SCONE
Internet-Draft
Intended status: Informational
Expires: 9 November 2025

W. Eddy
A. Tiwari
A. Frindell
Meta

8 May 2025

APIs To Expose SCONE Metadata to Applications
draft-eddy-scone-api-01

Abstract

This document describes API considerations to provide applications with network-supplied information about acceptable network flow rates. Since this information is expected to be signalled from the network within the stack below the application using SCONE protocol signalling, it needs to be made accessible to applications in order for them to pick proper video rates, or to otherwise confine the application behavior within network-defined limits.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at https://datatracker.ietf.org/drafts/current/.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on 9 November 2025.

Copyright Notice

Table of Contents

1. SCONE Background and Introduction

Video traffic is already 70% of all traffic on the Internet and is expected to grow to 80% by 2028. New formats like short form videos have seen tremendous growth in recent years. Both in developed and emerging markets video traffic forms 50-80% of traffic on mobile networks. These growth trends are likely to increase with new populations coming online on mobile-first markets and the observation that unlike text content, video content consumption is not being limited by literacy barriers. On the other hand, the electromagnetic spectrum is a limited resource. In order to ensure that mobile networks continue functioning in a healthy state despite this incredible growth, communication service providers (CSPs) will be required to make infrastructure investments such as more licensed spectrum, cell densification, massive MIMO etc. In order to flatten the rate of growth, CSPs in several markets attempt to identify and

throttle video traffic based on user data plans. There are several problems with this kind of throttling:

1. CSPs can not explicitly measure the effect that throttling has on the end user's quality of experience (QoE) making this an open loop approach.

2. Traffic detection and throttling for every flow is compute intensive for CSPs. With distributed UPF (user plane function) in 5G mobile networks more nodes in CSP network may need to support traffic detection and throttling. Traffic detection can have inaccuracies and these inaccuracies are expected to increase as the content delivery industry moves towards end-2-end encryption like TLS 1.3 and encrypted client hello (ECH).

3. The unpredictable and non-transparent behavior of traffic throttlers used by CSPs confuse the bandwidth estimation and congestion control protocols being used within end-2-end video delivery sessions between content server and client. This results in poor quality of experience (QoE) for the end user.

4. Content and Application Providers (CAPs) are designing algorithms to detect the presence of such traffic throttlers to counter their detrimental effects. These algorithms have their own inaccuracies in detection and add compute resources on the CAP side.

An alternative approach is for CAPs to self-adapt the traffic corresponding to video flows. Since CAPs control the client and server endpoints and can measure end user QoE, they are in a better position to do this self-adaptation in a close loop manner. This alternative approach has already been proven to improve user QoE in production deployments [YouTube].

For this alternative approach to work a standardized secure on-path network interface is required which will enable CSP controlled network elements to signal the desired traffic profile characteristics to the CAP client/server endpoints. The Standard Communication with Network Elements (SCONE) protocol (previously known as SADCDN and SCONEPRO) is an IETF working group motivated by this alternate approach.

1.1. SCONE API Motivations

The general problem statement for SCONE is described in the video optimization requirements document [I-D.joras-scone-video-optimization-requirements], including the shaping or throttling that CSPs perform. While this problem currently has especially large impact on a few large content providers, solutions for SCONE are generally applicable to any applications that use QUIC [RFC9000] and are subject to throttling within CSP networks.

General use of SCONE metadata for any applications can be facilitated via an open Application Programming Interface (API) that could be implemented in appropriate QUIC stacks, web browsers, or other libraries.

There are two aspects to consider for an API:

1. How will applications learn about network information that is discovered by SCONE lower in the stack? This is a primary consideration in this document.

2. How will applications signal their type (e.g. video streaming) or other relevant properties to the stack, to indicate that they are SCONE-capable? This is a secondary consideration in this document, because currently networks that perform throttling have built-in methods to implicitly determine the appropriate flows to throttle.

The SCONE metadata may be available at different places in the protocol stack implementation spanning operating system, QUIC library, browser, and application code. This document tries to initially make no assumptions about how the SCONE signalling works, and so considers possibilities to integrate the metadata into APIs provided from OSes, QUIC libraries, web browsers, etc. There are open questions at the moment about SCONE signaling via on-path devices, what type of information is conveyed, and how it is represented. The API capabilities may be limited by the protocol decisions, and realistic concerns about signaling across network domain boundaries, etc.

During early SCONE discussions, there have been suggestions that the API might take inspiration from Explicit Congestion Notification (ECN), as ECN also exposes information from the network (congestion experienced codepoints) to end hosts, and the design contends with potential for abuse, crosses network domain boundaries, and has other desirable properties. Some key differences from ECN in usage relavent to a SCONE API have been pointed out:

1) ECN information is consumed either by transport protocols (e.g. TCP, MPTCP, and SCTP) or congestion control algorithms operating above e.g. UDP or UDP-Lite [RFC8303] [RFC8304], but typically below an application. For instance, within QUIC stacks used for video streaming, ECN can be consumed by the QUIC congestion control, but is not exposed to the application.

2) The exposure of SCONE metadata is intended to be at the level of data flows (e.g. to aid application decisions about what media quality to fetch), whereas ECN is consumed at the level of packets (within an individual flow).

While ECN is not a solution for SCONE [I-D.tomar-scone-ecn], it is productive to consider as an example based on similarities, including:

* Signaling is coming from the network, and may cross different network domains.

* Signaling points can also drop packets, and the signaling participation is indtended to avoid excessive packet drops.

The purpose of this document is only to demonstrate that general API exposure of the SCONE metadata is achievable without prescribing a specific API solution. It is envisioned that one or more specific API solutions will be defined either through IETF or W3C, to correspond with the SCONE protocol specification. At least in its current form, this document is not intended to be published as an RFC.

2. Conventions and Definitions

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

3. Application Stack Designs

There are a variety of different application stack designs that are relevant. The main assumption for SCONE in general is that QUIC is used.

Applications could, for instance, (1) include their own QUIC implementation, (2) use QUIC directly through a linked software library, or (3) run within a web browser, using QUIC more indirectly via browser APIs.

Specific protocol solutions for SCONE are being defined by the working group. In general, the SCONE network rate-limiting information could be discovered by an end-system in several different ways; potential network signaling approaches are described in other documents (e.g. [I-D.ihlar-scone-masque-mediabitrate] or [I-D.thoji-scone-trone-protocol]). The SCONE working group solutions currently focus on signaling ia the QUIC stack, with the information inserted by an on-path SCONE network element.

Other approaches that do not seem to be as actively pursued at the moment are: (1) signaling via other IP or transport methods below QUIC (e.g. IP extension headers, UDP options, etc.) that might be inserted on the path, and (2) signaling via the OS, with the information coming in network advertisements separate from the transport connection (e.g. via Router Advertisements or DHCP). Therefore, OS-provided APIs and socket API extensions are not considered further in this document, since signaling is assumed to be implemented using QUIC..

It is important to note that QUIC library APIs are not standardized; they differ between common QUIC libraries, and so this document only suggests in a general sense of how the QUIC stack should convey this information to applications.

4. Potential SCONE Metadata Provided By An API

SCONE is chartered to provide "throughput advice". Some SCONE protocol solutions are currently providing a "rate signal" that represents a maximum bitrate. Work is also proposed defining a Video Session Data Rate (VSDR) metric [I-D.druta-scone-video-session-data-rate].

In order to define an API, the set of data being conveyed and its exact semantics need to be further worked on. For instance, a rate alone may not provide sufficient throughput advice without additional characterization of the averaging window, burst tolerance, or other parameters that may be of practical concern to an implementation.

The remainder of this document considers API extensions in general to provide SCONE data, whether it be a single data rate, or more comprehensive characterization.

5. Potential Browser API Extensions

For browser applications, there are multiple different browser APIs that might be extended to include SCONE metadata; notably including:

* W3C Network Information

* WebTransport using QUIC

* HTTP Live Streaming (HLS) or Dynamic Adaptive Streaming over HTTP (DASH) client libraries

* Any Javascript HTTP requests that directly or indirectly use HTTP/3.

In either of these cases, the corresponding W3C API definitions are the proper place for actual definition of API extensions, and this document is merely exploring possibilities.

The exploration is primarily around the ability to convey SCONE signaling information that is discovered from the network path up to applications. In addition, to indicate an application's desire to use SCONE signaling in the first place, some small API extension is also be required, unless relying totally on the underlying stack or network to infer which flows should be receiving SCONE treatment (e.g. as networks already infer which flows to throttle).

5.1. Potential Network Information API Inclusion

The W3C Network Information API [W3C-NetInfo] is supported to some extent by several, but not all, common web browsers today. It provides the possibility for an appliction to determine what underlying connection types or "effective" connection types (e.g. cellular. bluetooth, etc.) may be in use, with a corresponding set of performance parameter estimates including:

* Round Trip Time (RTT) in milliseconds providing a delay estimate.

* downlink in megabits per second providing an effective bandwidth estimate based on recently observed application layer throughput or properties of the underlying connectivity technology.

* downlinkMax in megabits per second representing an upper bound on the downlink speed of the first network hop, based on the underlying connectivity technology.

The downlink and downlinkMax could be leveraged as places to put the SCONE-discovered rate limit for an application, since anything greater than the SCONE-discovered rate would not be expected to be usable for the application.

Alternatively, another field could be added to the NetworkInformation interface in order to specifically provide the SCONE metadata.

In any case, a good property of the Network Information API is that an application can hook event handlers to be notified of changes, so that if there are limits that kick-in or are lifted midway into an application's lifetime (e.g. due to some mobility, etc.), the application will be able to be easily notified and adapt.

5.2. Potential WebTrans API Inclusion

In the future, WebTransport (WEBTRANS) might be used by SCONE's targeted types of applications, such as browser-based adaptive streaming. The IETF WEBTRANS working group is liasing with W3C as the IETF defines the protocol specification, whereas the W3C defines the API to use it. This case is similar to the IETF RTCWEB and W3C WebRTC WG coordination in the past. The same model of collaboration between IETF and W3C should work for SCONE metadata, and the information provided could be discussed with the WEBTRANS WG in the IETF and notified to the W3C later, either through common participation and/or formal liason statement.

The existing WebTrans API definition from W3C includes a "getStats()" method, that is defned in order to asynchronously gather and report statistics for a WebTransport underlying connection. It returns a WebTransportConnectionStats object that is defined as a dictionary, including a number of items such as:

* bytesSent
* packetsSent
* bytesLost
* packetsLost
* bytesReceived
* packetsReceived
* smoothedRtt
* rttVariation
* minRtt
* WebTransportDatagramStats datagrams
* estimatedSendRate

The estimatedSendRate is an unsigned long long, in bits per second, calculated by the congestion control algorithm in the user agent. This would be in the "upstream" direction to a CSP, though, rather than the "downstream" from a CSP, so is not useful to a client application in receiving indication of a downstream throttling rate from the network.

Since other measurements are already included and the WebTransportConnectionStats is a dictionary, it seems natural to extend it to include additional optional fields, such as an allowed media rate, or other types of fields providing the application information that the underlying host or stack have discovered about the presence of throttling or explicit signaling of allowed media rate on a path.

Such extensions might be including at a "MAY" level of conformance statement (rather than "SHALL" as used by all of the currently-defined information elements), as the allowed media rate will not be universally present or even useful for all WebTransport applications. Alternatively, it could be set to a "null" value similar to how the estimatedSendRate is sent when it is unknown by the user agent.

5.3. Potential HLS/DASH Support

Client libraries for HLS and DASH will use the underlying Javascript APIs or other underlying APIs, and might rely on them for SCONE metadata support, as discussed in the next subsection.

5.4. Other JavaScript API Options

Typical HTTP adaptive streaming applications using existing browser API options would be ideal to support as directly as possible. There are different ways to transfer HTTP/3 data provided to JavaScript applications that might be applicable.

For instance, things like jQuery.ajax() or the "Fetch" API may be used. In this case, there is little or no information about the network path state provided, though there are either jqXHR or Response objects returned that (for instance) allow HTTP headers to be conveyed, and in both cases could be extended to include SCONE metadata. These are returned at the completion of the HTTP transfer, however. It might be more difficult to support more dynamic updates such as providing the metadata to an application mid-transfer so that an application might quickly switch to other media rates for future video segments being pre-fetched.

6. Potential QUIC API Inclusion

While there are no standard QUIC APIs, and there are multiple different styles in use, many QUIC implementations include objects in the API that represent the QUIC connections directly, and allow setting callbacks for connection-related events, or allow direct querying of connection state. SCONE metadata could either be supported as a type of callback event, triggered when the metadata is received, or it could be included within other connection state in a polled or interrogated data structure.

Other QUIC implementations may leave I/O and management of sockets or other aspects to the application, external to the QUIC library. In this case:

1. If the SCONE metadata is visible as part of the QUIC connection, then it could be provided through the QUIC implementation’s API.

2. If the SCONE metadata is visible to the OS or as part of a socket API, it could be provided to the application via the underlying OS or socket abstraction libraries used by applications.

Regarding identification of the application flow type, options for a QUIC API may include adding "SCONE-capable" type of flag or an optional flow-type tag that can be set by applications. Compared to the complexity of existing QUIC APIs, these could be small additions.

6.1. Potential MoQ API Extension

While Media over QUIC (MoQ) is being defined, it is intended for media streaming over QUIC, which might be applicable to SCONE, in case adaptive rate streams are detected and throttled by CSPs. As yet, there is no standard MoQ API, an MoQ session is currently scoped either to a QUIC connection or a WebTransport session, so it should not be difficult to expose information learned by either transport stack to MoQ applications. Since MoQ applications are media flows, it may be very simple for an application flow type to be conveyed or inferred via an eventual MoQ API..

7. Security Considerations

General SCONE security considerations are discussed in the other documents covering specific network-to-host signaling methods. Privacy concerns have also been discussed in [I-D.tomar-scone-privacy].

Existing APIs that expose information about the network path to applications have documented security considerations, especially with regard to user privacy. For instance, there may be concerns that such information can be used to assist in fingerprinting users, defeating anonymization, or otherwise exposing more information about the user to the application than the user might explicitly consent to. Such concerns have been document, for example, with regard to the Network Information API.

By providing additional information about throttling rate limits within the path, SCONE could increase the amount of information availble on top of that provided by the current APIs. For instance, if the rate information is very fine-grained, it could be useful in fingerprinting.

Generally, however, the CSP throttling information is currently very coarse grained, as it is used today. Additionally, the application providers authenticate their users, and their is not an expectation of anonymity in popular platforms today.

Beyond this, it is also the case that information provided by SCONE can already be learned by CAP endpoints through various other mechanisms (e.g. the effect of on-path throttlers is clearly visible by observing application traffic packet flows). SCONE simply makes the signaling explicit, rather than requiring it to be observed and inferred separately.

8. IANA Considerations

This document has no IANA actions.

9. Editor’s Notes

This section to be removed prior to any potential publication within an RFC.

* The "CSP" term is overloaded, especially with regard to web technology, and might be changed to "carrier", "network operator", etc. in the future, but would need to be consistent with terminology in other SCONE documents.

10. References

10.1. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <https://www.rfc-editor.org/rfc/rfc2119>.

[RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <https://www.rfc-editor.org/rfc/rfc8174>.

[RFC9000] Iyengar, J., Ed. and M. Thomson, Ed., "QUIC: A UDP-Based Multiplexed and Secure Transport", RFC 9000, DOI 10.17487/RFC9000, May 2021, <https://www.rfc-editor.org/rfc/rfc9000>.

10.2. Informative References

[I-D.druta-scone-video-session-data-rate] Druta, D., Halepovic, E., and T. Karagioules, "Video Session Data Rate for SCONE protocol", Work in Progress, Internet-Draft, draft-druta-scone-video-session-data-rate-00, 30 January 2025, <https://datatracker.ietf.org/doc/html/draft-druta-scone-video-session-data-rate-00>.

[I-D.ihlar-scone-masque-mediabitrate] Ihlar, L. M. and M. Khlewind, "MASQUE extension for signaling throughput advice", Work in Progress, Internet-Draft, draft-ihlar-scone-masque-mediabitrate-02, 3 March 2025, <https://datatracker.ietf.org/doc/html/draft-ihlar-scone-masque-mediabitrate-02>.

[I-D.joras-scone-video-optimization-requirements] Joras, M., Tomar, A., Tiwari, A., and A. Frindell, "SCONE Video Optimization Requirements", Work in Progress, Internet-Draft, draft-joras-scone-video-optimization-requirements-00, 4 November 2024, <https://datatracker.ietf.org/doc/html/draft-joras-scone-video-optimization-requirements-00>.

[I-D.thoji-scone-trone-protocol] Thomson, M., Huitema, C., Oku, K., Joras, M., and L. M. Ihlar, "Transparent Rate Optimization for Network Endpoints (TRONE) Protocol", Work in Progress, Internet-Draft, draft-thoji-scone-trone-protocol-00, 3 March 2025, <https://datatracker.ietf.org/doc/html/draft-thoji-scone-trone-protocol-00>.

[I-D.tomar-scone-ecn]
Tomar, A., Ihlar, L. M., Eddy, W., Swett, I., Tiwari, A., and M. Joras, "SCONE Need for Defining A New On-Path Signaling Mechanism", Work in Progress, Internet-Draft, draft-tomar-scone-ecn-01, 7 May 2025, <https://datatracker.ietf.org/doc/html/draft-tomar-scone-ecn-01>.

[I-D.tomar-scone-privacy]
Tomar, A., Eddy, W., Tiwari, A., and M. Joras, "SCONE Privacy Properties and Incentives for Abuse", Work in Progress, Internet-Draft, draft-tomar-scone-privacy-01, 7 May 2025, <https://datatracker.ietf.org/doc/html/draft-tomar-scone-privacy-01>.

[RFC8303] Welzl, M., Tuexen, M., and N. Khademi, "On the Usage of Transport Features Provided by IETF Transport Protocols", RFC 8303, DOI 10.17487/RFC8303, February 2018, <https://www.rfc-editor.org/rfc/rfc8303>.

[RFC8304] Fairhurst, G. and T. Jones, "Transport Features of the User Datagram Protocol (UDP) and Lightweight UDP (UDP-Lite)", RFC 8304, DOI 10.17487/RFC8304, February 2018, <https://www.rfc-editor.org/rfc/rfc8304>.

[W3C-NetInfo]
W3C, "The Network Information API", 11 May 2020, <https://wicg.github.io/netinfo/>.

[YouTube] YouTube, "YouTube Plan Aware Streaming", 21 March 2024, <https://datatracker.ietf.org/meeting/119/materials/slides-119-sconepro-youtube-plan-aware-streaming-01>.

Acknowledgments

This document represents collaboration and inputs from others, including:

* Anoop Tomar

* Matt Joras

* Bryan Tan

Additional, helpful critique and comments were provided by:

* Lucas Pardue

* Ted Hardie
* Michael Welzl
* Gorry Fairhurst
* Brian Trammell

Authors' Addresses

Wesley Eddy
Meta
Email: wesleyeddy@meta.com

Abhishek Tiwari
Meta
Email: atiwari@meta.com

Alan Frindell
Meta
Email: afrind@meta.com