IIT KHARAGPUR

NPTEL ONLINE
CERTIFICATION COURSES

NPTEL

CLOUD COMPUTING

ARCHITECTURE - Deployment Models

PROF. SOUMYA K. GHOSH
DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
IIT KHARAGPUR

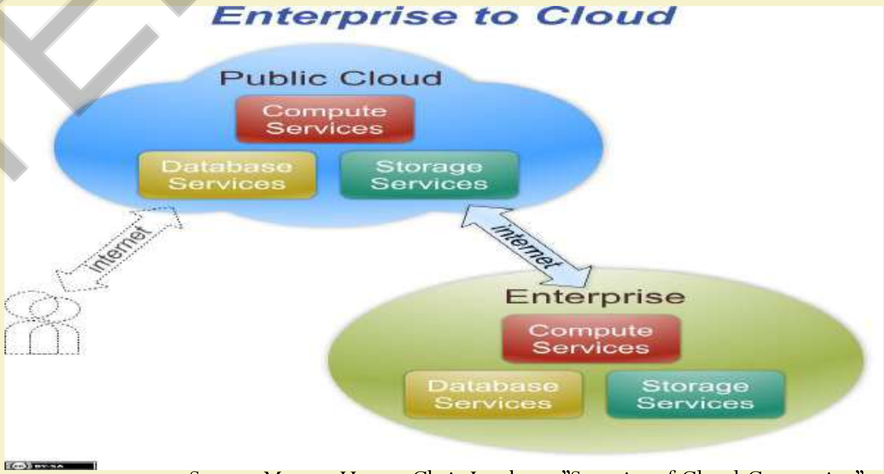
Deployment Models

- Public Cloud
- Private Cloud
- Hybrid Cloud
- Community Cloud

Public Cloud

- Cloud infrastructure is provisioned for open use by the general public. It may be owned, managed, and operated by a business, academic, or government organization, or some combination of them. It exists on the premises of the cloud provider.

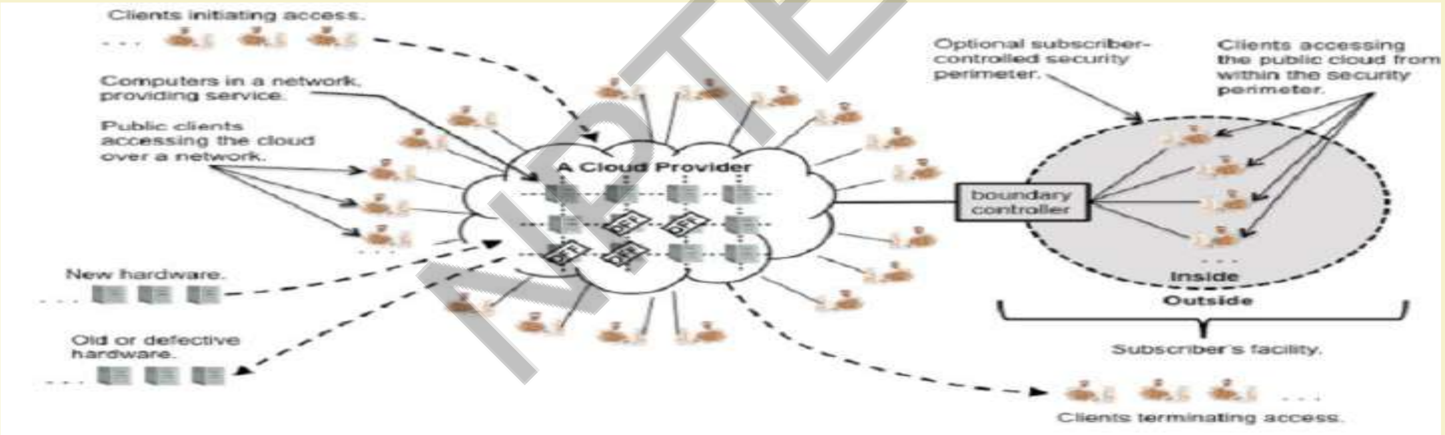
- Examples of Public Cloud:
- Google App Engine
- Microsoft Windows Azure
- IBM Smart Cloud
- Amazon EC2



Source:Marcus Hogue,Chris Jacobson,"Security of Cloud Computing"

Public Cloud

- In Public setting, the provider's computing and storage resources are potentially large; the communication links can be assumed to be implemented over the public Internet; and the cloud serves a diverse pool of clients (and possibly attackers).


Clients initiating access.
Computers in a network, providing service.
Public clients accessing the cloud over a network.
A Cloud Provider
New hardware.
Old or defective hardware.
Optional subscriber-controlled security perimeter.
Clients accessing the public cloud from within the security perimeter.
boundary controller
Inside
Outside
Subscriber's facility.
Clients terminating access.

Source: LeeBadger, and Tim Grance "NIST DRAFT Cloud Computing Synopsis and Recommendations "

Public Cloud

- **Workload locations are hidden from clients (public):**
  - In the public scenario, a provider may migrate a subscriber's workload, whether processing or data, at any time.
  - Workload can be transferred to data centres where cost is low
  - Workloads in a public cloud may be relocated anywhere at any time unless the provider has offered (optional) location restriction policies
- **Risks from multi-tenancy (public):**
  - A single machine may be shared by the workloads of any combination of subscribers (a subscriber's workload may be co-resident with the workloads of competitors or adversaries)
    - Introduces both reliability and security risk

Public Cloud

- Organizations considering the use of an on-site private cloud should consider:
  - **Network dependency (public):**
    - Subscribers connect to providers via the public Internet.
    - Connection depends on Internet's Infrastructure like
      - Domain Name System (DNS) servers
      - Router infrastructure,
      - Inter-router links

Public Cloud

- **Limited visibility and control over data regarding security (public):**
  - The details of provider system operation are usually considered proprietary information and are not divulged to subscribers.
  - In many cases, the software employed by a provider is usually proprietary and not available for examination by subscribers
  - A subscriber cannot verify that data has been completely deleted from a provider's systems.
- **Elasticity: illusion of unlimited resource availability (public):**
  - Public clouds are generally unrestricted in their location or size.
  - Public clouds potentially have high degree of flexibility in the movement of subscriber workloads to correspond with available resources.

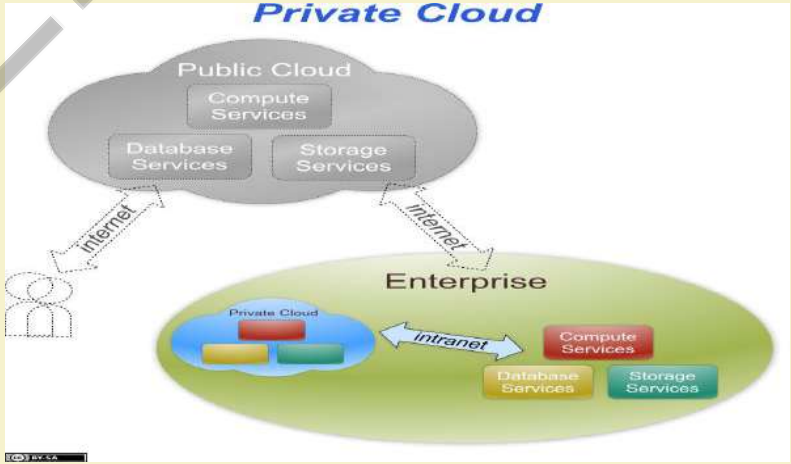
Public Cloud

- **Low up-front costs to migrate into the cloud (public)**
- **Restrictive default service level agreements (public):**
  - The default service level agreements of public clouds specify limited promises that providers make to subscribers

Private Cloud

- The cloud infrastructure is provisioned for exclusive use by a single organization comprising multiple consumers (e.g., business units). It may be owned, managed, and operated by the organization, a third party, or some combination of them, and it may exist on or off premises.

- Examples of Private Cloud:
  - Eucalyptus
  - Ubuntu Enterprise Cloud - UEC
  - Amazon VPC (Virtual Private Cloud)
  - VMware Cloud Infrastructure Suite
  - Microsoft ECI data center.

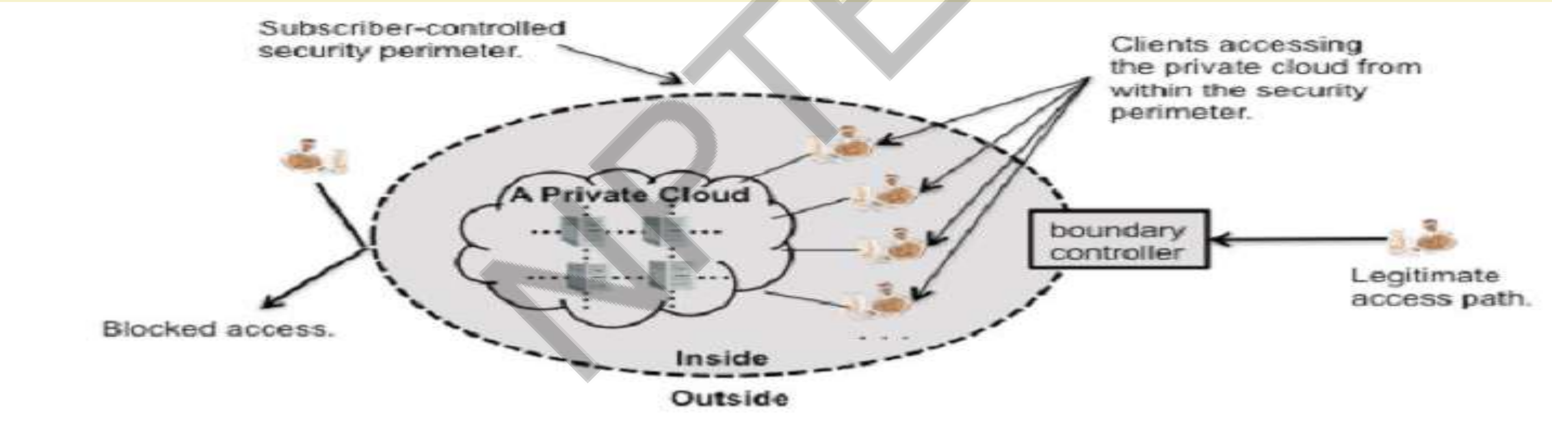


Private Cloud

- Contrary to popular belief, private cloud may exist off premises and can be managed by a third party. Thus, two private cloud scenarios exist, as follows:
- On-site Private Cloud
  - Applies to private clouds implemented at a customer's premises.
- Outsourced Private Cloud
  - Applies to private clouds where the server side is outsourced to a hosting company.

On-site Private Cloud

- The security perimeter extends around both the subscriber's on-site resources and the private cloud's resources.
- Security perimeter does not guarantees control over the private cloud's resources but subscriber can exercise control over the resources.


Subscriber-controlled security perimeter.
Clients accessing the private cloud from within the security perimeter.
A Private Cloud
boundary controller
Legitimate access path.
Blocked access.
Inside
Outside

Source: LeeBadger, and Tim Grance "NIST DRAFT Cloud Computing Synopsis and Recommendations "

On-site Private Cloud

- Organizations considering the use of an on-site private cloud should consider:
  - **Network dependency (on-site-private):**
  - **Subscribers still need IT skills (on-site-private):**
    - Subscriber organizations will need the traditional IT skills required to manage user devices that access the private cloud, and will require cloud IT skills as well.
  - **Workload locations are hidden from clients (on-site-private):**
    - To manage a cloud's hardware resources, a private cloud must be able to migrate workloads between machines without inconveniencing clients. With an on-site private cloud, however, a subscriber organization chooses the physical infrastructure, but individual clients still may not know where their workloads physically exist within the subscriber organization's infrastructure

On-site Private Cloud

- **Risks from multi-tenancy (on-site-private):**
  - Workloads of different clients may reside concurrently on the same systems and local networks, separated only by access policies implemented by a cloud provider's software. A flaw in the software or the policies could compromise the security of a subscriber organization by exposing client workloads to one another
- **Data import/export, and performance limitations (on-site-private):**
  - On-demand bulk data import/export is limited by the on-site private cloud's network capacity, and real-time or critical processing may be problematic because of networking limitations.

On-site Private Cloud

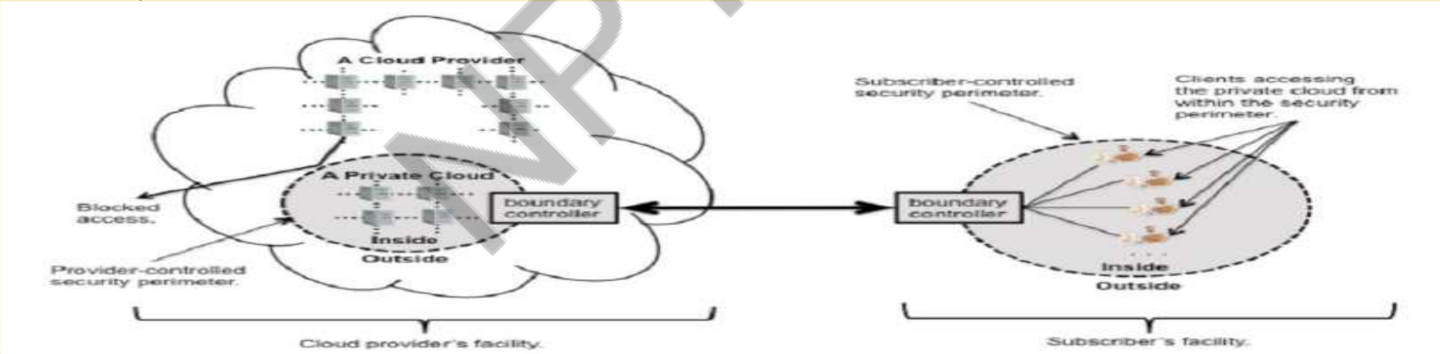
- **Potentially strong security from external threats (on-site-private):**
  - In an on-site private cloud, a subscriber has the option of implementing an appropriately strong security perimeter to protect private cloud resources against external threats to the same level of security as can be achieved for non-cloud resources.
- **Significant-to-high up-front costs to migrate into the cloud (on-site-private):**
  - An on-site private cloud requires that cloud management software be installed on computer systems within a subscriber organization. If the cloud is intended to support process-intensive or data-intensive workloads, the software will need to be installed on numerous commodity systems or on a more limited number of high-performance systems. Installing cloud software and managing the installations will incur significant up-front costs, even if the cloud software itself is free, and even if much of the hardware already exists within a subscriber organization.

On-site Private Cloud

- **Limited resources (on-site-private):**
  - An on-site private cloud, at any specific time, has a fixed computing and storage capacity that has been sized to correspond to anticipated workloads and cost restrictions.

Outsourced Private Cloud

- Outsourced private cloud has two security perimeters, one implemented by a cloud subscriber (on the right) and one implemented by a provider.
- Two security perimeters are joined by a protected communications link.
- The security of data and processing conducted in the outsourced private cloud depends on the strength and availability of both security perimeters and of the protected communication link.


A Cloud Provider
A Private Cloud
boundary controller
Blocked access.
Inside
Outside
Provider-controlled security perimeter.
Cloud provider's facility.
Subscriber-controlled security perimeter.
Clients accessing the private cloud from within the security perimeter.
boundary controller
Inside
Outside
Subscriber's facility.

Outsourced Private Cloud

- Organizations considering the use of an outsourced private cloud should consider:
  - **Network Dependency (outsourced-private):**
    - In the outsourced private scenario, subscribers may have an option to provision unique protected and reliable communication links with the provider.
  - **Workload locations are hidden from clients (outsourced-private):**
  - **Risks from multi-tenancy (outsourced-private):**
    - The implications are the same as those for an on-site private cloud.

Outsourced Private Cloud

- **Data import/export, and performance limitations (outsourced-private):**
  - On-demand bulk data import/export is limited by the network capacity between a provider and subscriber, and real-time or critical processing may be problematic because of networking limitations. In the outsourced private cloud scenario, however, these limits may be adjusted, although not eliminated, by provisioning high-performance and/or high-reliability networking between the provider and subscriber.
- **Potentially strong security from external threats (outsourced-private):**
  - As with the on-site private cloud scenario, a variety of techniques exist to harden a security perimeter. The main difference with the outsourced private cloud is that the techniques need to be applied both to a subscriber's perimeter and provider's perimeter, and that the communications link needs to be protected.

Outsourced Private Cloud

- **Modest-to-significant up-front costs to migrate into the cloud (outsourced-private):**
  - In the outsourced private cloud scenario, the resources are provisioned by the provider
  - Main start-up costs for the subscriber relate to:
    - Negotiating the terms of the service level agreement (SLA)
    - Possibly upgrading the subscriber's network to connect to the outsourced private cloud
    - Switching from traditional applications to cloud-hosted applications,
    - Porting existing non-cloud operations to the cloud
    - Training

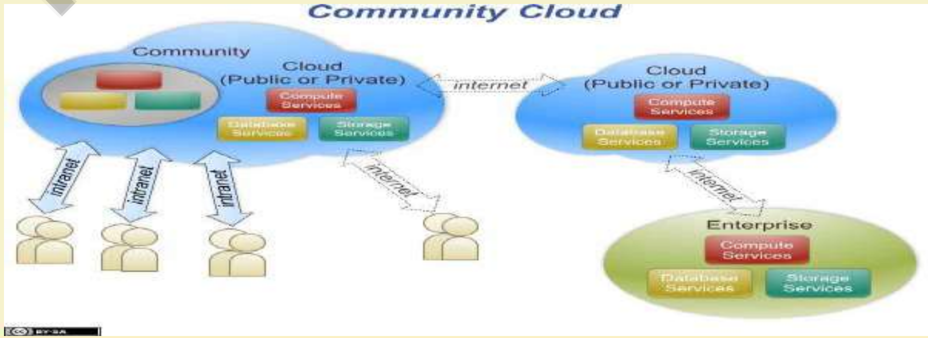
Outsourced Private Cloud

- **Extensive resources available (outsourced-private):**
  - In the case of the outsourced private cloud, a subscriber can rent resources in any quantity offered by the provider. Provisioning and operating computing equipment at scale is a core competency of providers.

Community Cloud

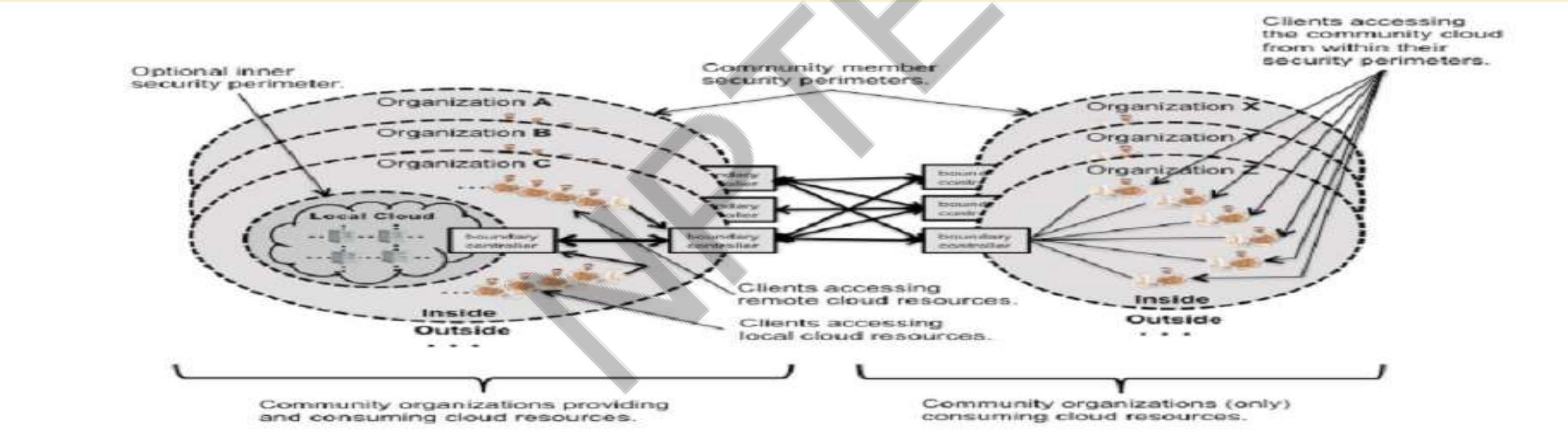
- Cloud infrastructure is provisioned for exclusive use by a specific community of consumers from organizations that have shared concerns (e.g., mission, security requirements, policy, and compliance considerations). It may be owned, managed, and operated by one or more of the organizations in the community, a third party, or some combination of them, and it may exist on or off premises.

- Examples of Community Cloud:
  - Google Apps for Government
  - Microsoft Government Community Cloud



On-site Community Cloud

- Community cloud is made up of a set of participant organizations. Each participant organization may provide cloud services, consume cloud services, or both
- At least one organization must provide cloud services
- Each organization implements a security perimeter



Optional inner security perimeter.
Community member security perimeters.
Clients accessing the community cloud from within their security perimeters.
Organization A
Organization B
Organization C
Organization X
Organization Y
Organization Z
Local Cloud
boundary controller
boundary controller
boundary controller
Clients accessing remote cloud resources.
Clients accessing local cloud resources.
Inside
Outside
Inside
Outside
Community organizations providing and consuming cloud resources.
Community organizations (only) consuming cloud resources.

Source: LeeBadger, and Tim Grance "NIST DRAFT Cloud Computing Synopsis and Recommendations "

On-site Community Cloud

- The participant organizations are connected via links between the boundary controllers that allow access through their security perimeters
- Access policy of a community cloud may be complex
  - Ex. :if there are N community members, a decision must be made, either implicitly or explicitly, on how to share a member's local cloud resources with each of the other members
  - Policy specification techniques like role-based access control (RBAC), attribute-based access control can be used to express sharing policies.

On-site Community Cloud

- Organizations considering the use of an on-site community cloud should consider:
  - **Network Dependency (on-site community):**
    - The subscribers in an on-site community cloud need to either provision controlled inter-site communication links or use cryptography over a less controlled communications media (such as the public Internet).
    - The reliability and security of the community cloud depends on the reliability and security of the communication links.

On-site Community Cloud

- **Subscribers still need IT skills (on-site-community).**
  - Organizations in the community that provides cloud resources, requires IT skills similar to those required for the on-site private cloud scenario except that the overall cloud configuration may be more complex and hence require a higher skill level.
  - Identity and access control configurations among the participant organizations may be complex
- **Workload locations are hidden from clients (on-site-community):**
  - Participant Organizations providing cloud services to the community cloud may wish to employ an outsourced private cloud as a part of its implementation strategy.

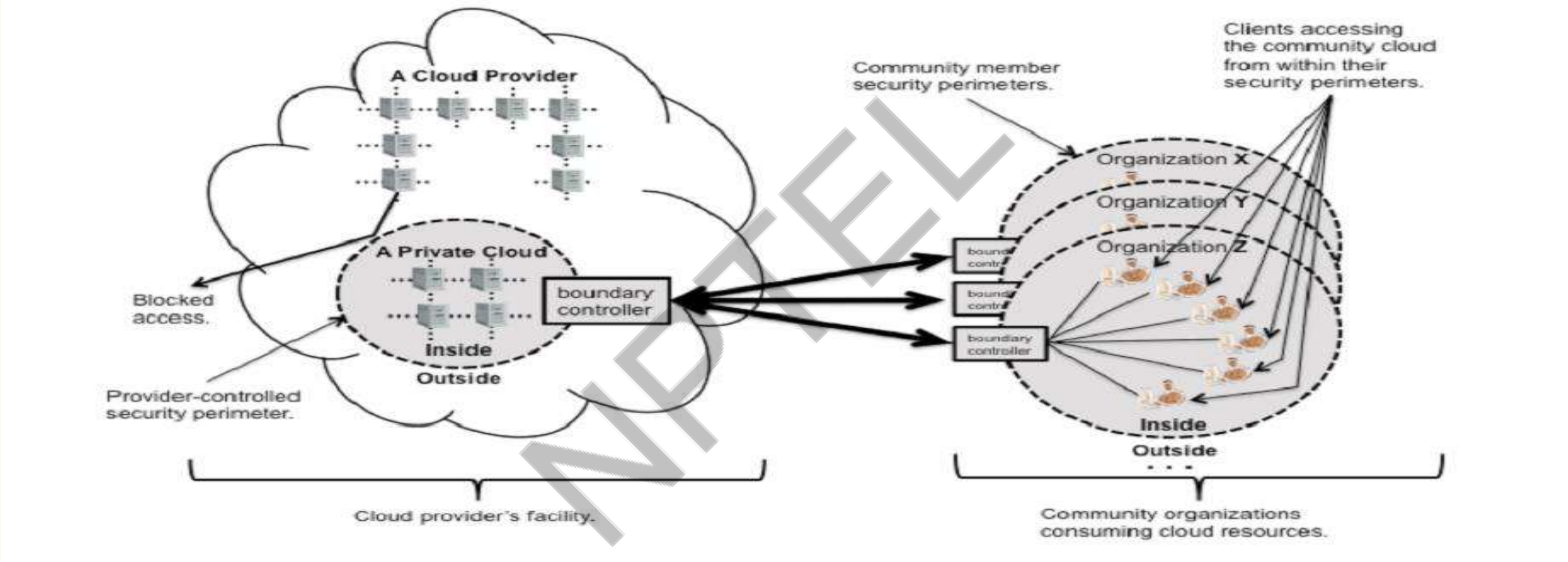
On-site Community Cloud

- **Data import/export, and performance limitations (on-site-community):**
  - The communication links between the various participant organizations in a community cloud can be provisioned to various levels of performance, security and reliability, based on the needs of the participant organizations. The network-based limitations are thus similar to those of the outsourced-private cloud scenario.
- **Potentially strong security from external threats (on-site-community):**
  - The security of a community cloud from external threats depends on the security of all the security perimeters of the participant organizations and the strength of the communications links. These dependencies are essentially similar to those of the outsourced private cloud scenario, but with possibly more links and security perimeters.

On-site Community Cloud

- **Highly variable up-front costs to migrate into the cloud (on-site-community):**
  - The up-front costs of an on-site community cloud for a participant organization depend greatly on whether the organization plans to consume cloud services only or also to provide cloud services. For a participant organization that intends to provide cloud services within the community cloud, the costs appear to be similar to those for the on-site private cloud scenario (i.e., significant-to-high).

Outsourced Community Cloud


A Cloud Provider
A Private Cloud
boundary controller
Inside
Outside
Blocked access.
Provider-controlled security perimeter.
Cloud provider's facility.
Community member security perimeters.
Clients accessing the community cloud from within their security perimeters.
Organization X
Organization Y
Organization Z
boundary controller
Inside
Outside
Community organizations consuming cloud resources.

Source: LeeBadger, and Tim Grance "NIST DRAFT Cloud Computing Synopsis and Recommendations "

Outsourced Community Cloud

- Organizations considering the use of an on-site community cloud should consider:
- **Network dependency (outsourced-community):**
  - The network dependency of the outsourced community cloud is similar to that of the outsourced private cloud. The primary difference is that multiple protected communications links are likely from the community members to the provider's facility.
- **Workload locations are hidden from clients (outsourced-community).**
  - Same as the outsourced private cloud

Outsourced Community Cloud

- **Risks from multi-tenancy (outsourced-community):**
  - Same as the on-site community cloud
- **Data import/export, and performance limitations (outsourced-community):**
  - Same as outsourced private cloud
- **Potentially strong security from external threats (outsourced-community):**
  - Same as the on-site community cloud
- **Modest-to-significant up-front costs to migrate into the cloud (outsourced-community):**
  - Same as outsourced private cloud

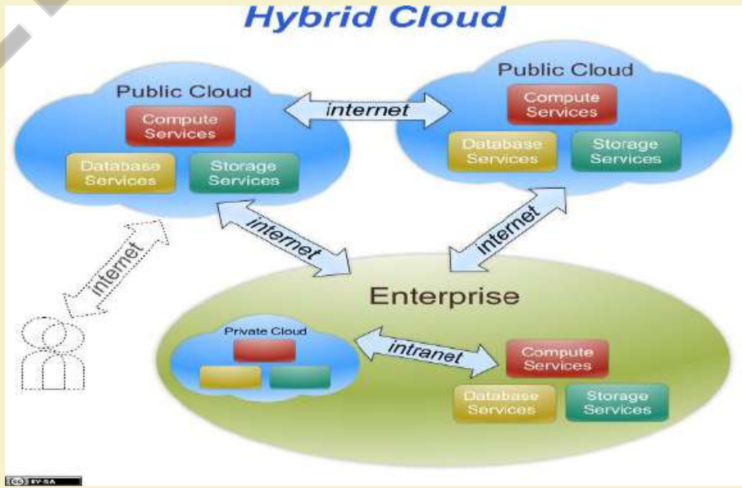
Outsourced Community Cloud

- **Extensive resources available (outsourced-community).**
  - Same as outsourced private cloud

Hybrid Cloud

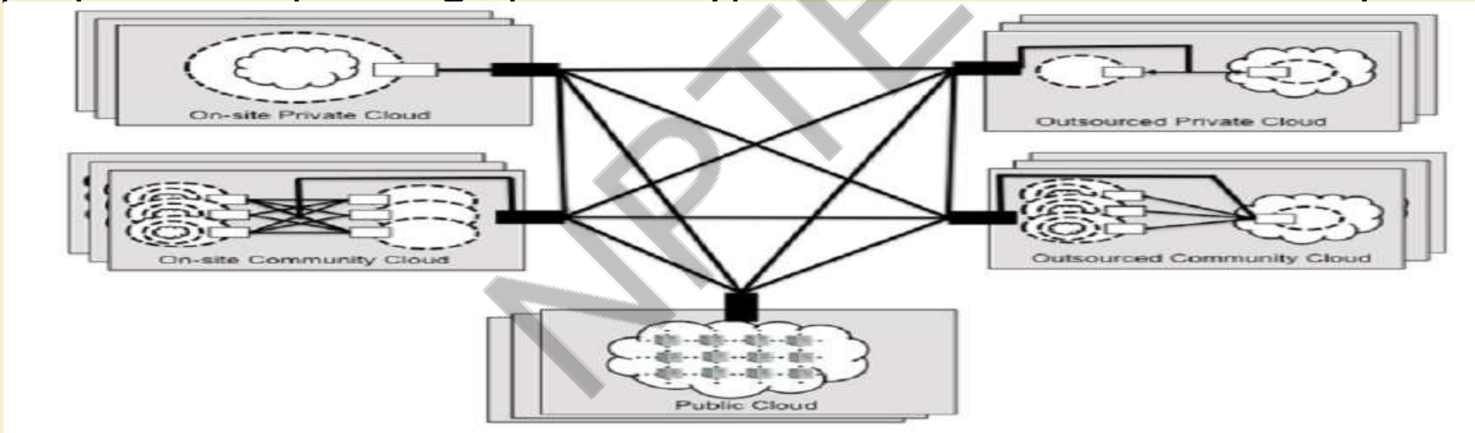
- The cloud infrastructure is a composition of two or more distinct cloud infrastructures (private, community, or public) that remain unique entities, but are bound together by standardized or proprietary technology that enables data and application portability

- Examples of Hybrid Cloud:
  - Windows Azure (capable of Hybrid Cloud)
  - VMware vCloud (Hybrid Cloud Services)



Hybrid Cloud

- A hybrid cloud is composed of two or more private, community, or public clouds.
- They have significant variations in performance, reliability, and security properties depending upon the type of cloud chosen to build hybrid cloud.



Source: LeeBadger, and Tim Grance "NIST DRAFT Cloud Computing Synopsis and Recommendations "

Hybrid Cloud

- A hybrid cloud can be extremely complex
- A hybrid cloud may change over time with constituent clouds joining and leaving.

Thank You!

IIT KHARAGPUR

NPTEL ONLINE
CERTIFICATION COURSES

NPTEL

CLOUD COMPUTING

Virtualization

PROF. SOUMYA K. GHOSH

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

IIT KHARAGPUR

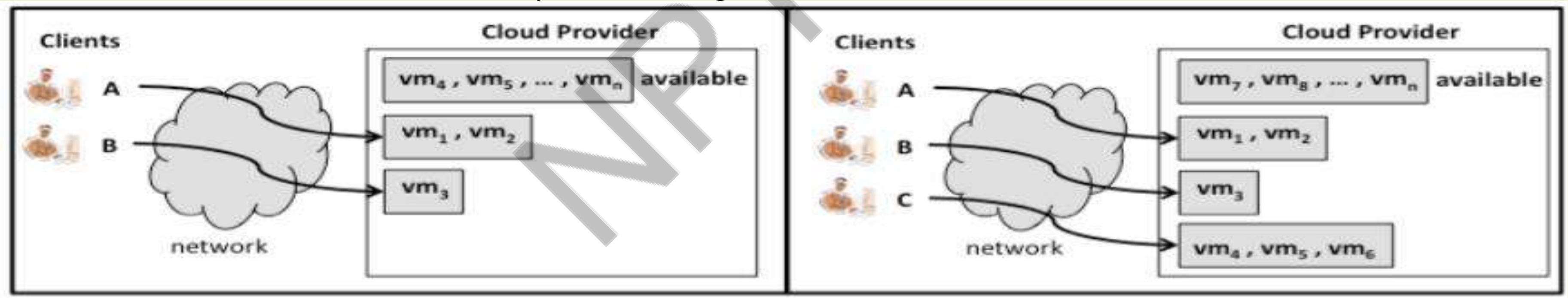
IaaS – Infrastructure as a Service

- What does a subscriber get?
  - Access to virtual computers, network-accessible storage, network infrastructure components such as firewalls, and configuration services.

- How are usage fees calculated?
  - Typically, per CPU hour, data GB stored per hour, network bandwidth consumed, network infrastructure used (e.g., IP addresses) per hour, value-added services used (e.g., monitoring, automatic scaling)

IaaS Provider/Subscriber Interaction Dynamics

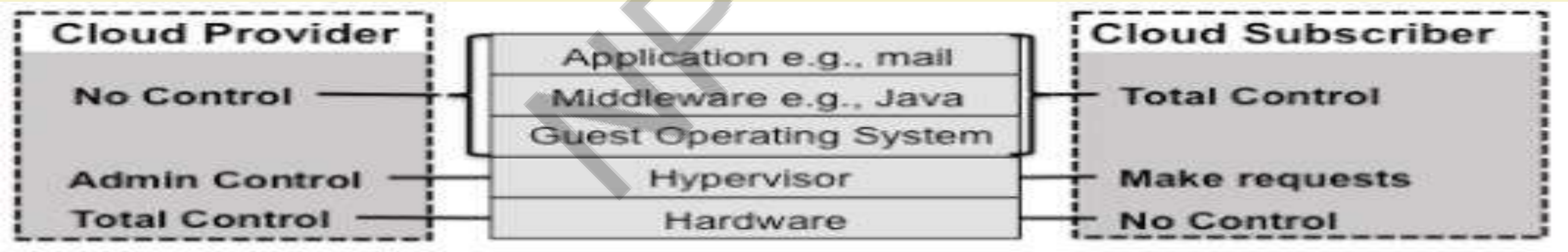
- The provider has a number of available virtual machines (vm's) that it can allocate to clients.
  - Client A has access to vm1 and vm2, Client B has access to vm3 and Client C has access to vm4, vm5 and vm6
  - Provider retains only vm7 through vmN



Source: LeeBadger, and Tim Grance "NIST DRAFT Cloud Computing Synopsis and Recommendations "

IaaS Component Stack and Scope of Control

- IaaS component stack comprises of hardware, operating system, middleware, and applications layers.
- Operating system layer is split into two layers.
    - Lower (and more privileged) layer is occupied by the Virtual Machine Monitor (VMM), which is also called the Hypervisor
    - Higher layer is occupied by an operating system running within a VM called a guest operating system



Source: LeeBadger, and Tim Grance "NIST DRAFT Cloud Computing Synopsis and Recommendations "

IaaS Component Stack and Scope of Control

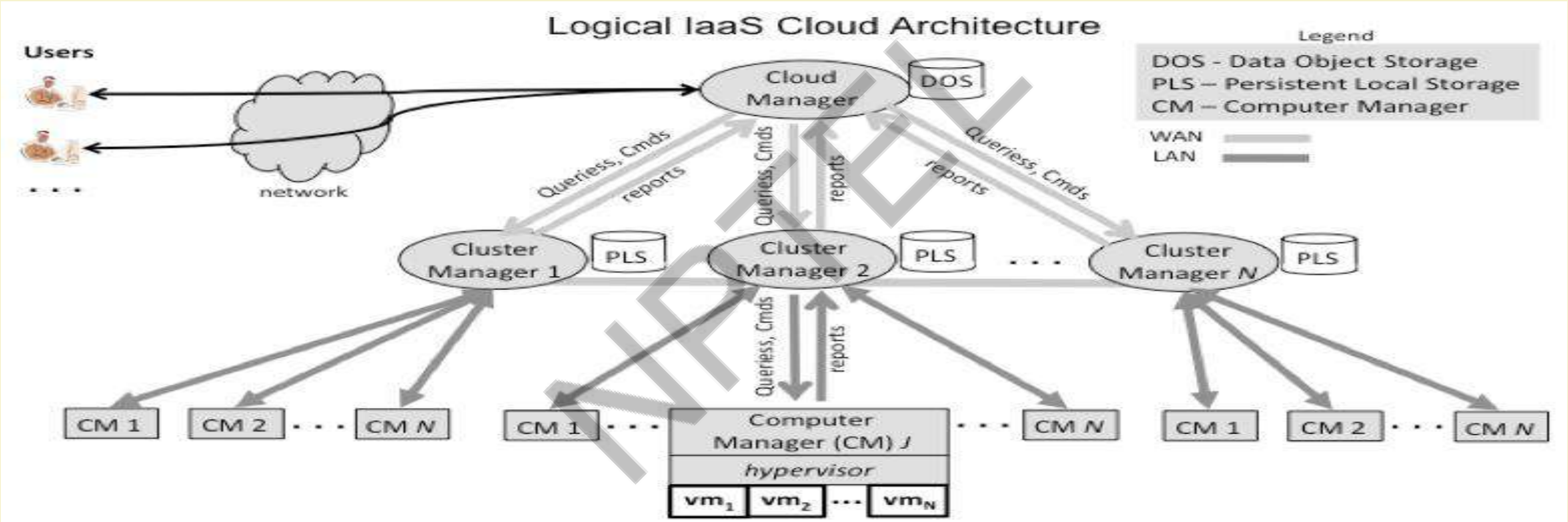
- In IaaS Cloud provider maintains total control over the physical hardware and administrative control over the hypervisor layer
- Subscriber controls the Guest OS, Middleware and Applications layers.
- Subscriber is free (using the provider's utilities) to load any supported operating system software desired into the VM.
- Subscriber typically maintains complete control over the operation of the guest operating system in each VM.

IaaS Component Stack and Scope of Control

- A hypervisor uses the hardware to synthesize one or more Virtual Machines (VMs); each VM is "an efficient, isolated duplicate of a real machine" .
- Subscriber rents access to a VM, the VM appears to the subscriber as actual computer hardware that can be administered (e.g., powered on/off, peripherals configured) via commands sent over a network to the provider.

IaaS Cloud Architecture

- Logical view of IaaS cloud structure and operation



Source: LeeBadger, and Tim Grance "NIST DRAFT Cloud Computing Synopsis and Recommendations "

IaaS Cloud Architecture

- Three-level hierarchy of components in IaaS cloud systems
  - *Top level* is responsible for *central control*
  - *Middle level* is responsible for *management of possibly large computer clusters* that may be *geographically distant* from one another
  - *Bottom level* is responsible for *running the host computer systems* on which virtual machines are created.
- Subscriber queries and commands generally flow into the system at the top and are forwarded down through the layers that either answer the queries or execute the commands

IaaS Cloud Architecture

- Cluster Manager can be geographically distributed
- Within a cluster manger computer manger is connected via high speed network.

Operation of the Cloud Manager

- Cloud Manager is the public access point to the cloud where subscribers sign up for accounts, manage the resources they rent from the cloud, and access data stored in the cloud.
- Cloud Manager has mechanism for:
  - Authenticating subscribers
  - Generating or validating access credentials that subscriber uses when communicating with VMs.
  - Top-level resource management.
- For a subscriber’s request cloud manager determines if the cloud has enough free resources to satisfy the request

Data Object Storage (DOS)

- DOS generally stores the subscriber’s metadata like user credentials, operating system images.
- DOS service is (usually) single for a cloud.

Operation of the Cluster Managers

- Each *Cluster Manager* is responsible for the operation of a collection of computers that are connected via high speed local area networks
- *Cluster Manager* receives resource allocation commands and queries from the *Cloud Manager*, and calculates whether part or all of a command can be satisfied using the resources of the computers in the cluster.
- *Cluster Manager* queries the *Computer Managers* for the computers in the cluster to determine resource availability, and returns messages to the *Cloud Manager*

Operation of the Cluster Managers

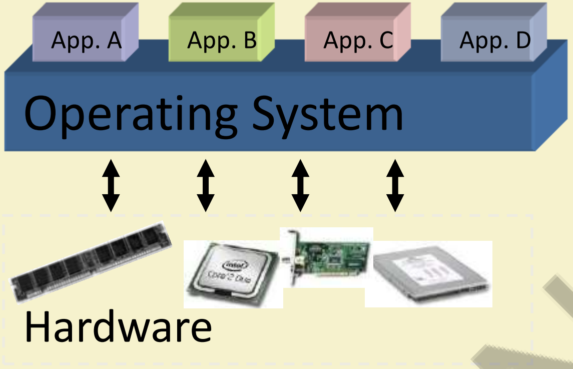
- Directed by the Cloud Manager, a Cluster Manager then instructs the Computer Managers to perform resource allocation, and reconfigures the virtual network infrastructure to give the subscriber uniform access.
- Each Cluster Manager is connected to Persistent Local Storage (PLS)
- PLS provide persistent disk-like storage to Virtual Machine

Operation of the Computer Managers

- At the lowest level in the hierarchy computer manger runs on each computer system and uses the concept of virtualization to provide Virtual Machines to subscribers
- Computer Manger maintains status information including how many virtual machines are running and how many can still be started
- Computer Manager uses the command interface of its hypervisor to start, stop, suspend, and reconfigure virtual machines

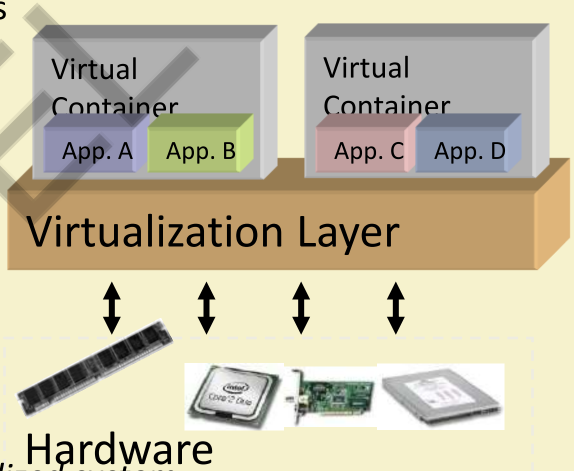
Virtualization

- Virtualization is a broad term (virtual memory, storage, network, etc)
- Focus: **Platform virtualization**
- Virtualization basically allows one computer to do the job of multiple computers, by sharing the resources of a single hardware across multiple environments



‘Non-virtualized’ system

A single OS controls all hardware platform resources



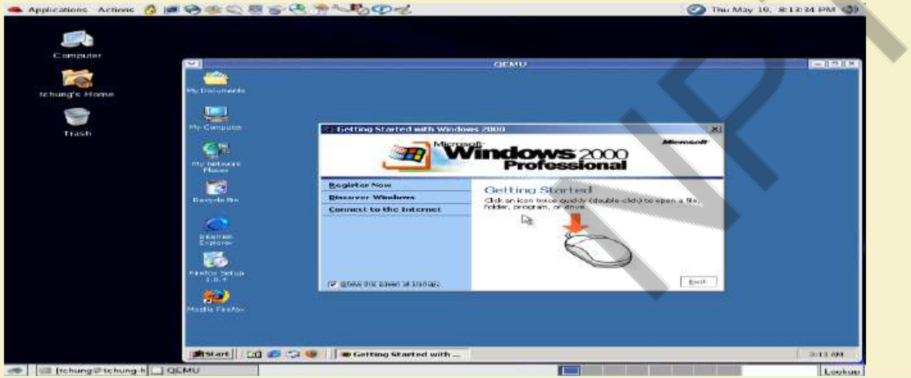
Virtualized system

It makes it possible to run multiple Virtual Containers on a single physical platform

Source: www.dc.uba.ar/events/eci/2008/courses/n2/Virtualization-Introduction.ppt

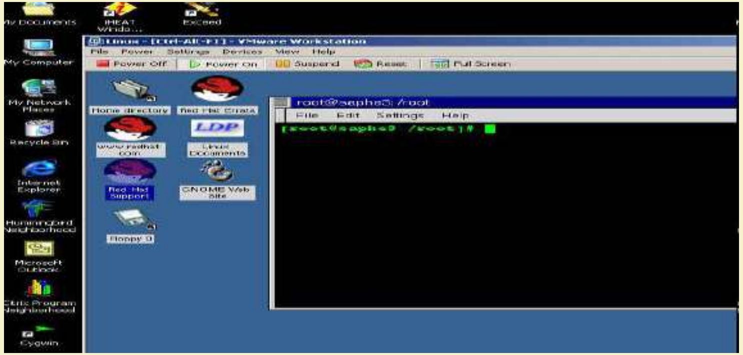
Virtualization

- Virtualization is way to run **multiple operating systems** and **user applications** on the same hardware
  - E.g., run both Windows and Linux on the same laptop
- How is it different from **dual-boot**?
  - Both OSes run **simultaneously**
- The OSes are completely **isolated** from each other



Click an icon twice quickly (double-click) to open a file, folder, program, or drive.

☑ Show this screen at startup



```
[root@sapho3 /root]#
```

Hypervisor or Virtual Machine Monitor

Research Paper :Popek and Goldberg, "Formal requirements for virtualizable third generation architectures", CACM 1974 (http://portal.acm.org/citation.cfm?doid=361011.361073).

A **hypervisor** or **virtual machine monitor** runs the guest OS directly on the CPU. (This only works if the guest OS uses the same instruction set as the host OS.) Since the guest OS is running in user mode, privileged instructions must be intercepted or replaced. This further imposes restrictions on the instruction set for the CPU, as observed in a now-famous paper by Popek and Goldberg identify three goals for a virtual machine architecture:

- *Equivalence*: The VM should be indistinguishable from the underlying hardware.
- *Resource control*: The VM should be in complete control of any virtualized resources.
- *Efficiency*: Most VM instructions should be executed directly on the underlying CPU without involving the hypervisor.

Source: www.dc.uba.ar/events/eci/2008/courses/n2/Virtualization-Introduction.ppt

Hypervisor or Virtual Machine Monitor

Popek and Goldberg describe (and give a formal proof of) the requirements for the CPU's instruction set to allow these properties. The main idea here is to classify instructions into

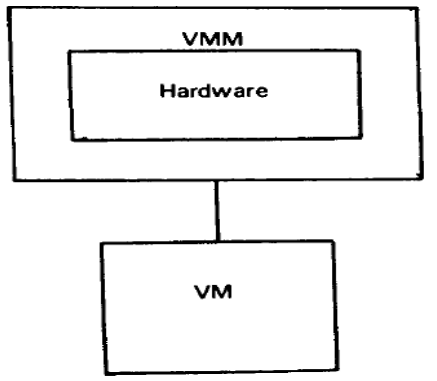
- **privileged** instructions, which cause a trap if executed in user mode, and
- **sensitive** instructions, which change the underlying resources (e.g. doing I/O or changing the page tables) or observe information that indicates the current privilege level (thus exposing the fact that the guest OS is not running on the bare hardware).
- The former class of sensitive instructions are called **control sensitive** and the latter **behavior sensitive** in the paper, but the distinction is not particularly important.

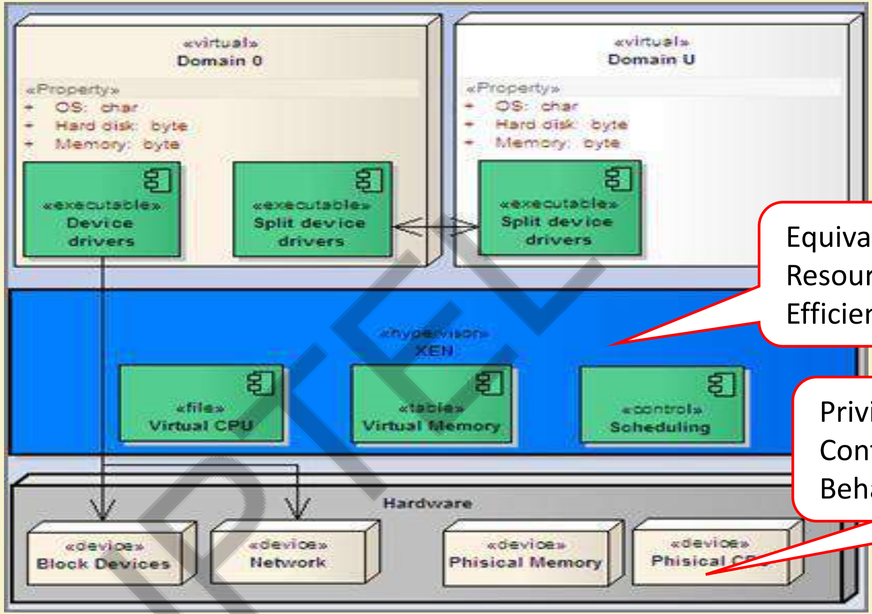
What Popek and Goldberg show is that we can only *run a virtual machine with all three desired properties if the sensitive instructions are a subset of the privileged instructions*. If this is the case, then we can run most instructions directly, and any sensitive instructions trap to the hypervisor which can then emulate them (hopefully without much slowdown).

Source: www.dc.uba.ar/events/eci/2008/courses/n2/Virtualization-Introduction.ppt

VMM and VM

Fig. 1. The virtual machine monitor.





Equivalence
Resource Control
Efficiency

Privileged instructions
Control sensitive
Behavior sensitive

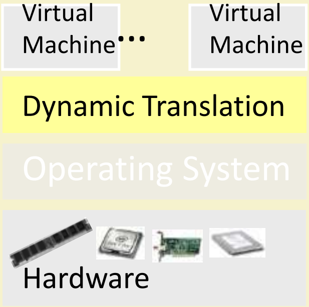
- For any conventional third generation computer, a VMM may be constructed if the set of sensitive instructions for that computer is a subset of the set of privileged instructions
- A conventional third generation computer is recursively virtualizable if it is virtualizable and a VMM without any timing dependencies can be constructed for it.

Source: www.dc.uba.ar/events/eci/2008/courses/n2/Virtualization-Introduction.ppt

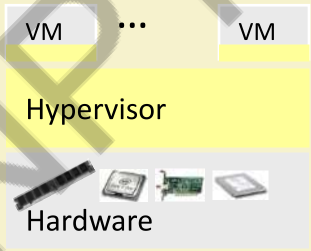
Approaches to Server Virtualization

Evolution of Software Solutions

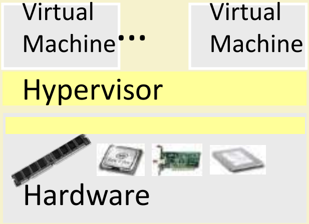
- 1st Generation: Full virtualization (Binary rewriting)
  - Software Based
  - VMware and Microsoft



- 2nd Generation: Para-virtualization
  - Cooperative virtualization
  - Modified guest
  - VMware, Xen



- 3rd Generation: Silicon-based (Hardware-assisted) virtualization
  - Unmodified guest
  - VMware and Xen on virtualization-aware hardware platforms



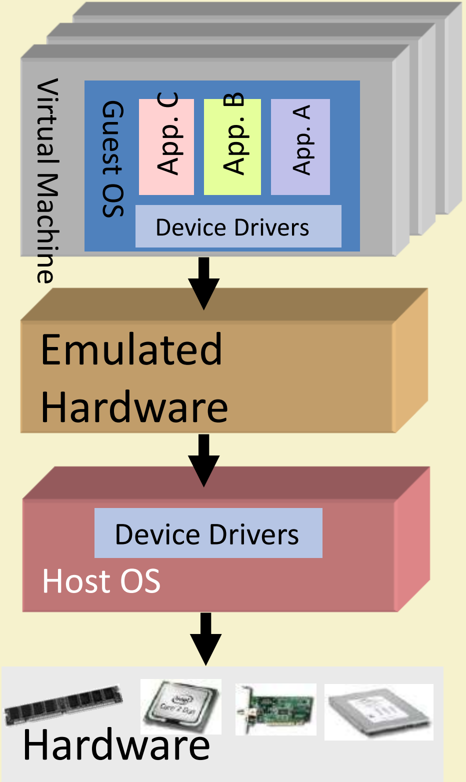
Time →

🟨 Virtualization Logic

Source: www.dc.uba.ar/events/eci/2008/courses/n2/Virtualization-Introduction.ppt

Full Virtualization

- 1st Generation offering of x86/x64 server virtualization
- Dynamic binary translation
  - Emulation layer talks to an operating system which talks to the computer hardware
  - Guest OS doesn't see that it is used in an emulated environment
- All of the hardware is emulated including the CPU
- Two popular open source emulators are QEMU and Bochs



Source: www.dc.uba.ar/events/eci/2008/courses/n2/Virtualization-Introduction.ppt

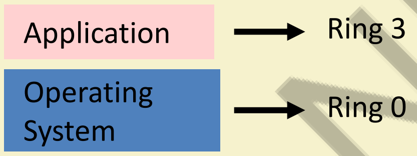
Full Virtualization - Advantages

- Emulation layer
  - Isolates VMs from the host OS and from each other
  - Controls individual VM access to system resources, preventing an unstable VM from impacting system performance
- Total VM portability
  - By emulating a consistent set of system hardware, VMs have the ability to transparently move between hosts with dissimilar hardware without any problems
    - It is possible to run an operating system that was developed for another architecture on your own architecture
    - A VM running on a Dell server can be relocated to a Hewlett-Packard server

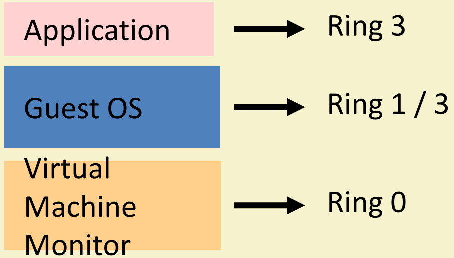
Source: www.dc.uba.ar/events/eci/2008/courses/n2/Virtualization-Introduction.ppt

Full Virtualization - Drawbacks

- Hardware emulation comes with a performance price
- In traditional x86 architectures, OS kernels expect to run privileged code in Ring 0
  - However, because Ring 0 is controlled by the host OS, VMs are forced to execute at Ring 1/3, which requires the VMM to trap and emulate instructions
- Due to these performance limitations, para-virtualization and hardware-assisted virtualization were developed



Traditional x86 Architecture



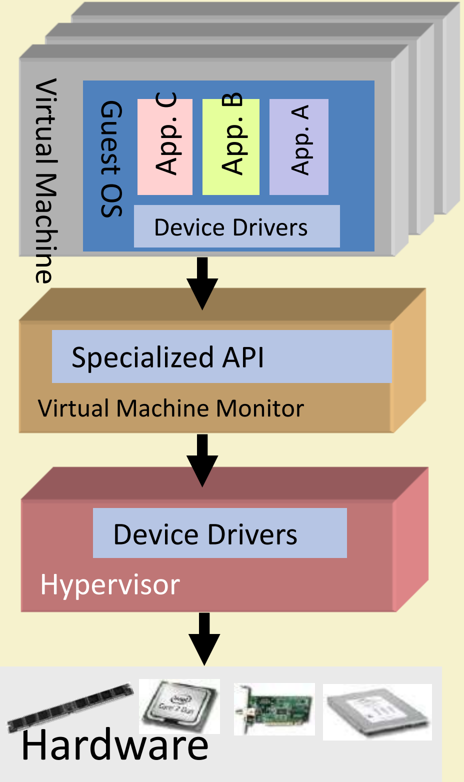
Full Virtualization

Source: www.dc.uba.ar/events/eci/2008/courses/n2/Virtualization-Introduction.ppt

Para-Virtualization

- Guest OS is modified and thus run kernel-level operations at Ring 1 (or 3)
  - Guest is fully aware of how to process privileged instructions
  - Privileged instruction translation by the VMM is no longer necessary
  - Guest operating system uses a specialized API to talk to the VMM and, in this way, execute the privileged instructions
- VMM is responsible for handling the virtualization requests and putting them to the hardware

Server virtualization approaches



Source: www.dc.uba.ar/events/eci/2008/courses/n2/Virtualization-Introduction.ppt

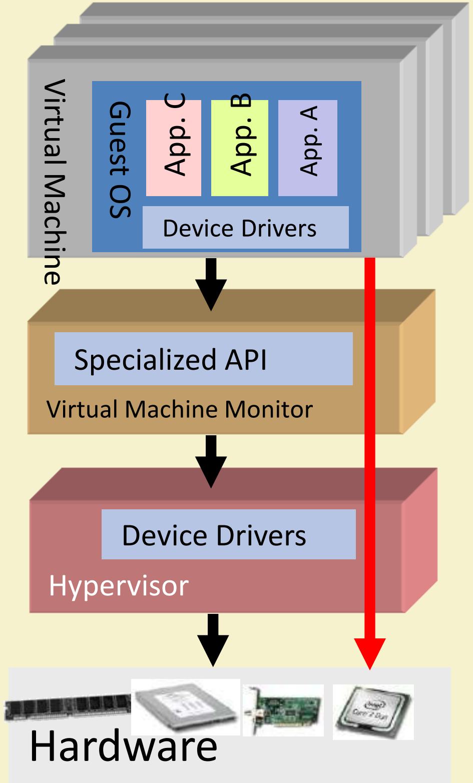
Para-Virtualization

- Today, VM guest operating systems are para-virtualized using two different approaches:
- – ***Recompiling the OS kernel***
  - Para-virtualization drivers and APIs must reside in the guest operating system kernel
  - You do need a modified operating system that includes this specific API, requiring a compiling operating systems to be virtualization aware
    - – Some vendors (such as Novell) have embraced para-virtualization and have provided para-virtualized OS builds, while other vendors (such as Microsoft) have not
- – ***Installing para-virtualized drivers***
  - In some operating systems it is not possible to use complete para-virtualization, as it requires a specialized version of the operating system
  - To ensure good performance in such environments, para-virtualization can be applied for individual devices
  - For example, the instructions generated by network boards or graphical interface cards can be modified before they leave the virtualized machine by using para-virtualized drivers

Source: www.dc.uba.ar/events/eci/2008/courses/n2/Virtualization-Introduction.ppt

Hardware-assisted virtualization

- Guest OS runs at ring 0
- VMM uses processor extensions (such as Intel®-VT or AMD-V) to intercept and emulate privileged operations in the guest
- Hardware-assisted virtualization removes many of the problems that make writing a VMM a challenge
- VMM runs in a more privileged ring than 0, a *Virtual-1* ring is created



Source: www.dc.uba.ar/events/eci/2008/courses/n2/Virtualization-Introduction.ppt

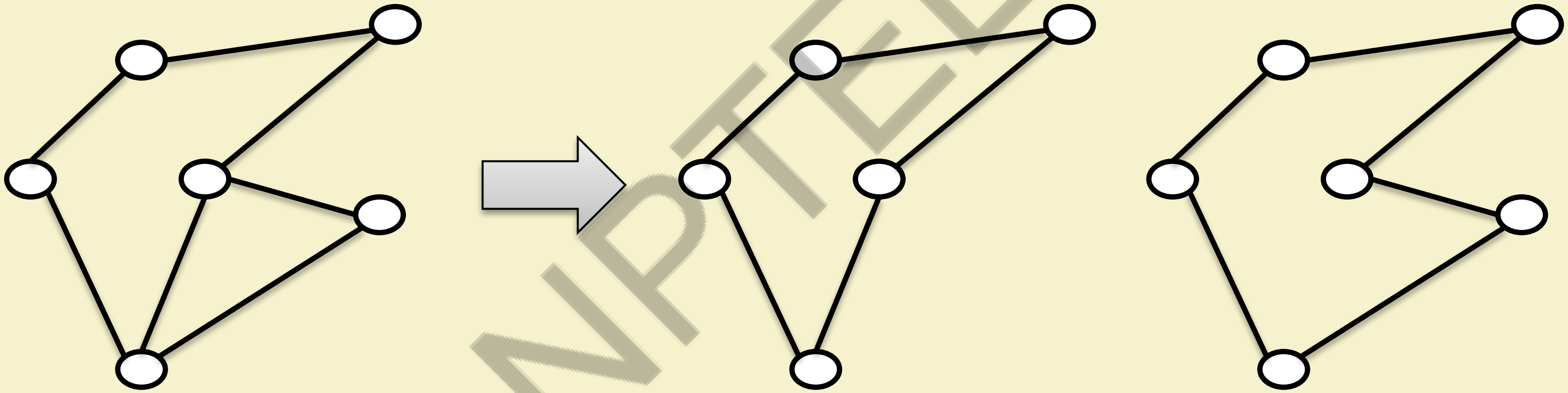
Hardware-assisted virtualization

- **Pros**
  - It allows to run unmodified OSs (so legacy OS can be run without problems)
- **Cons**
  - Speed and Flexibility
    - An unmodified OS does not know it is running in a virtualized environment and so, it can’t take advantage of any of the virtualization features
      - It can be resolved using para-virtualization partially

Source: www.dc.uba.ar/events/eci/2008/courses/n2/Virtualization-Introduction.ppt

Network Virtualization

Making a physical network appear as multiple logical ones



Why Virtualize ?

- Internet is *almost* "paralyzed"
  - Lots of makeshift solutions (e.g. overlays)
  - A new architecture (aka clean-slate) is needed
- Hard to come up with a *one-size-fits-all* architecture
  - Almost impossible to predict what future might unleash
- Why not create an *all-sizes-fit-into-one* instead!
  - Open and expandable architecture
- Testbed for future networking architectures and protocols

Related Concepts

- Virtual Private Networks (VPN)
  - Virtual network connecting distributed sites
  - Not customizable enough

- Active and Programmable Networks
  - Customized network functionalities
  - Programmable interfaces and active codes

- Overlay Networks
  - Application layer virtual networks
  - Not flexible enough

Network Virtualization Model

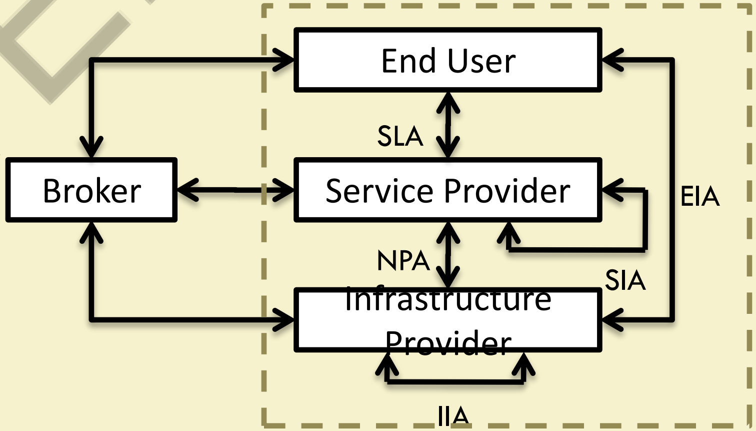
- Business Model
- Architecture
- Design Principles
- Design Goals

Business Model

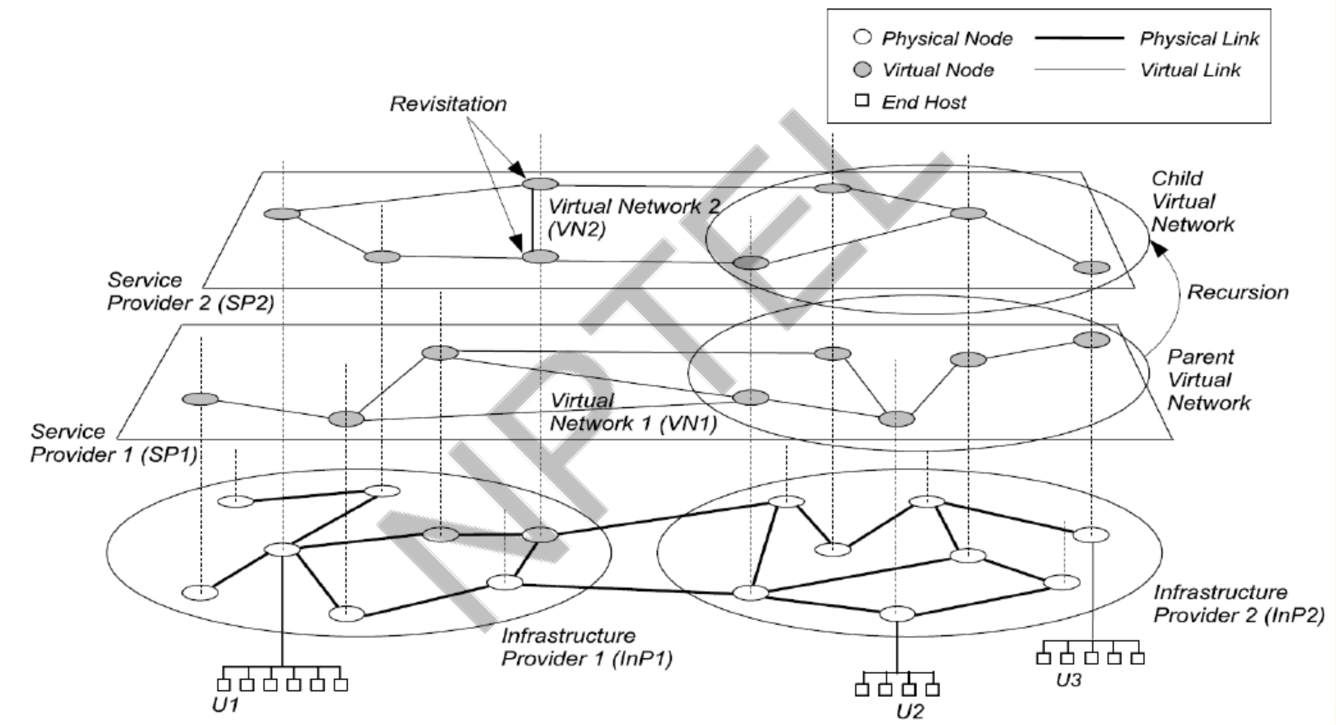
Players

- Infrastructure Providers (*InPs*)
  - Manage underlying physical networks
- Service Providers (*SPs*)
  - Create and manage virtual networks
  - Deploy customized end-to-end services
- End Users
  - Buy and use services from different service providers
- Brokers
  - Mediators/Arbiters

Relationships



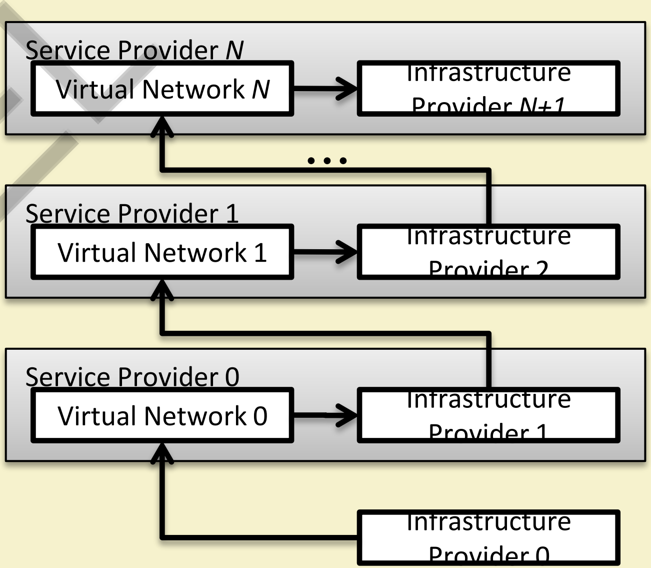
Architecture



Design Principles

- □ *Concurrence* of multiple heterogeneous virtual networks
  - Introduces diversity
- □ *Recursion* of virtual networks
  - Opens the door for network virtualization economics
- □ *Inheritance* of architectural attributes
  - Promotes value-addition
- □ *Revisitation* of virtual nodes
  - Simplifies network operation and management

Hierarchy of Roles



Design Goals (1)

- Flexibility
  - Service providers can choose
    - arbitrary network topology,
    - routing and forwarding functionalities,
    - customized control and data planes
  - No need for co-ordination with others
    - IPv6 fiasco should never happen again

- Manageability
  - Clear separation of policy from mechanism
  - Defined *accountability* of infrastructure and service providers
  - Modular management

Design Goals (2)

- Scalability
  - Maximize the number of co-existing virtual networks
  - Increase resource utilization and amortize CAPEX and OPEX
- Security, Privacy, and Isolation
  - Complete isolation between virtual networks
    - *Logical* and *resource*
  - Isolate faults, bugs, and misconfigurations
    - Secured and private

Design Goals (3)

- Programmability
  - Of network elements e.g. routers
  - Answer *"How much"* and *"how"*
  - Easy and effective without being vulnerable to threats

- Heterogeneity
  - Networking technologies
    - Optical, sensor, wireless etc.
  - Virtual networks

Design Goals (4)

- Experimental and Deployment Facility
  - PlanetLab, GENI, VINI
  - Directly deploy services in real world from the testing phase
- Legacy Support
  - Consider the existing Internet as a member of the collection of multiple virtual Internets
  - Very *important* to keep all concerned parties satisfied

Definition

Network virtualization is a networking environment that allows multiple service providers to dynamically compose multiple heterogeneous virtual networks that co-exist together in isolation from each other, and to deploy customized end-to-end services on-the-fly as well as manage them on those virtual networks for the end-users by effectively sharing and utilizing underlying network resources leased from multiple infrastructure providers.

Typical Approach

- Networking technology
  - IP, ATM
- Layer of virtualization

- Architectural domain
  - Network resource management, Spawning networks
- Level of virtualization
  - Node virtualization, Full virtualization

Thank You!

Introduction to XML:
eXtensible Markup Language

Prof. Soumya K Ghosh
Department of Computer Science and Engineering
IIT KHARAGPUR

XML ??

- Over time, the acronym “XML” has evolved to imply a growing family of software tools/XML standards/ideas around
  - How XML data can be represented and processed
  - application frameworks (tools, dialects) based on XML

- Most “popular” XML discussion refers to this latter meaning

- We’ll talk about both.

Presentation Outline

- What is XML (basic introduction)
  - Language rules, basic XML processing
- Defining language dialects
  - DTDs, schemas, and namespaces
- XML processing
  - Parsers and parser interfaces
  - XML-based processing tools
- XML messaging
  - Why, and some issues/example
- Conclusions

What is XML?

- ***A syntax*** for "encoding" text-based data (words, phrases, numbers, ...)
- ***A text-based syntax.*** XML is written using ***printable Unicode*** characters (no explicit binary data; character encoding issues)
- ***Extensible***. XML lets you define your own ***elements*** (essentially ***data types***), within the constraints of the syntax rules
- ***Universal format***. The syntax rules ensure that all XML processing software ***MUST*** identically handle a given piece of XML data.

If you can read and process it, so can anybody else

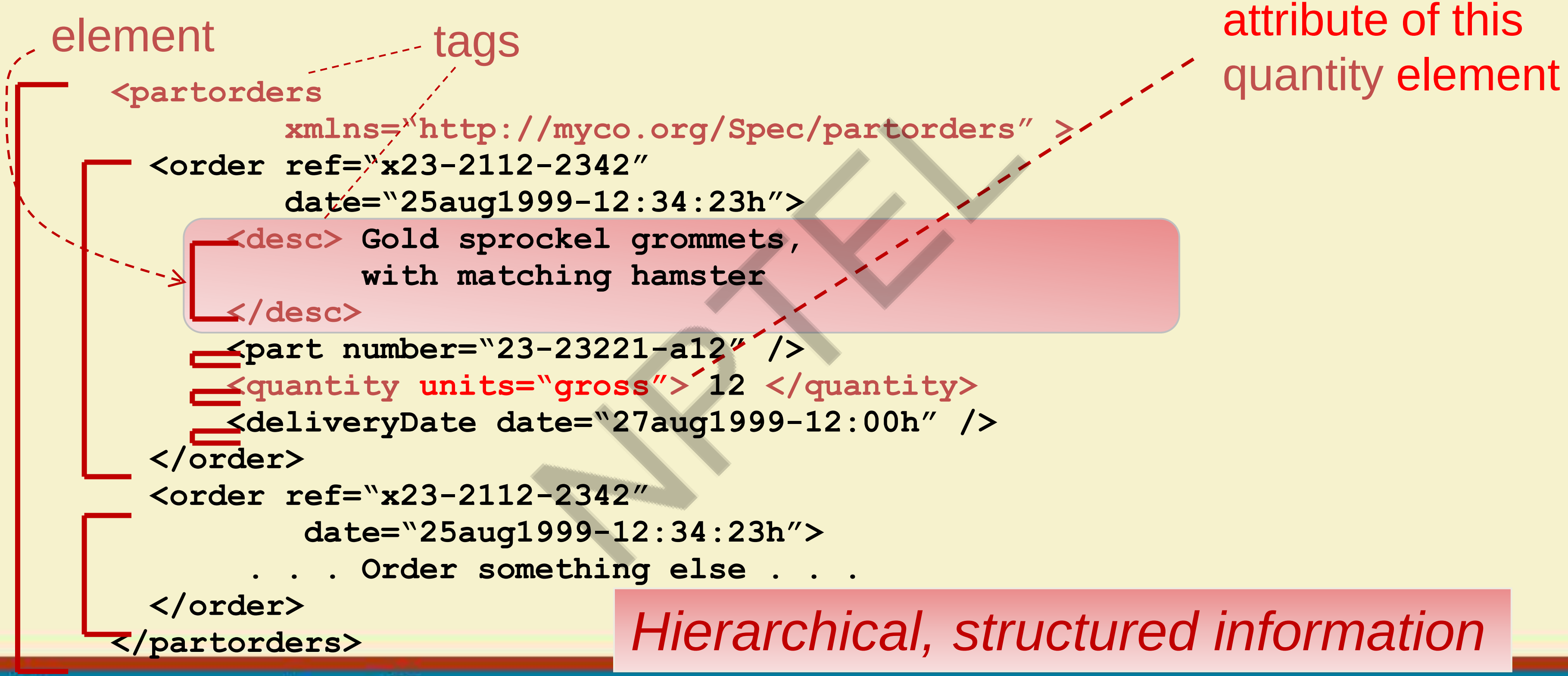
What is XML: A Simple Example

XML Declaration ("this is XML")

Binary encoding used in file

```
<?xml version="1.0" encoding="iso-8859-1"?>
<partorders
    xmlns="http://myco.org/Spec/partorders">
 <order ref="x23-2112-2342"
      date="25aug1999-12:34:23h">
    <desc> Gold sprockel grommets,
         with matching hamster
    </desc>
    <part number="23-23221-a12" />
    <quantity units="gross"> 12 </quantity>
    <deliveryDate date="27aug1999-12:00h" />
 </order>
 <order ref="x23-2112-2342"
       date="25aug1999-12:34:23h">
    . . . Order something else . . .
 </order>
</partorders>
```

Example Revisited



element

tags

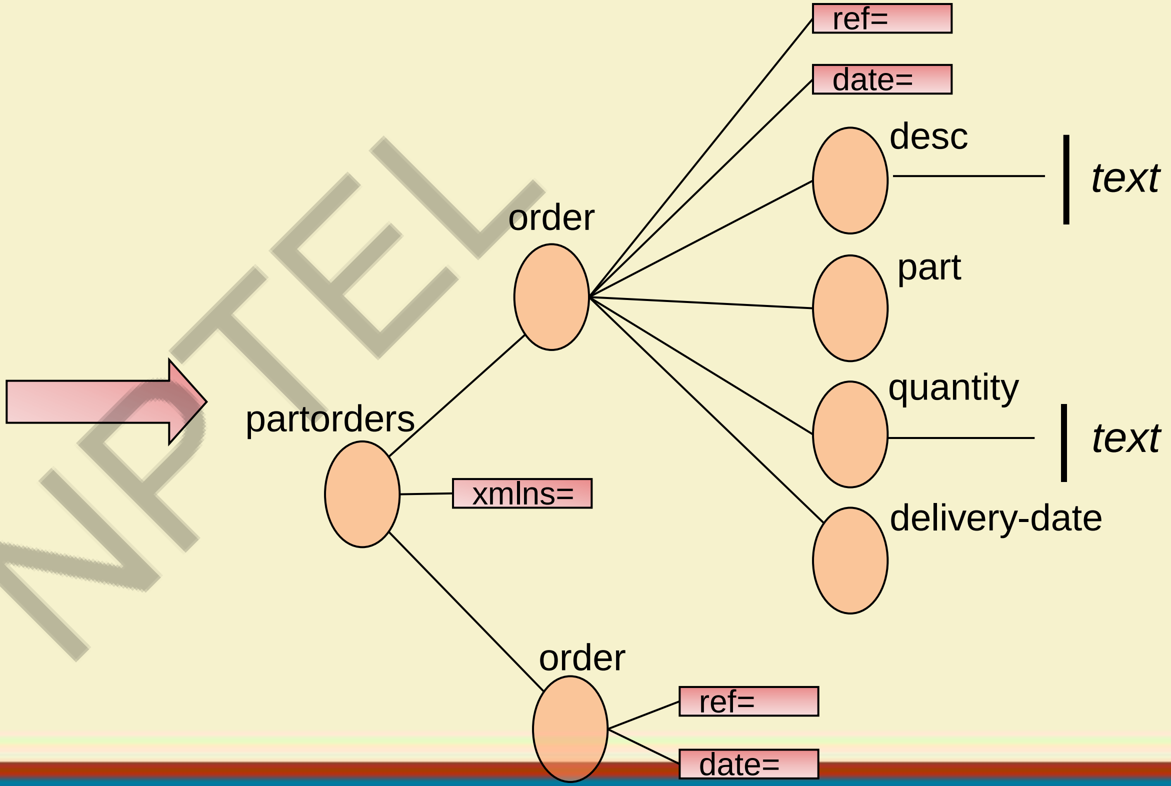
attribute of this quantity element

```
<partorders
         xmlns=“http://myco.org/Spec/partorders” >
  <order ref=“x23-2112-2342”
         date=“25aug1999-12:34:23h”>
      <desc> Gold sprockel grommets,
             with matching hamster
      </desc>
      <part number=“23-23221-a12” />
      <quantity units=“gross”> 12 </quantity>
      <deliveryDate date=“27aug1999-12:00h” />
  </order>
  <order ref=“x23-2112-2342”
          date=“25aug1999-12:34:23h”>
       . . . Order something else . . .
  </order>
</partorders>
```

Hierarchical, structured information

XML Data Model - A Tree

```
<partorders xmlns="...">
   <order date="..."
          ref="...">
     <desc> ..text..
     </desc>
     <part />
     <quantity />
     <delivery-date />
   </order>
   <order ref=".." .../>
</partorders>
```



XML: Why it's this way

- ***Simple*** (like HTML -- but not quite so simple)
  - Strict ***syntax*** rules, to eliminate syntax errors
  - syntax ***defines*** structure (hierarchically), and ***names*** structural parts (element names) -- it is ***self-describing data***

- ***Extensible*** (unlike HTML; vocabulary is not fixed)
  - Can create your own ***language*** of tags/elements
  - ***Strict syntax*** ensures that such markup can be reliably processed

- Designed for a ***distributed environment*** (like HTML)
  - Can have data all over the place: can retrieve and use it reliably

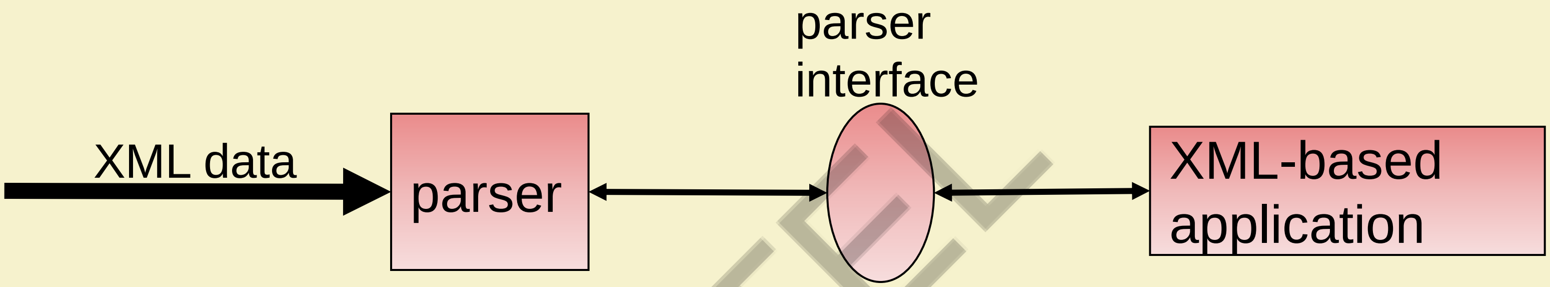
- Can ***mix*** different data types together (unlike HTML)
  - Can mix one set of tags with another set: resulting data can still be reliably processed

XML Processing

```xml
<?xml version="1.0" encoding="utf-8" ?>
<transfers>
   <fundsTransfer date="20010923T12:34:34Z">
     <from type="intrabank">
        <amount currency="USD"> 1332.32 </amount>
        <transitID> 3211 </transitID>
        <accountID> 4321332 </accountID>
        <acknowledgeReceipt> yes </acknowledgeReceipt>
     </from>
     <to account="132212412321" />
   </fundsTransfer>
   <fundsTransfer date="20010923T12:35:12Z">
     <from type="internal">
        <amount currency="CDN" >1432.12 </amount>
        <accountID> 543211 </accountID>
        <acknowledgeReceipt> yes </acknowledgeReceipt>
     </from>
     <to account="65123222" />
   </fundsTransfer>
</transfers>
```

xml-simple.xml

XML Parser Processing Model



- The parser must verify that the XML data is syntactically correct.
- Such data is said to be ***well-formed***
  - The minimal requirement to “be” XML
- A parser **MUST** stop processing if the data isn’t well-formed
  - E.g., stop processing and “throw an exception” to the XML-based application. The XML 1.0 spec ***requires*** this behaviour

XML Processing Rules: Including Parts

```
<?xml version="1.0" encoding="utf-8" ?>

<!DOCTYPE transfers  [
   <!-- Here is an internal entity that encodes a bunch of
        markup that we'd otherwise use in a document  -->

  <!ENTITY messageHeader
     "<header>
        <routeID> info generic to message route </routeID>
        <encoding>how message is encoded </encoding>
      </header> "
  >
]>
<transfers>
   &messageHeader;
   <fundsTransfer date="20010923T12:34:34Z">
     <from type="intrabank">
        . . . Content omitted . . . .
</transfers>
```

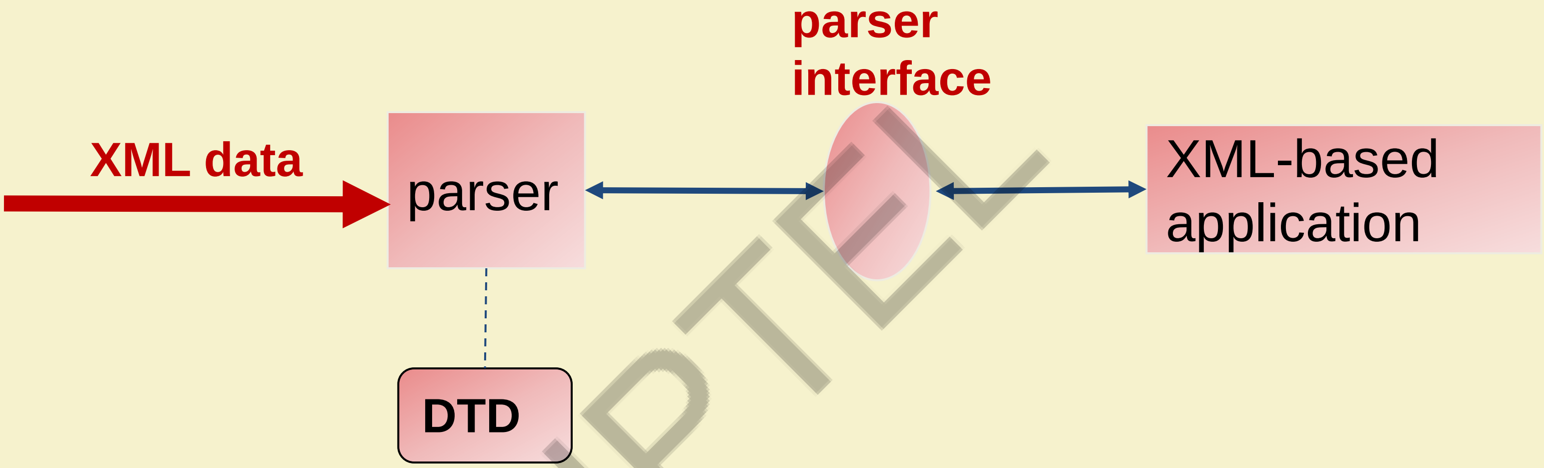
Document Type Declaration (DTD)

Internal Entity declaration

Entity reference
&name;

xml-simple-intEntity.xml

XML Parser Processing Model



XML Parsers, DTDs, and Internal Entities

- The parser processes the DTD content, identifies the internal entities, and checks that each entity is well-formed.

- There are explicit syntax rules for DTD content -- well-formed XML must be correct here also.

- The parser then replaces every occurrence of an ***entity reference*** by the referenced entity (and does so recursively within entities)

- The “resolved” data object is then made available to the XML application

XML Processing Rules: External Entities

Put the entity in another file -- so it can be shared by multiple resources.

```
<?xml version="1.0" encoding="utf-8" ?>

<!DOCTYPE transfers  [
      . . .

  <!ENTITY messageHeader
       SYSTEM "http://www.somewhere.org/dir/head.xml"
  >

]>
<transfers>
   &messageHeader;
   <fundsTransfer date="20010923T12:34:34Z">
     <from type="intrabank">
        . . . Content omitted . . .   .
</transfers>
```

External Entity declaration

***Location** given via a URL*

xml-simple-extEntity.xml

XML Parsers and External Entities

- The parser processes the DTD content, identifies the external entities, and “tries” to resolve them

- The parser then replaces every occurrence of an ***entity reference*** by the referenced entity, and does so recursively within all those entities, (like with internal entities)

- But …. what if the parser can’t find the external entity (firewall?)?

- That depends on the application / parser type
  - There are ***two types of XML parsers***
  - one that MUST retrieve all entities, and one that can ignore them (if it can’t find them)

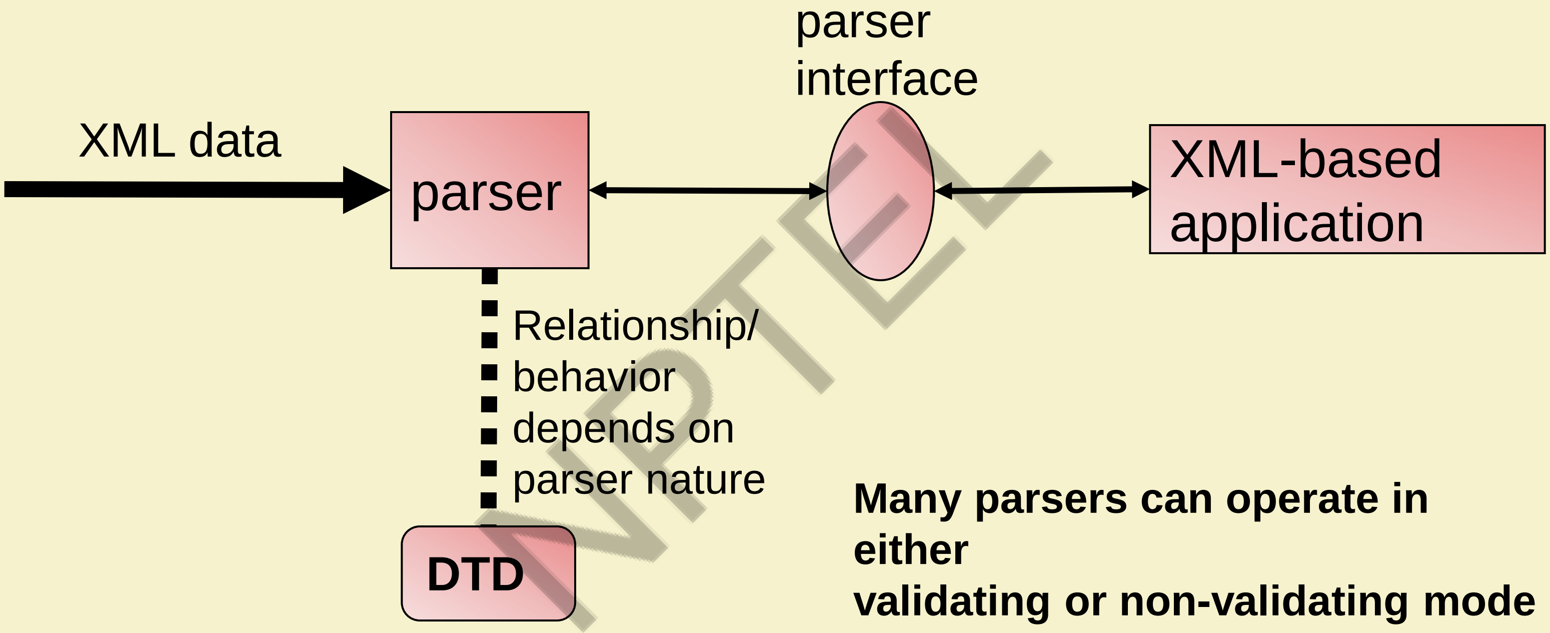
Two types of XML parsers

- **Validating parser**
  - ***Must*** retrieve all entities and must process ***all*** DTD content. Will stop processing and indicate a failure if it cannot
  - There is also the implication that it will test for compatibility with other things in the DTD -- instructions that define syntactic rules for the document (allowed elements, attributes, etc.). We'll talk about these parts in the next section.

- **Non-validating parser**
  - Will try to retrieve all entities defined in the DTD, but will ***cease processing the DTD*** content at the first entity it can't find, But this is not an error -- the parser simply makes available the XML data (and the names of any unresolved entities) to the application.

Application behavior will depend on ***parser type***

XML Parser Processing Model



Many parsers can operate in either
validating or non-validating mode (parameter-dependent)

Special Issues: Characters and Charsets

- XML specification defines what characters can be used as whitespace in tags: `<element  id  =  "23.112"  />`

- ***You cannot use EBCIDIC character 'NEL' as whitespace***
  - Must make sure to not do so!

- What if you want to include characters not defined in the encoding charset (e.g., Greek characters in an ISO-Latin-1 document):

- Use ***character references***. For example:

  &#9824; -- the spades character (♠)

  9824th character in the Unicode character set

- Also, binary data must be encoded as ***printable characters***

Presentation Outline

- What is XML (basic introduction)
  - Language rules, basic XML processing
- Defining language dialects
  - DTDs, schemas, and namespaces
- XML processing
  - Parsers and parser interfaces
  - XML-based processing tools
- XML messaging
  - Why, and some issues/example
- Conclusions

How do you define language dialects?

- Two ways of doing so:
  - **XML Document Type Declaration (DTD)** -- Part of core XML spec.
  - **XML Schema** -- New XML specification (2001), which allows for stronger constraints on XML documents.

- Adding dialect specifications implies ***two classes*** of XML data:
  - ***Well-formed*** An XML document that is syntactically correct
  - ***Valid*** An XML document that is both well-formed *and* consistent with a specific DTD (or Schema)

- What DTDs and/or schema specify:
  - Allowed element and attribute names, hierarchical nesting rules; element content/type restrictions

- Schemas are more powerful than DTDs. They are often used for ***type validation***, or for relating database schemas to XML models

Example DTD (as part of document)

xml-simple-valid.xml

```
<!DOCTYPE transfers  [
  <!ELEMENT  transfers  (fundsTransfer)+ >
  <!ELEMENT  fundsTransfer  (from, to)  >
  <!ATTLIST fundsTransfer
         date  CDATA  #REQUIRED>
  <!ELEMENT  from  (amount, transitID?, accountID,
                   acknowledgeReceipt ) >
  <!ATTLIST from
         type  (intrabank|internal|other)  #REQUIRED>
  <!ELEMENT  amount  (#PCDATA)  >
      . . . Omitted DTD content . . . .
  <!ELEMENT  to  EMPTY  >
  <!ATTLIST to
         account  CDATA  #REQUIRED>
]>
<transfers>
   <fundsTransfer date="20010923T12:34:34Z">
      . . . As with previous example . . .
```

Example “External” DTD

- Reference is using a variation on the DOCTYPE:

simple.dtd

```
<!DOCTYPE transfers  SYSTEM
     "http://www.foo.org/hereitis/simple.dtd”  >

<transfers>
   <fundsTransfer date="20010923T12:34:34Z">
      . . . As with previous example . . .

      . . .

</transfers>
```

- Of course, the DTD file must be there, and accessible.

Thank You!!

Introduction to XML:
eXtensible Markup Language

Prof. Soumya K Ghosh
Department of Computer Science and Engineering
IIT KHARAGPUR

XML Schemas

- A new specification (2001) for specifying validation rules for XML

Specs:	http://www.w3.org/XML/Schema
Best-practice:	http://www.xfront.com/BestPracticesHomepage.html

- Uses ***pure XML*** (no special DTD grammar) to do this.
- Schemas are more powerful than DTDs - can specify things like integer types, date strings, real numbers in a given range, etc.
- They are often used for ***type validation***, or for relating database schemas to XML models
- They don’t, however, let you declare entities -- those can only be done in DTDs.
- The following slide shows the XML schema equivalent to our DTD

XML Schema version of our DTD (Portion)

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
           elementFormDefault="qualified">
   <xs:element name="accountID" type="xs:string"/>
   <xs:element name="acknowledgeReceipt" type="xs:string"/>
   <xs:complexType name="amountType">
      <xs:simpleContent>
         <xs:restriction base="xs:string">
            <xs:attribute name="currency" use="required">
               <xs:simpleType>
                  <xs:restriction base="xs:NMTOKEN">
                     <xs:enumeration value="USD"/>
                       . . . (some stuff omitted) . . .
                  </xs:restriction>
               </xs:simpleType>
            </xs:attribute>
         </xs:restriction>
      </xs:simpleContent>
   </xs:complexType>
   <xs:complexType name="fromType">
      <xs:sequence>
         <xs:element name="amount" type="amountType"/>
         <xs:element ref="transitID" minOccurs="0"/>
         <xs:element ref="accountID"/>
         <xs:element ref="acknowledgeReceipt"/>
      </xs:sequence>
       . . .
```

simple.xsd

XML Namespaces

- Mechanism for identifying different “spaces” for XML names
  - That is, ***element*** or ***attribute*** names
- This is a way of identifying different ***language dialects***, consisting of names that have specific semantic (and processing) meanings.
- Thus `<key/>` in one language (might mean a security key) can be distinguished from `<key/>` in another language (a database key)
- Mechanism uses a special **`xmlns`** attribute to define the namespace. The namespace is given as a ***URL string***
  - But the URL does not reference anything in particular (there may be nothing there)

Mixing language dialects together

Namespaces let you do this relatively easily:

```
<?xml version= "1.0" encoding= "utf-8" ?>

<html xmlns="http://www.w3.org/1999/xhtml1"
      xmlns:mt="http://www.w3.org/1998/mathml” >
<head>
   <title> Title of XHTML Document </title>
</head><body>
<div class="myDiv">
    <h1> Heading of Page </h1>
     <mt:mathml>
           <mt:title> ... MathML markup . . .
     </mt:mathml>
     <p> more html stuff goes here </p>
</div>
</body>
</html>
```

Default ‘space’ is *xhtml*

***mt:** prefix indicates ‘space’ mathml (a different language)*

Presentation Outline

- What is XML (basic introduction)
  - Language rules, basic XML processing
- Defining language dialects
  - DTDs, schemas, and namespaces
- XML processing
  - Parsers and parser interfaces
  - XML-based processing tools
- XML messaging
  - Why, and some issues/example
- Conclusions

XML Software

- **XML parser** -- Reads in XML data, checks for syntactic (and possibly DTD/Schema) constraints, and makes data available to an application. There are three 'generic' parser APIs

– SAX	Simple API to XML (event-based)
– DOM	Document Object Model (object/tree based)
– JDOM	Java Document Object Model (object/tree based)

- Lots of XML parsers and interface software available (Unix, Windows, OS/390 or Z/OS, etc.)

- SAX-based parsers are fast (often as fast as you can stream data)

- DOM slower, more memory intensive (create in-memory version of entire document)

- And, validating can be ***much slower*** than non-validating

XML Processing: SAX

A) SAX: Simple API for XML

- http://www.megginson.com/SAX/index.html
- An ***event-based*** interface
- Parser reports events whenever it sees a tag/attribute/text node/unresolved external entity/other
- Programmer attaches “event handlers” to handle the event

- **Advantages**
  - Simple to use
  - Very fast (not doing very much before you get the tags and data)
  - Low memory footprint (doesn’t read an XML document entirely into memory)

- **Disadvantages**
  - Not doing very much for you -- you have to do everything yourself
  - Not useful if you have to dynamically modify the document once it’s in memory (since you’ll have to do all the work to put it in memory yourself!)

XML Processing: DOM

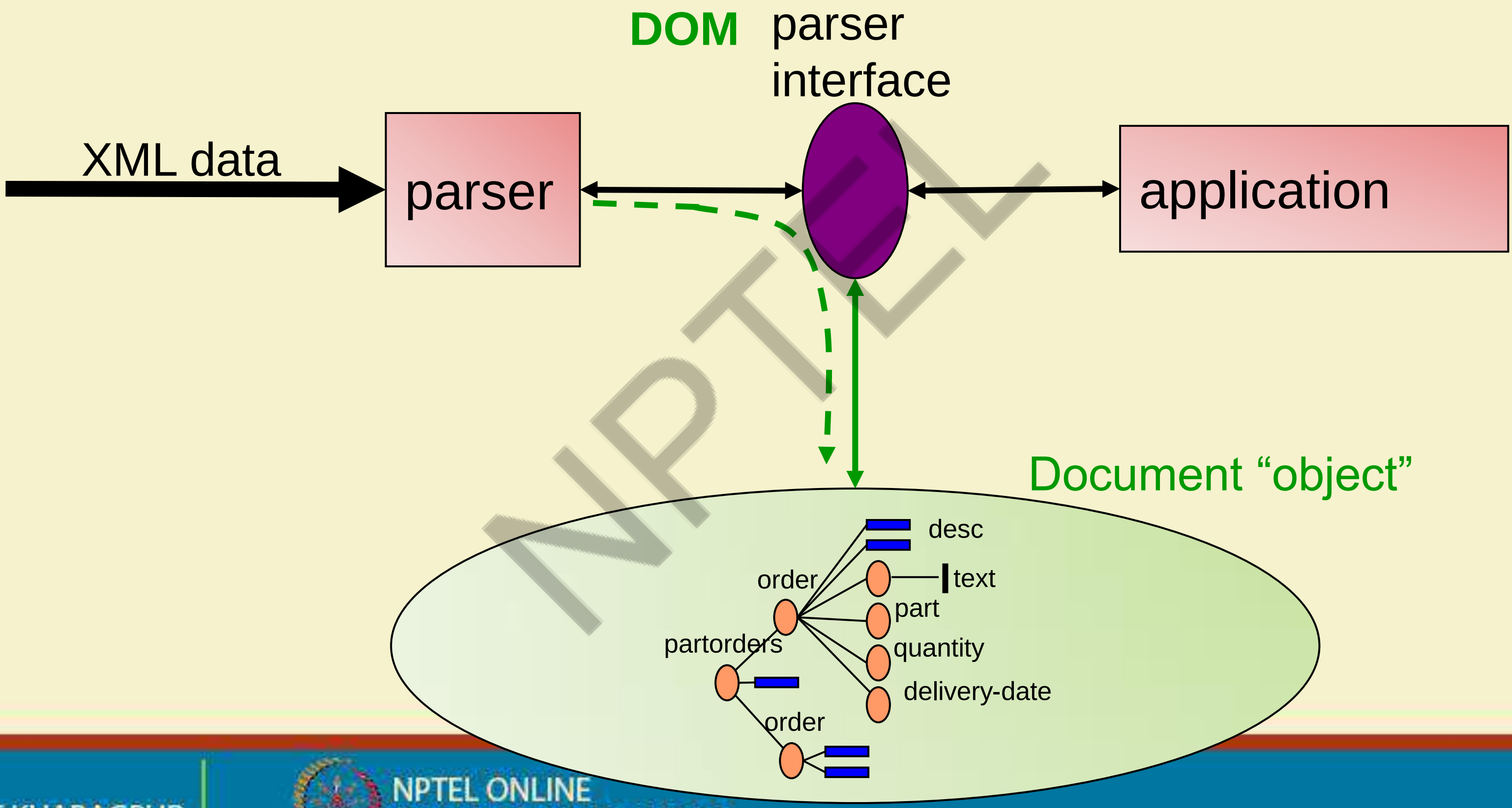
B) DOM: Document Object Model

- http://www.w3.org/DOM/
- An ***object-based*** interface
- Parser generates an ***in-memory tree*** corresponding to the document
- DOM interface defines methods for accessing and modifying the tree

- **Advantages**
  - Very useful for dynamic modification of, access to the tree
  - Useful for querying (I.e. looking for data) that depends on the tree structure [element.childNode("2").getAttributeValue("boobie")]
  - Same interface for many programming languages (C++, Java, ...)

- **Disadvantages**
  - Can be slow (needs to produce the tree), and may need lots of memory
  - DOM programming interface is a bit awkward, not terribly object oriented

DOM Parser Processing Model



XML Processing: JDOM

C) JDOM: *Java* Document Object Model

- http://www.jdom.org
- A Java-specific ***object-oriented*** interface
- Parser generates an in-memory tree corresponding to the document
- JDOM interface has methods for accessing and modifying the tree

- **Advantages**
  - Very useful for dynamic modification of the tree
  - Useful for querying (I.e. looking for data) that depends on the tree structure
  - Much nicer Object Oriented programming interface than DOM
- **Disadvantages**
  - Can be slow (make that tree...), and can take up lots of memory
  - New, and not entirely cooked (but close)
  - Only works with Java, and not (yet) part of Core Java standard

XML Processing: dom4j

C) dom4j: XML framework for Java

- http://www.dom4j.org
- Java framework for reading, writing, navigating and editing XML.
- Provides access to SAX, DOM, JDOM interfaces, and other XML utilities (XSLT, JAXP, …)
- Can do “mixed” SAX/DOM parsing -- use SAX to one point in a document, then turn rest into a DOM tree.

- **Advantages**
  - Lots of goodies, all rolled into one easy-to-use Java package
  - Can do “mixed” SAX/DOM parsing -- use SAX to one point in a document, then turn rest into a DOM tree
  - Apache open source license means free use (and IBM likes it!)
- **Disadvantages**
  - Java only; may be concerns over open source nature (but IBM uses it, so it can’t be that bad!)

Some XML Parsers (OS/390's)

- **Xerces (C++; Apache Open Source)**
  http://xml.apache.org/xerces-c/index.html
- **XML toolkit (Java and C+++; Commercial license)**
  http://www-1.ibm.com/servers/eserver/zseries/software/xml/
  I believe the Java version uses XML4j, IBM's Java Parser. The latest version is always found at:
  http://www.alphaworks.ibm.com
- **XML for C++ (IBM; based on Xerces; Commercial license)**
  http://www.alphaworks.ibm.com/tech/xml4c
- **XMLBooster (parsers for COBOL, C++ …; Commercial license; don't know much about it; OS/390? [dunno])**
  http://www.xmlbooster.com/
  Has free trial download,: can see if it is any good ;-)
- **XML4Cobol (don't know much about it, any COBOL85 is fine)**
  http://www.xml4cobol.com

- www.xmlsoftware.com/parsers/ -- Good generic list of parsers

Some parser benchmarks:

- http://www-106.ibm.com/developerworks/xml/library/x-injava/index.html (Sept 2001)
- http://www.devsphere.com/xml/benchmark/index.html (Java) (late-2000)

- **Basically**

– SAX faster	xDOM slower
– SAX less memory	xDOM more memory
– SAX stream processing	xDOM object / persistence processing

  - nonvalidating is always faster than validating!

XML Processing: XSLT

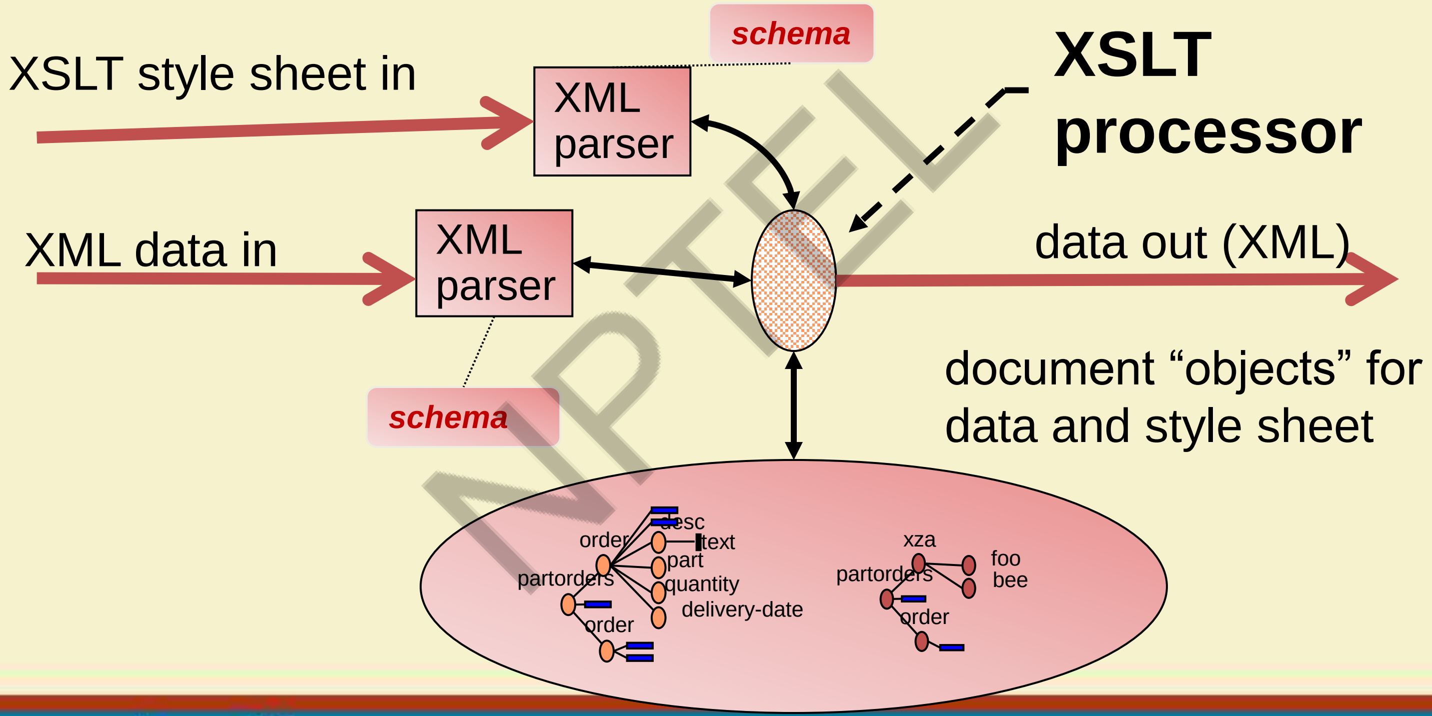
D) XSLT eXtensible Stylesheet Language -- *Transformations*

- http://www.w3.org/TR/xslt
- An XML language for processing XML
- Does tree transformations -- takes XML and an XSLT style sheet as input, and produces a new XML document with a different structure

- **Advantages**
  - Very useful for tree transformations -- much easier than DOM or SAX for this purpose
  - Can be used to query a document (XSLT pulls out the part you want)
- **Disadvantages**
  - Can be slow for large documents or stylesheets
  - Can be difficult to debug stylesheets (poor error detection; much better if you use schemas)

XSLT processing model

- **D)** XSLT Processing model



Presentation Outline

- What is XML (basic introduction)
  - Language rules, basic XML processing
- Defining language dialects
  - DTDs, schemas, and namespaces
- XML processing
  - Parsers and parser interfaces
  - XML-based processing tools
- XML messaging
  - Why, and some issues/example

XML Messaging

- Use XML as the format for sending messages between systems
- Advantages are:
  - Common syntax; self-describing (easier to parse)
  - Can use common/existing transport mechanisms to “move” the XML data (HTTP, HTTPS, SMTP (email), MQ, IIOP/(CORBA), JMS, ….)
- Requirements
  - Shared understanding of dialects for transport (required registry [namespace!]) for identifying dialects
  - Shared acceptance of ***messaging contract***
- Disadvantages
  - Asynchronous transport; no guarantee of delivery, no guarantee that partner (external) shares acceptance of contract.
  - Messages will be much larger than binary (10x or more) [can compress]

Common messaging model

- XML over HTTP
  - Use HTTP to transport XML messages
  -

```
POST /path/to/interface.pl  HTTP/1.1
Referer:  http://www.foo.org/myClient.html
User-agent: db-server-olk
Accept-encoding: gzip
Accept-charset: iso-8859-1, utf-8, ucs
Content-type: application/xml; charset=utf-8
Content-length: 13221
. . .

<?xml version=“1.0” encoding=“utf-8” ?>
<message>  . . . Markup in message . . .
</message>
```

Some standards for message format

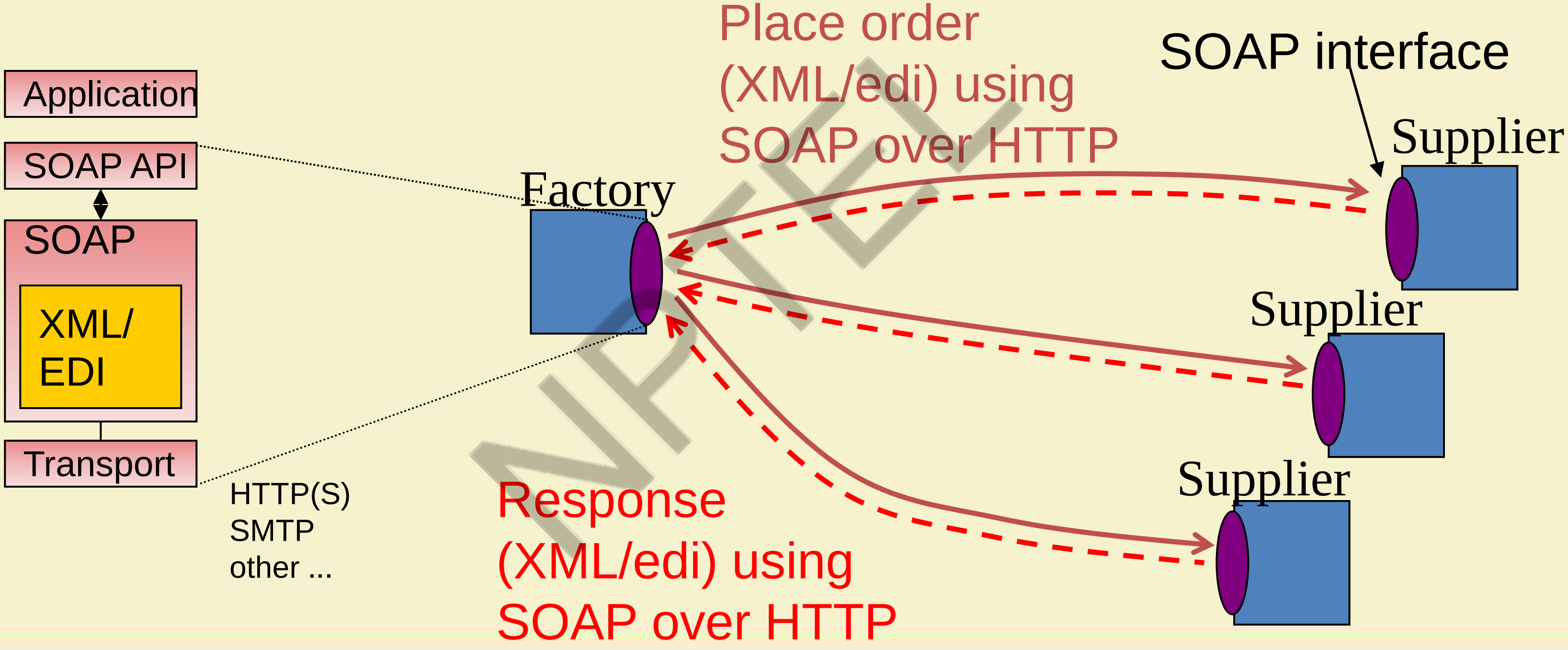
- Define dialects designed to "wrap" remote invocation messages

- **XML-RPC** http://www.xmlrpc.com
  - Very simple way of encoding function/method call name, and passed parameters, in an XML message.

- **SOAP** (Simple object access protocol) http://www.soapware.org
  - More complex wrapper, which lets you specify schemas for interfaces; more complex rules for handling/proxying messages, etc. This is a core component of Microsoft's .NET strategy, and is integrated into more recent versions of Websphere and other commercial packages.

XML Messaging + Processing

- XML as a ***universal format*** for data exchange

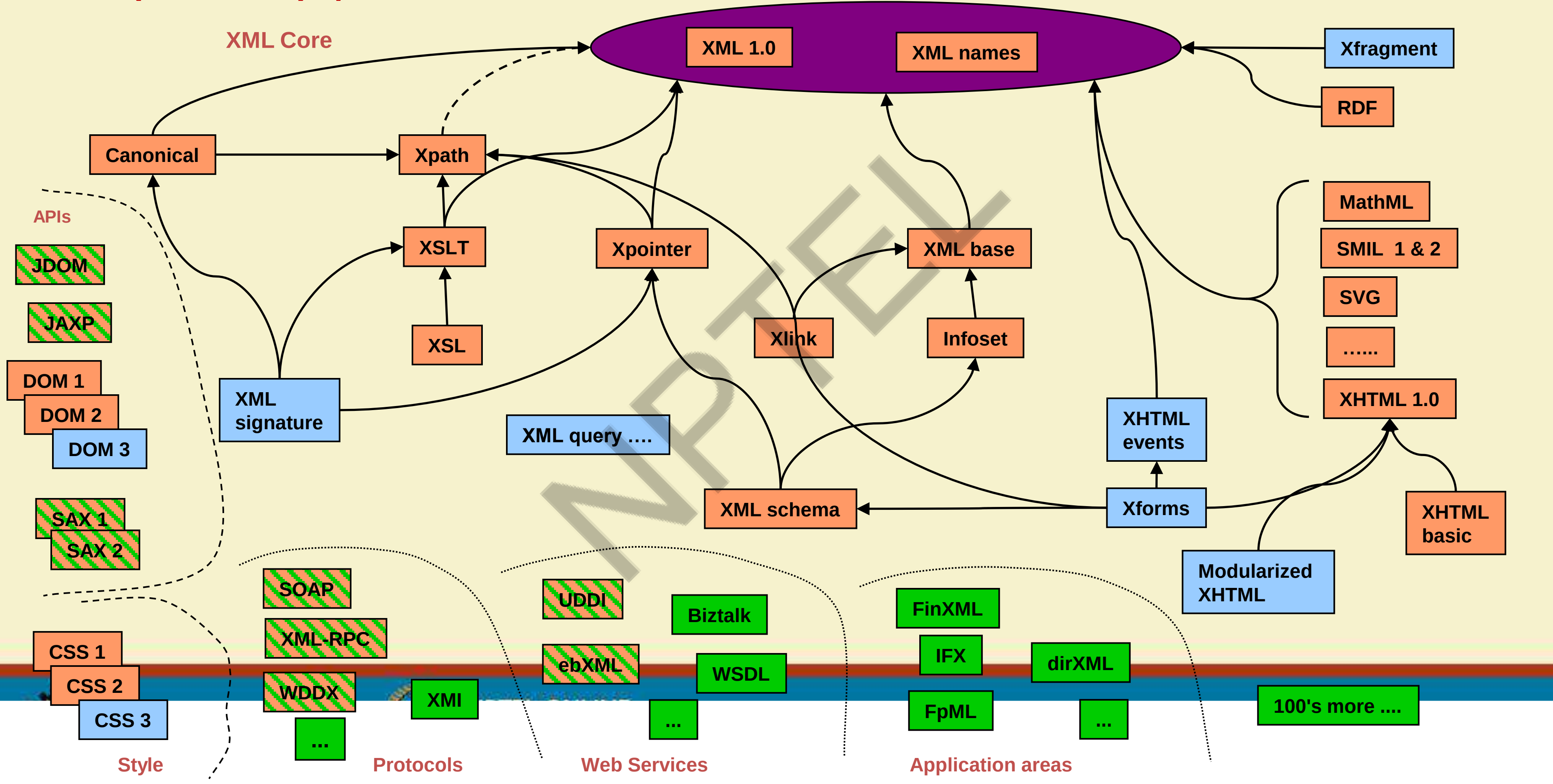


Presentation Outline

- What is XML (basic introduction)
  - Language rules, basic XML processing
- Defining language dialects
  - DTDs, schemas, and namespaces
- XML processing
  - Parsers and parser interfaces
  - XML-based processing tools
- XML messaging
  - Why, and some issues/example
- Conclusions

W3C rec
industry std
W3C draft
'Open' std

XML (and related) Specifications



XML Core

XML 1.0

XML names

Xfragment

RDF

Canonical

Xpath

APIs

JDOM

JAXP

DOM 1

DOM 2

DOM 3

SAX 1

SAX 2

XSLT

XSL

Xpointer

XML signature

Xlink

XML base

Infoset

MathML

SMIL 1 & 2

SVG

……

XHTML 1.0

XHTML events

XML query ….

XML schema

Xforms

XHTML basic

Modularized XHTML

CSS 1

CSS 2

CSS 3

Style

SOAP

XML-RPC

WDDX

XMI

…

Protocols

UDDI

ebXML

Biztalk

WSDL

…

Web Services

FinXML

IFX

FpML

dirXML

…

100's more ….

Application areas

Thank You!!

IIT KHARAGPUR

NPTEL ONLINE
CERTIFICATION COURSES

NPTEL

CLOUD COMPUTING

Web Services, Service Oriented Architecture

PROF. SOUMYA K. GHOSH

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

IIT KHARAGPUR

What are “Web Services”?

“Software application identified by a URI, whose interfaces and bindings are capable of being defined, described, and discovered as XML artifacts” – W3C Web Services Architecture Requirements, Oct. 2002

“Programmable application logic accessible using Standard Internet Protocols…” – Microsoft

“An interface that describes a collection of operations that are network accessible through standardized XML messaging …” – IBM

“Software components that can be spontaneously discovered, combined, and recombined to provide a solution to the user’s problem/request … ” - SUN

History!

- Structured programming
- Object-oriented programming
- Distributed computing
- Electronic Data Interchange (EDI)
- World Wide Web
- Web Services

Distributed Computing

- When developers create substantial applications, often it is more efficient, or even necessary, for different task to be performed on different computers, called N-tier applications:
  - A 3-tier application might have a user interface on one computer, business-logic processing on a second and a database on a third – all interacting as the application runs.
- For distributed applications to function correctly, application components, e.g. programming objects, executing on different computers throughout a network must be able to communicate.

  E.g.: DCE, CORBA, DCOM, RMI etc.
- Interoperability:
  - Ability to communicate and share data with software from different vendors and platforms
  - Limited among conventional proprietary distributed computing technologies

Electronic Data Interchange (EDI)

- Computer-to-computer exchange of business data and documents between companies using standard formats recognized both nationally and internationally.
- The information used in EDI is organized according to a specified format set by both companies participating in the data exchange.
- Advantages:
  - Lower operating costs
    - Saves time and money
  - Less Errors => More Accuracy
    - No data entry, so less human error
  - Increased Productivity
    - More efficient personnel and faster throughput
  - Faster trading cycle
    - Streamlined processes for improved trading relationships

Web Services

- Take advantage of OOP by enabling developers to build applications from existing software components in a modular approach:
    - Transform a network (e.g. the Internet) into one library of programmatic components available to developers to have significant productivity gains.
- Improve distributed computing interoperability by using open (non-proprietary) standards that can enable (theoretically) any two software components to communicate:
    - Also they are easier to debug because they are text-based, rather than binary, communication protocols

Web Services *(contd...)*

- Provide capabilities similar to those of EDI (Electronic Data Interchange), but are simpler and less expensive to implement.
- Configured to work with EDI systems, allowing organisations to use the two technologies together or to phase out EDI while adopting Web services.
- Unlike WWW
  - Separates visual from non-visual components
  - Interactions may be either through the browser or through a desktop client (Java Swing, Python, Windows, etc.)

Web Services *(contd…)*

- Intended to solve *three* problems:
  - **Interoperability**:
    - Lack of interoperability standards in distributed object messaging
    - DCOM apps strictly bound to Windows Operating system
    - RMI bound to Java programming language
  - **Firewall traversal**:
    - CORBA and DCOM used non-standard ports
    - Web Services use HTTP; most firewalls allow access though port 80 (HTTP), leading to easier and dynamic collaboration
  - **Complexity**:
    - Web Services: developer-friendly service system
    - Use open, text-based standards, which allow components written in different languages and for different platforms to communicate
    - Implemented incrementally, rather than all at once which lessens the cost and reduces the organisational disruption from an abrupt switch in technologies

Web Service: Definition Revisited

- An application component that:
  - Communicates via open protocols (HTTP, SMTP, etc.)
  - Processes XML messages framed using SOAP
  - Describes its messages using XML Schema
  - Provides an endpoint description using WSDL
  - Can be discovered using UDDI

Example: Web based purchase

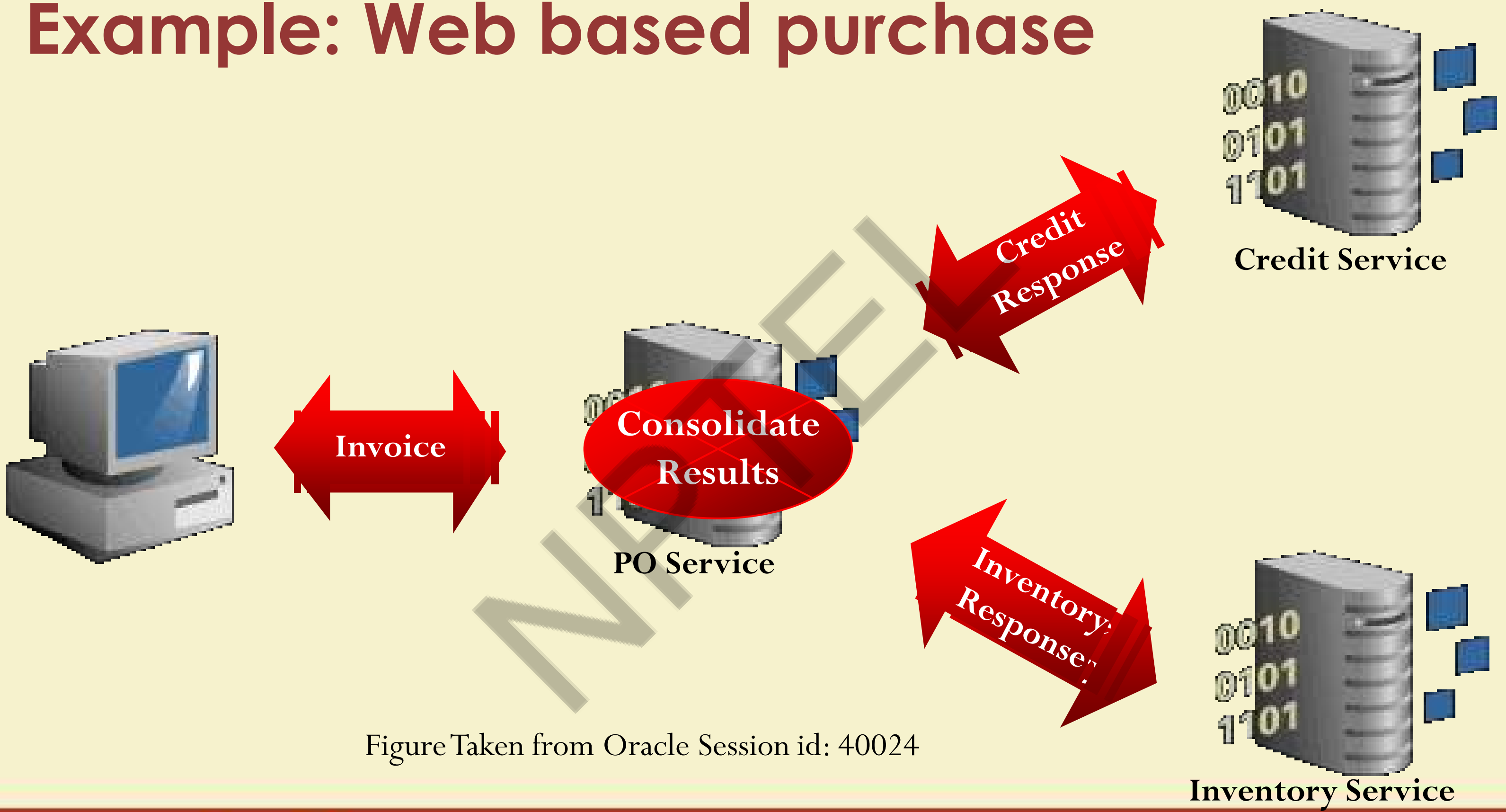
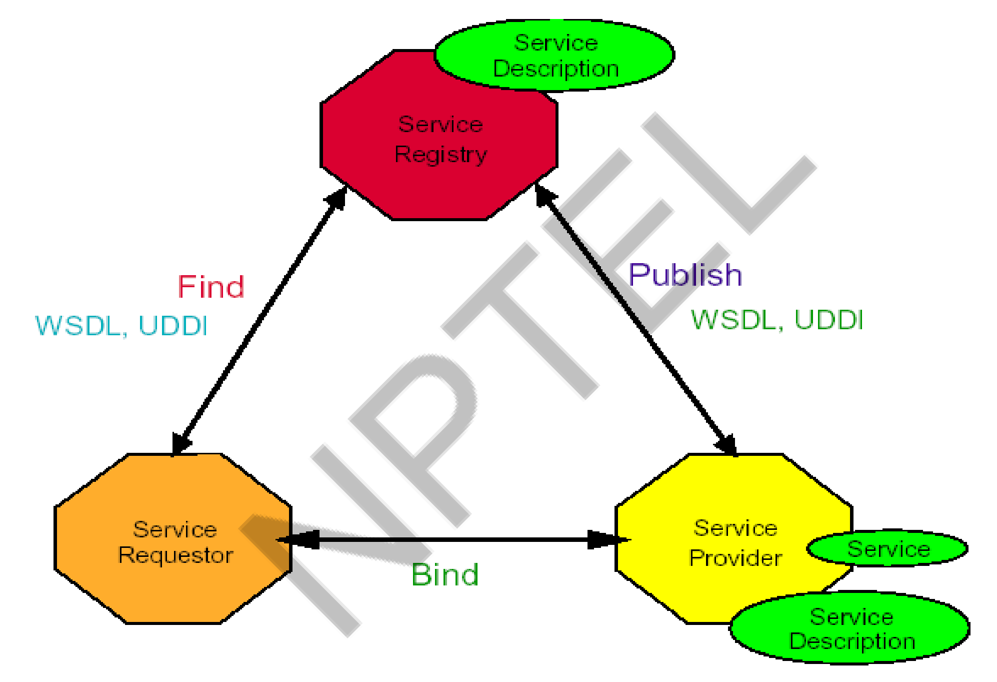


Figure Taken from Oracle Session id: 40024

Service Oriented Architecture (SOA)

- IBM has created a model to show Web services interactions which is referred to as a Service-Oriented Architecture (SOA) consisting of relationships between three entities:
  - A service provider;
  - A service requestor;
  - A service broker
- IBM's SOA is a generic model describing service collaboration, not just specific to Web services.
  - See: http://www-106.ibm.com/developerworks/webservices/

Web Service Model



Web Service Model *(contd...)*

- Roles in Web Service architecture
  - Service provider
    - Owner of the service
    - Platform that hosts access to the service
  - Service requestor
    - Business that requires certain functions to be satisfied
    - Application looking for and invoking an interaction with a service
  - Service registry
    - Searchable registry of service descriptions where service providers publish their service descriptions

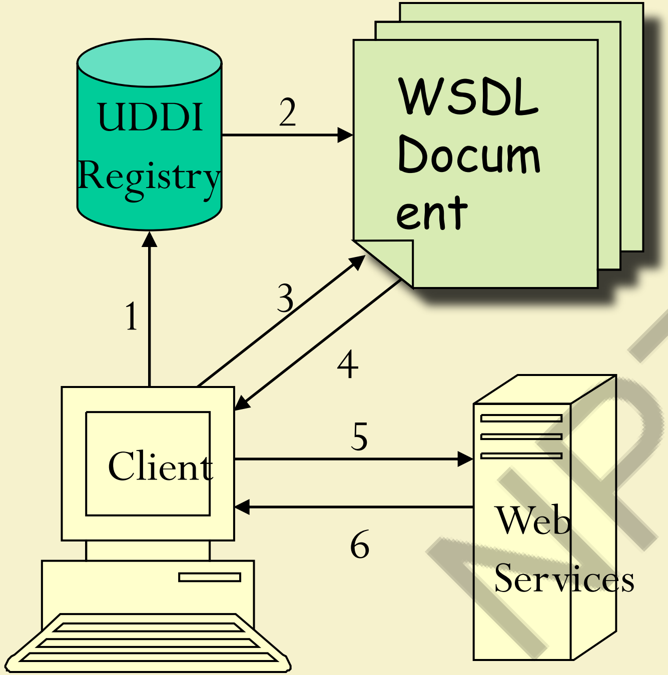
Web Service Model *(contd...)*

- Operations in a Web Service Architecture
  - Publish
    - Service descriptions need to be published in order for service requestor to find them
  - Find
    - Service requestor retrieves a service description directly or queries the service registry for the service required
  - Bind
    - Service requestor invokes or initiates an interaction with the service at runtime

Web Service Components

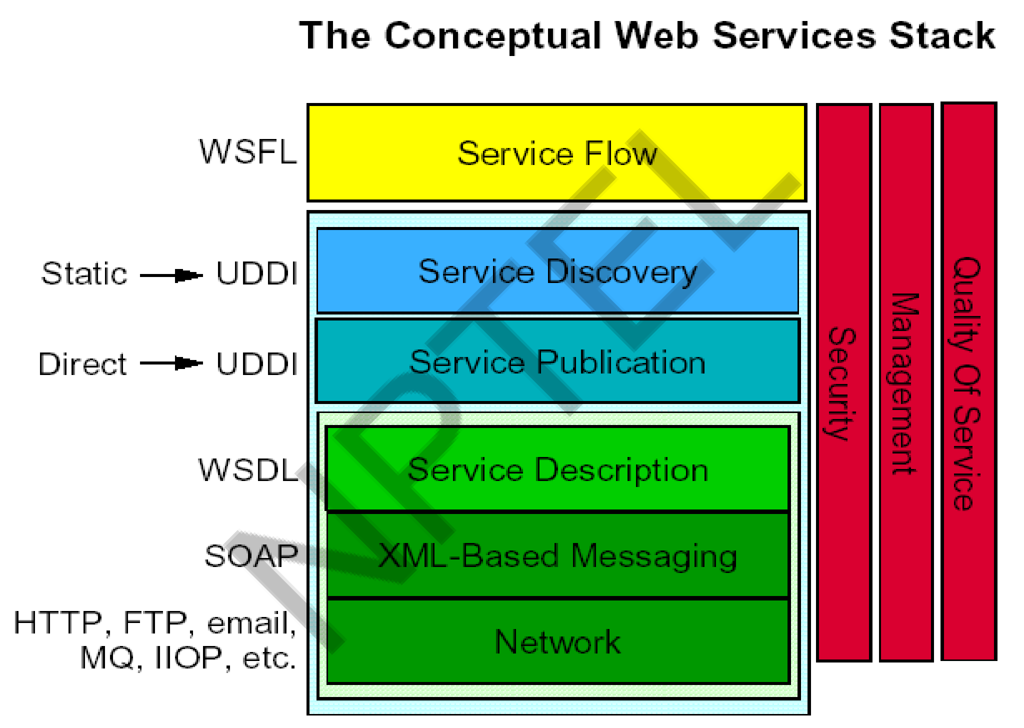
- **XML** – eXtensible Markup Language
  - A uniform data representation and exchange mechanism.
- **SOAP** – Simple Object Access Protocol
  - A standard way for communication.
- **WSDL** – Web Services Description Language
  - A standard meta language to described the services offered.
- **UDDI** – Universal Description, Discovery and Integration specification
  - A mechanism to register and locate WS based application.

Steps of Operation



1. Client queries registry to locate service.
2. Registry refers client to WSDL document.
3. Client accesses WSDL document.
4. WSDL provides data to interact with Web service.
5. Client sends SOAP-message request.
6. Web service returns SOAP-message response.

Web Service Stack



XML

- Developed from Standard Generalized Markup Method (SGML)
- Widely supported by W3C
- Essential characteristic is the separation of content from presentation
- Designed to describe **data**
- XML document can optionally reference a *Document Type Definition (DTD),* also called a *Schema*
  - XML parser checks syntax
  - If an XML document adheres to the structure of the schema it is *valid*

XML *(contd...)*

- XML tags are not predefined
    - You must **define your own tags.**
- Enables cross-platform data communication in Web Services

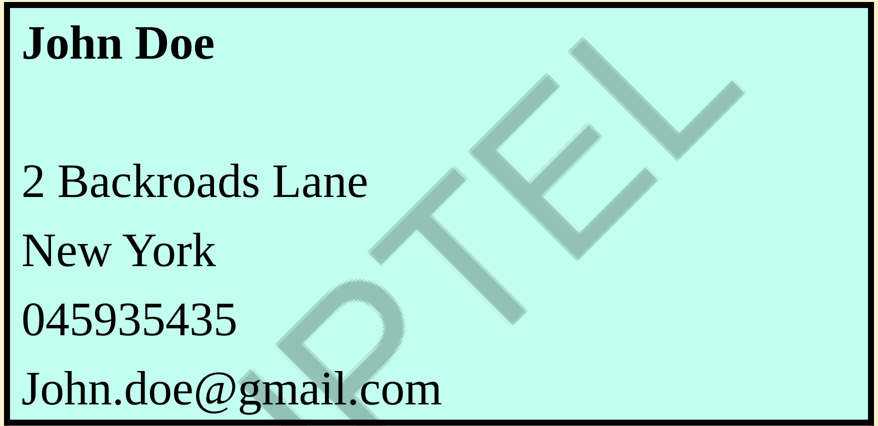
XML vs HTML

An HTML example:

```
<html>
<body>
    <h2>John Doe</h2>
    <p>2 Backroads Lane<br>
         New York<br>
         045935435<br>
         john.doe@gmail.com<br>
         </p>
</body>
</html>
```

XML vs HTML *(contd…)*

- This will be displayed as:



John Doe

2 Backroads Lane

New York

045935435

John.doe@gmail.com

- HTML specifies how the document is to be displayed, and not what information is contained in the document.

- Hard for machine to extract the embedded information. Relatively easy for human.

XML vs HTML *(contd…)*

- Now look at the following:

```
<?xml version=1.0?>
<contact>
   <name>John Doe</name>
   <address>2 Backroads Lane</address>
   <country>New York</country>
   <phone>045935435</phone>
   <email>john.doe@gmail.com</email>
</contact>
```

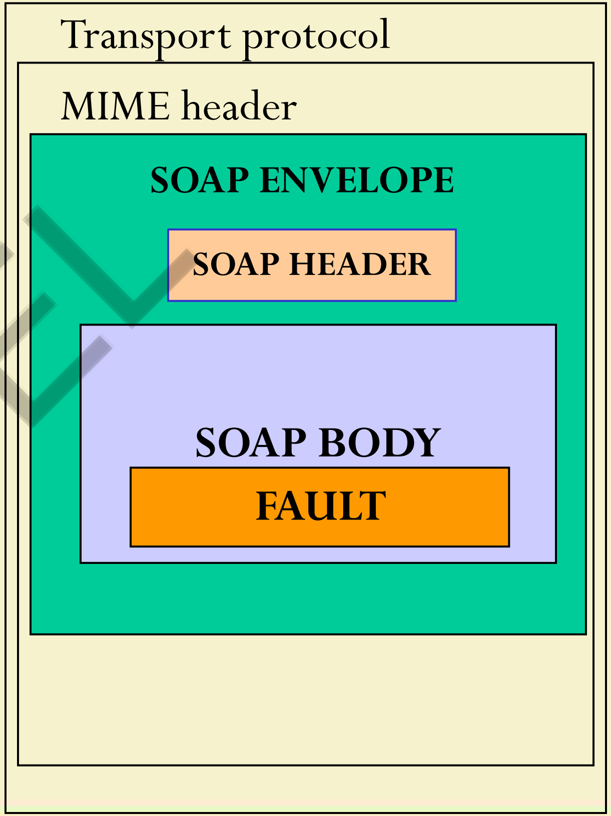
- In this case:
  - The information contained is being marked, but not for displaying.
  - Readable by both human and machines.

SOAP

- **S**imple **O**bject **A**ccess **P**rotocol
- Format for sending messages over Internet between programs
- XML-based
- Platform and language independent
- Simple and extensible
- Uses mainly HTTP as a transport protocol
  - HTTP message contains a SOAP message as its payload section
- Stateless, one-way
  - But applications can create more complex interaction patterns

SOAP Building Blocks

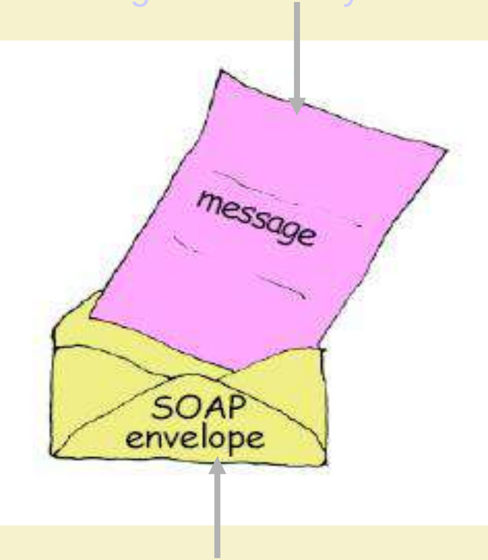
- Envelope (required) – identifies XML document as SOAP message
- Header (optional) – contains header information
- Body (required) –call and response information
- Fault (optional) – errors that occurred while processing message



SOAP Message Structure

- Request and Response messages
  - Request invokes a method on a remote object
  - Response returns result of running the method
- SOAP specification defines an “envelop”
  - “envelop” wraps the message itself
  - Message is a different vocabulary
  - Namespace prefix is used to distinguish the two parts

Application-specific message vocabulary



SOAP Envelop vocabulary

SOAP Request

```
POST /InStock HTTP/1.1
Host: www.stock.org
Content-Type: application/soap+xml; charset=utf-8 Content-Length: 150

<?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.stock.org/stock">
        <m:GetStockPrice>
            <m:StockName>IBM</m:StockName>
        </m:GetStockPrice>
   </soap:Body>
</soap:Envelope>
```

SOAP Response

```
HTTP/1.1 200 OK
Content-Type: application/soap; charset=utf-8
Content-Length: 126

<?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.stock.org/stock">
        <m:GetStockPriceResponse>
                <m:Price>34.5</m:Price>
        </m:GetStockPriceResponse>
    </soap:Body>
</soap:Envelope>
```

Why SOAP?

- Other distributed technologies failed on the Internet
  - Unix RPC – requires binary-compatible Unix implementations at each endpoint
  - CORBA – requires compatible ORBs
  - RMI – requires Java at each endpoint
  - DCOM – requires Windows at each endpoint
- SOAP is the platform-neutral choice
  - Simply an XML wire format
  - Places no restrictions on the endpoint implementation technology choices

SOAP Characteristics

- SOAP has three major characteristics:
  - Extensibility – security and WS-routing are among the extensions under development.
  - Neutrality - SOAP can be used over any transport protocol such as HTTP, SMTP or even TCP.
  - Independent - SOAP allows for any programming model.

SOAP Usage Models

- RPC-like message exchange
  - Request message bundles up method name and parameters
  - Response message contains method return values
  - However, it isn't required by SOAP
- SOAP specification allows any kind of body content
  - Can be XML documents of any type
  - Example:
    - Send a purchase order document to the inbox of B2B partner
    - Expect to receive shipping and exceptions report as response

SOAP Security

- SOAP uses HTTP as a transport protocol and hence can use HTTP security mainly HTTP over SSL.
- But, since SOAP can run over a number of application protocols (such as SMTP) security had to be considered.
- The *WS-Security specification* defines a complete encryption system.

WSDL - Web Service Definition Language

- WSDL : XML vocabulary standard for describing Web services and their capabilities
- Contract between the XML Web service and the client
- Specifies what a request message must contain and what the response message will look like in unambiguous notation
- Defines where the service is available and what communications protocol is used to talk to the service.

WSDL Document Structure

- A WSDL document is just a simple XML document.
- It defines a web service using these major elements:
  - **port type** - The operations performed by the web service.
  - **message -** The messages used by the web service.
  - **types -** The data types used by the web service.
  - **binding -** The communication protocols used by the web service.

A Sample WSDL

```
<message name="getTermRequest">
   <part name="term" type="xs:string"/>
</message>

<message name="getTermResponse">
   <part name="value" type="xs:string"/>
</message>

<portType name="glossaryTerms">
   <operation name="getTerm">
      <input message="getTermRequest"/>
      <output message="getTermResponse"/>
   </operation>
</portType>
```

Binding to SOAP

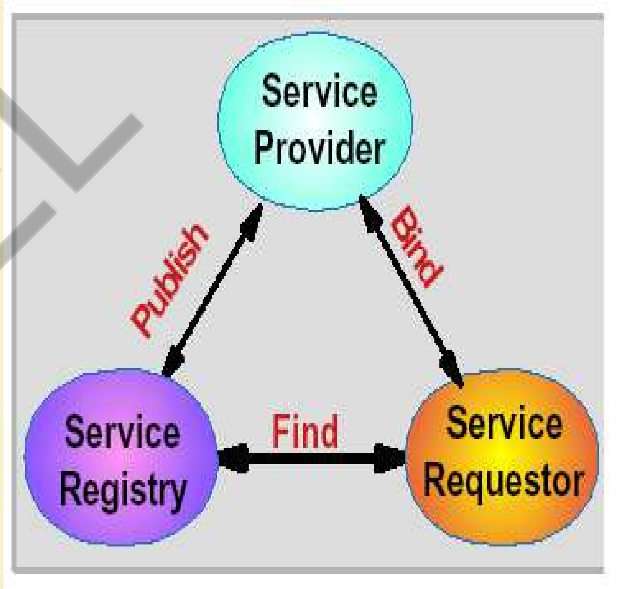
```
<message name="getTermRequest">
   <part name="term" type="xs:string"/>
</message>

<message name="getTermResponse">
   <part name="value" type="xs:string"/>
</message>

<portType name="glossaryTerms">
  <operation name="getTerm">
      <input message="getTermRequest"/>
      <output message="getTermResponse"/>
  </operation>
</portType>

<binding type="glossaryTerms" name="b1">
<soap:binding style="document"
transport="http://schemas.xmlsoap.org/soap/http" />
  <operation>
    <soap:operation
     soapAction="http://example.com/getTerm"/>
    <input>
      <soap:body use="literal"/>
    </input>
    <output>
      <soap:body use="literal"/>
    </output>
  </operation>
</binding>
```

UDDI - Universal Description, Discovery, and Integration

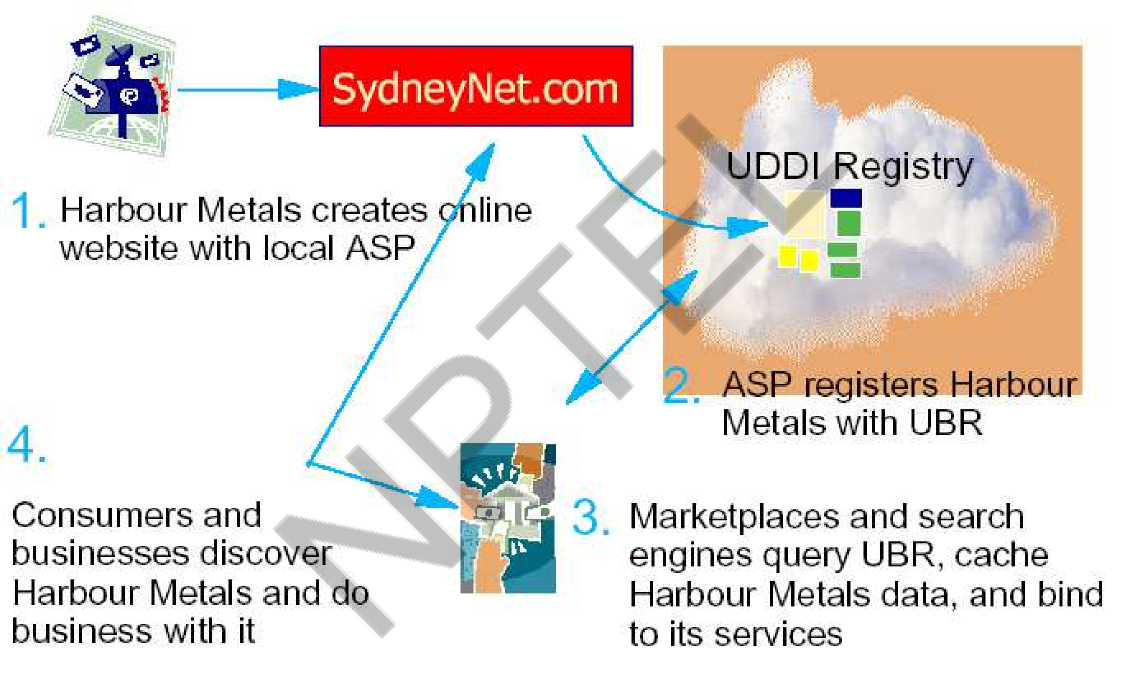
- A framework to define XML-based registries
- Registries are repositories that contain documents that describe business data and also provide search capabilities and programmatic access to remote applications
- Businesses can publish information about themselves and the services they offer
- Can be interrogated by SOAP messages and provides access to WSDL documents describing web services in its directory

UDDI Roles and Operations

- Service Registry
  - Provides support for publishing and locating services
  - Like telephone yellow pages
- Service Provider
  - Provides e-business services
  - Publishes these services through a registry
- Service requestor
  - Finds required services via the Service Broker
  - Binds to services via Service Provider



How can UDDI be Used?



1. Harbour Metals creates online website with local ASP

2. ASP registers Harbour Metals with UBR

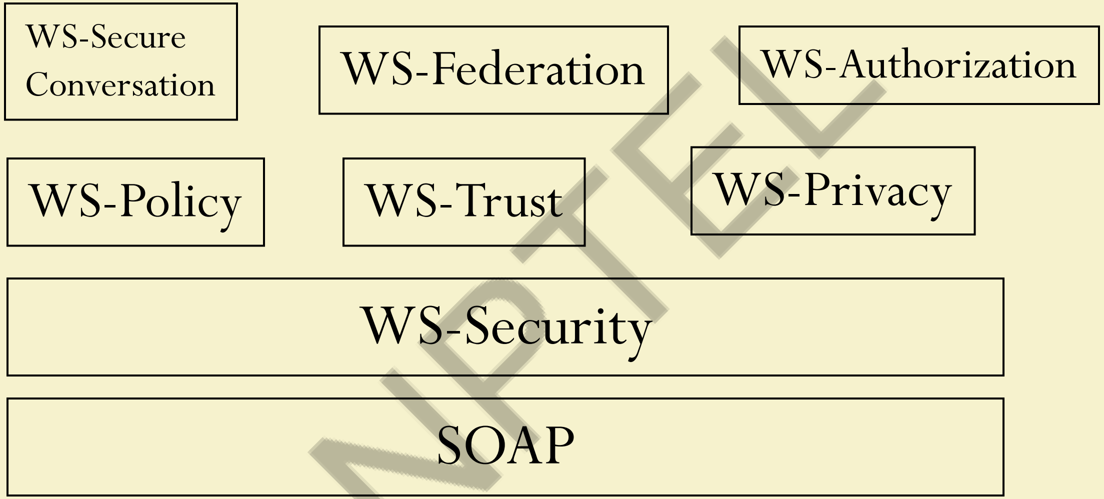
3. Marketplaces and search engines query UBR, cache Harbour Metals data, and bind to its services

4. Consumers and businesses discover Harbour Metals and do business with it

UDDI Benefits

- Making it possible to discover the right business from the millions currently online
- Defining how to enable commerce once the preferred business is discovered
- Reaching new customers and increasing access to current customers
- Expanding offerings and extending market reach

Web Services Security Architecture



Thank You!