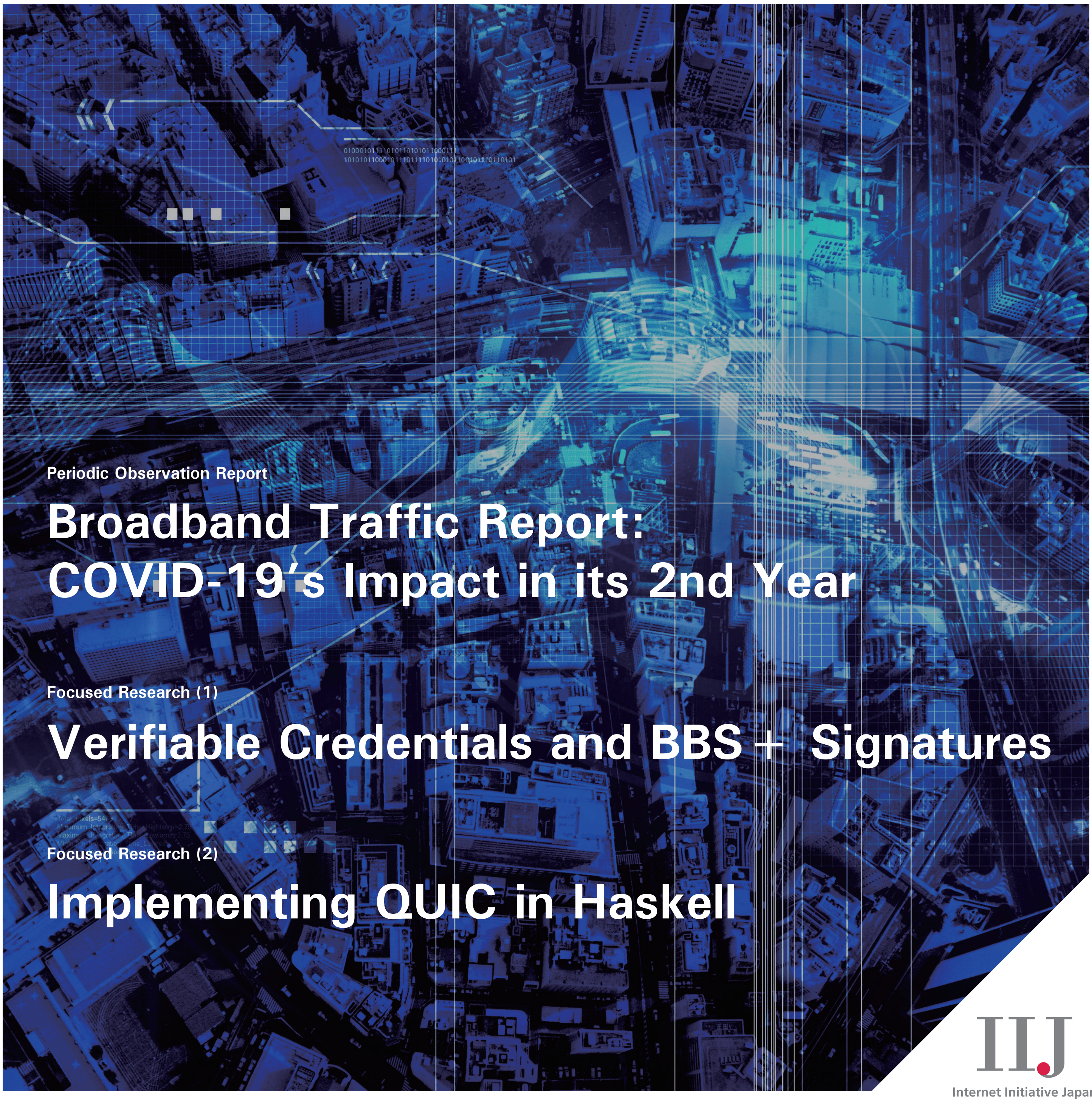
IIR

Internet Infrastructure Review

Nov.2021
Vol. 52



Internet Initiative Japan

Internet Infrastructure Review

November 2021 Vol.52

Executive Summary

In September 2021, the Japanese government launched its new Digital Agency. The Digital Agency's website[*1] says that it will rapidly build Japan's public-private infrastructure over the next five years to create a society in which the benefits of digitalization reach everyone. The agency's organizational chart also bears out its keen focus on digitalizing public services as well as the services of government ministries and agencies.

We might not give too much thought to government services until our own time comes to use them, and indeed, when I moved a few years ago, I was at times struck by the sheer amount of paperwork I had to complete and how inconvenient it all was. And amid the current COVID-19 pandemic, the popular press has been critical of the government, claiming that the payment of government handouts and the rollout of vaccines could have proceeded more smoothly if greater progress had been made on the use of information & communication technology (ICT) in government services.

How does the Japanese government's use of ICT stack up against the rest of the world? The United Nations' global e-government rankings[*2], released in July 2020, put Japan in 14th place out of the 193 UN Member States, while the "International Digital Government Rankings"[*3] released by the Institute of Digital Government at Waseda University in September 2020 have Japan in seventh place out of 64 leading ICT nations. No doubt it depends on the evaluation methodology, but it seems fair to say that while the Japanese government does not have a commanding lead over the rest of the world when it comes to digitalization, it is not as far behind as public criticism might suggest.

That said, promoting the use of ICT and pursuing digital transformation initiatives across society as a whole will be key to improving the lives of all. The Internet is crucial infrastructure for making this happen, and at IIJ, we hope to contribute toward such digital transformation efforts through our role in supporting the Internet.

The IIR introduces the wide range of technology that IIJ researches and develops, comprising periodic observation reports that provide an outline of various data IIJ obtains through the daily operation of services, as well as focused research examining specific areas of technology.

Our periodic observation report in Chapter 1 provides our analysis of IIJ's fixed broadband and mobile traffic. We are now in the second year marked by major changes in Internet traffic due to the COVID-19 pandemic. The results of this analysis elucidate how changes in Internet traffic reflect societal developments and changes in technology, including the impact of behavioral restrictions on traffic, the shift from PPPoE to IPoE in fixed broadband, the shift from HTTP to HTTPS, and the rise of the QUIC protocol, as also discussed in Chapter 3.

The first focused research report, in Chapter 2, looks at Verifiable Credentials (VCs), which lie at the core of self-sovereign identity (SSI), and discusses BBS+ signatures, which make VCs possible. As digital transformation initiatives advance, SII is likely to become increasingly important as it allows users to independently manage their own digital identities. We also discussed SSI in IIR Vol. 43 (https://www.iij.ad.jp/en/dev/iir/043.html), and the development of technologies to enable SSI has progressed in the two years since then. The report in Chapter 2 also touches on the differences between traditional digital certificates and VCs, VC implementations from Japan and abroad, standardization, and future challenges.

The second focused research report, in Chapter 3, discusses an effort to implement QUIC, which was recently standardized in RFC 9000, in Haskell. As well as participating in the discussion of new protocols, the author actually implements them and tests interoperability with other implementations, putting a lot of effort into ensuring a high level of completeness when it comes time to use those implementations. Many implementations use event-driven programming, but by taking advantage of Haskell's features and adopting a threaded programming approach, the author has been able to test the specifications from a different perspective than other implementers. The specific implementation points covered in this chapter help to provide a deeper understanding of the QUIC protocol.

Through activities such as these, IIJ strives to improve and develop its services on a daily basis while maintaining the stability of the Internet. We will continue to provide a variety of services and solutions that our customers can take full advantage of as infrastructure for their corporate activities.

Junichi Shimagami
Mr. Shimagami is a Managing Director and the CTO of IIJ. His interest in the Internet led to him joining IIJ in September 1996. After engaging in the design and construction of the A-Bone Asia region network spearheaded by IIJ, as well as IIJ's backbone network, he was put in charge of IIJ network services. Since 2015, he has been responsible for network, cloud, and security technology across the board as CTO. In April 2017, he became chairman of the Telecom Services Association of Japan MVNO Council.

*1 Digital Agency, "What is the Digital Agency?" (https://www.digital.go.jp/en).
*2 Department of Economic and Social Affairs, United Nations, "UN E-Government Survey 2020" (https://publicadministration.un.org/egovkb/en-us/Reports/UN-E-Government-Survey-2020).
*3 Institute of Digital Government at Waseda University, "International Digital Government Rankings" (https://idg-waseda.jp/ranking.htm).

Broadband Traffic Report: COVID-19's Impact in its 2nd Year

1.1 Overview

In this report, we analyze traffic over the broadband access services operated by IIJ and present the results each year[*1][*2][*3][*4]. Here, we again report on changes in traffic trends over the past year, based on daily user traffic and usage by port.

As in 2020, home Internet usage again increased under the COVID-19 pandemic, with broadband traffic staying in an uptrend. Meanwhile, with people venturing outdoors less, mobile usage has been largely range-bound.

Figure 1 graphs the overall average monthly traffic trends for IIJ's fixed broadband services and mobile services. IN/OUT indicates the direction from the ISP perspective. IN represents uploads from users, and OUT represents user downloads. Because we cannot disclose specific traffic numbers, we have normalized the data, setting the OUT observations for January 2020 for both services to 1.

Broadband services traffic surged from March to May 2020, when COVID-19 cases were starting to ramp up in Japan. It fell slightly in June after Japan's state of emergency was lifted but turned up again from August. Over the past year, broadband IN traffic increased 20% and OUT traffic increased 23%. While these are smaller increases than the year-earlier figures of 43% and 34%, the growth rates do appear to have returned to their former levels. Mobile services traffic, meanwhile, remained range-bound overall during this period amid lower rates of use outside the home/office, despite an increase in the use of services for remote work. Over the past year, mobile IN traffic increased 39% and OUT traffic fell 1%. A year earlier, IN was up 28% and OUT down 7%.

The broadband figures include IPv6 IPoE traffic. IPv6 traffic on IIJ's broadband services comprises both IPoE and PPPoE traffic[*5]. As of June 2021, IPoE accounted for almost a third of all traffic, at 31% of IN and 30% of OUT broadband traffic overall, year-on-year increases of 7 and 10 percentage points, respectively. With PPPoE congestion having become quite noticeable amid COVID-19, users are increasingly shifting to IPoE, and use of IPoE thus continues to rise.

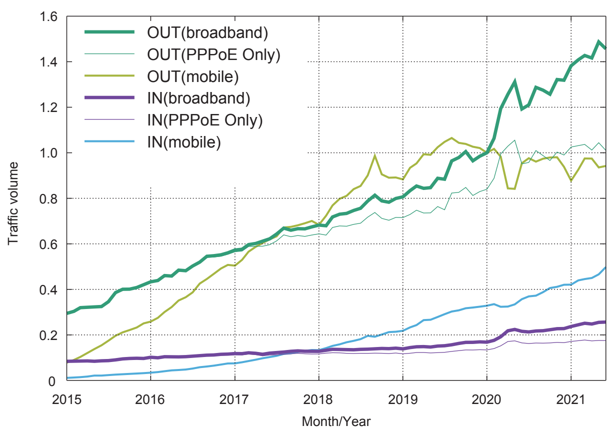


Figure 1: Monthly Broadband and Mobile Traffic

*1 Kenjiro Cho. Broadband Traffic Report: The Impact of COVID-19. Vol.48. pp4-9. September 2020.
*2 Kenjiro Cho. Broadband Traffic Report: Moderate Growth in Traffic Volume Ongoing. Vol. 44. pp4-9. September 2019.
*3 Kenjiro Cho. Broadband Traffic Report: Download Growth Slows for a Second Year Running. Vol. 40. pp4-9. September 2018.
*4 Kenjiro Cho. Broadband Traffic Report: Traffic Growth Slows to a Degree. Internet Infrastructure Review. Vol. 36. pp4-9. September 2017.
*5 Akimichi Ogawa and Satoshi Kubota. Tettei Kaisetsu v6 Plus. Lambda Note. January 2020 (https://www.jpne.co.jp/books/v6plus/, in Japanese).

We now look at broadband traffic by time of day on weekdays and weekends amid COVID-19. Traffic volume here is the sum of PPPoE and IPoE. Figures 2 and 3 show traffic for the following seven weeks: the week of February 25, 2020, before Japan's school closures; the week of April 20, 2020, corresponding to Japan's first state of emergency; the week of June 22, 2020, after the state of emergency was lifted; the week of August 31, 2020, when the second COVID-19 wave was easing; the week of January 18, 2021, the second state of emergency; the week of March 22, 2021, after Tokyo's state of emergency was lifted; and the week of July 5, which marked the start of the fifth wave of COVID-19, involving variant strains. We graph hourly average traffic volume figures for each of these weeks, partitioned into weekdays (Monday–Friday, excluding public holidays) and weekends (Saturday/Sunday). The lines in the lower part of each plot represent uploads, but here we focus on download volume.

First, we look at weekday traffic. Comparing February and April 2020 to see the impact of the first state of emergency, we see that traffic was up substantially in the daytime and that it also increased during evening peak hours. When the state of emergency was lifted in June, the additional daytime traffic fell to less than half what it had been, but peak hours saw almost no decline. Daytime traffic subsequently edged upward but did not return to its April 2020 levels until March of this year. Daytime traffic fell a little in July, which seems to reflect that schools were in session and remote work had declined a bit. Focusing on the 20:00–22:00 peak hours, we see a fairly consistent increase.

Weekend traffic, meanwhile, shows less variability than weekdays. Weather has a greater impact than school or remote work on the proportion of people at home on weekends. For example, poor weather on January 23 and 24 this year pushed home Internet usage upward; and on March 27 and 28, the cherry blossoms were in full bloom west of the Kanto region, resulting in more people going out and thus lower traffic levels. Traffic during nighttime peak hours is roughly the same as on weekdays.

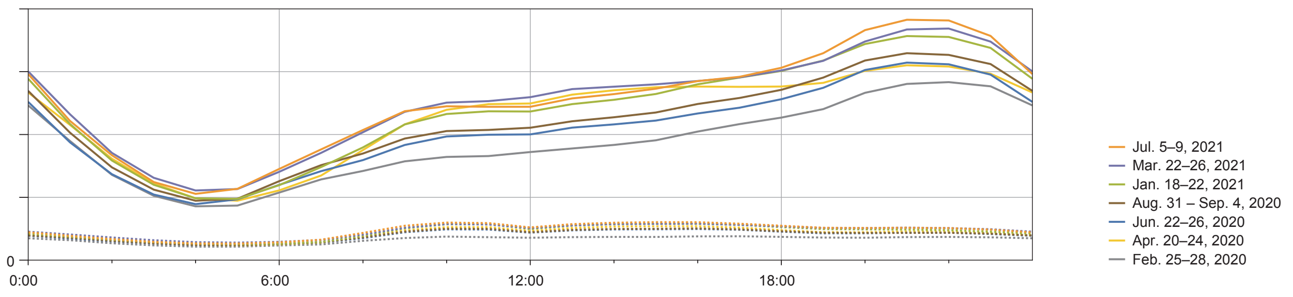


Figure 2: Hourly Average Broadband Traffic on Weekdays

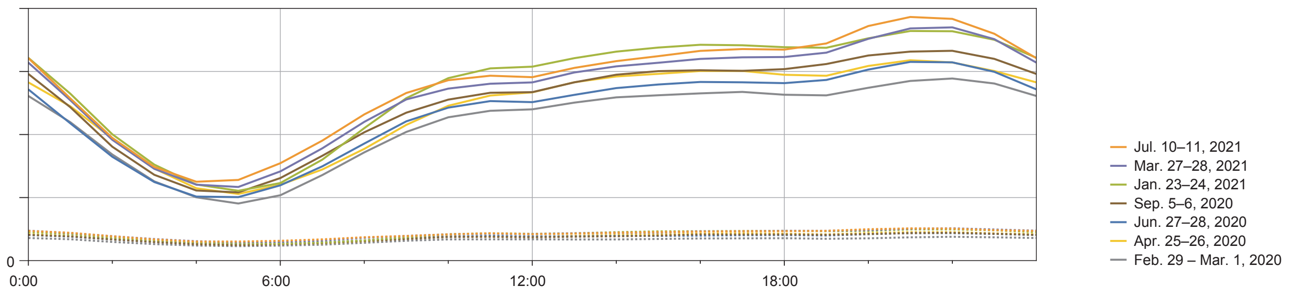


Figure 3: Hourly Average Broadband Traffic on Weekends

Note that IPoE traffic is not included in the following analysis, as detailed data is not available because we use Internet Multifeed Co.'s transix service for IPoE.

1.2 About the Data

As with previous reports, for broadband traffic, our analysis uses data sampled using Sampled NetFlow from the routers that accommodate the fiber-optic and DSL broadband customers of our personal and enterprise broadband access services. For mobile traffic, we use access gateway billing information to determine usage volumes for personal and enterprise mobile services, and we use Sampled NetFlow data from the routers used to accommodate these services to determine the ports used.

Because traffic trends differ between weekdays and weekends, we analyze traffic in one-week chunks. In this report, we look at data for the week of May 31 – June 6, 2021, and compare those data with data for the week of June 1–7, 2020, which we analyzed in the previous edition of this report.

Results are aggregated by subscription for broadband traffic, and by phone number for mobile traffic as some subscriptions cover multiple phone numbers. The usage volume for each broadband user was obtained by matching the IP address assigned to users with the IP addresses observed. We gathered statistical information by sampling packets using NetFlow. The sampling rate was set to around 1/8,192, taking into account router performance and load. We estimated overall usage volumes by multiplying observed volumes with the reciprocal of the sampling rate.

IIJ provides both fiber-optic and DSL broadband services, but fiber-optic access now accounts for the vast majority of use. Of users observed in 2021, 99% were using fiber-optic connections.

1.3 Users' Daily Usage

First, we examine daily usage volumes for broadband and mobile users from several angles. Daily usage indicates the average daily usage calculated from a week's worth of data for each user.

Since our 2019 report, we use daily usage data only on services provided to individuals. The distribution is heavily distorted if we include enterprise services, where usage patterns are highly varied. So to form a picture of overall usage trends, we determined that using only the individual data would yield more generally applicable, easily interpretable conclusions. Note that the analysis of usage by port in the next section does include enterprise data because of the difficulty of distinguishing between individual and enterprise usage.

Figures 4 and 5 show the average daily usage distributions (probability density functions) for broadband and mobile users. Each compares data for 2020 and 2021 split into IN (upload) and OUT (download), with user traffic volume plotted along the X-axis and user frequency along the Y-axis. The X-axis

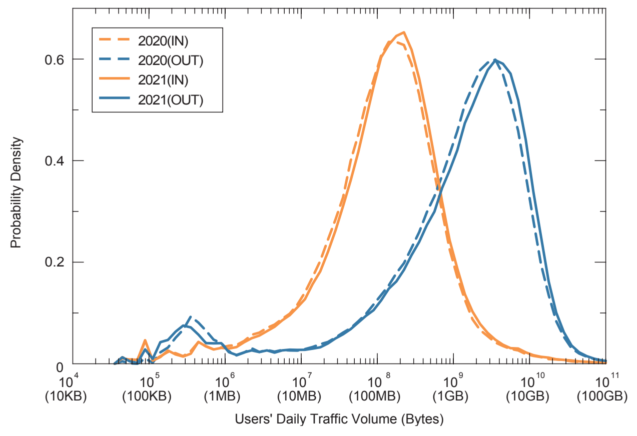


Figure 4: Daily Broadband User Traffic Volume Distribution Comparison of 2020 and 2021

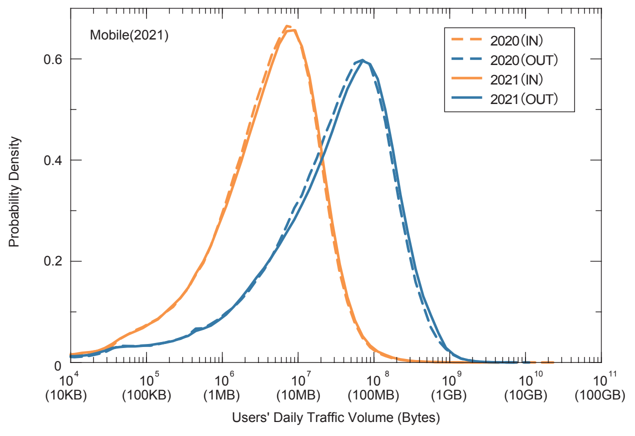


Figure 5: Daily Mobile User Traffic Volume Distribution Comparison of 2020 and 2021

shows volumes between 10KB (104) and 100GB (1011) using a logarithmic scale. Most users fall within the 100GB (1011) range, with a few exceptions.

The IN and OUT broadband traffic distributions are close to a log-normal distribution, which looks like a normal distribution on a semi-log plot. A linear plot would show a long-tailed distribution, with the peak close to the left and a slow gradual decrease toward the right. The OUT distribution is further to the right than the IN distribution, indicating that download volume is more than an order of magnitude larger than upload volume. The peaks of both the IN and OUT distributions for 2021 are further to the right than the peaks of the 2020 distributions, indicating that overall user traffic volumes are increasing. Compared with what we have seen in the past, there is almost no change in the distributions this time around, which is also evident from the lack of growth in total PPPoE traffic.

The peak of the OUT distribution, which appears toward the right in the plot, has been steadily moving rightward over the past few years, but heavy-user usage levels have not increased much, and as a result, the distribution is becoming less symmetric. The IN distribution on the left, meanwhile, is generally symmetric and closer to a log-normal distribution.

Similarly, the peaks of the mobile distributions in Figure 5 have moved slightly to the right, indicating that overall traffic has increased, albeit only slightly. Mobile usage volumes are significantly lower than for broadband, and limits on mobile data usage mean that heavy users, which fall on the right-hand side of the distribution, account for only a small proportion of the total, so the distribution is asymmetric. There are also no extremely heavy users. The variability in each user's daily usage volume is higher for mobile than for broadband owing to there being users who only use mobile data when out of the home/office as well as limits on mobile data. Hence, the daily average for a week's worth of data shows less variability between users than the data for individual days. Plotting the distributions for individual days in the same way results in slightly lower peaks and correspondingly higher tails on both sides, but the basic shape and modal values of the distribution remain largely unchanged.

Table 1 shows trends in the mean and median daily traffic values for broadband users as well as the mode (the most frequent value, which represents the peak of the distribution). When the peak is slightly off the center of the distribution, the distribution is adjusted to bring the mode toward the center. All of the values increased this time around. Comparing 2020 and 2021, the IN mode rose from 158MB

Table 1: Trends in Mean and Mode of Broadband Users' Daily Traffic Volume

	IN(MB/day)			OUT(MB/day)		
Year	Mean	Median	Mode	Mean	Median	Mode
2007	436	5	5	718	59	56
2008	490	6	6	807	75	79
2009	561	6	6	973	91	100
2010	442	7	7	878	111	126
2011	398	9	9	931	144	200
2012	364	11	13	945	176	251
2013	320	13	16	928	208	355
2014	348	21	28	1124	311	501
2015	351	32	45	1399	443	708
2016	361	48	63	1808	726	1000
2017	391	63	79	2285	900	1259
2018	428	66	79	2664	1083	1585
2019	479	75	89	2986	1187	1995
2020	609	122	158	3810	1638	3162
2021	684	136	200	4225	1875	3981

to 200MB and the OUT mode rose from 3,162MB to 3,981MB, translating into growth factors of 1.3 for both IN and OUT.

Meanwhile, because the means are influenced by heavy users (on the right-hand side of the distribution), they are significantly higher than the corresponding modes, with the IN mean at 684MB and the OUT mean at 4,225MB in 2021. The 2020 means were 609MB and 3,810MB, respectively.

For mobile traffic, the mean and mode are close owing to the lack of heavy users. As Table 2 shows, the IN mean has fallen slightly while all other values are up. In 2021, the IN mode was 8MB and the OUT mode was 71MB, while the means were IN 10MB and OUT 86MB. The 2020 modes were IN 7MB and OUT 63MB, and the means were IN 10MB and OUT 79MB.

Figures 6 and 7 plot per-user IN/OUT usage volumes for random samples of 5,000 users. The X-axis shows OUT (download volume) and the Y-axis shows IN (upload volume), with both using a logarithmic scale. Users with identical IN/OUT values fall on the diagonal.

The cluster spread out below and parallel to the diagonal in each of these plots represents typical users with download volumes an order of magnitude higher than upload volumes. For broadband traffic, there was previously a clearly recognizable cluster of heavy users spread out thinly about the upper right of the diagonal, but this is now no longer discernible. Variability between users in terms of usage levels and IN/OUT ratios is wide, indicating that there is a diverse range of usage styles. For mobile traffic, the pattern of OUT being an order of magnitude larger also applies, but usage volumes are lower than for broadband, and there is less variability between IN and OUT. For both

Table 2: Trends in Mean and Mode of Mobile Users' Daily Traffic Volume

	IN(MB/day)			OUT(MB/day)		
Year	Mean	Median	Mode	Mean	Median	Mode
2015	6.2	3.2	4.5	49.2	23.5	44.7
2016	7.6	4.1	7.1	66.5	32.7	63.1
2017	9.3	4.9	7.9	79.9	41.2	79.4
2018	10.5	5.4	8.9	83.8	44.3	79.4
2019	11.2	5.9	8.9	84.9	46.4	79.4
2020	10.4	4.5	7.1	79.4	35.1	63.1
2021	9.9	4.7	7.9	85.9	37.9	70.8

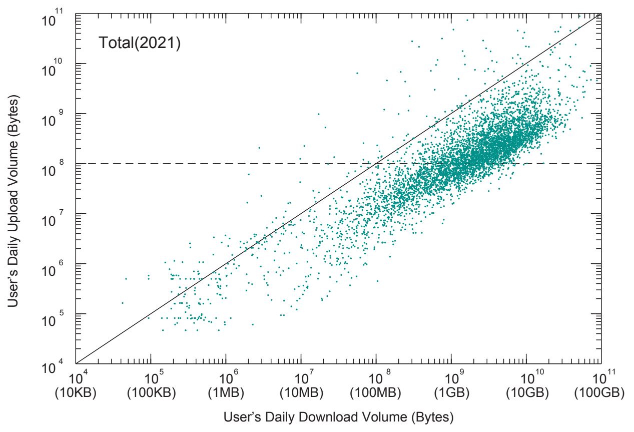


Figure 6: IN/OUT Usage for Each Broadband User

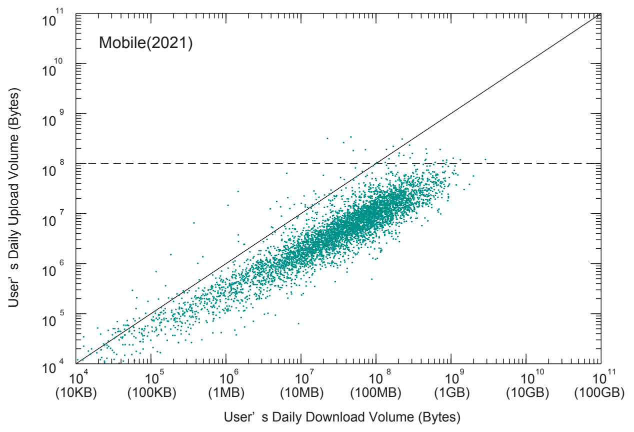


Figure 7: IN/OUT Usage for Each Mobile User

broadband and mobile, there is almost no difference between these plots and those for 2020.

Figures 8 and 9 show the complementary cumulative distribution of users' daily traffic volume. On these log-log plots, the Y-axis values represent the proportion of users with daily usage levels greater than the corresponding X-axis values. These plots are an effective way of examining the distribution of heavy users. The linear-like decline toward the right-hand side of the plots indicates that the distributions are long-tailed and close to a power-law distribution. Heavy users appear to be distributed statistically and do not appear to constitute a separate, special class of user.

The broadband distributions are largely unchanged from last year. But on mobile, the bump observed at the bottom right of the IN distribution last year due to a heavy volume of uploads has disappeared, and the slope is now fairly linear.

Traffic is heavily skewed across users, such that a small proportion of users accounts for the majority of overall traffic volume. For example, the top 10% of broadband users account for 48% of total OUT and 76% of total IN traffic, while the top 1% of users account for 15% of OUT and 50% of IN traffic. The skew has not changed much from last year. As for mobile, the top 10% of users account for 48% of OUT and 49% of IN traffic, while the top 1% account for 12% of OUT and 16% of IN traffic. The skew here is also mostly unchanged from last year's report.

1.4 Usage by Port

Next, we look at a breakdown of traffic and examine usage levels by port. Recently, it has become difficult to identify applications by port number. Many P2P applications use dynamic ports on both ends, and a large number of client/server applications use port 80, which is assigned to HTTP, to avoid firewalls. Hence, generally speaking, when both parties are using a dynamic port numbered 1024 or higher, the traffic is likely to be from a P2P application, and when one of the parties is using a well-known port lower than 1024, the traffic is likely to be from a client/server application. In light of this, we take the lower of the source and destination port numbers when breaking down TCP and UDP usage volumes by port.

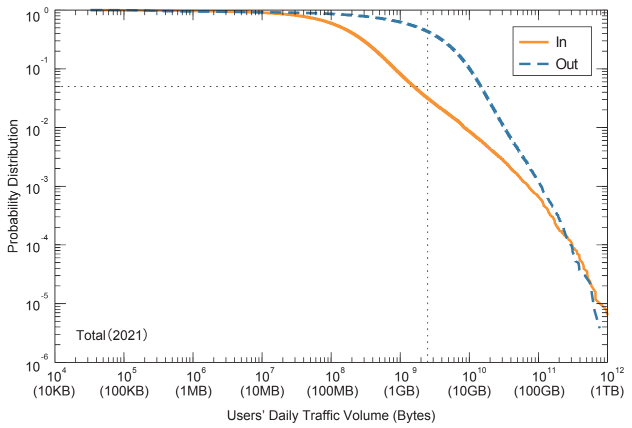


Figure 8: Complementary Cumulative Distribution of Broadband Users' Daily Traffic Volume

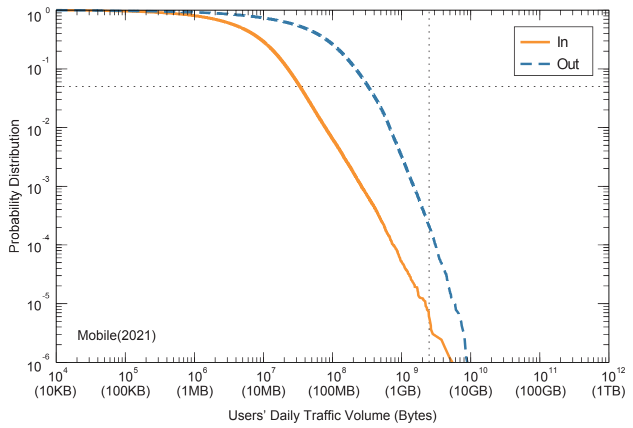


Figure 9: Complementary Cumulative Distribution of Mobile Users' Daily Traffic Volume

Table 3 shows the percentage breakdown of broadband users' usage by port over the past five years. In 2021, 72% of all traffic was over TCP connections, down 5 percentage points vs. 2020. The proportion of traffic over port 443 (HTTPS) was 54%, a 2-point increase from last year. The proportion of traffic over port 80 (HTTP) fell from 17% to 12%. The figure for UDP port 443, which is used by the QUIC protocol, rose from 11% to 16%, so HTTP declined by roughly the amount that QUIC increased.

TCP dynamic port traffic, which has been in decline, fell to 6% in 2021. Individual dynamic port numbers account for only a tiny portion, with the most commonly used port 31000 only making up 0.6%. Port 1935, which is used by Flash Player, makes up 0.2%, but almost all other traffic is VPN related.

Table 4 shows the percentage breakdown by port for mobile users. The figures are close to those for broadband on the whole. This is likely because apps similar to those for PC platforms are now also used on smartphones, and because the proportion of broadband usage on smartphones is rising.

Figure 10 compares overall broadband traffic for key port categories across the course of the week from which observations were drawn in 2020 and 2021. We break the data into four port buckets: TCP ports 80 and 443, dynamic ports (1024 and up), and UDP port 443. The data are normalized so that peak overall traffic volume on the plot is 1. Comparing 2020 and 2021, we see that UDP port 443 has become more prominent than TCP port 80 in 2021. The increase in weekday daytime traffic observed in 2020 is down a little. The overall peak is between 19:00 and 23:00.

Figure 11 shows the trend for TCP ports 80 and 443 and UDP port 443, which account for the bulk of mobile traffic.

Table 3: Broadband Users' Usage by Port

year	2017	2018	2019	2020	2021
protocol port	(%)	(%)	(%)	(%)	(%)
TCP	**83.9**	**78.5**	**81.2**	**77.2**	**71.9**
(< 1024)	72.9	68.5	73.3	70.5	65.8
443(https)	43.3	40.7	51.9	52.4	53.5
80(http)	28.4	26.5	20.4	17.2	11.6
22(ssh)	0.1	0.1	0.2	0.2	0.2
993(imaps)	0.2	0.2	0.3	0.2	0.1
(>= 1024)	11.0	10.0	7.9	6.7	6.1
31000	0.1	0.1	0.2	0.4	0.6
8080	0.3	0.3	0.5	0.4	0.4
1935(rtmp)	1.1	0.7	0.3	0.4	0.2
UDP	**10.5**	**16.4**	**14.1**	**19.4**	**24.5**
443(https)	3.8	10.0	7.8	10.5	15.9
8801	0.0	0.0	0.0	1.1	0.9
4500(nat-t)	0.2	0.2	0.3	0.6	0.8
ESP	**5.1**	**4.8**	**4.4**	**3.2**	**3.3**
GRE	**0.1**	**0.1**	**0.1**	**0.1**	**0.2**
IP-ENCAP	**0.3**	**0.2**	**0.2**	**0.1**	**0.1**
ICMP	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**

Table 4: Mobile Users' Usage by Port

year	2017	2018	2019	2020	2021
protocol port	(%)	(%)	(%)	(%)	(%)
TCP	**84.4**	**76.6**	**76.9**	**75.5**	**70.3**
443(https)	53.0	52.8	55.6	50.7	44.4
80(http)	27.0	16.7	10.3	7.4	5.0
993(imaps)	0.4	0.3	0.3	0.2	0.2
1935(rtmp)	0.2	0.1	0.1	0.1	0.1
UDP	**11.4**	**19.4**	**17.3**	**18.0**	**23.8**
443(https)	7.5	10.6	8.3	9.3	16.3
4500(nat-t)	0.2	4.5	3.0	1.8	3.7
8801	0.0	0.0	0.0	1.4	0.7
3480	0.0	0.0	0.0	0.4	0.3
12222	0.1	2.3	3.4	0.8	0.2
ESP	**0.4**	**3.9**	**5.8**	**6.4**	**5.8**
GRE	**0.1**	**0.1**	**0.0**	**0.1**	**0.1**
ICMP	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**

Comparing the figures with 2020, UDP port 443 has risen further here also, and the lunchtime peak sticks out a bit more. When compared with broadband, we note that mobile traffic levels remain high throughout the day, from morning through night. The plot shows that usage times differ from those for broadband, with three separate mobile traffic peaks occurring on weekdays: morning commute, lunch break, and evening from 17:00 to 22:00.

1.5 Conclusion

Looking back on the situation during the COVID-19 pandemic over the past year and a half, weekday daytime broadband traffic increased substantially from May to March of 2020 as the brakes were put on human movement and a greater proportion of people stayed at home. But excluding this period, traffic volume looks to be rising steadily largely in line with an underlying growth curve. So, although traffic has seen an annual increase of around 40% due to COVID-19, the increase has not been so dramatic as initially feared, and while we do observe ups and downs due to changes in stay-at-home rates associated with COVID case numbers, traffic continues to grow steadily overall.

Also, in contrast with PPPoE, which is subject to bottlenecks and other constraints, IPoE traffic is growing apace and driving growth in broadband traffic overall. The use of IPoE can be expected to continue rising ahead.

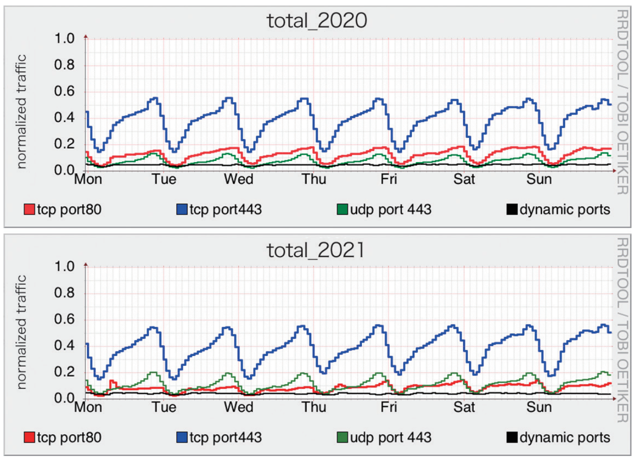


Figure 10: Broadband Users' Port Usage Over a Week 2020 (top) and 2021 (bottom)

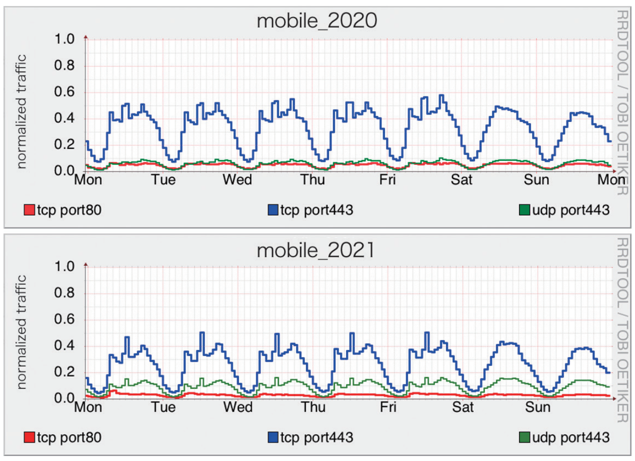


Figure 11: Mobile Users' Port Usage Over a Week 2020 (top) and 2021 (bottom)

Kenjiro Cho
Research Director, Research Laboratory, IIJ Innovation Institute Inc.

Verifiable Credentials and BBS+ Signatures

2.1 Introduction

The concept of self-sovereign identity (SSI) is drawing attention as a new type of digital identity. A digital identity represents who you are in digital space and consists of a collection of attributes such as name, date of birth, gender, and email address[*1]. Digital identities have traditionally been managed by applications, enterprise systems, or identity providers such as GAFAM. The idea of SSI is to allow the owner of an identity to independently manage the identity.

Two years have passed since we discussed SSI in IIR Vol. 43[*2]. Over that time, the technologies and mechanisms needed to make SSI a reality have continued to advance; this includes Verifiable Credentials (VC), Decentralized Identifiers (DIDs), digital agents, digital wallets, and governance frameworks. These technologies are comprehensively explained in other documents[*3][*4], and here we limit our focus to providing an overview of VCs, which can be considered the core of SSI. We also briefly discuss the implementation of VCs using BBS+ signatures, which have been attracting interest in the community since last year.

2.2 Credentials and Verifiable Credentials

The term "credentials" can mean various things depending on context[*5], but here we refer to the World Wide Web Consortium (W3C) specification[*6] and use the term to mean "a set of one or more claims about a subject made by an issuer". For example, a driver's license is a type of credential in that it is a set of claims (e.g., the holder's name, address, date of birth, photograph, types of vehicles that can be driven) that the issuer (e.g., department of motor vehicles) makes about the subject (i.e., the license holder).

Credentials allow us to have the credential issuer vouch for who we are. For example, if I were trying to open a bank account, the bank teller would not trust me if I simply claimed, without any evidence, that "I am a male living in ABC City and my birthdate is XYZ". Showing my driver license, which serves as my credentials, in this case means that the license issuer certifies my claims, and this enhances the credibility of my claims[*7].

For a claim made using credentials to be accepted, however, the credentials must be credible to the party to which the claim is made. So who issued the credentials? Have the credentials been rewritten or forged by someone else? Have they expired or been revoked? Only once these things are verified and validated can the information in the credentials be accepted.

With physical credentials, the verifier looks at the information on the document and determines its authenticity by checking any special printing, such as watermarks, if present. This sort of verification process is often difficult and requires specialized skill.

VCs are digitalized credentials, so they can be verified by a computer. This does not mean that the credentials document is simply scanned into a digital image. Digital signatures are used to verify the issuer's identity and whether or not the document has been tampered with. This approach is based on the results of cryptographic research into what's known as anonymous credentials or attribute-based credentials, and it can also accommodate privacy-enhancing mechanisms using zero-knowledge proof technology.

*1 ISO/IEC 24760-1 defines identity as "a set of attributes related to an entity".

*2 Internet Infrastructure Review Vol. 43, "2. Blockchain-based Identity Management and Distribution" (https://www.iij.ad.jp/en/dev/iir/043.html).

*3 Alex Preukschat and Drummond Reed, "Self-Sovereign Identity - Decentralized digital identity and verifiable credentials", Manning Publications, May 2021 (https://www.manning.com/books/self-sovereign-identity).

*4 Kengo Suzuki and Kento Goro, "Identity wa dare no mono? Hyperledger Indy & Aries de jitsugen suru bunsan identity" [Who do identities belong to? Decentralized identities made possible by Hyperledger Indy & Aries] Impress R&D, May 2021, (https://nextpublishing.jp/isbn/9784844379447, in Japanese).

*5 Internet Infrastructure Review Vol. 26 "1.4.3 ID Management Technology" (https://www.iij.ad.jp/en/dev/iir/026.html).

*6 Verifiable Credentials Data Model 1.0 (https://www.w3.org/TR/vc-data-model).

*7 A driver's license is first and foremost a credential to show that you are qualified to drive. Driver's licenses are also commonly used as a form of ID since they contain a set of key attributes, such as name, address, gender, date of birth, and face photo. The government has also identified integrating driver's licenses with Japan's Individual Number Cards as a goal.

2.3 Illustration of Verifiable Credentials in Use

To provide a more tangible idea of how VCs are used, let's imagine a world in which certificate of residence (a common identification document in Japan) are represented as VCs and consider what happens from issuance through to the use of these credentials.

Mr. A, who lives in X City, decides to sign up for a family account on a service provided by Company B. Company B offers a discount to residents of X City. To receive the discount, Mr. A needs to show that he and his family live in X City.

So Mr. A visits the X City residential services office and asks for a VC version of his certificate of residence.

X City residential services verify Mr. A's identity using an appropriate method. They may, for example, ask him to present a photo ID in person or to provide other VCs online.

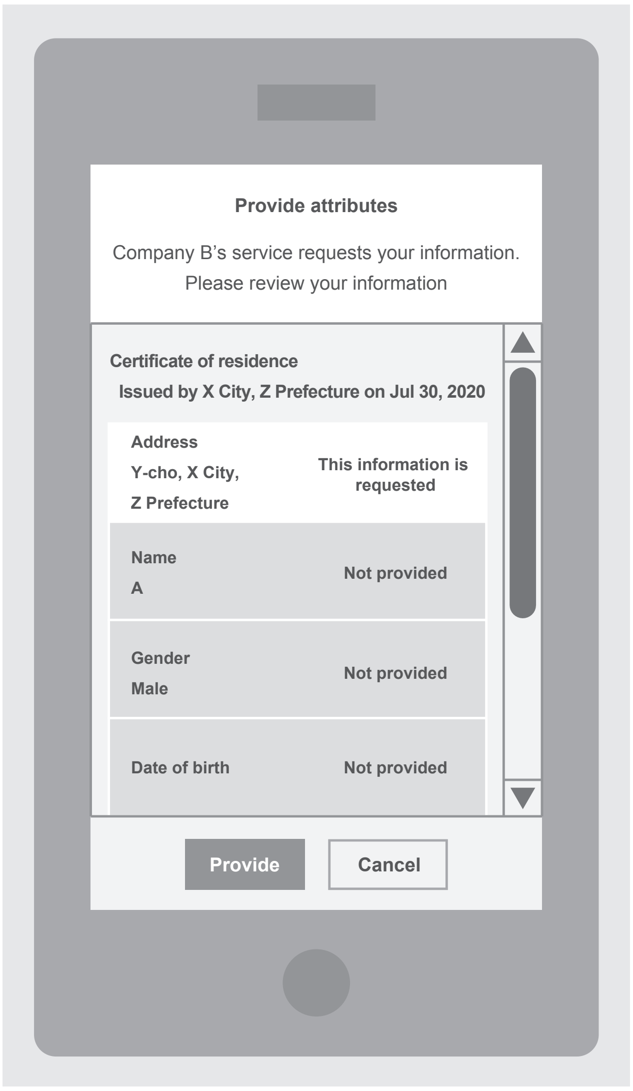
Upon confirming Mr. A's identity, X City residential services obtain the attributes of Mr. A and his family from the resident information database and issue a VC, which is equivalent to a certificate of residence and contains the attributes of Mr. A and family, including address, name, date of birth, gender, and date resident status was obtained. Mr. A stores the VC in his smartphone.

Mr. A then applies to Company B for the service. By presenting the VC issued by X City to Company B, Mr. A can show that he and his family reside in X City.

Both Mr. A and Company B want to exchange only a minimum of personal information. So, Mr. A presents the credentials to Company B with only the address of Mr. A and family disclosed (i.e., selective disclosure) and the rest of the information hidden (name, gender, date of birth, and date of residential status). Figure 1 illustrates this. In this example, we assume that Company B has specified what attributes it needs (address), but it is also possible for Mr. A to choose which attributes are provided.

Company B verifies the credentials and confirms that Mr. A and his family reside in X City, as asserted by X City. This allows Mr. A and his family to use Company B's service at a discounted price.

Note that the VC Mr. A received from X City is not specific to Company B's services. For example, Mr. A can subsequently show some other company—call it Company C—that he lives in X City or perhaps that members of his family are over 20 years old. It is also envisioned that, in addition to using a single VC, people will be able to combine multiple VCs to provide the desired attributes.


Provide attributes

Company B's service requests your information.
Please review your information

Certificate of residence
Issued by X City, Z Prefecture on Jul 30, 2020

Address Y-cho, X City, Z Prefecture	This information is requested
Name A	Not provided
Gender Male	Not provided
Date of birth	Not provided

Figure 1: Illustration of Providing Credentials

2.4 The Verifiable Credentials Ecosystem

In the example above, we encountered X City, which issued the VC, Mr. A, who received the VC and presented it to another party, and Company B, which verified the credentials presented. The actors in a VC scenario and the relationships between them are called the Verifiable Credentials ecosystem, which can be laid out as in Figure 2.

The issuer is the person or organization that issues the VC. In the previous example, this is X City.

The holder receives the VC issued by the issuer and stores it in his/her smartphone or other device. The holder then presents the VC to verifiers as needed. In the example above, Mr. A was the holder of a VC version of his certificate of residence.

The subject is an entity about which the VC makes claims. In most cases, the subject is the same as the holder, but they can be different entities in some cases, such as when the subject is an infant and the holder is the infant's guardian. In the example above, the subject encompasses Mr. A as well as his family members.

The verifier is the person or organization that verifies the VC presented by the holder and uses the information it contains. In the example above, this is Company B and Company C.

The Verifiable Data Registry is data storage used by the issuers, holders, and verifiers. It records information required for performing verification, such as the issuer's identifier and public key and credential revocation registries. Anyone can refer to this information, but it cannot be altered. As such, it is often implemented on a blockchain.

VCs and self-sovereign identity are often mentioned in conjunction with blockchain, and it is common to think that VC itself is recorded on a blockchain, but this is a misconception. The Verifiable Data registry is a registry that anyone can refer but not alter, so it is not considered an appropriate place for VCs containing personal data[*8].

As the previous example illustrates, the two major VC events occur when the issuer issues credentials and when the holder presents them to a verifier. The holder asks the issuer to issue credentials and is thus issued with a VC that contains the subject's attributes. The holder saves this on her smartphone or other device and later presents only the necessary parts to verifiers, who then verify the credentials. The result is that the verifier is able to confirm that the subject has the attributes as certified by the issuer.

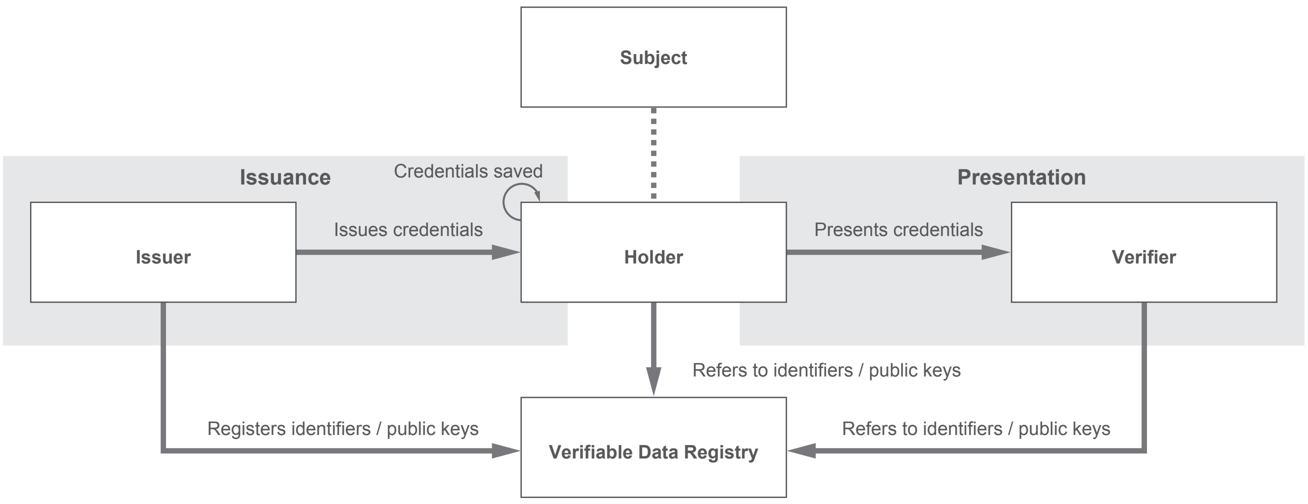


Figure 2: Verifiable Credentials Ecosystem

*8 Stephen Curran, "Why Distributed Ledger Technology (DLT) for Identity?", Hyperledger (https://www.hyperledger.org/blog/2021/04/21/why-distributed-ledger-technology-dlt-for-identity).

2.5 How Verifiable Credentials and Traditional Digital Certificates Differ

Credentials that use digital signatures are actually not a new concept. Digital certificates are constantly being verified when we communicate via HTTPS on a daily basis. OpenID Connect, which is often used for digital identity federation, also stores identity information in digitally signed ID tokens to facilitate verification. In that sense, these digital certificates and tokens also serve as verifiable credentials.

So how do VCs and traditional digital certificates differ? We see three main points. Not all VCs check all of these points, but we seem to call credentials VCs when at least one of these characteristics is present.

1. Has a mechanism for providing only the minimum of data required
2. A holder is always present between the issuer and the verifier
3. Uses a decentralized identifier (DID)

Let’s start with the first point. Many VCs have a mechanism for minimizing the data that the credential holder discloses. One of the most notable is the use of a cryptographic technique called zero-knowledge proofs. A zero-knowledge proof allows the holder to present only the attributes in credential that the verifier requires while keeping other attributes hidden. It is also possible to disclose only the fact that the hidden attributes satisfy certain conditions. For example, the holder can hide the name, address, and date of birth on a driver’s license while also showing that he is qualified to drive a standard automobile and that he is at least 20 years of age. This sort of mechanism is key to protecting the privacy of the holder and subject.

The second point also has to do with protecting the holder’s privacy. If we consider the issuer to be the Identity Provider (IdP) and the verifier to be the Relying Party (RP), then the VC mechanism can be seen as similar to existing identity federation mechanisms such as OpenID Connect and SAML. VCs differ from these standards in that they do not allow direct interaction between the issuer and the verifier; there is always a holder between the two. This aspect of VCs is one reason they play a central role in SSI. It is useful because the holder may not want the issuer and verifiers to know his every move in terms of what information he has provided to what sort of providers and when.

The third point relates to decentralized identifiers (DIDs), which, along with VCs, are the cornerstone of SSI. DIDs are identifiers that can refer to people, organizations, and things, and they are associated with a public key that is needed to verify the digital signature. The association between the DID and the public key is guaranteed in a decentralized manner using blockchain or the like without the need for a trusted third party such as a registration authority. One does not need to use DIDs to realize the benefits of VCs, but they are often used together to unlock the advantages of both in tandem.

2.6 Developments in Verifiable Credentials

Vaccination certificate implementations that use these characteristics of VCs and other initiatives are being trialled.

In April 2020, the COVID-19 Credentials Initiative (CCI) was launched to enable the application of VCs to facilitate the interoperable use of privacy-preserving digital credentials for COVID-19-related purposes[*9]. The CCI has now joined Linux Foundation Public Health (LFPH)[*10]. In June 2021, LFPH launched the Global COVID Certificate Network (GCCN), a cross-border initiative for the exchange of vaccination certificates[*11]. Meanwhile, in January 2021, Microsoft, Oracle, Salesforce, and others also launched the COVID-19 Credentials Initiative (CCI), which is working to digitalized vaccine certificates based on VCs[*12].

Similarly, the European Digital Identity Framework unveiled by the European Commission in June 2021 put forward the concept of a Digital Identity Wallet usable by all citizens and residents of EU member states. Although it does not specifically mention the use of VCs and SSI, the heavy influence of VCs is apparent given that the model and use cases comprise issuers, holders, and verifiers and that holders can selectively disclose attributes[*13].

The use of VCs is also expanding to include other areas, with examples being eKYC (online know your customer) for microfinance by the NPO Kiva[*14] and the IATA Travel Pass[*15] from the International Air Transport Association (IATA).

In Japan, Keio University, together with five Japanese companies and in cooperation with Microsoft, commenced demonstration testing of student identity system that uses VCs and DIDs in October 2020[*16]. And in a March 2021 white paper, the Trusted Web Promotion Council mentions VCs as one of the building blocks for realizing trustable communication[*17].

A slew of products supporting such use cases is being developed. The Linux Foundation's Hyperledger project is heavily engaged in developing a range of technologies, with a particular focus on Hyperledger Indy[*18], a distributed ledger for providing DIDs, Hyperledger Aries[*19], an agent for handling VCs, and Hyperledger Ursa[*20], a cryptographic library for use by these projects. Azure AD, Microsoft's Identity as a Service (IDaaS) offering, also includes VC functionality and has been in public preview since April 2021[*21].

---

*9 CCI (COVID-19 Credentials Initiative) (https://www.covidcreds.org/).

*10 LFPH (Linux Foundation Public Health) (https://www.lfph.io/).

*11 Introducing the Global COVID Certificate Network (GCCN) (https://www.lfph.io/2021/06/08/gccn/).

*12 Vaccination Credential Initiative (VCI) (https://vaccinationcredential.org/).

*13 European Digital Identity - European Commission (https://ec.europa.eu/info/strategy/priorities-2019-2024/europe-fit-digital-age/european-digital-identity_en).

*14 Kiva Protocol, built on Hyperledger Indy, Ursa and Aries. Sierra Leone has adopted this protocol and built a platform that allows people perform identity verification in around 11 seconds for the purpose of small-scale financing. (https://www.hyperledger.org/blog/2021/01/20/kiva-protocol-built-on-hyperledger-indy-ursa-and-aries-powers-africas-first-decentralized-national-id-system).

*15 IATA - Travel Pass Initiative (https://www.iata.org/en/programs/passenger/travel-pass/).

*16 "Keio University Commences Demonstration Experiment of Next-Generation Digital Identity Platform: To Issue Certificates of Enrollment and Certificates of Expected Graduation to Smartphone Applications" (https://www.keio.ac.jp/en/press-releases/2020/Nov/13/49-76286/).

*17 Trusted Web White Paper ver 1.0 (https://www.kantei.go.jp/jp/singi/digitalmarket/trusted_web/pdf/documents_210331-2.pdf, in Japanese).

*18 Hyperledger Indy (https://www.hyperledger.org/use/hyperledger-indy).

*19 Hyperledger Aries (https://www.hyperledger.org/use/aries).

*20 Hyperledger Ursa (https://www.hyperledger.org/use/ursa).

*21 Identity verification solutions - Microsoft Security (https://www.microsoft.com/en-us/security/business/identity-access-management/verifiable-credentials).

2.7 Verifiable Credentials Implementations

While W3C is working to standardize VCs, this standardization effort is focused on the data model. Specific details vary widely from implementation to implementation. An explanatory document[*22] by CCI and LFPH refers to these variations in implementation as “flavors”.

Here, we look at JSON-LD ZKP with BBS+, a flavor that has attracted a lot of attention at the Internet Identity Workshop (IIW)[*23] and in related circles. JSON-LD ZKP with BBS+ is a relatively new scheme that was unveiled by New Zealand-based company MATTR at the April 2020 IIW. It has been well received by the community[*24], and non-MATTR engineers are now also involved in developing and discussing the scheme’s standard as part of the W3C Credentials Community Group (CCG)[*25] and the Decentralized Identity Foundation’s (DIF) Crypto Working Group[*26]. It is being developed in the open on GitHub[*27], where we have also made a few contributions.

Key aspects of JSON-LD ZKP with BBS+ are that it uses the JSON-LD format to encode credentials, and it uses BBS+ signatures, which work well with zero-knowledge proofs, as the digital signature scheme.

The JSON-LD specification is not as well known as JWTs (JSON Web Tokens) in a digital identity context, but it is widely used in the Semantic Web and Search Engine Optimization (SEO) domains. An advantage of JSON-LD is that it incorporates Linked Data elements into JSON data and can thereby uniquely identify the terms used to describe data using URIs while retaining the compactness of JSON. Metadata in JSON-LD format is embedded in many websites these days. Figure 3 shows an example of a credential represented in JSON-LD.

BBS+ signatures are multi-message digital signatures[*28][*29][*30] that extend BBS group signatures[*31]. They are a type of elliptic curve cryptography that uses an operation called

```
{
    "@context": [                                 // JSON-LD context
        "https://www.w3.org/2018/credentials/v1",
        "https://schema.org",
        ...
    ],
    "id": "http://example.edu/creds/1234",        // Credential identifier
    "type": "VerifiableCredential",               // Credential type
    "issuer": "https://example.edu/issuers/1",    // Credential issuer
    "issuanceDate": "2021-06-22T00:00:00Z",       // Credential issue date
    "expirationDate": "2022-06-22T00:00:00Z",     // Credential expiry date
    "credentialSubject": {
        "id": "did:example:ebfeb1f712ebc6f1c276e12ec21",  // Subject identifier
        "type": "Person",                                 // Subject type
        "birthDate": "1970-01-01",                        // Subject DOB
        "name": "John Smith",                             // Subject name
        ...                                               // Other attributes
    },
    "proof": { ... }                              // Signature value needed for verification etc.
}
```

Figure 3: Example of JSON-LD Credentials

*22 A Path Towards Interoperability: CCI Released a Paper on Different Flavors of Verifiable Credentials (https://www.lfph.io/2021/02/11/cci-verifiable-credentials-flavors-and-interoperability-paper/).

*23 Internet Identity Workshop (https://internetidentityworkshop.com/).

*24 Why the Verifiable Credentials Community Should Converge on BBS+ (https://www.evernym.com/blog/bbs-verifiable-credentials/).

*25 BBS+ Signatures 2020, W3C Community Group Draft Report (https://w3c-ccg.github.io/ldp-bbs2020/).

*26 DIF - Applied Crypto Working Group (https://identity.foundation/working-groups/crypto.html).

*27 mattrglobal/jsonld-signatures-bbs: A linked data proof suite for BBS+ signatures (https://github.com/mattrglobal/jsonld-signatures-bbs/).

*28 Jan Camenisch and Anna Lysyanskaya, “Signature Schemes and Anonymous Credentials from Bilinear Maps”, CRYPTO 2004 (http://dx.doi.org/10.1007/978-3-540-28628-8_4).

*29 Man Ho Au, Willy Susilo, and Yi Mu, “Constant-Size Dynamic k-TAA”, SCN 2006 (http://dx.doi.org/10.1007/11832072_8).

*30 Jan Camenisch, Manu Drijvers, and Anja Lehmann, “Anonymous Attestation Using the Strong Diffie Hellman Assumption Revisited”, Trust 2016 (http://dx.doi.org/10.1007/978-3-319-45572-3_1).

*31 Dan Boneh, Xavier Boyen, and Hovav Shacham, “Short Group Signatures”, CRYPTO 2004 (http://dx.doi.org/10.1007/978-3-540-28628-8_3).

pairing. They differ from the commonly used RSA and ECDSA signatures in that it is possible to sign a list of multiple pieces of data (rather than a single piece of data). The structure also makes it easy to combine with zero-knowledge proof technology, so you can verify a signature as being valid while still hiding some elements in the list of signed data, and you can hide selected elements while still providing proof that they meet some criteria.

JSON-LD ZKP with BBS+ canonicalizes credentials represented in JSON-LD into a data form called statements using LD canonicalization. BBS+ signatures are then used to sign and verify the list of statements. For example, the JSON-LD credentials in Figure 3 are converted into a list of statements as shown in Figure 4 and then signed. Using BBS+ signatures to sign the list of statements allows you to control whether each particular statement is shown or not. It is not yet possible, however, to provide high-level proofs showing that a particular value within a statement (name, date of birth, etc.) satisfies certain conditions (e.g., date of birth falls within a certain range) while keeping that value hidden.

2.8 The Future of Verifiable Credentials

Some issues remain to be resolved before VCs and JSON-LD ZKP with BBS+ can be put to practical use. Here, we go over three key issues and look at approaches and efforts aimed at solving them.

■ Issue 1: Interoperability with existing digital identity technologies

The first challenge is ensuring interoperability between the new concept that VCs represent and existing digital identity specifications and products. The OpenID Foundation (OIDF) is looking at addressing this by using the Self-Issued OpenID Provider (SIOP) framework, which is originally part of OpenID Connect, to handle VCs on top of OpenID Connect. Engineers from MATTR, the original proponent of JSON-LD ZKP with BBS+, are involved in this work.

■ Issue 2: Standardizing the various specifications

The JSON-LD ZKP with BBS+ and LD canonicalization specifications mentioned above are still being discussed and not yet finalized as standard specifications. In the case of JSON-LD ZKP with BBS+, the W3C CCG is developing the specification and, in parallel with this, DIF's Crypto Working Group is also holding discussions, as mentioned earlier. The details are being standardized as W3C specifications as they are finalized, with future details to be discussed and worked out by the DIF's Crypto Working Group. For example, the means of making high-level proofs possible, such as showing that a person is 20 years or older while keeping date of birth hidden, is on the DIF Crypto Working Group's agenda for discussion. As for LD canonicalization, the W3C's Linked Data Signatures Working Group currently being set up is expected to pursue work on this in the form

```
<did:example:ebfeb1f712ebc6f1c276e12ec21> <http://schema.org/birthDate> "1970-01-01"^^<http://schema.org/Date> .
<did:example:ebfeb1f712ebc6f1c276e12ec21> <http://schema.org/name> "John Smith" .
<did:example:ebfeb1f712ebc6f1c276e12ec21> <http://www.w3.org/1999/02/22-rdf-syntax-ns#type> <http://schema.org/Person> .
<http://example.edu/creds/1234> <http://www.w3.org/1999/02/22-rdf-syntax-ns#type> <https://www.w3.org/2018/credentials#VerifiableCredential> .
<http://example.edu/creds/1234> <https://www.w3.org/2018/credentials#credentialSubject> <did:example:ebfeb1f712ebc6f1c276e12ec21> .
...
```

Figure 4: List of Signed Statements (Excerpt). Each Line is Called a Statement.

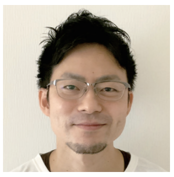
of RDF Dataset Canonicalization (RDC) and Linked Data Integrity (LDI). As of this writing (August 2021), the aim according to the Proposed Charter[*32] is to begin work in September 2021 and produce a W3C Recommendation by September 2023, or within two years.

■ Issue 3: Resilience to quantum computing

The third issue to highlight, and a long-term one, is that of post-quantum cryptography, which we also covered back in IIR Vol. 49[*33]. The security of BBS+ signatures relies on the hardness of the discrete logarithm problem on elliptic curves. It is known that quantum computers will be able to efficiently solve this problem. So, unfortunately, BBS+ signatures and JSON-LD ZKP with BBS+, which uses them, are not quantum resistant. The same goes for the Camenisch-Lysyanskaya (CL) signatures used in Hyperledger Indy as well as the RSA, ECDSA, and EdDSA signatures often used in JWTs. Post-quantum anonymous credentials based on lattice-based signature schemes and Zero-Knowledge Scalable Transparent Arguments of Knowledge (ZK-STARK) have also been proposed, but much room for improvement, including performance enhancements, remains before they become practically viable.

2.9 Conclusion

We have looked at the current status of and future issues for VCs, a topic that continues to gain attention, and one of the implementations in the form of the JSON-LD ZKP with BBS+ flavor. Personally, I expect VCs to be used as and when appropriate rather than completely replacing conventional digital certificates and ID tokens. The real value of VCs is evident in situations where the privacy of people, organizations, and things must be protected, particularly when there is a need to minimize what data is provided. And VCs that use JSON-LD make possible credential statements with strong expressive power and interoperability, facilitating digital identity bridging across a wide range of organizations and industries. Many issues remain to be resolved before VCs are used in real-world applications, but we will be keeping an eye on efforts to standardize and popularize their use, and we hope to make our own contributions toward the development of the community in this area as well as society as a whole.



Dan Yamamoto

Senior Engineer, Office of Emergency Response and Clearinghouse for Security Information, Advanced Security Division, IIJ.
Dr. Yamamoto began his current role in 2021. He investigates and researches digital identity and information security issues.

*32 Linked Data Signatures Working Group Charter (https://w3c.github.io/lds-wg-charter/index.html).

*33 Internet Infrastructure Review Vol. 49, "Trends in Post-Quantum Cryptography—2020" (https://www.iij.ad.jp/en/dev/iir/049.html).

Implementing QUIC in Haskell

One of IIJ's goals is to contribute to the development of the Internet, and one way our lab does this is through its involvement in standardizing new protocols. For years, we have been helping to develop more complete specifications. Our work involves discussing new protocol specifications, implementing those specifications, and testing interoperability with other implementations.

Since 2013, I have participated in the standardization of HTTP/2 and TLS 1.3. Over the last two and a half years, I have been involved in the standardization of QUIC and HTTP/3, which are closely related to these two protocols. In this report, I explain how I implemented QUIC and HTTP/3.

3.1 QUIC and HTTP/3

QUIC is a new transport protocol that uses UDP. It is defined as a large specification incorporating the following features.

- Reliability, flow control, and congestion control provided by TCP
- Multiplexing with asynchronous streams derived from HTTP/2 (stream fragmentation and reassembly)
- Security features provided by TLS 1.3 (key exchange, authentication of peers, encryption of data)

The basic units in QUIC are called *packets*. A packet can contain multiple units of data called *frames*. There are several types of frames: e.g., application data is stored in STREAM frames, and ACK (acknowledgement) information is stored in ACK frames. HTTP as defined in the QUIC protocol is called HTTP/3.

3.2 Why Implement it in Haskell?

As with my implementations of HTTP/2 and TLS 1.3, I am implementing QUIC and HTTP/3 in the Haskell programming language. My reasons for choosing Haskell are as follows.

- Rich data types allow for concise problem representation, and strong type checking can detect many coding errors.
- Lightweight threads are provided as standard, enabling threaded programming with better code readability than with event-driven programming and a small overhead when switching and creating threads, where state management tends to be cumbersome (any reference to threads below means lightweight threads).
- Many data types are immutable and can be safely shared between threads.
- STM (Software Transactional Memory) is provided as standard, enabling threaded programming without deadlocks.

Most of the QUIC implementations by other teams use event-driven programming, whereas I use threaded programming. I feel that threaded programming not only improves code readability but also allows me to test specifications from a different perspective than other implementers.

Below, I describe specific implementation points.

3.3 QUIC Streams and Connections

QUIC divides communications into streams in order to multiplex within a single connection. HTTP/2 uses streams for the same purpose, but while HTTP/2 streams can only carry HTTP requests and responses, QUIC streams can carry data for any application.

After working on a QUIC API for quite a while, I discovered the following abstractions.

- The role of QUIC connections corresponds to that of network I/O management handled by the OS.
- QUIC streams correspond to TCP connections.

TCP connections here means the simplest form of TCP connections that only exchange one piece content, as in HTTP/1.0. Viewing things from this angle, I realized that streams can be controlled with an API that mimics the socket API. Part of the current API appears below.

Haskell type annotations are separated by a right arrow. The return type is on the far right. The other parts of the type signature are the argument types. When `IO` appears to the left of the type, it means the method has side effects, such as input and output operations. When `IO` does not appear, the data type is immutable and has no side effects. `()` denotes that there is no return value, and `ByteString` is, of course, the byte string type. So, `IO ()` means that there is no meaningful return value and that only the function's side effects are of interest.

```
-- Abstract data type representing a stream
data Stream

-- Function for creating streams
stream :: Connection -> IO Stream

-- Function for closing streams
closeStream :: Stream -> IO ()

-- Function that accepts streams created by peers
acceptStream :: Connection -> IO Stream

-- Function that receives data from streams
recvStream :: Stream -> Int -> IO ByteString

-- Function for sending data to streams
sendStream :: Stream -> ByteString -> IO ()
```

When implementing an HTTP/1.0 server in Haskell, the usual convention is to use a synchronous approach of starting one thread for each TCP connection from a client, reading a request, writing a response, and then terminating the thread. In HTTP/2, you need to manage multiple threads to enable multiplexing. When implementing HTTP/3, the QUIC library handles this multiplexing. So, when using the above API, it is possible to use the conventional synchronous approach of starting one thread per stream.

3.4 Accepting Connections on a Server

The type annotation of the function that starts a server is as follows.

```
run :: ServerConfig -> (Connection -> IO ()) -> IO ()
```

That is, `run` takes a server configuration and a server application function (a function that receives a connection and does some processing, including input and output) as arguments. The Dispatcher thread launched by this function opens a listening (wildcard) socket for each network interface. When a new connection is accepted, the threads that make up the connection are started (see Section 3.5).

There are six types of QUIC packets. The body of Initial packets, 0-RTT packets, Handshake packets, and 1-RTT packets is encrypted and the header is protected. The body of Version Negotiation packets and Retry packets is not encrypted, nor is the header protected. To analyze these packets in a consistent, unified manner, I devised a method of dividing the analysis into two stages.

(1) Parse parts of the header that are not protected (determine the packet type etc.)
(2) Decrypt encrypted text and remove header protection

Stage (1) is performed by the Dispatcher thread. The Dispatcher thread looks at the results of the analysis in (1) and creates a new connection if it is an Initial packet, or performs the migration process if it is an appropriate 1-RTT packet (see Section 3.9). The specification does not allow the server to accept Version Negotiation packets or Retry packets, so these are simply discarded.

Stage (1) is also performed by the Reader thread described below, and (2) is performed by the Receiver thread described below. The two-stage analysis idea has yielded a common data structure for the header information, resulting in more concise code than in earlier implementations.

3.5 Threads that Make up a Connection

When starting a new connection, the Dispatcher thread starts the main thread for that connection and asks it to create the connection. The main thread starts a group of threads that make up the connection, as shown in Figure 1, and waits for them to finish.

When the connection is created, a connected socket is created. So packets for this connection are read by the Reader thread, not the Dispatcher thread. The Reader performs the packet analysis in (1) above, and passes the parsed header information, protected header, and encrypted body to the Receiver thread through the queue (`RecvQ`).

The Receiver thread performs (2) above, extracts the packet's frames, and processes each of them. STREAM frames are reassembled and passed to the Server thread through the queue (`InputQ`). When it receives an ACK frame, the Receiver thread deletes the corresponding information from the information-retransmission container (`SentPackets`) described in Section 3.10.

The Server thread is what invokes the server application function. The output is sent to the Sender thread through a queue (`OutputQ`). The Server thread is responsible for

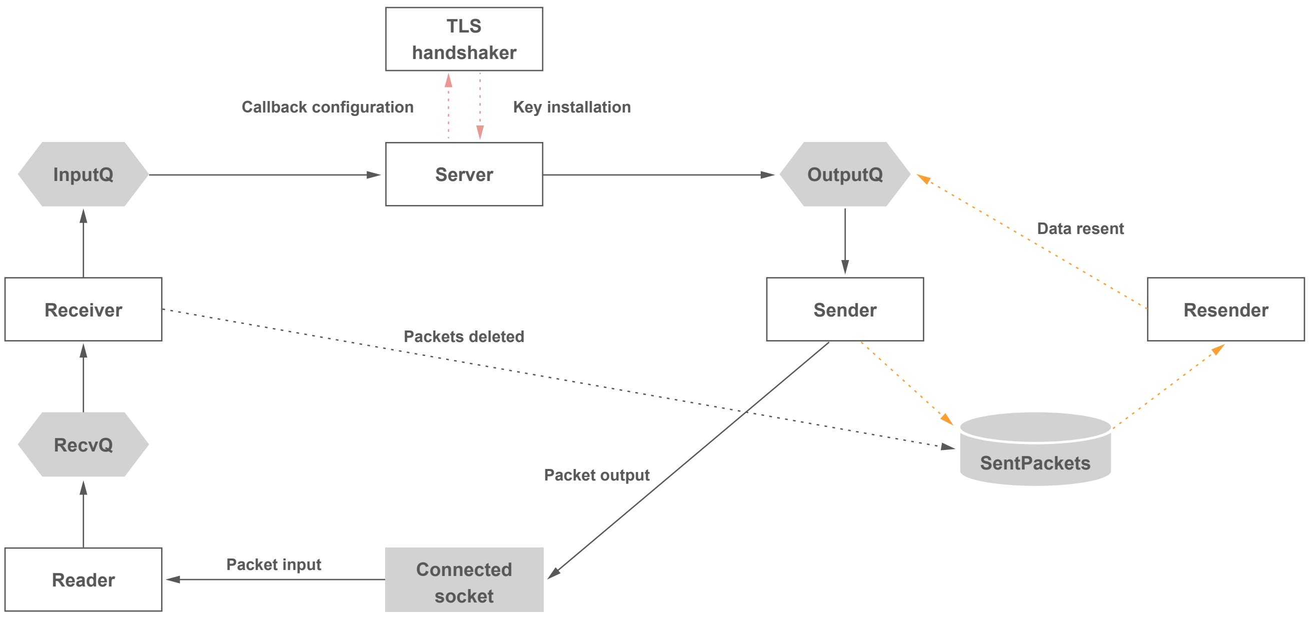


Figure 1: Threads that Make up a Connection

launching the TLS handshaker thread to perform key exchange and synchronize key availability timing before launching the application.

The Server thread is what invokes the server application function. The output is sent to the Sender thread through a queue (OutputQ). The Server thread is responsible for launching the TLS handshaker thread to perform key exchange and synchronize key availability timing before launching the application.

When the Resender thread detects a packet loss, it retrieves the relevant information from the information-retransmission container and resends it by putting it into the OutputQ.

STM is used for the queues and other data sharing, so these threads do not deadlock. If any one thread causes a fatal error, the entire thread group terminates. When this happens, resources are properly released and no leaks occur.

3.6 Connected Sockets

TCP lets you generate a connected socket from a wildcard socket using the accept() system call. The accept() system call cannot be used with UDP, however.

For example, suppose your server has a wildcard socket {UDP, 192.0.2.1, 443, *, *} and a client requesting a connection on 203.0.113.1:3456. The connected socket you want to generate is {UDP, 192.0.2.1, 443, 203.0.113.1, 3456}. A simple way to do this is as follows.

(1) Open a new UDP socket and set the SO_REUSEADDR option.
(2) Call the bind() system call with 192.0.2.1:443.
(3) Call the connect() system call with 203.0.113.1:3456.

Unfortunately, on BSD-based OSs, (2) causes an error. Linux allows (2), but race conditions can occur. These problems can be solved as follows.

(1) Open a new UDP socket and set the SO_REUSEADDR option.
(2) Call the bind() system call with *:443.
(3) Call the connect() system call with 203.0.113.1:3456. In this case, the local address is set to 192.0.2.1.

This method works fine on many operating systems and does not cause race conditions. However, you need to be careful with privileges. Suppose that, in TCP, a process with root privileges creates a wildcard socket for a privileged port. Even if this process relinquishes root privileges for security reasons, the accept() system call can still be executed. Linux, however, requires the process to at least have the CAP_NET_BIND_SERVICE capability to generate a UDP-connected socket using the above method.

3.7 Closing Connections

When using TCP with the socket API, a call to the `close()` system call by the application immediately returns control to the application, and the OS is then responsible for subsequently terminating TCP. The QUIC implementation also needs to enable this sort of control.

In my implementation, when the server (or client) application function terminates, all threads except the main thread terminate and unnecessary information is discarded. The main thread also starts a separate thread to handle the termination procedure with the minimum information needed to resend the CONNECTION _ CLOSE frame if need be.

In QUIC, an ACK is not returned for packets that contain a CONNECTION _ CLOSE frame. Once the peer has received a CONNECTION _ CLOSE frame, it immediately stops sending packets. So after sending the CONNECTION _ CLOSE frame, we wait a while to make sure that no more packets will arrive from the peer. If packets do arrive, this may indicate that the CONNECTION _ CLOSE frame has been lost, so the packet with the CONNECTION _ CLOSE frame is resent.

3.8 TLS Handshake

QUIC uses TLS 1.3 to perform handshakes to authenticate peers and exchange keys. TLS 1.3 messages are detached from the TLS record layer and stored in a simple data format in CRYPTO frames.

Figure 2 illustrates a full handshake in QUIC.

The client generates Initial keys based on the randomly generated connection ID. The TLS 1.3 ClientHello message is put into a CRYPTO frame, which is then put into the Initial packet, which is encrypted using the Initial key and sent. Note that privacy is not protected because Initial keys can also be generated on intermediate devices.

Upon receiving this, the server generates the Initial key and decrypts the Initial packet. Next, it generates the Handshake key and 1-RTT key based on the retrieved ClientHello. It then puts the generated ServerHello into the Initial packet, encrypts it with the Initial key, and sends it. Other TLS messages are put into Handshake packets and encrypted with the Handshake key before being sent.

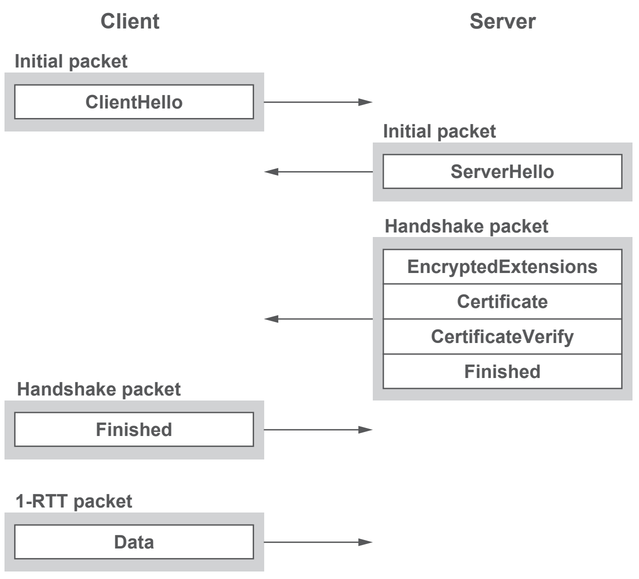


Figure 2: Full QUIC Handshake

A client that receives these packets generates the Handshake key and 1-RTT key. It also puts the generated Finished message into a Handshake packet, encrypts this with the Handshake key, and sends it. At this point, 1-RTT packets capable of storing application data can be sent.

For the second connection, the client can generate a 0-RTT key from the stored information and send an Initial packet followed by a 0-RTT packet capable of storing application data encrypted with the 0-RTT key.

I tried extending the TLS library in various ways to make TLS 1.3 features available in QUIC. Major modifications were needed in order to separate the record layer, but I figured out that starting a dedicated TLS thread was a good way of reusing the TLS library without making any further modifications beyond that.

The callback mechanism proved effective in keeping the state within the scope of the TLS library so that the Client/Server threads do not have to manage the TLS 1.3 state. When a key is generated, a specified callback is used to install the key in the shared data area. And using STM makes it possible for other threads to gauge when the key was installed.

The server-side TLS handshaker thread terminates after sending a NewSessionTicket message in a 1-RTT packet. Meanwhile, the client-side TLS handshaker thread terminates after a set delay upon receiving a HANDSHAKE _ DONE frame.

3.9 Migration

Client IP addresses and port numbers can change. This happens, for example, when the network interface switches from mobile phone to Wi-Fi, or when the port mapping on a NAT gateway between the client and server changes. Connection migration is a feature for keeping connections alive in situations like this.

If the client IP address or port number changes, the server side of my implementation will receive 1-RTT packets on the listening socket. By examining the connection ID, it can determine that a migration has occurred rather than a bad packet.

In this case, the Dispatcher thread starts the Migrator thread (Figure 3), which creates a new connected socket, starts a Reader thread that will use that socket, and performs path validation. For details on path validation, see RFC 9000[*1].

Until a new connected socket is created, the Dispatcher thread passes any packets that arrive to the Migrator thread, and the Migrator thread passes them to the Receiver thread. It also closes the old connected socket after a set delay, thereby terminating the old Reader thread.

We will now look at how migration is handled on the client side, starting with the case in which connected sockets are used.

(1) Detect somehow that a new preferred network interface is available.
(2) Call the migration API. Once a new socket is created and the `connect()` system call is called, the OS sets the remote address and port based on the call's arguments. The routing table is then searched using the remote address to find the network interface to which the route points. The IP address of that network interface is chosen as the socket's local address. Local ports are chosen randomly.
(3) Use the connected socket that was created and `send()` to send packets.

The advantage of this method is that path validation can be performed in step (2), and the disadvantage is that OS-specific methods are needed for step (1). Meanwhile, another option is to use wildcard sockets.

- When `sendto()` is called, the OS sets the remote address and port based on the call's arguments. The routing table is also searched using the remote address to find the network interface to which the route points. The IP address of that network interface is chosen as the socket's local address. The local port is chosen randomly when `sendto()` is first called.

The advantage of this method is that migrations happen automatically without the need to keep track of the preferred network interface or provide a special migration API. The disadvantages are the cost and poor performance involved in sending packets and the lack of an opportune time for path validation.

As each method has its advantages and disadvantages, I plan to provide both so that either can be selected via the settings when launching a client.

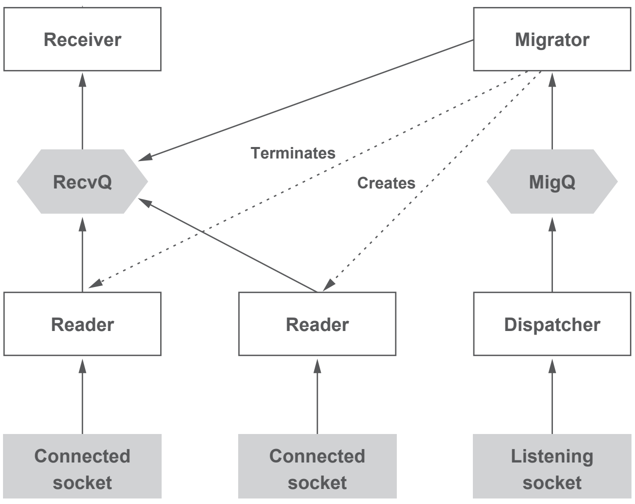


Figure 3: Connection Migration Flow Chart

*1 Reference: J. Iyengar, M. Thomson, "QUIC:A UDP-Based Multiplexed and Secure Transport", RFC 9000, 2021

3.10 ACK Processing Algorithmn

In QUIC, new packet numbers are used for retransmissions. Unlike TCP, which reuses sequence numbers when resending, QUIC has no ACK ambiguity problem. When a packet is resent, the packet number and ciphertext both change. For this reason, RFC 9000 refers to the "retransmission of information" rather than simply to the resending of packets.

In standard TCP, the ACK specifies the sequence number that should be delivered next, so it is not possible to determine whether other TCP segments have been delivered to the peer. QUIC ACK frames, meanwhile, can list packet numbers that have been received.

To enable the retransmission of information carried in packets that have been sent, an information-retransmission container is prepared and the information stored therein at time of transmission. The following three operations can be performed with information-retransmission containers.

- When sending, insert information with the packed number as the key.
- When an ACK is received, delete the information with the packet number as the key.
- If no ACK is returned after a set delay, retrieve and delete the information from the container and retransmit it.

A common data structure in Haskell providing this functionality is the PSQ (Priority Search Queue). We specify the packet number as the key, the transmission time as the priority, and the information as the value.

When I implemented the information-retransmission container with PSQs, I noticed that performance would drop significantly at times. In a normal implementation, for example, say that ACKs are returned as follows.

```
[0,1,2,3]
[4,5,6,7]
[8,9,10,11]
```

That is, the implementation processes ACKs in response to ACKs and dynamically manages which packet numbers need to be ACK'd. At one point, however, Firefox Nightly returned the following ACKs.

```
[0,1,2,3]
[0,1,2,3,4,5,6,7]
[0,1,2,3,4,5,6,7,8,9,10,11]
```

ACKs in response to ACKs are not processed, so unnecessary packet numbers are not deleted. The specification permits this form of ACK. Denoting the size of the PSQ as n and the number of packet numbers specified in the ACK as m, the complexity of the entire delete operation is O(m log n). When m becomes large, as with the Firefox Nightly build I encountered, the delete operation becomes very costly.

I realized that predicates could be used to solve this problem. A list of packet numbers like `[4,5,7,8,9]`, for instance, is represented in an ACK frame in the form of ranges like so: `[(4,5),(7,9)]`. This can be converted into a predicate as follows.

```
predicate :: PlainPacket -> Bool
predicate pkt = (4 <= n && n <= 5) || (7 <= n && n <= 9)
  where
    n = packetNumber pkt
```

Haskell provides as standard a data structure called finger trees (FingerTree), a sequence representation that is easily manipulable at both ends, like a bidirectional list. Finger trees have an operation for splitting themselves into a finger tree that contains only elements matching a predicate and a finger tree holding the non-matching elements, which runs in O(n) time. So using finger trees and predicate-based splitting instead of PSQs, I was able to reduce the computational overhead when ACKs are received.

3.11 Reassembling Streams

QUIC packets do not span multiple IP packets. That is, they are not fragmented or reassembled at the IP level. Data within streams, on the other hand, can span multiple QUIC packets. So the sender needs to split the stream data into appropriately sized fragments, and the receiver needs to reassemble it.

When a STREAM frame arrives, the fragment is inserted into a reassembly container. Then, if there is a continuous set of fragments starting at the expected offset, they are removed and put into the `recvStream` queue. Hence, the reassembly container has insert and retrieve & delete operations.

In the old implementation, I used a one-way list for the reassembly container. Inserts and retrieve & delete operations both ran in O(n) time. When I profiled data transfers in a production environment, I found stream reassembly to be a bottleneck.

This prompted me to adopt a different data structure for the reassembly container: a skew heap populated with finger trees. Elements can be prepended or appended to a finger tree in O(1) time to represent a continuous series of fragments. Computation complexity is reduced: inserts take O(log n) and retrieve & delete operations take O(n) time.

3.12 Flow Control

Flow control is a mechanism whereby senders limit the volume of packets they send to within the bounds of what the receiver can handle. QUIC uses a scheme in which receivers tell senders how much data they can receive (credit). This is often conflated with congestion control, described in Section 3.13, but it is a separate mechanism.

In my implementation, flow control is done at the stream API level.

- `sendStream` sends data within the bounds allowed by the peer, and if the amount exceeds the limit, it waits for credit from the peer.
- `recvStream` assumes that the application will consume this data and sends credit for the amount of data received to the peer.

3.13 Loss Detection and Congestion Control

QUIC loss detection and congestion control are defined in RFC 9002[*2]. Loss detection uses both ACK-based and probe timeout-based methods. And congestion control uses an algorithm based on NewReno. I implemented the pseudocode given in the RFC faithfully in Haskell. In the process of doing so, I discovered, and reported, a number of inconsistencies in the specifications. In recognition of this, the name of this article's author (Kazu Yamamoto) has been added to the RFC 9002 contributors list.

Loss detection and congestion control logs are exported in qlog format[*3] and fed into the qvis[*4] visualization suite to monitor the program's operation and find errors.

*2 Reference: J. Iyengar, I. Swett, "QUIC Loss Detection and Congestion Control", RFC 9002, 2021

*3 Reference: R. Marx, "QUIC and HTTP/3 event definitions for qlog", Internet-Draft, 2020

*4 qvis, "Welcome to qvis v0.1, the QUIC and HTTP/3 visualization toolsuite!" (https://qvis.quictools.info/).

3.14 Testing

My QUIC library and HTTP/3 library implement a variety of unit tests. In this section, I discuss the use of some noteworthy unit tests and external tests.

■ Loss detection

To test if loss detection is working correctly, I implemented a virtual network that relays UDP datagrams through a relay thread. The relay thread drops UDP datagrams based on given scenarios. Naturally, I have implemented tests that randomly drop UDP datagrams. I also comprehensively cover patterns involving handshake packet loss, something that is apt to cause problems, such as tests that drop the client's first packet and tests that drop the second.

■ h3spec

Tests can easily miss error cases. For HTTP/2, `h2spec`[*5] is an excellent test tool for checking if servers can handle error cases. I realized that I could easily test error cases by creating hooks for the Haskell QUIC library. One of the hooks is shown below.

```
onTransportParametersCreated :: Parameters -> Parameters
```

When transport parameters are created, this hook converts one of the parameters from one value to another. An error case can be created by converting to a value that causes an error. Based on this idea, I have released a tool called `h3spec`[*6] for testing error cases against QUIC or HTTP/3 servers. At present, it provides 32 QUIC error tests and 16 HTTP/3 error tests. `h3spec` has been used to test the Haskell QUIC library as well as other implementations, and it has thus played a role in making implementations more stable.

■ QUIC tracker

QUIC tracker is a service that executes a range of tests on public servers once a day and publishes the results. I registered our public server for the service and found a lot of bugs. I was eventually able to pass all test cases except for two unsupported items.

3.15 Acknowledgments

I would like to thank the following individuals. Kazuho Oku gave me the idea for generating connected sockets. Tatsuhiro Tsujikawa discussed the migration API with me. Robin Marx taught me all about qlog and qvis.

Kazu Yamamoto

Head of Development Group, Research Laboratory, IIJ Innovation Institute Inc.
Dr. Yamamoto is interested in applying the concurrent technology of the Haskell programming language to network programming.
He pens the "QUIC wo Yukkuri Kaisetsu" [QUIC at an Easy Pace] series in Japanese on the IIJ Engineers Blog.

*5 h2spec, "A conformance testing tool for HTTP/2 implementation" (https://github.com/summerwind/h2spec).

*6 h3spec, "Test tool for error cases of QUIC and HTTP/3" (https://github.com/kazu-yamamoto/h3spec).

November 2021 Vol.52

Internet Initiative Japan

About Internet Initiative Japan Inc. (IIJ)

IIJ was established in 1992, mainly by a group of engineers who had been involved in research and development activities related to the Internet, under the concept of promoting the widespread use of the Internet in Japan.
IIJ currently operates one of the largest Internet backbones in Japan, manages Internet infrastructures, and provides comprehensive high-quality system environments (including Internet access, systems integration, and outsourcing services, etc.) to high-end business users including the government and other public offices and financial institutions.
In addition, IIJ actively shares knowledge accumulated through service development and Internet backbone operation, and is making efforts to expand the Internet used as a social infrastructure.

Internet Initiative Japan Inc.

Address: Iidabashi Grand Bloom, 2-10-2 Fujimi, Chiyoda-ku,
Tokyo 102-0071, Japan
Email: info@iij.ad.jp URL: https://www.iij.ad.jp/en/