Network Interface Architectures
(See other document for figures)

- Networks are becoming the canonical I/O device
  - Trend fostered by:
    - Splintering of machines into their components (wireless peripherals, etc)
    - Expansion of embedded devices (networked toasters?)
  - In a fashion analogous to the change in trends when an entire processor could be put on a chip, proliferation of network devices has increased since the introduction of single chip network devices.
  - Internet architectures are all about networks (Fig 1)
  - Challenges of network architecture: juggling of events and data movement

- Structure of a Network System (Fig 2)
  - Design issues:
    - How is communication integrated into the processing and storage functions of the node architecture?
    - What is the communication abstraction? (OS calls / Load/Store / PIO)
    - How does data get from place to place?

- Network performance
  - Components of latency:
    - Speed of light
    - Send / Receive overhead (in the thousands of instructions range)
    - Network congestion
    - Yields a total transit time of
      Send overhead + time of flight + size/BW +Receive overhead
  - Effective Bandwidth = Bandwidth * Transfer time / Total time
  - Max stated bandwidth intentionally unrealizable – bus headroom.
  - Increased bandwidth decreases efficiency due to fixed overhead, so make packets bigger on wider pipes.
  - Even assuming total reliability, still need to consider saturation and backpressure.

- Abstractions
  - Encapsulation is the most basic abstraction; higher level objects are passed within lower level envelopes. However, size differences between levels can lead to segmentation and/or fragmentation, which must be dealt with in the network stack.
  - Digital signaling
    - Signal as encoded bitstream. (i.e. Manchester for Ethernet)
    - Information is serialized via framing.

- Routing Styles
  - Destination based: At each step, send packet to a node closer to the destination, or to a higher level node believed to know where to find such a closer node.
  - Arithmetic: Given a grid structure for routing, follow Manhattan distance path.

- Virtual circuit: Pre-build routing table (swizzle)
- Source based routing: For all source/destination tuples, know route, and include route in packet.

- Parallel Machine Networks (Fig 4)
  - Concurrency intensive, in contrast to processor intensive
  - Challenges focus on moving data
  - Interconnect takes advantage of regular topology to have required low latency, high bandwidth, high reliability network.
  - 3 classes:

Message passers	**Shared physical memory**	**Local physical memory**
Dataflow	Alewife	IPSC
Iwarp	Dash	Ncube
J Machine	Flash	IBMsp
M Machine		Clusters
*-T	*Many*	
Few commercial machines		*Easiest*

- Switches and Routing
  - Coarsest view: When travelling through the net cloud, at each step somehow get closer.
    - Recognize receipt of packet
    - Select output port
      - For grid/butterfly routing, can be derived arithmetically (1 bit changes in butterfly, etc)
      - For source based routing, choose based on included route.
      - Otherwise, look up in destination table or virtual circuit.
  - Layer 7 switch
    - With sufficient processing power, one can examine the content of packets in flight to deliver more relevant routing. This somewhat breaks encapsulation abstraction barriers, but could lead to increased performance.
  - Crossbar vs. SRAM Switch
    - Tradeoff between physical and virtual parallelism
    - Bigger buffers – smarter scheduling?

- Simple strategies (Fig 5, Fig 6)
  - Simple DMA based networking (Ex. nCube, iPSC/2)
    - Takes only a few instructions to start a transfer and framing is simple.
    - However, knowledge of packet content is lacking, and data transferred this way will not automatically arrive in the desired location, namely it must be copied or swapped out of kernel space into user space.
  - Simple Programmed IO (ex. CM5)
    - Fast small messages
    - Net queues in user space

        - Processor intensive – polling vs. interrupt handling.

- Physical parallelism (Intel Paragon) (Fig 7)
    - Intuition: many computational tasks consist of logically concurrent activities in a single loop of code. Implementation: map logical parallelism to physical parallelism.
    - Use programmed IO to build packets.
    - Message built in processor cache and then dragged to Network processor cache on a miss.
    - Downside: 2-3 times number of bus crossings/byte
    - Concurrency intensive – must deal with:
        - Begin/complete
        - DMA complete
        - Status from network

- Virtual parallelism (Meiko CS-2) (Fig 8)
    - 1 processor doing multithreaded emulation of multiprocessor
    - Dedicated network processor that can drive from Memory to Net
    - To build a message, send data wherein the first segment of data is a sequence of instructions that act upon the remaining data to build the packet.

- Stanford Flash
    - See Fig 9