Tarpan: A Router That Supports Evolvability

Harshal Sheth and Andrew Sun*

Westford Academy, Westford, MA

Program for Research in Mathematics, Engineering and Science for High School Students (PRIMES),

Massachusetts Institute of Technology, Cambridge, MA

September 25, 2017

Abstract

Tarpan is an extension to the Border Gateway Protocol that allows the Internet to incrementally transition to a new routing protocol to eventually replace BGP. It allows any number of new routing protocols to disseminate routing data in-band within existing BGP advertisements. Tarpan also creates a simple API framework for new protocols to implement. The Wiser protocol was implemented using this framework, and it was found that Tarpan has a minimal effect on BGP advertisement processing performance. Tarpan shows that transmitting new protocols' control information in-band within BGP advertisements is feasible and can enable Internet evolvability.

*Mentored by Raja Sambasivan, Rafik B. Hariri Institute for Computing and Computational Science and Engineering, Boston University, Boston, MA

1 Introduction

The Internet is a vital part of our lives, allowing anyone, anywhere, to access information and communicate with others. It is comprised of an *inter*connected *net*work of routers that forward network traffic from one router to another. These routers play a role in inter-domain routing; in other words, they route data between regions of the Internet. The control information used to perform this routing correctly is distributed using an inter-domain routing protocol called the Border Gateway Protocol (BGP). By providing routers a means of obtaining information about other routers and routing paths, BGP is a fundamental component of the Internet's routing infrastructure and makes possible the delivery of the world's online traffic.

While BGP does work, it has numerous well-known problems. For example, considering the simplified topology shown in Figure 1, BGP would choose the shortest route: the one hop path from A to B. However, the shortest route contains a nearly saturated network link, which is sub-optimal for performance. In this case, choosing the longer route from A to C to B would be a better choice, even though the path contains more autonomous systems (ASes) to go through.

A failure to reliably choose optimal paths is only one of BGP's many shortfalls. Another issue is that ASes cannot assert a significant degree of influence over what routes other ASes utilize, and thus are unable to control the amount of traffic they receive or which routes they prefer to use. Also, BGP's path dissemination works slowly and can oscillate in the case that a router repeatedly comes online and fails. The security of BGP is also weak, as it is vulnerable to many attacks that could

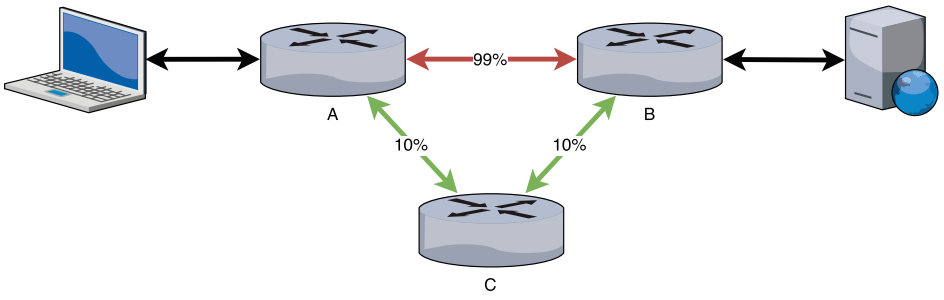


Figure 1: Router topology where BGP would make a poor route choice by choosing a heavily saturated link over a two-hop route with lower utilization.

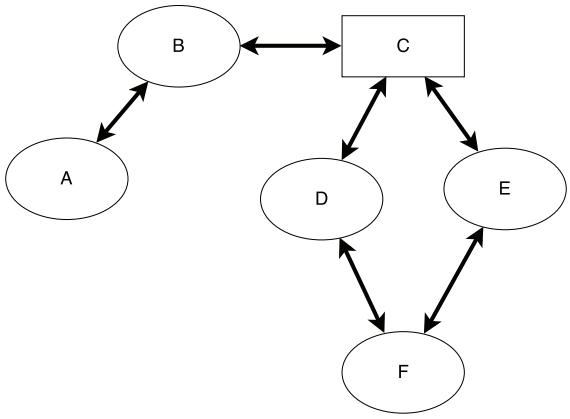


Figure 2: Diagram of a router topology where ellipses are routers that support a new routing protocol, and the rectangle (router C) is a gulf that does not support the new protocol.

compromise data security.

Perhaps the biggest issue with the Border Gateway Protocol is its rigidity: it is difficult to make significant changes or improvements during its operation because it requires directly connected neighbors to use the same protocol versions and features. Even with a protocol that is clearly superior to BGP, its deployment would represent a significant challenge. Because the Internet is composed of many domains and ASes controlled by independent Internet Service Providers (ISPs), each with their incentives and resources, it would not be feasible for all these ISPs to atomically switch to one of these newer protocols. Besides, it is not readily clear which of these newer protocols is superior, and this disagreement would prevent a synchronized switch from occurring. The other more reasonable option is an incremental deployment of such a new protocol. In addition, each router must have the ability to support multiple protocols simultaneously, or it risks developing rigidity similar to that of BGP. As such, there will be instances in which routers with varying support for new protocols must interact.

As shown in Figure 2, this creates a scenario in which groups of routers that support a new protocol, called islands, are separated from other groups of routers that do not support this newer protocol, called gulfs. While most newly developed inter-domain routing protocols function correctly within a single island, crossing gulfs would require additional structures and methods, many of which limit the advantages of that new protocol or require a fallback onto another protocol.

Therefore, the inability to cross gulfs is a major barrier to the rapid adoption of a new protocol. Because an AS would only benefit from adopting a new protocol if its directly connected peers support the same protocol, it dramatically limits the appeal of introducing a new protocol for many ASes, consequently making it difficult for the Internet to evolve.

Previous work has identified two crucial features that would enable Internet evolvability [1]. The first feature is *pass-through support*, which would enable the crossing of gulfs and would make it advantageous for ASes to use newer protocols even if their directly connected neighbors do not reciprocate this support. The second feature is *multi-protocol advertisements*, which would allow ASes to use multiple protocols simultaneously. If this feature were not present, then a new protocol would be complicated to remove and replace, thus having the same problem as BGP.

These two features were then incorporated into D-BGP [1], a modified version of BGPv4 [2], and a proof-of-concept implementation of D-BGP, called Beagle [1], was created and evaluated. However, this implementation was rudimentary and relied solely on out-of-band communication, which meant that a router must contact a separate server to retrieve the routing data of a custom protocol.

A new approach focusing on adding these same evolvability features within BGP using an in-band mechanism is presented in this paper, which involves transmitting the information embedded within BGP advertisements. An in-band version of D-BGP, an implementation of this protocol called Tarpan, and a systematic evaluation to understand its performance characteristics are discussed.

2 Background

2.1 The Border Gateway Protocol

The overarching goal of Internet routing protocols is to create paths between source and destination Autonomous Systems (ASes). ASes are collections of routing prefixes operated by a single ownership entity, usually an Internet Service Provider (ISP). There are thousands of ASes, all of which are interconnected through other ASes. This *inter*connected *net*work of ASes forms the

fundamental infrastructure of the Internet. This network can be viewed as a massive, constantly evolving connected graph. Internet traffic flows across this network to go from its source AS to its destination AS, after which it is delivered to its ultimate destination using other intra-AS routing protocols, like OSPF.

ASes must be able to discover and use paths within the network to deliver traffic to its destination. In one possible approach to routing, each AS would maintain a complete, up-to-date graph of this network and use it to compute optimal routes. In fact, this strategy is employed by the link-state protocols such as Open Shortest Path First (OSPF) and Intermediate System to Intermediate System (IS-IS). Link-state protocols enable all routers within a network to maintain the same corpus of information, ensuring that every router makes the same routing decisions. Usage of such link-state routing protocols is limited to within domains, as the scale of the Internet and independent goals of ISPs requires path vector protocols for inter-domain routing.

Path vector protocols, like BGP, disseminate routing information in a manner similar to word of mouth. When an AS or link between ASes comes online, the AS sends an advertisement of this route to its neighbors, to which it is directly connected. These neighbors then append their own AS number to the path vector within the incoming advertisement, and then send out this path in an advertisement to their respective neighbors, thus spreading the new path information across the network. The list of ASes contained within the path vector field is used to detect loops in the path. After routing information has been disseminated *upstream*, other routers can use it to direct traffic back *downstream*, to the AS that originally broadcast the advertisement. The main advantage of such path vector protocols is that ASes can be independently controlled by ISPs, each of which can selectively avoid exposing certain paths and make their own independent routing decisions if it serves their interests.

The Border Gateway Protocol, a path-vector protocol, is the only protocol in use today. However, it still has a number of problems:

- Each router can decide whether or not to disseminate an advertisement to its peers, which can prevent other routers from discovering certain paths.

- Paths are selected based on the number of "hops" from source to destination, even when potentially better alternatives exist (Figure 1).
- Routers have limited influence over neighboring ASes, and thus can assert little control over the traffic they receive.
- Updates to route information are slow to propagate across the network.
- BGP's security is weak and is vulnerable to attacks.

2.2 Alternative Protocols

In response to the many shortfalls of BGP, a number of alternatives have been proposed. Some of these protocols make incremental improvements over BGP to fix certain critical issues, while others entirely replace BGP.

- **Wiser** is an extension to BGP that adds "cost" information to route advertisements. The cost is an arbitrary value that represents the unfavorability of sending a packet across that link. It could be derived from the monetary cost of transmitting data across a link, from the saturation of that link, or from some other metrics. Routers perform cost normalization by scaling the costs of incoming advertisements such that the sum of the other ISP's advertised costs matches the sum of the ISP's own advertised costs. Cost normalization ensures comparability between ASes costs and prevents artificial inflation of costs. Upstream routers can use these normalized costs to make routing decisions that are better for both the source and destination ASes. However, Wiser requires bidirectional communication between peers to perform accurate path normalization, while BGP, being a path vector protocol, is unidirectional [3].
- **Pathlets** is an entirely new routing protocol that advertises fragments of paths, rather than entire paths as BGP does. Consequently, routers are presented with multiple path choices from which full paths can be constructed [4].
- **SCION** is a routing protocol that allows each router to advertise multiple paths, which gives

sources more control over the direction and volume of incoming traffic. Like Pathlet routing, this enables more sophisticated path-based routing [5].

- **S-BGP** provides a scalable method for verifying and authorizing BGP advertisements. This authentication addresses critical security vulnerabilities within BGP [6].

- **BGPsec** addresses critical security vulnerabilities within BGP using an approach that builds on S-BGP. It uses digital signatures within advertisements to ensure the legitimacy of incoming routing paths [7].

3 Evolvability and D-BGP

Previous work has identified features necessary for evolvability, specified a modified version of BGP called D-BGP that incorporates these features, created a limited D-BGP proof-of-concept, Beagle, and evaluated its performance overhead [1].

3.1 Evolvability Features

These extensions and alternatives to BGP solve many of the problems that currently exist with BGP, yet none are in widespread use. The reason lies within BGP's rigidity. The design of the protocol assumes that all neighbors of a BGP-speaking AS also speak BGP. Because almost all inter-domain routing is done using BGP, this has not presented an issue in the past. However, it makes the introduction of new protocols into the network more difficult. It is infeasible and unreasonable to expect that ISPs would all be able to switch atomically from BGP to another improved protocol. Thus, if newer protocols are to be introduced to the Internet, they must be deployed incrementally.

Such an incremental deployment of a new protocol would inevitably create groups of ASes supporting the new protocol, dubbed islands, separated by routers that do not support the new protocol, dubbed gulfs. While routers within an island would be able to take advantage of the benefits of the new protocol, the presence of the gulf between two islands forces them to downgrade

to a more basic protocol that is understood by all routers along the path. Thus, gulfs prevent the Internet from evolving to utilize inter-domain routing protocols other than BGP.

Prior work has identified two features that are necessary for a protocol to allow evolution [1]. The first feature is *multi-protocol advertisements*, which allows ISPs to utilize multiple protocols simultaneously. Without this support, it would be necessary to choose a single successor to BGP, which, like BGP, would be difficult to replace. The second feature is *pass-through support*, which enables any new protocols' control information to be carried through gulfs and thus reach other islands using that same protocol. This removes the restriction that prevents routing protocols from traversing gulfs and allows the advantages of the protocol to be seen across islands.

3.2 D-BGP Design

D-BGP is designed as an extension to the Border Gateway Protocol Version 4 (BGPv4). It must be an extension of BGP so that it is able to communicate with other routers still running plain BGP. D-BGP incorporates the previously identified evolvability features. Protocol control information is placed in a BGP *optional transitive attribute* to enable pass-through support. The optional nature of these attributes is useful because backward compatibility would be necessary when deploying D-BGP itself.

D-BGP's specialized advertisement, called an *integrated advertisement* (IA), consists of a plurality of data fields. It includes a shared data container which can carry control information for multiple protocols at once, enabling the required *multi-protocol advertisements*. The first of the fields is a specialized *path vector* field. Both island IDs and AS numbers are used in this field, and as such it provides a relationship between islands and AS numbers. This path vector field acts as a baseline that other protocols must use to avoid loops.

Following the path vector are *path descriptors*, which contain routing information for each protocol. For example, Wiser may decide to put a path cost field in this section.

Integrated advertisements also contain *island descriptors*, which contain protocol information specific to certain islands. For example, Wiser may decide to add an island-specific IP address for

other islands to connect to when performing cost normalization.

Islands that use link-state protocols must fall back to a path vector protocol when crossing gulfs. In this case, the router on the edge of the island must place the island ID in BGP's path vector as a whole as if the island were a single router. This process ensures the consistency and usability of the path vector field in determining the path described by the advertisement. When using this advertisement for making routing decisions, traffic is sent to the island as normal, and then further routing is performed within the island using the link-state protocol's control information [1].

3.3 Beagle Implementation

The previously introduced implementation of D-BGP, called Beagle, is a limited implementation of D-BGP and does not truly implement all of the evolvability features identified above. D-BGP would dictate that Integrated Advertisements in-band within regular BGP advertisements. However, Beagle uploads this information to an external server and places a small lookup key within the advertisement. Other routers must then retrieve this information from this external lookup service. This out-of-band strategy, while useful as a proof-of-concept, has relatively poor performance. There were a number of other features concerning path vectors and descriptors that were also not fully implemented.

While Beagle's implementation of D-BGP was incomplete, it showed that D-BGP as a protocol is promising, as it did not require an excessive amount of code to introduce a new protocol, and did not cause excessive overhead in advertisements.

4 In-Band Design

Tarpan[1], which modifies the design of D-BGP to use in-band communication when possible, is proposed. In-band communication allows the eventual implementation to remain faithful to the previously identified evolvability features while still demonstrating the advantages of evolvability.

[1]Like Beagle, the Tarpan implementation is based on an open-source routing software package named Quagga, a fork of Zebra. Both the quagga and the tarpan are extinct species of the *Equus* genus.

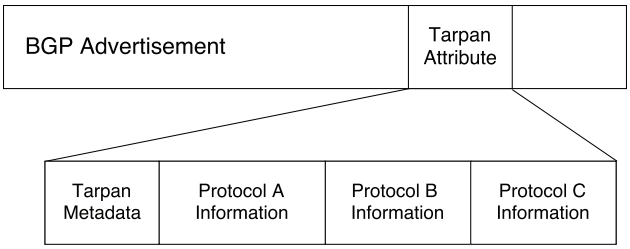


Figure 3: A diagram of the Tarpan data structure that is inserted into advertisements within a BGP optional transitive attribute. It enables both pass-through support and multi-protocol advertisements.

Because interoperability with regular BGP routers is a priority, Tarpan is designed to operate within an optional transitive attribute in a BGP advertisement. Optional attributes are meant to signify that the router is not required to understand what the attribute means. The transitive part specifies that the router should pass the attribute along as-is, even if it does not recognize the attribute. Therefore, optional transitive attributes are ideal for Tarpan.

This Tarpan attribute serves as a centralized, shared data container. Besides the path vector information, all other information is also contained within this attribute. The layout of this Tarpan data structure is shown in Figure 3. It contains metadata, which includes a version number, AS numbers, island IDs, and other general information. In essence, this metadata encompasses similar information to D-BGP. The Tarpan attribute also contains the control information for any number of protocols, therefore it is capable of serving arbitrarily complex routes.

Routers are able to choose which custom protocols to support. They can add control information or modify it as needed for those protocols. Tarpan supporting routers must pass through the control information of unsupported custom protocols as-is. Simple BGP routers will pass through the entire Tarpan data structure as-is because it is contained within a transitive attribute.

Different inter-domain routing protocols, like SCION and BGP, express their paths differently; SCION advertises multiple path options, while BGP only expresses a single path. Because of these differences in expression, Tarpan populates the path vector field using the lowest common denominator between all the possible paths by mixing AS numbers and island IDs.

Additional control information for each protocol is contained within that protocol's segment in the advertisement. This control information includes data that can be used for traversing gulfs or

ensuring proper functioning of that protocol. For example, BGPsec would store its digital signatures in this section. Because all of these protocol-specific segments are contained within the Tarpan attribute, they will be carried in-band with the advertisement.

Certain protocols require bidirectional communication to function fully. For example, the Wiser protocol needs this two-way communication to perform accurate cost normalization across gulfs. However, Tarpan, like the BGP on which it is based, is a unidirectional protocol. Wiser can embed an Internet Protocol (IP) address and AS number within its control information, allowing upstream routers to establish an out-of-band channel through which the bidirectional communication can be completed. Thus, any custom protocol can operate from within the Tarpan data structure.

5 Implementation

The implementation of Tarpan was focused on demonstrating the viability of an in-band approach to evolvability, and additionally to enable a systematic evaluation of the performance characteristics of this in-band approach. As such, only a subset of the features specified in Section 4 were implemented.

The Tarpan implementation was built on top of the Quagga routing software [8], an open-source software package containing implementations of various Internet routing protocols, including BGP. The Tarpan API was implemented in C, but the implementations of each custom protocol may also use any compatible language. For example, the Wiser implementation that was created to cooperate with Tarpan was written in C++.

5.1 Quagga Modifications

Figure 4 shows a diagram of the methods modified while implementing the Tarpan router. These changes affected approximately 1,400 lines of code.

When an advertisement is received, Quagga calls the `bgp_update_receive` method, which parses the contents of the BGP advertisement. Then, to extract the data from the attributes, the

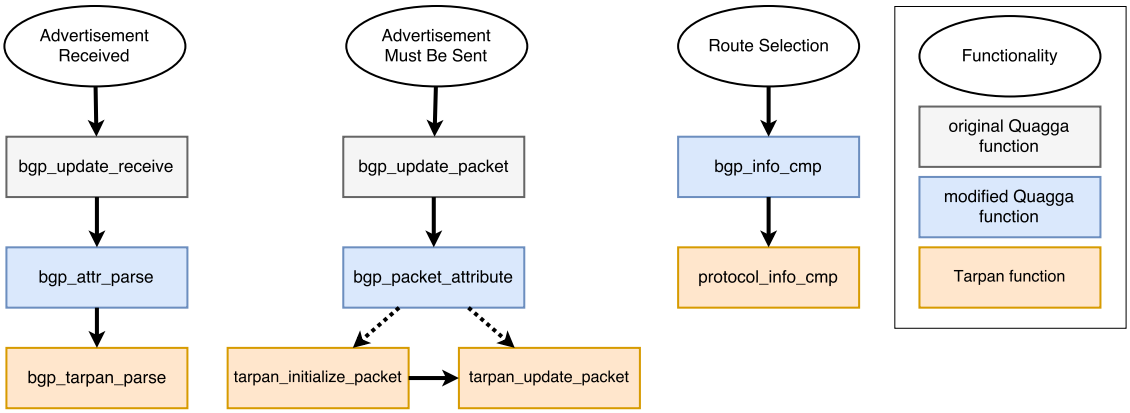


Figure 4: Diagram of methods that were modified or added when implementing Tarpan within Quagga. Each process is shown as a series of method calls (arrows), some of which were modified to perform Tarpan functionalities.

`bgp_attr_parse` method is called. In this method, code is added to check for the presence of a Tarpan attribute. If a Tarpan attribute is present, `bgp_tarpan_parse` is called to deserialize the attribute's data. If there is no Tarpan attribute present, it creates one, calling the Tarpan API implementation to generate an empty Tarpan data structure.

When Quagga decides to propagate an advertisement to a peer, it calls the `bgp_update_packet` method, which serializes the stored data structure and sends it over the connection to the peer. During this process, the `bgp_packet_attribute` method is called to serialize the attributes. Code was inserted in this method to serialize the Tarpan data structure and send it to the peer.

During the process of route selection, Quagga calls the `bgp_info_cmp` method. This call is then redirected to the `protocol_info_cmp` method, which compares two possible routes and selects the preferable one. If the Tarpan router is running multiple protocols simultaneously, then the API method to use is specified in the configuration.

5.2 Tarpan API

In implementing Tarpan within Quagga, a generalized API (specified in Table 1) was created for custom protocols. This way, it is trivial to add a new custom protocol or switch between sets of protocols using Tarpan. To add a protocol implementation, it is necessary only to add a message

Method	Purpose
`packet_received_handler`	Handles incoming advertisements containing a protocol's information
`initialize_packet`	Initializes control information when sending a new advertisement
`update_packet`	Called when forwarding an existing advertisement
`protocol_info_cmp`	Selects the preferable of two possible routes

Table 1: The Tarpan API, which allows the implementation of any custom protocol.

format specification and to implement the required methods of the Tarpan API. These methods will then be called as needed by the Tarpan implementation.

5.3 Protocol Buffers

The Tarpan attribute contained a data structure that was defined and serialized using Google's Protocol Buffers library [9]. This library provides fast and space-efficient serialization of arbitrary data structures. Protocol Buffers are also designed to allow extending existing messages to contain new fields. Servers that do not contain the newer version of the message definition will simply pass through the unknown data, while still allowing access to the known information. This is crucial to Tarpan's evolvable design.

5.4 Difficulties with Quagga

Due to the way Quagga is written, modifying it to add Tarpan functionality is challenging.

Interning To save memory, Quagga's internal data structures are interned. This means that they are hashed, and stored in a hashtable to deduplicate memory. In Quagga, memory deduplication is important because it is likely that there will be a large amount of duplicate data being received by the router, such as identical strings in the communities and path origin attributes. Because Quagga is written in heavily optimized C, interning data structures involves manual hashing, error-prone manipulation of pointers, and careful lifecycle management. Interning proved to be the cause of many segmentation faults and inconsistencies when implementing Tarpan within Quagga.

Simulation Environment The simulation environment in which we tested our modifications to Quagga was only compatible with older Linux kernel versions, and had dependencies on old, unsupported software. This necessitated the fallback to Ubuntu 12.04, a release which is no longer officially supported; the current version of Ubuntu is 16.04.

6 Performance Evaluation

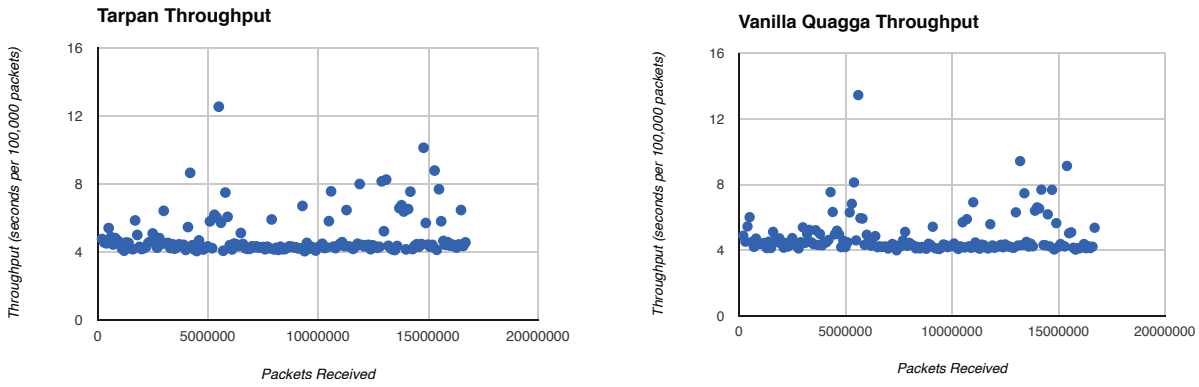
The purpose of these performance evaluations was to ensure that Tarpan does not have a significant negative impact on the performance of routers.

All of these experiments were conducted on a virtual machine provided by the Massachusetts Open Cloud [10]. The virtual machine had 16 vCPU cores and 64 GB of RAM. Using the miniNExT [11] extension of mininet [12], we could design network topologies and simulate the behavior of many routers within it. The `bgpsimple` [13] script, a fairly basic Perl script that acts as a BGP peer and can send arbitrary routing tables to a specified BGP peer, was used as a simple means of injecting real-world routing tables into the networks.

6.1 Throughput Experiment

The purpose of the throughput experiment is to determine if there is any sizable decrease in performance using Tarpan to process regular BGP advertisements versus using Quagga.

Setup The topology used in this experiment, which was simulated using miniNExT, included four Tarpan/Quagga virtual hosts connected in a "chain" configuration. Eight load generation nodes were connected to one of the routers at the end of this chain. The `bgpsimple` script was run on all eight nodes approximately simultaneously, and an instrumented version of both Tarpan and Quagga routers was running on the single router node receiving the advertisements. The amount of time required to process sets of 100,000 advertisements was collected over a 15 minute period.



(a) Tarpan

(b) Quagga

Figure 5: Graphs of Tarpan and Quagga throughput in seconds per 100,000 packets (lower is better). They show that both routers have very similar performance characteristics, and process packets at nearly the same rate.

Results and Discussion The results of the tests described above are depicted in Figure 5, where Figure 5a shows Tarpan's processing rate, or inverted throughput, and Figure 5b shows that of Quagga. From the striking similarities between these two, it can be concluded that Tarpan and Quagga have very similar performance characteristics. This is expected, as Tarpan is built on top of Quagga.

Tarpan has a minimal effect on advertisement handling throughput, adding an average of 0.55 μs of processing time to each advertisement, which is approximately 1.1% slower than Quagga. An interesting feature of these data is the fan-shape towards the latter times. While the average values remained fairly constant throughout, drops in speed become more common as time passed, likely because of the growing size of the routing table as more routes were advertised. Further optimizations may be made to lower this overhead. For example, there was some extremely verbose logging in the Tarpan code to assist with debugging, which may have had a significant effect on performance.

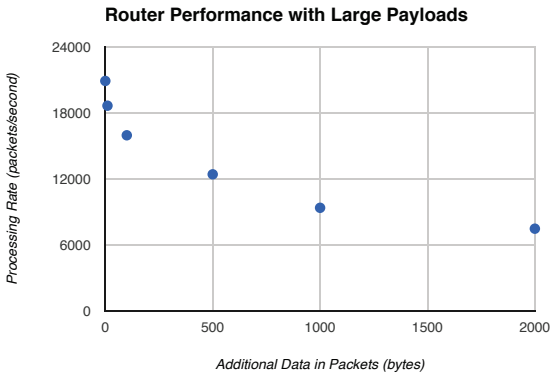


Figure 6: Throughput degradation in vanilla Quagga with larger packet sizes.

6.2 Large Payload Experiment

The purpose of the large payload experiment was to determine how much transferring the routing information in-band affects the performance of other non-Tarpan routers on the Internet. Newer routing protocols may require the transfer of more data in attributes than is typically done today, and that may degrade performance in other routers.

Setup A modified version of the `bgpsimple` script was used to send advertisements to a vanilla Quagga instance with attributes containing a fixed string of different lengths.

Results and Discussion As shown in Figure 6, performance in the vanilla Quagga router decreased sublinearly as packet size increases. The initial introduction of an additional attribute caused a large drop in performance, likely because of the overhead in processing and storing that attribute. Larger packets caused less performance degradation per byte than smaller packets. Based on these results, it may be advantageous to use a single larger attribute instead of multiple smaller attributes.

These results did not exactly match the expected behavior. It was expected that the time spent processing each packet would increase linearly with the amount of additional data stored in the packet. However, the results indicate that this is actually a sublinear relationship: as the attribute size increases, it has less of an impact on performance on a per byte basis.

It is not clear if this is a result of the testing methodology or if it is truly characteristic of

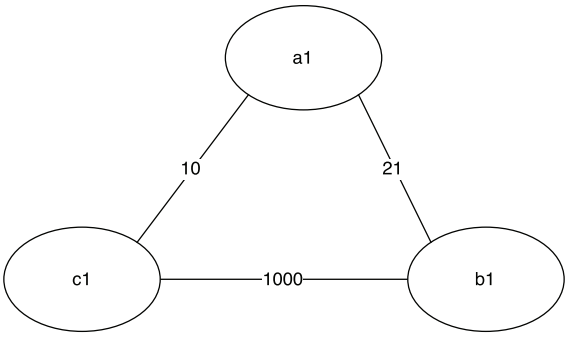


Figure 7: The small Wiser test topology. This topology was designed to ensure that the basic premise of Wiser, path costs, was indeed taken into account. In this topology, a regular BGP implementation would choose the direct route between `b1` and `c1`, while Wiser would pick the route through `a1` because it has a lower total cost.

Quagga's performance. One possible reason for this behavior is that all of the packets contain the same string. This could have triggered some optimizations in Quagga, as it manually deduplicates memory with interning. It is also possible that this behavior is affected by packet fragmentation, buffering overhead, or Nagle's algorithm. Further investigation is warranted, such as including random strings of data to suppress the interning mechanism and measuring the time taken in specific subroutines within Quagga.

7 Deploying Protocols

The purpose of these experiments was to verify that Tarpan does work as intended, and moreover to verify that additional protocols' control information can indeed be carried over a Tarpan attribute. A secondary purpose was to confirm that bidirectional protocols can operate on top of BGP successfully with the use of limited out-of-band data.

Setup To verify that protocols do indeed work properly over Tarpan, the Wiser protocol was implemented, including the bidirectional support. To ensure that the implementation was correct and functioned properly, especially over gulfs, the topologies pictured in Figures 7 and 8 were created. These topologies were designed to test different functions of Wiser, including route selection, cost propagation, efficacy over gulfs, and bidirectional communication. The route selections produced

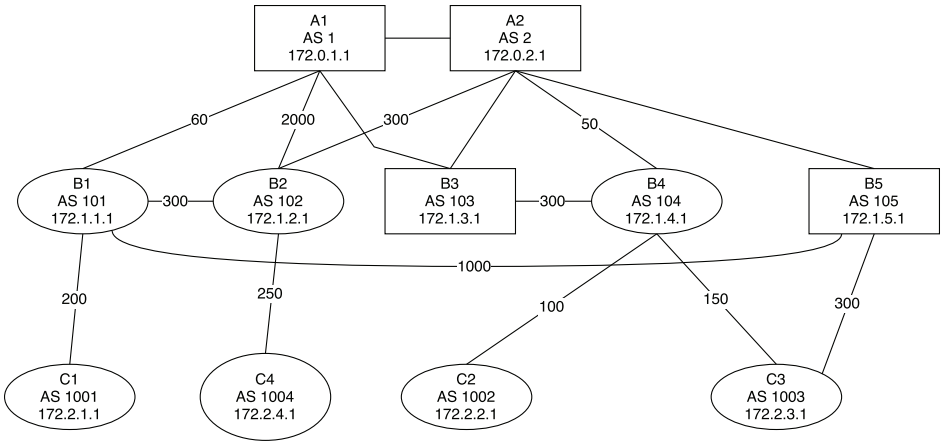


Figure 8: The large Wiser test topology. This topology was designed to model real-world Internet architectures, with tier 1 ISPs represented as ASes 1 and 2. Underneath these top-level ASes are a number of other ASes with smaller range. The topology as a whole is designed to have a number of islands with many gulfs between them, effectively forcing Wiser to perform even when its peers have fragmented support for the protocol.

by Wiser in these topologies were verified by hand.

Results and Discussion The implementation of Wiser did indeed choose the correct, lowest-cost paths to the destination, as well as successfully transferred data over gulfs. However, path cost normalization, which operates out-of-band, was disabled to simplify verification on the large test topology, though it did function as intended within the small test topology.

8 Future Work

There are a number of avenues that can be pursued in the future. We plan to perform further performance benchmarks using more realistic, real-world topologies, as well as involving more metrics, such as memory usage and link saturation. We are also planning to implement additional protocols to ensure that all such protocols can function in-band within Tarpan. We will explore the effects of running myriad protocols within a simulated Internet, and explore the performance characteristics of the network when running these protocols simultaneously. We will also look into route convergence properties, and optimal strategies for selecting routes when protocols are in disagreement. Finally, we will try to formally prove the incremental deployability of Tarpan itself.

9 Conclusions

The Internet's inter-domain routing infrastructure is largely powered by the Border Gateway Protocol. This protocol is outdated and has many problems, some of which have proposed solutions. However, it is difficult to deploy these solutions, since BGP is a rigid protocol that assumes all of its neighbors are also running BGP. Previous work has identified certain characteristics of an extension to BGP that would enable evolvability: pass-through support and multi-protocol advertisements. While D-BGP incorporates these features, its Beagle implementation uses out-of-band communication, which is suboptimal for practical reasons. Tarpan is a new protocol that supports these evolvability features, and emphasizes the in-band transmission of protocol control information. Tarpan is designed as an extension to BGP and is implemented on top of the open-source routing software package Quagga. Tarpan's performance was evaluated using raw throughput of routing data ingestion and an implementation of the Wiser protocol on top of Tarpan. It was found that Tarpan introduces a negligible amount of overhead in processing attributes, and that the Wiser protocol is relatively simple to implement using the Tarpan API. Further, the effect of increasingly large attribute sizes was tested, and performance decreased sublinearly with increasing packet size. This work has shown that an in-band, evolvable protocol is feasible to implement and does not suffer from significant overheads compared to BGP. Tarpan supports the evolution to more sophisticated inter-domain routing protocols that can operate simultaneously and within existing BGP advertisements.

References

[1] Mentor, D. Tran-Lam, A. Akella, and P. Steenkiste, "Bootstrapping evolvability for inter-domain routing with D-BGP," in *Proceedings of the Conference of the ACM Special Interest Group on Data Communication*, ser. SIGCOMM '17. New York, NY, USA: ACM, 2017, pp. 474–487. [Online]. Available: http://doi.acm.org/10.1145/3098822.3098857

[2] Y. Rekhter, T. Li, and S. Hares, "A border gateway protocol 4 (BGP-4)," Jan. 2006, RFC 4271, DOI: 10.17487/RFC4271.

[3] R. Mahajan, D. Wetherall, and T. Anderson, "Mutually controlled routing with independent ISPs," in *Proceedings of the 4th USENIX Conference on Networked Systems Design &*

Implementation, ser. NSDI'07. Berkeley, CA, USA: USENIX Association, 2007, pp. 26–26. [Online]. Available: http://dl.acm.org/citation.cfm?id=1973430.1973456

[4] P. B. Godfrey, I. Ganichev, S. Shenker, and I. Stoica, "Pathlet routing," in *Proceedings of the ACM SIGCOMM 2009 Conference on Data Communication*, ser. SIGCOMM '09. New York, NY, USA: ACM, 2009, pp. 111–122. [Online]. Available: http://doi.acm.org/10.1145/1592568.1592583

[5] X. Zhang, H.-C. Hsiao, G. Hasker, H. Chan, A. Perrig, and D. G. Andersen, "SCION: Scalability, control, and isolation on next-generation networks," in *2011 IEEE Symposium on Security and Privacy (SP)*. IEEE, 2011, pp. 212–227.

[6] S. Kent, C. Lynn, and K. Seo, "Secure border gateway protocol (S-BGP)," *IEEE Journal on Selected areas in Communications*, vol. 18, no. 4, pp. 582–592, 2000.

[7] M. Lepinski, "BGPsec Protocol Specification," [work in progress]. [Online]. Available: https://tools.ietf.org/html/draft-ietf-sidr-bgpsec-protocol-23

[8] "Quagga software routing suite," http://www.nongnu.org/quagga/. [Online]. Available: http://www.nongnu.org/quagga/

[9] Google, "Protocol Buffers," https://developers.google.com/protocol-buffers/, version 2.

[10] "Massachusetts Open Cloud (MOC)," 2017. [Online]. Available: http://www.bu.edu/hic/research/highlighted-sponsored-projects/massachusetts-open-cloud/

[11] bschlinker, "USC-NSL/miniNExT." [Online]. Available: https://github.com/USC-NSL/miniNeXT

[12] Mininet Team, "Mininet: An instant virtual network on your laptop," 2017. [Online]. Available: http://mininet.org/

[13] A. Korolyov, "BGP simple," https://github.com/xdel/bgpsimple, 2011. [Online]. Available: https://github.com/xdel/bgpsimple