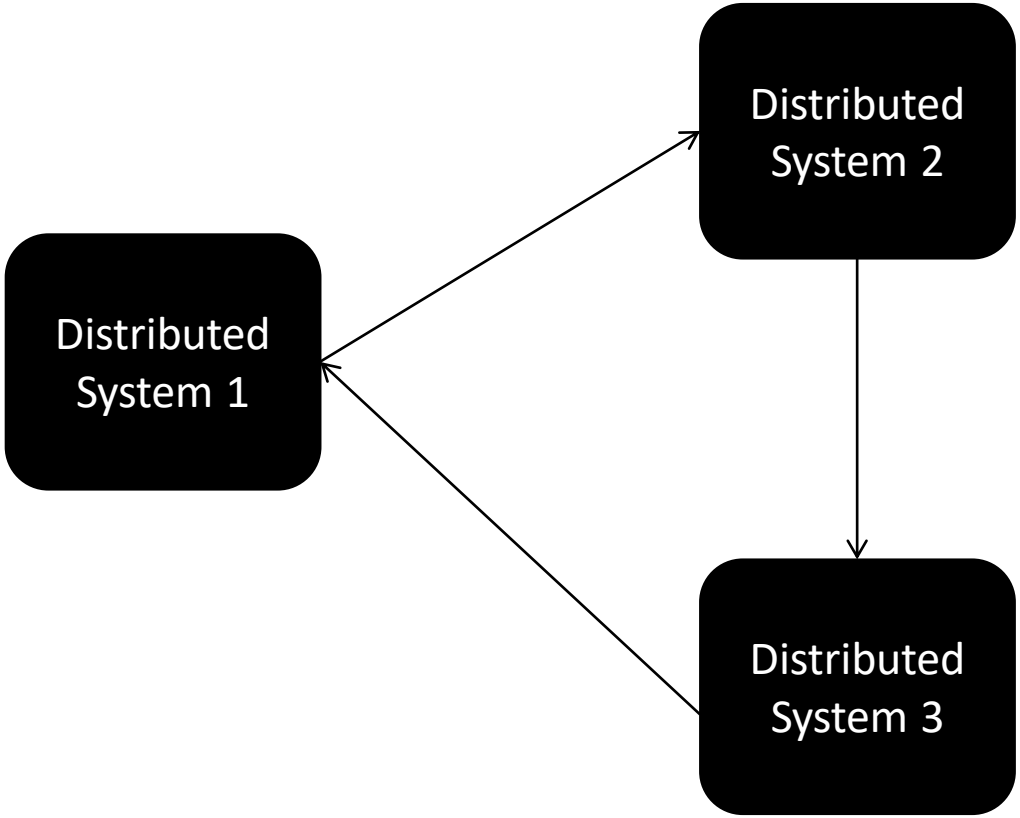
Web Application Development and Web Services

Venkatesh Vinayakarao
venkateshv@cmi.ac.in
http://vvtesh.co.in

Chennai Mathematical Institute

If You Think Math is Hard Try Web Design. **– PixxelzNet.**

How to Achieve Interoperability?

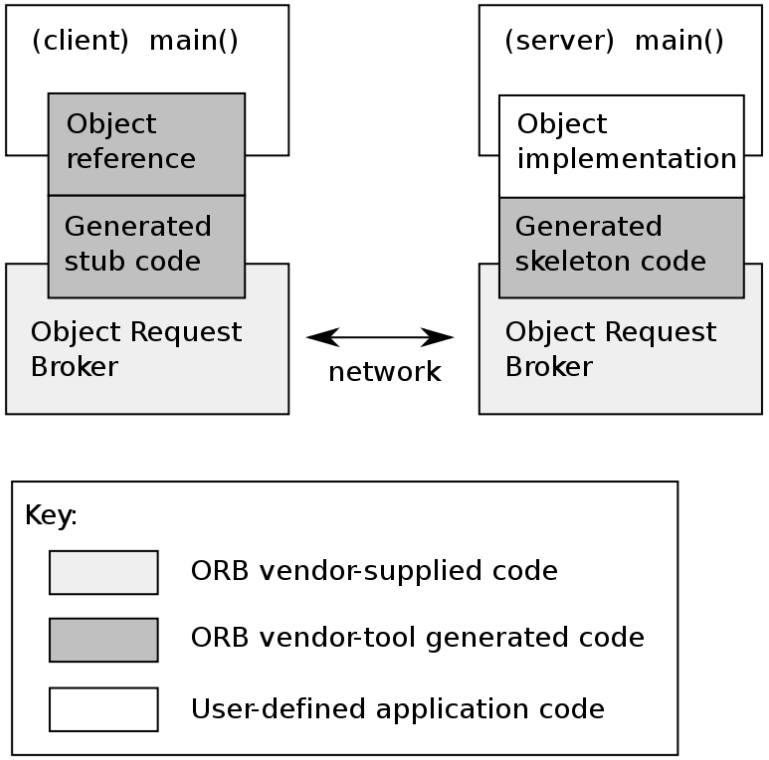


Interoperability Solutions

- Many Solutions
  - File Transfer
  - Shared DB
  - Remote Procedure Calls
  - Message Passing
- Middleware platforms aimed at making it more structured and easier
  - CORBA, DCOM, RMI, …
  - Web Services

Interoperability Solutions

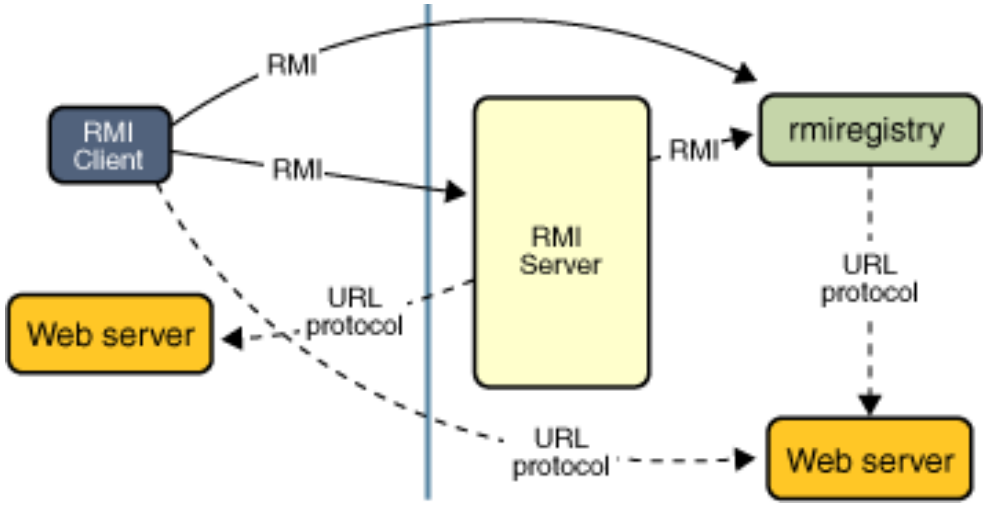
- CORBA (1991)
  - Standards-based, vendor-neutral, and language-agnostic.
  - Communicate by message passing over network
  - Read Corba: Gone But (Hopefully) Not Forgotten, Queue Vol 5, No. 4.



https://www.omg.org/spec/CORBA/
https://en.wikipedia.org/wiki/Common_Object_Request_Broker_Architecture
https://docs.oracle.com/javase/8/docs/technotes/guides/idl/jidlExample.html

More Interoperability Solutions

- Distributed Component Object Model (DCOM) (Microsoft)
- RMI (Sun Microsystems)
  - Invoke method on a remote object.



https://docs.oracle.com/javase/tutorial/rmi/overview.html

Web Services

- A "service" is a software component provided through an (often, network-accessible) endpoint.
- Service consumer and provider use messages to exchange invocation request and response information in the form of self-containing documents.

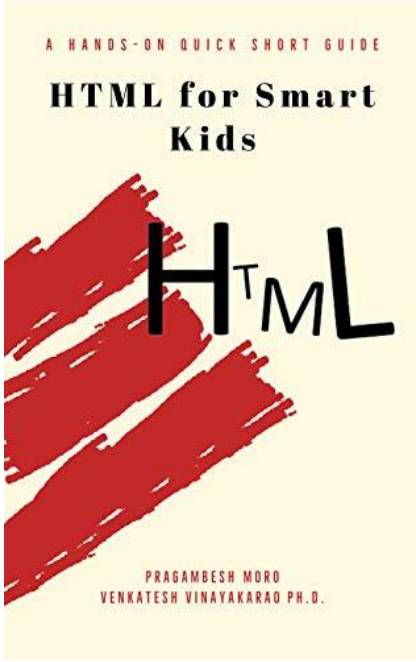
What do you understand by "**Web**"?

Early Static Web

- Developed in 1990 at CERN
- NCSA Mosaic 1.0 was the first browser, released by the National Center for Supercomputer Applications (NCSA).

Creating Web Pages

- Write HTML code.
- Move it to a *Web Server*.
- Access it over the web.



The Dynamic Web

- Httpd 1.0 web server allowed Common Gateway Interface (CGI).
- CGI allows a browser client to request data from a program running on a Web server.

CGI Script

```
#! /usr/local/bin/perl
# Display the form data set sent in a GET or POST request.

print "Content-type: text/html\n\n";
print "<html><head><title>Form Data</title></head> \n";
print "<body bgcolor=\"#FFFFFF\"\n>"

if ($ENV{'REQUEST_METHOD'} eq 'POST') {
    read (STDIN, $buffer, $ENV{'CONTENT_LENGTH'});
    @pairs = split(/&/, $buffer);
} elsif ($ENV{'REQUEST_METHOD'} eq 'GET') {
    @pairs = split(/&/, $ENV{'QUERY_STRING'});
} else {
    print "<p>$ENV{'REQUEST_METHOD'} message received</p>";
}
foreach $pair (@pairs) {
    ($name, $value) = split(/=/, $pair);
    $value =~ tr/+/ /;
    $value =~ s/%([a-fA-F0-9][a-fA-F0-9])/pack("C", hex($1))/eg;
    $name =~ tr/+/ /;
    $name =~ s/%([a-fA-F0-9][a-fA-F0-9])/pack("C", hex($1))/eg;
    print "<p>Field $name has the value $value</p> \n";
    $FORM{$name} = $value;
} print "</body></html> \n";
```

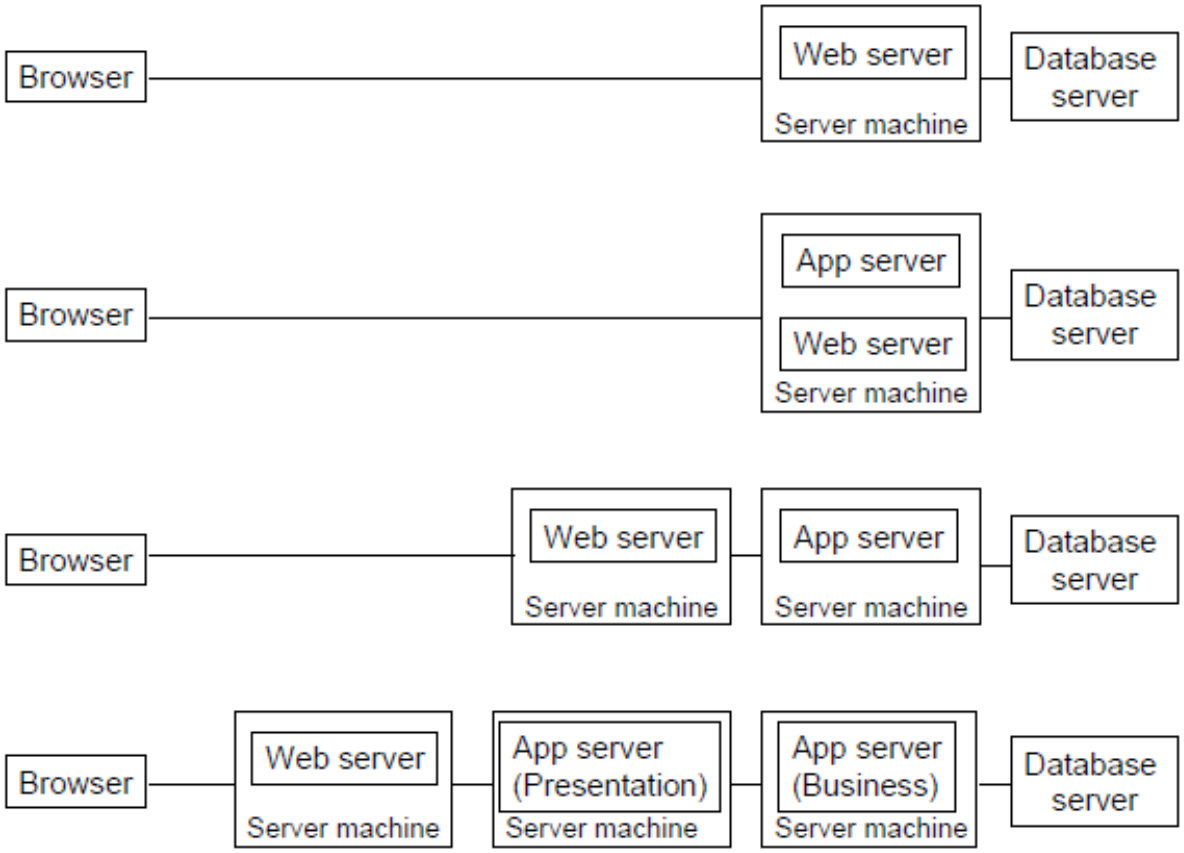
Server-Side (javascript) Scripting

```
<html>
    <head>  <title>Server-Side JavaScript Example Author Listing</title>  </head>
    <body>
        <h1>Author List</h1>
        <server>
            if (!database.connected()){
                database.connect("ODBC","bookdb","admin","","");
            }
            if (database.connected()) {
                qs = "SELECT au_id, au_fname, au_lname FROM authors";
                results =  database.cursor(qs);
                write("<table border=2 cellpadding=2 cellspacing=2>" +
                      "<tr><th>ID</th><th>First Name</th><th>Last Name </th></tr> \n");
                while(results.next()) {
                      write("<tr><td>" + results.au_id + "</td> + "<td>" +
                            results.au_fname + "</td>" +
                            "<td>" + results.au_lname + "</td></tr> \n");
                }
                results.close(); write("</table> \n");
            }
            else {
                write("<p>Database connection failed");
            }
        </server>
    </body>
</html>
```

ASP Page

```
<%
    Dim conn, rs
    Set conn = Server.CreateObject("ADODB.Connection")
    Set rs = Server.CreateObject("ADODB.Recordset")
    conn.Open "bookdb", "sa", "password"
    Set rs = conn.Execute("select au_id, au_fname, au_lname from authors")
%>
<html>
    <head>  <title>ASP Example Author Listing</title></head>
    <body>
        <h1>Author List</h1>
        <table>
            <tr><th>ID</th><th>First Name</th><th>Last Name</th></tr>
            < % Do Until rs.EOF %>
                <tr><td><%=rs("au_id") %></td>
                <td><%=rs("au_fname") %></td>
                <td><%=rs("au_lname") %></td></tr>
                < % rs.movenext
                Loop
            %>
        < /table>
    < /body>
< /html>
```

Evolution of Web and App Servers



Software as a Service (SaaS)

https://od-api.oxforddictionaries.com/api/v2/entries/en-us/ubiquitous

```
7   # TODO: replace with your own app_id and app_key
8   app_id = '<my app_id>'
9   app_key = '<my app_key>'
10  language = 'en'
11  word_id = 'Ace'
12
13  url = 'https://od-api.oxforddictionaries.com:443/api/v2/entries/'  + language + '/'  + word_id.lower()
14  #url Normalized frequency
15  urlFR = 'https://od-api.oxforddictionaries.com:443/api/v2/stats/frequency/word/'  + language + '/?corpus=nmc&lemma='
+ word_id.lower()
16
17  r = requests.get(url, headers = {'app_id' : app_id, 'app_key' : app_key})
18
19  print("code {}\n".format(r.status_code))
20  print("text \n" + r.text)
21  print("json \n" + json.dumps(r.json()))
```

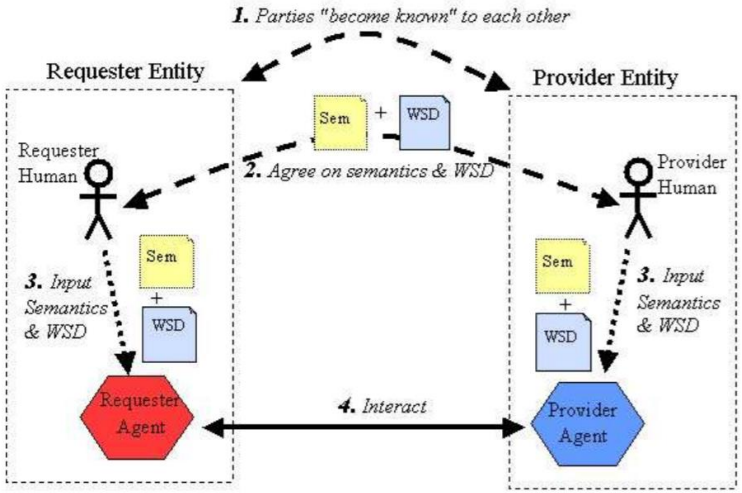
API Service from Oxford Dictionary
https://developer.oxforddictionaries.com/

{
"definitions": [
"present, appearing, or found everywhere"]
}

Response in JSON format

Web Services

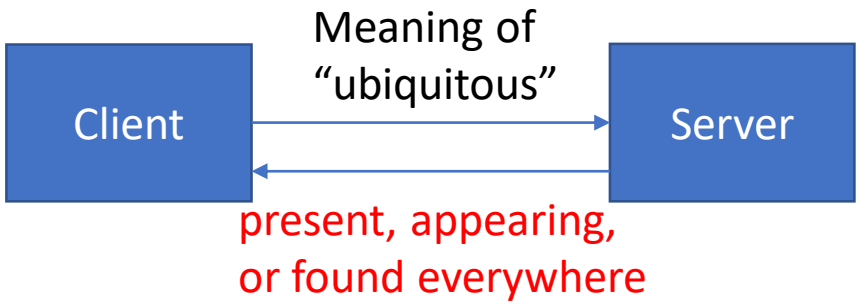
- A Web service is a software system designed to support interoperable machine-to-machine interaction over a network.

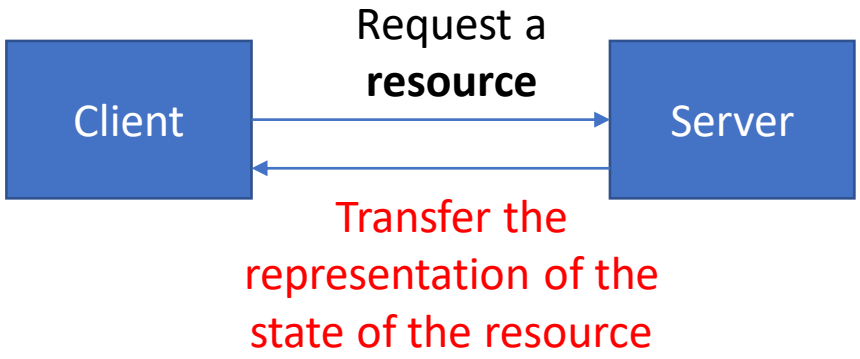


https://www.w3.org/TR/ws-arch/wsa.pdf

REST API

- REST = Representational State Transfer
  - Proposed by Roy Fielding in 2000.





Resource

- Any information that can be named is a resource
  - Document, image, or any other object.
- Description of the state of the resource at any timestamp is known as resource representation
  - Representation consists of data describing the resource.
- Resource methods are used to transfer the resource state representations.
  - Need not be always HTTP (GET/POST/…).

RESTful Web Services API

- Let us retrieve an existing configuration:
  - http://example.com/network-app/configurations/678678
  - HTTP GET /configurations/{id}

- Similarly, we can POST, PUT, and DELETE.
  - HTTP POST /devices
  - HTTP POST /configurations
  - HTTP PUT /devices/{id}/configurations
  - HTTP DELETE /devices/{id}/configurations/{id}

https://restfulapi.net/rest-api-design-tutorial-with-example/

HTTP

- HTTP Methods

HTTP Method	Purpose
POST	Create
GET	Retrieve
PUT	Update
DELETE	Delete

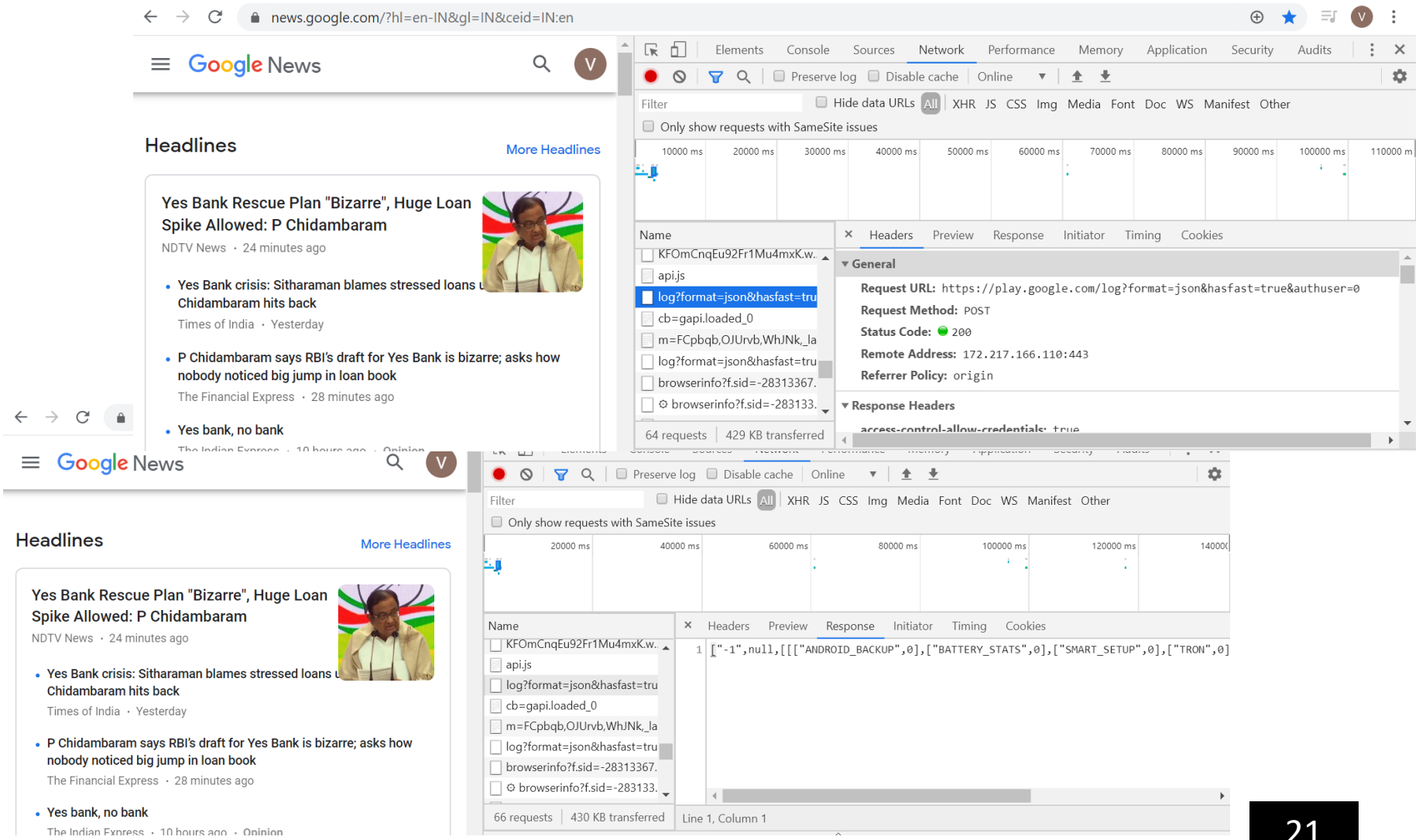
- *"An idempotent HTTP method is an HTTP method that can be called many times without different outcomes."*
  - POST is NOT idempotent.
  - GET, PUT, DELETE are idempotent.

HTTP Response Codes

- 2xx
  - Success
  - Example: 200 = OK, 201 = Created, 202 = Accepted (if it is a long-running task)
- 4xx
  - Client Error
  - Example: 400 = Bad Request, 404 = Not Found.
- 5xx
  - Server Error
  - Example: 500 = Internal Server Error

https://restfulapi.net/http-status-codes/

REST in Real World



Designing REST API

- Identify the object model
- Create Model URIs
- Determine Representations
- Assign HTTP Methods

```
<device id="12345">
    <link rel="self" href="/devices/12345"/>
    <deviceFamily>apple-es</deviceFamily>
    <OSVersion>10.3R2.11</OSVersion>
    <platform>SRX100B</platform>
    <serialNumber>32423457</serialNumber>
    <connectionStatus>up</connectionStatus>
    <ipAddr>192.168.21.9</ipAddr>
    <name>apple-srx_200</name>
    <status>active</status>
</device>
```

Web Services for a Banking Application

- Designing the REST API
  - Object Model
    - Customer, Account
  - Create Model URIs
    - /customers/{customerId}
    - /customers/{customerId}/accounts
    - /customers/{customerId}/accounts/{accountId}
  - Determine Representations
    - Represent all Account information as an XML/JSON
    - Represent all Customer information as XML/JSON
  - Assign HTTP Methods
    - Open Account = Create an Account Resource ➔ HTTP POST
    - Close Account = Delete the Account ➔ HTTP DELETE

Implementing RESTful web services

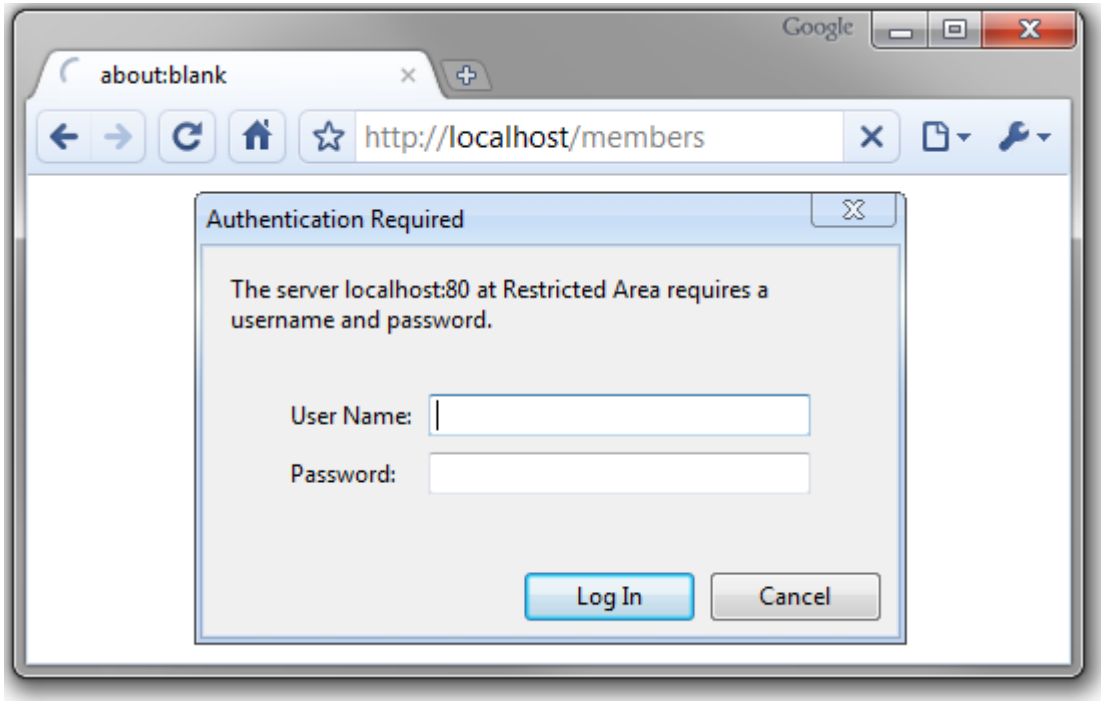
- Java API for RESTful web services (JAX-RS) [JSR 311] is specification.
- Jersey is a popular JAX-RS implementation.
- JAX-RS Annotations helps in building web services

```
@Path("/configurations")
public class ConfigurationResource
{
    @Path("/{id}")
    @GET
    public Response getConfigurationById(@PathParam("id") Integer id){
        ...
    }
}
```

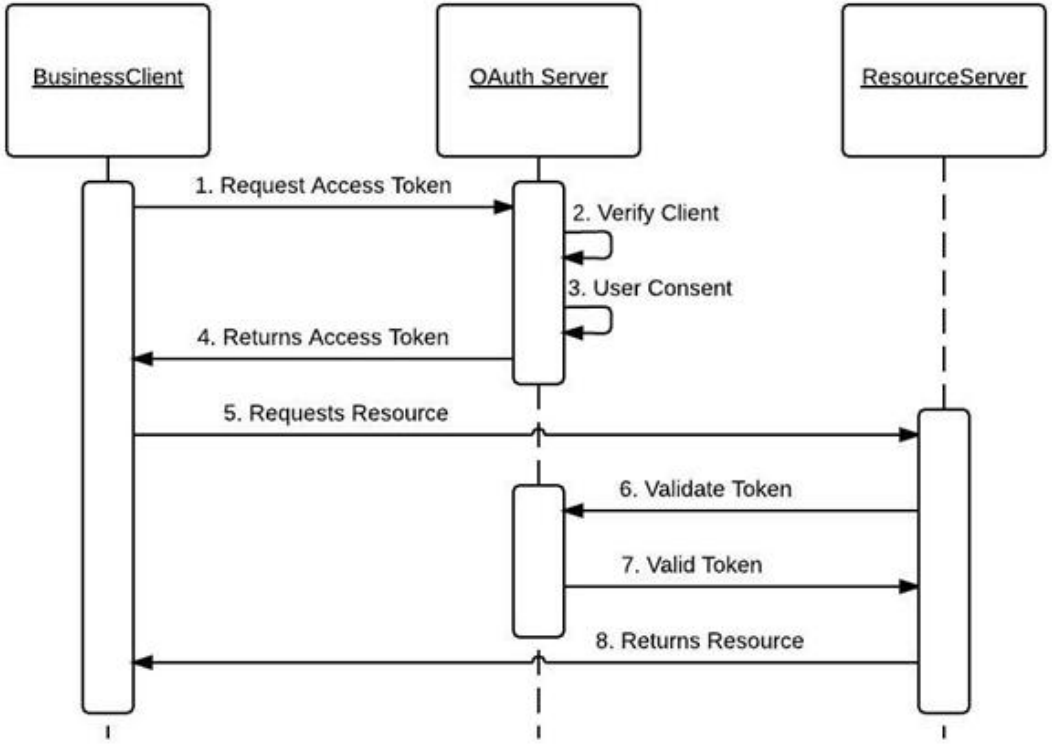
Authentication

- Basic HTTP Authentication
  - User enters the credentials
- Query String Authentication
  - URL has the credentials
- API Keys
  - Sever generated keys are used to identify the user.
- Token-based Authentication
  - oAuth method
  - Most secure form of authentication out of these four.

Basic HTTP Authentication



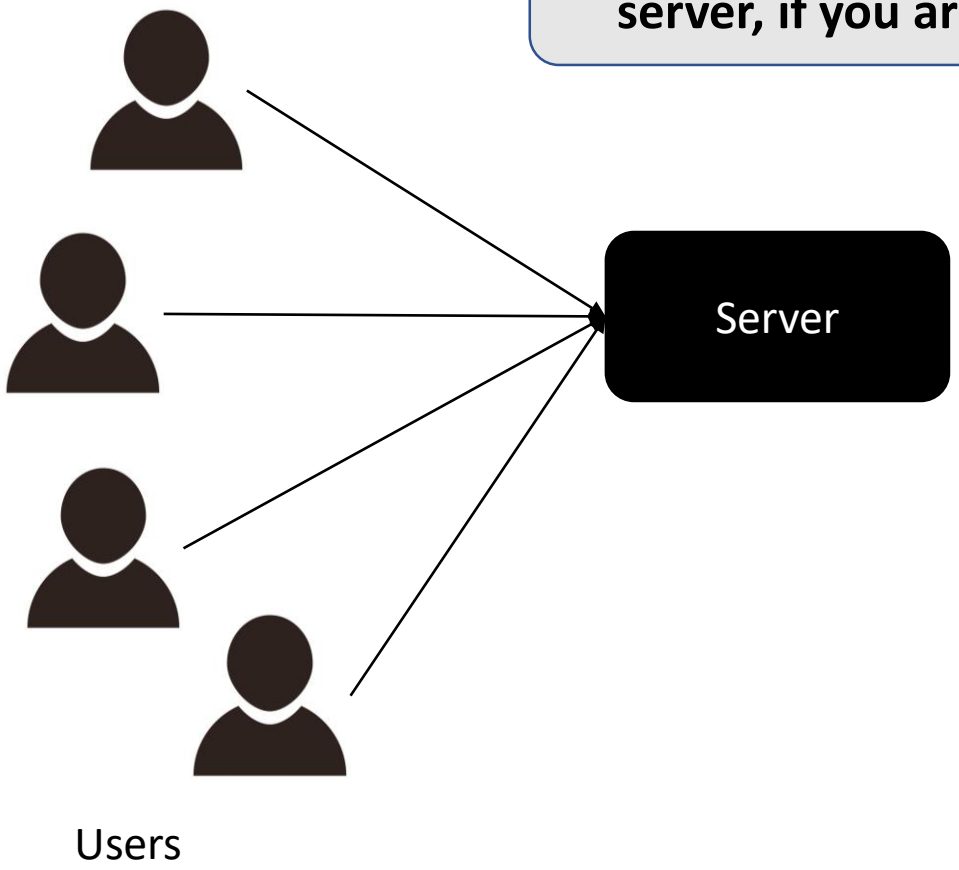
oAuth 2.0 Architecture



https://docs.oracle.com/cd/E82085_01/160027/JOS%20Implementation%20Guide/Output/oauth.htm

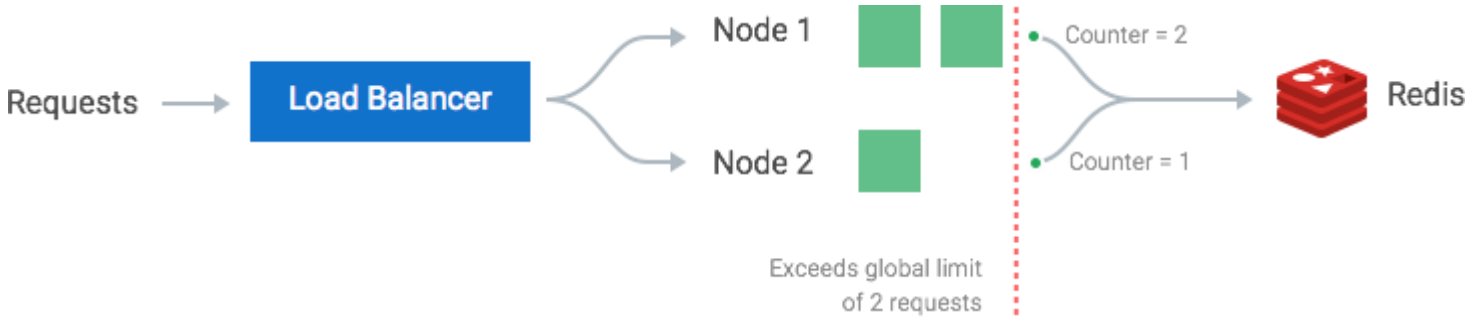
Web Services – Rate Limiting

Can you think of a way to bring down a server, if you are one of the users?



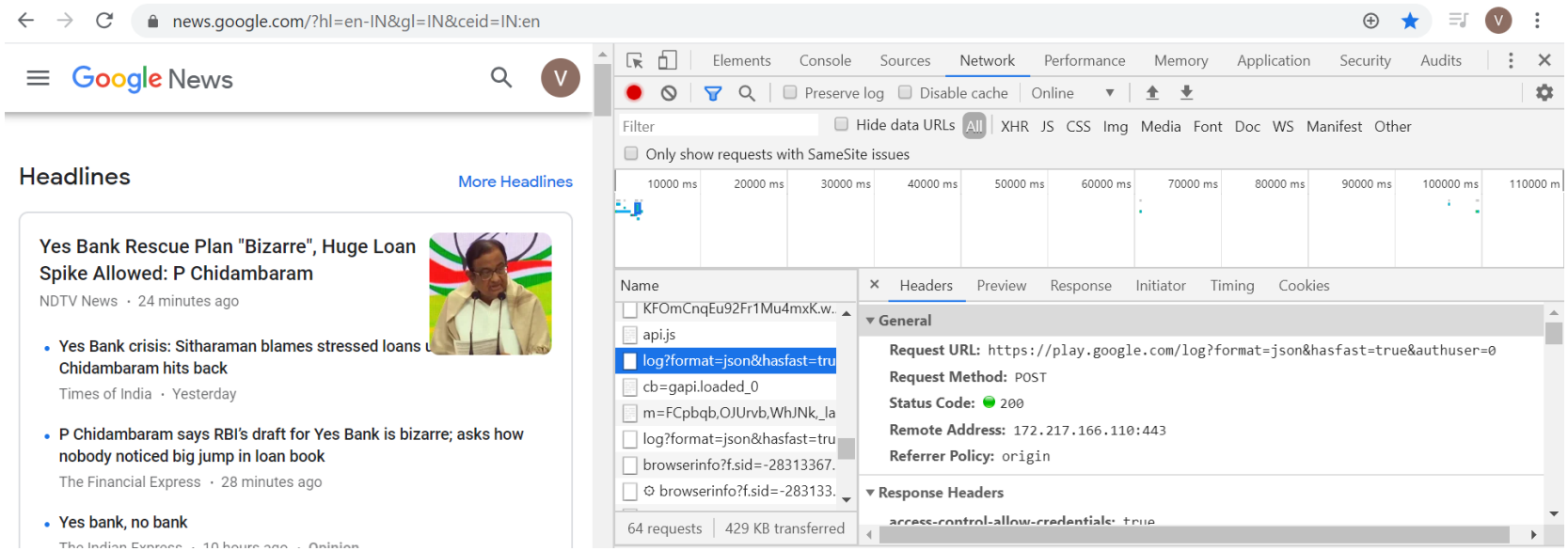
Rate Limiting

- A Leaky Bucket Solution
  - Queue up and service at a specific rate.
- Fixed Window Approach
  - Every request is served in a fixed time slot.
  - If the counter exceeds a threshold, the request is discarded.



https://konghq.com/blog/how-to-design-a-scalable-rate-limiting-algorithm/

Putting it all Together!



Private Cloud

- Many companies build and use their own private cloud.
  - Each private cloud is a single-tenant server or cluster of servers
  - Total control over the resources of the physical hardware layer.
  - No risk of resource or capacity contention.
  - Best suited for privacy and compliance.
  - Expensive!
- Smaller companies that cannot afford a private cloud buy infrastructure (from IaaS) on a public cloud.
- There are also corporates that believe in hybrid cloud.
  - For economies of scale.

Public Cloud

- Storage and Computing services offered by third-party providers over the public Internet, making them available to anyone who wants to use or purchase them.
- Often pay-as-you-go service.
- Sold on-demand.
- No management and maintenance overhead.
- May have restrictions due to security concerns (say, can’t open certain ports).

Hybrid Cloud

- Combines a public cloud and a private cloud by allowing data and applications to be shared between them.
- As demand fluctuates, hybrid cloud computing gives businesses the ability to seamlessly scale their on-premises infrastructure up to the public cloud.
  - No need to make massive capital expenditures to handle short-term spikes.
  - Companies will pay only for resources they temporarily use.

Thank You