ネットワークアプリケーション

第2回 ネットワークアプリケーションの プログラミングモデル

石井 健太郎
（423研究室・オフィスアワー水3限）

スケジュール

9月15日　第1回「TCP/IPプロトコルスイート」
9月29日　第2回「ネットワークアプリケーションのプログラミングモデル」
10月6日　第3回「アプリケーションプロトコルの設計（1）」
10月13日　第4回「アプリケーションプロトコルの設計（2）」
10月20日　第5回「アプリケーションプロトコルの設計（3）」
10月27日　第6回「アプリケーションプロトコルの設計（4）」　演習（第3演習室）
11月10日　第7回「サーバサイドウェブプログラミング（1）」
11月17日　第8回「サーバサイドウェブプログラミング（2）」

スケジュール

11月24日	第9回「サーバサイドウェブプログラミング（3）」	
12月1日	第10回「サーバサイドウェブプログラミング（4）」	演習（第3演習室）
12月8日	第11回「クライアントサイドウェブプログラミング（1）」	
12月15日	第12回「クライアントサイドウェブプログラミング（2）」	
12月22日	第13回「クライアントサイドウェブプログラミング（3）」	
1月12日	第14回「クライアントサイドウェブプログラミング（4）」	演習（第3演習室）
1月19日	第15回「まとめと演習」	演習（第3演習室）

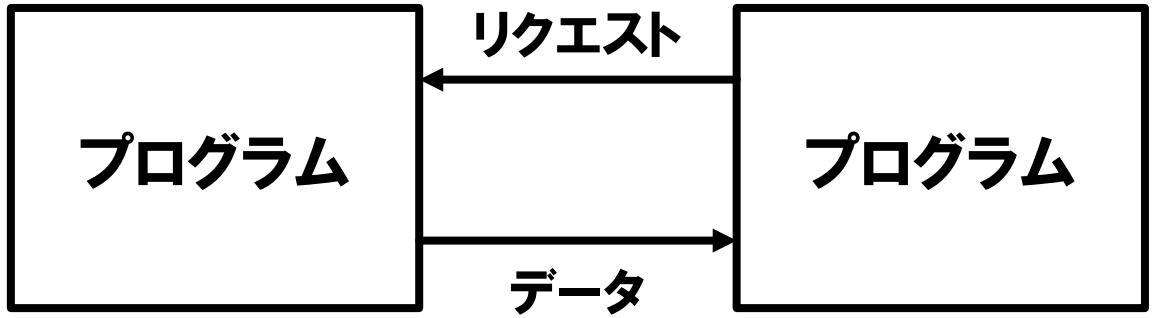
- レポートを提出してください
  - 講義終了時でも可

ソケット通信でプログラム同士が連携するには

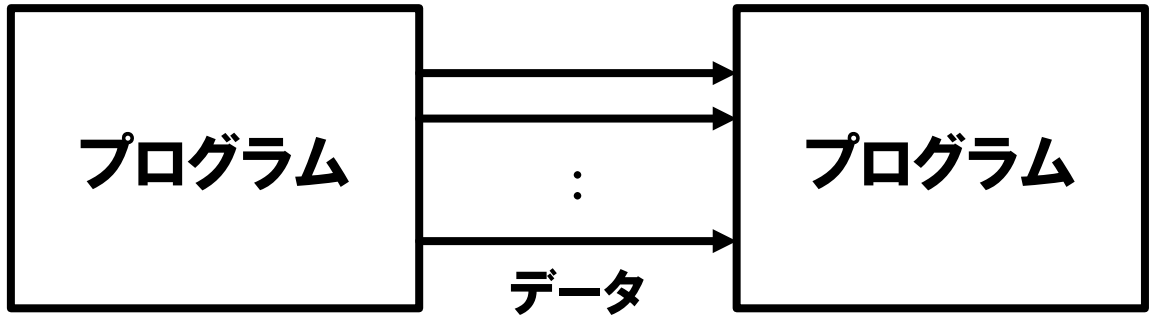
- 実現したいアプリケーションにより，下記の選択を行う必要がある
  - メッセージパッシングかストリーミングか
  - テキストデータかバイナリデータか
  - TCPかUDPか

メッセージパッシングかストリーミングか

- メッセージパッシング



- ストリーミング



メッセージパッシングかストリーミングか

- メッセージパッシング
  - 交換されるデータが状況によって変わる場合
  - 同期をとって動作する場合

- ストリーミング
  - 交換されるデータ形式が決まっている場合
  - 最新のデータのみを必要とする場合 (cf. UDPの説明)

テキストデータかバイナリデータか

- テキストデータ（ASCII文字列）
  - トラブルが少ない
    - バイトオーダーは関係ない
    - データがあっているのか確認しやすい
  - 区切り文字（列）を利用してリスト構造を表現しやすい

- バイナリデータ
  - データのサイズを抑えられる
  - バイトオーダーに注意
    - ネットワークに流れるデータ（IPヘッダ・TCPヘッダなども含めて）は
      ビッグエンディアンが使用されている
    - htonl(), htons(), ntohl(), ntohs()

（参考）テキストデータかバイナリデータか

- Base64エンコーディング
  - 64種類の文字でデータを表現する

- URLエンコーディング
  - 特殊文字やマルチバイト文字を「％（16進数）」で表現する

TCPかUDPか

- TCP
  - エンドツーエンドの信頼性の保証・制御行う
    - 到着順序保証・フロー制御・輻輳制御
  - コネクション型

- UDP
  - エンドツーエンドの制御を行わない
    - （多くの場合）TCPより高速
  - コネクションレス型

TCPかUDPか

- TCP
  - もれなくデータを転送したい場合
  - データの順序が重要である場合
  - TLS/SSLにより通信路を暗号化したい場合
  - 交換されるデータに対して，伝送路の帯域・端末の処理能力に余裕がある場合
  - 交換されるデータに対して，伝送路の帯域がひっぱくしている場合

- UDP
  - データにもれがあっても問題ない場合
    - 動画・音声のストリーミング
    - 重要性の低いメッセージ
  - データの順序が重要でない場合
    - 最新のデータのみを必要とする場合
  - あて先があらかじめ決められない場合
    - ブロードキャストによる対話相手の発見
  - 交換されるデータに対して，伝送路の帯域・端末の処理能力がそこそこたいへんな場合

TCPかUDPか

- TCP
  - 接続するまでの手順は多い．接続してからのプログラミングは比較的簡単．
  - socket (), bind (), listen (), accept (), connect (), send (), recv (), close ()

- UDP
  - 1回の送受信の記述は簡単．送信の都度，あて先を指定する．
  - socket (), bind (), sendto (), recvfrom ()

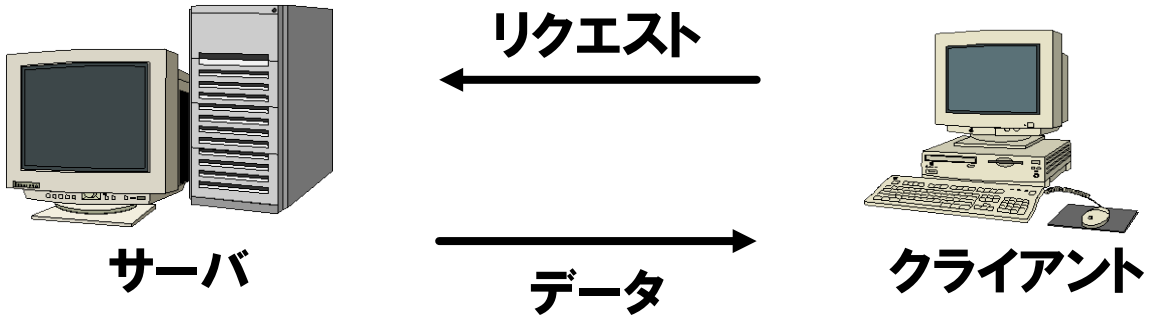
（参考）デジタル通信・アナログ通信

- ソケット通信を題材として話を進めているが，
  ほかのデジタル通信（0・1を送受信する通信）でも
  ある程度は知識を適用できる
  - シリアル通信
  - 基礎演習の論理回路
  - 可視光・赤外線通信（テレビのリモコン）
  - …

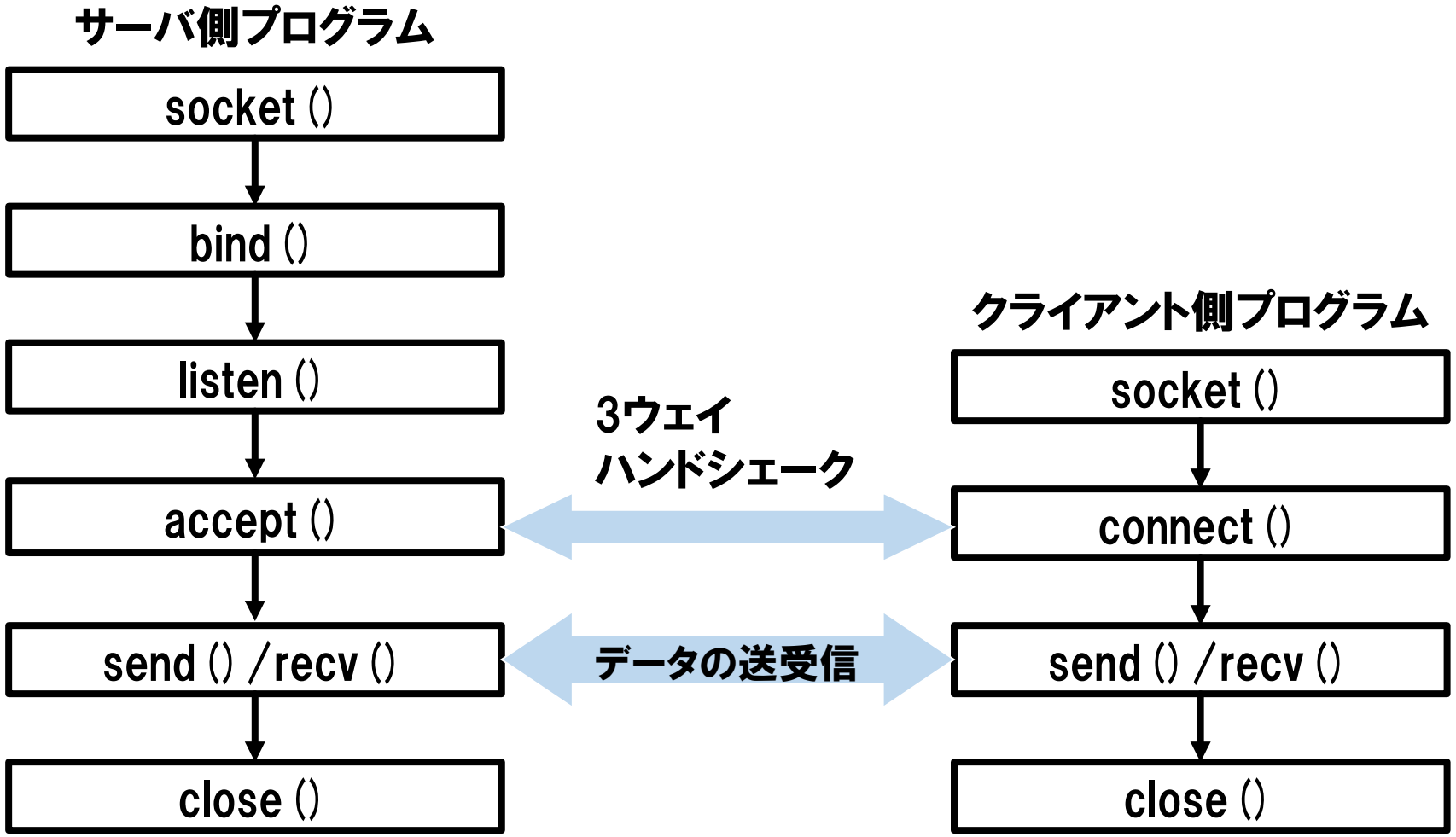
- アナログ通信（例えばAMラジオ・FMラジオなど）では適用できない

サーバクライアントモデル

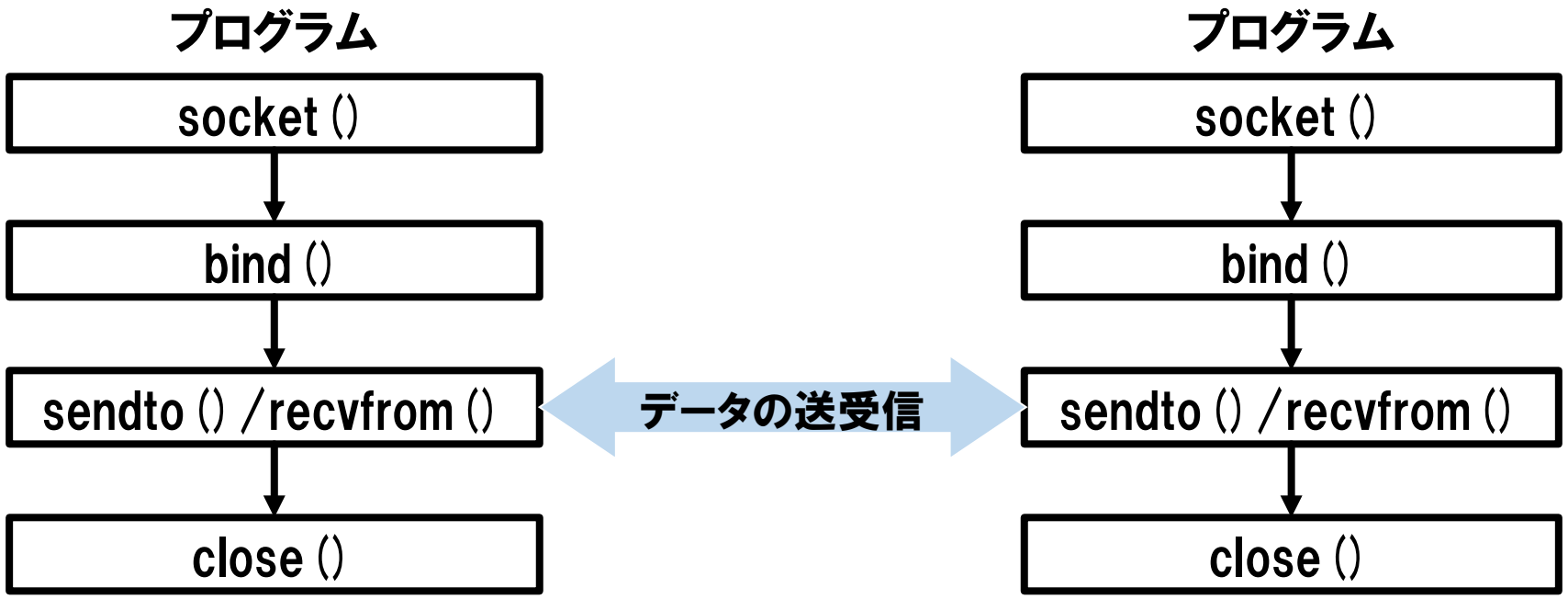
- サーバ（サービス提供者）とクライアント（客・サービス利用者）にあてはめてシステムを考えることが多い
  - TCPでは，接続待機・接続要求がこのモデルに基づいている



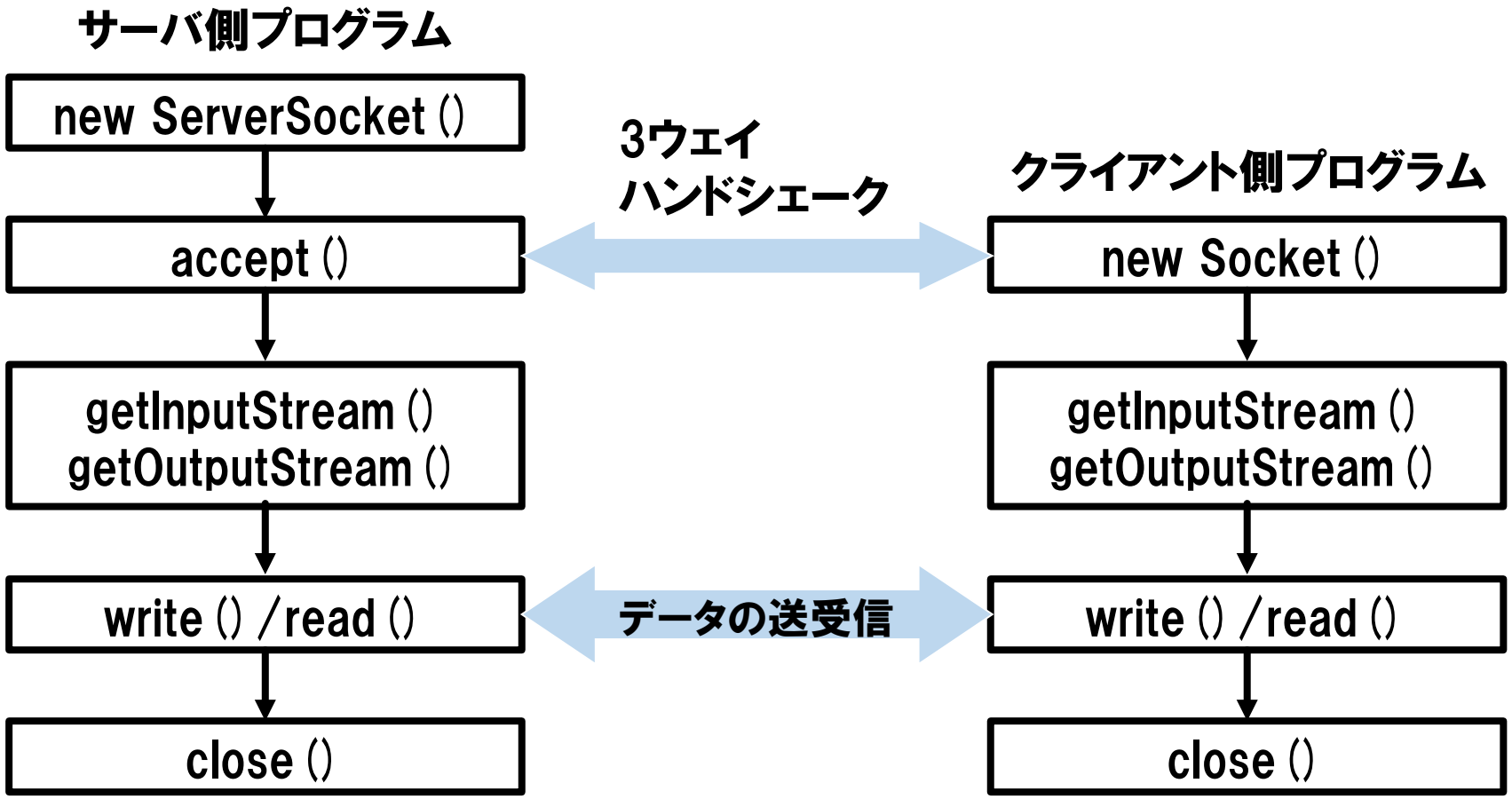
TCPによるデータの送受信



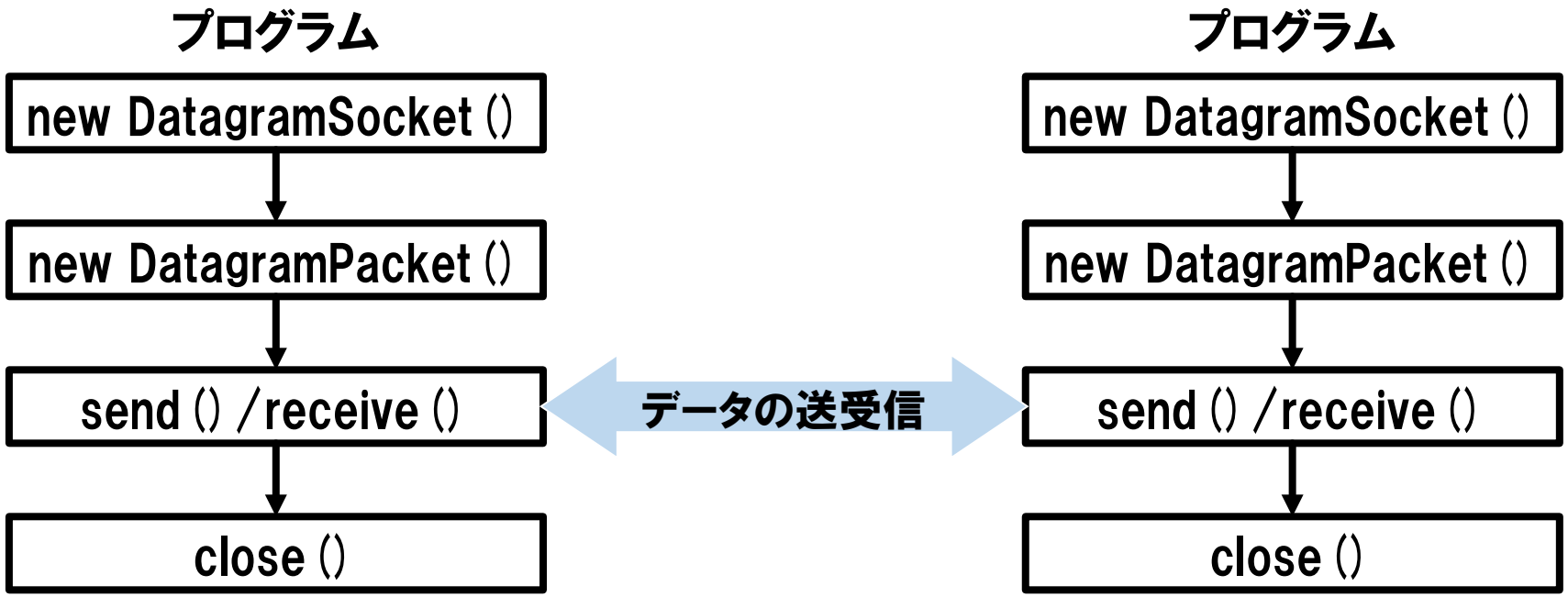
UDPによるデータの送受信



（参考）TCPによるデータの送受信（Java編）



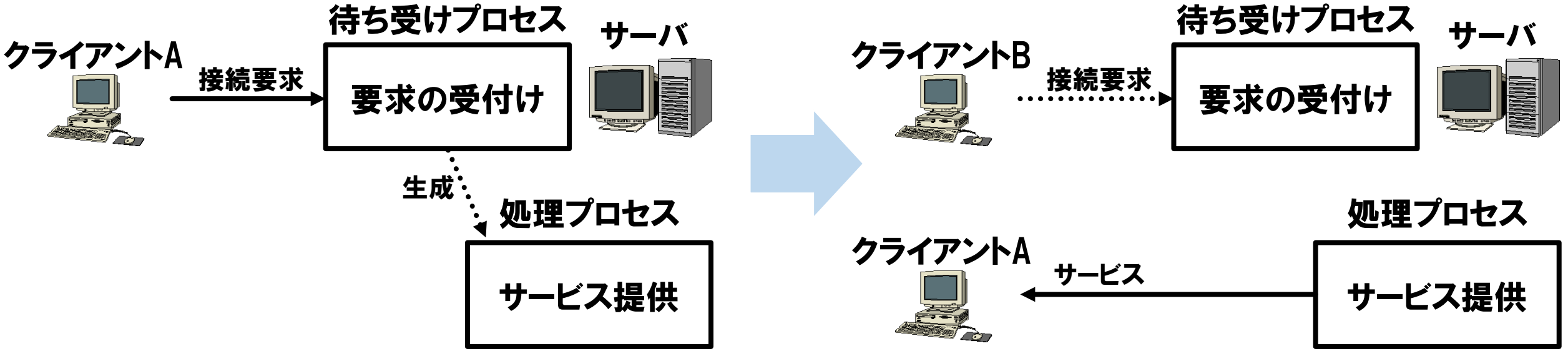
（参考）UDPによるデータの送受信（Java編）



マルチプロセス化・マルチスレッド化

- accept()，recv()，recvfrom()　はブロッキングコール
  - あるプログラムの接続・データを待っているとほかの処理を一切行えない

- マルチプロセス化（fork）・マルチスレッド化による並行処理が必要



最終課題について

- 選択制
  - 3つのテーマの中から好きなものを選んで提出する
    - 独自のアプリケーションプロトコル
    - サーバサイドウェブプログラミング
    - クライアントサイドウェブプログラミング
  - 複数のテーマの課題が提出された場合，最も高い点数のテーマのみ採用する

- 先願優先主義
  - 内容に重複があった場合，（特許と同じ）先願優先主義にて採点する
  - つまり，同じような内容の課題が提出された場合，
    あとに提出された課題の評価を減点する

最終課題について

- プログラム実行に基づく評価
  - プログラムのソースコードを提出
    → こちらでコンパイル・設置のうえ実行した結果により点数をつける
    - どんなに設計がよくできていても機能が充実していても，
      実行できない場合点数がつきません．
      ただし，実行できない場合はメールにて連絡するようにします．
  - プログラムの起動方法・必要な外部ライブラリ・設定ファイルの配置方法など，
    プログラムを実行するために必要な説明を記した簡単なドキュメントも提出
  - その他任意で，ドキュメント以外の参考資料も提出可
    - 例えば，正しく動作しているときの動画などもOK

- 最終提出期限
  - 試験期間の直前か直後にする予定（スケジュールが固まってきたら決定）

宿題

- 自分にとってなじみのある通信アプリケーションを1つ選び，
  本日学んだ通信方式の分類について調査する
  - まず，選んだアプリケーションを示し，
    それがどのようなアプリケーションなのかを概説する
  - 次に，そのアプリケーションが
    どの分類の通信方式を採用しているか可能な限り説明する
    - 完全にプロトコルが公開されている題材はみつかりにくいかもしれないが，
      分類できる項目が多い題材であるのが望ましい

- レポート（A4用紙1～2枚）にして，次回の講義終了時までに提出
  - 様式自由，ただし，学生番号と名前をわかる場所に書くこと