CS 524 A Introduction to Cloud Computing Lecture 9

The structure of the Cloud Data Center
The three-tier enterprise architecture
The World-Wide Web infrastructure

Outline

- Physical characteristics
- The design of the application server farm and load balancing
- Traditional three-tier enterprise architecture
- Enterprise architecture in Cloud
- The World-wide Web architecture
- HTML pages—static and dynamic
  - Client-side scripting
  - Server-side scripting
  - Hybrid
  - Data structure representation: XML and JSON
  - Overview of the AJAX technologies
- HTTP
  - Elements of the protocol
  - Caching
  - Stateless servers and cookies
    - Storing the session state
    - User tracking

Terminology: The Three Data center Components

- *Compute*
- *Storage*
- *Networking*

COMPUTE

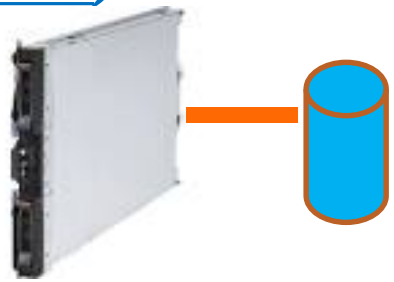
- High-performance, multicore computers [also called *servers*—but we should not confuse these with *web servers* or other application servers (which are *processes*)
- Come in the form of
  - *Rack-mounted servers* (inserted horizontally into an 19-inch-wide rack, the height measure units (U), as in 1U, 2U, 3U.

  - *Blade servers* (smaller, typically **no** cooling equipment or devices). Need *chassi*

Blade servers are preferable for the Cloud data centers

Storage

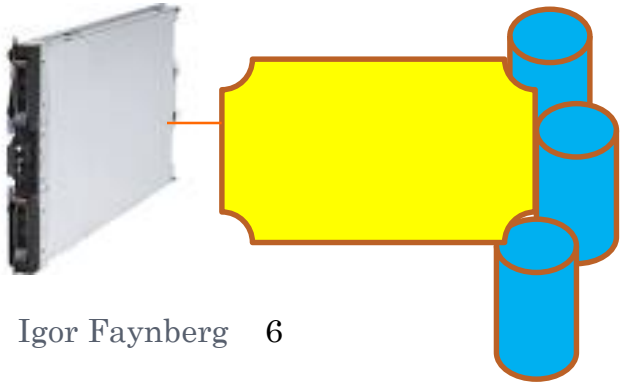
- ***Direct-attached storage (DAS)***
  - hard-disk drive
  - a point-to-point link



- ***Network-Attached Storage (NAS)*** and
- ***Storage Area Network (SAN)***

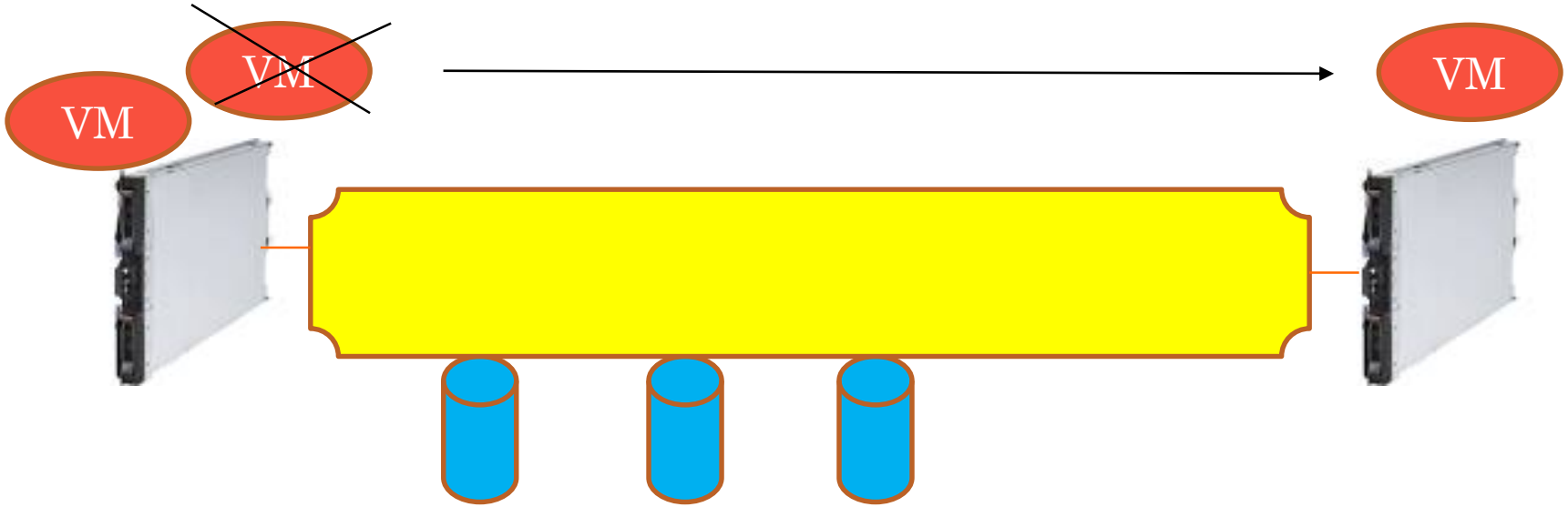
both rely on a dedicated network for storage access

A unit in NAS is a file or an object
but
A unit in SAN is a disk block



What is better for the cloud, DAS or NAS?

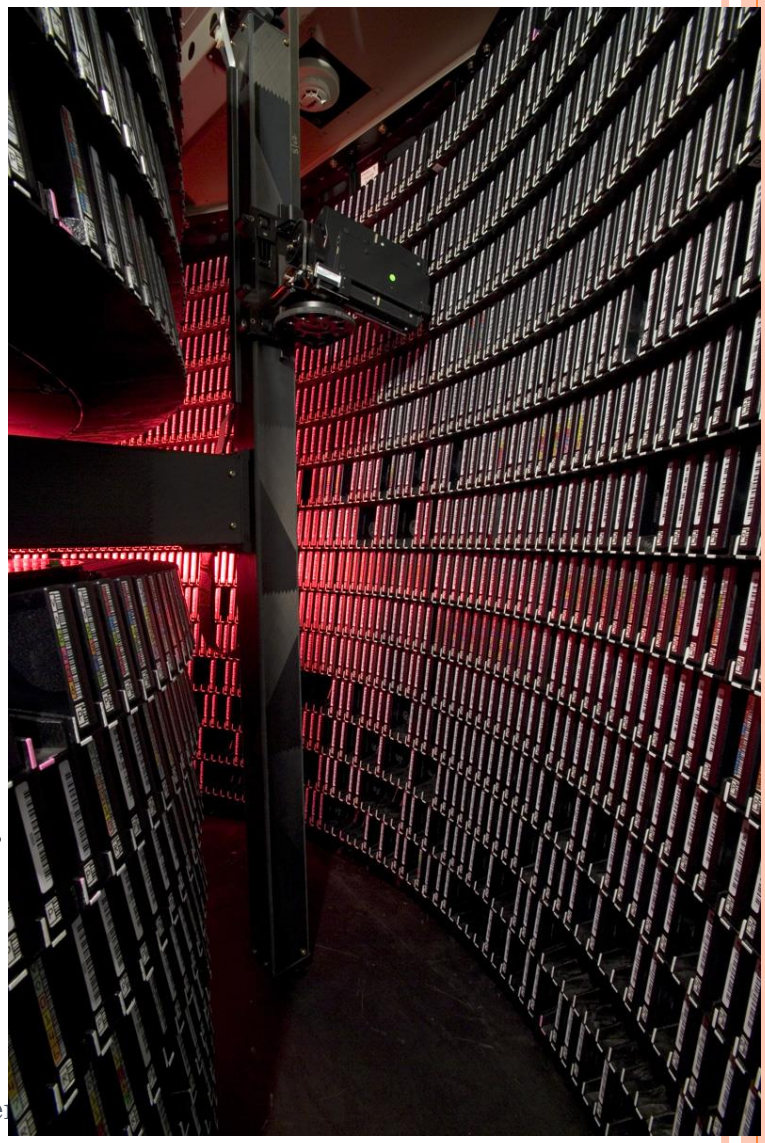
Consider the VM migration case:



Storage Classification Terminology

- *Online* storage is always accessible (just as our PC files on a disk)
- *Offline* storage (used for archiving) needs to be physically mounted before access
  - Optical juke boxes
  - Magnetic tapes
  - Solid-state (think memory sticks)
  - …

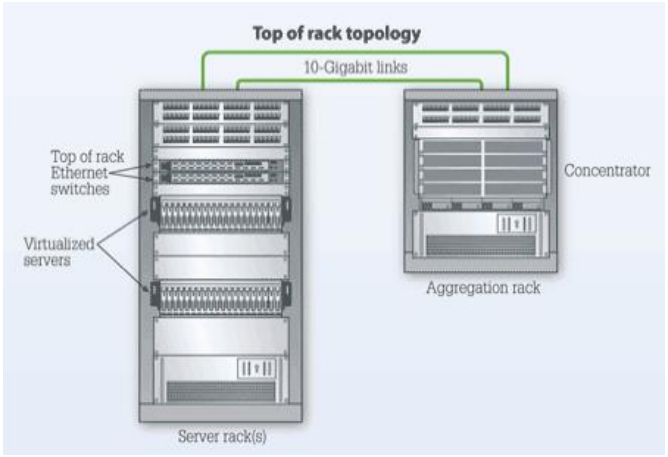
The current practice is to use robots for that.



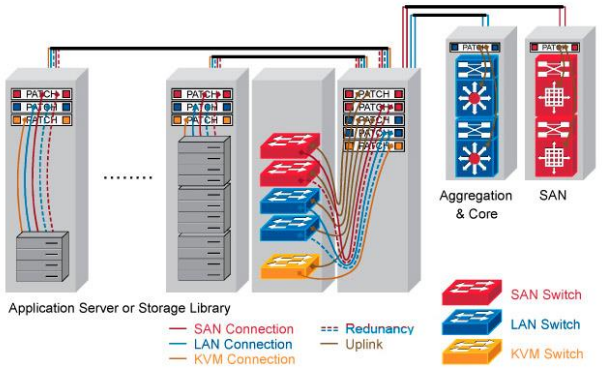
Amazon robotic tape library
Source: www.extremetech.com

Networking (or… what to do with all these cables?)

- *Top-of-Rack (ToR)* approach
  - Each rack has a switch at the end to which all servers connect

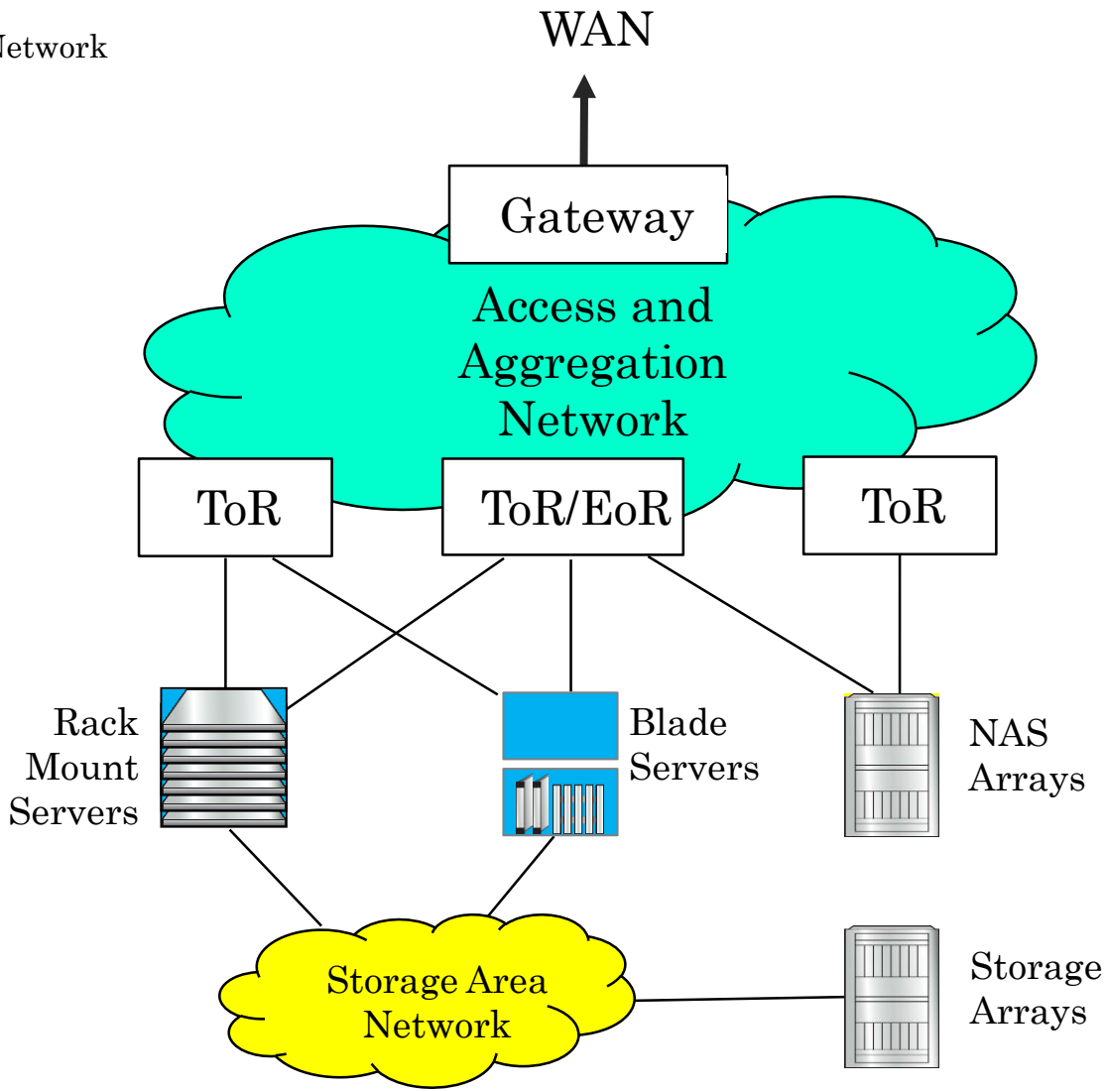


- *End-of-Row (EoR)* approach
  - Each row has a switch connecting all its servers (may need fiber connection)

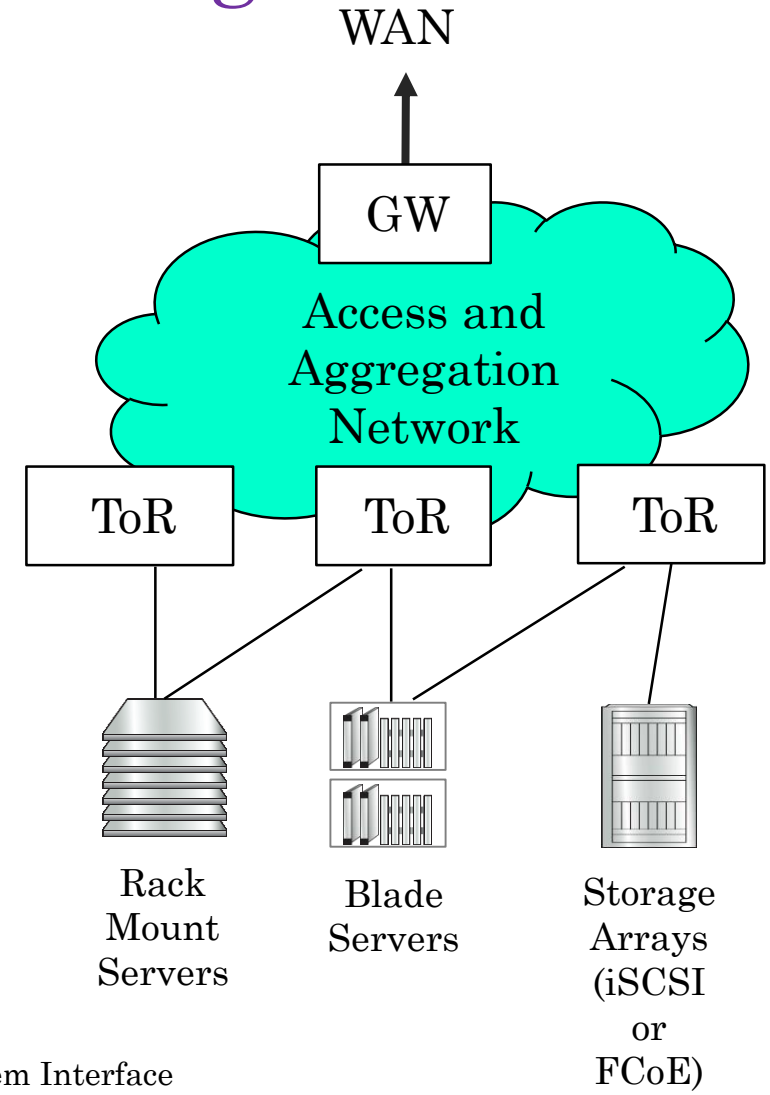


A Traditional Data Center

EoR: End of Row
GW: Gateway
ToR: Top of Rack
WAN: Wide-Area Network



The next-generation data center

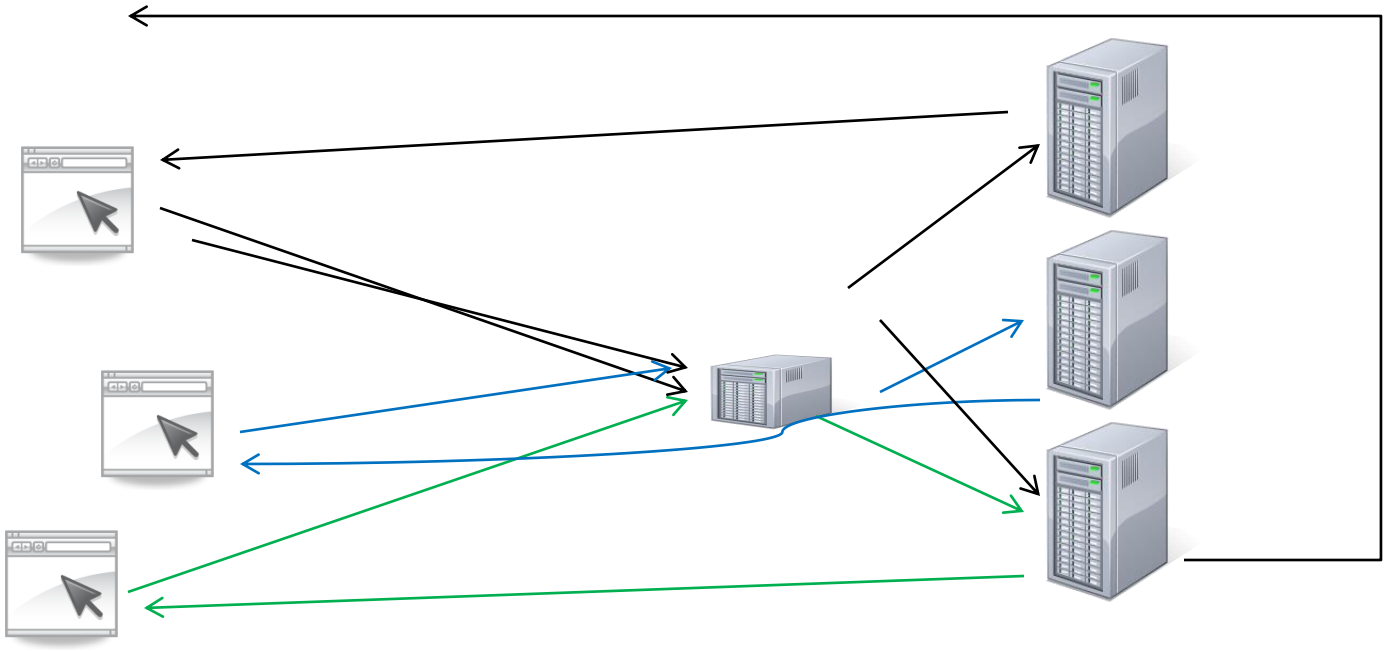


FCoE: Fibre Channel over Ethernet
GW: Gateway
iSCSI: Internet Small Computer System Interface
ToR: Top of Rack
WAN: Wide-Area Network

No separate storage network

The Design of an Application Server Farm and Load Balancing

- Need: A one-machine server can quickly be overloaded when it hosts a popular page
- Solution: Introduce several *identical* servers, with the *front end* distributing the load among them

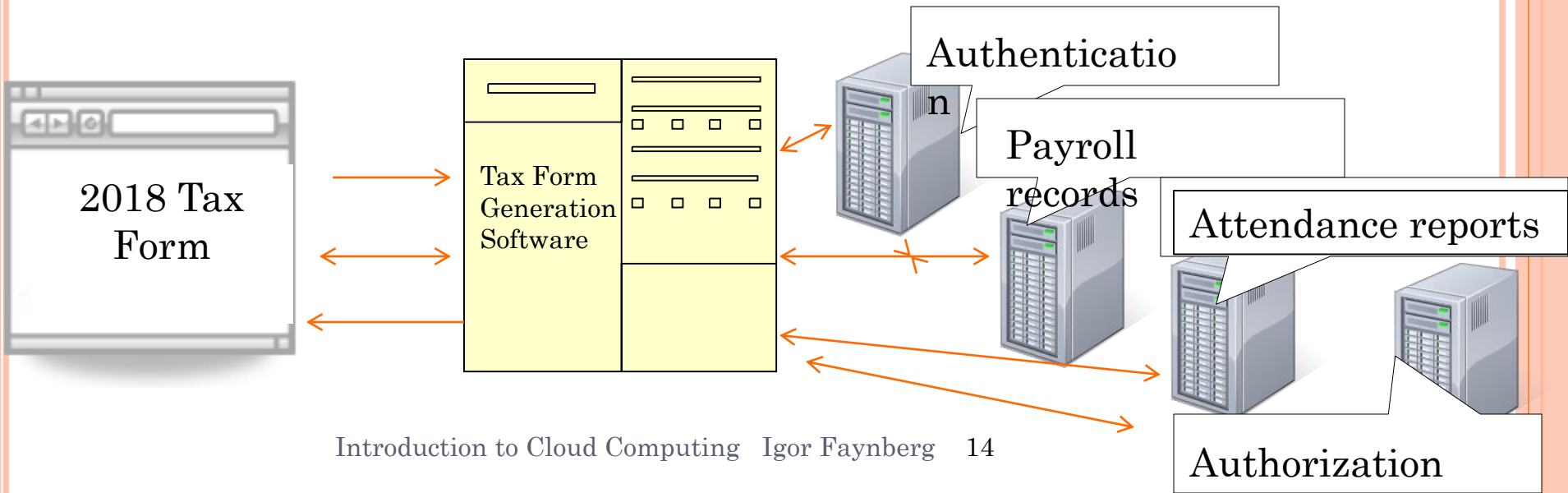


Requirements and (more general) solutions

- Requirement: *Statelessness* of the server (state kept at the client)
- Solutions for distribution:
  - Use DNS to return the servers' addresses (e.g., in round-robin)
  - Make the front end a layer-2 switch or a router; broadcast the request to all servers and let them choose to respond (for example, based on the original IP address)
  - Make the front end a layer-7 *midbox* and execute any scheduling policy (round-robin, weighted round-robin, or content-based)

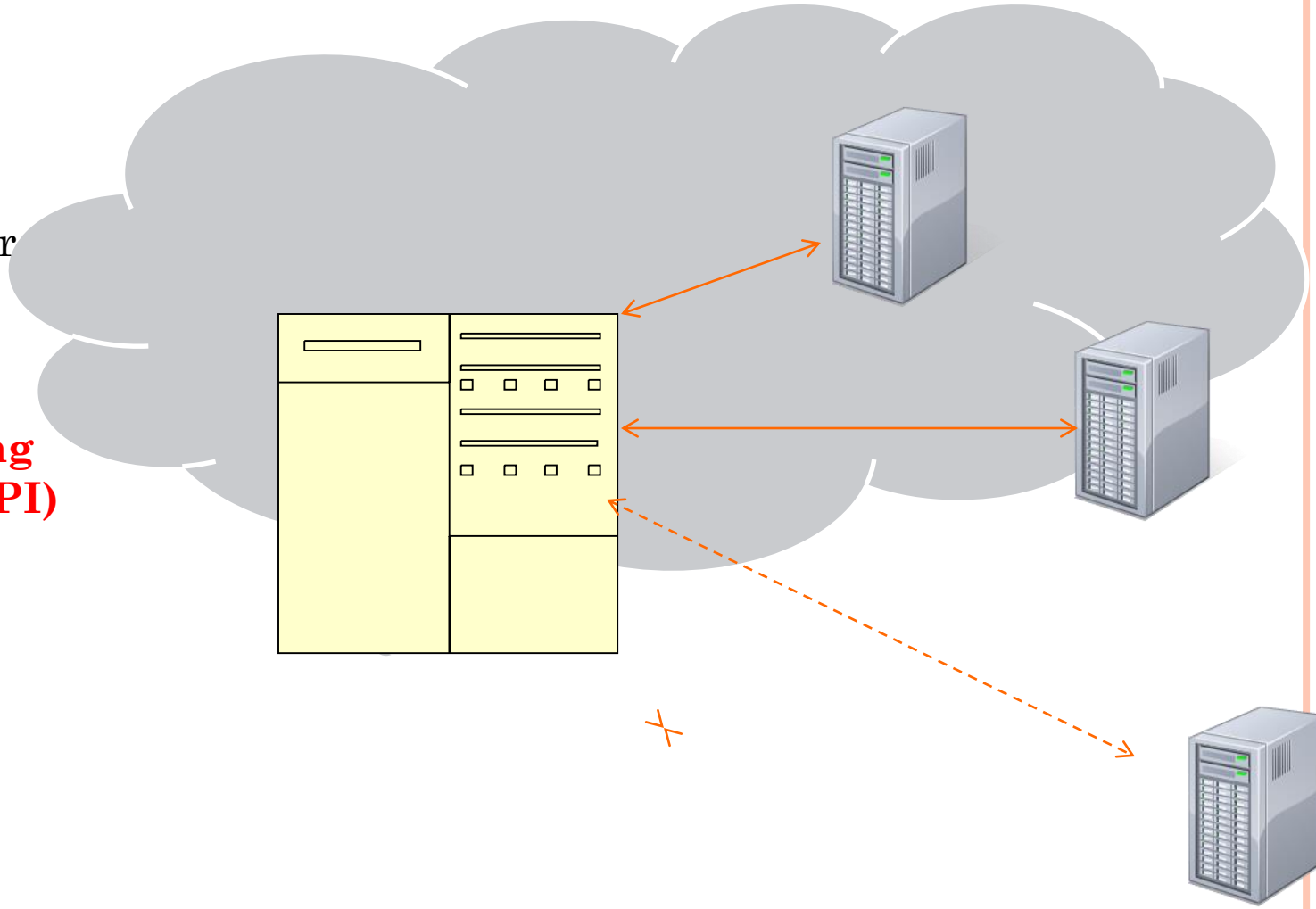
Three-tier enterprise architecture

- Tier 1(a client) allows to structure queries and the responses with a simple graphical interface. When running in the browser, it is often called *a thin client*.
- Tier 2 (a server) provides the *business processes logic* and the data access.
- Tier 3 (often a set of hosts running database software) provides the actual data.



Additional Development: (for better or for worse)

The Web Application Protocol has become a mechanism for providing the remote **Application Programming Interface (API)**



…In Cloud there is the new demand for scalability

Cloud service orchestration

- Combination of architecture, tools, and processes to deliver a defined service
- Connecting and automating the work flows
- Ensuring the fulfillment
- Producing accurate billing

The WWW Model

Web *pages* with *resources* linked through *hypertext*

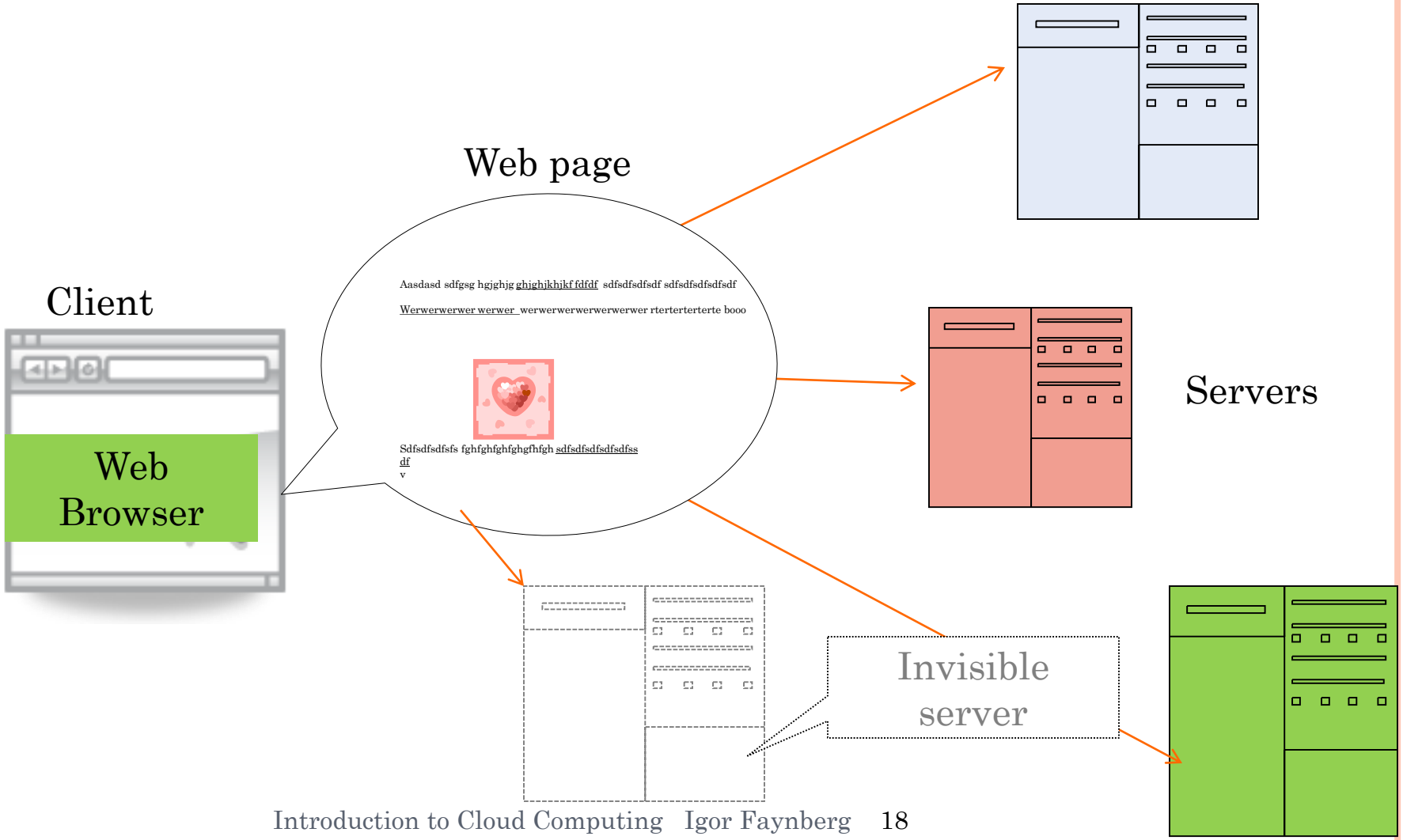
- Foreseen by MIT's Vannevar Bush in his article in *The Atlantic* in July, 1945)

> "... The human mind … operates by association. With one item in its grasp, it snaps instantly to the next that is suggested by the association of thoughts, in accordance with some intricate web of trails carried by the cells of the brain. It has other characteristics, of course; trails that are not frequently followed are prone to fade, items are not fully permanent, memory is transitory. Yet the speed of action, the intricacy of trails, the detail of mental pictures, is awe-inspiring beyond all else in nature."

http://www.theatlantic.com/magazine/archive/1945/07/as-we-may-think/3881/2/

- Realized in present from (more or less) by Tim Berners-Lee at CERN in 1989
- First commercialized by Marc Andreessen in *Mosaic* (1993), at which point the ".com" era started

Hyperlinks and the Hyper-Text Transfer Protocol



The way to look at it

- Each accessible item on a page (and a page itself) is a *resource*
- Each resource is assigned a *Uniform Resource Locator (URL)*, which consists of three parts:
  - *Protocol* (or *scheme*)
  - *DNS name of the host where the resource is located*
  - *The "path" to the resource* (file to read, or program to execute)

Protocol	Host	Path
http://	https://sit.instructure.com/	courses/36066/modules/items/897188

But it is a bit more complex

- A resource is actually assigned a *Universal Resource Identifier (URI)*, as defined in RFC 3986 (http://tools.ietf.org/html/rfc3986)
- A URI can be either
  - a *Universal Resource Name (URN)*, which uniquely identifies a resource without specifying its location (think passport) ; or
  - a *URL*, which actually specifies the resource location (think GPS coordinates); or
  - Both a URN and URL (think driver's license).

The definition says *nothing* about how to interprete a URI or act on the resource (what if the name ends with ".exe"?)

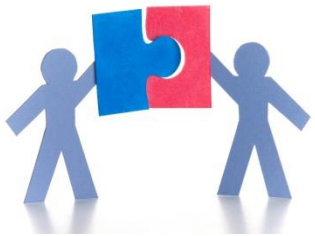
Examples

A single resource can be identified by multiple URIs

- ftp://ftp.ietf.org/rfc/rfc3986
- http://www.ietf.org/rfc/rfc3986.txt
- ldap://[2001:db8::7]/c=GB?objectClass?one
- mailto:Cloud.Administrator@example.com
- news:comp.infosystems.www.servers.unix
- tel:+1-816-555-1212
- telnet://192.0.2.16:80/
- urn:oasis:names:specification:docbook:dtd:xml:4.1.2

The browser

- (is becoming more and more *an operating system*)
- parses the resource name, interprets, and fetches the resource and then deals with it by using
  - *plug-ins* that "work inside" the browser (the practice that will soon disappear)



  - *helper applications* that run on their own (like MS Power point)
  - the code embedded in the page (more on that later)
- Because e-mail preceded the web, the application types for the browser are specified using the *Multi-purpose Internet Mail Extensions (MIME)* format

The hyperText Markup Language (HTML)

- evolved from TeX (and *mm)* languages developed before graphic interface became available
- is standardized in the World-Wide Web Consortium (W3C) (http://www.w3.org/)
- will support (in its latest, HTML 5.0 version http://dev.w3.org/html5/spec/introduction.html#history-1)
  - Hyperlinks, images, lists, equations, etc.
  - eXtensible Mark-up Language XML representations
  - Forms (thus allowing output to be collected and sent back)
  - Scripting
  - Cascading Style Sheets (CSS) (rules for tagged content appearance, already in HTML 4.0)
  - Video and audio
  - Browser storage
  - In-line vector graphics (in addition to bitmap image formats—JPEG and GIF)

Strict definition of structure and representation

- eXtensible Mark-up Language (XML) defines *tagged* structures (to help uniform searching)

```
<?xml version="1.0"?>
<course>
    <catalogue number> CS 524 </catalogue number>
    <name> Introduction to Cloud Computing </name>
</course>
<course>
    <catalogue number> CS 520 </catalogue number>
    <name> Introduction to Operating Systems</name>
</course>
```

- *eXtensible Stylesheet Language Transformations (XSLT)* http://www.w3.org/TR/xslt20/#xslt-mime-definition defines how an XML document can be transformed into another form (e.g, HTML)

Other uses of XML

- As a language for specifying objects in inter-process communications

  Example: *Simple Object Access Protocol* (SOAP) for a remote procedure call over HTTP

- As a language for specifying all kinds of things (network management objects, ingredients in a drink, last wills, musical scores, etc.)
- As a re-formulation of HTML

  eXtended HyperText Mark-up Language (XHTML)

JavaScript Object Notation (JSON)

- is a "lightweight, text-based, language-independent data interchange format," which is actually close to *JavaScript*
- is perceived as a competitor to XML (there are compilers both ways)—take a look at http://www.json.org/example.html
- is derived from the *ECMAScript Programming Language Standard*. European Computer Manufacturers Association, "ECMAScript Language Specification 3rd Edition", December 1999, http://www.ecma-international.org/publications/files/ ecma-st/ECMA-262.pdf.

Form processing in HTML (Going the other way, too!

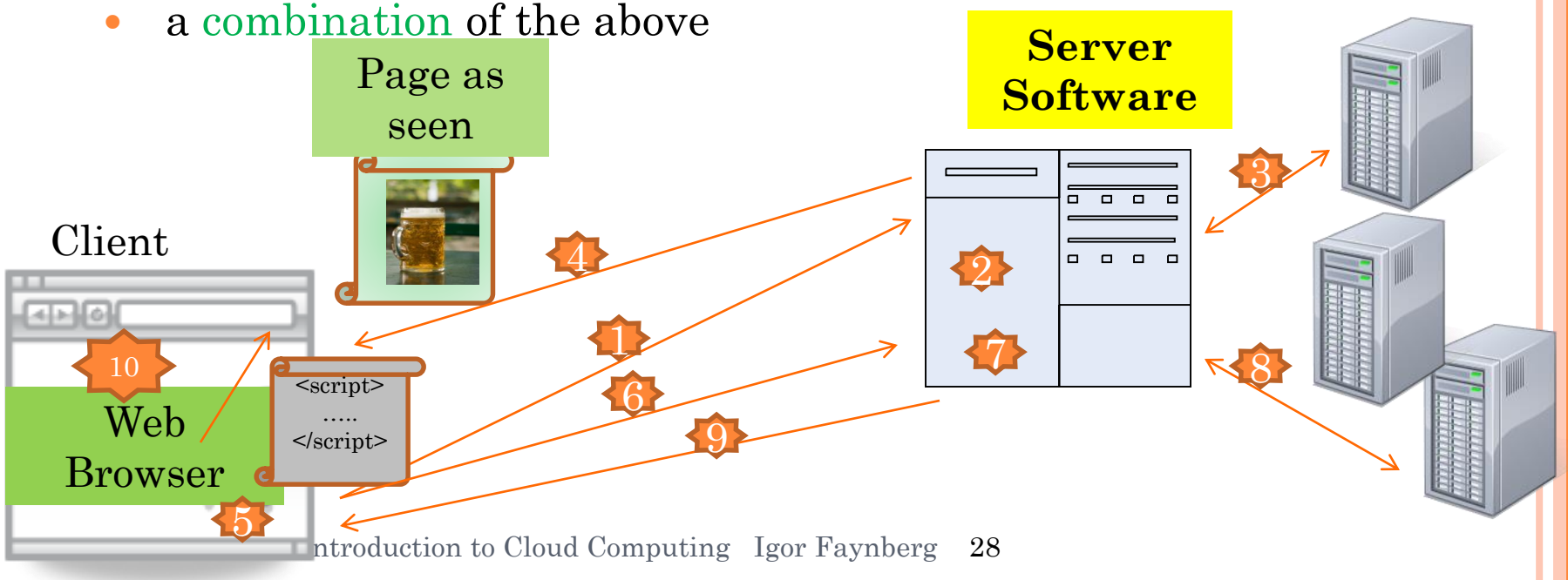
- Has started in HTML 2.0
- Uses <form>, </form> tag
- Can use *radio buttons* (exlusive)
  - Bitter Ale <input name = “beer_type” type=radio value=“B”>
  - Lager <input name = “beer type” type=radio value=“L”>
  - Porter <input name = “beer type” type=radio value=“P”>
  - Stout <input name = “beer type” type=radio value=“S”>
- Can use *check-boxes* (non-exclusive)
- Has a *submit* button, which sends the form back in plain ASCII text

  “&” for field separation, “+” for space. An example:

  Customer=me&address=here&order=B&shipping=immediate&boxes=10
  0

Dynamic Web Pages

- The resources may be of much more than just static (i.e., files) nature
- The dynamic content is generated
  - by the software running on the cloud servers; or
  - by the software running on the client's host (controlled by the browser); or
  - a combination of the above



Client Scripting

- Enabled by the <script> tag and interpreted by the browser:
  - *JavaScript*
  - the *Microsoft Visual Basic Scripting Edition language (VBScript)*
- Embedded between *<applet> and* </applet> tags and run on a virtual machine controlled by a browser

  Java Applets executed on Java Virtual Machine (JVM) [ubiquitous]
- Real processor machine-level binary code (very dangerous)

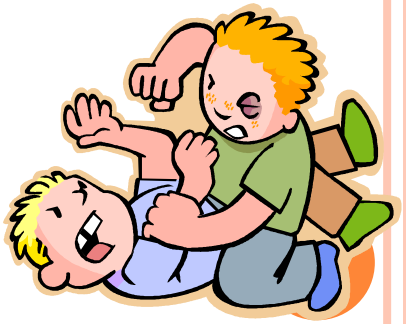
  Microsoft ActiveX controls—compiled to x86 machine code

Server Scripting

- Need: Form processing
- *Common Gateway Interface (CGI)*
  - a standardized interface to back-end (*Perl*—about the only survivor carrying *Pascal* genes—is a typical scripting language; another one is *Python*)
  - adherent to a convention of keeping the scripts in the *cgi-bin* directory (specified in the URL)
  - (typically) used to parse the forms and return results
- *Hypertext Preprocessor (PHP, .php)*
  - (an original acronym of *Personal Home Page*)
  - scripting language that can be embedded into HTML
  - a powerful language for interfacing databases
  - designed to work with *Apache*
- *JavaServer Pages (JSP, .jsp)*

  The *Java* language instead of PHP
- *Active Server Pages .NET (ASP.NET, asp.net)*

  Used to be Microsoft *Visual Basic* instead of PHP; now .NET



A PHP example: The Original Form

```
<html>
<body>
<form action="cases.php" method="post">
<p>
    Please enter your name:
    <input type="text" name="name">
</p>
<p>
     How many cases of beer do you want?
 <input type="text" name="cases">
</p>
<input type="submit">
</form>
</body>
</html>
```

A PHP example: The script "Cases.php"

```
<html>
<body>
<h1> Confirmation </h1>
<p>
    Dear <?php echo $name>,
<p>
    We are sending you <?php echo $cases+1> cases of
  Guinness.
</p>

</body>
</html>
```

A PHP example: The new Page

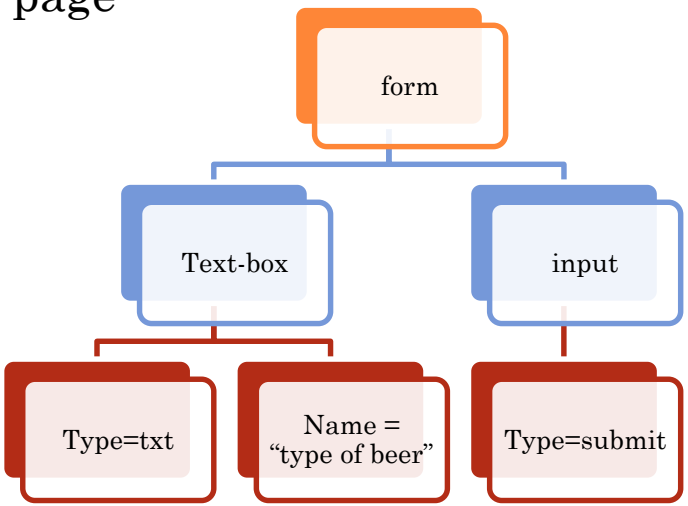
```
<html>
<body>
<h1> Confirmation </h1>
<p>
Dear Alice,
<p>
We are sending you 11 cases of Guinness.
</p>

</body>
</html>
```

The Document Object Model (DOM)
http://www.w3.org/DOM/

- "a platform- and language-neutral interface that will allow programs and scripts to dynamically access and update the content, structure and style of documents"

- the way JavaScript represents the browser state and the HTML page it contains

- used for dynamic updates to a *part* of a page without changing the whole page



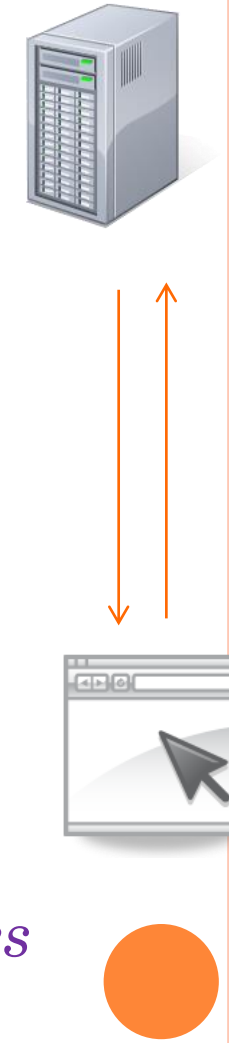
Combining the Client- and Server Scripting: *AJAX*

Asynchronous Javascript and XML (AJAX) is a set of technologies:

- JavaScript for programming
- HTML to present information
- XML to specify data structures
- *Document Object Model* (DOM)
- A mechanism (asynchronous I/O) for programs to send and receive XML data without blocking the display while waiting for a response

  Think about moving a mouse over objects on the screen, causing them to be downloaded

The HyperText Transfer Protocol (HTTP)



- Request/Response protocol defined in RFC 2616 (http://www.ietf.org/rfc/rfc2616.txt)
  - Headers in ASCI text
  - Contents encoded (MIME-like)
- Running over TCP (now using persistent connections)
- Providing, in fact, itself a transport mechanism for various applications beyond the Web for which it was intended
- Geared toward object-oriented paradigm, specifying eight *methods* that act on the *resources* identified in the *Request-URI*

HTTP Request Methods

- *GET* obtains the resource representation (may be *conditional* depending on the "if"-headers)
- *HEAD* obtains only the HTTP header (typically, to check *meta-information*) rather than the body associated with the resource
- *POST* "*requests that the origin server accept the entity enclosed in the request as a new subordinate of the resource*"
- *PUT* requests that the resource be stored
- *PATCH (standardized in 2010: see RFC 5789,* http://tools.ietf.org/html/rfc5789) modifies the resource representation partially
- *DELETE* requests that the resource be deleted
- *OPTIONS* requests information about the communication options available on the request/response chain associated with the resource
- *TRACE* initiates a loop-back of the original message
- *CONNECT* requests connection to a proxy that can be a tunnel

HTTP Responses

- *1xx*: Information
- *2xx*: Success with data (if present)
- *3xx*: Redirection
- *4xx*: Client error (also request for authentication, as in *401*, accompanied by a challenge)
- *5xx*: Server error

Example HTTP Headers

In requests only:

- *User-Agent*: browser information
- *Accept:* types of pages the client can handle
- *If-Modified-Since:* check for freshness
- *If-None-Match:* check for freshness of a number of cached pages based on their *ETags*
- *Authorization:* client's credentials
- *Referer [sic!]:* the URL from which the request was redirected
- *Cookie:* the cookie (http://www.ietf.org/rfc/rfc2109.txt) previously set by the server

In responses only:

- *Set-Cookie*: the cookie to store
- *Content encoding*
- *Server:* server information
- *Last-Modified:* time of the last change of the resource

In both requests and responses:

- *Date:* date and time the message was sent
- *Cache control:* directives for dealing with cache
- *ETag*

Caching (in the browser)

- Easy if the *Expires* header was present when the page was returned, but for a server it may be difficult to predict expiration
- Uncertain when heuristics (such as guessing based on the *Last-Modified*) are used
- Improved with
  - the *Conditional Get*
  - use of the *ETag* header
    - Returned with each new page
    - Supplied by a client among other ETtags with the *If-None-Match* header

***ETag* carries a hash of the page, which serves as a tag to its content**

THE PROCEDURE

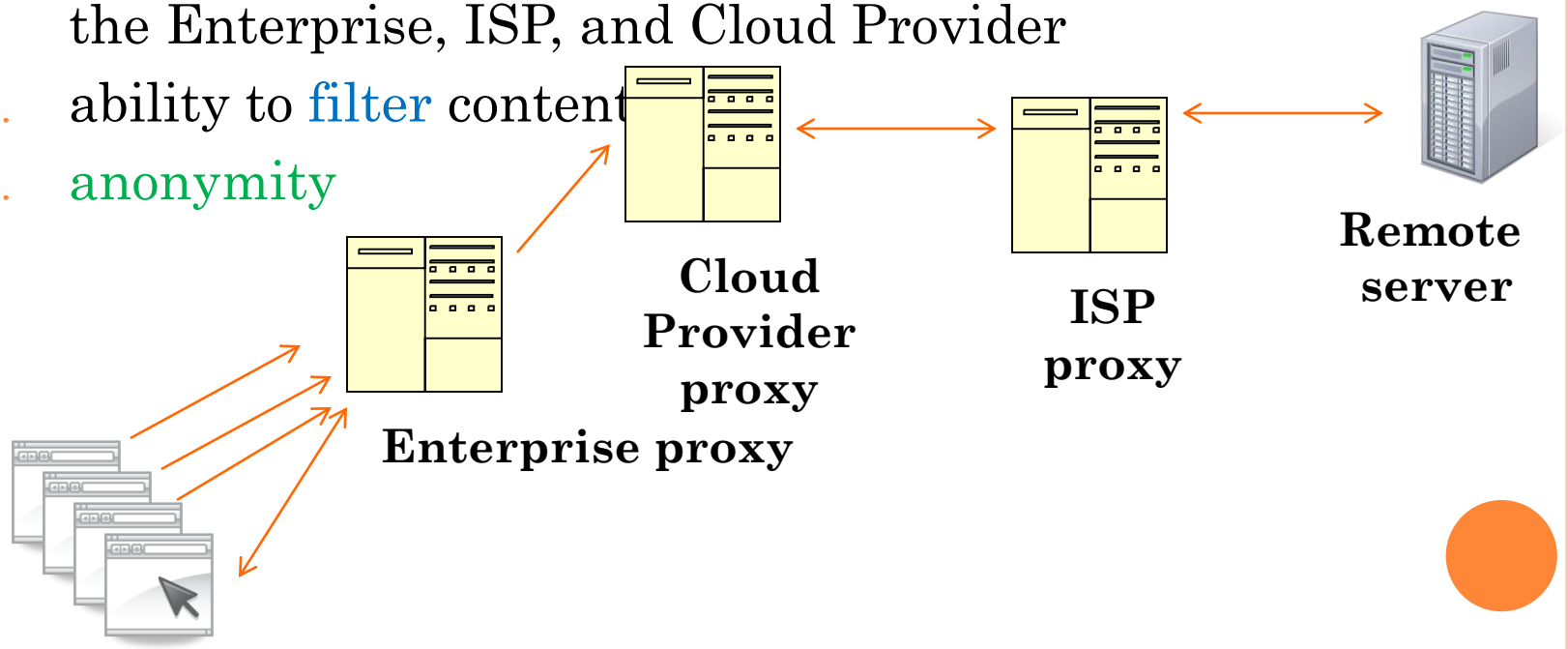
- In the response to GET *x*, the web server returns the representation of the resource *x* along with its ETag value *H(x)*

The HTTP "ETag" field has, say the value of *FC67A554632FAEDDC*

- The client may cache the resource along with its ETag.
- Later, the client attempts to retrieve the same resource again, and so it sends its previously saved copy of the ETag along with the request in a "If-None-Match" field.
  - If-None-Match now is *FC67A554632FAEDDC*. On this subsequent request, the server may now compare this with the ETag for the current version of the resource. If the ETag values match, meaning that the resource has not changed, then the server may send back a very short response with an HTTP **304 Not Modified** status use that.
  - If the ETag values do not match, then a full response including the resource's content of *x* is returned, and the client replaces the value of Etag with the new one.

Caching in Web Proxies

- Strategy: share cache among multiple users in another server called *a Proxy*
- Benefits:
  1. reduced use of bandwidth (and networking fees) for the Enterprise, ISP, and Cloud Provider
  2. ability to filter content
  3. anonymity



Cookies: Session State and User Tracking

- Cookies (RFC 2109 http://www.ietf.org/rfc/rfc2109.txt and RFC 2965 http://www.ietf.org/rfc/rfc2965.txt) allow the server to save the session state at the client
- Cookies can be (and are!) used for tracking
- Headers: *Set-Cookie* (server); *Cookie* (by client)