中国科学技术大学
University of Science and Technology of China

# Congestion Control for Large-Scale RDMA Deployments

Yibo Zhu,Haggai Eran,Daniel Firestone,Chuanxiong Guo, et al.

SIGCOMM 15

**授课教师：赵功名**

**中国科大计算机学院**

**2025年秋·高级计算机网络**

# Outline


I. Introduction

II. The Need For DCQCN

III. The DCQCN Algorithm

IV. Buffer Settings

V. Analysis OF DCQCN

VI. Results

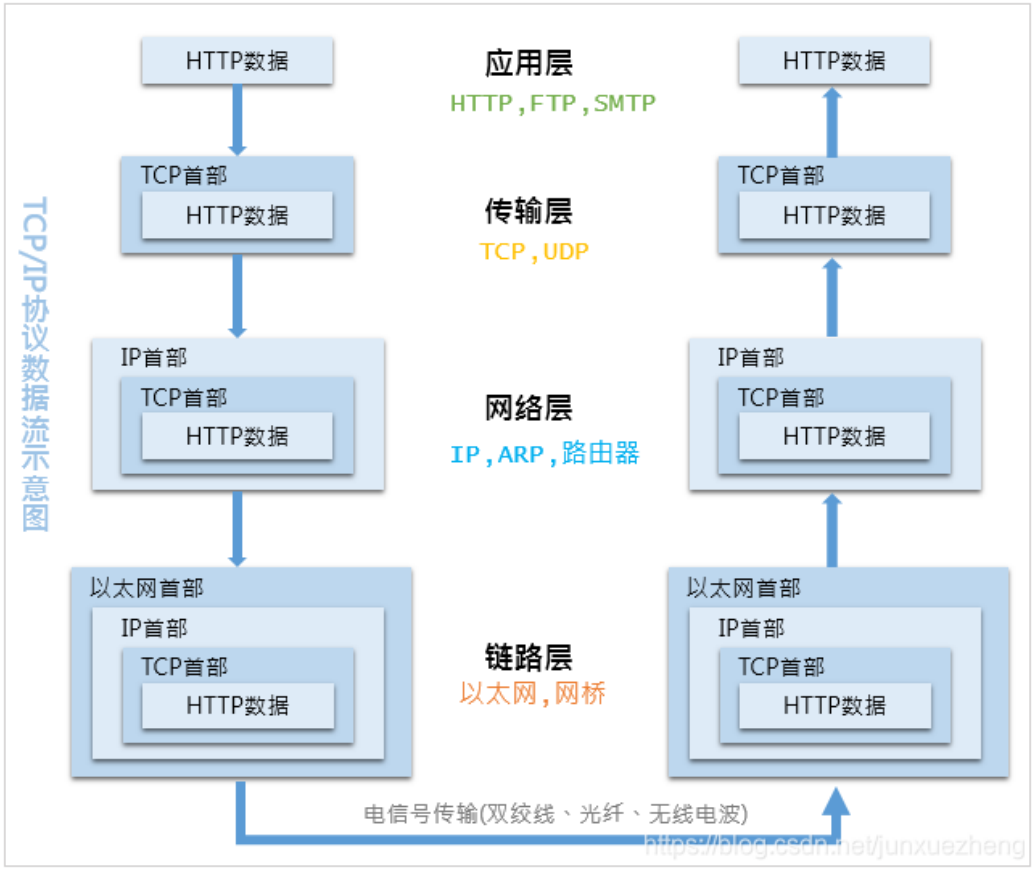
VII. Discussion

VIII. Review

# Introduction

**标准TCP/IP协议栈无法满足现代数据中心要求**

- 现代数据中心应用（如云存储）需要高带宽（40Gbps或更高）和超低延迟（每跳小于10微秒），且要求低CPU开销。标准的TCP/IP协议栈无法满足这些要求
- 远程直接内存访问（RDMA）满足要求。RDMA通过绕过主机网络栈，减少了CPU开销并降低了延迟



# Introduction

## RDMA

RDMA（Remote Direct Memory Access）是一种计算机网络协议，允许计算机在不经过操作系统和CPU的情况下，直接访问远程主机的内存

- **数据传输路径：**

普通网卡：需要经历用户空间→内核空间→网卡的三级拷贝流程，造成CPU资源浪费

RDMA网卡：通过零拷贝技术实现用户内存与网卡直连，实验室环境下可实现微秒级延迟（TCP/IP为毫秒级）

- **内核介入程度：**

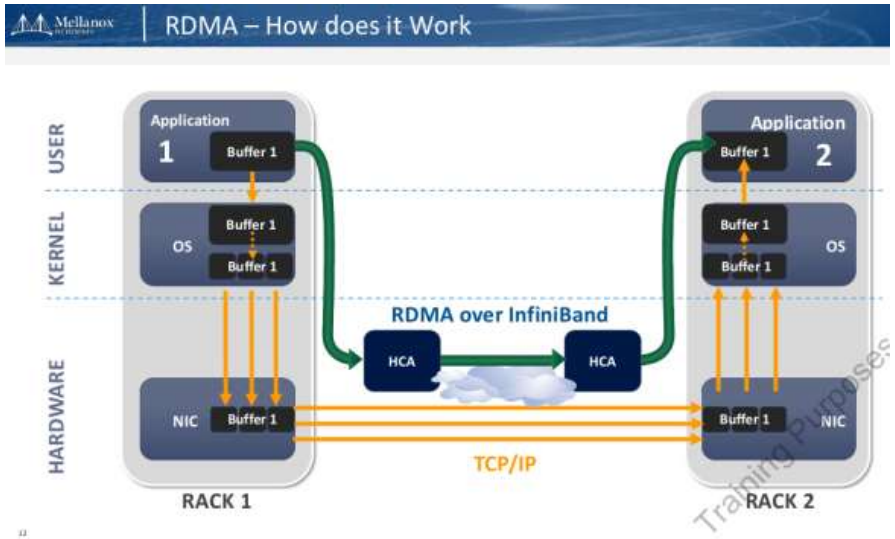
普通网卡依赖内核协议栈处理CRC校验、数据分段等操作

RDMA硬件卸载协议处理至网卡芯片，用户态驱动直接操作硬件

- **内存管理机制：**

普通网卡需通过DMA间接访问内存

RDMA支持单边操作模式，可通过注册物理地址直接读写远端内存



# Introduction

## RDMA核心技术架构

- **协议实现：**

InfiniBand：专为RDMA设计的网络架构，提供原生支持

RoCEv2：基于融合以太网的实现方案，兼容现有网络基础设施

iWARP：TCP/IP协议上的RDMA实现，牺牲部分性能换取兼容性

- **关键组件：**

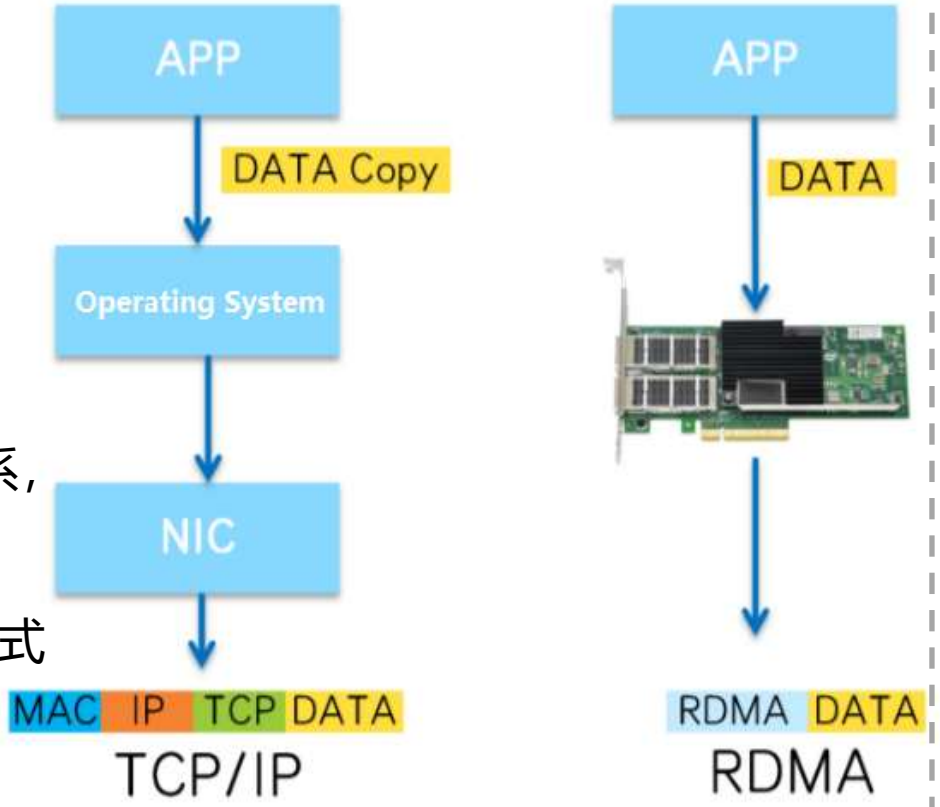
内存注册（Memory Region）：建立虚拟地址到物理地址的映射关系，保障内存访问安全性

队列对（Queue Pair）：发送队列和接收队列构成，支持异步通信模式

- **性能指标：**

100Gb网络环境下可实现亚微秒级延迟

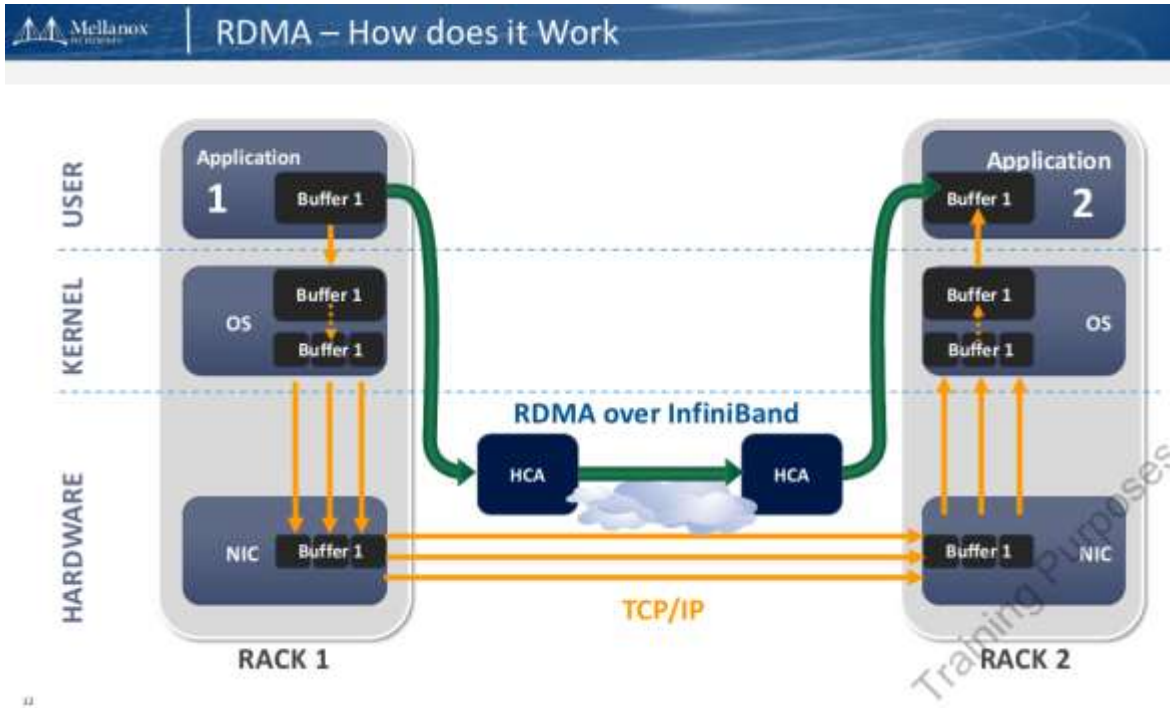
带宽利用率可达90%以上（传统TCP/IP协议栈通常低于60%）



# Introduction

## RDMA无法与以太网兼容

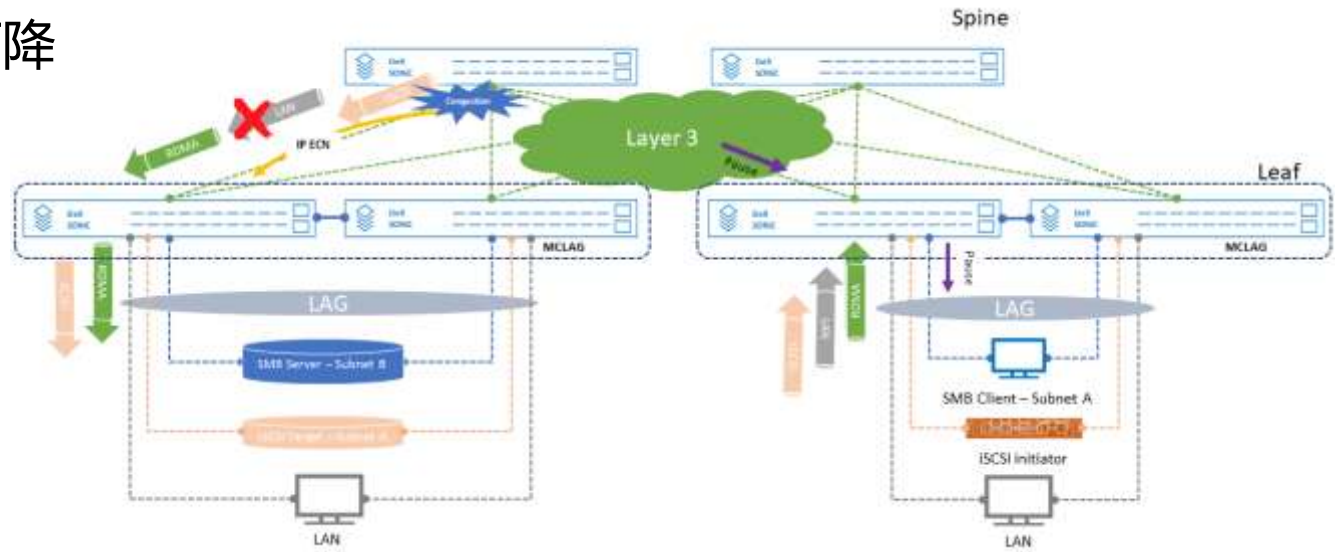
- RDMA技术通过InfiniBand（IB）网络使用。IB使用专用的网络栈，并且在L2层通过信用流量控制来防止因缓冲区溢出导致的数据丢包，从而保持数据传输的高效性
- InfiniBand协议栈无法与现代数据中心的IP和以太网技术兼容



# Introduction

## RoCE

- 为了在以太网和IP网络上实现RDMA，定义了RoCE和RoCEv2标准，RoCEv2在保持IB传输层的同时，用IP和UDP封装代替了IB的L3层，并用以太网代替了IB的L2层
- RoCEv2需要在无损的L2网络上高效运行。为此，RoCEv2使用了优先级流量控制（PFC）机制
- PFC通过强制上游实体暂停数据传输，避免缓冲区溢出，但由于PFC的粗粒度操作，无法区分不同的流，这可能会导致拥塞传播和性能下降



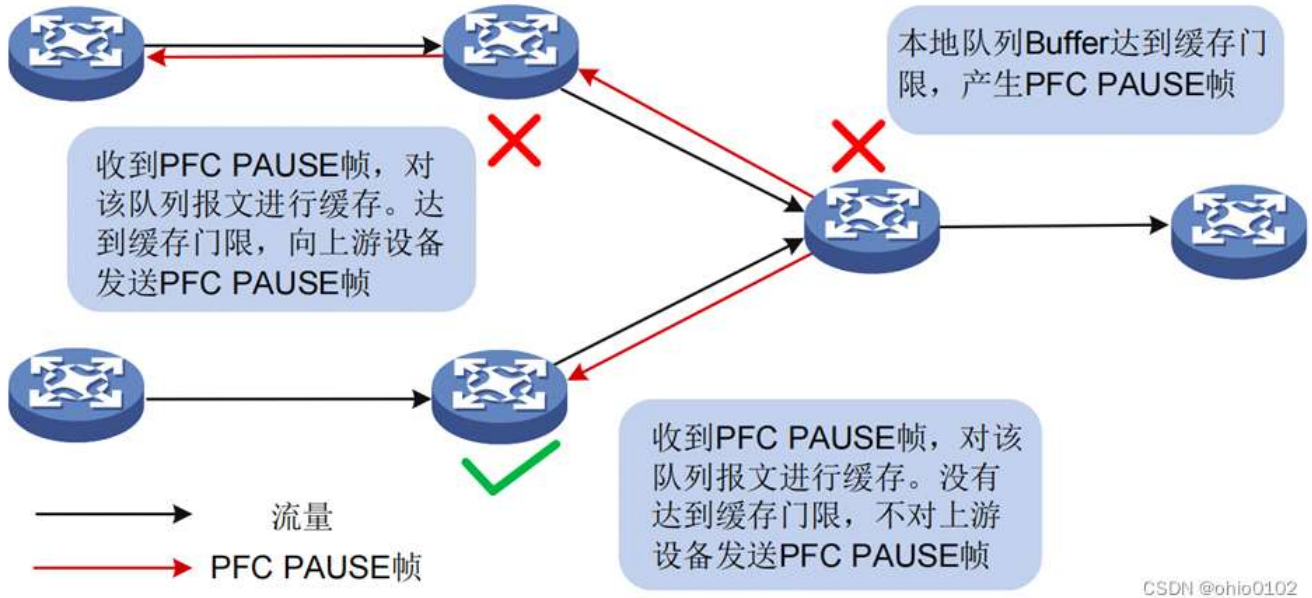
# Introduction

## PFC

基于优先级的流量控制（PFC）是一种增强型以太网流控机制，旨在为网络中的高优先级流量提供无丢包传输服务

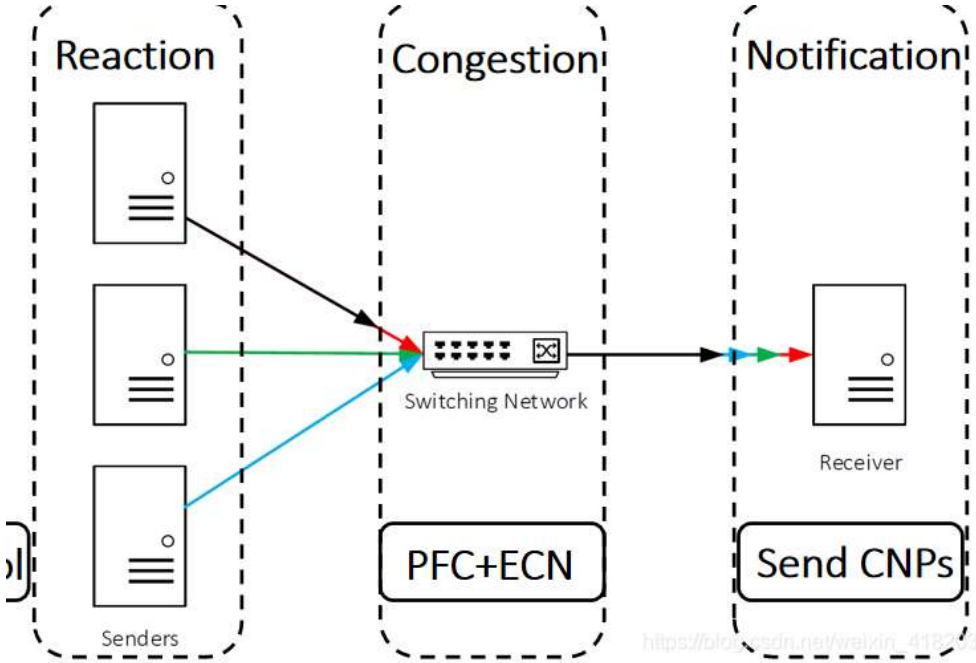
**工作流程**:

- 当下游设备的某个优先级队列缓存使用量超过预设的PFC门限值时，会向上游设备发送PFC反压帧（Pause帧），要求暂停该优先级流量的发送
- 上游设备收到反压帧后，会停止向该队列发送对应优先级的报文
- 当队列缓存降低到门限值以下时，下游设备会发送反压停止帧，通知上游恢复发送



# Introduction

- 目前的一些方案，如QCN、DCTCP和iWarp，无法满足所有这些要求：
  - QCN不能在L3网络上工作
  - DCTCP和iWarp有较慢的启动阶段，可能导致高峰存储负载下的性能不佳
- 为了满足这些需求，本文提出了Datacenter QCN（DCQCN）协议。DCQCN基于RoCEv2标准的拥塞控制组件，已经在Mellanox的NIC中实现，并且正在微软的数据中心中部署



# Outline

I. Introduction

II. The Need For DCQCN

III. The DCQCN Algorithm

IV. Buffer Settings

V. Analysis OF DCQCN

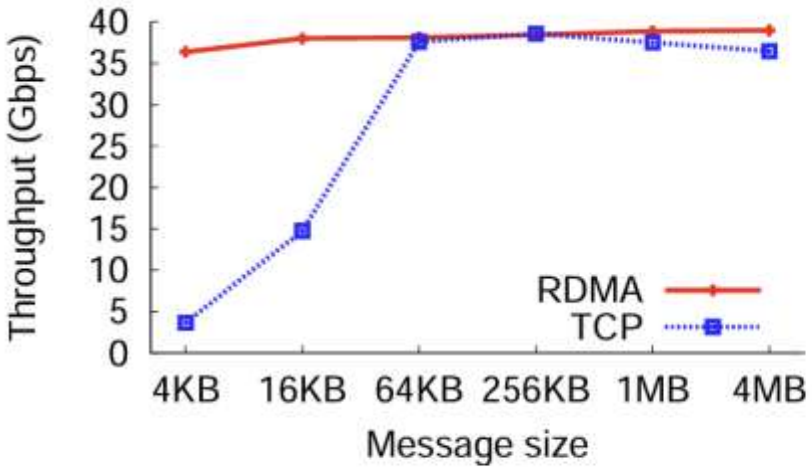
VI. Results

VII. Discussion

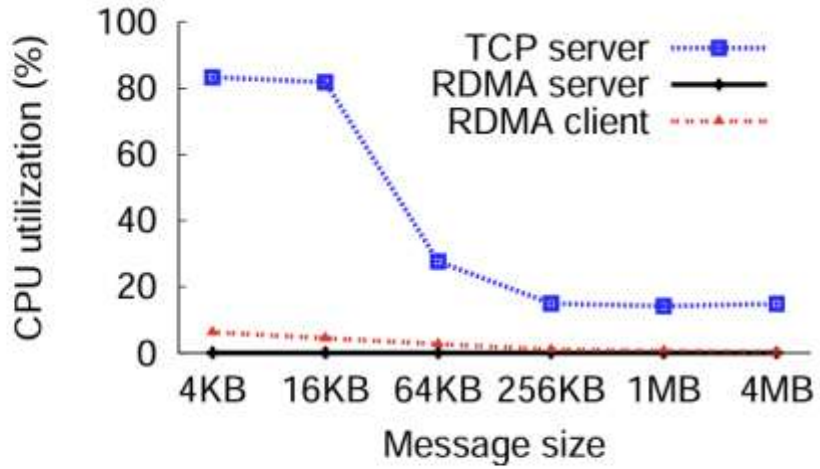
VIII. Review

# The Need For DCQCN

- 为了证明DCQCN的必要性。比较RoCEv2与传统TCP协议栈在吞吐量、CPU开销和延迟方面的表现。实验使用了两台机器（Intel Xeon E5-2660 2.2GHz，16核，128GB RAM，40Gbps网卡，Windows Server 2012R2），并通过Iperf测试TCP吞吐量，而使用定制工具测量RDMA吞吐量
- 实验结果显示，TCP的CPU开销很高，即使使用优化技术，TCP仍然消耗超过20%的CPU资源；而RDMA客户端的CPU利用率保持在3%以下，RDMA服务器几乎不消耗CPU资源



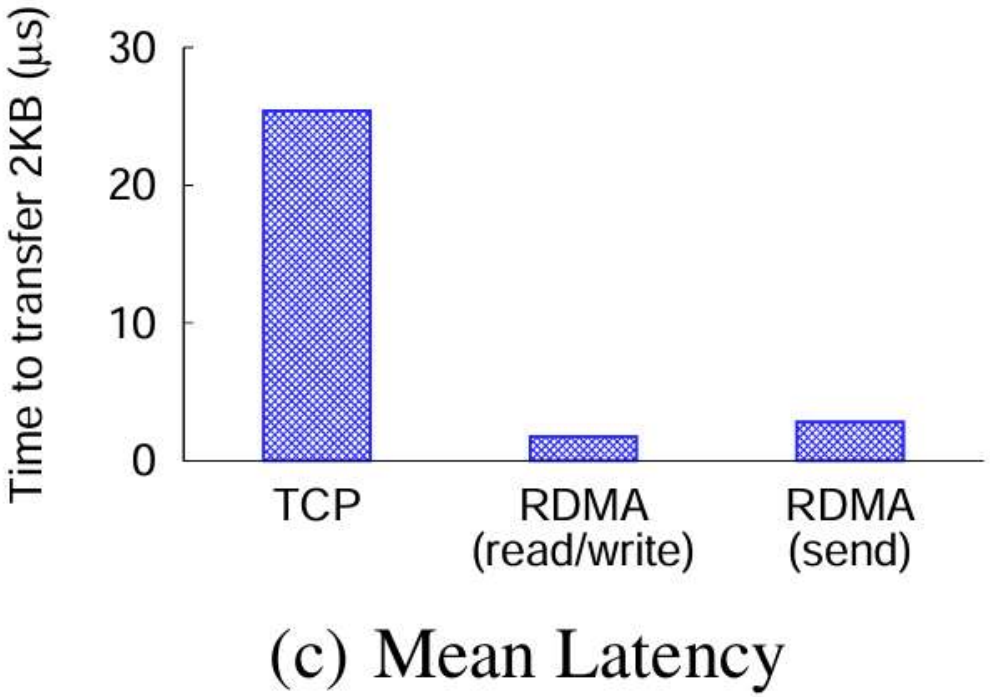

(a) Mean Throughput




(b) Mean CPU Utilization

# The Need For DCQCN

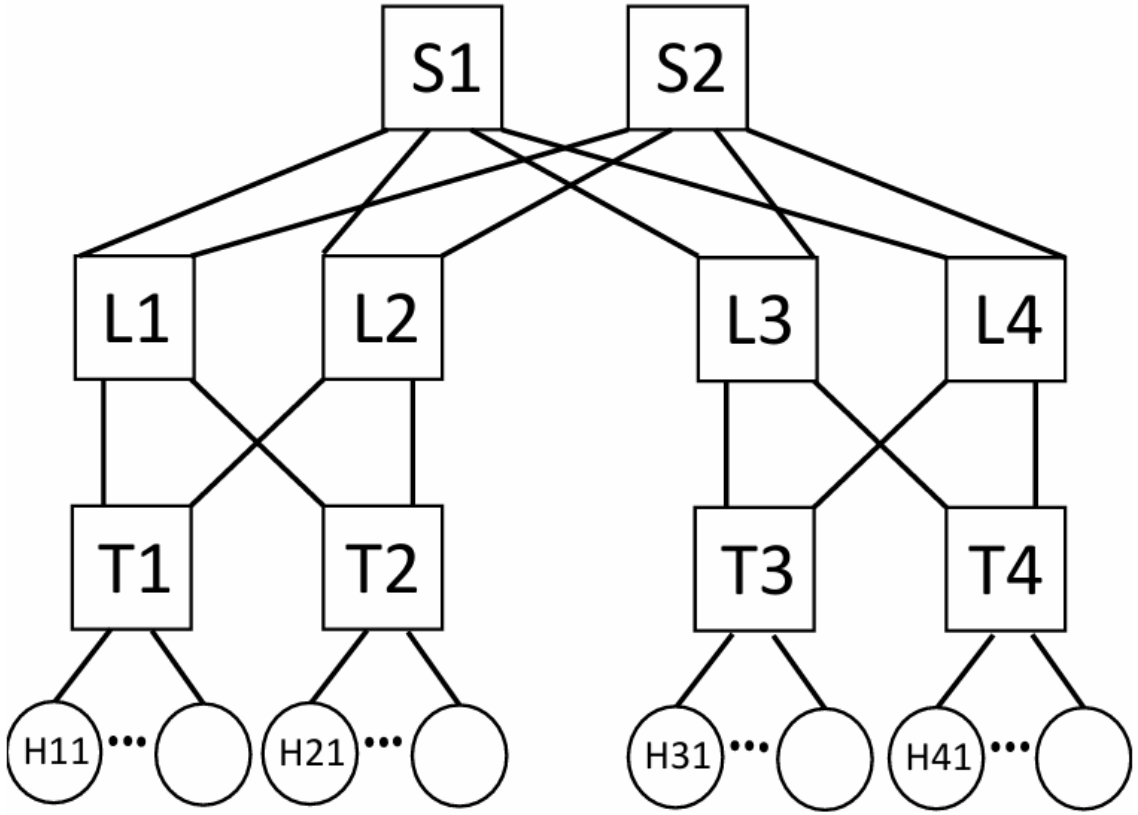
- 传统的TCP协议栈无法在低CPU开销和超低延迟的同时提供高带宽，而RoCEv2协议下的RDMA可以
- 延迟是小型传输的关键指标。在比较2KB消息的传输延迟时，TCP的延迟（25.4微秒）显著高于RDMA（1.7微秒用于Read/Write操作，2.8微秒用于Send操作）。这些实验结果表明，在低延迟和高吞吐量的要求下，RDMA优于传统的TCP/IP协议



(c) Mean Latency

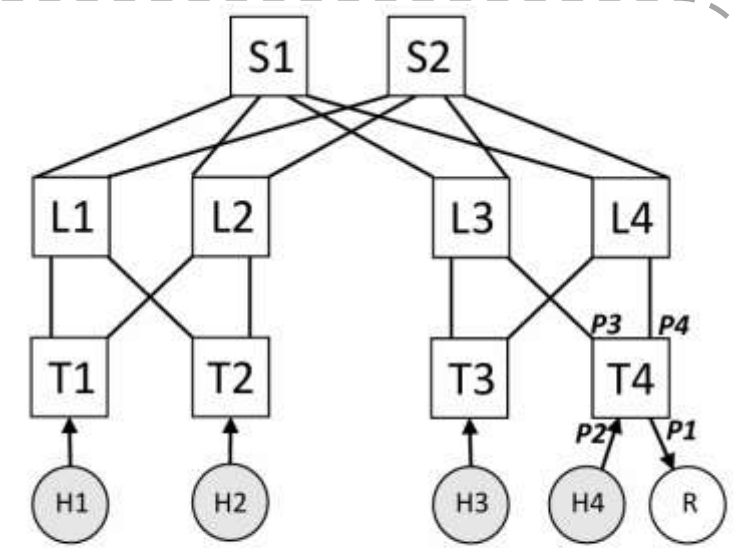
# The Need For DCQCN

- PFC对于RoCEv2的正常运行至关重要，PFC防止了以太网交换机和网卡的缓冲区溢出。PFC机制通过在队列超过阈值时向上游实体发送PAUSE消息来控制数据流
- 然而，PFC是基于端口和优先级来控制流量的，它无法针对单独的流进行精细化控制，从而会导致流间的不公平性，并且会传播拥塞，影响RoCEv2的性能
- 下面举例说明

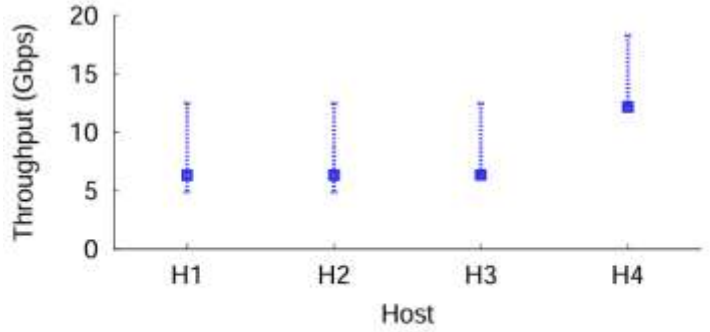


# The Need For DCQCN

- **场景**：主机 H1-H4 向接收端 R 发送数据，共享链路 T4-R 这一瓶颈链路
- 当 R 向 T4 发送 PAUSE 消息时，T4 会通过端口 P2、P3、P4 向上游节点发送 PAUSE 消息
- **结果**：主机 H4 的吞吐量高于 H1-H3
- **解释**：当T4的队列开始堆积时，它会暂停输入链路（端口P2-P4）。但P2仅承载单个流（来自H4），而P3和P4可能承载多个流——由于H1、H2和H3必须共享这两个端口（具体取决于ECMP的流映射机制），因此H4获得的吞吐量高于H1-H3




(a) Topology

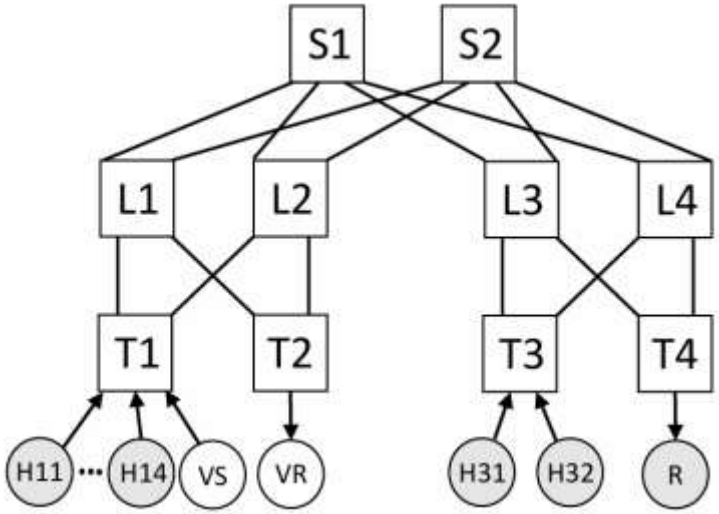



(b) Throughput of individual senders

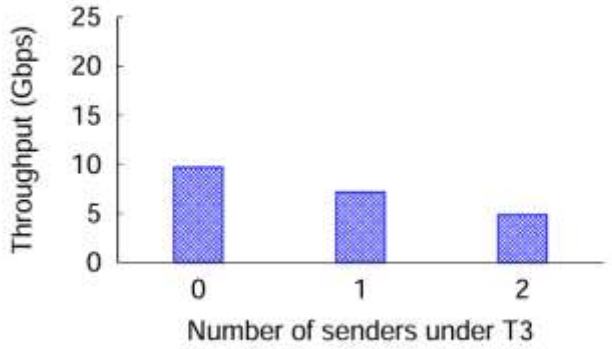
# The Need For DCQCN

**PFC 导致的 “受害流” 问题**

- PAUSE 消息会向上游级联传播，某一流可能因非自身路径上的拥塞而受到影响，当被暂停时，数据包会在上游节点排队累积
- **场景 1**：H11-H14 向 R 发送数据，VS 向 VR 发送数据→T4 向 L3、L4 发送 PAUSE，最终导致 T1 被暂停
- **场景 2**：新增 H31、H32 向 R 发送数据→S1、S2 被暂停的时间更长
- 结果：即使H31和H32到R的路径与VS到VR的路径没有任何公共链路，中位吞吐量仍从10Gbps进一步降至4.5Gbps。这是因为H31和H32与H11-H14在L3和L4链路上产生竞争，导致S1和S2交换机更长时间处于PAUSE状态



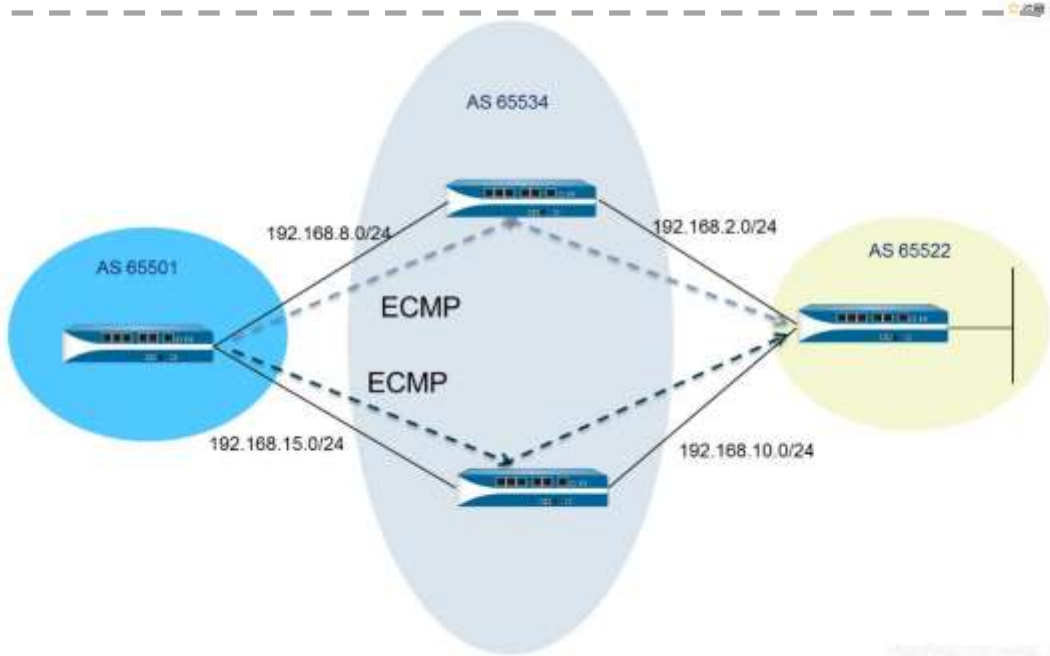

(a) Topology

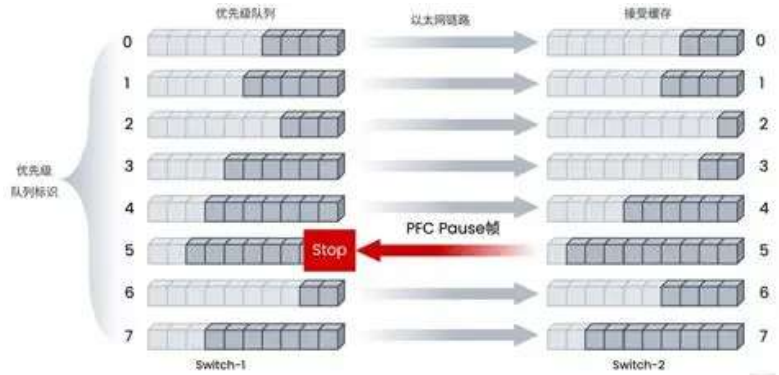



(b) Median throughput of victim flow

# The Need For DCQCN

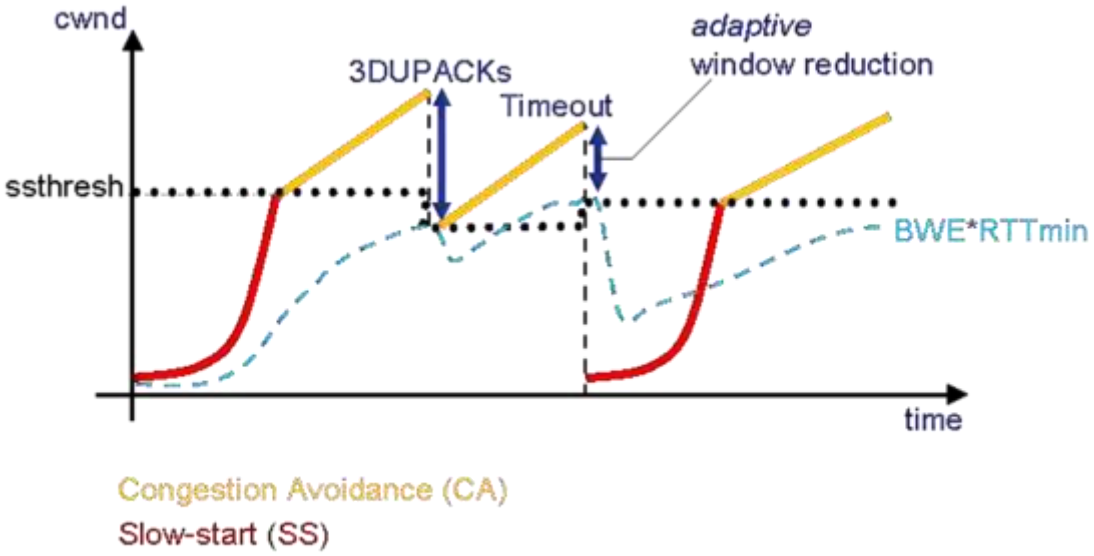
- 一些现有的方案尝试解决PFC的局限性
- 例如，有人提出通过ECMP（等成本多路径）来缓解流量不公平的问题。理论上，ECMP可以通过将流量分配到多个链路上来避免拥塞问题。然而，实验结果表明，ECMP并非总是有效，特别是在高流量环境下，ECMP无法完全解决PFC的限制
- PFC标准本身提供了一个优先级的概念，旨在缓解头部阻塞问题。尽管PFC支持多达8个优先级类别，但实验表明，当拓扑结构扩展或发送者增加时，PFC的性能问题仍然存在



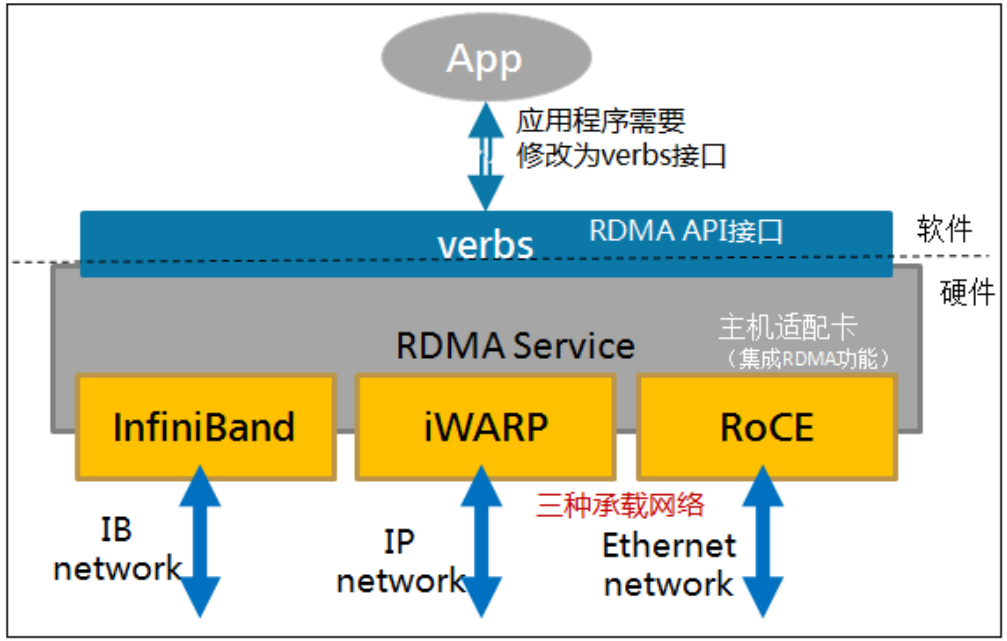
# The Need For DCQCN

- 解决PFC问题的根本方法是使用流级拥塞控制，如果能够对每个流进行独立的拥塞控制，PFC的触发频率将大大减少，从而避免流的不公平性和拥塞扩散的问题
- Quantized Congestion Notification（QCN）标准是为解决PFC问题而提出的一种方案。QCN在L2域中启用流级拥塞控制，但无法应用于L3网络，因为QCN基于L2地址来识别流，而IP路由网络中Ethernet头部信息会被剥离，无法识别每个流。这使得QCN不能直接用于RoCEv2等L3网络中



# The Need For DCQCN

- 扩展QCN协议到L3网络很难。为了将QCN应用到IP路由网络，必须使用IP五元组作为流标识，并且需要在交换机和NIC中添加IP和UDP头来支持拥塞通知的反馈。这种扩展需要对硬件进行修改，这在实际部署中非常困难，并且需要较长的开发和验证时间
- 现有的拥塞控制方案（如TCP-Bolt和iWarp）并不完全满足我们的需求。TCP-Bolt虽然没有慢启动阶段，但它仍然在软件中实现，导致高CPU开销和较高延迟。iWarp方案则依赖于全TCP栈，在大规模数据中心环境中无法提供像RoCEv2那样的低延迟和高吞吐量
- 基于此，论文提出DCQCN协议



# Outline

I. Introduction

II. The Need For DCQCN

**III. The DCQCN Algorithm**

IV. Buffer Settings

V. Analysis OF DCQCN

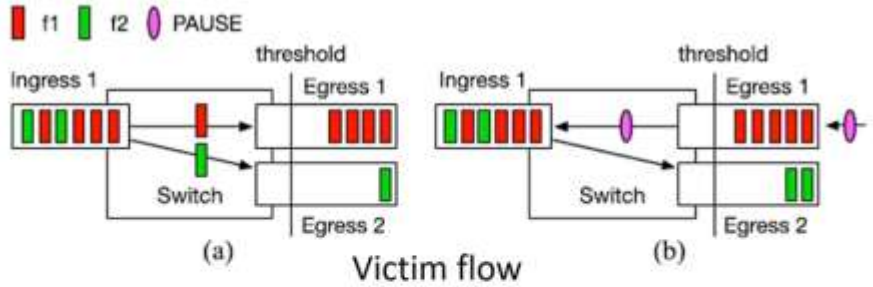
VI. Results

VII. Discussion

VIII. Review

# The DCQCN Algorithm

- DCQCN是一种基于速率的端到端拥塞控制协议，建立在QCN和DCTCP的基础上。大部分DCQCN的功能都在NIC中实现
- DCQCN旨在满足以下三个核心需求：
  ① 能够在无损、L3路由的网络中运行
  ② 端主机CPU开销低
  ③ 在无拥塞的情况下能够快速启动
- DCQCN还需要提供快速收敛的带宽分配，避免队列的震荡，并确保高链路利用率




**PFC is a coarse-grained mechanism, which does not distinguish between flows.**

**Solution:**

**A flow-level congestion control protocol**

- Incur low CPU overhead on end hosts
- Function over lossless, L3 routed, datacenter network
- Provide hyper-fast start in the common case of no congestion

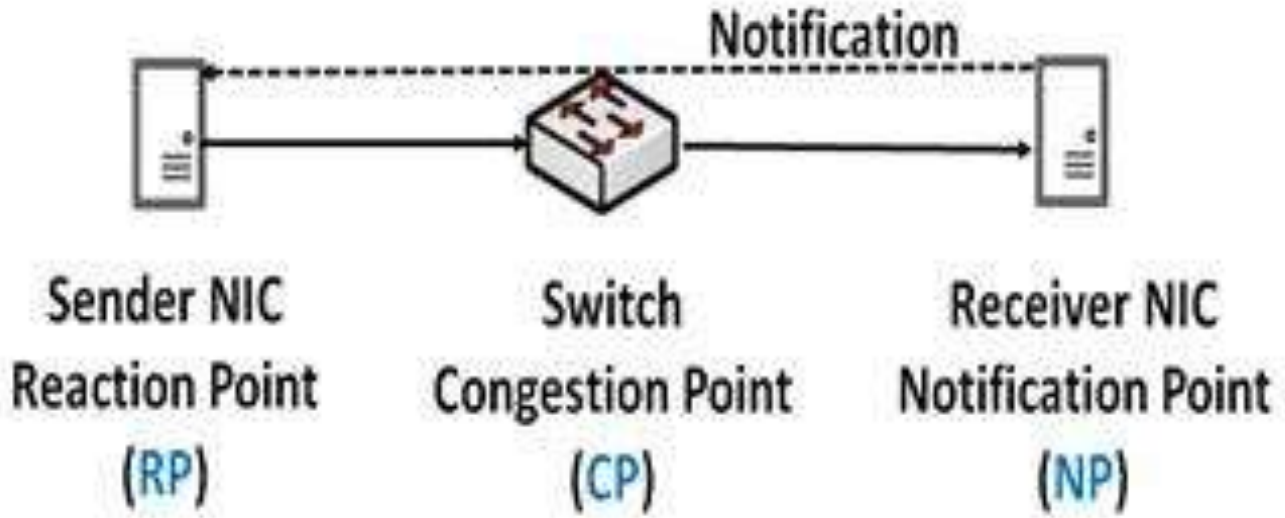
# The DCQCN Algorithm

在DCQCN的设计中，考虑了一些实际问题：

- 无法要求交换机提供任何定制的功能
- 由于协议是实现于NIC上的，必须关注实现的开销和复杂性

DCQCN的算法包括：

- 发送方（反应点，RP）
- 交换机（拥塞点，CP）
- 接收方（通知点，NP）



# The DCQCN Algorithm

- **CP算法：** 在拥塞点（CP），当出站队列的长度超过设定的阈值时，使用ECN（显式拥塞通知）标记到达的包。这一过程是通过RED（随机早期检测）算法实现的
- **NP算法：** 当ECN标记的数据包到达通知点（NP）时，表示网络发生了拥塞，NP将该信息传递给发送方。RoCEv2标准定义了显式拥塞通知包（CNP）来实现这一过程。NP算法定义了何时生成CNP，并且每个流的CNP生成是基于状态机的。若一个流没有在N微秒内发送CNP，则立即生成CNP，否则每N微秒生成一个CNP

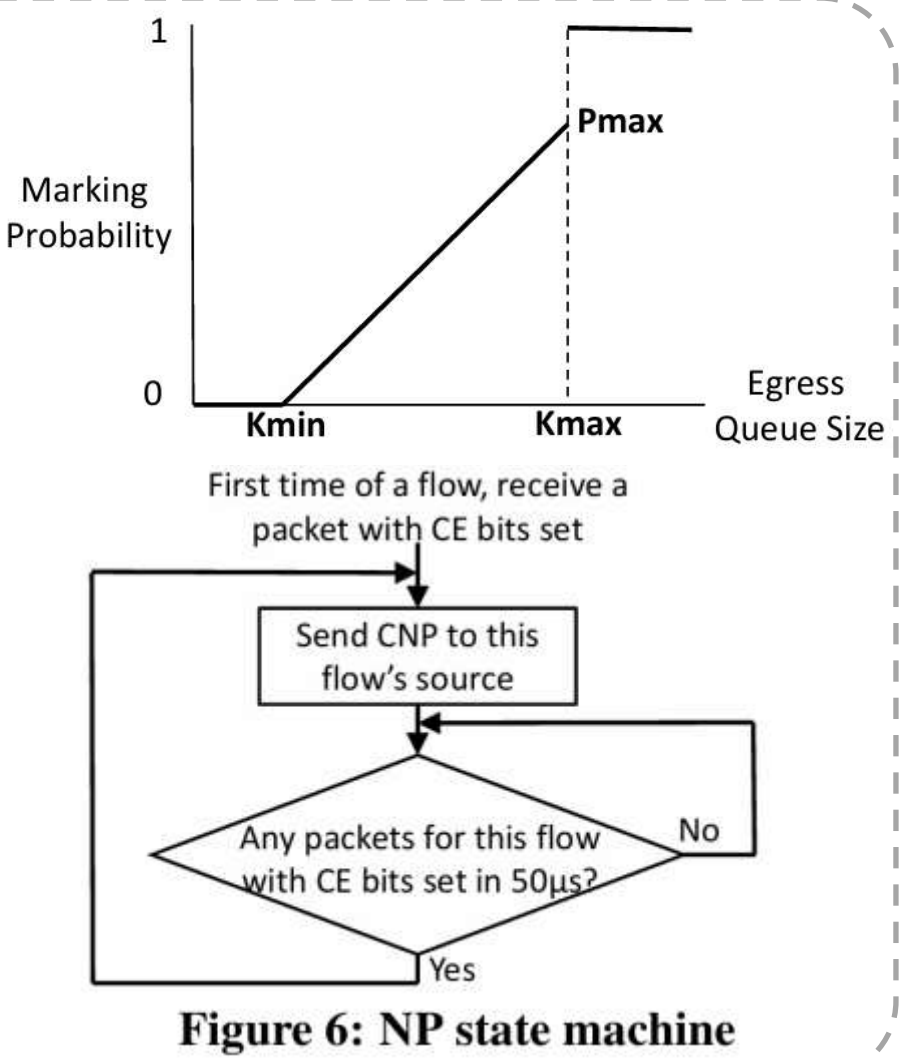
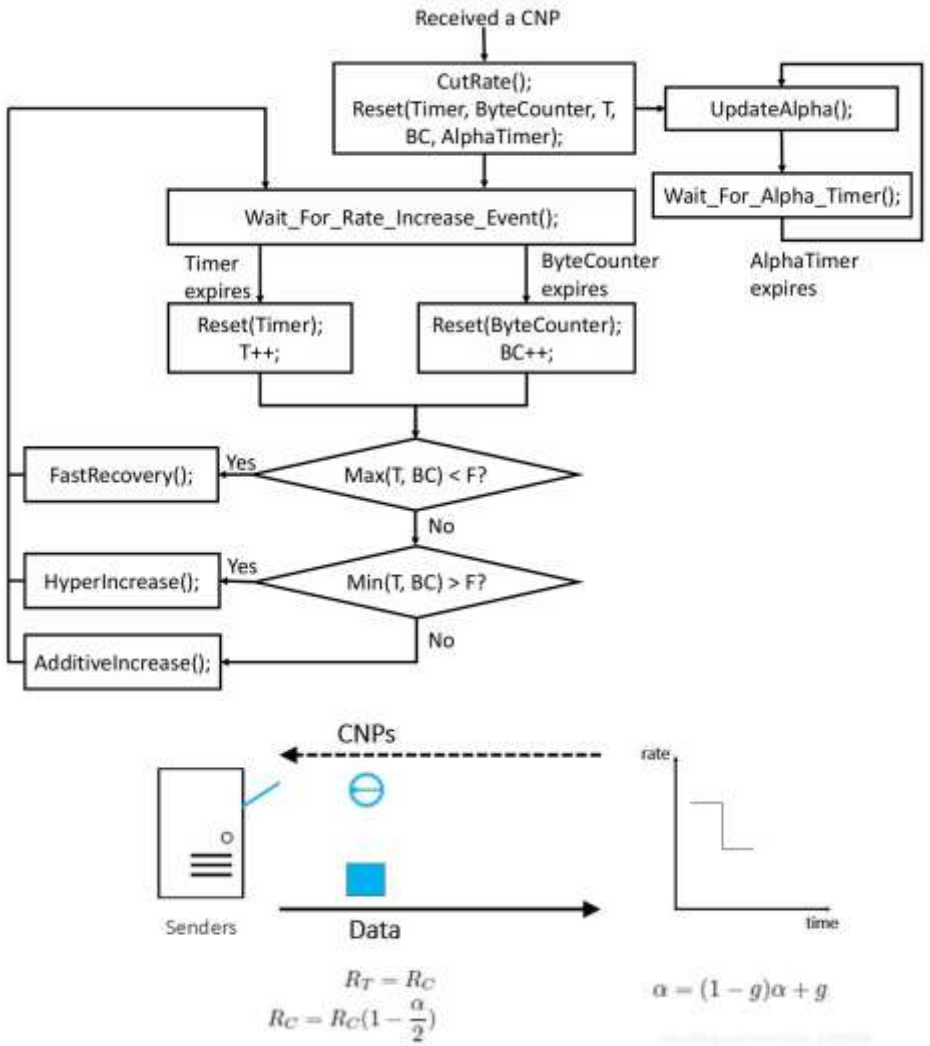



Figure 6: NP state machine

# The DCQCN Algorithm

**RP算法：**

- 发送方（RP）收到CNP后，会减少当前发送速率（$R_C$），并更新速率减少因子（α），像DCTCP一样，RP会将当前速率保存为目标速率（$R_T$）以备后续恢复。速率的更新遵循以下公式：
  - $R_T = R_C$
  - $R_C = R_C * (1 - \alpha/2)$
  - $\alpha = (1 - g)\alpha + g$
- RP在收到CNP后还会使用定时器和字节计数器增加速率，这与QCN中的速率恢复方法类似



# The DCQCN Algorithm

- 若RP在K个时间单位内未收到反馈，更新α （注意K必须大于CNP生成计时器）

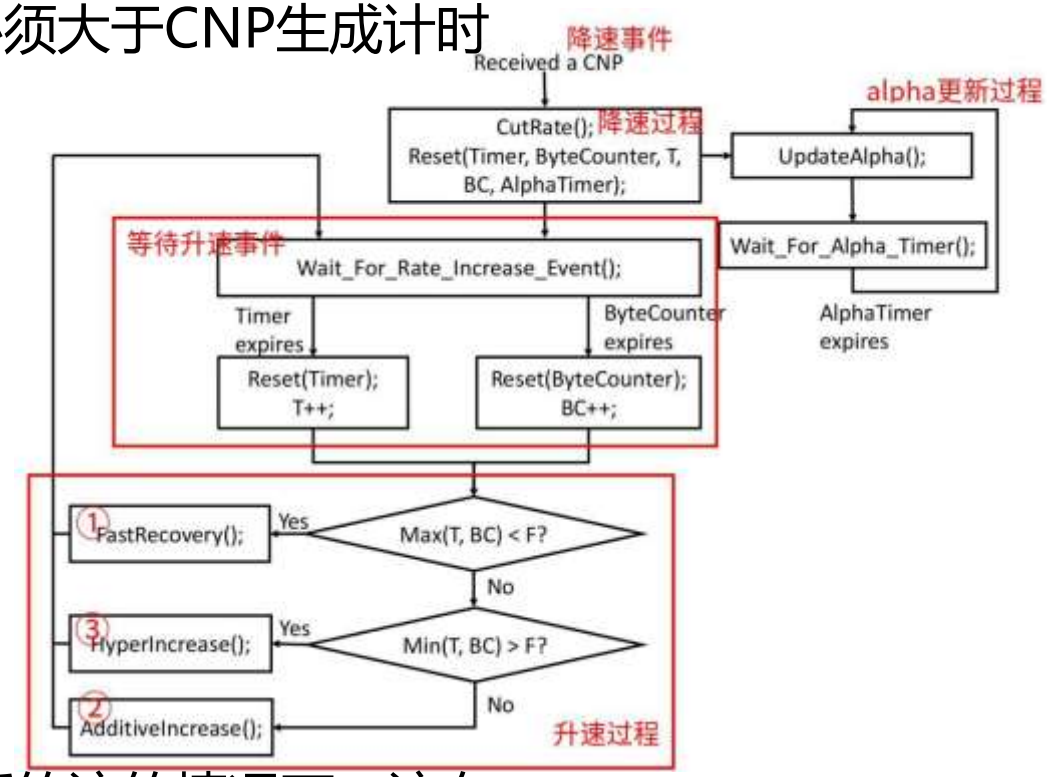
$$\alpha = (1 - g)\alpha$$

- 流的速率恢复分为两个阶段：
  - 快速恢复（将速率迅速增加到目标速率）

$$R_C = (R_T + R_C) / 2$$

  - 加性增加（流速率缓慢接近目标速率）

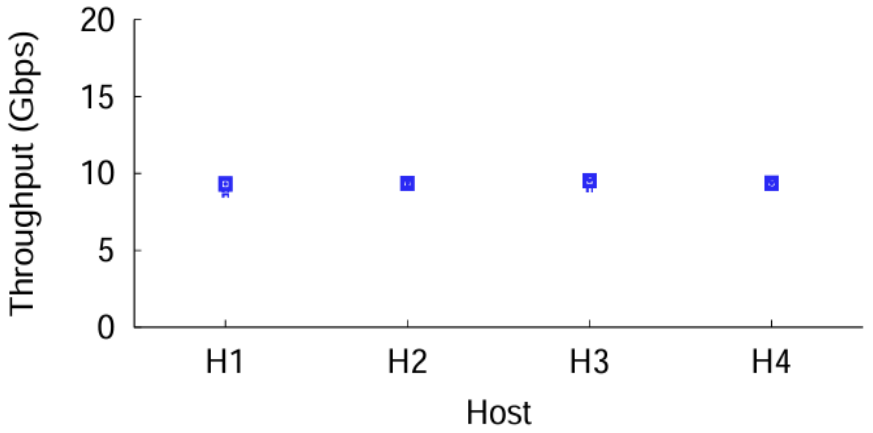
$$R_T = R_T + R_{AI} \qquad R_C = (R_T + R_C) / 2$$

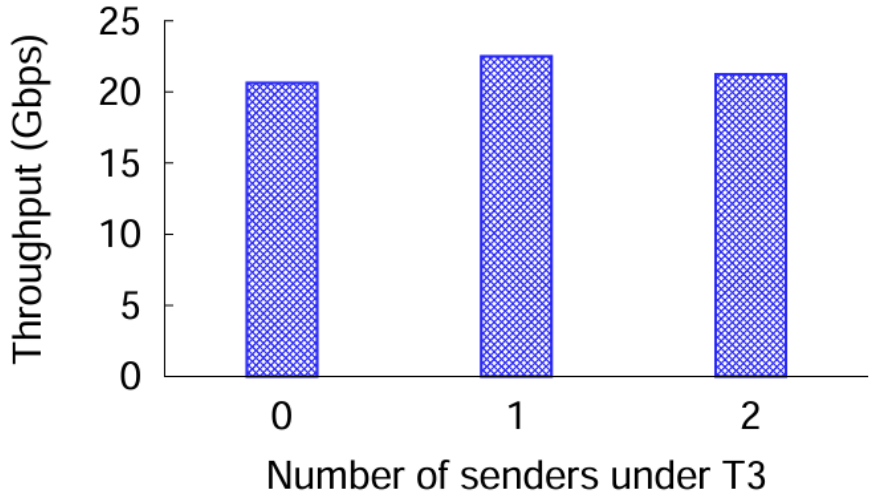



- DCQCN的设计没有包括慢启动阶段，在没有其他活跃的流的情况下，流在开始时即以满线速进行传输。这一设计旨在优化常见的情境，即流量传输量较小，且网络没有拥塞

# The DCQCN Algorithm

- 通过提供基于流的拥塞控制，DCQCN能够缓解PFC的局限性
- 为验证这一点，我们重复了第2.2节中的实验，并启用了DCQCN
- 实验结果表明，DCQCN成功解决了PFC带来的不公平性问题，并且提高了吞吐量。在四个发送流的情况下，每个流都能平等分享瓶颈链路的带宽，且吞吐量波动很小



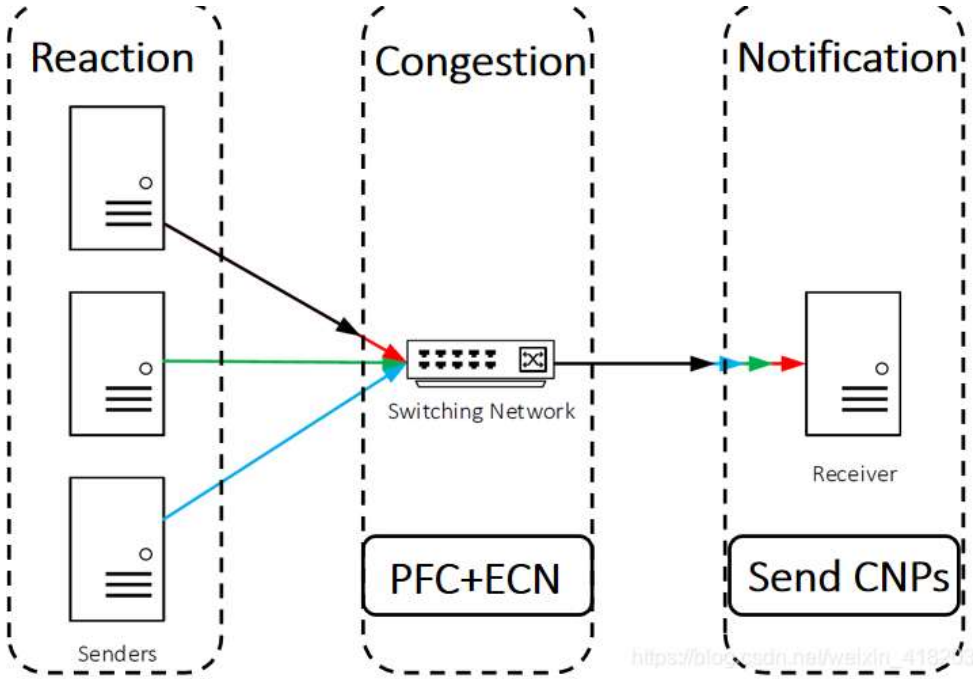



# The DCQCN Algorithm

**CNP生成机制：** DCQCN对于反向路径的拥塞不特别敏感，因为发送速率并不依赖于RTT（往返时间）估计。尽管如此，我们仍然为CNP设置了高优先级，以避免错过CNP生成的时间窗口，并加快收敛速度

**基于速率的拥塞控制：**DCQCN采用基于速率的拥塞控制方案，相比基于窗口的方案，速率控制更简单，也能提供更细粒度的控制

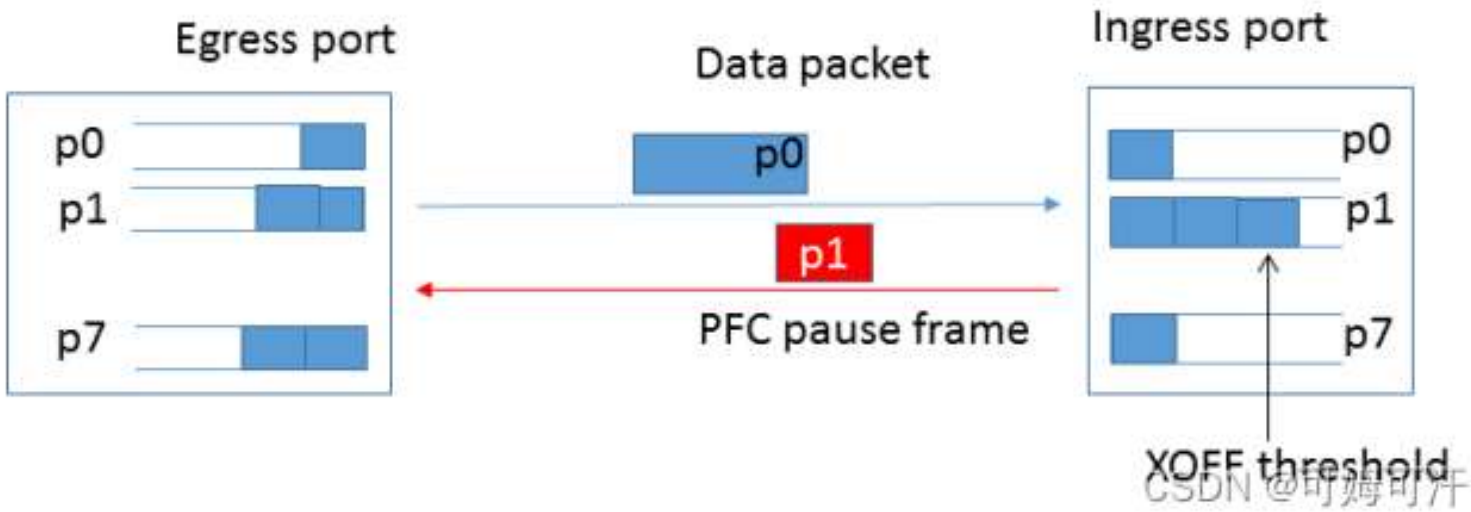
**参数设置：**与QCN和DCTCP相比，DCQCN在某些关键方面有所不同。例如，DCQCN没有量化的反馈机制（不像QCN那样），也没有基于每个确认的反馈（不像DCTCP那样）。因此，不能盲目使用QCN和DCTCP的参数设置



# The DCQCN Algorithm

**PFC的必要性：**DCQCN并未消除对PFC的需求。采用DCQCN时，数据流以线路速率启动。若未启用PFC，则可能导致数据包丢失和性能下降

**硬件实现：**NP（网络处理器）与RP（速率限制器）状态机均部署在网卡上。RP状态机除需维护少量其他状态（如α当前值）外，还需为每个被限流的流维护一个定时器和一个计数器。这些状态均保存在网卡芯片上



# Outline

I. Introduction

II. The Need For DCQCN

III. The DCQCN Algorithm

**IV. Buffer Settings**

V. Analysis OF DCQCN

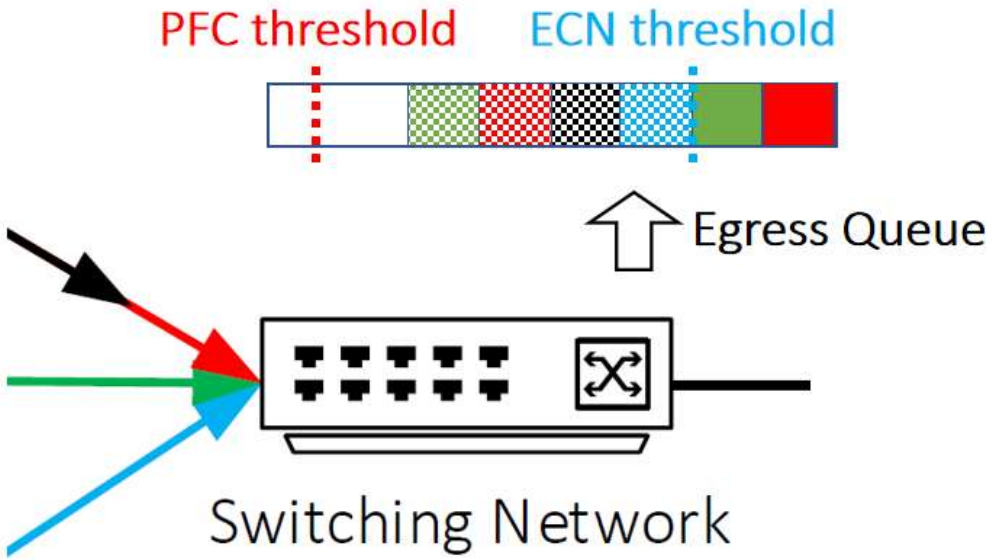
VI. Results

VII. Discussion

VIII. Review

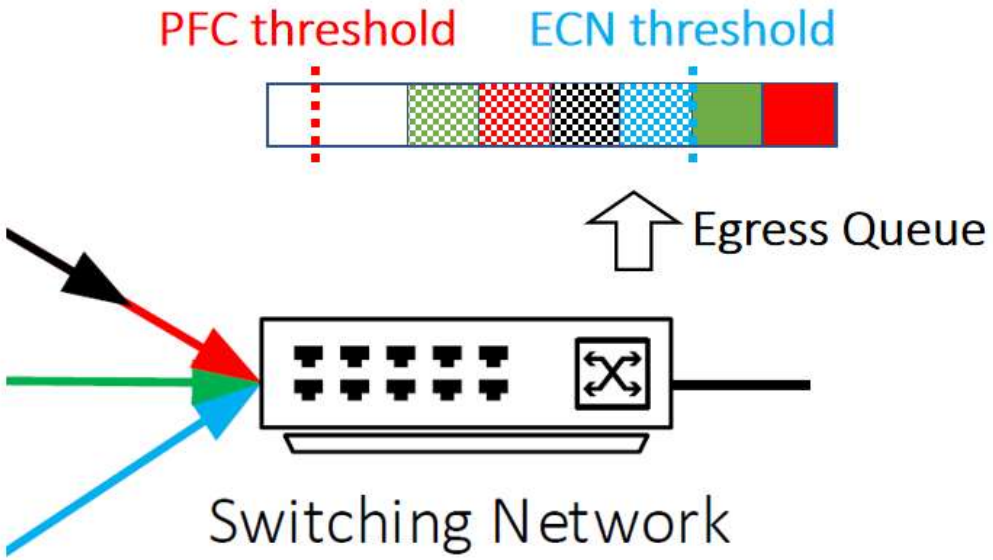
# Buffer Settings

- DCQCN需要平衡两个相互冲突的要求：
  - PFC不能过早触发，即在ECN有机会发送拥塞反馈之前不触发PFC
  - PFC不能触发太晚，导致缓冲区溢出造成数据包丢失
- 为了满足这些要求，需要精确设置交换机的三个关键参数：$t_{flight}$（头部缓冲区）、$t_{PFC}$（PFC阈值）和$t_{ECN}$（ECN阈值）



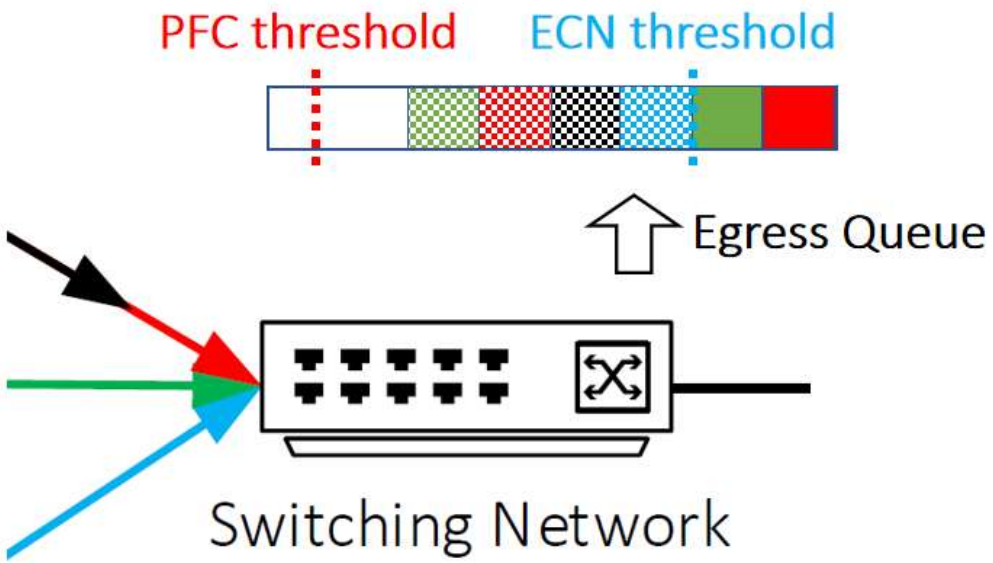
# Buffer Settings

- $t_{flight}$表示在发送PAUSE消息到上游设备之前，交换机需要预留的缓冲区大小，以避免由于PAUSE消息的传播延迟导致数据包丢失。根据最坏情况下的计算，$t_{flight}$的值应为每个端口每个优先级22.4KB，这样可以确保在发送PAUSE消息期间交换机能够处理传输中的数据包



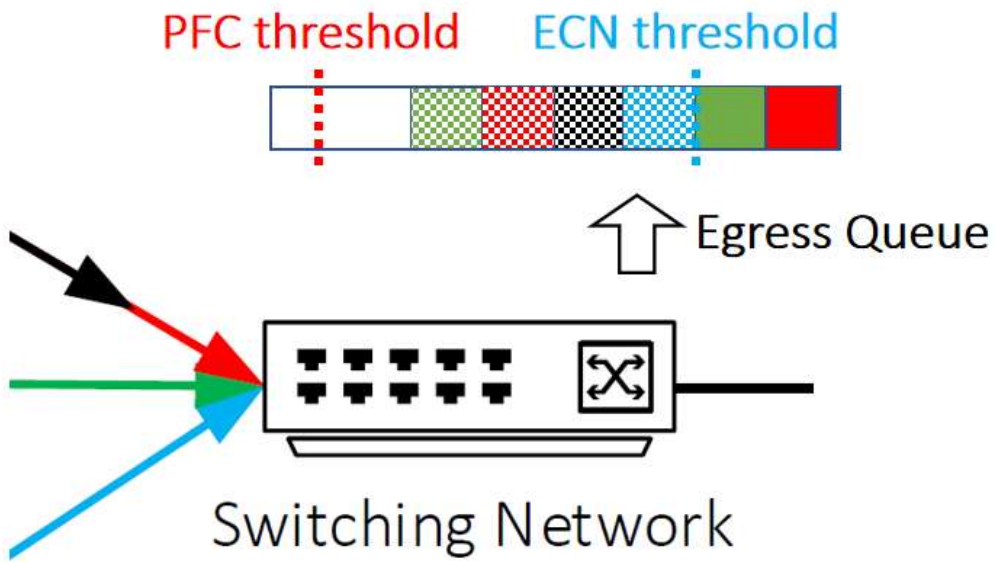
# Buffer Settings

- $t_{PFC}$是入口队列的最大大小，当队列超过此阈值时，交换机会发送PAUSE消息，暂停上游设备的数据传输。根据计算，$t_{PFC}$的最大值为24.47KB。这一值确保交换机在触发PAUSE消息时，能够有效管理缓冲区并防止数据包丢失。RESUME消息将会在队列大小恢复到$t_{PFC}$以下时被发送



# Buffer Settings

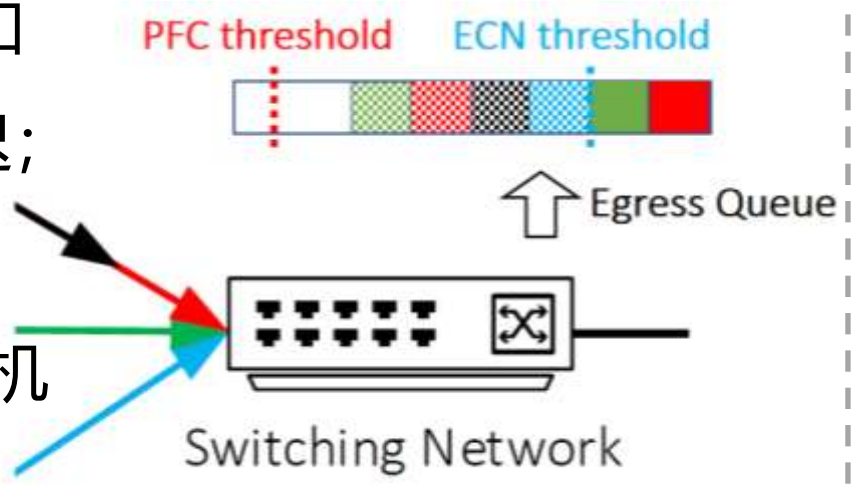
- $t_{ECN}$是出口队列的阈值，当队列大小超过该值时，交换机会标记数据包以告知网络存在拥塞。在DCQCN的设计中，$t_{ECN}$的值必须比$t_{PFC}$更小，以确保在PFC触发之前，ECN已经开始标记数据包。通过计算，得出$t_{ECN}$的最佳设置应小于0.85KB，这比一个MTU的大小还要小，因此不可行



# Buffer Settings

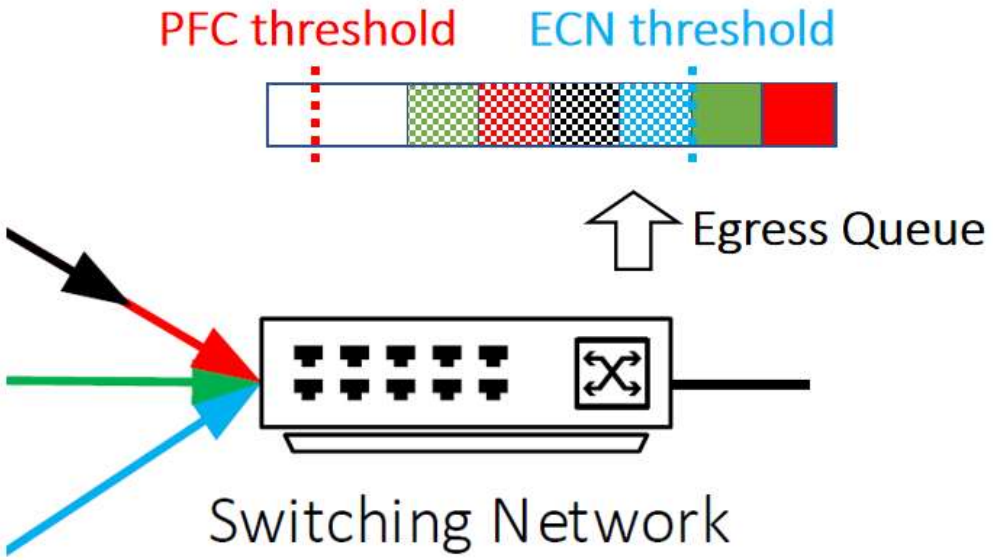
- 为了更好地管理缓冲区的使用，$t_{PFC}$值不应是一个固定值，而应依赖于交换机的空闲缓冲区大小
- 随着缓冲区的使用量变化，$t_{PFC}$的值应动态调整。如果缓冲区较为空闲，则可以容忍更长的PFC触发延迟；反之，则应更早触发PFC
- 基于此动态设置，参数β被引入来调整$t_{PFC}$的触发时机

  $t_{PFC} = \beta(B-8nt_{flight} -S) /8$
- β值越高，PFC触发频率越低；β值越低，则越频繁触发PFC



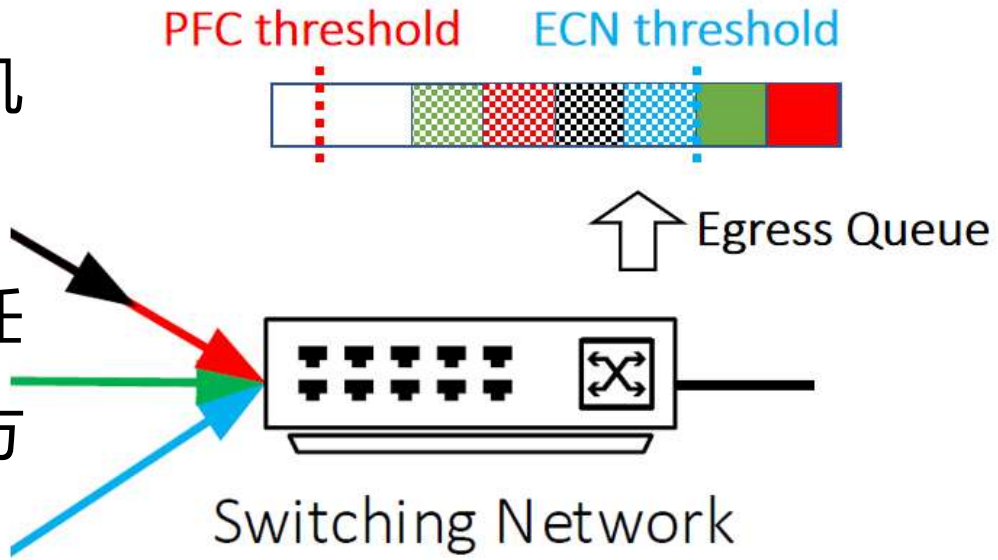
# Buffer Settings

- 在任何出口端口触发ECN之前，有：
  - $s \leq n^* t_{ECN}$
- 为确保ECN始终先于PFC触发，设定：
  - $t_{ECN} < \beta(B-8n\, t_{flight})/(8n(\beta+1))$
- 显然，更大的β值为$t_{ECN}$留出更多余量，在测试平台中，采用β=8，这使得 $t_{ECN}$ <21.75KB



# Buffer Settings

- 上述分析采用保守策略，确保即使在最坏情况及使用全部8个PFC优先级时，交换机在ECN之前也不会触发PFC。若优先级数量减少或交换机缓冲区增大，阈值设定将相应调整
- 分析并不表示PFC永远不会被触发。仅确保在任何交换机上，PFC不会在ECN之前触发。发送方接收ECN反馈并降低发送速率需要一定时间，在此期间PFC仍可能被触发。如前所述，依赖PFC使发送方能够以线速启动



# Outline

I. Introduction

II. The Need For DCQCN

III. The DCQCN Algorithm

IV. Buffer Settings

**V. Analysis OF DCQCN**

VI. Results

VII. Discussion

VIII. Review

# Analysis OF DCQCN

- 使用流体模型来分析DCQCN的性能，并确定其最佳参数设置。流体模型用于模拟在多个流共享同一瓶颈时，如何根据DCQCN协议调整流量速率
- 流体模型中的变量和参数在表1和表2中列出
- 假设有N个流竞争同一个瓶颈链路，链路容量为C，在模型中，DCQCN协议通过流量速率的调整来减少队列积压，并快速恢复到公平的带宽分配状态
- 忽略了PFC的影响，假设DCQCN在PFC触发之前就已经有效地减轻了拥塞

| Variable | Description |
|---|---|
| $R_c$ | Current Rate |
| $R_t$ | Target Rate |
| $\alpha$ | See Equation (1) |
| $q$ | Queue Size |
| $t$ | Time |

**Table 1: Fluid model variables**

| Parameter | Description |
|---|---|
| $K_{min}, K_{max}, P_{max}$ | See Figure 5 |
| $g$ | See Equation (1) |
| $N$ | Number of flows at bottleneck |
| $C$ | Bandwidth of bottleneck link |
| $F$ | Fast recovery steps (fixed at 5) |
| $B$ | Byte counter for rate increase |
| $T$ | Timer for rate increase |
| $R_{AI}$ | Rate increase step (fixed at 40Mbps) |
| $\tau*$ | Control loop delay |
| $\tau'$ | Interval of Equation (2) |

**Table 2: Fluid model parameters**

# Analysis OF DCQCN

- 流体模型的核心是通过计算每个流的速率变化来模拟网络中的拥塞情况：
- 公式（5）表示在瓶颈链路上，数据包被标记为ECN的概率，当队列长度超过$K_{min}$时，数据包会被标记；当队列长度在$K_{min}$和$K_{max}$之间时，标记概率会随着队列长度的增加而线性增加；当队列长度超过$K_{max}$时，所有数据包都会被标记
- 公式（6）表示瓶颈队列的演化，假设所有流的速率相等。在这个模型中，队列的长度会根据流的速率和瓶颈链路的容量变化。公式中的速率变化反映了每个流如何根据拥塞控制协议调整其速率，以避免队列长度过长

$$p(t)=\begin{cases}0, & q(t)\le K_{\min}\\ \frac{q(t)-K_{\min}}{K_{\max}-K_{\min}}p_{\max}, & K_{\min}<q(t)\le K_{\max}\\ 1, & q(t)>K_{\max}\end{cases} \tag{5}$$

$$\frac{dq}{dt}=NR_C(t)-C \tag{6}$$

$$\frac{d\alpha}{dt}=\frac{g}{\tau'}\left(\left(1-(1-p(t-\tau*))^{\tau' R_C(t-\tau*)}\right)-\alpha(t)\right) \tag{7}$$

$$\begin{aligned}\frac{dR_T}{dt}=&-\frac{R_T(t)-R_C(t)}{\tau}\left(1-(1-p(t-\tau*))^{\tau R_C(t-\tau*)}\right)\\&+R_{AI}R_C(t-\tau*)\frac{(1-p(t-\tau*))^{FB}p(t-\tau*)}{(1-p(t-\tau*))^{-B}-1}\\&+R_{AI}R_C(t-\tau*)\frac{(1-p(t-\tau*))^{FTR_C(t-\tau*)}p(t-\tau*)}{(1-p(t-\tau*))^{-TR_C(t-\tau*)}-1}\end{aligned} \tag{8}$$

$$\begin{aligned}\frac{dR_C}{dt}=&-\frac{R_C(t)\alpha(t)}{2\tau}\left(1-(1-p(t-\tau*))^{\tau R_C(t-\tau*)}\right)\\&+\frac{R_T(t)-R_C(t)}{2}\frac{R_C(t-\tau*)p(t-\tau*)}{(1-p(t-\tau*))^{-B}-1}\\&+\frac{R_T(t)-R_C(t)}{2}\frac{R_C(t-\tau*)p(t-\tau*)}{(1-p(t-\tau*))^{-TR_C(t-\tau*)}-1}\end{aligned} \tag{9}$$

# Analysis OF DCQCN

- 公式（7）描述了速率减少因子（α）随时间的变化。α的更新基于流量反馈的延迟（τ*），并且它随着流的速率变化而调整。公式中的更新机制确保了当网络发生拥塞时，流的速率会迅速减少，从而避免队列积压
- 公式（8）和（9）描述了发送方的速率更新规则。每个流的目标速率（$R_T$）会根据拥塞反馈进行调整，同时发送速率（$R_C$）也会根据速率变化进行更新。这些更新基于DCQCN协议中的字节计数器和定时器机制，以实现流量的平滑增加和减少

$$p(t) = \begin{cases} 0, & q(t) \le K_{\min} \\ \frac{q(t)-K_{\min}}{K_{\max}-K_{\min}} p_{\max}, & K_{\min} < q(t) \le K_{\max} \\ 1, & q(t) > K_{\max} \end{cases} \tag{5}$$

$$\frac{dq}{dt} = NR_C(t) - C \tag{6}$$

$$\frac{d\alpha}{dt} = \frac{g}{\tau'}\left(\left(1-(1-p(t-\tau*))^{\tau' R_C(t-\tau*)}\right) - \alpha(t)\right) \tag{7}$$

$$\begin{aligned} \frac{dR_T}{dt} = & -\frac{R_T(t)-R_C(t)}{\tau}\left(1-(1-p(t-\tau*))^{\tau R_C(t-\tau*)}\right) \\ & + R_{AI}R_C(t-\tau*)\frac{(1-p(t-\tau*))^{FB}p(t-\tau*)}{(1-p(t-\tau*))^{-B}-1} \\ & + R_{AI}R_C(t-\tau*)\frac{(1-p(t-\tau*))^{FTR_C(t-\tau*)}p(t-\tau*)}{(1-p(t-\tau*))^{-TR_C(t-\tau*)}-1} \end{aligned} \tag{8}$$

$$\begin{aligned} \frac{dR_C}{dt} = & -\frac{R_C(t)\alpha(t)}{2\tau}\left(1-(1-p(t-\tau*))^{\tau R_C(t-\tau*)}\right) \\ & + \frac{R_T(t)-R_C(t)}{2}\frac{R_C(t-\tau*)p(t-\tau*)}{(1-p(t-\tau*))^{-B}-1} \\ & + \frac{R_T(t)-R_C(t)}{2}\frac{R_C(t-\tau*)p(t-\tau*)}{(1-p(t-\tau*))^{-TR_C(t-\tau*)}-1} \end{aligned} \tag{9}$$

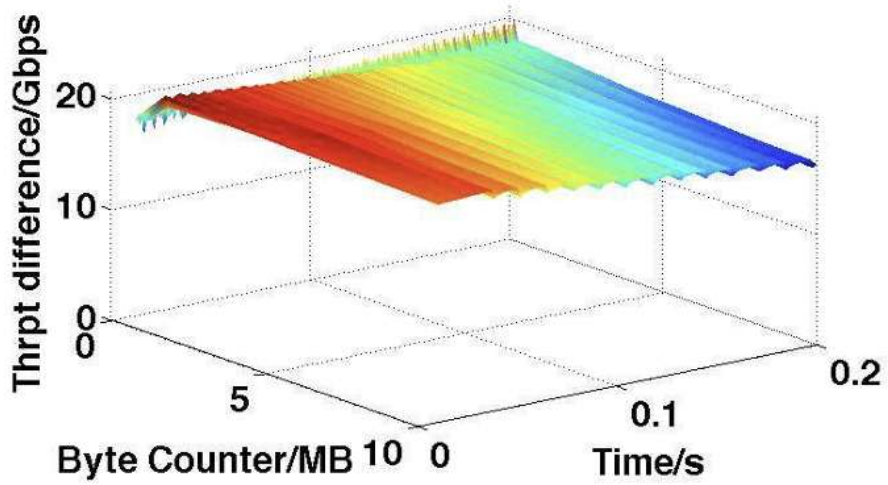
# Analysis OF DCQCN

- 将方程(6)-(9)的左侧设为零，可以发现当满足下面方程时，该流体模型具有唯一不动点
  - $R_C(t) = C/N$
- 该固定点表示所有流获得公平带宽份额C/N的状态
- 但已验证在合理的参数设置下，可以求解得p值（固定点处的ECN标记概率）小于1%。根据公式（5），当启用RED-ECN时，队列长度将接近$K_{min}$，因为此时p值接近0


- 将方程(6)-(9)的左侧设为零，可以发现当满足下面方程时，该流体模型具有唯一不动点
  - $R_C(t) = C/N$
- 该固定点表示所有流获得公平带宽份额C/N的状态
- 但已验证在合理的参数设置下，可以求解得p值（固定点处的ECN标记概率）小于1%。根据公式（5），当启用RED-ECN时，队列长度将接近$K_{min}$，因为此时p值接近0

# Analysis OF DCQCN

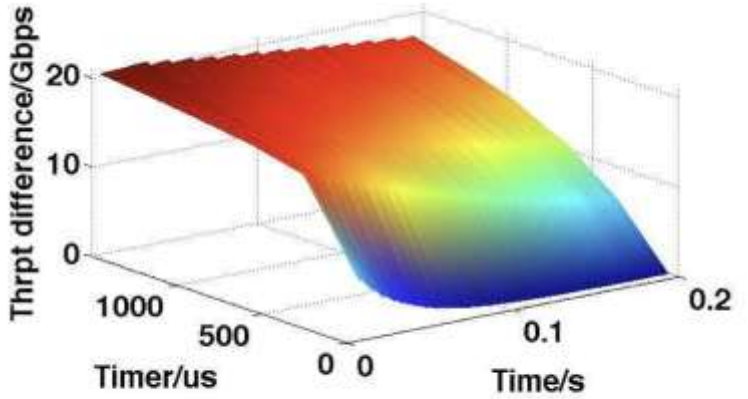
- 重点关注一个简单的两流系统，其中一个流的初始速率为40Gbps，另一个流的初始速率为0Gbps。在每个参数设置下，通过数值分析来解决前200毫秒的模型，并研究不同参数对DCQCN协议收敛性的影响。实验结果显示，DCQCN协议的收敛速度和队列稳定性依赖于参数的选择
- 从QCN和DCTCP的推荐参数开始，具体包括B=150KB、T=1.5ms、$K_{min}$= $K_{max}$=40KB、 $P_{max}$=1和g=1/16。然而，实验结果表明，在这些参数设置下，流无法很好地收敛。QCN的字节计数器主导了速率增加，这使得较快的流增长得更快，导致收敛问题



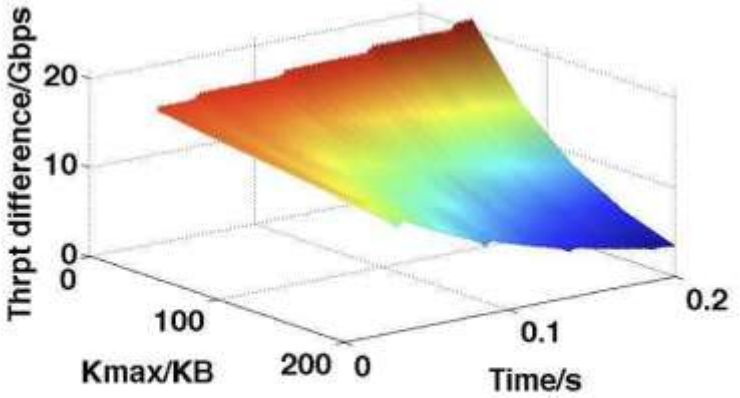

(a) Sweep Byte Counter with strawman parameters

# Analysis OF DCQCN

- 为了解决流无法收敛的问题，我们尝试减慢字节计数器的速度，结果表明减慢字节计数器能够缓解这个问题，但它会降低收敛速度。通过加速速率增加定时器
- 另一种解决方法是使用类似RED的概率包标记方案，而不是像DCTCP那样的“切断”行为（即队列长度超过一定阈值时所有包都会被标记）。我们通过使用较小的$P_{max}$来增加较大流获得更多CNP的概率，从而使其更快地回退



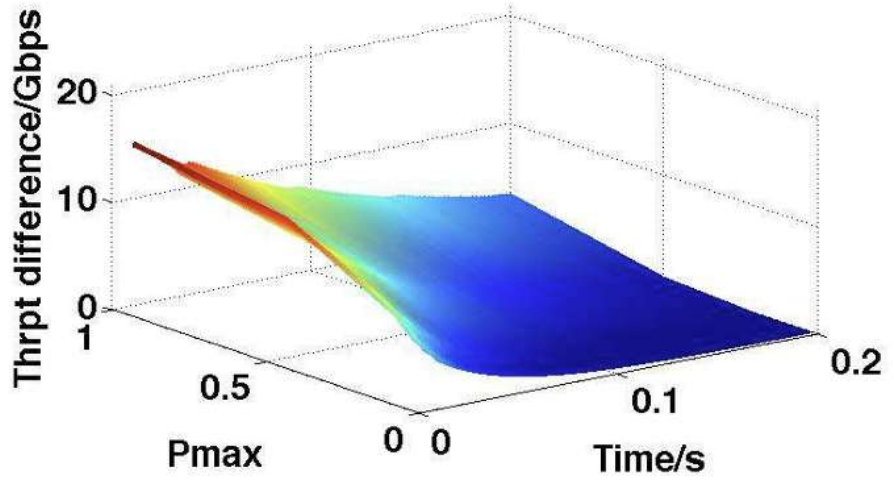

(b) Sweep Timer with 10MB Byte Counter




(c) Sweep $K_{max}$ with strawman parameters

# Analysis OF DCQCN

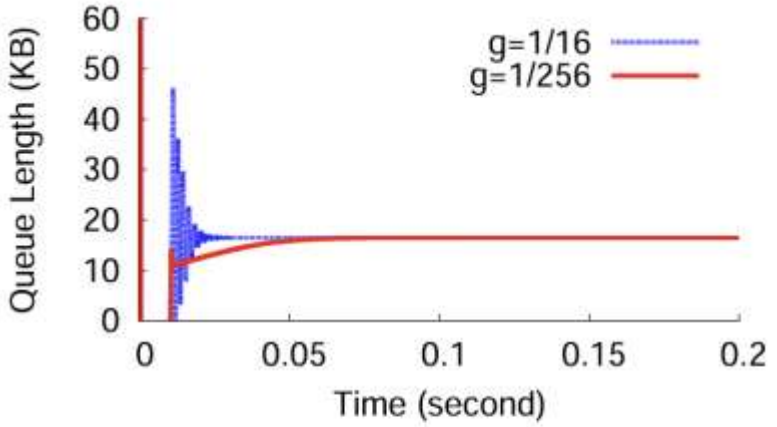
- 通过这些调整，我们找到了一个更合适的参数设置，具体为：使用RED-ECN并将$K_{max}$设置为200KB，$P_{max}$设置为1%，$K_{min}$设置为5KB。虽然5KB的$K_{min}$看起来较浅，但流体模型预测在该设置下，队列长度通常比$K_{min}$大一个数量级。因此，这个设置能够保证100%的吞吐量



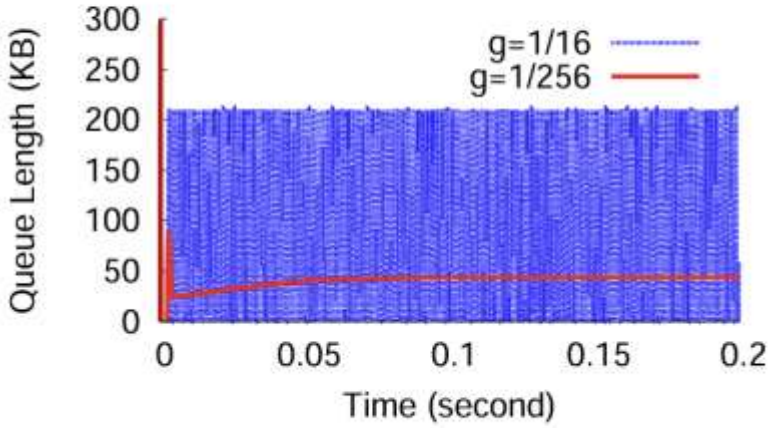

(d) Sweep $P_{\max}$ with $K_{\max} = 200KB$

# Analysis OF DCQCN

- 此外，我们还通过实验测试了不同的g值对队列长度和队列稳定性的影响。实验结果表明，较小的g值有助于减少队列长度和波动，虽然较小的g值会稍微减慢收敛速度，但由于较低的波动，这一权衡是值得的
- 我们还分析了RAI和F对协议性能的影响。RAI与g一起影响DCQCN的可扩展性。在当前设置下，16:1的incast负载不会出现缓冲区饥饿现象。如果减小RAI的值，收敛速度会降低，但可以确保在32:1的incast负载下不会出现缓冲区饥饿




(a) 2:1 Incast




(b) 16:1 Incast

# Outline


I. Introduction

II. The Need For DCQCN

III. The DCQCN Algorithm

IV. Buffer Settings

V. Analysis OF DCQCN

**VI. Results**

VII. Discussion

VIII. Review

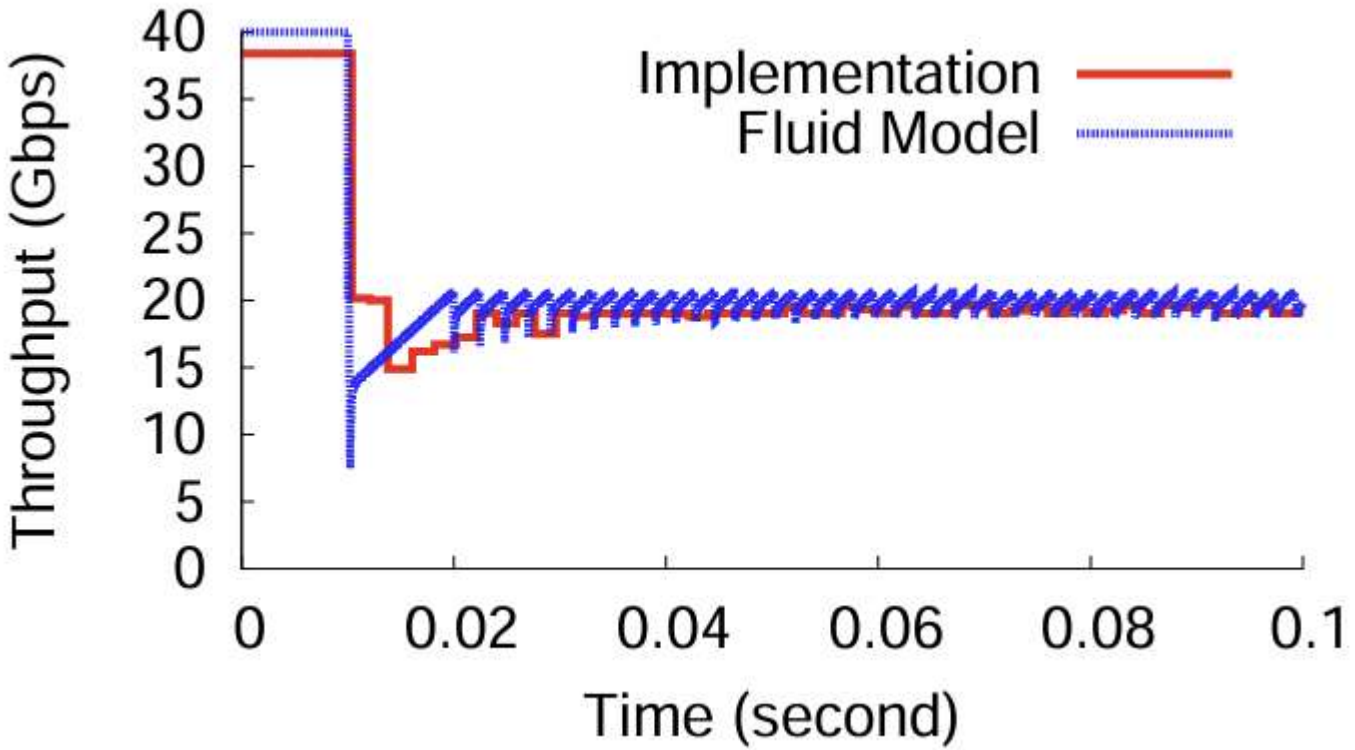
# Results

**验证流体模型**

- 为了分析DCQCN协议的收敛性特性，需要将流体模型扩展到具有不同速率的流。以两个流为例，分别描述每个流的当前速率和目标速率的演变，

$$\frac{dq}{dt} = R_{C1}(t) + R_{C2}(t) - C$$

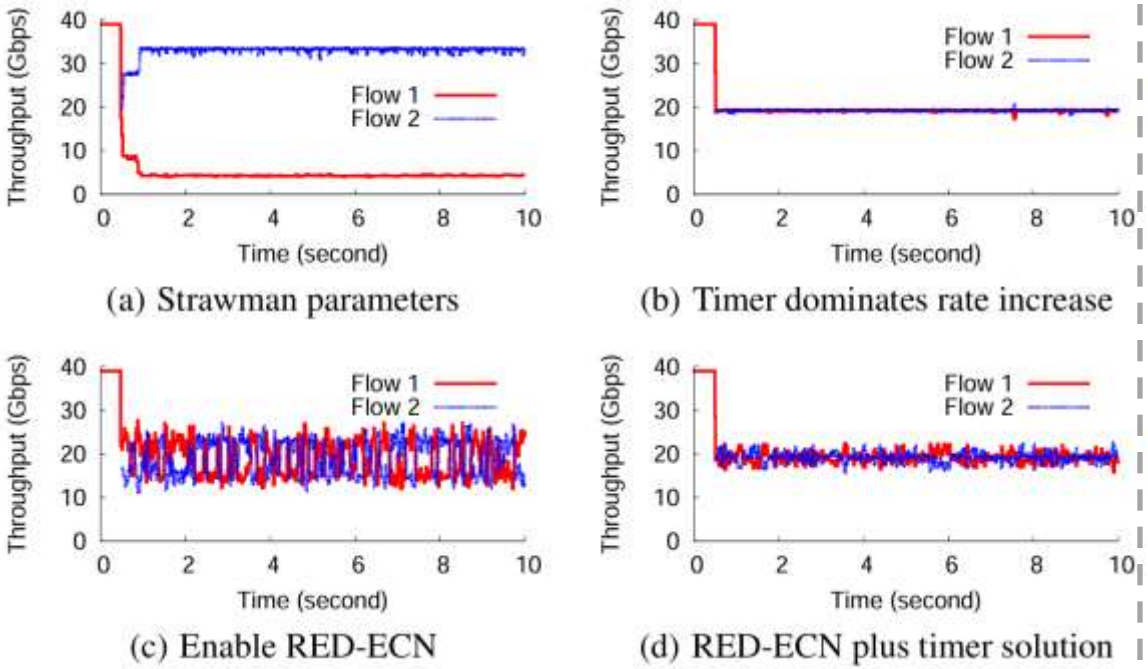
- 通过数值求解该模型，以了解各种参数（表2）对DCQCN行为的影响，特别是收敛性和队列积压的变化。实验表明，流体模型与实现的效果非常匹配



# Results

**验证参数设置**

- （a）：正如模型所预测的那样，两个竞争流之间存在不公平性
- （b）：快速率增长计时器能够缓解不公平性问题
- （c）：第二种方案——即RED-ECN，即使不改变计时器也能缓解不公平性问题
- （d）：结合快速定时器与RED-ECN机制能获得更优的性能表现



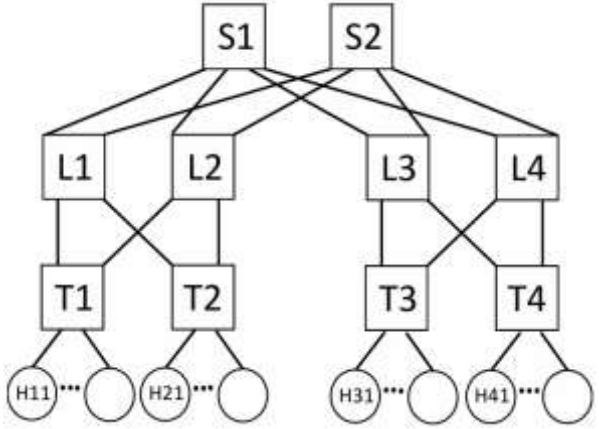

(a) Strawman parameters (b) Timer dominates rate increase

(c) Enable RED-ECN (d) RED-ECN plus timer solution

# Results

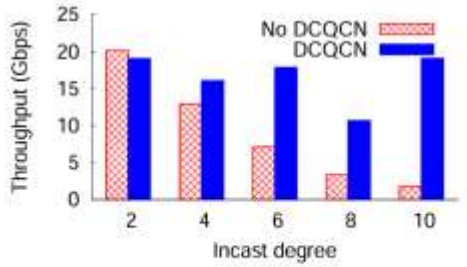
基准流量测试

- （a）（b）：在没有DCQCN的情况下，随着incast程度的增加，用户流量的吞吐量急剧下降
- 采用DCQCN时，第10百分位吞吐量非常接近该理论值，体现出高度公平性

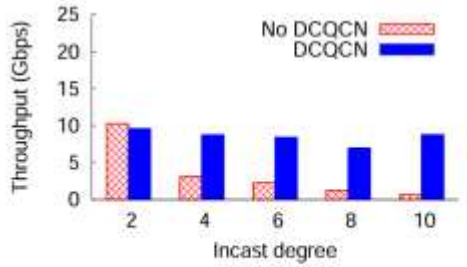



| Parameter | Value |
|---|---|
| Timer | $55\mu s$ |
| Byte Counter | 10MB |
| $K_{\max}$ | 200KB |
| $K_{\min}$ | 5KB |
| $P_{\max}$ | 1% |
| $g$ | 1/256 |

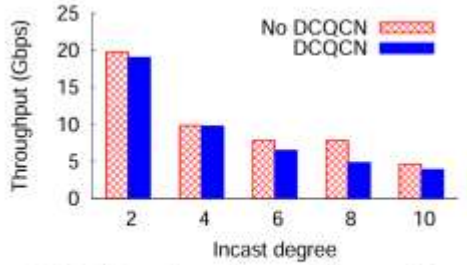
**Figure 14: DCQCN Parameters used in our datacenters**



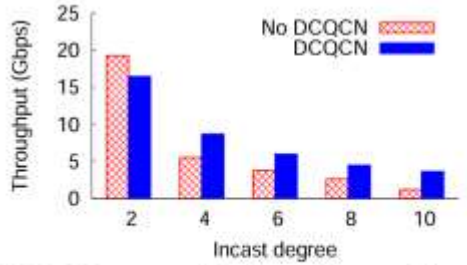

(a) Median throughput of user flows.




(b) 10th percentile throughput of user flows.



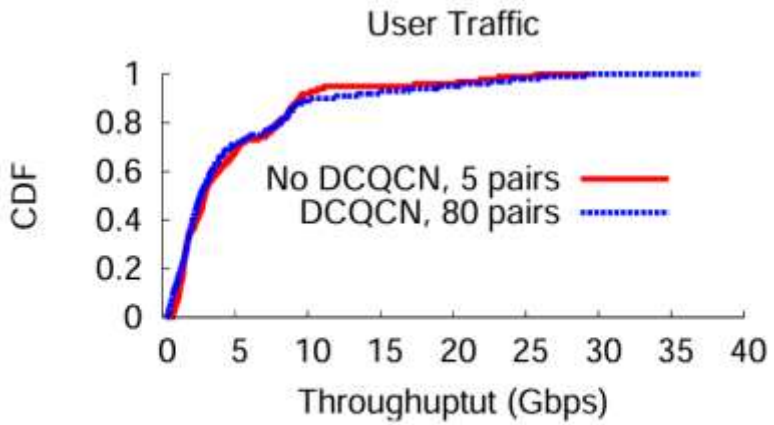

(c) Median throughput of incast flows.



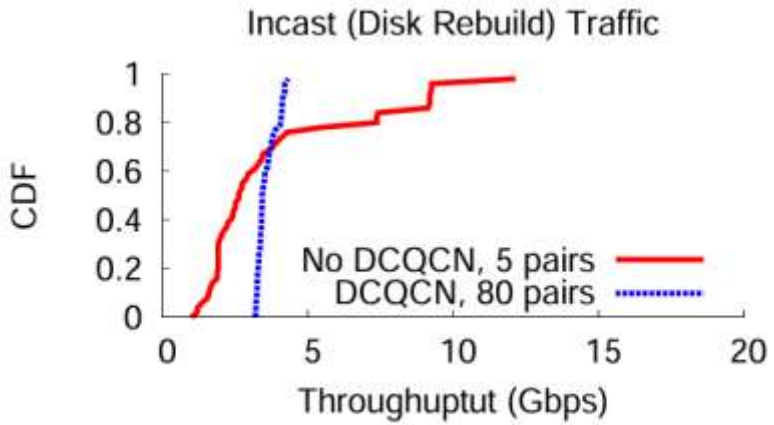

(d) 10th percentile throughput of incast flows.

# Results

- (a)：未启用DCQCN时5组通信对的用户流量性能，与启用DCQCN后80组通信对的性能表现相当。即采用DCQCN后，我们可处理16倍的用户流量且性能不衰减
- (b)：即便是磁盘重建流量的性能，在使用DCQCN时也比未使用时更均匀公平




(a) CDF of User traffic




(b) CDF of Incast traffic

# Results

- PFC的必要性与缓冲区阈值配置的重要性：

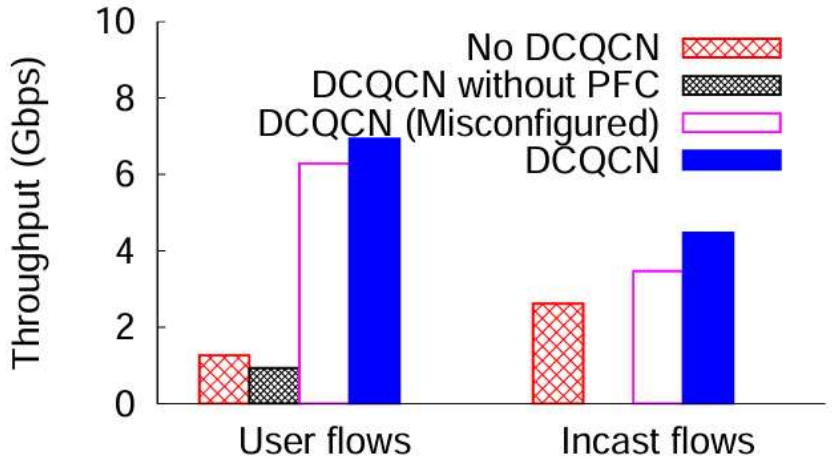



**Figure 18: 10th percentile throughput of four configurations for 8:1 incast**

- DCQCN还能有效降低延迟
- 采用DCQCN的队列长度显著短于DCTCP方案

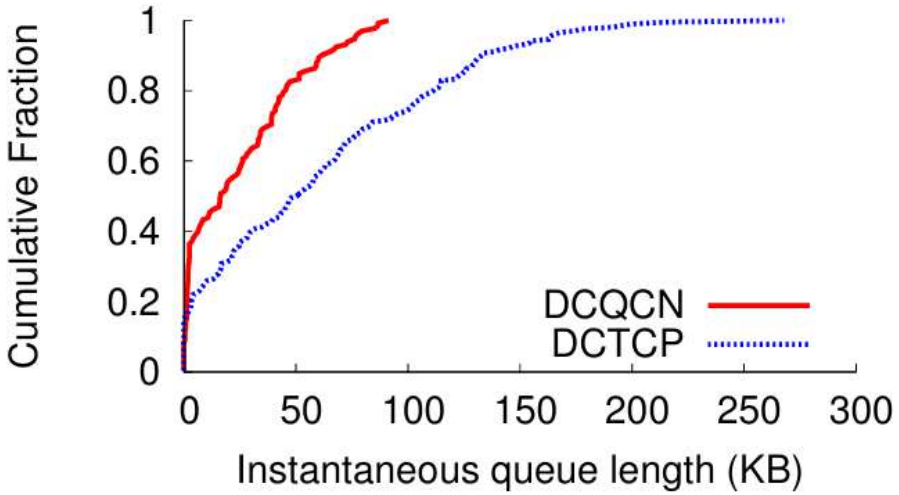



**Figure 19: Queue length CDF**

# Outline

I. Introduction

II. The Need For DCQCN

III. The DCQCN Algorithm

IV. Buffer Settings

V. Analysis OF DCQCN

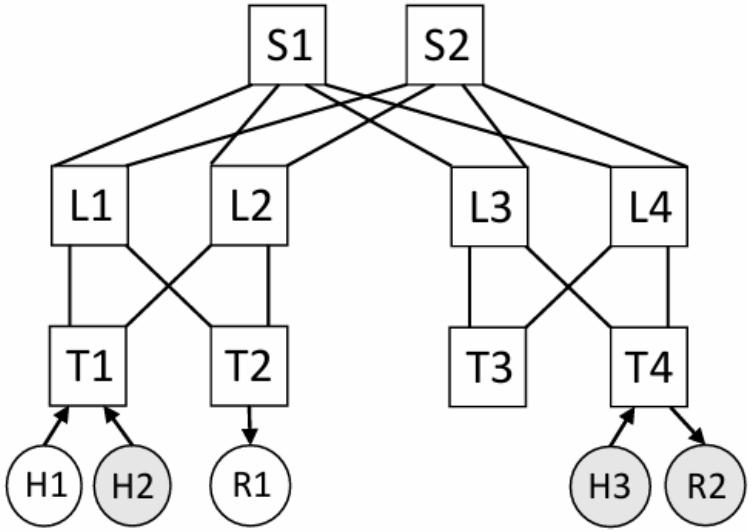
VI. Results

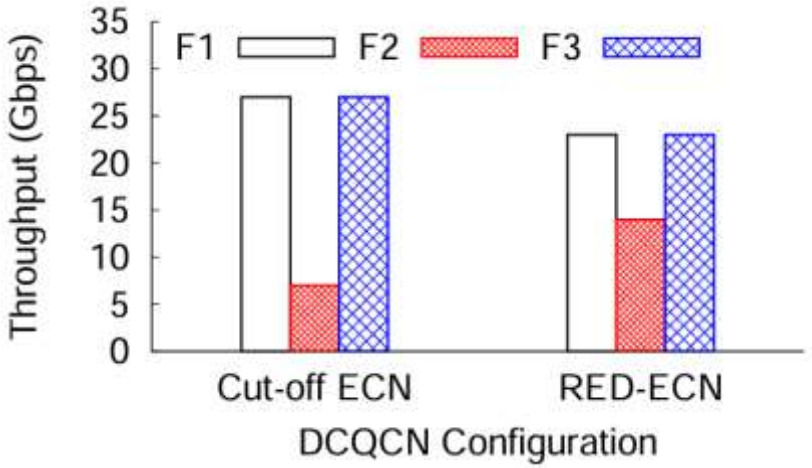
**VII. Discussion**

VIII. Review

# Discussion

- 多瓶颈（parking lot）场景下 DCQCN 的行为
  f1: H1 → R1
  f2: H2 → R2
  f3: H3 → R2
- 当 ECMP 把 f1、f2 都映射到 T1 的同一条上行链路时，f2 同时经过两个瓶颈（T1 上行、T4→R2），而 f1、f3 各自只有一个瓶颈。
- 按 max-min 公平，三条流都应拿到 20 Gbps，但常见拥塞控制（如 DCTCP）里，多瓶颈流收到拥塞信号的概率更高，反而会降到更低的速率
- 通过采用更平缓的RED-like标记方案，能缓解，但不能解决



# Discussion

- 非拥塞导致的丢包以及 RoCEv2 的丢包恢复机制
- PFC 能防止因缓冲区溢出导致的丢包，但在大规模网络中仍可能出现：包损坏端口间歇性故障交换机或服务器误配置 等情况引起的丢包
- 与 TCP 不同，RoCEv2 传输层假设丢包很少发生；例如使用 READ 操作时，协议不会告诉发送端数据是否正确到达
- 丢包是由接收端通过 序列号缺口发现，并由接收端发起恢复。ConnectX-3 Pro 使用的是简单的 go-back-N 重传机制。
- 在丢包率很低的环境下，这种简单机制是足够的，但 Figure 21 显示：一旦丢包率超过约 0.1%，吞吐量会快速下降

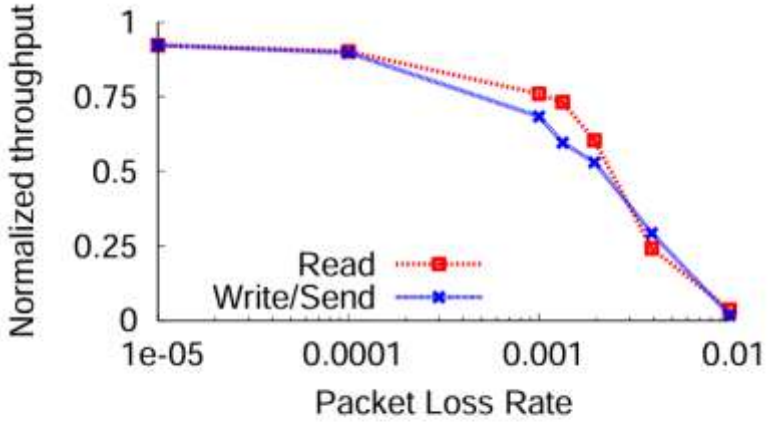
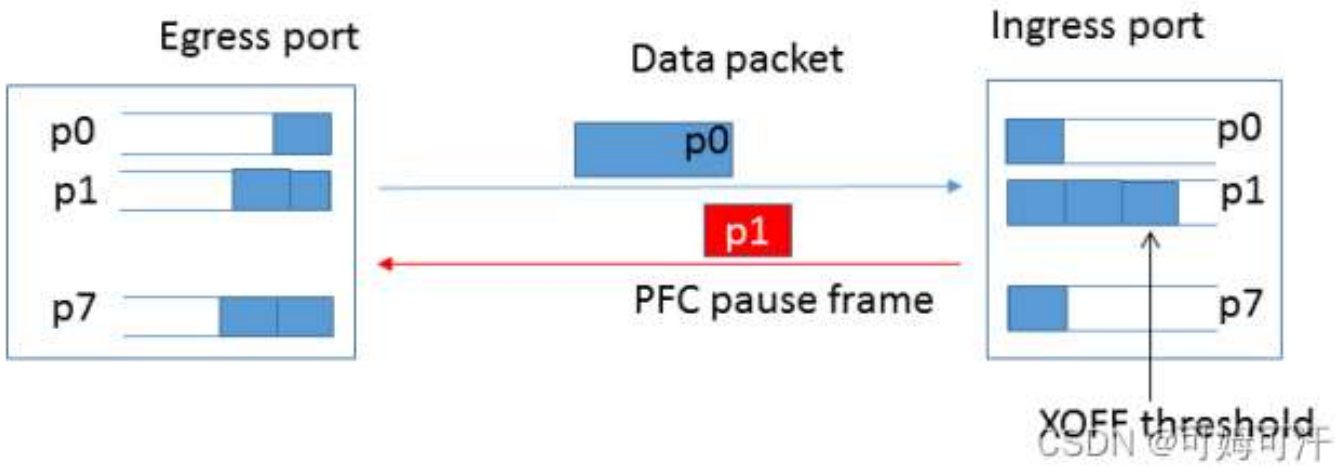



**Figure 21: Impact of packet loss rate on throughput**

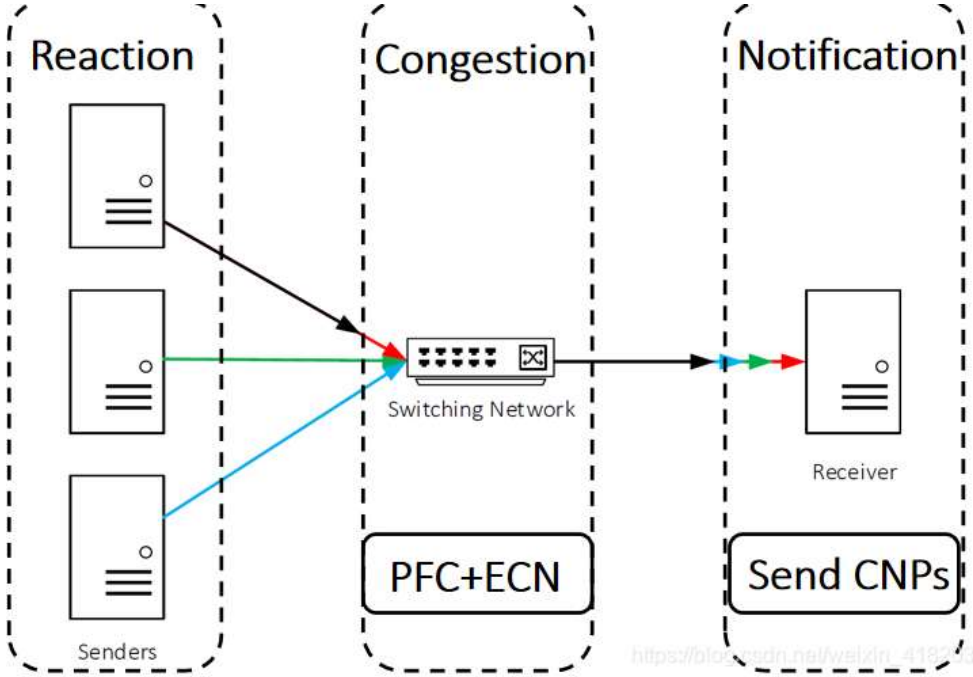
# Discussion

- PFC 是否会导致路由死锁
- 常见担忧是：PAUSE 机制会让交换机之间互相等待缓冲区，从而形成循环依赖，导致死锁
- 在典型的 Clos 结构数据中心中：服务器只接 ToR一对服务器之间的流量不会经过其他服务器，也不会穿过超过两个 ToR，所以在正常设备下，很难形成环形缓冲区依赖
- 因此，在没有故障或严重误配置的前提下，PFC 不会引发死锁



# Discussion

- DCQCN 没有把“对 TCP 友好”作为设计目标
- 论文没有研究 DCQCN 和 TCP 流量共存时的详细交互
- 在数据中心环境里，这不算硬需求，因为可以在交换机上通过 802.3 priority tag 把 TCP 和 DCQCN 流量隔离开：给不同类型流量配置不同的发送优先级不同的缓冲使用上限不同的速率限制
- 因此，可以在工程上直接把 TCP 与 RDMA/DCQCN 放到不同优先级队列中，避免互相影响



# Outline

I. Introduction

II. The Need For DCQCN

III. The DCQCN Algorithm

IV. Buffer Settings

V. Analysis OF DCQCN

VI. Results

VII. Discussion

**VIII. Review**

# Review

- 论文提出并实现了 DCQCN（Datacenter QCN），一种可在 大规模、L3 路由、无损 RoCEv2 数据中心网络中运行的端到端速率控制型拥塞控制协议，并已经在 Mellanox NIC 上落地实现
- 解决了 RoCEv2 依赖 PFC 所导致的 拥塞扩散、HOL 阻塞、不公平、吞吐量低下 等关键问题，使 RDMA 流量在高带宽场景下依旧能保持 高吞吐、低排队、低延迟和公平性
- 核心创新是：
  - 将 PFC 的快速、粗粒度保护与DCQCN的精细、慢速端到端调节结合
  - 使用硬件速率限制器做速率控制，实现比窗口机制更平滑的队列控制
  - 构建流体模型用于参数分析与稳定性调优
  - 提出适应大规模数据中心的 RED-ECN 配置方法，显著提升吞吐与公平性