March 26 Lecture Transcript – DNS

Slide 2:

Earlier in the course, we have learned that any machine/computer that wants to be 'a part' of the Internet (i.e., be able to exchange IP packets with other machines in the Internet) has to get assigned a unique IP address. The first part of this address will be (is) actually derived from the IP address of network that this computer currently resides on, and the second half is locally unique to this computer (but generally a random number) …

One of the main challenges associated with IP addresses is the fact that, for the human users, they are very hard to remember. Imagine if you had to memorize the 4 (3-digit) number sequence for every single Web server that you visit on a regular basis.

Domain Name System (DNS) names are sequences of characters/words assigned to different machines and generally intended to serve as 'easy to remember' substitutes for one or a group of IP addresses. For example, the IP address of the machine that run our departmental Web server is 130.63.94.24. However, this machine also has a DNS name – www.cse.yorku.ca - that (in many applications/programs) can be used as a substitute for its IP address. It should be noted, nevertheless, that DNS names – even though easy(er) to remember – also come with some disadvantages, such as:

- unlike IP addresses, they do not contain/provide any information about the actual (physical) location of the respective machine;
- are often of arbitrary size and composition (in terms of the number and type of characters), and as a result are NOT as convenient for 'automated' processing as IP addresses, which are all of the same size (32 bits) and composition.

Slide 3:

There is a common misconception that DNS names and URL addresses are the same. Note, here, that:

- DNS names are used to refer to machines (only!)
- A URL are used to refer to a specific file on a specific machine.

In fact, a URL generally consist of three parts:

1) The name of the protocol which will facilitate the transmission of the file referred to in this URL over the Internet (typically HTTP protocol).
2) The DNS name of the machine on which the given file is hosted (the host machine).
3) The actual path of the given/target file on the host machine.

Also note that the 2nd part of a URL (the host machine's DNS name) could be replaced with the actual IP address of this machine. To confirm this, try downloading a Web-page from our departmental server by using a URL in which the server's DNS name (www.cse.york.ca) is replaced with its IP address (130.63.94.24).

Slide 4:

The Domain Name System **(DNS system)** – the system that allows/supports the use of DNS names – is actually a world-wide distributed database of IP-address/DNS-name records spread over (i.e., consisting

of) a large number of (cooperating) servers. The communication between these (DNS) servers is facilitated through the use of DNS protocol, and these servers run on port 53 of their respective host machines.

DNS protocol is also run by regular host-machines (think of your laptop or desktop) to facilitates the exchange of information (IP-address/DNS-name records) between the host-machines and DNS servers.

The figure provided in this slide shows that every time a user opens (e.g.) a Web browser and types in a URL address containing the DNS name of the respective Web-server (Step 1), the browser 'engages' the DNS protocol in order to generate and send a DNS query (about the actual IP address of this Web-server) to the local DNS server (Steps 2 and 3). The local DNS server resolves the request and sends the Web-server's IP address back, (Step 4) and only after that the host machine is finally able to create and send IP/TCP packets to the Web-server in question (Steps 5 and 6) …

Slide 5:

In addition to DNS-name to IP-address mapping/translation, DNS system provides a number of other services, including:

- Reverse IP-address to DNS-name mapping/translation;
- Host aliasing – mapping of more complex to simpler (to remember) DNS names;
- Email (server) name resolution;
- Mapping of one DNS name to multiple IP addresses, which is used in cases where (e.g.) Web service of a domain is supported by multiple server machines in order to achieve load balancing and/or server redundancy.

Slide 6:

This slide shows a DNS record containing the DNS-name to IP-address resolution for 'cnn.com' (the symbolic name of the machine hosting CNN's Web-site). The record clearly shows that there are, actually, multiple machines that perform the function of the CNN's Web server …

This record is obtained using nslookup. This service is freely available on most operating systems. You are strongly encouraged to explore the use of nslookup on your own computer.

Slide 7:

When it comes to the actual structure of DNS names, it should be noted that:

1) DNS names are organized in a 'hierarchical manner'. Namely, most DNS names are composed of multiple 'parts' separated by a 'dot'. (E.g., indigo.eecs.yorku.ca). Each part indicates that the respective machine (in this case a machine/server named '**indigo**' which resides on our departmental EECS network) belongs to different hierarchically-structured (inter-related) domains. In this particular example, **indigo** is a machine on **EECS network/domain**, which is a part of a bigger **YORKU network/domain**, which itself is a part of even a bigger **CA domain**. This example also clearly illustrates that the names of bigger/parent domains are (always) aligned right-to-left, with the left-most part of a DNS name being the actual (local) name of the respective machine.

2) Theoretically, there could be up to 127 different hierarchical levels in the DNS (name) structure; though, most real-world cases do not involve more than 4 levels. (I.e., there are typically only up to 4 dot-separated segments in a DNS name.)
3) The name of each level (i.e., each dot-separated segment) can have only up to 63 characters.
4) At each particular DNS level, all sub-domains/children must have different names to ensure the global uniqueness of DNS names - the same way that IP addresses are (must be) globally unique.

Slide 8:

At the top-level of the DNS name hierarchy, there are 2 main groups of (sub)domains:

1) Generic – these are domains that enclose other smaller domains comprising organizations/businesses of similar type, such as:
    - **.edu** (educational organization/business)
    - **.com** (commercial organization/business)
    - **.org** (non-profit organizations)
    - **.net** (IT/networking technology organization), etc
2) Country – these are domains that enclose other smaller domains of organizations/entities geographically located within the same country (e.g., **.ca (Canada)**, **.fr (France)**, **.it (Italy)**, **.cn (China)**, **.in (India)**, etc.)

Slide 9:

In this slide, a detailed list of different Generic sub-domains is provided …

Slide 10:

This slide illustrates the principle behind how the full DNS name of an Internet-connected machine is derived.

Here, assume that a machine (named **challenger**) is situated in a 'local' domain (named **atc**) which is a part of a larger domain/network (named **fhda**) which itself is a part/child of the top-level domain **.edu**. Then, the full DNS name of this machine is derived by moving 'bottom-up' through the hierarchy of the 3 involved domains.

Slide 11:

When it comes to DNS, it is important to note that the principle of 'hierarchical organization' does not apply only to DNS names (i.e., the way these names are derived), **but also to the organization and operation of the actual DNS servers!**

Namely, as explained in Slide 4, DNS servers form a distributed system of 'cooperating' machines which:

1) Aggregately store (and exchange) all of the Internet's DNS records;
2) Supply the Internet's end-hosts with the requested DNS records.

At the top of the DNS hierarchy are the so-called **Root DNS servers**. These servers store only (a few) critical records – specifically, the record/information about the DNS-names/IP-addresses of the **Top-Level Domain (TLD) servers** (i.e., servers that contain information/records pertaining one particular top-level domain including the reference to **Authoritative DNS servers** within this domain).

Authoritative DNS servers typically belong to (i.e., are set-up and administered by) individual organizations/companies, and they contain the DNS records of all publicly accessible machines residing in this domain. Put another way, the role of these servers is to supply the TLD servers (and the rest of the Internet) with the DNS information/records of the machines from their respective domains …

While the goal of Root, TLD and Authoritative DNS servers is to inform the 'rest of the world' about the DNS records of the machines residing in their respective sub-domains (i.e. they support 'outward' flow of DNS information), the role of **Local DNS servers** is opposite to that. The role of **Local DNS servers** is to help the machines residing in their respective domains to acquire DNS records from the 'rest of the world' (i.e., they support 'inward' flow of DNS information).

To better illustrate this point, consider the following:

- If you want to set-up a Web-server and then inform the rest of the Internet about the DNS name (i.e., IP address) of his server, you would need to engage the Authoritative DNS server of your domain.
- If you use a browser to retrieve a page from a Web-server, and your browser/computer needs to resolve/map this server's symbolic name to its actual IP address, you would need to engage the Local DNS server of your domain.

Slide 12:

The figure in this slide shows the geographic location of the 13 logical Root-level DNS servers …

Slide 13:

It was previously explained that every time a browser is provided with a symbolic URL address (DNS name), the browser will engage the DNS protocol in order to resolve this address/name. The part of the browser that actually is responsible for 'running' the DNS protocol and issuing/receiving DNS record is called '**DNS resolver**'. Now, when looking for a particular DNS record …

- In the simplest case, the resolver will contact and obtain the requested information from its Local DNS server.
- However, in a more complex but very possible case, the Local DNS server (at the time when it receives the resolver's request) may not have/contain the requested DNS record. In general, there are 2 ways how a resolver can go about obtaining the DNS record of a particular machine if this record in not found in its Local DNS server:
    1) **Iterative Resolution**: In this approach, the resolver (on its own!) starts probing the Root DNS server and subsequent lower-level DNS servers one-by-one until it reaches the Authoritative DNS server in-charge of the DNS name that needs to be resolved. (This approach is illustrated in the provided figure.) Iterative resolution does not assume any cooperation among the DNS servers – the resolver has to perform all the inquiries on its own.
    2) **Recursive Resolution**: In this approach, the DNS servers are willing to co-operate in order to 'assist' the inquiring machine, and each other, in resolving the original DNS query. (**This approach is illustrated in Slide 14.**) That is, recursive resolution assumes the highest level of cooperation among the DNS servers during the resolution of a specific DNS inquiry. For example, in this approach, if the Local DNS server does not contain the required DNS record, it will (on its own) forward the request to the Root-level DNS server, and the Root-level DNS server will do the same … until the request reaches the right Authoritative DNS server. The

response from the Authoritative DNS server will 'trickle back' to the inquiring machine along the same path (but, clearly, in the reverse direction).

Slide 15:

The figure provided in this slide shows that in most (actual) real-world cases, the process of resolving a DNS request is a combination of both – the Iterative and Recursive approach. In particular, the resolution between end-hosts and their respective Local DNS servers generally tends to be 'recursive', while the resolution between Local DNS servers and top-level DNS servers tends to be 'iterative'.

Slide 16:

If we keep in mind that an average Web-page consists of (approx.) 300 different objects (e.g., images, iframes, JavaScripts, etc.) potentially located at as many different servers/locations, and if we also consider that the retrieval of each of these objects (by the browser rendering the given page) could cause a separate DNS request for the respective server's IP address to be issued 'into' the DNS system, then it is easy to understand/visualized the overall enormous volume of DNS requests that need to be handled by the DNS servers at any point in time …

One way of reducing the overall number of DNS requests that are imposed on the DNS system and its servers is through the use of 'DNS Caches'. DNS Cache is an area of memory in a DNS server or an end-host computer used for storing of recently obtained DNS records, and it allows that already/previously obtained DNS records be (re)used to serve subsequent identical requests.

- For example, as soon as you request a new Web-page, say xyz.com, your browser will issue a DNS request to resolve the given (Web-server's) symbolic name into the server's actual IP address. The DNS response that arrives back will be stored in your browser's/computer's DNS cache, so any subsequent attempts to retrieve the same page (i.e., requests for this server's IP address) will be resolved locally – from the browser's/computer's DNS cache, and will not be forwarded to the Local (or any other) DNS server.
- In a similar manner, a Local DNS server maintains in their own DNS Cache a copy of all already obtained DNS record, and they reuse these stored records to serve all repeat DNS requests, instead of forwarding these requests to higher-level DNS servers …

Of course, it should be kept in mind that the symbolic-name / IP-address mappings can (and do) change over time. (E.g., a symbolic name could be re-assigned to a different IP address.) Thus, every issued/stored DNS record contains an 'expiry period/date', and after this period the DNS record is removed from all the DNS Caches in which it has been stored.

Slide 17:

There are two main types of DNS records/messages: Requests and Responses.
Both types of messages comprise the same type of Header, and additionally:

- Requests also contain the so-called 'Question Section';
- Responses contain 'Questions Section' as well as 'Answer', 'Authoritative' and 'Additional' Section.

The figure provided in this slide illustrates the content of DNS Requests and Responses.

Slide 18:

The purpose and content of different sections in DNS Request and Response messages are outlined in this slide.

Slide 19:

In this slide, more in-depth look into the Header of DNS Requests and Responses is provided.

DNS Header consists of the following 6 2-byte long fields:

1) **Identification** – 16-bit field use to uniquely identify a **specific DNS Request**. This exact value is copied/placed in the Identification field of the respective DNS Response.

2) **Flags** – 16-bit field containing the following subfields:
   2.a) **QR** (Query/Response)– 1-bit value set to '0' if the respective message is a DNS Request, and set to value '1' if the respective message is a DNS Response
   2.b) **OpCode** (Operation Code) – 4-bit field that specifies the type of query carried in this DNS message. E.g., if the value of this field is 0, then the message carries the standard 'symbolic-name to IP-address' query, if the value of the field is 1, then the message carries the inverse 'IP-address to symbolic-name' query.
   2.c) **AA** (Authoritative Answer Flag) – 1-bit field used to indicated whether this response is created by an Authoritative DNS server (in which case the value of the field is 1), or by a non-authoritative DNS server (in which case the value of the field is 0).
   2.d) **TC** (Truncation Flag) – 1-bit field set to 1 if the message has been truncated due to its excessive length, and set to 0 otherwise.
   2.e) **RD** (Recursion Desired) – 1-bit field set to 1 if the 'issuer' of this DNS requests prefers that the server receiving the query attempts to answer the query recursively.
   2.f) **RA** (Recursion Available) – 1-bit field set to 1 if the server creating the response supports recursive queries.
   2.g) **rCode** (Response Code) – 3-bit field that is set to 0-s in every DNS Request, but changed in the respective DNS Response in case that the Request contained some error(s) – e.g., DNS server was unable to respond because there was a problem with how the request was constructed or due to a problem with the server itself …

3) **Number of Question Records** – 16-bit field that specifies the number of questions in the Question section of the message.
4) **Number of Answer Records** - 16-bit field that specifies the number of answers in the Answer section of the message.
5) **Number of Authoritative Records** - 16-bit field that specifies the number of records in the Authoritative section of the message.
6) **Number of Additional Records** - 16-bit field that specifies the number of records in the Additional section of the message.

Slide 20:

In this slide, a Wireshark capture of the DNS Response message containing the resolution for a DNS Query concerning (the symbolic name) www.ietf.org is shown.

Slide 21:

In this slide, a closer look into the 'flag fields' of the DNS Response message from Slide 20 is shown.

You are encouraged to study the value of each of the flags, and related them to their actual meaning …