Dr Markus Hagenbuchner
markus@uow.edu.au

CSCI319

- Distributed Systems -
Distributed Web-based Systems
Study objectives:

1. Understand the role of distributed web based systems

2. Explain how the eight design principles can be adopted in the realization of distributed web based systems.

3. Understand how scalability is achieved in apache and in content aware web server clusters.

4. Explain persistent connections and content aware caching.

5. Understand the HTT Protocol
Content

• Basic Web based systems
  – Web documents and dynamic content

• Distributed Web servers
  – How to distribute services?
  – Architectures (i.e. server cluster)
  – Client and server processes
  – Also: Communication, Naming, Fault tolerance, Synchronization, Consistency and replication, Security.
Traditional Web-Based Systems

The overall organization of a traditional Web services:

1. Get document request (HTTP)
2. Server fetches document from local file
3. Response
Web Documents

A Web server specifies the type of a returned document by MIME type. Some common MIME types and subtypes:

<table>
<thead>
<tr>
<th>Type</th>
<th>Subtype</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Text</td>
<td>Plain</td>
<td>Unformatted text</td>
</tr>
<tr>
<td></td>
<td>HTML</td>
<td>Text including HTML markup commands</td>
</tr>
<tr>
<td></td>
<td>XML</td>
<td>Text including XML markup commands</td>
</tr>
<tr>
<td>Image</td>
<td>GIF</td>
<td>Still image in GIF format</td>
</tr>
<tr>
<td></td>
<td>JPEG</td>
<td>Still image in JPEG format</td>
</tr>
<tr>
<td>Audio</td>
<td>Basic</td>
<td>Audio, 8-bit PCM sampled at 8000 Hz</td>
</tr>
<tr>
<td></td>
<td>Tone</td>
<td>A specific audible tone</td>
</tr>
<tr>
<td>Video</td>
<td>MPEG</td>
<td>Movie in MPEG format</td>
</tr>
<tr>
<td></td>
<td>Pointer</td>
<td>Representation of a pointer device for presentations</td>
</tr>
<tr>
<td>Application</td>
<td>Octet-stream</td>
<td>An uninterpreted byte sequence</td>
</tr>
<tr>
<td></td>
<td>Postscript</td>
<td>A printable document in Postscript</td>
</tr>
<tr>
<td></td>
<td>PDF</td>
<td>A printable document in PDF</td>
</tr>
<tr>
<td>Multipart</td>
<td>Mixed</td>
<td>Independent parts in the specified order</td>
</tr>
<tr>
<td></td>
<td>Parallel</td>
<td>Parts must be viewed simultaneously</td>
</tr>
</tbody>
</table>
Interactive slide

Mime:

How can the various document types be interpreted correctly by a browser?

What document types can be found on the Web?
Mime: Multi-purpose Internet Mail Exchange

How can the various document types be interpreted correctly by a browser?

Through code migration (i.e. plugins loaded from a central code repository)

Types of documents on the Web?

Any type of digital information.
Multitiered Architectures

Modern Web services support dynamically created content. The principle of using server-side CGI programs:

1. Get request
2. HTTP request handler
3. Start process to fetch document
4. Database interaction
5. HTML document created
6. Return result

Web server
CGI process
Database server
Interactive slide

Meaning of acronym CGI?

What is CGI used for, and why is it useful?

Can a client request the CGI source code?
Meaning of acronym CGI: Common Gateway Interface

What is CGI used for, and why is it useful?
Serves as a transparent interface between client and data.
• Create personalized content dynamically from a template.
• Provide access control mechanisms.
• Convert document format.
• …in a transparent fashion.

Can a client request the CGI source code?
No
Processes – Clients (1)

The logical components of a client side processes (i.e. a Web browser):

- User interface
- Browser engine
- Rendering engine
  - Network comm.
  - Client-side script interpreter
  - HTML/XML parser
  - Display back end
Processes – Clients (2)

Originally, Web browsers were designed to support HTTP only. A Web proxy process was introduced to allow the browser to access FTP content:

Nowadays, Web browsers support a large range of document formats due to the availability of code migration methods. But proxies remain popular due to additional functions that they can perform: authentication, caching.
Interactive slide

Purpose of proxy?

- 

What else can the proxy be used for?

- 

Explain how a proxy can address client-side security?

-
Purpose of proxy? Allows to achieve

- transparency.
- Translation of global protocol to a local protocol.

What else can the proxy be used for?

- Speed improvement through caching amongst several users.

Explain how a proxy can address client-side security?

- Can introduce an additional layer of security (authentication, firewall)
Server processes: The Apache Web Server

The general organization of the Apache Web server.

[Diagram of Apache Web server architecture]
Interactive slide

Why apache?

Purpose of hocks?

What is achieved through the use of hocks?

Examples of what one hock may perform?
Why apache? Scalable, extremely versatile Web server architecture.

Purpose of hocks? Each hock is responsible for executing a specific group of related functions that may need to be executed as a result of a request.

What is achieved through the use of hocks? Scalability through parallelism and pipelining of tasks.

Examples of what one hock may perform?
Resolve URL from a local file, write to a local log file, perform authentication, check access rights, read from a file, execute a CGI, etc.
Web Server Clusters (1)

The principle of using a server cluster in combination with a front end to implement a Web service.
Web Server Clusters (2)

In a *content-aware* cluster of Web servers, the switch “knows” what a server has done before. Requests are relayed so as to maximise cache hits (to servers that have processed the same request before).
Interactive slide

Purpose of switch?

Purpose of dispatcher?

Purpose of distributor?

Scalability of the approach?
Purpose of switch? Relay requests. Can aim at selecting a server which, for instance, handled a similar request in the past (to maximize chance of accessing cached data), or which is located closest to client.

Purpose of dispatcher? Monitors availability of Web server, and selects an available one.

Purpose of distributer? First point of contact for the switch, reports to dispatcher.

Scalability of the approach? Performs automated load balancing on arbitrary number of nodes. Requests to servers with a high load are redirected to available servers.
Communication (1)

Example HTTP: When using non-persistent connections. A separate TCP connection is established to load every component of a Web document (i.e. load embedded images):

(a)
Interactive slide

What is a non-persistent connection, and where can this be found?

When are non-persistent connections becoming inefficient?

How could this be overcome?
What is a non-persistent connection, and where can this be found? Connection is closed by server after one request has been processed. HTTP version 1.

When are non-persistent connections becoming inefficient? When Web documents contain embedded content such as images, multi-media content, etc.

How could this be overcome? Establish sever connections simultaneously to achieve parallelism with requests. But this will impact the load on the server, and may not be effective in slow networks.
Communication (2)

Example (b): Using persistent connections in HTTP.
Interactive slide

What is a persistent connection, and where can this be found?

What is realized through persistent connections?

When are persistent connections becoming inefficient?
What is a persistent connection, and where can this be found? Connection stays open. Multiple requests can be made through a single connection. Supported by HTTP version 1.1 and greater.

What is realized through persistent connections? Pipelining of requests through the same connection.

When are persistent connections becoming inefficient? When servers are not designed to handle compound requests efficiently. Client make fast successive requests cause increased server load. Increased vulnerability to denial of service (DOS) attacks.
## HTTP Methods

Request operations (primitives) supported by HTTP:

<table>
<thead>
<tr>
<th>Operation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Head</td>
<td>Request to return the header of a document</td>
</tr>
<tr>
<td>Get</td>
<td>Request to return a document to the client</td>
</tr>
<tr>
<td>Put</td>
<td>Request to store a document</td>
</tr>
<tr>
<td>Post</td>
<td>Provide data that are to be added to a document (collection)</td>
</tr>
<tr>
<td>Delete</td>
<td>Request to delete a document</td>
</tr>
</tbody>
</table>
Interactive slide

What information can be expected to be received in response to operation Head?

What is happening when trying to issue a Delete, Push, or Post request on a protected item?
Interactive slide

What information can be expected to be received in response to operation Head?
An error code (if document does not exist, or if insufficient access rights), date of last modification (to identify whether client’s cached version is up to date).

What is happening when trying to issue a Delete, Push, or Post request on a protected item? A response containing an error code such as code 403 (Forbidden)
HTTP Messages (1)

With HTTP all request methods result in messages of the same form:

![Diagram showing HTTP message structure]

(a)
Interactive slide

Request Line optional: yes/no?
Message Headers optional: yes/no?
Message Body optional: yes/no?

What content can be found in the following message fields:

Operation:
Reference:
Version:
Interactive slide

Request Line optional: **no**
Message Headers optional: **yes**
Message Body optional: **yes**

What content can be found in the following message fields:

**Operation:** Operation to carry out (i.e. GET, PUT, etc)
**Reference:** Document to be affected by the operation.
**Version:** HTTP version the client is expecting.
HTTP Messages (2)

With HTTP all response messages are expected to be in the form as shown here:

```
+--------+--------+--------+
| Version| Status code | Phrase |
+--------+--------+--------+
| Message header name | Value |
| Message header name | Value |
| ... |
+--------+--------+--------+
```

(b)
What content can be found in the following message fields:

Version:

Status code and phrase:

Message body:
What content can be found in the following message fields:

**Version:** HTTP version of the message.

**Status code and phrase:** a 3-digit code and a brief explanation of the meaning (i.e. 404 (Not Found), or 405 (Method not allowed)).

**Message body:** Is an optional field which can contain data of a file (i.e. the content of a http file if a http file was requested).
HTTP Messages (3)

Some valid values for HTTP message header names.

<table>
<thead>
<tr>
<th>Header</th>
<th>Source</th>
<th>Contents</th>
</tr>
</thead>
<tbody>
<tr>
<td>Accept</td>
<td>Client</td>
<td>The type of documents the client can handle</td>
</tr>
<tr>
<td>Accept-Charset</td>
<td>Client</td>
<td>The character sets are acceptable for the client</td>
</tr>
<tr>
<td>Accept-Encoding</td>
<td>Client</td>
<td>The document encodings the client can handle</td>
</tr>
<tr>
<td>Accept-Language</td>
<td>Client</td>
<td>The natural language the client can handle</td>
</tr>
<tr>
<td>Authorization</td>
<td>Client</td>
<td>A list of the client’s credentials</td>
</tr>
<tr>
<td>WWW-Authenticate</td>
<td>Server</td>
<td>Security challenge the client should respond to</td>
</tr>
<tr>
<td>Date</td>
<td>Both</td>
<td>Date and time the message was sent</td>
</tr>
<tr>
<td>ETag</td>
<td>Server</td>
<td>The tags associated with the returned document</td>
</tr>
<tr>
<td>Expires</td>
<td>Server</td>
<td>The time for how long the response remains valid</td>
</tr>
<tr>
<td>From</td>
<td>Client</td>
<td>The client’s e-mail address</td>
</tr>
<tr>
<td>Host</td>
<td>Client</td>
<td>The DNS name of the document’s server</td>
</tr>
</tbody>
</table>
Interactive slide

Explain Accept-Charset and give one possible value:

Explain Accept-Encoding and give one possible value:

Explain Expires:
Explain **Accept-Charset** and give one possible value: This accounts for (human) languages which may differ in the way they are encoded on a computer, a charset identifier is a unique reference to a specific encoding of a language. Example: ISO8859 refers to a standard western charset, or GB2312 refers to the simplified Chinese charset.

Explain **Accept-Encoding** and give one possible value: The type of encoding of a document (i.e. plain text, gzip, jpeg, encryption methods, etc.)

Explain **Expires**: Explicitly tells a client for how long the content of a response remains valid.
### HTTP Messages (4)

Some more HTTP message headers:

<table>
<thead>
<tr>
<th>Header</th>
<th>Source</th>
<th>Contents</th>
</tr>
</thead>
<tbody>
<tr>
<td>If-Match</td>
<td>Client</td>
<td>The tags the document should have</td>
</tr>
<tr>
<td>If-None-Match</td>
<td>Client</td>
<td>The tags the document should not have</td>
</tr>
<tr>
<td>If-Modified-Since</td>
<td>Client</td>
<td>Tells the server to return a document only if it has been modified since the specified time</td>
</tr>
<tr>
<td>If-Unmodified-Since</td>
<td>Client</td>
<td>Tells the server to return a document only if it has not been modified since the specified time</td>
</tr>
<tr>
<td>Last-Modified</td>
<td>Server</td>
<td>The time the returned document was last modified</td>
</tr>
<tr>
<td>Location</td>
<td>Server</td>
<td>A document reference to which the client should redirect its request</td>
</tr>
<tr>
<td>Referer</td>
<td>Client</td>
<td>Refers to client’s most recently requested document</td>
</tr>
<tr>
<td>Upgrade</td>
<td>Both</td>
<td>The application protocol the sender wants to switch to</td>
</tr>
<tr>
<td>Warning</td>
<td>Both</td>
<td>Information about the status of the data in the message</td>
</tr>
</tbody>
</table>
Explain difference between If-Modified-Since and Last-Modified:

Explain where the HTTP message “Upgrade” is useful:
Explain difference between If-Modified-Since and Last-Modified: A client requests a document and expects the document to be returned only if the document has been modified since a given date and time.

Explain where the HTTP message “Upgrade” is useful: To negotiate the use of the latest version of HTTP supported by both sender and receiver. Initial request may use a low version number to ensure that a server does not respond with an error message.
Naming (1)

The common structure for URLs can allow to access same resource using different “names”. Example: (a) Using only a DNS name. (b) Combining a DNS name with a port number. (c) Combining an IP address with a port number.

<table>
<thead>
<tr>
<th>Scheme</th>
<th>Host name</th>
<th>Pathname</th>
</tr>
</thead>
<tbody>
<tr>
<td>http ://</td>
<td><a href="http://www.cs.vu.nl">www.cs.vu.nl</a></td>
<td>/home/steen/mbox</td>
</tr>
<tr>
<td></td>
<td></td>
<td>(a)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Scheme</th>
<th>Host name</th>
<th>Port</th>
<th>Pathname</th>
</tr>
</thead>
<tbody>
<tr>
<td>http ://</td>
<td><a href="http://www.cs.vu.nl">www.cs.vu.nl</a></td>
<td>80</td>
<td>/home/steen/mbox</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>(b)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Scheme</th>
<th>Host name</th>
<th>Port</th>
<th>Pathname</th>
</tr>
</thead>
<tbody>
<tr>
<td>http ://</td>
<td>130.37.24.11</td>
<td>80</td>
<td>/home/steen/mbox</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>(c)</td>
</tr>
</tbody>
</table>
Port and Pathname in the naming scheme of HTTP are optional. What are their default values?

Which of the following hostnames are valid?
122.65.132.11.1.97
My_server
300.320.1.32
Port and Pathname in the naming scheme of HTTP are optional. What are their default values? 80, and {'/’, “index.htm”, “default.htm”, “index.html”, any path}. Can be configured freely on the server.

Which of the following hostnames are valid?
122.65.132.11.1.97
My_server
300.320.1.32
All three can be valid depending on the scheme used:
Naming (2)

Schemes are defined by means of URIs. Examples of URIs:

<table>
<thead>
<tr>
<th>Name</th>
<th>Used for</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>http</td>
<td>HTTP</td>
<td><a href="http://www.cs.vu.nl:80/globe">http://www.cs.vu.nl:80/globe</a></td>
</tr>
<tr>
<td>mailto</td>
<td>E-mail</td>
<td><a href="mailto:steen@cs.vu.nl">mailto:steen@cs.vu.nl</a></td>
</tr>
<tr>
<td>file</td>
<td>Local file</td>
<td>file:/edu/book/work/chp/11/11</td>
</tr>
<tr>
<td>data</td>
<td>Inline data</td>
<td>data:text/plain;charset=iso-8859-7,%e1%e2%e3</td>
</tr>
<tr>
<td>telnet</td>
<td>Remote login</td>
<td>telnet://flits.cs.vu.nl</td>
</tr>
<tr>
<td>tel</td>
<td>Telephone</td>
<td>tel:+31201234567</td>
</tr>
<tr>
<td>modem</td>
<td>Modem</td>
<td>modem:+31201234567;type=v32</td>
</tr>
</tbody>
</table>
Interactive slide

Definition of URL:
Definition of URI:

How does a web-browser interpret the various URIs?
Interactive slide

Definition of URL: Uniform Resource Locator
Definition of URI: Uniform Resource Identifiers

How does a web-browser interpret the various URIs?

– Inbuild functions.
– Plugins (Code migration).
– Calls to system library functions.
– Calls to external applications.
– User feedback.
– Printing of a (error) message.
Interactive slide

Do you agree: Client side caching is not required if proxy caches are used.

Do you agree: Server side caching is not required if proxy caches are used.

How to reduce the risk that proxy caches contain out of date information?
Interactive slide

Do you agree: Client side caching is not required if proxy caches are used. Answer depends on reasoning.

Do you agree: Server side caching is not required if proxy caches are used. Answer depends on reasoning.

How to reduce the risk that proxy caches contain out of date information? Proxy to take a pro-active role by querying server for updates, or discard information which may be out of date, or request from server to inform proxy of relevant updates.
Replication for Web Hosting Systems

The general organization of a CDN as a feedback-control system:

Uncontrollable parameters (disturbance / noise)

Initial configuration → Corrections → Web hosting system → Observed output

+/-  +/-  +/-

Replica placement → Consistency enforcement → Request routing

Reference input → Analysis → Measured output
Definition of CDN:

Name advantages of a feedback control system in a CDN:
Definition of CDN: Content delivery networks

Name advantages of a feedback control system in a CDN: Allows adjustments to daily changes in system load, can prevent DOS on Web server, mask failure of a Web server by relaying requests to non-faulty nodes, strategic distribution of Web content, automation of information distribution, and more.
Adaptation Triggering

One normal and three different access patterns reflecting flash-crowd behavior.
Interactive slide

Explain flash crowd:

Explain different types of a flash crowd, and how these can be predicted.
Explain flash crowd: Is a sudden burst in requests. It can be the result of a DOS attack.

Explain different types of a flash crowd, and how these can be predicted.

1.) Increased requests due to predictable events (i.e requests that occur at “busy” times of a day, are the result of information seeking behavior of the users of the system). Predict through analysis of long term access patterns.

2.) Increased requests due to unpredictable events (large number of dummy requests which exist for no reason other than to do harm). Predict through analysis of short term access patterns.
Adjustment Measures

The principal working of the Akamai CDN.

1. Get base document
2. Document with refs to embedded documents
3. Regular DNS system
4. DNS lookups
5. Get embedded documents
6. Get embedded documents (if not already cached)
7. Embedded documents

Return IP address client-best server

Cache

CDN server

Origin server
Replication of Web Applications

Alternatives for caching and replication with Web applications.
Interactive slide

Explain the three different caches of the CDN:

1. Database copy:
2. Content aware cache:
3. Content-blind cache:

What strategies would you use if all CDN servers are on high load?
Security (1)

The open nature of the Internet requires a security architecture that protects clients and servers. The predominant approach is through Secure Socket Layer (SSL) realized as Transport Layer Security (TLS) below the application layer in the Internet protocol stack as illustrated here:

<table>
<thead>
<tr>
<th>HTTP</th>
<th>FTP</th>
<th>Telnet</th>
<th>TLS</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>TLS</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Transport layer</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Network layer</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Data link layer</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Physical layer</td>
</tr>
</tbody>
</table>
Security (2)

TLS with mutual authentication.

---

1. Possibilities
2. Choices
3. $[K^+_S]_{CA}$
4. $[K^+_C]_{CA}$
5. $K^+_S([R]_C)$
Summary

Web based system design:

- Architectures: Multi-tiered, clusters
- Processes: Client and server processes
- Naming: Structured naming in HTTP
- Fault tolerance: Client side caching, server replication
- Synchronization: i.e. Distributed Authoring use locking (WebDAV, not covered)
- Consistency and Replication: Caching, feedback control
- Security: Mainly Encryption and authentication