RESEARCH-ARTICLE

Evaluating OSPF Convergence with ns-3 DCE

NICOLAS RYBOWSKI

OLIVIER BONAVENTURE

WNS3 '22: Proceedings of the 2022 Workshop on ns-3 (June 2022)
https://doi.org/10.1145/3532577.3532597
ISBN: 9781450396516

Evaluating OSPF Convergence with ns-3 DCE

Nicolas Rybowski
UCLouvain (ICTEAM)
Louvain-la-Neuve, Belgium
nicolas.rybowski@uclouvain.be

Olivier Bonaventure
UCLouvain (ICTEAM)
Louvain-la-Neuve, Belgium
olivier.bonaventure@uclouvain.be

ABSTRACT

OSPF routers flood link-state packets to distribute the network topology. When a link fails, the attached routers flood an updated link-state packet. These packets trigger updates to the forwarding tables of potentially all routers inside the network. This process can take from a few hundreds of milliseconds to a few seconds in large networks. The convergence affects the forwarding of data packets as transient micro-loops and black holes can happen while routers update their forwarding tables asynchronously.

We leverage ns-3, DCE and the BIRD open-source implementation of OSPF to develop a ns-3 model of an OSPF router. We propose techniques to detect micro-loops and black holes. We report our experience in simulating the convergence of OSPF. We also evaluate the scalability of our model and discuss further work.

CCS CONCEPTS

• **Networks → Routing protocols**; **Topology analysis and generation**; **Network simulations**; **Packet-switching networks**; **Point-to-point networks**.

KEYWORDS

ns-3, IGP Convergence, OSPF

ACM Reference Format:
Nicolas Rybowski and Olivier Bonaventure. 2022. Evaluating OSPF Convergence with ns-3 DCE. In *Proceedings of the WNS3 2022 (WNS3 2022), June 22–23, 2022, Virtual Event, USA.* ACM, New York, NY, USA, 7 pages. https://doi.org/10.1145/3532577.3532597

1 INTRODUCTION

IP routers use routing protocols such as OSPF, IS-IS or BGP to exchange routing information. When a router starts, it computes its routing and forwarding tables based on the messages received from the other routers in the same network. Enterprise and Internet Service Provider (ISP) networks mainly use link-state routing protocols to distribute the entire network topology. A key benefit of these protocols is that they can react faster than distance-vector protocols whenever a link or router fails. Measurement studies have shown that these failures are frequent in large networks [18, 20, 30]. When such an event occurs, the routers attached to the failed link/node flood a new link-state packet. Each router that receives such a packet updates its copy of the topology and updates its forwarding table after having recomputed its shortest paths.

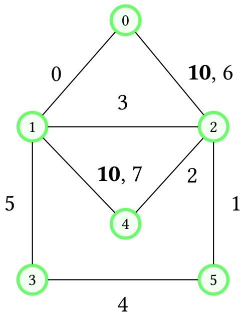

Figure 1: Simple House Network

It is interesting to analyze in more details the network behavior when a link fails. For this, we need to examine two different aspects: the data-plane, i.e., the forwarding of IP packets, and the control-plane, i.e., the exchange of routing messages. Let us consider the simple network shown in Figure 1. The numbers associated with each link are the link IDs. The links weight defaults to 1 except for Links 6 and 7 which have a weight of 10. When the network has converged, Node 3 can reach Node 2 by sending its packets via either Node 1 or Node 5. In practice, today's routers implement Equal Cost Multipath (ECMP) and spread the load over both paths. Let us now analyze the failure of Link 3. To quickly detect the failure, routers can either leverage the underlying physical layer [29] or use the Bidirectional Forwarding Detection (BFD) protocol [19]. In practice, routers usually need a few tens of milliseconds to detect the failure [13]. When Node 1 detects that it is no more directly connected to Node 2, it starts two different recovery processes. First, it can perform a fast reroute in the dataplane by immediately updating its forwarding table to send the packets that were previously forwarded to Node 2 along another path. In this simple example, Node 1 can send its packets to Node 4 and they will reach their final destination. Node 4 is a Loop-Free-Alternate (LFA) of Node 2 [3]. This is one of the simplest fast reroute techniques. Many others have been proposed and deployed [7].

However, the LFA path via Node 4 is a transient path that will last until Node 1 updates its forwarding table. The second, naive, recovery process would be for Node 1 to update its forwarding table as soon as it has detected the failure. After this update, the forwarding table of Node 1 uses Node 3 as a nexthop to reach Node 2. Unfortunately, Node 3 load balances the flows towards this destination via Node 5 and Node 1. It results in a micro-loop that lasts until Node 3 receives the new link-state packet from Node 1 and updates its forwarding table. Despite the deployment of IP

WNS3 2022, June 22–23, 2022, Virtual Event, USA
© 2022 Association for Computing Machinery.
ACM ISBN 978-1-4503-9651-6/22/06…$15.00
https://doi.org/10.1145/3532577.3532597

and MPLS-based fast reroute techniques, these micro-loops are responsible for packet losses that result from remote link failures in large networks [17].

Researchers have proposed a wide range of fast reroute techniques operating in the data-plane [7]. In the control-plane, researchers have analyzed the behaviors of routers [25], the convergence of link-state routing protocols [13, 14] and proposed techniques to mitigate micro-loops [9, 12, 26]. However, these techniques have never been implemented in real routers. This is partially because to experiment with routing protocols, researchers would either use an emulation technique such as Mininet [15] or real testbeds. Emulation of large networks is impractical since all routing protocols would compete for the same CPU. Real testbeds are costly to reproduce network with a hundred routers or more. In this paper, we leverage ns-3 and its Direct Code Execution (DCE) [28] extension to port BIRD [23] on ns-3.

This paper is organized as follows. In Section 2, we first describe how we added BIRD to ns-3. It covers, among others, the router model used in the simulations and different types of failures. In Section 3, we present two measurement methods used to estimate the convergence time. Then, Section 4 reports the convergence time measurements simulations for two different network topologies. In Section 5 we discuss the scalability evaluation of our approach. Finally, Section 6 presents related works.

2 BIRD IN NS-3

In this section, we describe the experimental setup used to simulate different topologies and measure their convergence time under various failures conditions.

To be as close as possible to reality, we leverage DCE [28] to simulate the behavior of a real IGP. In particular, we use the BIRD [23] OSPFv2 implementation with some custom patches (∼25 LoC). They include, among others, user-space sleeps modelling the various internal delays from the network nodes.

We chose BIRD over other open-source IP routing suites - such as Quagga, XORP [16] or FRRouting - because the two first are not maintained anymore since 2018 and 2012, respectively. We gave a first try to FRRouting, an active fork of Quagga. Its integration with DCE required many changes and we experienced livelock issues during simple simulations involving the Zebra daemon. That being said, at the time of writing this document, an ongoing effort tries to port the FRRouting suite on DCE [11].

The integration of BIRD in ns-3 with DCE is quite straightforward. We use the Linux kernel NUSE library [27] to ensure that all the network related functionalities used by BIRD are implemented. Only slight modifications of DCE are required to support BIRD: (*i*) adding the native syscalls *ffs*, *localtime_r* and *longjmp* to the ns-3 libc wrapper and (*ii*) adding the support for private anonymous memory mapping in the `dce_mmap64` syscall wrapper. Other syscall updates might be required to run BIRD code not used in our environment. However, the modifications mentioned above are sufficient for our BIRD usage.

2.1 Network Configuration

We developed a toolchain based on ns-3. It takes as input files describing network topologies and, if required, failures. After parsing them, it constructs the network topology in ns-3 and schedules the failures if any. Then, the program configures a BIRD instance on each node and runs the simulation. Finally, the logs are collected, parsed and interpreted to produce plots.

Topology definition. All the networks described in this document are defined in test files. Each line defines a uni-directional link represented with the (*source*, *destination*, *metric*, *delay*) tuple. The delays are specified in microseconds. This format allows a fine tuning of the network parameters, e.g., by having links with asymmetric costs or redundant links. Failures definition files follow the same format as for the topology definition but with an additional value: *instant of failure*. It is defined as the number of seconds since the simulation start (T_0).

BIRD configuration. The configuration of the BIRD daemon on each node is fully automated. The links weight and cost are defined in the input file specifying the topology. Flags allowing the usage of ECMP or configuring the maintenance timer value are also available in our CLI.

ns-3 setup. All the nodes in the network are interconnected with point-to-point channels[1]. Each interface, loopback included, is addressed with a unique `/32` IPv4 prefix. In order to reduce the amount of control messages exchanged by the IGP, we limit the number of advertised prefixes. To do so, we only distribute loopback addresses and no external routes are defined. The wired interfaces are configured with peer addresses in order to enable unnumbered interfaces [24]. The maximum number of prefixes advertised in a given network thus corresponds to the number of nodes it contains. All the outgoing packets are forced to use the loopback address as their source address. The loopback address also serves as node identifier in the OSPF configuration. Each node belongs to the OSPF backbone area and no other area is defined.

The network startup sequence is the following. At T_0+1s, every node is fully configured. That is, its interfaces are up and addressed and the BIRD instances are configured. From T_0+5s, a BIRD daemon is started every 500ms with an additional delay uniformly distributed on the [0,10]ms interval. This random delay is used to avoid the synchronization of the OSPF timers between the network's nodes.

2.2 Router Model

François et al. [13] define the IGP convergence as the contribution of multiple factors: $D + O + F + SPT + FIB + DD$ where D is the failure detection time, O is the time to originate an LSA, F is the time to flood an LSA to the whole topology, SPT is the shortest-path tree computation delay and DD is the time to update the router's linecards. In this work, both the LSA generation delays and the linecard update time are not modelled. The other contributions model are described in the rest of this section.

OSPF routines are triggered by different timers and delays allowing aggregation of multiple events in a single computation. The ns-3 simulator does not represent internal delays of the nodes, e.g., the ones due to their CPU load or the cost of the operations they perform. Hence, we have to introduce user-space sleeps in

[1]https://www.nsnam.org/doxygen/classns3_1_1_point_to_point_channel.html

BIRD in order to model such delays. They are randomly chosen on a uniformly distributed range. The `rand()` calls in BIRD are intercepted by ns-3 which maintains, for each process, a seeded random variable uniformly distributed on [0,RAND_MAX][2]. This seed is left unchanged for all the experiments presented in Section 4. The boundaries of the delays ranges, taken as is from the work of Francois et al. [13], are specified in Table 1.

Table 1: Internal Node's Delays Parameters

Delay source	Value
LSP processing	$[2,4]ms$
Maintenance timer	$\{10,25,50,100\}ms$
SPT computation	$[2,4]ms$
FIB prefix update	$[100, 110]\mu s$/prefix

SPT computation delay. A common optimization is throttling the computation of the SPT with an exponential backoff strategy [13]. Upon reception of an LSA, an "initial delay" timer is triggered. All LSAs received before its expiration are collected and processed all at once after the expiration. The SPT computation is thus delayed at least by the initial delay.

The BIRD OSPFv2 implementation only supports a fixed maintenance timer[3]. The operations performed after its expiration are, among others, the LSA aging and SPT computation. A flag indicating the need for SPT re-computation is set upon LSA arrival. The SPT computation is thus delayed at most by the maintenance timer.

FIB update. One could naively update each FIB entry after an SPT computation. Now consider a FIB containing hundreds of routes and an event modifying a single one. The full update could take a certain amount of time knowing that the update of a single prefix is not instantaneous. To avoid such unwanted delay, an optimization called *incremental FIB update* [13] is used. This method updates only the routes changed by a network event. It is the strategy used by the BIRD OSPFv2 implementation.

2.3 Modelling Network Failures

We consider two different failure scenarios. First, the classical single-link failure, which remains today the most frequent outage according to Chiesa et al. [7]. Second, an entire node failure. This may be represented as the failure of all its links at the same time. The single-link failure model is reused to that end. For the rest of this document, we define T_F as the time of failure.

Single-link failure model. At T_F, a ns-3 blackhole model dropping all the packets forwarded on a link is applied on both its *PointToPointNetDevice* using the *SetReceiveErrorModel* callback. Then, both interfaces are disabled respectively at $T_F + 15ms$ and $T_F + 18\ ms$, triggering the IGP routines at delayed intervals. Those values are arbitrarily fixed (*i*) to respect the sub-20ms link failure detection time measured by Francois et al. [13] and (*ii*) to allow easier network behavior description after a failure in the next sections. Changing those values to a random variable uniformly distributed on a given interval can be easily achieved as discussed in Section 2.2.

[2]https://github.com/direct-code-execution/ns-3-dce/blob/f2d1902339de95b68f81afecd34c3b1350b7b551/model/dce-manager.cc#L369

[3]https://gitlab.nic.cz/labs/bird/-/blob/master/proto/ospf/ospf.c#L452

3 DETECTING MICRO-LOOPS

We described in Section 2 how a topology running a modern implementation of OSPFv2 is set up in ns-3 and DCE. Now, we consider two different approaches to estimate the convergence time of the IGP after a failure. One is based on constant bit-rate UDP flows and the other on FIB exploration. We detail how micro-loops and black holes are detected with both of them. We also have to take into account the fact that the network could converge transiently. That is, each node shows consistent forwarding at a given time. Then after a new FIB update, black holes or micro-loops appear in the topology, introducing temporary packet loss. Therefore, we consider the *instant of convergence* (IoC) as the last moment at which the forwarding becomes and remains consistent on each node.

In Section 1, we introduced a very simple but interesting network for our tools validation. It contains 6 nodes and 8 bidirectional links as illustrated in Figure 1. The latency of each link is 5ms. This choice of IGP weights and the network connectivity enable various equal-cost paths and many possibilities for micro-loops.

For multiple illustrations in the next sections, we will consider the following situation. For the sake of simplicity, we define that ECMP is disabled. We suppose that Node 3 reaches Node 0 via Node 1 and Node 4 via Nodes 5 and 2. On its side, Node 5 reaches all the nodes via Node 2, except for Node 3 for which its has a direct connection.

3.1 UDP Flows

The idea is to sample the FIB of each node in the network at regular intervals right after T_F. For each pair (*source*, *destination*), we set up a UDP flow whose packets embed a unique timestamp. This timestamp is the EPOCH at which a packet is sent from the source. The flow period P is 5ms. On each receiver, the incoming UDP flows are collected per source node. When the forwarding is consistent, Identity 1 is verified.

$$timestamp_{src,t} = timestamp_{src,t-1} + P \tag{1}$$

Inconsistencies detection. If a blackhole appears in the network, the identity becomes:

$$timestamp_{src,t} = timestamp_{src,t-1} + \Delta x + P \tag{2}$$

with Δx being the range of lost timestamps. In case of micro-loops, either Identity 2 is observed because the packet's TTL reaches 0, or older timestamps finally arrive. The network is considered as being in a normal forwarding state when Identity 1 is verified for at least a given amount of time, e.g., 500ms.

Let us consider our simple example scenario with the failure of Link 3. Once Node 2 has updated its FIB, the UDP packets are sent to Node 5. We define T_X, the timestamp of the first packet sent by Node 2 after its FIB update. Node 5 still uses its FIB from before the failure. It will thus resend the packets to Node 2 and so on. The TTL of such packet is decremented each time it is forwarded by one of the two nodes. If the loop lasts long enough, the TTL reaches 0 and the packet is simply dropped. When Node 5 updates its FIB, it is able to correctly forward the packets to Node 3. It will thus forward a mix of recent packets sent by Node 2 and older ones which previously looped. Consider a loop lasting for $10ms$ and T_Y as the timestamp sent from Node 2 at $T_X + 10ms$. T_X reaches Node 5 at $T_X + 5ms$ and is back on Node 2 at $T_X + 10ms$. Node 3 eventually receives T_Y then T_X. It expects T_X to be equal to $T_Y + 5ms$ but it is not the case. It thus detected a micro-loop.

Method limitation. The major issue of this method is the FIB sampling rate. A node could converge right after the forwarding of a probe packet at T_x. However, the convergence will only be considered at $T_x + P$, leading to an overestimation of the convergence time. We might increase the sampling rate but simulating packets in the network is costly and increases the simulation time. This overhead could strongly limit the size of the networks we can simulate. We thus have a trade-off between the accuracy of the convergence time estimation and the simulation overhead.

The second issue is the potential re-ordering. The IGP metrics might reflect the link delays but it is defined as dimensionless [21]. It means that the UDP flows might be distributed on multiple equal-cost paths, each having a different total delay. This may eventually lead to packet reordering on the receiver side. Such re-ordering would be considered as a micro-loop, according to our definition.

3.2 FIB Traversal

Right after T_F, we dump the FIB of each router when the FIB of any router is updated. Then at each timestamp, we follow for each pair (*source*, *destination*) every equal-cost paths. The goal is to rebuild the paths followed by a packet to reach each destination from each source.

During the post-processing, if any route contains a micro-loop or a black hole, the network convergence is not reached at the current timestamp. Let's take our example scenario and consider the failure of Link 3. When Node 2 detects the failure and updates its FIB, we collect the FIB of each node. We thus have a snapshot of the network FIBs at a given timestamp. First, we build the routes from Node 2 to all the others nodes. In order to reach Node 1, Node 2 uses Node 5 as its nexthop. However, the FIB of Node 5 is still the one from before the failure and thus uses Node 2 as its nexthop to reach Node 1. We thus found a micro-loop at the current timestamp by following the FIBs. Then, we build the routes from Node 1 to all the other nodes at the current timestamp. Since Node 1 detected quickly the failure but did not re-compute its SPT yet, it has no nexthop towards Node 2 currently. We thus found a black hole by exploring the FIBs.

4 SAMPLE SIMULATIONS

In this section, we evaluate the convergence time of two small scaled networks under the failure scenarios described in Section 1. All the experiments described here were run on a Virtual Machine containing 12 cores and 98 GB of RAM. We first reuse the network described in Section 3 and shown in Figure 1. Links 6 and 7 are not part of any route after the initial convergence due to their weight. Therefore, their failure does not impact the convergence time after the outage.

Figure 2 shows the convergence time when ECMP is disabled for each link failure. With the FIB measurement method and the SPF tree delay set to $100ms$, we observe that the convergence oscillates

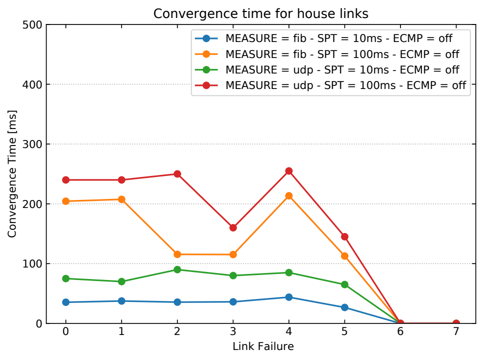

Figure 2: Convergence for House Links after Link Failures (ECMP off)

around 100 and $200ms$. This oscillation is due to the distance between the failure and the rerouting nodes. The maintenance timer synchronization could also impact this oscillation.

For example, during the failure of Link 2, the rerouting node is Node 1, as it is directly connected to both Node 4 and Node 2. Node 1 receives an LSA from Node 2 before the expiration of its own timer, allowing the whole network convergence in around $100ms$ after T_F. Another example is the failure of Link 3 for which we detailed in Section 3 the state of the FIBs. When Link 3 fails, Node 1 detects the failure in less than $20ms$. It then redirects its traffic to Node 3 whose FIB remains unchanged. Node 2, the second node directly attached to the failure, use Node 5 for its rerouting. It is able to send its LSA before the Node's 5 timer expiration, allowing the whole network convergence in a single FIB update on 3 routers.

This synchronization does not happen every time. When we consider the failure of Link 0, Node 3 updates its FIB since it uses Node 1 to reach Node 0. However, in this particular case, the LSA indicating the failure leaves Node 1 at $T_F + 112ms$ and reaches Node 3 around $T_F + 117ms$, while the timer of Node 3 expired around $T_F + 100ms$. It thus has to wait another $100ms$ before recomputing its shortest paths. Hence, the whole network needs two timer expirations before converging. As expected, the UDP measurements follow the same curves as the FIB ones, but with some overhead.

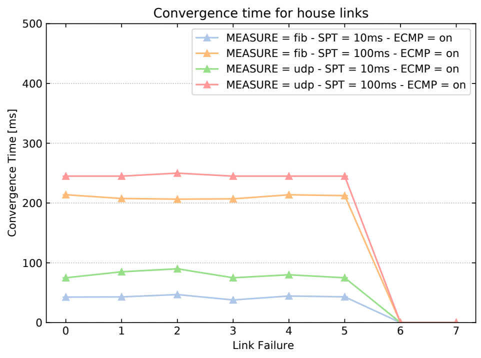

Figure 3: Convergence for House Links after Link Failures (ECMP on)

Figure 3 shows the same simulations but with ECMP enabled. We expect results similar to the ones observed when ECMP is disabled for the failures that required more than one timer expiration for the network convergence. Also, due to the larger route diversity allowed by ECMP, we expect longer convergence delays for nodes which converged previously within only one timer duration. This behavior is indeed observed for example during the failure of Link 3. When ECMP was disabled, the FIB of Node 3 remained unchanged since it did not use Link 3. Now that ECMP is enabled, Node 3 uses both Links 1 and 3 to reach Node 2. We also have seen that the LSAs sent by Node 1 to Node 3 miss the first timer after T_F on Node 3. Hence, with ECMP enabled, rerouting through Node 3 takes one additional timer expiration, delaying the whole network convergence to around 200ms.

For the next experiments, we deactivate the UDP measurement method due to (i) the simulation overhead caused by the UDP packets and (ii) the inaccuracy inherent to the sampling rate. We also only consider networks with ECMP enabled.

4.1 GEANT Network

The second topology is the GEANT network[4] around 2005. This is a European research network connecting the National Research and Education Networks (NREN) from multiple European countries. We use the same topology as the one experimented by Francois et al. [13]. This allows us to compare our results, obtained with a recent simulator and protocol implementation, with the ones measured more than one decade ago.

In 2005, this network contained 22 routers and 36 bidirectional links. 21 of the nodes were located in Europe and the last one in New-York, USA. The continental links had a low delay while the one connecting New-York had a large delay.

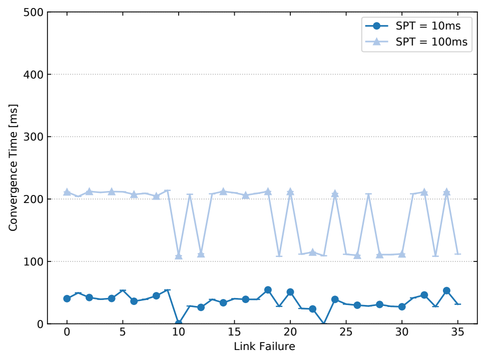

Figure 4: Convergence for GEANT after Links Failure

Links failures. Figure 4 shows the convergence time estimated with the FIB measurement method and the maintenance delay set to {10,100}ms. We see once again an oscillation between 100 and 200ms when the delay is set to 100ms. This is quite different from the results shown by Francois et al. [13]. We explain that because of the different OSPF implementation used. The one leveraged in [13] used an exponential backoff mechanism to delay the SPT computation. Its initial delay, set to {10,100}ms, is started upon reception of an event triggering the SPT re-computation. In contrast, BIRD uses the fixed maintenance timer described in Section 2. Hence, the steps found in our simulations results.

[4]https://network.geant.org/

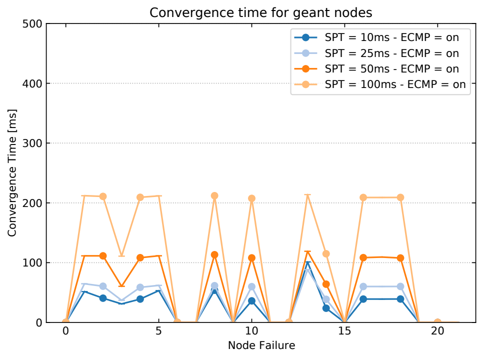

Figure 5: Convergence for GEANT after Node Failure

Nodes failures. Once again we observe oscillations between multiples levels of convergence time and some nodes for which no convergence is required. The latter situation is also noted by Francois et al. [13]. That is, some nodes do not provide transit toward other nodes. Hence, their failures do not impact the global reachability.

We observe that, in general, setting low values for the maintenance timer is not beneficial. For example, the convergence time difference is quite small when the maintenance delay is set to 10 or 25ms. That being said, the number of computations of the SPT will more than double when we decrease the timer value from 25ms to 10ms.

For larger maintenance timer values, we observe that some nodes failures are less costly than others. During such events, the first LSAs received in the topology might be insufficient to fully describe the failure. The nodes thus have to wait all the LSAs before being able to reach their final convergence state. The number of rerouting routers required for a given failure also impacts the convergence duration. It increases with the number of routers.

5 SCALABILITY EVALUATION

In this section, we evaluate the scalability of the simulator for larger network topologies. We only simulate the initial OSPF convergence of the networks for 5 simulated minutes. In practice, we measure the total runtime and the peak memory consumption for two different network topologies with increasing size. We first simulate a simple ring of routers. This topology is the simplest we can consider. It contains n nodes and n bidirectional links. We then simulate a full mesh containing n nodes and $\frac{n\times(n-1)}{2}$ bidirectional links. This topology is the worst case for an IGP. Indeed, each node has $n-1$ neighbors hence each LSA sent during the initial convergence will be duplicated $n-1$ times. We do not consider other topologies since the ones described here respectively give a lower and an upper bound for the measured metrics.

Figure 6 shows the total runtime measured for the ring and full mesh topologies for n linearly increased by 10 in the range [10,100].

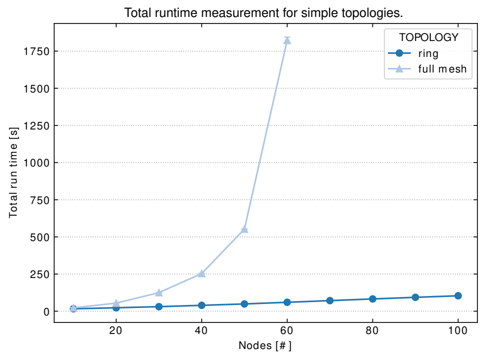

Figure 6: Total Runtime of the Simulation for Two Simple Topologies

This runtime includes the duration of the network setup, i.e., input files parsing, topology setup, BIRD daemon configuration, topology check, etc. We see that for the simple ring, the runtime increases linearly with the number of nodes. When $n = 100$, the runtime does not exceed 2 minutes. However, for the full mesh topology, we observe an exponential curve. For $n = 60$ (1770 links), we measure 30 minutes of simulation and for $n = 70$ (2415 links) we had to stop the simulator after 14 hours. We see that, in terms of runtime, the number of node around a hundred is not really a limit. The real limitation comes from the number of links, and by extension, the number of simulated messages.

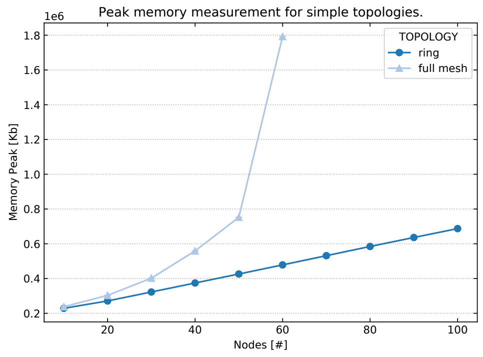

Figure 7: Peak Memory Usage of DCE for the Simulation of Two Simple Topologies

Figure 7 shows the peak memory measured during the simulations performed for Figure 6. Once again, we observe a linear increase, ranging from 250 MB to 750 MB, for the simple ring topology. The full mesh topology presents the same exponential behavior, going up to 1.8 GB of consumed memory.

Those measurements show the lower and upper limits we could expect in terms of link number. Our simulator should thus be usable on bigger topologies than the ones we explored in Section 4, as long as the number of links does not grow exponentially with the number of nodes. That being said, DCE provides different fiber (execution context in the DCE semantic) managers and binary loaders [10]. The experiments presented in this section were executed with the *UcontextFiberManager* (the most efficient fiber manager according to the DCE documentation) and the default *CoojaLoaderFactory*. Using the alternative, but unstable, DCE execution modes might improve the performances observed in this section.

6 RELATED WORK

The convergence of routing protocols has been analyzed by researchers and network operators using different techniques. Network operators often rely on lab measurements to validate the convergence of routers in small network topologies [25]. Thanks to these measurements, it becomes possible to define the key delays for a simulation model. Some operators have also proposed shadow configuration that relies on network namespaces to emulate a network on a single or set of servers [2]. Other emulation techniques include GNS-3 [22] or Mininet [15]. Unfortunately, it is difficult to emulate OSPF convergence on large networks since all emulated routers need to recompute their shortest path trees almost at the same time and thus consume a lot of CPU time.

Discrete events simulators avoid this problem. Unfortunately, the closest, and highly cited related works [5, 13], did not release their simulator to enable other researchers to reproduce their results. The first versions of ns-3 DCE included the Quagga routing daemon [6]. Unfortunately, this port has not been officially maintained. Our port of BIRD brings a complete routing protocol implementation which can be used by researchers within ns-3.

7 DISCUSSION

In this paper, we have ported the BIRD routing version to ns-3 leveraging the Direct Code Execution module. This enables researchers to simulate the convergence of OSPF in large ISP networks. Our module includes two different methods to analyze this convergence: (i) a full mesh of UDP flows and (ii) dumps of the forwarding tables that are processed to identify black holes and micro-loops. We have illustrated their operation on medium size networks and reproduced findings from earlier studies.

Our future work will be to leverage this BIRD module to analyze the performance of new OSPF extensions, compare the performance of OSPF and BGP in datacenter networks [1] or analyze OSPF extensions to prevent micro-loops during a network convergence [8, 12].

ARTEFACTS

The source code of our BIRD integration within ns-3 is available at https://github.com/nrybowski/ns3-sim/tree/wns3-22. Detailed documentation about our toolchain is available in the repository but here is a quick overview of the various elements. The ns-3 simulator is embedded in Docker containers, allowing reproducible results on various platforms. The toolchain entry point is a Rust CLI, automating the configuration and execution of ns-3 instances. It allows the parallel execution of multiple ns-3 instances, each simulating a specific scenario, e.g., a given link failure. The simulator uses a custom ns-3 helper allowing easy topology generation and BIRD daemon configuration. The various graphs shown in this paper are produced with the *NPF* tool [4]. For each of them, we defined a configuration file summarizing the simulations to perform and their parameters. The tool then launches our Rust CLI with the simulation parameters, collects the results and produces the plots.

The artefacts are not yet in a state allowing upstream contribution, but we plan to refactor the code to comply with the requirements of the different upstream projects.

ACKNOWLEDGMENTS

We would like to thank Louis Navarre, Tom Rousseaux, Maxime Piraux and our anonymous reviewers for their accurate reviews which helped us to improve this paper.

REFERENCES

[1] Anubhavnidhi Abhashkumar, Kausik Subramanian, Alexey Andreyev, Hyojeong Kim, Nanda Kishore Salem, Jingyi Yang, Petr Lapukhov, Aditya Akella, and Hongyi Zeng. 2021. Running {BGP} in Data Centers at Scale. In *18th USENIX Symposium on Networked Systems Design and Implementation (NSDI 21)*. 65–81.

[2] Richard Alimi, Ye Wang, and Y Richard Yang. 2008. Shadow Configuration as a Network Management Primitive. In *Proceedings of the ACM SIGCOMM 2008 conference on Data communication*. 111–122.

[3] Alia Atlas and Alex D. Zinin. 2008. Basic Specification for IP Fast Reroute: Loop-Free Alternates. RFC 5286. https://doi.org/10.17487/RFC5286

[4] Tom Barbette. 2022. Network Performance Framework (NPF). https://github.com/tbarbette/npf.

[5] Anindya Basu and Jon Riecke. 2001. Stability Issues in OSPF Routing. *ACM SIGCOMM Computer Communication Review* 31, 4 (2001), 225–236.

[6] Daniel Camara, Hajime Tazaki, Emilio Mancini, Thierry Turletti, Walid Dabbous, and Mathieu Lacage. 2014. DCE: Test the Real Code of your Protocols and Applications over Simulated Networks. *IEEE Communications Magazine* 52, 3 (2014), 104–110.

[7] Marco Chiesa, Andrzej Kamisiński, Jacek Rak, Gábor Rétvári, and Stefan Schmid. 2021. A Survey of Fast-Recovery Mechanisms in Packet-Switched Networks. *IEEE Communications Surveys & Tutorials* 23, 2 (2021), 1253–1301.

[8] Francois Clad, Pascal Mérindol, Stefano Vissicchio, Jean-Jacques Pansiot, and Pierre Francois. 2013. Graceful Router Updates in Link-state Protocols. In *2013 21st IEEE International Conference on Network Protocols (ICNP)*. IEEE, 1–10.

[9] Francois Clad, Stefano Vissicchio, Pascal Mérindol, Pierre Francois, and Jean-Jacques Pansiot. 2014. Computing Minimal Update Sequences for Graceful Router-wide Reconfigurations. *IEEE/ACM Transactions on Networking* 23, 5 (2014), 1373–1386.

[10] DCE community. 2016. *DCE Internals Documentation.* https://ns-3-dce.readthedocs.io/en/latest/how-it-works.html

[11] DCE community. 2021. *FRRouting Integration in DCE.* https://github.com/direct-code-execution/ns-3-dce/pull/134

[12] Pierre Francois and Olivier Bonaventure. 2007. Avoiding Transient Loops During the Convergence of Link-state Routing Protocols. *IEEE/ACM Transactions on Networking* 15, 6 (2007), 1280–1292.

[13] Pierre Francois, Clarence Filsfils, John Evans, and Olivier Bonaventure. 2005. Achieving Sub-second IGP Convergence in Large IP Networks. *ACM SIGCOMM Computer Communication Review* 35, 3 (2005), 35–44.

[14] Mukul Goyal, Mohd Soperi, Emmanuel Baccelli, Gagan Choudhury, Aman Shaikh, Hossein Hosseini, and Kishor Trivedi. 2011. Improving Convergence Speed and Scalability in OSPF: A Survey. *IEEE Communications Surveys & Tutorials* 14, 2 (2011), 443–463.

[15] Nikhil Handigol, Brandon Heller, Vimalkumar Jeyakumar, Bob Lantz, and Nick McKeown. 2012. Reproducible Network Experiments using Container-based Emulation. In *Proceedings of the 8th international conference on Emerging networking experiments and technologies*. 253–264.

[16] Mark Handley, Orion Hodson, and Eddie Kohler. 2003. XORP: An Open Platform for Network Research. *ACM SIGCOMM computer communication review* 33, 1 (2003), 53–57.

[17] Urs Hengartner, Sue Moon, Richard Mortier, and Christophe Diot. 2002. Detection and Analysis of Routing Loops in Packet Traces. In *Proceedings of the 2nd ACM SIGCOMM Workshop on Internet measurment*. 107–112.

[18] Gianluca Iannaccone, Chen-nee Chuah, Richard Mortier, Supratik Bhattacharyya, and Christophe Diot. 2002. Analysis of Link Failures in an IP Backbone. In *Proceedings of the 2nd ACM SIGCOMM Workshop on Internet measurment*. 237–242.

[19] Dave Katz and David Ward. 2010. Bidirectional Forwarding Detection (BFD). RFC 5880. https://doi.org/10.17487/RFC5880

[20] Athina Markopoulou, Gianluca Iannaccone, Supratik Bhattacharyya, Chen-Nee Chuah, Yashar Ganjali, and Christophe Diot. 2008. Characterization of Failures in an Operational IP Backbone Network. *IEEE/ACM transactions on networking* 16, 4 (2008), 749–762.

[21] John Moy. 1998. OSPF Version 2. RFC 2328. https://doi.org/10.17487/RFC2328

[22] Jason C Neumann. 2015. *The Book of GNS3: Build Virtual Network Labs using Cisco, Juniper, and More.* No Starch Press.

[23] Filip Ondrej, Mares Martin, Zajicek Ondrej, and Matejka Jan. 2019. The BIRD Internet Routing Daemon. https://bird.network.cz/.

[24] Zajicek Ondrej. 2014. Unnumbered Interface Configuration in BIRD OSPF. https://bird.network.cz/pipermail/bird-users/2014-June/009118.html.

[25] Aman Shaikh and Albert Greenberg. 2001. Experience in Black-box OSPF Measurement. In *Proceedings of the 1st ACM SIGCOMM Workshop on Internet Measurement*. 113–125.

[26] Mike Shand, Stewart Bryant, Stefano Previdi, Clarence Filsfils, Pierre Francois, and Olivier Bonaventure. 2013. Framework for Loop-Free Convergence Using the Ordered Forwarding Information Base (oFIB) Approach. RFC 6976. https://doi.org/10.17487/RFC6976

[27] Hajime Tazaki, Ryo Nakamura, and Yuji Sekiya. 2015. Library Operating System with Mainline Linux Network Stack. *Proceedings of netdev* (2015).

[28] Hajime Tazaki, Frédéric Uarbani, Emilio Mancini, Mathieu Lacage, Daniel Camara, Thierry Turletti, and Walid Dabbous. 2013. Direct Code Execution: Revisiting Library OS Architecture for Reproducible Network Experiments. In *Proceedings of the ninth ACM conference on Emerging networking experiments and technologies*. 217–228.

[29] Jean-Philippe Vasseur, Mario Pickavet, and Piet Demeester. 2004. *Network Recovery: Protection and Restoration of Optical, SONET-SDH, IP, and MPLS.* Elsevier.

[30] David Watson, Farnam Jahanian, and Craig Labovitz. 2003. Experiences with Monitoring OSPF on a Regional Service Provider Network. In *23rd International Conference on Distributed Computing Systems, 2003. Proceedings.* IEEE, 204–213.