Wrocław University of Technology

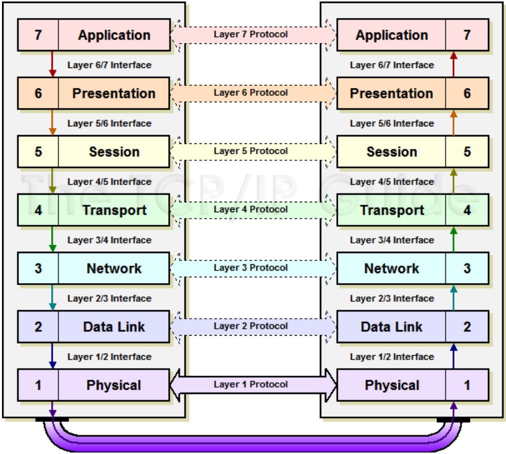
Chair of Cybernetics and Robotics

Communication protocols

Mariusz Janiak
p. 331 C-3, 71 320 26 44

Contents

OSI model



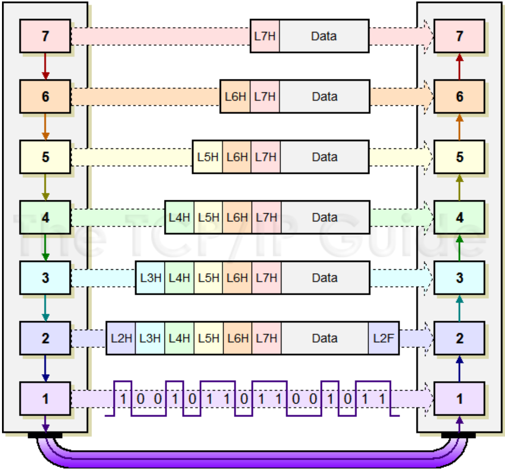
[1]

1 http://www.tcpipguide.com

OSI model

1. **Physical** (Bit) – Transmission and reception of raw bit streams over a physical medium
2. **Data link** (Bit/Frame) – Reliable transmission of data frames between two nodes connected by a physical layer
3. **Network** (Packet/Datagram) – Structuring and managing a multi-node network, including addressing, routing and traffic control
4. **Transport** (Segments) – Reliable transmission of data segments between points on a network, including segmentation, acknowledgement and multiplexing
5. **Session** (Data) – Managing communication sessions, i.e. continuous exchange of information in the form of multiple back-and-forth transmissions between two nodes
6. **Presentation** (Data)– Translation of data between a networking service and an application; including character encoding, data compression and encryption/decryption
7. **Application** (Data) – High-level APIs, including resource sharing, remote file access, directory services and virtual terminals

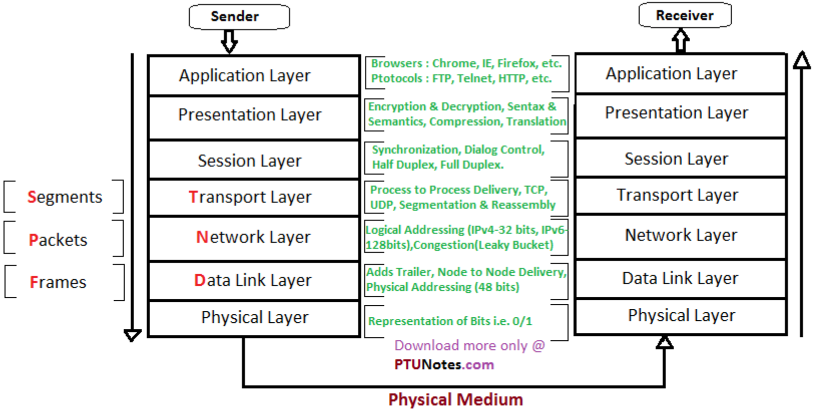
OSI model



2

2 http://www.tcpipguide.com

OSI model



3

3 http://ptunotes.com

Embedded Networking

Common wired interfaces

- RS232
- RS485/RS422
- CAN
- Ethernet
- USB
- ...

Embedded Networking

CAN Bus

- Characteristics
  - 8-byte payload
  - half-duplex
  - 11-bit identifier (CAN 2.0A) or 29-bit extended (CAN 2.0B)
  - Throughput up to 1Mb/s (to 40m)
  - Protocol overhead – $64/111 = 0.576$ best, $64/135 = 0.473$ worst
  - Hard real-time – hardware arbitration.
- Multimaster, MultiCast Protocol without Routing
- Good error detection capabilities (method for discriminating between temporary errors and permanent failures)
- Protocols: CANopen, DeviceNet, SAE J1939

Embedded Networking

Ethernet

- Characteristics
  - 1500-byte payload
  - Usually full-duplex
  - 48-bit addresses
  - High throughput – now standard is 1GB/s
  - Bad real-time properties – Carrier Sense Multiple Access with Collision Detection (CSMA/CD)
- Can be used with or without TCP or UDP
- Hubs, switches, etc. support large networks
- Real-time protocols: RTnet, EtherCAT, ProfiNet, Powerlink, Ethernet-IP, Sercos, . . .

Embedded Networking

RTnet

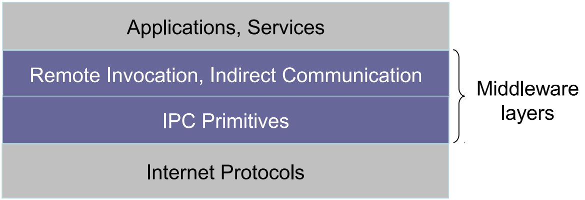
- Hard real-time network protocol stack for *Xenomai* and *RTAI*
- Open source license
- Operates on standard Ethernet hardware
- Supports several popular NIC chip sets, including Gigabit Ethernet
- Implement UDP/IP, TCP/IP (basic features), ICMP and ARP in a deterministic way
- POSIX socket API for real-time user space processes and kernel modules
- FreeRTOS port is available (RTmac, RTcfg, TDMA, Socket API, UDP/IP, ARP)

www.rtnet.org

Communication paradigms

Types of communication paradigms

- Inter-Process Communication (IPC)
- Remote Invocation
- Indirect Communication



Communication paradigms

Inter-Process Communication

- Relatively low-level support for communication
  - Direct access to internet protocols (Socket API)
- Socket is a communication end-point to which an application can write or read data
- Socket abstraction is used to send and receive messages from the transport layer of the network
- Each socket is associated with a particular type of transport protocol
  - UDP Socket – Connection-less and unreliable communication
  - TCP Socket – Connection-oriented and reliable communication

Communication paradigms

Inter-Process Communication – UDP Sockets

- UDP provides connectionless communication, with no acknowledgements or message transmission retries
- Communication mechanism
  - Server opens a UDP socket *SS* at a known port *sp*
  - Socket *SS* waits to receive a request
  - Client opens a UDP socket *CS* at a random port *cx*
  - Client socket *CS* sends a message to server IP and port *sp*
  - Server socket *SS* may send back data to *CS*

Communication paradigms

Inter-Process Communication – UDP Sockets (cont.)

- Messages may be delivered out-of-order
- Communication is not reliable
- Sender must explicitly fragment a long message into smaller chunks before transmitting (a maximum size of 548 bytes is suggested for transmission)
- Receiver should allocate a buffer that is big enough to fit the sender's message

Communication paradigms

Inter-Process Communication – TCP Sockets

- TCP provides in-order delivery, reliability and congestion control
- Communication mechanism
    - Server opens a TCP socket *SS* at a known port *sp*
    - Server waits to receive a request (*accept* call)
    - Client opens a TCP socket *CS* at a random port *cx*
    - *CS* initiates a connection initiation message to server IP and port *sp*
    - Server socket *SS* allocates a new socket *NSS* on random port *nsp* for the client
    - *CS* can send data to *NSS*

Communication paradigms

Inter-Process Communication – TCP Sockets (cont.)

- TCP Sockets ensure in-order delivery of messages
- Applications can send messages of any size
- TCP Sockets ensure reliable communication using acknowledgements and retransmissions
- Congestion control of TCP regulates sender rate, and thus prevents network overload

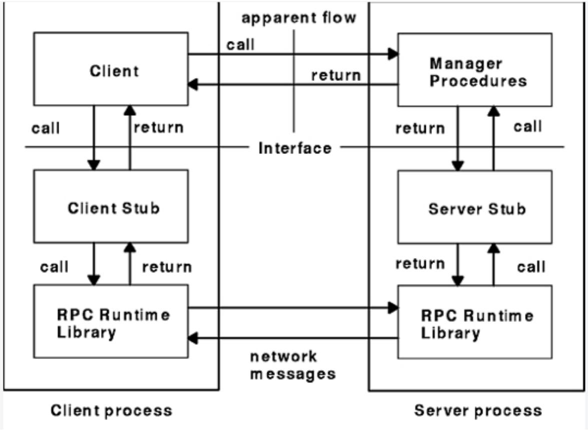
Communication paradigms

Remote Invocation

- An entity runs a procedure that typically executes on an another computer without the programmer explicitly coding the details for this remote interaction
    - A middleware layer will take care of the raw-communication
    - Programmer can transparently communicate with remote entity
- Two types of remote invocations
    - Remote Procedure Call (RPC) – Sun's RPC (ONC RPC), XML/RPC
    - Remote Method Invocation (RMI) — CORBA, Java RMI
- RMI strongly resembles RPC but in a world of distributed objects

Communication paradigms

Remote Invocation (cont.)



[4]

4 http://blog.facilelogin.com/2011/02/rpc-with-java.html

Communication paradigms

Remote Invocation (cont.)

- RPC enables a sender to communicate with a receiver using a simple procedure call – no communication or message-passing is visible to the programmer
- Parameter passing via Marshaling – procedure parameters and results have to be transferred over the network as bits
- Data representation – data representation has to be uniform
- Passing Parameters by: value, reference (only RMI)

Communication paradigms

Remote Invocation (cont.)

- Synchronous vs Asynchronous
  - Synchronous blocks the client until the server returns – blocking wastes resources at the client
  - Asynchronous are used if the client does not need the result from server – server immediately sends an ACK back to client
  - deferred synchronous RPCs – client triggers an asynchronous RPC on server, on completion, server calls-back client to deliver the results

Communication paradigms

Remote Invocation (cont.)

- Space Coupling – where the procedure resides should be known in advance
- Time Coupling – on the receiver, a process should be explicitly waiting to accept requests for procedure calls
- Lack robustness – cascading points of failure
- Typically built on top of TCP – impacts scalability and time-determinism
- Best-suited to smaller, closely-coupled systems

Communication paradigms

Indirect Communication

- Indirect communication uses middleware to
  - Provide one-to-many communication
  - Some mechanisms eliminate space and time coupling
    - Sender and receiver do not need to know each other's identities
    - Sender and receiver need not be explicitly listening to communicate
- Supports a variety of transports including multicast UDP
- Better suited for high-performance and real-time
- Types of indirect communication
  - Group communication
  - Publish-subscribe
  - Data-Distribution

Communication paradigms

Indirect Communication – Group communication

- One-to-many communication – Multicast communication
- Abstraction of a group
  - Group is represented in the system by a *groupId*
  - Recipients join the group
  - A sender sends a message to the group which is received by all the recipients
- Middleware services
  - Group membership
  - Handling the failure of one or more group members
- Efficient use of bandwidth
- Identity of the group members need not be available at all nodes
- Time coupling – data are not buffered

Communication paradigms

Indirect Communication – Publish-subscribe

- Message-based, anonymous communication
  - Message sent to Topic
  - Multiple readers can subscribe to Topic with or without filters
  - Each message delivered to all subscribers that pass filter
- Participants are decoupled
  - in space: no need to be connected or even know each other
  - in flow: no need to be synchronized
  - in time: no need to be up at the same time
- Large number of producers distribute information to large number of consumers
- Good solution for highly dynamic, decentralized systems

Communication paradigms

Indirect Communication – Publish-subscribe (cont.)

- Only messages no concept of data
- Each message is interpreted without context
- Messages must be delivered FIFO or according to some "priority" attribute
- No Caching of data
- Simple QoS: filters, durability, lifespan

Communication paradigms

Indirect Communication – Data distribution

- Provides a *Global Data Space* that is accessible to all interested applications
  - Data objects addressed by Domain, Topic and Key
  - Subscriptions are decoupled from Publications
  - Contracts established by means of QoS
  - Automatic discovery and configuration
- Subsystems are decoupled in time, space, and synchronization
- Messages represent update to data-objects
- Data-Objects identify by a key
- Middleware maintains state of each object
- Objects are cached. Applications can read at leisure
- Smart QoS: Ownership, History, Deadline

Communication paradigms

Network frameworks

- Adaptive Communication Environment (ACE)
  http://www.dre.vanderbilt.edu/~schmidt/ACE.html
- ZeroC Ice
  https://zeroc.com
- ZeroMQ
  http://zeromq.org/
- OpenDDS
  http://www.opendds.org
- RTI Connext DDS
  https://www.rti.com
- . . .

Data representation and marshalling

- Heterogeneity (everybody is different)
  - Different operation systems
  - Different programming languages
  - Different hardware architectures
- Problem when data structures must be sent in message in distribueted system

Data representation and marshalling

- Different hosts may use different data representations
  - Sizes of integers, floating points, characters (ASCII vs Unicode)
  - Big vs. Little endian
  - Data structure layout in memory
    - Padding of arrays and structs
  - Pointers and structured data
    - Pointer representation might differ
    - Trees, lists, etc. must be serialized
  - Objects and functions (contain code!)
    - Typically don't transmit code

Data representation and marshalling

- *Marshalling* is the process of taking a collection of data items and assembling them into a form suitable for transmission
- *Unmarshalling* is the process of disassembling them on arrival to produce an equivalent collection of data items at the destination
- Strategy 1: Receiver Makes Right
  - Send data in sender's native form
  - Receiver fixes it up if necessary
- Strategy 2: Canonical Intermediate Representation
  - Sender marshals data into common format
  - Receiver unmarshals data into its own format
- *External data representation* is an agreed standard for the representation of data structures and primitive values

Data representation and marshalling

Data Schema

- Explicit typing – self-describing data (tags)
  - additional information added to message to help in decoding
  - e.g., ONC XDR (RFC 4506)
- Implicit: typing – the code at both ends "knows" how to decode the message
  - interoperability depends on well defined protocol specification
  - very difficult to change
  - e.g., ISO's ASN.1, XML, protocol buffers, JSON

Data representation and marshalling

Data Representation and Marshalling frameworks

- Sun XDR (representation of most used data types)
- ASN.1/BER (ISO standard, based on "type-tags", open)
- CDR (used in CORBA RMI, binary layout of IDL types)
- Java Object Serialization (JOS)
- XML (used in SOAP: "RMI" protocol for Web Services)
- Google Protocol Buffers
- Apache Thrift
- Boost Serialization
- MessagePack (CMP – C without heap allocation)
- Lightweight Communications and Marshalling (LCM)
- ...

The End

Thank you for your kind attention.