Grid勉強会

11/06/21

TCP

高瀬　亘

使用書籍

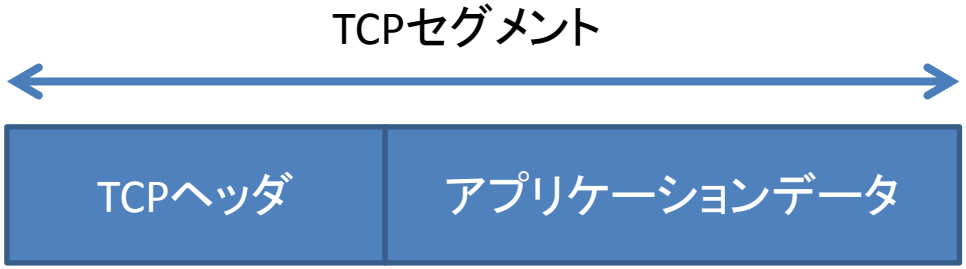
- マスタリングTCP/IP 応用編
  - オーム社
  - Philip Miller 著
  - 苅田幸雄 監訳

TCPの機能と役割

- TCP : Transmission Control Protocol
- コネクション型のエンドシステム間の高信頼性プロトコル
- IPだけでは信頼性にかけるので、TCPによって信頼性の高い通信システムを提供
- 送られたどのデータについても一定時間内に応答確認を行うよう命令する
  - この応答確認が届かない場合、TCPではデータを再送するため、IPなどの下層プロトコルを補い信頼性を保っている
- 複数のアプリケーションをサポートするように設計されている
  - Telnet, FTP等

TCPの動作：基本的なデータ転送

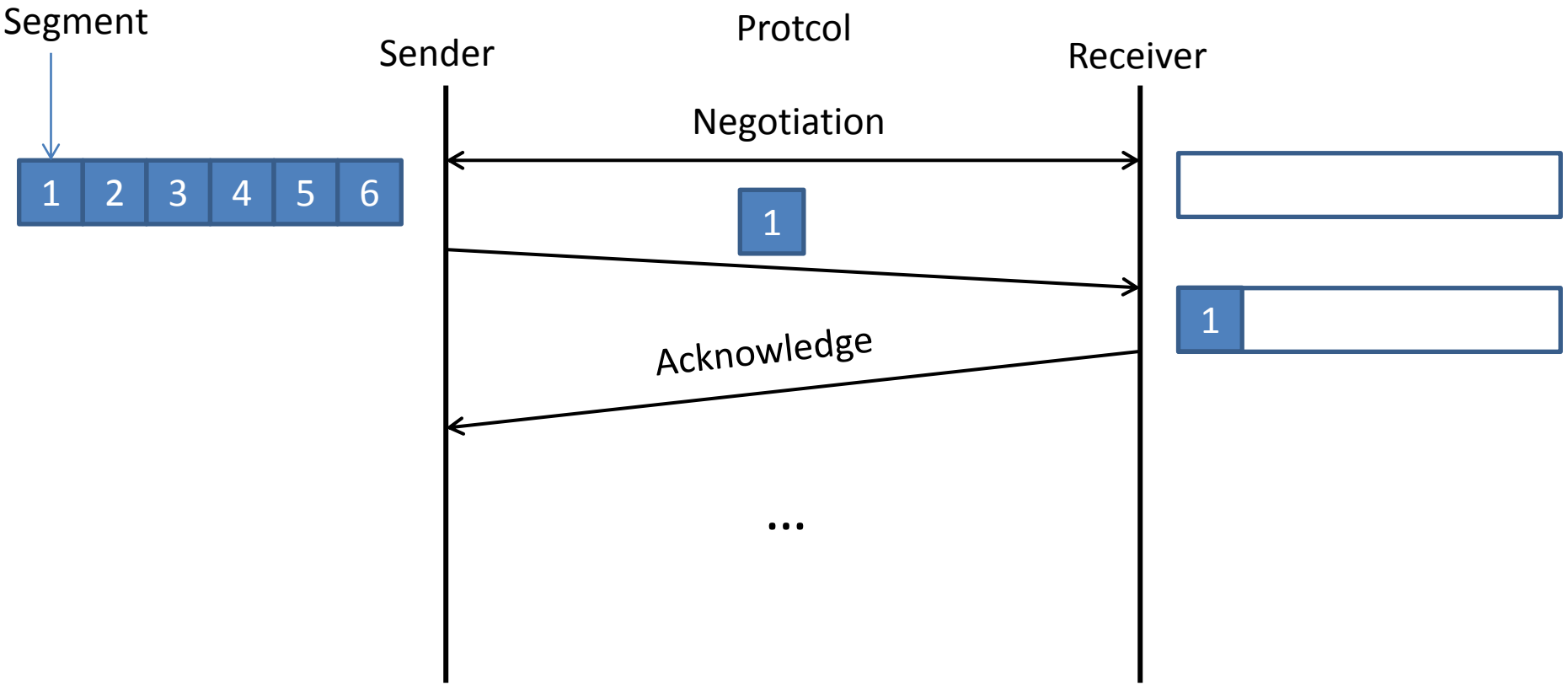
- データストリームの発信と受信を同時に行うことができる
  - TCP層の通信がすべて全二重の性質を持つわけではない
- セグメント
  - TCP層から転送されるデータのかたまり
  - セグメントのサイズと転送のタイミングはTCPモジュールが決める



TCPの動作：信頼性

- どのセグメントに対しても応答確認が必要
- セグメントごとにシーケンス番号が付けられ、遅延や喪失があった場合に再送を行う
  - 送信側は、一定時間内に応答確認がなかった場合再送する
  - 受信側でセグメントが重複した場合は、シーケンス番号にもとづいてセグメントを破棄
  - 送信側で応答確認が重複した場合は、応答確認番号にもとづいて応答確認を破棄

TCPの動作：信頼性

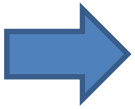

Segment
1
2
3
4
5
6
Sender
Protcol
Receiver
Negotiation
1
1
Acknowledge
...

TCPの動作：信頼性

1. 送信側は、シーケンス番号に添えてデータを送る
   - 送信後、セグメントのオクテット数だけシーケンス番号を増やしておく
2. 受信側は、シーケンス番号+受信したオクテット数を応答確認番号として返す
3. 送信側は、応答確認番号が次に送信するシーケンス番号になっている事でデータが無事送られた事を確認
4. 1に戻る

TCPの動作：信頼性

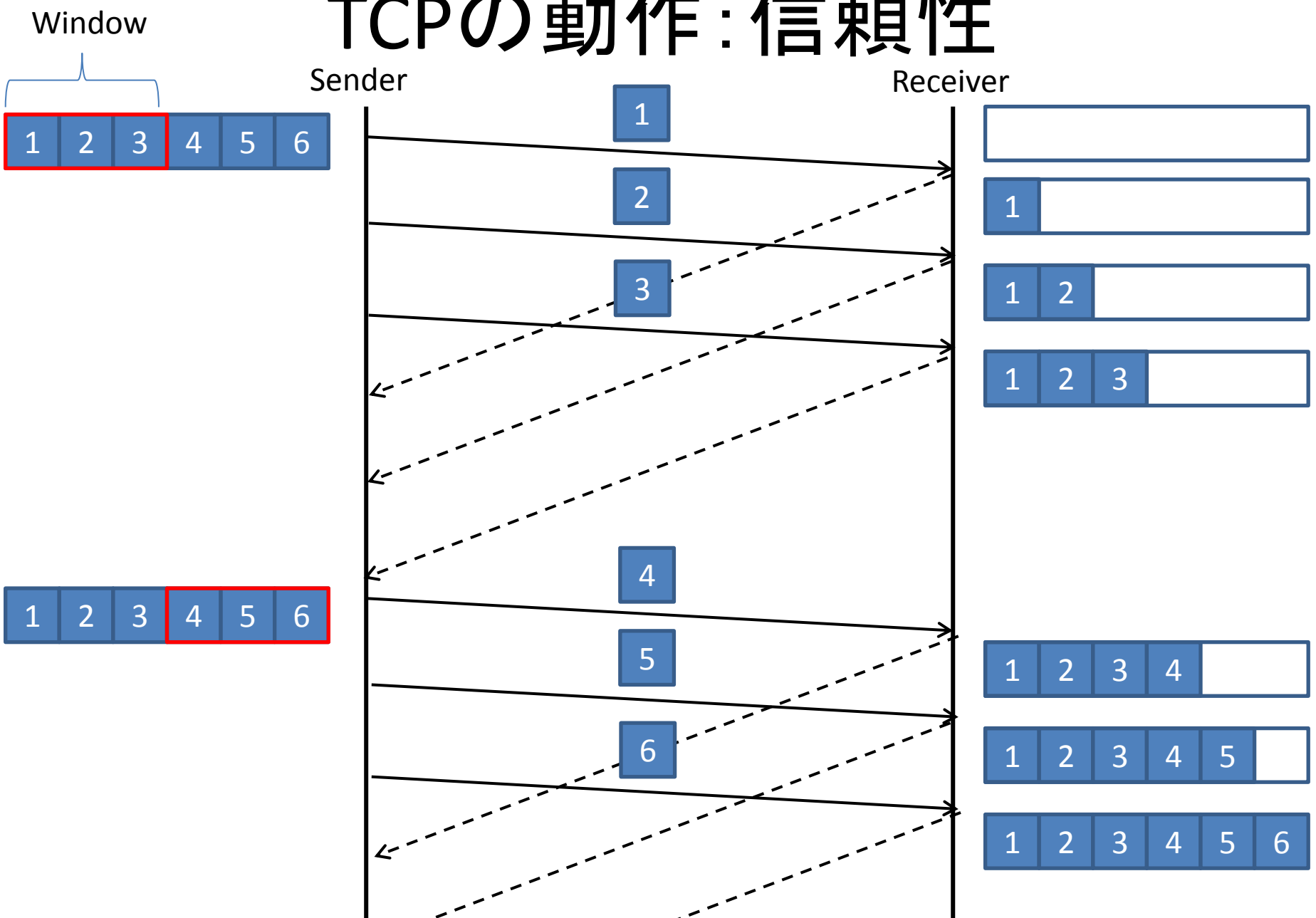
- しかしこの方法では、利用可能なネットワーク帯域幅とリソースを有効に活用していない
  - セグメントごとに応答確認を待たないといけない
  - WANでは大幅な遅延もありうる



もっと効率の良い方法を採用

- ウィンドウサイズごとに応答確認を行う
- 一度に複数のセグメントを送信できる

TCPの動作：信頼性


Window
1 2 3 4 5 6
Sender
Receiver
1
2
3
4
5
6
1 2 3 4 5 6
1
1 2
1 2 3
1 2 3 4
1 2 3 4 5
1 2 3 4 5 6

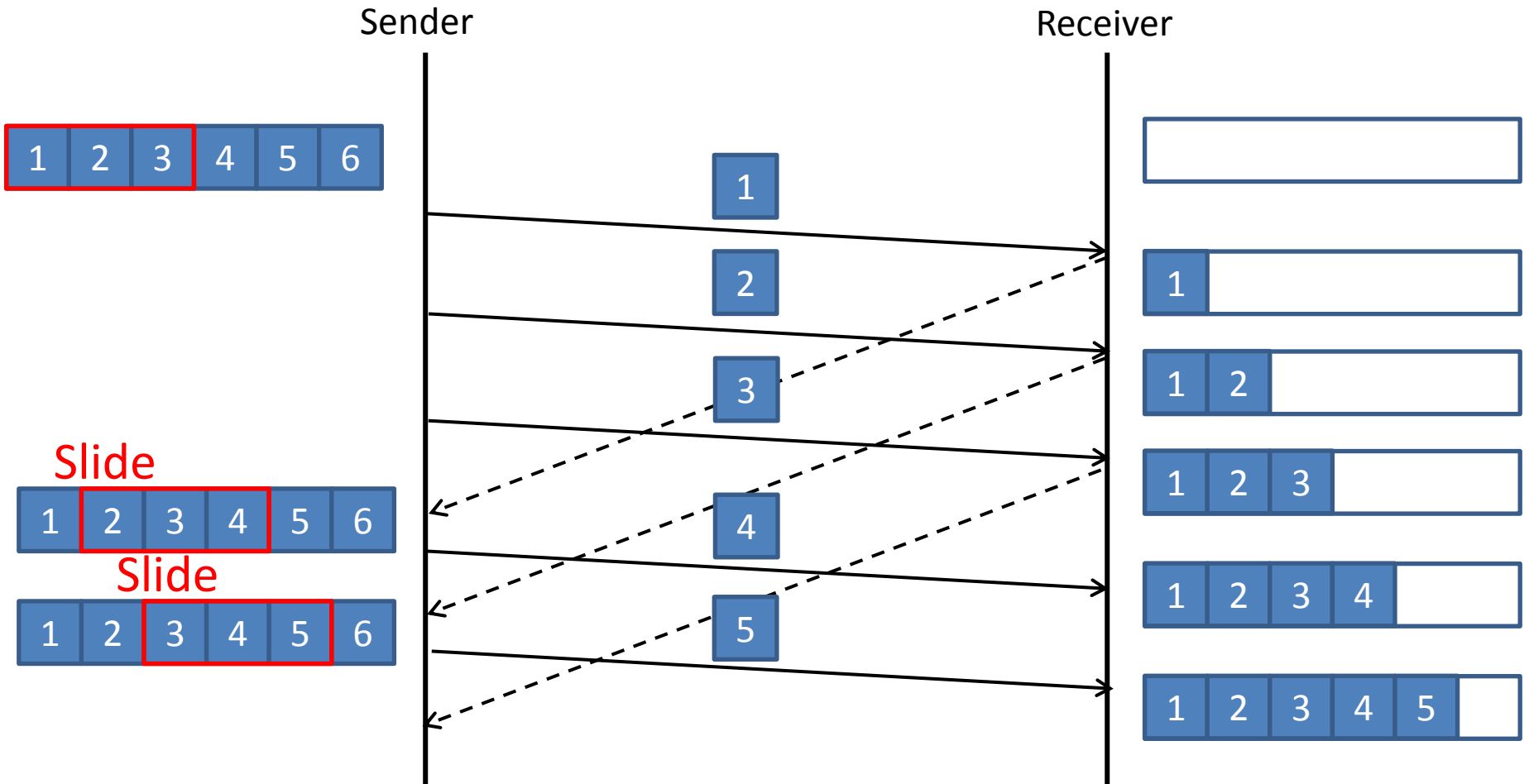
TCPの動作：信頼性

さらに効率の良い方法を採用

- スライディングウィンドウ
  - 1つ応答確認があったら、ウィンドウをずらして次のセグメントを送る
  - 常に応答確認のすんでないセグメントがある

TCPの動作：信頼性

- 応答確認があるごとにウィンドウをスライド



TCPの動作：信頼性

- さらに効率を良くするために
- いくつかのセグメントに対してまとめて応答確認する

- まとめて送る
- まとめて応答する

フロー制御

- 送信側から送る事のできるデータ量は受信側が決める
- 受信側が送るどの応答確認にもウィンドウが設けられる
  - これが受信側で受け入れ準備ができているオクテット数を示す
  - 送るべきデータの量が送信側に分かる
  - 受信側はウィンドウを調整することで自分のバッファの状態を知らせることができる
    - 受信データは一旦バッファに蓄えられ、その後本来の送り先へ移されるため、本来の送り先への移動に時間がかかるとその分バッファの空きが少なくなる

フロー制御

- 送出バッファに置かれたデータは応答確認がくるまで削除できない
  - 再送の可能性があるため
- 受信データも到着後すぐに削除できない
  - 受信処理に時間がかかる可能性があるため

フロー制御

- 無能ウィンドウ症候群
  - 受信側に小さいバッファスペースが見つけては、それを目がけて小さなデータを送りネットワークリソースを不効率に使うこと

- 上記症候群を解決するための公式

RCV.BUFF – RCV.USER – RCV.WND >= min (Fr * RCV.BUFF, EFF.snd.MSS)

RCV.BUFF：バッファスペース総量
RCV.USER：受信バッファ上の未削除データ
RCV.WND：ウィンドウ機能を使用して広告されるバッファースペース量
Fr：0.5が推奨
Eff.snd.MSS：コネクション確立時に決めた有効最大セグメントサイズ

フロー制御

- 前ページの公式を使用して・・・
  - 受信側
    - 利用可能なバッファスペースを通知するとき、適切なサイズのウィンドウが利用できるようになるまで、ウィンドウを縮小した値を伝える
  - 送信側
    - 応答確認待ちのデータがある場合に、小さいセグメントの送信数を制限（Nagleアルゴリズム）
      - 小さいセグメントはまとめて送信する
    - TCPモジュールがプッシュされたものも含めてユーザデータを全てバッファして、応答確認を待つ

※プッシュ：速やかにデータをアプリケーションに渡して欲しいという要求

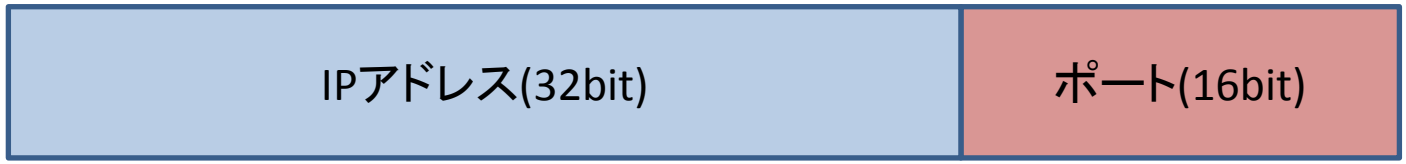
多重化

- TCPは、1つのホスト内の多数のプロセスに、同時にコネクション型環境を提供
- ポートとIPアドレスの組み合わせによりソケットを生成
  - ポートでアプリケーションを指定

    HTTP : 80

    SSH : 22

    FTP : 21

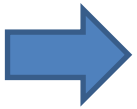


コネクション

- コネクションオープンの方法
  - パッシブオープン
    - 送られてくるサービス要求をプロセスが受け入れる
    - 来たら受け入れる
    - TelnetやFTPのサーバプロセス
  - アクティブオープン
    - 他のホストとのコネクションをプロセスが積極的に開始
    - こちらからいく
    - TelnetやFTPのクライアントプロセス

コネクション

- コネクションを確立できるのは、両端がともにリモートプロセスのポート識別子を識別している時
  - ウェルノウンポートを別にすればポート識別子は動的に割り当てられる
    - パッシブオープンの場合は、0か未指定の識別子

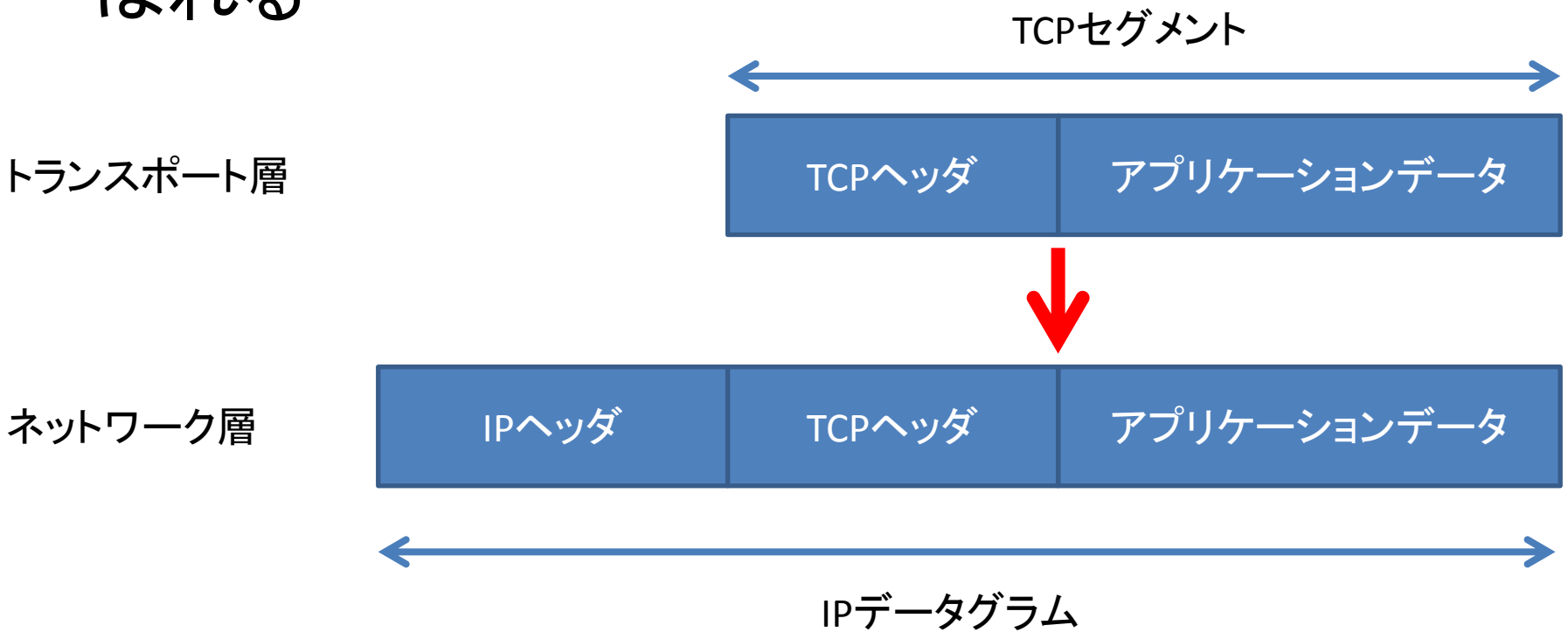


つまり、コネクションが確立できるのはアクティブオープンを受け取った場合に限られる

- しかし、任意のリモートホストとコネクションを確立する自由も、サーバプロセスには残されている

TCPセグメントヘッダ

- TCPセグメントはIPデータグラムデータとして運ばれる



TCPセグメントヘッダ

- サーバは多数の未知のホストにサービスを提供しなければならない
  - 約束事として1024までのポートをウェルノウンポートとして定義してある
- TCPセグメントヘッダは、送信側が受信側に対して様々な情報を伝えるために付加するもの

TCPセグメントヘッダ

- シーケンス番号
  - セグメントが運ぶデータの先頭オクテット（≒バイト）のシーケンス番号
- 応答確認番号
  - 受信側が次に受け取るセグメントの先頭オクテットのシーケンス番号
- データオフセット
  - TCPセグメントヘッダの長さ
- 未使用ビット
  - 0に設定

TCPセグメントヘッダ

- URG
  - 緊急データを表すフラグ
- ACK
  - 応答確認有効を表すフラグ
- PSH
  - プッシュ機能を要求するフラグ
- RST
  - コネクションのリセットフラグ
- SYN
  - コネクション確立時に同期をとるフラグ
- FIN
  - コネクションを閉じるためのフラグ

TCPセグメントヘッダ

- ウィンドウ
  - 受け入れ可能なデータのオクテット数を受信側に知らせる
- チェックサム
  - データがエラーなして受信されたかを確認する
- 緊急ポインタ
  - データストリームのどこに緊急データがあるかを表す
- オプション（最大セグメントヘッダオプション）
  - 受信可能な最大セグメントサイズを表す
- パディング
  - アプリケーションデータとTCPヘッダとの境界の調整役

次回

- 11/06/23 13:30