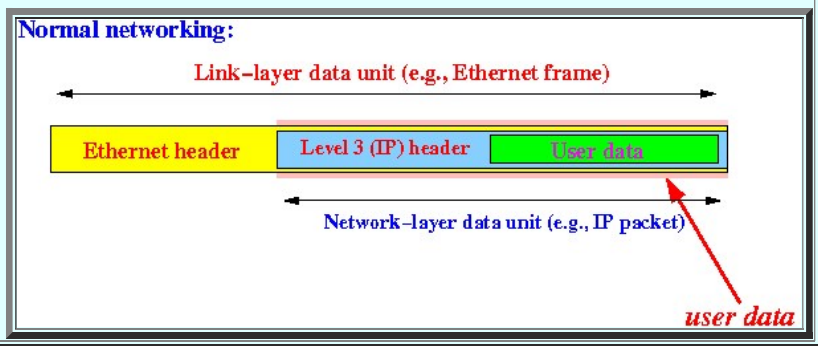
Intro to IP Tunneling

- **Normal network comminication**
  - **Normal network communication:**

    - In **normal** network communication (that we have **studied** so far):
      - **User data** is put **inside** an **level 3 (network layer) packet**
      - The **level 3 (network layer) packet** is sent inside a **physical network frame**

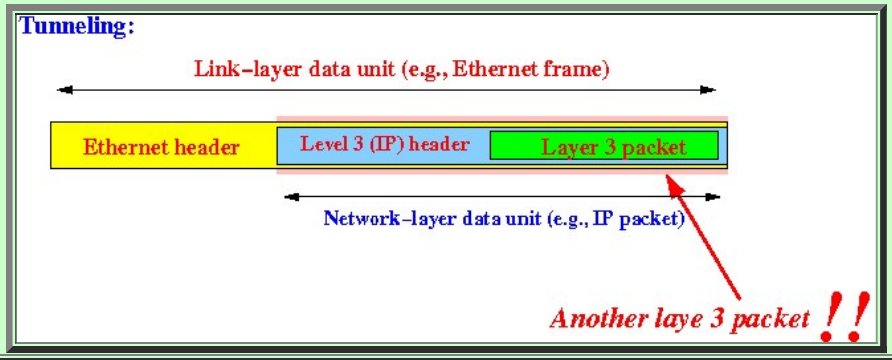
    Example:

- **Tunneling**
  - ***Tunneling*:**

    - **Tunneling** = an ***encapsulation* technique** where:
      - a ***network-level* data unit (packet)** is **transmitted** as **payload (= data)** inside ***another network-level* data unit (packet)**

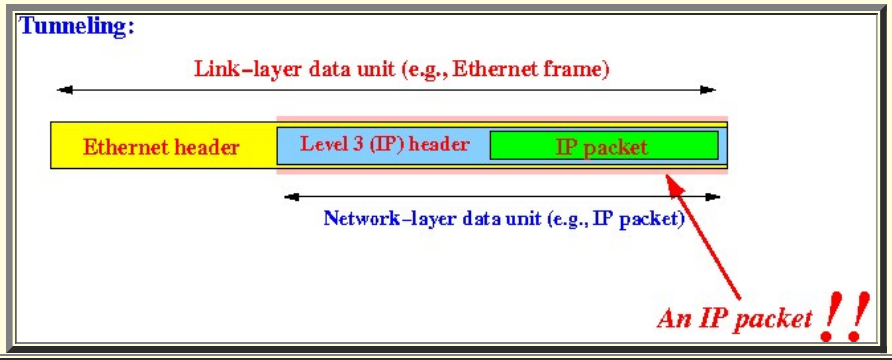
    Graphically explained:

- **IP tunneling**
  - **IP tunneling:**

    - **IP tunneling** = the ***encapsulation* technique** where one **IP packet** is **transmitted** as **payload (= data)** inside **another *network-level* packet**

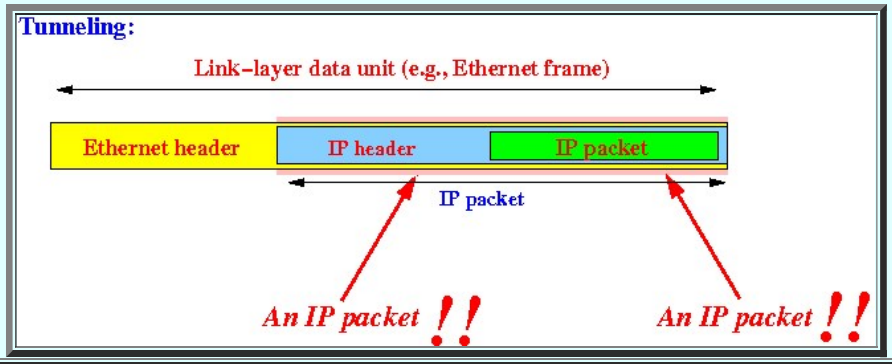
    Graphically explained:

- **IP-in-IP tunneling**
  - **IP-in-IP tunneling:**

    - **IP-in-IP tunneling** = the ***encapsulation* technique** where one **IP packet** is **transmitted** as **payload (= data)** inside ***another* IP packet**

      Graphically explained:



- ***Applications* that make use of *tunneling***
  - **Use of tunnels:**

    - The **create** a ***virtual* private network (VPN)**
      - This is the **most commonly used application** of **tunneling**

    - To **provide *novel* network services**

      Examples:

      - **IP multicast** (the **MBone** was **constructed** using **tunnels**)

    - To **connect** two or more ***non*-common networks** using a **common network**

      Example:

      - When **IPv6** was **first** introduced, the ***small* number** of **IPv6 networks** were connected through **IPv4 tunnels**

- **Dictatic notes**
  - The **IP tunneling feature** was **introduced** in ***in 1995*** to the **Internet Standard**

    - The **tunneling feature** is ***documented*** in the **Internet RFC 1853**: **click here**

  - **NOTE:**

    - I will use **IP-in-IP** to explain the **tunneling concept**

- **Encapsulation**
  - **Tunneling** is a **form** of:

    - ***encapsulation*** !!!!
      - The ***outer* IP packet** is used to **transfer (= "tunnel")** the ***inner* IP packet** to the **"processing location"** !!!

  - You have seen **encapsulation** before - it was used in **routers**: click here

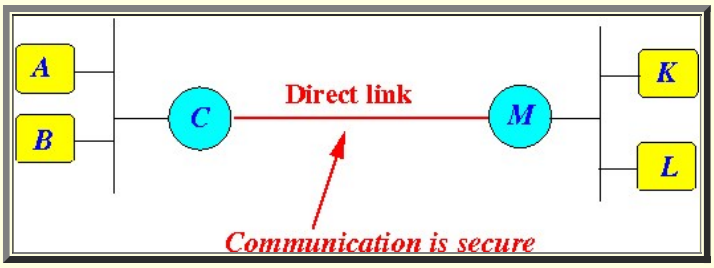
    - **Encapsulation** was used to:
      - Transmit a **(virtual) IP packet** inside a **(real) Physical (Ethenet) frame**

Operation of an IP-in-IP tunnel

- **Direct link and a virtual link**
  - **Direct link:**

    - **Direct link** = a **communication line** that connects 2 nodes ***directly***

      Graphically:

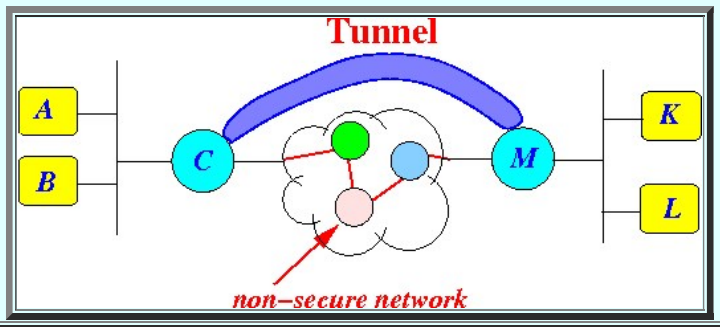
      

    - **Communication** through a **direct link** owned by the ***same*** **company** is very **secure** (from **eave dropping**)
      - To **eave drop**, the **intruder** will have to **tap** the **direct link**

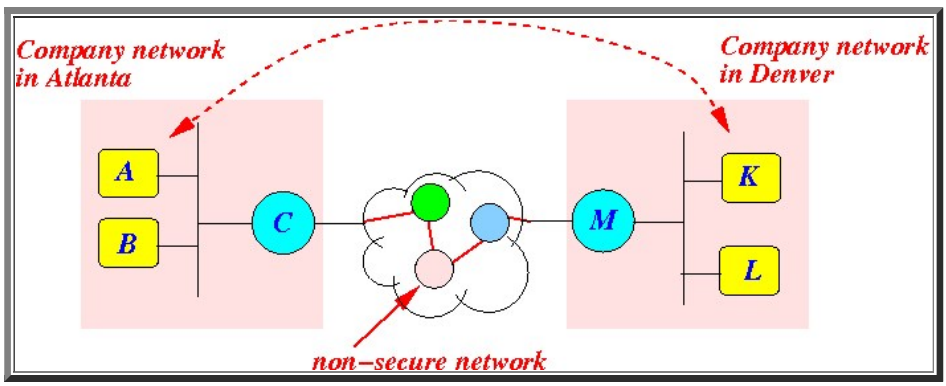
  - **Tunnel** ---- ***Virtual*** **direct link**:

    - **Tunnel** = a ***virtual*** **"direct" link**
      - The **messages** are **transmitted** over ***many*** **nodes**
        - The **nodes** can belong to ***different*** **owners**
      - The **messages** transmitted through the (**insecure**) **tunnel** is **made secure** through **encryption technqiues**

      Graphically:

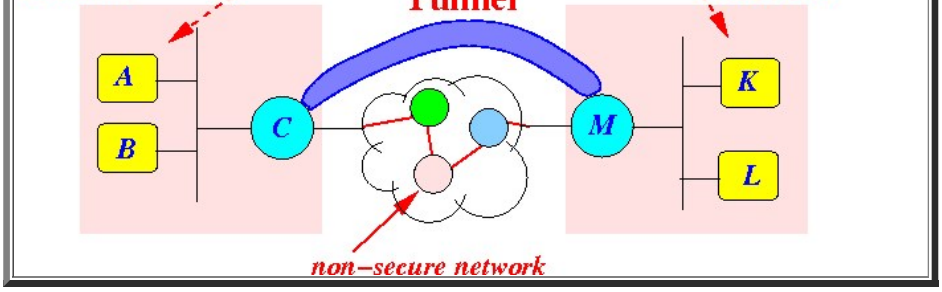
      

- ***Operation*** **of a secure IP-in-IP tunnel**
  - **Suppose** a **company** has **offices** in **2 locations**:

  - **Suppose** we want to **setup** a ***secure*** **tunnel** between the **company networks**:



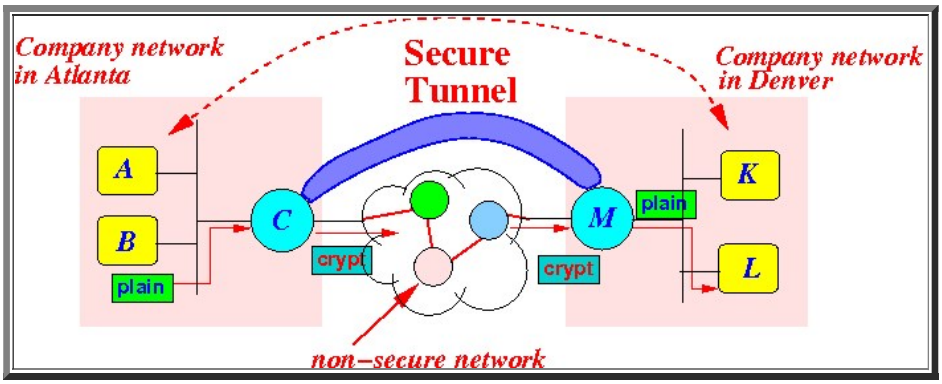
In other words:

- **IP packets** sent on a (any) **company network** can be **plain (= not-secured)**
- **IP packets** sent from **C ⇒ M** must be ***secured*** (= ***encrypted***

  I.e.:

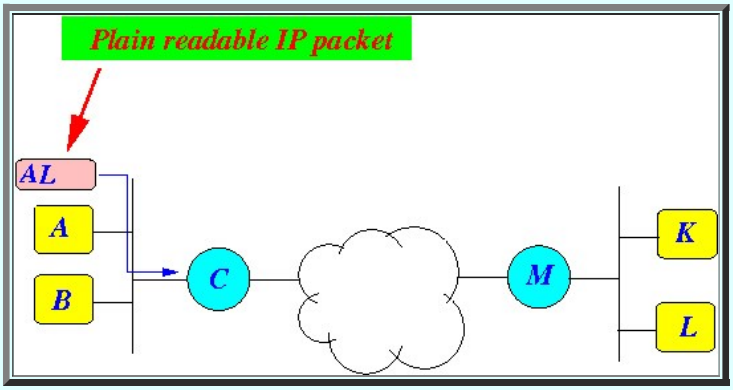
  - The **content** of the **IP packets** sent from **C ⇒ M** must **not** be ***plain***

Graphically:



- **Packet transmission over a secure IP-in-IP tunnel**
  - **How** to **realize** the **secure transmission** between **2 networks**:

1. Host **A** sends a ***(plain)* IP packet** to its **router C**:

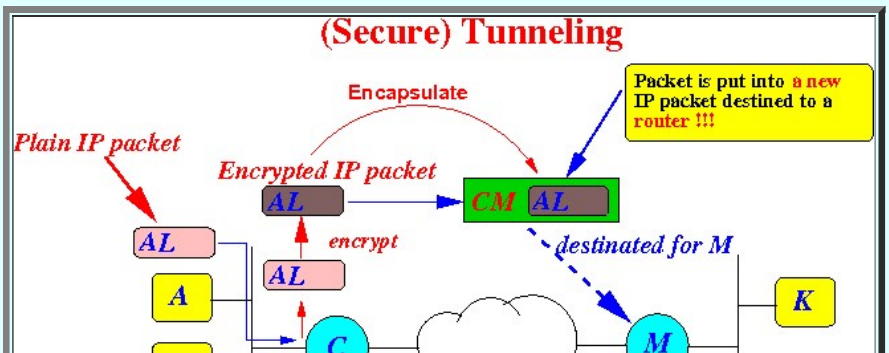


---

2. The **router C** processes the **(plain) IP packet** as **follows**:

   1. **Router C encrypts** the **(plain) IP packet**
   2. **Router C** then transmits the ***encrypted* IP packet** inside a ***new* IP packet** to **destination router *M***

   Graphically:



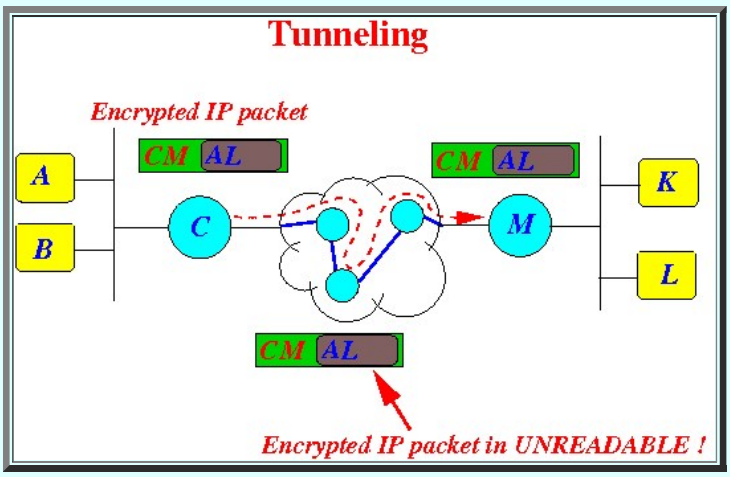
Postponed discussion:

- We will **discuss** ***how*** the **router C** can know **when** to **perform** ***encapsulate*** later
  - The **discussion** is given **here**: **click here**

  (Obviously, **router C** should not do this ***all*** **the time** !!!)

---

3. The ***encapsulated*** **IP packet** is then **routed** using **IP forwarding** to the **destination** ***M***:

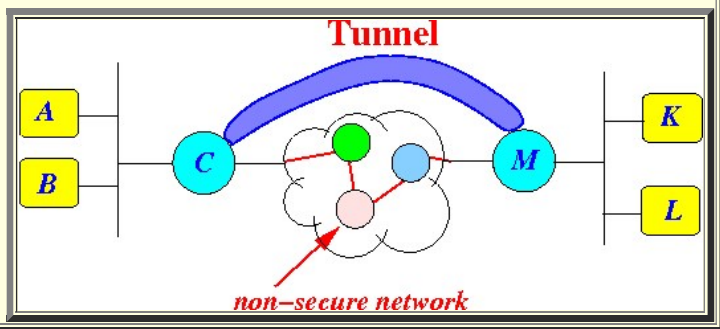


Notice that:

- ***Nodes*** can **forward** the **IP packet** to the **destination** ***M*** using the ***outer*** **IP packet**:
  - **Destination IP address** of **outer packet** is **router M** !!!

- **But** the **nodes** can ***not*** **decrypt (= decode and read)** the ***encrypted*** **inner IP packet** !!!

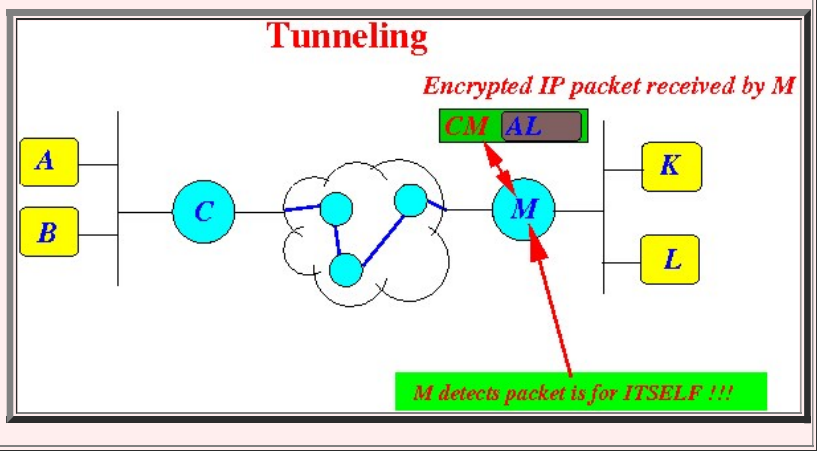
  So it ***feels*** like the **packet transmission** is on a ***direct*** **link**:



---

4. When the **destination router** ***M*** receives the **(encapsulated) IP packet**:

- **Router** ***M*** detects that the **IP packet** contains ***its own*** **IP address** ***M*** !!!



Note:

- This is ***not*** **normal**:
  - **IP packets** (so far in the **course**) contains **destination addresses** of **host (= computers)**
  - **IP packets** are **not** transmitted to ***routers*** !!!

Creative usage:

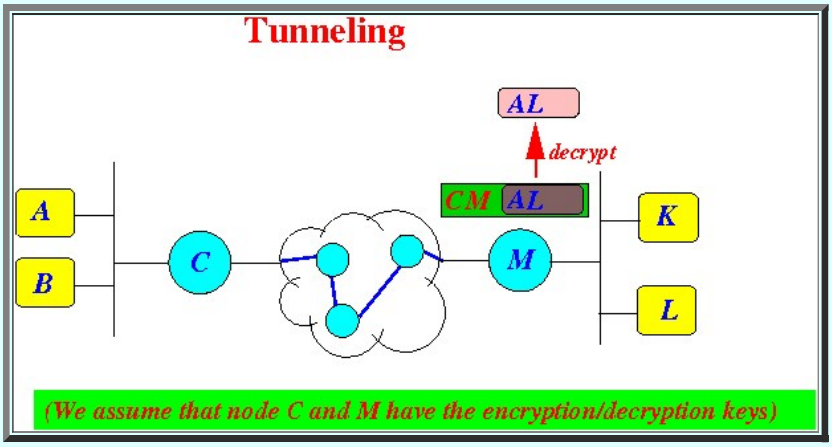
- When **destination IP address** is **equal** to a **router IP address**:
  - The **router** will perform a ***de-capsulation*** **operation** !!!

In **this example**:

- The **router *M*** will first **decrypt** the **inner IP packet** and
- Then the **router *M*** will **forward** the **de-crypted packet**

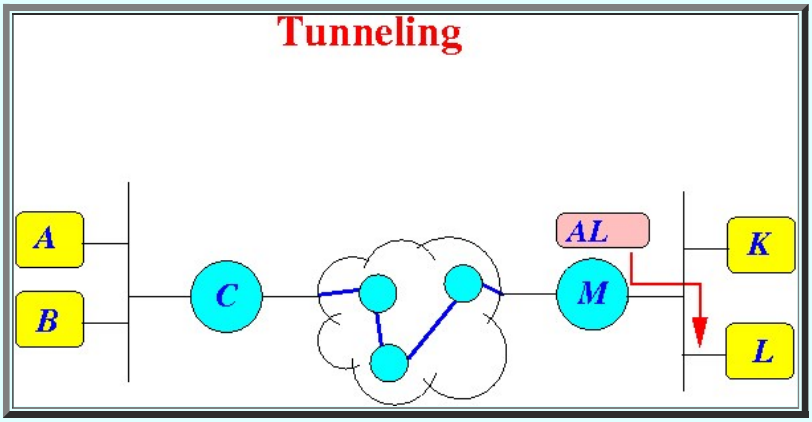
---

5. **Router *M*** decrypts the **inner IP packet**:



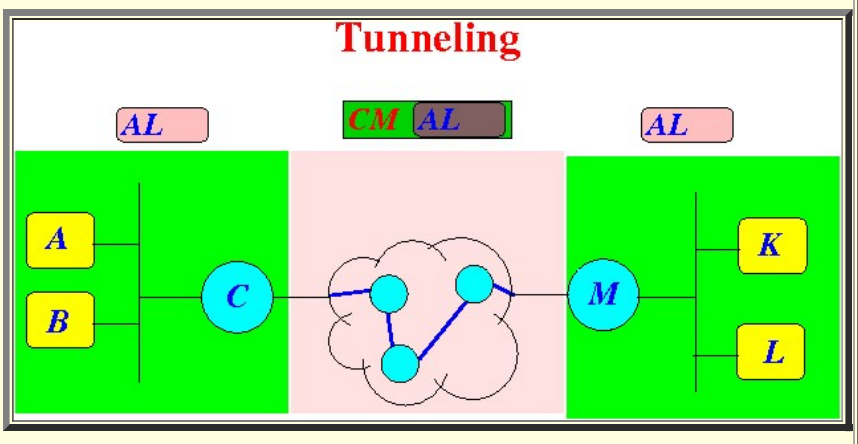
---

6. And **finally**, the **router *M*** forwards the ***decrypted*** **IP packet** to the **final destination *L***:

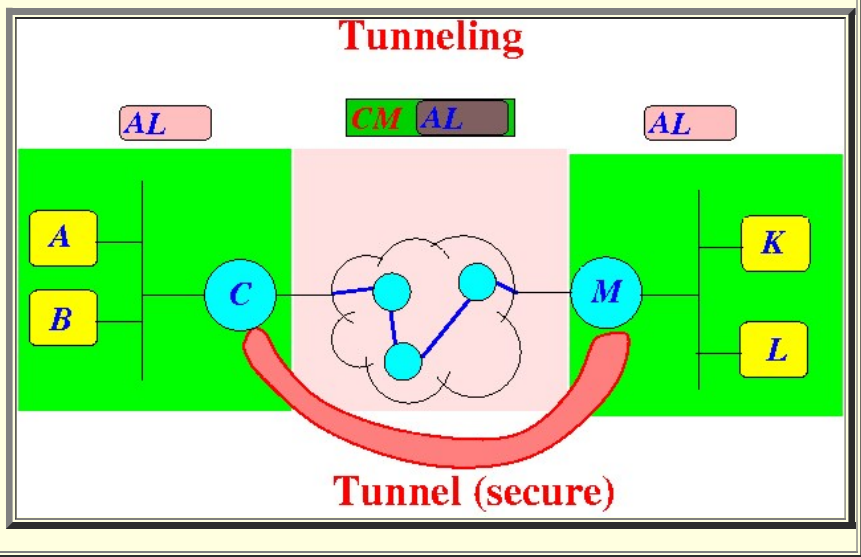


- **Summary:**

- The following **diagram** shows the ***regions*** where the **IP packet** are ***readable/unreadable***:



- It **feels** like there is a ***secure*** **tunnel** between ***C*** and ***M***:



Implementing the *encapsulation* procedure

- **Real and virtual interface**
  - **Real network interface**:

    - **Real network interface** = a **network card** that a **node** uses to **tranmsmit** ***physical*** **frames**

      Example:

      - **Ethernet** interface

  - **Virtual network interface**:

    - ***Virtual*** **network interface** = an **abstract** representation (= **object**) of a computer **network interface**
    - A ***Virtual*** **network interface** can be **mapped**:

      - To a ***real*** **network interface** **or**
      - To a **IP destination** (this is the **more common** usage)

- **Layer 2 forwarding table with virtual interfaces**
  - Recall the **format** of the ***(Layer 2) Forwarding Table***

```
Layer 2 forwarding table

+---------------+--------------------+
| Dest NetID    | Port (Interface)   |
+---------------+--------------------+
| IP netw 1     |        #1          |
+---------------+--------------------+
| .....         |  .....             |
+---------------+--------------------+
```

  - **Layer 2 forwarding table** with ***virtual*** **interfaces**:

```
Layer 2 forwarding table

+---------------+----------------------+
| Dest NetID    | Port (Interface)     |
+---------------+----------------------+
| IP netw 1     |        #1            |  <---- Real interface
+---------------+----------------------+
| IP netw 2     |  Destination IP addr |  <---- Virtual interface
+---------------+----------------------+
```

- **Packet processing prior to encapsulation**
  - **Note:**

    - It is **clearly** **stupid** to:

      - **Transmit** an **IP packet** using ***another*** **IP packet**

    - The ***novelty*** of **transmit** an ***inner*** **IP packet** using an ***outer*** **IP packet** is the **fact** that:

      - The ***inner*** **IP packet** is ***transformed*** in some way.

  - **Processing prior** to **encapsulation**:

    - The **virtual interface** **often** has an **associated** ***special*** **(packet) processing handler** to provide ***optional*** **processing**
    - The **handler routine** can be **part** of the **virtual interface** specification:

```
Layer 2 forwarding table

+---------------+--------------------------------+
```

Dest NetID	Port (Interface)
IP netw 1	#1
IP netw 2	Destination IP addr, handler

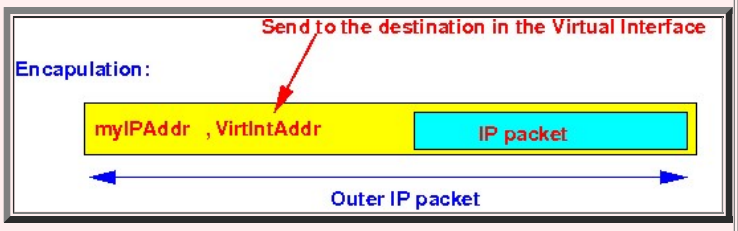
- **How to use a Layer 2 forwarding table with virtual interfaces**

  Processing on each **type** of **interfaces**:

  - **Forwarding** a **packet** on a ***real* (= physical) network interface**
    - The **packet** is **transmitted** in a ***physical* network frame**

      E.g.: in an **Ethernet frame**

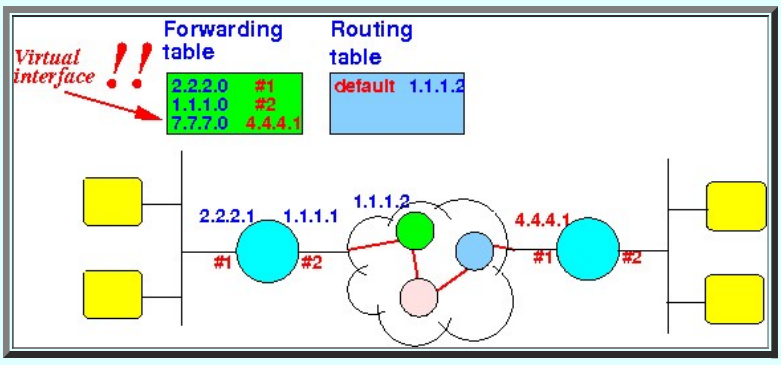
  - **Forwarding** a **packet** on a ***virtual* network interface**
    - The **IP packet** is ***optionally* pre-processed**
      - **Example:** ***encrypted***
    - Then the **packet** is **encapsulated**:

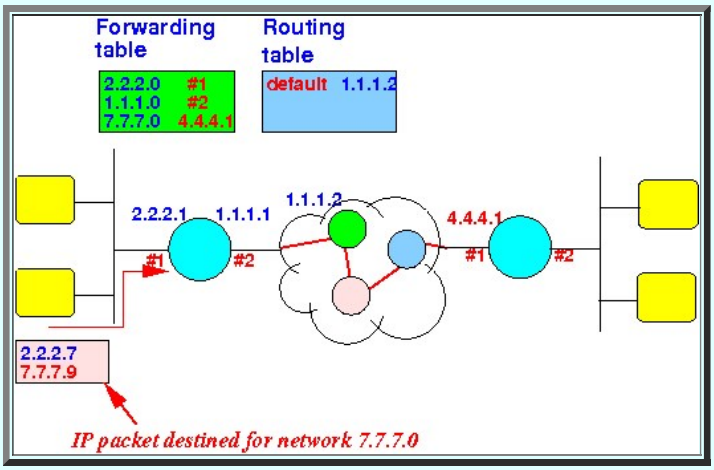
    - The ***outer* IP packet** is then **forwarded** (using the **forwarding/routing tables**)

- **Example:**

  - **Suppose** a **router** has the following ***virtual* interface entry**:

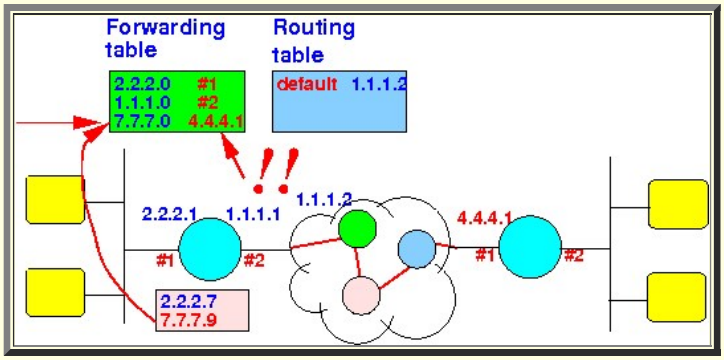
    

  - Then an **IP packet** destined for the **IP network** `7.7.7.0`:

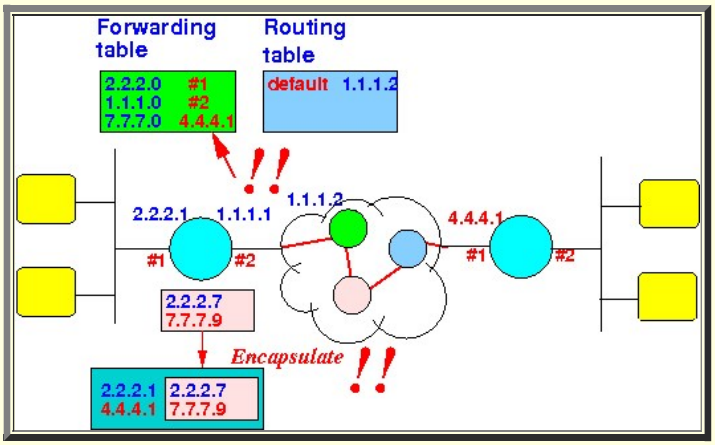
    

will be **forwarded** as **follows**:

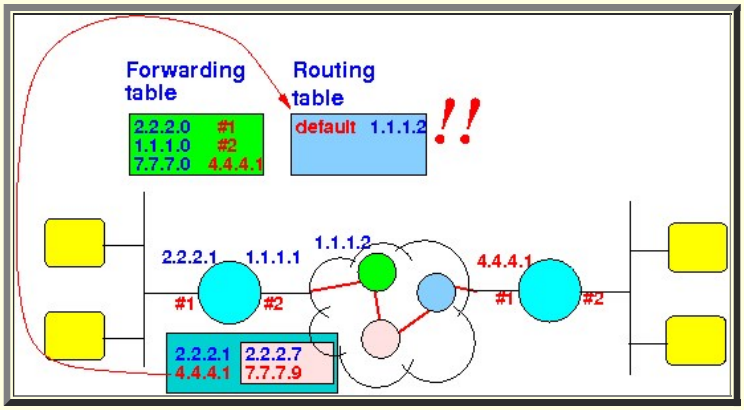
- **Network** `7.7.7.0` found in **forwarding table** with ***virtual*** **entry**:



The **router** will ***encapsulate*** (optionally with some ***pre-processing***):

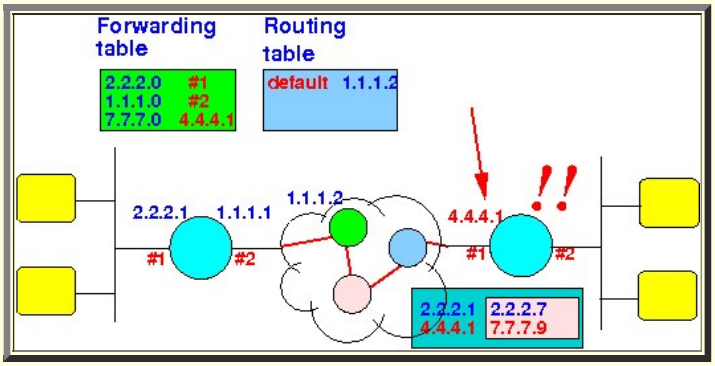


- Then the ***encapsulated*** **IP packet** is **forwarded**:



According to the **routing table**, the ***encapsulated*** **IP packet** will be **forwarded** to the ***default*** **router** !!!

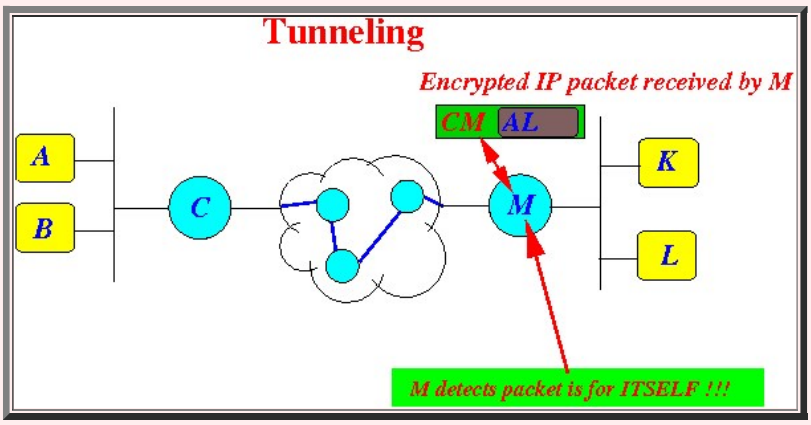
- The ***encapsulated*** **IP packet** will **eventually** arrive at the ***destination*** **router** of the **virtual interface**:



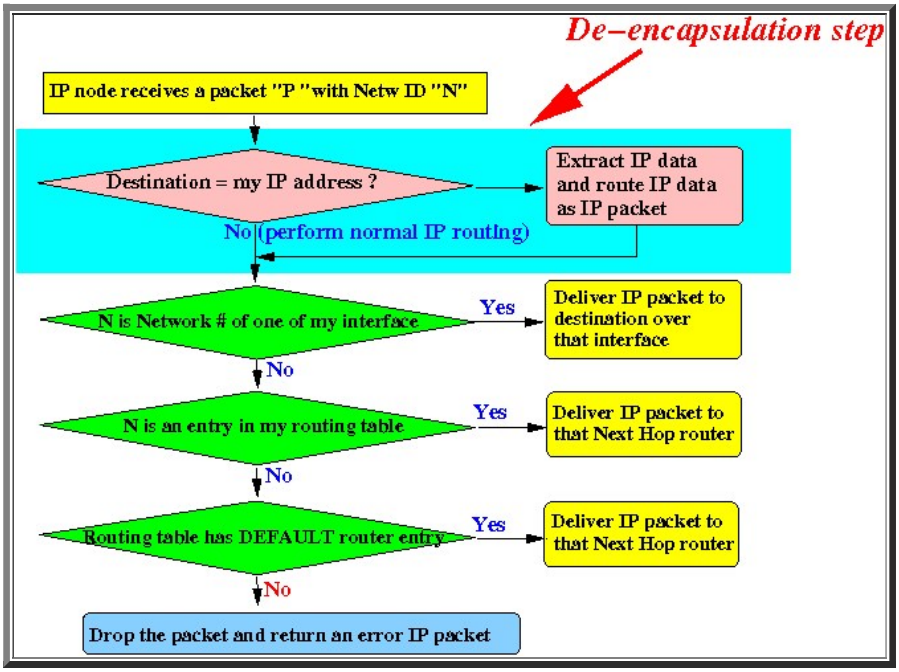
Implementing the *de-capsulation* procedure

- **Implementing the *de-encapsulation* procedure**
  - **Recall** on ***how*** a **router** can **determine** that it need to perform ***de-encapsulation***:

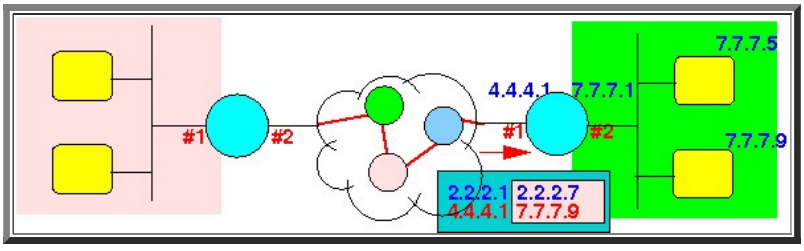
    - When the **destination router *M*** receives the **encapsulated IP packet**, ***M*** detects that the **IP packet** contains ***its own* address *M***:



  - The **IP forwarding algorithm** with **support** for ***tunneling***:

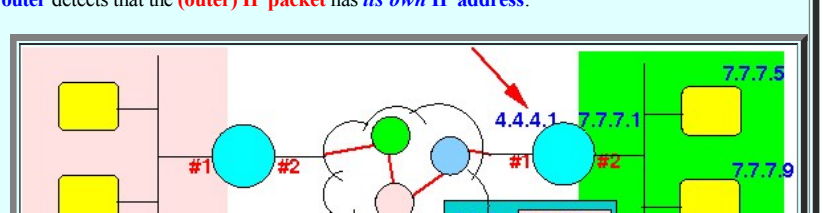


- **Example of the *de-capsulation* procedure**
  - **Suppose** the **encapsulated IP packet** is **received**:

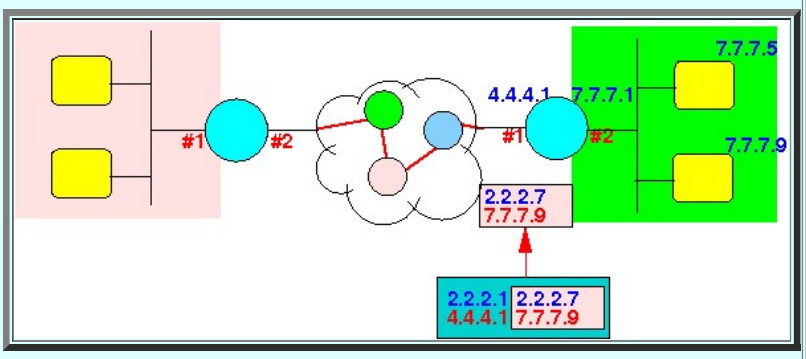


  - The **encapsulated IP packet** is processed as **follows**:

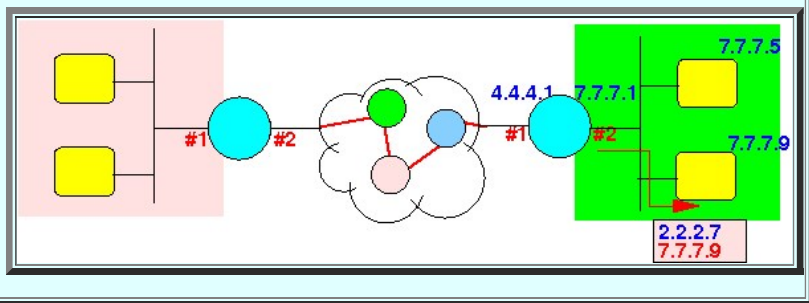
    - The **router** detects that the **(outer) IP packet** has ***its own* IP address**:



- The **router** will ***de-encapsulate*** **(extract)** the **inner IP packet**:

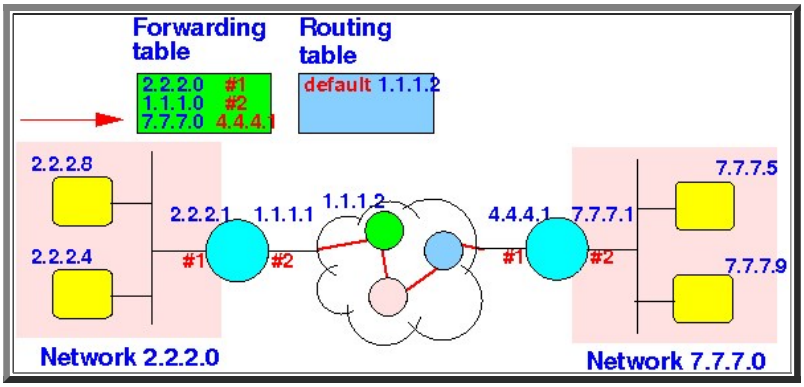


- The ***inner*** **IP packet** is then **forwarded/routed**:



Two-way (bi-directional) tunnels and *overlay* networks

- **Notable fact**
  - The **tunnel setup**:



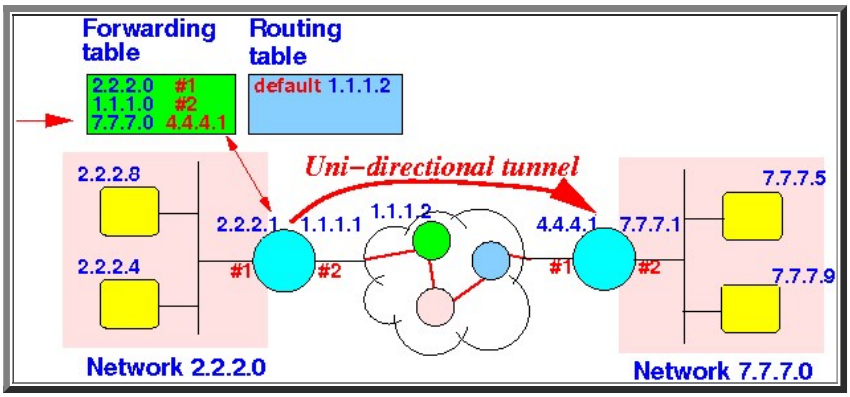
    created by the **virtual interface**:

```
Network ID  |  Interface (port)
------------+------------------
  7.7.7.0   |   4.4.4.1
```

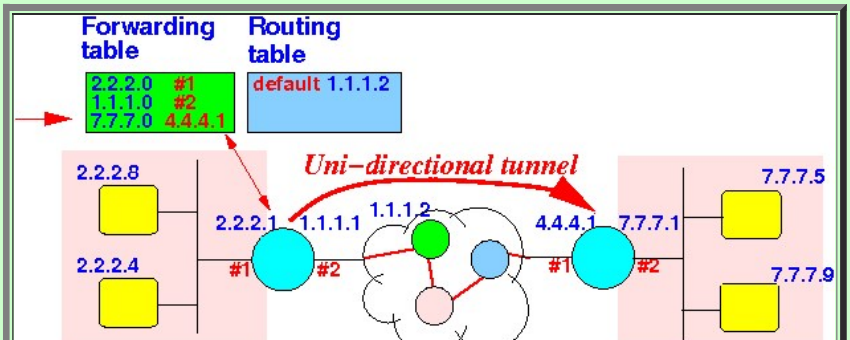
    is a

    - ***uni-directional*** **tunnel**
      - From **router 2.2.2.1**
      - To **router 4.4.4.1**

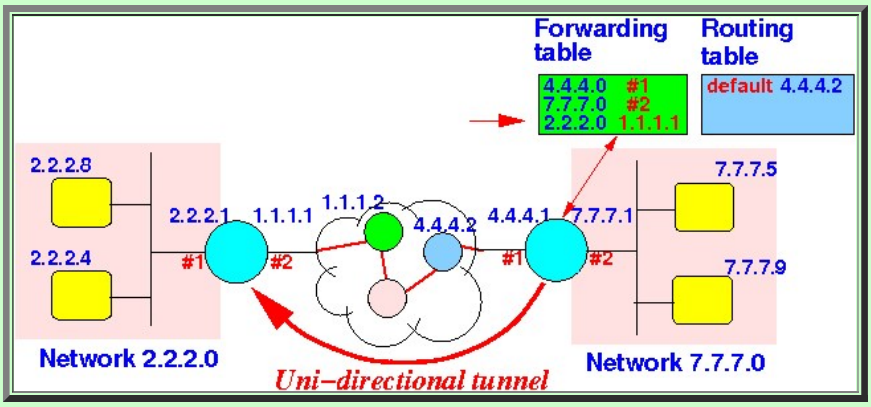
  - **Graphically:**



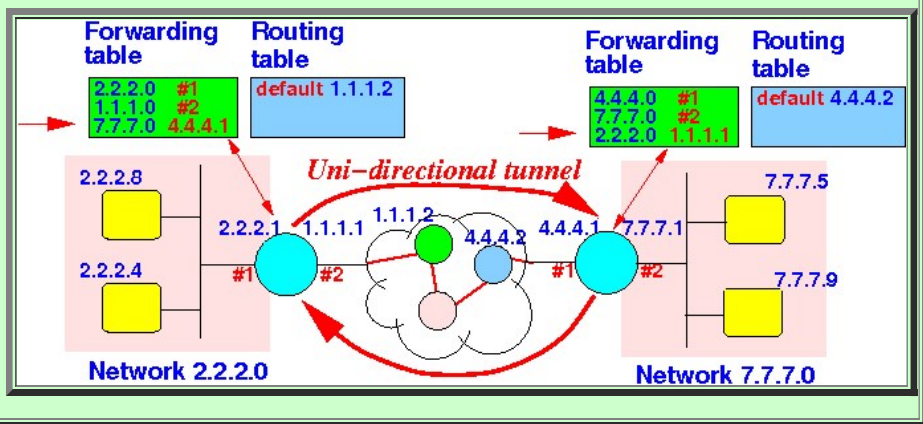
- ***Two-way*** **(bi-directional) IP-in-IP tunneling**
  - **Fact:**
    - A ***bi*-directional tunnel** (= **2 way**) is **created** using:
      - **Two *uni*-directional** tunnels
  - **How to** set up a ***two*-way tunnel**:
    - We first **setup** a ***one-way*** **tunnel**:



- Then **setup another** ***one-way*** **tunnel backwards**:



- Then the **2** ***one-way*** **tunnels** will make up a ***bi-directional*** **tunnel**:

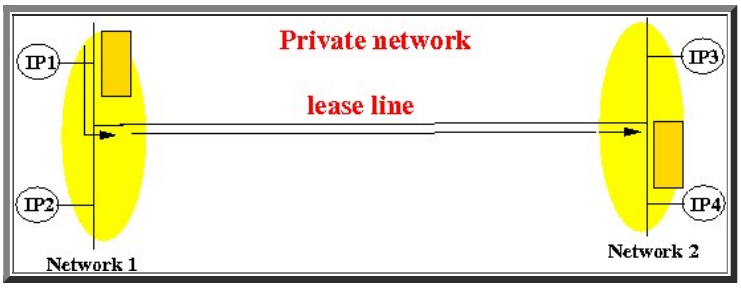


Applications of IP Tunneling: Secure IP

- **Secure IP networking**
  - ***Private* network**:

    - **Private Network (PN)** = a ***secure* communication network** usually own by a **coorporation** and **built** with **leased telecommunication lines**
      - **Communication** on a ***private* network** is ***very* hard to tap** (eaves drop)

    Graphically:

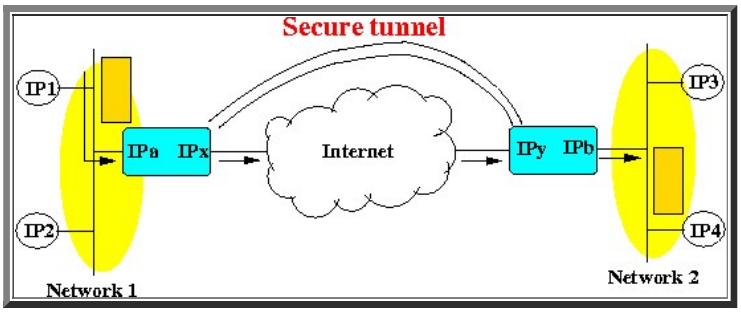
    

    (Commication is **secure** because the **leased line *cannot*** be **easily tapped**)

  - ***Secure* IP**: (**IPsec**)

    - **Secured IP communication** = a ***secure* communication** built **on top** of the **(unsecure) IP network**

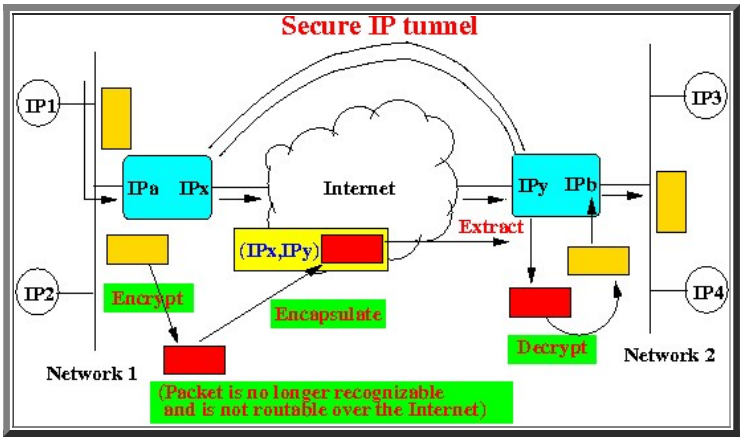
  - **Graphically:**

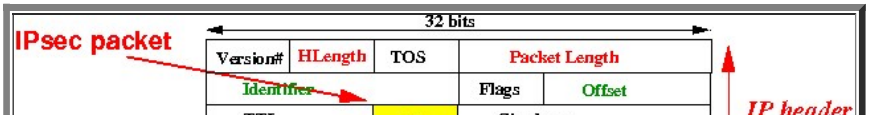
  - **How to** prevent ***eaves dropping*** and achieve **privacy protection:**

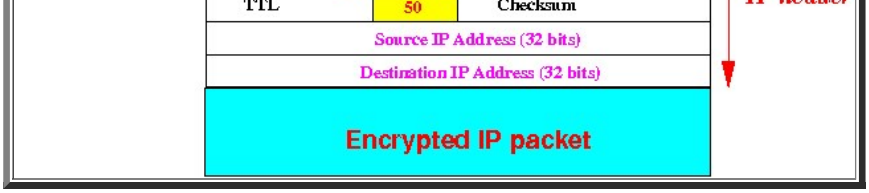
    - **Encrypt** the ***inner* IP packet**
    - Transmit the **encrypted IP packet securely** with the ***outer* IP packet**

  - **Graphically:**

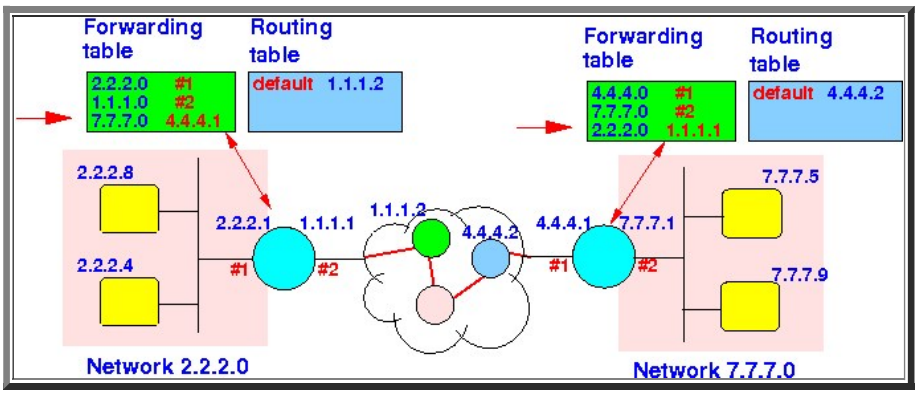
    

- **Packet format of IPsec**
  - **Packet format** used in **Secure IP**:



- **Example Secure IP**
  - **Tunnel setup:**

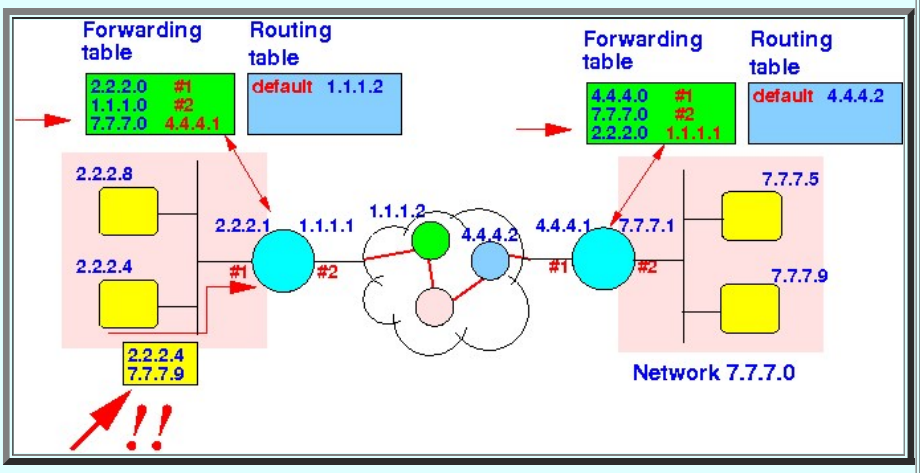


Note:

- The *optional* **processing** used **before** the **IP packet** is **encapsulated** is:
  - **Encryption** (and **authentication**)

  - **How** an **(plain) IP packet** will be **transmitted**:

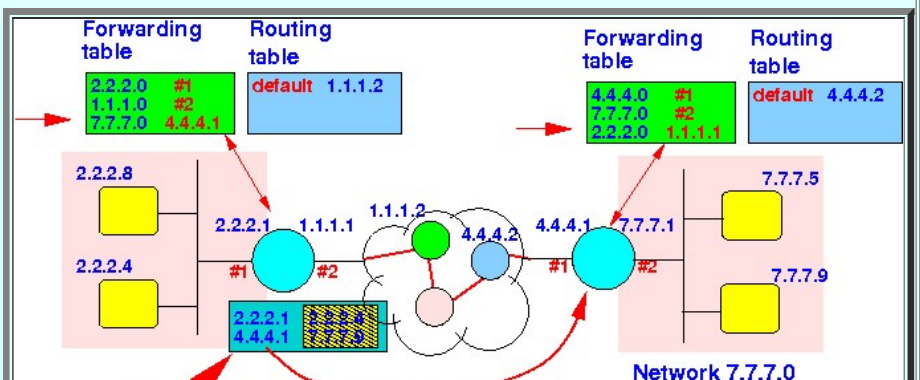
- Host **2.2.2.4** transmit a **plain IP message** to **7.7.7.9**:



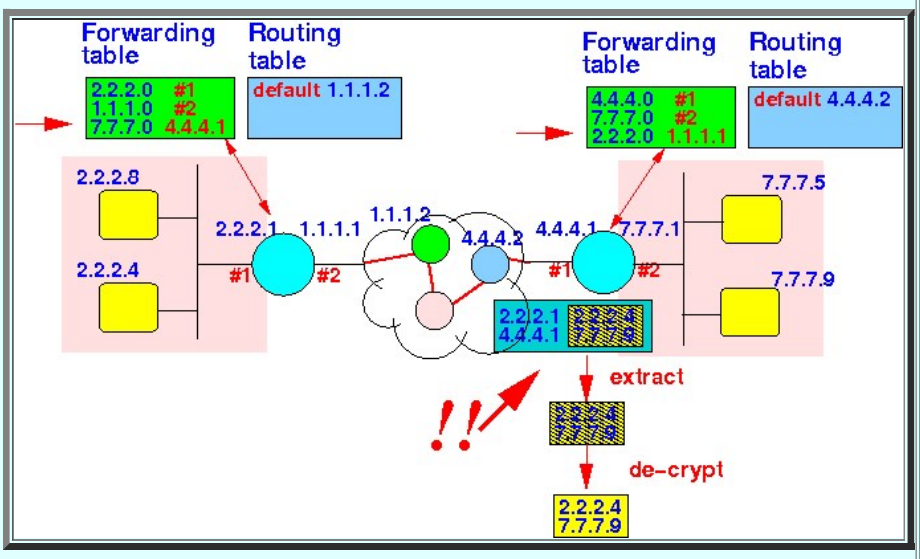
- The *virtual* **interface** entry will **cause** the **IP packet** to be **encrypted** and **encapsulated**:

  The *other* **IP packet** will use the **packet code 50** to indicate it is a **IPsec** encapsulation

- The *encapsulated* **IP packet** will be **sent** to the **router 4.4.4.1**:

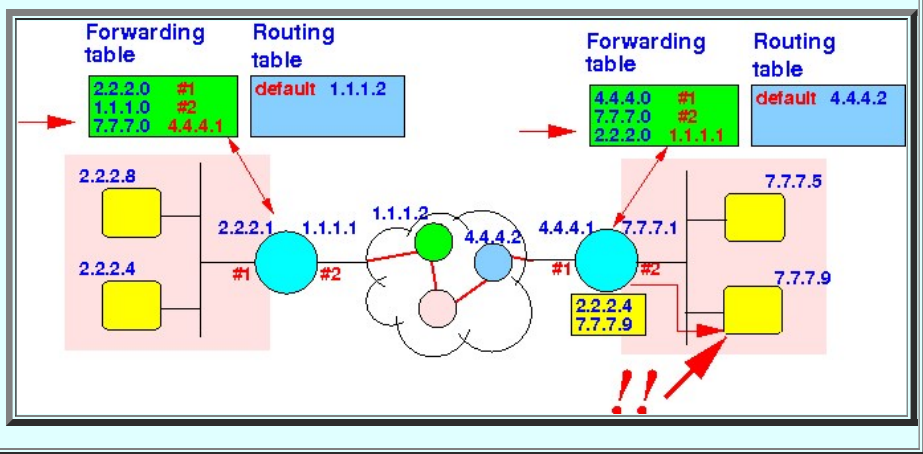


- When the *encapsulated* **IP packet** arrives at **4.4.4.1**, the **router** detects its *own* **IP address** and **de-encapsulate**



The **router** will **decrypt** the ***inner*** **packet** because the ***outer*** **code = 50**

- The ***de-crypted*** **IP packet** is **forwarded** further:



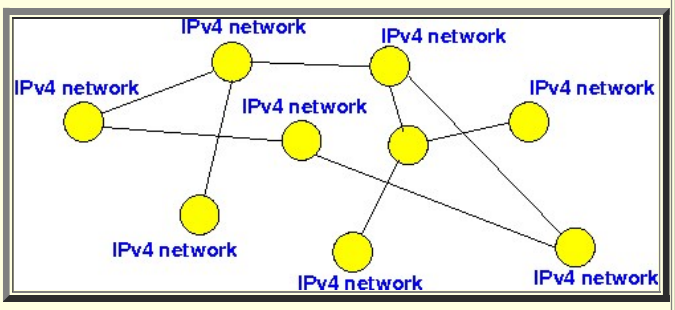
- **Postscript**
  - **IPsec** also provides an **authentication mechanism** that I have **omitted**

  - **External material** with the **complete** description of **IPsec**: **click here**

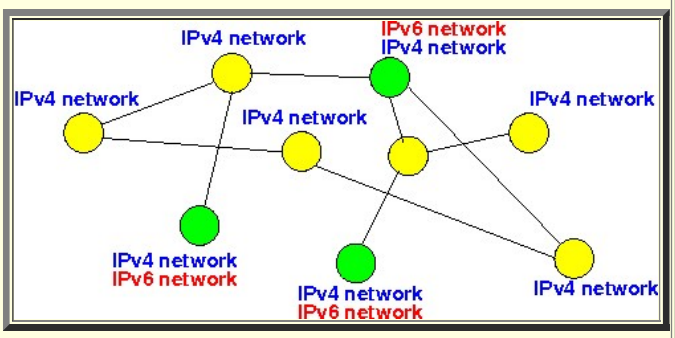
Applications of IP Tunneling: IPv6 over IPv4

- **Deploying IPv6...**
  - **Fact:**

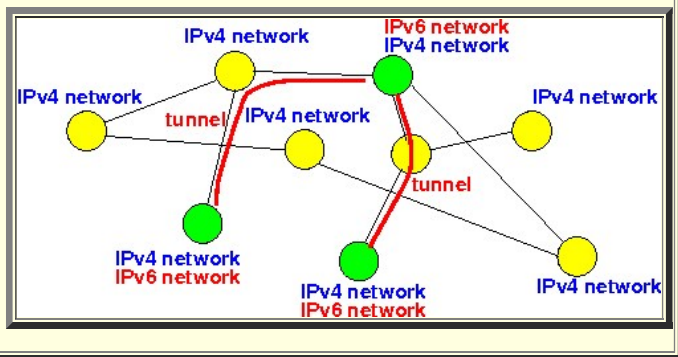
    - **In the beginning**: there was **no IPv6** networks

    - Then a **few institutes** (**pioneers** implements and **deploys** the **IPv6 network software**:

    - **During** the **initial deployment**, the **IPv6 networks** were **connected** with **IPv4-tunnels**:

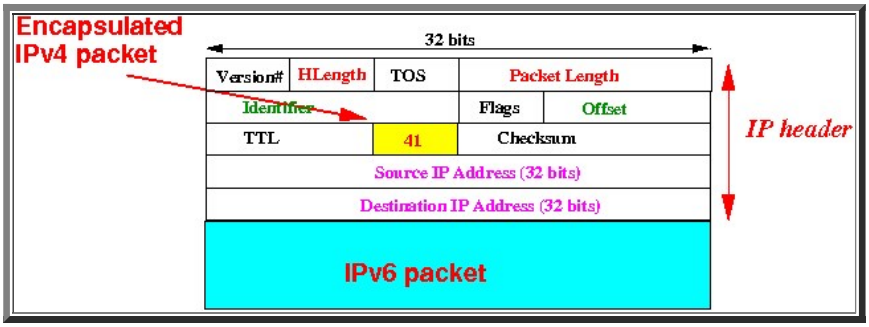
      

  - **In other words:**

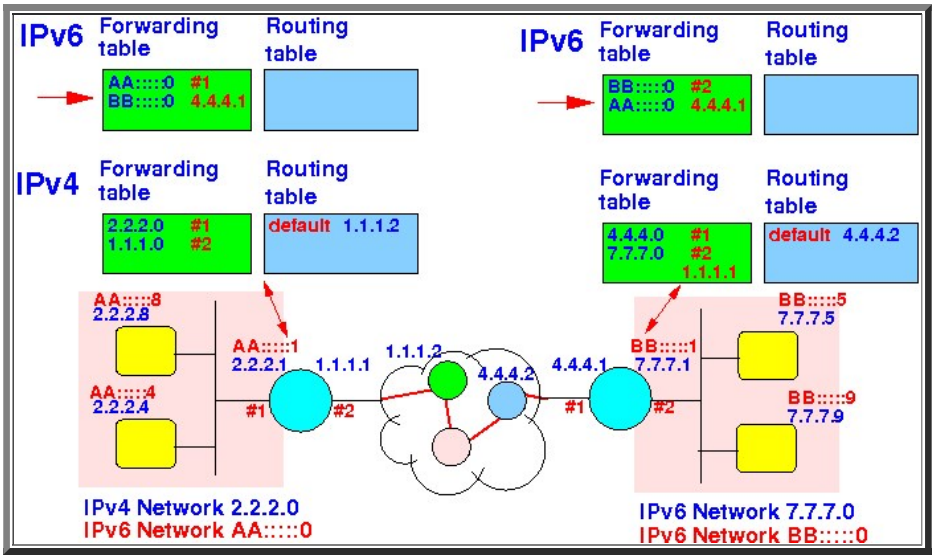
    - **Tunneling** encapsulates **IPv6 packets** within **IPv4 packets** so:

      - **IPv6 packets** can be **sent** over an **IPv4 network**, allowing ***isolated*** **IPv6 networks** to communicate ***without*** the need to **upgrade** the **IPv4 routers** that ***between*** **them**.

- **Packet format used in IPv6-in-IPv4 encapsulation**
  - **Packet format** used in **IPv6 in IPv4** encapsulation:

- **Example IPv6 in IPv4 tunneling**
  - **Tunnel setup:**



Note:

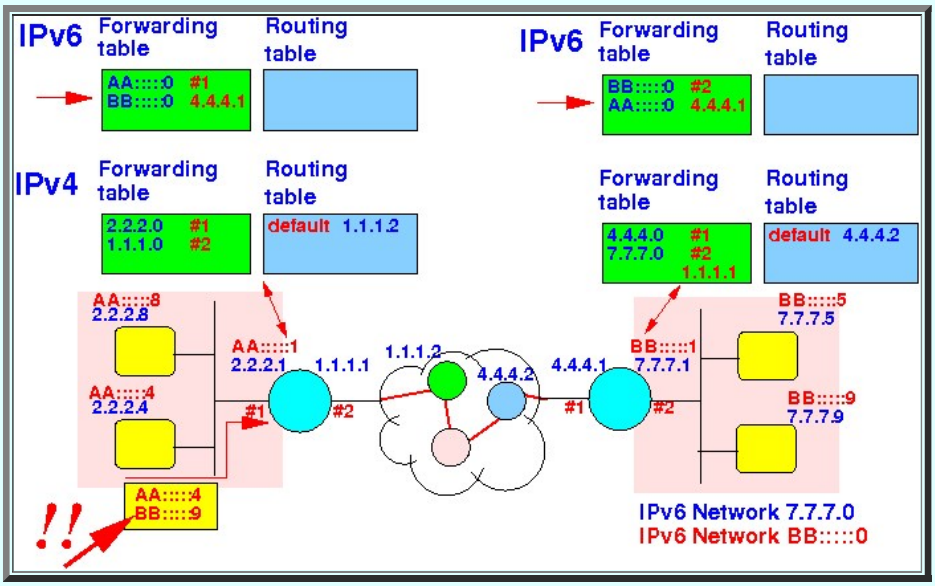
- The **edge routers** must have a ***dual* stack**, running:
  - **IPv4** and
  - **IPv6**

  routing software

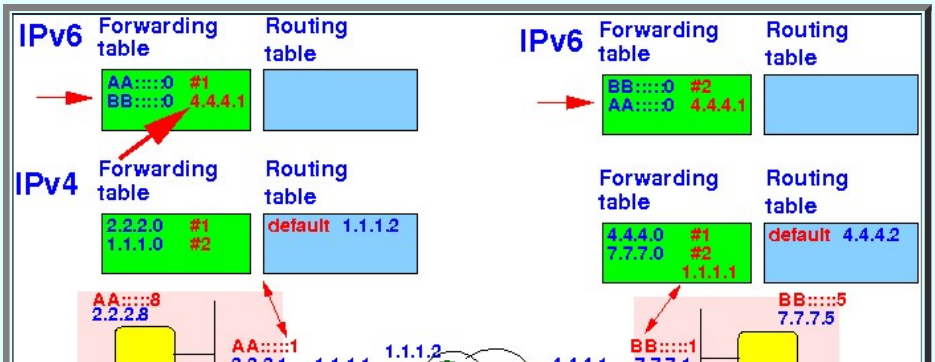
- The **IPv6 implementation** contains the **IPv4 encapsulation software** to **deploy** over **IPv4** !!!

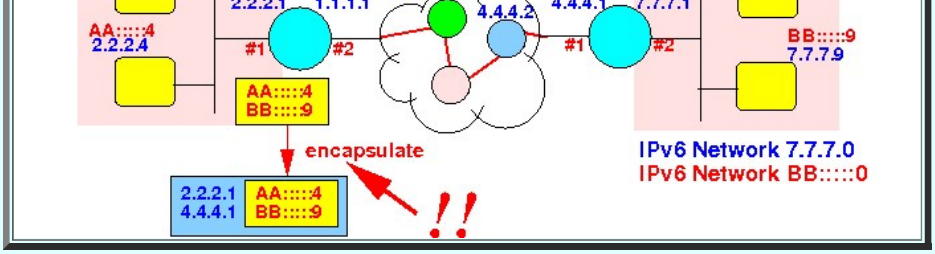
  - **How** an **IPv6 packet** will be **transmitted**:

- Host **AA::::4** transmit an **IPv6 message** to **BB::::9**:



- The ***virtual* interface** entry and the **IPv6 encapsulation** software will **cause** the **IPv6 packet** to be **encapsulated** in an **IPv4 packet**:

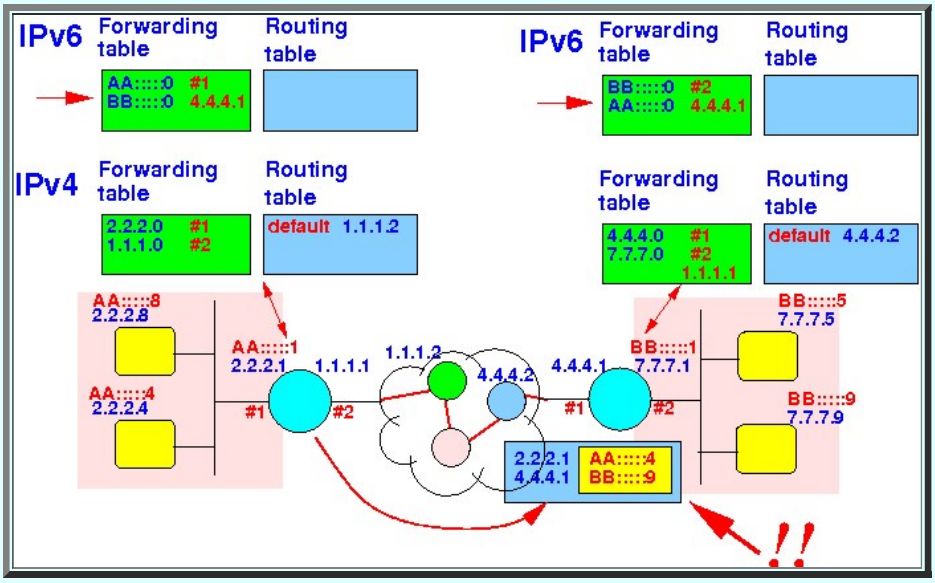




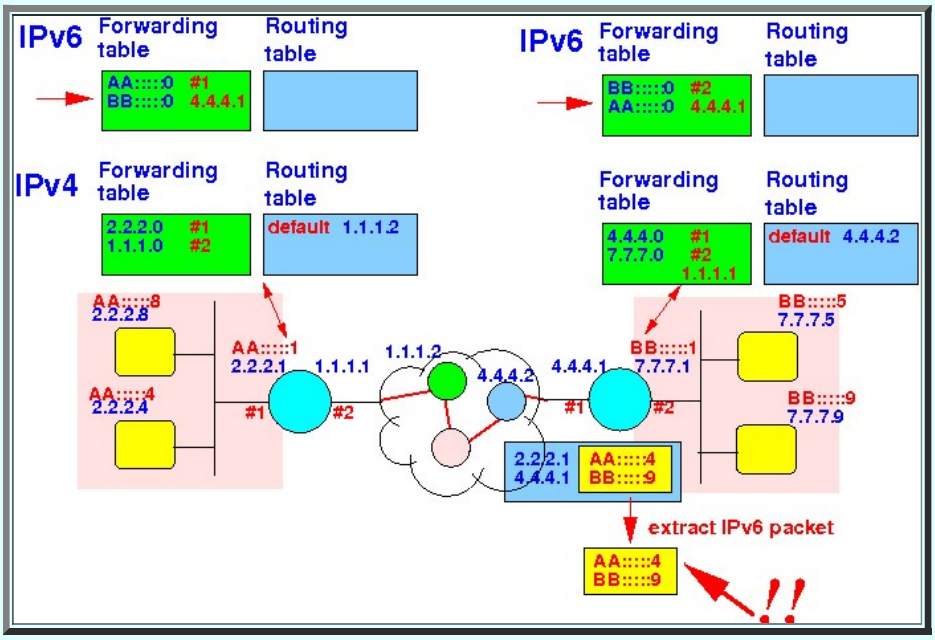
The **IPv4 packet** will use the **packet code 41** to indicate it is a **IPv6** encapsulation

---

- The ***encapsulated* IPv4 packet** will be **sent** to the **router 4.4.4.1**:

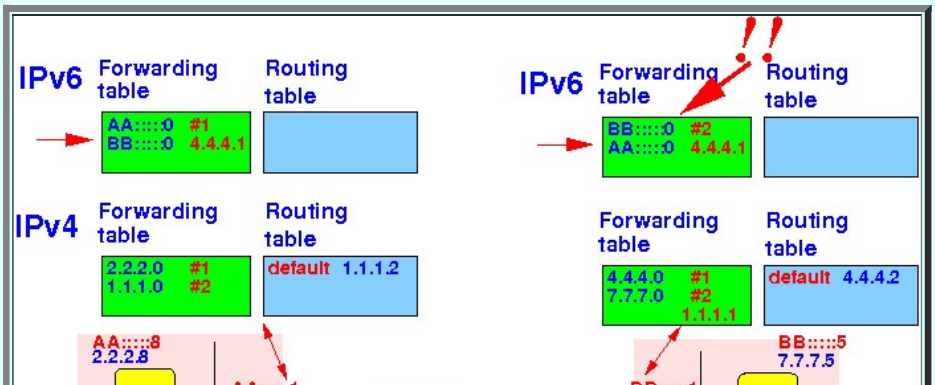


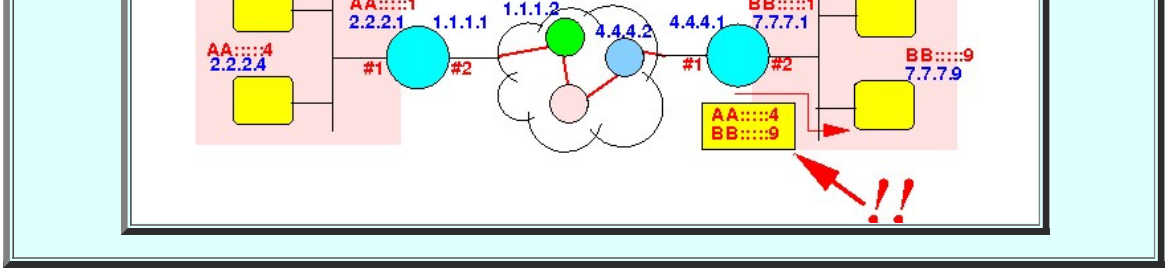
- When the ***encapsulated* IP packet** arrives at **4.4.4.1**, the **router** detects its ***own* IP address** and **de-encapsulate**



The **router** will **process** the ***inner* packet** an an **IPv6 packet** because the ***outer* code = 41**

- The **IPv6 packet** is **forwarded** further by the **IPv6 routing software** in the router:





- **Postscript**
  - **6bone**:

    - **6bone** = the **tunneled network** consisting of the **IPv6 networks**

      Wikipedia: click here

  - **Nowadays**:

    - ***Every* router** (and **networked computer**) runs **dual stack**

      (Has both **IPv4** and **IPv6** software)

      - **IPv6** is now the **standard**

    - **Most network services** (such as **Google, Yahoo, ...**) provide **access** on **both IPv4** and **IPv6** networks

      Mainly because:

      - **Most computers** in the **USA** still runs **IPv4 software** in **2016**

      (That can change soon).

Applications of IP Tunneling: implement IP multicasting (MBone)

- **Multicasting**
  - **Multicasting:**
    - **Multicast** = a **transmission** from a **sender** is **received** by ***multiple*** **receivers**

- **IP-multicasting (multicast tunnels)**
  - **IP-multicasting** uses a **special set of IP-addresses** that are ***not*** **assigned to IP-hosts/IP-networks**.
  - In other words:
    - **Multicast packets** uses a ***non-routable*** **IP address**
      - I.e., the **routing tables** do ***not*** contain **entries** for **multicast addresses**

  - The **first implementation** of **IP-multicast** uses **IP-tunneling** to **networks** that supported **multicast operation**.
    - The ***overlay*** **network** that supported **multicast transmission** was known as:
      - The ***MBone*** **network**

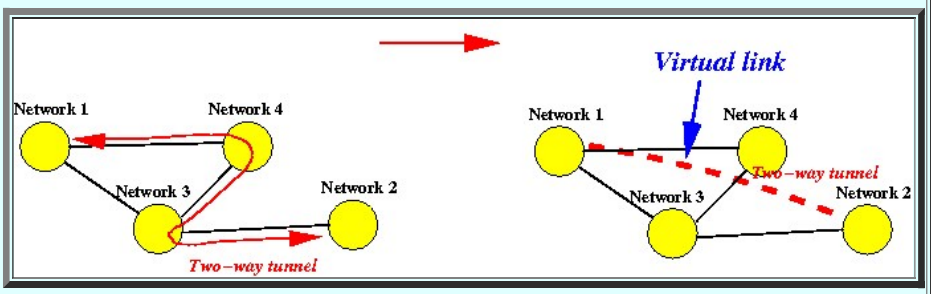
        Wikipedia: click here

  - The **MBone** is **exactly** the **same concept** as the **IPv6-bone** when **IPv6** was being **deployed**....

Overlay networks

- ***Overlay* network**
  - **Tunnels** and ***virtual* links**:

    - We can view a **tunnel** as a **"virtual (direct) link"** consisting of ***multiple network hops***



Fact:

- We can **create** a **network** using ***virtual* links (tunnels)** !!!

  - ***Overlay* network**:

- ***Overlay* network:**
  - ***Overlay* networks** = a **network** that is **created** by using **tunnels *on top*** of another network

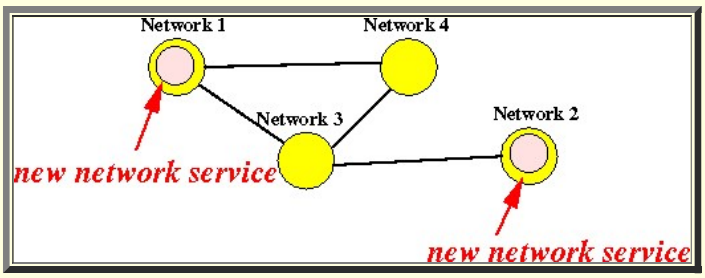
    Wikipedia: click here

- ***How* is the *overlay* network used ?**
  - **Fact:**

- The ***overlay* network** technqiue is a **powerful technique** used to **inplement *novel* services** on top of the ***existing* network**

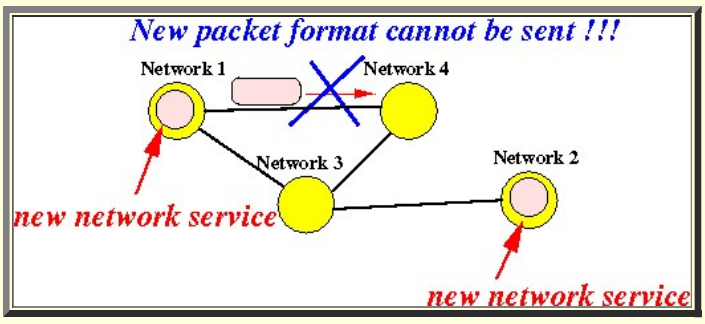
  - **Example:** (**IPv6** and **Multicasting**)

- Suppose **networks 1 and 2** have **implemented** a ***new* network service** (such as **IPv6**):

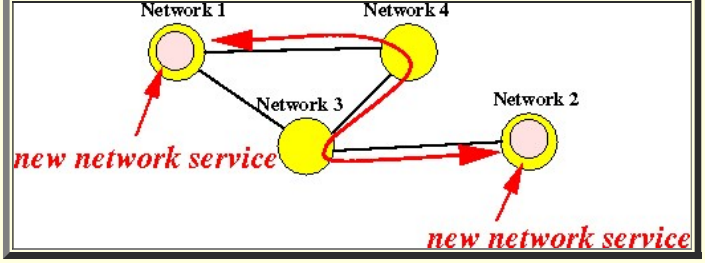


- The ***new* network service** uses its ***own* packet format** !!

  These **packets *cannot*** be **sent** on the **existing network**:



- We can **create** a ***tunnel*** between **nodes** that **support** the ***new* network service**:



- The ***new*** **format packets** can now be ***encapsulated*** inside an **IP packet** and sent ***directly*** between the **networks**: