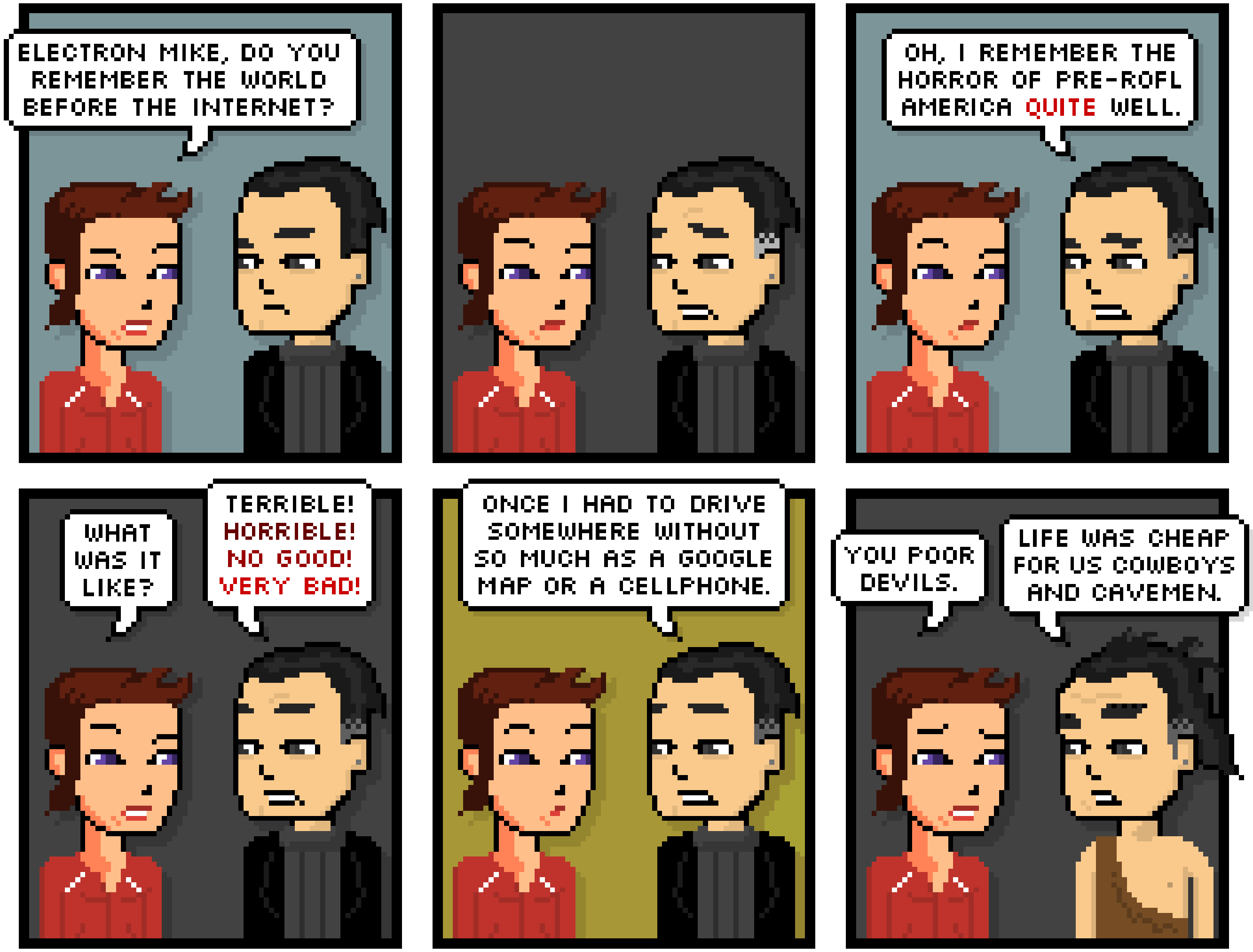
NETWORKS

## Today

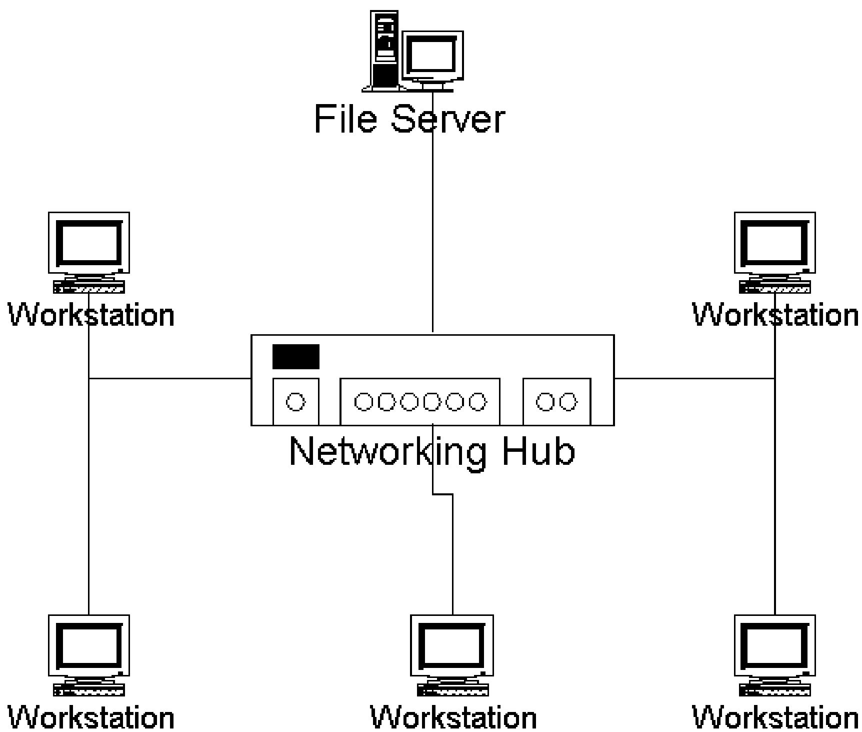
- We now move on to the fourth topic in the course:
  - Network programming
- Before we look at the support that Java provides for network programming, we will look at some basic information on networks.
- We will also look at *threads*, which we will need for some of the network stuff that we do.

## Network basics

- The reason we are interested in networks is that we want to pass data between computers.
- What we will focus on today are:
  - The issues in doing this
  - Standard ideas to deal with the issues
  - Some example instantiations of these ideas.
- In short the ISO/OSI model and TCP/IP.


ELECTRON MIKE, DO YOU REMEMBER THE WORLD BEFORE THE INTERNET?
OH, I REMEMBER THE HORROR OF PRE-ROFL AMERICA QUITE WELL.
WHAT WAS IT LIKE?
TERRIBLE! HORRIBLE! NO GOOD! VERY BAD!
ONCE I HAD TO DRIVE SOMEWHERE WITHOUT SO MUCH AS A GOOGLE MAP OR A CELLPHONE.
YOU POOR DEVILS.
LIFE WAS CHEAP FOR US COWBOYS AND CAVEMEN.
(C) 2005 R STEVENS :: DIESELSWEETIES.COM

- Think of the *local area network* or LAN as the basic building block.



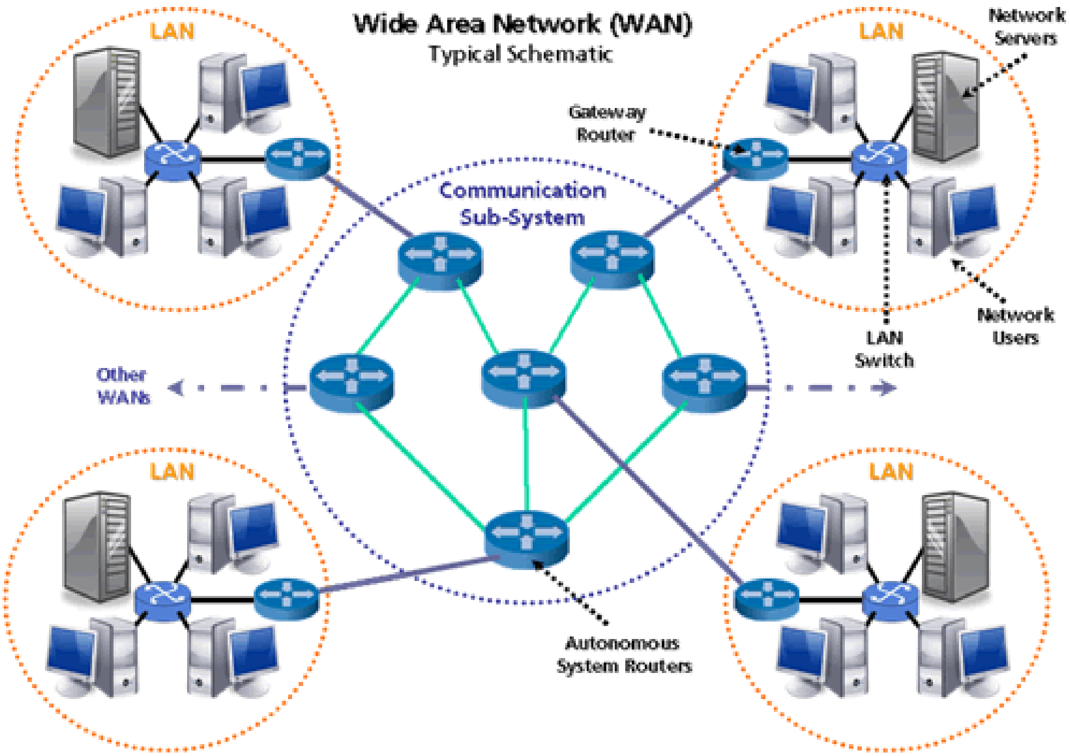
- No hard and fast definition, but:

  Group of computers and other hardware, in some geographic local area, connected through a switch.

- I would define both the BC network and what I have at home as LANs.
- Different LAN technologies:
  - Ethernet (802.3, 802.11)
  - Token ring

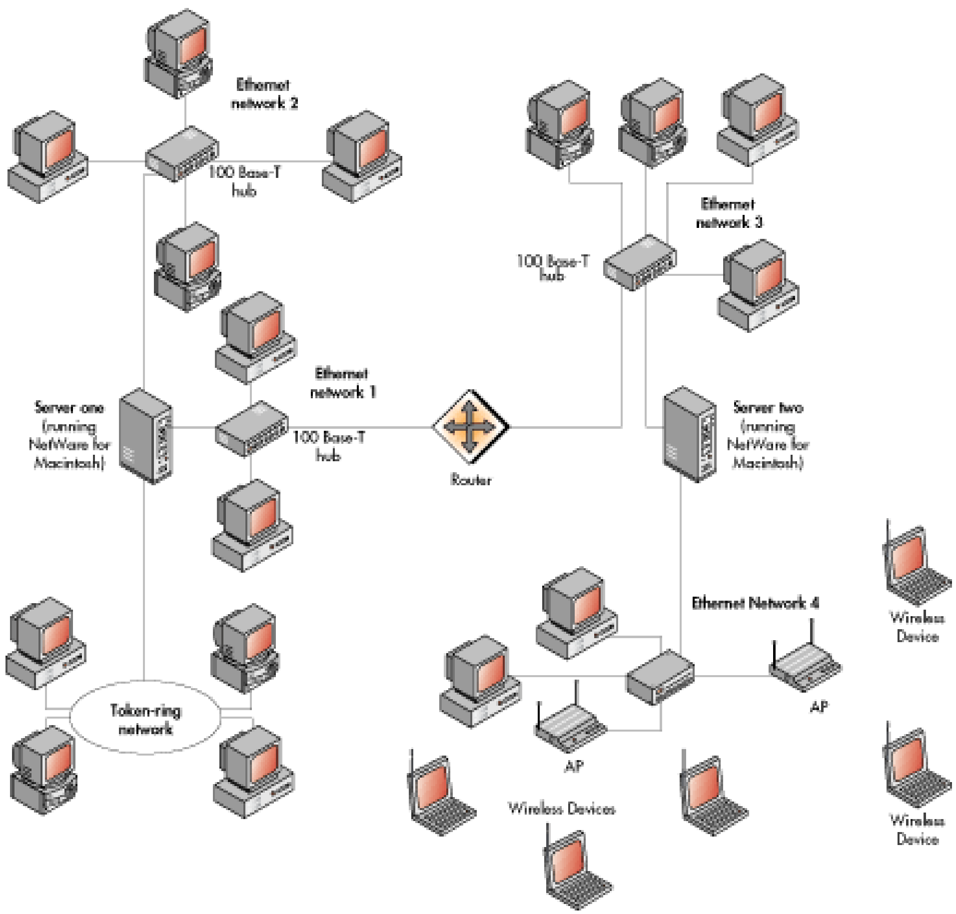
  but typically think of a LAN as being homogenous in terms of this technology.

- Sometimes talk of *wide area networks*, WANs, as several LANs connected together.




- Typically this is in the context of some organization that owns all the LANs.

- Also talk of *internetworks* as connections between different network technologies:



# Challenges

- The basic challenge is to get these different technologies to work together:
  - Different media
  - Different speeds
  - Different ways of passing data
- Typical computer science solution:

  *abstraction*
- The ISO Open System Interconnection (OSI) model splits the process of data transfer up, into simpler classes of task.

# The OSI model

| | |
|---|---|
| Layer 7 | Application |
| Layer 6 | Presentation |
| Layer 5 | Session |
| Layer 4 | Transport |
| Layer 3 | Network |
| Layer 2 | Data Link |
| Layer 1 | Physical |

- "All people seem to need data processing"

- Upper layers:
  - Application
  - Presentation
  - Session

  deal with applications issues and typically are all software.

- Lower layers:
  - Transport
  - Network
  - Data link
  - Physical

  handle data transport
- Physical and data link tend to be implemented in hardware and software.
- Physical is responsible for placing information on the medium.

- While the OSI model is a nice way of thinking about what is necessary to communicate, it is not itself a mechanism for communication.
- Communication is made possible by using communication protocols.
- A *protocol* is a set of rules and conventions that implement the functions laid down by one or more layers in the OSI model.

# Some protocols

- LAN protocols

  Operate at physical and datalink layers.
- WAN protocols

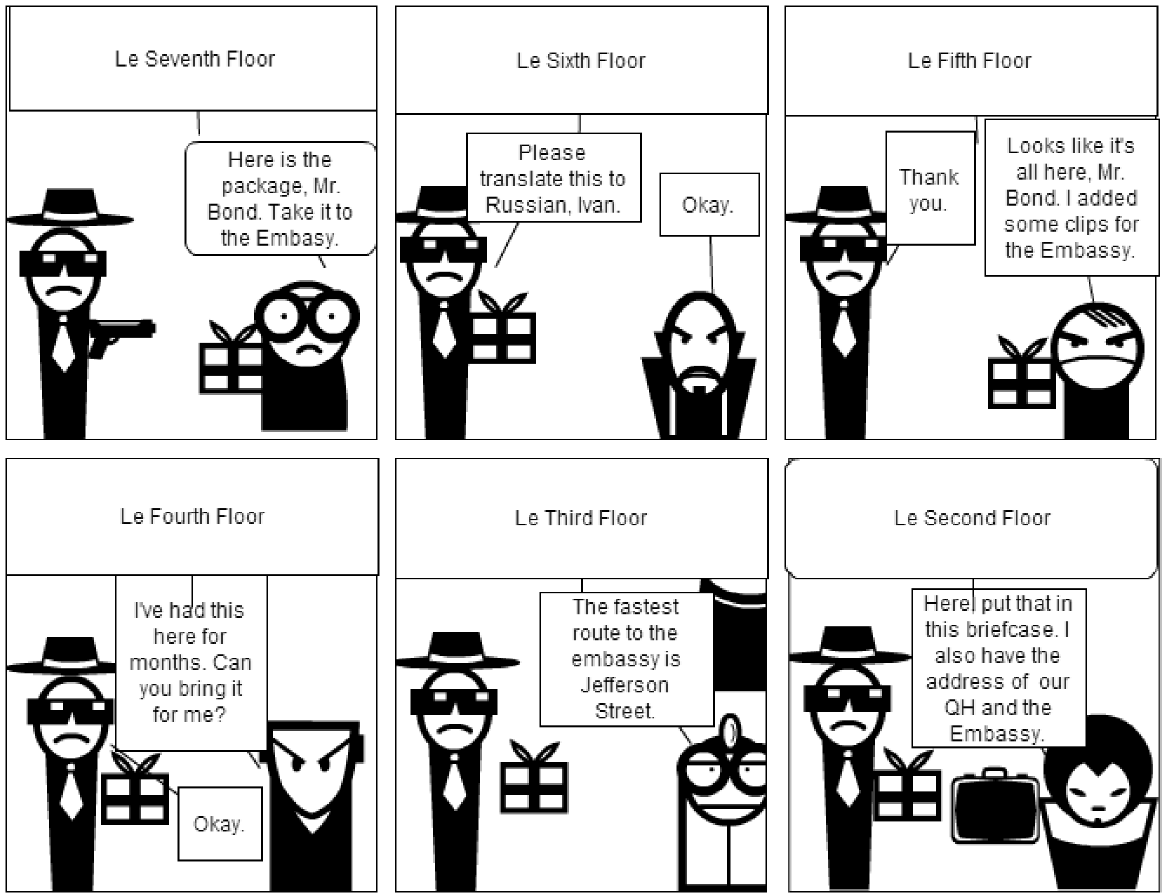
  Operate at lowest three layers.
- Routing protocols

  Network layer protocols that exchange information between routers to select the rignt path for network traffic.
- Network protocols

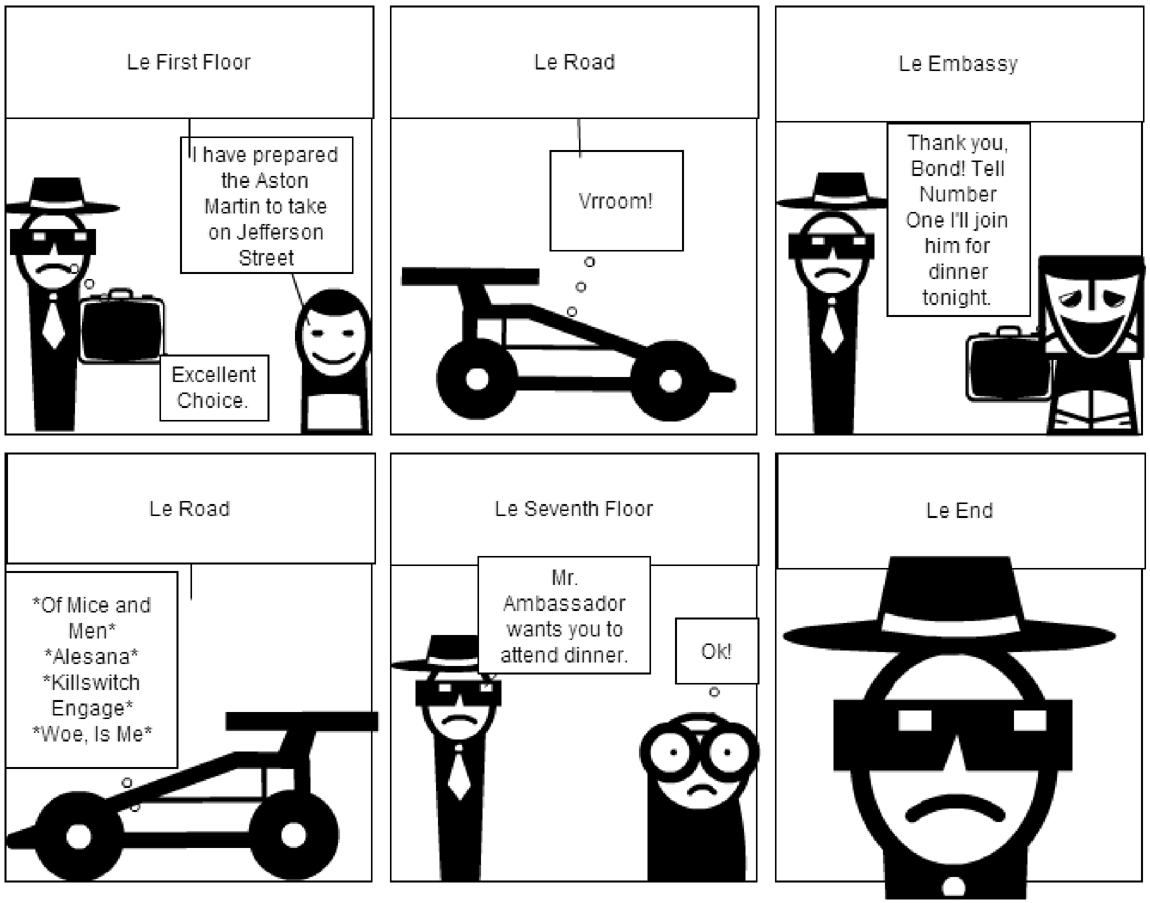
  Upper level protocols in a protocol suite. User, for example, by routing protocols.

# How data is really transferred

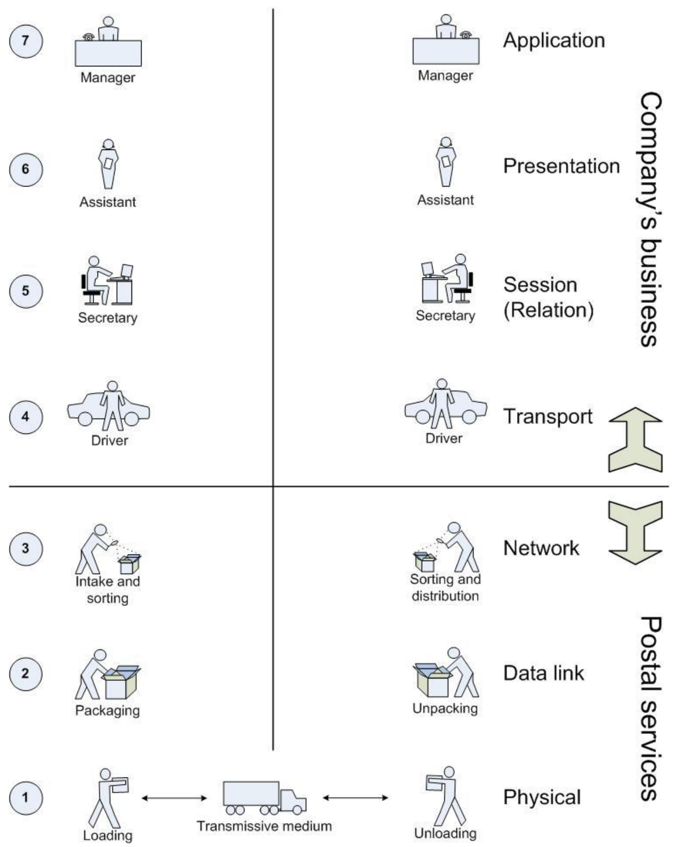
- Here are some different analogies for the role of the different layers in data transfer.
- None of them work exactly, but I think all of them help to communicate the idea.
- The first imagines a message being passed between the British Embassy and the Russian Embassy in some city.



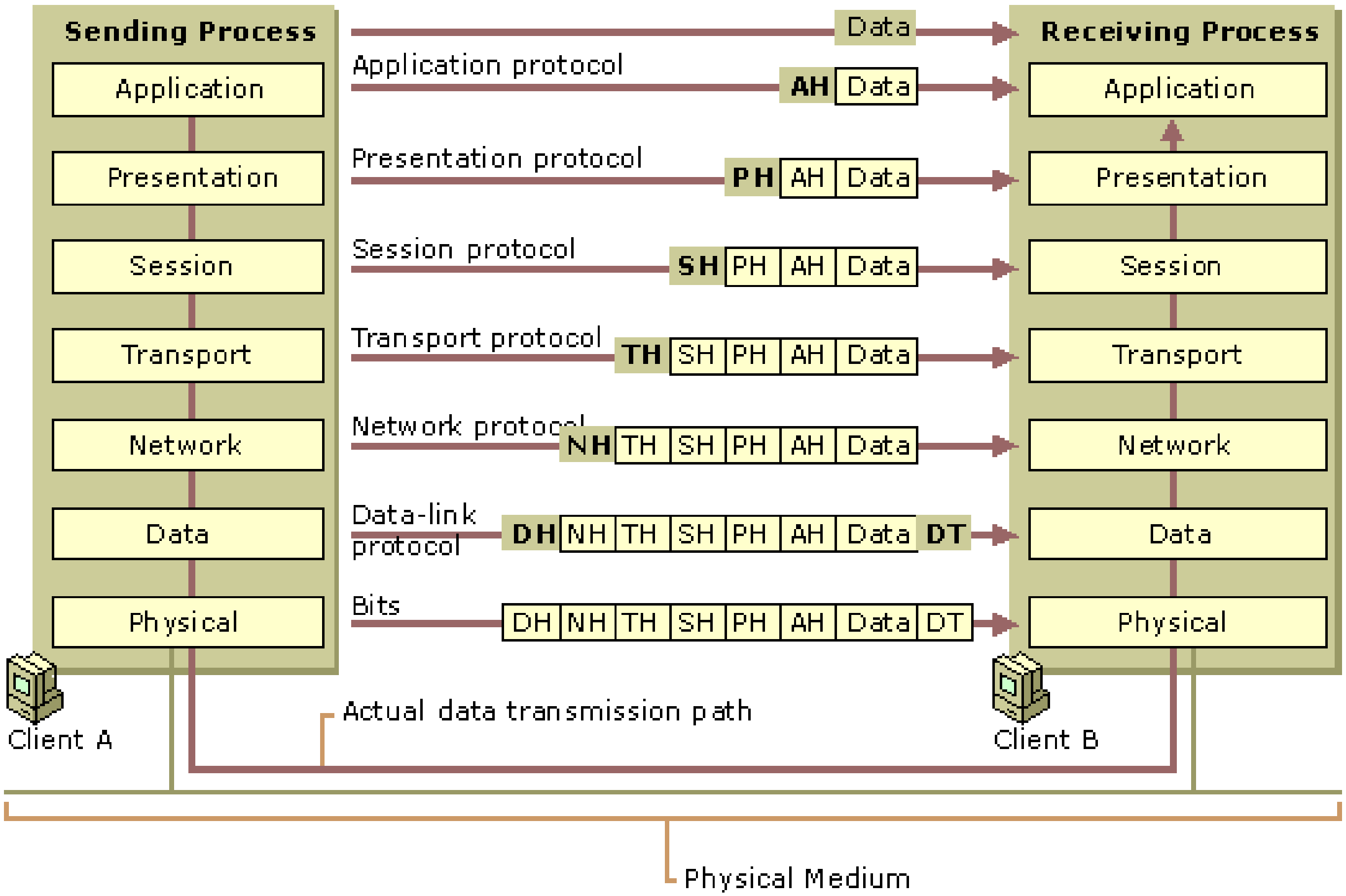
Le Seventh Floor
Here is the package, Mr. Bond. Take it to the Embasy.
Le Sixth Floor
Please translate this to Russian, Ivan.
Okay.
Le Fifth Floor
Thank you.
Looks like it's all here, Mr. Bond. I added some clips for the Embassy.
Le Fourth Floor
I've had this here for months. Can you bring it for me?
Okay.
Le Third Floor
The fastest route to the embassy is Jefferson Street.
Le Second Floor
Here! put that in this briefcase. I also have the address of our QH and the Embassy.


Le First Floor
I have prepared the Aston Martin to take on Jefferson Street
Excellent Choice.
Le Road
Vrroom!
Le Embassy
Thank you, Bond! Tell Number One I'll join him for dinner tonight.
Le Road
*Of Mice and Men* *Alesana* *Killswitch Engage* *Woe, Is Me*
Le Seventh Floor
Mr. Ambassador wants you to attend dinner.
Ok!
Le End

- The next example imagines the way that two businesses communicate.

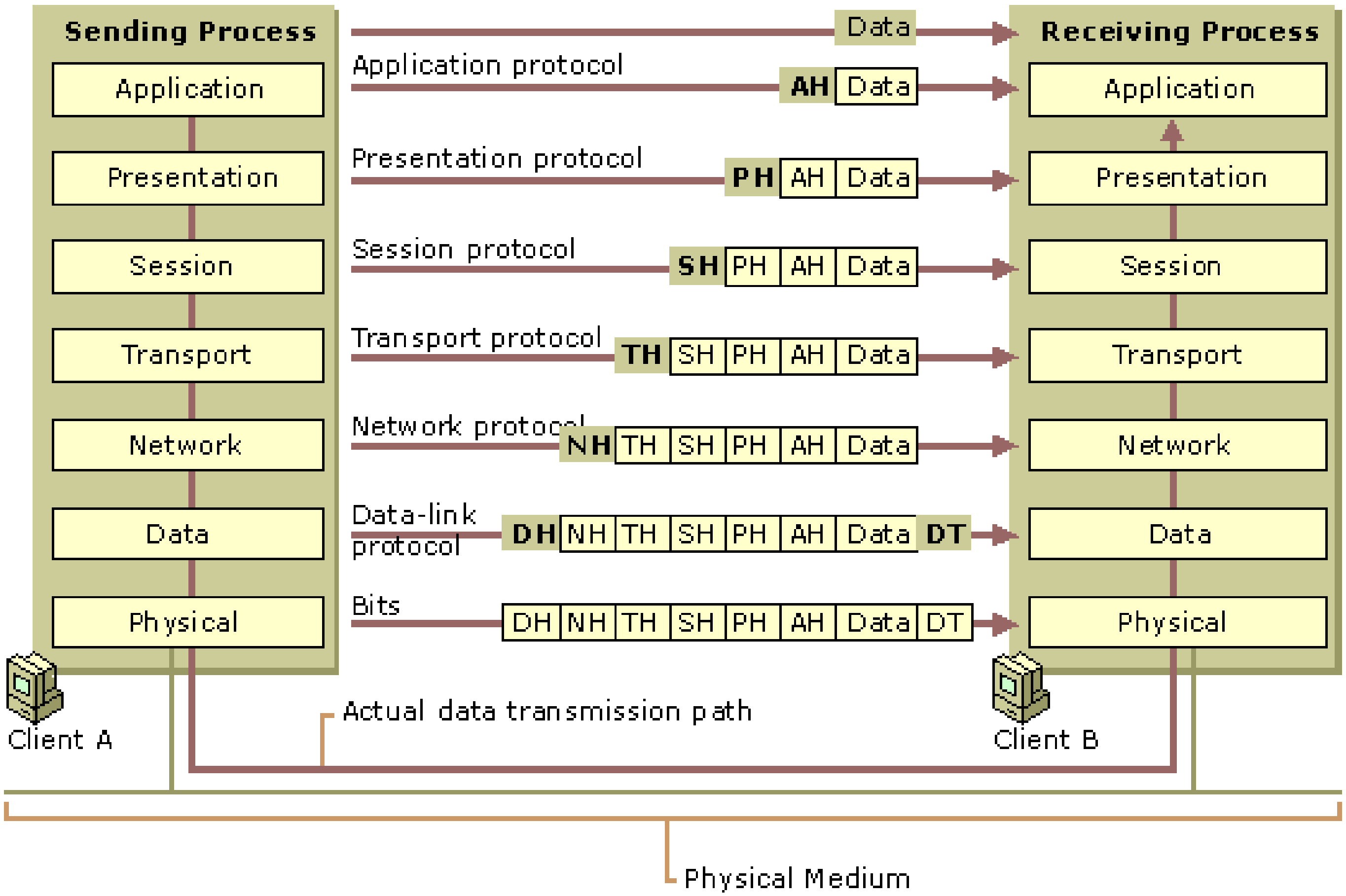



RM – OSI and letter communication parallel



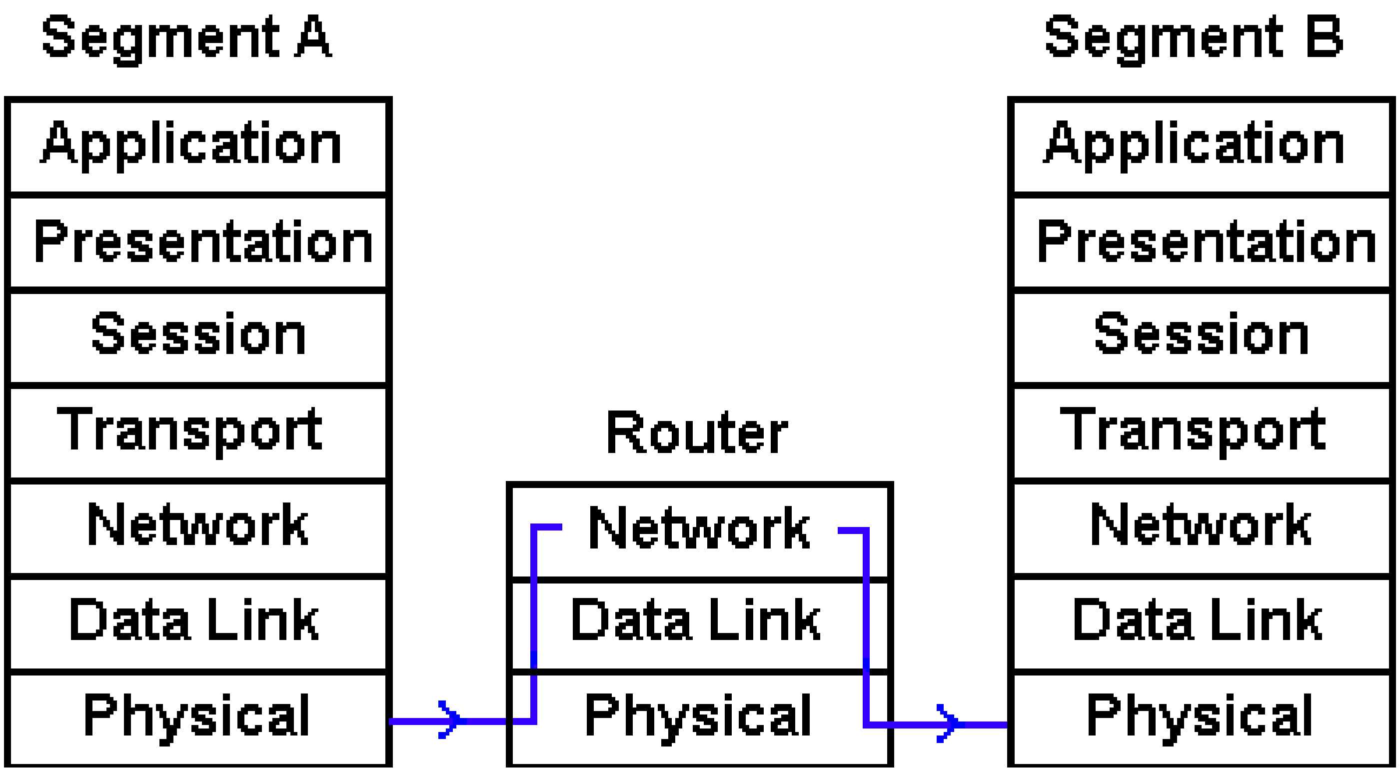

- In a sense data does pass through all the levels of the model.

- Application sends data to a peer application (ie a browser requesting a web-page).
- Application data attaches control information as a header and passes data to the presentation layer.
- Presentation layer adds a header and passes to the session layer.
- …
- Physical layer adds a header and places the data on the network hardware.
- At receiver physical layer removes the data, strips off the header and passes it up to the datalink layer
- …
- Application layer reads the data.

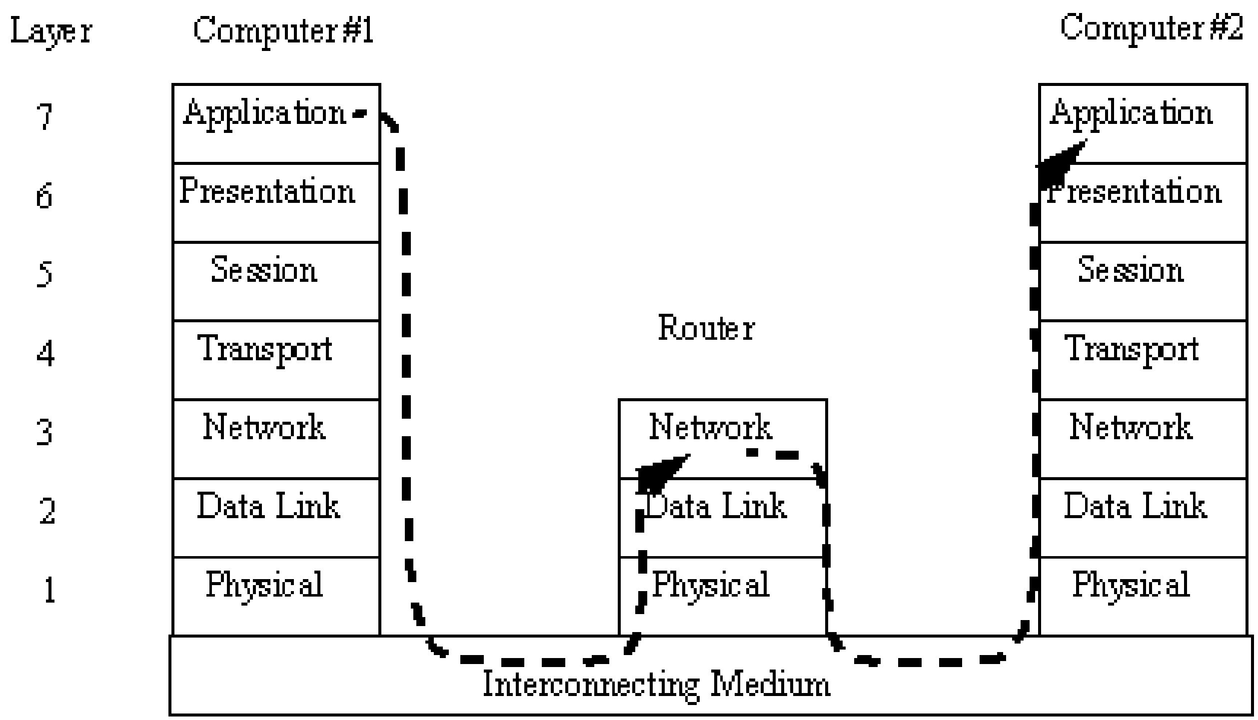



- In another sense, layers communicate.

- We can think of software that implements protocols at each layer in the model, “talking” to software at the same level in other systems.
- For example, routing software (network layer) on one machine talking to routing software on another, making use of the data and transport layers.



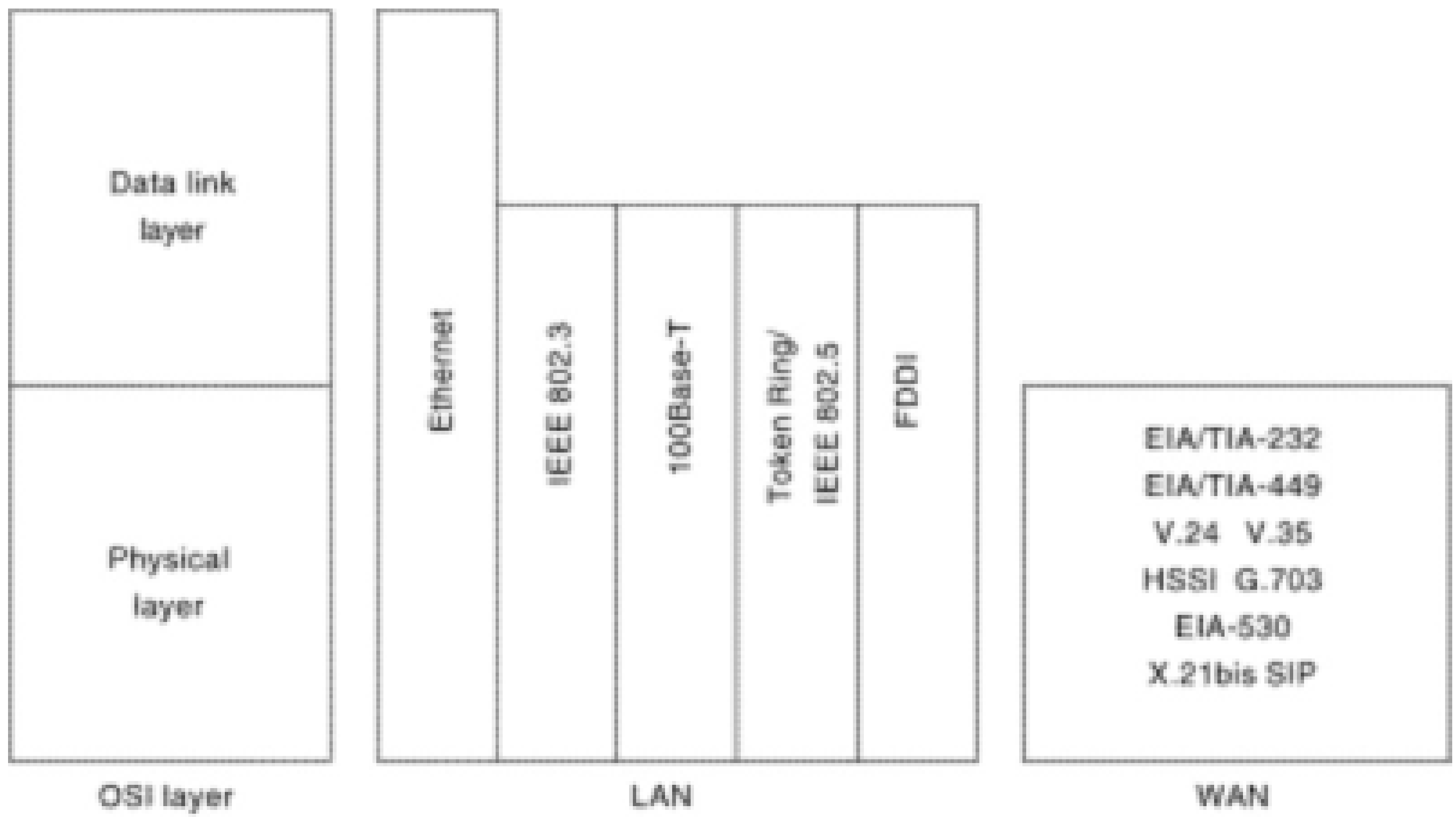

- So data is passed between networks via the router.




- And, of course, data between applications passes through the other layers as well.

## Physical layer

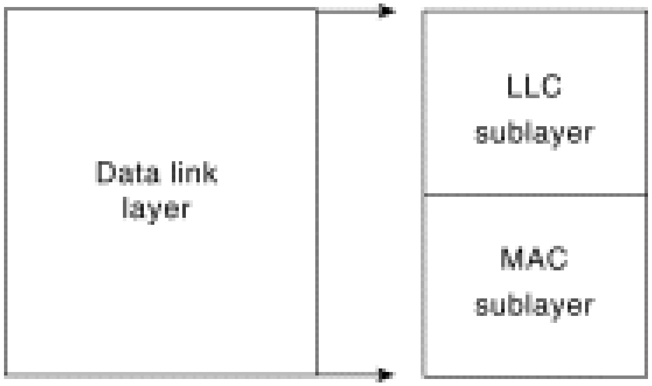
- Physical layer defines the physical properties of the physical link.
  - Electrical
  - Mechanical
  - Functional
- Defines characteristics like voltage levels and timing.


Data link layer
Physical layer
OSI layer
Ethernet
IEEE 802.3
100Base-T
Token Ring/ IEEE 802.5
FDDI
LAN
EIA/TIA-232
EIA/TIA-449
V.24 V.35
HSSI G.703
EIA-530
X.21bis SIP
WAN

# Datalink layer

- The data link layer provides reliable transfer of data over the physical link.
- Defines the topology
  - Ring
  - Bus
- Defines flow control to make sure devices are not overloaded
- Reorders frames that are out of sequence
- Flags transmission errors.

- The data link layer has two sublayers




- The *logical link control* layer manages communications over a single network link.

  IEEE 802.2

- This is the connection on which everything else is built.

- The *media access control* (MAC) layer manages access to the physical network.

  IEEE MAC specification defines MAC addresses. These provide unique identification at the data link layer.

## Network layer

- Defines the network address
  - Separate from the MAC address
- The Internet protocol (IP) defines network addresses so that it is easy to route from them.
  - Route easily established from source address, destination address and subnet mask.
- Routers can thus use the network address to decide how to route packets.

## Transport layer

- Segments data from the session layer for transport across the network.
- Provides error checking
  - Checking for lost packets
- Provides error recovery
  - Requesting retransmission
- TCP and UDP are transport layer protocols.

## Session layer

- The session layer handles communication sessions:
  - Establishes
  - Manages
  - Terminates

## Presentation layer

- Provides coding and conversion functions to the application layer.
- Having standard mechanisms for encoding different types of data:
  - MPEG
  - QuickTime
  - TIFF
  - JPEG
  - SSL

  means that applications don't have to provide these services — they just decide what to send and let the presentation layer protocols figure it out.

## Application layer

- Interacts directly with the software the user runs.
- HTTP, FTP, SMTP are example application level protocols

## Connection-oriented v connectionless

- Connection-oriented services have three phases:
  - Open connection
  - Data transfer
  - Close connection
- Provide resource reservation
  - Allows for DoS attacks
- Allow for detecting and resending lost packets.
- Connectionless services have none of these overheads, but cannot detect lost packets.

# Addressing

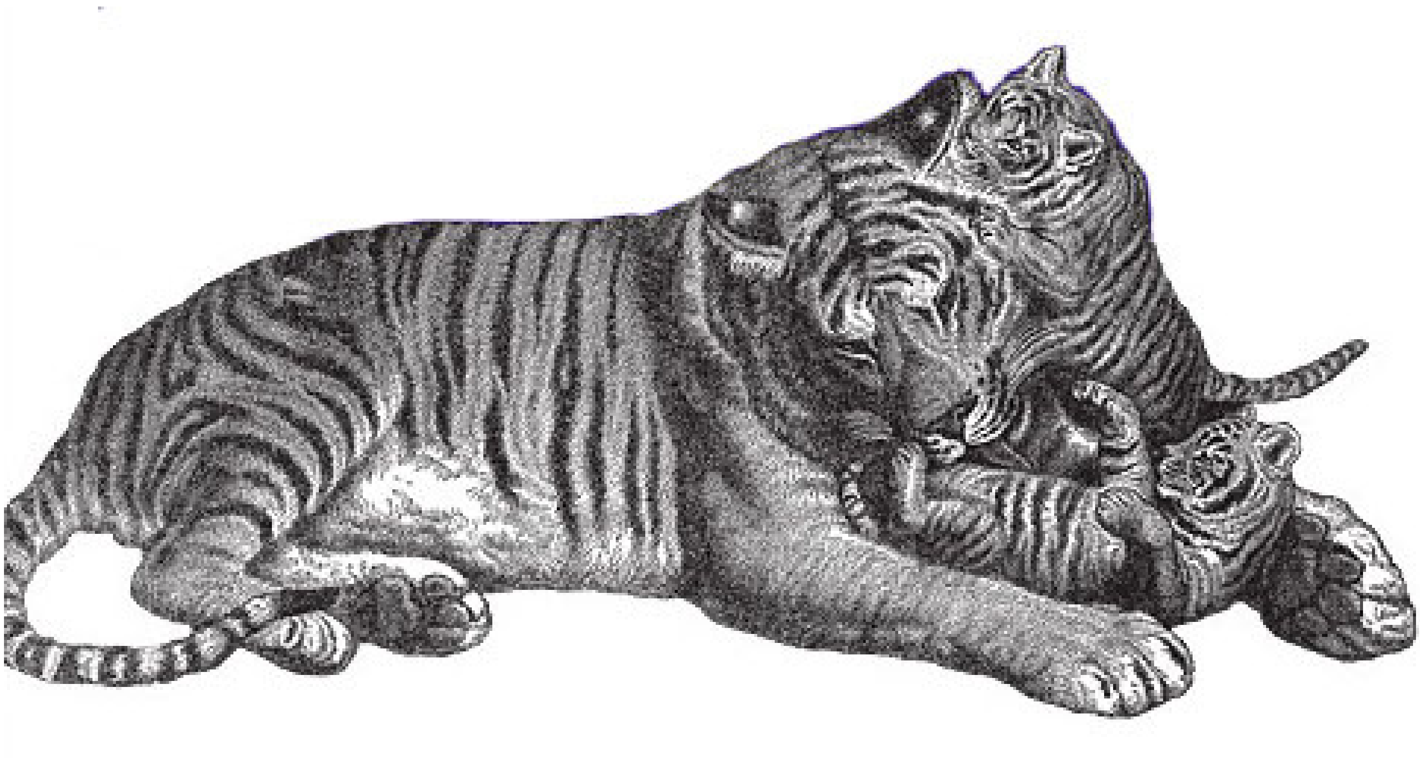
- MAC addresses provide a unique hardware address for a physical device
  - 48 bits total
  - 24 bits are an organizationally unique identifier (OUI) administered by the IEEE
    Identifies vendor
  - 24 bits assigned by vendor
- Since network addresses are used to route traffic, need to map network to MAC addresses.
- Address resolution protocol (ARP) is used by TCP/IP.

- Start by assuming a device knows the source and destination address of the data it needs to send.
- From this it can tell whether the destination is on the same network or a remote network.
- When it needs to send data to a device on the same network it needs to find the destination MAC address.
- Checks its ARP table — if address is there, done.
- If not stored, the device broadcasts the destination IP address to every device on the network.
- The device that has a matching address sends its MAC address.
- Source device stores this correspondance and then sends the data.

- When a device needs to send data to a device on a remote network it looks for the MAC address of the default gateway, just as above.
- The gateway forwards the data on to the destination network.
- The destination gateway uses ARP to find the MAC address of the destination device.

## Threads

- Often we need to deal with *concurrency*
  - Different bits of code running more or less independently in time.
- Once upon a time these had to be separate processes.
  - Rather heavyweight.
- A more modern approach is that of *threads*
- These provide fine-grained concurrency *within* a process.
- Here we discuss the basic ideas behind the use of threads in Java.



- This material is drawn from *Learning Java*, Patrick Niemeyer, Jonathan Knudsen, O'Reilly.

## What are threads

- A thread is a flow of control within a program.
  - Like processes, but threads within a program can easily share state.
- Threads are like classes at Brooklyn College (or anywhere).
  - Separate entities
  - Share resources
  - Only one entity uses a resource at the same time
  - Need coordination to manage access to resources.
- Threads also have local data that is distinct.

## `Thread` and `Runnable`

- All execution in Java is associated with a `Thread` object.
  - That is what `main()` launches.
- New threads are born when a new instance of:

  ```
  java.lang.Thread
  ```

  is created
- This object is what we manipulate to control and coordinate execution of the thread.
- To allow us to do this manipulation, we create an object that implements the:

  ```
  java.lang.Runnable
  ```

  interface.

- `Runnable` is simple.
- We just have to implement one method:

```
public interface Runnable {
    abstract public void run();
}
```

- Every thread then starts by executing the `run()` method of some `Runnable` object.
- To see how we might do this, let’s look at the `ThreadExample` program which is on the class webpage.

```
class ThreadOne implements Runnable{

   Thread myThread;

   public ThreadOne(){
      myThread = new Thread(this);
      myThread.start();
   }

   public void run(){
      while(true){
      System.out.println("One!");
      }
   }
}
```

- Creating an instance of this class will start a thread that executes the code in `run()`.
  - We tell the thread which `run()` to execute by the call we make to the thread's constructor.
- Note that this is not the only way to set up a thread, but it is the only one we will cover.

## Scheduling

- The complete project `SimpleExample` has another object `ThreadTwo` which prints out `Two!`.
- With my Java implementation, when you run the program, it prints out `One!` for a while, and then prints out `Two!` for a while.
- This suggests that in my Java implementation, threads are *time-sliced*.
- Each one runs for a while in some order (it seems that the first thread to be started is the first one to run).
- On other Java implementations, you might get different behavior.
- All the specification says is as follows.

- All threads have a priority value.
- Any time a higher priority thread becomes runnable, it preempts any lower priority threads and starts executing.
- By default, threads with the same priority are scheduled round-robin.
- This means that once a thread begins to run it continues until:
  - It sleeps due to a `sleep()` or `wait()`;
  - It waits for the lock for a `synchronized` method;
  - It blocks on I/O;
  - It explicitly yields control using `yield()`; or
  - It terminates.

## Controlling threads

- There are a few methods that allow us to control the execution of threads.
- We have already used `start()`.
- `stop()`, `suspend()` and `resume()` are deprecated.
- We will discuss `sleep()`, `wait()` and `notify()`.
- There are also `join()` and `interrupt()`.

# `sleep()`

- Sometimes we need to tell a thread to take a break.
- The method `sleep()` will do this.
  - It takes an argument that is the number of milliseconds to sleep for.
- `sleep()` is a class method of `Thread`, so it can be called either using:

  ```
  Thread.sleep()
  ```

  or by calling it on a specific instance of `Thread`:

  ```
  myOwnLittleThread.sleep()
  ```

- The project `ThreadExWSleep`, on the class webpage, is a version of `ThreadExample` where the body of the two threaded objects is altered like this:

```
public void run(){
  while(true){
    System.out.println("One!");
    try{
      myThread.sleep(1000);
    }
      catch(InterruptedException e){
      // Guess we won't get to sleep after all
      }
    }
}
```

- A sleeping thread can be woken up by an `InterruptedException` so we need to specify what to do if this happens.

# Synchronization

- When threads share resources, they need to be *synchronized*
  - Otherwise things can get confusing
- Imagine two threads trying to use the same variable at the same time.
- (In our class example, imagine two classes trying to use the same classroom at the same time).
- Even when you take scheduling (in the Java thread sense) into account it can be an issue.
- Java provides some simple monitor-based methods for controlling access.

- The most basic idea is that of `synchronized` methods.
- Here is an example from `ThreadExWGraphics` (class webpage again) where two threads change the color of an graphical object:

```
public synchronized void changeColor(Color color){
   if (this.theColor != color){
      this.theColor = color;
   }
   repaint();
}
```

- Only one thread at a time is allowed to execute any `synchronized` method of an object.
  - The object is locked.
- Other threads are blocked until they can aquire the lock on the object.

- Note that locks are *reentrant*, so a thread does not block itself.
- It can call itself recursively, and it can call other synchronized methods of the same object.

## `wait()` and `notify()`

- `wait()` and `notify()` provide more direct synchronization of threads.
- When a thread executes a synchronized method that contains a `wait()`, it gives up its hold on the block and goes to sleep.
- The idea is that the tread is waiting for some necessary event to take place.
- Later on, when it wakes up, it will start to try to get the lock for the synchronized object.
- When it gets the lock, it will continue from where it left off.

- What wakes the thread up from `wait`ing is a call to `notify()` on the same synchronized object.

- The project `ThreadWNotifyWait` illustrates these functions.
- This is like the previous example, except that `ThreadOne` and `ThreadTwo` call `wait()` when they have moved the ellipse.
- They don't get control back until `ThreadThree` calls `notify()`.

- While there is lots more to know about threads, that is all we will cover.
- I'll hand out some material on threads next time.

## Summary

- This lecture started our section on networking.
- It introduced some basic ideas about networks from a software perspective:
  - Mainly dealt with the OSI model
- It also talked a little bit about threads.
- Next week we will get into the support that Java provides for network programming.