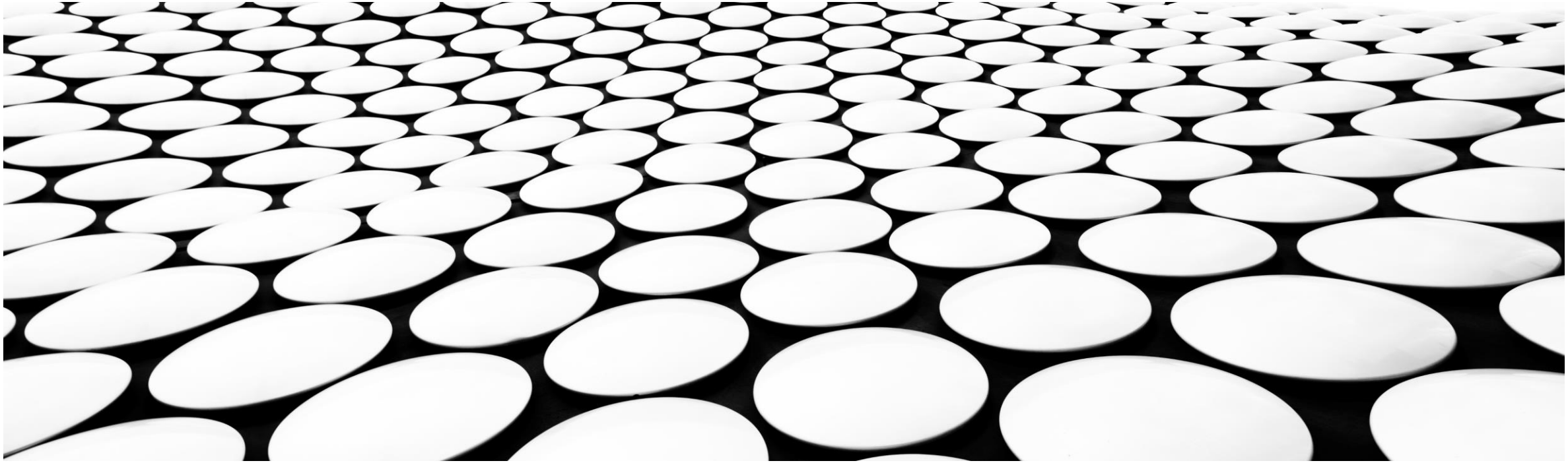
Interprocess Communication

COEN-317: Distributed Systems
Robert Bruce
Department of Computer Science and Engineering
Santa Clara University



Why care about Interprocess Communication?

Interprocess Communication (IPC) impacts distributed systems:

- IPC establishes a means of sharing data between processes.
- IPC enables process synchronization.

Two models for Interprocess Communication

1. **Remote Procedure Call: RPC** [1]
   - Allows an application to run on a remote machine while returning the output to a local machine [2].
2. **Message-Oriented Middleware: MOM** [1]
   - A form of persistent asynchronous communication that is independent of the sending or receiving processes [3].

[1] p. 163, *Distributed Systems* (3rd edition) by Maarten van Steen and Andrew S. Tanenbaum.
[2] p. 174, *Distributed Systems* (3rd edition) by Maarten van Steen and Andrew S. Tanenbaum.
[3] p. 206, *Distributed Systems* (3rd edition) by Maarten van Steen and Andrew S. Tanenbaum.

Remote Procedure Call: RPC

Remote Procedure Call:
1. Machine A executes a process on machine B [1].
2. Machine A suspends operations until machine B completes the process [1].
3. Machine A receives results from machine B once the process is completed [1].

What is the advantage with Remote Procedure Call?
- Simplicity: the communication layer is hidden between machine A and machine B [3]

What are the disadvantages with Remote Procedure Call?
- Reliability: if machine A or machine B fails, the entire RPC fails [1].
- Inefficiency: machine A is suspended while waiting for machine B to execute the process [2].

[1] p. 174, *Distributed Systems* (3rd edition) by Maarten van Steen and Andrew S. Tanenbaum.
[2] p. 185, *Distributed Systems* (3rd edition) by Maarten van Steen and Andrew S. Tanenbaum.
[3] p. 193, *Distributed Systems* (3rd edition) by Maarten van Steen and Andrew S. Tanenbaum.

Communication via sockets

Sockets:

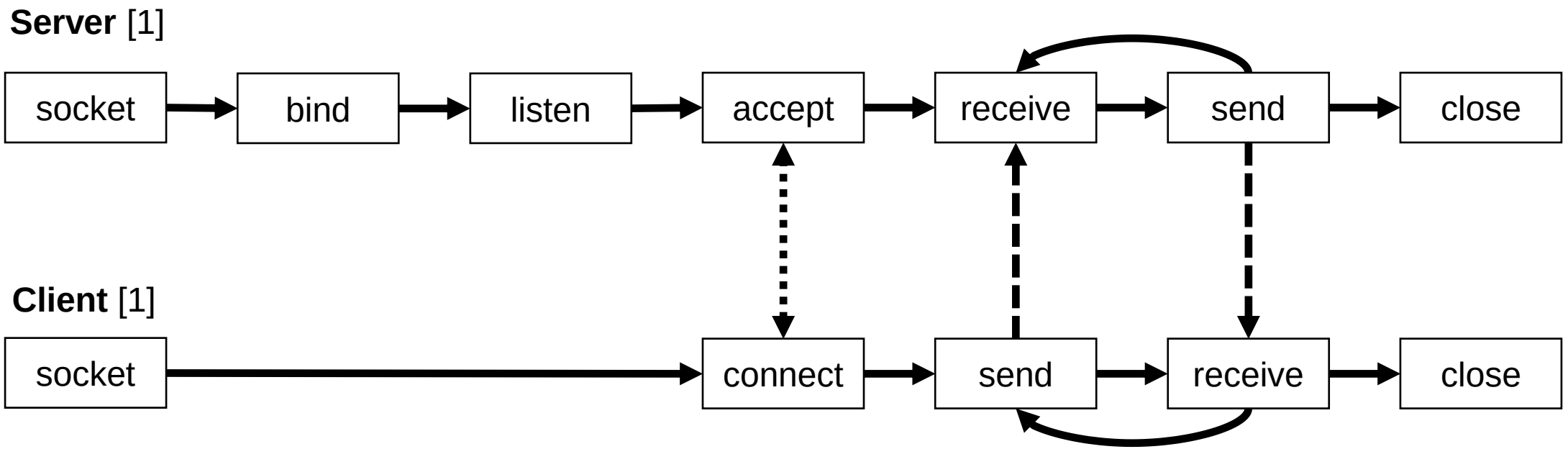
- Enables the sending and receiving of data between two machines via a port [1].
- Created by Berkeley Unix in 1970s [1].
- Sockets are integrated into the POSIX standard [1].

Socket operations:

- *socket***:** "creates a new communication end point" [2].
- *bind***:** "Attach a local address to a socket" [2].
- *listen*: "Tell operating system what the maximum number of pending connection requests should be" [2].
- *accept*: "Block caller until a connection request arrives" [2].
- *connect*: "Actively attempt to establish a connection" [2].
- *send*: "Send some data over the connection" [2]
- *receive*: "Receive soem data over the connection" [2]
- *close*: "Release the connection" [2]

[1] p. 193, *Distributed Systems* (3rd edition) by Maarten van Steen and Andrew S. Tanenbaum.
[2] p. 194, *Distributed Systems* (3rd edition) by Maarten van Steen and Andrew S. Tanenbaum.

Communication between two machines via socket



[1] p. 195, *Distributed Systems* (3rd edition) by Maarten van Steen and Andrew S. Tanenbaum.

Socket communication via ZeroMQ

ZeroMQ is a socket-based library and framework.

Advantages of ZeroMQ over Berkeley sockets:

- ZeroMQ simplifies communication by establishing types of socket-pair communication [1]:

  1.request-reply

  2.publish-subscribe

  3.pipeline
- ZeroMQ supports many-to-one communication as opposed to one-to-one with Berkeley sockets [1].
- ZeroMQ supports one-to-many communication (multi-casting) [1].
- ZeroMQ is asynchronous [1].

  Machine A can send a message before a recipient machine B has established a connection to machine A.

[1] p. 199, *Distributed Systems* (3rd edition) by Maarten van Steen and Andrew S. Tanenbaum.

Socket communication via ZeroMQ

ZeroMQ features three communication models:

- *request-reply*: "used in traditional client-server communication" [1]
- *publish-subscribe*: "clients subscribe to specific messages published by servers" [2]
- *pipeline*: a push-pull model in which one or more clients (pull) read messages from a server (push) [3].

[1] p. 200, *Distributed Systems* (3rd edition) by Maarten van Steen and Andrew S. Tanenbaum.
[2] p. 201, *Distributed Systems* (3rd edition) by Maarten van Steen and Andrew S. Tanenbaum.
[3] p. 202, *Distributed Systems* (3rd edition) by Maarten van Steen and Andrew S. Tanenbaum.

Message-Oriented Middleware: MOM

Message-oriented middleware: A persistent, asynchronous form of communication [1].

- Neither the sender nor receiver must wait for a message to be delivered [1].
- Messages are reliably sent and temporary stored for later retrieval by the recipient [1].
- There is no guarantee the recipient will read the message [1].

[1] p. 206, *Distributed Systems* (3rd edition) by Maarten van Steen and Andrew S. Tanenbaum.

Message-Oriented Middleware: MOM

Messages-Oriented Middleware are stored in a message queue.

The following commands enable management of a message queue:

- *put*: "Append a message to a specified queue" [1].
- *get*: "Block until the specified queue is nonempty, and remove the first message" [1].
- *poll*: "Check a specified queue for messages, and remove the first. Never block" [1].
- *notify*: "Install a handler to be called when a message is put into the specified queue" [1].

[1] p. 208, *Distributed Systems* (3rd edition) by Maarten van Steen and Andrew S. Tanenbaum.

Message-Oriented Middleware: examples

WebSphere MQ (message queue) [1]:

- Developed by IBM.
- Comprised of: queue manager, message channel agent, and message channels.

AMQP (Advanced Message Queuing Protocol) [1]:

- An open-standard for message queuing systems [1].
- Comprised of: "applications, queue managers, and queues" [1].
- Messaging utilizes a consumer-producer model [2].
- Supports persistent messaging [2].
- Implemented through RabbitMQ and Apache's Qpid [1].

[1] p. 218, *Distributed Systems* (3rd edition) by Maarten van Steen and Andrew S. Tanenbaum.
[2] p. 220, *Distributed Systems* (3rd edition) by Maarten van Steen and Andrew S. Tanenbaum.

Network topology and multicast communication

Broadcasting: sending messages to every node in a network.

Multicasting: sending messages to only a subset of a network.

Tree-based network:

- Organize the network nodes in a hierarchical tree network [1].
- Some nodes rely on other nodes to forward messages through the hierarchy [1].
- This network topology can lead to failure in messaging.

Mesh-based network :

- Organize the network nodes in a non-hierarchical mesh network [2].
- Redundant network connections results in more than one path between nodes [2].
- This network topology offers overlap between network connections and tends to be resilient when network nodes fail [2].

[1] pp. 221-222, *Distributed Systems* (3rd edition) by Maarten van Steen and Andrew S. Tanenbaum.
[2] p. 222, *Distributed Systems* (3rd edition) by Maarten van Steen and Andrew S. Tanenbaum.

For further reading...

1. *Unix Network Programming, Volume 2: Interprocess Communications* (second edition) by W. Richard Stevens

2. ZeroMQ
https://zeromq.org/

3. Advanced Message Queuing Protocol
https://www.amqp.org/

4. Rabbit MQ
https://www.rabbitmq.com/