

# Tunnels and VPN\*s

March 30, 2012

\*virtual private networks

## Administrative – submittal instructions

- answer the lab assignment's questions in written report form, as a text, pdf, or Word document file (no obscure formats please)
- email to [csci530l@usc.edu](mailto:csci530l@usc.edu)
- exact subject title must be "tunnelslab"
- deadline is start of your lab session the following week
- reports not accepted (zero for lab) if
  - late
  - you did not attend the lab (except DEN or prior arrangement)
  - email subject title deviates

## Administrative – Fall lab grader

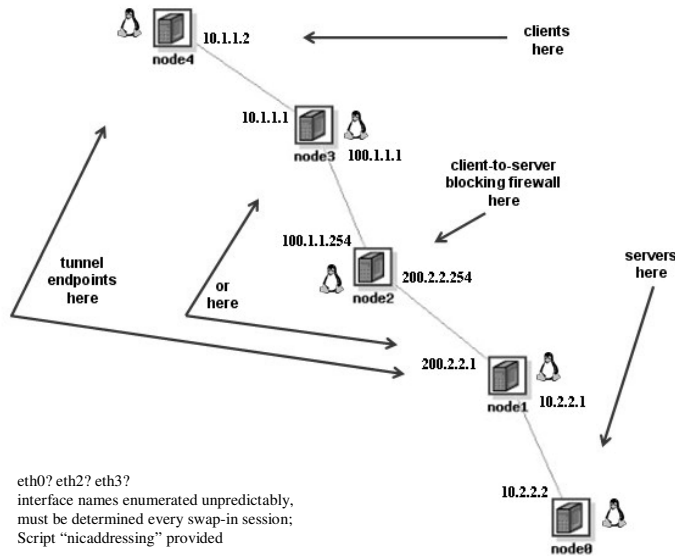
- will require 1 or 2 graders for CS530L
- seek students who took this course
- please send expression of interest
  - if potentially interested
  - if a continuing student
  - if you're good!
  - I'll put you on my "list"
- I'd like to identify graders to hire sooner rather than later (before end of current Spring semester)

<http://www-scf.usc.edu/~csci530l/grader-position.htm>

## What's a tunnel?

- encapsulation of data packets in data packets
- inner packets opaque to outer packets' network
- may or may not be encrypted– that's outside "tunnel" definition

# Lab experiment topology



# Tcpdump of ipip — packet becomes payload

```

root@node4:~# ping -s 18 -p 414243 -c1 node0
PATTERN: 0x414243
PING node0-link0 (10.2.2.2) 18(46) bytes of data.
26 bytes from node0-link0 (10.2.2.2): icmp_seq=1 ttl=62 time=2.46 ms

--- node0-link0 ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 2.469/2.469/2.469/0.000 ms
root@node4:~#

root@node3:~# tcpdump -s 1000 -xxnnti eth0 not stp and not arp and src host 10.1.1.2
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on eth0, link-type EN10MB (Ethernet), capture size 1000 bytes
IP 10.1.1.2 > 10.2.2.2: ICMP echo request, id 46860, seq 1, length 26
0x0000: 4500 002e 0000 4000 4001 23c9 0a01 0102  E....@.@.#....
0x0010: a02 0202 0800 de4b b70c 0001 6bfd fc4a  ....K...K...J
0x0020: a912 0500 4341 4243 4142 4341 4243  BC...CABCABC

IP header starts
IP 10.1.1.1 > 200.2.2.1: IP 10.1.1.2 > 10.2.2.2: ICMP echo request, id 46860, seq 1, length 26 (ipip-prot=4)
0x0000: 4500 0042 0400 4000 3f04 0cb3 6d01 0101  E..B..@.?...d..
0x0010: c802 0201 4500 002e 0000 4000 3f01 24c9  ...E.....@.?.$.
0x0020: 0a01 0102 0a02 0202 0800 de4b b70c 0001  ....K...K....
0x0030: 6bfd fc4a a912 0500 4341 4243 4142 4341  K..J....CABCABC
0x0040: 4243  BC
  
```

**node3's red incoming-packet & outgoing-payload are IP-identical\***

\*allowing TTL decrement and checksum recalc

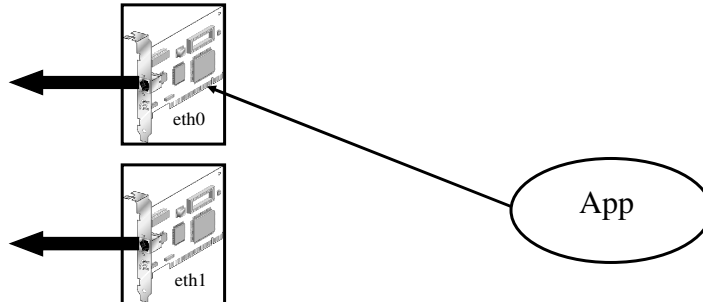
## Lab tunnels you will build

	true tunnel	non-tunnel channel
unencrypted	IP over IP	
encrypted	OpenVPN	ssh stunnel

## Tunnels spawn new interfaces

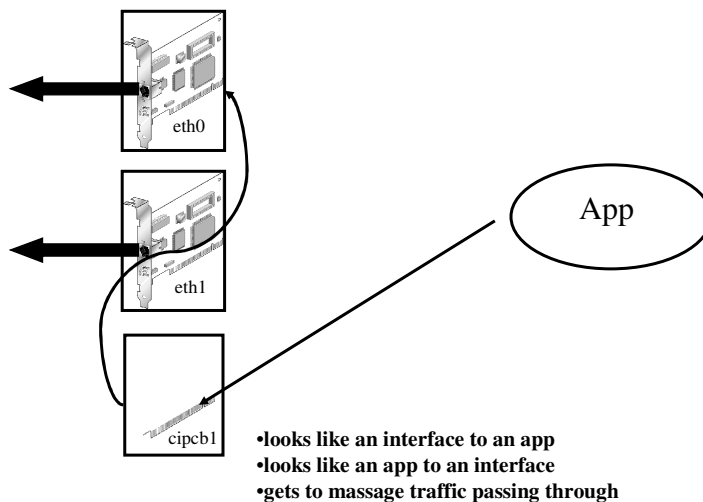
Physical (hardware)	Virtual (software)
<ul style="list-style-type: none"><li>●eth0</li><li>●eth1</li></ul>	<ul style="list-style-type: none"><li>●tunl0 (ip-ip)</li><li>●tap0 (OpenVPN)</li><li>●ipsec0 (IPSec)</li><li>●ppp0 (ppp-ssh)</li><li>●cipcb1 (CIPE)</li></ul>

## Using hardware interfaces



(Technical note: the choice of interface by an app is indirect. App source code expresses only an IP address. Downstream, IP software in network stack maps the address into an interface via the routing table.)

## Using software interfaces

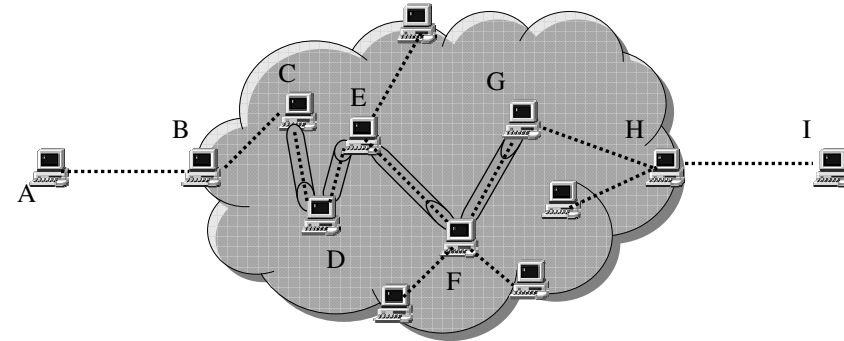


## What's a VPN

- a *virtual* net overlaid on an underlying net
- a *private* net retaining exclusivity through confidentiality
  - implemented by encryption
  - applying cryptographic methods you have studied

# TUNNELS

## Tunnel within a network

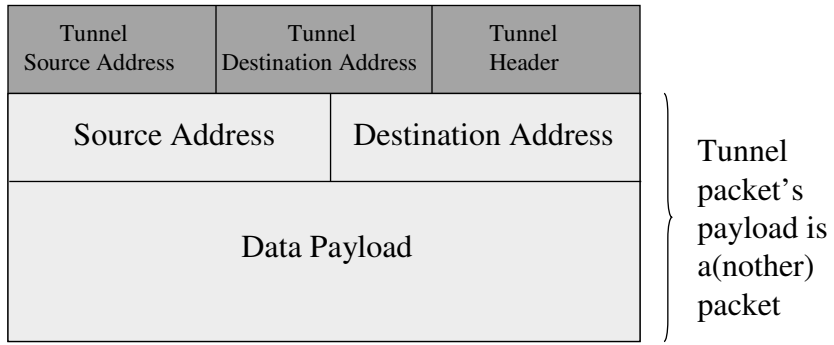


- ..... - Packet stream of protocol X
- - Packet stream of protocol Y
- (with dots) ——— - Packet stream: “X over Y” or “X tunneled in/through Y”

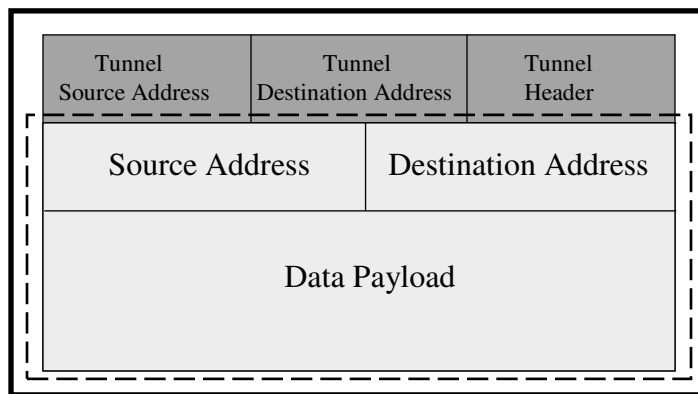
## A packet to be tunneled

Source Address	Destination Address
Data Payload	

# Tunnel packet

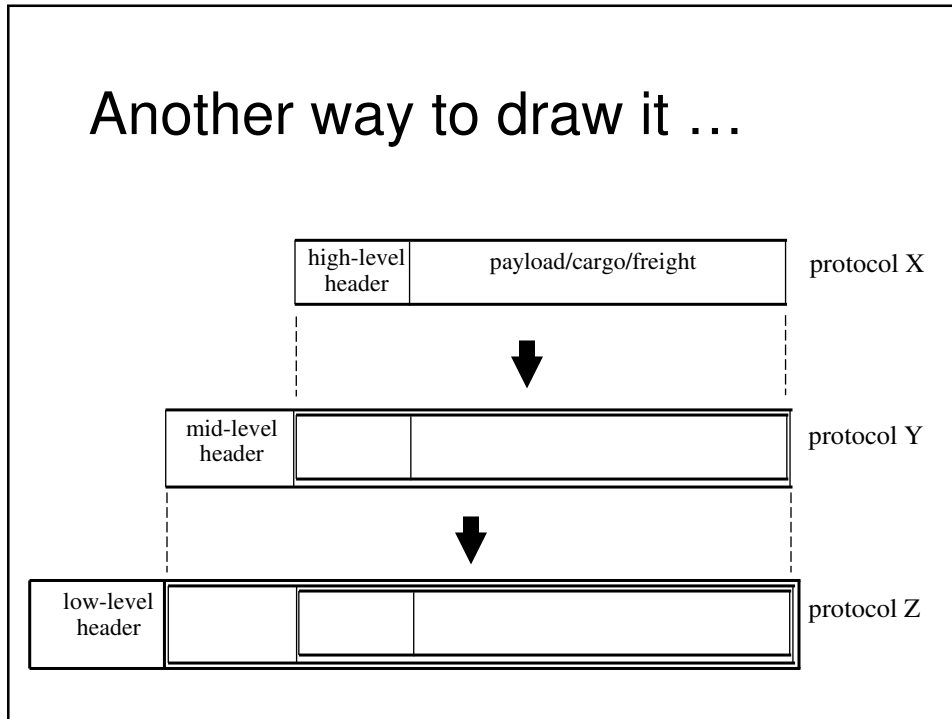


# X over Y tunneling



Packet of protocol X  
Packet of protocol Y

## Another way to draw it ...



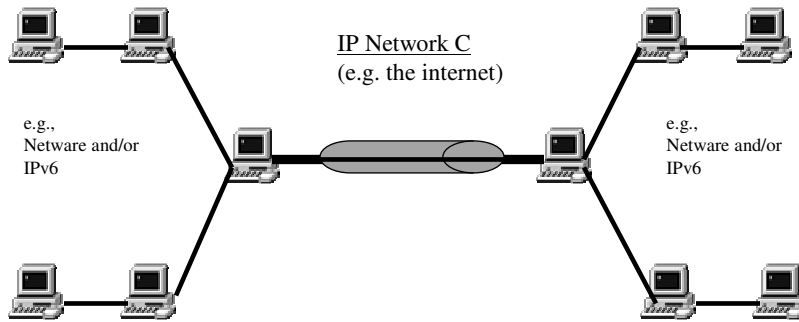
## Uses of tunneling

- carry payloads over domains where otherwise illegal
  - carry *protocols* that are illegal
  - carry *addresses* that are illegal
- apply common services to multiple traffic flows

# 'Illegal' protocols over IP

IPX and/or IPv6 Network A

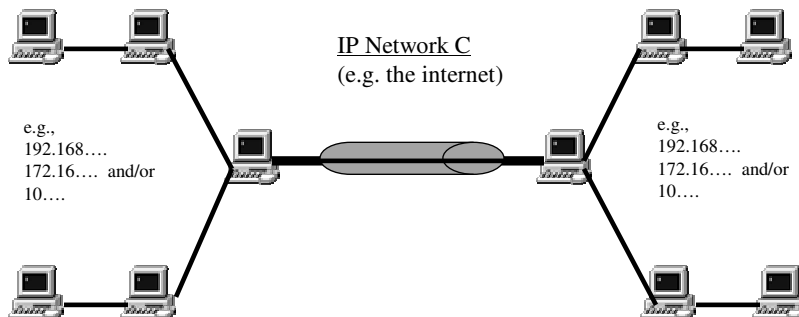
IPX and/or IPv6 Network B



# 'Illegal' addresses over IP

Private IP Network A

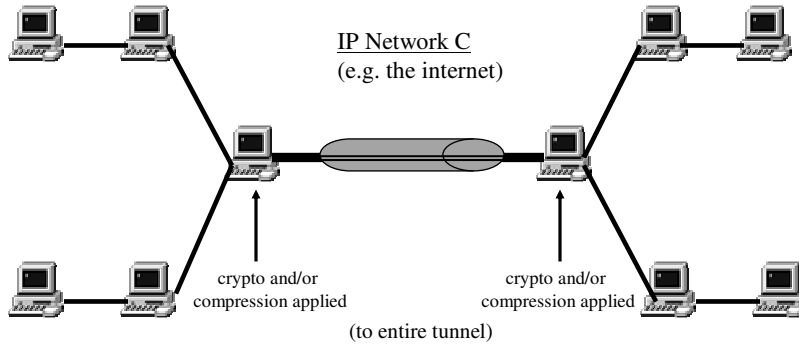
Private IP Network B



# Applying common services

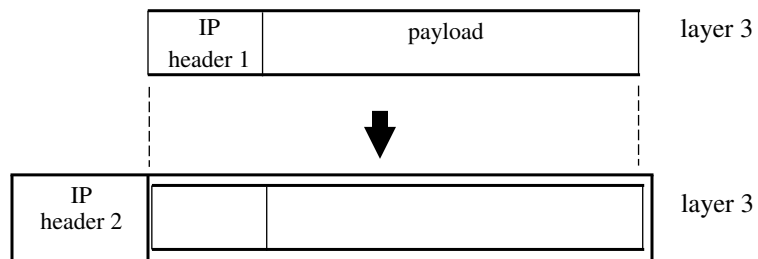
IPX Network A

IPX Network B

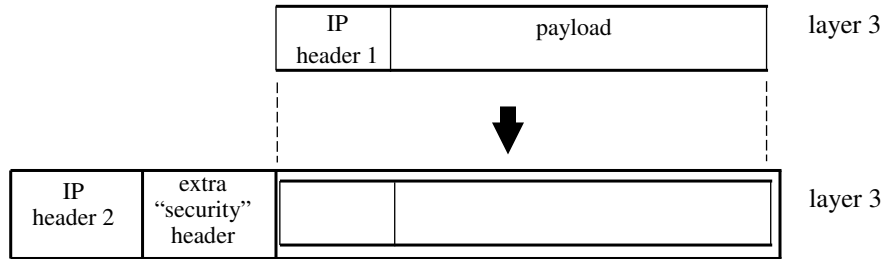


by e.g. ssh or stunnel (ssl) or OpenVPN or IPSec

# Layer 3 tunneling example: IP over IP



## Layer 3 tunneling example: IPsec

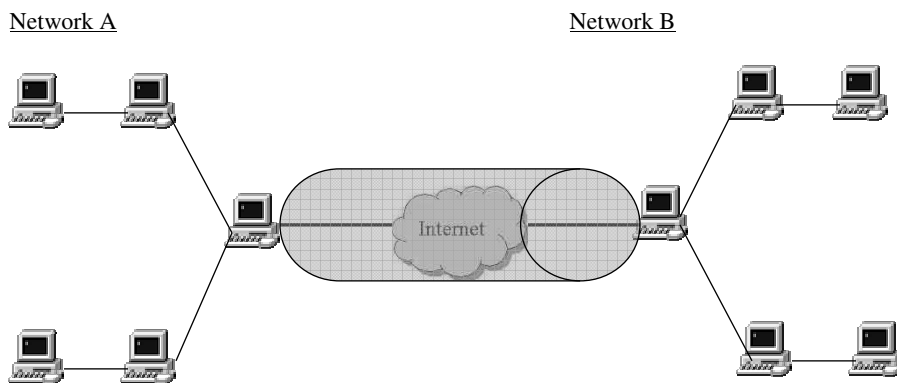


# VPNS

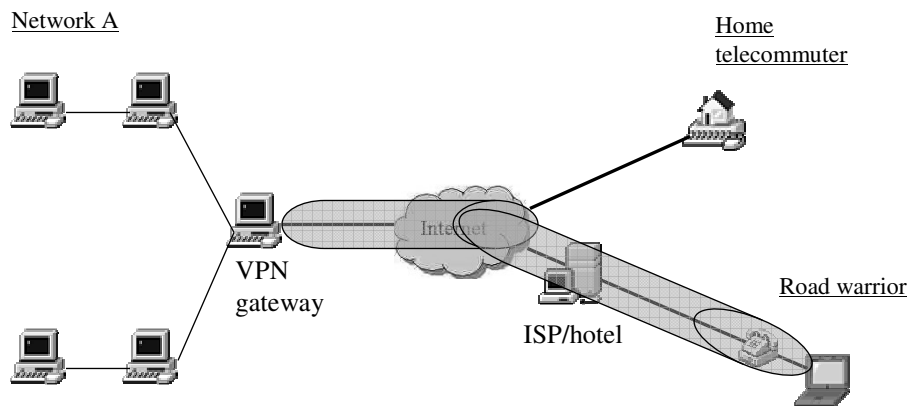
# Placement-based Architectures

- Site-to-site Intranet VPN
- Remote access VPN

## Site-to-site VPN via internet



## Remote access VPN via internet connection



lab exercise product 1  
**IPIP**

## What is it?

- Conveys an IP packet between machines
  - ... not as a packet
  - ... but as cargo in another packet
- Destination shucks carrier packet, releases cargo as packet into local networking machinery
- “Tunnel” since one packet “passes through” another
- Implemented in linux by module ipip.o

## S.S. Badger

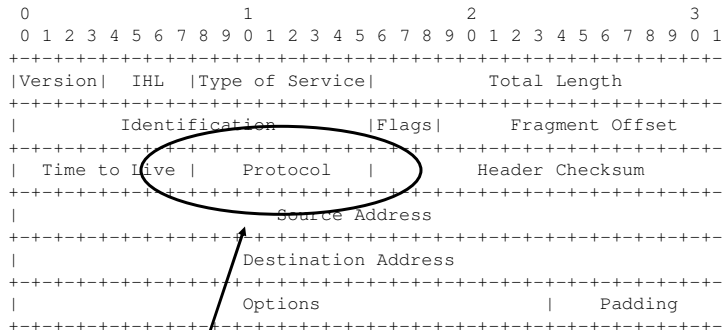


- Conveys a car between states
  - ... not as a car/motor-vehicle
  - ... but as cargo in a boat
- Destination throws away boat, releases car as a motor vehicle onto local roadways
- “Tunnel” since one vehicle “passes through” another
- Implemented by Lake Michigan Carferry Service



# IP itself is an IP subprotocol

## IP Header Format

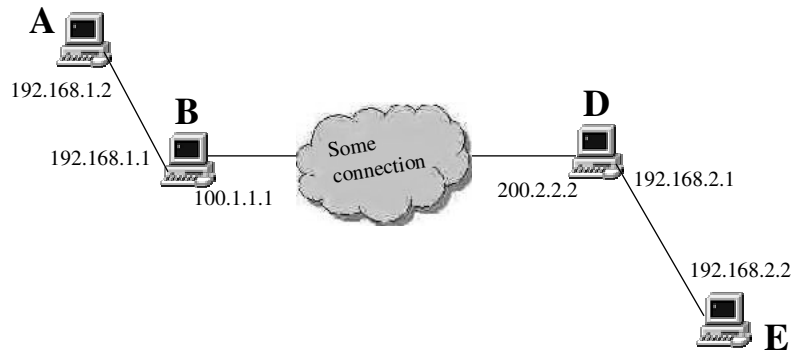


4 for IP  
 (6 for TCP  
 17 for UDP  
 50 for ESP, etc)

# Sample LAN

Local Network – 192.168.1.0

Remote Network – 192.168.2.0



Workstations – A and E

Gateways – B and D

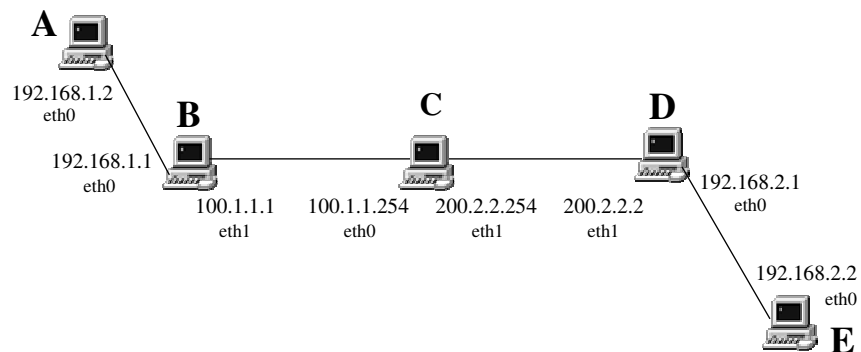
## “Some connection”

- Could be the internet
- Could be a single intermediate machine
- Equivalent, for the 2 gateways

## Sample LAN

Local Network – 192.168.1.0

Remote Network – 192.168.2.0

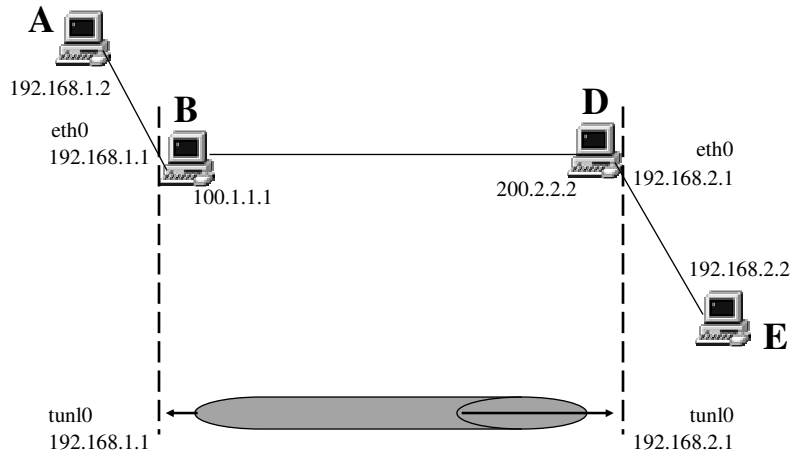


Workstations – A and E  
Gateways – B and D  
Internet surrogate – C (B's ISP; D's ISP)

# Wanted: a 2<sup>nd</sup> bridge to cross

Local Network – 192.168.1.0

Remote Network – 192.168.2.0



lab exercise product 2  
ssh

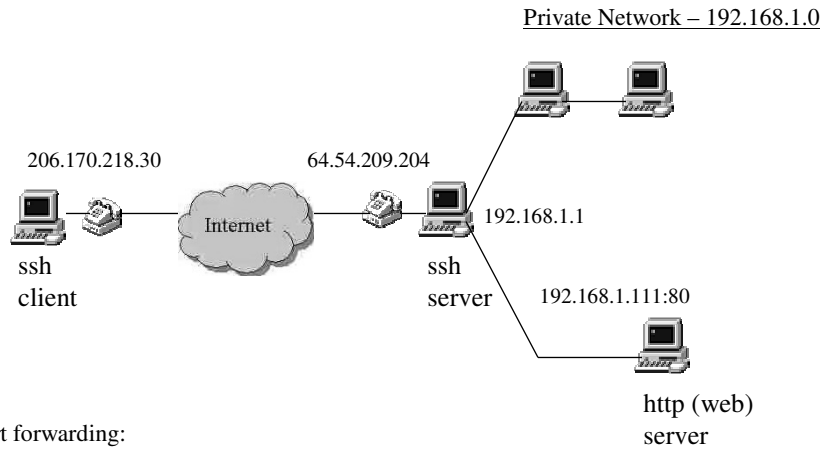
## A client-server pair of programs

- ssh - client
  - /usr/bin/ssh
- sshd - server
  - /usr/sbin/sshd
  - assigned port number 22

## ssh – why secure?

- all session/command traffic passes through ssh/sshd (sshd runs on port 22)
- encrypted going out/decrypted coming in
- for duration of session/command
- uses RSA (public-key) authentication
- then strong-key symmetrical encryption

# ssh feature: port forwarding



ssh port forwarding:

correspond some port on the client (e.g., 3000) to  
some port (e.g., 80) on a machine reachable thru the server....

Example: `http://127.0.0.1:3000` in client's browser gets served from 192.168.1.111

# ssh syntax

## Normal log in

```
ssh remote-user@remote-IP
```

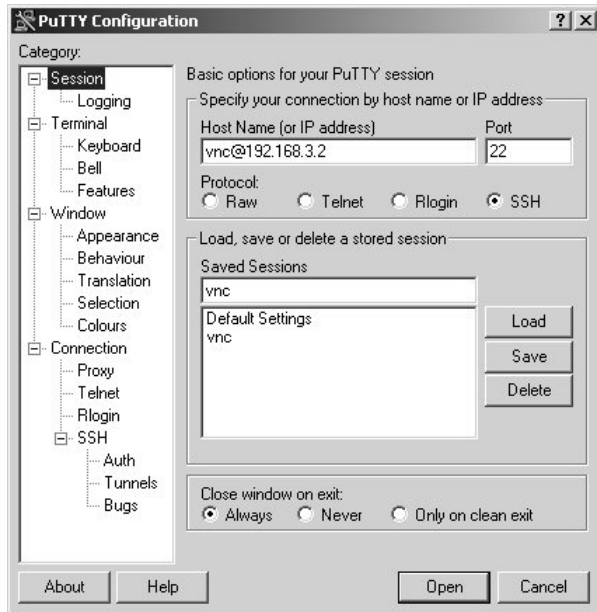
e.g., `ssh root@64.54.209.204`

## Adding a tunnel

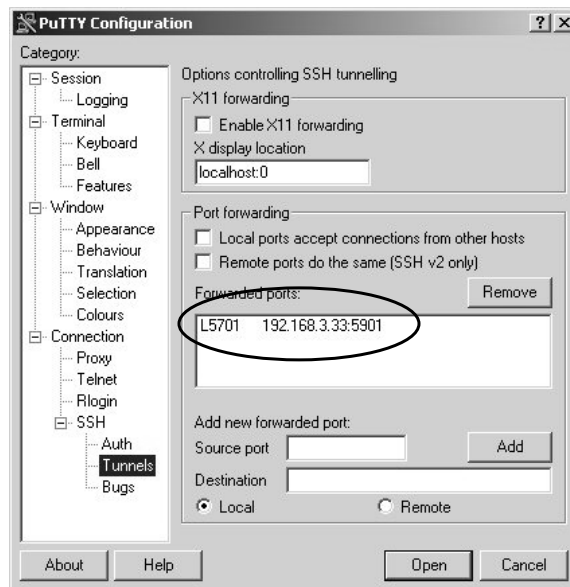
```
ssh -L local-port:target-IP:remote-port remote-user@remote-IP
```

e.g., `ssh -L 3000:192.168.1.111:80 root@64.54.209.204`

# puTTY



# puTTY



## lab exercise product 3

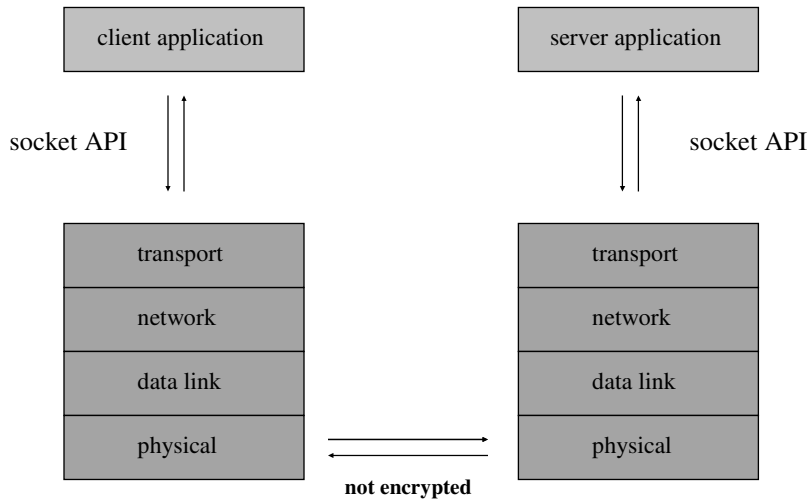
# stunnel

## Encrypt the talk between clients and servers who don't

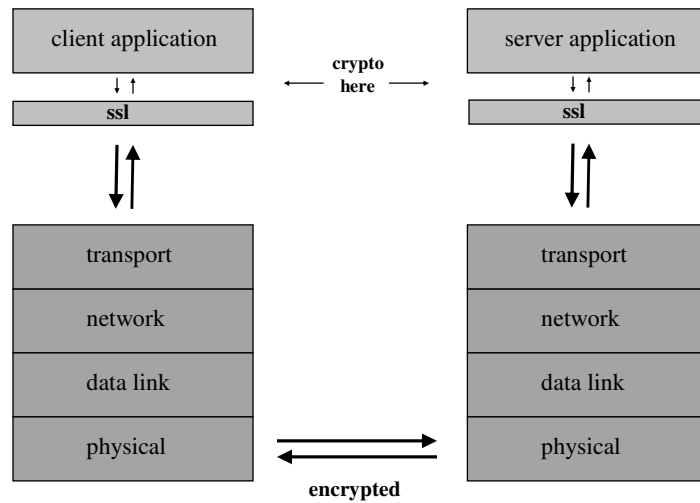
“The stunnel program is designed to work as SSL encryption wrapper between remote clients and local (inetd-startable) or remote servers. The concept is that having non-SSL aware daemons running on your system you can easily set them up to communicate with clients over secure SSL channels.

[stunnel man page](#)

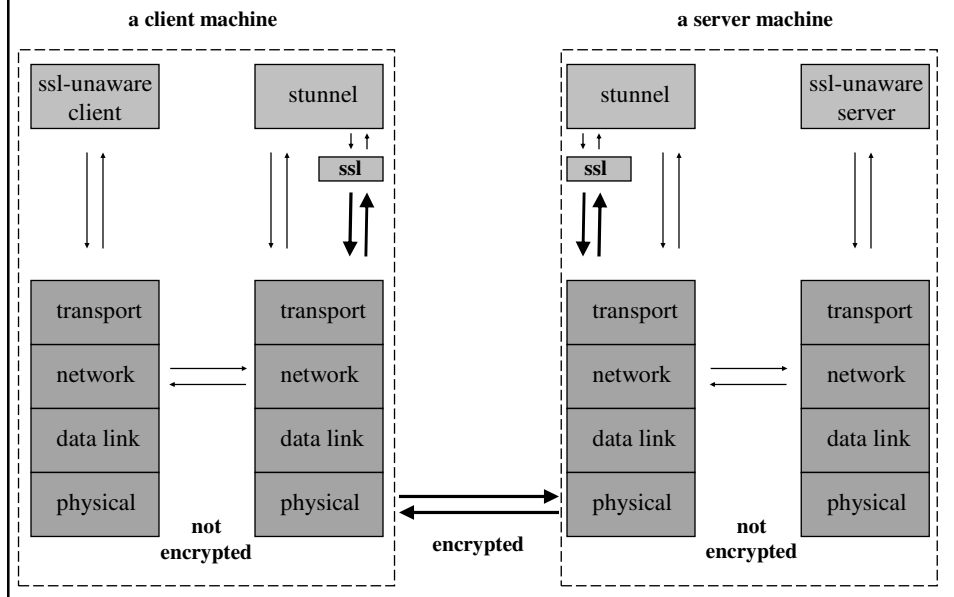
# Ordinary ssl-unaware applications



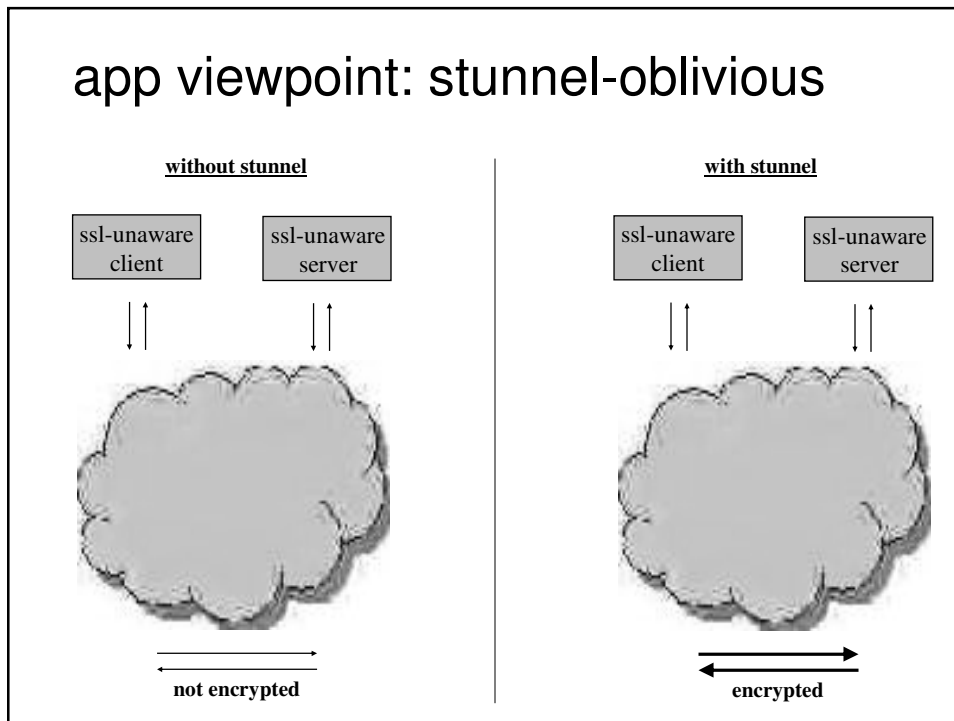
# SSL-aware applications



# stunnel – 3 TCP conversations



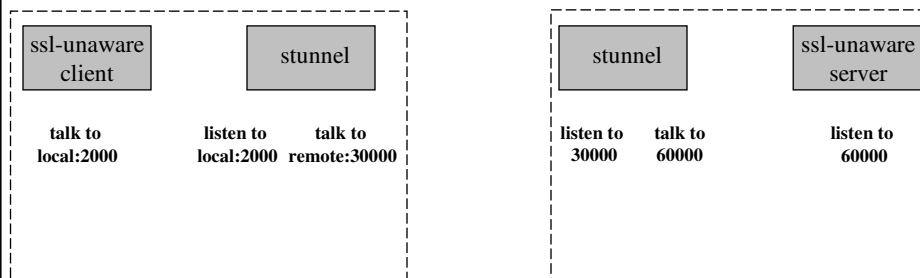
# app viewpoint: stunnel-oblivious



## Ports: non-stunnel scenario



## stunnel – 3 TCP conversations



## Vanilla config files

```
# stunnel client  
  
client=yes  
  
[stunnel service name]  
  accept = 127.0.0.1:2000  
  connect = 192.168.3.12:30000
```

```
# stunnel server at 192.168.3.12  
  
cert = /etc/stunnel/stunnel.pem  
  
[example service name]  
  accept = 30000  
  connect = 60000
```

## stunnel server needs certificate

- create it with

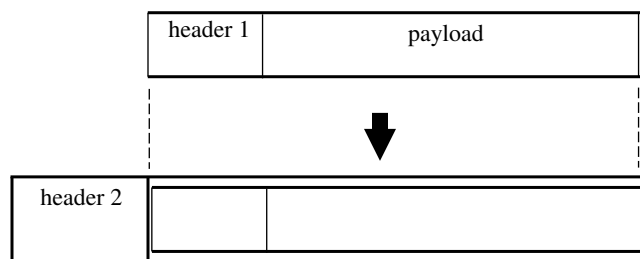
```
cd /etc/stunnel  
openssl req -new -x509 -days 3650 -nodes -out stunnel.pem -keyout stunnel.pem
```

- reference it in stunnel server's config file
- <http://www.stunnel.org/faq/certs.html#ToC5>

## stunnel's not really a tunnel

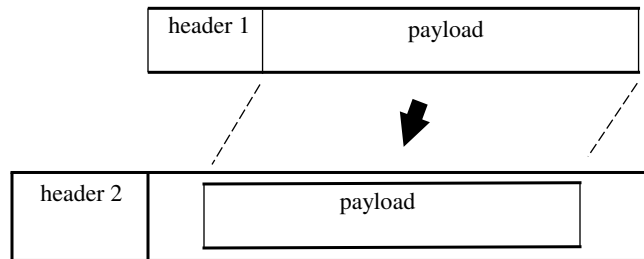
- stunnel is a conversation endpoint
- and a (different) conversation startpoint
- arriving packets are stripped of header at endpoint
- their content repackaged, new header, at startpoint
- headers do not nest/accumulate as in tunnels

## True tunneling



headers accumulate

## Payload forward/relay/proxy



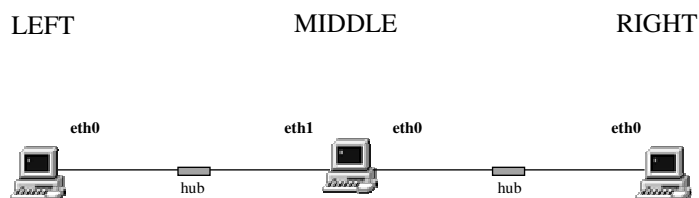
headers replace each other

lab exercise product 4  
**OpenVPN**

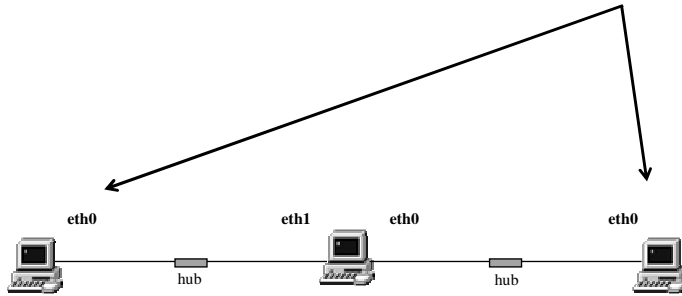
## Lab's OpenVPN tunnel scenarios

- a routed tunnel, unencrypted
- a routed tunnel, encrypted using static, preshared secret keys
- a bridged tunnel, encrypted using SSL

Given this setup...

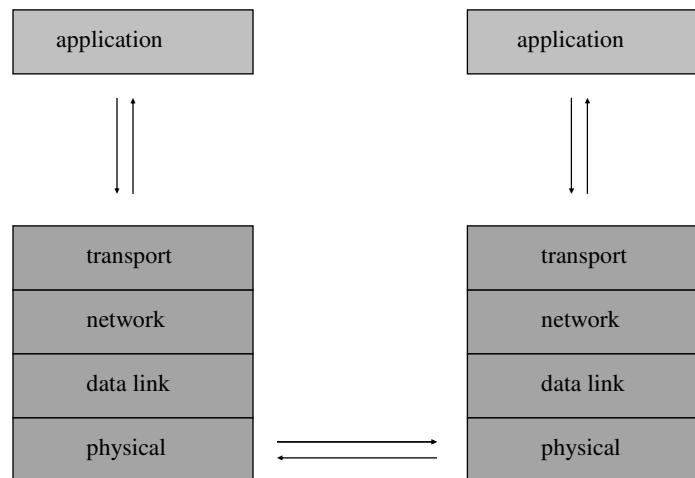


... 2 configs could make 'em ping

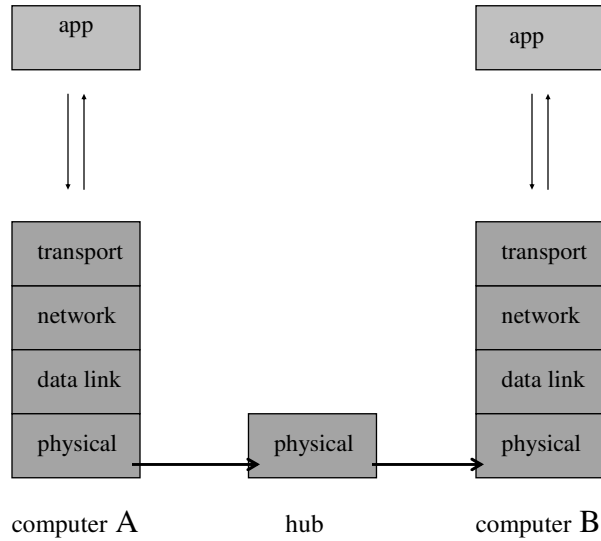


1. **Routing**  
make 2 LANs out of it (2 broadcast domains)  
end-to-end connection achieved by *routing* the *IP packets*
2. **Bridging**  
make 1 consolidated LAN out of it (single broadcast domain)  
end-to-end by *bridging* the *ethernet frames*

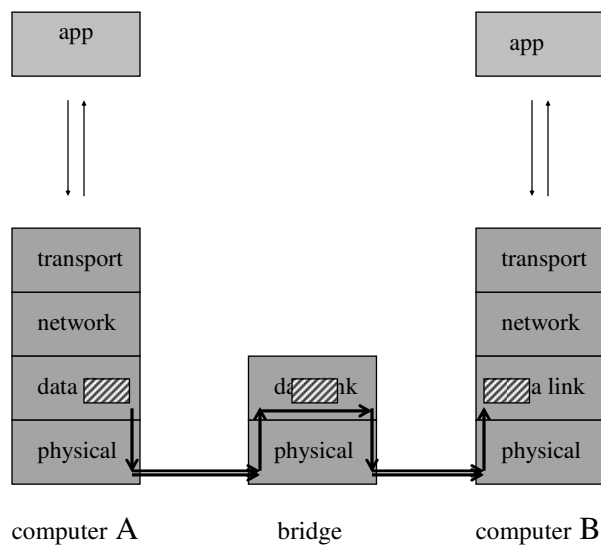
## Info's usual trans-layer itinerary



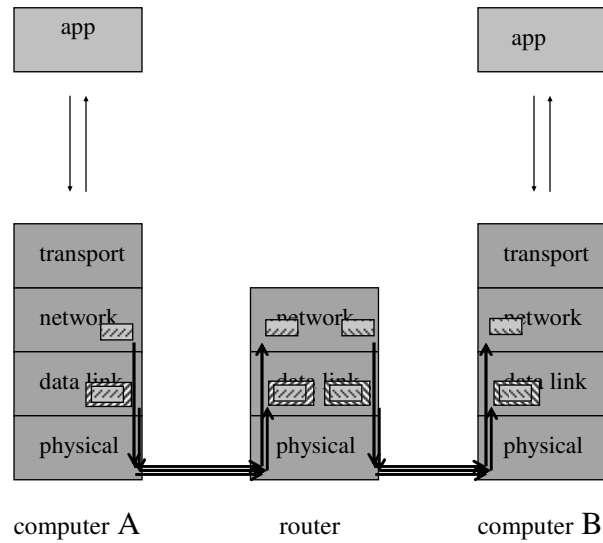
## Signals via hub ("layer 1 device")



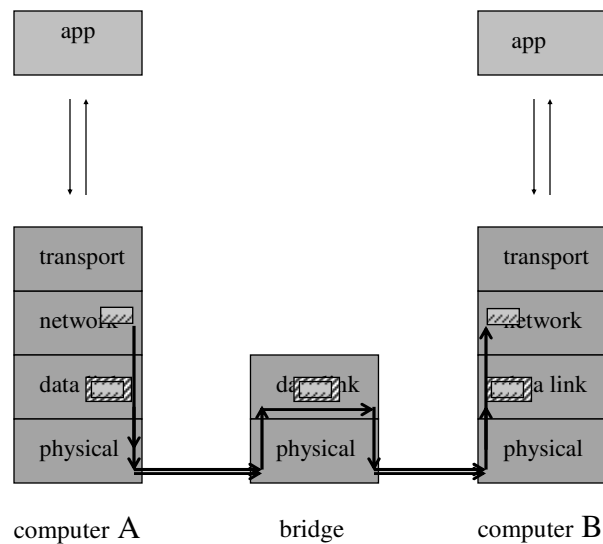
## Frames via bridge ("layer 2 device")



## Packets via router ("layer 3 device")



## Note, bridge scenario: frame's contained packet untouched



## OpenVPN features

- unique certificate/key-pair for every client
- choice of ciphers
- bridged case
  - extends LAN-local IP to remote joiner
  - allows broadcast-dependent apps (printer sharing)
  - makes remoteness transparent
    - routing does it *mostly*
    - bridging does it *entirely*

## Info – IP over IP

- IP in IP Tunneling
  - <http://www.rfc-editor.org/rfc/rfc1853.txt>
- IP Encapsulation within IP
  - <http://www.rfc-editor.org/rfc/rfc2003.txt>

## Info - ssh

- “Getting Started with ssh”  
<http://kimmo.suominen.com/ssh/>
- ssh resource page  
<http://www.csri.utoronto.ca/~djast/ssh.html>
- free clients for Windows
  - puTTY  
<http://www.chiark.greenend.org.uk/~sgtatham/putty/>
  - OpenSSH for Windows  
<http://sshtwindows.sourceforge.net/>

## Info - stunnel

- <http://www.stunnel.org/>
- <http://freshmeat.net/articles/view/1781>

## Info OpenVPN

- <http://openvpn.net/>
- <http://en.wikipedia.org/wiki/OpenVPN>
- client for Windows
  - <http://openvpn.se/>