ACE Overview
ADAPTIVE Communication Environment

Kyoungho An
Dept. of EECS, Vanderbilt University
July 26, 2012

Presentation Roadmap

- ACE Overview
- Benefits of Using ACE
- The Structure and Functionality of ACE
  - The ACE OS Adapter Layer
  - C++ Wrapper Facades for OS Interfaces
  - Frameworks
  - Distributed Services and Components
  - High-level Distributed Computing Middleware Components
- Reference

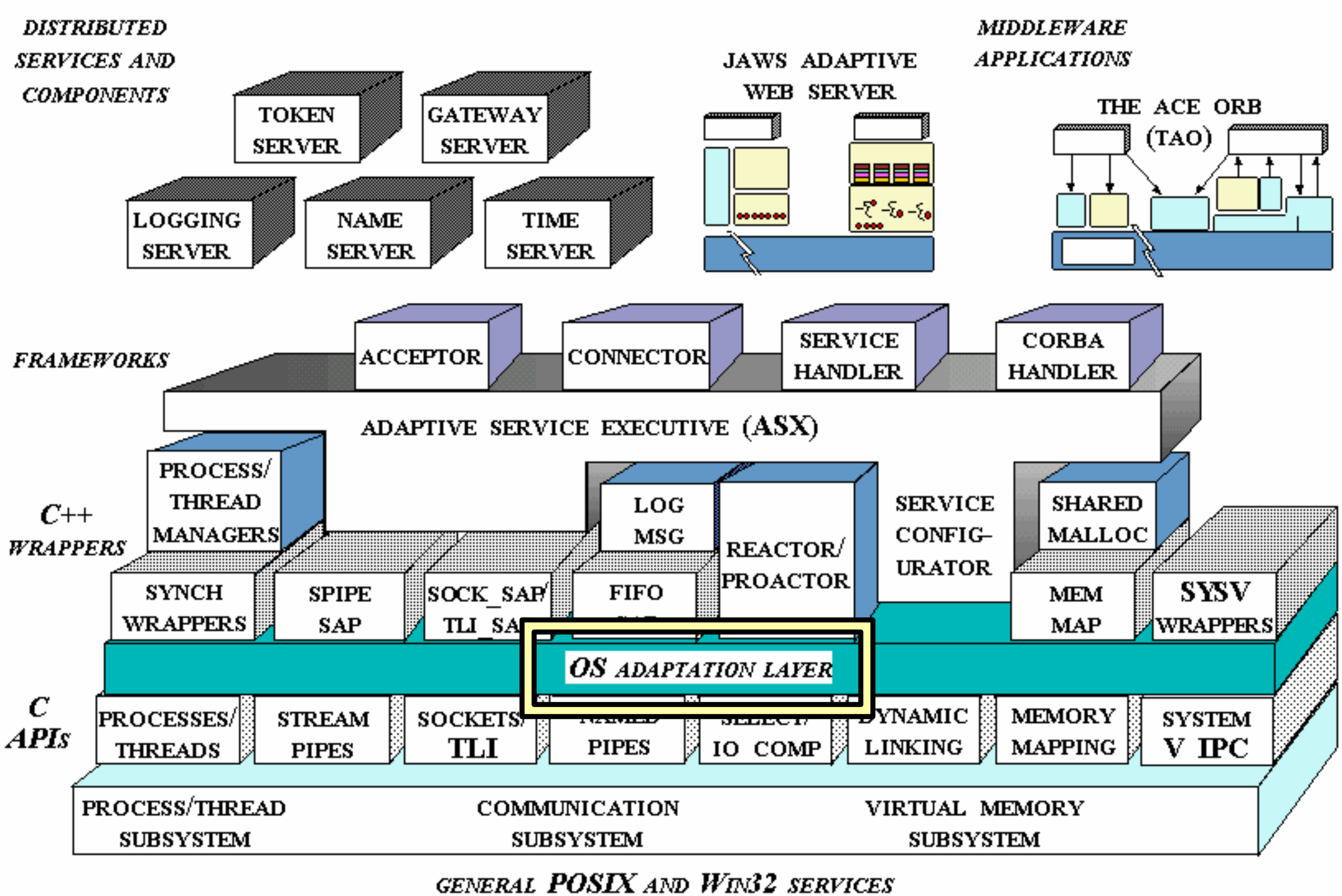
ACE Overview

- Object-oriented network framework implementing core design patterns for concurrent network software
- Provide C++ wrapper façades and framework components across various OS platforms
- Communication software tasks provided by ACE
  - Event demultiplexing and event handler dispatching
  - Signal handling
  - Service initialization
  - IPC
  - Shared memory management
  - Dynamic reconfiguration of distributed services
  - Concurrent execution and synchronization

Benefits of Using ACE

- Increased portability
  - Easy to port applications to other OS platforms
- Increased software quality
  - flexibility, extensibility, reusability, modularity through using key design patterns
- Increased efficiency and predictability
  - Support a wide range of application QoS requirements
  - Low latency for delay-sensitive applications
  - High performance for bandwidth-intensive applications
- Provide standard high-level middleware
  - The ACE ORB (TAO), which is an open-source standard-compliant implementation of CORBA

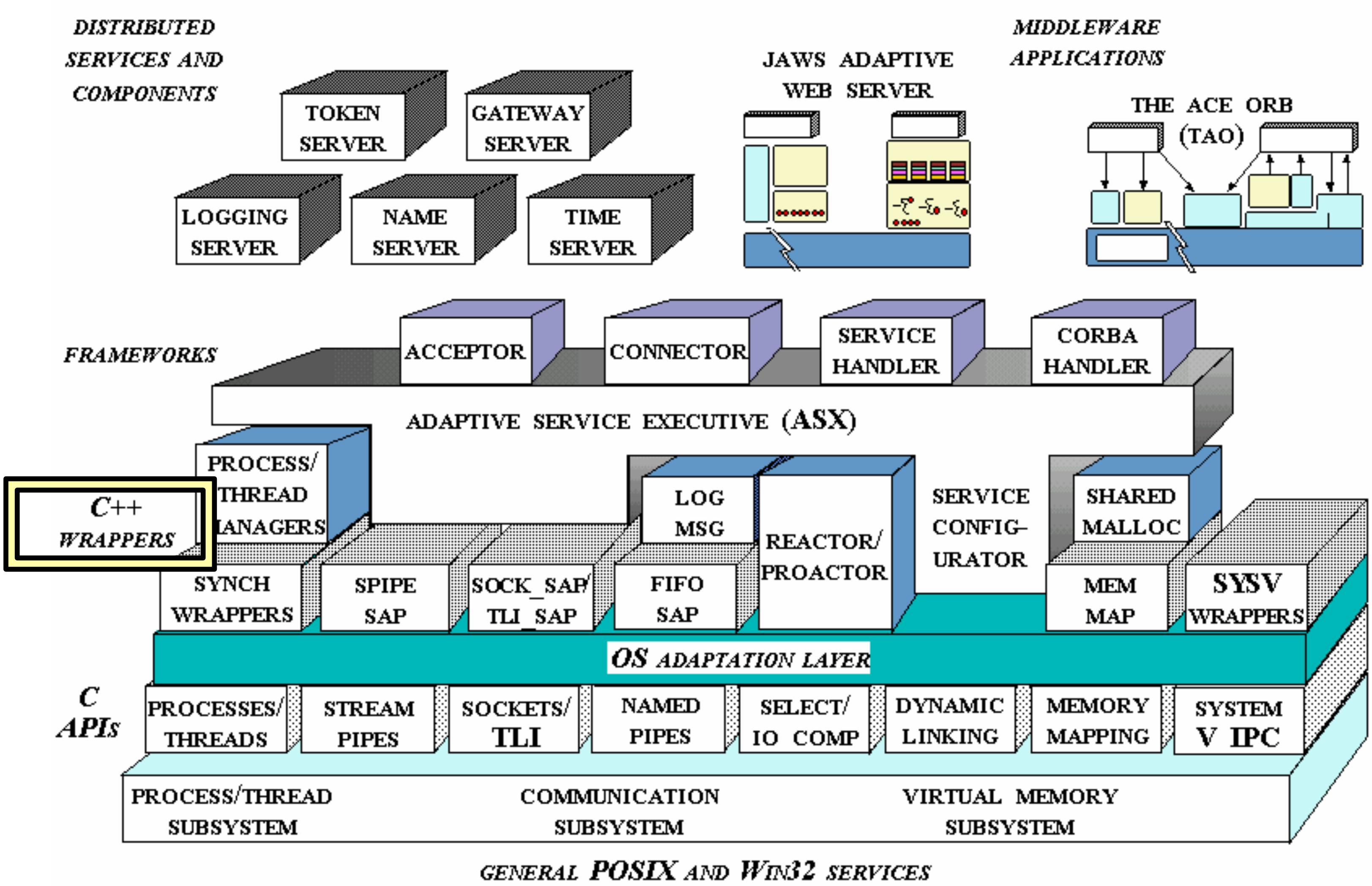
The Structure and Functionality of ACE



The ACE OS Adapter Layer

- Reside directly atop the native OS APIs written in C
- Shield the other layers and components in ACE from platform-specific dependencies associated with the following OS APIs
  - Concurrency and synchronization
  - IPC and shared memory
  - Event demultiplexing mechanisms
  - Explicit dynamic linking
  - File system mechanisms
- ACE ported OS platforms
  - Windows, MacOS X, Linux, RTOSs, iOS, Android, etc.

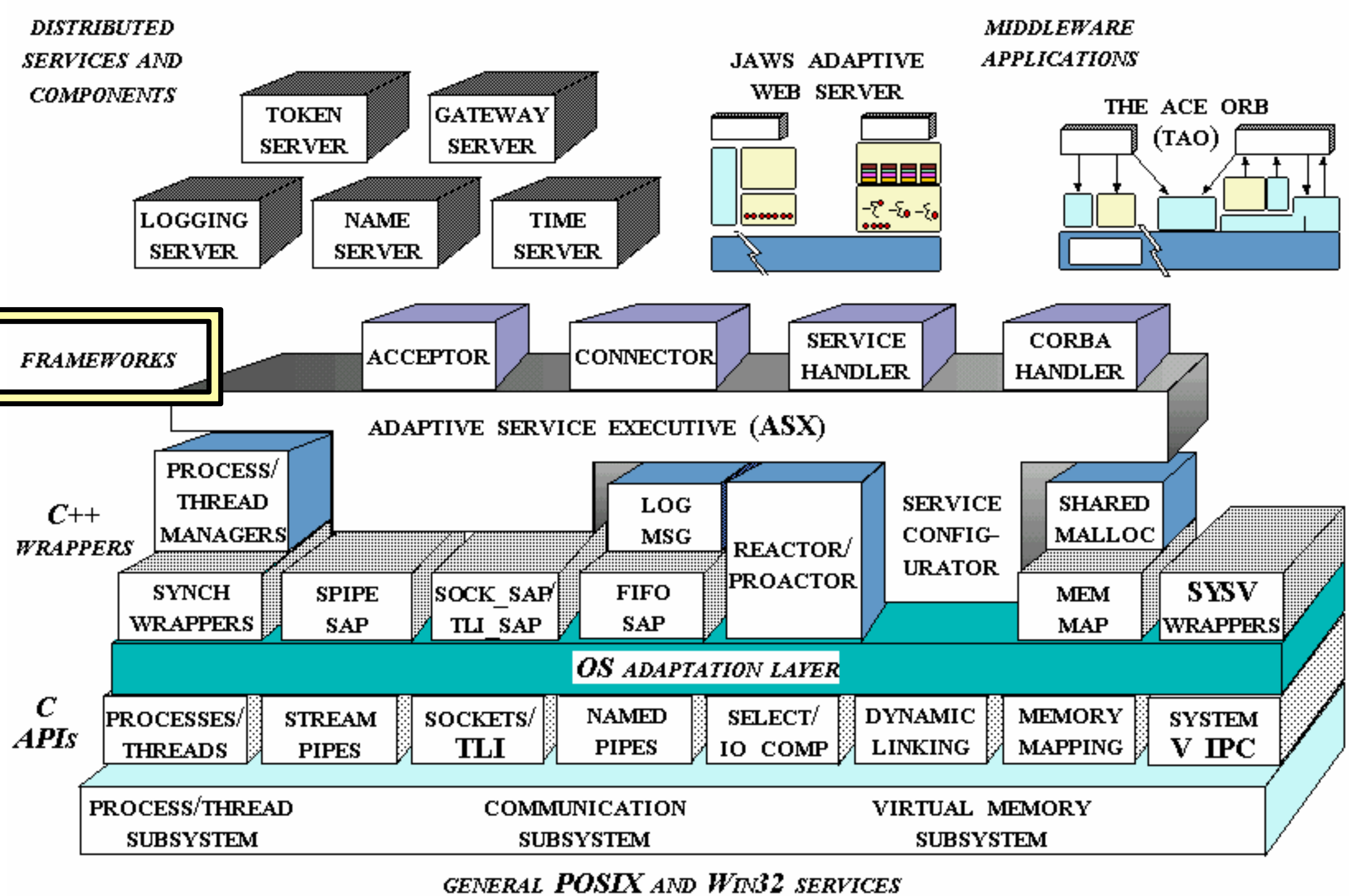
The Structure and Functionality of ACE



C++ Wrapper Facades for OS Interfaces

- Possible to program directly atop ACE's OS adaptation layer
- However… most ACE developers use the C++ wrapper façade layer
- Simplify application development by providing typesafe C++ interfaces that encapsulate and enhance the following
  - Concurrency and synchronization components
  - IPC and filesystem components
  - Memory management components
- C++ wrappers are strongly typed
  - Detect system violations at compile-time rather than run-time

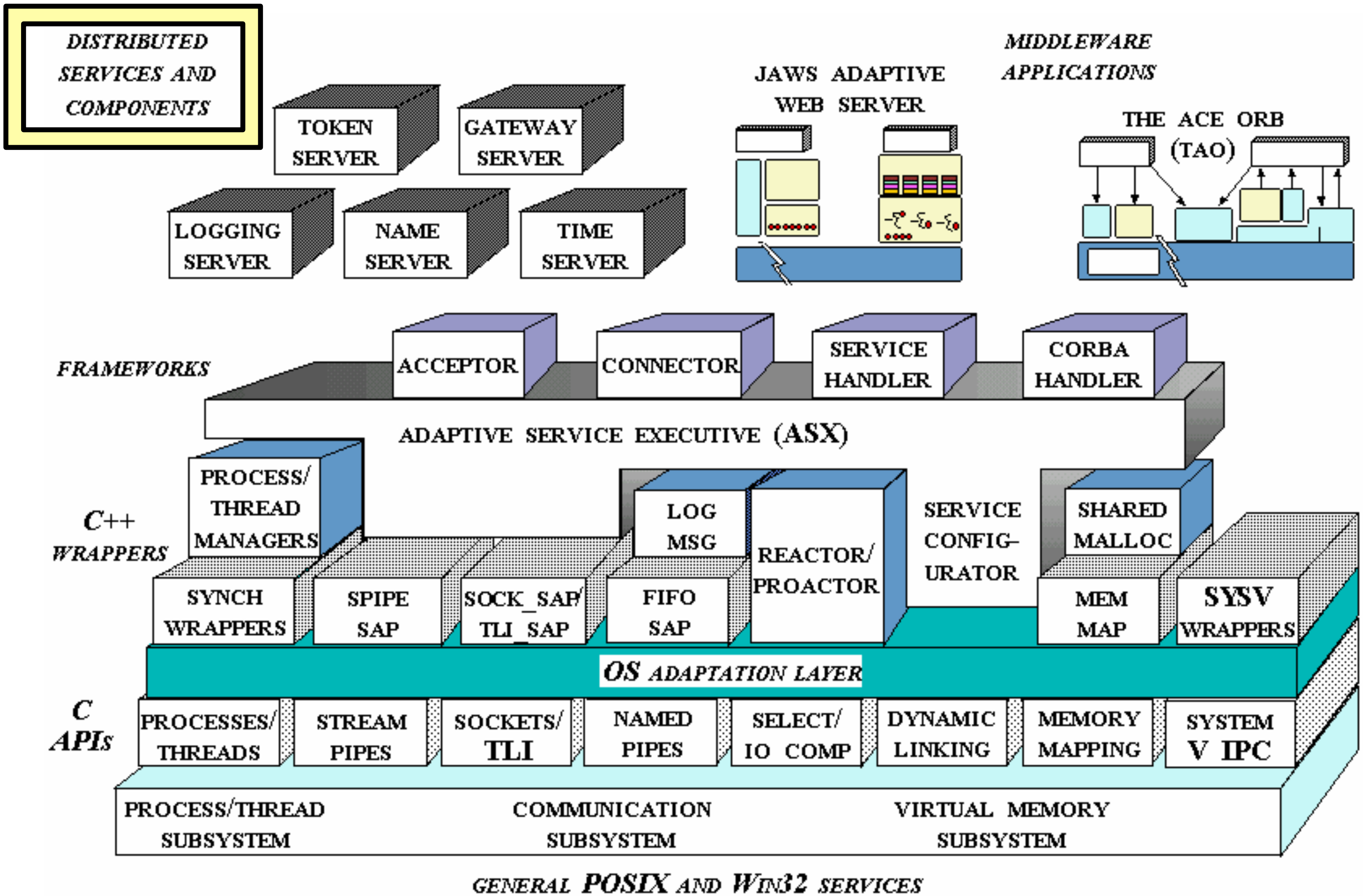
The Structure and Functionality of ACE



Frameworks

- ACE Framework
  - Event demultiplxeing components
    - Reactor, Proactor
  - Service initialization components
    - Acceptor, Connector
  - Service configuration components
    - Service Configurator
  - Hierarchically-layered stream components
    - Streams
- ACE Framework Implementation
  - C++ language features (templates, inheritance, dynamic binding)
  - Design patterns (Abstract Factory, Strategy, Service Configurator)
  - OS mechanisms (multi-treading, dynamic linking)

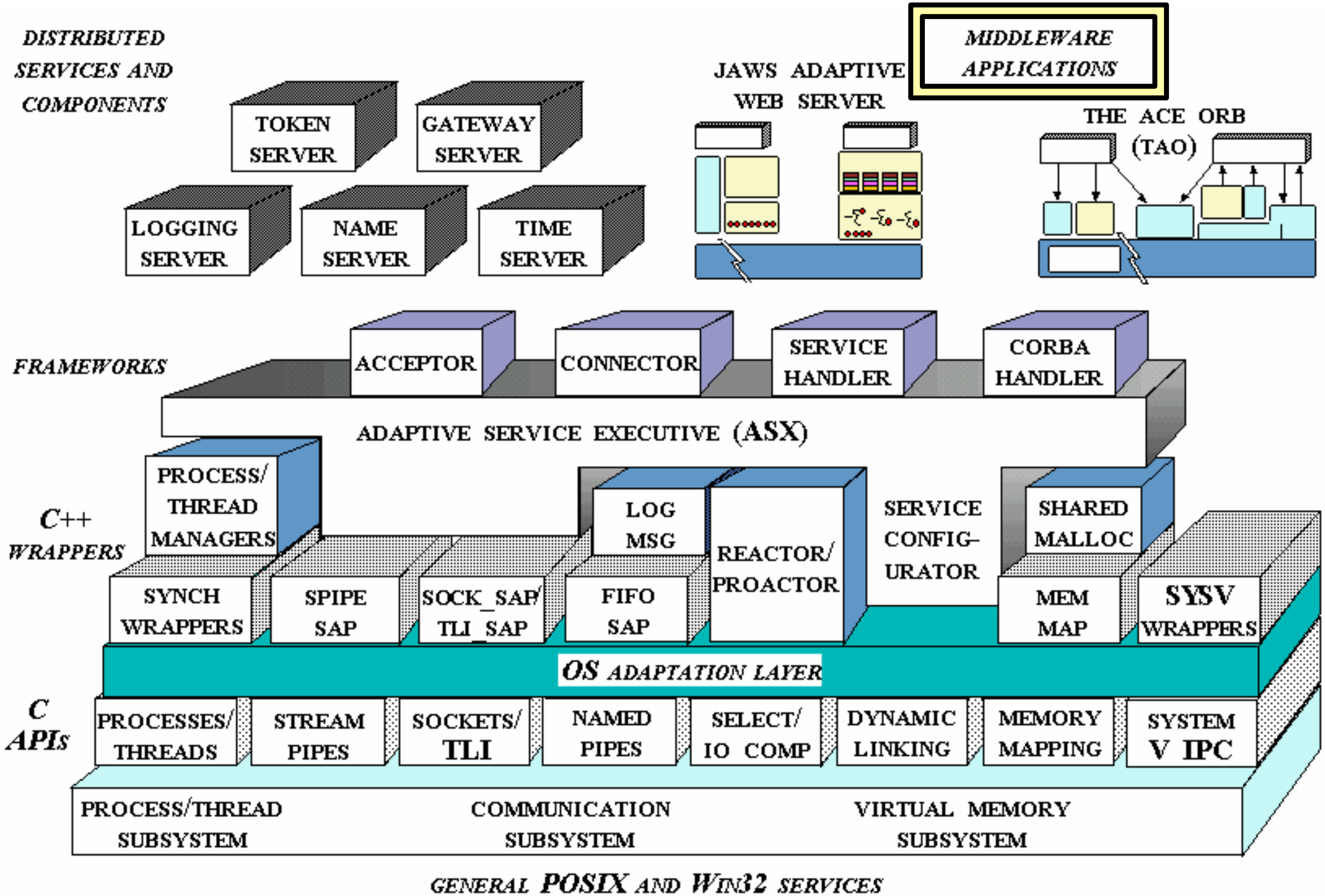
The Structure and Functionality of ACE



Distributed Services and Components

- Provide a standard library of distributed services
- Not part of the ACE framework library
- However… play two roles in ACE
  - Factoring out reusable distributed application building blocks
    - naming, event routing, logging, time synchronization
  - Demonstrating common use-cases of ACE components

The Structure and Functionality of ACE



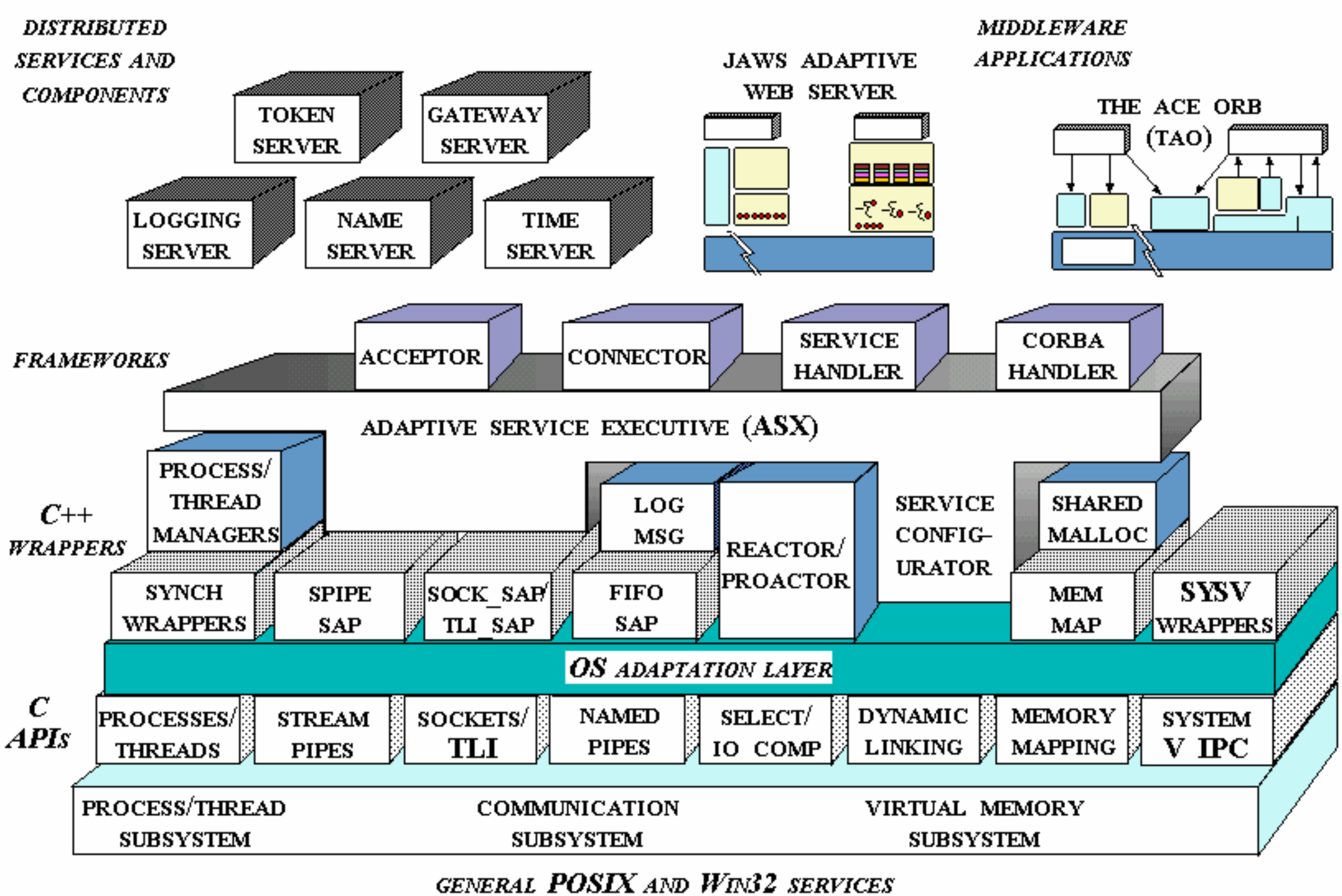
Distributed Middleware Components

- Developing robust, extensible, and efficient communication applications is challenging
  - Network addressing and service identification
  - Encryption, compression, and network byte-ordering conversions between heterogeneous end-systems
  - Process and thread creation and synchronization
  - Library interfaces to IPC mechanisms
- Higher-level distributed middleware (CORBA, DCOM, RMI)
  - Alleviate complexity of developing communication applications
    - Authentication, authorization, and data security
    - Service location and binding
    - Service registration and activation
    - Demultiplexing and dispatching in response to events
    - Implementing message framing atop byte stream-oriented communication protocol like TCP

Distributed Middleware Components

- ACE의 higher-level middleware applications
  - The ACE ORB (TAO)
    - Real-time implementations of CORBA using ACE
    - Based on the standard OMG CORBA reference model
    - Overcome the shortcomings of conventional ORBs for high-performance and real-time applications
  - JAWS
    - High performance, adaptive Web server using ACE
    - JAWS components and frameworks
      - Concurrency Strategy (Thread per request vs. Thread pool)
      - I/O Strategy (synchronous vs. asynchronous)
      - Protocol Handlers (HTTP 1.0 vs. HTTP 1.1)
      - Cached Virtual File System (LRU vs. LFU)

The Structure and Functionality of ACE



Questions?

Reference

- http://www.cs.wustl.edu/~schmidt/ACE-overview.html