コンピュータシステムII（１１）

情報基盤センター
天野　浩文

1

前回のおさらい（1）

- **デッドロック**（deadlock）
  - 「すでに確保した資源がありながら，他のプロセス（スレッド）が先に確保してしまったために待たされている資源がある」という状態にある一群のプロセス（スレッド）が，いずれも先に進めなくなって膠着状態に陥ること．
- 4つの必要条件
  1. **相互排除**（mutual exclusion）
  2. **確保と待機**（hold and wait）
  3. **非プリエンプション**（no preemption）
  4. **巡回待機**（circular wait）

2

前回のおさらい（2）

- **デッドロックの抑止**（prevention）
  - デッドロックの必要条件4つのうちの**いずれかが成立しないように**，プロセス（スレッド）からの資源要求の順序・タイミングに一定の制約を課す．
  - 条件1：「相互排除」を取り除くのは一般には不可能なので，他の3つの条件のいずれかが成立しないようにする．
- **デッドロックの回避**（avoidance）
  - システム全体の資源の割り当て状況を監視しながら，デッドロックが発生する可能性のある資源割り当て要求が出されたら，それの処理を遅らせる．
  - 資源の割り当て状況やプロセス相互の待機状況を常に監視しておかなければならない．

3

前回のおさらい（3）

- 不幸にもデッドロックが起こってしまったら
  - **そのままにしておいても自然に解消されることは絶対にない**．
- **デッドロックからの回復**（recovery from a deadlock）
  - デッドロックに関与しているプロセス（スレッド）のいずれかを強制終了させる．
  - デッドロックに関与しているプロセス（スレッド）のいずれかを，まだ実行開始していなかったことにする．
    - やりかけた操作をすべて元の状態に戻して，最初から実行しなおす．

4

前回のおさらい（4）

- **デッドロックの検出**（detection）
  - デッドロックの回避やデッドロックからの回復を行うために必要.
  - **待ちグラフ**（wait-for**グラフ**）を用いる方法
  - **時間切れ**（timeout）方式
- ただし，並列処理・分散処理においては，単一プロセッサの場合よりも，より複雑で困難な問題になる.

5

前回のおさらい（5）

- **トランザクション**（Transaction）
  - 「一つの意味的にまとまった仕事」
  - 途中まで実行されて中断・中止された状態では，意味のある（正しい）結果を残せない，ということ.
  - データベース（database, DB）の分野で使われることが多く，「ひとまとまりの処理を構成するデータベース操作の集まり」などと定義されることもある.
- トランザクションの処理は，以下の4つの性質を満たさなければならない.
  - ACID**特性**（ACID property）
    - **原子性**（Atomicity）
    - **一貫性**（Consistency）
    - **隔離性**（Isolation）
    - **耐久性**（Durability）

6

前回のおさらい（6）

- 隔離性に関するより厳密な定義
  - 複数のトランザクションを並行処理（同時実行）した場合でも，その結果は
    - データベースの，可能なすべての初期状態に対して，
    - アクセスされるすべてのデータ項目について，
    - それらのトランザクションを何らかの順序で逐次処理した場合の結果（**変化の履歴を含む**）と一致しなければならない.
  - 特定の初期状態のときだけ等価な直列スケジュールが存在する，というのではダメ.
  - データ項目Aの上で観測されるトランザクション実行順とデータ項目Bの上で観測されるトランザクション実行順が異なるようなことも，あってはならない.

7

前回のおさらい（7）

- いくつものデータ項目にアクセスするトランザクションがデータベースへの書き込みをその都度行っていると...
  - 途中で止まったときに，すでに書き込まれた結果を元に戻すのは大変.
- そこで，通常は，以下のような方法が採られる.
  - トランザクション実行中の書き込みは，当該データ項目の作業用コピーに対して行う. 元のデータベースにはただちに反映させない.
  - トランザクションが正常終了できるとわかったときに，初めて，すべての書き込み結果を，元のデータベースに反映させる
    ⇒**コミット**（commit）操作
  - トランザクションが正常に終了できないとわかったときは，当該トランザクションを強制終了し，作業用コピーを捨てる.
    （実行前の状態に戻ることができる）
    ⇒**アボート**（abort）操作
- 広域分散システム上では…
  - コミットすべきか，アボートすべきかの判断が困難
  - コミット・アボートが正常に終了することを保証することも困難

8

前回のおさらい（8）

- **二相コミット（Two-Phase Commit）プロトコル**の原理
  - 仮定
    - 各演算ノードごとに，そこのデータを管理する「リソースマネージャ」が1つ存在する．
    - 個々のデータ項目の監視まではできないがリソースマネージャの「投票」の結果を管理できる「コーディネータ」が，システム全体で1つ存在する．
  - **1つの分散トランザクションの作業用コピーの上での処理が終了した後のコミット操作を，次の2つの段階に分割する．**
    - 第一段階
      - 各演算ノードでコミットの準備ができたかどうかを確認する．
    - 第二段階
      - 各演算ノードのリソースマネージャからの「投票」を受けて，コミット／アボートの判断を行い，各ノードでコミットを行う．

9

クライアントサーバシステム

10

クライアントサーバモデル（1）

- サーバ（server）と，クライアント（client）によって，システムを構成する方式
  - サーバ：あるサービスを提供するもの
  - クライアント：そのサービスを受けるもの
- 主な用途
  - ネットワーク上で動作するアプリケーションや，広域分散処理システム上のサービスに多く見られる．
    - クライアントがサーバにサービスの実行を要求（処理を要求）し，
    - サーバがそれに応答する（実行結果を返す）．
  - 通常は，ひとつのサーバがネットワーク上に分散する複数の（多数の）クライアントにサービスを提供することが多い．
    - ただし，単一の計算機の内部でサーバプロセスとクライアントプロセスが動作する形態もある．

11

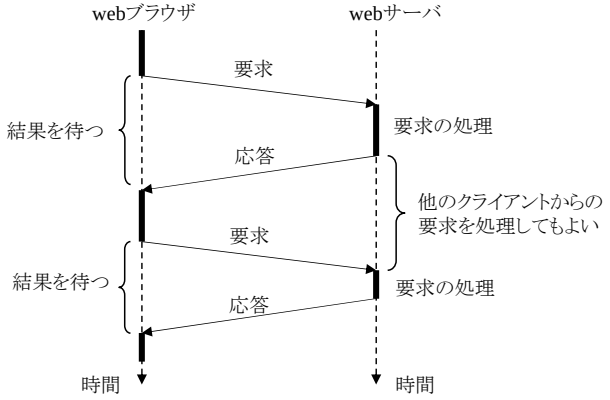
クライアントサーバモデル（2）

- サーバとクライアントが通信するための規約（プロトコル，protocol）が，**サービスごとに定められている．**
  - この規約に従うことにより，異なる製造元から提供されたサーバとクライアントが通信することも可能になる．

12

クライアントとサーバの通信の例（1）

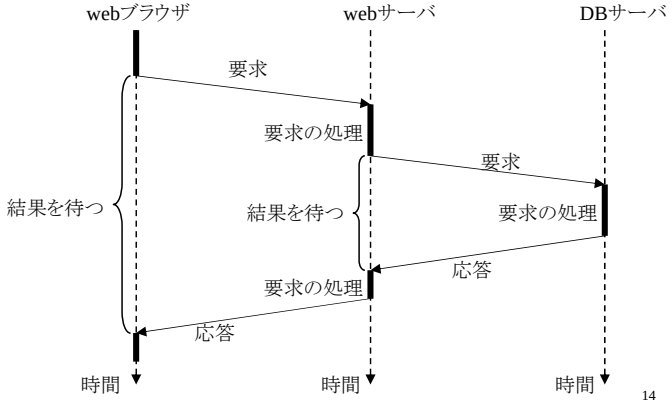
- 要求と応答



13

クライアントとサーバの通信の例（2）

- サーバは，別のサービスのクライアントにもなりうる．



14

クライアントサーバシステム

- クライアントサーバモデルに基づいてサービスを構築したシステム
- クライアントサーバシステムによるサービスの例
  - DNS（Domain Name System）
  - 電子メール
  - FTP（File Transfer Protocol）
  - World-Wide Web（WWW）
  - ソケット（socket）

15

DNS

- インターネット上のホスト名とIPアドレスを対応付けるサービス
  - 例：「isabelle.cc.kyushu-u.ac.jpのIPアドレスは？」
    ⇒「133.5.7.33」
    - IPアドレス：インターネット上の計算機・通信機器に付与された番号のようなもの
- サーバ：DNSサーバ
  - 通常は，個人が所属する組織（大学等）または利用しているインターネットサービスプロバイダが提供する．
- 主なクライアント
  - リゾルバ（resolver）
    - 例：OSの中に用意された手続き gethostbyname
    - このサービスを必要とする他のサービスから呼び出されることが多く，これ自体が単体で使用されることは少ない．

16

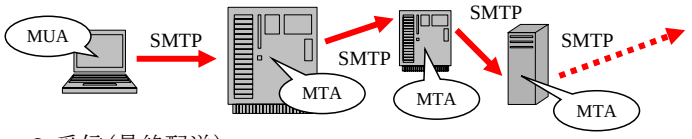
電子メール

- 最も古いネットワークアプリケーションの1つ
- いくつものプロトコルを組み合わせて運用されている.
  - メッセージ転送プロトコルの例
    - SMTP（Simple Mail Transfer Protocol）
    - ESMTP（Extended SMTP）
  - 最終配送プロトコルの例
    - POP3（Post Office Protocol Version 3）
    - IMAP（Internet Message Access Protocol）
  - アルファベットだけでは表現できないデータをメッセージに埋め込むためのプロトコル
    - MIME（Multipurpose Internet Mail Extensions）
- 他のサービスと組み合わせて提供されることもある.
  - Webメール
    - 例：GraceMail

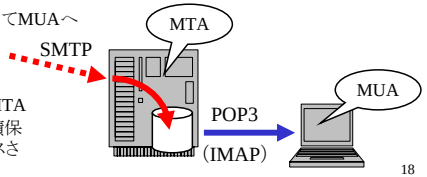
17

電子メールの運用形態の例（1）

- 送信（転送）
  - MUA（Mail User Agent）と呼ばれるクライアントプログラムからサーバ（SMTPサーバ）への転送
  - MTA（Mail Transfer Agent）間の転送もSMTPで



- 受信（最終配送）
  - POP3またはIMAPによってMUAへ

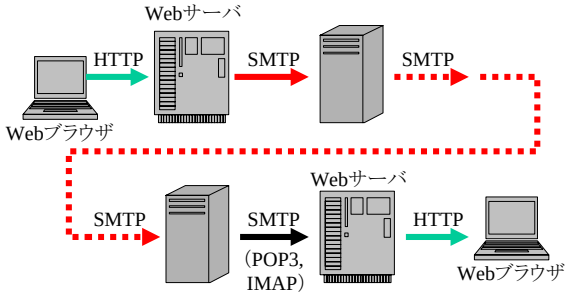


メール配送経路の最終端のMTAは，ユーザごとのメールを蓄積保存していて，MUAからアクセスされるのを待っている.

18

電子メールの運用形態の例（2）

- Webと組み合わせたサービス

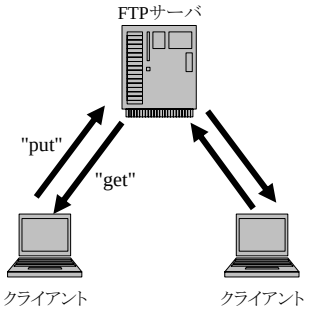


実際には，標準のWebサーバにはこの機能が備わっていないので，新たなプログラムを追加する必要がある.（例：GraceMail）

19

FTPサービス

- インターネット上の計算機間でファイルを転送するサービス
  - サーバ：FTPサーバ
  - クライアントの例：
    - OSに備わっている ftp コマンド
- プロトコルの名前がFTP
- ほぼ同格の計算機間の転送にも，多数のクライアントが共通の「ファイル置き場」を利用するようなサービスにも用いられる.
  - クライアントから見て，ファイルを送るのが"put"，受け取るのが"get"



20

WWW

- 誕生は1989年...
  - ヨーロッパの原子核研究機構で，報告書・論文・画像・観測データなどのさまざまなデータファイルを多くの研究者がネットワーク共有するためのサービスとして始まった．
  - その後の爆発的な普及と，広汎な機能拡張
- インターネット上の数百万ものサーバ上の文書をリンクで結んだ「ハイパーテキスト（hypertext）」を共有するしくみ
  - ハイパーテキストのコンセプトは元からあった．
    - Webサービスのためのハイパーテキストの文法がHTML（hypertext markup language）
  - クライアントがハイパーテキストの転送を要求したり，サーバがクライアントにデータを転送するためのプロトコルがHTTP（hypertext transfer protocol）

21

現在のWebサービス

- HTML＋HTTPだけではなくなった．
  - 今やHTMLはページの外枠だけ，などということも
  - さまざまなコンテンツタイプのサポート
    - 音楽，動画，etc.
  - サーバ側による動的Webページ生成
    - CGI（Common Gateway Interface）
    - PHP（PHP: Hypertext Preprocessor）
    - JSP（Java Sever Pages），ASP（Active Server Pages）
  - クライアントによる動的Webページ生成
    - JavaScript
      - プログラミング言語 Java とは別物
    - Java applet
      - プログラミング言語 Java のサブセット
    - Active-X
      - 事実上，プログラムにできることなら何でも実行可能

実際には，ページ生成以上のことも実行可能．

実際には，ページ生成以上のことも実行可能．

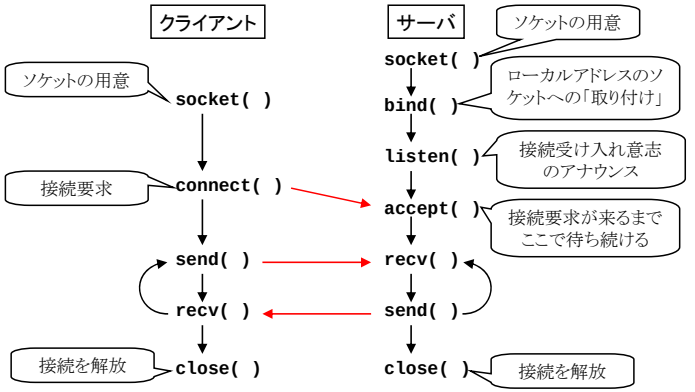
22

ソケット

- プロセスとプロセスが通信するための最も基礎的なサービス
  - バークレイ・ソケット（Berkeley socket）と呼ばれることがある．
    - Berkeley UNIXで最初に採り入れられたため．
    - 現在は，他の多くのOSでも提供されている．
  - 他の多くのネットワークサービスが，これを用いて実現されている．
- ほぼ対等な2つのプロセスのうち，一方がサーバ，他方がクライアントとなる．
  - サーバは，「ソケット」と呼ばれる「通信口」を用意し，そのソケットを自分が動作している計算機のIPアドレスに対応付けて，クライアントからの接続要求を待つ．
    - 自分のIPアドレスを取得するためにリゾルバを利用
  - クライアントは，自分の側のソケットを用意した後，サーバが用意しているはずのソケットへの接続を要求する．
    - 相手のIPアドレスを取得するためにリゾルバを利用
  - 接続確立後は，サーバとクライアントが対称的に send/recvを行うことができる．

23

典型的なソケット通信の流れ



24

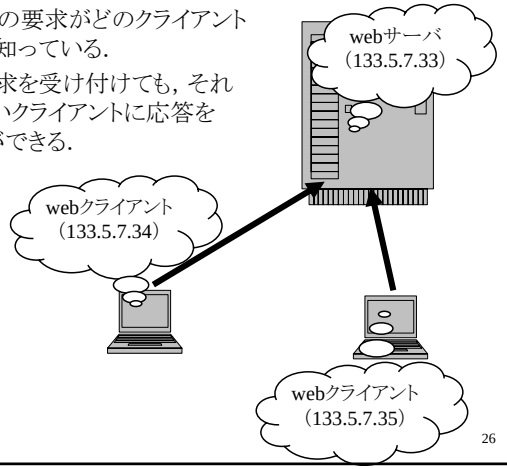
サーバ側のソケット

- サーバは，1つのソケットに対して複数の接続要求を受け入れるように構成することもできる．
  - SMTPサーバやWebサーバはそのような接続要求を受け入れるシステムの例

25

単一ソケットに対する複数の接続要求

- サーバは，どの要求がどのクライアントから来たか，知っている．
  - 複数の要求を受け付けても，それぞれ正しいクライアントに応答を返すことができる．



26

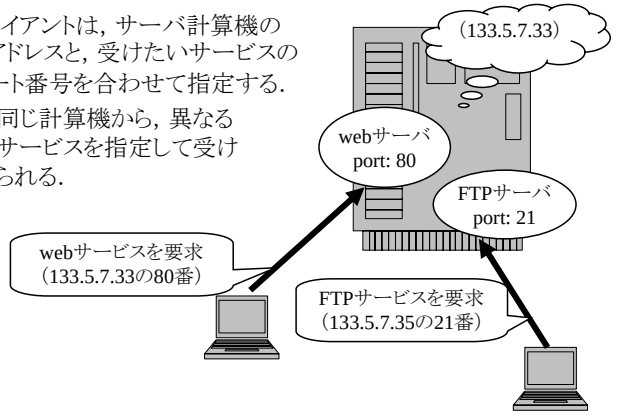
ポート

- 1つの計算機が，いずれもソケットを用いて実現された複数のサービス（ソフトウェアとしてのサーバ）を動かしている場合，クライアントは，どうやって自分が必要とするサービスを受けられるか？
  - **ポート**（port）
    - ソケットを作成するときに，1～65535（16ビット）の番号を指定する．
    - クライアントは目的のサーバが指定したのと同じポート番号を指定すれば，そのサーバに接続できる．
- ソケットとは，**IPアドレスとポート番号を組み合わせたもの**

27

単一の計算機が複数のサービスを提供するとき

- クライアントは，サーバ計算機のIPアドレスと，受けたいサービスのポート番号を合わせて指定する．
  - 同じ計算機から，異なるサービスを指定して受けられる．



28

予約されているポート

- 1024未満のポート番号は，よく知られているネットワークサービス用で使用されている．
  - 予約されている番号の例
    - 21：FTP
    - 25：SMTP
    - 80：HTTP
    - 110：POP3
  - 自分で新たなサービス用のプログラムを構築する場合には，1024以上の番号を使用する．
    - ただし，1024～49151も，ユーザの利便性のため登録制になっている．
    - プライベートなネットワークのクライアントに一時的なポートを割り当てる場合などは，49152～65535を使用すべき．

29

inetdデーモン

- OSに標準的に備わっているネットワークサービスは，クライアントからの接続を**常時待ち続ける**．
  - 無限ループの形をしたプログラムが，プロセスとして動き続ける．
    - このようなプロセスを**デーモン**（daemon）という．
    - しかし，利用頻度の低いものでも，デーモンとして動作させ続けると，資源（メモリ空間，CPU時間）を消費する．
- inetdデーモン
  - 常時動作させるほど利用頻度の高くないサービスを，要求があるたびに新たなプロセスとして起動するためのデーモン
  - inetdデーモンは，あらかじめ設定されたポートへの接続要求を常時監視しており，要求がきたら，該当するサーバをプロセスとして起動することができる．

30

ピアツーピアシステム

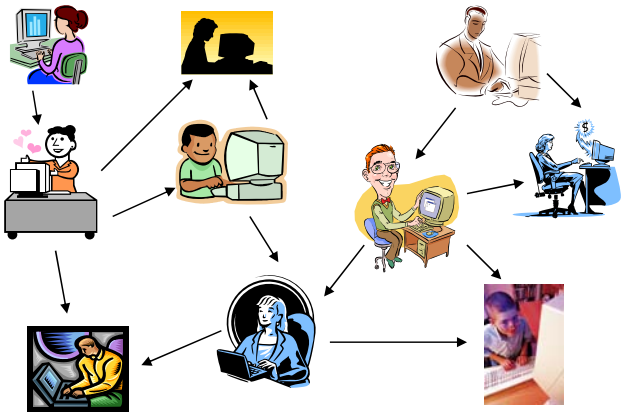
31

ピアツーピア

- クライアントサーバシステムの特徴
  - 非対称
    - 固定されたサーバと，（不特定多数の）クライアント
      - サーバは「ずっとそこにいる」が，クライアントは「必要なときだけ出てくる」だけでよい．
    - 1990年代くらいまでは，ネットワークサービスを構築する方法はこれだけで十分かとも思われていたのだが...
- **ピアツーピア**（peer-to-peer，P2P）
  - ネットワークサービスを構築するもうひとつの方法
    - どの端末も対等な立場で参加
    - サービスを受けたり，提供したり
    - PCの性能向上が背景に
  - 1999年ごろから，Napsterと呼ばれる音楽データ交換サービスで注目を浴びるようになった．

32

P2Pシステムに固定的なクライアント・サーバはない



33

P2Pは紛争のタネ？

- 決して「P2P＝音楽ファイル共有」ではないのだが…
  - P2Pがこれで有名になってしまった．
  - データ共有以外に，今のところあまり有望な応用が見つかっていない．
    - インターネット電話，インスタントメッセンジャーなどをP2Pの応用先と考える人もいるようだが…普及度は？
  - データがコピーできるということになると，「カネを払わずに有料のコンテンツを」と考える人間が出るのは世の常で…

34

歴史的経緯

- 第一世代のP2Pアプリケーション（Napsterなど）
  - 「会員」は，自分のハードディスクにあるデータファイルを，中央のサーバに登録する．
    - どこかの会員が「欲しい」と思ったデータが中央のデータベースで見つかれば，その会員のハードディスクから直接ダウンロードすることができる．
  - 「音楽ファイル」の交換に使われてしまった．
    - 裁判所からNapster社の運営停止命令が出る騒ぎに
- 第二世代（Gnutellaなど）
  - 中央のデータベースが不要に
  - すべて端末同士のやりとりだけで
  - 著作権保有者は誰を相手に裁判を？
    - 当然，誰か次に簡単に突き止められる相手をやり玉に
    - 「犯人」の所属する大学・企業，「犯人」が使用したインターネットサービスプロバイダなど

35

法的論争

- 家庭用VCRのメーカがテレビ局の著作権を侵害していることにはならない（米国で判例あり）．
  - 留守番録画して自分が楽しむのが普通であり，不特定多数の人に配ったりは簡単にはできないから
- 「P2Pソフトウェアも VCRと同じで，使用者のモラルの問題だ」という主張は，どうも通らないようだ．
  - 不特定多数の会員に公開されてしまう．
  - いったん登録されアクセス可能になったデータの拡散のスピードが異常に速い．
- 「自分は著作権フリーのデータしか公開していない」という主張
  - 一部のP2Pソフトは，「こっそり」ユーザのハードディスクを中継データの一時的な格納場所に使用する．
  - 知らないうちに不法行為の片棒をかつぐことに

36

組織の自己防衛

- 音楽ファイルあるいはビデオファイルの交換は，非常に大きな通信負荷を伴う．
  - 組織（大学・企業等）を外部のネットワークにつなぐ回線の通信速度は有限
  - 接続サービスを提供するインターネットサービスプロバイダにとっても脅威となる．
  - これを理由にP2Pソフトウェアの利用を禁止する動き
- 「匿名性」をうたうP2Pソフトウェアであっても...
  - 著作権保護団体が同じソフトウェアを利用すれば，違法なファイルがどの組織から流れているかは検知可能
  - 不作為を問われる訴訟の恐れ
    - 2003年11月には，日本レコード協会が全国約1200の国公私立大学・短大に対し、学内ネットワークを使用した音楽の不正利用を防止するよう要請する文書を送付
    - 「知らなかった」では済まない状況に

37

九州大学では

- WinMX，Winny のようなP2P型ファイル共有（交換）ソフトウェアの学内LANでの利用を禁止
  - 特定の（固定された）メンバの間での合法的なファイル共有なら，他にも方法があること．
  - 知らない間に不法なファイル交換の共犯になるのを防ぐ方法が他にないこと．
  - 特定のポートをファイヤウォールで封鎖するという方法では遮断できないこと．

38

まとめ

- **クライアントサーバシステム**
  - サーバ（あるサービスを提供するもの）と，クライアント（そのサービスの提供を受けるものから構成される．
  - サービスごとに，サーバとクライアントの間の通信規約が定められている．
    - 例：DNS，電子メール，FTP，WWW，ソケットなど
- **ピアツーピアシステム**
  - 固定的なサーバ・クライアントが存在しない．
    - サービスを提供したり，受けたり
  - 残念ながら，今のところ，ほとんど『悪者』扱い

39

次回（来年1月10日）

- 演習の第2回をやります．
  - 持ち込みその他の条件は，前回と同じです．
  - 出題範囲は，第9回～第11回（今回）の講義で扱った内容とします．

40