细粒度时间负载均衡方法研究进度总结

一、背景介绍

通过对之前数据中心网络流量特征的调查研究，可以发现近年来，数据中心网络的性能得到了快速发展，传输性能大幅提升。然而，随着数据中心的不断改进，网络流量传输呈现出许多新的特性：

1. 数据包处理时延降低，交换机发送单个数据包的平均时延只有左右，而单个数据包穿越整个数据中心网络仅需；
2. 网络流量传输呈现突发式，即发送的流量由许多持续时间在的microburst组成，且不同microburst之间的时间间隔也为不等；
3. 现今商用数据中心网络中的大部分丢包都是由于microburst导致的交换机buffer溢出导致；
4. 商用数据中心网络交换机连接链路的利用率虽然不高，但是buffer占用率却一直保持较高的水平。

由于上述流量特征的存在，现有的绝大多数负载均衡方法都无法在做到有效均衡负载的同时，解决buffer溢出产生大量丢包的问题，一是因为现有大多数的均衡方法处理时延都在都在几十个毫秒以上，甚至一些中心控制的均衡方法处理时间达到了秒级别，所以面对持续时间只有的突发不能即使均衡其流量致使交换机buffer溢出；二是由于均衡方法的均衡粒度较粗，很多方法处理的是flow级别的流量，有的是较细的flowlet，但是对于microburst来说依然无法有效分散其流量，使得流量在交换机的某个发送端口出累积至buffer溢出。

二、方法设计

通过总结现有均衡方法的优势和不足，我们提出了两种改进方法，以实现达到较好的均衡效果的同时，也能对细粒度时间下突发式流量做出及时反应，减小丢包情况的发生。

1.基于改进的flowlet的方式

现有的基于flowlet的常见均衡方法有CONGA和LetFlow等，但是现有的这些都存在着一些局限。例如CONGA，依赖于piggyback的方式来获取网络状态信息本身就会有较大的时延，至少比RTT还要大，这对于上述级别传输的microburst来说太大了，而LetFlow依赖于flowlet的产生机制，即想要改变数据包的发送端口的条件是当前路径上发生了明显的拥塞情况，而我们想要减少microburst导致的丢包就必须在拥塞发生之前就做出处理，所以原生flowlet对于microburst无法有效应对。除此之外，CONGA和LetFlow均需要对网络中的Leaf Switch进行修改，使其能够支持flowlet的产生和发送，这样的方法部署起来并不方便。

我们的改进方法尽可能保留flowlet产生机制带来的网络状态自适应调节特性，同时，还要能够更加主动的应对microburst流量，还有，不对现有的数据中心网络做过多的修改以方便其部署。

首先，我们的方法主要在主机侧实现，在主机侧部署flowlet table控制flowlet的产生和发送，在网络侧依赖于现今绝大多数数据中心网络都在采用的ECMP方法。这样一来，就保证了不对现有网络做任何修改。

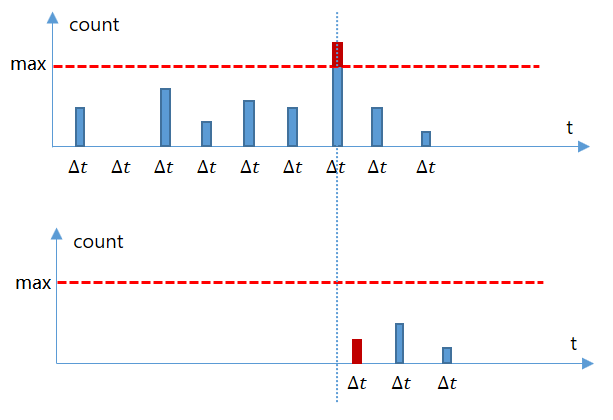
 其次，如同上述所言，flowlet的产生机制过于被动，依赖于当前路径的拥塞情况，所以我们对flowlet table的产生机制进行了改进，表项中原有的port number（该值在主机侧无意义）我们将其换为了counter，用于统计flowlet发送过程中每个内传输的所有数据包的总大小，设置counter的阈值max，如若单个内数据包尺寸之和不大于max，则如同原flowlet做相同处理；若在单个内数据包尺寸之和一旦超出max就立即结束该flowlet，重置该flowlet table表项，并且对新到达的packet作为新的flowlet的开始进行转发。该处理过程如图1所示：

Figure 1

发送出去的flowlet在数据中心网络中传输时都视作单独的一个flow（通过五元组定义），即主机侧的处理对上层网络隐藏。在对同一flowlet的各个packet发送的过程中，修改其五元组中的源TCP port number，在目的主机处接受该packet时将其源TCP port number进行还原。如此一来，同一flowlet的所有packet都会在网络中被认定为一个独立的flow并沿相同路径发送。

除了上述的改进之外，因为我们将flowlet的产生和发送部署在主机侧，其流量特征和交换机有较大差异，所以我们需要重新确定产生flowlet时规定的刷新间隔的大小，以保证既能够以较细粒度产生flowlet，又能有效减小目的主机处数据包乱序的发生。

2.基于packet的方式

由于packet是目前数据中心网络中我们可以操作的最小粒度，所以基于packet进行负载均衡方法可以做到理论上最佳的均衡效果。现有的基于packet的均衡方法有Drill和Hermes等。

Drill主要是部署在Leaf Switch上，且需要通过中心控制器为各个leaf switch下发子拓扑信息，这对其部署造成了较大的不便。其次，在对称性较差的网络中，由于每个flow是在某个子拓扑内传输，所以导致其对ECMP方法的提升非常局限。

Hermes方法部署在主机侧，它根据主动加piggyback两种方式获取路径状态信息后实时动态调整flow的转发路径，然而它调整flow转发路径的机制非常谨慎，尽管这可以减小路径调整对传输性能的影响，但是这却使得它对于microburst无法及时处理。

现有方法的这些局限给了我们设计的方向，于是我们有了新的基于packet的改进方法。

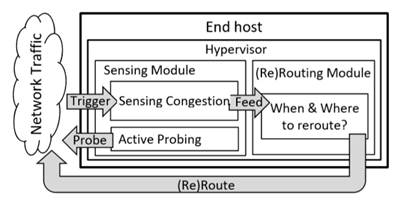
 首先，依然是部署在主机侧，使得该方法在数据中心网络的大规模部署更加可行。我们的方法主要为图2所示结构。

Figure 2

图2中，sensing module的sensing congestion部分是根据传输flow时对方回传的ACK报文来检测网络状态，而active probing部分是主机周期性发送探测报文来检测网络状态，根据检测到的状态后面的routing module来决定是否改变发送状态。

具体如何转发还要先由控制器来根据网络拓扑对每个源宿节点对之间的可用路径进行子拓扑划分，然后将划分的结果下发给各个相应的主机，并且每当拓扑发生变化时都会触发子拓扑划分机制，以实时更新每对源宿主机之间子拓扑划分情况。划分的依据参照Drill方法。在每个子拓扑内部各条可用路径都是对称的，数据包将会尽可能平均的在这些路径上发送。

新的基于packet的均衡方法具体过程如下所述：

首先，sensing module中的active周期性的检测各个子拓扑的状态，具体的通过发送probe packet的ECN和RTT来综合判断该子拓扑的传输质量，而每个rack只需要一台主机作为probe agent来与其他rack的probe agent相互发送probe packet就行，这样可以减小检测过程的网络资源开销。而在数据传输过程中sensing congestion也会根据回传的ack的ECN和RTT，实时更新当前传输的子拓扑的状态。

然后在有新的flow需要发送时会根据上述两种方法检测到的所有可用子拓扑的状态信息选择一条较好的进行发送，每个flow每次只能在一个子拓扑内传输。

在flow传输过程中，会根据其实时更新的子拓扑状态信息判断是否切换其他子拓扑进行发送，判断机制的设定较为谨慎，即防止频繁切换带来的性能损耗。

通过该改进方式可以实现，在网络较为对称时，所有可用路径划分的子拓扑数目较少，且每个子拓扑中的路径数较多，而在网络对称性较差时，子拓扑数目会变多，且每个子拓扑中的路径数会减小，最终均衡效果类似于Hermes。不过无论对称性如何，当遇到microburst突发流量时，子拓扑中多条可用路径会分散掉该突发带来的数据包，减小microburst导致丢包情况的发生。

三、性能验证

设计出以上两种方法后，下一步我们将着手通过仿真对比实验，对这些方法进行性能评估，以验证其是否能够在保证较好均衡效果的同时，减少microburst导致的丢包情况的发生。

我们的实验平台为ns3仿真平台，将通过模拟数据中心网络的拓扑，测试基于改进的flowlet的均衡方法和CONGA和LetFlow之间的性能差异，以及基于packet的方法和Drill和Hermes的差异，最终撰写实验报告验证我们的结论。