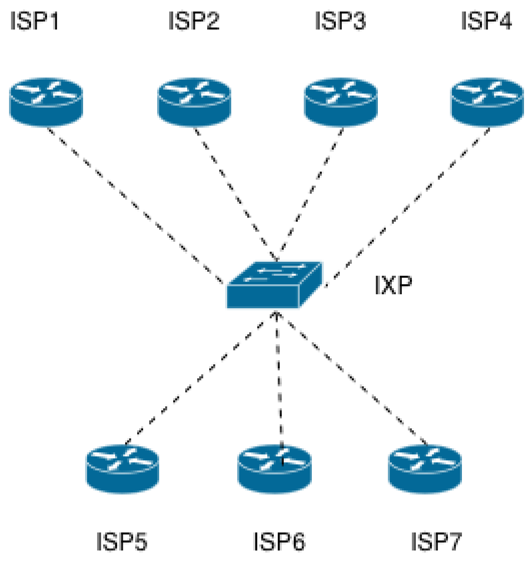
Route Servers

Internet Exchange Points often provide one or more router server instances on the peering LAN. This allows participants to have a much simpler BGP setup (especially on large exchanges).

IXP Without Route Servers

In the following diagram, ISPs at this small exchange would need still need to build a large number of peering sessions to be able to pass traffic amongst themselves.



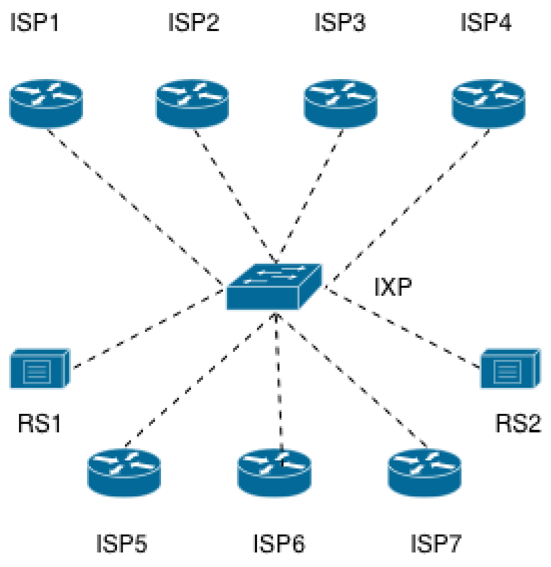
They'd each need to configure 6 BGP sessions to the other BGP peers at the exchange for a total of 21 peering sessions

	ISP1	**ISP2**	**ISP3**	**ISP4**	**ISP5**	**ISP6**	**ISP7**
ISP1		X	X	X	X	X	X
ISP2	X		X	X	X	X	X
ISP3	X	X		X	X	X	X
ISP4	X	X	X		X	X	X
ISP5	X	X	X	X		X	X
ISP6	X	X	X	X	X		X
ISP7	X	X	X	X	X	X	

Q: What happens when a new peer joins the exchange? How many **new** peering sessions are needed?

IXP With Route Servers

The following diagram shows an exchange where two BGP route servers that are provided by the exchange operator.



By peering with two BGP route servers, ISPs only need to configure two BGP sessions each for a total of 14 peering sessions.

	ISP1	**ISP2**	**ISP3**	**ISP4**	**ISP5**	**ISP6**	**ISP7**
RS1	X	X	X	X	X	X	X
RS2	X	X	X	X	X	X	X

Q: What happens when a new peer joins the exchange? How many **new** peering sessions are needed?

The BIRD Internet Routing Daemon

The BIRD project aims to develop a fully functional dynamic IP routing daemon primarily targeted on (but not limited to) Linux, FreeBSD and other UNIX-like systems and distributed under the GNU General Public License

You can find more information at BIRD

Configuring BIRD as an IPv4 route server

We have configured a very simple route server configuration based on the example at:

- https://gitlab.labs.nic.cz/labs/bird/wikis/Simple_route_server

There are two route servers for each IXP in the lab. The rs1 server on each IXP is configured but the second route server on each exchange needs configured.

You can connect to the two route servers in the lab using:

```
ssh sysadm@rs2.ixp1.ws.nsrc.org
ssh sysadm@rs2.ixp2.ws.nsrc.org
```

Connect to correct server for your IXP and as **root** go to the directory where the configs are stored:

```
sudo -s
cd /etc/bird/
```

You should be able to see two config files there called **bird6.conf** and **bird.conf**

Normally you wouldn't have access to these files on a production IXP but we're going to get you to configure your own session in this workshop.

The file **bird.conf** should contain something that looks like:

```
/*
 *    Route server configuration - generated by './mk-rs-config.sh'
 */
log syslog all;
timeformat protocol "%s";

# Override router ID
router id 192.168.1.9;

protocol kernel {
        import none;            # Default is import all
        export none;            # Default is export none
}

# This pseudo-protocol watches all interface up/down events.
protocol device {
}

template bgp template_64010 {
   local as 64010;
   source address 192.168.1.9;
   import all;
   export all;
   import keep filtered;
   import limit 10000 action restart;
   rs client;
}
```

You don't need to change anything in this section!

You should be able to look at these commands and see some similarities between the Cisco configs you have been working on earlier in the week.

Adding details for a peer

The details below show the **additional** configuration needed for each ISP. Find the details for your ISP and add them to the end of file **bird.conf**.

You will need to be careful that no one else is editing the file at the same time. Your instructor will arrange a method of managing this during the class - if that hasn't been done ask before you start!

Remember that in a production network there are two route servers, **rs1** and **rs2** - normally these changes need to be done on both servers.

```
protocol bgp isp1 from template_64010 {
   description "eBGP - ISP1";
   neighbor 192.168.1.1 as 64011;
   password "cisco;
}

protocol bgp isp2 from template_64010 {
   description "eBGP - ISP2";
   neighbor 192.168.1.2 as 64012;
   password "cisco;
}

protocol bgp isp3 from template_64010 {
   description "eBGP - ISP3";
   neighbor 192.168.1.3 as 64013;
   password "cisco;
}

protocol bgp isp4 from template_64010 {
   description "eBGP - ISP4";
   neighbor 192.168.1.4 as 64014;
   password "cisco;
}

protocol bgp isp5 from template_64010 {
   description "eBGP - ISP5";
   neighbor 192.168.1.5 as 64015;
   password "cisco;
}

protocol bgp isp6 from template_64010 {
   description "eBGP - ISP6";
   neighbor 192.168.1.6 as 64016;
   password "cisco;
}
```

```
protocol bgp isp7 from template_64010 {
   description "eBGP - ISP7";
   neighbor 192.168.1.7 as 64017;
   password "cisco;
}
```

Loading the new configuration

You need to tell the running **BIRD** process to reconfigure itself based on the changes you made. We do this using the command **birdc**:

```
root@rs1-ixp1:/etc/bird# birdc
BIRD 1.4.5 ready.
bird> configure
Reading configuration from /etc/bird/bird.conf
Reconfigured
bird>
```

Let's assume we are adding the changes for ISP2. While we're still in the **birdc** program we can run the command:

```
bird> sh protocols isp2
name     proto    table    state  since       info
isp2     BGP      master   up     1433461446  Established
bird>
```

This shows that the BGP session is now working.

And then we can check the prefixes being advertised:

```
bird> sh route protocol isp2
100.66.12.0/23     via 192.168.1.6 on eth1 [isp2 2015-06-04 from
192.168.1.2] (100) [AS64016i]
100.66.14.0/23     via 192.168.1.7 on eth1 [isp2 2015-06-04 from
192.168.1.2] (100) [AS64017i]
100.66.8.0/23      via 192.168.1.4 on eth1 [isp2 2015-06-04 from
192.168.1.2] (100) [AS64014i]
100.66.10.0/23     via 192.168.1.5 on eth1 [isp2 2015-06-04 from
192.168.1.2] (100) [AS64015i]
100.66.4.0/23      via 192.168.1.2 on eth1 [isp2 2015-06-04] * (100)
[AS64012i]
100.66.6.0/23      via 192.168.1.3 on eth1 [isp2 2015-06-04 from
192.168.1.2] (100) [AS64013i]
100.66.2.0/23      via 192.168.1.1 on eth1 [isp2 2015-06-04 from
192.168.1.2] (100) [AS64011i]
bird>
```

When you're happy that things are working use the **exit** command.

Configuring BIRD as an IPv6 route server

You can connect to the two route servers in the lab using:

```
ssh sysadm@rs1.ixp1.ws.nsrc.org
ssh sysadm@rs2.ixp1.ws.nsrc.org
```

Connect to one of these and as **root** go to the directory where the configs are stored:

```
sudo -s
cd /etc/bird/
```

You should be able to see two config files there called **bird6.conf** and **bird.conf**

Normally you wouldn't have access to these files on a production IXP but we're going to get you to configure your own session in this workshop.

The file **bird6.conf** should contain something that looks like:

```
/*
 *    Route server configuration - generated by './mk-rs-config.sh'
 */
log syslog all;
timeformat protocol "%s";

# Override router ID
router id 192.168.1.8;

protocol kernel {
        import none;            # Default is import all
        export none;            # Default is export none
}

# This pseudo-protocol watches all interface up/down events.
protocol device {
}

template bgp template_64010 {
   local as 64010;
   source address fd90:192:168:1::8;
   import all;
   export all;
   import keep filtered;
   import limit 10000 action restart;
   rs client;
}
```

You don't need to change anything in this section!

You should be able to look at these commands and see some similarities between the Cisco configs you have been working on earlier in the week.

Adding details for a peer

The details below show the **additional** configuration needed for each ISP. Find the details for your ISP and add them to the end of file **bird.conf**.

You will need to be careful that no one else is editing the file at the same time. Your instructor will arrange a method of managing this during the class - if that hasn't been done ask before you start!

Remember that there are two route servers, **rs1** and **rs2** - these changes need to be done on both servers.

```
protocol bgp isp1 from template_64010 {
   description "eBGP - ISP1";
   neighbor fd90:192:168:1::1 as 64011;
   password "cisco;
}

protocol bgp isp2 from template_64010 {
   description "eBGP - ISP2";
   neighbor fd90:192:168:1::2 as 64012;
   password "cisco;
}

protocol bgp isp3 from template_64010 {
   description "eBGP - ISP3";
   neighbor fd90:192:168:1::3 as 64013;
   password "cisco;
}

protocol bgp isp4 from template_64010 {
   description "eBGP - ISP4";
   neighbor fd90:192:168:1::4 as 64014;
   password "cisco;
}

protocol bgp isp5 from template_64010 {
   description "eBGP - ISP5";
   neighbor fd90:192:168:1::5 as 64015;
   password "cisco;
}

protocol bgp isp6 from template_64010 {
   description "eBGP - ISP6";
   neighbor fd90:192:168:1::6 as 64016;
   password "cisco;
}
```

```
protocol bgp isp7 from template_64010 {
   description "eBGP - ISP7";
   neighbor fd90:192:168:1::7 as 64017;
   password "cisco;
}
```

Loading the new configuration

You need to tell the running **BIRD6** process to reconfigure itself based on the changes you made. We do this using the command **birdc6**:

```
root@rs1-ixp1:/etc/bird# birdc6
BIRD 1.4.5 ready.
bird> configure
Reading configuration from /etc/bird/bird6.conf
Reconfigured
bird>
```

Let's assume we are adding the changes for ISP5. While we're still in the **birdc** program we can run the command:

```
bird> sh protocols isp5
name     proto    table    state  since       info
isp2     BGP      master   up     1433461451  Established
bird>
```

This shows that the BGP session is now working.

And then we can check the prefixes being advertised:

```
bird> sh route protocol isp5
fd90:100:66:10::/60 via fd90:192:168:1::5 on eth1 [isp5 2015-06-04] * (100)
[AS64015i]
bird>
```

When you're happy that things are working use the **exit** command.