ネットワークが一般的に利用されるようになっており、OSにおいてネットワークを管理する機能が必須なものとなっています。教科書では第12章　ネットワークの制御です。

オペレーティングシステム
（2024年 第10回）

ネットワークと分散処理について

前回の課題について

詳細は「第9回の課題について.pdf」をみてください

デッドロックが起きないようにする、すなわち「防止」の観点から、以下のような点が考えられます。

- 複数の資源を必要とする処理では、一度にまとめて資源を確保する(「確保・待ち」の回避)

  講義資料p.10では、資源AとBをプロセス1と2がそれぞれ取り合っていましたが、たとえば、プロセス1がAとBを一度に確保してしまえば、プロセス2がBを確保することはなく、デッドロックは起きません

- 複数の資源を必要とするときは、確保する順序を同一方向にする(「循環待ち」の回避)

  講義資料p.10では、プロセス1がA→Bの順、プロセス2がB→Aの順で取りにいったので、一方を確保した後に他方を待つことでデッドロックになっています。どちらもA→Bの順で確保すれば、Aを確保できなかったプロセス(この場合はプロセス2)がBを確保することはなく、デッドロックになりません

- 使用済みの資源は直ちに解除する

  これは、発生しないことを保証するものではありませんが、常に気をつけるべきことで、デッドロックの発生確率を下げることができます

などです。

モニタに関する補足（Javaの同期機能について）

- 「synchronized」を使用して排他制御を行う
  - ブロックに指定するもの … 教科書 図8.10

```
synchronized(ロックするオブジェクト) {
  処理 – クリティカルセクション
}
```

  - メソッドに指定するもの

```
synchronized 戻り型 メソッド名(引数) {
  処理
}
```

    講義資料のモニタと同様のもの

- 同期は notify/notifyAll と wait メソッドを用いる
- モニタと比較して、条件変数がない

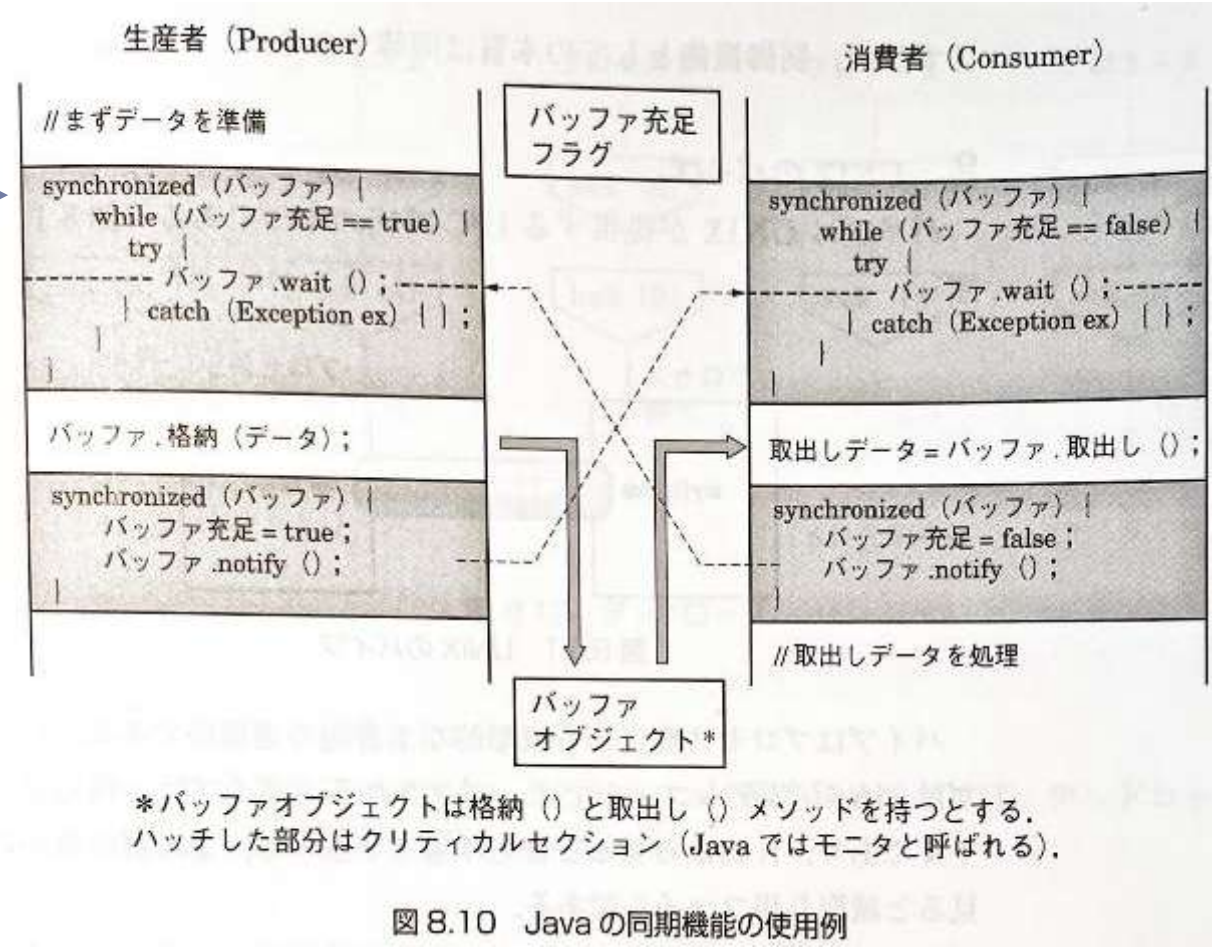


図 8.10　Java の同期機能の使用例

コンピュータネットワーク

- コンピュータネットワーク
  複数のコンピュータを接続する技術。また、接続されたシステム全体
  - コンピュータ同士を接続し、処理を分散する/データを分散する
    - 最も初期のネットワークは、集中管理型の汎用大型機と専用端末を独自のケーブルで接続したもの
  - 通信 … 機械を使い電気や電波によって情報やデータを伝達する（電気通信）、狭義の電気通信は特定の相手と情報を送受信すること
  - コンピュータネットワークの構成要素
    接続の構造、通信装置、通信を行う応用プログラム
- コンピュータネットワークとオペレーティングシステム
  - オペレーティングシステムにおける通信装置の仮想化
    同じアプリケーションプログラムを異なる種類の通信装置で利用したい、同じ通信装置を用いながら別のアプリケーションプログラムを作成する、などアプリケーションプログラムから通信に関する部分を分離し、多種多様な通信装置に対応できる構成を採用

コンピュータを接続して処理やデータを分散する方式は古くからありますが、1970年代からデータ通信の進歩や通信基盤の整備にともなって、大型のホストマシンと端末といった集中管理型でなく積極的に分散処理を行うことが広まりました。コンピュータネットワークはその基盤となる技術です。

コンピュータネットワークの構成要素は、原理的には、通信網（伝送路や交換器などの構成）、コンピュータに装着される通信装置、通信しあうアプリケーションプログラムで構成されます。
このような構成では、アプリケーションプログラムで通信装置や通信手順も処理する必要があり、通信装置が替わっても変更しないですませたいという要求があります。

利用する側からはネットワークの詳細を意識せずにしたいわけです。そこで（メモリやファイルの管理と同様）仮想化を行います。

ネットワーク機能の位置づけ

- 入出力として
  通信 … 通信回線を介した入出力
  OSは通信制御機能を提供
- 論理的通信路として
  OSはアプリケーションプログラム同士が通信するための論理的な通信路を提供（APIを提供）
- 共用資源として
  コンピュータ間で二次記憶装置やプリンタなどを共用する場合、OSはそれらにアクセスする機能を提供（ネットワークOS）
- プログラムの存在場所として
  別のコンピュータ上のプログラムを呼出して、実行結果を返してもらう
  OSはこの呼出し（遠隔手続き呼出し）機能を提供

ネットワークはいろいろな見方ができます。

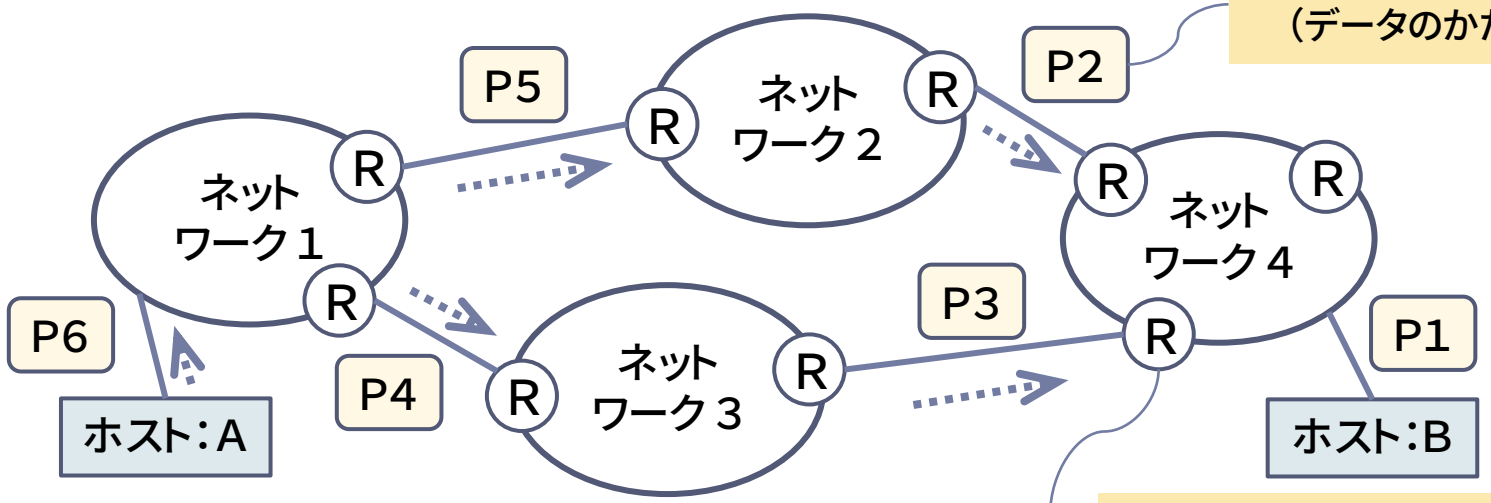
通信回線を使った入出力として見る場合には、OSが通信制御機能を提供していると考えられます。

入出力を使用してコンピュータ間でファイルやプリンタを共有するという見方もあります。この場合、OSはそれらの機器にアクセスする機能を提供します。

ファイルの内容がプログラムのこともあり、（プログラムの作成や修正ではなく）それを別のコンピュータから呼び出して実行するということもあります。その場合、OSはその呼出し機能を提供します。

基本的な考え方

- パケット分割による情報の送受信



パケット：情報の伝送単位（データのかたまり）

ルータ：ネットワーク間を相互接続する通信機器
各ネットワーク間でIPパケットをやり取りできるようにする装置

  - 各パケットには連続番号が付けられているが、ホストＢにパケットが到着する順序は不定
  - パケットはネットワークのどのルートを通ってもよい

- パケット交換
  回線交換のように連続的なデータではなく、回線を占有しないで、送信するデータをパケット（あるいはフレーム）という単位に分割して通信する方式
  - ルータは受け取ったパケットをメモリに一度蓄積し、その後ルート選択する

インターネットでのデータを交換する基本的な考えは、パケットに分割されたデータを個々に送り出して、受け取るということです。パケット交換と呼ばれます。

固定電話のような回線交換との違いは、送り手と受け手の間に占有された接続を確立して連続データを流すというのではなく、回線を占有せず（すなわち、他の人もデータを流す中で）、送信するデータをパケットの単位に分割して通信するという点です。ルータは、受け取ったパケットをメモリに蓄積して、その後ルート選択します。

重要な点は、送られたパケットが消失したり、送られた順とは違った順で届いたりするかもしれないということで、それに対処できるような仕組みになっています。

- プロトコル
  複数の者が対象となる事項を確実に実行するための手順等について定めたもの。「規定」
  - マシンやソフトウェア同士のやりとりに関する概念を指すためにも用いられるようになった
  - コンピュータ間で受け渡されるデータの形式と意味を定めたもの
- プロトコルスタック
  コンピュータネットワークの通信に関するソフトウェアのモジュール群の階層
  - 各プロトコルモジュールは上下の他のモジュールと互いに通信する
    一般に、これを、プロトコルが積み重ねられたものとして見ることになる
    最下層のプロトコルはハードウェアとのやり取りを行う層
  - プロトコルスタックをカーネルに内包し、ネットワーク機能を応用プログラムに提供する

プロトコルは、他の分野でも使われる言葉ですが、ネットワークでよく使用されるものです。

ネットワーク関連以外でもソフトウェアの分野では、機能を実現するモジュールを箱にみたて、下位のレベルから順に積み上げた階層で表現する方法がよく採られます。ソフトウェアスタックと呼びます。

(ソフトウェアの観点では)ネットワークの通信に関するソフトウェアモジュールの階層関係をプロトコルスタックと呼んでいます。

プロトコルスタックは通信装置というハードウェアを制御するので、OSの中に組み入れ、それによってコンピュータネットワークの機能をAPIとして提供します。

プロトコルスタックの目標

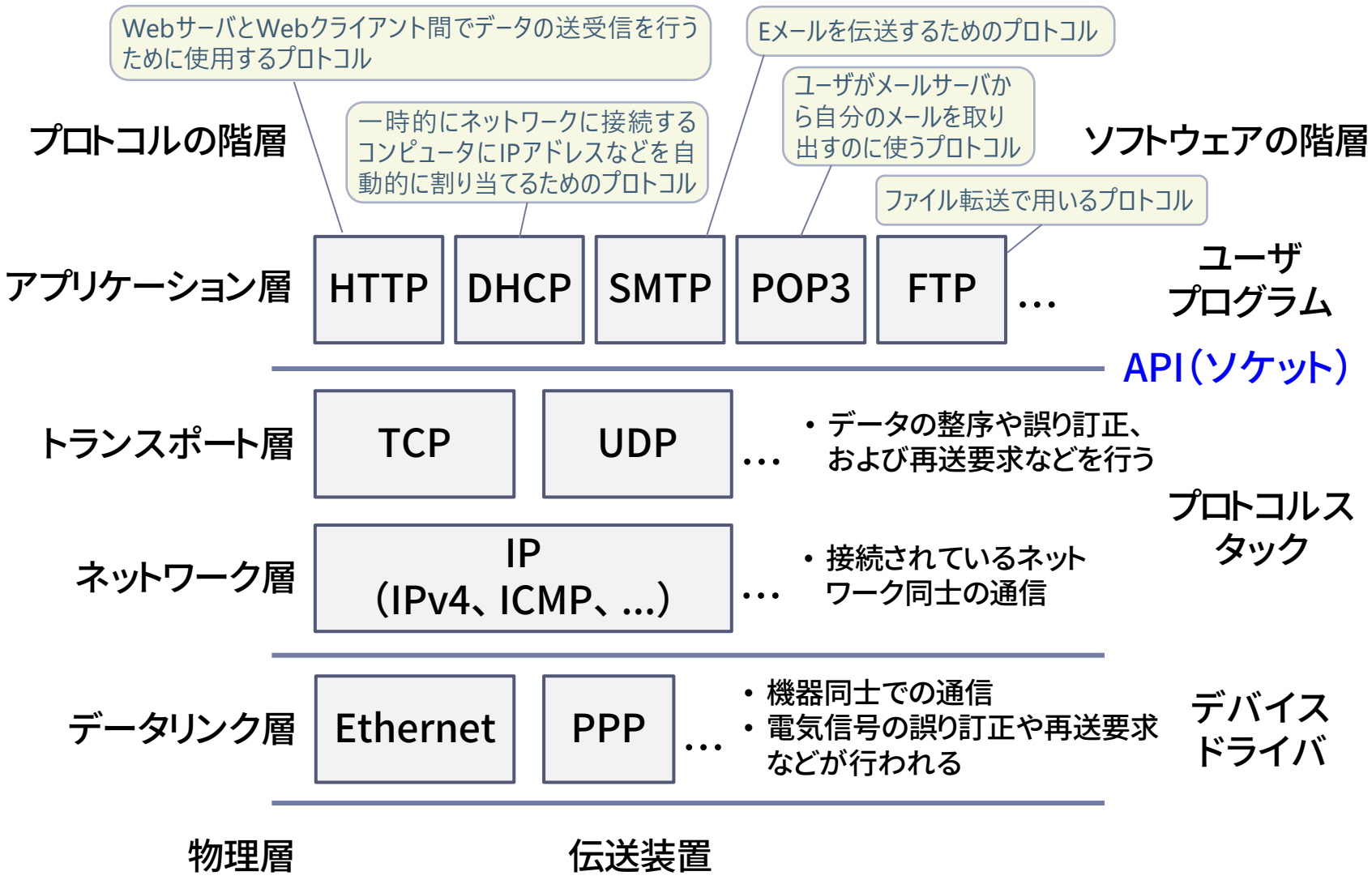
- 通信装置の仮想化
  アプリケーションプログラムに装置独立な通信の基盤を提供
- プロトコルの隠ぺい
  プロトコル独立なAPIの提供
- エラー処理
  応用プログラムが処理本体に専念できるよう通信に伴う各種エラー処理を実施
- サービスの構築
  応用プログラムに対して共通のデータ通信基盤を提供する
- 通信装置を効率よく利用する枠組みを提供する

「ソケット」と呼ばれるプロトコルスタックとAPIがUNIX、Windowsなどで広く採用されている

プロトコルスタックの目標を示しています。

特に本質的な役割は、アプリケーションプログラムを通信装置やプロトコルから独立にして、一貫したAPIを利用できるようにすることです。

プロトコルとソフトウェア階層



プロトコルとソフトウェアの階層との対応を示します。
下から、物理層はハードウェアの通信装置を扱うものであり、特定のハードウェアに依存することなく多種多様な装置が使用できます。

データリンク層は通信装置間でのデータの送受信を行う層です。この層のEthernetはよく聞く名前です。（PPPは、電話回線などでの接続のために作られたものです）

次のネットワーク層以上が、インターネット、TCP/IPの規格です。

ネットワーク層はパケットの転送を行う層で、プログラム間でのデータを格納するパケットと経路の基本的な制御を行います。
トランスポート層はデータ転送でのエラーチェックや再送制御などを行い、高信頼なデータ転送、または信頼性は低いが高速なデータ転送のための制御を行います。
アプリケーション層はアプリケーションによって規定されるプロトコルです。

一般にアプリケーション層のプロトコルは、ユーザプログラムによって規定されており、この層のプロトコル処理はユーザプロセスによって行われます。
トランスポート層とネットワーク層の部分がOSのプロトコルスタックとして実装されており、ここで煩雑なTCP/IPなどの処理を隠蔽しています。
トランスポート層とアプリケーション層のインターフェースがソケットのAPIとなっています。

参考（OSI基本参照モデル）

- OSI（Open Systems Interconnection）の7階層モデル
  - 国際標準化機構（ISO）によって策定された、コンピュータの持つべき通信機能を階層構造に分割したモデル（OSI基本参照モデル、OSIモデル などとも呼ばれる）
  - 通信機能（通信プロトコル）を7つの階層に分けて定義
    - 物理層：データを通信回線に送出するための電気的な変換や機械的な作業を受持つ
    - データリンク層：通信相手との物理的な通信路を確保し、通信路を流れるデータのエラー検出などを行う
    - ネットワーク層：相手までデータを届けるための通信経路の選択や、通信経路内のアドレス（住所）の管理を行う
    - トランスポート層：相手まで確実に効率よくデータを届けるためのデータ圧縮や誤り訂正、再送制御などを行う
    - セッション層：通信プログラム同士がデータの送受信を行うための仮想的な経路（コネクション）の確立や解放を行う
    - プレゼンテーション層：セッション層から受け取ったデータをユーザーが分かりやすい形式に変換したり、アプリケーション層から送られてくるデータを通信に適した形式に変換したりする
    - アプリケーション層：データ通信を利用した様々なサービスを人間や他のプログラムに提供する
- セッション層とプレゼンテーション層はあまり活用されておらず、それらを除いた5階層モデルで考えることが多い

プロトコルの階層は、OSIで7階層のモデルとして定義されていて、このように分かれています。

前ページで説明した層は、それぞれこの参照モデルで定義された層に相当するものになります。

ここで、セッション層とプレゼンテーション層はあまり活用されておらず、それら二つを除いた5階層として前ページのモデルで考えることが多いです。

ソケット、TCP/IP

- ソケット
  - C言語によるアプリケーション開発でのプロセス間通信、特にコンピュータネットワークに関するライブラリを構成する
  - 4.2BSD UNIX（1983年リリース）で初めてAPIとして実装された

  ネットワークの抽象化インタフェースとしてのデファクトスタンダードとなっている

- TCP/IP

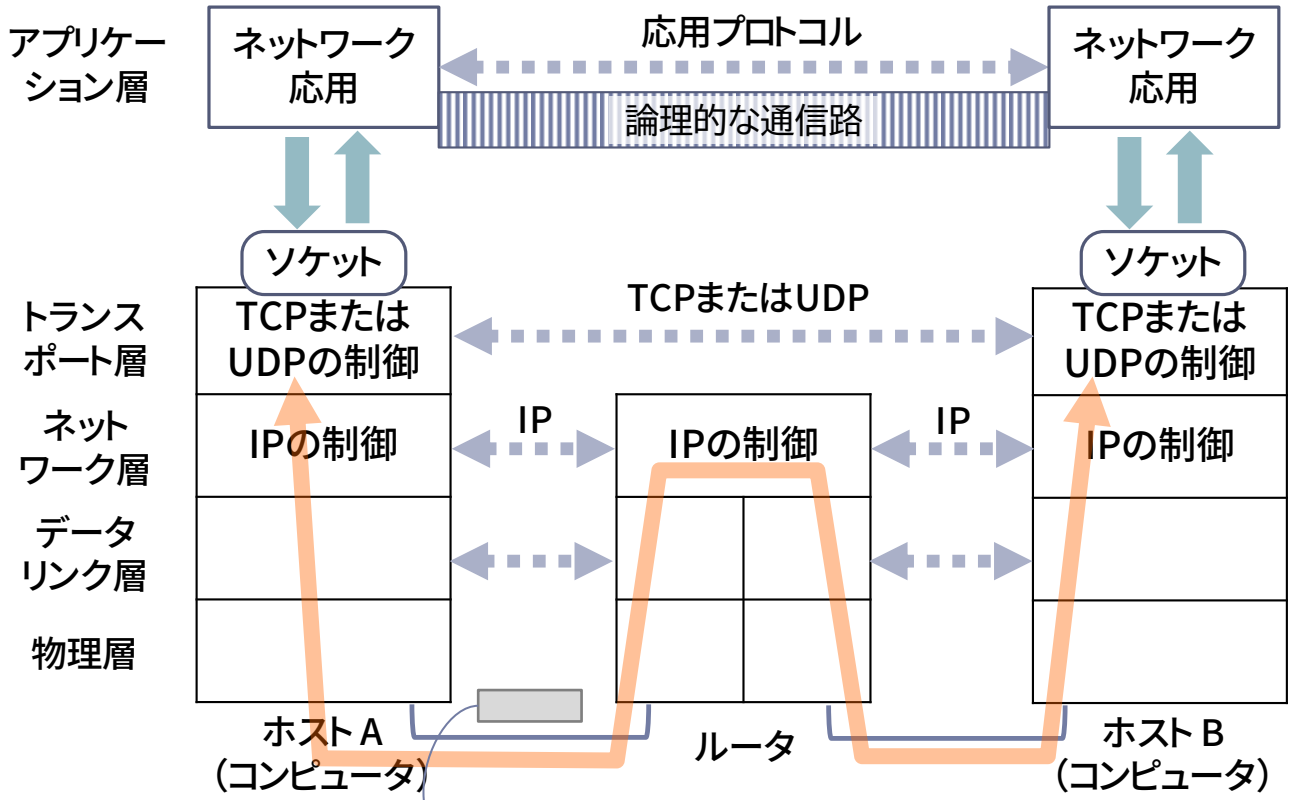
  インターネットなどで標準的に用いられる通信プロトコル（通信手順）
  - TCP（Transmission Control Protocol）とIP（Internet Protocol ）を組合わせたもの
  - また、TCP とIPを含むインターネット標準のプロトコル群全体の総称

  IPと組み合わせてTCPではなくUDPなどが用いられることもあるが「IPを中心とする標準的な通信プロトコルの総称」を表すことが多い

ソケットはアプリケーションからみた通信データの出入り口です。（プロセス間通信の一種です）

OSI参照モデルのネットワーク層に相当するIPは、その上位プロトコルであるTCPと組合わせて利用されることが多く、IPを中心とする標準的な通信プロトコルとしてTCP/IPと呼ばれることもあります。

TCP、UDP、IPなどの位置づけ



IPは、ネットワーク内の任意のホスト(コンピュータや端末)間の通信を可能にするためのプロトコルです。IPが処理する情報(パケット)はルータを何段も中継して目的のホストに届きます。

IPプロトコルは、ネットワークに接続されているすべてのコンピュータに対して、IPアドレスと呼ばれる番号を付与し、その番号を用いて送信元ホストから宛先ホストへデータ(パケット)を中継し、転送するものです。

TCPやUDPで、2つのホスト内にあるアプリケーション間での通信を実現します。これらによって、アプリケーション間に論理的な通信路が提供されます

パケット：ユーザデータ本体の他、発信元と送信先のアドレスや、パケットの種類、通し番号等の制御情報などを含む

- IPは複数のネットワークを繋ぎあわせて相互に通信可能にするプロトコル
  これを用いて世界的に様々な組織の管理するネットワークを相互接続してできたオープンなネットワークをインターネットと呼んでいる

IPアドレス

- IPアドレス
  TCP/IPを使ったネットワークにおいて、お互いのシステム（ホストやノードと言う）を区別するための識別子（コンピュータや通信機器1台1台に割り振られた識別番号）
  IPアドレスによって通信相手を識別・特定する
  - IPアドレスのサイズは、IPv4では32ビット、IPv6では128ビット

0から255までの10進数の数字を4つ並べて表現

  - 単なる数値の羅列であるIPアドレスはこのままでは人間にとっては覚えにくいため、コンピュータやネットワークに名前（ドメイン名やホスト名）がつけられている場合が多く、「DNS」（Domain Name System）というシステムによってIPアドレスとの相互変換が行われる

ネットワーク層では、ネットワーク全体にわたって通信相手を識別しなければなりません。このため、ノード（ホスト）ごとにアドレスが与えられています。

IPではこのアドレスとして「IPアドレス」が定義されています。
IPv4の場合は32ビット、IPv6の場合は128ビットでの値で指定されます。

DNSによって、たとえば123.124.125.1のような数値のIPアドレスから www.example.xx.jp のようなドメイン名（ホスト名）への対応付けができ、認識しやすいドメイン名でネットワーク上の資源にアクセスできます。

図は「@IT」Windows Server Insider/Windowsネットワークの基礎：第6回 TCP/IPの概要 による
http://www.atmarkit.co.jp/ait/articles/1503/05/news135.html

TCP、UDPプロトコル

- TCPはIPを基盤に、その上層で利用されるプロトコル
  IPネットワーク上の2地点間で「信頼性のある」通信を可能にする
  - 「信頼性がある」とは、送信したデータが送信した順に確実に相手にまで届くこと
    - 通信経路の途中でパケットが大幅に遅延したり、送信順とは異なる順番で届いたり、一部がエラーで化けたり、パケットそのものが廃棄されたりするなど、さま ざまなことが起こる可能性がある。このような状況を検出し、必要なら送信元に再送信を依頼したり、どうしても応答が得られないなら、アプリケーションにエ ラーとして伝える、などの処理を行う
  - TCPのさらに上層では、用途やソフトウェアに応じて様々なプロトコルが規定されている

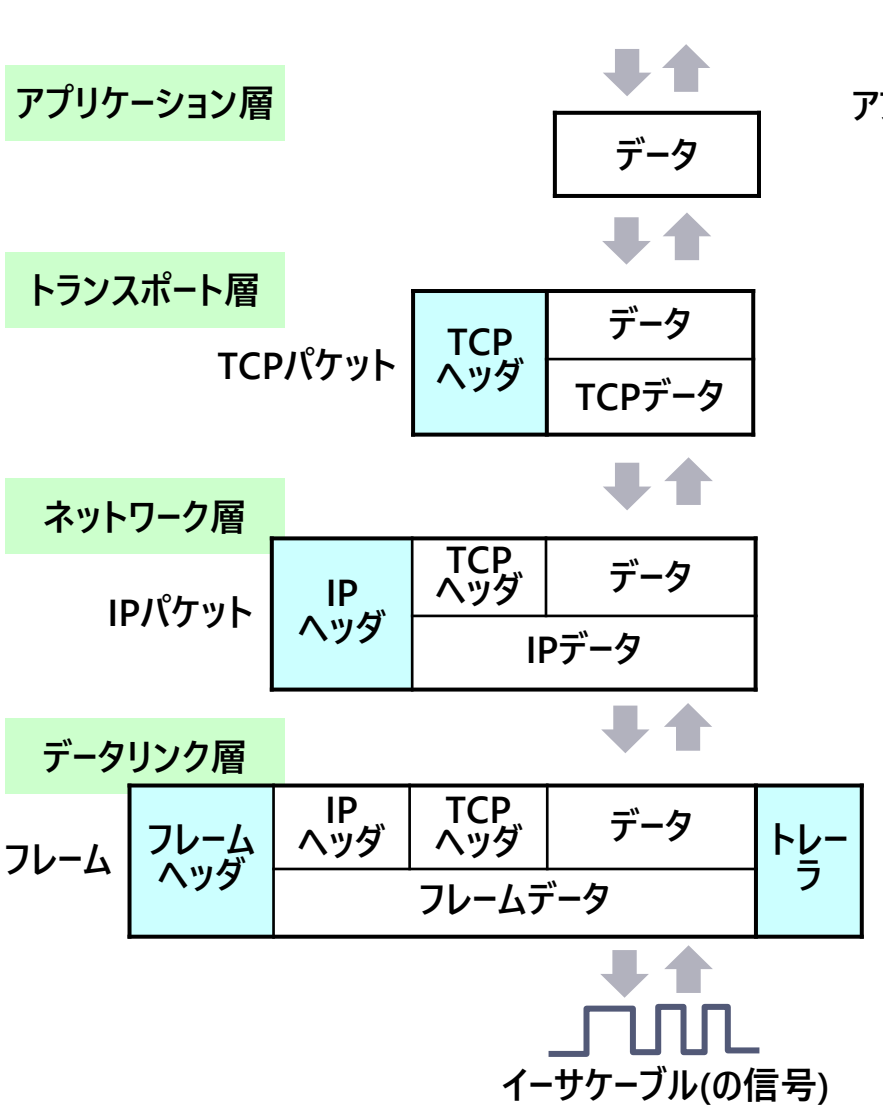
    例えば、WebではHTTP（Hypertext Transfer Protocol)が用いられ、HTTPはTCPを、TCPはIPをそれぞれ利用してデータを転送している
- UDP
  パケットが壊れていないかチェックする機能はあるが、エラー訂正のための再送などは行われない

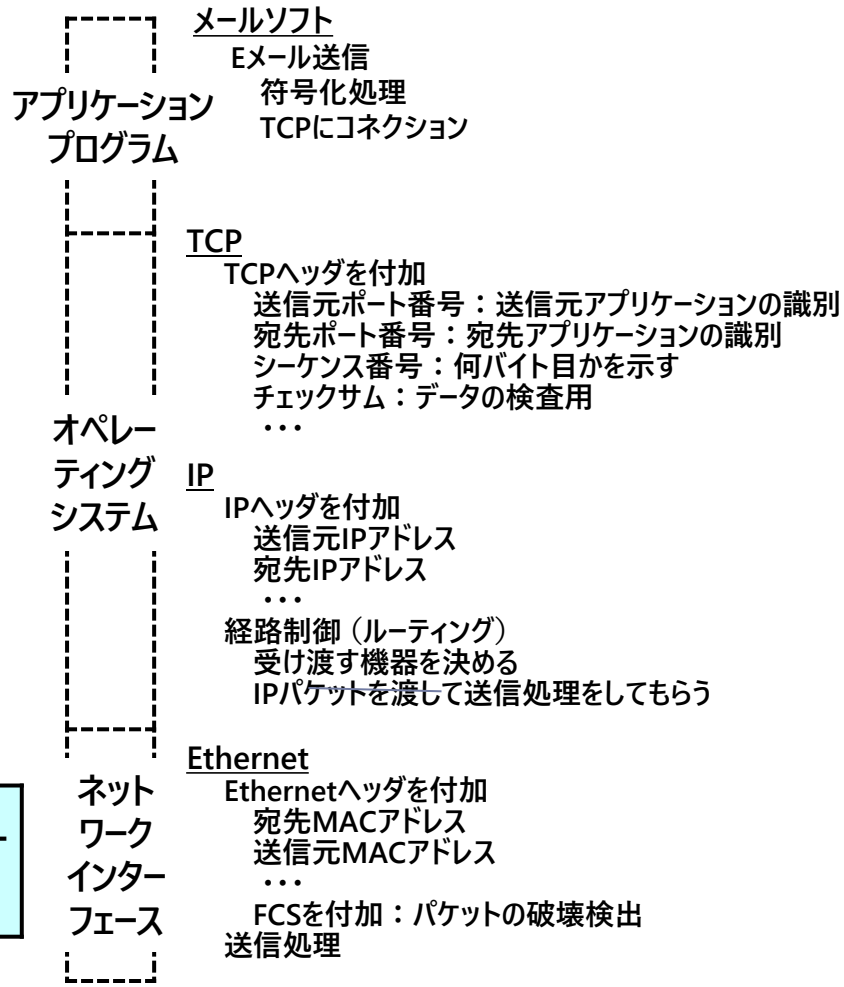
OSI参照モデルのトランスポート層に相当するTCPはコネクション型（通信を開始する前に相手との接続関係を確立する方式）のプロトコルで、信頼性の高い通信が実現されます。

なお、 同じくトランスポート層に相当するUDPはコネクションレス型です。エラー時の再送処理などを行わず、オーバヘッドが少ないという利点がありますが、信頼性は低くなります。

また、TCPでは1対1の通信しかできませんが、UDPではブロードキャスト（ネットワーク内の不特定多数に送信）やマルチキャスト（決められた特定の複数へ送信）が利用できます。

プロトコルのカプセル化





IPにおいては、送るべきデータをカプセル化します。

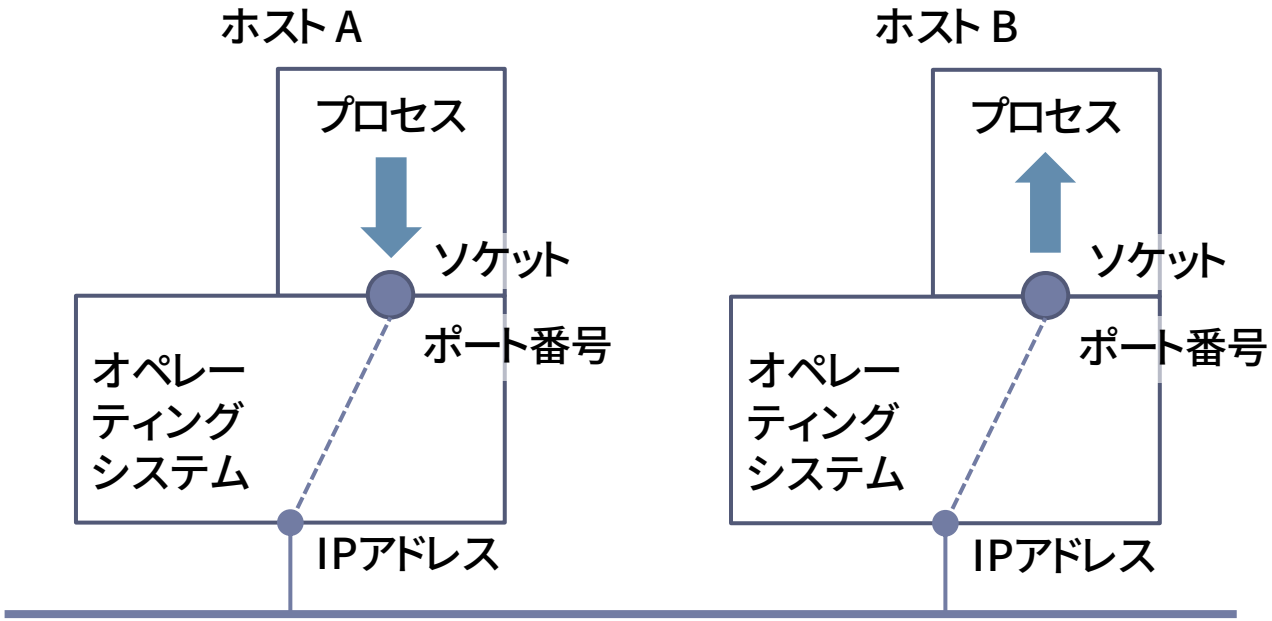
たとえば、ネットワーク層でIPによってデータを送受するとき、データの本体（IPデータ）にIPヘッダが付加されます。IPデータの中身は、上位層（TCP）でのTCPヘッダとデータに分かれていますが、ここでは、IPデータは単なるデータとして扱われます。すなわち、TCPヘッダ＋データがカプセル化されていることになります。

層の上側に向かうとヘッダが次々外されて必要なデータを得、下側に向かうと必要なヘッダが付加されることになります。

図の右側のアプリケーション（メールソフト）の動作説明では、メール送信で下層に向かってヘッダを付加して行く様子が記述されています。

通信用プログラミングインターフェース

- ソケットによる通信の概要
  ソケットはアプリケーションから見た通信データの出入り口
  - 相手アプリからの通信データはソケットから取り出す
  - 相手に渡すデータは、ソケットに流し込む
  - ソケットは、アプリケーションからの要求で作られ、ポートが割当てられる
    - ポートは特定のプロセスやサービスの種類を特定するもの（番号）



アプリケーションどうしがソケット機能を使って通信します。

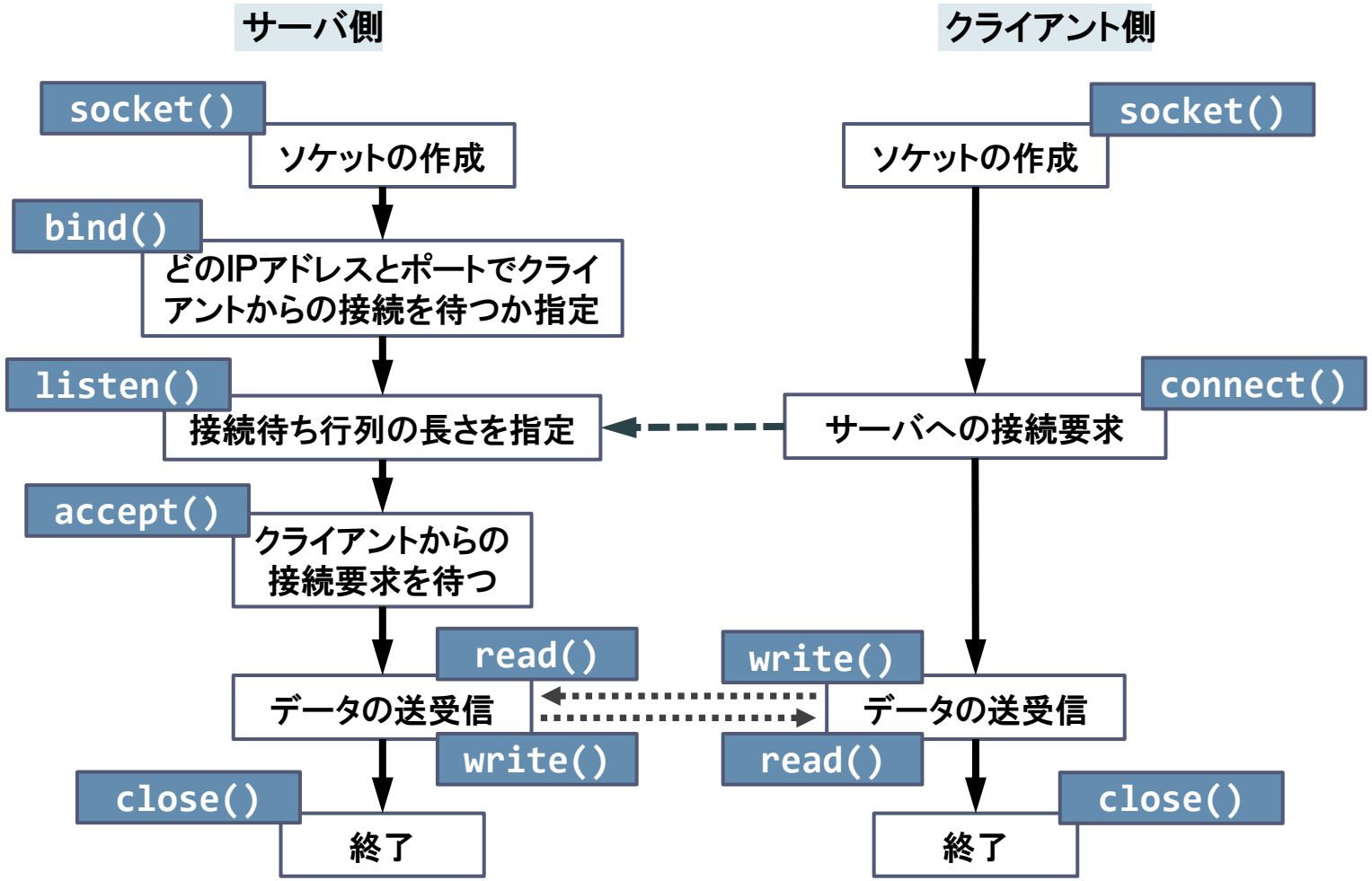
ソケットは電気のコンセントのようなものであり、相手からの通信データはソケットから取り出し、相手にわたすデータはソケットに流すことになります。

アプリケーションからの要求でソケットが作られると、ソケットにはポートが割り当てられてポート番号が与えられます。
ポート番号は、通信に使用するプログラムを識別するための番号で16ビットの整数です。

IPアドレスでネットワーク上のコンピュータ（ホスト）を一意に識別しますが、該当コンピュータのどのプログラムに通信パケットを届けるかを決定するためにポート番号が使用されます。（HTTPを使って通信する際は、該当サーバのIPアドレスとポート番号（たとえば80番）を指定してパケットを送るなど）

ソケットAPIを用いた通信プログラムの流れ

- コネクション型の場合



図中のsocket()やbind()などで示されるソケットAPIを用いて、TCPに対応した通信を行う場合のプログラムの流れはこのようになります。

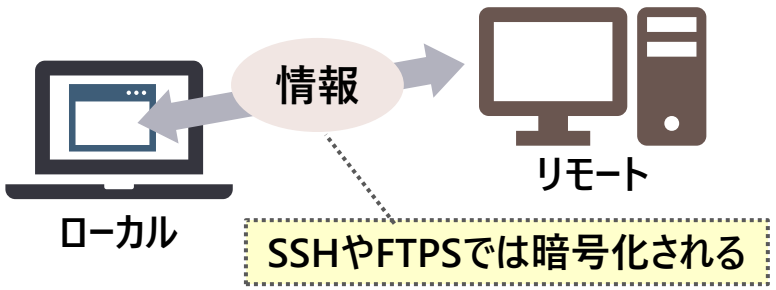
サーバ側では、ソケットを作成してどのクライアントからの接続を待つかを指定し、クライアントからの接続を待ちます。
クライアントでは、ソケットを作成し、サーバへ接続要求します。
接続できると送受信が行え、終了すると切断してソケットを削除し、通信を終了します。

connect()やread()、write()では待ちが生じることがありますが、その処理はOSが行います。

ネットワークを介したOS機能の利用

コンピュータネットワーク内で、あるコンピュータから、ネットワーク接続された他のコンピュータ(リモートマシン)にアクセスが行える仕組み

- リモートマシンへの接続 … 相手側のアプリケーションを実行
  telnet、SSH でリモートログイン
- リモートマシンとのファイル転送
  (ローカルからリモートへ、逆も)
  FTP、FTPS など



- OSによるファイル共有(分散ファイルシステム)
  OSが提供するネットワークでのファイル共有は、基本的にファイルシステムの一部であり、ローカルファイルと同じように共有ファイルを操作できる
  - Windowsのファイル共有は、SMBプロトコルがサービスを提供する
    - Linux(UNIX系OS)ではSambaと呼ぶソフトウェアでSMBのサービスが提供される
  - UNIX系OSのファイル共有は、NFSがサービスを提供する

telnetは、TCP/IP接続されたマシンにネットワーク経由でログインして、そのマシンを遠隔操作するための端末の機能を提供するプロトコルです。telnetの場合、パスワード情報を含め全てのデータが暗号化されずに送信されますが、SSHではパスワード情報を含め全てのデータが暗号化されて送信されます。

ファイル転送では、FTPやFTPSプロトコルで送受信を行います。

通信を暗号化しない操作は使用を推奨されないようになっています。

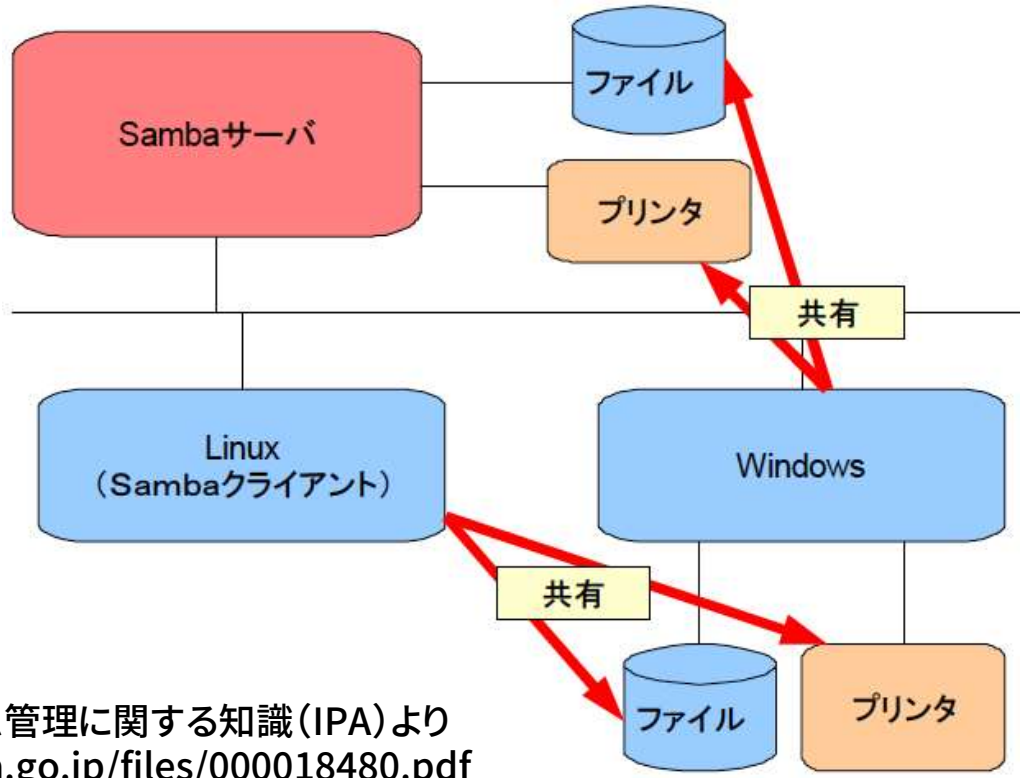
分散処理では、ネットワークに接続されたコンピュータをあたかも1台のものであるかのように利用できることが重要です。別のコンピュータに存在するファイルをその存在場所を意識させずに、ローカルなファイルと同じように操作できる機能がファイル共有です。

WindowsではSMBと呼ばれる(アプリケーション層の)プロトコルがTCPやUDPで他のコンピュータとやりとりし、そのサービスを提供します。
UNIXではSambaと呼ばれるソフトウェアでSMBの機能を提供しており、両者でファイル共有ができます。

UNIX系のOSどうしでは、NFS(ネットワークファイルシステム)によってファイル共有を行います。

Samba

- LinuxとWindowsとの間でファイルやプリンタといった資源を共有するためのソフトウェア
  - Sambaサーバでは参加するワークグループ、共有するディレクトリやアクセスするユーザおよびコンピュータの設定等を行う
  - Windowsからは「ネットワークコンピュータ」としてアクセスすることが可能であり、Linuxからは Sambaクライアントを使用してアクセスを行う



Linuxのシステム管理に関する知識（IPA）より
http://www.ipa.go.jp/files/000018480.pdf

LinuxでSambaサーバを構築してファイルやプリンタを共有できるようにします。

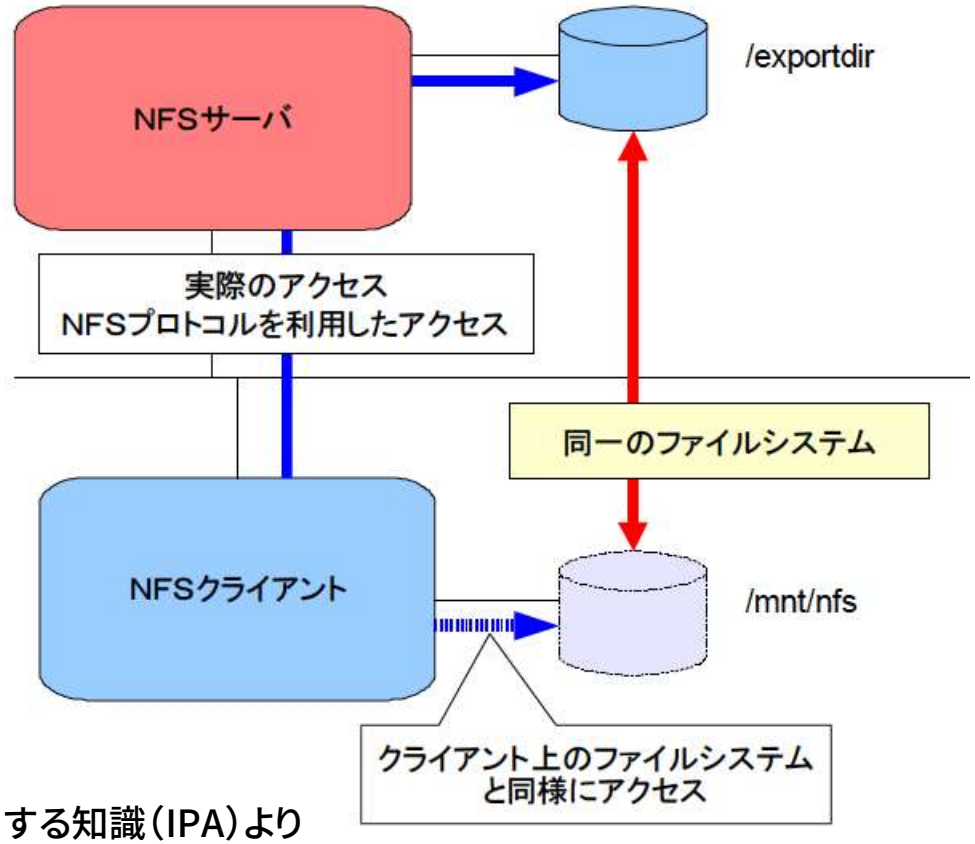
Windowsからは、SMBによりネットワークコンピュータとしてアクセスできます。
Linuxからは、Sambaクライアントを使用してアクセスできます。

NFS（Network File System）

- サーバ上にあるファイルシステムを、クライアントでマウントすることにより、クライアント上にあるファイルシステムと同様に使用できるサービスを提供する

NFSサーバ上のどのファイルシステムを他のコンピュータが共有してよいのか設定を行う

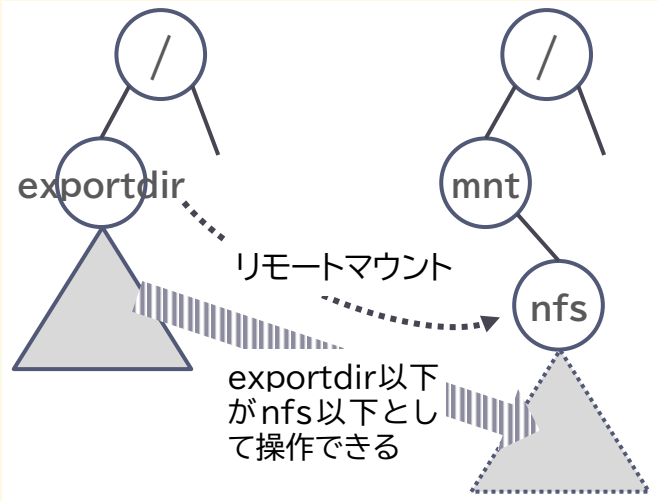
共有ファイルを使用するためには、他のファイルシステムと同様にmountコマンドでマウント（OSに認識させ、利用できるようにする操作）を行う



Linuxのシステム管理に関する知識（IPA）より
http://www.ipa.go.jp/files/000018480.pdf

NFSの場合、クライアントでは自身のルートからのディレクトリーファイルの木の中にマウント（リモートマウント）することで、クライアントにある他のファイルと同じようにアクセスできます。

/mnt/nfs に /exportdir 以下の部分木を接続するイメージです。



教科書との対応

- IPパケットの構造について、補足しています
- 遠隔手続き呼出し、分散オブジェクト技術については割愛します
- クライアントサーバモデルについては、次回に触れます

教科書での説明と記載箇所が違うところがありますので、読むときに注意してください。

第10回の課題

(1) 通信プロトコルに関する記述のうち、適切なものはどれか

ITパスポート平成27年秋期

ア　アナログ通信で用いられる通信プロトコルはない

イ　国際機関が制定したものだけであり、メーカが独自に定めたものは通信プロトコルとは呼ばない

ウ　通信プロトコルは正常時の動作手順だけが定義されている

エ　メーカやOSが異なる機器同士でも、同じ通信プロトコルを使えば互いに通信することができる

(2) ネットワークでメッセージをパケットに分割して送ることの必要性やメリットを述べよ

今回の課題です。クラスウェブのレポートで提出してください。

期限は、6/22 の午前中とします。

事後学習・事前学習

- 今回の講義資料に基づいて内容を振り返り、教科書などの該当箇所を読む
- 教科書第13章（13.1、13.2）に目を通す

今回の講義内容の振り返りと次回の準備をお願いします。