CS61A Lecture 20

2011-07-25
Colleen Lewis

Goals for the day

- Introduction to some technical details about the internet
  - IP Addresses
  - Sockets
  - 3-way handshake
- Callbacks – call this function when X happens
- More practice with BIG programss

Internet Basics Videos

http://www.youtube.com/watch?v=q0XCaUkfNk

http://www.youtube.com/watch?v=qv0XCaUkfNk

Clicker poll 😊

When you send information through the internet, the information is broken up into multiple:

A) Packages
B) Packets
C) Pairs
D) Lists
E) Other

Clicker poll 😊

These direct your packets within the internet to help them find their way?

A) ISPs
B) Routers
C) Servers
D) Wires
E) More than one of the above

Clicker poll 😊

Webpages ARE:

A) Files on routers
B) Files on servers
C) Files on public computers
D) Servers
E) Routers
Lots of applications on your computer connect to the internet

- All have the same IP address
- They may use different ports (like apartment number)
- Each application creates a unique line of communication called a socket

Creating a line of communication (socket) is a lot like starting a cell phone conversation.

Our IM - Three way handshake

Hello

Welcome

Thanks

Our IM World

<table>
<thead>
<tr>
<th>im-client</th>
<th>im-server</th>
</tr>
</thead>
<tbody>
<tr>
<td>STk Sockets</td>
<td></td>
</tr>
<tr>
<td>Internet (TCP/IP)</td>
<td></td>
</tr>
</tbody>
</table>

3-way handshake

Problems with the 3-way handshake?
A) Wasteful – could just do a 2-way handshake
B) Not enough – need a 4-way handshake
C) The connect might stop working
D) Multiple answers above are correct
E) None of the above

Starting a Server

STk> (load "-cs61a/lib/im-server.scm")
okay
STk> (im-server-start)
Server starting...
Server IP address: 128.32.48.187, server port: 46990
(im-server-start) done. okay

A client connecting to a server

STk> (load "-cs61a/lib/im-client.scm")
okay
STk> (im-enroll "128.32.48.187" 46990)
Sending 'hello' request to server.
Waiting for 'welcome' from server.
Response received: (server cs61a-tf welcome ())
Received 'welcome' message.
Sending 'thanks'!
(im-enroll) done. okay

This created a socket
A client can send a message to another client

```
STk> (im 'cs61a-tf "hi - how are you?")
ok
STk> (im-exit)
```

What do you need to know?

- What is a socket?
  - An established line of communication
- What is a 3-way handshake?
  - Sequence of messages to set-up a socket
- Why do we need a 3-way handshake?
  - To ensure the communication line is bi-directional
- How do you run the code?

```
Server       Client
(im-server-start) (im-enroll "123.4.56.8" 333)
(im 'name "my message")
(im-server-close) (im-exit)
```

Servers and Clients Common Functionality

**Request ADT im-common.scm**

```
(define (make-request src dst action data)
  (list src dst action data))
(define (request-src req)
  (list-ref req 0))
(define (request-dst req)
  (list-ref req 1))
(define (request-action req)
  (list-ref req 2))
(define (request-data req)
  (list-ref req 3))
```

Clients and servers send requests

```
(define (send-request req write-port)
  ...)
(define (get-request read-port)
  ...)
```

Don't worry about how these work.
They work.

```
Im-enroll-psuedo
```

```
(define (im-enroll-psuedo server-address port)
  (set! socket-to-server
    (make-client-socket server-address port))
  (set! port-to-server
    (socket-output socket-to-server))
  (set! port-from-server
    (socket-input socket-to-server))

  (send-request
    (make-request 'colleen 'server 'hello "")
    port-to-server)

  (get-request port-from-server)
  ;; omitted - confirm it is the message 'welcome
  (send-request
    (make-request 'colleen 'server 'thanks "")
    port-to-server)
  (setup-request-handler port-from-server))
```

```
Im-enroll
```

```
(define (im-enroll server-address port)
  (set! socket-to-server ...
  (set! port-to-server ...
  (set! port-from-server ...

Based upon the code from the previous slide, the 3 variables that were set with set! must be:

A) Shared by the client and server
B) Defined in the client code
C) Defined in the server code
D) Neither – these must be NEW variable
E) Nothing can be inferred
request-handler is called
when a new message is available

(define (request-handler-psuedo)
  (let ((req (get-request port-from-server)))
    (let ((action (request-action req))
           (message (request-data req)))
      (cond
       ((equal? 'receive-msg action)
        (display message))
       ((equal? 'goodbye action)
        ...))))

when-port-readable

(define (setup-request-handler-psuedo port-from-server)
  (when-port-readable ; spec. form
    request-handler))

Telling the hardware how to let
you know there is a new request

- (im-server-start)
  - Establishes client-request-handler to handle requests
- (im-enroll "123.4.56.8" 333)
  - Initiates 3-way handshake
  - Establishes request-handler to handle requests

These are called callbacks.
Meaning – here’s a function, call it when X happens

im-enroll-psuedo

(define (im-enroll-psuedo server-address port)
  (set! socket-to-server
    (make-client-socket server-address port))
  (set! port-to-server (socket-output socket-to-server))
  (set! port-from-server (socket-input socket-to-server))
  (send-request
    (make-request 'colleen 'server 'hello **)
    port-to-server)
  (get-request port-from-server)
  ;; omitted - confirm it is the message 'welcome
  (send-request
    (make-request 'colleen 'server 'thanks **)
    port-to-server)
  (setup-request-handler port-from-server)

when-port-readable

- What function in the server do you think calls
  when-port-readable?
A. (im-server-start)
B. (im-server-close)
C. (handshake sock)
D. (register-client name sock)
E. All of the above

im

(define (im-psuedo who message)
  (send-request
    (make-request 'colleen who 'send-msg message)
    port-to-server))