

6.033 Spring 2019

Lecture #11

- **Reliable Transport**
- **Window-based Congestion Control**

Internet of Problems

How do we **route** (and address) scalably, while dealing with issues of policy and economy?



BGP

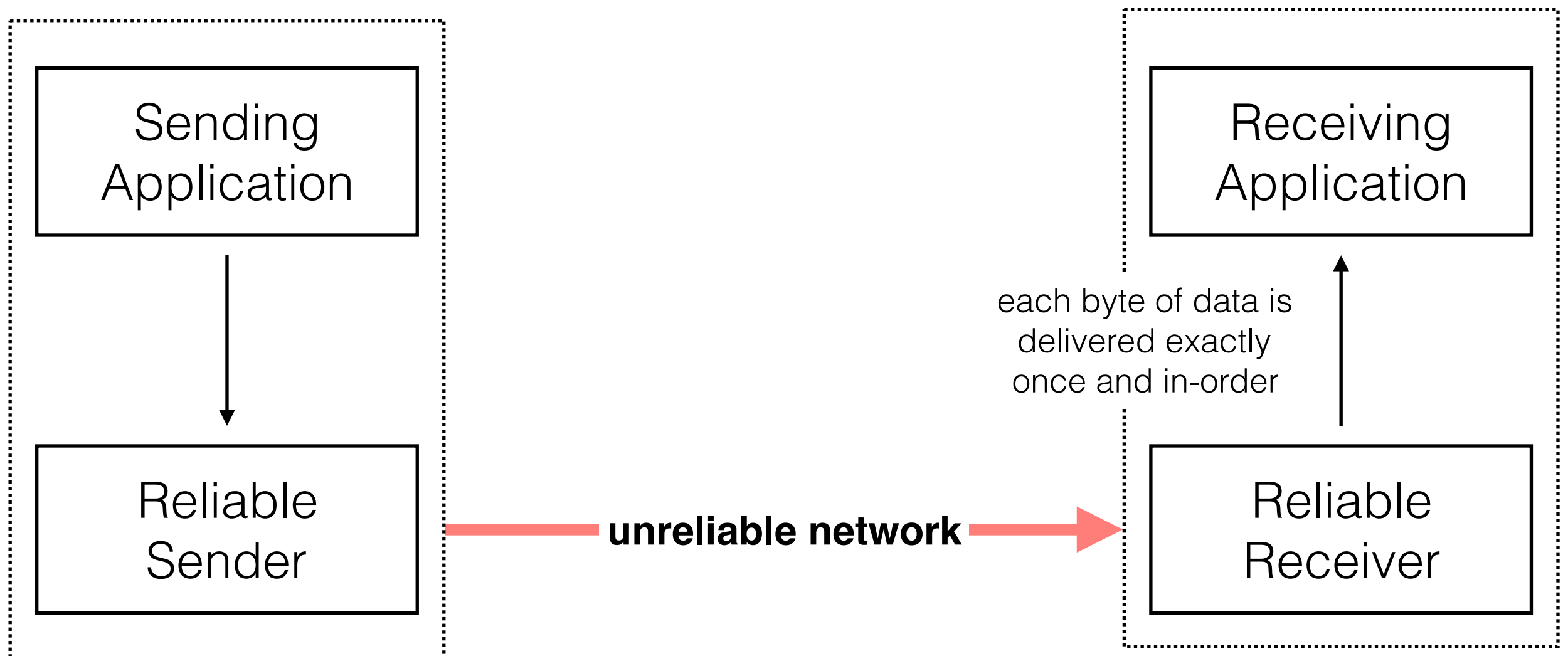
How do we **transport** data scalably, while dealing with varying application demands?



TCP

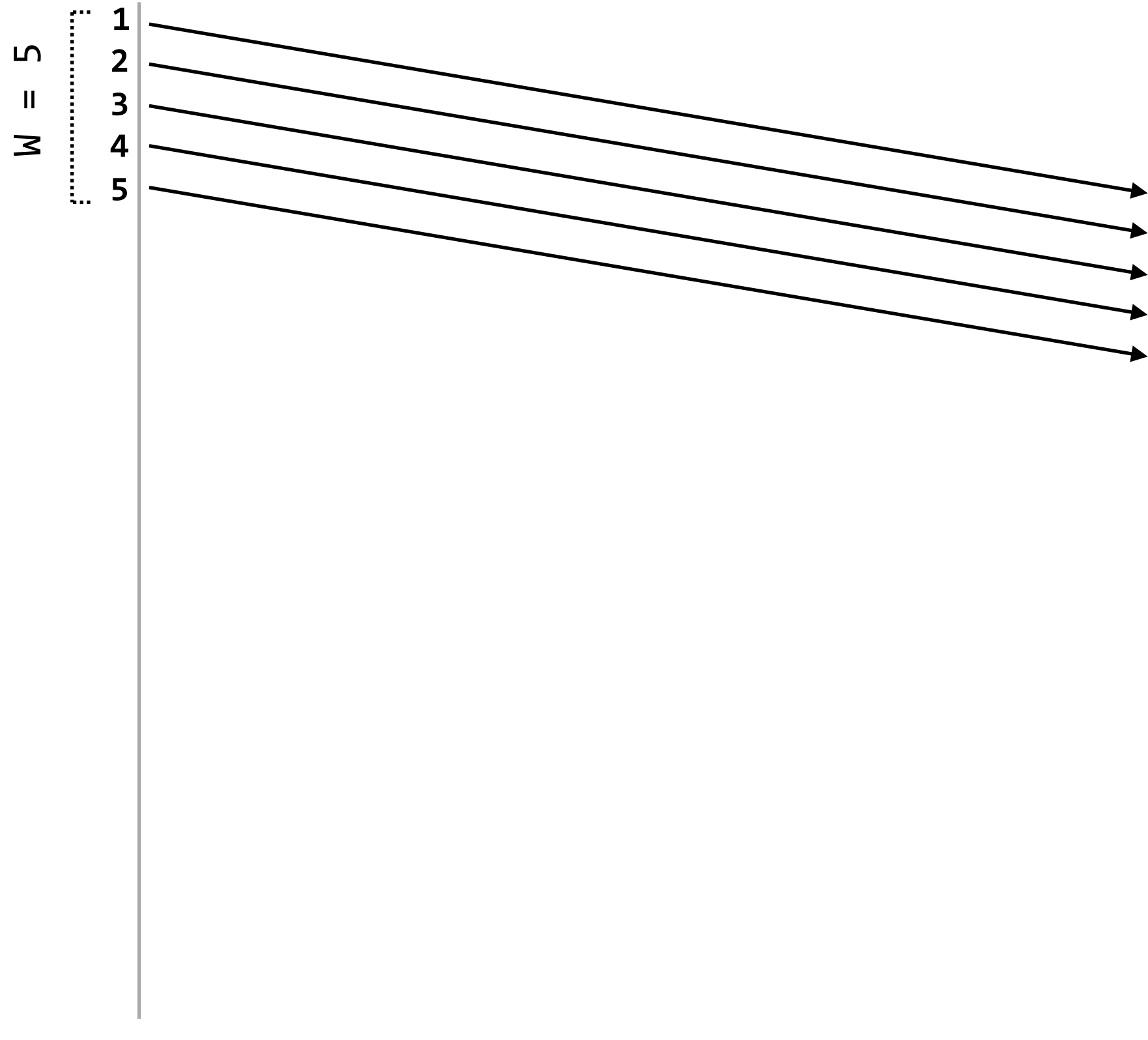
How do we **adapt** new applications and technologies to an inflexible architecture?

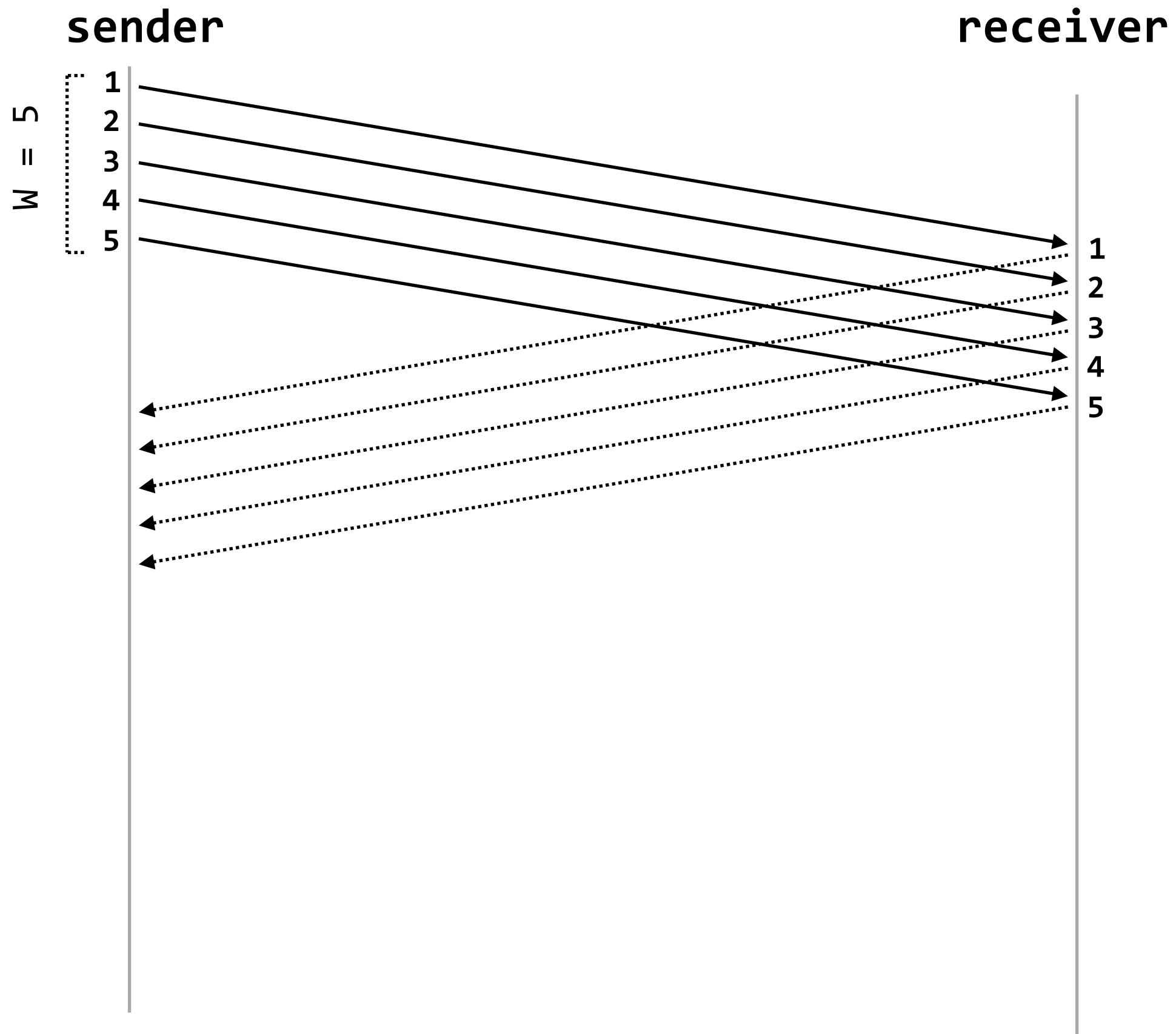
Reliable Transport



sender

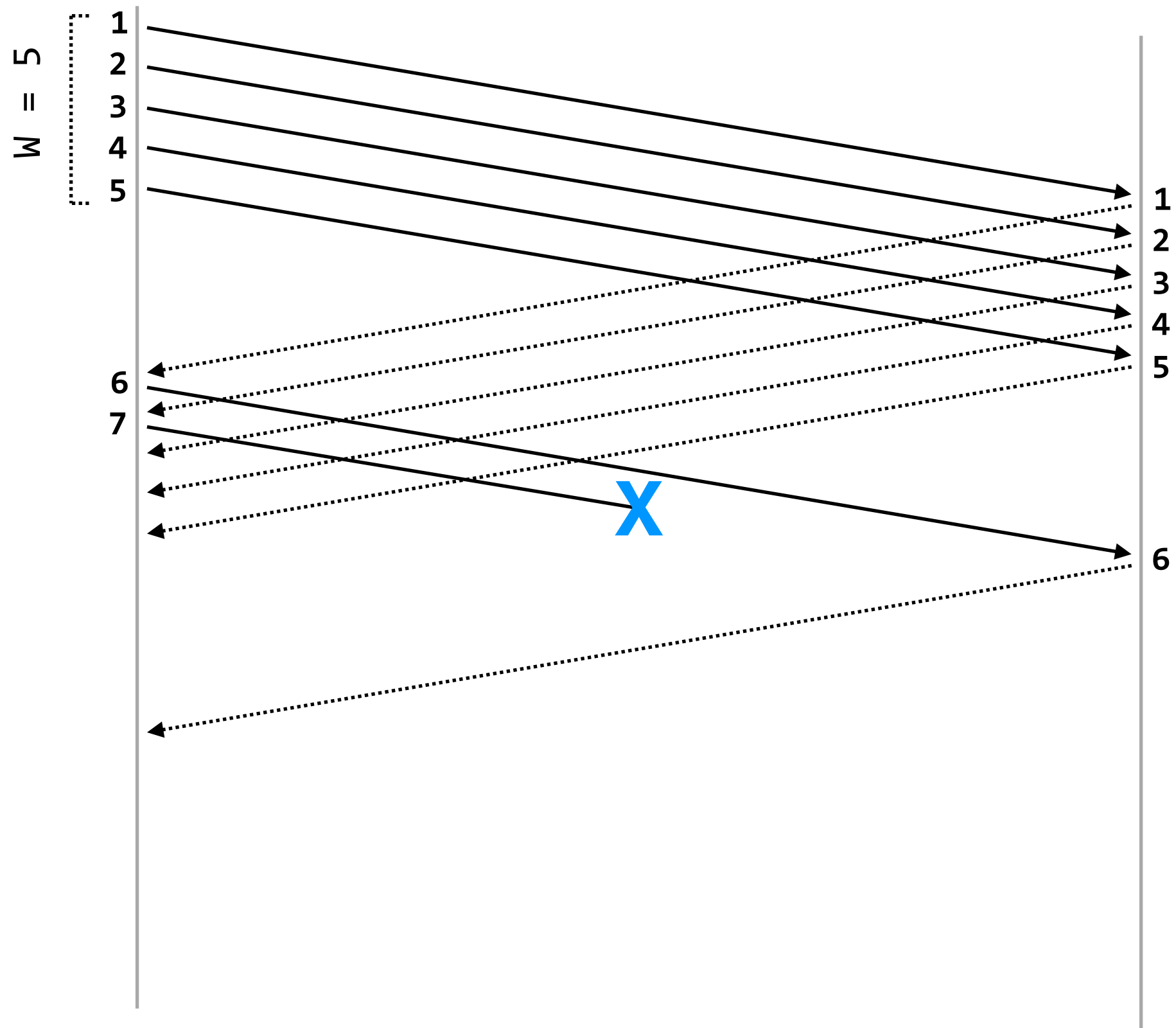
receiver

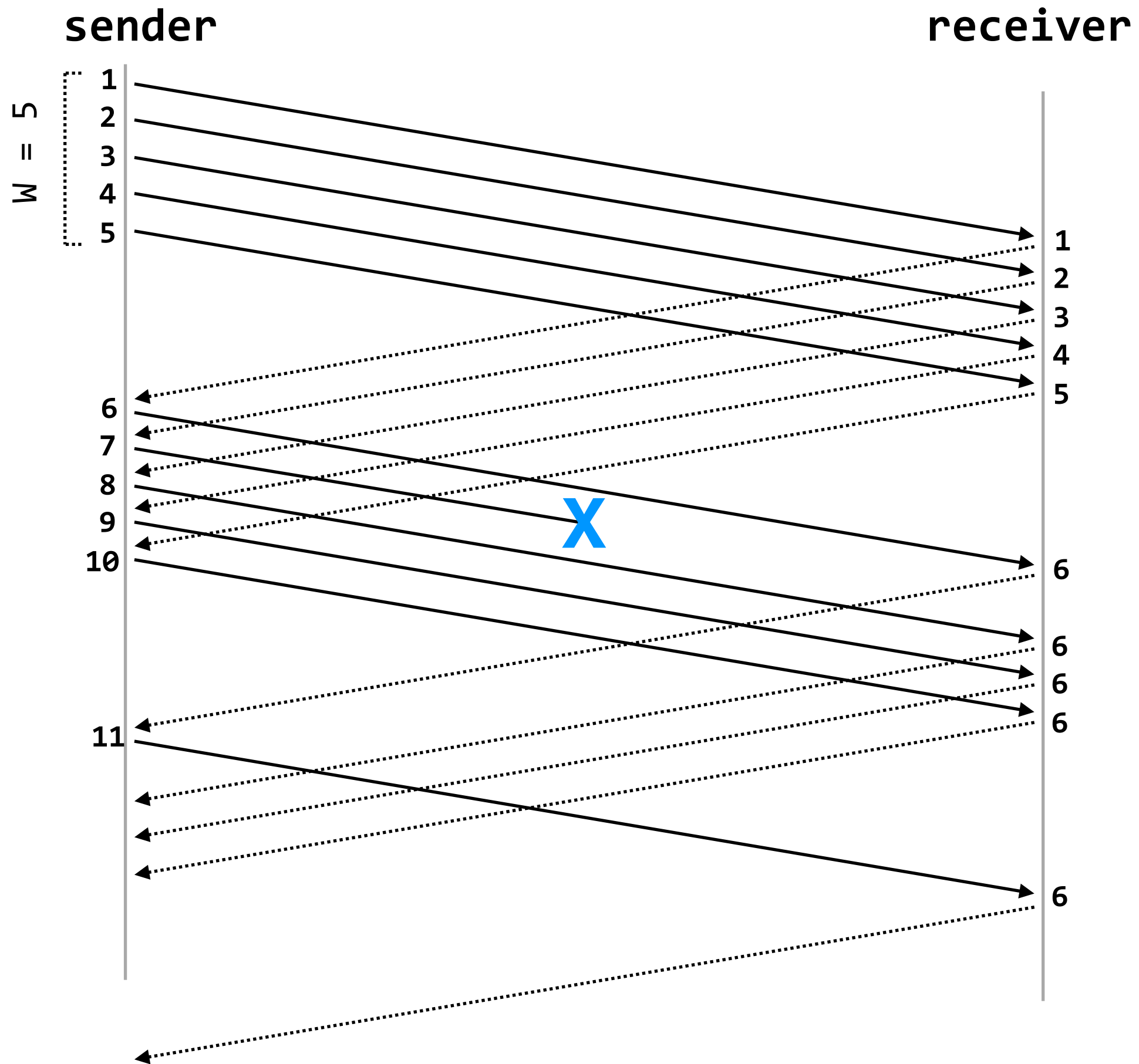


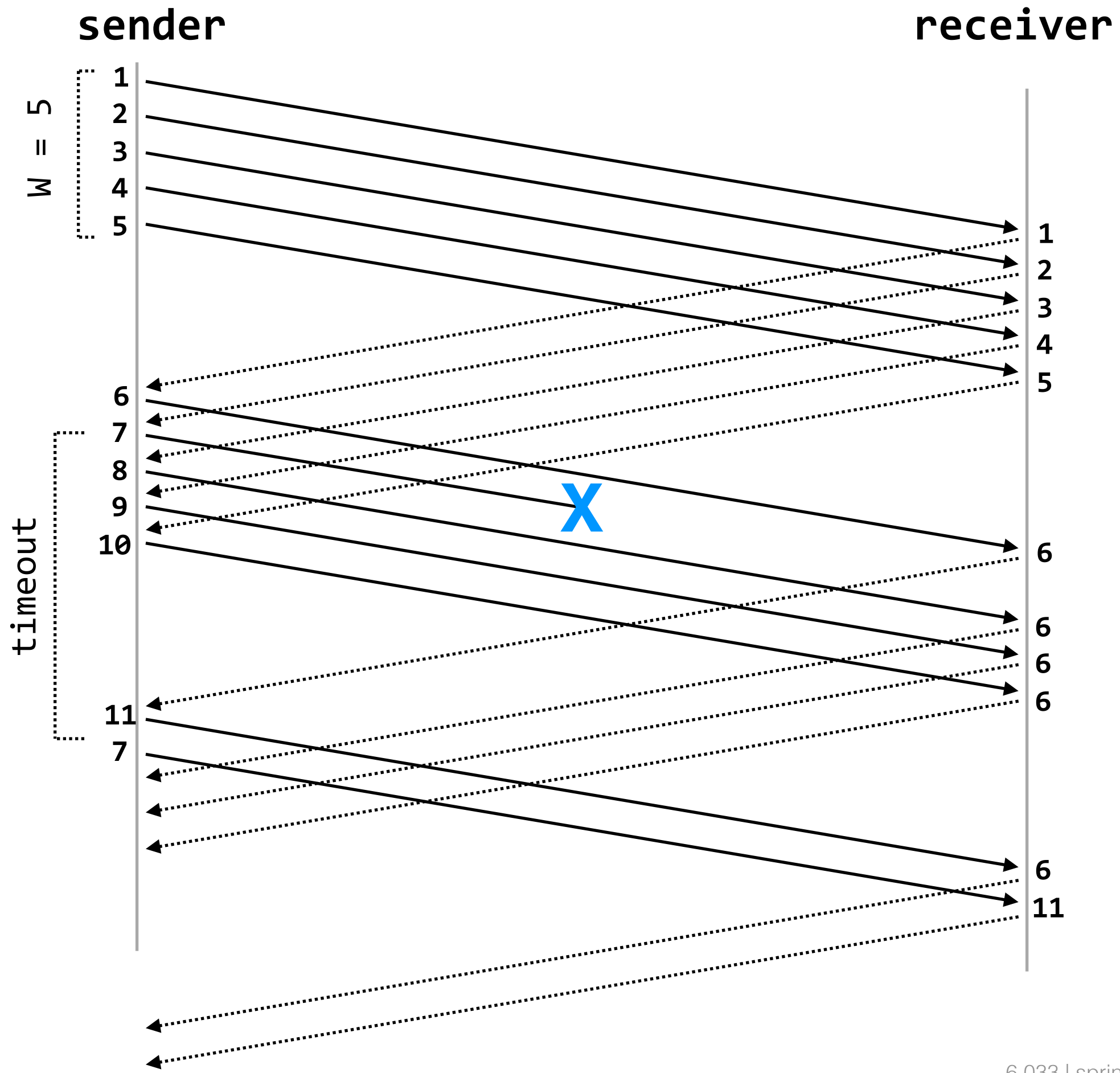


sender

receiver

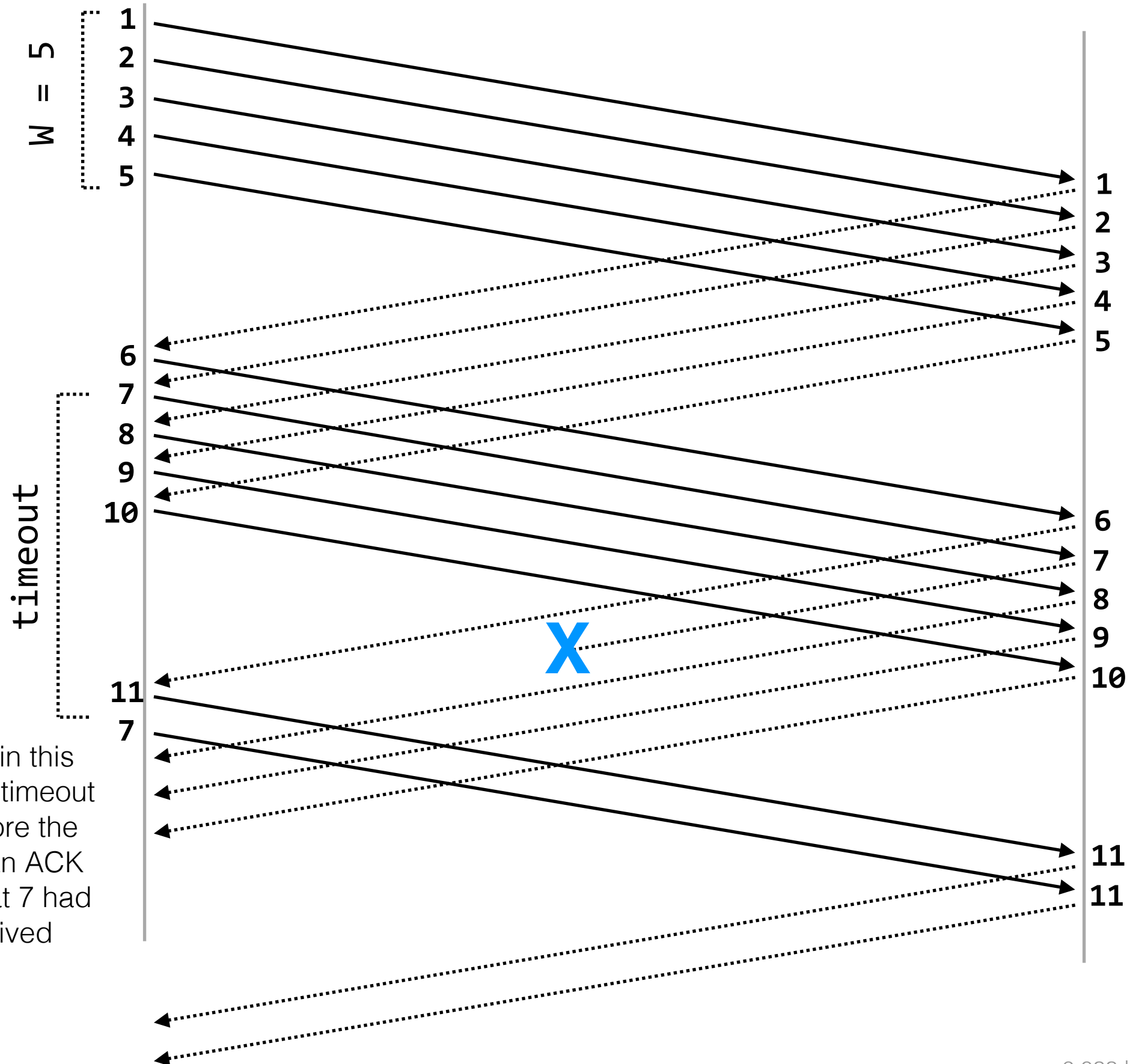


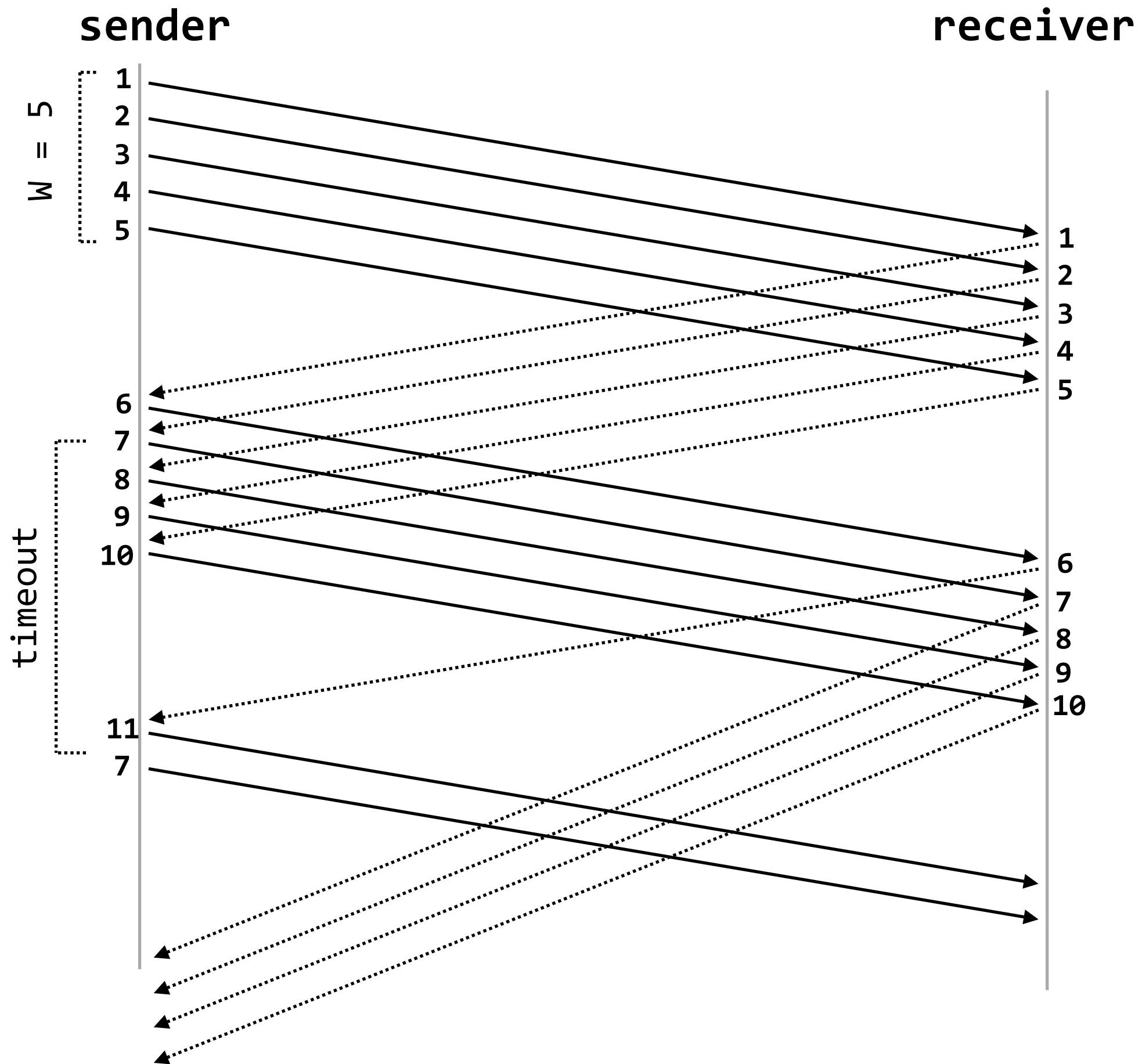




sender

receiver



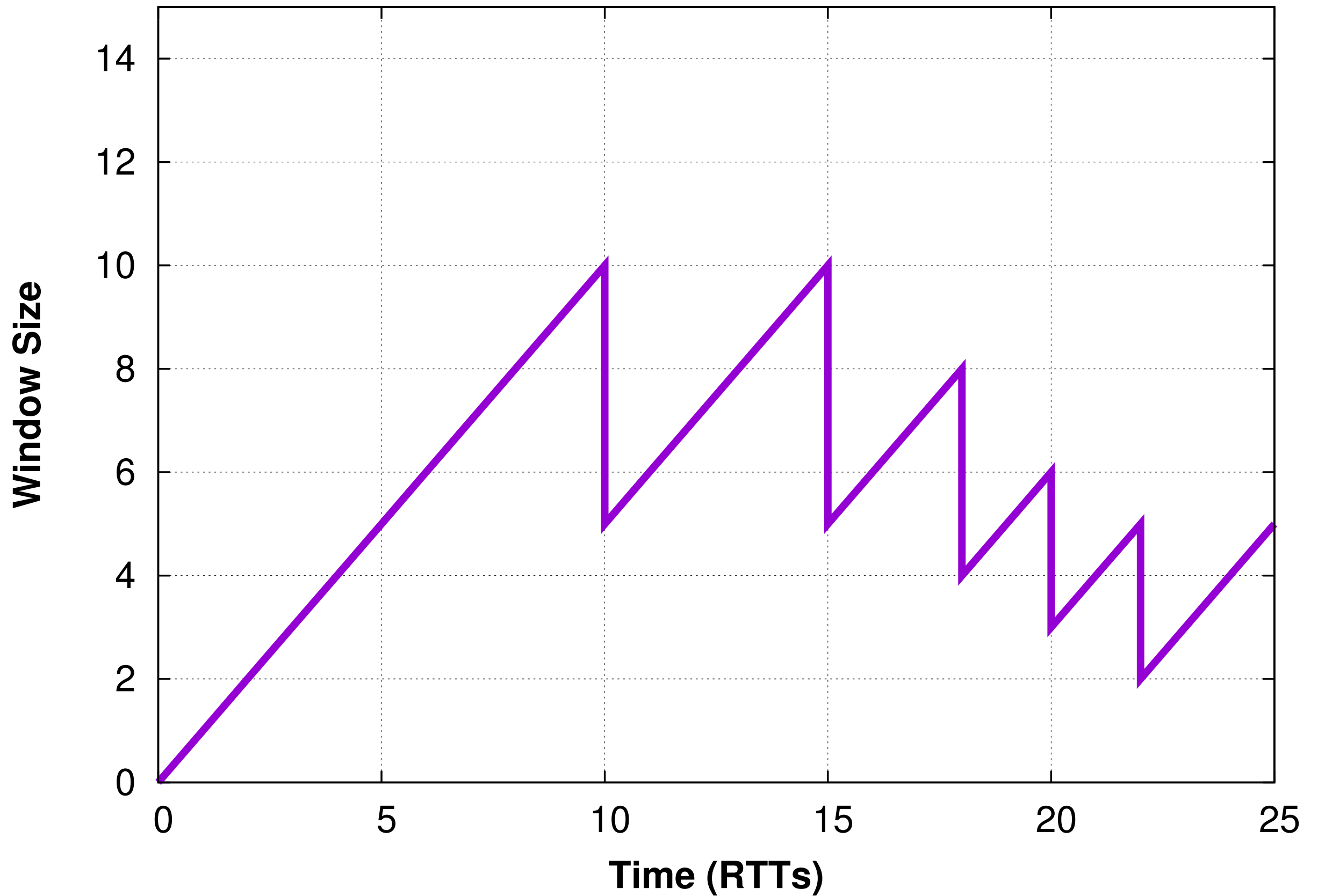


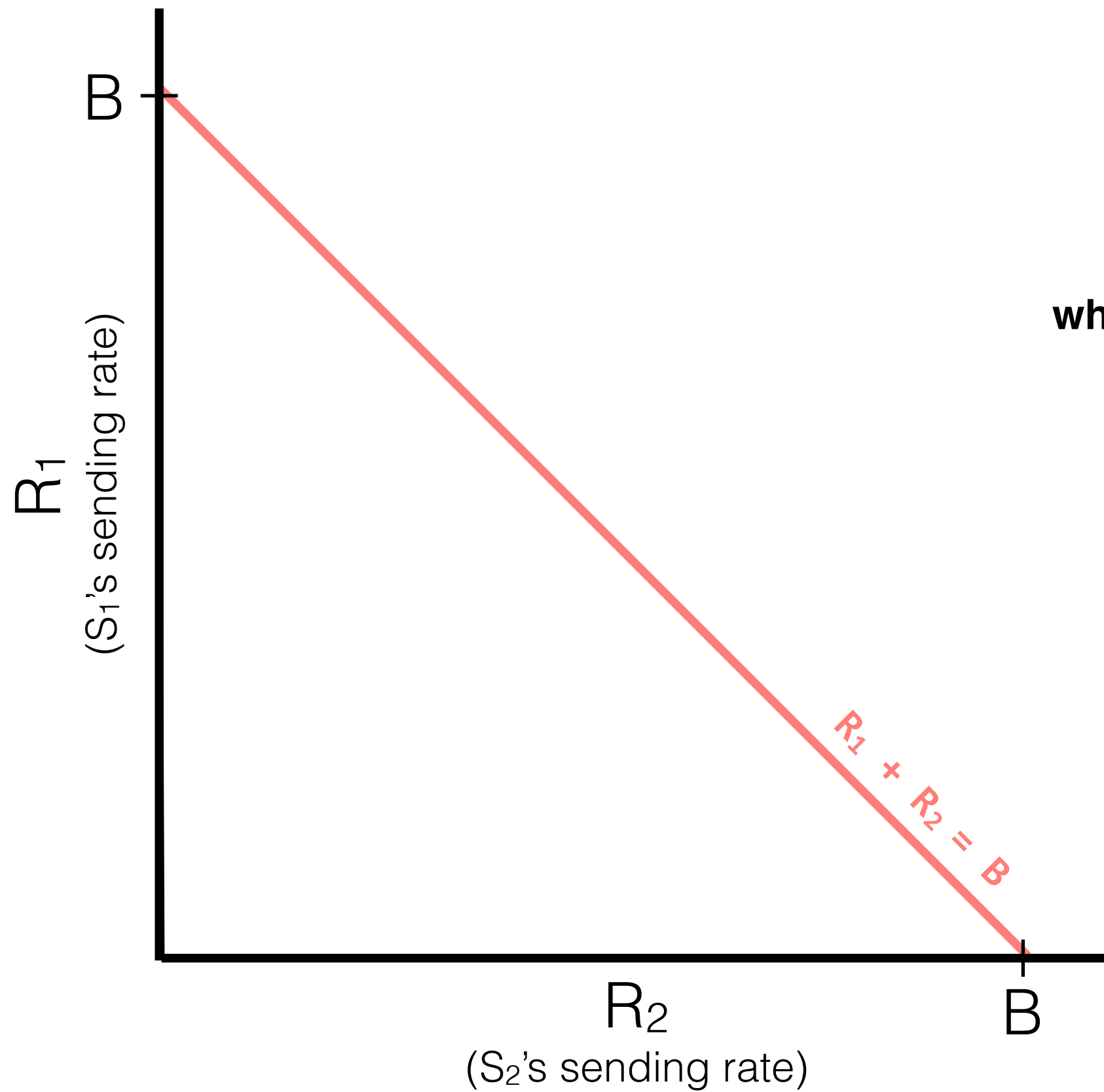
question: what is the correct value for W ?

too small \rightarrow underutilized network
too large \rightarrow congestion

question: how can a single reliable sender, using a sliding-window protocol, set its window size to maximize utilization — but prevent congestion and unfairness — given that there are many other end points using the network, all with different, changing demands?

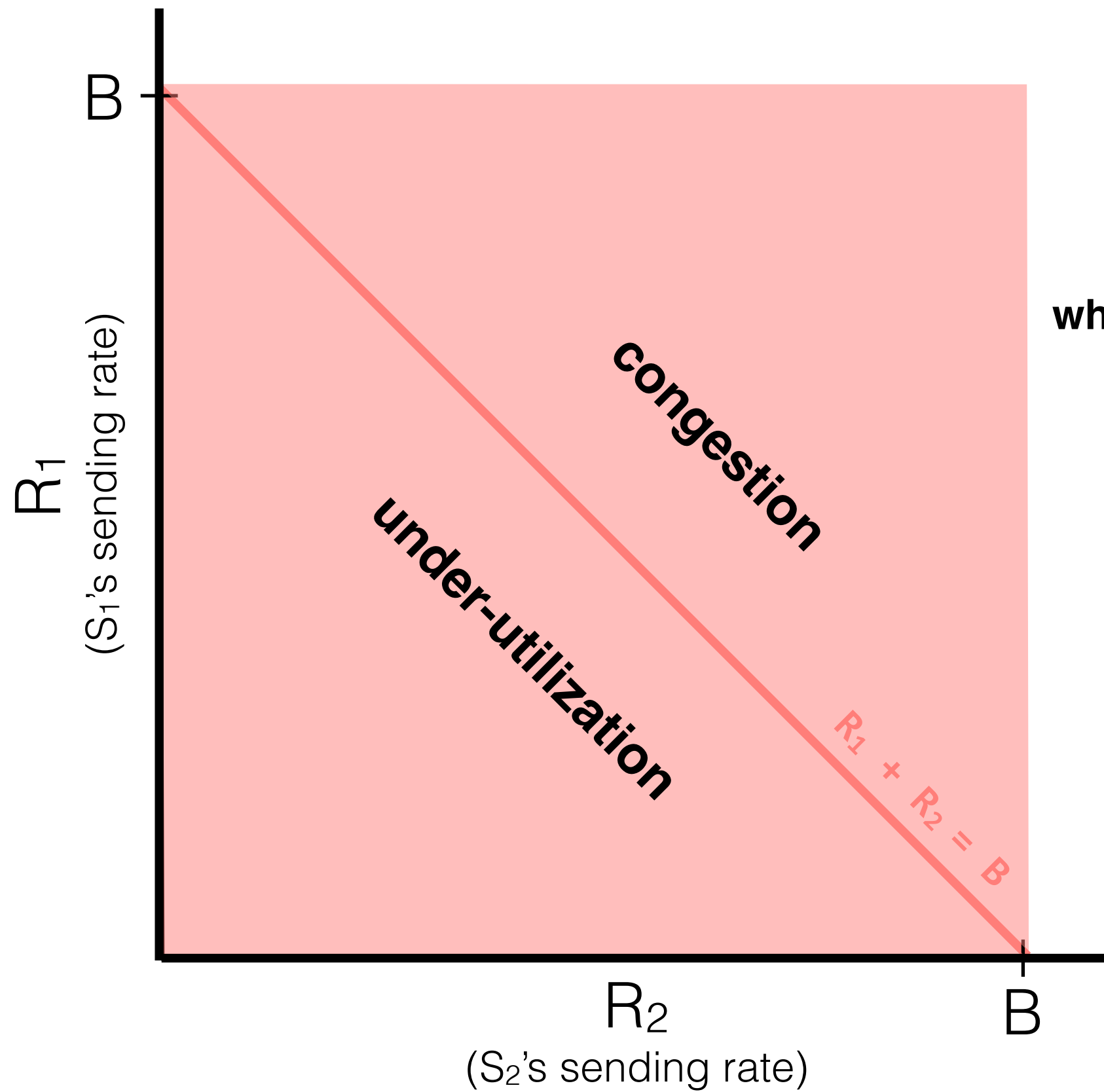
AIMD





efficiency
(utilization)

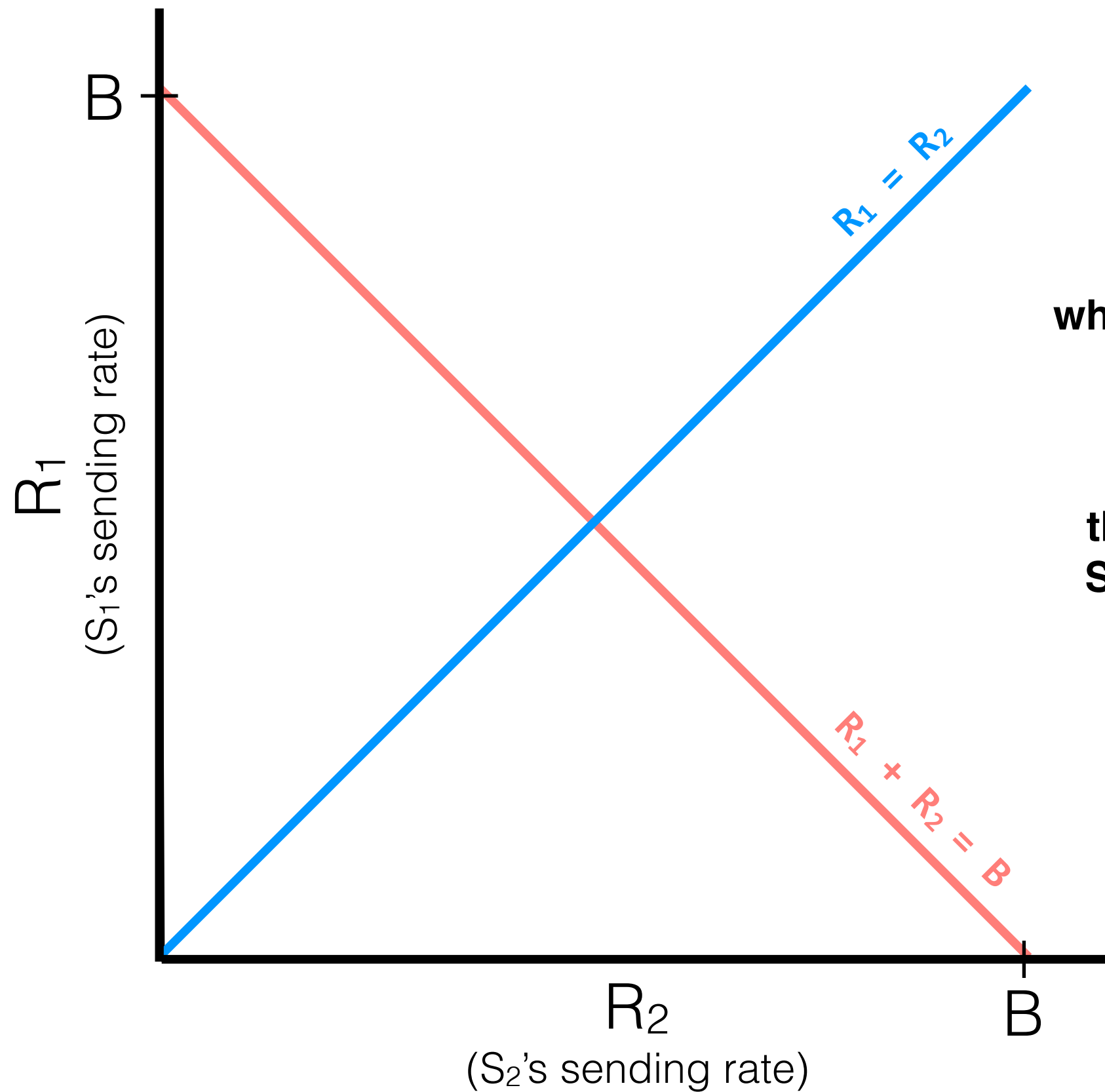
**the network is fully utilized
when the bottleneck link is “full”**



efficiency

(utilization)

**the network is fully utilized
when the bottleneck link is “full”**

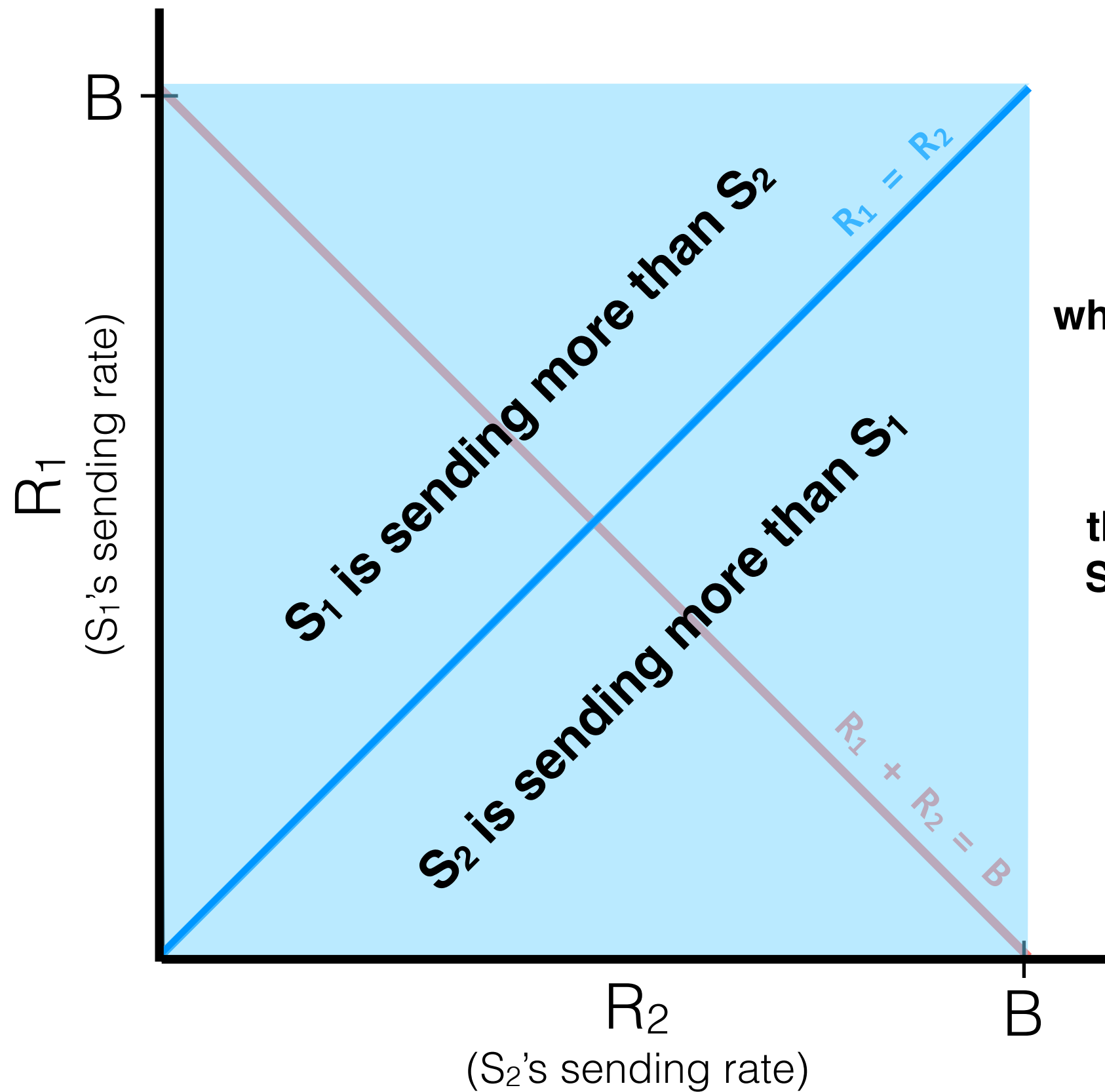


efficiency
(utilization)

**the network is fully utilized
when the bottleneck link is “full”**

fairness

**the network is fair when S₁ and
S₂ are sending at the same rate**

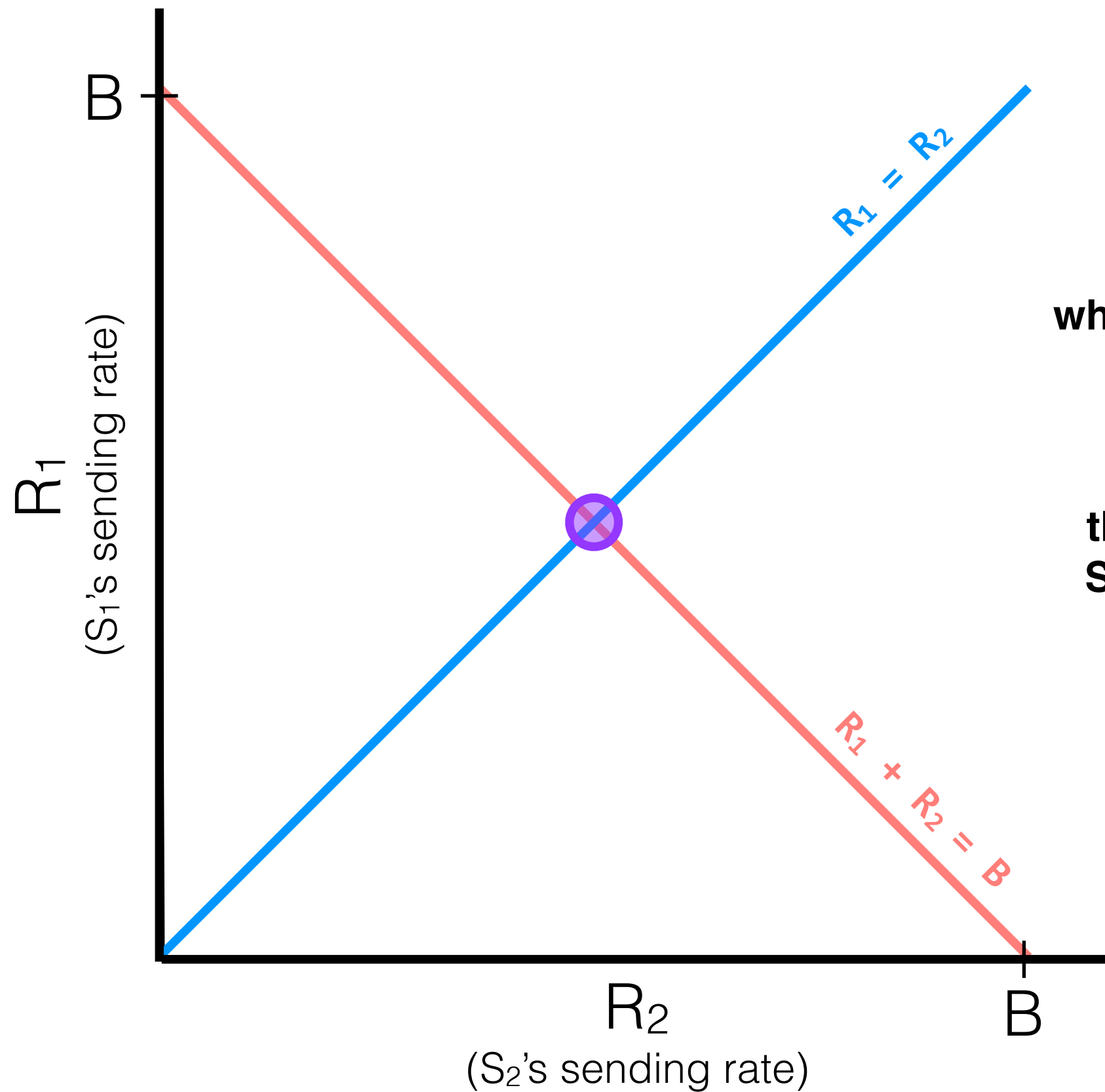


efficiency
(utilization)

the network is fully utilized
when the bottleneck link is “full”

fairness

the network is fair when S_1 and
 S_2 are sending at the same rate

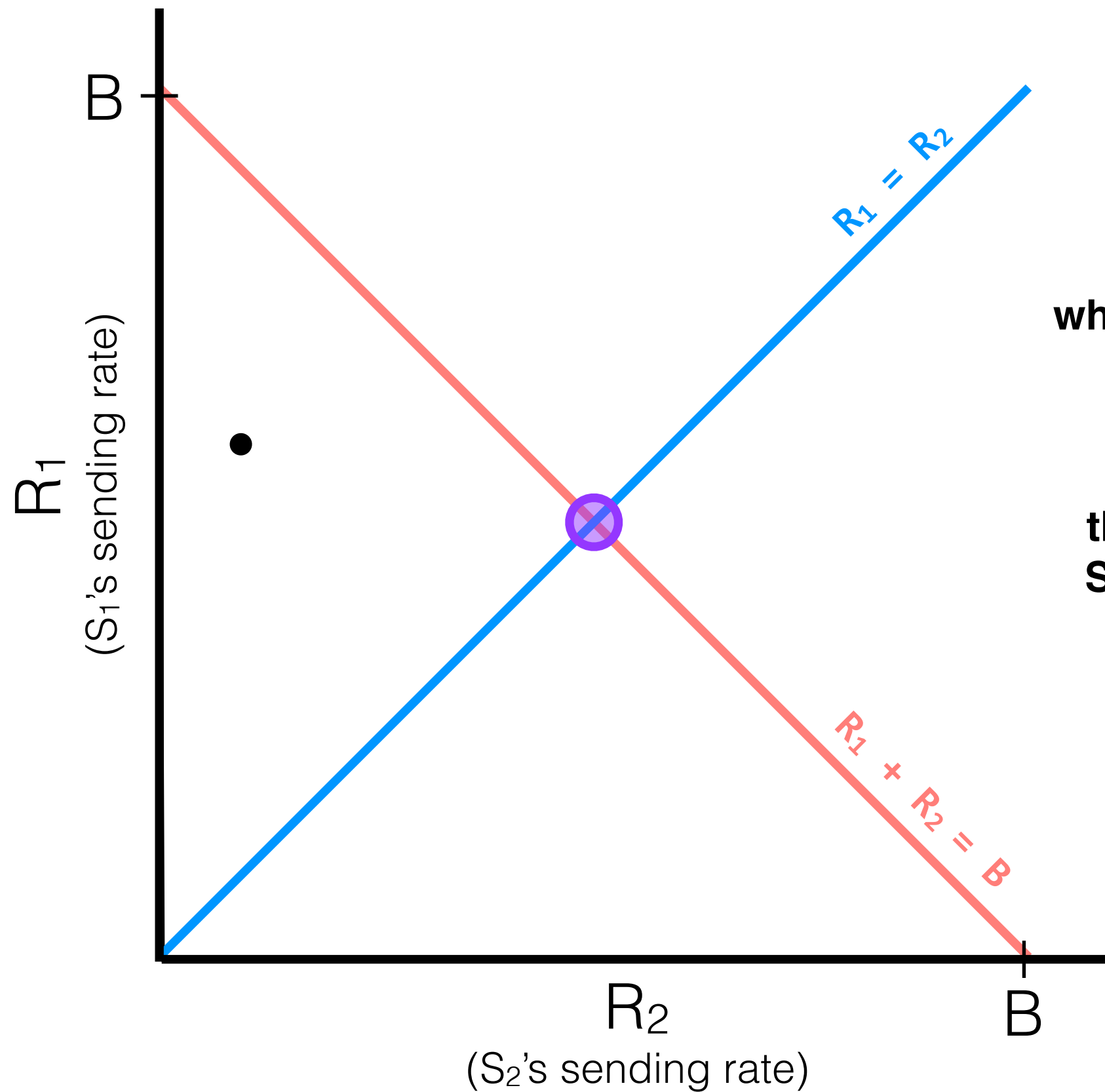


efficiency
(utilization)

**the network is fully utilized
when the bottleneck link is “full”**

fairness

**the network is fair when S_1 and
 S_2 are sending at the same rate**

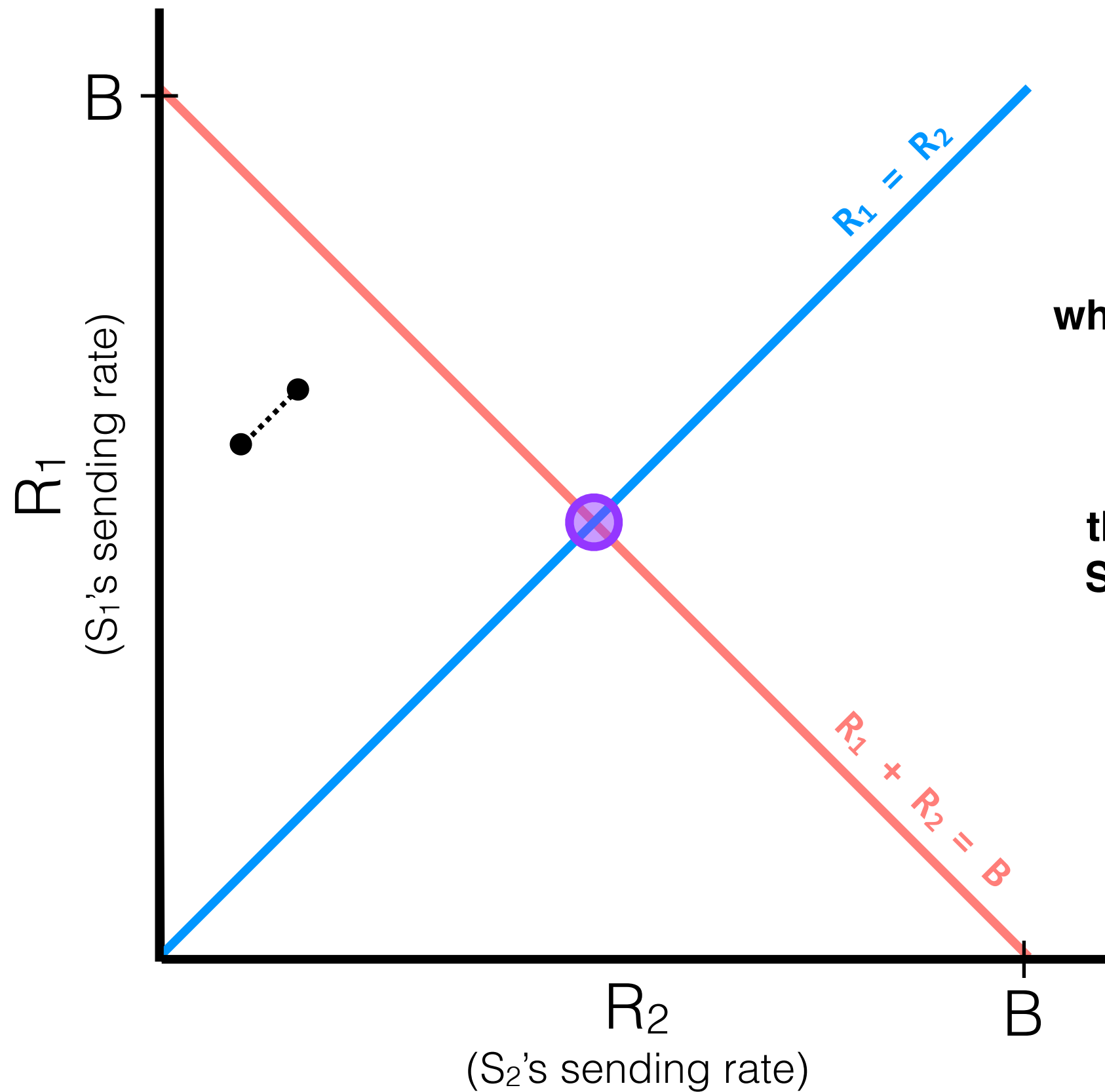


efficiency
(utilization)

**the network is fully utilized
when the bottleneck link is “full”**

fairness

**the network is fair when S_1 and
 S_2 are sending at the same rate**

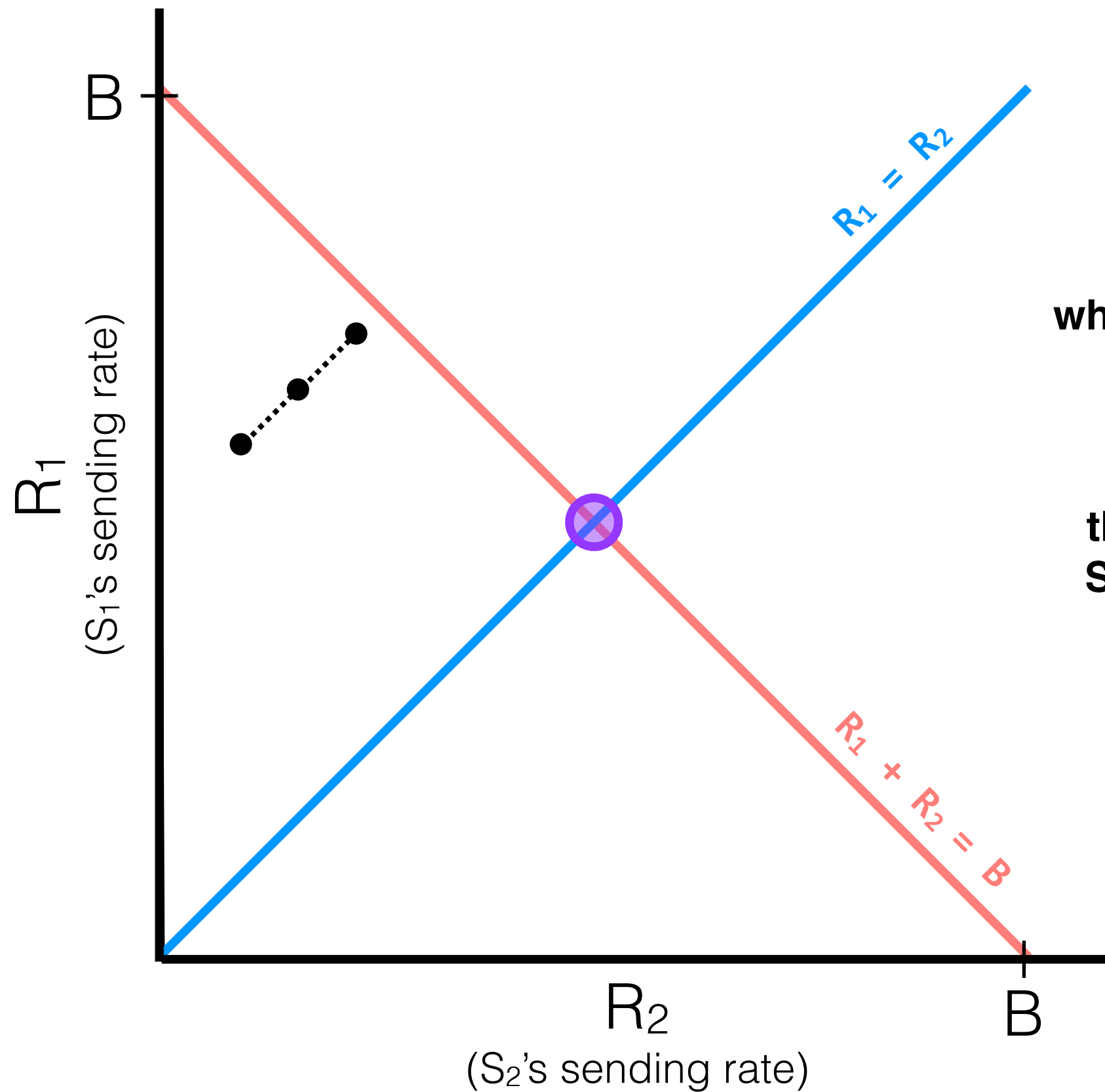


efficiency
(utilization)

**the network is fully utilized
when the bottleneck link is “full”**

fairness

**the network is fair when S_1 and
 S_2 are sending at the same rate**

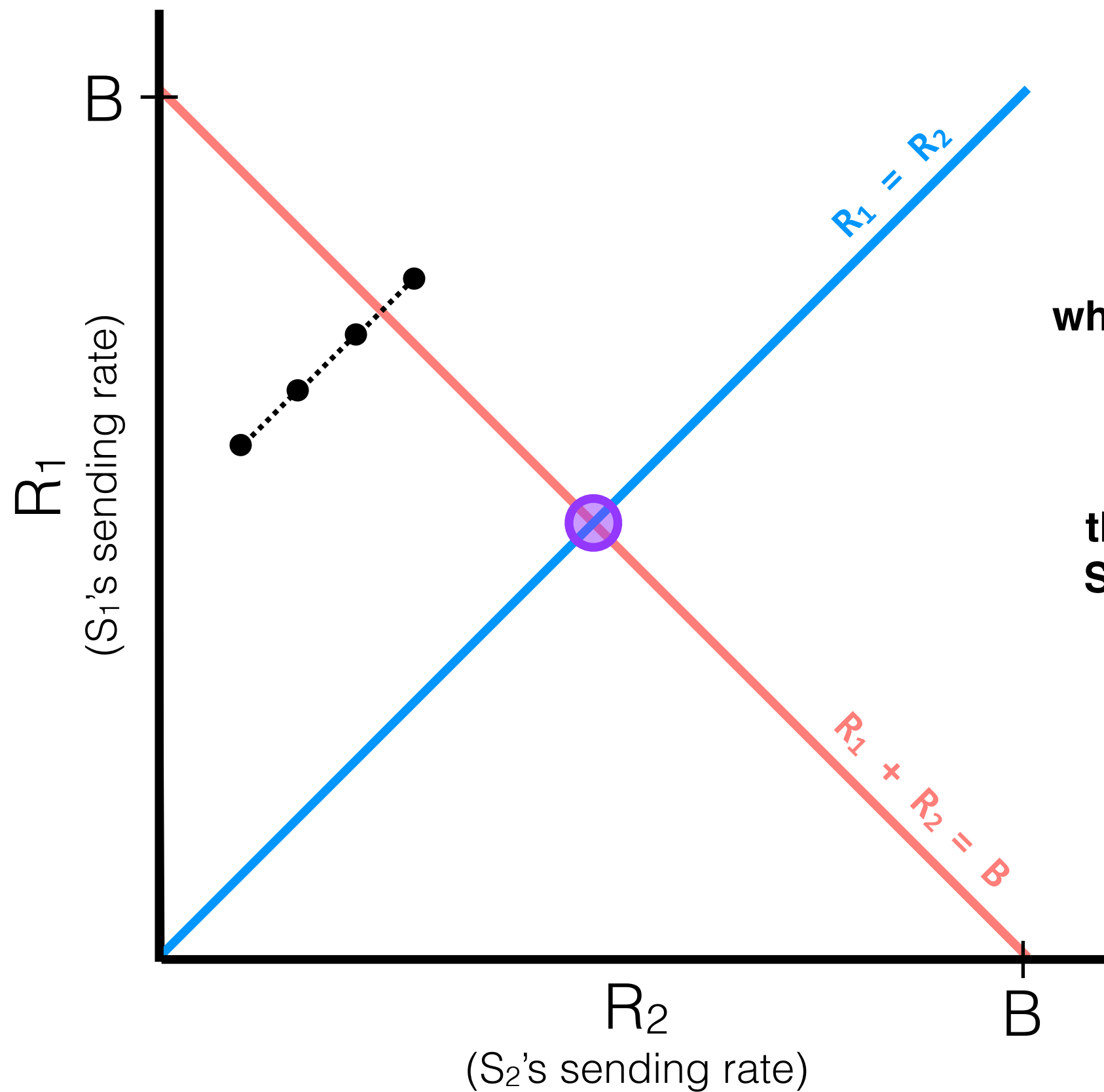


efficiency
(utilization)

**the network is fully utilized
when the bottleneck link is “full”**

fairness

**the network is fair when S_1 and
 S_2 are sending at the same rate**

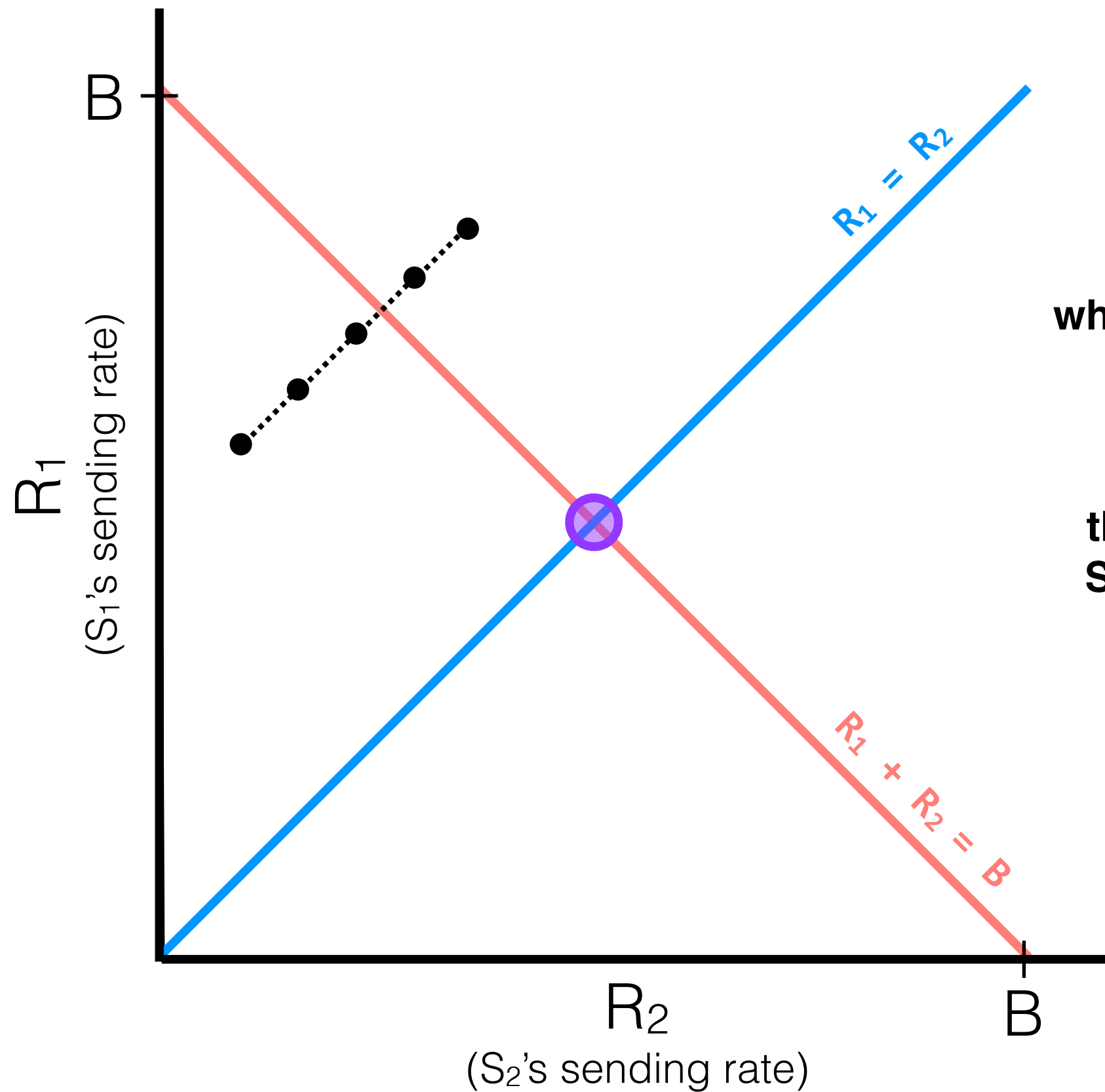


efficiency
(utilization)

**the network is fully utilized
when the bottleneck link is “full”**

fairness

**the network is fair when S_1 and
 S_2 are sending at the same rate**

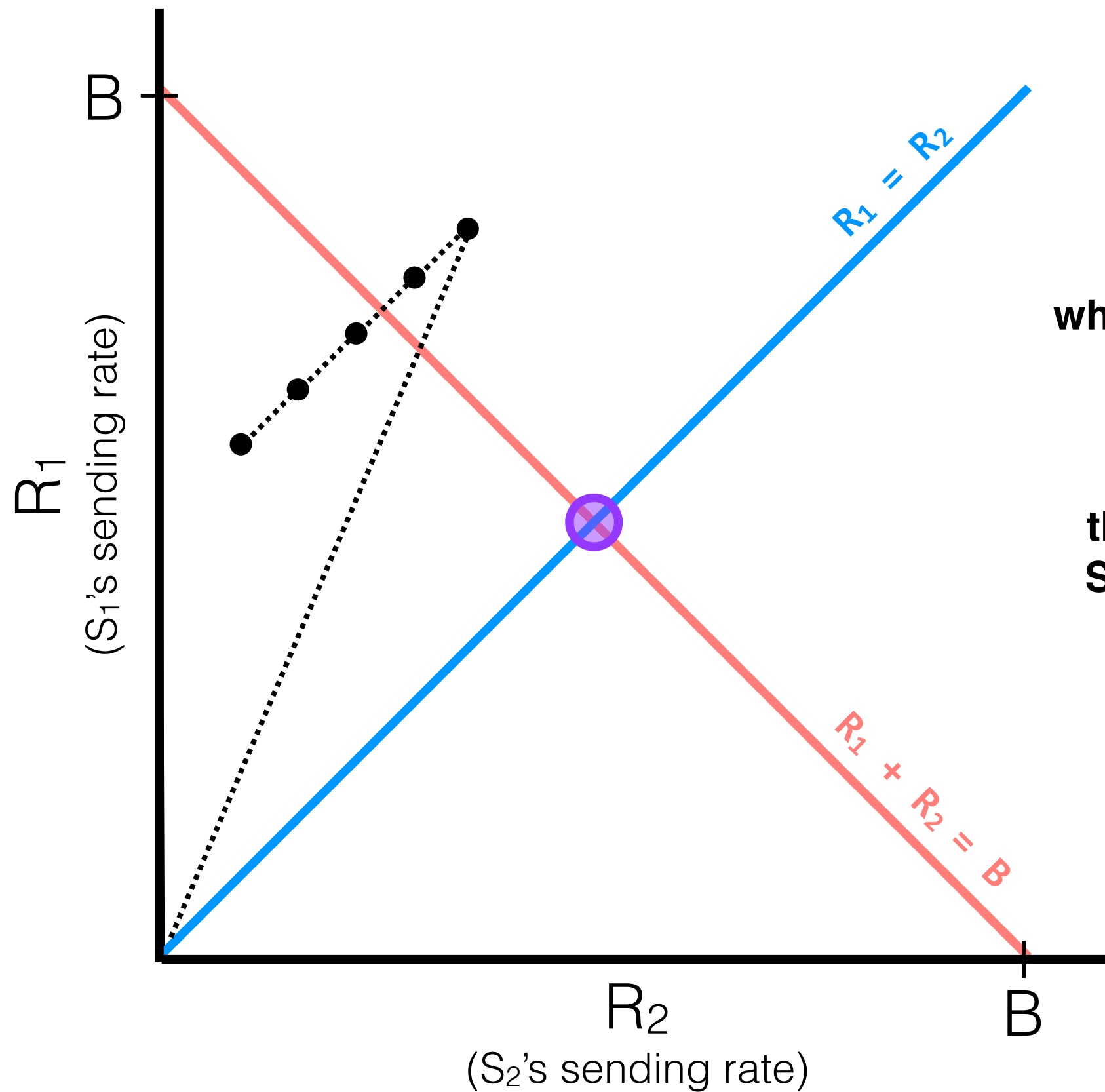


efficiency
(utilization)

the network is fully utilized
when the bottleneck link is “full”

fairness

the network is fair when S_1 and
 S_2 are sending at the same rate

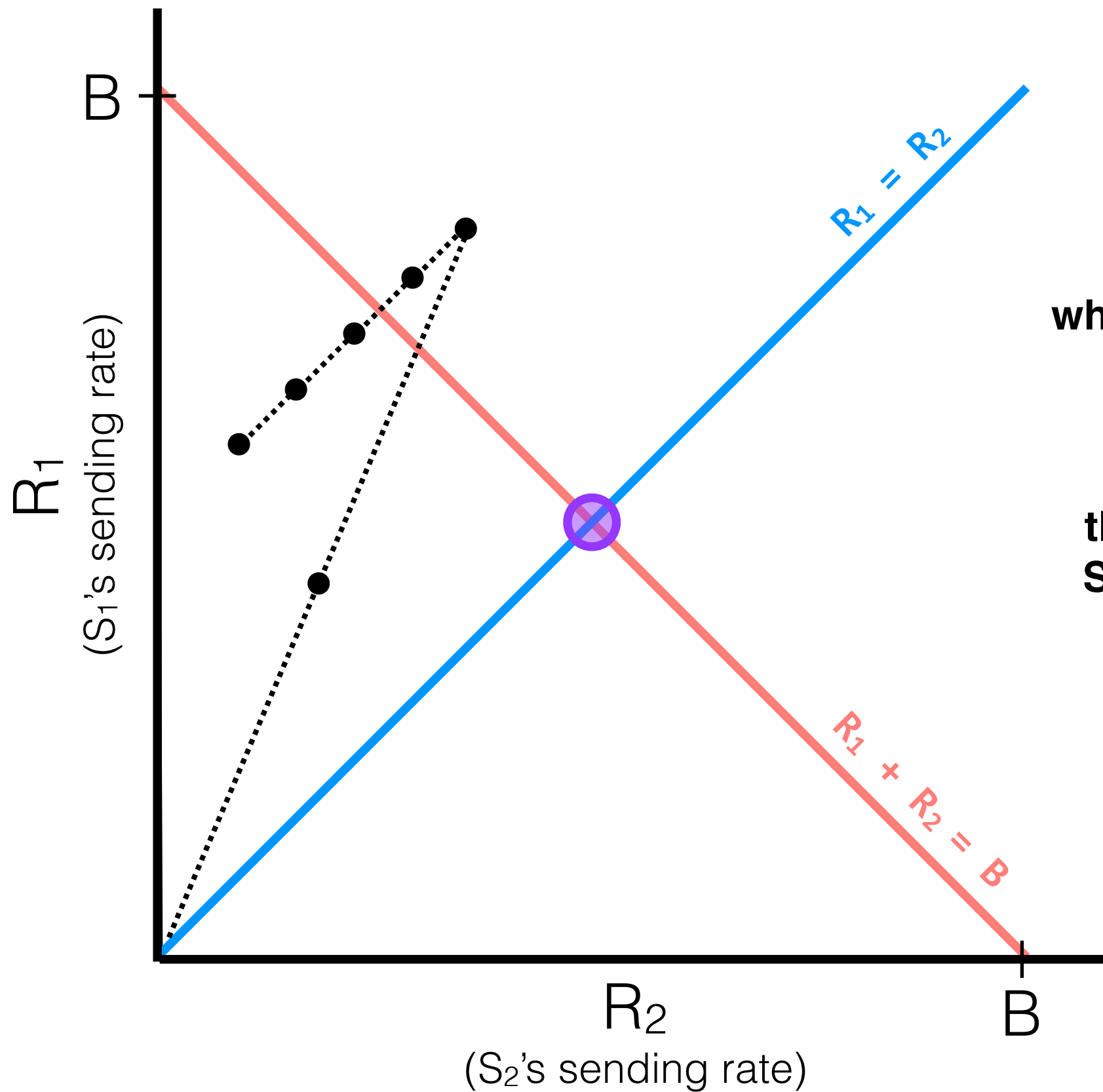


efficiency
(utilization)

**the network is fully utilized
when the bottleneck link is “full”**

fairness

**the network is fair when S₁ and
S₂ are sending at the same rate**

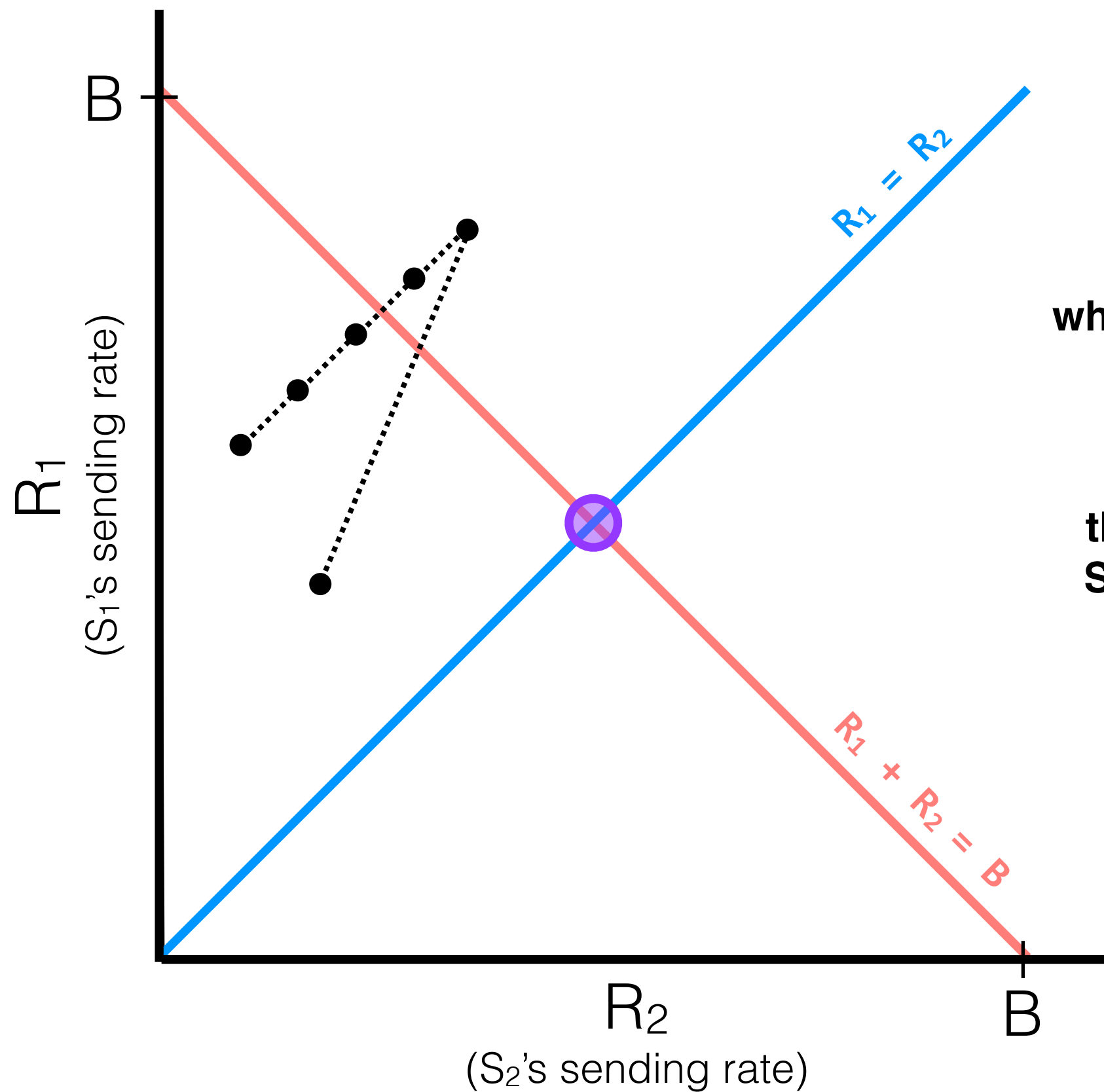


efficiency
(utilization)

the network is fully utilized
when the bottleneck link is “full”

fairness

the network is fair when S_1 and
 S_2 are sending at the same rate

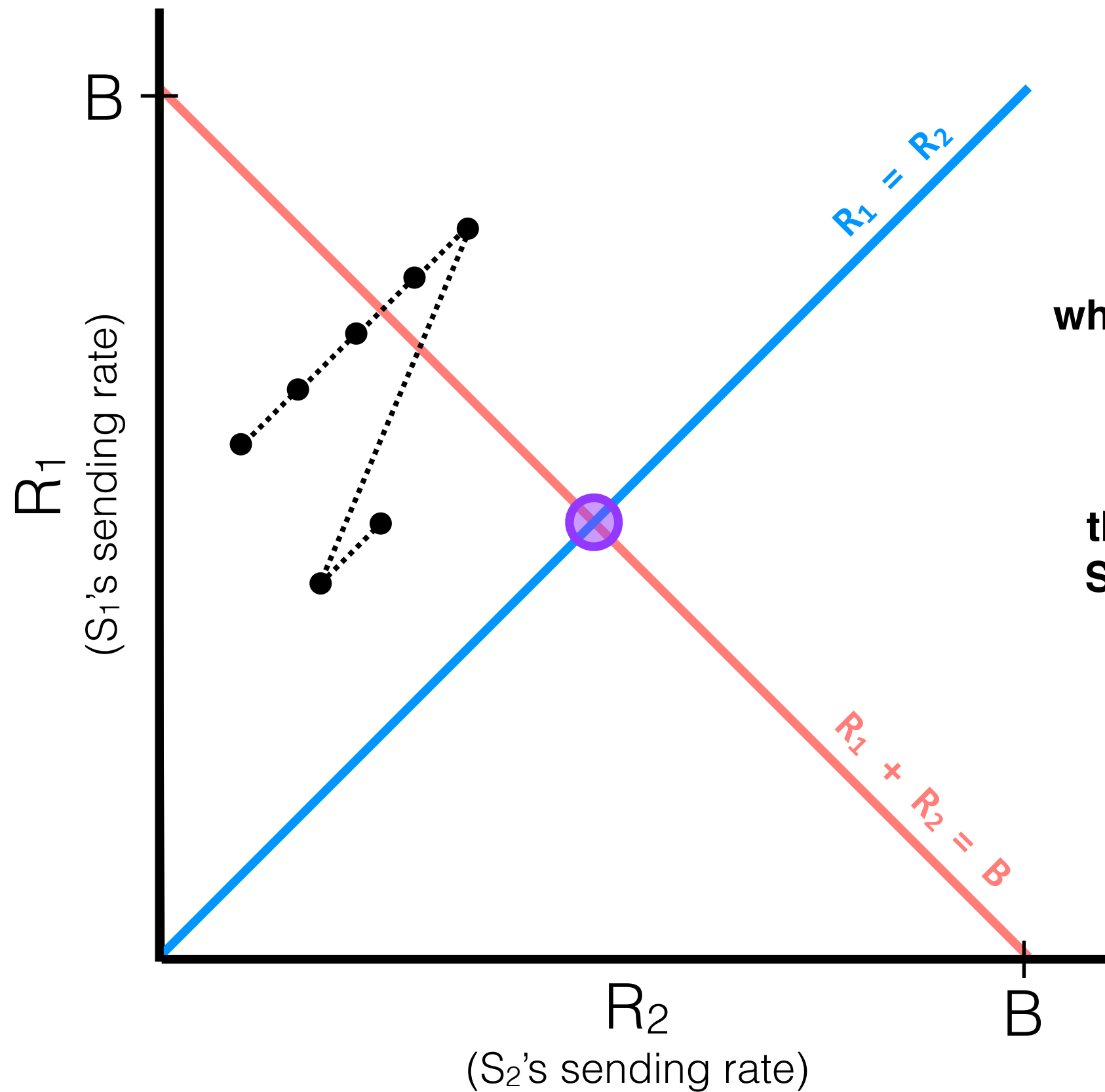


efficiency
(utilization)

the network is fully utilized
when the bottleneck link is “full”

fairness

the network is fair when S_1 and
 S_2 are sending at the same rate

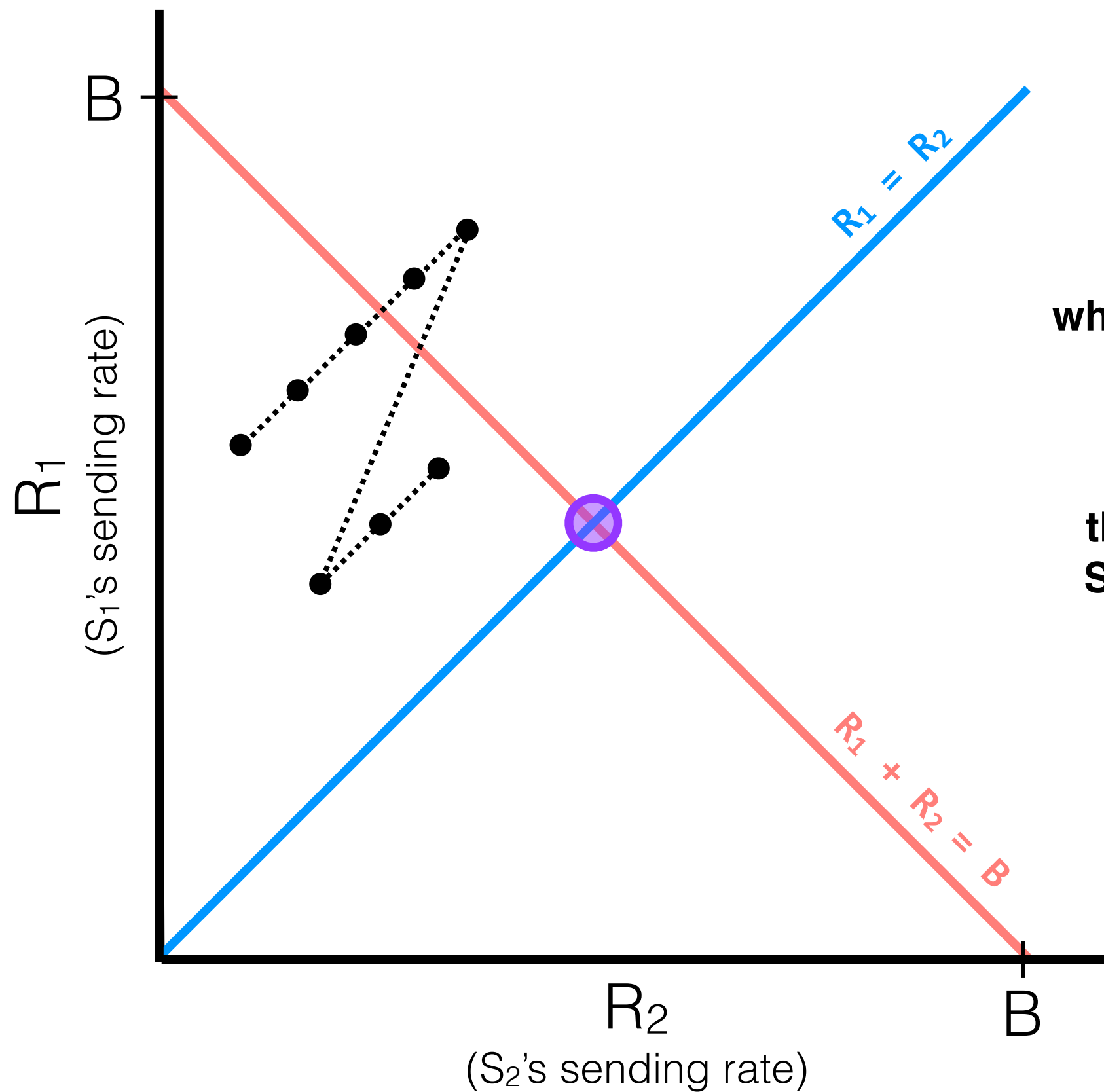


efficiency
(utilization)

**the network is fully utilized
when the bottleneck link is “full”**

fairness

**the network is fair when S_1 and
 S_2 are sending at the same rate**

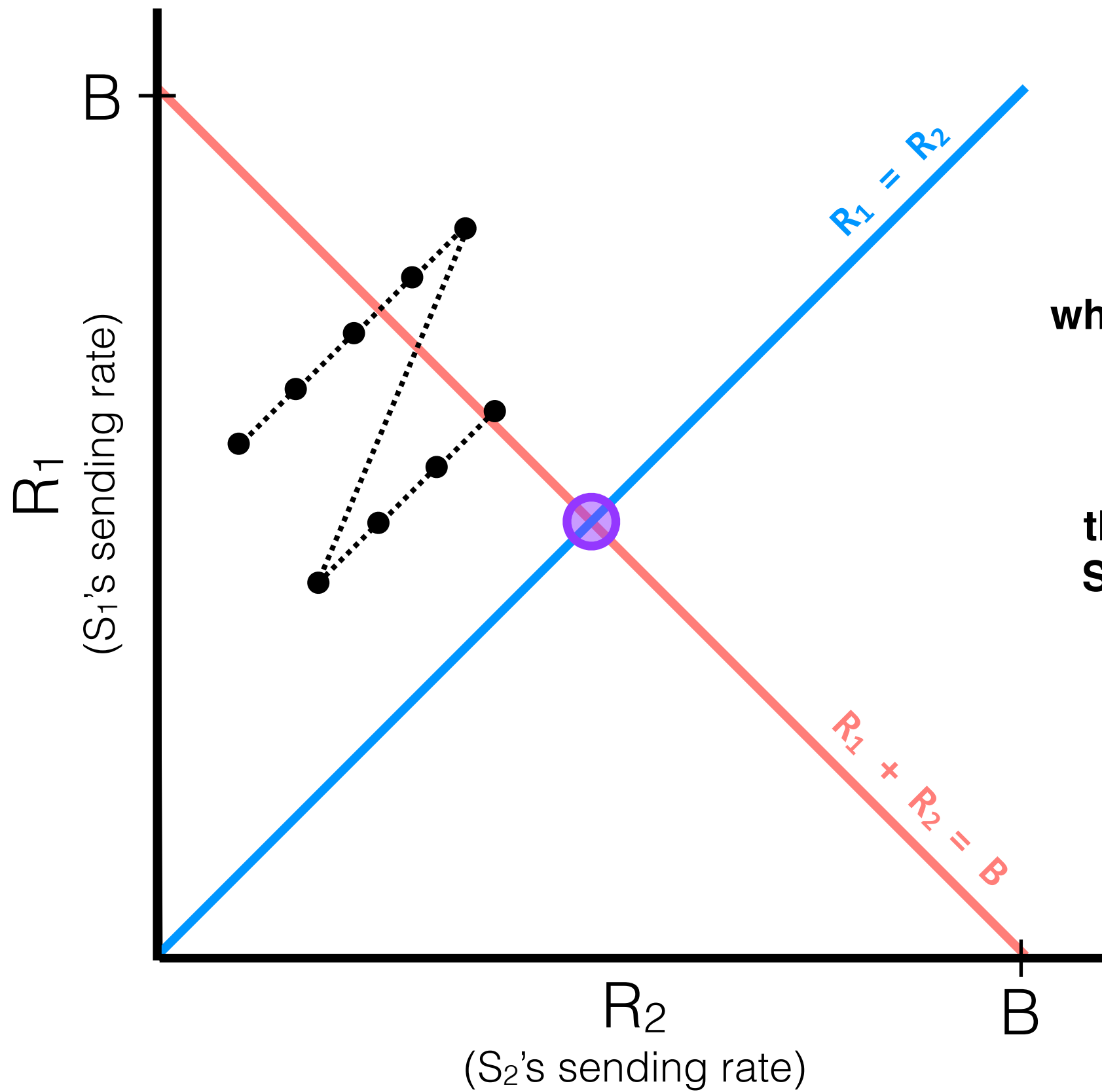


efficiency
(utilization)

**the network is fully utilized
when the bottleneck link is “full”**

fairness

**the network is fair when S_1 and
 S_2 are sending at the same rate**

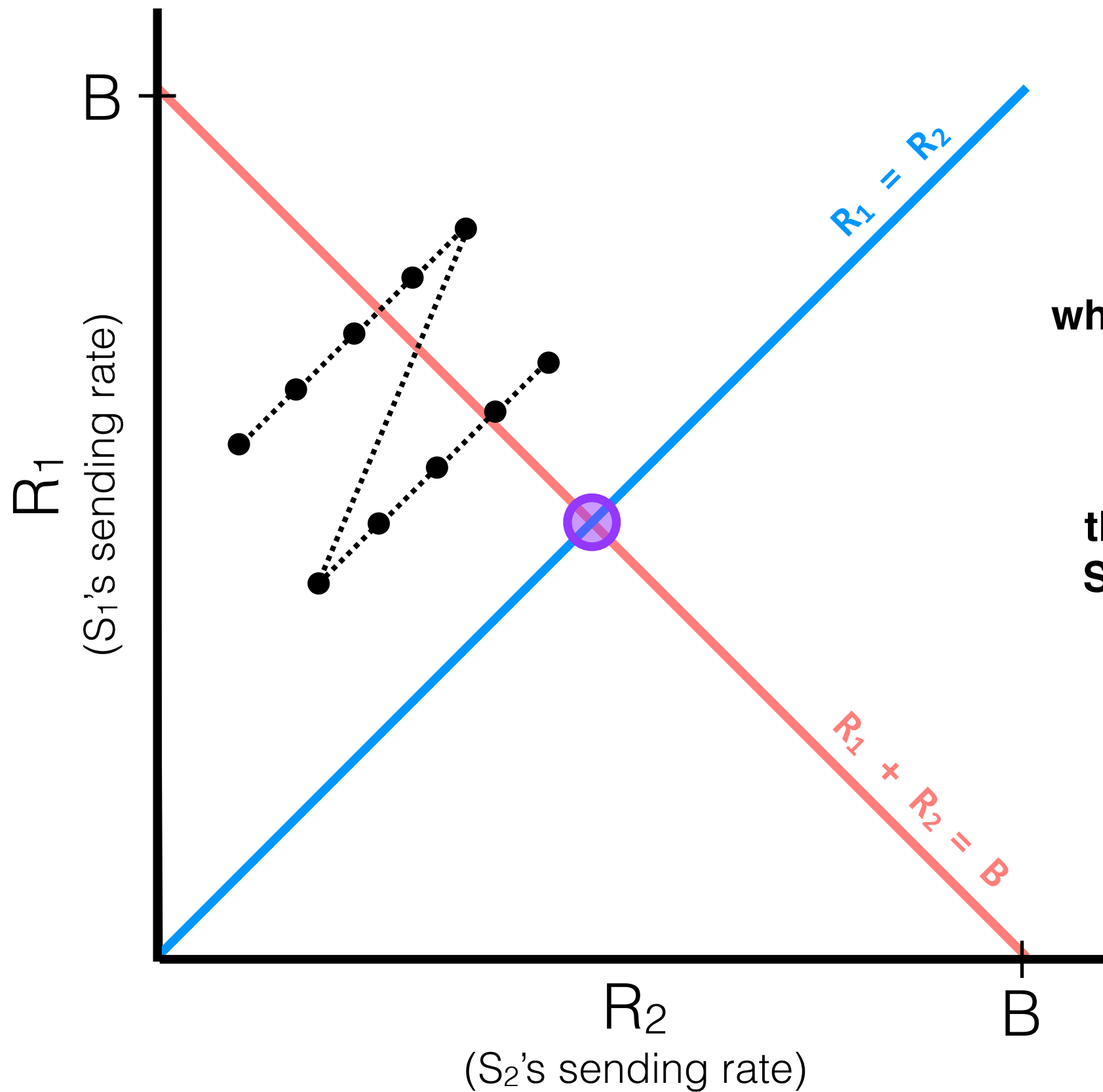


efficiency
(utilization)

the network is fully utilized
when the bottleneck link is “full”

fairness

the network is fair when S_1 and
 S_2 are sending at the same rate

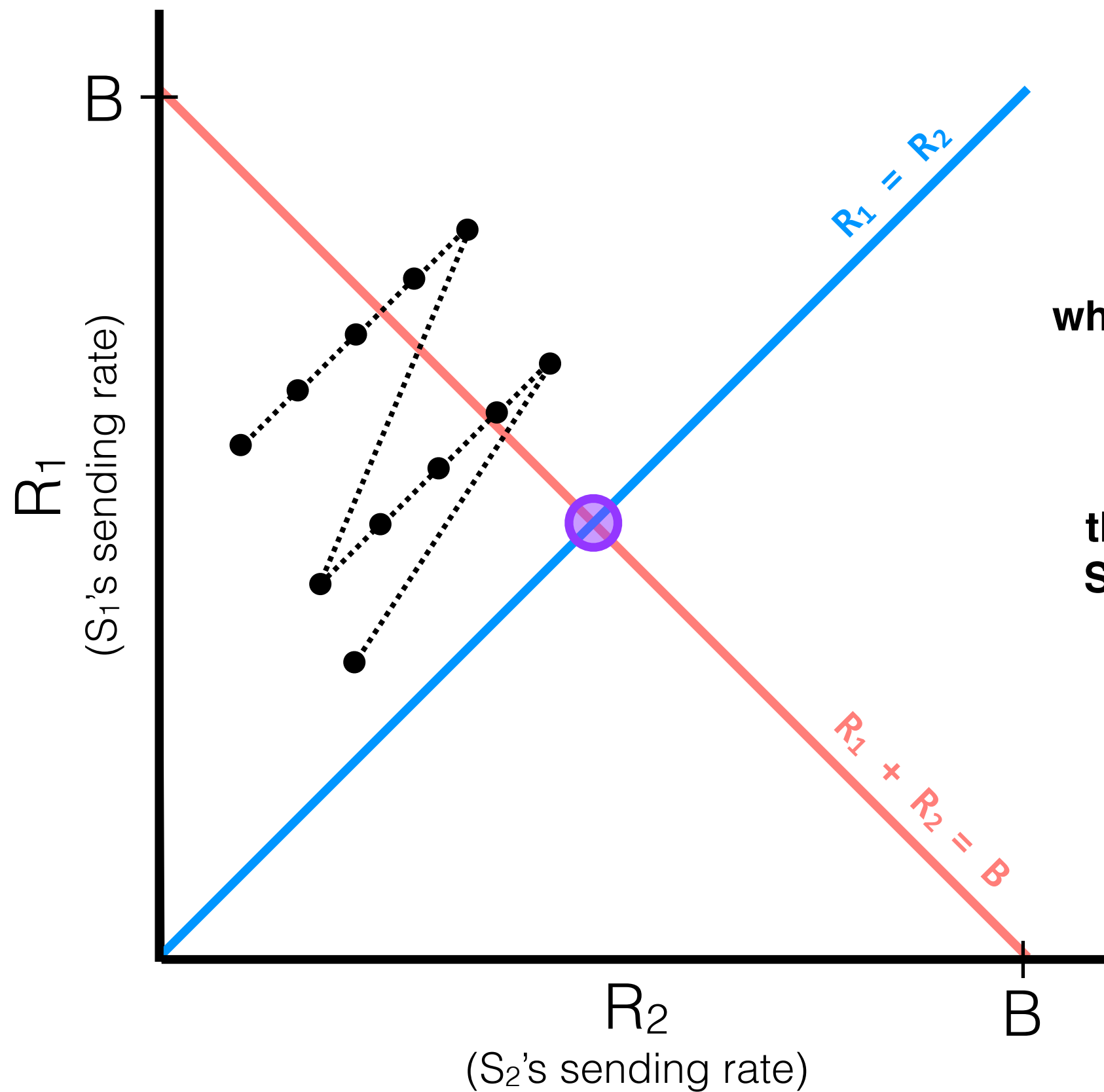


efficiency
(utilization)

the network is fully utilized
when the bottleneck link is “full”

fairness

the network is fair when S_1 and
 S_2 are sending at the same rate

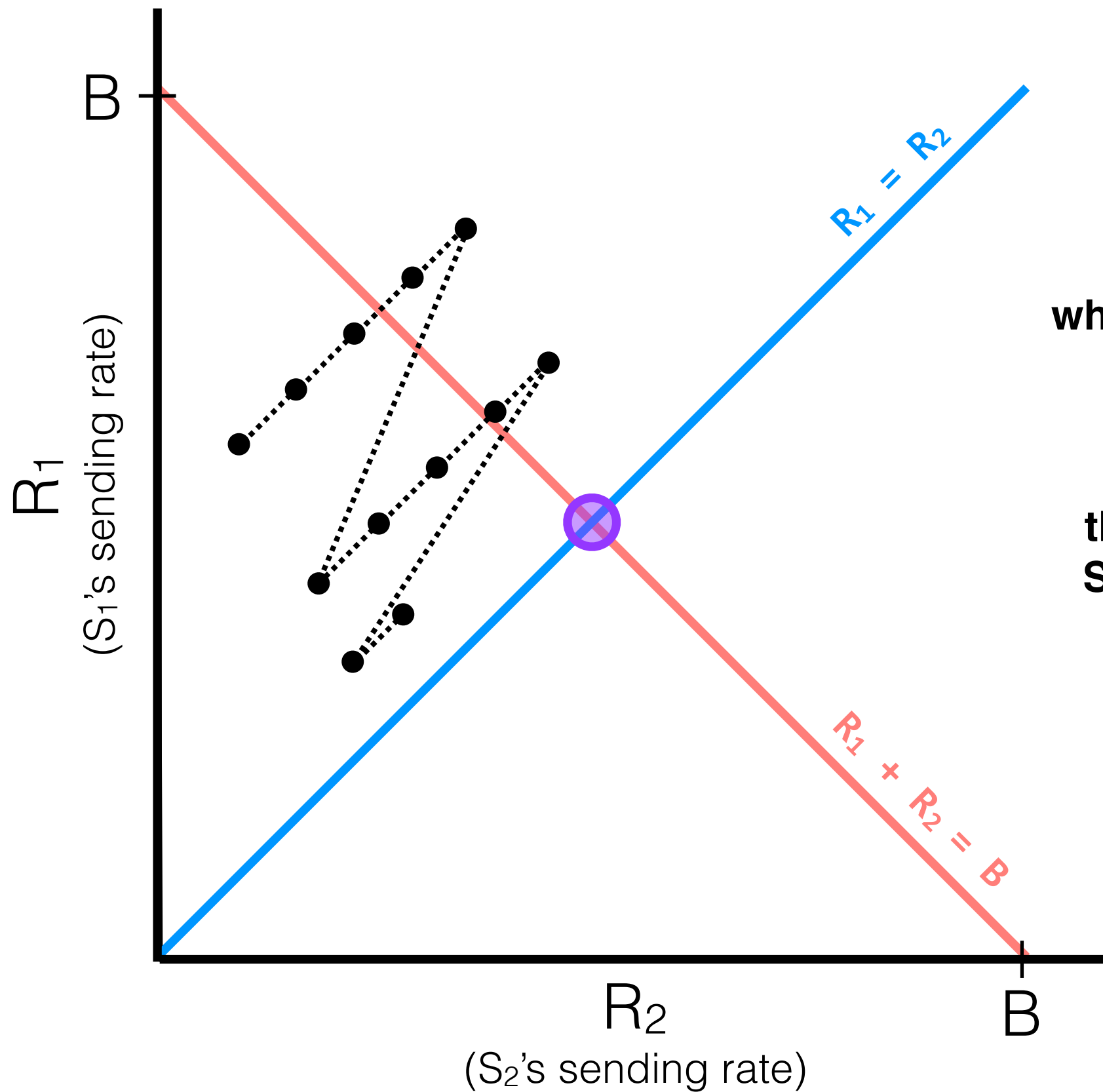


efficiency
(utilization)

the network is fully utilized
when the bottleneck link is “full”

fairness

the network is fair when S_1 and
 S_2 are sending at the same rate

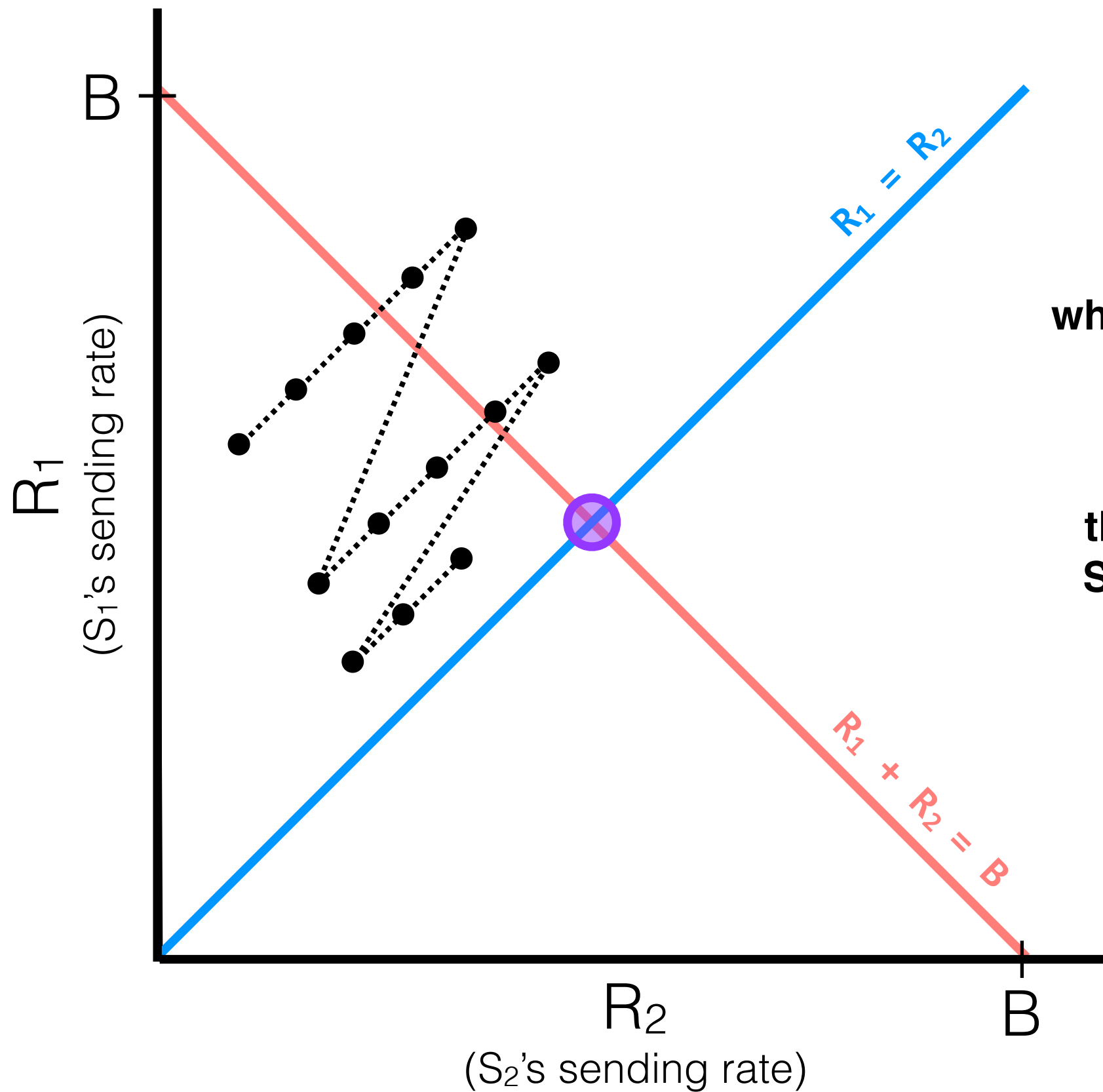


efficiency
(utilization)

the network is fully utilized
when the bottleneck link is “full”

fairness

the network is fair when S_1 and
 S_2 are sending at the same rate

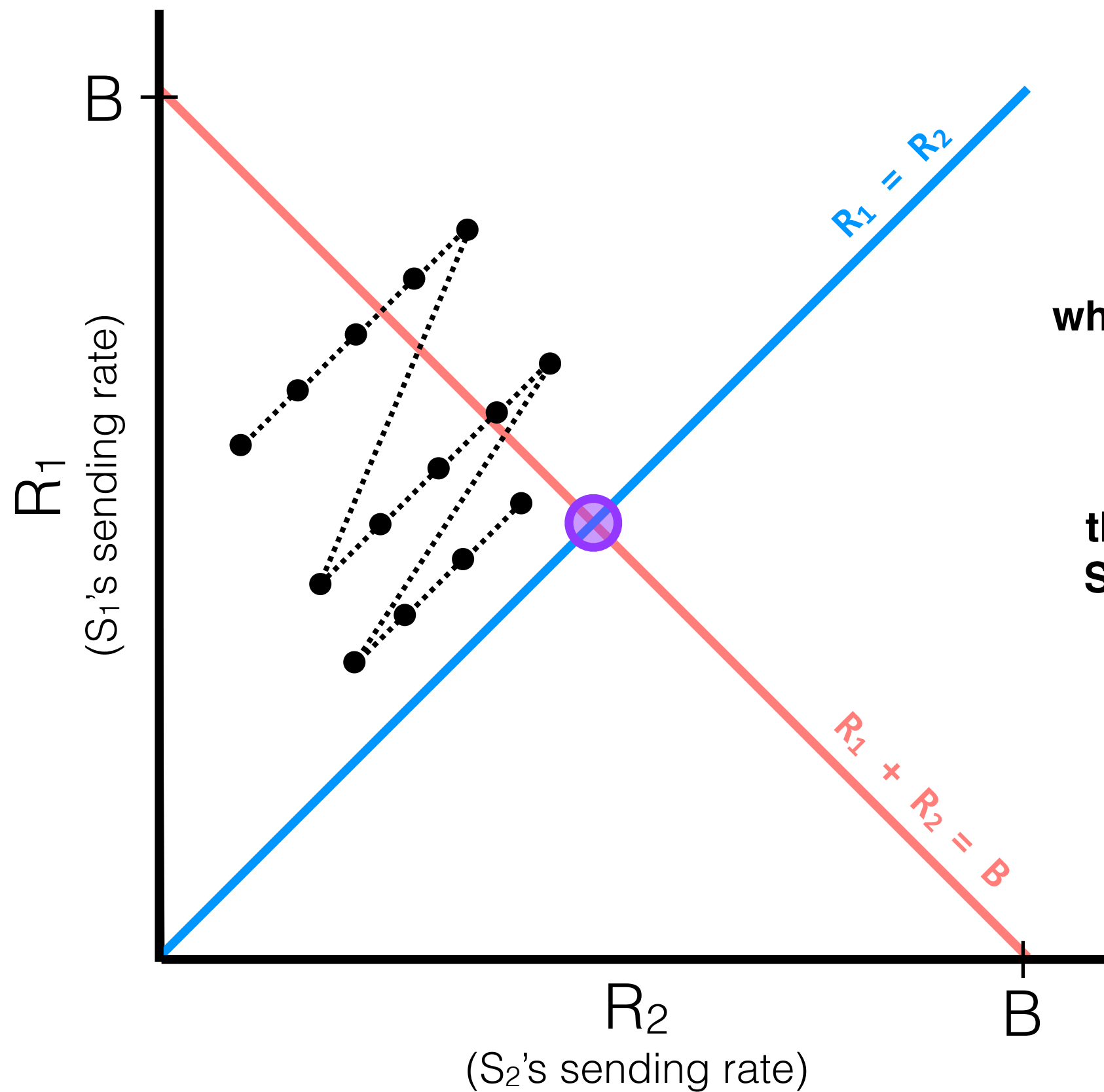


efficiency
(utilization)

the network is fully utilized
when the bottleneck link is “full”

fairness

the network is fair when S_1 and
 S_2 are sending at the same rate

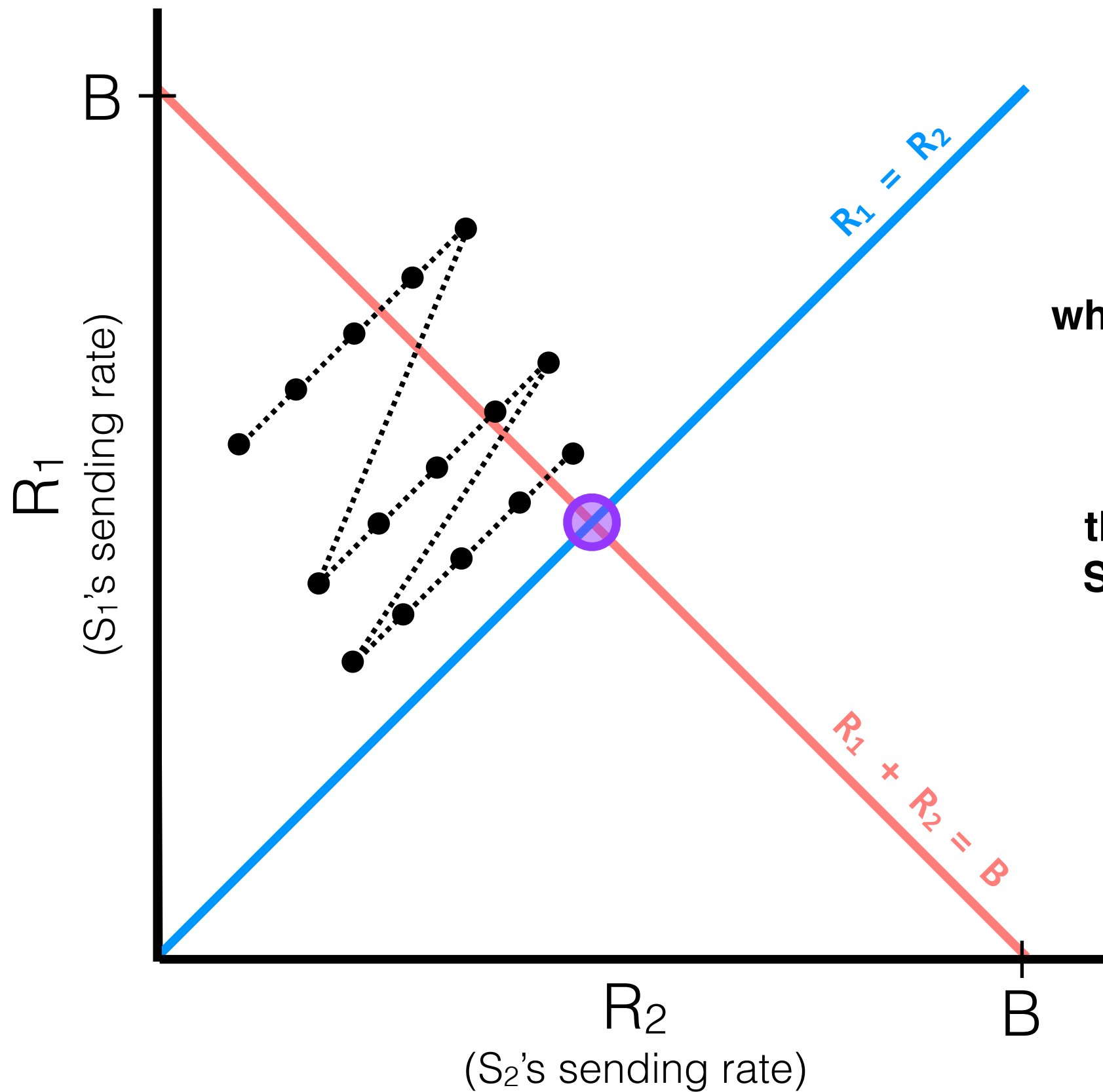


efficiency
(utilization)

the network is fully utilized
when the bottleneck link is "full"

fairness

the network is fair when S_1 and
 S_2 are sending at the same rate

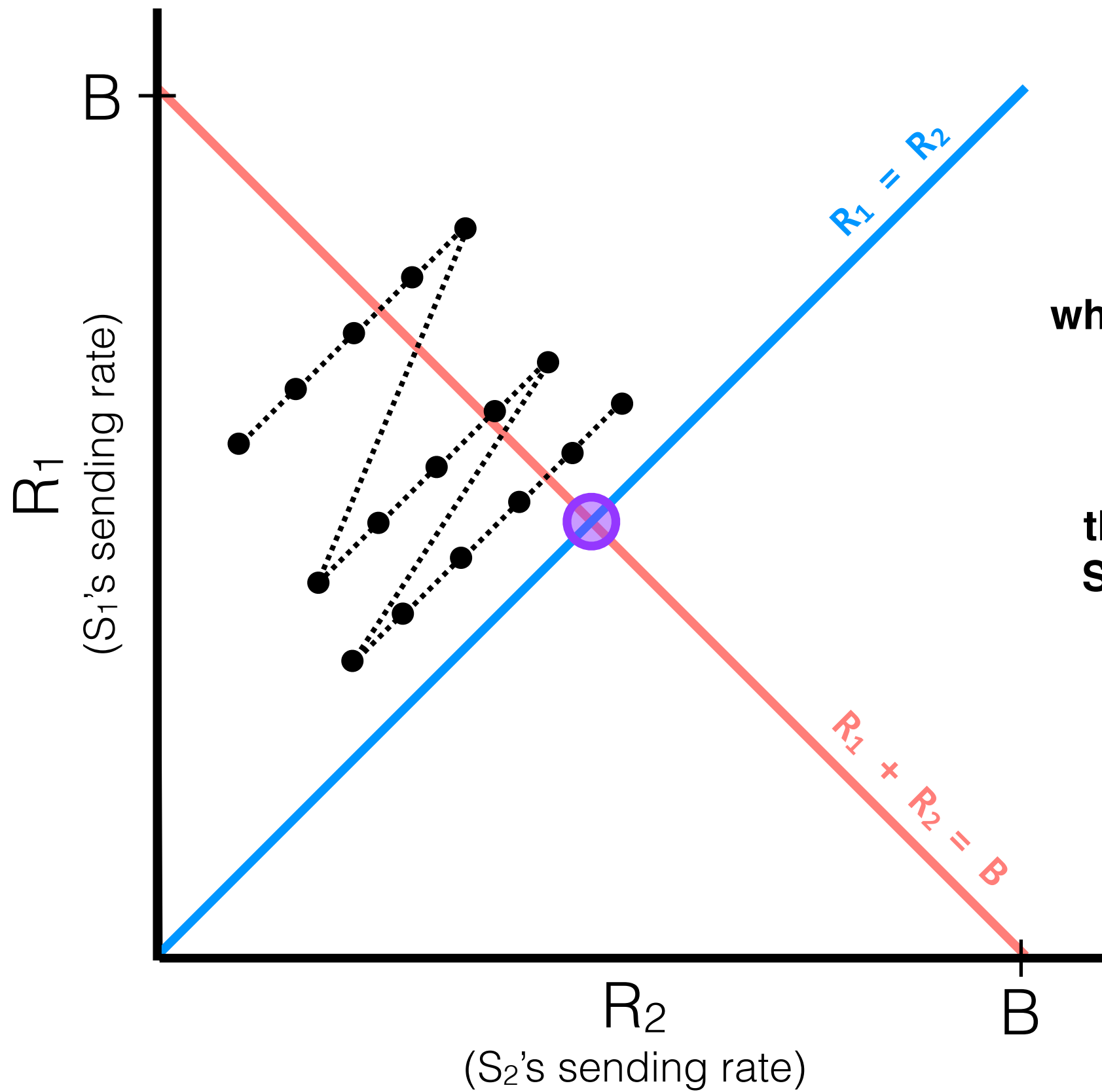


efficiency
(utilization)

the network is fully utilized
when the bottleneck link is "full"

fairness

the network is fair when S_1 and
 S_2 are sending at the same rate

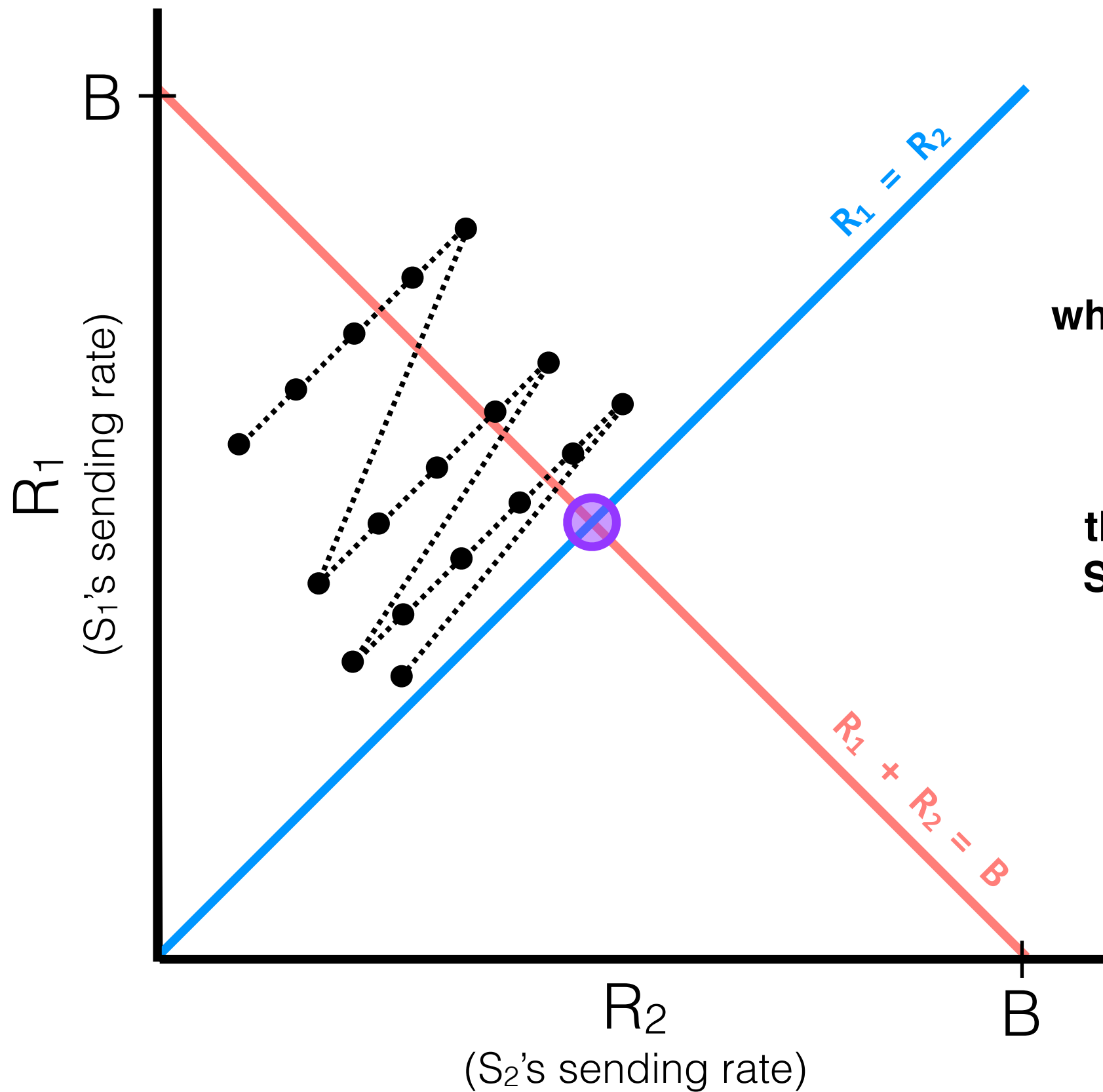


efficiency
(utilization)

the network is fully utilized
when the bottleneck link is “full”

fairness

the network is fair when S_1 and
 S_2 are sending at the same rate

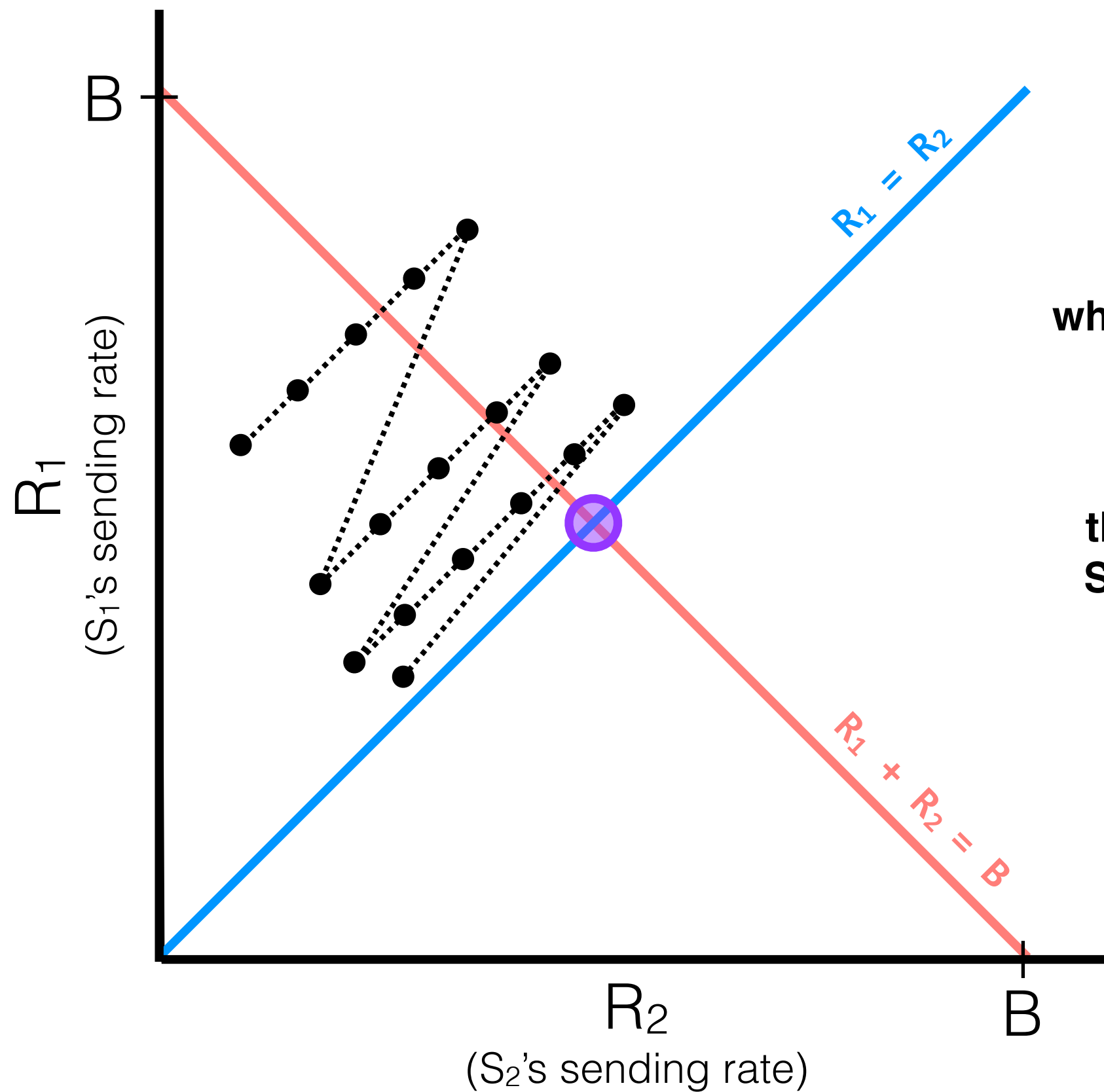


efficiency
(utilization)

the network is fully utilized
when the bottleneck link is “full”

fairness

the network is fair when S_1 and
 S_2 are sending at the same rate



efficiency
(utilization)

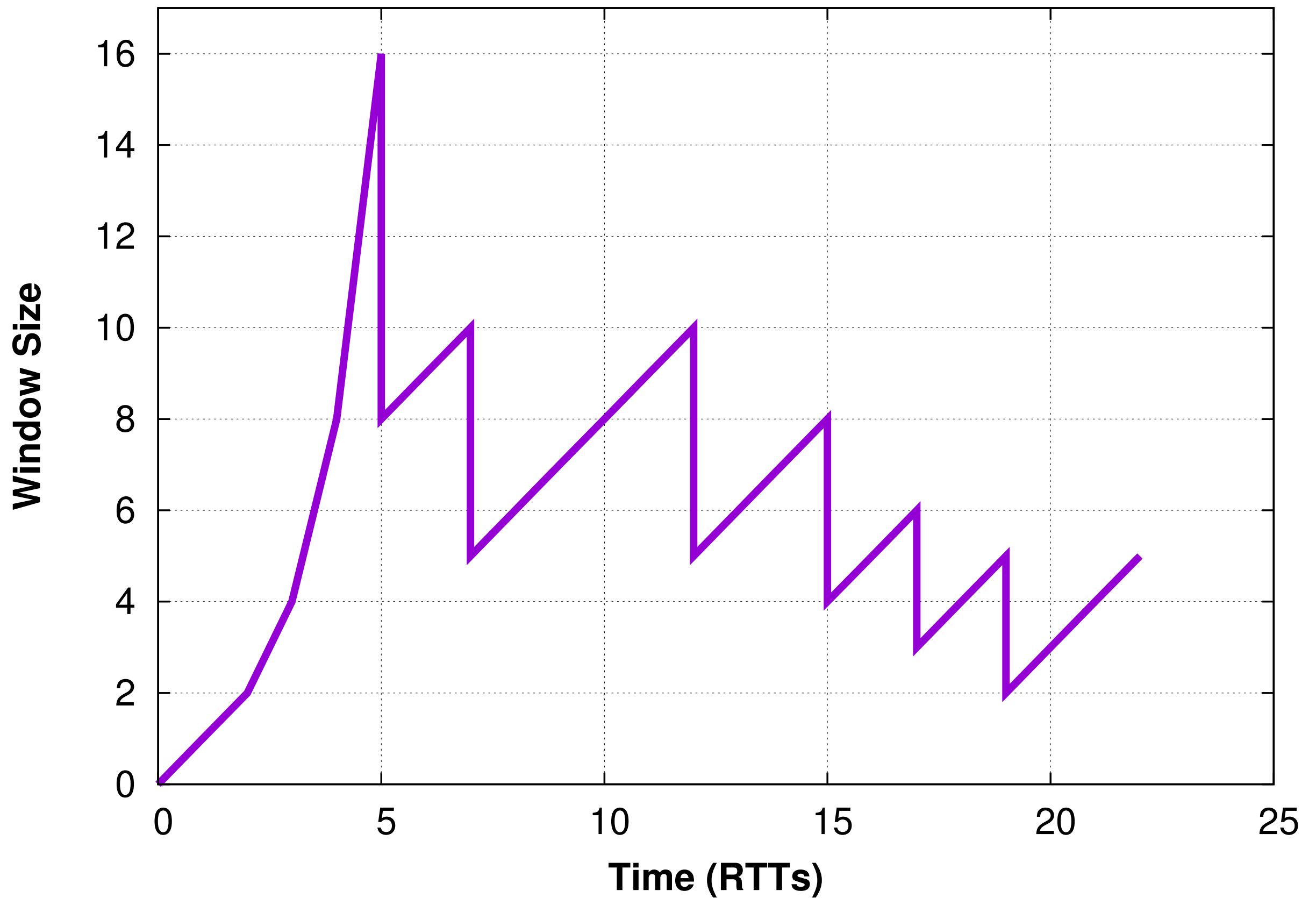
the network is fully utilized
when the bottleneck link is "full"

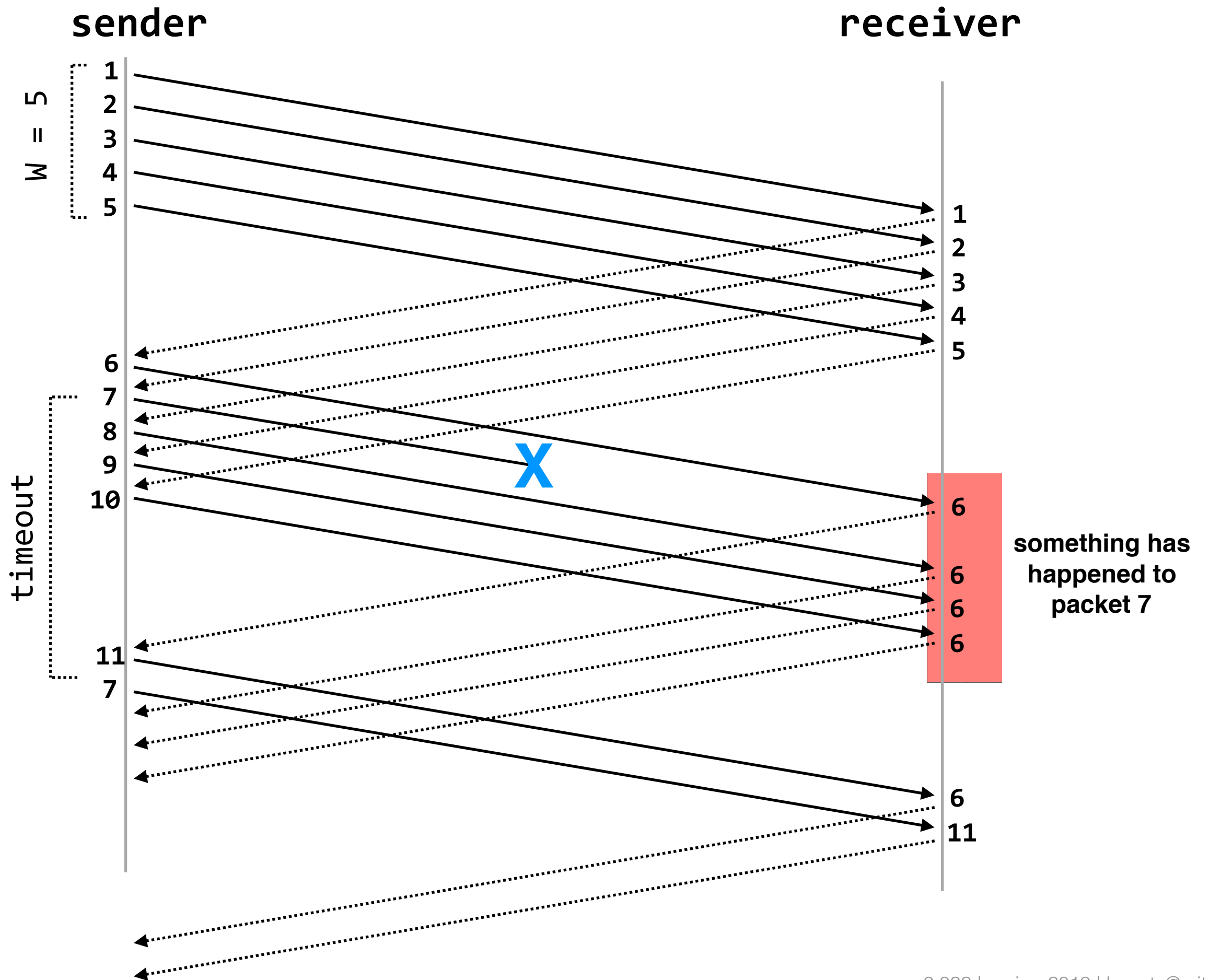
fairness

the network is fair when S_1 and
 S_2 are sending at the same rate

eventually, R_1 and R_2 will come to oscillate around the **fixed point**

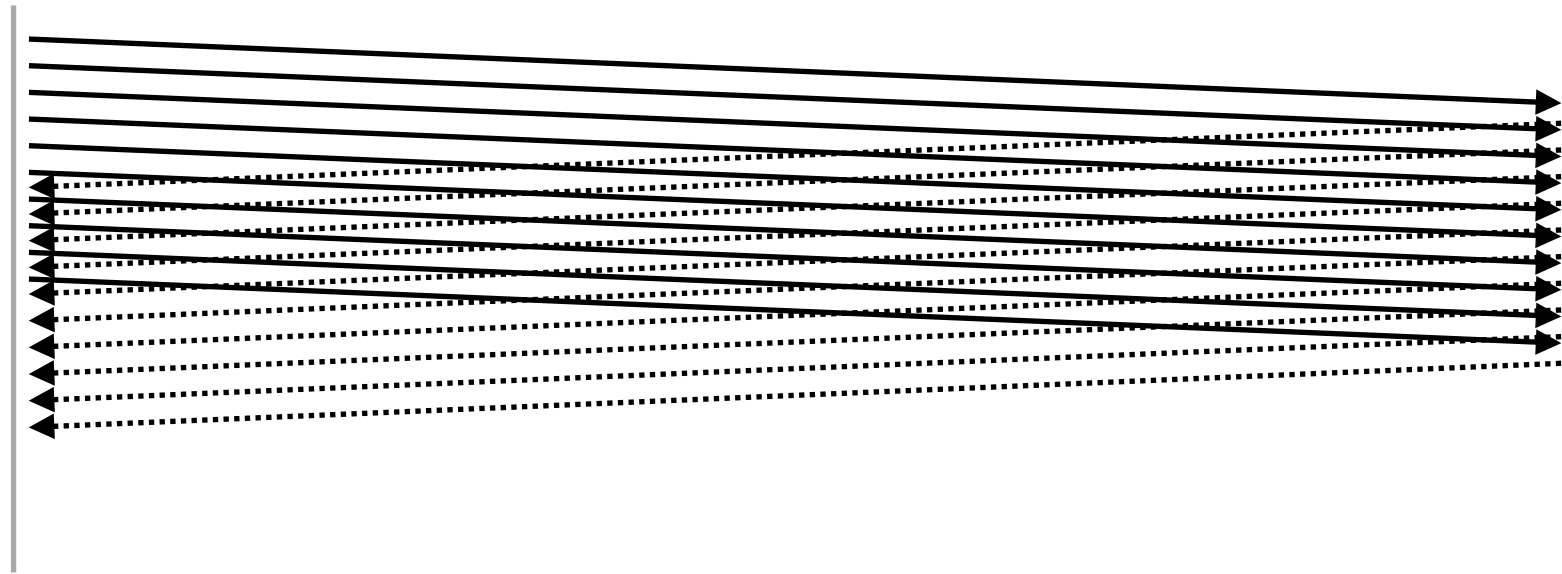
AIMD + Slow Start





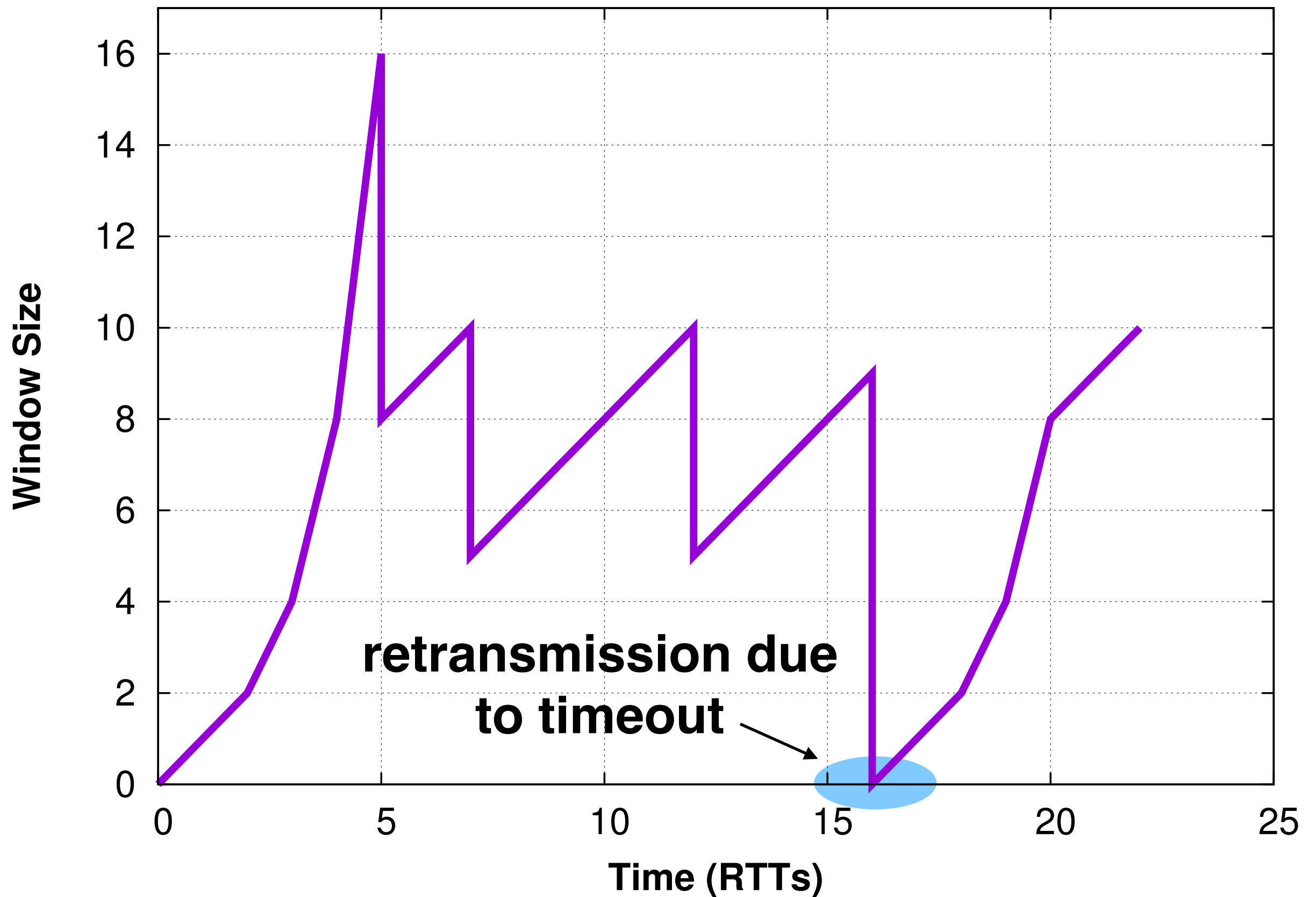
sender

receiver



**in practice, if a single packet is lost, the three “dup”
ACKs will be received before the RTO for that packet
expires**

AIMD + Slow Start



- **TCP** provides **reliable transport** along with **congestion control**: senders increase their window additively until they experience loss, and then back off multiplicatively. Senders also use slow-start and fast-retransmit/fast-recovery to quickly increase the window and recover from loss.
- TCP has been a massive success, but **senders don't react to congestion until queues are already full**. Is there a better way?