Performance of Networked Systems

Assignment 1

Daniele Di Cesare
VU Amsterdam
email@vu.nl

Gabriel Marica
VU Amsterdam
email@vu.nl

Lecturer:
Prof.dr. Rob van der Mei

November 28, 2025

Abstract

This document is the submission of Daniele Di Cesare and Gabriel Marica for the first assignment of Performance of Networked Systems. Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magnam aliquam quaerat voluptatem. Ut enim aeque doleamus animo, cum corpore dolemus, fieri tamen permagna accessio potest, si aliquod aeternum et infinitum impendere.

Contents

I. Planning of cellular telephone networks with video-conferencing services

1. Problem Modeling

Assumptions:

1. Poisson process of “calls” with rate λ
2. Average call duration is β
3. Number of lines is N
4. Spatial arrival intensity is λ'
5. Cell area is A

Parameters:

1. $A = 1.2\text{Km}^2$
2. $\lambda = \lambda' A = \frac{25}{\text{h}\cdot\text{Km}^2} \cdot 1.2\text{Km}^2 = 30\frac{1}{\text{h}}$
3. $\beta = \frac{1}{12}\text{h}$
4. $N = 4$

We want to find the blocking probability.

2. Blocking Probability

Using Erlang Blocking Formula we know that the probability of having k blocked channels is:

$$\pi_k = \frac{(\lambda\beta)^k/k!}{\sum_{i=0}^{N}(\lambda\beta)^i/i!}$$

Since N = 4 and $\lambda\beta = 2.5$ we get that the blocking probability of a call is:

$$\begin{aligned}\pi_N &= \frac{(\lambda\beta)^N/N!}{\sum_{i=0}^{N}(\lambda\beta)^i/i!} \\ &= \frac{2.5^4/24}{1 + 2.5 + \frac{2.5^2}{2} + \frac{2.5^3}{6} + \frac{2.5^4}{24}} \\ &\approx 0.15\end{aligned}$$

So a call will be blocked 15% of the time.

3. Parameter Modifications

If the call arrival rate triples, while the average call duration becomes three times as small the blocking probability stays the same.

We can immagine each call occuping a slot of random size β, with this change the slot is divided into three equal parts $\frac{\beta}{3}$, and each part is used by a different caller since the rate tripled.

An analogy would be a leaky bucket (more on that later :P). If you triple how often you pour water but each time you pour a third the water the average water level will stays the same. So the probability that the bucket is full stays the same.

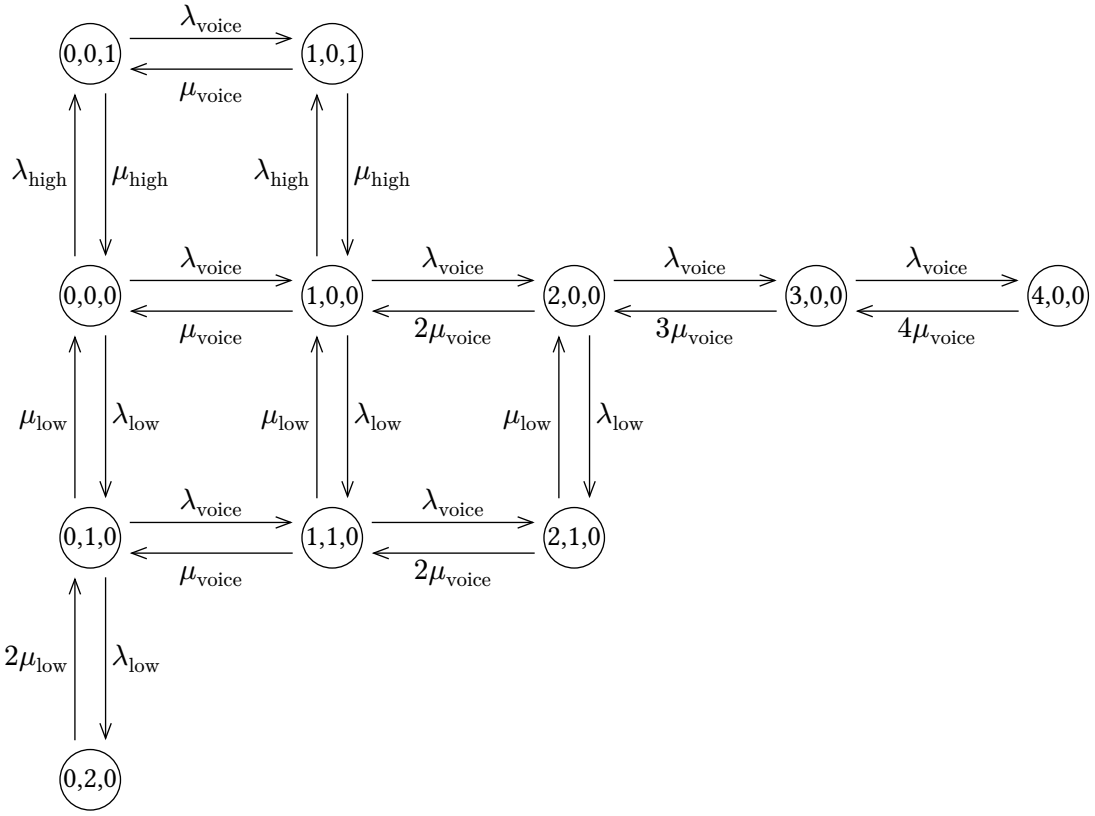
4. Multi-Rate Model Markov Chain

TODO: write assumptions

Parameters:

1. $A = 1.2\text{Km}^2$
2. $P_{\text{low}} = 80\%$

3. $\lambda_{\text{voice}} = \lambda_{\text{voice}}{}'A = \frac{25}{\text{h}\cdot\text{Km}^2} \cdot 1.2\text{Km}^2 = 30\frac{1}{\text{h}}$ (same as before)
4. $\beta_{\text{voice}} = \frac{1}{12}\text{h}$ (same as before)
5. $\lambda_{\text{low}} = \lambda_{\text{video}}{}'AP_{\text{low}} = \frac{0.8}{\text{h}\cdot\text{Km}^2} \cdot 1.2\text{Km}^2 \cdot 0.8 = \frac{96}{125}\frac{1}{\text{h}}$
6. $\lambda_{\text{high}} = \lambda_{\text{video}}{}'A(1 - P_{\text{low}}) = \frac{0.8}{\text{h}\cdot\text{Km}^2} \cdot 1.2\text{Km}^2 \cdot 0.2 = \frac{24}{125}\frac{1}{\text{h}}$
7. $\beta_{\text{low}} = \beta_{\text{high}} = \frac{3}{10}h$
8. $N = 4$



5.

$$(\lambda_{\text{voice}} + \lambda_{\text{low}} + \lambda_{\text{high}})\pi(0,0,0) = \mu_{\text{voice}}\pi(1,0,0) + \mu_{\text{low}}\pi(0,1,0) + \mu_{\text{high}}\pi(0,0,1) \tag{1}$$

$$(\lambda_{\text{voice}} + \lambda_{\text{low}} + \lambda_{\text{high}} + \mu_{\text{voice}})\pi(1,0,0) = 2\mu_{\text{voice}}\pi(2,0,0) + \mu_{\text{low}}\pi(1,1,0) + \mu_{\text{high}}\pi(1,0,1) + \lambda_{\text{voice}}\pi(0,0,0) \tag{2}$$

$$(\lambda_{\text{voice}} + \lambda_{\text{low}} + 2\mu_{\text{voice}})\pi(2,0,0) = 3\mu_{\text{voice}}\pi(3,0,0) + \mu_{\text{low}}\pi(2,1,0) + \lambda_{\text{voice}}\pi(1,0,0) \tag{3}$$

$$(\lambda_{\text{voice}} + 3\mu_{\text{voice}})\pi(3,0,0) = 4\mu_{\text{voice}}\pi(4,0,0) + \lambda_{\text{voice}}\pi(2,0,0) \tag{4}$$

$$4\mu_{\text{voice}}\pi(4,0,0) = \lambda_{\text{voice}}\pi(3,0,0) \tag{5}$$

$$(\lambda_{\text{voice}} + \lambda_{\text{low}} + \mu_{\text{low}})\pi(0,1,0) = \mu_{\text{voice}}\pi(1,1,0) + 2\mu_{\text{low}}\pi(0,2,0) + \lambda_{\text{low}}\pi(0,0,0) \tag{6}$$

$$(\lambda_{\text{voice}} + \mu_{\text{voice}} + \mu_{\text{low}})\pi(1,1,0) = 2\mu_{\text{voice}}\pi(2,1,0) + \lambda_{\text{voice}}\pi(0,1,0) + \lambda_{\text{low}}\pi(1,0,0) \tag{7}$$

$$(2\mu_{\text{voice}} + \mu_{\text{low}})\pi(2,1,0) = \lambda_{\text{voice}}\pi(1,1,0) + \lambda_{\text{low}}\pi(2,0,0) \tag{8}$$

$$2\mu_{\text{low}}\pi(0,2,0) = \lambda_{\text{low}}\pi(0,1,0) \tag{9}$$

$$(\lambda_{\text{voice}} + \mu_{\text{high}})\pi(0,0,1) = \mu_{\text{voice}}\pi(1,0,1) + \lambda_{\text{high}}\pi(0,0,0) \tag{10}$$

$$(\mu_{\text{voice}} + \mu_{\text{high}})\pi(1,0,1) = \lambda_{\text{voice}}\pi(0,0,1) + \lambda_{\text{high}}\pi(1,0,0) \tag{11}$$

TODO: derive solutions

6.

$$\pi\big(n_{\text{voice}}, n_{\text{low}}, n_{\text{high}}\big) = \frac{1}{G} \cdot \frac{\rho_{\text{voice}}^{n_{\text{voice}}}}{n_{\text{voice}}!} \cdot \frac{\rho_{\text{low}}^{n_{\text{low}}}}{n_{\text{low}}!} \cdot \frac{\rho_{\text{high}}^{n_{\text{high}}}}{n_{\text{high}}!}, \text{for } \big(n_{\text{voice}}, n_{\text{low}}, n_{\text{high}}\big) \in S$$

$$G := \sum_{n \in S} \frac{\rho_{\text{voice}}^{n_{\text{voice}}}}{n_{\text{voice}}!} \cdot \frac{\rho_{\text{low}}^{n_{\text{low}}}}{n_{\text{low}}!} \cdot \frac{\rho_{\text{high}}^{n_{\text{high}}}}{n_{\text{high}}!}$$

$$\text{with } \rho_{\text{voice}} := \lambda_{\text{voice}}\beta_{\text{voice}}, \rho_{\text{low}} := \lambda_{\text{low}}\beta_{\text{low}}, \rho_{\text{high}} := \lambda_{\text{high}}\beta_{\text{high}}$$

This gives us:
$G = 12.611312913333334$
$\pi(0,0,0) = 0.07929388532915936$
$\pi(1,0,0) = 0.1982347133228984$
$\pi(2,0,0) = 0.247793391653623$
$\pi(3,0,0) = 0.2064944930446858$
$\pi(4,0,0) = 0.12905905815292865$
$\pi(0,1,0) = 0.018269311179838318$
$\pi(1,1,0) = 0.045673277949595796$
$\pi(2,1,0) = 0.05709159743699474$
$\pi(0,2,0) = 0.0021046246479173745$
$\pi(0,0,1) = 0.004567327794959579$
$\pi(1,0,1) = 0.011418319487398949$

7.

TODO: explain blocking prob TODO: 1 - probability of being accepted $B_{\text{voice}} = \pi(4,0,0) + \pi(2,1,0) + \pi(0,2,0) + \pi(1,0,1) = 0.1996735997252397$
$B_{\text{low}} = \pi(3,0,0) + \pi(4,0,0) + \pi(1,1,0) + \pi(2,1,0) + \pi(0,2,0) + \pi(0,0,1) + \pi(1,0,1) = 0.45640869851448085$
$B_{\text{high}} = \pi(2,0,0) + \pi(3,0,0) + \pi(4,0,0) + \pi(0,1,0) + \pi(1,1,0) + \pi(2,1,0) + \pi(0,2,0) + \pi(0,0,1) + \pi(1,0,1) = 0.7224714013479422$

8.

TODO: explain kaufman roberts steps $g(0) = 1$
$g(1) = 30 \cdot \frac{1}{12} q(0) = 2.5$
$g(2) = \frac{1}{2}\left(30 \cdot \frac{1}{12} q(1) + \frac{96}{125} \cdot \frac{3}{10} \cdot 2q(0)\right) = 3.3554$
$g(3) = \frac{1}{3}\left(30 \cdot \frac{1}{12} q(2) + \frac{96}{125} \cdot \frac{3}{10} \cdot 2q(1) + \frac{24}{125} \cdot \frac{3}{10} \cdot 3q(0)\right) = 3.2377666666666673$
$g(4) = \frac{1}{4}\left(30 \cdot \frac{1}{12} q(3) + \frac{96}{125} \cdot \frac{3}{10} \cdot 2q(2) + \frac{24}{125} \cdot \frac{3}{10} \cdot 3q(1)\right) = 2.518146246666667$
$G = q(0) + g(1) + g(2) + g(3) + g(4) = 12.611312913333334$
$q(0) = \frac{g(0)}{G} = 0.07929388532915936$
$q(1) = \frac{g(1)}{G} = 0.19823471332289838$
$q(2) = \frac{g(2)}{G} = 0.2660627028334613$
$q(3) = \frac{g(3)}{G} = 0.2567350987892412$
$q(4) = \frac{g(4)}{G} = 0.1996735997252397$
$B_{\text{voice}} = q(4) = 0.1996735997252397$
$B_{\text{low}} = q(3) + q(4) = 0.45640869851448096$
$B_{\text{high}} = q(2) + q(3) + q(4) = 0.7224714013479422$

9. Kaufman-Roberts Recursion in Python

TODO: explain what the program is doing

```python
def kaufman(C: int, p: list[float], b: list[int]):
    K = len(p)
    q = [0.0] * (C + 1)
    q[0] = 1.0

    for c in range(1, C + 1):
        s = 0.0
        for k in range(K):
            if b[k] <= c:
                s += p[k] * b[k] * q[c - b[k]]
        q[c] = s / c

    G = sum(q)
    q = [qi / G for qi in q]

    B = [0.0] * K
    for k in range(K):
        s = 0.0
        for c in range(C + 1 - b[k], C + 1):
            s += q[c]
        B[k] = s

    return B

def main():
    C = 4
    p = [30 * 1/12, 96/125 * 3/10, 24/125 * 3/10]
    b = [1, 2, 3]
    B = kaufman(C, p, b)
    # Bv = 0.1996735997252397, Bl = 0.45640869851448096, Bh = 0.7224714013479422
    print(f"Bv = {B[0]}, Bl = {B[1]}, Bh = {B[2]}")

if __name__ == "__main__":
    main()
```

II. Optimal distribution of channels over neighboring cells in mobile voice networks

10.
TODO

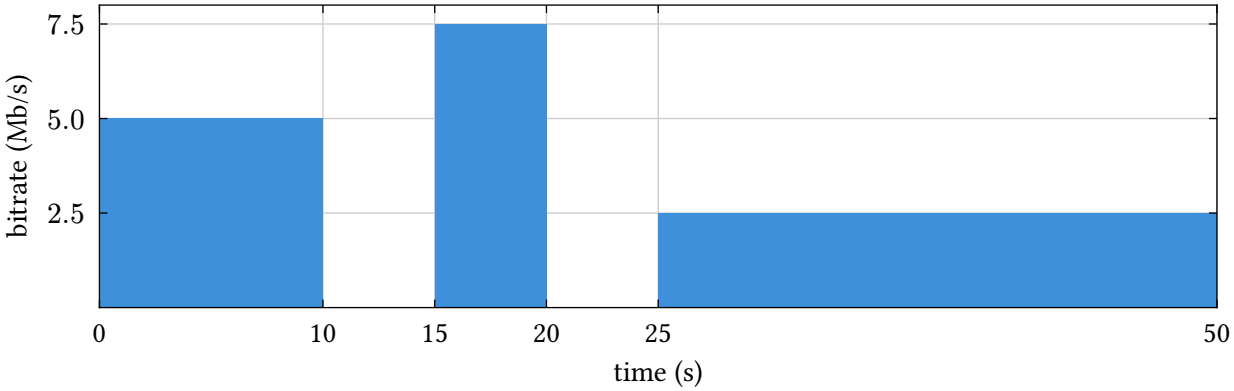
11.
TODO

12.
TODO

III. Traffic Management in IP networks

Before shaping/policing we have this incoming traffic.

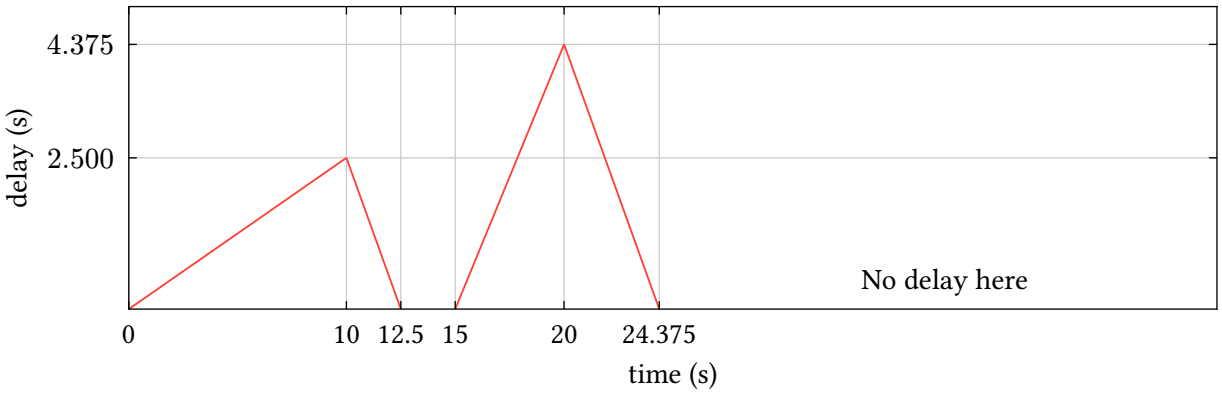
Bitrate before shaping



13. Traffic Shaping Delay Graph

1. For the first burst we have a peak rate of 5Mb/s distributed over 10 seconds. Since the shaping rate is 4Mb/s the shaping buffer level rises at rate $5\text{Mb/s} - 4\text{Mb/s} = 1\text{Mb/s}$. Which means that at the end of the burst the buffer level is $1\text{Mb/s} \cdot 10\text{s} = 10\text{Mb}$ and the delay is $\frac{10\text{Mb}}{4\text{Mb/s}} = 2.5\text{s}$.
2. For the first burst we have a peak rate of 7.5Mb/s distributed over 5 seconds. Since the shaping rate is 4Mb/s the shaping buffer level rises at rate $7.5\text{Mb/s} - 4\text{Mb/s} = 3.5\text{Mb/s}$. Which means that at the end of the burst the buffer level is $3.5\text{Mb/s} \cdot 5\text{s} = 17.5\text{Mb}$ and the delay is $\frac{17.5\text{Mb}}{4\text{Mb/s}} = 4.375\text{s}$.
3. For the third and final burst we have a peak rate of 2.5Mb/s, which is less than the 4Mb/s shaping rate. This means that the third burst is not delayed.

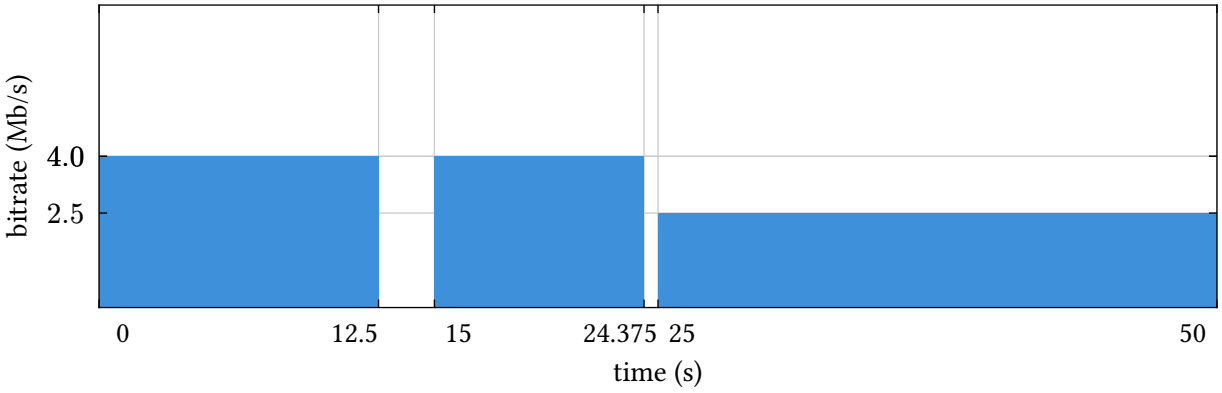
Delay intorduced by the shaper



14. Traffic Shaping Bitrate Graph

1. For the first burst we have a peak rate of 5Mb/s distributed over 10 seconds. Which means that the total ammount of traffic is $5\text{Mb/s} \cdot 10\text{s} = 50\text{Mb}$. Since the shaping rate is 4Mb/s the shaped burst is going to be 4Mb/s distributed over $\frac{50\text{Mb}}{4\text{Mb/s}} = 12.5\text{s}$.
2. For the second burst we have a peak rate of 7.5Mb/s distributed over 5 seconds. Which means that the total ammount of traffic is $7.5\text{Mb/s} \cdot 5\text{s} = 37.5\text{Mb}$. Since the shaping rate is 4Mb/s the shaped burst is going to be 4Mb/s distributed over $\frac{37.5\text{Mb}}{4\text{Mb/s}} = 9.375\text{s}$.
3. For the third and final burst we have a peak rate of 2.5Mb/s, which is less than the 4Mb/s shaping rate. This means that the third burst remains unchanged.

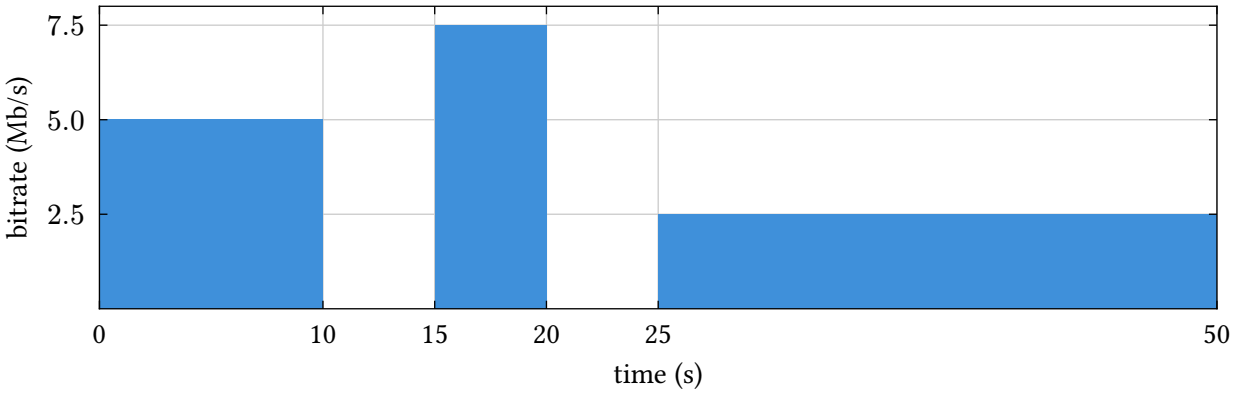
Bitrate after shaping



15. Traffic Policing Bitrate Graph

Trick question the bitrate is the same as before. Policing with marking (not when we drop the packets) does not reduce the bitrate, it only signals which packets exceed the allowed rate.

Bitrate before/after policing

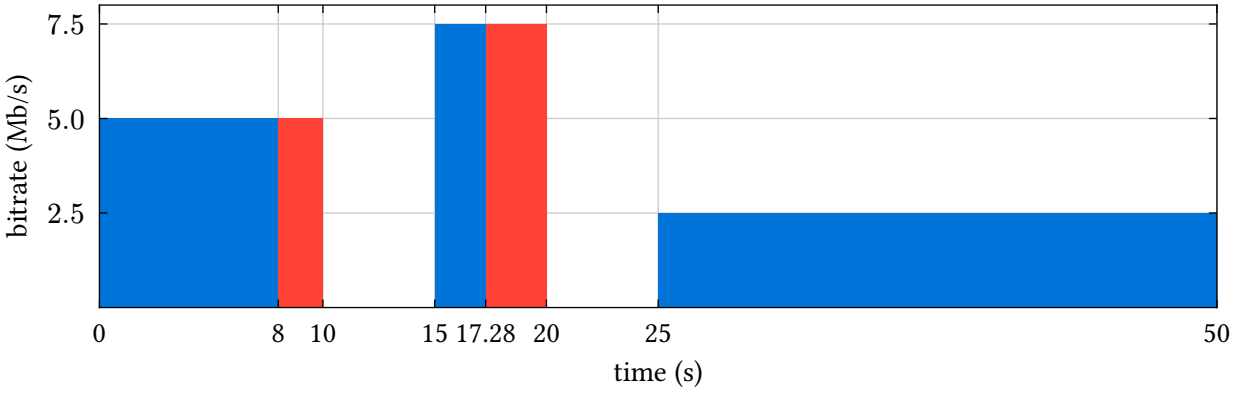


16. Traffic Policing Conforming Graph

Note that the "water volume" takes $\frac{8\text{Mb}}{4\text{Mb/s}} = 2s$ to become empty once it has reached the burst tolerance. Since we have a 5 second spacing between each burst we can consider each burst separately.

1. For the first burst we have a peak rate of 5Mb/s distributed over 10 seconds. Since the leak rate is 4Mb/s and the burst tolerance is 8Mb the "water volume" rises at rate $5\text{Mb/s} - 4\text{Mb/s} = 1\text{Mb/s}$, and after $\frac{8\text{Mb}}{1\text{Mb/s}} = 8\text{s}$ the "water volume" reaches the burst tolerance. Which means that in the first 8 seconds (in blue) the traffic is marked as conforming and between 8 and 10 seconds (in red) only $\frac{4}{5}$ of the packets are marked as conforming, and the remaining $\frac{1}{5}$ non-conforming.
2. For the second burst we have a peak rate of 7.5Mb/s distributed over 5 seconds. Since the leak rate is 4Mb/s and the burst tolerance is 8Mb the "water volume" rises at rate $7.5\text{Mb/s} - 4\text{Mb/s} = 3.5\text{Mb/s}$, and after $\frac{8\text{Mb}}{3.5\text{Mb/s}} \approx 2.28\text{s}$ the "water volume" reaches the burst tolerance. Which means that in the first 2.28 seconds (in blue) the traffic is marked as conforming and between 2.28 and 5 seconds (in red) only $\frac{4}{7.5}$ of the packets are marked as conforming, and the remaining $\frac{3.5}{7.5}$ non-conforming.
3. For the third and final burst we have a peak rate of 2.5Mb/s, which is less than the 4Mb/s leak rate. This means that the third burst is all compleately marked as conforming.

Conforming traffic after policing



IV. Performance of *TCP*-based networks

17. Downside of *TCP* Slow Start

The main downside of *TCP* Slow Start is that it can significantly underutilize available bandwidth at the beginning of a connection. Because *TCP* does not know the amount of available bandwidth at startup, it begins with a small congestion window that increases exponentially. This requires multiple *RTT*s to ramp up, which can lead to noticeable performance degradation, as the network is not used at full capacity during this phase.

This issue is especially apparent when transferring many small files, by the time the congestion window grows large enough to fully utilize the bandwidth, the file transfer is already complete. For example, this was a known problem in early versions of *HTTP*, where many small files were commonly transferred.

18. Slow Start Transfer Times

15KB

The client first establishes a *TCP* connection using the three-way handshake, which takes one *RTT* (60 ms). After the connection is established, the client transmits a single request packet to the server. This packet requires half *RTT* to reach the server. The server then processes the request for 50 ms and immediately begins transmitting the first response packet, which arrives at the client after the remaining half *RTT*. Up to this point, the transfer time is 170 ms (60ms + 50ms + 60ms).

During Slow Start, the server's congestion window doubles every *RTT*. After one *RTT*, the client receives two packets. After another *RTT*, the window doubles again and the client receives four packets. After one additional *RTT*, the window doubles to eight packets, although the server only needs to send 5.140 bytes of data, corresponding to four packets. At that point, the transfer completes.

The total time required is therefore:
- 1 *RTT* for the *TCP* handshake (60 ms)
- 50 ms of server processing
- 4 *RTT*s during Slow Start (4 x 60 ms)

This results in a total transfer time of 350 ms.

25KB and 40KB

The procedure is identical to the 15KB case, except that transferring these larger files requires one additional congestion window doubling. This adds one more *RTT* to the Slow Start phase. Therefore, the total transfer time for both the 25KB and 40KB files is 410 ms.

Final Results

The results can be summarized in the following table.

File size	Number of packets	Slow Start *RTT*s	Transmission Time
15KB	11	4	350ms
25KB	18	5	410ms
45KB	29	5	410ms