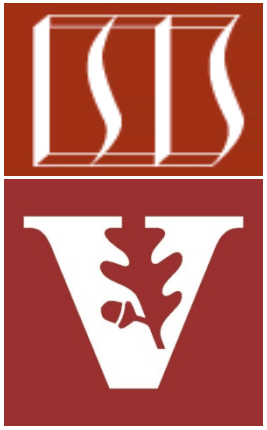
Overview of JAWS Web Server Case Study: Part 1

Douglas C. Schmidt
d.schmidt@vanderbilt.edu
www.dre.vanderbilt.edu/~schmidt



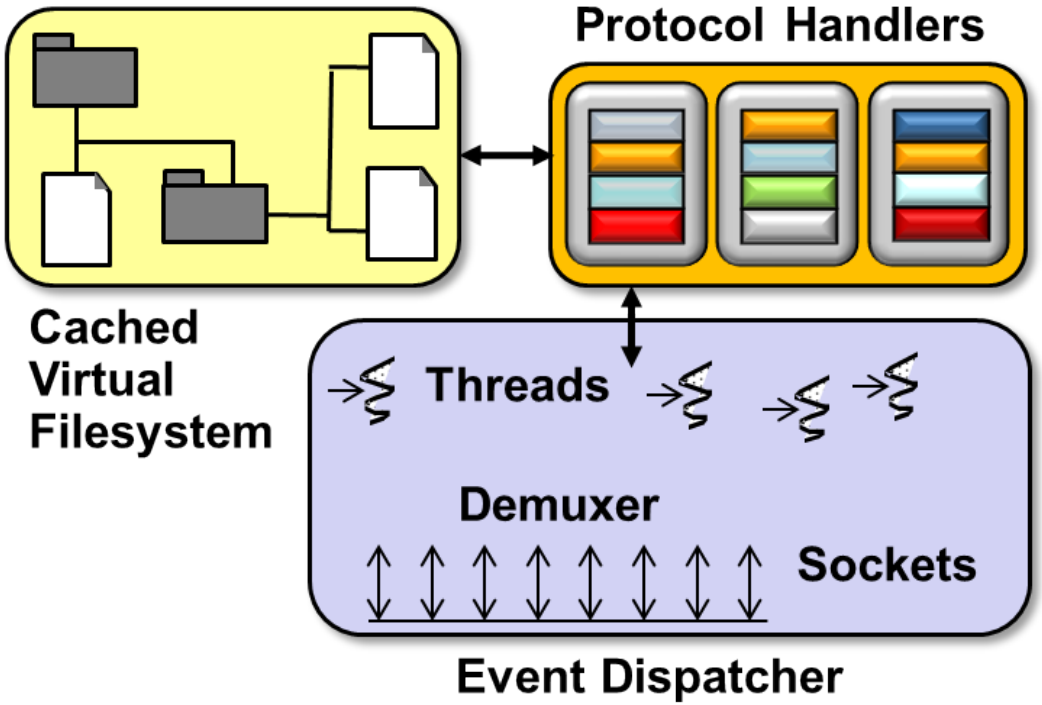
Professor of Computer Science

Institute for Software Integrated Systems

Vanderbilt University
Nashville, Tennessee, USA

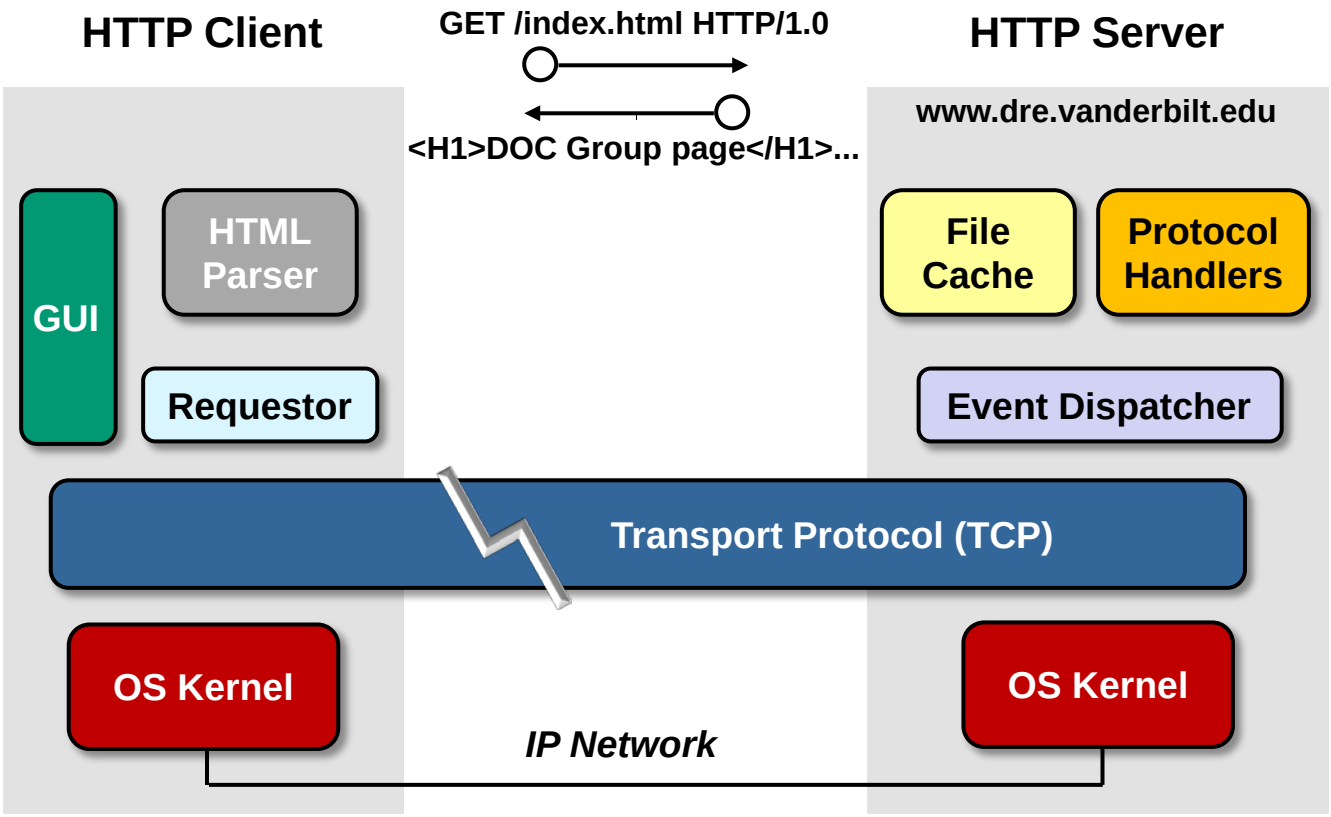
Topics Covered in this Part of the Module

- Describe the pattern-oriented JAWS web server case study



Web Server Overview

Goal: Download web content scalably, efficiently, & robustly

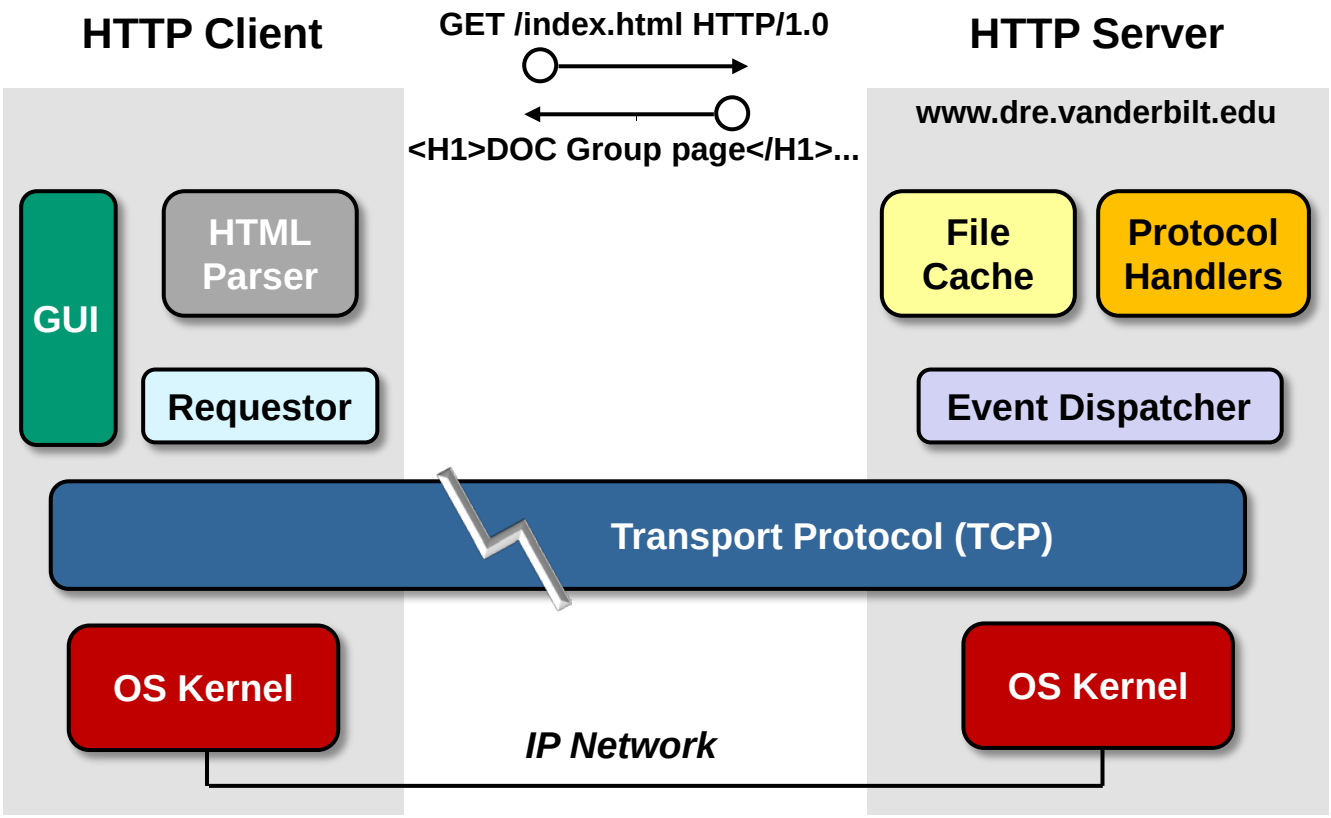


Web Server Overview

Goal: Download web content scalably, efficiently, & robustly

Key Requirements

- Portable to multiple operating systems & protocols
  - e.g., Windows, UNIX, real-time operating systems, etc.

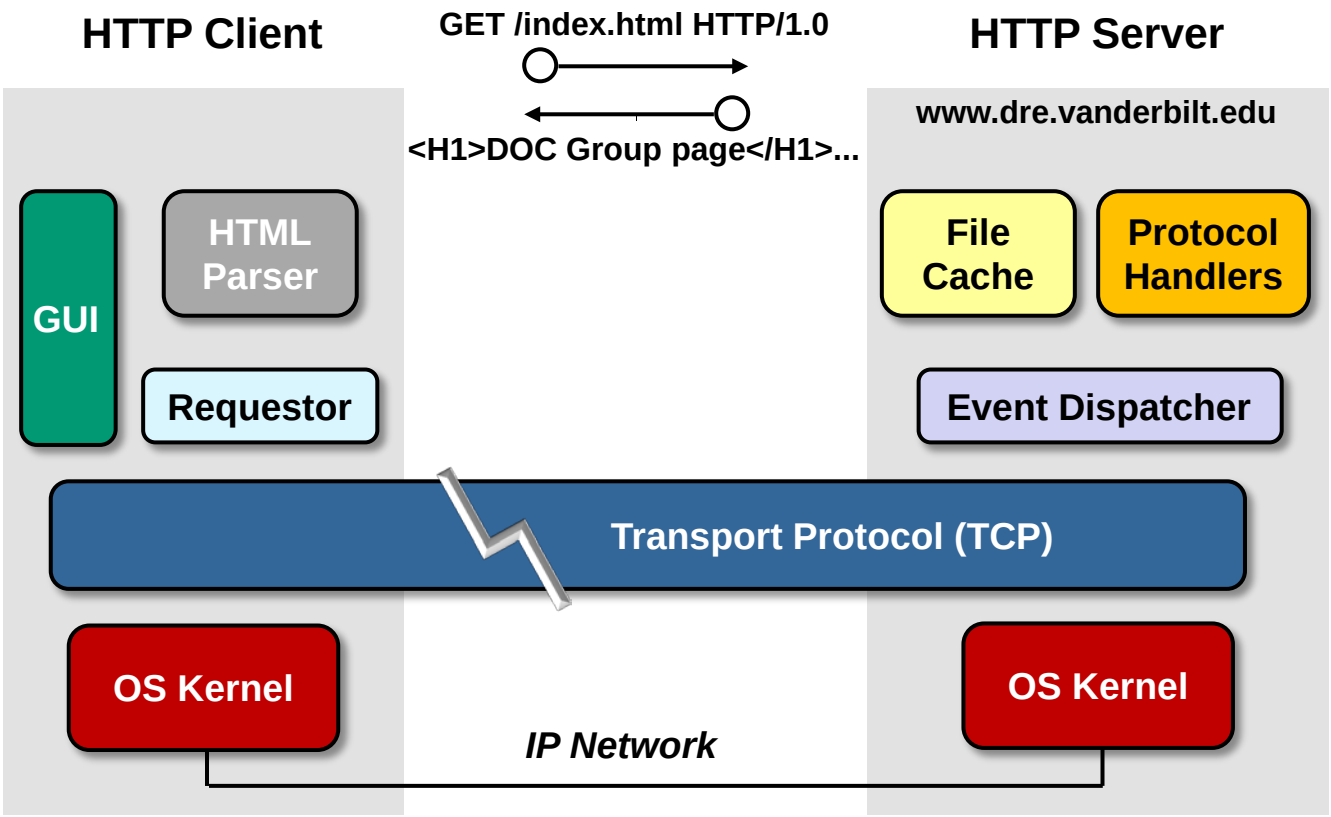


Web Server Overview

Goal: Download web content scalably, efficiently, & robustly

Key Requirements

- Portable to multiple operating systems & protocols
- Support many web server design alternatives
  - e.g., concurrency models, event demuxing models, file caching models, etc.

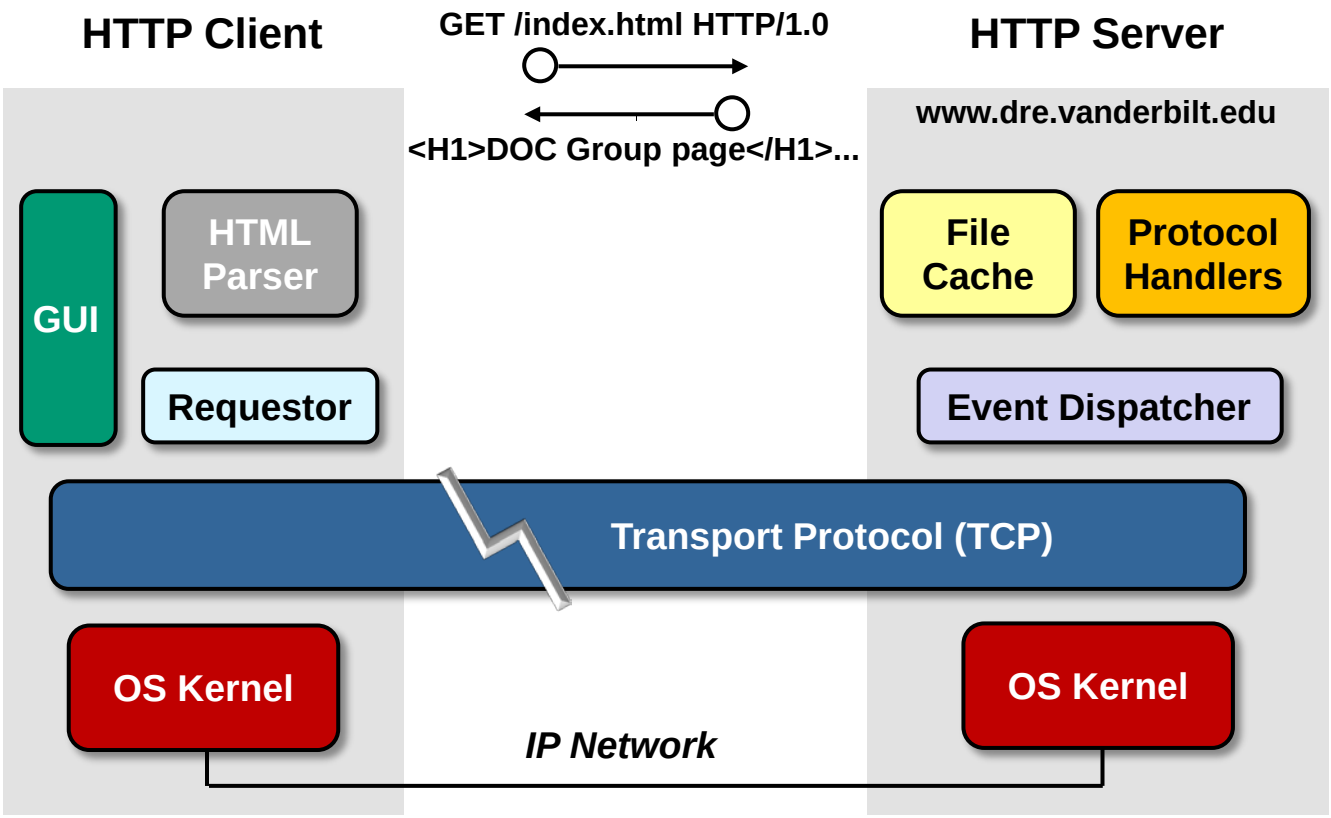


Web Server Overview

Goal: Download web content scalably, efficiently, & robustly

Key Requirements

- Portable to multiple operating systems & protocols
- Support many web server design alternatives
- Leverage advances in multi-processor hardware/software
  - e.g., multi-core, async I/O, etc.

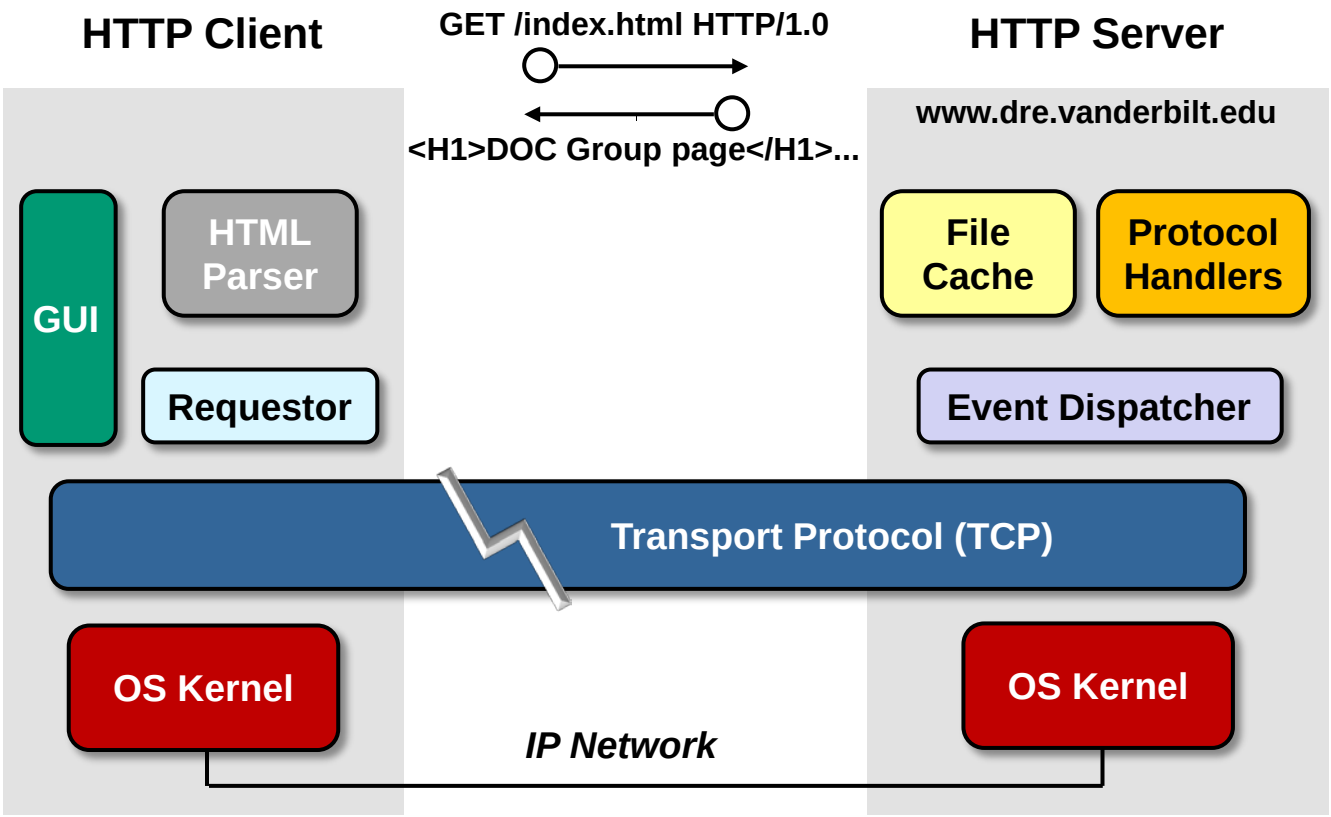


Web Server Overview

Goal: Download web content scalably, efficiently, & robustly

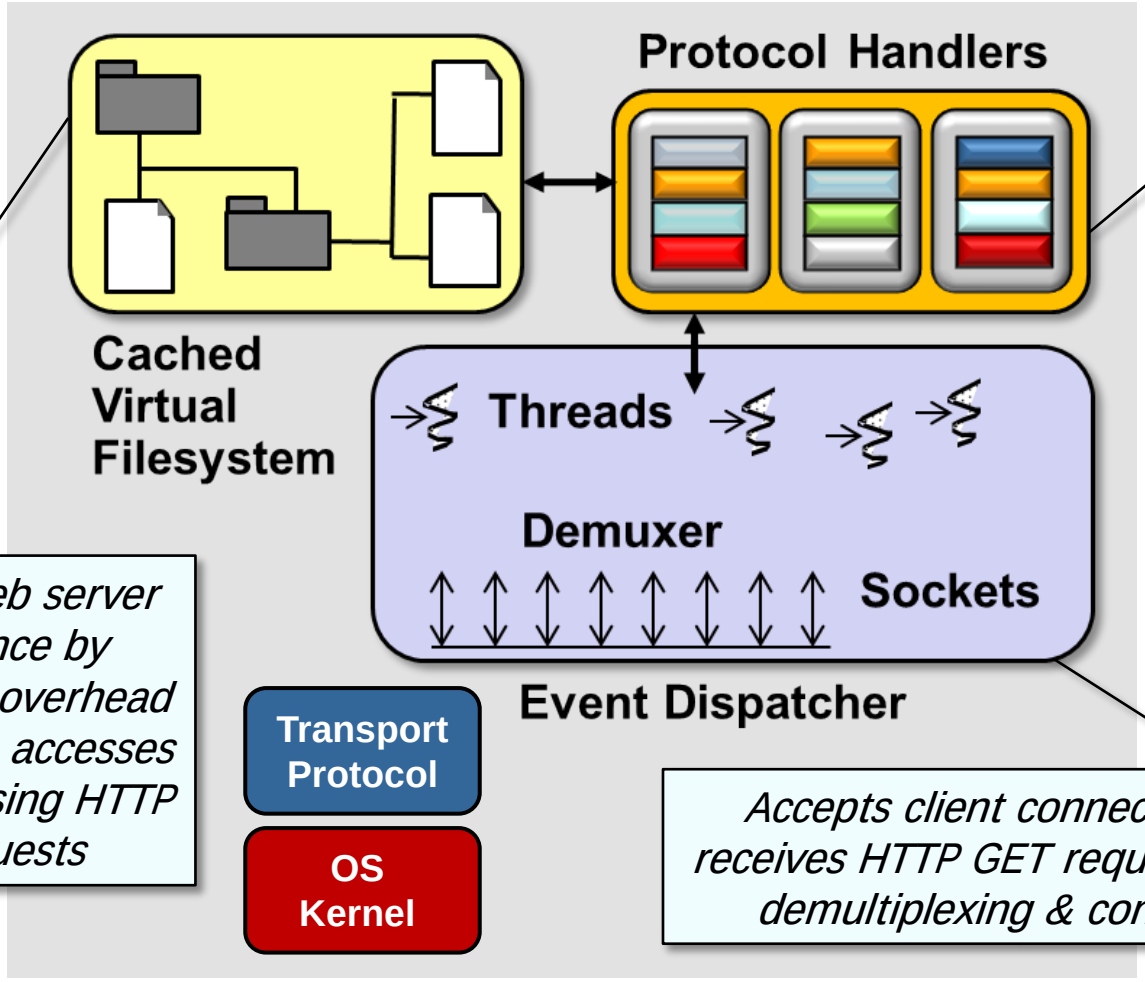
Key Requirements

- Portable to multiple operating systems & protocols
- Support many web server design alternatives
- Leverage advances in multi-processor hardware/software
- Operate effectively in various settings
  - e.g., even in the face of erroneous or malicious clients



JAWS Web Server Architecture

- JAWS is a pattern-oriented web server implemented using ACE frameworks



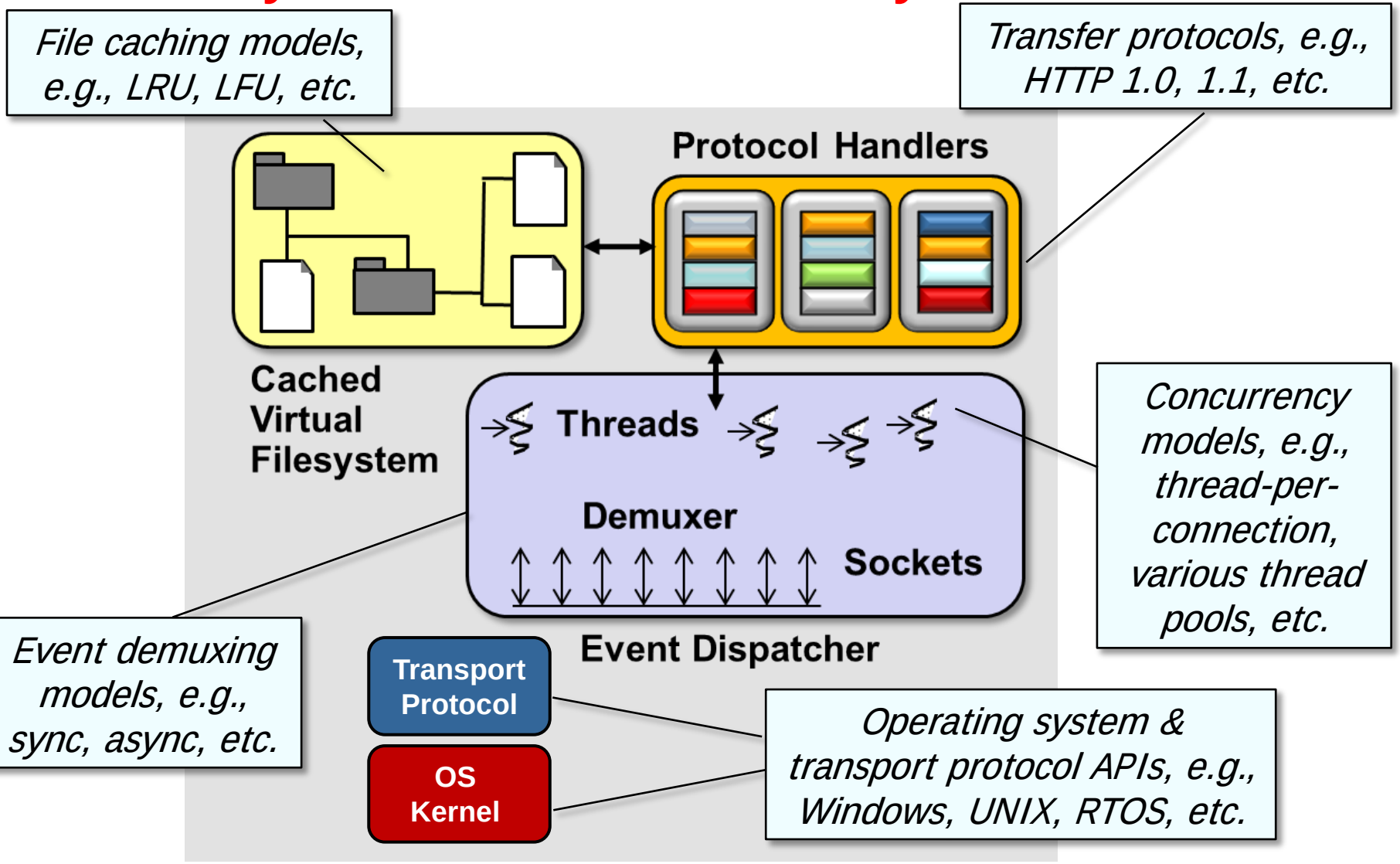
Performs parsing & protocol processing of HTTP requests

Improves web server performance by reducing the overhead of file system accesses when processing HTTP GET requests

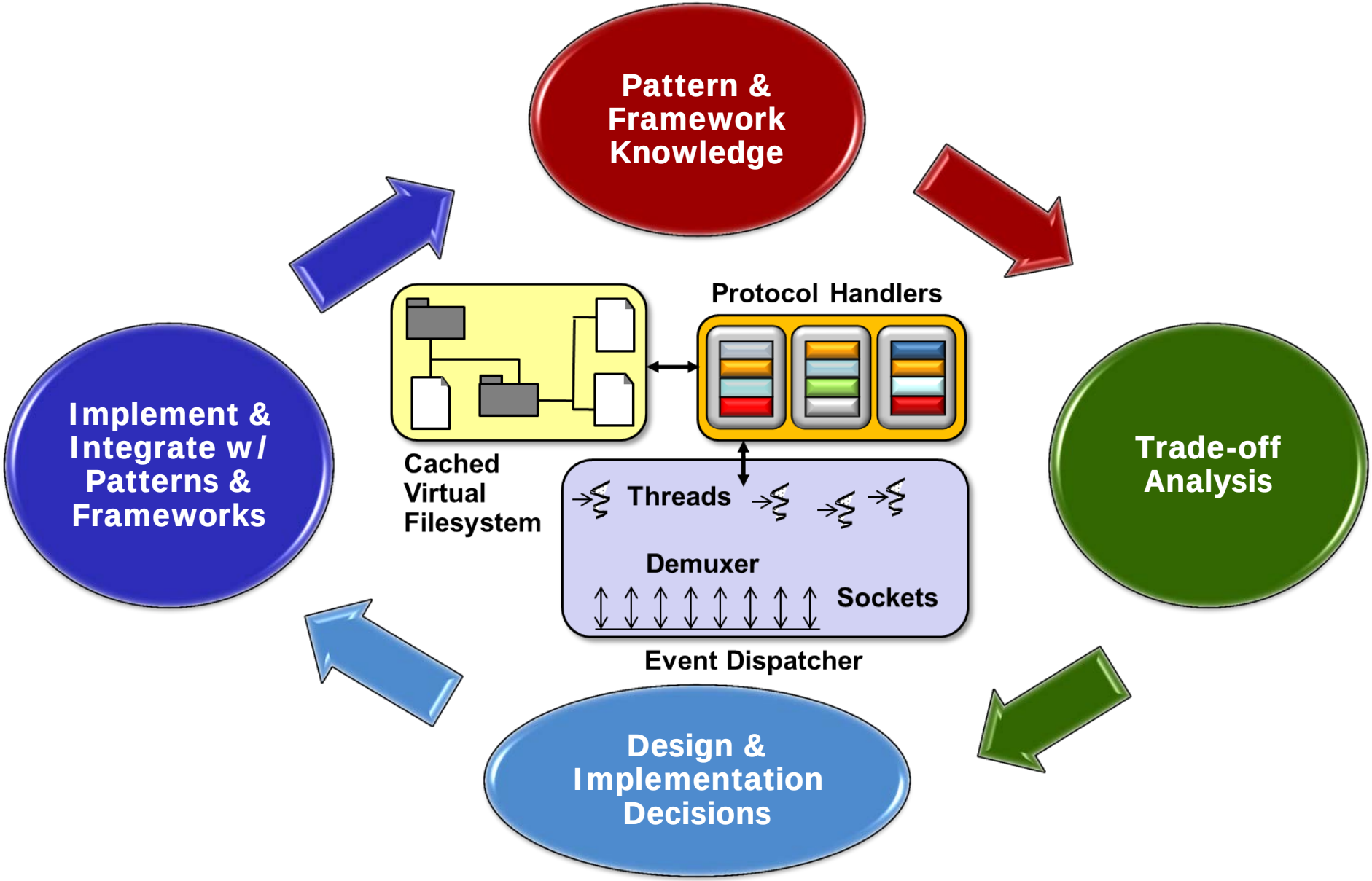
Accepts client connection request events, receives HTTP GET requests, & performs event demultiplexing & concurrency strategies

JAWS is available in open-source ACE release at download.dre.vanderbilt.edu

Key Sources of Variability in JAWS

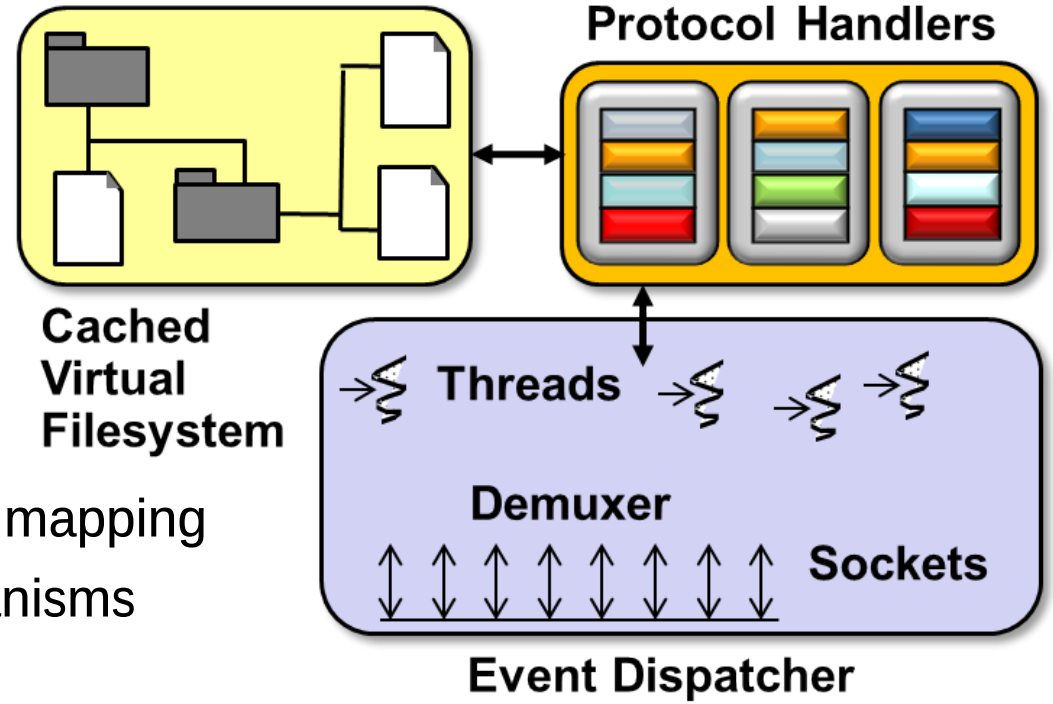


A Variation-oriented Design Process



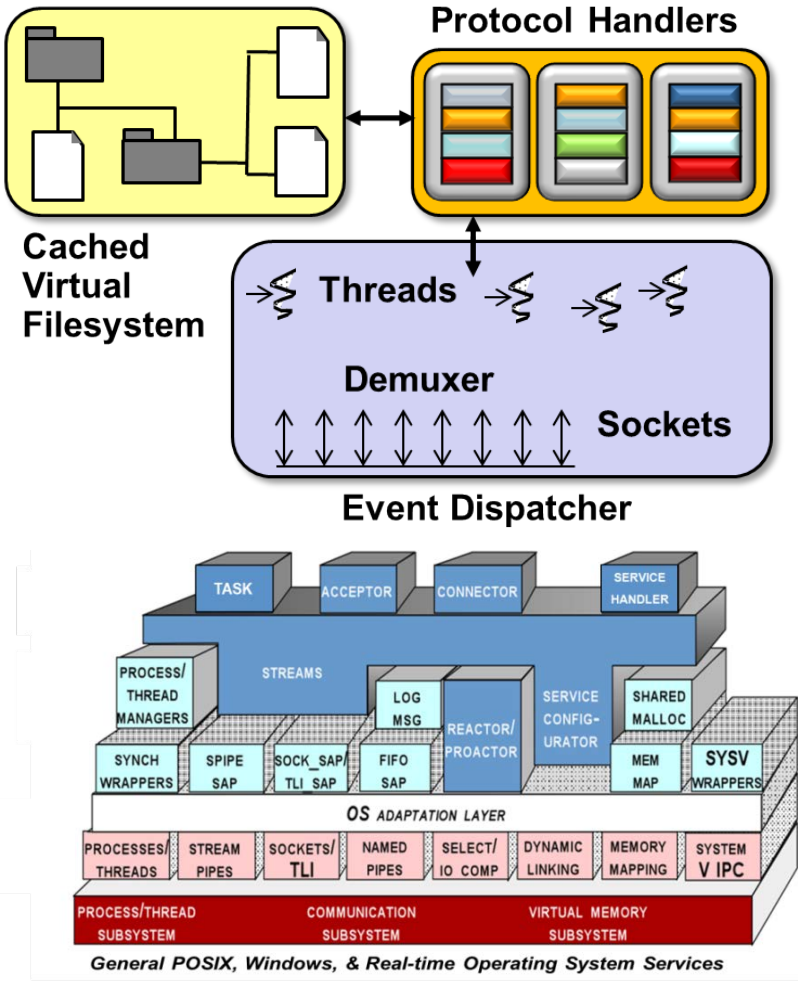
Summary

- JAWS is a pattern-oriented open-source web server that's been optimized for high performance, e.g.:
  - Uses lightweight concurrency models
  - Minimizes locking
  - Applies file caching & memory mapping
  - Uses "gather-write" I/O mechanisms
  - Minimizes logging
  - Pre-computes HTTP responses
  - Avoids excessive `time()` calls
  - Optimizes the transport interface



Summary

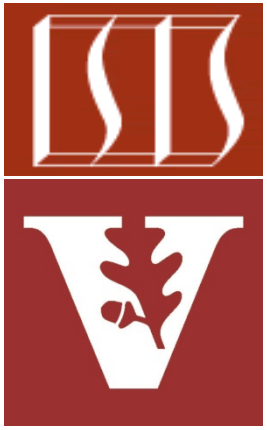
- JAWS is a pattern-oriented open-source web server that's been optimized for high performance, e.g.:
  - Uses lightweight concurrency models
  - Minimizes locking
  - Applies file caching & memory mapping
  - Uses "gather-write" I/O mechanisms
  - Minimizes logging
  - Pre-computes HTTP responses
  - Avoids excessive `time()` calls
  - Optimizes the transport interface



- There are multiple versions of JAWS in ACE
  - **`ACE_ROOT/apps/{JAWS,JAW2,JAW3}`**

Overview of JAWS Web Server Case Study: Part 2

Douglas C. Schmidt
d.schmidt@vanderbilt.edu
www.dre.vanderbilt.edu/~schmidt



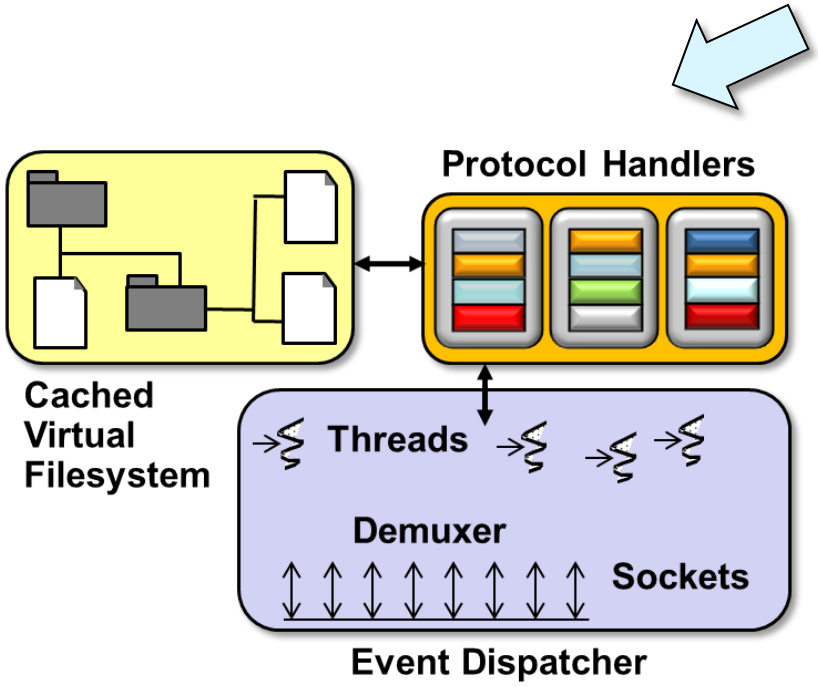
Professor of Computer Science

Institute for Software Integrated Systems

Vanderbilt University
Nashville, Tennessee, USA

Topics Covered in this Part of the Module

- Describe the pattern-oriented JAWS web server case study
- Summarize the patterns in the JAWS web server design

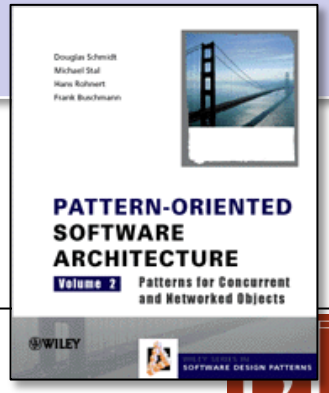


Design Problem	Pattern(s)
Encapsulating low-level OS APIs	Wrapper Facade
Decoupling event demuxing & connection management from protocol processing	Reactor & Acceptor-Connector
Scaling up performance via multi-threading	Half-Sync/Half-Async & Active Object
Synchronized request queue	Monitor Object
Minimizing multi-threading overhead	Leader/Followers
Using asynchronous I/O effectively	Proactor
Efficiently demuxing asynchronous operations & completions	Asynchronous Completion Token
Enhancing server (re)configurability	Component Configurator
Minimizing unused server resources	Activator
Transparently parameterizing synchronization into components	Strategized Locking
Ensuring locks are released properly	Scoped Locking
Minimizing unnecessary locking	Thread-Safe Interface

Outline of the Design Space for POSA2 Patterns

Effectively design & configure app access to interfaces & implementations of evolving services & components

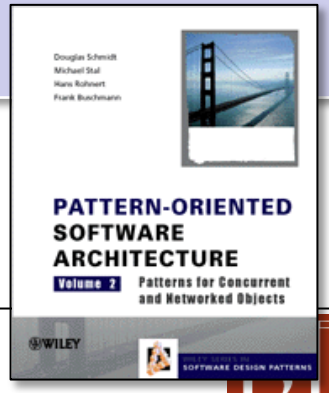
Service Access & Configuration Patterns	Event Handling Patterns	Concurrency Patterns	Synchronization Patterns
Wrapper Facade	Reactor	Active Object	Strategized Locking
Component Configurator	Proactor	Half-Sync/Half-Async	Scoped Locking
Interceptor	Acceptor-Connector	Leader/Followers	Thread-Safe Interface
Extension Interface	Asynchronous Completion Token	Monitor Object	Double-Checked Locking Optimization
		Thread-Specific Storage	



Outline of the Design Space for POSA2 Patterns

Simplify development of flexible & efficient event-driven apps

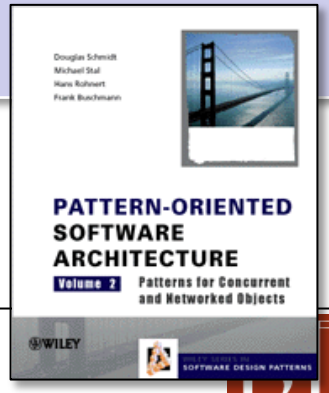
Service Access & Configuration Patterns	Event Handling Patterns	Concurrency Patterns	Synchronization Patterns
Wrapper Facade	Reactor	Active Object	Strategized Locking
Component Configurator	Proactor	Half-Sync/Half-Async	Scoped Locking
Interceptor	Acceptor-Connector	Leader/Followers	Thread-Safe Interface
Extension Interface	Asynchronous Completion Token	Monitor Object	Double-Checked Locking Optimization
		Thread-Specific Storage	



Outline of the Design Space for POSA2 Patterns

Enhance design & performance of multi-threaded concurrent & networked software

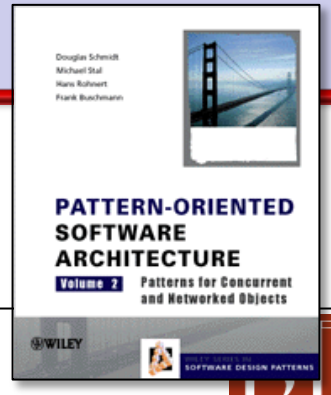
Service Access & Configuration Patterns	Event Handling Patterns	Concurrency Patterns	Synchronization Patterns
Wrapper Facade	Reactor	Active Object	Strategized Locking
Component Configurator	Proactor	Half-Sync/Half-Async	Scoped Locking
Interceptor	Acceptor-Connector	Leader/Followers	Thread-Safe Interface
Extension Interface	Asynchronous Completion Token	Monitor Object	Double-Checked Locking Optimization
		Thread-Specific Storage	



Outline of the Design Space for POSA2 Patterns

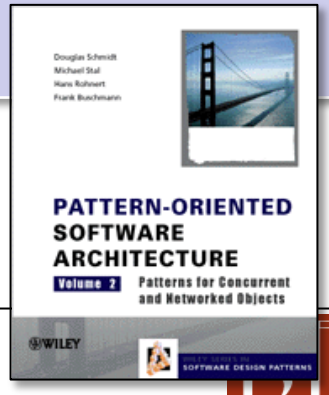
Provide flexible solutions to common problems related to synchronizing concurrent objects

Service Access & Configuration Patterns	Event Handling Patterns	Concurrency Patterns	Synchronization Patterns
Wrapper Facade	Reactor	Active Object	Strategized Locking
Component Configurator	Proactor	Half-Sync/Half-Async	Scoped Locking
Interceptor	Acceptor-Connector	Leader/Followers	Thread-Safe Interface
Extension Interface	Asynchronous Completion Token	Monitor Object	Double-Checked Locking Optimization
		Thread-Specific Storage	



Outline of the Design Space for POSA2 Patterns

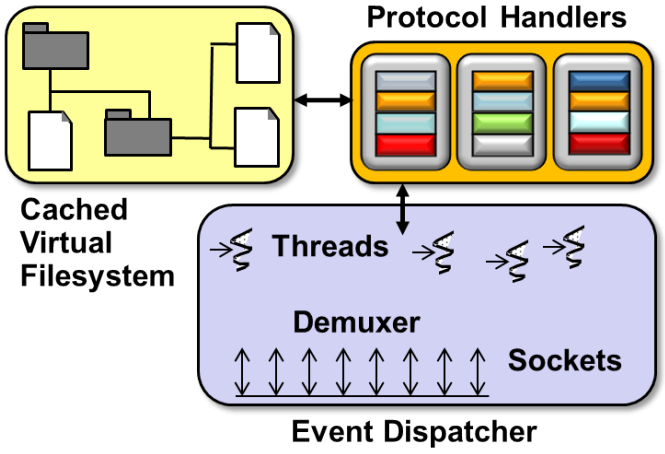
Service Access & Configuration Patterns	Event Handling Patterns	Concurrency Patterns	Synchronization Patterns
Wrapper Facade √	Reactor √	Active Object √	Strategized Locking √
Component Configurator √	Proactor √	Half-Sync/Half-Async √	Scoped Locking √
Interceptor	Acceptor-Connector √	Leader/Followers √	Thread-Safe Interface √
Extension Interface	Asynchronous Completion Token √	Monitor Object √	Double-Checked Locking Optimization √
		Thread-Specific Storage	

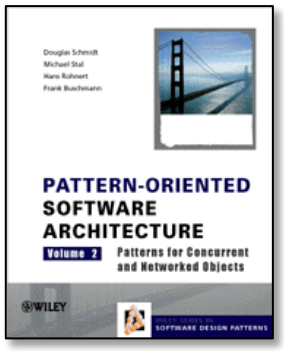


www.dre.vanderbilt.edu/POSA has info on POSA2 book

Design Problems & Pattern-Oriented Solutions

- Many design problems faced developing JAWS can be resolved via POSA2 (& GoF) patterns

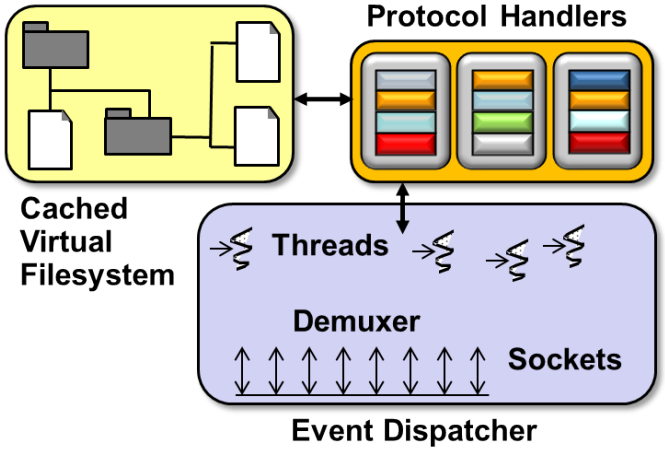


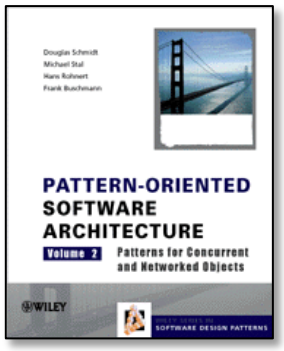


Design Problem	Pattern(s)
Encapsulating low-level OS APIs	Wrapper Facade
Decoupling event demuxing & connection management from protocol processing	Reactor & Acceptor-Connector
Scaling up performance via multi-threading	Half-Sync/Half-Async & Active Object
Synchronized request queue	Monitor Object
Minimizing multi-threading overhead	Leader/Followers
Using asynchronous I/O effectively	Proactor
Efficiently demuxing asynchronous operations & completions	Asynchronous Completion Token
Enhancing server (re)configurability	Component Configurator
Minimizing unused server resources	Activator
Transparently parameterizing synchronization into components	Strategized Locking
Ensuring locks are released properly	Scoped Locking
Minimizing unnecessary locking	Thread-Safe Interface

Design Problems & Pattern-Oriented Solutions

- Many design problems faced developing JAWS can be resolved via POSA2 (& GoF) patterns

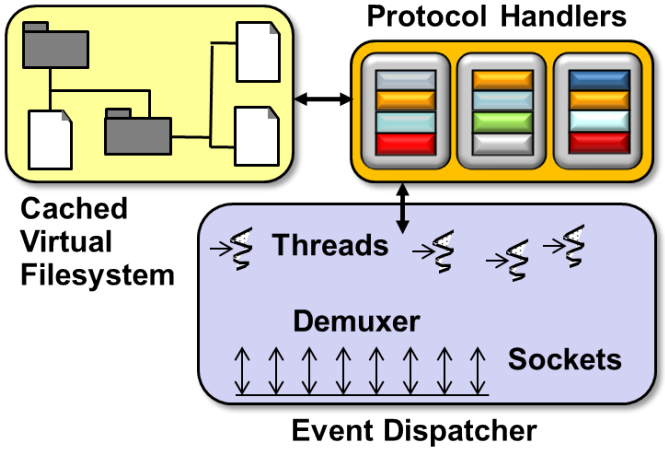


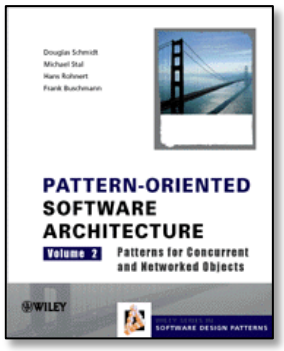


Design Problem	Pattern(s)
Encapsulating low-level OS APIs	Wrapper Facade
Decoupling event demuxing & connection management from protocol processing	Reactor & Acceptor-Connector
Scaling up performance via multi-threading	Half-Sync/Half-Async & Active Object
Synchronized request queue	Monitor Object
Minimizing multi-threading overhead	Leader/Followers
Using asynchronous I/O effectively	Proactor
Efficiently demuxing asynchronous operations & completions	Asynchronous Completion Token
Enhancing server (re)configurability	Component Configurator
Minimizing unused server resources	Activator
Transparently parameterizing synchronization into components	Strategized Locking
Ensuring locks are released properly	Scoped Locking
Minimizing unnecessary locking	Thread-Safe Interface

Design Problems & Pattern-Oriented Solutions

- Many design problems faced developing JAWS can be resolved via POSA2 (& GoF) patterns

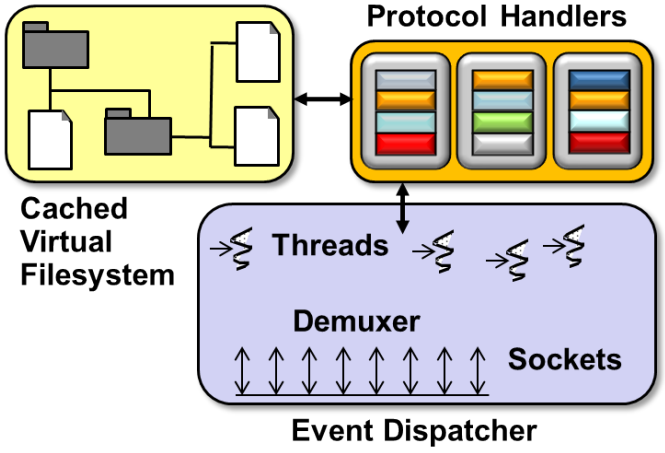


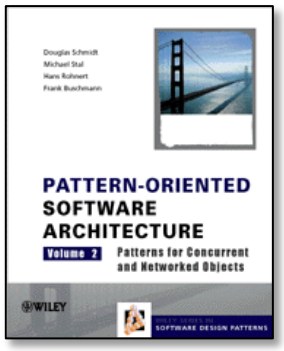


Design Problem	Pattern(s)
Encapsulating low-level OS APIs	Wrapper Facade
Decoupling event demuxing & connection management from protocol processing	Reactor & Acceptor-Connector
Scaling up performance via multi-threading	Half-Sync/Half-Async & Active Object
Synchronized request queue	Monitor Object
Minimizing multi-threading overhead	Leader/Followers
Using asynchronous I/O effectively	Proactor
Efficiently demuxing asynchronous operations & completions	Asynchronous Completion Token
Enhancing server (re)configurability	Component Configurator
Minimizing unused server resources	Activator
Transparently parameterizing synchronization into components	Strategized Locking
Ensuring locks are released properly	Scoped Locking
Minimizing unnecessary locking	Thread-Safe Interface

Design Problems & Pattern-Oriented Solutions

- Many design problems faced developing JAWS can be resolved via POSA2 (& GoF) patterns

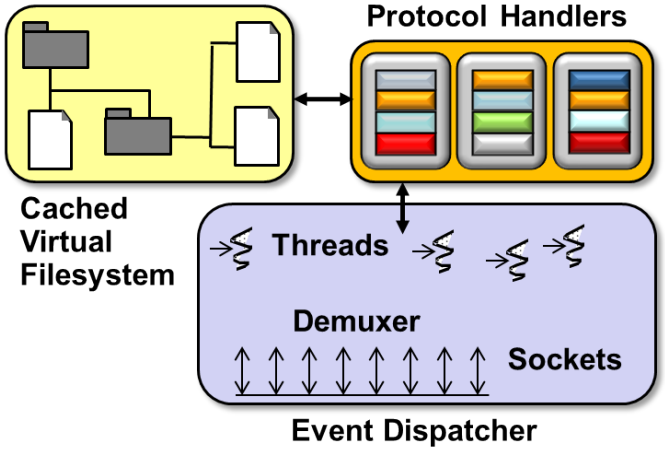


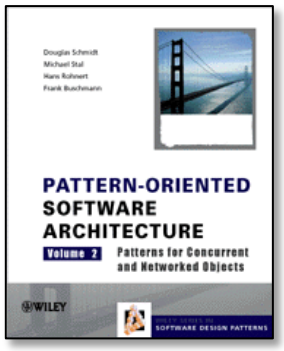


Design Problem	Pattern(s)
Encapsulating low-level OS APIs	Wrapper Facade
Decoupling event demuxing & connection management from protocol processing	Reactor & Acceptor-Connector
Scaling up performance via multi-threading	Half-Sync/Half-Async & Active Object
Synchronized request queue	Monitor Object
Minimizing multi-threading overhead	Leader/Followers
Using asynchronous I/O effectively	Proactor
Efficiently demuxing asynchronous operations & completions	Asynchronous Completion Token
Enhancing server (re)configurability	Component Configurator
Minimizing unused server resources	Activator
Transparently parameterizing synchronization into components	Strategized Locking
Ensuring locks are released properly	Scoped Locking
Minimizing unnecessary locking	Thread-Safe Interface

Design Problems & Pattern-Oriented Solutions

- Many design problems faced developing JAWS can be resolved via POSA2 (& GoF) patterns

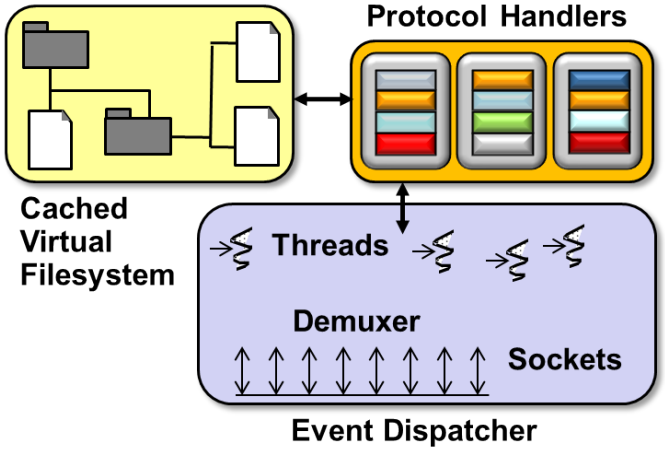


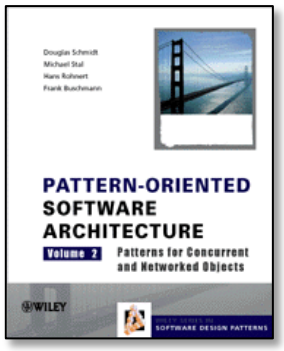


Design Problem	Pattern(s)
Encapsulating low-level OS APIs	Wrapper Facade
Decoupling event demuxing & connection management from protocol processing	Reactor & Acceptor-Connector
Scaling up performance via multi-threading	Half-Sync/Half-Async & Active Object
Synchronized request queue	Monitor Object
Minimizing multi-threading overhead	Leader/Followers
Using asynchronous I/O effectively	Proactor
Efficiently demuxing asynchronous operations & completions	Asynchronous Completion Token
Enhancing server (re)configurability	Component Configurator
Minimizing unused server resources	Activator
Transparently parameterizing synchronization into components	Strategized Locking
Ensuring locks are released properly	Scoped Locking
Minimizing unnecessary locking	Thread-Safe Interface

Design Problems & Pattern-Oriented Solutions

- Many design problems faced developing JAWS can be resolved via POSA2 (& GoF) patterns

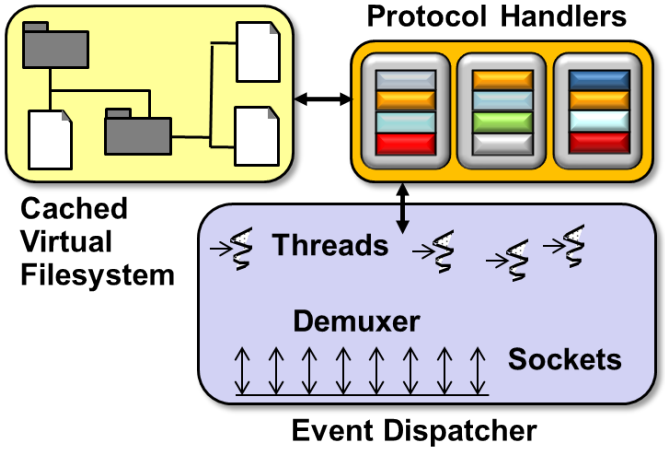


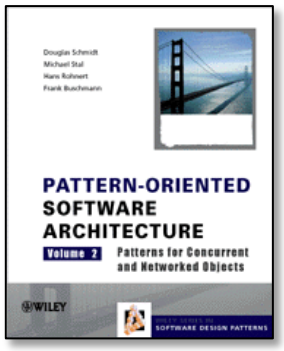


Design Problem	Pattern(s)
Encapsulating low-level OS APIs	Wrapper Facade
Decoupling event demuxing & connection management from protocol processing	Reactor & Acceptor-Connector
Scaling up performance via multi-threading	Half-Sync/Half-Async & Active Object
Synchronized request queue	Monitor Object
Minimizing multi-threading overhead	Leader/Followers
Using asynchronous I/O effectively	Proactor
Efficiently demuxing asynchronous operations & completions	Asynchronous Completion Token
Enhancing server (re)configurability	Component Configurator
Minimizing unused server resources	Activator
Transparently parameterizing synchronization into components	Strategized Locking
Ensuring locks are released properly	Scoped Locking
Minimizing unnecessary locking	Thread-Safe Interface

Design Problems & Pattern-Oriented Solutions

- Many design problems faced developing JAWS can be resolved via POSA2 (& GoF) patterns

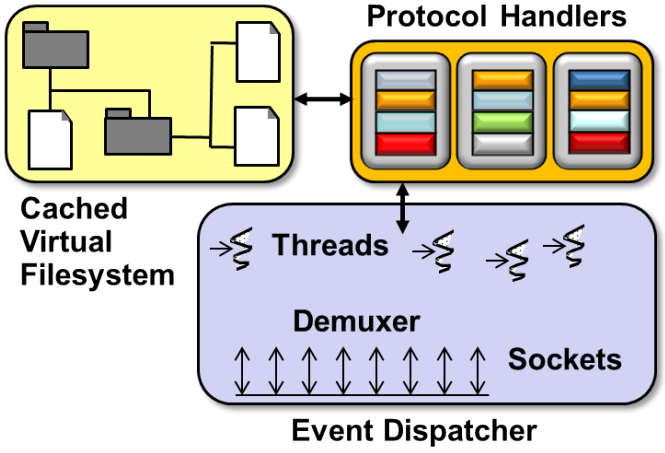


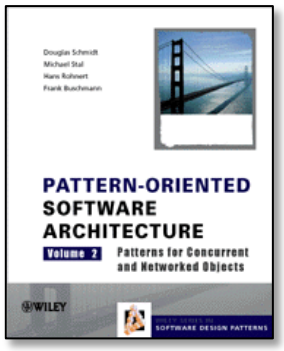


Design Problem	Pattern(s)
Encapsulating low-level OS APIs	Wrapper Facade
Decoupling event demuxing & connection management from protocol processing	Reactor & Acceptor-Connector
Scaling up performance via multi-threading	Half-Sync/Half-Async & Active Object
Synchronized request queue	Monitor Object
Minimizing multi-threading overhead	Leader/Followers
Using asynchronous I/O effectively	Proactor
Efficiently demuxing asynchronous operations & completions	Asynchronous Completion Token
Enhancing server (re)configurability	Component Configurator
Minimizing unused server resources	Activator
Transparently parameterizing synchronization into components	Strategized Locking
Ensuring locks are released properly	Scoped Locking
Minimizing unnecessary locking	Thread-Safe Interface

Design Problems & Pattern-Oriented Solutions

- Many design problems faced developing JAWS can be resolved via POSA2 (& GoF) patterns

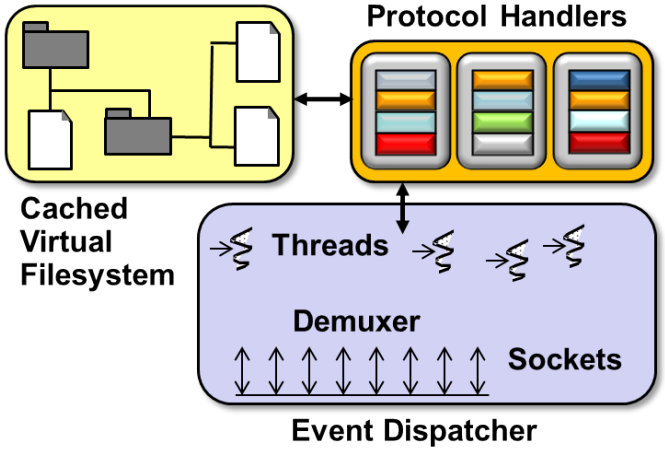


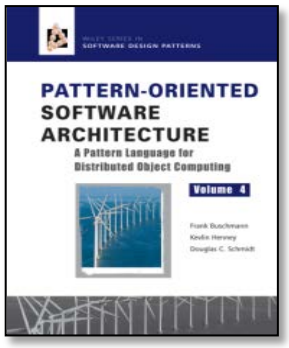


Design Problem	Pattern(s)
Encapsulating low-level OS APIs	Wrapper Facade
Decoupling event demuxing & connection management from protocol processing	Reactor & Acceptor-Connector
Scaling up performance via multi-threading	Half-Sync/Half-Async & Active Object
Synchronized request queue	Monitor Object
Minimizing multi-threading overhead	Leader/Followers
Using asynchronous I/O effectively	Proactor
Efficiently demuxing asynchronous operations & completions	Asynchronous Completion Token
Enhancing server (re)configurability	Component Configurator
Minimizing unused server resources	Activator
Transparently parameterizing synchronization into components	Strategized Locking
Ensuring locks are released properly	Scoped Locking
Minimizing unnecessary locking	Thread-Safe Interface

Design Problems & Pattern-Oriented Solutions

- Many design problems faced developing JAWS can be resolved via POSA2 (& GoF) patterns

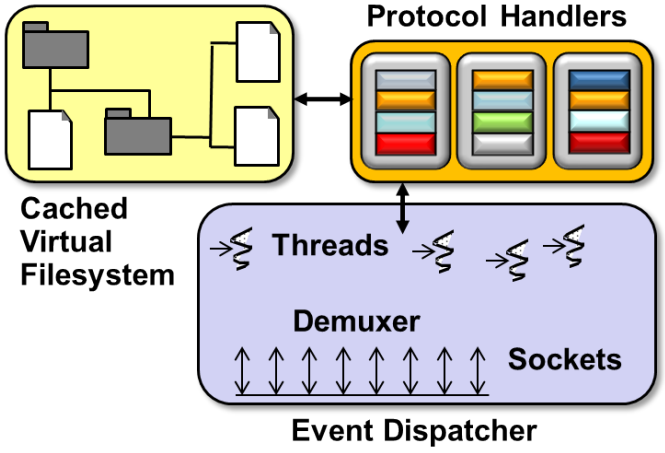


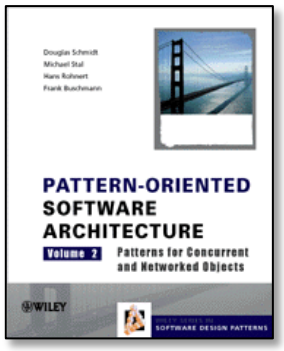


Design Problem	Pattern(s)
Encapsulating low-level OS APIs	Wrapper Facade
Decoupling event demuxing & connection management from protocol processing	Reactor & Acceptor-Connector
Scaling up performance via multi-threading	Half-Sync/Half-Async & Active Object
Synchronized request queue	Monitor Object
Minimizing multi-threading overhead	Leader/Followers
Using asynchronous I/O effectively	Proactor
Efficiently demuxing asynchronous operations & completions	Asynchronous Completion Token
Enhancing server (re)configurability	Component Configurator
Minimizing unused server resources	Activator
Transparently parameterizing synchronization into components	Strategized Locking
Ensuring locks are released properly	Scoped Locking
Minimizing unnecessary locking	Thread-Safe Interface

Design Problems & Pattern-Oriented Solutions

- Many design problems faced developing JAWS can be resolved via POSA2 (& GoF) patterns

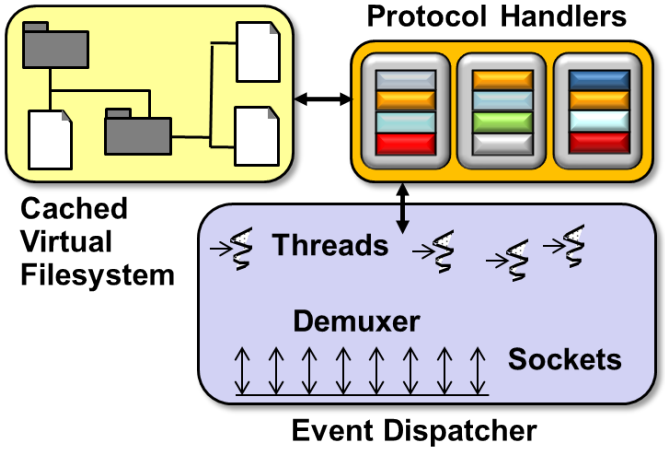


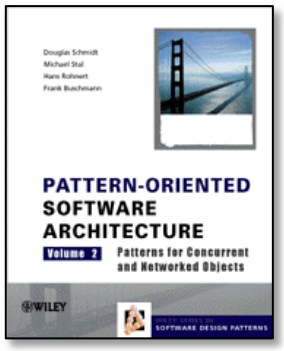


Design Problem	Pattern(s)
Encapsulating low-level OS APIs	Wrapper Facade
Decoupling event demuxing & connection management from protocol processing	Reactor & Acceptor-Connector
Scaling up performance via multi-threading	Half-Sync/Half-Async & Active Object
Synchronized request queue	Monitor Object
Minimizing multi-threading overhead	Leader/Followers
Using asynchronous I/O effectively	Proactor
Efficiently demuxing asynchronous operations & completions	Asynchronous Completion Token
Enhancing server (re)configurability	Component Configurator
Minimizing unused server resources	Activator
Transparently parameterizing synchronization into components	Strategized Locking
Ensuring locks are released properly	Scoped Locking
Minimizing unnecessary locking	Thread-Safe Interface

Design Problems & Pattern-Oriented Solutions

- Many design problems faced developing JAWS can be resolved via POSA2 (& GoF) patterns

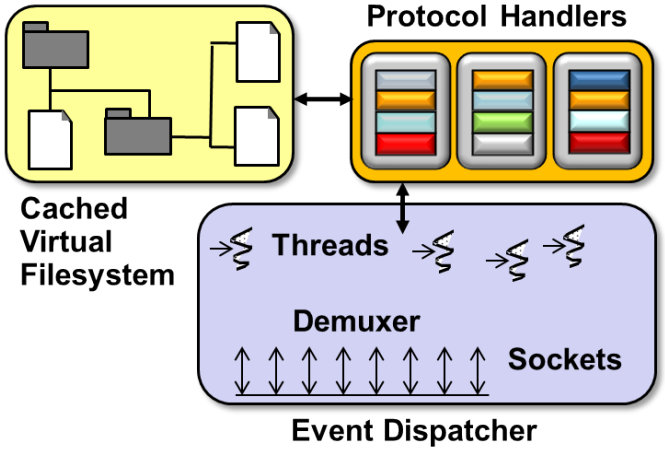


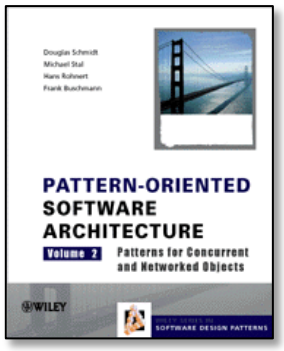


Design Problem	Pattern(s)
Encapsulating low-level OS APIs	Wrapper Facade
Decoupling event demuxing & connection management from protocol processing	Reactor & Acceptor-Connector
Scaling up performance via multi-threading	Half-Sync/Half-Async & Active Object
Synchronized request queue	Monitor Object
Minimizing multi-threading overhead	Leader/Followers
Using asynchronous I/O effectively	Proactor
Efficiently demuxing asynchronous operations & completions	Asynchronous Completion Token
Enhancing server (re)configurability	Component Configurator
Minimizing unused server resources	Activator
Transparently parameterizing synchronization into components	Strategized Locking
Ensuring locks are released properly	Scoped Locking
Minimizing unnecessary locking	Thread-Safe Interface

Design Problems & Pattern-Oriented Solutions

- Many design problems faced developing JAWS can be resolved via POSA2 (& GoF) patterns

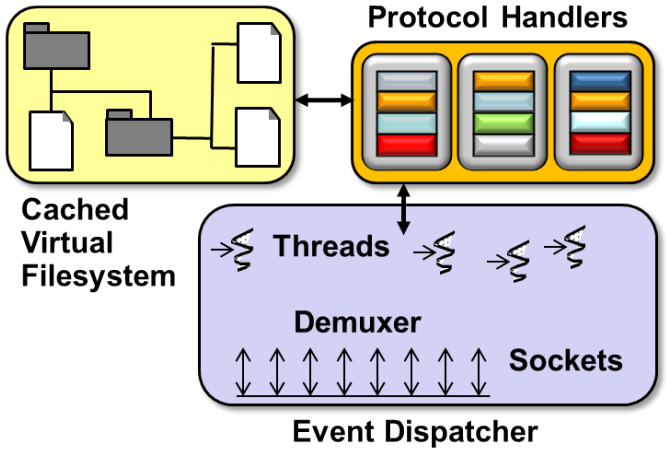


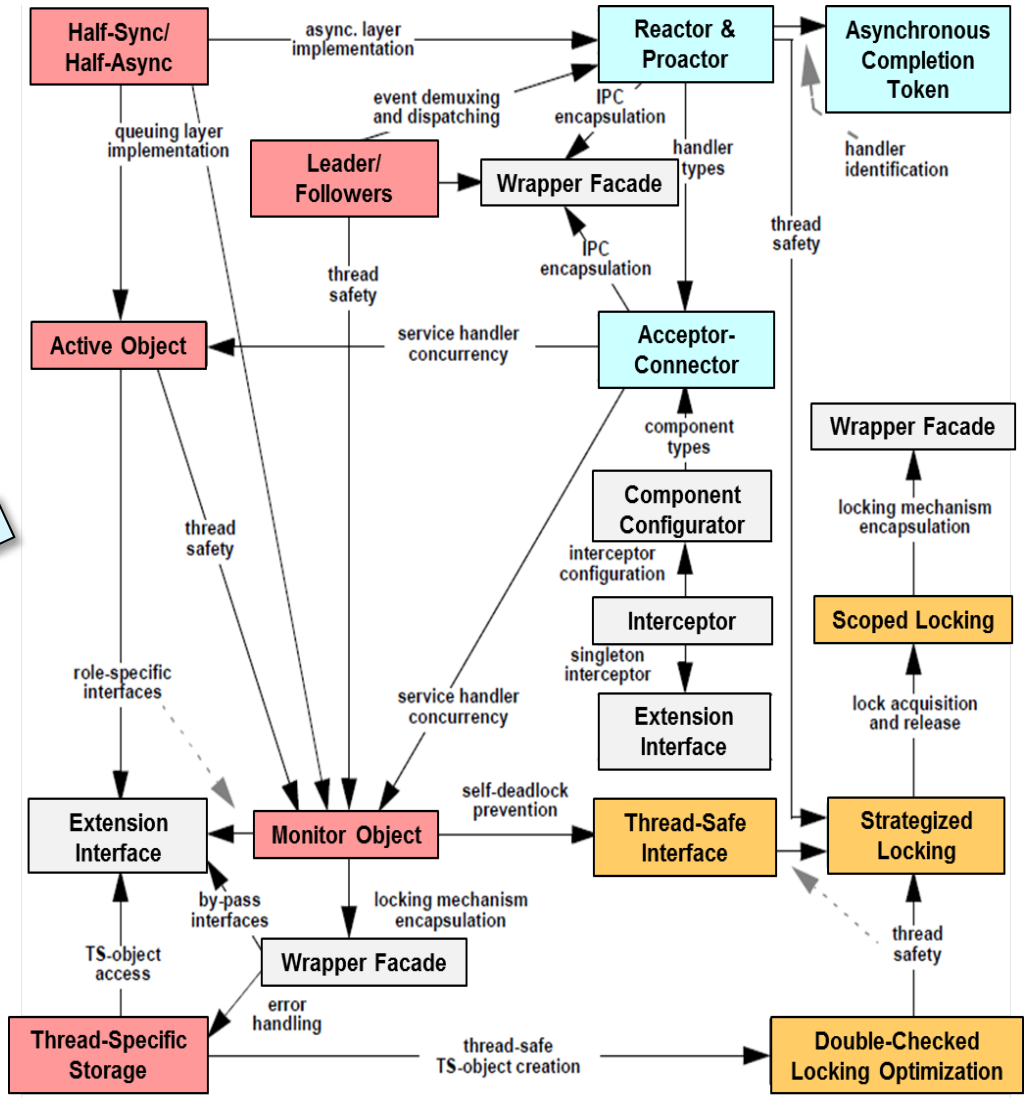


Design Problem	Pattern(s)
Encapsulating low-level OS APIs	Wrapper Facade
Decoupling event demuxing & connection management from protocol processing	Reactor & Acceptor-Connector
Scaling up performance via multi-threading	Half-Sync/Half-Async & Active Object
Synchronized request queue	Monitor Object
Minimizing multi-threading overhead	Leader/Followers
Using asynchronous I/O effectively	Proactor
Efficiently demuxing asynchronous operations & completions	Asynchronous Completion Token
Enhancing server (re)configurability	Component Configurator
Minimizing unused server resources	Activator
Transparently parameterizing synchronization into components	Strategized Locking
Ensuring locks are released properly	Scoped Locking
Minimizing unnecessary locking	Thread-Safe Interface

Summary

- POSA2 patterns direct developers towards *strategic* elements of concurrent & networked software
  - The impact of many accidental & inherent complexities can be alleviated if strategic elements are properly addressed

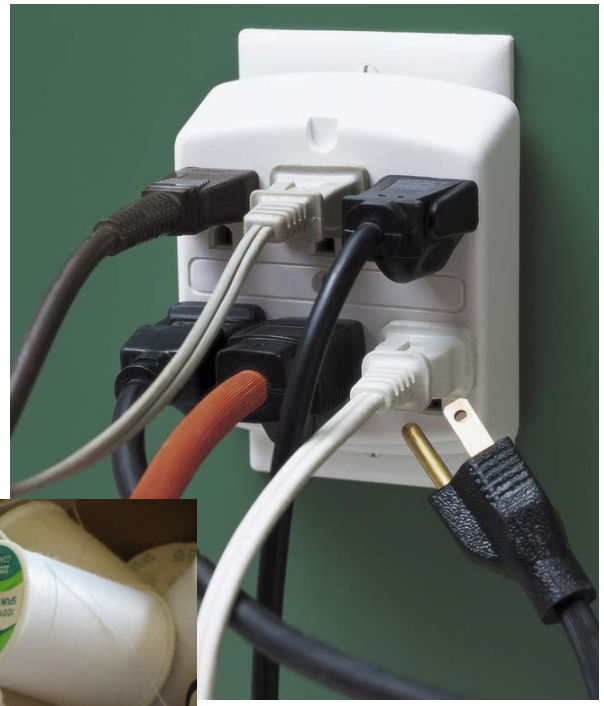




Naturally, these patterns aren't limited just to web servers!!

Summary

- POSA2 patterns direct developers towards *strategic* elements of concurrent & networked software
  - The impact of many accidental & inherent complexities can be alleviated if strategic elements are properly addressed
- POSA2 patterns also redirect developers from preoccupation with low-level OS & networking protocols & mechanisms
  - While having a solid grasp of these topics is important, they are *tactical* in scope

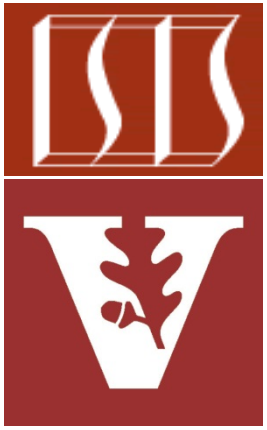


Summary

- POSA2 patterns direct developers towards *strategic* elements of concurrent & networked software
  - The impact of many accidental & inherent complexities can be alleviated if strategic elements are properly addressed
- POSA2 patterns also redirect developers from preoccupation with low-level OS & networking protocols & mechanisms
  - While having a solid grasp of these topics is important, they are *tactical* in scope
- Combining POSA2 patterns with patterns from GoF & other sources helps create more powerful & comprehensive pattern languages

Overview of JAWS Web Server Case Study: Part 3

Douglas C. Schmidt
d.schmidt@vanderbilt.edu
www.dre.vanderbilt.edu/~schmidt



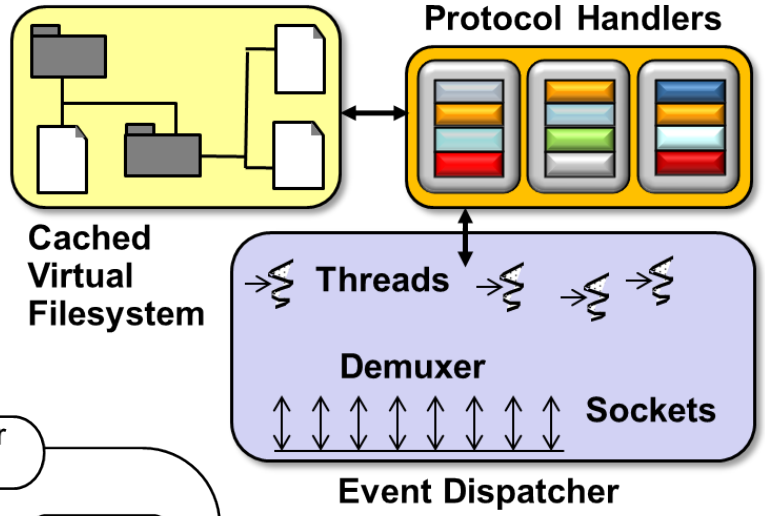
Professor of Computer Science

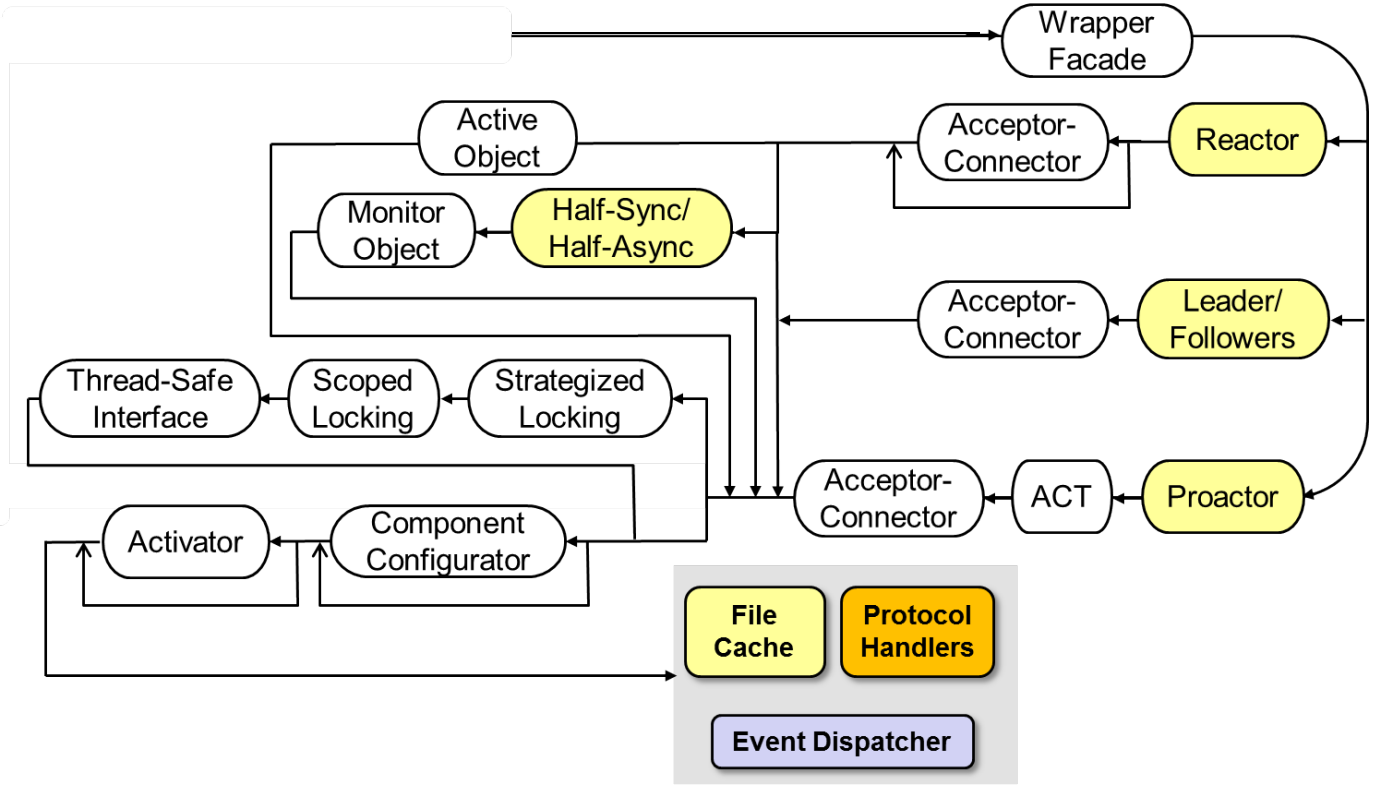
Institute for Software Integrated Systems

Vanderbilt University
Nashville, Tennessee, USA

Topics Covered in this Part of the Module

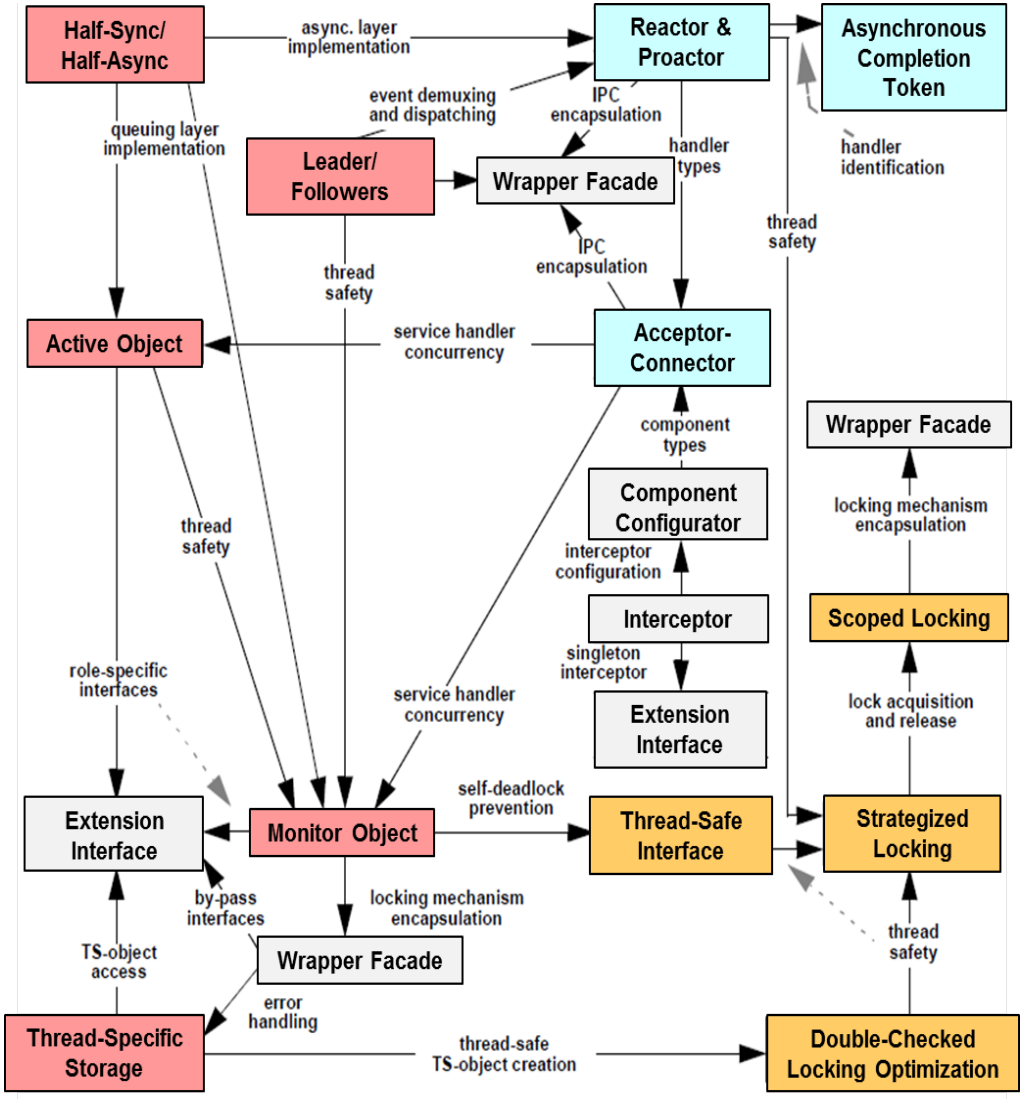
- Describe the pattern-oriented JAWS web server case study
- Summarize the patterns in the JAWS web server design
- Present a pattern language for the JAWS web server





Recap of Pattern Languages

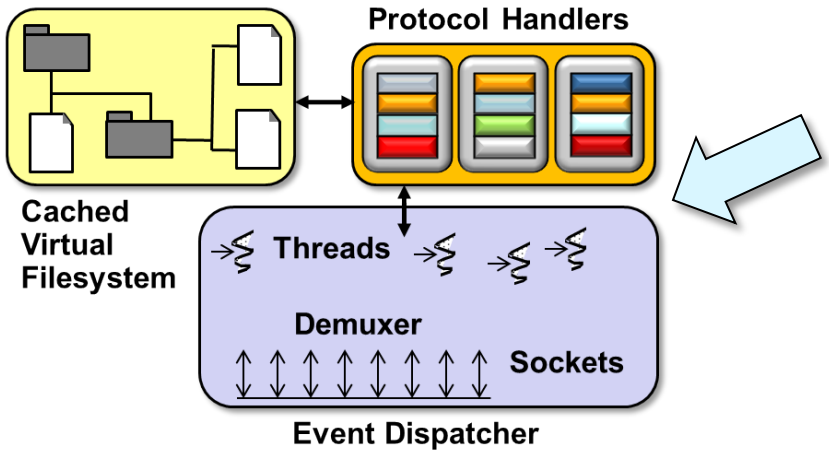
- Pattern languages are groups of related patterns that define a vocabulary & process for the orderly resolution of development problems in particular domains



en.wikipedia.org/wiki/Pattern_language has broader pattern language usages

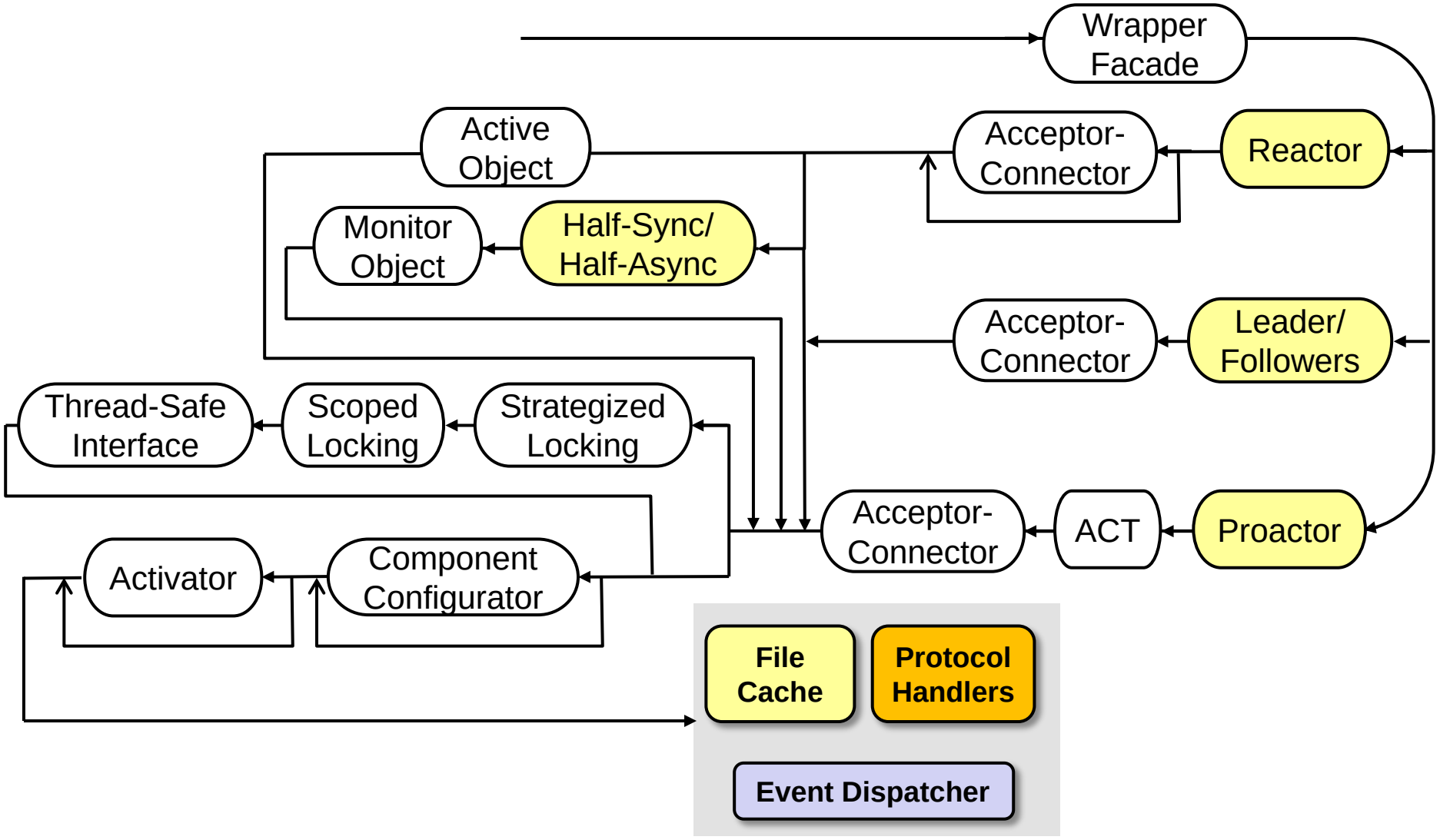
Recap of Pattern Languages

- Pattern languages are groups of related patterns that define a vocabulary & process for the orderly resolution of development problems in particular domains
- The patterns in a pattern language build on each other to help generate a system by documenting a successive progression of design decisions, transformations, & *alternatives*



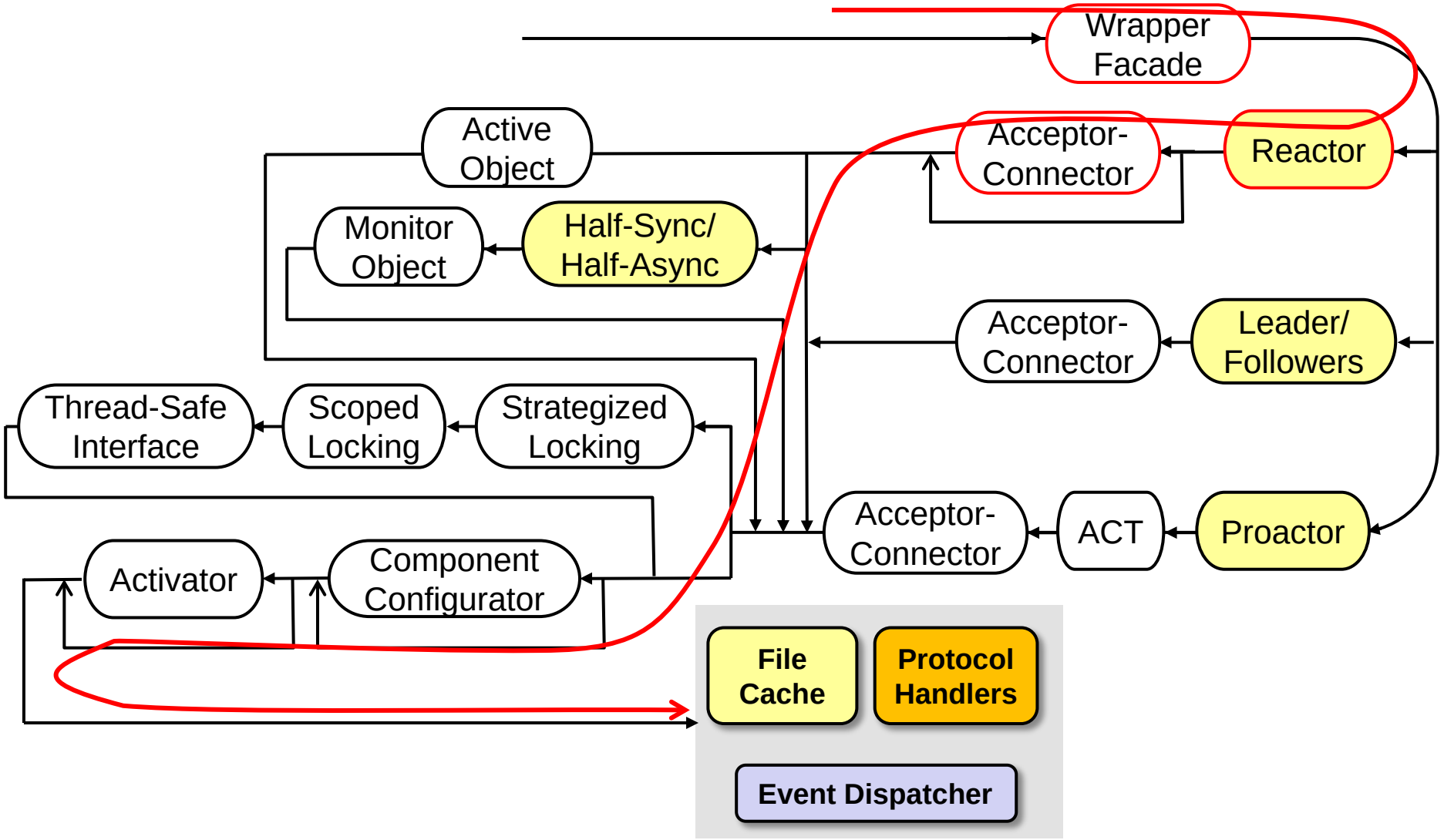
Design Problem	Pattern(s)
Encapsulating low-level OS APIs	Wrapper Facade
Decoupling event demuxing & connection management from protocol processing	Reactor & Acceptor-Connector
Scaling up performance via multi-threading	Half-Sync/Half-Async & Active Object
Synchronized request queue	Monitor Object
Minimizing multi-threading overhead	Leader/Followers
Using asynchronous I/O effectively	Proactor
Efficiently demuxing asynchronous operations & completions	Asynchronous Completion Token
Enhancing server (re)configurability	Component Configurator
Minimizing unused server resources	Activator
Transparently parameterizing synchronization into components	Strategized Locking
Ensuring locks are released properly	Scoped Locking
Minimizing unnecessary locking	Thread-Safe Interface

A Pattern Language for the JAWS Web Server



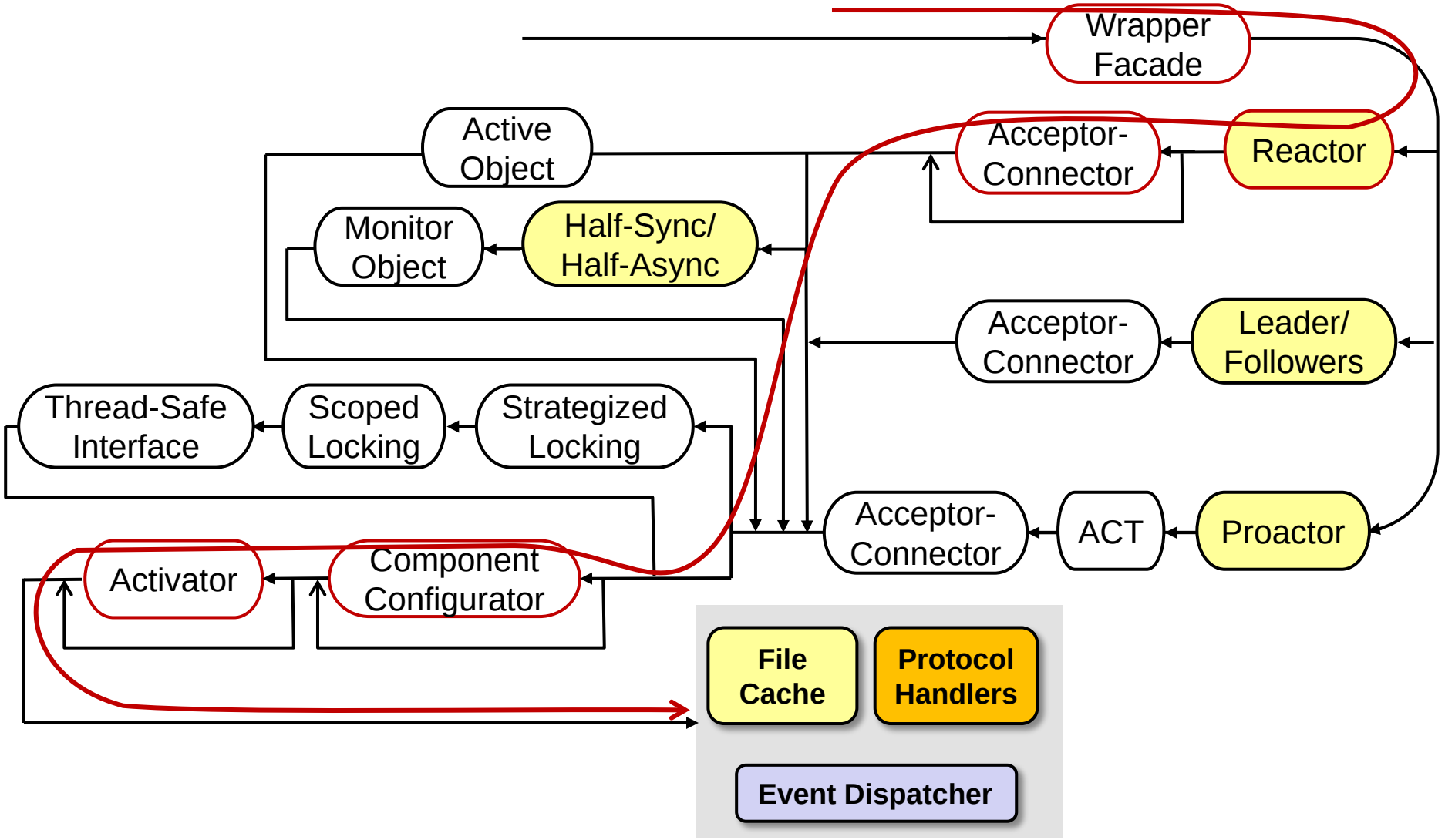
This pattern language for the JAWS web server is not comprehensive

A Pattern Language for the JAWS Web Server



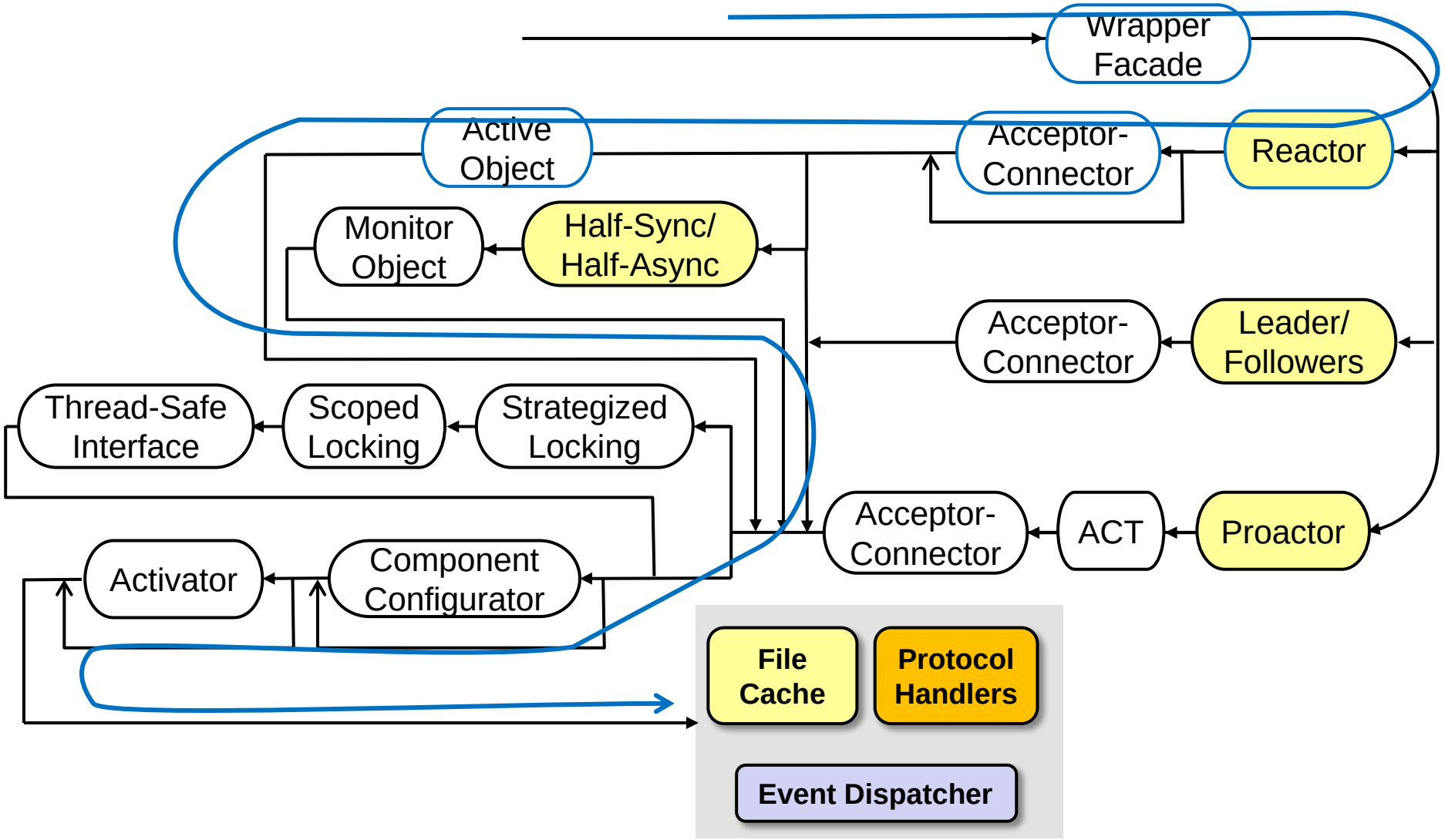
Simple, reactive, but non-scalable

A Pattern Language for the JAWS Web Server



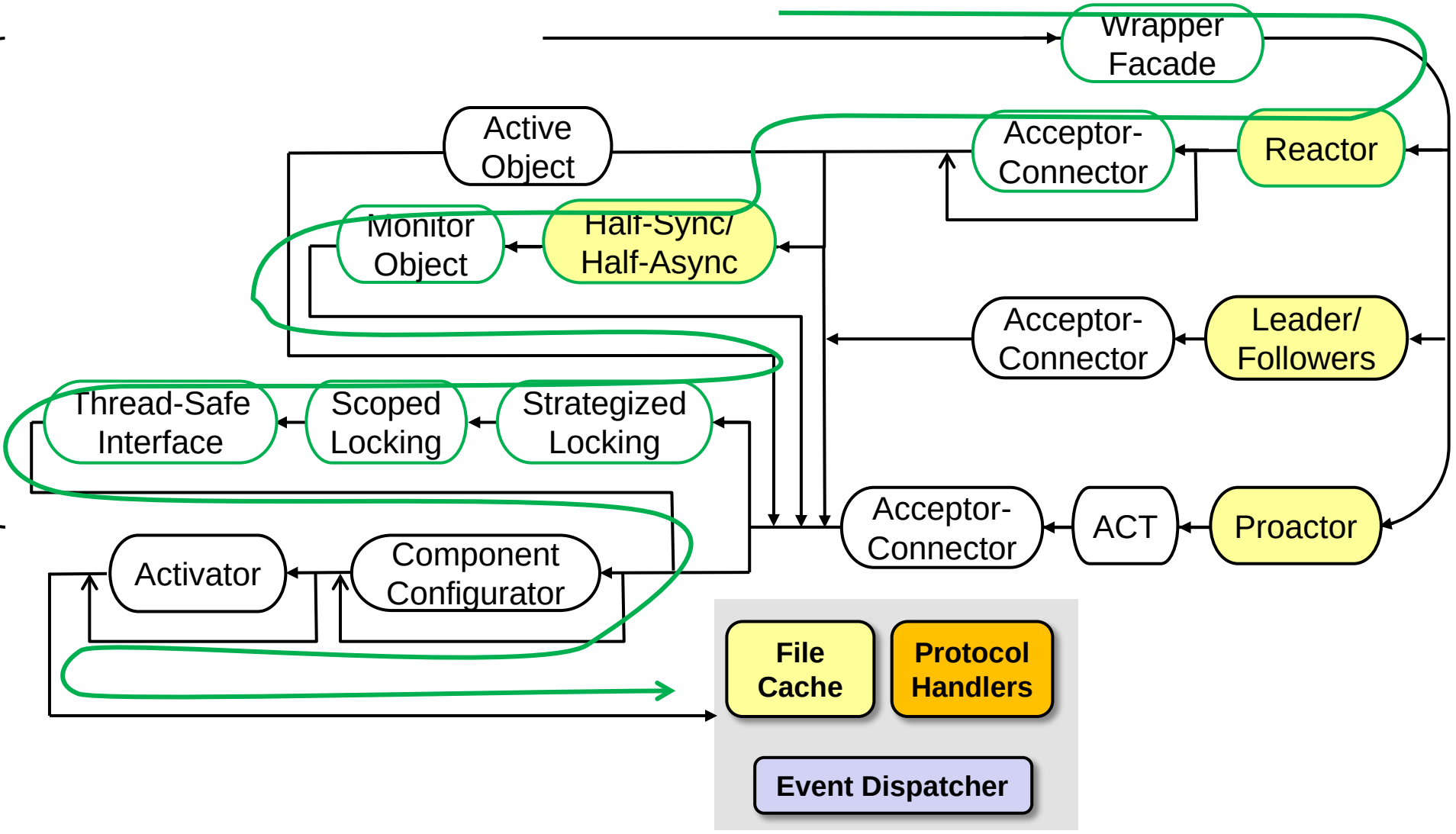
Simple, reactive, flexible, & more (coarse-grained) scalable

A Pattern Language for the JAWS Web Server



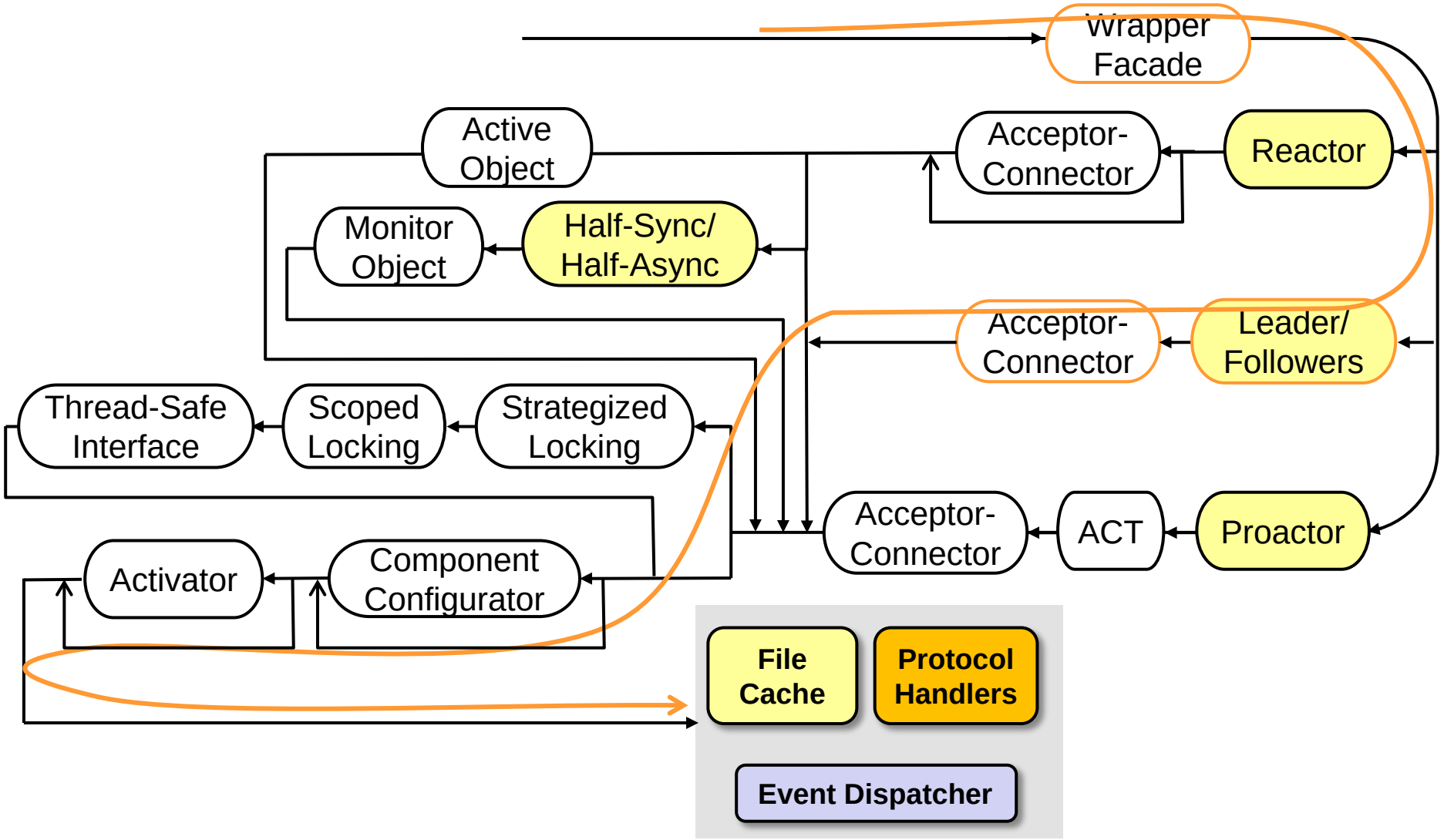
Simple, concurrent, & more (fine-grained) scalable

A Pattern Language for the JAWS Web Server



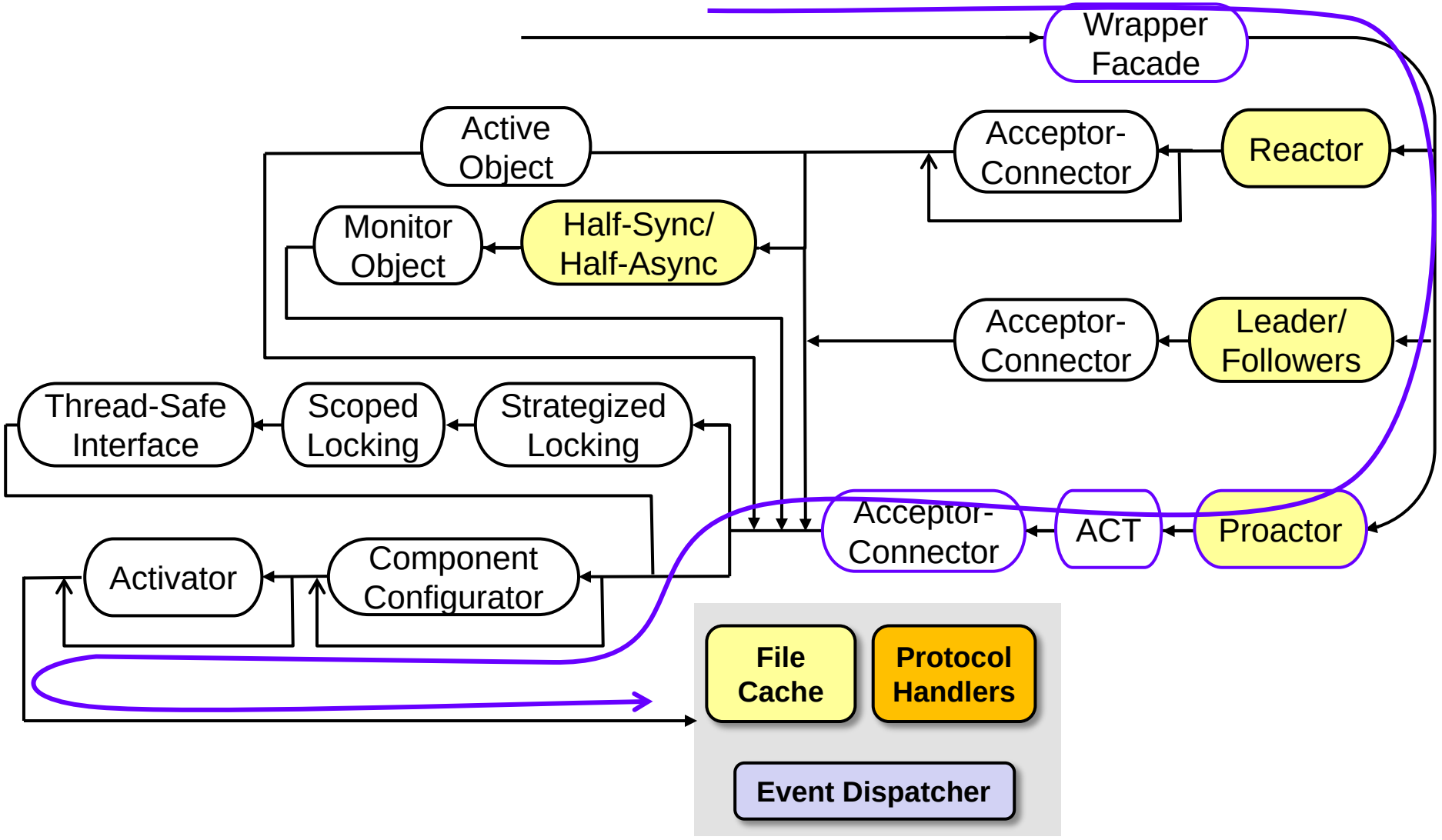
Concurrent & scalable, but may be less predictable

A Pattern Language for the JAWS Web Server



Concurrent, predictable, but may be less scalable

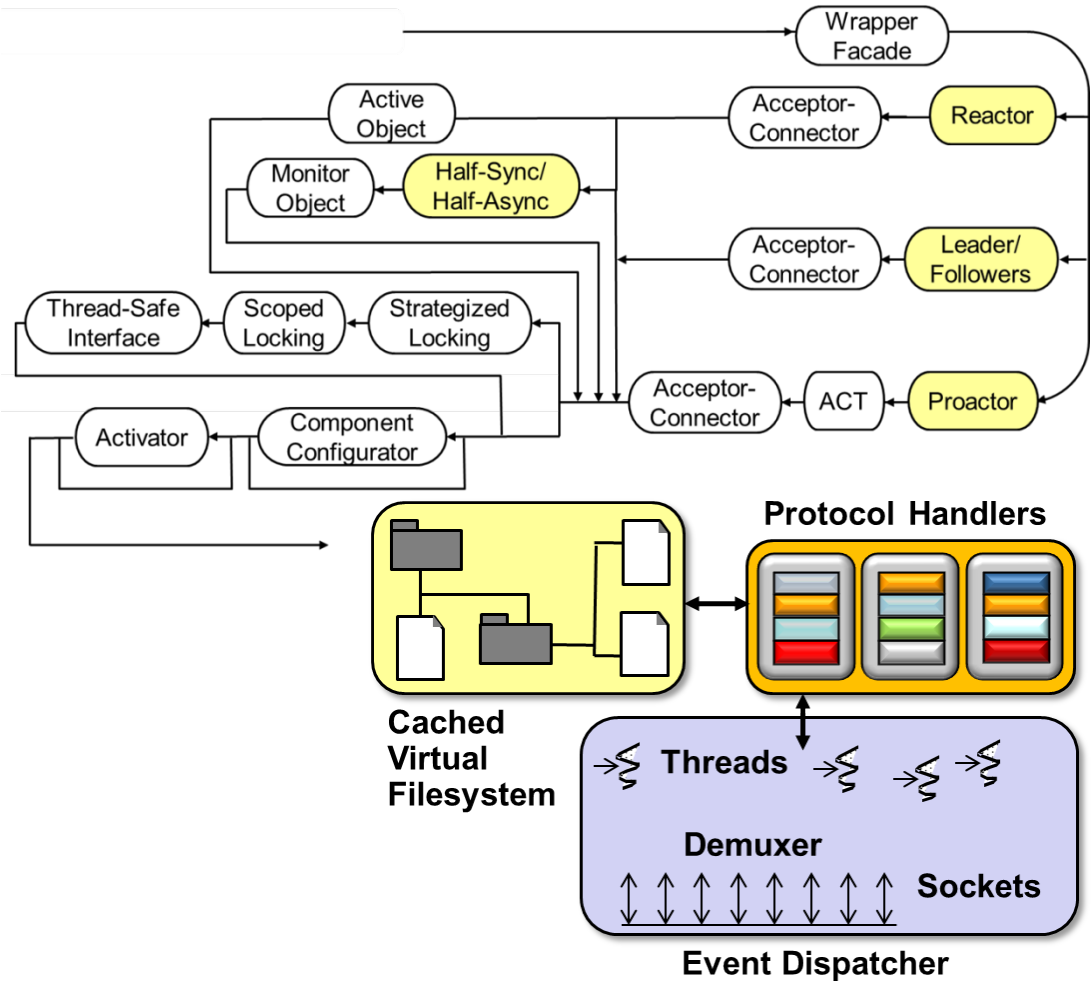
A Pattern Language for the JAWS Web Server



Asynchronous, concurrent, scalable, but limited portability

Summary

- The pattern language applied to JAWS helps to systematically document & evaluate alternative paths through the design space



Summary

- The pattern language applied to JAWS helps to systematically document & evaluate alternative paths through the design space
- Wrapper facades & frameworks provided by ACE middleware can implement this pattern language