WPI

Cloud Control with Distributed Rate Limiting

Raghaven et all

Presented by: Brian Card

CS 577 - Fall 2014 - Kinicki

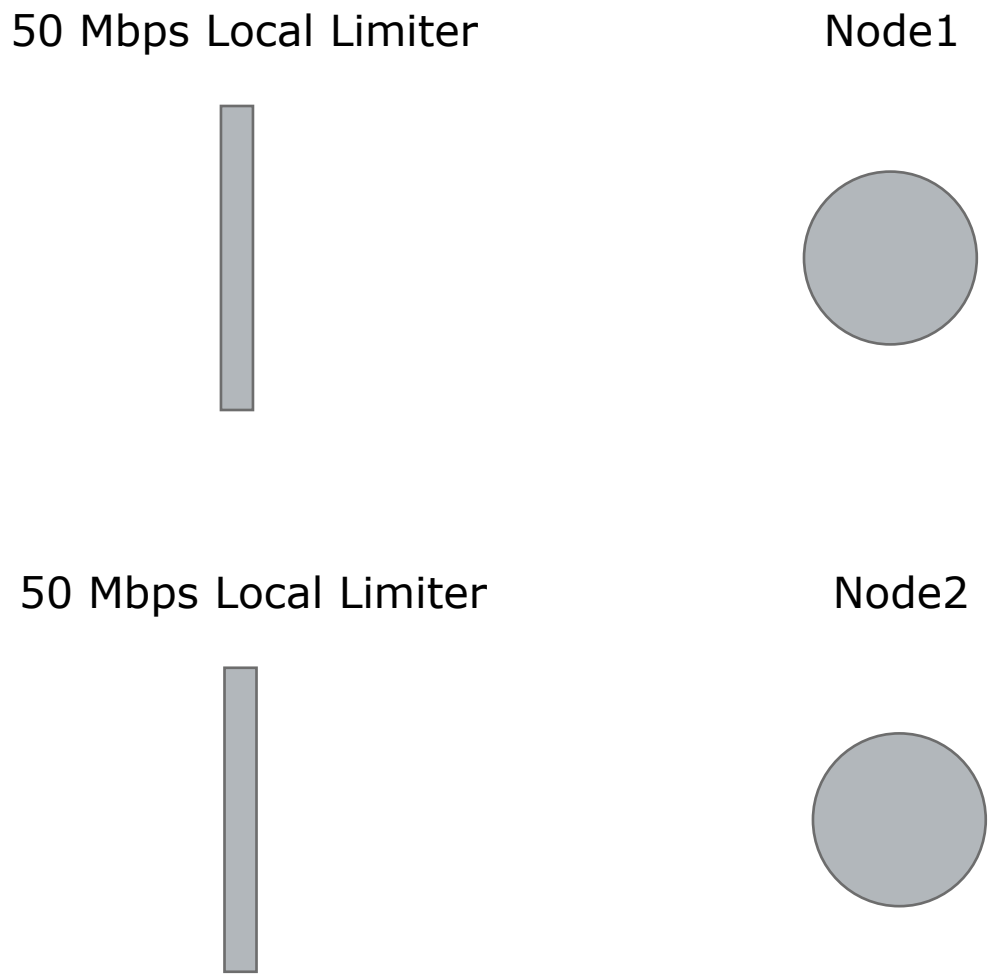
Outline

- Motivation
- Distributed Rate Limiting
  - Global Token Bucket
  - Global Random Drop
  - Flow Proportional Share
- Simulation and Results
- Conclusions

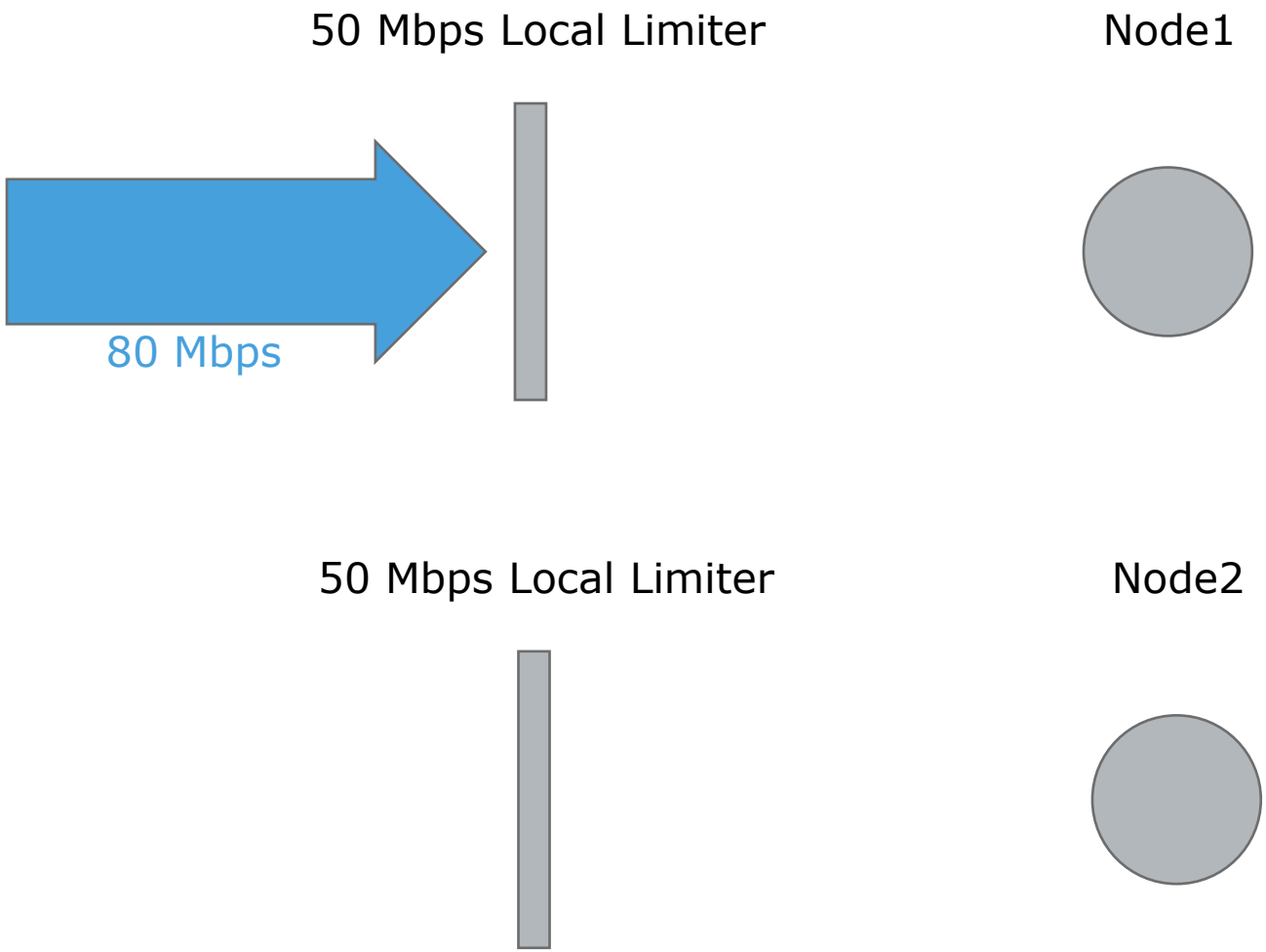
Motivation

- Distributed Cloud Based Services are becoming more prevalent
- PaaS vendors want to charge for cloud services
- In a traffic base pricing model, how do you meter traffic in a distributed system?

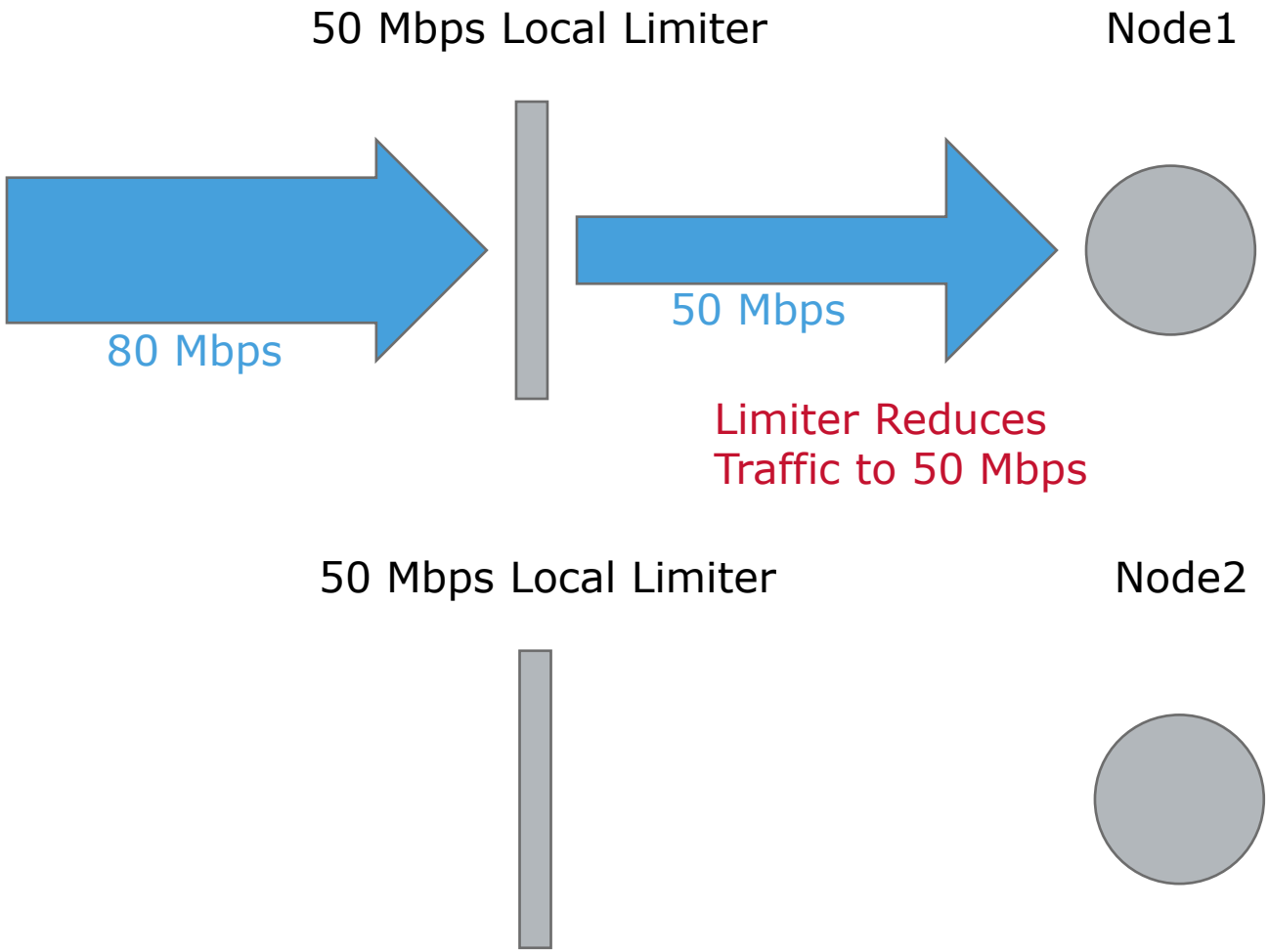
Bad Example, 100 Mps for 2 Nodes



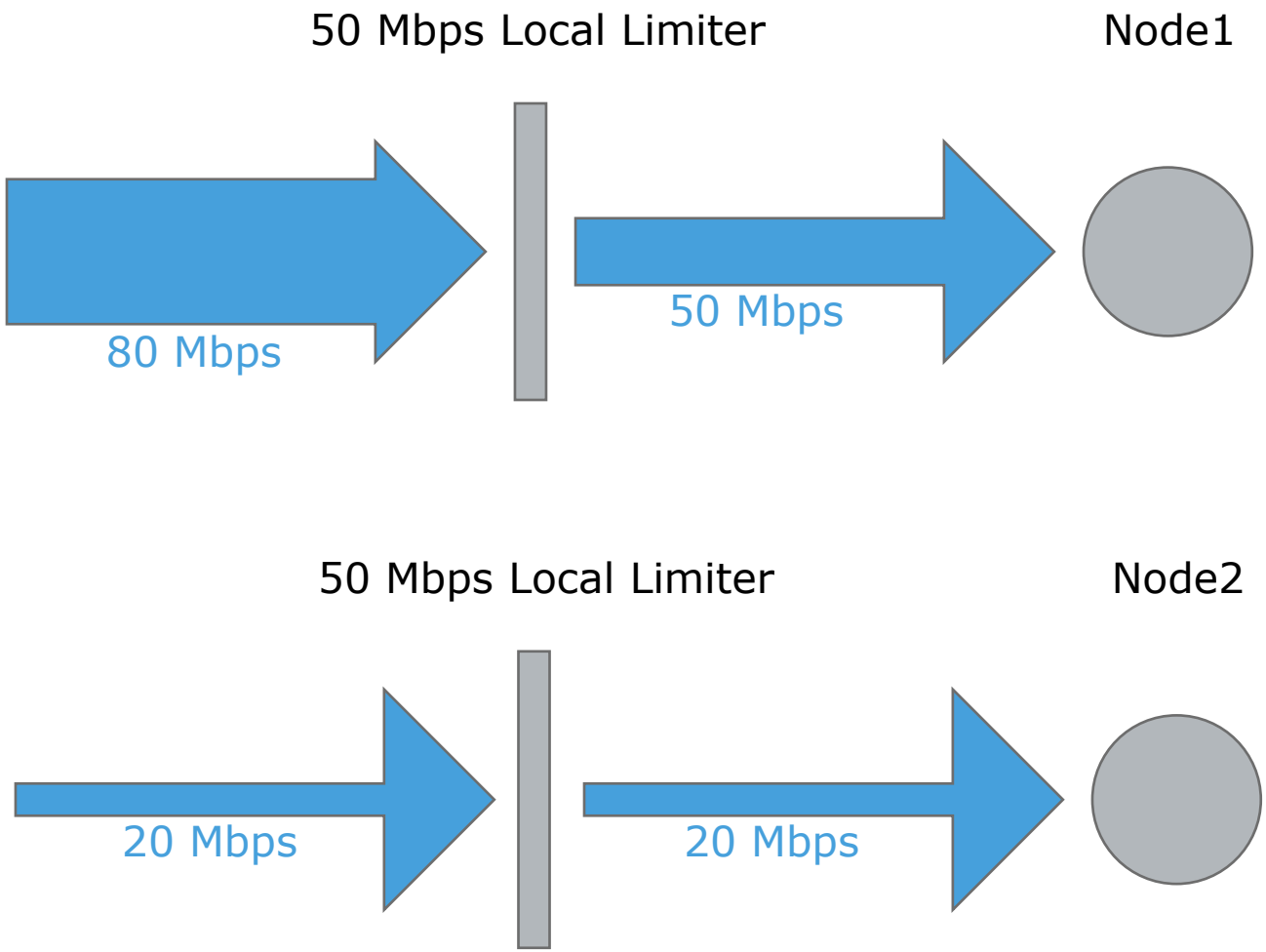
Bad Example, 100 Mps for 2 Nodes



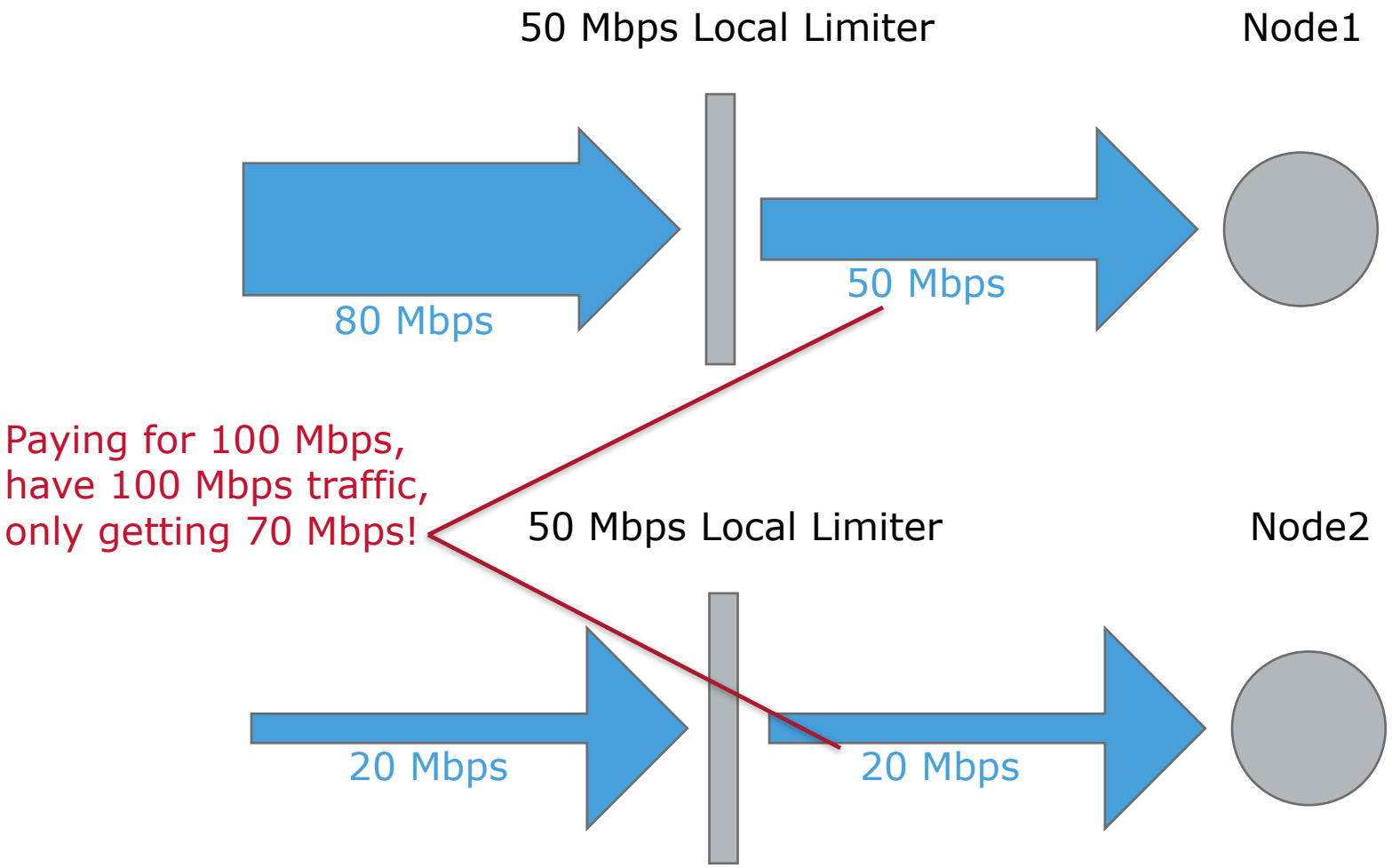
Bad Example, 100 Mps for 2 Nodes



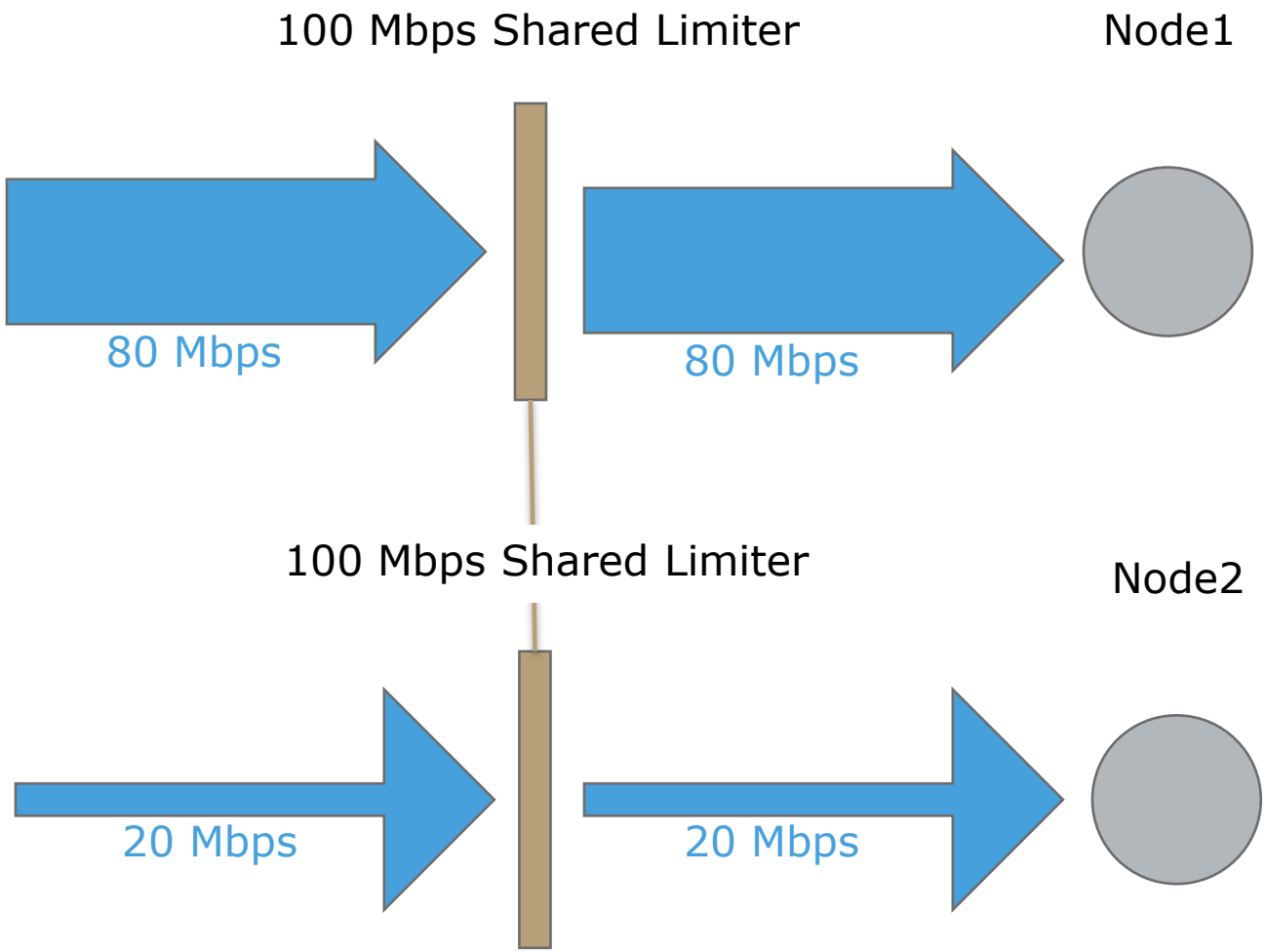
Bad Example, 100 Mps for 2 Nodes



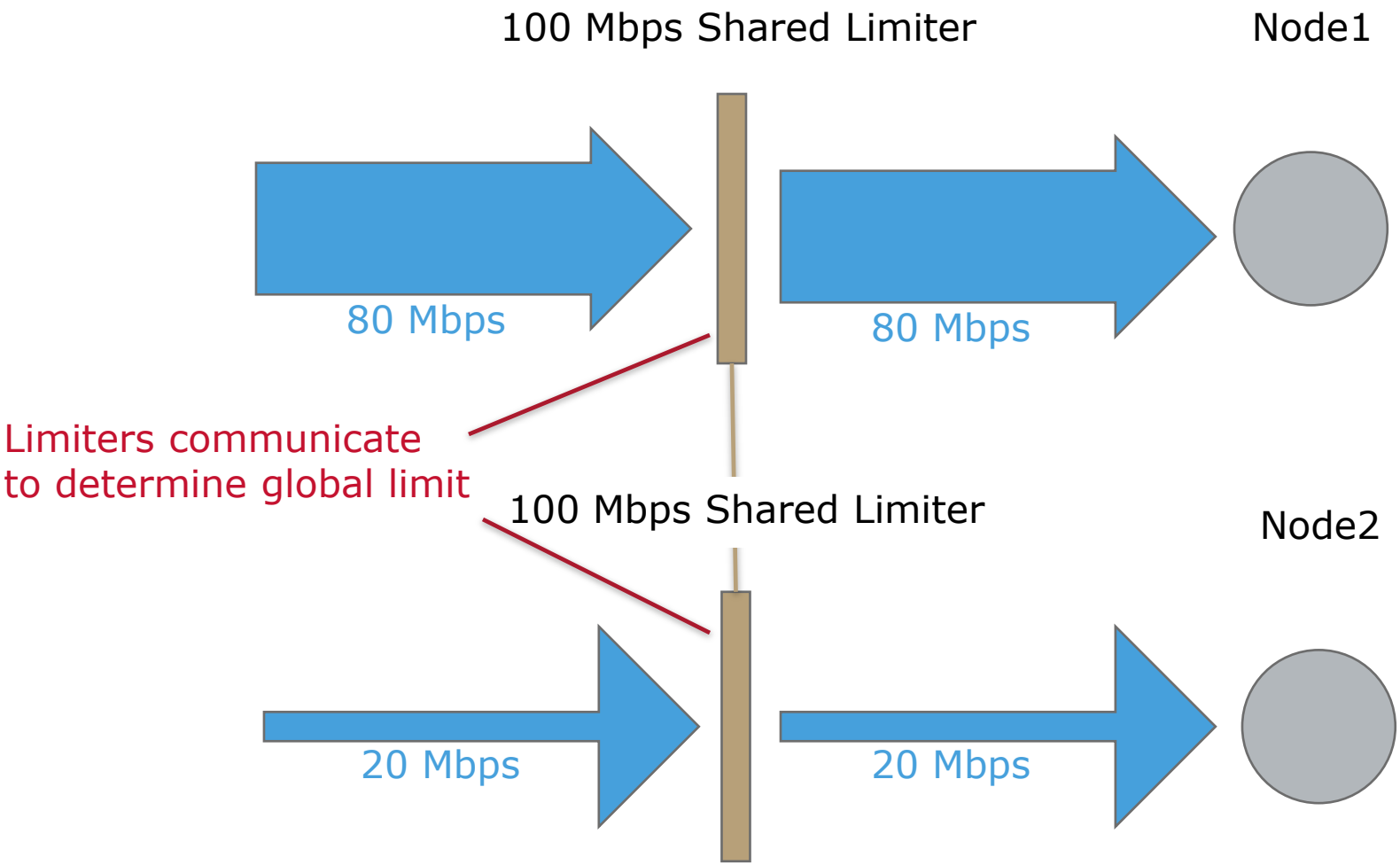
Bad Example, 100 Mps for 2 Nodes



A Better Approach: Distributed Rate Limiter



A Better Approach: Distributed Rate Limiter



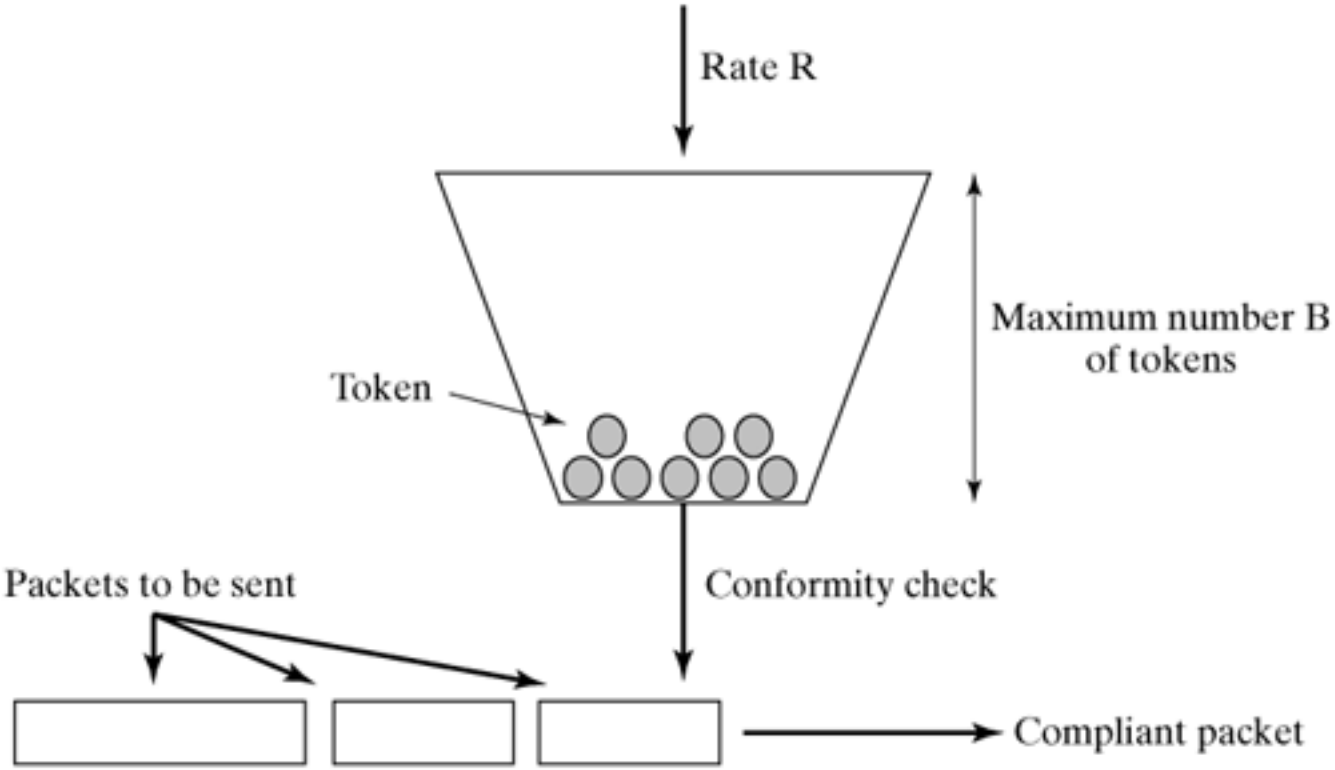
Design of Distributed Rate Limiting

- When global limit is exceeded, packets are dropped
- Limiters estimate incoming traffic and communicate results to other limiters
- Communication between limiters is performed using the Gossip Protocol over UDP

Token Bucket

- Token Buckets are a well known mechanism used to rate limit in networking applications
- Tokens are generated at a rate R
- Packets are traded for a token
- Can handle bursts up to the number of tokens in the bucket
- Bursts drain bucket and subsequent traffic is limited until new tokens are generated

Token Bucket (cont.)



Wehrle, *Linux Networking Architecture*. Prentice Hall, 2004
http://flylib.com/books/en/3.475.1.95/1/

Use of Token Bucket

- Authors compare results to Centralized Token Bucket where a single bucket is used to distribute all of the traffic
  - Single bucket where all limiters must pull tokens from
- This scheme is not practical but serves as the baseline for comparing the results

Distributed Rate Limiting Algorithms

- Global Token Bucket
- Global Random Drop
- Flow Proportional Share

Global Token Bucket (GTB)

- Simulate a global bucket
  - Tokens are shared between limiters
  - When a byte arrives it's traded for a token in the global bucket
  - Each limiter maintains an estimate of the global bucket
  - Limiters broadcast their arrivals to the other limiters which reduces global count
  - 'Estimate Interval' defines how frequently updates are sent
- ✖ Miscounting tokens from stale observations impacts effectiveness

Global Random Drop (GRD)

- RED-like probabilistic dropping scheme
- Instead of counting tokens, estimate a global drop probability
- Apply drops locally based on percentage of traffic received
- Aggregate drop rate should be near the global drop rate

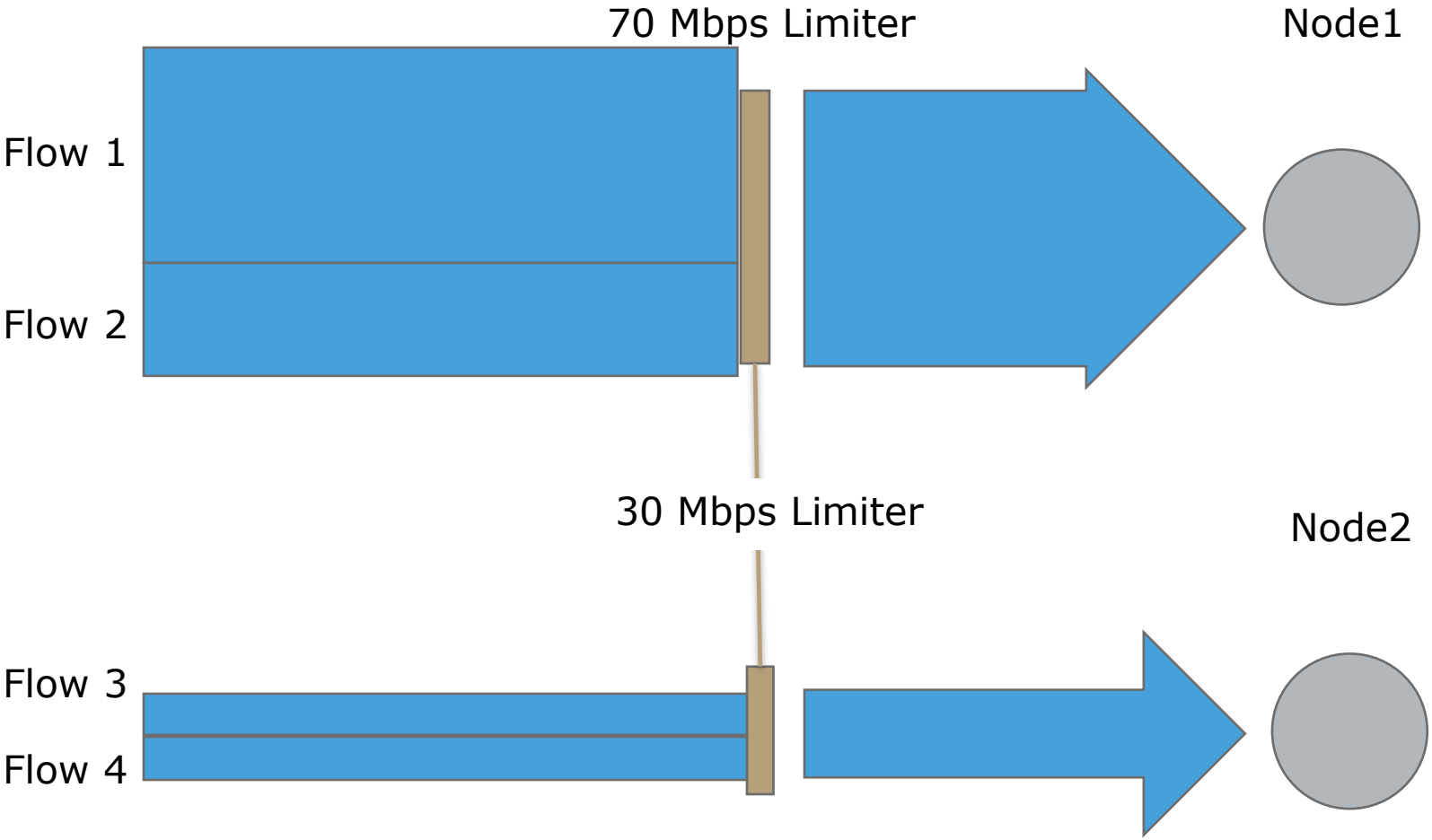
Flow Proportional Share (FPS)

- Optimized for TCP flows (assumes TCP congestion control)
- Tries to ensure fairness between flows
- Each limiter has a local bucket, no global bucket
- Token generation rate is proportional to the number of flows at that limiter

Flow Proportional Share

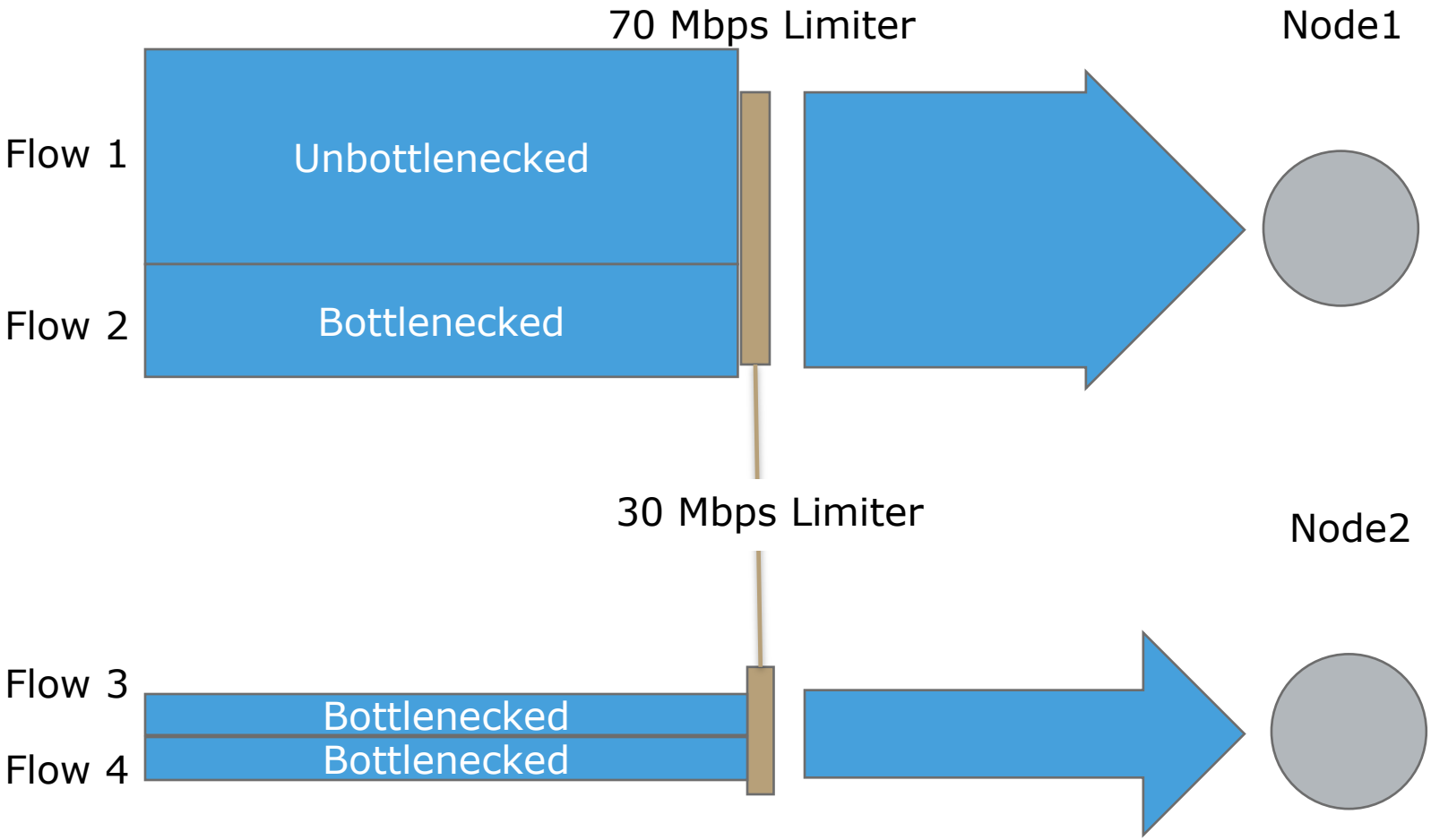
- Flows are classified as either bottlenecked or unbottlenecked
  - *bottlenecked* flows use less than the local rate limit
  - *unbottlenecked* flows use more than the local rate limit (or equal)
  - Flows are unbottlenecked if the limiter is preventing them from passing more traffic
- Idea here is to give weight to the unbottlenecked flows because they are the ones fighting for traffic

FPS – Bottlenecks



* Not to scale

FPS – Bottlenecks



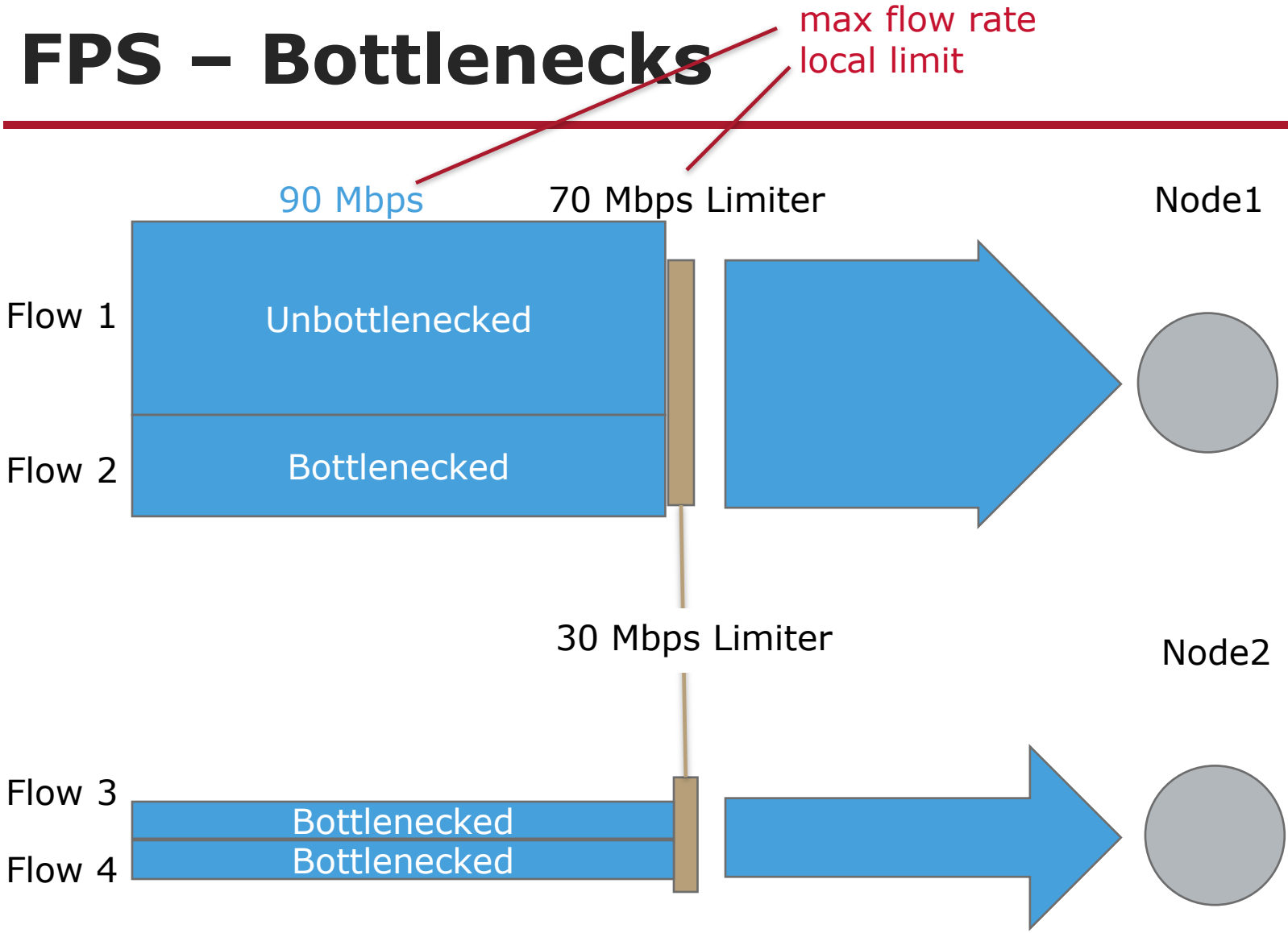
* Not to scale

FPS Weight Calculation
Local Arrival Rate ≥ Local Limit

- Make a fixed size set of all unbottlenecked flows
  - Not all flows to avoid scaling issues with per flow state
- Pick the largest flow, then divide that by the local rate to find the weight

- Ideal weight = local limit / max flow rate
- local limit = (ideal weight * limit) / (remote weights + ideal weight)

FPS – Bottlenecks



* Not to scale

FPS Weight Calculation
Local Arrival Rate < Local Limit

- Calculate the local flow rate
- Ideal weight is calculated proportional to the other flow weights:
  - ideal = (local flow rate * sum of all remote weights not including this rate) / (local limit – local demand)
- Idea is to reduce the local limit to match the arrival rate

Pseudo-code

```
FPS-ESTIMATE()
1    for each flow f in sample set
2        ESTIMATE(f)
3    localdemand ← r_i
4    if localdemand ≥ locallimit then
5        maxflowrate ← MAXRATE(sample set)
6        idealweight ← locallimit / maxflowrate
7    else
8        remoteweights ← Σ_{j≠i}^{n} w_j
9        idealweight ← (localdemand · remoteweights) / (limit − localdemand)
10   locallimit ← (idealweight · limit) / (remoteweights + idealweight)
11   PROPAGATE(idealweight)

FPS-HANDLE-PACKET(P: Packet)
1    if RAND() < resampleprob then
2        add FLOW(P) to sample set
3    TOKEN-BUCKET-LIMIT(P)
```

Figure 2: Pseudocode for FPS. w_i corresponds to the weight at each limiter i that represents the normalized flow count (as opposed to rates r_i as in GRD).

Use of EWMA

- Estimated Weighted Moving Averages (EWMA) are used to smooth out estimated arrival rates
- Also used in Flow Proportional Share to reduce oscillations between two states

Evaluation

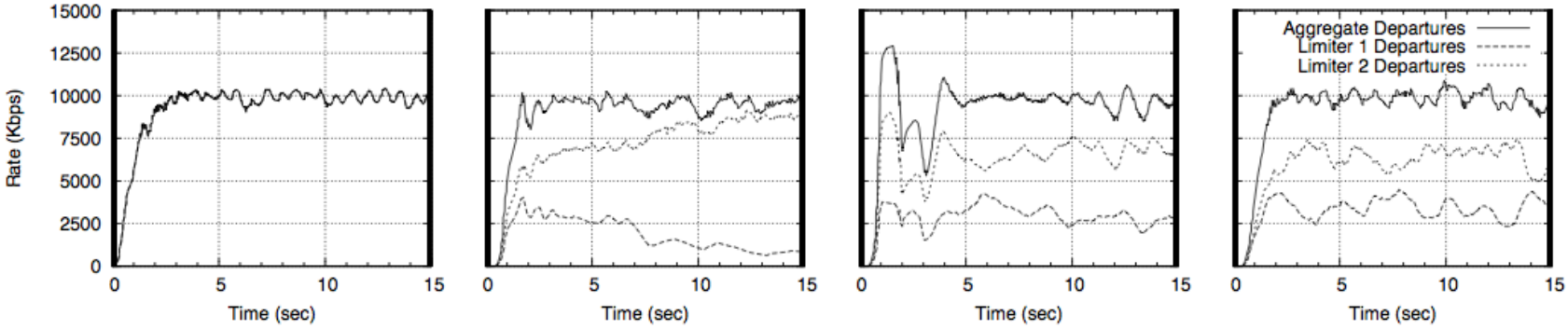
- Comparison to Centralized Token Bucket
- Fairness to Centralized Token Bucket
- Simulations with Departing and Arriving Flows
- Simulations with Mixed Length Flows
- Fairness of Long vs Short Flows
- Fairness of Bottlenecked Flows in FPS
- Fairness With Respect to RTT
- PlanetLab 1/N vs FPS experiments

Setup

- Limiters run on Linux
- ModelNet is used as the network simulator
- Kernel version 2.6.9
- TCP NewReno with SACK

Comparison to Centralized Token Bucket

- 10 Mbps global limit
- 50 ms estimation interval, 20 second run
- 3 TCP flows to limiter 1, 7 TCP flows to limiter 2



(a) Centralized token bucket. (b) Global token bucket. (c) Global random drop. (d) Flow proportional share.

Figure 3: Time series of forwarding rate for a centralized limiter and our three limiting algorithms in the baseline experiment—3 TCP flows traverse limiter 1 and 7 TCP flows traverse limiter 2.

Arrival Rate Patterns

- Shows how susceptible the algorithm is to bursting
- GTB and GRD are less like our mark (CTB)

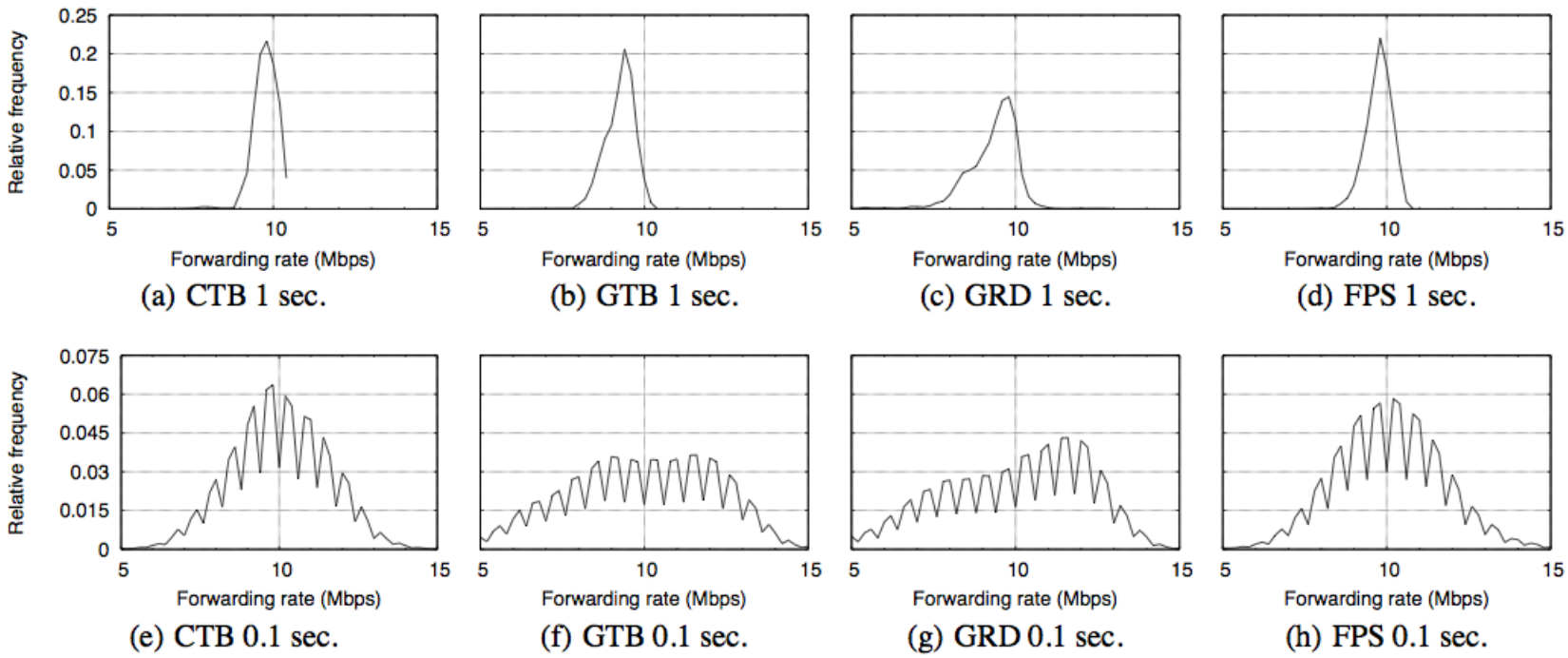


Figure 4: Delivered forwarding rate for the aggregate at different time scales—each row represents one run of the baseline experiment across two limiters with the "instantaneous" forwarding rate computed over the stated time period.

Fairness Compared to CTB

- Above the diagonal is more fair than Central Token Bucket, below the line is less fair
- GRD and FPS are more fair than CTB in most cases

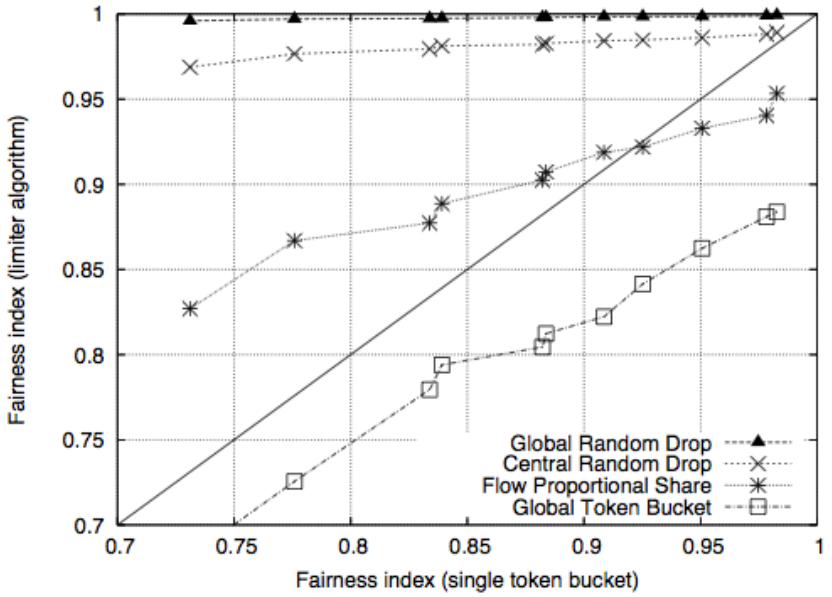
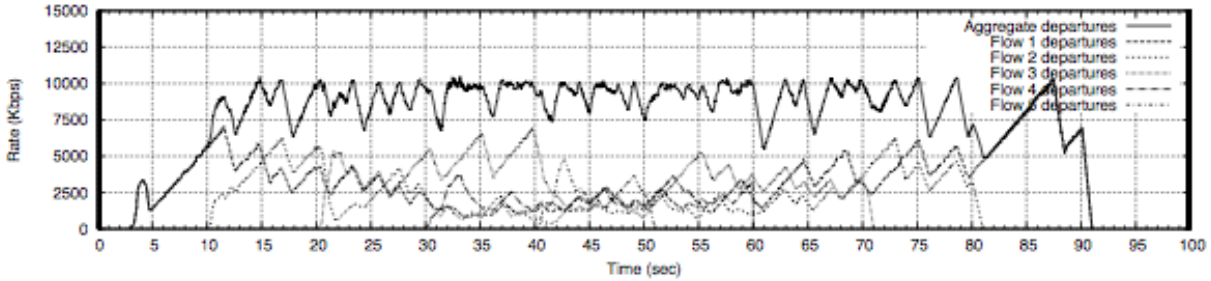


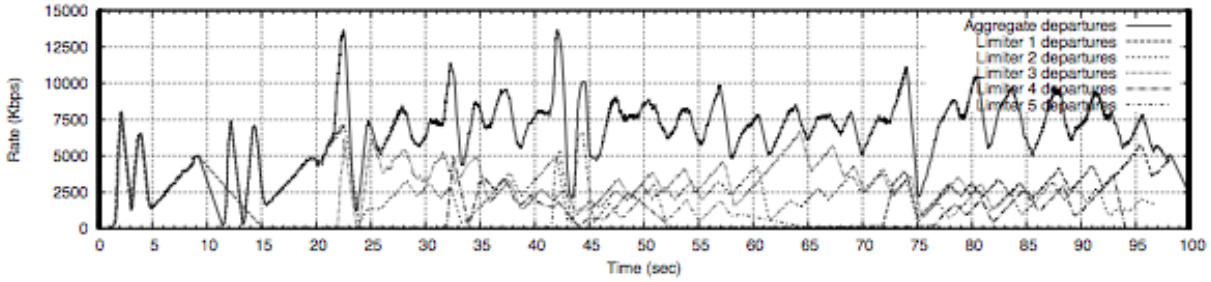
Figure 5: Quantile-quantile plots of a single token bucket vs. distributed limiter implementations. For each point (x, y), x represents a quantile value for fairness with a single token bucket and y represents the same quantile value for fairness for the limiter algorithm.

Departing and Arriving Flows

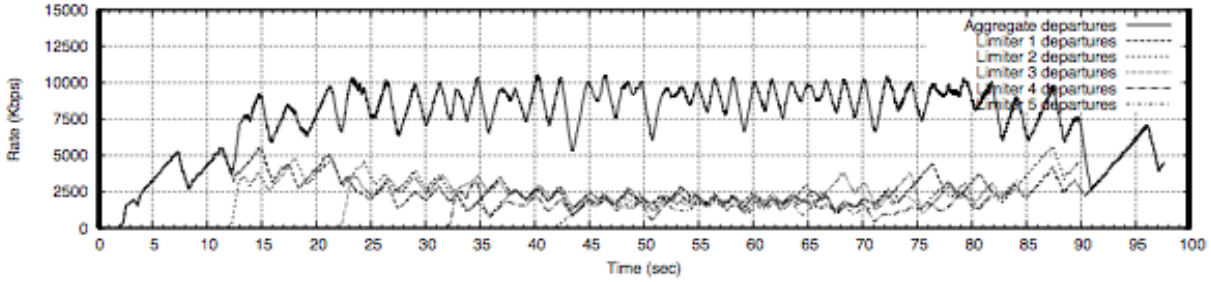
- Every 10 seconds, add a new flow up to 5 flows
- After 50 seconds, start removing flows
- Notice the reference algorithm CTB is not very fair



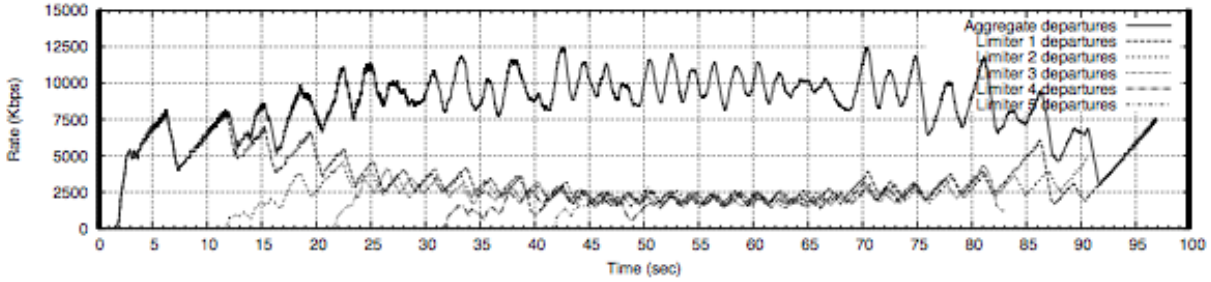
(a) Central token bucket



(b) Global random drop with 500-ms estimate interval

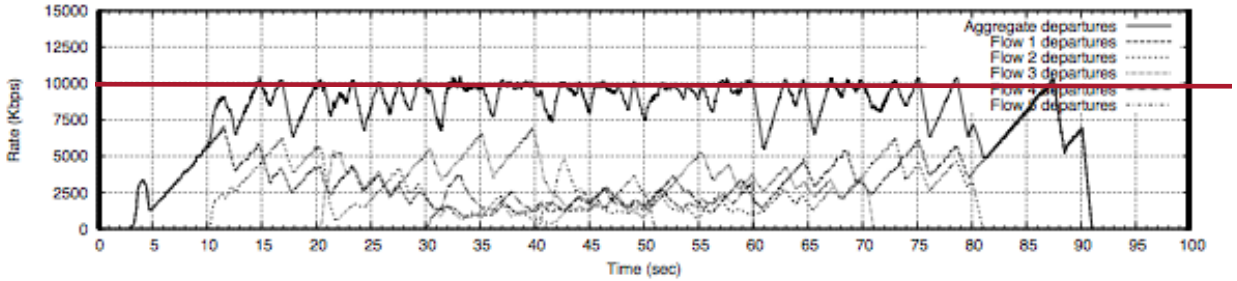


(c) Global random drop with 50-ms estimate interval

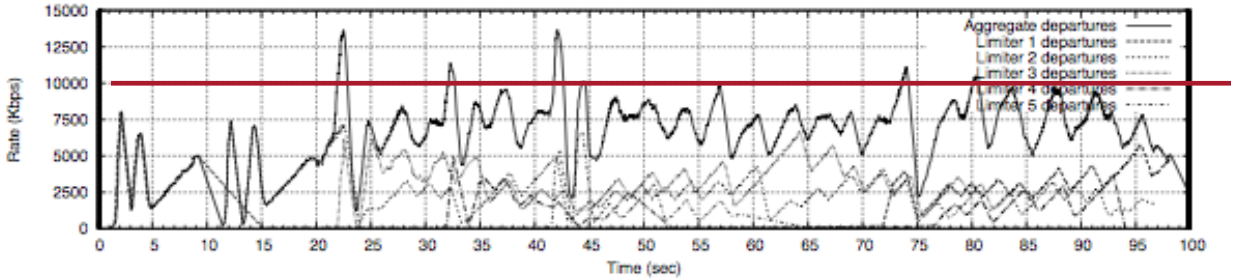


(d) Flow proportional share 500-ms estimate interval

Figure 6: Time series of forwarding rate for a flow join experiment. Every 10 seconds, a flow joins at an unused limiter.

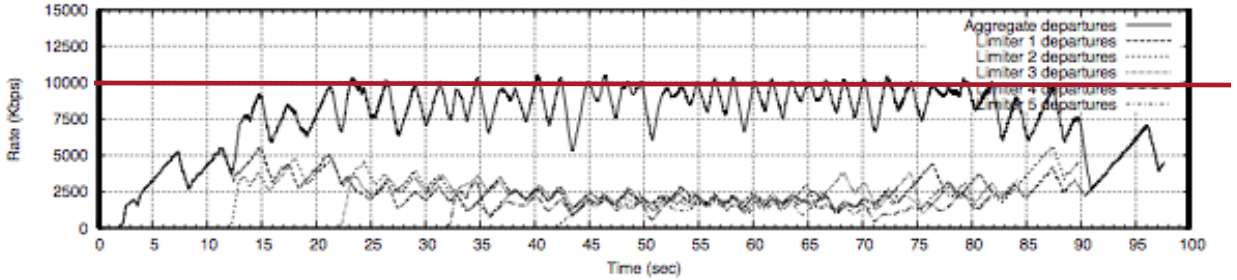


(a) Central token bucket

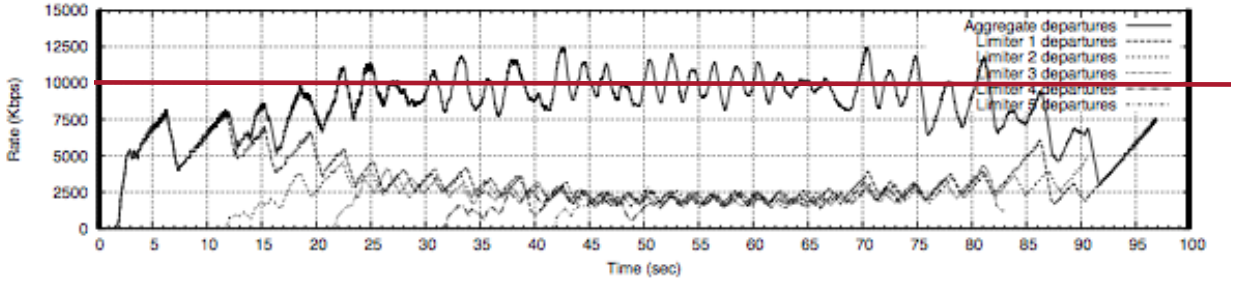


(b) Global random drop with 500-ms estimate interval

GRD is over the global limit here



(c) Global random drop with 50-ms estimate interval



(d) Flow proportional share 500-ms estimate interval

FPS is over the global limit here

Figure 6: Time series of forwarding rate for a flow join experiment. Every 10 seconds, a flow joins at an unused limiter.

Mixed Length Flows

- 10 long lived TCP flows through one limiter
- Short lived flows with Poisson distribution through another
- Measuring fairness between different types of flows
- GRD is the most fair followed by FPS and the CTB

Table 1; Fairness of Long vs Short Flows

	CTB	GRD	FPS
Goodput (bulk mean)	6900.90	7257.87	6989.76
(stddev)	125.45	75.87	219.55
Goodput (web mean)	1796.06	1974.35	2090.25
(stddev)	104.32	93.90	57.98
Web rate (h-mean) [0,5000)	28.17	25.84	25.71
[5000, 50000)	276.18	342.96	335.80
[50000, 500000)	472.09	612.08	571.40
[500000, ∞)	695.40	751.98	765.26
Fairness (bulk mean)	0.971	0.997	0.962

Table 1: Goodput and delivered rates (Kbps), and fairness for bulk flows over 10 runs of the Web flow experiment. We use mean values for goodput across experiments and use the harmonic mean of rates (Kbps) delivered to Web flows of size (in bytes) within the specified ranges.

Changes in Bottlenecked Flows for FPS

- 2 limiters, 3 flows to limiter 1, 7 flows to limiter 2
- 10 Mbps global limit
- At 15 seconds the 7 flows are restricted to 2 Mbps by a bottleneck
  - Should be 8 Mbps to limiter 1 and 2 Mbps to limiter 2
- At 31 seconds a new flow arrives at limiter 2
  - Should split 8 Mbps between the 4 flows, plus 2 Mbps for other 7 flows, so 4 Mbps at limiter 1 and 6 Mbps at limiter 2

Changes in Bottlenecked Flows for FPS

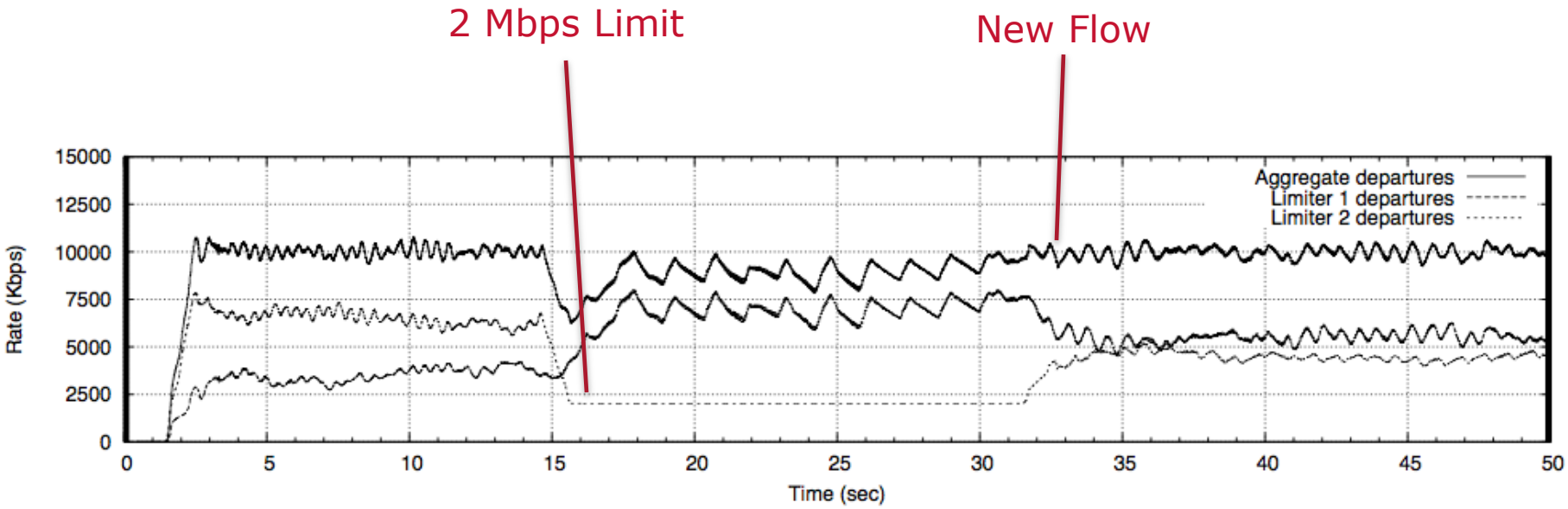


Figure 7: FPS rate limiting correctly adjusting to the arrival of bottlenecked flows.

Changes in Bottlenecked Flows for FPS

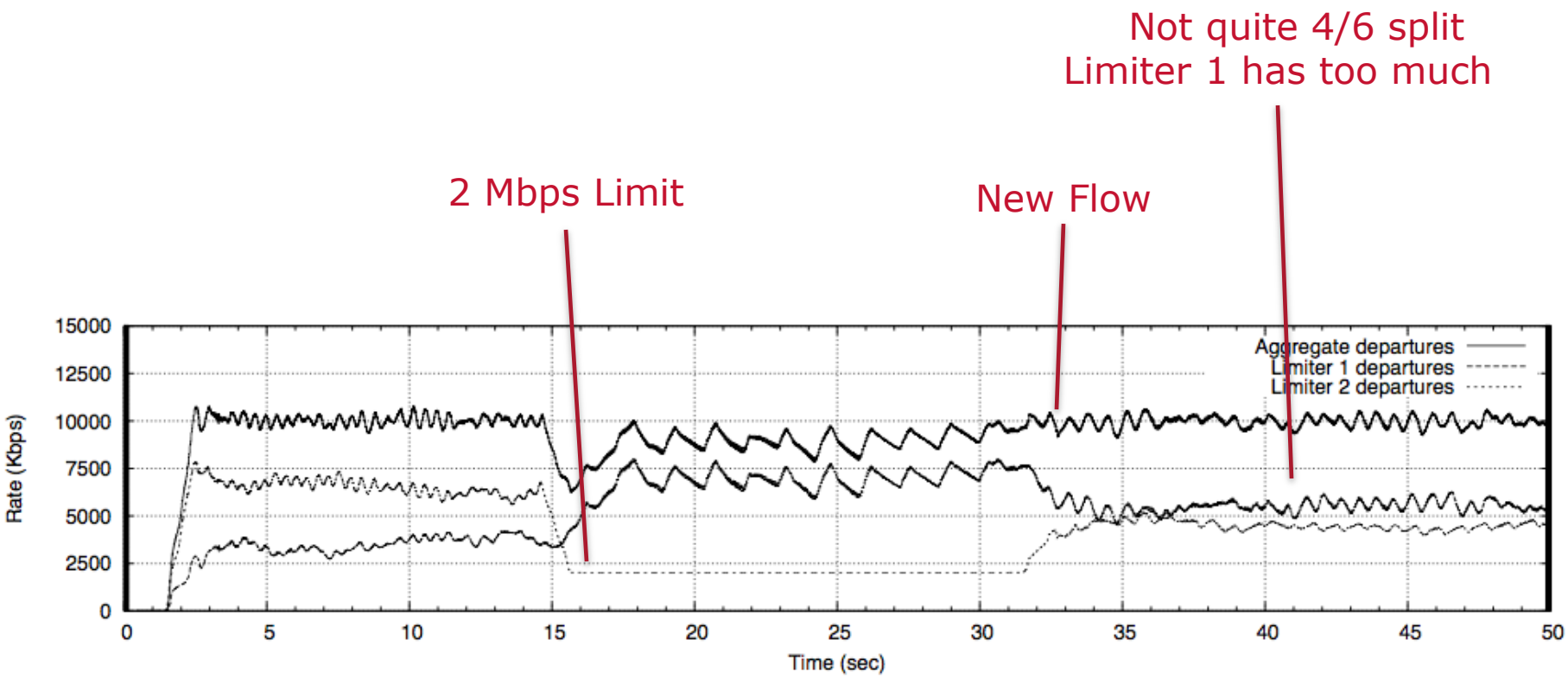


Figure 7: FPS rate limiting correctly adjusting to the arrival of bottlenecked flows.

Fairness With Respect to RTT

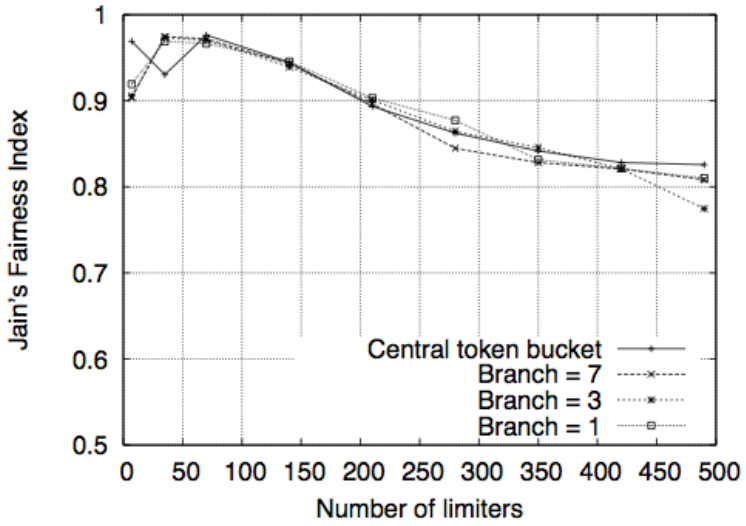
- Same as baseline experiment except changing RTT times of flows
- FPS is most fair

	CTB	GRD	FPS
Aggregate (Mbps)	10.57	10.63	10.43
Short RTT (Mbps)	1.41	1.35	0.92
(stddev)	0.16	0.71	0.15
Long RTT (Mbps)	0.10	0.16	0.57
(stddev)	0.01	0.03	0.05

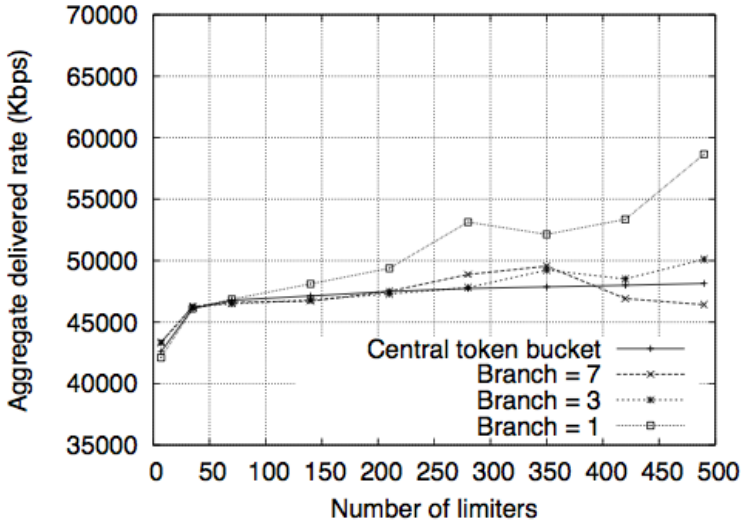
Table 2: Average throughput for 7 short (10-ms RTT) flows and 3 long (100 ms) RTT flows distributed across 2 limiters.

Gossip Branching Factor

- Higher branch factor increases limiter communication
- Notice fairness degradation at large numbers of limiters



(a) FPS fairness.



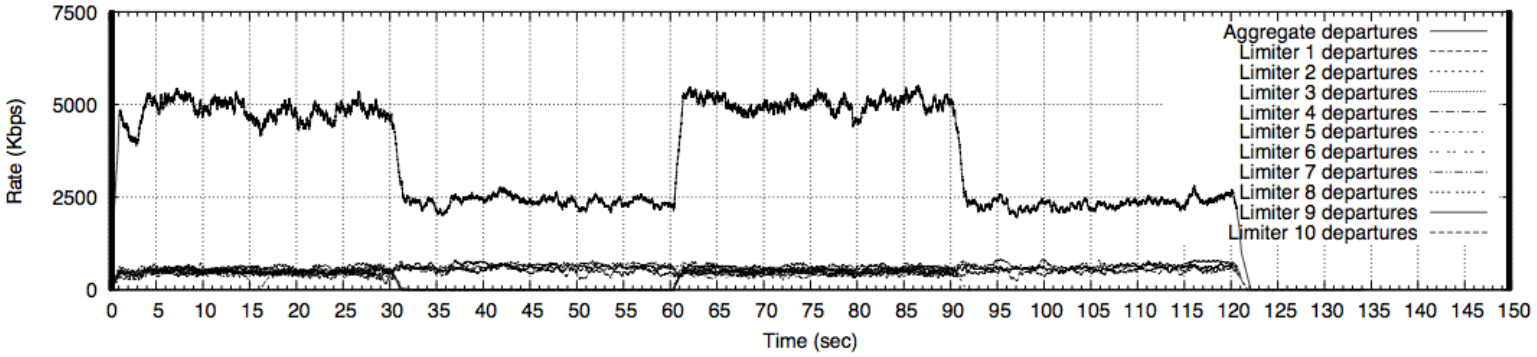
(b) FPS delivered rate.

Figure 8: Fairness and delivered rate vs. number of limiters in the scaling experiment.

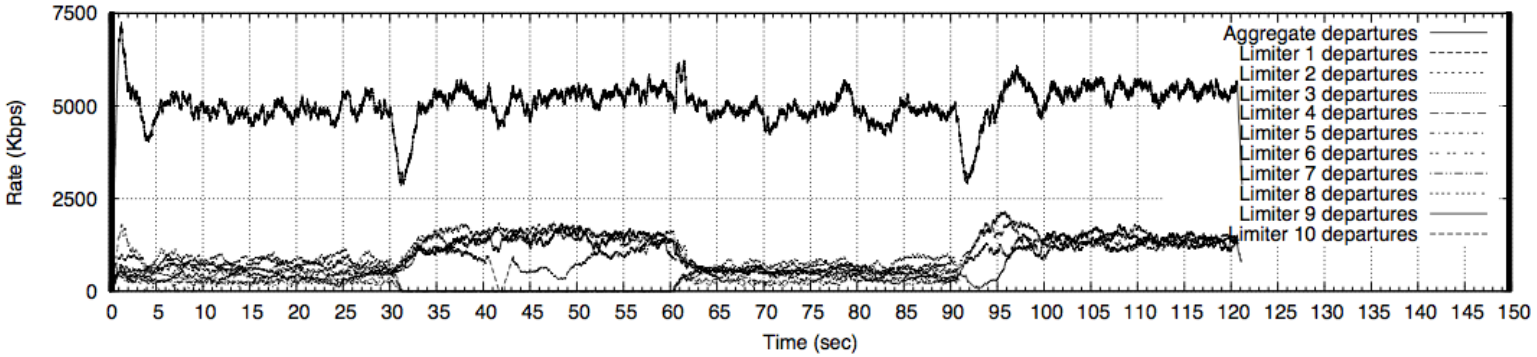
PlanetLab test- 1/N vs FPS

- 10 PlanetLab servers serving web content
- 5 Mbps global limit
- After 30 seconds 7 of the servers cut out
- FPS re-allocates the load to the 3 servers
- After another 30 seconds all servers come back
- FPS re-allocates the load to all 10 servers

PlanetLab test- 1/N vs FPS



(a) $\frac{1}{N}$ limiting.



(b) Flow Proportional Share

Figure 9: A time-series graph rate limiting at 10 PlanetLab sites across North America. Each site is a Web server, fronted by a rate limiter. Every 30 seconds total demand shifts to four servers and then back to all 10 nodes. The top line represents aggregate throughput; other lines represent the served rates at each limiter.

Conclusions

- Several algorithms trying to tackle distributed rate limiting
- FPS performs well for TCP based flows, other techniques suitable for mixed flows
- FPS can perform better than the reference implementation CTB in several scenarios
- Overall interesting approach to DRL with a couple of small quirks