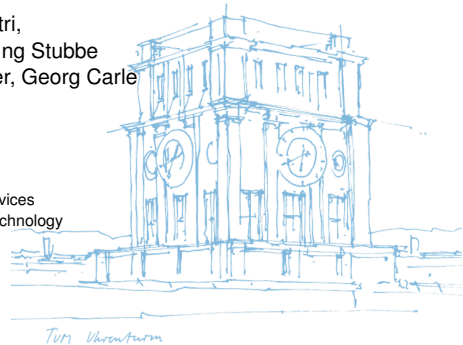
Chair of Network Architectures and Services
School of Computation, Information, and Technology
Technical University of Munich

Forward Error Correction and Weighted Hierarchical Fair Multiplexing for HTTP/3 over QUIC

Kilian Holzinger, Daniel Petri,
Stefan Lachnit, Marcel Kempf, Henning Stubbe
Sebastian Gallenmüller, Stephan Günther, Georg Carle

Tuesday 27th May, 2025

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



Introduction

- QUIC is the transport of the HTTP/3 web protocol
- Web application performance is crucial
- Clients need to receive the *right* data at the *right* time

Introduction

- QUIC is the transport of the HTTP/3 web protocol
- Web application performance is crucial
- Clients need to receive the *right* data at the *right* time

Challenge	Contribution
Unreliable network paths	Complement retransmission-based recovery with forward error correction (FEC) coded information

Introduction

- QUIC is the transport of the HTTP/3 web protocol
- Web application performance is crucial
- Clients need to receive the *right* data at the *right* time

Challenge	Contribution
Unreliable network paths	Complement retransmission-based recovery with forward error correction (FEC) coded information
Send the right data at the right time	Weighted hierarchical max-min fair multiplex scheduling

Introduction

- QUIC is the transport of the HTTP/3 web protocol
- Web application performance is crucial
- Clients need to receive the *right* data at the *right* time

Challenge	Contribution
Unreliable network paths	Complement retransmission-based recovery with forward error correction (FEC) coded information
Send the right data at the right time	Weighted hierarchical max-min fair multiplex scheduling
Priority signaling	Additional HTTP/3 priority parameters

Introduction

- QUIC is the transport of the HTTP/3 web protocol
- Web application performance is crucial
- Clients need to receive the *right* data at the *right* time

Challenge	Contribution
Unreliable network paths	Complement retransmission-based recovery with forward error correction (FEC) coded information
Send the right data at the right time	Weighted hierarchical max-min fair multiplex scheduling
Priority signaling	Additional HTTP/3 priority parameters

$\Rightarrow$ Assessment using prototype of experimental protocol changes

Related Work

QUIC with FEC

- Internet standard drafts on packet level [18, 13] and on frame level [3]
- Several research articles: *Adding FEC to QUIC* [11], *QUIC-FEC* [4], *rQUIC* [12], *FIEC* [10], *QUIRL* [9]
- Our approach most similar to recent QUIRL [9]

Related Work

QUIC with FEC

- Internet standard drafts on packet level [18, 13] and on frame level [3]
- Several research articles: *Adding FEC to QUIC* [11], *QUIC-FEC* [4], *rQUIC* [12], *FIEC* [10], *QUIRL* [9]
- Our approach most similar to recent QUIRL [9]
- QUIRL:
  - Advantageous convolutional Tetrys coding scheme (RFC 9407 [2])
  - Evaluation of bulk HTTP/3 transfers
  - Improvements to video streaming QoE metrics
  - Application specific scheduling of redundant information
  - Measurements over Starlink and emulated paths
  - Code available only partially

Related Work

QUIC with FEC

- Internet standard drafts on packet level [18, 13] and on frame level [3]
- Several research articles: *Adding FEC to QUIC* [11], *QUIC-FEC* [4], *rQUIC* [12], *FIEC* [10], *QUIRL* [9]
- Our approach most similar to recent QUIRL [9]
- QUIRL:
  - Advantageous convolutional Tetrys coding scheme (RFC 9407 [2])
  - Evaluation of bulk HTTP/3 transfers
  - Improvements to video streaming QoE metrics
  - Application specific scheduling of redundant information
  - Measurements over Starlink and emulated paths
  - Code available only partially

HMM Scheduling

- Hierarchical link sharing for Ethernet traffic classes [8]
- Implemented in Linux [8] and Omnet++ simulator [7]
- Similar approaches are described for SDN and Ethernet switches [21, 16, 17]

Related Work

QUIC with FEC

- Internet standard drafts on packet level [18, 13] and on frame level [3]
- Several research articles: *Adding FEC to QUIC* [11], *QUIC-FEC* [4], *rQUIC* [12], *FIEC* [10], *QUIRL* [9]
- Our approach most similar to recent QUIRL [9]
- QUIRL:
  - Advantageous convolutional Tetrys coding scheme (RFC 9407 [2])
  - Evaluation of bulk HTTP/3 transfers
  - Improvements to video streaming QoE metrics
  - Application specific scheduling of redundant information
  - Measurements over Starlink and emulated paths
  - Code available only partially

HMM Scheduling

- Hierarchical link sharing for Ethernet traffic classes [8]
- Implemented in Linux [8] and Omnet++ simulator [7]
- Similar approaches are described for SDN and Ethernet switches [21, 16, 17]

HTTP Prioritization

- Study revealed: Different browsers send different signals [6]
- Weighted incremental scheduling beneficial for low latency and largest contentful paint (LCP) [5]
- Better control over prioritization is needed [1, 20]

Forward Error Correction for HTTP/3

Tetrys Coding Scheme [19] (RFC 9407 [2])

-
-
-
-
-
-

Tetrys Coding Scheme [19] (RFC 9407 [2])

Systematic convolutional random linear erasure correction coding scheme for unicast communication

-
-
-
-
-
-

Tetrys Coding Scheme [19] (RFC 9407 [2])

Systematic convolutional random linear erasure correction coding scheme for unicast communication

- Protection against packet erasures
-
-
-
-
-

Tetrys Coding Scheme [19] (RFC 9407 [2])

Systematic convolutional random linear erasure correction coding scheme for unicast communication

- Protection against packet erasures
- Dynamic encoding window, containing non-acknowledged symbols
- Back channel for acknowledgement $\Rightarrow$ burst loss tolerant
-
-
-

Tetrys Coding Scheme [19] (RFC 9407 [2])

Systematic convolutional random linear erasure correction coding scheme for unicast communication

- Protection against packet erasures
- Dynamic encoding window, containing non-acknowledged symbols
- Back channel for acknowledgement $\Rightarrow$ burst loss tolerant
- Source Symbols carry original data, Repair Symbols add redundancy
-
-

Tetrys Coding Scheme [19] (RFC 9407 [2])

Systematic convolutional random linear erasure correction coding scheme for unicast communication

- Protection against packet erasures
- Dynamic encoding window, containing non-acknowledged symbols
- Back channel for acknowledgement $\Rightarrow$ burst loss tolerant
- Source Symbols carry original data, Repair Symbols add redundancy
- Repair Symbols consist of random linear combination of encoding window
- Using Galois Field arithmetic (GF(256)), coefficients drawn from seeded pseudo-random number generator

Forward Error Correction for HTTP/3

Integration into QUIC

Protocol extension:

- Based on draft [3] by QUIRL [9] authors
- New transport parameters to negotiate FEC extension
- Additional QUIC frame types

Forward Error Correction for HTTP/3

Integration into QUIC

Protocol extension:

- Based on draft [3] by QUIRL [9] authors
- New transport parameters to negotiate FEC extension
- Additional QUIC frame types
  - `SOURCE_SYMBOL`:
    - `SID` field: Identifier
    - `fec_session` field: Multiple FEC contexts
    - `payload` field: Encapsulates protected `STREAM` frame

Forward Error Correction for HTTP/3

Integration into QUIC

Protocol extension:

- Based on draft [3] by QUIRL [9] authors
- New transport parameters to negotiate FEC extension
- Additional QUIC frame types
  - `SOURCE_SYMBOL`:
    - `SID` field: Identifier
    - `fec_session` field: Multiple FEC contexts
    - `payload` field: Encapsulates protected `STREAM` frame
  - `REPAIR`
    - `smallest_sid`, `largest_sid` to convey encoding window
    - `seed_offset` allows multiple REPAIR frames with same dimensions but different coefficients

Forward Error Correction for HTTP/3

Integration into QUIC

Protocol extension:

- Based on draft [3] by QUIRL [9] authors
- New transport parameters to negotiate FEC extension
- Additional QUIC frame types
  - `SOURCE_SYMBOL`:
    - `SID` field: Identifier
    - `fec_session` field: Multiple FEC contexts
    - `payload` field: Encapsulates protected `STREAM` frame
  - `REPAIR`
    - `smallest_sid`, `largest_sid` to convey encoding window
    - `seed_offset` allows multiple REPAIR frames with same dimensions but different coefficients
  - `SYMBOL_ACK`

Forward Error Correction for HTTP/3

Integration into QUIC

Protocol extension:

- Based on draft [3] by QUIRL [9] authors
- New transport parameters to negotiate FEC extension
- Additional QUIC frame types
  - `SOURCE_SYMBOL`:
    - `SID` field: Identifier
    - `fec_session` field: Multiple FEC contexts
    - `payload` field: Encapsulates protected `STREAM` frame
  - `REPAIR`
    - `smallest_sid`, `largest_sid` to convey encoding window
    - `seed_offset` allows multiple REPAIR frames with same dimensions but different coefficients
  - `SYMBOL_ACK`

Prototype implementation:

- Packets with FEC-decoded information treated as lost in the congestion controller
- Integrated in Cloudflare quiche
- libmoepgf for fast Galois Field arithmetic with SIMD intrinsics

Forward Error Correction for HTTP/3

Repair Symbol Scheduling

Application specific parameters:

- **Repair delay tolerance**:

- **Burst loss tolerance**:

Forward Error Correction for HTTP/3

Repair Symbol Scheduling

Application specific parameters:

- **Repair delay tolerance**: Tolerable recovery time in addition to path one-way delay
  - Assumption: Loss events are short-timed

  $\Rightarrow$ Sending repair information later decreases chances to be affected by same loss event
- **Burst loss tolerance**:

Forward Error Correction for HTTP/3

Repair Symbol Scheduling

Application specific parameters:

- **Repair delay tolerance**: Tolerable recovery time in addition to path one-way delay
  - Assumption: Loss events are short-timed

  $\Rightarrow$ Sending repair information later decreases chances to be affected by same loss event
- **Burst loss tolerance**: Count of lost packets that the FEC mechanism should be able to recover

Forward Error Correction for HTTP/3

Repair Symbol Scheduling

Application specific parameters:

- **Repair delay tolerance**: Tolerable recovery time in addition to path one-way delay
  - Assumption: Loss events are short-timed
  - $\Rightarrow$ Sending repair information later decreases chances to be affected by same loss event
- **Burst loss tolerance**: Count of lost packets that the FEC mechanism should be able to recover

Bulk / non-incremental transfers:

Streaming / incremental transfers:

Forward Error Correction for HTTP/3

Repair Symbol Scheduling

Application specific parameters:

- **Repair delay tolerance**: Tolerable recovery time in addition to path one-way delay
  - Assumption: Loss events are short-timed
  - $\Rightarrow$ Sending repair information later decreases chances to be affected by same loss event
- **Burst loss tolerance**: Count of lost packets that the FEC mechanism should be able to recover

Bulk / non-incremental transfers:

- Optimized metric: Time to completion
- During transfer, rely on retransmissions, avoid overhead of redundancy
- End of transfer: Send repair symbols

Streaming / incremental transfers:

Forward Error Correction for HTTP/3

Repair Symbol Scheduling

Application specific parameters:

- **Repair delay tolerance**: Tolerable recovery time in addition to path one-way delay
  - Assumption: Loss events are short-timed
  - $\Rightarrow$ Sending repair information later decreases chances to be affected by same loss event
- **Burst loss tolerance**: Count of lost packets that the FEC mechanism should be able to recover

Bulk / non-incremental transfers:

- Optimized metric: Time to completion
- During transfer, rely on retransmissions, avoid overhead of redundancy
- End of transfer: Send repair symbols

Streaming / incremental transfers:

- Optimized metric: Byte-wise one-way delay
- Intermittent sending of Repair Symbols

Hierarchical Max-Min Fair Weighted Round-Robin Stream Scheduling

Hierarchical Max-Min Fair Weighted Round-Robin Stream Scheduling

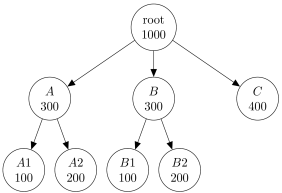
Original work [8]:

Adaptations for QUIC:

Hierarchical Max-Min Fair Weighted Round-Robin Stream Scheduling

Original work [8]:

- **Static** hierarchy of **Ethernet** priorities

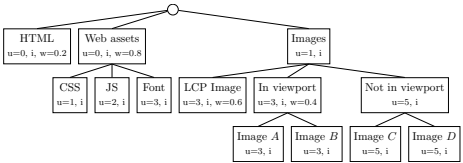


Example Ethernet priority hierarchy

Source: [8]

Adaptations for QUIC:

- **Dynamic** hierarchy of **stream** priorities

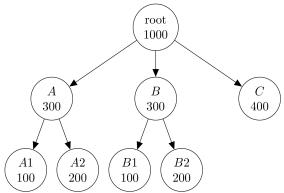


Example priority hierarchy of a web application

Hierarchical Max-Min Fair Weighted Round-Robin Stream Scheduling

Original work [8]:

- **Static** hierarchy of **Ethernet** priorities
- **Packet-granular** max-min fair scheduling

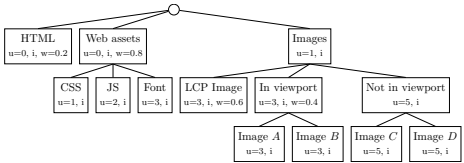


Example Ethernet priority hierarchy

Source: [8]

Adaptations for QUIC:

- **Dynamic** hierarchy of **stream** priorities
- **Byte-granular** max-min fair scheduling



Example priority hierarchy of a web application

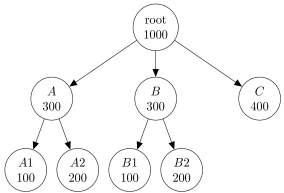
Hierarchical Max-Min Fair Weighted Round-Robin Stream Scheduling

Original work [8]:

- **Static** hierarchy of **Ethernet** priorities
- **Packet-granular** max-min fair scheduling
- **Absolute or relative** weights dividing **link bandwidth**

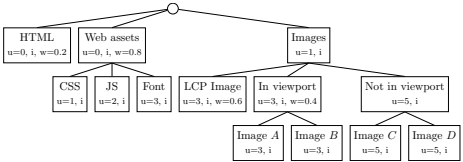
Adaptations for QUIC:

- **Dynamic** hierarchy of **stream** priorities
- **Byte-granular** max-min fair scheduling
- Relative weights dividing **scheduling round**



Example Ethernet priority hierarchy

Source: [8]



Example priority hierarchy of a web application

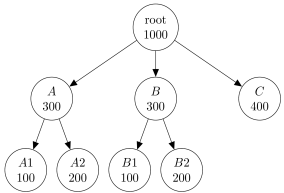
Hierarchical Max-Min Fair Weighted Round-Robin Stream Scheduling

Original work [8]:

- **Static** hierarchy of **Ethernet** priorities
- **Packet-granular** max-min fair scheduling
- **Absolute or relative** weights dividing **link bandwidth**
- Proven lower limit of scheduling round size

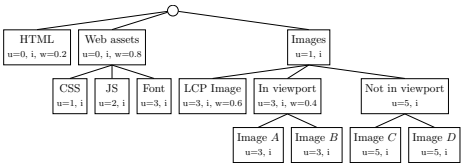
Adaptations for QUIC:

- **Dynamic** hierarchy of **stream** priorities
- **Byte-granular** max-min fair scheduling
- Relative weights dividing **scheduling round**
- Round size based on path MTU



Example Ethernet priority hierarchy

Source: [8]



Example priority hierarchy of a web application

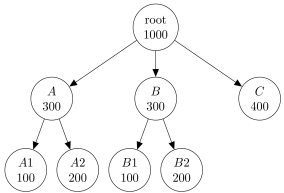
Hierarchical Max-Min Fair Weighted Round-Robin Stream Scheduling

Original work [8]:

- **Static** hierarchy of **Ethernet** priorities
- **Packet-granular** max-min fair scheduling
- **Absolute or relative** weights dividing **link bandwidth**
- Proven lower limit of scheduling round size
- **No** support for **strict priorities**

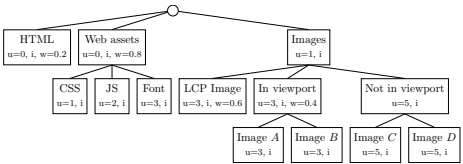
Adaptations for QUIC:

- **Dynamic** hierarchy of **stream** priorities
- **Byte-granular** max-min fair scheduling
- Relative weights dividing **scheduling round**
- Round size based on path MTU
- Support for **strict priorities**



Example Ethernet priority hierarchy

Source: [8]



Example priority hierarchy of a web application

New HTTP Priority Parameters

Extensible Prioritization Scheme (EPS) for HTTP/3
(RFC 9218) [15]

New HTTP Priority Parameters

Extensible Prioritization Scheme (EPS) for HTTP/3 (RFC 9218) [15]

- Two parameters, encoded in Structured Field Values (SFV) format (RFC 8941 [14])
  - Urgency u: strict priority with 0 highest, 7 lowest
  - Incremental i: boolean
- If i, then parallel transactions sent in equally weighted round-robin

New HTTP Priority Parameters

Extensible Prioritization Scheme (EPS) for HTTP/3 (RFC 9218) [15]

- Two parameters, encoded in Structured Field Values (SFV) format (RFC 8941 [14])
  - Urgency u: strict priority with 0 highest, 7 lowest
  - Incremental i: boolean
- If i, then parallel transactions sent in equally weighted round-robin

Additional experimental parameters:

- List of parents `exp_p` in ascending order
- Burst loss tolerance `exp_b`
- Repair delay tolerance `exp_d`
- Relative weight `exp_w`

New HTTP Priority Parameters

Extensible Prioritization Scheme (EPS) for HTTP/3 (RFC 9218) [15]

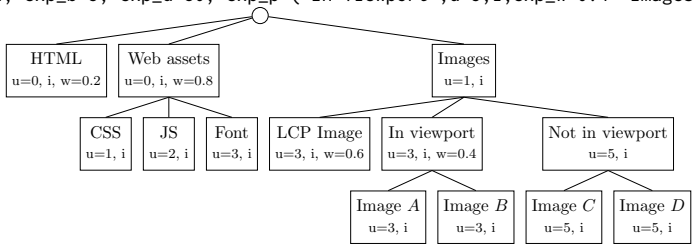
- Two parameters, encoded in Structured Field Values (SFV) format (RFC 8941 [14])
  - Urgency *u*: strict priority with 0 highest, 7 lowest
  - Incremental *i*: boolean
- If *i*, then parallel transactions sent in equally weighted round-robin

Additional experimental parameters:

- List of parents `exp_p` in ascending order
- Burst loss tolerance `exp_b`
- Repair delay tolerance `exp_d`
- Relative weight `exp_w`

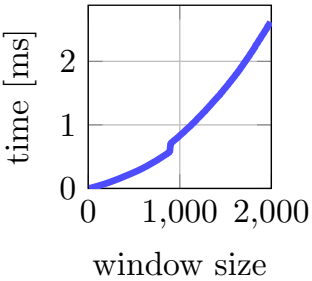
Example priority SFV string for *Image A*

```
u=3, i, exp_b=3, exp_d=80, exp_p=("In viewport";u=3;i;exp_w=0.4 "Images";u=1)
```



Evaluation

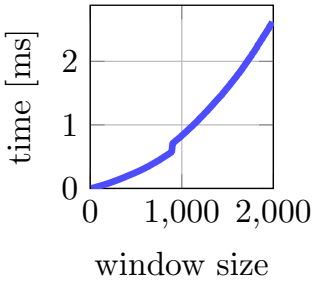
FEC Coding Performance in Benchmark



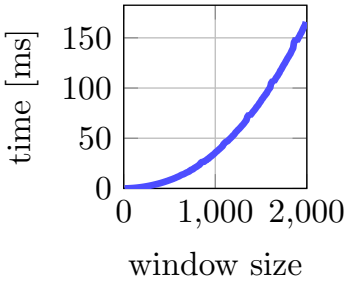
Encoding time of a single repair symbol

Evaluation

FEC Coding Performance in Benchmark



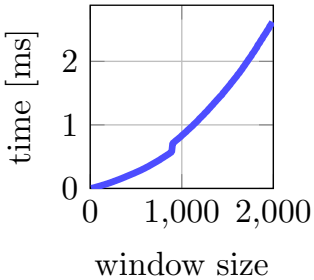
Encoding time of a single repair symbol



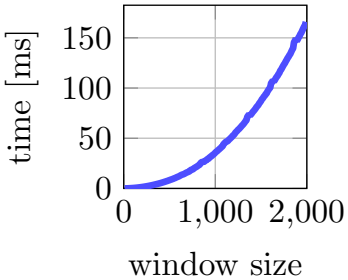
Decoding time with 5% of source symbol loss at the **begin** of the window

Evaluation

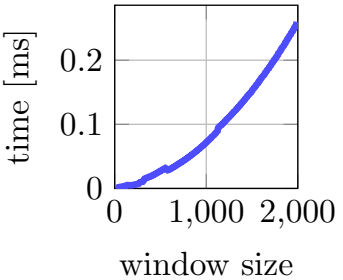
FEC Coding Performance in Benchmark



Encoding time of a single repair symbol



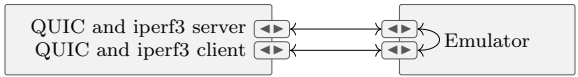
Decoding time with 5% of source symbol loss at the **begin** of the window



Decoding time with 5% of source symbol loss at the **end** of the window

Evaluation

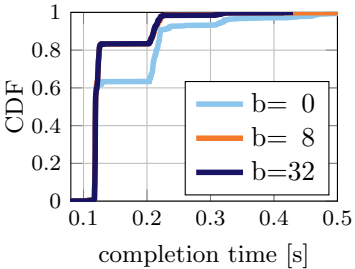
Testbed Setup



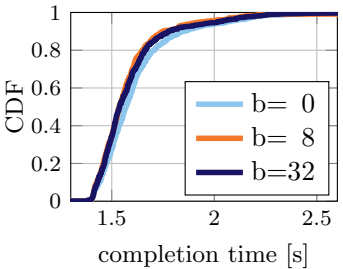
- Server and client on same machine for same clock source
- Emulation with netem
  - Gilbert-Elliot stateful loss
  - 50 ms bidirectional delay
  - 100 Mbit/s path capacity

Evaluation

Cumulative distribution of transfer completion times of non-incremental HTTP/3 transactions



File size: 32 kB

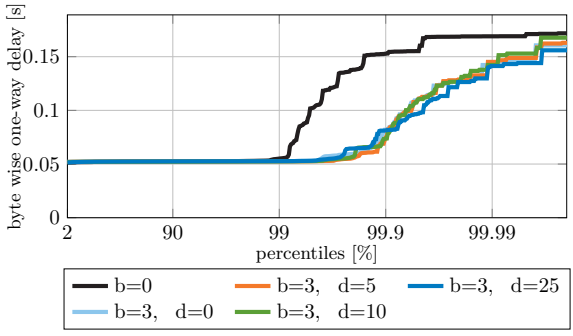


File size: 1 MB

- 1000 samples for each configuration
- Effect more pronounced for smaller transactions

Evaluation

Incremental Web Assets

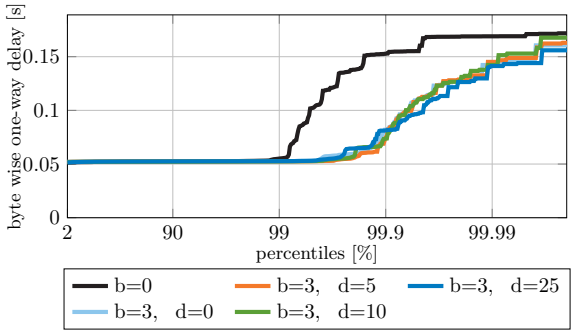


Statistical distribution of the measured byte-wise delay of a streaming web resource with a logarithmic horizontal axis (HDR plot)

- 10 Mbit/s bidirectional cross traffic to emulate temporal stochastic loss process
- Application workload: 5 kB chunks @ 60 Hz, 100 MB in total for each configuration

Evaluation

Incremental Web Assets



Statistical distribution of the measured byte-wise delay of a streaming web resource with a logarithmic horizontal axis (HDR plot)

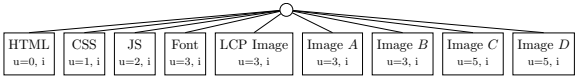
- 10 Mbit/s bidirectional cross traffic to emulate temporal stochastic loss process
- Application workload: 5 kB chunks @ 60 Hz, 100 MB in total for each configuration

⇒ Significant improvement of end-to-end one-way delay

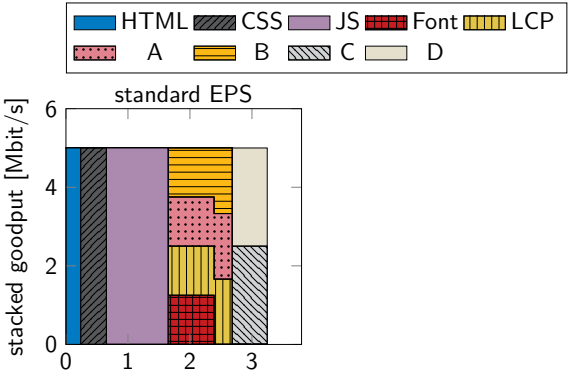
⇒ Repair delay tolerance has minimal effect

Evaluation

Hierarchical Max-Min Fair Stream Scheduling



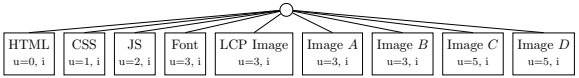
Flat scheduling hierarchy, compatible with standard EPS



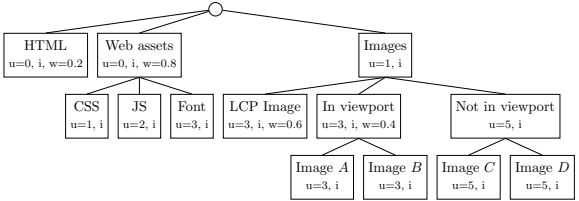
Offline scheduler behavior with 5 Mbit/s link rate

Evaluation

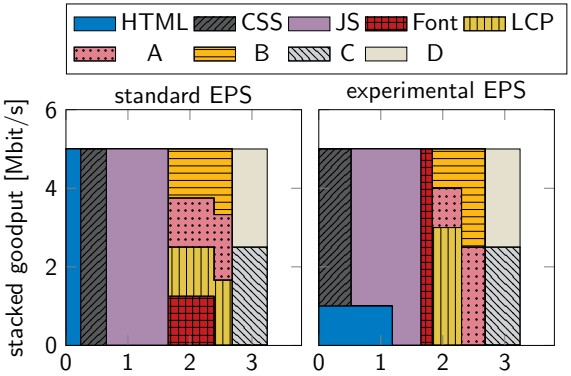
Hierarchical Max-Min Fair Stream Scheduling



Flat scheduling hierarchy, compatible with standard EPS



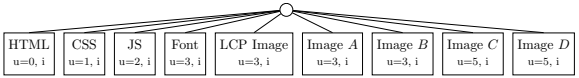
Scheduling hierarchy achieved with experimental EPS



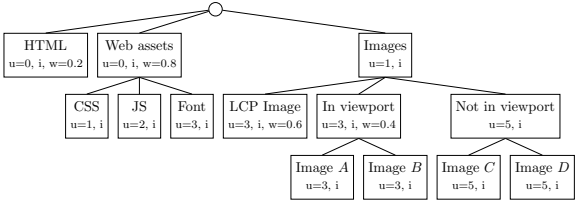
Offline scheduler behavior with 5 Mbit/s link rate

Evaluation

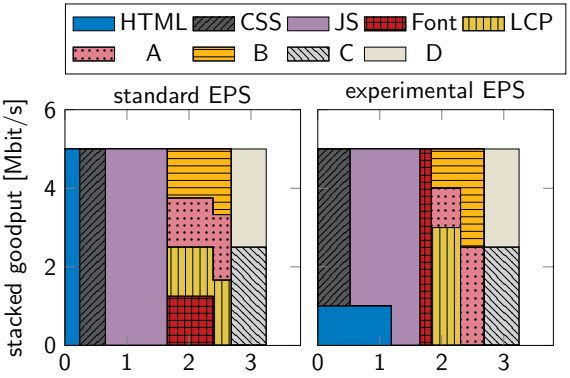
Hierarchical Max-Min Fair Stream Scheduling



Flat scheduling hierarchy, compatible with standard EPS



Scheduling hierarchy achieved with experimental EPS

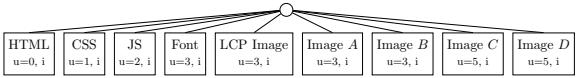


Offline scheduler behavior with 5 Mbit/s link rate

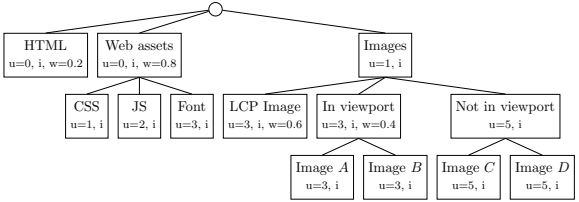
⇒ HTML is loaded in 1:4 ratio to other assets, JS and CSS strictly loaded in parallel within their class, LCP finalizes earlier

Evaluation

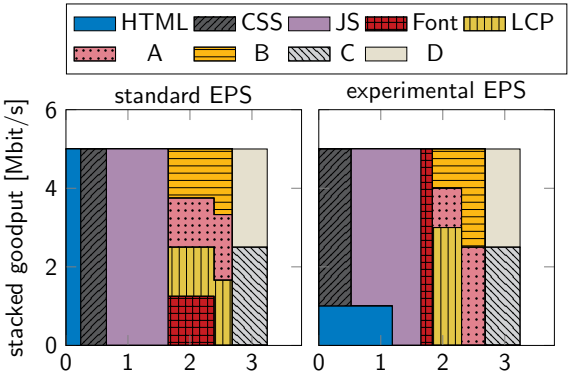
Hierarchical Max-Min Fair Stream Scheduling



Flat scheduling hierarchy, compatible with standard EPS



Scheduling hierarchy achieved with experimental EPS



Offline scheduler behavior with 5 Mbit/s link rate

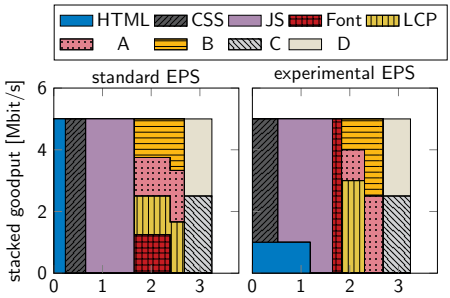
⇒ HTML is loaded in 1:4 ratio to other assets, JS and CSS strictly loaded in parallel within their class, LCP finalizes earlier

⇒ mix and match of strict priorities and weighted incremental transfers becomes feasible

Evaluation

Hierarchical Max-Min Fair Stream Scheduling

Offline scheduler behavior with 5 Mbit/s link rate

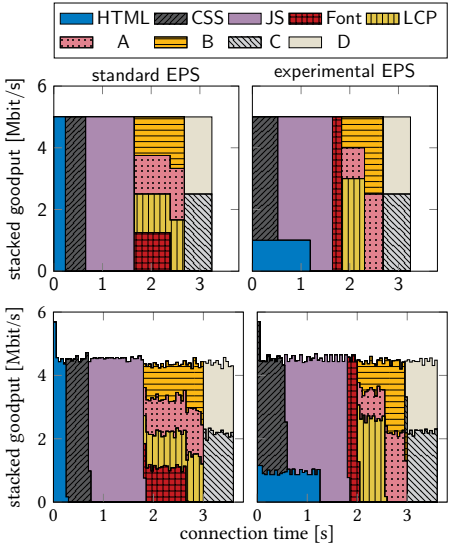


Evaluation

Hierarchical Max-Min Fair Stream Scheduling

Offline scheduler behavior with 5 Mbit/s link rate

Testbed behavior of prototype online scheduler, 5 Mbit/s link rate, 10 ms RTT

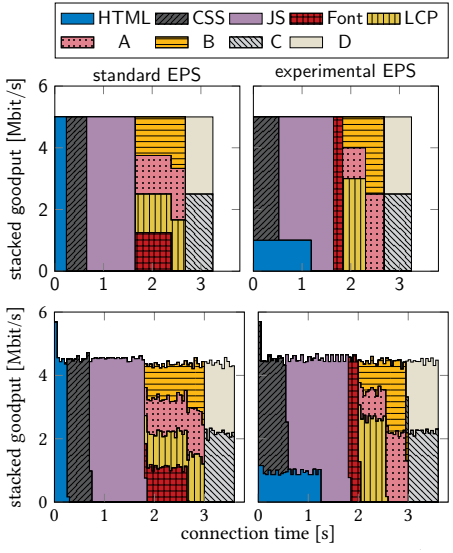


Evaluation

Hierarchical Max-Min Fair Stream Scheduling

Offline scheduler behavior with 5 Mbit/s link rate

Testbed behavior of prototype online scheduler, 5 Mbit/s link rate, 10 ms RTT



- Online behavior closely matches offline schedule
- Excess bandwidth fairly distributed
- More parallelism ⇒ potentially less head of line blocking

Conclusion

Investigated ways to make QUIC faster:

- Integration of Tetrys FEC coding scheme
  - Repair Symbol scheduling strategies for non-incremental and incremental web workloads
  - Significant timing improvements achievable
  - Robust also to bursty loss patterns
- Added hierarchical weighted round-robin scheduler
  - Finer control over order of sent data
  - Fallback safe with standard EPS
  - HMM-fair on byte-granularity
  - Improves parallelism
- Defined experimental EPS signals to control behavior of extensions

Slides (pdf)[1]

Paper (pdf)[2]

Code[3]

[2] http://www.net.in.tum.de/fileadmin/bibtex/publications/papers/holzinger2025quic_slides.pdf

[2] http://www.net.in.tum.de/fileadmin/bibtex/publications/papers/holzinger2025quic.pdf

[3] https://github.com/holzingk/quic-fec-eps/

Bibliography

[1] F. Bulgarella et al.
Performance measurements of QUIC communications.
In *Proceedings of the 2019 Applied Networking Research Workshop*, pages 8–14, 2019.

[2] J. Detchart, E. Lochin, J. Lacan, and V. Roca.
Tetrys: An On-the-Fly Network Coding Protocol.
RFC 9407, 2023.

[3] F. Michel and O. Bonaventure.
Forward Erasure Correction for QUIC loss recovery, 2023.

[4] P. Garrido, I. Sanchez, S. Ferlin, R. Aguero, and O. Alay.
rQUIC: Integrating FEC with QUIC for robust wireless communications.
In *GLOBECOM*, pages 1–7. IEEE, 2019.

[5] A. Gupta et al.
Improving HTTP/3 Quality of Experience with Incremental EPS, 2024.

[6] J. Herbots et al.
HTTP/3's Extensible Prioritization Scheme in the Wild.
In *IRTF-ANRW*, pages 1–7, 2024.

[7] A. Iyidogan, M. Bosk, and F. Rezabek.
Hierarchical Resource Sharing and Queuing in OMNeT++ and INET Framework.
In *OMNeT++ Community Summit*, 2022.

[8] N. Luangsomboon and J. Liebeherr.
HLS: A Packet Scheduler for Hierarchical Fairness.
In *2021 IEEE 29th International Conference on Network Protocols (ICNP)*, pages 1–11. IEEE.

Bibliography

[9] F. Michel and O. Bonaventure.
QUIRL: Flexible QUIC Loss Recovery for Low Latency Applications.
IEEE/ACM Transactions on Networking, 2024.

[10] F. Michel, A. Cohen, D. Malak, Q. De Coninck, M. Médard, and O. Bonaventure.
FIEC: Enhancing QUIC with application-tailored reliability mechanisms.
IEEE/ACM Transactions on Networking, 2022.

[11] F. Michel, Q. D. Coninck, and O. Bonaventure.
Adding Forward Erasure Correction to QUIC.
CoRR, abs/1809.04822, 2018.

[12] F. Michel, Q. De Coninck, and O. Bonaventure.
QUIC-FEC: Bringing the benefits of Forward Erasure Correction to QUIC.
In *2019 IFIP Networking Conference (IFIP Networking)*, pages 1–9. IEEE, 2019.

[13] D. Moskvitin et al.
Forward Erasure Correction for Short-Message Delay-Sensitive QUIC Connections.
Standard Draft, 2023.

[14] M. Nottingham and P.-H. Kamp.
Structured Field Values for HTTP.
RFC 8941, 2021.

[15] K. Oku and L. Pardue.
Extensible Prioritization Scheme for HTTP.
RFC 9218, 2022.

Bibliography

[16] D. Ran et al.
HSDBA: a hierarchical and scalable dynamic bandwidth allocation for programmable data planes.
Frontiers of Information Technology & Electronic Engineering, 25(10), 2024.

[17] P. Raussi et al.
Prioritizing protection communication in a 5G slice: Evaluating HTB traffic shaping and UL bitrate adaptation for enhanced reliability.
The Journal of Engineering, 2023(9), 2023.

[18] V. Roca et al.
Sliding Window Random Linear Code (RLC) Forward Erasure Correction (FEC) Schemes for QUIC.
Informational Draft, 2020.

[19] P. U. Tournoux et al.
On-the-fly erasure coding for real-time video applications.
IEEE Transactions on Multimedia, 13(4), 2011.

[20] A. Yu and T. A. Benson.
Dissecting performance of production QUIC.
In *Proceedings of the Web Conference 2021*, pages 1157–1168, 2021.

[21] Z. Zhang et al.
vpifo: Virtualized packet scheduler for programmable hierarchical scheduling in high-speed networks.
ACM SIGCOMM '24, 2024.