3. ROTEAMENTO INTER-AS VIA BGP EM SDN/OPENFLOW

Esta seção apresenta uma visão geral sobre os conceitos e funcionamento do roteamento inter-domínio (entre diferentes Sistemas Autônomos) utilizando o protocolo BGP e sua integração com a arquitetura SDN/Openflow.

3.1 Roteamento Inter-AS

Uma forma de analisar a Internet é observá-la como um conjunto de redes autônomas interconectadas, sob a mesma gerência técnica, compartilhando uma política de roteamento interno e externo bem definida (HAWKINSON; BATES, 1966). A esse conjunto de redes dar-se o nome Sistema Autônomo, ou AS (do inglês *Autonomous System*), e cada AS é identificado por um número de 16 ou 32 *bits* (também conhecido como ASN, do inglês *AS Number*). A partir desse conceito de AS, é possível dividir o roteamento na Internet em dois tipos: roteamento interior ao AS, cujos protocolos mais utilizados são OSPF, RIP e ISIS, e roteamento externo, ou roteamento entre AS's, cujo protocolo utilizado é o BGP (do inglês, *Border Gateway Protocol*).

Quando um par de sistemas autônomos concorda em trocar informações de roteamento, cada um precisa designar um roteador para conversar com o roteador do outro AS; neste caso, os dois roteadores se tornam "vizinhos BGP" um do outro. Em geral, o processo de roteamento em cada AS tem origem no armazenamento das rotas de redes pertencentes ao próprio AS, aprendidas através de protocolos de roteamento interno. Posteriormente, o roteador de borda (ou roteadores de borda) de cada AS inicia uma sessão BGP entre si. Neste caso, temos uma sessão BGP externa, ou simplesmente, *eBGP*. Caso a sessão BGP seja feita entre dois roteadores de um mesmo AS, então temos uma sessão BGP interna ou sessão *iBGP*. A Figura 3.1 mostra a troca de informações de roteamento entre AS's: o AS1 envia suas informações para os AS's vizinhos (AS2 e AS3) e essas informações são encaminhadas até chegar ao AS5. O AS5 armazena os caminhos para o AS1 e envia o melhor caminho como resposta.

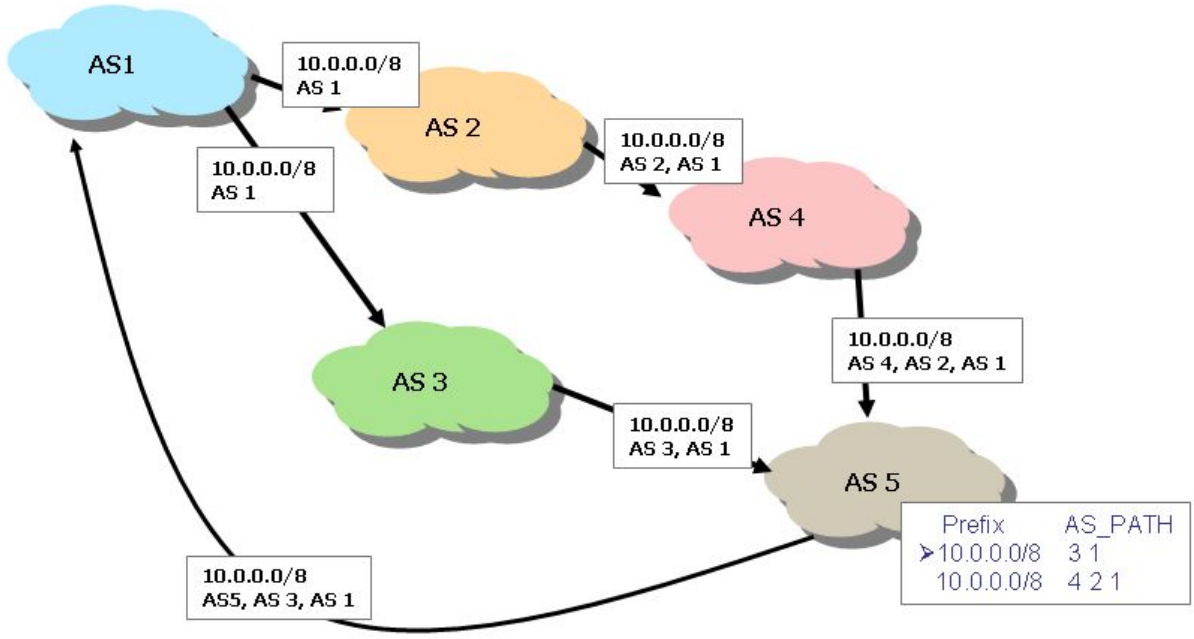


Figura 3.1 - Troca de informações de roteamento entre ASs

O BGP é um protocolo de roteamento bastante flexível e robusto, que foi projetado para ser escalável e evitar *loops* de roteamento em topologias arbitrárias. Ele destaca-se pela facilidade de definir roteamento baseado em políticas e por ser fundamentado em um conjunto de regras não técnicas definidas pelos Sistemas Autônomos.

Um roteador BGP constrói sua tabela de roteamento a partir de informações trocadas pelos "vizinhos BGP" (*BGP neighbors*), cuja atualização é feita de forma incremental. A atualização completa da tabela de roteamento é realizada apenas quando se estabelece a sessão entre *neighbors*, ou seja, apenas nesse primeiro momento a tabela completa é enviada entre os vizinhos. A flexibilidade e possibilidade de roteamento baseado em políticas do BGP estão intimamente ligadas ao conjunto de atributos que ele disponibiliza, bem como ao algoritmo de tomada de decisão na escolha de rotas. Alguns atributos do BGP são:

- AS-Path: seqüência de ASNs que uma rota cruza para alcançar uma determinada rede de destino - rotas com menor AS-path são preferidas;
- Local-Pref: atributo usado para influenciar na definição do caminho preferencial de saída para uma determinada rota, usado apenas entre roteadores iBGP;

- MED: o atributo MED (do inglês, *Multi Exit Discriminator*) tem por objetivo informar para os vizinhos BGP externos qual o melhor caminho para uma determinada rota do próprio AS, influenciando assim o tráfego de entrada do AS;
- Community: atributo utilizado para marcar as rotas com alguma característica que as agrupa (ex: rotas aprendidas a partir do cliente X, rotas que não devem ser repassadas, etc);
- Weight: apesar de não ser propriamente um atributo BGP, é utilizado para decisão local da rota de saída em um roteador BGP;
- Next-Hop: este atributo recebe o endereço IP da interface do próximo roteador, cujo valor varia de acordo com o tipo da sessão: eBGP ou iBGP.

A configuração do protocolo BGP pode ser dividida em configurações básicas e avançadas, e depende da plataforma em que se está configurando (Cisco, Juniper, Extreme, Brocade, Quagga, BIRD, GoBGP, etc). Considerando o escopo deste curso, serão abordadas apenas as configurações básicas demonstradas na plataforma Quagga. As tarefas de configuração básica consistem em:

- **Habilitar o roteamento BGP**: este passo é necessário para habilitar o processo do BGP no roteador em questão, indicando qual o ASN do provedor, o IP que identifica o roteador em que se está configurando e a rede que ele irá anunciar. Para isso use os seguintes comandos no modo de configuração global no console do roteador:
    - router bgp <asn> (ex: router bgp 100)
    - bgp router-id <ip-addr> (ex: bgp router-id 10.0.0.2)
    - network <prefix/mask> (ex: network 192.168.0.0/16)
- **Configurar vizinhos BGP (neighbors)**: é necessário configurar os vizinhos BGP manualmente, sejam eles eBGP ou iBGP. Na configuração do vizinho é preciso especificar o endereço IP e o número AS do roteador remoto, através do seguinte comando:
    - neighbor <ip-addr> remote-as <asn> (ex: neighbor 10.0.0.1 remote-as 200)

Outras configurações possíveis são: BGP Soft Reconfiguration (que permite alteração em configurações sem reiniciar a sessão BGP); BGP In/Out Policies (permite a configuração de políticas de filtragem BGP de entrada ou saída); etc.

3.2 Integração entre BGP e SDN/Openflow

Uma outra maneira de implantar a comunicação Inter-AS via BGP é através da arquitetura SDN/Openflow. A implementação de BGP através do *Openflow* tem sido bem sucedida em diversos casos demonstrados na indústria e na academia, permitindo a ampliação das capacidades do BGP, facilidade no gerenciamento de políticas e até mesmo escalabilidade. Alguns exemplos de iniciativas que têm sido conduzidas nesse sentido são listadas a seguir:

- RouteFlow[8]: é um projeto *open source* que tem o objetivo de prover serviços de roteamento IP virtualizados sobre qualquer *hardware* que tenha *Openflow* habilitado. A intenção é criar uma arquitetura de roteamento que possa ser executada em computadores de propósito geral, combinando o desempenho dos *hardwares* comerciais com a flexibilidade de uma camada de roteamento de código aberto. Assim, espera-se fornecer: migração de protocolos da rede legada para SDN; um *framework open-source* para virtualização de rede; modelos de rede do tipo RaaS (do inglês, *Routing-as-a-Service*); além de uma forma simples de fazer roteamento intra e inter-domínio que tenha interoperabilidade entre equipamentos legados.
- SIR – SDN Internet Router[9]: é um agente que pode ser adicionado ao roteador para tornar visível e disponibilizar informações que não poderiam ser expostas por si só, como tabelas BGP e tráfego por prefixo BGP ou por ASN. Essas informações podem ser utilizadas para fazer engenharia de tráfego, planejamento de capacidade, melhorar a decisão das rotas instaladas na FIB, entre outras soluções, e podem ser utilizadas em diferentes equipamentos pois utilizam BGP e netflow/sflow/ipfix na obtenção dos dados.
- Software Defined Internet Exchange Point[10] (SDX): Tenta trazer os benefícios de SDN para o roteamento interdomínio, principalmente em IXPs (do inglês, *Internet Exchange Point*), devido à sua capacidade de conectar diversas redes e de deixar os conteúdos mais populares mais próximos dos usuários. Com a utilização de SDN, é possível oferecer um controle direto sobre as regras de processamento de pacotes, considerando diversos campos dos cabeçalhos de rede e uma grande variedade de

[8] http://routeflow.github.io/RouteFlow/
[9] https://github.com/dbarrosop/sir
[10] http://sdx.cs.princeton.edu/
https://noise-lab.net/projects/software-defined-networking/sdx/

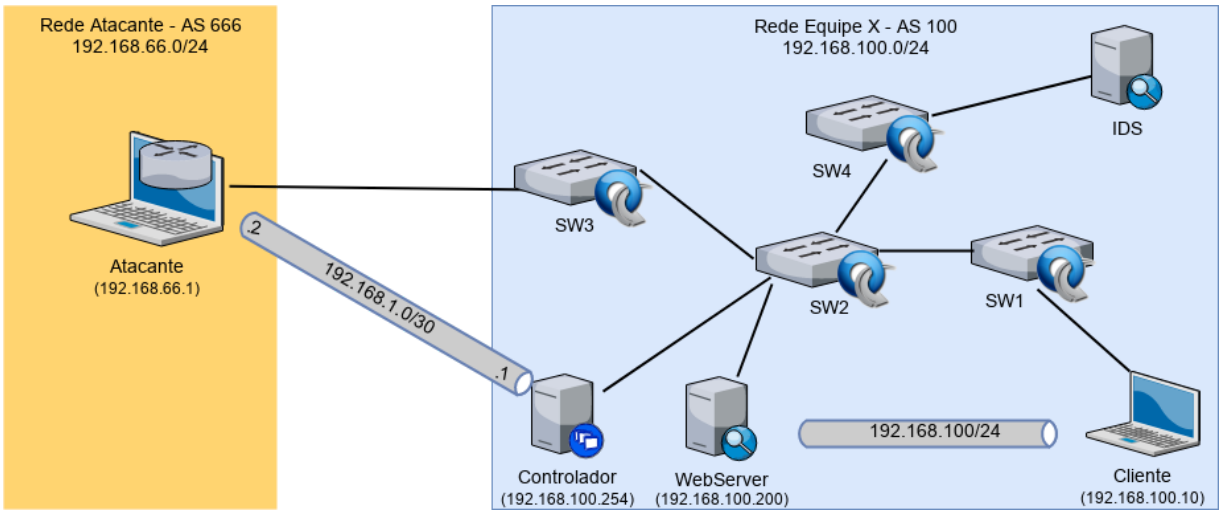
ações. Assim, a proposta de SDX tem a capacidade de habilitar novas aplicações, tais como: troca de tráfego específico por aplicação, por exemplo com um *peering* apenas para tráfego de streaming de vídeo; escalabilidade, em relação ao tamanho da tabela e à sobrecarga computacional; engenharia de tráfego de entrada, inclusive utilizando políticas administrativas; redirecionamento de tráfego para *middleboxes*; servidores de balanceamento de carga; bloqueio de tráfego indesejado, etc.

3.3 Exercícios de fixação

1. O que é um AS e como os AS's se comunicam na Internet?
2. Quais são os principais atributos para seleção de rotas BGP?
3. Como SDN pode ser empregada no roteamento Inter-AS? Quais os benefícios?
4. Como o Controlador pode configurar as rotas BGP nos comutadores OpenFlow?

3.4 Roteiro de laboratório

O objetivo deste laboratório é construir e utilizar uma aplicação SDN/Openflow para estabelecimento de sessões BGP no ambiente de teste FIBRE entre o AS 100 (rede do aluno) e AS 666 (rede hipotética do atacante). A topologia utilizada neste laboratório é a mesma anterior, conforme Figura 3.2.



3.2. Topologia proposta para os experimentos da Oficina.

Grande parte das configurações desta prática serão executadas através do console de alguma das máquina virtuais (Controlador, Atacante ou Cliente), através uma conexão SSH. Para saber o endereço IP da máquina você deverá verificar a interface do OCF.

3.4.1 Integrando BGP na aplicação SDN-IPS

A aplicação SDN-IPS é baseada no controlador Ryu versão 4.15, sendo divida em dois principais componentes: i) SDNIPSApp, classe responsável pelo tratamento dos eventos Openflow, por armazenar e gerenciar a topologia da rede e por atender a requisições de caminho na instalação de fluxos através da API sul baseada em Openflow 1.0; ii) SDNIPSWSGIApp, classe responsável pela interface REST da aplicação, que recebe as requisições da API norte do usuário e requisita serviços à aplicação SDNIPSApp. Nesta prática o aluno deverá alterar a classe SDNIPSApp e implementar as funções que são utilizadas para estabelecimento da sessão BGP, a saber: bgp_create(), bgp_add_neighbor() e bgp_add_prefix(). Ambas as funções farão uso da biblioteca BGP Speaker do Ryu, conforme demonstrado a seguir.

1) O primeiro passo é editar a função bgp_create() a fim de programá-la para que ela utilize a biblioteca do Ryu para configurar os parâmetros básicos do BGP (número AS e ID do roteador). Modifique a função bgp_create() no arquivo fibre-2opencall-sdn-ips.py complementando-a para que ela fique como mostrado abaixo (altere a parte que possui um comentário "# MUDAR AQUI - INICIO" até "# MUDAR AQUI - FIM"):

```
def bgp_create(self, as_number, router_id):
    # MUDAR AQUI - INICIO
    try:
        self.bgp_speaker = BGPSpeaker(as_number=as_number, router_id=router_id,
                        best_path_change_handler=self.best_path_change_handler,
                        peer_down_handler=self.peer_down_handler,
                        peer_up_handler=self.peer_up_handler)
    except:
        return (False, 'Failed to create BGP speaker')
    # MUDAR AQUI - FIM
```

```
    ...
```

Atente-se para respeitar as indentações do código original e do trecho acima. Em linhas gerais esse código apenas cria uma instância da classe BGPSpeaker(), configurando o número AS e o ID do roteador. Deve-se notar também a presença de três funções para tratar eventos do BGP, a saber: best_path_change_handler() - chamada sempre que uma rota é inserida ou removida (parâmetro is_withdraw=True) no processo de aprendizagem do BGP; peer_down_handler() - chamada sempre que uma vizinhança BGP é estabelecida; e peer_up_handler() - sempre que uma vizinhança BGP é desfeita. Essas funções são importantes para que a aplicação SDN faça alguma modificação no comportamento padrão do BGP (conforme exemplos apresentados na seção de aprendizagem 3.2). Estas funções já foram previamente definidas no código, recomenda-se que o leitor analise-as para entender sua lógica de funcionamento.

2) Em seguida, é necessário definir a função de configuração de prefixos BGP. Para isso, modifique a função bgp_add_prefix() no arquivo fibre-2opencall-sdn-ips.py complementando-a para que ela fique como mostrado abaixo (altere a parte que possui um comentário "# MUDAR AQUI - INICIO" até "# MUDAR AQUI - FIM"):

```
def bgp_add_prefix(self, prefix):
    # MUDAR AQUI - INICIO
    try:
        self.bgp_speaker.prefix_add(prefix)
    except:
        return (False, 'Failed to add prefix')
    # MUDAR AQUI - FIM
    ...
```

Atente-se para respeitar as indentações do código original e do trecho acima. O leitor irá notar que a função acima faz uso da instância do BGP Speaker criada anteriormente e apenas insere um novo prefixo aos anúncios BGP do roteador atual. Vale salientar que os prefixos anunciados via BGP nesta biblioteca podem ser de qualquer família de endereços (IPv4, IPv6, L2VPNv4, etc), embora este laboratório limitar-se-á à IPv4.

3) Por fim, será definida a função que faz o registro de novos vizinhos BGP. A partir das informações do vizinho (endereço IP e número AS remoto), a biblioteca BGP Speaker tenta estabelecer a sessão BGP. Para isso é preciso modificar a função bgp_add_neighbor() no arquivo fibre-2opencall-sdn-ips.py complementando-a para que fique conforme mostrado abaixo (altere a parte que possui um comentário "# MUDAR AQUI - INICIO" até "# MUDAR AQUI - FIM"):

```
def bgp_add_neighbor(self, address, remote_as):
    # MUDAR AQUI - INICIO
    try:
        self.bgp_speaker.neighbor_add(address, remote_as)
    except:
        return (False, 'Failed to add BGP neighbor')
    # MUDAR AQUI - FIM
    ...
```

Atente-se para respeitar as indentações do código original e do trecho acima. O leitor irá notar que a função acima faz uso da instância do BGP Speaker criada anteriormente e apenas insere uma chamada à função de adição de vizinhos do BGP Speaker. Vale destacar que esta função possui diversos parâmetros adicionais (ex: modo de conexão passivo ou ativo, senha MD5, valor do atributo MED, conjunto de *address families* suportados, etc), recomenda-se a leitura da API do BGP Speaker para mais informações.

3.4.2 Configurando a sessão BGP do AS 100 na aplicação SDN-IPS

1) Esta prática pressupõe que o roteiro anterior do capítulo tenha sido executado com sucesso. Portanto, caso tenha desligado o ambiente ao final da prática anterior, é necessário religar o controlador Ryu. Ele irá recarregar as configurações que foram realizadas anteriormente a partir do arquivo **`sdn-ips-config.json`** na mesma pasta da aplicação.

2) Na máquina Controlador (em outra aba), vamos utilizar a API REST da aplicação SDN-IPS para fazer a configuração da sessão BGP. O primeiro passo é configurar o ASN e Router-ID da sessão BGP. Para isso execute o seguinte comando:

```
curl -s -X POST -d '{"as_number" : 100, "router_id" : "192.168.1.1"}' \
  http://localhost:8080/sdnips/bgp/create | python -m json.tool
```

3) O próximo passo é configurar as redes que serão anunciadas na sessão BGP, a saber o prefixo de rede do AS 100 (192.168.100.0/24). Para isso execute o seguinte comando:

```
curl -s -X POST -d '{"prefix" : "192.168.100.0/24"}' \
  http://localhost:8080/sdnips/bgp/add_prefix | python -m json.tool
```

4) Por fim, deve-se adicionar o vizinho BGP com o qual deseja-se comunicar, ou seja, o roteador do AS 666. É necessário especificar, portanto, o endereço IP do roteador vizinho (192.168.1.2) e o número do AS remoto (666). Para isso execute o seguinte comando:

```
curl -s -X POST -d '{"remote_as" : 666, "address" : "192.168.1.2"}' \
  http://localhost:8080/sdnips/bgp/add_neighbor | python -m json.tool
```

5) É possível acompanhar nos logs do Ryu que a aplicação SDN-IPS está tentando estabelecer a sessão BGP com o peer remoto, porém sem sucesso (de fato essa sessão será configurada na etapa seguinte). Para visualizar esse comportamento, volte ao console em que a aplicação do Ryu foi iniciada (passo 1) e observe as mensagens de "*Will try to reconnect to 192.168.1.2 after 30 secs*":

```
(460) accepted ('127.0.0.1', 51248)
API method core.start called with args: {'router_id': '192.168.1.1', 'label_range': (100, 100000), 'waiter': <ryu.lib.hub.Eve
: 179, 'local_as': 100, 'allow_local_as_in_count': 0, 'refresh_stalepath_time': 0, 'cluster_id': None, 'local_pref': 100, 're
127.0.0.1 - - [30/Nov/2017 15:54:32] "POST /sdnips/bgp/create HTTP/1.1" 200 119 0.029547
(460) accepted ('127.0.0.1', 51249)
API method network.add called with args: {'prefix': '192.168.100.0/24'}
127.0.0.1 - - [30/Nov/2017 15:54:32] "POST /sdnips/bgp/add_prefix HTTP/1.1" 200 119 0.003864
(460) accepted ('127.0.0.1', 51250)
API method neighbor.create called with args: {'connect_mode': 'both', 'cap_mbgp_evpn': False, 'remote_as': 666, 'cap_mbgp_vpn
r_octet_as_number': True, 'cap_mbgp_ipv6': False, 'is_next_hop_self': False, 'cap_mbgp_ipv4': True, 'cap_mbgp_ipv4fs': False,
ipv6fs': False, 'is_route_server_client': False, 'cap_enhanced_refresh': False, 'peer_next_hop': None, 'password': None, 'ip_
lse, 'cap_mbgp_vpnv4': False, 'cap_mbgp_vpnv6fs': False}
127.0.0.1 - - [30/Nov/2017 15:54:32] "POST /sdnips/bgp/add_neighbor HTTP/1.1" 200 119 0.003354
Will try to reconnect to 192.168.1.2 after 30 secs: True
Will try to reconnect to 192.168.1.2 after 30 secs: True
```

3.4.3 Configuração a sessão BGP do AS 666 com o Quagga

O Quagga é uma plataforma de roteamento em software bastante utilizada que provê implementação de diversos protocolos como OSPF, RIP e BGP na plataforma Unix (Linux, BSD, Solaris, etc). A arquitetura do Quagga consiste em diversos daemons, em particular:

zebra, o daemon principal do Quagga que atua como uma camada de abstração com a pilha de rede do Kernel Linux, permitindo por exemplo instalação das rotas na FIB; bgpd, que é o daemon utilizado para estabelecer as sessões BGP.

Neste laboratório o Quagga será executado na máquina Atacante, portanto deve-se acessar essa máquina via SSH para executar os comandos abaixo.

1) Na máquina Atacante, o primeiro passo é instalar o Quagga. Para isso, na máquina Atacante execute o seguinte comando:

```
su
apt-get install quagga telnet tcpdump
```

2) Em seguida deve-se criar a configuração mínima para funcionamento do daemon zebra. Para isso, modifique o arquivo /etc/quagga/zebra.conf com o seguinte comando:

```
cat >/etc/quagga/zebra.conf <<EOF
hostname as666
password zebra
log file /var/log/quagga/zebra.log
ip forwarding
ipv6 forwarding
line vty
EOF
```

3) Deve-se criar, também, a configuração mínima para funcionamento do daemon bgpd. Para isso, modifique o arquivo /etc/quagga/bgpd.conf com o seguinte comando:

```
cat >/etc/quagga/bgpd.conf <<EOF
hostname as666
password zebra
log file /var/log/quagga/bgpd.log
line vty
EOF
```

4) Em seguida deve-se habilitar os daemons do quagga que serão executados (zebra e bgpd). Para isso, modifique o arquivo /etc/quagga/daemons deixando-o da seguinte maneira:

```
zebra=yes
bgpd=yes
```

5) Agora deve-se ajustar as permissões dos arquivos e iniciar o serviço Quagga:

```
chown quagga:quagga /etc/quagga/*
chmod 640 /etc/quagga/*
/etc/init.d/quagga restart
```

6) O próximo passo é fazer a configuração do roteador BGP do AS 666, através do console do quagga. Para acessar o console do bgpd no Quagga, execute o seguinte comando (ao ser questionado pela senha, informe a senha configurada anteriormente “zebra” - sem aspas):

```
telnet localhost 2605
```

7) Para fazer a configuração BGP do AS 666 é necessário configurar o Router-ID, número AS local, rede anunciada e vizinhos BGP. Deve-se fazer isso a partir do console do bgpd aberto no passo anterior. No console do bgpd execute os seguintes comandos:

```
enable
configure terminal
 router bgp 666
  bgp router-id 192.168.1.2
  network 192.168.66.0/24
  neighbor 192.168.1.1 remote-as 100
  neighbor 192.168.1.1 soft-reconfiguration inbound
  exit
 exit
write memory
```

A saída esperada é algo como ilustrado abaixo:

```
root@Atacante:/home/rnp/[redacted]# telnet localhost 2605
Trying 127.0.0.1...
Connected to localhost.
Escape character is '^]'.

Hello, this is Quagga (version 0.99.23.1).
Copyright 1996-2005 Kunihiro Ishiguro, et al.


User Access Verification

Password:
as666> enable
as666# configure terminal
as666(config)#  router bgp 666
as666(config-router)#  bgp router-id 192.168.1.2
as666(config-router)#  network 192.168.66.0/24
as666(config-router)#  neighbor 192.168.1.1 remote-as 100
as666(config-router)#  neighbor 192.168.1.1 soft-reconfiguration inbound
as666(config-router)#  exit
as666(config)#  exit
as666# write memory
Configuration saved to /etc/quagga/bgpd.conf
as666#
```

3.4.4 Testando a conectividade entre o AS 100 e AS 666

1) É possível visualizar nos logs da aplicação Ryu SDN-IPS que a sessão BGP foi estabelecida. Para isso, volte ao console do Controlador SDN e observe uma saída similar ao ilustrado abaixo:

```
Will try to reconnect to 192.168.1.2 after 30 secs: True
Connection to peer: 192.168.1.2 established
Peer up: 192.168.1.2 666
the best path changed: remote-as=666 prefix=192.168.66.0/24 next-hop=192.168.1.2 action=add
net.ipv4.ip_forward = 1
```

2) É possível verificar também as sessões BGP ativas no console do Quagga. Para isso, volte ao console do Atacante e execute os seguintes comandos:

```
show ip bgp summary
show ip bgp neighbors 192.168.1.1 received-routes
```

A saída esperada é algo como ilustrado abaixo:

```
as666# show ip bgp summary
BGP router identifier 192.168.1.2, local AS number 666
RIB entries 3, using 216 bytes of memory
Peers 1, using 2528 bytes of memory

Neighbor        V    AS MsgRcvd MsgSent   TblVer  InQ OutQ Up/Down  State/PfxRcd
192.168.1.1     4   100      20      23        0    0    0 00:04:15        1

Total number of neighbors 1
as666#
```

```
as666# sh ip bgp neighbors 192.168.1.1 received-routes
BGP table version is 0, local router ID is 192.168.1.2
Status codes: s suppressed, d damped, h history, * valid, > best, i - internal,
              r RIB-failure, S Stale, R Removed
Origin codes: i - IGP, e - EGP, ? - incomplete

   Network          Next Hop            Metric LocPrf Weight Path
*> 192.168.100.0    192.168.1.1                            0 100 i

Total number of prefixes 1
```

3) Por fim, a partir da máquina Cliente deve-se executar um teste de conectividade através de PING. Acesse o console da máquina cliente via SSH e execute os seguintes comandos:

```
ping -c 2 192.168.66.1
traceroute -n 192.168.66.1
```

A saída esperada é algo como ilustrado abaixo:

```
root@Cliente:~# ping -c 2 192.168.66.1
PING 192.168.66.1 (192.168.66.1) 56(84) bytes of data.
64 bytes from 192.168.66.1: icmp_seq=1 ttl=63 time=1.47 ms
64 bytes from 192.168.66.1: icmp_seq=2 ttl=63 time=0.840 ms

--- 192.168.66.1 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1001ms
rtt min/avg/max/mdev = 0.840/1.158/1.477/0.320 ms
root@Cliente:~# traceroute -n 192.168.66.1
traceroute to 192.168.66.1 (192.168.66.1), 30 hops max, 60 byte packets
 1  192.168.100.254  0.578 ms  0.556 ms  0.538 ms
 2  192.168.66.1  1.097 ms  1.084 ms  1.034 ms
root@Cliente:~#
```

Com isso finaliza-se o módulo a conectividade BGP entre os dois AS's de forma integrada ao SDN/Openflow. Na próxima seção serão expostos os principais aspectos relacionados ao uso de sistemas de detecção de intrusão e como ativá-lo de forma integrada ao ambiente SDN ora proposto.