WildCard Masking – Обратная (Инверсная) маска

Существующая маска для IP адреса выросла из классового деления адресов, на заре эпохи IP:

Класс A: 8 бит для номера сети 24 бита для номера хоста
Класс B: 16 бит на сеть и 16 бит на хост
Класс C: 24 бита на сеть 8 бит на хост

Когда стало слишком расточительным делить адреса подобным образом появилась маска, представляющая собой 32-х битное (из стольких же бит состоит и IP адрес) поле из подряд идущих единиц с начала поля, и после подряд идущих нулей. Единицы определяют те биты в IP адресе которые формируют номер сети, нули те биты в адресе которые формируют номер хоста.

IP адрес, десятичное: 10. 10. 0. 1
IP адрес, двоичное: 00001010.00001010.00000000.00000001
Маска, двоичное: 11111111.11111111.11111100.00000000
Маска, десятичное: 255. 255. 252. 0

Представление маски подобным образом, вполне, соотносится с термином битовой маски, т.е. единицы и нули определяют действия над конкретными битами в исходном числе, но плохо соотносится с форматом IP адреса – номер сети всегда определяется битами вначале, номер хоста битами в конце. Поэтому представление маски в виде 32-х битного поля является избыточным. Для однозначного определения маски можно определить только количество подряд идущих единиц с начала IP адреса от 0 до 32 – префиксное обозначение, обычно записывается через дробь после IP адреса: для примера выше 10.10.0.1/22 – 22 бита номер сети и 32-22=10 бит номер хоста. Если говорит про IPv6 адрес, то там определяется только префиксная запись маски/адреса – 2001:d8:a15e::1/48

Теперь представим маску таким образом, чтобы единицы определяли те биты в IP адресе которые формируют номер хоста,

а нули те биты которые формируют номер сети, в результате получаем инверсную маску:

```
IP адрес, десятичное: 10. 10. 0. 1/22
IP адрес, двоичное: 00001010.00001010.00000000.00000001
Маска, двоичное: 11111111.11111111.11111100.00000000
Инверсная маска, двоичное: 00000000.00000000.00000011.11111111
Маска, десятичное: 255. 255. 252. 0
Инверсная маска, десятичное: 0. 0. 3. 255
```

Это то что умеет делать любой, или почти любой, сетевой калькулятор, здесь не двоичная арифметика, точнее не арифметика, а базовые манипуляции по переводу чисел между системами счисления.

Получение инверсной маски из прямой, и обратное действие осуществляется инвертированием битового поля – замена 0 на 1, а 1 на 0. Если использовать десятичное представление то получение инверсной маски вычисляется следующим образом: от 255 отнимается число соответствующее значению октета в прямой маске, для примера выше:

```
  255.255.255.255
- 255.255.252. 0
=   0.  0.  3.255
```

Перевод из инверсной в прямую, также производится вычитанием из 255 значение октета соответствующего инверсной маске:

```
  255.255.255.255
-   0.  0.  3.255
= 255.255.252. 0
```

Используя термин инверсная маска, я умышленно исказил общепринятое название русскоязычного термина – обратная маска (invers mask, wildcard mask) – потому что обратная маска это гораздо более мощный механизм чем просто другое обозначение битов для нумерации сети и хоста в IP адресе. Обратная маска не обязана содержать подряд идущие единицы или нули, и единицы это не просто обозначение области хоста в IP адресе. Единицы это обозначение битов в IP адресе, которые могут меняться при

проверке условий, а нули фиксируют неизменные биты. То есть русскоязычный термин обратная маска больше соответствует «wildcard mask», нежели «invers mask», хотя в технической документации употребляется оба в одинаковых смыслах.

Область применения обратной маски это условные операции с IP адресами в cisco like интерфейсе и идеологии (не только cisco устройства). К этим областям относятся в частности списки доступа (ACL) – позволяет создавать не просто условия хост из этой сети, а гораздо более гибкие правила и определение сетей, а также в конфигурировании протоколов маршрутизации, OSPF например – позволяет создавать компактные правила для анонса не подряд идущих сетей.

Решим задачу запрета доступа с нечётных хостов из сети 192.168.0.0/24 куда либо. Если в нашем распоряжении есть только прямая маска, то нам надо составить правило на каждое из нечётных чисел в этой сети. От 192.168.0.1 до 192.168.0.253. IP 192.168.0.255 является широковещательным адресом и нам его обрабатывать не надо. Получаем (в нотации cisco):

```
access-list 101 deny ip 192.168.0.1 255.255.255.255 any
 (255.255.255.255 можно заменить на специально слово host, сделаем это)
access-list 101 deny ip host 192.168.0.1 any
access-list 101 deny ip host 192.168.0.3 any
access-list 101 deny ip host 192.168.0.5 any
access-list 101 deny ip host 192.168.0.7 any
…
access-list 101 deny ip host 192.168.0.93 any
…
access-list 101 deny ip host 192.168.0.253 any
access-list 101 permit ip any any
```

Итого: 128 строчек.

В конце мы разрешили доступ всему что мы не запретили. Теперь надо ассоциировать этот ACL с нужным интерфейсом, Fa0/1 например:

```
int fa0/1
ip access-group 101 in
```

Попробуем сократить полученный ACL, используя то свойство, что нечётное число в двоичном представлении всегда имеет значение 1 в нулевом разряде.

```
192.168.0.0  = 11000000.10101000.00000000.0000000|0 - чёт.
192.168.0.1  = 11000000.10101000.00000000.0000000|1 - не чёт.
192.168.0.2  = 11000000.10101000.00000000.0000001|0 - чёт
192.168.0.3  = 11000000.10101000.00000000.0000001|1 - не чёт
192.168.0.4  = 11000000.10101000.00000000.0000010|0 - чёт
192.168.0.5  = 11000000.10101000.00000000.0000010|1 - не чёт
192.168.0.6  = 11000000.10101000.00000000.0000011|0 - чёт
192.168.0.93 = 11000000.10101000.00000000.0101110|1 - не чёт
```

Используем обратную маску, напомню, что 1 – биты которые могут меняться, 0 которые не могут. Маска /24 = 255.255.255.0 в инверсной маске имеет 0.0.0.255 или 00000000.00000000.00000000.11111111. То есть последний октет может меняться во всех битах, но мы знаем что 0 бит надо оставить неизменным, поэтому изменим обратную маску следующим образом: 00000000.00000000.00000000.11111110 или 0.0.0.254. Заметим что если мы преобразуем эту запись в прямую маску, по правилам для инверсной маски то получится 255.255.255.1, что является неверной записью для маски, и с этим сетевой калькулятор уже не справится (или я их не видел).

Значит наша сеть 192.168.0.0/24 должна в последнем октете нулевом бите всегда иметь 1, то есть 192.168.0.1 с обратной маской 0.0.0.254.

```
Сеть 192.168.0.1, двоичное: 11000000.10101000.00000000.00000001
Обратная маска 0.0.0.254, двоичное: 00000000.00000000.00000000.11111110
```

Единицы в обратной маске это изменяемые биты в адресе – это биты в последнем октете со 1 по 7, с их использованием и нулевым битом всегда равным единице, мы можем составить любое нечётное число от 1 до 255.

Теперь напишем ACL, т.к. значение IP 192.168.0.255, имеет специальный смысл, разрешим его в первой строчке, иначе оно тоже будет запрещено:

```
access-list 101 permit ip host 192.168.0.255 any
access-list 101 deny ip 192.168.0.1 0.0.0.254 any
access-list 101 permit ip any any
```

Итого: 3 строчки вместо 128, можно сказать «Ого!».

Усложним задачу, нечётным должен быть предпоследний октет в подсетях 192.168.0.0/16. То есть для сетей 192.168.1.0/24, 192.168.3.0/24 и т.д. доступ надо закрыть. Изменим нашу обратную маску – инверсная маска для /16 = 0.0.255.255, предпоследний октет, мы по аналогии с первым примером сделаем 254, итого получим 0.0.254.255. И наши сети должны иметь нечётный предпослений октет 192.168.1.0 с обратной маской 0.0.254.255.

```
Сеть 192.168.1.0, двоичное: 11000000.10101000.00000001.00000000
Обратная маска 0.0.254.255, двоичное: 00000000.00000000.11111110.11111111
```

И наш ACL, предварительно разрешим IP 192.168.255.255 имеющий специальный смысл:

```
access-list 101 permit ip host 192.168.255.255 any
access-list 101 deny ip 192.168.1.0 0.0.254.255 any
access-list 101 permit ip any any
```

А теперь сделаем так чтобы в нечётных подсетях 192.168.0.0/16 в предпоследнем октете должны быть запрещены хосты (числа в последнем октете) с 1 по 63. Задачу с запрещением нечётных предпоследних октетов мы решили в примере выше, теперь посчитаем хосты с 1 по 63. 63 в двоичном представлении равно 111111, следовательно нам надо выбрать числа от 00000001 до 00111111. Предыдущая обратная маска 0.0.254.255 выбирала все числа в последнем октете, то есть могли меняться все 8 бит, нам надо менять всего лишь 6 бит, т.е числа от 0 до 63, итого

для последнего октета мы имеем запись 00111111 и вся маска 0.0.254.63.

Сеть 192.168.1.0, двоичное: 11000000.10101000.00000001.00000000
Обратная маска 0.0.254.63, двоичное: 00000000.00000000.11111110.00111111

Значение хоста 0, для нечётных предпоследних октетов, мы разрешим отдельным правилом ACL, т.к они по условию задачи не должны быть запрещены. Для этого случая обратная маска в последнем октете должна жёстко задать одно число, то есть ни один из битов не должен меняться, значит этот октет в обратной маске будет 0.

Сеть 192.168.1.0, двоичное: 11000000.10101000.00000001.00000000
Обратная маска 0.0.254.0, двоичное: 00000000.00000000.11111110.00000000

И наш ACL, правила permit со значением 255 в последнем октете у нас нет, т.к. в запрещающее правило это значение не попадает:

```
access-list 101 permit ip 192.168.1.0 0.0.254.0 any
access-list 101 deny ip 192.168.1.0 0.0.254.255 any
access-list 101 permit ip any any
```

В качестве закрепления материала, можно попробовать составить последний ACL с использованием только инверсной маски (т.е. где 0 и 1 идут строго подряд, в начале и соответственно в конце записи), и посчитать выгоду в количестве строчек.

Возможный вопрос, что делать если надо запретить чётные сети: в чётных числах нулевой бит всегда равен 0, соответственно вместо единички ставим 0. Для первого примера, всё тоже, но чётные хосты:

Сеть 192.168.0.0, двоичное: 11000000.10101000.00000000.00000000
Обратная маска 0.0.0.254, двоичное:

00000000.00000000.00000000.11111110

ACL короче, потому что IP 192.168.0.255, не попадает в запрещёные:

```
access-list 101 deny ip 192.168.0.1 0.0.0.254 any
access-list 101 permit ip any any
```

Очень много споров слышал по поводу: насколько часто данный механизм применяется на практике. За 4 года работы в отрасли применял только один раз, но этот раз позволил сократить ACL с несколько тысяч записей, если бы использовались стандартные маски (или просто инверсия к стандартной маске) до одной, собственно пример этого есть выше.