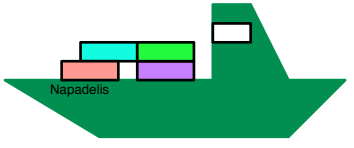
Computer Networking



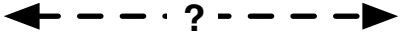
Pierre Rioux

ACE Lab meeting

November 2013

In the old days

Site 1



Site 2

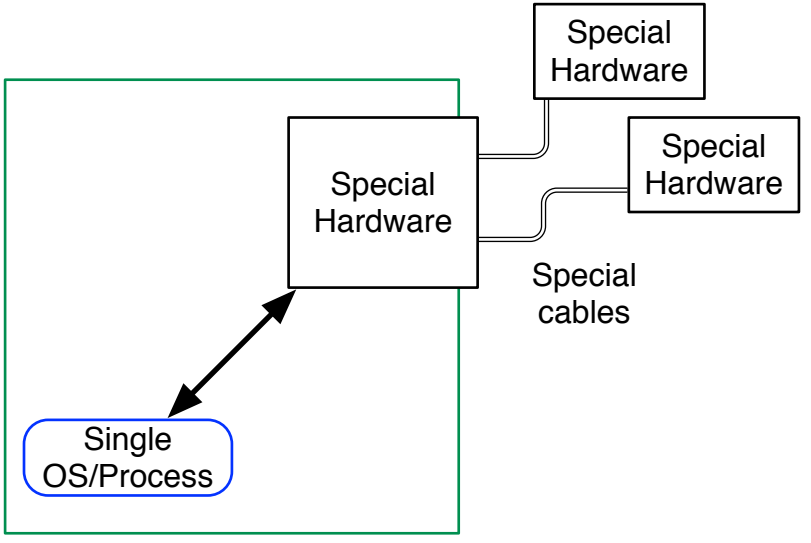
Computers were mostly standalone systems

When there were connections, they were for specific hardcoded services designed by the manufacturer

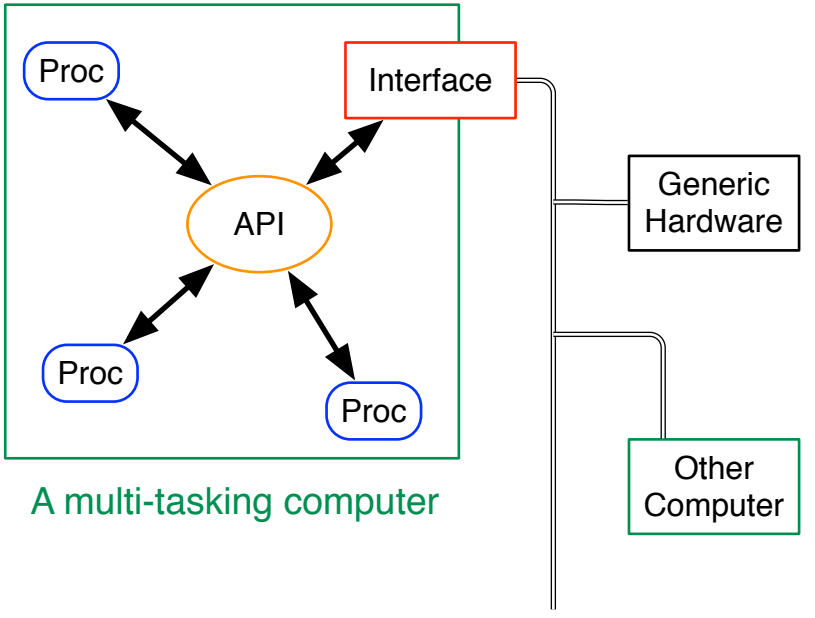
There was a dedicated cable, hardware, and hardware registers for each such service

The connections only worked between computers of the same manufacturer

A generic solution

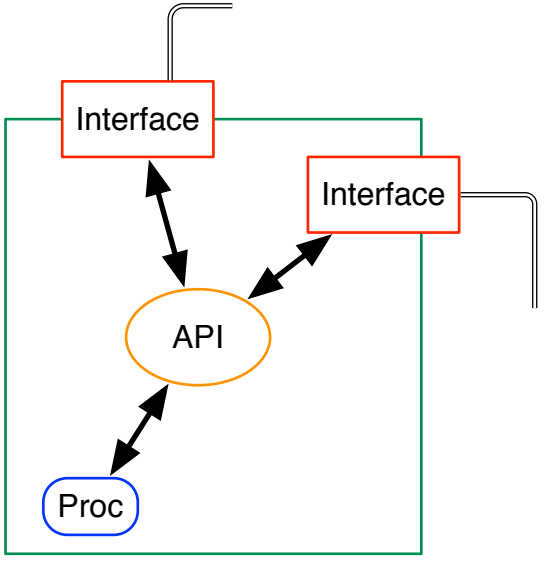


A single-tasking computer

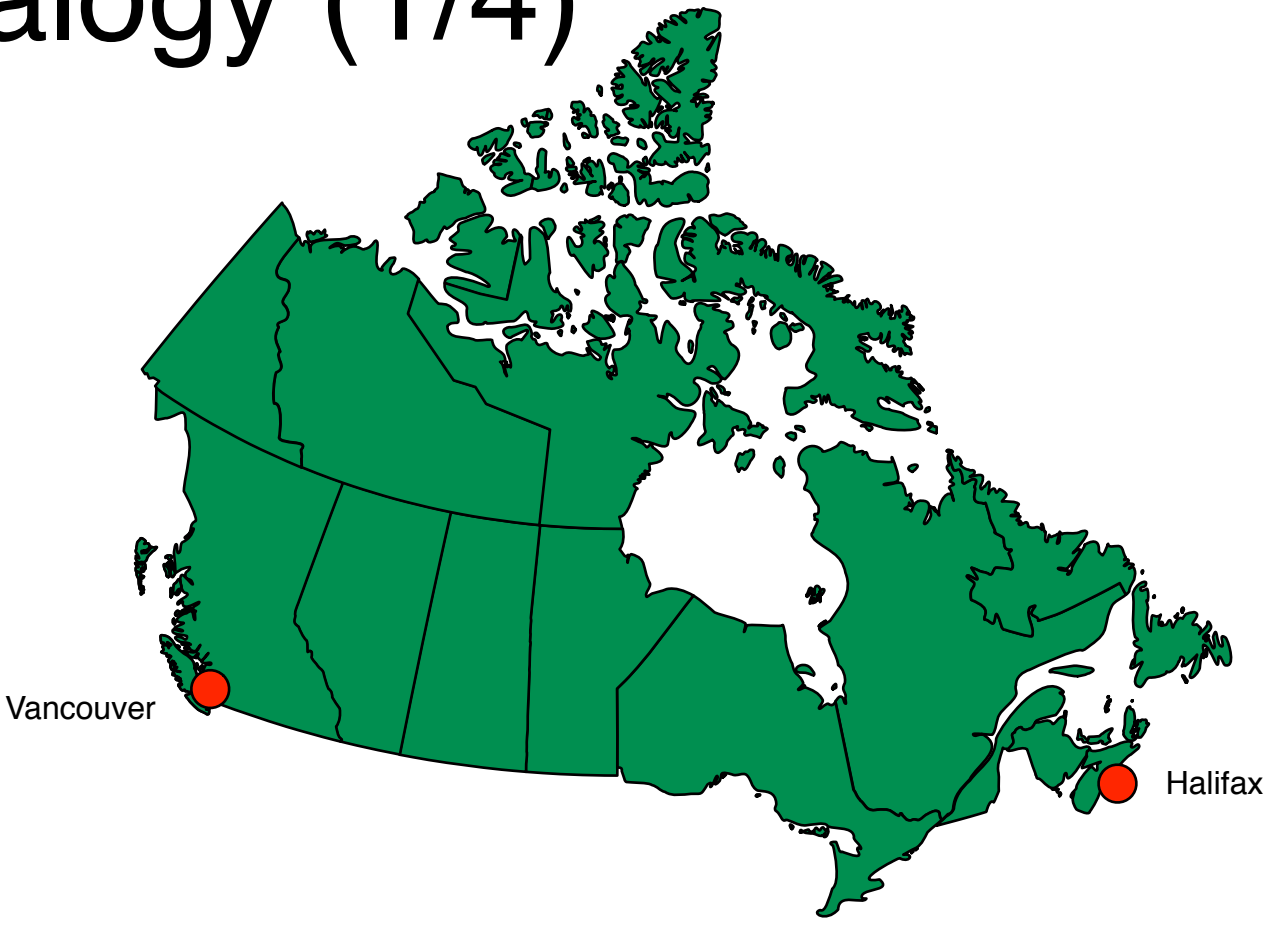


A multi-tasking computer

An analogy (1/4)

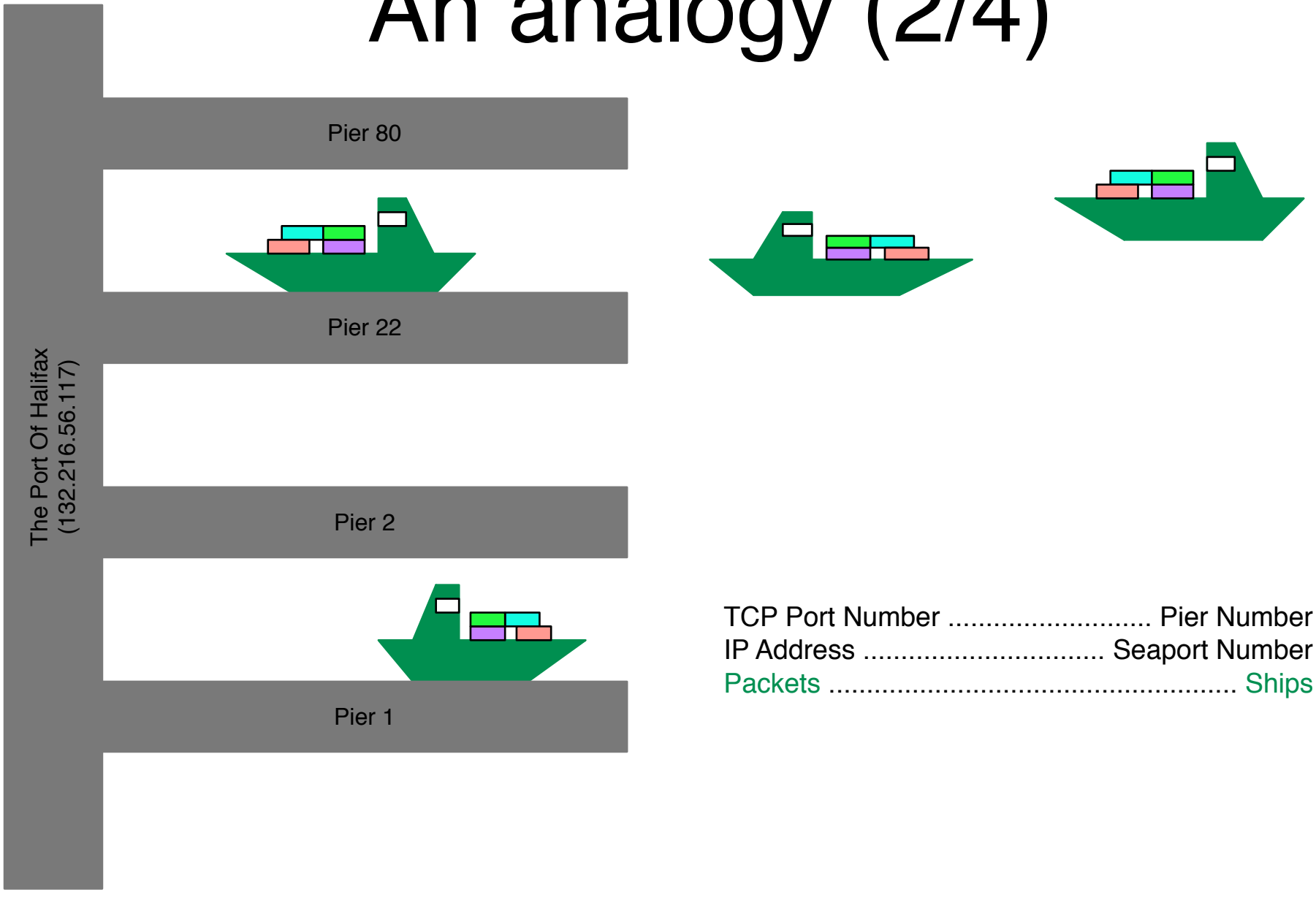


A computer



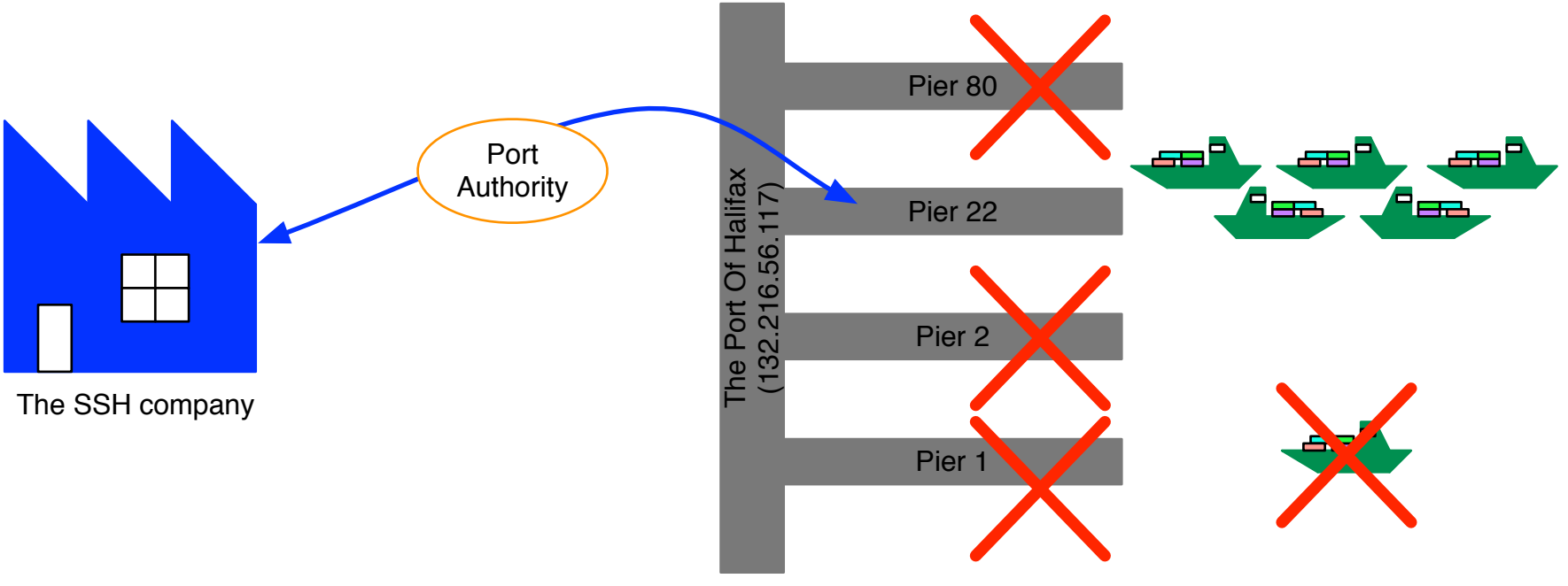
Computer ..... Country
Interface ..... Seaport
API (or OS) ..... Government or Port Authority

An analogy (2/4)



TCP Port Number ..... Pier Number
IP Address ..... Seaport Number
Packets ..... Ships

An analogy (3/4)



Process ..... Company
Data ..... Containers
Stream ..... Series of ships
API ..... Port Authority

An analogy (4/4)

A COUNTRY can have several SEAPORTS, each with several PIERS, some of which are rented to individual COMPANIES.

A COMPUTER can have several INTERFACES, each with numbered TCP PORTS, some of which are dedicated to individual PROCESSES.

SHIPS travel from one specific PIER of one SEAPORT, to another specific PIER of another COUNTRY's SEAPORT.

PACKETS travel from one specific TCP PORT NUMBER of one INTERFACE, to another specific TCP PORT NUMBER of another COMPUTER's INTERFACE.

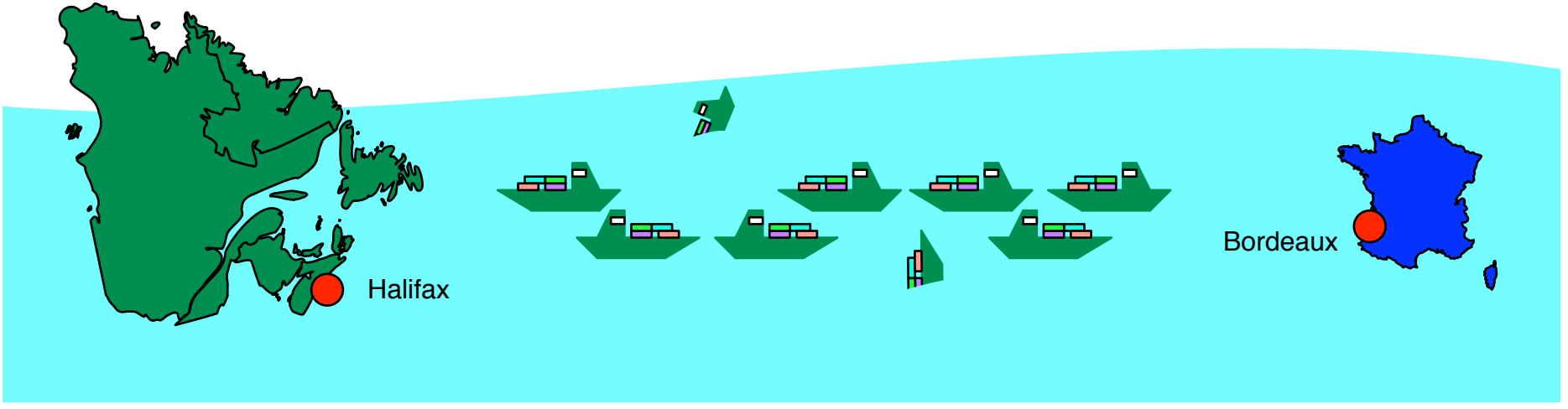
A series of SHIPS provide a steady transport of GOODS.

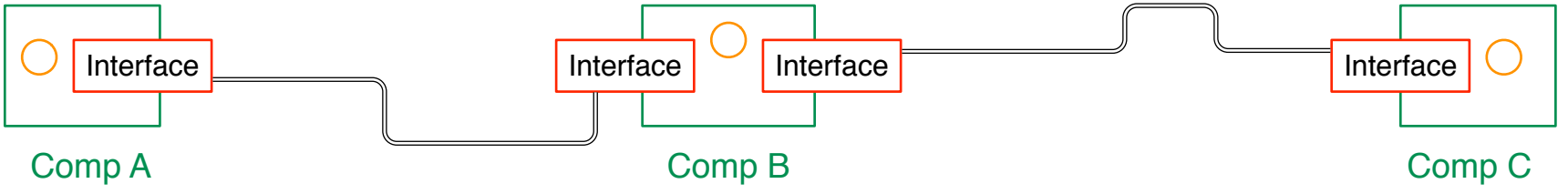
A series of PACKETS provide a steady stream of BYTES.

SHIPS can be lost or damaged at sea.

PACKETS can be lost or damaged in the network.

TCP/IP

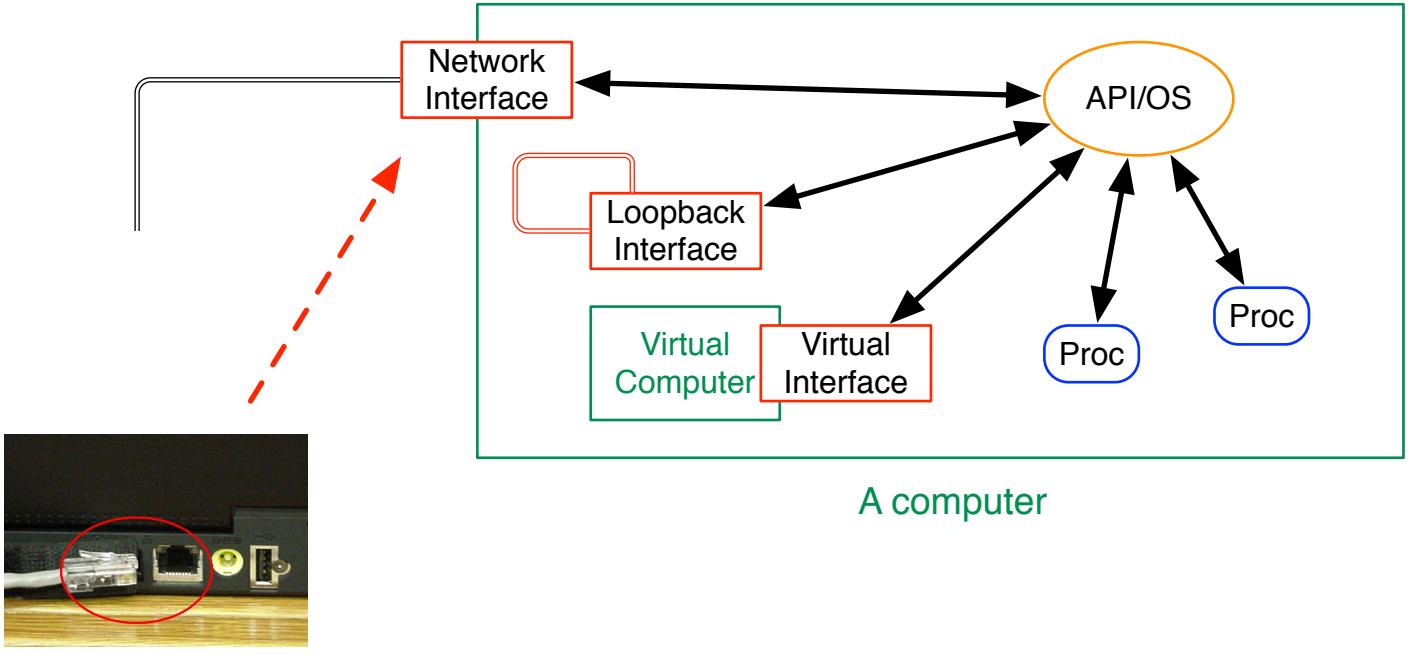




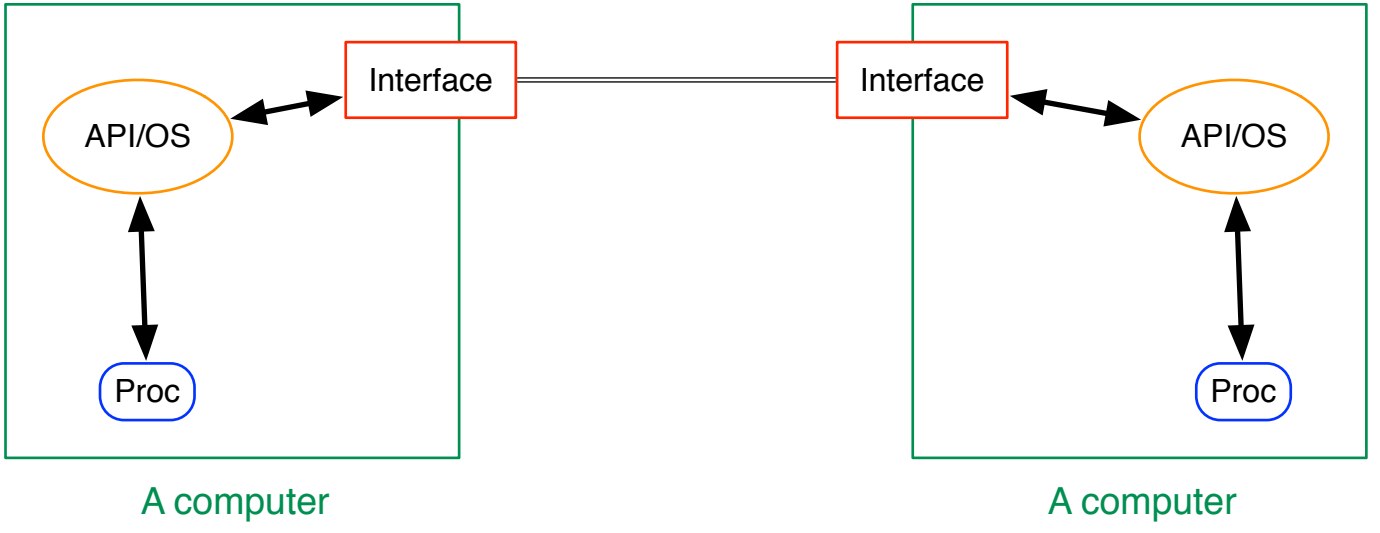
IP (Internet Protocol) is the protocol that knows how to send packets from A to C, but with no guarantee of quality. Implementation of IP is done by the networking software in the OS of each computer.

TCP (Transmission Control Protocol) is the protocol based ON TOP of IP that guarantees a proper stream of bytes using checksums and serial numbers. Applications (processes) typically use TCP.

Types of interfaces

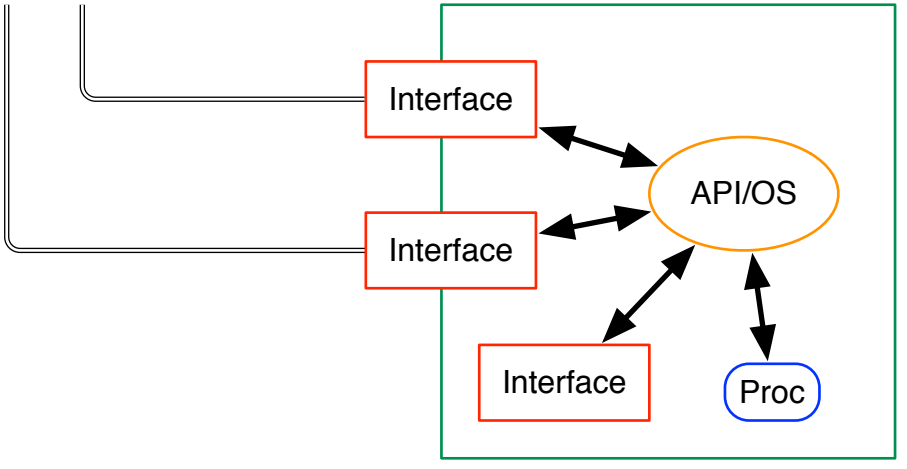


A standard datastream



The final result is highly symmetrical. However, arriving there requires an asymmetrical dialog.

A listener (server) process



A typical listener process will contact the API/OS and ask it to reserve a TCP port number one or several interfaces. Interfaces are chosen by their IP addresses. The process usually then waits for a connection.

Port numbers between 1 and 1023 can only be reserved by processes that belong to root.

Port numbers between 1024 and 65535 are available to all process, as long as they are not already reserved or in use.

(Optional tech details) The system calls, in order:

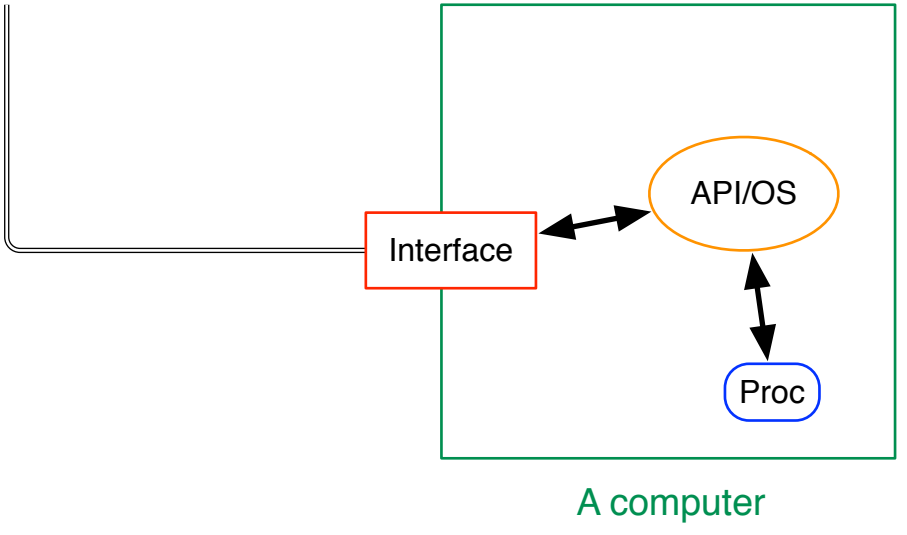
1- socket() : creates the data structure for the connection, and will behave like a file descriptor.

2- listen() : configures the socket for local listening socket (port, binding interface etc).

3- accept() : when a remote connection is detected, creates a new socket to handle it. The original socket remains.

4- read() and write() as usual.

A connector (client) process



A typical client process will contact the API/OS and ask it to setup a connection to a remote IP address, on a particular port number.

The API/OS will negotiate with the remote computer as part of the TCP protocol to make sure there is a listener ready over there. It will also choose which interface is best suited for the connection.

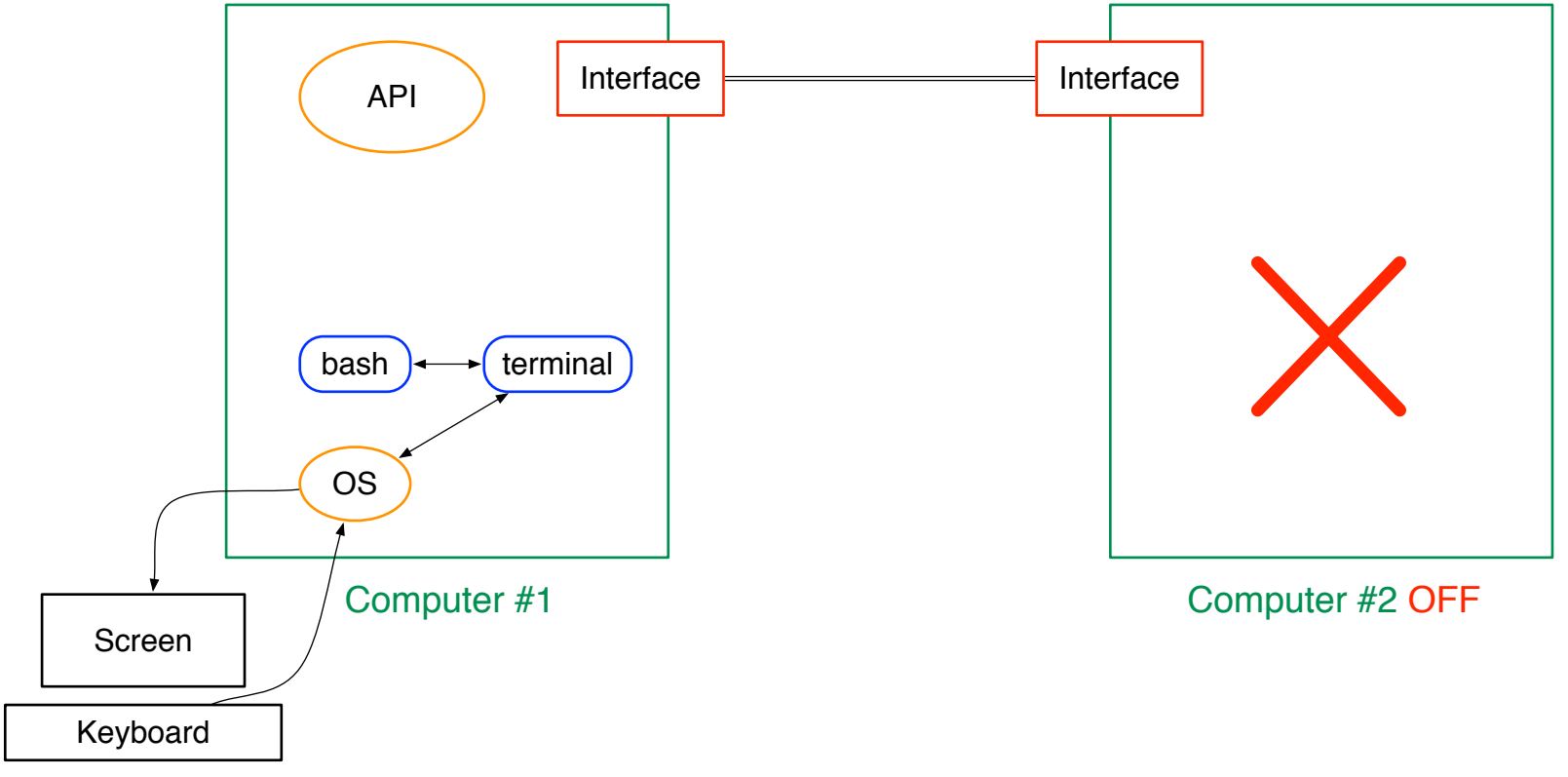
(Optional tech details) The system calls, in order:

1- socket() : creates the data structure for the connection, and will behave like a file descriptor.

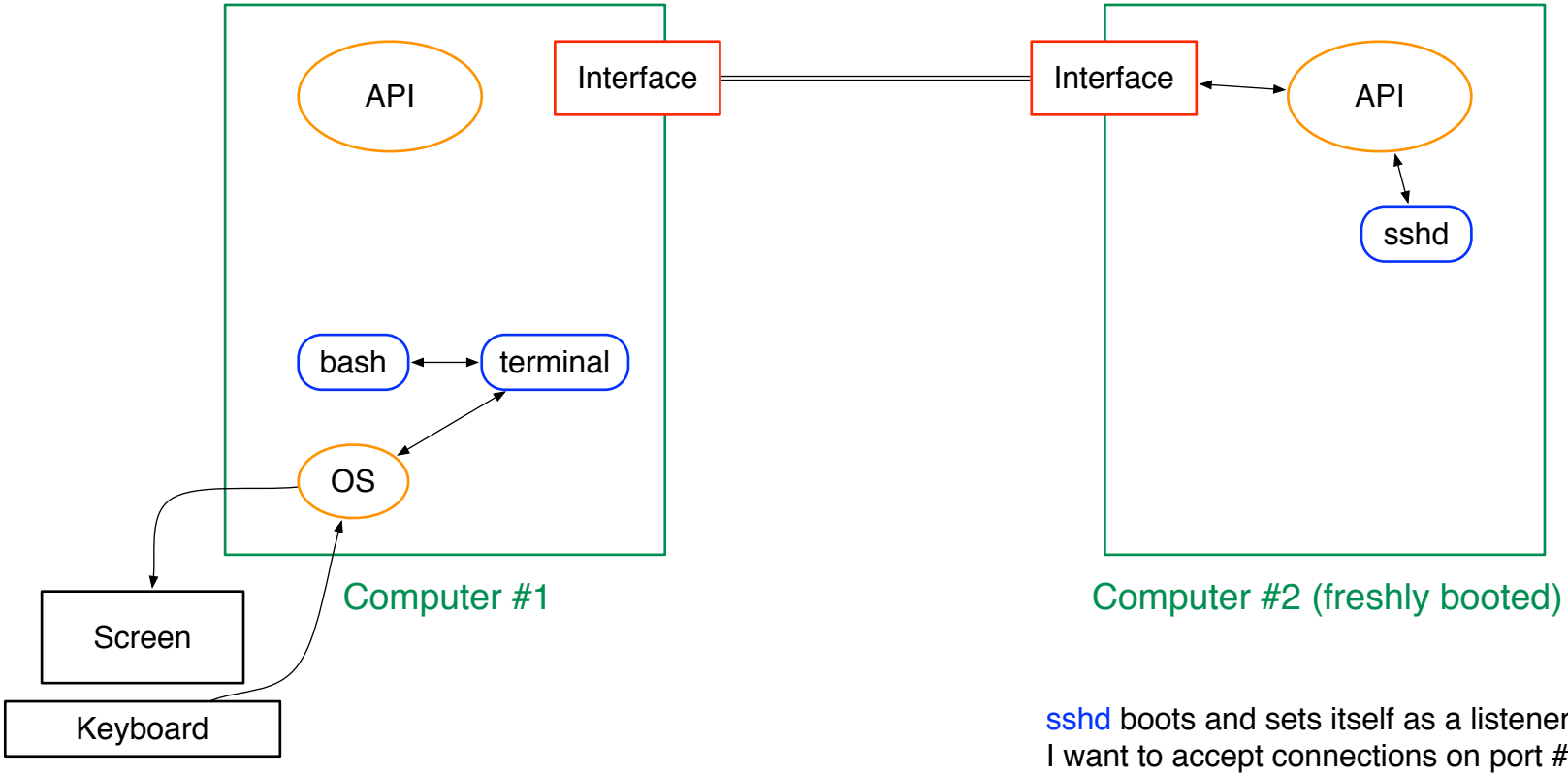
2- connect() : provides the information for the remote destination (IP address, port, etc).

3- read() and write() as usual.

A SSH session

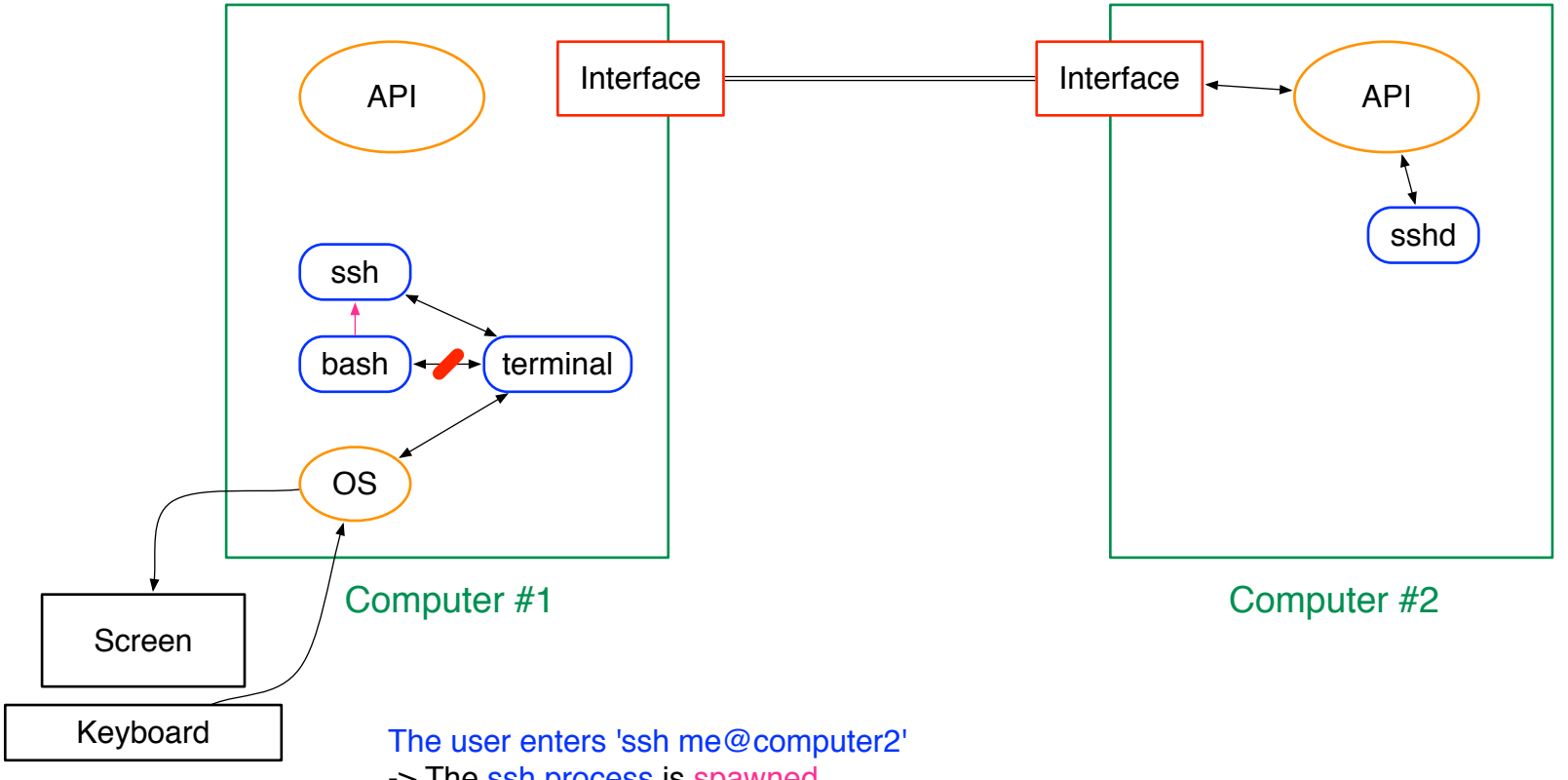


A SSH session 2



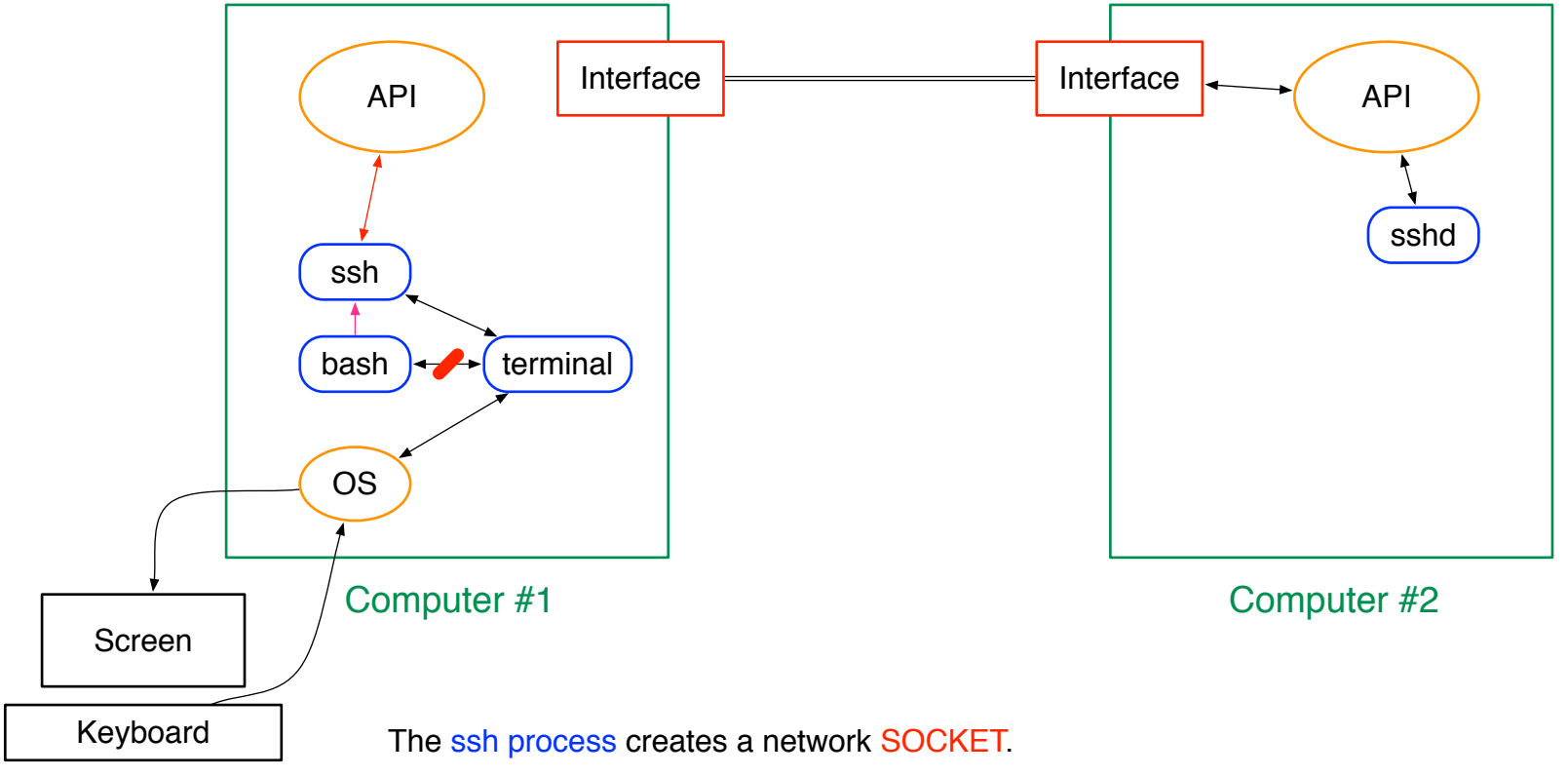
sshd boots and sets itself as a listener:
I want to accept connections on port #22.

A SSH session 3



The user enters 'ssh me@computer2'
-> The ssh process is spawned.
-> bash is suspended, keyboard I/O sent to ssh

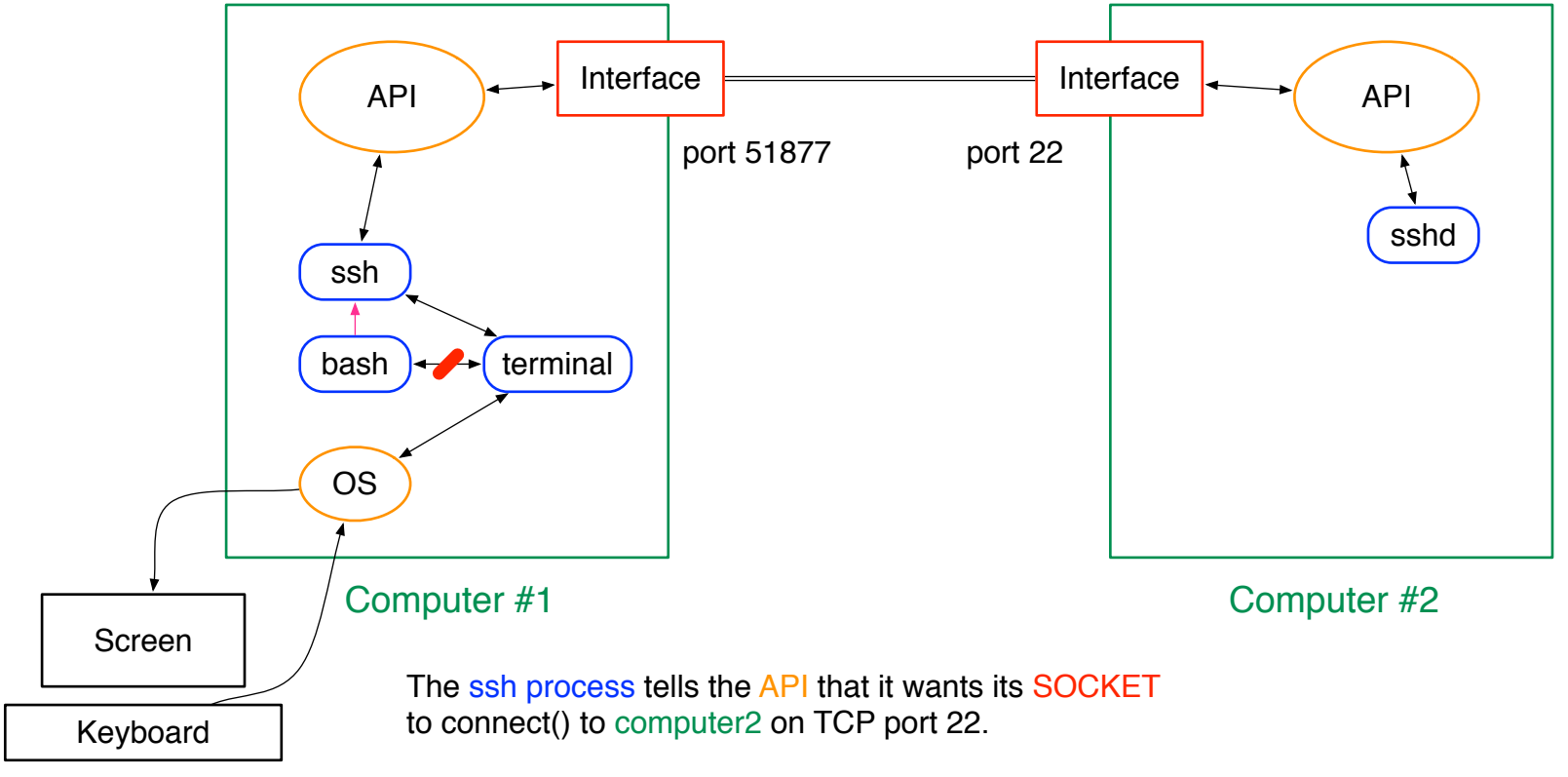
A SSH session 4



The ssh process creates a network SOCKET.

(Optional details) This will appear internally much like any other UNIX file handle, with the ability to read() and write().

A SSH session 5

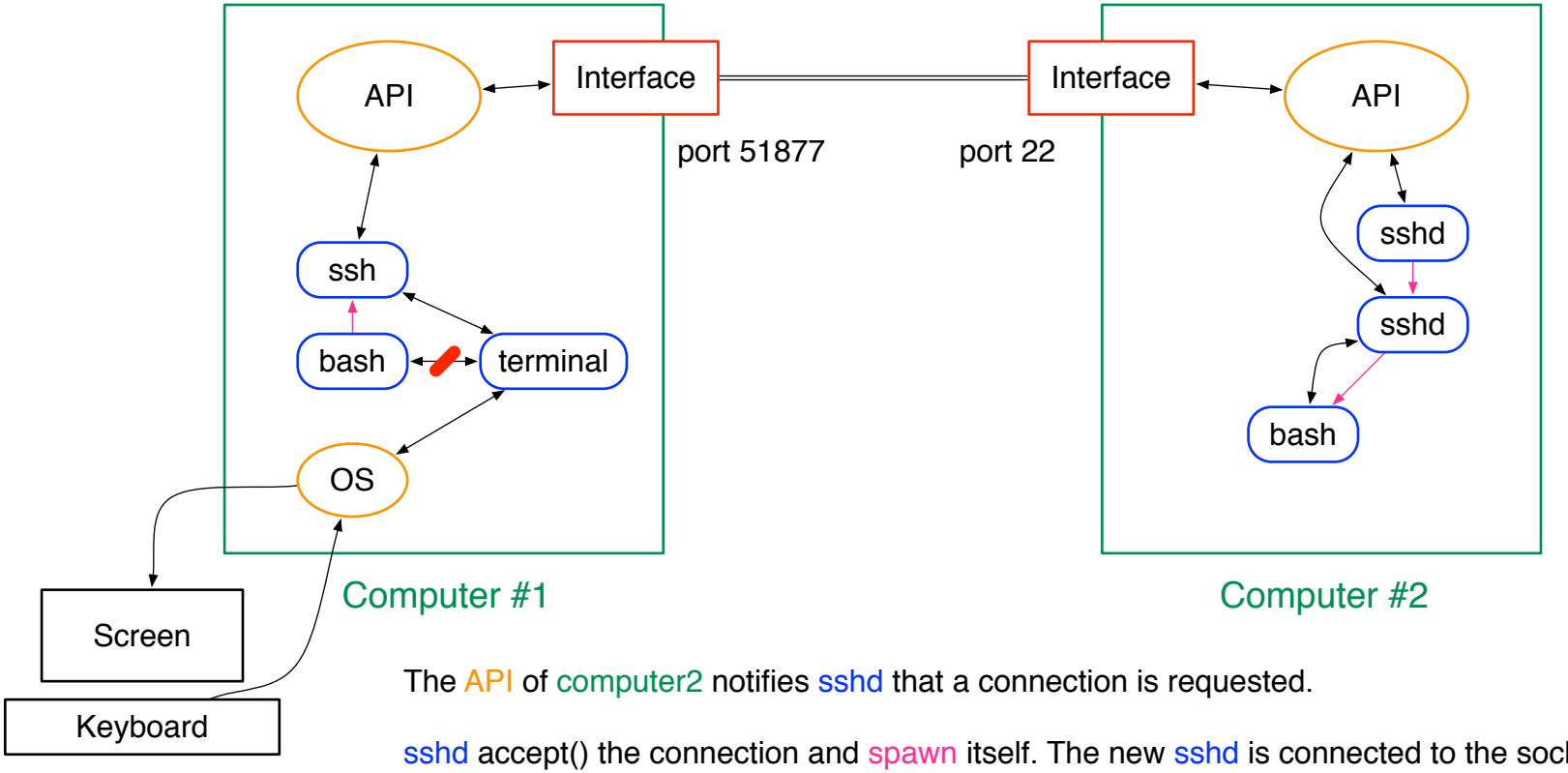


The ssh process tells the API that it wants its SOCKET to connect() to computer2 on TCP port 22.

This triggers a dialog between the two APIs of computer1 and computer2.

As a consequence, an arbitrary free port #51877 is reserved on computer1 for the connection.

A SSH session 6



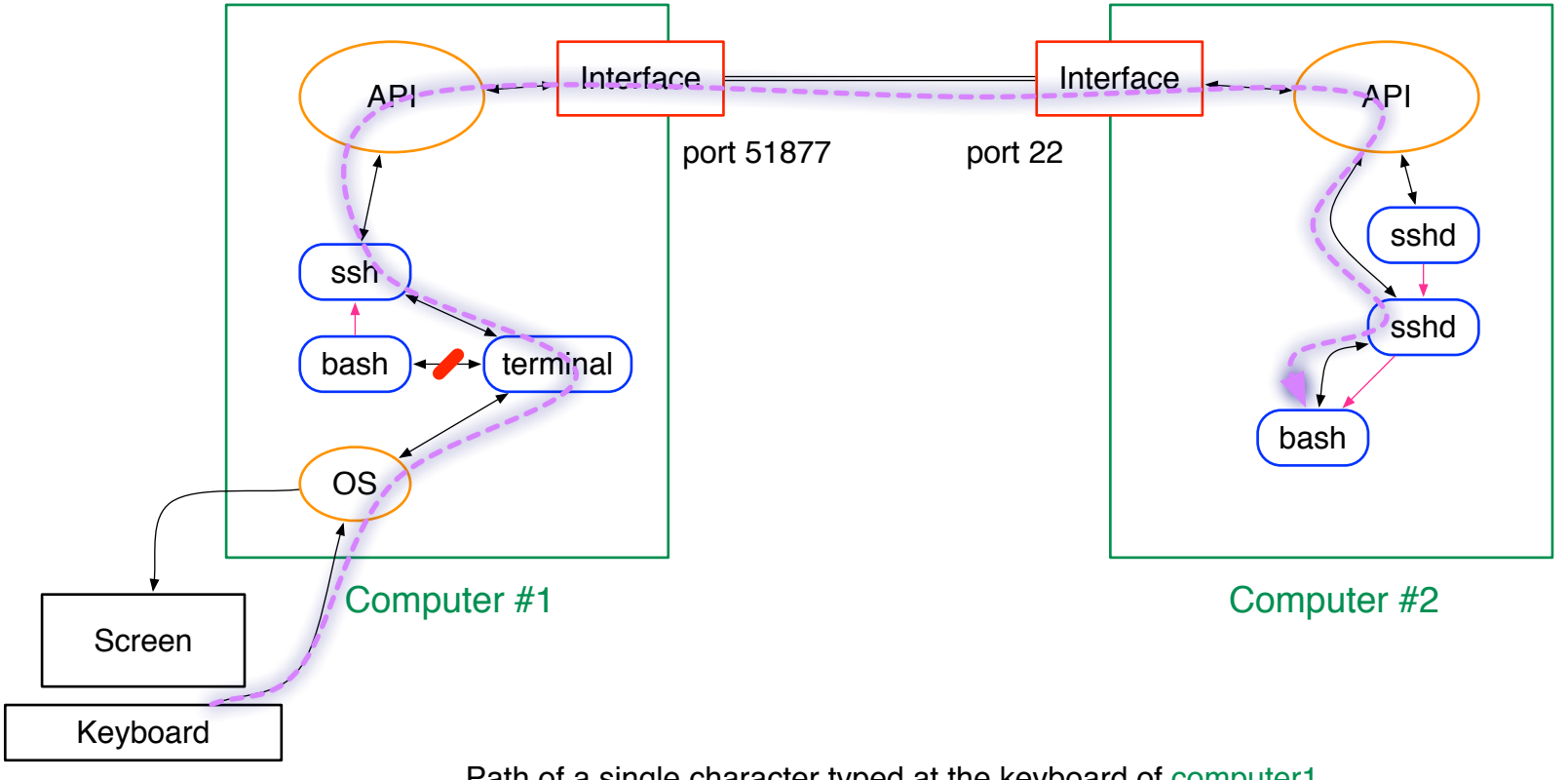
The API of computer2 notifies sshd that a connection is requested.

sshd accept() the connection and spawn itself. The new sshd is connected to the socket returned by accept().

The new sshd spawns a bash and sends and receives unencrypted data to it.

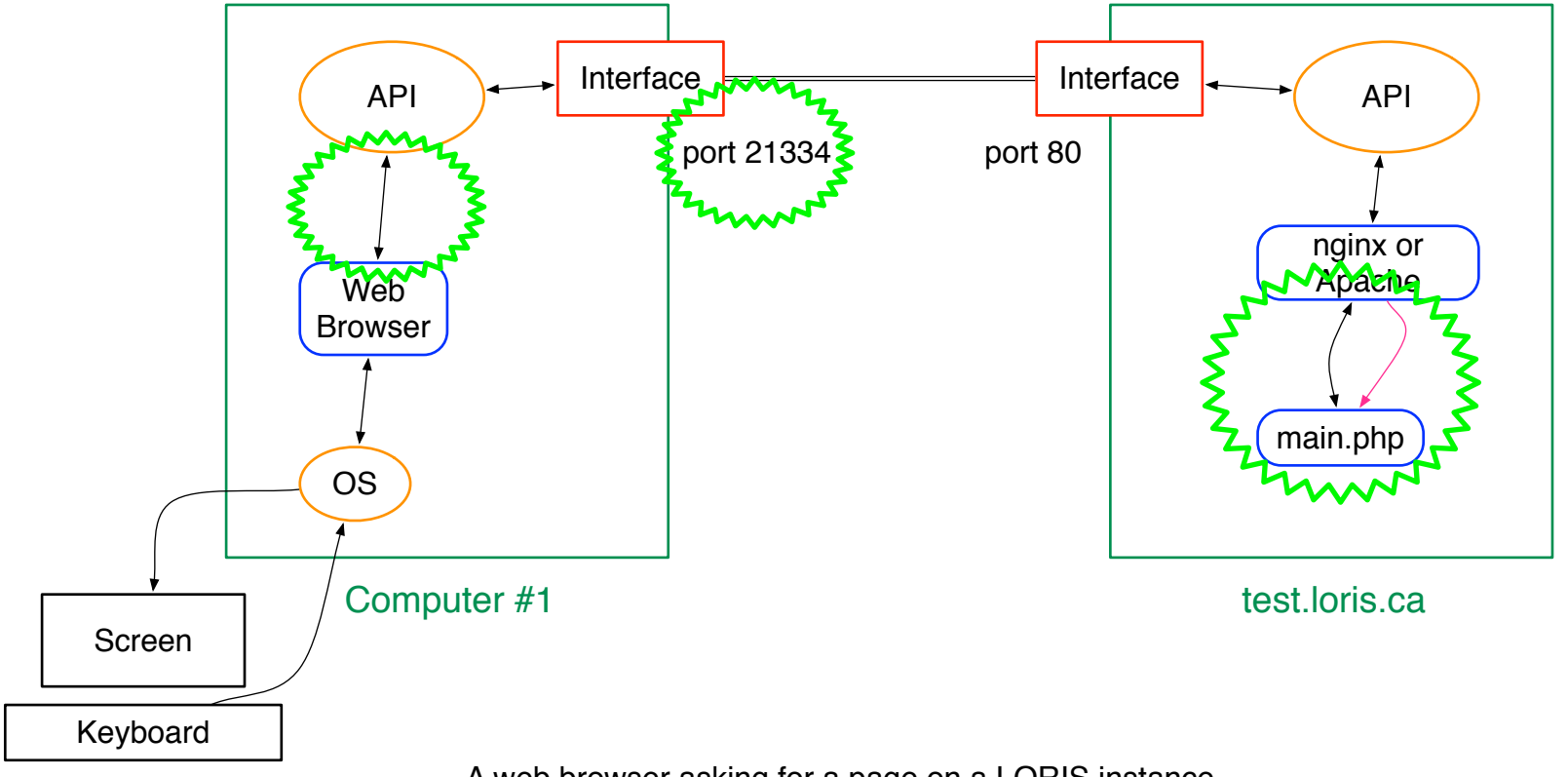
The original sshd can continue listening to new connections.

A SSH session 7



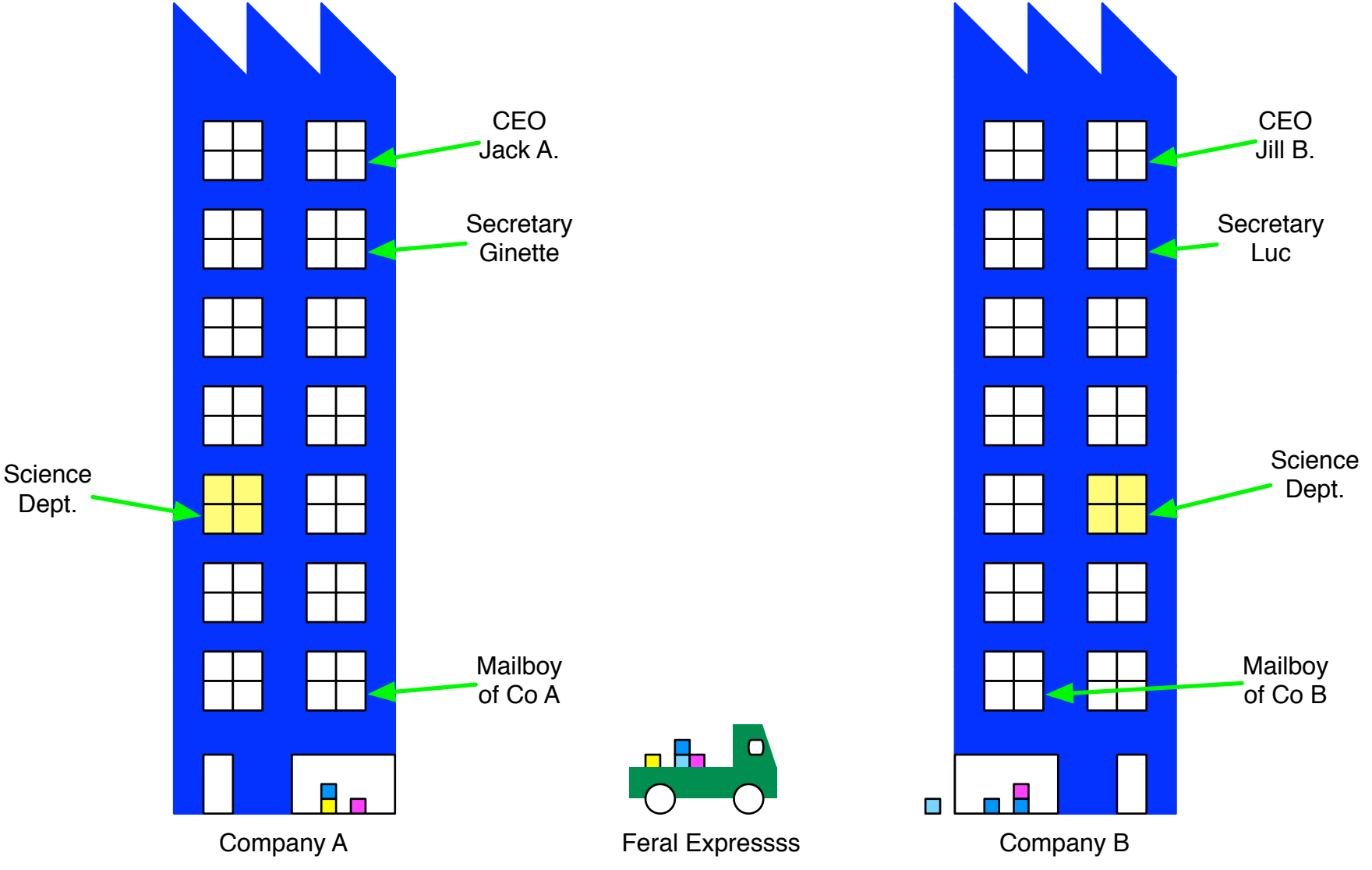
Path of a single character typed at the keyboard of computer1.

A Web Browser

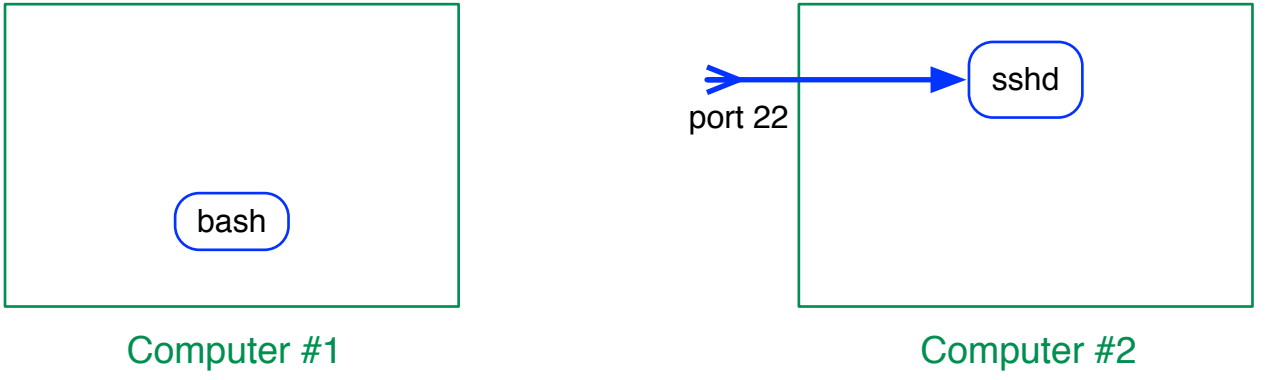


A web browser asking for a page on a LORIS instance.

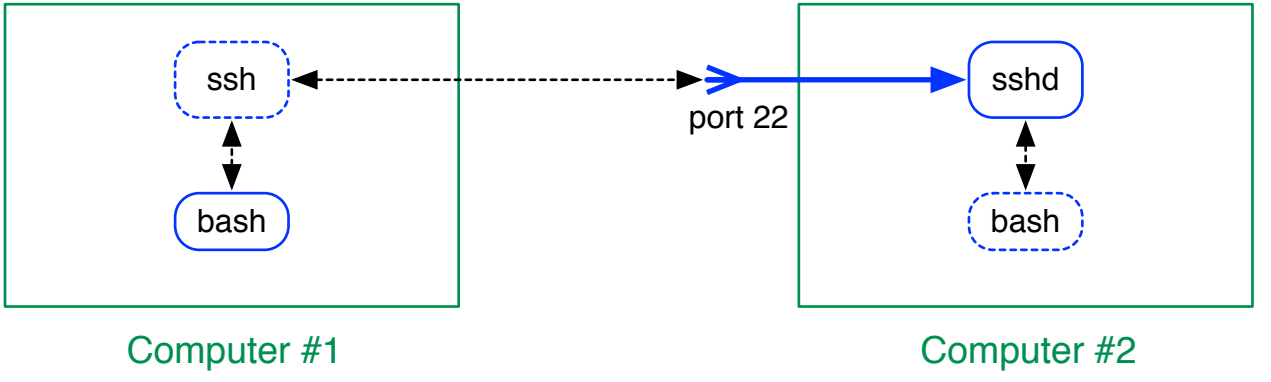
Encapsulation & Tunneling



A Simpler Representation

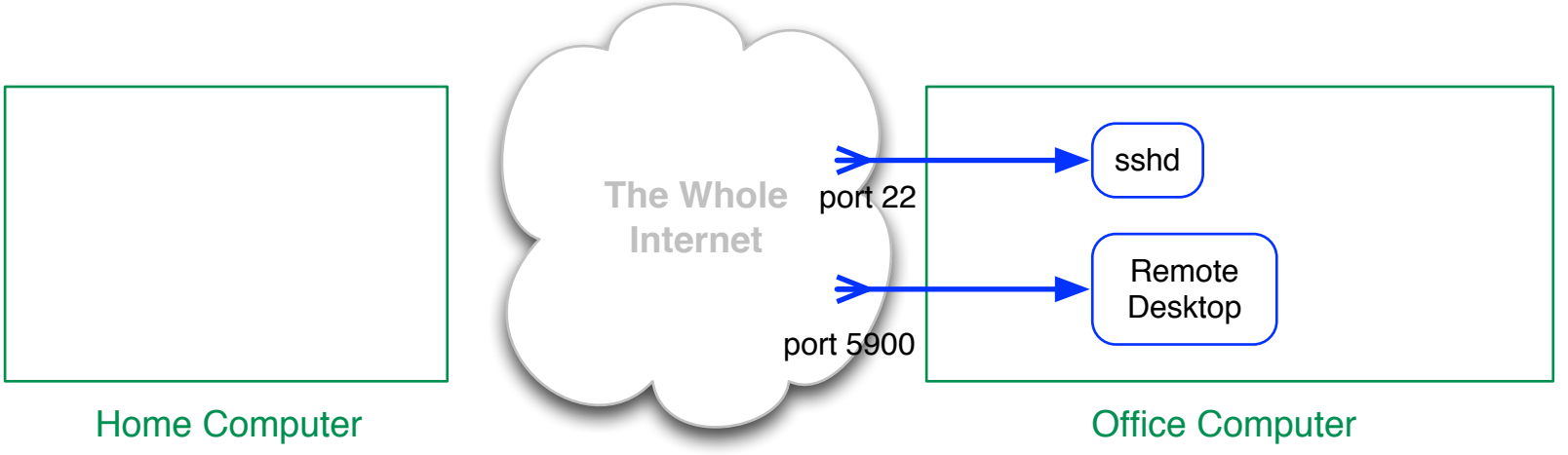


Before a SSH connection is established.

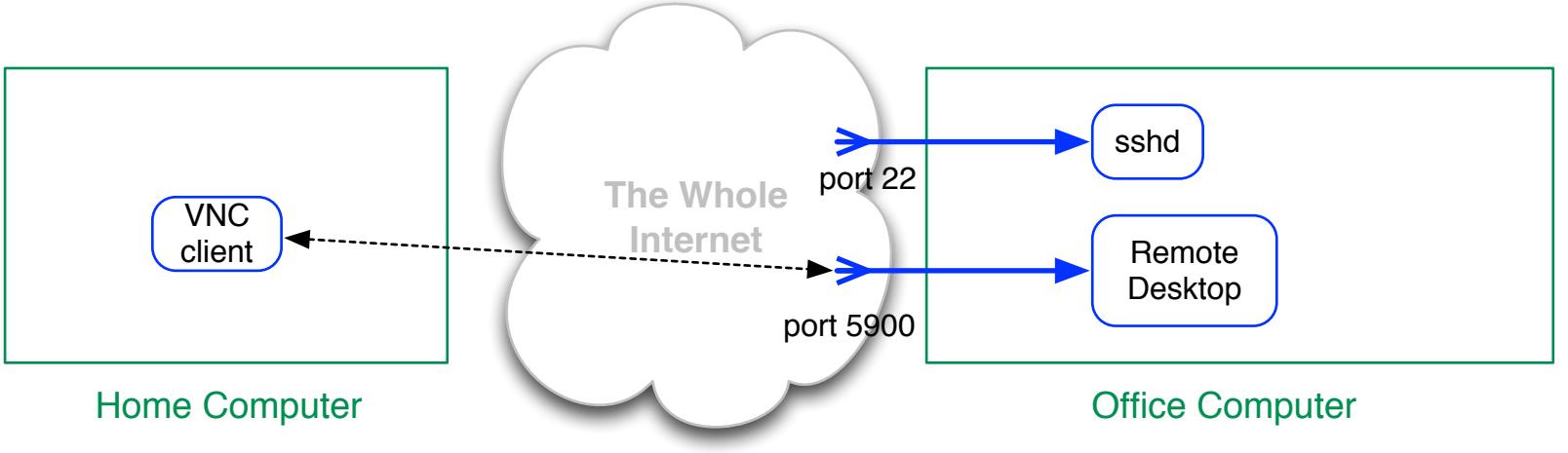


After a SSH connection is established.

Remote Desktop Service

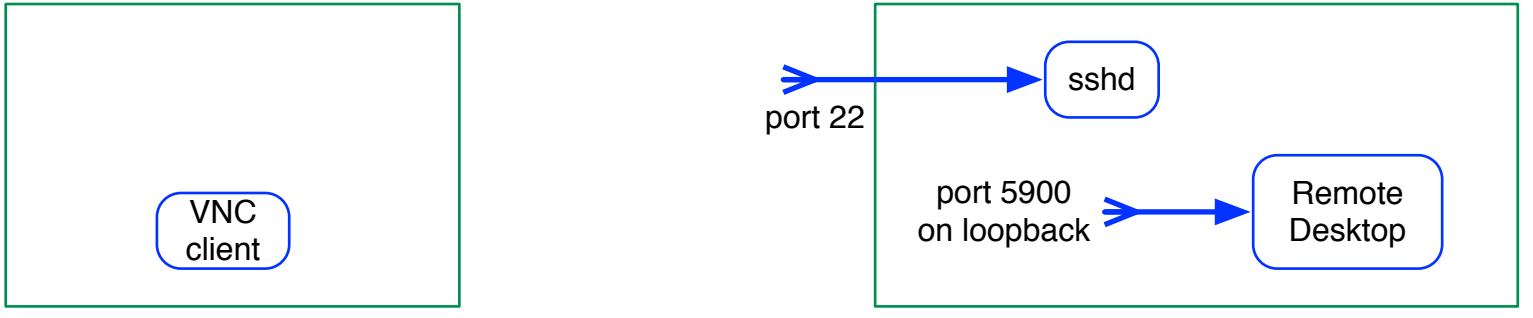


An insecure setup for Remote Desktop



After using a VNC client, the user can see his office computer from home.

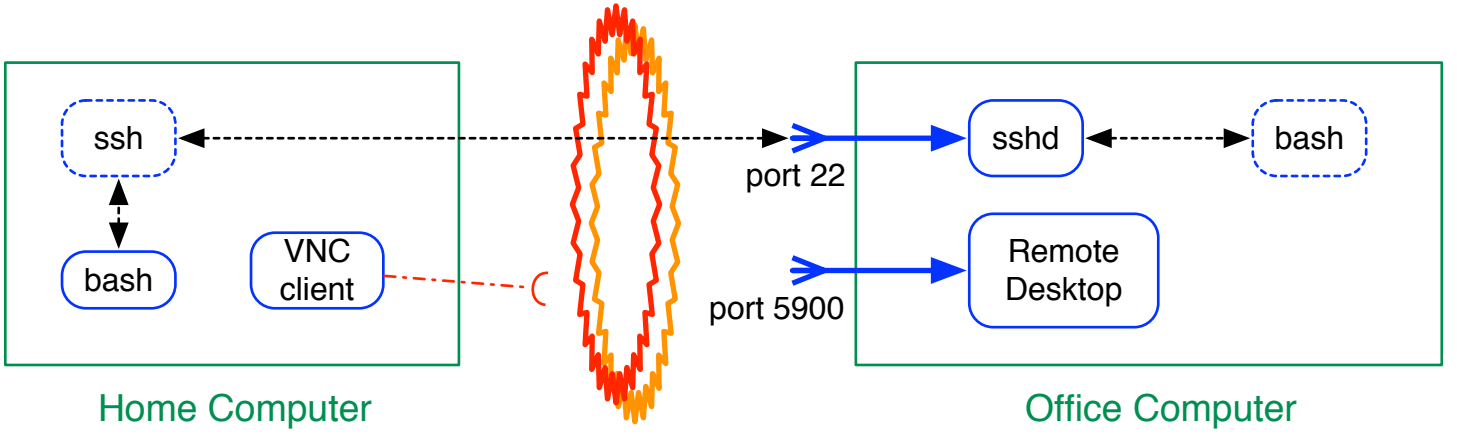
Remote Desktop Service



An more secure setup for Remote Desktop

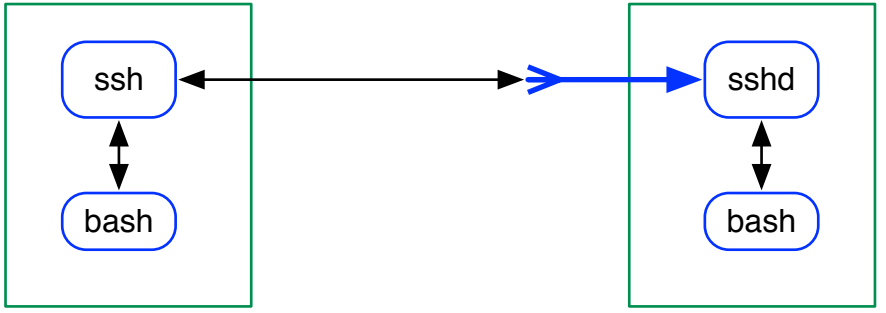
However, why would a remote desktop
server not be open to, huh, remote hosts?!?

Firewall Blocking Port 5900



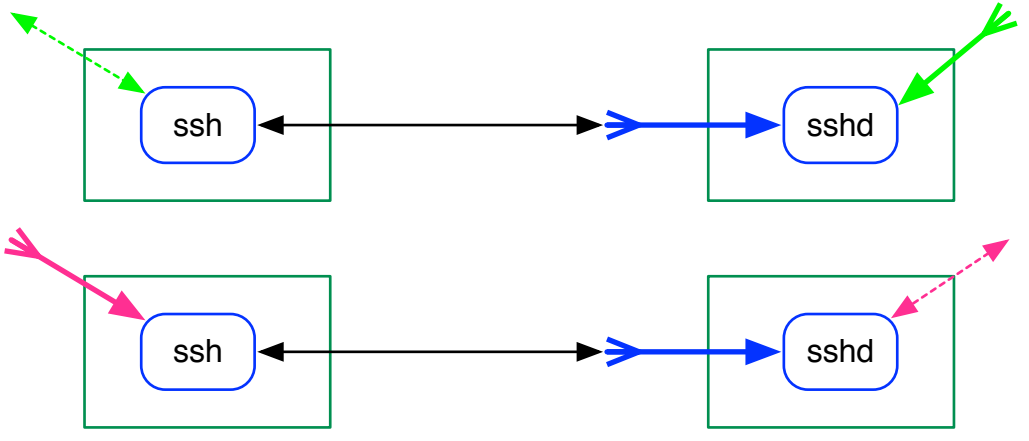
The University Firewall blocks everything except for port 22

SSH Tunneling (1/2)

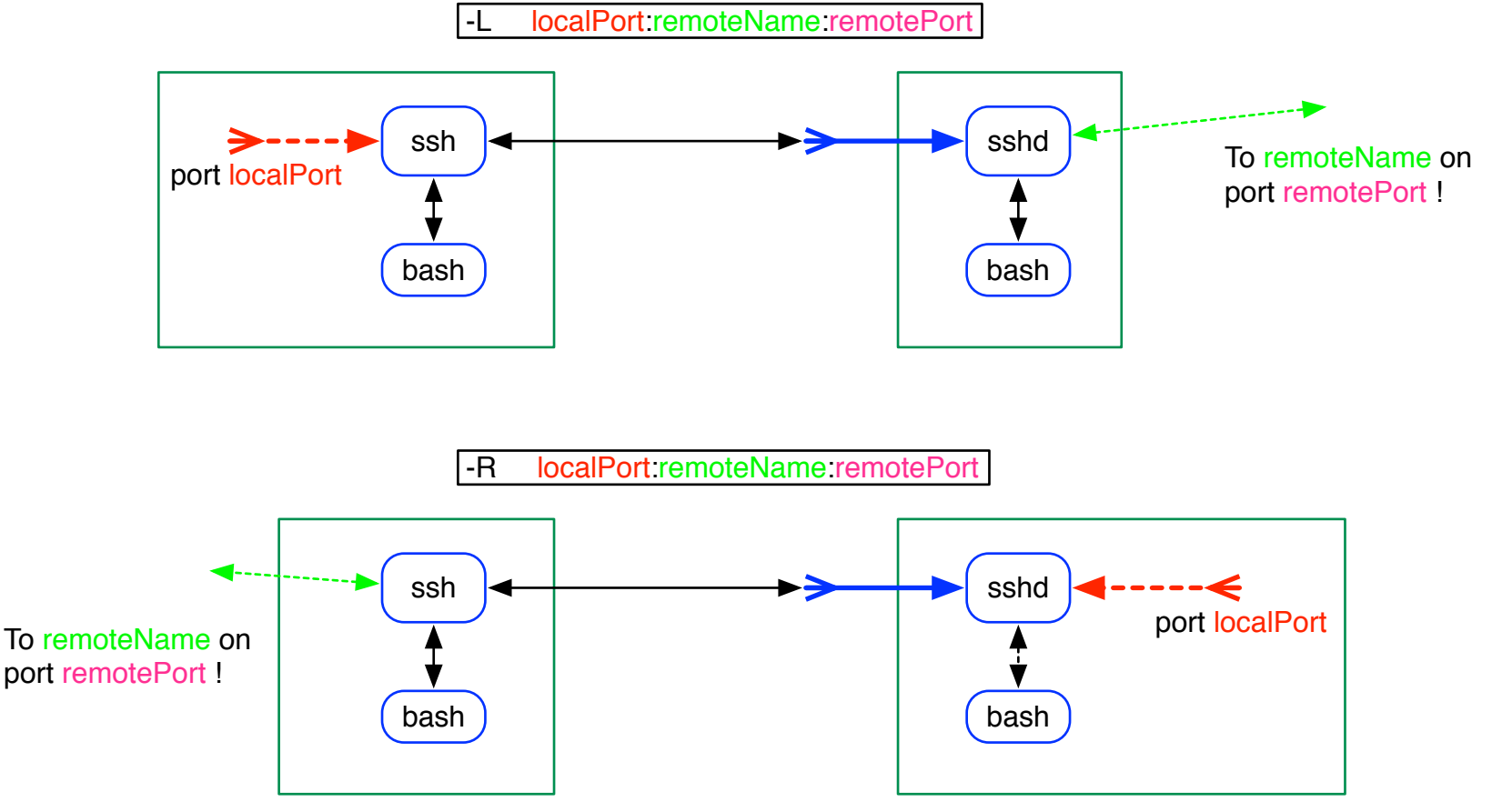


The most abstract and simple representation yet!

1) ssh and sshd talks to each other at this point
2) the basic function is to connect bash to bash
3) a tunnel can be added to sshd ...
4) and/or a tunnel can be added to ssh .

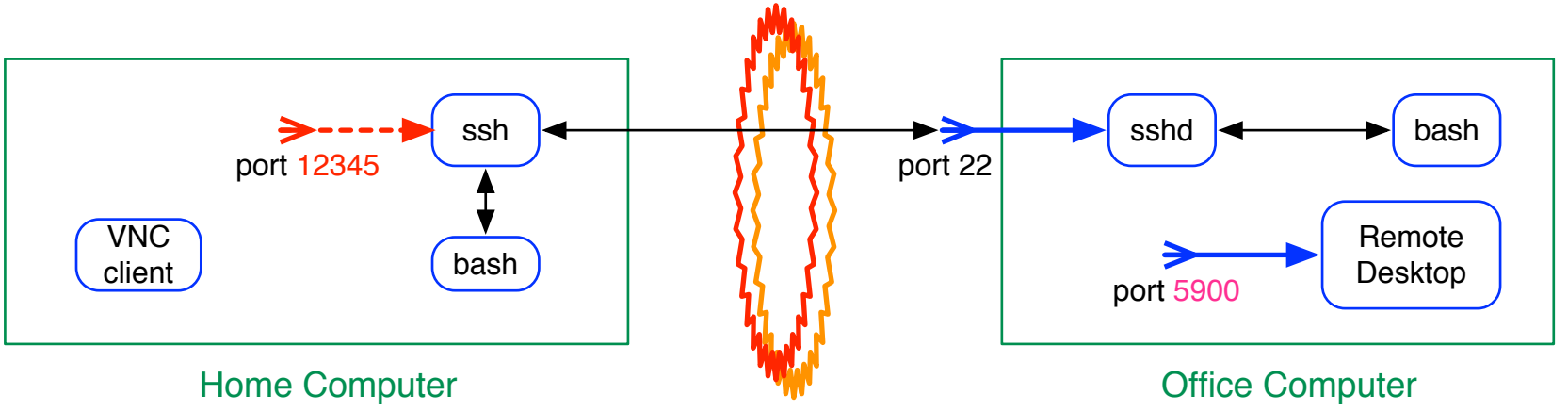


SSH Tunneling (2/2)

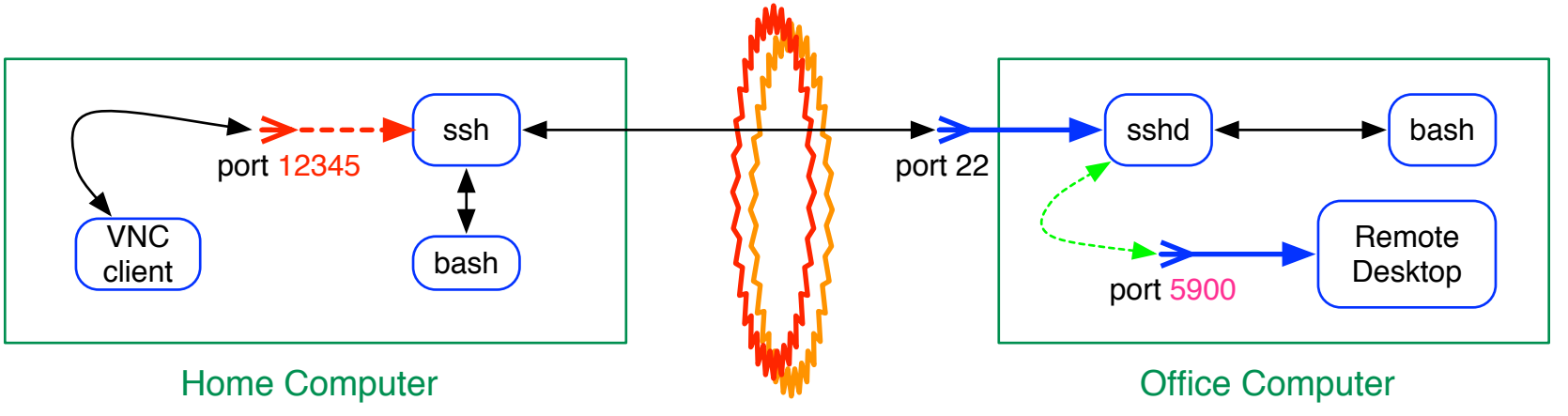


Tunneling Remote Desktop

-L 12345:officecomp:5900

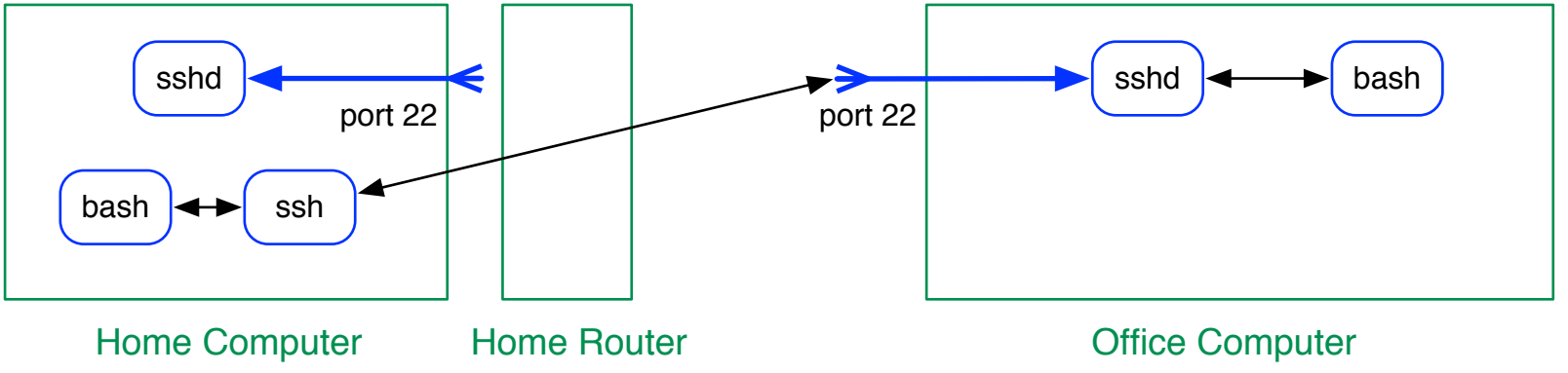


Starting the SSH session with the special option -L

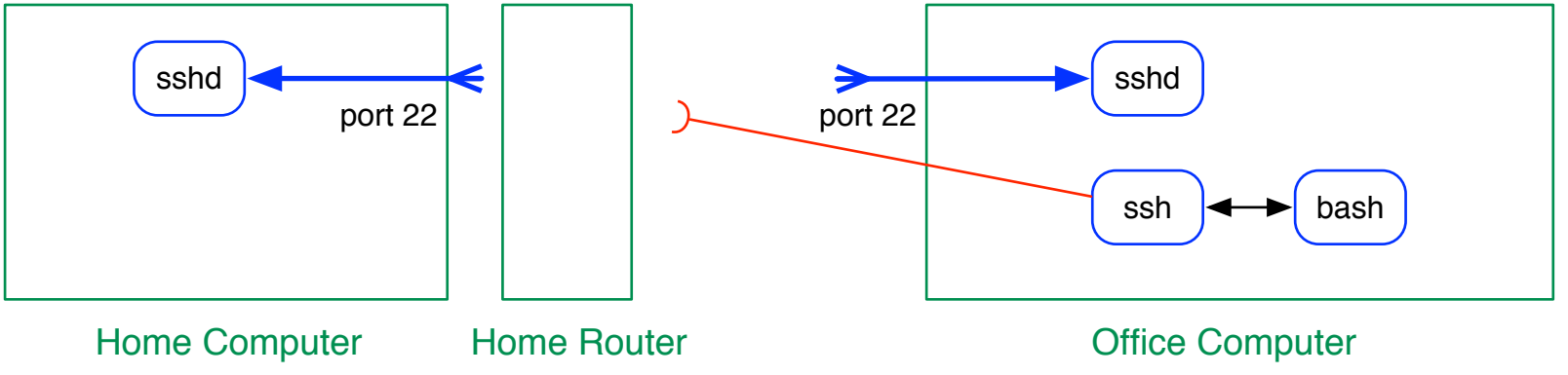


Then starting the VNC client with "localhost" and "12345" as the computer to connect to.

SSH To Home

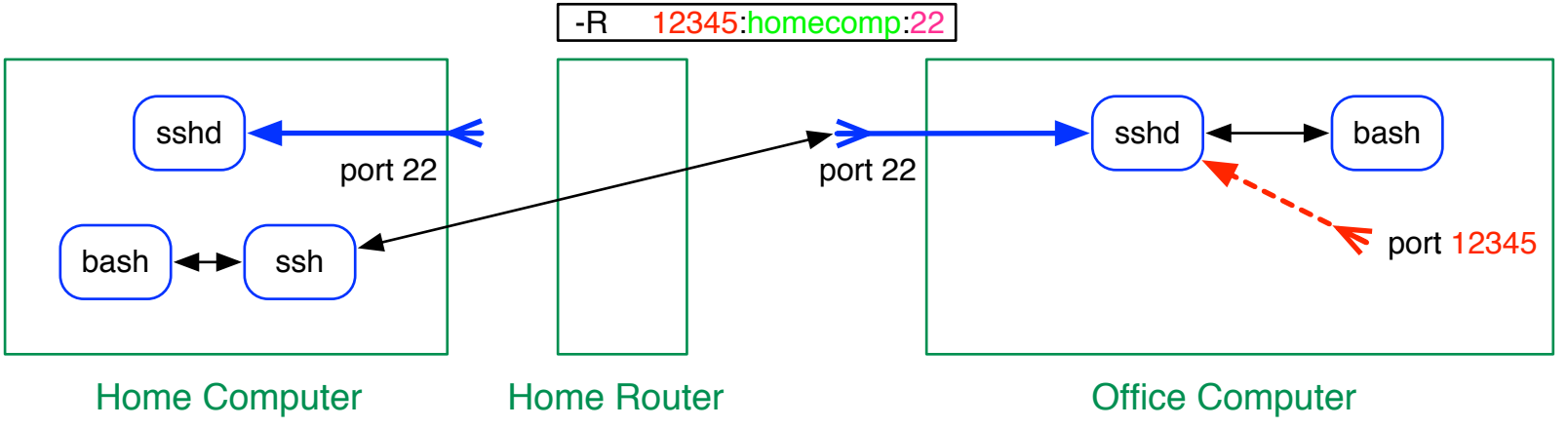


The home router allows you to connect FROM home TO the office

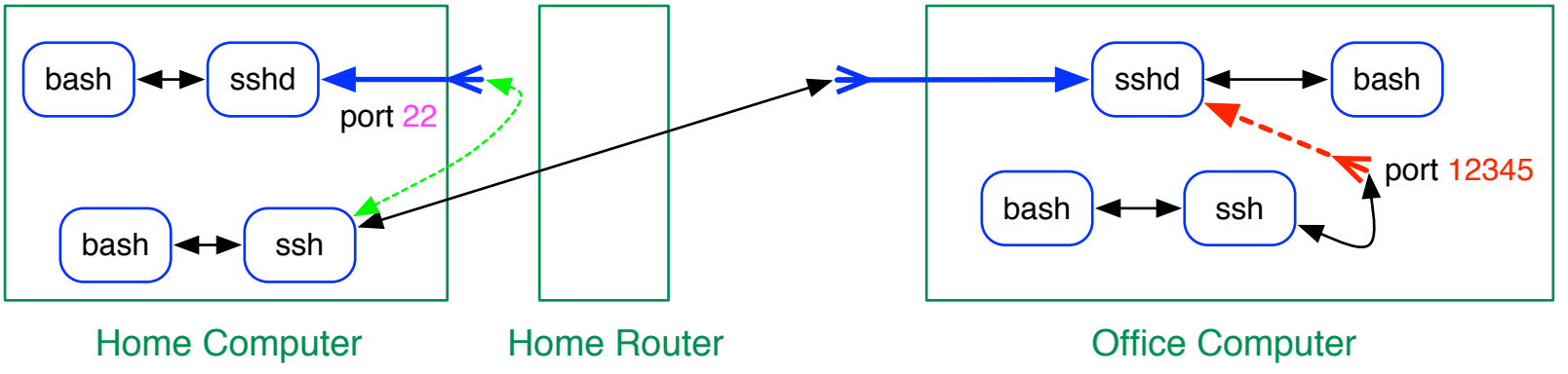


The home router prevents connecting FROM the office TO home

SSH To Home

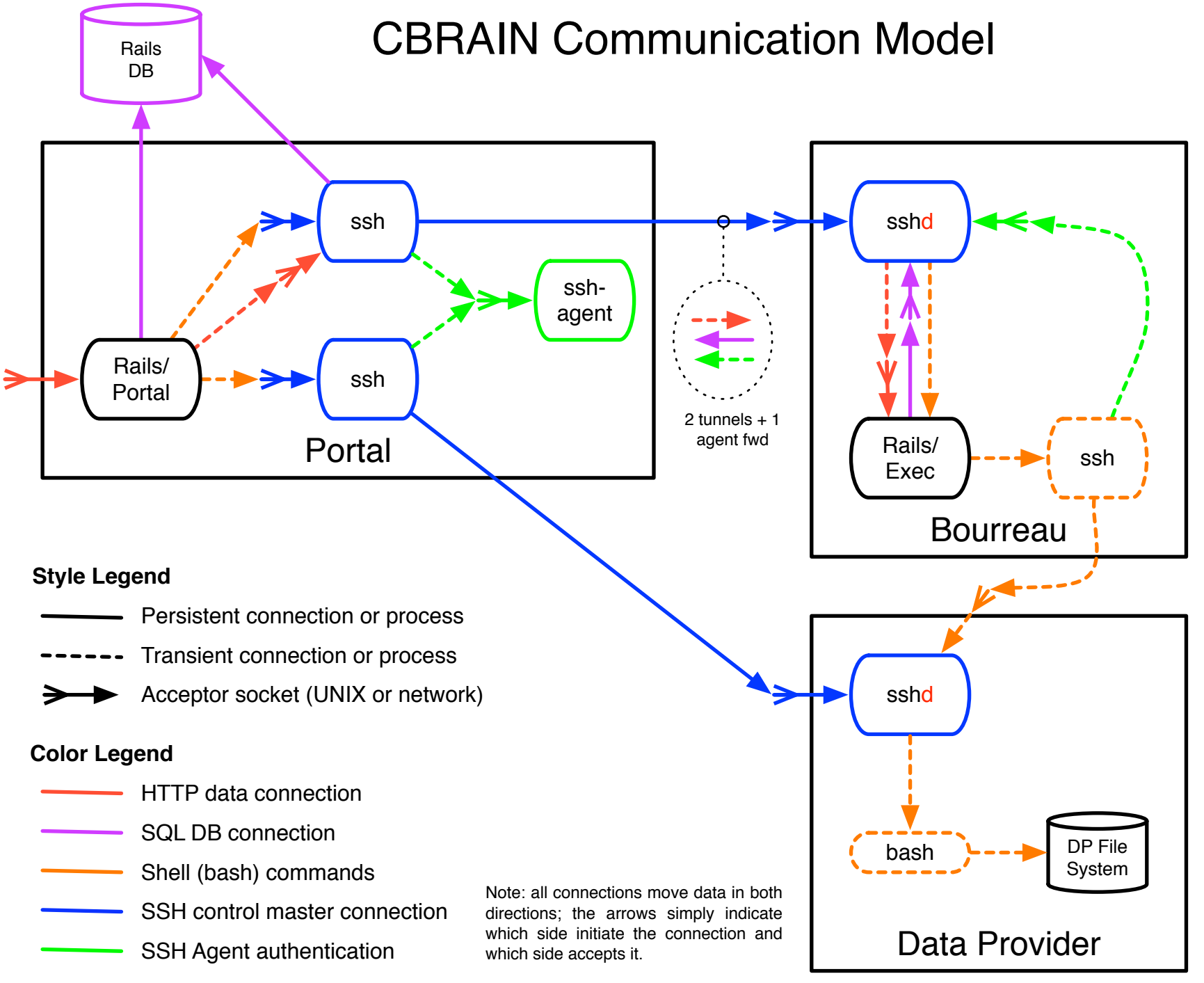


Before going to the office, connect there with the -R option and leave the shell open



At the office, use SSH to connect to localhost port 12345;
the SSH process at home will forward the connection to the sshd
of the home machine! We are tunneling SSH into SSH !

CBRAIN Communication Model



Style Legend

- Persistent connection or process
- Transient connection or process
- Acceptor socket (UNIX or network)

Color Legend

- HTTP data connection
- SQL DB connection
- Shell (bash) commands
- SSH control master connection
- SSH Agent authentication

Note: all connections move data in both directions; the arrows simply indicate which side initiate the connection and which side accepts it.

The END