# dot1x radius dhcp correlations

Access technologies

*INs and OUTs of dot1x/radius/dhcp*

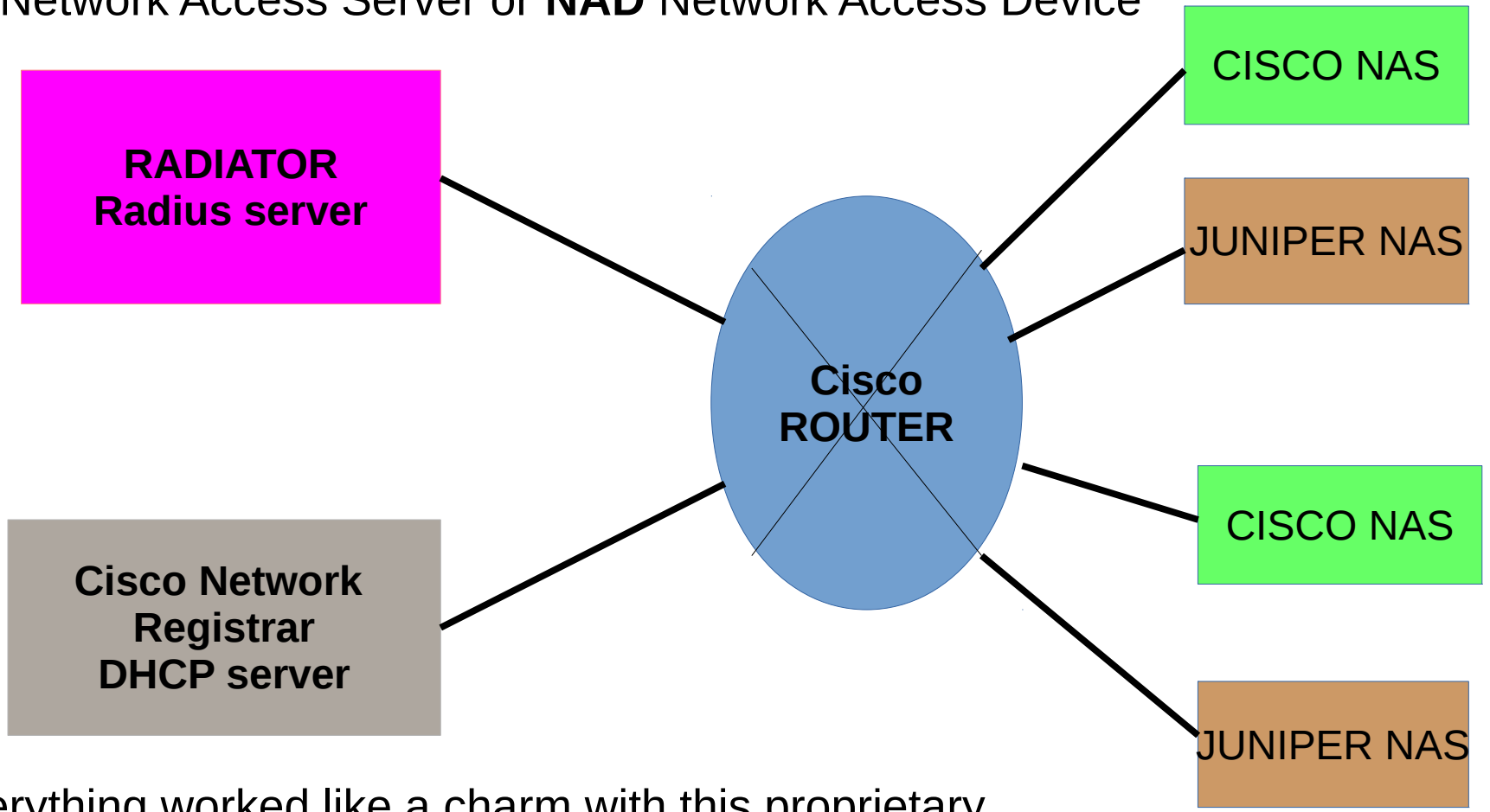
How to get user information

(ip/username bindings)

from  dot1x/radius/dhcp

# RADIUS / DHCP proven configuration

**NAS** Network Access Server or **NAD** Network Access Device




Everything worked like a charm with this proprietary installation ...

# Radius/DHCP servers

- RADIUS servers :
  - FreeRADIUS
  - Radiator
  - Cisco Prime Access Registrar
  - Cisco ISG
  - Microsoft NPS (Network Policy Server)
  - FreeDiameter
  - GNU radius
- DHCP servers :
  - ISC dhcp server
  - Windows dhcp
  - FreeRADIUS/dhcp
  - Cisco Prime Network Registrar
  - Mikrotik
  - Cisco ISG
  - Cisco DHCP server RADIUS proxy
  - ISC KEA

# Open source/Freeware ?

*Can we do the same with open source/freeware s/w and different h/w ?*

*Yes !!!*

*Answer in next slides ..*

# EAP (Extensible Authentication Protocol)

- Rfc2284 1998
- Rfc3784 2004

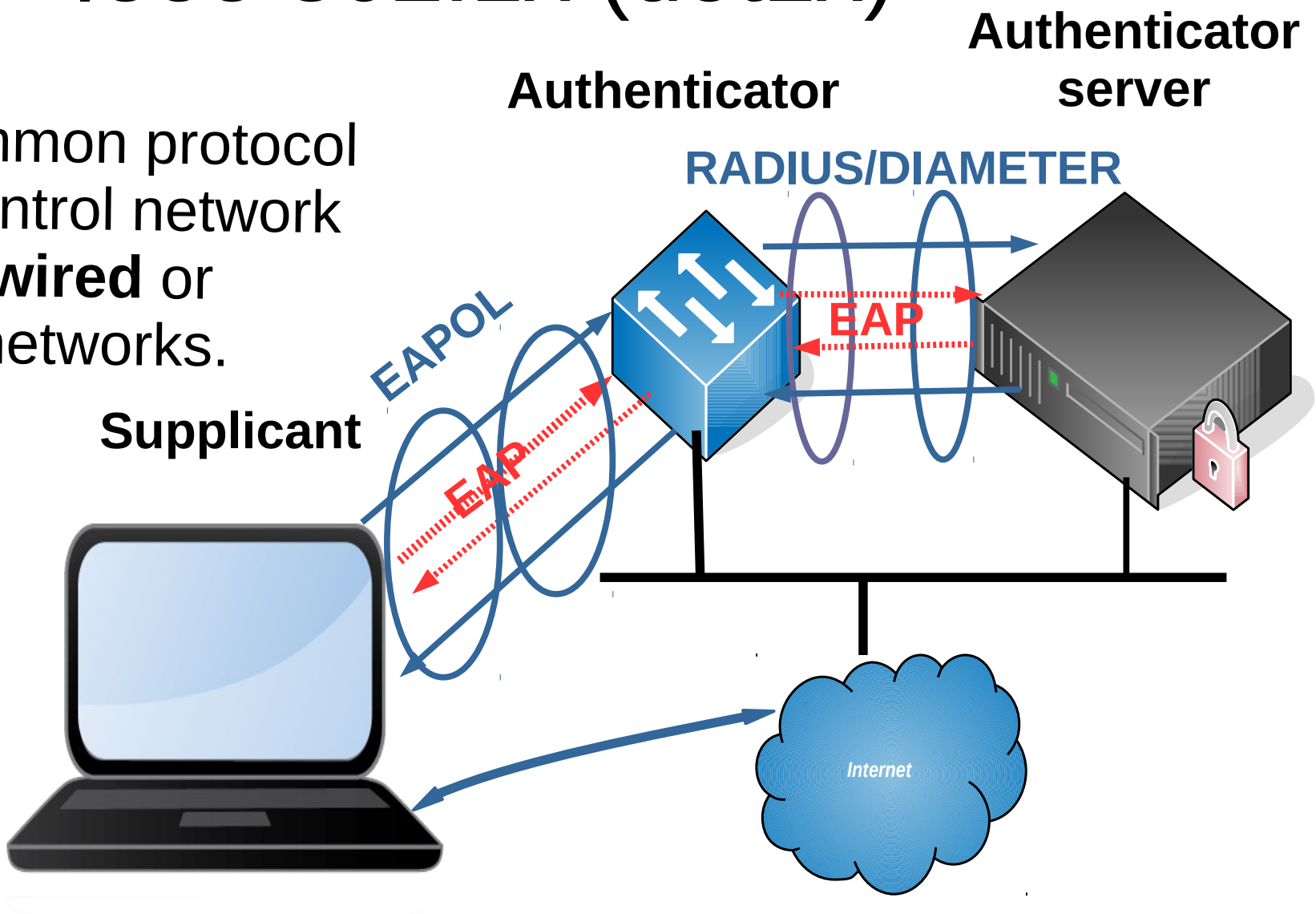
It's a framework for exchanging authentication messages. The wireless standard 802.11 supports with EAP 100 different authentication methods.

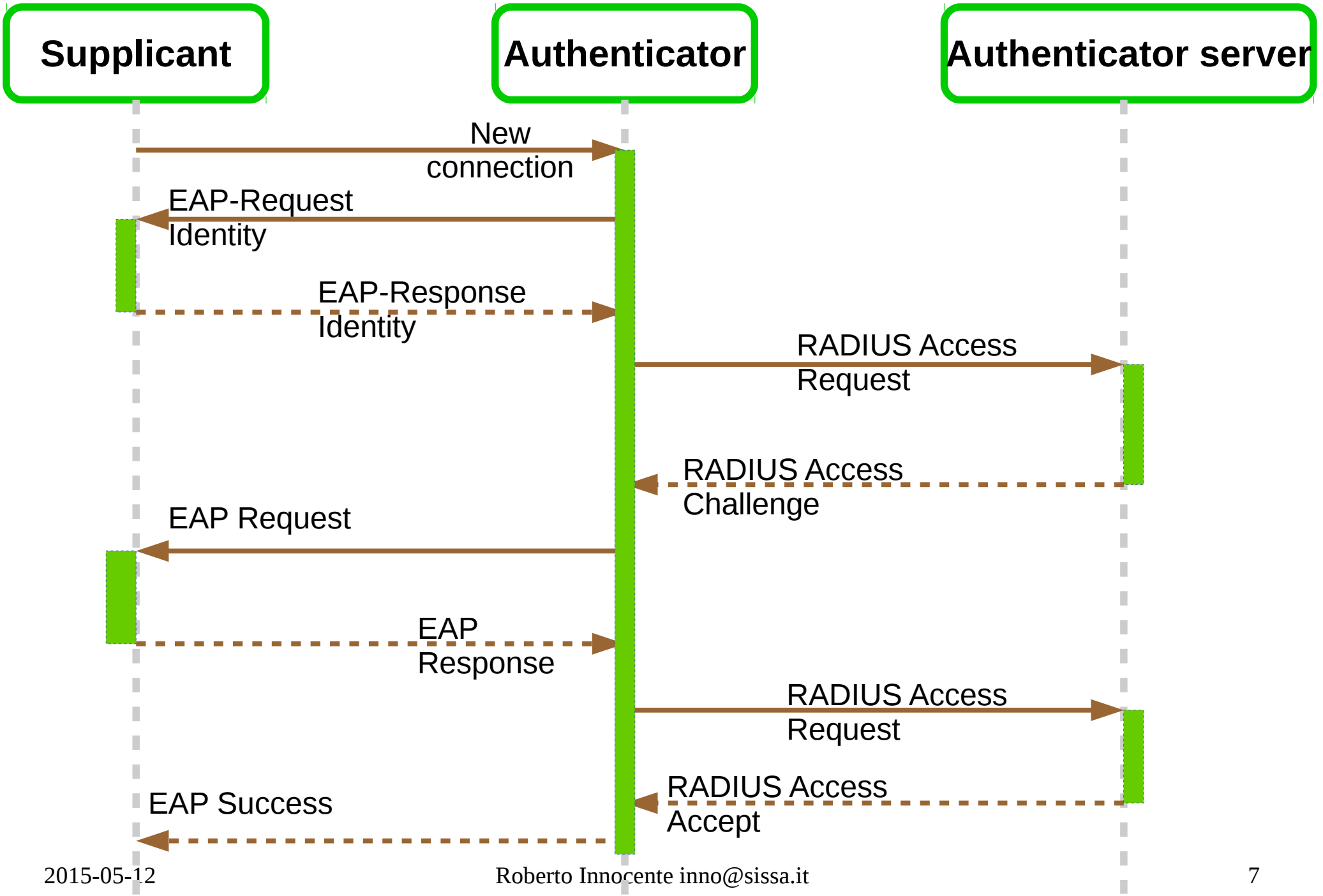
EAP defines only the message format.

Therefore for every network  is defined a way to encapsulate EAP in layer 2 : 802.1x defines EAPOL (EAP over LAN), 802.11i accepts it over WLAN.

# Ieee 802.1x (dot1x)

Today common protocol used to control network access in **wired** or **wireless** networks.




Supplicant
Authenticator
Authenticator server
New connection
EAP-Request Identity
EAP-Response Identity
RADIUS Access Request
RADIUS Access Challenge
EAP Request
EAP Response
RADIUS Access Request
RADIUS Access Accept
EAP Success

# Reliable UDP

Is it a misnomer ?

By definition UDP is an unreliable protocol, but with little effort we can implement over it an old trick : the **Stop-And-Wait** protocol.

The sender for **max-retries** times tries to :

- Send message
- Wait **timeout** seconds for a feedback

After max-retries times the sender gives up, signals a problem and uses a fallback strategy.

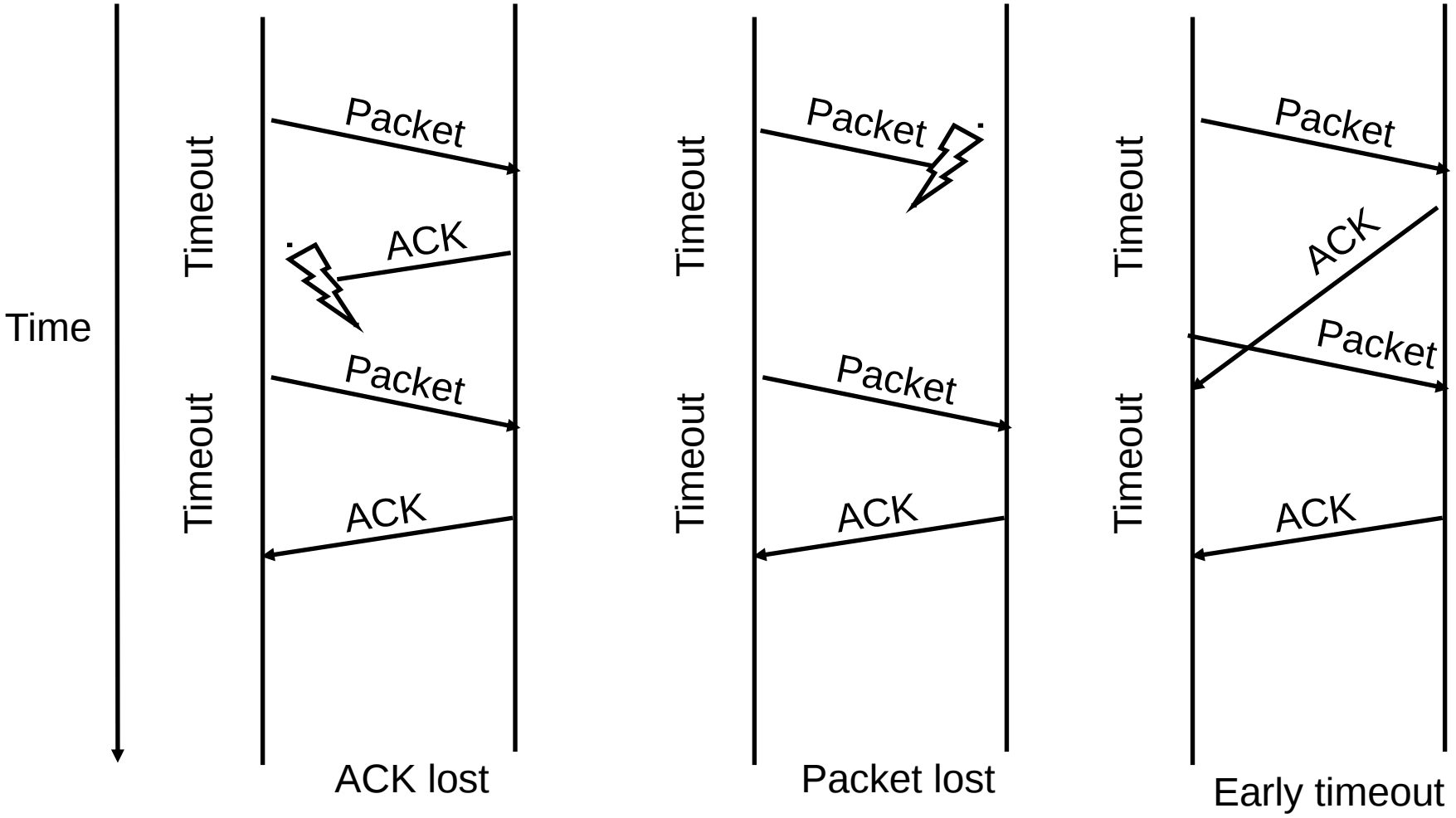
The risk of this protocol is that it can happen that a message is sent multiple times.

Therefore is not indicated for bankomat transactions, but remains very useful in situation in which you need to transmit a long term state , like

- The light is on
- The light is off

In which retransmission is not a problem. RADIUS and DHCP use this kind of reliable UDP.

# Reliable UDP/2



# RADIUS(RFC2865)

Was devised as a protocol to exchange AAA info between a Radius server and Access Servers :

**A**uthentication : check user supplied credentials against a table/database (file, mysql, ldap, ..)

**A**uthorization : tells the Access Server the user can/can't access net and how

**A**ccounting : keeps records of net resources used by users (connection times, bytes transferred)

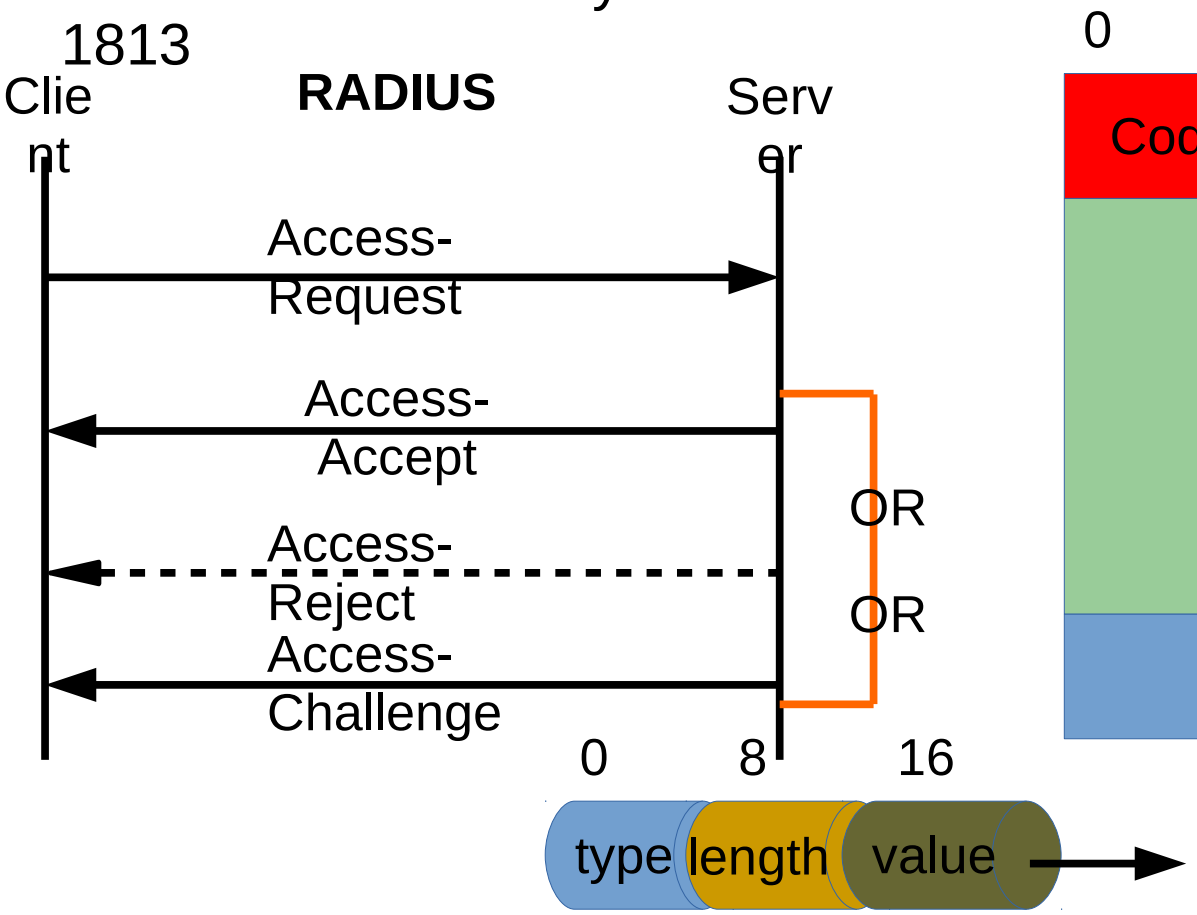
In the '90 the Network Access Servers were mostly racks of modems.

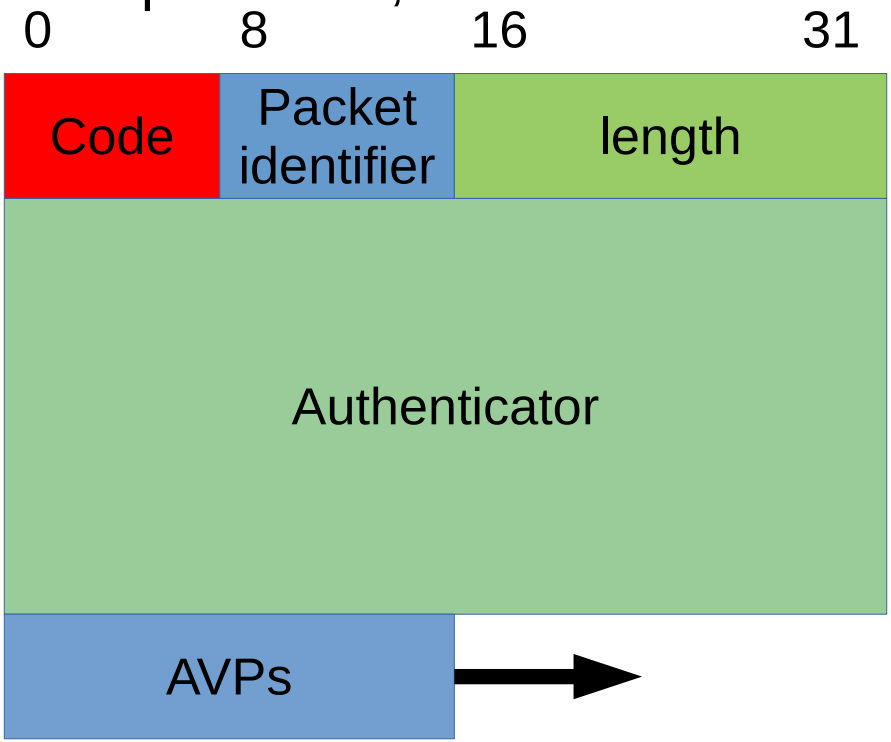
Original version developed by Livingstone Ent. Inc. in 1991 then bought by Alcatel.

Radius is a client/server protocol that uses UDP ports 1812,1813.

# Radius protocol in a slide

Radius uses normally UDP : auth server on port 1812, acct server on 1813







Attribute 1 = User-Name

Attribute 31 = Calling-Station-ID

eg AVP=\0x01\0x06inno

# RADIUS codes

RADIUS **Code** in 1st protocol octet :

- Access-Request (=1)
- Access-Accept (=2)
- Access-Reject (=3)
- Accounting-Request (=4)
- Accounting-Response (=5)

**Packet Identifier** : 1 byte used to match requests and responses

# Radius attributes used

Used by NAS sending Requests:

- **User-Name**
- **Password**
- **Calling-Station-Id** : the mac address of the connecting client

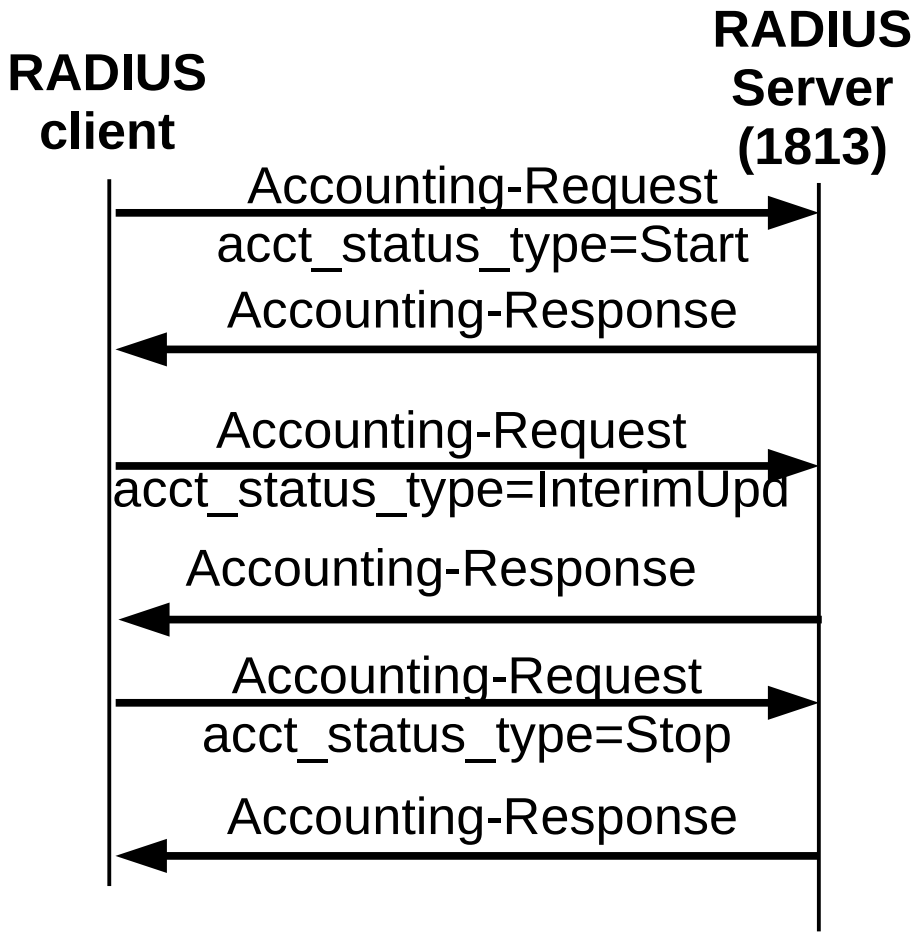
Honored when received in an Access-Accept message :

- **Acct-Interim-Interval** : the interval in second between interim updates
- **Tunnel-Private-Group-ID** : the vlan assigned
- **Tunnel-Type** : eg VLAN
- **Tunnel-Medium-Type** : eg 802 (which includes 802.11)
- **Filter-ID** : use %d.out and %d.in for interface output acl or interface input acl

# Radius acct exchanges(RFC2866)

Radius accounting is delivered on a different port(1813) than Authentication and Authorization, it can also usually be delivered on another host.
Because radius uses UDP, it needs some error control :

- While UDP is often used without checksum control, radius requires it
- For every packet transmitted it requires a reply : in a TCP situation the accounting records would not require any reply, but because of UDP they do and the NAS continues to transmit them till it gets a reply (in radius parlance a 'Response')



# Radius accounting exchanges

- Transmitted to the  radius-acct  port (default=1813) of the server
- Accounting-Request (Code 4):

| Accounting Packets | When |
|---|---|
| Acct-Status-Type 1 = Start | At network access authorization time |
| Acct-Status-Type 2 = Stop | At disconnect |
| Acct-Status-Type 3 = Interim-Update | When authenticator has news about client |
| Acct-Status-Type 7 = Accounting on | |
| Acct-Status-Type 8 = Accounting off | |

- Accounting-Response (Code 5)
- On Unix/Linux radiusd accounting daemon is a son of radiusd authorization daemon

# Radius **Interim-Update**(RFC2869)

- Its  an accounting packet.
- It was in the past called  **Alive**
- Introduced for  ISPs and telecoms to have intermediate steps in the accounting to avoid that long sessions could encounter NAS reboots and therefore remain without an accounting-stop record needed for billing
- When the client sends an Access-Request the server answer with a requested Interim-Update interval (often 600 secs)
- It was renamed to **Interim-Update** because now it is also used to send new info the NAS collects about the client

# MAB (MAC Authentication Bypass)

Many devices (like printers) dont offer dot1x authentication.

There are 2 way outs : to  disable dot1x on the port or to provide a fall-back mechnism. The 2nd solution is a bit more safe.

Because of this many  NAS vendors provide a fall back : if the device doesn't answer to dot1x requests then the mac (in very simple format only hex digits, no colons, spaces or dashes) is encapsulated as username and password in a Radius Access-Request and sent to the Radius server that if the devices are known to the server can reply with an Access-accept.

(In new IOSs as password is sent the MD5 of the mac)

In freeradius you simply specify the MACs in a file.

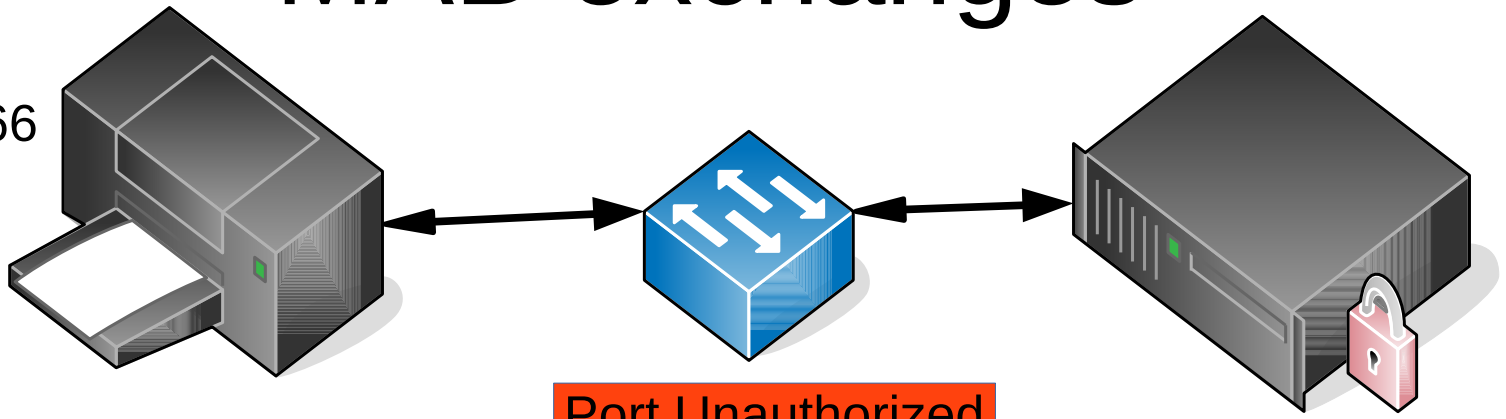
# MAB and radius Service-Type

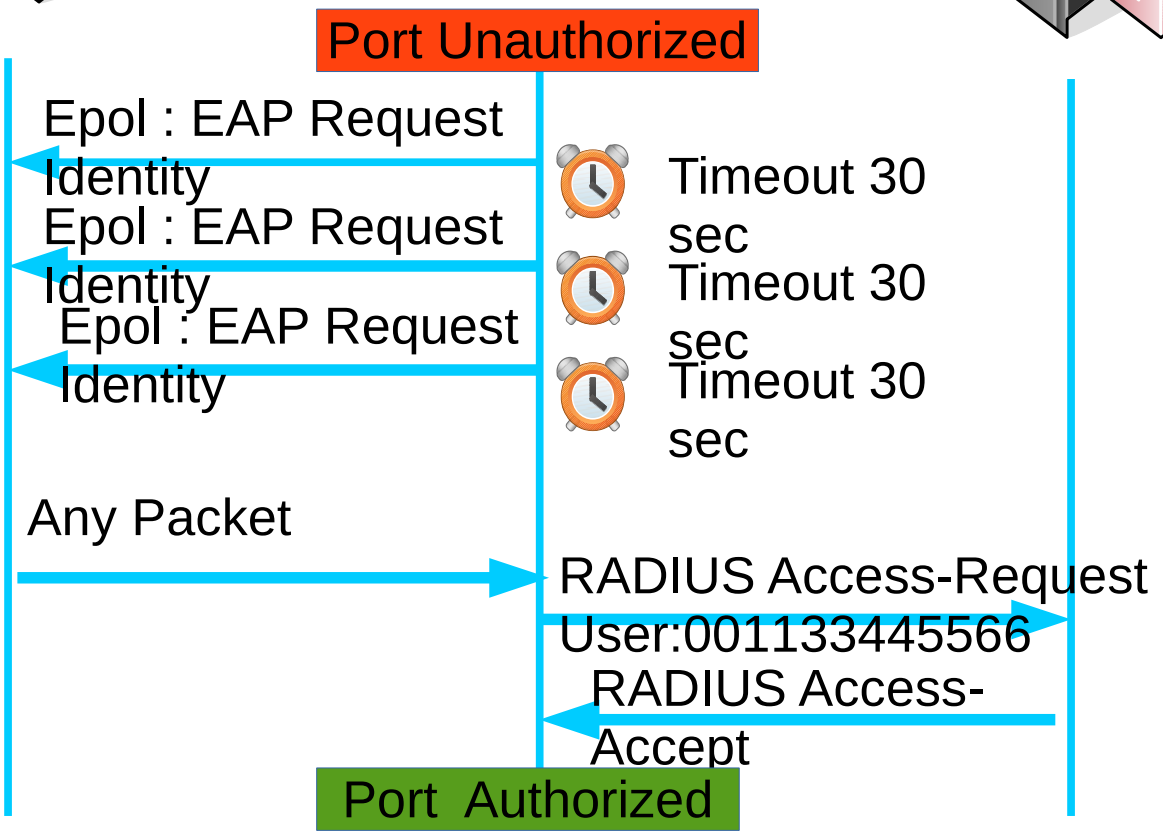
Cisco uses the service-type attribute to signal the type of authentication and sets :

- Service-Type=**Framed** (signals an 802.1X authentication)
- Service-Type=**Login** (signals WebAuth)
- Service-Type=**Call-Check** (signals MAB)

# MAB exchanges

Printer
MAC
001133445566





# MAB and radius Service-Type

Cisco uses the service-type attribute to signal the type of authentication and sets :

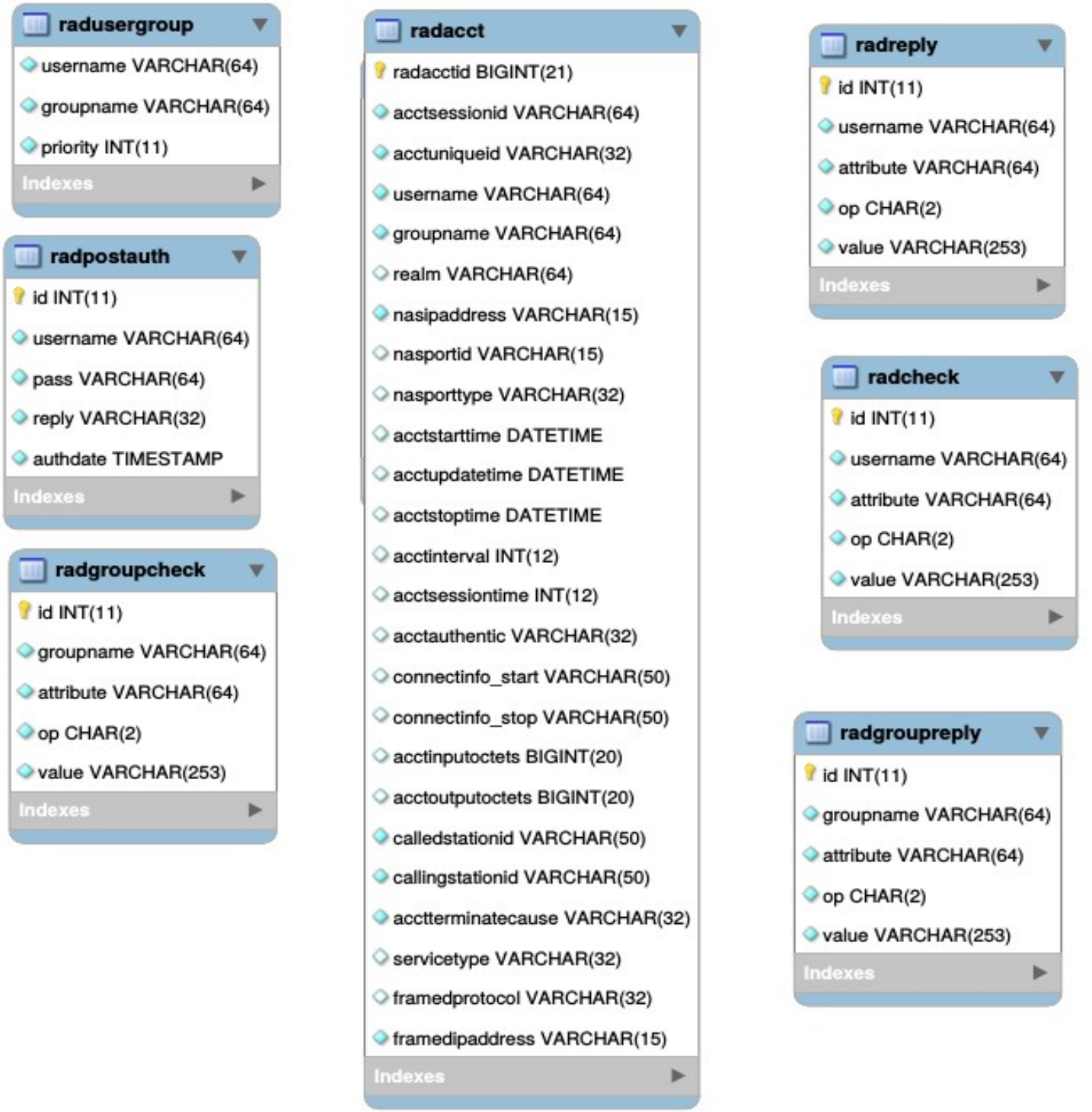
- Service-Type=**Framed** (signals an 802.1X authentication)
- Service-Type=**Login** (signals WebAuth)
- Service-Type=**Call-Check** (signals MAB)

To protect from user logging from captive portals with MACs of printers :)

# Radius with backend databases

- FreeRadius SQL module (rlm_sql) :
  - rlm_sql_mysql, rlm_sql_oracle, rlm_sql_postgresql
- FreeRadius LDAP module :
  - The default correspondences between ldap and freeradius attributes is in the ldap module
  - Radius attributes
    - **Tunnel-Type** from ldap **radiusTunnelType**
    - **Tunnel-Medium-Type** from ldap **radiusTunnelMediumType**
    - **Tunnel-Private-Id-Group** from ldap **radiusTunnelPrivategroupId** and is generally used to establish VLAN

# Freeradius db schema




2015-05-12

# Freeradius db query result

```
mysql> select username,nasipaddress,nasportid,acctstarttime ,framedipaddress from radacct;
+-------------+-----------------+-------------+--------------------------+----------------------+
| username | nasipaddress | nasportid  | acctstarttime             | framedipaddress  |
+-------------+-----------------+-------------+--------------------------+----------------------+
| inno          |169.254.1.10 | 6002        | 2015-04-30 11:12:34 | 169.254.1.180       |
| inno          | 169.254.1.10 | 6001        | 2015-04-30 11:46:15 | 169.254.1.180      |
| radeka       |169.254.1.10 | 6002        | 2015-04-30 11:46:15 | 169.254.1.181       |
| ritossa       | 169.254.1.10 | 6003        | 2015-04-30 11:46:16 | 169.254.1.182      |
| giunta        | 169.254.1.10 | 6004        | 2015-04-30 11:46:16 | 169.254.1.183      |
+-------------+-----------------+------------+--------------------------+----------------------+
5 rows in set (0.03 sec)
```

# freeRadius : normalizing mac addresses

MAC client addresses are presented in the most various forms:

001122334455 , 00-11-22-33-44-55,0011.2233.4455, 001122.334455,00:11:22:33:44:55

Can anyway, in a very simple way, be **normalized to a common format** : eg. 001122334455

(the same format used by MAB, but an RFC propends for the - format)

For freeradius in the file /etc/raddb/sites-enabled/default at the beginning of the section **authorize** add:

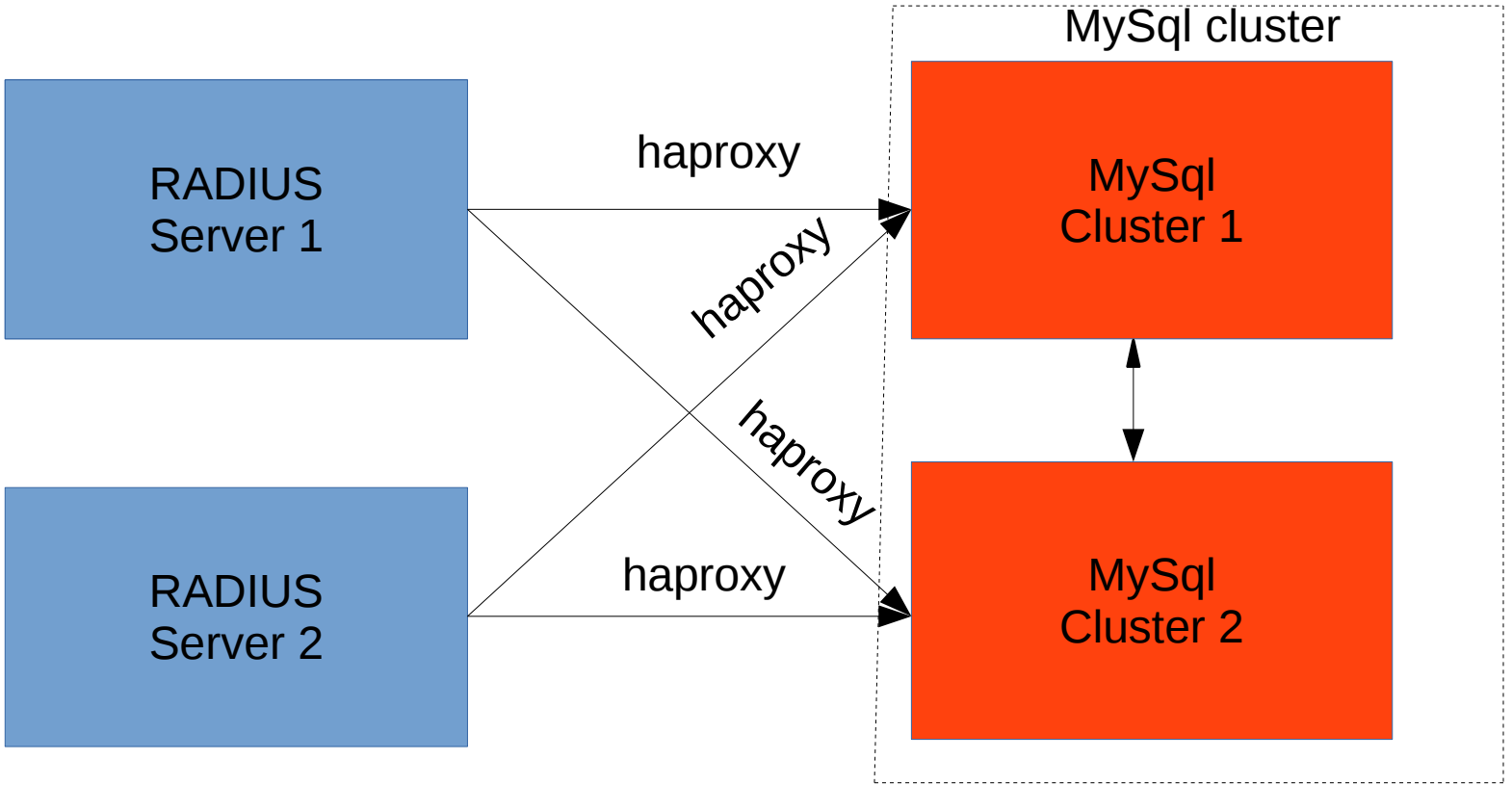
**rewrite_calling_station_id**

The same at the beginning of section **accounting. Calling-Station-ID** is the radius attribute used to transmit the mac of the client.

In the file /etc/raddb/policy.d/canonicalization you can modify the procedure **rewrite_calling_station_id** to follow your tastes.

**….. DEMO ….**

# Radius high availability



# Host configuration

Today computers to be able to communicate need at least these information :

- An IP address they can use as their own address
- An IP  netmask (because today computers are on subnets or CIDR)
- The IP address of a router (to reach computers outside their subnet)
- The IP address of a name server (to reach computers by name)

Historically these protocols were used for these purpose and each superseeded the previous :

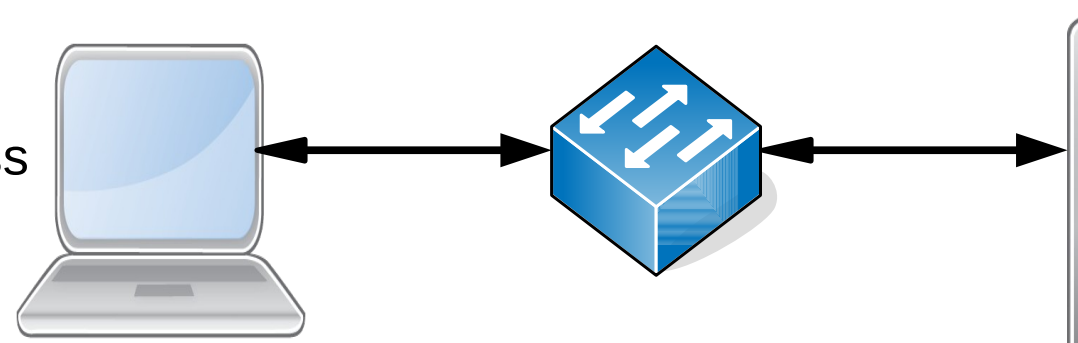
- RARP (Reverse ARP)
- BOOTP (Bootstrap protocol)
- DHCP (Dynamic Host Configuration Protocol)

# Dhcp

DHCP Client

DHCP Server

Client Port 68
Server port 67
Client IP address 0s
Server IP address 1s



Client Port 68
Server port 67
Client IP address 1s
Server IP address SIP

| 0s | 1s | 68 | 67 | Dhcp request |
|---|---|---|---|---|
| CIP | SIP | CP | SP | |

UDP From 0.0.0.0.68 to 255.255.255.255.67

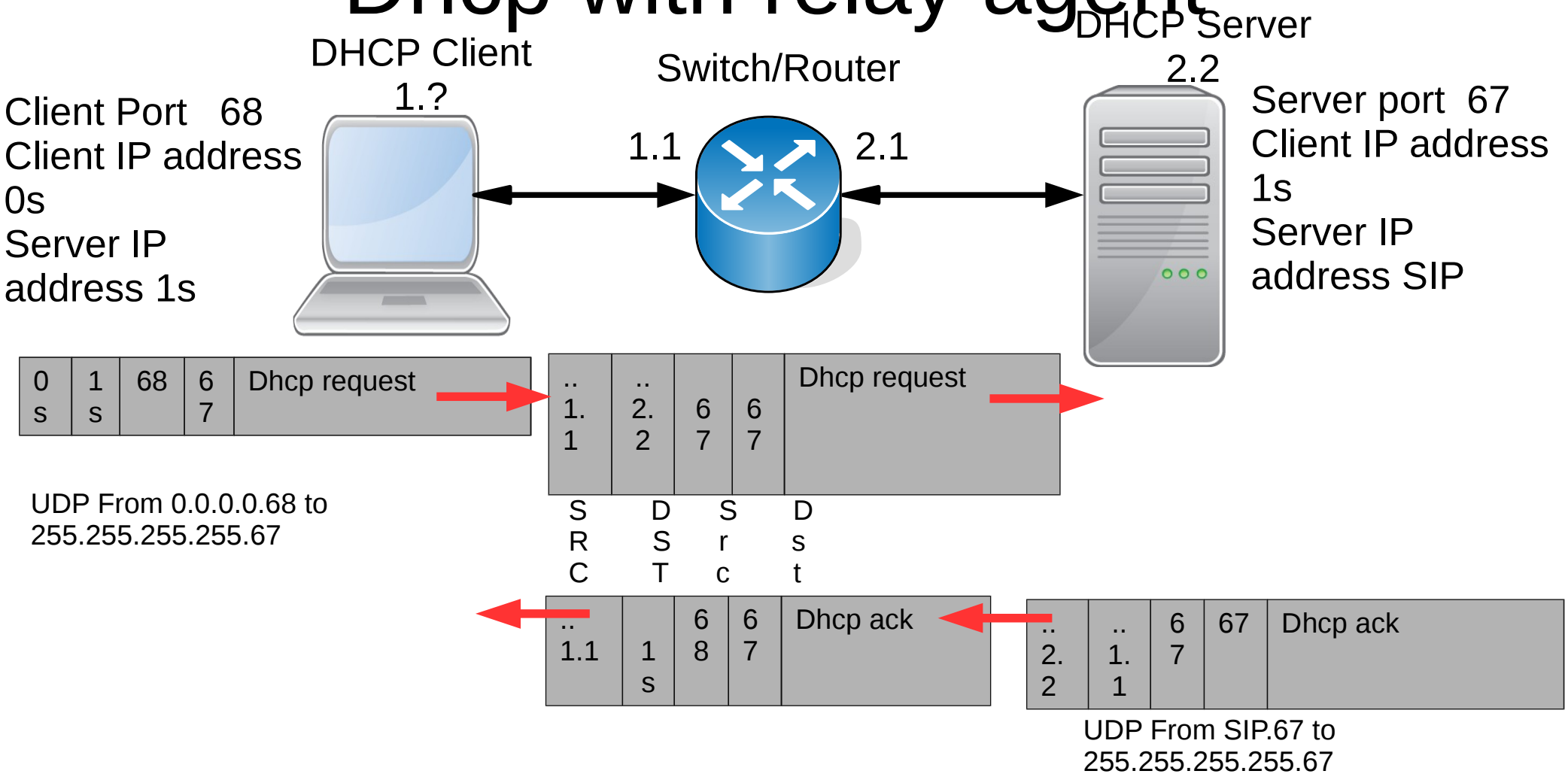
| SIP | 1s | 67 | 68 | Dhcp reply |
|---|---|---|---|---|
| SIP | CIP | SP | CP | |

UDP From SIP.67 to 255.255.255.255.68

# Relay Agents

- With the evolution of networks it was clear it is an overkill to keep some services (therefore hosts and ip-addresses) available on every subnet
- Not only : in many cases the discover of such services required to use broadcasting (not funny in large networks)
- Therefore it was introduced the concept of a Relay Agent : something that doesnt need to have an ip address but that can collect packets for some services and dispatch them to servers he knows and back.
- This mechanism is usually called on routers and switches **ip helper address**
- E.g**.** every broadcast packet for port 67 (bootps), is not broadcasted on the LAN, but is delivered as a unicast to a DHCP server, that the NAS knows . Other ports can be dispatched too.
- Cisco :
  - Ip helper-address *address*
  - Ip forward protocol udp 67

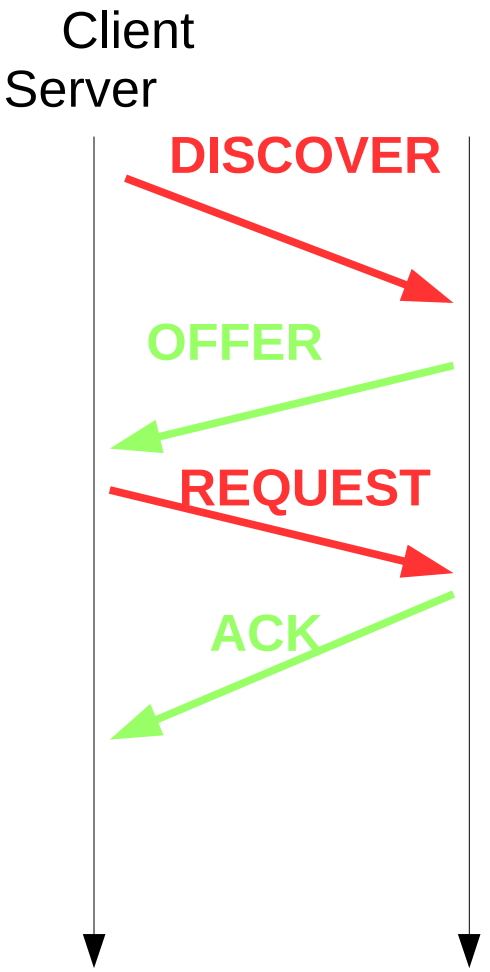
# Dhcp with relay agent




Client listen only on port 68, relay agent only on 67 and server also only on 67. Therefore answer to a relay is different that answer to a client because it has to be **from port 67 to port 67**.

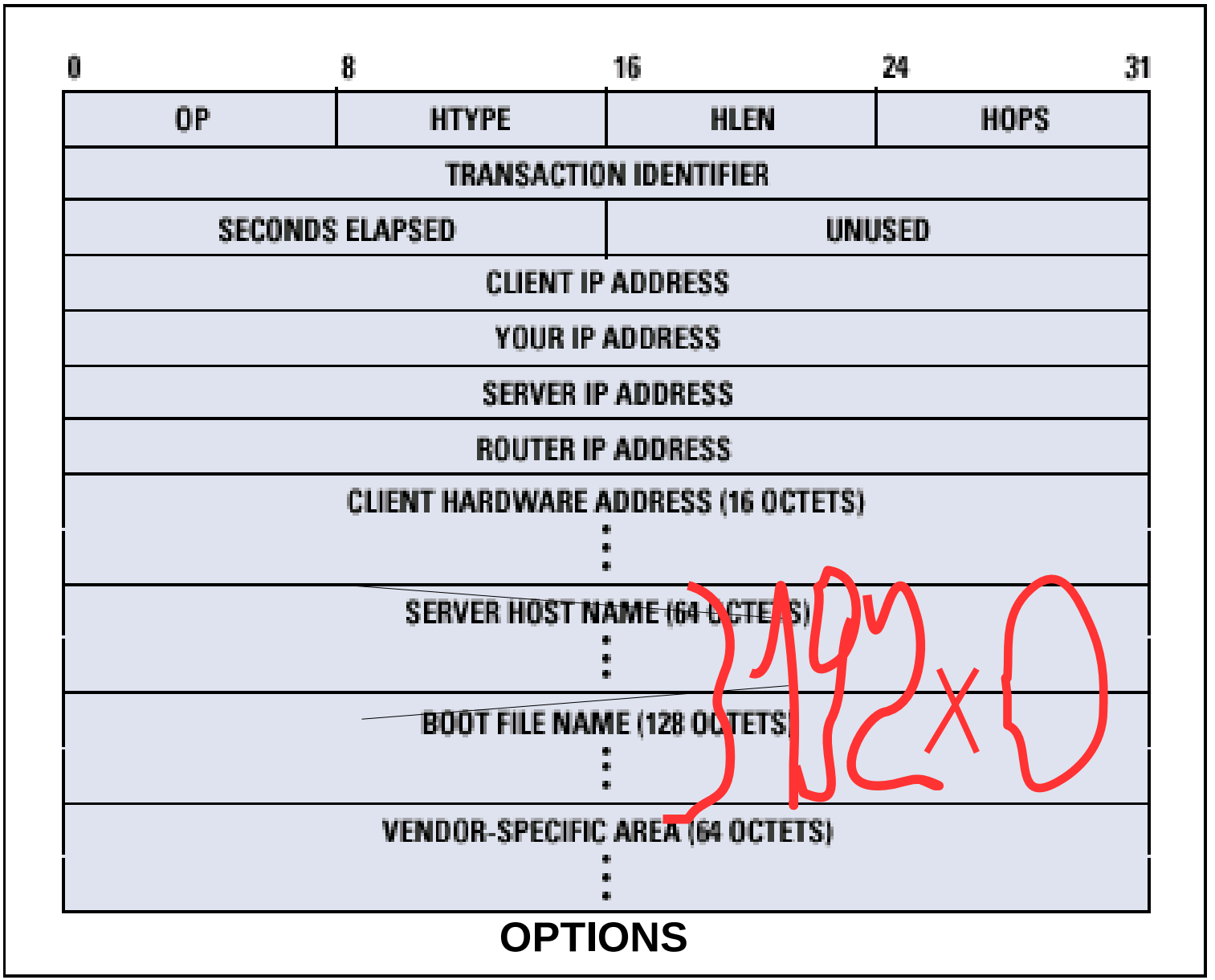
# DHCP protocol in 2 slides

- Normally uses UDP L ( tcp allowed) :
  - Client port 68 (bootpc)
  - Server port 67 (bootps)
- To remember the phases : remember **D.O.R.A**.



# DHCP packets

**DHCP request/ack :**



# DHCP

**REQUESTS**

- OP=1,Opt 53=1 **Discover**
- OP=1,Opt 53=3 **Request**
- OP=1,Opt 53=8 **Inform**

**ANSWERS**

- OP=2,Opt 53=2 **Offer**
- OP=2,Opt 53=5,6 **ACK or NAK**

**ACRONYMS**

- **Cip** client ip address
- **Chaddr** client mac address
- **Yip** your IP address(0.0.0.0)
- **Giaddr** Gateway IP address
- **Siaddr** Server IP address

# DHCP **LEASEQUERY**

**At the beginning was introduced to give the possibility to the NAS to recover the ongoing sessions trough queries to the DHCP server**

- A new dhcp request msg (RFC4388 Feb 2006 - RFC6188 ):
  - OP=1, Opt 53=10 Leasequery
- Possible answers:
  - OP=2,Opt 53=11 Lease unassigned
  - OP=2,Opt 53=12 Lease unknown
  - OP=2,Opt 53=13 Lease active
- The DHCP server collects many info about the clients, this info can now be queried by eg an access server to recover its database after a reboot, or by other nodes needing that information
- ISC DHCP implements Leaseque

**…….. DEMO ……..**

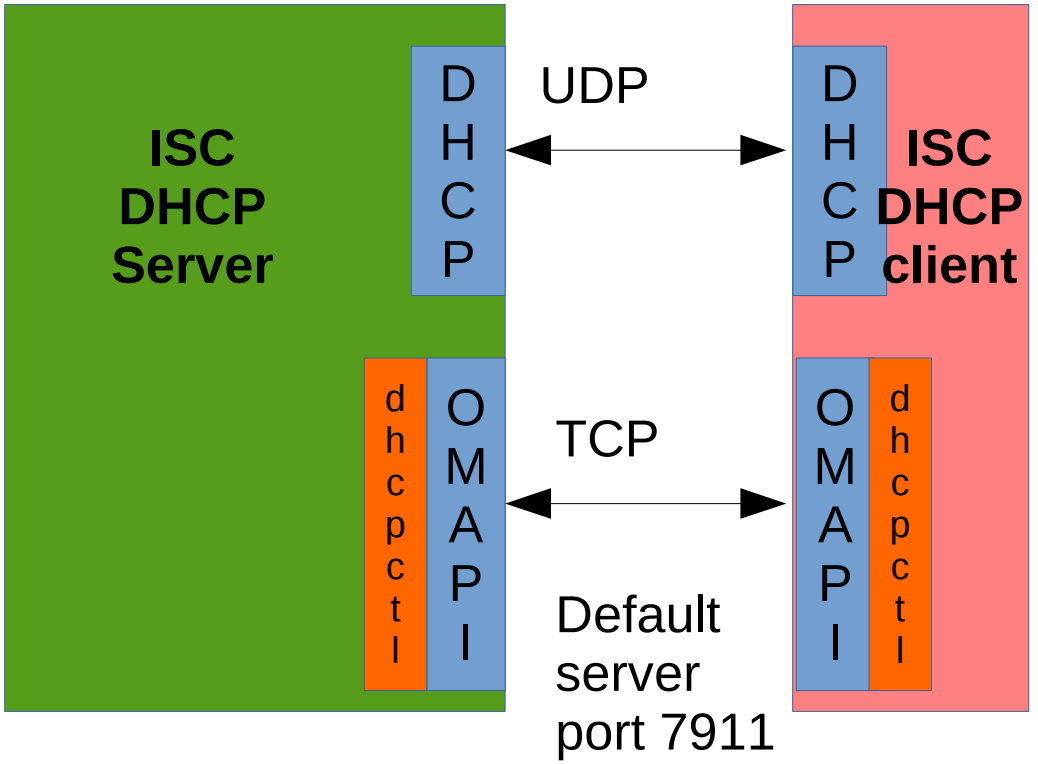
# ISC DHCP **OMAPI**

**OMAPI** (Object Mangement API)

A programming layer designed to control remote application and querying their state.

A thin API over it is called **dhcpctl :**

```
dhcpctl_initialiaze();

dhcpctl_connect (&connection, "127.0.0.1", 7911, 0);

dhcpctl_new_object (&lease, connection, "lease");

memset (&ipaddrstring, 0, sizeof ipaddrstring);

inet_pton(AF_INET, "10.0.0.101", &convaddr);

omapi_data_string_new (&ipaddrstring, 4, MDL);

memcpy(ipaddrstring->value, &convaddr.s_addr, 4);

dhcpctl_set_value (lease, ipaddrstring, "ip-address");

dhcpctl_open_object (lease, connection, 0);

dhcpctl_wait_for_completion (lease, &waitstatus);
```

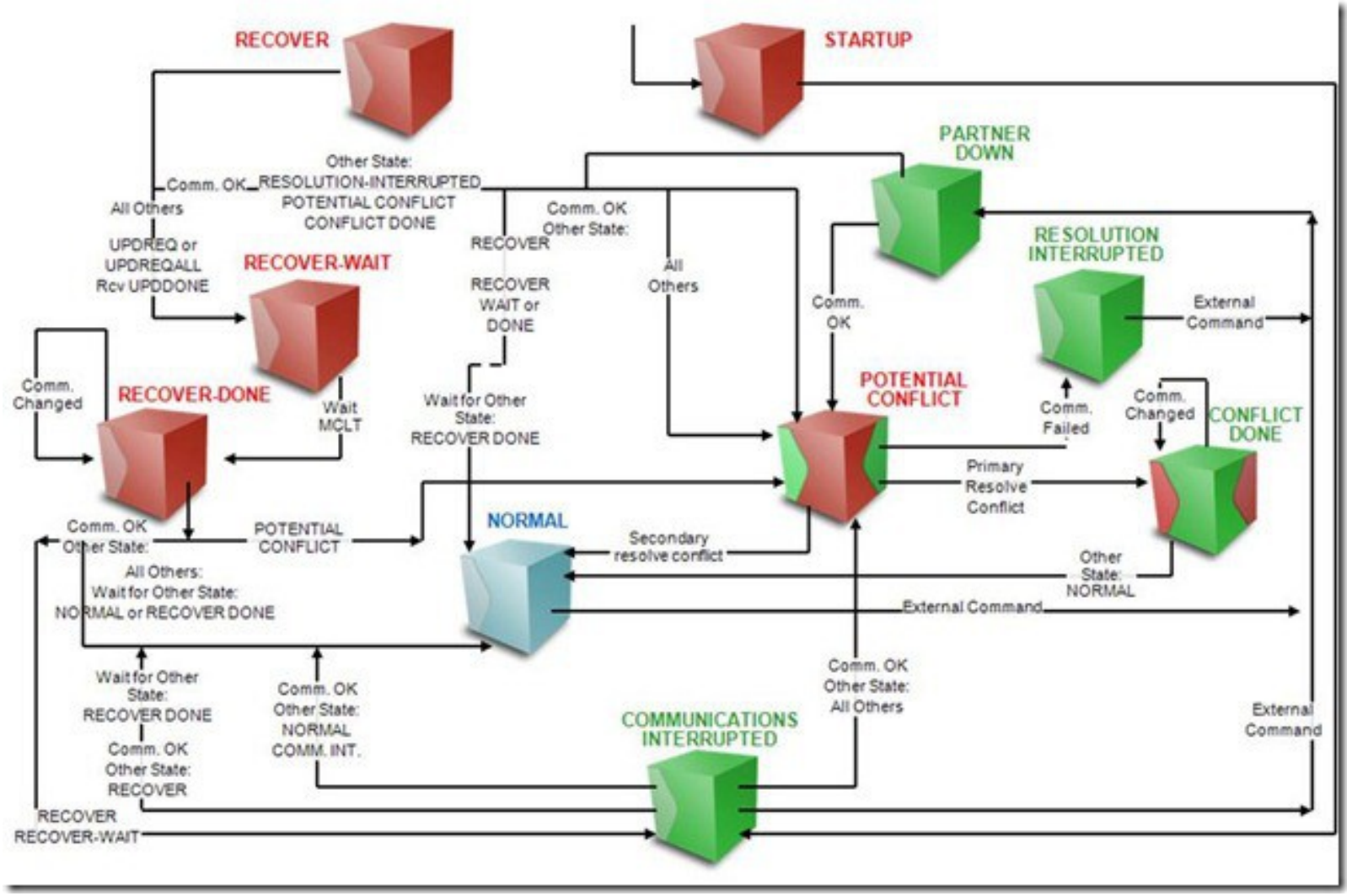



**….. DEMO ….**

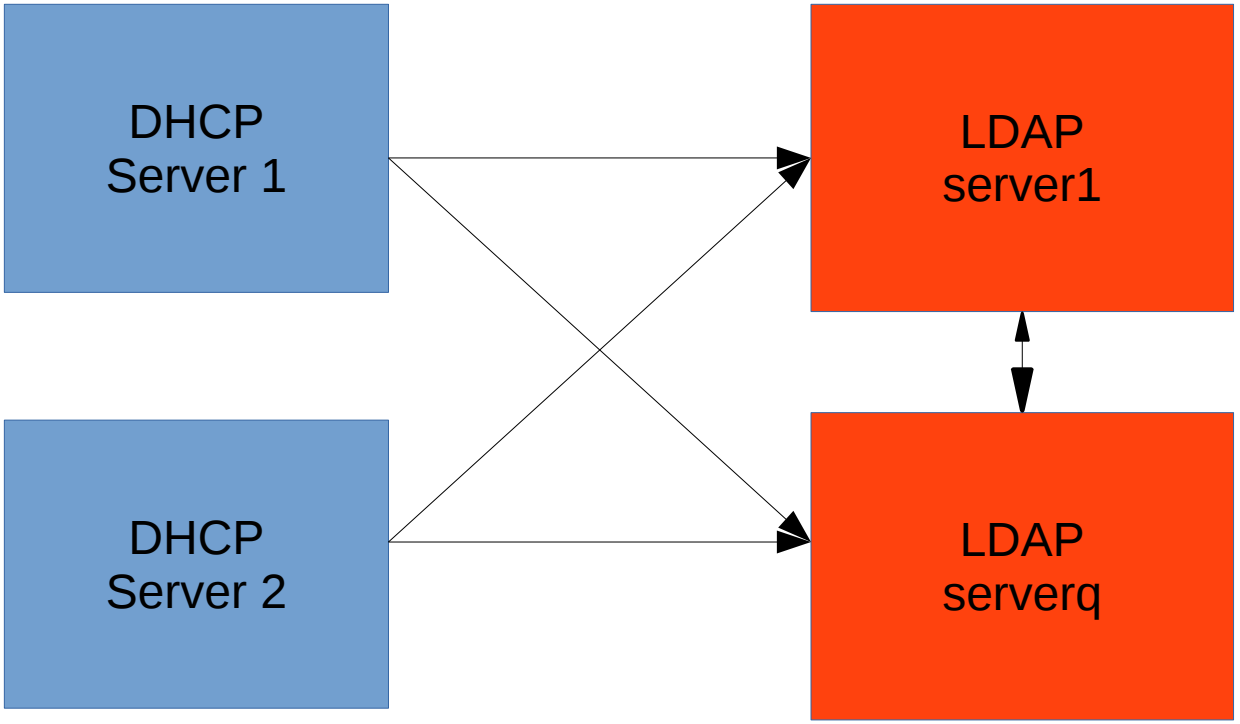
# ISC DHCP failover

- From 4.3 isc dhcp supports failover as per rfc *draft-ietf-dhc-failover-12.txt :* the protocol went from a 13 pages document for v 1 to a 144 pages for v 12
- Failover protocol allows 2 server to share a common address pool of which about ½ will be assigned to each (in case of failure of 1 server the other will continue to use its ½ of the pool free)
- If you set the server in state *partner-down* then it will try to get back also the ½ pool assigned to the other (you can so this using *omshell*)

# Mess of states in draft 12 dhcp failover



# Dhcp high availability



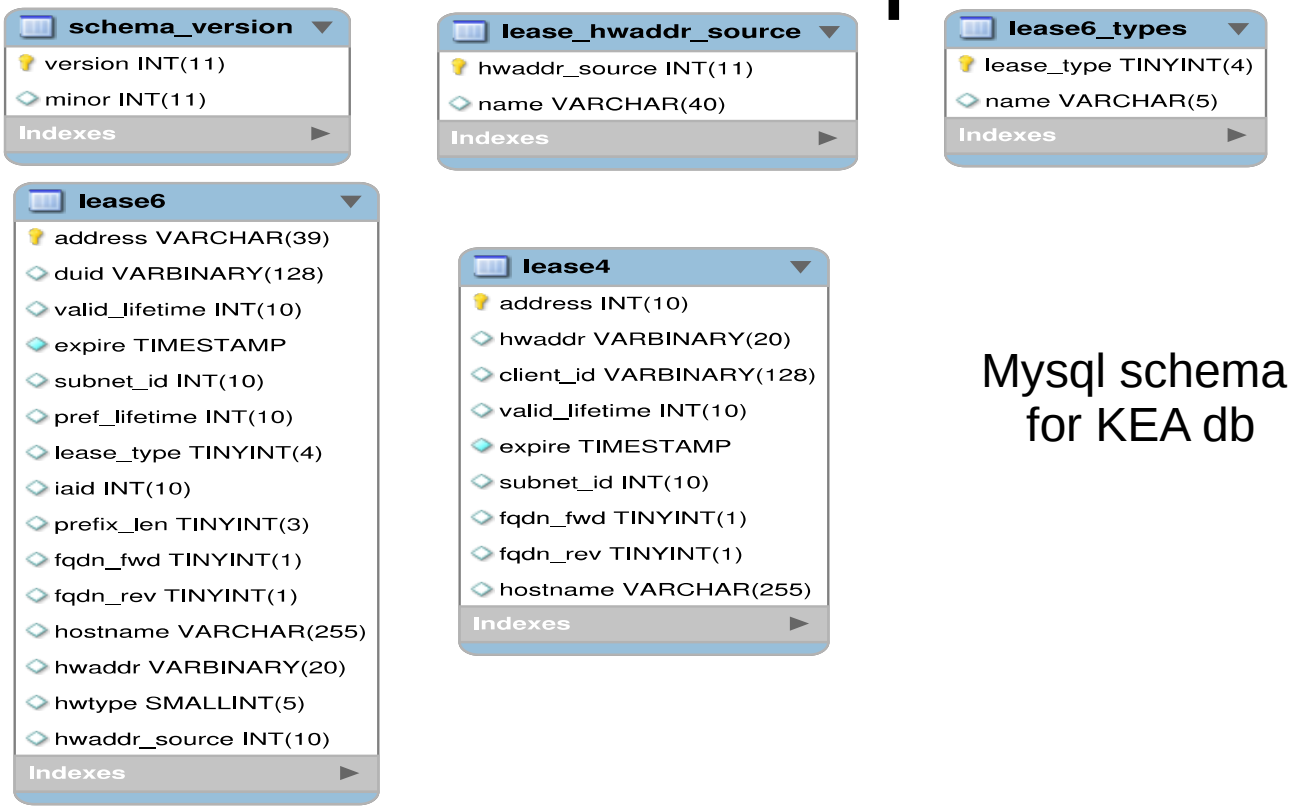
# ISC KEA

Is a new (started 2010) DHCP server under development in C++.

It uses a mysql/postgres database backend for leases and this is a great advancement versus the standard ISC DHCP server.

The new configuration file uses JSON syntax.

# ISC KEA dhcp server

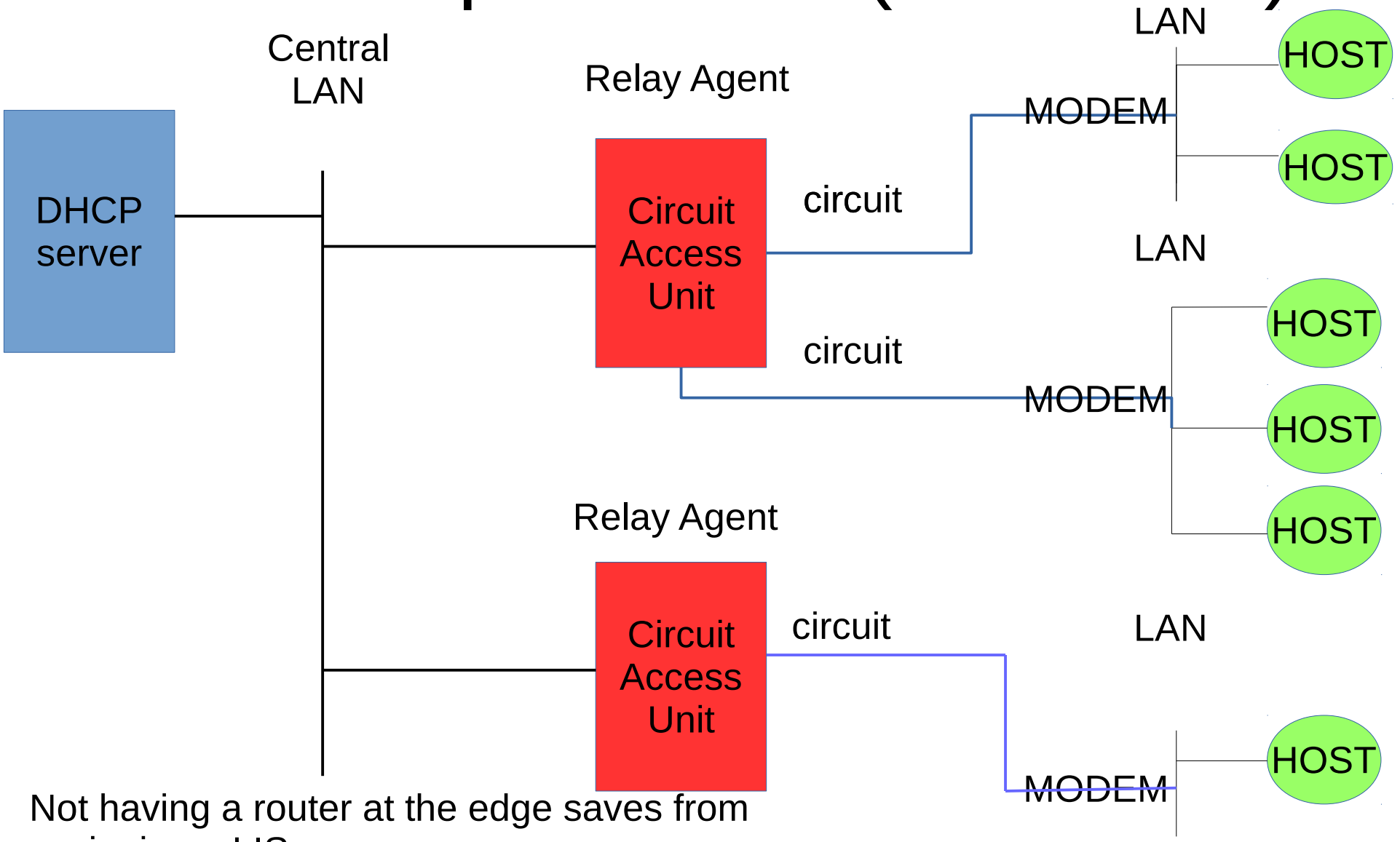



Mysql schema
for KEA db

# DHCP spoofing

- Initial DHCP discover packets are broadcasted at layer 2 over the broadcast domain ( usually LAN or VLAN), because the client at that point knows nothing about network configuration
- Whoever is on that subnet can then listen to it and try to reply like if it is a legitimate server
- They can then kidnap the client assigning a network and ip address  and themselves as default gateway, performing a **man in the middle attack**

# DHCP option 82/1 (RFC3046)




Not having a router at the edge saves from assigning a LIS
(Logical Internet Subnet) for each premise and is quite less expensive too ..

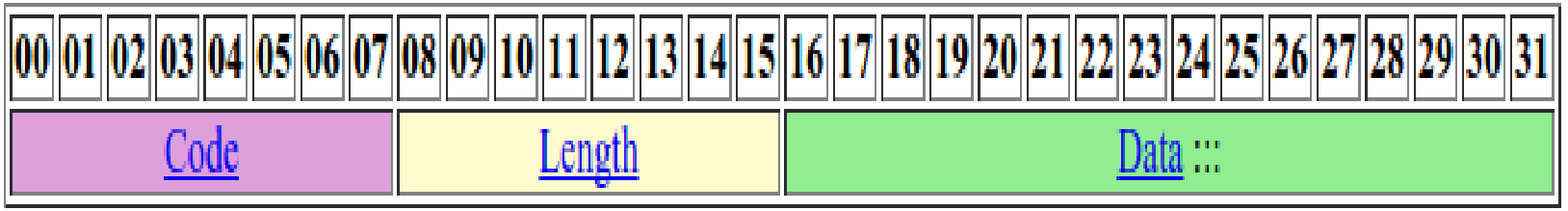
# DHCP option Relay agent information aka option 82 (RFC3046)

With the evolution of the access networks it became common to have an high speed modem connect in a LAN multiple customer premises and multiple hosts per site. It is useful in such a situation to use DHCP to configure these hosts. This public use of DHCP poses anyway some security and scaling problems.

- Therefore a special DHCP option was introduced in **RFC3046 (Jan 2001)** according to which the **relay agent** that passes dhcp packets to servers inserts in the dhcp packet an option with other information (that allows to identify the path from where it came). The server eventually uses that information to use specific pools of addresses and sends it back in its replies. This option is then removed again by the **relay agent** when it passes the replies to the clients. The clients are not supposed to ever receive or send this **dhcp option 82**.

# Option 82/1 (RFC3046)

**BOOTP/DHCP option 82:**




**Code.** 8 bits. Always set to 82.
Option code.

**Length.** 8 bits.
Size of the option data in bytes.

**Data.** Variable length.
Contains the suboptions for this option.

# Option 82/3 (RFC3046)

The option 82 information is made up of a sequence of suboptions.

- **Agent Circuit ID** (suboption 1)—An ASCII string that identifies the interface on which a client DHCP packet is received.
- **Agent Remote ID** (suboption 2)—An ASCII string assigned by the relay agent that securely identifies the client.
- …..

Cisco commands:

Juniper commands :

- set dhcp relay agent sub-option circuit-id …....
- set dhcp relay agent sub-option remote-id …..

# DHCP snooping/1

Probably you know what means *to snoop* ..

Introduces some security for DHCP. The switch snoops all dhcp packets on client ports and checks them for being appropriate.

- Divides ports between *trusted* and *untrusted*
- Untrusted ports can only pass DHCP **discover/request/inform**
- Only trusted ports can pass other DHCP traffic: **offer/ack/nack**
- If a DHCP **offer** will be heard on an untrusted port the port will be shutdown

# DHCP snooping/2

- Device sends a **DHCP discover,** from an untrusted port, the switch forwards it to the server
- Servers sends a **DHCP offer** from a trusted port, the switch forwards it to the device trough the connecting port
- Device sends a **DHCP request**, from an untrusted port, switch forwards it to the server and puts a reservation inside its db for the ip/mac pair
- Server sends back a **DHCP ACK** from a trusted port, the switch forwards it to the device and inserts it in its **ip/mac database** the entry
- A **DHCP release** causes the switch to delete the entry in its db, the same an expiry of the lease

# DHCP snooping configuration/1

- Cisco:
  - Ip dhcp snooping # on all the switch
    - # after this all ports are now untrusted and dhcp server pkt not fw
  - Ip dhcp snooping vlan 51-55 #on vlan 51-55
  - Int gi3/47
    - Ip dhcp snooping **trust**
    - Ip dhcp snooping **limit rate** 100
  - show ip dhcp **snooping binding**

# DHCP snooping binding database

- Router# show ip dhcp snoop bind
- MacAddress IpAddress Lease(sec) Type VLAN Interface
- ----------- ---------- ---------- ---------- ---- ----------
- 00:01:00:01:00:05 1.1.1.2 49810 dhcp-snooping 512 GigabitEthernet1/1
- 00:01:00:01:00:02 1.1.1.3 49810 dhcp-snooping 512 GigabitEthernet1/1
- 00:01:00:01:00:04 1.1.1.4 49810 dhcp-snooping 1536 GigabitEthernet1/1
- 00:01:00:01:00:03 1.1.1.5 49810 dhcp-snooping 1024 GigabitEthernet1/1
- 00:01:00:01:00:01 1.1.1.6 49810 dhcp-snooping 1 GigabitEthernet1/1
- Router# clear ip dhcp snoop bind

# DHCP snooping and Radius

- Sound implementation of DHCP snooping demands that a **Radius Interim-Update** packet is sent to the Radius server as soon as new information is known about a client
- Therefore as soon as an ip address is bounded into the snooping database the Radius server is informed of that sending the ip address as  a **Framed-IP-Address** Attribute-Value-Pair
- This is inserted  in the freeradius utmp database and appears on radwho command output after the **Location field**

**…... DEMO …..**

# Dhcp snooping and interim-update for alcatel

On the left the radius attributes sent on an Interim-Update by alcatel switches when the switch comes to know the IP address of the client and dhcp snooping is enabled and 802.1x accounting is enabled too. Note the vendor specific attribute :

**Alcatel-Lucent-Client-IP-Addr**

versus the usual

**Framed-IP-address**

- User Name
- NAS-IP-Address
- NAS-Port
- Acct-Session
- Acct-Authentic (to be 1 -radius- for 802.1x users)
- Acct-Terminal-Cause (currently not supported)
- Alcatel-Lucent-Auth-Group (VlanId)
- Alcatel-Lucent-Slot-Port
- Alcatel-Lucent-Client-IP-Addr
- Alcatel-Lucent-Group-Desc (vlan name)

# Alcatel 802.1x parameters

Switch sends up to *max-req* EAP Request-Identity to the supplicant every *tx-period* seconds. The supplicant is polled during a *supp-timeout* period before giving up for a *quiet-period* seconds.

The polling of the supplicant is retried *retry* times.

- Aaa radius server
- Aaa authentication 802.1x rad1 rad2
- Vlan port 3/18 802.1x enable
- 802.1x 3/18 quiet-period 50 tx-period 25 supp-timeout 25
- 802.1x 3/18 max-req 3
- 802.1x 3/18 supp-polling retry 10
- 802.1x reauthentication
- 802.1x reauthentication re-authperiod 25

# ISC DHCP server opt 82

- Supports partially option 82, in particular doesnt support radius attributes
- I **patched a 4.3.2 isc dhcp** server in such a way that it supports them.
- It keeps the **radius attributes in the lease database**, so that it can restart with them and it replies with the attributes *to the NAS* on replies, to *leasequery clients* and to *omapi clients*

# Snooping and Option 82 using 802.1x

**RFC4014** Remote Authentication Dial-In User Service (**RADIUS**) Attributes Suboption for the Dynamic Host Configuration Protocol (**DHCP**) Feb 2005

A new suboption ( 7=radius-attributes) of dhcp option 82 is defined :

```
SubOpt   Len     RADIUS attributes
 code
 +-------+-----+------+------+------+------+--...-+------
 |  7    | N   | o1   | o2   | o3   | o4   |      | oN |
 +-------+-----+------+------+------+------+--...-+------
```

The radius attributes :

| # | Attribute |
|---|---|
| **1** | **User-Name** (RFC 2865 [3]) |
| 6 | Service-Type (RFC 2865) |
| 26 | Vendor-Specific (RFC 2865) |
| 27 | Session-Timeout (RFC 2865) |
| 88 | Framed-Pool (RFC 2869) |
| 100 | Framed-IPv6-Pool (RFC 3162 [7]) |

Can be inserted by the dhcp relay agent in an option 82, **suboption 7 (radius-attributes)**

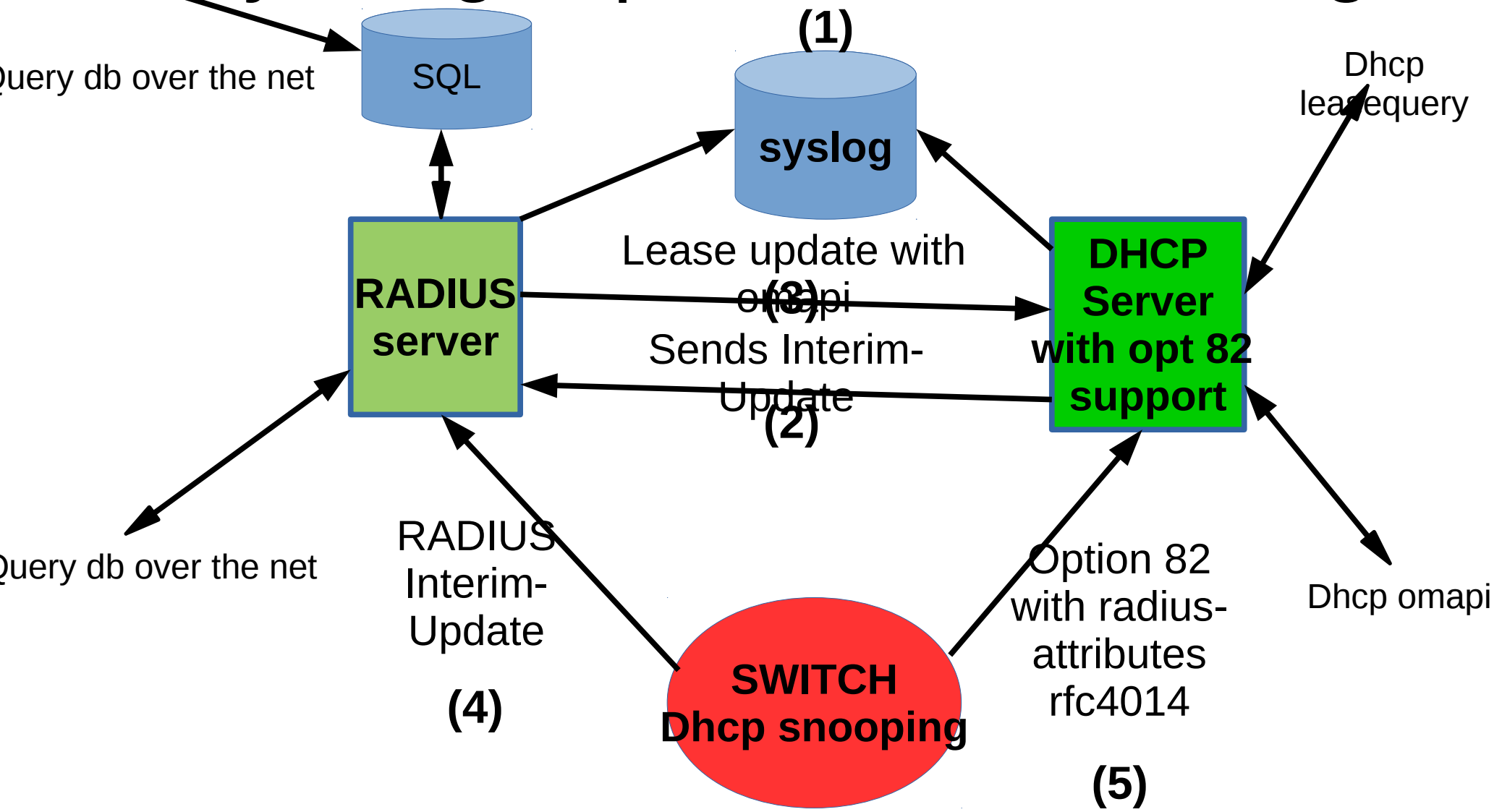
An option 82 capable dhcp server stores these attibutes in its leases and send them back in answers

**….. DEMO lease structure…..**
**….. DEMO leasequery…..**

# Option 82 suboption 7 (radius-attributes)

- Radius attributes are stored in the way they are stored in the radius packets(different from the way option or suboptions are stored in dhcp packets ) as AVP(Attribute Value Pair) :
  - **TLV (Type, Length, Value)** but here length comprises the 2 header bytes so that for example User-Name='inno' is packed as :
    - \0x01\0x06inno
- And the complete 7 suboption becomes :
  - \0x07\0x06\0x01\0x06inno
- All radius attributes should be stored in 1 suboption 7

# Ways to get ip/username bindings



# Lab setup for Hands on sessions