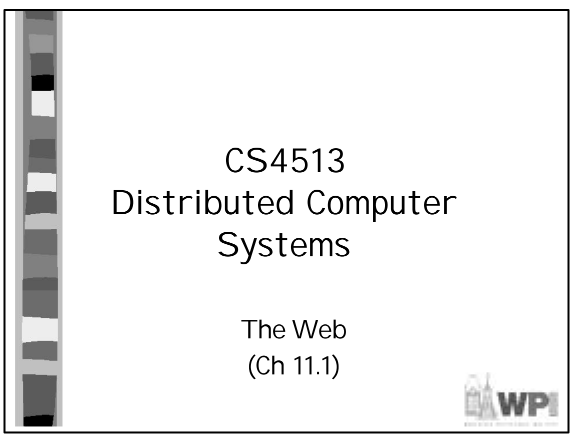
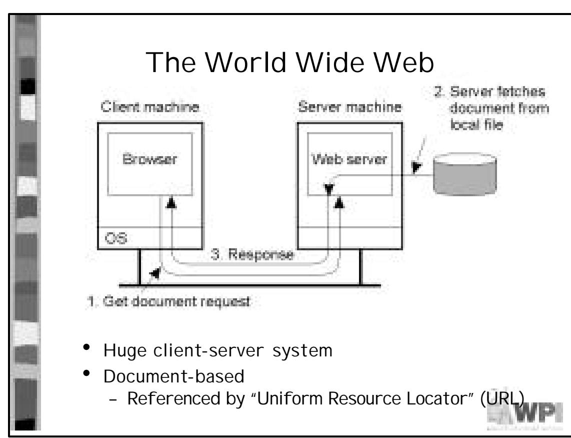
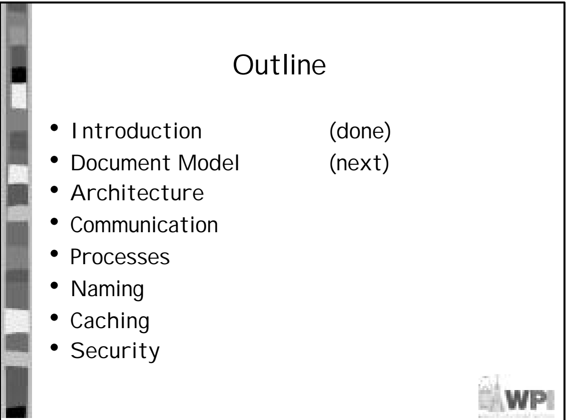
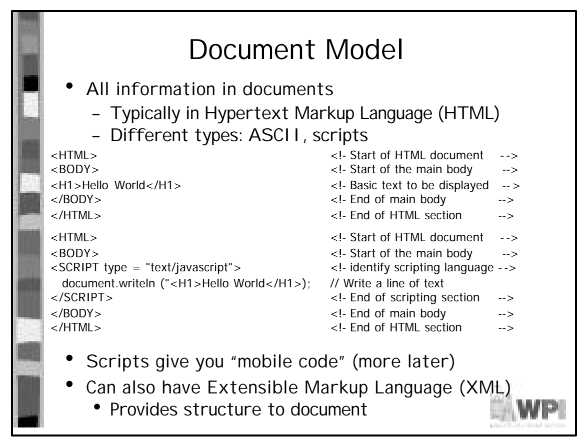








XML DTD

```
(1)   <!ELEMENT article (title, author+,journal)>
(2)   <!ELEMENT title (#PCDATA)>
(3)   <!ELEMENT author (name, affiliation?)>
(4)   <!ELEMENT name (#PCDATA)>
(5)   <!ELEMENT affiliation (#PCDATA)>
(6)   <!ELEMENT journal (jname, volume, number?, month? pages, year)>
(7)   <!ELEMENT jname (#PCDATA)>
(8)   <!ELEMENT volume (#PCDATA)>
(9)   <!ELEMENT number (#PCDATA)>
(10)  <!ELEMENT month (#PCDATA)>
(11)  <!ELEMENT pages (#PCDATA)>
(12)  <!ELEMENT year (#PCDATA)>
```

(#PCDATA is primitive type, series of chars)

- Definition above refers to a journal article. Specifies type.
  - In a Document Type Definition (DTD)
  - Provides structure to XML documents

WPI

XML Document

```
(1)   <?xml = version "1.0">
(2)   <!DOCTYPE article SYSTEM "article.dtd">
(3)   <article>
(4)     <title>Prudent Engineering Practice for Cryptographic Protocols</title>
(5)     <author><name>M. Abadi</name></author>
(6)     <author><name>R. Needham</name></author>
(7)     <journal>
(8)       <jname>IEEE Transactions on Software Engineering</jname>
(9)       <volume>22</volume>
(10)      <number>12</number>
(11)      <month>January</month>
(12)      <pages>6 – 15</pages>
(13)      <year>1996</year>
(14)    </journal>
(15)  </article>
```

- An XML document using the XML definitions from previous slide
- Formatting rules usually applied by embedding in HTML

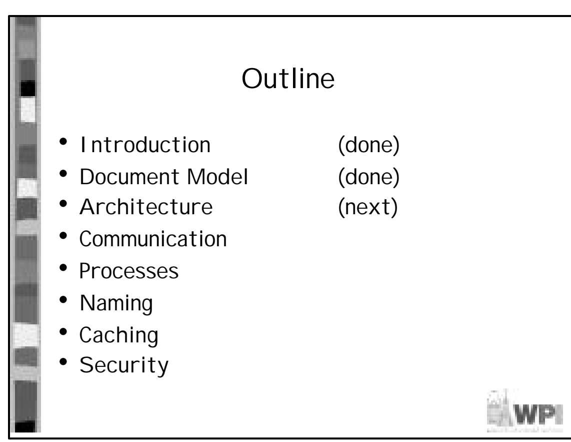
WPI

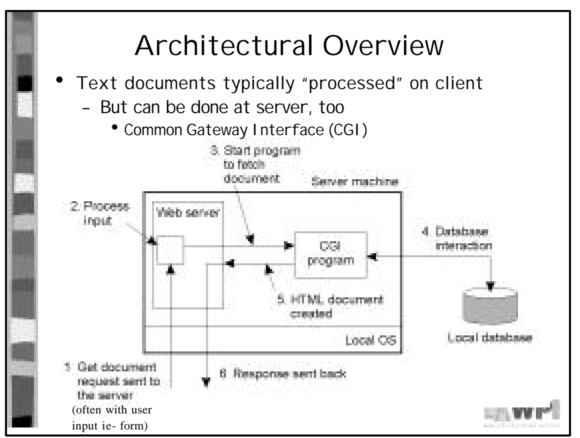
Document Types

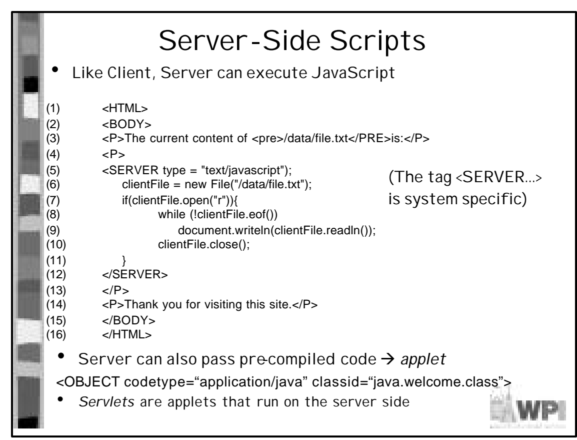
- Beyond text → can include other types
  - Multipurpose Internet Mail Extensions (MIME)

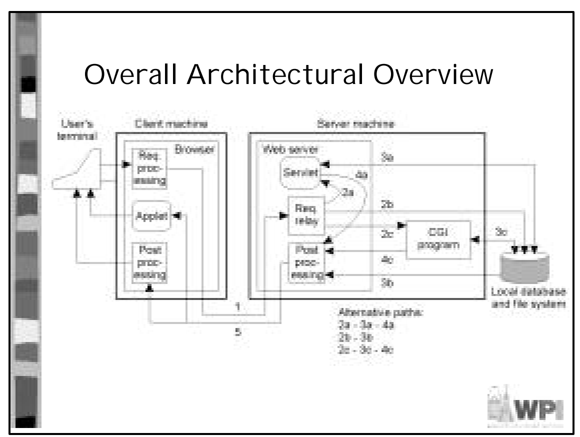
Type	Subtype	Description
Text	Plain	Unformatted text
	HTML	Text including HTML markup commands
	XML	Text including XML markup commands
Image	GIF	Still image in GIF format
	JPEG	Still image in JPEG format
Audio	Basic	Audio, 8-bit PCM sampled at 8000 Hz
	Tone	A specific audible tone
Video	MPEG	Movie in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

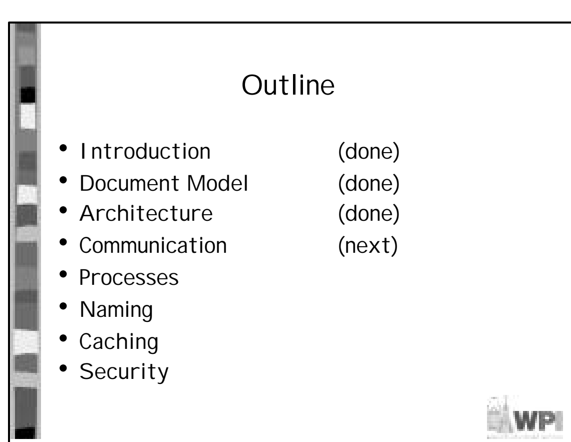
- Includes types and sub-types
- *Application* specifies application-specific data type

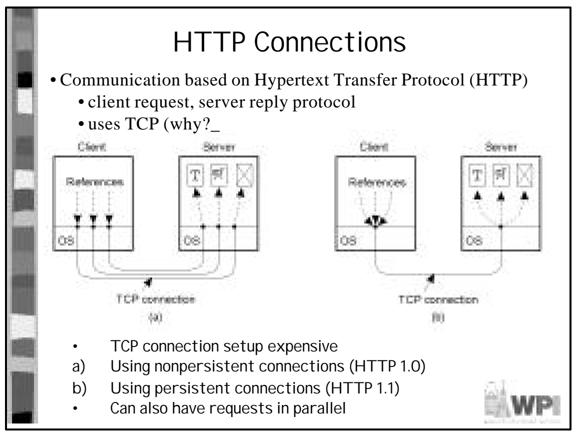


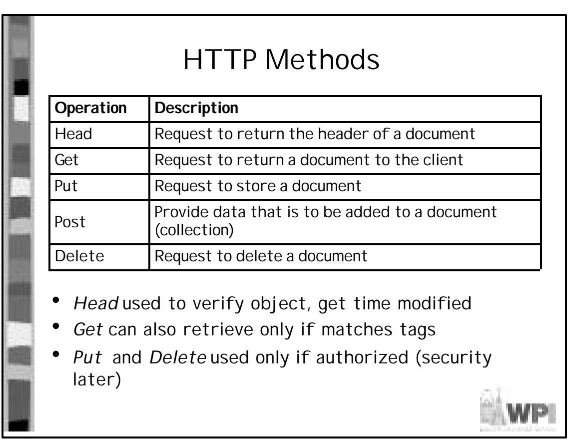




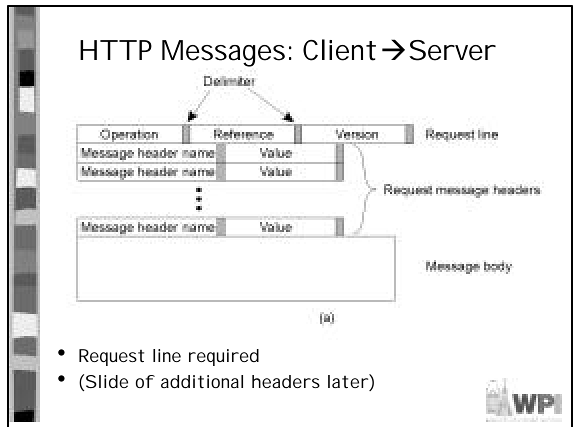


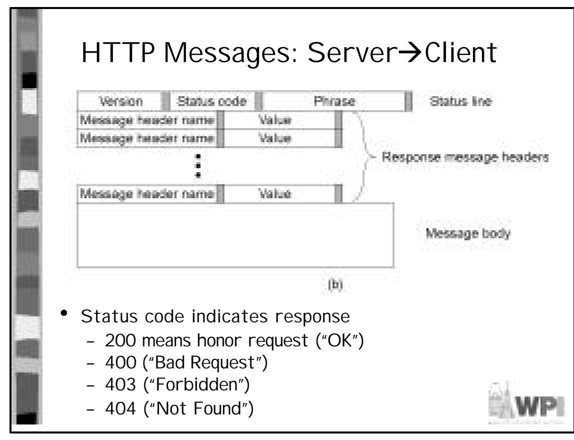






Operation	Description
Head	Request to return the header of a document
Get	Request to return a document to the client
Put	Request to store a document
Post	Provide data that is to be added to a document (collection)
Delete	Request to delete a document

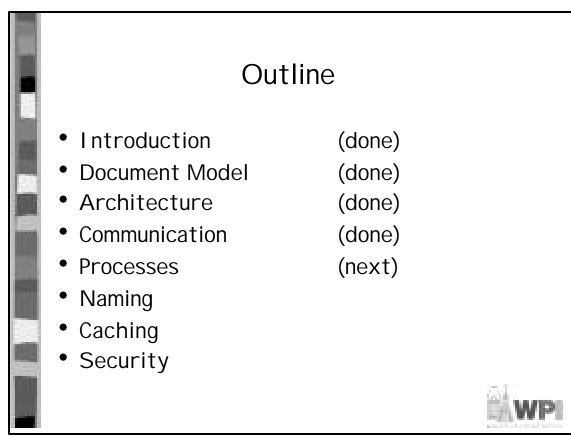


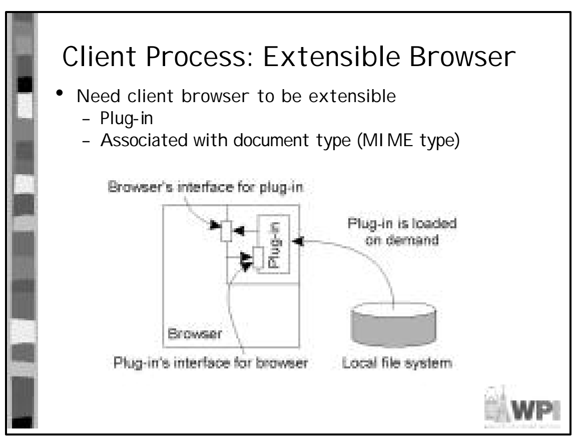


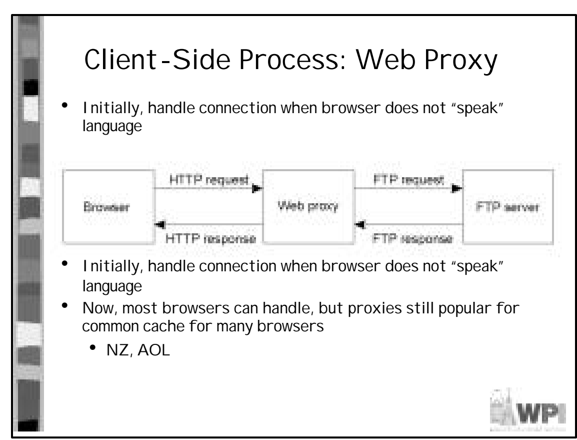
HTTP Additional Headers

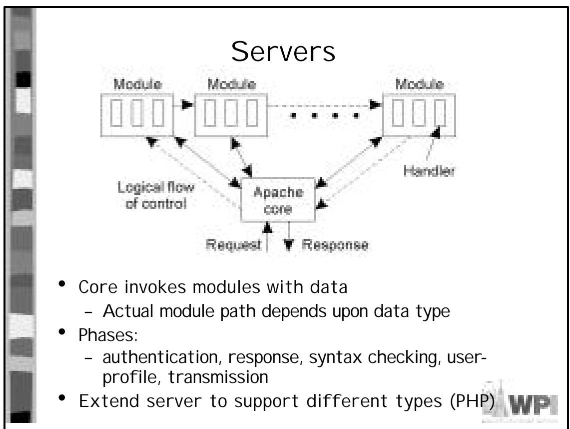
- Augment Client request or Server Response
- *Accept* encoding of gzip
- *Upgrade* to Secure HTTP
- *Redirect* for load balance

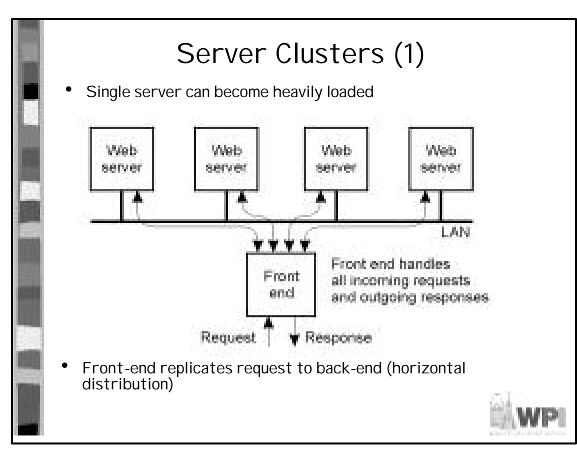
Header	Source	Contents
Accept	Client	The type of documents the client can handle
Accept-Charset	Client	Character sets are acceptable for the client
Accept-Encoding	Client	Document encodings the client can handle
Accept-Language	Client	The natural language the client can handle
Authorization	Client	A list of the client's credentials
WWW-Auth	Server	Security challenge to the client
Date	Both	Date and time the message was sent
ETag	Server	Tags associated with the returned document
Expires	Server	The time how long the response remains valid
From	Client	The client's e-mail address
Host	Client	The TCP address of the document's server
If-Match	Client	The tags the document should have
If-None-Match	Client	The tags the document should not have
If-Modified- Since	Client	Only return a document if newly modified
If-Unmodified-Since	Client	Return a document only if it has not been modified since the specified time
Last-Modified	Server	Time the returned document was last modified
Location	Server	Reference to which the client should redirect
Referer	Client	Client's most recently requested document
Upgrade	Both	App protocol the sender wants to switch to
Warning	Both	Information about the status of the data

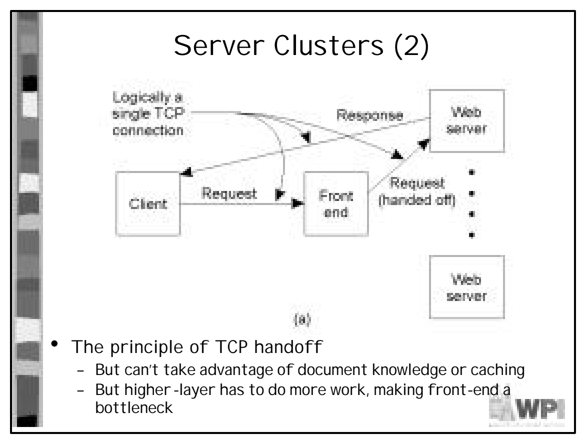


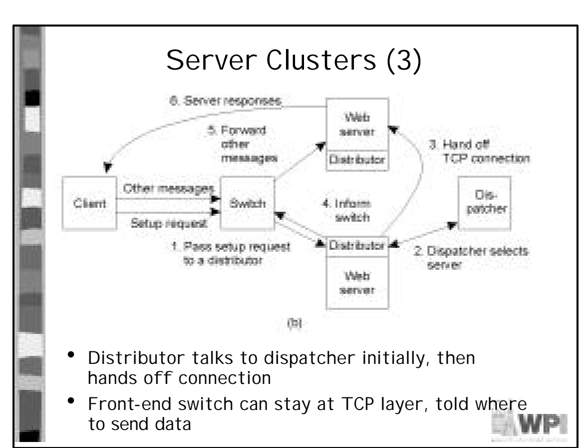

Client Process: Extensible Browser
• Need client browser to be extensible
– Plug-in
– Associated with document type (MIME type)
Browser's interface for plug-in
Plug-in
Plug-in is loaded on demand
Browser
Plug-in's interface for browser
Local file system

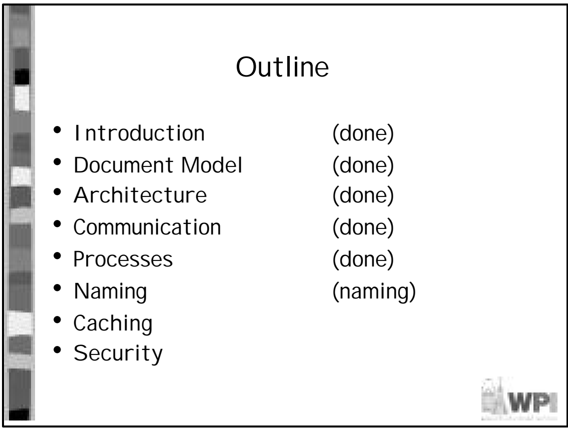

Client-Side Process: Web Proxy
• Initially, handle connection when browser does not "speak" language
Browser
HTTP request
HTTP response
Web proxy
FTP request
FTP response
FTP server
• Initially, handle connection when browser does not "speak" language
• Now, most browsers can handle, but proxies still popular for common cache for many browsers
• NZ, AOL

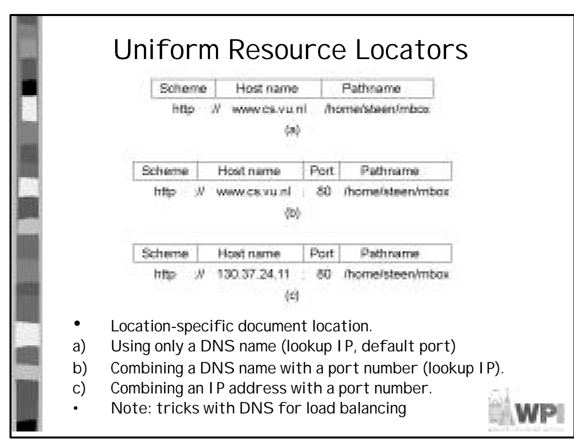

Servers
Module
Module
Module
Logical flow of control
Apache core
Handler
Request
Response
• Core invokes modules with data
– Actual module path depends upon data type
• Phases:
– authentication, response, syntax checking, user-profile, transmission
• Extend server to support different types (PHP)


Server Clusters (1)
• Single server can become heavily loaded
Web server
Web server
Web server
Web server
LAN
Front end
Front end handles all incoming requests and outgoing responses
Request
Response
• Front-end replicates request to back-end (horizontal distribution)


Server Clusters (2)
Logically a single TCP connection
Response
Web server
Client
Request
Front end
Request (handed off)
Web server
(a)
• The principle of TCP handoff
– But can't take advantage of document knowledge or caching
– But higher-layer has to do more work, making front-end a bottleneck

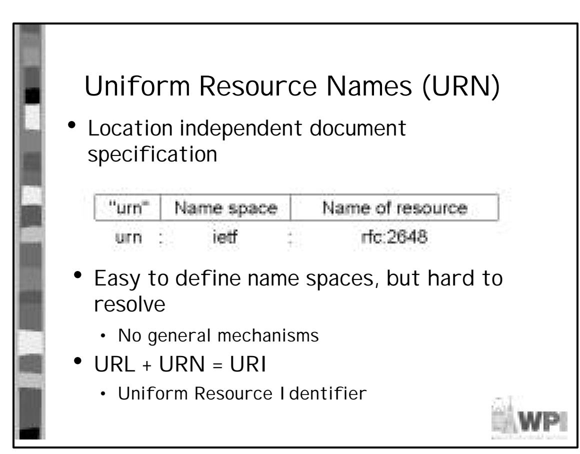

Server Clusters (3)
6. Server responses
5. Forward other messages
Web server
Distributor
3. Hand off TCP connection
Client
Other messages
Setup request
Switch
4. Inform switch
Dis-patcher
1. Pass setup request to a distributor
Distributor
2. Dispatcher selects server
Web server
(b)
• Distributor talks to dispatcher initially, then hands off connection
• Front-end switch can stay at TCP layer, told where to send data

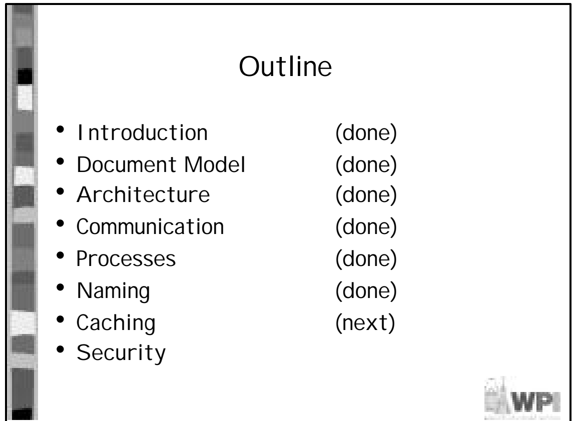


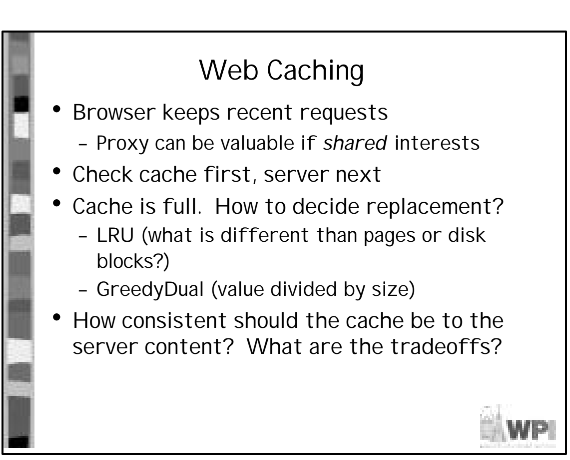


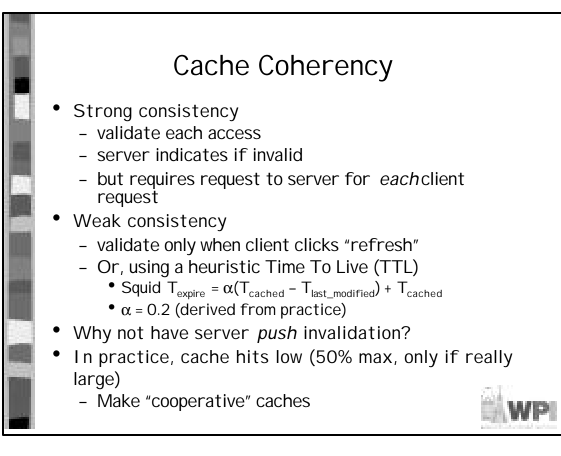
URL Examples

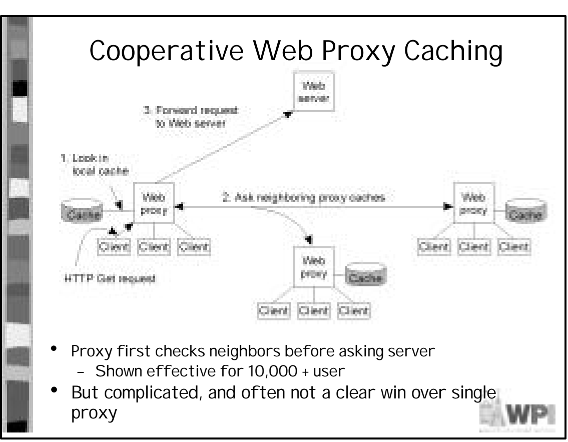
Scheme Name	Used for	Example
http	HTTP	http://www.cs.vu.nl:80/globe
ftp	FTP	ftp://ftp.cs.vu.nl/pup/minx/README
file	Local file	file:/edu/book/work/chp/11/11
data	Inline data	data:text/plain;charset=iso-8859-7,%e1%e2%e3
telnet	Remote login	telnet://flits.cs.vu.nl
tel	Telephone	tel:+31201234567
modem	Modem	modem:+31201234567;type=v32







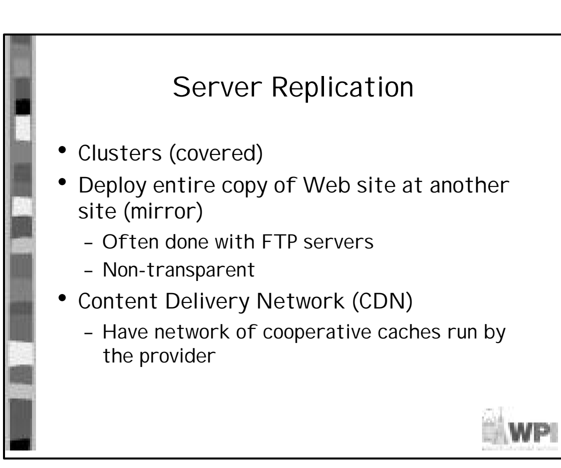


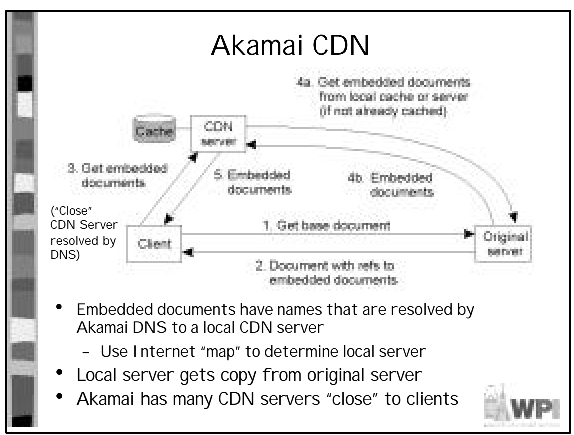


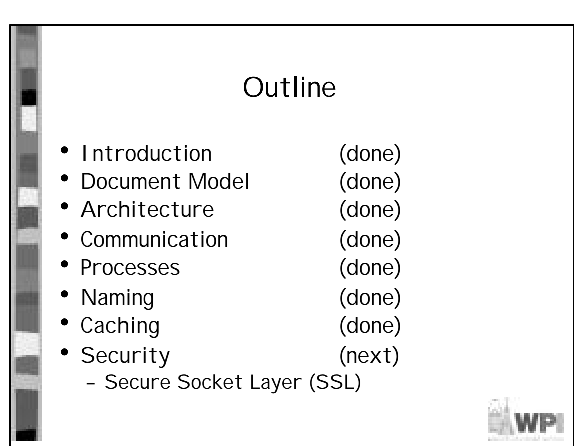
Misc Caching

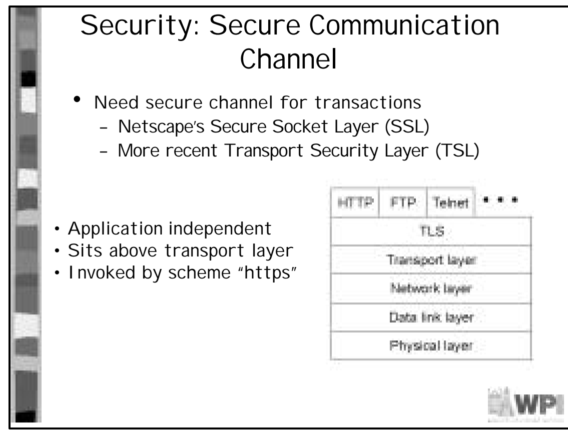
- Static vs. Dynamic Documents
  - Caching only effective for static documents (non CGI)
    - But Web increasingly dynamic (personalized)
    - Cookies used since server (mostly) stateless
  - Make proxies support active caching
    - Generate the HTML
    - Need copies of server-side scripts/code
    - Accessing databases harder
- Caching large documents
  - Can only send changes from original
  - Often, connection request is the large cost

WPI









Establishing an SSL connection

1. Client sends SSL version number, cipher settings, randomly generated data and other information server needs.
2. Server sends server SSL version number, cipher settings, randomly generated data, servers own certificate.
   - (Optional) Server may request client's certificate. Client authenticates server certificate by using public key of certificate authority (CA)
3. Client creates *premaster key* for session and encrypts it with servers public key (obtained from server's certificate) and sends to server.
   - (Optional) Client sends encrypted data based on own private key if client needs authentication.
4. Server generates *master secret*, sends to server
5. Both client and server use master secret to generate *session keys*, which are symmetric keys for encryption/decryption of exchanged information during SSL session.
6. Client and server inform each other session key has been created.
7. SSL handshake is complete.