分散システム 第4,5回
― 通信 ―

大連理工大学・立命館大学 国際情報ソフトウェア学部

大森 隆行

講義内容

- 通信
  - プロトコル
  - RPC (遠隔手続き呼び出し)
  - メッセージ型一時通信
    - ソケット
  - メッセージ型永続通信
    - メッセージキューイングシステム
  - ストリーム型通信
  - マルチキャスト通信

通信プロトコル

- 通信には標準化されたプロトコルが必要
  - 異なるプロトコルで通信すると…？
    - e.g., 異なる文字コード、各社固有のプロトコル
- プロトコルは様々なレベルで必要
  - e.g., ビット列の伝送を行うための電気的規格、電子メールをやり取りするときの手順

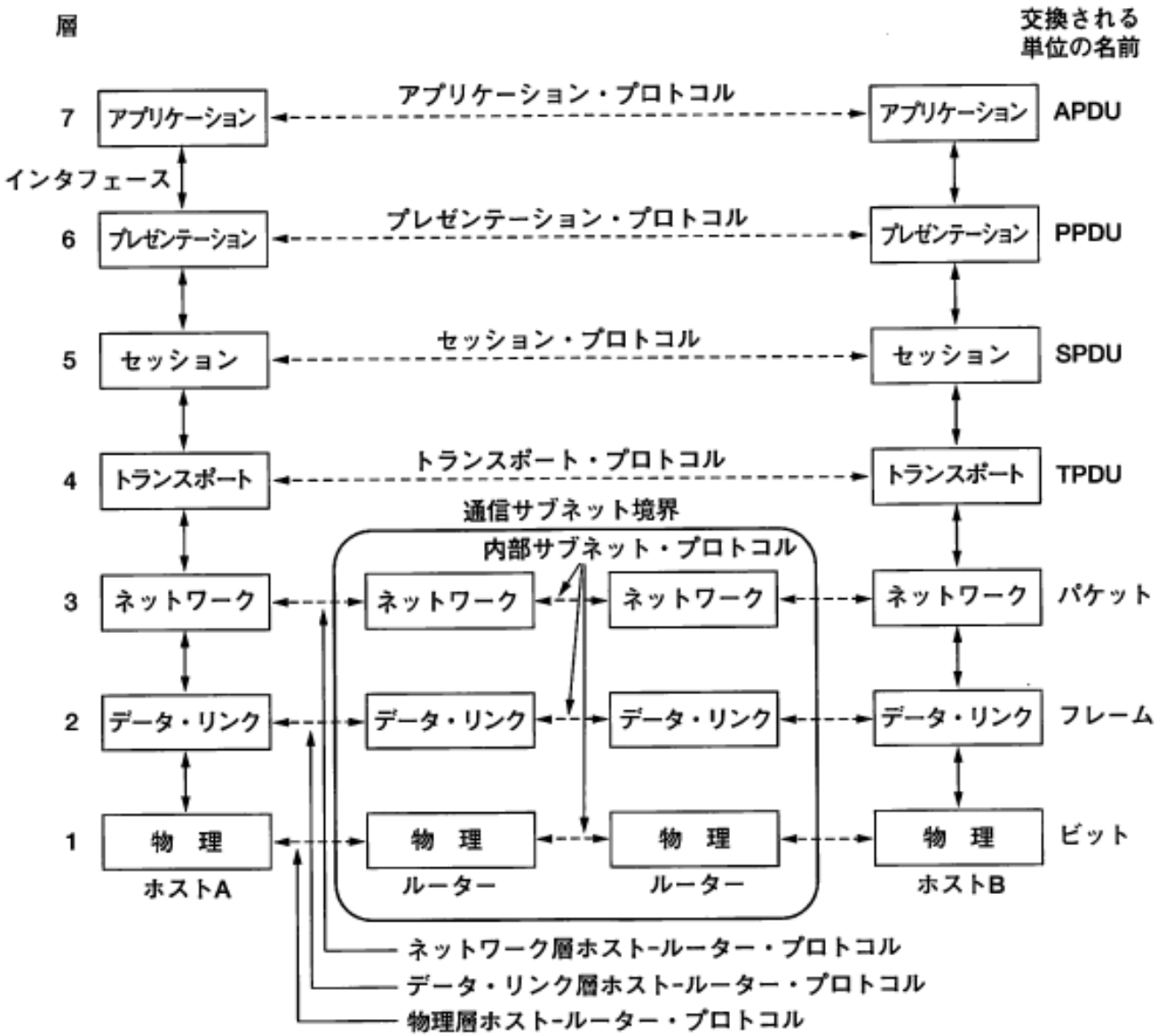
→階層化プロトコルが必要

OSI基本参照モデル

- ISO (International Organization for Standardization) により策定
- 英語では、
  OSI (Open Systems Intercommunication) reference model

- 通信プロトコルを7つの層に分けて定義

OSI基本参照モデル



アプリケーション層

プレゼンテーション層

セッション層

トランスポート層

ネットワーク層

データリンク層

物理層

OSI基本参照モデル

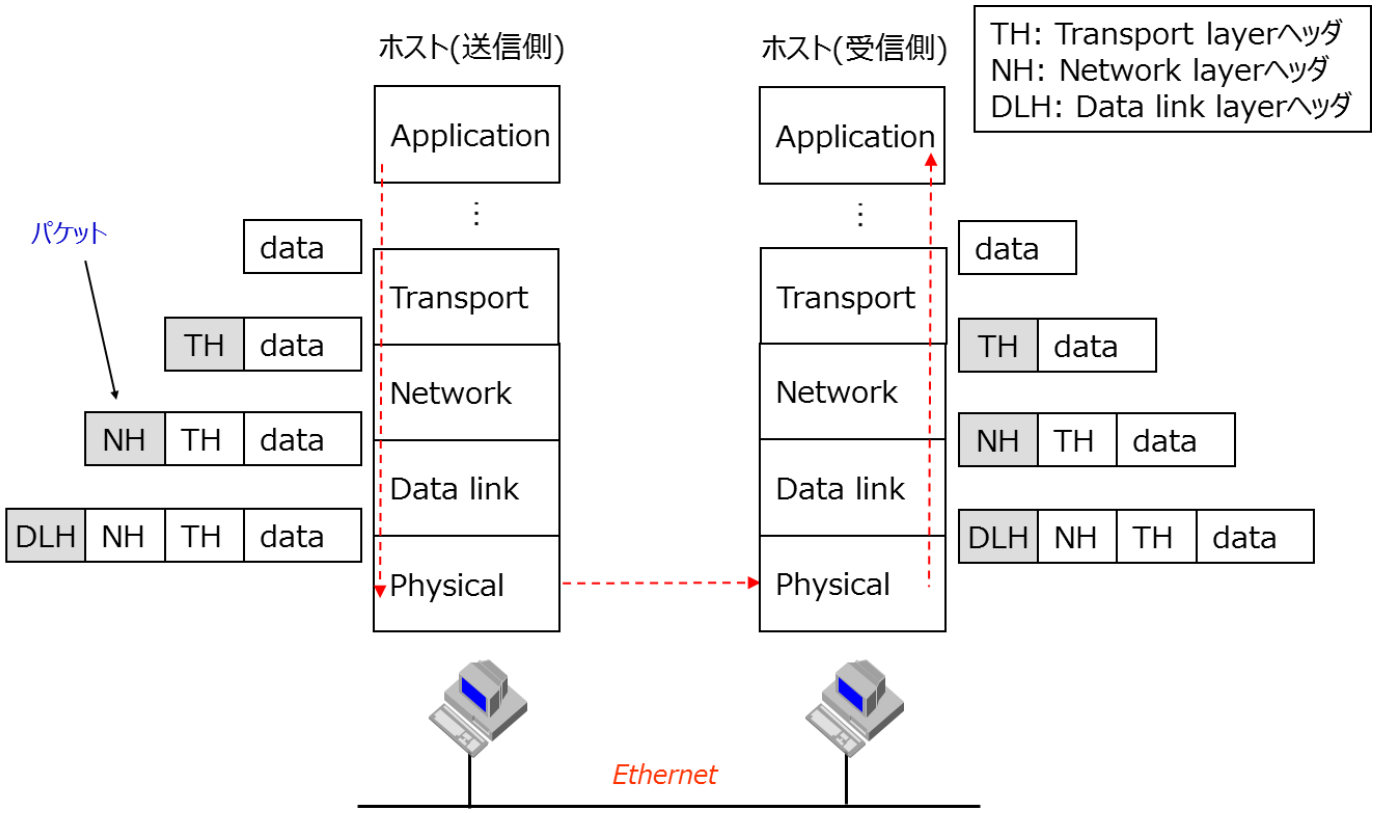
- アプリケーション層
  - アプリケーション固有の規格
- プレゼンテーション層
  - アプリケーションで扱うデータの表現形式
    (文字コード、圧縮方式、暗号化方式等)
- セッション層
  - データ転送のための同期や通信方式に関する規定
- トランスポート層
  - 信頼性のあるデータ転送を提供
  - エラー検出・回復機能等

OSI基本参照モデル

- ネットワーク層
  - 送信元から送信先へのパケット送信
  - ネットワークアドレス制御等
- データリンク層
  - データフレームの送受信、フロー制御、伝送エラー制御等
- 物理層
  - 通信チャネル上で生のビットを伝送
  - ネットワークに対する機械的、電気的、時間的インタフェースを定義

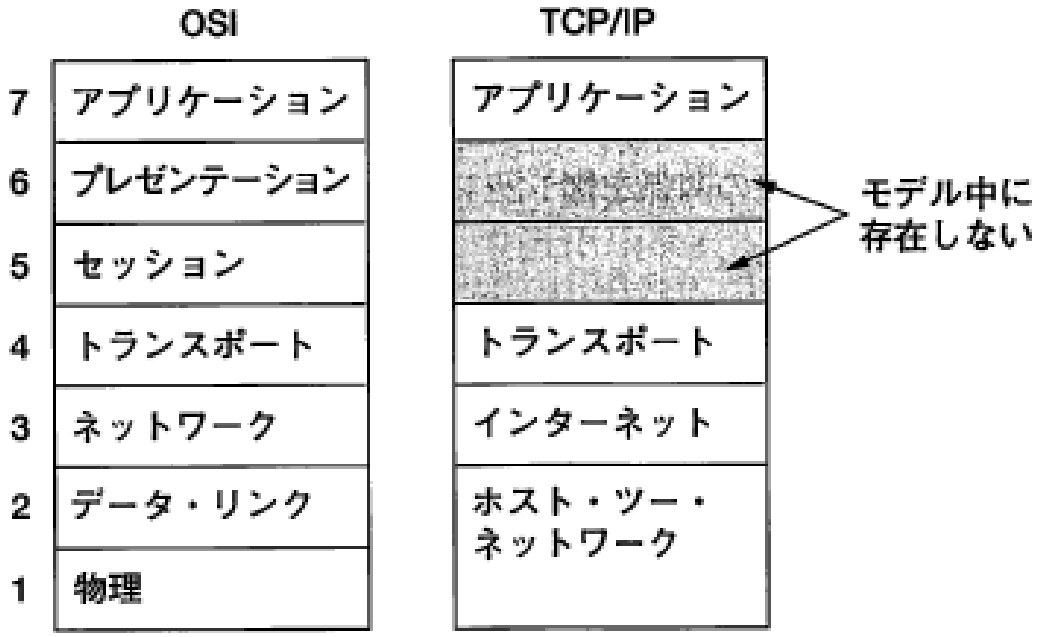
OSI基本参照モデル

- 送信時は、各階層ごとにヘッダを付与
- 受信時は、各階層ごとにヘッダを除去
- 下位層にとって、上位層からのデータは
  メッセージと区別が付かない



インターネットプロトコルスイート

- インターネットプロトコルスイート (Internet protocol suite) (TCP/IPプロトコルスイートとも呼ぶ)
  - ARPANET (Advanced Research Projects Agency Network) で使用
  - 現在主流となっている



プロトコルの種類

一般的なプロトコルの区別として、以下の2つがある

- コネクション型
  - データ交換前に明示的にコネクションを確立
  - 同期的な通信が可能
  - (例) TCP
- コネクションレス型
  - コネクション確立は不要
  - 送信者の準備ができれば送信
  - 非同期通信に使用
  - (例) UDP

確認問題

- OSI基本参照モデルでは、通信プロトコルをいくつの層に分けて定義しているか。
- OSI基本参照モデルの各層の名称を答えよ。
- 以下はコネクション型のプロトコルによる通信の特徴を説明したものである。空欄を埋めよ。
  - データ交換前に明示的に(　1　)を確立する。
  - 同期的な通信が(2 可能・不可能)。

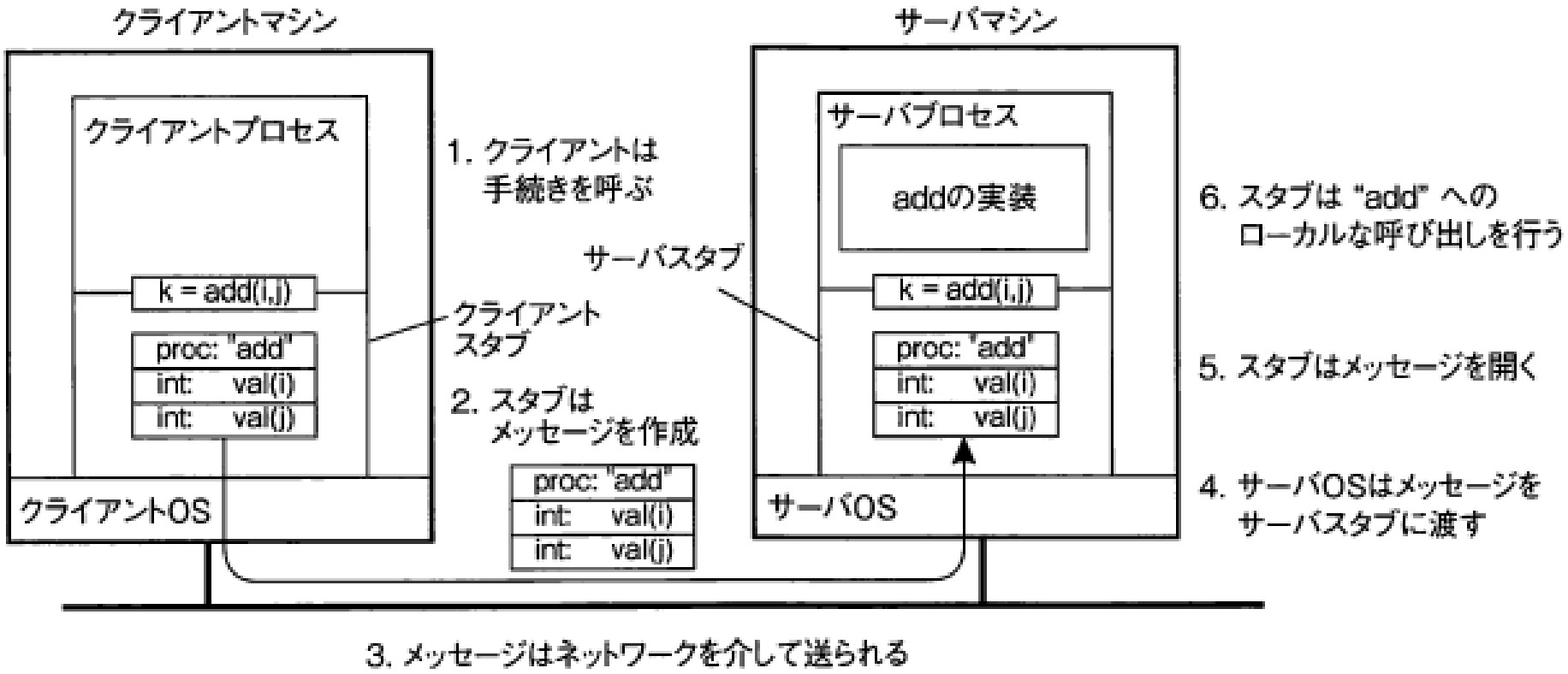
講義内容

- 通信
  - プロトコル
  - RPC (遠隔手続き呼び出し)
  - メッセージ型一時通信
    - ソケット
  - メッセージ型永続通信
    - メッセージキューイングシステム
  - ストリーム型通信
  - マルチキャスト通信

遠隔手続き呼び出し(RPC)

- RPC: Remote Procedure Call
  - ネットワークを介して他のマシンの手続きを呼び出す仕組み
  - プログラマの立場から見るとメッセージ交換が隠蔽される
    - 通常の関数呼び出しと同じように見える

RPCによる遠隔通信

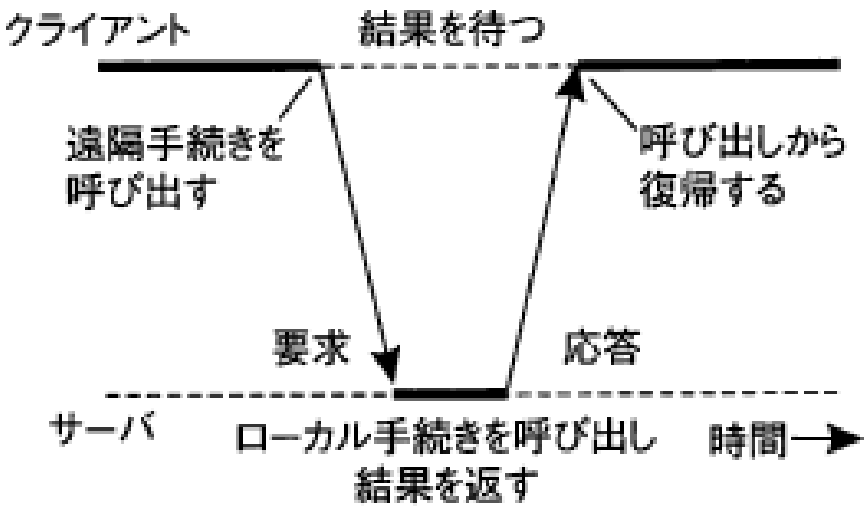


- スタブ：上位モジュールに呼ばれるモジュールのこと
(RPCでは通信を隠蔽する役割を担う)
- パラメータマーシャリング(parameter marshaling)：
パラメータを1つのメッセージに包み込むこと
  - 引数の解釈方法はサーバ、クライアントで同じでなければならない

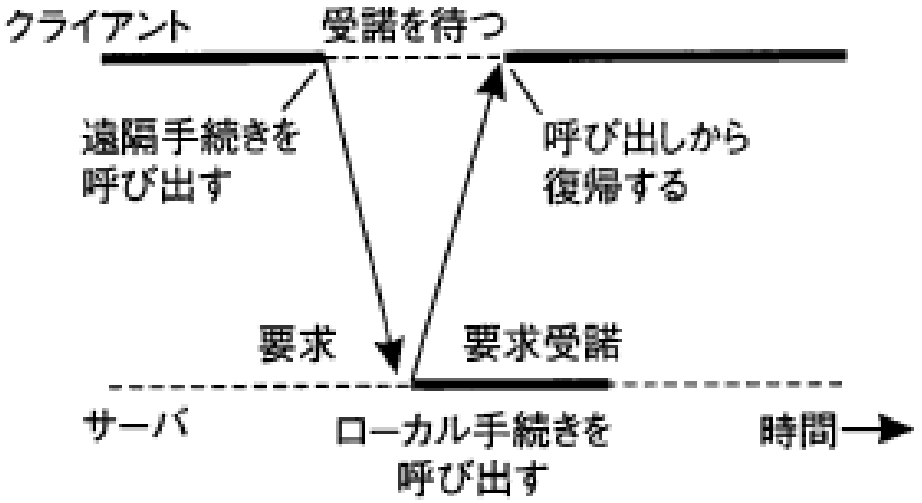
RPCによる遠隔通信

- ポインタの扱い
  - メモリ空間を直接指し示すポインタはマシン間で共有できない
    → ポインタの先のデータをコピーする等の対策が必要
- インタフェース定義言語
  (IDL: Interface Definition Language)
  - アプリケーションからRPCを利用する際のインタフェースを定義
    - 呼び出し可能な関数やマーシャリングの方法等

非同期型RPC



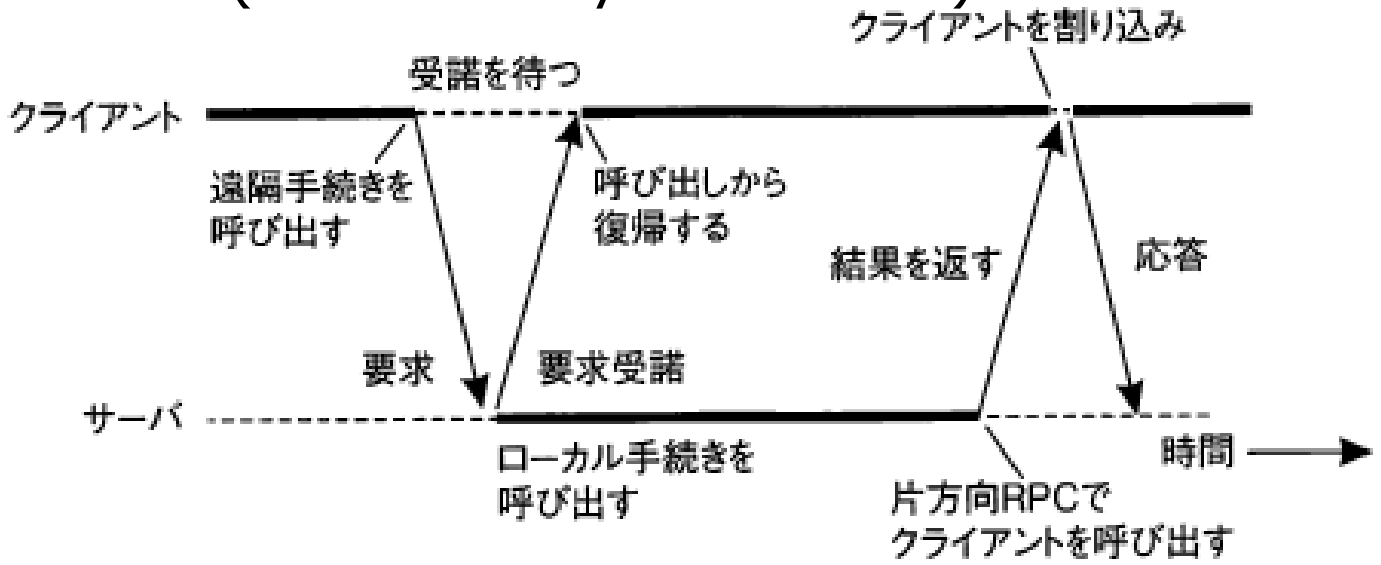
同期型RPC
(Synchronous RPC)



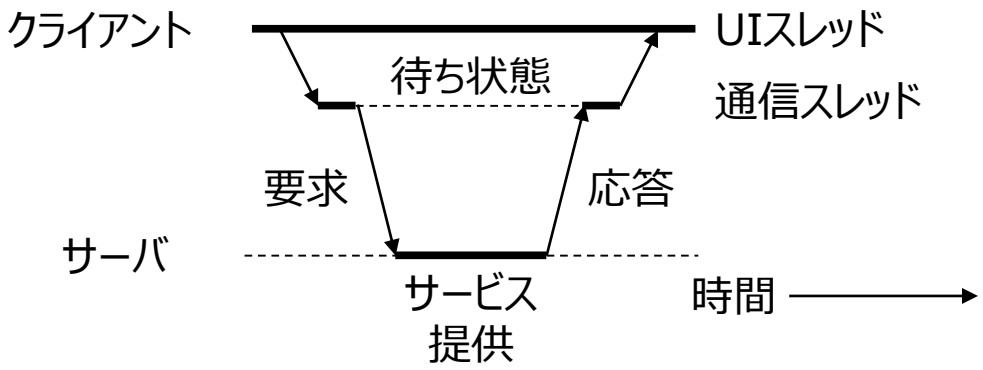
非同期型RPC
(Asynchronous RPC)

非同期型RPC

- 保留同期型 (Deferred Synchronous) RPC



- (参考) Ajax (Asynchronous Javascript and XML)



確認問題

- RPCによる遠隔通信の手順を説明したものである。空欄を埋めよ。

1. クライアントが手続きを呼ぶ。
2. クライアントの(　1　)がサーバに送るメッセージを生成する。この際、パラメータは(　2　)により、1つのメッセージに包み込まれる。
3. メッセージはネットワークを介してサーバに送られる。
4. サーバOSはメッセージをサーバの(　1　)に渡す。
5. サーバの(　1　)がメッセージを開く。
6. サーバ側で手続きのローカル呼び出しが行われる。

- ()内から正しいものを選べ
  - (同期型・非同期型)RPCでは、クライアントがサーバ側での処理の完了を待たずに次の処理へ移行できる。
- 空欄に当てはまる語句を答えよ。
  - クライアントはサーバ側での処理完了を待たずに次の処理を行うが、後からクライアント側に結果を渡すためにクライアントに割り込みをかける方式を(　1　)型RPCと呼ぶ。

講義内容

- 通信
  - プロトコル
  - RPC (遠隔手続き呼び出し)
  - メッセージ型一時通信
    - ソケット
  - メッセージ型永続通信
    - メッセージキューイングシステム
  - ストリーム型通信
  - マルチキャスト通信

メッセージ型一時通信

- Berkeleyソケット
  - トランスポート層のインタフェースを標準化
  - 1970年代 Berkeley UNIXで導入
  - ネットワークを介するプロセス間通信を抽象化
- ソケット
  - 下位のネットワークを通じて送信されるデータへの書き込み、受信されるデータの読み込みができる通信のエンドポイント(endpoint)

ソケット通信

- ソケットに関する主なシステムコール

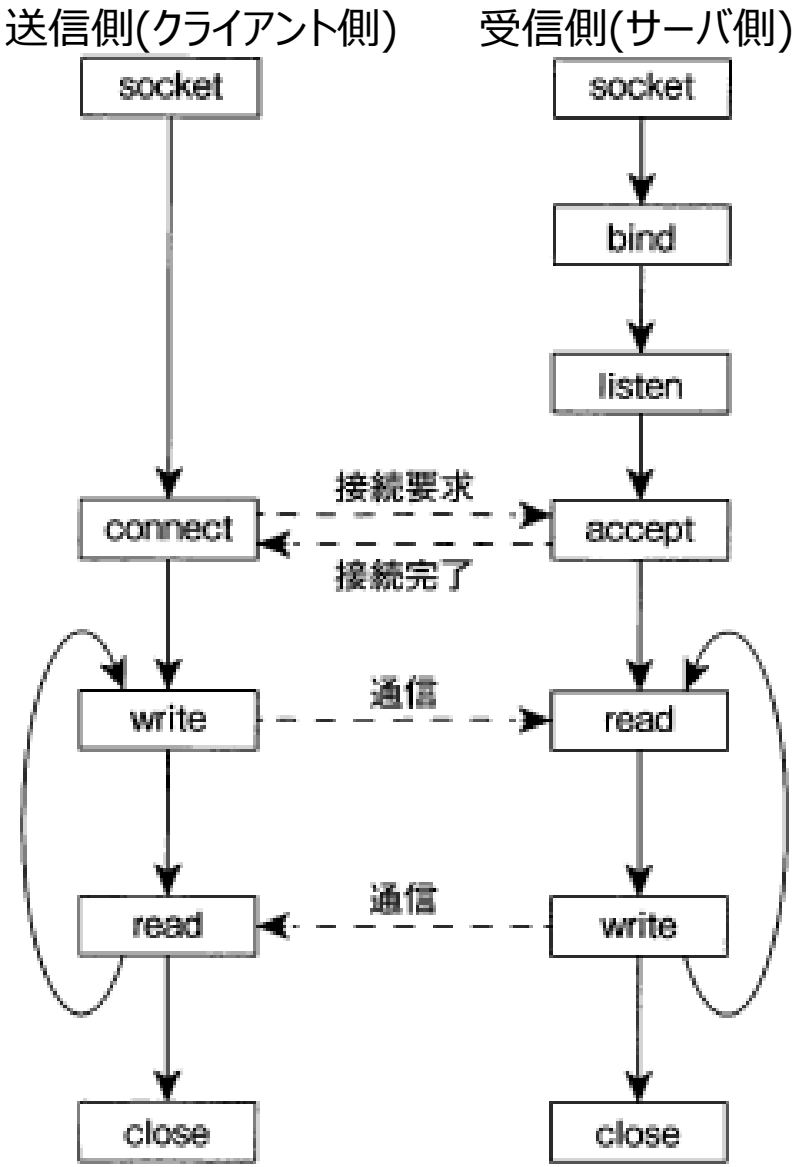
socket	新しい通信のためのソケットを生成
bind	ソケットにローカルアドレスを与える
listen	ソケットへの接続を受け付け可能を表明
accept	コネクションへの接続を待つ
connect	送信側から接続要求
close	ソケットを閉じる
write	ソケットへの書き込み
send	ソケットへの書き込み
read	ソケットから読み込み
recv	ソケットから読み込み

UNIXで用意されているシステムコールの使用例

クライアント側のコード例

```
#include <sys/socket.h>
…
int main() {
  struct sockaddr_in server_addr;
  char buf[BUFSIZE];
  int buf_len;

  /* ソケットの生成 */
  if ((sockfd = socket(PF_INET, SOCK_STREAM, 0)) < 0} { .. }

  … /* server_addrにサーバのアドレス・ポートを設定 */

  /* コネクションの確立 */
  if (connect(sockfd, (struct sockaddr *)&server_addr,
              sizeof(server_addr)) > 0) { .. }

  write(sockfd, buf, buf_len); /* メッセージの送信 */
  read(sockfd, buf, BUFSIZE); /* メッセージの受信 */
  close(sockfd); /* ソケットを閉じる */
}
```

ソケットを用いたプロセス間通信

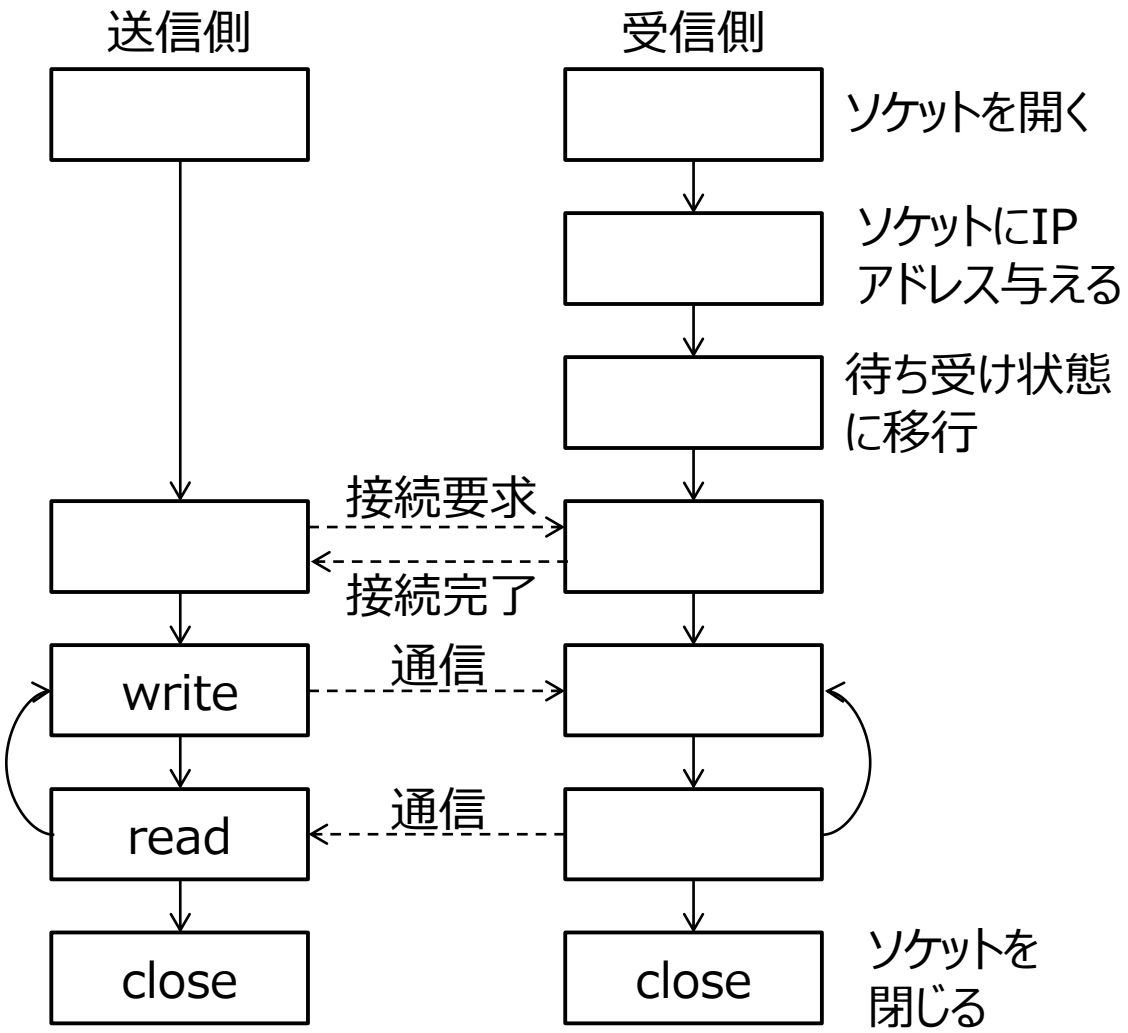


- socket
  - 通信のためのエンドポイント生成
  - 使用するプロトコルファミリや通信方式を指定
- listen
  - コネクションの受付準備 (passive mode)
- accept, connect
  - 受信側プロセスはブロックされる
  - 接続要求が来るとコネクションを確立
  - 送信側と接続されたエンドポイントを持つ新たなソケットを生成
  - 並行サーバの場合はforkで子プロセスを生成

確認問題

- 右図はソケット通信の過程を図示したものである。
空欄にあてはまる適切なものを選択肢から選んで答えよ。

選択肢：
accept, bind,
connect, listen,
socket,
read, write



講義内容

- 通信
  - プロトコル
  - RPC (遠隔手続き呼び出し)
  - メッセージ型一時通信
    - ソケット
  - メッセージ型永続通信
    - メッセージキューイングシステム
  - ストリーム型通信
  - マルチキャスト通信

通信の種類

- 一時通信 (transient communication)
  - 受信側はメッセージ投入時に実行されていなければならない
  - (例) 電話
  - (例) RPC、ソケット通信
- 永続通信 (persistent communication)
  - 受信側はメッセージ投入時に実行されている必要がない
  - メッセージは”ある場所”に蓄積される
  - (例) Eメール

メッセージキューイングシステム
(message-queueing system)

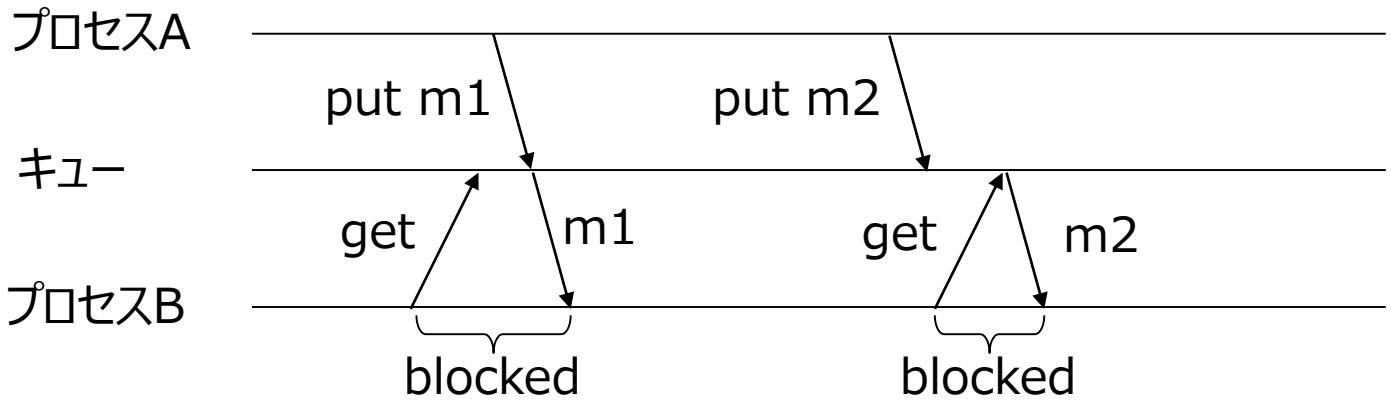
- メッセージを一時的にキューに蓄積
- いつメッセージが送信先に利用されるかは送信者にはわからない
- キューをバッファとして利用するだけでなく、配達保証や配達回数保証を提供することも

メッセージキューイングシステムの基本インタフェース

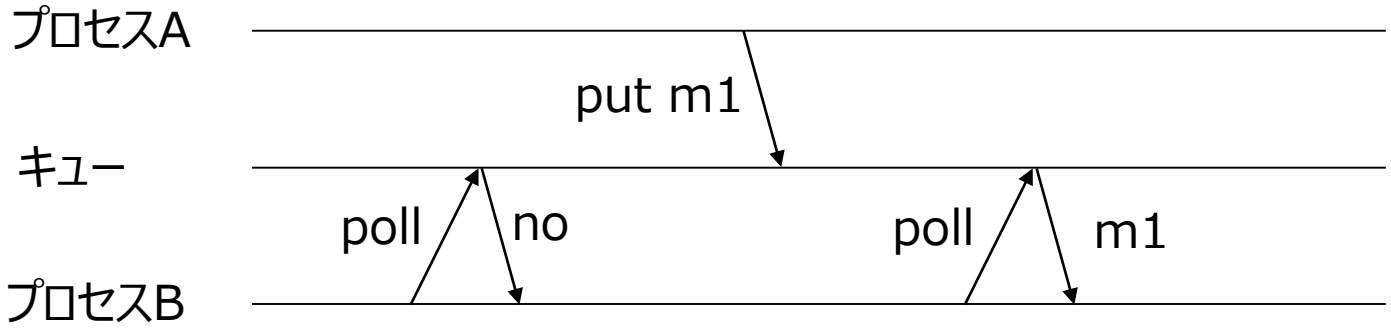
put	キューにメッセージを追加
get	キューのメッセージを取得 (キューが空だとブロック)
poll	キューのメッセージを取得 (キューが空でもブロックされない)
notify	キューにメッセージが追加されたときに呼ばれるハンドラを設定

メッセージキューイングシステム

キューの操作(put & get)

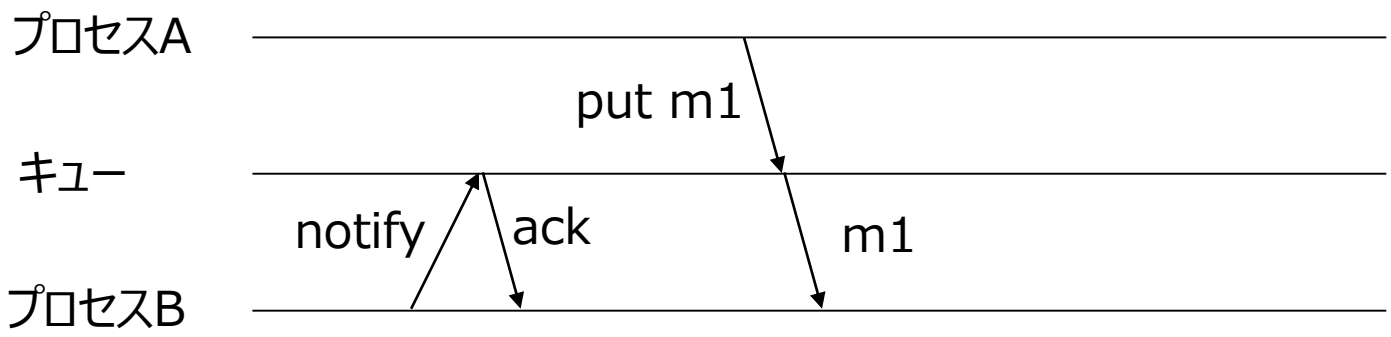


キューの操作(poll)



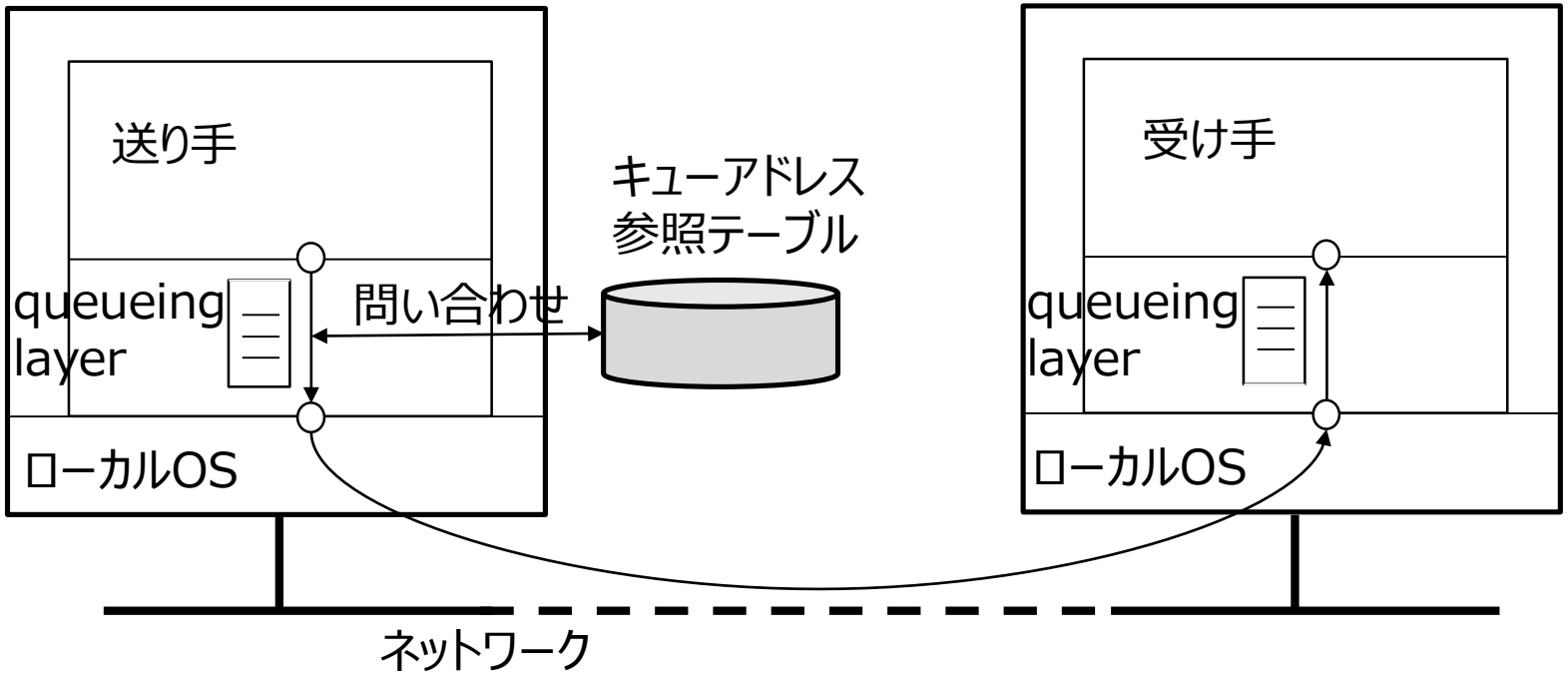
メッセージキューイングシステム

キューの操作(notify)



キューイングシステムの一般的構成

- ミドルウェアが提供するもの
  - 送信キューと受信キュー
  - キュー名とアドレスのデータベース



確認問題

- 以下の各文は正しいか。○か×で答えよ。
  - プロセス間の一時通信の特徴として、受信側はメッセージ投入時に実行されている必要がないことが挙げられる。
  - メッセージキューイングシステムにおいては、送信側は送信したメッセージがいつ受信側に利用されるかはわからない。
  - メッセージキューイングシステムにおいて、キューが空の場合にメッセージを取得しようとすると、メッセージが取得可能になるまでの間、プロセスが必ずブロックされてしまう。

講義内容

- 通信
  - プロトコル
  - RPC (遠隔手続き呼び出し)
  - メッセージ型一時通信
    - ソケット
  - メッセージ型永続通信
    - メッセージキューイングシステム
  - ストリーム型通信
  - マルチキャスト通信

ストリーム型通信

- (例) オーディオストリーム、ビデオストリーム
- 原音が44100Hzでサンプリング
  → 1/44100秒で演奏しなければ不正確な再生になる
- 時間依存の情報を交換するための分散システムは
  どのような機能を持たなければならないか？

- 連続表現メディア(continuous representation media) :
  データアイテム間の時間的な関係がデータの正確な
  解釈のために必須
  - (例) 動画：順番に画像を表示しなければならない
- 離散表現メディア(discrete representation media) ：
  データアイテム間の時間的な関係は必須ではない
  - (例) テキスト情報、静止画像

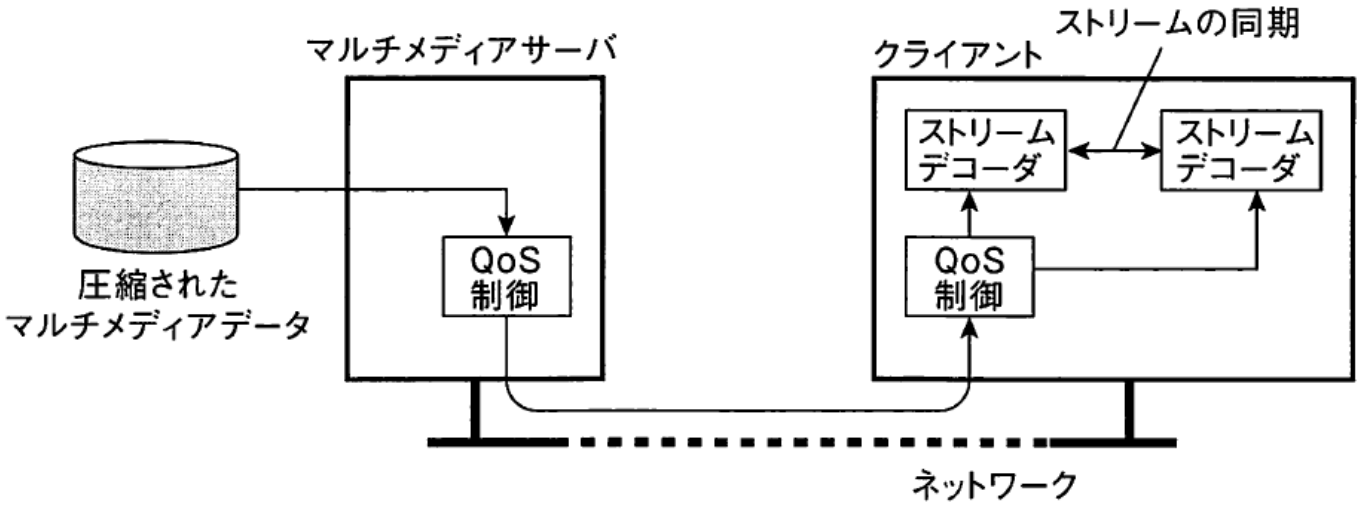
データストリーム

- データストリーム：データの流れ
  - 連続メディアをサポートするためにタイミングの扱いが必須

- 非同期型転送モード(asynchronous transmission mode)：アイテムの送信は順々に行われるが、いつ送信されるべきかについての制約はない
- 同期型転送モード(synchronous transmission mode)：端末間で最大遅延が定められる(データがそれよりも早く到着してもOK)
- 等時性転送モード(isochronous transmission mode)：時間通りにデータを転送しなければならない(遅延時間の幅が定められる)
  - オーディオ、ビデオの配信において特に重要

データストリーム

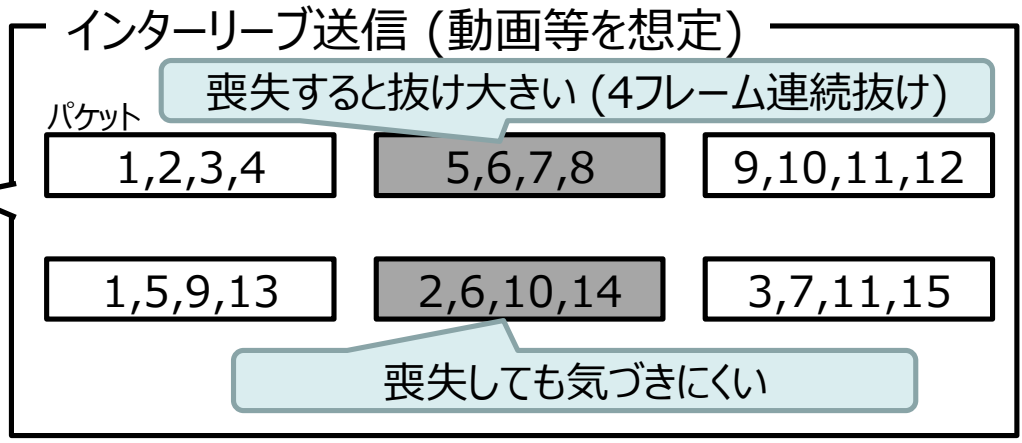
- 等時性転送を用いた連続データストリーム
  - 単純ストリーム：データが単一な並びのみから構成
  - 複合ストリーム：複数のサブストリームから構成
    - サブストリームどうしも時間依存
    - (例) ステレオ方式オーディオ：
      左右2つの音源＝2つのサブストリームが互いに同期



QoS: Quality of Service

サービス品質

- 連続データストリームの場合のQoS
  - 適時性 (timeliness)
  - ボリューム (volume)
  - 信頼性 (reliability)
- QoSの実施
  - パケットごとに優先度を設定可能
  - バッファの使用
  - 誤り訂正技術
  - インターリーブ送信
    - データの時間的順序を入れ替えて送信



ストリーム間の同期

- 離散データストリームと連続データストリーム間
  - それほど厳しい要件ではない
  - (例) スライドショーに音声を付ける
- 連続データストリームどうし
  - ステレオオーディオストリーム
    - サンプリング周期44100Hz
      → 1/44100 = 23マイクロ秒ごとに同期
  - ビデオとオーディオ(リップシンクロ)
    - 1フレームごとにオーディオ情報をグループ化することができる(カラーテレビの規格30Hzだと、33ミリ秒単位)
- 実際にはストリームごとに遅延の大きさが異なる
  → 送信側でストリームをマージ(多重化)

確認問題

- 以下の各文は正しいか。○か×で答えよ。
  - 同期型転送モードでは一定の最大遅延が定められ、データはそれより早く到着していれば問題ない。
  - 等時性転送では全く遅延なくデータが配信されることが要求される。
  - インターリーブ送信ではストリームデータの送信順を変更することで、パケットが喪失した時の影響を減らすことができる。
  - 複合ストリームの送信では、サブストリーム間の同期も考慮しなければならない。

講義内容

- 通信
  - プロトコル
  - RPC (遠隔手続き呼び出し)
  - メッセージ型一時通信
    - ソケット
  - メッセージ型永続通信
    - メッセージキューイングシステム
  - ストリーム型通信
  - マルチキャスト通信

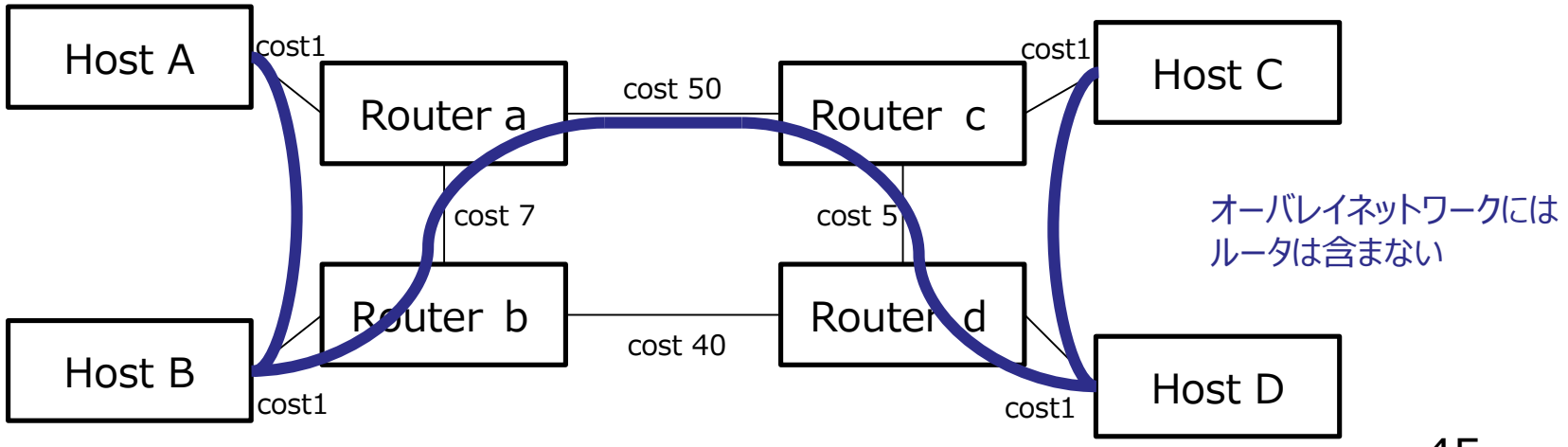
マルチキャスト通信

- 同時に複数のマシンと通信
  - ブロードキャストとは違って、決められたマシンに対して送信

- アプリケーションレベルのマルチキャスト
  - 各ノード間にオーバレイネットワークを構成
    → その構成員に情報を転送
  - オーバレイネットワーク(overlay network)：実際のネットワーク上に作る仮想的なネットワーク
    → 下位がどのようなネットワークになっているか意識する必要がない

オーバレイの構築

- 最適なオーバレイネットワークの構築のための指標例
  - リンクストレス(link stress)：
    同じメッセージが同じリンクを辿る回数
    - (例) AからDに送るとき、Router a-b間、Host B-Router b間は2回通る
  - 相対遅延ペナルティ(RDP: relative delay penalty)：
    下位ネットワークを直接使った場合の遅延と
    オーバレイネットワークを使った場合の遅延の比率
    - (例) B-D間：(1+7+50+5+1)/(1+40+1)=1.52



エピデミックプロトコル

エピデミック(epidemic):伝染性の

- 概要
  - 分散システム内で情報を速やかに伝播させる方法
  - ゴシップベース情報散布、ゴシッピングとも
  - 情報散布(information dissemination)を制御する中央の機能はない
  - 伝播した情報の削除は困難
- 各ノードの状態
  - infected: 拡散させるべきデータdを持っている
  - susceptible: dを持っていない
  - removed: dを持っているが拡散しない
    - 自分の知っているノードがすでに新しい情報を持っている等
- anti-entropy model: 一般的な伝播モデル (→次スライド)

エピデミックプロトコル

- anti-entropy model:
  - 方法
    - Push based: Pはランダムに自分が知っているサーバQを選択し、情報を伝播させる
    - Pull based: QはPから新しい更新を取得する
    - Push-pull based: P,Qは互いに情報を送信する
  - 特徴
    - infectedなノードが少ない → 伝播速度が遅い
    - infectedなノードが多い
      → susceptibleノードが選択される確率が低い
    - infectedなノードが多い場合はPull basedが有効

確認問題

- 以下の説明に合う用語を答えよ。
  - (1) マルチキャスト通信等において、実際のネットワークの上に作る仮想的なネットワーク
  - (2) (1)において同じメッセージが同じリンクを辿る回数
  - (3) 下位ネットワークを直接使った場合の遅延と(1)を使った場合の遅延の比率
- エピデミックプロトコルにおいて、infectedな(既に伝播するべき情報を持っている)ノードが大半を占める場合、push basedな伝播手法は効率が悪い。この理由を説明せよ。

参考文献

- 「分散システム」
  水野 忠則 監修、共立出版、2015
- 「分散システム 原理とパラダイム 第2版」
  アンドリュー・S・タネンバウム 他 著、
  ピアソン・エデュケーション、2009
- 「コンピュータネットワークム 第4版」
  アンドリュー・S・タネンバウム 著、日経BP社2003
- 「分散処理」 谷口秀夫編著、オーム社、2005