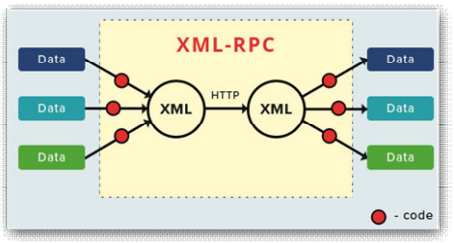


Programación Orientada a Objetos
XMLRPC

Introducción a Redes TCP/IP - HTTP - XML - RPC

Esp. Ing. César Omar Aranda v2.0

Objetivos/Contenidos Fundamentales

1. Adquirir los conceptos básicos de redes de datos basadas en TCP/IP
2. Describir aspectos fundamentales de HTTP
3. Conocer el formato XML y sus elementos
4. Comprender el modo de funcionamiento de las RPC
5. Mencionar aspectos útiles de implementación
6. Analizar el desempeño de un ejemplo simple usando XML-RPC

Referencias/Bibliografía

- Forouzan, B.A. (2002). Transmisión de Datos y redes de Comunicaciones. Editorial McGrawHill, Madrid.
- Coulouris, G. (2001). Sistemas Distribuidos. 3ra Edición. Addison Wesley. Madrid.
- Castro Letchaler, A.R Y Fusario, R.J. (2012). Comunicaciones. Editorial Marcombo.
- http://xmlrpc.scripting.com/spec.html
- https://sourceforge.net/projects/xmlrpcpp/
- https://bestwebsoft.com/what-is-xml-rpc/
- https://blog.papercut.com/write-xml-rpc-clients/

Red de Datos

- También de nominada como Red de Computadoras, Red de Comunicaciones de Datos o Red Informática
- Es un conjunto de equipos de cómputo, denominados nodos, y software conectados entre sí por medio de dispositivos que envían y reciben impulsos eléctricos, ondas electromagnéticas o cualquier otro medio para el transporte de datos.
- Como en todo proceso de comunicación, requiere de un emisor, un mensaje, un medio y un receptor.
- La finalidad principal para la creación de una red de datos es compartir la información y los recursos, independientemente de la distancia, ofrecer servicios, asegurar la confiabilidad y la disponibilidad de la información, aumentar la velocidad de transmisión de los datos y reducir los costos.

Introducción al Modelo OSI

OSI (Open System Interconections) es un modelo de referencia creado en 1978 por la ISO (International Standarization Organization).

✓Objetivo
- Definir un conjunto de normas que permitan interconectar diferentes equipos, generalmente de sistemas heterogéneos, posibilitando la comunicación entre ellos.

Un Sistema de dice Abierto cuando cumple con las normas que le facilitan la interconexión a nivel software y hardware con otros sistemas.

Se aprobó en 1983, y luego que el CCITT (Comité Consultivo Internacional de Telegrafía y Teléfonos) se uniera a ISO.

Conocido como el Estándar Internacional ISO 7498 o bien la Recomendación CCITT X.200

Estándar de comunicación

Es una definición estructural y funcional de la red adoptada, incluye guías y reglas que se refieren a los componentes y los protocolos de comunicación que deben usarse.

También denominado “estándar de transmisión de datos”.

Debe tener en cuenta:
- ✓Medios utilizados y Equipos que pueden vincularse a una red
- ✓Establecimiento y mantenimiento de la comunicación entre nodos
- ✓Detección/Actuación/corrección ante un error
- ✓Rapidez y destino de los paquetes en una red
- ✓Diseño de fiabilidad de la red
- ✓Tipos de comunicaciones para administrar la red

Protocolo

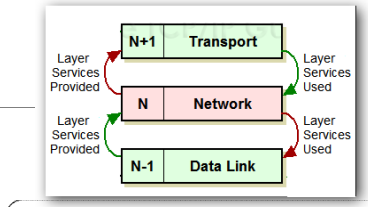
Un protocolo es un conjunto de reglas que regulan el flujo de la información, y que establecen la forma en que deben efectuarse las comunicaciones:

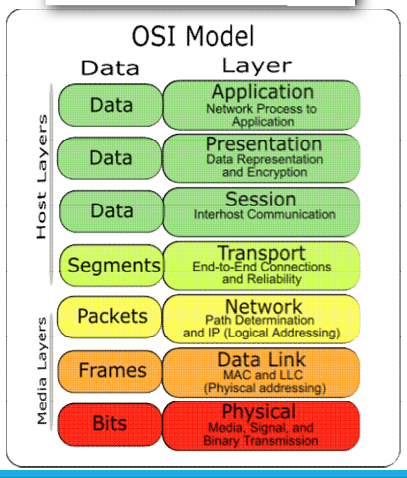
- ✓Cuál es la estructura (formato) de la información
- ✓Quién y cómo comienza el diálogo.
- ✓Quién puede transmitir en cada momento.
- ✓Cómo termina la comunicación.

Además de estas reglas de control de flujo (incluidas la temporización y la secuencia) se deben establecer mecanismos de control de detección de errores y recuperación de datos en caso de error.

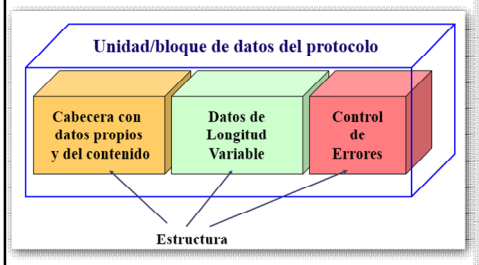
Definición del modelo OSI

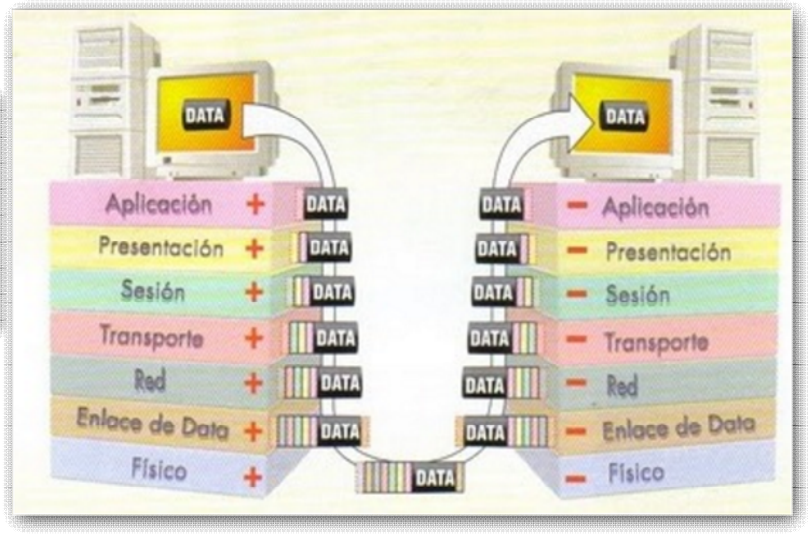
- ✓Se establecerá una capa cada vez que se necesite un nivel diferente de abstracción en el problema de la comunicación.
- ✓Cada capa debe implementar funciones bien definidas y delimitadas
- ✓Las funciones implementadas en cada capa deben seleccionarse de tal forma que permitan la definición de protocolos normalizados para su materialización
- ✓El paso de información entre capas debe ser mínimo
- ✓El número de capas del modelo debe estar equilibrado, suficiente para las diferentes funciones pero simple y comprensible



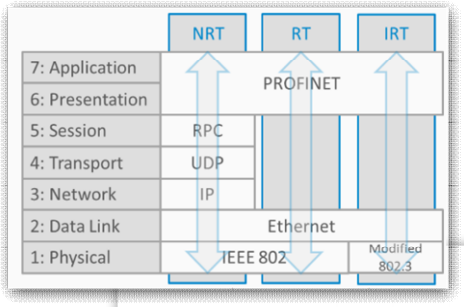


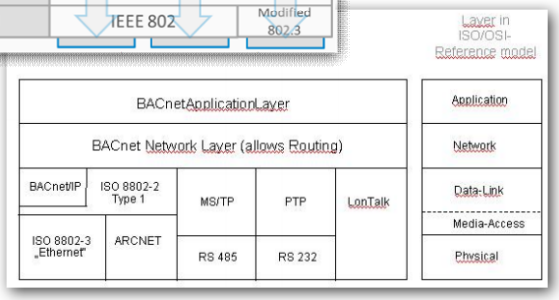
PDU y Apilamiento de protocolos

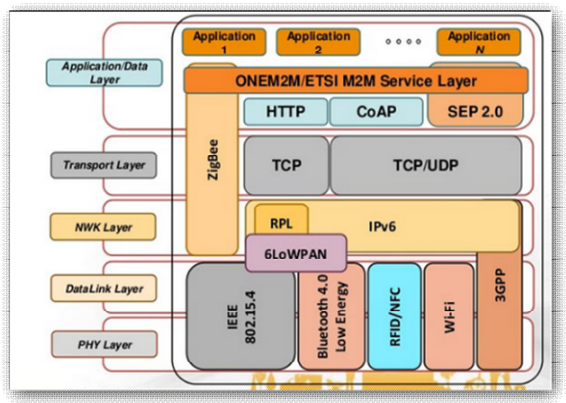




OSI como Modelo de Referencia



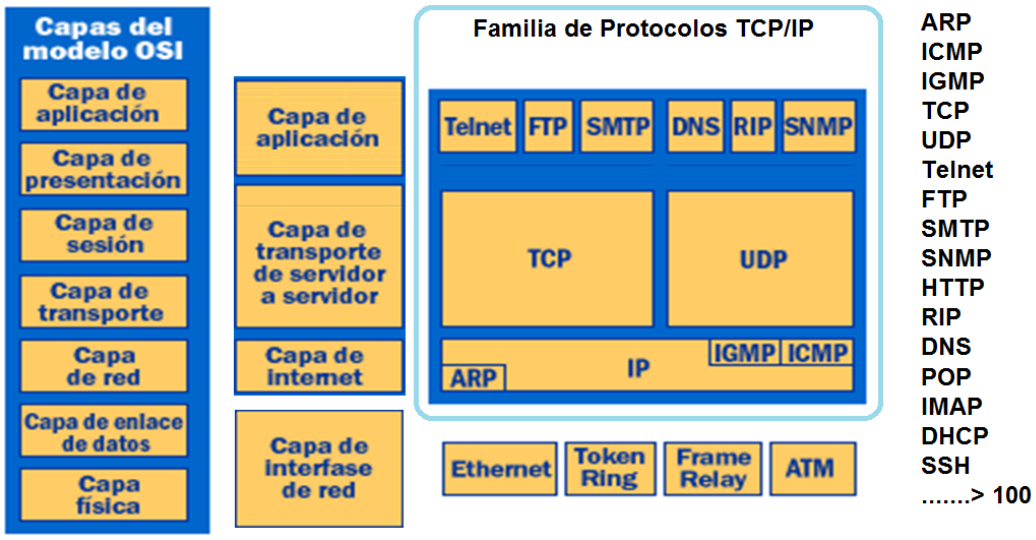




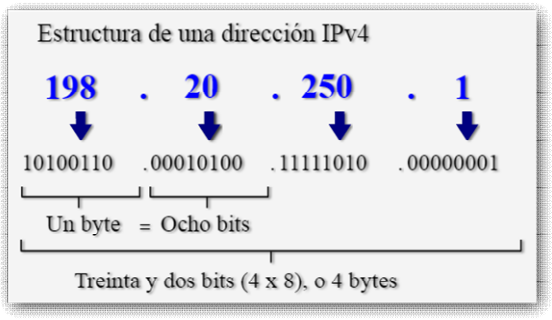
TCP/IP: Generalidades

- ✓Primeros proyectos de investigación iniciados en 1973 por DARPA (EEUU)
- ✓Evolucionan como Internetting => ARPANET => Internet
- ✓TCP/IP mantiene la red aunque la conexión se haya perdido: tiene la capacidad de sobrevivir a caídas del hardware de la subred
- ✓Posibilita conectar redes heterogéneas
- ✓Modelo creado antes de OSI (de la ISO)
- ✓TCP/IP es el estándar de Internet

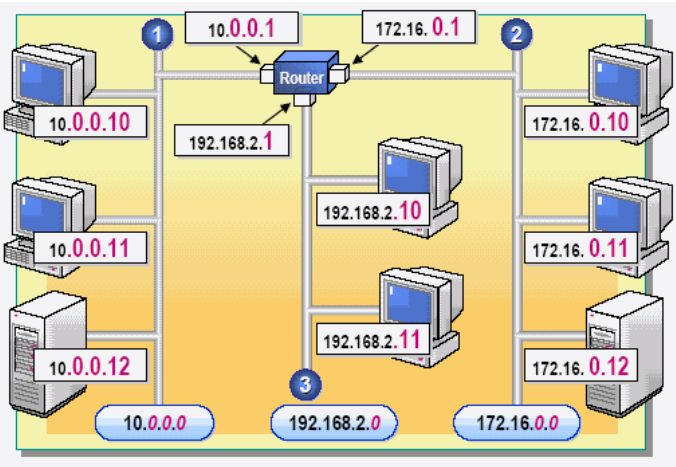
Modelos TCP/IP vs. OSI/ISO



Direccionamiento IP

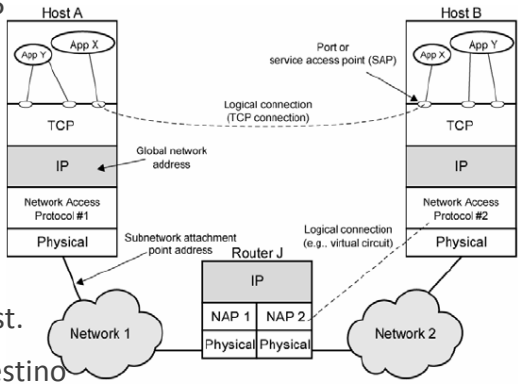


Clases	Número de Redes	Número de Nodos	Rango de Direcciones IP
A	127	16,777,215	1.0.0.0 a la 127.0.0.0
B	4095	65,535	128.0.0.0 a la 191.255.0.0
C	2,097,151	255	192.0.0.0 a la 223.255.255.0



Puertos y Sockets TCP/UDP

- ✓Puerto es un número de 16 bits (pueden definirse 65536 puertos por host)
- ✓Un puerto se asigna a una aplicación y es capaz de enviar/recibir datos por él
- ✓Se reservan los puertos de números < 1024 para TCP-IP
- ✓Los puertos superiores quedan para uso libre
  - ✓20 FTP (Canal de datos).
  - ✓21 FTP (Canal de control).
  - ✓23 Telnet.
  - ✓80 HTTP
  - ✓53 Petición de nombre DNS.
- ✓ Socket: canal de comunicaciones vía TCP entre dos host.
- ✓Socket: definido por IP+Port del origen e IP+Port del destino



HTTP

- HTTP (HyperText Transfer Protocol) es el protocolo de transferencia de hipertexto
- Es un protocolo cliente-servidor que articula los intercambios de información entre los clientes Web y los servidores http.
- Define la sintaxis, la semántica y un conjunto de reglas que utilizan los elementos software de la arquitectura Web (cliente-servidor, proxies) para comunicarse.
- Nació para la transferencia de texto. En la actualidad se usa tanto para la transferencia de hipertexto como para la transferencia de archivos y otros datos.

HTTP

Una transacción HTTP consiste básicamente en:

- Conexión
  - establecimiento de una conexión cliente-servidor. Por el puerto TCP/IP 80 es el más conocido pero se pueden usar otros no reservados.
- Solicitud
  - envío por parte del cliente de un mensaje de solicitud al servidor.
- Respuesta
  - envío por parte del servidor de una respuesta al cliente.
- Cierre
  - fin de la conexión por parte del cliente y el servidor.

HTTP es un protocolo sin estado es decir que no guarda ninguna información sobre conexiones anteriores. En la práctica esta necesidad se resuelve mediante otros mecanismos (cookies, variables de sesión de servidor, web tokens).

Ejemplo de Petición HTTP (de pagina Web)

```
GET / HTTP/1.1
Host: www.24x7linux.com
User-Agent: Mozilla/5.0 (X11; U; Linux i686; en-US; rv:1.2b) Gecko/20021016
Accept:text/xml,application/xml,application/xhtml+xml,text/html;q=0.9, text/plain;q=0.8,video/x-mng,image/png,image/jpeg,image/gif;q=0.2, text/css,*/*;q=0.1
Accept-Language: es-es, en-us;q=0.66, en;q=0.33
Accept-Encoding: gzip, deflate, compress;q=0.9
Accept-Charset: ISO-8859-15, utf-8;q=0.66, *;q=0.66
Keep-Alive: 300
Connection: keep-alive
```


Ejemplo: Respuesta HTTP del servidor

```
HTTP/1.1 200 OK
Date: Sun, 10 Nov 2002 22:50:55 GMT
Server: Apache/1.3.26 (Unix) mod_bwlimited/1.0 PHP/4.2.2 mod_log_bytes/0.3
FrontPage/5.0.2.2510 mod_ssl/2.8.9 OpenSSL/0.9.6b
Content-Type: text/html
Age: 130
Connection: close

<-- archivo index.html que contiene la página solicitada -->
```

Generalidades de XML

- ✓eXtensible Markup Language
- ✓Estándar W3C para la creación de lenguajes de “etiquetas”
  - Descripción de la información (de los datos)
  - Independiente de la aplicación
- ✓Subconjunto de SGML
- ✓Etiquetado semántico, no de estilo
- ✓Componentes
  - Elementos (Delimitados por etiquetas)
  - Atributos (Contenidos en las etiquetas)
  - Entidades (Para referenciar elementos externos)
  - Componentes avanzados (como secciones CDATA y Processing Instructions)

Normas básicas

- ✓XML es “case sensitive”
- ✓Todos los tags deben cerrarse correctamente.
- ✓Todos los elementos han de anidarse correctamente.
- ✓La primera línea es la declaración XML:
  - <?xml version=“1.0” encoding=“utf-8>
- ✓Ha de existir siempre un elemento raíz.
- ✓Los valores de los atributos deben estar delimitados por comillas dobles.
- ✓Los nombres de los elementos
  - sólo pueden empezar por una letra o guión bajo,
  - pueden solamente contener letras, dígitos, guión, guión bajo y punto,
  - no pueden empezar por las letras «xml» (o combinación equivalente).

Ejemplo de Documento XML

```xml
<?xml version="1.0" encoding="UTF-8"?>
<?xml-stylesheet type="text/css" href="DocumentoEjemplo.css"?>

<!DOCTYPE clientes SYSTEM "DocumentoEjemplo.dtd">
<clientes>
    <cliente id="AYX1234">
        <nombre>Fernando</nombre>
        <direccion>San Martin 1252. Mendoza</direccion>
        <cuenta>
            <id>1234AD</id>
            <desde>1997</desde>
            <saldo>-24,98</saldo>
        </cuenta>
    </cliente>
</clientes>
```

```xml
<?xml version="1.0">
<methodCall>
  <methodName>getUserAllDetails</methodName>
  <params>
    <param>
      <value>
        <string>alec</string>
      </value>
    </param>
  </params>
</methodCall>
```


Modelo Cliente/Servidor

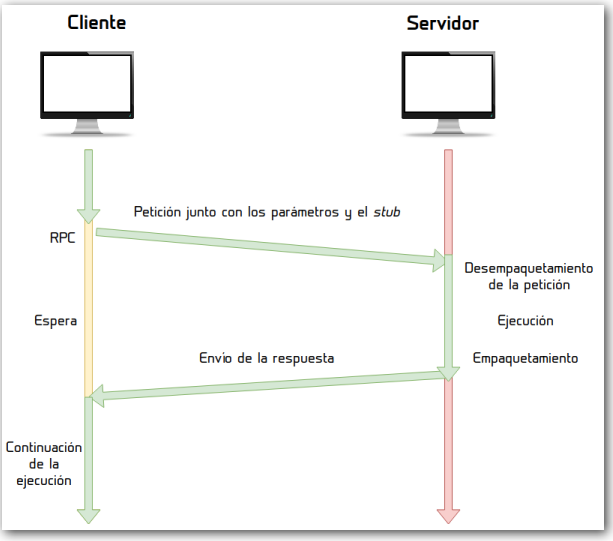
El sistema completo se descompone en pares: cliente-servidor.

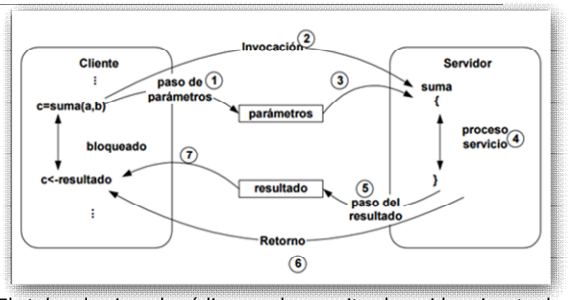
- **Servidores** (procesos u objetos) Son especializados, pocos, y proporcionan servicios. (Exporta una interfaz).
- **Clientes** (procesos u objetos) Son solicitantes de servicios. (Importa una interfaz).

Algunas características de este modelo son:

- Un cliente solicita un servicio, un servidor lo provee.
- La relación entre cliente-servidor es mediante interfaces.
  - A puede ser servidor y B puede ser cliente de un servicio X.
  - B puede ser servidor y A puede ser cliente de un servicio Y.
  - En la práctica la mayoría de los servidores son dedicados.
  - La relación cliente servidor puede ser anidada.
    - A invoca al servidor B, y B invoca al servidor C,…

RPC: Remote Procedure Call





El *stub* es la pieza de código que le permite al servidor ejecutar la tarea asignada. Provee la información necesaria para que el servidor convierta las direcciones de los parámetros que el cliente envió en direcciones de memoria válidas dentro del ambiente del servidor. La representación de datos en cliente y servidor (*big-endian* o *little-endian*) podría discrepar, el *stub* también provee la información necesaria para solucionar esta situación. De forma general, es el fragmento código encargado de enmascarar toda discrepancia entre el cliente y servidor. Es necesario que las bibliotecas de *stubs* estén instaladas tanto en el cliente como en el servidor.

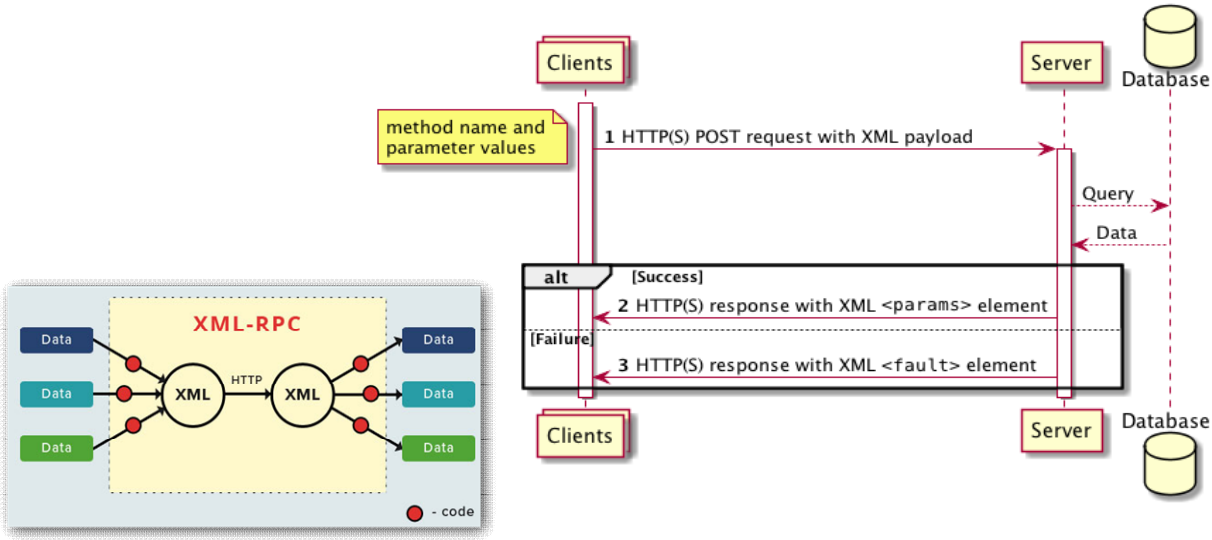
RPC: Llamada a Procedimientos Remotos

- Cualquier interfaz o servicio local puede ser potencialmente una interfaz remota:
  - Impresora: Enfila un archivo, inicia la cola, vacía la cola, etc.
  - Sistema de archivos: Crear, borrar, leer y escribir archivos.
  - Manejador de candados: Obtener un candado de lectura o escritura, liberar un candado, etc.
- Es necesario una abstracción de la comunicación para poder proporcionar servicios remotos.
  - Simple: Análogo a los llamados locales a procedimientos o llamadas al sistema.
    - Hacer una llamada a una función de dejar que la función misma se preocupe por los detalles de la comunicación.
  - Conveniente: se pueden programar como simples llamados a funciones locales.
- Los clientes llaman a los servidores.
  - Los servidores tienen un nombre o identificador único.

XML-RPC

- Protocolo de llamada a procedimiento remoto que funciona usando Internet.
- Un mensaje XML-RPC es una solicitud HTTP-POST o una respuesta, donde el cuerpo de está formateado en XML.
- El procedimiento se ejecuta en el servidor.
- El cuerpo del mensaje de una petición posee un único elemento <methodCall> que contiene un subelemento <methodName> con una cadena que representa el nombre del método que se va a invocar.
- Si el método utiliza parámetros, debe tener un subelemento <params>, con tantos <param> diferentes como parámetros tenga el método, Cada uno de los cuales posee un valor.
- Para especificar los valores de los parámetros disponemos de marcas que nos permiten especificar escalares, estructuras y listas.

XML-RPC



Tipos de datos (simples y estructurados)

- <i4> o <int>
  - entero de 4 bytes con signo
- <boolean>
  - 0 (falso) o 1 (cierto)
- <string>
  - cadena ASCII
- <double>
  - coma flotante con signo y de doble precisión
- <dateTime.iso8601>
  - día y hora en formato iso8601
- <base64>
  - binario codificado en formato base-64

```
<array>
   <data>
      <value><i4>120</i4></value>
      <value><string>Menu_A</string></value>
      <value><boolean>0</boolean></value>
      <value><i4>-31</i4></value>
   </data>
</array>

<struct>
   <member>
      <name>numero_dientes</name>
      <value><i4>18</i4></value>
   </member>
   <member>
      <name>diametro</name>
      <value><double>38.50</double></value>
   </member>
   <member>
      <name>fecha_fabricacion</name>
      <value><dateTime.iso8601>19980717T14:08:55</dateTime.iso8601></value>
   </member>
</struct>
```

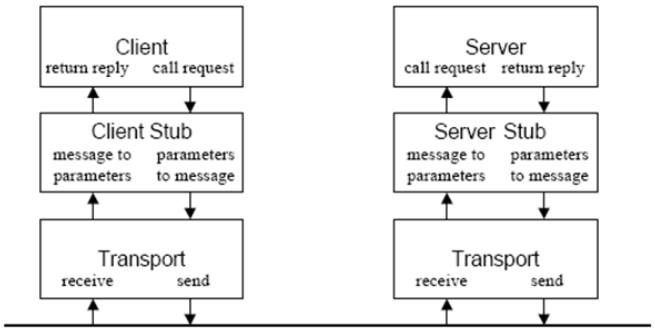

Ampliaremos detalles al trabajar sobre los aspectos prácticos.

Nos vemos en el próximo módulo!!

Por cualquier consulta mi correo es: unidatos@gmail.com

Anexo RPC

RPC: Remote Procedure Call



- Debe de haber un enlace entre el lenguaje y el sistema RPC.
- Genera un "stub" que representa el RPC en el cliente.
- El stub crea un mensaje empacado con los parámetros.
- Se bloquea.
- Los mecanismos de RPC proporcionan un despachador.
- El despachador llama al "stub" del servidor.
- El stub desempaca el mensaje y llama a un procedimiento local.
- Hace un procedimiento similar para regresar la respuesta.

Aspectos principales de RPC

Control de flujo: transferencia síncrona del control.

- El que invoca hace una solicitud y se bloquea.
- El invocado recibe la solicitud y contesta.
- El que invoca continua su flujo de ejecución.

Sintaxis de la invocación.

- RPC´s tienen parámetros de entrada y salida.
- No hay información de variables globales.
- El paso de apuntadores no tiene sentido.

Tipos de RPC´s

- Codificados en un lenguaje de programación especial (como Ada distribuido, Argus, Arjuna, Java, C++).
- Codificados en un lenguaje para definición de interfaces (como OSF/DCE, Corba/IDL, Sun/RPCGEN)

Algunos temas a resolver al diseñar RPC

Paso de parámetros y conversión de datos.

- ¿Cuáles tipos de datos pueden ser enviados y cómo se representan los mensajes?

Binding (atar o vincular).

- ¿Cómo un cliente localiza a un servidor y cómo un servidor registra sus servicios?

Manejo de fallas y excepciones

- ¿Cómo reportar errores? Los fallos potenciales son:
  - El cliente es incapaz de localizar al servidor.
  - El mensaje de petición del cliente al servidor se perdió.
  - El mensaje de respuesta del servidor al cliente se perdió.
  - El servidor falló (crashes) después de recibir una petición.
  - El cliente falló (crashes) después de enviar una petición

Seguridad

- ¿Cómo administrarla en RPC?