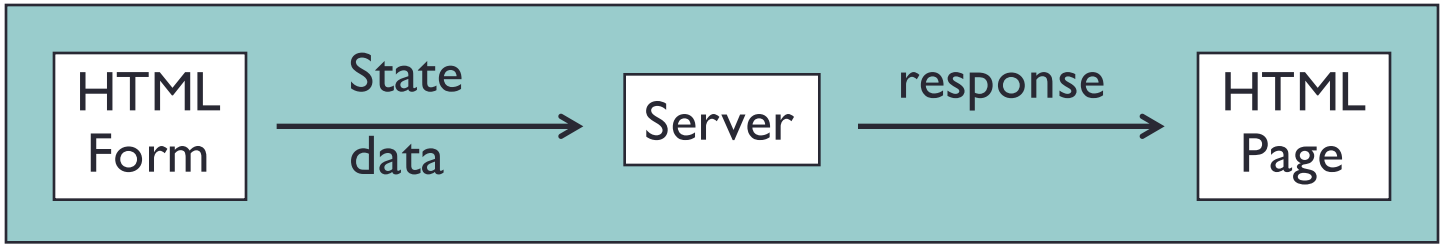
State Handling with Sessions

CS 4640
Programming Languages for Web Applications

[Based in part on SWE 432 and SWE 632 materials by Jeff Offutt, GMU]
[Robert W. Sebesta, "Programming the World Wide Web]

Session State Information

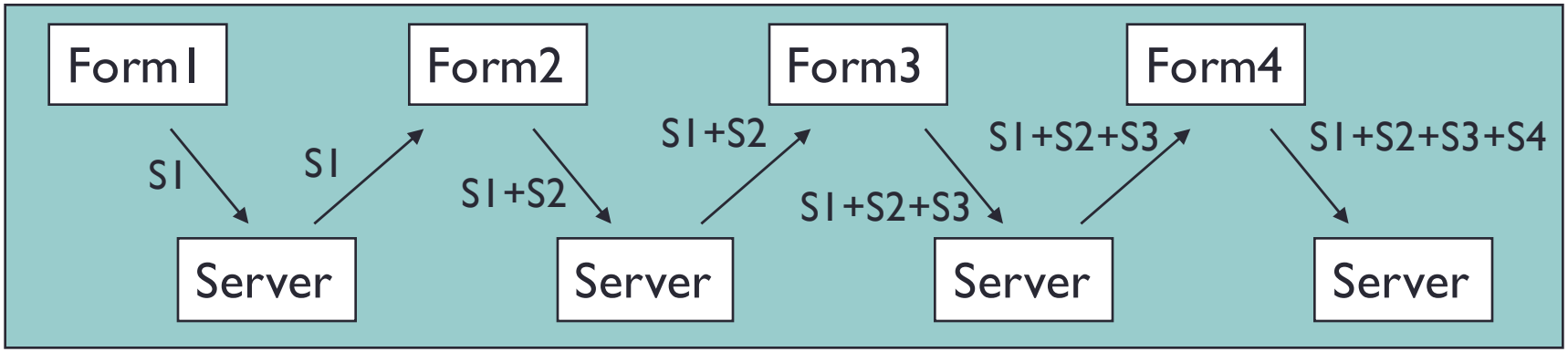
The initial versions of the web suffered from a lack of state



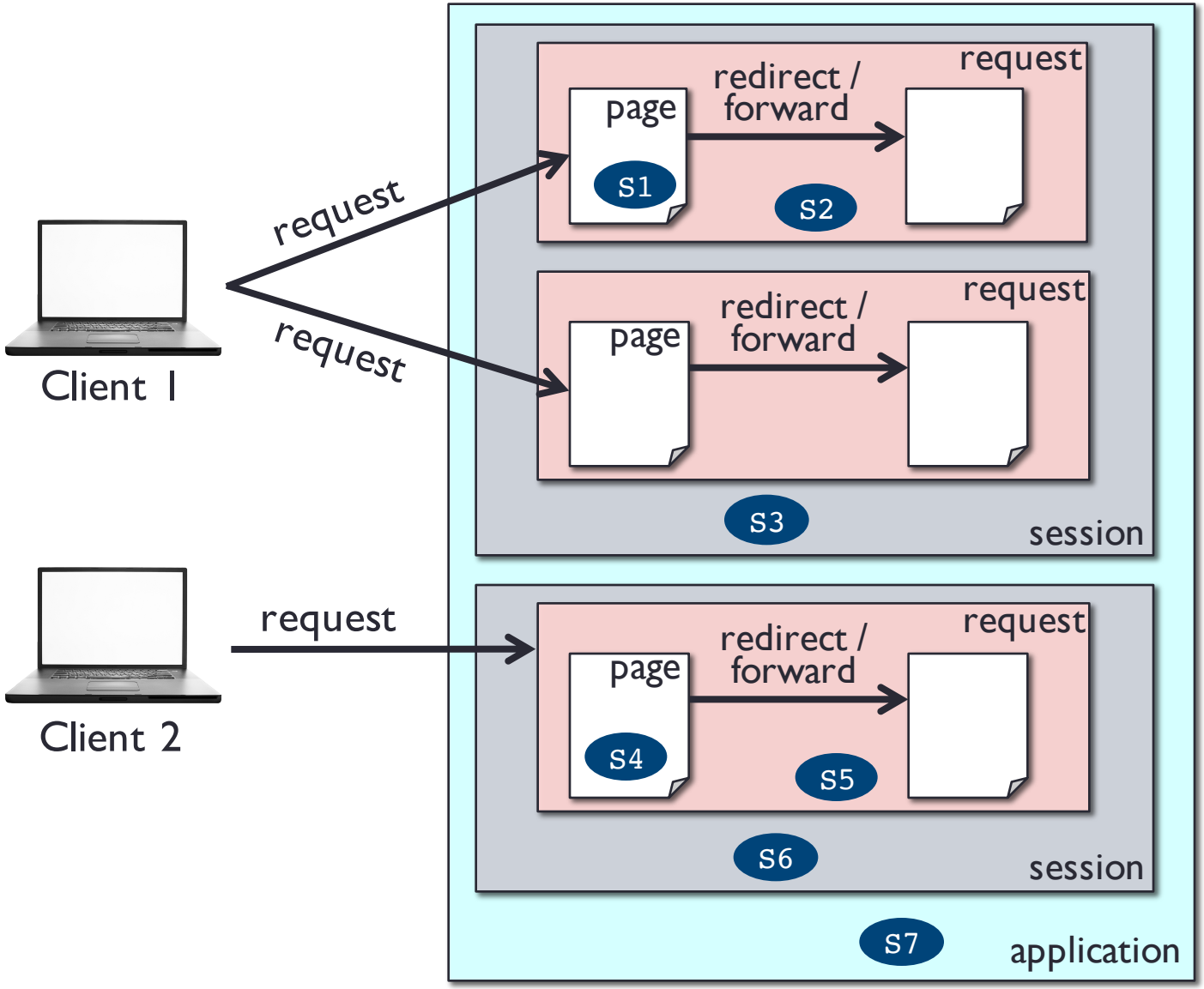
If a web app needed multiple screens, there was no way for state to be accumulated or stored

- This is due to the stateless property of HTTP

In reality, we want to keep track of the information (i.e., state)



Scoping in Web App (in general)



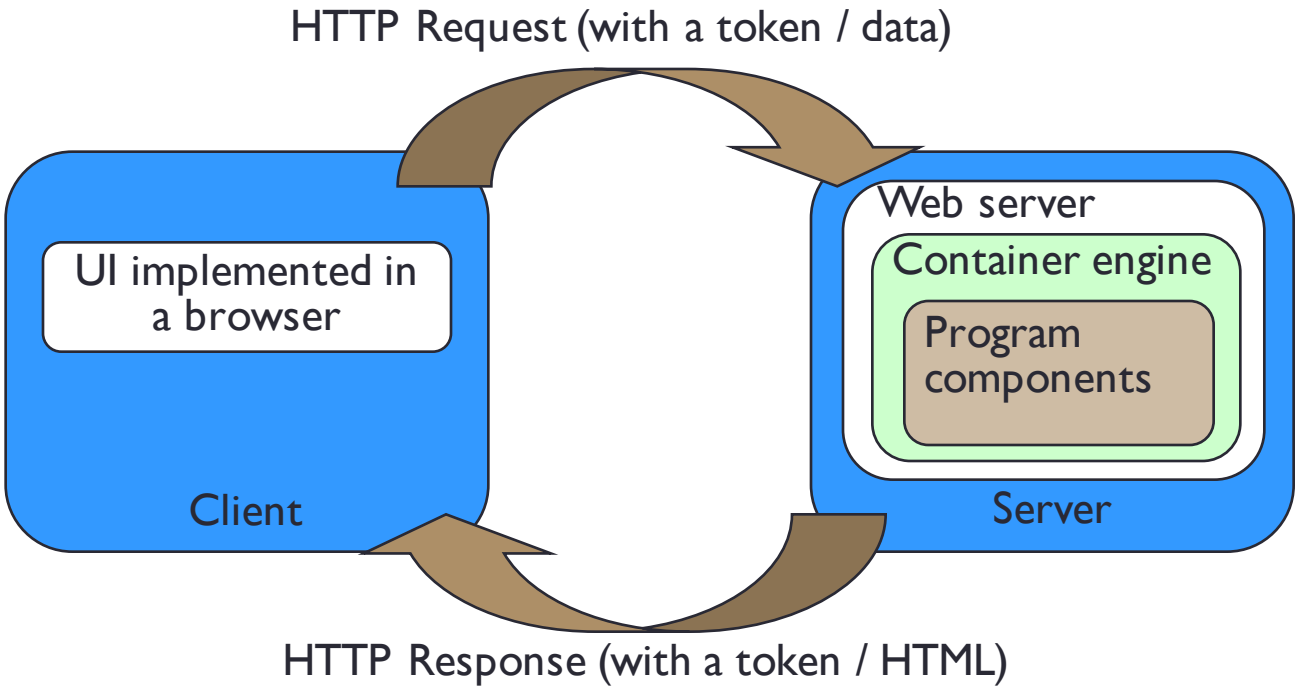
Assume `Si` is a variable (or object) storing the state (i.e., current data)

revisit

Session Tracking Methods

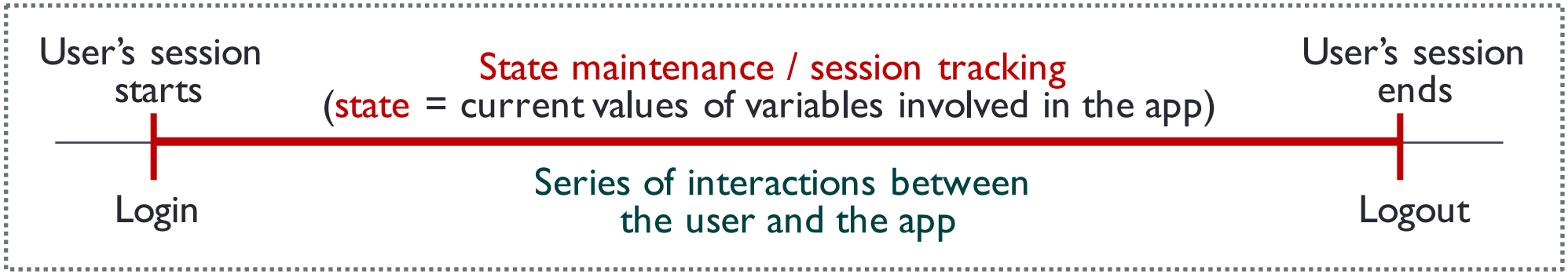
- URL rewriting – include data as extra parameters
- Hidden form fields
- Cookies – client-side
- Server-side session objects

All four methods work by exchanging a token between the client and server



Sessions

- A single coherent use of the system by the same user



- All web components sharing a session must start session to join the existing session (or create a new session if none available)
- `$_SESSION` global array stores data in the current active object
- Data disappear when the `$_SESSION` object is destroyed
- The session object is destroyed after the session ends, usually 30 minutes after the last request

Session Definition

A session is defined by

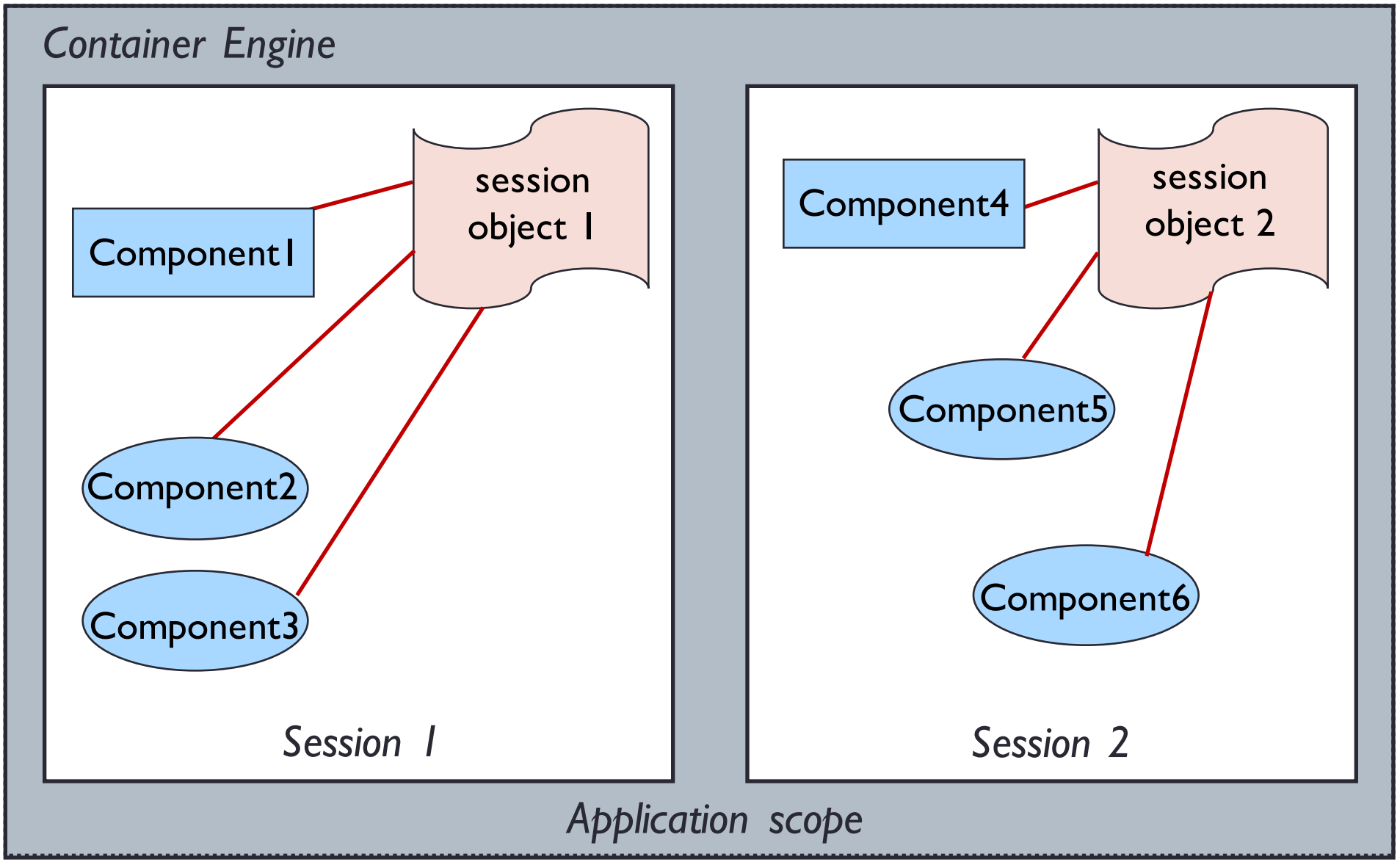
1. The web server
    - PHP container
2. The client
    - IP address
    - Browser

- Session objects are kept on the server
- Each session object uses different parts of memory (instances of data values) on the server

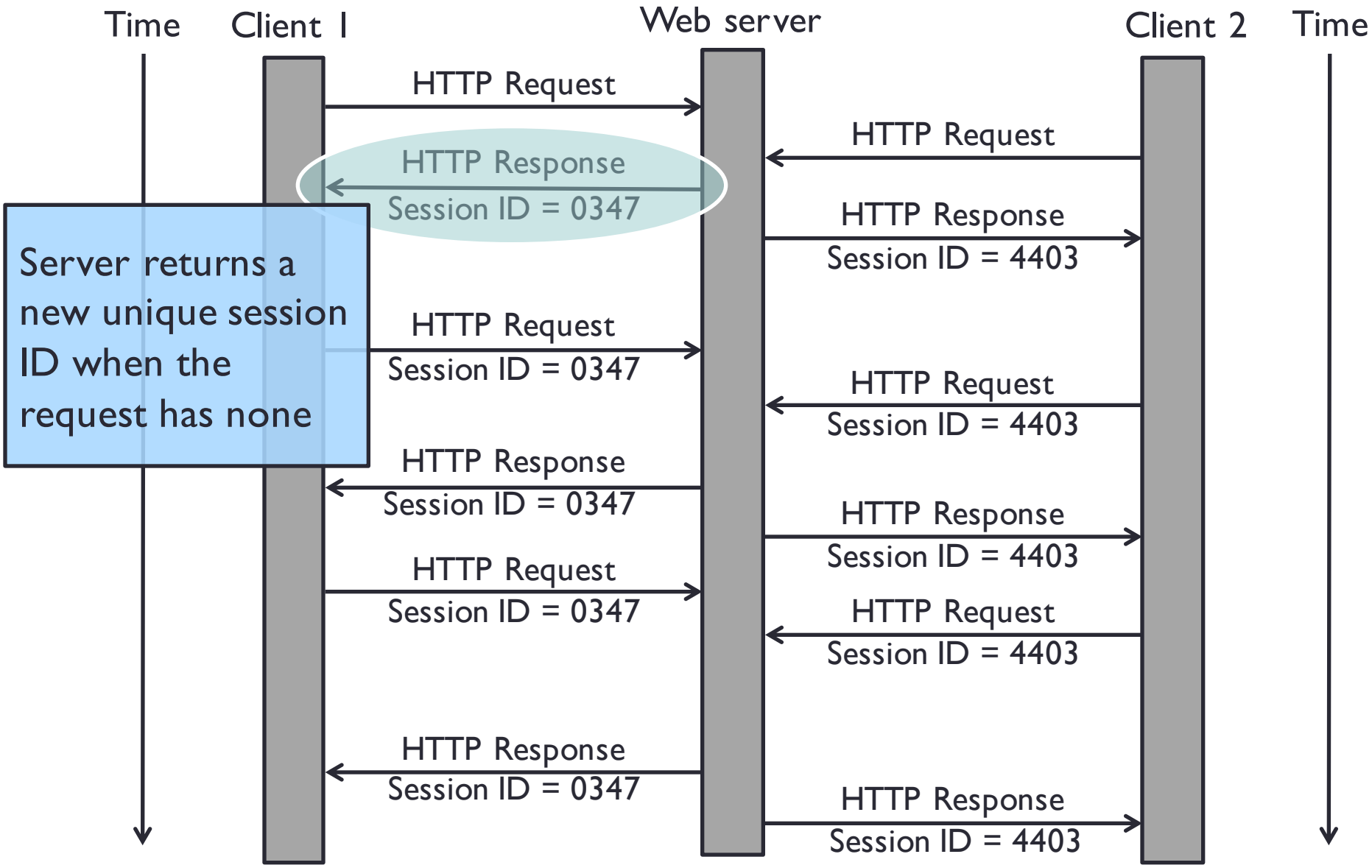
Sharing Data: Session Object

- One program component can store a value in the session object
- Another component can retrieve, use, and modify the value
- Depends on the PHP container
  - Software components are threads, not processes
  - PHP container stays resident and can keep shared memory

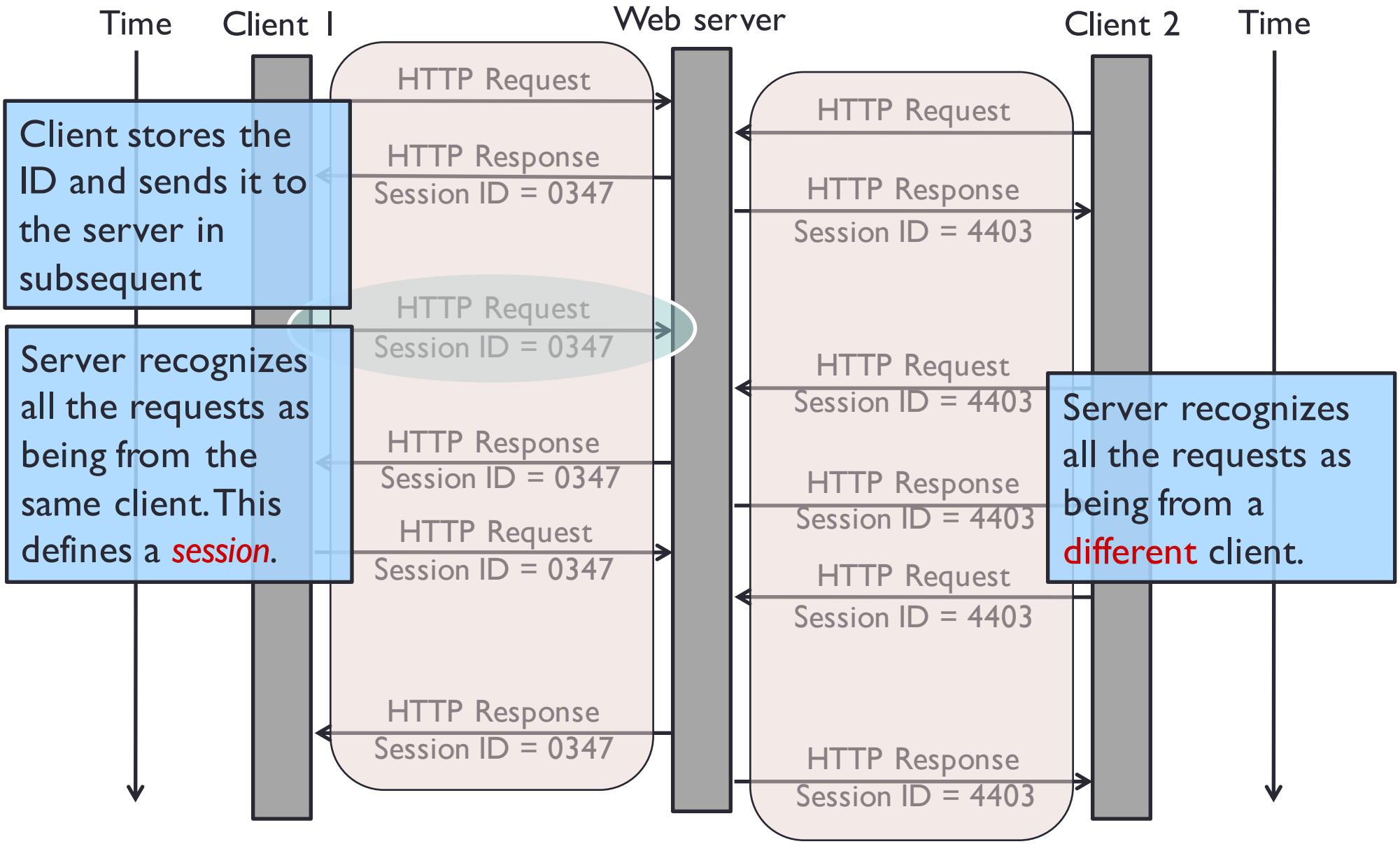
Session Scope



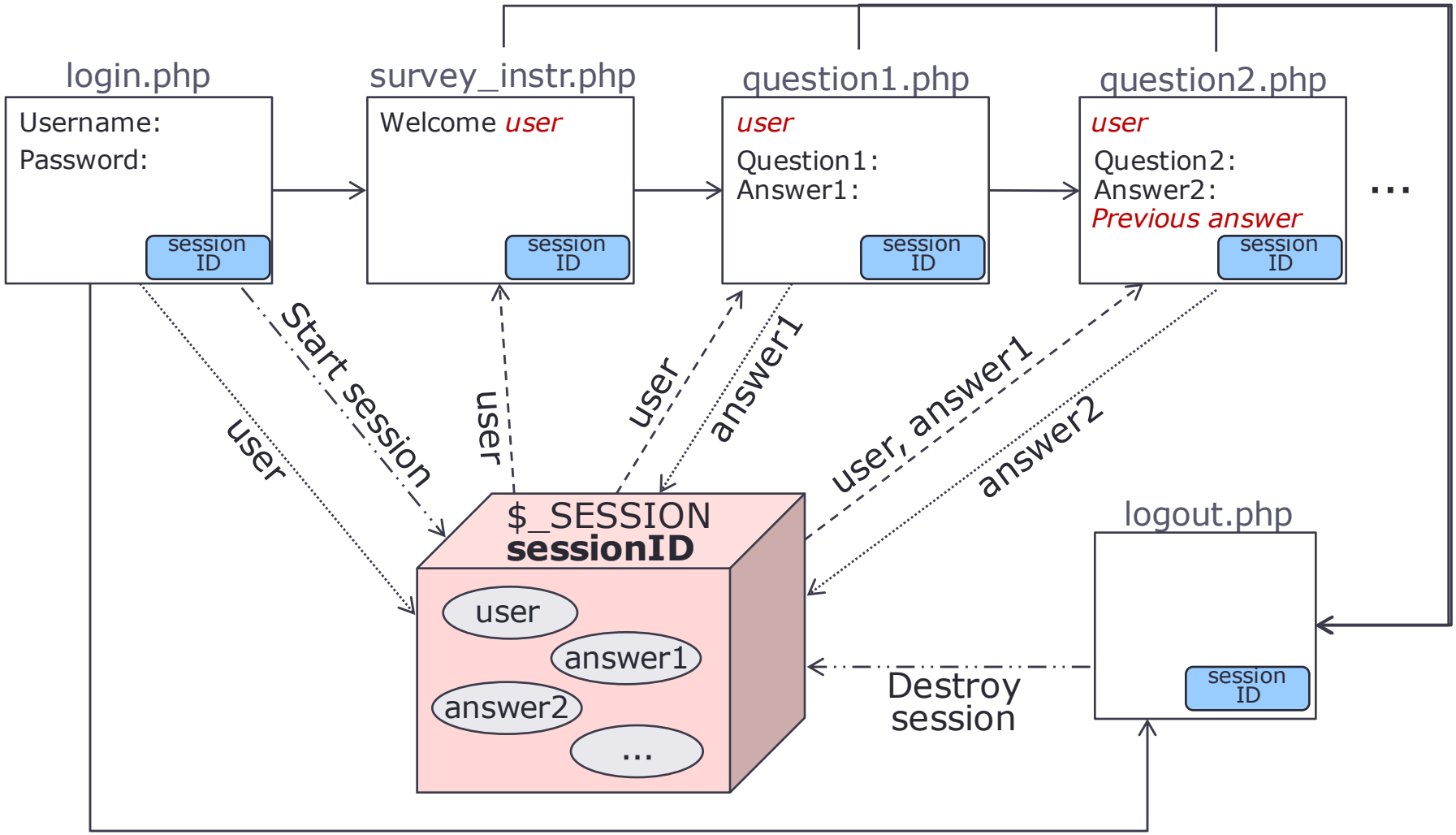
Sessions (Overview)



Sessions (Overview)



Sharing Data with Session Object



[You will implement this later]

Page-centric design

→ Send a request

⋯→ Store or update info in a session object

–·–·→ Instantiate or destroy a session

- - - - -→ Retrieve info from a session object

Using Session objects

- Initialize a session object

  ```
  session_start()
  ```

  Once the session is available, session data are accessible from an implicit $_SESSION global array variable

- Store information (state) in the session object, use name as a key

  ```
  $_SESSION[name] = value
  ```

- Get values from session objects, use name as a key

  ```
  $_SESSION[name]
  ```

Using Session Objects

- Get the number of param/value pairs stored in a session object

```
count($_SESSION)
```

- To view state stored in a session object, iterate a `$_SESSION` array or specify a parameter name `$_SESSION[`*name*`]`

```
foreach ($_SESSION as $key => $value)
{
    echo "$key maps to $value <br/>";
}
```

- Retrieve a session ID

```
session_id()
```

Using Session objects

- Remove a session variable, use name as a key:

```
unset($_SESSION[name])
```

- Terminate a session

```
session_destroy()
```

Completely remove a session object

Thought Questions

- What user-specific data are stored on the server? What user-specific data are stored on the client?
- How does the server recognize whether the requests are from the same user or different user?
- If a user wants to view his/her information previously stored in the server-side session, how does the server respond to the request?
- Assuming a user wants to update (or delete) his/her information previously stored in the server-side session, how does the server respond to the request?
- What can happen if someone (let's say, a hacker) knows the user's *sessionID*?
- What can be done to potentially prevent or minimize the chance of the user session being hacked?

More General Concept on Maintaining State

Single-user session state

- Cookies and session object

Multi-user session state

- Context object (will not be tested)
- Data persistence

Sometimes we want to share session data among multiple clients (group of users)

Why do we need them?

- Social network system – allow multiple users to interact
- Group working – online meeting
- Online bidding
- Reservation system, invitation system

Summary

- Managing state is fundamental to any program
- Managing state is the most unique aspect of designing and programming web applications
- Software vendors are creating new frameworks all the time
  - Most of them introduce additional state handling techniques
- Many professional developers make fundamental mistakes with managing state
  - Books and tutorials describe syntax, but not concepts

State management is the most common source of software faults in web apps

Extra slide

More general concepts on maintaining states

(will not be tested)

Context Scope