Internet Connectivity

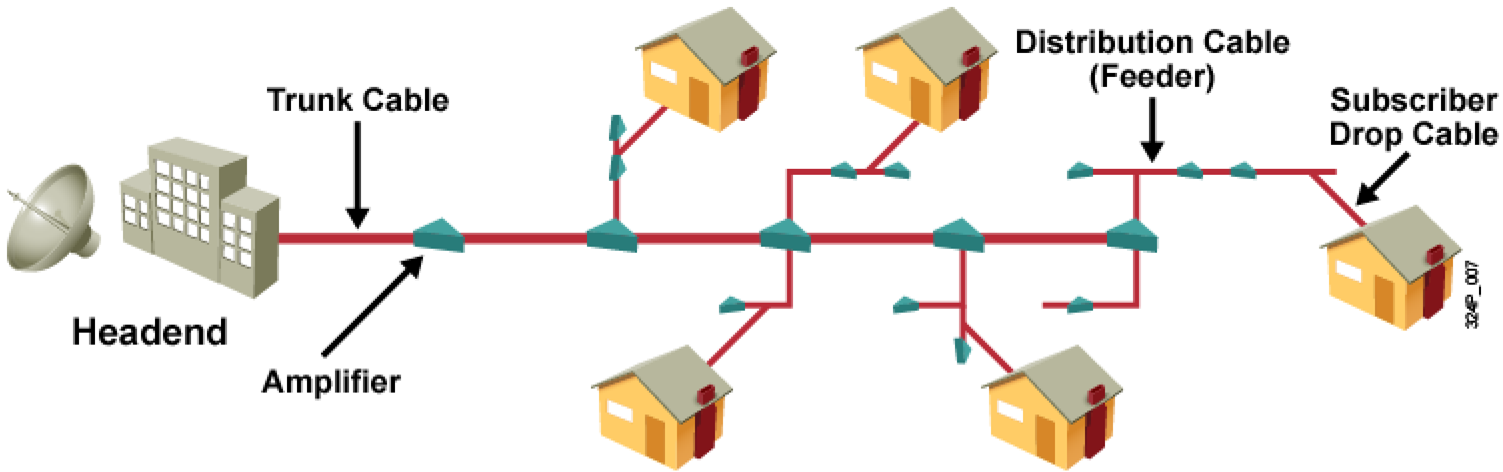
ROUTE Module 6

Agenda

- Introduction
- Cable Antenna TV
- Digital Subscriber Line
- Tunneling
- IPsec

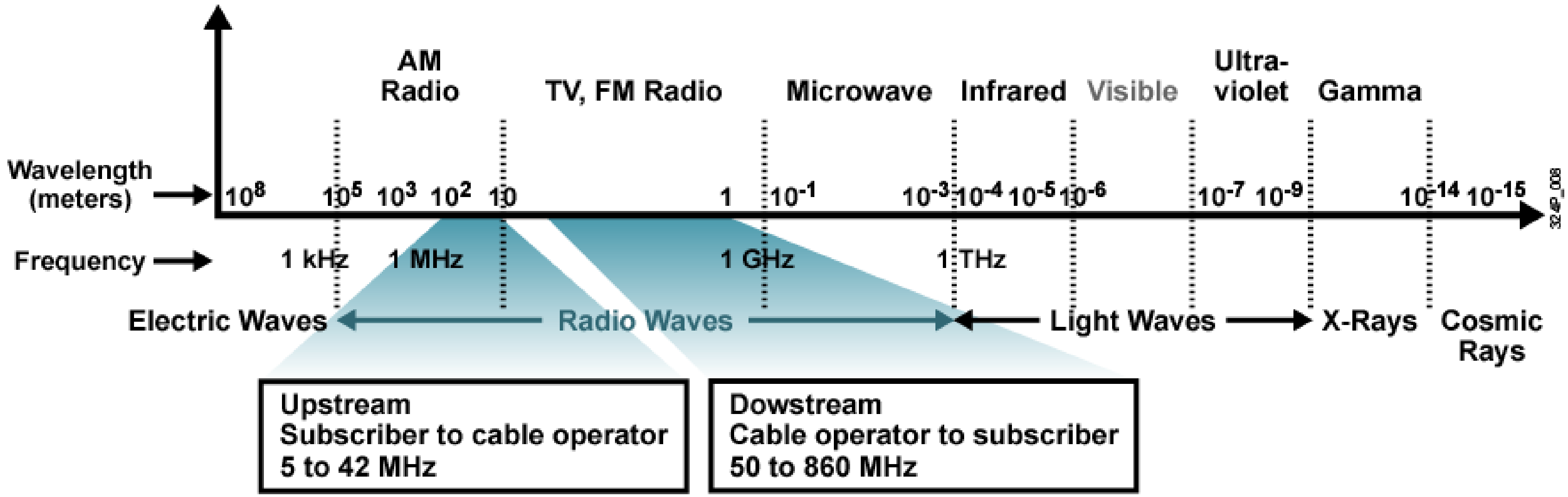
CATV

What Is Cable Network?



- CATV solved delivery of TV signal to places where signal receptions were poor
- Current CATV systems use combination of optical and coaxial cables to deliver not just TV but also (IP) data

Bandwidth Frequency

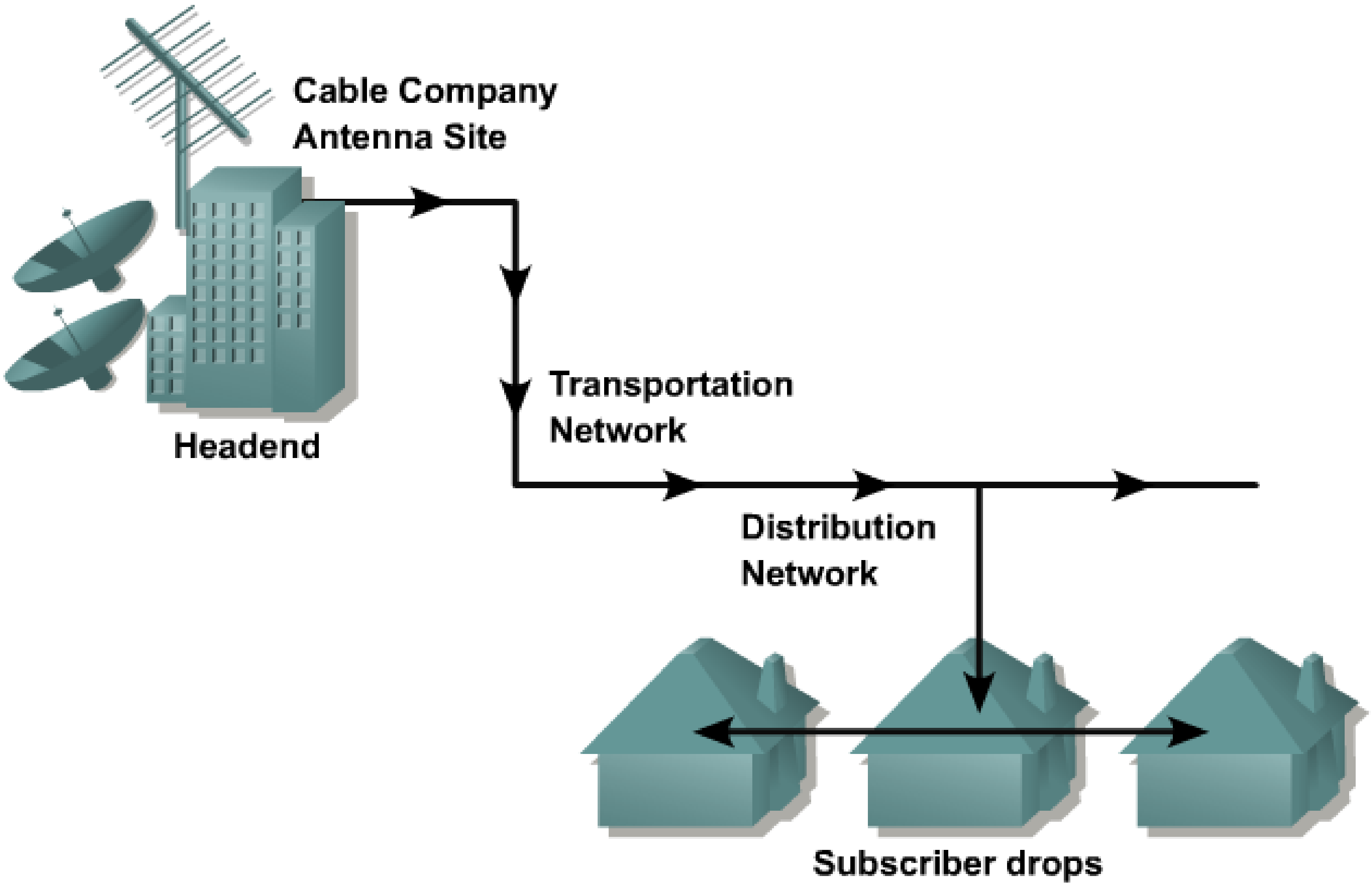


- Bandwidth between 5 and 860 MHz is used for data transfers
  - Downstream: 810 MHz (downstream BW)
  - Upstream: 37 MHz (upstream BW)
  - Guard band: protective separation BW between 42 and 50 MHz

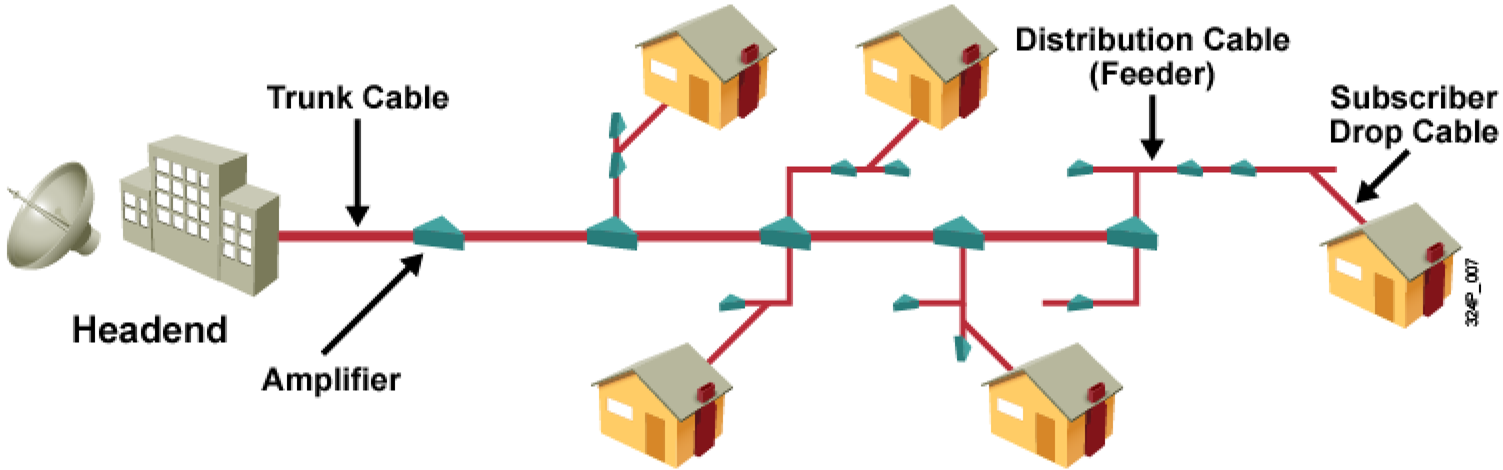
CATV System Components ①

- **Antenna site (AS)**
  - *Antenna is installed over here*
- **Headend, Head End (HE)**
  - *TV signal from antenna is processed here*
- **Transportation Network (TN)**
  - Network interconnecting AS and HE or HE and DN
- **Distribution Network (DN)**
  - Network between subscribers
- **Subscriber Drop**
  - Cable and necessary technical equipment (set-top box) interconnection subscriber to DN

CATV System Components ②

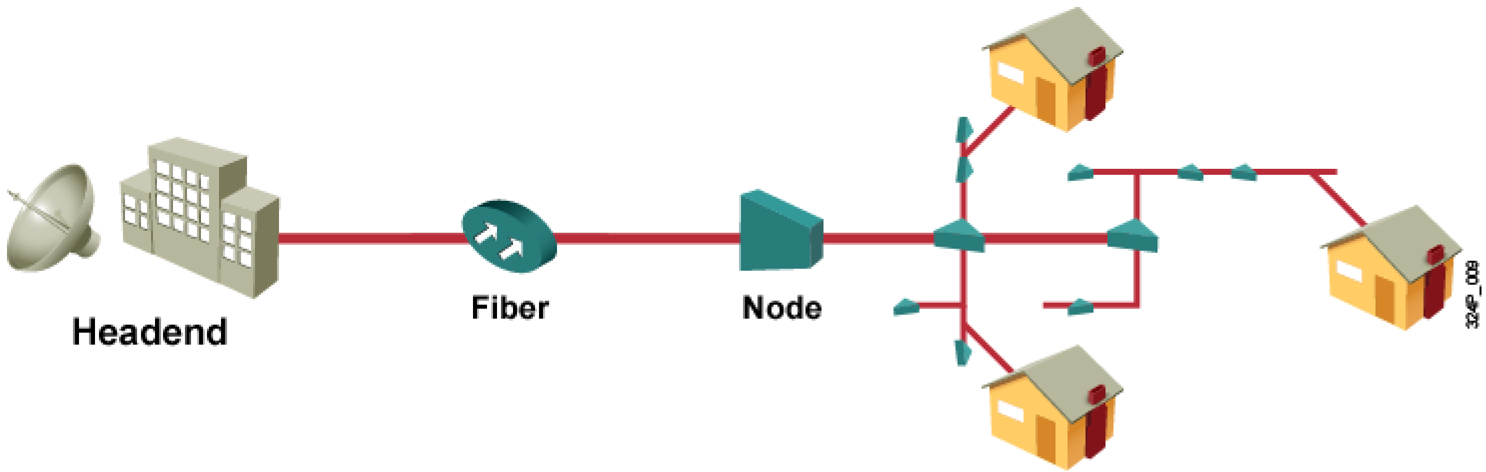


Old CATV Architecture



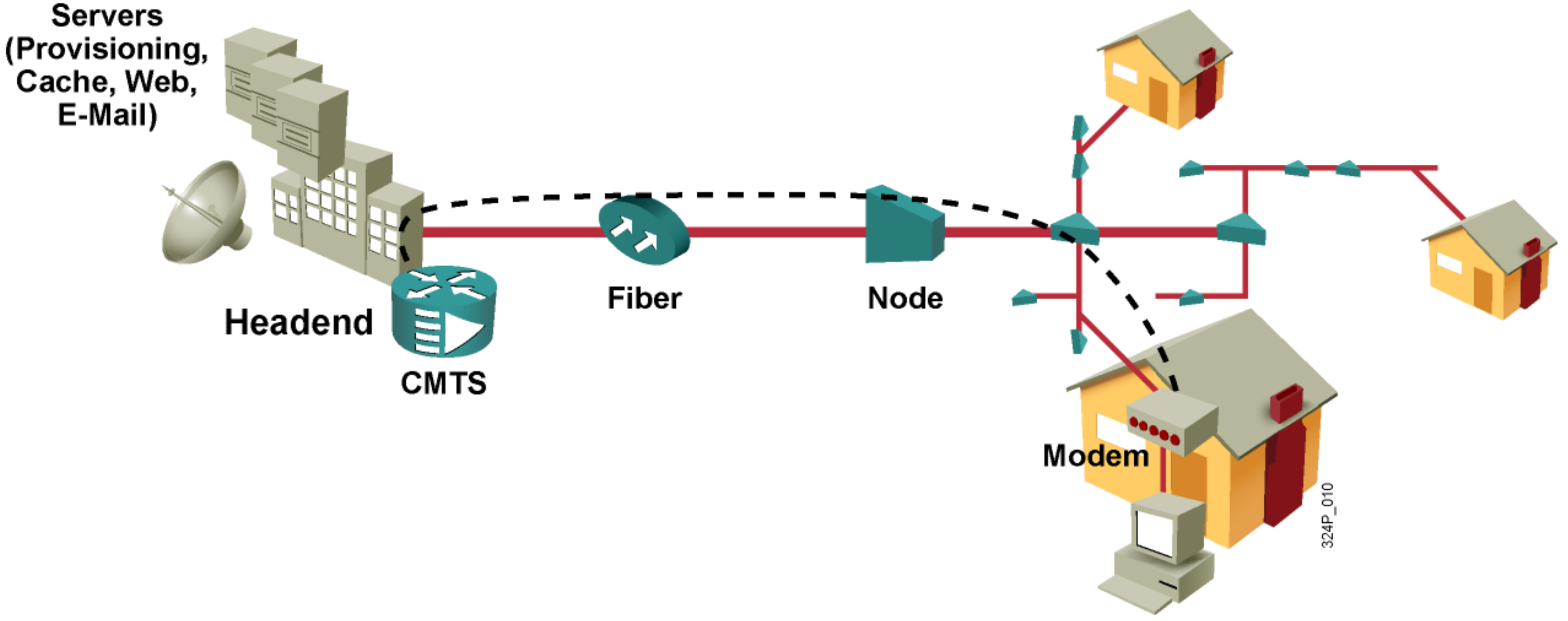
- Cabling is divided into two groups:
  - **Trunk** is backbone for service area (coax, 19 mm)
  - Trunk splits into **feeders** (coax, 13 mm)
  - Subscribers are connected via **taps** to feeders
- Range of this network is limited
  - Coaxial cable has signal attenuation
  - Amplifier regenerates not even signal but also noise

New CATV Architecture



- **Hybrid Fiber Coax (HFC)** = combination of optical and metallic cabling
- TN and nowadays even feeder are implemented via optical fiber
- Advantages
  - Reduces the number of amplifiers
  - Optical cable is more space convenient and protected against EMI
  - Covers longer distances

Data Transfers in CATV



- Two keynote devices
  - On customer side **cable modem (CM)**
  - On providers side **concentrator (CMTS = cable modem termination system)**
- Subscribers on one segment share BW
  - But are unable to communicate between each other – all communication traverse CMTS first

Throughput and Security Issues

- Users in one service are share coaxial cable resources
  - For one feeder there should be no more than 2000 subscribers
  - Number of amplifiers in cascade should be no more than 5
- Insufficient bandwidth could be solved with:
  - Allowing some TV channels for data transfers instead
  - Decreasing of service area size

- *All technical and operational standards of CATV are inside…*

DOCSIS

- **Data Over Cable Service Interface Specification a.k.a. DOCSIS** is international standard maintained by CableLabs
- In Europe slightly different BW for channels hence over here is **EuroDOCSIS**
- DOCSIS fully specifies behavior of CATV on L1 and L2 like channel width, modulation and access methods
- From v3 even standards on L3 (IPv6 management and deployment)
- Versions
  - 1.0 – initial proposal
  - 1.1 – extended technical specification
  - 2.0 – support for QoS and symmetrical real-time applications
  - 3.0 – support for channel bonding

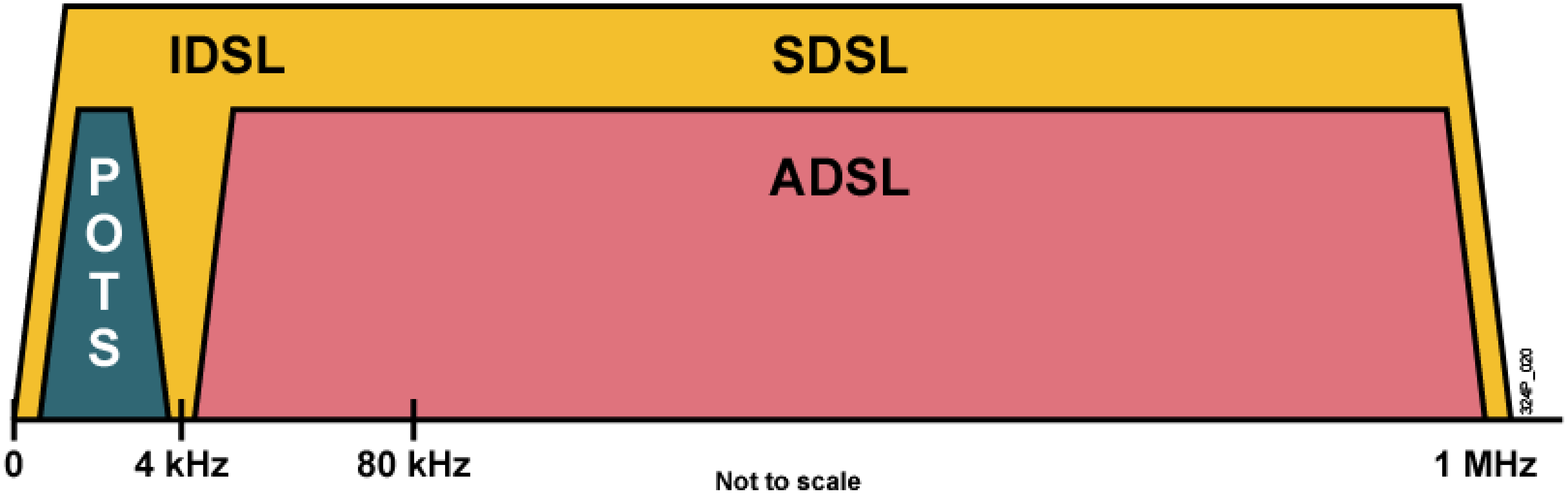
DOCSIS Speeds

Version	Downstream: Channel configuration: Minimum selectable channels	Minimum channels supported by HW	Selected number of channels	Maximum number of channels	DOCSIS throughput	EuroDOCSIS throughput	Upstream: Channel configuration: Minimum selectable channels	Minimum channels supported by HW	Selected number of channels	Maximum number of channels	Throughput
1.x	1	1	1	1	42.88 (38) Mbit/s	55.62 (50) Mbit/s	1	1	1	1	10.24 (9) Mbit/s
2.0	1	1	1	1	42.88 (38) Mbit/s	55.62 (50) Mbit/s	1	1	1	1	30.72 (27) Mbit/s
3.0	1	4	m	No maximum defined	m × 42.88 (m × 38) Mbit/s	m × 55.62 (m × 50) Mbit/s	1	4	n	No maximum defined	n × 30.72 (n × 27) Mbit/s

Channel configuration: Number of downstream channels	Channel configuration: Number of upstream channels	Downstream throughput: DOCSIS	Downstream throughput: EuroDOCSIS	Upstream throughput
4	4	171.52 (152) Mbit/s	222.48 (200) Mbit/s	122.88 (108) Mbit/s
8	4	343.04 (304) Mbit/s	444.96 (400) Mbit/s	122.88 (108) Mbit/s
16	4	686.08 (608) Mbit/s	889.92 (800) Mbit/s	122.88 (108) Mbit/s
24	8	1029.12 (912) Mbit/s	1334.88 (1200) Mbit/s	245.76 (216) Mbit/s
32	8	1372.16 (1216) Mbit/s	1779.84 (1600) Mbit/s	245.76 (216) Mbit/s

DSL

What is Digital Subscriber Line?



- Classic telephone network transfer voice on local loop in frequency range between 300 and 3400 Hz
- Idea behind DSL is to use available bandwidth above 4 kHz for data transfer
  - Thanks to frequency separation voice and data could be tranfered at the same time
  - Available speeds are much more faster than plain modem connection
- DSL is L1 technology

Terms

- **xDSL**
  - One of many DSL technology variants
- **DSL Access Multiplexor (DSLAM)**
  - Device on DSL service provider side (telephone exchange) which serves as concentrator of DSL users
- **Broadband Remote Access Server (BRAS)**
  - Router on DSL service provider side which processes data flows of each DSL user
  - BRAS terminates ATM VC from subscribers and sends their traffic via L2TP to ISP
- **Splitter**
  - Frequency separator for voice (bellow 4 kHz) and data (above 4 kHz)
- **Microfilter**
  - Low-pass frequency filter for telephone

Basic Facts

- DSL is always-on technology
  - DSL works directly on telephone link hence there's no initial dialup
  - DSL modem connects to closest DSLAM
- DSL service provider (DSL SP) and internet service provider (ISP) are strictly differentiated when using DSL
  - DSL SP is telephone operator
  - Data from subscriber are transferred through DSLAM to BRAS and from BRAS via L2TP to IP center of ISP
  - ISP is first one which concerns about L3 content of IP packet
- Data between DSL modem and DSLAM are transferred as ATM cells – design choice with many good aspects
  - *To understand whole set of encapsulation processes in DSL is slightly more complicated* ☺

DSL Variants ①

- *There exists many different DSL technology variants!*
- Differences rely in many properties:
  - Symmetric vs. asymmetric download/upload speed nature
  - Number of used pair in wiring (1 or 2 pairs)
  - Maximal available speed for both directions
  - Channel modulation
  - Support of voice and data coexistence
  - Maximal distance of wiring

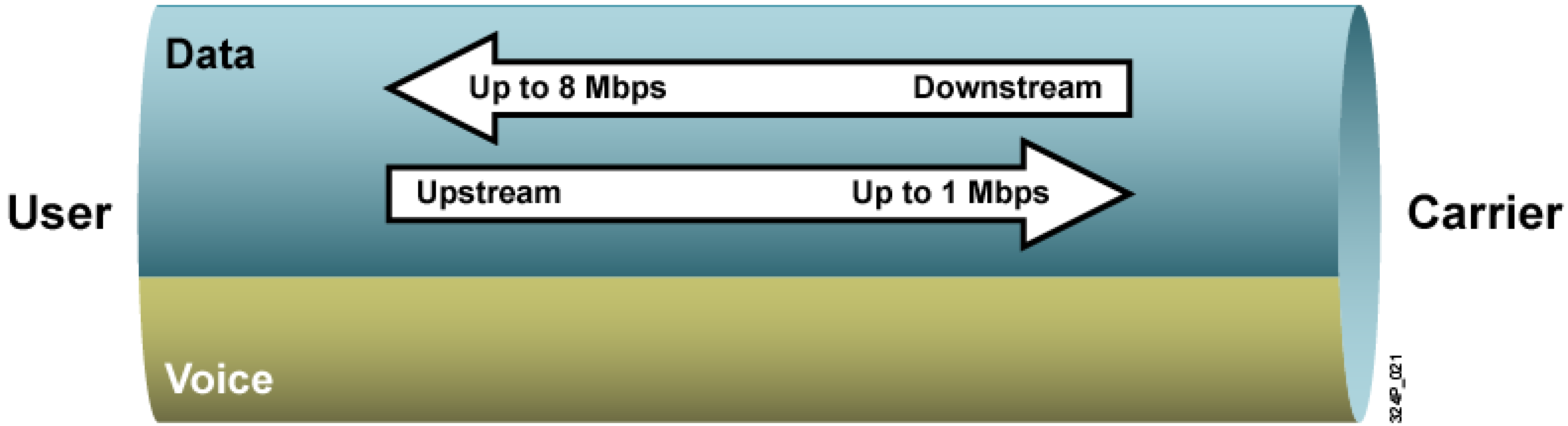
DSL Variants ②

DSL Technology	Nature	Max speed (Down / Up)	Voice + Data
ADSL2+	Asymmetric	24 M / 1.4 M	Yes
ADSL2	Asymmetric	12 M / 3.5 M	Yes
ADSL	Asymmetric	8 M / 1 M	Yes
VDSL	Symmetric or Asymmetric	52 M / 13 M	Yes
IDSL	Symmetric	144 k / 144 k	No
SDSL	Symmetric	768 k / 768 k	No
HDSL	Symmetric	2 M / 2 M	No
G.SHDSL	Symmetric	2.3 M / 2.3 M	No

Factors Influencing Speed and Range

- **Signal attenuation**
- **Bridge tap**: Cable end connected to the local loop
- **Load coil**: A loading coil is a wrap of wire placed at specific intervals along the local loop that extends the local loop distance
- **Wire gauge**
- **Impedance mismatch**: Noise or an echo in the local loop that is caused by changes in wire gauge, wire splices, or corrosion is called impedance mismatch
- **Crosstalk**
- **AM radio interference**
- **Fiber-optic cable**

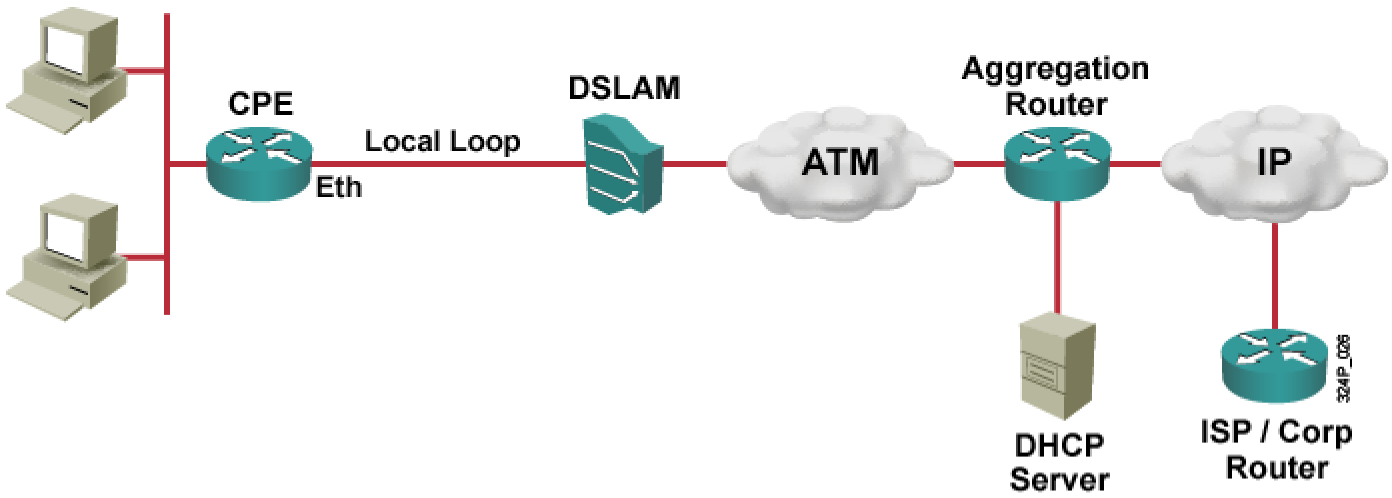
Asymmetric DSL



- ADSL exists on the same twisted-pair telephone line as the POTS

Version	Standard name	Common name	Downstream rate	Upstream rate	Approved in
ADSL	ANSI T1.413-1998 Issue 2	ADSL	08.08.0 Mbit/s	1.0 Mbit/s	1998
ADSL	ITU G.992.1	ADSL (G.DMT)	12.0 Mbit/s	1.3 Mbit/s	1999-07
ADSL	ITU G.992.1 Annex A	ADSL over POTS	12.0 Mbit/s	1.3 Mbit/s	2001
ADSL	ITU G.992.1 Annex B	ADSL over ISDN	12.0 Mbit/s	1.8 Mbit/s	2005
ADSL	ITU G.992.2	ADSL Lite (G.Lite)	01.51.5 Mbit/s	0.5 Mbit/s	1999-07
ADSL2	ITU G.992.3	ADSL2	12.0 Mbit/s	1.3 Mbit/s	2002-07
ADSL2	ITU G.992.3 Annex J	ADSL2	12.0 Mbit/s	3.5 Mbit/s	
ADSL2	ITU G.992.3 Annex L	RE-ADSL2	05.05.0 Mbit/s	0.8 Mbit/s	
ADSL2	ITU G.992.4	splitterless ADSL2	01.51.5 Mbit/s	0.5 Mbit/s	2002-07
ADSL2+	ITU G.992.5	ADSL2+	24.0 Mbit/s	1.3 Mbit/s	2003-05
ADSL2+	ITU G.992.5 Annex M	ADSL2+M	24.0 Mbit/s	3.3 Mbit/s	2008

Data Transfers in ADSL



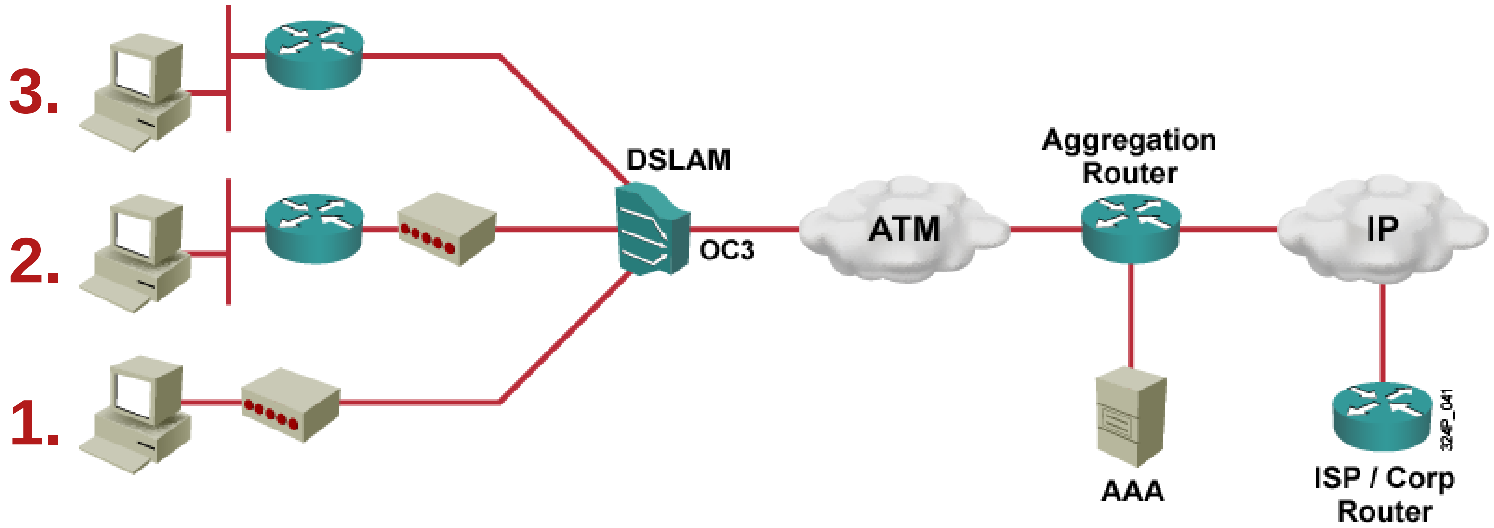
- IP packets in ADSL are carried in ATM cells
- Differences between ADSL data transfers rely in the way how IP packets are delivered to BRAS
- Three approaches
  1. **RFC 1483/2684 Bridged**
  2. **PPPoE** (PPP over Ethernet, switched solution)
  3. **PPPoA** (PPP over ATM, routed solution)

Why PPP?

- *Most ADSL users are commercial customers*
- Commerce is usually closely connected with problems like
  - Identification of user and following authentication with username and password
  - Independence on access to medium or DSL SL
  - Accounting of access and data delivery
- Plain transfer of IP packets in ATM cells doesn’t allow above features to be easily implemented
  - Access method behind RFC 1483/2684 Bridged
- Point-to-Point Protocol has all of those features
- Idea behind PPPoE or PPPoA is to encapsulate IP packets into PPP and then transfer them through ADSL network

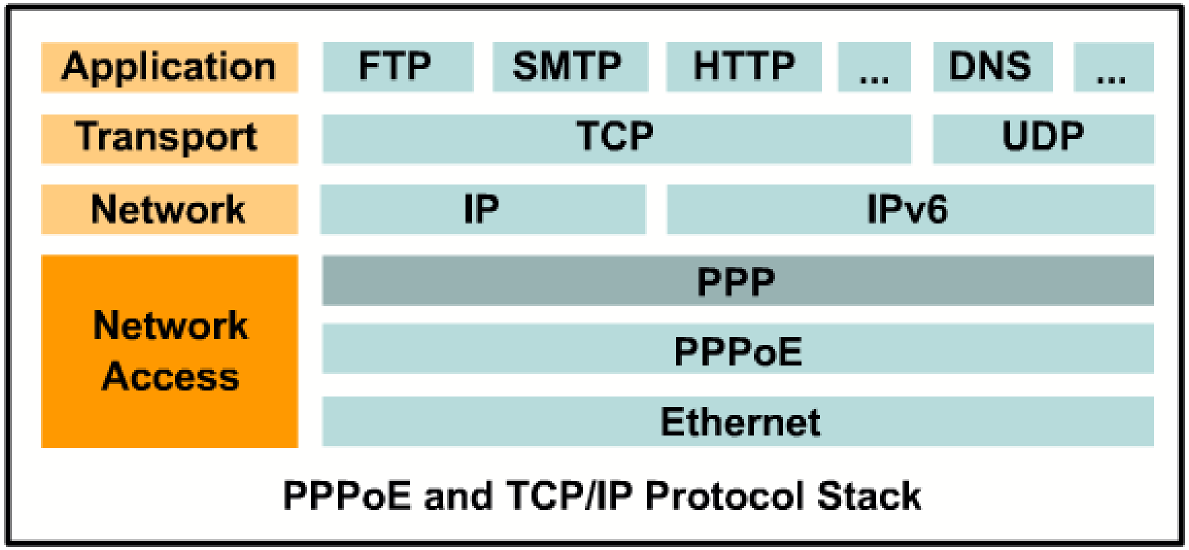
PPPoE

PPP over Ethernet (PPPoE)



- Options how to deploy PPPoE on DSL:
  1. On PC is PPPoE client SW installed, modem has Ethernet and DSL interface
  2. On PC is just TCP/IP, router has built-in PPPoE client inside, modem has Ethernet and DSL interface
  3. On PC is just TCP/IP, router has built-in DSL modem and PPPoE client

PPPoE Protocol Stack



- Encapsulation order in PPPoE (RFC 2516) is

  TCP/UDP → IP → PPP → PPPoE → Ethernet

- PPPoE defines helper 6B long header on L2 right between PPP and Ethernet
  - Purpose of PPPoE is to identify PPPoE session

PPPoE Variants

- PPPoE could be deployed on two interface types
  1. Ethernet
  2. ATM (DSL interface)

1. PPPoEoE (PPPoE over Ethernet)

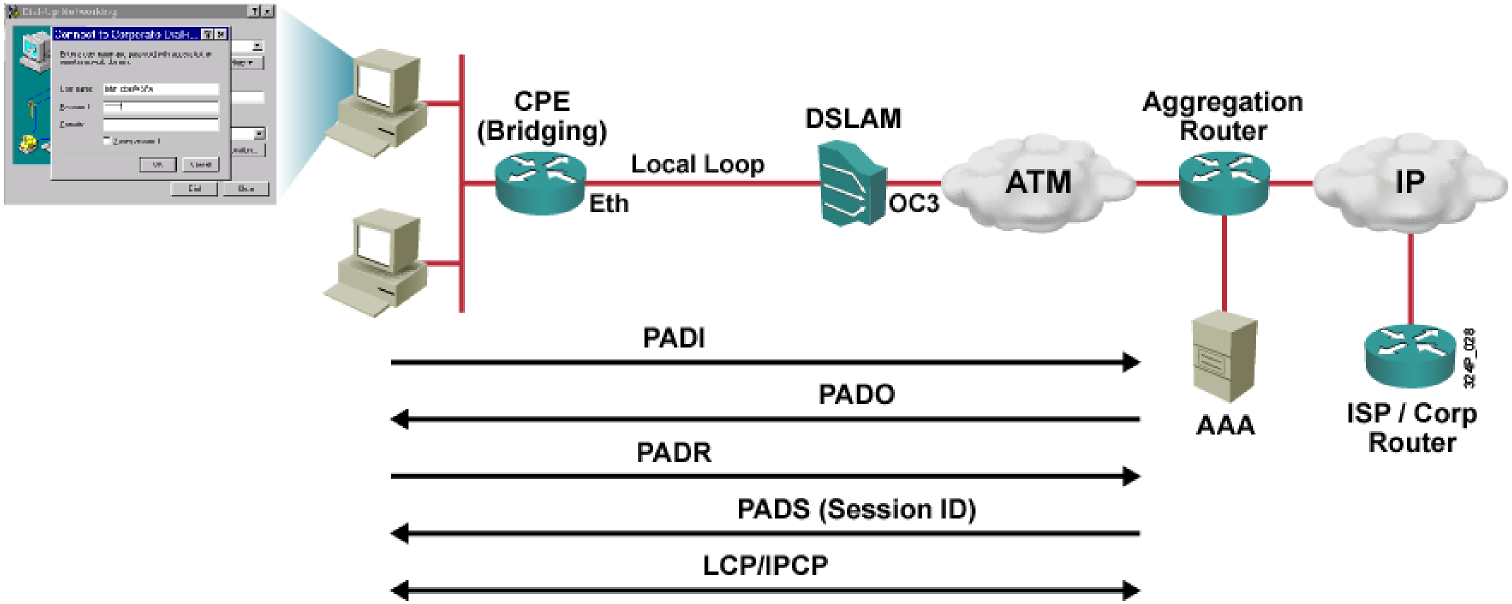
- Data encapsulated into PPP, PPPoE and Ethernet header are send out through Ethernet interface
- Typical option for SW PPPoE clients and for routers communicating with DSL modem through Ethernet interface

2. PPPoEoA (PPPoE over ATM)

- Data encapsulated into PPP, PPPoE and Ethernet header are send out through ATM interface
- Typical for routers with built-in DSL modem

- *DO NOT mistake previous two modes with PPPoA!!!*

Creating PPPoE Session

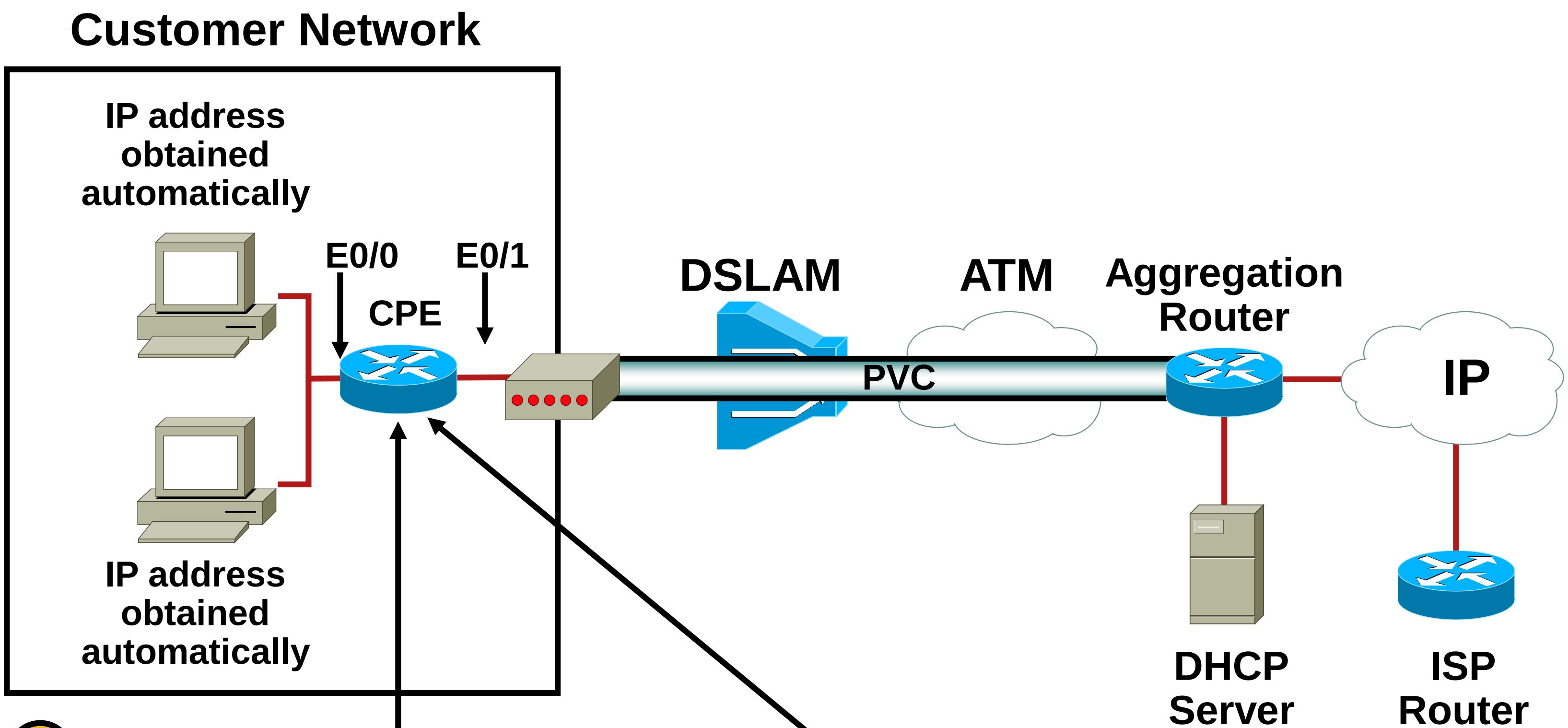


- **PPPoE Active Discovery Initiation**
  - Determine MAC address of BRAS
- **PPPoE Active Discovery Offer**
  - Response from BRAS consists of MAC address, name and other info
- **PPPoE Active Discovery Request**
  - Signing of client to BRAS
- **PPPoE Active Discovery Session-confirmation**
  - Session ID assignment to client
- **PPPoE Active Discovery Termination**
  - Tear down PPPoE session

Configuration Steps of PPPoE

- Not necessarily in that order:
  - Configuration of egress Ethernet/ATM interface (according to PPPoEoE or PPPoEoA)
  - Configuration of Dialer interface
  - Routing configuration
  - Decrementing MTU to 1492B on egress interface
  - Decrementing TCP MSS to 1452B on ingress interface
  - Configuration of “sideway things” (DHCP, NAT/PAT, firewall, ...)

Configuring PPPoEoE ①



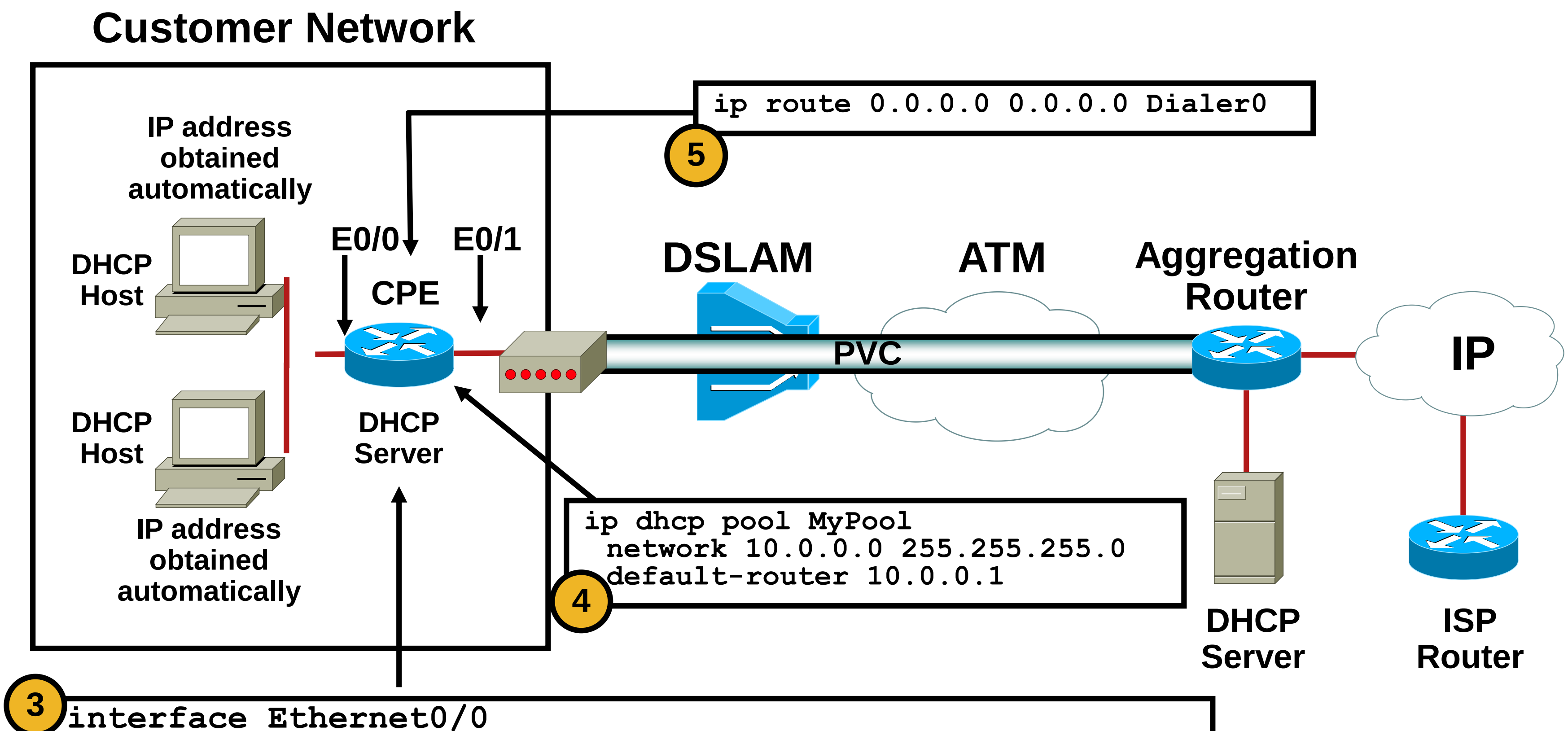
1

```
interface Ethernet0/1
   no ip address
   pppoe enable
   pppoe-client dial-pool-number 1
```

2

```
interface Dialer0
   ip address negotiated
   ip mtu 1492
   encapsulation ppp
    ppp chap hostname LOGIN
    ppp chap password PASS
   dialer pool 1
```

Configuring PPPoEoE ②




```
interface Ethernet0/0
   ip address 10.0.0.1 255.255.255.0
   ip nat inside
   ip tcp adjust-mss 1452
!
interface Dialer0
   ip nat outside
!
ip nat inside source list 101 interface Dialer0 overload
access-list 101 permit ip 10.0.0.0 0.255.255.255 any
```

Why Alter MTU and MSS? ①

- **MTU** = maximal allowed length of packet that could traverse through interface
- **MSS** = maximal allowed length of TCP segment
- It's important to notice, that…
  - PPP header itself has 2B
  - PPPoE header itself has 6B
  - This overhead takes from usual 1500B MTU 8B
- Therefore it's necessary on egress interface (Dialer) lower MTU on value 1492B
  - Command **`ip mtu 1492`** on interface
  - Packets exceeding this value will be fragmented

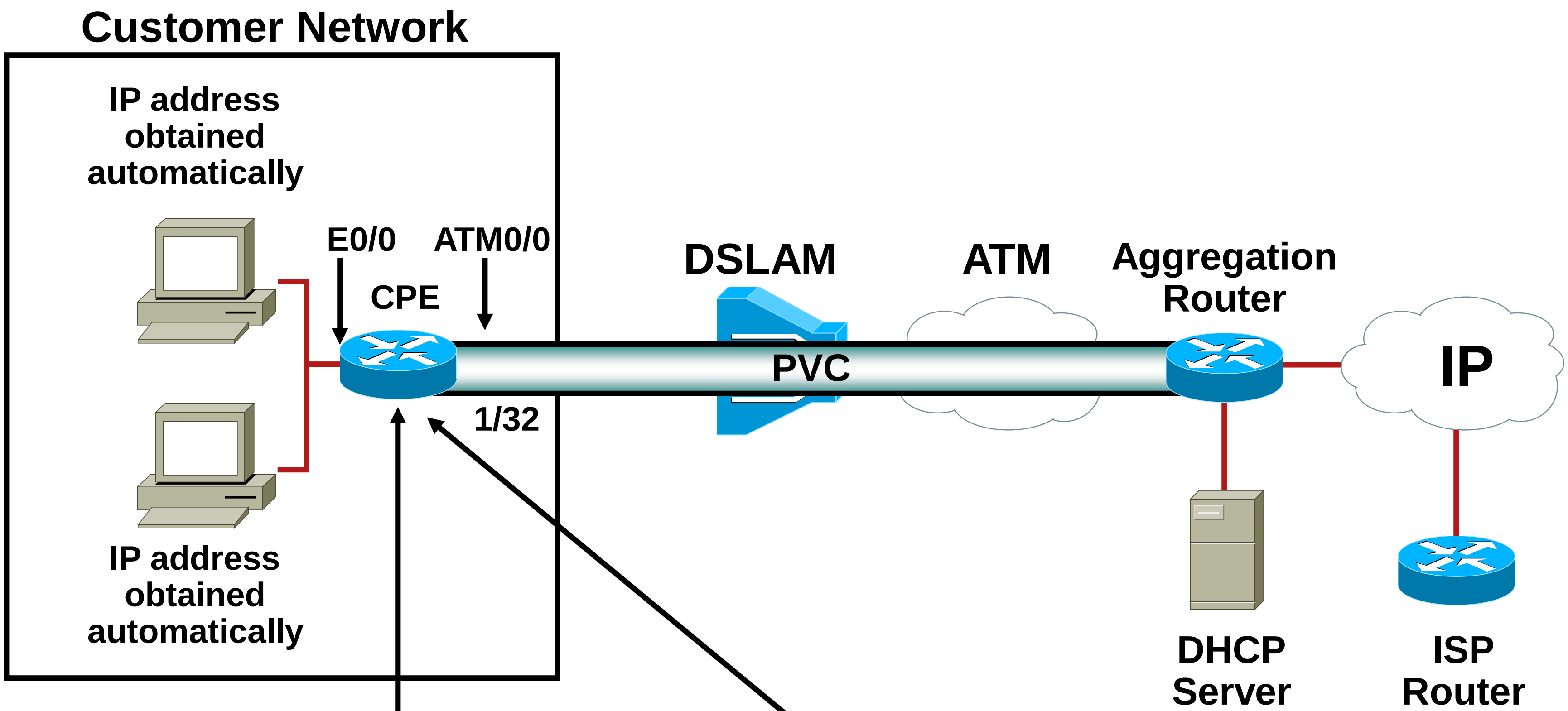
Why Alter MTU and MSS? ②

- Issue with length of TCP segment
  - Standard length of MSS is $MTU - 40B$ (20B for IPv4 header, 20B for TCP header)
  - Packets with maximally length 1460B would traverse through DSL router
  - When sending out those packets unnecessary fragmentation could take place
  - That increases risk of flapping or slow TCP connections
- Solution is to decrement MSS to value1452B on ingress interface
  - Command `ip tcp adjust-mss` causes rewriting of MSS field in all SYN packets

Final PPPoEoE Configuration

```
hostname CPE
!
ip dhcp pool MyPool
 network 10.0.0.0 255.0.0.0
 default-router 10.0.0.1
!
interface Ethernet0/1
 no ip address
 pppoe enable
 pppoe-client dial-pool-number 1
!
interface Ethernet0/0
 ip address 10.0.0.1 255.0.0.0
 ip nat inside
 ip tcp adjust-mss 1452
!
interface Dialer0
 ip address negotiated
 ip mtu 1492
 encapsulation ppp
 dialer pool 1
 ip nat outside
 ppp chap hostname mylogin
 ppp chap password mysecret
!
ip nat inside source list 101 interface Dialer0 overload
access-list 101 permit ip 10.0.0.0 0.255.255.255 any
!
ip route 0.0.0.0 0.0.0.0 Dialer0
```

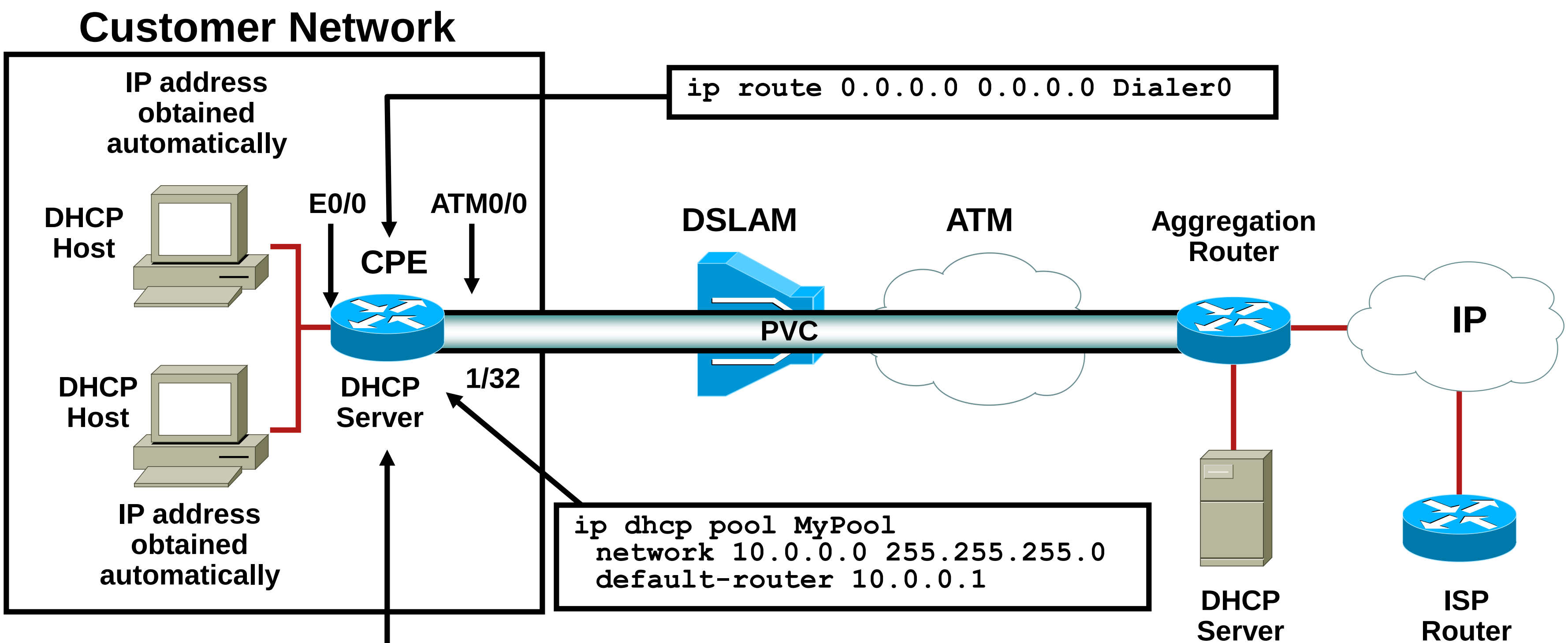
Configuring PPPoEoA ①




```
interface ATM0/0
  no ip address
  dsl operating-mode auto
  pvc 1/32
    pppoe-client dial-pool-number 1
```

```
interface Dialer0
   ip address negotiated
   ip mtu 1492
   encapsulation ppp
   dialer pool 1
```

Configuring PPPoEoA ②




```
ip route 0.0.0.0 0.0.0.0 Dialer0
```

```
ip dhcp pool MyPool
 network 10.0.0.0 255.255.255.0
 default-router 10.0.0.1
```

```
interface Ethernet0/0
   ip address 10.0.0.1 255.255.255.0
   ip nat inside
   ip tcp adjust-mss 1452
!
interface Dialer0
   ip nat outside
!
ip nat inside source list 101 interface Dialer0 overload
access-list 101 permit ip 10.0.0.0 255.255.255.0 any
```

Final PPPoEoA Configuration

```
hostname CPE
!
ip dhcp pool MyPool
 network 10.0.0.0 255.0.0.0
 default-router 10.0.0.1
!
interface ATM0/0
 no ip address
 dsl operating-mode auto
 pvc 1/32
  pppoe-client dial-pool-number 1
!
interface Ethernet0/0
 ip address 10.0.0.1 255.0.0.0
 ip nat inside
 ip tcp adjust-mss 1452
!
interface Dialer0
 ip address negotiated
 ip mtu 1492
 encapsulation ppp
 dialer pool 1
 ip nat outside
 ppp chap hostname mylogin
 ppp chap password mysecret
!
ip nat inside source list 101 interface Dialer0 overload
access-list 101 permit ip 10.0.0.0 0.255.255.255 any
!
ip route 0.0.0.0 0.0.0.0 Dialer0
```

PPPoEoE vs. PPPoEoA

- Difference is entirely in way how to configure egress interface, all other configuration steps are just same

PPPoEoE

```
interface Ethernet0/1
 no ip address
 pppoe enable
 pppoe-client dial-pool-number 1
```

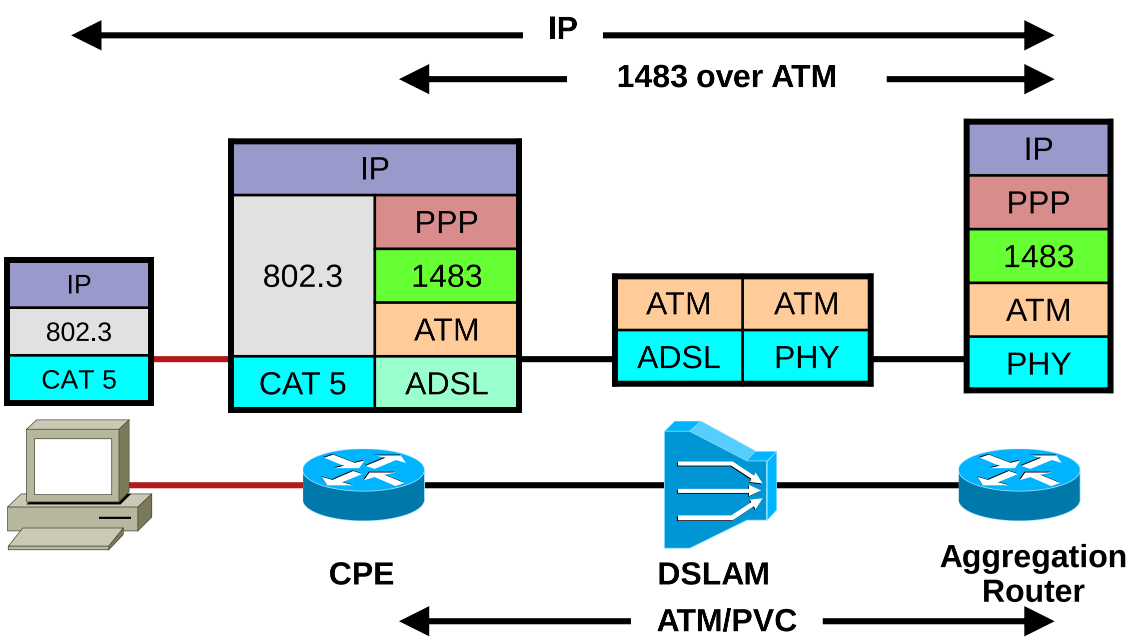
PPPoEoA

```
interface ATM0/0
 no ip address
 dsl operating-mode auto
 pvc 1/32
  pppoe-client dial-pool-number 1
```

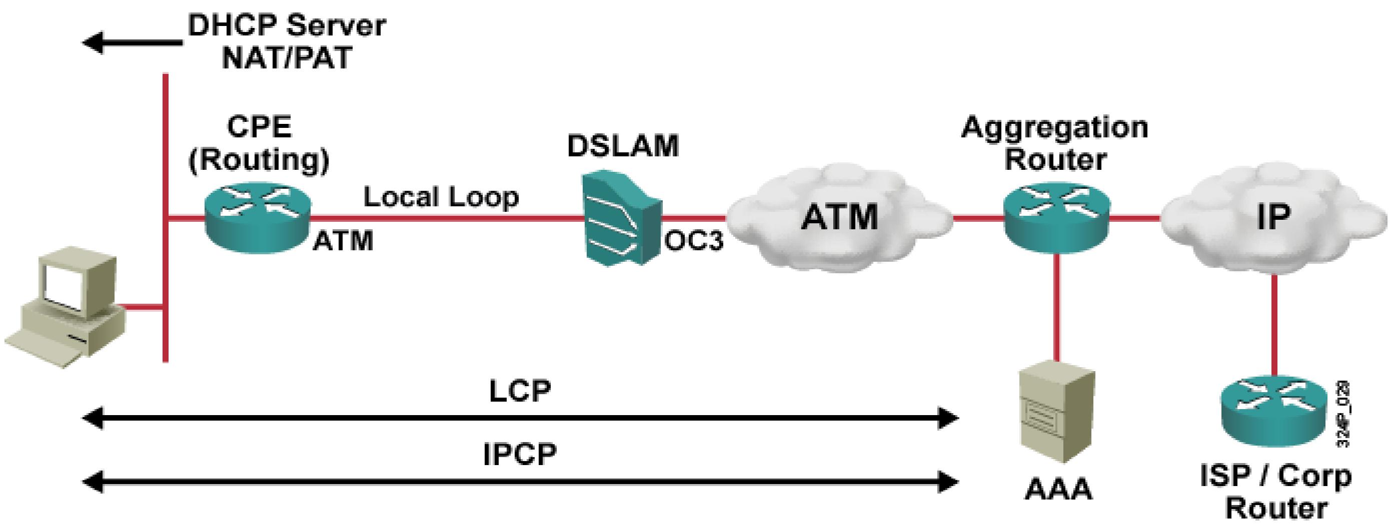
PPPoA

PPP over ATM (PPPoA)

- Simple approach – internal network is Ethernet, DSL network is ATM and between those two networks packets are routed
- PPP frames are transferred without PPPoE+Ethernet headers in ATM
- PPPoA session is created between router and BRAS

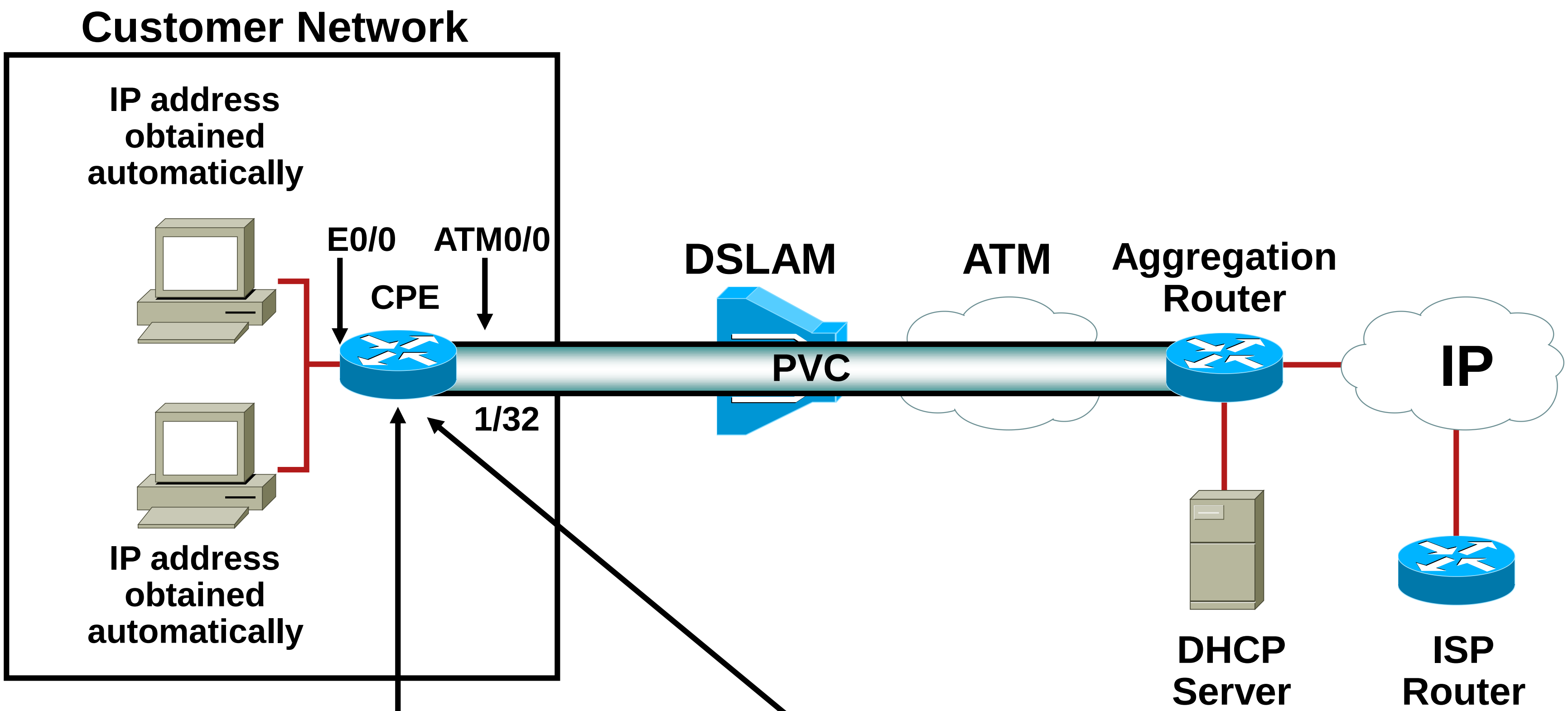


Creating PPPoA Session



- *There's no "helper" encapsulation*
- *There's no need to alter MTU and MSS*
- CPE creates PPP connection carried through ATM VC againt BRAS

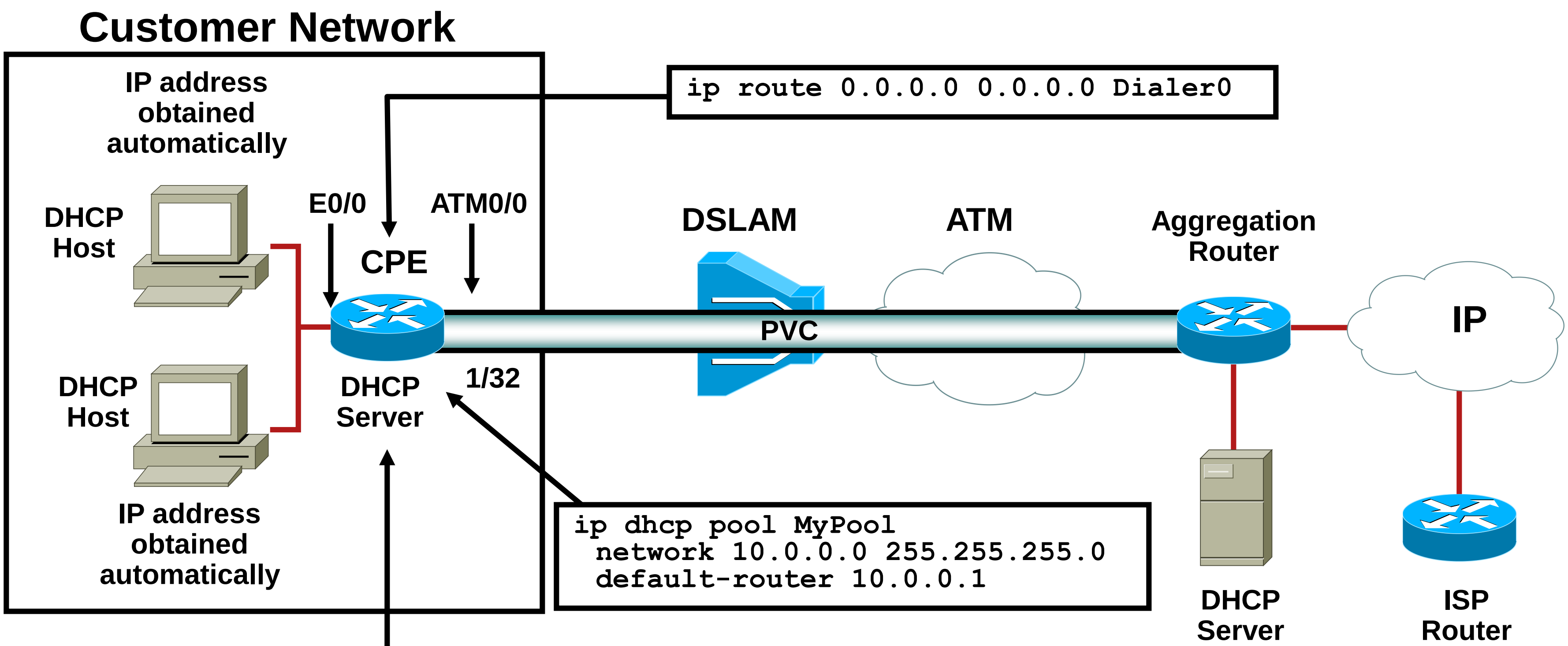
Configuring PPPoA ①




```
interface ATM0/0
  no ip address
  dsl operating-mode auto
  pvc 1/32
    encapsulation aal5mux ppp dialer
    dialer pool-member 1
```

```
interface Dialer0
   ip address negotiated
   encapsulation ppp
   dialer pool 1
```

Configuring PPPoA ②




```
interface Ethernet0/0
   ip address 10.0.0.1 255.255.255.0
   ip nat inside
!
interface Dialer0
   ip nat outside
!
ip nat inside source list 101 interface Dialer0 overload
access-list 101 permit ip 10.0.0.0 255.255.255.0 any
```

Final PPPoA Configuration

```
hostname CPE
!
ip dhcp pool MyPool
 network 10.0.0.0 255.0.0.0
 default-router 10.0.0.1
!
interface ATM0/0
 no ip address
 dsl operating-mode auto
  pvc 1/32
  encapsulation aal5mux ppp dialer
  dialer pool-member 1
!
interface Ethernet0/0
 ip address 10.0.0.1 255.0.0.0
 ip nat inside
!
interface Dialer0
 ip address negotiated
 encapsulation ppp
 dialer pool 1
 ip nat outside
 ppp chap hostname mylogin
 ppp chap password mysecret
!
ip nat inside source list 101 interface Dialer0 overload
access-list 101 permit ip 10.0.0.0 0.255.255.255 any
!
ip route 0.0.0.0 0.0.0.0 Dialer0
```

PPPoEoX vs. PPPoA

PPPoEoX

- Ethernet or ATM interface is assigned to Dialer with command **pppoe-client dialer-pool-number**
- On Dialer MTU is decremented to1492B
- On ingress interface TCP MSS is decremented to 1452B

```
interface Ethernet0
 no ip address
 pppoe enable
 pppoe-client dial-pool-number 1

interface ATM1
 no ip address
 pvc 1/32
  pppoe-client dial-pool-number 1
!
interface Dialer0
 ip mtu 1492
```

PPPoA

- Proper encapsulation is configured on ATM interface and this interface is assigned to dialer with command **dialer pool-member**

```
interface ATM0/0
 no ip address
 dsl operating-mode auto
 pvc 1/32
  encapsulation aal5mux ppp dialer
  dialer pool-member 1
```

Useful Commands

```
show pppoe session

show pppoe summary

show controllers dsl / show controllers atm

debug pppoe events

debug pppoe packets
```

Network Address Translation

Private Addresses

- *There is a limited number of public IPv4 addresses!*
  - RFC 1918, RFC 6598

RFC 1918

Block size	IP address range	number of addresses	largest CIDR block (subnet mask)
24-bits	`10.0.0.0 – 10.255.255.255`	16,777,216	10.0.0.0/8
20-bits	`172.16.0.0 – 172.31.255.255`	1,048,576	172.16.0.0/12
16-bits	`192.168.0.0 – 192.168.255.255`	65,536	192.168.0.0/16
22-bits	`100.64.0.1 – 100.127.255.254`	4,194,302	100.64.0.0/10

RFC 6598

- RFC 6598 addresses are used to address users CPEs to avoid collisions between ISP and user's private pool

Private Addresses in IPv6

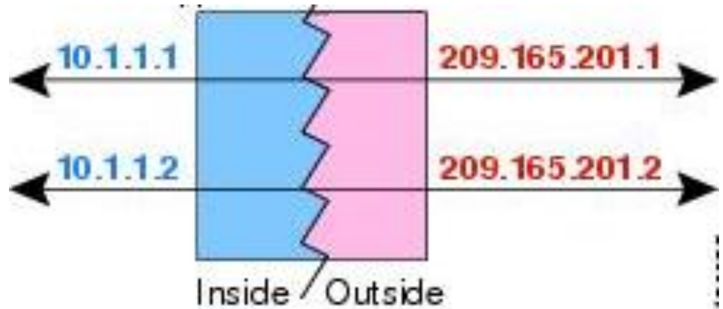
- RFC 4193 Unique Local addresses (ULA)

RFC 4193 Block	Flag	Global ID (random)	Subnet ID	Number of interface addresses in subnet
fc00::/7	L	xx:xxxx:xxxx	yyyy	18,446,744,073,709,551,616
	1	40 bits	16 bits	64 bits

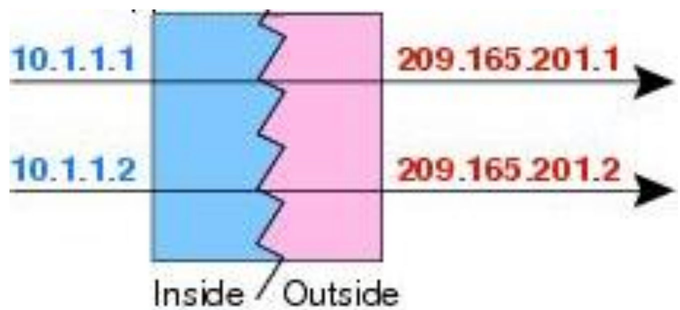
- ULA Prefix with L flag 0 is not defined, thus, only `fd00::/8` can be used
- 40 bits of Global ID MUST be random (hard to achieve in practice)

Cisco Variants

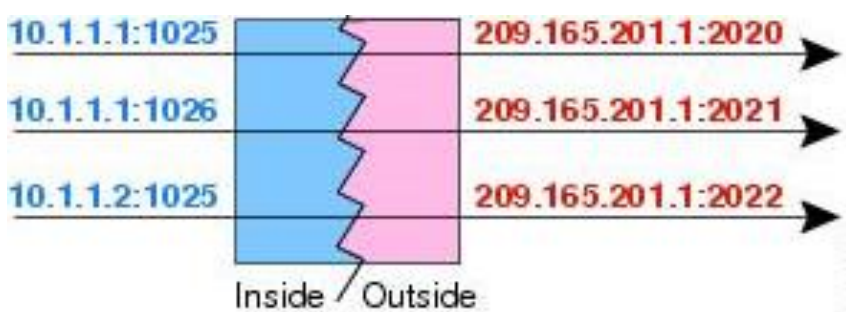
- **Source NAT** vs. **Destination NAT**
- **Static NAT**



- **Dynamic NAT**

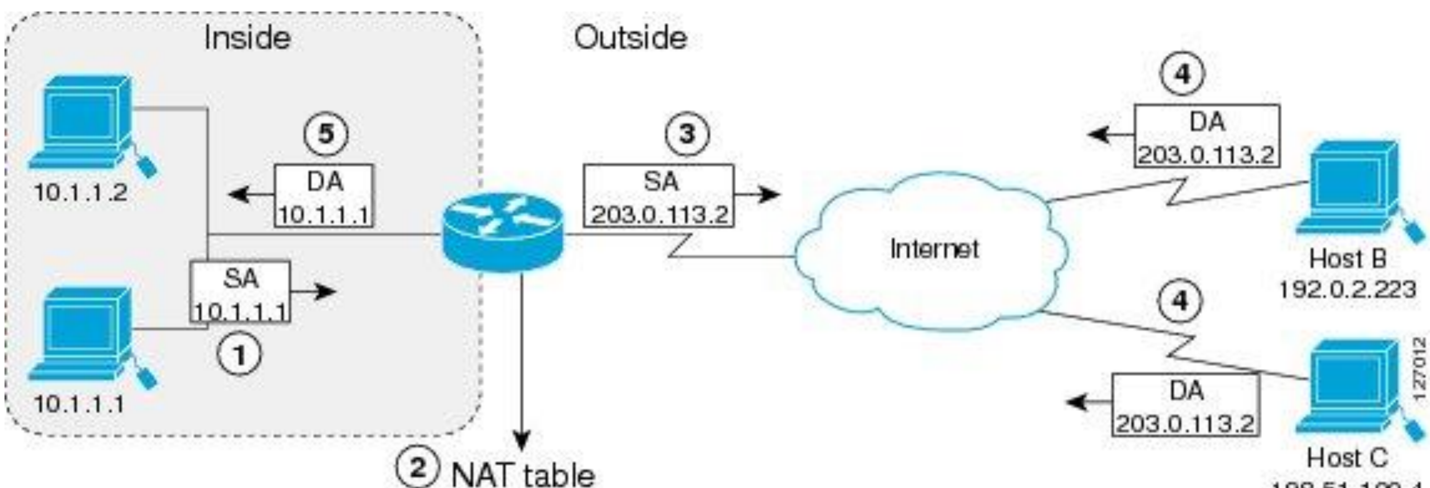


- **Dynamic PAT**



Inside/Outside vs. Local/Global

- Inside = behind the NAT
- Outside = in front of the NAT
- Local = *Loco* identifier
- Global = *Good* identifier



Protocol	Inside Local IP address:port	Inside Global IP address:port	Outside Global IP address:port	Outside Local IP address
TCP	10.1.1.2:1723	203.0.113.2:1723	198.51.100.4:23	198.51.100.4:23
TCP	10.1.1.1:1024	203.0.113.2:1024	192.0.2.223:23	192.0.2.223:23

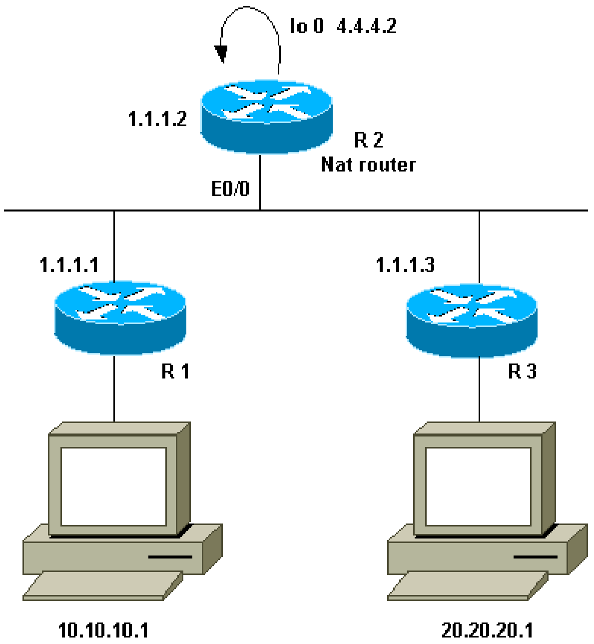
Configuring NAT Interfaces

- Marking inside / outside interfaces

```
(config-if)# ip nat {inside|outside}
```

- Agnostic NAT using NVI interface
  - NAT on-the-stick

```
(config-if)# ip nat enable
```



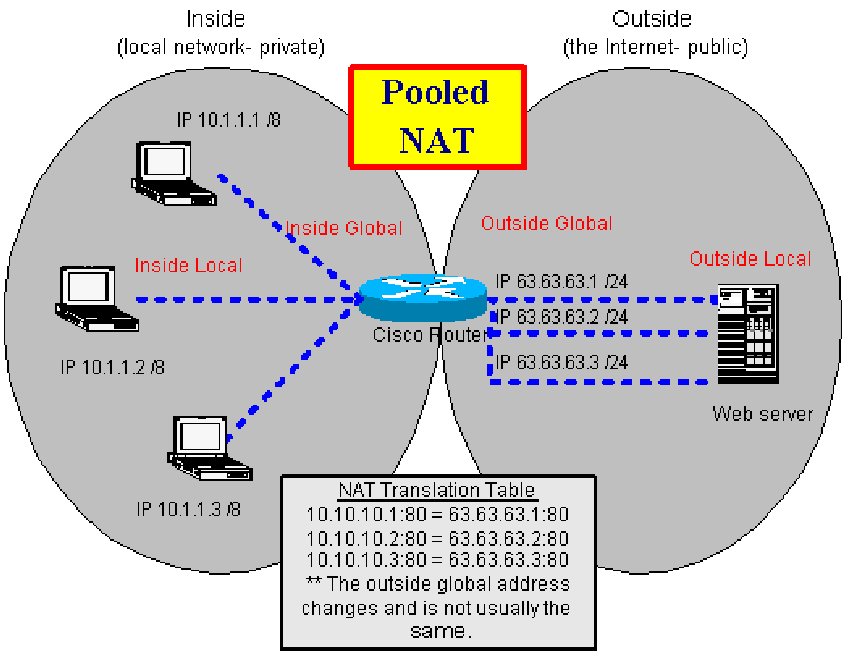
Configuring NAT Addresses

- Inside address range matched by pool

```
(config)# access-list {1-99} permit source source-wildcard
```

- Outside pool of addresses

```
(config)# ip nat pool poolname start-ip end-ip {netmask mask
  | prefix-length length}
```



Configuring NAT

- Static NAT

```
(config)# ip nat inside source local-ip global-ip
```

- Dynamic NAT

```
(config)# ip nat inside source list acl-ref pool pool-name
```

- Dynamic PAT
  - Single interface

```
(config)#
 ip nat inside source list acl-ref interface iface overload
```

  - Pool of addresses

```
(config)#
 ip nat inside source list acl-ref pool pool-name overload
```

Verifying NAT

```
router-6# show ip nat statistics
 Total active translations: 1 (1 static, 0 dynamic; 0 extended)
 Outside interfaces:
 Ethernet0, Serial2.7
 Inside interfaces:
 Ethernet1
 Hits: 5  Misses: 0
 Expired translations: 0
 Dynamic mappings:
 -- Inside Source
 access-list 7 pool test refcount 0
 pool test: netmask 255.255.255.0
 start 172.16.11.70 end 172.16.11.71
 type generic, total addresses 2, allocated 0 (0%), misses 0
```

```
Router# show ip nat translations

Pro    Inside global      Inside local       Outside local      Outside global
icmp   171.16.68.5:15     10.10.10.1:15      171.16.68.1:15     171.16.68.1:15
---    171.16.68.5        10.10.10.1           ---                ---
```

Tunneling

What Is Tunneling?

- Many times it's useful to create "illusion" of the new network above the existing one. Here are some motivations:
  - *Existing network doesn't recognize protocol which we would need to transfer across it or service we would like to use*
  - *We would like to use existing network as transport tool but we want it to be completely invisible from point of view of internal network*
  - *We need to interconnect multiple sites with potentially private IP address space*
  - *We don't trust existing network and we want to securely transfer data across it*
- **Tunneling** = technique where packet is reencapsulated into the new packet
  - Former packet becomes payload of new packet and therefore its content (L3 header) is not in attention of routers

Protocols Used in Tunneling

- **Passenger protocol**
  - *We would like to transfer datagrams of this protocol inside tunnel*
  - E.g. IPX, AppleTalk, IPv4, IPv6
- **Encapsulating/Tunneling protocol**
  - Header of this protocol is prepended before passenger protocol
  - It's used to identify passenger protocol and secure transmission with authentication, encryption, etc.
  - E.g. GRE, IPsec, L2F, PPTP, L2TP
- **Carrier protocol**
  - Existing network uses this protocol for transport and inside it encapsulating protocol is carried wrapped around passenger protocol
  - E.g. Frame-relay, ATM, Ethernet

Encapsulating Protocols

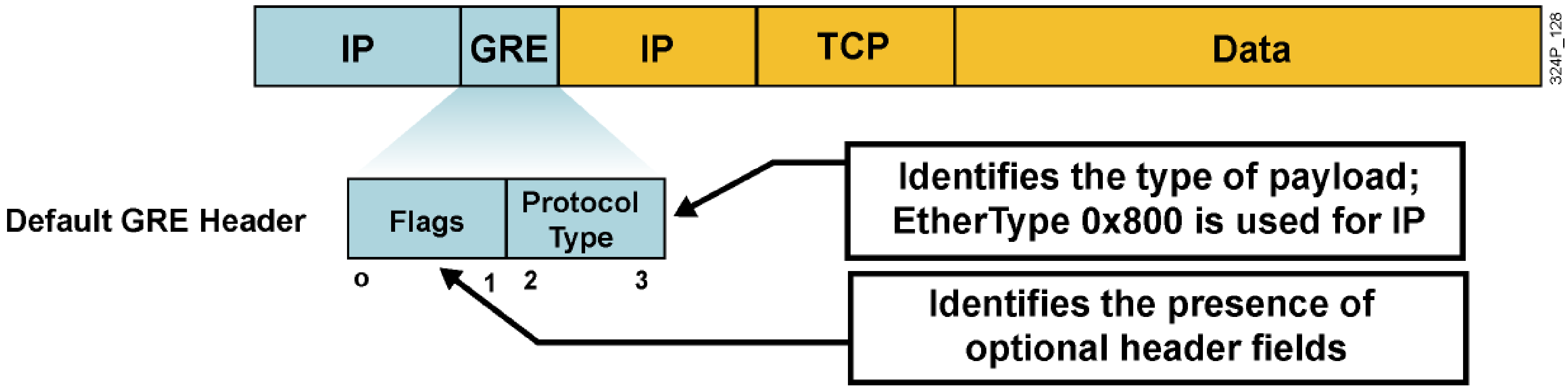
- Tunneling could be achieved with or even without support of encapsulating protocol
- **Tunneling WITH encapsulating protocol**
  - Encapsulating protocol wraps around original data and then is inserted into new packet in carrier protocol
  - Authentication support, multiple tunnels between same devices, encryption
  - More features means potentially more overhead
  - E.g. GRE, L2TP, PPTP
- **Tunneling WITHOUT encapsulating protocol**
  - Original packet is directly inserted into the new one
  - Limited support of advanced tunneling features
  - Minimal overhead
  - E.g. IP-in-IP, IPv6-in-IPv4

Generic Routing Encapsulation (GRE)

- **GRE** is encapsulating/tunneling protocol on L3
  - Supports multiple passenger protocols
  - Creates virtual point-to-point connection between pair of routers
  - Uses IP as carrier protocol
  - Allows transmission of multicast traffic (NBMA nature)
- GRE was originally invented by Cisco, but nowadays it's open standard specified in [RFC 2784]

GRE Header ①

- GRE is stateless without any signalization and traffic flow control
- GRE doesn't provide any security (no authentication, no encryption, no message integrity, no trustworthiness)
- Overhead of GRE tunnel is 24B (20B for new IPv4 header and 4B for GRE header)



GRE Header ②

- GRE **Flags** are stored in first 2B of header:
  - **Checksum Present (bit 0)**
  - **Key Present (bit 2)**
  - **Sequence Number Present (bit 3)**
  - **Version Number (bits 13–15)**: GRE has version 0, PPTP has version 1
- **Protocol Type** specify type of passenger protocol, usually it has same value as in field EtherType L2 Ethernet frame

Configuring GRE Tunnel

- GRE tunnels are represented by virtual **Tunnel interface** on router
- Tunnel interface must have:
  - Own IP address (just like any other interface)
  - IP address of sender and receiver of (carrier protocol) packets
  - Set proper tunneling mode
- Pair of Tunnel endpoint interfaces on opposite routers must met this criteria:
  - Tunnel endpoints own IP addresses must be in same network segment – *just like any other two directly interconnected interfaces*
  - IP addresses of sender and receiver must correspond on both endpoints – *IP address of receiver on one side must be IP address of sender on the opposite site and vice versa*
- Tunnel interface bandwidth is by default 9 Kbps
  - *Surprisingly it's recommended to change to reflect real situation* ☺

GRE Configuration


```
hostname Brno
!
interface Serial0/0
 ip address 192.3.4.5 255.255.255.0
 no shutdown
!
interface Tunnel0
 bandwidth 1000
 tunnel source s0/0
 tunnel destination 223.1.2.3
 tunnel mode gre ip ! OPTIONAL
 ip address 10.0.0.1 255.255.255.0
!
router ospf 1
 network 10.0.0.1 0.0.0.0 area 0
```

```
hostname Jesenik
!
interface Serial0/0
 ip address 223.1.2.3 255.255.255.0
 no shutdown
!
interface Tunnel7
 bandwidth 1000
 tunnel source s0/0
 tunnel destination 192.3.4.5
 tunnel mode gre ip ! OPTIONAL
 ip address 10.0.0.2 255.255.255.0
!
router ospf 1
 network 10.0.0.2 0.0.0.0 area 0
```

Tunnel Interface Status

- State „up, protocol up“ when using Tunnel interface for GRE is shown when following conditions are met:
  - Interface has defined source and destination addreses with commands **`tunnel source`** and **`tunnel destination`**
  - Actual interface with IP address specified in **`tunnel source`** is in state „up, protocol up“ – *source interface is working*
  - Route to address specified in **`tunnel destination`** must be present in routing table – *destination IP must be reachable according to router’s RT*
  - Whenever GRE Keepalive feature is enabled then opposite side of tunnel should be able to response to Keepalive packets – transport network should be able to deliver packets between tunnel endpoints

IPsec for Dummies

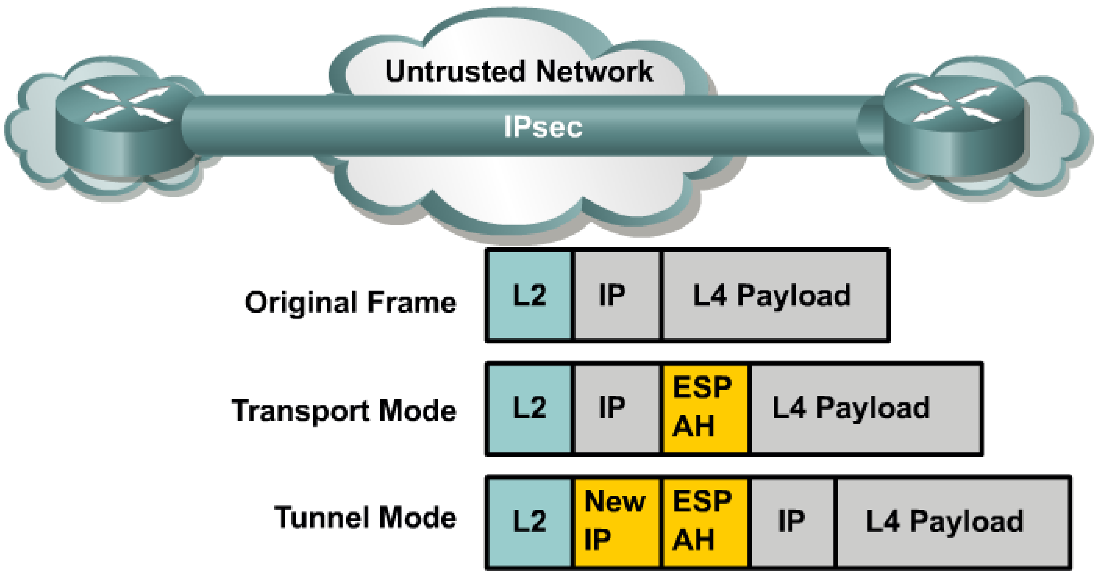
IP Security

- **IPsec** is series of IETF standards describing ways how to secure transmission of IP packet
- IPsec provides:
  - Data **confidentiality** – *Nobody can read it!*
  - Data **integrity** – *Nobody alter data as they traversed through network!*
  - Data **origin authentication** – *We know exactly and certainly who send it!*
  - **Anti-replay** protection
- IPsec uses 3 supporting protocols
  - **Internet Key Exchange (IKE)** for secure transfer of shared keys and NAT-T support (UDP ports 500 a 4500)
  - **Authentication Header (AH)** for sender authentication, data integrity and optional anti-replay protection
  - **Encapsulating Security Payload (ESP)** for data encryption, sender authentication, data integrity and optional anti-replay protection

AH and ESP

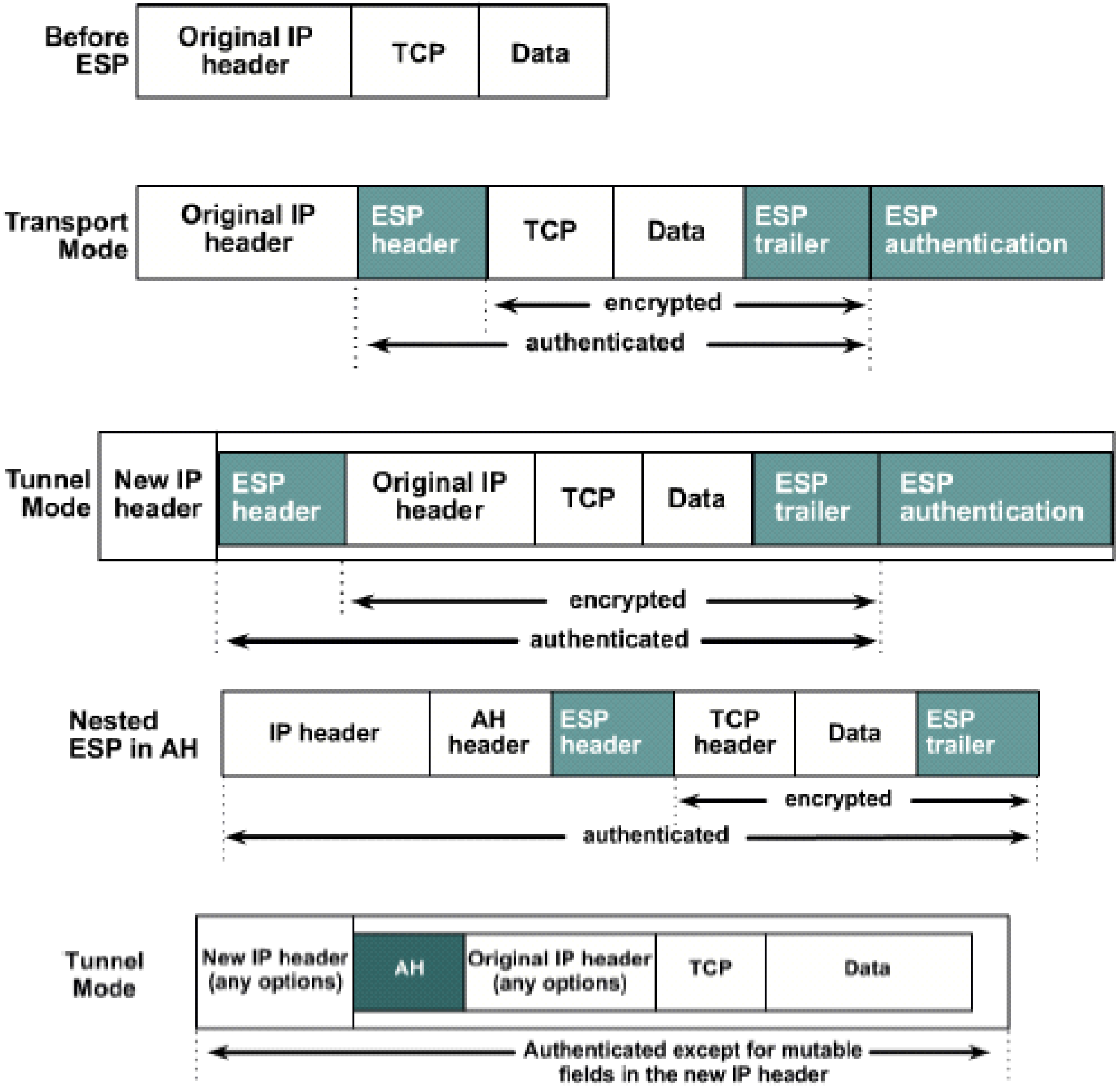
- AH protects packet payload and fixed IP header fields
  - Doesn’t provide encryption
  - Doesn’t like NAT (because NAT rewrites IP headers)
- ESP protects payload with encryption
  - Doesn’t secure IP header
  - Authentication is provided optionally
- AH is nowadays used rarely (even ASA doesn’t support AH), on the other hand ESP is used very often
- AH and ESP could be used simultaneously

IPsec Modes of Operation ①



- **Transport Mode**
  - Original header is used – routing is intact
  - Only the payload of the IP packet is usually encrypted and/or authenticated
  - Transport mode is used on Cisco router only when router itself is sender of packet
- **Tunnel Mode**
  - Entire IP packet is encrypted and/or authenticated
  - Adds new IP header

IPsec Modes of Operation ②



Security Association

- **Security Association (SA)** is a complete list of negotiated parameters between IPsec peers
- SA contains following operating information
  - *How is authentication of peers done?*
  - *In which operational mode should IPsec work?*
  - *Which algorithm and key is used for data encryption?*
  - *Which algorithm is used for data integrity?*
  - *How and how often should be keys replenished?*
- ISAKMP (IKE) is responsible for creation and maintenance SA between peers

IPsec Tunnel Creation

1. Host A sends interesting traffic to Host B.

2. Routers A and B negotiate an IKE Phase 1 session.

3. Routers A and B negotiate an IKE Phase 2 session.

4. Information is exchanged via the IPsec tunnel.

5. The IPsec tunnel is terminated.

IKE Phase 1 ①

- **IKE Phase 1** creates secure channel for IPsec peers authentication
- IKE Phase 1 has three steps
  1. Negotiating ISAKMP policies
  2. Diffie-Hellman key exchange
  3. Peers authentication

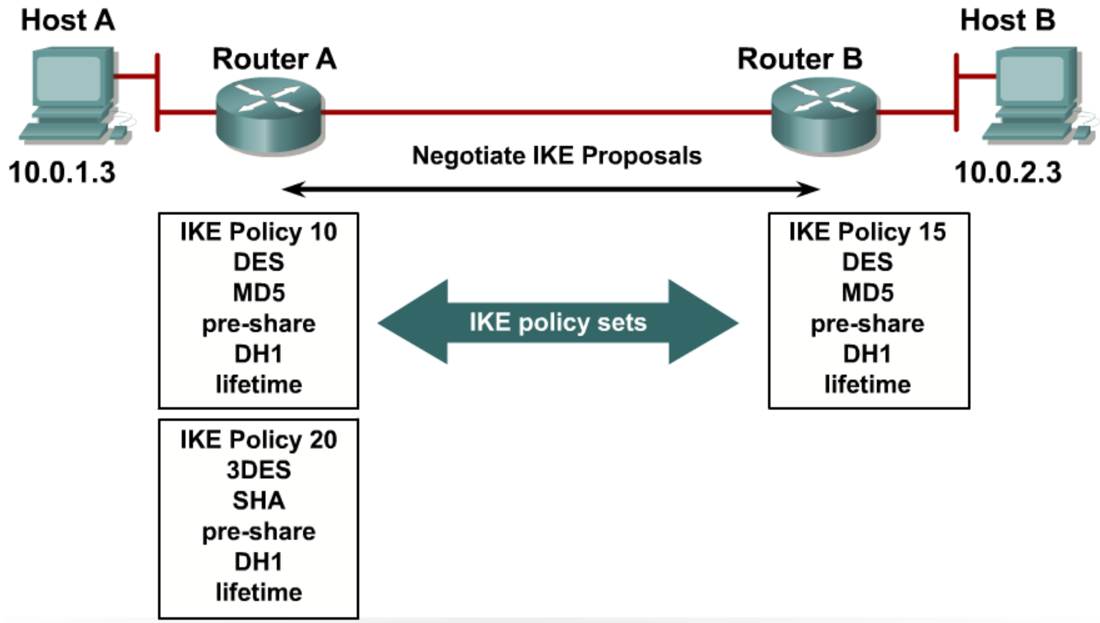
1. Negotiating ISAKMP policies
   - *Which encryption algorithm to use?*
   - *Which hashing algorithm to use?*
   - *Which Diffie-Hellman group?*
   - *How to authenticate peer?*

3. Peers authentication
   - According to way negotiated in Step 1 (pre-shared, RSA nonce, RSA signature)

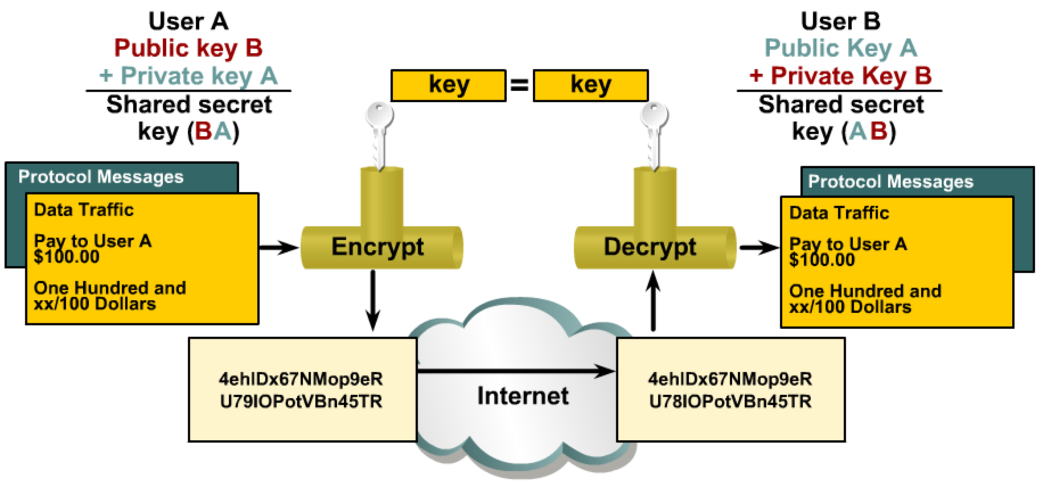
- Properties of IPsec tunnel itself are not negotiated, this is done in IKE Phase 2

IKE Phase 1 ②

1. Step



2. Step



IKE Phase 2

- In **IKE Phase 2** properties of IPsec tunnel between peers are negotiated
  - *Which protocol to use – AH, ESP, AH+ESP?*
  - *Which encryption algorithm to use?*
  - *Which hashing algorithm to use?*
  - *Which operational mode of IPsec to use?*
  - *What are keys used for encryption and decryption?*
  - *What is a lifetime of this negotiated properties?*
- First four properties are called **transform set (TS)**

Configuring of IPsec

- Successful implementation of IPsec consists of:
  - Create at least one ISAKMP policy for IKE Phase 1
  - Create at least one transform set for IKE Phase 2
  - Create crypto map and ACL describing interesting traffic
  - Apply crypto map on egress interface

Configuring ISAKMP Policy

```
crypto isakmp policy 1
 encr 3des
 hash md5
 authentication pre-share
 group 2
 lifetime 3600
 exit
crypto isakmp key 0 HESLO addr 192.0.2.254
```

- Encryption: 3DES
  - Alternatives: DES, AES
- Hash: MD5
  - Alternatives: SHA
- Authentication: pre-shared key
  - Alternatives: RSA based
- DH group: 2 (1024b)
  - Alternatives: 1 (768b), 5 (1536b)
- Lifetime: 3600s
- Pre-shared key for peer with address 192.0.2.254

Configuring Transform Set

- Transform set defines
  - Usage of AH and/or ESP
  - Encryption algorithm and length of key
  - Hashing algorithm
  - Operational mode – either transport or tunnel
- Transform sets are identified by their names
- Configuration snippet:

```
crypto ipsec transform-set AH-ESP-3DES-SHA ah-sha-hmac esp-3des

crypto ipsec transform-set ESP-AES-SHA esp-aes 256 esp-sha-hmac
```

Creation of Crypto Map

- Crypto map bonds together:
  - IPsec peers
  - ACL matching exact traffic to be encrypted between those peers
  - Target transform set to use
  - Way how to exchange keys (either manually or with support of ISAKMP)
  - Lifetime of SA and keys
- Crypto map must minimally consists of…
  - Peer definition
  - ACL reference
  - Transform set reference
- ACL defines which packets should be passed through IPsec tunnel
  - Usually statement in form "*from our network to peer's network*"
  - Stay away from usage statement with `any`!

Configuring Crypto Map

- Example:
  - Block 10: Lifetime is 10 000 KB or 1800 s, two alternative TS, newly generated keys with DH group 5
  - Block 20: Minimal configuration with one TS

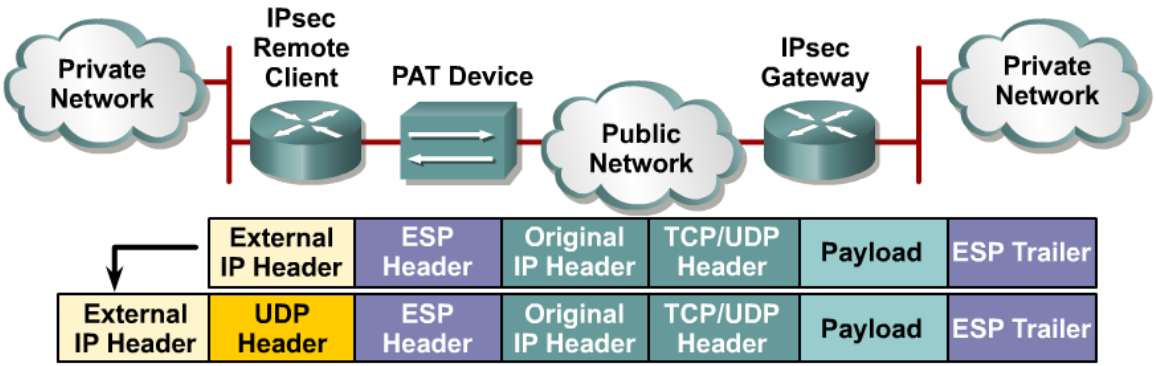
```
crypto map CM-S0/0 10 ipsec-isakmp ! Prvý blok, číslo 10
 set peer 1.2.3.4
 set security-association lifetime kilobytes 10000
 set security-association lifetime seconds 1800
 set transform-set ESP-AES-SHA AH-ESP-3DES-SHA
 set pfs group5
 match address 100

crypto map CM-S0/0 20 ipsec-isakmp ! Druhý blok, číslo 20
 set peer 5.6.7.8
 set transform-set ESP-AES-SHA
 match address 110

interface Serial 0/0
 crypto map CM-S0/0
```

Final IPsec Configuration

```
crypto isakmp policy 1
 encr 3des
 hash md5
 authentication pre-share
 group 2
 lifetime 3600

crypto isakmp key 0 prd31 address 192.0.2.254

crypto ipsec transform-set ESP-AES-SHA esp-aes 256 esp-sha-hmac

crypto map CM-S0/0 10 ipsec-isakmp
 set peer 192.0.2.254
 set transform-set ESP-AES-SHA
 match address 100

crypto map CM-S0/0 local-address Lo0 ! Loopback addresses peering

access-list 100 permit ip 10.0.0.0 0.255.255.255 192.168.10.0 0.0.0.255

int s0/0
 crypto map CM-S0/0
```

Final Notes

- ESP has huge issue with NAT hence **NAT-T (NAT Traversal)** was specified in [RFC 3715]
- NAT-T must be allowed through any firewall
  - UDP/500
  - UDP/4500



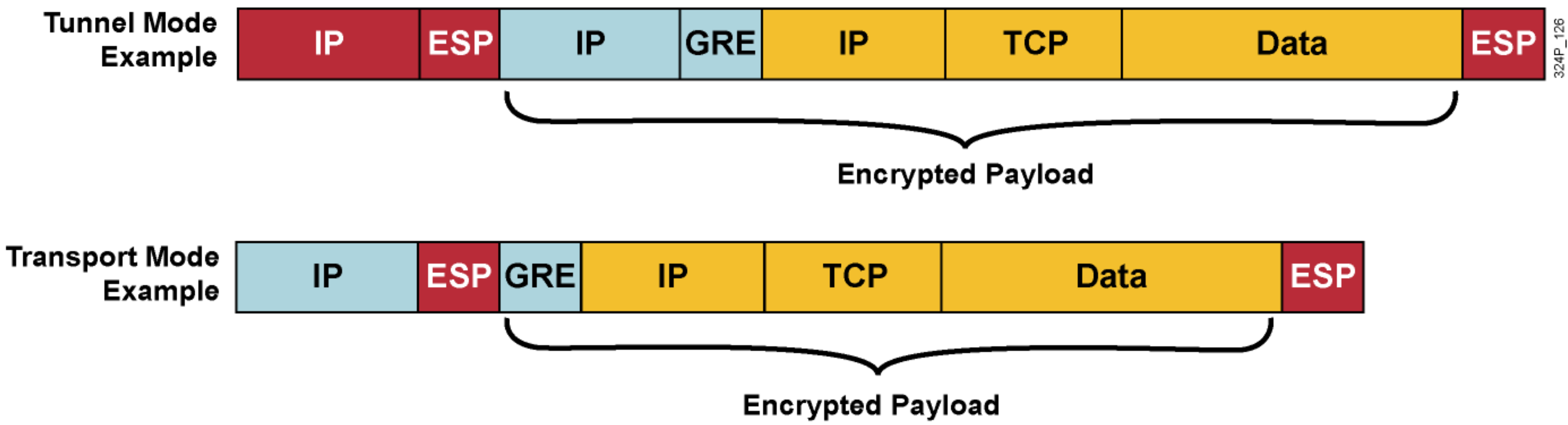
- For mobile clients is recommended new technology SSLVPN instead of robust and technologically complicated IPsec

Secure GRE Tunnels

IPsec + GRE?

- On the one hand IPsec is great for secure transfer of data, BUT…
  - Supports only IP
  - Older IOSes don’t support IPsec + multicast
  - Until recent only way how to configure IPsec was with cryptomap on egress interface hence it wasn’t possible to…
    - …create Tunnel interface representing IPsec tunnel
    - …activate routing protocol above this tunnel
- On the other hand GRE is great tunneling protocol, unfortunately its security features are terrible
- Solution = secure GRE tunnels with IPsec

IPsec + GRE!



- GRE could use IPsec in either tunnel or transport mode
  - Transport mode is efficient – saves 20 B on additional IP header per every packet
- Encapsulation order:
  - Passenger protocol → GRE → IPsec → IP

Configuration Ways ①

- Two existing ways
  - **Crypto maps** (in every IOS)
  - **IPsec profiles** applied directly on Tunnel interface (only in newer IOSes)
- Crypto map way for IPsec + GRE is similar to IPsec crypto map only...
  - ...however it's necessary to be aware of fact that egress interface transmits GRE packets instead of "plain IP packets"
  - Command **`set peer`** in crypto map must match with IP address in **`tunnel destination`** command on Tunnel interface
  - ACL in crypto map must match GRE packet type with source and destination address referred in **`tunnel source`** resp. **`tunnel destination`** commands
  - 12.2(13)T and older IOSes must have crypto map applied both on Tunnel interface and also egress interface (Document ID 14125)
  - All other configuration steps are similar

Crypto Map Configuration Example


```
hostname Jesenik
!
crypto map KRYPTUJ 1 ipsec-isakmp
 match address 150
 set transform-set TS
 set peer 223.1.2.3
!
interface Serial0/0
 ip address 192.3.4.5 255.255.255.0
 crypto map KRYPTUJ
!
interface Tunnel0
 tunnel source s0/0
 tunnel destination 223.1.2.3
 ip address 10.0.0.1 255.255.255.0
 crypto map KRYPTUJ ! Unnecessary on 12.2(13)T and newer
!
access-list 150 permit gre host 192.3.4.5 host 223.1.2.3
```

Configuration Ways ②

- Newer IOSese support more convenient IPsec profile way with use of:
  - IPsec profiles
  - Tunnel interface command **`tunnel protection`**
- IPsec profile is simplified version of crypto map
  - without **`match address`** for ACL
  - without **`set peer`**
- On Tunnel interface exists command **`tunnel protection`** that bonds tunnel with IPsec profile
- With this configuration way there's no need to create crypto map and ACL
  - All other configuration steps are still necessary – defining ISAKMP policies, pre-shared password, transform-sets
  - Be aware – GRE Keepalives feature isn't supported when using IPsec profiles (Document ID 64565)

IPsec Profiles Configuration Example


```
hostname Jesenik
!
crypto ipsec profile KRYPTUJ
 set transform-set TS
!
interface Serial0/0
 ip address 192.3.4.5 255.255.255.0
!
interface Tunnel0
 tunnel source s0/0
 tunnel destination 223.1.2.3
 tunnel protection ipsec profile KRYPTUJ
 ip address 10.0.0.1 255.255.255.0
```

Useful Commands

```
show crypto ipsec
show crypto session

clear crypto session

debug crypto isakmp
debug crypto ipsec
```

Context-Based Access Control

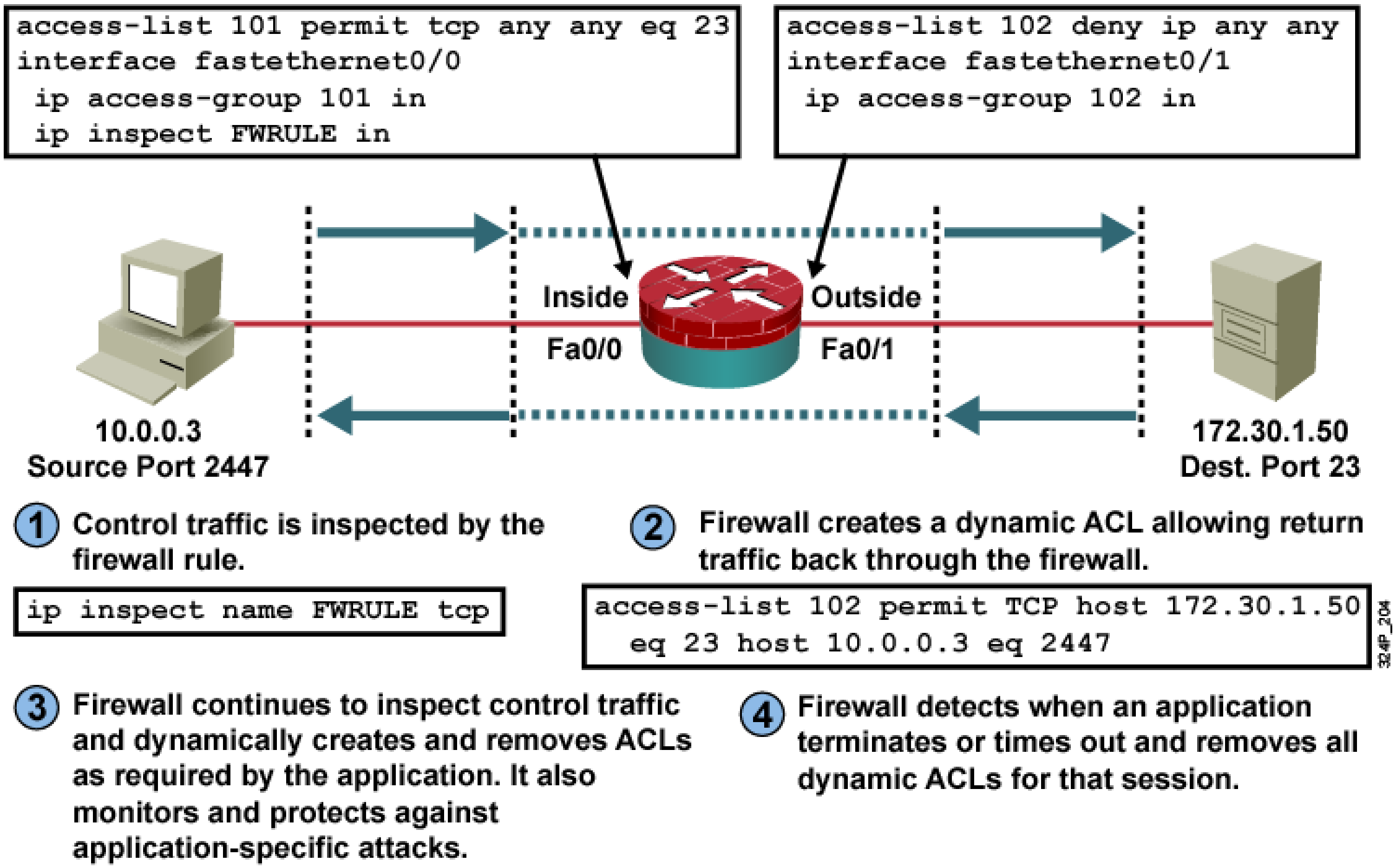
Cisco IOS ACLs Revisited

- Firewall can be implemented using ACLs
- ACLs provide traffic filtering by these criteria:
  - Source and destination IP addresses
  - Source and destination ports
- Using ACLs for firewall filtering can lead to several shortcomings:
  - Ports opened permanently to allow traffic, creating a security vulnerability.
  - The ACLs do not work with applications that negotiate ports dynamically.
- Cisco IOS Firewall addresses these shortcomings of ACLs.

IOS Firewall

- Stateful firewall: **Context-Based Access Control (CBAC)** or **IP Inspect**
- Example of usage:
  - Create set of protocols to analyze with IP Inspect
  - Apply the set to the appropriate interface and traffic direction
    - Direction from trusted to untrusted network (new connections go from trusted networks)
  - Apply extended ACL **`deny ip any any`** to inbound direction on egress interface
- IP Inspect maintains session table to check “returning” traffic.
  - `IF` match is found `THEN` traffic is bypassed through ACL
- All packets inspected by CBAC are processed switch – huge impact on router’s performance

IP Inspect Example



The `ip inspect name` Parameters

Parameter	Description
inspection-name	Names the set of inspection rules (protocols). The set can consist of several protocols.
protocol	The protocol to inspect.
alert {on | off}	(Optional) For each inspected protocol, the generation of alert messages can be set to on or off. If no option is selected, alerts are generated based on the setting of the ip inspect alert-off command.
audit-trail {on | off}	(Optional) To turn on CBAC audit trail messages, which will be displayed on the console after each CBAC session closes. Example: %FW-6-SESS_AUDIT_TRAIL: tcp session initiator (192.168.1.13:33192) sent 22 bytes -- responder (192.168.129.11:25) sent 208 bytes
timeout *seconds*	(Optional) Specify the number of seconds for a different idle timeout to override the global TCP or UDP idle timeouts for the specified protocol. This timeout overrides the global TCP and UDP timeouts but does not override the global Domain Name Service (DNS) timeout. To specify the DNS idle timeout use ip inspect dns-timeout command in global configuration mode.

Configuration

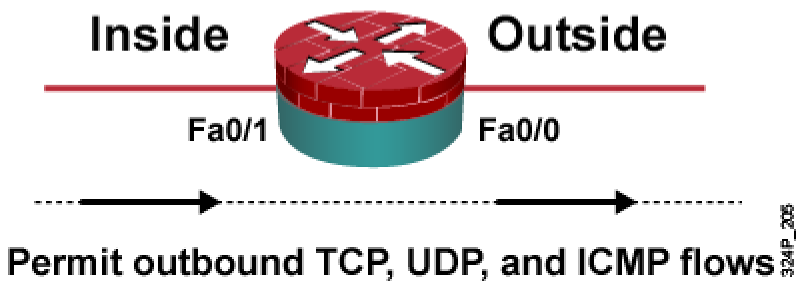
- Inspection rules are applied on the interface where traffic initiates
  - ACL can also permits only wanted traffic
  - On all other interfaces, apply the ACL in the inward direction that denies all traffic

```
Router(config-if)# ip inspect inspection-name {in | out}
```

Parameter	Description
inspection-name	Names the set of inspection rules
in	Applies the inspection rules to inbound traffic
out	Applies the inspection rules to outbound traffic

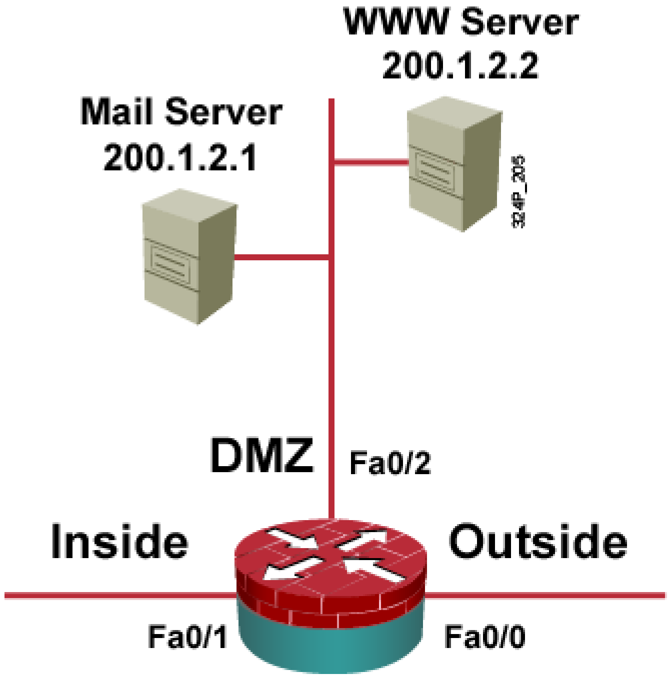
Example: CBAC for Two Interfaces Scenario

```
ip inspect name OUTBOUND tcp
ip inspect name OUTBOUND udp
ip inspect name OUTBOUND icmp
!
interface FastEthernet0/0
 ip access-group OUTSIDEACL in
!
interface FastEthernet0/1
 ip inspect OUTBOUND in
 ip access-group INSIDEACL in
!
ip access-list extended OUTSIDEACL
 permit icmp any any packet-too-big
 deny ip any any log
!
ip access-list extended INSIDEACL
 permit tcp any any
 permit udp any any
 permit icmp any any
```



Example: CBAC for Three Interfaces Scenario

```
interface FastEthernet0/0
 ip inspect OUTSIDE in
 ip access-group OUTSIDEACL in
!
interface FastEthernet0/1
 ip inspect INSIDE in
 ip access-group INSIDEACL in
!
interface FastEthernet0/2
 ip access-group DMZACL in
!
ip inspect name INSIDE tcp
ip inspect name OUTSIDE tcp
!
ip access-list extended OUTSIDEACL
 permit tcp any host 200.1.2.1 eq 25
 permit tcp any host 200.1.2.2 eq 80
 permit icmp any any packet-too-big
 deny ip any any log
!
ip access-list extended INSIDEACL
 permit tcp any any eq 80
 permit icmp any any packet-too-big
 deny ip any any log
!
ip access-list extended DMZACL
 permit icmp any any packet-too-big
 deny ip any any log
```



Verifying IP Inspect Firewall

```
show ip inspect name inspection-name
show ip inspect config
show ip inspect interfaces
show ip inspect session [detail]
show ip inspect statistics
show ip inspect all
```

- Displays inspections, interface configurations, sessions, and statistics

```
Router# show ip inspect session
Established Sessions
 Session 6155930C (10.0.0.3:35009)=>(172.30.0.50:34233) tcp SIS_OPEN
 Session 6156F0CC (10.0.0.3:35011)=>(172.30.0.50:34234) tcp SIS_OPEN
 Session 6156AF74 (10.0.0.3:35010)=>(172.30.0.50:5002) tcp SIS_OPEN
```

IP Inspect Reports

```
Router(config)#
ip inspect audit-trail
```

- Summary messages about monitored protocols

```
Router(config)#
no ip inspect alert-off
```

- Realtime alerts about suspicious activity in monitored protocols, disabled by default

```
Router(config)# logging on
Router(config)# logging host 10.0.0.3
Router(config)# ip inspect audit-trail
Router(config)# no ip inspect alert-off
```

Troubleshooting IP Inspect

- Cisco IOS Firewall Design Guide
  - http://www.cisco.com/en/US/prod/collateral/vpndevc/ps5708/ps5710/ps1018/product_implementation_design_guide09186a00800fd670.html

```
debug ip inspect function-trace
debug ip inspect object-creation
debug ip inspect object-deletion
debug ip inspect events
debug ip inspect timers
debug ip inspect detail

debug ip inspect protocol
```

Since IOS 12.4(20)T the command **`debug ip inspect`** is replaced by **`debug policy-firewall`**

Slides adapted by Vladimír Veselý and Matěj Grégr
partially from official course materials
but the most of the credit goes to CCIE#23527 Ing. Peter Palúch, Ph.D.

Last update: 2016-03-28