Merge Those Metrics : Towards Holistic (Protocol) Logging

Robin Marx
robin.marx@kuleuven.be
KULeuven
Diepenbeek - Belgium

Joris Herbots
joris.herbots@uhasselt.be
Hasselt University – tUL - EDM
Diepenbeek - Belgium

ABSTRACT

In this position paper for the IAB workshop on "Measuring Network Quality for End-Users", we observe that there is an apparent threefold split in how we currently log and analyze metrics for different protocol layers and the business logic running on top. We opine that this split makes it more challenging to answer some of the questions the workshop posited, as it makes metrics and behaviours at lower layers harder to correlate with higher layer Quality of Experience indicators. We motivate that a possible solution to this issue is to move towards a consolidated cross-layer and cross-vantage point logging approach that can provide a more holistic view of a system's behaviour that can then be analyzed with specialized tools. To kickstart discussion, we debate recent efforts in this direction and highlight several ensuing opportunities and potential practical challenges.

1 INTRODUCTION & MOTIVATION

In a modern networked application, many different protocols, implementations and business logic flows are collaborating to complete even a single end-to-end request. While conceptually it is useful to separate these aspects into individual layers (e.g., Layer 3 (L3) is network/IP, L4 transport/TCP, L7 application/HTTP, and L8 and above are higher level protocols (e.g., DASH, WebTransport) and application-specific business logic), it would be wrong to assume these layers can operate fully independently without impacting each other. As such, improvements on a lower layer (say L4 moving from TCP to QUIC) could evoke much smaller or indeed even opposite effects [4, 5, 9, 13, 22] on higher layer behaviour and, consequently, end-user Quality of Experience (QoE). Similarly, higher layers misusing lower layer features can easily lead to performance regressions [5, 12]. As the impacts of changes are typically implied through the occurrence of certain events (e.g., packet received) or measured in a variety of metrics (e.g., average throughput in the last x seconds) at the different layers, we can thus say there is **a clear benefit to correlating these events and metrics in a holistic view across protocols and higher level logic** [1, 3, 5, 14].

1.1 Three's a Crowd

As such, it is somewhat strange to observe that there is an apparent **threefold split in metric usage across (protocol) layers**, leading to separate approaches in storage/file formats, analysis tool usage and operator expertise niches, making it more challenging to actually perform the useful cross-layer correlation. Firstly, at Metric Level 1 (*M1*), L3 and L4 (and sometimes L1 and L2) metrics are often deduced from (in-network) packet captures (pcaps), storing the raw wire image or a direct derivative (e.g., flows). Secondly, at *M2*, L5 to L7/L8 metrics are typically logged together from inside software libraries or platforms (e.g., browsers) at the implementing endpoints. Thirdly, for *M3*, business and application-specific metrics are usually exfiltrated directly from the custom application code (e.g., using 3rd party analytics providers). We posit that this split is mostly **motivated by two core factors**.

Firstly, **implementation location**. L4 and below (L4-) is often implemented in the kernel, while L5-L7 is typically found in separate (re-usable) libraries or platforms in user space. While business logic also mostly resides in user space, it still tends to be separate from the higher-layer protocols it employs (e.g., JavaScript running in the browser). These separate codebases typically have well-defined interfaces, which sadly rarely allow (direct) lower-level metric exfiltration. This has even given way to an entire class of complex algorithms and protocols that aim to optimize QoE by using higher order metrics derived from (delayed) observations. For example, HTTP Adaptive Streaming uses its own estimates of RTT and guesstimates available bandwidth by averaging segment download times, rather than using more accurate transport-level ACK and packet loss signals directly. This seems suboptimal, as some approaches have shown that integrating L4 with L8+ more directly can have concrete benefits for video streaming QoE [1, 18]).

Secondly, especially the split between *M1* and *M2* is motivated by the fact that **L5+ is often encrypted** to improve user security and privacy (e.g., with TLS). As such, kernel implementations and especially in-network captures simply have no means by which to access higher-layer information.

In a way, this three-way split is logical and for a long time **has been "good enough" for most use cases**. For example, network administrators find sufficient information in L3/L4 captures to monitor network health, while product owners tend to only focus on highest-level metrics anyway. However, as the authors of this work have experienced first-hand in recent years during Web page loading performance research, there exists a whole class of issues for which this typical **lack of cross-layer correlation can lead to problems that are difficult to analyze and optimize** (e.g., developers misusing HTTP (preload) Resource Hints or HTTP Server Push [10], browsers employing worst-case stream prioritization schedulers [12, 22], researchers misconfiguring TCP stacks assuming defaults deliver optimal performance [23]). Similar stories and anecdotes can undoubtedly be found for other setups (e.g., TCP "Performance Enhancing Proxies" leading to unexpected end-to-end regressions).

This all is exacerbated by the fact that, due to the splits, some core information is typically lost. For example, *M1* packet captures do not contain important protocol state that is not reflected on the wire (e.g., congestion control logic) and other key transport metrics (e.g., latency, packet loss) are only inferred indirectly through sometimes imperfect methods [7]. Extracting this information from the kernel implementations has also long been non-trivial and frequently remains unattempted. In all, **it seems that cross-layer correlation is still rare and difficult, requiring significant investment when needed to solve cross-layer issues**.

1.2 We Need To Go Deeper

While this imperfect but workable setup has been tenable for a long time, the recent arrival of encrypted transport layer protocols (with QUIC being the prime example), seems to challenge the status quo, making metric acquirement and cross-layer correlation simultaneously easier and more difficult, depending on the use case. On one hand, QUIC's encryption of almost all its transport-level metadata makes it (nearly) **impossible to derive crucial network health information** (e.g., latencies, packet loss) from (in-network) packet captures. Storing encrypted pcaps would not scale due to the required storage capacity, while decrypting them post-hoc has inherent privacy and security issues, as this also unveils all L7+ user payload data. While proposals exist to add explicit signalling bits for these aspects in the QUIC packet header [8] or in a separate "path layer" [7], these have often failed to find implementation support from large deployments due to for example privacy reasons [20]. On the other hand, the fact that most QUIC implementations currently reside in user space and are typically co-located with HTTP/3 logic, means that, compared to kernel-based TCP stacks, it is **much easier to query QUIC metrics** and thus also to better correlate L4 behaviour with L7 impact and vice versa.

Especially at the *M1*/*M2* split, this has in the past few years led to somewhat of **a paradigm shift away from packet captures towards utilizing endpoint logging instead, especially for L4**. We ourselves have for example extensively explored this opportunity with the "qlog" project [14], which proposes a standardized schema for structured endpoint (protocol) event logging. This approach resolves scalability issues (endpoints only log the events deemed necessary, rather than the entire wire image) and alleviates privacy concerns (payload and sensitive fields are easily omitted or obfuscated). Even when just applied to L4 (QUIC) and L7 (HTTP/3), qlog has found widespread adoption [13] and in some cases has even replaced L4+ packet capture usage altogether (e.g., Facebook's mvfst stack does not have options to exfiltrate TLS keys for decryption of pcaps). It has also been shown to work at scale, with large deployments (e.g., protocol labs, Facebook [13]) logging billions of daily events. In combination with our cross-layer "qvis" tools and visualizations [14], this approach has indeed shown quite promising in detecting, analysing and resolving complex issues [14]. Finally, while originally primarily intended for discrete protocol event logging, qlog's flexibility allows for more comprehensive scenarios. For example, the spindump project [19] uses qlog to exfiltrate aggregated in-network measurements, while our own work on HTTP Adaptive Streaming [2] registers higher-layer ABR info such as buffer occupancy or playhead movement, and even user interactions with video controls alongside QUIC and H3 events.

This paradigm shift has however not just been QUIC/H3 specific. For example, Netflix's "tcp_log/black box logging" [15] adds a new custom option to the OpenBSD kernel to output augmented pcaps, which contain low-level congestion control state (initially mainly used to help debug BBR behaviour). The COP^2 project [21] utilizes advances in eBPF functionality to extract protocol KPIs (e.g., for TCP, MPTCP and DNS) and aggregates them with *M1* flow information via ipfix. We ourselves have used similar eBPF techniques to prototype qlog support for TCP as well [14]. Finally, very recently, Iurman et al. have discussed a new approach to cross-layer telemetry [3], combining low-level, hop-by-hop L3+ IOAM information encoded in IPv6 extension headers with high-level, end-to-end *M3* tracing information. This allows developers to for example attribute slow microservice request completion times to buffer occupancy issues/congestion at the router level, for which the authors also provide custom tools. While their system seems currently not to aggregate (much) information of the layers between L3 and L8+, these can conceptually be easily added in their general approach.

In conclusion, while most of the current cross-layer efforts focus on just a subset of the problem space, it is clear that

opportunities exist to combine these into a singular approach. Conceptually, we could move to a single storage format/representation and methodology that can combine and aggregate all cross-layer metrics/events, from L3 to L8+, thereby removing the threefold split and enabling much more powerful end-to-end and cross-layer analysis for complex issues. However, we can also see **several major (practical) challenges that must be overcome** for this to become a reality, even while we feel many existing systems can be re-purposed or extended. We discuss these aspects in more detail in the remainder of this text.

2 OPPORTUNITIES & CHALLENGES

In this Section, we discuss several opportunities and challenges inherent in **creating a methodology, format, workflow and tooling for truly unified cross-layer and cross-vantage point metric/event correlation in complex deployments**. We start small by considering per-vantage point aspects, extrapolate this to grouping logs across vantage points in a distributed system, then discuss how to process this holistic data in tooling and finally how to share information with 3rd parties.

2.1 Per-Vantage Point Metric Merging

Let us first consider **aggregating metrics from different subsystems running on an individual vantage point** (e.g., one client, server, load balancer, proxy, router). Ideally we would co-locate all metrics/events in one "merged" file/output stream, producing a single source of truth for the local stack. Additionally, the output should be consistent and similar across protocols and use cases, to enable the creation of re-usable tools that work across a variety of setups.

The first challenge to make it feasible to practically interpret this merged log and correlate events in tooling, is the need to devise a **coherent overarching schema and output format to describe the often very divergent semantic concepts in a consistent manner**. For example, output containing both raw packet captures and JSON-based analytics payloads would not be optimal. With qlog, we have tried to make this easier by employing JSON, a simple text-based format, and by keeping high-level event definitions as simple as possible; each event is a three-tuple of timestamp, event type and event-specific data. While this is easily extendable and allows flexibility where needed, it still poses many subtle issues when defining event types and data layout [11] (e.g., do we use "packet_sent" events to describe the wire image or a higher-layer "packets_acknowledged" event to describe the impact on internal state or both?). Additionally, there is the question of making the same schema/concepts work across different protocols and applications. Even when just comparing TCP and QUIC, there are subtle differences in the definition of connections, packets and payloads, preventing the direct re-use of event definitions across protocols. **While full coherence, consistency and equivalence across use cases is probably not needed, some form of schema consolidation is likely necessary**, requiring concerted efforts of domain experts.

A second challenge is that it is practically **difficult to create such a merged log from inside the subsystem implementations directly**. On one hand, this is due to the lack of infrastructure. For example, there is no OS-level API to coordinate the merging of events, nor are there typically interfaces at implementation boundaries (e.g., browsers are unlikely to expose JavaScript APIs to send L8+ metrics down for output alongside browser logs). On the other hand, there are performance considerations, where not all layers of the system might want to output events at all times, or not in the singular output format. As such, it is much more likely that the merged output format will not be a "capture format" that is used by implementations directly, but rather a "tool interoperability format", that is generated by the post-hoc merging of several individual logs from different subsystems (e.g., a qlog for a TCP+TLS+HTTP/2+WebApp stack could be created by converting and merging a pcap, a browser log and JavaScript output). In all, we see this as an opportunity rather than a problem: this allows us to **build on top of existing systems, incrementally adding new converters to extract information from available sources where most useful**. Alongside this, subsystems can be updated piecemeal to output in the final format directly, or make the conversion and metric extraction easier (as we have seen happen with qlog for QUIC, and eBPF/IOAM in other work discussed in Section 1.2).

A third challenge that derives from the second is that we need a way to **synchronize logs from different subsystems to ensure a proper order of events in the merged output and thus allow the correct derivation of cross-layer event causality**. Conceptually, this should be feasible using timestamps, however minor differences in precision, added delay by logging subsystems, events happening at the same exact time (limited by timestamp precision) and potentially very high-speed transfers could challenge the robustness of this approach. As such, other synchronization primitives are likely necessary. For some cases, this might be derived from information available in adjacent subsystems (e.g., packet/frame numbers, request IDs/URLs), while in others it could help to semantically delineate separate phases of a process (e.g., start/finish of a request might correlate to connection start/finish). In practice however, **we expect that this will be one of the hardest problems to solve**, in some cases requiring manual human correction.

In conclusion, aiming to aggregate all layers into a single semantically consistent output format from the start seems

an unattainable utopia. However, starting from existing best practices and implementations, it is possible to convert and merge subsystem output into a more holistic view, **allowing us to incrementally build towards the final desired outcome**.

2.2 Cross-Vantage Point Log Grouping

Assuming we have been able to (partially) solve the challenges in the previous Section, and can obtain merged logs from individual vantage points, let us now consider **grouping merged logs from across multiple vantage points in distributed deployments** (e.g., across load balancers, proxies, edge nodes, routers, and origins). Ideally, we would be able to group all logs belonging to a single high-level business logic concept (e.g., a microservice call, web page load) so they can be studied together in a single tool to provide an end-to-end view of the system.

The first challenge is to **identify logs that conceptually belong together**. This is similar to the synchronization problem discussed in Section 2.1, but made more manageable by the fact that we start from merged, cross-layer logs and that thus some common parameters (e.g., request ID, QUIC Connection IDs) should be easier to align. This is the concept used in many common "tracing" setups (e.g., splunk, dynatrace, opentelemetry), that typically mainly track *M3* information across vantage points. This setup was also shown to be extendable to lower metric levels by for example Iurman et al. [3], who repeat the L8+ request ID in individual packets by using IPv6 extension headers. One challenge however is that not all these signals are necessarily available at all individual vantage points. Especially some network intermediaries (e.g., routers, load balancers) might not act upon higher level information, being limited to L2-L4 logic. One example is the Spindump project [19], which acts as an in-network metric collector mainly at the L3/L4 layer (e.g., using the QUIC spinbit). To be able to group logs of such an entity together with other vantage points, lower-level data might be needed (e.g., IP addresses), which in turn might be challenging due to privacy concerns. Relatedly, while some signals (like the request ID) might be end to end, others will be hop-by-hop (e.g., IP addresses and Connection IDs as connections are terminated, load-balanced, (re-)multiplexed, etc. at intermediary nodes). This might require pairwise comparisons between logs to discover the end-to-end chain, increasing complexity. Still, we feel that, especially within managed networks, grouping of logs at individual vantage points by a higher-level construct should be practical to achieve, though the methods employed might be highly deployment specific (e.g., relying on per-packet tagging vs heuristic grouping on higher layers).

A secondary challenge is the need for **logistical support to store and query logs from individual vantage points** so they can be easily grouped and processed. For this, we assume that sufficiently large deployments already have such setups in place that can be augmented or reused for this purpose. For example, Facebook's qlog deployment enters each event as an individual entry in an existing log-processing relational database, tagged with the Connection ID (CID) of the QUIC connection the event occurred on [14]. Events can then later easily be (SQL) queried by individual data fields and grouped by CID. For grouping across vantage points in the presence of connection termination, custom deployment-specific events can be used to indicate how an incoming CID relates to an outgoing one and vice versa. Similar setups can for example be found in IPFIX systems, where in-network measurement entities send data to one or more "collectors" for storage. The COP^2 project [21] proposes to re-use this architecture to also capture TCP statistics collected at individual vantage points. Spindump [19] also uses a comparable methodology.

Overall, we feel that the challenges in this Section should be quite feasible to resolve, albeit often in a very deployment-specific fashion. As such, a generalized methodology needs to be flexible and to allow for various options for syncing, grouping and correlating logs across vantage points.

2.3 Tooling and Analysis

Once we are able to obtain and group (partially) cross-layer logs from one or more vantage points, we need **methods to efficiently categorize and analyze the log contents**, both automatically and manually.

Firstly, especially for large-scale deployments, we require **data mining algorithms to automatically categorize and fingerprint logs** (e.g., normal behaviour, existing anomaly present, unexpected behaviour) and to filter out those logs that would benefit from further analysis by human experts. For this task, existing methods again exist (e.g., utilizing manually tuned heuristics, advanced protocol models or machine learning systems), but these seem to be split in the same way as the logs they aim to process. We feel that aptly processing merged logs would require both a combination of these existing systems with new methodologies (potentially extrapolated from established techniques) to make optimal use of the co-location of the cross-layer information. Similar things can be said for automated root-cause analysis or detailed anomaly detection algorithms. The availability of such methods can also allow better evaluation of the cross-layer impact of a given change through automated A/B testing.

Secondly, once anomalous logs have been identified that require manual analysis, operators should be able to load them into **powerful tools and visualizations that allow**

the exploration of the merged logs in their cross-layer depth and cross-vantage point breadth. Additionally, tools should integrate some of the automated methods discussed before to help identify problematic areas of the log, as well as potentially easily contrast the unexpected behaviour with its "normal" counterparts. The creation of such tools is a major challenge, as not all data can be displayed in full detail at all times, requiring various levels of abstraction of visualization and analysis support. With our qvis toolsuite [14] we have made an initial effort in this direction, providing ways to visualize qlog information for multiple complementary aspects (e.g., congestion control, stream multiplexing, packetization) and to track a single QUIC connection across separate vantage points in a single sequence diagram tool. Our experience has made clear that creating this type of tool will require a major engineering effort. Luckily, we feel these new tools will be (initially) mostly necessary for the discussed ethereal cross-layer issues that are difficult to analyze using existing tooling.

Thirdly, we see an opportunity to **advance (deterministic) network emulation and trace playback systems**. Especially in research settings, network emulation/simulation is often done using higher-level models that not always accurately reflect realistic network conditions or do not allow consistently emulating the (exact) same characteristics (e.g., the use of linux tc/netem utilities). While more realistic approaches exist (e.g., deterministic NS-3 mode, Mahi-Mahi's per-packet trace playback [16]), these systems can sometimes be hard to use to mimic existing deployed setups. In contrast, one or more merged logs containing full internal state information (which packets were detected lost when, congestion control behaviour, buffer draining rates, etc.) can in theory be used to emulate not only the exact network conditions prevalent at the time of the anomalous behaviour (as measured directly in the real-world deployment), but also the operation of the exact application under consideration. As such, (partial) merged logs could be employed to recreate exact conditions to test a fix to a bug identified by analyzing that same log. While challenges exist with this method and it is not optimal for use in all scenarios, we feel it is nevertheless an exciting new direction to explore.

In conclusion, the envisioned availability of cross-layer and cross-vantage point logs allows for advanced analysis opportunities, but requires additional efforts for interpretation and tooling development.

2.4 Data Sharing

Besides analyzing the collected logs, we also see opportunities for **sharing them with outside parties, both live for operators of intermediate networks and as curated datasets for researchers**.

Firstly, as discussed in Section 1.2, the arrival of encrypted L4 protocols like **QUIC provides major challenges for network operators**, who have traditionally depended on observable TCP metadata to measure network health (e.g., latency, packet loss), to ensure a layer of security (e.g., firewalls) or even to attempt to improve performance (e.g., Performance Enhancing Proxies). With the absence of clear and/or broadly deployed replacement options in QUIC (e.g., undersupported spinbit), alternative methods must be considered. One such approach when extracting metrics from vantage points is to propagate them by tagging individual packets [3, 7]. However, this method mainly seems workable within the boundaries of a single deployment, as the tagging needs to be/is ideally enabled/disabled at the edges, and can introduce significant overhead. An alternative option discussed in a recent paper by Krämer et al. [6] is the use of a non-transparent/cooperative proxy using an outer tunnel between the client and the proxy to communicate metrics while leaving the inner end-to-end connection encrypted. While powerful, this approach does not seem universally deployable for our use case, due to the requirement of active client participation. However, we envision a similar **cooperative setup, but between the network operator and the server/application deployment instead**. Assuming the deployment already tracks vantage point logs (as discussed in Section 2.2), it would be possible for outside parties to request these logs (or at least their salient aggregated metrics) when needed for troubleshooting or security checks. Such a mechanism was proposed initially by Kazuho Oku [17], utilizing a special METRICS packet, where on-path devices actively request information from the endpoints. While this type of setup would not be optimal for many use cases, and requires additional logistical provisions as well as assurances for access control and privacy, we still feel this can be a partial solution to QUIC's observability puzzle, especially in tightly co-located deployments (e.g., CDNs placing servers in ISP networks).

Secondly, merged logs that are aggregated and stored at scale (e.g., within a real-world deployment at a large company) have **the opportunity to become valuable datasets, uniquely suited to the study of modern internet dynamics**. Currently, for internet measurement studies, academics seem to rely primarily on two types of datasets. Firstly, passive measurements, often in the form of packet captures (or more high-level as aggregated flows), are often taken at a limited amount of network vantage points (e.g., a single IXP, a university network). While these datasets are usually somewhat representative of real-world traffic, for privacy reasons they tend to be limited to only *M1* level data. Secondly, active measurement datasets can be more geographically distributed (e.g., when utilizing RIPE Atlas or similar systems) and can provide cross-layer insights. However, they

can be somewhat less representative of realistic traffic as they are either generated utilizing (micro)benchmark applications (e.g., using speedtests, ping), not run on constrained devices (e.g., desktops vs low-end mobile phones), or not executed from inside outlier networks (e.g., university networks rather than the cheapest cellular option). In contrast, large companies with worldwide deployments (e.g., Facebook, Google, CDNs), have access to **logs for realistic applications, last-mile networks and representative user devices**. If these datasets can be properly sanitized for privacy, security (and potentially intellectual property) related constraints, we feel they could revolutionize protocol and internet measurement studies, answering currently somewhat elusive questions (e.g., packet loss distributions/causes, practical MTU limits, latency and jitter outliers and their effects) and provide insight into cross-layer effects in some of the most optimized applications known.

As the critical reader might note, companies could already share these datasets today without our proposed approach. While the cynical reader might pose they refrain to do so mainly out of competitive reasons, the naive authors of this work feel the practical and privacy-related aspects of current logging methods (e.g., packet captures and their size) might also play an important part in their decisions. As such, the move to consolidated vantage point logs that can be more easily reduced to salient (protocol) information, could provide an additional incentive to rethink their stance, as companies also stand to gain from additional insights found by academics [13]. We have received positive informal early signals along these lines from at least Facebook and Cloudflare with regards to qlog data for QUIC and HTTP/3. While this might feel like the most unrealistic outlook in this text to some, we posit that **it is one of the most crucial aspects needed to answer posed questions** like "what is the latency under typical working conditions?" and "how reliable is the connectivity across longer time periods?", as these can only really be answered by considering a sufficiently large, realistic dataset (incorporating real-world outliers) to which currently only large companies seem to have access.

In conclusion, while sharing merged logging data might raise almost as many problems as it can solve, we encourage readers to discuss these concepts for their merits.

3 CONCLUSION

In this text, we have discussed a conceptual move away from an apparent three-way split in protocol event and metric logging, towards a more consolidated cross-layer and cross-vantage point approach. While this method is mostly useful and necessary for some more esoteric issues, and while the split isn't always as hard or disruptive as our text might have conveyed, we still feel there are enough benefits to be found to discuss and start this effort, at least for a subset of use cases. For example, in the case of QUIC and HTTP/3 protocol debugging, this approach has already shown to be exceedingly powerful.

Our belief is further motivated by the ability to re-use most of the existing systems and to build the new approach incrementally on top, by for example converting and then merging existing logs from individual metric levels into a single cross-layer view. Even a partial execution can be powerful, as again shown by our results in the qlog and qvis projects. Most of the extra needed work, such as defining a high-level semantic format and providing tools, is focused at fully utilizing the power of the consolidated approach. This work has outlined some of the challenges inherent therein, but also highlighted some of the potential gains that can be expected from such an effort.

Finally, we feel strongly that to answer some of the key questions in internet research in a nuanced way, large companies should be willing to share datasets, detailing measured (low-level behaviour) in worldwide networks. This is simplified implicitly by our proposed approach, which allows the provision of powerful data in a privacy-aware fashion.

We hope this text can serve as a solid basis for continued discussion on this topic.

ACKNOWLEDGMENTS

Joris Herbots' involvement in this work was supported by the Special Research Fund (BOF) of Hasselt University, with reference number BOF19OWB07. The authors thank Mike Vandersanden for his feedback on earlier versions of this text.

REFERENCES

[1] Sadjad Fouladi, John Emmons, Emre Orbay, Catherine Wu, Riad S. Wahby, and Keith Winstein. 2018. Salsify: Low-Latency Network Video through Tighter Integration between a Video Codec and a Transport Protocol. In *15th USENIX Symposium on Networked Systems Design and Implementation (NSDI 18)*. USENIX Association, Renton, WA, 267–282. https://www.usenix.org/conference/nsdi18/presentation/fouladi

[2] Joris Herbots, Maarten Wijnants, Wim Lamotte, and Peter Quax. 2020. Cross-Layer Metrics Sharing for QUICker Video Streaming. In *Proceedings of the 16th International Conference on Emerging Networking EXperiments and Technologies* (Barcelona, Spain) *(CoNEXT '20)*. Association for Computing Machinery, New York, NY, USA, 542–543. https://doi.org/10.1145/3386367.3431901

[3] Justin Iurman, Frank Brockners, and Benoit Donnet. 2021. Towards Cross-Layer Telemetry. In *Proceedings of the Applied Networking Research Workshop* (Virtual Event, USA) *(ANRW '21)*. Association for Computing Machinery, New York, NY, USA, 15–21. https://doi.org/10.1145/3472305.3472313

[4] Matt Joras. 2021. Video Ingest over QUIC. https://mailarchive.ietf.org/arch/msg/quic/rNHbTTWzrkXJ91ryFXbz_Gn3p3w/.

[5] Matt Joras and Yang Chi. 2021. How Facebook Is Bringing QUIC to Billions. https://plus.qconferences.com/plus2021/presentation/how-facebook-bringing-quic-billions.

[6] Zsolt Krämer, Mirja Kühlewind, Marcus Ihlar, and Attila Mihály. 2021. Cooperative Performance Enhancement Using QUIC Tunneling in 5G Cellular Networks. In *Proceedings of the Applied Networking Research Workshop* (Virtual Event, USA) *(ANRW '21)*. Association for Computing Machinery, New York, NY, USA, 49–51. https://doi.org/10.1145/3472305.3472320

[7] Mirja Kühlewind, Tobias Bühler, Brian Trammell, Stephan Neuhaus, Roman Müntener, and Gorry Fairhurst. 2017. A path layer for the Internet: Enabling network operations on encrypted protocols. In *2017 13th International Conference on Network and Service Management (CNSM)*. IEEE, 1–9.

[8] Ike Kunze, Klaus Wehrle, and Jan Rüth. 2021. L, Q, R, and T: Which Spin Bit Cousin is Here to Stay?. In *Proceedings of the Applied Networking Research Workshop* (Virtual Event, USA) *(ANRW '21)*. Association for Computing Machinery, New York, NY, USA, 22–28. https://doi.org/10.1145/3472305.3472319

[9] Adam Langley, Alistair Riddoch, Alyssa Wilk, Antonio Vicente, Charles Krasic, Dan Zhang, Fan Yang, Fedor Kouranov, Ian Swett, Janardhan Iyengar, Jeff Bailey, Jeremy Dorfman, Jim Roskind, Joanna Kulik, Patrik Westin, Raman Tenneti, Robbie Shade, Ryan Hamilton, Victor Vasiliev, Wan-Teh Chang, and Zhongyi Shi. 2017. The QUIC Transport Protocol: Design and Internet-Scale Deployment. In *Proceedings of the Conference of the ACM Special Interest Group on Data Communication* (Los Angeles, CA, USA) *(SIGCOMM '17)*. Association for Computing Machinery, New York, NY, USA, 183–196. https://doi.org/10.1145/3098822.3098842

[10] Brad Lassey. 2020. Intent to Remove: HTTP/2 and gQUIC server push. https://groups.google.com/a/chromium.org/g/blink-dev/c/K3rYLvmQUBY.

[11] Robin Marx. 2021. qlog: the philosophical update. https://github.com/quicwg/wg-materials/blob/main/ietf111/qlog.pdf.

[12] Robin Marx., Tom De Decker., Peter Quax., and Wim Lamotte. 2019. Of the Utmost Importance: Resource Prioritization in HTTP/3 over QUIC. In *Proceedings of the 15th International Conference on Web Information Systems and Technologies (WEBIST 2019)*. INSTICC, SciTePress, 130–143. https://doi.org/10.5220/0008191701300143

[13] Robin Marx, Joris Herbots, Wim Lamotte, and Peter Quax. 2020. Same Standards, Different Decisions: A Study of QUIC and HTTP/3 Implementation Diversity. In *Proceedings of the Workshop on the Evolution, Performance, and Interoperability of QUIC* (Virtual Event, USA) *(EPIQ '20)*. Association for Computing Machinery, New York, NY, USA, 14–20. https://doi.org/10.1145/3405796.3405828

[14] Robin Marx, Maxime Piraux, Peter Quax, and Wim Lamotte. 2020. Debugging QUIC and HTTP/3 with Qlog and Qvis. In *Proceedings of the Applied Networking Research Workshop* (Virtual Event, Spain) *(ANRW '20)*. Association for Computing Machinery, New York, NY, USA, 58–66. https://doi.org/10.1145/3404868.3406663

[15] Netflix. 2020. tcplog_dumper. https://github.com/Netflix/tcplog_dumper.

[16] Ravi Netravali, Anirudh Sivaraman, Somak Das, Ameesh Goyal, Keith Winstein, James Mickens, and Hari Balakrishnan. 2015. Mahimahi: Accurate Record-and-Replay for HTTP.. In *USENIX Annual Technical Conference*. 417–429.

[17] Kazuho Oku. 2018. *Performance Metrics Subprotocol for QUIC*. Internet-Draft draft-kazuho-quic-perf-metrics-00. IETF Secretariat. https://datatracker.ietf.org/doc/html/draft-kazuho-quic-perf-metrics-00

[18] Mirko Palmer, Thorben Krüger, Balakrishnan Chandrasekaran, and Anja Feldmann. 2018. The QUIC Fix for Optimal Video Streaming. In *Proceedings of the Workshop on the Evolution, Performance, and Interoperability of QUIC* (Heraklion, Greece) *(EPIQ'18)*. Association for Computing Machinery, New York, NY, USA, 43–49. https://doi.org/10.1145/3284850.3284857

[19] Ericsson Research. 2021. Spindump: a latency measurement tool. https://github.com/EricssonResearch/spindump.

[20] SpinbitSupport. 2018. IETF 103 - spin bit discussion. https://github.com/quicwg/wg-materials/blob/master/ietf103/minutes.md.

[21] Olivier Tilmans and Olivier Bonaventure. 2019. COP2: Continuously Observing Protocol Performance. *arXiv preprint arXiv:1902.04280* (2019).

[22] Maarten Wijnants, Robin Marx, Peter Quax, and Wim Lamotte. 2018. HTTP/2 Prioritization and Its Impact on Web Performance. In *Proceedings of the 2018 World Wide Web Conference* (Lyon, France) *(WWW '18)*. ACM, 1755–1764. https://doi.org/10.1145/3178876.3186181

[23] Konrad Wolsing, Jan Rüth, Klaus Wehrle, and Oliver Hohlfeld. 2019. A Performance Perspective on Web Optimized Protocol Stacks: TCP+TLS+HTTP/2 vs. QUIC. In *Proceedings of the Applied Networking Research Workshop* (Montreal, Quebec, Canada) *(ANRW '19)*. Association for Computing Machinery, New York, NY, USA, 1–7. https://doi.org/10.1145/3340301.3341123