Networking Basics

Artem Gonchar, 2017

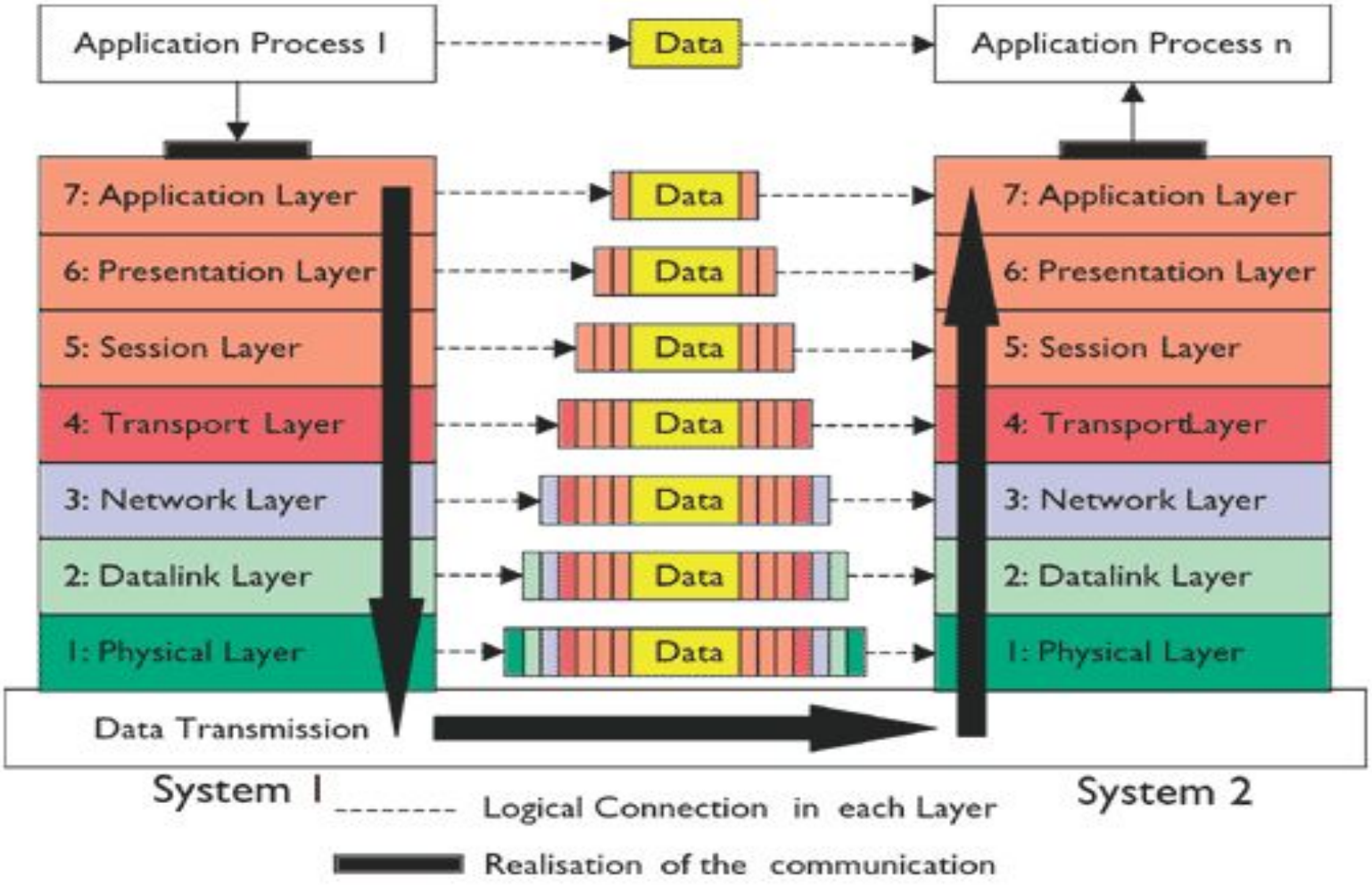
Agenda

- UDP vs TCP (usage in PortaSwitch)
- Routing (static, dynamic, gateways)
- Bonding (overview, configuration in RHEL6 & RHEL7, recommendations, bond in net. manager)
  - Network manager

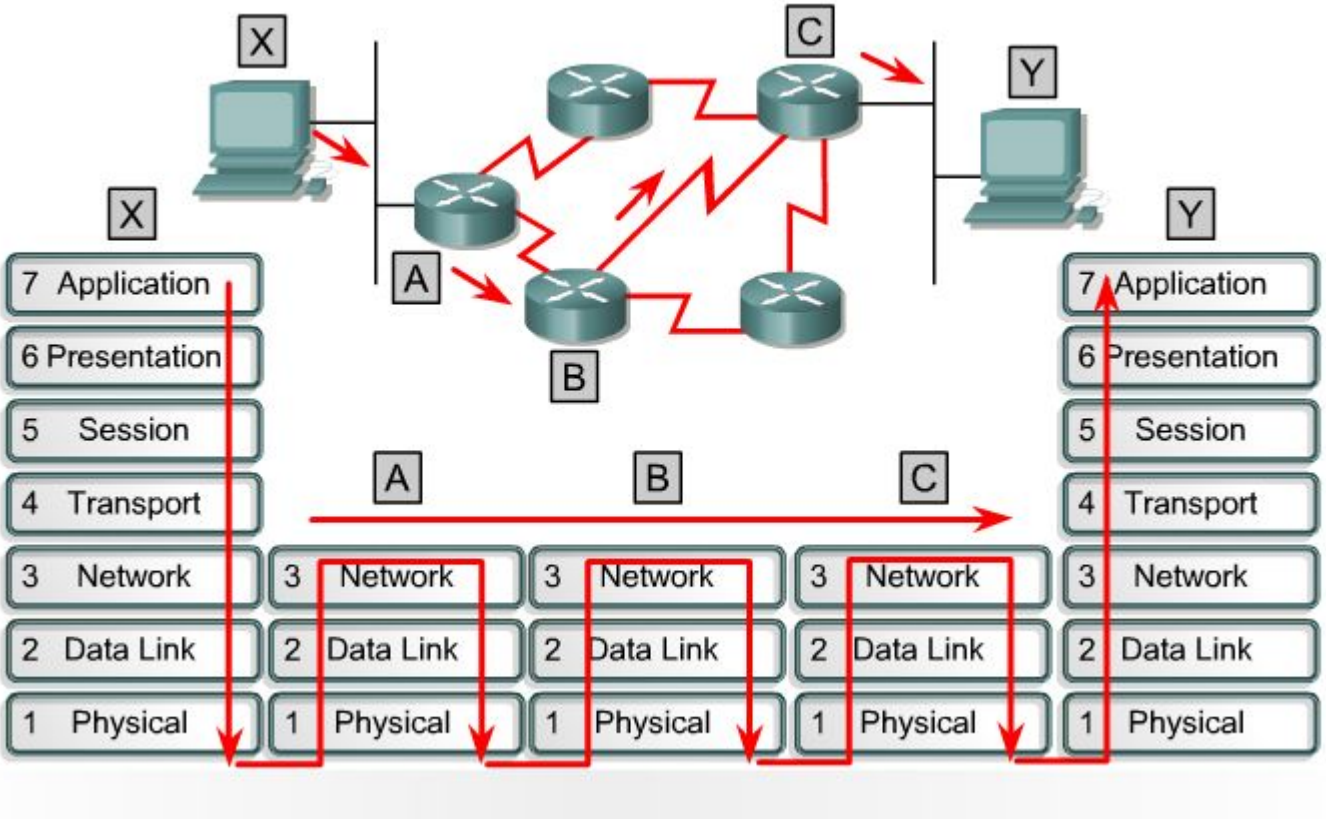
UDP vs TCP (usage in PortaSwitch)

7) Application	Data	Network process to application
6) Presentation		Data representation, encryption and decryption, convert machine dependent data to machine independent data
5) Session		Interhost communication, managing sessions between applications
4) Transport	Segments	Reliable delivery of packets between points on a network. Proto: UDP, TCP
3) Network	Packet/Datagram	Addressing, routing and (not necessarily reliable) delivery of datagrams between points on a network. Protocols: IP/IPv4/IPv6, IPX, IPsec, RIP Devices: Router
2) Data link	Bit/frame	Transfers data between network entities and provides means to detect and correct errors that may occur in the physical layer. Protocols are Ethernet, the Point-to-Point Protocol (PPP) . Devices: switch, bridge
1) Physical	Bit	Performs character encoding, transmission, reception and decoding. Protocols: Ethernet - 100BASE-TX, 100BASE-FX, 100BASE-T, 1000BASE-T, 1000BASE-SX; DSL, Wi-Fi. Devices: hub, repeater etc.

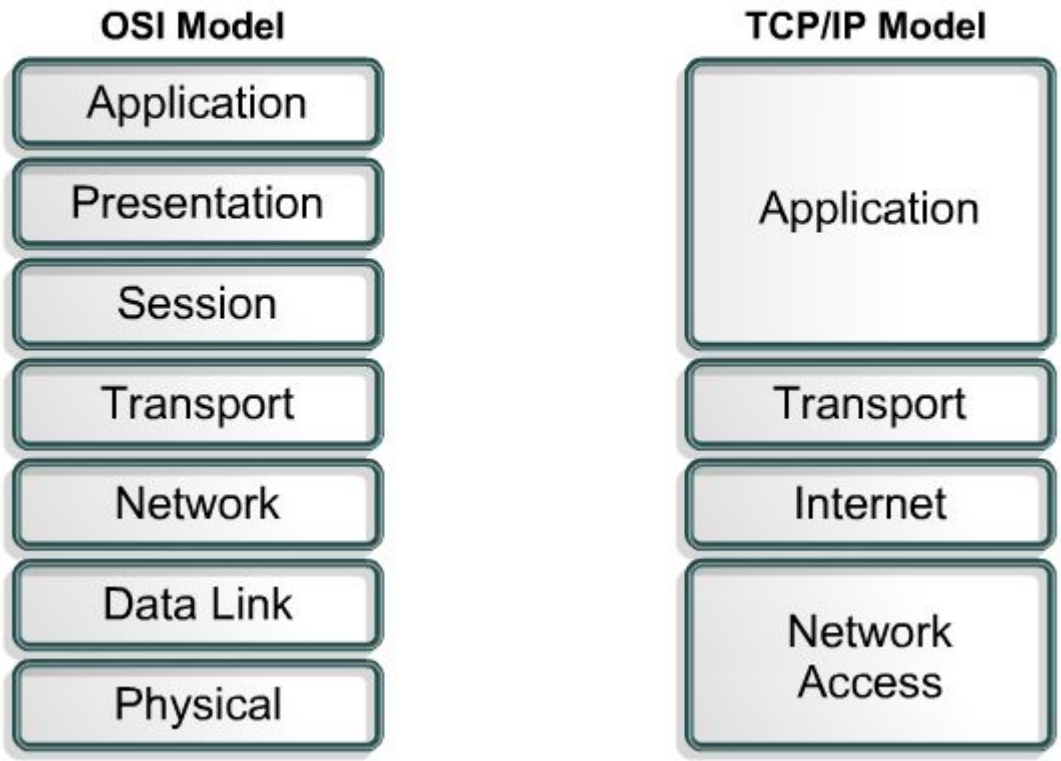
OSI Encapsulation



OSI (Open Source Interconnection) 7 Layer Model



OSI Model vs TCP/IP Model



TCP (Transport Control Protocol)

- is a connection-oriented transport layer protocol
- provides reliable full-duplex data transmission

TCP Segment Format

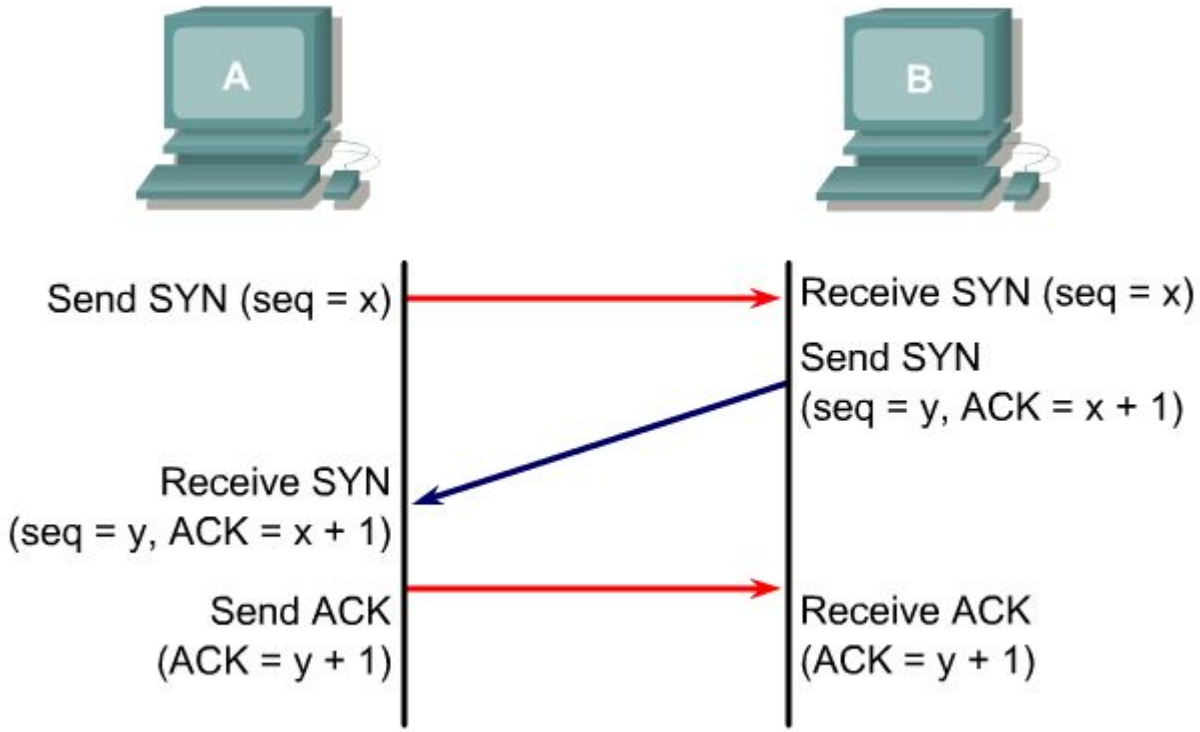
TCP Header

Offsets	Octet	0								1								2								3							
Octet	Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
0	0	Source port																Destination port															
4	32	Sequence number																															
8	64	Acknowledgment number (if ACK set)																															
12	96	Data offset				Reserved 0 0 0				NS	CWR	ECE	URG	ACK	PSH	RST	SYN	FIN	Window Size														
16	128	Checksum																Urgent pointer (if URG set)															
20 ...	160 ...	Options (if *data offset* > 5. Padded at the end with "0" bytes if necessary.) ...																															

TCP Segment Format

- **Source port** – Number of the port that sends data
- **Destination port** – Number of the port that receives data
- **Sequence number** – Number used to ensure the data arrives in the correct order
- **Acknowledgment number** – Next expected TCP octet
- **HLEN** – Number of 32-bit words in the header
- **Reserved** – Set to zero
- **Code bits** – Control functions, such as setup and termination of a session
- **Window** – Number of octets that the sender will accept
- **Checksum** – Calculated checksum of the header and data fields
- **Urgent pointer** – Indicates the end of the urgent data
- **Option** – One option currently defined, maximum TCP segment size
- **Data** – Upper-layer protocol data

TCP session initiation



Well-known services that use TCP

- FTP (20/TCP);
- SSH (22/TCP);
- Telnet (23/TCP);
- SMTP (25/TCP);
- HTTP (80/TCP);
- HTTPS (443/TCP);

UDP (User Datagram Protocol)

- is a simple protocol that exchanges datagrams without guaranteed delivery
- relies on higher-layer protocols to handle errors and retransmit data
- does not use windows or ACKs

UDP Header

Offsets	Octet	0								1								2								3							
Octet	Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
0	0	Source port																Destination port															
4	32	Length																Checksum															

Well-known services that use UDP

- DNS (53/UDP);
- NTP (123/UDP);
- Online gaming;
- Video streaming services;
- RTP;
- SIP;

UDP and TCP in PortaSwitch

PortaBilling Master server

Subcomponent	Interacts with	Protocol	Ports	Details
Billing Engine	Main DB Server	TCP	3306/ 3307	MySQL Database server connection
RADIUS	PortaSIP® Cluster	UDP	1812/ 1813	Incoming RADIUS requests
Alerter	Main DB Server	TCP	3306/ 3307	MySQL Database server connection
Disconnector	Main DB Server	TCP	3306/ 3307	MySQL Database server connection
	Other nodes (VoIP, WiMAX, WiFi)	TFTP / HTTP		
Diameter	Other nodes (VoIP, WiMAX, WiFi)	UDP		Incoming Diameter requests

PortaBilling Web server

Protocols and ports used by the Web server

Subcomponent	Interacts with	Protocol	Ports	Details
TaskStack	Main DB Server	TCP	3306/ 3307	MySQL Database server connection
	Replica DB Server	TCP	3306/ 3307	MySQL Database server connection (SELECT requests only)
Apache / FCGI	PortaSIP® Media server / other nodes (VoiP, WiMAX, WiFi)	TFTP / HTTP		
	Main DB Server	TCP	3306/ 3307	MySQL Database server connection
	Replica DB Server	TCP	3306/ 3307	MySQL Database server connection (SELECT requests only)
Cassandra	Other nodes (VoiP, WiMAX, WiFi)	TFTP / HTTP		

PortaSIP Cluster

Subcomponent	Protocol	Ports	Details
Dispatching node			
EdgeProxy	UDP	5062	SIP port for control SIP traffic
EdgeProxy, SIP cluster protector	UDP	5060	production SIP traffic
EdgeProxy, SIP cluster protector	TCP	5060	production SIP traffic
EdgeProxy, SIP cluster protector	TLS	5061	encrypted production SIP traffic
Mail proxy	IMAP	8081	IMAP transport
Mail proxy	IMAPS	8091	IMAP over SSL
Mail proxy	SMTP	8101	SMTP transport
Mail proxy	UDP	5067	Mail Proxy control traffic
Processing Node controller	UDP	5063	Media Unit Controller UDP transport
Processing Node controller	TCP	5068	Telnet interface
Limit controller	UDP	5070	port for communication with the ProcessingNode controller
SMPP proxy	SMPP	2775	incoming SMPP connections
SMPP proxy	UDP	5064	port for communication with the ProcessingNode controller

Processing Node			
MWI	UDP	5264	The network port to bind the MWI daemon to
IMGate	SMPP	2775	incoming SMPP connection
IMGate	UDP	5960	port for incoming SIP SIMPLE messages
IMGate	TCP	5960	port for incoming SIP SIMPLE messages
IMGate	UDP	5961	port for outgoing SIP SIMPLE messages
IMGate	TCP	5961	port for outgoing SIP SIMPLE messages
IMGate	UDP	2775	port where the IMGate server creates a listening socket for SMPP connections
RTP proxy	UDP	35000–65000	RTP proxy port
B2BUA	UDP	5070	Base SIP port for B2BUA workers
B2BUA	TCP	5064	B2BUA telnet interface
B2BUA	TCP	5000	Call controller API
B2BUA	UDP	5059	B2BUA sibling communication
Registrar	TCP	5065	Registar transport ports
Registrar	UDP	5065	Registar transport ports
Subscription manager	TCP	5066	Subscription manager transport ports
Subscription manager	UDP	5066	Subscription manager transport ports
IMAP server	IMAP	143	IMAP transport
IMAP server	IMAPS	993	IMAP over SSL
Log Master	TCP	10000–10800	

Routing (static, dynamic, gateways)

Routing is the process of selecting a path for traffic in a network, or between or across multiple networks.

- it is a feature provided by capabilities of IP protocol.

IP routing provides a possibility to determine what addresses are **locally reachable** as opposed to **not directly known** destinations.

Any IP which is not on the machine itself or locally reachable, is only reachable through **another** IP routing device.

Given a destination IP address, **D**, and network prefix, **N**:
if (*N matches a directly connected network address*)
Deliver datagram to D over that network link;
else if (*The routing table contains a route for N*)
Send datagram to the next-hop address listed in the routing table;
else if (*a default route exists*)
Send datagram to the default route;
else
Send a forwarding error message to the originator;

Routers

- A router is a networking device that forwards data packets between computer networks.

- A router is connected to two or more data lines from different networks.

When a data packet comes in on one of the lines, the router reads the address information in the packet to determine the ultimate destination. Then, using information in its routing table or routing policy, it directs the packet to the next network on its journey.

Routing table

- Routing table is a data table stored in a router or a networked computer that lists the routes to particular network destinations, and in some cases, metrics (distances) associated with those routes.

- Static routes are entries made in a routing table by non-automatic means and not by the result of some network topology "discovery" procedure.

The routing table consists of at least three information fields:

1. the network id: i.e. the destination subnet
2. metric: metric (abstract distance or cost) of the path through which the packet is to be sent
3. next hop: The next hop, or gateway, is the address of the next station to which the packet is to be sent on the way to its final destination
4. interface: indicates what locally available interface is responsible for reaching the gateway

A default gateway in computer networking is the node that is assumed to know how to forward packets on to other networks. All packets for destinations not established in the routing table are sent via the default route.

Linux PC operating as a Router

● allows a PC on Linux OS to receive packets on one interface and transmit them on another

The process of accepting and transmitting IP packets is known as **forwarding**.

net/ipv4/ip_forward – enables/disables forwarding globally

net/ipv4/conf/$DEV/forward – to override the global value on a particular interface

Policy Based Routing

Usually, route selection is based **completely on the destination address using longest prefix match lookup** (most specific route to the destination will be chosen).

Since Linux kernel 2.2, **policy based routing** is supported through
- multiple routing tables;
- routing policy database (RPDB).

Now there are three routing table available: **local, default** and **main**

Utilities like “**netstat -nr**”, “**route -n**” or “**ip route**” (without specifying the table) show output of **main** table

Route Selection

Kernel route search order is:

- first in the **routing cache**
- then in the main **routing table**

The **routing cache** is a hash table used for quick access to recently used routes.

Using IP utility

Display IP addresses configuration:

> ip a | grep -A2 "eno[1-2]: "

```
2: eno1: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc mq state UP qlen 1000
   link/ether 00:1e:c9:ef:e7:3a brd ff:ff:ff:ff:ff:ff
   inet 78.40.240.208/27 brd 78.40.240.223 scope global eno1
--
3: eno2: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc mq state UP qlen 1000
   link/ether 00:1e:c9:ef:e7:3c brd ff:ff:ff:ff:ff:ff
   inet 78.40.244.35/24 brd 78.40.244.255 scope global eno2
```

Display routing information (from main table):

> ip route show

```
default via 78.40.244.1 dev eno2
5.144.80.0/20 via 78.40.240.201 dev eno1
10.0.0.0/9 via 78.40.244.13 dev eno2
10.1.1.1 via 78.40.244.13 dev eno2
78.40.240.192/27 dev eno1  proto kernel  scope link  src 78.40.240.208
78.40.244.0/24 dev eno2  proto kernel  scope link  src 78.40.244.35
128.1.0.0/16 via 78.40.244.13 dev eno2
169.254.0.0/16 dev eno1  scope link  metric 1002
```

Using IP utility

Display routing cache:
> ip route show cache

Display routing cache:
> **ip route flush cache**

Add new route:
ip route add <IP/Net> via <Gateway IP> dev <Int>
> **ip route add default via 192.168.1.1**
> ip route add 10.10.70.0/24 via 78.40.240.220 dev eno2

Note: when you add a new static route gateway must be reachable from the interface you add a static route to. So, it gateway should be from the same subnet or path to gateway should be specified beforehand in the routing table.

Delete route:
> **ip route del 10.10.70.0/24 via 78.40.240.220 dev eno2**

Change route:
> ip route change default via 78.40.244.2 dev eno2

Using IP utility

Check what route will be used to destination:

- ***ip route get to <IP>***

> ip route get to 8.8.8.8
> 8.8.8.8 via 192.168.192.2 dev eth1 src 192.168.198.7
> cache

Network Config Files

● are located in are located in the /etc/sysconfig/network-scripts/ directory

● three categories of files that exist in this directory:
- Interface configuration files
- Interface control scripts
- Network function files

Network Configuration Files

/etc/hosts – contains list of host names that cannot or shouldn't be resolved by DNS servers;

/etc/resolv.conf – specifies the IP addresses of DNS servers and the search domain;

/etc/sysconfig/network – specifies routing and host information for all network interfaces. It is used to contain directives which are to have global effect and not to be interface specific. Default gateway and interface for default gateway is usually defined there.

/etc/sysconfig/network-scripts/ifcfg-interface-name – network configuration specific for each network interface (IP, netmask, HWADD, boot protocol, etc.)
/etc/sysconfig/network-scripts/route-interface – to store static route configuration per-interface

Network Configuration Files

Saving static routes in file to survive server reboot:

> cat /etc/sysconfig/network-scripts/route-eno2

192.168.0.0/16 via 78.40.244.13
172.16.0.0/12 via 78.40.244.13
10.0.0.0/9 via 78.40.244.13
172.100.101.254/32 via 78.40.244.13
172.17.192.1/32 via 78.40.244.13
172.18.9.0/24 via 78.40.244.13
172.18.10.0/24 via 78.40.244.13
10.1.1.1/32 via 78.40.244.13
128.1.0.0/16 via 78.40.244.13
172.100.0.0/16 via 78.40.244.115
203.223.175.26 via 78.40.244.222
203.223.175.27 via 78.40.244.222
203.223.175.28 via 78.40.244.222

Why we should use command ‘ip’

Let’s check routing table using netstat –nr and route –n

==========

> netstat -nr

Kernel IP routing table

Destination	Gateway	Genmask	Flags	MSS	Window	irtt	Iface
0.0.0.0	91.228.242.93	0.0.0.0	UG	0	0	0	bond0.500
10.20.0.0	10.20.10.253	255.255.0.0	UG	0	0	0	bond0.10
10.20.10.0	0.0.0.0	255.255.255.0	U	0	0	0	bond0.10
91.228.242.64	0.0.0.0	255.255.255.192	U	0	0	0	bond0.500

==========

> route -n

Kernel IP routing table

Destination	Gateway	Genmask	Flags	Metric	Ref	Use	Iface
0.0.0.0	91.228.242.93	0.0.0.0	UG	0	0	0	bond0.500
10.20.0.0	10.20.10.253	255.255.0.0	UG	0	0	0	bond0.10
10.20.10.0	0.0.0.0	255.255.255.0	U	0	0	0	bond0.10
91.228.242.64	0.0.0.0	255.255.255.192	U	0	0	0	bond0.500

==========

Why we should use command ‘ip’

And now the same with “ip route show”

```
> ip r
default
        nexthop via 91.228.242.93  dev bond0.500 weight 1
        nexthop via 91.228.242.94  dev bond0.500 weight 1
10.20.0.0/16
        nexthop via 10.20.10.253  dev bond0.10 weight 1
        nexthop via 10.20.10.254  dev bond0.10 weight 1
10.20.10.0/24 dev bond0.10  proto kernel  scope link  src 10.20.10.65
91.228.242.64/26 dev bond0.500  proto kernel  scope link  src 91.228.242.65
```

So, it turns out that multi-path routing for load-balancing is configured, but only ip is able to see it.
That’s why we should get used to using IP for all networking-related operations

Channel bonding

Channel bonding enables two or more network interfaces to act as one, simultaneously increasing the bandwidth and providing redundancy.

The Linux bonding driver provides a method for aggregating multiple network interfaces into a single logical "bonded" interface. The behavior of the bonded interfaces depends upon the mode; generally speaking, modes provide either hot standby or load balancing services.

Additionally, link integrity monitoring is performed.

Bonding mode

mode=*<value>*

Allows specifies the bonding policy. The *<value>* can be one of:

balance-rr or **0** — Sets a round-robin policy for fault tolerance and load balancing. Transmissions are received and sent out sequentially on each bonded slave interface beginning with the first one available.

active-backup or **1** — Sets an active-backup policy for fault tolerance. Transmissions are received and sent out via the first available bonded slave interface. Another bonded slave interface is only used if the active bonded slave interface fails.

balance-xor or **2** — Sets an XOR (exclusive-or) policy for fault tolerance and load balancing. Using this method, the interface matches up the incoming request's MAC address with the MAC address for one of the slave NICs. Once this link is established, transmissions are sent out sequentially beginning with the first available interface.

broadcast or **3** — Sets a broadcast policy for fault tolerance. All transmissions are sent on all slave interfaces.

Bonding mode

802.3ad or **4** — Sets an IEEE 802.3ad dynamic link aggregation policy. Creates aggregation groups that share the same speed and duplex settings. Transmits and receives on all slaves in the active aggregator. Requires a switch that is 802.3ad compliant.
balance-tlb or **5** — Sets a Transmit Load Balancing (TLB) policy for fault tolerance and load balancing. The outgoing traffic is distributed according to the current load on each slave interface. Incoming traffic is received by the current slave. If the receiving slave fails, another slave takes over the MAC address of the failed slave. This mode is only suitable for local addresses known to the kernel bonding module and therefore cannot be used behind a bridge with virtual machines.
balance-alb or **6** — Sets an Adaptive Load Balancing (ALB) policy for fault tolerance and load balancing. Includes transmit and receive load balancing for **IPv4** traffic. Receive load balancing is achieved through **ARP** negotiation. This mode is only suitable for local addresses known to the kernel bonding module and therefore cannot be used behind a bridge with virtual machines.

Active-backup mode hint

For active-backup mode, Linux kernel sets the same MAC address for both enslavedinterfaces (it takes MAC address from the primary interface).
For example:
first (primary) slave interface has HWADDR: xx.xx.xx.xx
second slave interface - HWADDR: yy.yy.yy.yy
BUT "ip a" or “ifconfig” will show that both enslaved interfaces and bond interface have the same MAC xx.xx.xx.xx

Manual configuration of channel bonding

1) Make sure that bonding kernel module is loaded (use lsmod). Load it if it is not. Then create file /etc/modprobe.d/bonding.conf and write such line there:
alias bond<N> bonding
E.g.:
alias bond0 bonding

2) Create file /etc/sysconfig/network-scripts/ifcfg-bondX:

```
> cat ifcfg-bond0
DEVICE=bond0
IPADDR=83.245.1.152
NETMASK=255.255.255.192
BOOTPROTO=static
ONBOOT=yes
BONDING_OPTS="mode=1 arp_interval=60 arp_ip_target=83.245.1.129,83.245.1.157 primary=eth0"
NETWORK=83.245.1.128
```

Parameters for the bonding kernel module must be specified as a space-separated list in the **BONDING_OPTS="bonding parameters"** directive in the ifcfg-bondN interface file. Do not specify options for the bonding device in /etc/modprobe.d/bonding.conf, or in the deprecated /etc/modprobe.conf file!

Manual configuration of channel bonding

3) Add the MASTER and SLAVE directives to their configuration files of the network interfaces to be bound together:

> cat ifcfg-eth0

MASTER=bond0
SLAVE=yes
ONBOOT=yes
USERCTL=no
BOOTPROTO=none

> cat ifcfg-eth3

DEVICE=eth3
USERCTL=no
ONBOOT=yes
MASTER=bond0
SLAVE=yes
BOOTPROTO=none

- sudo systemctl restart network

That's all!

Manual configuration of channel bonding

```
bond0 Link encap:Ethernet  HWaddr D4:AE:52:BA:53:CF
      inet addr:83.245.1.152  Bcast:83.245.1.191  Mask:255.255.255.192
      inet6 addr: fe80::d6ae:52ff:feba:53cf/64 Scope:Link
      UP BROADCAST RUNNING MASTER MULTICAST  MTU:1500  Metric:1
      RX packets:8275135860 errors:0 dropped:0 overruns:0 frame:0
      TX packets:1593338699 errors:0 dropped:0 overruns:0 carrier:0
      collisions:0 txqueuelen:0
      RX bytes:680218213821 (633.5 GiB)  TX bytes:506990210076 (472.1 GiB)
eth0   Link encap:Ethernet  HWaddr D4:AE:52:BA:53:CF
      UP BROADCAST RUNNING SLAVE MULTICAST  MTU:1500  Metric:1
      RX packets:4565320948 errors:0 dropped:0 overruns:0 frame:0
      TX packets:1593333676 errors:0 dropped:0 overruns:0 carrier:0
      collisions:0 txqueuelen:1000
      RX bytes:442512427362 (412.1 GiB)  TX bytes:506989760258 (472.1 GiB)
      Interrupt:36 Memory:d6000000-d6012800
eth3   Link encap:Ethernet  HWaddr D4:AE:52:BA:53:CF
      UP BROADCAST RUNNING SLAVE MULTICAST  MTU:1500  Metric:1
      RX packets:3709814912 errors:0 dropped:0 overruns:0 frame:0
      TX packets:5023 errors:0 dropped:0 overruns:0 carrier:0
      collisions:0 txqueuelen:1000
      RX bytes:237705786459 (221.3 GiB)  TX bytes:449818 (439.2 KiB)
      Interrupt:42 Memory:dc000000-dc012800
```

Useful SYSFS commands

To view all existing bonds, even if they are not up:

> *cat /sys/class/net/bonding_masters*

bond0

Check slave interfaces:

> *cat /sys/class/net/bond0/bonding/slaves*

eth0 eth3

Check primary slave:

> *cat /sys/class/net/bond0/bonding/primary*

eth0

Check active slave:

> *cat /sys/class/net/bond0/bonding/active_slave*

eth0

Check ARP IP targets:

> cat /sys/class/net/bond0/bonding/arp_ip_target

83.245.1.129 83.245.1.157

Useful SYSFS commands

Bonding statistics:

> cat /proc/net/bonding/bond0

Ethernet Channel Bonding Driver: v3.6.0 (September 26, 2009)
Bonding Mode: fault-tolerance (active-backup)
Primary Slave: eth0 (primary_reselect always)
Currently Active Slave: eth0
MII Status: up
MII Polling Interval (ms): 0
Up Delay (ms): 0
Down Delay (ms): 0
ARP Polling Interval (ms): 60
ARP IP target/s (n.n.n.n form): 83.245.1.129, 83.245.1.157
Slave Interface: eth0
MII Status: up
Speed: 1000 Mbps
Duplex: full
Link Failure Count: 80
Permanent HW addr: d4:ae:52:ba:53:cf
Slave queue ID: 0
Slave Interface: eth3
MII Status: up
Speed: 1000 Mbps
Duplex: full
Link Failure Count: 58
Permanent HW addr: d4:ae:52:ba:53:d5
Slave queue ID: 0

Bonding parameters

arp_interval=<time_in_milliseconds>
Specifies (in milliseconds) how often ARP monitoring occurs.
It is essential that both arp_interval and arp_ip_target parameters are specified, or, alternatively, the miimon parameter is specified.

The ARP monitor works by periodically checking the slave devices to determine whether they have sent or received traffic recently (the precise criteria depends upon the bonding mode, and the state of the slave). Regular traffic is generated via ARP probes issued for the addresses specified by the arp_ip_target option.

It is critical that either the miimon or arp_interval andarp_ip_target parameters be specified, otherwise serious network degradation will occur during link failures. Very few devices do not support at least miimon, so there is really no reason not to use it.

arp_ip_target=<ip_address>[,<ip_address_2>,…<ip_address_16>]
Specifies the target IP address of ARP requests when the arp_interval parameter is enabled. At least one IP address must be given for ARP monitoring to function.

Bonding parameters

downdelay=<time_in_milliseconds>
Specifies (in milliseconds) how long to wait after link failure before disabling the link. This option is only valid for the miimon link monitor.

updelay=<time_in_milliseconds>
Specifies the time, in milliseconds, to wait before enabling a slave after a link recovery has been detected. This option is only valid for the miimon link monitor. The updelay value should be a multiple of the miimon value; if not, it will be rounded down to the nearest multiple. The default value is 0.

miimon=<time_in_milliseconds>
Specifies (in milliseconds) how often MII link monitoring occurs.
Specifies the MII link monitoring frequency in milliseconds. This determines how often the link state of each slave is inspected for link failures. A value of zero disables MII link monitoring. A value of 100 is a good starting point.

primary=<interface_name>
Specifies the interface name, such as eth0, of the primary device. The primary device is the first of the bonding interfaces to be used and is not abandoned unless it fails. A string (eth0, eth2, etc) specifying which slave is the primary device. The specified device will always be the active slave while it is available. Only when the primary is off-line will alternate devices be used. This is useful when one slave is preferred over another, e.g., when one slave has higher throughput than another.

RHEL 7 peculiarities

According to documentation configuration file for master bonding interface has to have **TYPE=Bond** parameter (in case if it is controlled by **Network Manager**!)

Starting from RHEL7 we have a new feature – teaming. Basically, it uses the same concept as channel bonding but it is supposed to have some enhancements over traditional bonding.
We do not support it for now.

Network Manager

In Red Hat Enterprise Linux 7, the default networking service is provided by NetworkManager, which is a dynamic network control and configuration daemon that attempts to keep network devices and connections up and active when they are available. But the traditional ifcfg type configuration files are still supported.

In PortaSwitch NetworkManager.service is enabled, but it doesn't manage devices (i.e. old network script method is still used). It is planned to switch to NetworkManager in MR63+.

Application or Tool	**Description**
NetworkManager	The default networking daemon
nmtui	A simple curses-based text user interface (TUI) for NetworkManager
nmcli	A command-line tool provided to allow users and scripts to interact with NetworkManager Dwddddddayyayaaya

Terms of Network Manager

NM operates with the following terms: **Connection** and **Device**

Device represents physical interface (eno1, em1, etc) and **Connection** represents a number of settings typical for different types of connections (e.g. DHCP, Wi-fi, static, VPN) and describes settings such as IP address, DNS servers, etc.

NM manages connections. For one specific device (e.g. eno1) there may be a lot of different connections, but only one can be active at the same time.

Using nmcli to manipulate with networking

NetworkManager can configure network aliases, IP addresses, static routes, DNS information, and VPN connections, as well as many connection-specific parameters.

Show NM connections:
nmcli connection show

Show settings of a specific connection:
nmcli connection show <con name>

Show only active connections:
nmcli connection show --active

Show all devices:
nmcli device status

Modify connection:
nmcli connection modify <con name> <attributes>
nmcli connection modify eno01 +ipv4.dns 8.8.8.8

Edit connection via interactive console:
nmcli connection edit <con name>

Using nmcli to manipulate with networking

Activate connection:
nmcli connection up <con name>

Shutdown connection:
nmcli con down <con name>
nmcli dev disconnect <device name>

Create a connection:
nmcli con add type ethernet con-name test-lab ifname eno1 ip4 10.10.10.10/24 gw4 10.10.10.254

Reload connections (re-read configuration files and re-activate them):
nmcli connection reload

More commands can be found in the official Red Hat 7 manual on NMCLI:
https://access.redhat.com/documentation/en-US/Red_Hat_Enterprise_Linux/7/html/Networking_Guide/sec-Using_the_NetworkManager_Command_Line_Tool_nmcli.html

How to disable Network Manager

Add option **“NM_CONTROLLED=NO”** to /etc/sysconfig/network-scripts/ifcfg-<name> scripts. After that ***nmcli connection reload*** command should be issued to take the changes into effect.
Disable NetworkManager completely (if needed):

sudo systemctl stop NetworkManager.service
sudo systemctl mask NetworkManager.service

HOW TO RESTRICT MODIFICATION OF RESOLV.CONF BY NETWORK MANAGER

Add **“dns=none”** option to /etc/NetworkManager/NetworkManager.conf;
Restart NM:

sudo systemctl restart NetworkManager.service

After that Network Manager will stop updating /etc/resolv.conf even if there are new DNS servers added to connections (either via nmcli or to ifcfg-* scripts manually).

Thank you!