# ICS sharing between switch and management computer

Wednesday, November 9, 2016 9:55 AM

EBTABLES:https://wiki.debian.org/BridgeNetworkConnectionsProxyArp

ebtables -t nat -A POSTROUTING -o wlan0 -j snat --to-src $MAC_OF_BRIDGE --snat-arp --snat-target ACCEPT
ebtables -t nat -A PREROUTING -p IPv4 -i wlan0 --ip-dst $IP -j dnat --to-dst $MAC --dnat-target ACCEPT
ebtables -t nat -A PREROUTING -p ARP -i wlan0 --arp-ip-dst $IP -j dnat --to-dst $MAC --dnat-target ACCEPT
EBTABLES_ATOMIC_FILE=/root/ebtables-atomic ebtables -t nat --atomic-save
EBTABLES_ATOMIC_FILE=/root/ebtables-atomic ebtables -t nat --atomic-commit

sudo ebtables -t nat -A POSTROUTING -o wlan0 -j snat --to-src 0c:60:76:40:9b:40 --snat-arp --snat-target ACCEPT
sudo ebtables -t nat -A PREROUTING -p IPv4 -i wlan0 --ip-dst 192.0.2.6 -j dnat --to-dst 54:ee:75:98:a0:29 --dnat-target ACCEPT
sudo ebtables -t nat -A PREROUTING -p ARP -i wlan0 --arp-ip-dst 192.0.2.6 -j dnat --to-dst 54:ee:75:98:a0:29 --dnat-target ACCEPT
sudo EBTABLES_ATOMIC_FILE=/root/ebtables-atomic ebtables -t nat --atomic-save
sudo EBTABLES_ATOMIC_FILE=/root/ebtables-atomic ebtables -t nat --atomic-commit

## Below one finally worked ☺ I just had to enable the IPV4 forwarding finally!!! ☺

IP gateway configuration:(https://help.ubuntu.com/community/Internet/ConnectionSharing#Advanced_Gateway_Configuration)
==========

sudo iptables -A FORWARD -o wlan0 -i eth0 -s 10.42.0.1/24 -m conntrack --ctstate NEW -j ACCEPT
sudo iptables -A FORWARD -m conntrack --ctstate ESTABLISHED,RELATED -j ACCEPT
sudo iptables -t nat -F POSTROUTING
sudo iptables -t nat -A POSTROUTING -o wlan0 -j MASQUERADE

## Ubuntu Internet Gateway Method (iptables)

You will need two network cards in the gateway computer, or a PPP interface and a network card. One network card (or PPP interface) connects to the Internet. We will call this card eth0. The other card connects to your internal network. We will call this eth1. It is also possible to do ICS with a single network card. In this case, use eth0 for the Internet and eth0:0 for the internal network.

1. Internet <<==>> eth0 <> Ubuntu gateway <> eth1 <<==>> Client PC
2. Internet <<==>> ppp0 <> Ubuntu gateway <> eth1 <<==>> Client PC
3. Internet <<==>> eth0 <> Ubuntu gateway <> eth0:0 <<==>> Client PC

### Gateway set up

The following example will focus on the most common gateway setup: an Ubuntu computer with two wired network adapters (eth0 and eth1) hosting ICS to a static internal network configured for the 192.168.0.x subnet.

For this example, eth0 is used to represent the network card connected to the Internet, and eth1 represents the network card connected to a client PC. You can replace eth0 and eth1 as needed for your situation. Also, any private IP subnet can be used for the internal network IP addresses.

In summary:

eth0 = the network adapter with internet (external or WAN).
eth1 = the network adapter to which a second computer is attached (internal or LAN).
192.168.0.x = IP subnet for eth1

Your setup may be different. If so, make sure to change them accordingly in the following commands.

Configure internal network card
Configure your internal network card (eth1) for static IP like so:

```
sudo ip addr add 192.168.0.1/24 dev eth1
```

The external and internal network cards cannot be on the same subnet.

Configure NAT
Configure iptables for NAT translation so that packets can be correctly routed through the Ubuntu gateway.

```
sudo iptables -A FORWARD -o eth0 -i eth1 -s 192.168.0.0/24 -m conntrack --ctstate NEW -j ACCEPT
sudo iptables -A FORWARD -m conntrack --ctstate ESTABLISHED,RELATED -j ACCEPT
sudo iptables -t nat -F POSTROUTING
sudo iptables -t nat -A POSTROUTING -o eth0 -j MASQUERADE
```

Screen clipping taken: 11/9/2016 10:08 AM

The first rule allows forwarded packets (initial ones). The second rule allows forwarding of established connection packets (and those related to ones that started). The third rule does the NAT.

IPtables settings need to be set-up at each boot (they are not saved automatically), with the following commands:

1. Save the iptables:

```
sudo iptables-save | sudo tee /etc/iptables.sav
```

1. Edit `/etc/rc.local` and add the following lines before the "exit 0" line:

```
iptables-restore < /etc/iptables.sav
```

Enable routing

1. Configure the gateway for routing between two interfaces by enabling IP forwarding:

```
sudo sh -c "echo 1 > /proc/sys/net/ipv4/ip_forward"
```

1. Edit `/etc/sysctl.conf`, and (up to **10.04**) add these lines:

```
net.ipv4.conf.default.forwarding=1
net.ipv4.conf.all.forwarding=1
```

**The `/etc/sysctl.conf` edit is required because of the following bug in Hardy and later releases:** Launchpad Bug Report

1. From **10.10** onwards, it suffices to edit `/etc/sysctl.conf` and uncomment:

```
#net.ipv4.ip_forward=1
```

... so that it reads:

```
net.ipv4.ip_forward=1
```

## Client set up

Any OS can connect to the Internet as an ICS client as long as networking has been configured correctly. The following example will focus on how to set up an Ubuntu ICS client. For this example, it is assumed that the client is connected to an Ubuntu gateway, which has been configured to share ICS on the 192.168.0.x subnet according to the gateway set up outlined above.

For this example, eth0 is the network card on the client which is connected (by crossover cable) to eth1 on the Ubuntu gateway. You can replace eth0 as needed for your situation. Also, any private IP subnet can be used for the internal network IP address, as long as it matches the subnet on the gateway.

Disable networking

```
sudo /etc/init.d/networking stop
```

Give the client a static IP address

```
sudo ip addr add 192.168.0.100/24 dev eth0
```

This IP address can be anything within the gateway's private IP range.

Configure routing

```
sudo ip route add default via 192.168.0.1
```

This address should match the IP address on the gateway's internal network card (eth1 in the above example).

Screen clipping taken: 11/9/2016 10:08 AM

Configure DNS servers

Unless your ICS gateway can also perform DNS, you must manually configure the client with your ISP DNS servers. If you do not know your ISP's DNS servers, you can use OpenDNS servers instead.

1. Backup your current `/etc/resolve.conf` file:

```
sudo cp /etc/resolv.conf /etc/resolv.conf.backup
```

1. Open `/etc/dhcp3/dhclient.conf` with your favorite text editor:

```
sudo nano /etc/dhcp3/dhclient.conf
```

1. Search for the line that starts "prepend domain-name-servers", and change it to look like this:

```
prepend domain-name-servers 208.67.222.222,208.67.220.220;
```

208.67.222.222 and 208.67.220.220 are OpenDNS DNS servers. If you wish to use your ISP's DNS servers, use them here instead of the OpenDNS servers.

Restart networking

```
sudo /etc/init.d/networking restart
```

Once this is finished, your client will now have access to the Internet via ICS. Please direct any questions/comments to the Internet Connection Sharing Documentation thread.

---

A beginner's working example of a Ubuntu Desktop with 2 NIC cards, sharing Internet connection:
http://ubuntuforums.org/showthread.php?p=3713684

Screen clipping taken: 11/9/2016 10:09 AM

**Use this IP address on the switch:**
Ethernet adapter Ethernet:

Connection-specific DNS Suffix . :
Link-local IPv6 Address . . . . . : fe80::51d7:4965:31c1:8d92%23
IPv4 Address. . . . . : 10.42.0.128
Subnet Mask . . . . . : 255.255.255.0
Default Gateway . . . . . : 0.0.0.0
10.42.0.1

On the switch I had to edit /blablablab/dhcp/dhcp.conf to prepent-dns-headers as per the link above
And also had to run dhclient as /sbin/dhclient.

After that Internet was enabled.

Below is the link for windows:
http://www.windowscentral.com/how-set-and-manage-network-bridge-connection-windows-10

VLAN Tagging on the switch:

> I have actually seen this on a cheapo-switch. Someone had connected a switch between a trunk port which had a couple vlans. The frames were forwarded with the vlan tagging intact. The other ports on that switch where able to use the un-tagged vlan.
>
> A switch only needs the source/destination mac to decide which ports to forward the frames to, so this isn't too surprising, a tagged frame still has the source and destination macs, in the same location in the frame header.
>
> Keep in mind that Ethernet actually supports many different frame-types on the same wire. It was designed to be pretty flexible about what it can do.