Dissecting Performance of Production QUIC

Alexander Yu, Theophilus A. Benson

摘要

谷歌基于 UDP 四层网络协议的标准版本 IETF QUIC，由于其超越 TCP 的优秀性能，已经在各大互联网公司中迅速部署。如今，由于 QUIC 的迅速部署，对其在工业生产下的性能分析十分缺乏。目前已有的分析用的是未经优化过的，未经内核调试的开源 QUIC 服务器：这些分析无法代表生产部署的情况。因而产生了一个问题：在生产环境中 QUIC 的性能是否真的优于 TCP。在本文中，我们实施了第一个在 QUIC 和 TCP 之间的比较研究，在实验中连接 Google, Facebook, 以及 Cloudflare 的生产节点，并配置不同的网络条件、工作流程以及客户端部署。

为了理解我们的结果，我们开发了一个工具以系统地可视化造成两个协议性能差别的原因。通过我们的工具，我们获得了几个关键的观察结论。首先，虽然 QUIC 有一些优于 TCP 的内在优势，例如最坏情况下的 1-RTT 握手，但是其总体性能很大程度上取决于服务器对拥塞算法的选择以及在边缘网络情况下拥塞控制的部署。其次，我们发现一些 QUIC 客户端的特殊的配置选择可以带来性能优化。最后，我们证明了 QUIC 对队头阻塞问题的移除在实际中对网页性能的提升微乎其微。总之，我们的观察说明 QUIC 的性能与其部署的设计、漏洞和配置选择有内在联系，说明 QUIC 的测试并不总是能反映协议的内容，而且通常难以在部署中广泛应用。

关键词：QUIC; HTTP/3; Transport Protocal; Multiplexing; Congestion Control

Abstract

IETF QUIC, the standardized version of Google’s UDP-based layer-4 network protocol, has seen increasing adoption from large Internet companies for its benefits over TCP. Yet despite its rapid adoption, performance analysis of QUIC in production is scarce. Most existing analyses have only used unoptimized open-source QUIC servers on non-tuned kernels: these analyses are unrepresentative of production deployments which raises the question of whether QUIC actually outperforms TCP in practice. In this paper, we conduct one of the first comparative studies on the performance of QUIC and TCP against production endpoints hosted by Google, Facebook, and Cloudflare under various dimensions: network conditions, workloads, and client implementations.

To understand our results, we create a tool to systematically visualize the root causes of performance differences between the two protocols. Using our tool we make several key observations. First, while QUIC has some inherent advantages over TCP, such as worst-case 1-RTT handshakes, its overall performance is largely determined by the server’s choice of congestion-control algorithm and the robustness of its congestion-control implementation under edge-case network scenarios. Second, we find that some QUIC clients require non-trivial configuration tuning in order to achieve optimal performance. Lastly, we demonstrate that QUIC’s removal of head-of-line (HOL) blocking has little impact on web-page performance in practice. Taken together, our observations illustrate the fact that QUIC’s performance is inherently tied to implementation design choices, bugs, and configurations which implies that QUIC measurements are not always a reflection of the protocol and often do not generalize across deployments.

Keywords：QUIC; HTTP/3; Transport Protocal; Multiplexing; Congestion Control

1 引言

随着在线服务的不断发展，网页性能已经成为了在线服务提供商和在线内容提供商的关注重点，例如 Google, Facebook 以及 Netflix 等。由此，新的网页协议的诞生并不令人惊讶，例如 HTTP2 和 HTTP3。在这之中，一个关键的协议得到了更多的利用，那就是 QUIC 协议，同时其成为了新兴协议的基础，如 HTTP3。如今，已有大量的针对其性能的研究。而不幸的是，大部分工作都是基于未优化过的版本，或者是探索了一个限定的荷载范围。我们需要一个更具有实践和现实意义的工具来标准化探索 QUIC 的性能。这种需求随着不同的 QUIC 部署的大量出现而变得越发重要。但是，目前没有一个可以在不同的部署和客户端之间相互操作的一般性工具，导致不同的研究者在实验中获得了矛盾的结论：QUIC 和 TCP 的性能究竟孰优孰劣？

在这篇论文中，其目标是研制一个简单轻量但是普遍适用的工具来标准化分析 QUIC 部署，并识别是实现中的什么问题导致了性能上的差异。我们的方法构建遵循以下原则：首先，与其探索大部分未经优化的开源 QUIC 部署，不如分析已经投入使用的生产节点，例如 Google 和 Facebook。基于这些节点，我们可以取得未开源的代码、配置以及内核优化的优势。其次，为了获得对 QUIC 的更全面的认识，我们不仅比较了服务端的部署，而且比较了多种 QUIC 客户端。为了支持这些原则，我们开发了一个测试工具可以分析不同网络情况下的部署行为，同时涉及了测试脚本以分析协议的行为。

本次复现选择该工作，其目的是为了更好的了解、学习 QUIC 协议，了解该协议的具体特点和运作过程，同时进一步了解当前已有的关于 QUIC 的各项实现和工作，并通过部署等实践方法进行动手尝试。此外，通过本次复现工作，学习网络编程以及网络分析的方法，利用所复现论文的实验方法实现具体的网络环境和性能的分析。在思想方法上，学习构造网络性能分析实验的逻辑体系和方法，以指导今后的学习和研究。

2 相关工作

2.1 QUIC

QUIC[1]作为 HTTP3 的核心，其基本将 TCP 和 TLS 复合成了单个建立在 UDP 之上的传输协议。也就是说，QUIC 提供了与 TCP 同等的可靠性，例如可靠的数据交付以及严格的流量控制，同时也提供了安全和流的复合。

与 TCP 不同的是，QUIC 直接将流复用整合到了传输协议的设计上。这样的设计降低了对数据包传输的排序限制，数据只需要在流的级别上进行有序传输，而不用在连接的级别上进行有序传输。因此，QUIC 得以减轻 TCP 上存在的队头阻塞问题。然而，QUIC 的流多路复用并不影响处理所有数据的总时间。所以，剖析和分析 QUIC 复用对应用程序性能的影响显得更为重要。

QUIC 另一个由于 TCP/TLS 协议栈的优势在于，对于新的或已经存在的对话，其建立安全连接的往返次数更少。与 QUIC 不同的是，TCP 并没有将 TLS 的协商加入到协议中，这意味着 TLS 必须自行完成握手，使得往返次数增加。因此，QUIC 默认使用 TLS 的机制使得 TLS 的握手不用单独进行。也就是说，QUIC 完成 HTTPS 请求的时间至少比 TCP 少一个 RTT。

QUIC 的损失恢复设计建立在 TCP 实际部署十年的经验之上，使得其可以利用并简化很多 TCP 的机制，例如快速重传[2]、选择性确认（TCP SACK）[3]、重传确认（TCP RACK）[4]等。

同时，QUIC 还为新的应用协议 HTTP/3 提供了条件，以充分应用 QUIC 的流多路复用能力。HTTP/3 不同于前一个版本 HTTP/2，其从报头中删除流元数据，使用新的报头压缩算法，并采用新的优先级方案。结合这些功能可以提高 QUIC 的性能，因为与 HTTP/2 相比，HTTP/3 的有效载荷大小更小，并且 HTTP/3 下的资源优先级更容易获得.

QUIC 的用户空间特性为传输层的配置和实验提供了独特的机会，利用 QUIC 可以使得部署更容易测试，且更容易使用不同的流量控制和拥塞控制策略以适应不同的需求。然而，这种不均匀的特性也容易导致不同部署间的性能差异。

2.2 基于 gQUIC 的性能分析

gQUIC 是谷歌版本的 QUIC，其实现与 IETF 版本的 QUIC 有较大差异。这些差异意味着基于之前版本 QUIC 的工作可能并不适用于如今的 QUIC 版本。简而言之，这些工作的局限性使得基于最新版本的研究迫在眉睫。对时延设计的标准方法关注于对 Chromium 的开源 gQUIC 服务器在本地的表现。利用这个方法，实验者完全控制了会影响性能的不同维度，例如有效荷载分布以及网络状况。然而，他们同样对配置内核和应用设置负责。因此，在各个研究之间配置的差异导致了对于 gQUIC 和 TCP 性能的完全相反的结论[5-10]：一部分人认为 gQUIC 优于 TCP，而另一部分人相反。根据根本原因的分析，这些研究有的分析 gQUIC 的 0-RTT 连接建立，有的分析对队头阻塞问题的移除，有的分析拥塞控制算法以及重传机制，这些都是造成 gQUIC 和 TCP 性能差异的主要因素。对于前一个版本 gQUIC 来说，协议和实际部署之间的差异并不必要，因为只有 Google 的部署能够同时分析大量用户的流量。[11]随着当前对 QUIC 部署的激增，许多内容提供商决定开发内部解决方案，所以需要比较这些出现的不同版本的 QUIC。

gQUIC 的部署缺乏多样性，这也会导致对根本原因分析的误导。例如，当讨论拥塞控制的时候，研究者通常使用“QUIC 的拥塞控制”[5,7]来描述，然而这是欠缺考虑的，因为目前并没有针对 QUIC 的专门的拥塞控制标准。然而，即使是基于相同拥塞控制算法的不同部署的 QUIC，也可能会造成性能上的差异。

3 本文方法

3.1 本文方法概述

为了强调对生产环境下性能分析，文章只选择合适的应用协议进行标准化测试。具体来说，为了标准化 QUIC 性能，选择了使用 HTTP/3 协议（运行于 QUIC 之上）的服务器，而对于 TCP，选择使用了 HTTP/2（运行于 TCP 之上）的协议，如图 1。在实验中，文章使用了两种实验流程：单一对象资源以及多对象网页资源。在单一对象资源实验中，可以分析原始协议的行为，而在多对象网页资源实验中，可以进一步探索在页面加载过程中的协议行为。对于这两种实验流程，我们选择了静态资源和静态网页，从而避免了用户证书审核的步骤，从而避免激发基于用户的业务逻辑，使得性能的测量更准确。

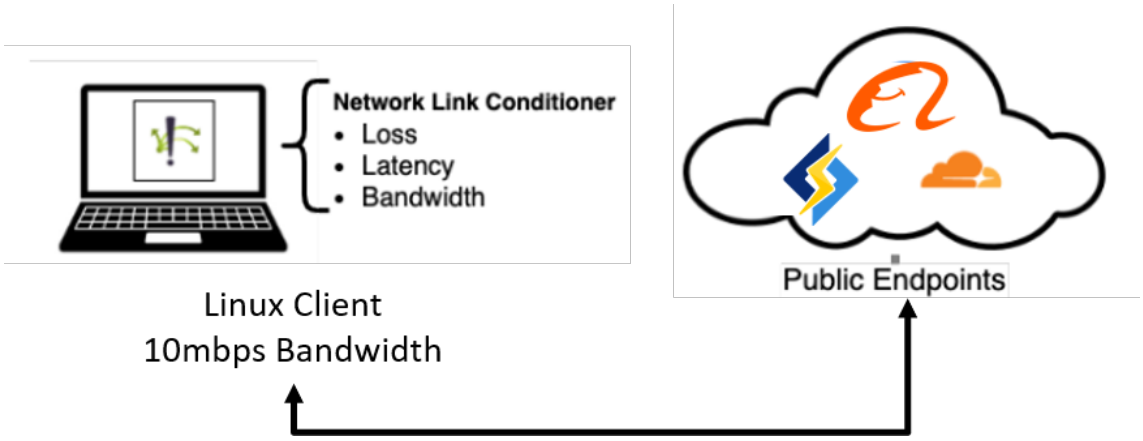


图 1: 实验设置

3.2 单一对象实验

在实验中，首先使用一个包含 HTTP/2 和 HTTP/3 客户端的集合确定一些由 HTTP/2 和 HTTP/3 端点产生的，有趣的或未曾想到的性能差异。在实验中使用特殊组合的对象大小、网络环境、客户端集合以及生产部署来产生有趣的测试结果。实验利用 Python 脚本来产生基于客户端日志的可视化结果。这些可视化结果可以帮助我们识别 HTTP/2 和 HTTP/3 方法上的性能偏差。图 2概括了单一对象标准化实验的实验流程：首先利用客户端，在给定的网络条件下对所选的单一资源进行访问，在过程中对整个过程进行记录形成日志文件，最后根据日志文件进行可视化并进行分析。

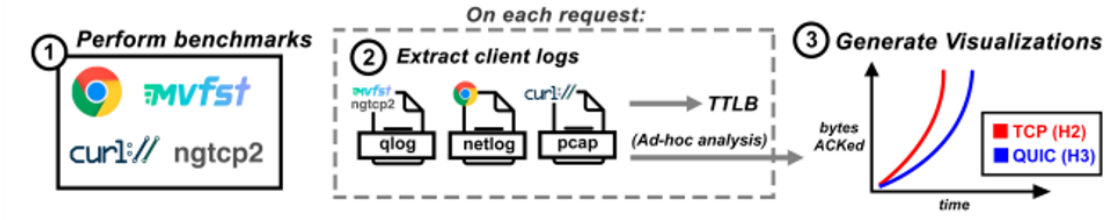


图 2: 单一对象实验

3.3 多对象网页实验

多对象网页的性能分析需要分离单一对象的实验，因为网页的加载可能会包含成百上千个单一的网络请求。由此，对多对象的网页分析需要关联较高级别的用户体验质量（QoE）评价指标，例如首屏展现平均值（Speed Index, SI）以及网页加载速度（Page Load Time, PLT）。与单一对象实验类似，其实验流程为：利用客户端在给定网络条件下对所选的网页进行访问，并将访问过程记录为日志文件，最后根据日志文件进行可视化分析, 如图 3。

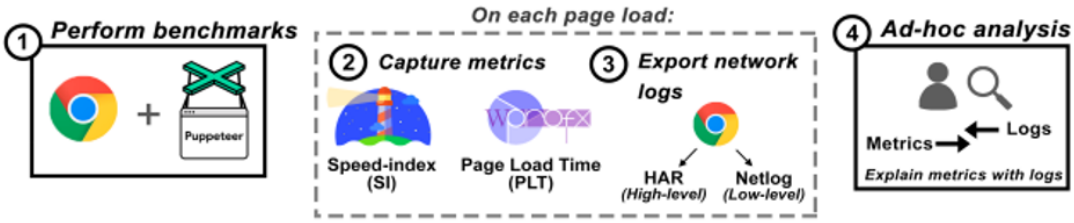


图 3: 多对象网页实验

4 复现细节

4.1 与已有开源代码对比

本次复现论文给出部分源代码，但是所给出代码较为分散，同时需要自行配置运行环境和相关参数。所以复现时首先通过阅读其代码理解其运行逻辑，然后在本地配置相关运行环境，之后参考所给出的源代码进行拆分整合，形成能够运行的代码，具体细节如下：

1. 论文所给代码将单一对象实验和多对象网页实验合并在一起，然后通过 Linux Shell 脚本进行运行。但是在实际操作中，无法直接运行所给源代码，故将单一对象实验和多对象网页实验拆分，并分别进行修改和重构，使其能够在本地环境中运行。

2. 论文运行的环境较为复杂，需要安装和配置环境的操作较多，具体细节见之后的实验环境一节。所配置的环境包括：Facebook Proxygen, ngtcp2, Chrome HTTP/2, Chrome HTTP/3, cURL, Lighthouse, Puppeteer。

3. 由于国内网站访问限制等问题，复现的论文使用的 Google、Facebook 网站无法在虚拟机实验环境中访问，故将访问的 url 配置信息修改为国内可访问的 Alibaba、Litespeed、Cloudflare 网站，并根据论文的实验条件，寻找适合的单一对象资源和多对象网页资源进行替换。由于网站的修改导致有部分实验所访问的资源与原实验中的资源数量和大小上有些许出入，同时 Alibaba 网站暂未支持 HTTP/3，故实验结果有所出入。

4. 在源代码中使用的 Litehouse 配置信息使得源代码无法直接运行，通过学习和阅读 Lighthouse 的使用方式和配置方法，对代码中的多处配置进行修改：分别对 HTTP/2 和 HTTP/3 客户端的配置进行不同的修改，修改后代码可以直接运行。

5. 在源代码中，对 Chrome HTTP/2 和 HTTP/3 的使用均利用 Puppeteer 的接口调用运行，而其余客户端则使用 docker 进行运行，使得环境配置更加复杂而易出错。对该部分代码进行修改，直接利用 Linux Shell 脚本对其余已经安装配置好的客户端进行操作和运行，并生成脚本存储在本地设置的位置。

6. 源代码中几乎没有对本地服务器配置的内容，为了测试在本地服务器的实验效果，分别安装配置了 Apache2, Nginx HTTP/3, Lightspeed HTTP/3 进行测试。对以上三个服务器的安装和配置花费了较多时间和精力，但是在代码中运行的情况不甚理想，暂未能实现在对本地服务器的访问实验。

7. 源代码中 Python 脚本的可视化工作对不同的访问信息进行分析，同时其中的较多内容与论文作者自己的环境和设置有关，故在对可视化内容的复现中做了较多修改，使其能够将修改后的复现日志进行可视化。

Procedure 1 Single-Object.

Input: Resource Information *ResInfo,* Resorce Url *url*
Output: Time-To-Last-Byte *TTLB*
Load the configuration information ResInfo first

```
if Client is H2(orH3) then
    if H2(or H3) folder is not exist then
        establishfilefolder
    end
    if H2(or H3) log file is not exist then
        establishlogfile
    end
    Open H2(orH3) log file
     Get url
     for i in Runtimes do
        Configure H2(orH3) Client
         Request the Single Object
         if Request error then
            break
        end
        Record the request information
         Close the connection
         TTLB = Total_time − Blocked_time − Queued_time − DNS_time
         Write TTLB in Record file
    end
end
```

Procedure 2 Multi-Object.

Input: Resource Information *ResInfo,* Resource Url *url*
Output: Speed Index *SI,* Page Load Time *PLT*
Load the configuration information ResInfo first

```
if Client is H2(orH3) then
    if H2(or H3) folder is not exist then
        establishfilefolder
    end
    if H2(or H3) log file is not exist then
        establishlogfile
    end
    Open H2(orH3) log file
     Get url
     for i in Runtimes do
        Configure H2(orH3) Client
         Request the Web Page
         if Request error then
            break
        end
        Record the request information
         Record the number of Resources
         Record the size of Resourses
         Close the connection
         Calculate the SI and PLT
         Write SI and PLT in Record file
    end
end
```

4.2 实验环境搭建

4.2.1 Linux 环境搭建

整个实验运行在 VMware 虚拟机上安装的 Ubuntu 20.04 TLS 系统中。

4.2.2 HTTP/2 HTTP/3 客户端环境搭建

为运行整个实验，需要安装以下几个开源客户端环境，并根据所给的注解网站进行环境配置。Facebook Proxygen[1]，Ngtcp2[2], cURL[3], Puppeteer[4]。其中，Chrome 客户端内置于 Puppeteer 中，通过调用 API 对其进行配置和使用。

4.2.3 本地服务器环境搭建

为了在本地搭建的服务器中进行测试，分别搭建了以下几个开源服务器端：Apache HTTP/2；Nginx HTTP/3；Lightspeed HTTP/3。

4.3 创新点

在原复现论文中，所访问的服务器均基于 HTTP/3 协议建立，据此分析 QUIC 的性能。然而，虽然目前 QUIC 的部署越来越多，也有很多内容提供商基于 QUIC 提供了 HTTP/3 的服务，但是不可否认的是，目前仍有相当一部分网站没有提供 HTTP/3 服务，而仍然使用 HTTP/2。因此，在本次复现中，将增加基于 HTTP/2 的网站进行实验分析。

5 实验结果分析

本部分对实验所得结果进行分析，详细对实验内容进行说明，实验结果进行描述并分析。

对于单一对象的基准测试，实验利用 Time-To-Last-Byte(TTLB) 指标来评价表现，TTLB 即客户端发送第一个数据包与服务器接受最后一个数据包之间的时间差。而对于多对象网页实验的基准测试，实验利用 Speed Index(SI) 以及 Page Load Time(PLT) 两个指标来评价表现，SI 代表的是页面填充的速度，而 PLT 则表示页面填充的时间。实验中使用 Google Lighthouse 在客户端中获取 SI 数据，使用 WProfX 计算来自 Chrome 追踪时间的 PLT。在以下部分的实验图例中，蓝色方块代表 HTTP/3 的速度更快，红色方块代表 HTTP/2 的速度更快。

5.1 单一对象实验结果分析

在单一对象的实验中，分别选取了 Alibaba 和 Litespeed 网站上不同大小的资源进行访问，资源大小分别为：100KB, 1MB, 5MB。

对于单一对象的资源，TCP 和 QUIC 的性能差异应该是比较小的，因为 QUIC 对于单一 Web 对象的获取只使用双向流发送和请求数据，其效率与 TCP 大致相同。但是根据实验结果来看，HTTP/2 和 HTTP/3 在不同网络状况和资源大小的情况下，TTLB 指标有明显差异。

[1]https://github.com/facebook/proxygen

[2]https://github.com/ngtcp2/ngtcp2

[3]https://curl.se/download.html

[4]http://puppeteerjs.com/

如图 4的第一行，在没有损失和延迟的时候，HTTP/3 的效果明显由于 HTTP/2，而在资源大小增加之后，其性能差距开始缩小。原因可能是：首先，QUIC 的 1-RTT 机制使其在握手的过程中相比 TCP 降低了一部分延迟，不同的内容提供商在获得资源这个过程中的行为是一致的，说明相比内容提供商不同的实现和优化，协议不同的设计机制可能带来更好的性能。其次，相比于长时间的连接来说，1-RTT 的机制在短链接上带来的性能优势更加明显。因此，QUIC 的 1-RTT 机制在负载较小的情况下对连接的影响比较大，随着负载的增加其优势会不断降低。

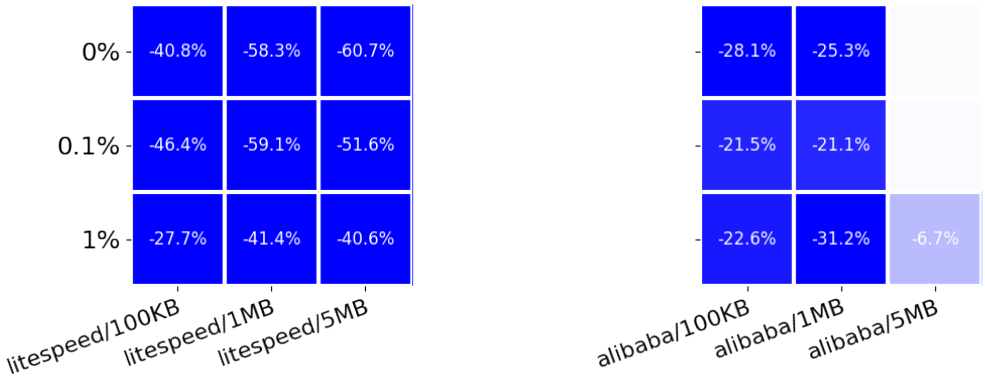


图 4: 单一对象实验-增加损失

对于增加延迟的实验来说，其实验结果与增加损失的实验相差无几，如图 5右边在 litespeed 网站的实验结果，HTTP/3 在较小的资源上性能最优，而随着资源的增大性能优势逐渐降低。然而在 Alibaba 的网站上性能则不同：在增加延迟后，对于 100KB 资源的实验中，HTTP/2 性能更优。其原因可能是：由于在复现实验中，所访问的网站有所限制，国内能够访问到的支持 HTTP/3 的大型网站较少，所以复现中使用了未能支持 HTTP/3 的 Alibaba 网站作为代替，使得实验中的实验结果与复现论文中的实验结果出现差异。除此之外，由于使用 HTTP/3 的客户端对 HTTP/2 的网站中的单一资源进行访问，在增加了延迟的情况下，由于 QUIC 和 TCP 的握手机制的差异，可能出现超时的情况，导致 HTTP/2 的性能由于 HTTP/3。最后，可以看到实验结果中，对 Alibaba 网站较大资源的访问还是 HTTP/3 的性能更优，说明在利用 HTTP/3 客户端对 HTTP/2 服务器的访问中，HTTP/3 的资源下载速度快于 HTTP/2，由此出现了小资源上 HTTP/2 性能更优，而资源增大后 HTTP/3 性能更优的现象。

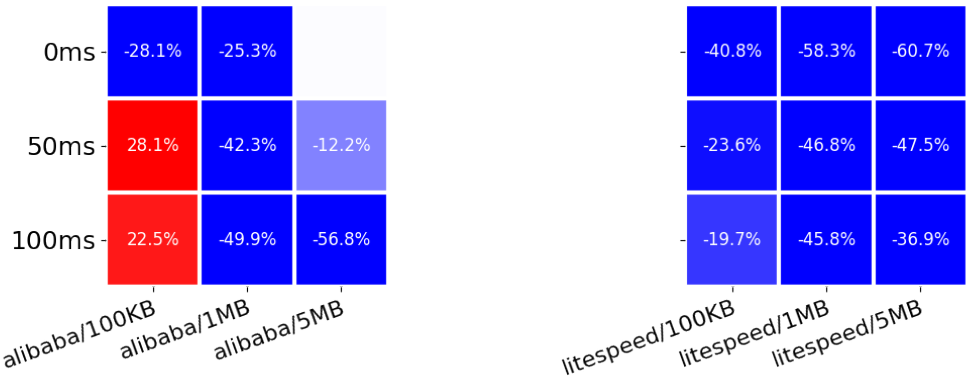


图 5: 单一对象实验-增加延迟

5.2 多对象网页实验结果分析

在多对象网页实验中，由于国内网站访问的限制，无法在虚拟机的环境下对复现论文中的部分网页进行访问（Google, Facebook），同时对于可访问的网站，其在国内对网站节点访问时网络波动较大，使得整个实验过程可能存在较大误差。但是，我们仍可以从实验中观察到 QUIC 和 TCP 协议之间的部

分差异。

在多对象网页实验中，分别对不同大小的网页资源进行访问，将所访问的网页划分为 3 个等级：Small, Medium, Large。对于网页加载时间来说，QUIC 在访问多对象的网页资源时，其性能应该明显优于 TCP，所复现论文中的实验结果也是如此，如图 6。这是因为 QUIC 在对多对象网页的访问是，HTTP/3 使用多路复用流（Stream）的机制，使得在整个访问过程中其速度快于基于 TCP 的 HTTP/2，同时在 QUIC 中对于队头阻塞问题的解决方法也使得其速度有一定的提升。

图 6: 原文多对象网页实验-PLT

然而，网络环境的不稳定使得情况恰好相反，如图 7所示。由于网络环境的不稳定，使得 QUIC 中的流多路复用机制中一个连接中的每一条流都可能出现损失和延迟，因此在整个页面访问过程中其所消耗的时间多于稳定的 TCP，加上 QUIC 中的拥塞机制并非针对 QUIC 所设计的机制，其并没有根据该情况所设计的机制来保证连接，所以实验结果中 HTTP/2 的性能基本都由于 HTTP/3。

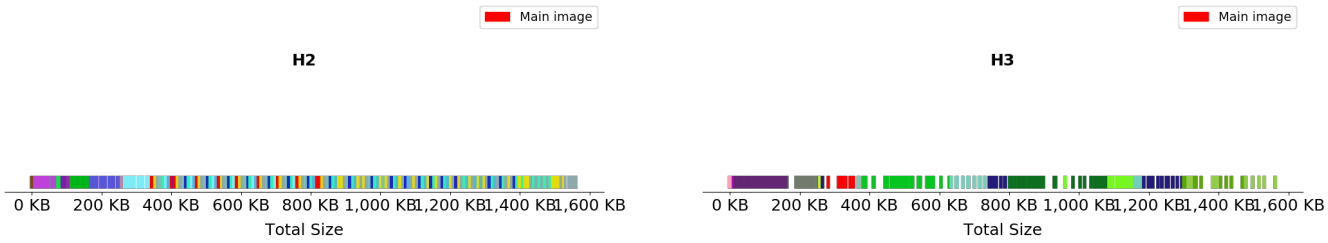
此外，从实验结果中可以看出，当网页资源增大时，HTTP/3 的性能逐渐提升，这与上一节中单一对象资源的实验类似：HTTP/3 在后续的资源获取速度上优于 HTTP/2，一旦建立了稳定的连接和流，便能以更快的速度获取资源。同时 QUIC 中对队头阻塞问题的缓解也使得在连接稳定时性能的提高。

图 7: 多对象网页实验-PLT

而对于页面填充速度来说，填充时间还与客户端对网页的渲染速度、网页图片资源的数量等有关，如图 8所示。随着网页尺寸的增大，HTTP/3 对整个网页的填充速度不断变慢，其与 HTTP/2 的差距也越大。

图 8: 多对象网页实验-SI

网页填充的过程还与 HTTP/2 和 HTTP/3 不同的机制有关，如图 9所示。在对 HTTP/2 和 HTTP/3 页面加载的相关网络日志分析中，可以看到，HTTP/2 对资源的加载采用轮询的方式：对每个资源进行分片加载，这就使得资源加载完成的时间总体靠后，而 HTTP/3 对资源的加载采用顺序的方式，使得靠前的主要资源优先加载，但是所有资源加载完成的时间比 HTTP/2 更长。



(a) HTTP2 (b) HTTP3

图 9: 多对象网页实验-H2/H3 资源加载过程

因此，该机制对页面加载时间 PLT 几乎没有影响，因为无论以怎样的顺序加载资源，其花费的总时间是不变的。然而，该机制对页面的渲染时间 SI 则有一定的影响，因为只有当资源完全加载完毕时，客户端才能对页面进行渲染并加载资源，在图 9中可以看到，HTTP/3 对主要图片的加载完成时间比 HTTP/2 更靠前。又如图 10所示，在按固定时间对网页截图的实验中可以看到，HTTP/3 加载主要图片的时间更快，其首屏渲染时间优于 HTTP/2。

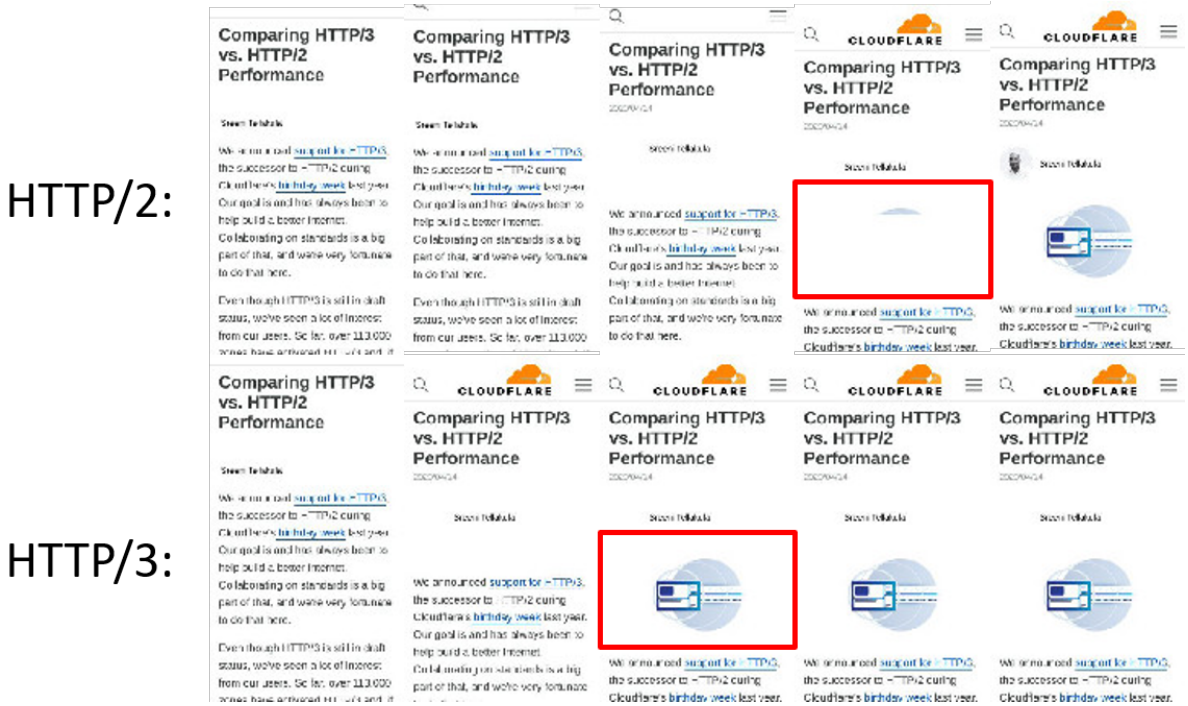


图 10: 多对象网页实验-网页加载截屏

6 总结与展望

在本次复现任务中，首先根据所复现论文的实验环境进行配置，然后阅读论文所给出的源代码，对其进行整理，构建能够在本地环境中运行的代码结构，最后进行实验。实验主要分为两个部分：单一对象资源实验以及多对象网页资源实验。在两个实验中，对复现论文中的访问对象进行调整，使实验能够在国内的网络环境中进行并获得实验结果。两个实验的实验流程类似：首先利用本地配置的客户端对资源进行访问和获取，然后记录整个过程中的信息并形成日志文件，最后根据日志内容进行分析和可视化。根据实验结果，可以获得以下结论：QUIC 协议相比 TCP 具有一定的优势，主要体现在 QUIC 中的 1-RTT 机制以及对队头阻塞问题的缓解上，然而，在网络条件不稳定的情况下，基于 TCP 的 HTTP/2 协议比基于 QUIC 的 HTTP/3 协议更具稳定性。此外，HTTP/2 和 HTTP/3 协议对资源的获取过程有差异，HTTP/2 采用轮询的方式获取资源，比起采用顺序获取资源的 HTTP/3 协议，其获得主

要资源的时间更长，但是其对网页所有资源的获取时间更短。

本次复现任务中，仍然有很多不足之处。首先，由于国内网站访问以及缺乏支持 HTTP/3 网站等原因，部分实验未能如期完成，使得实验结果不足，尽管根据现有的实验结果仍能看出所分析的两个协议存在的差异，但是在实验过程中带来的不稳定性目前没有克服。其次，在复现计划中准备使用本地服务器进行实验的任务没有完成，在复现的过程中我花费了大量的时间对本地支持 HTTP/3 的服务器进行配置和调整，同时也调研了大量关于国内支持 HTTP/3 的网站的问题，同时也花费了很多时间在调整代码上，但是最终仍没有解决该问题，所以在本地的实验没能完成，成为本次复现任务的一大缺陷。最后，由于投入了大量时间对上述问题进行解决，使得本次复现任务的创新性不足，实际上对网络性能的分析虽然难以做大的创新性的调整，但是仍然有很多实验可以尝试，遗憾的是没能如期进行。

虽然目前仍然有很多问题没能解决，但是在将来需要进行更进一步的网络协议分析工作的时候，仍然有很多任务可以尝试。首先，在复现论文中提到，目前没有专门针对于 QUIC 的拥塞控制算法，同时在实验中也可以看出 QUIC 和 TCP 不同的拥塞控制算法其结果有明显的差异，在今后的工作中，可以尝试在 QUIC 中使用不同的拥塞控制算法进行实验和分析，并针对目前较为广泛的网络应用进行调整和尝试。其次，在本次复现任务中没能做到在本地服务器进行性能测试的实验，在今后的工作中可以申请不同提供商的云服务器，自行配置 web 服务器然后再进行实验，这样不仅能克服国内网站访问的问题，同时还能够自行优化服务器和协议进行不同的实验。最后，在复现论文发表时 QUIC 协议和 HTTP/3 协议仍未标准化，所以论文中使用的仍然是应用较多的未标准化版本，随着 QUIC 协议和基于 QUIC 的 HTTP/3 协议已经正式标准化，在今后的工作中可以根据标准化的版本进行实验和分析。

参考文献

[1] LANGLEY A, RIDDOCH A, WILK A, et al. The quic transport protocol: Design and internet-scale deployment[R]. 2017: 183-196.

[2] ALLMAN M, PAXSON V, BLANTON E. TCP congestion control[R]. 2009.

[3] MATHIS M, MAHDAVI J, FLOYD S, et al. TCP selective acknowledgment options[R]. 1996.

[4] CHENG Y, CARDWELL N. Making linux TCP fast[C]//Netdev conference. 2016.

[5] CARLUCCI G, DE CICCO L, MASCOLO S. HTTP over UDP: an Experimental Investigation of QUIC [C]//Proceedings of the 30th Annual ACM Symposium on Applied Computing. 2015: 609-614.

[6] COOK S, MATHIEU B, TRUONG P, et al. QUIC: Better for what and for whom?[C]//2017 IEEE International Conference on Communications (ICC). 2017: 1-6.

[7] KAKHKI A M, JERO S, CHOFFNES D, et al. Taking a long look at QUIC: an approach for rigorous evaluation of rapidly evolving transport protocols[C]//proceedings of the 2017 internet measurement conference. 2017: 290-303.

[8] NEPOMUCENO K, de OLIVEIRA I N, ASCHOFF R R, et al. QUIC and TCP: a performance evaluation [C]//2018 IEEE Symposium on Computers and Communications (ISCC). 2018: 00045-00051.

[9] RÜTH J, WOLSING K, WEHRLE K, et al. Perceiving QUIC: Do users notice or even care?[C]//Proceedings of the 15th International Conference on Emerging Networking Experiments And Technologies. 2019: 144-150.

[10] YU Y, XU M, YANG Y. When QUIC meets TCP: An experimental study[C]//2017 IEEE 36th International Performance Computing and Communications Conference (IPCCC). 2017: 1-8.

[11] MADARIAGA D, TORREALBA L, MADARIAGA J, et al. Analyzing the adoption of QUIC from a mobile development perspective[C]//Proceedings of the Workshop on the Evolution, Performance, and Interoperability of QUIC. 2020: 35-41.