MetaComputing

MetaComputing / Distributed Computing Communications

CS 594 Spring 2001

Thanks to Mark Baker for the use of this material.

MetaComputing / Distributed Computing

- For two or more MetaComputing entities (compute nodes, MPPs, data, real-time instruments, printers, scanners) to interact they need 3 basic items:
  - **Communications**
  - **Naming of entities**
  - **Interfaces/Protocols** (or just standard methods for now)
    - Note than in Message Passing systems we only needed 2 out of the 3. In MetaComputing the extra one is due to the Heterogeneous nature of the system involved.

Communications

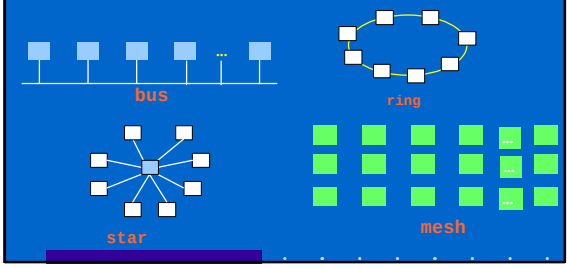
- What's a computer network?
  - A set of nodes connected by communication links.
  - A set of networks connected by nodes.
- Nodes can be general-purpose computers:
  - Workstations.
  - Servers.
- Nodes can also be more specialised systems:
  - Routers.
  - Switches.

Communications

- Communication Requirements
  - *Connectivity* - Nodes communicating via links.
  - *Recursive connectivity* - Networks communicating via gateways.
  - *Addressing* - Identifying a node.
  - *Routing* - Forwarding messages to a node.
  - *Broadcast and Multicast* - Sending messages to multiple destinations at once.
  - *Multiplexing* - Multiple parties sharing a link.
  - *Reliability* - Recovering from errors.
  - *Performance* - Low latency/high bandwidth.

Communications

- Networks can have different topologies



Communications

- No matter what the physical topology the system using the network will want to be able to define its own view of the network
  - Logical vs Physical
    - Network Layer / Environment / SW view
  - I.e. MPI topology calls
  - Ethernet broadcast vs the internet MBONE tree

Communications

- Issues in networking (low-level)
  - Error Control - recovery.
  - Multiplexing and de-multiplexing - joining multiple data streams, or splitting one stream into multiple.
  - Fragmentation - divide data into network-size pieces.
  - Addressing - identifying network entities.
  - Routing - finding a path through the network.
  - Flow control - avoid overloading slower entity.
  - Congestion control - avoid overloading network.

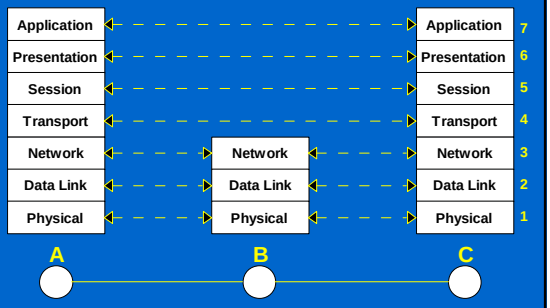
Communications

- A complex problem can be simplified by layering:
  - Layer N relies on services provided by layer N-1.
  - Layer N provides service to layer N+1.
- Interfaces between layers completely define services - an API.
- Layers hide complexity:
  - Service provided by layer is independent of layer's implementation.
  - Changes inside layer do not affect other layers.

Communications

- Protocols define the interface and interactions between entities
  - Applications don't worry about protocols, just the immediate interface to the next layer down.
- What is a networking protocol?
  - A set of rules for how network elements communicate.
  - Specifies the format of messages exchanged.
  - Specifies actions to take on receipt of message.
- Protocol specifications must be exact

Communications



Communication Layers

- Application layer
  - Provides process-to-process communication.
  - All other layers exist to support this layer.
  - Examples: Email, FTP, Telnet, WWW...
- Presentation layer
  - Converts data to a common format.
    - Little-endian v. big-endian byte orders.
    - Size of data structures.

Communication Layers

- Session layer
  - Binds two transport streams into a "relationship" - e.g., audio + video.
  - Performs authentication during session setup.
- Transport layer
  - End-to-end communication.
  - Reliable in-order delivery.
  - Multiplexes higher-level streams.
  - Flow and congestion control.

Communication Layers

- Network layer
  - Accepts packets from upper layers and injects them into network.
  - Specifies host addresses.
  - Implements packet routing.
  - Controls congestion - discards packets if needed.

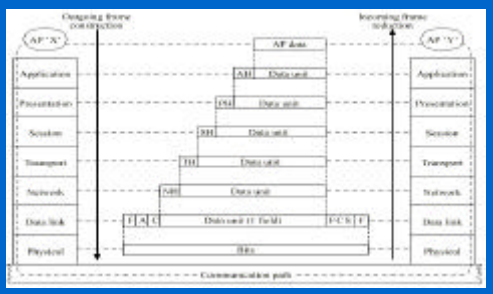
Communication Layers

- Data link layer
  - Provides point-to-point error-free communication over a single link.
  - Matches speed between sender and receiver.
  - Framing and error control.
  - Media Access Control (MAC).
- Physical layer
  - Communicates raw bits.
  - Deals with voltage levels, frequencies, ...
  - Usually the realm of electrical engineers...

Protocol Data Units (PDUs)

- Send side
  - Layer N takes PDU from layer N+1.
  - Adds its own fields (header!!) to form a new PDU.
  - Passes PDU to layer N-1.
- Receive side
  - Layer N takes PDU from layer N-1.
  - Strips and processes its own fields.
  - Passes PDU to layer N+1.

PDUs



Layering harmful?

- A powerful technique for structuring systems - but religious adherence has its problems.
- Different layers may duplicate functionality
  - Multiplexing at transport, network, and data link.
  - Fragmentation at transport and network.
  - Error recovery at transport and data link.
  - Congestion control at transport and data link.

Layering harmful?

- Different layers may need access to same information - maximum transmission unit, timestamps.
- Must compromise between modularity and performance.
  - Compare PVM internals
  - MPI-CH
  - Nexus

Layering

- How to avoid the overheads?
  - Direct access to the network hardware ?
    - Memory mapped hardware
      - Elan chip set (Meiko)
      - Paragon and NX vs NX2

Layering

- How do we handle multiple interfaces to multiple hardware networks on the same machine?
  - Which layer does the hard work?
    - User API, environment, OS?
  - Naming (interfaces, services (SAPs), messages or just end-points)
  - who handles the errors?!
  - What about really fast interfaces (eui, fm, VIA, VBE)
  - IBM SP2
    - css0 vs en0 (swXnY vs fXnY)

Application Layer Protocols

- A number of standard application-layer protocols
  - File Transfer Protocol (**FTP**).
  - Simple Mail Transfer Protocol (**SMTP**).
  - Network News Transfer Protocol (**NNTP**).
  - HyperText Transfer Protocol (**HTTP**).
- You can also roll your own…
  - If you want a good metacomputing system.. then yes you need too.. (see PACX)

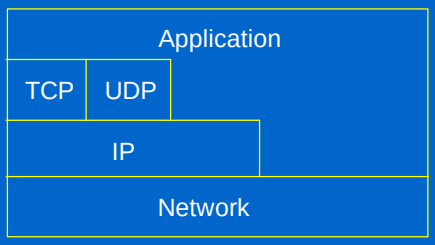
Transport layers

- Transmission Control Protocol (TCP):
  - Provides reliable byte stream service.
  - Header fields: sequence numbers, port numbers, checksum.
- User Datagram Protocol (UDP):
  - Provides unreliable unordered datagram service.
  - Header fields: port numbers, checksum.

Transport layers

- Internet Protocol (IP):
  - The key piece of the architecture.
  - Can use many different link layers.
  - Treats each network in Internetwork as a link.
  - No quality of service (QoS) guarantees.
  - Can lose packets and misorder packets.
  - Header fields: host addresses, hop counts.
    - Note - no port number.

Typical Application Comm Layers



Flow and Congestion Control

- The TCP backoff algorithm, which employs rapid rate reduction on packet loss and slow rate increase in the absence of packet loss, is very effective for data transmission, as in the data intensive distributed supercomputing applications.
- It allows many TCP connections to share the network fairly an efficiently without explicit co-ordination.

Flow and Congestion Control

- TCP strategy is not appropriate for data streaming (multimedia), where the transmission rate is relatively high but predictable and constant.
- The TCP backoff strategy can be squeezed out by rogue connections that do not reduce their transmission rate in the presence of congestion and packet loss.
  - See PACX and its custom UDP protocol.

Flow and Congestion Control

- Protocols like TCP do not provide all the resources required to meet MetaComputing needs:
  - No guarantees of QoS
    - Unlike ATM for example
  - No explicit call backs when things start to fail
    - Due to no QoS metrics
    - Network Weather Service (NWS) can provide information on link(s) characteristics
    - SNIPE can switch links automatically for you.

Performance Metrics

- Critical performance metrics for networks protocols and for the applications are throughput and jitter.
- *Throughput* is the number of user data bits communicated per second.
- *Latency* is the time from message start from the source to message arrival at the destination.
- *Jitter* is the variance in the latency.

Performance Metrics

- The inadequate performance of network infrastructure has led to a focus of maximising network throughput - which was a bottleneck.
- As throughput has improved, attention has shifted to latency and jitter.
- It is possible that a protocol that increase throughput can increase latency.
- Long latencies are undesirable for some applications, such as for distributed applications with high data / course grain parallelism.

Performance Metrics

- Protocols have been developed for distributed applications, with mechanisms to reduce the latency caused by the protocol stack and the operating system.
- Jitter is highly undesirable for audio, video and multimedia applications, but ok for most MetaComputing applications.

Protocols

- Traditional network protocols supported the movement of bits from source to destination.
  - E.g. ftp
- Communications media are improving quite rapidly - becoming cheaper, faster, more flexible, dependable and ubiquitous.
- Network protocols of the future will be designed specifically to support the applications, and the needs of the applications will determine the communication services provided.
  - I.e. I need 100MB/Sec sequenced, reliable comms for 2 minutes at 02:00 GMT between points A and B.

Protocols

- The classical network protocol provides a point-to-point, sender initiated, data transfer service, delivering messages in the order in which they were sent with good reliability and with throughput as the measure of performance.
- However, each application has its own reasons for communicating and its own types of information to be communicated - this may impose very different requirements on the network protocols.

What different application systems need

- MultiMedia teleconf applications require;
  - Data streaming protocols for audio and video, often multi-cast at high rates.
  - Reliable data transport protocols for collaboration, again often multi-cast.
  - Co-ordinated control protocols and membership protocols for collaboration.
  - Protocols for the integration of complex, independently developed modules.

What different application systems need

- Data-intensive applications require:
  - Data transport protocols for the efficient, rapid, and reliable transport of huge data sets.
    - Boundary conditions, delta values, checkpoints..
  - Protocols for the integration of complex, independently developed modules.
    - Not all wrote in the same MPI implementation or under the same F90/HPF compiler or Corba IDL (… IIOP2 ?)
  - Protocols for the encapsulation and integration of existing modules and movement of program code to remote sites where the data is located.
    - Legion and SNIPE has good facilities for this. Netsolve will soon, as will HARNESS.

What different application systems need

- Data-intensive distributed applications require:
  - Low-latency reliable transport protocols, sometimes multi-cast.
    - PVM internals did not provide this and hense did not scale very well.
  - Low-latency control and synchronisation protocols that are scalable to quite large numbers of nodes.
    - Group communications is required to allow replication of services and state.
  - Projects for better group communications include MAGPIE from Amsterdam University.

Classes of protocols

- Despite the multitude of network protocols that have been proposed, a few major classes of protocols are emerging:
  - Data transport protocols.
  - Streaming protocols.
  - Group communication protocols.
  - Distributed object proposals.

Data Transfer Protocol

- The DTP are the work-horse of the internet and exemplified by Transmission Control Protocol (TCP).
- They typically provide reliable, source ordered delivery of messages with flow control and buffer management.
- Most DTP are point-to-point, but multi-cast protocols are becoming more common.

Streaming Protocols

- The streaming protocols are typically used for audio, video, multi-media, and for instrumentation applications.
- The applications do not require reliable delivery and can accept message loss, particularly if selective loss algorithms are used.
- Multi-casting is common in these applications, but unreliable source ordered delivery suffices.

Streaming Protocols

- Even when the traffic is bursty, flow control is ineffective and bandwidth reservation is preferable.
- Low jitter is important.
- Unfortunately the datagrams orientation of the Internet does not yet support streaming protocols well.

Group Communications

- What distinguishes group communication protocols from data transport and streaming protocols is that group communication protocols are concerned with more than just movement of data.
- In a distributed system, to permit effective co-operation, and to provide fault tolerance, a group of several processes must keep copies of the same data and the copies of the data must be consistent as the application executes.

Group Communications

- Maintaining consistency is difficult for the application programmer, particularly in the presence of faults.
- Group communication protocols assist the application programmers in maintaining the consistency of replicated data by maintaining the memberships of process groups and by multi-casting messages to those process groups.

Consistency

- In a distributed system, processing and data may be replicated for increased reliability or availability or faster access.
- In particular, several processes may perform different tasks of the application and may each have a copy of the same data so that they co-operate.
- Alternatively, several processes may be replicated for fault tolerance with each replica performing the same task of the application and holding a copy of the same data.
- In both cases, the copies of the replicated data must be kept consistent as the application executes.

Consistency and group comms

- Group communications help maintain the consistency of replicated data.
  - But the design of such protocols maybe be insufficient to guarantee complete consistency of state. I.e. a 2 phase commit algorithm might also be needed.
- Such protocols distribute update messages to all the processes holding copies of the replicated data using a multicast service.
- They provide a reliable, totally ordered message delivery service, which ensures that each process performs the same sequence of updates in the same order.

Consistency and group comms

- Because of their objective to support fault tolerance, these protocols are very robust and handle a wide range of faults including processor and network partitioning.
- They include fault detection, group membership, and recovery protocols that provide a consistent view of faults and membership changes system-wide.

Consistency and group comms

- Early group communications were inefficient, but performance has improved.
  - MBONE vs GLOBE communications.
  - For example the LMP (Lightweight Multicast Protocol).
- Current group communications operate very efficiently in LAN environments - efficient Internet-based protocols are still being developed.
  - Harness is based on multiple tree structures where the root of each is part of a multi-ring.

Consistency protocols

- *Purely computational* - distributed supercomputing, where tasks might be allocated to processors by a distributed scheduler, co-ordinated by a group communications protocol.
- *Mediate human interaction* - collaborative virtual environments, where the group communications protocol can ensure that all users see the same updates of the shared workspace.
- *Control of physical devices* - where different processes control different parts of the instrumentation, co-ordinated by the group communications protocol.
- Or a ***mix*** as will be the case in future metacomputing applications.
  - See the CAVE examples from the Globus home page.

Distributed Object Protocols (DOP)

- A new area of protocol development is DOP for heterogeneous distributed systems, exemplified by the Internet Inter-Orb Protocol (IIOP) developed by the Object Management Group (OMG) for the Common Object Request Broker Architecture (CORBA).
- CORBA supports the use of existing legacy codes, provides inter-operability across diverse platforms, and hides the distributed nature of the computation as well as location of objects from applications.

Distributed Object Protocols (DOP)

- CORBA and IIOP are well-suited to the distributed collaborative virtual environments and real-time instrumentation applications, which involve the integration of complex distributed systems from existing software.
- Also important is Java's Remote Method Invocation (RMI) - this supports the movement of code to remote machines.

Future Protocols

- Network protocols are critical to the future of computing.
- Almost all computing in the future will be distributed; the network protocols provide the glue that holds the computing together.
- Much of the computing of the future will involve the integration of components developed independently, and use these components in ways that could not have been foreseen by their developers.
- Such integration is only possible with simple, well-defined, stable interfaces; such characteristics have been achieved by many network protocols but by little else in computing.
  - For simple, see Jini from Sun.