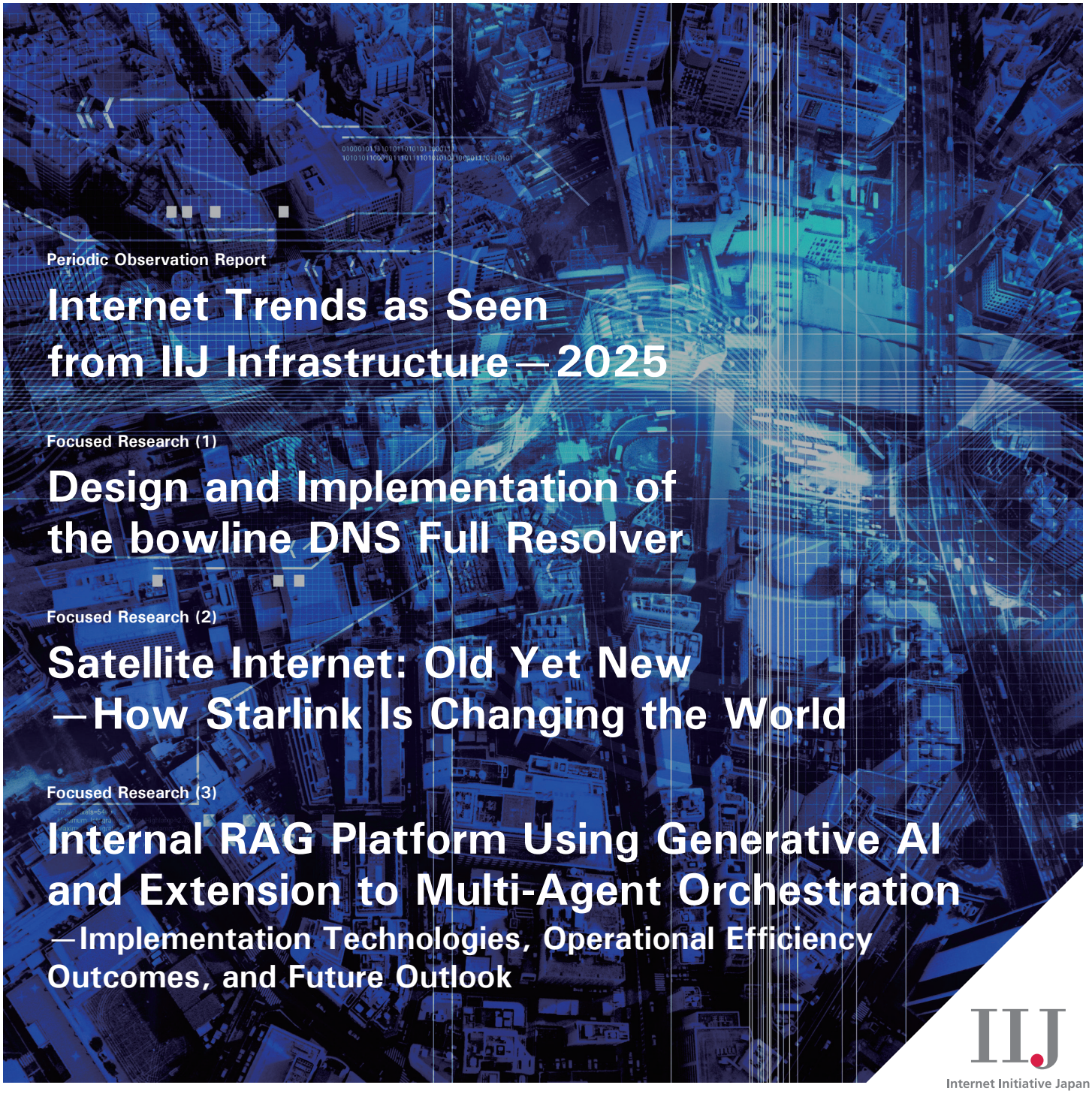
IIR

Internet
Infrastructure
Review

Feb.2026
Vol. 68



Periodic Observation Report

Internet Trends as Seen from IIJ Infrastructure—2025

Focused Research (1)

Design and Implementation of the bowline DNS Full Resolver

Focused Research (2)

Satellite Internet: Old Yet New —How Starlink Is Changing the World

Focused Research (3)

Internal RAG Platform Using Generative AI and Extension to Multi-Agent Orchestration —Implementation Technologies, Operational Efficiency Outcomes, and Future Outlook

IIJ
Internet Initiative Japan

Internet Infrastructure Review

February 2026 Vol.68

Executive Summary

2025 has been called a watershed year for AI agents. GPT-5 was announced in August 2025, three years after the release of ChatGPT in 2022 sent shockwaves around the world. GPT-5 not only supports multimodal inputs, such as images and video, but also features significantly enhanced external integration capabilities, making it possible for autonomous agents to interact with external data sources as needed. The deployment of AI agents is also being driven by growing momentum within enterprises, which, having largely worked through their experimental phases with AI, are now looking to use it in real-world business applications.

In this issue, we bring you the latest Internet trends along with examples of IIJ's development efforts involving generative AI and AI agents.

Chapter 1 presents our periodic observation report on Internet trends. IPv6 traffic on IIJ's backbone increased a hefty 25.2% year over year and now accounts for around 24% of overall traffic. IPv6-enabled rates on mobile devices have also improved (iOS: 86.9%, Android: 35.5%). The report also discusses trends as seen from IIJ's Internet backbone.

Chapter 2 discusses the design philosophy and implementation of bowline, a full-service DNS resolver developed in-house by IIJ. Implemented in Haskell, bowline has been released as open source. I encourage you to read the article for the background discussion and other details on the Haskell implementation, but I would also like to note that this kind of internal software development is what underpins the reliability of the services IIJ provides.

Chapter 3 looks at the history of LEO (Low Earth Orbit) satellite Internet, an area of rising interest in recent years. The report also discusses the latest developments, with a focus on Starlink. When the Noto Peninsula earthquake struck Japan on New Year's Day 2024, LEO satellites came into play, serving as a means of communication in a disaster situation. A whole range of possibilities lies ahead, including high-speed, low-latency networks using inter-satellite laser communication and applications for interplanetary communication. The IIJ Group officially began offering Starlink in December 2025.

Chapter 4 introduces sbdGPT, an internal RAG (Retrieval-Augmented Generation) platform developed and operated by IIJ, and its multi-agent extensions. IIJ has been working to integrate its vast stores of knowledge distributed across the company and using generative AI to improve operational efficiency. Since we put sbdGPT into service in the summer of 2023, it has delivered efficiency gains equivalent to around 1,500 hours per month. We are in the process of greatly expanding the scope of AI utilization at IIJ. These efforts include a multi-agent architecture that also incorporates external information, integration with Deep Research tools to provide advanced research capabilities, and the in-house development of Panorama, a tool for automatically generating plans, proposals, and strategies.

I hope the wide range of topics covered in this issue, from familiar areas such as Internet trend analysis and the full-service DNS resolver we developed in-house to cutting-edge topics like satellite Internet and generative AI, will prove useful to our readers. Guided by our mission of supporting society through technology, we will continue to evolve, in terms of both providing stable services and taking on the challenge of new technologies.

Naoshi Someya

Managing Executive Officer; Network Services Business Unit; Director, Cloud Division, IIJ
Mr. Someya joined IIJ in 1998 and was seconded shortly thereafter to IIJ Technology (which was merged into IIJ in 2010). At IIJ Technology, he was involved in the launch of the systems integration (SI) business and worked on building numerous Internet systems as well as providing consulting services. In 2016, he transferred to IIJ's Service Business Division, where he was responsible for medium-term strategy for the cloud business. In 2019, he became head of the cloud business. As of this fiscal year, he serves as editor-in-chief of the IIR, with his aim being to proactively deliver practical, cross-cutting technical insights from across IIJ to readers.

Internet Trends as Seen from IIJ Infrastructure —2025

Each year in the IIR, we present an analysis of data obtained through the operation of IIJ's network and server infrastructure, which ranks among the largest in Japan. This year, we again analyze changes in trends over the past year from the perspective of BGP routes, DNS query analysis, IPv6 and mobile, and the Internet backbone.

Topic 1

BGP and Routes

We start by looking at IPv4 full-route information advertised by our network to other organizations (Table 1) and the number of unique IPv4 addresses contained in the IPv4 full-route information (Table 3).

The annual increase in total routes recovered to over 40,000 (Figure 1), bringing the total to over 990,000 routes. Note that as of this writing (early October 2025), the total has already surpassed 1,000,000 routes, but as with other observation points, this milestone appears to have been passed without any real commotion. By prefix length, the increases in the number of routes stood out for /24 and /23 as well as for /16 and /20. While the number of /18 routes looks to have decreased according to Table 1, the figures for aggregated routes only (removing routes for which shorter-prefix route information exists; Table 1-a) instead show a three-digit increase, suggesting these are in high demand due to address transfers in recent years. The number of unique IPv4 addresses turned around after a two-year downtrend for a sharp rise of just under 74 million (/8 blocks x 4.4). This also exceeds the 2022 value and

Table 1: Number of Routes by Prefix Length for Full IPv4 Routes

Date	/8	/9	/10	/11	/12	/13	/14	/15	/16	/17	/18	/19	/20	/21	/22	/23	/24	total
Sep. 2016	16	13	36	101	267	515	1050	1767	13106	7782	12917	25229	38459	40066	67270	58965	335884	603443
Sep. 2017	15	13	36	104	284	552	1047	1861	13391	7619	13385	24672	38704	41630	78779	64549	367474	654115
Sep. 2018	14	11	36	99	292	567	1094	1891	13325	7906	13771	25307	39408	45578	88476	72030	400488	710293
Sep. 2019	10	11	37	98	288	573	1142	1914	13243	7999	13730	25531	40128	47248	95983	77581	438926	764442
Sep. 2020	9	11	39	100	286	576	1172	1932	13438	8251	14003	25800	40821	49108	101799	84773	473899	816017
Sep. 2021	16	13	41	101	303	589	1191	2007	13408	8231	13934	25276	41915	50664	106763	91436	497703	853591
Sep. 2022	16	13	39	101	298	592	1208	2064	13502	8292	13909	25051	43972	52203	109071	96909	536520	903760
Sep. 2023	16	14	39	102	298	577	1196	2064	13490	8245	13809	25059	43863	51012	109514	98178	550621	918097
Sep. 2024	16	16	37	93	295	573	1165	2059	13224	8220	13718	24624	43786	51827	111483	99239	579274	949649
Sep. 2025	16	14	41	92	298	576	1200	2125	13768	8257	13491	24718	44863	52244	112330	102820	616490	993343

Table 1-a: Number of Routes by Prefix Length for Full IPv4 Routes (Aggregated Routes Only)

Date	/8	/9	/10	/11	/12	/13	/14	/15	/16	/17	/18	/19	/20	/21	/22	/23	/24	total
Sep. 2024	16	9	34	88	264	487	1024	1667	10277	4417	6815	14827	19079	22624	59317	42062	283007	466017
Sep. 2025	16	9	41	85	263	485	1031	1655	11026	4460	6943	14831	20459	22686	59763	43628	304448	491829
Change	0	0	4	-3	-1	-2	7	-12	749	43	128	4	1380	62	446	1566	21441	25812

Table 2: Number of Routes by Prefix Length for Full IPv6 Routes

Date	/16-/28	/29	/30-/31	/32	/33-/39	/40	/41-/43	/44	/45-/47	/48	total
Sep. 2016	153	1294	216	8110	3092	1445	371	1492	1006	14291	31470
Sep. 2017	158	1757	256	9089	3588	2117	580	1999	1983	18347	39874
Sep. 2018	168	2279	328	10897	4828	2940	906	4015	2270	24616	53247
Sep. 2019	192	2671	606	12664	6914	3870	1566	4590	4165	34224	71462
Sep. 2020	205	3164	641	14520	9063	4815	2663	5501	4562	45160	90294
Sep. 2021	223	3628	705	20650	13050	10233	4170	11545	5204	61024	130432
Sep. 2022	298	4247	895	21926	15147	12509	4108	13840	6994	73244	153208
Sep. 2023	316	4357	923	23228	17427	14828	5518	16453	9579	86881	179510
Sep. 2024	322	5360	934	24739	20198	17657	4672	19418	12470	95628	201398
Sep. 2025	271	5089	1062	25643	22627	20719	5430	21021	14522	101103	217487

marks a new record high since the inception of this periodic observation report.

Next, we look at IPv6 full-route information (Table 2) and the number of unique IPv6 /64 blocks in the IPv6 full-route information (Table 3).

The annual increase in total routes was around 16,000. This marks the lowest value since 2019 and the first time in this periodic observation report that the year-on-year growth rate has dipped below 10% (+8%). Looking at aggregated routes only, however, the increase (+9,000) was on par with the previous year's and ranks fourth overall in this periodic observation report's history. The number of unique /64 blocks also surpassed 1 trillion (+23%), suggesting that IPv6 deployment and the expansion of IPv6 networks continue to progress steadily. By prefix length, the number of routes for /16–/28 prefix blocks and /29 blocks fell substantially, but the drop in the former is attributable to a reduction in routes for which shorter prefixes are announced elsewhere. Looking at aggregated routes only, we instead see an increase of 6 (Table 2-a).

Lastly, let's also look at IPv4/IPv6 full-route Origin AS figures (Table 4). In the past year, APNIC was allocated an additional 2048 32-bit only ASNs.

Table 2-a: Number of Routes by Prefix Length for Full IPv6 Routes (Aggregated Routes Only)

Date	/16-/28	/29	/30-/31	/32	/33-/39	/40	/41-/43	/44	/45-/47	/48	total
Sep. 2024	223	5311	521	22942	6861	4776	1773	5348	3712	38335	89802
Sep. 2025	229	5037	546	23628	7820	6265	2282	6635	4235	42376	99053
Change	6	-274	25	686	959	1489	509	1287	523	4041	9251

Table 3: Total Number of Unique IPv4 Addresses in Full IPv4 Routes and Total Number of Unique IPv6 /64 Blocks in Full IPv6 Routes

Date	No. of IPv4 addresses	No. of IPv6 /64 blocks
Sep. 2016	2,824,538,880	26,432,856,889
Sep. 2017	2,852,547,328	64,637,990,711
Sep. 2018	2,855,087,616	258,467,083,995
Sep. 2019	2,834,175,488	343,997,218,383
Sep. 2020	2,850,284,544	439,850,692,844
Sep. 2021	3,036,707,072	461,117,856,035
Sep. 2022	3,068,374,784	532,578,391,219
Sep. 2023	3,055,604,992	700,359,397,494
Sep. 2024	3,033,333,504	896,502,953,452
Sep. 2025	3,107,136,512	1,102,106,091,904

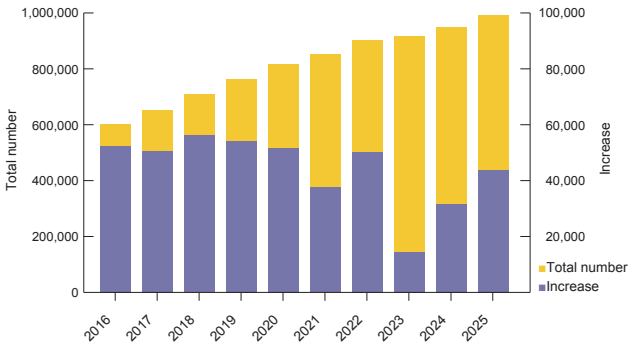


Figure 1: Total Number of Full IPv4 Routes and Annual Increases

Table 4: IPv4/IPv6 Full-Route Origin AS Numbers

ASN	16-bit(1〜64495)					32-bit only(131072〜4199999999)				
Advertised route (IPv6-enabled)	IPv4+IPv6	IPv4 only	IPv6 only	total	(IPv6 -enabled)	IPv4+IPv6	IPv4 only	IPv6 only	total	(IPv6 -enabled)
Sep. 2016	9116	33555	158	42829	(21.7%)	2406	9391	146	11943	(21.4%)
Sep. 2017	9603	32731	181	42515	(23.0%)	3214	12379	207	15800	(21.7%)
Sep. 2018	10199	31960	176	42335	(24.5%)	4379	14874	308	19561	(24.0%)
Sep. 2019	10642	31164	206	42012	(25.8%)	5790	17409	432	23631	(26.3%)
Sep. 2020	11107	30374	229	41710	(27.2%)	7653	19668	574	27895	(29.5%)
Sep. 2021	11465	29219	302	40986	(28.7%)	9514	21108	5242	35864	(41.1%)
Sep. 2022	11613	28398	369	40380	(29.7%)	10816	22211	5764	38791	(42.7%)
Sep. 2023	11770	27617	460	39847	(30.7%)	12640	22128	2067	36835	(39.9%)
Sep. 2024	12068	26720	476	39264	(31.9%)	13905	22737	2386	39028	(41.7%)
Sep. 2025	12239	25835	438	38512	(32.9%)	15319	23223	2568	41110	(43.5%)

The number of 16-bit Origin ASes has been falling for 10 years straight, and this year's decline was the largest since we began these periodic observations. Because the decrease in "IPv4 only" again exceeded the increase in 32-bit-only ASes, the total number of IPv4 only Origin ASes fell for a third consecutive year (Figure 2). The number of IPv6 only Origin ASes also saw its first double-digit drop. The number of 32-bit-only Origin ASes increased by roughly the same amount as in the previous edition and has finally come to account for a majority of all Origin ASes. The increases in "IPv4 only" and "IPv6 only," meanwhile, were much lower than last time, and the share of total accounted for by "IPv4+IPv6" Origin ASes rose from 58% last time to 68%. These results suggest that the practice of treating IPv4 and IPv6 separately (separate ASes) continues to unwind, and whether this trend continues is something we will continue to monitor in future editions.

Topic 2

DNS Query Analysis

IIJ provides a full-service resolver to enable DNS name resolution for its users. Here, we discuss the state of name resolution, and analyze and reflect upon data from servers provided mainly for consumer services, based on a day's worth of full-service resolver observational data obtained on October 22, 2025.

The full-service resolver provides a name resolution function that replies to DNS queries from user devices. Specifically, to resolve a name, it starts by looking at the IP address of an authoritative server for the root zone (the highest level zone), which it queries, and then goes through other authoritative servers to find the records it needs. If the full-service resolver repeatedly queries other servers like this, it can result in increased load and delays, so the information obtained is cached, and when the same query is received again, the response is sent from the cache. Recently, DNS-related functions are implemented on devices that lie on route paths, such as consumer-level routers and firewalls, and these devices are sometimes also involved in relaying DNS queries and applying control policies. Some applications, such as Web browsers, also have their own implementations of name resolver functionality and in some cases resolve names based on a policy that differs from the OS settings.

ISPs notify users of the IP address of full-service resolvers via various protocols, including PPP, DHCP, RA, and PCO, depending on the connection type, and they enable automatic configuration of which nameserver to use for name resolution on user devices. ISPs can notify users of multiple full-service resolvers, and users can specify which nameserver to use by altering settings in their OS, browser, or elsewhere. When more than one nameserver is configured on a user device, which one ends up being used depends on the device's implementation or the application, so any given full-service resolver is not aware of how many queries a user is sending in total. When running full-service resolvers, therefore, this means that you need to keep track of query trends and always try to keep some processing power in reserve because changes in behavior

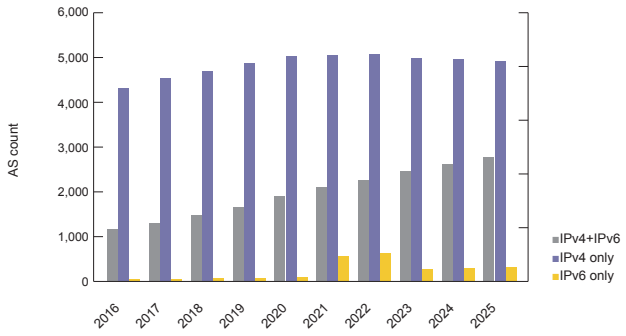


Figure 2: IPv4/IPv6 Full-Route Origin AS Counts (Combined Total)

or status on the user end can conceivably result queries suddenly being concentrated on a particular resolver.

Observational data on the full-service resolver provided by IIJ show fluctuations in user query volume throughout the day, with volume hitting a daily trough of about 0.13 queries/sec per source IP address at around 3:30 a.m., and a peak of about 0.27 queries/sec per source IP address at around 12:25 p.m. The minimum was down 0.02pt and the maximum down 0.05pt vs. the previous year. The breakdown shows that IPv4 accounted for around 43% of queries and IPv6 for around 57%, with IPv6's share having risen by around 16pt from the previous year, marking the first time since we began these periodic observations that IPv6 queries have outnumbered IPv4. Turning to protocols, UDP accounted for almost all (97.67%) of the queries. That said, TCP queries have been rising gradually in recent years, from 0.189% of total in 2021 to 0.812% in 2022, 1.419% in 2023, 1.561% in 2024, and 2.335% in 2025. This is possibly due to an increase in queries using DNS over TLS (DoT), and there may also be implementations out there that use TCP queries for some purpose such as connectivity or operation checks.

Recent years have seen a tendency for queries to rise briefly at certain round-number times, such as on the hour marks in the morning. We again saw similar increases in 2025. We also observed, as in the previous year, increases in query volume at 14 and 9 seconds before hour marks. This is a pattern we have seen in recent years, with query volume rising sharply at the hour mark and then tapering off gradually, but with the sudden spikes that occur ahead of the hour mark, query volume quickly returns to roughly where it had been. Hence, because a large number of devices are sending queries in almost perfect sync, we surmise that lightweight, quickly completed tasks of some sort are being executed. Last year, we observed query volume actually falling at the top of each hour from 8 a.m. to 10 p.m. and rising gradually thereafter, but this year we observed increases at all hour marks. We suspect this reflects some implementation changes on client devices that use name resolution.

Looking at the query record types, A records that query the IPv4 address corresponding to the host name, AAAA records that query IPv6 addresses, and HTTPS records used to resolve Web services account for 98% of the total. The trends in A and AAAA queries differ by IP protocol, with more AAAA record queries being seen for IPv6-based queries. Of IPv4- based queries, around 71% are A record queries and 11% AAAA record queries (Figure 3). With IPv6-based queries, meanwhile, A record queries account for around 42% and AAAA record queries around 34% of the total (Figure 4).

Compared with the previous year, we observe a 9-percentage-point increase in A record queries for IPv4 and a 2-percentage-point increase for IPv6. Meanwhile, HTTPS record queries, which we started to see in 2020, again declined as in 2024. They accounted for around 16% of IPv4 and 22% of IPv6 queries, decreases of 1 percentage point for IPv4 and 2 percentage points for IPv6 from the

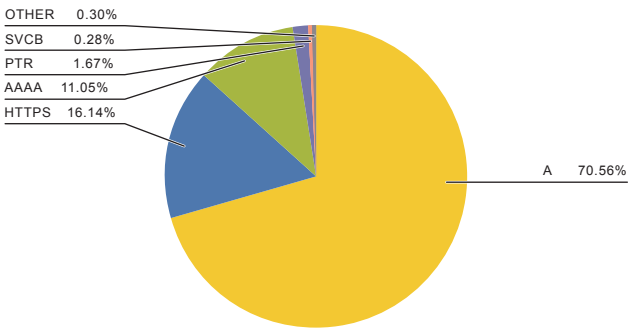


Figure 3: IPv4-based Queries from Clients

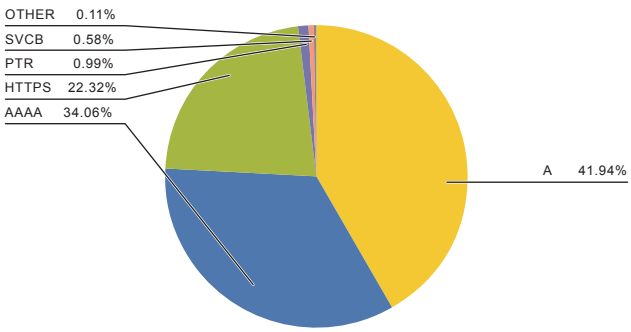


Figure 4: IPv6-based Queries from Clients

previous year. SVCB records, which we started to see in 2022, accounted for 0.28% of IPv4 and 0.58% of IPv6 queries, and while these queries are still only a small fraction of the total, they are progressing steadily. This may be attributable to the use of implementations of Discovery of Designated Resolvers (DDR), designed to allow clients to detect encryption-capable full-service resolvers.

Topic 3

IPv6 & Mobile

In this section, we again report on IPv6 traffic on the IIJ backbone, source ASNs, and the main protocols used. Like last year, we also look at IPv6-enabled rates on mobile services by device OS.

Traffic

Figure 5 shows traffic measured using IIJ backbone routers at core POPs (points of presence—3 in Tokyo, 2 in Osaka, 2 in Nagoya). The data cover the eight months from February 1 to September 30, 2025.

Over the period, Internet traffic increased 5.6% year over year in total, with an increase of 25.2% for IPv6 and 0.6% for IPv4. Last year, growth in both IPv6 and IPv4 was largely flat, but IPv6 traffic grew markedly in 2025.

Figure 6 graphs traffic indexed to 100 as of February 1, 2025. IPv4 hovers around 100, whereas IPv6 ranges between 100 and 140, generally moving around the 120 mark.

Next, Figure 7 shows IPv6 as a proportion of total traffic. Over the period, it ranged from a minimum of 20.5% to a maximum of 26.9%, averaging 23.9%. This is an increase of around 4 points from the previous year. We did not expect to see much growth because it had been stagnant in the previous year, but we were pleasantly surprised.

Table 5 tracks the IPv6 ratio since 2017. IPv6 traffic accounted for only around 4% of total back when we began these periodic observations, but its use has now expanded to the point that it now accounts for some 24%.

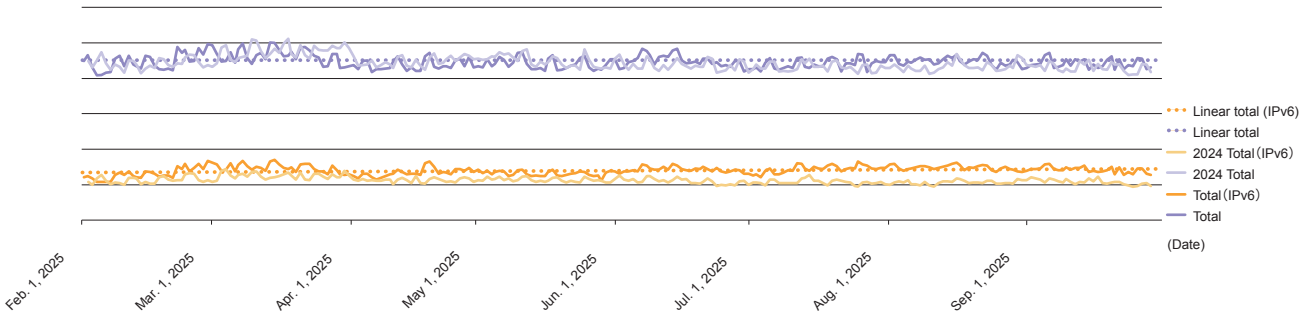


Figure 5: Traffic Measured on Backbone Routers at IIJ's Core POPs

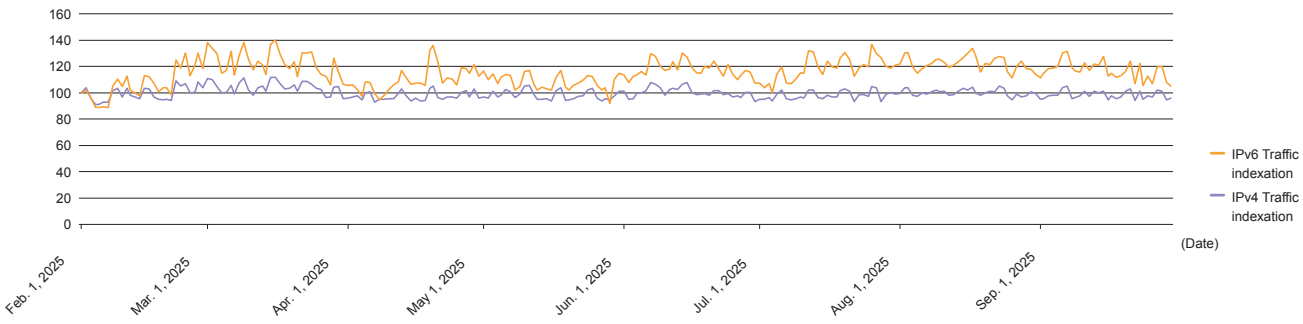


Figure 6: Traffic Indexed to 100 as of February 1, 2025

Traffic Source Organization (BGP Source AS)

Next, Figures 8 and 9 show the top IPv6 and IPv4 traffic source organizations (BGP Source AS Number) for February 1 – September 30, 2025.

For IPv6, traffic within IIJ traffic (AS2497<=>AS2497) accounts for 69%. This is an increase of 3 points from the previous year's reading of 66%.

Looking non-IIJ ASes, Company A, a major Japanese content provider, moves up from No. 2 last year to take the No. 1 spot (6% of traffic) from Company B, a major U.S. search provider. Company B slides back into No. 2 with 4%, and Company C, a major U.S. e-commerce and cloud services provider, comes in at No. 3 with 2%. The rest of the lineup has not changed much, but as was the case last year, no single entity stands out dramatically, and we would expect the rankings to change depending on the observation period chosen.

Note that Company A (No. 1 for IPv6) comes in at No. 5 for IPv4, so one can imagine that it is actively building and providing IPv6-enabled services.

Table 5: IPv6 as a Proportion of Total Traffic (Since 2017)

	2017 IIR Vol. 37	2018 IIR Vol. 41	2019 IIR Vol. 45	2020 IIR Vol. 49	2021 IIR Vol. 53	2022 IIR Vol. 57	2023 IIR Vol. 61	2024 IIR Vol. 65	2025 IIR Vol. 68
IPv6 ratio	4%	6%	10%	10%	11.2%	15.1%	20.1%	20.16%	23.9%

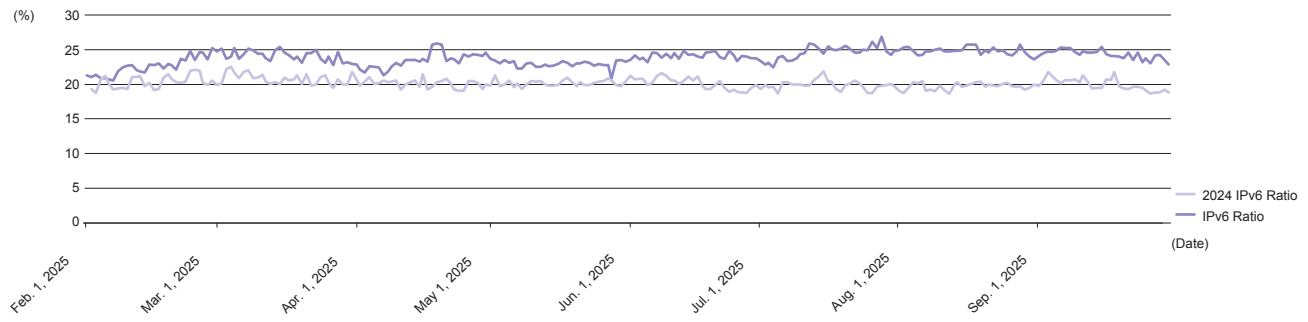


Figure 7: IPv6 as a Proportion of Total Traffic

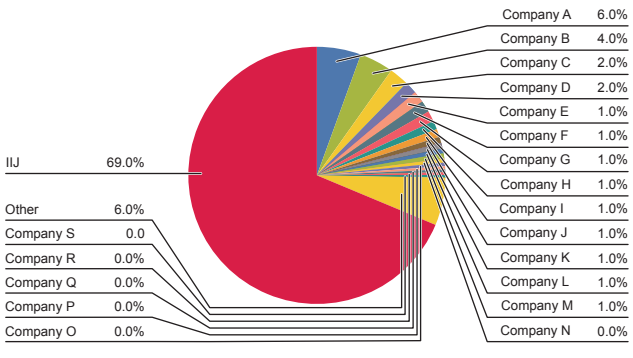


Figure 8: IPv6 Traffic by Source Organization (BGP AS Number)

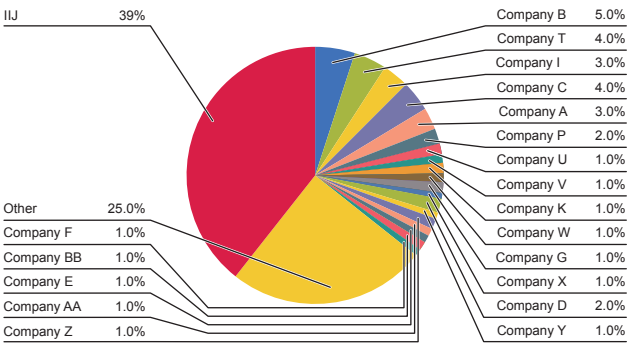


Figure 9: IPv4 Traffic by Source Organization (BGP AS Number)

Protocols Used

Figure 10 plots IPv6 traffic according to protocol number (Next Header) and source port number, and Figure 11 plots IPv4 traffic according to protocol number and source port number (for the week of Monday, September 29 – Sunday, October 5, 2025).

In the IPv6 space, as was the case last year, the top four protocols—HTTPS, QUIC, NAT Traversal, and ESP in that order—accounted for 92.4% of usage. HTTPS accounted for 76.3% (+2.3 points year over year), QUIC 8.7% (-0.3), and HTTP 0.8% (-0.2). So the proportion accounted for by HTTP-related protocols (around 85.9%, +1.9) is rising steadily, and encryption is also increasingly being implemented.

For HTTP-related protocols in the IPv4 space, HTTPS accounted for 55.8% (+0.6 points year over year), QUIC 6.7% (+1.0), and HTTP 4.2% (-1.1), bringing the total for HTTP-related protocols to 66.8% (+0.6). As with IPv6, it appears encryption is increasingly being implemented here as well. Compared with IPv6, HTTP-related protocols account for a lower proportion of total and a greater volume of traffic is classified as "other," suggesting that IPv4 features in a wider variety of use cases (protocols) and that there are many applications and servers that do not yet support IPv6.

Traffic patterns remain largely similar to last year. IPv6 usage is heavier at night, appearing to be nearly double that of daytime usage. But on weekends, daytime usage increases

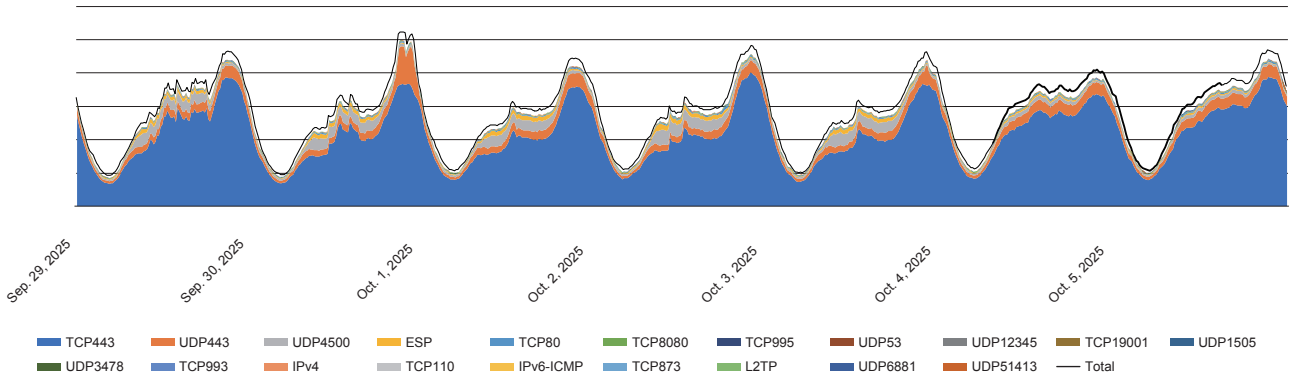


Figure 10: Breakdown of IPv6 Traffic by Source Port Number

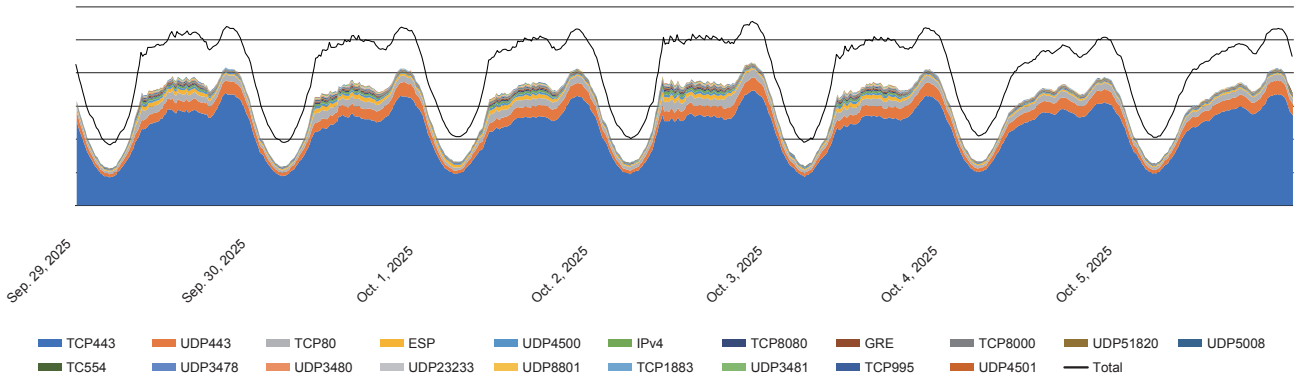


Figure 11: Breakdown of IPv4 Traffic by Source Port Number

while nighttime usage declines slightly. Compared with IPv4, the shape of the curve differs significantly: IPv4 is used fairly evenly throughout the hours when people are awake and active, whereas IPv6 appears to see heavier usage during private hours (outside of working hours).

IPv6 on Mobile Devices

In this edition, we again look at IPv6-enabled rates on personal mobile service (IIJmio Mobile Service) connections. We also look at differences by device OS and at whether there are differences depending on device manufacturer.

The IPv6-enabled rate for devices connected to the IIJmio Mobile Service was 62.9%. This represents an annual increase of around 2 percentage points, from 60.6% last year and 58.73% the year before that. By device OS, 86.9% of Apple iOS devices (including Apple OSes for other mobile devices such as iPadOS) had IPv6 enabled, while the figure was 35.5% for Android devices. The Android IPv6-enabled rate was up 5 percentage points for a second year running, which contributed to the overall rise in the IPv6-enabled rate.

Next, we look at IPv6-enabled connections on the IIJmio Mobile Service, ranked by manufacturer in order of connection count. The pie chart in Figure 12 shows Apple (iPhone, iPad, etc.) accounts for the vast majority of connections at 74%. Google Pixel follows at 10.2%, with Motorola putting in a strong showing in third place at 8.5%. Japanese manufacturers such as FCNT, Sharp,

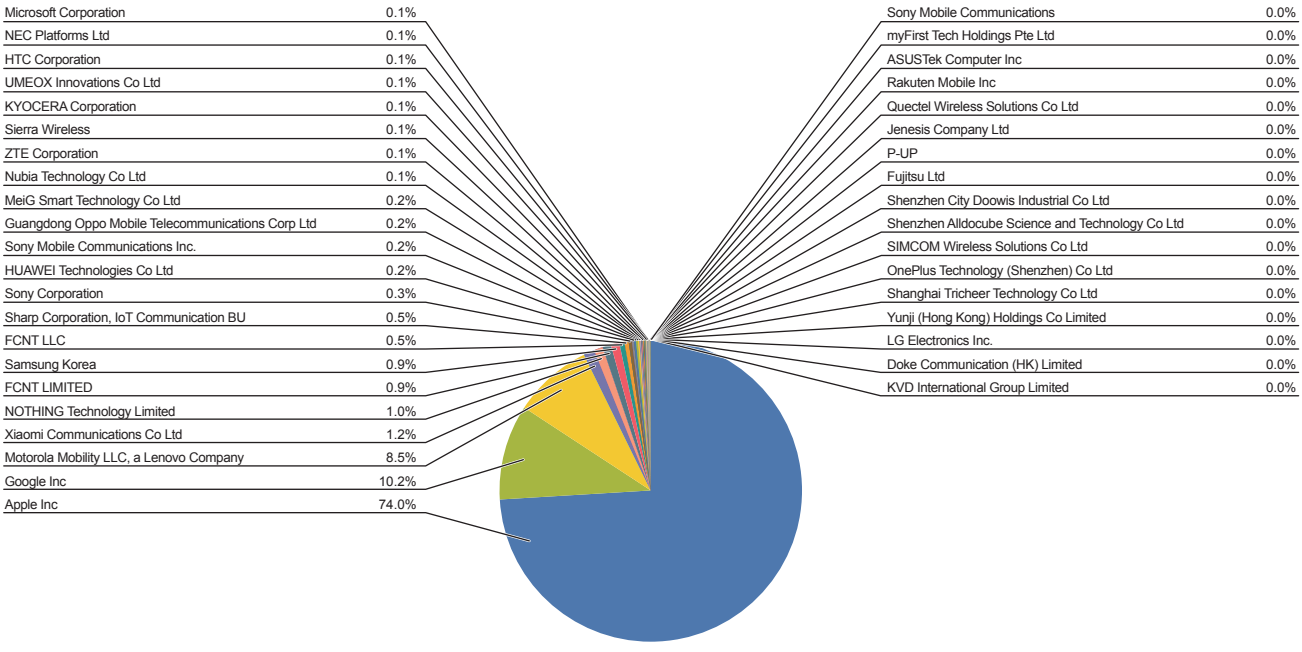


Figure 12: IPv6 Support by UE Manufacturer

Sony, Kyocera, and NEC Platforms are also part of the mix, but the connection counts are low and many of the devices do not have IPv6 enabled, so they all come in below 1%. In last year's report, we noted that the FCNT arrows We2 appears to have IPv6 enabled by default, and we hope to see IPv6 adopted as standard in even more devices going forward.

Summary

- IPv6 traffic on the IIJ backbone grew sharply in 2025, up 25.2% year over year, and total traffic volume increased 5.6% year over year. The pronounced growth in IPv6 in particular indicates steady progress in the shift toward IPv6 across the Internet infrastructure.
- IPv6 as a proportion of total traffic averaged 23.9%, a record high. Having expanded from around 4% in 2017 to about 24% over roughly eight years, IPv6 is becoming firmly established.
- By source ASN, excluding internal IIJ traffic, Company A at No. 1 accounted for 6% of traffic, Company B at No. 2 for 4%, Company C at No. 3 for 2%. Company A appears to be actively deploying IPv6-enabled services.
- By protocol, HTTPS is dominant, accounting for 76.3% of IPv6 traffic. Together with QUIC, NAT Traversal, and ESP, encrypted HTTP-related protocols and VPN protocols account for over 90% of traffic, indicating that secure communications have become the norm.
- The IPv6-enabled rate for mobile devices was 62.9%, with iOS devices (including iPadOS) at 86.9% and Android devices at 35.5%. Growth in the Android IPv6-enabled rate thus contributed to the overall increase. By manufacturer, Apple accounts for an overwhelming majority, while in the Android camp, Google Pixel and Motorola devices are going strong.

Topic 4

Internet Backbone Trends

In this section, we cover trends related to interfaces and RPKI as they relate to interconnectivity on IIJ's internet backbone infrastructure.

■ Interconnection Interface Trends and Requirements

To facilitate Internet interconnectivity, service operators need to agree on a standardized set of interfaces. As of October 2025, IIJ primarily uses 400G-FR4, 100G-LR4, and 10G-LR for interconnection interfaces. The recent trend of reevaluating operator interconnections based on 10G interfaces continued in 2025 as well. Factors in determining interconnection interfaces include traffic volumes and the number of available interfaces on each side, and the migration of interconnections to 100G as parties come to mutual agreements on the conditions is ongoing.

Figures 13 and 14 show the current breakdown of interconnection interface types on the IIJ backbone. The 2024 data were created using the same extraction criteria as for 2025. Note that the resulting percentages differ from those shown in IIR Vol.65 (https://www.iij.ad.jp/en/dev/iir/065.html).

Use of 400G interfaces for interconnections remains limited. In the charts, percentages for 400G are rounded down to 0% because the counts for 100G/10G have increased, but a small number of 400G interfaces do indeed exist. The decline in 10G interfaces is evident from the overall figures. We believe a major factor in this over the past year is the migration of interfaces at Internet Exchange Points (IXPs) from 10G to 100G.

10G interfaces that had been connected via Link Aggregation (LAG) are being migrated to 100G in step with equipment refresh cycles. And operators continue to migrate interconnections to 100G once, following equipment upgrades, both parties can provision 100G. For 10G handoffs after capacity upgrades, it has become increasingly common to provide them by bundling physical links via MPO-based

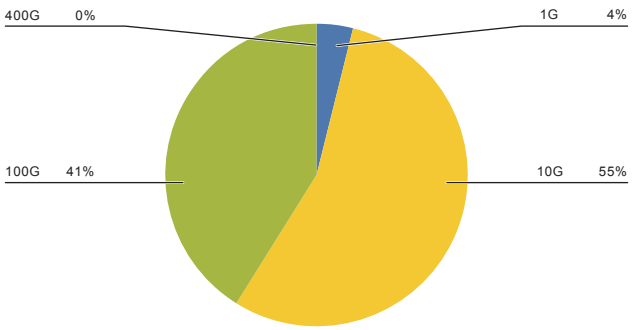


Figure 13:Breakdown of Interconnection Interfaces on IIJ's Internet Backbone (October 2024)

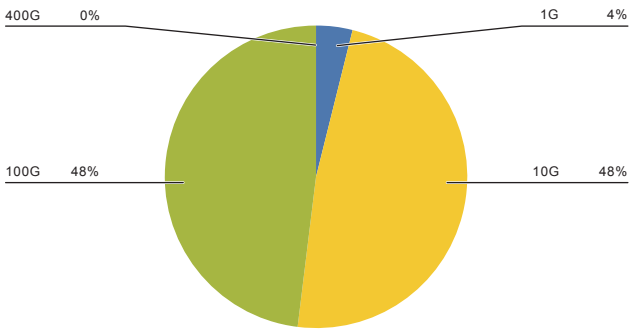


Figure 14:Breakdown of Interconnection Interfaces on IIJ's Internet Backbone (October 2025)

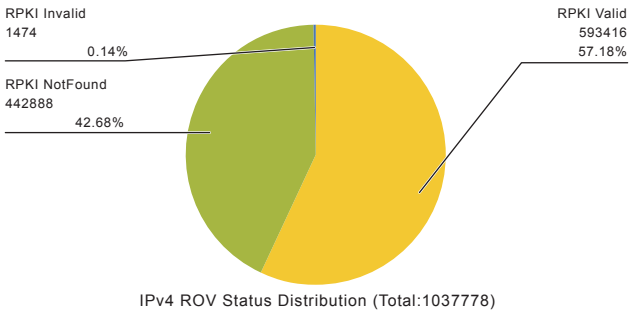
breakout from a 100G port. Some equipment does not provide standalone 10G interfaces, and instead supports splitting higher-speed interfaces into lower-speed interfaces for use. At IIJ as well, following equipment renewals, we have had more opportunities to adopt this feature when continuing to use 10G interfaces.

The share of 100G continues to increase and it is the primary interface for interconnections at present. In terms of 100G interface types, we continue to use 100G-LR4. We have not yet introduced single-lambda 100G-LR.

Current State of RPKI

Here, we provide an update on the state of RPKI. We look at current data on ROAs, signed objects that attest to the legitimacy of IP address resources held by an organization. To build a picture of ROA registrations, we analyzed validation results for Internet routes based on data obtained from an RPKI ROV validator operated by IIJ Lab (Figure 15, Figure 16). Across all IPv4 routes on the Internet, 57.18% are Valid (ROA registered, routing verified), 42.68% are Not-Found (ROA not yet registered), and 0.14% are Invalid (discrepancies in ROA registration, treated as unauthorized routes). In the IPv6 space, 63.32% of routes are Valid, 36.25% are Not-Found, and 0.44% are Invalid.

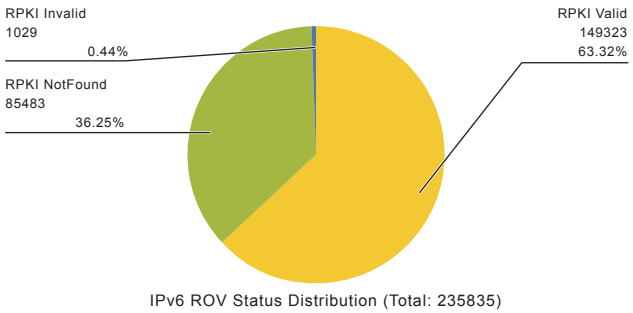
For both IPv4 and IPv6, the share of Valid routes has increased since October 2024. And because the percentages for Not-Found and Invalid have declined, we can infer that the issuance of ROAs for IP addresses has progressed. We note quite an improvement for Invalid in particular, a positive sign that organizations have been reviewing and correcting their ROA registrations for Internet routes.



IPv4 ROV Status Distribution (Total:1037778)

For reference: Oct. 2024 data: Valid 53.38%, Not-Found 46.15%, Invalid: 0.47%

Table 15: ROA Registration Data from the RPKI Monitor (IPv4) as of October 2025



IPv6 ROV Status Distribution (Total: 235835)

For reference: Oct. 2024 data: Valid 55.30%, Not-Found 40.24%, Invalid 4.46%

Table 16: ROA Registration Data from the RPKI Monitor (IPv6) as of October 2025

Now let's look at the ROA registration status of Internet routes that IIJ generates and advertises (Table 6). IIJ participates in the Internet using Global AS number AS2497, and thus the Origin AS for routes IIJ advertises is AS2497. As of October 2025, 43.65% of routes originating from AS2497 are Valid. This means ROAs are registered, confirming legitimacy, for just under half of all routes. Yet, ROV returns Not-Found for over 50% of the routes, indicating no ROA has been registered. One significant obstacle to ROA registration is the need for organizations that own IP addresses to register and issue their own ROAs. While IIJ has registered ROAs for nearly all IP addresses it owns (excluding special cases), there is evidently a lack of progress among users who bring their own IP addresses to IIJ. Since IIJ cannot currently register ROAs on behalf of users, they must complete the registration themselves. IIJ offers support in this regard, so we encourage you to take advantage of this to help us increase the ROA registration rate for AS2497.

When including routes for which IIJ provides transit service to customers, 56.72% of all routes are Valid according to ROV. This figure is higher than that from last year's data, again indicating decent progress in terms of ROA issuance within Japan as well.

Table 6: ROA Registration Status of Internet Routes Generated and Advertised by IIJ

	No. of routes originated by AS2497	No. of routes IIJ transits and advertises to the Internet
Valid	79	4405
Unknown	102	3757
Invalid	0	4
Valid rate	43.65%	56.72%

1.BGP and Routes
Tomohiko Kurahashi
Operation Research & Development Section, Operation Engineering Department, Infrastructure Engineering Division, Network Services Business Unit, IIJ

2.DNS Query Analysis
Yoshinobu Matsuzaki
Operation Research & Development Section, Operation Engineering Department, Infrastructure Engineering Division, Network Services Business Unit, IIJ

3.IPv6 & Mobile
Taisuke Sasaki
Infrastructure Development Department, MVNO Division, Mobile Services Business Unit, IIJ

4.Internet Backbone Trends
Yuichi Yomogita
Network Engineering Section 1, Network Engineering Department, Infrastructure Engineering Division, Network Services Business Unit, IIJ

Design and Implementation of the bowline DNS Full Resolver

At IIJ, we develop our own DNS full resolver (also known as a caching DNS server; we refer to this simply as a full resolver below). We named our software bowline after the bowline knot, often called the king of knots. At this point, we have finished implementing most of the functions necessary for ISP operations, such as logging and monitoring, and are verifying stability through trial operations. This article describes the design and implementation of bowline.

IIJ provides caching DNS servers using multiple full resolver implementations. One of bowline's aims is to increase the number of independent implementations and improve attack resilience. Another key point is that IIJ has complete control over it. Having it under our control means we should be able to quickly implement countermeasures against new attack methods and replace running servers.

We implemented bowline in Haskell, and we have released it as open source. Our reasons for using Haskell are explained in "3.2 Why Implement it in Haskell?" in our previous article "Implementing QUIC in Haskell"[*1]. The most important point is that Haskell provides lightweight threads (simply referred to as "threads" below), which makes it possible to structure software more flexibly and with better clarity than with event-driven programming.

2.1 Development History and Library Structure

In 2010, as an anti-spam initiative, the author implemented a framework in Haskell that integrates SPF, Sender ID, DomainKeys, and DKIM. To use these technologies, DNS lookup functionality is essential. Initially, I used a well-known DNS stub resolver library written in C through a foreign function interface (FFI), but I discovered that it did not work well under Haskell's advanced concurrency due to frequent assertion failures.

I therefore stopped using that library and developed a DNS stub resolver library written entirely in Haskell (named dns). Because of the language's features, writing everything in Haskell makes it much easier to achieve high concurrency. Indeed, Haskell's practical applicability has been demonstrated in multiple Internet services.

I started developing bowline in 2022 together with my colleague Hibino. Hibino handled the key functions of a full resolver: iterative resolution, caching, and DNSSEC validation. I have been developing libraries in Haskell for HTTP/2, TLS 1.3, QUIC, and HTTP/3 since 2013, and I was interested in applying them to DNS, so I primarily developed the transport layer.

The prototype of bowline used the dns library, but in our experience, dns lacked extensibility and also had memory fragmentation issues. To resolve these shortcomings without worrying about backward compatibility, we decided to develop a new suite of DNS libraries. The libraries are divided by function and all begin with the prefix dnsext. They provide the following functionality.

- dnsext-types: Extensible, non-fragmenting basic data types and encoders/decoders for basic RRs (Resource Records)
- dnsext-dnssec: Encoders/decoders for DNSSEC-related RRs, DNSSEC validator
- dnsext-svcb: Encoders/decoders for the recently standardized SVCB (Service Binding) RR
- dnsext-utils: Utility functions such as logging and caching
- dnsext-do53: Client-side DNS over UDP and TCP
- dnsext-dox: Client-side DNS over HTTP/2, HTTP/3, TLS, and QUIC
- dnsext-iterative: Iterative resolution algorithm; server-side DNS over UDP, TCP, HTTP/2, HTTP/3, TLS, and QUIC
- dnsext-bowline: The bowline full resolver, the dug DNS lookup command[*2], and the ddrd daemon[*3], which enables discovery of encrypted DNS servers

dnsext-dnssec and dnsext-svcb serve as examples of how dnsext-types can be extended.

*1 Internet Infrastructure Review (IIR) Vol. 52, "Implementing QUIC in Haskell" (https://www.iij.ad.jp/en/dev/iir/052.html).
*2 IIJ Engineers Blog: "Introducing the DNS lookup command dug" (https://eng-blog.iij.ad.jp/archives/27527, in Japanese).
*3 IIJ Engineers Blog: "Discovering encrypted DNS servers" (https://eng-blog.iij.ad.jp/archives/31843, in Japanese).

2.2 Thread Structure

Generally, a full resolver is expected to operate as follows.

- Receive recursive query requests (Recursion Desired flag on) from stub resolvers
- Search the cache for each request, and if a "positive response" or "negative response" exists, return it to the stub resolver
- If not found, repeatedly send iterative queries (Recursion Desired flag off) to authoritative servers, obtain a positive or negative response, return it to the stub resolver, and store a new entry in the cache

Iterative resolution using the network takes much more time than cache lookups, which are a memory operation. So the software must be designed so that iterative resolution does not adversely affect the processing of other requests and responses.

Historically, UDP has been the primary transport for DNS, and when multiple requests arrive from the same stub resolver, the full resolver returns responses as soon as each RR is resolved. With connection-oriented transports, meanwhile, the full resolver receives requests in the order the stub resolver sends them. If the resolver tries to return responses in that same order, iterative resolution for one request can block responses to subsequent requests. To avoid this, connection-oriented transports require the resolver to return responses as they become available (pipelining). In other words, even with connection-oriented transports, the resolver must behave in the same way it does over UDP.

Suppose we designed the system such that a single thread handles both cache lookups and iterative resolution. That thread may block while performing iterative resolution. To process subsequent requests without delay, the system would then need to spawn a new thread for every request. With this design, the number of threads responsible for core functionality could become huge, making the system fragile.

We therefore decided to use separate worker threads for cache lookups ("cache-lookup workers") and iterative resolution ("iterative-resolution workers"). On a cache miss, a cache-lookup worker delegates iterative resolution to an iterative-resolution worker. Cache-lookup workers do not block, but iterative-resolution workers may block. A fixed number of cache-lookup workers and iterative-resolution workers are created at server startup. Smooth pipelining can be achieved if the number of iterative-resolution workers is sufficiently larger than the number of cache-lookup workers.

Figure 1 shows the thread structure we designed given the above considerations. The dashed rectangle represents bowline as a whole, and the gray rectangles represent threads. The receiver and sender handle the transport. For UDP, one pair stays resident per network interface. For

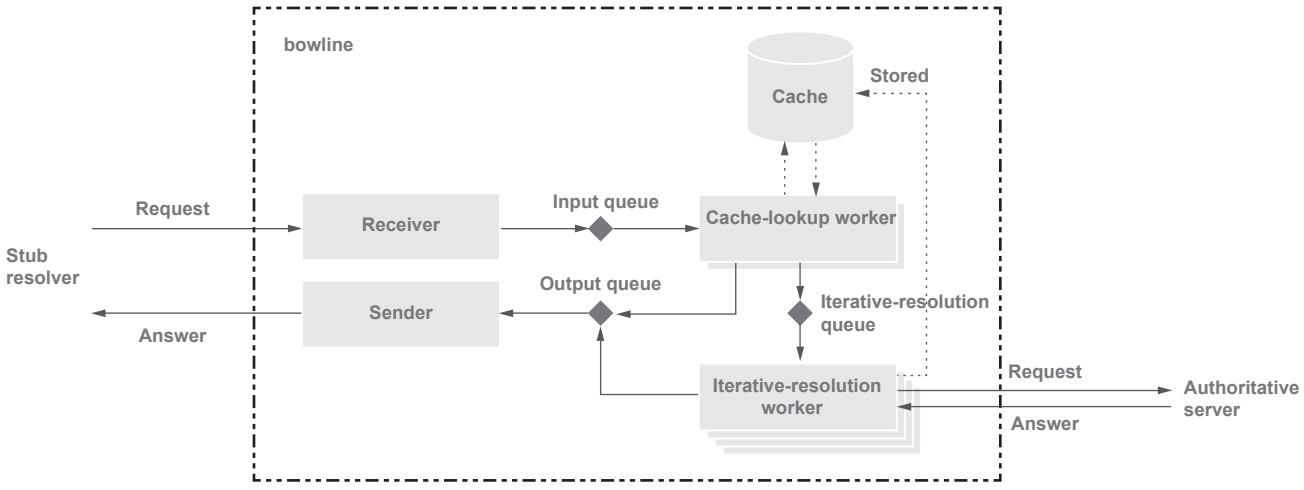


Figure 1: bowline's Thread Structure

connection-oriented transports, one listener thread stays resident per network interface, and a receiver/sender pair is spawned each time a connection is created. For HTTP/2, QUIC, and HTTP/3, auxiliary threads are also started.

The receiver decodes DNS requests from the stub resolver, converts them to a Haskell data type called Input, and places them in the global input queue. The Input contains a reference to the output queue of the sender corresponding to that receiver.

A cache-lookup worker receives an Input from the global input queue and searches the cache. This operation does not block. If the lookup succeeds, it represents the result in a data type called Output and stores it in the output queue referenced by the Input. If it fails, it relays the Input to the global iterative-resolution queue. By default, bowline has four cache-lookup workers.

An iterative-resolution worker reads an Input from the iterative-resolution queue and performs iterative resolution. This operation may block. If all iterative-resolution workers are blocked, the entire iterative-resolution subsystem cannot make progress, so the number of iterative-resolution workers must be sufficiently large. Each iterative-resolution worker represents the result of iterative resolution as an Output and stores it in the output queue. By default, bowline uses 128 iterative-resolution workers.

The sender reads Output from its own output queue, encodes it into a DNS response, and sends it to the stub resolver.

2.3 Iterative Resolution Algorithm

Authoritative DNS servers behave as follows in response to client queries.

- If the queried domain name is in a zone the server manages and an RR of the queried RR type exists, it returns that value
- If the queried domain name is in a zone the server manages and the query name does not exist or an RR of the queried RR type does not exist, it returns the SOA RR to provide the TTL for caching the negative response
- If the queried domain name is in a zone the server manages and delegation information for a child zone exists, it returns the delegation information (NS RR) and glue (A RR or AAAA RR) for the child zone so that the resolver can continue iterative resolution

In iterative resolution, the full resolver acts as a client when querying authoritative servers. In the RFC 1034 iterative algorithm, the query name and query type remain fixed to the values specified by the stub resolver.

Suppose, for example, that the stub resolver requests resolution of the TXT RR for www.example.jp. The full resolver ultimately queries the authoritative servers for "example.jp." for the TXT RR of www.example.jp, and similarly uses the same request when querying the root (".") authoritative servers and the intermediate "jp." authoritative servers.

This mechanism arguably gives Internet eavesdroppers many opportunities to observe the original query name and query RR type. So in the interests of protecting privacy, QNAME minimization was proposed. In QNAME minimization, authoritative name servers are queried using only the necessary portions of the original domain name. And for the intermediate query RR type, RFC 7816 uses NS RRs, while RFC 9156 uses A or AAAA RRs.

Table 1 summarizes the query names and query RR types used when resolving the TXT RR for www.example.jp.; bowline only uses QNAME minimization as defined in RFC 9156 for iterative resolution. The RFC 9156 column in Table 1 is explained in detail below.

1. Query the "." authoritative server for the A RR of "jp." to obtain the authoritative servers for "jp."
2. Query the "jp." authoritative server for the A RR of "example.jp." to obtain the authoritative servers for "example.jp."
3. Query the "example.jp." authoritative server for the

A RR of “www.example.jp.” and receive an answer, indicating there is no further delegation.

4. Query the “example.jp.” authoritative server for the TXT RR of “www.example.jp.”

The seemingly redundant query at the end occurs to check for delegation while hiding the original query RR type. If the original query RR type happens to match the RR type used by the algorithm, the number of queries is reduced by one.

If the query name exists as expected, one or more authoritative server names are obtained at each stage, and each authoritative server has one or more A/AAAA RRs. As mentioned above, A/AAAA RRs may be returned as glue along with NS RRs.

2.3.1 Implementation of Iterative Resolution

When bowline receives a request from a stub resolver, it first performs root priming before beginning iterative resolution against authoritative servers as a client. Root priming involves querying the built-in candidate authoritative servers for “.” (hint information) for the latest NS RR of “.” and simultaneously resolving the A/AAAA RRs. The result is cached. So as long as it is within its validity period, subsequent requests will use the cached value.

Once this special processing completes, bowline repeats the following steps until it obtains the final answer. First, it forms two groups: authoritative servers whose IP addresses are known and those whose IP addresses are unknown.

Next, for authoritative servers with known IP addresses, it randomly shuffles the IP addresses and selects two such that the authoritative server names do not overlap. It then queries those servers in parallel with requests following the QNAME minimization algorithm.

In our implementation, this client functionality is provided by the dnsext-do53 library, which can spawn query threads for multiple servers to race against each other and take the first response returned.

If the queries fail, it moves on to the next two candidates. If all queries to authoritative servers with known IP addresses fail, it moves on to the authoritative servers with unknown IP addresses. At this stage, it randomly shuffles the authoritative servers and takes the first one, selects either A or AAAA at random, and performs a new iterative resolution for the authoritative server name. If resolution fails, it proceeds to the next name. If resolution succeeds, it performs the intended query for this step against one of the obtained IP addresses.

If all queries fail, the overall lookup fails and an error is returned to the stub resolver. If any attempt succeeds and it obtains an exact match for the target RR, iterative resolution is complete. Otherwise, because it has obtained the names of lower-level authoritative servers, it repeats this step using a query name one label longer.

Table 1: Examples of QNAME Minimization

	Authoritative server	RFC 1034	RFC 7816	RFC 9156
1	.	www.example.jp. TXT	jp. NS	jp. A
2	jp.	www.example.jp. TXT	example.jp. NS	example.jp. A
3	example.jp.	www.example.jp. TXT	www.example.jp. NS	www.example.jp. A
4	example.jp.		www.example.jp. TXT	www.example.jp. TXT

2.3.2 Iterative Resolution Error Cases

NODATA is an error where the domain name exists but there is no value for the queried RR type (values for other RR types exist) and there is no next authoritative server to query. In the RFC 1034 algorithm, which sends the full query name, if NODATA is returned, that is the final result. But with QNAME minimization, the resolver must extend the query name and continue searching. For example, querying the ".jp." authoritative server for "ad.jp." yields NODATA, but querying for "iij.ad.jp." yields authoritative server information. This is because "jp." and "ad.jp." are the same zone.

NXDOMAIN is an error indicating that the domain name does not exist. If an intermediate domain name results in NXDOMAIN, RFC 8020 specifies that no domain names exist beneath it. In practice, however, looking up a lower-level name may still succeed. So even if an intermediate name returns NXDOMAIN, bowline continues querying with longer domain names. Only when the queried name ultimately returns NXDOMAIN does it report this error.

SERVFAIL indicates a server failure, REFUSED that the query was rejected for some reason, and FORMERR a format error. When these errors occur, bowline falls back to the next candidate IP address and continues searching.

2.3.3 Visualization with dug

The aforementioned dug is a DNS lookup command with two modes. In one mode, it acts as a simple stub resolver and requests recursive resolution from a full resolver. It can also query authoritative servers with the Recursion Desired flag turned off.

In the other mode, it uses the same iterative resolution implementation as bowline and provides a visualization of the process. The block on the right-hand side of the page shows an example of iterative resolution using dug. The -i flag specifies iterative mode, -vv increases verbosity, and +cdflag (check disabled) disables DNSSEC validation.

```
% dug -i -vv iij.ad.jp. txt +cdflag
...
root-priming: query "." NS
    query @2001:7fe::53#53/UDP "." NS
    query @198.97.190.53#53/UDP "." NS
    query @2001:7fe::53#53/UDP "." NS: win
root-priming: verification success - RRSIG of NS: "."
    "a.root-servers.net." 198.41.0.4#53 2001:503:ba3e::2:30#53
    "b.root-servers.net." 170.247.170.2#53 2801:1b8:10::b#53
    "c.root-servers.net." 192.33.4.12#53 2001:500:2::c#53
    "d.root-servers.net." 199.7.91.13#53 2001:500:2d::d#53 (※1)
    "e.root-servers.net." 192.203.230.10#53 2001:500:a8::e#53
    "f.root-servers.net." 192.5.5.241#53 2001:500:2f::f#53
    "g.root-servers.net." 192.112.36.4#53 2001:500:12::d0d#53
    "h.root-servers.net." 198.97.190.53#53 (※2) 2001:500:1::53#53
    "i.root-servers.net." 192.36.148.17#53 2001:7fe::53#53
    "j.root-servers.net." 192.58.128.30#53 2001:503:c27::2:30#53
    "k.root-servers.net." 193.0.14.129#53 2001:7fd::1#53
    "l.root-servers.net." 199.7.83.42#53 2001:500:9f::42#53
    "m.root-servers.net." 202.12.27.33#53 2001:dc3::35#53
iterative: query "jp." A
    query @2001:500:2d::d#53/UDP "jp." A (※1)
    query @198.97.190.53#53/UDP "jp." A (※2)
    query @2001:500:2d::d#53/UDP "jp." A: win
delegation - no DS, check disabled: "." -> "jp."
zone: "jp.":
    "a.dns.jp." 203.119.1.1#53 2001:dc4::1#53
    "b.dns.jp." 202.12.30.131#53 2001:dc2::1#53 (※4)
    "c.dns.jp." 156.154.100.5#53 (※5) 2001:502:ad09::5#53
    "d.dns.jp." 210.138.175.244#53 2001:240::53#53
    "e.dns.jp." 192.50.43.53#53 2001:200:c000::35#53
    "f.dns.jp." 150.100.6.8#53 (※3) (※6) 2001:2f8:0:100::153#53
    "g.dns.jp." 203.119.40.1#53
    "h.dns.jp." 161.232.72.25#53 2a01:8840:1bc::25#53
iterative: query "ad.jp." A
    query @150.100.6.8#53/UDP "ad.jp." A (※3)
    query @2001:dc2::1#53/UDP "ad.jp." A (※4)
    query @150.100.6.8#53/UDP "ad.jp." A: win
delegation - no delegation: "jp." -> "ad.jp."
cache-soa: no verification - check-disabled: "jp."
iterative: query "iij.ad.jp." A
    query @156.154.100.5#53/UDP "iij.ad.jp." A (※5)
    query @150.100.6.8#53/UDP "iij.ad.jp." A (※6)
    query @156.154.100.5#53/UDP "iij.ad.jp." A: win
delegation - no DS, check disabled: "jp." -> "iij.ad.jp."
zone: "iij.ad.jp.":
    "dns0.iij.ad.jp." 210.130.0.5#53 2001:240::105#53 (※8)
    "dns1.iij.ad.jp." 210.130.1.5#53 (※7) 2001:240::115#53
resolve-exact: query "iij.ad.jp." TXT
    query @210.130.1.5#53/UDP "iij.ad.jp." TXT (※7)
    query @2001:240::105#53/UDP "iij.ad.jp." TXT (※8)
    query @2001:240::105#53/UDP "iij.ad.jp." TXT: win
no verification - check-disabled: "iij.ad.jp."
;; HEADER SECTION:
;Standard query, NoError, id: 0
;Flags: Recursion Available

;; QUESTION SECTION:
;iij.ad.jp.               IN          TXT

;; ANSWER SECTION:
iij.ad.jp.   3600(1 hour) IN          TXT
"20f10da4-fb66-42ac-941e-133d9c6c09ba"
iij.ad.jp.   3600(1 hour) IN          TXT
"v=spf1 include:spf.iij.ad.jp include:spf.dox.jp -all"
iij.ad.jp.   3600(1 hour) IN          TXT
"_globalsign-domain-verification=qSNEOr9rmlIgx-CLJgTN-uypossXQTKD4GuWN-Jwp0"

;; AUTHORITY SECTION:

;; ADDITIONAL SECTION:

;; 172usec
```

Of the candidate authoritative server IP addresses, those actually used are annotated with "(※number)." "win" indicates which of the two competing lookups won. If there is delegation, the IP addresses are listed; otherwise, "no delegation" is displayed.

2.4 DNSSEC Iterative Resolution Algorithm

DNSSEC uses digital signatures, a type of public-key cryptography, to establish an authentication chain for delegation information. There are two DNSSEC-related RRs within a zone.

- DNSKEY: The public key used to verify digital signatures (RRSIGs) provided by the zone (DNSKEYs include KSKs and ZSKs, but we do not distinguish them in this article)
- RRSIG: A digital signature over RRs managed by the zone

Verifying an RRSIG RR using a DNSKEY RR allows us to confirm that the data has not been tampered with. However, we cannot tell whether the claimed domain name is truly legitimate. We therefore need to prove that it is delegated from the parent domain. The following RR is provided for this purpose.

- DS: An RR for registering the cryptographic hash value of a zone's DNSKEY in the parent zone

For iterative resolution requests with the DO (DNSSEC OK) flag on, authoritative servers behave as follows.

- If there is delegation to a child zone: when returning delegation information such as NS RRs, return DS RRs as well
- If there is no delegation to a child zone: If the returned RRs have corresponding RRSIG RRs, return those as well

The validation of delegation combined with QNAME minimization ends up being somewhat complex, so rather than providing a general explanation, we use Figure 2 to briefly explain how bowline resolves the A RR for "www.example.jp". Note that we assume the DS RR for "." is provided in advance as a trust anchor.

- Query a built-in candidate "." authoritative server for the "." DNSKEY RR (b), and select the DNSKEY RR that matches the hash value (a) for "." provided as a trust anchor. The response also includes the RRSIG RR over the DNSKEY RR (c). Using the public key contained in the "." DNSKEY RR, verify the signature in the RRSIG RR. If verification succeeds, trust the "." DNSKEY RR.

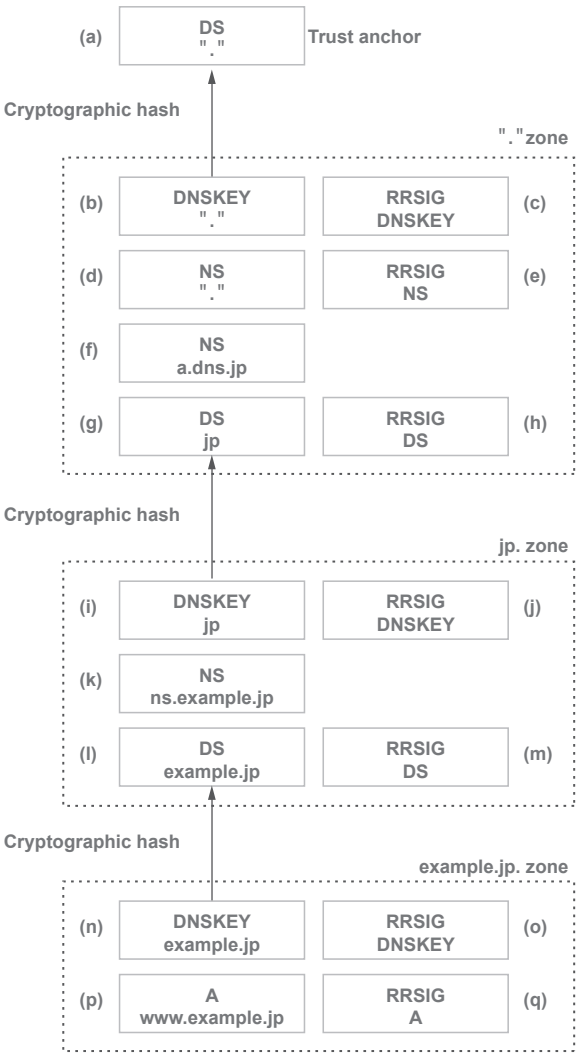


Figure 2: DNSSEC Authentication Chain

- Query a built-in candidate "." authoritative server for the NS RR of "." (d) to obtain the list of "." authoritative servers. Verify the signature in the accompanying RRSIG RR (e) with the "." public key, and trust the "." authoritative servers (root priming).
- Query a "." authoritative server for the A RR of "jp." This yields NS RRs (f) indicating "jp." authoritative servers and A/AAAA RRs, but there is no corresponding RRSIG because this is not information managed by the "." zone. It also yields the DS RR (g) containing the hash of the DNSKEY RR of "jp." with an accompanying RRSIG RR (h). Verify this signature with the "." public key. If successful, trust the delegation from "." to "jp."
- Query a "jp." authoritative server for the DNSKEY RR of "jp." (i), and extract the DNSKEY matching the hash for "jp." The response includes the DNSKEY RR and its RRSIG RR (j). If verification of the signature using the "jp." public key contained in the DNSKEY RR succeeds, trust the "jp." DNSKEY RR.
- Query a "jp." authoritative server for the A RR of "example.jp." This yields NS RRs (k) and A/AAAA RRs indicating "example.jp." authoritative servers. It also yields the DS RR (l) containing the hash of the DNSKEY RR of "example.jp." along with its RRSIG RR (m). Verify this signature with the "jp." public key. If successful, trust the delegation from "jp." to "example.jp."
- Query an "example.jp." authoritative server for the DNSKEY RR of "example.jp." (n), and extract the DNSKEY matching the hash for "example.jp." The response includes the RRSIG RR (o) over the DNSKEY. Using the public key contained in the "example.jp." DNSKEY, verify the signature in the RRSIG RR. If verification succeeds, trust the "example.jp." DNSKEY RR.
- Query an "example.jp." authoritative server for the A RR of "www.example.jp." (p), which also yields the RRSIG RR (q). Verify this signature and trust the final answer.

Since not all zones support DNSSEC, the chain of trust may be broken mid-way. In bowline, if the chain of trust is broken, the DO flag is turned off for subsequent iterative resolution.

2.4.1 Parent–Child Coexistence Problem

As described above, authoritative servers return DS RRs in response to queries for names in a child zone so that the resolver can continue the lookup. So in most cases it is not necessary to explicitly query for DS RRs. But there are cases in which DS RRs may not be returned even though the chain of trust has not been broken.

This occurs when the parent zone and the child zone are managed by the same authoritative server. As an example, let the parent zone be "a." and the child zone be "b.a.", and assume that both zones are managed by the same authoritative name server.

Suppose we query the authoritative server for "a." requesting the A RR of "b.a." If the parent and child are not co-located, the server returns delegation information for "b.a.", namely NS RRs and DS RRs. But in this example, the parent and child are co-located, and because a DNS query does not specify which zone it targets, the longest-match rule applies and the "b.a." zone is taken to be the target. If an A RR exists, it is returned; otherwise an SOA RR is returned, and no DS RR is obtained.

In this case, we need to determine that delegation exists even though a DS RR was not returned, and explicitly query for the DS RR. The method bowline uses is as follows.

- If an A RR for "b.a." exists, then an RRSIG RR is returned along with it. If the Signer's Name field in the RRSIG RR is "b.a.", then the parent and child are co-located.
- If an A RR for "b.a." does not exist, an SOA RR is returned. Extract its domain name (the RR's NAME field). If it is "b.a.", the parent and child are co-located.

2.4.2 DO Flag Specified by Stub Resolver

The DO flag is also used by the stub resolver to tell the full resolver whether it supports DNSSEC. The full resolver behaves as follows.

If the DO flag is off: Validate DNSSEC as much as possible, strip DNSSEC-related RRs from the response, and do not set the response's AD (Authenticated Data) flag.

If the DO flag is on: Return all DNSSEC-related RRs as well. If all validations succeed, set the response's AD flag.

If delegation for DNSSEC disappears partway through, do not set the response's AD flag. If validation fails anywhere, return SERVFAIL.

2.4.3 Proof of Non-Existence

DNSSEC also includes proof of non-existence (NSEC/NSEC3 RRs). The technical details are beyond the scope of this article, but for details, see "Integrating DNSSEC into a DNS Full Resolver Implementation—Proving Negative Responses with NSEC/NSEC3"[*4].

2.5 Cache Data Structure

For cache data structure, we use a PSQ (Priority Search Queue), which has properties of both a search tree and a priority queue. The key is the request (domain name, RR type, etc.), the priority is the TTL (Time To Live), and the value is the "iterative resolution result." In other words, we can efficiently look up an iterative resolution result from a request and also delete cache entries according to TTL.

Positive responses are divided into (1) those without DNSSEC signatures, (2) those with signatures that are not validated, and (3) those for which signature validation succeeds. The cache lifetime is basically the RR's TTL. But for DNSSEC it is also constrained by the RRSIG RR's TTL and the signature validity period.

Negative responses (NODATA and NXDOMAIN) are not divided up; they are cached using the TTL value obtained from the SOA RR. If the response is DNSSEC-signed, NSEC/NSEC3 RRs and RRSIG RRs are also returned. These serve as proof of non-existence for NODATA or NXDOMAIN, so they are cached together with the SOA.

For SERVFAIL, REFUSED, and FORMERR, an SOA RR cannot be obtained. Since we want to obtain a positive response or a NODATA/NXDOMAIN response if possible, we fall back to the next candidate IP address. If the candidates are exhausted, we cache these error responses to prevent attacks that repeatedly send the same query. For the TTL, we use the value specified in the configuration file.

Responses are assigned a priority called ranking. Glue information obtained from responses with the AA (Authoritative Answer) flag off has lower priority than information obtained

*4 IIJ Engineers Blog: "Integrating DNSSEC into a DNS Full Resolver Implementation—Proving Negative Responses with NSEC/NSEC3" (https://eng-blog.iij.ad.jp/archives/24512, in Japanese).

from authoritative responses (AA flag on). So the cache entries for the former are overwritten once the latter are obtained. Even if glue information exists in the cache, the full resolver does not return it to the stub resolver but instead returns results only after obtaining authoritative information.

2.6 How Much Control Did We Have?

One of bowline's goals was to enable us to respond quickly when problems were identified. In this section, we present examples of rapid responses that show we have achieved this.

First, let's discuss vulnerabilities. During bowline's development, the operations team told us about attacks related to the number of domain name compression pointers, and we also learned of the KeyTrap attack from external sources. As these vulnerabilities were present in bowline, we immediately fixed them.

As for issues discovered while testing bowline, the operations team reported that communication sometimes failed when a stub resolver used QUIC and HTTP/3 as transport. The Haskell quic library we were using at the time used connected UDP sockets to improve performance. With connected sockets, if an intermediate NAT changes the port number, packets no longer arrive. The quic library has a migration feature enabling it to maintain the connection using a new connected socket if the port changes after a QUIC connection has been established.

However, the NAT that caused this phenomenon was changing the port during the process of establishing a connection. To address this, we added major modifications to make use of unconnected sockets, which are common with UDP usage, although this does result in lower performance.

Further, and this really concerns dug rather than bowline, some encrypted DNS servers were returning multiple TLS session tickets. At the time, the Haskell tls library was implemented on the assumption that there would only be one session ticket. We discovered that when one of the session tickets returned by encrypted DNS servers was selected, session resumption would sometimes fail. Presumably, there are multiple TLS endpoints that all return their session tickets, such that when only one is selected, it may not be what the receiving TLS endpoint expects. So we modified the tls library to use all session tickets.

2.7 Conclusion

The bowline site can be found at the following URL. It includes links to Linux and macOS binaries, instructions for using bowline from Docker Hub, and installation instructions for Debian.

- https://iijlab.github.io/dnsext/bowline.html

The bowline source code can be obtained from GitHub at the following URL. We eventually plan to register it on Hackage, the Haskell package archive, so that users can easily build it using the Haskell build system as well.

- https://github.com/iijlab/dnsext

We provide progress updates on the bowline project at the following URL.

- https://www.iijlab.net/projects/Underpinning/dns.html

Last but not least, I would like to thank my colleagues at IIJ for their insightful comments on draft versions of this article.

Kazuhiko Yamamoto
Development Group Leader, IIJ Research Laboratory
In 2022, Dr. Yamamoto transitioned from full-time to contract employment at IIJ and relocated to his hometown in Yamaguchi Prefecture, where he works remotely. He has recently been chasing Japanese Spanish mackerel and bigfin reef squid in the Seto Inland Sea.

Satellite Internet: Old Yet New—How Starlink Is Changing the World

Starlink's emergence has transformed the conventional view of satellite[*1] Internet. And, inspired by Starlink, a host of new players also continue to emerge. In this focused research report, we look back on the history of satellite Internet and take a closer look at Starlink. We also explore the possibilities for where satellite Internet may be headed.

3.1 Satellite Internet

3.1.1 Satellite Orbit Types and Characteristics

The orbits of satellites circling Earth can be classified into three types based on altitude: GEO, MEO, and LEO (Figure 1). To summarize the characteristics of each, a GEO (Geostationary Earth Orbit) is at an altitude of around 36,000km and is synchronized with Earth's rotation. GEO satellites remain in a fixed position over ground stations and can cover a wide area, but with the drawback of large latency. A MEO (Medium Earth Orbit) ranges from around 2,000km to 35,000km and is used for systems such as GPS satellites. MEO offers moderate latency and balanced coverage. LEO (Low Earth Orbit) ranges from around 160km to 2,000km. It has the advantages of low latency and high bandwidth, but covering the entire Earth requires many satellites, increasing operational costs (Table 1).

3.1.2 The Era of Research Networks

SATNET (Satellite Network, also known as the Atlantic Packet Satellite Network) was a satellite-based network developed in the 1970s by the U.S. Defense Advanced Research Projects Agency (DARPA). This was an experimental project to connect ARPANET (Advanced Research Projects Agency Network) to Europe via satellite (Intelsat IV) and can be regarded as the forerunner of satellite Internet.

Research from Japan includes the AI3 (Asian Internet Interconnection Initiatives) project by WIDE Project (Widely Integrated Distributed Environment Project). Using satellites from Japan's JSAT (SKY Perfect JSAT) to connect research institutions across the Asia-Pacific region, especially Southeast Asia, the purpose of AI3 is to build Internet infrastructure and conduct R&D on networking technologies. The project began some 20 years after the research on SATNET, during a period when the Internet was spreading across the globe as a research network. The project developed and deployed Unidirectional Links (UDL) and contributed to standardization of UDLR (Unidirectional Link Routing) technology. By operating IPv6 and multicast over UDL, it contributed to connectivity across a broader area of Asia.

3.1.3 Development of Commercial Services

Commercial Internet services using GEO satellites began in earnest in the late 1990s. In 1996, Hughes Network Systems launched DirecPC as the first consumer satellite broadband service. In 2004, WildBlue provided high-speed service using Ka-band satellites, but it was acquired by

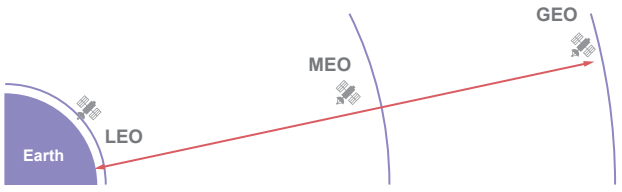


Figure 1: Relative Positions of Earth and Satellites

Table 1: Characteristics of Satellite Orbits

Characteristic	GEO (Geostationary Earth Orbit)	MEO (Medium Earth Orbit)	LEO (Low Earth Orbit)
Altitude	~36,000km	~2,000–35,000km	~160–2,000km
Latency	~500–600ms	~100–300ms	~25–80ms
Apparent motion	Stationary (always in same position)	Moves slowly	Moves rapidly
Orbital period	24 hours (synchronized with Earth's rotation)	Several hours	~90–120 minutes
Global coverage	A few satellites (3–4)	Several dozen satellites	100s–1000s of satellites
Coverage area	Wide (one satellite covers ~1/3 of Earth)	Medium (multiple for global coverage)	Narrow (many needed)

*1 "Satellite" in this article refers to artificial satellites, not natural satellites such as the Moon.

Viasat in 2009. Today, services are available from providers such as Hughesnet, Viasat, Konnect, SES Astra, Inmarsat, ExBird, SKY Perfect JSAT (EsBird), and China Satcom.

Commercial Internet services using MEO satellites began with the launch of four satellites in 2013 by O3b Networks, founded in 2007. O3b was acquired by SES in 2016, with its services now being offered as O3b mPOWER (second generation; launches began in 2022; operations began in April 2024) as part of SES Networks. Currently, 10 satellites are in service.

Turning to commercial services using LEO satellites, Iridium launched its service in November 1998 and Globalstar in February 1999. Both companies collapsed once but later recovered and continue to provide services, mainly for voice communications. The Teledesic broadband concept began in the early 1990s (founded in 1990) and progressed to FCC approval in 1997 as a competitor to Iridium/Globalstar, but it was discontinued in 2003.

OneWeb marked a leap forward for LEO, and Starlink was a major success. On the heels of Starlink, Project Kuiper is currently in preparation for launch, and Chinese players are also attempting to catch up.

OneWeb was conceived by Greg Wyler in 2012 and started operations in 2019. Overcoming funding difficulties and geopolitical challenges, it merged with Eutelsat in 2023. Now as Eutelsat OneWeb, it operates 648 satellites and is expanding its services mainly in Europe and the United States.

Starlink was born out of Elon Musk's vision. Announced in 2015, the project takes advantage of the mass production of satellites and reusable rocket technology, with over 8,500 satellites having been deployed. It has over 7.5 million users and provides service in over 150 countries.

Project Kuiper is an Amazon subsidiary established in 2019 and is preparing to launch services by the end of 2025. It has around 129 satellites in orbit.

Qianfan (Thousand Sails) is a Chinese satellite initiative led by Shanghai Spacecom Satellite Technology (SSST). Developed with Starlink in mind, it conducted its first launch in August 2024 and is reported to have around 90 satellites in orbit.

GuoWang is a Chinese national project led by China SatNet. It too is being developed with Starlink in mind. Its first launch was in December 2024, and it is reported to have around 95 satellites in orbit.

3.2 Starlink

3.2.1 Starlink Overview

Starlink is a service of SpaceX (Space Exploration Technologies Corp.), a private American company. Founder Elon Musk announced the idea in 2015. Since launching a limited beta service in 2019 with 60 satellites, it has grown rapidly. As of October 2025, it has over 8,500 or so satellites in orbit, with around 7.5 million users across more than 150 countries. In Japan, service has been available since October 2022.

Figure 2 shows Starlink's architecture.

Connections to the Internet are made via the User Terminal → Starlink constellation → gateway → POP → Internet.

A Starlink User Terminal is required to connect to Starlink. Aside from purchasing one from the official website or

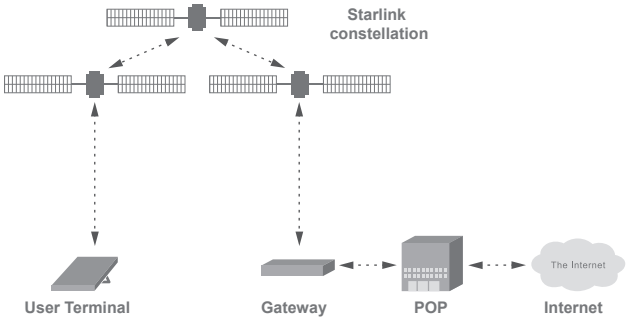


Figure 2: The Starlink Architecture

authorized retailers, there are various other ways to obtain one.

Several types of User Terminal are available. In addition to the Standard model, there is an ultra-compact Mini, as well as a Performance model designed for harsh environments and maritime use (Figure 3).

The User Terminal uses its location and Starlink satellite orbital data to select a Starlink satellite with which it can communicate. A phased-array antenna electronically steers the beam automatically, so users do not need to make any adjustments. Starlink satellites operate in low Earth orbit and form a constellation. Around 8,500 satellites are said to be in service, but because satellites are continually being added and maintenance is ongoing, the exact number is known only to SpaceX.

The satellitemap.space website for tracking satellite constellations and satellites is quite well made. It uses public data from space-track.org to provide a visualization of constellations and satellites based on orbital information (Figures 4, 5, and 6).

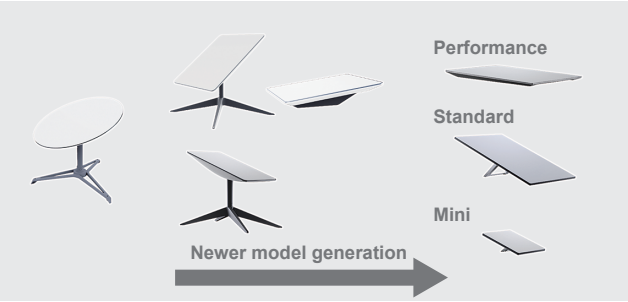


Figure 3: Different Generations of the User Terminal
The design transitioned from a movable stand to a kickstand.

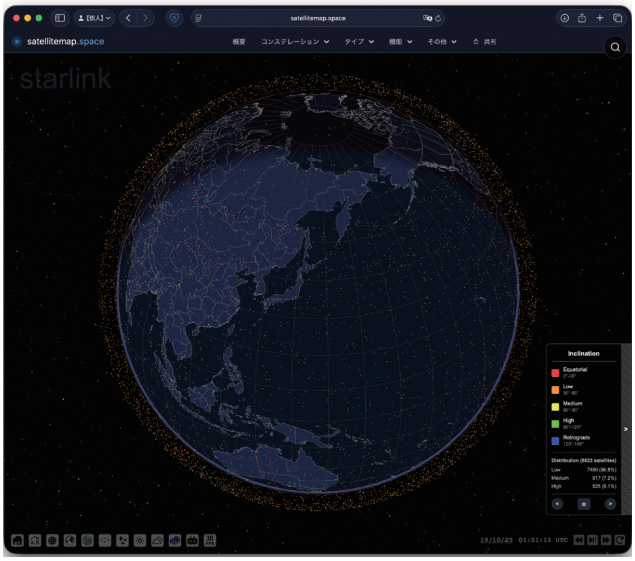


Figure 4: Starlink Satellites Distributed Around the Globe
At this point, there appear to be 8,622 satellites.

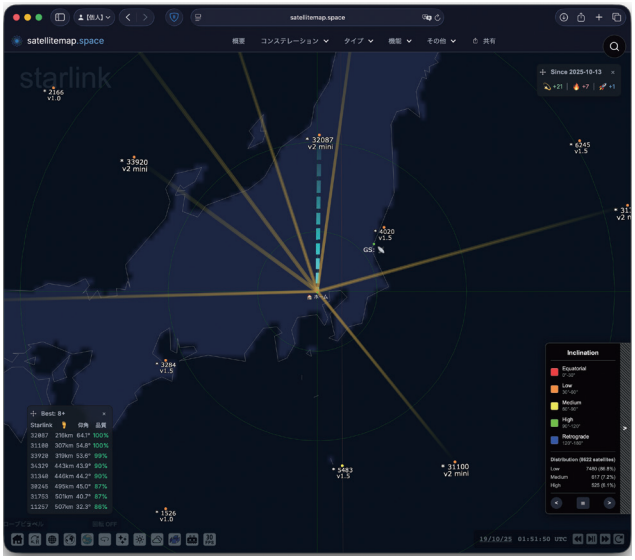


Figure 5: Satellite Positions as seen from Tokyo (Author's Location) at a Particular Time
This depicts the User Terminal selecting a nearby satellite. The selection algorithm is undisclosed, so this is strictly a guess.

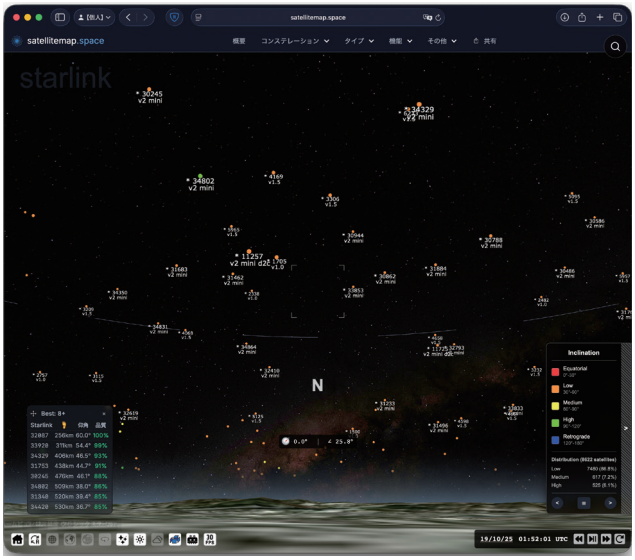


Figure 6: Positions of Satellites Visible from the Ground in Tokyo
Several dozen satellites are within the field of view.

Since SpaceX does not disclose the visualization algorithm, the information is provided solely for reference purposes, but it should convey the general picture.

Starlink currently has V1.5 and V2mini satellites in service. The upcoming generation of satellite is the V3, which is to be deployed on SpaceX's new Starship rocket system (Figure 7). V3 reportedly has 10 times the performance of previous satellites, and together with new model User Terminals, the plan is to support communication speeds of 1 Gbps.

SpaceX plans to improve the capabilities of its overall constellation by adding more satellites and replacing legacy models with newer ones.

Gateways serve as bridges between satellites and the terrestrial Internet. They relay data from satellites and send it to POPs. They are known to have been installed at four locations in Japan: Hokkaido, Akita, Ibaraki, and Yamaguchi (Figure 8).

A POP (Point of Presence) is an Internet access point, and the segment from the User Terminal to the POP appears as a Layer 2 network. For IPv4, the POP assigns an ISP Shared Address (100.64.0.0/10) via DHCP, and users reach the Internet over IPv4 via CGNAT. Some plans allow users to obtain a public IPv4 address via DHCP. For IPv6, a routable public IPv6 prefix is delegated via DHCPv6-PD.

3.2.2 Starlink in Action

The volcanic eruption in Tonga on January 15, 2022, was one of the largest underwater eruptions in recorded history. Submarine cables were severed, and Tonga was cut off from data communications, but Starlink moved quickly and ended up being key to restoring connectivity. When Russia invaded Ukraine in February 2022, Starlink was rapidly deployed as an alternative to ground communication infrastructure that had been destroyed, supporting Ukraine's military and civilian communications. In the aftermath of Japan's Noto Peninsula earthquake of January 1, 2024, KDDI, SoftBank, and Docomo turned to Starlink to support the restoration of communications in affected areas. And Starlink has continued to play a role in disaster-stricken areas around the world since.

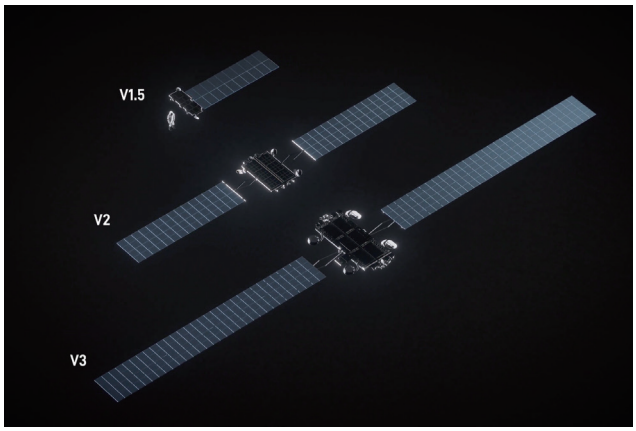


Figure 7: Starlink Satellite Size Comparison[*2]
V3 satellites are to be deployed once Starship enters service.

Figure 8: Gateways in Japan
There are four gateway sites: Hokkaido, Akita, Ibaraki, and Yamaguchi. Traffic is aggregated at the POP in Tokyo.

*2 SpaceX, October 24, 2025 post on X (https://x.com/SpaceX/status/1977873370688700846).

3.2.3 Starlink's Strengths

Starlink founder Elon Musk's vision is to "make humanity a multi-planetary species." The target is Mars. Fulfilling the roadmap to Mars will require massive levels of funding and technological breakthroughs. Starlink is the funding source for technological development (Figure 9).

The key to Starlink's success was the rocket (Falcon 9) it developed to carry satellites (Figure 10). A defining feature of the Falcon 9 is that the first stage and the fairing can be reused. Once used, the rocket returns toward the launch site and lands on a droneship positioned at sea, and from there it is recovered and then refurbished.

Once a rocket is recovered, it takes around 10 days to refurbish and make it reusable again. Reusability is evidently quite high, with some boosters having been reused over 30 times. Some 30 rockets are currently in service. In 2024, SpaceX carried out 132 launches, and as of this writing it had 170 launches planned for 2025. Each launch inserts 26 Starlink satellites into orbit, which translates into annual deployment capacity of over 4,400 satellites.

SpaceX also manufactures the many satellites it places into orbit. The rockets also have a second stage, which is not reused, and SpaceX manufactures those as well. With the ability to produce thousands of satellites, the infrastructure to launch them, the capabilities needed to operate a satellite constellation, the development and testing of new rockets, the vertical integration of operations from satellites through to rockets, and a strategy of funding operations with revenue from Starlink and launch services for other companies, SpaceX is taking on challenges no one has ever tackled before.

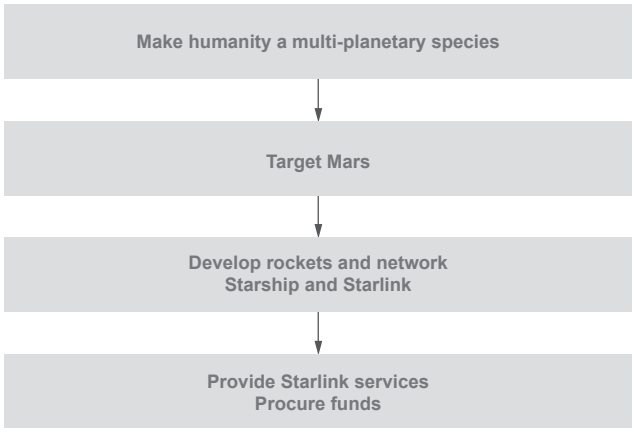


Figure 9: SpaceX's Purpose

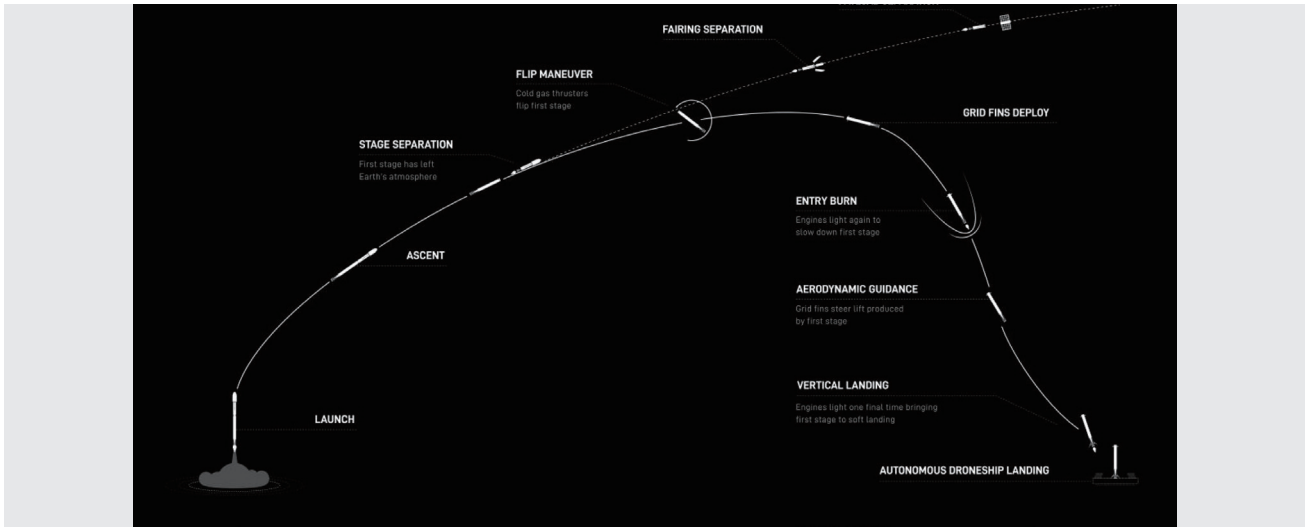


Figure 10: Falcon 9 Sample Mission Profile
The rocket makes an autonomous landing on a droneship[*3].

*3 SpaceX documentation (https://www.spacex.com/assets/media/falcon-users-guide-2025-05-09.pdf).

3.3 The Future of Satellite Internet

3.3.1 The Changing Face of Internet Infrastructure

The terrestrial communications network is what supports today's Internet. It uses optical fiber to provide not only land-based connections but also intercontinental connectivity via submarine cables, thereby linking the world. Even mobile wireless network base stations are connected by these communications paths. However, deploying and maintaining this terrestrial network is highly expensive.

Communications networks based on Low Earth Orbit satellites have emerged as a potentially new type of infrastructure. Sections previously deployed using optical fiber could be replaced by optical links between satellites. Satellites are equipped with laser communication devices called inter-satellite links (ISL), and building a mesh network across multiple Starlink satellites can facilitate the efficient processing of communications from the ground. It is because SpaceX has dramatically reduced the cost of launching satellites that we can now conceive of these possibilities.

3.3.2 Will Space-Based Communications Be Faster?

Optical links between satellites have some advantages over optical fiber. The speed of light in a vacuum is 300,000km/s, but it drops to around 200,000km/s in optical fiber. For example, the distance from the East to the West Coast of North America is about 8,000km. If the distance from the ground to a satellite is 550km each way, that puts the total path length at around 9,100km. Traveling 8,000km through terrestrial optical fiber takes light about 40ms. Via satellite, the distance increases to 9,100km, but in principle it should take around 30ms. If it does work out to 10ms faster one way or 20ms faster for a round trip, there appear to be many advantages for satellite Internet (Figure 11).

Starlink's inter-satellite laser links have already achieved speeds of 200 Gbps. There are plenty of terrestrial systems offering faster speeds, but if the satellite network becomes dense enough to maintain routes via a mesh, satellite-based infrastructure may offer advantages in more and more scenarios. Looking ahead, if laser-capable terminals are mass produced and adopted by users on the ground, low-cost, ultra-high-speed networks could conceivably be built on satellite infrastructure.

3.3.3 Extension to Interplanetary Communication

SpaceX's Mars plan involves the further evolution of Starlink. Communication time from Earth to Mars ranges from 3 to 22 minutes depending on planetary distance, meaning a round trip of 6 to 44 minutes. Absorbing such delays with conventional TCP/IP will be difficult. In the world of satellite Internet, research toward a multi-planetary future will no doubt continue. What sort of research outcomes will we see? How will this affect life on Earth? There is much to look forward to ahead.

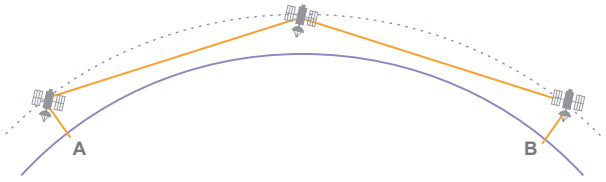


Figure 11: Relationship between Inter-satellite Communications and Distances on Earth
Simple comparisons are difficult due to a range of factors including the Earth's surface being curved, connections to satellites not always being made over the shortest distance, slack in submarine cables, and delays in repeaters.

Takashi Taniguchi
Fellow, Marketing Division, Marketing Management Unit; concurrently assigned to Space Business Promotion Office, IIJ
Mr. Taniguchi joined IIJ in 1995, working in infrastructure construction and operations for consumer services. From 2006, he was seconded to the games industry, where he worked on system development and operations for AAA titles and social games, returning to IIJ in 2022. Prompted by the emergence of Starlink, he became even more interested in the space field, where he draws on his past experience to take on bold new challenges. He has a cousin who is a world champion in fighting games.

Internal RAG Platform Using Generative AI and Extension to Multi-Agent Orchestration
—Implementation Technologies, Operational Efficiency Outcomes, and Future Outlook

4.1 Introduction

AI technologies, especially large language models (LLMs), have made huge advances in recent years thanks to companies such as OpenAI, Google, and Anthropic, dramatically changing how enterprises use knowledge. At IIJ, we also began running an internal RAG platform called sbdGPT in the summer of 2023. This article discusses the background to its development, our objectives, and the system architecture, as well as its extension to multi-agent orchestration and a proposal document generation tool integrated with the internal RAG platform.

4.2 Background to and Objectives of Developing an Internal RAG

IIJ provides over 100 services for business customers and has amassed a vast and diverse information base in doing so, encompassing manuals, knowledge-sharing sites, news, inquiry emails, and more. Because these information assets are distributed and managed across multiple different platforms, accessing them across systems and quickly searching for and retrieving the information needed has long been a major challenge for sales staff and engineers. As such, the time spent repetitively querying internal sources and discovering information has hindered operational efficiency.

Against this backdrop, we began developing an internal RAG platform with the goal of collecting and processing document data scattered throughout the company, consolidating it into a central database, and using generative AI to provide specific answers in accord with user questions.

4.3 Internal RAG Architecture and Data Optimization

The internal RAG platform integrates multiple data sources, with the data being converted into vector embeddings for storage in a database. With the exception of the LLM and embedding models, we use IIJ services and open-source software to keep costs down while enabling independent, flexible development iterations and operations management.

Figure 1 provides an overview of the architecture.

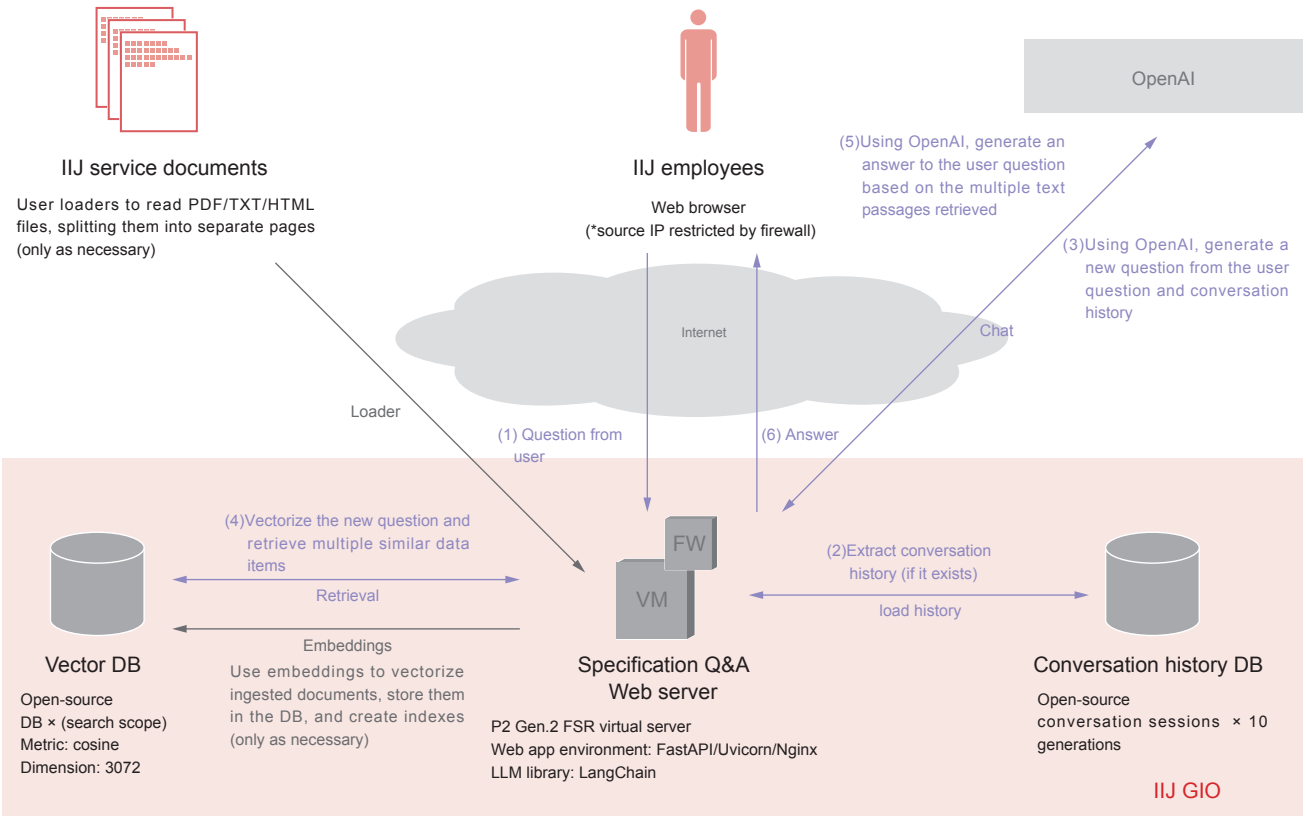


Figure 1: Overview of the Internal RAG Architecture

• Components

Knowledge base (IIJ service documents)
- Service detail materials (PDF files)
- Online manuals (HTML files)
- Internal technical QA emails (message files)
- Knowledge-sharing sites (HTML files)
- News articles (HTML files)

Web server platform
- IIJ GIO Infrastructure P2 Gen.2 Flexible Service VMs
- IIJ GIO Infrastructure P2 Gen.2 perimeter firewall
- FastAPI (OSS)
- Uvicorn/Gunicorn (OSS)
- Nginx (OSS)

Generative AI / Embedding models
- Models provided by OpenAI

LLM development framework
- LangChain (open-source version)

Vector DB
- ChromaDB (open-source version)

Conversation management DB
- PostgreSQL (open-source version)

*Except for the generative AI and embeddings, IIJ service infrastructure and open-source software are used.

Each knowledge dataset stored in the vector database targets files (PDFs, HTML, etc.) distributed across multiple platforms (Figure 2). We use the loaders LangChain provides for each file format to extract the content as text.

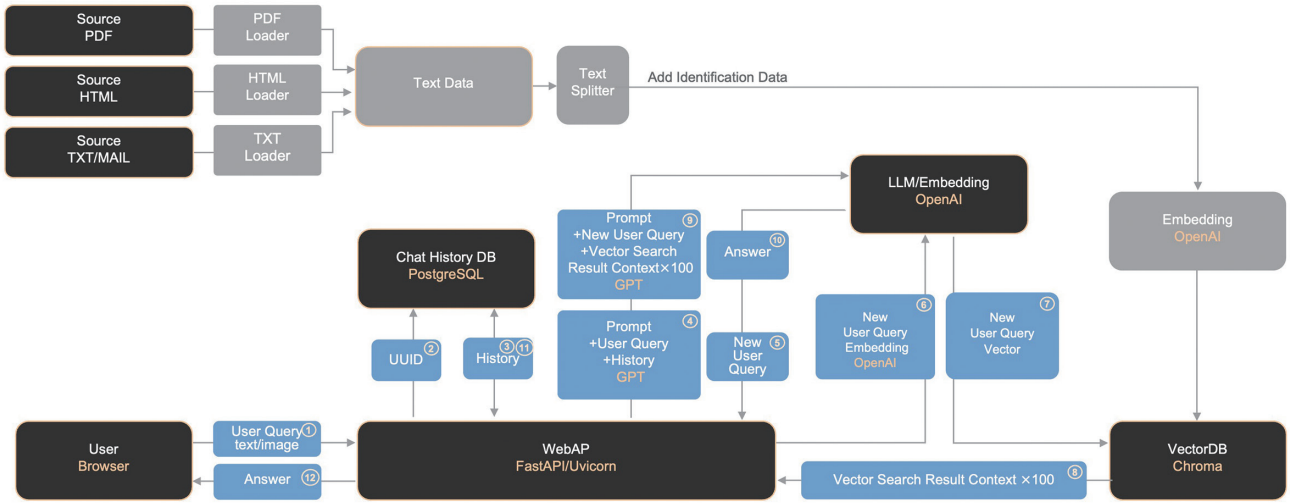


Figure 2: Operational Flow for the Internal RAG

Initially, we used the common approach of splitting the large volume of extracted text into appropriate lengths (chunking) using text splitters and stored it in the vector database in that form. In many cases, however, this approach failed to produce the expected answers to user queries.

Our investigation revealed that this was due to service documents being meaningfully (semantically) complete at the file level (PDF, HTML, etc.), such that short passages of text extracted in piecemeal fashion (e.g., chunks of data around 200 characters long) lacked sufficient contextual information about what was being said about which particular service. So semantic similarity was not being fully harnessed by the vector searches, and relevant information was not being properly retrieved as a result.

To address this, when chunking we now generate identification information from existing metadata contained in each file, such as the service name and title information, and attach it to each chunk. This has improved vector-search hit rates and answer accuracy.

4.4 Fact-Checking Measures

AI-generated answers may contain hallucinations (fabricated information). So fact-checking (verification of sources) is crucial when using AI-generated answers for business purposes. To ascertain what information the AI referenced when generating an answer, it is important to have the AI provide citations, and users need to check those citations to ensure the accuracy of the content.

Usability takes a big hit if the fact-checking process is complicated, however. So we built a mechanism into the tool to allow access to the data sources with minimal effort. Specifically, we implemented a feature that lets users bring up the source of the knowledge data (PDF, HTML, etc.) simply by clicking a citation link button displayed at the end of the generated answer.

Non-HTML data sources are stored on the Web server ahead of time and served statically via the browser, enabling immediate access to sources such as PDFs. This simplifies the user flow for determining the reliability of AI answers.

The identification information attached to each chunk also includes the data source file's FQDN (including page numbers in the case of PDFs). In the system prompt, we therefore specify a citation output format, as shown below, so that sources are presented immediately below the generated answer. This means users can simply click a button to display the citation details as text and instantly access the source file on the Web server.

```
# Example citation output format
<div hidden id="{{ id }}">
 <p class="cite_title">Cited Context 1</p>
<a href="{{ source file FQDN }}" target="_blank">{{  source file FQDN }}</a>
 <p>{{ context text }}</p>
 </div>
 <button class="open-button" onclick="refOpen('{{ id }}')">Main citations</button>
```

4.5 Business Efficiency Outcomes and Multi-Agent Extension of RAG Platform

Since we began using the internal RAG platform in the summer of 2023, usage over its roughly two years in operation has expanded to the point where around 30,000 queries are processed per month. This is helping achieve internal business efficiency gains on the order of 1,500 hours per month.

With the system in operation for some time, however, it also became clear that there are limitations to answers based on internal information using RAG alone. In particular, we found that in an increasing number of cases, it was difficult to provide appropriate answers without referring to information on the Internet such as the latest market trends, competitor comparisons, and differences in specifications vs. third-party products. The need to go beyond simple specification checks to provide higher-level decision-making support,

such as market research and trend analysis by sales staff and engineers, also started to emerge.

To address these issues, we extended the internal RAG platform and implemented a multi-agent architecture in which multiple specialized agents collaborate to generate answers. The conventional RAG mechanism vectorized the internal data (IIJ service manuals, knowledge-sharing sites, inquiry emails, etc.) and used generative AI to produce answers, but handling queries requiring external information, such as comparisons of IIJ's service specifications with those of other companies or those dealing with proposals based on the latest technology trends, was problematic.

To resolve this constraint, we refactored the internal RAG platform and built a multi-agent platform. The main agent automatically calls other agents as needed based on routing logic, such as the IIJ Service Agent, an agentized version of the internal RAG platform, and the WebSearch Agent, which retrieves the latest external information.

In terms of the actual control structure, we use a state-transition graph managed by LangGraph to orchestrate inter-agent collaboration. The main agent analyzes the question text, determines whether it can be solved with internal data or whether external information is required, and then calls the relevant subagent(s). The subagent results are returned to the main agent and ultimately presented to the user as an integrated response. This makes it possible to generate comprehensive answers combining internal information with up-to-date external information.

The subagents also support parallel execution, enabling simultaneous processing across multiple information domains, thereby producing results at speeds impossible to achieve manually. In addition, we provide enhanced transparency via visualizations of the LLM's inference process. The UI successively streams information on which agents were invoked and which information sources were used to generate the answer, allowing users to track the LLM's reasoning process in real time.

As a result, we were able to extend the answer coverage to areas over which the existing internal RAG platform could not provide full coverage, helping to greatly improve the user experience. The internal agent platform built on multi-agent principles overcame the information shortage issues that were difficult to address with the single-agent approach and represents an important step in significantly expanding the scope of AI utilization.

4.6 Combining Deep Research and RAG

Google's Deep Research, announced at the end of 2024, drew attention for its ability to autonomously search vast online information sources and analyze and integrate what it finds, enabling it to perform multi-step investigations much like a researcher or analyst would. In February 2025, OpenAI also released its own Deep Research offering, and in June of the same year, it made this available as the Responses API, providing practical components that can be incorporated into existing agent environments.

The defining feature of Deep Research is its ability to autonomously iterate through a multi-step cycle of planning, searching, scrutinizing, integrating, and generating. It executes multiple web searches, recalibrates its plans based on verified data, and ultimately delivers a highly reliable summary report. While some tasks can therefore take around 30 minutes to complete, it is a powerful tool for research in domains requiring a high level of comprehensiveness and evidence, such as technical literature reviews, competitor and market research, and regulatory comparisons.

Because our internal agent platform already used a multi-agent architecture based on LangChain and LangGraph, adding Deep Research as a subagent was relatively straightforward.

Specifically, we defined a Deep Research agent with @tool and implemented a function that calls the Responses API inside it. When a user query is received, Deep Research iteratively searches the web and, as needed, goes through several stages of reasoning and integration to return a structured response that includes a report, data tables, and a list of sources.

In addition, we introduced a mechanism that puts a ClarifyQuery (query clarification) agent at the front end of Deep Research. This prevents ambiguous queries from being passed directly to Deep Research, which would increase search retries and result in ballooning costs and latency. This agent reviews the user's input and, if necessary, poses follow-up questions to clarify the query before executing Deep Research. This facilitates comprehensive and efficient research that is aligned with user intent.

We also made the inference parameters provided by the Deep Research API (summary, effort, verbosity) adjustable by users, allowing them to select the depth of investigation and output volume according to their use case. For example, for a concise executive summary, users can select effort = low and verbosity = low, while for careful investigations such as specification comparisons, they might specify effort = high and verbosity = high. This allows for flexible control over output granularity for the same query with just a few clicks.

The UI also provides visualizations of processing progress and cost. In addition to streaming the AI's inference process, we also display the total number of tokens used and model consumption costs so that users can see the cost incurred in generating an answer. As developers, meanwhile, we can monitor token consumption and cached token usage, so we are able to use this information to optimize the tool.

A new consideration that emerged in the process of introducing Deep Research was the need for autonomous research capabilities that also include internal data. In its standard form, Deep Research is designed primarily for publicly available information on the Internet. It did not provide a mechanism for directly integrating with all internal knowledge bases. While we thought it would be technically feasible to connect internal knowledge sites to Deep Research via the APIs provided, this turned out to be unrealistic because of the high non-technical hurdles, namely the need for internal coordination, agreement, and approvals.

So our initial approach was to combine the IIJ Service Agent (internal RAG) and the WebSearch Agent, with the processing steps finely controlled via prompts. With the WebSearch Agent exploring the latest information online while the IIJ Service Agent (internal RAG) searches through internal information, this made it possible to ultimately return integrated results to the user. But with earlier LLMs (GPT-4.1 and earlier), ensuring sufficient comprehensiveness and consistency was difficult due to limits on reasoning steps and weak tool-invocation control.

Accordingly, we adopted as the main agent GPT-5, which was released during our development verification process and claimed to be optimized for agentic tasks. GPT-5 offers enhanced capabilities in terms of tool invocation, instruction following, long-context understanding, and reasoning retention across tool calls, greatly improving interoperability with tools in agentic workflows. As a result, by invoking each subagent multiple times to collect comprehensive

information, we were able to generate optimal answers based on roughly 30 to 80 citations, including internal data.

This represented a significant step forward as we were now able to integrate both internal data and external information sources to provide highly comprehensive, reliable answers even to queries for which we would previously have determined information to be insufficient. And as developers, the ability to resolve the issue simply by dropping in the latest LLM without any major changes to existing architecture was key and really brought home the extensibility of the agentic platform and the potential for its ongoing evolution.

4.7 Developing a Proposal Document Generation Tool

Once operation of the internal RAG platform had stabilized, we started developing a new web application called Panorama, a proposal document generation tool. When creating PowerPoint-format proposals in accord with given requirements, sales staff and engineers were spending a lot of time on the task of selecting and editing slides from internal templates. Panorama was developed to solve that problem. It provides a simple interface for doing everything from requirements input through to automatic generation of a proposal draft (Figure 3).

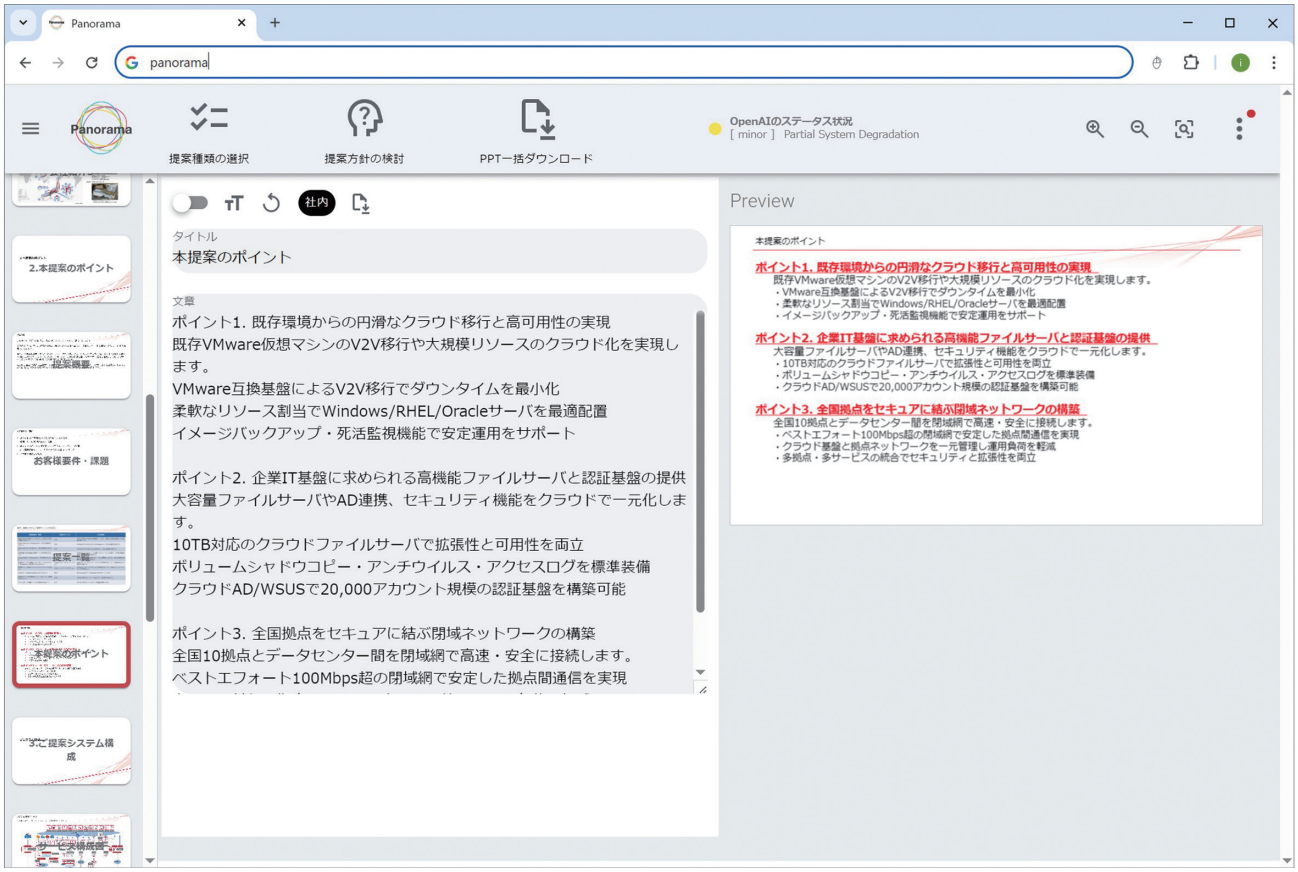
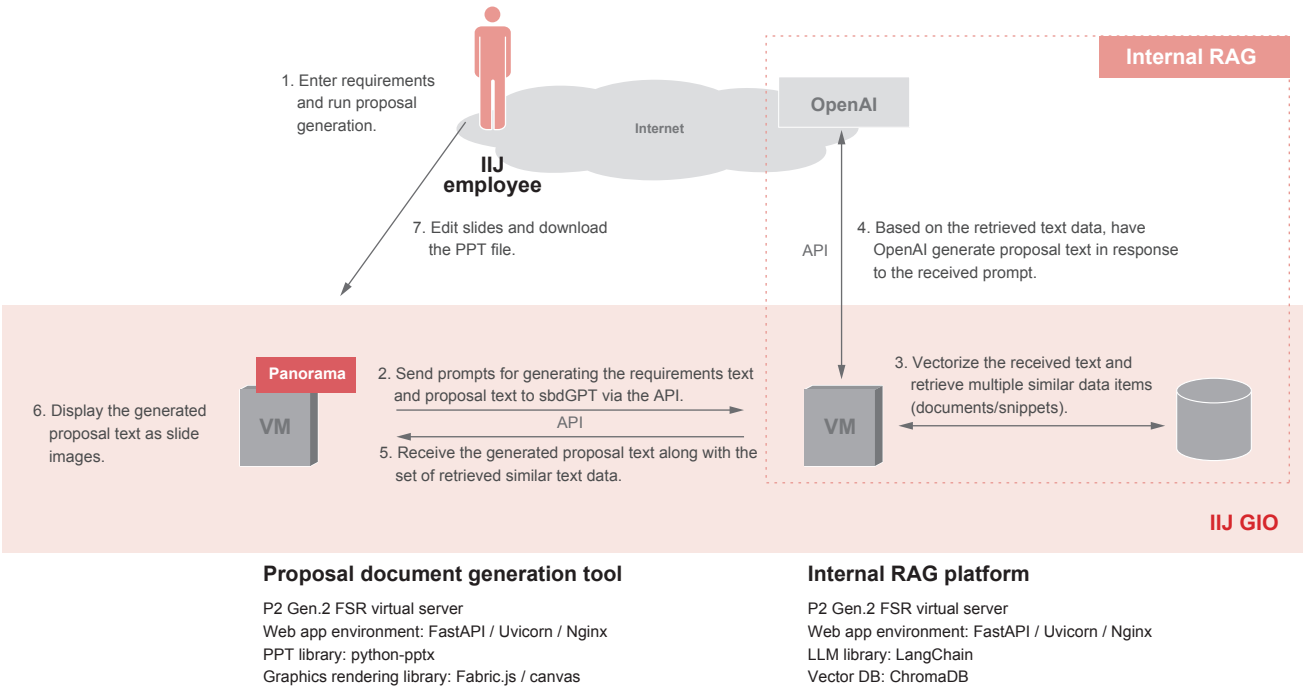


Figure 3: Proposal Document Tool's User Interface

The proposal document generation tool works as follows. After the user enters requirements and desired services via the browser and submits the request, the Web server then calls the internal RAG platform API. Using the input data and the system prompt, the tool automatically generates the most appropriate service proposal text (Figure 4). The generated output is converted into JSON describing each slide's structure and layout, and this is reflected in the browser-based editor and rendered in real time on an HTML5 Canvas for preview purposes. Fabric.js is used for rendering and editing, and users are able to review and revise the architecture diagram layout and proposal text in the browser before downloading the final output as a PowerPoint file.

In technical terms, the system consists of a frontend built with Vue.js + fabric.js and a backend built with FastAPI + python-pptx. On the frontend, edits to slides and architecture diagrams are reflected in real time, and all edit data is centrally managed in JSON format. The data are sent via FastAPI and used to generate PPTX files on the backend. On the backend, Python is used to manipulate and update XML configuration files corresponding to template slides based on the JSON data, and based on the resulting modifications, the final proposal document file is generated using the python-pptx library.

Through these mechanisms, the AI-generated proposal text and automatically plotted architecture layout diagrams are reflected in the slides, enabling users to create consistent-quality proposals in a short amount of time with minimal editing work. Integration with the internal RAG platform also means that proposal text automatically reflects internal



Proposal document generation tool

P2 Gen.2 FSR virtual server
Web app environment: FastAPI / Uvicorn / Nginx
PPT library: python-pptx
Graphics rendering library: Fabric.js / canvas

Internal RAG platform

P2 Gen.2 FSR virtual server
Web app environment: FastAPI / Uvicorn / Nginx
LLM library: LangChain
Vector DB: ChromaDB

Figure 4: Overview of Proposal Document Tool System Architecture

knowledge such as IIJ service specifications, the rationale for recommendations, and key proposal points. Source information is displayed clearly on the editing screen, and users can instantly confirm data sources at the click of a mouse, making it easy to fact check proposal content.

Going forward, we plan to incrementally add template slides and add more features such as PDF-format exports and enhanced diagram editing features.

4.8 Looking Ahead

The evolution of the latest language models and agent technologies is making the autonomous execution of complex tasks a reality, and agentic systems designed to support business automation and efficiency improvements are growing rapidly in importance.

For now, we plan to continue expanding the capabilities of the AI agent platform, progressively developing and adding tools designed for specific tasks. We will also examine mechanisms for creating a mutually reinforcing cycle between incentives for the creators of referenced data sources and increases in citation frequency.

Over the medium to long term, we aim to transition to an architecture that uses local LLMs and constitutes a complete system within IIJ's own infrastructure, with our goal being to develop an environment that we can offer to customers as a package integrated with IIJ's wide range of services.

Kazunori Ebine

AI Platform Promotion Office, System Development Division, Network Services Business Unit, IIJ
Mr. Ebine joined IIJ in 2005. He worked on building systems for the public and private sectors, handling a wide range of responsibilities from requirements definition through design, implementation, and operation. He has contributed to revenue growth by proposing and rolling out cloud services and building technology delivery structures. Having previously led AI-driven efficiency improvements and cross-departmental initiatives as a Deputy General Manager and a Manager, he is currently spearheading the development and planning of AI platforms.

February 2026 Vol.68

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.

IIJ-MKTG020-0066

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/