CSCI 5105

Instructor: Abhishek Chandra

Today

- Message-oriented communication

Message-oriented Communication

- Allows various combinations of persistence and synchronization
- Different protocols implement different combinations
  - ZeroMQ, MPI: Transient Communication
  - Message-queuing: Persistent communication

Background: TCP/IP Sockets

- Provide low-level primitives:
  - listen, connect, accept, send/recv, read/write
- Problems?

ZeroMQ

- Supports transient communication
  - Built on top of TCP/IP
- Provides higher level communication patterns
  - Common in many distributed programs
  - Makes it easy for programmers
- Enables multiple forms of communication
  - One-to-one, one-to-many, many-to-one

5

ZeroMQ Messaging Patterns

- Request-reply:
  - Client sets up a request type socket
  - Server sets up a reply type socket
  - Similar to RPC
- Publish-subscribe:
  - Server sets up pub type socket
  - Client sets up sub type socket
  - Only matching messages delivered to clients
  - Provides multicast capability

6

ZeroMQ Messaging Patterns (Contd.)

- Pipeline:
  - Servers push out messages to a pipeline
  - Clients pull from the pipeline
  - Any server can push, any client can pull a message
  - Can be used to distribute messages to multiple clients
  - Similar to producers-consumers

7

Message-Passing Interface (MPI)

- Sockets designed for network communication (e.g., TCP/IP)
  - Support simple send/receive primitives
  - General-purpose
- MPPs and COWs require
  - Advanced primitives
  - Other forms of buffering, synchronization
- Large number of incompatible proprietary libraries and protocols
  - Need for a standard interface

8

Message-Passing Interface (MPI)

- Message-passing interface (MPI)
  - Hardware independent
  - Designed for parallel applications
- Communication between groups of processes
  - Each endpoint is a (groupID, processID) pair
- Messaging primitives support different forms of transient communication
  - MPI_bsend: transient asynchronous
  - MPI_ssend: delivery-based transient synchronous

9

Message-Queuing Model

- Support asynchronous persistent communication
  - Intermediate storage for message while sender/receiver are inactive
- Communicate by inserting messages in queues
  - Applications can have private queues, or shared queues
- Loosely coupled communication
  - Sender is only guaranteed that message will be eventually inserted in recipient's queue
  - No guarantees on when or if the message will be read
- Examples: IBM Websphere MQ, MS Message Queuing, Amazon Simple Queue Service

10

Message-Queueing: Mechanics

- Addressing:
  - Messages have to be addressed to destination queue
  - Need a system-wide address/name
- Message sizes:
  - Limited or allow fragmentation
- Basic interface:
  - put, get, poll, notify
- Callback functions: invoked upon message receipt

11

Message-Queuing Entities

- Queue managers: Handle message sending/receiving from queues
  - Storage, communication, notification
- Relays: Message routers
- Message brokers: App-level gateways
  - Can transform messages between formats
- Publish/Subscribe systems
  - Brokers match applications to messages

12