ネットワークアプリケーション

第3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・オフィスアワー火3限）

スケジュール

9月27日　第1回「TCP/IPプロトコルスイート」
10月4日　第2回「Javaによるウィンドウプログラミング」
10月11日　第3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18日　第4回「Javaによるネットワークプログラミング」
10月25日　第5回「Javaによるネットワークプログラミング」
11月8日　第6回「Javaによるネットワークプログラミング」
11月15日　第7回「Javaによるネットワークプログラミング」　最終課題（1）
11月17日　第8回「ウェブプログラミングについて」

スケジュール

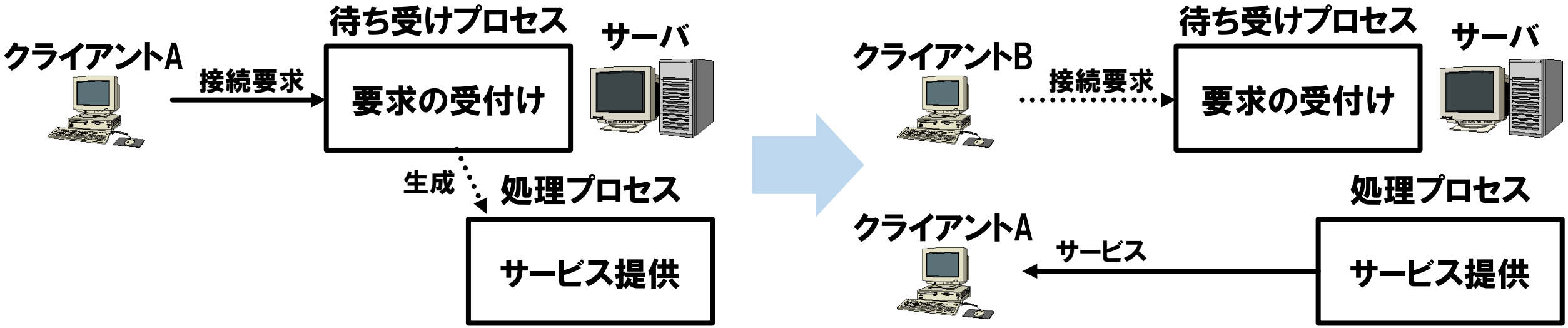
11月22日　第9回「JavaScriptによるクライアントサイドウェブプログラミング」
11月29日　第10回「JavaScriptによるクライアントサイドウェブプログラミング」
12月6日　第11回「JavaScriptによるクライアントサイドウェブプログラミング」
12月13日　第12回「JavaScriptによるクライアントサイドウェブプログラミング」
12月20日　第13回「JavaScriptによるクライアントサイドウェブプログラミング」
1月10日　第14回「JavaScriptによるクライアントサイド...」　**最終課題 (2)**
1月19日　第15回「まとめと演習」

- 演習課題を提出してください

- 本日はUDPによるデータの送受信プログラムを作成する

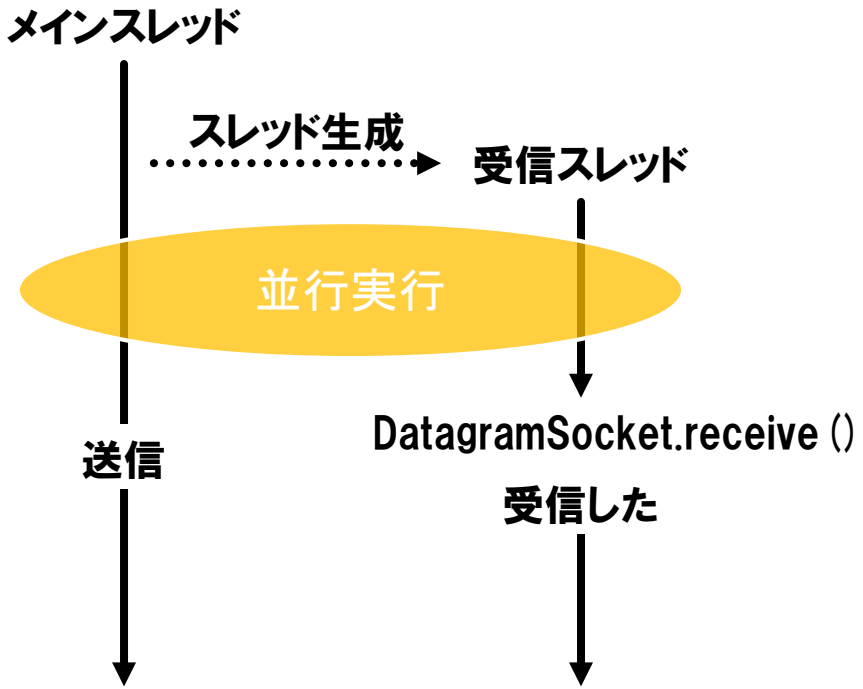
マルチプロセス化・マルチスレッド化

- これから習う DatagramSocket.receive (), ServerSocket.accept (), BufferedReader.readLine () などはブロッキングコール
  - 相手がたのプログラムの接続・データを待っているとほかの処理を行えない
- 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
  - 下図はマルチプロセス化の概念図. この講義ではマルチスレッド化で対処する.



マルチプロセス化・マルチスレッド化

- これから習う DatagramSocket.receive () , ServerSocket.accept () , BufferedReader.readLine () などはブロッキングコール
  - 相手がたのプログラムの接続・データを待っているとほかの処理を行えない



マルチスレッド化

- まずは Java におけるスレッドの生成方法について習う

- スレッドは Threadクラス のサブクラスとして定義して用いる
  - Runnableインタフェース を実装するという方法もある(がこの講義では割愛)
    https://docs.oracle.com/javase/jp/8/docs/api/java/lang/Thread.html

- Thread thread = new Thread();
  thread.start();
  として，インスタンス生成してから，スレッドをスタートする
  - new Thread() すると，コンストラクタが(現在のスレッドで)実行される
  - thread.start() すると， run() メソッドが(新たなスレッドで)実行される

マルチスレッド化

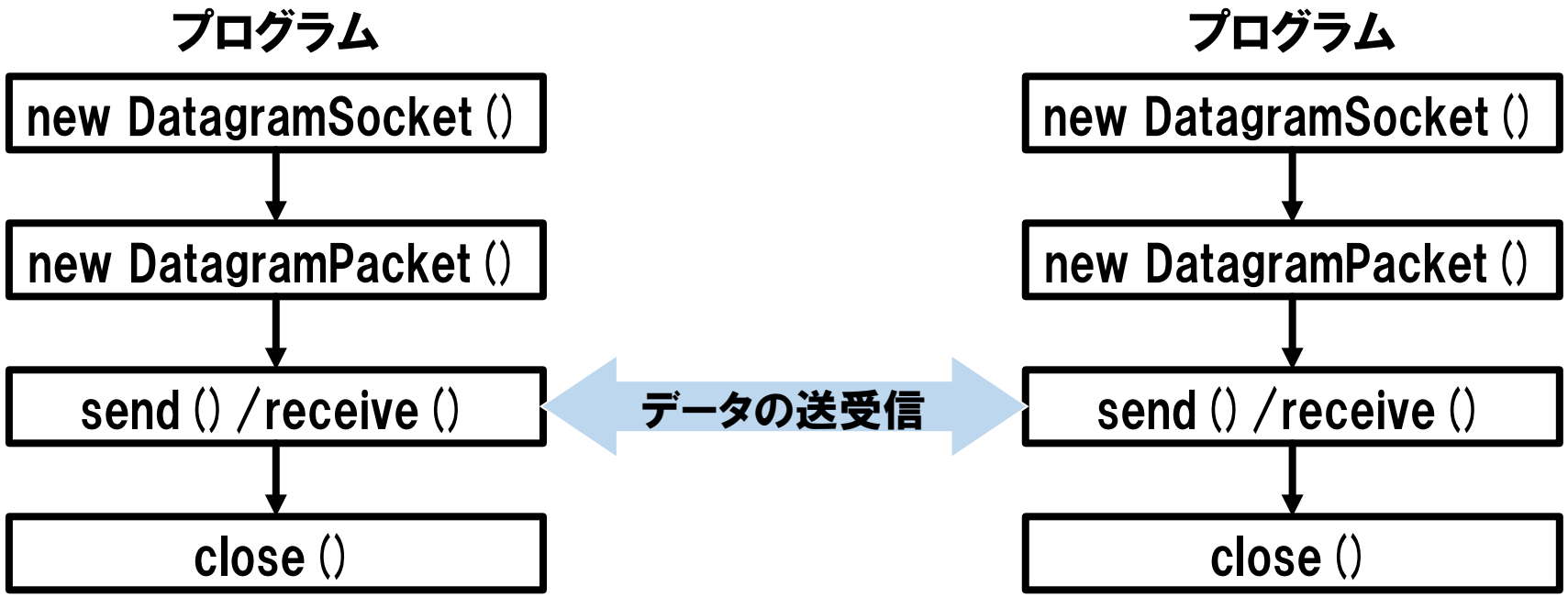
```
this.senderReceiver = new UDPSenderReceiver(this, "localhost", 7777, 8888);
this.senderReceiver.start();
```

UDPによるデータの送受信

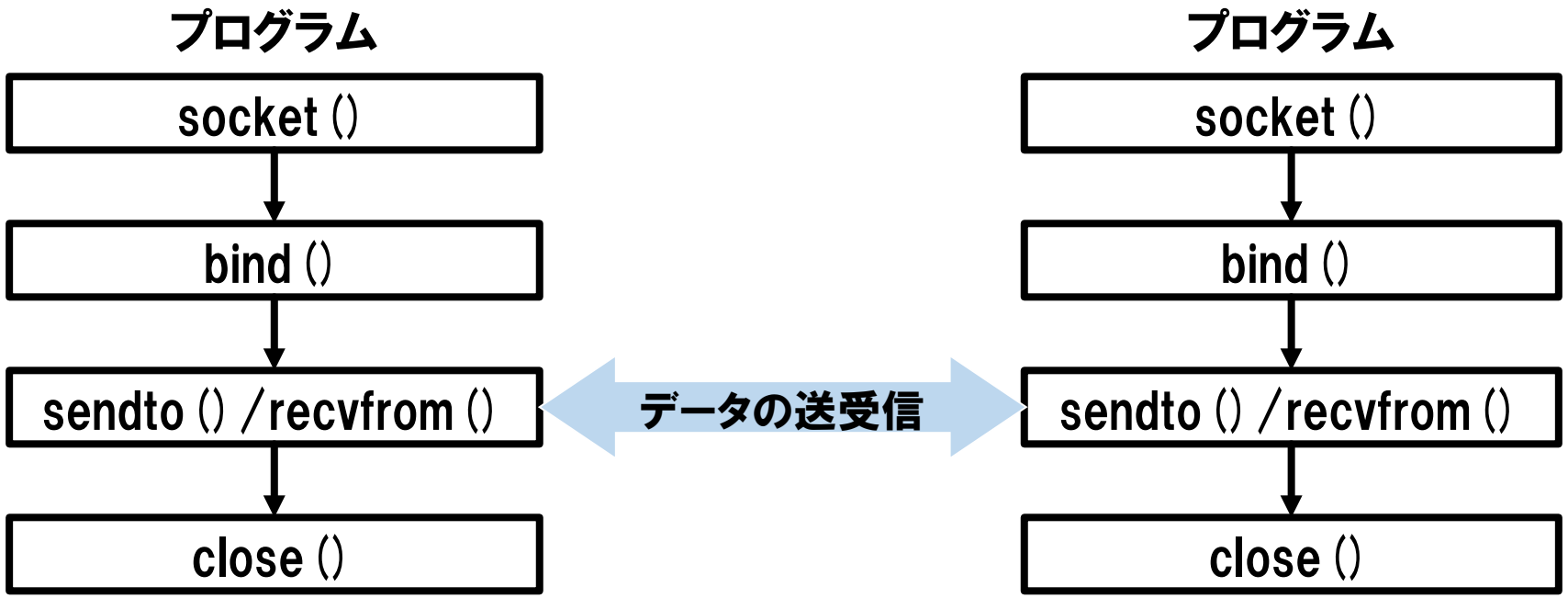
- コネクションレス（接続しない）
- データは byte型の配列 として送受信される
  - テキストデータのやりとりとバイナリデータのやりとりの方法は同じ

- DatagramSocketクラス と DatagramPacketクラス を用いる
  - cf. TCPは Socketクラス と ServerSocketクラス を用いる

UDPによるデータの送受信



（参考）UDPによるデータの送受信（C言語編）



UDPによるデータの送信

- テキストデータの送信

```
try {
    byte[] bytes = string.getBytes();
    DatagramPacket packet = new DatagramPacket(bytes, bytes.length, this.address);
    this.socket.send(packet);
} catch (SocketException se) {
    se.printStackTrace();
} catch (IOException ioe) {
    ioe.printStackTrace();
}
```

- バイナリデータの送信

```
try {
    byte[] bytes = ByteBuffer.allocate(4).putInt(i).array();
    DatagramPacket packet = new DatagramPacket(bytes, bytes.length, this.address);
    this.socket.send(packet);
} catch (SocketException se) {
    se.printStackTrace();
} catch (IOException ioe) {
    ioe.printStackTrace();
}
```

UDPによるデータの受信

- テキストデータの受信

```
byte[] buffer = new byte[1024];
DatagramPacket packet = new DatagramPacket(buffer, buffer.length);

try {
    this.socket.receive(packet);
} catch (IOException ioe) {
    ioe.printStackTrace();
    break;
}

byte[] data = packet.getData();
String string = new String(data);
System.out.println(string);
// this.application.setString(string);
```

- バイナリデータの受信

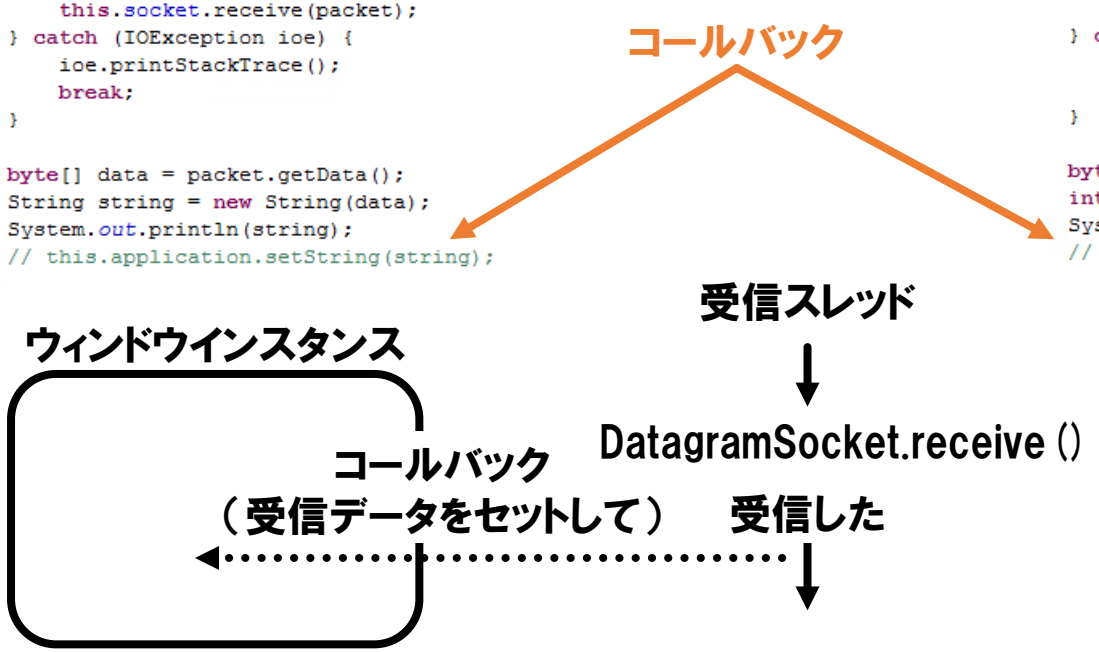
```
byte[] buffer = new byte[1024];
DatagramPacket packet = new DatagramPacket(buffer, buffer.length);

try {
    this.socket.receive(packet);
} catch (IOException ioe) {
    ioe.printStackTrace();
    break;
}

byte[] data = packet.getData();
int i = ByteBuffer.wrap(data).getInt();
System.out.println(i);
// this.application.setInt(i);
```

受信したときの処理

- 受信したときの処理を
  - 受信スレッドで処理できるときはそのまま処理する（エコーバックするなど）
  - ウィンドウに反映させたいときはコールバックインスタンスを用いる

```
byte[] buffer = new byte[1024];
DatagramPacket packet = new DatagramPacket(buffer, buffer.length);

try {
    this.socket.receive(packet);
} catch (IOException ioe) {
    ioe.printStackTrace();
    break;
}

byte[] data = packet.getData();
String string = new String(data);
System.out.println(string);
// this.application.setString(string);
```

```
byte[] buffer = new byte[1024];
DatagramPacket packet = new DatagramPacket(buffer, buffer.length);

try {
    this.socket.receive(packet);
} catch (IOException ioe) {
    ioe.printStackTrace();
    break;
}

byte[] data = packet.getData();
int i = ByteBuffer.wrap(data).getInt();
System.out.println(i);
// this.application.setInt(i);
```



演習

- UDPによるデータの送受信を用いて，ウィンドウアプリケーションを作成する
  - 基本課題
    - マウスのクリック位置を相
    - ログラムに
    - そのデータを文字で表示させる
    - 受信したデータに応じた位
    - 描画
      見た目におもしろいプログラ
  - 発展課題
    - クリック位置だけじゃなく，いろんな
    - 信して，おもしろいアプリケーションを考えよう
      - 例えば，ドラッグすると相手が
      - お絵かきソフトとするなど

- プログラム（UDP*.java）の
  - ，1つの
  - ファイルにコピー
  して，
  次回の講義開始時までに，
  - 出フォルダ（X
  - イブ）にファイルで提出
    - X:¥IN科専門¥石井講師¥ネットワークアプリケーション¥第3回
    - ファイル名は「<学生番号>.txt」とする（ハイフンなし）　例: N14999.txt