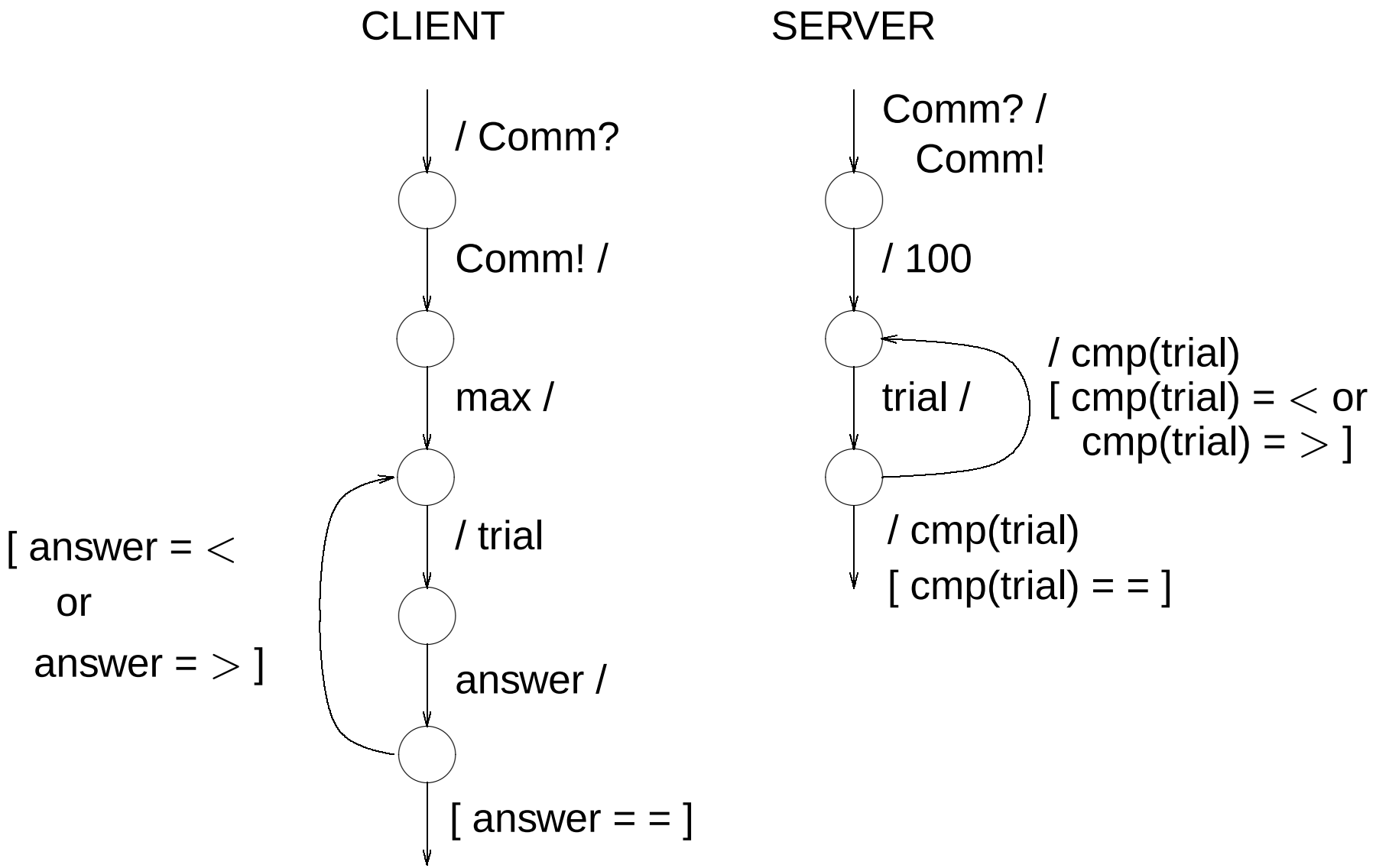
Threads in der Client/Server-Programmierung mit Java

Zahlenraten: Protokoll



Zahlenraten: Client

```
try {
  socket = new Socket(host, port);
  try {
    in = new BufferedReader(new InputStreamReader(socket.getInputStream()));
    out = new PrintWriter(socket.getOutputStream(), true);
    dialog();
  } catch (IOException e) {
      System.err.println(e);
    } finally {
        try {
          if (in != null)
            in.close();
        } catch (IOException e) {
          }
      }
} catch (IOException e) {
    System.err.println(e);
  }
```

Java-Implementierung: Client (contd.)

```
protected void dialog() throws IOException {
  String line;
  int max;
  int trial;
  try {
    max = getmax();
    System.out.println("Ich habe mir eine Zahl zwischen 1 und " +
                       max + " ausgedacht!");
    System.out.println("Kannst Du sie eraten?");
    for (int trialno = 1; trialno != 0; ) {
      trial = gettrial(trialno);
      puttrial(trial);
      switch (getanswer()) {
        case 0: {
          int maxtrialno = (int)Math.ceil(Math.log(max)/Math.log(2.0));
          System.out.println("Congratulations!");
          System.out.println("Du hast " + trialno + " Versuche benötigt!");
          if (trialno > maxtrialno)
            System.out.println("Es wäre in " + maxtrialno +
                               " zu schaffen gewesen");
          trialno = 0;
          break;
        }
```

Java-Implementierung: Client (contd.)

```
          case -1: {
            System.out.println("Meine Zahl ist kleiner!");
            trialno++;
            break;
          }
          case 1: {
            System.out.println("Meine Zahl ist größer!");
            trialno++;
            break;
          }
        }
      }
    } catch (UserProtocolException e) {
        System.err.println(e);
      }
  }
```

Java-Implementierung: Client (contd.)

```
int getanswer() throws UserProtocolException, IOException {
  String line;
  line = in.readLine();
  if (line == null)
    throw new UserProtocolException("Server closed connection.");
  if (line.startsWith("="))
    return 0;
  else
    if (line.startsWith("<"))
      return -1;
    else
      if (line.startsWith(">"))
        return 1;
      else
        throw new UserProtocolException("Protocol error.");
}
```

Java-Implementierung: Server

```
try {
  listen_socket = new ServerSocket(port);
  while (true) {
    Socket client_socket = listen_socket.accept();
    client_socket.setSoTimeout(10000);
    try {
      client_in = new BufferedReader(new InputStreamReader(
                                          client_socket.getInputStream()));
      client_out = new PrintWriter(client_socket.getOutputStream(), true);
      dialog();
    } catch (IOException e) {
        System.err.println(e);
      } finally {
          try {
            if (client_in != null)
              client_in.close();
          } catch (IOException e) {
            }
        }
  }
} catch (IOException e) {
    System.err.println(e);
  }
```

Java-Implementierung: Server (contd.)

```
protected void dialog() throws IOException {
  int number;
  int trial;
  try {
    putmax();
    number = (int)(Math.random() * 100.0) + 1;
    do {
      trial = gettrial();
      if (number == trial)
        putanswer("=");
      else
        if (number < trial)
          putanswer("<");
        else
          putanswer(">");
    } while (number != trial);
  } catch (UserProtocolException e) {
      System.err.println(e);
    }
}
```

Multi-Server

```
public MultiServer(int port) {
  super("MultiServer");
  this.port = port;
  try {
    listen_socket = new ServerSocket(port);
  } catch (IOException e) {
      fail(e, "Exception creating server socket");
    }
  threadgroup = new ThreadGroup("Server Connections");
  connections = new Vector();
  vulture = new Vulture(this);
  this.start();
}
```

Multi-Server (contd.)

```
  public void run() {
    try {
      while (true) {
        Socket client_socket = listen_socket.accept();
        Connection c = new Connection(client_socket, threadgroup, 3, vul-
ture);
        synchronized (connections) {
          connections.addElement(c);
        }
      }
    } catch (IOException e) {
        fail(e, "Exception while listening for connections");
      }
  }
```

Multi-Server: Connection

```
public Connection(Socket client_socket, ThreadGroup threadgroup,
                  int priority, Vulture vulture) {
  super(threadgroup, "Connection-" + connection_number++);
  this.setPriority(priority);
  client = client_socket;
  this.vulture = vulture;
  try {
    client.setSoTimeout(10000);
    client_in = new BufferedReader(new InputStreamReader(
                                      client.getInputStream()));
    client_out = new PrintWriter(client.getOutputStream(), true);
  } catch (IOException e) {
      try {
        client.close();
      } catch (IOException e2) {
        }
      System.err.println("Exception while getting socket streams:" + e);
      return;
    }
  this.start();
}
```

Multi-Server: Connection

```
public void run() {
  try {
    dialog();
  } catch (IOException e) {
    } finally {
        try {
          client.close();
        } catch (IOException e2) {
          }
        synchronized (vulture) {
          vulture.notify();
        }
      }
}
```

Multi-Server: Vulture

```
public synchronized void run() {
  for (;;) {
    try {
      this.wait(5000);
    } catch (InterruptedException e) {
      }
    synchronized (server.connections) {
      for (int i = 0; i < server.connections.size(); i++) {
        Connection c;
        c = (Connection)server.connections.elementAt(i);
        if (!c.isAlive()) {
          server.connections.removeElementAt(i);
          i--;
        }
      }
    }
  }
}
```