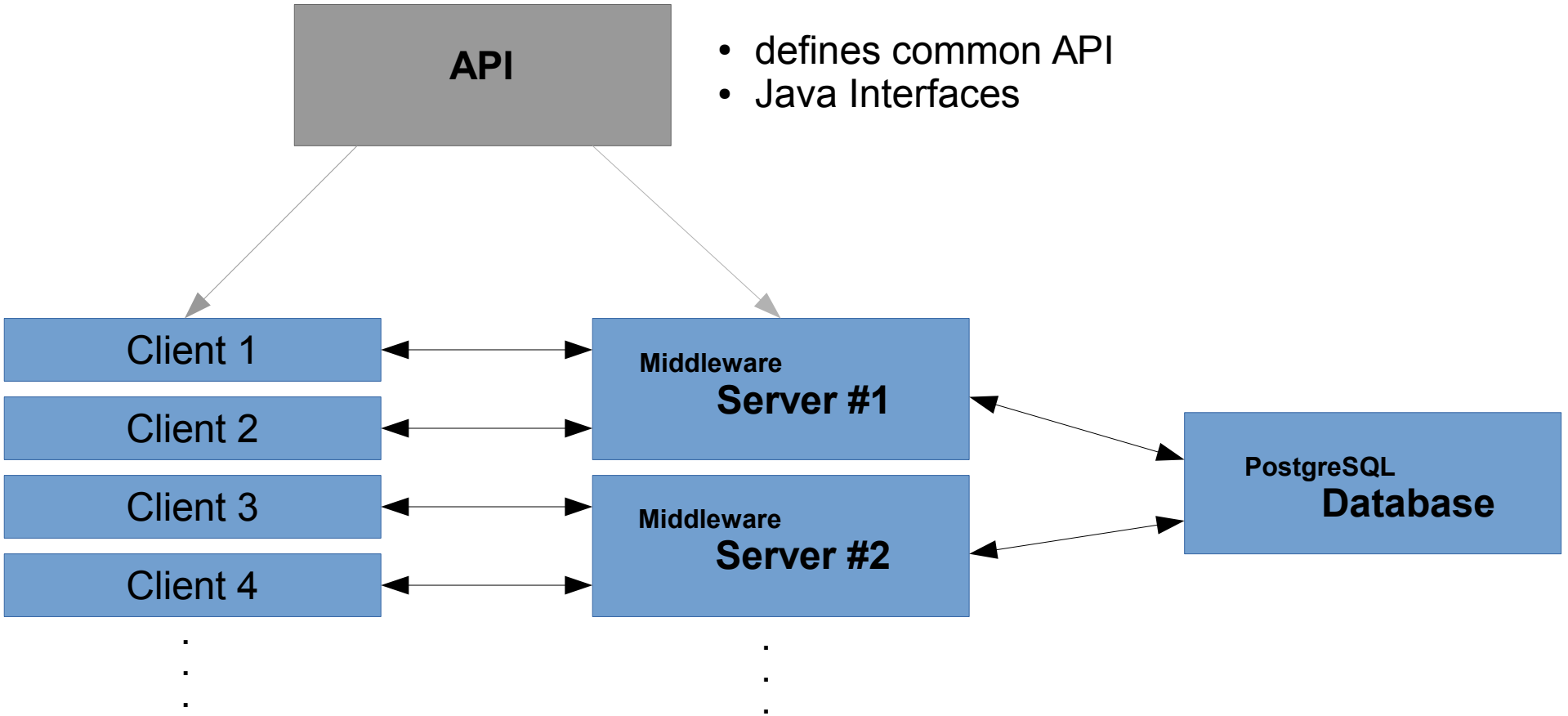
SimpleMQ

a simple message passing/queuing system

Advanced Systems Lab, Marcel Mohler, ETH Zurich

Multi tier system



- defines common API
- Java Interfaces

- > 100 clients
- create/delete/query queues
- send/receive messages

- at least 2 server

- single database
- plain PostgreSQL

Clients

- started with unique ID and middleware IP
- send request, wait for ACK, then proceed (closed loop system)
- evenly distribute clients on middleware server

```
// single client main loop
while(true) {
  with same probability {
     queryAllQueues();
     queryQueue();
     sendSpecificMessage();
     sendAllMessage();
  }
}
```

Middleware Server – Initializer

- single thread listening on preconfigured port
- at first connection attempt from client

  → create **Connection** with unique port (per client)

  → store in ConnectionList

- ConnectionList contains Connection and busy-flag

Middleware Server – Distributor

- single thread that iterates over ConnectionList
- checks each **Connection:** if not busy and message awaiting

  → assign Worker from fixed size pool (depending on machine cores) and hand over **Connection**
- in case no **Worker** available

  → block until available

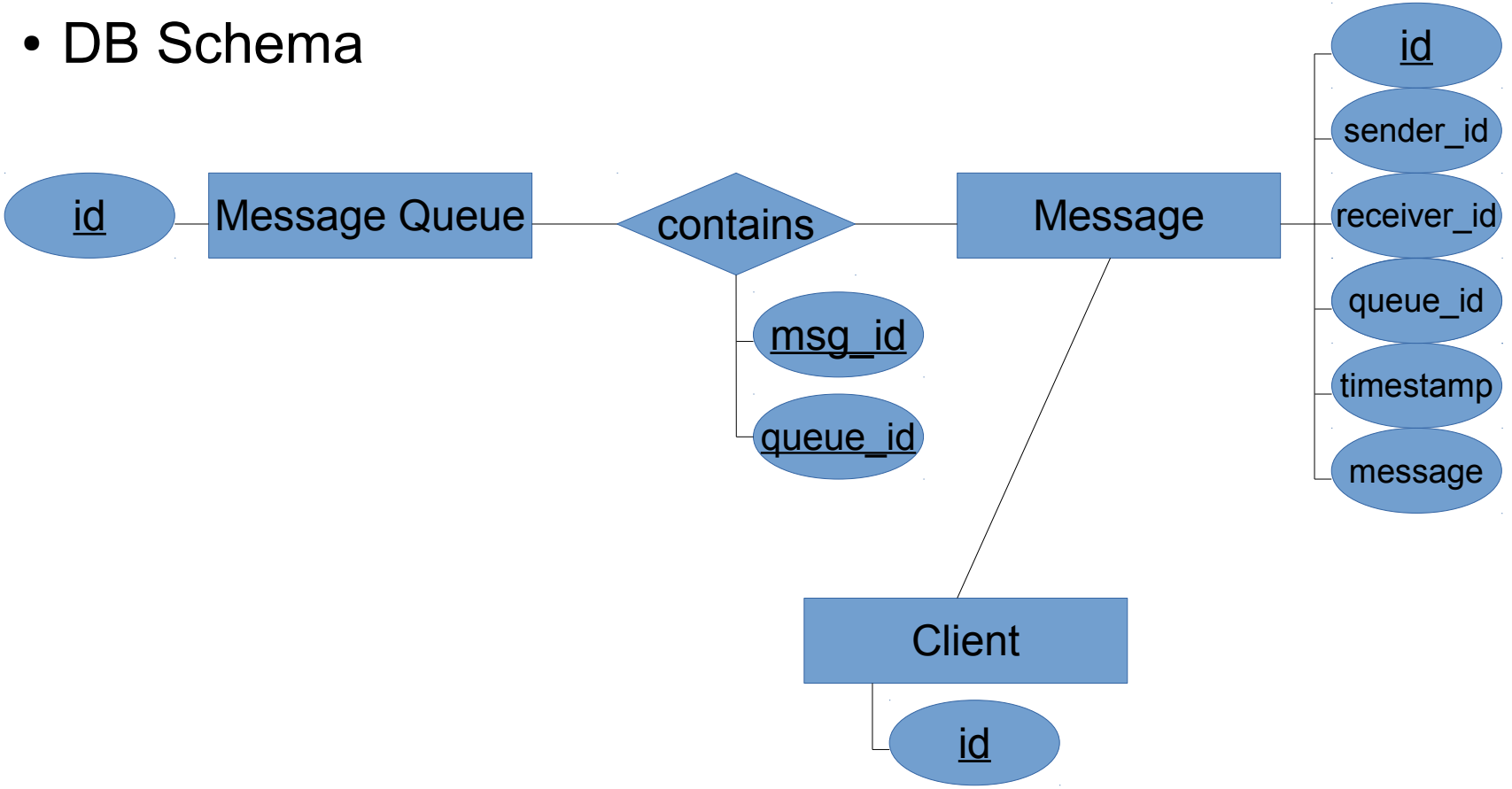
Middleware Server - Worker

- handles actual client request
- allow multiple runs to prevent *switching* overhead *but*: counter to prevent starvation

```
// Single worker thread
{
   acknowledgeClientConnection();
   receiveRequest();
   readDB() || updateDB();
   acknowledgeClientCompletion();
   while(counter < 5) {
     checkForFurtherRequests();
   }
}
```

Database

- DB Schema



- contains table to allow more parallelism (?)
- use stored procedures