

# Large-Scale Simulation Models of *BGP* \*

Xenofontas A. Dimitropoulos  
George F. Riley

College of Engineering  
Department of ECE Georgia Institute of Technology Atlanta, GA 30332-0250  
{fontas,riley}@ece.gatech.edu

## Abstract

*The complex nature of the Border Gateway Protocol (BGP) is not amenable to analytical modeling, and thus simulation-based analysis methods are needed to understand its behavior. To this end, we investigate techniques that make large-scale BGP simulations feasible. The described techniques partially ameliorate the memory and execution time bottlenecks to yield large-scale BGP simulations. Our focus is cast on efficient sharing of BGP routing tables, execution time optimizations for simulation trials, and proper partitioning of parallel distributed BGP simulations. Moreover, we survey the requirements for realistic, Internet-like BGP simulations and develop a simulation toolset to expedite realistic configuration of a BGP simulator.*

## 1. Introduction

*BGP* is the default interdomain protocol of the Internet. Despite its lengthy deployment, *BGP* has recently received increased attention by the research community. This interest has been cast on several aspects of its behavior, namely convergence process, security mechanisms, path vector properties, lack of traffic engineering and performance metrics. *BGP* plays a multidisciplinary role in the Internet: besides carrying the duties of a regular routing protocol it embeds commercial relationships in the routing infrastructure via the specification of routing policies. Moreover, the complexity of *BGP* which is inherent in distributed systems, is exacerbated by the abrupt and irregular growth of the Internet. For these reasons, the study of the protocol as deployed in the Internet is difficult. This is demonstrated by the fact that some design and implementation problems were diagnosed only after deployment [2], rather than in testbeds or by analysis.

In studies of the protocol, simulation analysis has been used often. In contrast to analytical methods, simulation models can preserve the details of the protocol and unlike testbed techniques, enable larger experiments. However, simulation tools have their own scalability limits, primarily due to the memory requirements that grow quickly as the level of detail and the size of the model grow. In many scenarios, medium size simulations, which are fairly easy to construct, are inadequate. Moreover, the evolution of the Internet measurements area in the last few years has made available substantial information that can be used to enhance the accuracy of simulations. These measurement studies include AS level topologies [22] and router level topologies [12] [23] [8]. Therefore, scalable and accurate *BGP* simulation are essential.

This work elaborates on techniques that make large-scale *BGP* simulations feasible. Both new and older scalability methods are evaluated on a *BGP* simulator. The discussed methods focus on the memory and execution time bottlenecks, with the following contributions:

1. A routing table information sharing architecture.
2. A generic technique that reduces the execution time of simulation trials of the same configuration.
3. An evaluation of different partitioning schemes for parallel distributed *BGP* simulations.
4. A survey of challenges in the development of realistic *BGP* simulations.
5. A simulation toolset to expedite *BGP* related research. The toolset generates simulation configuration, taking input data from *BGP* monitoring projects, schedules multiple simulation runs, and automates partitioning and configuration of parallel distributed *BGP* simulations.

The discussed methods are implemented and evaluated on our *BGP* implementation for the popular ns-2 [15] simulator, called BGP++. BGP++ was created by porting Zebra [11] *BGP* daemon (bgpd) into the ns-2 simulator. Our previous work [4] discusses challenges and methodology in integrating Zebra software with ns-2. The simulator supports most *BGP* features and several extensions, such as

---

\* This work is supported in part by NSF under contract numbers ANI-9977544 and ANI-0136969, and in part by DARPA under contract number N66002-00-1-8934.

AS confederations and route reflection. One of the features of BGP++ is that the configuration parsing engine of Zebra bgpd is maintained in the simulator, meaning that simulated routers parse configuration files with the same command syntax with Zebra BGP routers.

The remainder of this paper is organized as follows: Section 2 describes in detail our optimizations for large scale BGP simulations. Section 3 elaborates on realistic BGP simulations and introduces a toolset to expedite simulation studies. Section 4 talks about works related to BGP simulation. Finally, Section 5 discusses future research directions.

## 2. Simulation Optimizations

In this section we describe techniques to address the excessive memory consumption and the long execution time problems in large-scale sequential and distributed BGP simulations. Simulation is used to quantify the efficacy of the proposed techniques. Unless otherwise stated, the following setup is employed: The topology is a random subgraph of the Internet AS topology as seen from RouteViews [22] of size  $N$ ; each AS has a single BGP router that originates  $p$  prefixes. The policies between the ASes are inferred using the heuristics described in [7]. The latter classifies policies in three categories Provider-Customer, Peer-Peer and Sibling-Sibling. For simplicity we modify Sibling-Sibling relationships to Peer-Peer<sup>1</sup>. The inferred relationships are configured using BGP communities as follows: a provider advertises to a customer all the routes it knows; a customer advertises to a provider routes that are either locally originated or learned by its customers; likewise a peer advertises to its peers routes that are either locally originated or learned by its customers. These configurations are consistent with typical operator practices. The processing workload of a router is not simulated and the delay and the data rate of all links are set to 10ms and 10Mbps, respectively. Each simulation is run until the system reaches steady state, i.e. no updates are exchanged. For the remainder of this paper we refer to this setup as  $Internet(N, p)$ .

### 2.1. Memory Optimizations

This section articulates on memory usage of BGP simulations and methods to ameliorate excessive memory consumption. Memory consumption depends mainly on the following parameters: size of topology, size of routing tables, number of neighbors per router, simulation dynamics, and the simulator memory footprint. A BGP router maintains three routing information bases (RIBs), an Adj-RIB-in, a Loc-RIB and an Adj-RIB-out [19]. The Adj-RIB-in stores routes received by neighboring BGP routers, the Loc-RIB stores routes selected by the decision process and the

Adj-RIB-out stores routes advertised to other BGP routers. Routes in the Adj-RIB-out are subject to export policies.

The following formula approximates the effect of different simulation parameters on memory demand:

$$Memory \approx \alpha N + \beta p N^2 + \gamma p N^2 r + \delta \quad (1)$$

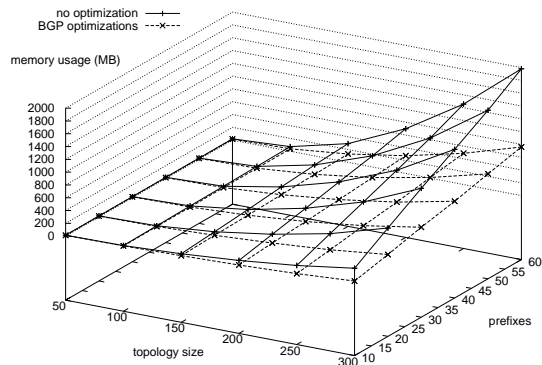
where  $N$  is the number of BGP routers,  $p$  the average number of prefixes originated per router,  $r$  the average number of neighbors per router,  $\alpha$ ,  $\beta$ ,  $\gamma$  and  $\delta$  proportionality constants. Term  $\alpha N$  represents the memory cost to initialize the BGP routers, in other words the memory cost for routers with empty routing tables; term  $\beta p N^2$  accounts for the Loc-RIBs memory consumption, assuming that each router after convergence has  $pN$  entries in the routing table; term  $\gamma p N^2 r$  is the worse possible memory consumption for the Adj-RIB-ins and Adj-RIB-outs, assuming that each router receives  $pN$  routes from each of its  $r$  neighbors<sup>2</sup>. Formula 1 does not account for other minor sources of memory consumption.

The first optimization we discuss is a simple architecture to have the Loc-RIB(s), Adj-RIB-in(s) and Adj-RIB-out(s) share the same memory; it is a variation of GNU Zebra [11] memory recycle scheme. A BGP router stores several attributes for a routing entry, including AS path, origin, next hop, local preference, multi-exit discriminator (MED), communities, extended communities and unknown<sup>3</sup>. These attributes account for most of the memory required for a table entry, and therefore our method attempts to share the attribute information between information bases. In this scheme, each route is associated with a pointer to a data structure, *struct attr*, which has an entry for each of the attributes. For attributes with size less or equal to 32 bits the entry is the value of the attribute, otherwise a pointer to another data structure specific to the attribute in question. For example, *struct attr* contains the value of the MED attribute, which is a 32-bit unsigned integer, while it contains a pointer to an AS path data structure, which can be of arbitrary size. The actual AS path data structure is associated with a reference counter and stored in a global hash table; there is one global hash table for each type of attribute with size greater than 32 bits. A BGP router, after allocating and initializing memory for a new attribute, searches the relevant hash table for the newly allocated attribute; in case of a match, the allocated memory is deallocated and a reference to the attribute in the hash table is used. In addition to that, the reference counter, which is used to track the number of pointers to an attribute, is incremented. If a router wishes to remove an attribute, it decrements its reference counter. When the reference counter becomes zero the attribute is removed from the hash table and deallocated. Similarly, a second level hash table is used to store and share

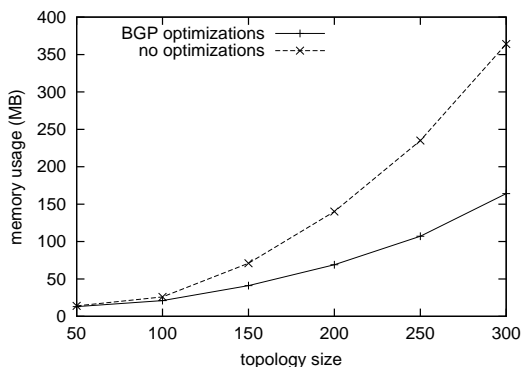
<sup>1</sup> This modification has no impact on evaluating the discussed techniques.

<sup>2</sup> The introduction of policies tends to decrease memory demand by limiting the number of paths known to a router.

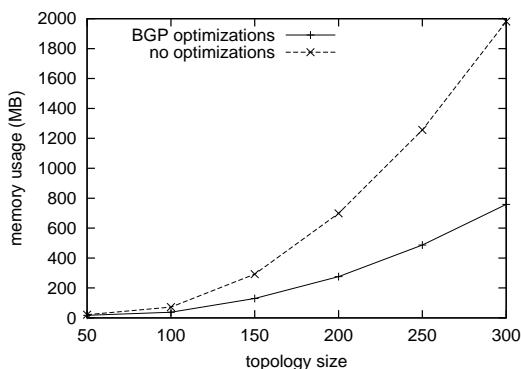
<sup>3</sup> BGP specifications instruct to transit unknown attributes.



(a) Memory usage versus topology size and originated prefixes per router



(b) Cross-section of 1(a) for 10 originated prefixes per router



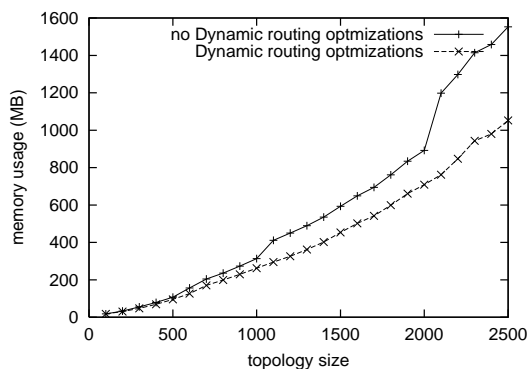
(c) Cross-section of 1(a) for 60 originated prefixes per router

**Figure 1. Memory savings by sharing information among the information bases**

*struct attr* structures. This scheme enables sharing of memory among multiple information bases within a *BGP* router, as well as among multiple *BGP* routers. It mitigates the impact of the  $\beta pn^2$  and  $\gamma pn^2 r$  terms of equation 1.

Simulation is employed to evaluate the efficacy of this memory sharing architecture. Figure 1(a) illustrates the total memory consumption with and without sharing versus  $N$  and  $p$  in an *Internet*( $N, p$ ) setup. Cross-sections of the figure are shown at 1(b) and 1(c) for 10 and 60 originated prefixes per router respectively. The highest memory reduction achieved, is 62% with a mean of 47% with respect to the total memory. Note that the total memory consumption is the result of several sources of memory demand besides the information bases. The memory reduction with respect to the memory required for the unmodified information bases is even bigger. The same approach could be used to conserve memory for other data structures, e.g. flap dampening information, MRAI suppressed advertisements.

The second optimization is based on a routing method called Nix-Vector routing, which has been shown to reduce memory consumed for routing tables in ns-2 [20]. This is achieved by computing routes on demand as needed, rather than pre-computing all possible routes. *BGP* operation presumes an underlying Interior Gateway Protocol (IGP) to keep *BGP* routers connected. Moreover, both *BGP* and IGP maintain the FIB which is used to forward packets. The Nix-Vector routing technique can be used in the context of *BGP* simulations to replace the functionality of an IGP protocol. We evaluate the memory conserved by using Nix-Vector routing as compared to ns-2 default static routing. Figure 2 illustrates the memory consumption in an *Internet*( $N, 1$ ) setup. The mean memory reduction is 22% and the peak 36%. For  $p$  larger than 1, the memory required for *BGP* information bases dominates and the savings of this approach cannot be demonstrated.



**Figure 2. Memory savings by using Nix Vector routing**

## 2.2. Execution Time Optimizations

In this section we describe a technique to reduce simulation time, which is a significant limiting factor in large-scale packet-level simulations.

Previous approaches have focused on trading model detail for speed. To exemplify, several researchers have used fluid model methods for large-scale TCP simulations. Furthermore, the work in [9] ignores the protocol stack below the application layer to yield fast large-scale simulations. Even in an idealistic scenario, where unlimited memory resources were available, long execution times would deter large-scale studies of *BGP* behavior. Long execution times are not atypical; in a recent comparison of network simulators [16], execution times of several hours for a problem of substantial size were reported. Assuming that a simulation scenario should be repeated several times to make the results statistically significant, simulation trials may take days run. An approach to mitigate the problem is described that explores the observation that non-trivial time is spent to initialize the models, and that the initialization is repeated for each of the trials. The approach is useful when multiple simulations of the same setup have to be run.

The initialization of the simulation models consumes typically a considerable part of the simulation time. In our approach, the state of a simulation is saved as soon as the initialization has been completed and before the random number generator is seeded. Then, each trial starts from the saved image and enters immediately the event processing phase of the simulation, circumventing initialization. If  $r$  is the ratio of the initialization to the total execution time and  $k$  the number of trials, the speedup is:

$$Speedup = \frac{ET_{normal}}{ET_{optimized}} = \frac{1}{1 - r + \frac{r}{k}}$$
<sup>4</sup>

For  $r = 0.2$  and  $k = 50$  the speedup is 1.244. Although this method is generic, it is evaluated in *BGP* context. State saving is attained using Condor [26], which is a workload management system that supports checkpointing and restarting a process. Two modifications are implemented in *BGP++* using Condor's checkpointing library. The first modification provides a command to request a checkpoint to be written in a file as soon as initialization has completed. The initialization phase includes construction of network objects, calculation of static routes if needed, initialization of *BGP* data structures and parsing of *BGP* configuration files. The second modification enables checkpointing the simulator process at any point during a simulation, and an optional reconfiguration of the simulation as soon as it restarts. To clarify, when the simulator process is restarted from a checkpoint, it reads commands from a tcl file, which can optionally change the configuration.

<sup>4</sup> It is assumed that the overhead to restart a simulation from a saved image is negligible.

The ratio of the initialization to the total execution time is quantified in an *Internet(N, 1)* setup. Figure 3 depicts the total execution time and the initialization time for different  $N$ . The mean  $r$  is 0.17. The total execution time depends on the dynamics of the *BGP* system, which in turn depend on the seed of the simulation. This variation explains why the total execution time curve is not smooth. In a second run, where the second optimization of section 2.1 is deployed,  $r$  drops to 0.08 since calculation of static routes during initialization is bypassed.

The checkpointing feature could be used in more aggressive scenarios. For example, one could save the state of a *BGP* simulation once it has converged, and then run multiple experiments that point to measure the effect of a *BGP* withdrawal. This would result in greater execution time savings than reported here.

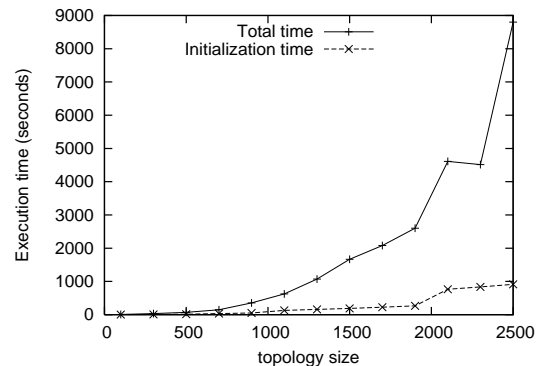


Figure 3. Total execution time and initialization time versus topology size.

## 2.3. Parallel Distributed Simulations

The use of parallel distributed techniques has been exploited to lead to faster and larger network simulations in tools such as *pdns*[21] and *SSFnet* [3]. *BGP++* has been extended to support parallel distributed simulation using *pdns*. The *pdns* simulator is a space-parallel, conservative synchronization simulator derived from ns-2[15]. “Space-parallel” indicates that the simulated model is partitioned and distributed on different processes (federates on *pdns* terminology) on different computing platforms. A conservative synchronization protocol bounds the lag of the simulation times between two federates to insure causal consistency. The simulation is divided into epochs; each epoch is followed by a synchronization interval, in which the length of the next epoch is computed. Also, during the synchronization interval, cross-federate events are delivered. Let  $ET_{ij}$  and  $CT_{ij}$  be the execution and communication time of federate  $i$  at epoch  $j$ , respectively. Also, let  $ST_j$  be the synchronization time at epoch  $j$ , then the execution time of

the event processing phase of a simulation is:

$$ET = \sum_0^{N-1} (\max_i \{ET_{ij} + CT_{ij}\} + ST_j) \quad (2)$$

where  $N$  is the number of epochs. A similar formalization can be found at [17]. Proper distribution of the model has significant impact on the performance of the simulation. In *pdns*, each federate is assigned a different part of the simulated topology. Consequently, distributing of the model becomes a graph partitioning problem. Partitioning has a direct impact on the terms  $ET_{ij}$ ,  $CT_{ij}$ ,  $ST_j$  and  $N$  of equation 2. In particular, partitioning should:

- Balance the processing workload to minimize the maximum  $ET_{ij}$  over  $i$ .
- Minimize the maximum communication load between any two federates. This has a direct impact on both  $CT_{ij}$  and  $ST_j$  since cross-federate events are delivered during synchronization.
- Maximize the length of the epochs to increase parallelism and reduce  $N$ .

The most challenging of these objectives is to accurately load balance a distributed simulation. This is because we cannot know in advance the processing load of a given partition. Recent works by Yocum et al. [28] and Liu et al. [14] propose methods to predict the load of a partition using static configuration information. Their work concentrates on distributed emulation, but their techniques are applicable in the context of space-parallel distributed simulations as well. Both schemes model the processing and communication load as weights of the vertices and edges of a graph and use graph partitioning algorithms to find good partitions. In particular, the graph partition problem is formulated as follows:

*Given an undirected graph  $G = (V, E)$ , where  $V$  the set of vertices and  $E$  the set of edges find a partition of  $G$  in  $k$  parts, that minimizes the edge-cut under the constrain that the sum the vertices' weights in each part is balanced.*

The remainder of this section discusses approaches to weight a graph and evaluates the performance of different partitioning algorithms in the context of distributed *BGP* simulations. In a typical *BGP* simulation each node is a *BGP* router, which establishes sessions with its neighbors and exchanges updates based on its configuration. The processing workload associated with a router can be broken down into maintenance workload, which refers to maintenance tasks such as sending keepalive messages, and dynamic workload which refers to dynamic tasks such as update message processing and propagation. Two approaches to model workload are considered. In the first, the CPU cycles spent for a simulated router are considered proportional to the number of neighbors of the router. This approach does not capture *BGP* dynamics and asymmetries that arise from policies and configuration. However, it does

capture the maintenance workload and provides a reasonable assumption in the absence of mechanisms to predict *BGP* dynamics. Consequently, the weight of each vertex is set to the degree of the vertex. In the second approach, all routers are equally weighted, hence the algorithm partitions the topology into parts that have roughly the same number of *BGP* routers. This is equivalent to not weighting the vertices of the graph, hence it is referred as unweighted. Similarly, links that cross federates are treated equally in terms of communication load, and therefore we don't apply distinct weights. This approximation captures faithfully the house-keeping traffic, but it may fail to capture asymmetries due to *BGP* dynamics and policies.

To access the performance of different partitioning algorithms the Chaco [10] and Metis [13] graph partitioning packages are used. The following algorithms are compared: Chaco multilevel Kernighan-Lin (KL), Chaco spectral, Chaco linear, METIS multilevel KL and Chaco random. In the latter, vertices are assigned randomly in partitions subject to the balance constraint; it is used as a worse case reference. Figure 4(a) shows the execution time for a variant of the *Internet(N, 1)* setup denoted as *Internet(N, 1, 200)*, in which the simulation time is fixed to 200 seconds. The topology is distributed over two federates by applying the above algorithms to an unweighted graph. We observe that the examined algorithms yield similar performance, while the savings with respect to a balanced random partitioning increase with the model size. The rest of the experiments use the METIS multilevel KL algorithm since it renders slightly better performance. We now compare the two weighting approaches described above. Contrary to our expectations, figure 4(b) illustrates that weighting a vertex by its degree yields worse performance than not weighting, in *Internet(N, 1, 200)* simulations. We identify the dynamics of the simulation as a potential reason and seek to investigate further. In the *Internet(N, 1, 200)* setup the increased "entropy" due to the indeterministic behavior of *BGP* convergence, as well as the routing policies, lead to imbalances in the dynamic workload that can overwhelm the symmetry of the maintenance workload. An *Internet(N, 0, 200)* setup is examined to reveal the impact of *BGP* dynamics; in this setup routers don't originate prefixes, which means that the workload for each partition is determined by the housekeeping tasks of each router, and is closely related to the degree of the router. Figure 4(c) illustrates that the weighted approach exhibits worse performance even in the absence of *BGP* dynamics.

Further analysis uncovers that weighting nodes by their degree results in an increase on the communication load because the partitioning algorithms performance worsens. The METIS multilevel KL partitioning algorithm finds worse solutions in terms of edge-cut when the graph is weighted. Similar results apply for the other partitioning algorithms as well. Table 1 shows the edge-cut of the partitions found by

the algorithm for weighted and unweighted graphs, where  $k = 2$ . The surge of the edge-cut results in a significant communication volume that lengthens the event processing as well as the synchronization phase of the simulation. This corresponds to the  $CT_{ij}$  and  $ST_j$  terms of equation 2. We verify that similar results hold for larger number of partitions. Thus, it is evident that the unweighted graph combined with METIS multilevel KL algorithm give the best results among the approaches examined. We speculate that the average performance of the partitioning algorithm for weighted graphs is because of the power-law properties of the AS graph. The power-law degree distribution of the Internet AS topology, results in power-law distributed vertex weights, which make compliance to the balance constrain complex and produce worse edge-cuts.

The impact of the examined partitioning approaches on lookahead has not been considered thus far. Lookahead is the maximum length for which a logical process can execute before synchronization is needed; it directly affects the length of the epochs. In *pdns*, the lookahead is derived from the latency of the communication channels between federates. In particular the following formula is used:

$$Lookahead = \min_i \{d_i + m/b_i\}$$

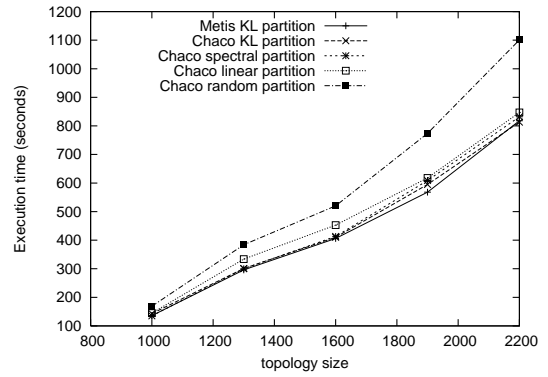
where  $d_i$  and  $b_i$  are the delay and bit rate of channel  $i$  and  $m$  the minimum possible packet size. It represents the time it takes for the smallest possible packet to travel across the fastest link. In the experiments of this paper the communication links have equal delay and bandwidth. Consequently, the lookahead does not depend on the partition and is not considered. If the characteristics of the communication channels are sensitive parameters and have different values, then the links can be weighted by their lookahead

5.

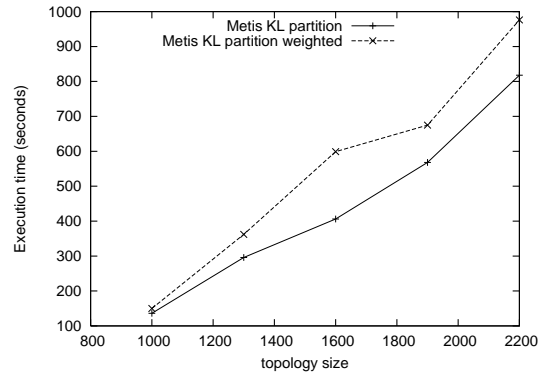
Topology size	unweighted	weighted
1000	33	119
1300	62	451
1600	90	442
1900	113	484
2200	175	719
2500	218	832

**Table 1. Edge-cut found by METIS multilevel KL algorithm for unweighted and weighted graph split into two parts**

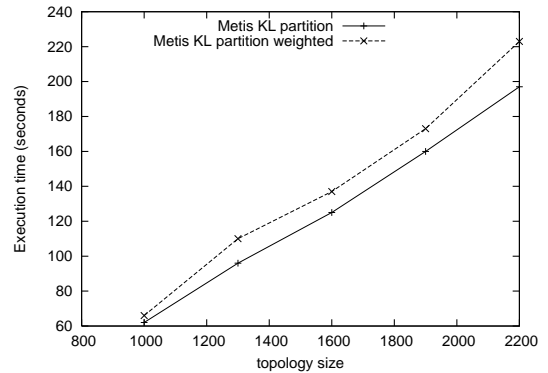
5 A transformation of the lookahead can be used to convert a maximization problem to minimization.



(a) Total execution time using different partitioning algorithms



(b) Total execution time of weighted versus unweighted partitioning in an *Internet*( $N, 1, 200$ ) setup



(c) Total execution time of weighted versus unweighted partitioning in an *Internet*( $N, 0, 200$ ) setup

**Figure 4. Partitioning of Parallel Distributed simulations**

### 3. Realistic Simulation Setup

The creation of realistic simulation models is a challenging endeavor. The reliability of conclusions drawn from simulation analysis depend highly on the accuracy and the correctness of the simulation. This section describes a simulation configuration that closely models the Internet’s *BGP* infrastructure, discusses limitations towards this end, and introduces a toolset to generate Internet-like *BGP* configurations as well as to expedite execution and scheduling of large simulation runs. A faithful simulation of Internet’s *BGP* infrastructure should account for:

- Internet topologies at the AS and router level.
- Routing policies between service providers.
- Models of iBGP connectivity patterns. With connectivity patterns, we mean the topological properties of the overlay iBGP network as well as iBGP connectivity alternatives, e.g. route reflection, *BGP* confederations, full mesh.
- Router processing delay models.

Other information such as IGP protocols used, IP blocks allocated, links’ delay and bandwidth, may also be necessary depending on the goals of the simulation analysis. In recent attempts to understand *BGP* behavior, researchers have created monitoring projects that collect numerous data from existing deployed *BGP* routers. The RouteViews project provides a union of the *BGP* tables of the largest service providers of the Internet and can be used to give a fairly accurate map of the Internet AS topology. *BGP* policies hide some links from the *BGP* tables, however optimization projects are in progress [1]. Router level topologies are available in several projects [12], [23], [8], although of debatable accuracy. The policies between the ASes can be inferred using heuristics from [7], [24]. To our knowledge, there are no models of iBGP connectivity patterns or monitoring projects that can provide this information. *BGP* processing delay is shown through simulation studies to bear impact on *BGP* convergence time [18]. Preliminary steps to characterize and model *BGP* processing time include [27] and [5].

We now describe a toolset to expedite simulation studies that uses RouteViews data to generate simulation configurations. Figure 5 illustrates the conceptual diagram of its architecture; the toolset consists of three components, a configuration generator, a partitioner, and a scheduler.

The configuration generator component parses an AS map derived from the *BGP* tables of RouteViews that has been annotated with relationships. The relationships are translated to *BGP++* configuration files. *BGP* communities and filters are used to translate relationships to configuration. The user has the option to specify different parameters that control the size of the topology, the routes advertised, route advertisement interval, tracing, memory saving scheme and others.

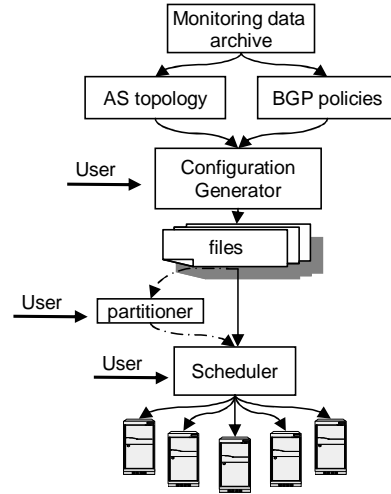


Figure 5. Block diagram of simulation toolset

The partitioner handles partitioning for parallel distributed simulations. The user can choose among different partitioning algorithms supported through METIS and Chaco graph partitioning packages. Then, given a sequential configuration file for ns-2, it generates the rather complicated configuration for *pdns* and its *BGP* extensions. Finally, the execution component handles scheduling of multiple simulation runs. A master slave architecture is implemented, in which a master process schedules and distributes simulations to slave machines, taking into account available memory, existing CPU workload, and other criteria.

### 4. Related Work

The *BGP* model by Premore in the SSFnet simulator was the first detailed simulation model of *BGP*, and is currently the most widely used. It supports most important *BGP* features and extensions. A limitation of SSFnet is that it has considerable memory demand, thereby preventing simulations larger than a few hundred of *BGP* routers.

The recent work by Hao and Koppol[9] addresses the challenge of large scale *BGP* simulations by ignoring the protocol stack below the application layer. This simplification, combined with a memory saving scheme, make large-scale *BGP* simulations feasible. The memory saving scheme attempts to dynamically construct the AS path and yields reduced memory consumption at steady state, however its performance and memory requirements during convergence are not clear.

Other efforts include [25] and [6] that have ported the SSFnet *BGP* implementation in Genesis and ns-2, respectively. Our work extends the previous works by evaluating

optimizations that make detailed as well as large simulations feasible.

## 5. Conclusions

This work addresses challenges in large-scale *BGP* simulations. Both memory and execution time optimizations are described for the *BGP++* simulator that enable large-scale simulations. Although improvements in scalability have been demonstrated, further optimizations are possible. More efficient *BGP* routing table sharing techniques can be investigated, exploiting the redundancy of routing information in *BGP* simulations. An open research direction is the evaluation of the tradeoff between a relaxation of the partitioning balance constraint, the edge-cut, and the simulation performance.

## Acknowledgments

The authors would like to thank Donghua Xu for providing code to generate pdns configuration from ns-2 configuration, Patrick Verkaik for helpful discussions and providing code to configure *BGP* communities in *BGP++* and Christina Routi for many valuable comments on the manuscript.

## References

- [1] H. Chang, R. Govindan, S. Jamin, S. J. Shenker, and W. Willinger. Towards capturing representative AS-level Internet topologies. *Computer Networks Journal*, 44:737–755, April 2004.
- [2] Cisco Systems. Endless BGP convergence problem in Cisco IOS software releases. Field notice, Cisco Systems Inc, 2000.
- [3] J. H. Cowie, D. M. Nicol, and A. T. Ogielski. Modeling the global Internet. *Computing in Science and Engineering*, Jan. 1999.
- [4] X. Dimitropoulos and G. Riley. Creating realistic BGP models. In *Proceedings of Eleventh International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems (MASCOTS'03)*, October 2003.
- [5] A. Feldmann, H. Kong, O. Maennel, and A. Tudor. Measuring bgp pass-through times. In *Proc. of 5th annual Passive and Active Measurement Workshop*, May 2004.
- [6] T. D. Feng, R. Ballantyne, and L. Trajkovi. Implementation of BGP in a network simulator. In *Proc. of Advanced Simulation Technologies Conference 2004 (ASTC'04)*, April 2004.
- [7] L. Gao. On inferring autonomous system relationships in the internet. In *Proc. IEEE Global Internet Symposium*, Nov. 2000.
- [8] R. Govindan and H. Tangmunarunkit. Heuristics for Internet map discovery. In *IEEE INFOCOM 2000*, pages 1371–1380, Tel Aviv, Israel, March 2000. IEEE.
- [9] F. Hao and P. Koppol. An Internet scale simulation setup for bgp. *Computer Communication Review*, 33(3):43–57, 2003.
- [10] B. Hendrickson and R. Leland. The Chaco user's guide, 1994.
- [11] K. Ishiguro. GNU Zebra. Software on-line: <http://www.zebra.org>.
- [12] k claffy, T. E. Monk, and D. McRobb. Internet tomography. *Nature*, January 1999. <http://www.caida.org/tools/measurement/skitter/>.
- [13] G. Karypis and V. Kumar. *MeTis: Unstructured Graph Partitioning and Sparse Matrix Ordering System*.
- [14] X. Liu and A. A. Chien. Traffic-based load balance for scalable network emulation. In *in Proceedings of the ACM Conference on High Performance Computing and Networking*, November 2003.
- [15] S. McCanne and S. Floyd. The LBNL network simulator. Software on-line: <http://www.isi.edu/nsnam>, 1997. Lawrence Berkeley Laboratory.
- [16] D. Nicol. Scalability of network simulators revisited. In *Proceedings of the Communication Networks and Distributed Systems Modeling and Simulation Conference*, Orlando, FL, February 2003.
- [17] D. M. Nicol. Scalability, locality, partitioning and synchronization in PDES. In *Proceedings of the Parallel and Distributed Simulation Conference (PADS'98)*, 1998. Banff, Canada.
- [18] D. M. Nicol, S. W. Smith, and M. Zhao. Efficient Security for BGP Route Announcements. Technical Report TR2003-440, Dartmouth College, Computer Science, Hanover, NH, May 2003.
- [19] Y. Rekhter and T. Li. RFC 1771, Border Gateway Protocol 4, Mar. 1995.
- [20] G. F. Riley, M. H. Ammar, and E. W. Zegura. Efficient routing using nix-vectors. In *2001 IEEE Workshop on High Performance Switching and Routing*, May 2001.
- [21] G. F. Riley, R. M. Fujimoto, and M. H. Ammar. Parallel/Distributed ns. Software on-line: [www.cc.gatech.edu/computing/compass/pdns/index.html](http://www.cc.gatech.edu/computing/compass/pdns/index.html), 2000. Georgia Institute of Technology.
- [22] University of Oregon Route Views Project.
- [23] N. Spring, R. Mahajan, and D. Wetherall. Measuring ISP topologies with Rocketfuel. In *ACM SIGCOMM*, August 2002.
- [24] L. Subramanian, S. Agarwal, J. Rexford, and R. H. Katz. Characterizing the Internet hierarchy from multiple vantage points. In *Proc. of IEEE INFOCOM 2002, New York, NY*, Jun 2002.
- [25] B. K. Szymanski, Y. Liu, and R. Gupta. Parallel network simulation under distributed Genesis. In *In Proceedings of ACM/IEEE/SCS of Workshop on Parallel and Distributed Simulation (PADS)*, June 2003.
- [26] T. Tannenbaum and M. Litzkow. Checkpointing and migration of UNIX processes in the Condor distributed processing system. *Dr Dobbs Journal*, February 1995.
- [27] J. Xia and J. Hua. Benchmarking and simulation of BGP processing, Dec 2002.
- [28] K. Yocum, E. Eade, J. Degeys, D. Becker, J. Chase, and A. Vahdat. Toward scaling network emulation using topology partitioning. In *Proceedings of Eleventh International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems (MASCOTS'03)*, October 2003.