CS 31: Intro to Systems
Networked Hangman

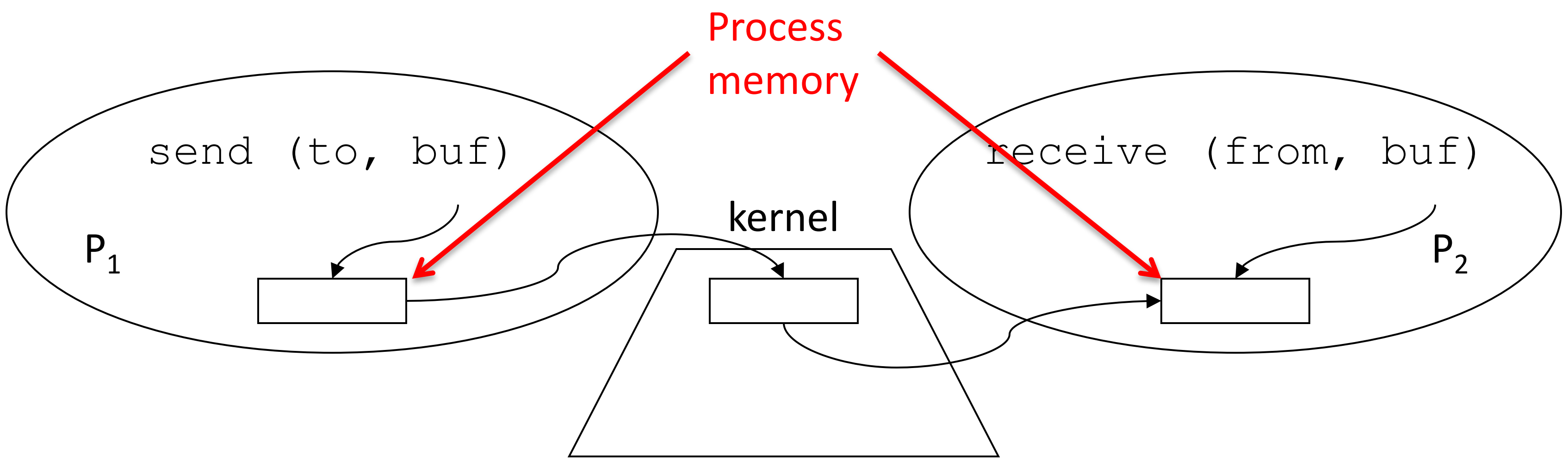
Kevin Webb
Swarthmore College

May 28, 2016

Agenda

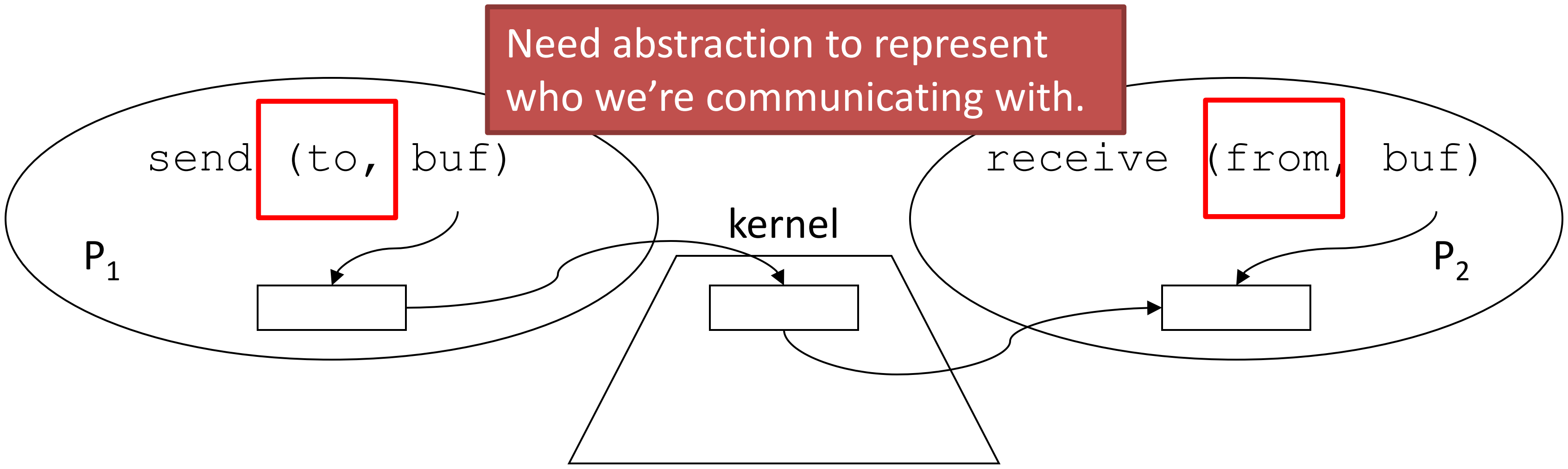
- Brief overview of network abstractions
- An example protocol: hangman game
- Try writing our own network code (Python)

Message Passing (local)



- Operating system mechanism for IPC
  - `send (destination, message_buffer)`
  - `receive (source, message_buffer)`
- Data transfer: in to and out of kernel message buffers

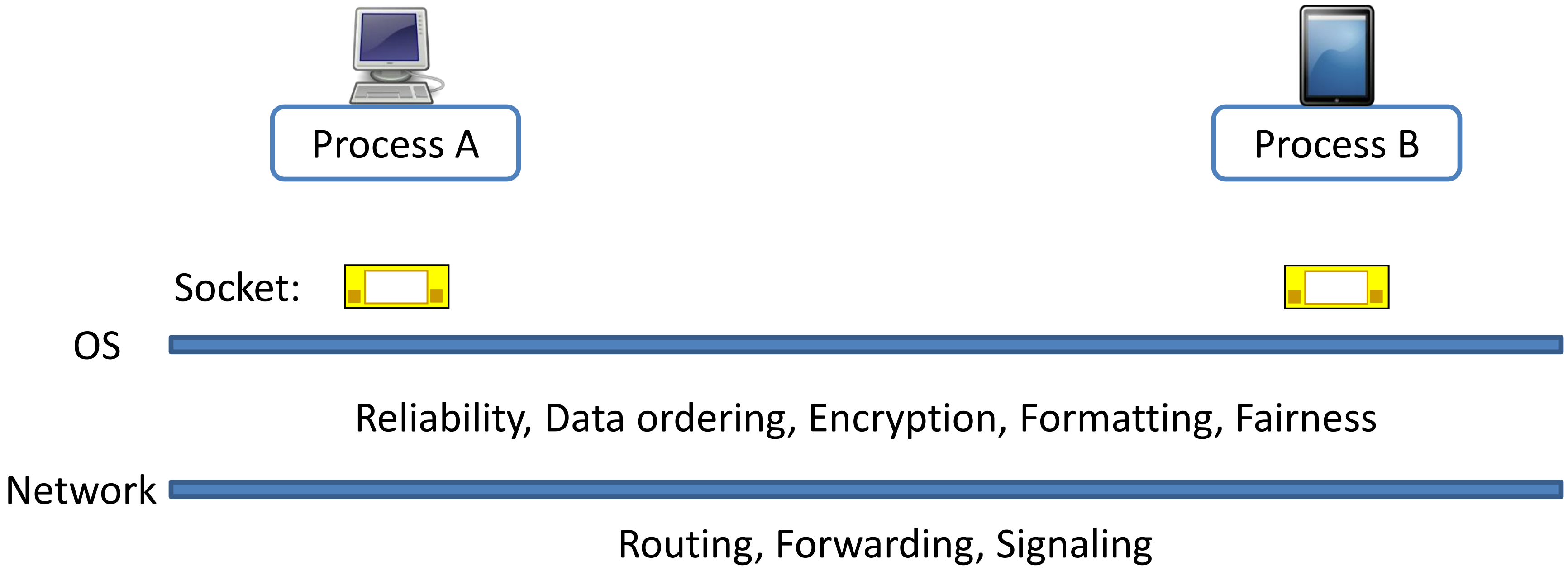
Message Passing (local)



- Operating system mechanism for IPC
  - `send (destination, message_buffer)`
  - `receive (source, message_buffer)`
- Data transfer: in to and out of kernel message buffers

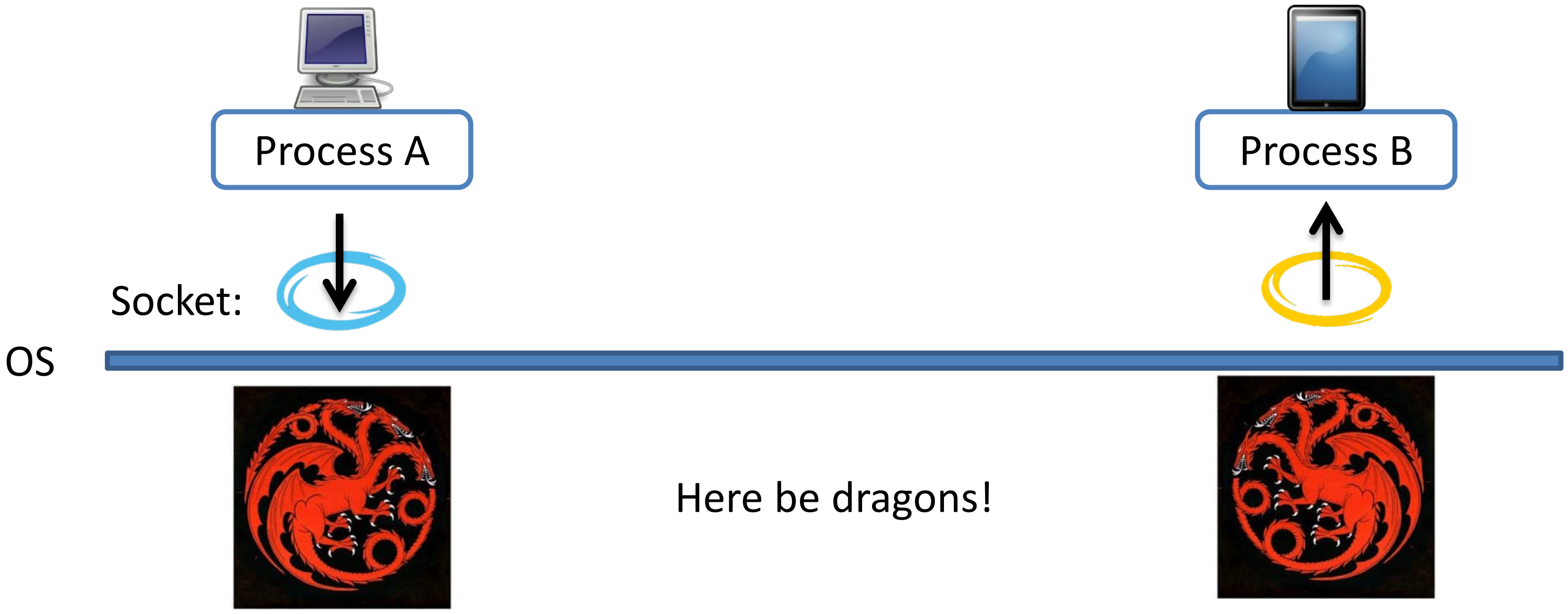
Sockets

- Socket: abstraction of communication endpoint
  - Provided by OS
  - Simple interface: send() / recv()

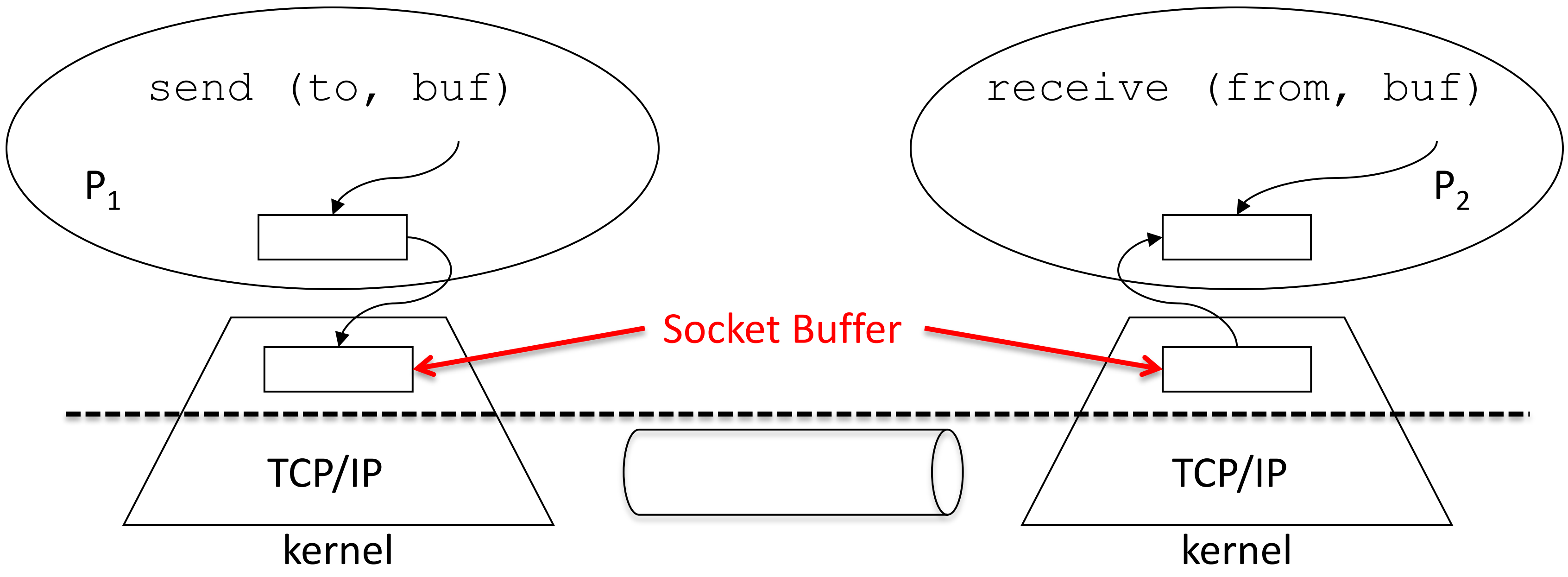

Process A
Process B
Socket:
OS
Reliability, Data ordering, Encryption, Formatting, Fairness
Network
Routing, Forwarding, Signaling

Sockets

- Socket: abstraction of communication endpoint
  - Provided by OS
  - Simple interface: send() / recv()



Message Passing (network)



- Same synchronization
- Data transfer
  - Copy to/from OS socket buffer
  - Extra step across network: hidden from applications

Questions

- Communication model: Who are the parties?
- Protocol: Who sends, who receives, and when?

Client / Server Model

- Server:
  - Opens a socket that accepts new connections
  - Waits for connection to come in
  - Creates a new socket for pair-wise communication over that new connection
  - Wait for connection to come in, repeat…
- Server is connected to by client
  - Web, file system, streaming music, game, etc.

Client / Server Model

- Client:
  - Opens a socket
  - Initiates connection to server
  - Communicates to server over socket

- Examples:
  - Firefox (web client)
  - Thunderbird (mail client)
  - What you’ll be making soon: hangman client

Protocol

- Rules for communication that dictate:
  - message format
  - whose turn it is to send, when to recv

- Example: HTTP

Hangman

Server

- Send categories (string terminated by \r\n)
- Repeat:
  - Send game status (string terminated by \r\n)

Client

- Connect to server
- Send greeting: “HELLO\r\n”
- Select a category: “CATEGORY N\r\n”
- Repeat:
  - Send letter guess: “GUESS N\r\n”

Try telnet

```
telnet sesame.cs.swarthmore.edu 9000
```

```
HELLO
CATEGORY 1
GUESS t
GUESS s
GUESS e
…
```

Note: telnet will automatically put in the \r\n when you press enter.

Writing a client

- Typing all these commands in telnet is a drag

- Connect to a CS lab machine
  - Starter code in ~kwebb/public/cs31/hangman-client.py
  - Starter code will read input from user
  - You need to add socket calls

- s = socket.socket(…) will create a socket
- Then you can call methods on that socket:
  - s.send(string_to_send)
  - string_received = s.recv()

Search online:
"python socket" to get
the documentation.

Connect to sesame.cs.swarthmore.edu on port 10000