什么是 Hash?

文档版本	01
发布日期	2020-12-29

华为技术有限公司

商标声明


和其他华为商标均为华为技术有限公司的商标。
本文档提及的其他所有商标或注册商标，由各自的所有人拥有。

注意

您购买的产品、服务或特性等应受华为公司商业合同和条款的约束，本文档中描述的全部或部分产品、服务或特性可能不在您的购买或使用范围之内。除非合同另有约定，华为公司对本文档内容不做任何明示或默示的声明或保证。

由于产品版本升级或其他原因，本文档内容会不定期进行更新。除非另有约定，本文档仅作为使用指导，本文档中的所有陈述、信息和建议不构成任何明示或暗示的担保。

华为技术有限公司

地址： 深圳市龙岗区坂田华为总部办公楼 邮编：518129

网址： https://www.huawei.com

客户服务邮箱： support@huawei.com

客户服务电话： 4008302118

目 录

1 什么是 Hash?

Hash一般翻译做“散列”，就是把固定或任意长度的输入，通过散列算法变换成固定长度的输出，该输出就是散列值。这种转换是一种压缩映射，通常散列值的空间远小于输入的空间，不同的输入可能会散列成相同的输出，而且不可能从散列值来唯一确定输入值。简单而言Hash就是一种将固定或任意长度的消息压缩到某一固定长度的消息摘要的处理。

在数据通信领域，网络设备通常都支持Hash算法，它们使用Hash算法来确定如何对数据报文/流进行负载分担。通过Hash计算的结果会影响负载分担的效果，因此如何利用好Hash算法，在负载分担部署当中就显得尤为重要。

2 负载分担

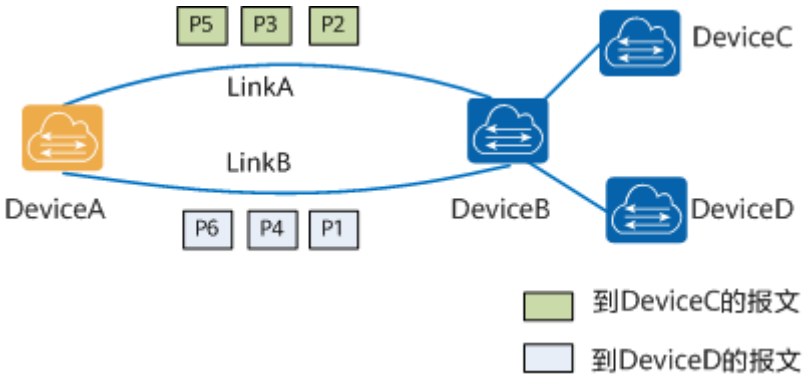
2.1 什么是负载分担?

在网络部署当中，当存在多条转发路径的时候，常常会部署负载分担功能。通过部署负载分担，可以基于报文内容等进行逐流转发，或者基于随机数、轮转方式进行逐包转发，以达到充分利用链路，提高转发效率的目的。

- 逐流负载分担：按照一定的规则，如根据五元组（源IP地址、目的IP地址、协议号、源端口号、目的端口号），将报文分成不同的流。同一条流的报文，经过Hash计算后，会在同一条链路上转发。

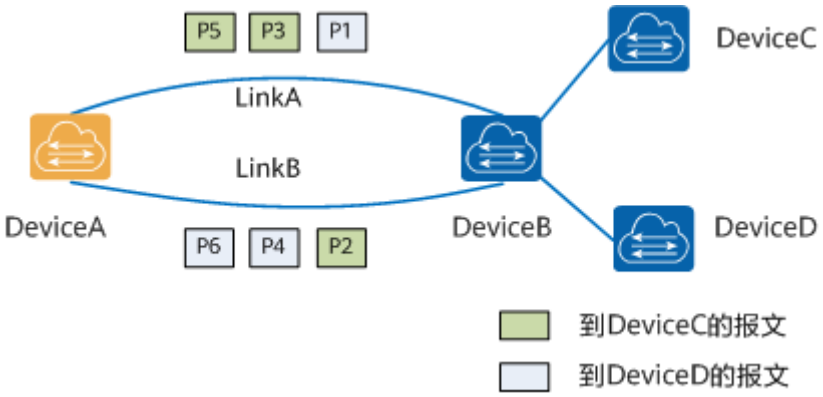
  如图2-1所示，假设DeviceA上有6个报文要通过DeviceA和DeviceB之间的LinkA和LinkB进行负载分担，其发送顺序为P1、P2、P3、P4、P5、P6，其中P2、P3和P5去往DeviceC，P1、P4、P6去往DeviceD。假设负载分担采用逐流方式，则去往DeviceC的报文都通过LinkA发送，去往DeviceD的报文都通过LinkB发送；或者去往DeviceC的报文都通过LinkB发送，去往DeviceD的报文都通过LinkA发送。

  图 2-1 逐流负载分担示意图



- 逐包负载分担：在转发时，按报文到来的次序，将报文均匀地分摊到参与负载的各条链路上，如图2-2所示。

图 2-2 逐包负载分担示意图



基于逐包负载分担方式的实际部署较少，因为该方式可能导致同一个用户的流量经过网络的不同路径传输后，到达目的服务器出现报文乱序或多份的情况。常见的负载分担方式一般选择基于逐流进行负载分担。

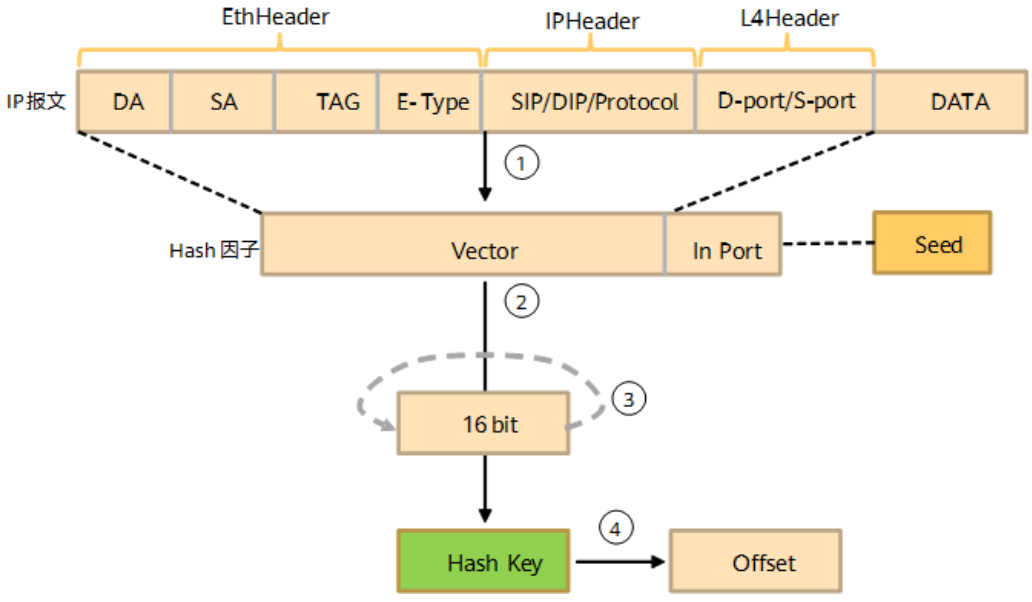
说明

本文中使用的命令和配置举例，部分可能只有某些款型才支持，本文中不再详细说明。了解具体的支持情况，请查询产品文档。

2.2 如何通过 Hash 算法实现负载分担?

常见的负载分担处理过程包含，输入（流量、报文的有效字段）、处理（通过Hash算法进行计算）和输出（根据计算结果将流量通过相应的出接口转发）。其中，通过Hash计算的结果会直接影响负载分担的效果，因此如何利用好Hash算法进行计算，在负载分担部署当中就显得尤为重要。

图 2-3 Hash 算法流程



如图2-3所示，Hash计算的流程如下：

1. 获取报文的信息。对于普通的IP报文，为源MAC、目的MAC、源IP、目的IP、VLAN、四层端口号等。按照配置的Hash模型，将这些信息作为Hash因子的参考值。可通过命令设置。
2. 根据Hash算法和Seed对Hash因子进行计算，得到Hash Key。

   其中，Hash算法是芯片提供固定种类的算法，不同的算法对于不同的流量模型计算的效果不同，有多种算法以供选择。可通过命令设置。

   另外，Seed是一个数值，用于参与计算。在相同Hash因子的情况下，Seed值会影响计算出的Hash Key的值。可通过命令设置。
3. 为了让同一个Hash因子衍生更多的变化，以更加灵活地适应不同的Hash场景，设备芯片可以对Hash Key的值进行0~15位的偏移。可通过命令设置。
4. 将Hash Key转化为Offset值，Offset值与出口个数进行取余运算，根据结果决定报文从设备的哪一个接口发出去。

3 负载分担的典型应用

ECMP 负载分担

等价多路径路由ECMP（Equal-Cost Multi-Path routing）实现了等价多路径负载均衡和链路备份的目的。ECMP应用于多条不同链路到达同一目的地址的网络环境中。ECMP的负载分担方式可以分为逐流负载分担或者逐包负载分担：

- 逐流负载分担能保证包的顺序，保证了同一数据流的帧在同一条下一跳路由转发，而不同数据流在不同的下一跳路由上转发；
- 逐包负载分担可以提高ECMP的带宽利用率，使等价多路径路由分担更均匀，但存在数据包乱序的问题，需要确保流量接收的设备或终端支持报文乱序组包的功能。

逐包负载分担方式中，又进一步分为以下两种模式：

- random（随机）模式：即报文的下一跳是在多条等价路由中随机产生的。对于已知单播，如果报文的IP地址和MAC地址没有变化，可以选择配置基于随机模式进行逐包负载分担方式使流量负载均衡。
- round-robin（轮转）模式：即到达同一目的地址的每条等价路由轮流进行流量的转发。对于已知单播，如果报文的长度比较接近，可以选择配置基于轮转方式进行逐包负载分担方式使流量负载均衡。

逐流负载分担方式中，针对不同类型的报文，负载分担方式以**表3-1**为例。例如，对于IPv4报文，默认情况下根据源IP、目的IP、目的端口号、源端口号进行负载分担，也可以通过命令行配置负载分担模式。

表 3-1 不同报文的负载分担方式

报文（以入接口为准）	默认负载分担模式	可配置的负载分担模式
IPv4报文	**src-ip**、**dst-ip**、**l4-src-port**、**l4-dst-port**	**src-ip**、**dst-ip**、**l4-src-port**、**l4-dst-port**、**protocol**、**vlan**、**src-interface**、**dscp**
IPv6报文	**src-ip**、**dst-ip**、**l4-src-port**、**l4-dst-port**	**flow-label**

报文（以入接口为准）	默认负载分担模式	可配置的负载分担模式
GRE报文	非IP报文基于内层的**inner-src-ip**、**inner-dst-ip**、**inner-l4-dport** 、**inner-l4-sport**；IP报文基于**src-ip**、**dst-ip**、**l4-src-port**和**l4-dst-port**。	**inner-src-ip**、**inner-dst-ip**、**inner-l4-dport** 、**inner-l4-sport**

Eth-Trunk 负载分担

以太网链路聚合Eth-Trunk简称链路聚合，它通过将多条以太网物理链路捆绑在一起成为一条逻辑链路，从而实现增加链路带宽的目的。同时，这些捆绑在一起的链路通过相互间的动态备份，可以有效地提高链路的可靠性。Eth-Trunk的负载分担方式可以分为逐包负载分担以及逐流负载分担。

- 逐包负载分担可以提高Eth-Trunk的带宽利用率，但存在数据包乱序的问题，需要确保流量接收的设备或终端支持报文乱序组包的功能。逐包负载分担方式中，又分为以下两种模式：
  - random（随机）模式：即报文的出接口是随机产生的，根据报文到达Eth-Trunk出接口的时间计算得出。对于已知单播，如果报文的IP地址和MAC地址没有变化，可以选择配置基于随机模式进行逐包负载分担方式使流量负载均衡。
  - round-robin（轮转）模式：即Eth-Trunk的各个成员接口轮流进行流量的转发。对于已知单播，如果报文的长度比较接近，可以选择配置基于轮转方式进行逐包负载分担方式使流量负载均衡。
- 逐流负载分担能保证包的顺序，保证了同一数据流的帧在同一条物理链路转发，而不同数据流在不同的物理链路上转发从而实现分担负载。逐流负载分担方式中，针对不同类型的报文，负载分担方式以表3-2为例。例如，对于IPv4报文，默认情况下根据源IP、目的IP、目的端口号、源端口号进行负载分担，也可以通过命令行配置负载分担模式。

表 3-2 不同报文的负载分担方式

报文（以入接口为准）	默认负载分担模式	可配置的负载分担模式	备注
IPv4报文	**src-ip**、**dst-ip**、**l4-src-port**、**l4-dst-port**	**src-ip**、**dst-ip**、**l4-src-port**、**l4-dst-port**、**protocol**	负载分担方式的配置，与报文类别有关，与报文的转发流程无关。 系统会识别以太帧携带的三层报文类别，比如报文被识别为IPv4，此时即使只做L2转发，依然会选择IPv4报文的对应负载分
IPv6报文	**src-ip**、**dst-ip**、**l4-src-port**、**l4-dst-port**	**src-ip**、**dst-ip**、**protocol**、**l4-src-port**、**l4-dst-port**	

报文（以入接口为准）	默认负载分担模式	可配置的负载分担模式	备注
非以上类别的L2报文	**src-mac**、**dst-mac**	**src-mac**、**dst-mac**、**src-interface**、**eth-type**	担模式进行负载分担。当报文无法识别为IPv4、IPv6报文时，系统才基于L2报文的负载分担模式（**src-mac**、**dst-mac**、**src-interface**、**eth-type**）进行负载分担。

4 常见的 Hash 功能

在确定了Hash算法配置之后，还有三种负载分担方式会对流量转发有影响，介绍如下。

弹性 Hash 功能

在Eth-Trunk中链路增加或减少时，尽量少的切换链路上的流量，只有部分流量进行链路切换。例如，有一个链路聚合组中包含3条成员链路，分别根据Hash-KEY值进行数据转发，其中一条链路故障无法转发数据时：

- 未配置弹性负载分担情况下另外两条链路会重新分配流量。
- 配置了弹性负载分担，另外两条链路上之前分配的流量不会发生变化，只是将故障链路上的流量大致均匀地分配到这两条链路上，这样对业务造成的影响较小。

当故障链路恢复后，会从这两条链路卸载一部分流量到故障恢复的这条链路上，各链路的流量分配和故障前流量分配也不会完全一致。

可通过命令**load-balance enhanced resilient profile** *profile-name*进行配置。

同源同宿 Hash 功能

属于同一个网络连接的双向数据报文必须从同一个输出接口输出。网络流量分析中，仅分析通信双方的单向流量是不够的，比如某台服务器要查看通信双方的信息时，往往需要结合通信双方的双向流量才能全面分析流量包含的信息，因此需要将通信双方的报文转发到同一台服务器上进行分析，这就要求设备在特定的转发流程中支持同源同宿。

在ECMP负载分担中，可通过命令**hashmode***hashmode-id*配置。在Eth-Trunk负载分担中，可通过命令**mode symmetry**进行配置。

内外层 Hash 功能

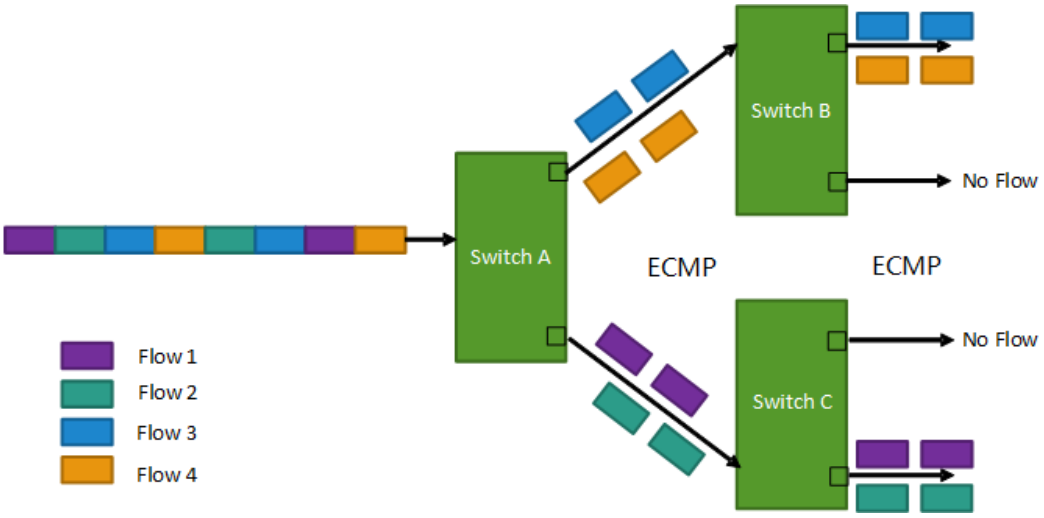
对于GRE、IPinIP等封装报文进行负载分担，封装的外层字段的信息是固定的，例如源地址和目的地址就是隧道的起始设备和终结设备的Loopback接口的IP地址，不发生变化，不能进行合理的负载分担；所以需要设备具备按内层字段（封装之前的原始报文）进行Hash的能力，保证Hash的均衡性。可以通过命令**tunnel** { **outer-header** | **inner-header** }进行配置。

5 如何解决 Hash 极化问题

Hash极化，也被称为Hash不均，是指流量经过2次或2次以上Hash后出现的负载分担不均匀的现象。常见于跨设备的多次Hash场景，即第一级进行ECMP Hash，第二级再进行ECMP Hash或者Eth-Trunk Hash。在同一设备上，若存在ECMP的出接口为多个Eth-Trunk也可能会出现Hash极化。

问题描述

图 5-1 Hash 极化示意图



如图5-1所示，Switch A的入接口有4种流量，出接口为2条等价链路，经Hash计算，流量3和流量4走上面的链路到Switch B；流量1和流量2走下面的链路到Switch C。在Switch B出接口同样为2条等价链路，若采用与Switch A相同或者类似的Hash算法，其Hash的结果将为流量3和流量4走上面的链路，而下面的链路没有流量。Switch C的情况类似。这种经过多次Hash后，ECMP或者Eth-Trunk各成员口之间流量极度不均匀的现象称为Hash极化。

实际上，交换机Hash功能的实现很大程度上取决于芯片，所以当使用同类型芯片的交换机位于网络中相邻的层级时，就可能会出现Hash极化问题。同一设备上若存在Eth-Tunk+ECMP的二级Hash，也存在该极化的风险。

因此，在多级网络中部署ECMP或者Eth-Trunk负载分担，需要考虑出现Hash极化问题的风险。

如何避免 Hash 极化问题

如果流量转发出现负载分担不均或者Hash极化现象，可以通过调整设备上的Hash算法来解决。例如，对于图5-1中的类似组网的跨设备的极化场景，可以在SwitchA与SwitchB上人为配置不同的Hash算法或偏移量。

了解了Hash算法，我们可以看出，影响Hash计算的结果有以下因素：

- Hash因子：可根据流量模型进行配置；
  相关设置命令示例：
  ip [**src-ip** | **dst-ip** | **l4-src-port** | **l4-dst-port** | **protocol**] *
  ipv6 [**src-ip** | **dst-ip** | **protocol** | **l4-src-port** | **l4-dst-port**] *
  l2 [**src-mac** | **dst-mac** | **vlan** | **eth-type**] *
- Hash算法：通过**hash-mode** *hash-mode-id*参数进行选择；
- Seed值：通过**seed** *seed-data*进行设置。当网络中存在多个厂商设备时，建议不同厂商配置为一致；
- 偏移量：通过**universal-id** *universal-id*参数进行设置。通常为一种Hash算法对应一种偏移量，当网络中存在多个厂商设备时，建议不同厂商配置为一致；
- Offset算法：芯片固定，无法修改。

当网络中出现Hash极化、负载分担不均的问题时，可以通过调整不同层级的网络设备的Hash计算参数来进行调整，使得Hash计算差异化，从而解决问题。

