KHALIFA UNIVERSITY

Spring 2014
Computer Networks
CMPE323

Laboratory Experiment No. 10: Miscellaneous Protocols

Aims and Objectives:

- Introducing Dynamic Host Configuration Protocol (DHCP),
- and Domain Name System (DNS) protocols.

Materials Required:

- IP routers,
- PCs with Ethernet adapters,
- and straight-through/crossover/rollover cables.

Change Log:

- 20-5-2014: original document – mkhonji.
- 21-5-2014: changed `r1.example.com` to `pc1.example.com`, and removed the sentence "*for the name servers instead*" – mkhonji.
- 23-5-2014: changed the command `ip dns enable` to `ip dns server` – mkhonji.

1 Introduction

This lab covers a few common protocols, namely:

- Dynamic Host Configuration Protocol (DHCP) — a protocol that can dynamically inform the clients which parameters they should set. Such parameters could be: client's IP address and its subnet mask, client's default gateway address, the DNS server to use.

  The full list of such parameters can be found in the page `https://www.iana.org/assignments/bootp-dhcp-parameters/bootp-dhcp-parameters.xhtml`. As you see, the list is quite extensive. However, we will only use the most commonly used options.

- Domain Name System (DNS) — a protocol that allows using *names* (such as Google.com) instead of numbers (such as IPv4, IPv6, phone numbers, port numbers, public keys, or even other names) throughout what is called Resource Records (RR). For example, instead of memorizing the IP address of Google servers (which are many), you memorize the name `google.com`.

  The DNS protocol is fairly extensive as far as it relates to the RRs which can be found in the page `https://www.iana.org/assignments/dns-parameters/dns-parameters.xhtml`. However, in this lab we will cover the most widely-used DNS RR, namely the `A` RR which resolves names into IPv4 numbers. E.g. `ping google.com` would work by transparently resolving `google.com` into some IPv4 address such as `173.194.35.32`.

1.1 DHCP

Figure 1 depicts the interaction example between a DHCP client and two DHCP servers, using the DHCP message types: `DHCPDISCOVER, DHCPOFFER, DHCPREQUEST, DHCPACK, DHCPRELEASE`.

All DHCP messages are formated as depicted in Figure 2. What makes different DHCP messages have different types is the inclusion of of the option `53` which indicates the type of the message.

1.1.1 DHCP header fields

- `op` — operation code, 0 for request, 1 for response.
- `htype` — hardware type, 1 for Ethernet.
- `hlen` — hardware address length, 6 for Ethernet as MAC addresses are made of 6 octets.
- `hops` — hops between the client and the DHCP server. Usually 0 unless some DHCP relay exists in the path by which the hop count increases.
- `xid` — transaction ID to set different request/response messages apart from each other.
- `secs` — seconds elapsed since the first request.

```
                Server          Client          Server
            (not selected)                    (selected)

                  v               v               v
                  |               |               |
                  |     Begins initialization     |
                  |               |               |
                  | _____________/|\____________  |
                  |/DHCPDISCOVER | DHCPDISCOVER  \|
                  |               |               |
              Determines          |          Determines
             configuration        |         configuration
                  |               |               |
                  |\             |  ____________/ |
                  | \________    | /DHCPOFFER     |
                  | DHCPOFFER\   |/               |
                  |           \  |                |
                  |       Collects replies        |
                  |             \|                |
                  |     Selects configuration     |
                  |               |               |
                  | _____________/|\____________  |
                  |/ DHCPREQUEST  |  DHCPREQUEST\ |
                  |               |               |
                  |               |     Commits configuration
                  |               |               |
                  |               | _____________/|
                  |               |/ DHCPACK      |
                  |               |               |
                  |    Initialization complete    |
                  |               |               |
                  .               .               .
                  .               .               .
                  |               |               |
                  |      Graceful shutdown        |
                  |               |               |
                  |               |\ ____________ |
                  |               | DHCPRELEASE  \|
                  |               |               |
                  |               |        Discards lease
                  |               |               |
                  v               v               v
```

Figure 1: DHCP client-server interaction timeline — Source RFC2131.

- `flags` — as of RFC 2131 only one flag bit is defined, which is the `BROADCAST` flag. If the flag is set (left-most bit) then DHCP servers would respond to requests by broadcasts (a case when clients cannot

```
0                   1                   2                   3
0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|     op (1)    |   htype (1)   |   hlen (1)    |   hops (1)    |
+---------------+---------------+---------------+---------------+
|                            xid (4)                            |
+-------------------------------+-------------------------------+
|           secs (2)            |           flags (2)           |
+-------------------------------+-------------------------------+
|                          ciaddr  (4)                          |
+---------------------------------------------------------------+
|                          yiaddr  (4)                          |
+---------------------------------------------------------------+
|                          siaddr  (4)                          |
+---------------------------------------------------------------+
|                          giaddr  (4)                          |
+---------------------------------------------------------------+
|                                                               |
|                          chaddr  (16)                         |
|                                                               |
|                                                               |
+---------------------------------------------------------------+
|                                                               |
|                          sname   (64)                         |
+---------------------------------------------------------------+
|                                                               |
|                          file    (128)                        |
+---------------------------------------------------------------+
|                                                               |
|                          options (variable)                   |
+---------------------------------------------------------------+
```

Figure 2: DHCP message format — Source RFC2131.

handle unicast IP packets when not having an IP configured).

- `ciaddr` — the current IP address of the client in case the client has one.
- `yiaddr` — the IP address given to the client by the server.
- `siaddr` — next server's IP address. The server is basically informing the client which server should it communicate to if the client wishes to renew its IP address. Usually set to the IP address of the DHCP server itself.
- `giaddr` — the IP address of the DHCP relay agent in case there is one between the client and the DHCP server. DHCP relay agents are used to allow DHCP clients messages reach DHCP servers that are located in different networks. For example, a single DHCP server can exist in servers

network and handle requests from all clients across different networks (thanks to DHCP relays).

- `chaddr` — the hardware address of the client (MAC address in case of Ethernet).
- `sname` — server's host name (optional).
- `file` — path to a file name that contains an image that can in turn allow the DHCP clients to boot.
- `options` — a list of optional fields.

1.1.2 DHCP common options

The options are basically Type-Length-Value (TLV) elements that are defined by: type of the element, length of the value, and the value itself.

- DHCP message type:

```
 Code   Len  Type
+-----+-----+-----+
|  53 |  1  | 1-9 |
+-----+-----+-----+
```

  1. `DHCPDISCOVER`.
  2. `DHCPOFFER`.
  3. `DHCPREQUEST`.
  4. `DHCPDECLINE`.
  5. `DHCPACK`.
  6. `DHCPNAK`.
  7. `DHCPRELASE`.
  8. `DHCPINFORM`.

- Subnet mask:

```
 Code   Len        Subnet Mask
+-----+-----+-----+-----+-----+-----+
|  1  |  4  |  m1 |  m2 |  m3 |  m4 |
+-----+-----+-----+-----+-----+-----+
```

- DNS server:

```
 Code   Len         Address 1            Address 2
+-----+-----+-----+-----+-----+-----+-----+-----+--
|  6  |  n  |  a1 |  a2 |  a3 |  a4 |  a1 |  a2 |  ...
+-----+-----+-----+-----+-----+-----+-----+-----+--
```

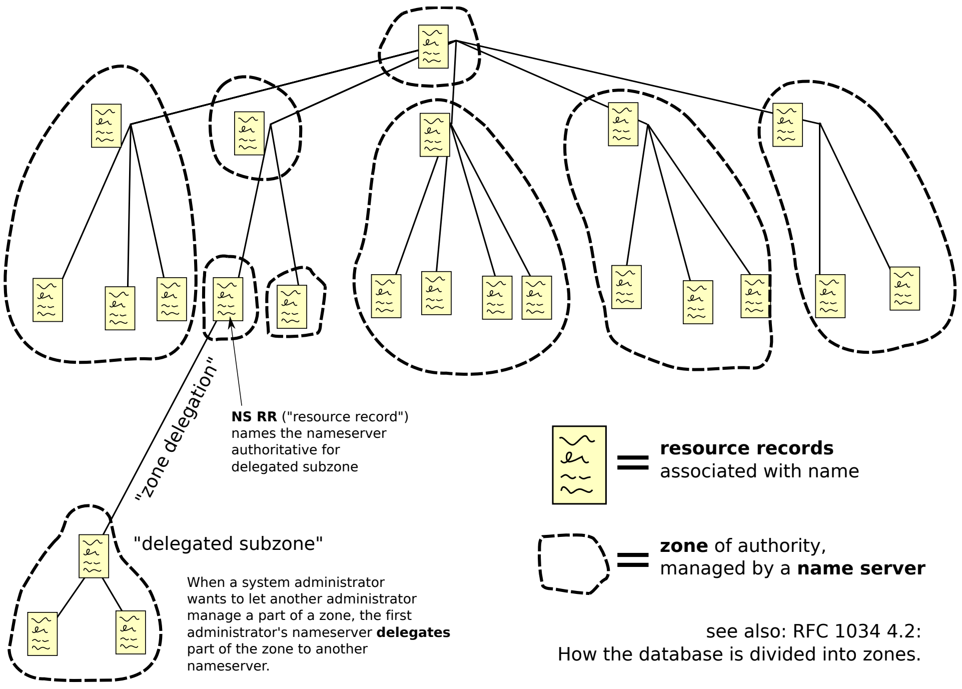
"zone delegation"
NS RR ("resource record") names the nameserver authoritative for delegated subzone
= **resource records** associated with name
= **zone** of authority, managed by a **name server**
"delegated subzone"
When a system administrator wants to let another administrator manage a part of a zone, the first administrator's nameserver **delegates** part of the zone to another nameserver.
see also: RFC 1034 4.2: How the database is divided into zones.

Figure 3: DNS name space — Source: `http://en.wikipedia.org/wiki/File:Domain_name_space.svg`.

1.2 DNS

Figure 3 depicts the hierarchical architecture of DNS which allows different DNS servers to be responsible for different subsets of the name space.

To explain how DNS works, let's take the following example: "`www.google.com.`".

All DNS resolvers attempt to resolve the `www.google.com.`. All of the resolver types return an answer (i.e. resolve the name) if the name is known previously (i.e. cached). However, the resolvers behave differently if the name is not known (i.e. not cached):

- Non-recursive resolvers — such resolvers will attempt to resolve the name `www.google.com.` in case they know the answer (e.g. cached). However, if they don't know the answer, they will return the address of *another* DNS server that is expected to know the answer.

  For example, if some non-recursive DNS server is queried for `www.google.com.` and it does know the server that knows about names below `google.com.` but does not know about the full `www.google.com.` name, then it will respond with the address of the authoritative DNS server that is responsible for names under the name `google.com`.

  If an implementation contacts a non-recursive DNS resolver to resolve names such as `www.google.com.`, the implementation will have to perform the recursion by itself (by recursively following the suggested authoritative name servers).

- Recursive resolvers — when such resolver is queried, it will attempt to resolve the queried name. If the name exists in its cache, it will return it immediately. However, if the queried name does not exist in its cache, it will contact other DNS servers and (if needed) follow recursively with other DNS servers until it knows the answer to the original query.

  Implementations that query recursive DNS servers, they are not required to perform the recursion by themselves as the recursive DNS server will perform it until the answer to the original query is found.

All DNS messages are formatted as depicted in Figure 4.

```
+---------------------+
|        Header       |
+---------------------+
|       Question      | the question for the name server
+---------------------+
|        Answer       | RRs answering the question
+---------------------+
|      Authority      | RRs pointing toward an authority
+---------------------+
|      Additional     | RRs holding additional information
+---------------------+
```

Figure 4: DNS message format — Source RFC1035.

The DNS header is formatted as depicted in Figure 5, where `ID` is the transaction ID, `QR` is a bit that is 0 when the message is a DNS query and 1 when it's a response, `Opcode` specifies the query type (e.g. 0 for standard query, 1 for inverse query), `AA` is a bit that is 1 if the response is from an authoritative DNS server and 0 if otherwise, `TC` is a bit that indicates if the message was truncated along the path, `RD` is a bit that indicates if recursion DNS resolution is preferred, `RA` is a bit that indicates if recursive DNS resolution is available, `Z` is a set of bits that are reserved as of RFC1035 (some of its bits are used by now), `RCODE` is the response code (e.g. 0 for no error, 1 for request format error, 2 for internal server error, etc), `QDCOUNT` is the total number of queries, `ANCOUNT` is the total number of answers, `NSCOUNT` is the total number of authoritative DNS server answers, `ARCOUNT` is the total number of additional RRs.

The `Question` format is depicted in Figure 6, where `QNAME` is the name to be resolved, `QTYPE` is the queried RR type (e.g. A, MX, TXT, etc), and `QCLASS` is the queried RR class (e.g. 1 for Internet (aka IN)).

All query answer RRs, authoritative RRs and additional RRs share the same format as depicted in Figure 7, where `NAME` is the domain name, `TYPE` is the RR type (e.g. A, MX, TXT, etc), `CLASS` is the data class (e.g. 1 for IN), `TTL` is the total number of seconds the answer is advised to be cached, `RDLENGTH` is the data length in octets, `RDATA` is the RR's data (e.g. IPv4 address, another name, etc).

```
                                1  1  1  1  1  1
  0  1  2  3  4  5  6  7  8  9  0  1  2  3  4  5
+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+
|                      ID                       |
+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+
|QR|   Opcode  |AA|TC|RD|RA|   Z    |   RCODE   |
+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+
|                    QDCOUNT                    |
+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+
|                    ANCOUNT                    |
+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+
|                    NSCOUNT                    |
+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+
|                    ARCOUNT                    |
+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+
```

Figure 5: DNS header format — Source RFC1035.

```
                                1  1  1  1  1  1
  0  1  2  3  4  5  6  7  8  9  0  1  2  3  4  5
+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+
|                                               |
/                     QNAME                     /
/                                               /
+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+
|                     QTYPE                     |
+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+
|                     QCLASS                    |
+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+
```

Figure 6: DNS question format — Source RFC1035.

Two commonly used RR types are `A` and `MX` for IPv4 and SMTP server addresses. Their respective `RDATA` fortmats is depicted in Figures 8 and 9.

2 Lab Preparation

1. Connect a PC to the routers console port using a rollover cable[1].
2. Erase the configuration of the routers[2].
3. Physically connect the lab as depicted in Figure 10.

[1]If using Linux: `screen /dev/ttySx` were `x` is the serial interfaces ID that is connected to the console cable. If using Windows: Use Hyperterminal or Putty to connect to `COMx` ports.

[2]`enable`, `erase startup-config`, `reload`, and make sure to answer `no` to all yes/no questions while hitting *enter* for all `confirm` prompts.

```
                                1  1  1  1  1  1
  0  1  2  3  4  5  6  7  8  9  0  1  2  3  4  5
+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+
|                                               |
/                                               /
/                      NAME                     /
|                                               |
+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+
|                      TYPE                     |
+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+
|                     CLASS                     |
+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+
|                      TTL                      |
|                                               |
+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+
|                   RDLENGTH                    |
+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--|
/                     RDATA                     /
/                                               /
+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+
```

Figure 7: DNS RR format — Source RFC1035.

```
+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+
|                    ADDRESS                    |
+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+
```

Figure 8: RDATA format of A RRs — Source RFC1035.

```
+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+
|                  PREFERENCE                   |
+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+
/                   EXCHANGE                    /
/                                               /
+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+
```

Figure 9: RDATA format of MX RRs — Source RFC1035.

4. Configure the interfaces of the routers as depicted in Figure 10. However, do not configure the PCs as we will use DHCP for that.

5. Configure the routing table of the routers. You're free to use static routes[3] or RIPv4[4].

6. Run Wireshark on all involved lab PCs as depicted in Figure 10.

[3]Execute the following in the global configuration mode `ip route <network> <netmask> <nexthop>`

[4]Execute the following in the global configuration mode `router rip, version 2, network 0.0.0.0` (replace commas by LF).

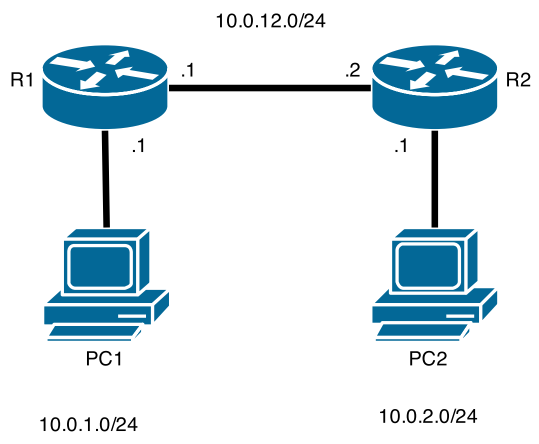

Figure 10: Physical lab topology.

3 Lab experiments

3.1 Setting up a DHCP server

[50 points]

Note: when done, show your work to the lab engineer for grading purposes.

Important note: A DHCP server does not have to be in the router. It is also recommended to use a dedicated DHCP server depending on the load. However, in this lab we will enable the DHCP server that is available inside the Cisco router for simplicity only.

DHCP server configuration steps on a Cisco router:

1. `enable`.
2. `config terminal`.
3. `service dhcp`.
4. `ip dhcp excluded-address <low_address> <high_address>`[5]
5. `ip dhcp pool <pname>`.
6. `network <network_ip> <subnet_mask>`.
7. `default-router <gateway_ip_addr>`.
8. `dns-server <dns_server_addr>`.

Then move to the PCs and perform the following:

[5]This will cause the DHCP server to never lease any IP address in the given interval inclusive. This is used for scenarios where a specific range of IP addresses is statistically assigned to some nodes that might not speak DHCP. For this lab, you may exclude the first n usable IPv4 addresses as long as n is less than 254.

1. Most Operating Systems run an implementation of DHCP as a background process. It is possible that, while you were configuring your DHCP server (or reading this text), your PC has obtained its settings. Verify this by executing: `ifconfig ethX, route -n, cat /etc/resolv.conf` to verify the settings for PC's IP, network mask, default gateway/router, DNS server address respectively.

2. If the setting is not obtained, you can enforce it as follows: `dhclient -r` to release the settings (if any), and then `dhclient` to obtain the settings. This depends on the installed DHCP client.

As Wireshark instances were open on the PCs, answer the following questions:

- What are the exchanged DHCP message types (e.g. DHCPOFFER) and their direction (e.g. from PC to server)?
- What is the purpose of each of the messages?
- Which field(s) in which packet(s) contain the IP address that is assigned to your PC?
- What are the option codes, lengths and values for network mask, default gateway and DNS server.

Note: while your clients are configured to use some DNS server to resolve names, name resolution will not work as the configured DNS server is not configured yet. However, it should work once you perform the steps in the next section.

3.2 Setting up a DNS server

[50 points]

Note: when done, show your work to the lab engineer for grading purposes.

Important note: Using the built-in DNS server of Cisco routers is discouraged due to the limited resources on Cisco routers. This lab will use the built-in DNS server for demonstration purposes only. It is highly recommended to use a more reliable DNS server such as Bind9 instead when possible

DNS server configuration steps on a Cisco router:

1. `enable`.
2. `config terminal`.
3. `ip dns server`.
4. `ip dns primary example.com soa ns.example.com admin.example.com`[6]
5. `ip host pc1.example.com <ip_address>`[7].

[6]`example.com` is your point of delegation (i.e. you can authoritatively respond to queries for `*.example.com` such as `pc1.example.com`. The address `ns.example.com` is the primary name server for this domain. `admin.example.com` is the email address `admin@example.com` of the administrator of this domain name.

[7]This will create a DNS A RR that maps `pc1.example.com` with `<ip_address>`

6. `ip host pc2.example.com <ip_address>`.

7. `ip host ns.example.com <ip_address>`[8].

Assuming you're configuration is correct, you should be able to use the PCs to `ping` each other by using their domain names instead. E.g. `ping pc1.example.com` should succeed. If it fails, start the debugging process by analysing Wireshark instances and observing the output from the DNS server after executing the command `debug domain`.

Once done, answer the following questions by observing Wireshark instances and based on your understanding of the functionality of DNS:

- Explain all of the values of the fields in the DNS query. I.e. justify their values.

- Explain all of the values of the fields in the DNS reply. I.e. justify their values.

3.3 Extra questions (not graded)

- Describe how can an attacker abuse the DHCP protocol, execute it and then describe a possible solution to the problem.

- Describe how can an attacker abuse the DNS protocol (e.g. DNS cache poisoning), execute it, and then describe a possible solution to the problem.

A Relevant protocols

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|Version|  IHL  |Type of Service|          Total Length         |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|         Identification        |Flags|      Fragment Offset    |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|  Time to Live |    Protocol   |         Header Checksum       |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                       Source Address                          |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                    Destination Address                        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                    Options                    |    Padding    |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 11: Internet Protocol (IP) version 4 header format — Source RFC791.

[8] Use the IP address of the primary DNS server for `example.com`. Note: to avoid circular dependencies, glue records should be used.

```
0      7 8     15 16    23 24    31
+--------+--------+--------+--------+
| Source Port     | Destination Port|
+--------+--------+--------+--------+
|     Length      |    Checksum     |
+--------+--------+--------+--------+
|          data octets ...
+---------------- ...
```

Figure 12: UDP Header Format — Source: RFC768.