The Internet Protocol Journal

October 2021 Volume 24, Number 3

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, *Autonomic Computing* refers to the self-managing characteristics of distributed computing resources, adapting to unpredictable changes while hiding intrinsic complexity to operators and users. The concept has been expanded to computer networks by the *Autonomic Networking Integrated Model and Approach* (ANIMA) working group of the *Internet Engineering Task Force* (IETF). They recently published six *Request For Comments* (RFCs) about autonomic networking. Our first article provides an overview of the ANIMA model and describes these specifications and several usage scenarios in detail.

During the last two weeks of September 2021, several *Voice over IP* (VoIP) providers became the target of a *Distributed Denial of Service* (DDoS) attack. The victims included the provider that I use for my office telephone service, as well as its upstream provider. According to some reports, the attack left several critical institutions without telephone service, including some 911 emergency call centers. As I write this, my service appears to have been restored, but only after a large-scale re-engineering of the network that my provider uses. DDoS mitigation is not an easy task, especially for services that are real-time in nature such as telephone calls. Although I don't expect to learn all of the details of this incident, I do hope that we can cover the topic in more general terms in future articles. If you know any experts on DDoS mitigation, please ask them to get in touch! In the meantime, check out the article entitled "May I ask who's calling, please? A recent rise in VoIP DDoS attacks," which you can find on *The Cloudflare Blog*.

Security has been a recurring theme in this journal. Most of the protocols used in today's Internet were originally designed without comprehensive security in mind, but the IETF has produced security enhancements for many of the core protocols. Securing the routing system itself has proven challenging because it requires wide-spread deployment in order to be effective. In this issue, Geoff Huston presents Part 1 of a two-part article entitled "A Survey on Securing Inter-Domain Routing."

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

Autonomic Networking Gets Serious

by Michael Behringer, independent;
Carsten Bormann, Universität Bremen TZI;
Brian E. Carpenter, The University of Auckland;
Toerless Eckert, Futurewei;
Jéferson Campos Nobre, UFRGS;
Sheng Jiang, Huawei Technologies;
Yizhou Li, Huawei Technologies; and
Michael C. Richardson, Sandelman Software Works Inc.

In May 2021, six *Request For Comments* (RFCs) about autonomic networking were published [5–10] as a result of the work of the *Autonomic Networking Integrated Model and Approach* (ANIMA) working group of the *Internet Engineering Task Force* (IETF). These RFCs complete the initial charter of that working group, which was started in late 2014 (see [11] for a summary of its inception); however, the first documents to be discussed in the IETF and *Internet Research Task Force* (IRTF) were posted in 2012[13]). This foundation now allows the industry to build IETF-standardized network solutions for an *Autonomic Networking Infrastructure* (ANI) into every network device.

This article starts with an overview of the reasoning behind autonomic networking and a description of an early usage scenario. It then gives an overview of the newly published specifications and how they will interwork with existing network management, before concluding with several specific use cases.

One way to summarize autonomic networking is "plug and play" for professional networks. It can mean plug and play "for the ISP" or "for the enterprise" or "for industrial networks." This step is a significant one forward from the well-known idea of plug and play for home networks, which the IETF addresses in the HOMENET working group.

IBM coined the term "autonomic computing" in 2001. The autonomic nervous system acts largely unconsciously and regulates bodily functions such as heart rate. IBM defined autonomic computing as "self-managing distributed computing resources, adapting to unpredictable changes while hiding intrinsic complexity from operators and users." This definition led naturally to the idea of autonomic networking, which became a topic of discussion and work in the IRTF *Network Management Research Group*. The result was RFCs [1,2], which describe the outline of an envisioned autonomic networking infrastructure and ultimately resulted in the creation of the ANIMA Working Group. Since then, various aspects of the problem space were addressed in research, and in proprietary implementations by some vendors. But as always, the need is for interoperability, so proprietary methods have to give way to industry standards. This interoperability task is the job of the ANIMA Working Group.

The goal is self-management of networks, including self-configuration, self-optimization, self-healing, and self-protection (sometimes collectively called *self-X*). Autonomic networking puts operational intelligence into algorithms at the node level, to minimize dependency on human administrators and central management. Autonomic nodes will discover information about the surrounding network and negotiate parameter settings with their neighbors and other nodes. Later, nodes may also have learning and cognitive capability, that is, the ability to self-adapt their decision-making process based on information and knowledge sensed from their environment.

Science fiction? Not really. Distributed routing protocols as introduced with the ARPANET in the 1970s and later in the Internet are at their core autonomic: self-configuring, self-optimizing, and self-healing. Examples include *Open Shortest Path First* (OSPF) and *Intermediate System-to-Intermediate System* (IS-IS). But over the decades, even those protocols have evolved to become provisioning monsters requiring the human configuration of obscure parameters and policies. A whole industry and research discipline for network *Operations, Administration, and Management* (OAM) evolved to define architectures consisting of ever-more-complex layers between the human intent for the service-level objectives of the network (and by implication its protocols) and all the detailed parameters that need to be provisioned consistently and dynamically into each network device whenever there is any change. (As evidence, consider that the IETF alone has published more than 120 *Yet Another Next Generation* (YANG) modules and sub-modules, each of which contains many individual parameters.)

In today's networks, routing and traffic-engineering parameters are almost exclusively implemented through a centralized set of *Software-Defined Networking* (SDN) controller and orchestrator tools configured by human operators. Although a great improvement on older methods, these solutions are still difficult and expensive to build, maintain, validate, predict, secure, and above all to make reliable and resilient. These problems are rarely seen from the outside, except when network services are under oversight of regulatory entities that publish reports of those problems, such as [12].

SDN architectures are also highly proprietary—very often from a single vendor—and they typically require significant customization through programming for any multi-vendor network deployment. They therefore require network owners to not only hire network operators, but also have them become SDN developers. And sometimes, expensive experts have to travel unexpectedly at any hour of the day to fix or update systems. These issues largely arise because of the lack of the automation inside switches and routers that autonomic networking aims to enable.

Nevertheless, these SDN methods are the best option for existing large networks. They are marketed with terms that evolved in the last few years, such as *Zero-Touch Networks, Intent-Based Networking,* or *Self-Driving Networks*. In the metaphor of a network being a car, today's networks are like children's pedal cars guided from behind by an attentive parent, whereas ANIMA wants them to be like a self-driving car.

The long-term vision for autonomic networking is broader than the newly published standards. The autonomic networking infrastructure defined in the recent ANIMA RFCs is intended to provide foundational building blocks. These building blocks are meant to fit seamlessly with existing network and SDN/OAM designs and to improve their metrics such as simplicity, reliability, and security. Likewise, the ANI allows designers to more easily embed automation into network devices whenever there is a need. It is worth noting that today, unlike in the past, it is economic to provide enough computing power in network elements to support autonomy.

What Can the Autonomic Networking Infrastructure Do for You?

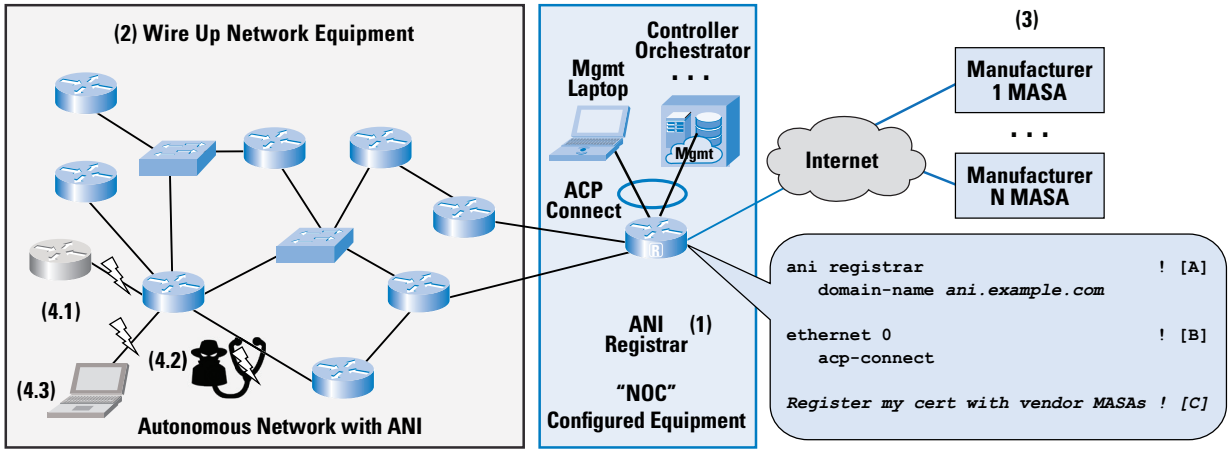
Instead of jumping directly into an explanation of how the ANI works, we first give a simple example of what the operator experience of a simple autonomic network could be.

In Figure 1, an operator wants to deploy a new network of devices such as routers and switches, namely those in the box labeled (2). These devices may be scattered across different physical locations, such as different offices or buildings. The actual reception of the new, factory-fresh equipment, unpacking, and physical attachment to pre-existing links may be performed in different locations by personnel who need to know only how to connect power and network cables accurately.

In contrast, without autonomic solutions, this process is very complex, insecure, and error-prone, and the description of all the challenges experienced would be much longer than this article. The challenges may be as simple as connecting a new device into a wrong Ethernet port, whereas any port would work for autonomic bootstrap. An operator must often ask the local installer to repeatedly power-cycle a device to activate a new or fixed configuration, a process that will be automatic in the ANI. In the worst case, the operator must ask the local installer to perform complex actions such as connecting a laptop to the device and configuring obscure and badly documented features. This situation can result in bizarre telephone interactions such as the operator asking the installer "Please take a photo of that screen and message it to me."

To avoid this situation, many device installations nowadays are done by staging. The device is first shipped to a central location where expert operators pre-configure and secure it on a trusted network, and then it is shipped again to the final deployment location.

Figure 1: An Example Autonomic Network



This process is more secure and more predictable, but it is a lot more expensive and slower. Eliminating the need for staging is hence one of the main advantages of the autonomic bootstrap process.

With the ANI, the operator only sets up a seed router—called the *ANI registrar*—for example in a *Network Operations Center* (NOC). The rest is fully automatic and secure, with local installation of new equipment by less-expert personnel ("plug in power cable, plug data cable into any free Ethernet port"). The NOC setup consists of only three simple steps:

A. Set up the router labeled (1) as the registrar and assign a name to the ANI.

B. Configure some local port(s) to provide link-layer access to the ANI, to connect management equipment such as a laptop for manual access or an SDN controller.

C. Register the certificate of the registrar with the *Manufacturer Authorized Signing Authority* (MASA) services of the vendors whose routers and switches are being used in the new network. (We will soon describe what that registration does.)

Before this seed setup is in place, you may physically interconnect new routers or switches (2), but they will not do anything. When they have connectivity to a configured registrar, they will automatically form an ANI as follows:

Each new ANI device (at that stage called a *pledge*) automatically obtains a connection with the ANI registrar and attempts to enroll, receiving an ANI certificate so that it can participate. But the registrar first needs to prove to the ANI device that it is its "owner." To do that, the registrar communicates (for example over the Internet) with the MASA of the vendor of that device.

That MASA has the information that this pledge is actually owned by this registrar's network and returns a security voucher that the registrar can present to the pledge, such that the pledge may now trust the registrar and therefore accepts an ANI certificate from the registrar. This process runs completely automatically without any further hand holding or configuration. This part of the ANI is known as *Bootstrapping Remote Secure Key Infrastructure* (BRSKI)[10] (pronounced "Brewski").

After a new device is enrolled with an ANI certificate, it begins to establish a secure *Autonomic Control Plane* (ACP) connection with all its neighbors, authenticated and authorized mutually by ANI certificates of the device. This step too happens without further handholding or configuration. ACP connectivity is always established or re-established between any neighboring ANI routers or switches, regardless of any change in topology. It cannot be affected by faulty operator or SDN configuration of these devices. The goal of the ACP is quite simple: *If there is a physical path to a router or switch, the ACP will automatically provide encrypted and authenticated IPv6 connectivity to it that an operator cannot remove or misconfigure.* This function is exactly the type needed to avoid operational breakdowns such as [12].

Assume all devices were physically connected to each other as shown in Figure 1 and the ANI registrar is connected last (after it was configured). As a result, within minutes, all the devices will have run through BRSKI and set up the ACP. As a result, the network operator now has secure IP connectivity over the ACP from the management laptop and SDN controller to all ANI devices and can configure them manually or through SDN automation using this connectivity. Each ANI device has a permanent and private IP address within the ANI that does not change, even if the device is physically moved in the network.

How is this procedure different from 30-year-old Ethernet technology? Surely you can simply buy a set of inexpensive Ethernet switches, interconnect them, attach a configuration system at one point, and have achieved the same thing?

Indeed, the simplicity of operating Ethernet networks was an inspiration for the ANI, but beyond that, the ANI is fundamentally different. The ANI is above all secure, whereas the default behavior of traditional switches is not. An ANI device can join the ANI only if the operator actually owns it, as cryptographically certified by its manufacturer's MASA, for example via sales records, meaning that a stolen device cannot be enrolled for the ANI in another network. It also means that a device not belonging to this network operator (4.1) cannot be enrolled in this ANI network to launch an attack. To be clear, the operator has not relinquished any control or authority to the manufacturer by this process; only the operator decides which devices may attach to the network and what they may or may not do. The manufacturer's only role is to certify that each device is genuine.

All ACP traffic is encrypted hop-by-hop; therefore, an attacker cannot snoop or spoof any management traffic that uses the ACP, including any legacy unencrypted management protocol (4.2).

Lastly, ANI devices, even after having formed the ACP, are still unconfigured, ideally meaning that they should behave like current unconfigured routers: there is nothing running that could provide undesirable network connectivity to any hosts that attach, like some insecure or malicious laptop (4.3). Such an attached device would get no connectivity whatsoever. As a result, there is never a window of opportunity for attackers to impair unprotected equipment. Instead, the NOC has all the time it needs to remotely provision the devices. In later stages, such provisioning will occur autonomically, as we shall see.

Compared to many other zero-touch solutions, the ANI does not focus only on so-called *day-0/day-1* behavior up until the network is operational. Instead its services last through the whole life cycle. The ANI provides automated certificate renewal for all ANI devices to maintain and refresh its security model. The ACP protects any network OAM traffic that uses it. By its use of hop-by-hop encryption it also continuously protects the whole network and attached OAM equipment from traffic injection or spoofing attacks.

The use of the MASA service is one of the crucial benefits of the ANI process to enable reliable and secure device deployment without prior staging. Without a MASA, if an unconfigured device is connected to an unintended or hostile network, systems that use its default credentials can easily "kidnap" it. Furthermore, an attacker could then intercept the enrollment process in order to gain access to the whole network. For a network connection to become hostile, it is often sufficient for some virus-impaired device (such as a PC) to be on the same LAN or for the attacker to have impaired other network services such as the *Domain Name System* (DNS). Using a MASA to restrict access to cryptographically authorized devices closes off this avenue of attack.

Nevertheless, the MASA concept has raised concerns over the extent of control or observation by the manufacturer. In fact, the MASA can do neither. It can only generate cryptographic vouchers to inform the device and the person who owns it, thereby precluding configuration by anyone else. Manufacturers can operationalize this service in many ways, according to their customers' requirements. The workflow described previously, where the owner communicates with the MASA during the enrollment of the device into its owner's network, is just the simplest option for many owners because it offloads the difficult steps onto the manufacturer.

> *When a distinguished but elderly scientist states that something is possible, he is almost certainly right. When he states that something is impossible, he is very probably wrong.*[14]

Technical Outline of the ANIMA Model

As always in network management, literally thousands or millions of details cannot be standardized, or even described centrally. What we can do is define a model, a platform, and a toolkit, just as the *Simple Network Management Protocol* (SNMP) and the *Network Configuration Protocol* (NETCONF) have done in the past.

The main terminology we will use is the following. More details about these terms is available in RFC 7575[1] and RFC 8993[8]:

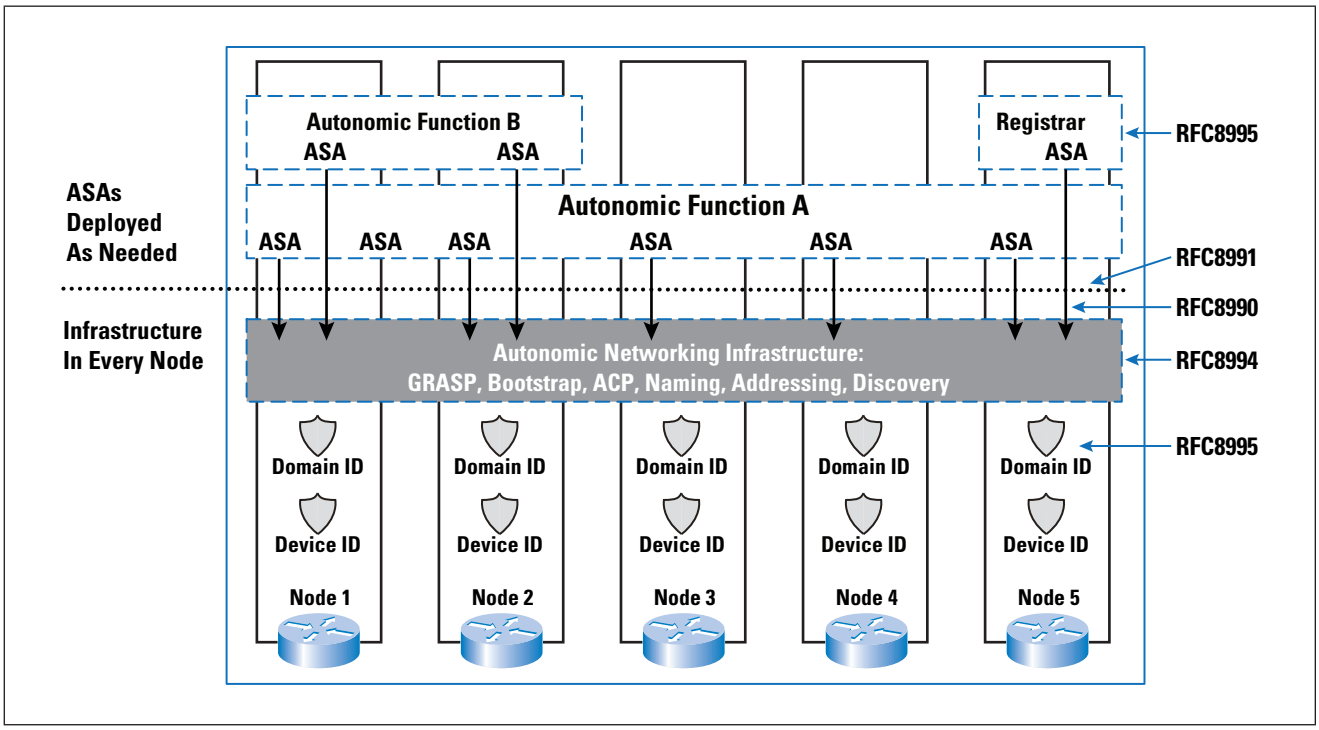
- *Autonomic Function:* A specific self-managing feature or function
- *Autonomic Service Agent* (ASA): An agent that implements an autonomic function, in part (for a distributed function) or whole
- *Autonomic Node:* A node that embodies autonomic functions
- *Autonomic Control Plane* (ACP): A self-configuring, fully secure, virtual network used for all autonomic messaging

The main items in the model follow:

- Bootstrapping and trust infrastructure[10]. This item covers how nodes are authenticated and securely admitted to an autonomic network, and how they establish mutual trust.
- Secure *Autonomic Control Plane* (ACP)[9]. This part is an automatically constructed and encrypted virtual network that contains only authenticated nodes that rightfully belong to a particular autonomic domain.
- Discovery for autonomic nodes: This item is a mechanism by which nodes attached to the ACP can discover each other. In practice, discovery occurs at a finer grain than nodes, because it really operates at the level of the capabilities and objectives of a node.
- Negotiation and synchronization for autonomic nodes: After nodes have discovered each other, they can synchronize data between themselves, or actively negotiate parameters and resources.
- Autonomic functions operate by negotiating and synchronizing data with their peers in other nodes, and by directly configuring manageable devices in their own scope.
- Discovery, synchronization, and negotiation proceed by use of the *GeneRic Autonomic Signaling Protocol* (GRASP)[5].
- *Autonomic Service Agents* (ASAs) are composed of one or more autonomic functions, typically using GRASP via an *Application Programming Interface* (API)[6].
- Centrally defined policy or configuration rules may be obtained by an ASA via GRASP synchronization, or if appropriate by conventional methods such as an interface to NETCONF or *Domain Name System Service Discovery* (DNS-SD).

Figure 2 shows an outline of the model as a whole, described in detail in RFC 8993[8].

Figure 2: Layered Model of Network with Autonomic Functions



The only way of discovering the limits of the possible is to venture a little way past them into the impossible.[14]

Some Details of Self-configuring Security

ANIMA does not attempt a monolithic bootstrap of a network from a predefined configuration. Instead, it proceeds step-by-step, and security comes first. The first stage of creating a secure autonomic control plane is bootstrapping a suitable key infrastructure that covers all the nodes that will constitute the ACP. This process is done, as previously described, by BRSKI.[10] The process uses manufacturer-installed X.509 certificates (in IEEE 802.1AR IDevID format), in combination with a MASA. The network administrator decides which devices are authorized to join the network (for example, by serial number), but relies on the manufacturer to validate the certificate of each device whenever the device attempts to join the network via a local "join proxy." These proxies all use a single "domain registrar" node that mediates the authorizing service. The join proxies themselves join the network by the same process; a GRASP mechanism is used for joining nodes (known as *pledges*) to find proxies, and for proxies to find each other and the registrar. Only the registrar needs to be configured in advance.

The ACP forms itself among pledges as soon as the pledges have completed their BRSKI enrollment. It is best described as a *Virtual Routing and Forwarding* (VRF) instance. It is based on a virtual router at each node, consisting of a separate IPv6 forwarding table to which the virtual interfaces of the ACP are attached, and an associated IPv6 routing table separate from the data plane.

Packet transmission is visible only as IPv6 link-local packets, encapsulating the autonomically created overlay network. This choice was made to ensure that there is no dependency on any pre-existing data plane (either IPv4 or IPv6), because autonomic functions must be able to operate *even if the normal data plane and normal routing are broken*. Even then, the ACP provides a secure channel to reach each node for (re-)configuration, without requiring a physically isolated console port. To start the ACP, all that is required is for each node to create its own IPv6 link-local address on each physical interface, as any modern network device does by default. The VRF consists of point-to-point IPv6 links and is secured using *Internet Protocol Security* (IPsec) with *Internet Key Exchange Protocol Version 2* (IKEv2) or *Datagram Transport Layer Security* (DTLS). From the viewpoint of autonomic service agents, the ACP uses an automatically generated IPv6 Unique Local Address prefix, and it uses *Routing Protocol for Low-Power and Lossy Networks* (RPL) internally. Like BRSKI, the ACP bootstraps itself, starting with a GRASP-based discovery process.

The security that the ANI itself requires is a simple but effective based "group-walled-garden" model for *Private Key Infrastructure* (PKI). It provides strong protection against intruders because of its certificate-based model with automated renewals. It also provides for simple ejection of impaired nodes through certificate revocation, certificate status verification, or short-lived certificates. Further levels of security are easily added when necessary. For example, the ANI itself already uses the common certificate-derived role-based security that distinguishes registrars from other nodes, so that no arbitrary impaired node can overtake the domain by acting as a fake registrar. You can expand such role-based security to other crucial roles in autonomic functions.

Of course, it would be naive to assume that, even with this key infrastructure and encrypted network, no malicious device, code, or user will ever penetrate the autonomic system. A malicious ASA could, for example, attempt a *Denial of Service* (DoS) attack within the ACP. The ANI platform provides services such as authentication, confidentiality, credential management, connectivity, and discovery to ASAs. An interesting analogy is *Transport Layer Security* (TLS), which provides authentication and confidentiality to web services. However, TLS cannot prevent the web services themselves from being untrustworthy, for example by breaking expectations of confidentiality by selling user data. In the same way, ASAs need to be intrinsically trustworthy on their own, regardless of whether they use the ANI. All legitimate ASAs should be designed to take appropriate precautions, and a watchdog ASA could be implemented to detect suspicious activity.

After the secure control plane has configured itself, the next stage is to bootstrap connectivity for network management. When this connectivity is achieved, conventional mechanisms (such as an SDN controller) can already reliably and securely reach remote nodes and configure them safely without risk of cutting themselves off.

In addition, fully autonomic management mechanisms (that is, ASAs) can start up. To understand how this process works, we first need to add more details about the GRASP protocol.

GRASP

The *GeneRic Autonomic Signaling Protocol* (GRASP)[5] is used for signaling between ASAs, including special-purpose mini-ASAs that support BRSKI (discovery of join proxies and the domain registrar) and ACP creation (discovery of ACP neighbors). Readers will notice that these operations must take place *before* ACP security is in place, so they use a highly restricted subset of GRASP that is limited to specific link-local operations.

After that, GRASP runs over the ACP to guarantee security, so there are no restrictions on allowed operations and any two ASAs in the local domain may trust and communicate with each other. GRASP provides discovery, flooding, synchronization, and negotiation mechanisms for the objectives that ASAs support.

Rather than being a traditional type-length-value protocol, GRASP messages use *Concise Binary Object Representation* (CBOR), which provides an extensible data model derived from *JavaScript Object Notation* (JSON), but with a simple and efficient binary encoding. The flexibility of CBOR enables GRASP to accommodate a very wide range of data types, with protocol elements often mapping directly into various high-level language representations.

The word "objective" has a special meaning in GRASP. It is a data structure whose main contents are a *name* and a *value*. An objective occurs in three contexts: *discovery*, *negotiation*, and *synchronization*. A single ASA may support multiple independent objectives.

The *name* of an objective is simply a unique string describing its purpose.

The *value* consists of a single configurable parameter or a set of parameters of some kind. The parameter(s) apply to a specific service, function, or action. They may in principle be anything that can be set to a specific logical, numerical, or string value, or a more complex data structure. Basically, an objective is defined in the way that best suits its application; that is the great advantage of CBOR encoding. If desired, for example, the *value* of an objective could be expressed in the JSON data model. When an objective is shared between ASAs by flooding, synchronization, or negotiation, each ASA will maintain its own copy of the objective and its latest value.

GRASP messages allow for *discovery* of an ASA that handles a given objective name; *flooding* a given objective to all ACP nodes (the simplest form of synchronization); *synchronization* of the value of a given objective between two peer ASAs; and *negotiation* of the value of a given objective with a peer ASA.

An API for GRASP has been defined[6] and implemented as part of a *Python* 3 prototype, making it very easy to implement demonstration ASAs in *Python*. A partial GRASP implementation has also been made as part of an ACP implementation in the *Rust* language.

Talking to the NOC

As noted previously, a key requirement for the success of ANIMA is smooth integration with existing network management tools and in particular with NOCs. To this end, an integration mechanism has been documented.[4] The simplest approach is for trusted edge devices in the ACP to "leak" the (otherwise encrypted) ACP natively to certain network management hosts, presumed to be well secured. These edge devices would act as default routers to those management hosts and provide them with IPv6 connectivity into the ACP. A more complex approach would allow the management hosts simultaneous connectivity into the ACP and the traditional data plane.

A related issue is that if the NOC uses *DNS Service Discovery* (DNS-SD) to announce management services to managed nodes, these announcements will not be automatically available in the ACP, which for security reasons will not have routed access to the data plane where the DNS is available. This situation again can be solved by a trusted edge device that obtains service information from DNS-SD and redistributes it within the ACP, possibly by the GRASP flooding mechanism. For example, the information for a service named *syslog* could be flooded in a GRASP objective named *SRV.syslog*. Here, the flexibility of CBOR encoding is of great value because a JSON-like structure of service data is common.

Extending that point, since GRASP easily conveys JSON (or practically any other format), it is possible to integrate ASAs communicating via GRASP into almost any part of an existing network management system. For example, an ASA acting as a NETCONF client could retrieve YANG documents from a NOC database via GRASP and the ACP.

Autonomic Function Example 1: Address Management

A use case that has been fully defined is a GRASP-based mechanism for managing and assigning IP address prefixes.[7] Firstly, we define two GRASP objectives for IPv4 or IPv6 prefix management at the edge of large-scale *Internet Service Provider* (ISP) networks. The first objective can be represented thus (in a simplified form):

```
["PrefixManager", [IP_version, prefix_length, prefix]]
```

and the second as:

```
["PrefixManager.Params", parameter_info]
```

The first objective will be used in GRASP negotiations between two "prefix manager" ASAs in nodes that need to delegate address space to subsidiary routers (using standard IPv6 prefix delegation), when one node is short of spare prefixes and the other one has an adequate pool of unused prefixes. If negotiation succeeds, prefixes will be transferred from the pool of one ASA pool to the other ASA pool. If negotiation fails, the ASA that is short of prefixes will use GRASP discovery to find another ASA that can help it. Each participating ASA will require persistent storage to manage its own address pool and to survive power outages or other failures such as network partitions. This feature will completely obviate any need for human management of an ISP's distributed pool of prefixes, beyond initially configuring the maximum pool in one place.

The second objective may be flooded to all "prefix manager" ASAs to convey relevant policy, which can be enforced during prefix delegation by individual agents. For example, if the flooded parameter information is as follows:

```
[
   [["role", "A"],["prefix_length", 34]],
   [["role", "B"],["prefix_length", 44]],
   [["role", "C"],["prefix_length", 56]]
]
```

...it would mean that devices of type A are allowed to receive IPv6 prefixes of length 34 bits, and so on.

You could use this mechanism in a variety of ways. One use case is where the three roles previously discussed correspond to three functions in an IP Radio Access Network: Radio Network Controller Site Gateways, Aggregation Site Gateways, and Cell Site Gateways. These devices will determine their own roles, and then select the prefix length they are allowed to request and offer to each other accordingly. Only central actions are to define the policy to be flooded out and to assign the operator's total address space to a single device that will progressively delegate it to gateways that request prefixes.

This example illustrates that GRASP's use of CBOR and its easy representation of JSON-like formats gives it great expressiveness and flexibility. While much work remains to be done on individual autonomic functions, the ANI and GRASP provide a solid and flexible foundation for further development.

Autonomic Function Example 2: Automating IP Multicast

One common interesting challenge for writing distributed autonomic service agents is solving problems that require decisions about the network topology—in a distributed fashion.

A simple example is automating deployment of a service such as IP Multicast, which needs to determine a small set of designated rendezvous routers, where a key requirement is their location balanced between the center and the edges of a network.

Using the ANI and GRASP, it is practical to build such distributed algorithms, for example using common criteria, such as calculation of one's own average path length as an indicator of centrality, and then running a distributed election algorithm that accounts for this and other criteria such as node performance and speed of attachment links to elect a few top contenders for the role, which then auto-configure the service and their precedence in it.

Autonomic Function Example 3: Automatic Protocol Security

We will end by considering an important early operational role for distributed autonomic behavior. That could start soon with very pragmatic incremental in-network automation, perhaps developed by operators as simple scripts in a scripting language such as *Python* or *Tcl* that can run locally on routers.

Consider an existing network where basic services are already running, for example, IPv4 and/or IPv6 addressing and routing. A software upgrade to the routers that adds support for the ANI could be installed, without affecting any of the pre-existing configuration and services. One of the most desirable services is protocol security, for example in routing protocols such as OSPF, IS-IS, and many others.

Most protocols have their own security mechanism and/or keying material requirements. However, security is often not configured because there is no automated key management, including key rollover and revocation. Without good automation of key management, either networks fail to enable protocol security, or operators set up a single, network-wide password that is never changed. With the ANI, automation of such functions becomes much simpler, by using GRASP, running securely inside the ACP.

With this information in mind, you could easily write a *Python* or *Tcl* script using the GRASP API to auto-configure routing protocol security:

- *Discover* ANI neighbors on links that use the same routing protocol.
- *Generate* a random key.
- *Negotiate* the key with a neighbor.
- *Configure* a routing protocol key locally on the router.
- Periodically wake up, renegotiate, and configure a new key.
- Take suitable action if a neighbor disappears or re-appears.

Some protocols may not even have security included in the protocol itself, for example *Protocol Independent Multicast* (PIM). Instead, you need to secure packets via *IPsec Security Associations* (SAs). For those protocols, the previous script would then auto-configure the IPsec SA instead of an in-protocol key parameter. Such scripts are, of course, autonomic service agents by another name.

In summary, GRASP with ANI can solve the recurring core problems of in-network automation between routers:

Q: How do I communicate with a peer (link-local or across other routers) without having any configured IP connectivity?

A: ACP provides this connectivity automatically with no human intervention.

Q: How do I discover what peers with what type of services are available (especially when not link-local)?

A: GRASP discovers the peers.

Q: Should I trust these peers?

A: Your trust comes from the ANI certificate used for the ACP. No nodes that have not been registered for the domain and authenticated by their manufacturer can join.

Q: How can I avoid re-inventing a new protocol to coordinate with peers?

A: Use GRASP.

Securing existing protocols is only one example where you can use ANIMA immediately. Many or all the benefits apply equally to any other in-network function with similar issues: establishing and adjusting *Quality of Service* (QoS) and other policies; auto-configuring decentralized protocol instances; monitoring, fault isolation, and troubleshooting; and even auto-configuring the most basic user network configuration, such as IP prefix distribution as in the previous example. When completely new services are required, ASAs should be developed in languages best suited for such a task. This immediate applicability to real-world problems provides a significant deployment incentive.

Summary and Conclusion

The ANI is a foundation for network automation and it serves two purposes:

- For existing network OAM designs it provides core functions to more easily build and deploy networks with secure, resilient network management. ANI provides automated public key deployment and renewal and zero-touch auto-configured in-band network management connectivity that is protected from being brought down by operator or network management tool errors.
- For ongoing further automation of network OAM (with or without an ultimate goal of fully autonomic networking), the ANI provides fundamental functions to build distributed, in-network automation agents (ASA) without having to re-implement their core dependencies each time: security, mutual trust, connectivity, and network-wide and peer-to-peer common signaling (via GRASP).

As a system, ANI may look overwhelming at first with its large set of constituent components (buzzword bingo), but it is fundamentally a very pragmatic approach, with the goal of making network complexity self-managing.

- *The basis of ANI is a set of long-term, well-known, and widely-used protocol components:* IPv6, X.509, IPsec, DTLS, RPL, CBOR, etc.
- The core innovations of ANI are built on top of this foundation: BRSKI, Voucher, MASA on top of X.509, ACP on top of IPsec, DTLS and RPL, and GRASP on top of CBOR.
- *ANI is highly modular:* All components are defined to be fully reusable individually or in concert. Adopt and deploy only the subset you need.

> *Any sufficiently advanced technology is indistinguishable from magic.*[15]

References and Further Reading

[1] Alex Clemm, Michael Behringer, Sheng Jiang, Max Pritikin, Laurent Ciavaglia, Steinthor Bjarnason, and Brian Carpenter, "Autonomic Networking: Definitions and Design Goals," RFC 7575, June 2015.

[2] Michael Behringer, Sheng Jiang, and Brian Carpenter, "General Gap Analysis for Autonomic Networking," RFC 7576, June 2015.

[3] Toerless Eckert, Max Pritikin, Kent Watsen, and Michael Richardson, "A Voucher Artifact for Bootstrapping Protocols," RFC 8366, May 2018.

[4] Toerless Eckert and Michael Behringer, "Using an Autonomic Control Plane for Stable Connectivity of Network Operations, Administration, and Maintenance (OAM)," RFC 8368, May 2018.

[5] Carsten Bormann, Brian Carpenter, and Bing Liu, "GeneRic Autonomic Signaling Protocol (GRASP)," RFC 8990, May 2021.

[6] Brian Carpenter, Bing Liu, Wendong Wang, and Xiangyang Gong, "GeneRic Autonomic Signaling Protocol Application Program Interface (GRASP API)," RFC 8991, May 2021.

[7] Sheng Jiang, Zongpeng Du, Brian Carpenter, and Qiong Sun, "Autonomic IPv6 Edge Prefix Management in Large-Scale Networks," RFC 8992, May 2021.

[8] Michael H. Behringer, Brian Carpenter, Toerless Eckert, Laurent Ciavaglia, and Jéferson Campos Nobre, "A Reference Model for Autonomic Networking," RFC 8993, May 2021.

[9] Toerless Eckert, Michael H. Behringer, and Steinthor Bjarnason, "An Autonomic Control Plane (ACP)," RFC 8994, May 2021.

[10] Max Pritikin, Michael Richardson, Toerless Eckert, Michael H. Behringer, and Kent Watsen, "Bootstrapping Remote Secure Key Infrastructure (BRSKI)," RFC 8995, May 2021.

[11] Brian Carpenter, "Autonomic Networking," *IETF Journal*, 2014, **https://www.ietfjournal.org/autonomic-networking/**

[12] FCC, "June 15, 2020 T-Mobile Network Outage Report, A Report of the Public Safety and Homeland Security Bureau Federal Communications Commission" PS Docket No. 20-183, October 2020.
https://docs.fcc.gov/public/attachments/DOC-367699A1.docx

[13] Alex Clemm, Max Pritikin, Michael H. Behringer, and Steinthor Bjarnason, "A Framework for Autonomic Networking," Internet Draft, work in progress, October 2013.
https://datatracker.ietf.org/doc/html/draft-behringer-autonomic-network-framework-01

and

Max Pritikin, Michael H. Behringer, and Steinthor Bjarnason, "Bootstrapping Trust on a Homenet," Internet Draft, work in progress, February 2014.
https://datatracker.ietf.org/doc/html/draft-behringer-homenet-trust-bootstrap-02

[14] Arthur C. Clark, "Hazards of Prophecy: The Failure of Imagination," in *Profiles of the Future* (1962).

[15] Arthur C. Clark, *Profiles of the Future* (revised edition, 1973).

MICHAEL BEHRINGER worked for 18 years at Cisco Systems, where starting in 2010 he led the Autonomic Networking project. Since 2017 he has been working as an independent consultant. E-mail: **michael.h.behringer@gmail.com**

CARSTEN BORMANN likes bringing the Internet to odd places. Honorary professor for Internet Technology at the Universität Bremen, his research interests are in protocol design and networking system architectures. He is behind many of the IETF's *Internet of Things* efforts, including *Constrained RESTful Environments* (CoRE) and the *Constrained Application Protocol* (CoAP), the *Concise Binary Object Representation* (CBOR), and the *Concise Data Definition Language* (CDDL). He co-chairs the Thing-to-Thing Research Group (T2TRG) in the *Internet Research Task Force* (IRTF). He has authored and co-authored 43 Internet RFCs. E-Mail: **cabo@tzi.org**

BRIAN E. CARPENTER, M.Sc., Ph.D., is an Honorary Professor at the University of Auckland. Previously he worked in networking at IBM and CERN. He has chaired the IETF, the Board of the Internet Society, and the Internet Architecture Board. E-mail: **brian.e.carpenter@gmail.com**

TOERLESS ECKERT is co-chair and liaison contact of the *Autonomic Networking Integrated Model and Approach* (ANIMA) Working Group of the IETF. He worked for 18 years in the Cisco IOS development group at Cisco Systems UK and USA, where he started to develop the Autonomic Networking architecture. Since 2016, he is a Distinguished Engineer at Futurewei USA. Email: **tte@cs.fau.de**

SHENG JIANG, Ph.D., is a Distinguished Engineer of the Network Technology Laboratory, Huawei Technologies. He co-chairs the *Autonomic Networking Integrated Model and Approach* (ANIMA) Working Group of the IETF and has authored 28 RFCs. E-mail: **jiangsheng@huawei.com**

YIZHOU LI, M.Sc., is a Principal Engineer in Network Technology Lab, Huawei Technologies. Previously she worked in networking at Singapore Telecom. She is the Secretary of the *Network Virtualization Overlays* (NVO3) Working Group of the IETF. E-mail: **liyizhou@huawei.com**

JÉFERSON CAMPOS NOBRE, M.Sc., Ph.D., is assistant professor at the Institute of Informatics, *Federal University of Rio Grande do Sul* (UFRGS), Brazil. He is co-chair of the IETF-LAC Task Force from the *Latin American and Caribbean Network Operators Group* (LACNOG) and co-secretary of the *Network Management Research Group* (NMRG) of the *Internet Research Task Force* (IRTF). He has been involved in Autonomic Networking research since 2007. E-mail: **jcnobre@inf.ufrgs.br**

MICHAEL C. RICHARDSON, B.Sc. Physics/Computer Science, is an open source and open standards consultant. An autodidact, he wrote mail transfer agents as a teenager, and in the 1990s, after failing at high-energy physics, found his calling designing and building embedded networking products in the security sector. E-mail: **mcr@sandelman.ca**

A Survey on Securing Inter-Domain Routing

Part 1 – BGP: Design, Threats, and Security Requirements

by Geoff Huston, APNIC

The *Border Gateway Protocol* (BGP) is the inter-domain routing protocol on the Internet, and after some 30 years of operation, BGP is now one of its more venerable core protocols. One of the major ongoing concerns related to BGP is its lack of effective security measures, and as a result the routing infrastructure of the Internet continues to be vulnerable to various forms of attack.

In Part 1 of this study, we will look at the design of BGP, the threat model, and the requirements from a security framework for BGP. In Part 2 we will look at the various proposals to add security to the routing environment and also evaluate the current state of the effort in the *Internet Engineering Task Force* (IETF) to provide a standard specification of the elements of a secure BGP framework.

Introduction

The Internet is a decentralised collection of interconnected component networks (*Autonomous Systems*). These networks are composed of *end hosts* (who originate and/or receive IP packets, and are identified by IP addresses) and active forwarding elements (routers) whose role is to direct IP packets as they pass through the network. The routing system is responsible for propagating the relative location of IP addresses to each routing element, so that routers can make consistent and optimal routing decisions in order to pass a packet from its source to its destination. Routing protocols are used to perform this information propagation.

The routing system of the Internet is divided into a two-level hierarchy. One level is *intra-domain* routing, which the set of autonomous routing systems operating within each component network use. The other level is a single *inter-domain* routing system that maintains the inter-network connectivity information that straddles these component networks. A single inter-domain routing protocol, BGP[1] has provided inter-domain routing services for the disparate component networks on the Internet since the late 1980s.[2] Given the central role of routing in the operation of the Internet, BGP is one of the critical protocols that provide essential coherence to the Internet.

The underlying distributed distance vector computations of BGP rely heavily on informal trust models associated with information propagation to produce reliable and correct results. You could liken them to a hearsay network—information is flooded across a network as a series of point-to-point exchanges, with the information being incrementally modified each time it is exchanged between BGP speakers. The design of BGP was undertaken in the relatively homogeneous and mutually trusting environment of the early Internet.

Consequently, its approach to information exchange was not designed primarily for robustness in the face of various forms of negotiated trust or overt hostility on the part of some routing actors.

Hostile actors are a fact of life in today's Internet. It's quite reasonable to characterise today's Internet environment as one where trust must be explicitly negotiated rather than assumed by default. This environment is no longer consistent with the inter-domain trust framework that BGP originally assumed. The BGP mutual trust model involves no explicit presentation of credentials, no propagation of instruments of authority, nor any reliable means of verifying the authenticity of the information being propagated through the routing system. Hostile actors can attack the network by exploiting this trust model in inter-domain routing to their own ends.

An attacker can easily transform routing information in ways that are extremely difficult for any third party to detect. For example, false routing information may be injected, valid routing information removed, or information altered to cause traffic redirection.[3,4,5] You can use this approach to prevent the correct operation of applications, to conduct fraudulent activities, and to disrupt the operation of part (or even all) of the network in various ways. The consequences range from relatively inconsequential (minor degradation of application performance due to sub-optimal forwarding paths) through to catastrophic (major disruption to connectivity and comprehensive loss of any form of cohesive Internet). To resist this subversion of integrity of routing information, each BGP speaker must have:

- Sufficient information at hand to verify the authenticity and completeness of the information being provided to it via the inter-domain routing system, and
- The ability to generate authoritative information such that other BGP speakers may verify the authenticity of information that this speaker is passing into the inter-domain routing system.

A key question is whether we can add further information into the inter-domain routing environment such that attempts to pervert, remove, or withhold routing information may be readily and reliably detected. Any proposed scheme must also be evaluated for its impact on the scaling properties of BGP.

To ground any such evaluation of BGP, it's useful to briefly review the design of the BGP protocol.

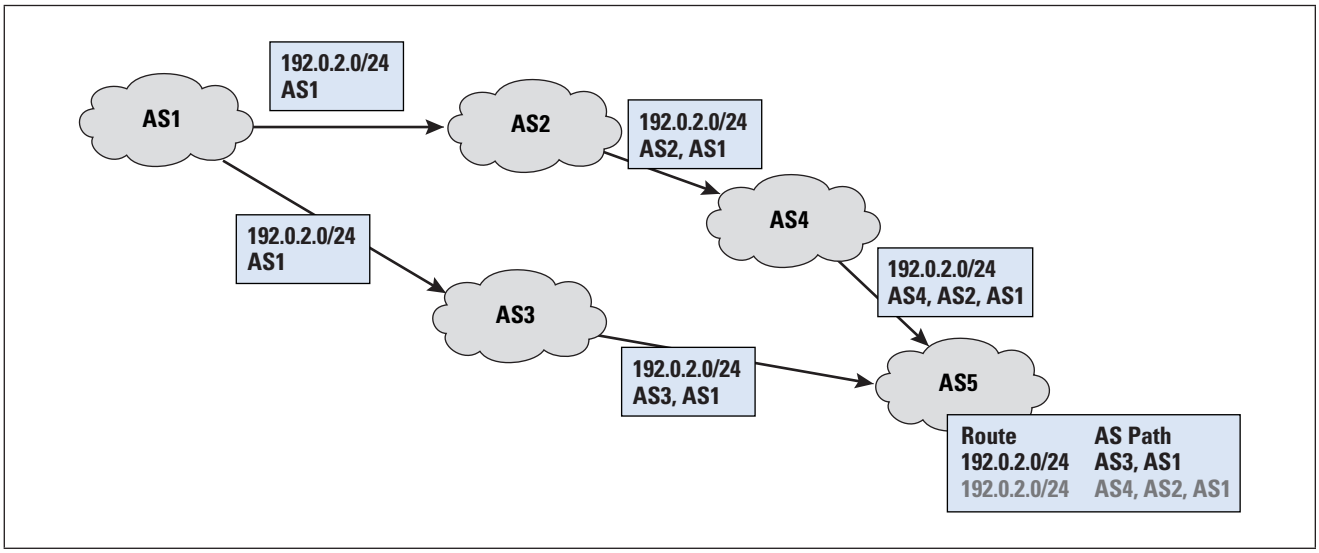
The Design of BGP

BGP underwent numerous refinements over its early operational life. The protocol was originally described in RFC 1105 in June 1989,[6] allowing the inter-domain architecture of the Internet to move on from a constrained architecture of *core* and attached *stub* domains into a framework of peer routing domains without any central core.

A refinement to this protocol, BGP-2, was described in RFC 1163 in June 1990,[7] and a further refinement, BGP-3, was described in RFC 1267 in October 1991.[8] The current version, BGP-4, was first deployed within the Internet in 1993. The RFC describing this protocol, RFC 1771,[9] was published in March 1995, and was subsequently refined with the publication of RFC 4271 in January 2006.[1] The core protocol has been stable for some years now, although further refinement has been applied through the use of negotiated capabilities undertaken at BGP session startup.

BGP is an instance of what we commonly refer to today as a *Bellman-Ford Distance Vector* routing algorithm.[10,11] This algorithm allows a collection of connected devices (BGP speakers) to each learn the relative topology of the connecting network. Its basic approach is very simple: each BGP speaker tells all its other neighbours about what it has learned if the new learned information alters the local view of the network. This scenario is a lot like a social rumour network, where everyone who hears a new rumour immediately informs all their friends. BGP works in a very similar fashion: each time a neighbour informs a BGP speaker about reachability to an IP address prefix, the BGP speaker compares this new reachability information against its stored knowledge that it gained from previous announcements from other neighbours. If this new information provides a "better" path to the prefix, then the local speaker moves this prefix and associated next-hop forwarding decision to the local forwarding table and informs all its immediate neighbours of a new path to a prefix, implicitly citing itself as the next hop. BGP keeps track of the propagation of route advertisements across the inter-domain space by recording the sequence of network *Autonomous Systems* (ASs) that propagate the route in a route attribute called the *AS Path*. A "better" route is one with a shorter AS path, and a loop is detected when a BGP speaker sees its own AS in the received AS Path (Figure 1).

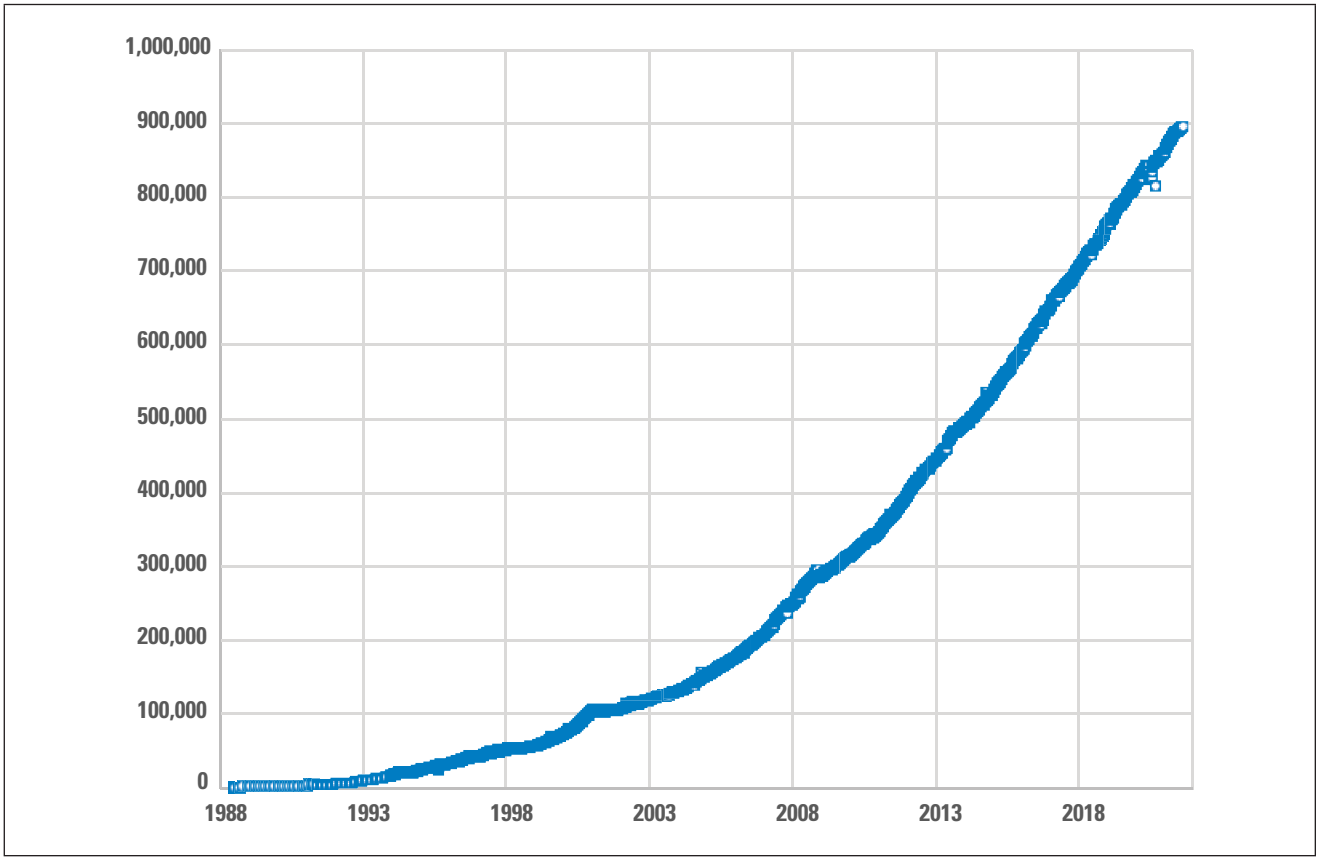
Figure 1: The Propagation of a route in BGP



In addition, there is a withdrawal mechanism, where a BGP speaker determines that it no longer has a viable path to a given prefix, in which case it announces a *withdrawal* to all its neighbours. When a BGP speaker receives a withdrawal, it stores the withdrawal against this neighbour. If the withdrawn neighbour happened to be the currently preferred next hop for this prefix, then the BGP speaker examines its per-neighbour data sets to determine which stored announcement represents the best path from those that are still extant. If it can find such an alternative path, it copies this path into its local forwarding table and announces this new preferred path to all its BGP neighbours. If there is no such alternative path, it announces a withdrawal to its neighbours, indicating that it no longer can reach this prefix.

Across the deployment lifetime of BGP-4, the IPv4 Internet has grown from an average of 20,000 distinct routing entries in 1993 to almost 1 million routing entries in 2021.[12] Figure 2 shows the growth of the size of the Internet IPv4 routing table over time.

Figure 2: Internet IPv4 Routing Table Size, from [12]



BGP and TCP

BGP is not a link-level topology maintenance protocol. It assumes the existence of a relatively robust IP forwarding environment at the link level between BGP peers. This assumption has allowed BGP to use the *Transmission Control Protocol* (TCP) as a reliable transport protocol to support the transactions of the protocol across a BGP peer session.

TCP manages reliable message delivery and flow control between the BGP peers and allows BGP to operate across end-to-end connections whether they reside on the same subnet or across the Internet. There is no requirement for BGP speakers to be connected on a common media connection, and the choice of TCP allows this flexibility of connectivity by requiring only that a BGP peering session is supported by an IP network.

The TCP stream is divided into messages using BGP-defined markers, where each message is between 19 and 4096 octets long, extensible to 65,535 octets.[11] The use of a reliable transport service implies that BGP itself need not explicitly confirm receipt of protocol messages, removing much of the protocol overhead seen in other routing protocols that sit directly on top of a media-level connection. There are no message identifiers, no message number initiation protocols, no explicit acknowledgement of messages, nor any provision to manage lost, reordered, or duplicated messages. TCP handles all of that. The use of a reliable transport protocol also obviates the need for BGP to periodically refresh the routing state by automatically reflooding the entire routing information set between BGP speakers. After the initial exchange of routing information, a pair of BGP routers exchange only incremental changes to routing information.

BGP Messages

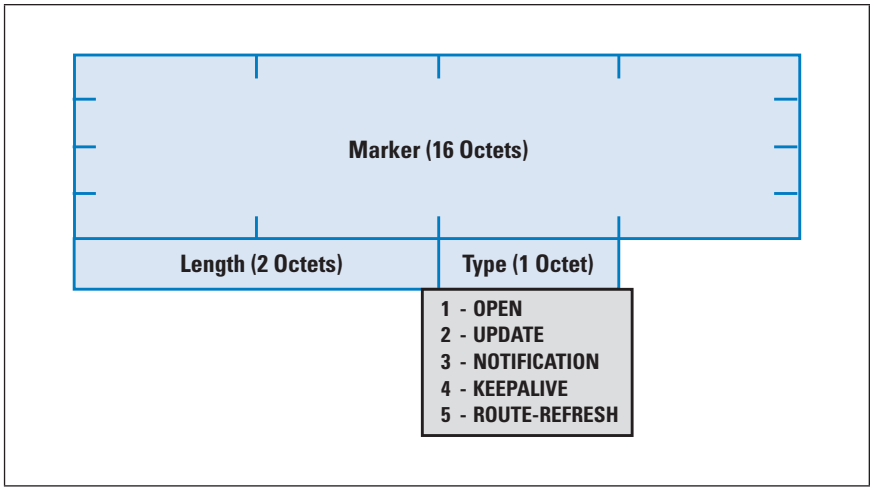
Because TCP is a *stream protocol* rather than a *record-oriented protocol*, BGP uses record marking within the TCP stream to delineate logical protocol units, or *messages* with a 16-byte marker as the BGP message delimiter. A 2-byte length and a 1-byte type field follow the marker, making the minimum BGP message size 19 bytes. The repertoire of defined messages follows:

- An OPEN message to start a BGP session
- An UPDATE message to exchange reachability information
- A NOTIFICATION message, which is used to convey a reason code prior to termination of the BGP session
- A KEEPALIVE message, used to confirm the continued availability of the BGP peer
- A ROUTE-REFRESH request message to request a resend of the routing information.

Figure 3 on the next page shows the common format of BGP messages.

BGP uses an explicit OPEN message to commence a BGP peering session. This message exchange confirms the identity of the BGP speakers and includes the option for a capability negotiation to understand what optional or extended capabilities each BGP speaker supports. A session is active only when both BGP speakers have sent their OPEN messages and neither has rejected the other's offered capabilities through a NOTIFICATION response.

Figure 3: BGP Common Header Message Format



When the session is active, BGP operates via the exchange of UPDATE messages. Each UPDATE message contains a set of address prefixes that are unreachable (withdrawals), followed by a set of common route object attributes and a set of address prefixes that share this set of attributes (announcements). The withdrawn prefixes are those prefixes where the local BGP speaker sees no reachability, and now wants to withdraw a previous advertisement of reachability. No routing attributes are associated with these withdrawn prefixes.

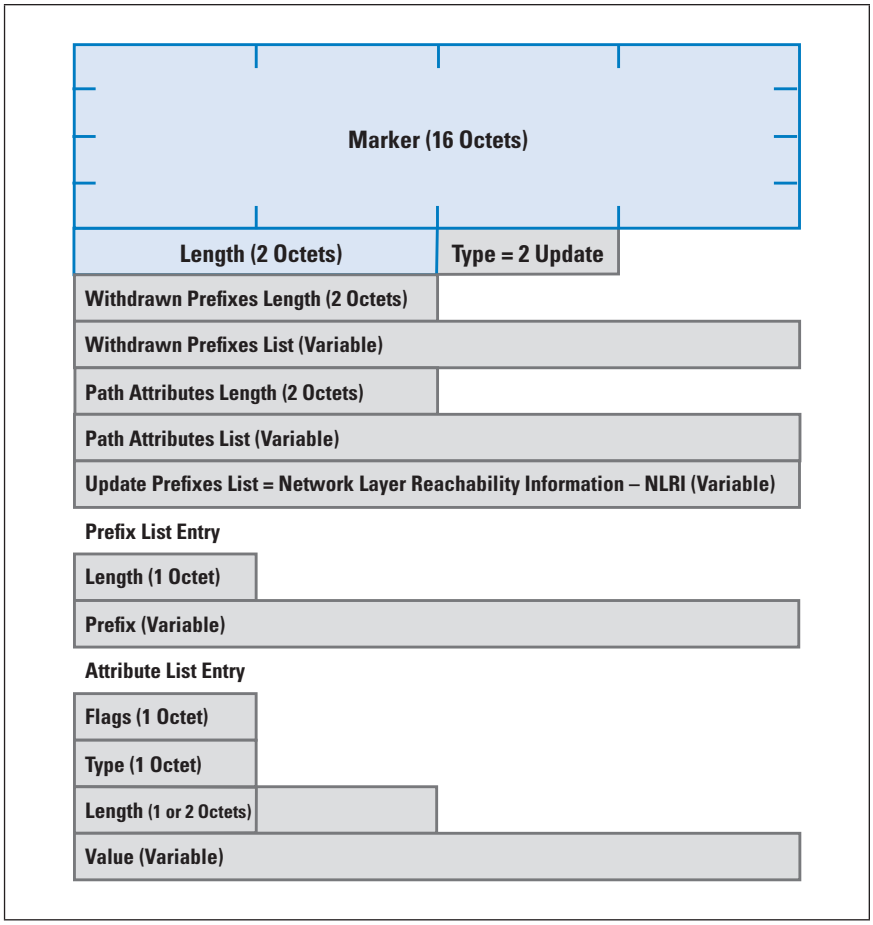
The announced prefixes are those prefixes where the local BGP instance has an updated view of the reachability of a prefix that was previously withdrawn or unannounced or has an updated view of the routing attributes of the locally selected "best" route for a prefix. BGP may group multiple prefixes together in a single UPDATE message but can do so only if all the updated prefixes share a common set of attributes. Within an UPDATE message, the withdrawn prefix set or the announced prefix set may be empty, but not both. Figure 4 on the following page shows the layout of the BGP UPDATE message.

AS Path Attribute

BGP binds together the concept of network address blocks and autonomous systems into a path vector-based routing technology. Every route object represented within a BGP-4 route database contains an address prefix and an associated path vector of AS values. BGP does not indicate the precise path a packet should follow within an AS, nor does it maintain a complete map of the topology of the Internet at a link-by-link level. BGP uses a level of abstraction that views the Internet as a set of per-AS routing domains, and the role of BGP is to maintain a routing map of the network at this AS level, associating every reachable address prefix with an AS transit path from the current location to the originating AS of the address prefix.

One of the most important route object attributes in BGP is the AS Path attribute of UPDATE messages that contain announced routes.

Figure 4: BGP UPDATE Message Format



As address prefix reachability information traverses the Internet in the form of individual route objects in BGP, this BGP routing information is augmented by the list of autonomous systems that have processed this route information thus far, forming the AS Path attribute of a route object. Each BGP speaker adds its own AS value to the AS Path attribute of the route object when passing the route object through an *External BGP* (eBGP) session.

This AS Path attribute allows straightforward suppression of the looping of routing information, using the simple algorithm that a local AS will reject any forwarded route object that already contains its own AS in the AS Path attribute. Also, the length of the AS Path vector forms the BGP route metric. A local BGP system, when attempting to select one from numerous potential route objects that refer to the same address prefix, will, in the absence of any local policy directive, prefer the route object with the shortest AS Path length.

In addition to undertaking the role of path metric and loop detector, the AS Path attribute serves as a versatile mechanism for policy-based routing, where a local AS can alter the default preferences for route selection based on local policy settings coupled with pattern-matching rules to be performed on the AS Path.

Withdrawals have no associated AS Path.

BGP Route Selection Process and Routing Policies

A BGP speaker may receive two or more announcements for the same address prefix from different peers. The "best" announcement is selected as the locally used announcement, and this announcement is the one that is announced to its BGP peers. BGP defines an ordered sequence of comparisons to determine which route object is selected by the local BGP speaker as the preferred route to use:

- Prefer the route object with the highest value for LOCALPREF attribute value.
- Prefer the route object with the shortest AS PATH attribute length.
- Prefer the lowest origin value.
- Prefer the lowest MULTI EXIT DISCRIMINATOR attribute value.
- Prefer the minimum *Interior Gateway Protocol* (IGP) cost to the NEXT HOP address given in the route object.
- Prefer eBGP over *Interior BGP* (iBGP)-learned routes.
- If using iBGP, prefer the lowest *BGP Identifier* value.

Although network administrators usually employ routing policies depending on their needs,[14,15] within the generic BGP route selection process the highest-priority selection rule is that a route for a more specific address prefix is to be preferred over that of a covering prefix.

The BGP Threat Model

One approach to providing a taxonomy for threats in routing in general, and BGP in particular, is to view a BGP peer session as a conversation between two BGP speakers and pose numerous questions relating to this conversation. These questions include:

- *How do we talk?* The manner in which the BGP session between the BGP speakers is secured such that the conversation is not altered, disrupted, or hijacked and is protected from unauthorised eavesdropping
- *Whom am I talking to?* Verification of the identity of the other party and verification that they are authorised to speak for the routing entity that they purport to represent.
- *What are you saying?* Verification of the authenticity and completeness of the routing information being passed in the BGP session.
- *Why should I believe you?* Verification that the routing information represents the current state of the forwarding system.
- *How recent is your information, and is it still valid?* Verification of how long routing information is valid and whether the information is still current.

You can further deconstruct each of these security questions to a set of specific objectives, as well as recognise a set of specific threats.

Securing a BGP Session

A BGP session between two BGP speakers is assumed to have some level of integrity at the session transport level.

BGP assumes that the messages one party sends are precisely the same messages the other party receives, and that the messages have not been altered or reordered, have not had spurious messages added into the stream, or have messages removed from the conversation stream in any way, and given that BGP uses a TCP transport session, some of these assumptions are reasonable but others less so.

As with any long-held TCP session, a BGP peer session is vulnerable to eavesdropping, spurious session reset, session capture, message alternation, and *Denial-of-Service* (DoS) attacks, all through what we might think of as conventional TCP attack vectors.

The threat at the BGP level is that a third party may attempt to break into the TCP session as an interception attack in the middle, and thereby alter the BGP message flow between the two end points. One form of threat is by injection, where the attacker injects spurious messages into the BGP session. Direct on-the-wire interception allows the attacker to have knowledge of the TCP sequence numbers, thereby making injection a trivial task. Even if the attacker is not able to intercept or eavesdrop the BGP session, it is still possible to attempt to guess the current sequence number.

While this guessing is often impractical in the case of injecting data into the session, if all that is to be injected is a TCP Reset, then the sequence number guess only has to sit within the current TCP window in order to be recognised as a valid reset TCP message.[16] Another form of threat is by active intermediation, where the attacker sits on the connection between the two BGP speakers and intercepts all traffic in both directions. In this case, the attacker has complete control of the BGP message stream and can perform any form of message alteration. A variation of this form of threat is *session hijacking*, where the third party intrudes upon an active BGP session and injects its own traffic into the message stream—and that traffic allows the third party to take over the session and masquerade as one of the parties to the BGP session. Because timing is important in the overall performance of BGP, another form of attack at the session level is to delay messages. Although the content of the messages is unaltered, the implicit timing signals within the message stream are altered by this form of intervention, potentially causing the local BGP speaker to behave differently and fall out of sync with its routing peers.

Another form of attack is a *replay attack*, where older BGP messages are replayed into a hijacked TCP session. One form of this replay attack could be to replay a pair of messages that withdraw and then announce the same address prefix.

Route Flap Damping (RFD)[17,18] is a widespread defensive BGP configuration that monitors the frequency of BGP updates for a given prefix from each peer, and if the update rate exceeds a locally set threshold, the advertisement of this prefix by the peer will be locally suppressed for a damping interval. The replay of updates could be used to trigger an RFD response in the remote BGP speaker.[19] If a route is fully dampened through RFD, the BGP speaker will not advertise updates for this prefix for a damping interval (commonly 60 minutes), possibly causing a route disruption within that time frame. Another form of replay attack is to replay a route advertisement for a previously withdrawn prefix, possibly in conjunction with some form of prefix hijack attack.

Another form of threat is withholding traffic. BGP uses keepalive timers to determine remote end "liveness." By intercepting and withholding all messages for the hold-down timer interval, a third party can force the BGP session to be terminated and reset. This action causes the entire route set to be re-advertised upon session resumption so that repeated attacks of this form can be an effective form of denial of service for BGP.

It is also possible to undertake a *saturation attack* on a BGP speaker by sending it a rapid stream of invalid TCP packets. In this case, the processing capability of the BGP speaker is put under pressure, and the objective of the attack is to overwhelm the BGP speaker and cause the BGP session to fail and be reset. This type of attack is particularly problematic if the BGP session uses *Message Digest 5* (MD5) or *Internet Protocol Security* (IPsec) as session protection protocols, because the cryptographic function overhead also applies to the injected packets, increasing the processing overhead on these spurious injected packets.

The underlying aspect of the BGP protocol is that BGP itself has no enforced minimum level of message protection. BGP messages are, by default, placed into the TCP stream without encryption or additional message wrapping of message sequencing. Any threat that is applicable to long-held TCP sessions applies to this default mode of BGP operation.

Verifying BGP Identity

BGP sessions commence by passing the local AS to the remote end of the session in the BGP OPEN message and receiving the AS of the remote end in the received OPEN message. BGP itself does not verify these asserted AS identities, and it is theoretically possible for a remote party to masquerade itself as another AS and assert an identity in BGP that the other party cannot directly verify by, and neither can any third party that subsequently receives this routing information. Most BGP implementations provide a level of protection against this threat by applying a constraint that the local BGP speaker will initiate a peer session only with a configured remote IP address, and reject all other TCP connection attempts.

Furthermore BGP will not complete the BGP OPEN message exchange if the AS in the OPEN message does not match the AS number associated with the remote end IP address in the configuration.

This approach places a heavy reliance on the out-of-band process of BGP configuration, and if an attacker can compromise or take control over BGP equipment connected to the Internet or use social engineering to convince a network administrator to configure incorrect information into the BGP configuration, then it is possible to masquerade as a different party in BGP and potentially inject incorrect information into the routing system.

The real question here is: "Are you really who you claim to be?" Here it is necessary for the BGP speaker to be able to confirm the validity of the peer claim that it is speaking for an AS.

Verifying BGP Information

The objective here is to verify the authenticity and completeness of the routing information being passed in the BGP session. The intention of BGP is that a local BGP speaker provides to all its BGP peers a complete feed of its locally selected route objects.

When a session is opened with a remote BGP speaker, the local BGP instance believes everything it is told without further qualification. The threat is that a BGP peer can deliberately feed false information to the local BGP instance, which BGP itself will be unable to detect as false. The false information could be in the form of suppression of routing information, or alteration of the route object that is being passed, or the invention of spurious route objects. The BGP speaker could be asserting that an AS Path is genuine when it reflects an artificial path, or that it has the authority to originate an advertisement for a prefix when, in fact, no such authority exists.

A BGP speaker may preserve all the attributes of a route object, but alter the prefix set to be the equivalent collection of more specific prefixes. The deliberate alteration of routing information can cause the local BGP instance to make an incorrect choice of a local best path and also cause the local BGP instance to propagate this incorrect information to its neighbours.

Not only could the BGP speaker be passing incorrect attributes for an address prefix in order to bias the local route selection process, but it also could be providing incorrect information regarding the prefix itself. The prefix that is the subject of the route object could be a prefix that has never been allocated and should not be legitimately routed, or the prefix could be an aggregate address prefix that spans both allocated and unallocated address space.

Prefix hijacking is a major threat to the integrity of the BGP routing. The fundamental weakness here is that BGP provides no explicit means of verifying the authenticity of the address prefixes that are listed in a BGP UPDATE message, nor the authenticity of the attributes of the prefix, including the origination information and the AS Path vector. The threat here is that by deliberately altering this information, the local BGP speaker can be induced to make incorrect route selection decisions and thereby make incorrect forwarding decisions for IP traffic.

A known common problem illustrative of exploiting this vulnerability is operational misconfiguration,[20] which could result in propagating more specific routes and other forms of route leakage, or withholding that may affect the routing decisions made by other BGP speakers. This form of verification of intentionality by a remote BGP speaker is far more challenging—while these forms of security mechanisms are intended to verify that the received information matches the original information that was passed into the routing system, they are incapable of verifying that such information is consistent with the true intent of the originator of the information.

Verifying Forwarding Paths

The overall intention of the BGP protocol is to distribute the current binding of address to location such that individual routers can make accurate judgements about how to populate their local forwarding tables and hence make optimal local decisions for each packet that passes along the shortest path to its ultimate destination.

BGP does not provide any ability for a local BGP speaker to validate that the route advertisements it receives from a BGP peer accurately represent the current state of the network forwarding system. The threat model here is that a bad actor in the routing system may make a different forwarding decision to that being advertised in the routing system.

This situation can represent a subversion of local policies, theft of carriage capacity, deliberate denial of service, or the potential to eavesdrop on a conversation or support the interception and alteration of application-level transactions. Even a completely secured control plane does not avert such vulnerabilities.[21]

The Consequences of Attacks on the Routing System

The ability to alter the routing system provides a broad array of potential consequences.[3] The consequences fall into numerous broad categories, which are briefly described here:

1. *The ability to eavesdrop*. The forwarding system can be altered so as to pass all traffic to a class of destination addresses through a certain path. This change allows the attacker to attempt to pass all such traffic through an eavesdropping location prior to conventional delivery. In such a case the parties may not be aware that an eavesdropping attack is taking place.

2. *Denial of service*. The simplest form of a DoS is where traffic to an address prefix is passed to a point where it is then discarded. Routing loops also are a form of DoS, where not only will the traffic to a destination address prefix never reach its intended destination, but the traffic will be held in the loop for the life of the packet *Time to Live* (TTL) field. For sufficiently short loops the potential exists for the loop to act as a link load amplifier, where the traffic on the loop is several times the traffic load being addressed to the affected destination address prefix.

3. *The potential to masquerade*. Subversion of routing allows sites to masquerade as other sites; the routing system misdirects the traffic to the masquerading site. The consequences of such an attack can vary from the specific, where a particular site is targeted, to the more generic, where authoritative *Domain Name System* (DNS) servers are the subject of the masquerading attack, and the DNS responses are believed to be authentic. In this case if the masquerading occurs at the root level of the DNS hierarchy, incorrect information can be provided to any query, allowing for the attack to then be extended to any site.

4. *The ability to steal addresses and obscure identity*. Routing an unallocated address is subtly different from routing an already allocated address. Here the consequence is not displacement of traffic forwarding to incorrect locations in the network, but the assertion of the existence of addresses and forwarding paths to those addresses that should not exist in the network in the first place. The consequence is the ability to use addresses on the network that have no allocation registration information associated with them, allowing the originator of the routing attack some degree of ease to mount an anonymous attack at the application level. Such forms of attack have been observed to be associated with SPAM and botnet controllers where anonymity of the attack coordinator is desired.

Security Requirements

The primary requirements for securing BGP are securing both the transmission of the data payload of the BGP protocol and the semantics of that payload.

The security requirements for transmission are such that the data that a BGP speaker receives can be cryptographically verified to have been sent by the BGP peer, the data is not a replay of previously transmitted data, and no data has been removed from the transmission.[22]

There is no strict requirement for encryption of the BGP payload, because the routing information being exchanged is not intrinsically confidential to the two parties involved. The security requirements for the semantics of the payload concern specifically some selected fields (transitive attributes) of the BGP UPDATE message. The BGP speaker must be able to verify that the advertised prefix is valid, and that the originating AS has been duly authorised by the legitimate right-of-use holder for that prefix.

The BGP speaker should also be able to validate that the AS Path in the UPDATE represents a valid inter-AS transit path through the network in terms of inter-AS topology and AS transit policies, and that the prefix reachability information has been propagated along the reverse inter-AS Path.[22]

It is noted that route withdrawals and nontransitive announcement attributes are local, and thus do not need to be transitively protected in a similar fashion to route origination and the AS Path attribute of announcements. You can adequately protect withdrawals and local attributes with BGP peer session protection.

The associated requirements for a secure inter-domain routing system include that the additional use of security credentials and verification of routing information should not alter the temporal properties of the BGP protocol, and that authentication of the security credentials should occur in the same time frame as the BGP message processing operation. It is also a requirement that piecemeal incremental deployment should be feasible.[23,24,25] A secure operational mode should be a capability negotiation with each BGP peer, with the ability to support backward compatibility with those BGP peers that do not recognise such a capability. It seems to be a good idea to start deployment of BGP security on the most-connected nodes and incrementally deploy it towards least-connected nodes.

Additionally, it suggests the question: How does a party that uses security credentials deal with information arriving from a peer that does not use any security credentials? Having no security credentials does not necessarily mean that the information is wrong, of course. But importantly, in these piecemeal deployment scenarios there should be some incremental benefit of piecemeal deployment to those actors who choose to supply such security credentials and those who choose to validate routing information using these credentials.

A routing system, secure or otherwise, should never make route selections that include routing loops. It is preferred that in a fully secured environment a secure routing system would be able to converge on best paths that are either identical to or no worse than an unsecured BGP speaker would select, assuming that such paths can be validated in a secure environment. In an environment of partial adoption of secure routing systems, it is recognised that a BGP speaker may use local preference settings that prefer sub-optimal paths that have preferred security credentials over unsecured paths.

The trust model of routing appears to involve two forms of trust. The first is a trust environment related to the public network and the legitimacy of use of a public address and a public AS number. It is necessary to be able to verify that a particular party has the right to use these number resources in a public context. The closest fit in the form of a trust model for verification of this assertion of right of use is a public authority that can provide authoritative information on the distribution of these numbers.

This approach leads to a rooted hierarchy model of trust, where the trust anchor is this public authority.

The second form is a trust environment in private contexts, where the use of an address or AS number is bounded by a specific context of use, and the trust in an assertion of a right of use is one made in the context of this bounded environment. In this environment, there is no clear ability to use public authorities as a trust anchor, and other means of trust that may involve reputation, or web of trust concepts may be appropriate.

A general security approach to BGP should be able to encompass that diversity of deployment environments and the corresponding diversity of authority models.

Tools for Securing BGP

The vulnerabilities of BGP arise from four fundamental weaknesses in the BGP and inter-domain routing environment, including:

- No mechanism to protect the integrity, currency, and source authenticity of BGP messages
- No mechanism to verify the authenticity of an address prefix and an AS origination of this prefix in the routing system
- No mechanism to verify the authenticity of the attributes of a BGP UPDATE message
- No mechanism to verify that the local cache *Routing Information Base* (RIB) information is consistent with the current state of the forwarding table

The other observation about BGP security is that it appears that by far the most straightforward form of attack is to obtain control and configuration access to a deployed router and use this compromised platform as the base for launching attacks on the routing system. In the face of such an encompassing attack on the control instruments of the routing system, BGP session-level security needs to be placed in some perspective. It is not possible to prevent routers from attempting to generate false information as long as routers themselves are in a position to be compromised.

The consequent vulnerability on the routing system, as distinct from a narrower view of BGP, is that there is no mechanism that limits the extent to which a misbehaving routing element can make inaccurate claims about reachability in the routing system.

The Security Toolset for BGP Session Protection

The available tools for securing BGP start at the level of the BGP TCP session and encompass the tools that are used to protect TCP and the two ends of the TCP session.

The TCP protection mechanisms include the generalized TTL security mechanism,[26,27] which is intended to limit the effective radius of potential attack on the session to hosts that lie on or within the worst-case hop-count radius between the two BGP speakers and host-level defences against TCP SYN attacks.[28] In many ways, this form of defence is effective when using multi-hop BGP sessions in that the attacker cannot subvert the defence, but it still leaves the session vulnerable to any attacker that lies within the TTL radius.

You can get greater levels of session protection by using cryptographic protection. Over time the IETF has worked on three approaches to protect the BGP TCP through cryptographic protection. They include:

- The use of IPsec.[29] IPsec has not been widely used for BGP sessions, and the reasons why relate to the complications for rekeying *Internet Key Exchange* (IKE/IPsec) sessions and the potential *Distributed Denial-of-Service* (DDoS) vector.[30]
- The *TCP MD5 Signature Option*.[31] Although the MD5 signature option has some potential weaknesses when compared with IPsec,[29] MD5 is considered preferable to no form of TCP protection at all, particularly with respect to the TCP Reset injection attack. However, there are issues with re-keying a long-held session, and the BGP speakers probably need to use graceful restart mechanisms in conjunction with MD5 to perform a re-key of the session.
- The *TCP Authentication Option*,[32] which the IETF has marked as a replacement for the earlier MD5 approach. The TCP Authentication Option supports stronger crypto algorithms compared to MD5. It uses a two-fold security approach that reduces the critical reliance on a user-configured key. This approach also allows the configuration of up to 64 keys for a session and provides a simple key coordination mechanism by giving the ability to change keys (move from one key to another) within the same connection without causing any TCP connection closure. By comparison, changing TCP MD5 keys during an established connection might cause a flap or restart in the connection, which in the context of BGP may have operational implications.

From time to time the topic of BGP over *Transport Layer Security* (TLS)[33] is raised, and it is possible that sooner or later we might hear of BGP over *Quick UDP Internet Connections* (QUIC).[34,43] The salient question is one of balancing the additional burden of adding more transport choices to BGP implementations with the likely benefits that these additional choices may provide. As we've seen in the IPv6 transition and more recently in the increasing diversity of choices for encrypting DNS transactions, adding more options can offer just confusion and impede adoption instead of accelerating it.

However, the most important guideline in securing BGP sessions is to use multi-hop BGP and multi-access LAN sessions sparingly and preferably use a direct 1:1 channel connection when such a choice is available.

The Security Toolset for BGP Message Protection: RPKI

In addition to message integrity protection that transparent session-level protection mechanisms provide, the tools to provide protection of the integrity of BGP messages relate to the use of digital signatures to provide a set of credentials that allow relying parties to verify the correctness of the information carried as the message payload in BGP.

The reason for the use of digital signatures as opposed to an integrity check using some form of shared secret was obvious after the observation that the number and identities of all eventual recipients of the information are not known in advance, and non-repudiation is desirable.[3] Verification of the contents of a message is not only a test of whether the message has been altered in any way during its transit between BGP speakers, but also a test of whether the message represents correct origination information and correct operation of the processing of the message during the message propagation (*authenticity*).

This requisite implies a need to establish a means of verification of information where the author of any security credentials relating to origination and propagation is not necessarily known to the relying party that is attempting to validate the information. This need typically invokes a form of validation that relies upon third-party transitive trust, where the relying party is attempting to build a testable chain of trust between its trust anchor and the party or action that is the subject of the verification operation. Conventionally, this requirement implies the use of some form of *Public Key Infrastructure* (PKI). In this case, we are not looking to use such a PKI to validate claims of identity, authority to perform a particular function, or some form of verifiable attribution. We need some form of mechanism to associate a public key with an IP address prefix or an AS number in a sense of functional control, where the certification authorities in this PKI are attesting that the certified subject has *functional control* of a collection of IP number resources (AS numbers and IP address prefixes). The associated certificate issuance practices are intended to support transitive trust in such attestations of association.

We have adopted a structure using X.509 public key certificates and a certificate extension that uses a canonical list of IP address resources and AS numbers[35] as the foundation for this *Number Resource PKI* (RPKI).[36] Verification of a digital signature entails a test of the authenticity and current validity of the associated certificate that describes the public key of the address or AS number holder in the context of a structured set of signed relationships between certificate issuers and subjects. In other words, the holder of the matching private key is the current functional controller of those IP addresses and AS numbers and can digitally sign authorities and attestations about such number resources on the basis of that functional control.2

Given that the discourse of BGP messages is about address prefixes and AS numbers, the RPKI provides a solid foundation for digital signatures to be associated with various routing actions that are described in BGP messages.[37] It does not attest in any way to the identity of these number resource holders.

Anchoring the model of authority and trust in the RPKI certificate structure has resulted in a framework where the issued certificates are aligned with the IP address and AS number allocation and assignment framework. If an Internet Registry has issued a set of IP addresses and AS numbers to an entity, then this registry would be able to publish a public key certificate that associates a private key provided by the entity with the IP resource set. Further allocations from a registry to a registry address holder would result in re-issuance of the certificate for the address holder with a larger resource set, while reduction in this set would result in both re-issuance and revocation of the previous certificate. This certificate framework would allow auditing of the certificate state by inspecting the registry contents of the Internet Registries, because the intention of this PKI is to mirror the overall state of the number registries with the set of issued certificates.

The RPKI is different from many other PKIs because the requirements related to adding digital signatures to the routing domain are different from many other PKI deployment environments. The common question that the PKI attempts to answer is: "Is this data authentic?" The data is signed with a digital signature, and the key used to generate that signature is described in a certificate. The validity of that certificate can be ascertained by using a collection of certificates and *Certificate Revocation Lists* (CRLs), such that a relying party may validate the data by using its local trust anchor(s) and constructing a *validation path* of issuer-subject chained certificates from a trust point to the digital signature. If this collection of certificates is bundled with the digital signature and the data itself, then the only data items that need to be distributed outside the data flow are the PKI trust anchors.

Distributing the RPKI Data Collection

When the RPKI is combined with a use case for the routing domain, we are looking at a design space that is somewhat atypical in the PKI world. For example, in the WebPKI the certificates that are passed between server and client in the initial exchange of a *Transport Level Security* (TLS) session are related to the particular domain name used in the TLS session being established.[33] The critical distinction here between the secure client/server transactions using the WebPKI and the promulgation of routing information in the routing system using RPKI is that the routing system continuously presents the *entire* routing domain to each relying party. Each relying party needs to have continual access to the entire RPKI certificate and CRL collection, rather than the TLS practice of processing individual signatures and certificates on an as-needed basis.

This requirement for all participating entities to have access to all the RPKI data at all times poses a design challenge about how to manage this RPKI and use it in a routing protocol such as BGP.

A basic approach here is for each Internet Registry to publish its certificate products in its own publication point. This paradigm is analogous to the pre-*Content Delivery Network* (CDN) model of web content publication, where each element is independently published. Of course, in this case while publication is easy, the onus is shifted to the relying party client, or BGP validator, who has to assemble a local cache of all RPKI signed data. It becomes the task of clients of the RPKI to maintain a local cache of the entire RPKI by continuously sweeping across these publication points looking for, and retrieving, changes, and validating all such signed objects as they are received.

At this point BGP updates could be passed to this local RPKI engine, and the data in the update can be compared against the validated information contained in the local RPKI cache. If RPKI validation was performed at the point of acceptance into the local cache (that is, discarding all RPKI products that cannot be validated within the framework of the RPKI validation procedures), then you could verify the route information against the assembled (and validated) crypto data without a high on-demand crypto processing overhead. An alternative approach is to express the validation outcomes from the local RPKI cache as a filter list. If this list were maintained on a router, then the overheads in passing route objects through such a filter would be little different from the many other routing policy maps used in operational configurations.

The drawback in this distributed approach is the need for these clients to constantly sweep all the RPKI publication points to ensure that their local cache is up-to-date. The meaning of "up-to-date" is relative here, but it is worth remembering that the average time to propagate a BGP update across the global Internet depends on the average AS path length (around 4 to 5 autonymous systems on average at present) and the interaction with the BGP *Minimum Route Advertisement Interval* (MRAI) timers. Whereas the worst case would be 300 seconds (assuming that the full MRAI delay would be applied on each eBGP session), the fastest case is well under a second. So how quickly should the local cache be populated to keep up with the propagation of routing information in BGP? Before leaping to a target time, it is also worth remembering the scaling question. With around 100,000 distinct ASs in the Internet routing system, today's worst-case scenario is some 100,000 RPKI clients performing a sweep across 100,000 distinct RPKI publication points every few seconds (or even more frequently if the RPKI system is intended to be highly responsive).

In some ways, this scenario puts the load on the wrong side of the information distribution process. By making the relatively infrequent publication process one that involves a local action without any associated notification of a change, then the burden is shifted to the client set, that has to poll every publication point continuously just to ascertain if anything has changed. To put it as plainly as possible, this particular information distribution design is completely broken! If the client set is known in advance (such as is the case in the DNS in synchronising the information across primary and secondary authoritative services), you could use notification mechanisms. But in the case of the RPKI system, the publishers of authoritative information have no information as to who the clients who need to be notified of a change in the part of a *Certificate Authority* (CA) of the RPKI data collection even are. Hence, notification is not a viable option in this framework.

You can mitigate these relatively formidable scaling issues by changing the publication behaviour, in a manner analogous to the way in which CDNs have improved web performance by shifting content publication models to various permutations of anycast-related models of content replication. In the context of the RPKI, these permutations could entail the use of a smaller set of RPKI publication points that many RPKI certificate issuers share, or a reduction in the number of independent CAs who each publish their own products through the extensive use of *Registration Agents* (RAs). The information being published is signed, so there is no particular benefit to retrieving the data from any particular publication point. As long as the client can validate the data, the client can be assured that the data it has retrieved is most likely to be genuine, irrespective of the location used for the retrieval. It is possible to use third-party aggregators in such a role; these aggregators would take on the task of continuous monitoring of all RPKI CA publication points and publish an aggregated data set of all current RPKI data. You could take this model further into a *push* model by having clients register their interest in updates from the intermediary and allow the intermediary to send them information updates as they are received from the primary CA publication point sources.

Again, it must be noted that the information is signed, so the potential that the intermediary could alter the RPKI information is limited. The design gap in such mediated distribution approaches is to provide a mechanism for clients of these aggregated intermediaries to be assured that the collection the intermediary has provided is the entire collection of RPKI data, and any credible intermediary approach would need to explicitly address this problem of *information completeness*.

However, although these approaches reduce the load imposed on the RPKI clients by increasing the load on information publication, such aggregated publication models also create critical points of concentration of routing data, and a sustained denial-of-service attack against such aggregate publication points could significantly affect the routing system as the local RPKI caches lose currency and coherency.

These approaches have their own strengths and risks. Highly distributed publication models impose undue costs on clients because the clients need to maintain an aggregate and current data collection in their local cache. Aggregate data-publishing models relieve load from clients but have some unresolved issues in terms of assured completeness of the aggregated data collection; they also run the risk of creating new points of vulnerability in terms of the consequence of DOS attacks launched against these aggregated publication points.

The current RPKI operational framework that is used in the *Route Origination Validation* (ROV) tool[38] uses the approach of an *out-of-band* RPKI *pull* system together with some use of aggregated RPKI publication points. The local cache currency performance level of an RPKI client is phrased in units of minutes rather than seconds, and the overall system operates at a level of coherency that is at a time scale of hours rather than minutes. The initial design of this RPKI distribution system is for each client to operate autonomously and maintain a local cache to keep synchronised with the current state of all the RPKI publication points using the *rsync* protocol[39] together with the concept of a *manifest*.[40] This manifest allows a client to ensure that it has retrieved the entirety of the data available at each RPKI publication point. The rsync protocol was subsequently found to be a poor choice for this role,[41] and these days the *RPKI Repository Delta Protocol* (RRDP) is the preferred RPKI repository synchronisation tool.[42]

In terms of the application of the RPKI to the BGP environment, we should ask an obvious question here: If the intent of the flooding system is to provide a reliable and efficient way to flood current information to all clients, then why not just use BGP itself? BGP is an Internet-wide information flooding protocol using a *push*-based approach that is intended to ensure that all BGP speakers have a consistent and current collection of reachable route objects. If the set of clients that want to maintain an up-to-date synchronised local cache is isomorphic to the set of BGP speakers, then adding a BGP message payload type in the same manner that *Address Family Indicator/ Subsequent Address Family Indicator* (AFI/SAFI) indicators are already used in Multi-Protocol BGP today seems only logical.

Part of the reason why the RPKI has had to re-invent this particular wheel of reliable flooding lies in the strictures imposed on the standardisation effort in the IETF, where the *Secure Inter-Domain Routing* (SIDR) Working Group was constrained from proposing changes to the BGP protocol itself.

In retrospect, this constraint appears to have been a rather suboptimal and, in hindsight, extremely poor piece of guidance from the *Internet Engineering Steering Group* (IESG) at the time.

If we could contemplate changes to BGP, then one approach to the RPKI distribution tasks is to maintain the association of the validation material with the data, and in the context of the routing environment it would staple a collection of certificates (and CRLs) to each route object. In a sense this approach would attempt to reproduce the TLS model in BGP, where each prefix being updated would have a subset of the RPKI certificates stapled to the update that would permit an associated signed attestation to be validated within the framework of the RPKI. This method is not without additional impositions, and it would impose costs on the operation of the BGP protocol and BGP speakers. Stapling crypto credentials to BGP updates would bloat both the volume of stapled data (through the use of long validation chained paths and long-term certificate issuance policies which, in turn, create extended CRL lists) and the amount of crypto processing of these stapled digital credentials. There would be a significant level of the retransmission of certificates on a pair-wise basis in such a system if the protocol were to bundle the entire RPKI validation chain data with every routing protocol update. The validation processing load would also likely be beyond the processing capabilities of most routers, and there are considerations of the maximum message size in the BGP protocol itself (which, until RFC 8654,[13] published in October 2019, was 4,096 octets), which limited the amount of attached data that can be placed into BGP.

None of these issues is intractable, and many proposals have been made to attempt to optimise such additional loads and processing demands. We will look at some of these proposals in Part 2 of this survey.

Coming in Part 2

In Part 2, we will take these various requirements and tools and look at the various proposals that have been published for securing BGP. We will also evaluate the current state of the effort in the IETF to standardise a secure BGP Framework.

References

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

[2] Yakov Rekhter, "Experience with the BGP Protocol," RFC 1266, October 1991.

[3] Sandra Murphy, "BGP Security Vulnerabilities Analysis," RFC 4272, January 2006.

[4] Abbie Barbir, Sandra Murphy, and Yi Yang, "Generic Threats to Routing Protocols," RFC 4593, October 2006.

[5] Hitesh Ballani, Paul Francis, and Xinyang Zhang, "A study of prefix hijacking and interception in the Internet," *ACM SIGCOMM Computer Communications Review*, Volume 37, No. 4, October 2007.

[6] Kirk Lougheed and Yakov Rekhter, "Border Gateway Protocol (BGP)," RFC 1105, June 1989.

[7] Kirk Lougheed and Yakov Rekhter, "Border Gateway Protocol (BGP)," RFC 1163, June 1990.

[8] Kirk Lougheed and Yakov Rekhter, "Border Gateway Protocol 3 (BGP-3)," RFC 1267, October 1991.

[9] Yakov Rekhter and Tony Li, "A Border Gateway Protocol 4 (BGP-4)," RFC 1771, March 1995.

[10] Richard Bellman, "On a routing problem," *Quarterly of Applied Mathematics*, Volume 16, No.1, April 1958.

[11] Lester Randolph Ford, Jr., "Network Flow Theory," RAND Corporation, Paper P-923, August 1956.
https://apps.dtic.mil/sti/pdfs/AD0422842.pdf

[12] Geoff Huston, *The BGP Report*,
https://bgp.potaroo.net

[13] Keyur Patel, David Ward, and Randy Bush, "Extended Message Support for BGP," RFC 8654, October 2019.

[14] Tim Griffin and Geoff Huston, "BGP Wedgies," RFC 4264, November 2005.

[15] Feng Wang and Lixin Gao, "On Inferring and Characterizing Internet Routing Policies," in *IMC '03: Proceedings of the 3rd ACM SIGCOMM Conference on Internet Measurement*, October 2003.

[16] Mitesh Dalal, Randall R. Stewart, and Anantha Ramaiah, "Improving TCP\x27s Robustness to Blind In-Window Attacks," RFC 5961, August 2010.

[17] Curtis Villamizar, Ravi Chandra, and Ramesh Govindan, "BGP Route Flap Damping," RFC 2439, November 1998.

[18] RIPE Routing Working Group Recommendations on Route Flap Damping, January 2013.
https://www.ripe.net/publications/docs/ripe-580

[19] Kotikalapudi Sriram, Doug Montgomery, Oliver Borchert, Okhee Kim, and D. Richard Kuhn, "Study of BGP Peering Session Attacks and Their Impacts on Routing Performance," *IEEE Journal on Selected Areas in Communications*, Volume 24, No. 10, Oct. 2006.

[20] Ratul Mahajan, David Wetherall, and Tom Anderson, "Understanding BGP Misconfiguration," in *SIGCOMM '02: Proceedings of the 2002 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications*, 2002.

[21] Sharon Goldberg, Shai Halevi, Aaron D. Jaggard, Vijay Ramachandran, and Rebecca Wright, "Rationality and Traffic Attraction: Incentives for Honest Path Announcements in BGP," *ACM SIGCOMM Computer Communications Review*, Volume 38, No. 4, August 2008.

[22] Blaine Christian and Tony Tauber, "BGP Security Requirements," Internet Draft, work in progress, November 2008, **draft-ietf-rpsec-bgpsecrec-10**

[23] Xinming He, Christos Papadopoulos, and Pavlin Radoslavov, "A framework for incremental deployment strategies for router-assisted services," in *Proceedings IEEE INFOCOM 2003: Twenty-Second Annual Joint Conference of the IEEE Computer and Communications Societies*, Volume 2, March 2003.

[24] Martin Suchara, Ioannis Avramopoulos, and Jennifer Rexford, "Securing BGP Incrementally," in *CoNEXT '07: Proceedings of the 2007 ACM CoNEXT Conference*, December 2007.

[25] Jennifer Rexford and Joan Feigenbaum, "Incrementally-Deployable Security for Interdomain Routing," in *CATCH '09: Cybersecurity Applications & Technology Conference for Homeland Security*, March 2009.

[26] Vijay Gill, John Heasley, and David Meyer, "The Generalized TTL Security Mechanism (GTSM)," RFC 3682, February 2004.

[27] Vijay Gill, John Heasley, David Meyer, Pekka Savola, and Carlos Pignataro, "The Generalized TTL Security Mechanism (GTSM)," RFC 5082, October 2007.

[28] Wesley M. Eddy, "TCP SYN Flooding Attacks and Common Mitigations," RFC 4987, August 2007.

[29] Karen Seo and Stephen Kent, "Security Architecture for the Internet Protocol," RFC 4301, December 2005.

[30] Brian Weis, "Why IPsec and BGP don't play well together in real networks," Security Area Working Group presentations, IETF 66, July 2006.
https://www.ietf.org/proceedings/66/slides/saag-2.pdf

[31] Andy Heffernan, "Protection of BGP Sessions via the TCP MD5 Signature Option," RFC 2385, August 1998.

[32] Ronald P. Bonica, Allison Mankin, and Joe Touch, "The TCP Authentication Option," RFC 5925, June 2010.

[33] Eric Rescorla, "The Transport Layer Security (TLS) Protocol Version 1.3," RFC 8446, August 2018.

[34] Jana Iyengar and Martin Thomson, "QUIC: A UDP-Based Multiplexed and Secure Transport," RFC 9000, May 2021.

[35] Charles Lynn, Karen Seo, and Stephen Kent, "X.509 Extensions for IP Addresses and AS Identifiers," RFC 3779, June 2004.

[36] Geoff Huston, Robert Loomans, and George Michaelson, "A Profile for X.509 PKIX Resource Certificates," RFC 6487, February 2012.

[37] Matt Lepinski, Andrew Chi, and Stephen Kent, "Signed Object Template for the Resource Public Key Infrastructure (RPKI)," RFC 6488, February 2012.

[38] Geoff Huston and George Michaelson, "Validation of Route Origination Using the Resource Certificate Public Key Infrastructure (PKI) and Route Origin Authorizations (ROAs)," RFC 6483, February 2012.

[39] Andrew Tridgell and Paul Mackerras, "The rsync algorithm," 1996.
https://rsync.samba.org/

[40] Matt Lepinski, Stephen Kent, Geoff Huston, and Rob Austein, "Manifests for the Resource Public Key Infrastructure (RPKI)," RFC 6486, February 2012.

[41] George Michaelson, and Byron Ellacott, "rsync Considered Inefficient and Harmful," IETF 89, March 2014.
https://www.ietf.org/proceedings/89/slides/slides-89-sidr-6.pdf

[42] Rob Austein, Tim Bruijnzeels, Bryan Weber, and Oleg Muravskiy, "The RPKI Repository Delta Protocol (RRDP)," RFC 8182, July 2017.

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

[44] Tom Strickx and Celso Martinho, "Understanding How Facebook Disappeared from the Internet," *The Cloudflare Blog*, October 4, 2021.
https://blog.cloudflare.com/october-2021-facebook-outage/

[45] Mark Handley, "Why the Internet only just works," *BT Technology Journal*, Volume 24, No. 3, July 2006.

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**

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
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
Hidde Beumer
Pier Paolo Biagi
Tyson Blanchard
John Bigrow
Orvar Ari Bjarnason
Axel Boeger
Keith Bogart
Mirko Bonadei
Roberto Bonalumi
Julie Bottorff Photography
Gerry Boudreaux
L de Braal
Kevin Breit
Thomas Bridge
Ilia Bromberg
Václav Brožík
Christophe Brun
Gareth Bryan
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
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
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
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
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
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
Jose Antonio Lazaro Lazaro
Rick van Leeuwen
Simon Leinen
Robert Lewis
Christian Liberale

Martin Lillepuu
Roger Lindholm
Link Light Networks
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
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
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
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
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
Gregory Robinson
Ron Rockrohr
Carlos Rodrigues
Magnus Romedahl
Lex Van Roon
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
Philip Schneck
Peter Schoo
Dan Schrenk
Richard Schultz
Timothy Schwab
Roger Schwartz
SeenThere
Scott Seifel
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
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
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
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
Vista Global Coaching & Consulting
Dario Vitali
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

Afilias

Akamai

APRICOT Asia Pacific Regional Internet Conference on Operational Technologies

COMCAST

DE-CIX

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

amsix amsterdam internet exchange

DNS-OARC

IWL

ISC Internet Systems Consortium

IPXO

Limelight NETWORKS

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