UNIT-3.2

In Internet circles, everything eventually gets driven by a working group of one sort or another. A working group is an assembled, cooperative collaboration of researchers working on new research activities that would be difficult for any one member to develop alone. The Open Cloud Consortium (OCC) and the Distributed Management Task Force (DMTF) are the examples of cloud-related working groups.

1 The Open Cloud Consortium

The Open Cloud Consortium (OCC) is a member driven organization that supports the development of standards for cloud computing and frameworks for interoperating between clouds; develops benchmarks for cloud computing; supports reference implementations for cloud computing, preferably open source reference implementations; manages a testbed for Cloud Computing called the Open Cloud Testbed (OCT); and sponsors workshops and other events related to Cloud Computing. The OCC has a particular focus in large data clouds. It has developed the MalStone Benchmark for large data clouds and is working on a reference model for large data clouds

The purpose of the Open Cloud Consortium is to support the development of standards for cloud computing and to develop a framework for interoperability among various clouds. The OCC supports the development of benchmarks for cloud computing and is a strong proponent of open source software to be used for cloud computing. OCC manages a testing platform and a test-bed for cloud computing called the Open Cloud Test-bed. The group also sponsors workshops and other events related to cloud computing.

The OCC is organized into several different working groups.

The Working Group on **Standards and Interoperability for Clouds** That Provide On-Demand Computing Capacity focuses on developing

standards for interoperating clouds that provide on-demand computing capacity. One architecture for clouds that was popularized by a series of Google technical reports describes a *storage cloud* providing a distributed file system, a *compute cloud* supporting MapReduce, and a *data cloud* supporting table services. The open source Hadoop system follows this architecture. These types of cloud architectures support the concept of on demand computing capacity.

TheWorking Group on **Wide Area Clouds and the Impact of Network Protocols on Clouds**. The focus of this working group is on developing technology for wide area clouds, including creation of methodologies and benchmarks to be used for evaluating wide area clouds. This working group is tasked to study the applicability of variants of TCP (Transmission Control Protocol) and the use of other network protocols for clouds.

The Working Group on **Information Sharing, Security, and Clouds** has a primary focus on standards and standards-based architectures for sharing information between clouds. This is especially true for clouds belonging to different organizations and subject to possibly different authorities and policies. This group is also concerned with security architectures for clouds. An example is exchanging information between two clouds, each of which is HIPAA-compliant, but when each cloud is administered by a different organization.

Finally, there is an **Open Cloud Test-bed** Working Group that manages and operates the Open Cloud Test-bed. Currently, membership in this working group is limited to those who contribute computing, networking, or other resources to the Open Cloud Test-bed. For more information on the Open Cloud Consortium, the reader is encouraged to visit the OCC website.

The Open Cloud Test-bed uses Cisco C-Wave and the UIC Teraflow Network for its network connections. C-Wave makes network resources available to researchers to conduct networking and applications research. It is provided at no cost to researchers and allows them access to 10G Waves

(Layer-1 p2p) on a per-project allocation. It provides links to a 10GE (giga-bit Ethernet) switched network backbone. The Teraflow Test-bed (TFT) is an international application network for exploring, integrating, analyzing, and detecting changes in massive and distributed data over wide-area high performance networks. The Teraflow Test-bed analyzes streaming data with the goal of developing innovative technology for data streams at very high speeds. It is hoped that prototype technology can be deployed over the next decade to analyze 100-gigabit-per-second (Gbps) and 1,000-Gbps streams.

2. The Distributed Management Task Force

The DMTF creates open manageability standards spanning diverse emerging and traditional IT infrastructures including cloud, virtualization, network, servers and storage.

The Distributed Management Task Force enables more effective management of millions of IT systems worldwide by bringing the IT industry together to collaborate on the development, validation and promotion of systems management standards. The group spans the industry with 160 member companies and organizations, and more than 4,000 active participants crossing 43 countries. The DMTF board of directors is led by 16 innovative, industry-leading technology companies. They include Advanced Micro Devices (AMD); Broadcom Corporation; CA, Inc.; Dell; EMC; Fujitsu; HP; Hitachi, Ltd.; IBM; Intel Corporation; Microsoft Corporation; Novell; Oracle; Sun Microsystems, Inc.; Symantec Corporation and VMware, Inc. With this deep and broad reach, DMTF creates standards that enable interoperable IT management. DMTF management standards are critical to enabling management interoperability among multi-vendor systems, tools and solutions within the enterprise.

The DMTF started the Virtualization Management Initiative (VMAN).

Definition - What does Virtualization Management mean?

Virtualization management is the process of overseeing and administering the operations and processes of a virtualization environment. It is part of IT management that includes the collective processes, tools and technologies to ensure governance and control over a virtualized infrastructure.

The VMAN unleashes the power of virtualization by delivering broadly supported interoperability and portability standards to virtual computing environments.

VMAN enables IT managers to deploy preinstalled preconfigured solutions across heterogeneous computing networks and to manage those applications through their entire life cycle.

Virtualization has enhanced the IT industry by optimizing use of existing physical resources and helping reduce the number of systems deployed and managed. This consolidation reduces hardware costs and mitigates power and cooling needs. However, even with the efficiencies gained by virtualization, this new approach does add some IT cost due to increased system management complexity.

Since the DMTF builds on existing standards for server hardware, management tool vendors can easily provide holistic management capabilities to enable IT managers to manage their virtual environments in the context of the underlying hardware. This lowers the IT learning curve, and also lowers complexity for vendors implementing this support in their solutions. With the technologies available to IT managers through the VMAN Initiative, companies now have a standardized approach to

1. Deploy virtual computer systems
2. Discover and take inventory of virtual computer systems
3. Manage the life cycle of virtual computer systems
4. Add/change/delete virtual resources
5. Monitor virtual systems for health and performance

2.1 Open Virtualization Format

The Open Virtualization Format (OVF) is a new standard that has emerged within the VMAN Initiative.

Open Virtualization Format (OVF) – a standard for packaging and deploying virtual appliances.

The OVF simplifies interoperability, security, and virtual machine life-cycle management by describing an open, secure, portable, efficient, and extensible format for the packaging and distribution of one or more virtual appliances.

The OVF specifies procedures and technologies to permit integrity checking of the virtual machines (VM) to ensure that they have not been modified since the package was produced.

The OVF also provides mechanisms that support license checking for the enclosed VMs, addressing a key concern of both independent software vendors and customers.
The OVF allows an installed VM to acquire information about its host virtualization platform and runtime environment, which allows the VM to localize the applications it contains and optimize its performance for the particular virtualization environment.

One key feature of the OVF is virtual machine packaging portability. Since OVF is, by design, virtualization platform-neutral, it provides the benefit of enabling platform-specific enhancements to be captured. It supports content verification and integrity checking based on industry-standard public key infrastructure and provides a basic scheme for management of software licensing.

Another benefit of the OVG is a simplified installation and deployment process. The OVF streamlines the entire installation process using metadata to validate the entire package and automatically determine whether a virtual appliance can be installed. It also supports both single-VM and multipleVM configurations and packages containing complex, multitier services consisting of multiple interdependent VMs. Since it is vendor- and platform-independent, the OVF does not rely on the use of a specific host platform, virtualization platform, or guest operating system.

3 Standards for Application Developers

The purpose of application development standards is to ensure uniform, consistent, high-quality software solutions.

Programming standards are important to programmers for a variety of reasons. Some researchers have stated that, as a general rule, 80% of the lifetime cost of a piece of software goes to maintenance.

Furthermore, hardly any software is maintained by the original author for its complete life cycle. Programming standards help to improve the readability of the software, allowing developers to understand new code more quickly and thoroughly. If you ship source code as a product, it is important to ensure that it is as well packaged and meets industry standards comparable to the products you compete with. For the standards to work, everyone developing solutions must conform to them. Application standards that are commonly used across the Internet in browsers, for transferring data, sending messages, and securing data.

3.1 Browsers (Ajax)

Ajax, or its predecessor AJAX (Asynchronous JavaScript and XML), is a group of interrelated web development techniques used to create interactive web applications or rich Internet applications. Using Ajax, web applications can retrieve data from the server asynchronously, without interfering with the display and behavior of the browser page currently being displayed to the user. The use of Ajax has led to an increase in interactive animation on web pages. Despite its name, JavaScript and XML are not actually *required* for Ajax. Moreover, requests do not even need to be asynchronous. The original acronym AJAX has changed to the name Ajax to reflect the fact that these specific technologies are no longer required.
In many cases, related pages that coexist on a web site share much common content. Using traditional methods, such content must be reloaded every time a request is made. Using Ajax, a web application can

request only the content that needs to be updated. This greatly reduces networking bandwidth usage and page load times. Using asynchronous requests allows a client browser to appear more interactive and to respond to input more quickly. Sections of pages can be reloaded individually. Users generally perceive the application to be faster and more responsive. Ajax can reduce connections to the server, since scripts and style sheets need only be requested once.

An Ajax framework helps developers create web applications that use Ajax. The framework helps them to build dynamic web pages on the client side. Data is sent to or from the server using requests, usually written in JavaScript. On the server, some processing may be required to handle these requests, for example, when finding and storing data. This is accomplished more easily with the use of a framework dedicated to process Ajax requests.

ICEfaces Ajax Application Framework

ICEfaces is an integrated Ajax application framework that enables Java EE application developers to easily create and deploy thin-client rich Internet applications in pure Java. ICEfaces is a fully featured product that enterprise developers can use to develop new or existing Java EE applications at no cost. ICEfaces is the most successful enterprise Ajax framework available under open source. The ICEfaces developer community is extremely vibrant, already exceeding 32,000 developers in 36 countries. To run ICEfaces applications, users need to download and install the following products:

- Java 2 Platform, Standard Edition
- Ant
- Tomcat
- ICEfaces
- Web browser

ICEfaces leverages the entire standards-based Java EE set of tools and environments. Rich enterprise application features are developed in pure Java in a

thin-client model. No Applets or proprietary browser plug-ins are required. ICEfaces applications are JavaServer Faces (JSF) applications, so Java EE application development skills apply directly and Java developers don't have to do any JavaScript-related development.Because ICEfaces is a pure Java enterprise solution, developers can continue to work the way they normally do. They are able to leverage their existing Java integrated development environments (IDEs) and test tools for development. ICEfaces supports an array of Java Application Servers, IDEs, third-party components, and JavaScript effect libraries. ICEfaces pioneered a technique called Ajax Push. This technique enables server/ application-initiated content rendering to be sent to the browser. Also, ICEfaces is the one of the most secure Ajax solutions available.Compatible with SSL (Secure Sockets Layer) protocol, it prevents cross-site scripting, malicious code injection, and unauthorized data mining. ICEfaces does not expose application logic or user data, and it is effective in preventing fake form submits and SQL (Structured Query Language) injection attacks. ICEfaces also supports third-party application server Asynchronous Request Processing (ARP) APIs provided by Sun Glassfish(Grizzly), Jetty, Apache Tomcat, and others.

3.2 Data (XML, JSON)

Extensible Markup Language (XML) is a specification for creating custom markup languages. It is classified as an extensible language because it allows the user to define markup elements. Its purpose is to enable sharing of structured data. XML is often used to describe structured data and to serialize objects. Various XML-based protocols exist to represent data structures for data interchange purposes. Using XML is arguably more complex than using JSON (described below), which represents data structures in simple text formatted specifically for data interchange in an uncompressed form. Both XML and JSON lack mechanisms for representing large binary data types such as images.

XML, in combination with other standards, makes it possible to define the content of a document separately from its formatting. The benefit here

is the ability to reuse that content in other applications or for other presentation environments. Most important, XML provides a basic syntax that can be used to share information among different kinds of computers, different applications, and different organizations without needing to be converted from one to another.

An XML document has two correctness levels, *well formed* and *valid*. A well-formed document conforms to the XML syntax rules. A document that is not well formed is not in XML format, and a conforming parser will not process it. A valid document is well formed and additionally conforms to semantic rules which can be user-defined or exist in an XML schema. An XML schema is a description of a type of XML document, typically expressed in terms of constraints on the structure and content of documents of that type, above and beyond the basic constraints imposed by XML itself. A number of standard and proprietary XML schema languages have emerged for the purpose of formally expressing such schemas, and some of these languages are themselves XML-based.

XML documents must conform to a variety of rules and naming conventions. By carefully choosing the names of XML elements, it is possible to convey the meaning of the data in the markup itself. This increases human readability while retaining the syntactic structure needed for parsing. However, this can lead to verbosity, which complicates authoring and increases file size. When creating XML, the designers decided that by leaving the names, allowable hierarchy, and meanings of the elements and attributes open and definable by a customized schema, XML could provide a syntactic foundation for the creation of purpose-specific, XML-based markup languages.

Before the advent of generalized data description languages such as XML, software designers had to define special file formats or small languages to share data between programs. This required writing detailed specifications and special-purpose parsers and writers. XML's regular structure and strict parsing rules allow software designers to leave the task of parsing to standard tools, since XML provides a general, data model-oriented

framework for the development of application-specific languages. This allows software designers to concentrate on the development of rules for their data at relatively high levels of abstraction.

JavaScript Object Notation (JSON)

JSON is a lightweight computer data interchange format. It is a text-based, human-readable format for representing simple data structures and associative arrays (called objects). The JSON format is specified in Internet Engineering Task Force Request for Comment (RFC) 4627. The JSON format is often used for transmitting structured data over a network connection in a process called serialization. Its main application is in Ajax web application programming, where it serves as an alternative to the XML format. JSON is based on a subset of the JavaScript programming language. It is considered to be a language-independent data format. Code for parsing and generating JSON data is readily available for a large variety of programming languages. The json.org website provides a comprehensive listing of existing JSON bindings, organized by language.

Even though JSON was intended as a data serialization format, its design as a subset of the JavaScript language poses security concerns. The use of a JavaScript interpreter to dynamically execute JSON text as JavaScript can expose a program to bad or even malicious script. JSON is also subject to cross-site request forgery attacks. This can allow JSON-encoded data to be evaluated in the context of a malicious page, possibly divulging passwords or other sensitive data. This is only a problem if the server depends on the browser's Same Origin Policy to block the delivery of the data in the case of an improper request. When the server determines the propriety of the request, there is no problem because it will only output data if the request is valid. Cookies are not adequate for determining whether a request is authorized and valid. The use of cookies is subject to cross-site request forgery and should be avoided with JSON. As you can see, JSON was built for simple tasks and can be useful, but there is some risk involved in using it—especially given the alternative solutions available today.

3.3 Solution Stacks (LAMP and LAPP)

LAMP

LAMP is a popular open source solution commonly used to run dynamic web sites and servers. The acronym derives from the fact that it includes **L**inux, **A**pache, **M**ySQL, and **P**HP (or Perl or Python) and is considered by many to be the platform of choice for development and deployment of high-performance web applications which require a solid and reliable foundation. The combination of these technologies is used primarily to define a web server infrastructure or for creating a programming environment for developing software. While the creators of these open source products did not intend for them all to work with each other, the LAMP combination has become popular because of its open source nature, low cost, and the wide distribution of its components (most of which come bundled with nearly all of the current Linux distributions). When used in combination, they represent a solution stack of technologies that support application servers.

Linux, Apache, PostgreSQL, and PHP(or Perl or Python)

The LAPP stack is an open source web platform that can be used to run dynamic web sites and servers. It is considered by many to be a more powerful alternative to the more popular LAMP stack. These advanced and mature components provide a rock-solid foundation for the development and deployment of high-performance web applications. LAPP offers SSL, PHP, Python, and Perl support for Apache2 and PostgreSQL. There is an administration front-end for PostgreSQL as well as web-based administration modules for configuring Apache2 and PHP. PostgreSQL password encryption is enabled by default. The PostgreSQL user is trusted when connecting over local Unix sockets. Many consider the LAPP stack a more secure out-of-the-box solution than the LAMP stack. The choice of which stack to use is made by developers based on the purpose of their application and the risks they may have to contend with when users begin working with the product.

4 Standards for Messaging

The term *messaging,* however, covers a lot of ground, and not all of it is specific to networking. A *message* is a unit of information that is moved from one place to another. The term *standard* also is not always clearly defined. Different entities have differing interpretations of what a standard is, and we know there are open international standards, *de facto* standards, and proprietary standards. A true standard is usually characterized by certain traits, such as being managed by an international standards body or an industry consortium, and the standard is created jointly by a community of interested parties. The Internet Engineering Task Force (IETF) is perhaps the most open standards body, because it is open to everyone. Participants can contribute, and their work is available online for free. In the following sections, we discuss the most common messaging standards used in the cloud—some of which have been used so much so that they are considered *de facto* standards.

4.1 Simple Message Transfer Protocol (SMTP)

Simple Message Transfer Protocol is arguably the most important protocol in use today for basic messaging. Before SMTP was created, email messages were sent using File Transfer Protocol (FTP). A sender would compose a message and transmit it to the recipient as if it were a file. While this process worked, it had its shortcomings. The FTP protocol was designed to transmit files, not messages, so it did not provide any means for recipients to identify the sender or for the sender to designate an intended recipient. If a message showed up on an FTP server, it was up to the administrator to open or print it (and sometimes even deliver it) before anyone even knew who it was supposed to be receiving it. SMTP was designed so that sender and recipient information could be transmitted with the message. SMTP is a two-way protocol that usually operates using TCP (Transmission Control Protocol) port 25. Though many people don't realize it, SMTP can be used to both send and receive messages. Typically, though, workstations use POP (Post Office Protocol) rather than SMTP to receive messages. SMTP is usually

used for either sending a message from a workstation to a mail server or for communications between mail servers.

4.2 Post Office Protocol (POP)

POP is a lightweight protocol whose single purpose is to download messages from a server. This allows a server to store messages until a client connects and requests them. Once the client connects, POP servers begin to download the messages and subsequently delete them from the server (a default setting) in order to make room for more messages. Users respond to a message that was downloaded using SMTP. The POP protocol is defined by RFC 1939 and usually functions on TCP port 110.

4.3 Internet Messaging Access Protocol (IMAP)

Once mail messages are downloaded with POP, they are automatically deleted from the server when the download process has finished. Thus POP users have to save their messages locally, which can present backup challenges when it is important to store or save messages. Many businesses have compulsory compliance guidelines that require saving messages. It also becomes a problem if users move from computer to computer or use mobile networking, since their messages do not automatically move where they go. To get around these problems, a standard called Internet Messaging Access Protocol was created. IMAP allows messages to be kept on the server but viewed and manipulated (usually via a browser) as though they were stored locally.

4.4 Syndication (Atom, Atom Publishing Protocol, and RSS)

Content syndication provides citizens convenient access to new content and headlines from government via RSS (Really Simple Syndication) and other online syndication standards. Governments are providing access to more and more information online. As web sites become more complex and difficult to sift through, new or timely content is often buried. Dynamically presenting “what’s new” on the top of the web site is only the first step. Sharing headlines and content through syndication standards such as RSS (the little orange [XML] button, ATOM, and others) essentially allows a government to control a small window of content across web sites that choose to display

the government's headlines. Headlines may also be aggregated and displayed through "newsreaders" by citizens through standalone applications or as part of their personal web page.

Portals can automatically aggregate and combine headlines and/or lengthier content from across multiple agency web sites. This allows the value of distributed effort to be shared, which is more sustainable. Press releases may be aggregated automatically from different systems, as long as they all are required to offer an RSS feed with content tagged with similar metadata. Broader use of government information online, particularly time sensitive democratic information, justifies the effort of production and the accountability of those tasked to make it available.

Benefits: Ability to scan headlines from many sources, all in one place, through a newsreader. Time-saving awareness of new content from government, if the RSS feed or feeds are designed properly. Ability to monitor new content from across the council, as well as display feeds on their own web site. Awareness of new content position councilors as guides to government for citizens. Ability to aggregate new content or headlines from across multiple office locations and agencies. This allows a display of "joined-up" government despite structural realities. Journalists and other locally focused web sites will be among the primary feed users.

Limitations: Dissemination via syndication is a new concept to governments just getting used to the idea of remote online public access to information. Governments need to accept that while they control the content of the feed, the actual display of the headlines and content will vary. Popular RSS feeds can use significant amounts of bandwidth. Details on how often or when a feed is usually updated should be offered to those grabbing the code behind the orange [XML] button, so they "ping" it once a day instead of every hour. Automated syndication requires use of a content management system. Most viable content management systems have integrated RSS functions, but the sophistication, ease of use, and documentation of these tools vary. There are three variants of RSS, as well as the emerging ATOM standard. It is recommended that a site pick the standard most applicable to

their content rather than confuse users with different feeds providing the same content.

RSS

RSS is a family of web feed formats used to publish frequently updated works—such as blog entries, news headlines, audio, and video—in a standardized format. An RSS document includes full or summarized text, plus metadata such as publishing dates and authorship. Web feeds benefit publishers by letting them syndicate content automatically. They benefit readers who want to subscribe to timely updates from favored web sites or to aggregate feeds from many sites into one place. RSS feeds can be read using software called a reader that can be web-based, desktop-based, a mobile device, or any computerized Internet-connected device. A standardized XML file format allows the information to be published once and viewed by many different programs. The user subscribes to a feed by entering the feed's URI into the reader or by clicking an RSS icon in a browser that initiates the subscription process. The RSS reader checks the user's subscribed feeds regularly for new work, downloads any updates that it finds, and provides a user interface to monitor and read the feeds.

Atom and Atom Publishing Protocol (APP)

The name Atom applies to a pair of related standards. The Atom Syndication Format is an XML language used for web feeds, while the Atom Publishing Protocol (AtomPub or APP) is a simple HTTP-based protocol for creating and updating web resources, sometimes known as web feeds. Web feeds allow software programs to check for updates published on a web site. To provide a web feed, a site owner may use specialized software (such as a content management system) that publishes a list (or "feed") of recent articles or content in a standardized, machine-readable format. The feed can then be downloaded by web sites that syndicate content from the feed, or by feed reader programs that allow Internet users to subscribe to feeds and view their content. A feed contains entries, which may be headlines, full-text articles, excerpts, summaries, and/or links to content on a web site, along with various metadata.

The Atom format was developed as an alternative to RSS. Ben Trott, an advocate of the new format that became Atom, believed that RSS had limitations and flaws—such as lack of ongoing innovation and its necessity to remain backward compatible—and that there were advantages to a fresh design. Proponents of the new format formed the IETF Atom Publishing Format and Protocol Workgroup. Web feeds are used by the weblog community to share the latest entries' headlines or their full text, and even attached multimedia files. These providers allow other web sites to incorporate the weblog's "syndicated" headline or headline-and-short-summary feeds under various usage agreements.

Atom and other web syndication formats are now used for many purposes, including journalism, marketing, "bug" reports, or any other activity involving periodic updates or publications. Atom also provides a standardized way to export an entire blog, or parts of it, for backup or for importing into other blogging systems. A program known as a feed reader or aggregator can check web pages on behalf of a user and display any updated articles that it finds. It is common to find web feeds on major web sites, as well as on many smaller ones. Some web sites let people choose between RSS- or Atom-formatted web feeds; others offer only RSS or only Atom.

Web Services (REST)

REpresentational State Transfer (REST) is a style of software architecture for distributed hypermedia systems such as the World Wide Web.

The terms "representational state transfer" and "REST" were introduced in 2000 in the doctoral dissertation of Roy Fielding, one of the principal authors of the Hypertext Transfer Protocol (HTTP) specification.

REST refers to a collection of network architecture principles which outline how resources are defined and addressed.

To describe any simple interface which transmits domain-specific data over HTTP without an additional messaging layer such as SOAP or session tracking via HTTP cookies.

It is possible to design a software system in accordance with Fielding's REST architectural style without using HTTP and without interacting with the World Wide Web.

It is also possible to design simple XML+HTTP interfaces which do not conform to REST principles, but instead follow a model of remote procedure call. Systems which follow Fielding's REST principles are often referred to as "RESTful.

Application state and functionality are abstracted into resources. Every resource is uniquely addressable using a universal syntax for use in hypermedia links, and all resources share a uniform interface for the transfer of state between client and resource.

State transfer uses a protocol which is client-server based, stateless and cacheable, and layered. Fielding describes REST's effect on scalability thus:

REST's client-server separation of concerns simplifies component implementation, reduces the complexity of connector semantics, improves the effectiveness of performance tuning, and increases the scalability of pure server components. Layered system constraints allow intermediaries—proxies, gateways, and firewalls—to be introduced at various points in the communication without changing the interfaces between components, thus allowing them to assist in communication translation or improve performance via large-scale, shared caching. REST enables intermediate processing by constraining messages to be self-descriptive: interaction is stateless between requests, standard methods and media types are used to indicate semantics and exchange information, and responses explicitly indicate cacheability

An important concept in REST is the existence of resources, each of which is referenced with a global identifier (e.g., a URI in HTTP). In order to manipulate these resources, components of the network (user agents and origin servers) communicate via a standardized interface (e.g., HTTP) and exchange representations of these resources.

Any number of connectors (clients, servers, caches, tunnels, etc.) can mediate the request, but each does so without "seeing past" its own request. Thus an application can interact with a resource by knowing two things: the identifier of the resource, and the action required—it does not need to know whether there are caches, proxies, gateways, firewalls, tunnels, or anything else between it and the server actually holding the information. The application does, however, need to understand the format of the information (representation) returned, which is typically an HTML, XML, or JSON document of some kind, although it may be an image, plain text, or any other content

REST provides improved response time and reduced server load due to its support for the caching of representations. REST improves server scalability by reducing the need to maintain session state.

REST requires less client-side software to be written than other approaches, because a single browser can access any application and any resource. REST depends less on vendor software and mechanisms which layer additional messaging frameworks on top of HTTP. It provides equivalent functionality when compared to alternative approaches to communication, and it does not require a separate resource discovery mechanism, because of the use of hyperlinks in representations.

RESTful implementation allows a user to bookmark specific "queries" (or requests) and allows those to be conveyed to others across email, instant messages, or to be injected into wikis, etc. Thus this "representation" of a path or entry point into an application state becomes highly portable.

A RESTFul web service is a simple web service implemented using HTTP and the principles of REST. Such a web service can be thought of as a collection of resources comprising three aspects:

1. The URI for the web service
2. The MIME type of the data supported by the web service (often JSON, XML, or YAML, but can be anything)

3. The set of operations supported by the web service using HTTP methods, including but not limited to POST, GET, PUT, and DELETE

SOAP

SOAP, originally defined as Simple Object Access Protocol, is a protocol specification for exchanging structured information in the implementation of Web Services in computer networks.

It relies on XML as its message format and usually relies on other application-layer protocols, most notably Remote Procedure Call (RPC) and HTTP for message negotiation and transmission. SOAP can form the foundation layer of a web services protocol stack, providing a basic messaging framework on which web services can be built.

The SOAP architecture consists of several layers of specifications for message format, message exchange patterns (MEPs), underlying transport protocol bindings, message processing models, and protocol extensibility. SOAP is the successor of XML-RPC. SOAP makes use of an Internet application-layer protocol as a transport protocol.

Both SMTP and HTTP are valid application-layer protocols used as transport for SOAP, but HTTP has gained wider acceptance because it works well with today's Internet infrastructure; specifically, HTTP works well with network firewalls.

SOAP may also be used over HTTPS (which is the same protocol as HTTP at the application level, but uses an encrypted transport protocol underneath) with either simple or mutual authentication; this is the advocated WS-I method to provide web service security as stated in the WS-I Basic Profile 1.1. This is a major advantage over other distributed protocols such as GIOP/IIOP or DCOM, which are normally filtered by firewalls.

XML was chosen as the standard message format because of its widespread use by major corporations and open source development efforts.

Advantages of using SOAP over HTTP are that SOAP allows for easier communication through proxies and firewalls than previous remote execution technology. SOAP is versatile enough to allow for the use of different transport protocols. The standard stacks use HTTP as a transport protocol, but other protocols are also usable (e.g., SMTP). SOAP is platform-independent, language-independent, and it is simple and extensible.

Because of the verbose XML format, SOAP can be considerably slower than competing middleware technologies such as CORBA (Common Object Request Broker Architecture). This may not be an issue when only small messages are sent. To improve performance for the special case of XML with embedded binary objects, Message Transmission Optimization Mechanism was introduced.

Communications (HTTP, SIMPLE, and XMPP)

HTTP is a request/response communications standard based on a client/ server model. A client is the end user, the server is the web site. The client making a HTTP request via a web browser or other tool sends the request to the server. The responding server is called the origin server. HTTP is not constrained to use TCP/IP and its supporting layers, although this is its most popular application on the Internet.

SIMPLE, the Session Initiation Protocol for Instant Messaging and Presence Leveraging Extensions, is an instant messaging (IM) and presence protocol suite based on Session Initiation Protocol, and it is managed by the IETF. Like XMPP, SIMPLE is an open standard.

Extensible Messaging and Presence Protocol (XMPP) is also an open, XML-based protocol originally aimed at near-real-time, extensible instant messaging and presence information (e.g., buddy lists) but now expanded into the broader realm of message-oriented middleware.