データリンクの基礎技術

1999.4.23

概要

- データリンクの提供している機能は何か
- 使用されている技術
  - » デザイン・コンセプト
  - » 誤り処理 (error control)
  - » フロー制御 (flow control)
  - » 基本的なプロトコル

データリンクとは

- 機能
  - » 同一の伝送メディア上のホスト間の通信
    - 同一の物理媒体を共有しているホスト間通信
    - 異なる伝送メディアとの通信にはゲートウェイが必要
      - – Layer 1 gateway: repeater
      - – Layer 2 gateway: bridge
      - – Layer 3 gateway: router
  - » 物理層サービスを利用しての通信機能の提供
    - パケット交換を実現
    - データリンク・フレーム (datalink frame)
  - » ネットワーク層に対するインタフェース

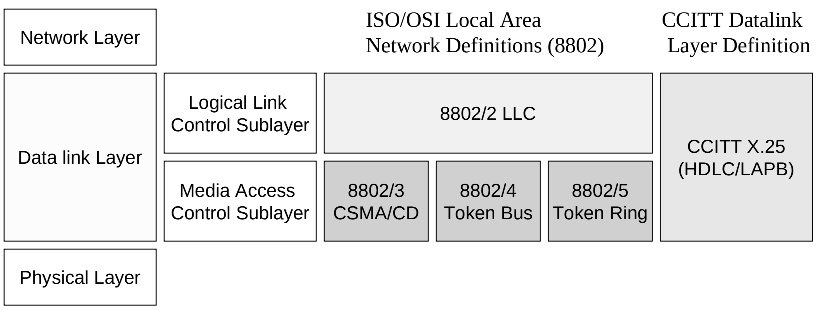
歴史

- 1970年代
  - » 公衆データ回線用のデータ転送プロトコルの普及
    - X25(HDLC)
- 1980年代
  - » LAN技術の普及
    - Ethernet
    - Token Ring
    - FDDI
  - » 無線技術の登場
    - Wireless Ethernet
- 1990年代
  - » LANトラヒックの公衆データ回線上での収容
    - Frame Relay
    - ATM
  - » 無線技術の広がり
    - Wireless Ethernet
    - データ通信
  - » LAN技術の広帯域化
    - Fast Ethernet
    - Gigabit Ethernet

2つの王国 (1)

- Point-to-Point connection
  - » CCITT X.25 (HDLC/LAPB)
  - » ディジタル専用回線やISDNで利用
- Broadcast Channel (LAN)
  - » IEEE 802 (ISO 8802)
    - LLC (Logical Link Control)
    - MAC (Media Access Control)
  - » Ethernet, FDDI, Token Ringなどが全て IEEE802 グループとして標準化
  - » ISO8802グループとして国際標準化

2つの王国 (2)



Copyright ©1999, Suguru Yamaguchi, All right reserved

なぜデータリンク層は必要か

- パケット交換の基本的な機能を提供
  » 物理層が提供するビットレベルでの伝送機能を用いて、パケットレベルでの交換機能を実現
- 物理層の特性を考慮した安定したデータ転送機能を提供
  » 遅延 (propagation delay)
  » 誤り (errors)
- ネットワーク層に対するインタフェース
  » 特に共通化したインタフェースが提供される場合、メディアに対する差違を吸収できる
  » IEEE802.2 と IEEE802.x

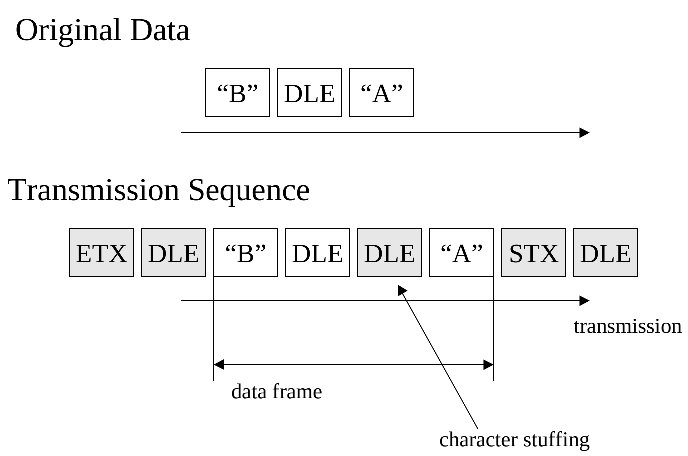
フレーム

- フレームの定義と同期
  » フレーム(frame)
    - データリンク層での処理単位
    - 物理層はビット誤りを発生する可能性がある
      – 適当な長さのブロックに対して冗長な情報を追加して誤りを検出、あるいは、訂正する
      – 誤り制御処理のためのフレーム
    - ビット列中からのフレームの取り出し
      – フレームの同期問題
        » キャラクタ
        » ビット列

フレームの取り出し (1)

- キャラクタベースでのフレームの取り出し
  - » 物理層としては文字を単位としたデータ伝送
    - CCITT X.21
      - – 物理層フレームは8bit
      - – コード体系はASCII(IA5)
  - » データ転送の最初と最後に特別な文字の挿入
    - STX, ETX
  - » DLE (DataLink Escape) によって特別な文字をデータと識別
    - バイナリデータの転送では特別な文字がパターン出現する可能性有り
    - character stuffing

フレームの取り出し (2)



フレームの取り出し (3)

- ビット列をベースとしたフレームの取り出し
  - » 特定のビット列をデータの最初と最後に挿入
    - • 同期
  - » データ中に現れる「同じ」ビット列を細工
    - • bit stuffing
      - – 特定のビット列がフレームの最初と最後だけに常に存在するようにする
  - » 例
    - • 特別なビット列: 01111110
    - • 送信側ではデータ中に5ビット“1”が続いたら“0”を挿入
    - • 受信側ではデータ中に5ビット“1”が続いたら次の“0”を削除

フレームの取り出し (4)

Original Data

01011111111110011

Transmission Sequence

end bit pattern　　　start bit pattern

01111110 0100111110111110011 01111110

→ transmission

bit stuffing

データリンクの基本形態

- Unacknowledged Connectionless Service
  - » 送信者が受信者と同期することと無くフレームを送信
  - » 誤りが発生しても特別な処理をしない
    - 上位層が誤りを制御
  - » 高速、エラーの少ないリンク向けの制御方式
    - LAN, real-time traffic (voice/speech)
- Acknowledged connectionless Service
  - » 受信者は各フレーム毎に受信確認 (acknowledge) を送信
  - » 誤り制御をデータリンク層で実施
  - » 信頼性を向上

データリンクの基本形態

- Connection-oriented Service
  - » データ送信前にコネクションを設定
    - 送信側と受信側が同期した処理が可能
  - » 信頼性（誤り、順番）を保証
  - » 処理は一般に複雑
  - » 低速、エラーの多いリンクに有効

誤り制御

- Error Control
- 目的
  - » 通信路上での伝送誤りを発見・訂正
  - » 送信フレームが正しく送られているかどうか
  - » 送信フレームの順番が正しいかどうか
- 技術
  - » 符号技術
    - • 誤り訂正符号 (Error Correction Code)
    - • 誤り検出符号 (Error Detection Code)
  - » プロトコル技術
    - • タイマ (timer)
    - • 再送 (retransmission)

誤り検出符号

- ビット誤り
  - » パリティ (parity) 法が代表的
  - » ASCIIでは7ビットでコード表現
  - » 8ビット目にパリティ
    - • even parity: コード内での１の数が常に偶数
    - • odd parity: コード内での１の数が常に奇数
  - » 8ビットフレーム内での単一ビット誤りの発見可能
- バースト誤り
  - » 連続したビット誤り
  - » 交換機やネットワークアダプタでの電気的なノイズに起因することが多い
  - » 巡回符号が広く使われる
    - • CRC (Cyclic Redundancy Check code)
    - • 幾つかの標準が存在

CRC

- フレームの後ろに誤り検出用のCRCチェックビット をつける
- ビット列の多項式表現
  - » nビットからなるビット列
    - $(a_{n-1}, \ldots.., a_1, a_0)$ ただし a_i は0または1
  - » 多項式表現
    - $F(x)=a_{n-1}x^{n-1}+\ldots+a_2x^2+a_1x+a_0$
- 多項式演算
  - » modulo 2
  - » 排他的論理和 (eXclusive OR) と等価

CRC

- CRCの計算手順
  - » 生成多項式 $G(x)$ ただし次数は r $(r < m)$
  - » mビットのフレームを送信するとし、その多項式表現を $M(x)$ (次数 m-1) とする
  - » フレームに r 個の “0” をつけたもの $x^rM(x)$ を作成
  - » 多項式 $x^rM(x)$ を $G(x)$ で割った余り $R(x)$ を計算
  - » 送信する符号語 $F(x)$を作る
    - $F(x) = x^rM(x) + R(x)$

CRC

- 誤りの検出
  - » $F(x)$は常に$G(x)$で割り切れる
  - » 割り切れなかった場合には誤りが存在
  - » rビット以下のバースト誤りは検出可能
  - » それ以外の誤りは$G(x)$に依存

標準化された**CRC**

- 3つの標準
  - » CRC-12
    - $x^{12}+x^{11}+x^3+x^2+x+1$
  - » CRC-16
    - $x^{16}+x^{15}+x^2+1$
  - » CRC-CCITT
    - $x^{16}+x^{12}+x^5+1$

標準化された**CRC**

- 用途
  - » CRC-12は6ビット文字表現に対して
  - » CRC-16およびCRC-CCITTは8ビット文字表現に対し有効
- 検出特性 (CRC-16およびCRC-CCITT)
  - » 全ての単一および2ビット連続ビット誤り
  - » 全ての奇数個のビット誤り
  - » 16ビット以下のバースト誤り
  - » 99.997%の17ビットバースト誤り
  - » 99.998%の18ビット以上のバースト誤り

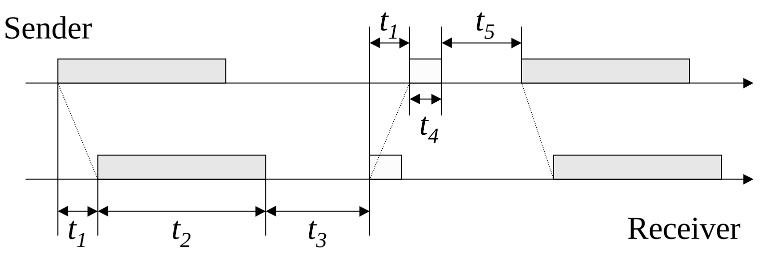
誤り訂正符号

- Error Correction Code
  - » ハミング符号 (hamming code) の利用
  - » 符号語間の距離
    - d個の誤り検出: d+1以上
    - d個の誤り訂正: 2d+1以上

基本的なプロトコル

- 誤り制御のため
  - » 受信側でフレームに誤りを発見
  - » 送信者側にもう一度フレームを送り直してもらう
    - 再送 retransmission
  - » 実行方法にはいろいろある
- 基本的なプロトコル
  - » Stop-and-wait
  - » Go-back-N
  - » Selective-repeat

Stop-and-wait Protocol



t_1: 伝送遅延
t_2: フレーム伝送時間
t_3: フレーム処理時間
t_4: ACK伝送時間
t_5: ACK処理時間

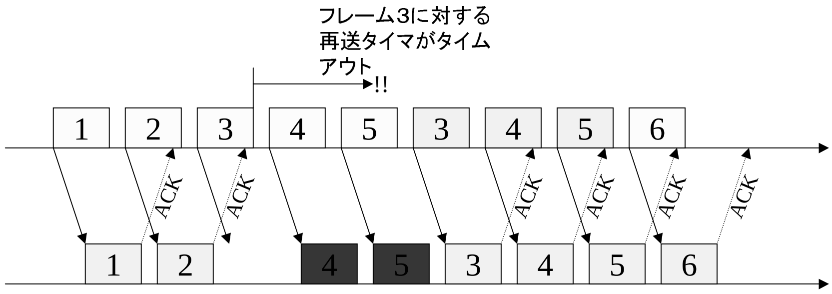
Stop-and-wait Protocol

- 手法
  - » 1フレーム送信毎にACK受信を待つ
  - » 送信側タイマは $2t_1+t_2+t_3+t_4$以上に設定
  - » 送信側タイマがタイムアウトするとフレームを再送
- 特徴
  - » 単純
  - » バッファは送受信側それぞれで1フレーム分でよい
  - » 回線使用効率が非常に悪い

Go-back-N Protocol

- 手法
  - » Stop-and-Wait Protocolを改善
  - » 連続してフレームを送出
  - » 誤りが発生した場合
    - 誤りが発生したフレームを正しく受信するまで受信したフレームを破棄
    - 正しい受信までACKも返さない

Go-back-N Protocol



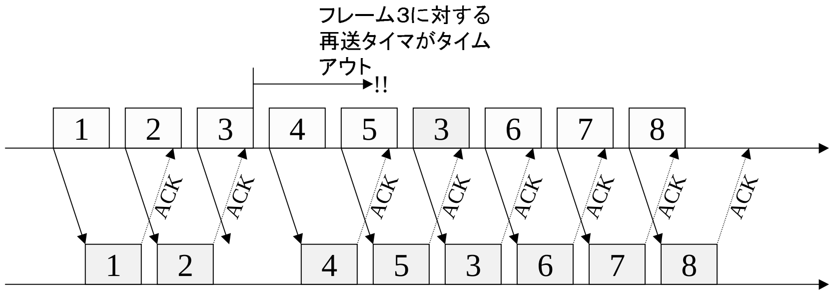
Go-back-N Protocol

- 特徴
  - » 誤りが発生すると正しく送られたフレームでも破棄され、無駄が大きい
    - 特に再送タイマを大きく設定する場合不利
  - » 送信側で再送のためのバッファが必要
    - ACKを受け取っていないフレームについては再送のためにバッファリングしておく必要あり
    - Go-back-N の “N” はACKを待たずに送ることができるフレーム数 (on-the-fly frame数)
    - バッファは再送タイマの値に依存

Selective-repeat Protocol

- 方式
  - » Go-back-N プロトコルを改良
  - » 必要なフレームだけを再送
- 特徴
  - » 明らかに伝送効率は Go-back-N よりも良い
    - 正しく受信されているフレームを無駄に破棄しない
  - » 処理が複雑
    - 受信側でバッファが必要
    - 順番制御（フレームシーケンス制御）が必要

Selective-Repeat Protocol



フロー制御

- Stop-and-Wait, Go-back-N では受信側では複雑な処理が発生しない
- Selective-Repeat では、受信側で順番制御
  - » 受信側にバッファが存在
  - » 送信側がうまく送らないと受信側バッファで溢れが発生する可能性有り
- フロー制御 (flow control)
  - » 受信側との間で何らかの手法を用いて、バッファ溢れを抑制する

Window Flow Control

- ウィンドウ・フロー制御
  - » ウィンドウサイズをWとする
    - 受信側のバッファサイズと同じと考えて良い
  - » 送信側で (1) 既に送信したフレームで (2) ACKを受け取っていないフレームの数（あるいはバイト数）を Nr とする
  - » Nr < W の間は送信を行う
  - » sliding window flow control

まとめ

- データリンクの基本的な技術を紹介
  - » 誤り処理 (error control)
  - » フロー制御 (flow control)
  - » 基本的なプロトコル
- 実は他の階層でも使われる技術