TCSS 562: SOFTWARE ENGINEERING FOR CLOUD COMPUTING

Cloud Enabling Technology

Wes J. Lloyd
School of Engineering and Technology
University of Washington – Tacoma

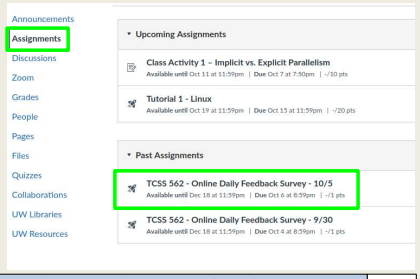
MW 5:50-7:50 PM

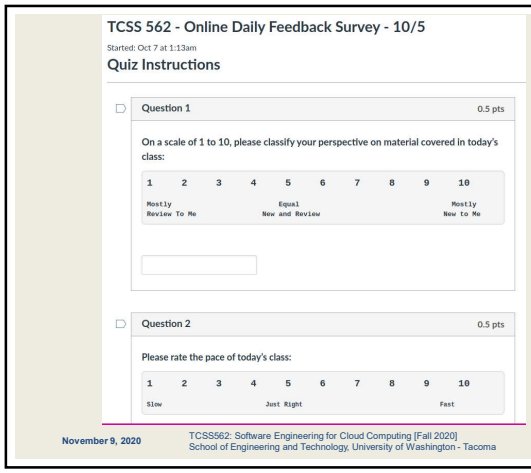
OBJECTIVES – 11/9

- **Questions from 11/4**
- **Quiz 2 – to be posted next week**
- **From: Cloud Computing Concepts, Technology & Architecture:** Chapter 5 - Cloud Enabling Technology
- **2nd hour:**
- Tutorial #6
- Tutorial questions (4, 5, 6)
- Team planning

ONLINE DAILY FEEDBACK SURVEY

- Daily Feedback Quiz in Canvas – Take After Each Class
- Extra Credit for completing





MATERIAL / PACE

- Please classify your perspective on material covered in today's class (20 respondents):
- 1-mostly review, 5-equal new/review, 10-mostly new
- **Average – 6.30 (↓ - *previous 6.74*)**
- Please rate the pace of today's class:
- 1-slow, 5-just right, 10-fast
- **Average – 5.40 (↓ - *previous 5.58*)**

FEEDBACK FROM 11/4

- ***Can you please go over how to do the second part of Tutorial 4, starting from point 9 on page 22? Should we create 2 new lambda functions as we did for "hello" ?***
- Yes
- **Caesar cipher** is a simple encryption method
- The idea is use each character's ASCII value, and apply a numeric SHIFT to the value
- Shift UP to encode
- Shift DOWN to decode
- 'a', shift +10 → 'k'
- 'G', shift +30 → 'e'

Character	ASCII
a	97
b	98
c	99
d	100
e	101
f	102
g	103
h	104
i	105
j	106
k	107
l	108
m	109

Character	ASCII
n	110
o	111
p	112
q	113
r	114
s	115
t	116
u	117
v	118
w	119
x	120
y	121
z	122

Character	ASCII
A	65
B	66
C	67
D	68
E	69
F	70
G	71
H	72
I	73
J	74
K	75
L	76
M	77

Character	ASCII
N	78
O	79
P	80
Q	81
R	82
S	83
T	84
U	85
V	86
W	87
X	88
Y	89
Z	90

Character	ASCII
0	48
1	49
2	50
3	51
4	52
5	53
6	54
7	55
8	56
9	57

FEEDBACK - 2

- **callservice.sh** should call the encode function with JSON (#1)
- Encode produces output (#2)
- Append the shift attribute to encode's output, and call decode to obtain JSON (#3)

(#1) Encode Input

```
{
    "msg": "ServerlessComputingWithFaaS",
    "shift": 22
}
```

(#2) Encode Output

```
{
"value": "OanranhaooYkilqpejcSepdBwwO",
"uuid": "036c9df1-4a1d-4993-bb69-f9fd0ab29816",
"error": "",
"vmuptime": 1539943078,
"newcontainer": 0
}
```

(#3) Decode Output

```
{
    "value": "ServerlessComputingWithFaaS",
    "uuid": "f047b513-e611-4cac-8370-713fb2771db4",
    "error": "",
    "vmuptime": 1539943078,
    "newcontainer": 0
```


FEEDBACK - 3

- There is some flexibility on the coding/implementation
  - See internet for examples of Caesar cipher Java implementation
- The main objective is to complete the 2-function application that accepts a message, shifts the message, and then shifts it back. Both functions should have API gateway endpoints, and callservice.sh should orchestrate the entire "pipeline".
- Running callservice.sh should allow testing of encode & decode

FEEDBACK - 3

- Weekly tutorials are going very fast and finding little tough to keep up.
- Deadline extensions
- Tutorial 3:
  - Wed Nov 4 → Fri Nov 6
- Tutorial 4:
  - Mon Nov 9 → Fri Nov 13

OBJECTIVES – 11/9

- **Questions from 11/4**
- **Quiz 2 – to be posted next week**
- **From: Cloud Computing Concepts, Technology & Architecture:** Chapter 5 - Cloud Enabling Technology
- **2nd hour:**
- Tutorial #6
- Tutorial questions (4, 5, 6)
- Team planning

OBJECTIVES – 11/9

- **Questions from 11/4**
- **Quiz 2 – to be posted next week**
- **From: Cloud Computing Concepts, Technology & Architecture:** Chapter 5 - Cloud Enabling Technology
- **2nd hour:**
- Tutorial #6
- Tutorial questions (4, 5, 6)
- Team planning

CLOUD ENABLING TECHNOLOGY

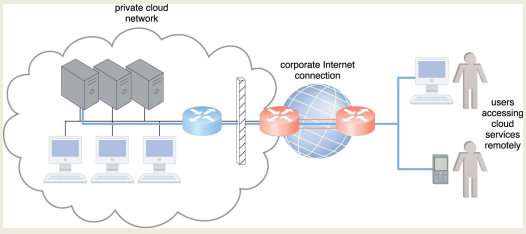
CLOUD ENABLING TECHNOLOGY

- Broadband networks and internet architecture
- Data center technology
- Virtualization technology
- Multitenant technology
- Web/web services technology

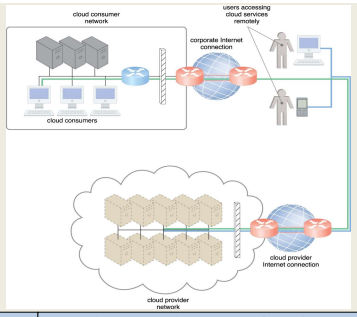
1. BROADBAND NETWORKS AND INTERNET ARCHITECTURE

- Clouds must be connected to a network
- Inter-networking: Users' network must connect to cloud's network
- Public cloud computing relies heavily on the **Internet**

PRIVATE CLOUD NETWORKING



PUBLIC CLOUD NETWORKING



INTERNETWORKING KEY POINTS

- Cloud consumers and providers typically communicate via the internet
- Decentralized provisioning and management model is not controlled by the cloud consumers or providers
- Inter-networking (internet) relies on connectionless packet switching and route-based interconnectivity
- Routers and switches support communication
- Network bandwidth and latency influence QoS, which is heavily impacted by network congestion

CLOUD ENABLING TECHNOLOGY

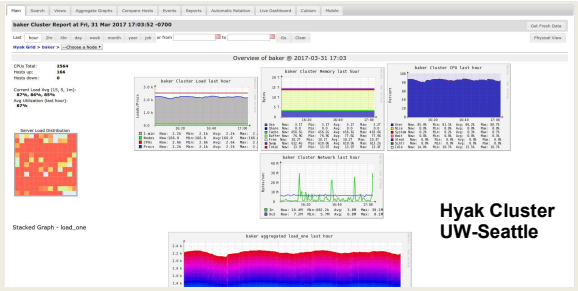
- Broadband networks and internet architecture
- Data center technology
- Virtualization technology
- Multitenant technology
- Web/web services technology

2. DATA CENTER TECHNOLOGY

- Grouping servers together (clusters):
- Enables power sharing
- Higher efficiency in shared IT resource usage (less duplication of effort)
- Improved accessibility and organization

- Key components:
  - Virtualized and physical server resources
  - Standardized, modular hardware
  - Automation support: ease server provisioning, configuration, patching, monitoring without supervision… ***tools are desirable***

CLUSTER MANAGEMENT TOOLS



DATA CENTER TECHNOLOGY – KEY COMPONENTS

- Remote operation / management
- **High availability support**: **redundant everything** Includes: power supplies, cabling, environmental control systems, communication links, duplicate warm replica hardware
- **Secure design**: physical and logical access control
- **Servers**: rackmount, etc.
- **Storage**: hard disk arrays (RAID), storage area network (SAN): disk array with dedicated network, network attached storage (NAS): disk array on network for NFS, etc.
- **Network hardware**: backbone routers (WAN to LAN connectivity), firewalls, VPN gateways, managed switches/routers

CLOUD ENABLING TECHNOLOGY

- Broadband networks and internet architecture
- Data center technology
- Virtualization technology
- Multitenant technology
- Web/web services technology

3. VIRTUALIZATION TECHNOLOGY

- Convert a physical IT resource into a virtual IT resource
- Servers, storage, network, power (virtual UPSs)
- Virtualization supports:
  - Hardware independence
  - Server consolidation
  - Resource replication
  - Resource pooling
  - Elastic scalability
- Virtual servers
  - Operating-system based virtualization
  - Hardware-based virtualization

VIRTUAL MACHINES

- Emulation/simulation of a computer in software
- Provides a substitute for a real computer or server
- Virtualization platforms provide functionality to run an entire operating system
- Allows running multiple different operating systems, or operating systems with different versions simultaneously on the same computer

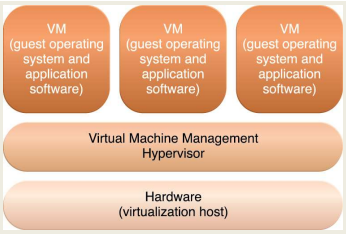
KEY VIRTUALIZATION TRADEOFF

- Tradeoff space:
  What is the "right" level of abstraction?

Degree of Hardware Abstraction

Abstraction Concerns:
Overhead
Performance
Isolation
Security

TYPE 1 HYPERVISOR



- Host OS and VMs run atop the hypervisor
- The boot OS is the hypervisor kernel
- Xen dom0

TYPE 1 HYPERVISOR

- Acts as a control program
- Miniature OS kernel that manages VMs
- Boots and runs on bare metal
- Also known as Virtual Machine Monitor (VMM)
- **Paravirtualization**: Kernel includes I/O drivers
- VM guest OSes must use special kernel to interoperate
- Paravirtualization provides hooks to the guest VMs
- Kernel traps instructions (i.e. device I/O) to implement sharing & multiplexing
- User mode instructions run directly on the CPU
- Objective: minimize virtualization overhead
- Classic example is XEN (dom0 kernel)

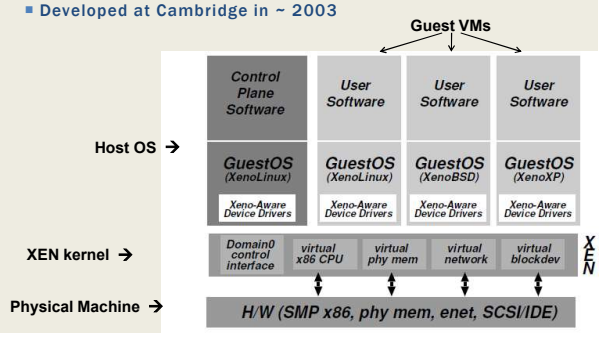
COMMON VMMS: PARAVIRTUALIZATION

- **TYPE 1**
- XEN
- Citrix Xen-server (a commercial version of XEN)
- VMWare ESXi
- KVM (virtualization support in kernel)

- Paravirtual I/O drivers introduced
  - XEN
  - KVM
  - Virtualbox

XEN

- Developed at Cambridge in ~ 2003



XEN - 2

- VMs managed as "domains"
- Domain 0 is the hypervisor domain
  - Host OS is installed to run on bare-metal, but doesn't directly facilitate virtualization (*unlike KVM*)
- Domains 1..n are guests (VMs) – not bare-metal

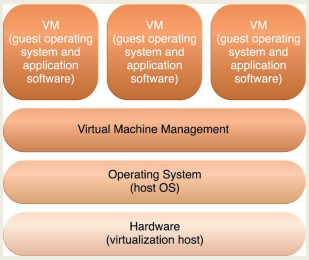
```
xentop - 17:53:48   Xen 3.1.2-398.el5
3 domains: 1 running, 2 blocked, 0 paused, 0 crashed, 0 dying, 0 shutdown
Mem: 8379564k total, 8377876k used, 1688k free    CPUs: 4 @ 2400MHz
      NAME  STATE    CPU(sec) CPU(%)     MEM(k) MEM(%)  MAXMEM(k) MAXMEM(%) VCPUS
 NETS NETTX(k) NETRX(k) VBDS   VBD_OO   VBD_RD   VBD_WR SSID
   centos --b---         46    0.0     532352    6.4    1064960      12.7     1
    1    27960     885    1        0     6313    37119          0
 centos-2 --b---         17    0.0    1056640   12.6    2113536      25.2     1
    1       50       0    1        0     3981      541          0
 Domain-0 -----r       2979   19.3    6568960   78.4   no limit       n/a     4
    4  1057374   290072    0        0        0        0          0
```

XEN - 3

- Physical machine boots special XEN kernel
- Kernel provides paravirtual API to manage CPU & device multiplexing
- Guests require modified XEN-aware kernels
- Xen supports full-virtualization for unmodified OS guests in hvm mode
- Amazon EC2 largely based on modified version of XEN hypervisor (EC2 gens 1-4)
- XEN provides its own CPU schedulers, I/O scheduling

TYPE 2 HYPERVISOR

- Adds additional layer



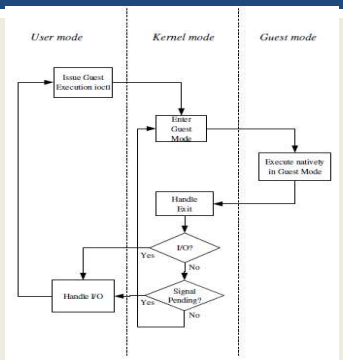
TYPE 2 HYPERVISOR

- Problem: Original x86 CPUs could not trap special instructions
- Instructions not specially marked
- Solution: Use Full Virtualization
- Trap ALL instructions
- "Fully" simulate entire computer
- Tradeoff: Higher Overhead
- Benefit: Can virtualize any operating system without modification

KERNEL BASED VIRTUAL MACHINES (KVM)

- x86 HW notoriously difficult to virtualize
- Extensions added to 64-bit Intel/AMD CPUs
  - Provides hardware assisted virtualization
  - New "guest" operating mode
  - Hardware state switch
  - Exit reason reporting
  - Intel/AMD implementations different
    - Linux uses vendor specific kernel modules

KVM – 2



KVM – 3

- KVM has /dev/kvm device file node
  - Linux character device, with operations:
    - Create new VM
    - Allocate memory to VM
    - Read/write virtual CPU registers
    - Inject interrupts into vCPUs
    - Running vCPUs
- VMs run as Linux processes
  - Scheduled by host Linux OS
  - Can be pinned to specific cores with "taskset"

KVM PARAVIRTUALIZED I/O

- KVM – Virtio
  - Custom Linux based paravirtual device drivers
  - Supersedes QEMU hardware emulation (full virt.)
  - Based on XEN paravirtualized I/O
  - Custom block device driver provides paravirtual device emulation
    - Virtual bus (memory ring buffer)
    - Requires hypercall facility
    - Direct access to memory

KVM DIFFERENCES FROM XEN

- KVM requires CPU VMX support
  - Virtualization management extensions
- KVM can virtualize any OS without special kernels
  - Less invasive
- KVM was originally separate from the Linux kernel, but then integrated
- KVM is type 1 hypervisor because the machine boots Linux which has integrated support for virtualization
- Different than XEN because XEN kernel alone is not a full-fledged OS

KVM ENHANCEMENTS

- Paravirtualized device drivers
  - Virtio
- Guest Symmetric Multiprocessor (SMP) support
  - Leverages multiple on-board CPUs
  - Supported as of Linux 2.6.23
- VM Live Migration
- Linux scheduler integration
  - Optimize scheduler with knowledge that KVM processes are virtual machines

VIRTUALIZATION MANAGEMENT

- Virtual infrastructure management (VIM) tools
- Tools that manage pools of virtual machines, resources, etc.
- Private cloud software systems can be considered as a VIM

- Considerations:
- Performance overhead
  - Paravirtualization: custom OS kernels, I/O passed directly to HW w/ special drivers
- Hardware compatibility for virtualization
- Portability: virtual resources tend to be difficult to migrate cross-clouds

VIRTUAL INFRASTRUCTURE MANAGEMENT (VIM)

- Middleware to manage virtual machines and infrastructure of IaaS "clouds"

- Examples
  - OpenNebula
  - Nimbus
  - Eucalyptus
  - OpenStack

VIM FEATURES

- Create/destroy VM Instances
- Image repository
  - Create/Destroy/Update images
  - Image persistence

- Contextualization of VMs
  - Networking address assignment
    - DHCP / Static IPs
  - Manage SSH keys

VIM FEATURES - 2

- Virtual network configuration/management
  - Public/Private IP address assignment
  - Virtual firewall management
  - Configure/support isolated VLANs (private clusters)
- Support common virtual machine managers (VMMs)
  - XEN, KVM, VMware
  - Support via libvirt library

VIM FEATURES - 3

- Shared "Elastic" block storage
  - Facility to create/update/delete VM disk volumes
    - Amazon EBS
    - Eucalyptus SC
    - OpenStack Volume Controller

CONTAINER ORCHESTRATION FRAMEWORKS

- Middleware to manage Docker application container deployments across virtual clusters of Docker hosts (VMs)
- Considered Infrastructure-as-a-Service
- **Opensource**
- Kubernetes framework
- Docker swarm
- Apache Mesos/Marathon
- **Proprietary**
- Amazon Elastic Container Service

CONTAINER SERVICES

- **Public cloud container cluster services**
- Azure Kubernetes Service (AKS)
- Amazon Elastic Container Service for Kubernetes (EKS)
- Google Kubernetes Engine (GKE)
- **Container-as-a-Service**
- Azure Container Instances (ACI – April 2018)
- AWS Fargate (November 2017)
- Google Kubernetes Engine Serverless Add-on (alpha-July 2018)

CLOUD ENABLING TECHNOLOGY

- Broadband networks and internet architecture
- Data center technology
- Virtualization technology
- Multitenant technology
- Web/web services technology

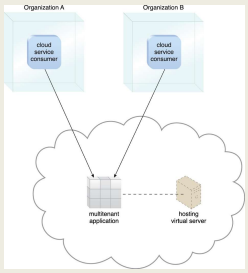
4. MULTITENANT APPLICATIONS

- Each tenant (like in an apartment) has their own view of the application
- Tenants are unaware of their neighbors
- Tenants can only access their data, no access to data and configuration that is not their own

- Customizable features
  - UI, business process, data model, access control
- Application architecture
  - User isolation, data security, recovery/backup by tenant, scalability for a tenant, for tenants, metered usage, data tier isolation

MULTITENANT APPS - 2

- Forms the basis for SaaS (applications)



CLOUD ENABLING TECHNOLOGY

- Broadband networks and internet architecture
- Data center technology
- Virtualization technology
- Multitenant technology
- Web/web services technology

5. WEB SERVICES/WEB

- Web services technology is a key foundation of cloud computing's "**as-a-service**" cloud delivery model
- SOAP – "Simple" object access protocol
  - First generation web services
  - WSDL – web services description language
  - UDDI – universal description discovery and integration
  - SOAP services have their own unique interfaces
- REST – instead of defining a custom technical interface REST services are built on the use of HTTP protocol
- HTTP GET, PUT, POST, DELETE

HYPERTEXT TRANSPORT PROTOCOL (HTTP)

- An ASCII-based request/reply protocol for transferring information on the web
- HTTP request includes:
  - request method (GET, POST, etc.)
  - Uniform Resource Identifier (URI)
  - HTTP protocol version understood by the client
  - headers—extra info regarding transfer request
- HTTP response from server
  - Protocol version & status code →
  - Response headers
  - Response body

HTTP status codes:
2xx — *all is well*
3xx — *resource moved*
4xx — *access problem*
5xx — *server error*

REST: REPRESENTATIONAL STATE TRANSFER

- Web services protocol
- *Supersedes SOAP* – Simple Object Access Protocol
- Access and manipulate web resources with a predefined set of stateless operations (known as web services)
- Requests are made to a URI
- Responses are most often in JSON, but can also be HTML, ASCII text, XML, no real limits as long as text-based
- HTTP verbs: GET, POST, PUT, DELETE, …


```
// SOAP REQUEST

POST /InStock HTTP/1.1
Host: www.bookshop.org
Content-Type: application/soap+xml; charset=utf-8
Content-Length: nnn

<?xml version="1.0"?>
<soap:Envelope
xmlns:soap="http://www.w3.org/2001/12/soap-envelope"
soap:encodingStyle="http://www.w3.org/2001/12/soap-
encoding">
<soap:Body xmlns:m="http://www.bookshop.org/prices">
  <m:GetBookPrice>
    <m:BookName>The Fleamarket</m:BookName>
  </m:GetBookPrice>
</soap:Body>
</soap:Envelope>
```

```
// SOAP RESPONSE
POST /InStock HTTP/1.1
Host: www.bookshop.org
Content-Type: application/soap+xml; charset=utf-8
Content-Length: nnn

<?xml version="1.0"?>
<soap:Envelope
xmlns:soap="http://www.w3.org/2001/12/soap-envelope"
soap:encodingStyle="http://www.w3.org/2001/12/soap-
encoding">
<soap:Body xmlns:m="http://www.bookshop.org/prices">
  <m:GetBookPriceResponse>
    <m: Price>10.95</m: Price>
  </m:GetBookPriceResponse>
</soap:Body>
</soap:Envelope>
```



```
// WSDL Service Definition
<?xml version="1.0" encoding="UTF-8"?>
<definitions  name ="DayOfWeek"
  targetNamespace="http://www.roguewave.com/soapworx/examples/DayOfWeek.wsdl"
  xmlns:tns="http://www.roguewave.com/soapworx/examples/DayOfWeek.wsdl"
  xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xmlns="http://schemas.xmlsoap.org/wsdl/">
  <message name="DayOfWeekInput">
    <part name="date" type="xsd:date"/>
  </message>
  <message name="DayOfWeekResponse">
    <part name="dayOfWeek" type="xsd:string"/>
  </message>
  <portType name="DayOfWeekPortType">
    <operation name="GetDayOfWeek">
      <input message="tns:DayOfWeekInput"/>
      <output message="tns:DayOfWeekResponse"/>
    </operation>
  </portType>
  <binding name="DayOfWeekBinding" type="tns:DayOfWeekPortType">
    <soap:binding style="document"
      transport="http://schemas.xmlsoap.org/soap/http"/>
    <operation name="GetDayOfWeek">
      <soap:operation soapAction="getdayofweek"/>
      <input>
        <soap:body use="encoded"
          namespace="http://www.roguewave.com/soapworx/examples"
          encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"/>
      </input>
      <output>
        <soap:body use="encoded"
        namespace="http://www.roguewave.com/soapworx/examples"
        encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"/>
      </output>
    </operation>
  </binding>
  <service name="DayOfWeekService" >
    <documentation>
      Returns the day-of-week name for a given date
    </documentation>
    <port name="DayOfWeekPort" binding="tns:DayOfWeekBinding">
      <soap:address location="http://localhost:8090/dayofweek/DayOfWeek"/>
    </port>
  </service>
</definitions>
```


REST CLIMATE SERVICES EXAMPLE

- USDA Lat/Long Climate Service Demo
- Just provide a Lat/Long

```
// REST/JSON
// Request climate data for Washington

{
 "parameter": [
  {
    "name": "latitude",
    "value":47.2529
  },
  {
    "name": "longitude",
    "value":-122.4443
  }
  ]
}
```


REST - 2

- App manipulates one or more types of resources.
- Everything the app does can be characterized as some kind of operation on one or more resources.
- Frequently services are CRUD operations (create/read/update/delete)
  - Create a new resource
  - Read resource(s) matching criterion
  - Update data associated with some resource
  - Destroy a particular a resource
- Resources are often implemented as objects in OO languages

REST ARCHITECTURAL ADVANTAGES

- **Performance**: component interactions can be the dominant factor in user-perceived performance and network efficiency
- **Scalability**: to support large numbers of services and interactions among them
- **Simplicity**: of the Uniform Interface
- **Modifiability**: of services to meet changing needs (even while the application is running)
- **Visibility**: of communication between services
- **Portability**: of services by redeployment
- **Reliability**: resists failure at the system level as redundancy of infrastructure is easy to ensure

WE WILL RETURN AT ~7:10PM

OBJECTIVES – 11/9

- Questions from 11/4
- Quiz 2 – to be posted next week
- From: Cloud Computing Concepts, Technology & Architecture:
  Chapter 5 - Cloud Enabling Technology
- 2nd hour:
- Tutorial #6
- Tutorial questions (4, 5, 6)
- Team planning

OBJECTIVES – 11/9

- Questions from 11/4
- Quiz 2 – to be posted next week
- From: Cloud Computing Concepts, Technology & Architecture:
  Chapter 5 - Cloud Enabling Technology
- 2nd hour:
- Tutorial #6
- Tutorial questions (4, 5, 6)
- Team planning

OBJECTIVES – 11/9

- Questions from 11/4
- Quiz 2 – to be posted next week
- From: Cloud Computing Concepts, Technology & Architecture:
  Chapter 5 - Cloud Enabling Technology
- 2nd hour:
- Tutorial #6
- Tutorial questions (4, 5, 6)
- Team planning

QUESTIONS