ACCT: An Intelligent Congestion Control Mechanism for Future Internet

Morteza Kheirkhah
(Samsung Research UK)

Joao Reis, Fan Yang, David Griffin and Miguel Rio
(University College London)

7th December 2023
IEEE GLOBECOM Conference

Motivation

- Traditional end-to-end congestion control schemes (e.g., TCP and its variants) can not meet the requirements of emerging Internet applications (e.g., VR, AR, XR, immersive media, etc.)
  - Highly diverse with conflicting requirements (High-bandwidth, low-latency, high-reliability)
- These schemes primarily design to achieve high network throughput with less regard about the end-to-end latency
- Applications with latency constraints rely on the network to meet their requirements by means of traffic engineering (segregating traffic/flows)
  - 5G network slicing, priority queue (**partially solving the problem**)

Motivation (cont.)

- The problem is aggravated over wireless networks.
  - Wireless channels fluctuate rapidly and largely, occasionally, in the order of milliseconds due to user mobility, interference from adjacent cell towers, changes in the number of active users
- Worse with emerging wireless technologies operating at high-frequency bands (e.g., mmWave and above)
  - Sensitive to the environment due to their signal propagation characteristics
- Even BBR, the state-of-art end-host-centric congestion control scheme, fails to track such capacity changes precisely

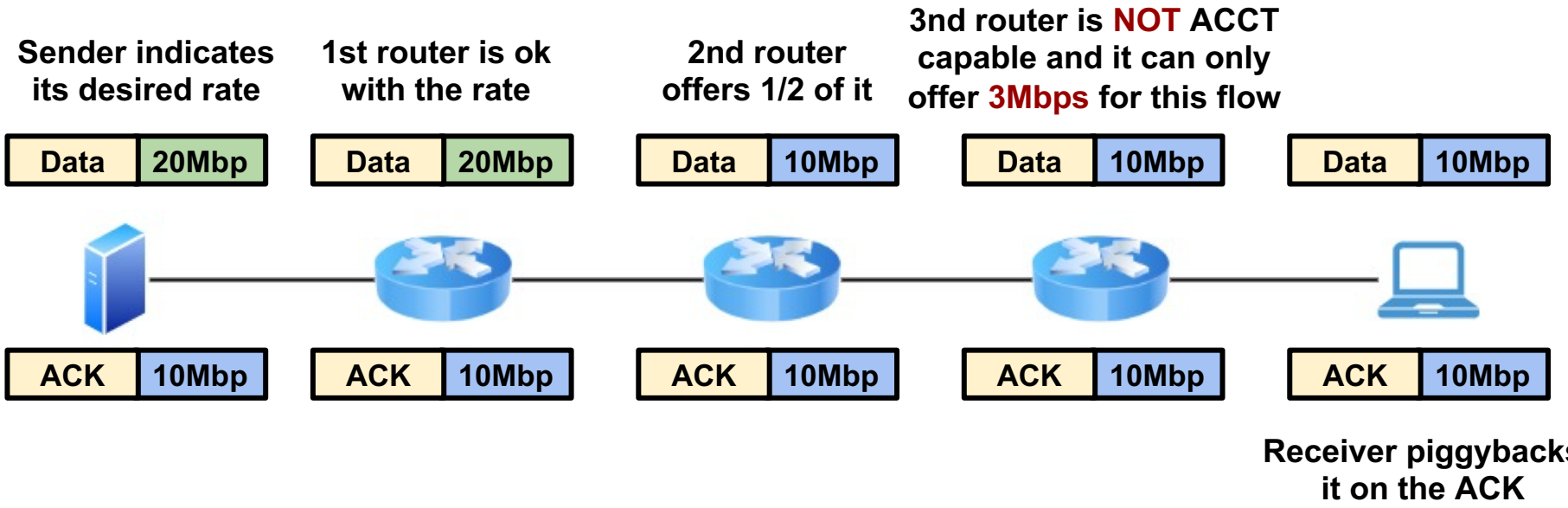
Motivation (cont.)

- A possible solution is to move towards network-centric approach (XCP, RCP)
- Each sender expresses its desired sending rate and other feedback information such as the RTT measurement to routers
- Each router along the network path provides multiple bits of feedback per packet to senders
- Senders should completely follow whatever the network says
- **Two serious problems**:
  - Does not scale well as routers need to perform complex operation per-packet
  - All network elements in all networks along the path should support the proposed signalling

ACCT

- An end-to-end, learning-based, rate control mechanism for Internet flows that is capable of considering **application** and **network** requirements simultaneously
- The ACCT sender is capable of exploiting **explicit** feedback from the network about the available capacity and also from the application about the desired sending rate
- The ACCT sender does not **blindly** adjust its sending rate to either of these signals
- ACCT designs a set of algorithms at end-hosts and routers to achieve its objectives
    - It places all complex tasks at **end-hosts**

ACCT illustration



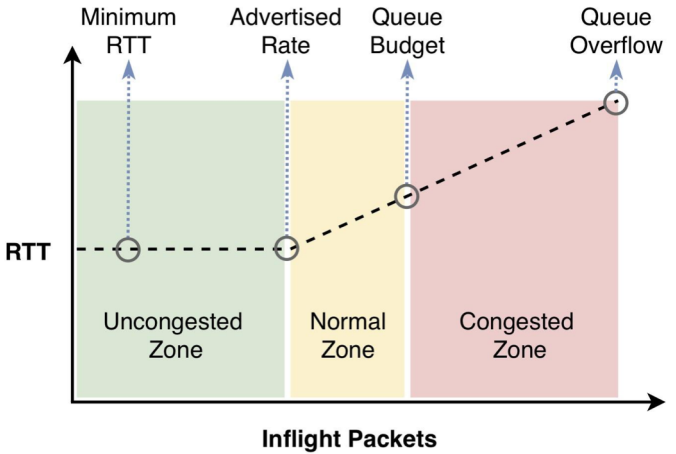
ACCT detects that the 3rd router is the bottleneck
it then regulates its sending rate on that basis

ACCT’s three main components

- **ACCT Sender.** An end-to-end congestion control algorithm that operates at the transport layer
- **ACCT RL Agent.** A RL agent that is responsible for adjusting the aggressiveness of ACCT flows at the ACCT sender’s side
- **ACCT Network Agent.** A distributed subsystem that resides at the network routers

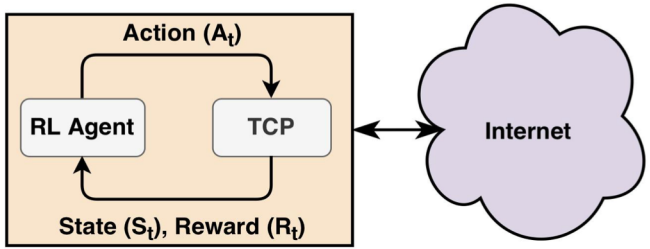
ACCT Sender

- It splits its operating zone into three distinct zones based on the explicit feedback it receives from the network and its measurements the queuing delay
- The key intuition behind this zone separation is that ACCT needs to adopt different behaviours/strategies in each zone



ACCT RL Agent

- The ACCT Sender regularly informs the RL Agent of the current operating zone (as an state input)
- In each operating zone, the ACCT Sender follows different reward functions to calculate the reward value
- The RL Agent proposes an action accordingly (which dictates the aggressiveness of the ACCT flow)



RL formulation

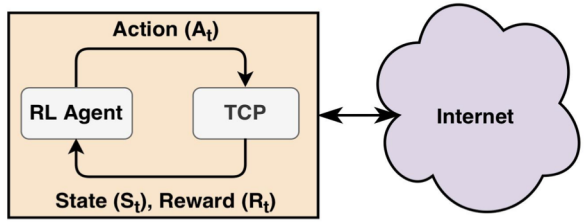
$$A_i = [1,\ 2,\ 3,\ 4,\ 5,\ 10,\ 20,\ 30,\ ...,\ 80,\ 90,\ 100]$$

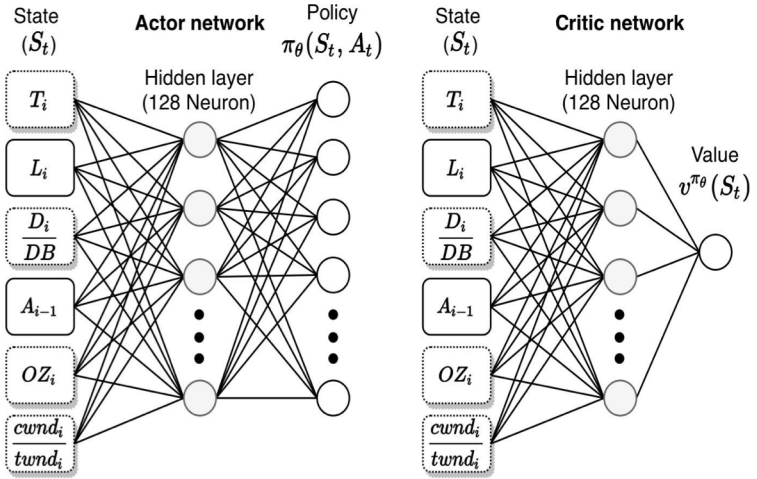
$$S_i = \{[T_i],\ L_i,\ [\frac{D_i}{DB}],\ A_{i-1},\ [Zone_i],\ [\frac{cwnd_i}{twnd_i}]\}$$

$$R_G = A_{i-1} - \omega L_i$$

$$R_Y = (as[N-1] + 1 - A_{i-1}) - \omega L_i - \zeta B_i$$

$$R_B = (as[N-1] + 1 - A_{i-1}) - \beta \frac{D_i}{DB} - \omega L_i - \zeta B_i$$





ACCT Network Agent

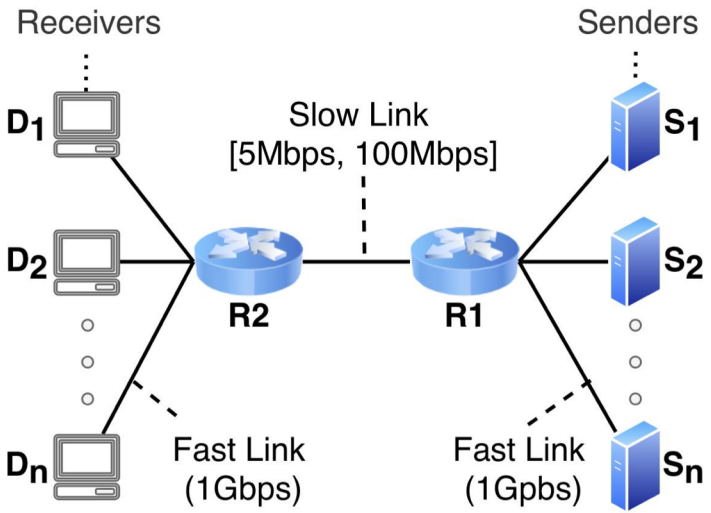
- The ACCT Network Agent follows a simple rate distribution scheme in which it distributes the available capacity across active flows
- It first estimates an equal distribution of rate between active flows
  - ES = total capacity / number of active flows
- The spare capacity of those flows with a smaller desired rate than the equal split (ES) is then allocated to flows with a higher desired rate than the ES

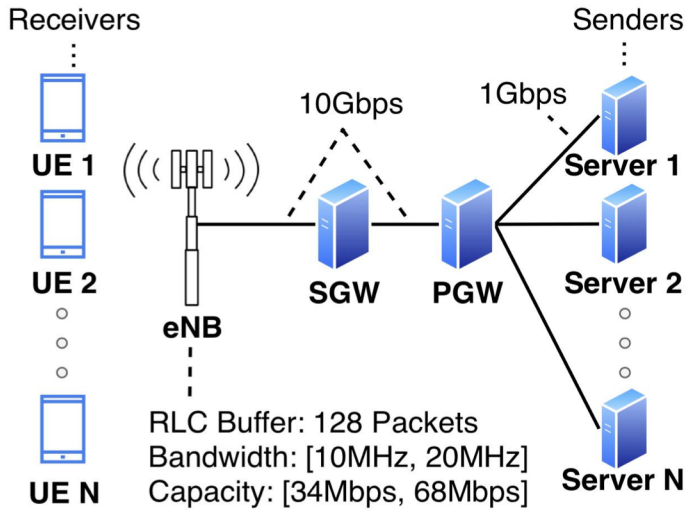
Implementation

- We implemented ACCT in NS-3 (C++)
- Our implementations support ACCT in both wired and wireless networks
- For wired networks, we modified the IP layer of the NS-3 networking stack to model ACCT-enabled routers
- For wireless networks, we modifying the PDCP and MAC layers of the LTE cellular networking stack
- We also modified the TCP modules to support ACCT signalling and implemented a new congestion control module for ACCT
- Our congestion module interacts with the ACCT RL agent via a technology agnostic protocol (Google Protocol Buffers). We wrote the RL agent (A3C) in Python

Simulation setup

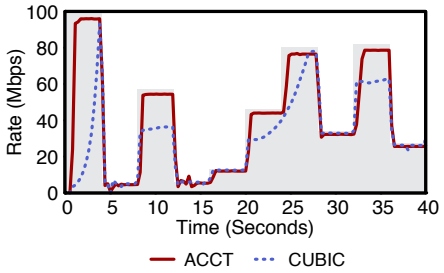
- The queueing budget is 5ms
- TCP send and receive socket buffer size was 2MB



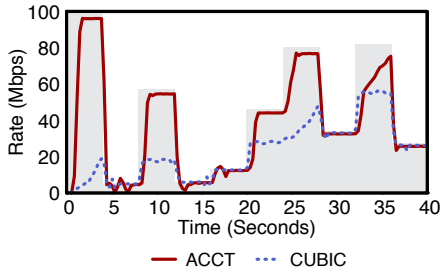


Evaluation (efficiency and responsiveness)

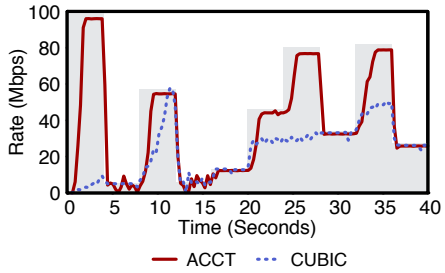
- Every 4 seconds we randomly changed the maximum capacity of the link between routers (in our wired network), ranging from 5Mbps to 100Mbps
- We turned off the slow start mechanism during the connection start-up



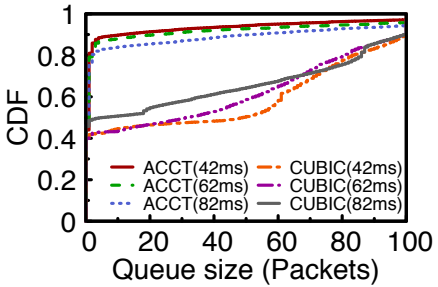
RTT: 42ms, BDP: 525KB



RTT: 62ms, BDP: 775KB

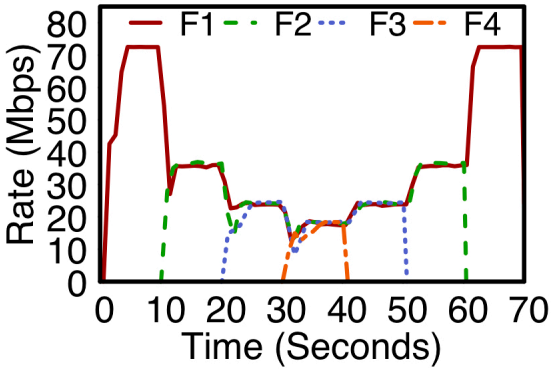


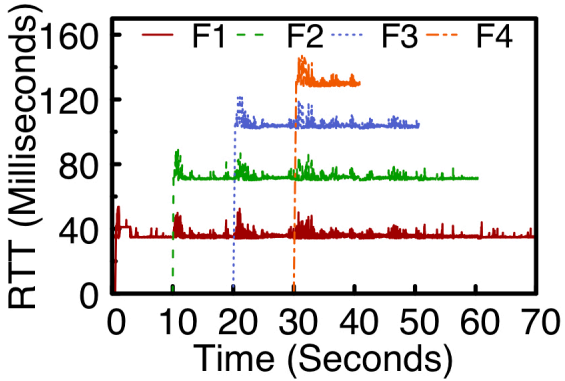
RTT: 82ms, BDP: 1MB



Evaluation (fairness)

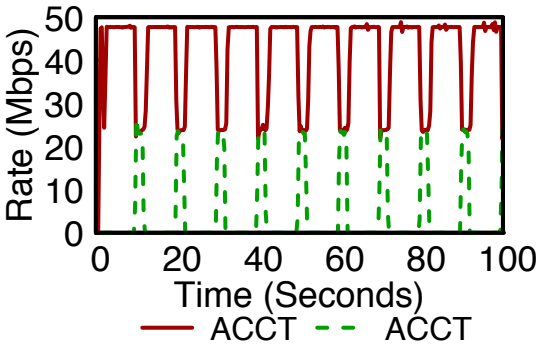
- Initially, 4 ACCT flows (each with different RTT) arrive at the UE with an inter-arrival time of 10 seconds, making the base station a bottleneck
- After 40 seconds, the flows leave the bottleneck again with 10 seconds gap between each flow's departure

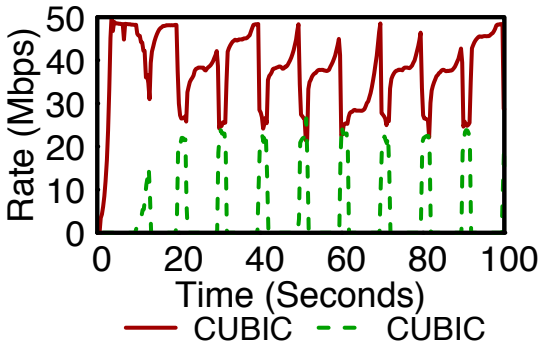


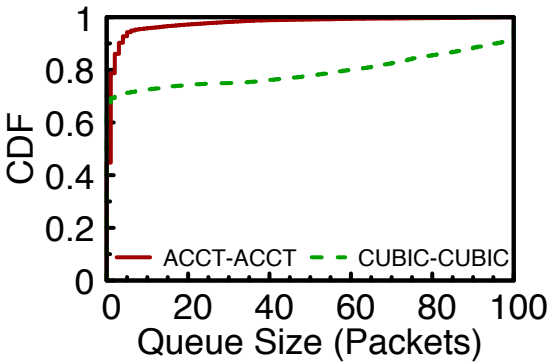


Evaluation (coexistence)

- An ACCT flow carrying file download traffic competes with another ACCT flow carrying video traffic
- Video traffic follows a ON/OFF pattern with 2-seconds ON and 8-seconds OFF

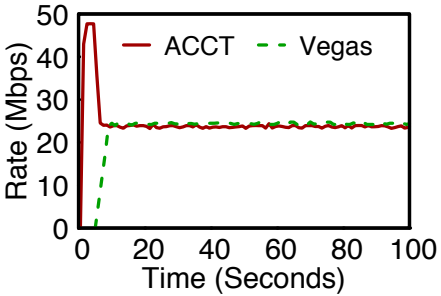


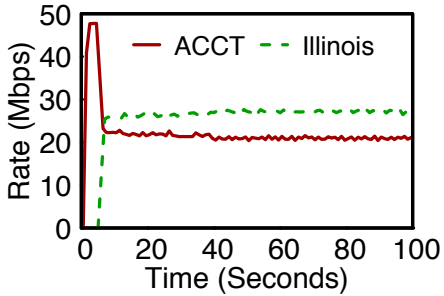


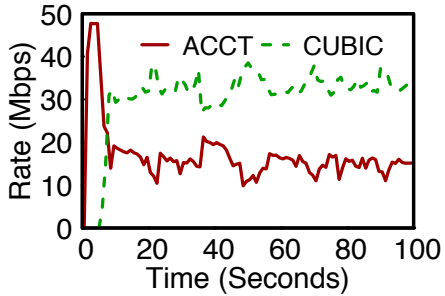


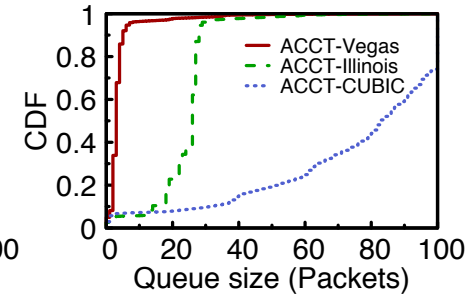
Evaluation (TCP-friendliness)

- Network fairness is preserved when ACCT competes with delayed based variants of TCP (e.g., Vegas, Illinois)
- When competing with CUBIC, it achieves its fair-share of resources but the end-to- end delay is compromised by CUBIC









Summary

- ACCT maximizes throughput and minimizes delay (within a small budget)
- ACCT converges quickly to its fair-share within an RTT
- ACCT has no bias against long RTT flows
- ACCT detects non-ACCT bottlenecks
- ACCT coexists well with non-ACCT flows

Quick links

- **Homepage**: https://www.uclmail.net/users/m.kheirkhah/
  - Everything about ACCT can be found here
- **Email**: morteza.kheirkhah@uclmail.net
  - Feel free to ask your questions
- **GitHub**: https://github.com/mkheirkhah/
  - I will add ACCT's source code here
- **LinkedIn**: https://www.linkedin.com/in/mkheirkhah/
  - Feel free to connect ☺