Decongestion Control

Barath Raghavan and Alex C. Snoeren
University of California, San Diego

UCSDCSE
Computer Science and Engineering

Network resource sharing

1. Statistical multiplexing + buffers

2. Admission control + reservations

3. Decongestion control + coding

Sharing resources

	Implicit	Explicit
End-point	HighSpeed Vegas FAST TCP BIC Westwood Scalable	TFRC PCP
In-network	RED ECN AQM	WFQ RCP Admission Control XCP

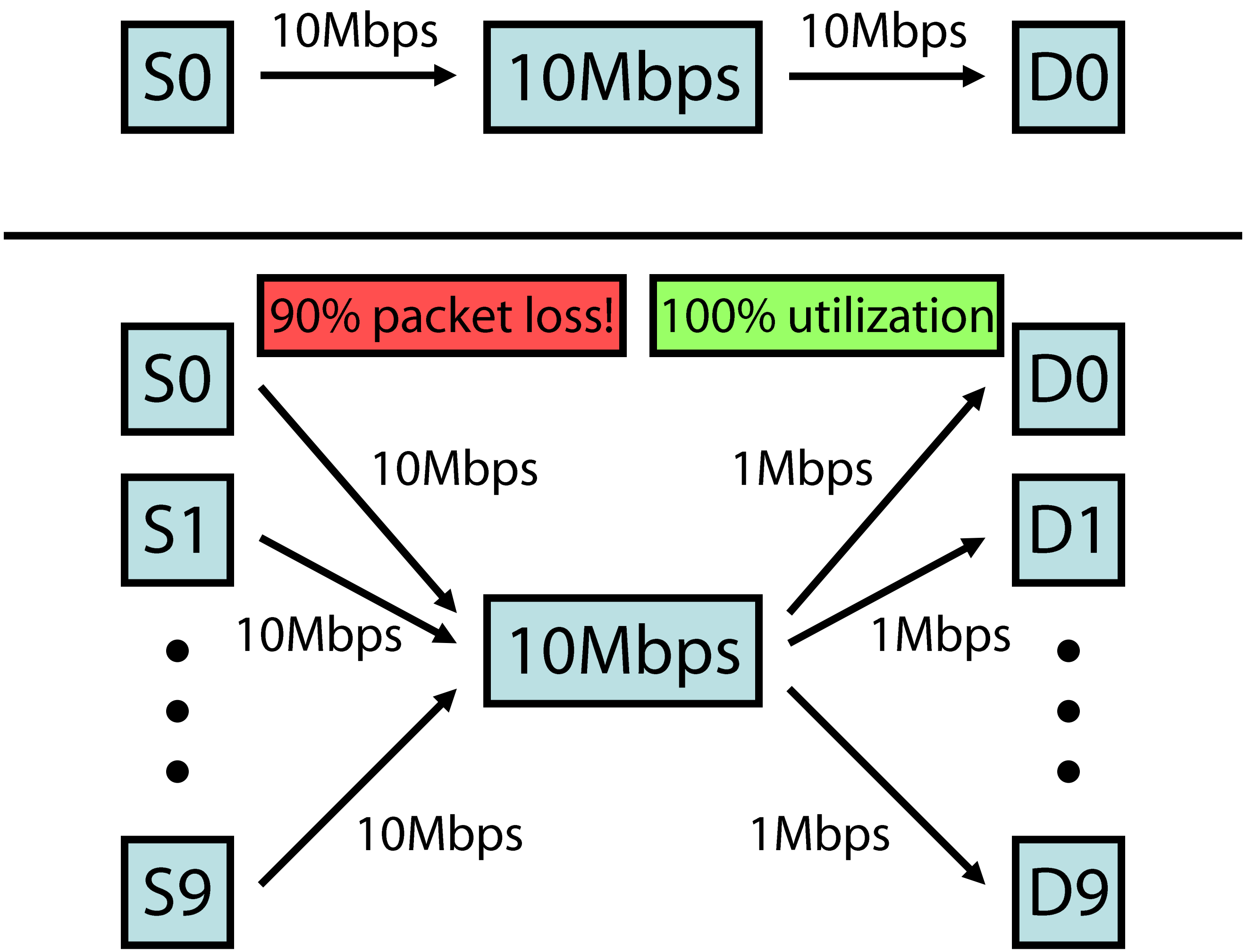
Sharing resources

	Implicit	Explicit
End-point	HighSpeed Vegas FAST TCP **Backoff** BIC Westwood Scalable	**Request and set** TFRC PCP
In-network	RED **Network hints** ECN AQM	WFQ RCP **Rate allocation** Admission Control XCP

Ignoring packet loss

1. A simple thought experiment
2. Decongestion control
3. Design considerations

Simple greedy transport

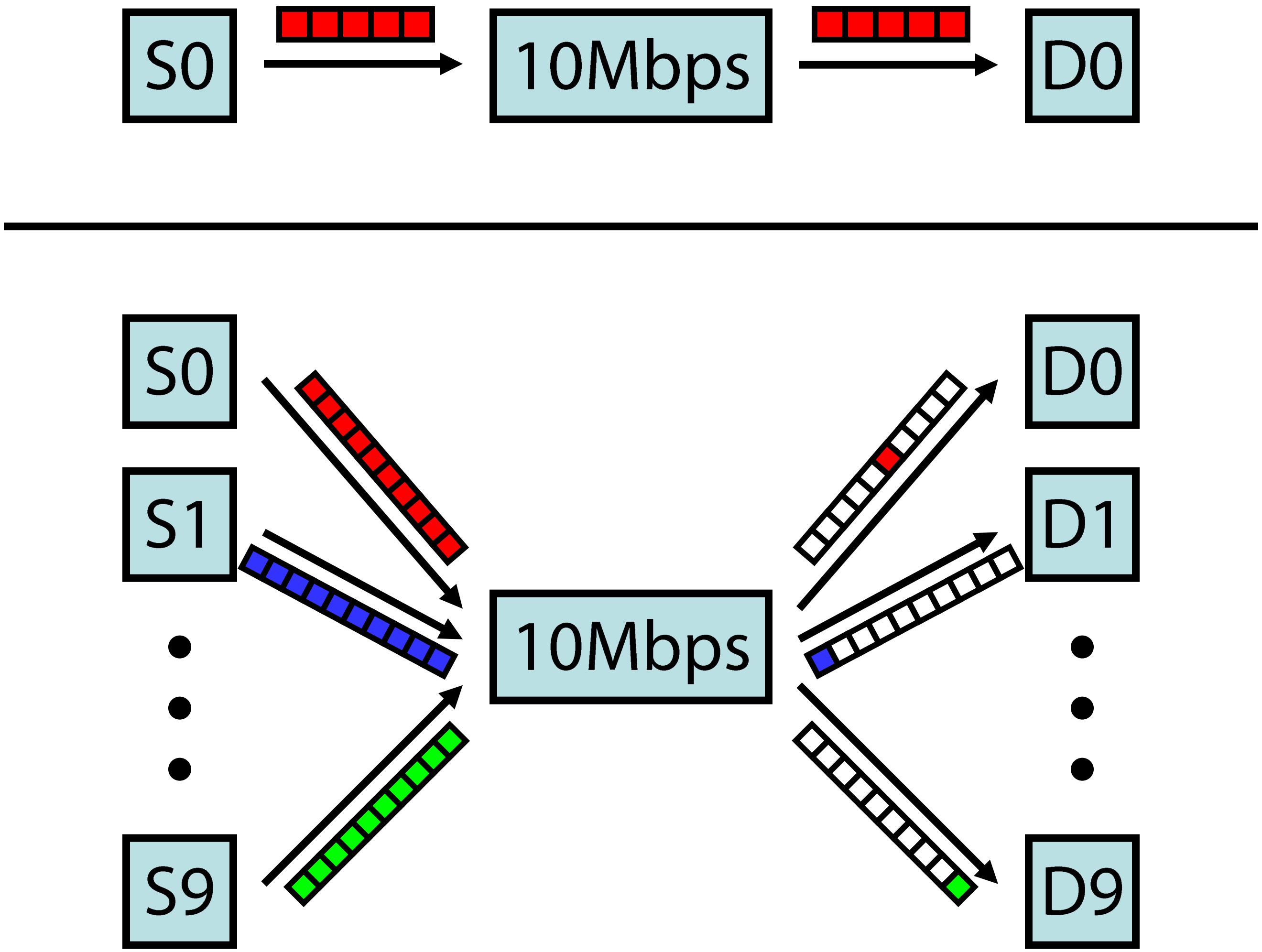


Decongestion control

Persistent network congestion is ok, if:

- End-to-end goodput is high
- End-to-end delay is low
- We maintain inter-user fairness

Decongestion control

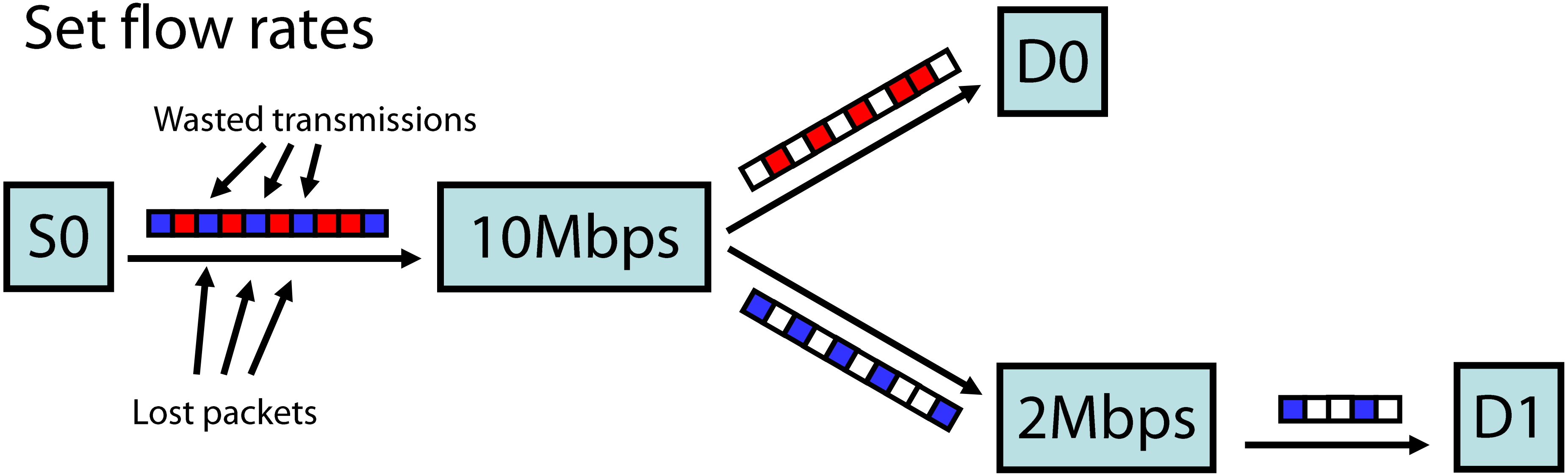


Decongestion control

- Send packets as fast as possible
  - Aligned with end-host incentives
- Erasure code the data
  - Most/all packets received will be useful
  - Dynamically change coding rate
- Drop packets fairly at routers

Challenge: transmission

Set flow rates

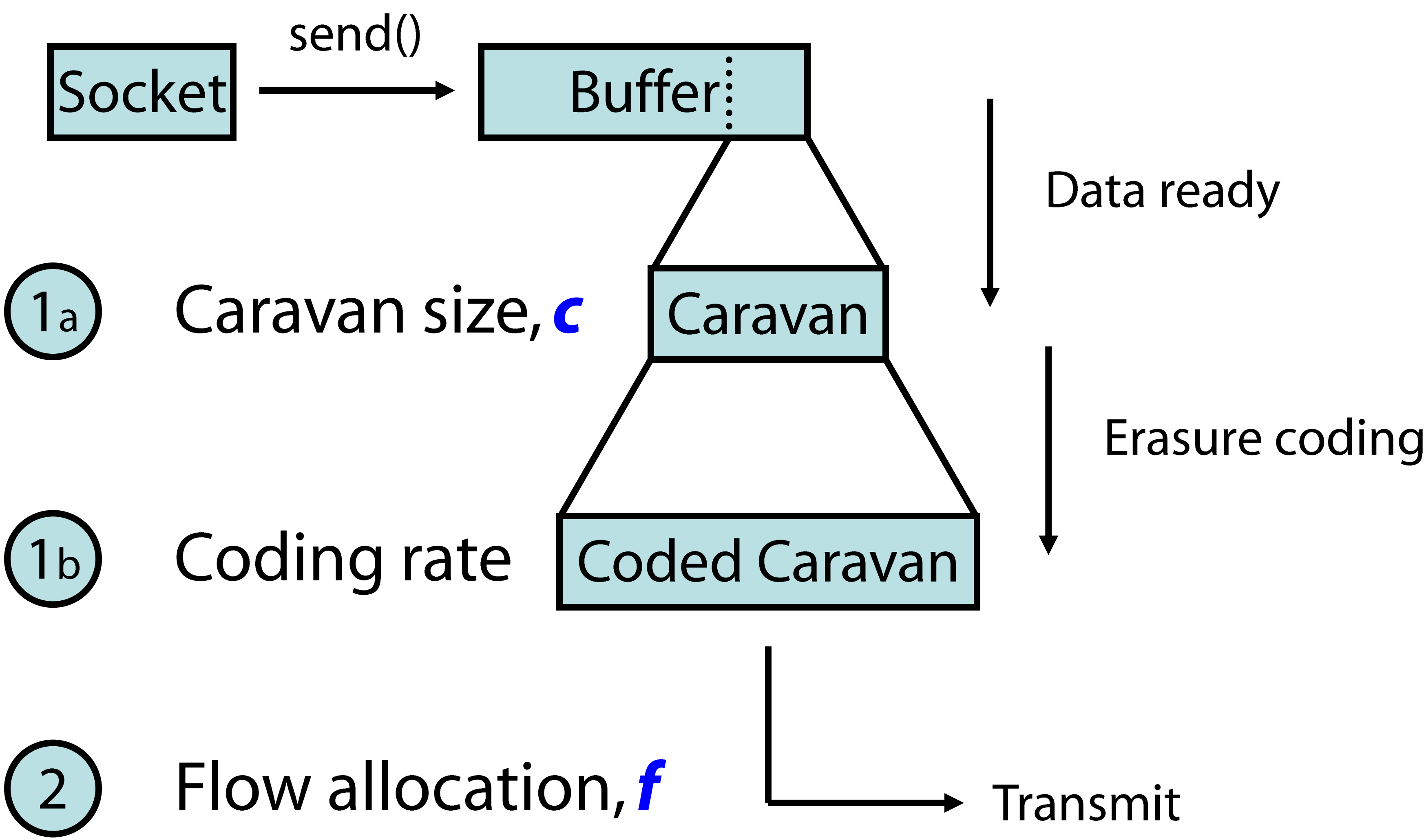


Set coding parameters

Design challenges

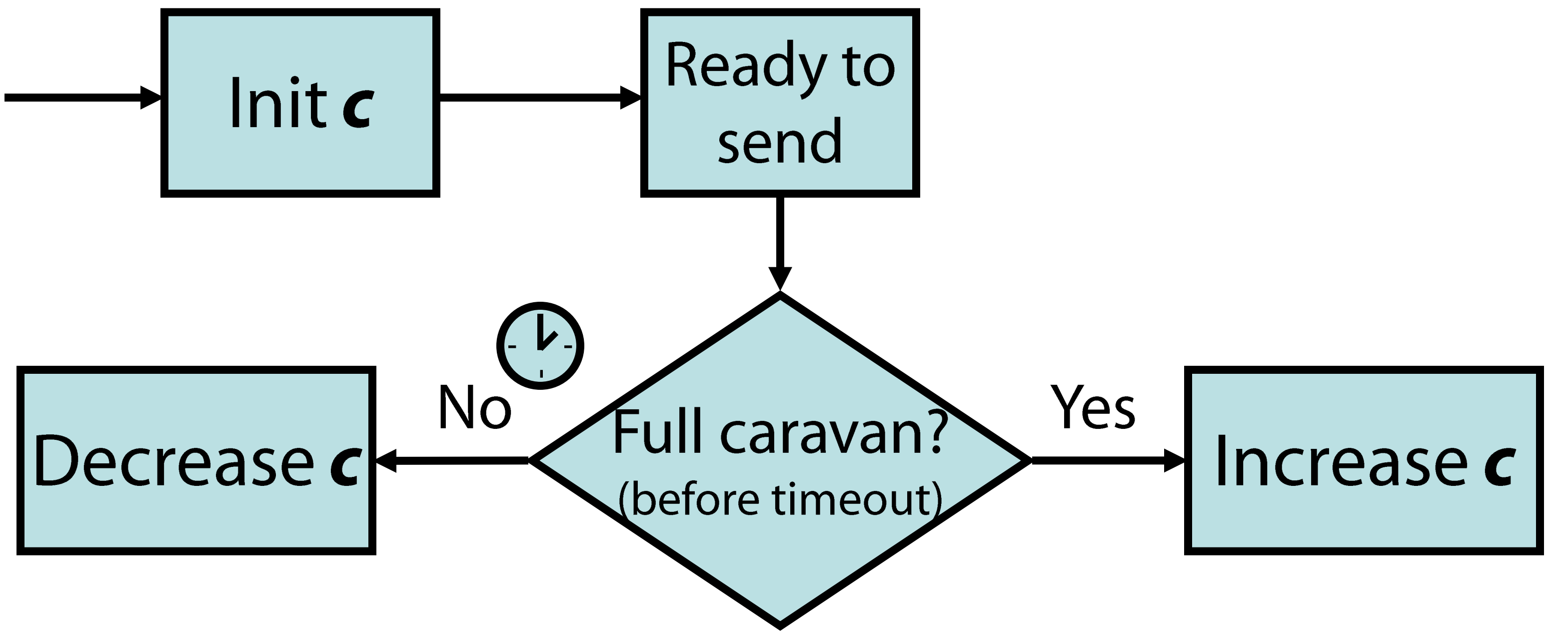
1. Setting transmission parameters

Sending data



Setting caravan size, *c*

- Tradeoff: coding overhead vs. latency
  - Bulk vs. interactive flows

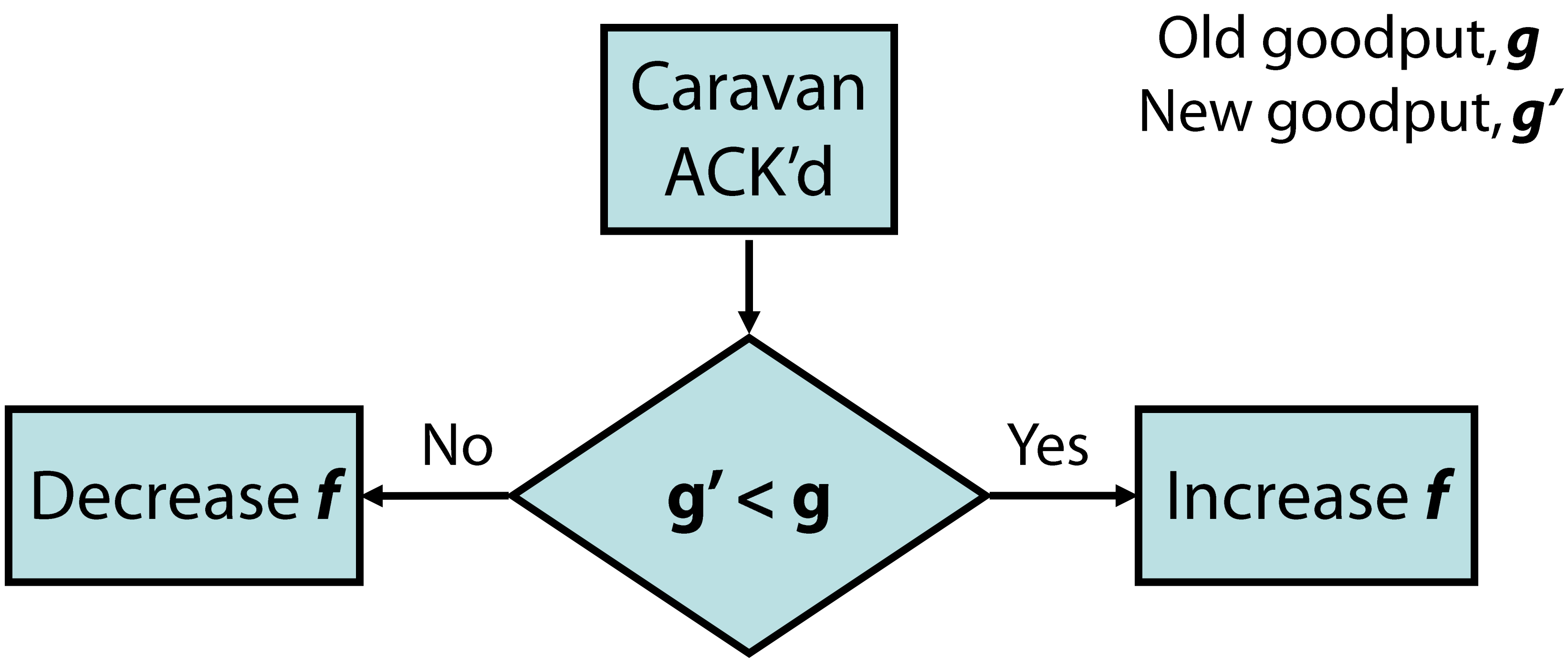


Setting coding rate

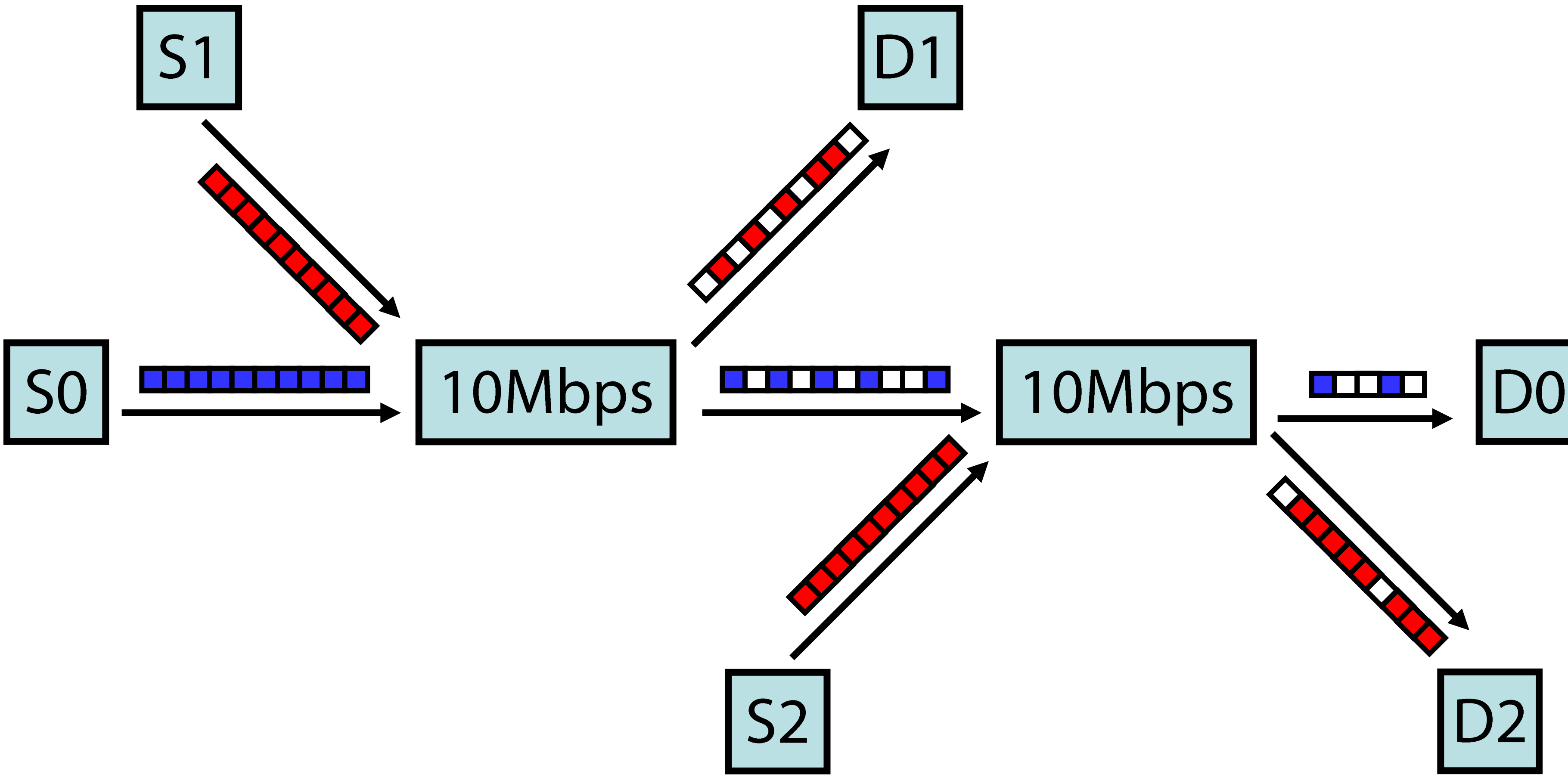
- Estimate goodput with receiver feedback
  - Set coding rate accordingly
- Tradeoff: type of erasure code
  - Standard: Reed-Solomon
  - Rateless: LT codes, online codes
  - Simple: Redundancy
- Coding rate doesn't impact other flows
  - Provides stability of traffic demands

Setting flow allocation, *f*

- End host has limited bandwidth
  - Must apportion bandwidth to its flows
  - Still wants to be greedy



Challenge: long vs. short paths



Design challenges

1. Setting transmission parameters
2. Enforcing fairness at bottlenecks

Flow fairness

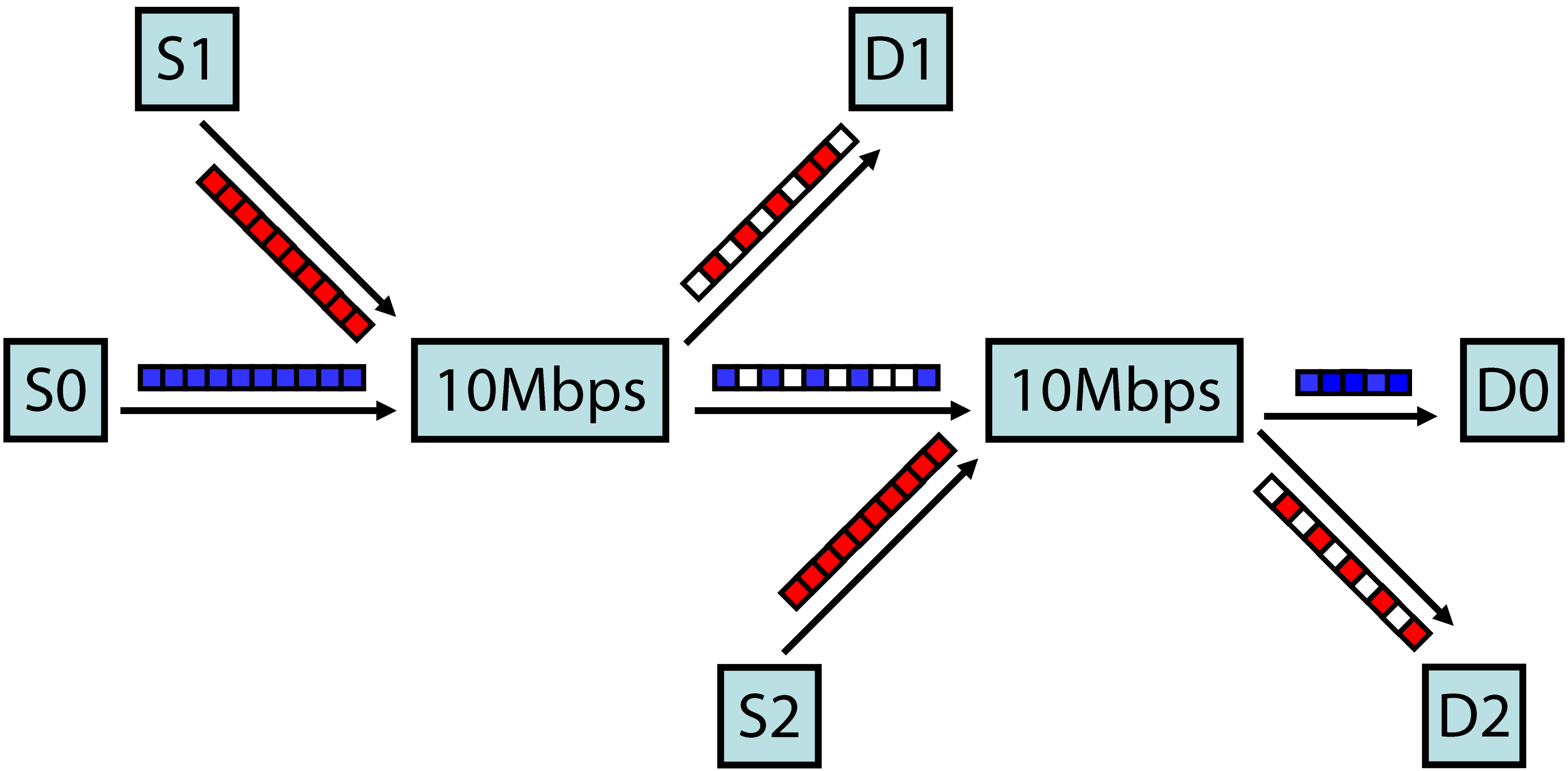
Idea:

- Throttle flows to their fair-share at routers

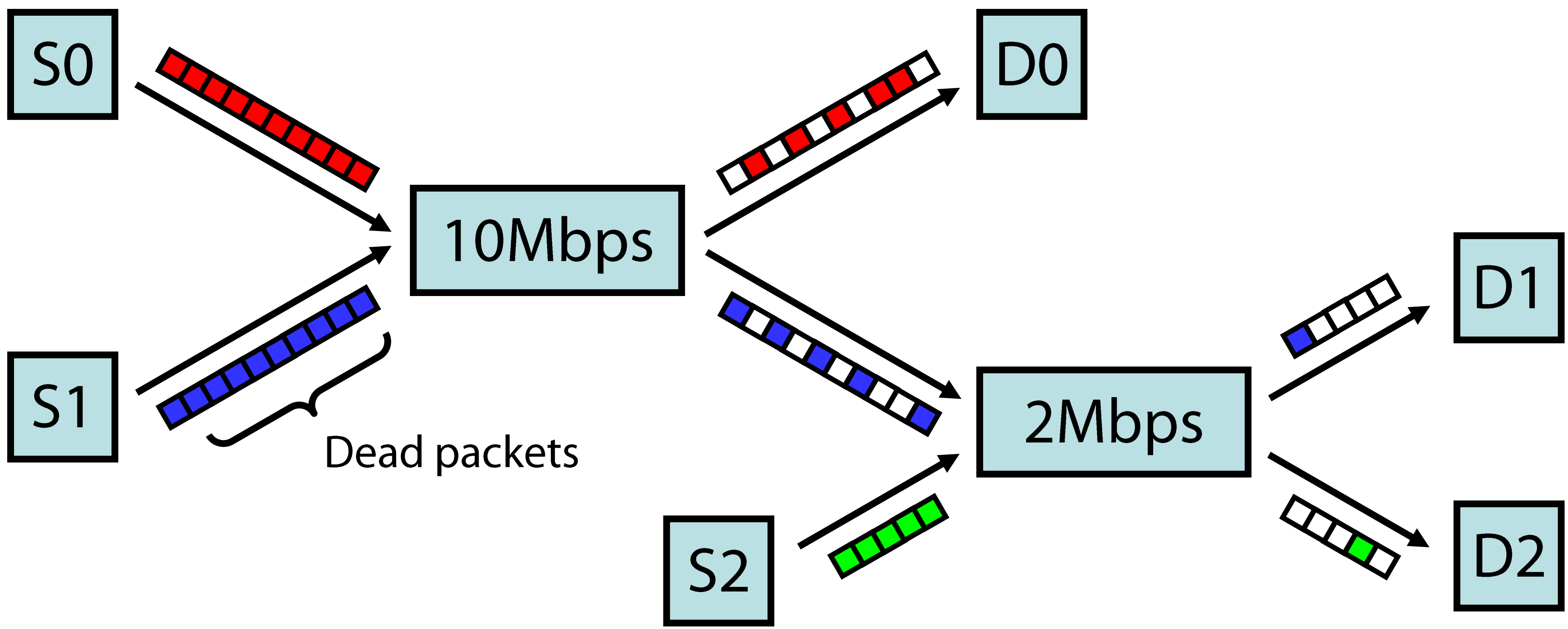
Implementation:

- Use fair *dropping* rather than fair *queueing*
  - For example, Approximate Fair Dropping (AFD)

Long paths with fair dropping



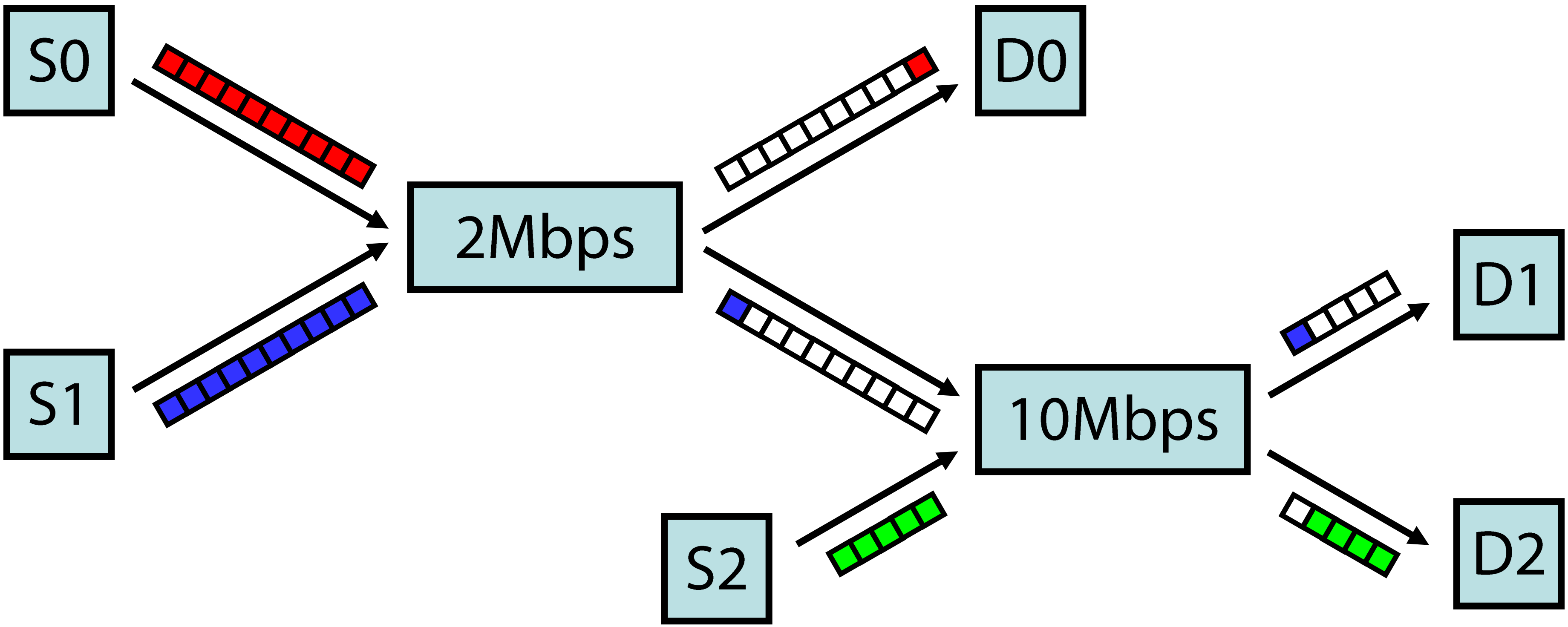
Challenge: link wastage



Design challenges

1. Setting transmission parameters
2. Enforcing fairness at bottlenecks
3. Avoiding "dead packets"

Conjecture: few dead packets



Design challenges

1. Setting transmission parameters
2. Enforcing fairness at bottlenecks
3. Avoiding "dead packets"

Potential benefits

- High total end-to-end goodput
  - Network is always delivering *coded* data
- Small router buffer requirements
  - Traffic is not bursty or sensitive to loss
- Incentive compatibility
  - Aligned with greedy sender behavior
- Traffic stability
  - Only flow arrival/departure affects path demand

Questions!

UCSDCSE
Computer Science and Engineering