The Internet Protocol Journal

August 2022 Volume 25, Number 2

A Quarterly Technical Publication for Internet and Intranet Professionals

In This Issue

You can download IPJ back issues and find subscription information at:
www.protocoljournal.org

ISSN 1944-1134

From The Editor

According to Wikipedia, the *Border Gateway Protocol* (BGP) "...is a standardized exterior gateway protocol designed to exchange routing and reachability information among *Autonomous Systems* (ASs) on the Internet. BGP is classified as a path-vector routing protocol, and it makes routing decisions based on paths, network policies, or rule-sets configured by a network administrator." We've covered numerous aspects of BGP in this journal, most recently in our two-part article by Geoff Huston entitled "A Survey on Securing Inter-Domain Routing." In this issue, a team of engineers from Juniper Networks describes a method for running BGP processing in parallel using a concept known as *sharding*.

In our second article, Geoff Huston takes a closer look at the transport and network functions in today's ever-changing Internet. Many network elements such as firewalls and *Network Address Translators* (NATs) use the transport protocol *header* to make decisions on how to handle traffic, but concerns about pervasive monitoring and information leakage have led to various forms of encryption-based solutions and an ongoing debate within the Internet technical community and beyond.

Using the Internet for teleconferencing or telephony is not a particularly new idea. I fondly remember taking part in experiments between the *Norwegian Defence Research Establishment* (NDRE), MIT's Lincoln Laboratories, *University of Southern California's Information Sciences Institute* (USC-ISI), and *University College London* (UCL) as early as 1977 when I was doing my military service at NDRE. You can find out more about these early developments by searching for the article "Linear Predictive Coding and the Internet Protocol." *Voice over IP* (VoIP) as we know it today became a reality in 2002 with the publication of RFC 3261, which describes the *Session Initiation Protocol* (SIP). In our final article, Jonathan Rosenberg gives a retrospective on 20 years of SIP.

Publication of *The Internet Protocol Journal* is made possible by the generous support of numerous individuals and organizations. Please consider making a donation or getting your company to sign up for a sponsorship. As always, we welcome your feedback and suggestions on anything you read in this journal. Letters to the Editor may be edited for clarity and length and can be sent to **ipj@protocoljournal.org**

—Ole J. Jacobsen, Editor and Publisher
ole@protocoljournal.org

Parallel BGP Protocol Processing

by Sanjay Khanna, Jaihari Loganathan, and Ashutosh Grewal, Juniper Networks

Managing large inter- and intra-*Border Gateway Protocol* (BGP) domains places a large computational load on the CPU of a router, adversely affecting its performance and increasing the BGP convergence time. To address these problems, we have architected a solution that splits a BGP *Routing Information Base* (RIB) across concurrently running BGP threads. These parallel running threads run the same code on multiple CPU cores concurrently. Each of these threads maintains a RIB *shard*, a subset of the RIB. This parallel BGP processing improves the read-side performance of processing incoming UPDATE messages. A set of parallel running I/O threads generate outbound UPDATE messages and improve the write-side performance of BGP. This entire design uses a lockless mechanism to allow parallel processing on each CPU core independently. A testbed representing a Tier 1 service provider *Route Reflector* network was used to verify and quantify the performance of the implementation. BGP in this topology receives several copies of the global Internet routing table (~800,000 routes). Our results show that performance improves as parallelism and scale are increased. The speedup we can attain gets better as more CPU cores are available for RIB sharding. These gains are bounded by the extent to which the BGP update processing can run in parallel.

Terms and Definitions

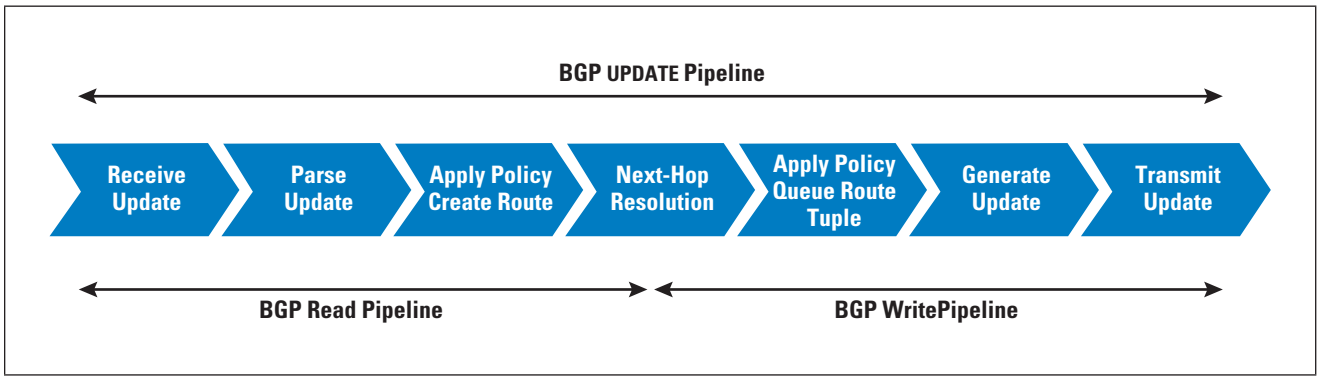
The BGP RIB conceptually consists of four parts:

- *Adj-RIBs-In:* Stores unprocessed routing information that has been learned from BGP updates received from peers. The routes contained in Adj-RIBs-In are considered feasible routes.
- *Loc-RIB:* Contains the routes that the BGP speaker has selected by applying the decision process (route selection, import policy) to the routes contained in Adj-RIBs-In. These routes populate the routing table (RIB) along with routes that other routing protocols discover.
- *Adj-RIBs-Out:* Contains the routes that the BGP speaker advertises to its peers in BGP UPDATE messages. Export routing policies determine what routes are placed in Adj-RIBs-Out.
- *Outbound Route Tuple is a route in Adj-RIBs-Out*. Every tuple consists of a prefix, associated BGP metrics, and information about peers in a peer group that will be sent to this tuple in an UPDATE message.

BGP Update Processing Pipeline

Figure 1 illustrates the BGP[1] UPDATE message processing pipeline. This pipeline can be further subdivided into the *Read-Side Pipeline* to identify the inbound processing of an UPDATE message, and the *Write-Side Pipeline*, which concerns generation of outbound UPDATE messages to be sent to peers.

Figure 1: BGP Pipeline



The read-side pipeline consists of the following stages:

- *Receive Update:* Routes are advertised between BGP speakers in an UPDATE message. Multiple routes that have the same PATH attributes are advertised in a single UPDATE message. This message is received over an established *Transmission Control Protocol* (TCP) socket.
- *Parse Update:* A BGP UPDATE message is parsed for prefixes, and PATH attributes are canonicalized into a local state after validating the data in *Protocol Data Units* (PDUs).
- *Apply Import Policy and Create Route:* BGP stores routing information learned from the inbound UPDATE messages in a RIB called *Adj-RIB-In*. BGP applies the import policy on incoming BGP routes and—if permitted by policy—performs a best-route selection. The best routes are used to populate the local RIB, called *Loc-RIB*. These routes are then used to program a *Forwarding Information Base* (FIB) and generate outbound updates to BGP peers.
- *Next-hop Resolution:* BGP uses a local routing table to find the reachability information for a BGP next-hop, which may be several hops away. For example, *Interior Gateway Protocol* (IGP) metric, intermediate address, and outgoing interface are parts of resolving reachability for the BGP next-hop. BGP may have several routes to the same IP destination that have different degrees of preference. Although all accepted routes are installed in Loc-RIB, BGP may choose one route or multiple routes (BGP multipath case) as active routes.

The write-side pipeline consists of the following stages:

- *Apply Export Policy and Queue Outbound Route Tuple:* BGP applies export policy to routes in Loc-RIB and generates outbound RIBs called *ADJ-RIBs-Out*. Adj-RIB-Out stores information that BGP uses to generate an outbound route tuple. Generally, several peers with the same policies are grouped into a *peer group*, and a single outbound route tuple is generated for each peer group.

- *Generate Update:* Route information in the outbound route tuple is converted into an UPDATE message. Destination prefixes that share the same PATH attributes are packed in a single message. This prefix packing reduces the number of UPDATE messages sent over TCP. Sending fewer messages improves local performance and does not impose extra work on the peer BGP routers.
- *Transmit Update:* UPDATE messages that were generated in a prior stage are sent over a TCP socket to a remote BGP peer. The same BGP UPDATE message is sent to multiple BGP peering sessions that share a common export policy, thereby amortizing the cost of UPDATE message generation.

BGP *convergence time* is the time taken by the router to process the incoming BGP UPDATE messages, passing them through the read pipeline in Figure 1, and distributing the results by generating UPDATE messages to its peers. As networks scale up in the number of peers, the number of parallel inbound feeds, and the size of the network in terms of the number of prefixes, this convergence time can become very high. The result can be slower convergence of the entire network when device and link failures occur. The slower convergence generally leads to traffic loss and a traffic black-hole in the network. Most operators of large networks want a faster convergence to reduce these downtimes. To help improve the convergence time of a router, and to exploit multi-core CPUs available on routing engines, we decided to run the previously mentioned pipeline functions in parallel. We encountered several hurdles to running BGP update processing in parallel:

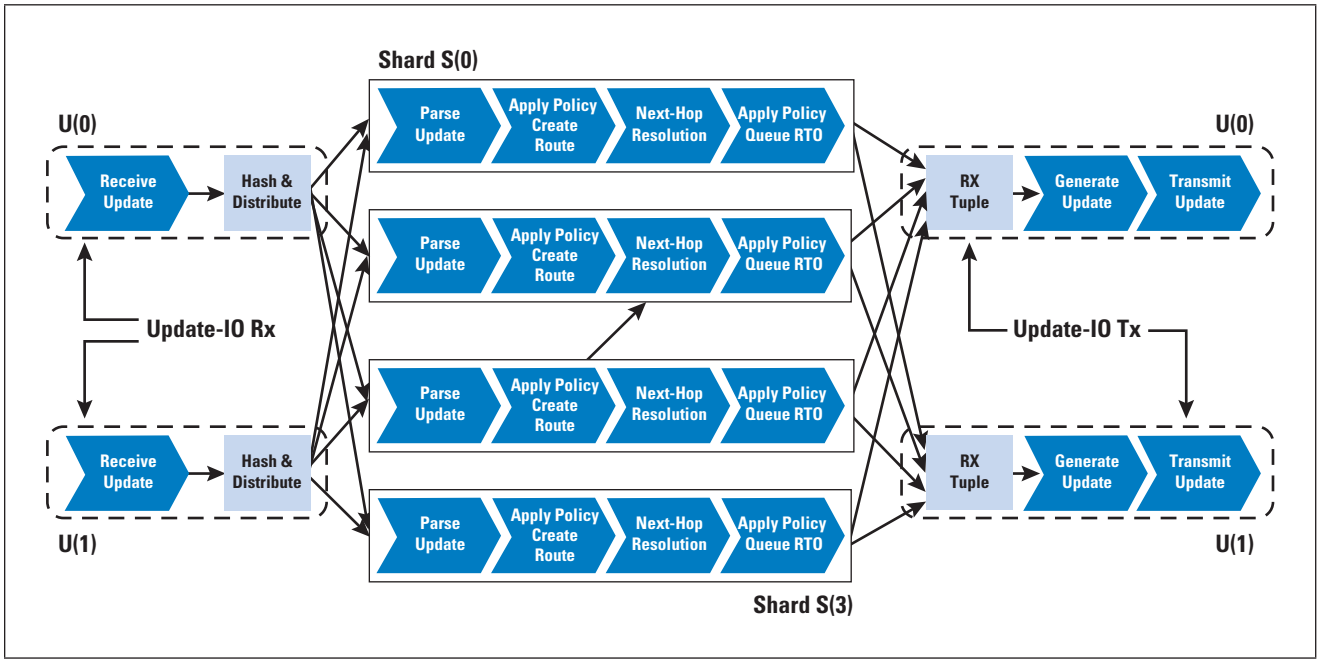
- RIBs are generally a shared collection of routes, and concurrent updates to RIBs need locks.
- Locking in general reduces concurrency and increases complexity.
- BGP next-hop resolution is a shared function and requires synchronization between parallel running threads.
- Prefix packing in outgoing UPDATE messages deteriorates because each concurrent pipeline produces more frequent and smaller UPDATE messages with fewer prefixes.
- Other protocols in an IP router need active BGP routes to implement other useful functions like *Layer 3 Virtual Private Network* (L3VPN) and programming the FIB.
- Interface state (like links and IP addresses) is a shared collection too, and concurrent threads meant more locking across concurrently running threads.

The BGP software architecture described in the following sections addresses these concurrency limitations and improves the BGP convergence time at very high scales. This solution performs better when input scale increases and the required number of concurrent pipelines increases.

Parallel BGP Protocol Processing

Figure 2 illustrates the high-level architecture of parallel BGP processing. The pipeline is broken into three logical parts running concurrently on two kinds of threads of execution: "Shard(S)" and "Update-IO(U)" threads. Shard threads process a subset of UPDATE message prefixes, execute most of the BGP read pipeline, and generate a tuple for an Update-IO thread. An Update-IO thread processes a tuple for a set of BGP peers in a peer group. Update-IO threads also receive the UPDATE message from peers (over TCP sockets) and distribute the messages to shard threads. These threads process tuples from shard threads and generate UPDATE messages to transmit to peers.

Figure 2: Parallel Processing of BGP UPDATE Messages



- *Update-IO Rx Processing:* Running in an update-IO thread, this stage receives update messages from peers, sanity checks the messages, and computes a hash on every prefix in the message to determine which shards will receive the message. In the best possible scenario a single shard gets the entire message, and in the worst-case scenario the same message is shared with every shard thread.
- *Shard Processing:* A shard thread receives a copy of the message and processes prefixes that it owns (ignoring the ones that it doesn't). The ownership of a prefix is decided by computing the hash on the prefix. Thereafter each shard will follow the entire read pipeline on its prefixes and generate tuples for the Update-IO thread matching the peer group.
- *Update-IO Tx Processing:* An Update-IO thread receives tuples and processes them into BGP UPDATE messages. These messages are sent towards the peer via TCP sockets. This stage is responsible for packing tuples from several shard threads into UPDATE messages to improve the packing of prefixes.

This architecture exploits the *Single Program Multiple Data* (SPMD) model of parallelism to do concurrent read and write pipeline processing. Concurrently executing threads do not share any state with each other, thereby eliminating any need for locks. Shards and Update-IO threads maintain a parallel ecosystem of collections of objects (like BGP peers, peer groups, RIBs, interfaces, configuration, etc.). Wherever needed, message passing is used to achieve eventual consistency of the routing state in the system. At any point of time, concurrently executing functions running at different rates may be in different states. However, all these functions executing on different cores will eventually reach the same final state as if there were a single executing pipeline. Reaching eventual consistency quickly is an important outcome of this architecture.

Sharding Several Kinds of BGP RIBs

This architecture requires all routes (BGP and otherwise) with the same IP address to always be assigned to the same shard so that the best active route calculation for all routes matching an IP address is handled in a single shard. This requirement guarantees the correctness of the active route selection algorithm. *Multiprotocol BGP* (MP-BGP) extensions[2] allow BGP to carry routing information for multiple network layers and address families. BGP routes for each of these address families are saved in several RIBs, and each RIB has its own network prefix in a route. For example, the IPv4 Unicast RIB has the IPv4 address and prefix mask length as the route destination. The L3VPN RIB has the IP address, route distinguisher, and mask length as the route destination. For shard assignment, the hash is computed only on the address and prefix length part of the route destination, and the rest is ignored.

Next-Hop Resolver

The main job of the resolver is to translate protocol next-hops into forwarding next-hops using *helper routes*. A protocol next-hop is an IP address of a remote BGP peer, most commonly an *Interior BGP* (IBGP) peer. A protocol next-hop, by itself, is insufficient to make a forwarding decision. To forward a packet, a router needs to know the directly connected next-hop. This information is derived from helper routes that provide reachability information for the protocol next-hop. Since the resolver is a central function and the concurrent shard threads also need the services of this resolution, a mechanism is needed to distribute resolver information to shard threads. One way to achieve this distribution is to run the resolver as a service and all shard threads to register next-hop IP addresses for resolution. BGP in each shard gets reachability notifications for registered IP addresses. These notifications populate the local BGP neighbor reachability information of the shard and trigger routing updates local to a shard. These updates can activate a BGP route when a next-hop is reachable, change the BGP route if reachability changes, and inactivate the BGP route if a next-hop is unreachable.

VPN and Sharding

You can apply sharding to *Virtual Route Forwarding* (VRF) and *Virtual Private Network* (VPN)[3] routing tables also. Each shard thread hosts a slice of the VPN and VRF RIB table as determined by the hash. The *Route Distinguisher* (RD) of VPN routes is excluded from hash calculations to allow routes with the same prefixes but different RDs to be correctly processed in the associated shard. VPN label allocation is a central service because a single pool of *Multiprotocol Label Switching* (MPLS) labels is generally available to send out. A shard thread that wants an MPLS label for a VPN route requests this centralized service for labels. Target routes—needed for VPN processing—are stored in a separate RIB. Since this RIB is smaller, it is duplicated in all shard threads.

Large Peer Groups and Update-IO Parallelism

A peer group is usually assigned to an Update-IO thread that manages packing and generation of updates for that group. Multiple groups get assigned to different update threads for parallelism. In certain use cases, one or more large peer groups can include a very large number of BGP peers, and as a result we would not be able to distribute the load of such peer groups over several Update-IO threads effectively. To handle such a special case, we split such configured peer groups into several logically split peer groups. Each split group is allocated a subset of peers from a large peer group and assigned to an Update-IO thread.

BGP Graceful Restart Handling with RIB Sharding

BGP *Graceful Restart* (GR)[4] processing requires sending of an *End-of-Rib* (EOR) notification after initial download of routes to a peer that is coming up after a failure. Shard threads contribute to the initial download of routes to a peer independently and in parallel. But the EOR message is sent in a coordinated fashion from the main thread after all shard threads complete the initial download of their slices of RIB to the peer. Likewise, consumption of an inbound EOR requires coordination from shard threads. The inbound EOR message is sent to all shard threads.

BGP Optimal Route Reflection and Sharding

You can configure BGP *Optimal Route Reflection* (ORR)[5] with *Intermediate System-to-Intermediate System* (IS–IS) and *Open Shortest Path First* (OSPF) on a Route Reflector to advertise the best path to the BGP ORR client groups. Configuration is done by using the IGP metric after calculating the *Shortest Path First* (SPF) from a client's perspective. For BGP ORR to work in a Route Reflector, BGP requires assistance from an IGP implementation to calculate an IGP metric for a prefix from a client's perspective. In a sharded BGP architecture, ORR must be built as a service like a resolver. Shard threads register ORR reachability to this service, and in turn receive notifications about reachability of the registered prefixes.

Configuration, CLI Show Commands, Telemetry, SNMP, and Sharding

The shard architecture requires that each shard thread processes configuration independently of other shard threads. This way each shard has its own configuration state to work with. To present a consistent view to the user, some of the *show* commands in a router must collate information about RIBs from all shard threads and present a system view of the RIBs to the user. The same is true about telemetry streaming and the *Simple Network Management Protocol* (SNMP) get/walk of tables in a shard. For debugging purposes, we also implemented a view into the RIBs of each shard thread.

Routing and BGP RIB Sharding

Figure 3a illustrates an approximate high-level view of how routing protocols were implemented in the JUNOS *Routing Protocols Daemon* (RPD) before BGP RIB sharding was implemented. It shows a single thread that runs all routing protocols (including BGP), and INFRA, which includes interfaces, routing tables, next-hop tables, route resolution, and FIB programming. When BGP RIB sharding and Update-IO are configured, then additional threads are spawned, as shown in Figure 3b.

Figure 3a: Single Threaded Routing Protocol Daemon

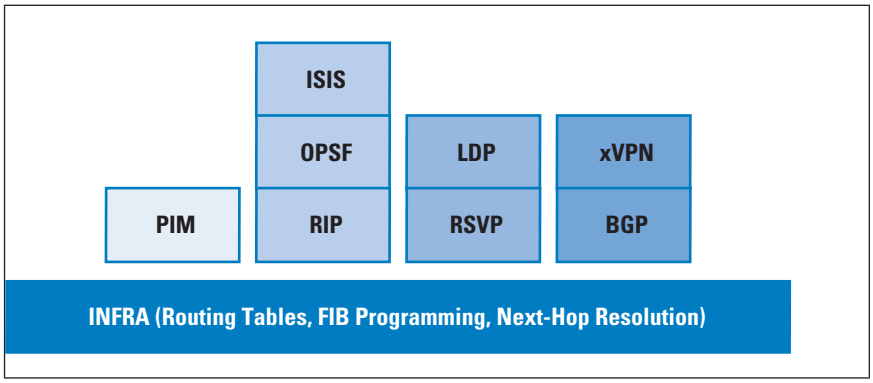
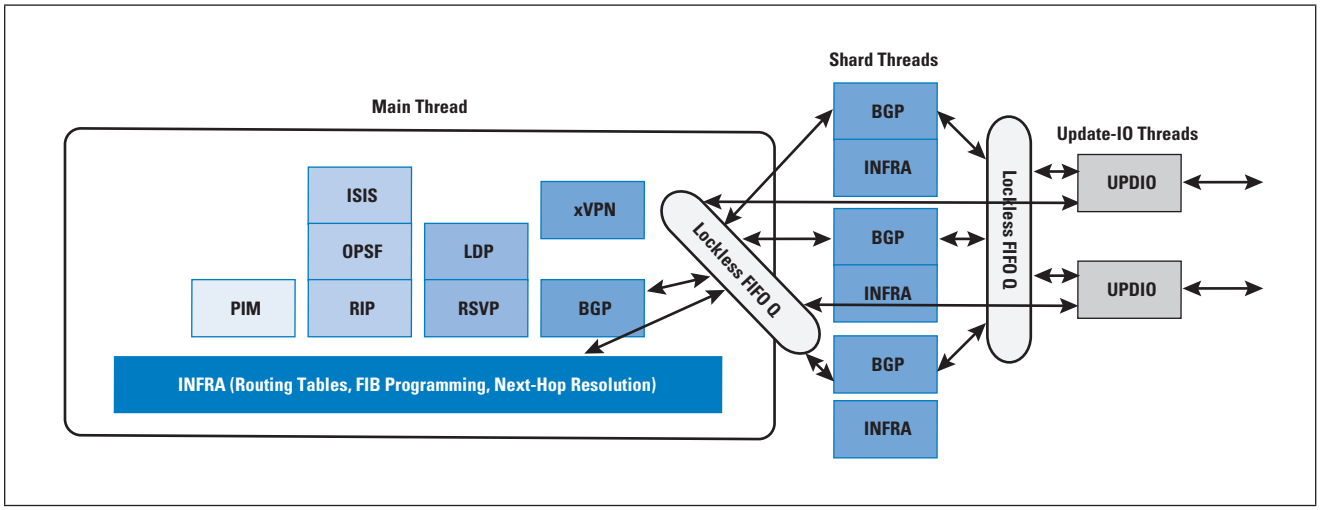


Figure 3b: Shard Routing Protocol Daemon



Main thread shards and Update-IO threads do not share any mutable state, and as a result no locks or critical sections are executing between these threads. As explained earlier, shard threads do the read-side processing of incoming messages in parallel and Update-IO threads do write-side processing of outgoing messages in parallel. Shard and Update-IO threads like the main thread process the routing configuration independently in a lockless manner. Coordination of peer state, interface state, and RIB routes and counters between main and other threads is done via *Interprocess Communication* (IPC) messages. These messages are sent over an IPC channel consisting of a pair of lockless *First-In, First-Out* (FIFO) queues. To allow for state compression, IPC channel queue depths are kept small. Socket read/write readiness is used for IPC channel back pressure between threads. Overall, all message passing between threads is very fast and efficient.

The following message types are sent from the main thread to the shard threads:

- Interface state, such as link information, address families configured, IP address, and state of the links is used by routing tables code and the BGP protocol. As a result, each shard has its own copy of interface state information.
- Route messages for non-BGP routes are distributed to a shard that is found via hash computation on the IP prefix of the route. Also, the entire route target RIB is sent as route messages to all shard threads.
- Configuration indications are sent as messages to signal availability of the updated configuration database.
- When *show* commands, SNMP *get* requests, and telemetry requests are sent, responses from shards are sent to the main thread to service the *Command-Line Interface* (CLI), SNMP requests, and telemetry streaming.
- BGP peer state transition messages are sent from the main thread to the shard threads. A handler in shard threads takes appropriate actions on BGP peer objects local to the shard.
- Next-hop resolution messages are sent from the resolver service in the main thread to shard threads. Shard threads register with this service for a BGP next-hop.
- VPN label messages are sent to shard threads whenever the main thread receives a request from a shard.

BGP in the main thread also shares state with Update-IO threads via IPC messages. These messages follow:

- The BGP peer *Finite State Machine* (FSM) in the main thread sends peer state messages to Update-IO threads. Such messages result in state changes for BGP peer objects in the Update-IO threads.
- *Route tuple messages:* Outbound BGP route messages are sent from the main thread to one of the Update-IO threads. Update-IO threads consume these messages to generate BGP UPDATE messages for a set of peers.

Shard threads send the following messages to the main and Update-IO threads:

- Route messages are sent from the shard to send active BGP-only routes to the main thread. The main thread adds these received routes to its RIB.
- Shards send *show*, SNMP, and telemetry responses to the main thread to assist in supporting these administrative functions.
- Shards send resolver registration requests for BGP next-hops, and they get resolver responses from the main thread.
- VPN label requests are sent to the main thread to get a pool of labels to support VPN functions in the shard threads.
- Outbound route tuples are sent to Update-IO threads per peer group.

When a shard thread receives UPDATE messages, it does the PDU processing on the prefixes that it owns and ignores those it does not. This processing uses the same hashing scheme that Update-IO threads use for distributing inbound prefixes. As a result, BGP routes are added to the RIBs of a shard thread. If the active route for a prefix in a shard is a BGP route, then it must be added to the FIB. Queueing points are used on the shard send side (before the IPC) to dampen the route churns due to link and peer flaps. The active route is added to the queue and then distributed (via IPC) to the main thread.

The main thread can program the FIB. Any subsequent state changes associated with BGP prefixes in shard threads may result in changes to the current active route in the shard. When this active route changes or is deleted in a shard, a new message is queued to be sent to the main thread.

The main thread centralizes answering BGP next-hop resolution, and it also programs the FIB. Any changes in the next-hop reachability are announced to shard threads. Shard threads react to such announcements and cause the necessary state changes in the Update-IO threads and the active route redistribution to the main threads. Update-IO threads generate outbound UPDATE messages and announce the prefix changes to their neighbors. In the end all threads, the FIB, and the network will have the consistent view of any prefix.

When a shard thread has run the BGP export policy and decided which BGP peers across all peer groups will receive a route, a new set of IBGP route announcement tuple messages (tuples) are queued for Update-IO threads. Shard threads multiplex several tuples back to an Update-IO thread. These tuples carry state about the BGP PATH attributes and prefixes. These PATH attributes are assumed immutable and are shared in a reference-counted manner between threads.

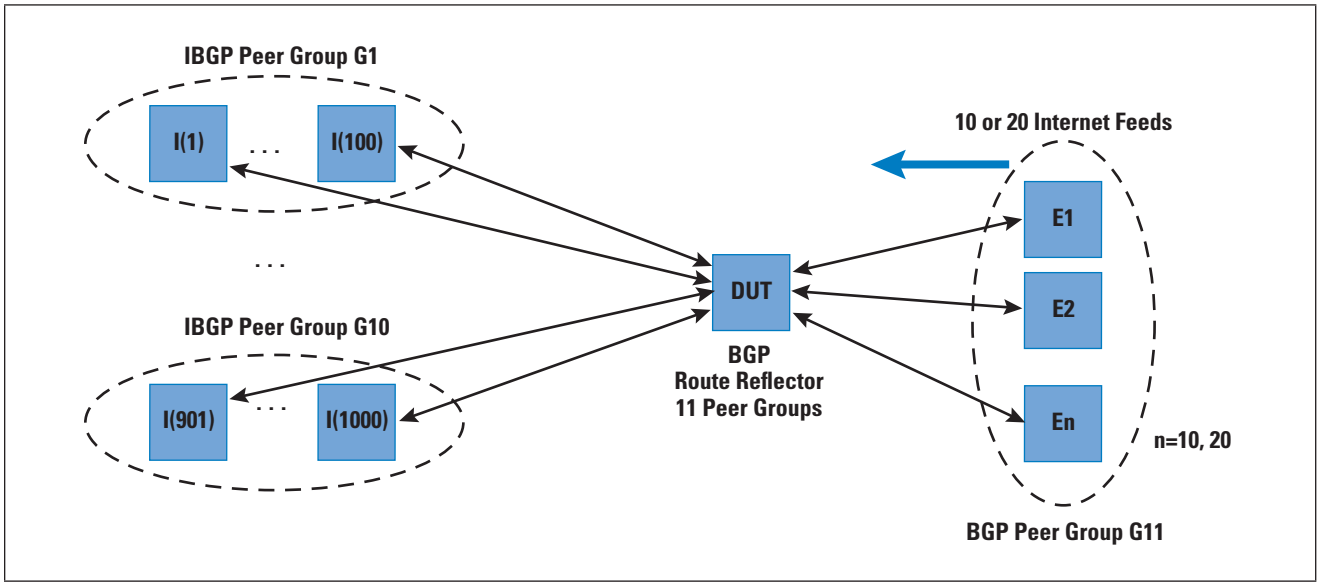
The Update-IO thread puts many tuples with the same BGP PATH attributes in a single BGP UPDATE message for transmission. Packing helps ensure that the number of BGP UPDATE messages is limited to prevent flooding of the local TCP/IP stack with too many send calls. Also, packing ensures that downstream BGP routers are not flooded with excessive messages. It ensures that gains from concurrent read processing in the shard threads to CPU overhead are not lost by handling more I/O messages.

Various queueing points are present in the main, shard, and Update-IO threads in our routing implementation. As these routes are learned and selected to distribute, the queueing points help us compress fast occurring state changes and reduce the churn from spreading from a source to the consumers. This churn reduction is very important when operating at scale and with several producers and consumers. Shard threads compress route changes for prefixes being redistributed to the main thread. For example, a route addition followed by a route deletion nullifies both. The main thread has a similar queueing logic to compress state churn when downloading routes to the FIB. Shard threads have a queue towards Update-IO threads to dampen the tuple churn in the BGP. Similarly, the Update-IO threads also use tuple queues to pack and suppress state before final state is disseminated to the peers.

Performance Measurements

To measure the gains of the approach described previously, we used a testbed where the *Device Under Test* (DUT) is a scaled BGP Route Reflector (Figure 4).

Figure 4: Route Reflector Topology



The RR receives several Internet feeds (800,000+ routes per feed) from an *External BGP* (EBGP) peer group of 10 BGP peers and reflects the best path for each destination towards 1,000 BGP client peers divided among 10 peer groups. Thus, a total of 11 BGP peer groups are in this testbed.

This setup represents a high scale of incoming UPDATE messages, and even higher scale of outbound UPDATE message generation. We used Internet feed instead of canned routes from a protocol tester like IXIA to mimic the real-world scenario where routes have variable PATH attributes. The DUT is a Linux Ubuntu with 18 servers with 8 cores (Xeon CPU E5-2640 v3 @ 2.6 GHz) and hyperthreading is turned off. The RR server on the DUT runs in a Linux Docker Container. The RR clients were also running our BGP implementation, where these clients drop all incoming PDUs after checking them for validity. This modification was done to ensure these 1,000 clients are never a bottleneck when receiving UPDATE messages from the DUT.

Table 1. Read-Side Performance Measurements

Number of Shard Threads	Convergence Time (Seconds)	Scale Up (S)
0	73	1.00
1	76	0.96
2	42	1.74
3	31	2.35
4	26	2.81
5	20	3.65
6	18	4.06

Table 2. BGP Write-Side Performance (4 Shards)

Number of Update-IO Threads	Convergence Time (Seconds)	Scale Up (S)
0	352	1.00
1	259	1.36
2	141	2.50
5	67	5.25
10	54	6.52

Table 3. BGP Full Pipeline Performance (4 Shards)

Number of Update-IO Threads	Convergence Time (Seconds)	Scale Up (S)
0	347	1.00
1	282	1.23
2	161	2.16
5	93	3.73
10	77	4.51
11	78	4.45

The testbed was run in three modes:

- Add a discard export policy to the DUT to ensure that no tuples to Update-IO threads are generated and we can get the measurements of the read side of the BGP pipeline. Table 1 shows the performance numbers for this mode of measurement. We noticed a maximum scale-up of 4x on six shards, and beyond six shards our gains were not beyond 4x.
- After the DUT has all the inbound Internet feeds and the RIBs have settled down, we delete the discard export policy, and this deletion triggers generation of tuples towards Update-IO threads and update messages start streaming toward the 1,000 peers. This step emulates the write side of the BGP pipeline and helps us measure that performance. Table 2 presents the measurements for this mode with 4 shard threads. The numbers of Update-IO threads were chosen to ensure 10 peer groups map to an even number of threads. We noticed gains linearly increasing as the number of threads increased.
- To measure the full pipeline performance, we repeated the previous 2 modes of testing without any discard export policy. We ran tests with 4 shard threads and up to 11 Update-IO threads. As expected, we noticed several fold improvements in the performance of BGP convergence time as we increased the number of I/O threads, as shown in Table 3.

Conclusions

Sharding in databases, web servers, and parallel computing concepts of SPMD have been used many times in the industry. This article is the first one where both concepts are used with BGP. We have designed and implemented a solution of splitting BGP into read and write pipelines and data (the RIBs). Our technique of sharing nothing among threads implementing BGP read and BGP write pipelines yields significant gains in scale and performance without impacting the convergence properties of the BGP protocol. This design also maintains BGP prefix packing and reduces the impact on local and remote routers. Lastly, we have implemented a design that keeps providing performance gains as parallelism increases, but the gains are limited by Amdahl's Law[6].

References and Further Reading

[1] Yakov Rekhter, Susan Hares, and Tony Li, "A Border Gateway Protocol 4 (BGP-4)," RFC 4271, January 2006.

[2] Tony Bates, Ravi Chandra, David Katz, and Yakov Rekhter, "Multiprotocol Extensions for BGP-4," RFC 4760, January 2007.

[3] Eric Rosen and Yakov Rekhter, "BGP/MPLS IP Virtual Private Networks (VPNs)," RFC 4364, February 2006.

[4] Srihari R. Sangli, Enke Chen, Rex Fernando, John Scudder, and Yakov Rekhter, "Graceful Restart Mechanism for BGP," RFC 4724, January 2007.

[5] Robert Raszuk, Bruno Decraene, Christian Cassar, Erik Aman, and Kevin Wang, "BGP Optimal Route Reflection (BGP ORR)," RFC 9107, August 2021.

[6] Amdahl, Gene M., "Validity of the Single Processor Approach to Achieving Large-Scale Computing Capabilities," *AFIPS Conference Proceedings* (30): 483–485, 1967.

SANJAY KHANNA holds a B.E. from Delhi University, a M.S., and a Ph.D. from Old Dominion University. Since 1993 he has worked in several IP networking-related jobs at IBM, Ericsson, Extreme Networks, and Juniper Networks. For the last several years he has been working on modernizing Juniper's routing stack. He is a member of ACM. He can be reached at: **skhanna@juniper.net**

JAIHARI LOGANATHAN has been working in the networking industry since the early 1990s. He has a bachelor's degree in computer science. He has worked on a variety of networking equipment and solutions from access modems, switches, security gateways, data center fabric, to core routers. His career spans several networking and cloud startups. He is a subject matter expert in many things networking. He has also helped start companies in search and networking space. He is currently working as a Distinguished Engineer at Juniper networks. He can be reached at: **jlogan@juniper.net**.

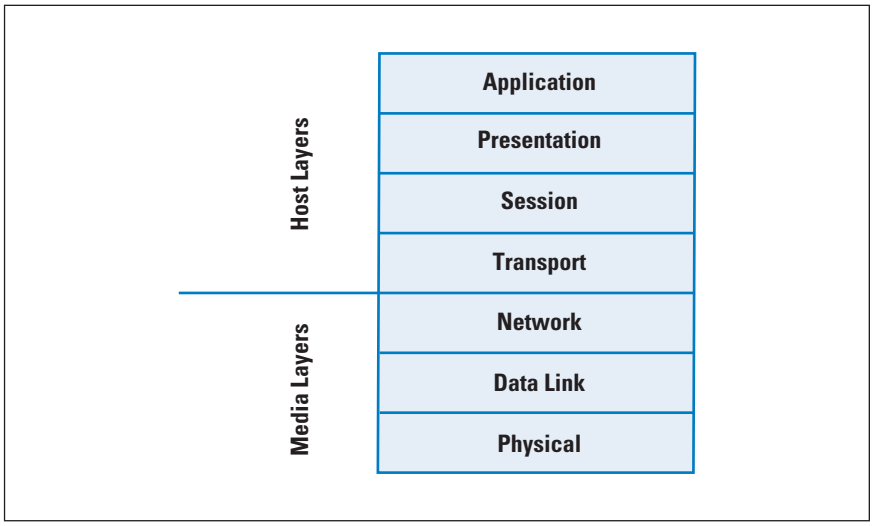
ASHUTOSH GREWAL holds a B.Tech. from the National Institute of Technology Durgapur and an M.S. from North Carolina State University. Since 2012, he has worked in the JUNOS routing protocols group at Juniper Networks as a Software Engineer on a variety of BGP, routing, and networking-related projects. He can be reached at: **agrewal@juniper.net**

Transport Versus Network

by Geoff Huston, APNIC

One of the basic tools in network design is the so-called "stacked" protocol model. This model was developed in the late 1970s as part of a broader effort to develop general standards and methods of networking. In 1983, the efforts of the *Consultative Committee for International Telephony and Telegraphy* (CCITT) and *International Organization for Standardization* (ISO) merged to form *The Basic Reference Model for Open Systems Interconnection*, usually referred to as the *Open Systems Interconnection Reference Model*, or the "OSI Model"[0]. This model included a seven-layer abstract model of networking that defined standard behaviours of both the overall network functions and the various components of the network (Figure 1).

Figure 1: OSI Reference Model



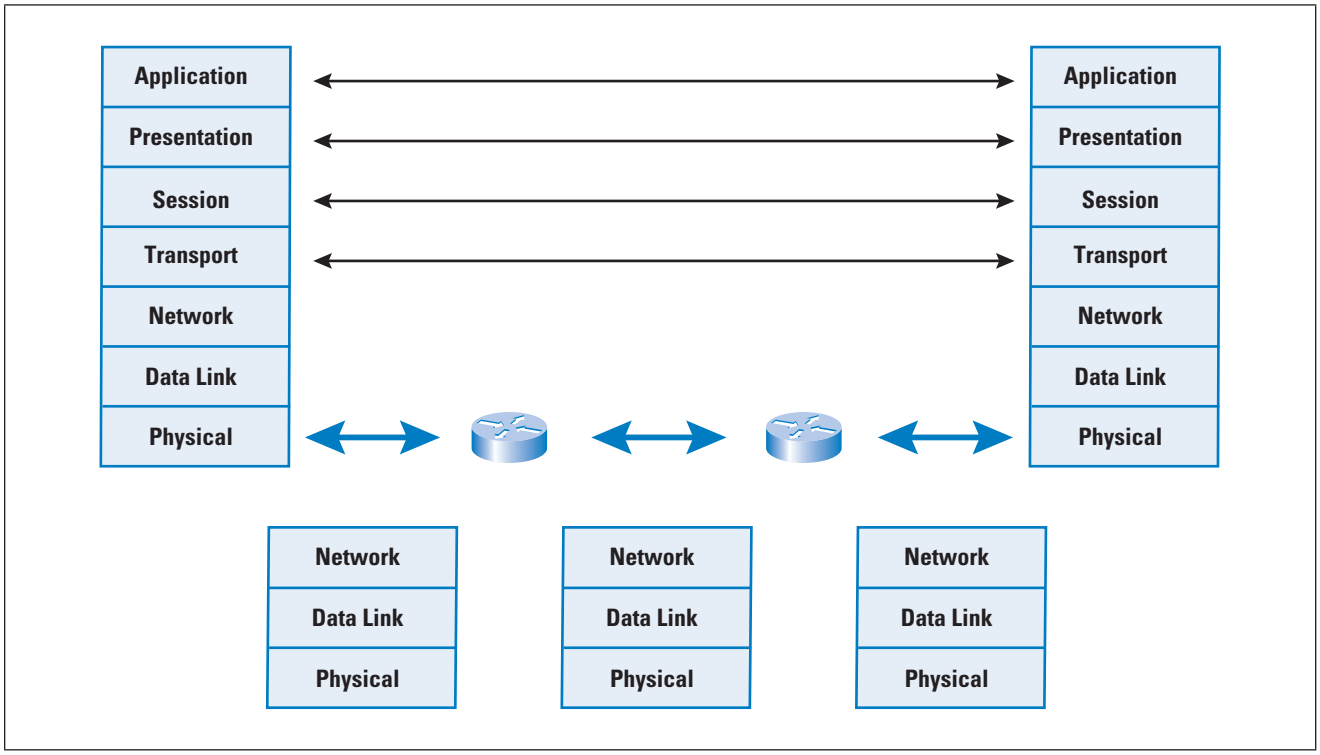
The model segmented functions into two parts:

- The *Media Layers* handle the encoding of binary data into the physical transmission media, the data link layer describes the data frames used between two inter-connected "nodes," and the network layer manages a multi-node network, including addressing and routing behaviours that manage the transmission of data between attached hosts.
- The *Host Layers* concern functions on end hosts. These layers encompass the transport layer that performs data segmentation into packets, end-to-end flow control, packet loss recovery, and multiplexing. The layers above the transport layer in the OSI model are the session layer, the presentation layer, and the application itself.

In a model of a network as a collection of interior *nodes* and a set of attached *hosts*, the interior nodes use only the network layer to make forwarding decisions for each packet that it handles, while the hosts use the transport layer to manage the data flow between communicating hosts.

The implication of this model is that there is a delineation between node and host functions and a clear delineation of the data that they need to perform their functions. Nodes do not need to have any knowledge of the settings used at the transport level, and, similarly, hosts have no need to access the network layer (Figure 2).

Figure 2: Host and Node Functions



In the context of the Internet Protocol Suite, the network-layer function is encoded as the IP header of a data packet, and the transport-layer function is encoded as the transport header, conventionally as either a *Transmission Control Protocol* (TCP) or a *User Datagram Protocol* (UDP) header (although IP also defines other headers). In terms of the Internet architectural model, a packet should be deliverable through the Internet no matter what transport header it may have attached to it.

Therefore, it should not matter in the slightest what value you put in the IP Protocol field in IP packet headers. It's really none of the network's business!

The same almost applies to the *Extension Header* fields in IPv6, although this area is one where, inexplicably, IPv6 scrambled the egg and some extension headers are addressed to network elements, namely the *Hop-by-Hop* and *Routing* Extension Headers, while the remainder are ostensibly addressed to the destination host. If Extension Headers were defined exclusively as host (or destination) extensions, then the IPv6 networks should ignore them, while if they were intended to be network options, then hosts should ignore them. Perhaps it's another of those areas where theory and practice just don't align well.

可公開的

In a strict sense the Protocol field in the IPv4 packet header need never have been placed in the IPv4 header in the first place. The particular transport protocol that the communicating hosts use is none of the network's business, in theory, meaning that if the two communicating hosts decide to deliberately obscure the transport protocol control settings from the network, then that should not matter in the slightest to the network.

In today's public Internet it appears to matter a lot that the transport protocol header is visible to the network. In fact, not only should the transport protocol be visible to the network, but the particular transport protocol that the hosts select also matters to the network. That is because many elements of today's network not only peek into the transport headers of the packets that they carry, but they also rely on the information in this transport header. Firewalls are a classic example of this reliance, but there are also *Network Address Translators*, *Equal-Cost Multi-Path Load Balancers*, and *Quality-of-Service Policy Engines*, to name a few. These network functions make assumptions about the visibility of transport headers in the IP packet in order to make consistent decisions about packet handling for all packets within a single transport flow. Often these network functions take it one step further and they process packets with the well-known transport headers (typically restricted to just TCP and UDP) and discard all else. It's even gone further than that, and we have reached the point that today's generally accepted rule is that unfragmented IP packets that contain a TCP transport header that includes one end using Port 443, and unfragmented IP packets that contain a UDP transport header where one end uses Port 53 stand the best chance of getting their data payload through to the intended destination. Every effort to augment this remarkably constrained set of packet profiles increases the probability of network-based disruption of communication.

Encrypted Transport Headers

If it is a self-limiting action to use a novel transport protocol in the public Internet, then why are we even considering the option of encrypting transport protocols to make all transport headers opaque to the network?

揭示；暴露

One answer is "Edward Snowden." When Snowden made his pervasive monitoring revelations[1], the *Internet Engineering Task Force* (IETF) responded in what could be called a "like-for-like" reaction and came to a consensus position that "Pervasive Monitoring Is an Attack"[2]. The general response to this form of insidious attack was to increase the level of encryption of Internet traffic to lift the degree of difficulty in carrying out network-based surveillance. Not only does this IETF response encompass the use of *Transport Layer Security* (TLS) to encrypt session payloads on the Internet wherever possible and shift the application behaviour profiles to make this the default action, but it also shifted our attention to other areas of Internet communication where compromise of the trust model was thought to be an issue.

The actions of the *Domain Name System* (DNS) protocol have been drawn into this IETF universal obfuscation effort, as has the transmission of transport protocol headers. We are long past the time when hosts were ill-equipped to perform encryption functions, and now robust encryption is not a luxury option with limited use, but rather something every user should reasonably expect to use as a minimum requirement. If the objective is to limit the information leakage in all aspects of the communications environment on the Internet, then the control meta-data is as important as the data itself. Applying confidentiality to transport header fields can certainly improve users' privacy and can help mitigate certain attacks or manipulation of packets by devices on the network path[3].

However, I suspect that this privacy argument is only one part of the story, and while these measures to encrypt Internet traffic play to a popular concern of the surveillance state operating in a largely unchecked manner, they may not lie at the heart of why obscuring host functions from the network is a path that some parts of the Internet ecosystem vigorously pursue today with transport header encryption.

It's not clear that the objective is here, and as with all interdependent complex systems, deliberately obscuring one aspect of the system from another typically offers both benefits and downsides. In April 2019, the *Internet Architecture Board* (IAB) published RFC 8546, titled "The Wire Image of a Network Protocol"[4]. It's a short document (9 pages) by today's RFC standards, but brevity does not necessarily imply clarity. This document appears to have cloaked its message in such a dense level of abstract terminology that it managed to say very little of practical use! The IAB document appears to have been prompted by the protracted debate in the QUIC Working Group over the use of the visible spin bit in the QUIC transport protocol[5], and I suspect that it started as an effort to argue for some levels of transport behaviour visibility to the network, but the IAB's prognostications on the topic have offered little useful or informative comment apart from illustrating the level of collective angst that this issue has generated! The IAB is not the most prolific of commentators, and any matter that provokes an IAB response, no matter how cryptic that response may be, does illustrate that the topic is one of general concern rather than being just a rather esoteric tussle buried deep down in the design of a particular protocol.

This topic of encrypted transport headers is a transport topic, so it is natural to ask whether the IETF's *Transport Area* can do any better than the IAB in providing a clear and informed exposition of the issues here. The Transport Area Working Group of the IETF has completed an RFC on this topic, "Considerations around Transport Header Confidentiality, Network Operations, and the Evolution of Internet Transport Protocols"[6]. To quote from its abstract: "This document discusses the possible impact when network traffic uses a protocol with an encrypted transport header. It suggests issues to consider when designing new transport protocols or features."

This document strikes me as an effort to produce a slightly more practically focused commentary on header encryption than the earlier IAB effort. At 49 pages it certainly cannot be considered a brief document, but does this extended commentary do any better in terms of clarity of the arguments being considered?

The document first looks at some rationales for the use of information on the network contained in headers. It cites the situation of link aggregation, and the problem of packet re-ordering in such scenarios. The common response to re-ordering is for the network to peer down into the transport header to gain a more granular view of a traffic flow than that which can be derived from source and destination IP address pairs. It is the IPv4 proxy for the IPv6 *Flow Label*. (Although the IPv6 Flow Label is so confused as to its intended role it's hard to understand how the IPv6 Flow Label field is useful in any case whatsoever!)

The document references differential service efforts that attempt to perform selective damage on traffic flows under the guise of "Quality of Service." (That "Quality" label always seems to me to have an Orwellian connotation, and a more honest label would be "Selective Service Degradation," or even just "Carriage Standover Services"). The document also enumerates the ways network operators can perform network analysis of using transport-level information, including traffic profile analysis, latency, and jitter and packet loss. However, the document strikes me as presenting a somewhat disingenuous set of rationales. For me, it is akin to a voice telephony operator justifying its eavesdropping on phone conversations on the basis of a baseless assertion that the information gathered by such wiretapping, or in other words knowledge of what people are saying to each over a telephone connection, can be used to make the telephone network better! The document also uses the last recourse of the desperate, by invoking a nebulous concept of "security," claiming that if network operators were no longer able to eavesdrop on the transport parameters of active sessions, then somehow the operator's ability to run a secure network would be compromised in some unspecified way.

Obviously, none of the rationales presented in this document can withstand much in the way of close scrutiny.

詳細的檢查

明顯的

It also appears to take a privacy-oriented stance in its analysis, and it seems to me that the privacy argument is largely an overt excuse for a more substantial difference of opinion between *content* and *carriage*. To a large extent, the issue from the perspective of the application is that the efforts of network operators to perform "traffic grooming" through transport header manipulation amounts to little more than inflicting damage on application data flows, and thereby pushes the network to a lower level of carriage efficiency. And this issue of the use of networks to selectively degrade transport performance in the name of network service quality is perhaps where we should look for the real tensions between networks and hosts in today's Internet.

馬伕

Transport Protocol Meddling 干涉

To look down this path we might want to start with the tensions between hosts and networks on the Internet.

In the telephone world, the network operator controlled all traffic. What you leased from the network was either a virtual circuit capable of passing a real-time voice conversation, or a fixed-capacity channel between two end points. If you used one of these channels, you couldn't go any faster than the contracted speed, and if you went slower, you did not release common capacity for anyone else to use. Obviously, the network charged more for leases of higher capacity. Packet networks changed all that. The network had no enforcement, and various applications (or traffic flows) competed with each other for the common transmission resource. Networks that wanted to control the allocation of shared common communications resources to clients had a problem.

This allocation control was the motive for a large body of work on the Internet during the 1990s and 2000s over what was called *Quality of Service* (QoS)[7]. The network operator wanted to offer (no doubt for some premium) a "higher-quality" service to some clients and some traffic profiles. But if a network has a fixed-capacity offering a larger slice of the network resources to some clients, inevitably it will offer less capacity to the others. One common theme of much of this work was that while it was possible for the network to disrupt a communication session in various ways to make it go slower, it was a lot more challenging (or even impossible in many ways) to make a session go faster.

Thus, in order to offer preferential treatment to a class of traffic flows, a good way was to make all the other flows go slower! The intended effect was to clear some space for sessions that were intended to be favoured to expand their sending windows and occupy this cleared network space. So-called *Performance-Enhancing Proxies* were not really able to make the selected TCP sessions go faster per se, but they were able to make other concurrent TCP sessions go slower, and thereby make some space for the selected sessions to have a lower packet-loss probability and hence achieve a higher data-throughput rate. One way of using this form is session throttling to drop packets. A subtler way, but also very effective, is to alter the TCP control parameters. If the offered TCP window size parameter is reduced, then senders will conveniently throttle their sending rate accordingly.

Here comes the reason why the network wants to meddle transport headers.

Pretty obviously, this selective behaviour of throttling active TCP sessions by networks was not something that applications viewed as a sympathetic act, and there have been two major responses from the application side. One is the use of a different congestion-control algorithm that is a lot less sensitive to packet loss and more sensitive to changes in the end-to-end bandwidth delay product across network paths. This method is called the *Bottleneck Bandwidth and Round-trip* (BBR) TCP control protocol, which is a relatively new TCP sender-side control algorithm.

But BBR is still susceptible to on-path manipulation of the TCP window size, and protecting the session from this form of network interference is where encrypted transport headers emerged and became an important objective. This response is the second one, executed by obscuring where the TCP control information is actually carried in the packet.

As we've already noted, you just can't remove a visible transport header from IP packets in the Public Internet, and even encrypting the TCP header would probably incur the same drop response from the network. But hosts have the option to ignore these transport header settings. So, while the host can't remove a visible transport header, they can make the headers meaningless.

One option is to use a "dummy" outer TCP wrapper as fodder for networks that want to peek at the transport layer and manipulate the session settings while hiding the real TCP control header inside an encrypted payload. There would be little in the way of a visible network signature that this manipulation is happening, apart from the observation that the TCP end hosts would appear to be unresponsive to manipulation of their window parameters.

However, the problem with this approach is that these days the application is actually trying not only to take control over its transport session parameters from a meddling network, but also to assert the same control over the platform in which the application is hosted. In theory, the application could use "raw IP" interfaces into the platform I/O routines, but in practice in deployed systems it is close to impossible. Platforms used in production systems tend to treat applications with suspicion. (Given the proliferation of malware, this level of paranoia on the part of the platform is probably warranted.) It is quite a challenge to disable all forms of how the platform handles the transport protocols and pass control of the transport protocol from the kernel into the applications space.

For this reason, it is logical to take the approach QUIC uses, where the shim wrapper of QUIC uses UDP as a visible transport header and pushes the TCP header into the encrypted payload part of the IP packet. UDP is close to ideal in this case as there are no transport controls in the protocol, just the local port numbers. QUIC looks to the network a lot like a UDP session that uses a TLS-like session encryption because in so many ways it is a UDP session that uses TLS. The change is that the end-to-end TCP flow control is now truly an end-to-end flow because only the two applications at the "ends" of the QUIC transport can see end-to-end transport-control parameters that are embedded in the end-to-end encrypted UDP payload. The host platform control over UDP packets is perfunctory, and the application is then allowed to assume complete control over the transport behaviour of the session.

Content Versus Carriage

Perhaps this shift to opaque transport headers goes a little further than just a desire for greater levels of protected autonomous control by applications. The shift that QUIC represents could be seen as the counter move by content providers to another round of a somewhat tired old game play by network operators to extract a tax from content providers by holding their content traffic to ransom, or, as it came to be known, a tussle over *Network Neutrality*.
爭執

There have been times when network operators have implemented measures to throttle certain forms of traffic that they asserted was using their network in some vaguely unspecified manner that was "unfair" in some way. The vagueness of all this discussion is probably attributable to a baser desire on the part of the carriage operator, which was to extort a carriage toll from content providers in a crude form of basic blackmail: "My network, my rules. You customer, you pay!"

I suspect that many carriage providers in this industry, who are witnessing the content providers take all the money off the table, believe that they are the victims here. Their efforts to restore some of their lost revenue base has meant that they are looking to restore a "fair share" of revenue in forcing the giants of the content space to pay for their share of carriage costs. However, if the enforcement mechanism of this extortion pressure is through playing with the transport-control parameters of the traffic that transits the carriage network (or, in other words, holding the traffic to ransom), then the obvious response is to push the transport controls under the same encryption veil as the content itself to prevent such real-time manipulation of the traffic profile. And this explanation of why QUIC is so important is perhaps a more compelling one.

If this situation is a tussle for primacy in the tensions between carriage and content, then it looks like the content folks are gaining the upper hand. Through encryption at every level in the host part of the protocol stack, including at the transport layer, the content folks are withholding information from the carriage providers that would allow the carriage providers to selectively discriminate and play content providers off against each other. If all that the network can do is limited to fully encrypted UDP packet streams, then one stream looks much like another, and selective discrimination is just not feasible. And if that's not enough, then padding and deliberate packet variation can blur most efforts at traffic profiling.

But when I say "content" I really mean "apps," and when I say "apps" I actually mean "browsers," so in reality I am really talking about Chrome, and when I say Chrome, I mean Google.

The massive dominance of mobile traffic in the industry and the massive dominance of Android in the mobile device environment tilts this space to an extraordinary degree.
使偏斜

Given this inherent level of control of all mobile devices, coupled with control of the majority browser platform in this space, it is hard to conceive how Google could possibly lose in this tussle. However, it is likely that if Google wins this particular battle with the carriage providers, there will be further battles to come. It is highly likely that the carriage industry will follow the lead from traditional print media and head to politicians with the case that Google's destruction of the business model for the provision of national communications infrastructure is counter to national interests, and political intervention is necessary to restore some balance into the market and allow the market for carriage to be a viable investment vehicle. Or, to put in more crudely, if Google has destroyed the residual value of the contained carriage market, then Google should now pay carriage operators to restore its viability.

At this point all technical considerations of encryption and information leakage, and even all market considerations of the viability of various business models, just walk out the door, and in their place comes a bevy of lawyers and politicians. Strategic national interest is always a strong argument to make, and when we get over the various nebulous threats by actors to quit national markets, we then get down to the real question of: "What is a tenable business relationship between carriage and content?"

朦朧的

In such a politically charged space the choices at that point are either that the various market players will compromise and reach some outcome that they can all live with, or the politicians will attempt to impose an outcome that will in all likelihood be far more disagreeable for all!

Whatever the outcome in the next few years, it should be fun to watch this drama play out. Don't forget to bring popcorn!

References and Further Reading

[0] Wikipedia, "OSI model,"
https://en.wikipedia.org/wiki/OSI_model

[1] Cullen Jennings, Brian Trammell, Christian Huitema, Bruce Schneier, Ted Hardie, Richard Barnes, and Daniel Borkmann, "Confidentiality in the Face of Pervasive Surveillance: A Threat Model and Problem Statement," RFC 7624, August 2015.

[2] Stephen Farrell and Hannes Tschofenig, "Pervasive Monitoring Is an Attack," RFC 7258, May 2014.

[3] Al Morton and Kathleen Moriarty, "Effects of Pervasive Encryption on Operators," RFC 8404, July 2018.

[4] Brian Trammell, "The Wire Image of a Network Protocol," RFC 8546, April 2019.

[5] Geoff Huston, "Just One Bit," *The ISP Column*, March 2018.
https://www.potaroo.net/ispcol/2018-03/onebit.html

[6] Godred Fairhurst and Colin Perkins, "Considerations around Transport Header Confidentiality, Network Operations, and the Evolution of Internet Transport Protocols," RFC 9065, July 2021.

[7] Geoff Huston, "QoS — Fact or Fiction?" *The Internet Protocol Journal*, Volume 3, No. 1, March 2000.

[8] Geoff Huston, "A Quick Look at QUIC," *The Internet Protocol Journal*, Volume 22, No. 1, March 2019.

[9] Geoff Huston, "Anatomy: Inside Network Address Translators," *The Internet Protocol Journal*, Volume 7, No. 3, September 2004.

[10] Dave Oran, "Considerations in the Development of a QoS Architecture for CCNx-Like Information-Centric Networking Protocols," RFC 9064, June 2021.

GEOFF HUSTON, B.Sc., M.Sc. A.M., is the Chief Scientist at APNIC, the Regional Internet Registry serving the Asia Pacific region. He has been closely involved with the development of the Internet for many years, particularly within Australia, where he was responsible for building the Internet within the Australian academic and research sector in the early 1990s. He is author of numerous Internet-related books, and was a member of the Internet Architecture Board from 1999 until 2005. He served on the Board of Trustees of the Internet Society from 1992 until 2001. At various times Geoff has worked as an Internet researcher, an ISP systems architect, and a network operator. E-mail: **gih@apnic.net**

SIP phones have replaced traditional telephones even in the offices of The Internet Protocol Journal.

20 Years of SIP — A Retrospective 回顧

by Jonathan Rosenberg, Five9

June of 2022 marked the twentieth anniversary of the publication of the *Session Initiation Protocol* (SIP), documented in RFC 3261[1,2,3,4]. When it was published in June of 2002, it set records for the longest specification produced by the *Internet Engineering Task Force* (IETF), at 269 pages. The IETF produces the technology standards that make the Internet work. Protocols like *Internet Protocol* (IP), *Transmission Control Protocol* (TCP), and *Hypertext Transfer Protocol* (HTTP)—all now part of mainstream vernacular—came out of the IETF. It was a monumental effort to produce, involving a dedicated author team that worked full time for months to ensure that the specifications were correct, consistent, and complete. I had the great fortune to be the lead author of this document, an accomplishment that was the defining moment for my career.

The RFC 3261 author team included Robert Sparks, Jon Peterson, Alan Johnston, Allison Mankin, Jonathan Rosenberg, Gonzalo Camarillo, and Henning Schulzrinne.

In the 20 years (and almost countless extensions to SIP) that followed, it is hard to dispute that SIP has been a major success. At its core, SIP enabled the transformation of the telecommunications industry from one based on hardware to one based on software—colloquially known as *Voice over IP* (VoIP). A 20th anniversary is the ideal time for a retrospective, to consider both its positives and negatives. On the plus side, this transformation resulted in the re-engineering of the phone network, the creation of new markets and market categories, and the creation of jobs and livelihoods. On the negative side, it has exacerbated the scourge of robocalling.

用通俗語，俗稱

[ɪg ˋ zæsɚ ˌbet]
使惡化

The Phone Network Re-Engineered

Prior to SIP, the telephone network was built using telephone switches based on custom hardware. These switches were made by a small set of vendors and were a completely verticalized solution from the physical networking layer to the application layer software.

With the mainstream adoption of IP networks, it became possible to replace that hardware with general-purpose computers running SIP-based software applications. This replacement resulted in a dramatic reduction of cost compared to the prior generation. SIP further reduced this cost by enabling these software applications to run on machines in a small number of data centers that might be far away from the people talking to each other, while still keeping the audio delays to a minimum. This centralization of the software was a dramatic shift in how the phone network worked.

This new paradigm had the most immediate impacts on the way corporate phone systems were built. Before SIP, businesses needed to put hardware-based phone systems called *Private Branch Exchanges* (PBXs) in each building and wire them up on a separate network from the IP network used for everything else.

SIP allowed enterprise IT departments to ditch this separate network and reuse the IP network for voice. It also allowed them to put the software in their data centers and eliminate the hardware in each building. This development represented a huge improvement in costs and reduction in complexity. The final "icing on the cake" was that SIP enabled video, instant messaging, and presence too, spawning the creation of desktop applications—called *softphones*—that allowed users to place calls, have video meetings, and chat. Businesses far and wide adopted these phones. Today, almost all business phone systems are based on SIP.

With its success in corporations, pressure grew for the phone companies (that is, the telcos), to provide a way for businesses to connect their phone systems to the rest of the phone network using SIP. Prior to this time, businesses could use software within their corporate campus, but they needed to switch to hardware to connect to the rest of the world. And so, "SIP Trunking" was born, providing a way to send and receive calls into the traditional phone network using SIP-based software applications. Its adoption was rapid, and it was the first step in transforming the edge of the telco networks from hardware to software.

Around the same time, mobile phone operators were seeing an explosion in usage due to smartphones. These mobile phones had two distinct wireless connections—an old one just for voice, and a new one for data. To expand capacity, they needed to reclaim the voice channel and use it for data. They could do it by switching the voice to VoIP, which would require them to replace their own voice hardware with SIP-based software. The wireless industry produced an expansive set of specifications on how to build a SIP-based replacement for mobile phone networks, called the *Internet Multimedia Subsystem* (IMS). IMS was finally deployed in the late 2010s. Today, most mobile phone calls use a SIP client built into the phone and traverse a SIP network deployed and operated by the mobile carriers. This change is largely invisible to mobile phone users, but not entirely. SIP also enabled the usage of higher quality wideband voice for phone calls, creating an audio experience that is more like listening to music, and you may have noticed this difference in more and more calls you make.

In a similar fashion, wireline telco providers saw a surge in demand for data. To make the jump to next-generation data access technologies like fiber, they needed to get rid of their separate voice networks and move to voice over IP too. Today, if you have one of these higher speed data networks and still have an analog phone in your home, the analog signal is converted to VoIP using a SIP client in the modem at your house, and then processed by a SIP network that the carrier operates.

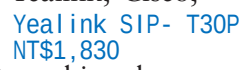
The final piece of the of the puzzle is how carriers themselves connect to each other. This process has gradually migrated to SIP too, using carrier versions of SIP trunking. This change is now accelerating, since the conversion is needed to enable the deployment of *Secure Telephone Identity Revisited* and *Signature-based Handling of Asserted information using toKENs* (STIR/SHAKEN), a SIP-based technology to combat robocalling[5,6,7].

Without a doubt, this transformation of the telecommunications technology stack—that SIP enabled—has massively impacted the world, enabling lower costs, more bandwidth for data, better quality for voice, and added video.

Market Category Creation

This transformation of the telecommunications industry also created entirely new markets and market categories that didn't exist before SIP. To enumerate just a few of them:

- *IP PBX:* The IP *Private Branch Exchange* (PBX) provides phone services for businesses. This market was created as a direct replacement for the legacy hardware-based PBX products that preceded it. Cisco Systems led this market, which never had a product in the PBX market, along with incumbents like Avaya, Siemens, and Nortel, many of which had legacy products along with the newer IP-based ones. This market is now itself shrinking, being replaced by *Unified Communications as a Service* (UCaaS).
- *SIP Trunking:* This market is estimated to be around $13B in 2021[8] and is a replacement for legacy hardware phone network access technologies.
- *SIP Hardphones:* Before SIP, the PBX vendors made their own phones, and a given phone could only work with their own hardware. With SIP, it became possible for vendors to produce phones that could work with many different IP PBXs. These phones were often produced at low cost. Vendors include Yealink, Cisco, Grandstream, and Avaya.



- *Session Border Controller* (SBC): The usage of SIP trunking drove demand for a new category of product that could serve as a SIP firewall of sorts, managing the boundary between an enterprise and a carrier, or between carriers. Ribbon and Oracle are the market leaders, with a market size estimated at USD $709M in 2022[9].
- *Internet Multimedia Subsystem* (IMS): Market leaders include Ericsson, Siemens, and Nokia. The market size was USD $1.8B in in 2019[10].
- *Communications Platform as a Service* (CPaaS): This market category is an entirely new one, enabled by the transformation of telecommunications to software. CPaaS vendors offer *Application Programming Inerfaces* (APIs) that allow developers to build telecom applications easily. These APIs allow for sending of SMSs, placing and receiving of phone calls, and so on. Twilio created this market and is still the market leader. SIP enabled the CPaaS vendors to gain low-cost and global access to telephone services, and without SIP, the market could not have existed. The market is huge and growing—estimated at USD $5.2B in 2021[11], (though most of it is for sending *Short Message Service* (SMS), where SIP has been less impactful).
- *Unified Communications as a Service* (UCaaS) puts the IP PBX in the cloud so businesses can consume voice and video communications services from the cloud.

Market leaders include RingCentral, 8x8, Cisco Systems, Microsoft, and Zoom. All of these vendors depend on SIP-based interconnection to the telephone network. This market is really big—estimated to be USD $28.9B in 2021![12].

- *Contact Center as a Service* (CCaaS) enables delivery of contact center software from the cloud, including voice response systems, agent desktop applications, and call distribution software. Vendors include Five9, Gensys, and NICE/InContact. Like UCaaS, these vendors depend on SIP to interconnect to the telephone network. This market was valued at USD $4.8B in 2021.[13]

The author (Jonathan Rosenberg) works at Five9.

When put together, SIP created or enabled these (no less than eight) distinct markets, representing approximately USD $50B in market value!

Job Creation

For me, the greatest source of satisfaction from the success of SIP is when I hear from someone that they have built their careers and their livelihood around this technology. SIP is complex, and like any complex technology that many vendors use in many ways in many markets, expertise in it becomes a marketable skill.

Many LinkedIn profiles list "SIP" as a skill. Many are software developers, but many other jobs require SIP expertise. SIP network engineers and technicians build, deploy, and operate SIP networks. Sales and marketing engineers configure and demonstrate SIP-based products. IT workers who manage business communications for their companies need to understand SIP too. A search on LinkedIn for people matching "SIP" yields approximately 239,000 results.

Many companies now exist that provide SIP certifications and training—for example the SIP School.[14] SIP is taught in many graduate classes that cover computer networking, and some even have dedicated courses just on VoIP.

https://www.thesipschool.com/

It's hard to know how many jobs SIP has created, but it would not be unreasonable to guess it is somewhere in the ballpark of 100,000 jobs. If you add the folks working in technical roles across the companies in the markets that SIP created, along with those working in telcos or in IT departments providing VOIP, it is easy to see how the number could be that large.

The Downside: Robocalling

Almost all technologies that have brought great benefits have come with some drawbacks. There is no better example than the automobile, which has brought countless benefits, but also caused 42,915 deaths in 2021 due to automobile accidents. The Internet too, has brought countless benefits, but has also brought with it problems that are becoming more apparent. SIP has had far less impact as other technologies, so its drawbacks are fewer, but they do exist.

Without a doubt, the biggest drawback has been the rise of robocalling and the fake caller IDs that come with it. Telemarketing calls predate SIP for sure. However, as SIP reduced the costs of placing calls and made it possible to make calls using off-the-shelf software, it caused a sharp increase in the volume of these unwanted calls.

在日期上早於

[ɪg ˋzæsɚˌbet] 使惡化

The problem is exacerbated by a design flaw in SIP—the lack of an authenticated caller ID. Without that, it is easy for callers to insert any phone number they want. This design defect was inherited from email, as SIP copied this aspect of its design from how email worked. After many years of failed attempts to resolve the problem, there is finally "light at the end of the tunnel" using a SIP-based technology called STIR/SHAKEN[5,6,7].

In Conclusion

It's been the highlight of my career to have had the fortune to be the lead author for a technology that, 20 years later, has had a profound impact on the world. By enabling the transformation of telecommunications from hardware to software, SIP drove a re-engineering of both mobile and wired phone networks that resulted in lower cost communications services and more bandwidth available for data. It brought video to the enterprise, created entirely new markets and some new market categories, and created at least 100,000 jobs. I try and remind myself of that fact every time I get one of those annoying robocalls.

References and Further Reading

[0] This article was adopted from Jonathan Rosenberg's blog:
https://www.jdrosen.net/blog/20-years-of-sip-a-retrospective

[1] Jonathan Rosenberg, Henning Schulzrinne, Gonzalo Camarillo, Alan Johnston, Jon Peterson, Robert Sparks, Mark Handley, and Eve Schooler, "SIP: Session Initiation Protocol," RFC 3261, June 2002.

[2] Jonathan Rosenberg and Henning Schulzrinne, "An Offer/Answer Model with the Session Description Protocol," RFC 3264, June 2002.

[3] Henning Schulzrinne and Jonathan Rosenberg, "The Session Initiation Protocol: Providing Advanced Telephony Access Across the Internet," *Bell Labs Technical Journal*, October–December 1998.

[4] William Stallings, Session Initiation Protocol, *The Internet Protocol Journal*, Volume 6, No. 1, March 2003.

[5] Numeracle, "STIR/SHAKEN: Everything you need to know about the FCC's Call Authentication Framework,"
https://www.numeracle.com/resources/stir-shaken-center

[6] IETF Datatracker, "Secure Telephone Identity Revisited (stir),"
https://datatracker.ietf.org/wg/stir/documents/

[7] Metaswitch, "What are the STIR/SHAKEN Standards?"
https://www.metaswitch.com/knowledge-center/reference/what-are-the-stir/shaken-standards

[8] The Business Research Company, "COVID-19 Impact On The Global Session Initiation Protocol (SIP) Trunking Services Market Outlook," March 8, 2022.
https://www.prnewswire.co.uk/news-releases/covid-19-impact-on-the-global-session-initiation-protocol-sip-trunking-services-market-outlook-878809709.html

[9] Future Market Insights, "Session Border Controller (SBC) Market Overview (2022–2032),"
https://www.futuremarketinsights.com/reports/session-border-controller-market

[10] Fior Markets, "Global IP Multimedia Subsystem (IMS) Market Size to Expand Significantly of USD 8.26 Billion by 2027,"
https://www.globenewswire.com/news-release/2022/05/19/2447271/0/en/Global-IP-Multimedia-Subsystem-IMS-Market-Size-to-Expand-Significantly-of-USD-8-26-billion-by-2027-Fior-Markets.html

[11] Future Market Insights, "Communications Platform as a Service (CPaaS) Market Outlook (2022–2032),"
https://www.futuremarketinsights.com/reports/communications-platform-as-a-service-cpaas-market

[12] Fortune Business Insights, "Unified Communication as a Service (UCaaS) Market Size, Share & COVID-19 Impact Analysis, By Component (Telephony, Unified Messaging, Collaboration Platforms), By Delivery Model (Managed Services, and Hosted/Cloud Services), By Organization Size (Large Enterprises, SME's), By Vertical (BFSI, IT and Telecommunications, IT-enabled Services (ITeS), Education, Retail and Consumer Goods), and Regional Forecast, 2021–2028,"
https://www.fortunebusinessinsights.com/industry-reports/toc/unified-communication-as-a-service-ucaas-market-101934

[13] Fortune Business Insights, "Contact Center as a Service (CCaaS) Market Size, Share & COVID-19 Impact Analysis, By Function (Interactive Voice Response (IVR), Multichannel, Automatic Call Distribution, Computer Telephony Integration (CTI), Reporting and Analytics, Workforce Optimization, Customer Collaboration, and Others), By Enterprise Size (Small & Medium Enterprises and Large Enterprises), By Industry (BFSI, IT & Telecommunications, Government, Healthcare, Consumer Goods & Retail, Travel & Hospitality, Media & Entertainment, and Others), and Regional Forecast, 2022–2029,"
https://www.fortunebusinessinsights.com/toc/contact-center-as-a-service-ccaas-market-104160

[14] The SIP School: **https://www.thesipschool.com/**

[15] The SIP Forum: **https://www.sipforum.org/**

JONATHAN ROSENBERG is the Chief Technology Officer and Head of AI for Five9. He was previously CTO for Cisco Webex and Skype. He has been a frequent contributor to the IETF, with 72 RFCs. He's the lead author of the Session Initiation Protocol (SIP) and related standards, such as ICE, STUN, TURN and SIMPLE. E-mail: **jdrosen@jdrosen.net**

Fragments

IAB Comments on FCC Notice on Secure Internet Routing

The *Internet Architecture Board* (IAB), which provides oversight for the protocols and procedures used by the Internet and also handles the liaison management for the *Internet Engineering Task Force* (IETF), appreciates the opportunity to submit comments in response to the *Federal Communication Commission's* (FCC) Notice of Inquiry, "Secure Internet Routing"[1]. The IETF is the main organization that works on standards relating to Internet technology. The mission of the IETF is to produce relevant technical documents that influence the way people design, use, and manage the Internet. The IETF is an open, diverse, global community of developers consisting of network operators, vendors, researchers and many other stakeholders.

The IETF originally developed the Internet protocol stack, including the routing system based on the *Border Gateway Protocol* (BGP), and continues to be responsible for maintaining and evolving the technical specifications that define the Internet and its protocols. The Internet's success has resulted from its flexible, modular architecture. BGP is the central protocol for providing global end-to-end connectivity across the world's heterogeneous network domains. It is fundamental to the operation of the Internet.

As in any protocol development, the adoption within the industry of new capabilities will vary. In recent decades, occurrences of BGP-related operational issues have increased. The existing BGP protocol stack is based on a design which can be extended, building on existing network investments. The IETF has two working groups dedicated to improving BGP interdomain routing, called *Inter-Domain Routing* (IDR) and *Global Routing Operations* (GROW). IDR is concerned with the correctness, robustness, and scalability of BGP. GROW is concerned with the operational problems associated with global routing systems, including measurement, policy, and security. The IETF will continue to evolve BGP to meet the needs of new network structures and applications, with a strong focus on security.

We believe in a continuous, modular, flexible evolution of the Internet and its protocols based on operational experience and requirements, where each service provider can determine their security needs based on their diverse requirements and in partnership with other providers. The success of future standardization efforts intended to increase routing security, will be highly dependent on educating BGP users about BGP operational issues and how well real-world deployment experience can be fed back into the multi-stakeholder standards development process, as opposed to a mandated top-down approach, which would fail to meet the diverse needs of the global community.

The FCC can support these efforts by supporting research and other work that help these communities to understand issues, develop solutions where needed, and deploy security technology more widely. The IAB believes that the IETF is an important partner in these efforts.

[1] "FCC Launches Inquiry To Reduce Cyber Risks," *The Internet Protocol Journal*, Volume 25, No. 1, April 2022, page 38.

Thank You!

Publication of IPJ is made possible by organizations and individuals around the world dedicated to the design, growth, evolution, and operation of the global Internet and private networks built on the Internet Protocol. The following individuals have provided support to IPJ. You can join them by visiting **http://tinyurl.com/IPJ-donate**

Kjetil Aas
Fabrizio Accatino
Michael Achola
Martin Adkins
Melchior Aelmans
Christopher Affleck
Scott Aitken
Jacobus Akkerhuis
Antonio Cuñat Alario
William Allaire
Nicola Altan
Shane Amante
Marcelo do Amaral
Matteo D'Ambrosio
Selva Anandavel
Jens Andersson
Danish Ansari
Finn Arildsen
Tim Armstrong
Richard Artes
Michael Aschwanden
David Atkins
Jac Backus
Jaime Badua
Bent Bagger
Eric Baker
Santosh Balagopalan
William Baltas
David Bandinelli
Benjamin Barkin-Wilkins
Feras Batainah
Michael Bazarewsky
David Belson
Richard Bennett
Hidde Beumer
Pier Paolo Biagi
Tyson Blanchard
John Bigrow
Orvar Ari Bjarnason
Axel Boeger
Keith Bogart
Mirko Bonadei
Roberto Bonalumi
Lolke Boonstra
Julie Bottorff Photography
Gerry Boudreaux
Leen de Braal
Kevin Breit
Thomas Bridge
Ilia Bromberg
Václav Brožík
Christophe Brun
Gareth Bryan
Ron Buchalski
Paul Buchanan
Stefan Buckmann
Caner Budakoglu
Darrell Budic
BugWorks
Scott Burleigh
Chad Burnham
Jon Harald Bøvre
Olivier Cahagne
Antoine Camerlo
Tracy Camp
Ignacio Soto Campos
Fabio Caneparo
Roberto Canonico
David Cardwell
Richard Carrara
John Cavanaugh
Lj Cemeras
Dave Chapman
Stefanos Charchalakis
Greg Chisholm
David Chosrova
Marcin Cieslak
Lauris Cikovskis
Guido Coenders
Brad Clark
Narelle Clark
Horst Clausen
Joseph Connolly
Steve Corbató
Brian Courtney
Beth and Steve Crocker
Dave Crocker
Kevin Croes
John Curran
André Danthine
Morgan Davis
Jeff Day
Julien Dhallenne
Freek Dijkstra
Geert Van Dijk
David Dillow
Richard Dodsworth
Ernesto Doelling
Michael Dolan
Eugene Doroniuk
Karlheinz Dölger
Michael Dragone
Joshua Dreier
Lutz Drink
Aaron Dudek
Dmitriy Dudko
Andrew Dul
Joan Marc Riera Duocastella
Pedro Duque
Holger Durer
Mark Eanes
Andrew Edwards
Peter Robert Egli
George Ehlers
Peter Eisses
Torbjörn Eklöv
Y Ertur
ERNW GmbH
ESdatCo
Steve Esquivel
Jay Etchings
Mikhail Evstiounin
Bill Fenner
Paul Ferguson
Ricardo Ferreira
Kent Fichtner
Armin Fisslthaler
Michael Fiumano
The Flirble Organisation
Gary Ford
Jean-Pierre Forcioli
Susan Forney
Christopher Forsyth
Andrew Fox
Craig Fox
Fausto Franceschini
Valerie Fronczak
Tomislav Futivic
Laurence Gagliani
Edward Gallagher
Andrew Gallo
Chris Gamboni
Xosé Bravo Garcia
Osvaldo Gazzaniga
Kevin Gee
Greg Giessow
John Gilbert
Serge Van Ginderachter
Greg Goddard
Tiago Goncalves
Ron Goodheart
Octavio Alfageme Gorostiaga
Barry Greene
Jeffrey Greene
Richard Gregor
Martijn Groenleer
Geert Jan de Groot
Ólafur Guðmundsson
Christopher Guemez
Gulf Coast Shots
Sheryll de Guzman
Rex Hale
Jason Hall
Darow Han
Handy Networks LLC
James Hamilton
Stephen Hanna
Martin Hannigan
John Hardin
David Harper
Edward Hauser
David Hauweele
Marilyn Hay
Headcrafts SRLS
Hidde van der Heide
Johan Helsingius
Robert Hinden
Asbjørn Højmark
Damien Holloway
Alain Van Hoof
Edward Hotard
Bill Huber
Hagen Hultzsch
Kauto Huopio
Kevin Iddles
Mika Ilvesmaki
Karsten Iwen
David Jaffe
Ashford Jaggernauth
Thomas Jalkanen
Martijn Jansen
Jozef Janitor
John Jarvis
Dennis Jennings
Edward Jennings
Aart Jochem
Nils Johansson
Brian Johnson
Curtis Johnson
Richard Johnson
Jim Johnston
Jonatan Jonasson
Daniel Jones
Gary Jones
Jerry Jones
Michael Jones
Amar Joshi
Javier Juan
David Jump
Anders Marius Jørgensen
Merike Kaeo
Andrew Kaiser
Christos Karayiannis
Daniel Karrenberg
David Kekar
Stuart Kendrick
Robert Kent
Jithin Kesavan
Jubal Kessler
Shan Ali Khan
Nabeel Khatri
Dae Young Kim
William W. H. Kimandu
John King
Russell Kirk
Gary Klesk
Anthony Klopp
Henry Kluge
Michael Kluk
Andrew Koch
Ia Kochiashvili
Carsten Koempe
Richard Koene
Alexader Kogan
Matthijs Koot
Antonin Kral
Robert Krejčí
Mathias Körber
John Kristoff
Terje Krogdahl
Bobby Krupczak
Murray Kucherawy
Warren Kumari
George Kuo
Dirk Kurfuerst
Darrell Lack
Andrew Lamb
Richard Lamb
Yan Landriault
Edwin Lang
Sig Lange
Markus Langenmair
Fred Langham
Tracy LaQuey Parker
Alex Latzko
Jose Antonio Lazaro Lazaro
Rick van Leeuwen
Simon Leinen
Robert Lewis
Christian Liberale
Martin Lillepuu

Roger Lindholm
Link Light Networks
Chris and Janet Lonvick
Sergio Loreti
Eric Louie
Adam Loveless
Josh Lowe
Guillermo a Loyola
Hannes Lubich
Dan Lynch
David MacDuffie
Sanya Madan
Miroslav Madić
Alexis Madriz
Carl Malamud
Jonathan Maldonado
Michael Malik
Tarmo Mamers
Yogesh Mangar
John Mann
Bill Manning
Harold March
Vincent Marchand
Normando Marcolongo
Gabriel Marroquin
David Martin
Jim Martin
Ruben Tripiana Martin
Timothy Martin
Carles Mateu
Juan Jose Marin Martinez
Ioan Maxim
David Mazel
Miles McCredie
Brian McCullough
Joe McEachern
Alexander McKenzie
Jay McMaster
Mark Mc Nicholas
Olaf Mehlberg
Carsten Melberg
Kevin Menezes
Bart Jan Menkveld
Sean Mentzer
William Mills
David Millsom
Desiree Miloshevic
Joost van der Minnen
Thomas Mino
Rob Minshall
Wijnand Modderman-Lenstra
Mohammad Moghaddas
Roberto Montoya
Charles Monson
Andrea Montefusco
Fernando Montenegro
Joel Moore
John More
Maurizio Moroni
Brian Mort
Soenke Mumm
Tariq Mustafa
Stuart Nadin
Michel Nakhla
Mazdak Rajabi Nasab
Krishna Natarajan
Naveen Nathan
Darryl Newman
Thomas Nikolajsen
Paul Nikolich
Travis Northrup
Marijana Novakovic
David Oates
Ovidiu Obersterescu
Tim O'Brien
Mike O'Connor
Mike O'Dell
John O'Neill
Jim Oplotnik
Packet Consulting Limited
Carlos Astor Araujo Palmeira
Alexis Panagopoulos
Gaurav Panwar
Chris Parker
Manuel Uruena Pascual
Ricardo Patara
Dipesh Patel
Alex Parkinson
Craig Partridge
Dan Paynter
Leif Eric Pedersen
Rui Sao Pedro
Juan Pena
Chris Perkins
Michael Petry
Alexander Peuchert
David Phelan
Harald Pilz
Derrell Piper
Rob Pirnie
Marc Vives Piza
Jorge Ivan Pincay Ponce
Victoria Poncini
Blahoslav Popela
Andrew Potter
Eduard Llull Pou
Tim Pozar
David Raistrick
Priyan R Rajeevan
Balaji Rajendran
Paul Rathbone
William Rawlings
Mujtiba Raza Rizvi
Bill Reid
Petr Rejhon
Robert Remenyi
Rodrigo Ribeiro
Glenn Ricart
Justin Richards
Rafael Riera
Mark Risinger
Fernando Robayo
Michael Roberts
Gregory Robinson
Ron Rockrohr
Carlos Rodrigues
Magnus Romedahl
Lex Van Roon
Marshall Rose
Alessandra Rosi
David Ross
William Ross
Boudhayan Roychowdhury
Carlos Rubio
Rainer Rudigier
Timo Ruiter
RustedMusic
Babak Saberi
George Sadowsky
Scott Sandefur
Sachin Sapkal
Arturas Satkovskis
PS Saunders
Richard Savoy
John Sayer
Phil Scarr
Gianpaolo Scassellati
Elizabeth Scheid
Jeroen Van Ingen Schenau
Carsten Scherb
Ernest Schirmer
Benson Schliesser
Philip Schneck
James Schneider
Peter Schoo
Dan Schrenk
Richard Schultz
Timothy Schwab
Roger Schwartz
SeenThere
Scott Seifel
Paul Selkirk
Yury Shefer
Yaron Sheffer
Doron Shikmoni
Tj Shumway
Jeffrey Sicuranza
Thorsten Sideboard
Greipur Sigurdsson
Fillipe Cajaiba da Silva
Andrew Simmons
Pradeep Singh
Henry Sinnreich
Geoff Sisson
John Sisson
Helge Skrivervik
Terry Slattery
Darren Sleeth
Richard Smit
Bob Smith
Courtney Smith
Eric Smith
Mark Smith
Tim Sneddon
Craig Snell
Job Snijders
Ronald Solano
Asit Som
Ignacio Soto Campos
Evandro Sousa
Peter Spekreijse
Thayumanavan Sridhar
Paul Stancik
Ralf Stempfer
Matthew Stenberg
Martin Štěpánek
Adrian Stevens
Clinton Stevens
John Streck
Martin Streule
David Strom
Colin Strutt
Viktor Sudakov
Edward-W. Suor
Vincent Surillo
Terence Charles Sweetser
T2Group
Roman Tarasov
David Theese
Douglas Thompson
Kerry Thompson
Lorin J Thompson
Fabrizio Tivano
Peter Tomsu Fine Art Photography
Joseph Toste
Rey Tucker
Sandro Tumini
Angelo Turetta
Michael Turzanski
Phil Tweedie
Steve Ulrich
Unitek Engineering AG
John Urbanek
Martin Urwaleck
Betsy Vanderpool
Surendran Vangadasalam
Ramnath Vasudha
Philip Venables
Buddy Venne
Alejandro Vennera
Luca Ventura
Scott Vermillion
Tom Vest
Peter Villemoes
Vista Global Coaching & Consulting
Dario Vitali
Rüdiger Volk
Jeffrey Wagner
Don Wahl
Michael L Wahrman
Laurence Walker
Randy Watts
Andrew Webster
Tim Weil
Jd Wegner
Westmoreland Engineering Inc.
Rick Wesson
Peter Whimp
Russ White
Jurrien Wijlhuizen
Derick Winkworth
Pindar Wong
Makarand Yerawadekar
Phillip Yialeloglou
Janko Zavernik
Bernd Zeimetz
Muhammad Ziad Ziayuddin
Tom Zingale
Jose Zumalave
Romeo Zwart
廖明沂.

Follow us on Twitter and Facebook

@protocoljournal

https://www.facebook.com/newipj

Call for Papers

The *Internet Protocol Journal* (IPJ) is a quarterly technical publication containing tutorial articles ("What is...?") as well as implementation/operation articles ("How to..."). The journal provides articles about all aspects of Internet technology. IPJ is not intended to promote any specific products or services, but rather is intended to serve as an informational and educational resource for engineering professionals involved in the design, development, and operation of public and private internets and intranets. In addition to feature-length articles, IPJ contains technical updates, book reviews, announcements, opinion columns, and letters to the Editor. Topics include but are not limited to:

- Access and infrastructure technologies such as: Wi-Fi, Gigabit Ethernet, SONET, xDSL, cable, fiber optics, satellite, and mobile wireless.
- Transport and interconnection functions such as: switching, routing, tunneling, protocol transition, multicast, and performance.
- Network management, administration, and security issues, including: authentication, privacy, encryption, monitoring, firewalls, troubleshooting, and mapping.
- Value-added systems and services such as: Virtual Private Networks, resource location, caching, client/server systems, distributed systems, cloud computing, and quality of service.
- Application and end-user issues such as: E-mail, Web authoring, server technologies and systems, electronic commerce, and application management.
- Legal, policy, regulatory and governance topics such as: copyright, content control, content liability, settlement charges, resource allocation, and trademark disputes in the context of internetworking.

IPJ will pay a stipend of US$1000 for published, feature-length articles. For further information regarding article submissions, please contact Ole J. Jacobsen, Editor and Publisher. Ole can be reached at `ole@protocoljournal.org` or `olejacobsen@me.com`

Supporters and Sponsors

Supporters

Internet Society

CISCO

Diamond Sponsors

Your logo here!

Ruby Sponsors

ICANN

Sapphire Sponsors

APNIC FOUNDATION

Emerald Sponsors

Akamai

amsix

APRICOT Asia Pacific Regional Internet Conference on Operational Technologies

COMCAST

DE-CIX

donuts inc

EQUINIX

Google

JPRS

JUNIPER NETWORKS

lacnic

LinkedIn

linx

netskope

NSRC Network Startup Resource Center

NTT Communications

RIPE NCC RIPE NETWORK COORDINATION CENTRE

TEAM CYMRU

VERISIGN

WIDE PROJECT

Corporate Subscriptions

AFRINIC The Internet Numbers Registry for Africa

DNS-OARC

Edgio

IPXO

ISC Internet Systems Consortium

IWL

qa|cafe

For more information about sponsorship, please contact **sponsor@protocoljournal.org**

The Internet Protocol Journal
Link Fulfillment
7650 Marathon Dr., Suite E
Livermore, CA 94550

CHANGE SERVICE REQUESTED

The Internet Protocol Journal is published quarterly and supported by the Internet Society and other organizations and individuals around the world dedicated to the design, growth, evolution, and operation of the global Internet and private networks built on the Internet Protocol.

Email: ipj@protocoljournal.org
Web: www.protocoljournal.org