Hwajung Lee

ITEC452 Distributed Computing

Lecture 7 Interprocess Communication (IPC): An Overview

Introduction to IPC (1)

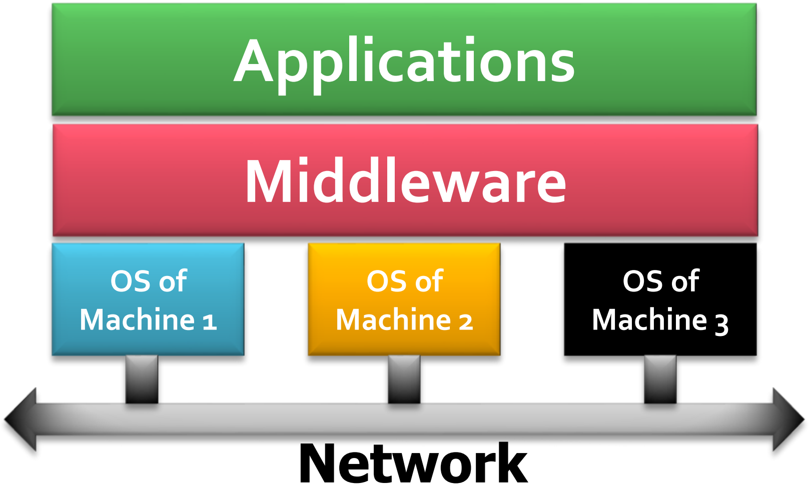
- Interprocess Communication (**IPC**) is **at the heart of distributed computing**.
- Processes and Threads
  - ***Process*** is the execution of a program
  - ***Threads*** are lightweight processes
    - Like a process, each thread maintains a separate flow of control, but threads share a common address space

Introduction to IPC (2)

- **Client-Server Model**
  - a widely accepted model for designing distributed system
  - Example: a search engine like Google®

Introduction to IPC (3)

- Middleware
  - Processes, processors, and objects may be scattered anywhere in a network.
  - From developing distributed applications, transparency is a desirable property.
  - The layer of software that makes it possible is called ***middleware***.



Introduction to IPC (4)

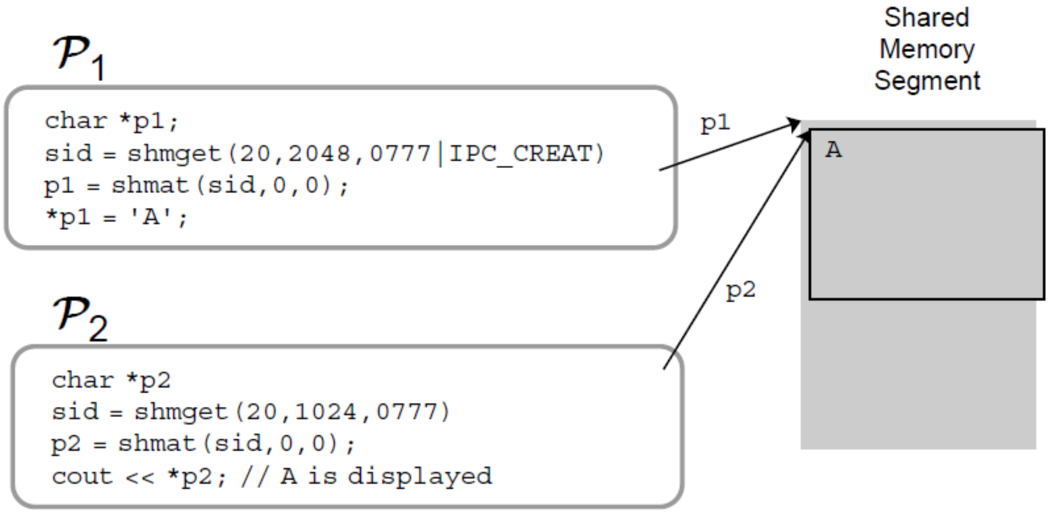
- Some important middleware services address the following issues:
  - How does a process locate another named process or object anywhere on the Internet?
  - How does a process in the application layer communicate with another process anywhere on the Internet?
  - How to isolate the application programs from differences in programming languages and communication protocols?
  - How is the security of the communication guaranteed without any knowledge about the trustworthiness of the operating systems at the two endpoints?

Four General Approaches of IPC

- Shared Memory
- Messages
- Pipes
- Sockets

Shared Memory

- Two or several processes can map a segment of their virtual space into an identical segment of physical memory.
  - Shared memory is ***the most efficient way*** of IPC
  - But it may ***require synchronization***



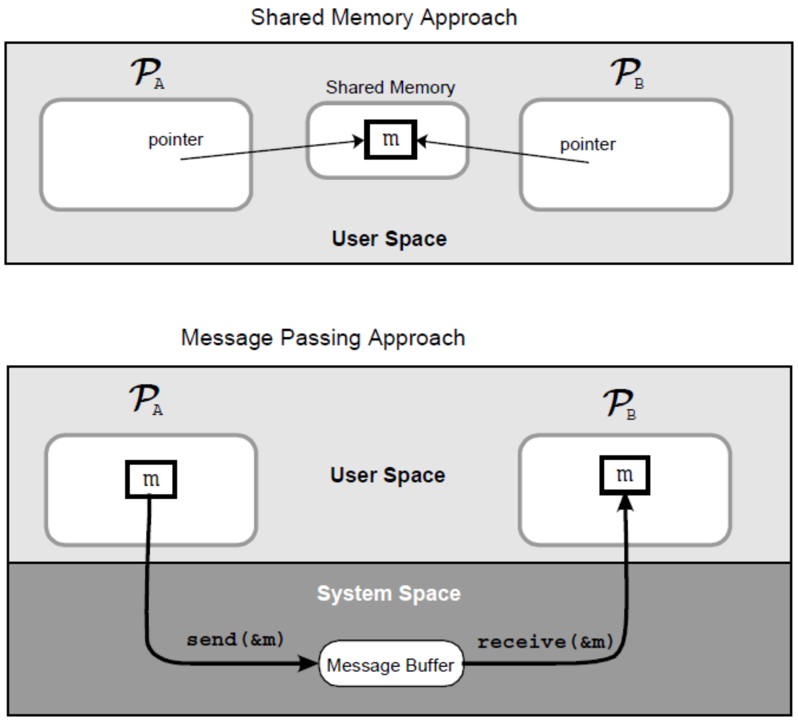
```
char *p1;
sid = shmget(20,2048,0777|IPC_CREAT)
p1 = shmat(sid,0,0);
*p1 = 'A';
```

```
char *p2
sid = shmget(20,1024,0777)
p2 = shmat(sid,0,0);
cout << *p2; // A is displayed
```

Image Source: http://medusa.sdsu.edu/cs570/Lectures/Chapter9.pdf

Messages (1)

- Most distributed applications are implemented using message passing.
  - Messages are ***less efficient*** than shared memory (***require buffering and synchronization***), but sometimes are more suitable due to the built-in synchronization



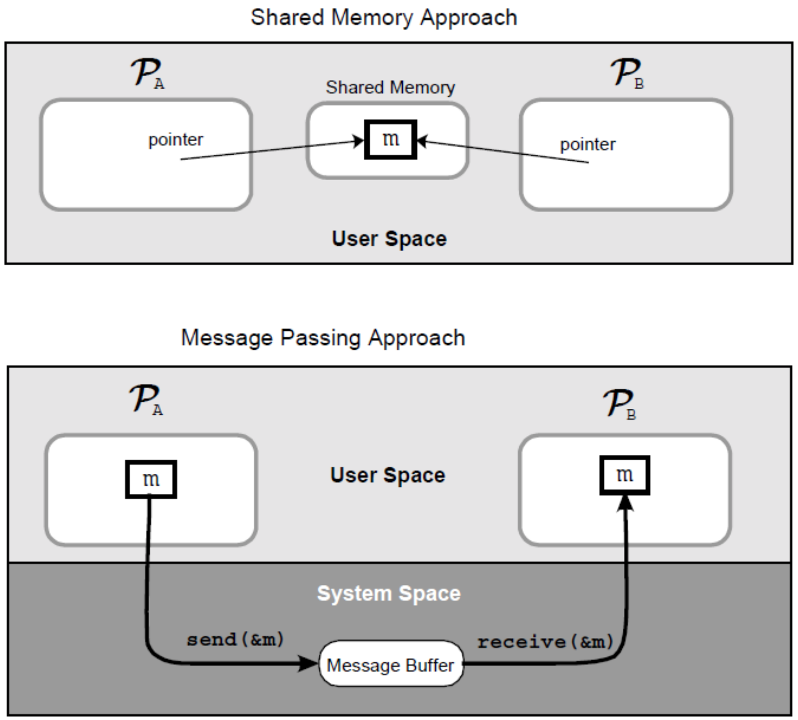
Shared Memory Approach

Message Passing Approach

Image Source: http://medusa.sdsu.edu/cs570/Lectures/Chapter9.pdf

Messages (2)

- The messaging layer is logically located just above the TCP/IP or the UDP/IP layer, but below the application layer.
- The implementation of sockets at the TCP or the UDP layer helps processes address one another using specific socket addresses.


Shared Memory Approach

Message Passing Approach

Image Source: http://medusa.sdsu.edu/cs570/Lectures/Chapter9.pdf

Messages (3)

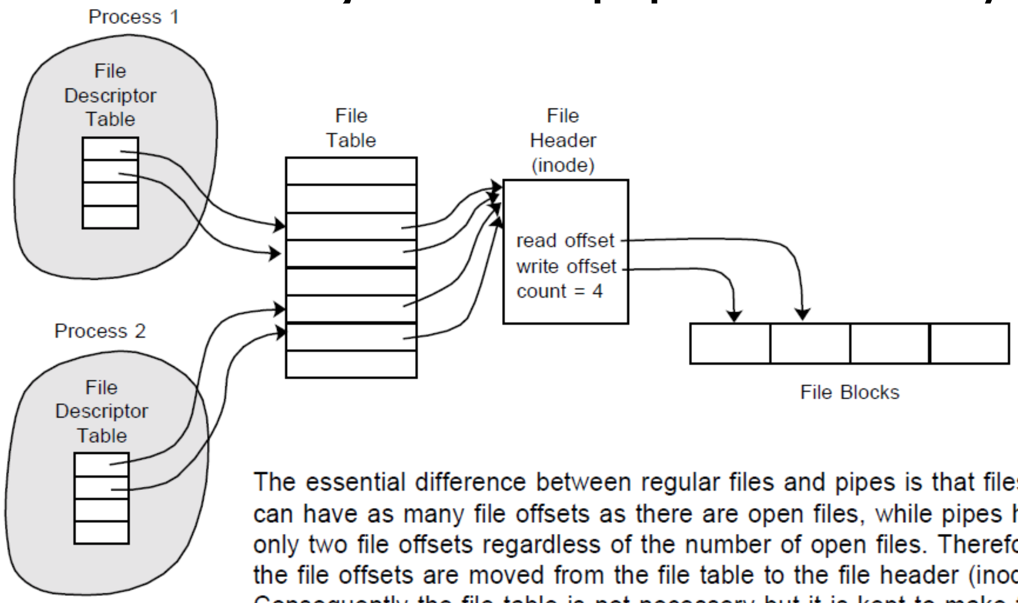
- Two Types of Messages
  - Transient Messages
    - A message is lost unless the receiver is active at the time of the message delivery and retrieves it during the life of the application
  - Persistent Messages
    - Messages are not lost, but saved in a buffer at the time of message delivery

Messages (4)

- Stream
  - Sequence of data items.
  - Communication using streams ***requires a connection establishment*** between a sender of a receiver

Pipes (1)

- Pipes are implemented in file system.
  - Pipes are basically files with only two file offsets: one for reading, another for writing.
  - Writing to a pipe and reading from a pipe is strictly in FIFO manner.



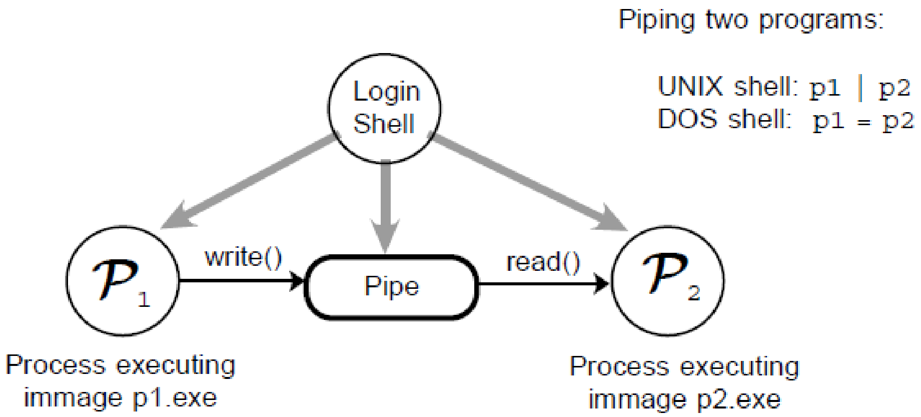
The essential difference between regular files and pipes is that files can have as many file offsets as there are open files, while pipes have only two file offsets regardless of the number of open files. Therefore, the file offsets are moved from the file table to the file header (inode). Consequently the file table is not necessery but it is kept to make file access consistent with all types of files.

Image Source: http://medusa.sdsu.edu/cs570/Lectures/Chapter9.pdf

cat file > my_pipe

Pipes (2)
Two Types of Pipes

ANONYMOUS (UNNAMED) PIPES


Login Shell
$\mathcal{P}_1$
write()
Pipe
read()
$\mathcal{P}_2$
Process executing immage p1.exe
Process executing immage p2.exe

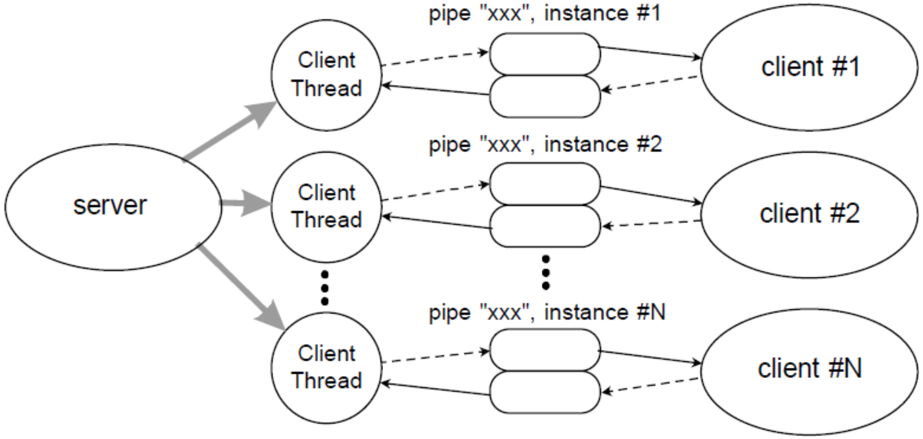
Piping two programs:

```
UNIX shell: p1 | p2
DOS shell:  p1 = p2
```

Unidirectional

To communicate two related processes in both directions, two anonymous pipes must be created.

NAMED PIPES



Full-duplex or half-duplex

Image Source: http://medusa.sdsu.edu/cs570/Lectures/Chapter9.pdf

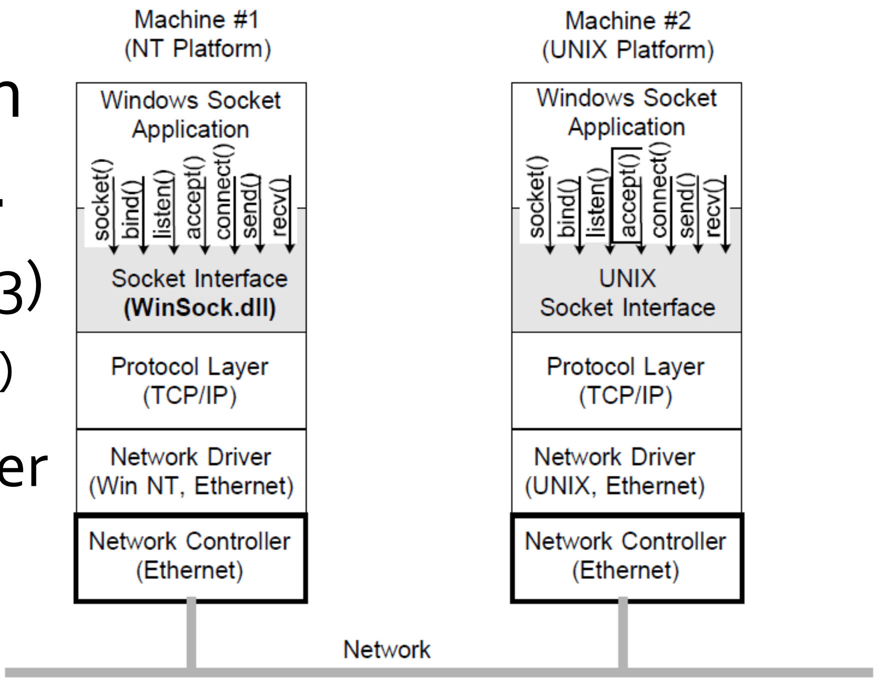
For example, one can create a pipe and set up gzip to compress things piped to it:

```
mkfifo my_pipe
gzip -9 -c < my_pipe > out.gz &
cat file > my_pipe
rm my_pipe
```

Sockets (1)

- Sockets are abstract endpoints of communication between a pair of processes.
  - originally used in BSD 4.2 (1983)

    Note: BSD (Berkeley Software Distribution)
  - by now widely accepted in other operating systems.
  - typically used in server/client communication.



TCP - Transmission Control Protocol
IP - Internet Protocol

Image Source: http://medusa.sdsu.edu/cs570/Lectures/Chapter9.pdf

Sockets (2)

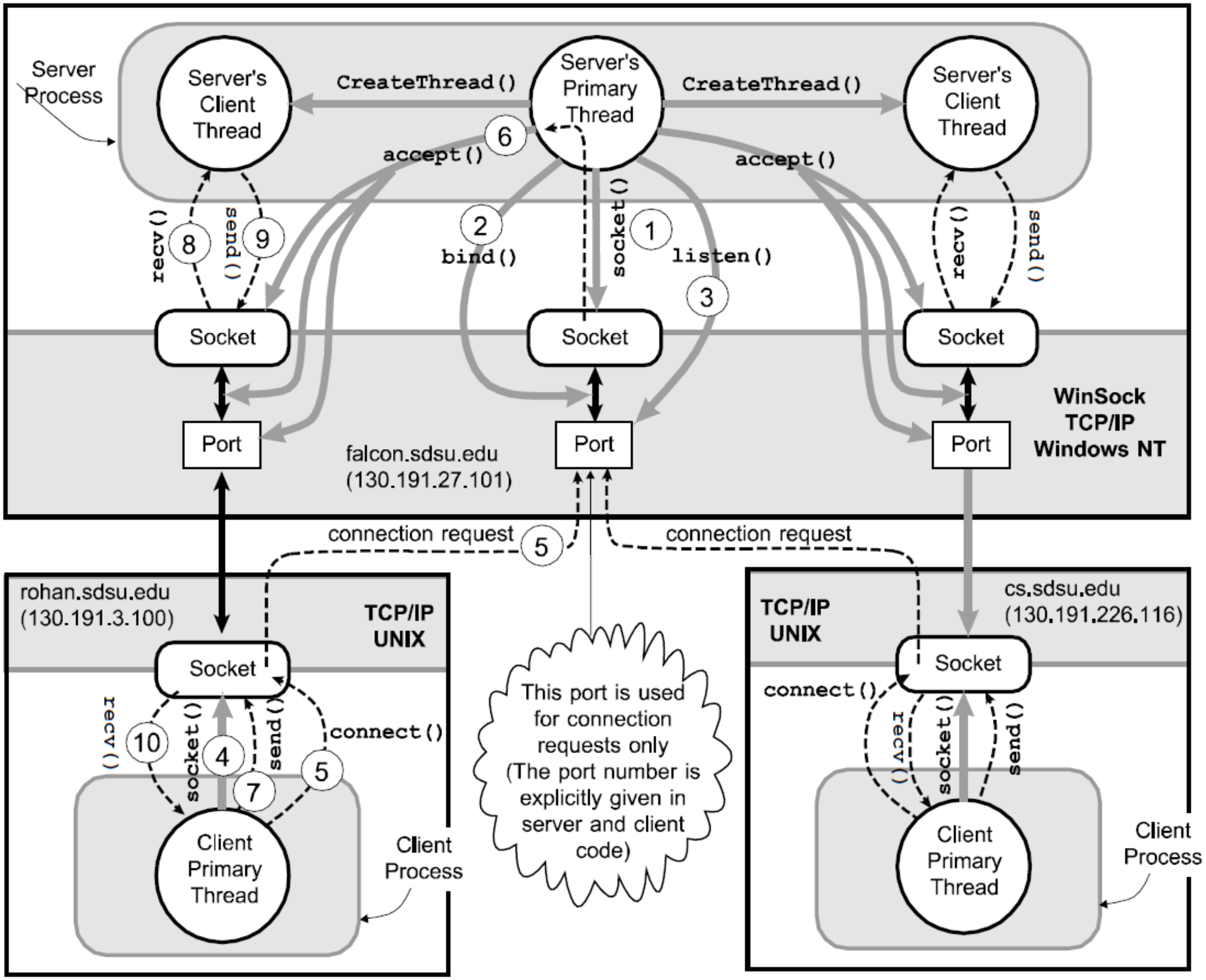


Image Source: http://medusa.sdsu.edu/cs570/Lectures/Chapter9.pdf

More IPCs (1)

- Remote Procedure Call (RPC)
  - A procedure call that helps a client communicate with a server running on a different machine that may belong to a different network and a different administrative domain
  - The task of packing the parameters of the call into a message is called ***parameter marshalling***.
- Remote Method Invocation (RMI)
  - A generalization of RPC in an object-oriented environment.

More IPCs (2)

- Web Services
  - Most web services are based on XML that is widely used for cross-platform data communication including:
    - Simple Object Access Protocol (SOAP)
      - SOAP allows a one-way message containing a structured data to be sent from one process to another using any transport protocol like TCP.

More IPCs (3)

- Web Service Description Language (WSDL)
  - It describes the public interface to the web service.
  - An XML-based service description that explains how a client should communicate using the web service
- Universal Description Discovery, and Integration specification (UDDI)
- Java Web Services

More IPCs (4)

- Event Notification
  - Event notification systems help establish a form of asynchronous communication among distributed objects on heterogeneous platforms.
  - **Jini®,** also called **Apache River**, a product of Sun Microsystems, provides event notification service for Java-based platforms. It allows subscribers in one JVM to receive notification of events of interest from another JVM.

More IPCs (5)

- Common Object Request Broker Architecture (CORBA)
  - The framework of a middleware that enables clients to transparently access remote objects across a distributed computing platform, regardless of the machines on which they are running or the language in which they are written.
    - Its specifications were drown by the **Object Management Group (OMG)** consisting of some 700 companies.
    - The core of CORBA is the **Object Request Broker (ORB)**.

More IPCs (6)

- Mobile Agents
  - A piece of code that migrates from one machine to another.
  - The code, which is an executable program, is called a ***script***
  - Agents carry data values or procedure arguments or results across machines
  - The use of an interpretable language like **Tcl** makes it easy to support mobile agent based communication on heterogeneous platforms.

More IPCs (7)

- Basic Group Communication Services
  - With the rapid growth of the World Wide Web and electronic commerce, group oriented activities have substantially increased in recent years.
  - Multicasts are useful in the implementation of specific group service.