Bolt

张子豪 计算机学院

背景介绍

- 数据中心网络趋势：数据中心网络向高带宽、低延迟方向发展，线速不断提高，如 100Gbps 已普及，200Gbps 逐渐被采用，400Gbps 以太网也在标准化进程中。
- 拥塞控制挑战：随着线速增加，拥塞控制（CC）需要在更突发的工作负载下做出更高质量和及时性的决策。拥塞控制的关键在于获取精确的拥塞信号以及最小化控制回路延迟。

两个关键见解

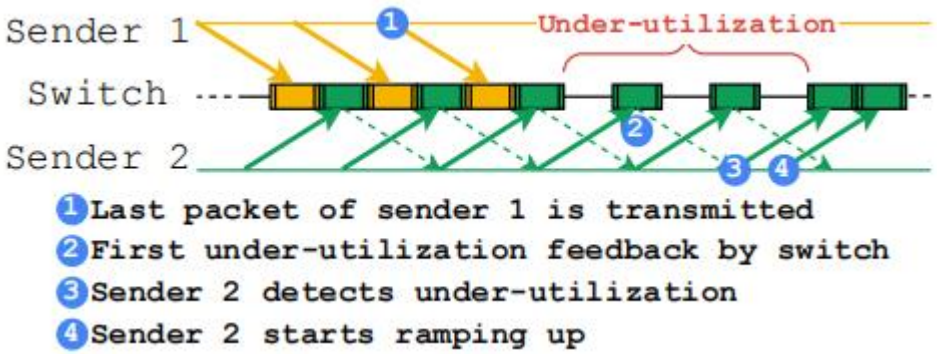
- granular feedback
- 需要细粒度的控制，高精度的CC算法
- control loop delay
- 减少control loop delay，因为现在DC的带宽太大了，即使时间长一点也会造成很大的拥塞

最小化Control Loop Delay

- control loop delay分为两个部分
- 1. feedback delay，反馈延迟，sender接收到任何一个feedback的时间
- observation period，

Feedback Delay

- 主要收集两种反馈信号，阻塞和利用率不足
- 1. 阻塞
- feedback delay越低越好，这样可以快速响应拥塞点
- 而且拥塞点的队列会加大RTT，所以尽量不排队就返回
- 2. 利用率不足，任何流都应该立刻知道是否有流完成，来提高利用率



Observation Period

- 两次调整cwnd的时间间隔是observation period
- 每 RTT 一次的决策可能会导致不稳定的振荡或相对较慢的收敛，这在高速网络中尤其成问题，因为在高速网络中流的到达和完成极其频繁。理想情况下，最短的观察周期应该是一个数据包的序列化时间，因为它是分组交换网络中数据包最细粒度的决策单元。然而，每个数据包的拥塞控制决策应该是增量式的，以应对在如此短的时间间隔内观察所产生的噪声。

Design

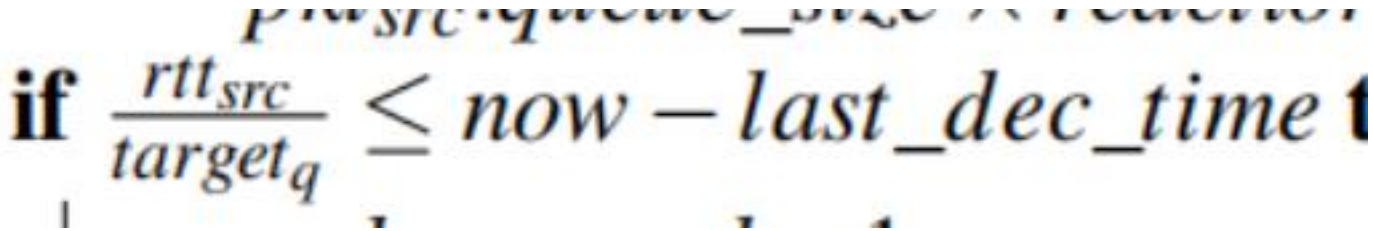
- 首先，通过在交换机上生成通知并直接将其反馈给发送方，将拥塞通知延迟降至最低
- 其次，发送方提前发出流完成事件的信号，以隐藏上升阶段的延迟并避免利用率不足
- 第三，在每次收到反馈后更新拥塞窗口（cwnd）以实现快速稳定，每次更新最多为每个数据包一次，以抵抗噪声

Bolt 通过 3 种主要机制来实现最小化反馈延迟和观察周期，同时为基于每个数据包的决策生成精确反馈：

1. SRC（亚 RTT 控制）将拥塞通知延迟降至绝对最小值。（§3.1）
2. PRU（主动上升）隐藏可预见的利用率不足事件的任何反馈延迟。（§3.2）
3. SM（供给匹配）从不可避免的利用率不足事件中快速恢复。（§3.3）

sub-RTT

- 拥塞点反馈+高优先级的src反馈包
- 防止过多src包，使用dec标志位
- Bolt 在 SRC 数据包上标记两个重要信息 —— 当前队列占用和链路容量。此外，它还反映原始数据包的发送时间戳
- sender:
- sender接收到src包时，会计算一个rttsrc，然后计算一个reaction_factor，拿流的速率/交换机的速率，计算该流在拥塞中的占比，乘上queue.size，算出该流对queue的占比



PRU - Proactive Ramp Up

- 监测流完成事件。
- 当流大于一个BDP时，才监测。在最后一个cwnd的所有包上打上last
- 当交换机收到带有last的包，
- 如果交换机无拥塞，那么在该流的egress队列中增加pru token的值，这个增量就是带有last数据包的个数。随后交换机会将token分发给不含有last的包，暗示这些流下一次发送可以增加发送速率；
- 但是有个例外，就是这些没有last的包，在之后的交换机中可能拥塞；BOLT的做法是再加一个NICflag，如果switch有PRU token且不拥塞，则保持NICflag+消耗token；反之则reset NICflag，阻止后边的switch再消耗token
- 流小于一个BDP，不使用PRU；
- 对于新流，即使流完成了，也不应该使用PRU；采用first标记(?)；

PRU- Proactive Ramp Up

- 注意到PRU采用基于RTT的反馈回路。那么PRU也是一个反馈信息，为什么不通过前面的sub-RTT减少反馈？
- 因为PRU是预测流的完成时间，如果提前反馈，可能导致流还没有完成就加速，这样反而会增加拥塞，而传统的RTT拥塞控制用在这里正好。

SM - Supply Matching

- 像链路和设备故障或路由改变这样的事件可能会导致链路利用率不足，且没有主动的信号提示。此外，如果将 PRU token分配给一个由于已经达到线速而无法增加速率，或者被下游交换机阻塞的流，那么这些token可能会被浪费。
- 传统CC：缓慢探测，AI；Bolt: 采用新的机制，MI
- 供：可发送的数据大小 需求：接收到的数据大小
- Bolt算法让每个switches监测自己端口的供需问题，以前的算法(例如HPCC)只会通过主机来判断利用率问题，二Bolt卸载到switch层面；

SM - Supply Matching

- 当令牌值超过一个最大传输单元（MTU）时，Bolt 会在数据包上保留 INC 标志，并允许发送者向网络中注入一个额外的数据包（算法 1 中的第 16 - 17 行）。然后供给令牌值会减去一个 MTU 以考虑所引发的未来需求。
- 如果一个交换机端口长时间没有收到数据包，供给令牌值可能会变得任意大，这在空闲一段时间后有数据包突发到达时会妨碍捕获瞬时利用率。为了解决这个问题，Bolt 将供给令牌值限制在最大为一个 MTU。

```
1 CalculateSupplyToken (pkt):
2     inter_arrival_time ← now − last_sm_time
3     last_sm_time ← now
4     supply ← BW × inter_arrival_time
5     demand ← pkt.size
6     sm_token ← sm_token + supply − demand
7     sm_token ← min(sm_token, MTU)
```

每个包都会计算对应的supply和demand

使用了一个lookup表

Implementation

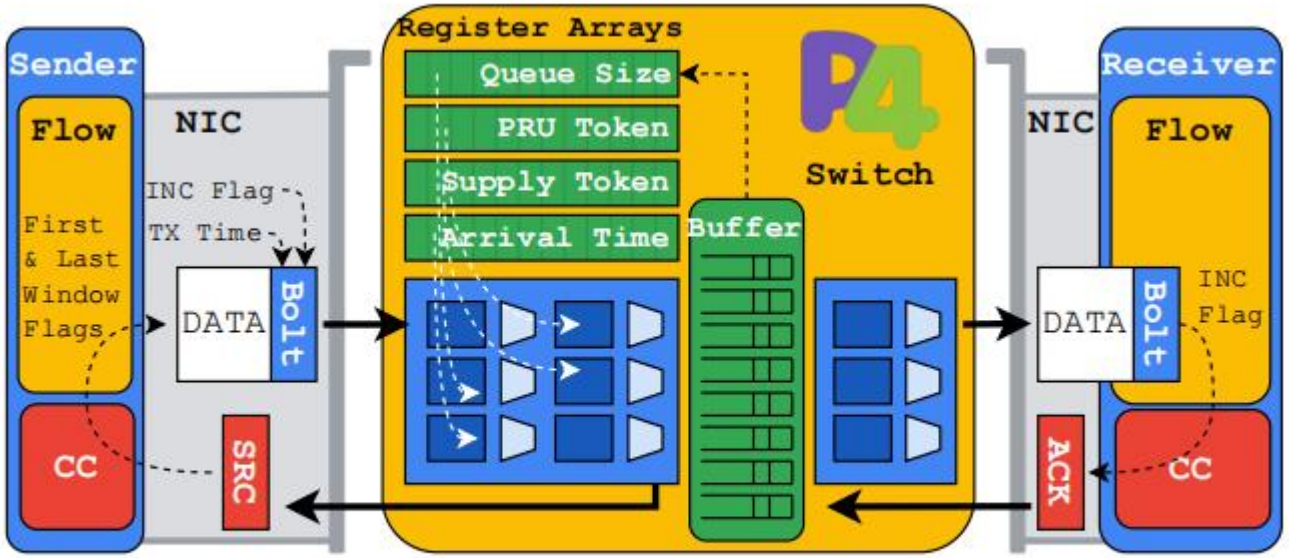


Figure 9: Bolt system overview

Host Prototype

- rttsrc：NIC发送数据时，在包中打时间戳，这是网卡硬件实现
- 传输层还复用针对同一服务器的RPC，以便在同一个网络连接上。新RPC的前cwnd字节并不一定被检测为连接的第一个窗口。为了缓解这个问题，我们的原型在连接空闲时重置已发送字节计数器，并在发送新的RPC时设置FIRST标志。
- PRU的最后一个窗口标记需要确定每个连接剩余数据的大小。在我们的原型中，连接通过应用程序的每次发送API调用来增加待发送字节计数器的大小。每次连接将数据包传输到网络时，计数器的值都会减少数据包的大小。因此，当该计数器的值小于cwnd时，会在数据包上设置LAST标志。

测试

真实部署

- 1，brownfield deployment问题。部分交换机可编程，部分不可编程。在Bolt之上再部署一个端到端的CC
- 2，主机也部署其他的算法，需要主机与老的CC并存。目前的实现:QoS隔离Bolt的队列
- 3，NIC会对流进行分批发送；即使cwnd<BDP, 如果此时数据包过多，仍触发SRC；解决方式是提高阈值；