

# INFO/CS 4302

# Web Information Systems

FT 2012

Week 6: Recap Session  
(Lecture 10)

*Theresa Velden*

# Topics in First 3<sup>rd</sup> of the course

- Internet History & Architecture
- Web History & Architecture
- Structured Document and Data Formats
  - XML, Namespaces
  - XML Schema, XPath, RELAX NG
  - XML Manipulation: XSLT, DOM
  - JSON/YAML
- Internet Surveillance (this week)

# **INTERNET HISTORY & ARCHITECTURE**

Who invented the Internet?

Internet  $\neq$  World Wide Web

# Principles of Internet Design ?

# Principles of Internet Design

- Distributed
- Open
- End-to-end

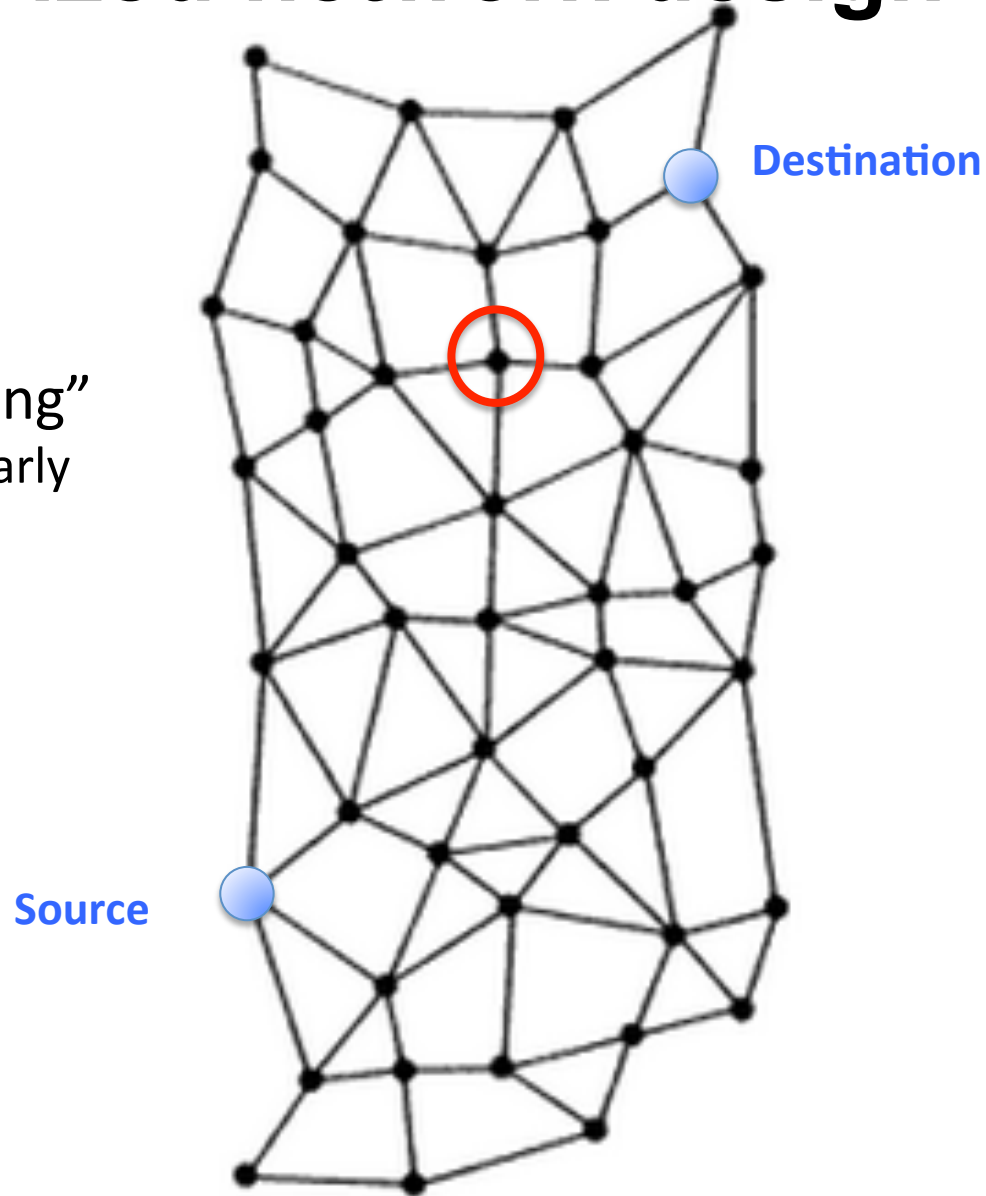
# Principles of Internet Design

- Distributed: e.g. network design & routing, packet switching
- Open: e.g. community based standards
- End-to-end: application specific functionality at end points

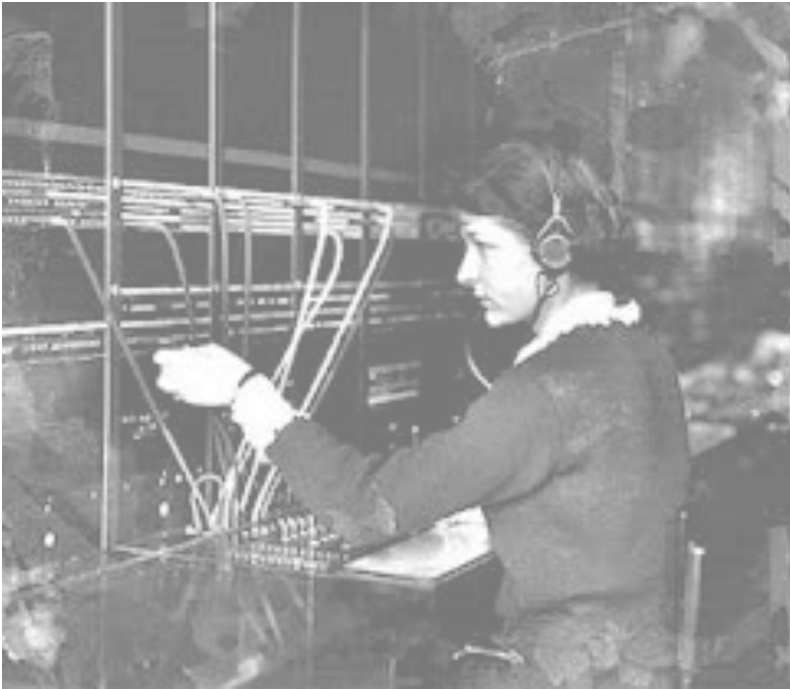


# De-centralized network design

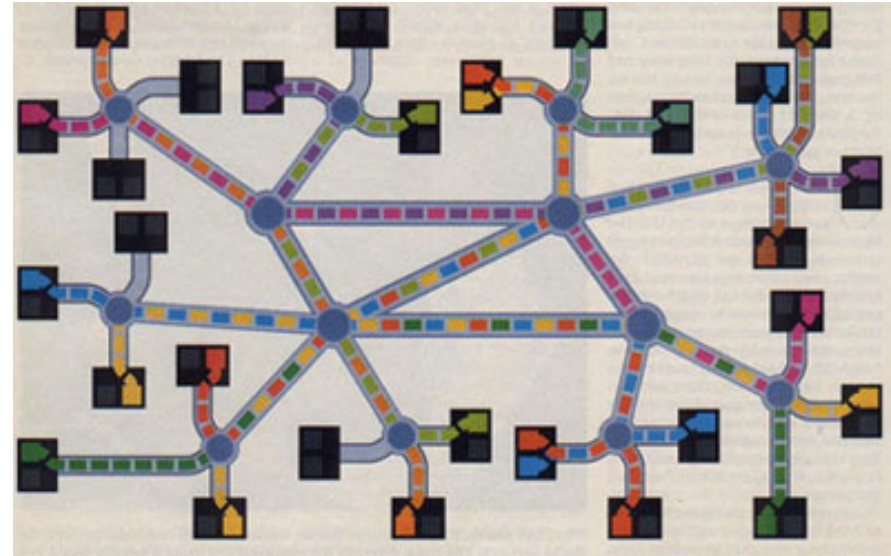
"hot-potato routing"  
(Paul Baran, RAND early  
1960s)



# circuit switching vs. packet switching

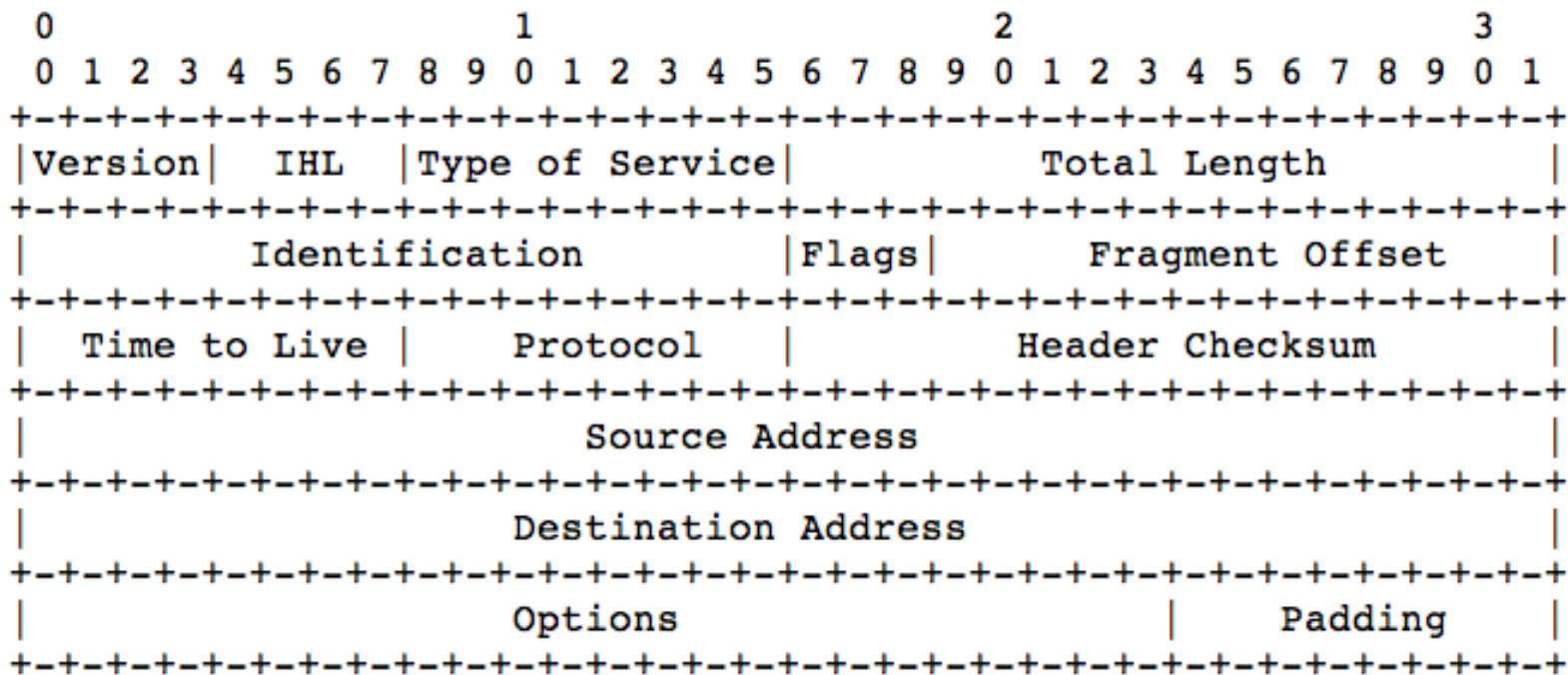


- Dedicated line for entire conversation incl. silences (-)
- Less efficient (-)
- More reliable (+)

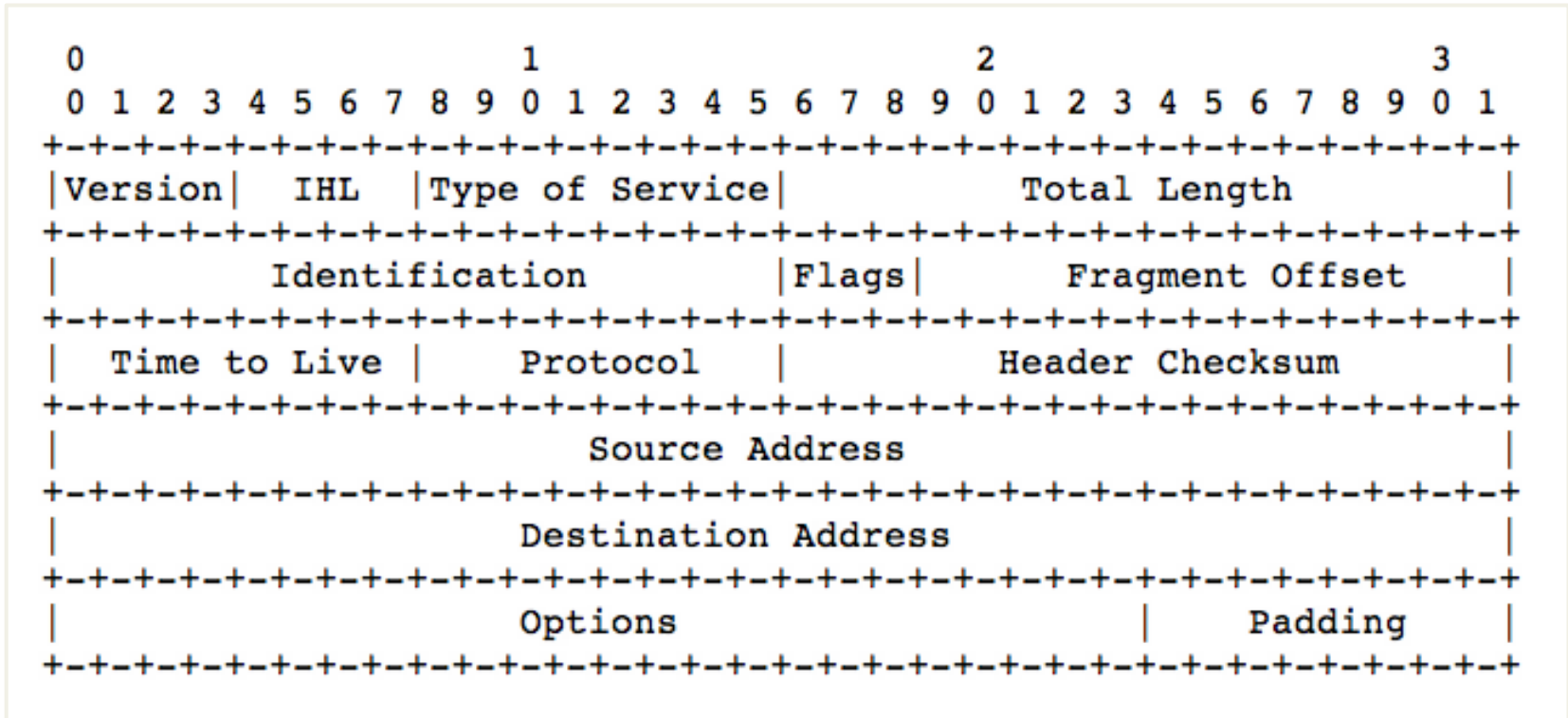


[From: <http://digitalfewsure.typepad.com/>]

- Full bandwidth for each packet (+)
- More efficient (+)
  - Users share network
  - Resource use only when needed
- Less reliable (-)



# Header of IP datagram



'best effort at delivery'

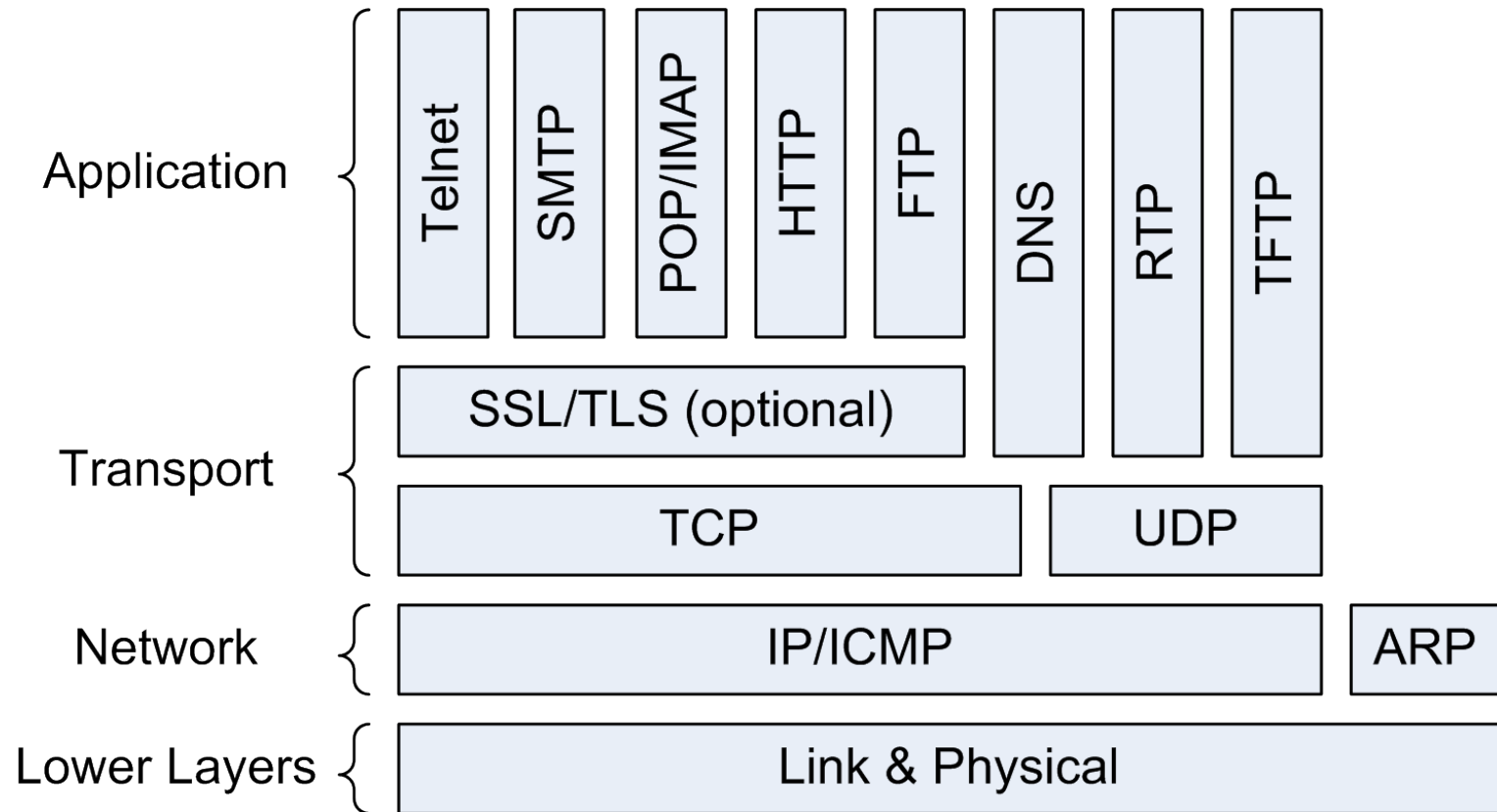
‘best effort at delivery’

= unreliable (packets may be dropped)

**Internet Protocol**

Can you name and very briefly describe at least three protocols, each from a different layer in the Internet protocol stack?

# TCP/IP Protocol Suite





What are TCP/UDP Sockets, also called  
“Virtual Ports” about?

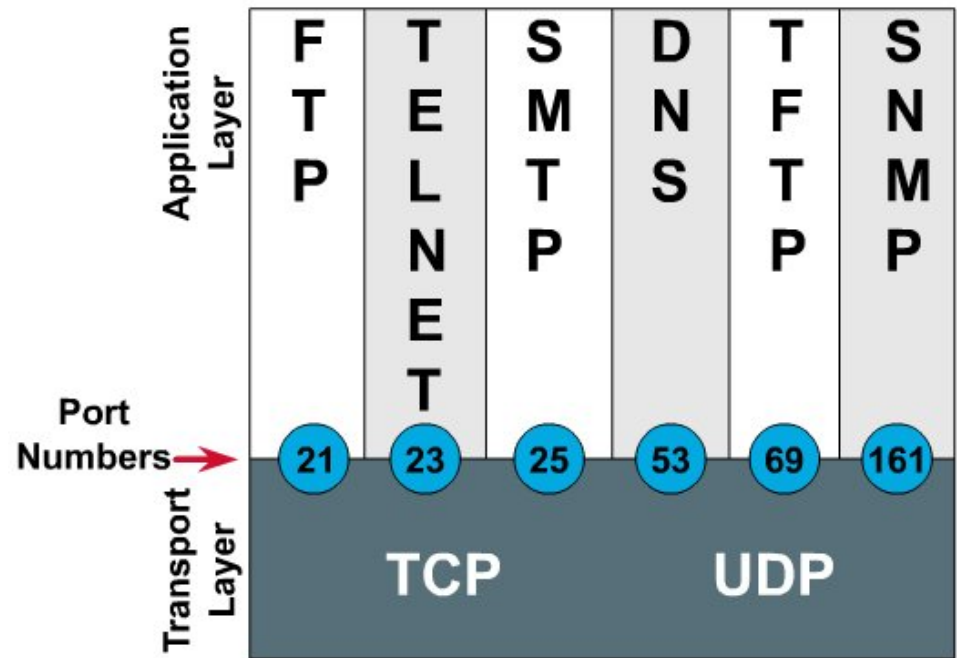
# TCP/UDP Sockets (“Virtual Ports”)

- IANA maintains a Service Name and Transport Protocol Port Number Registry:

“Service names and port numbers are used to distinguish between different services that run over transport protocols such as TCP, UDP, DCCP, and SCTP”.

- Services such as: ftp, smtp, whois...
- Needed for binding to applications (IP address/port number, using a protocol such as TCP or UDP)

## Port Numbers



Port Number Listings:

<http://www.ietf.org/assignments/service-names-port-numbers/service-names-port-numbers.txt>

[http://en.wikipedia.org/wiki/List\\_of\\_TCP\\_and\\_UDP\\_port\\_numbers](http://en.wikipedia.org/wiki/List_of_TCP_and_UDP_port_numbers)

How to get a message from point A (IP address A) to point B (IP address B) on the Internet?

# Autonomous Systems (AS)

“The classic definition of an Autonomous System is a **set of routers under a single technical administration, using an interior gateway protocol (IGP) and common metrics to determine how to route packets within the AS, and using an inter-AS routing protocol to determine how to route packets to other ASes.** Since this classic definition was developed, it has become common for a single AS to use several IGPs and, sometimes, several sets of metrics within an AS.” [source: RFC 4271 (BGP-4, January 2006) Autonomous System (AS)]

[source:

[http://www.cisco.com/web/about/ac123/ac147/archived\\_issues/ipj\\_9-1/autonomous\\_system\\_numbers.html](http://www.cisco.com/web/about/ac123/ac147/archived_issues/ipj_9-1/autonomous_system_numbers.html),

Geoff Huston, APNIC]

# 2-Layer Internet Routing Architecture

- Interdomain routing: Border Gateway Protocol (EBGP)  
RFC4271 (Jan 2006)
- Internal routing (Interior Gateway Protocol, IBGP)
  - Domain specific routing policies
  - Domains = Autonomous Systems
    - Not every domain network has a ASN (ASNs express distinct interdomain routing policies)

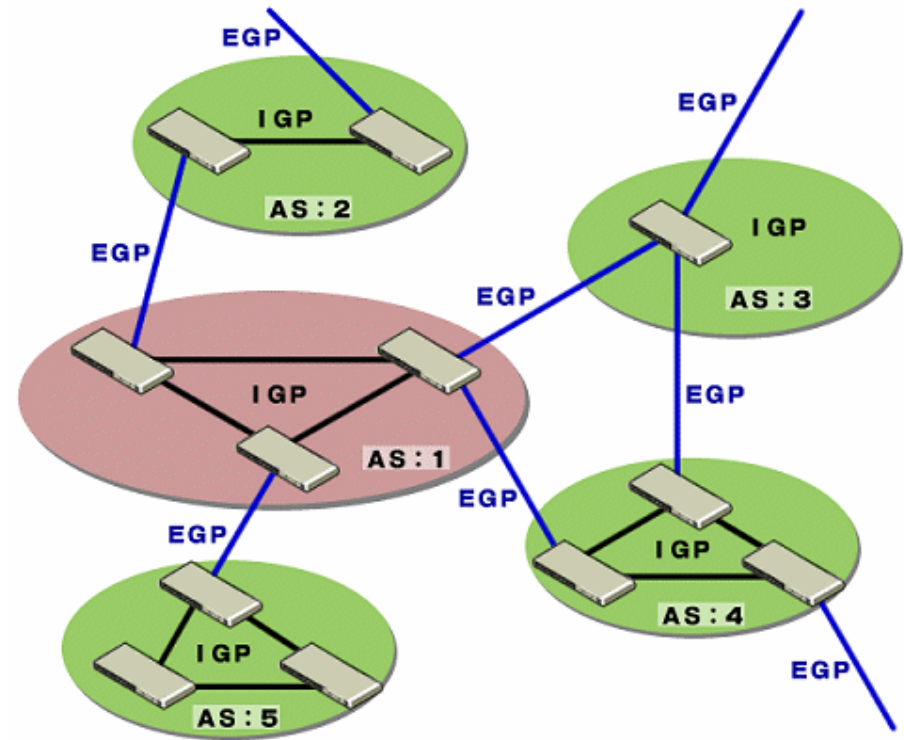


Image source: <http://www.atmarkit.co.jp/fnetwork/rensai/iprt01/iprt01.html>

# Border Gateway Protocol Routing Tables

- Essential to the decentralized nature of the Internet
- Uses Transfer Control Protocol (TCP), port 179
- Manually configured connections to peer (neighbors)
- Peers exchange routing information
- BGP Routing tables:
  - Paths (AS numbers) to destinations (only one per destination in forwarding table)
  - Broadcast 'update' to neighbors, add own AS number as prefix
  - Updates from neighbors stored in Routing Information Base (RIB)
  - Routers import no loops

# **WEB ARCHITECTURE**

What are the key architectural components of the Web & what is meant by the “principle of orthogonal specification”?

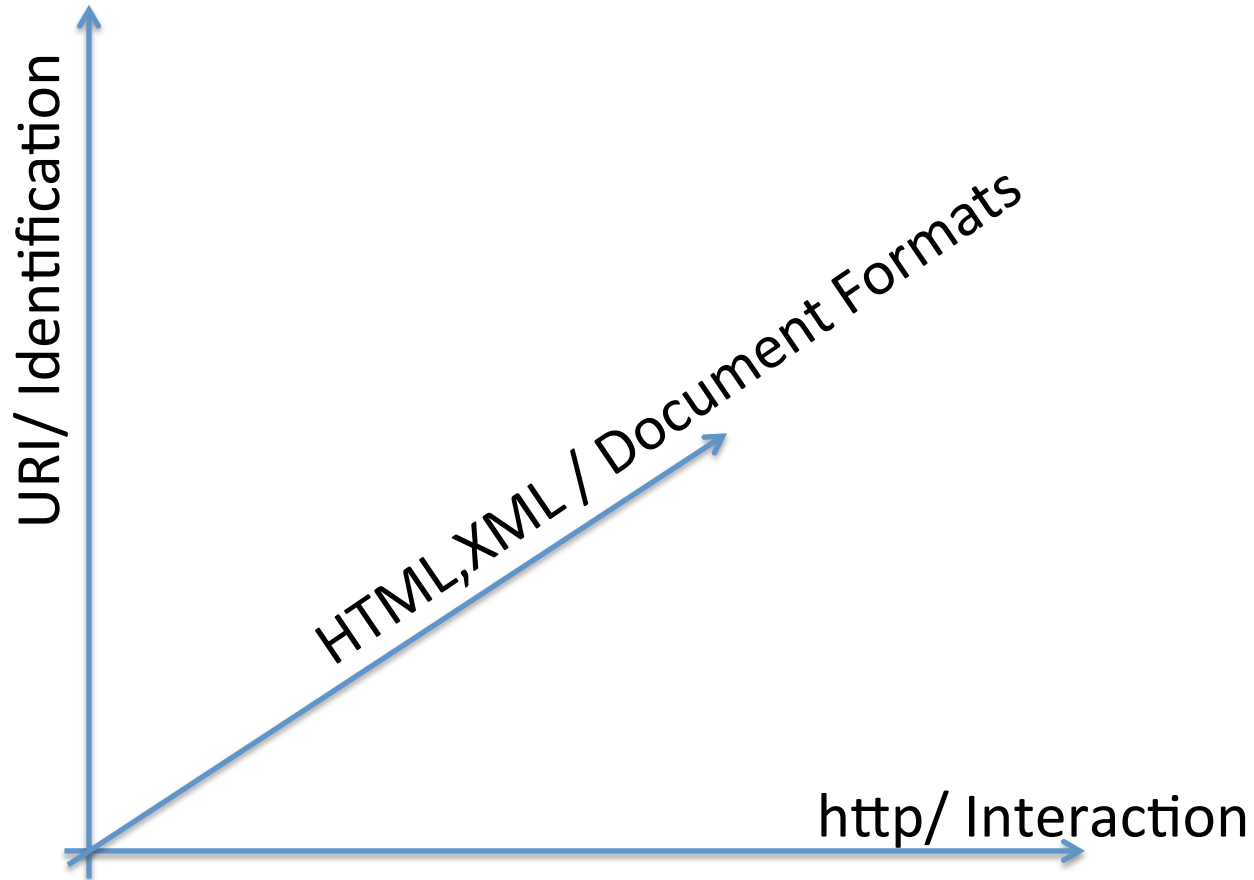


# Key Architectural Components of The Web

- 1. Identification (URI) of web resources**
- 2. Interaction (http) with web resources**
- 3. Standardized Document Formats (HTML, XML)**

W3C's Technical Architecture Group (TAG) released in 2004 a document describing architectural issues of the World Wide Web: <http://www.w3.org/TR/webarch/>

# Principle 'Orthogonal Specifications'



What is meant by 'dereferencing' a  
URI?

# What is meant by 'dereferencing' a URI?

"To use [the appropriate] access mechanism to perform an action on the URI's resource is to **'dereference' the URI.**"

[RFC 3986]

What is important to know about URI's and resources they identify?

# What is important to know about URI's and resources they identify?

## A **resource**

- is an entity that can be identified by one (or several) URI
- is an abstract concept: we cannot see, smell, touch, examine a resource
- is not necessarily retrievable through the internet
- a resource's URI is independent of access mechanism or document format of its representation
- A URI only identifies one resource

What are abstract resources, what are informational resources?

# Resource-Centric Architecture

- **Abstract resources:** their essence is not information
  - E.g. a dog, concepts,...
  - <http://www.example.com/ontology/meter>
- **Informational resources:** their essential characteristics can be conveyed in a message
  - Web pages, images, technical specifications
  - E.g. <http://www.flickr.com/user1/photos/image123.jpg>



How may the difference between an **informational resource** and an **abstract resource** be reflected in the response to an HTTP GET request?

How may the difference between an **informational resource** and an **abstract resource** be reflected in the response to an HTTP GET request?

Answer:

HTTP/1.1 200 OK (informational resource)

HTTP/1.1 303 See Other (abstract resource)

What is meant by **URI Equivalence**?

# URI Equivalence

- In principle, URIs equivalent if they identify the same resource
- Practically: based on string comparison and rules defined by scheme definitions (for http scheme: <http://www.ietf.org/rfc/rfc2616.txt>)

http://foo.com  
http://foo.com/  
http://foo.com:80  
http://www.foo.com  
http://132.24.3.1  
http://foo.com/index.html  
/index.html  
http://foo.com?a=b&c=d

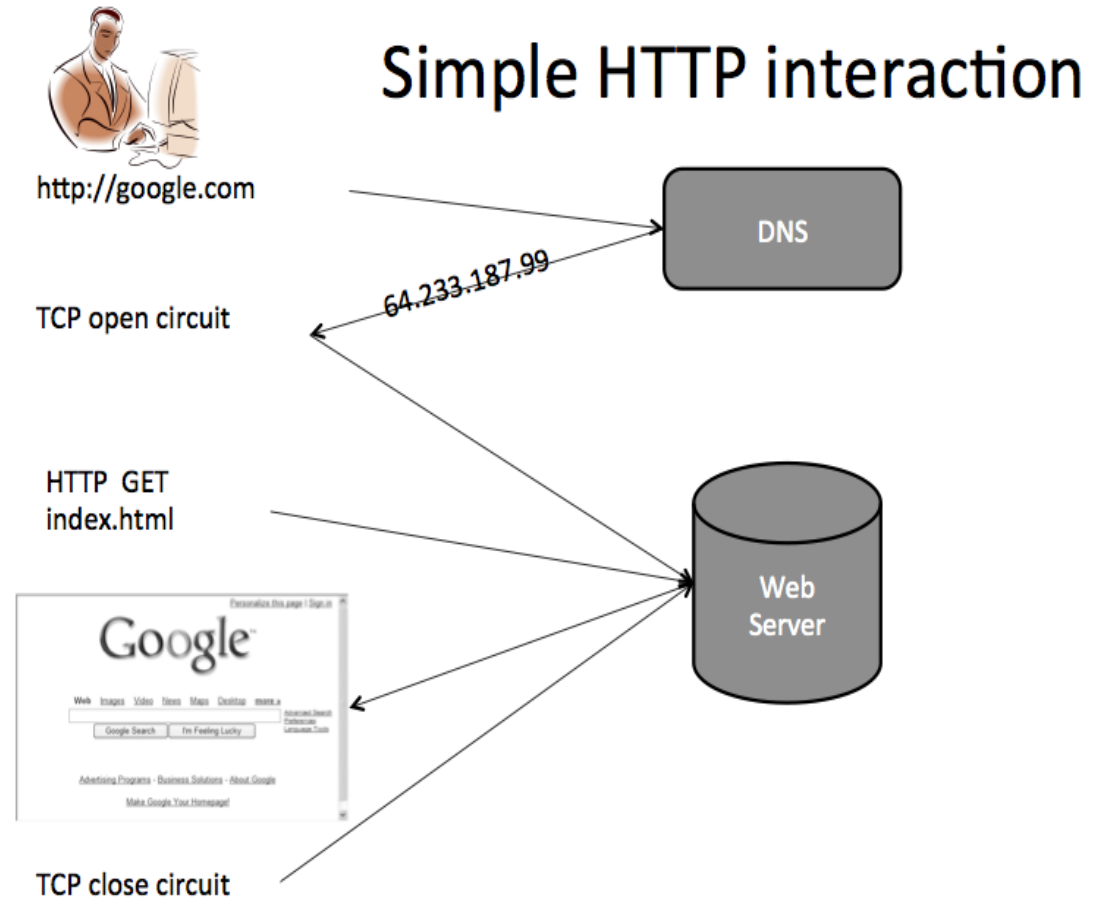


**Equivalent http URIs**

What other Internet protocols does  
http make use of?

# What other Internet protocols does http make use of?

- HTTP is a **request-response** protocol and basis for interaction between web agents and servers and is layered on top of TCP and uses DNS.



What are examples of http verbs?

# http Verbs

- Retrieve a representation of a resource: GET
- Create a new resource: PUT and get a new URI, POST and specify a new URI
- Modify an existing resource: PUT to an existing URI
- Delete an existing resource: DELETE
- Get metadata about an existing resource: HEAD
- See which verbs a resource understands: OPTIONS



# Types of http status codes

– **1xx: ?**

– **2xx: ?**

– **3xx: ?**

– **4xx: ?**

– **5xx: ?**

# http Status Codes

- **1xx: informational**
- 2xx: successful
- 3xx: redirection
- 4xx: client error
- 5xx: server error



This means that the server has received the request headers, and that the client should proceed to send the request body (in the case of a request for which a body needs to be sent; for example, a POST request).

# http Status Codes

- 1xx: informational
- **2xx: successful**
- 3xx: redirection
- 4xx: client error
- 5xx: server error



Standard response for successful HTTP requests.

# http Status Codes

- 1xx: informational
- 2xx: successful
- **3xx: redirection**
- 4xx: client error
- 5xx: server error



301

Moved Permanently

This and all future requests should be directed to the given URI.

# http Status Codes

- 1xx: informational
- 2xx: successful
- 3xx: redirection
- **4xx: client error**
- 5xx: server error



The requested resource could not be found but may be available again in the future. Subsequent requests by the client are permissible.

# http Status Codes

- 1xx: informational
- 2xx: successful
- 3xx: redirection
- 4xx: client error
- **5xx: server error**



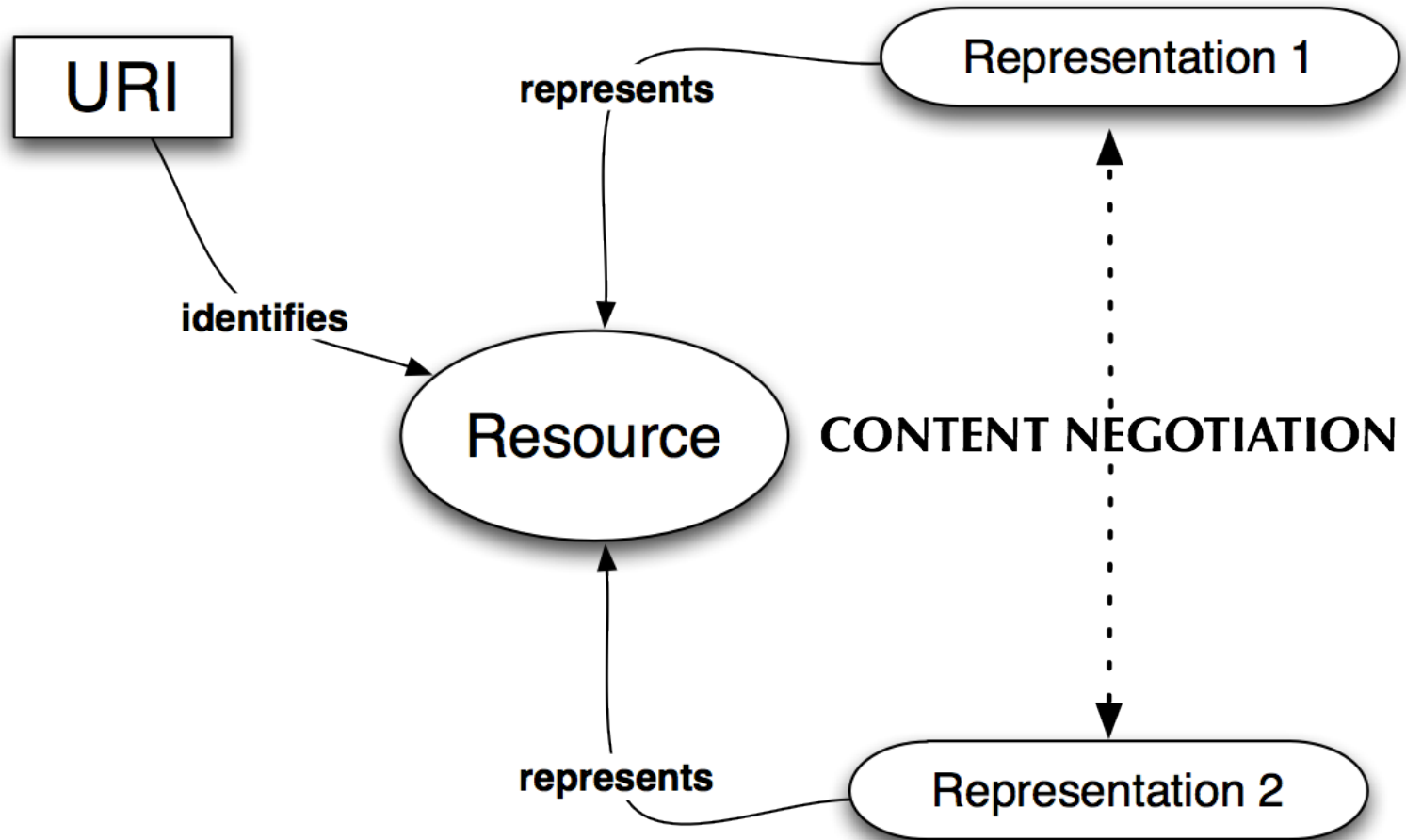
503

Service Unavailable

The server is currently unavailable (because it is overloaded or down for maintenance). Generally, this is a temporary state.

# What is Content Negotiation?

# Content Negotiation





# Content Negotiation

- Content negotiation refers to the practice of **making available multiple representations via the same URI**
- **Basic Idea:** serve the best variant of the representation of a resource based on
  - What variants are available on the web server
  - What the user clients can accept and with what preference (accept headers of http request message)
- One **cannot** determine the type of a resource representation by inspecting a URI for that resource.
  - For example: <http://example.com/page.html> provides no guarantee that representations of the identified resource will be served with the Internet media type "text/html"
  - the URI owner is free to configure the server to return a representation using PNG or any other data format

What is a representation of a web resource?

# A representation is

- The result of applying a service request upon a resource
- What the server determines to be the state of the resource
  - Parameters: time, space, request parameters
  - *“A resource representation encodes information about resource state”* [W3C Recommendation: Web Architecture (vol. 1)  
<http://www.w3.org/TR/webarch/>]
- A package
  - Metadata about the request, server actions, agent
  - Data (pay load) in a specific Internet Media Type (MIME)
- The entity processed by a web agent (browser, crawler)
  - Agents such as crawlers make extensive use of metadata (e.g. last-modified)
- The entity that is the source of links
  - `<a href=“http://google.com”>`

# **STRUCTURED FORMATS/XML**

# XML & Related Technologies overview

Purpose	Structured content	Define Document Structure	Access Document Items	Transform Document
	?	?	?	?
	?	?	?	
	?	?		

# XML & Related Technologies overview

Purpose	Structured content	Define Document Structure	Access Document Items	Transform Document
	<b>XML</b>	<b>XML Schema</b>	<b>XPath</b>	<b>XSLT</b>
	<b>JSON</b>	<b>RELAX NG</b>	<b>DOM</b>	
	<b>YAML</b>	<b>DTD</b>		

How is HTML different from XML?

# How is HTML different from XML?

- XML is a meta language and as such extensible whereas HTML has a fixed tag set
- HTML has more loose syntax rules (XHTML however is well-formed, i.e. fulfills the stricter syntax rules of XML)
- HTML is based on 7-bit ascii, XML on unicode



XML: When to use elements, when attributes to represent information?

# When to use elements, when attributes to represent information?

- Up to the designer; consider:
  - An element can only have one attribute with the same name
  - An attribute cannot be further structured
  - Attributes suitable for most identifiers and references, eg. id, href...

When is an XML document well-formed, when is it valid?

# When is an XML document well-formed, when is it valid?

- Well-formed (absolute): fulfills XML syntax rules
- Valid (relative): conforms to the respective XML schema

# XPath

- step in an XPath expression  
`axisname::nodetest[predicate]`

# XPath expression that delivers the movie titles of all movie nodes ?

```
<?xml version="1.0" encoding="UTF-8" ?>
<?xml-stylesheet type="text/xsl" href="display-catalogue.xsl"?>

<!-- catalogue_extended_revised.xml Created: 2012-09-18 20:03 -->

<catalogue>
  <movie ID="25">
    <title>The Others</title>
    <actors>
      <actor ID="31">
        <name>Nicole Mary Kidman</name>
        <birthdate>1967-06-20</birthdate>
      </actor>
      <actor ID="198">
        <name>Elaine Cassidy</name>
        <birthdate>1979-12-31</birthdate>
      </actor>
    </actors>
    <synopsis>The Others is a 2001 psychological horror film by the Spanish-Chilean
  </movie>
  <movie ID="26">
```

[...]

# XPath expression that delivers the movie titles of all movie nodes

**Solution(s):** //title/text() or /catalogue/movie/title/text()  
or /catalogue/movie/title/node()

```
<?xml version="1.0" encoding="UTF-8" ?>
<?xml-stylesheet type="text/xsl" href="display-catalogue.xsl"?>

<!-- catalogue_extended_revised.xml Created: 2012-09-18 20:03 -->

<catalogue>
  <movie ID="25">
    <title>The Others</title>
    <actors>
      <actor ID="31">
        <name>Nicole Mary Kidman</name>
        <birthdate>1967-06-20</birthdate>
      </actor>
      <actor ID="198">
        <name>Elaine Cassidy</name>
        <birthdate>1979-12-31</birthdate>
      </actor>
    </actors>
    <synopsis>The Others is a 2001 psychological horror film by the Spanish-Chilean
  </movie>
  <movie ID="26">
```

# XPath expression that counts the number of actors listed for each movie ?

```
<?xml version="1.0" encoding="UTF-8" ?>
<?xml-stylesheet type="text/xsl" href="display-catalogue.xsl"?>

<!-- catalogue_extended_revised.xml Created: 2012-09-18 20:03 -->

<catalogue>
  <movie ID="25">
    <title>The Others</title>
    <actors>
      <actor ID="31">
        <name>Nicole Mary Kidman</name>
        <birthdate>1967-06-20</birthdate>
      </actor>
      <actor ID="198">
        <name>Elaine Cassidy</name>
        <birthdate>1979-12-31</birthdate>
      </actor>
    </actors>
    <synopsis>The Others is a 2001 psychological horror film by the Spanish-Chilean
  </movie>
  <movie ID="26">
```

[...]



# XPath expression that counts the number of actors listed for each movie

**Solution(s):** `count(//actor)` or `count(/catalogue/movie/actors/actor)`

```
<?xml version="1.0" encoding="UTF-8" ?>
<?xml-stylesheet type="text/xsl" href="display-catalogue.xsl"?>

<!-- catalogue_extended_revised.xml Created: 2012-09-18 20:03 -->

<catalogue>
  <movie ID="25">
    <title>The Others</title>
    <actors>
      <actor ID="31">
        <name>Nicole Mary Kidman</name>
        <birthdate>1967-06-20</birthdate>
      </actor>
      <actor ID="198">
        <name>Elaine Cassidy</name>
        <birthdate>1979-12-31</birthdate>
      </actor>
    </actors>
    <synopsis>The Others is a 2001 psychological horror film by the Spanish-Chilean
  </movie>
  <movie ID="26">
```

[...]

# XPath expression that returns the birthday information of all actors ?

```
<?xml version="1.0" encoding="UTF-8" ?>
<?xml-stylesheet type="text/xsl" href="display-catalogue.xsl"?>

<!-- catalogue_extended_revised.xml Created: 2012-09-18 20:03 -->

<catalogue>
  <movie ID="25">
    <title>The Others</title>
    <actors>
      <actor ID="31">
        <name>Nicole Mary Kidman</name>
        <birthdate>1967-06-20</birthdate>
      </actor>
      <actor ID="198">
        <name>Elaine Cassidy</name>
        <birthdate>1979-12-31</birthdate>
      </actor>
    </actors>
    <synopsis>The Others is a 2001 psychological horror film by the Spanish-Chilean
  </movie>
  <movie ID="26">
```

[...]

# XPath expression that returns the birthday information of all actors

**Solution(s):** /catalogue/movie/actors/actor/birthdate/text()  
or //birthdate/text() or //birthdate/node()

```
<?xml version="1.0" encoding="UTF-8" ?>
<?xml-stylesheet type="text/xsl" href="display-catalogue.xsl"?>

<!-- catalogue_extended_revised.xml Created: 2012-09-18 20:03 -->

<catalogue>
  <movie ID="25">
    <title>The Others</title>
    <actors>
      <actor ID="31">
        <name>Nicole Mary Kidman</name>
        <birthdate>1967-06-20</birthdate>
      </actor>
      <actor ID="198">
        <name>Elaine Cassidy</name>
        <birthdate>1979-12-31</birthdate>
      </actor>
    </actors>
    <synopsis>The Others is a 2001 psychological horror film by the Spanish-Chilean
  </movie>
  <movie ID="26">
    [...]
```

# XPath expression that returns the birthday information of all actors

What if birthdate an attribute?

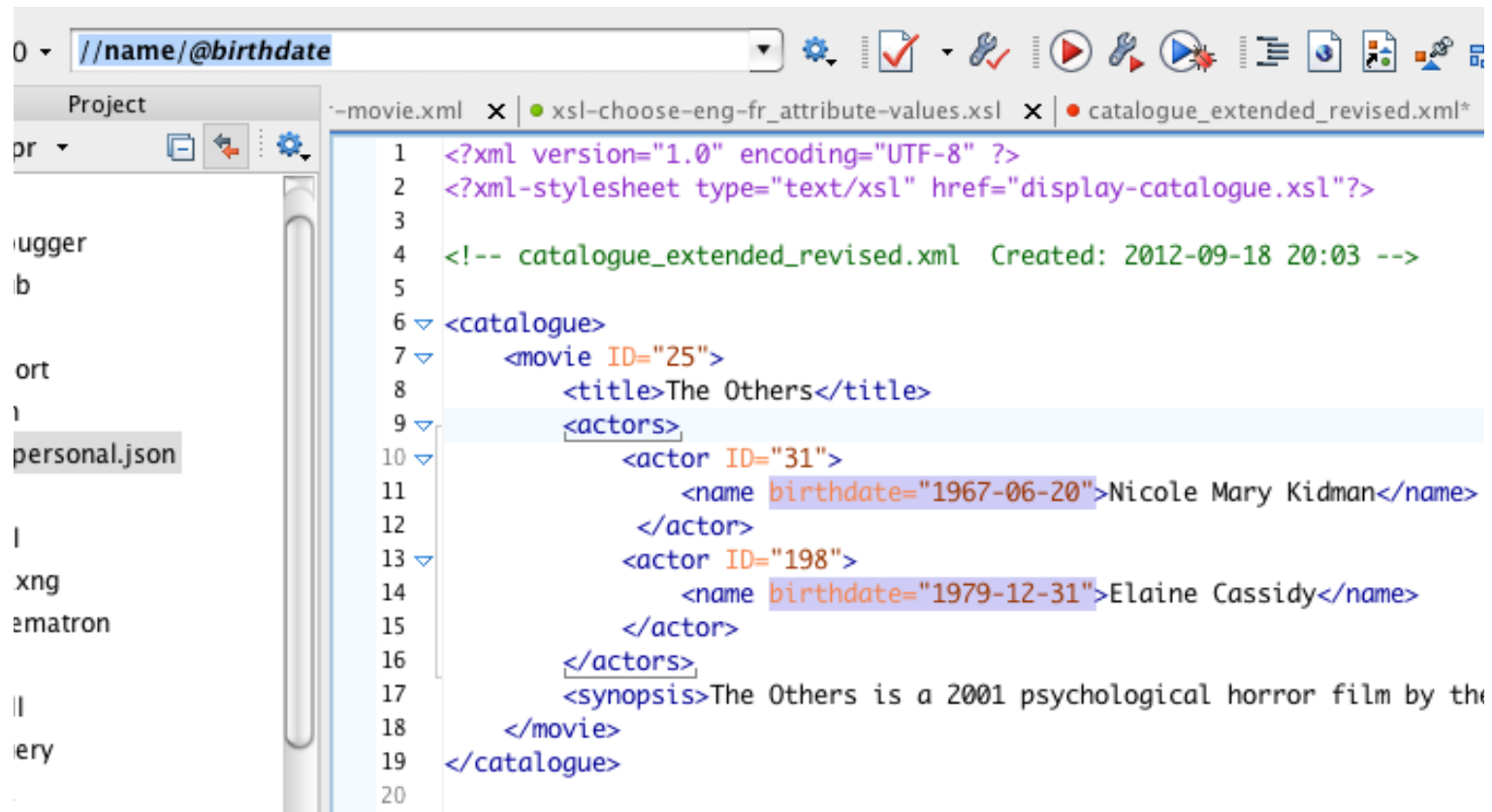
```
<?xml version="1.0" encoding="UTF-8" ?>
<?xml-stylesheet type="text/xsl" href="display-catalogue.xsl"?>

<!-- catalogue_extended_revised.xml Created: 2012-09-18 20:03 -->

<catalogue>
  <movie ID="25">
    <title>The Others</title>
    <actors>
      <actor ID="31">
        <name birthdate="1967-06-20">Nicole Mary Kidman</name>
      </actor>
      <actor ID="198">
        <name birthdate="1979-12-31">Elaine Cassidy</name>
      </actor>
    </actors>
    <synopsis>The Others is a 2001 psychological horror film by the Span
  </movie>
</catalogue>
```

# XPath expression that returns the birthday information of all actors

**Solution(s):** `//name/@birthdate` → `birthdate="1967-06-20"`



The screenshot shows an XML editor interface. At the top, the XPath expression `//name/@birthdate` is entered in the search bar. Below it, the XML code is displayed with line numbers 1 through 20. The XML structure is as follows:

```
1 <?xml version="1.0" encoding="UTF-8" ?>
2 <?xml-stylesheet type="text/xsl" href="display-catalogue.xsl"?>
3
4 <!-- catalogue_extended_revised.xml Created: 2012-09-18 20:03 -->
5
6 <catalogue>
7   <movie ID="25">
8     <title>The Others</title>
9     <actors>
10      <actor ID="31">
11        <name birthdate="1967-06-20">Nicole Mary Kidman</name>
12      </actor>
13      <actor ID="198">
14        <name birthdate="1979-12-31">Elaine Cassidy</name>
15      </actor>
16    </actors>
17    <synopsis>The Others is a 2001 psychological horror film by the
18  </movie>
19 </catalogue>
20
```

The XML code is color-coded, and the `birthdate` attributes in the actor elements are highlighted in blue. The editor's project pane on the left shows a file named `personal.json`.

# XPath

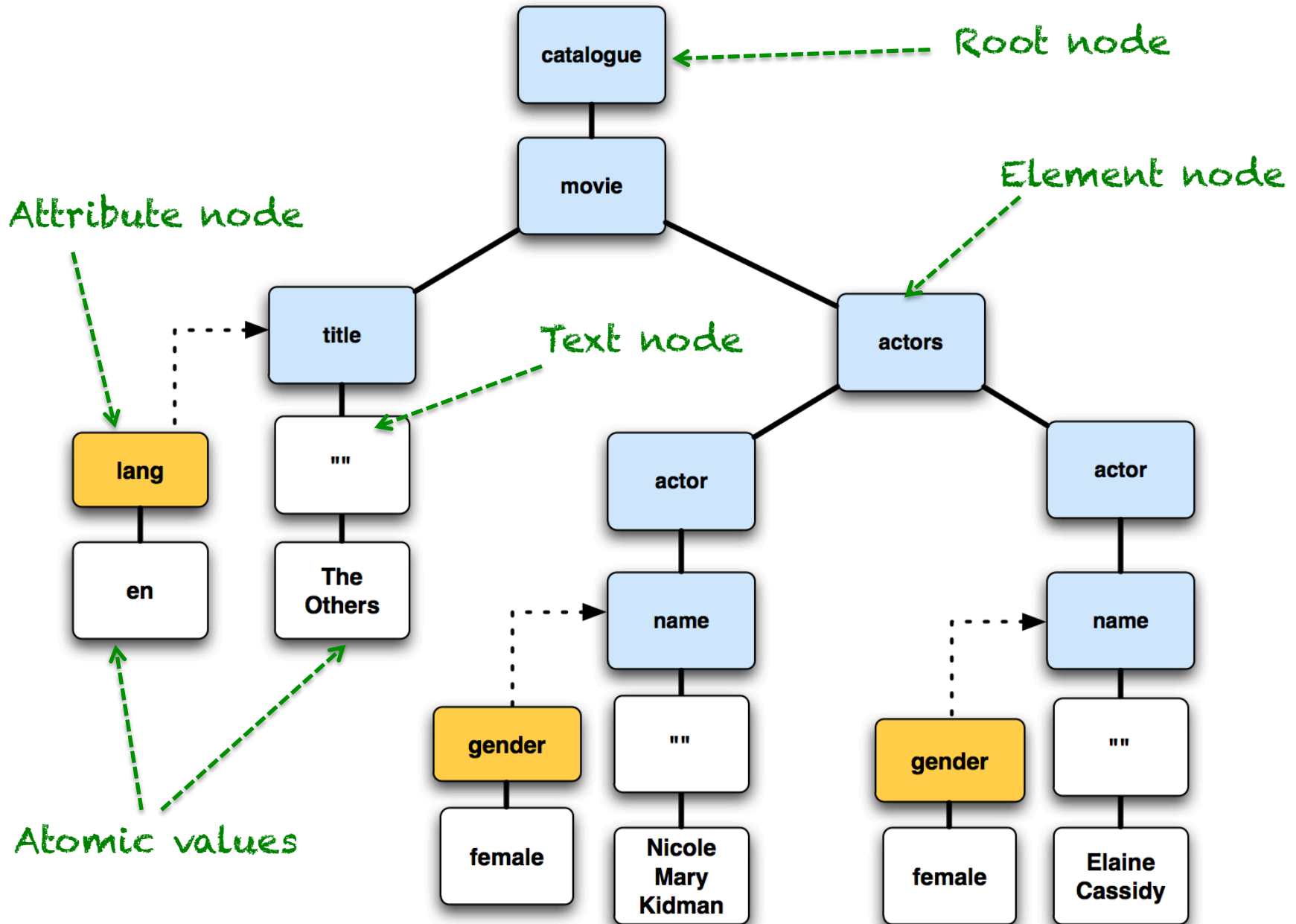
birthdate an element: can access text content as a node

The screenshot shows an IDE window with the following components:

- Top Bar:** XPath 2.0, //birthdate/node()
- Project Panel (Left):** sample.xpr, folders: css, debugger, epub, fo, import, json (personal.json selected), jsp, nvdI, relaxng, schematron, svg, wsdI, xquery, dita, docbook, ooxml, tei
- Editor (Right):** -movie.xml, xsl-choose-eng-fr\_attribute-values.xsl, catalogue\_extended\_revised.xml\*  
Code:

```
1 <?xml version="1.0" encoding="UTF-8" ?>
2 <?xml-stylesheet type="text/xsl" href="display-catalogue.xsl"?>
3
4 <!-- catalogue_extended_revised.xml Created: 2012-09-18 20:03 -->
5
6 <catalogue>
7   <movie ID="25">
8     <title>The Others</title>
9     <actors>
10      <actor ID="31">
11        <name>Nicole Mary Kidman</name>
12        <birthdate>1967-06-20</birthdate>
13      </actor>
14      <actor ID="198">
15        <name>Elaine Cassidy</name>
16        <birthdate>1979-12-31</birthdate>
17      </actor>
18    </actors>
19    <synopsis>The Others is a 2001 psychological horror film by the Spanis
20  </movie>
21  <movie ID="26">
22    <title>The Sea Inside</title>
23    <actors>
24      <actor ID="101">
```

# The XPath Data Model



# XPath Data Model

“Each node has a set of properties that are exposed to XPath in various ways.

- A name. Some nodes, like the root node, comments, and text nodes will always have a name of the empty string ("").  
[...]
- A string value. Text nodes contain the text characters from the source XML, [...] Attributes contain the attribute value  
[...] The root node and element nodes compute their string value by recursively concatenating the string values of all descendant nodes.
- Children. In theory, you can ask about children of any node, but only the root node and element nodes will actually have children.
- A position relative to all other nodes. The overall ordering is called the document order.”



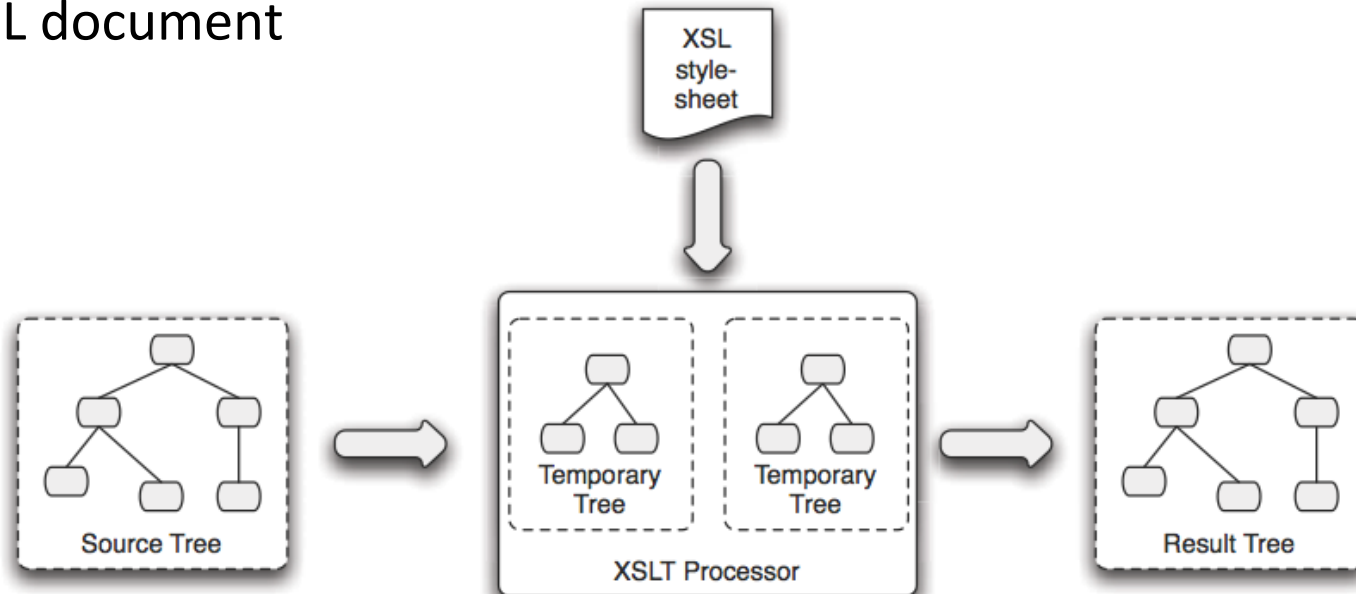
# XPath Data Model

“Additionally, some nodes have the following properties:

- A parent. Every node but the special root node has exactly one parent.
- Attributes. Elements may also contain attributes, which are treated specially and not considered children.
- Namespaces. Namespace nodes are also treated specially, and are not considered to have a child relationship with the element node to which they attach.”

# XSLT

- A transformation in the XSLT language is expressed in the form of an XSL stylesheet
  - root element: `<xsl:stylesheet>`
  - an xml document using the XSLT namespace, i.e. tags are in the namespace <http://www.w3.org/1999/XSL/Transform>
- The body is a set of templates or rules
  - The 'match' attribute specifies an XPath of elements in source tree
  - Body of template specifies contribution of source elements to result tree
- You need an XSLT processor to apply the style sheet to a source XML document



# Another XSLT example

```
<catalogue>
  <movie ID="25">
    <title>The Others</title>
    <actors>
      <actor ID="31">
        <name>Nicole Mary Kidman</name>
        <birthdate>1967-06-20</birthdate>
      </actor>
      <actor ID="198">
        <name>Elaine Cassidy</name>
        <birthdate>1979-12-31</birthdate>
      </actor>
      <actor ID="241">
        <name>Christopher Eccleston</name>
        <birthdate>1964-02-16</birthdate>
      </actor>
      <actor ID="261">
        <name>Alakina Mann</name>
        <birthdate>1990-08-01</birthdate>
      </actor>
      <actor ID="310">
        <name>Eric Sykes</name>
        <birthdate>1923-05-04</birthdate>
      </actor>
    </actors>
  </movie>
</catalogue>
```

XML Source document

# XSL stylesheet

```
<?xml version="1.0" encoding="UTF-8" ?>
<xsl:stylesheet
  xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
  xmlns:xs="http://www.w3.org/2001/XMLSchema"
  xmlns:fn="http://www.w3.org/2005/xpath-functions" version="2.0">

  <xsl:output encoding="UTF-8" indent="yes" method="text" />

  <xsl:template match="/">
    <xsl:apply-templates/>
  </xsl:template>

  <xsl:template match="movie">
    <xsl:apply-templates select="actors/actor"/>
  </xsl:template>

  <xsl:template match="actor">
    <xsl:apply-templates select="birthdate"/>
  </xsl:template>

  <xsl:template match="birthdate">
    <xsl:if test="fn:year-from-date(xs:date(text())) > 1990">
      <xsl:value-of select="preceding-sibling::node()"/>
    </xsl:if>
  </xsl:template>
</xsl:stylesheet>
```

output?

# **INTERNET & WEB NAMING SYSTEMS**

# Internet & Web Naming Systems

1. **IP addresses**
2. **Domain names** e.g.  
[www.cs.cornell.edu](http://www.cs.cornell.edu), [www.cornell.edu](http://www.cornell.edu)
3. **http URIs**, e.g. <http://www.cornell.edu/>
4. **XML Namespaces**, e.g.  
<http://www.cornell.edu/course-catalogue>

## Questions:

- What is being named, by whom?
- How are these naming systems related/dependent on one another (or not related)?
- How do they tie in with the Internet or Web Architecture?

# IP address

- Required for any device using the Internet (identify sender and receiver of messages using IP/TCP)
- Managed by the Internet Assigned Numbers Authority (IANA)
  - Blocks allocated to Five Regional Internet Registries (RIRs)
- 32 bit (IPv4) or 128 bit (IPv6) binary numbers

# Domain Name

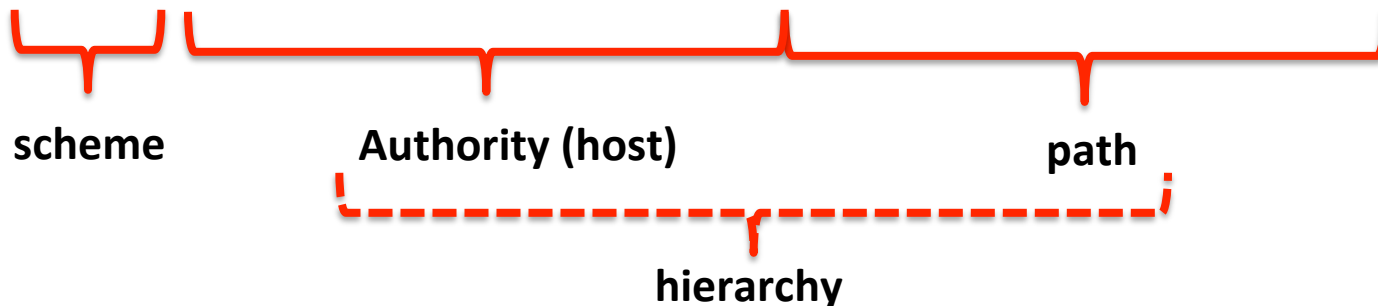
- Managed by Internet Corporation for Assigned Names and Numbers (ICANN)
- Follow rules of Domain Name System:
  - 2-layer virtual space mapping IP addresses to domain names
  - Hierarchically organized into zones: top level domains, second and third-level domain names,...
  - Each zone served by domain name servers
- A domain name identifies a realm of administrative authority
  - e.g. to assign domain names to an IP resource such as the hostname of a web server hosting a website  
[www.cs.cornell.edu](http://www.cs.cornell.edu)
- Part of Internet architecture; does not pre-suppose the web as an http application (invented in the 80's)



# http URIs

- http schema URIs are identifiers for web resources
- managed by 'authority': has been delegated to DNS authority (presupposes Internet Architecture)
  - Hence http URIs are embedded in the hierarchical namespace governed the owner of a domain name
    - who potentially could set up a http origin server as a host at the given address, listening for TCP connections on a given port
- http URI syntax:

`http://www.infosci.cornell.edu:80/Courses/info4302/2012fa/`

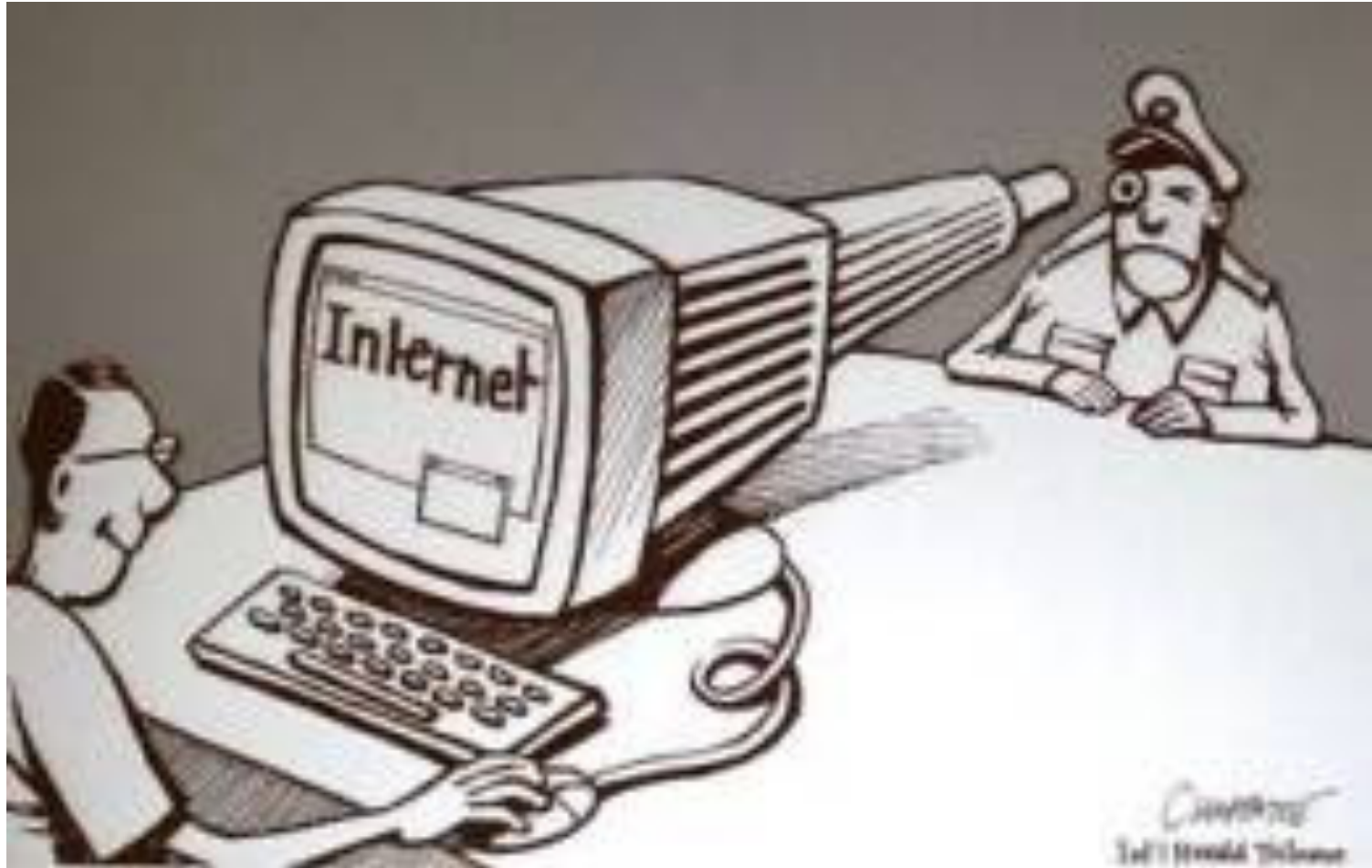


# XML Namespaces

- A namespace identifies a collection of element and attribute names
  - No information about syntax required when using those element names and attribute names → that is what XML Schema is for
  - Idea: restrict uniqueness of names to a defined context (the namespace)
    - In XML documents **qnames** (names qualified by namespace) are used to avoid name collisions within a document
- For unique identification namespace names are URIs
  - Commonly http URIs but not explicitly required by W3C recommendation (no formal relationship with http)
  - Require authors to register domain name with an Internet naming authority
  - XML processors treat namespace identifiers as opaque strings and not as resolvable resources

# Thursday's Topic: Internet Surveillance

Remember: Homework 4 due by 10AM on Thursday



Cartoon by: Patrick Chappatte (International Herald Tribune)