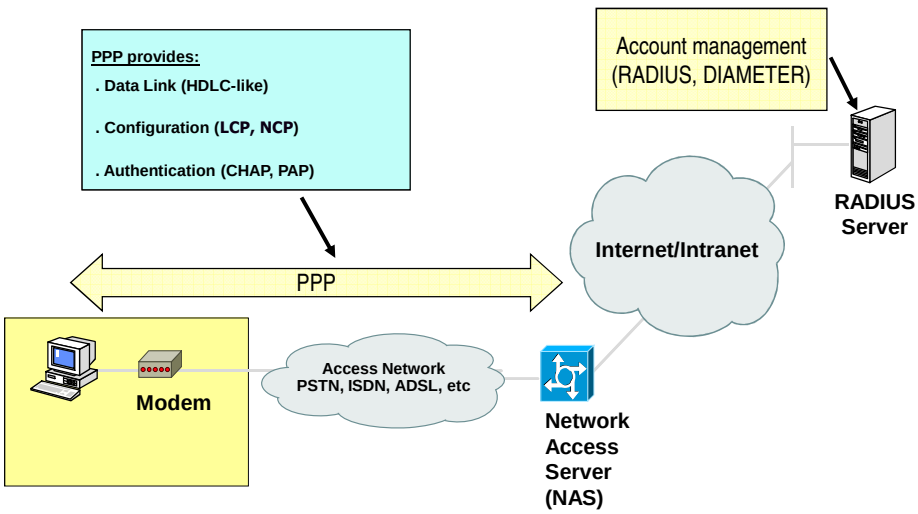
Lecture 1.1:

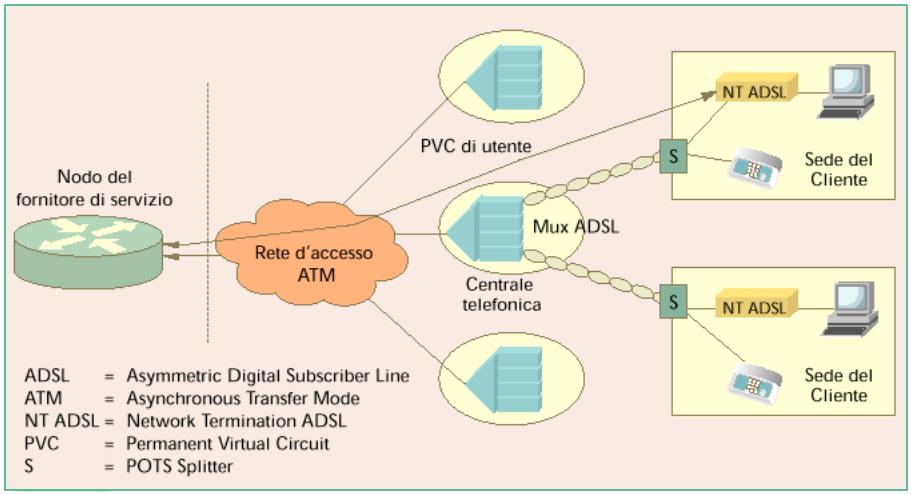
Point to Point Protocol (PPP)
An introduction

"*the watchword for a point-to-point protocol should be simplicity*"
(RFC 1547, PPP requirements).
… disattended by 50+ RFCs …

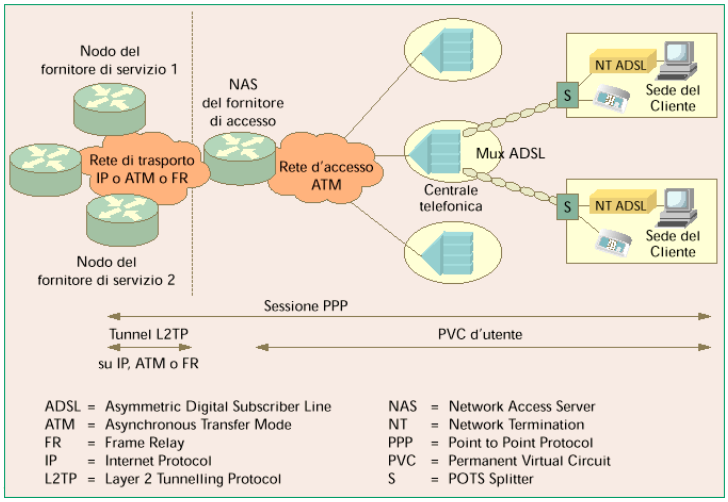
Recommended reading: PPP specification, RFC 1661 + 1662 (STD 51), July 1994

PPP: where (access)





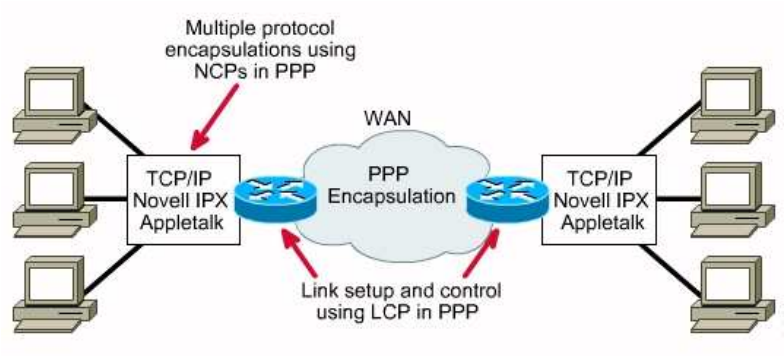
Accesso diretto al nodo del fornitore di servizio



Accesso al nodo del fornitore di servizio mediante NAS (Network Access Server) del fornitore di accesso

PPP: where (WAN)

➔ **Interconnection of small networks (e.g. branch offices) may be possible through a dedicated or dial-on-demand WAN connection**

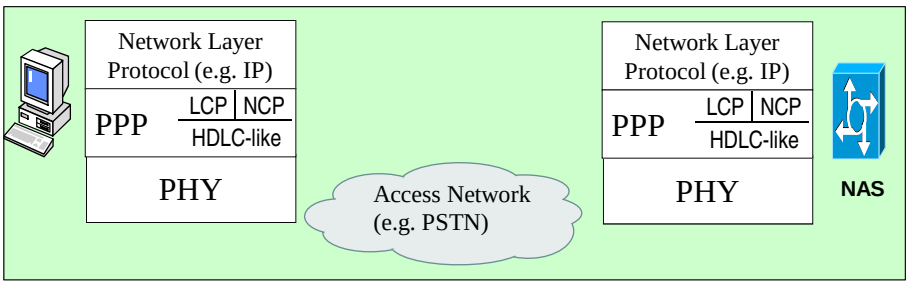


PPP Protocol Stack

➔ **Provides point to point link with <u>error detection capabilities</u>**
- ⇨ PC ←→ NAS

➔ **Runs over a transparent full-duplex PHY**
- → Assumed to deliver packets in order
- ⇨ E.g. circuit switched telephone connection
- ⇨ But also (semi)-permanent ADSL connection

➔ **Multi-protocol support**
- ⇨ May support IP as well as other network-layer protocols (IPX, Appletalk, etc)



PPP components

Three main components:

- **A method for encapsulating multi-protocol datagrams**
- **A Link Control Protocol (LCP) for establishing, configuring, and testing the data-link connection.**
- **A family of Network Control Protocols (NCPs) for establishing and configuring different network-layer protocols.**
  - IPCP = IP Control Protocol

What PPP does NOT provide

- **Multi-point**
  - Limited to single pair of TX/RX peers
- **Error correction, retransmission**
  - Provides only error checking via FCS
- **Delivery in order**
  - Relies on orderly deliver from PHY
- **Flow Control**
  - PPP transmits at maximum possible speed of the underlying PHY

Framing & Encapsulation

Frame Format

Flag 1 byte 01111110	Address 1 byte 11111111	Control 1 byte 00000011	Protocol 2 bytes -	Information (variable) -	FCS 2 bytes (or 4) -	Flag 1 byte 01111110

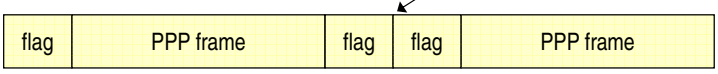
➔**Each frame starts & ends with reserved flag:**

⇨0111.1110 = 0x7e

➔**Just one flag required between consecutive frames**

⇨Two consecutive flags = empty frame

➔Just ignore



Address

➔**No station address assigned!**

⇨"all-station" address: 1111.1111 = 0xff

⇨Indeed: not necessary in a Point-to-point link!

Control field

➔**Fixed value:**

⇨Unnumbered Information

⇨0000.0011 = 0x03

i.e. Address & control NOT USEFUL (fixed values!!).
May be removed (through suitable negotiation) over slow links

Frame Check Sequence

➔**16 bits defauls**

⇨May be negotiated to 32 through LCP

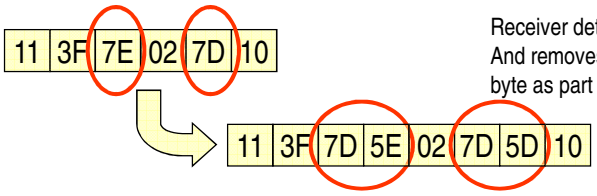
➔**Covers address, control, protocol, information (+ padding inside the information field, if employed)**

Flag 1 byte 01111110	Address 1 byte 11111111	Control 1 byte 00000011	Protocol 2 bytes -	Information (variable) -	FCS 2 bytes (or 4) -	Flag 1 byte 01111110

How to find the end of the Information?
Locate flag, then remove FSC!

Byte stuffing

- **Usual problem: how to send "reserved" bytes into information payload?**
  - E.g. 0111.1110 flag is used as frame delimiter!
- **Usual solution: stuffing via control escape octes:**
  - Send ("reserved" byte) as (control escape octet) + (reserved byte)XOR(0010.0000)
  - Extra XOR (bit 5 complemented) for improved robustness
- **Control escape octet**
  - 0111.1101 = 0x7d – of course NOW this is a reserved byte, too!



Receiver detects control escape
And removes it restoring original
byte as part of the frame (e.g. not as flag)

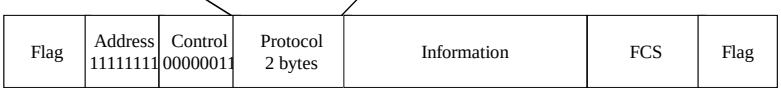
See RFC 1662 for bit-stuffing over bit synchronous links (versus byte-oriented links)

Encapsulation

Examples

Value	Protocol
0x0021	**Internet Protocol (IP)**
0x002d	**VJ compressed TCP/IP**
0xc021	**Link Control Protocol (LCP)**
0x8021	**Internet Protocol Control Protocol (IPCP)**
0xc023	**Password Authentication Protocol (PAP)**
0xc223	**Challenge Handshake Authentication Protocol (CHAP)**

- **2 bytes**
- **Special semantics:**
  - From 0xxx to 3xxx = network layer protocols
  - From 8xxx to Bxxx = associated NCPs
  - From 4xxx to 7xxx = upper protocols with no NCP
  - From Cxxx to Fxxx = Link-layer Control protocols



Information

➔**Information field:**

⇨from 0 bytes to MRU (Maximum Receive Unit)

➔**MRU default: 1500 bytes**

⇨Different values may be negotiated

➔**Padding: may be added to fill the frame up to MRU**

⇨Treated as information data (checked by FCS)

⇨PPP not responsible of recognizing and delimiting it

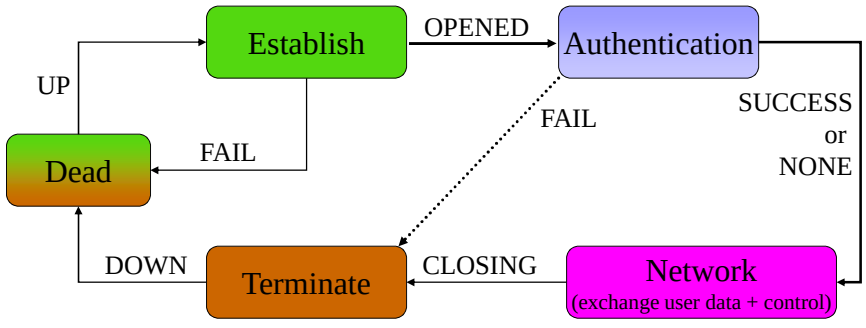
Link operation and phase diagram

PPP Steps

1) **configure the data link**
   - ⇨ Uses the Link Control Portocol (LCP)
2) **Link quality determination (optional)**
   - ⇨ Test the link to determine whether the link quality is sufficient to bring up network-layer protocols
3) **Authentication (optional, NOT mandatory)**
   - ⇨ The user MAY be authenticated, using a selected protocol
     - → PAP, CHAP, EAP-TLS, etc
4) **choose and configure one (or more!) network protocols**
   - ⇨ Uses the corresponding Network Control Protocol (NCP)
   - ⇨ For example, with IP, uses IPCP
5) **link active: send data packets**
6) **Link termination:**
   - ⇨ Explicit NCP or LCP command
   - ⇨ Other causes (timer expirations, external intervention, etc)

Giuseppe Bianchi

Phase Diagram



- → **Link dead = physical layer not ready**
  - ⇨ moves to establish when an external event occurs
  - ⇨ E.g. carrier signal detected, network configuration, etc
- → **Link establishment Phase: managed by LCP**
  - ⇨ Only LCP, other eventual packets discarded
  - ⇨ Link "opened" when LCP Configure-Ack send/received by both peers
- → **Authentication phase**
  - ⇨ The use of a SPECIFIC authentication protocol MUST be explicitly requested during the link establishment phase, via LCP packet exchange
  - ⇨ Default = no authentication
  - ⇨ Cannot move to NCP exchange until authentication is successful
- → **Network phase: multiple network protocols may be configured**
  - ⇨ each NCP may be opened and closed anytime
  - ⇨ Data packets are exchanged only at this phase

Giuseppe Bianchi

Link Termination phase

- **Multiple reasons why a link may need to close (at any time):**
  - Failed auth
  - Loss of carrier
  - Link quality failure
  - expiration of an idle-period timer
  - administrative closing of the link
    - E.g. user decides to disconnect
- **Link termination performed by LCP**
  - Non LCP packets not allowed
- **After termination, implementation signals PHY to disconnect**

LCP
Link Control Protocol

LCP goals

- **A signalling protocol, for:**
  - link set-up & configuration
  - Link termination
- **LCP used to automatically agree upon:**
  - the encapsulation format options,
  - varying limits on sizes of packets
  - detect loopbacks (!)
  - detect other common misconfiguration errors
- **LCP provides testing facilities**
  - to determine when a link is functioning properly and when it is failing
- **LCP allows to negotiate and provide support for authentication facilities**

Packet types

Three classes of LCP packets:

- **Link Configuration packets**
  - establish and configure a link
    - Configure-Request
    - Configure-Ack
    - Configure-Nak
    - Configure-Reject
- **Link Termination packets**
  - terminate a link
    - Terminate-Request
    - Terminate-Ack
- **Link Maintenance packets**
  - manage and debug a link
    - Code-Reject
    - Protocol-Reject
    - Echo-Request
    - Echo-Reply
    - Discard-Request

11 LCP packets Initially specified

LCP code	LCP packet
1	Configure-Request
2	Configure-Ack
3	Configure-Nak
4	Configure-Reject
5	Terminate-Request
6	Terminate-Ack
7	Code-Reject
8	Protocol-Reject
9	Echo-Request
10	Echo-Reply
11	Discard-Request

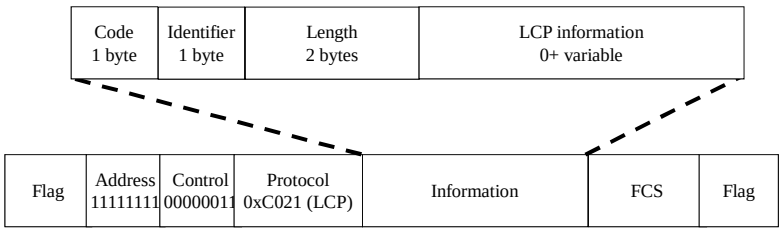
LCP-PDU format

➔ **Code (1 byte) = type of LCP packet**
- ⇨ If packet received with unknown code type, reply with code-reject

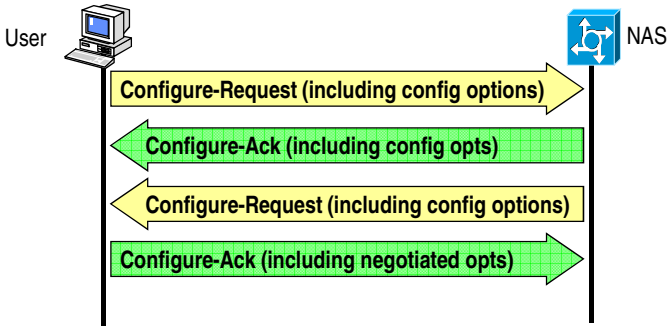
➔ **Identifier (1 byte)**
- ⇨ (typically a sequential) Value used to match requests with replies

➔ **Length (2 byte)**
- ⇨ Size of the LPC-PDU (including 4 bytes header)

➔ **Data: variable**
- ⇨ Bytes outside the specified length ➔ padding



Link establishment



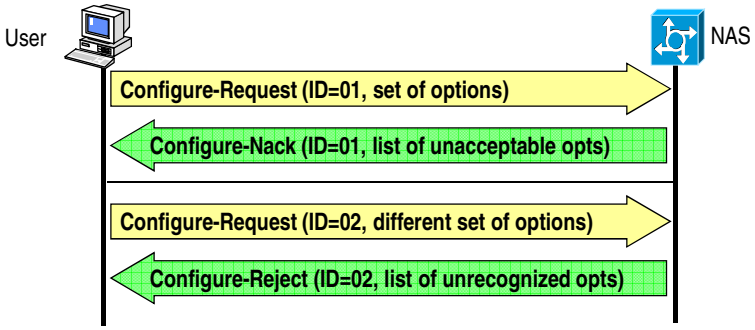
➔ **Remember: link is full duplex (bidirectional)!!**
- ⇨ Link establish: on BOTH directions
  - ➔Order does not matter – e.g. changing 2nd and 3rd message in the figure is OK
- ⇨ Negotiated options apply on a single direction (the receive direction)

➔ **Link establishent occurs in a SINGLE handshake**
- ⇨ ACK matches a single request (same ID)
- ⇨ includes ALL configuration options for the link

Link establishment via multiple handshake

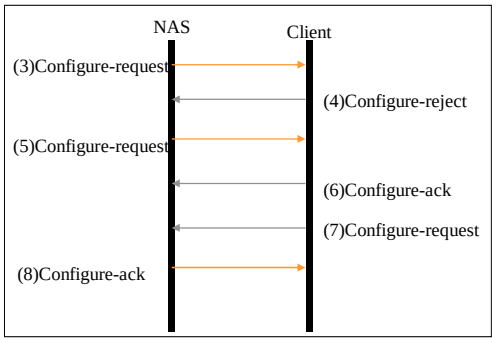
- **Multiple exchange may be needed to converge: iterative process**
- **Idea:**
  - There is a basic set of configuration parameters (defaults)
    - Which are not transmitted
    - Devised to handle all common configurations
  - The implementor can specify improvements to the default configuration:
    - Different configuration values
    - Additional configuration parameters
  - LCP self-configuration deploys an extensible option negotiation mechanism
    - each end of the link describes to the other its capabilities and requirements
    - The other end of the link may understand or not understand the description, and may accept or not the suggested configuration
- **An almost identical approach is delevoped in the family of NPCs**

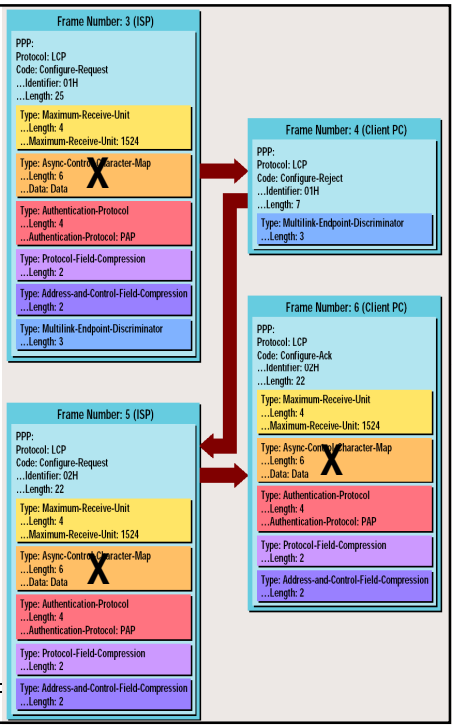
Nack vs Reject



- **NACK: every instance of the received Configuration Options is recognizable, but some values are not acceptable**
  - NACK options field filled with ONLY the unacceptable config opts from the Configure-Request.
  - Option values, when applicable, are modified to a value acceptable for the peer
- **Reject: some instances of the received configuration options are NOT recognized (or set by admin as not acceptable for negotiation)**
  - Reject options field filled with ONLY the unrecognized config opts from the Configure-Request.
- **General rule: ID always matches the request**

LCP Establish example

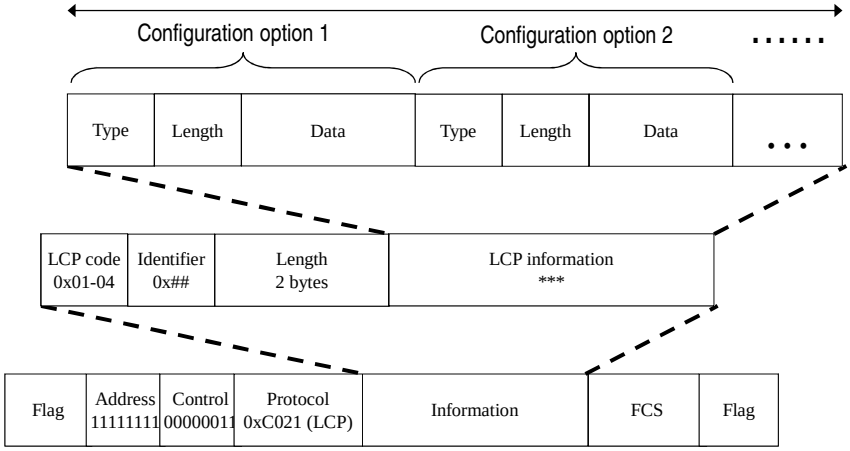




Giuseppe Bianchi

Coding options

0+ bytes, 0+ configuration options (0 → OK with all defaults!)



Giuseppe Bianchi

RFC 1661 Options

Options	Default	rationale
01 - Maximum receive unit	**1500**	Used to negotiate an MRU larger or smaller than 1500 bytes
03 - Authentication protocol	**None**	Used to negotiate an authentication protocol (initially: PAP, CHAP)
04 – Link quality protocol	**none**	Used to enable link quality monitoring (link quality reports – a protocol initially devised)
05 – Magic number	**none**	4 bytes random number. To catch loopbacks! ($2{,}3 \times 10^{-10}$ coincidence, solved through NACK]
07 - Protocol field compression	**Off**	Allows to use 1 byte protocol field instead of two (see compression details in RFC 1661)
08 - Address and control field compression	**Off**	Allows to avoid transmission of (fixed-pattern) address and control field (save 2 bytes)

Remember: ALL configs have an HALF-DUPLEX scope!!
(e.g. you may negotiate two different auth protocols on the two directions)

Newer options

- **Call Back (RFC1570)**
- **Multilink PPP related options (RFC 1990)**
  - Single logical PPP connection over multiple physical links
    - E.g. when bandidth on demand used
    - E.g. when two ISDN B channels used

Link termination

➔ **Two LCP packets:**
- ⇨ Terminate-Request
  - ➔ Peer wishing to close the connection
- ⇨ Terminate-Ack
  - ➔ Must be sent upon Terminate-Request
  - ➔ Usual rule: same ID

➔ **Terminate-Requests are repeatedly sent (with same ID) until:**
- ⇨ Terminate-Ack received, or
- ⇨ the lower layer indicates that it has gone down, or
- ⇨ a sufficiently large number have been transmitted such that the peer is down with reasonable certainty

Managing different PPP versions

➔ **Through the Code-Reject packet**

LCP code 0x07 (code-reject)	Identifier 0x##	Length 2 bytes	Copy of rejected LCP packet (possibly truncated to meet the MRU limits)

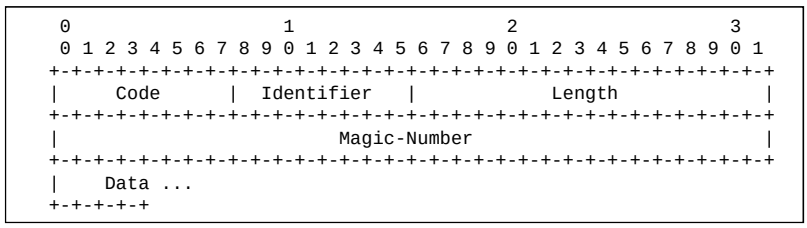
➔ **Sent by receiving peer when LCP packet with unrecognized code is received**
- ⇨ May happen with different PPP versions
- ⇨ E.g. current PPP versions include code 0x0C (12) = Identification

➔ **Typically reception of Code Reject requires link termination and problem reporting**
- ⇨ it is unlikely that the situation can be rectified automatically

➔ **Protocol-reject: completely different goal, but similar approach**
- ⇨ Notify a packet with unknown protocol field has been recognized

LCP debug packets

- **Echo Request**
- **Echo Reply**
  - Carry magic number
  - To round-trip test link
- **Discard-Request**
  - One-way test
  - Discarded at reception



Network Control Protocols (NPCs)

IP Control Protocol (IPCP)

PPP design strategy

- **Keep layer 2 clearly separate from layer 3**
  - Obvious, but…
  - When setting up a link, we also need to set up networking facilities in order to **use it**!
- **Point-to-Point links tend to exacerbate many problems with the current family of network protocols.**
  - For instance, assignment and management of IP addresses, is a big problem even in LAN environments
    - →Remember DHCP? ☺
  - As such, it is especially difficult over circuit-switched point-to-point links (such as dial-up modem servers).
- **Configuration of the networking protocol(s) employed delegated to specific Network Control Protocols (NCPs)**
  - An NCP devised for each network-layer protocol
    - →E.g. IPCP (RFC 1332):
      PPP Network Control Protocol specific for IP configuration

Network Control Protocols

- **NCPs Enter into play during the "network phase"**
  - Link already established and configured
  - AUTH performed, if required
- **NCP operation VERY similar to LCP**
  - Same LCP packet names (a subset of), same meaning
    - →Configure-Request, Configure-Ack,
    - →Configure-Nak, Configure-Reject
    - →Terminate-Request, Terminate-Ack
    - →Code-Reject
- **Key difference: negotiated options**
  - E.g. for IPCP:
    - →IP address of the PPP interface
    - →default gateway, DNS, other servers
    - →IP Header compression
    - →…

IP Control Protocol

➔ **IPCP responsibility:**

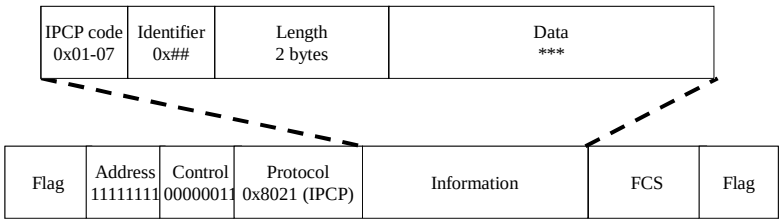
⇨ configure, enable, and disable the IP protocol modules on both ends of the point-to-point link

➔ **IPCP mechanisms:**

⇨ Same packet exchange mechanism as LCP

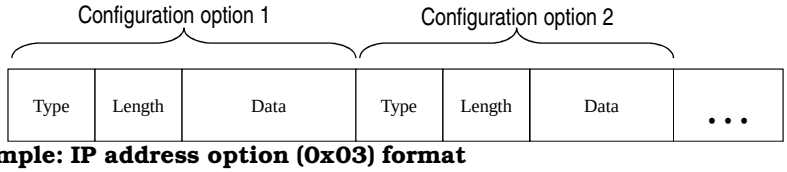
⇨ Same encapsulation

⇨ Same codes (but limited to codes 1-7)



IPCP options

➔ **Same option format**

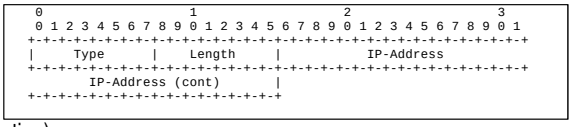


➔ **Example: IP address option (0x03) format**

➔ **Static addresses:**

⇨ Configure-Request: contains IP address

⇨ Configure-ACK: Acknowledges IP addr (retransmitted in ACK option)



➔ **Dymanically assigned addresses:**

⇨ Configure-Request (PC → NAS): contains option 0x03 with 0.0.0.0

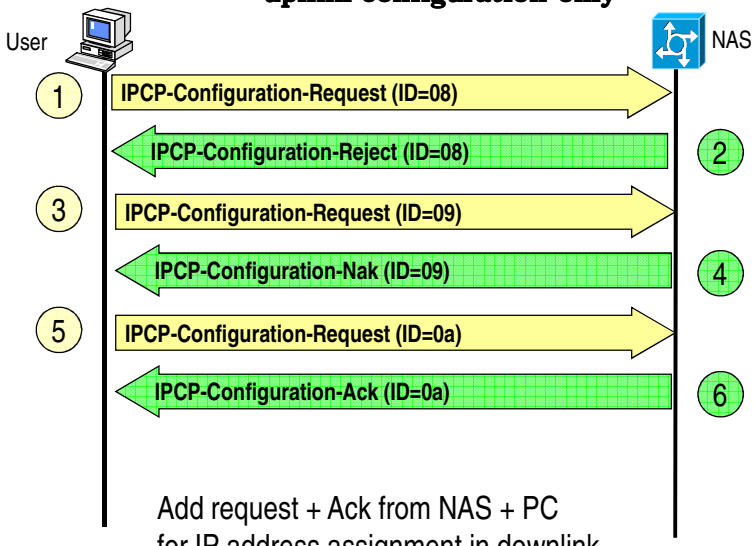
⇨ Configure-NACK (NAS→PC): contains option 0x03 with valid IP address

⇨ Configure-Request (PC → NAS): contains option 0x03 with valid IP address

⇨ Configure-ACK (NAS→PC): commits assignment & contains option 0x03 with valid IP address

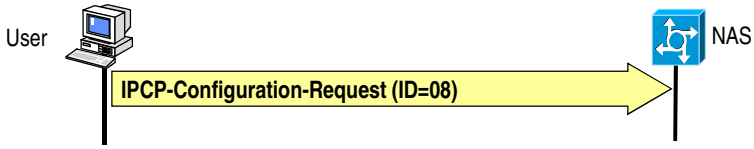
An example (from a real trace)

uplink configuration only



Add request + Ack from NAS + PC
for IP address assignment in downlink

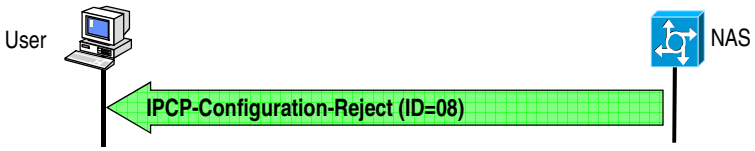
Packet 1




```
PPP IP Control Protocol
    Code: Configuration Request (0x01)
    Identifier: 0x08
    Length: 40
    Options: (36 bytes)
        IP compression protocol: 6 bytes
            IP compression protocol: VJ compressed TCP (0x2d)
            Data (2 bytes)
        IP address: 0.0.0.0
        Primary DNS server IP address: 0.0.0.0
        Primary WINS server IP address: 0.0.0.0
        Secondary DNS server IP address: 0.0.0.0
        Secondary WINS server IP address: 0.0.0.0
```

Asks for 6 configuration parameters:
IP address, Header compression, Primary DNS, Secondary DNS, Primary WINS, Secondary WINS

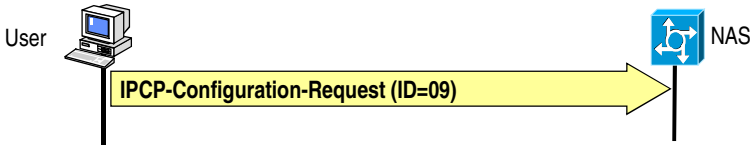
Packet 2




```
PPP IP Control Protocol
    Code: Configuration Reject (0x04)
    Identifier: 0x08
    Length: 22
    Options: (18 bytes)
        IP compression protocol: 6 bytes
            IP compression protocol: VJ compressed TCP (0x2d)
            Data (2 bytes)
        Primary WINS server IP address: 0.0.0.0
        Secondary WINS server IP address: 0.0.0.0
```

Does not support 3 configuration parameters requested:
Header compression, Primary WINS, Secondary WINS

Packet 3

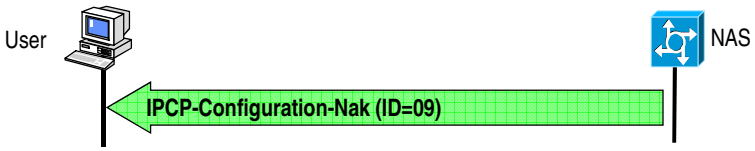



```
PPP IP Control Protocol
    Code: Configuration Request (0x01)
    Identifier: 0x09
    Length: 22
    Options: (18 bytes)
        IP address: 0.0.0.0
        Primary DNS server IP address: 0.0.0.0
        Secondary DNS server IP address: 0.0.0.0
```

Asks for the three supported configuration parameters ONLY:
IP address, Primary DNS, Secondary DNS

(DNS configuration options → see RFC 1877)

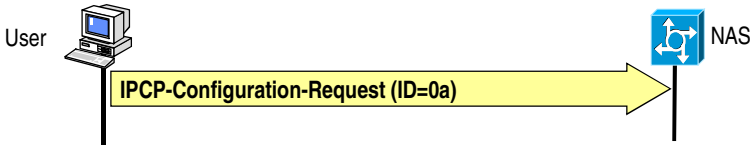
Packet 4




```
PPP IP Control Protocol
    Code: Configuration Nak (0x03)
    Identifier: 0x09
    Length: 22
    Options: (18 bytes)
        IP address: 83.184.169.230
        Primary DNS server IP address: 212.247.156.66
        Secondary DNS server IP address: 212.247.156.70
```

NACKs request (since it was a "question": all fields 0s), and responds with valid config: IP address, Primary DNS, Secondary DNS

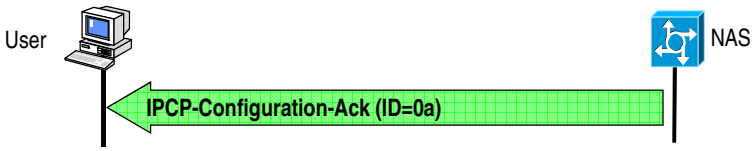
Packet 5




```
PPP IP Control Protocol
    Code: Configuration Request (0x01)
    Identifier: 0x0a
    Length: 22
    Options: (18 bytes)
        IP address: 83.184.169.230
        Primary DNS server IP address: 212.247.156.66
        Secondary DNS server IP address: 212.247.156.70
```

Explicit request with the tree suggested values: IP address, Primary DNS, Secondary DNS

Packet 6




```
PPP IP Control Protocol
    Code: Configuration Ack (0x02)
    Identifier: 0x0a
    Length: 22
    Options: (18 bytes)
        IP address: 83.184.169.230
        Primary DNS server IP address: 212.247.156.66
        Secondary DNS server IP address: 212.247.156.70
```

Acknowledges the three requested configuration parameters:
IP address, Primary DNS, Secondary DNS

Header Compression Option(s)

- **→Various algorithms proposed**
- **→Joint compression of IP+transport header**
  - ⇨Van Jacobson TCP/IP header compression
    - →Compression down to 3 bytes (versus initial 40)
  - ⇨Robust Header Compression (ROHC)
    - →Specific for real-time stack (RTP/UDP/IP)
    - →Greater reliability in the presence of packet loss
      - » Limited error propagation in case of packet drop
    - →Compression down to 1 byte (versus initial 40)

Lecture 1.2:

PPP over Ethernet

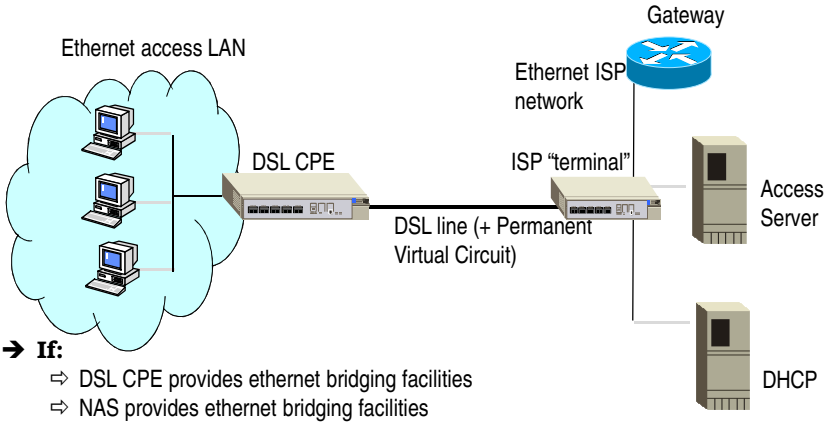
Recommended reading: RFC 2516,February 1999

What is PPPoE

➔A nice definition:

The PPPoE (Point to Point Protocol over Ethernet) specifies how an ISP and a remote PC can set up a session-based Internet connection on top of the session-less Ethernet protocol.

PPPoE: not needed…



➔ **If:**
- ⇨ DSL CPE provides ethernet bridging facilities
- ⇨ NAS provides ethernet bridging facilities
- ⇨ DLS network can transport ethernet frames

➔ **Then:**
- ⇨ Why do we need PPP (and PPPoE) at all?
- ⇨ DHCP would be more than sufficient!

But…

➔ **In many cases, ISPs must connect multiple remote hosts through the same customer premise access device**
- ⇨ Residential DLS, WiMax Subscriber Station, etc

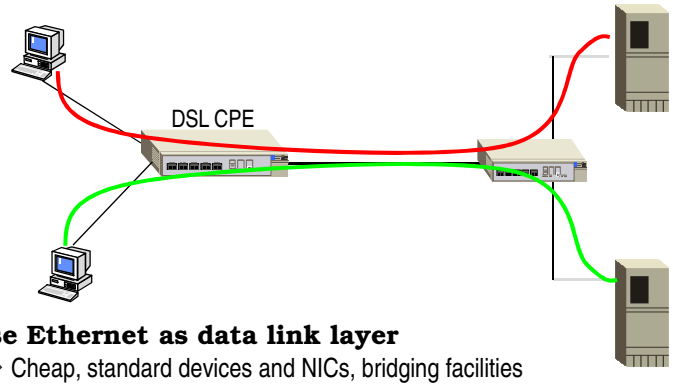
➔ **but they also want to provide selective access control and billing functionality…**
- ⇨ DHCP does not provide support for authentication
- ⇨ Hard to provide differentiated services with a pure Ethernet infrastructure
  - →Not all the users in the access LAN are the same
- ⇨ Accountability issues
  - →NAS can only account on a per-LAN basis, not on a per-user basis!!!

➔ **Exactly as they do provision dial-up services with PPP…**
- ⇨ All this is EASILY dealt with PPP!

➔ **… and with minimal/null configuration**
- ⇨ User should NOT interact with the DSL modem
- ⇨ Nor install an ATM NIC
- ⇨ Users well acquainted with their simple remote dial-up access connection

PPPoE idea

Enable point-to-point relationships in a multi-point multi-access domain such as Ethernet



➔ **Use Ethernet as data link layer**
- ⇨ Cheap, standard devices and NICs, bridging facilities

➔ **But encapsulate PPP into Ethernet Frames**
- ⇨ To create point-to-point **sessions** between user and (specific) server
- ⇨ To take advantage of all the deployed features with PPP
  - →DSL/WiMax users handled exactly as dial-up!

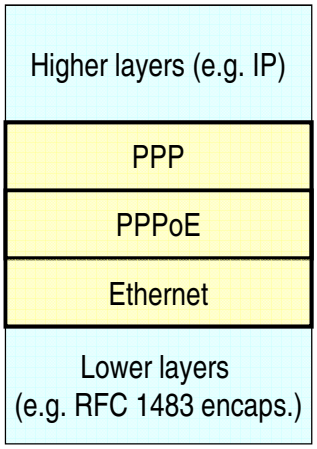
PPPoE phases

➔**Phase 1: discovery phase:**
- ⇨Allow discovery of ethernet addresses of the two involved peers
- ⇨Establish unique session identifier

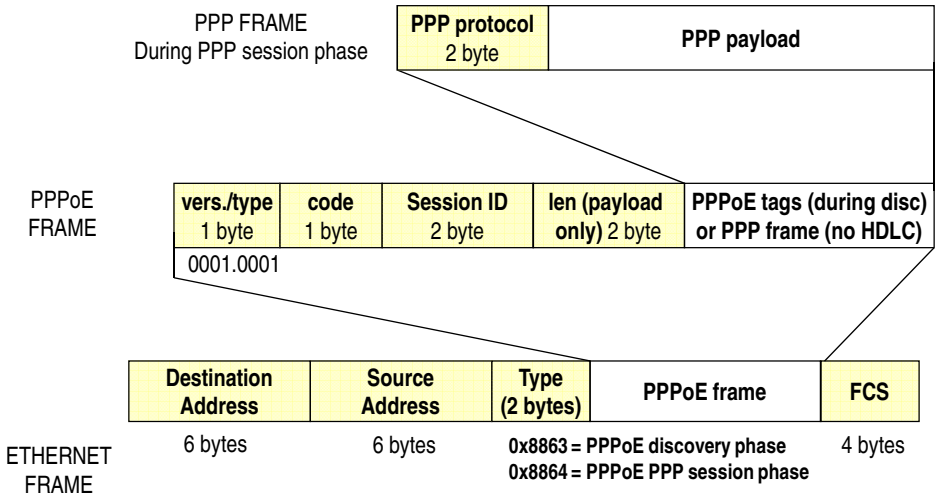
➔**Phase 2: PPP session stage**
- ⇨Standard PPP frames (LCP, PAP, CHAP, IPCP, etc) are exchanged

➔**Need for a new protocol**
- ⇨Encapsulating PPP frames inside ethernet frames is not sufficient…

Protocol stack



Packet format



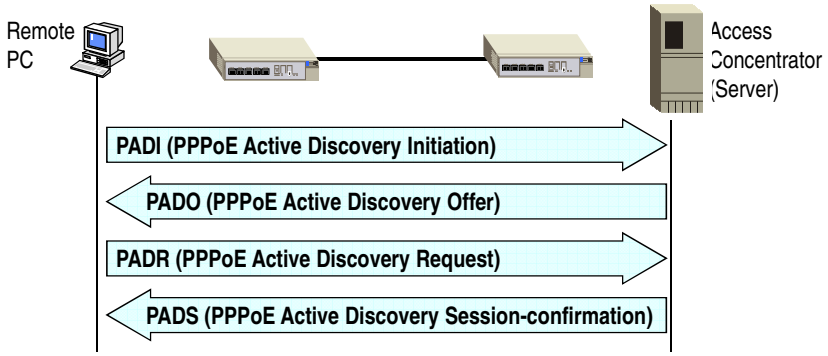
6 (PPPoE) + 2 (PPP) = 8 bytes overhead

Discovery phase

➔ **4-way handshake similar to DHCP!!**

➔ **But adapted to PPPoE goal:**

⇨ Identify Ethernet addresses, rather than assign IP addresses (deal with later by IPCP)



Discovery phase details /1

➔ **PADI (code = 0x09)**

⇨ Destination address = broadcast

⇨ Session ID = 0

⇨ Contains service-type TAG (to request a specific service)

➔ **PADO (code = 0x07)**

⇨ Destination address = unicast

⇨ Session ID = 0

⇨ Possibly more than one offer

➔ Multiple access concentrators may provide requested service and may respond

⇨ Must contain one TAG with the Access Concentrator name

➔ AC-Name

TAG = usual extensible approach: triplet (type, length, value) – see details in RFC 2516

Discovery phase details /2

➔**PADR (code = 0x19)**

- ⇨Destination address = unicast
  - →To the selected AC
- ⇨Session ID = 0
- ⇨Contains service-type TAG (to request a specific service), again

➔**PADS (code = 0x65)**

- ⇨Destination address = unicast
- ⇨AC generates UNIQUE session ID for the PPPoE session
  - →And fills session ID field in the PADS
- ⇨Service-Type TAG = service under which AC has accepted the PPPoE session

Session termination

➔**PADT**

- ⇨PPPoE Active Discovery Terminate packet
  - →Code: 0xa7
- ⇨Contains session ID of session which must be terminated
- ⇨May be transmitted at any instant of time

PPP session phase

- **Normal PPP packets are encapsulated into a PPPoE frame (code =0x00) and then into an Ethernet frame**
- **problem: MTU = 1492**
  - 1500 = Ethernet MTU minus 8 bytes PPP/PPPoE
  - Not easy to change the MTU in the OS
    - By default it is set to the Ethernet MTU
    - Some PPPoE implementations do not change OS MTU
  - Result: IP fragmentation and reduced performance
    - Which may become very poor during file transfer if Path_MTU_Discovery not supported by your TCP - unlikely