Low Latency Transport Mechanism

Behnam Montazeri

May 15, 2015

Assumptions

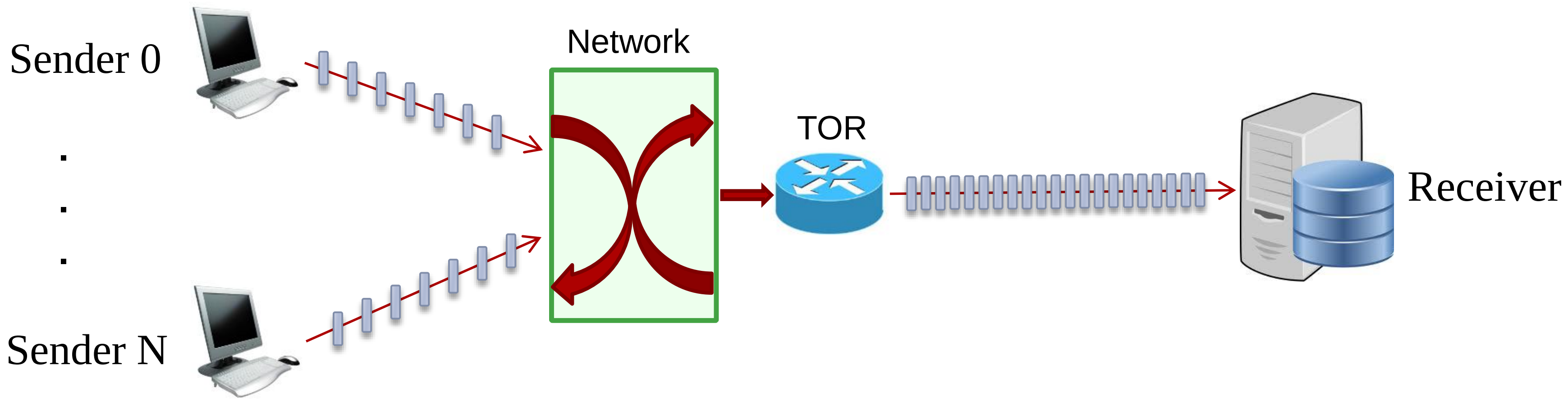
- **DataCenter Network**
- **Full Bisection Bandwidth Topology**
- **Low latency network**
  - ✓ 10Gb/s links speed or higher
- **(Near) Optimal load balancing**
  - ✓ Shortest Queue
  - ✓ Random Spray
  - ✓ Round Robin
- **Switches provide few priority levels**

Objectives

- **Low Latency**
  - ✓ As close as possible to hardware limit
  - ✓ Fewest Remaining Bytes First (FRBF)
  - ✓ Minimal Buffer Usage
- **Scalability**
  - ✓ One million client connection per server
  - ✓ Minimal per client state
- **Active Congestion Control**
- **Handle packet reordering**
- **Handle delay variations**

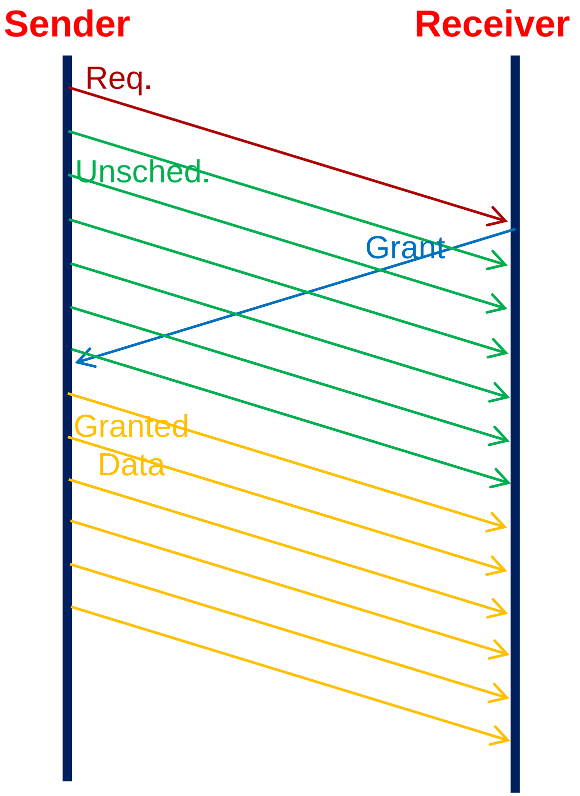
Observations

- Receiver has info about incoming messages
- Congestion happens at the edge at the receiver



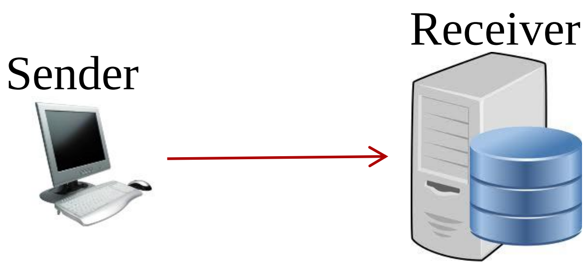
Overall Idea

- **Receiver Side Scheduling**
  - ✓ Sender sends request
  - ✓ Receiver grants permission for transmission
- **Allow preemption to favor short messages**
  - ✓ Scheduling policy: SRBF
  - ✓ Utilizing small number of network priorities
- **Avoid scheduling overhead**
  - ✓ Small unscheduled traffic covers for 1 RTT



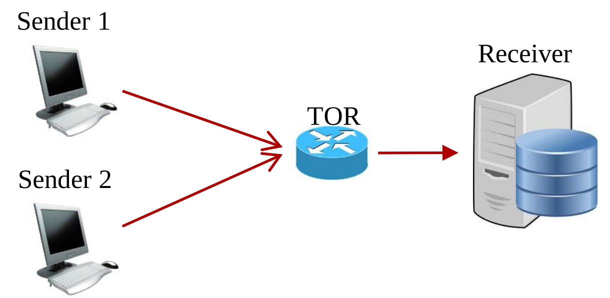
Scenario 1

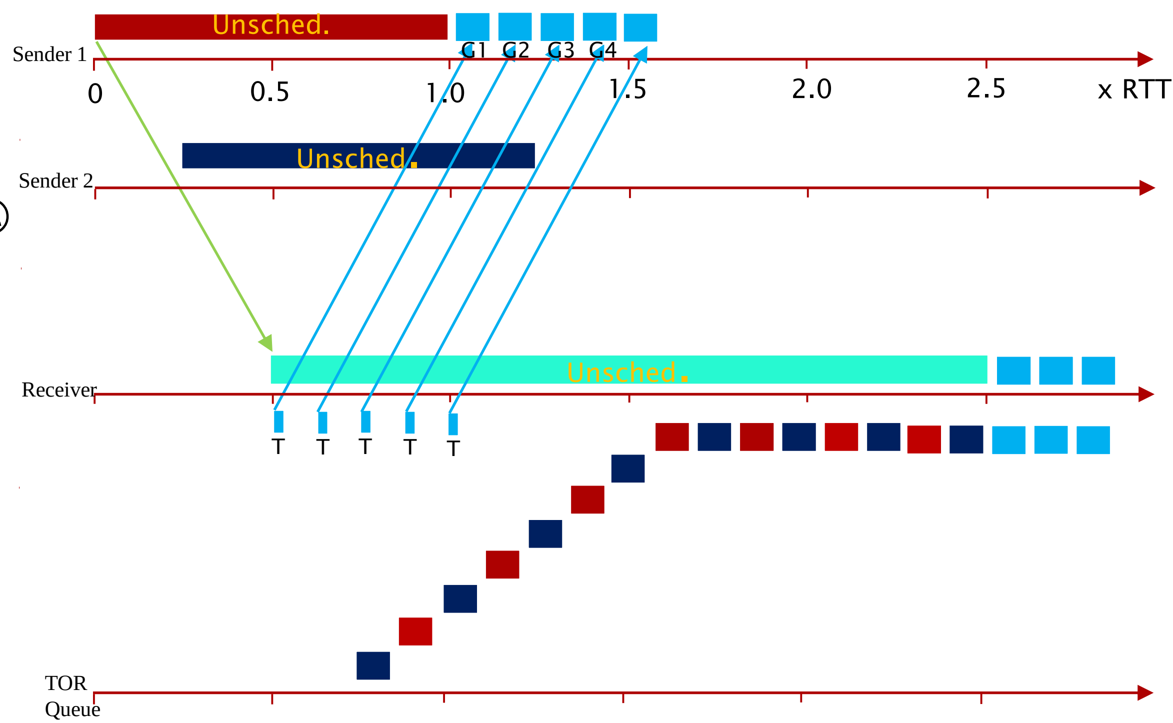
- **One sender, One receiver**
- **Network delay is fixed**
- **Ideas:**
  - ✓ Sender sends requests based on SRBF policy
  - ✓ Sends BDP worth of unscheduled data
  - ✓ Receiver sends one grant every packet time
    - Preempts large requests in favor of shorter requests



Scenario 2

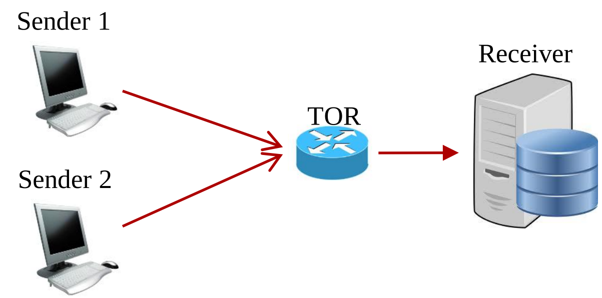
- **Two Senders, One receiver**
  - ✓ Tokens are sent every packet time
  - ✓ Observations
    - Packets may buffer at the TOR
    - Unsched. traffic
      - Small fraction of load
      - Covers RTT
      - High priority
    - Delay variations exist ☹

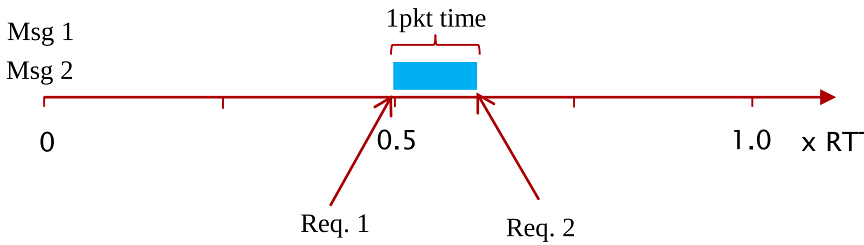




Scenario 2

- Msg2 shorter than Msg1
  - ✓ Msg1 should be preempted





FIFO

$$T_1 = \frac{rtt}{2} + \cancel{V_{max}} + m_1 + \overbrace{\frac{rtt}{2} + V}^{u_2}$$

$$T_2 = \frac{rtt}{2} + m_1 + m_2$$

Preempt by tokens

$$T_2 = \frac{rtt}{2} + m_2 + \overset{b}{\cancel{1}} + u_1$$

$$T_1 = \frac{rtt}{2} + m_1 + m_2$$

Preempt by priorities

$$T_2 = \frac{rtt}{2} + m_2 + u_1$$

$$T_1 = \frac{rtt}{2} + m_1 + m_2$$

$$Penalty_{Prio\ vs\ token} = \frac{b}{\frac{rtt}{2} + m_1 + u_1} = \frac{b}{\frac{rtt}{2} + u + 1 + u}$$

rtt = 8, V=5, b=1
u = 0.5rtt + V

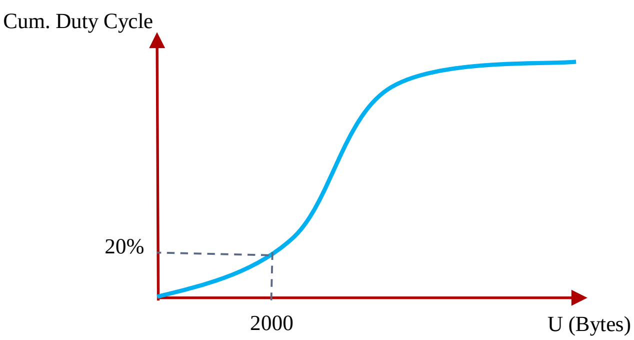
→ 4.3%

Scenario 2

Use token grants for preempting scheduled data

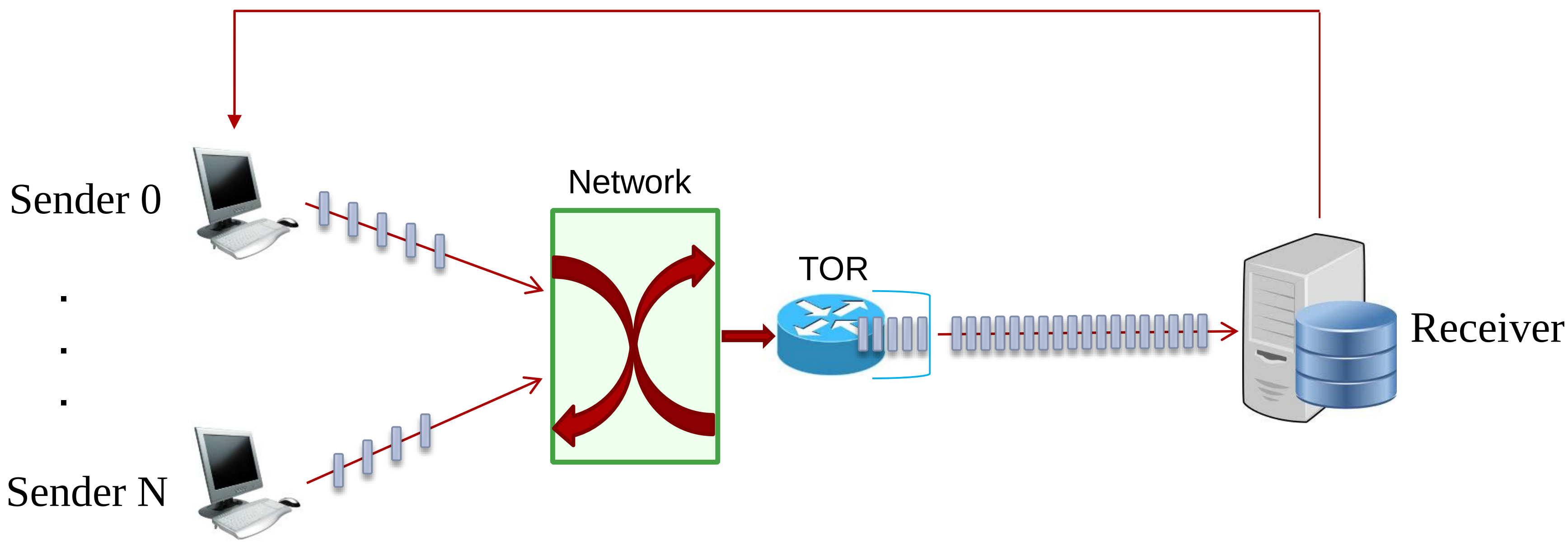
Many Senders

- **Sender:**
  - ✓ Sends U packets at PRIO_UNSCHED
  - ✓ Waits for token grants to arrive
- **Receiver:**
  - ✓ @ pkt time:
    - A) find shortest remaining req.
    - B) grant token to that req. Grant contains:
      - Permitted Bytes
      - Current value of U
      - Priority for that grant
  - ✓ @ new request
    - Update duty cycle distribution
    - Measure the duty cycle (UNSCHED_LOAD)
  - ✓ @UNSCHED_REFRESH_TIME
    - Update U based on duty cycle dist. and UNSCHED_MAX_LOAD



Many Senders: Short Comings

- **Delay variations are fundamental in networks**
  - ✓ Small amount of buffering helps to cover for delay variations
  - ✓ Large buffers hurt latency unless priorities are utilized

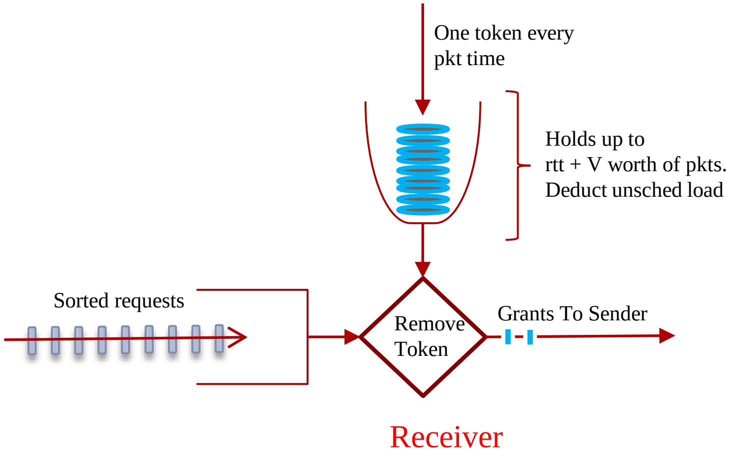


Many Senders: Short Comings

- **Delay Variation**
  - ✓ Idea: short buffer at TOR
  - ✓ Idea: Over commit outstanding tokens
- **Impact of unsched. packets on sched. Packets**
  - ✓ Idea: Token bucket scheduler
- **Impact of variable size pkts**
  - ✓ Idea: Use priorities
- **Impact of msg sizes on U**

Many Senders: Modified

- **Sender:**
  - ✓ Sends U packets at PRIO_UNSCHED
  - ✓ Waits for token grants to arrive
- **Receiver:**
  - ✓ @ pkt time:
    - A) Add a new token to bucket
    - B) cap bucket size to rtt + V
    - C) If token available, grant by SRBF policy. Grant contain
      - Permitted Bytes, Current value of U, Priority for that grant
  - ✓ @ new request
    - Update duty cycle distribution
    - Measure the duty cycle (UNSCHED_LOAD)
    - Subtract min(rem. req. size -1, U - 1) from bucket size
  - ✓ @UNSCHED_REFRESH_TIME
    - Update U based on duty cycle dist. and UNSCHED_MAX_LOAD



Unscheduled Load

- **Unsched. Load prevents bubbles at senders**
  - How should we decide on UNSCHED_LOAD
  - U should be large enough to prevent bubbles
  - U should be a small fraction of total load
  - Impact of msg. size on unsched. load
- **Idea:**
  - For msg size X:
    - Find the fraction of link capacity consumed by msg. size < X
    - The available BW to msg. size X = $\rho(msg.size < X)$
    - U for msg. size X then should be $(rtt + V) \times (1 - \rho(msg.size < X))$