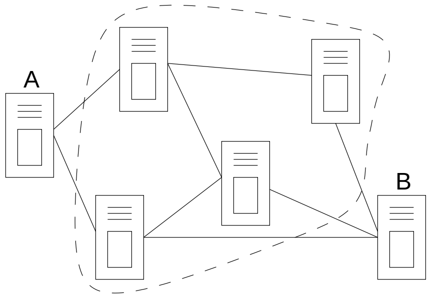
The Internet

The Internet looks like this:



The computers are known as *hosts*. Between hosts there are network links.

We are concerned with the application level: program at A talks to program at B.

We do not care about the network in between (dashed line).

Properties of Hosts

- A host has one or more *hostnames*.

  A hostname is a string referring to a host.

  Hostname examples: `eddie.cdf.utoronto.ca`, `www.cdf.utoronto.ca`, `marvin.cdf.utoronto.ca`

- A host has one or more *IP addresses*.

  An IP address is a 32-bit number assigned uniquely to a network adapter.

  Often written as 4 octets in dotted notation.

  IP address examples: 128.100.31.4, 128.100.31.3

- A host has 65536 *ports*.

  A port is an attachment point for applications.

  Server program listens to a port on its host.

  Client program connects to that port on that host.

Using Internet in Java

Java presents a simplified model of application level networking.

The classes are in `java.net.*`.

- `InetAddress`

  Represents an IP address. Maps hostnames to IP addresses.

- `ServerSocket`

  Allows a server program to attach to a port and listen to it.

- `Socket`

  Represents a connection between two programs.

  Also allows a client to connect to a server program on a given host at a given port.

InetAddress

Need to look up IP address by hostname.

Static methods of `InetAddress` for this:

- `static InetAddress getByName(String hostname) throws UnknownHostException`

  Returns an IP address of the host.

- `static InetAddress[] getAllByName(String hostname) throws UnknownHostException`

  Returns all IP addresses of the host.

- `static InetAddress getLocalHost() throws UnknownHostException`

  Returns an IP address of the local host.

Can substitute IP address in dotted notation for hostname. E.g.,

```
InetAddress.getByName("eddie.cdf.utoronto.ca")
InetAddress.getByName("128.100.31.4")
```

InetAddress (cont)

An `InetAddress` object has these methods:

- `String getHostAddress()`

  Returns the IP address in dotted notation.

- `byte[] getAddress()`

  Returns the 4 bytes of the IP address, in *network byte order*.

- `String getHostName()`

  Returns the hostname of the IP address.

- `boolean equals(Object o)`

  True iff `o` represents the same 4-byte IP address (based solely on the result of `getAddress`).

- `String toString()`

  Returns a string containing the hostname and the IP address.

Socket

A `Socket` object represents an established connection.
Constructors set up connections:

- `Socket(String hostname, int port)`
  `throws UnknownHostException, IOException`

  Connects to the given host at the given port.

- `Socket(InetAddress address, int port)`
  `throws IOException`

  Connects to the host at the given address, at the given port.

How to use a socket:

- `OutputStream getOutputStream() throws IOException`

  Gives the output byte stream of the connection.

- `InputStream getInputStream() throws IOException`

  Gives the input byte stream of the connection.

Socket (cont)

- `InetAddress getInetAddress()`

  `int getPort()`

  Find out whom I am talking to.

- `void close() throws IOException`

  Closes the connection.

ServerSocket

A `ServerSocket` object represents an attachment to a port.
Note: It does *not* represent a connection!
Constructors:

- `ServerSocket(int port, int backlog)`

  Attaches to the given port. If 0 is given, lets the system choose. A backlog queue of the given size is also set up.

- `ServerSocket(int port)`

  Attaches to the given port. If 0 is given, lets the system choose. The backlog queue has size 50.

To wait for connections from clients:

- `Socket accept() throws IOException`

  Waits for a connection from a client. Returns a `Socket` object representing the connection.

ServerSocket (cont)

Other methods of `ServerSocket`:

- `InetAddress getInetAddress()`

  `int getLocalPort()`

  Find out where I am attached to.

- `void close() throws IOException`

  Detaches.