CS2105 Lecture 2

Application Layer

20 January, 2014

CS2105 Lecture 2

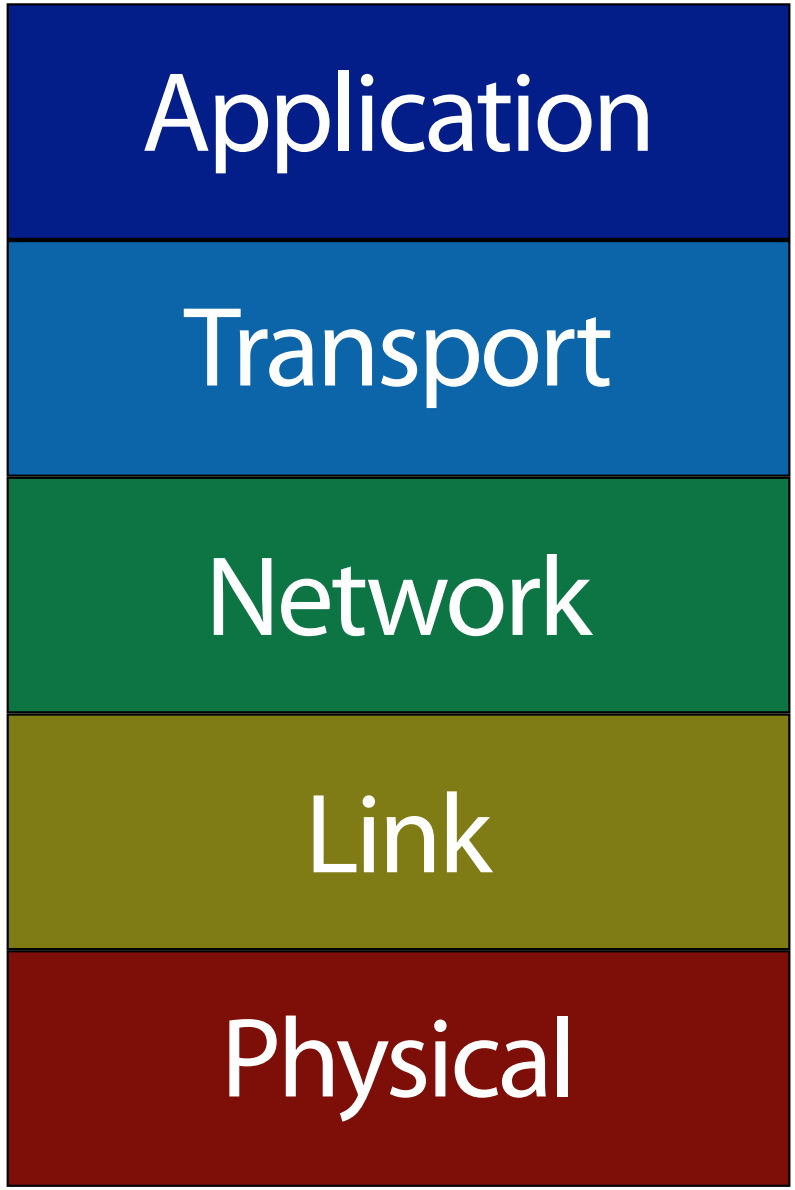
Application Layer

20 January, 2014

After this class, you are expected to:

- be able to choose the right architecture and transport-layer services for your own network application (and justify why).
- understand the basic HTTP interactions between the client and the server, including HTTP request (GET and HEAD) and HTTP response.
- understand the concepts of persistent connection, parallel HTTP connections and stateless protocol.
- understand the services provided by DNS and how a query is resolved.

"This Application Level Protocol is Used by Every Other Internet Application. If You Think I am Refering to the Web or HTTP, You Would Be Wrong."


Application
Transport
Network
Link
Physical

Networked applications runs on **hosts** and consists of **communicating processes**

The **server process** waits to be contacted

The **client process** initiates the communication

Application architecture:

client-server

peer-to-peer

hybrid

Need to identify the source and destination process

Address of a process:
(**host, port number**)

Host addresses
are **32-bit** integers
known as **IP addresses**,
represented by four numbers

Host addresses
are **32-bit** integers
known as **IP addresses**,
represented by four numbers

We are going to talk a lot more about IP addresses in Lecture 6.

Recap: *k* bits can represent 2^k different values.

Ports are **16-bit** integers
(1-1023 are reserved for OS)

IANA coordinates the assignment of port number.

IANA coordinates the assignment of port number.

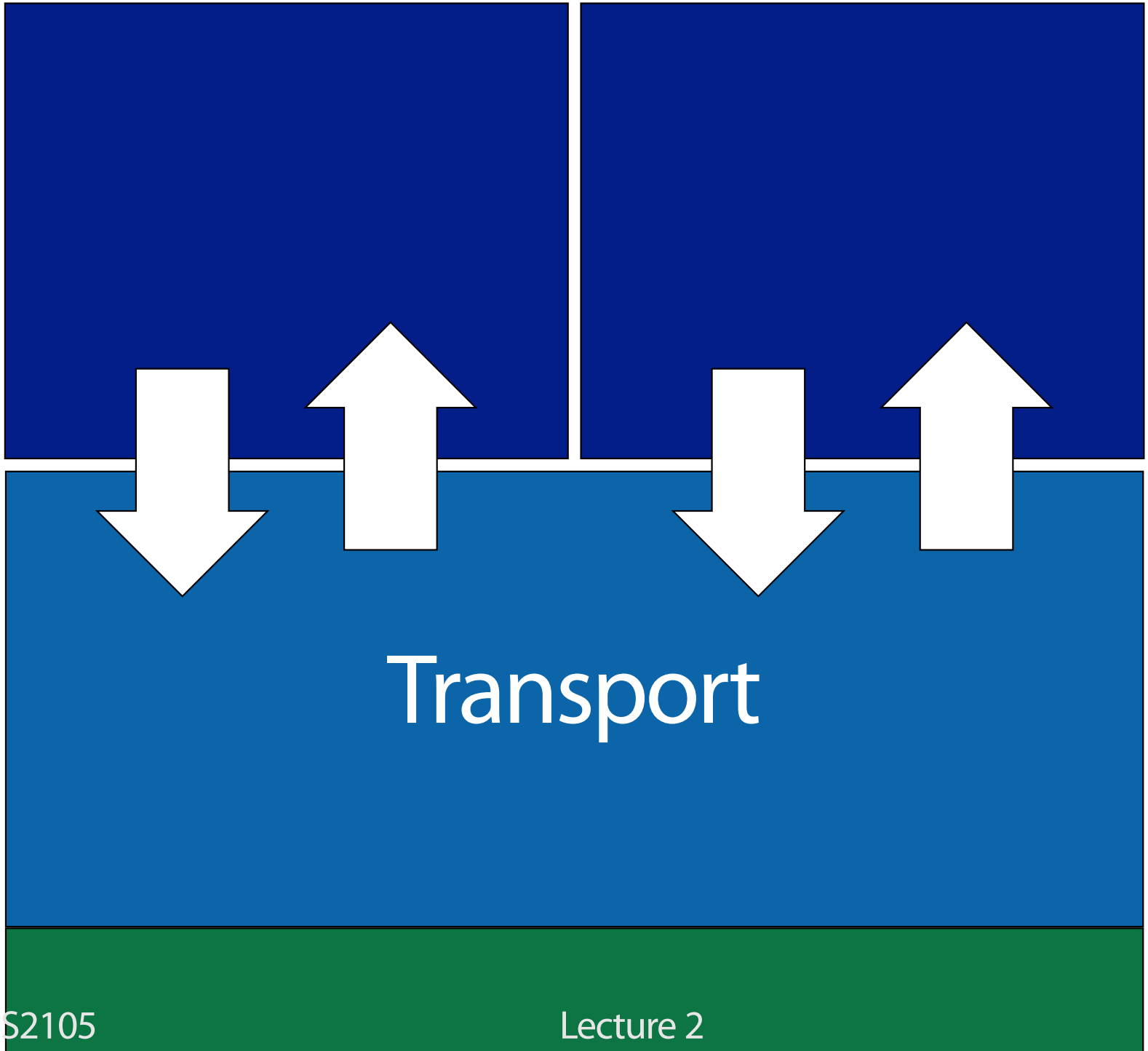
You can find the list of port numbers at `http://www.ietf.org/assignments/port-numbers`. Look for familiar port numbers, such as HTTP, SSH, Battle.Net. For the list of ports, including unofficial ones not registered with IANA, check out `http://en.wikipedia.org/wiki/List_of_TCP_and_UDP_port_numbers`.

Socket is the software interface between processes and the Internet.

initialize a socket
listen for a connection
accept a connection
request a connection
send a message
receive a message
close the socket

initialize a socket
listen for a connection
accept a connection
request a connection
send a message
receive a message
close the socket

The actual APIs for Java will be shown to you during Lecture 3


Transport

Transport service requirements:

loss-tolerance

or

critical?

Transport service requirements:

throughput-sensitive

or

elastic?

Transport service requirements:
time-critical or
not?

Transport service requirements:
time-critical or
not?

The other transport service requirement mentioned in the textbook is *security*,

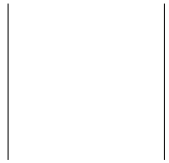
but we will move all discussion about security to Lecture 8 (Week 10).

Transport protocols:
TCP
and
UDP

TCP is
connection-oriented,
congestion-controlled,
and **reliable**.

TCP takes one round trip time (RTT) to establish a connection.

TCP takes one round trip time
(RTT) to establish a connection.



This slide is a gross simplification of the actual method TCP uses to establish a connection. The details will be discussed during the TCP lecture.

TCP provides

no gurantees on throughput and delay

UDP is
connection-less,
not congestion-controlled,
and **not reliable**.

when writing network application,

ask:

what architecture?

what type of services?

how messages are exchanged?

HTTP
Hyper-Text Transfer Protocol

HTTP

Hyper-Text Transfer Protocol

No time to cover everything about HTTP here. Please read up on your own about cookie (Section 2.2.4) and Web caching (Sections 2.2.5 and 2.2.6) to learn more about these two important but straightforward topics.

Web page
HTML file
Web object
URL

persistent

vs.

non-persistent

stateless

vs.

stateful

pipeline

vs.

sequential

```
GET /~cs2105/ HTTP/1.1
Host: www.comp.nus.edu.sg
User-Agent: Mozilla/5.0
Connection: close
```

```
GET /~cs2105/ HTTP/1.1
Host: www.comp.nus.edu.sg
User-Agent: Mozilla/5.0
Connection: close
```

Other common header fields are Content-Length, Content-Encoding, If-Modified-Since, Last-Modified, Server, Cookie, and Set-Cookie. We will use some of these in Assignment 1. For the full list and the purpose of each field, see `http://www.w3.org/Protocols/rfc2616/rfc2616-sec14.html`

```
HTTP/1.1 200 OK
Date: Wed, 19 Jan 2011 06:58:35 GMT
Server: Apache/2.2.6 (Unix)
Connection: close
Content-Type: text/html
```

```
HTTP/1.1 200 OK
Date: Wed, 19 Jan 2011 06:58:35 GMT
Server: Apache/2.2.6 (Unix)
Connection: close
Content-Type: text/html
```

Other common HTTP response codes include 301 Moved Permenantly, 304 Not Modified, 403 Forbidden, 404 Not Found, 500 Internal Server Error, and 503 Service Unavailable. For the full list and the purpose of each code, see

`http://www.w3.org/Protocols/rfc2616/rfc2616-sec10.html`

Demo
with telnet and curl

Demo
with telnet and curl

Sample commands:

- `telnet <hostname> 80`
- `curl -I <URL>`

You can get `curl` from `http://curl.haxx.se/download.html`

DNS

Domain Name Service

Two ways to identify a host:

hostname

(e.g., www.nus.edu.sg)

IP address

(e.g., 137.132.39.133)

DNS **translates between the two**

Demo

with nslookup and dig

Demo
with nslookup and dig

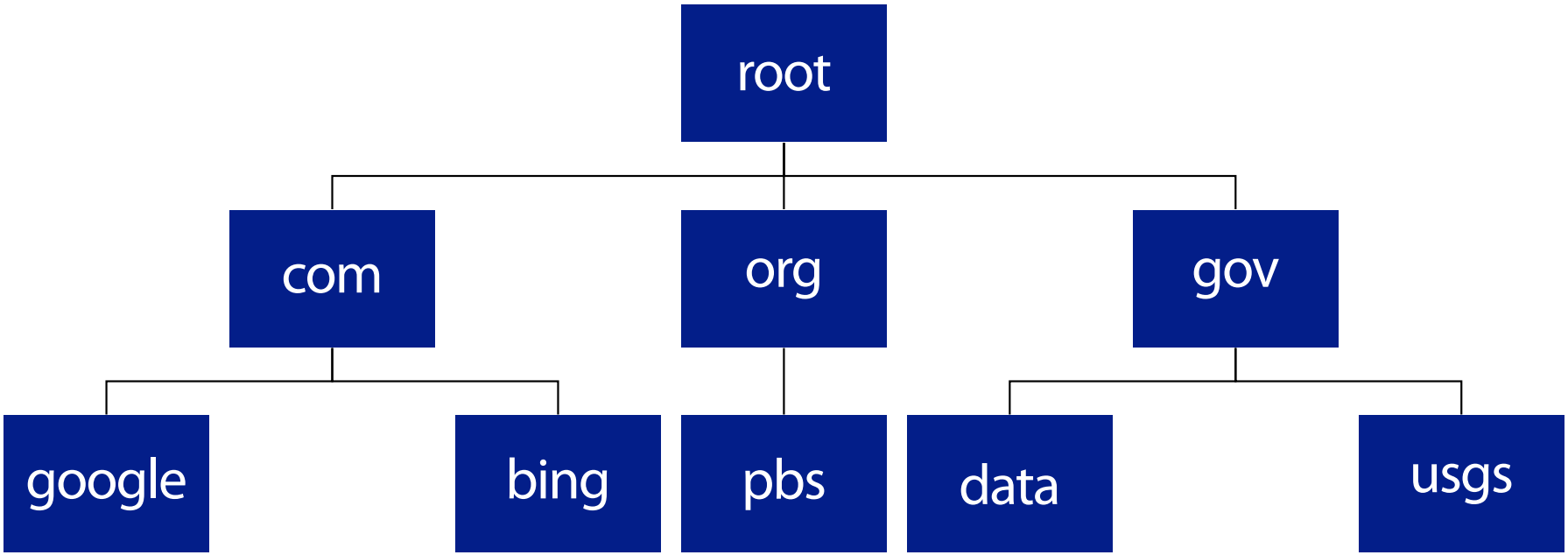
`dig` is installed on many UNIX-based systems. For Windows-based OS, installation instructions for `dig` is available at `http://samsclass.info/40/proj/digwin.htm`

Useful dig options include `+trace` and `+short`.

DNS resource record (name, value, type, TTL)

DNS record types
A, MX, CNAME, NS

DNS uses a **hierarchical distributed databases**


root
com
org
gov
google
bing
pbs
data
usgs

Root servers

Root servers

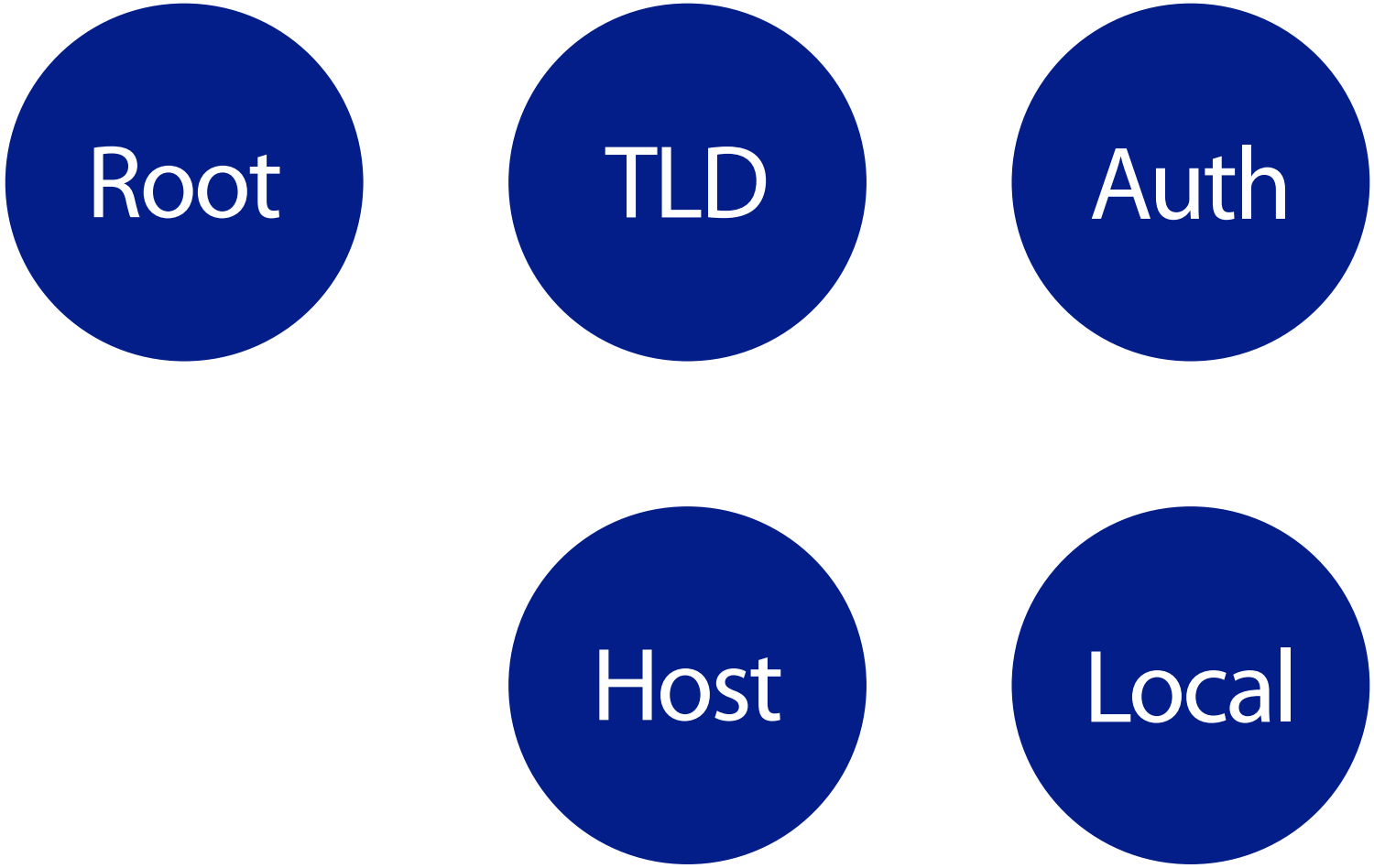
The list of all DNS root servers can be found on

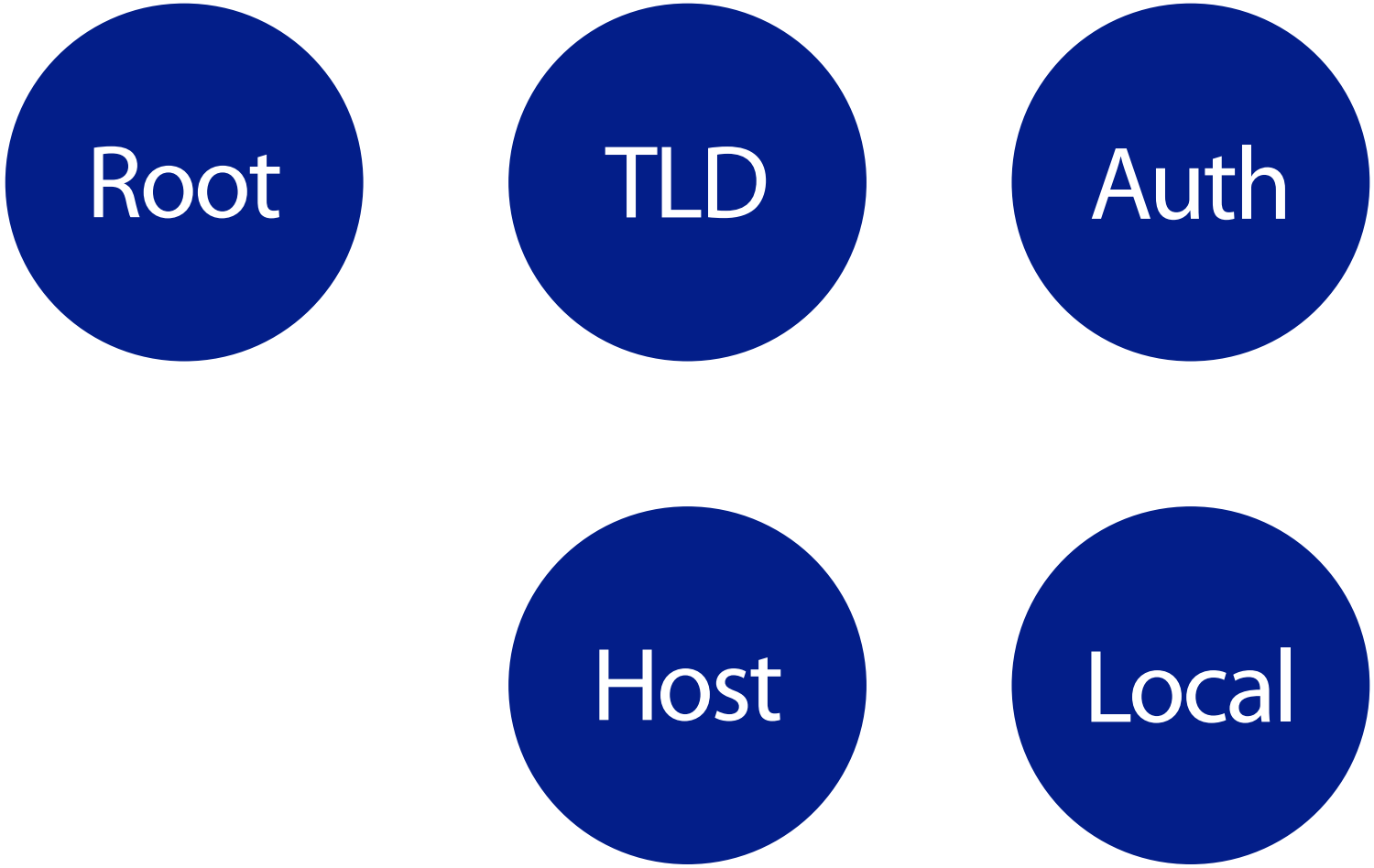
`http://www.root-servers.org/`.

TLD servers

Authoritative servers

Local DNS servers


Root
TLD
Auth
Host
Local


Root
TLD
Auth
Host
Local

DNS runs over **UDP**

DNS Caching