Special Topics: Machine Learning (ML) for Networking

COL867

Holi, 2025

Resource Allocation

Tarun Mangla

Hands-on Exercise

- Download data

- Git clone: git@github.com:tarunmangla/col867-holi25.git

Case Studies: ML for Specific Network Learning Tasks

- Application Classification
- Application Performance Monitoring
- Security
- **Resource Allocation**

Data center N/w : Multiple tenants
& n/w
B/w allocation

Resource Allocation

- Load balancing
- Traffic engineering
- Rate control → TCP Congestion control
- Quality of Service (QoS) management
- Network slicing
- …

Rate Control (at End-host)

- Transport layer : TCP cc
- Application layer : DASH / HTTP/2
  Video conferencing

TCP Congestion Control Basics

- **What**: decide the rate to send the data

- **Why**:
  - Prevent congestion collapse
  - Fair and efficient sharing of bandwidth

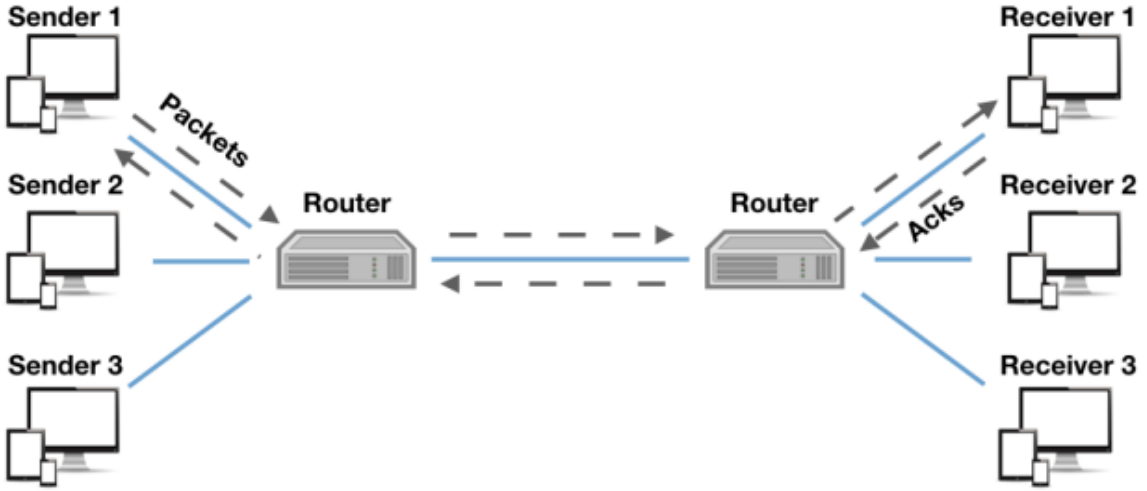


Figure 1: Multiple traffic flows sharing a link

- **How:**
  - End-to-end
  - Network-assisted : ECN or explicit congestion notification

Congestion Control Heuristics

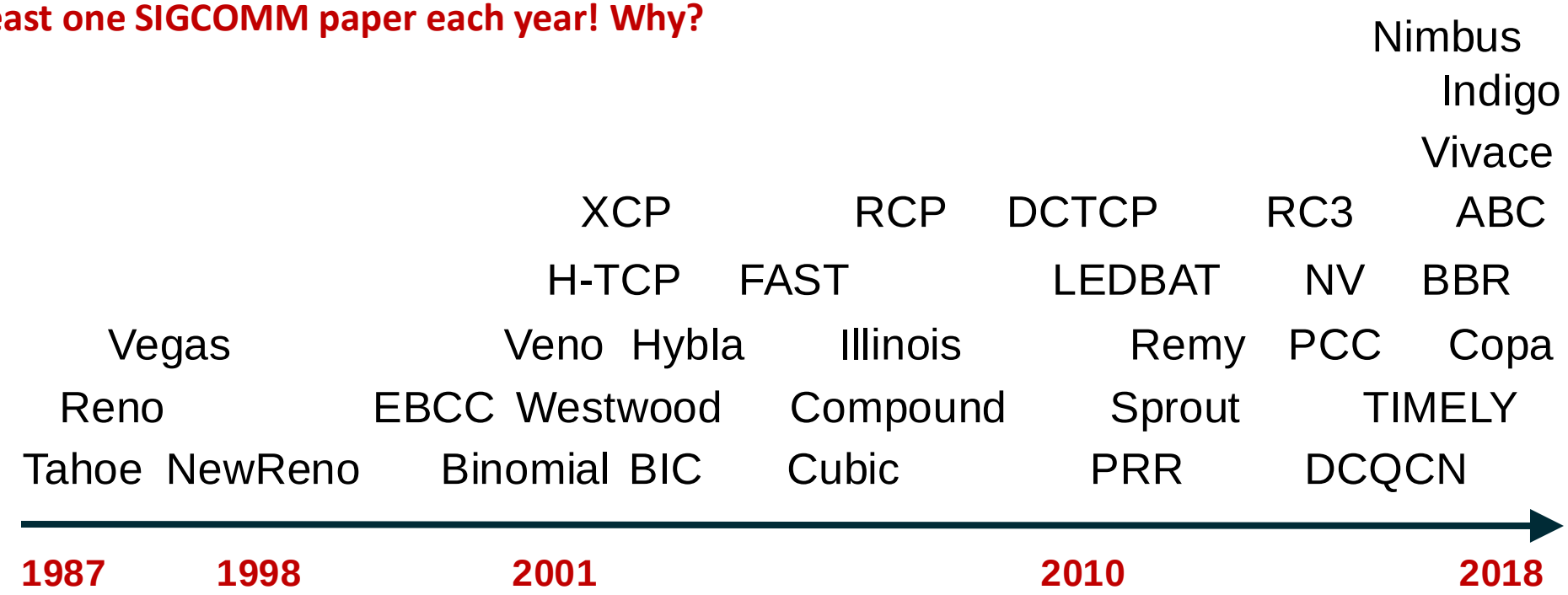
- What are the signals of congestion?
  - Packet loss
  - Increase in packet delay

- Loss-based: Tahoe, Reno, BIC, CUBIC ….
- Delay-based: TCP Vegas, TCP BBR* …

- Beyond this, CCA suggested for specific scenarios

Congestion Control Research Timeline

Still at least one SIGCOMM paper each year! Why?



Increasing bandwidth

Improving deployed algorithms

Changing application requirements

Different deployment scenarios

Using new algorithmic tools

Congestion control is a hard problem!

- Networks are complex
  - Non-congestion losses
  - High—RTT cross-continent links
- Networks are evolving
  - Increasing capacity
- Diverse networks
  - WiFi, LTE links
  - Data center networks
  - Satellite networks LEO / Starlink
- Diverse applications
  - Characterized by high flow churn →
  - Application require low latency
- …

Can we use ML?

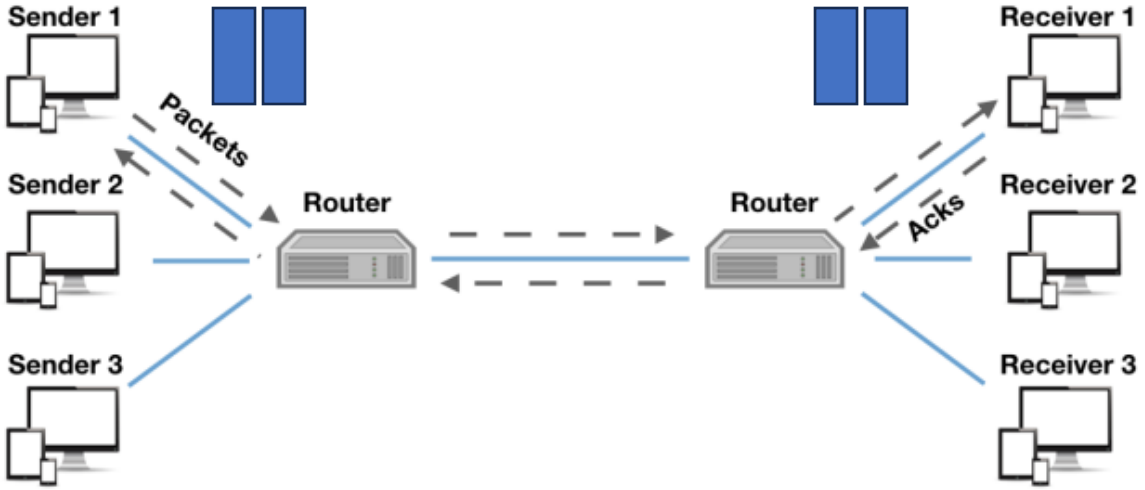
A hand-crafted solution is simply not enough

Why ML is a potential approach for CC?

When a packet is sent
Whether a packet is lost
When a packet is acked

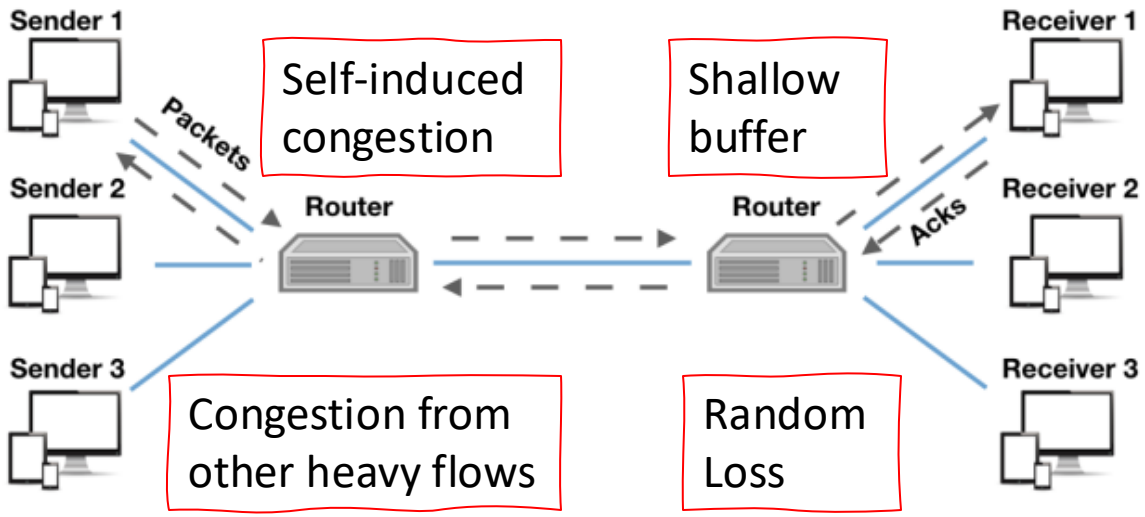
CUBIC assumption: Loss indicator of congestion

Change Sending Rate



Traditional approaches are based on strong assumptions about the network

Strong Assumptions Cause Problem



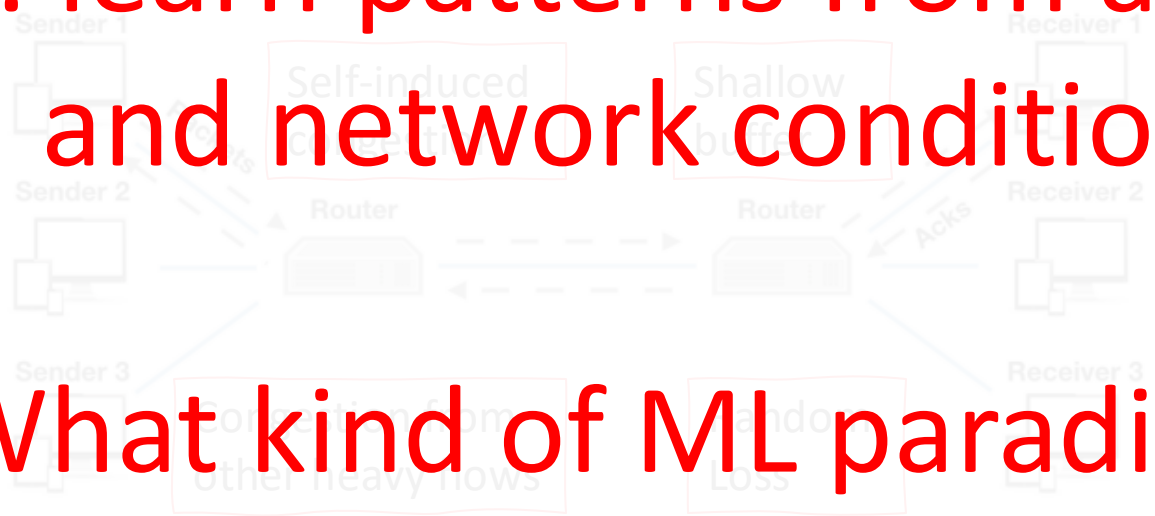
The best response can differ based on the underlying cause of packet loss.

BUT..

TCP CUBIC has the same response across all cases

Strong Assumptions Cause Problem

Solution: learn patterns from actual traffic and network conditions



What kind of ML paradigm?

The best response can differ based on the underlying cause of packet loss.

Use Reinforcement Learning

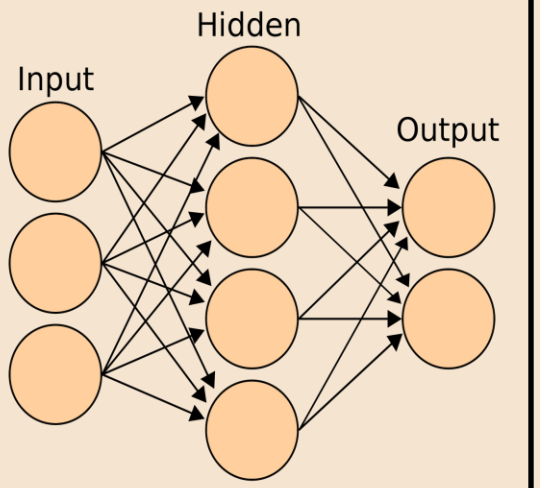
TCP Aurora (ICML19)

TCP CUBIC has the same response for all cases

Aurora Design

State s_t

Agent



Reward

$$x_t = \begin{cases} x_{t-1} * (1 + \alpha a_t) & a_t \geq 0 \\ x_{t-1}/(1 - \alpha a_t) & a_t < 0 \end{cases}$$

Action a_t

Environment

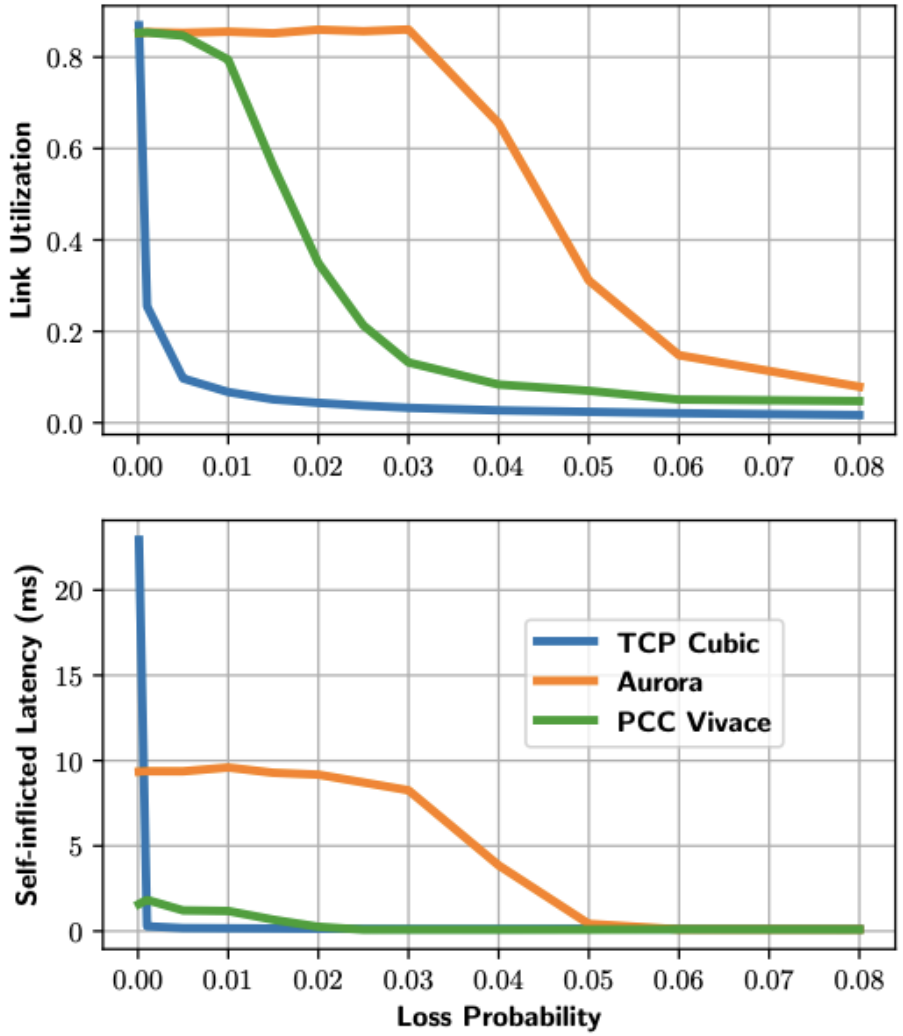
Training

- Simulate network links with a range of parameters
- Use open-source OpenAI gym environment
- Training environment:
  - Single traffic source

Bandwidth	Latency	Queue size	Loss rate
100-500 pps	50-500ms	2-2981 packets	0-5%

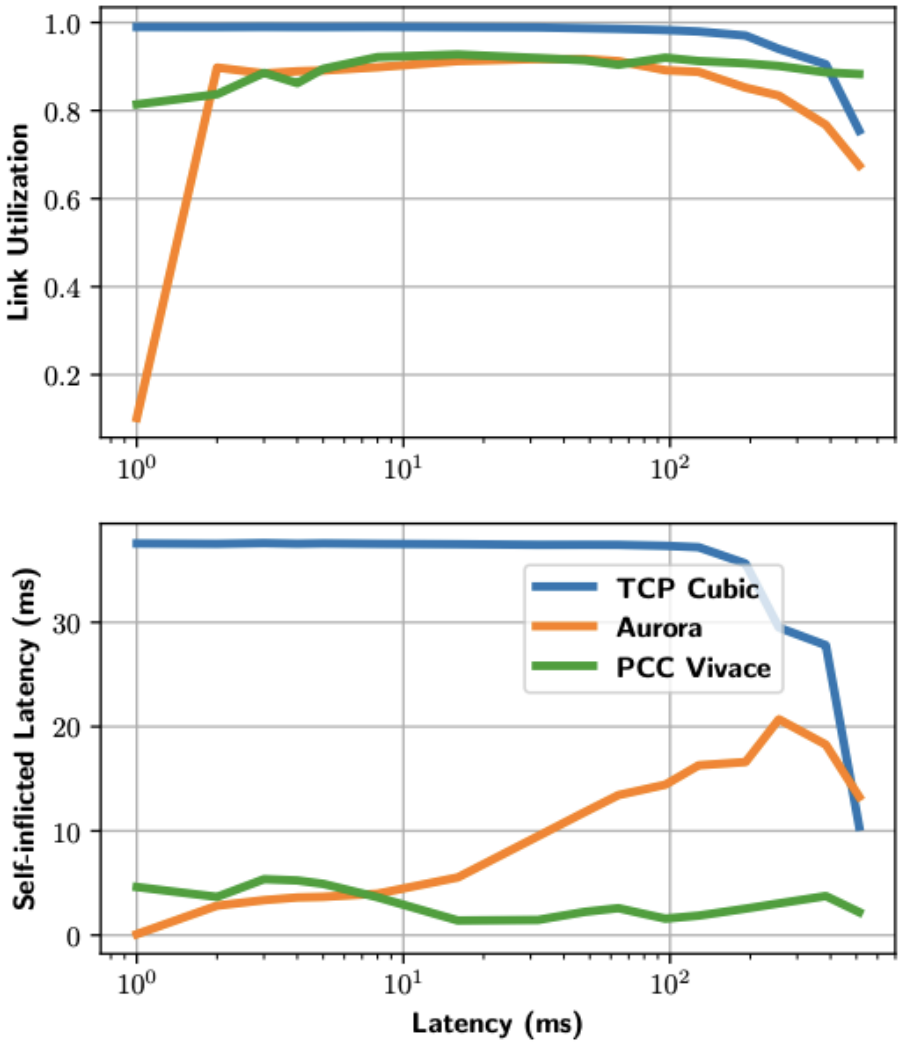
Results: Robustness to Loss

- Training only on loss rate of 0-5%
- Why does TCP Cubic loss utilization drop drastically?
- Aurora performs well even with loss rates > 5%



Results: Robustness to Latency

- Training: 50-500ms
- Test: 1-512ms
- Why does TCP CUBIC link utilization reduces after 200ms latency?
- Why is Aurora's link utilization low at latency = 1ms?



Open Questions

- Fairness: How will Aurora perform when competing with TCP CUBIC?
- Application requirements differ: multiple objective reinforcement learning?
- Generalization guarantees?

RL approaches in Networking: Recent Work

- Generalizability challenges
  - Training in a wide range of environments
  - Out-of-distribution samples
  - **Solution: Use curriculum learning**

- System challenges
  - Deep Neural Networks are costly
  - Massive cross-space communication if deployed in the user space
  - **Solution: Deploy them inside the kernel space (?)**