1 Chap0

8Fallacies

1.Network Reliable. |PowerSupply, HardDisk, NodeFailures, Configurations, Bugs| **Effect:**applicationHangs, crashes |**Countermeasures:** Redundancy HW&SW systems,middleware &application; CatchExceptions, CheckCodes, React;Retry ConnectingUponTimeouts 2.LatencyZero.|Latency:timeForData Transfer(speedOfLight)& Bandwidth:howMuchData transferred 3.Bandwidth is infinite. 4.The network is secure. 5.Topology doesn't change. 6.There is one administrator. 7.Transport cost is zero. 8.The network is homogeneous

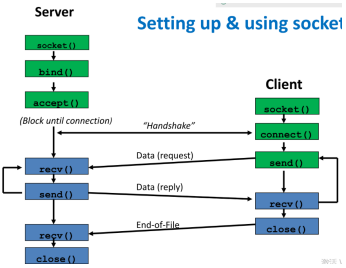
2 Chap1-CommunicationBasics

NETWORKING BASICS

ISO OSI: Open Systems Interconnection model, Basis for standards development on systems interconnection.

SOCKET

Move data/message/(invoke operation/-service and return result/failure) from Application I on Host A to Application K on Host B. **Client**:Issues requests to server(send & receive). **Server**:Starts up and listens for connections, requests, and sends/receives. **Client/Server examples**: telnet/telnetd, ftp/ftpd (sftp/sftpd), Firefox/Apache. **Socket**: network programming abstraction for communicating among processes (applications) based on (Unix) file descriptors. **File descriptor**:an integer representing an open file managed by the OS \In Unix any I/O is done by reading/writing from/to file descriptors. **Socket types**: Stream socket:java.net.ServerSocket, TCP based, Ordering guaranteed, Error-free \Datagram socket:java.net.DatagramSocket, UDP based \IPv4 & IPv6



NIO(Nonblocking sockets)

Synchronous: Single thread reading data from clients(stream) and blocked until ready(no multiple read) **Asynchronous**: Single thread reading data from clients: Thread → Channel: read data into buffer, Channel → Buffer: fill data into buffer, Thread → Buffer: check data in buffer (main thread not blocked) **Synchronous vs. Asynchronous**: S: A thread enters into action and waits until I/O is completed \Limited scalability, one thread per I/O connection(Overhead:context switching → time between diff. tasks) A: Passes the request immediatly to the OS-kernel and then do other tasks → worker thread `while (true) {` only do computation, never blocked, no context swtich **Java NIO Channels**: All IO operations can be done with channels(File, TCP, UDP) \Multiple types of channels(FileChannel (File on disk),DatagramChannel (UDP), SocketChannel (TCP, support concurrent read/write), ServerSocketChannel (TCP)) \Responsibilities(Read, write buffer)

U1 Finite state machines that describe **a communication session between a client and a server**. The first FSM represents the server and the second FSM represents the client. Both parties (client and server) keep the communication session open and exchange messages until one of them decides to close

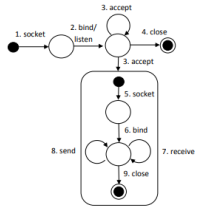


Figure 1.1: FSM for server.

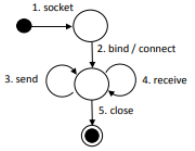
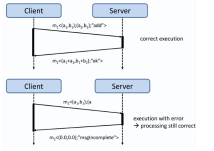


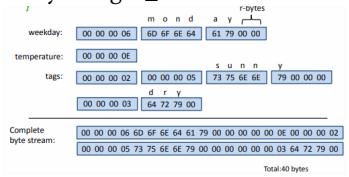
Figure 1.2: FSM for client.

it *accept* is `while loop`, detail in rectangle: create a new socket (and therad) for commu. with client **Simple protocol design** complex number as string: $c_i = (a,b)$, $op \in \{add, sub, mul, div\}$, C to S message format: $m_1 < c_1; c_2; op >$, Status: $st \in \{OK, msgIncomplete, ...\}$, S to C message format: $m_2 < C_r; st >$



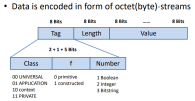
3 C2 EXTERNAL DATA REPRESENTATION, Presentation Layer

Heterogeneity HW: Diff. HW architectures store bytes:Big, Small Endian ProgrammingLanguage:Diff. PL store data types differently:AB, 0AB **Transformation between representations**: Transformation between local and remote representations|Information may lost **Two realizations**: 1.**Pairwise transformation between n local representations**(vollständigGraph,$\#n^2 - n$, Either sender or receiver has to transform) 2.**Transformation to and from canonical representation**(a single canonical C as intermediate representation|No local information about communication partner needed| $\#2 * (n-2)$, -2if canonical is one of n) **XDR** partOf NFS, OSPresentationLayer, encodes only data items, no meta information about their types +:easy, -:Receiver lost data description |exactly 32 bit integer is stored according to **big endian** +:Fixed length reduces computation. -:wasting |Data is encoded into blocks of multiples of 4: n-bytes contain data; r-bytes are used for padding with n + r mod 4 = 0 |int: int32; float=Sign+Exponent+Mantissa, String=length_int32+bytes, array=length_int32+ele

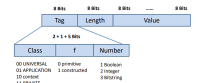



```
struct forecast:String
weekday;int temperature;
String tags<>;
```

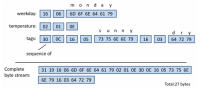
ASN.1:Abstract description of data types,telecommunication,internet protocol |Enables exchange in heterogeneous systems |**abstractSyntax** *compiler* **→ concreteSyntax Java, C++, they transfer syntax**



using encodingRules




```
Forecast::==SET{
weekday IA5String,temperature
 Interger,tags SEQUENCE OF
```



```
IS5String;
```

|encodes type information, +:receiver not need to know data description,-:additional overhead **Java object serialization,JOS** Stream-based transmission of serialized objects(Via TCP or UDP sockets) , Receiver of object needs implementation of class , Serialization does not require class specific code(Java reflection) , Class implements java.io.Serializable interface -: locked into Java(No support for heterogeneous systems), No support for versioning(If the serialized class changes, all network nodes have to be updated) **serialize:obj2bit**

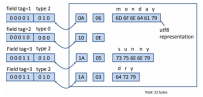
```
Socket s = new Socket("localhost"
, 8022);ObjectOutputStream
oos = new ObjectOutputStream
(s.getOutputStream()); oos
.writeObject(obj);
```

deserialize:bit2obj

```
ServerSocket ss = new
 ServerSocket(8022);Socket
 s = serverSocket.accept
();ObjectInputStream ois
= new ObjectInputStream(s.
getInputStream());obj=(Obj
)ois.readObject();
```

XMLDe facto standard for data exchange |Schema: `<xsd:element name="forecast"> <xsd:complexType > <xsd:all> <xsd:element name="weekday"type="xsd :string"/> <xsd:element name="temperature"type="xsd :integer"/> <xsd:element name="tags"><xsd:complexType> <xsd:sequence> <xsd:element name="tag"type="xsd:string"maxOccurs=ünbounded"/> </xsd:...>` |`<forecast> <weekday> monday</weekday> <temperature >14</temperature> <tags> < tag>sunny</tag> <tag>dry</ tag> </..>` **JSON** human-readable text to transmit data objects |`{" forecast":{"weekday":"monday ","temperature":14,"tags" :["sunny","dry"] }}` |+ **XML/JSON**:readable, defined as standard, JS support JSON(directly loaded Browser and deserialized) |- **XML/JSON**: verbose, badPreformance, longOverhead, slowWriteParse **ProtocolBuffersGoogle**: Similar concept like ASN.1, but not standard, efficient binary serialization, heterogeneous systems **data structures defined in .proto file(IDL) generate serialization code(Java, C#), then java, .NET projects request, response each other** |.proto:

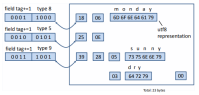
```
message
 forecast{required string
weekday =1 required int32
 temperature =2 repeated
```



```
string tags =3 }
```

|+:efficient writing/parsing,well documented,Versioning -:No RPC **Apache-Thrift:framwork, applied Hadoop and HBase** .thrift:

```
struct Forecast
{1: string weekday 2: i32
temperature 3: list<string>
```



```
tags}
```

|+:Multiple protocols to serve different purposes(binary, JSON),RPC,Open source,widely,Versioning **VariableLength:**