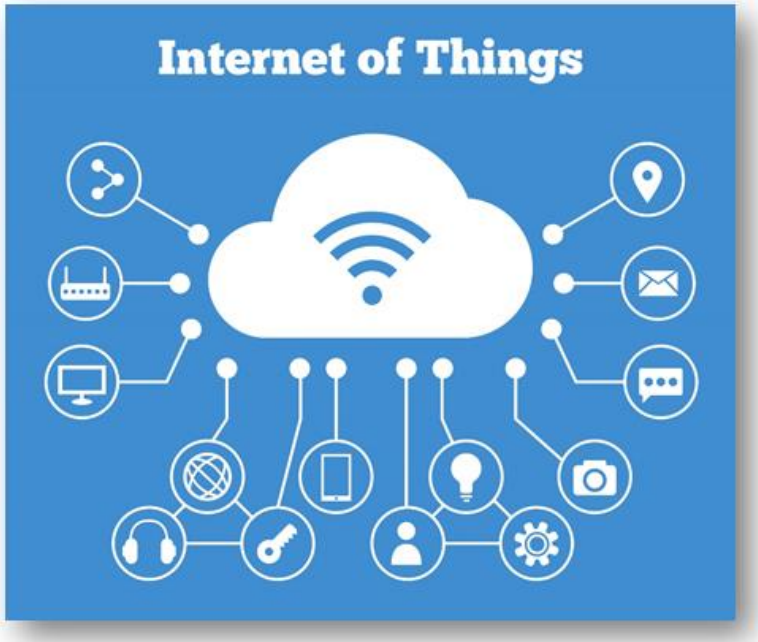
HTTP & REST

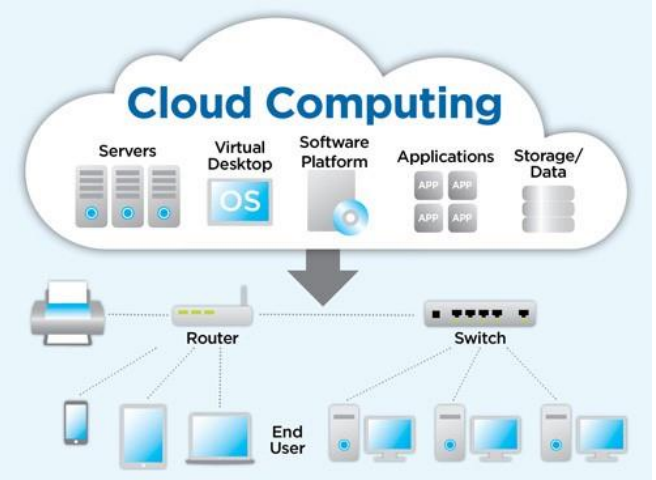
COCSC20

What is Cloud Computing?



What is Cloud Computing?

- Delivery of computing **services**
  - servers
  - storage
  - analytics
  - databases
  - networking
  - and much more...



- Another definition: network-based computing taking place over the Internet, while hiding complexity of underlying infrastructure using simple APIs.

What is Cloud Computing?

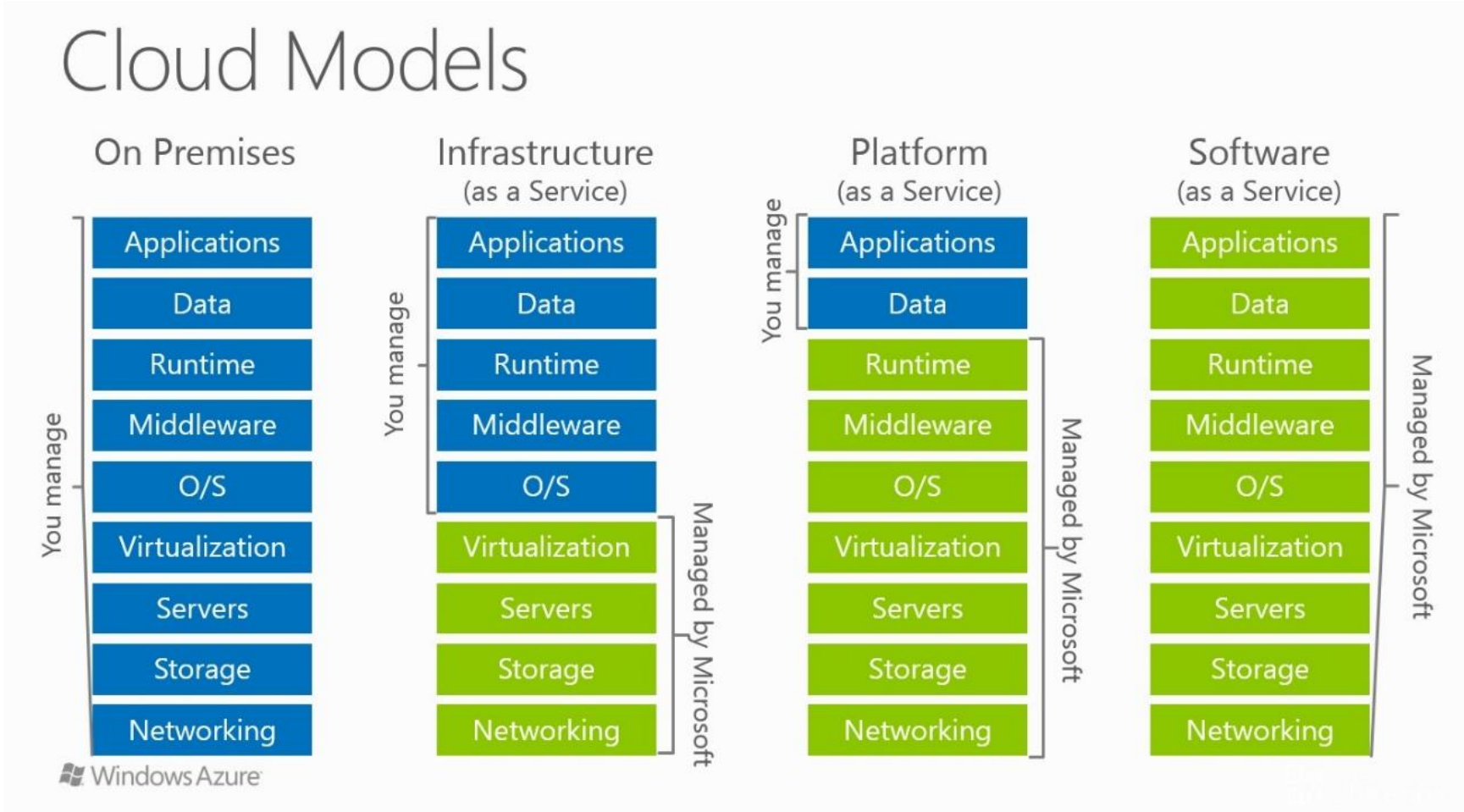
- Collection/group of **integrated and networked hardware, software, and Internet infrastructure** (called a platform)

- Platforms provide **on demand** services that are **always on** and **accessible anytime** and **anywhere**

What is Cloud Computing?

- Advantages:
  - New applications
  - Anytime/anywhere access
  - Homogeneity
  - Virtualization
  - Resilient
  - Cost
  - Sharing, collaboration
  - Management/maintenance
  - Security
  - …

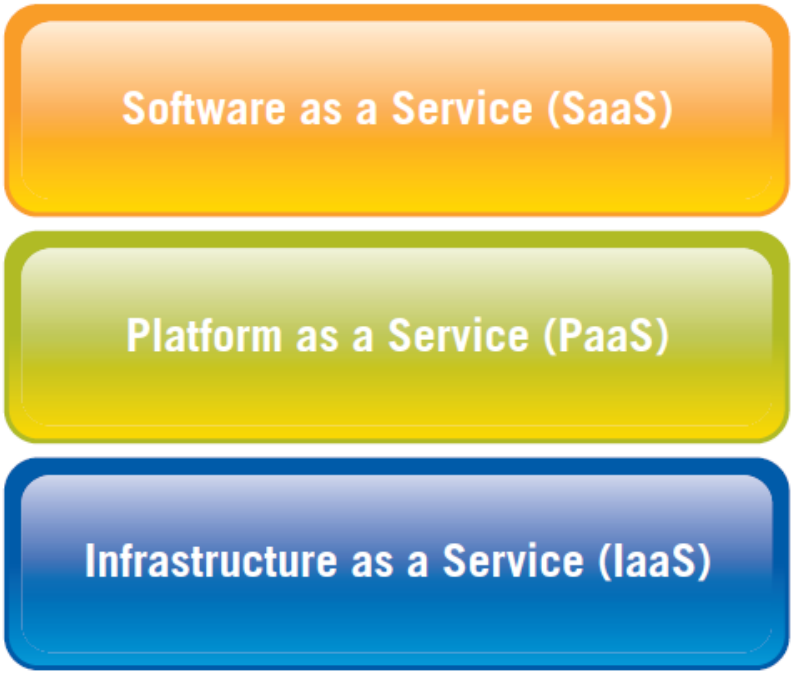
Cloud Models: IaaS, PaaS, SaaS



Definitions

- **Virtualization:** creation of a virtual resource such as a server, desktop, operating system, file, storage, or network
- **Middleware:** software that acts as a bridge between an operating system or database and applications, especially on a network
- **Runtime:** software designed to support the execution of computer programs

IaaS, PaaS, SaaS



Enduser application is delivered as a service. Platform and infrastructure is abstracted, and can deployed and managed with less effort.

Application platform onto which custom applications and services can be deployed. Can be built and deployed more inexpensively, although services need to be supported and managed.

Physical infrastructure is abstracted to provide computing, storage, and networking as a service, avoiding the expense and need for dedicated systems.

Simple example:

- IaaS: barebones computer
- PaaS: computer + OS (incl. development environment)
- SaaS: complete solution including application(s)

IaaS, PaaS, SaaS

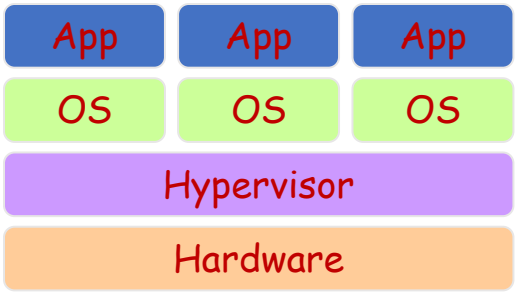
- IaaS: Amazon Web Services (AWS), Microsoft Azure, Google Compute Engine
- PaaS: Google App Engine, Heroku, OpenShift, AWS Elastic Beanstalk
- SaaS: Google Apps, Dropbox, Cisco Webex, Salesforce, Concur, GoToMeeting

Basic Cloud Characteristics

- “No-need-to-know”: interact with underlying infrastructure via API
- Flexibility and elasticity: scale systems up and down (allocate/release resources) based on needs
- Pay as much as used and needed (actual usage vs. service levels)
- Anytime anywhere access

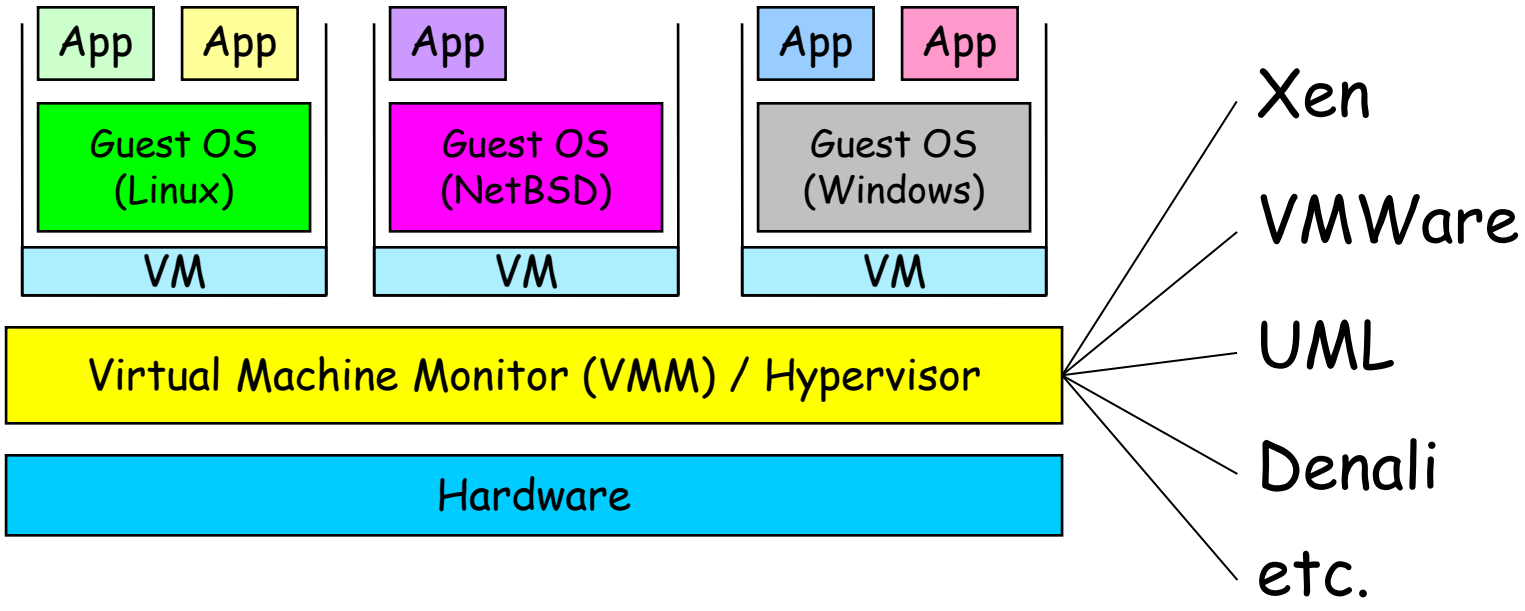
Virtualization

- Virtual workspaces:
  - An abstraction of an execution environment that can be made dynamically available to authorized clients by using well-defined protocols
  - Resource quota (e.g., CPU, memory share)
  - Software configuration (e.g., OS, provided services)
- Implemented on Virtual Machines (VMs):
  - Abstraction of a physical host machine
  - Hypervisor intercepts and emulates instructions from VMs, and allows management of VMs
  - VMWare, Xen, etc.



Virtualized Stack

Virtual Machines



Cloud Example: S3

- Amazon Simple Storage Service (S3)
- Unlimited storage
- Pay for what you use

	S3 Standard	S3 Standard – Infrequent Access	AWS Glacier
STORAGE			
First 50 TB/ month	$0.023 / GB	$0.0125 / GB	$0.004 / GB
Next 450 TB/ month	$0.022 / GB	$0.0125 / GB	$0.004 / GB
Over 500 TB/ month	$0.021 / GB	$0.0125 / GB	$0.004 / GB
REQUESTS			
PUT, COPY, POST, or LIST	$0.005 / 1,000 requests	$0.01 / 1,000 requests	
GET and all other requests	$0.004 / 10,000 requests	$0.01 / 10,000 requests	
Delete requests	Free	Free	Free, but with limits and potential surcharges
Lifecycle Transition Requests into S3 Standard IA		$0.01 / 1,000 requests	
Glacier archive and restore requests			$0.05 / 1,000 requests, see Glacier pricing for more details on retrieval fees

Cloud Example: EC2

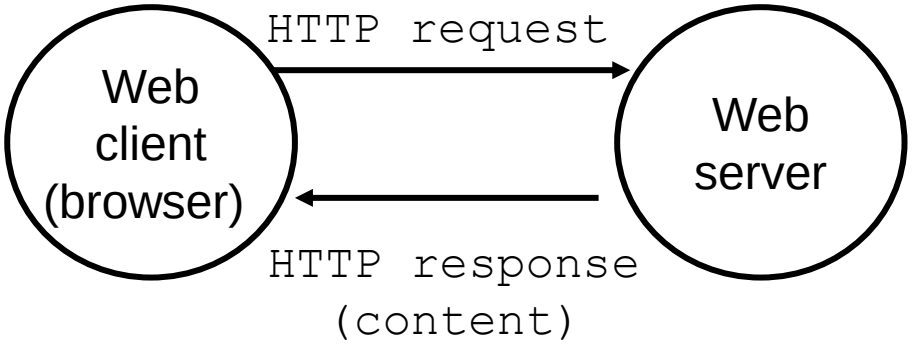
- Amazon Elastic Compute Cloud (EC2)
  - Virtual computing environments (“instances”)
  - Pre-configured templates for instances
  - Launch as many virtual servers as needed (“elastic”)
  - Xen and KVM hypervisor

Do You Use The Cloud?

Cloud for IoT

HyperText Transfer Protocol (HTTP)

- Clients and servers communicate using the **HyperText Transfer Protocol (HTTP)**
  - Client and server establish TCP connection
  - Client requests content
  - Server responds with requested content
  - Client and server close connection (usually)



Web Content

- Web servers return **content** to clients
  - a sequence of bytes with an associated MIME (Multipurpose Internet Mail Extensions) type
- Example MIME types

`text/html`	HTML document
`text/plain`	Unformatted text
`application/postscript`	Postcript document
`image/gif`	Binary image encoded in GIF format
`image/jpeg`	Binary image encoded in JPEG format

Static & Dynamic Content

- The content returned in HTTP responses can be either **static** or **dynamic**
  - Static content: content stored in files and retrieved in response to an HTTP request
    - Examples: HTML files, images, audio clips
  - Dynamic content: content produced on-the-fly in response to an HTTP request
    - Example: content produced by a program executed by the server on behalf of the client
- Bottom line: all web content is associated with a **file** that is managed by the server

URLs

- Each file managed by a server has a unique name called a URL (Universal Resource Locator)
- URLs for static content:
  - `http://www.cse.nd.edu:80/index.html`
  - `http://www.cse.nd.edu/index.html`
  - `http://www.cse.nd.edu`
    - Identifies a file called `index.html`, managed by a web server at `www.cse.nd.edu` that is listening on port 80
- URLs for dynamic content:
  - `http://www.cse.nd.edu:8000/cgi-bin/adder?15000&213`
    - Identifies an executable file called `adder`, managed by a web server at `www.cse.nd.edu` that is listening on port 8000, that should be called with two argument strings: `15000` and `213`

Anatomy of an HTTP Transaction

```
<unix> telnet www.aol.com 80          Client: open connection to server
Trying 205.188.146.23...              Telnet prints 3 lines to the terminal
Connected to aol.com.
Escape character is '^]'.
GET / HTTP/1.1                        Client: request line
host: www.aol.com                     Client: required HTTP/1.1 HOST header
                                      Client: empty line terminates headers .
HTTP/1.0 200 OK                       Server: response line
MIME-Version: 1.0                     Server: followed by five response headers
Date: Mon, 08 Jan 2001 04:59:42 GMT
Server: NaviServer/2.0 AOLserver/2.3.3
Content-Type: text/html               Server: expect HTML in the response body
Content-Length: 42092                 Server: expect 42,092 bytes in the resp body
                                      Server: empty line ("\r\n") terminates hdrs
<html>                                Server: first HTML line in response body
...                                   Server: 766 lines of HTML not shown.
</html>                               Server: last HTML line in response body
Connection closed by foreign host.    Server: closes connection
unix>                                 Client: closes connection and terminates
```

HTTP Requests

- HTTP request is a *request line*, followed by zero or more *request headers*

- Request line:
  - `<method> <uri> <version>`
  - `<version>` is HTTP version of request (`HTTP/1.0` or `HTTP/1.1`)
  - `<uri>` is typically URL for proxies, URL suffix for servers
  - `<method>` is either `GET`, `POST`, `OPTIONS`, `HEAD`, `PUT`, `DELETE`, or `TRACE`

HTTP Requests

- HTTP methods:
  - `GET`: Retrieve static or dynamic content
    - Arguments for dynamic content are in URI
    - Workhorse method (99% of requests)
  - `POST`: Retrieve dynamic content
    - Arguments for dynamic content are in the request body
  - `OPTIONS`: Get server or file attributes
  - `HEAD`: Like `GET` but no data in response body
  - `PUT`: Write a file to the server
  - `DELETE`: Delete a file on the server
  - `TRACE`: Echo request in response body
    - Useful for debugging

HTTP Responses

- HTTP response is a *response line* followed by zero or more *response headers*
- Response line:

  `<version> <status code> <status msg>`

  - <version> is HTTP version of the response
  - <status code> is numeric status
  - <status msg> is corresponding English text
    - 200 OK Request was handled without error
    - 403 Forbidden Server lacks permission to access file
    - 404 Not found Server couldn't find the file
- Response headers: `<header name>: <header data>`
  - Provide additional information about response
  - `Content-Type:` MIME type of content in response body
  - `Content-Length:` Length of content in response body

REST

- Representational State Transfer (REST) is a design pattern.
- A style of software architecture for distributed hypermedia systems such as the World Wide Web
- A collection of network architecture principles which outline how **resources** are defined and addressed.
- It is a certain approach to creating Web Services.
- To understand the REST design pattern, let's look at an example (learn by example).

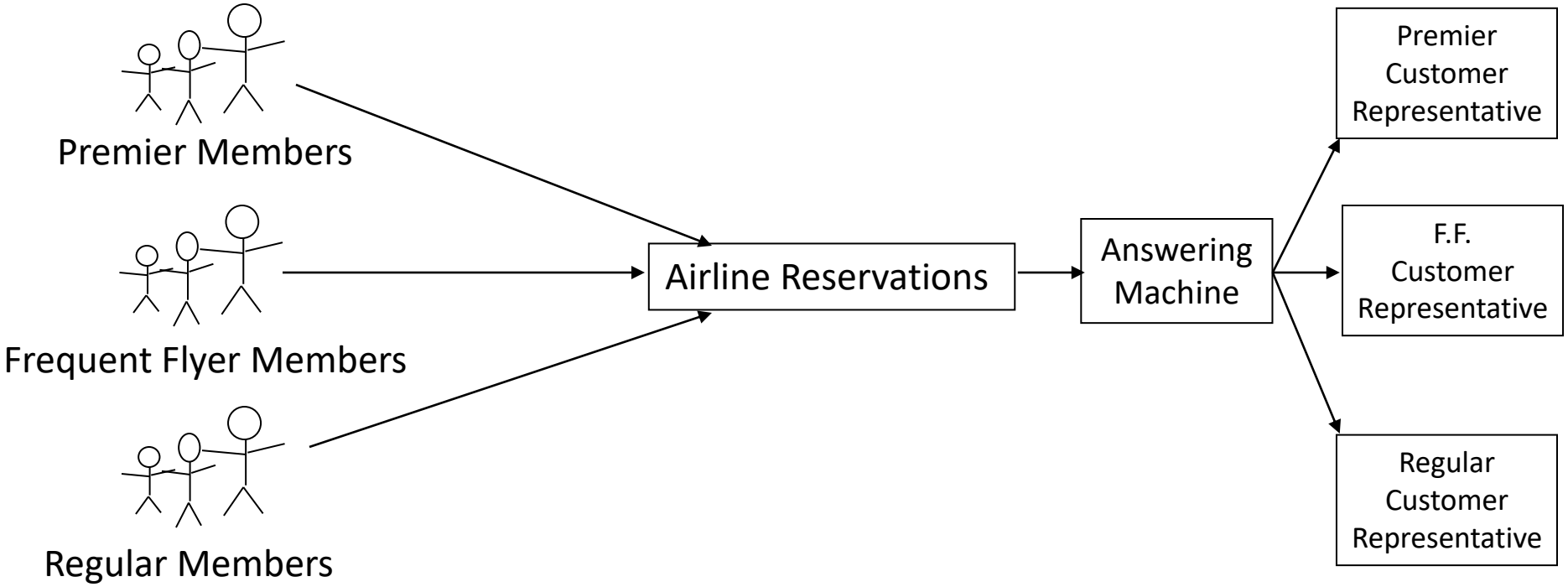
Example:
Airline Reservation Service

- Suppose that an airline wants to create a telephone reservation system for customers to call in and make flight reservations.
- The airline wants to ensure that its premier members get immediate service, its frequent flyer members get expedited service, and all others get regular service.
- There are two main approaches to implementing the reservation service...

Approach 1
"Press 1 for Premier, Press 2 for..."

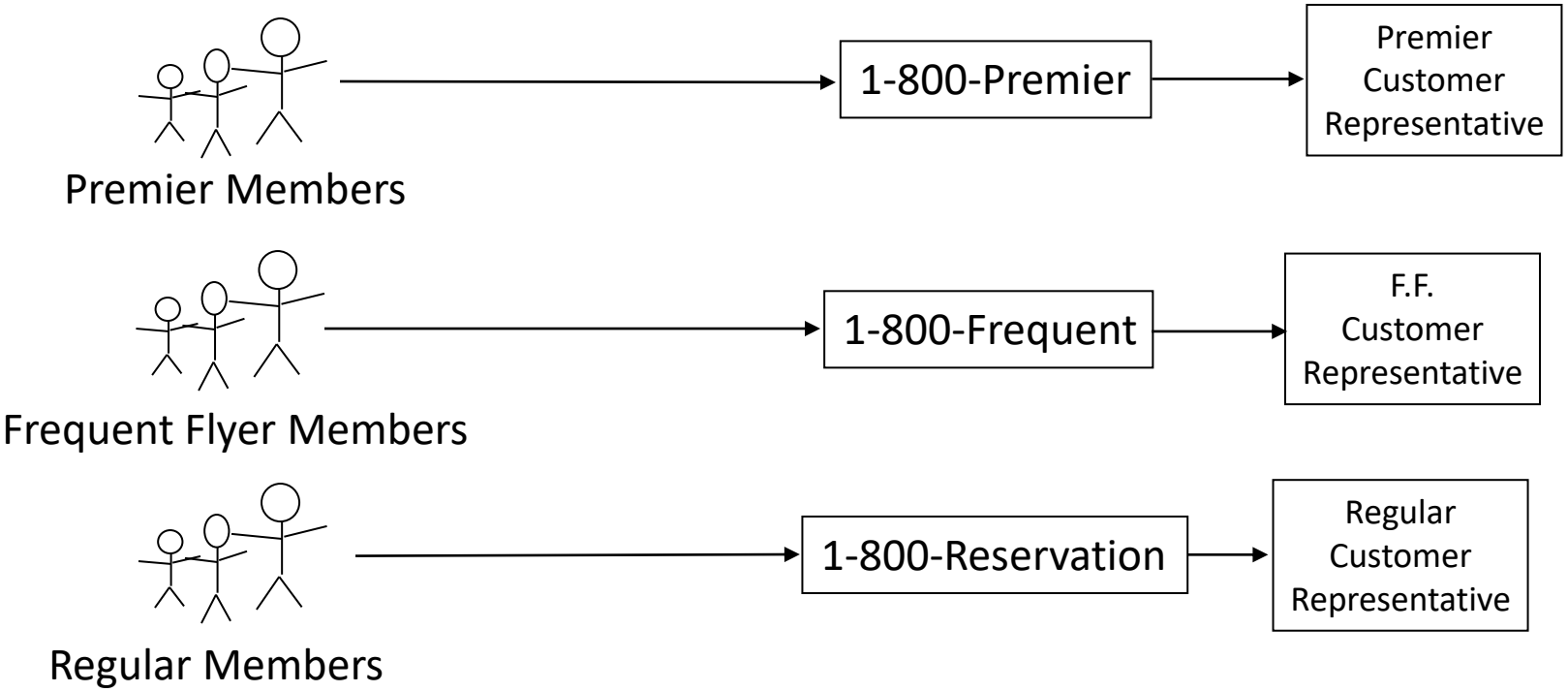
The airline provides a single telephone number.

Upon entry into the system a customer encounters an automated message, "Press 1 if you are a premier member, press 2 if you are a frequent flyer, press 3 for all others."



Approach 2
Telephone Numbers are Cheap! Use Them!

The airline provides several telephone numbers - one number for premier members, a different number for frequent flyers, and still another for regular customers.



Discussion

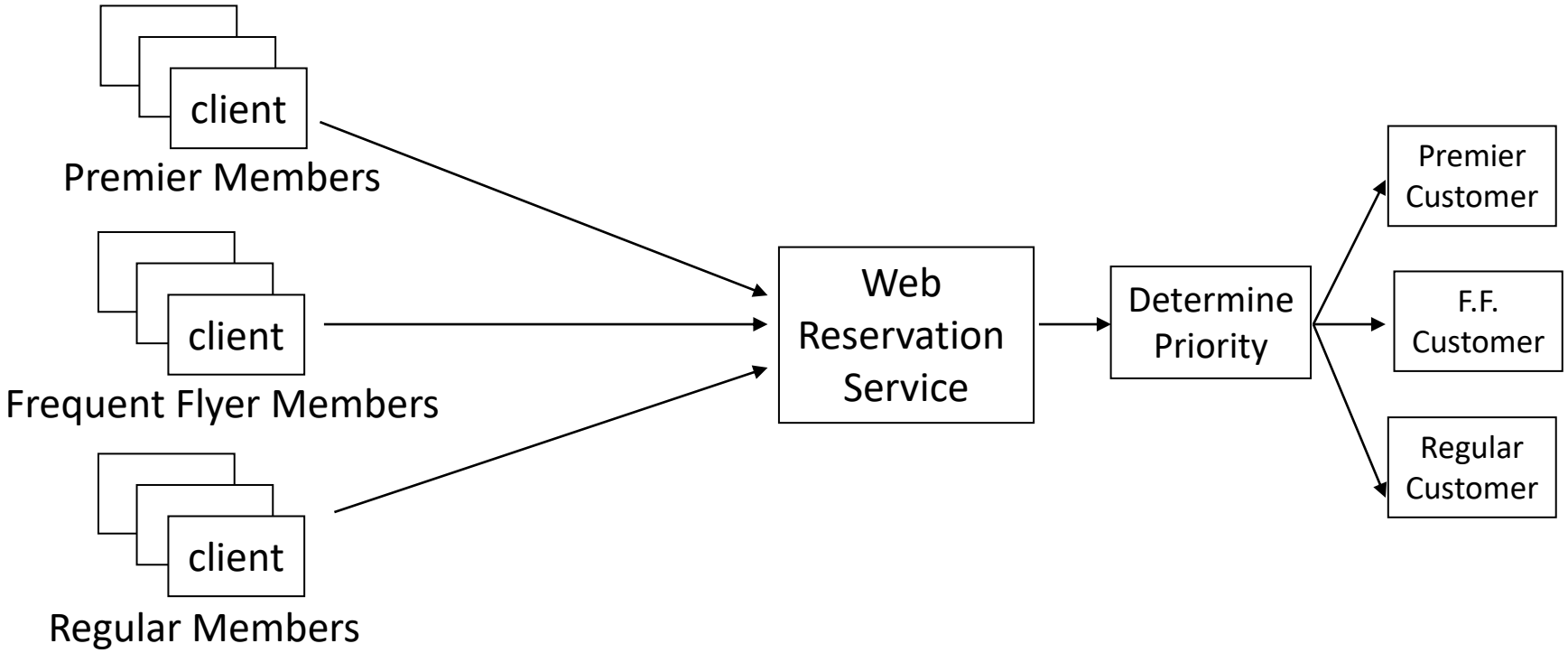
- In Approach 1 the answering machine introduces an extra delay, which is particularly annoying to premier members. (Doesn't everyone hate those answering systems)
- With Approach 2 there is no intermediate step. Premier members get instant pickup from a customer service representative. Others may have to wait for an operator.

Web-Based Reservation Service

- Suppose now the airline (kings-air.com) wants to provide a Web reservation service for customers to make flight reservations through the Web.
- Just as with the telephone service, the airline wants to ensure that its premier members get immediate service, its frequent flyer members get expedited service, all others get regular service.
- There are two main approaches to implementing the Web reservation service. The approaches are analogous to the telephone service...

Approach 1
One-Stop Shopping

The airline provides a single URL. The Web service is responsible for examining incoming client requests to determine their priority and process them accordingly.

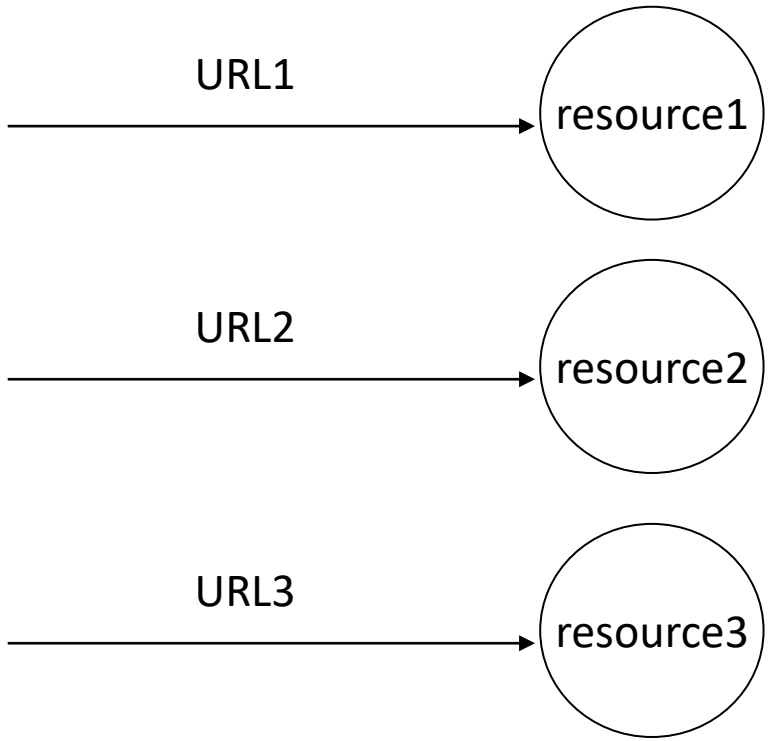


Approach 1 Disadvantages

- There is currently no industry accepted practice (rules) for expressing priorities, so rules would need to be made. The clients must learn the rule, and the Web service application must be written to understand the rule.
- This approach is based upon the incorrect assumption that a URL is "expensive" and that their use must be rationed.
- The Web service is a central point of failure. It is a bottleneck. Load balancing is a challenge.
- It violates Tim Berners-Lee Web Design, Axiom 0 (see next slide).

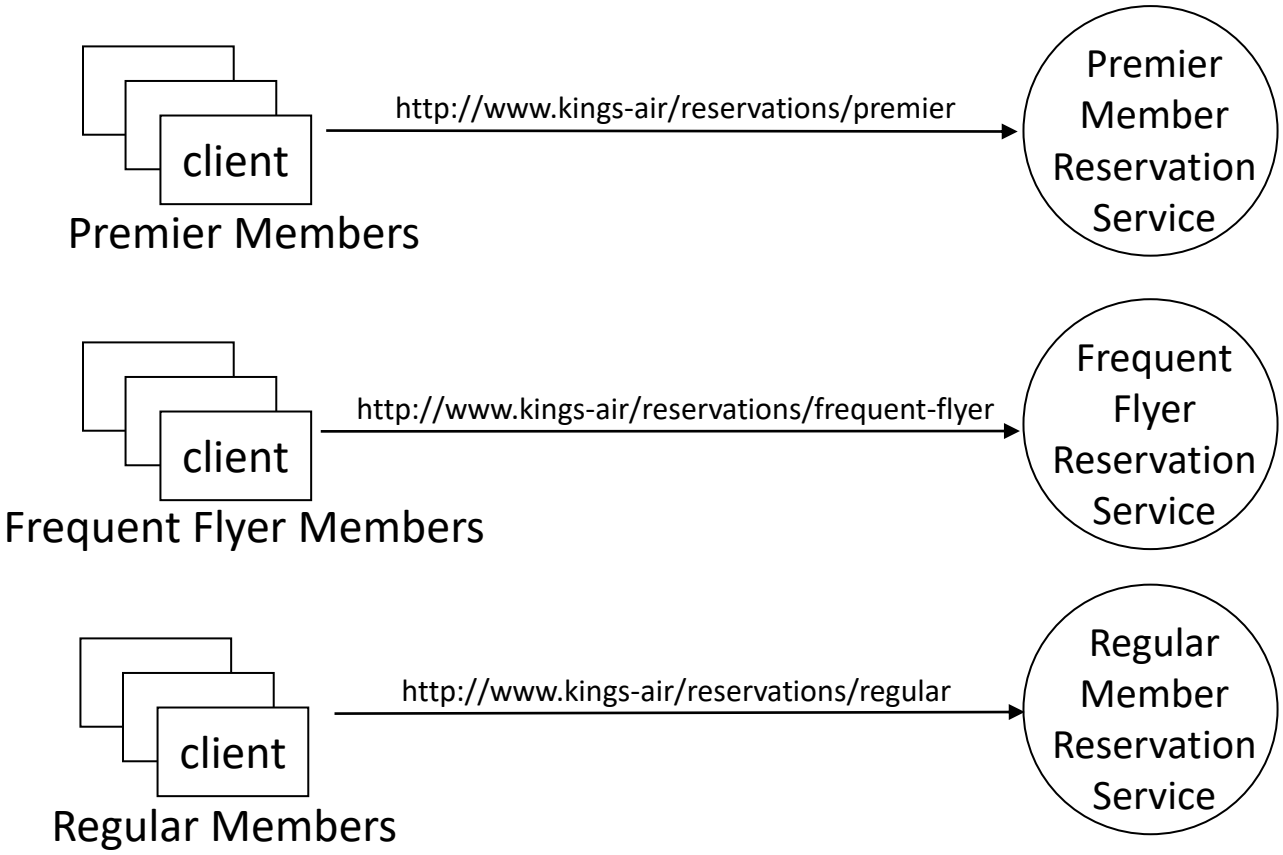
Web Design, Axiom 0
(Tim Berners-Lee, director of W3C)

- Axiom 0: all resources on the Web must be uniquely identified with a URI.



Approach 2:
URLs are Cheap! Use Them!

The airline provides several URLs - one URL for premier members, a different URL for frequent flyers, and still another for regular customers.



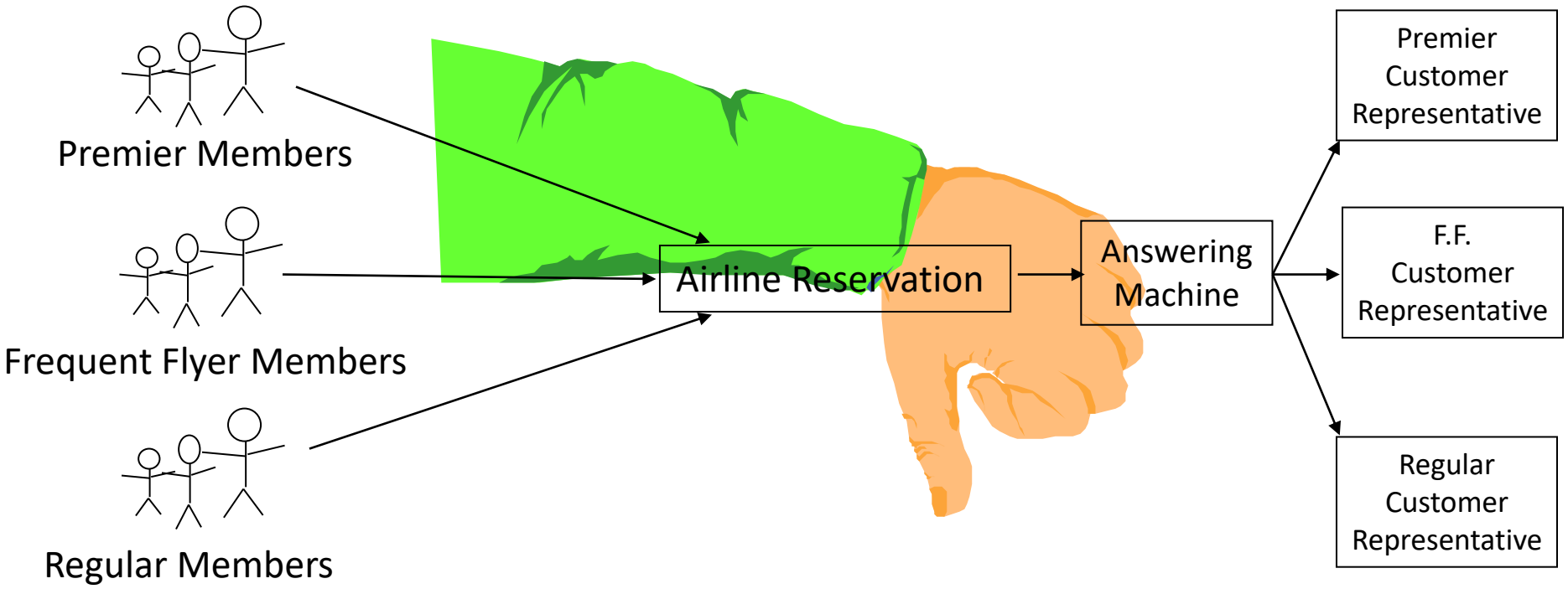
Approach 2 Advantages

- The different URLs are discoverable by search engines and UDDI registries.
- It's easy to understand what each service does simply by examining the URL, *i.e.,* it exploits the Principle of Least Surprise.
- There is no need to introduce rules. Priorities are elevated to the level of a URL. "What you see is what you get."
- It's easy to implement high priority - simply assign a fast machine at the premier member URL.
- There is no bottleneck. There is no central point of failure.
- Consistent with Axiom 0.

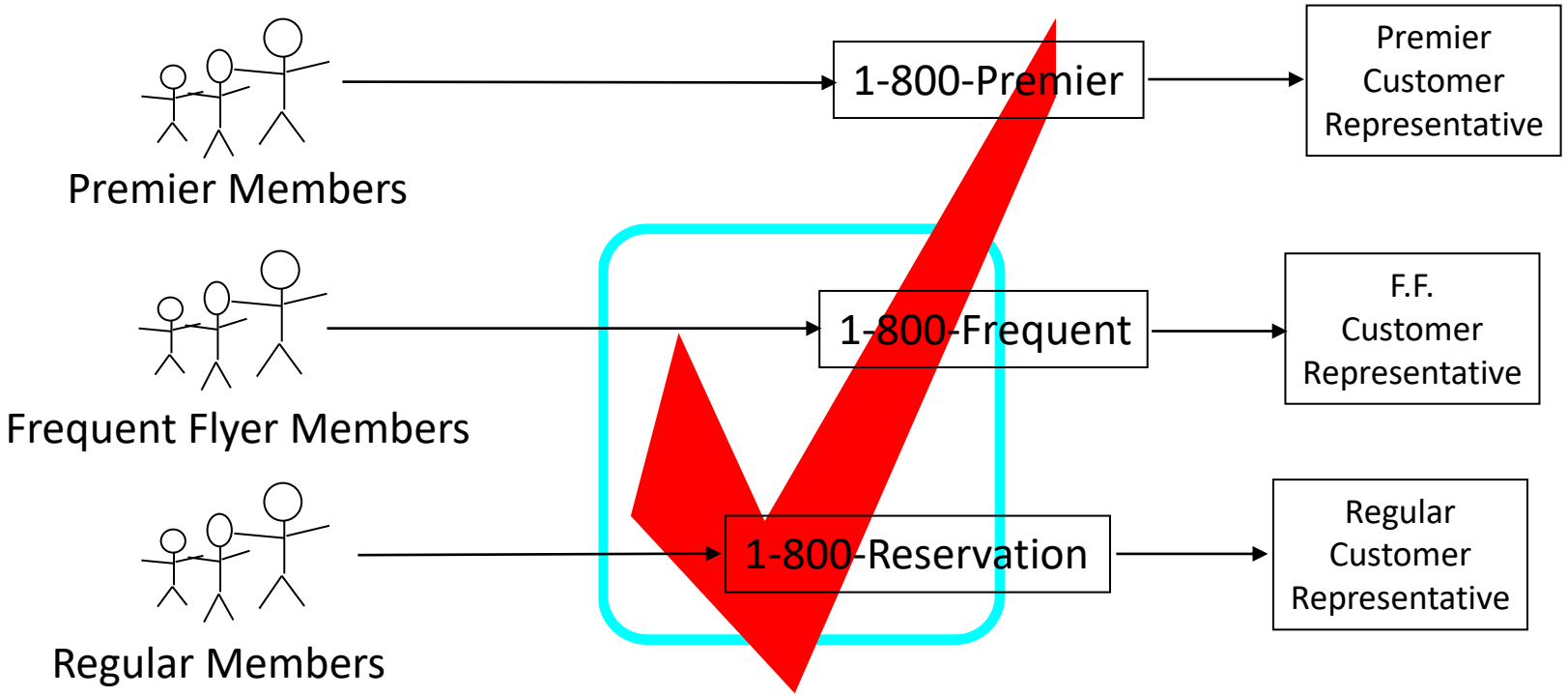
Recap

- We have looked at a reservation service.
- We have seen a telephone-based version and a Web-based version of the reservation service.
- With each version we have seen two main approaches to implementing the service.
- Which approach is the REST design pattern and which isn't? See the following slides.

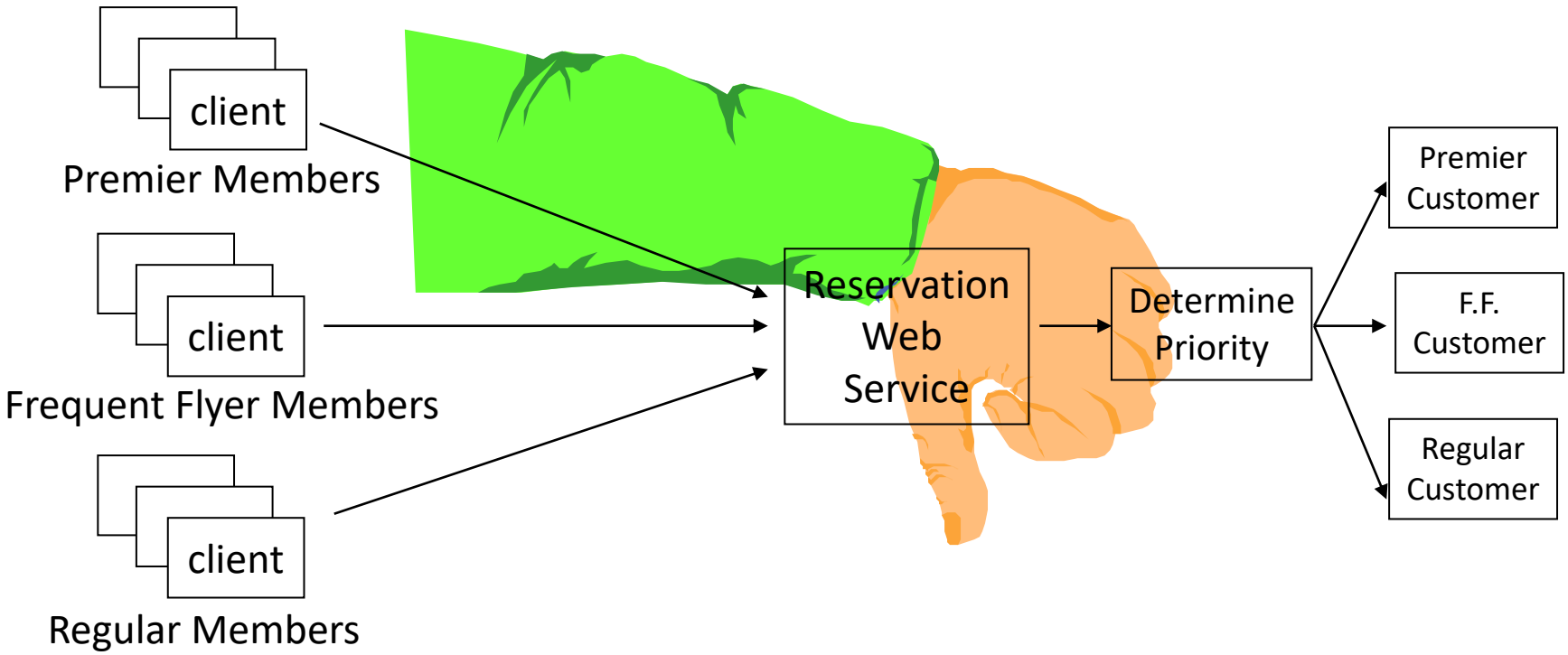
This Ain't the REST Design Pattern



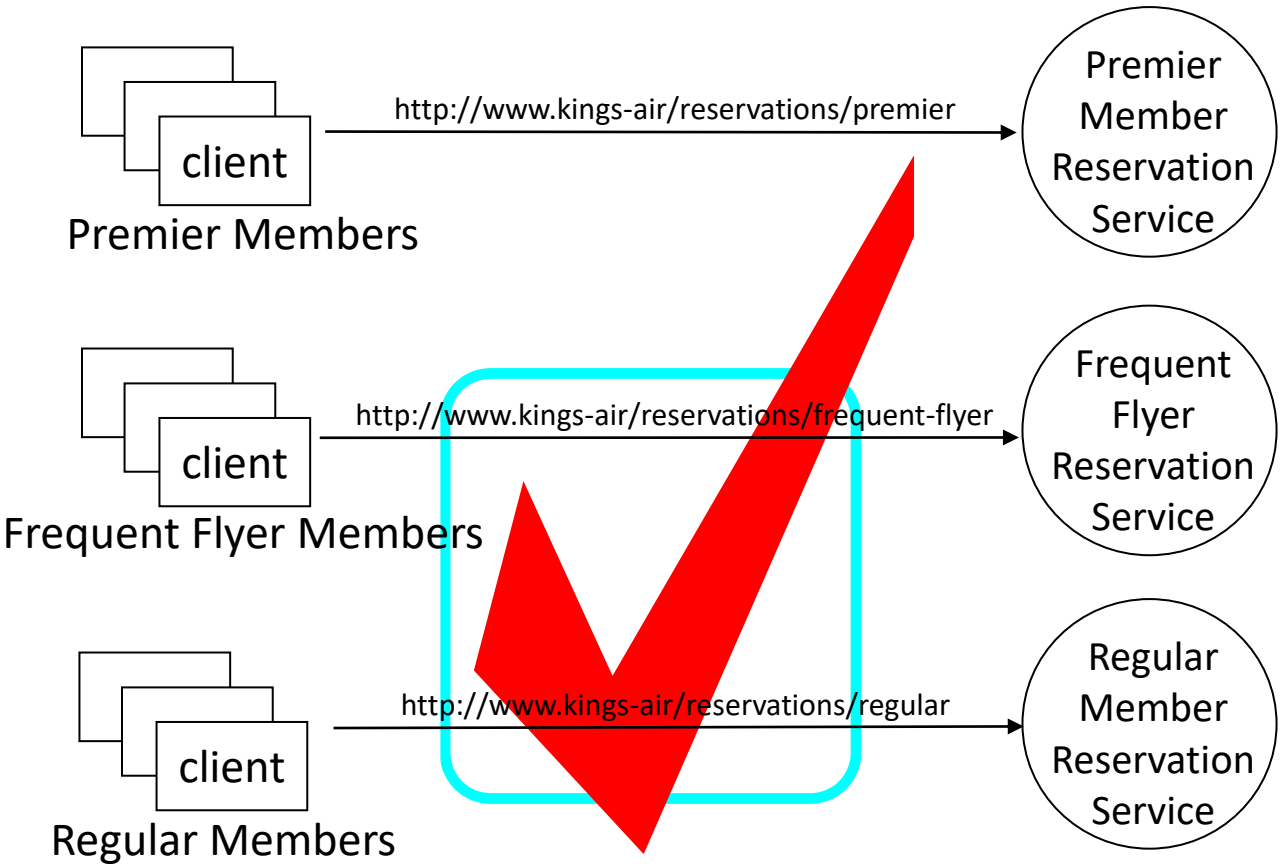
This is the REST Design Pattern



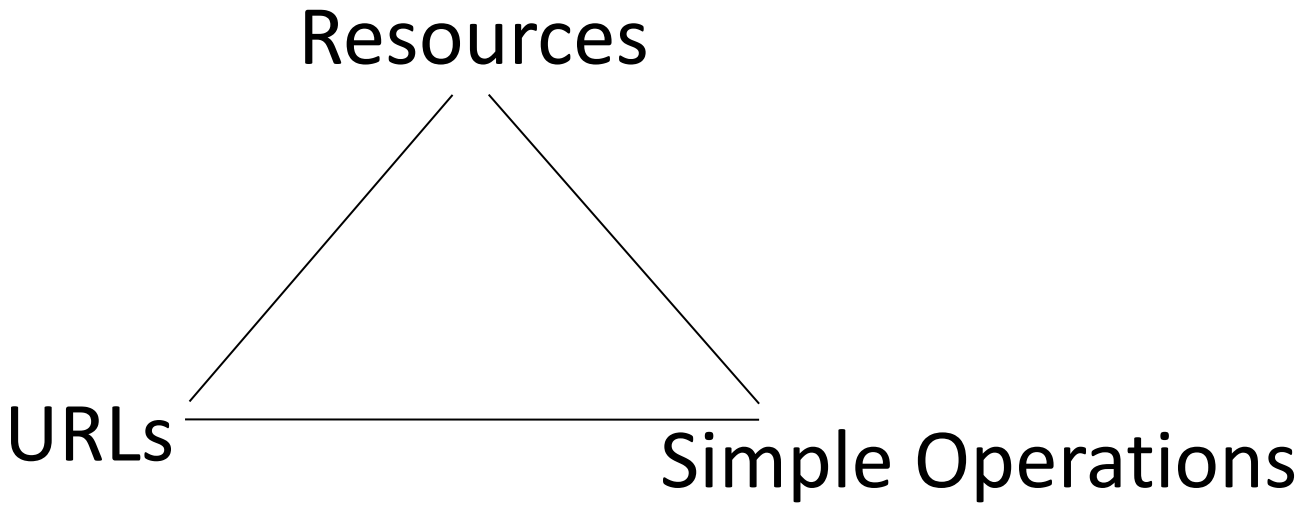
This ain't the REST Design Pattern



This is the REST Design Pattern



The Three Fundamental Aspects of the REST Design Pattern



In this tutorial we discussed how Resources and URLs are fundamental to REST. In a follow up tutorial we will discuss how Simple Operations are also fundamental to REST.

REST & HTTP

- The motivation for REST was to capture the characteristics of the Web which made the Web successful
  - URI Addressable resources
  - HTTP Protocol
  - Make a Request – Receive Response – Display Response
- Exploits the use of the HTTP protocol beyond HTTP POST and HTTP GET
  - HTTP PUT, HTTP DELETE

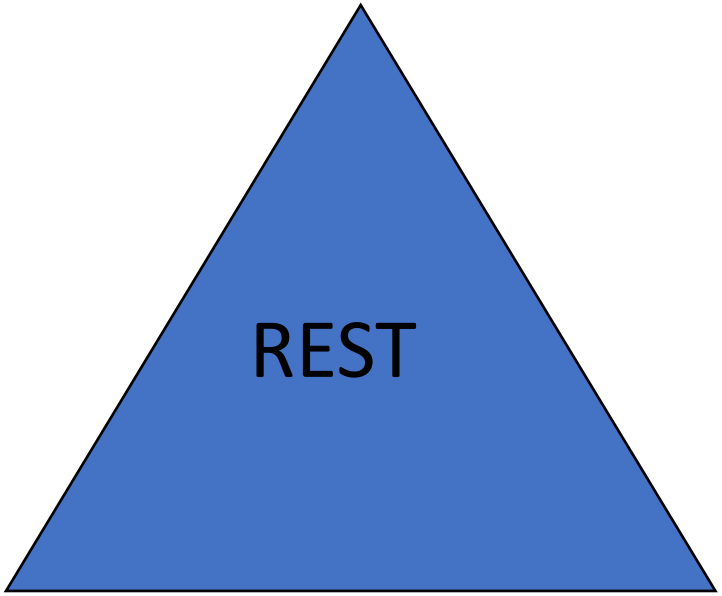
REST

- REST is not a standard
  - is an architectural style

- But it uses several standards:
  - HTTP
  - URL
  - XML/HTML/GIF/JPEG/etc (Resource Representations)
  - text/xml, text/html, image/gif, image/jpeg, etc (Resource Types, MIME Types)

REST Main Concepts

Nouns (Resources)
unconstrained
i.e., http://example.com/employees/12345



Verbs
constrained
i.e., GET

Representations
constrained
i.e., XML

Resources

- The key abstraction of information in REST is a resource
- A resource is a conceptual mapping to a set of entities
  - Any information that can be named can be a resource: a document or image, a temporal service (e.g., “today's weather in Berlin”), a collection of other resources, a non-virtual object (e.g., a person), etc.
- Represented with a global identifier (URI in HTTP)
  - http://www.boeing.com/aircraft/747

Naming Resources

- REST uses URI to identify resources
  - http://localhost/books/
  - http://localhost/books/ISBN-0011
  - http://localhost/books/ISBN-0011/authors
  - http://localhost/classes
  - http://localhost/classes/cs2650
  - http://localhost/classes/cs2650/students
- As you traverse the path from more generic to more specific, you are navigating the data

Verbs

- Represent the actions to be performed on resources

- HTTP GET
- HTTP POST
- HTTP PUT
- HTTP DELETE

HTTP GET

- How clients ask for the information they seek
- Issuing a GET request transfers the data from the server to the client in some representation
- GET http://localhost/books
  - Retrieve all books
- GET http://localhost/books/ISBN-0011021
  - Retrieve book identified with ISBN-0011021
- GET http://localhost/books/ISBN-0011021/authors
  - Retrieve authors for book identified with ISBN-0011021

HTTP PUT & POST

- HTTP POST creates a resource
- HTTP PUT updates a resource

- POST http://localhost/books/
  - Content: {title, authors[], …}
  - Creates a new book with given properties

- PUT http://localhost/books/isbn-111
  - Content: {isbn, title, authors[], …}
  - Updates book identified by isbn-111 with submitted properties

Representations

- How data is represented or returned to the client for presentation.
- Two main formats:
  - JavaScript Object Notation (JSON)
  - XML
- It is common to have multiple representations of the same data

Representations

- XML

```
<COURSE>
    <ID>CS2650</ID>
    <NAME>Distributed Multimedia Software</NAME>
</COURSE>
```

- JSON

```
{course
    {id: CS2650}
    {name: Distributed Multimedia Software}
}
```

Thank You

Contact me:

gauravsingal789@gmail.com

Gaurav.singal@nsut.ac.in

www.gauravsingal.in

LinkedIn: https://www.linkedin.com/in/gauravsingal789/

Twitter: https://twitter.com/gaurav_singal