Advanced Computer Networking (ACN)

Exercise 4 – Solution

Prof. Dr.-Ing. Georg Carle

Sebastian Gallenmüller, Max Helm, Benedikt Jaeger,
Marcel Kempf, Patrick Sattler, Johannes Zirngibl

Chair of Network Architectures and Services
School of Computation, Information, and Technology
Technical University of Munich

Outline

Announcements

For questions and problems:

- Always use this mail address: `acn@net.in.tum.de`

Deadline first version:

- The deadline for the first version of Tutorial 4 has passed 15 minutes ago
- Commit and push your solution, if you haven't already

Next week:

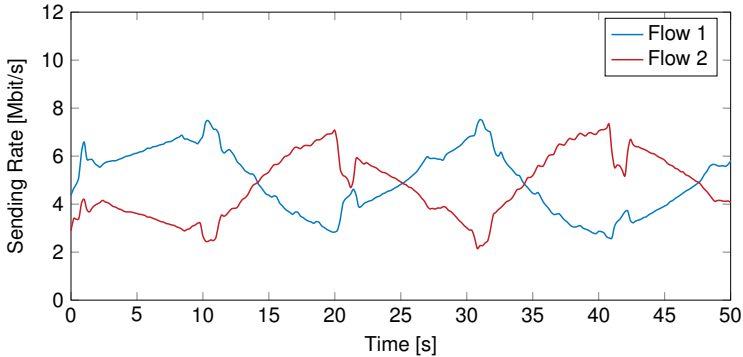
- Tuesday: Guest lecture
- Thursday: No lecture
- Deadline for Problem 2 of the projects is on **Tuesday, 19.12. 16:00**
- Second deadline for this tutorial is on **Thursday 21.12. 14:00**!
- First lecture in 2024 **Tuesday 9.1.2024 16:00**!

Announcements
Guest Lecture on Tuesday, Dec 19

- Presenter: Dr. Cornelius Diekmann
  - 2017 PhD: "Formal Verification of Network Security Management"
  - since 2017: Working for Google as Site Reliability Engineer (SRE)
- Topic: The Art of SLOs
  - Service Level Objectives (SLO) are used to define reliability goals for services
  - Experience how to define and achieve realistic SLOs for services
- Lecture will be streamed
- Lecture will NOT be recorded

1 a)

One of the flows uses Cubic congestion control, the other one BBR. Identify which of the flows uses BBR. No credits without short reasoning.



Tutorial 4 – Problem 1: TCP Congestion Control Fairness

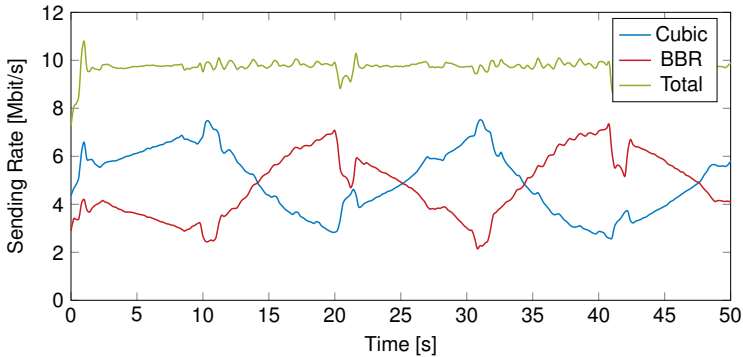
1 b)

Write a function `compute_sum()` which computes the total sending rate of two flows. It receives two lists and should return a list of the same size.

Tutorial 4 – Problem 1: TCP Congestion Control Fairness

1 b)

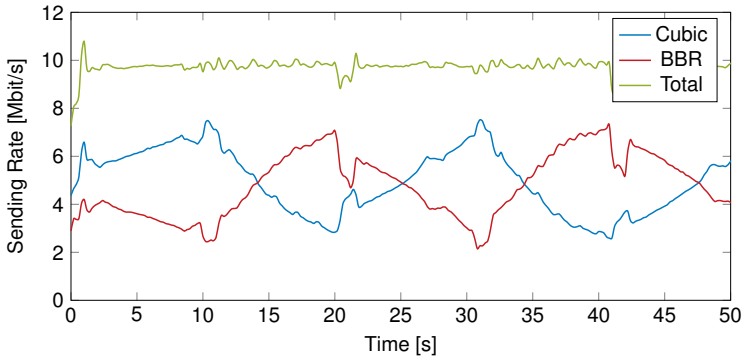
Write a function `compute_sum()` which computes the total sending rate of two flows. It receives two lists and should return a list of the same size.



Tutorial 4 – Problem 1: TCP Congestion Control Fairness

1 b)

Write a function `compute_sum()` which computes the total sending rate of two flows. It receives two lists and should return a list of the same size.



1 c)

Based on your results from b) estimate the bottleneck link's capacity. What happens if the total sending rate of the two flows exceeds this value?

Tutorial 4 – Problem 1: TCP Congestion Control Fairness

1 d)

The minimum RTT of both flows is 50 ms. Compute the path's BDP using the results from b) in kbit.

Tutorial 4 – Problem 1: TCP Congestion Control Fairness

1 d)

The minimum RTT of both flows is 50 ms. Compute the path's BDP using the results from b) in kbit.

- BDP : Bandwidth-Delay Product
- BDP = RTT $\cdot$ Bandwidth
- kbit = 10^3 bit
- From b): Bandwidth is 10 Mbit/s = $10 \cdot 10^6$ bit/s
- RTT is 50 ms = $50 \cdot 10^{-3}$ s

Tutorial 4 – Problem 1: TCP Congestion Control Fairness

1 d)

The minimum RTT of both flows is 50 ms. Compute the path's BDP using the results from b) in kbit.

- BDP : Bandwidth-Delay Product
- BDP = RTT · Bandwidth
- kbit = 10^3 bit
- From b): Bandwidth is 10 Mbit/s = $10 \cdot 10^6$ bit/s
- RTT is 50 ms = $50 \cdot 10^{-3}$ s

1 e)

In the following you will quantify the fairness of the two flows using Jain's Fairness Index. Explain two advantages of this index.

Tutorial 4 – Problem 1: TCP Congestion Control Fairness

1 d)

The minimum RTT of both flows is 50 ms. Compute the path's BDP using the results from b) in kbit.

- BDP : Bandwidth-Delay Product
- BDP = RTT · Bandwidth
- kbit = 10^3 bit
- From b): Bandwidth is 10 Mbit/s = $10 \cdot 10^6$ bit/s
- RTT is 50 ms = $50 \cdot 10^{-3}$ s

1 e)

In the following you will quantify the fairness of the two flows using Jain's Fairness Index. Explain two advantages of this index.

- It is within a fixed interval and always returns a value between 0 and 1
- It is scale free which means that its input does not have to be normalized
- It can be computed over an arbitrary number of flows
- It can be easily interpreted by humans. It is $\frac{k}{n}$ if there are k flows perfectly fair while the other $n - k$ shares are 0

Tutorial 4 – Problem 1: TCP Congestion Control Fairness

1 f)

Write a function `compute_fairness()` which computes Jain's Index.

Tutorial 4 – Problem 1: TCP Congestion Control Fairness

1 f)

Write a function `compute_fairness()` which computes Jain's Index.

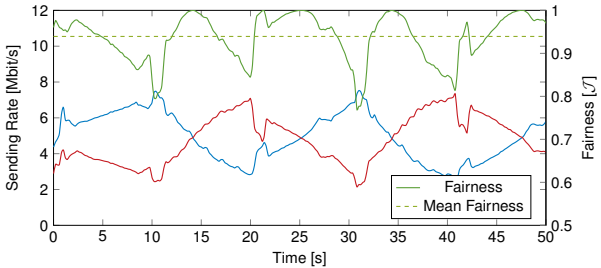
From RFC 5166: $\mathcal{J} = \frac{\left(\sum_i x_i\right)^2}{n \cdot \sum_i x_i^2}$, with shares $x_1, x_2, \ldots, x_n$

Tutorial 4 – Problem 1: TCP Congestion Control Fairness

1 f)

Write a function `compute_fairness()` which computes Jain's Index.

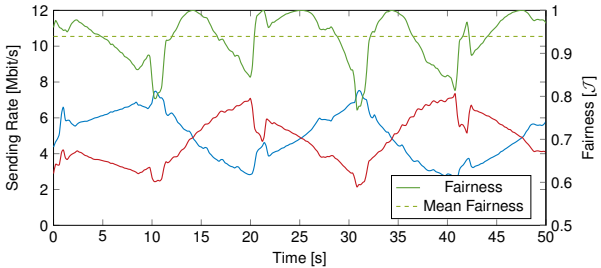
From RFC 5166: $\mathcal{J} = \frac{\left(\sum_i x_i\right)^2}{n \cdot \sum_i x_i^2}$, with shares $x_1, x_2, \ldots, x_n$



1 f)

Write a function `compute_fairness()` which computes Jain's Index.

From RFC 5166: $\mathcal{J} = \frac{\left(\sum_i x_i\right)^2}{n \cdot \sum_i x_i^2}$, with shares $x_1, x_2, \ldots, x_n$



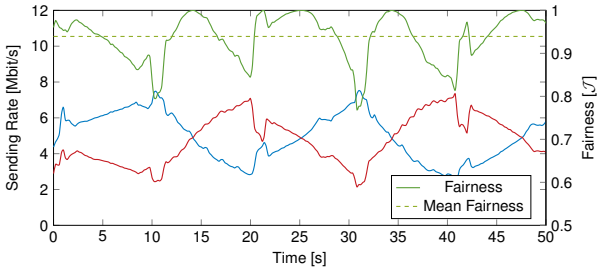
1 g)

Considering the results from e), assess the fairness between Cubic and BBR.

1 f)

Write a function `compute_fairness()` which computes Jain's Index.

From RFC 5166: $\mathcal{J} = \frac{\left(\sum_i x_i\right)^2}{n \cdot \sum_i x_i^2}$, with shares $x_1, x_2, \ldots, x_n$



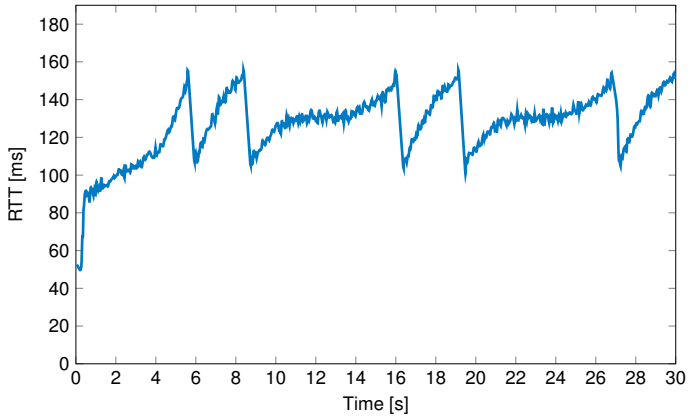
1 g)

Considering the results from e), assess the fairness between Cubic and BBR.

```
Flow1 transmitted 4.98 Mbit/s on average.
Flow2 transmitted 4.75 Mbit/s on average.
This results in a total fairness of 0.99945
```

2 a)

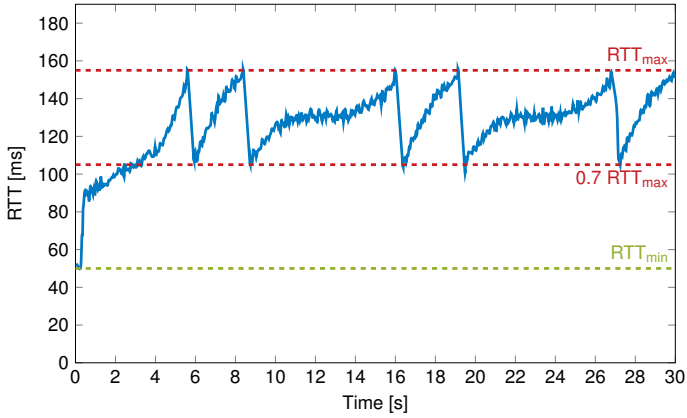
Have a look at the above graph. Which congestion control algorithm was used by the TCP sender?



Tutorial 4 – Problem 2: Exponential Weighted Moving Average

2 a)

Have a look at the above graph. Which congestion control algorithm was used by the TCP sender?



Tutorial 4 – Problem 2: Exponential Weighted Moving Average

2 b)

Write a function to compute the exponential weighted moving average. The function `ewma()` gets two parameters, a list of values and a value alpha as weight for the new samples.

Tutorial 4 – Problem 2: Exponential Weighted Moving Average

2 b)

Write a function to compute the exponential weighted moving average. The function `ewma()` gets two parameters, a list of values and a value alpha as weight for the new samples.

Input: Samples S,α
Output: EWMA E of S
Initialize E with first sample;
for s in S **do**
  E += $E[-1] \cdot (1 - \alpha) + s \cdot \alpha$
end
return E

Sample		EWMA
5		5
6	$5 \cdot 0.8 + 6 \cdot 0.2 =$	5.2
10	$5.2 \cdot 0.8 + 10 \cdot 0.2 =$	6.16
2	$6.16 \cdot 0.8 + 2 \cdot 0.2 =$	5.328
5	$5.328 \cdot 0.8 + 5 \cdot 0.2 =$	5.2624
...		

Tutorial 4 – Problem 2: Exponential Weighted Moving Average

2 b)

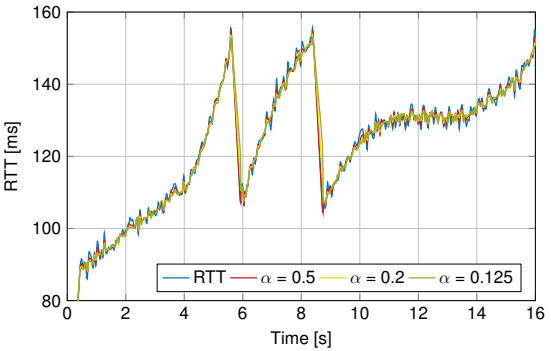
Write a function to compute the exponential weighted moving average. The function `ewma()` gets two parameters, a list of values and a value alpha as weight for the new samples.

Input: Samples S,α
Output: EWMA E of S
Initialize E with first sample;
for s in S **do**
  E += $E[-1] \cdot (1 - \alpha) + s \cdot \alpha$
end
return E

Sample		EWMA
5		5
6	$5 \cdot 0.8 + 6 \cdot 0.2 =$	5.2
10	$5.2 \cdot 0.8 + 10 \cdot 0.2 =$	6.16
2	$6.16 \cdot 0.8 + 2 \cdot 0.2 =$	5.328
5	$5.328 \cdot 0.8 + 5 \cdot 0.2 =$	5.2624
...		

2 c)

Compare the results when using different values for alpha and explain how it impacts the resulting average.



Tutorial 4 – Problem 2: Exponential Weighted Moving Average

2 d)

Write the function `compute_rto()` which computes the retransmission timeout (RTO) as explained in RFC 6298.

Tutorial 4 – Problem 2: Exponential Weighted Moving Average

2 d)

Write the function `compute_rto()` which computes the retransmission timeout (RTO) as explained in RFC 6298.

Initialize with first sample `R`:

```
SRTT <- R
RTTVAR <- R/2
RTO <- SRTT + max (G, K*RTTVAR)
```

Tutorial 4 – Problem 2: Exponential Weighted Moving Average

2 d)

Write the function `compute_rto()` which computes the retransmission timeout (RTO) as explained in RFC 6298.

Initialize with first sample `R`:

```
SRTT <- R
RTTVAR <- R/2
RTO <- SRTT + max (G, K*RTTVAR)
```

For each RTT sample `R'`:

```
RTTVAR <- (1 - beta) * RTTVAR + beta * |SRTT - R'|
SRTT <- (1 - alpha) * SRTT + alpha * R'
RTO <- SRTT + max (G, K*RTTVAR)
```

Tutorial 4 – Problem 2: Exponential Weighted Moving Average

2 d)

Write the function `compute_rto()` which computes the retransmission timeout (RTO) as explained in RFC 6298.

Initialize with first sample `R`:

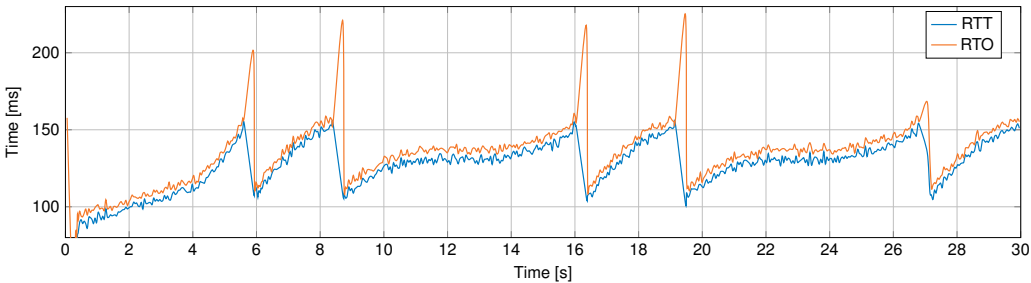
```
SRTT <- R
RTTVAR <- R/2
RTO <- SRTT + max (G, K*RTTVAR)
```

For each RTT sample `R'`:

```
RTTVAR <- (1 - beta) * RTTVAR + beta * |SRTT - R'|
SRTT <- (1 - alpha) * SRTT + alpha * R'
RTO <- SRTT + max (G, K*RTTVAR)
```

2 e)

Discuss the influence on TCP if the RTT is considerably overestimated or underestimated.



Tutorial 4 – Problem 3: QUIC

3 a)

Name all protocols which are usually (e.g. HTTP/1.1) used on top of IP when you visit https://acn.net.in.tum.de. Which protocols will be used when you would visit the same page with HTTP/3?

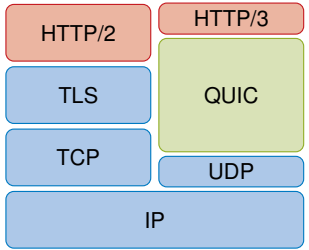
Tutorial 4 – Problem 3: QUIC

3 a)

Name all protocols which are usually (e.g. HTTP/1.1) used on top of IP when you visit https://acn.net.in.tum.de. Which protocols will be used when you would visit the same page with HTTP/3?

3 b)

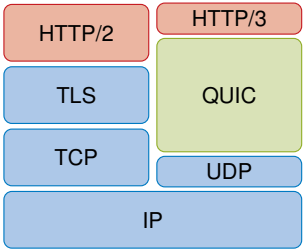
QUIC differentiates between packets and frames. Name all packet types available in QUIC.



Tutorial 4 – Problem 3: QUIC

3 a)

Name all protocols which are usually (e.g. HTTP/1.1) used on top of IP when you visit https://acn.net.in.tum.de. Which protocols will be used when you would visit the same page with HTTP/3?



3 b)

QUIC differentiates between packets and frames. Name all packet types available in QUIC.

- `https://www.rfc-editor.org/rfc/rfc9000.html#table-5`
- `https://www.rfc-editor.org/rfc/rfc9000.html#name-short-header-packets`

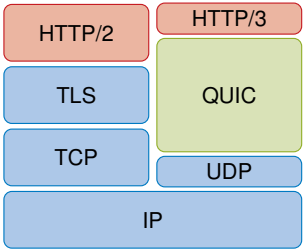
3 c)

Name 5 frame types specified in the RFC.

Tutorial 4 – Problem 3: QUIC

3 a)

Name all protocols which are usually (e.g. HTTP/1.1) used on top of IP when you visit https://acn.net.in.tum.de. Which protocols will be used when you would visit the same page with HTTP/3?



3 b)

QUIC differentiates between packets and frames. Name all packet types available in QUIC.

- `https://www.rfc-editor.org/rfc/rfc9000.html#table-5`
- `https://www.rfc-editor.org/rfc/rfc9000.html#name-short-header-packets`

3 c)

Name 5 frame types specified in the RFC.

- `https://www.rfc-editor.org/rfc/rfc9000.html#name-frame-types-and-formats`

3 d)

Which QUIC version is used in this connection. Paste the version ID as well as which version is specified by it. Also, find out which QUIC implementation was used to generate the qlog file.

3 d)

Which QUIC version is used in this connection. Paste the version ID as well as which version is specified by it. Also, find out which QUIC implementation was used to generate the qlog file.

→ Import `qlog` file to `https://qvis.quictools.info/`

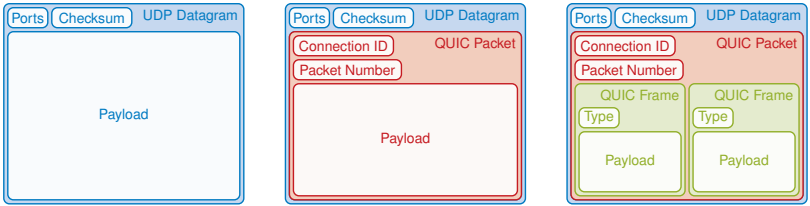
3 d)

Which QUIC version is used in this connection. Paste the version ID as well as which version is specified by it. Also, find out which QUIC implementation was used to generate the qlog file.

→ Import `qlog` file to `https://qvis.quictools.info/`

3 e)

Name the packet type included in this event and explain why this packet type has to be used. Name and briefly explain all frame types in this event.



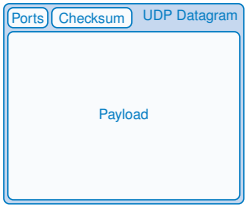
3 d)

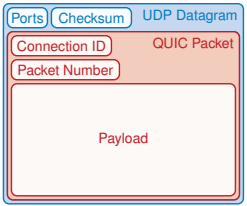
Which QUIC version is used in this connection. Paste the version ID as well as which version is specified by it. Also, find out which QUIC implementation was used to generate the qlog file.

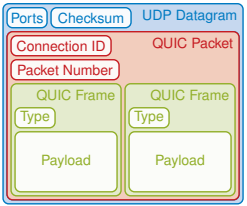
→ Import `qlog` file to `https://qvis.quictools.info/`

3 e)

Name the packet type included in this event and explain why this packet type has to be used. Name and briefly explain all frame types in this event.







3 f)

Which event carries the response of the server? How large is the file requested by the client?

Next Steps:

- Update your solution
- Do not copy-paste this sample solution