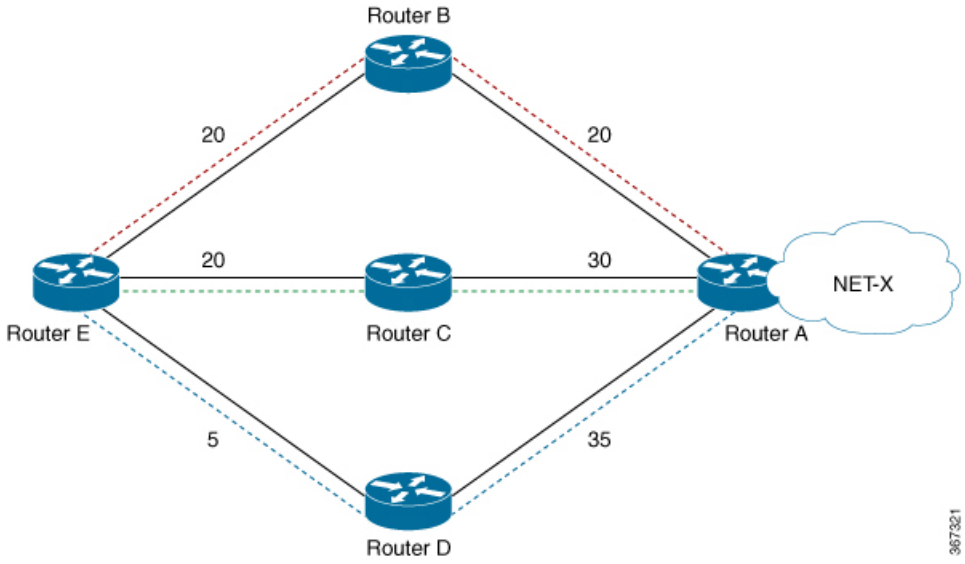
Implementing UCMP

The unequal cost multipath (UCMP) load-balancing provides the capability to load balance traffic proportionally across multiple paths, with different cost. Generally, higher bandwidth paths have lower Interior Gateway Protocol (IGP) metrics configured, so that they form the shortest IGP paths.

With the UCMP load-balancing enabled, protocols can use even lower bandwidth paths or higher cost paths for traffic, and can install these paths to the forwarding information base (FIB). These protocols still install multiple paths to the same destination in FIB, but each path will have a 'load metric/weight' associated with it. FIB uses this load metric/weight to decide the amount of traffic that needs to be sent on a higher bandwidth path and the amount of traffic that needs to be sent on a lower bandwidth path.

In the following example, there are 3 paths to get to Network X as follows:

Figure 1: Topology for UCMP



Paths	Cost from Router E to Net -X
E-B-A	40
E-C-A	50
E-D-A	40

IGP selects the lowest path links, i.e E-B-A and E-D-A. The path E-C-A is not considered for load balancing because of higher cost. The lowest path link E-D (5) is not a tie breaker, as the end to end cost to the Network X is considered.

ECMP vs. UCMP Load Balancing

Load balancing is a forwarding mechanism that distributes traffic over multiple links based on certain parameters. Equal Cost Multi Path (ECMP) is a forwarding mechanism for routing packets along multiple paths of equal cost with the goal to achieve almost equally distributed link load sharing. This significantly impacts a router's next-hop (path) decision.

In ECMP, it is assumed that all links available are of similar speed which inherently means that the hash values that are computed are equally shared over the multiple paths available.

For instance, if we have two paths available, the buckets (which in the end identify the links to be chosen) will be assigned in a 50% / 50% loadsharing. This can be problematic when one path is say a 10G link and the other link is a 1G link. In this case, you probably want to assign a (near) 90/10 type deviation, but considering that BGP is not bandwidth aware, the 10G path is still chosen 50% of the time as much as the 1G link. In this scenario, not all paths are of equal cost path.

What UCMP does in this case is apply a *weight* to a path which means that we are giving more hash buckets to one path that has a higher weight. The weight applied is *static* in the sense that it is derived by the DMZ bandwidth extended community either assigned to a peer or as configured via the Route Policy Language (RPL) route manipulation functionality.

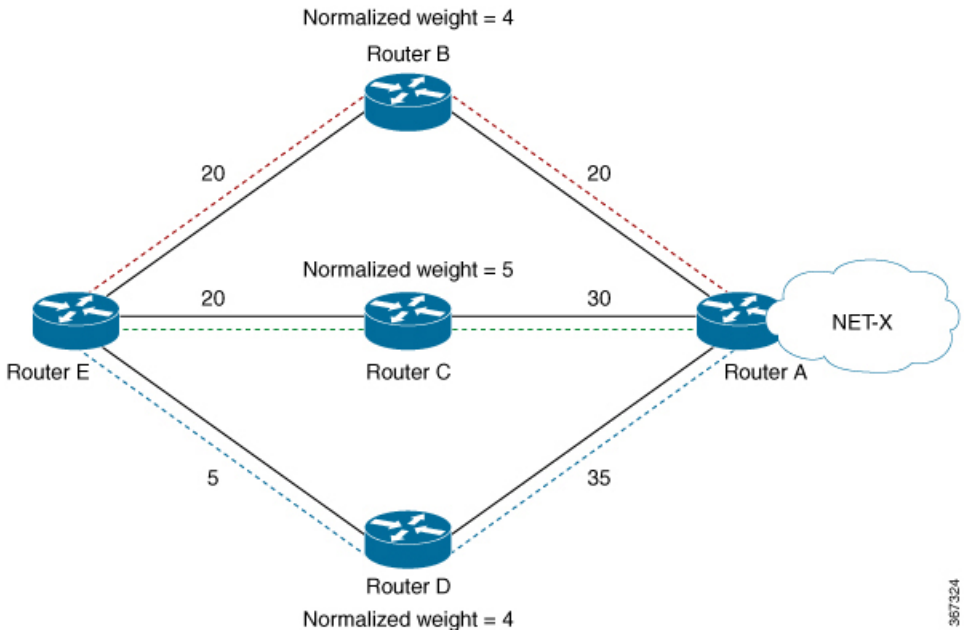
In general, a routing protocol decides a best path to a destination based on a metric. This metric is generally driven by the bandwidth of the circuit. When we have 3 paths available, say 1G/10G/100G, routing protocols generally discard the 1G/10G paths available. In defined cases, one may want to spread the load over the circuits based on the load they can carry. In this example, one may want to distribute traffic in a 1%/10%/89% fashion over the 1G/10G/100G paths available.

UCMP Minimum Integer Ratio

The UCMP Minimum Integer Ratio feature saves hardware resources when programming UCMP, by using optimized number of buckets.

To calculate the UCMP minimum integer ratio, find the greatest common divisor (GCD) and divide all the calculated normalized weights.

In the following Figure, we have three configured weights 40, 50, and 40, with GCD as 10. To calculate the normalized weight, divide the configured weight by GCD. In this example, we need to divide 40 by 10, 50 by 10, and 40 by 10, which is 4, 5, and 4 respectively. Therefore 4, 5, and 4 are the new normalized weights.



New normalized weights are: 40/10 = 4, 50/10 = 5, and 40/10 = 4

If GCD is 1, then Normalized Weight = (Path weight/Total weight) * Maximum bucket size

Configuring IS-IS With Weight

The following example shows the IS-IS weight configuration with IPv4. The same can be done for IPv6, with or without SR.

```
CPU0:router(config)# router isis 1
RP/0/RSP0/CPU0:router(config-isis)# interface HundredGigE 0/3/0/8
RP/0/RSP0/CPU0:router(config-isis-if)# address-family ipv4 unicast
RP/0/RSP0/CPU0:router(config-isis-if-af)# weight 200
RP/0/RSP0/CPU0:router(config-isis)# interface HundredGigE 0/3/0/9
RP/0/RSP0/CPU0:router(config-isis-if)# address-family ipv4 unicast
RP/0/RSP0/CPU0:router(config-isis-if-af)# weight 300
```

Verification

The following example verifies CEF entry. Then, for two paths with weights of 200 and 300 respectively, and GCD of 100; the expected normalized weights are 2 and 3.

```
Router# show cef ipv4 97.0.0.0 detail

97.0.0.0/24, version 537, internal 0x1000001 0x0 (ptr 0x71bcaee0) [1], 0x0 (0x71b98870),
0x0 (0x0)
 Updated Oct 16 06:34:46.197
 remote adjacency to HundredGigE 0/3/0/8
 Prefix Len 24, traffic index 0, precedence n/a, priority 2
  gateway array (0x71a6de10) reference count 13, flags 0x0, source rib (7), 0 backups
                [14 type 3 flags 0x8401 (0x71b02d90) ext 0x0 (0x0)]
  LW-LDI[type=3, refc=1, ptr=0x71b98870, sh-ldi=0x71b02d90]
  gateway array update type-time 1 Oct 16 06:34:46.196
 LDI Update time Oct 16 06:34:46.197
```

```
LW-LDI-TS Oct 16 06:34:46.197
  via 1.0.0.2/32, HundredGigE0/3/0/8, 4 dependencies, weight 200, class 0 [flags 0x0]
   path-idx 0 NHID 0x0 [0x7244d2a4 0x0]
   next hop 1.0.0.2/32
   remote adjacency
  via 2.0.0.2/32, HundredGigE0/3/0/9, 4 dependencies, weight 300, class 0 [flags 0x0]
   path-idx 1 NHID 0x0 [0x7244d2f8 0x0]
   next hop 2.0.0.2/32
   remote adjacency

   Weight distribution:
   slot 0, weight 200, normalized_weight 2, class 0
   slot 1, weight 300, normalized_weight 3, class 0

   Load distribution: 0 1 0 1 1 (refcount 14)

   Hash  OK  Interface                 Address
   0     Y   HundredGigE0/3/0/8    remote
   1     Y   HundredGigE0/3/0/9    remote
   2     Y   HundredGigE0/3/0/8    remote
   3     Y   HundredGigE0/3/0/9    remote
   4     Y   HundredGigE0/3/0/9    remote
```

Configuring IS-IS With Metric

The following example shows IS-IS metric configuration with IPv4. The same can be done with IPv6.

```
Router# enable
RP/0/RSP0/CPU0:router(config)# router isis 1
RP/0/RSP0/CPU0:router(config-isis)# interface HundredGigE0/3/0/8
RP/0/RSP0/CPU0:router(config-isis-if)# address-family ipv4 unicast
RP/0/RSP0/CPU0:router(config-isis-if-af)# metric 1
RP/0/RSP0/CPU0:router(config-isis)# interface HundredGigE0/3/0/9
RP/0/RSP0/CPU0:router(config-isis-if)# address-family ipv4 unicast
RP/0/RSP0/CPU0:router(config-isis-if-af)# metric 100
```

Verification

The following example verifies CEF entry, and checks for the two paths with metric values of 1 and 100, respectively. In this example, the best path route metric is 21 and the UCMP path route metric is 120. Therefore, the calculation is as follows:

The best path route metric, 21 = (1 configured + 20 added by IS-IS), weight 0xFFFFFFFF (4294967295)

The UCMP path route metric, 120 = (100 + 20), weight = (21/120) * 4294967295 = 751619276

GCD is one. So Normalized Weight is:

(4294967295 * 64) / (4294967295 + 751619276) = 54

(751619276 * 64) / (4294967295 + 751619276) = 9

```
Router# show cef ipv4 97.0.0.0 detail


97.0.0.0/24, version 773, internal 0x1000001 0x0 (ptr 0x71bcaee0) [1], 0x0 (0x71b98870),
0x0 (0x0)
 Updated Oct 16 06:36:08.632
 remote adjacency to HundredGigE0/3/0/8
 Prefix Len 24, traffic index 0, precedence n/a, priority 2
  gateway array (0x71a6d9d0) reference count 2, flags 0x0, source rib (7), 0 backups
```

```
               [3 type 3 flags 0x8401 (0x71b02b90) ext 0x0 (0x0)]
  LW-LDI [type=3, refc=1, ptr=0x71b98870, sh-ldi=0x71b02b90]
  gateway array update type-time 1 Oct 16 06:36:08.632
 LDI Update time Oct 16 06:36:08.632
 LW-LDI-TS Oct 16 06:36:08.632
   via 1.0.0.2/32, HundredGigE0/3/0/8, 14 dependencies, weight 4294967295, class 0 [flags
0x0]
    path-idx 0 NHID 0x0 [0x7244d2a4 0x0]
    next hop 1.0.0.2/32
    remote adjacency
   via 2.0.0.2/32, HundredGigE0/3/0/9, 14 dependencies, weight 751619276, class 0 [flags
0x0]
    path-idx 1 NHID 0x0 [0x7244d2f8 0x0]
    next hop 2.0.0.2/32
    remote adjacency


    Weight distribution:
    slot 0, weight 4294967295, normalized_weight 54, class 0
    slot 1, weight 751619276, normalized_weight 9, class 0
```

Configuring BGP With Weights

The following example shows BGP configuration with weights.

```
RP/0/RSP0/CPU0:router(config)# route-policy BW1
RP/0/RSP0/CPU0:router(config-rpl)# set extcommunity bandwidth (2906:45750000)
RP/0/RSP0/CPU0:router(config-rpl)# end-policy
RP/0/RSP0/CPU0:router(config)# !
RP/0/RSP0/CPU0:router(config)# route-policy BW2
RP/0/RSP0/CPU0:router(config-rpl)# set extcommunity bandwidth (2906:47250000)
RP/0/RSP0/CPU0:router(config-rpl)# end-policy
RP/0/RSP0/CPU0:router(config)# !
RP/0/RSP0/CPU0:router(config)# route-policy pass-all
RP/0/RSP0/CPU0:router(config-rpl)# pass
RP/0/RSP0/CPU0:router(config-rpl)# end-policy
RP/0/RSP0/CPU0:router(config)# !
RP/0/RSP0/CPU0:router(config)# router bgp 1
RP/0/RSP0/CPU0:router(config-bgp)# bgp bestpath as-path multipath-relax
RP/0/RSP0/CPU0:router(config-bgp)# address-family ipv4 unicast
RP/0/RSP0/CPU0:router(config-bgp-af)# maximum-paths eibgp 64
RP/0/RSP0/CPU0:router(config-bgp-af)# !
RP/0/RSP0/CPU0:router(config-bgp-af)# neighbor 1.0.0.2
RP/0/RSP0/CPU0:router(config-bgp-nbr)# remote-as 2
RP/0/RSP0/CPU0:router(config-bgp-nbr)# ebgp-multihop 255
RP/0/RSP0/CPU0:router(config-bgp-nbr)# dmz-link-bandwidth
RP/0/RSP0/CPU0:router(config-bgp-nbr)# address-family ipv4 unicast
RP/0/RSP0/CPU0:router(config-bgp-nbr-af)# multipath
RP/0/RSP0/CPU0:router(config-bgp-nbr-af)# route-policy BW1 in
RP/0/RSP0/CPU0:router(config-bgp-nbr-af)# route-policy pass-all out
RP/0/RSP0/CPU0:router(config-bgp-nbr-af)# !
RP/0/RSP0/CPU0:router(config-bgp-nbr-af)# neighbor 2.0.0.2
RP/0/RSP0/CPU0:router(config-bgp-nbr)# remote-as 2
RP/0/RSP0/CPU0:router(config-bgp-nbr)# ebgp-multihop 255
RP/0/RSP0/CPU0:router(config-bgp-nbr)# dmz-link-bandwidth
RP/0/RSP0/CPU0:router(config-bgp-nbr)# address-family ipv4 unicast
RP/0/RSP0/CPU0:router(config-bgp-nbr-af)# multipath
RP/0/RSP0/CPU0:router(config-bgp-nbr-af)# route-policy BW2 in
RP/0/RSP0/CPU0:router(config-bgp-nbr-af)# route-policy pass-all out
```

Verification

Step 1: Verify CEF entry:

Via 1.0.0.2: set extcommunity bandwidth (2906:45750000) – Weight = 45750000/125=366000 (125 ratio because baud)

Via 2.0.0.2: set extcommunity bandwidth (2906:47250000) – Weight = 47250000/125=378000

GCD is 6, so norm_weight = 61 and 63. Though 61 + 63 > 64.

Step 2: GCD of weights 61 and 63 is 1. Therefore, Normalised Weight = (Path weight/Total weight) * Maximum bucket size. The maximum bucket size value is 64. Total weight = 61+63 = 124.

norm_weight1 = (61/124) * 64 = 31, norm_weight2 = (63/124) * 64 = 32

You can verify the weight distribution in BGP, using the following command:

```
Router # show cef vrf default ipv4 97.0.0.0 detail


97.0.0.0/24, version 1965, internal 0x5000001 0x0 (ptr 0x71bcb620) [1], 0x0 (0x0), 0x0 (0x0)
 Updated Oct 16 08:15:02.958
 Prefix Len 24, traffic index 0, precedence n/a, priority 4
  gateway array (0x72a5e2f8) reference count 10, flags 0x2010, source rib (7), 0 backups
                [1 type 3 flags 0x48441 (0x71b02cd0) ext 0x0 (0x0)]
  LW-LDI[type=0, refc=0, ptr=0x0, sh-ldi=0x0]
  gateway array update type-time 1 Oct 16 08:15:02.958
 LDI Update time Oct 16 08:15:02.959


    Weight distribution:
    slot 0, weight 366000, normalized_weight 31
    slot 1, weight 378000, normalized_weight 32


  Level 1 - Load distribution: 0 1 0 1 0 1 0

1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1
0 1 0 1 0 1 0 1 0 1 1
  [0] via 1.0.0.2/32, recursive
  [1] via 2.0.0.2/32, recursive
```