A Network-centric TCP for Interactive Video Delivery Networks (VDN)

MD Iftakharul Islam, Javed I Khan

Department of Computer Science
Kent State University
Kent, OH

Outline

Congestion Control for Video Streaming

- Interactive video streaming such as Skype and Google Hangout use delay based congestion control.
- The delay-based congestion control however does not have accurate congestion information which results in:
  - High queuing delay in the router
  - Poor fairness which results in poor QoE
- Network centric TCP such as XCP, RCP and so on solve this problem by providing explicit feedback from router

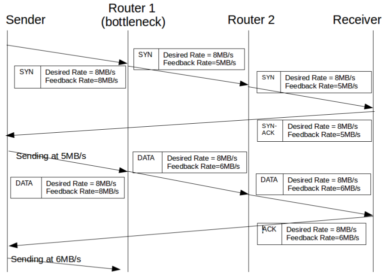


Figure: Work flow of NC-TCP

Interactive Video Delivery Network (VDN)

- XCP and RCP is however designed as TCP-friendly as they assumed the delay sensitive traffic can compete will loss sensitive traffic.
- Loss-based TCP flows results long queuing delay on the router (Bufferebloat problem)
- Solution:
  - Differentiated service and network slicing enable us to create exclusive video network where interactive video flows competes among themselves.
- We have designed a TCP for Interactive Video Network. We have shown that our newly designed TCP (NC-TCP) can perform better than XCP and delay based TCP in Video Delivery Network.

Why TCP?

- Video Streaming traditionally uses UDP. But UDP has many disadvantages:
    - It does not have any congestion control.
    - Application developers needs to implement their own congestion control.
    - Different implementation of congestion control interacts poorly with each other and stability of the network is compromised.
    - UDP is also not NAT and firewall friendly which are highly desirable attributes in today's Internet

Why TCP?

- TCP solves these problems, however TCP has some problems regarding interactive video streaming:
  - The ordered delivery of TCP delays some segment delivery if a prior segment is missing (head-of-line blocking problem).
  - Retransmission is also not required in interactive video streaming
  - These problems can be solved by implementing different TCP implementation. For example:
    - TCP Hollywood allows unordered segment delivery.
    - Retransmission can be avoided by extensive packet caching in the router.

Delay-based TCP

- Video streaming traditionally uses delay based congestion control.
- It modifies the congestion window based on RTT.
  - TCP Vegas, TCP FAST
- RTT based TCP loses throughput in the presence of reverse-path congestion.
- To solve this problem, LEDBAT uses one-way delay based congestion control.
- LEDBAT however suffers latecomer effect: second flow may starve first flow.
- One-way delay gradient has been used to overcome the late comers effect.
  - TCP CDG, TCP Inigo
  - Google Hangout (doesnt use TCP though)

Delay-based TCP

- Delay-based TCP controls the congestion window based on one-way delay observed in the receiver.
- It has no way of knowing about the actual queuing delay in the network. This is why, it faces several problems:
  - It cannot result near-zero queuing delay in the network.
  - It performs poorly in fairness. This is why some flows experience poor throughput which results poor video quality in video streaming.
- Recent advances in Software Defined Networks (SDN) can solve this problem

Network-centric TCP

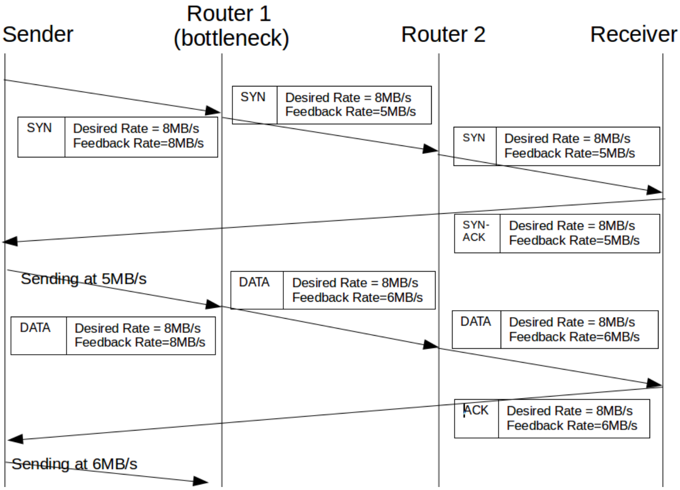


Figure: Work flow of NC-TCP

Network-centric TCP

- Routers play an active role in allocating throughput.
- NC-TCP uses rate-based congestion control rather than window based congestion control.
  - Window-based TCP produces bursty traffic. Rate-based congestion control shape the traffic before sending
  - It also ensures that the network does not have more traffic than it is able to handle.
  - We used **TCP-pacing**
- NC-TCP has been implemented as a new TCP option.

Type	Length
Expected Throughput	
Feedback Throughput	

Figure: NC option

NC-TCP Workflow

- Senders specify the required throughput in SYN and DATA segment.
- Routers along the path inspect the option header and allocate the feedback throughput.
- A router will set the feedback throughput only when the feedback throughput is smaller than the feedback throughput allocated by a previous router.
    - It ensures that feedback throughput is calculated based on the most congested link along the path
- Receiver copies the feedback throughput in the ACK segment.
- Sender sets the sending rate based on the ACK segment.

Feedback Throughput Calculation

- We have divided the feedback controller into two parts:
  - Delay Controller: *Proportional Integral (PI)* controller
  - Fairness Controller: Min-max fairness

Delay Controller

$$\sum r_i(t) = \alpha[c(t) - q(t)] \tag{1}$$

where the $\sum r_i(t)$ is the aggregate feedback throughput, $c(t)$ is the bottleneck link capacity, $q(t)$ is the queue length in bytes and α is the coefficient.

Fairness Controller

$$r_i(t) = \frac{\alpha[c(t) - q(t)]}{N(t)} \tag{2}$$

where $r_i(t)$ is the feedback throughput for flow i and $N(t)$ is the number of flows

Stability Analysis

- Let us assume that the feedback delay is t_f
- As the feedback rate becomes the sending rate after t_f,the sending rate at time t can be written as

$$x_i(t) = \frac{\alpha[C - q(t - t_f)]}{N} \tag{3}$$

- The queuing-delay gradient can be represented as

$$\dot{q(t)} = \sum x_i(t) - C \tag{4}$$

- As flows adapt their sending rate $x_i(t)$ at the same rate, $\sum x_i(t) = Nx_i(t)$ for all i.
- $$\dot{q(t)} = Nx_i(t) - C \tag{5}$$
- $$\dot{q(t)} = -\alpha q(t - t_f) + C(\alpha - 1) \tag{6}$$

Stability Analysis

- $\dot{q}(t) = -\alpha q(t - t_f) + C(\alpha - 1)$ is an autonomous differential equation
- The autonomous system is stable if $\ddot{q}(t^*) < 0$ where $q(t^*)$ represents the queuing delay at the equilibrium point
- This is why the system is stable if $\alpha > 0$.
- If the system is perturbed, the queue will be drained at $-\alpha q(t - t_f) + C(\alpha - 1)$ rate and eventually reaches the equilibrium point
- At the equilibrium point, $\dot{q}(t) = 0$. So we get

$$q(t - t_f) = \frac{C(\alpha - 1)}{\alpha} \tag{7}$$

- $\frac{C(\alpha-1)}{\alpha}$ is the queuing delay at the equilibrium point.
- Here we want the queuing delay to be be zero while maximizing the throughput. That is why we set $\alpha = 1$.

Encoder Rate Control

- The encoding rate of video cannot be changed quickly in order to avoid congestion (takes more than 500ms)
- The actual encoder output rate also fluctuates randomly around the input target rate.
- NC-TCP based application sets the target rate based on the feedback rate of the network. (Motivated from iTCP)
- As NC-TCP uses TCP-pacing, packets wait in the TCP's sending buffer (congestion window) to be scheduled to sent out
  - RTT produced by the NC-TCP is not exactly same as the propagation delay
  - RTT/RTT_{min} indicates how much encoder has overshot with respect to the network throughput. Here RTT_{min} is the minimum RTT which is considered as the *propagation delay*
- We set the encoder target rate as $R_t(t) = \frac{r_i(t)}{\frac{RTT}{RTT_{min}}}$ where $r_i(t)$ is the feedback throughput

NC-TCP Implementation

- We have implemented NC-TCP in Linux kernel. Source: https://github.com/tamimcse/Linux
- NC-TCP implementation has two parts: NC-TCP host and NC-TCP router
- We have modified the Linux TCP stack to implement the NC-TCP host.
- We have implemented the NC-TCP router as a Qdisc kernel module in Linux router (*ip_forward=1*).
- We also modified the kernel to trace system variables such as throughput, queue length, and so on.

Video Streaming Application

- We also developed a TCP based video streaming application based on GStreamer.
- https://github.com/tamimcse/gst-streamer (350 lines of C code)
- The application reads a video file, encode in H.264 constant bit rate.
- The bitrate is set by the feedback rate received from the TCP stack. It uses It uses getsockopt system call.
- The application stream the video frames to the client. Client displays the frames as soon as it receives it.

Experimental Setup

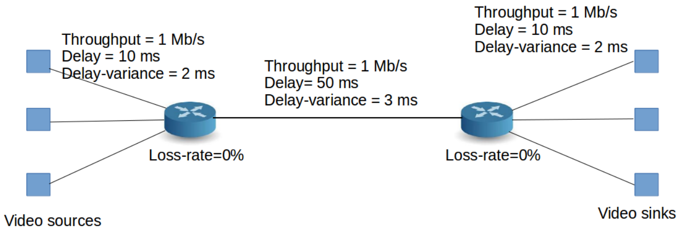


Figure: Topology

- The topology is created in Mininet.
- All the nodes in the picture is a Linux container (light-weight VM)
- All the links are virtual ethernet (veth) pair
- The delay and throughput on the links are set by **NetEm** and **htb** (hierarchical token bucket) Qdisc.
- The setup takes 200 lines of Python code. We also used Shell Scripts extensively.

Experiment

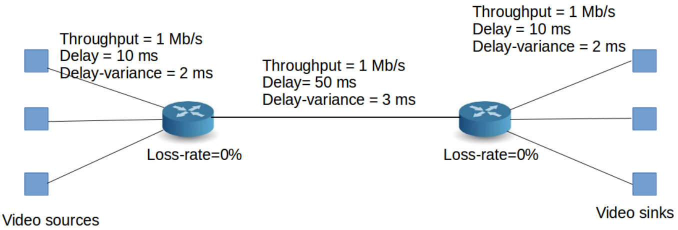


Figure: Topology

- We streamed a clip from **Big Buck Bunny** in our experiment.
- The sender produces *512X340* video at 30 fps in H.264 format
- The three senders stream video simultaneously to the three receivers
- The experiment is conducted for 80 seconds.

Evaluation

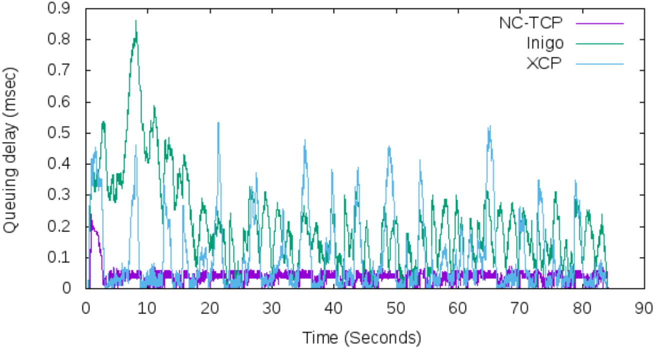
- Loss based TCP such as TCP CUBIC results in long queuing delay.
- Recently proposed Google BBR also results in long queuing delay.
- We compared NC-TCP to **TCP Inigo** and **XCP**
- We found TCP Inigo is a representative of one-way delay gradient based TCP.
- The workflow of XCP is similar to NC-TCP. But it has been designed for short-lived flows.
- We have found the implementation TCP CUBIC, TCP BBR and TCP CDG in Linux kernel.
- The Linux kernel implementation of TCP Inigo is also shared by it's author. We also have implemented the XCP in Linux kernel.
- We compared the **queuing delay** on the bottleneck router for different protocols.
- We also have compared the **fairness** for each protocol. Note that higher fairness indicates that all the flows get higher throughput from the bottleneck link

NC-TCP vs XCP

- **XCP** (eXplicit Congestion Control) is the first network-assisted congestion control algorithm where explicit rate is allocated by the router.
- XCP makes no assumption regarding the traffic characteristics whereas NC-TCP utilizes the traffic characteristics to optimize the throughput allocation.
- XCP uses window-based congestion control whereas NC-TCP uses rate based congestion control.
- XCP uses persistent queue size, the minimum queue size during a control interval. NC-TCP however uses instant queue size. This enables NC-TCP to reacts to congestion quicker than XCP.
- NC-TCP feedback throughput is calculated using $\frac{\alpha[c(t)-q(t)]}{N(t)}$ whereas XCP feedback is $\frac{\alpha RTT[c(t)-\sum x(t)]-\beta q(t)}{N(t)}$ where RTT is the average RTTs of all the flows and $\sum x(t)$ is the aggregate incoming-rate to the switch.

Bottleneck Queuing Delay
Single-bottleneck Topology

- The queuing delay is $\frac{q(t)}{B}$ where B is the bottleneck throughput.



Throughput

Single-bottleneck Topology

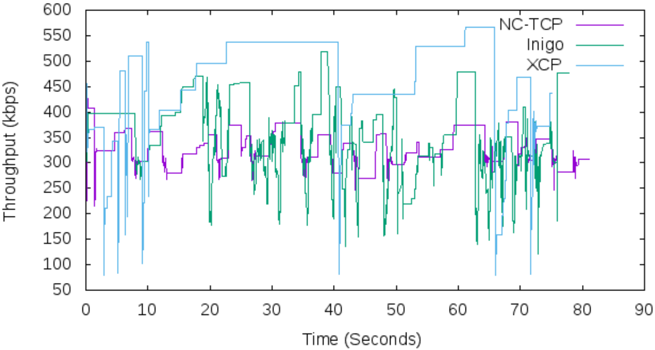
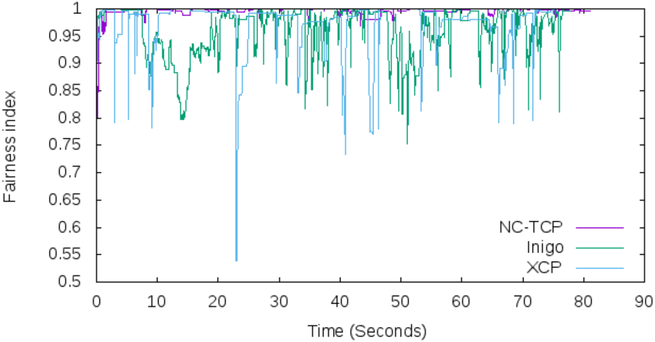


Figure: Throughput

Fairness Index

Single-bottleneck Topology

- The fairness index has been calculated using the formula $\frac{(\sum_{i=1}^{n} x_i)^2}{n\sum_{i=1}^{n} {x_i}^2}$ where x_i is the throughput of flow i and n is the total number of flows



Round-trip time

Single-bottleneck Topology

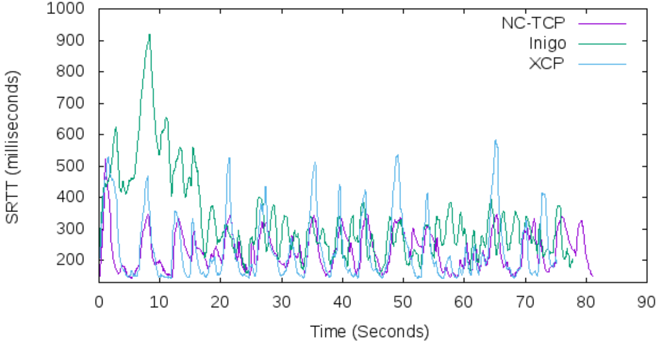


Figure: RTT

Self-fairness

Three TCP Inigo flows 15 seconds apart

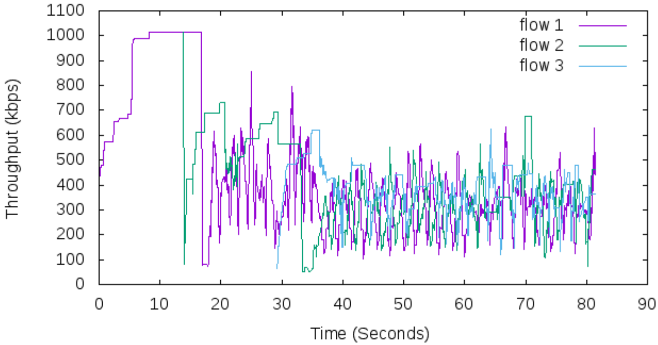


Figure: Three TCP Inigo flows 15 seconds apart

Self-fairness

Three NC-TCP flows 15 seconds apart

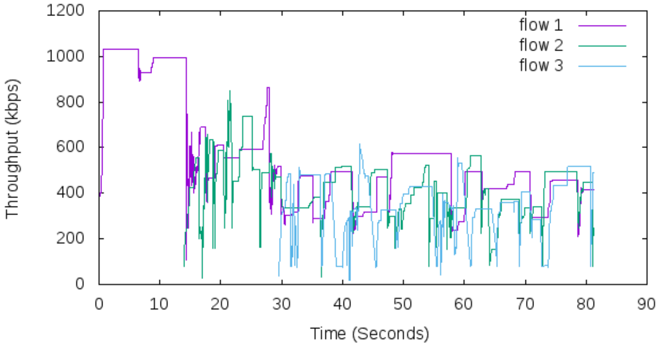


Figure: Three NC-TCP flows 15 seconds apart

RTT-fairness

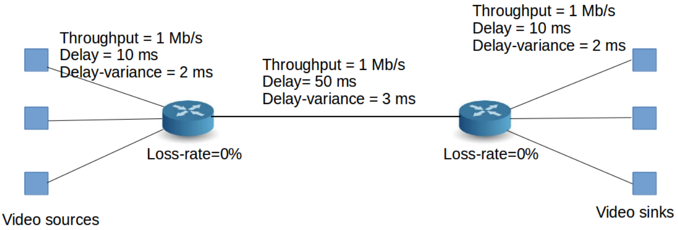


Figure: Topology

- We modified the topology such that each flow has different RTTs: **80ms**, **100ms** and **120ms**

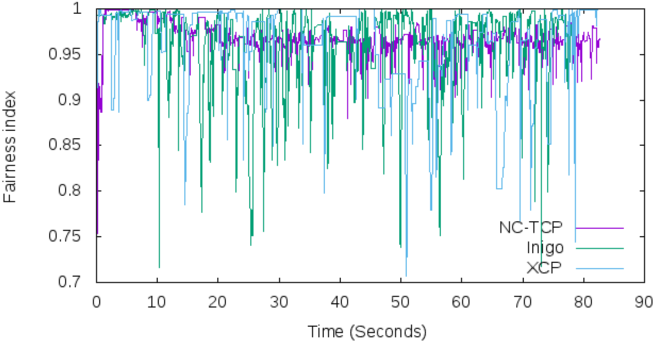


Figure: Fairness Index

Queuing delay
RTT-fairness

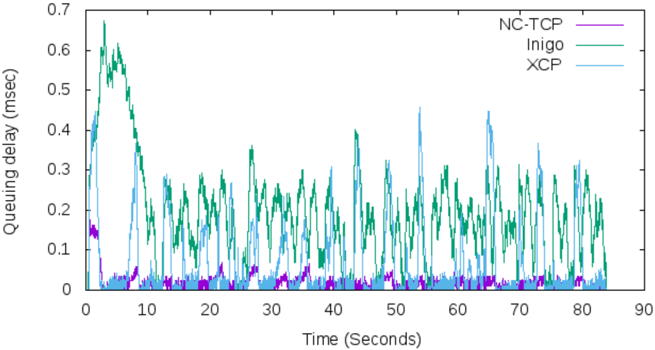
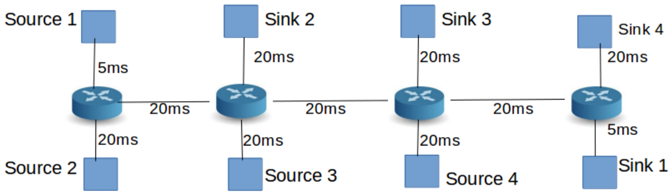


Figure: Queuing delay in the bottleneck router

Multiple bottlenecks topology

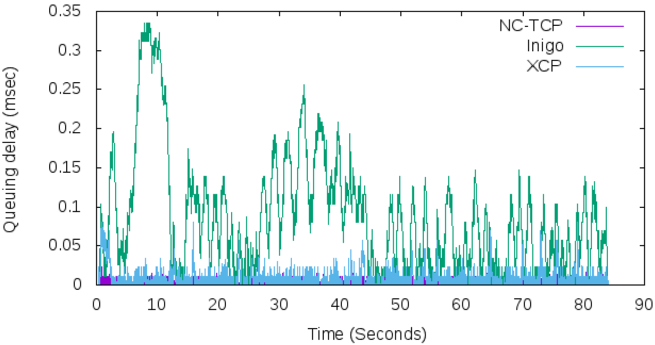


Flows:
Source 1 ===> Sink 1
Source 2 ===> Sink 2
Source 3 ===> Sink 3
Source 4 ===> Sink 4

Figure: The throughput of bottleneck and access links are 660kbps and 1024kbps respectively

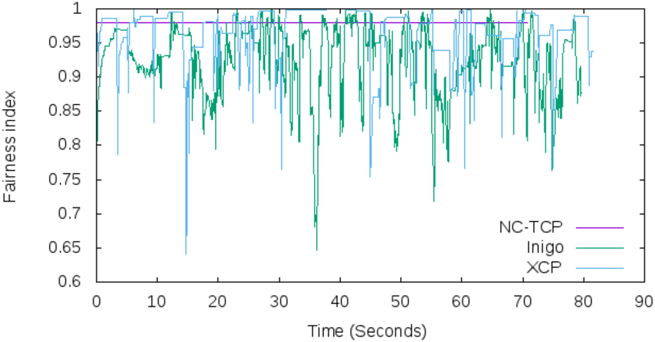
Queuing delay

Multiple bottlenecks topology



Fairness Index

Multiple bottlenecks topology



Differential Fairness

- Interactive VDN needs to support heterogeneous endpoints such as smart phones, laptops, TV and VR headset at the same time
- The videos used in different endpoints often differ in encodings, resolutions and frame rates.
- Network-centric congestion control has a major advantage over host-centric approach in this regard
- As a router is aware of the throughput requirement of each flow passing through it, it can apply differential or weighted fairness in throughput allocation

Visual Interruptions

Table: Number of visual interruptions

	Single bottleneck topology	Multiple bottleneck topology
TCP Inigo	2	20
XCP	0	0
NC-TCP	0	0

Thank You