Point-to-Point Protocols

J Pan
pan@uvic.ca
web.uvic.ca/~pan

Lecture review

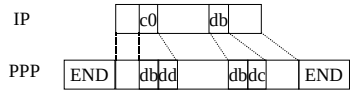
- We have some building blocks in DLC
  - frame control (flag, ESC)
    - bit/byte stuffing
  - flow control (seqno)
    - sliding window
  - error control (checksum, CRC, FEC, timer)
    - go-back-N, selective repeat
- But how these things work in real world?!

Point-to-Point Protocols

- SLIP [RFC 1055]
  - Serial Line Internet Protocol
- CSLIP [RFC 1144]
  - Compressed SLIP
- HDLC*
  - High-level Data Link Control
- PPP [RFC 1661]
  - the Point-to-Point Protocol

SLIP/PLIP

- Very simple encapsulation
  - flag: END (0xc0)
  - escape: ESC (0xdb)
    - "END": 0xdb 0xdd; "ESC": 0xdb 0xdc
- *Not* an Internet Standard [RFC 1055]
- But widely used in early days



4

On the other side ...

- no peer negotiation
  - known IP
- no protocol multiplex
  - IP only
- no flow control
  - reliant on upper layers
- no error control (detection, recovery)
  - reliant on upper layers

Another issue

- SLIP has low overhead, but
  - IP header: 20+ bytes
  - TCP header: 20+ bytes
  - payload: for Telnet/Rlogin, 1 byte!
- Look at these TCP/IP headers again
  - most are static for a particular flow
  - some are predictable
- Approach: delta encoding

CSLIP

- Von Jacobson's compression [RFC 1144]
  - TCP/IP header 40 bytes => as few as 3 bytes
  - maintain up to 16 TCP/IP connections*
- Greatly reduce response time
  - important for Telnet/Rlogin
  - supported widely in various systems
- Used in follow-on protocols as well

HDLC

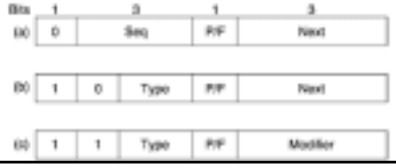
- History
  - IBM: SDLC (Synchronous Data Link Control)
  - ANSI: ADCCP (Advanced Data Communication Control Procedure)
  - ISO: HDLC (High-level Data Link Control)
  - CCITT: LAP (Link Access Procedure), LAPB
- All similar with many small differences
- An international standard, but ...

HDLC frame

- Flag: 01111110 (bit stuffing)
- Address: when there are multiple terminals
- Control: type, seqno, ack, etc.
- Data: variable length
- Checksum: CRC (CCITT: $x^{16}+x^{12}+x^5+1$)

HDLC control

- Information frame: w/ seqno and ack
- Supervisory frame: R, Rej, NR, SelRej
- Unnumbered frame: connection control; unreliable connectionless data

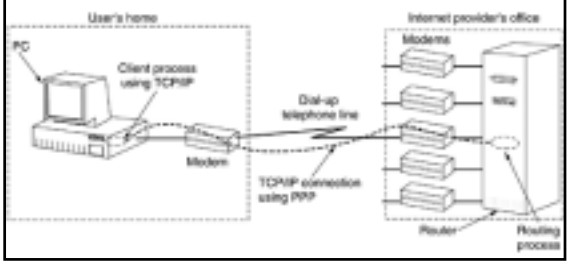


PPP/MPPP

- Main functionality
  - framing, error detection, multiplex [RFC 1661]
  - link control [RFC 1661]
  - network control [e.g., IPCP, RFC 1332]
- Major procedures
  - (physical channel established)
  - LCP: negotiate link parameters
  - (authentication)
  - NCP: negotiate network parameters

PPP dialup

- An example



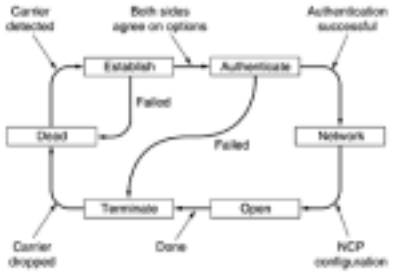
PPP frame

- Flag: 01111110
- [Address]: 11111111 (all stations)
- [Control]: 00000011 (unnumbered frame)*
- Protocol: LCP, NCP, IP, IPX, AppleTalk, ...
- Payload: variable length
- Checksum: CRC (16 or 32-bit)

PPP transparency

- Flag: 0x7e
  - synchronous link
    - hardware bit stuffing (one 0 after 5 consecutive 1's)
  - asynchronous link
    - "0x73": 0x7d 0x5e
    - "0x7d": 0x7d 0x5d
    - ASCII control character: e.g., "0x01" => 0x7d 0x21

PPP diagram



LCP packet

- Protocol: 0xc021; code-ID-length-info; I/R

Name	Direction	Description
Configure-request	I → R	List of proposed options and values
Configure-ack	I ← R	All options are accepted
Configure-nak	I ← R	Some options are not acceptable
Configure-reject	I ← R	Some options are not negotiable
Terminate-request	I → R	Request to shut the line down
Terminate-ack	I ← R	OK, line shut down
Code-reject	I ← R	Unknown request received
Protocol-reject	I ← R	Unknown protocol requested
Echo-request	I → R	Please send this frame back
Echo-reply	I ← R	Here is the frame back
Discard-request	I → R	Just discard this frame (for testing)

LCP negotiation

- Maximum-Receiver-Unit (MRU)
- Authentication-Protocol
- Quality-Control
- Magic-Number (loop-back detection)
- Protocol-Field-Compression (PFC)
- Address-and-Control-Field-Compression

IPCP

- Protocol
  - 0x8021: IPCP [RFC 1332]; 0x0021: IP
  - Configure-*, Terminate-*, Code-reject
- Negotiate
  - compression protocol
    - 0x002d: VJ's compression protocol
    - number of state slots (usually 16)
  - IP address
    - indicate one or request one (00:00:00:00)

PPP is more than just dialup

- PPPoA: PPP over ATM [RFC 2364]
  - some ADSL ISPs using ATM transport
- PPPoE: PPP over Ethernet [RFC 2516]
  - often used by DSL and cable modem ISPs
- Packet over SONET/SDH
  - mostly point-to-point backbone links
- L2TP: Layer-2 *Tunneling* Protocol (PPP/IP)
- PPTP*: Point-to-Point TP (PPP+GRE)

PPP authentication

- PAP: Password Authentication Protocol
  - username and password
  - PPP encapsulation: 0xc023
- CHAP: Challenge Handshake AP
  - no password exchanged; 3-way handshake
  - PPP encapsulation: 0xc223
- MSCHAP: Microsoft CHAP
- EAP: Extensible Authentication Protocol

PPP in action

- PPP in Microsoft Windows
  - Modems
    - COM1, COM2, COM3, COM4
  - Dial-Up Networking (DUN)
    - username, password
    - dial-in phone number
    - PPP selected
    - TCP/IP configured (usually via DHCP)

PPP in Linux

- Pre-PPP
  - get ppp software package (pppd)
  - have ppp kernel support (built-in or modular)
  - know login info (phone#, username, password)
  - find modem
    - /dev/modem => /dev/ttyS0 == COM1
    - winmodem: linmodem.org; USB modem
  - configure ppp options
    - /etc/ppp/*

PPP in Linux

- talk to modem
  - program: chat (expect-send conversation)
  - AT command strings: init, dialup a number
- talk to remote server
  - [login w/ username/password, run remote ppp]
  - LCP negotiation
  - PAP/CHAP (/etc/ppp/pap/chap-secrets)
  - NCP negotiation

PPP in Linux

- Post-PPP
  - show interface configuration
    - ifconfig ppp0
  - show routing configuration
    - netstat -r
  - show name server configuration
    - /etc/host.conf; /etc/resolv.conf
- Good thing is actually working :-)

Summary

- PPP
  - HDLC-like framing
  - byte stuffing
  - dumb address/control field
  - protocol multiplexing
  - FCS (checksum): error detection
  - no flow control, error recovery w/ U-frame
  - LCP and NCP

More questions?

- web.uvic.ca/~pan