Cloud computing is based on the client-server paradigm discussed in Section 2.13. The vast majority

of cloud applications take advantage of request/response communication between clients and stateless

servers. A stateless server does not require a client to first establish a connection to the server. Instead,

it views a client request as an independent transaction and responds to it.

The advantages of stateless servers are obvious. Recovering from a server failure requires considerable

overhead for a server that maintains the state of all its connections, whereas in the case of a stateless server a client is not affected while a server goes down and then comes back up between two

consecutive requests. A stateless system is simpler, more robust, and scalable. A client does not have

to be concerned with the state of the server. If the client receives a response to a request, that means

that the server is up and running; if not, it should resend the request later. A connection-based service

must reserve spaces to maintain the state of each connection with a client; therefore, such a system is

not scalable, and the number of clients a server could interact with at any given time is limited by the

storage space available to the server.

For example, a basic Web server is stateless; it responds to an HTTP request without maintaining a

history of past interactions with the client. The client, a browser, is also stateless since it sends requests

and waits for responses.The Hypertext TransferProtocol (HTTP) used by a browser to communicate with

the Web server is a request/response application protocol. HTTP uses the Transport Control Protocol

(TCP), a connection-oriented and reliable transport protocol. The use of TCP ensures reliable delivery

of large objects but exposes the Web servers to denial-of-service attacks when malicious clients fake

attempts to establish a TCP connection and force the server to allocate space for the connection.

A critical aspect of the development of networked applications is how processes and threads running

on systems with different architectures and possibly compiled from different programming languages

can communicate structured information with one another. First, the internal representation of the two

structures at the two sites may be different. One system may use Big-Endian and the other Little-Endian

representation. The character representations may also be different. Second, a communication channel

transmits a sequence of bits and bytes; thus, the data structure must be serialized at the sending site and

reconstructed at the receiving site.

Several other considerations must be analyzed before deciding on the architectural style of an application.

The term neutrality refers to the ability of the application protocol to use different transport

protocols such as TCP or UDP and, in general, to run on top of a different protocol stack. For example,

we shall see that SOAP can use TCP but also UDP, SMTP,3 or JMS4 as transport vehicles. Extensibility

refers to the ability to incorporate additional functions, such as security. Independence refers to the

ability to accommodate different programming styles.

Very often the application clients and the servers running on the cloud communicate using RPCs,

discussed in Section 2.13, but other styles of communication are possible. RPC-based applications use

stubs to convert the parameters involved in an RPC call. A stub performs two functions: marshalling the

data structures and serialization. A more general concept is that of an Object Request Broker (ORB),

the middleware that facilitates communication of networked applications. The ORB at the sending site

transforms the data structures used internally by a sending process to a byte sequence and transmits

this byte sequence over the network. The ORB at the receiving site maps the byte sequence to the data

structures used internally by the receiving process.

The CommonObject RequestBrokerArchitecture (CORBA)was developed in the early 1990s to allow

networked applications developed in different programming languages and running on systems with

different architectures and system software to work with one another. At the heart of the system is the

Interface Definition Language (IDL), used to specify the interface of an object. The IDL representation is then mapped to the set of programming languages, including C, C++, Java, Smalltalk, Ruby, LISP,

and Python. Networked applications pass CORBA by reference and pass data by value.

The Simple Object Access Protocol (SOAP) is an application protocol developed in 1998 for Web

applications; its message format is based on the Extensible Markup Language (XML). SOAP uses

TCP and, more recently, UDP transport protocols. It can also be stacked above other application layer

protocols such as HTTP, SMTP, or JMS. The processing model of SOAP is based on a network consisting

of senders, receivers, intermediaries, message originators, ultimate receivers, and message paths. SOAP

is an underlying layer of Web Services.

The Web Services Description Language (WSDL) (see www.w3.org/TR/wsdl) was introduced

in 2001 as an XML-based grammar to describe communication between endpoints of a networked

application. The abstract definition of the elements involved include services, collections of endpoints

of communication; types, containers for data type definitions; operations, descriptions of actions supported

by a service; port types, operations supported by endpoints; bindings, protocols and data formats

supported by a particular port type; and port, an endpoint as a combination of a binding and a network

address. These abstractions are mapped to concrete message formats and network protocols to define

endpoints and services.

Representational State Transfer (REST) is a style of software architecture for distributed hypermedia

systems. REST supports client communication with stateless servers. It is platform- and language independent,

supports data caching, and can be used in the presence of firewalls.

REST almost always uses HTTP to support all four Create/Read/Update/Delete (CRUD) operations.

It uses GET, PUT, and DELETE to read, write, and delete the data, respectively. REST is a much

easier-to-use alternative to RPC, CORBA, or Web Services such as SOAP or WSDL. For example, to

retrieve the address of an individual from a database, a REST system sends a URL specifying the

network address of the database, the name of the individual, and the specific attribute in the record

the client application wants to retrieve – in this case, the address. The corresponding SOAP version

of such a request consists of 10 lines or more of XML. The REST server responds with the address of

the individual. This justifies the statement that REST is a lightweight protocol. As far as usability is

concerned, REST is easier to build from scratch and to debug, but SOAP is supported by tools that use

self-documentation (e.g., WSDL to generate the code to connect).