CreateInternet.pl

What is This?

Wantegrity Inc carries out security and performance testing of network infrastructure for clients. Some of these tests require simulating large meshed networks. While it is possible to simulate large networks in VM's using various simulation tools, we find that real hardware is often required. Wantegrity has made one of their tools available for all to use. The script is named createInternet.pl . Its purpose is to quickly and easily create fully meshed matrices of BGP/OSPF networks/neigbors inside of real Cisco hardware. The matrix can be up to 10 x 10 in size. Various configuration options are available at the command line. This document contains basic information on how to set up the script.

For more information or assistance, you can contact Wantegrity at inquiry@wantegrity.com or call Mike at: 203-550-5502.

Prerequisites

You will need two (2) Cisco layer 3 switches that support VRF lite, BGP and OSPF. That usually means you need the switch enterprise image. We recommend the Cisco WS-C4958 because (as of 2018) they are relatively cheap on the second hand market and support most of the chassis based features where most large scale performance testing is needed. Note that the PERL script can be modified for use on two Cisco routers. Contact Wantegrity if you would like to have a script written to support other vendors or device models and platforms.

The two switches must be connected together via a trunk (ie: a tag switched link). And the trunk should allow all VLAN's or must be configured to allow all of the VLAN's that are defined in the script output.

Installation Instructions

1. You will need a LINUX machine with PERL installed.

2. On your LINUX machine, copy the script text from the section below into a file named createInternet.pl and set the file to read/execute.
3. At the command line of your LINUX machine, type ./createInternet.pl -h
4. Use the help instructions to create text output that can be used as configuration and back out items for creating BGP/OSPF fully meshed cloud networks.

The Script

The entire script is reproduced below. All of the text will need to be copied into a file (preferably named createInternet.pl) on your LINUX machine. And the file permissions should be set to executable.

```
#!/usr/bin/perl -w

################################################################################################################################

# createInternet.pl generates a BGP/OSPF cloud based upon two Cisco routers using VRF-LITE

# Author: Michael Hawkins, wantegrity inc.

# email : mhawkins@wantegrity.com

#

################################################################################################################################


################################################################################################################################

################################################################################################################################

# revision history - add revisions to top of list. entries should be added with version, initials, date and details

#

# v1.01c - MH - 04-25-2018 - fixed octet1&2 error in bgp neighbor item

# v1.01b - MH - 04-10-2018 - create full iBGP mesh with peer groups
```

```
# v1.01a - MH - 04-06-2018 - begin adding iBGP

# v1.01a - MH - 04-05-2018 - first basic working

# v0.01a - MH - 04-03-2018 - initial creation

##########################################################################################################################


##########################################################################################################################

##########################################################################################################################


use warnings;

use strict;


#use Data::Dumper;  # not needed after debugging


##########################################################################################################################

##########################################################################################################################


my $linehash = "##########################################################################################################################\n";


my %OPTIONS = (

          XSIZE => 3,

          YSIZE => 3,

          MESHNAME => "INTERNET",

          VLANBASE => 600,
```

```
        LOGLEVEL => 0,
        OCTET1 => 10,
        OCTET2 => 10,
        RDPREFIX => 10,
        RDSUFFIX => 10,
        OSPFPROCESS => 100
);


if ($#ARGV > -1) {
        for (@ARGV) {
                if($_ =~ /^-h$/i || $_ =~ /^--help$/i){
                        print $linehash;
                        print $linehash;
                        print "createInternet.pl generates configurations and backout text for creation of meshed network clouds.\n";
                        print $linehash;
                        print $linehash;
                        print "\n";
                        print $linehash;
                        print "WHAT YOU NEED\n";
                        print "You will need two Cisco layer 3 switches with at least one tagged (trunk) connection between them.\n";
                        print $linehash;
                        print "\n";
                        print "WHAT YOU GET\n";
```

```
print "The script output is split into two major sections each intended for one of the layer 3 switches.\n";
print "The configuration commands and the back out commands (to remove the configuration) are provided.\n";
print "The mesh networks that are created use VLAN's, VRF lite, OSPF and BGP. Each router hosts half of\n";
print "the VLAN's, VRF's, SVI's etc and the layer 3 connections run through the trunk between the partner switches.\n";
print "The OSPF and BGP neighbors are all fully meshed.\n";
print "\n";
print "WHAT CAN BE CHANGED\n";
print "The size, meshname, octets and RD prefix and suffixes can all be modified using command line options.\n";
print "This allows creation of multiple 'clouds'on the same pair of switches.\n";
print "Each cloud can also be joined together using some additional configuration effort.\n";
print "Examples:\n";
print "Create a 5x5 mesh: ./createInternet.pl XSIZE=5 YSIZE=5 MESHNAME=CORE > CORE-matrix-5x5.cfg\n";
print "Create a 3x4 mesh: ./createInternet.pl XSIZE=3 YSIZE=4 MESHNAME=ISP1 > ISP1-matrix-3x4.cfg\n";
print "\n";
print "YOU NEED TO BE CAUTIOUS\n";
print "This script has minimal error checking. Check the script output for correctness before applying\n";
print "the configuration to a device. This script is provided on an 'as-is' basis with no guarantee of\n";
print "correctness, suitability or usefulness whatsoever. USE OF THIS SCRIPT IS ENTIRELY AT YOUR OWN RISK.\n";
print "\n";
print $linehash;
print "Copyright Wantegrity Inc 2018. Contact: inquiry\@wantegrity.com https://www.wantegrity.com\n";
print $linehash;
```

```
                exit;
        }
        if ($_ =~ /=/) {
                my ($OPTION,$OPTVAL) = split /=/;
                if (exists ($OPTIONS{uc($OPTION)})) {
                        $OPTIONS{uc($OPTION)} = $OPTVAL;
                } else {
                        print "Invalid argument $OPTION.  Valid arguments are:\n";
                        print "XSIZE=<nodes (2-10)\n";
                        print "YSIZE=<nodes (2-10)>\n";
                        print "MESHNAME=<name of mesh>\n";
                        print "VLANBASE=<100-999>, default=600\n";
                        print "OCTET1=<1-99>, default=10\n";
                        print "OCTET1=<1-99>, default=10\n";
                        print "RDPREFIX=<1-99>, default=10\n";
                        print "RDSUFFIX=<1-99>, default=10\n";
                        print "OSPFPROCESS=<1-999>, default=100\n";
                        print "LOGLEVEL=[0-2], 0=off, 1=info, 2=debug\n";
                        exit;
                }
        } else {
                print "XSIZE=<nodes (2-10)>\n";
                print "YSIZE=<nodes (2-10)>\n";
```

```
                        print "MESHNAME=<name of mesh>\n";
                        print "VLANBASE=<100-999>, default=600\n";
                        print "OCTET1=<1-99>, default=10\n";
                        print "OCTET1=<1-99>, default=10\n";
                        print "RDPREFIX=<1-99>, default=10\n";
                        print "RDSUFFIX=<1-99>, default=10\n";
                        print "OSPFPROCESS=<1-999>, default=100\n";
                        print "LOGLEVEL=[0-2], 0=off, 1=info, 2=debug\n";
                        exit;
                }
        }
}

my @configA;        # layer 3 switch A configuration output
my @configB;        # layer 3 switch B configuration output
my @configAll;      # combined (for debugging)

my @backoutA;       # layer 3 switch A backout configuration output
my @backoutB;       # layer 3 switch B backout configuration output
my @backoutAll;     # combined (for debugging)

my $XSIZE = $OPTIONS{XSIZE};
my $YSIZE = $OPTIONS{YSIZE};
```

```
my $MESHNAME = $OPTIONS{MESHNAME};


my $RDPREFIX=$OPTIONS{RDPREFIX};

my $RDSUFFIX=$OPTIONS{RDSUFFIX};

my $OCTET1=$OPTIONS{OCTET1};

my $OCTET2=$OPTIONS{OCTET2};


my $OSPFproc = $OPTIONS{OSPFPROCESS};


my $VLANBASE = $OPTIONS{VLANBASE};               # starting VLAN and incrementing up for each


my $LOGLEVEL = $OPTIONS{LOGLEVEL};


my $countA = 0;

my $countB = 0;


#my $date = `date +%m-%d-%y`;       # unix date

#my $date =~ tr/\//-/;

#chomp($date);

#my $year = `date +%Y`;             # unix year

#chomp($year);

#my $month = `date +%b`;            # unix abbreviated month (Jan Feb Mar etc)
```

```
#chomp($month);

#my $time = `date +%H:%M`;          # unix time

#my $time =~ tr/:/-/;

#chomp($time);

#my $now="$date : $time";


#########################################################################################################################################

#########################################################################################################################################


print '! ** Router Matrix Generator ** - by M Hawkins www.wantegrity.com'."\n";

# create detailed notes

print "Generated by script $0 with size: ";

print "XSIZE=$XSIZE YSIZE=$YSIZE\n";


my $vlan=$VLANBASE;

my $configstr;

my $backoutstr;

my $teststr;


for (my $y = 1; $y <=$YSIZE; $y++){

          for (my $x = 1; $x <=$XSIZE; $x++){


                    # each node requires its own VRF
```

```
my $vrf_name = "$MESHNAME\_$x.$y";

my $loopback_IP = "$OCTET1.$OCTET2.$x.$y";


$configstr .= 'ip vrf '.$vrf_name."\n";

$configstr .= " rd $RDPREFIX$x:$RDSUFFIX$y\n";

$configstr .= "!\n";


# an OSPF/BGP loopback for each node

$configstr .= "interface Loopback $OCTET1$OCTET2$x$y\n";

$configstr .= " ip vrf forwarding $vrf_name\n";

$configstr .= " ip address $loopback_IP 255.255.255.255\n";

$configstr .= " no shut\n";

$configstr .= "!\n";


$configstr .= 'router ospf '.$OSPFproc.' vrf '.$vrf_name."\n";

$configstr .= ' router-id '.$loopback_IP."\n";

$configstr .= ' area 0 range '.$OCTET1.'.'.$OCTET2.'.'.$x.'.'.$y.' 255.255.255.255'."\n";

$configstr .= ' network '.$OCTET1.'.'.$OCTET2.'.'.$x.'.'.$y.' 0.0.0.0 area 0'."\n";

$configstr .= ' passive-interface default'."\n";

$configstr .= ' no passive-interface Loopback'.$OCTET1.$OCTET2.$x.$y."\n";

$configstr .=  "!\n";


# horizontal links
```

```
if($x<$XSIZE){      # right hand side

          # define right horizontal VLAN

          $configstr .= 'vlan '.$vlan."\n";

          $configstr .= ' name '.$vrf_name."_RIGHT\n";

          $configstr .= "!\n";


          # define right horizontal vlan SVI interface

          $configstr .= 'interface vlan'.$vlan."\n";

          $configstr .= ' ip vrf forwarding '.$vrf_name."\n";

          $configstr .= ' ip address '.$OCTET1.'.'.$x.'.'.$y.".1 255.255.255.252\n";

          $configstr .= " no shut\n";

          $configstr .= "!\n";


          $configstr .= 'router ospf '.$OSPFproc.' vrf '.$vrf_name."\n";

          $configstr .= ' area 0 range '.$OCTET1.'.'.$x.'.'.$y.'.'.'0 255.255.255.252'."\n";

          $configstr .= ' network '.$OCTET1.'.'.$x.'.'.$y.'.'.'0 0.0.0.3 area 0'."\n";

          $configstr .= ' no passive-interface vlan'.$vlan."\n";

          $configstr .= "!\n";


          $backoutstr .= 'no router ospf '.$OSPFproc.' vrf '.$vrf_name."\n";

          $backoutstr .= 'no interface vlan'.$vlan."\n";

          $backoutstr .= 'no vlan '.$vlan."\n";
```

```
}
if($x>1){ # left hand side
          # define left horizontal VLAN
          $configstr .= 'vlan '.($vlan-2)."\n";
          $configstr .= ' name '.$vrf_name."_LEFT\n";
          $configstr .= "!\n";

          # define left horizontal vlan SVI interface
          $configstr .= 'interface vlan'.($vlan-2)."\n";
          $configstr .= ' ip vrf forwarding '.$vrf_name."\n";
          $configstr .= ' ip address '.$OCTET1.'.'.($x-1).'.'.$y.".2 255.255.255.252\n";
          $configstr .= " no shut\n";
          $configstr .= "!\n";

          $configstr .= 'router ospf '.$OSPFproc.' vrf '.$vrf_name."\n";
          $configstr .= ' area 0 range '.$OCTET1.'.'.($x-1).'.'.$y.'.'.'0 255.255.255.252'."\n";
          $configstr .= ' network '.$OCTET1.'.'.($x-1).'.'.$y.'.'.'0 0.0.0.3 area 0'."\n";
          $configstr .= ' no passive-interface vlan'.($vlan-2)."\n";
          $configstr .=  "!\n";

          $backoutstr .= 'no router ospf '.$OSPFproc.' vrf '.$vrf_name."\n";
          $backoutstr .= 'no interface vlan'.($vlan-2)."\n";
          $backoutstr .= 'no vlan '.($vlan-2)."\n";
```

```
}
# vertical links
if($y<$YSIZE){      # top
          # define top vertical VLAN
          $configstr .= 'vlan '.($vlan+1)."\n";
          $configstr .= ' name '.$vrf_name."_TOP\n";
          $configstr .= "!\n";

          # define top vertical vlan SVI interface
          $configstr .= 'interface vlan'.($vlan+1)."\n";
          $configstr .= ' ip vrf forwarding '.$vrf_name."\n";
          $configstr .= ' ip address '.$OCTET1.'.'.$x.'.'.$y.".5 255.255.255.252\n";
          $configstr .= " no shut\n";
          $configstr .= "!\n";

          $configstr .= 'router ospf '.$OSPFproc.' vrf '.$vrf_name."\n";
          $configstr .= ' area 0 range '.$OCTET1.'.'.$x.'.'.$y.'.'.'4 255.255.255.252'."\n";
          $configstr .= ' network '.$OCTET1.'.'.$x.'.'.$y.'.'.'4 0.0.0.3 area 0'."\n";
          $configstr .= ' no passive-interface vlan'.($vlan+1)."\n";
          $configstr .=  "!\n";

          $backoutstr .= 'no router ospf '.$OSPFproc.' vrf '.$vrf_name."\n";
```

```
		$backoutstr .= 'no interface vlan'.($vlan+1)."\n";
		$backoutstr .= 'no vlan '.($vlan+1)."\n";


	}
	if($y>1){ # bottom
		# define bottom vertical VLAN
		my $bottomvlan = $vlan-($XSIZE*2)+1;
		$configstr .= 'vlan '.$bottomvlan."\n";
		$configstr .= ' name '.$vrf_name."_BOT\n";
		$configstr .= "!\n";

		# define bottom vertical vlan SVI interface
		$configstr .= 'interface vlan'.$bottomvlan."\n";
		$configstr .= ' ip vrf forwarding '.$vrf_name."\n";
		$configstr .= ' ip address '.$OCTET1.'.'.$x.'.'.($y-1).".6 255.255.255.252\n";
		$configstr .= " no shut\n";
		$configstr .= "!\n";

		$configstr .= 'router ospf '.$OSPFproc.' vrf '.$vrf_name."\n";
		$configstr .= ' area 0 range '.$OCTET1.'.'.$x.'.'.($y-1).'.'.'4 255.255.255.252'."\n";
		$configstr .= ' network '.$OCTET1.'.'.$x.'.'.($y-1).'.'.'4 0.0.0.3 area 0'."\n";
		$configstr .= ' no passive-interface vlan'.$bottomvlan."\n";
		$configstr .=  "!\n";
```

```
                $backoutstr .= 'no router ospf '.$OSPFproc.' vrf '.$vrf_name."\n";
                $backoutstr .= 'no interface vlan'.$bottomvlan."\n";
                $backoutstr .= 'no vlan '.$bottomvlan."\n";
        }

        # a BGP loopback for each node
#       $configstr .= "interface Loopback 1$OCTET1$OCTET2$x$y\n";
#       $configstr .= " ip vrf forwarding $vrf_name\n";
#       $configstr .= " ip address 1$OCTET1.$OCTET2.$x.$y 255.255.255.255\n";
#       $configstr .= " no shut\n";
#       $configstr .= "!\n";

        $configstr .= 'router bgp 65535'."\n";
        $configstr .= " address-family ipv4 vrf $vrf_name\n";

        $configstr .= " bgp router-id $loopback_IP\n";
        $configstr .= " network $loopback_IP mask 255.255.255.255\n";

#       $configstr .= " bgp router-id $OCTET1.$OCTET2.$x.$y\n";
#       $configstr .= " network 1$OCTET1.$OCTET2.$x.$y mask 255.255.255.255\n";

        $configstr .= " neighbor $MESHNAME peer-group\n";
```

```
$configstr .= " neighbor $MESHNAME remote-as 65535\n";

$configstr .= " neighbor $MESHNAME update-source Loopback $OCTET1$OCTET2$x$y\n";

$configstr .= " neighbor $MESHNAME version 4\n";


for (my $bgpy = 1; $bgpy <=$YSIZE; $bgpy++){

          for (my $bgpx = 1; $bgpx <=$XSIZE; $bgpx++){

                    if($bgpx!=$x || $bgpy!=$y){

                              $configstr .= " neighbor $OCTET1.$OCTET2.$bgpx.$bgpy peer-group $MESHNAME\n";

                    }

          }

}

$configstr .= "!\n";


$backoutstr .= 'router bgp 65535'."\n";

$backoutstr .= "no address-family ipv4 vrf $vrf_name\n";


$configstr .= "!\n";

$configstr .= "!\n";


$backoutstr .= 'no router ospf '.$OSPFproc.' vrf '.$MESHNAME.'_'.$x.'.'.$y."\n";

$backoutstr .= 'no interface Loopback'.$OCTET1.$OCTET2.$x.$y."\n";

$backoutstr .= 'no ip vrf '.$MESHNAME.'_'.$x.'.'.$y."\n";
```

```
push @configAll, $configstr;
push @backoutAll, $backoutstr;

if(($x+$y) % 2){
          # config B
          push @configB, $configstr;
          push @backoutB, $backoutstr;
          $countB++;
}else{
          # config A
          push @configA, $configstr;
          push @backoutA, $backoutstr;
          $countA++;
}

$configstr = "";
$backoutstr = "";

$vlan=$vlan+2;

if($vlan>990){
          print "! ##############################################################################################\n";
          print "! Network too big! Vlans are exhausted (>990). Aborting\n";
```

```
                        print "! ##########################################################################################################\n";
                        exit (1);
                }

                $OSPFproc++;

        }
}

        print "! Finished\n";
#       foreach my $line (@configAll){
#               print $line;
#       }

        print "! ##########################################################################################################\n";
        print "! Configuration A with ".$countA." items\n";
        print "! ##########################################################################################################\n";
        foreach my $line (@configA){
                print $line;
        }
        print "!\n";

        print "! ##########################################################################################################\n";
```

```
print "! Backout A with ".$countA." items\n";
print "! ##########################################################################################################\n";
foreach my $line (@backoutA){
          print $line;
}
print "!\n";

print "! ##########################################################################################################\n";
print "! Configuration B with ".$countB." items\n";
print "! ##########################################################################################################\n";

foreach my $line (@configB){
          print $line;
}
print "!\n";

print "! ##########################################################################################################\n";
print "! Backout B\n";
print "! ##########################################################################################################\n";
foreach my $line (@backoutB){
          print $line;
}
print "!\n";
```

```
print "! ##############################################################################################################\n";

print "! Configuration A has ".$countA." nodes\n";

print "! Configuration B has ".$countB." nodes\n";

print "! ##############################################################################################################\n";

print "! ##############################################################################################################\n\n\n";


exit(1);
```

Where to Next?

For additional information regarding configuration automation tools, guides and methods details, contact mhawkins@wantegrity.com

https://www.wantegrity.com

<END>