Distributed Systems

Bina Ramamurthy

Introduction

- Distributed system is the one in which hardware and software components at networked computers communicate and coordinate their activity by sharing resources such as information, data, compute cycles, bandwidth and storage.
- Examples: Internet, intranet, grid and mobile computing systems.

Internet

- Internet is a very large distributed system.
- Interconnection of a collection of heterogeneous networks of computers.
- Protocols: IP, TCP, HTTP
- Services: world wide web (www), file transfers (ftp), email, etc.

Fundamental terms: Protocol

- **Protocol** is a set of rules that end points in a telecommunication system use when exchanging information.
  - IP: Internet protocol defines an unreliable packet transfer protocol.
  - TCP: Transmission Control Protocol builds on IP to define a reliable data delivery protocol.
  - LDAP: Lightweight Directory Access Protocol builds on TCP to define a query-response protocol for querying the state of a remote database.
  - HTTP: Hyper Text Transfer Protocol builds on TCP to facilitate hyper-text document exchange.

Fundamental terms: Service

- Service is a network-enabled entity that provides a specific capability.
- Service = Protocol + Behavior
- A service definition permits many implementations.
- Examples: ability to move files, create processes, verify access rights
- An FTP server speaks File Transfer Protocol and supports remote read and write access to a collection of files.

Fundamental terms: API

- **Application Program Interface (API)** defines a standard interface for invoking a specified set of functionality.
- Examples: The Generic Security Service (GSS) API defines standard functions for verifying identity of communicating parties, encrypting messages and so forth.

Fundamental terms: SDK

- **Software Development Kit (SDK)** denotes a set of code designed to be linked with, and invoked from within, an application program to provide specified functionality.
- An SDK typically implements an API.
- Example: Different SDKs implement GSS-API using the Kerberos or PKI protocols, respectively.

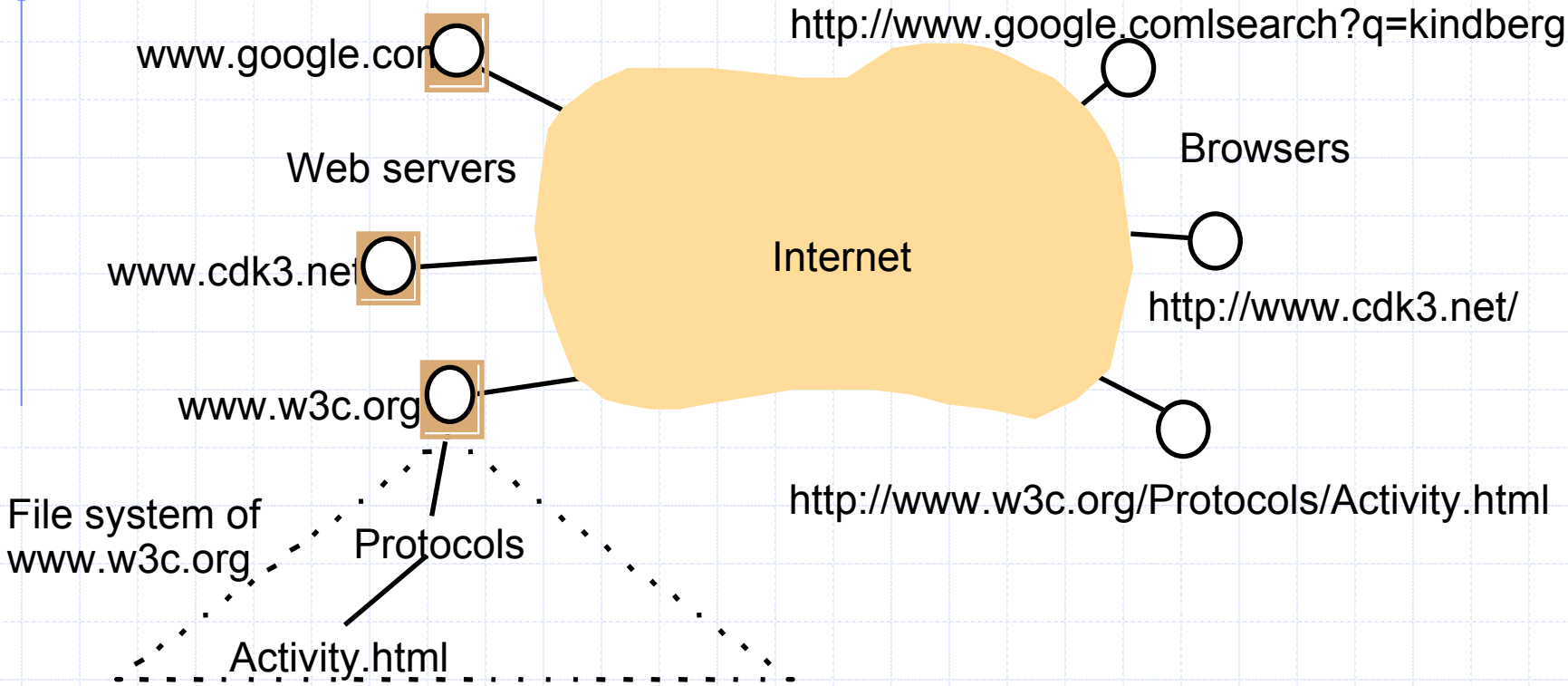
Major Challenges

- Heterogeneity of components
- Openness
  - interfacing and
  - addition and removal of components
- Security
- Scalability : ability to work well when number of users increases
  - Failure handling
  - Concurrency
  - Transparency
- Others

Client/Server

- Server: refers to a process on a networked computer that accepts requests from other (local or remote) processes to perform a service and responds appropriately.
- Client: requesting process in the above is referred to as the client.
- Request and response are in the form of messages.
- Client is said to invoke an operation on the server.
- Many distributed systems today are constructed out of interacting clients/servers.

Web servers and web browsers



Reading assignment: Section 1.3.1

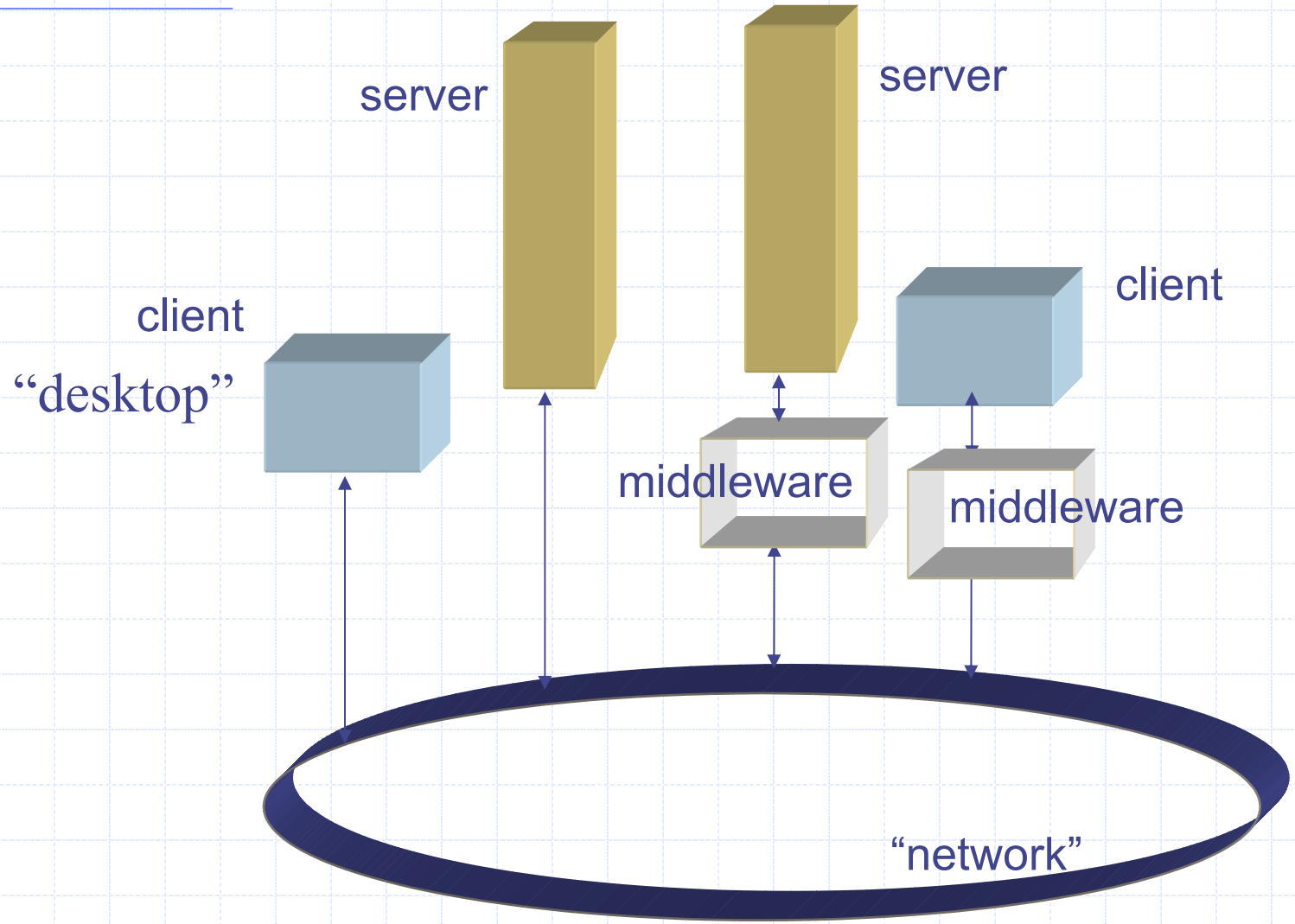
Transparencies

- ***Access transparency***: **enables local and remote resources to be accessed using identical operations.**
- ***Location transparency***: **enables resources to be accessed without knowledge of their location.**
- ***Concurrency transparency***: **enables several processes to operate concurrently using shared resources without interference between them.**
- ***Replication transparency***: **enables multiple instances of resources to be used to increase reliability and performance without knowledge of the replicas by users or application programmers.**
- ***Failure transparency***: **enables the concealment of faults, allowing users and application programs to complete their tasks despite the failure of hardware or software components.**
- ***Mobility transparency***: **allows the movement of resources and clients within a system without affecting the operation of users or programs.**
- ***Performance transparency***: **allows the system to be reconfigured to improve performance as loads vary.**
- ***Performance transparency***: **allows the system and applications to expand in scale without change to the system structure or the application algorithms.**

Middleware (as defined by NSF)

- Middleware refers to the software which is common to multiple applications and builds on the network transport services to enable ready development of new applications and network services.
- Middleware typically includes a set of components such as resources and services that can be utilized by applications either individually or in various subsets.
- Examples of services: Security, Directory and naming, end-to-end quality of service, support for mobile code.
- OMG's CORBA defines a middleware standard.
- OMG: Object Management Group
- CORBA: Common Object Request Broker Architecture

Middleware


server
server
client
client
"desktop"
middleware
middleware
"network"

Summary

- In this course, we will study distributed systems at the middleware level: how to define, design and implement services, how to use the middleware services in a distributed application.
- We will study Java RMI as a case study for simple distributed system.
- We will also introduce “webservices” API.