TRƯỜNG ĐẠI HỌC BÁCH KHOA HÀ NỘI

**VIỆN CÔNG NGHỆ THÔNG TIN VÀ TRUYỀN THÔNG**

-----🙞🙜🕮🙞🙜----- ****

**Các giao thức liên kết dữ liệu**

|  |  |  |
| --- | --- | --- |
|  | | |
| Thực hiện:  MSHV: | **Nguyễn Đăng Nguyên**  **20152722** |  |

**Hà Nội 09/2019**

DANH MỤC HÌNH ẢNH

[Hình 1: Các giao thức liên kết dữ liệu chính](#_Toc21022176) 8

[Hình 2: Giao thức liên kết dữ liệu không đồng bộ đơn giản](#_Toc21022177) 9

[Hình 3: Các loại giao thức không đồng bộ thông dụng](#_Toc21022178) 10

[Hình 4: Cấu trúc gói tin trong giao thức không đồng bộ](#_Toc21022179) 10

[Hình 5: Giao thức liên kết dữ liệu đồng bộ](#_Toc21022180) 12

[Hình 6: Các gói tin trong giao thức hướng kí tự 1](#_Toc21022181)3

[Hình 7: BSC frames 1](#_Toc21022182)3

[Hình 8: Data frame 1](#_Toc21022183)4

[Hình 9: Trường tiêu đề 1](#_Toc21022184)6

[Hình 10: Multiblock Frame 1](#_Toc21022185)6

[Hình 11: Multi Frames Transmission 1](#_Toc21022186)7

[Hình 12: Control Frame 1](#_Toc21022187)7

[Hình 13: Cấu trúc Control Frame 18](#_Toc21022188)

[Hình 14: Transparent text 19](#_Toc21022189)

[Hình 15: Các giao thức hướng bit 19](#_Toc21022190)

[Hình 16: Ward Christensen 21](#_Toc21022191)

[Hình 17: Cấu trúc gói 22](#_Toc21022192)

[Hình 18: Chuck Forsberg 24](#_Toc21022193)

[Hình 19: Điểm nối điểm với một sơ cấp và một thứ cấp 30](#_Toc21022194)

[Hình 20: Đa diểm với một sơ cấp và nhiểu thứ cấp 30](#_Toc21022195)

[Hình 21: Điểm nối điểm với hai sơ cấp và hai thứ cấp 30](#_Toc21022195)

[Hình 22: Cấu trúc khuôn dạng của frame của HDLC 32](#_Toc21022195)

[Hình 23: Cấu trúc khuôn dạng của frame của HDLC 33](#_Toc21022195)

[Hình 24: Cấu trúc khuôn dạng của frame của HDLC 34](#_Toc21022195)

[Hình 25: Sơ đồ kết nối của giao thức PPP 35](#_Toc21022195)

**THUẬT NGỮ**

|  |  |
| --- | --- |
| **Ký hiệu viết tắt** | **Ý nghĩa** |
| PPP | Point to Point protocol |
| HDLC | High Level Data Link Control |
| LLC | Logical Link Control |
| BLAST | Blocked ASynchronous Transmission |

**TH**

# MỤC LỤC

DANH MỤC HÌNH ẢNH

**THUẬT NGỮ**

[MỤC LỤC](#_Toc16868449)

[CHƯƠNG 1 TỔNG QUAN VỀ CÁC GIAO THỨC LIÊN KẾT DỮ LIỆU](#_Toc16868451)

[1.1 Giới thiệu tông quan về liên kết dữ liệu](#_Toc16868452)

[1.2 Giao thức không đồng bộ](#_Toc16868453)

[1.3 Giao thức đồng bộ](#_Toc16868453)

[1.4 Giao thức hướng ký tự](#_Toc16868453)

[1.5 Giao thức hướng bit](#_Toc16868453)

[CHƯƠNG 2 CÁC GIAO THỨC LIÊN KẾT KHÔNG ĐỒNG BỘ](#_Toc16868454)

[2.1 XMODEM](#_Toc16868455)

[2.2 YMODEM](#_Toc16868456)

[2.3 ZMODEM](#_Toc16868457)

[2.4 BLAST](#_Toc16868458)

[2.5 KERMIT](#_Toc16868458)

[CHƯƠNG 3 CÁC GIAO THỨC LIÊN KẾT ĐỒNG BỘ](#_Toc16868454)

[3.1 HDLC](#_Toc16868458)

[3.2 PPP](#_Toc16868458)

**KÊT LUẬN**

[TÀI LIỆU THAM KHẢO](#_Toc16868460)

# CHƯƠNG 1 TỔNG QUAN VỀ CÁC GIAO THỨC LIÊN KẾT DỮ LIỆU

## Giới thiệu tổng quan về liên kết dữ liệu

**Tầng liên kết dữ liệu**

Trong mạng và truyền thông, tầng liên kết dữ liệu là tầng hai của mô hình bảy tầng OSI. Nó đáp ứng các yêu cầu phục vụ của tầng mạng và phát sinh các yêu cầu phục vụ gửi tới tầng vật lý.

Tầng liên kết dữ liệu là tầng mạng có nhiệm vụ truyền dữ liệu giữa các nút mạng kề nhau trong một mạng diện rộng hoặc giữa các nút trong cùng một segment mạng cục bộ. Tầng liên kết dữ liệu cung các các phương thức chức năng và thủ tục để truyền dữ liệu giữa các thực thể mạng và có thể cung cấp phương tiện để phát hiện và có thể sửa lỗi có thể nảy sinh tại tầng vật lý. Ví dụ về các giao thức liên kết giữ liệu là giao thức Ethernet cho các mạng cục bộ và các giao thực PPP, HDLC và ADCCP cho các kết nối điểm tới điểm (point-to-point).

Nhiệm vụ của liên kết dữ liệu là gửi thông tin từ nới này đến nới khác. Tại tầng này, một số nút mạng không cần phải có khả năng gửi tới mọi nút, mà chỉ cần gửi tới một số nút khác. Trong các mối quan hệt xã hội cũng vậy, người ta cần quen biết ít nhất một người khác, nhưng không nhất thiết phải biết ông Nguyên ở Wankada, Nam Phi.

Tầng liên kết dữ liệu cung cấp dịch vụ chuyển dữ liệu qua các liên kết vật lý. Việc chuyển đó có thể đáng tin cậy hoặc không đáng tin cậy, nhiều giao thức liên kết dữ liệu không có ackowledgement (các thông điệp báo rằng đã nhận được một frame và đã chấp nhận frame đó), một số giao thức liên kết dữ liệu thậm chí còn không có bất cứ dạng checksum nào để kiếm tra lỗi truyền. Trong các trường hợp đó, các giao thức ở các tầng cao hơn phải cung cấp các chức năng điều khiển lưu lượng (flow control), kiểm lỗi và xác nhận và truyền lại (ackowledgments and retransmission).

Tầng này đôi khi được chia làm 2 tầng con. Tầng con thứ nhất có tên là Điều khiển liên kết logic (Logical Link Control, viết tắt LLC). Tầng con này multiplex các giao thức hoạt động phía trên tầng liên kết dữ liệu, và theo tùy chịn có thể cung cấp chức năng điều khiển lưu lượng, ackowledgment, và khôi phục lỗi.

Tầng con thứ hai có tên Điều khiển truy nhập môi trường (Media Acess Control, viết tắt MAC). Tầng con này quyết định tại mỗi thời điểm ai sẽ được phép truy nhập môi trường truyền dẫn. Có hai dạng điều khiển truy nhập môi trường phân tán và điều khiển tập trung. Cả hai đều có thể so sánh với việc liên lạc của con người:

Trong một mạng gồm một số người đang nói, nghĩa là trong một cuộc trò chuyện, người ta nhìn những người xung quanh để biết ai có vẻ sắp nói. Nếu hai người nói cùng lúc, họ sẽ ngừng lại và bắt đầu một trò chơi nhường nhịn rắc rối gồm các câu “Không, anh nói trước đi”

Trong Nghị viện Anh, người chủ tọa quyết định khi nào ai được nói, và chủ tọa được quyền hô “Trật tự” nếu có ai đó phạm luật.

Tầng con Điều khiển Truy nhập Môi trường còn quyết định một frame dữ liệu kết thúc tại đâu và frame tiếp theo bắt đầu từ đâu. Trong mạng thư tín, mỗi là thư là một frame dữ liệu, ta có thể biết nó bắt đầu và kết thúc ở đâu vì nó nằm trong một chiếc phong bì. Ta cũng có thể chỉ ra rằng một là thư sẽ mở đầu bằng một ngữ như “Em thân yêu”, và kết thúc bằng một câu kiểu như “Nhớ em nhiều”.

Trong các mạng cục bộ theo chuẩn IEEE 802, tầng liên kết dữ liệu được chia hai tầng con MAC và LLC, điều đó có nghĩa là có thể dùng giao thức LLC IEEE 802.2 cùng với mọi tầng MAC IEEE 802, chẳng hạn Ethernet, Token Ring, IEEE 802.11, vv.., cũng như các tầng MAC không theo chuẩn IEEE 802 chẳng hạn FDDI. Các giao thức liên kết dữ liệu khác, chẳng hạn Cisco HDLC, sử dụng HDLC’s low-level framing như một tầng MAC khi kết hợp với các tầng LLC khác nhau.

**Giao thức là gì?**

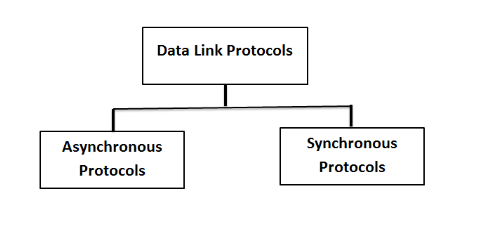
Giao thức (protocol) được hiểu là tập các luật hay quy ước nhằm thực hiện một nhiệm vụ đặc thù, trong nghĩa hẹp hơn giao thức là tập các luật hay đặc tính được dùng để thiết lập một hay nhiều lớp trong mô hình OSI.

Giao thức trong truyền dữ liệu là tập các luật hay các đặc tính được dùng để thiết lập một hay nhiều lớp trong mô hình OSI.

Giao thức kết nối dữ liệu chia ra làm hai nhóm con:

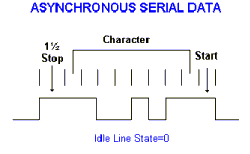
Giao thức không đồng bộ: Xử lý các ký tự trong dòng một cách độc lập.

Giao thức đồng bộ: Dùng nguyên dòng bit để chuyển sang thành kí tự có cùng chiều dài.



Hình 1

## Giao thức không đồng bộ



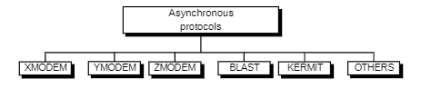
Hình 2

Các giao thức không đồng bộ được phát triển sớm trong lịch sử viễn thông. Nó trở nên phổ biến với việc phát minh ra máy đánh chữ và được sử dụng để gửi các bức điện tín trên khắp thế giới.

Các hệ thống không đồng bộ gửi các bytes dữ liệu giữa người gửi và người nhận bằng cách đóng gói dữ liệu vào trong một chiếc “phong bì”. Phong bì giúp vận chuyển ký tự qua liên kết vật lý giữa người gửi và người nhận. Transmitter tạo ra các phong bì này và người nhận sử dụng nó để trích xuất giữ liệu. Mỗi ký tự (1 byte dữ liệu) mà người gửi truyền đi được thêm vào đầu một bit bắt đầu và thêm vào cuối một bit kết thúc. 2 bits này giúp người nhận và người gửi có thể đồng bộ hóa.

Các phương thức truyền không đồng bộ này phù hợp với tốc độ dưới 32000 bit mỗi giây. Ngoài ra, các tín hiệu được gửi không chứ bất kỳ thông tin nào có thể sử dụng để xác thực nếu nó được nhận mà không sửa đổi. Điều này có nghĩa là các giao thức này không chứa thông tin để phát hiện lỗi và dễ bị lỗi.

Ngoài ra với mỗi ký tự gửi đi, hai bits cũng được gửi đi. Với một tài liệu văn bản chứa 1000 ký tự, mỗi ký tự 8 bits, do đó tổng số bit gửi đi sẽ lên tới 10000 bits hay 1250 ký tự, tức là đã có 250 ký tự được gửi theo kèm do các bit start và stop. Điều này thể hiện một chi phí lớn trong việc gửi dữ liệu, do đó các giao thức này trở thành một phương tiện không hiệu quả để gửi một lượng lớn dữ liệu.



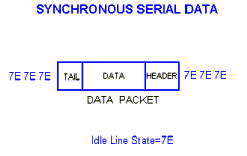
Hình 3

Các giao thức chủ yếu được dùng trong các modem.

Các giao thức này có điểm yếu là truyền chậm (do tồn tại các start bit, stop bit) nên hiện nay, đã có các giao thức truyền tốc độ cao dùng cơ chế đồng bộ.

Các giao thức không đồng bộ phù hợp với truyền dữ liệu tốc độ thấp và không có các phương pháp kiểm tra lỗi.

## Giao thức đồng bộ



Hình 4

Một trong những vấn đề liên quan đến truyền không đồng bộ là chi phí cao liên quan đến truyền dữ liệu. Ví dụ, đối với mỗi 8 bits được truyền thì phải có ít nhất 2 bits thêm. Đối với số lượng lớn dữ liệu, điều này nhanh chóng cộng lại. Ví dụ 1000 ký tự thì cần 12000 bits. Một vấn đề khác là thiếu hoàn toàn bất kỳ hình thức phát hiện lỗi. Điều này có nghĩa là người gửi không có cách nào để biết liệu người nhận có nhận ra chính xác dữ liệu được truyền không.

Trong các giao thức đồng bộ, hiệu quả cả hơn đạt được bằng cách nhóm các ký tự lại với nhau và loại bỏ các bit bắt đầu và kết thúc cho mỗi kí tự. Ta vẫn bao bọc lấy thông tin gửi như các giao thức không đồng bộ, tuy nhiên thay vì các ký tự thì là nhiều ký tự hơn giữa các chuỗi bắt đầu và kết thúc. Ngoài ra các bit start và stop được thay thế bằng một định dạng mới cho phép linh hoạt hơn. Một chuỗi kết thúc thêm được thêm vào để thực hiện kiểm tra lỗi.

Một chuỗi bắt đầu, được gọi là một tiêu đề, tiền tố mỗi ký tự và một chuỗi dừng, được gọi là đuôi. Đuôi được mở rộng để bao gồm các mã kiểm tra, được trình phát chèn vào và được người nhận sử dụng để xác định xem khối dữ liệu của các ký tự có được nhận có lỗi hay không.

Theo cách này, truyền dẫn đồng bộ khắc phục được 2 thiếu sót chính của phương pháp không đồng bộ, đó là không hiểu quả và thiếu phát hiện lỗi.

Có các biến thể của truyền dẫn đồng bộ, được chia làm hai nhóm chính, cụ thể là hướng kí tự và hướng bit. Đồng bộ hóa nhị phân là một ví dụ về định hướng ký tự, điều khiển liên két dữ liệu mức cacso (HDLC) là một ví dụ về định hướng bit.

Trong truyền dữ liệu không đồng bộ, nếu không có dữ liệu để truyền, không có gì được gửi. Chúng dựa vào bit start để khởi động, và sau đó bắt đầu để chuẩn bị giải mã ký tự đến. Tuy nhiên, trong truyền dẫn đồng bộ, vì bit start đã bị loại bỏ, bộ thu phải được giữ ở trạng thái sẵn sàng. Điều này đạt được bằng cách gửi một mã đặt biệt bởi người phát bất cứ khi nào nó không có dữ liệu để gửi.

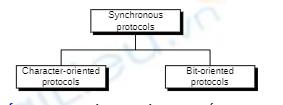
Trong các giao thức hướng bit, trạng thái line idle được đổi thành 7E, đồng bộ hóa máy thu với người gửi. Các bit start và stop bị loại bỏ, và mỗi ký tự được kết hợp với các bit khác tạo thành một gói dữ liệu.

Dữ liệu người dùng được prefix với trường header, và suffix với một trường trailer nơi mà chứa giá trị tổng các kí tự gửi đi.

Trường Header được dùng để lưu thông tin địa chỉ (của người gửi và người nhận), loại packet và dữ liệu điều khiển(như sự kiên, …vv)

Trường Data chứa dữ liệu của người dùng.

Trường Tail chứa thông tin mà người nhận có thể check để kiểm tra lỗi.



Hình 5

Tốc độ truyền đồng bộ là chọn lựa tố hơn so với không đồng bộ trong công nghệ mạng LAN, WAN hay MAN.

Giao thức hướng kí tự (còn được gọi là giao thức hướng byte) diễn dịch các frame hay gói cần truyền thành các ký tự liên tiếp nhau, mỗi ký tự gồm một byte(8 bit). Tất cả các thông tin đều ở dạng hiện hữu của ký tự (mã ASCII).

Giao thức hướng bit: diễn dich dữ liệu hay gói cần truyền thành của các bit đơn, tạo nghĩa cho chúng bằng cách sắp xếp vị trí trong frame và bằng phương thức xếp đặt chúng với các bit khác. Các thông tin điều khiển trong giao thức này có thể dùng một hay nhiều bit, tùy theo kiểu thông tin trong mẫu.

Trong giao thức hướng ký tự, các frame hay gói được chuyển thành chuỗi các kí tự. Trong giao thức hướng bit, các frame hay gói được diễn dich thành chuỗi các bit.

**1.3 Các giao thức hướng ký tự (Binary synchronous communication)**

Giới thiệu

+ Dùng cho cấu hình điểm – điểm và đa điểm

+ Cơ chế truyền bán song công (half-duplex).

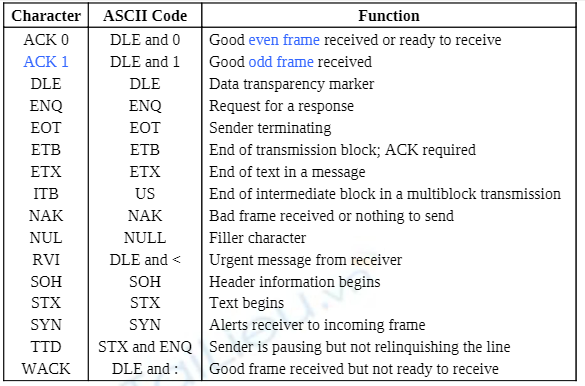
+ Dùng phương pháp kiểm tra lỗi và điều khiển lưu lượng stop and wait ARQ (BSC không hỗ trợ chế độ full-duplex hay giao thức của sổ trượt).

Các ký tự điều khiển:

+ Ký tự ACK không được dùng trong giao thức này.

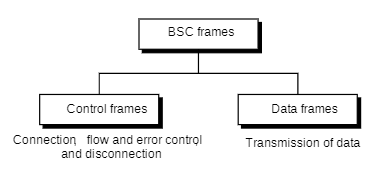
+ BSC dùng phương pháp stop and wait ARQ và ACK phải là ACK0 hay ACK1 cho các frame dữ liệu liên tiếp nhau.

Trong bảng biểu diễn các ký tự dùng mã ASCII, chú ý là các ký tự điều khiển có thể biểu diễn bằng nhiều ký tự.



**Hình 6**

BSC frames

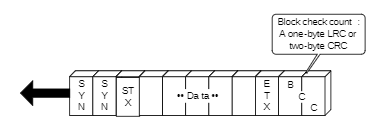


Hình 7

+ Control frame: Chỉ chứa các thông tin về điều khiển.

+ Data frame: chứa các thông tin về dữ liệu, nhưng cũng có các thông tin điều khiển dùng trong thông tin này.

Data frame



Hình 8

+ Chiều mũi trên là chiều truyền.

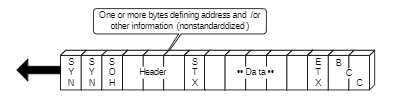
+ Frame có hai ký tự đồng bộ hay nhiều hơn. Các ký tự này cảnh báo máy thu là frame mới đến và cung cấp bit pattern cho máy thu nhằm đồng bộ thời gian với máy phát. Ví dụ mã ASCII của SYN là 00010110. Bit đầu thứ 8 của byte thường được thêm vào các số 0. Hai ký tự SYN cùng nhau sẽ có dạng 0001011000010110.

+ Tiếp theo 2 ký tự đồng bộ thì bắt đầu ký tự văn bản (STX: 0000010 start of text). Các ký tự này báo cho máy thu là đã hết thông tin điều khiển và byte kế tiếp là dữ liệu. Dữ liệu hay văn bản có thể là một số các ký tự. Ký tự chấm dứt text (end of text: ETX: 00000011) cho biết có sự chuyển tiếp từ văn bản sang nhiều kí tự điều khiển.

+ Sau cùng, một hay hai ký tự được gọi là đếm-kiểm tra (block check count: BCC) được thêm vào để kiểm tra lỗi. Trường BCC có thể có một ký tự kiểm tra lỗi dạng LRC hay hai ký tự kiểm tra lỗi CRC.

Trường tiêu đề

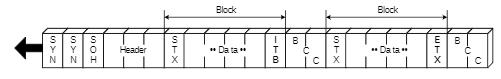
Một frame đơn như vừa mô tả thường ít được dùng, do phải có thểm địa chỉ của thiết bị thu, địa chỉ của thiết bị gửi,v à số nhận dạng frame (0 hay 1) cho trường hợp stop and wait ARQ (xem hình bên dưới). Các thông tin này thường được chứa trong một trường đặc biệt gọi là tiêu đề (header), được bắt đầu bằng kí tự start of header (SOH), Tiêu đề này đến sau ký tự SYN và trước ký tự STX, mọi thông tin nhận sau trường SOH nhưng ký tự STX là các thông tin tiêu đề.



Hình 9

Mutilblock Frame

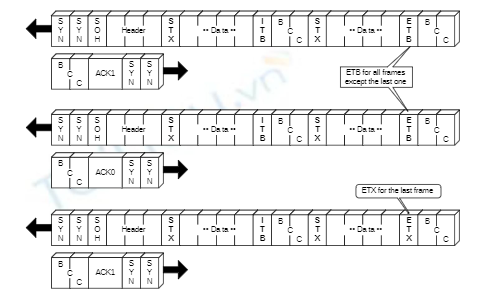
Khi chiều dài của một khối (block) tăng thì xác suất xuất hiện lỗi cũng gia tăng theo. Càng nhiều bit trong một frame thì khả năng bị lỗi càng cao, làm cho việc phát hiện lỗi càng trở lên khó khăn. Do đó, các văn bản trong bản tin thường đưuọc chia ra thành nhiều block. Mỗi block (trừ block cuối cùng) đều bắt đầu với một ký tự STX và chấm dứt bằng một khối text trung gian (ITB: intermediate text block). Block cuối bắt đầu dùng STX nhưng tận cùng dùng ETX. Liền kế ngay sau mỗi ITB hay ETX là trường BBC. Theo cách này, thi máy thu có thể kiểm tra lỗi cho từng block riêng biệt, cho phép gia tăng khả năng phát hiện lỗi. Nếu một block có lỗi thì cả frame phải được chuyển lại. Sau khi ETX đã đến và BCC cuối cùng đã được kiểm tra xong, máy thu mới gửi độc một xác nhận cho toàn frame. Hình vẽ bên dưới minh họa cấu trúc của frame nhiều block, ví dụ này chỉ dùng 2 block, tuy nhiên trong thực thế có thể nhiều hơn hai.



Hình 10

Truyền nhiều frame (Multiframe Transmission)

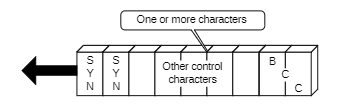
Trong ví dụ vừa rồi, một frame đơn mang toàn bản tin. Sau mỗi frame, bản tin được chấm dứt và kiểm tra chuyển sang đường thứ hai (thí dụ trong chế độ full-duplex). Một số bản tin, thường dai để có thể đặt vào format của một frame đơn, như thế, máy phát sẽ chia bản tin ra không những theo nhiều block mà còn thành nhiều frame. Nhiều frame có thể chuyển liên tục một bản tin. Để máy thu biết được là phần cuối của frame chưa phải là phần cuối của bản tin thì kí tự ETX trong tất cả các frame (trừ frame cuối cùng) được thay thế bằng kí tự ETB (End of transmission block). Máy thu phải xác nhận mỗi frame riêng biết nhưng không thể điều khiển toàn kết nỗi cho đến khi tìm được ký tự ETX tại frame cuối.



Hình 11

Frame điều khiển (Control Frames)

Một frame kiểm tra không thể bị hiểu lầm thành một kí tự kiểm tra. Một frame điều khiển được một thiết bị dùng để gửi tín hiệu điều khiển đề củng cố thông tin, cho thiết bị khác. Một frame điều khiển chứa các ký tự điều khiển nhưng không có data, chúng chứa các thông tin đặc biệt để tự vận hành lớp kết nối dữ liệu.



HÌnh 12

Control frame có ba mục đích chính:

+ Thiết lập kết nối.

+ Duy trì lưu lượng và kiểm tra lỗi trong khi truyền dẫn.

+ Kết thúc kết nối.

# 

Hình 13

Data transparency

Nếu một trường text khi truyền gồm mẫu 8 bit giống như ký tự điều khiển của BSC thì máy thu sẽ biên dịch thành một ký tự và hủy ý nghĩa của bản tin.

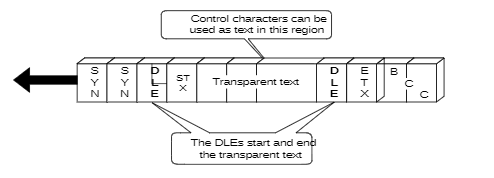
Thí dụ: Máy thu nhìn thấy một chuối bit 0000011 được đọc là ký tự ETX, nên như ta đã biết, khi máy thu nhận ra kí tự ETX, nó sẽ cho rằng 2 byte kế tiếp là BCC và bắt đầu kiểm tra lỗi. Thực ra mẫu 0000011 ở đây là dữ liệu chứ không phải là thông tin điều khiển. Hiểu lầm này được gọi là thiếu thông tin minh bạch.

Để một giao thức là hữu ích thì giao thức này phải là minh bạc, tức là có thể mang bất kì tổ hợp bit như dữ liệu mà không bị hiểu lầm là thông tin điều khiển.

Data transparecy trong thông tin số liệu được hiểu là ta có thể truyền các tổ hợp bit dữ liệu bất kì.

Tính minh bạch của BSC có thể được thực hiện thông qua quá trình bit nhồi (bit stuffing). Bao gồm hai tác động: định nghĩa vùng văn bản transperency dùng ký tự data link escape (DLE) và xử lý các ký tự DLE trong vùng transparency bằng các ký tự DLE extra.

Để định nghĩa vùng transparency, ta chèn vào một ký tự DLE ngay trước ký tự STX tại lúc bắt đầu trường text và DLE khác ngay trước ETX (hay ITB, ETB) tại cuối trường text. DLE đầu cho máy thu biết là text có thể chứa các ký tự điều khiển và phải bỏ qua chúng. DLE cuối cho máy thư biết là vùng transparency đã chấm dứt.



Hình 14

**1.3 Các giao thức hướng bit**

# 

Hình 15

Các bit được nhóm thành các mẫu tạo ký tự.

So sánh với các phương pháp theo hướng byte thì giao thức theo hướng bit có thể dống gói nhiều thông tin hơn trong một frame ngắn hơn và tránh dược vấn đề transparency.

Các giao thức theo hướng bit ngày nay ngày càng nhiều và dần phát triển thành các chuẩn. Đa số chúng được các nhà sản xuất thiết kế nhằm hỗ trợ cho các sản phẩm của mình. Trong số đó, chuẩn HDLC do ISO thiết kế và ngày càng trở thành cơ sở của các giao thức hướng bit hiện nay.

Năm 1975, IBM đã đi đầu trong việc phát triển giao thức theo hướng bit với giao thức SDLC và yêu cầu ISO chấp nhận để đưa ra làm chuẩn. Năm 1979, ISO trả lời bằng cách đưua ra HDLC, phát triển từ SDLC. Việc ISO chấp nhận chuẩn HDLC làm giao thức này được nhiều tổ chức chấp nhận và mở rộng. ITU-T là tổ chức đầu tiên chập nhận HDLC. Từ 1981, ITU-T đã phát triển một tập các giao thwucs được gọi là link access protocol (LAPs, LAPB, LAPD, LAPM, LAPX, ...) đều dựa trên HDLC. Các giao thức khác (ví dụ Frame Relay, PPP,...) được cả ITU-T và ANSI cũng dựa trên HDLC, và làm giao thwucs cho mạng LAN. Như thế hầu hết các giao thức theo hướng bit đều xuất phát từ HDLC, nên đó chính là nền tảng đề tìm hiểu các giao thức khác.

# CHƯƠNG 2 CÁC GIAO THỨC LIÊN KẾT KHÔNG ĐỒNG BỘ

## XMODEM

**Giới thiệu**



Hình 16

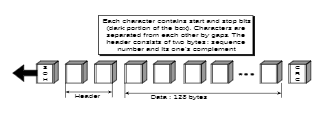
**XMODEM** là một giao thức truyền tệp đơn giản được phát triển dưới dạng quick hack bởi Ward Christensen để sử dụng trong chương trình thiết bị đầu cuối MODEM.A SM của anh ấy. Nó cho phép người dùng truyền tệp giữa các máy tính của họ khi cả hai bên sự dụng MODEM. Keith Petersen đã thực hiện một bản cập nhật nhỏ để luôn bật “Chết độ im lặng” và gọi kết quả là XMODEM.

XMODEM trở lên cực kỳ phổ biến trong thị trường hệ thống bảng tin sớm (BBS), phần lớn là do nó đơn giản để thực hiện. Nó cũng khá kém hiệu quả khi tốc độ modem tăng lên, vấn đề này đã dẫn đến việc phát triển một số phiên bản sửa đổi của XMODEM để cải thiện hiệu suất hoặc giải quyết các vấn đề khác với giao thức.

XMODEM, giống như hầu hết các giao thức truyền tệp, chia dữ liệu gốc thành một loạt các gói được gửi đến người nhận, cùng với thông tin bổ sung cho phép người nhận xác định xem gói đó có được nhận chính xác hay không.

XMODEM dùng để truyền file dùng đường truyền điện thoại giữa các PC. Nó là một giao thức stop and wait ARQ và truyền bán song công (half-duplex).

**Cấu trúc gói**



Hình 17

Trường đầu tiên là một byte tiêu để header (start of header: SOH).

Trường thứ hai là header gồm 2 byte: byte đầu là một chuối bit mang giá trị số frame và byte thứ hai được dùng để kiểm tra giá trị hợp pháp của chỗi bit.

Trường cố định gồm 128 byte dữ liệu (binary, ASCII, Boole, text).

Trường cuối cùng là CRC, chỉ dùng kiểm tra lỗi trong trường dữ liệu.

Bắt đầu truyền bằng cách gửi một frame NAK từ máy thu đến máy phát.

Mỗi khi máy phát gửi đi một frame thì phải chờ tín hiệu ACK trước khi gửi tiếp frame kế.

Nếu máy phát nhận được NAK thì frame vừa gửi sẽ được gửi lại.

Một frame cũng có thể được gửi lại nếu máy phát không nhận được tín hiệu xác nhận sau một thời gian định trước.

Ngoài tín hiệu ACK và NAK, máy thu còn có thể nhận được tín hiệu CAN (cancel), yêu cầu hủy việc truyền.

**Cấu hạn chế**

XMODEM được thiết kế đơn giản, không có nheieuf kiến thức về các giao thức truyền tếp khác. Do tính đơn giản của nó, có một số lỗi rất cơ bản có thể khiến việc chuyển tiền không thành công hoặc tệ hơn là dẫn đến một tệp không chính xác mà giao thức này không chú ý.

XMODEM được viết cho các máy CP/M và mang nhiều dấu ấn của hệ điều hành đó. Các tệp của CP/M luôn là bội số của 128 byte và kết thúc của chúng được đánh dấu trong một khối có ký tự <EOT>. Những đặc điểm này đã được thêm trực tiếp vào XMODEM. Tuy nhiên, các hệ điều hành khác không có tính năng đặc biệt này và việc giới thiệu rộng rãi MS-DOS vào những năm 1980 đã khiến XMODEM phải được cập nhật để thông báo <EOT> hoặc <EOF> là điểm đánh dấu cuối tập tin.

Vì giao thức XMODEM yêu cầu người gửi dừng lại và chờ một tin nhắn <ACK> hoặc <NAK> tin nhắn từ người nhận, nên nó có xu hướng khá chậm. Trong kỉ nguyên của modem 300 bit/s, toàn bộ gói 132 byte chỉ cần hơn 3,5 giây để gửi (132 bytes \* 8 bits mỗi byte / 300 bits mỗi giây). Nếu sau đó phải mất 0.2 giây để người nhận <ACK> quay lại người gửi và gói tiếp theo để bắt đầu nhận người nhận (0,1 giây theo cả hai hướng), tổng thời gian cho mỗi gói sẽ là 3,7 giây, thông lượng chỉ hơn 92%.

Khi tốc độ modem tăng lên, độ trễ cố định cần thiết để gửi <ACK>/<NAK> tăng tỷ lệ với thời gian cần thiết để gửi gói. Chẳng hạn, ở tốc độ 2400 bits/ giây, các gói chỉ mất 0.44 giây để gửi, do đó, nếu <ACK> và <NAK> vẫn mất 0.2 giây để quay lại (đây là độ trễ trong mạng, không phải thông lượng)

## 2.3 YMODEM

**Giới thiệu**



Hình 18

YMODEM là một giao thức truyền tệp được sử dụng giữa các máy vi tính được kết nối với nhau bằng modem. YMODEM được phát triển bởi Chuck Forsberg với tư cách là người kế thừa của XMODEM và MODEM7, và lần đầu tiên được triển khai trong chương trình CP/M YAM của mình. Nó chính thức được đặt tên là YMODEM vào năm 1985 bởi Ward Christensen.

YMODEM ban đầu cơ bản giống như XMODEM ngoại trừ việc nó đã gửi tên, kích thước và dấu thời gian của tệp trong một khối XMODEM thông thường. Gửi kích thước tệp đã giải quyết vấn đề của XMODEM về phần đệm không cần thiết ở cuối tệp.

Forsberg đã xây dựng tiêu chuẩn với một số tính năng tùy chọn, tin rằng các lập trình viên sẽ muốn triển khai càng nhiều càng tốt trên bất kỳ nên tảng nào. Ông đã thật vọng khi thấy rằng phần lớn các triển khai thực sự không cung cấp kích thước khối lớn hơn 1 kilobyte với CRC-16, trong khi tiếp tục sử dụng tên YMODEM, kết quả là một số lượng lớn các YMODEM không tương thích lẫn nhau.

**Cấu trúc**

Giao thức YMODEM về cơ bản tương tự như XMODEM, ngoài một số đặt điểm đặc biệt sau

+ Đơn vị dữ liệu là 1024 byte

+ Dùng hai tín hiệu CAN để hủy việc truyền tin.

+ Dùng phương pháp kiểm tra lỗi IIU-T, CRC-16.

+ Có thể truyền đồng thời nhiều file.

**Các biến thể**

YMODEM-1Ksử dụng kích thước khối 1 kilobyte thay vì 128 byte tiêu chuẩn. Các khối 1K là một tùy chọn trong tiêu chuẩn YMODEM ban đầu, nhưng biến thể này thiếu các tính năng còn lại và được mô tả như một biến thể của XMODEM-1k

YMODEM-g là một biến thể phát trực tiếp được sử dụng cho các kết nối không có lỗi, Nó đơn giản là loại bỏ CRC và không đợi ACK được nhận trước khi gửi gói tiếp theo. Giao thức này nhanh hơn YMODEM viif không có kiểm tra lỗi được thực hiện. Tuy nhiên dù giao thức này có vẻ nhanh hơn ZMODEM nhưng nó vẫn hiếm khi được sử dụng. Điều này là một phần do thiếu chức năng khác, những cũng có một vấn đề nghiêm trọng hơn. Trước khi 16550 UART được sử dụng rộng rãi, YMODEM-g không thể nhận ra các nguy cơ về tràn bộ đệm trên cổng nối tiếp.

## 2.4 ZMODEM

**Giới thiệu**

ZMODEM là một giao thức truyền tệp được phát triển bởi Check Forsberg vào năm 1986, trong một dự án do Telenet tài trợ để cải thiện việc chuyển tệp trên mạng X.25 của họ. Ngoài hiệu suất được cải thiện đáng kể so với các giao thức cũ, ZMODEM cũng cung cấp khả năng chuyển đổi khởi động lại, tự động khởi động bởi người gửi, CRC 32 bit và kí tự điều khiển hỗ trợ dịch truyển 8-bit clean cho phép sử dụng nó trên các mạng mà sẽ không vượt qua các ký tự điều khiển. ZMODEM trở lên cực kí phổ biến trên các hệ thống bảng tin (BBS) vào đầu những năm 1990, thay thế các giao thức trước đó như XMODEM và YMODEM.

ZMODEM là sử kết hợp của cả 2 giao thức XMODEM và YMODEM.

**Cải tiến**

Một trong những cải tiến quan trọng trong ZMODEM là giới thiệu hỗ trợ cửa sổ trượt để cải thiện hiệu suất.

Giao thức cửa sổ trượt giúp tránh được vấn đề độ trễ thời gian khi người gửi phải đợi tin nhắn ACK từ người nhận. Người nhận gửi cả ACK (hoặc NAK nếu bị lỗi) cùng với số gói mạ người đó đang xác nhận, người gửi có thể xử lý chúng và gửi lại gói bị lỗi theo yêu cầu của người nhận, điều này loại bỏ độ trễ xác nhận khi không có lỗi, chỉ để lại độ trễ truyền dữ liệu và phát hiện lỗi.

Hiệu suất của ZMODEM được cải thiện so với các giao thức phổ biến trước đây, nó thường thay thế ngay cả các giao thức đặc biệt như YMODEM-g.

**Biến thể**

Một số phiên bản biến thể của ZMODEM đã xuất hiện:

+ ZedZap: là một biến thể với 8 khối kb cho hiệu năng tốt hơn trên các modem tốc độ cao.

+ LeechZmodem

+ DSZ (DOS Send ZMODEM).

+ GSZ (Graphical Send ZMODEM).

## 2.5 BLAST

**Giới thiệu**

BLAST (Blocked Asynchronous Transmission) là sản phẩm của Paul Charbonnet, Jr một cựu nhân viên bán hàng ở Data General. Phiên bản gốc của nó được thiết kế và triển khai cho dòng m áy tính tổng hợp dữ liệu Nova bởi GW Smith, cựu kỹ sư hệ thống của Trung tâm nghiên cứu BorgWarner, người đã phát triển giao thức “ack-nak” cơ bản cho ứng dụng đo từ xa nói trên, bây giờ đã tạo ra một giao thức hoàn toàn mới với tất cả các tính năng được đề cập ở trên và anh ấy đã nghĩ ra từ viết tắt “BLAST”.

BLAST là giao thức không đồng bộ duy nhất đã được sử dụng vào lĩnh vực điện toán thập niên 1980 với tất cả các tính năng sau:

+ Mã hóa dữ liệu theo định hướng bit.

+ Phát hiện lỗi CRC (kiểm tra dự phòng theo chu kỳ).

+ Sử dụng cửa sổ trượt.

+ Chọn lọc và truyền lại các khối bị lỗi.

+ Truyền dữ liệu 2 chiều đồng thời.

BLAST mạnh hơn XMODEM, giao thức dùng chết độ full-duplex và dùng phương pháp kiểm soát lưu lượng dạng của số trượt.

## 2.6 KERMIT

**Giới thiệu**

Vào cuối những năm 1970, người dùng máy tính lớn của đại học Columbia chỉ có 35 kilobyte dung lượng lưu trữ mỗi người. Kermit được phát triển tại trường đại học dể học sinh có thể di chuyển các file giữa họ và đĩa mềm tại nhiều microcomputer xung quanh khuôn viên trường. Các máy tính lớn của IBM đã sử dụng một bộ ký tự EBCDIC còn các máy tính CP/M và DEC sử dụng ASCII, do đó, chuyển đổi giữa hai bộ kí tự là một trong những chức năng ban đầu được tích hợp trong Kermit. Việc chuyển tập tinh đầu tiên với Kermit xảy ra vào tháng 4 năm 1981. Giao thức ban đầu được thiết kế vào năm 1981 bởi Frank da Cruz và Bill Catchings.

Đại học Columbia đã phối hợp phát triển các phiên bản Kermit cho nhiều máy tính khác nhau tại trường đại học và các nơi khác, và phân phối phần mềm miễn phí. Kermit cho máy tính cá nhân của IBM trở lên cực kỳ phổ biến. Năm 1986, trường đại học thành lập dự án Kermit tiếp quản và bắt đầu thu phí sử dụng thương mại, dự án đã tự túc về tài chính. Để sử dụng phi thương mại, đại học Columbia đã tuyên bố rằng “Kermit là dành cho tất cả mọi người để sử dụng và chia sẻ, một khi bạn nhận được nó, hãy thoải mái chuyển nó cho bạn bè và đồng nghiệp của bạn. Mặc dù nó có bản quyền và không thuộc phạm vi công cộng, chúng tôi chỉ yêu cầu bạn không cố bán nó vì lợi nhuận và bạn chỉ sử dụng nó cho mục đích hòa bình và nhân đạo”.

Kể từ mùng 1 tháng 7 năm 2011, đại học Columbia đã ngừng tổ chức dự án này và phát hành nó cho mã nguồn mở.

**Tổng quan**

Kermit hiện đang là giao thức không đồng bộ được dùng nhiều nhất hiện này.

Giao thức truyền file này tương tự như hoạt động của XMODEM. Máy phát chờ NAK trước khi bắt đầu truyền. Kermit cho phép truyền các ký tự kiểm tra dạng text theo hai bước:

+ Đầu tiên, ký tự kiểm tra được dùng dạng text, được chuyển thành các ký tự in được thông qua việc thêm vào một số cố định và mã ASCII được dùng biểu diễn.

+ Bước hai, thêm ký tự # vào phía trước ký tự vừa chuyển đổi. Theo cách này, ký tự kiểm tra dùng như text được gửi đi như hai ký tự. Khi máy thu gặp ký tự # thì biết phải bỏ đi và ký tự kế chính là ký tự kiểm tra. Nếu máy phát muốn phát ký tự #, thì cần phải gửi đi hai ký tự này.

Các triển khai của Kermit được hỗ trợ bao gồm C-Kermit (cho Unix và OpenVMS) và Kermit-95 (Cho các phiên bản Windows từ Windows 95 trở đi và OS/2).

**CHƯƠNG 3. CÁC GIAO THỨC LIÊN KẾT ĐỒNG BỘ**

**3.1 HDLC**

**Giới thiệu**

Là giao thức truyền thông mục đich chung điểm-điểm hoạt động ở cấp liên kết dữ liệu. Nó dựa trên ISO 3309 và ISO 4335.

Nó nổi lên như một sự tiến hóa của SDLC trước đó. Nó cung cấp khả năng phục hồi lỗi trong trường hợp mất gói dữ liệu, lỗi chuối và các lỗi khác, do đó nó cung cấp liên lạc đáng tin cậy giữa máy phát và máy thu.

Đặc điểm cơ bản

Đặc điểm:

+ Hoạt động ở chế độ full-duplex.

+ Liên kết point-to-point hoặc multipoint.

+ Truyền dẫn đồng bộ.

+ Điều khiển lỗi.

+ Dùng cho các liên kết với giá trị lớn và nhỏ của a.

Ưu điểm:

+ Không phụ thuộc mã điều khiển.

+ Khả năng thích ứng.

+ Hiệu quả cao.

+ Độ tin cậy cao.

HDLC định nghĩa ba loại trạm, ba cấu hình liên kết và ba chế động hoạt động để truyền dữ liệu.

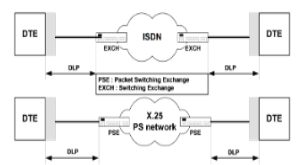
Ba loại trạm của HDLC là : sơ cấp, thứ cấp và kết hợp.

Các cấu hình mạng dùng HDLC:



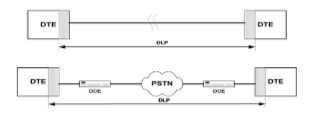
Hình 19

Điểm nối điểm với một sơ cấp và một thứ cấp



Hình 20

Đa điểm với một sơ cấp và nhiều thứ cấp



Hình 21

Điểm nối điểm với hai sơ cấp và hai thứ cấp

Các loại trạm làm việc:

* Trạm sơ cấp:
  + Điều khiển hoạt dộng của liên kết.
  + Phát ra cá khung lệnh.
  + Duy trì liên kết luận lý riêng cho các trạm phụ.
* Trạm thứ cấp:
  + Hoạt động dưới sự điều khiển của trạm chính.
  + Phát ra các khung đáp ứng.
  + Trạm sơ cấp duy trì các liên kết lý luận riêng cho trạm thứ cấp.
* Trạm tổ hợp:
  + Có thể phát ra các khung lệnh và khung đáp ứng.
  + Kết hợp đặc điểm của cả trạm sơ cấp và trạm thứ cấp.

Cấu hình liên kết:

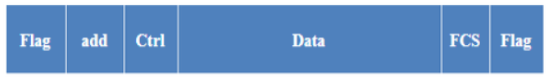
* Không cân bằng:
  + Một trạm chính và một hoặc nhiều trạm phụ
  + Cho phép full duplex và half duplex
* Cần bằng
  + 2 trạm tổ hợp
  + Hỗ trợ full duplex và half duplex.

Chế độ truyền:

* Normal Response Mode (NRM):
  + Cấu hình không cần bằng
  + Trạm chính khởi động việc truyền đến trạm phụ.
  + Trạm phụ chỉ có thể truyền dữ liệu để đáp ứng lại lệnh từ trạm chính.
  + Dùng dường truyền dạng multi-drop.
  + Thường máy chủ sẽ là trạm chính và các terminal là trạm phụ.
* Asynchronous Balanced Mode (ABM)
  + Cấu hình cân bằng
  + Tạm nào cũng có thể bắt đầu truyền mà không cần sự cho phép.
  + Được dùng rộng rãi.
  + Không tốn chí phí cho việc polling.
* Asynchronous Response Mode (ARM)
  + Cấu hình không cân bằng
  + Trạm phụ có thể bắt đầu truyền mà không cần sự cho phép của trạm chính.
  + Trạm chinh chịu trách nhiệm cho đường truyền.
  + Ít dùng.

**Cấu trúc khuôn dạng của Frame**

Cấu túc frame của HDLC có dạng tổng quát:



Hình 22

Trong đó:

* Flag: Mã đóng khung cho frame được chọn là 01111110
* Address: Ghi địa chỉ trạm đích của frame.
* Control: Định danh các loại frame khác nhau.
* Data: Ghi thồn tin cần chuyển đi.
* FSC: Mã kiểm tra lỗi.

Cờ điều khiển:

* Dùng để phân cách khung cả ở 2 đầu.
* Có thể dùng để kết thúc khung này và bắt đầu khung khác.
* Bên thu quét tìm cờ để đồng bộ.
* Ký thuật chèn thêm bit (bit suffing) được dùng để tránh lẫn lộn dữ liệu (chứa 01111110) và cờ.
  + 0 được chèn thêm vào mỗi khi chuỗi 5 số 1 liên tiếp xuất hiện.
  + Nếu bộ thu phát hienj 5 số 1, nó kiểm tra bit kế tiếp
    - Nếu bit đó là 0, nó xóa bit 0 đó.
    - Nếu bit đó là 1 và bit thứ 7 là 0, nó biết đây là cờ.
    - Nếu bit thứ 6 và 7 đều là 1, bộ phát ra lệnh hủy bỏ.



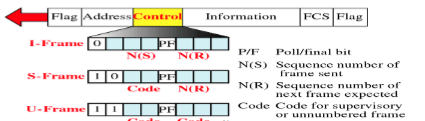
Hình 23

Trường địa chỉ:

* Dùng để nhậ diện trạm phụ đã gửi hoặc sẽ nhận khung.
* Thường dài 8 bit.
* Có thể mở rộng thành bộ số của 7 bit
  + LSB của mỗi octe chỉ thị đây là octe cuối cùng (1) hay chưa (0).
* Địa chỉ toàn 1 (11111111) là địa chỉ broadcast).

Trường điều khiển:

* Khác nhau tùy thuộc vào loại khung:
  + Thông tin – dữ liệu cần truyền đến người dùng
  + Giám sát – dùng ARQ khi piggyback không được dùng.
  + Không số - hỗ trợ cho việc điều khiển liên kết.
* 1 hoặc 2 bit đầu tiên của tường điều khiển dùng để nhận dạng loại khung.
* Bit Poll/Final
  + Dùng tùy theo ngữ cảnh.
  + Khung lệnh
    - Bit P
    - 1 để mời gọi đáp ứng của các trạm ngang hàng.
  + Khung đáp ứng
    - Bit F
    - 1 để chỉ thị đáp ứng đối với lênh mời gọi



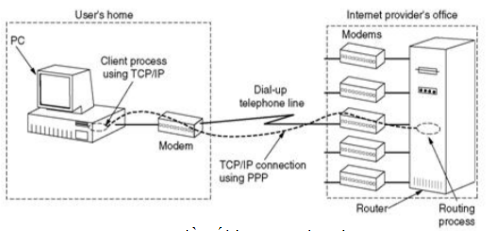
Hình 24

HDLC có 3 loại frame chính sau:

* Loaị U: dùng để thiết lập liên kết dữ liệu theo các phương thức khác nhau với giải phóng liên kết.
* Loại I: dùng chứa thông tin cần truyền đi và được đánh số để kiểm soát.
* Loại SL dùng để kiểm soát lỗi và kiểm soát luông dữ liệu.

**3.2 PPP**

**PPP** là giao thức được sử dụng chủ yếu khi người dùng truy cập internet từ nhà thông qua đường dây ddienj thoai quay số. PPP là giao thức cho 2 thiết bị ở hai đầu của một đường truyền kiểu point-to-point, trao đổi các frame chứa gọi dữ liệu của tầng mạng.

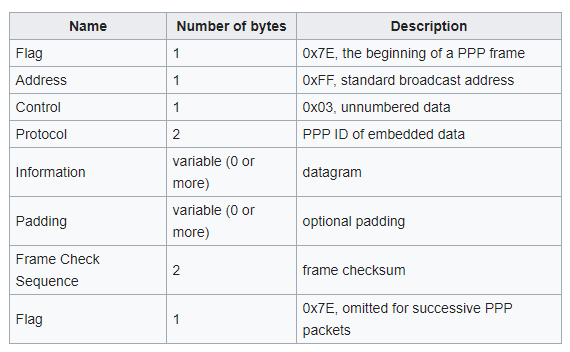


Hình 25

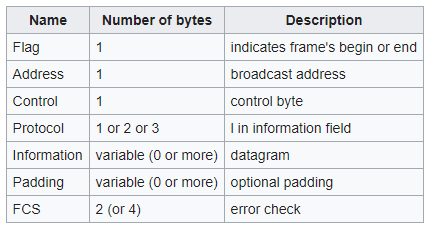
Sơ đồ kết nối của giao thức PPP

**Cấu trúc:**

Khung PPP là một biến thể của khung HDLC:

 **Đóng gói:**

Các frame của PPP được đóng gói trong mộ giao thức lớp thấp hơn cung cấp khung và có thể cung cấp các chức năng khác như tổng kiểm tra để phát hiện lỗi truyền. PPP trên các liên kết nói tiếp thường được gói gọn trong một khung tương tự như HDLC



**KÊT LUẬN**

Các giao thức liên kết dữ liệu cung cấp các phương thức để trao đổi dữ liệu qua tầng liên kết vật lý. Việc xuất hiện và cải tiến các giao thức này giúp việc trao đổi thông tin giữa các máy tính, điện thoại,..vv được nhanh hơn và chính xác hơn, giúp cải thiện đời sống của con người.

# TÀI LIỆU THAM KHẢO

[1] [Stevens 1994](https://en.wikipedia.org/wiki/Point-to-Point_Protocol#CITEREFStevens_1994), p. 26-27, sec 2.6: "PPP: Point-to-Point Protocol"

[2] ["Point-to-Point (PPP) Protocol Field Assignments*"*](https://www.iana.org/assignments/ppp-numbers/ppp-numbers.xhtml). IANA*. Retrieved 3 September 2015*.

[3] Friend, George E.; John L. Fike; H. Charles Baker; John C. Bellamy (1988). Understanding Data Communications (2nd ed.). Indianapolis: Howard W. Sams & Company. [ISBN](https://en.wikipedia.org/wiki/International_Standard_Book_Number) [0-672-27270-9](https://en.wikipedia.org/wiki/Special:BookSources/0-672-27270-9).

[4] Stallings, William (2004). [*Data and Computer Communications*](https://archive.org/details/datacomputercomm00stal_1) (7th ed.). Upper Saddle River: Pearson/Prentice Hall. [ISBN](https://en.wikipedia.org/wiki/International_Standard_Book_Number) [978-0-13-100681-2](https://en.wikipedia.org/wiki/Special:BookSources/978-0-13-100681-2).

[5] S. Tanenbaum, Andrew (2005). [*Computer Networks*](https://archive.org/details/isbn_9788177581652) (4th ed.). 482,F.I.E., Patparganj, Delhi 110 092: Dorling Kindersley(India)Pvt. Ltd.,licenses of Pearson Education in South Asia. [ISBN](https://en.wikipedia.org/wiki/International_Standard_Book_Number) [81-7758-165-1](https://en.wikipedia.org/wiki/Special:BookSources/81-7758-165-1).

[6] [Telecommunications: XMODEM: A Standard Is Born](https://books.google.com.au/books?id=9eJxx_ZGKngC&lpg=PA451&dq=%22Ward%20Christensen%22&pg=PA451#v=onepage&q=%22Ward%20Christensen%22&f=false), By Alfred Glossbrenner, PC Mag, 17 April 1984, Page 451-452, *... but the protocol itself was long ago placed in the public domain by its creator, Chicagoan Ward Christensen. Since its introduction in 1978, XMODEM ...*

[7] [In Focus: History lesson: Ward Christensen's free free-exchange software](https://books.google.com.au/books?id=EzAEAAAAMBAJ&lpg=PA26&dq=%22Ward%20Christensen%22&pg=PA26#v=onepage&q=%22Ward%20Christensen%22&f=false), By Michael Swaine, InfoWorld, 1 Nov 1982, Page 26

[8] Ward Christensen, ["Memories"](http://www.bbsdocumentary.com/software/AAA/AAA/CBBS/memories.txt), 25 November 1992

**[9]** [*"The Virtual Community"*](http://www.well.com/user/hlr/vcbook/vcbook4.html).

[10] Kline, David (July 1982). [*"Osborne—Behind Guerrilla Lines"*](https://archive.org/stream/kilobaudmagazine-1982-07/Microcomputing_1982_July#page/n43/mode/2up). Microcomputing. pp. 42–50*. Retrieved 15 February 2016*.

[11] Pournelle, Jerry (July 1983). [*"Interstellar Drives, Osborne Accessories, DEDICATE/32, and Death Valley"*](https://archive.org/stream/byte-magazine-1983-07-rescan/1983_07_BYTE_08-07_Videotex#page/n325/mode/2up). BYTE. p. 323*. Retrieved 28 August 2016*.

[12] [*Christensen, Ward*](https://en.wikipedia.org/wiki/Ward_Christensen) (1 January 1982). [*"XMODEM Protocol Overview"*](http://techheap.packetizer.com/communication/modems/xmodem.html).

[13] [*Forsberg, Chuck*](https://en.wikipedia.org/wiki/Chuck_Forsberg) (11 September 1986). [*"XMODEM/YMODEM PROTOCOL REFERENCE"*](http://techheap.packetizer.com/communication/modems/xmodem-ymodem_reference.html).