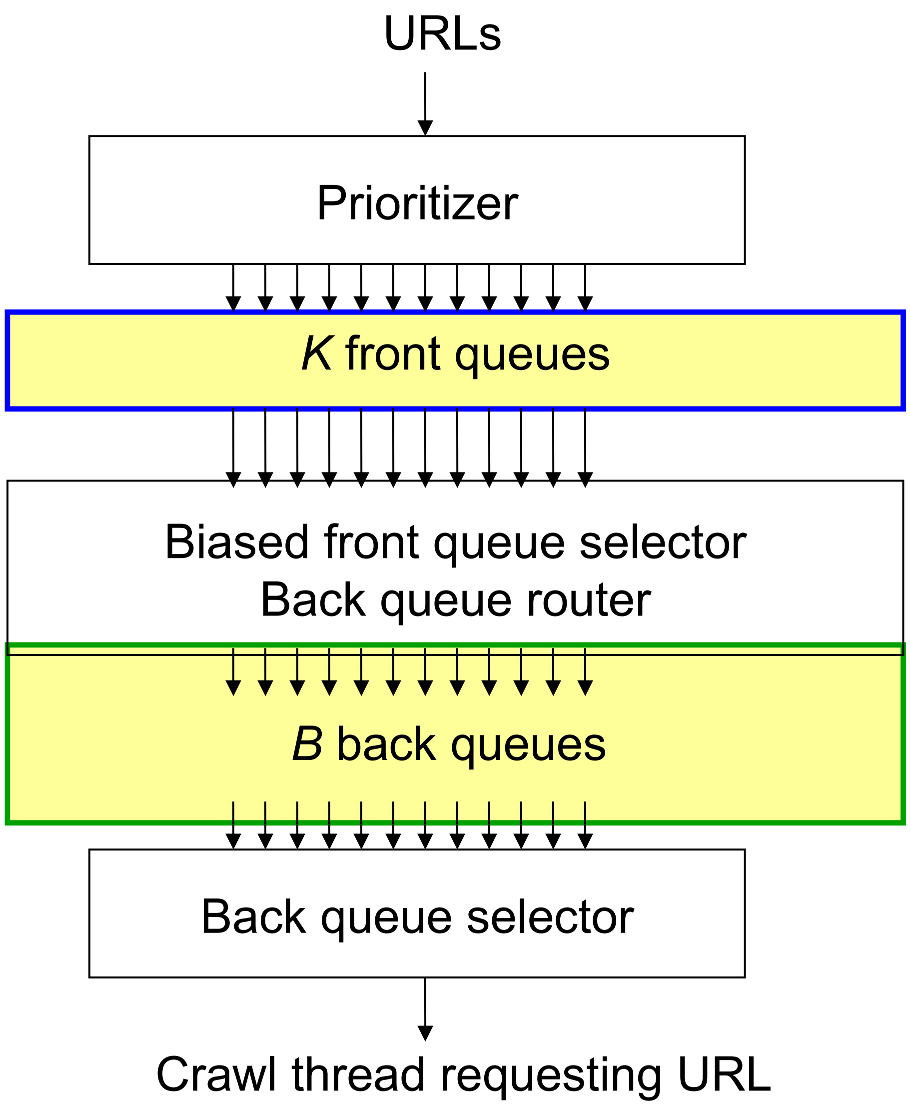
These goals may conflict each other.

(E.g., simple priority queue fails – many links out of a page go to its own site, creating a burst of accesses to that site.)

Politeness – challenges

- Even if we restrict only one thread to fetch from a host, can hit it repeatedly
- Common heuristic: insert time gap between successive requests to a host that is >> time for most recent fetch from that host

URL frontier: Mercator scheme



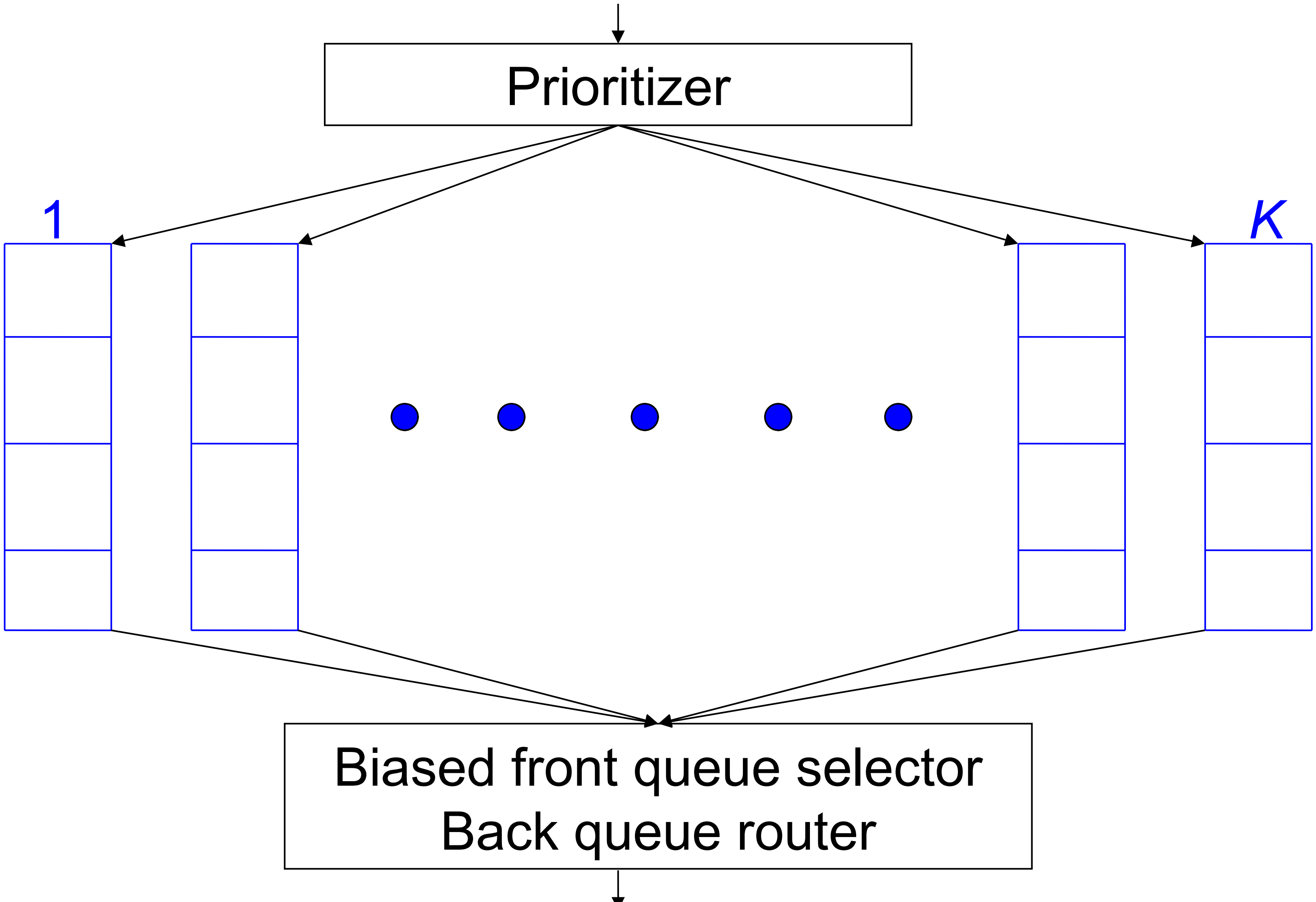
URLs flow in from the top into the frontier

Front queues manage prioritization

Back queues enforce politeness

Each queue is FIFO

Front queues



Sitemaps

- Sitemaps contain lists of URLs and data about those URLs, such as modification time and modification frequency
- Generated by web server administrators
- Tells crawler about pages it might not otherwise find
- Gives crawler a hint about when to check a page for changes

Sitemap Example

```xml
<?xml version="1.0" encoding="UTF-8"?>
<urlset xmlns="http://www.sitemaps.org/schemas/sitemap/0.9">
  <url>
    <loc>http://www.company.com/</loc>
    <lastmod>2008-01-15</lastmod>
    <changefreq>monthly</changefreq>
    <priority>0.7</priority>
  </url>
  <url>
    <loc>http://www.company.com/items?item=truck</loc>
    <changefreq>weekly</changefreq>
  </url>
  <url>
    <loc>http://www.company.com/items?item=bicycle</loc>
    <changefreq>daily</changefreq>
  </url>
</urlset>
```

Simple Crawler Thread

```
procedure CrawlerThread(frontier)
    while not frontier.done() do
        website ← frontier.nextSite()
        url ← website.nextURL()
        if website.permitsCrawl(url) then
            text ← retrieveURL(url)
            storeDocument(url, text)
            for each url in parse(text) do
                frontier.addURL(url)
            end for
        end if
        frontier.releaseSite(website)
    end while
end procedure
```

Freshness

- Web pages are constantly being added, deleted, and modified
- Web crawler must continually revisit pages it has already crawled to see if they have changed in order to maintain the *freshness* of the document collection
  - *stale* copies no longer reflect the real contents of the web pages

Freshness

- HTTP protocol has a special request type called HEAD that makes it easy to check for page changes
  - returns information about page, not page itself

Client request:
```
HEAD /csinfo/people.html HTTP/1.1
Host: www.cs.umass.edu
```

Server response:
```
HTTP/1.1 200 OK
Date: Thu, 03 Apr 2008 05:17:54 GMT
Server: Apache/2.0.52 (CentOS)
Last-Modified: Fri, 04 Jan 2008 15:28:39 GMT
ETag: "239c33-2576-2a2837c0"
Accept-Ranges: bytes
Content-Length: 9590
Connection: close
Content-Type: text/html; charset=ISO-8859-1
```

Storing the Documents

- Requirements for document storage system:
  - Random access
    - request the content of a document based on its URL
    - hash function based on URL is typical
  - Compression and large files
    - reducing storage requirements and efficient access
  - Update
    - handling large volumes of new and modified documents
    - adding new anchor text

Google BigTable

- Bigtable: A Distributed Storage System for Structured Data
- http://static.googleusercontent.com/media/research.google.com/en//archive/bigtable-osdi06.pdf

- What is Bigtable?
  - BigTable is a distributed storage system for managing semi-structured data at a large scale
  - BigTable does not support a full relational data model
  - Its scalable and self-managing

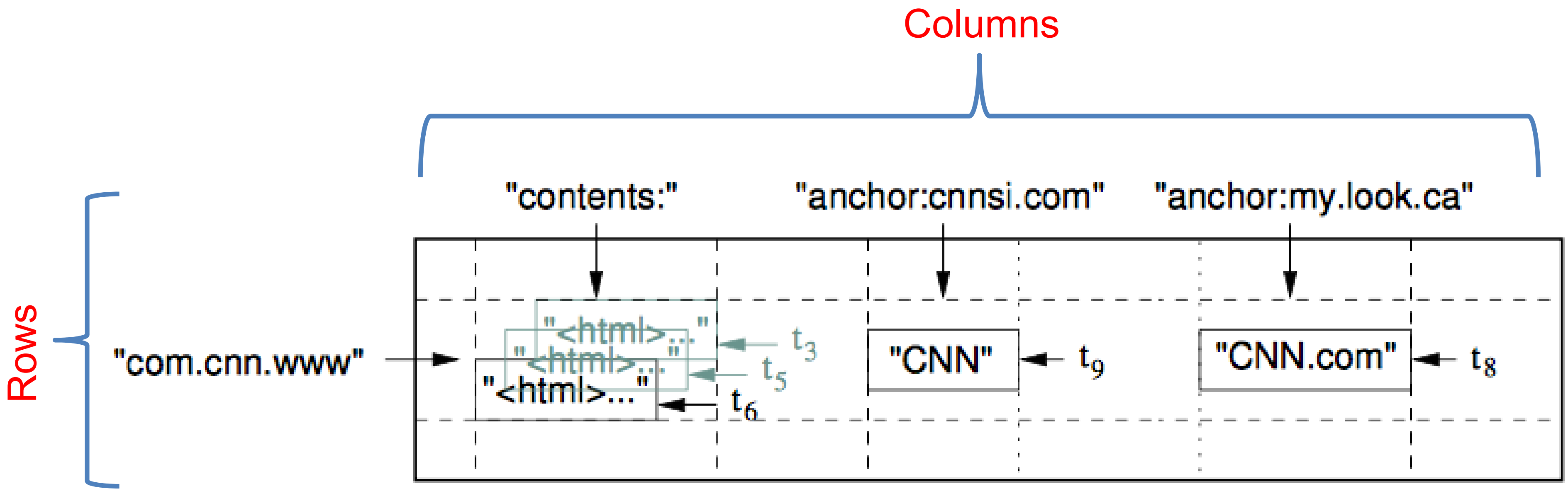
Motivation

- Lots of (semi-)structured data at Google
  - URLS
    - Contents, crawl metadata, links, anchors, pagerank
  - Per-user data
    - User preference settings, recent queries/search results
  - Geographic locations
    - Physical entities (shops, restaurants, etc.), roads, satellite image data, user annotations

- Scale is large
  - Billions of URLS, with many versions
  - Hundreds of millions of users, thousands of queries / sec

BigTable Data Model

- BigTabe is a sparse, distributed, persistent multi-dimensional sorted map.
- The map is *indexed* by a row key, column key, and a timestamp, and the value is a array of bytes (or string).
  - (row: string, column:string, time:int64) -> string



BigTable Data Model

- Row range dynamically partitioned into <u>tablets</u>
- Data in lexicographic order by row key
- Allows data locality
- Every read/write of data in a single row key is atomic.

- Column keys grouped into column families
- Each family has the same type
- Allows access control and disk or memory accounting

Tablets (1/2)

- Large tables are broken into tablets at row boundaries
  - A tablet holds a contiguous range of rows
  - ~ 100MB – 200MB of data per tablet

- A single machine is responsible for ~ 100 tablets
  - Fast recovery
    - Allows a 100 machines to each pick up 1 tablet from the failed machine
  - Fine-grained load balancing
    - Migrate tablets away from overloaded machines
    - Master makes load-balancing decisions.

Tablets (2/2)

- Read of short row ranges are efficient
  - Require communication with only a small number of machines
  - Clients get good locality for their data access

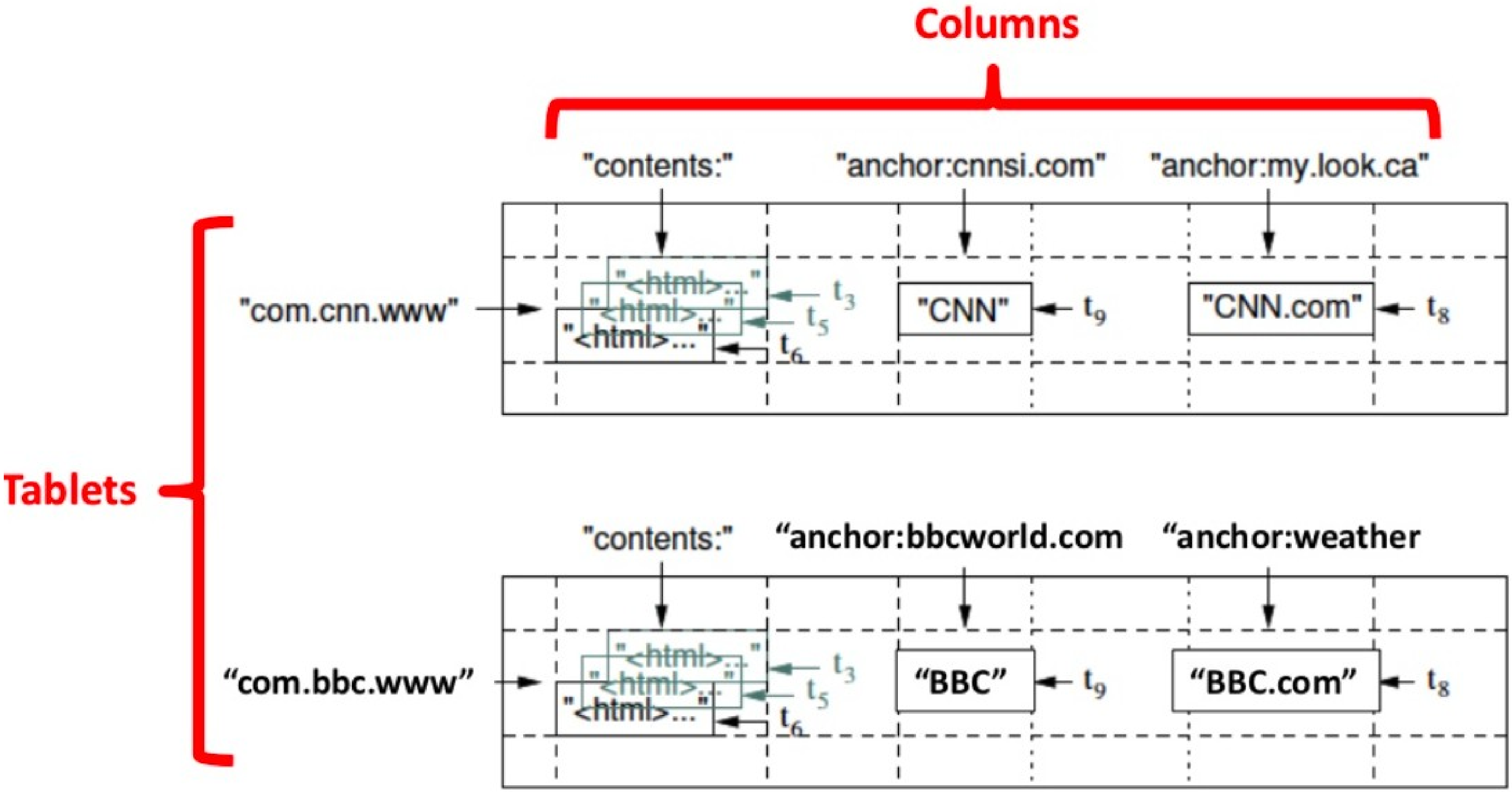
- map.google.com/index.html is stored using the key com.google.maps/index.html

  - Storing pages under the same domain near each other makes host and domain analysis more efficient

Column Families

- Column keys are grouped into sets called *column families*
- Data stored in a column family is usually of the same type
- A column family must be created before data can be stored
  - But, after a column family is created, any column key can be added to that column family
- Column key
  - **family : qualifier**

BigData Data Model



Timestamps

- Each cell in BigTable can contains multiple version of the same data

- BigTable timestamp
  - 64-bit integers
  - Time stamps can be assigned by :
    - BigTable
    - Explicitly by client applications

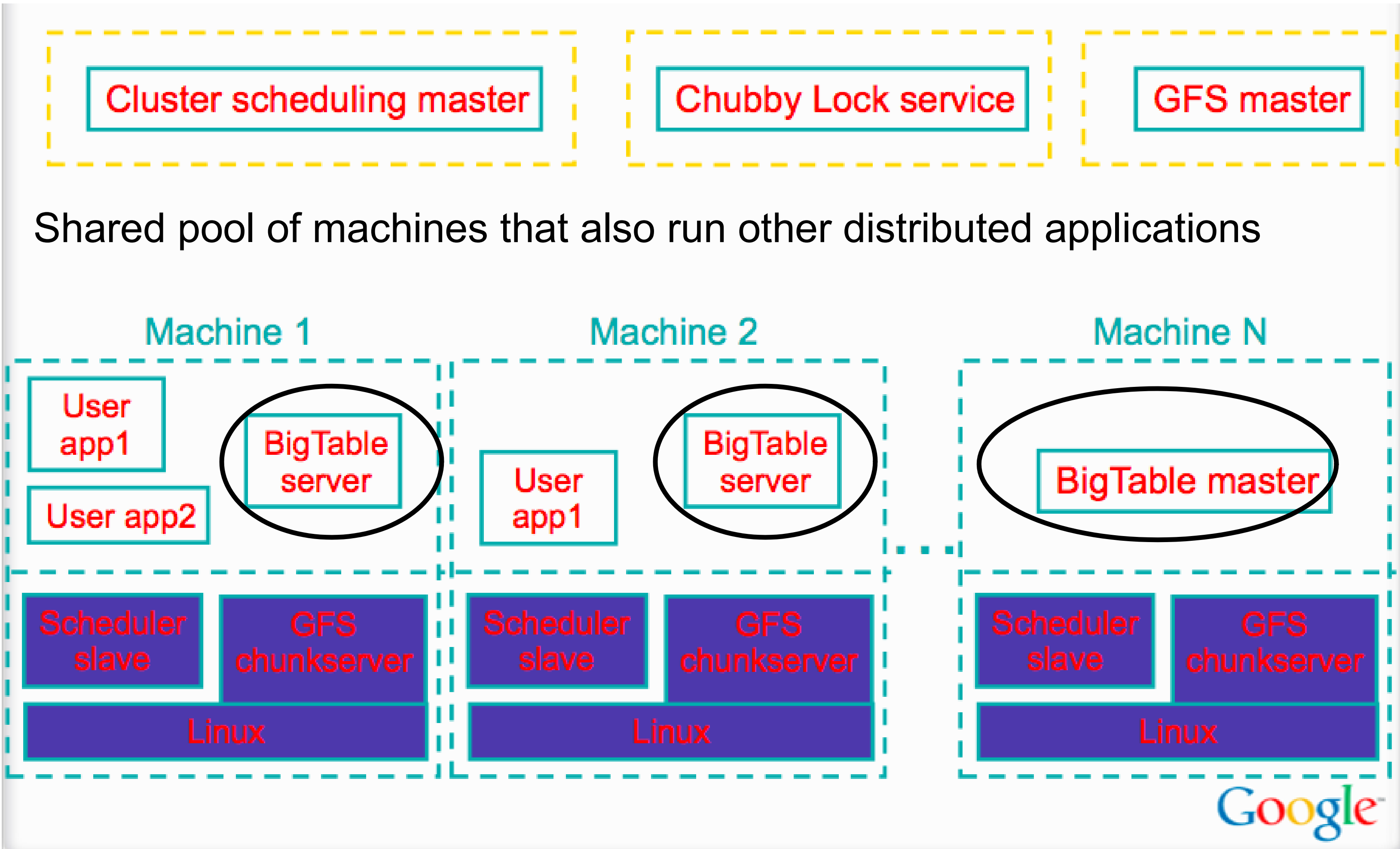
- Different versions of a cell are stored in decreasing timestamp order
  - The most recent versions can be read first

- BigTable maintains last n versions automatically

Building Blocks

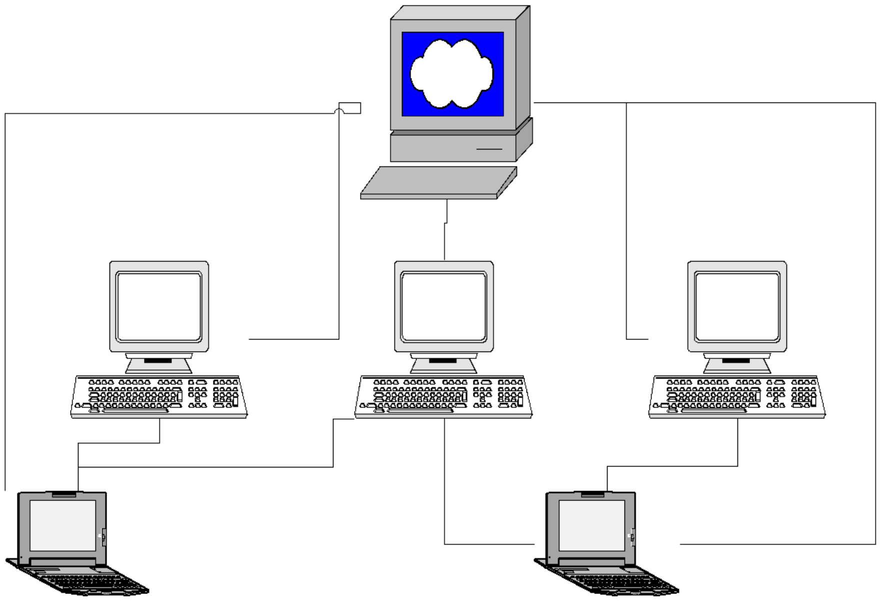
- Google File System (GFS)
  - stores persistent data (SSTable file format)
- Scheduler
  - schedules jobs onto machines
- Chubby
  - Lock service: distributed lock manager
  - master election, location bootstrapping
- MapReduce (optional)
  - Data processing
  - Read/write Bigtable data

Typical Cluster



Three Major Components

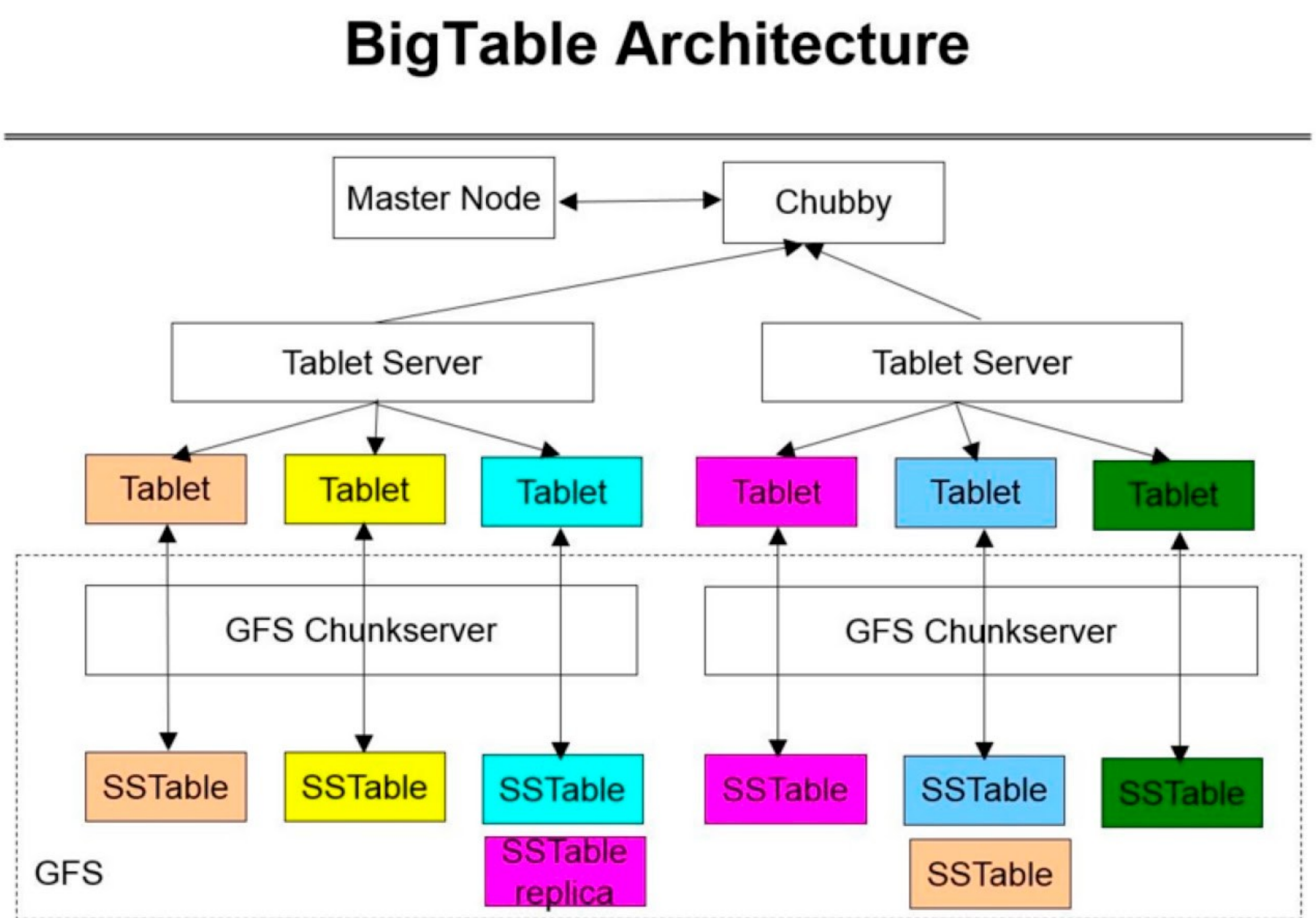
- The Master
  - One master
- The RegionServer
  - Many region servers
- The client



Components

- Region
  - A subset of a table's rows, like horizontal range partitioning
  - Automatically done
- RegionServer (many workers)
  - Manages data regions
  - Serves data for reads and writes (using a log)
- Master
  - Responsible for coordinating the workers
  - Assigns regions, detects failures
  - Admin functions

Big Picture


BigTable Architecture
Master Node
Chubby
Tablet Server
Tablet Server
Tablet
Tablet
Tablet
Tablet
Tablet
Tablet
GFS Chunkserver
GFS Chunkserver
SSTable
SSTable
SSTable
SSTable
SSTable
SSTable
SSTable replica
SSTable
GFS