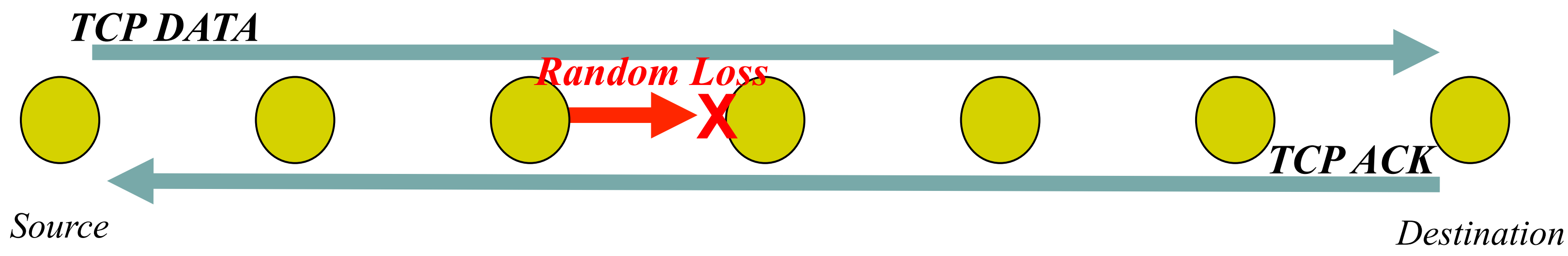
TCP over Network Coding

Presented by
Nadia Boukhatem
Télécom Paris

Joint Work with
Chien-Chia Chen, Mario Gerla, M.Y. Sanadidi
Network Research Lab
CSD, UCLA

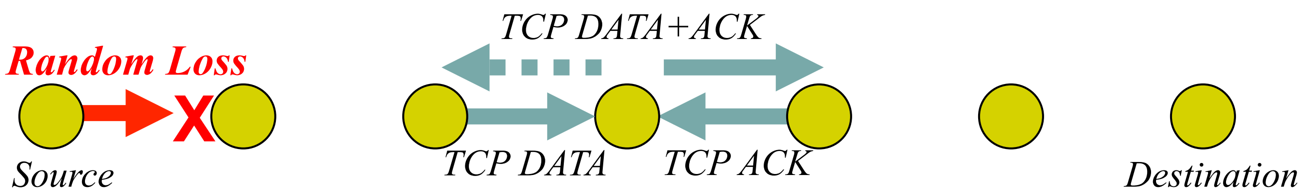
TCP in Wireless Networks

- Problems
  1. **random** losses are misinterpreted as **congestion**
  2. TCP DATA and ACK flows **contend** for the same shared medium



ComboCoding—
Combined Intra- and Inter-Flow Coding

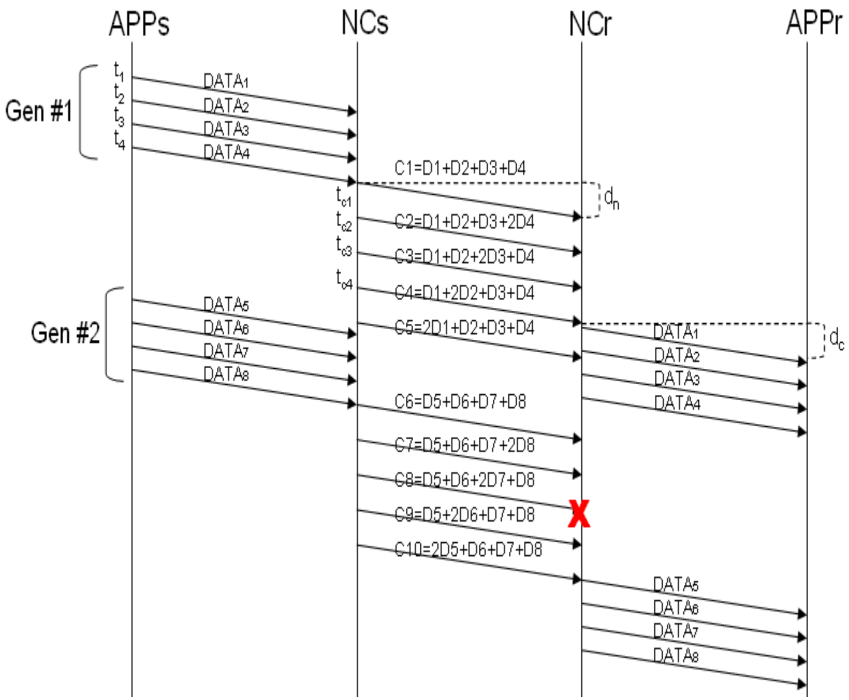
- To mitigate high loss
  →**Intra-Flow Coding** (Pipeline Coding)
  - Uses random linear coding to recover losses
- To mitigate DATA-ACK interference
  →**Inter-Flow Coding** (PiggyCode)
  - Opportunistically XOR DATA and ACK at relays
  - Mixing only DATA and ACK within the same TCP flow
- **Transparent to Upper/Lower Layers**



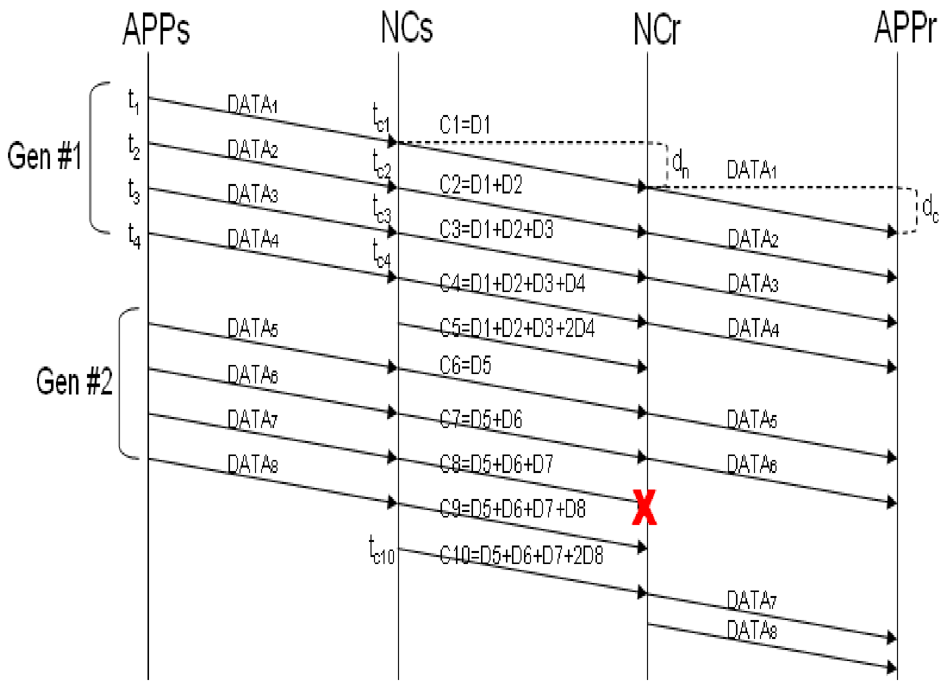
Intra-Flow Coding (Pipeline Coding)

- Benefits
  - Reduced delay
  - Improved throughput
  - Transparency to higher layers

Batch Coding

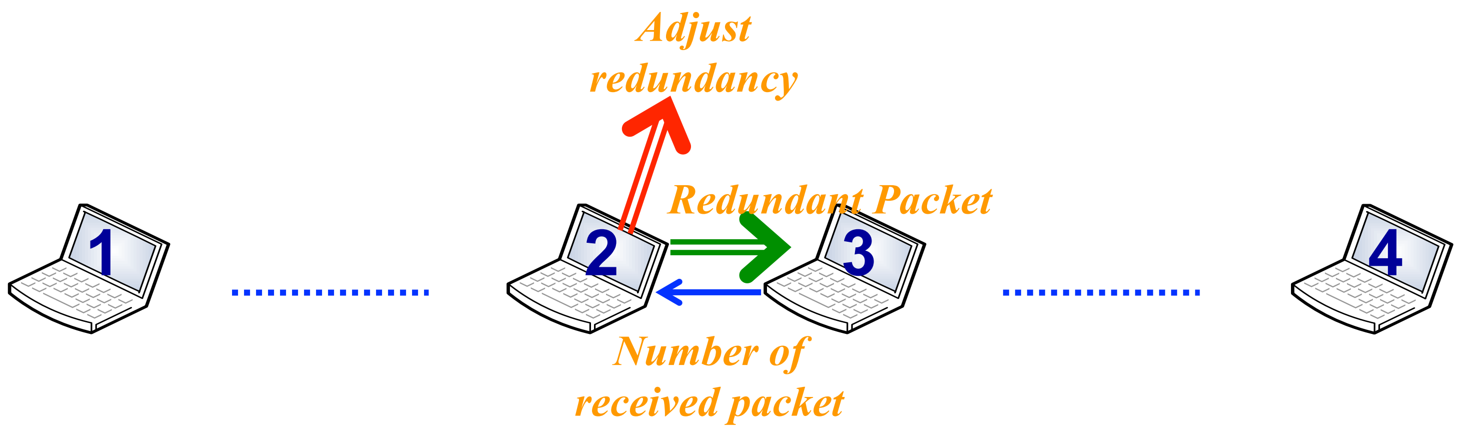


Pipeline Coding



Adapt to Varying Losses

- Link error rates are changing at all times



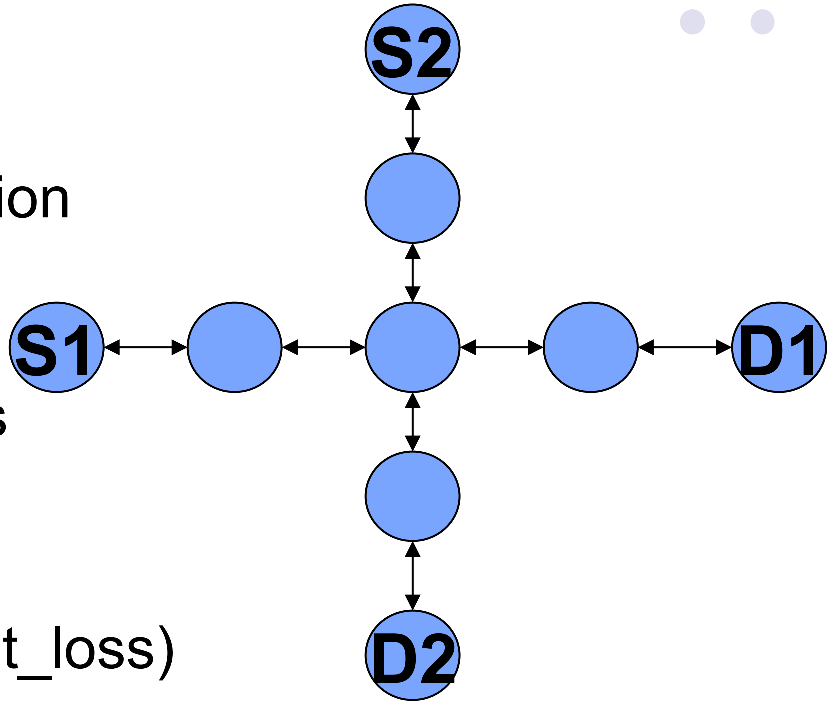
- Each node stamps "***number of received packet***" in packets header
  - Upstream node receives it
  - It adjusts ***link*** coding redundancy based on ***successful delivery*** (to the next hop)

Simulation Evaluation

- Simulator: Qualnet 4.5
- Coding schemes are implemented at Network Layer as a special type of routing protocols
- TCP-NewReno is chosen for the transport layer protocol
- Our previous work has shown the effectiveness of our coding scheme
- We focus on **fairness** and **friendliness** comparisons in this presentation

Simulation Configuration

- 802.11g Unicast at 54Mbps
  - CSMA/CA
  - RTS/CTS is DISABLED
  - MAC ACK and MAC retransmission (up to 7 times)
  - Promiscuous Mode ENABLED
- Traffic: 2 FTP/TCP-NewReno Flows
- Gen size: 16
- Base Redundancy K = 0.65
- Adaptive Redundancy = K + 1/(1-est_loss)



Simulation Setup

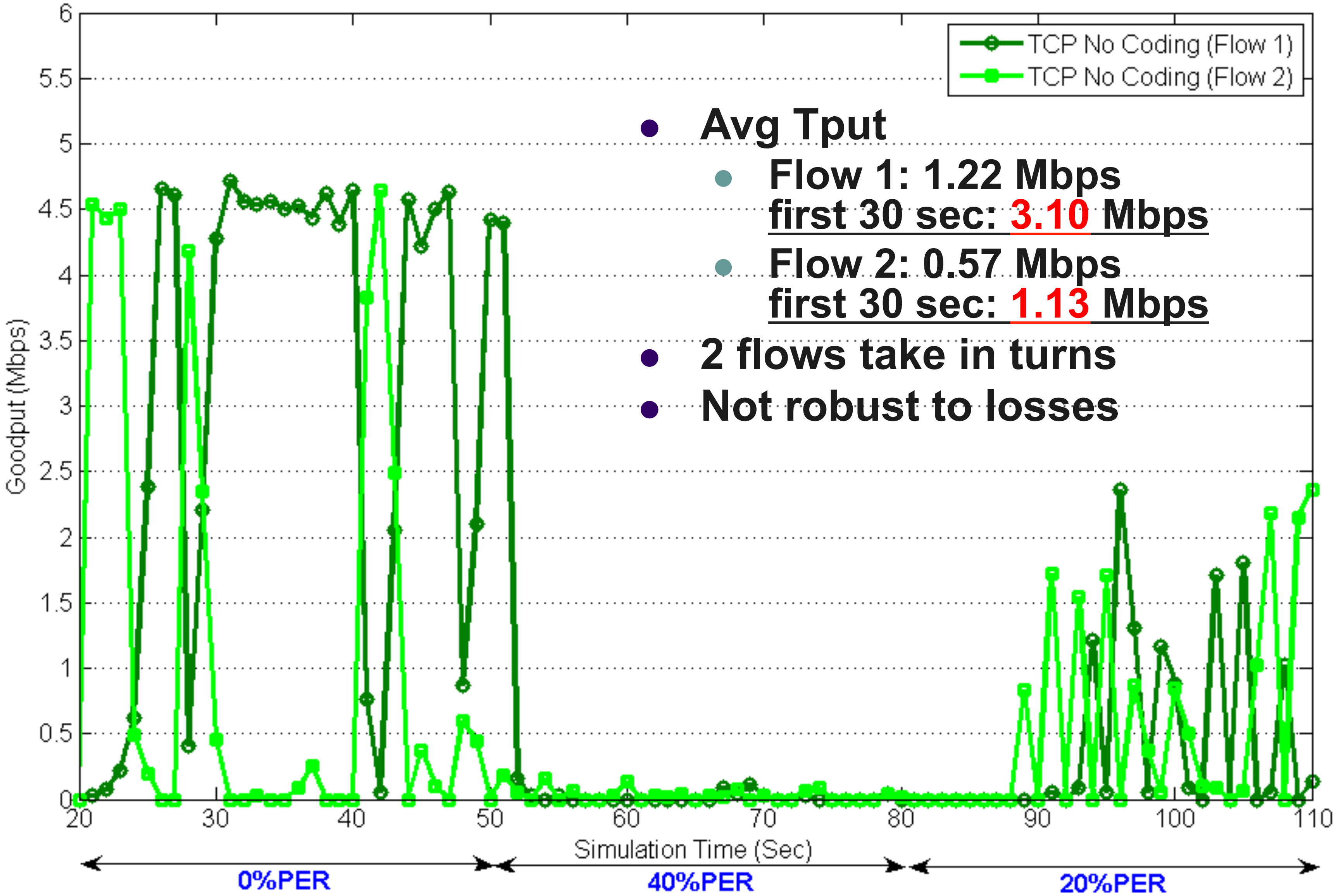
- 2 TCP Flows in 6 Runs

		NC	Redundancy
Run 1	TCP Flow 1	No	No
	TCP Flow 2	No	No
Run 2	TCP Flow 1	ComboCoding	Adaptive
	TCP Flow 2	ComboCoding	Adaptive
Run 3	TCP Flow 1	ComboCoding	Adaptive
	TCP Flow 2	No	No
Run 4	TCP Flow 1	ComboCoding	Adaptive
	TCP Flow 2	No	Adaptive
Run 5	TCP Flow 1	ComboCoding	No
	TCP Flow 2	No	No
Run 6	TCP Flow 1	No	Adaptive
	TCP Flow 2	No	Adaptive

- Simulation time: 110 seconds (FTP starts at 20 sec)
- Vary per link Packet Error Rate over time
  - **20~50** sec: **0%** PER
  - **50~80** sec: **40%** PER
  - **80~110** sec: **20%** PER

Run 1: No Coding No Redundancy

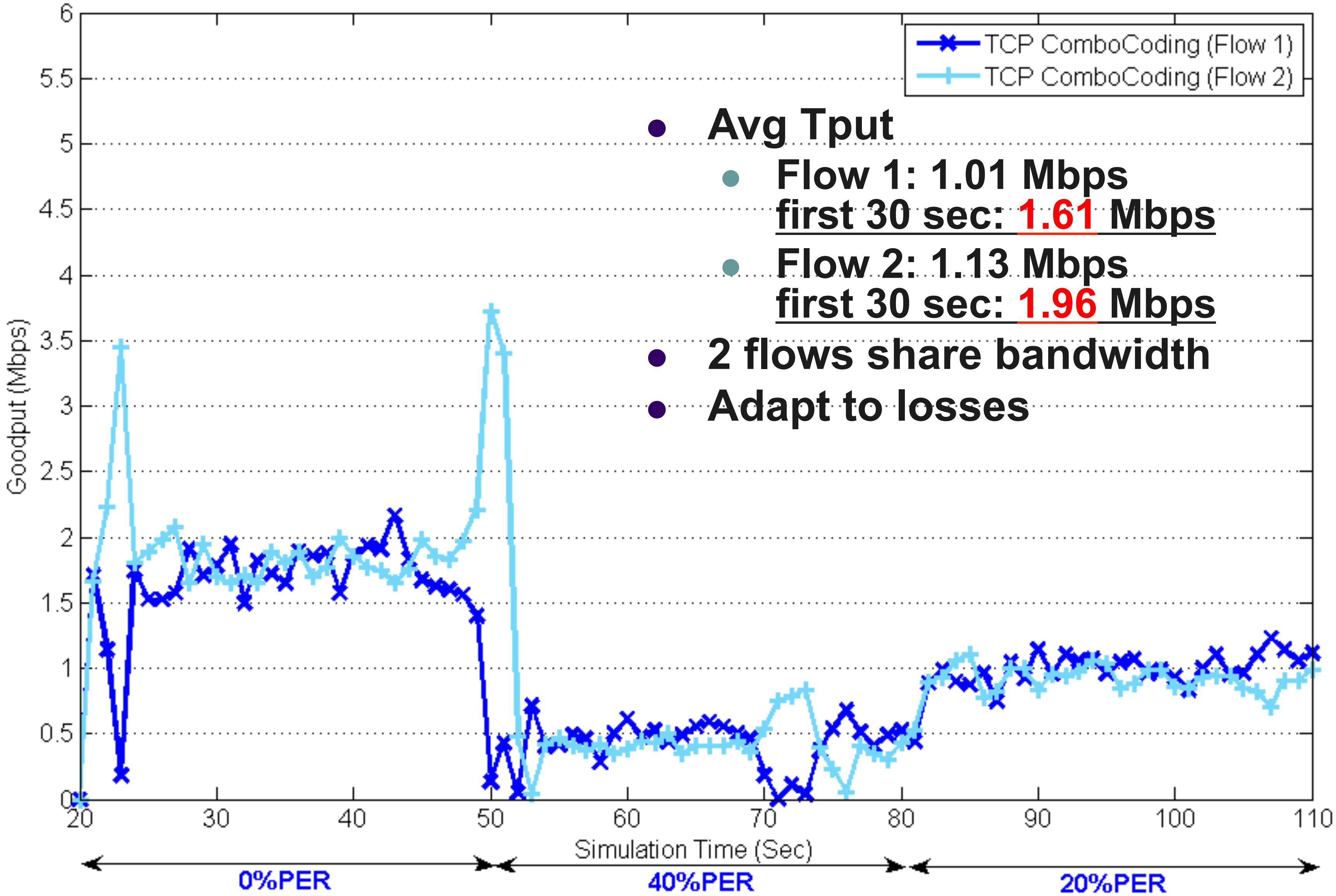
UCLA CSD



- Avg Tput
  - Flow 1: 1.22 Mbps
    first 30 sec: 3.10 Mbps
  - Flow 2: 0.57 Mbps
    first 30 sec: 1.13 Mbps
- 2 flows take in turns
- Not robust to losses

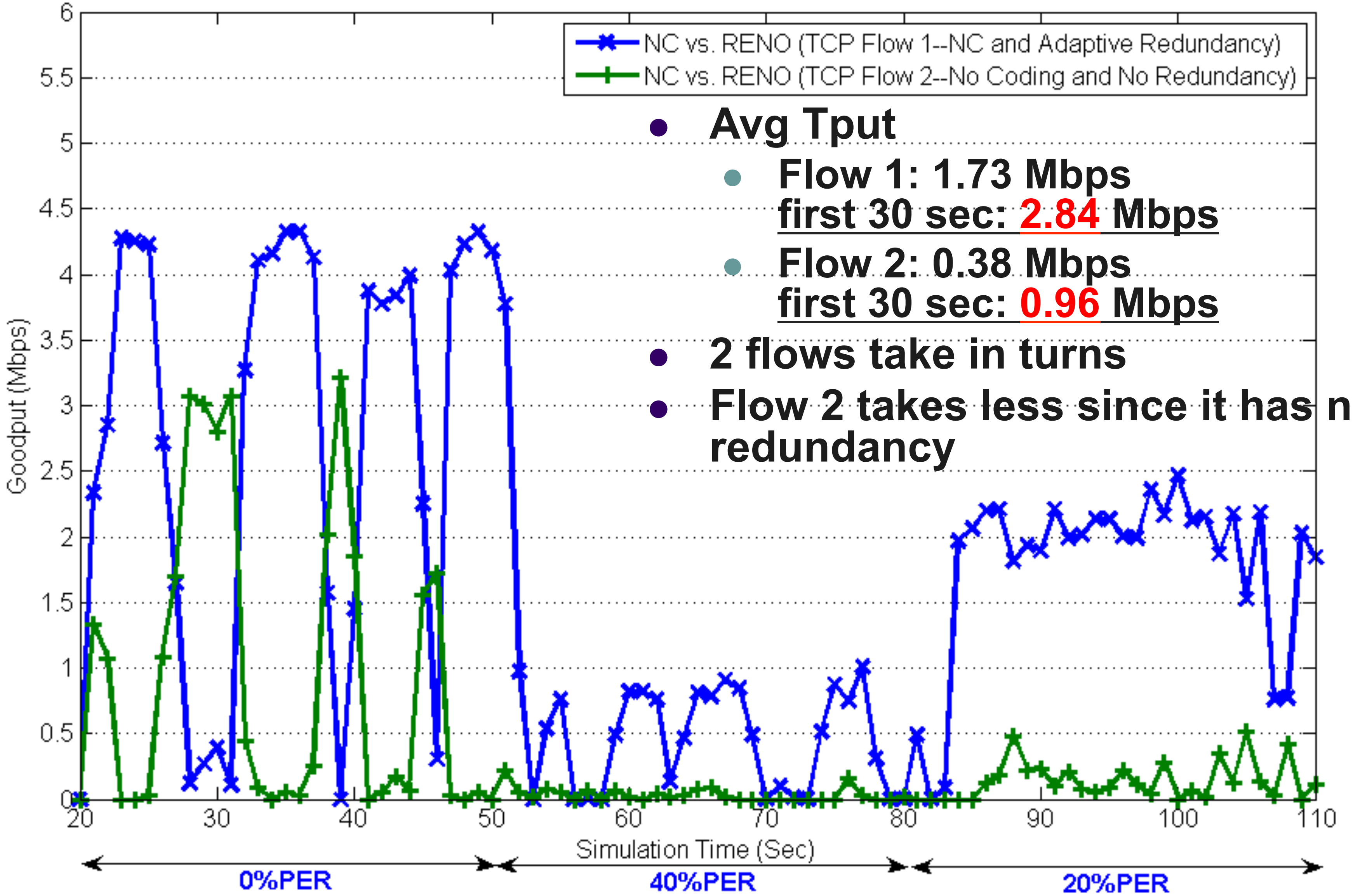
UCLA CSD

Run 2: ComboCoding with Adaptive Redundancy



- **Avg Tput**
  - **Flow 1: 1.01 Mbps**
    <u>first 30 sec: 1.61 Mbps</u>
  - **Flow 2: 1.13 Mbps**
    <u>first 30 sec: 1.96 Mbps</u>
- **2 flows share bandwidth**
- **Adapt to losses**

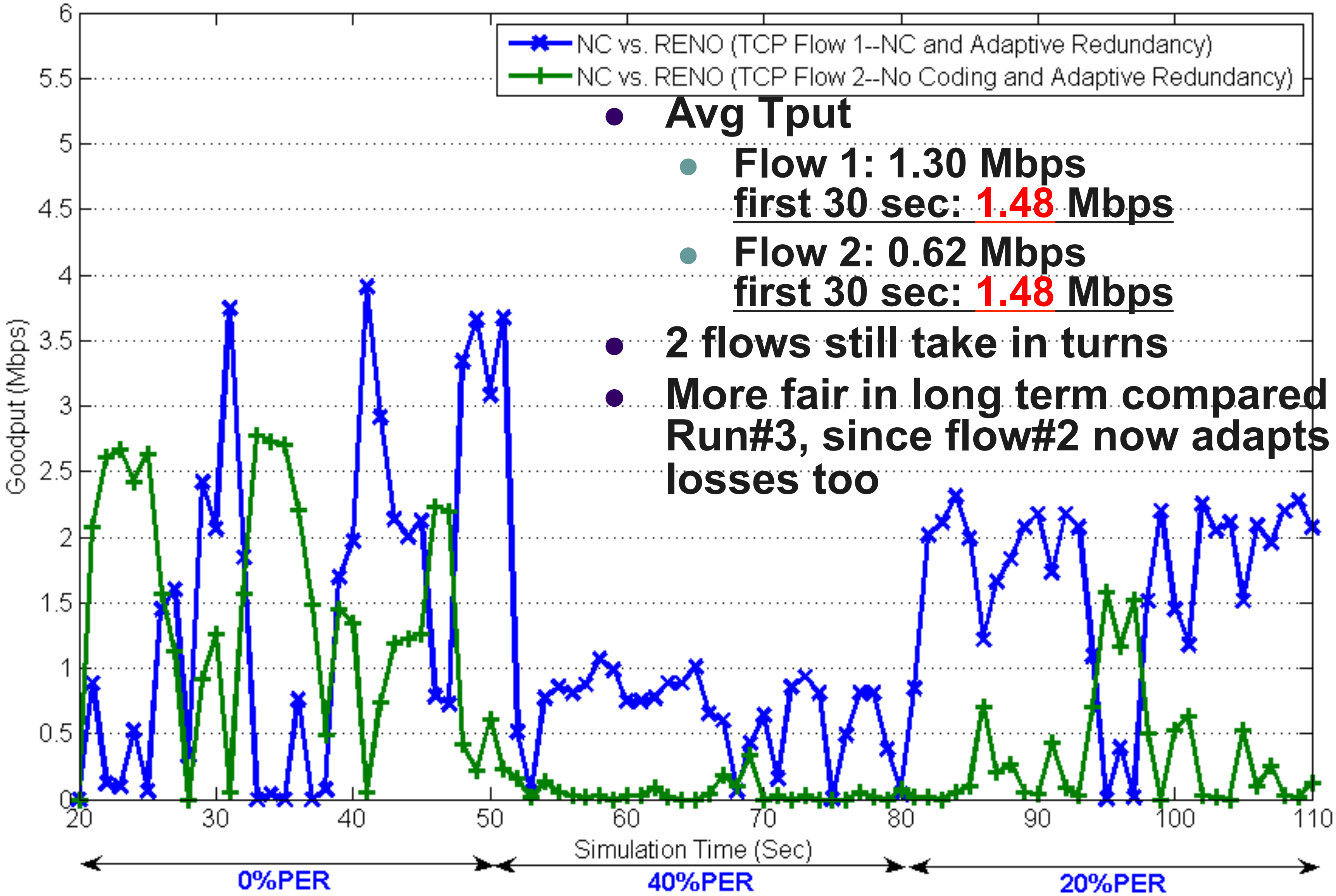
Run 3: No Coding vs. ComboCoding



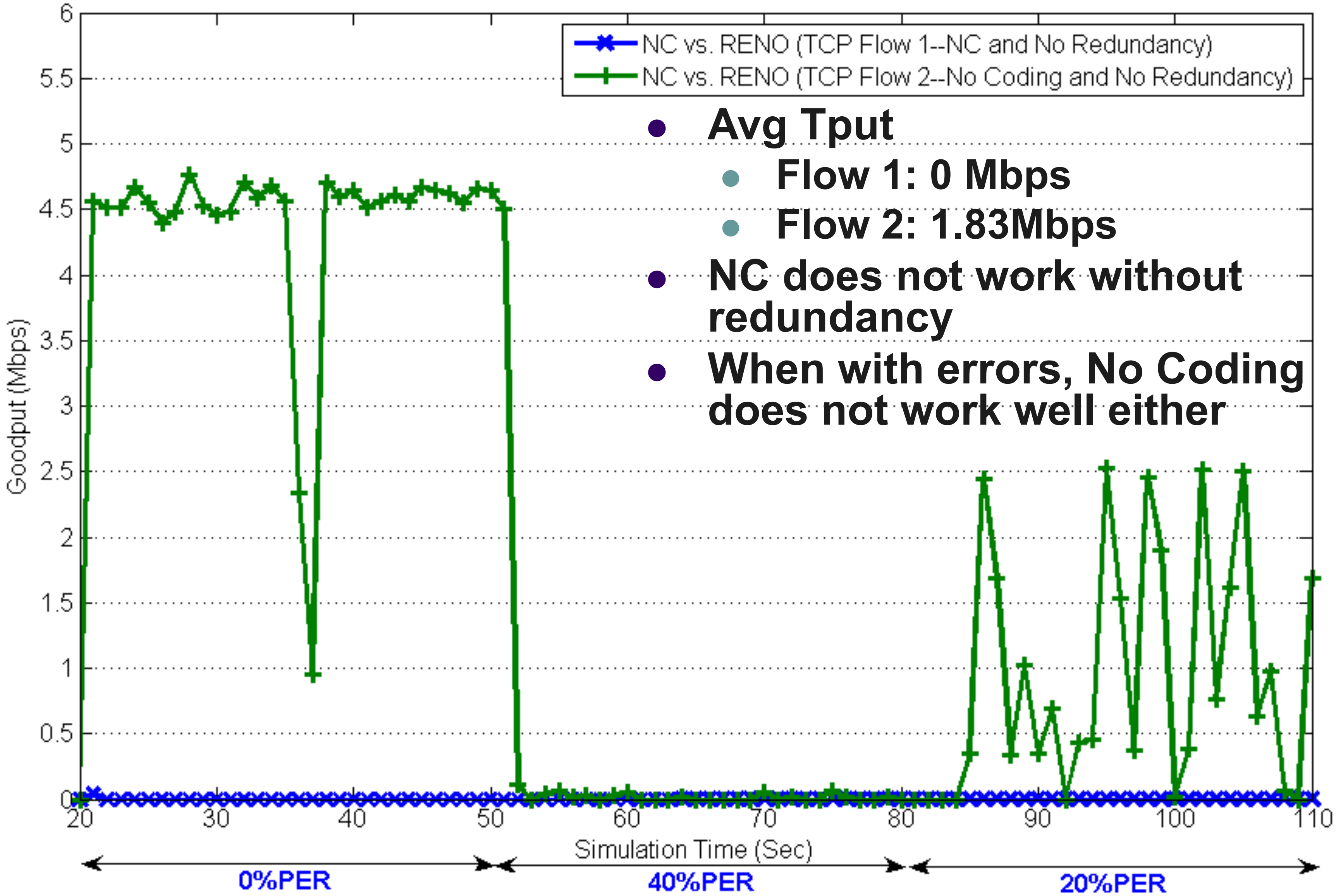
- Avg Tput
  - Flow 1: 1.73 Mbps
    first 30 sec: 2.84 Mbps
  - Flow 2: 0.38 Mbps
    first 30 sec: 0.96 Mbps
- 2 flows take in turns
- Flow 2 takes less since it has no redundancy

Run 4: Adaptive Reno vs. ComboCoding

- **Avg Tput**
  - **Flow 1: 1.30 Mbps**
    first 30 sec: 1.48 Mbps
  - **Flow 2: 0.62 Mbps**
    first 30 sec: 1.48 Mbps
- **2 flows still take in turns**
- **More fair in long term compared to Run#3, since flow#2 now adapts to losses too**



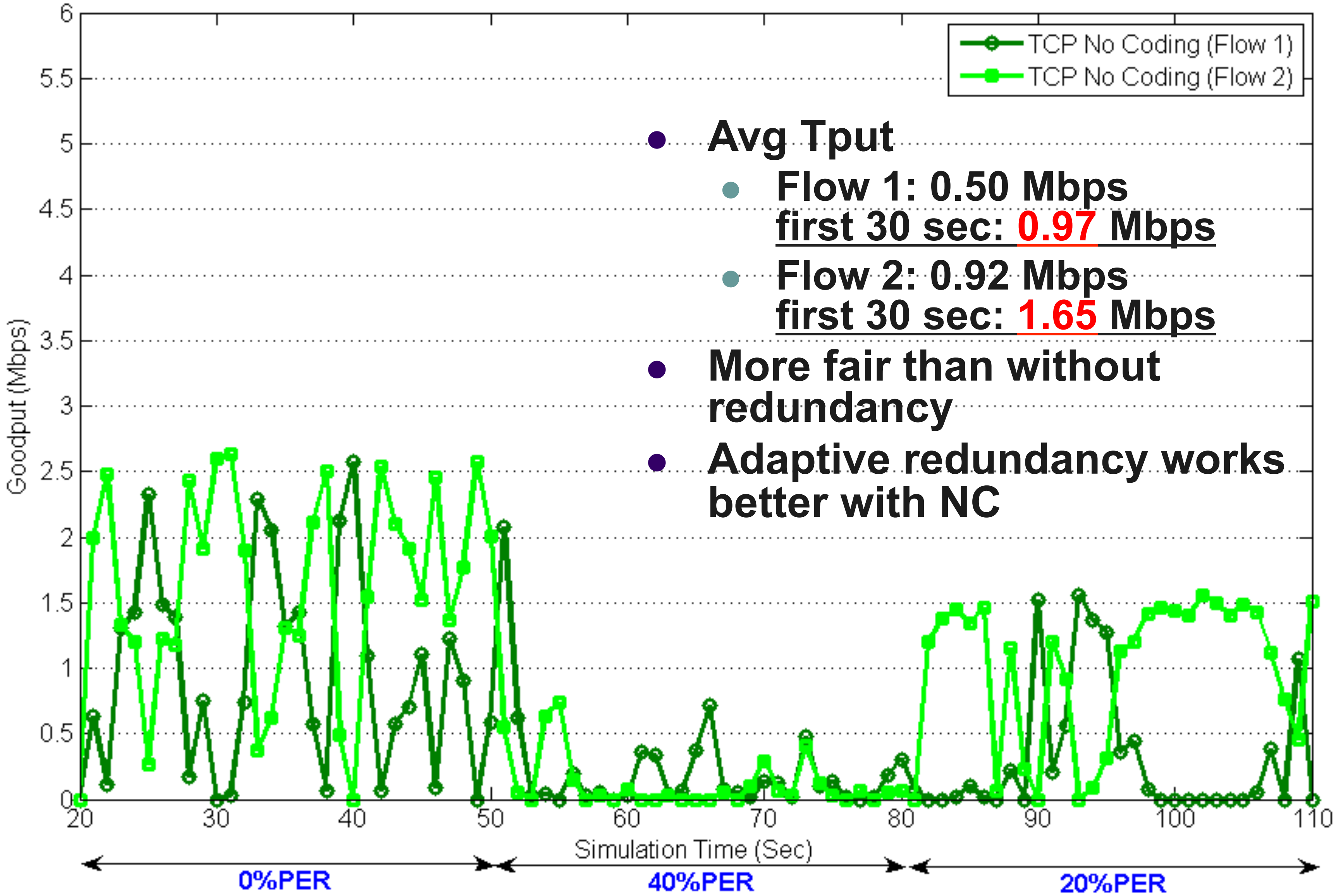
Run 5: No Coding vs. NC No Redundancy



- Avg Tput
  - Flow 1: 0 Mbps
  - Flow 2: 1.83Mbps
- NC does not work without redundancy
- When with errors, No Coding does not work well either

Run 6: No Coding with Adaptive Redundancy

- Avg Tput
  - Flow 1: 0.50 Mbps
    first 30 sec: 0.97 Mbps
  - Flow 2: 0.92 Mbps
    first 30 sec: 1.65 Mbps
- More fair than without redundancy
- Adaptive redundancy works better with NC



Conclusion

- TCP/NetCode works much better than TCP for random loss channels
- TCP/NetCode is intra-fair and more stable than TCP
- TCP/NC can coexist with TCP (although there is some unfriendliness due to the adaptive redundancy)
- Current work
  - Adaptive redundancy control should employ loss discrimination techniques to have a more accurate random error estimate

Related Work

- "On the impact of random losses on TCP performance in coded wireless mesh networks," INFOCOM 2010 by Prof. Ros et al.
  - Study applies only to the single hop, opportunistic coding between different flows (ie inter-flow coding), like COPE
  - Interflow -coding improves throughput, but is known to be vulnerable to random errors
  - Two links correct reception requirement and packet-loss synch across TCP flows are unique of Interflow-coding
- We use standard TCP NR and only Intraflow Coding:
  - Interflow coding applied only to DATA and ACKs

THANK YOU ☺