COT 4600 Operating Systems Fall 2009

Dan C. Marinescu

Office: HEC 439 B

Office hours: Tu-Th 3:00-4:00 PM

Lecture 6

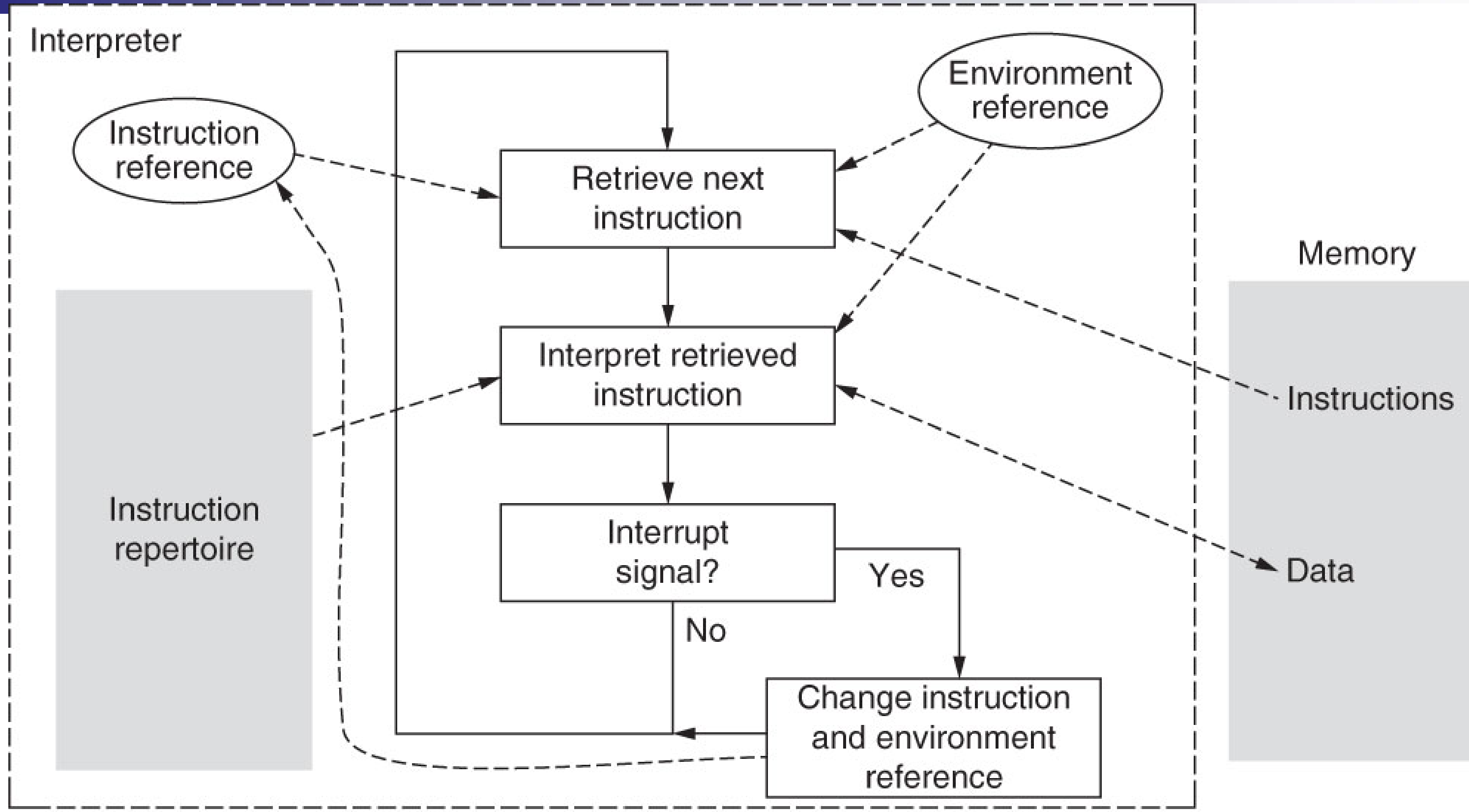
- Last time:
  - Names and the basic abstractions
  - 1. Storage
- Today:
  - 2. Interpreters
  - 3. Communication Links
  - Internet or what is behind the abstractions…
- Next Time
  - Naming in computing systems

Interpreters

- The active elements of a computer system
- Diverse
  - Hardware → Processor, Disk Controller, Display controller
  - Software →
    - script language: Javascropt, Pearl, Python
    - text processing systems: Latex, Tex, Word
    - browser : Safari, Google Chrome,Thunderbird
- All share three major abstractions/components:
  - Instruction reference ➔ tells the system where to find the next instruction
  - Repertoire ➔ the set of actions (instructions) the interpreter is able to perform
  - Environment reference ➔ tells the interpreter where to find the its environment, the state in which it should be to execute the next instruction

An abstract interpreter

- The three elements allow us to describe the functioning of an interpreter regardless of its physical realization.
- Interrupt ➔ mechanism allowing an interpreter to deal with the transfer of control. Once an instruction is executed the control is passed to an interrupt handler which may change the environment for the next instruction.
- More than a single interpreter may be present.




```
procedure INTERPRET()
    do forever
        instruction ← READ (instruction_reference)
        perform instruction in the context of environment_reference
        if interrupt_signal = TRUE then
            instruction_reference ← entry point of INTERRUPT_HANDLER
            environment_reference ← environment ref of INTERRUPT_HANDLER
```

Figure 2.5 from the textbook

Processors

- Can execute instructions from a specific instruction set
- Architecture
  - PC, IR, SP, GPR, ALU, FPR, FPU
  - State is saved on a stack by the interrupt handler to transfer control to a different virtual processor, thread.

Interpreters are organized in layers

- Each layer issues instructions/requests for the next.
- A lower layer generally carries out multiple instruction for each request from the upper layer.

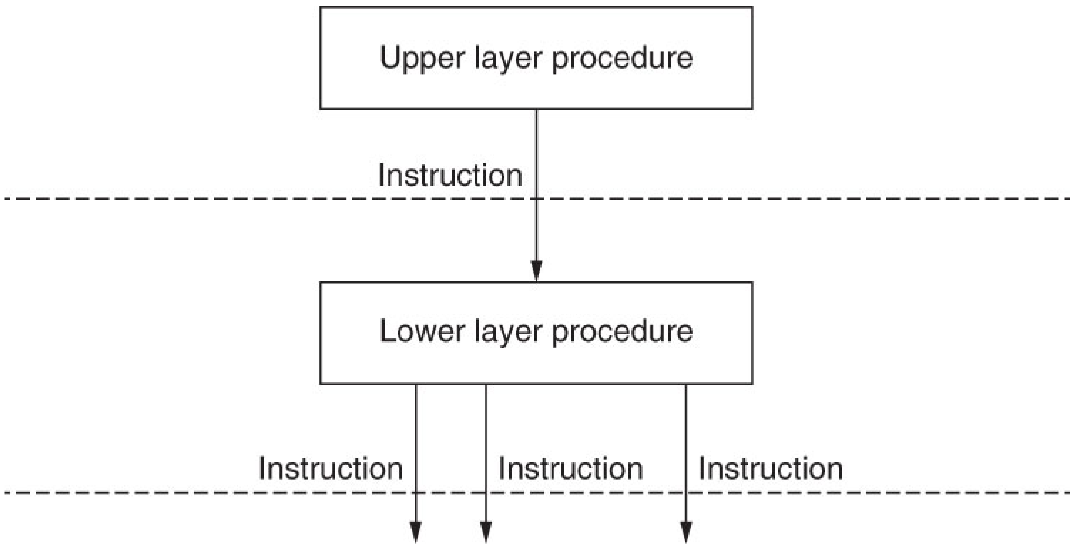


Figure 2.6 from the textbook

Example – a calendar management program

- Top layer a Java program with the following components:
  - The instruction reference ➔ get the information provided by the keyboard and mouse and interpret them
  - The repertoire ➔ add an event, delete an event, etc.
  - The environment ➔ the files holding the current calendar
- Next layer ➔ JVM which interprets the program
  - The instruction reference: next bytecode instruction
  - JVM instructions
  - The environment: IR, PC, etc
- Bottom layer ➔ the computer the JVM is running on

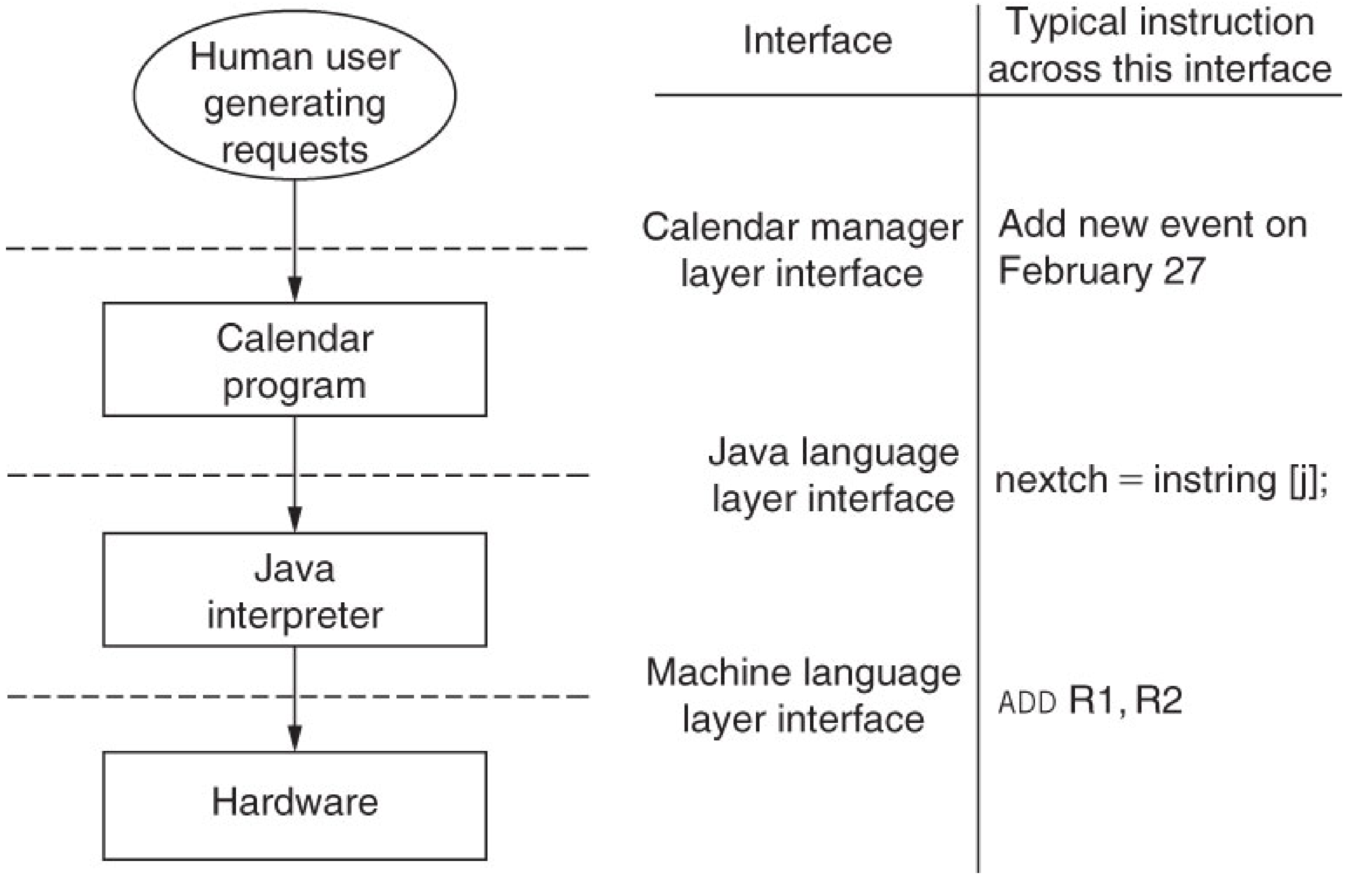


Figure 2.7 from the textbook

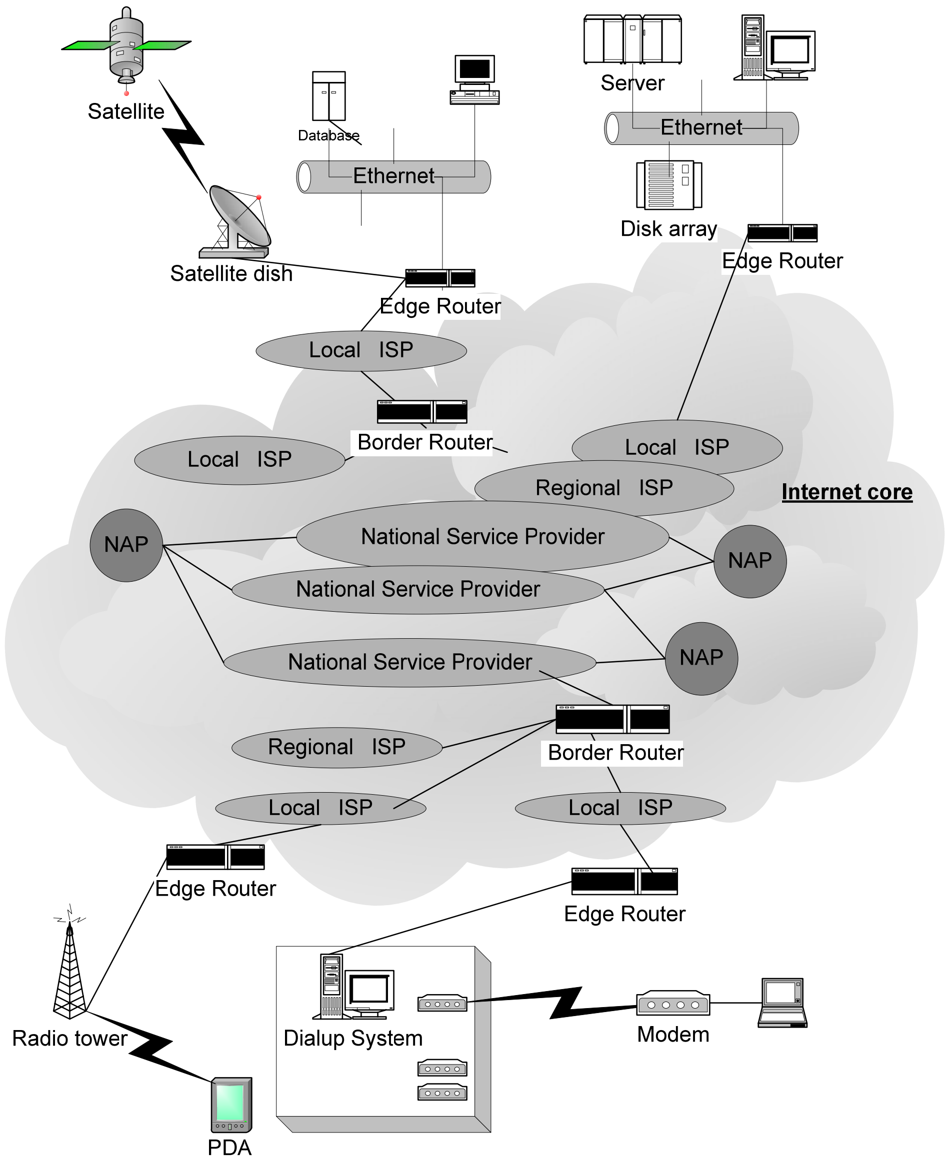
Communication Links

- Two operations
  - SEND (link_name, outgoing_message_Buffer)
  - RECEIVE (link_name, incoming_message_Buffer)
- Message ➔ an array of bits
- Physical implementation in hardware
  - Wires
  - Networks
    - Ethernet
    - Internet
    - The phone system

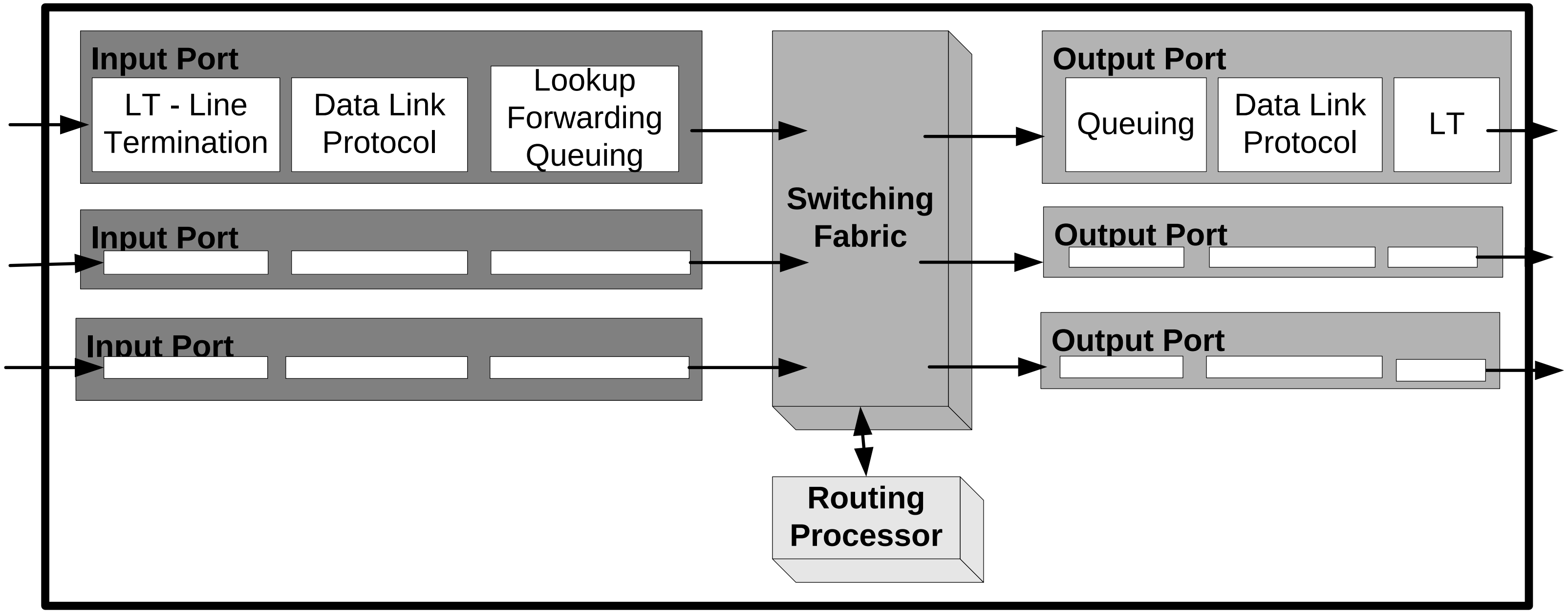
The Internet – an extreme example of what hides behind the communication link abstraction

- Internet Core and Edge
- The hardware
  - Router
  - Network adaptor
- Hourglass communication model
- Protocol stack
- It's along way to Tipperary – the way a message squizes through protocol layers

Internet Core and Edge

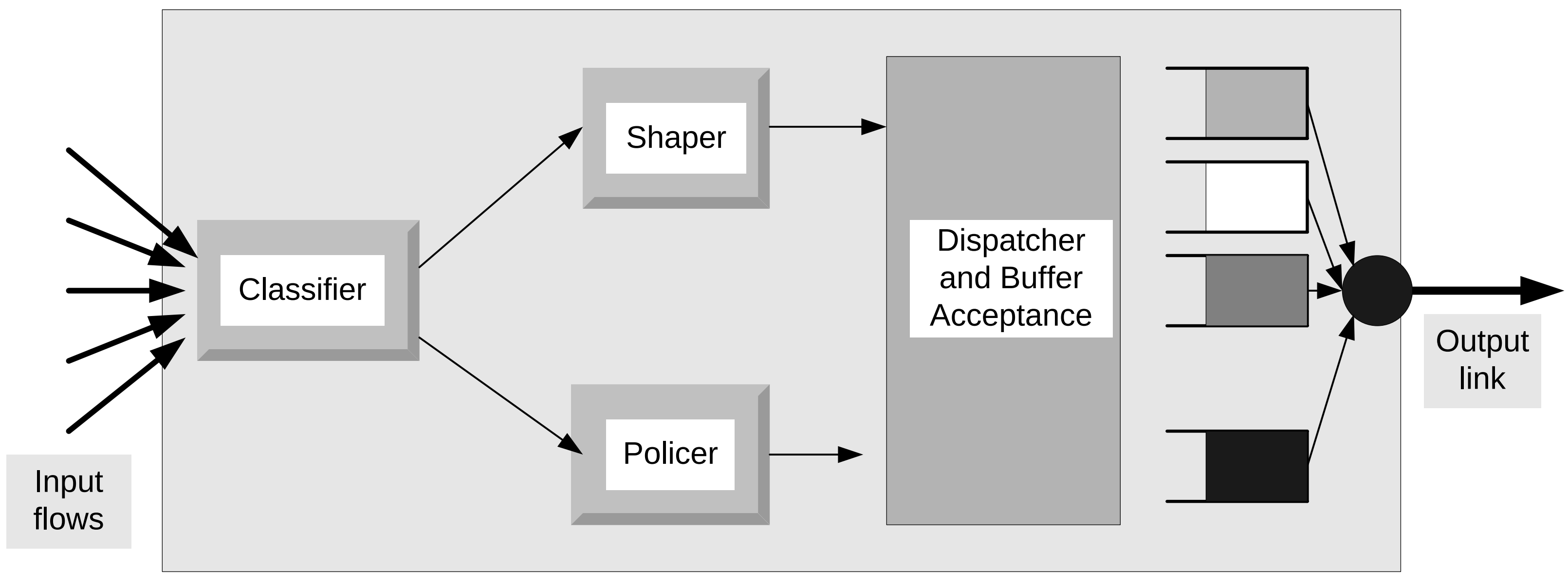


Router

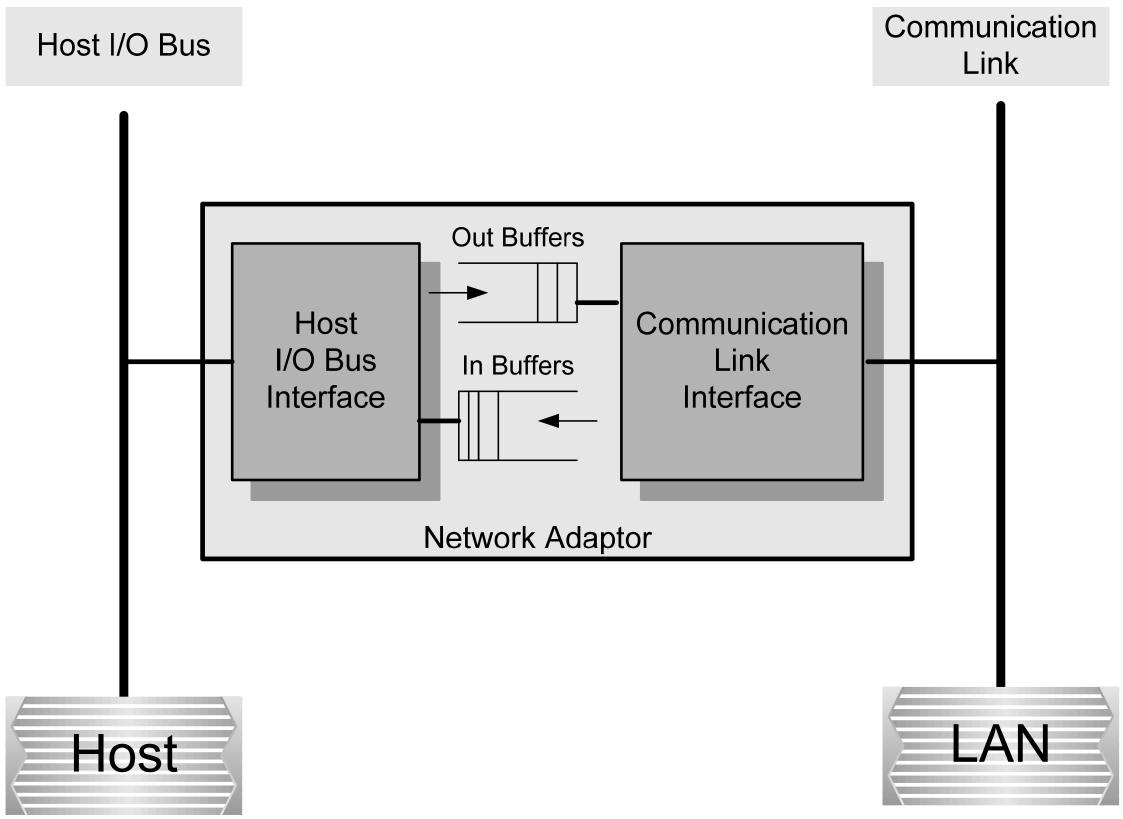


Input Port
LT - Line Termination
Data Link Protocol
Lookup Forwarding Queuing
Input Port
Input Port
Switching Fabric
Routing Processor
Output Port
Queuing
Data Link Protocol
LT
Output Port
Output Port

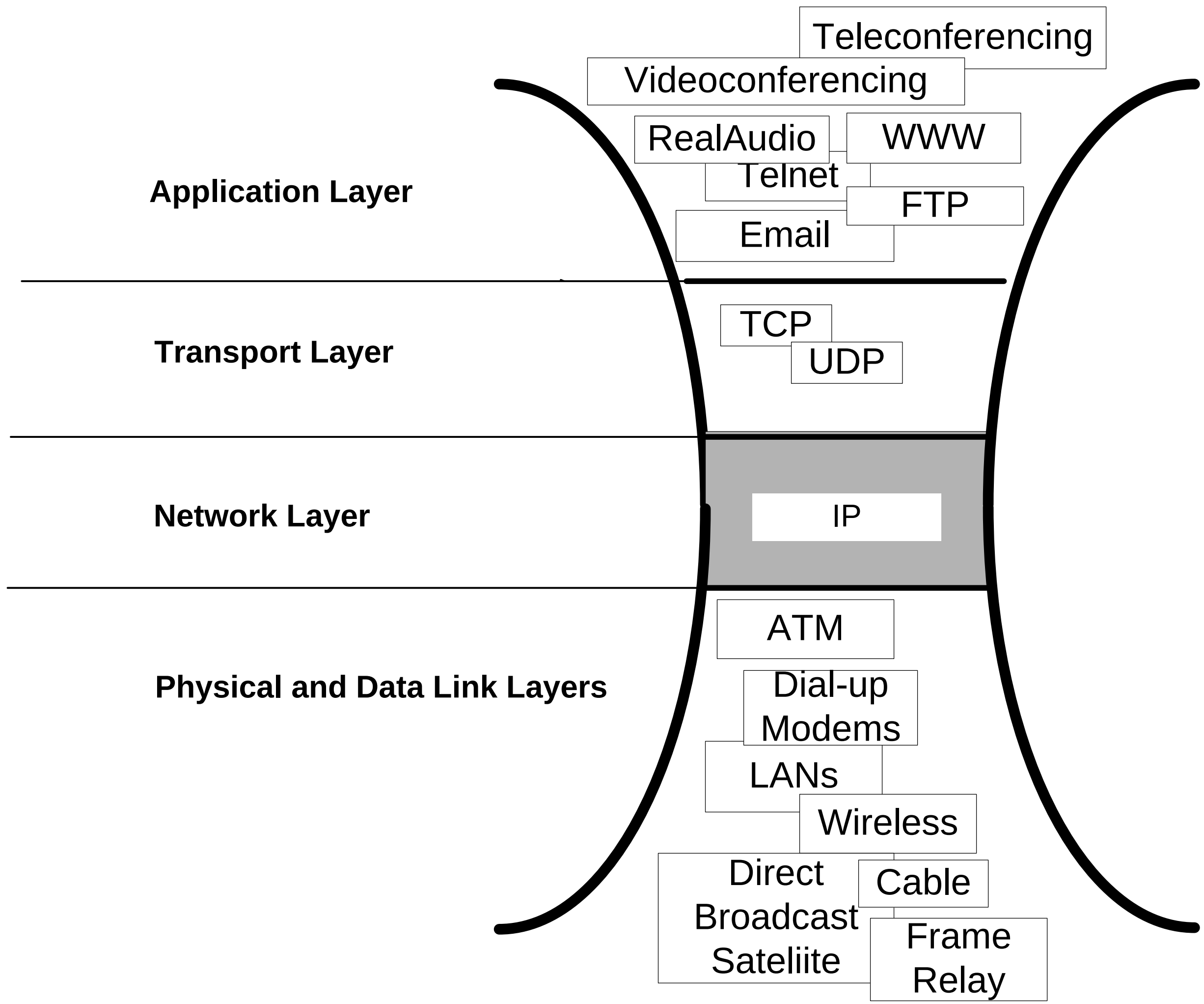
Router supporting QoS (Quality of Service)



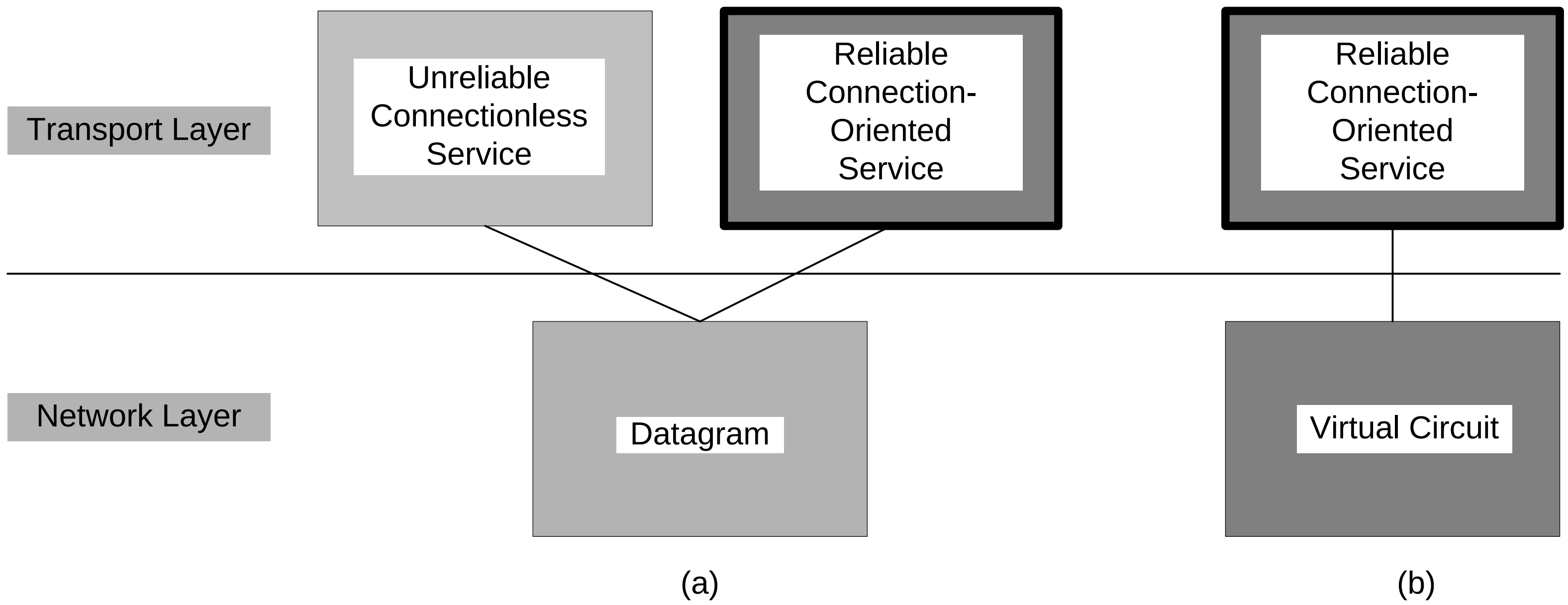
The network adaptor



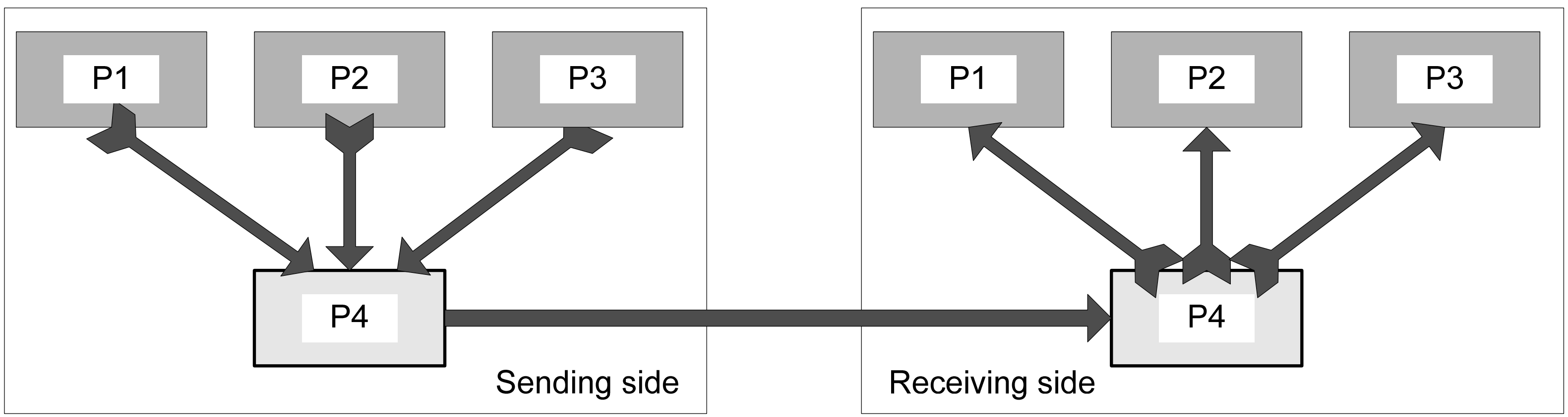
Hourglass communication model



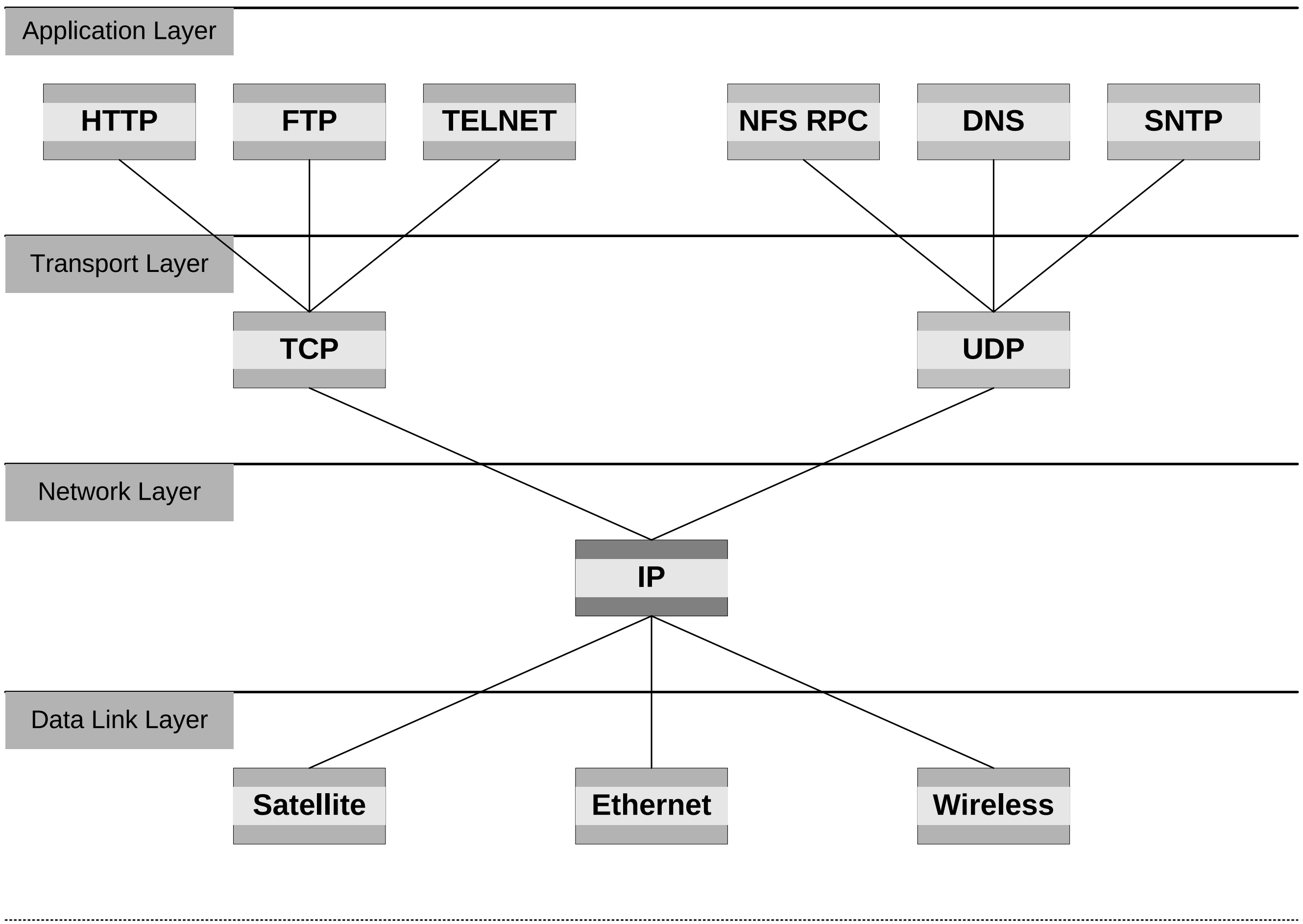
Transport and Network Services



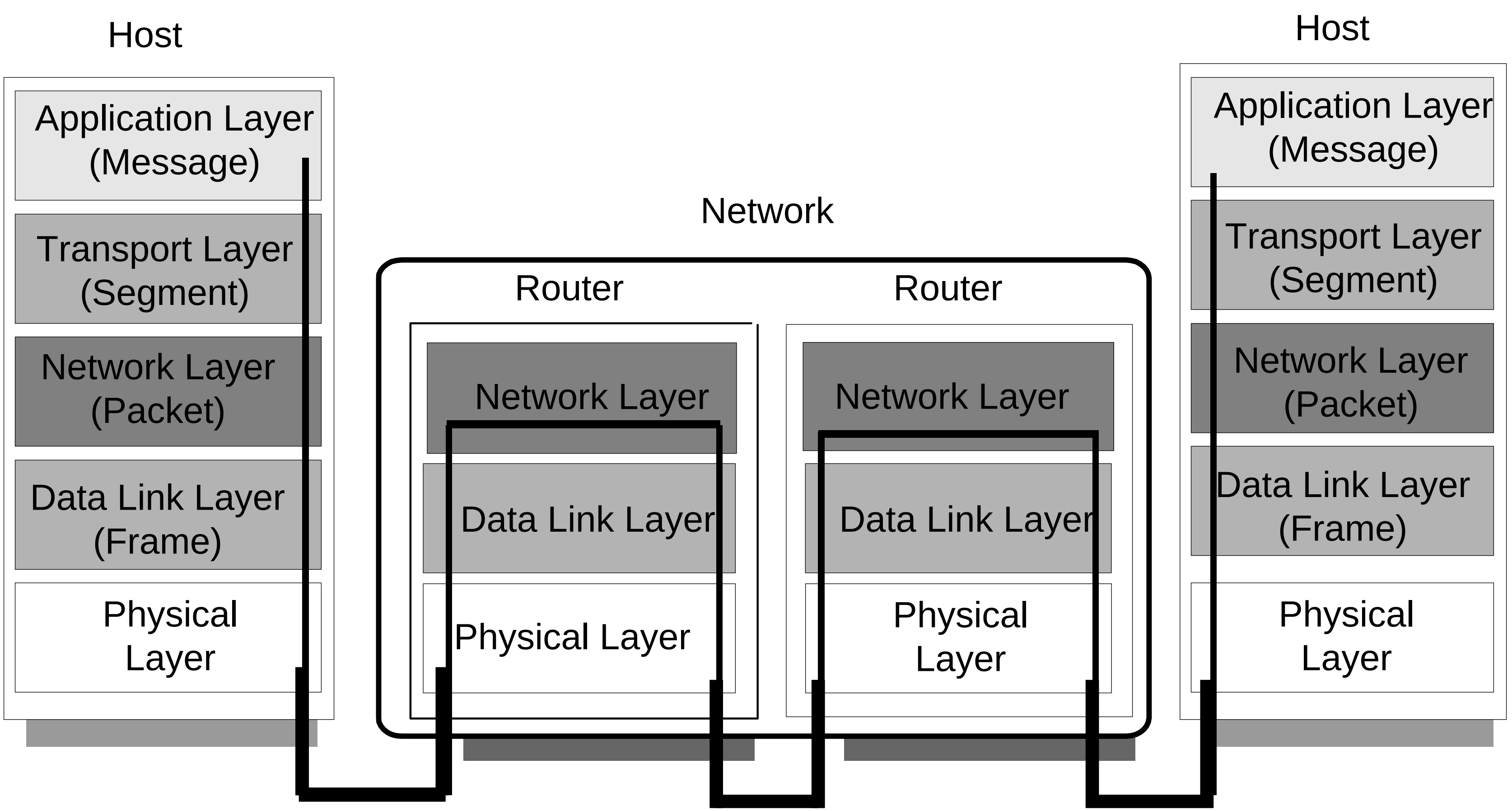
Multiplexing and Demultiplexing



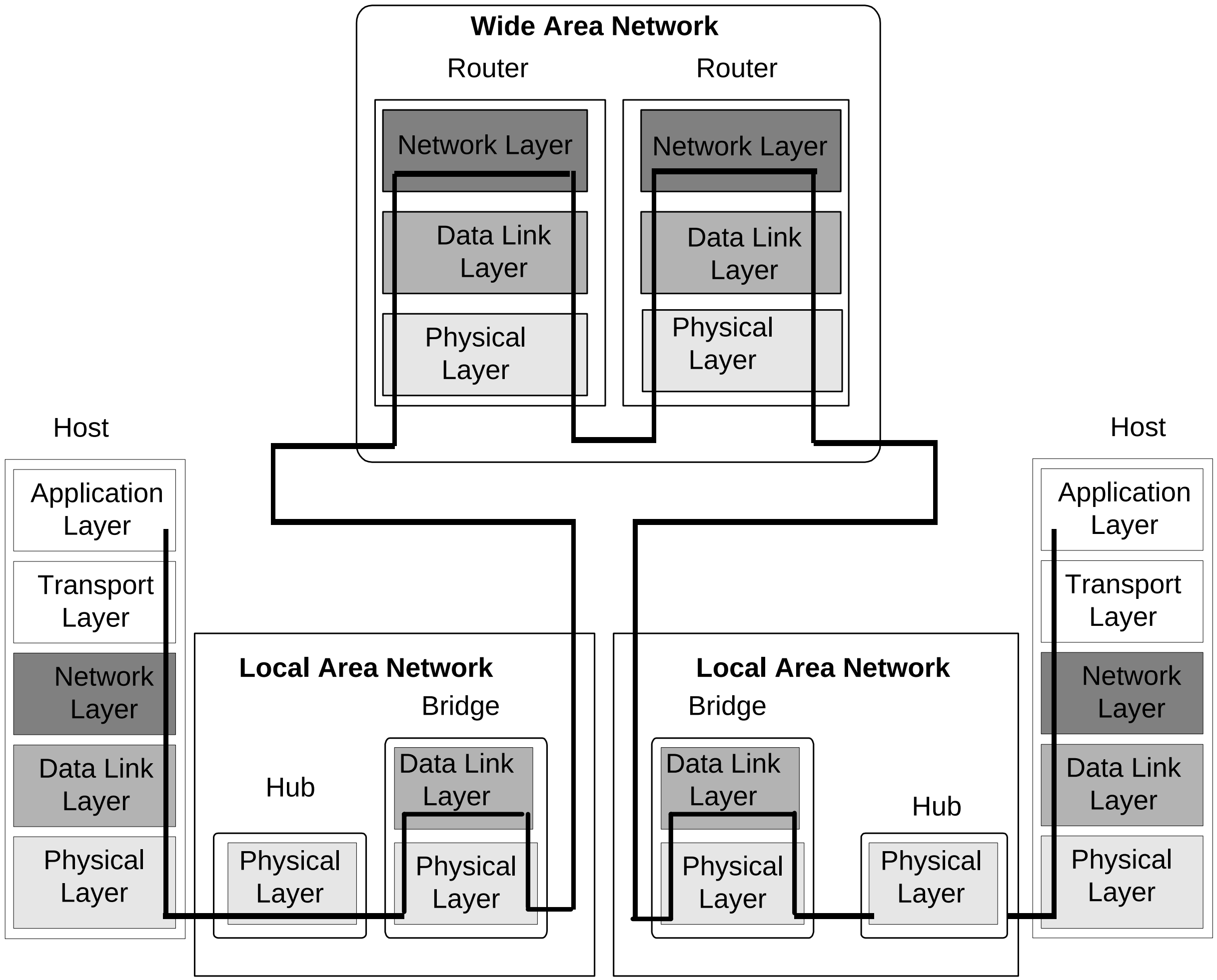
Application, Transport, Network, and Data Link Layer Protocols



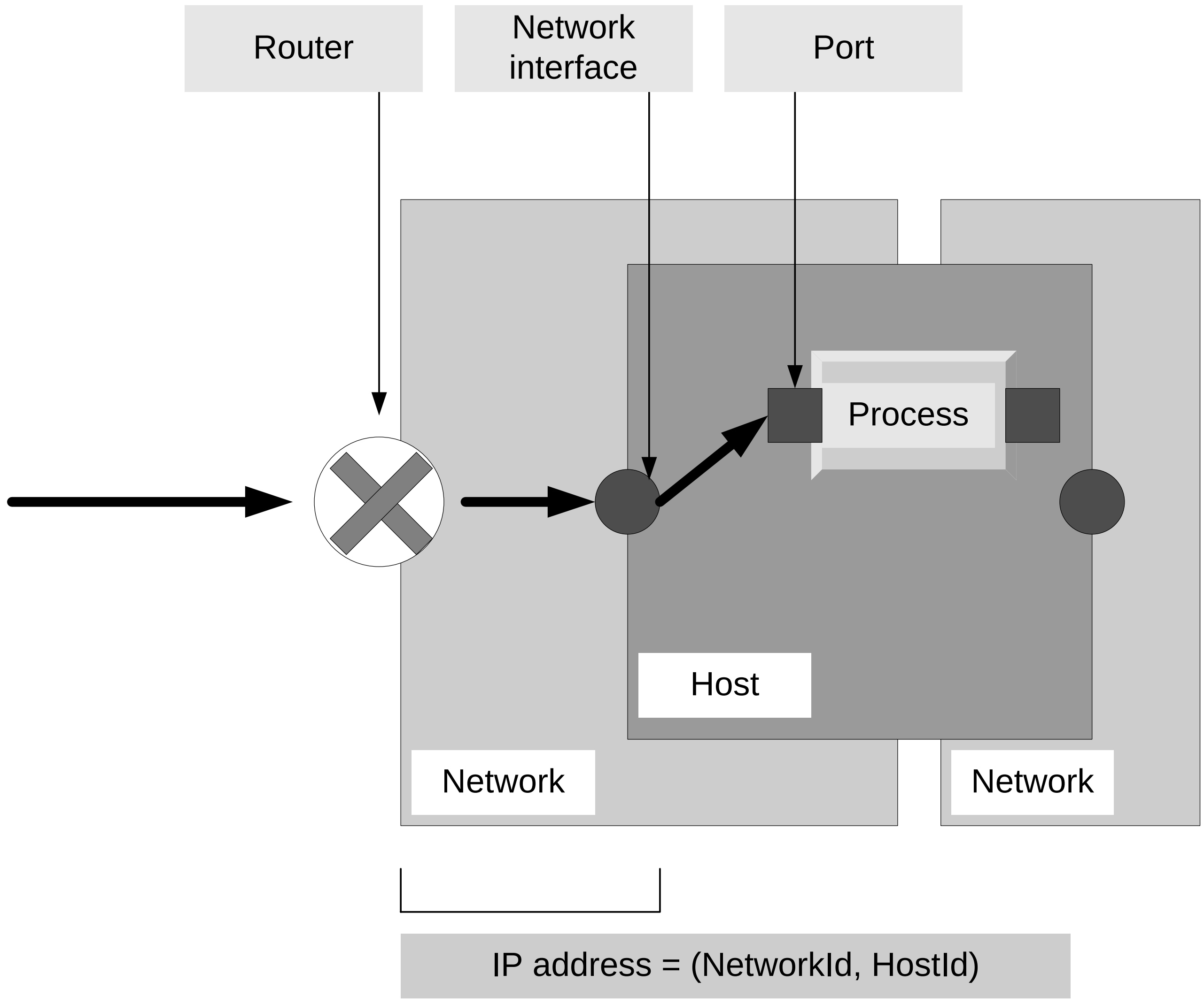
It's a long way to Tipperary it's a long way to go!!



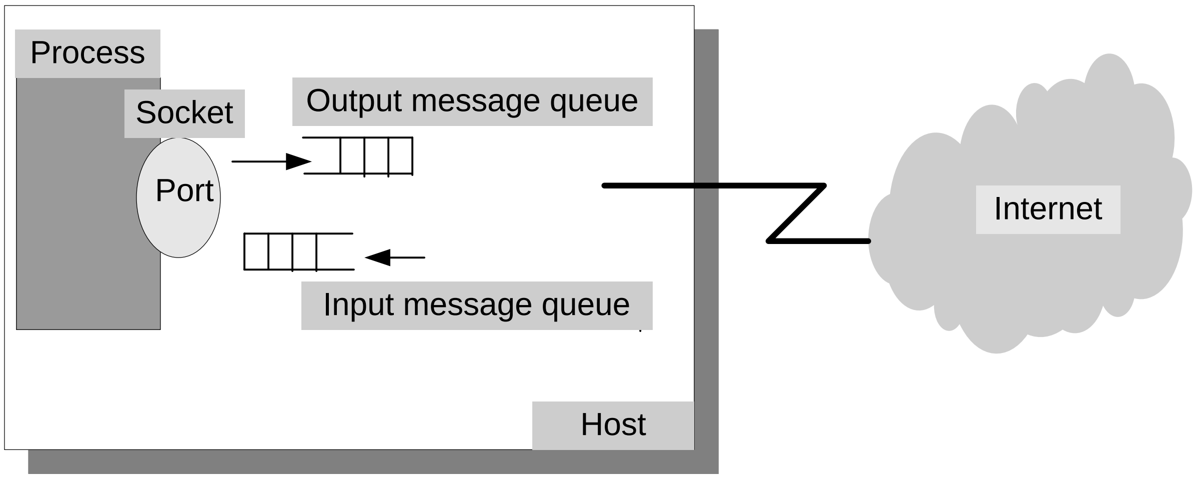
From Local Area to Wide Area Networks



Message delivery to processes



Sockets and Ports



Naming

- The tree abstractions manipulate objects identified by name.
- How could object A access object B:
  - Make a copy of object B and include it in A → <u>use by value</u>
    - Safe → there is a single copy of B
    - How to implement sharing of object B?
  - Pass to A the means to access B using its name → <u>use by reference</u>
    - Not inherently safe → both A and C may attempt to modify B at the same time. Need some form of concurrency control.

Binding and indirection

- Indirection → decoupling objects from their physical realization through names.
- Names allow the system designer to:
  1. organize the modules of a system and to define communication patterns among them
  2. defer for a later time
     - to create object B referred to by object A
     - select the specific object A wishes to use
- Binding → linking the object to names. Examples:
  - A compiler constructs
    - a table of variables and their relative address in the data section of the memory map of the process
    - a list of unsatisfied external references
  - A linker binds the external references to modules from libraries

Generic naming model

- Naming scheme → strategy for naming. Consists of:
  - Name space → the set of acceptable names; the alphabet used to select the symbols from and the syntax rules.
  - Universe of values → set of objects/values to be named
  - Name mapping algorithm → resolves the names, establishes a correspondence between a name and an object/value
  - Context → the environment in which the model operates.
    - Example: searching for John Smith in the White Pages in Orlando (one context) or in Tampa (another context).
    - Sometimes there is only one context → universal name space; e.g., the SSNs.
    - Default context

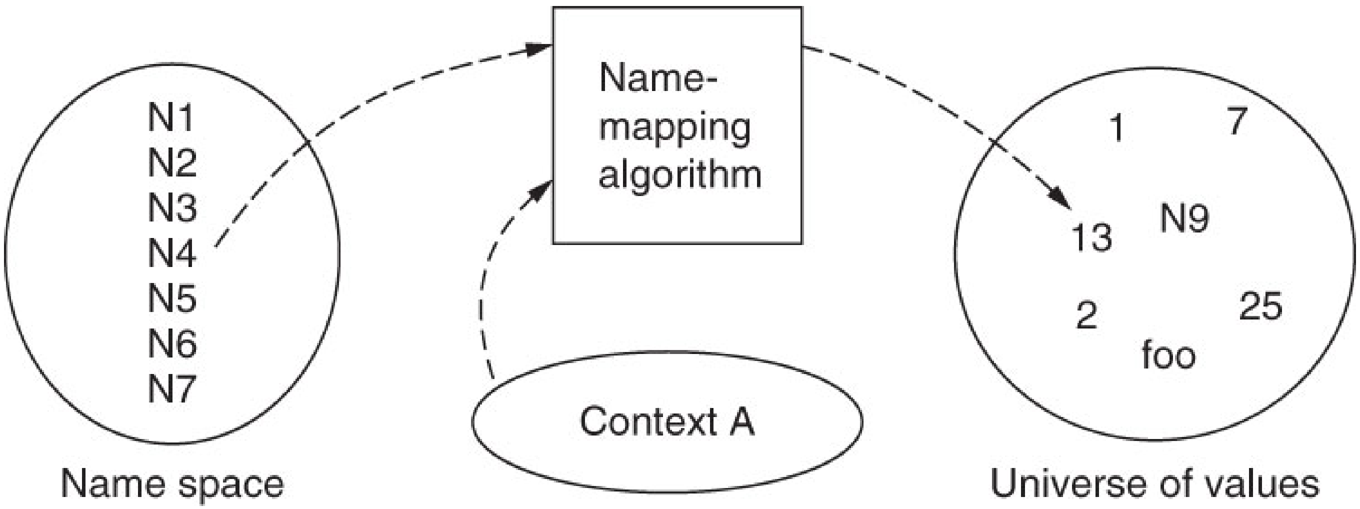


Figure 2.10 from the textbook

Operations on names in the abstract model

- Simple models: value ← RESOLVE (name, context)
  - The interpreter:
    - Determines the version of the RESOLVE (which naming scheme is used)
    - Identifies the context
    - Locates the object
  - Example: the processor
- Complex models support:
  - creation of new bindings: status ← BIND(name, value, context)
  - deletion of old bindings: status ← UNBIND(name, value)
  - enumeration of name space: list ← ENUMERATE(context)
  - comparing names status: result ← COMPARE(name1,name2)

Name mapping

- Name to value mapping
  - One-to-One → the name identifies a single object
  - Many-to-One → multiple names identify one objects (aliasing)
  - One-to-Many → multiple objects have the same name even in the same context.
- Stable bindings → the mapping never change. Examples:
  - Social Security Numbers
  - CustomerId for customer billing systems

Name-mapping algorithms

1. Table lookup
   1. Phone book
   2. Port numbers → a port the end point of a network connection
2. Recursive lookup:
   1. File systems – path names
   2. Host names – DNS (Domain Name Server)
   3. Names for Web objects - URL – (Universal Resource Locator)
3. Multiple lookup → searching through multiple contexts
   1. Libraries
   2. Example: the classpath is the path that the Java runtime environment searches for classes and other resource files

1. Table lookup

Name	Value
N1	7
N2	foo
N3	25
N4	13
N5	2
N6	1
N7	N9

Bindings

Context A

Figure 2.11 from the textbook

How to determine the context

- Context references:
  - Default → supplied by the name resolver
    - Constant → built-in by the name resolver
      - Processor registers (hardwired)
      - Virtual memory (the page table register of an address space)
    - Variable → supplied by the current environment
      - File name (the working directory)
  - Explicit → supplied by the object requesting the name resolution
    - Per object
      - Looking up a name in the phone book
    - Per name → each name is loaded with its own context reference (qualified name).
      - URL
      - Host names used by DNS

Dynamic and multiple contexts

- Context reference static/dynamic.
  - Example: the context of the “help” command is dynamic, it depends where you are the time of the command.
- A message is encapsulated (added a new header,) as flows down the protocol stack:
  - Application layer (application header understood only in application context)
  - Transport layer (transport header understood only in the transport context)
  - Network layer (network header understood only in the network context)
  - Data link layer (data link header understood only in the data link context)

2. Recursive name resolution

- Contexts are structured and a recursion is needed for name resolution.
- Root → a special context reference - a universal name space
- Path name → name which includes an explicit reference to the context in which the name is to be resolved.
  - Example: first paragraph of page 3 in part 4 of section 10 of chapter 1 of book “Alice in Wonderland.”
  - The path name includes multiple components known to the user of the name and to name solver
  - The least element of the path name must be an explicit context reference
- Absolute path name → the recursion ends at the root context.
- Relative path name → path name that is resolved by looking up its mot significant component of the path name

Example

- AliceInWonderland.Chapter1.Section10.Part4.Page3.FirstParagraph

  Most significant ← → Least significant

3. Multiple lookup

- Search path → a list of contexts to be searched

  Example: the classpath is the path that the Java runtime environment searches for classes and other resource files

- User-specfic search paths → user-specific binding
- The contexts can be in concentric layers. If the resolver fails in a inner layer it moves automatically to the outer layer.
- Scope of a name → the range of layers in which a name is bound to the same object.

Comparing names

- Questions
  - Are two names the same? → easy to answer
  - Are two names referring to the same object (bound to the same value)? → harder; we need the contexts of the two names.
  - If the objects are memory cells are the contents of these cells the same?

Name discovery

- Two actors:
  - The exporter → advertizes the existence of the name.
  - The prospective user → searches for the proper advertisement. Example: the creator of a math library advertizes the functions.
- Methods
  - Well-known names
  - Broadcasting
  - Directed query
  - Broadcast query
  - Introduction
  - Physical randezvoue

Computer System Organization

- Operating Systems (OS) → software used to
  - Control the allocation of resources (hardware and software)
  - Support user applications
  - Sandwiched between the hardware layer and the application layer
- OS-bypass: the OS does not hide completely the hardware from applications. It only hides dangerous functions such as
  - I/O operations
  - Management function
- Names → modularization

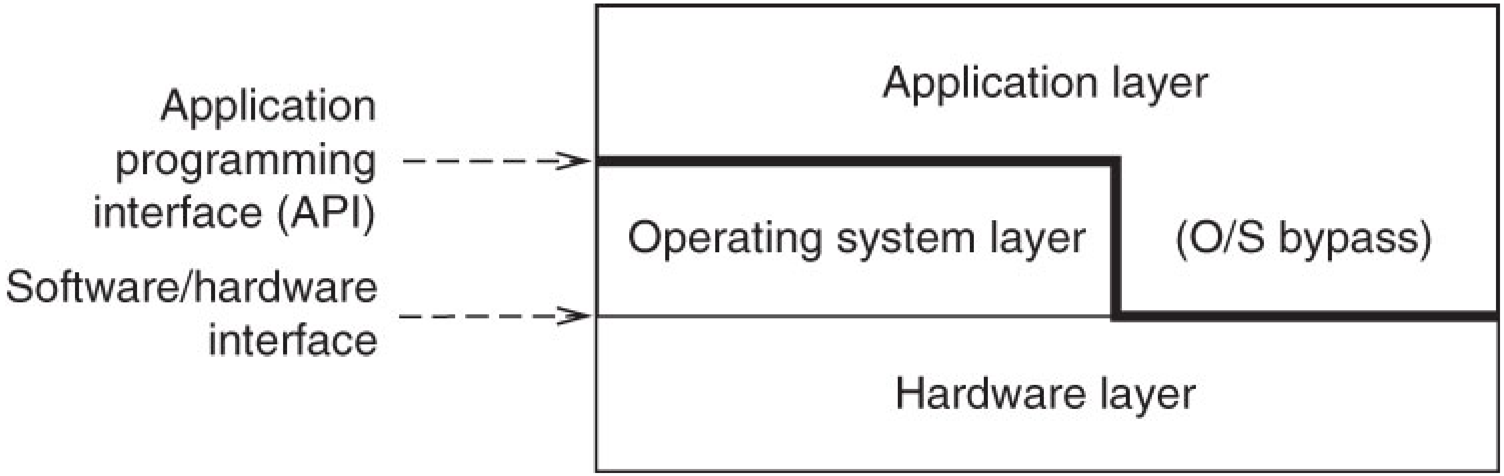


Figure 2.16 from the textbook

The hardware layer

- Modules representing each of the three abstractions (memory, interpreter, communication link) are interconnected by a bus.
- The bus → a broadcast communication channel, each module hears every transmission.
  - Control lines
  - Data lines
  - Address lines
- Each module
  - is identified by a unique address
  - has a bus interface
- Modules other than processors need a controller.

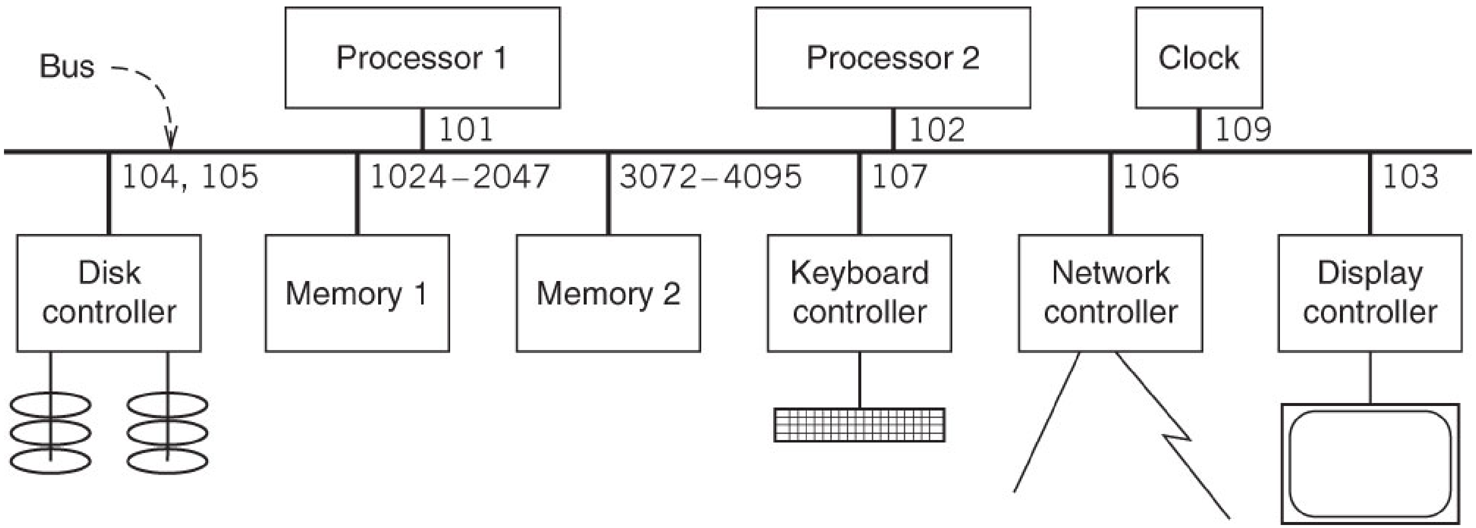


Figure 2.17 from the textbook