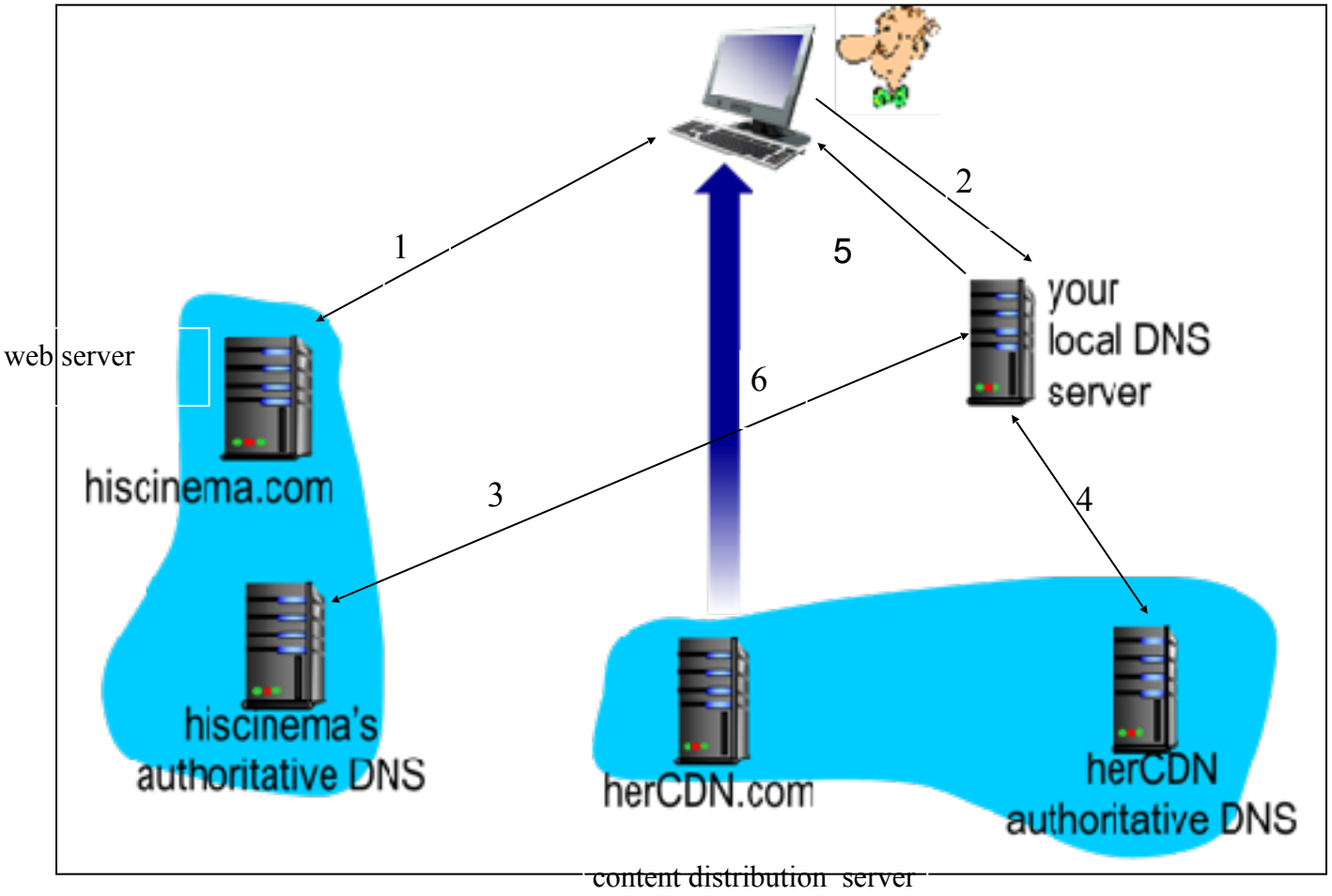
CPS 706 course project : Simple content distribution application

- In this course project, we will develop a simple content distribution system.
- Application will be written in Java or any programming language that you are comfortable with.
- We are going to implement following components:

1. Application client at user's side.
2. Dummy web server www.hiscinema.com
3. Dummy local DNS server.
4. Dummy authoritative DNS server for domain hiscinema.com which will return URL for herCDN.com
5. Dummy authoritative DNS server for domain herCDN.com which will return IP address for www.herCDN.com
6. Dummy content server www.herCDN.com which delivers content to our client using HTTP.

- The architecture of the application is shown in the figure below.

Architecture of dummy content delivery network



Implementation of local dummy DNS (DDNS) server

- Each DDNS has a table of records with record format as (name, value, type).
- We will assume that IP addresses from hisCinema.com and herCDN.com are well known. They will exist in record table in local DDNS.
- During development you can use arbitrary IP addresses but for the demo you will need to populate local DDNS table with IP addresses of lab machines.
- Application client contacts local DDNS using UDP protocol port (different from 53). For development use your own UDP ports and before demo you will receive group of ports to use.

Implementation of local dummy DNS (DDNS) server

- Local DDNS receives request from the client and forwards request to hisCinema.com DDNS.
- DDNS request and reply format should resemble DNS message format explained in the text-book.
- Assume that your local DDNS has the records: (herCDN.com, NSherCDN.com, NS) and (NSherCDN.com, IPher, A).
- (hiscinema.com, NShiscinema.com, NS) and (NShiscinema.com, IPhis, A).
- IP addresses IPher and IPhis will be assigned at the demo. For testing you can use your own.
  - If you use IPher=IPhis then UDP protocol ports for DDNSs must be different

Implementation of authoritative dummy (DDNS) server

- Authoritative DDNS for hisCinema.com and herCDN.com listen on UDP protocol ports allocated for DDNS service. Note that if you develop them on the same machine UDP ports will need to be different.
- Authoritative DDNS for hisCinema.com should have one more record type introduced for re-direction. It needs to point to herCDN.com.
- For example: (video.netcinema.com, herCDN.com, R).
- (herCDN.com, www.herCDN.com, CN) and (www.herCDN.com, IPwww, A).

Implementation of web servers www.hiscinema.com and www.herCDN.com

- Web server www.hiscinema.com should contain file index.html with 4 or more links of the type:
  - http://video.hiscinema.com/Fi where i is an integer.
- After selecting the file and parsing the line in html file for the URL client should send GET request to www.herCDN.com/Fi .
- If you are not comfortable in parsing the html file you can retrieve text file from the web server and interactively selct the the entry corresponding to the file you need.
- File transfer (either html or content file) should be implemented using TCP and message format should resemble HTTP as given in textbook.

Interactions between the client and web servers

- We will use version 1.1 of HTTP, as defined in RFC 2616, to transmit html and content files.
- A web server will be able to handle multiple simultaneous requests in parallel. This means that the web server is multi-threaded. In the main thread, the server listens to a fixed port. When it receives a TCP connection request, it creates a new TCP socket and services the request in a separate thread.
- The subset of HTTP 1.1 functionalities needed in this project is
  - "Only GET request needs to be implemented. The URL field in GET method contains the name of the requested file.
  - The server must implement HTTP code 200, 400, 404 and 505 correctly.
  - The server can run on JDK 1.4 or later.

Port number assignment

- In this project we need following protocol ports:
  - UDP protocol ports on DDNS servers to communicate with the client.
  - TCP ports between client and netcinema web server
  - TCP ports on dummy content client and server for content download.
- The same group of port numbers should be used for UDP and TCP because they use different protocols.

Port number assignment

- Each group will be assigned a unique port number group upon our receiving your group information. Before then, you are free to use one that is not widely used on the Internet. Please refer to section "About port numbers“ below
-

About port numbers

- At any given time, multiple processes can use either UDP or TCP (don't worry we'll learn TCP and UDP in detail in the weeks to come). Both TCP and UDP use 16-bit integer *port numbers* to differentiate between these processes. Both TCP and UDP define a group of well known ports to identify well-known services. For example, every TCP/IP implementation that supports FTP assigns well-known port of 21 (decimal) to the FTP server. TFTP servers, for the Trivial File Transfer Protocol, are assigned the UDP port of 69.
- Clients on the other hand, use *ephemeral ports*, that is short-lived ports. These port numbers are normally assigned automatically by TCP or UDP to the client. Clients normally do not care about the value of the ephemeral port; the client just needs to be certain that the ephemeral port is unique on the client host. The TCP and UDP codes guarantee this uniqueness.
- RFC 1700 (Request For Comments - this is the standard name for Internet documents) contains the list of port number assignments from the *Internet Assigned Authority (IANA)*. However, usually the file Port number is more up-to-date than RFC 1700. For registries also see Registries. The port numbers are divided into three ranges:
- The *well-known ports*: 0 through 1023. These port numbers are controlled and assigned by IANA. When possible the same port is assigned to a given server for both TCP and UDP. For example, port 80 is assigned for a Web server, for both protocols, even though all implementations currently use only TCP.

About port numbers

- The *registered ports:* 1024 through 49151. These are not controlled by the IANA, but the IANA registers and lists the uses of these ports as a convenience to the community. When possible the same port is assigned to a given service for both TCP and UDP. For example, ports 6000 through 6063 are assigned for an X Window server for both protocols, even though all implementations currently use only TCP. The upper limit of 49151 for these ports is new, as RFC 1700 lists the upper range as 65535 (FYI only).
- The *dynamic or private* ports, 49152 through 65535. The IANA says nothing about these ports. These are what we call *ephemeral* ports.
- **Some interesting points:** Unix systems have the concept of a reserved port, which is any port less than 1024. These ports can be assigned to a socket by a super user process. All the IANA well-known ports are reserved ports; hence the server allocating this port (such as the FTP server) must have super user privileges when it starts.

About port numbers

- Historically, Berkeley-derived implementations have allocated ephemeral ports in the range 1024-5000. This was fine in the early 1980s, when server hosts were not capable of handling more than 3977 clients at any given time. Therefore some systems allocate ephemeral ports differently to provide more ephemeral ports. For example, Solaris allocates ephemeral ports in the range 32768-65535.
- There are few clients (not servers) that require a reserved port as part of the client-server authentication: the *rlogin* and *rsh* clients are the common examples. These clients call the library function resvport to create a TCP socket and assign an unused port in the range 513-1023 to the socket. This function normally tries to bind port 1023 and if it fails, tries to bind 1022, and so on, until it either succeeds or fails on port 513.
- Note that during program development, you as the client-server application programmer must be responsible to avoid conflicts in choosing protocol port for your server application!