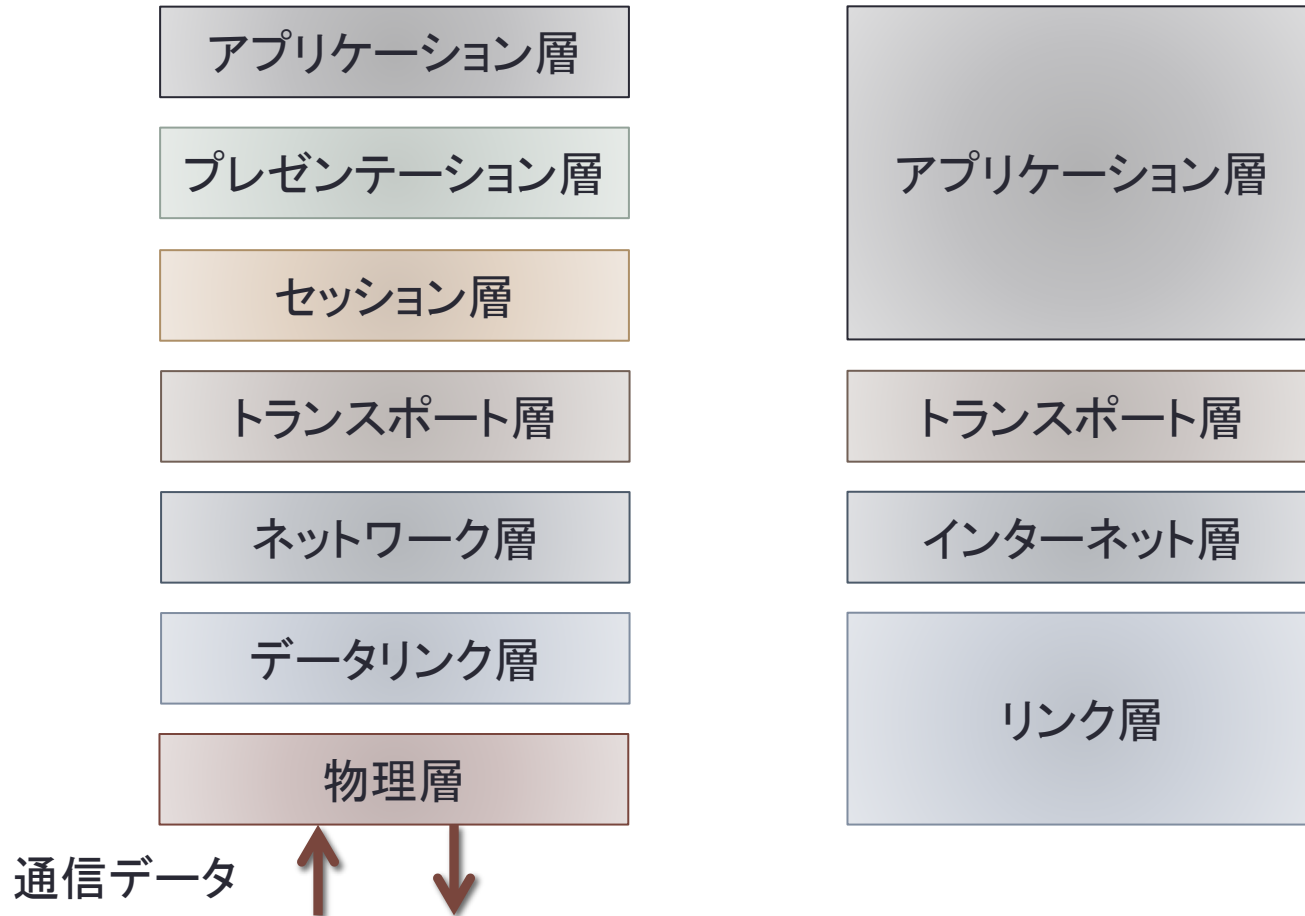


情報通信ネットワーク 第7回

理工学部情報科学科

松澤 智史

本日は……トランスポート層



今日のコンテンツ

- TCP(Transmission Control Protocol)
 - 低速になるが、データの信頼性を提供
 - 到達順の保証
 - データ損失の補填機能
- UDP(User Datagram Protocol)
 - 信頼性はないが、高速な通信を実現

復習

- 物理層

- 電気・光などの媒体を用いて,
0/1の情報をできる限り高速かつ正確に送る

- データリンク層

- マルチリンクにおいて送信元・先を明らかにし、自分宛でないデータは捨てる

- ネットワーク層

- ネットワークをまたがる通信相手への送受信を可能にする
- データ分割・復元を可能にする

TCP(Transmission Control Protocol)

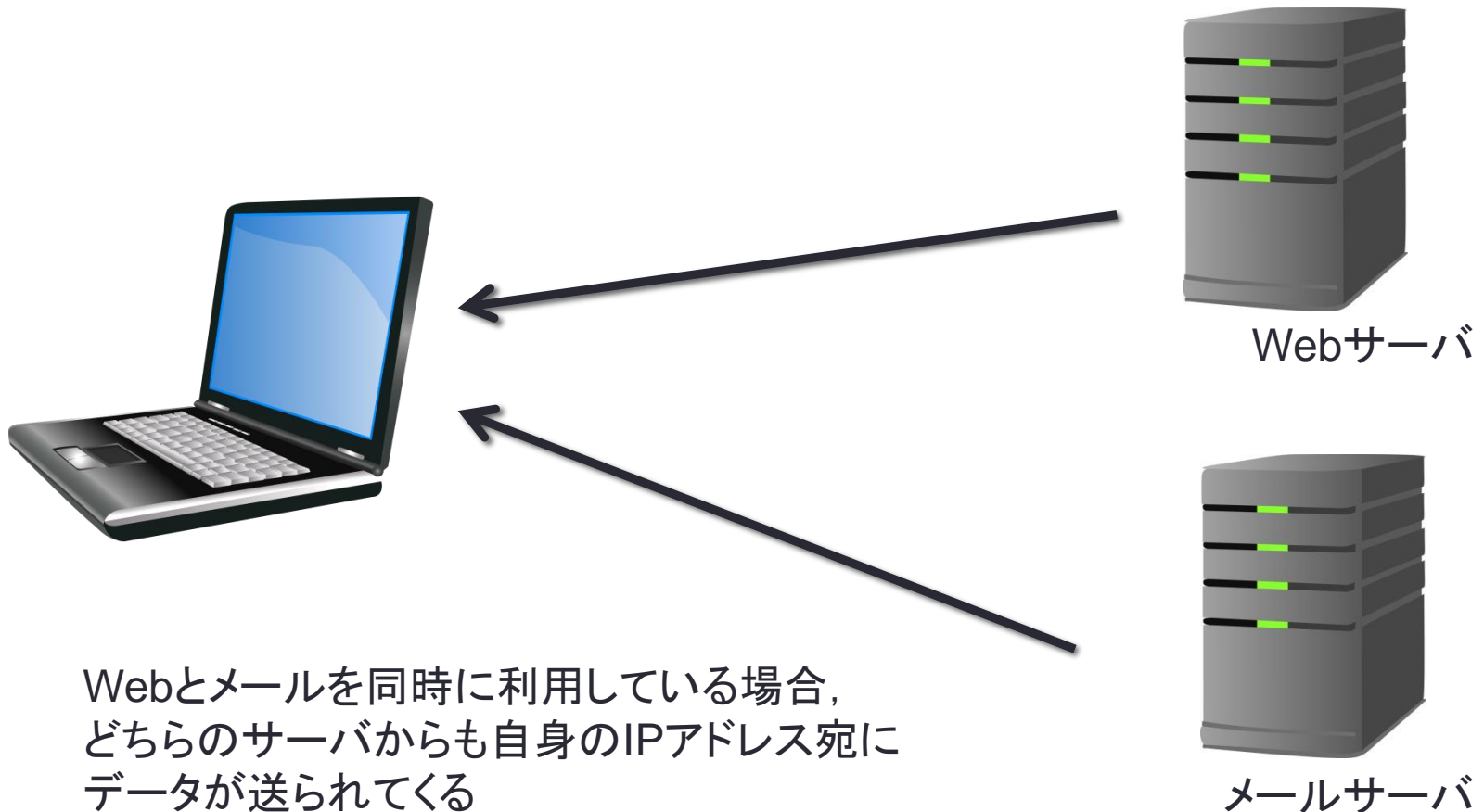
- ポート番号と呼ばれる識別子があり、サービスの区別を行う
- チェックサムで誤りの検知を行い、誤りがある場合は破棄する
 - オプションでチェックサム計算なしの指定も可能であるがTCPではチェックサム必須となっている
- コネクション指向のストリーム通信を行う
 - 仮想通信路(コネクション)を形成して複数のデータをやりとりする
- 信頼性を確保する
 - 個々のデータの受信確認を行い、送信順序を維持した受信を行う
 - データの欠落には再送を行い、欠落がなくなるまで再送を繰り返す
 - 順番を意識した上位プロトコルで形成できる
 - ウィンドウサイズによるフロー制御を行う

TCPヘッダ

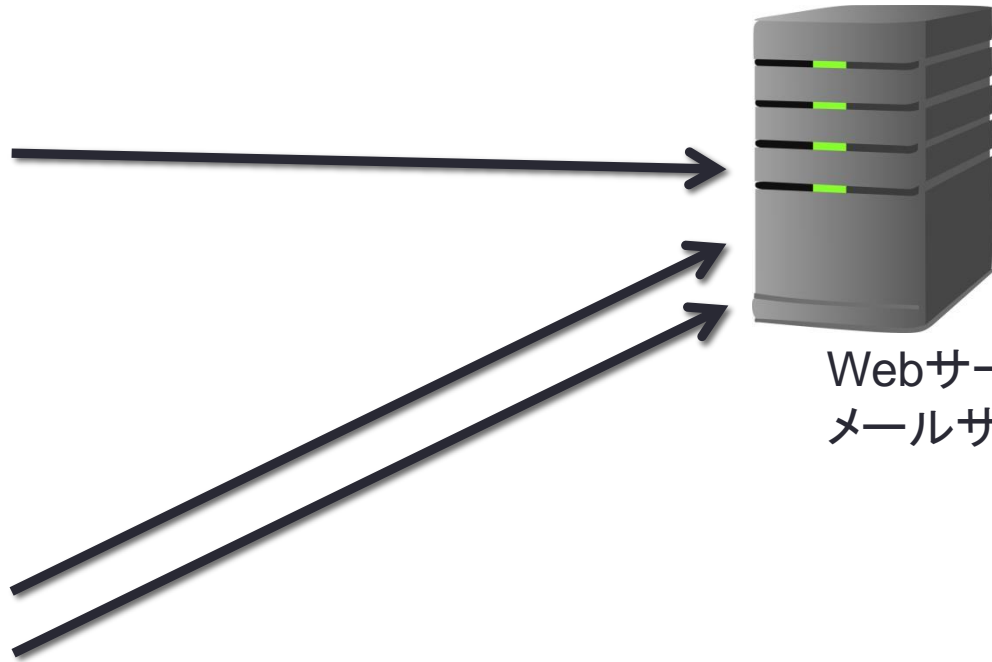
0	4	10	16	31
始点ポート番号 Source Port th_sport			終点ポート番号 Destination Port th_dport	
シーケンス番号 Sequence Number th_seq				
確認応答番号 Acknowledgement Number th_ack				
オフセット Data Offset th_off		フラグ Flags th_flags	ウィンドウサイズ Window th_win	
チェックサム Checksum th_sum			緊急ポインタ Urgent Pointer th_urp	

- シーケンス番号 送信したデータの位置(オクテット単位で表す)
- 確認応答番号 次に受信すべきシーケンス番号
- データオフセット TCPのデータの開始位置(4オクテット単位) 通常5
- フラグ コネクション確立・切断・強制切断・緊急・ACK・PUSH
- ウィンドウサイズ データを受信側の空きバッファ領域の大きさ

ポート番号によるサービス区別



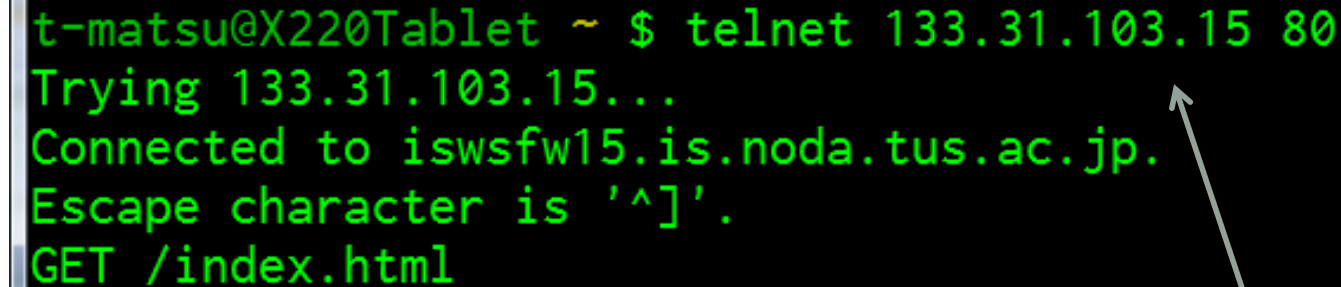
ポート番号によるサービス区別



Webサーバ 兼
メールサーバ

Webとメールのサービスを同一サーバで行っている
場合、クライアントからの問い合わせがどちらの
サービスのものであるかの判別ができない

telnetによるTCPポートアクセス



```
t-matsu@X220Tablet ~ $ telnet 133.31.103.15 80
Trying 133.31.103.15...
Connected to iswsfw15.is.noda.tus.ac.jp.
Escape character is '^]'.
GET /index.html
```

The image shows a terminal window with a black background and green text. The text displays the execution of a telnet command to connect to a specific IP and port, followed by the connection status and the manual input of an HTTP GET request.

HTTPの命令を手動入力してみよう

サーバ名とポート番号
80はWeb(HTTP)

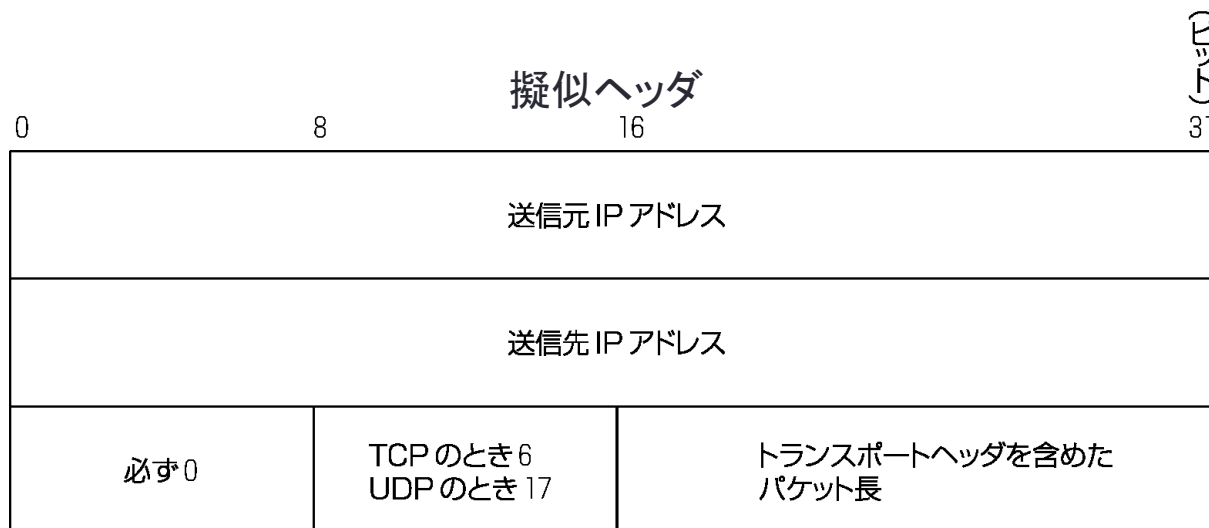
代表的なポート番号

ポート番号	プロトコル	説明
20	FTP(data)	ファイル転送(データ用)
21	FTP	ファイル転送(制御用)
22	SSH	暗号化された遠隔ログインプロトコル
23	TELNET	遠隔ログインプロトコル
25	SMTP	メール転送プロトコル
53	DNS	名前解決用のプロトコル
80	HTTP	WWW
110	POP3	メール受信プロトコル
443	HTTPS	暗号化されたWWW

チェックサム

アルゴリズム

- ・ TCPとUDPの場合は擬似ヘッダを作成する
- ・ チェックサムフィールドに0を入れる
- ・ データ長が奇数の場合は, 16ビット単位になるように調整する
- ・ 擬似ヘッダ, ヘッダ, データ長の部分を16ビット単位での1の補数で加算する
- ・ 求めた値の1の補数をヘッダチェックサムに格納する
- ・ チェックサムが0(すべて0)の場合はチェックサム計算なしとみなす



1の補数

- 補数

- ある基数法において、ある自然数 a に足したとき桁が1つ上がる (桁が1つ増える) 数のうち最も小さい数をいう
- または負の値の表現

- n の補数

- 10進数法において66の10の補数は34 (桁上がりする最小の値)
- 10進数法において66の9の補数は33 (桁上がりしない最大の値)

2進数法における1の補数

- 1の補数の求め方
 - 10010010の1の補数は01101101(つまり単純なビット反転)
 - 00000001(+1)の補数は11111110(-1)
- 1の補数の加算
 - $00000001(+1) + 00000001(+1) = 00000010(+2)$ ※ここまでは良い
 - $11111110(-1) + 00000001(+1) = 11111111(0)$
つまり
 - 0の表現が00000000と11111111の2種類ある
 - $11111111(0) + 00000001(+1) = 00000001(+1)$ となる
つまり
 - オーバーフローしたら下位ビットに1を挿入する
 - 正の数同士, 負の数同士の加算結果が00000000になることはない
 - 加算結果が00000000になるのは00000000+00000000だけ

チェックサム(計算例)

例:001001101001100101001000
わかり易いように8ビットにする

サンプルプログラム

<http://www.is.noda.tus.ac.jp/~t-matsu/3is/checksum8.c>

データ1	: 00100110	1の補数で加算	
データ2	: 10011001	→	合計 : 00001000
データ3	: 01001000		

受信側で検証

データ1	: 00100110
データ2	: 10011001
データ3	: 01001000
Checksum	: 11110111

1の補数表現

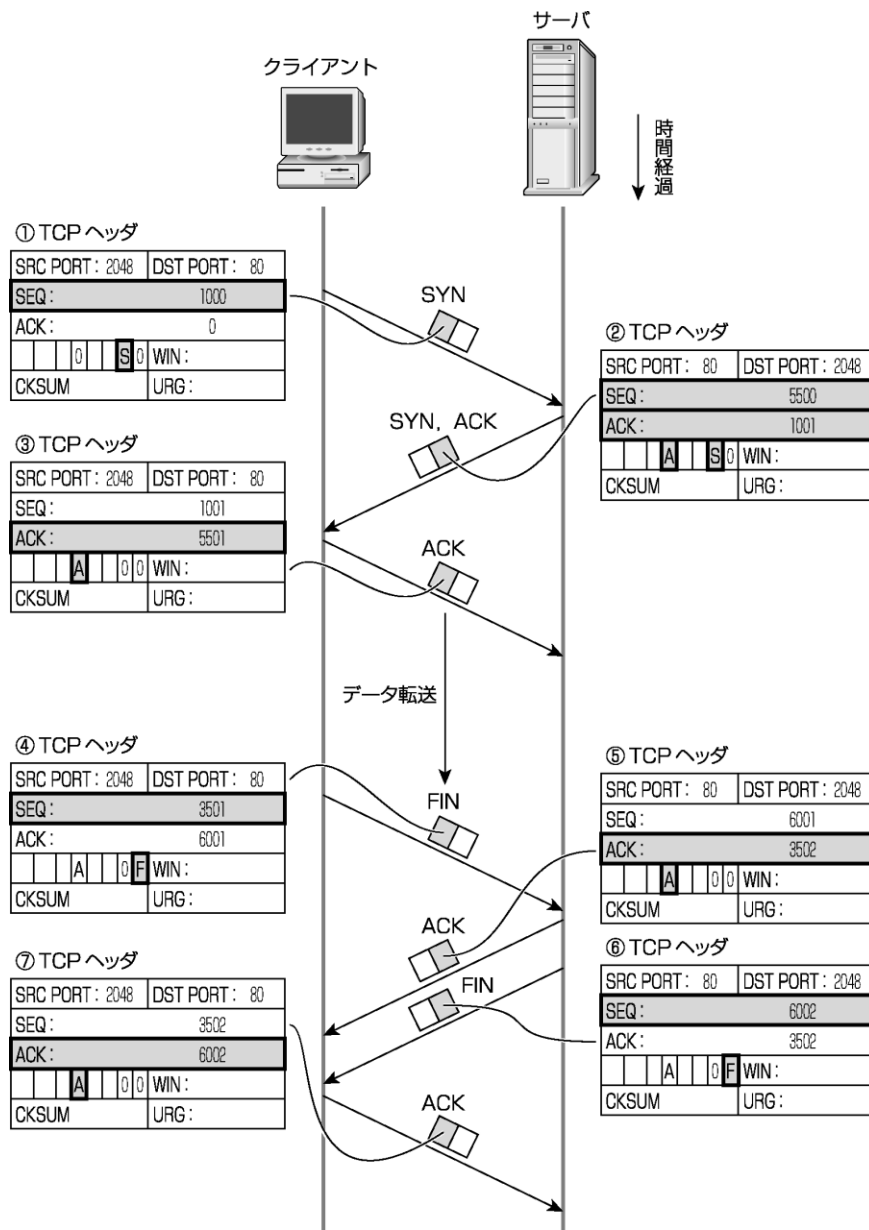
Checksum : 11110111

計 : 11111111

オール1の0になれば誤り無し

※ オール0の0になることはないため、オール0の0はチェックサムなしと扱える

TCPコネクションの確立と切断



POINT 1

SYNフラグの設定されたセグメントを双方が送信する

POINT 2

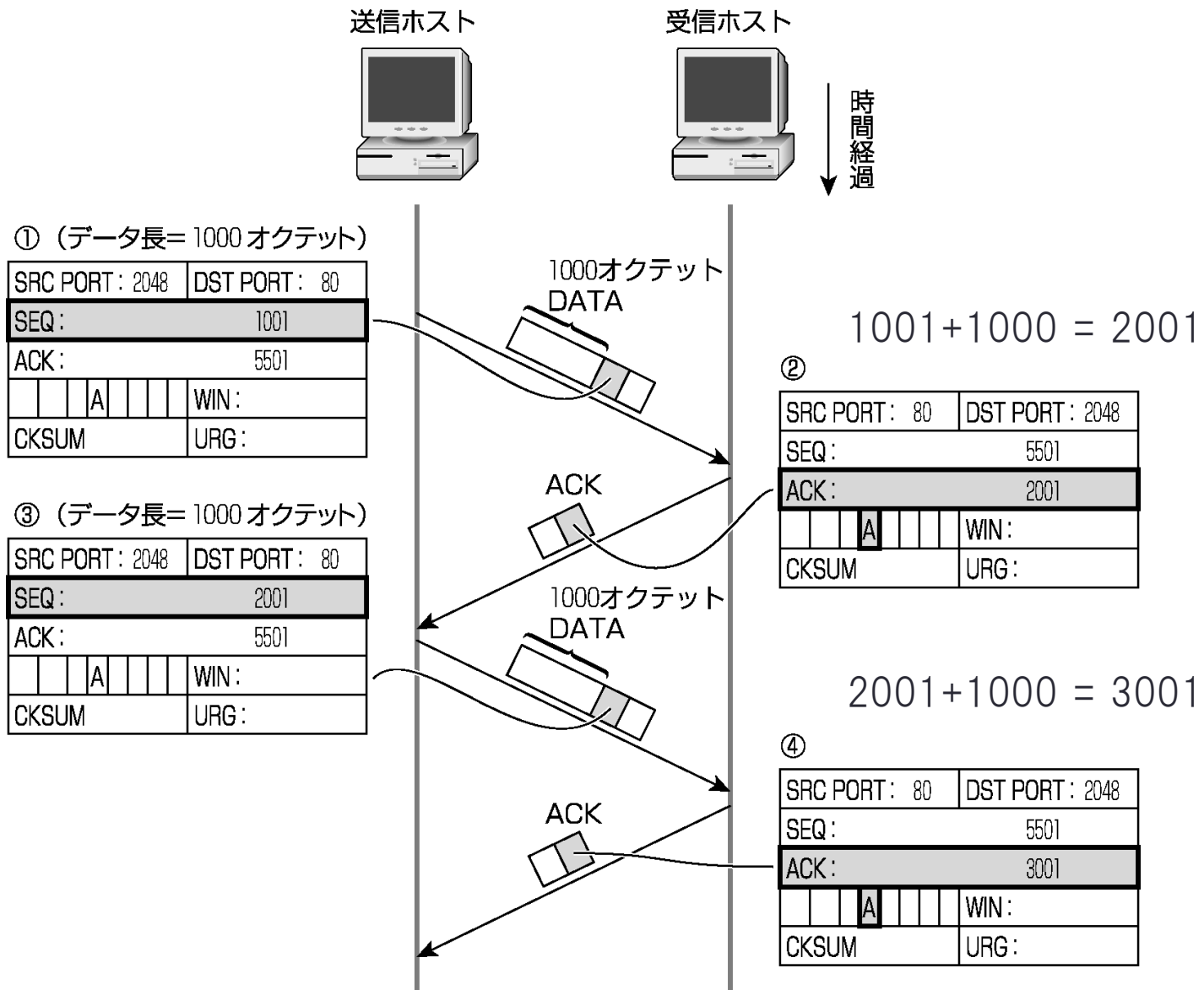
最初のSYNフラグの設定されたセグメント以外はACKフラグを設定する

POINT 3

FINフラグの設定されたセグメントも双方が送信する

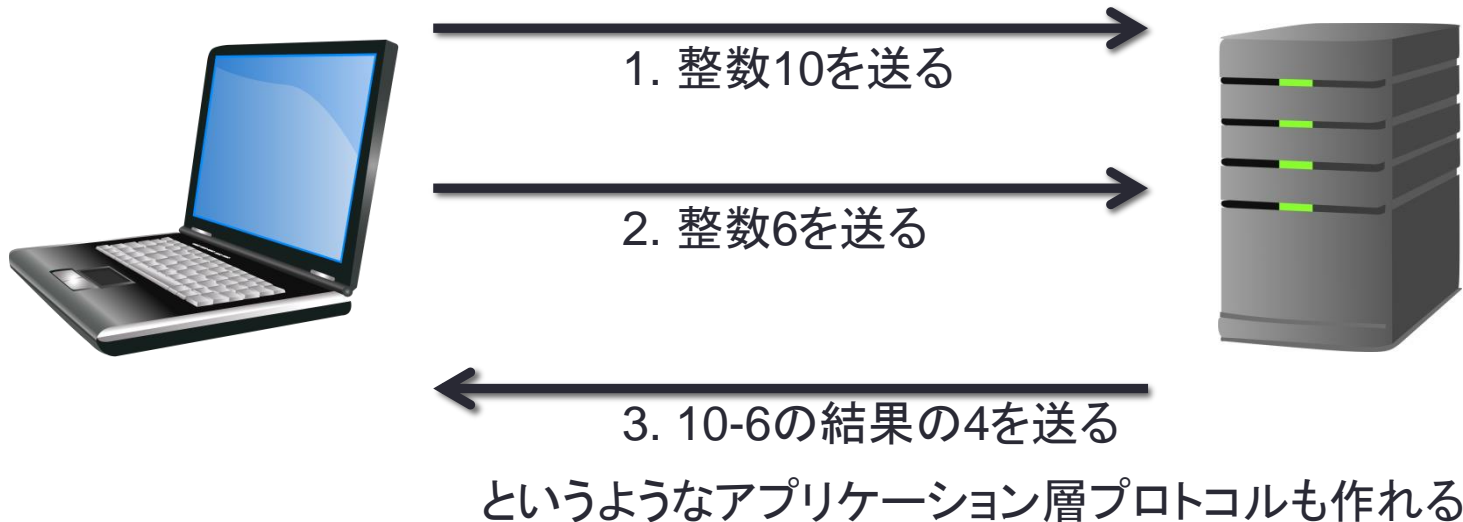
3wayハンドシェイクと呼ぶ

ACKによる信頼性の提供



TCPコネクションの利点

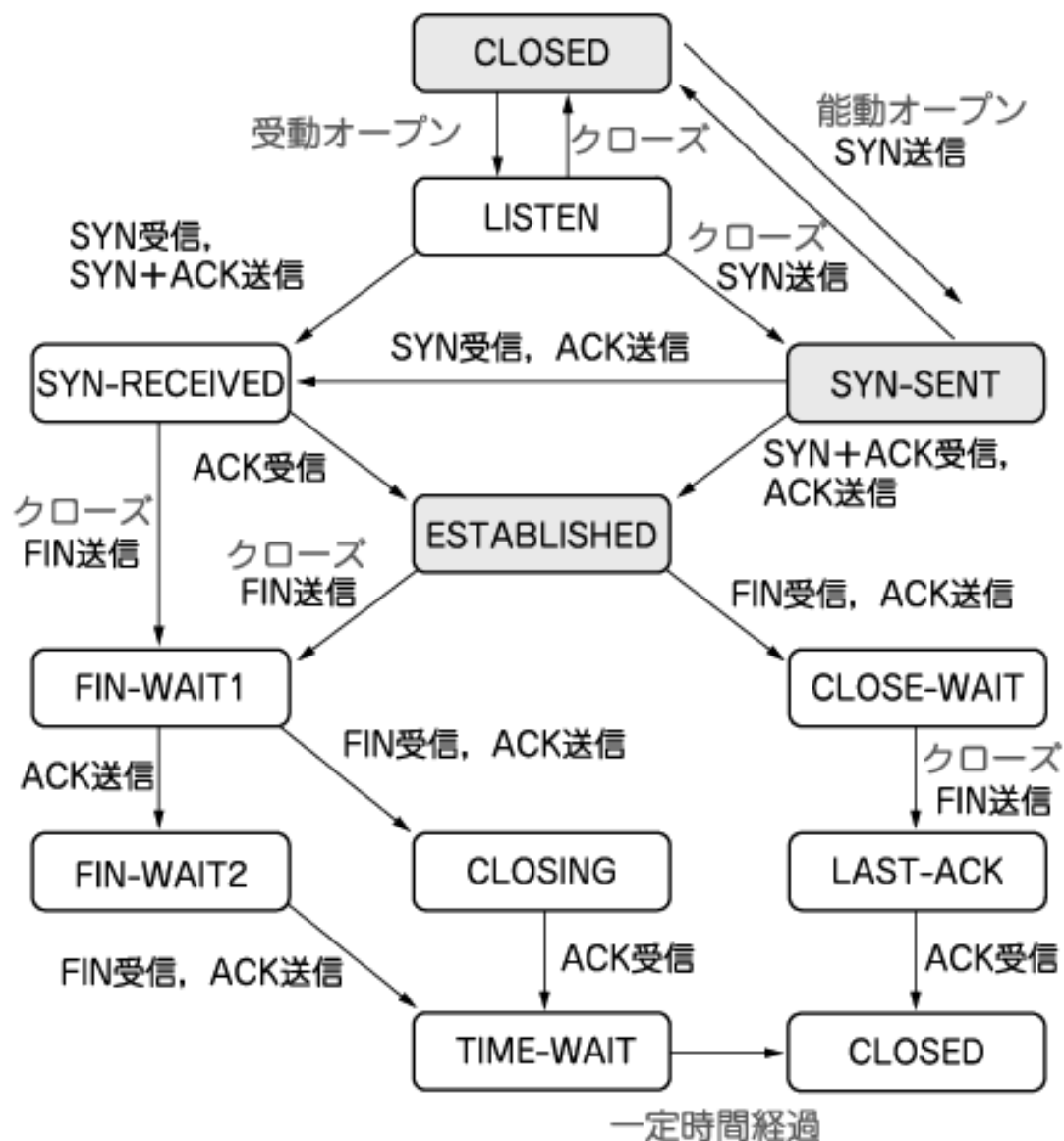
- 到達順番が保証される
 - 到達順が狂っても受信側TCPで順番を入れ替えて上位に渡す
- データの誤りや欠損がない
- 上位層で状態遷移を用いたプロトコルを作成できる



TCPのタイムアウトと再送

- 送信データに対するACK応答を監視する
 - ※信頼性確保のため
- 一定時間ACKが来ない場合は再送する
 - 待ち時間は指数関数的に増加させる
 - 1回目0.3秒後, 2回目0.7秒後, 3回目1.4秒後……
- 一定回数再送してもACKがない場合はRSTフラグを送り, コネクションを初期状態にする
 - デフォルトは5回

TCPの状態遷移



netstatによるTCP状態確認

```
t-matsu@X220Tablet ~ $ netstat -p TCP
```

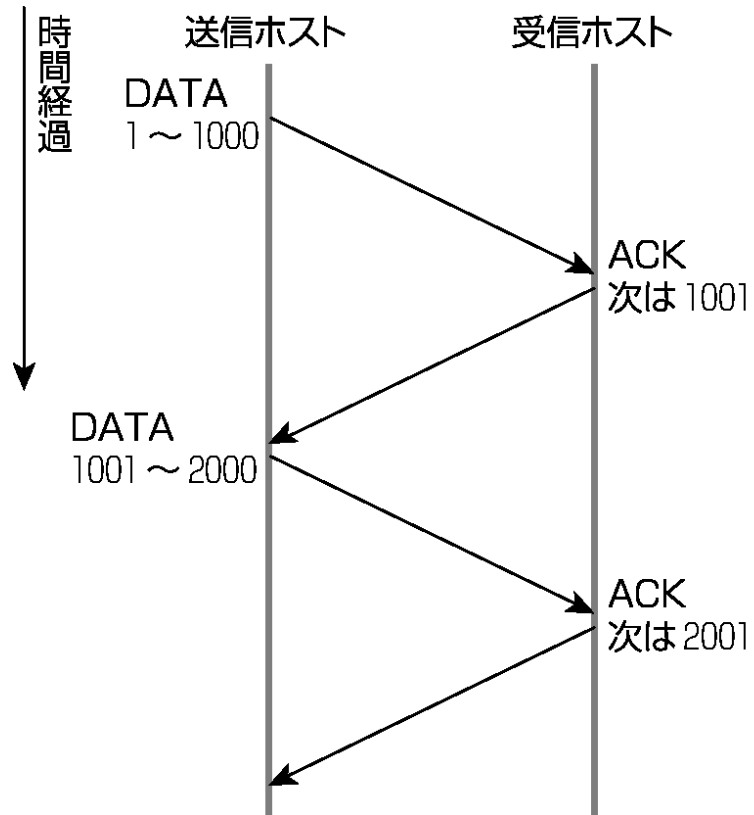
アクティブな接続

プロトコル	ローカル アドレス	外部アドレス	状態
TCP	127.0.0.1:5354	X220Tablet:49158	ESTABLISHED
TCP	127.0.0.1:5354	X220Tablet:49159	ESTABLISHED
TCP	127.0.0.1:49158	X220Tablet:5354	ESTABLISHED
TCP	127.0.0.1:49159	X220Tablet:5354	ESTABLISHED

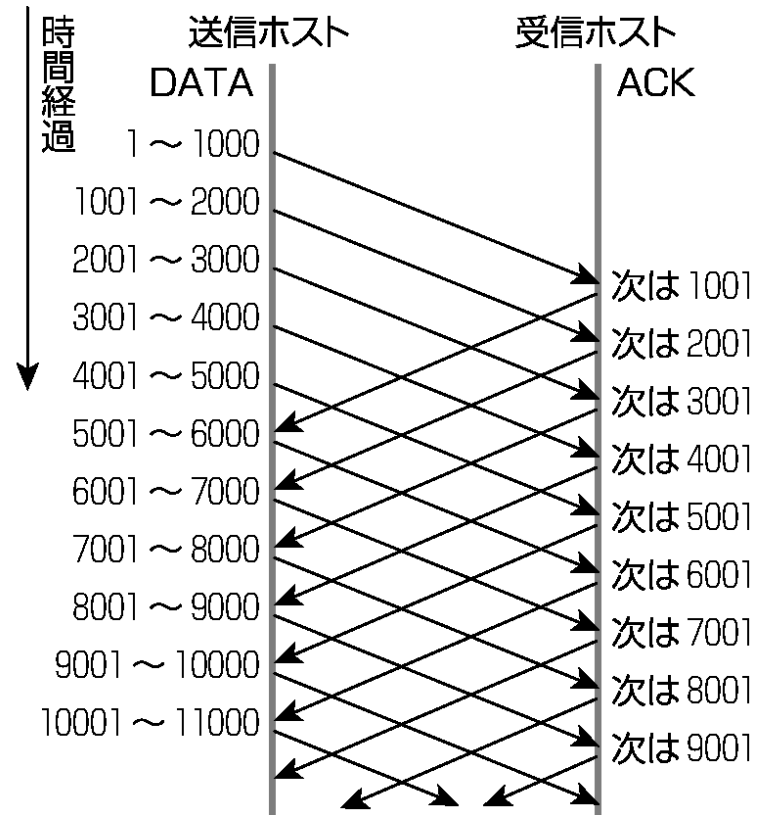
TCP	192.168.0.151:63532	mail:ssh	ESTABLISHED
TCP	192.168.0.151:63705	ec2-52-72-198-82:https	CLOSE_WAIT
TCP	192.168.0.151:63707	server-54-230-232-190:https	TIME_WAIT
TCP	192.168.0.151:63710	server-54-230-232-190:https	ESTABLISHED
TCP	192.168.0.151:63711	nrt12s13-in-f3:https	ESTABLISHED
TCP	192.168.0.151:63712	nrt12s01-in-f4:https	ESTABLISHED
TCP	192.168.0.151:63713	nrt12s01-in-f14:https	ESTABLISHED
TCP	192.168.0.151:63714	nrt12s13-in-f14:https	ESTABLISHED
TCP	192.168.0.151:63715	nrt12s12-in-f2:https	ESTABLISHED

```
t-matsu@X220Tablet ~ $
```

ウィンドウ制御

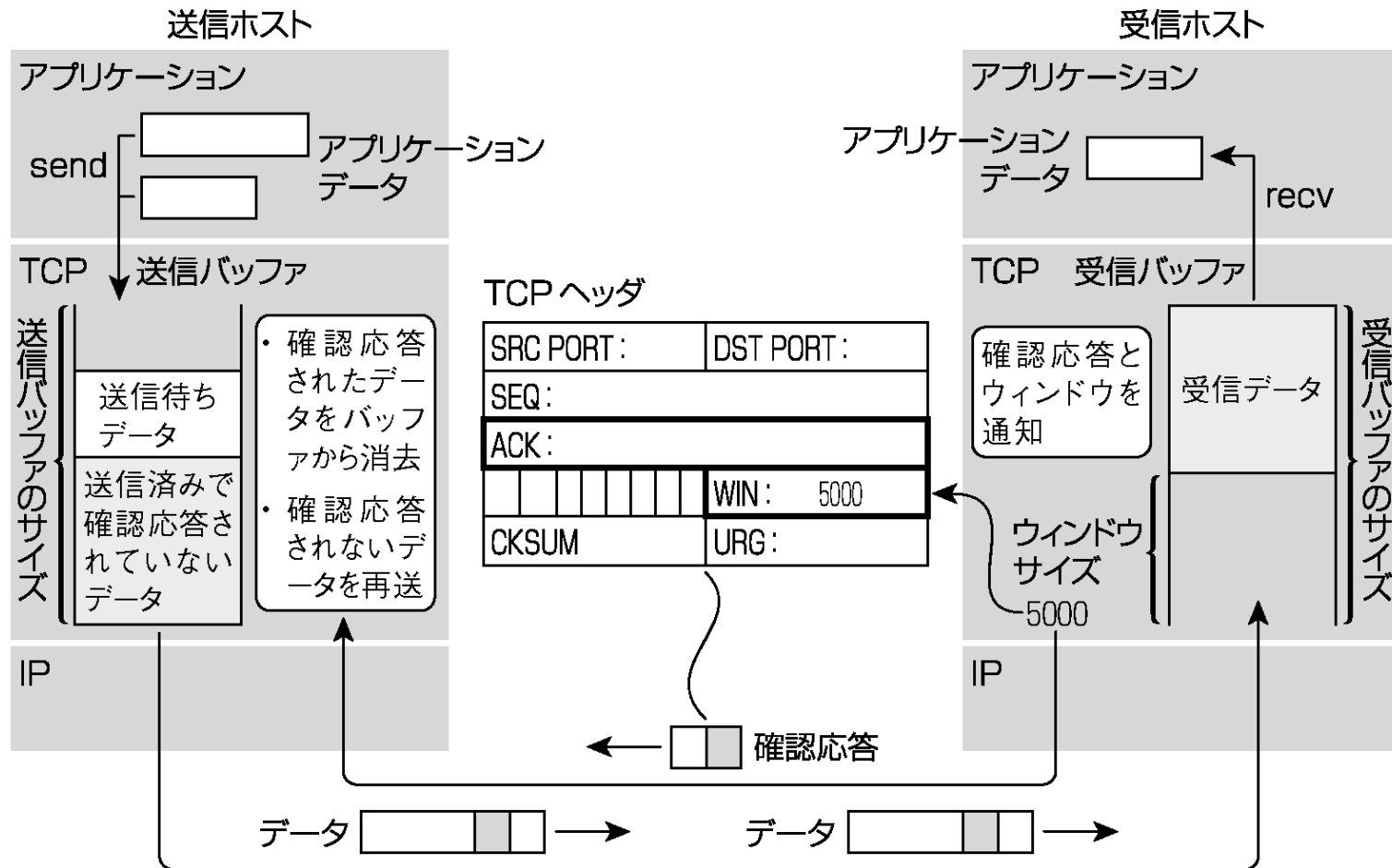


ウィンドウサイズ1000



ウィンドウサイズ5000

ウィンドウ制御(Contd)



UDP(User Datagram Protocol)

- 信頼性はない(IPの低信頼性を維持)
- ポート番号と呼ばれる識別子があり, サービスの区別を行う
- チェックサムで誤りの検知を行い, 誤りがある場合は破棄する
 - オプションでチェックサム計算なしの指定も可能
- コネクションレスの通信(パケット単位で独立)を行う

UDPヘッダ

0	16	31
始点ポート番号 Source Port uh_sport	終点ポート番号 Destination Port uh_dport	
パケット長 Length uh_ulen	チェックサム Checksum uh_sum	

始点ポート番号 送信ホストの使用するポート番号

終点ポート番号 受信ホストの使用するポート番号

パケット長 UDPヘッダとデータの長さ(単位8オクテット, 最大65535オクテット)

チェックサム データの信頼性を確かめる値 (後述)

今回のまとめ

- TCP
 - 信頼性のある通信を行う
 - コネクション指向のプロトコル
 - 再送によるデータ欠落の補填を行う
 - フロー制御が可能
- UDP
 - 信頼性のないコネクションレス通信(IPの信頼性を維持)を行うプロトコル
- TCP,UDP共通
 - ポート番号によるサービス区別を行う
 - チェックサムによる誤り検知が可能

質問あればどうぞ

次回はアプリケーション層！