Post 4

Thursday, January 24, 2019 11:23 PM

CH. 3 POINT-TO-POINT CONNECTIONS: P2

3 phases of establishing PPP session:

Phase 1	Link establish/config negotiation: • Before PPP exchanges datagrams [IP]: Open connection/negotiate options • Phase done when receiving rtr sends a config-ack frame back to rtr initiating connection
Phase 2	Link quality determination: [optional]: • LCP tests link to determine quality good enough to bring up network layer protocols ○ Can delay trans info until phase done
Phase 3	Network layer protocol config negotiation: • After LCP has finished link quality: • NCP can separately config network layer protocols/bring them up/down any time • LCP closes link? Informs layer protocols for appropriate action

Link remains config for comm until LCP/NCP frames close link: Or external event occurs

- *Example:* Inactivity/admin || LCP: Can term link any time: Example: Loss of link quality

LCP Operation: Link establishment/maintenance/termination

3 classes of LCP frames to accomplish work of each phases

1. **Link-establish frames:** Establish/config link:
   - Config-Req, Config-Ack, Config-Nak, Config-Reject
2. **Link-maintenance frames:** Manage/debug link:
   - Code-Reject, Protocol-Reject, Echo-Req, Echo-Reply, Discard-Req
3. **Link-termination:** Frames term link:
   - Term-Req, Term-Ack

Phase 1 LCP: Link Establishment: Must complete before any network layer packets exchanged

- During establishment: LCP opens connection/negotiates params
- Starts w/initiating device sending Config-Req frame to responder
- Config-Req frame includes var # of config options needed to set up on link
- Initiator includes options: How it wants link created/protocol/auth params
- **Responder processes req:**
  - **If options not good:** Responder sends Config-Nak/Config-Reject msg
    - If negotiation fails: Initiator must restart process w/new options
  - **If options good:** Responder responds w/Config-Ack msg
    - Moves to auth stage: Op of link handed over to NCP
    - When NCP finished configs [inclu. validating auth]: Line avail for data transfer
    - During data exchange: CLP transitions into link maintenance

Phase 2: Link Maintenance: LCP can use msgs to provide feedback/test link

Echo-Req/Echo-Reply/Discard-Req	Frames used for testing link
Code-Reject/Protocol-Reject	Frame types provide feedback when 1 device receives invalid frame • Unrecognized LCP code (frame type)/bad protocol identifier

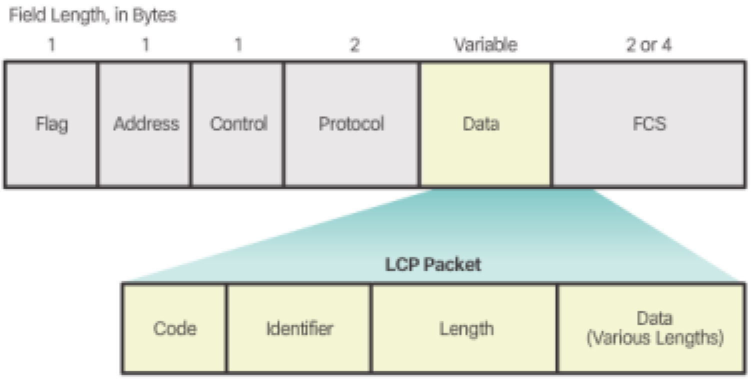
Phase 3: Link Termination: After transfer completes: NCP terms link

- Only terms network layer/NCP link || Link remains open until then
- If LCP terms link before NCP: NCP session also term

PPP can term link any time:

- Loss of carrier/auth failure/link quality/expiration of idle-period timer/admin
- LCP closes link by exchanging Term packets
- Device w/shutdown sends Term-Req msg: Other device replies w/Term-Ack

LCP Packet: Each packet has specific function in exchange of config info depending on type: Code field of LCP identifies it



Code	1 byte: ID packet type
Identifier	1 byte: Match packet req/replies
Length	2 bytes: Total length all fields of LCP packet
Data	0/more bytes: Fmt determined by code

LCP Packet Fields

Code	Packet Type	Description
1	**Config-Req**	Open/reset PPP connection • List of options w/changes to defaults
2	**Config-Ack**	All options in last Config-Req recognized • Both PPP peers send/receive Config-Acks: LCP negotiation done
3	**Config-Nak**	All LCP options recognized: Some values not good • Config-Nak: Includes mismatched options/acceptable values
4	**Config-Reject**	LCP options not good for negotiation • Includes not good options
5	**Term-Req**	Close PPP connection
6	**Term-Ack**	Response to Term-Req
7	**Code-Reject**	LCP code unknown: Includes rejected LCP packet
8	**Protocol-Reject**	PPP frame contains unknown Protocol ID • Includes rejected LCP packet • Sent by PPP peer in response to PPP NCP for LAN protocol not enabled
9	**Echo-Req**	Test PPP connection
10	**Echo-Reply**	Response to Echo-Request: Not related to ICMP echo req/replies
11	**Discard-Req**	Exercise link in outbound direction

PPP Config Options: Can be config to support optional funcs:
Optional Functions:

- Auth using PAP/CHAP
- Compression using Stacker/Predictor
- Multilink combines 2/more chans to increase WAN BW

To negotiate use of these options:

- LCP link-establishment frames contain option info in data field of LCP frame
- If config option not included in frame: Default value is assumed
- Phase complete when config ack frame sent/received

NCP Process: After link initiated: LCP passes control to appropriate NCP
Initially designed for IP packets: PPP can carry data from multiple network layer protocols

- Uses a modular approach
- Allows LCP to set up link/transfer details of network protocol to specific NCP
- Each network protocol has corresponding NCP: Each NCP has corresponding RFC
- NCPs use same packet fmt as LCPs

After LCP config/auth basic link: Right NCP invoked to complete specific config of protocol being used

- When successful: Protocol is in open state of established LCP link

- PPP can then carry corresponding layer protocol packets

IPCP: Responsible for config/enabling/disabling IPv4 modules on both ends of link: IPv6CP is an NCP w/same roles for IPv6

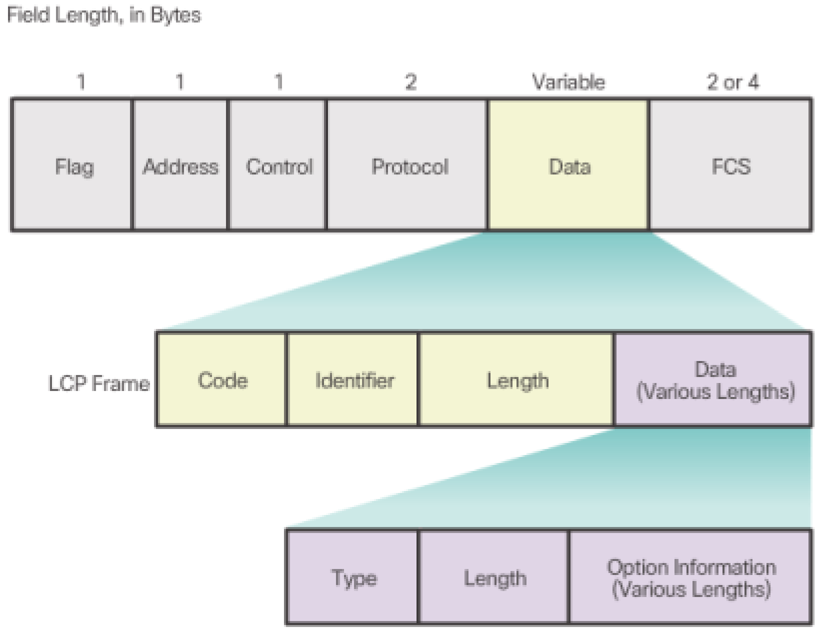
IPCP negotiates 2 options:

Compression	Allows devices to negotiate alg to compress TCP/IP headers/save BW • Van Jacobson TCP/IP header compression: Reduces size of TCP/IP headers to as few as 3 bytes • Sig improvement on slow serial lines: For interactive traffic
IPv4-Address	Allows initiating device to specify an IPv4 to use for routing IP over PPP link • Or request IPv4 for responder • Before this: Broadband tech (DSL/cable/dialup) links used IPv4 option

After NCP process complete:

- Link goes into open state: LCP takes over again in link maintenance phase
- Link traffic: Any possible combo of LCP/NCP/Network layer protocol packets
- When data transfer complete: NCP terms protocol link: LCP terms PPP connection

PPP Config Options



Option Name	Type	Length	Description
Auth Protocol	3	5 or 6	Field indicates auth protocol: PAP/CHAP
Protocol Compression	7	2	Flag indicating PPP protocol ID compressed to single octet • When 2byte protocol ID in range 0x00 – 0x00-FF
Addr/Control Field Compression	8	2	Flag indicate PPP addr/control field be removed from header • Address: Always 0xFF • Control: Always 0x03
Magic # (Error Detection)	5	6	Random # chosen to distinguish peer/detect looped back lines
Callback	13 or 0x0D	3	1 octet indicator of how callback to be determined

PPP may include following LCP options: Options config? Corresponding field value inserted into LCP option field

Auth	**Peer rtrs exchange auth msgs: 2 choices:** • PAP: Password Auth Protocol • CHAP: Challenge Handshake Auth Protocol
Compression	Increases effective throughput on PPP connections: Reduces amt of data in frame • Protocol decompresses frame at destination **2 compression protocols: Cisco:**

	1. Stacker 2. Predictor
Error detection	**ID's fault conditions:** Quality/Magic # help ensure reliable/loop-free data link • **Magic #:** Helps detect links in looped-back condition • Until successfully negotiated: Must be transmitted as 0 • Generated randomly at each end of connection
PPP Callback	**Used to enhance sec:** Cisco rtr can act as a callback client/server • Client: Makes initial call: Requests server call it back: Terms initial call • Callback rtr: Answers initial call/makes return call to client based on config statements **ppp callback** [**accept** | **request**]
Multilink	Load balancing over rtr ints that PPP uses: **AKA MP/MPPP/MLP:** • Method for spreading traffic across multiple phys WAN links • Provides packet fragmentation/reassembly/proper seq/multivendor interop • Load balancing on inbound/outbound traffic

PPP Basic Config

Enable PPP on int: Encapsulation method via serial

encapsulation ppp [*int config*]

Example:

```
R1(config)# int s0/0/0
R1(config-if)# encapsulation ppp
```

- No args: If not config on Cisco: Default encapsulation for serial ints is HDLC

Compression

R1(config-if) compress [**predictor | stac**]

- PPP SW compression on serial ints can be config after encapsulation enable
- Can affect sys performance
- If traffic already consists of zip/tar/mpeg/etc…don't use this option

Link Quality Monitoring: LCP provides optional link quality phase: Tests link: Determines quality sufficient to use L3 protocols

ppp quality percentage

- Ensures link meets quality req set; otherwise: Closes

Percentages calc for both incoming/outgoing directions

Outgoing: Calc: Compare total # packets/bytes sent to total # of packets/bytes received by dest node

Incoming: Calc: Compare total # packets/bytes received to total # packets/bytes sent by dest node

- If link quality % not maintain: Link deemed poor/taken down

LQM: Link Quality Monitoring: Implements time lag so link doesn't bounce up/down

Monitors data dropped on link/Avoids frame looping:

```
R1(config)# int s0/0/0
R1(config-if)# encapsulation ppp
R1(config-if)# ppp quality 80
```

PPP Multilink: AKA MP/MPPP/MLP/Multilink:

Config MPPP: 2 steps:

- **Create multilink bundle**

int multilink number Creates multilink int

- Int config: IP addr assigned to multilink int
- int enabled for multilink PPP
- int assigned multilink group #

Example:

```
int multilink 1
ip address 10.0.1.2 255.255.255.252
ipv6 address 2001:db8:cafe:1::2/64
ppp multilink
ppp multilink group 1
```

- **Assign ints to multilink bundle:**

Each int part of multilink group:

- Enabled PPP encapsulation
- Enabled multilink PPP

- Bound to multilink bundle using multilink group #

Verify

show interfaces

show interfaces serial Verify proper config of HDLC/PPP encapsulation

HDLC: Output of sh int serial should display HDLC

PPP: LCP/NCP states also display

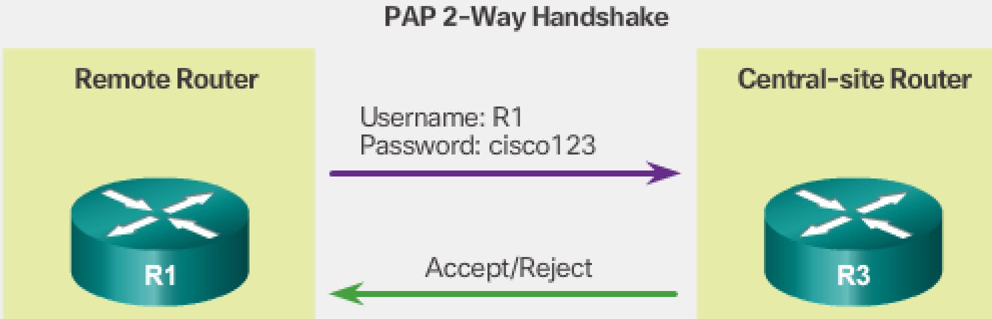
IPv6: IPCP also displays for IPv6CP

show ppp multilink Verify PPP multilink enabled: Verifies enable/hostnames local/remote endpoints/serial ints

PPP Auth Protocols: Defines extensible LCP: Allows negotiation of auth protocol for auth its peer

- Before allowing protocols to transmit over link
- RFC 1334: 2 protocols for auth: PAP/CHAP

PAP: Password Authentication Protocol:



- NO encryption
- Usrname/pass sent in plaintext
- If accepted: Connection allowed

PPP: Performs L2 auth in addition to other layers of auth/encryption/access control/gen sec

- Provides simple method for remote note to establish ID using 2-way handshake
- Not interactive
- when ppp encapsulation used: Usrname/passwd sent as 1 LCP data package
- Rather than server sending login prompt/waiting for response
- After PPP completes link establishment phase: Remote node keeps sending usrname/passwd pair across link
- Until receiving node ack's/terms connection

Completing: At receiving node: Username/passwd checked by auth server that allows/denies connection

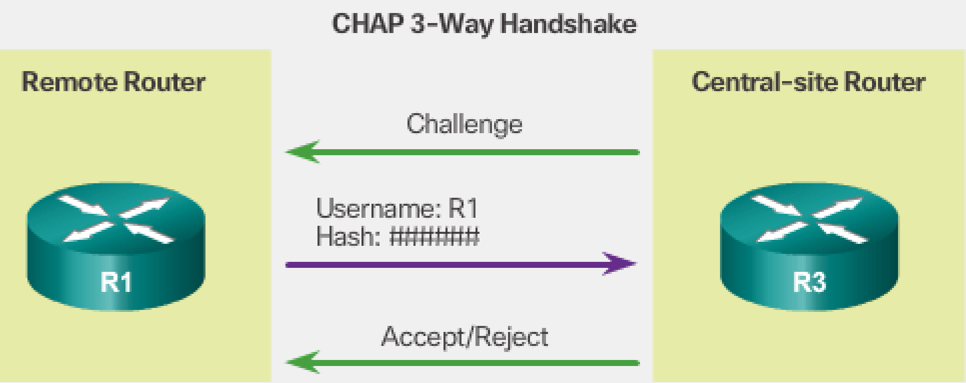
- Accept/Reject msg sent to requester

PAP: NOT strong auth protocol: No protection from playback/repeated trial-and-error attacks

PAP may be used in following envs:

- CHAP not supported
- Vendor incompatibilities
- When plaintext passwds must be avail to simulate a login

CHAP: Challenge Handshake Authentication Protocol:



- 3-way exchange of shared secret

After auth established w/PAP: Doesn't re-authenticate!

- Leaves network vuln to attack

CHAP: Periodic challenges to make sure remote node still has valid passwd

- Passwd value: Var/changes unpredictably while link exists

After the PPP link establish phase complete: Local rtr sends challenge msg to remote node:

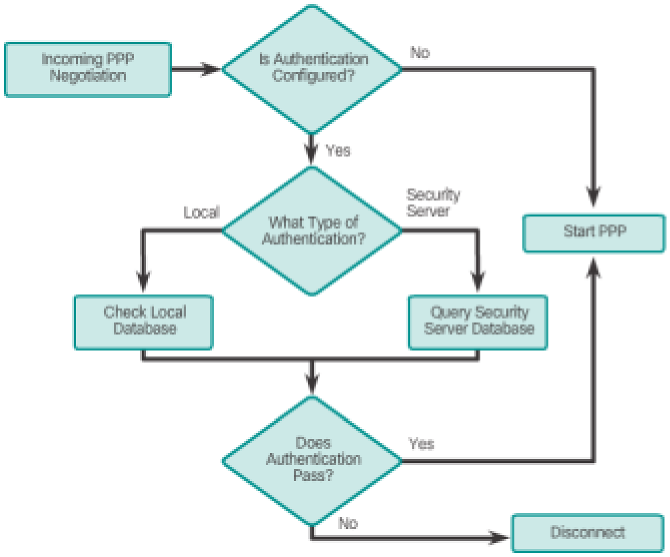
- Remote node: Responds w/value calc using 1-way hash function

- **Typically MD5:** Message Digest 5 based on passwd/challenge msg
- Local rtr checks response against its own calc of expected hash value
  - **If values match:** Initiating node ack's auth
  - **If NO match:** Initiating node terms connection

CHAP provides:

- Protection against playback attack: Uses var challenge value: Unique/unpredictable
- B/C of this: Resulting hash value also unique/random
- Use of repeated challenges limits time of exposure to attack
- Local rtr/3rd party auth server in control of freq/timing of challenges

PPP Encapsulation/Auth Process


Incoming PPP Negotiation
Is Authentication Configured?
No
Yes
What Type of Authentication?
Local
Security Server
Start PPP
Check Local Database
Query Security Server Database
Does Authentication Pass?
Yes
No
Disconnect

Config PPP Auth

ppp authentication [chap | chap pap | pap chap | pap] [if-needed] [list-name | default] [callin]

chap	Enables CHAP on serial int
pap	Enables PAP on serial int
chap pap	Enables both CHAP/PAP: CHAP before PAP
pap chap	Enables both PAP/CHAP: PAP before CHAP
if-needed	**Used w/TACACS/XTACACS** • Don't perform CHAP/PAP auth if usr already provided auth • Only avail on async ints
list-name	**Used w/AAA/TACACS+** • Specifies name of a list of TACACS+ methods of auth to use • If no list name specified: Sys uses default • Lists created with **aaa authentication ppp**
default	**Used w/AAA/TACACS+** • Created with **aaa authentication ppp**
callin	Specifies auth on incoming (received) calls only

Config PAP

R1

```
username R2 password class
int s0/0/0
ip address 10.0.1.1 255.255.255.252
ipv6 address 2001:db8:cafe:1::1/64
encapsulation ppp
ppp authentication pap
ppp pap sent-username R1 password class
```

R2

```
username R1 password class
int s0/0/0
```

```
ip address 10.0.1.2 255.255.255.252
ipv6 address 2001:db8:cafe:1::2/64
encapsulation ppp
pap authentication pap
ppp pap sent-username R2 password class
```

- Hostname on 1 router must match usrname on other router
- Passwds must match

Config CHAP

R1

```
username R2 password class
int s0/0/0
ip address 10.0.1.1 255.255.255.252
ipv6 address 2001:db8:cafe:1::1/64
encapsulation ppp
ppp authentication chap
```

R1

```
username R1 password class
int s0/0/0
ip address 10.0.1.2 255.255.255.252
ipv6 address 2001:db8:cafe:1::2/64
encapsulation ppp
ppp authentication chap
```

Troubleshooting

debug Used for troubleshooting/accessed from priv EXEC

- Output displays info about various rtr ops/traffic generated/received by it/error msgs
- Can consume sig amt of resources: Rtr is forced to process-switch packets being debugged
- Not used for monitoring

debug ppp Display info about op of PPP

- NCPs supported on either end of PPP connection
- Any loops that might exist
- Nodes that are/aren't properly negotiating
- Errors
- Causes for PAP/CHAP session failures
- Info specific to exchange of PPP connections using CBCP: Callback Control Protocol (used by MS clients
- Incorrect packet seq # info where MPPC compression enabled

```
debug ppp [ packet | negotiation | error | authentication | compression | cbcp ]
```

Param	Usage
packet	packets being sent/received: low lvl dumps
negotiation	PPP packets transmitted during PPP startup: Where options negotiated
error	Protocol errors/stats associated w/PPP connection negotiation/op
authentication	Auth protocol msgs: CHAP/PAP exchanges
compression	Info specific to exchange of PPP connections using MPPC • Useful for obtaining incorrect packet seq # info • Where MPPC enabled
cbcp	Protocol errors/stats associated w/PPP connection negotiations using MSCB

debug ppp packet Packet exchanges under normal PPP op: LCP state/LQM procedures/LCP magic #

debug ppp negotiation View LCP negotiations/auth/NCP negotiation

debug ppp error Display protocol errors/stats associated w/negotiation/op

Troubleshooting PPP Config w/Auth

Code Failure Values:

1	Challenge	**2**	Response
3	Success	**4**	Failure
id	ID # per LCP packet fmt	**len**	Packet length w/out header