CIDRarchy: CIDR-based ns-3 Routing Protocol for Large Scale Network Simulation

Pedro Moreira da Silva
pmms@inesctec.pt

Jaime Dias
jdias@inesctec.pt

Manuel Ricardo
mricardo@inesctec.pt

INESC TEC, Faculdade de Engenharia, Universidade do Porto
Rua Dr. Roberto Frias, 378, 4200-465 Porto, Portugal

ABSTRACT

ns-3 is the successor of ns-2, the most popular network simulator. Network simulators such as ns-3 play an important role on understanding, designing, and building Internet systems. But simulations are only as good as their models, and the simulation of large scale Internet systems using accurate and complex models is a challenging task. ns-3 simulates realistically the network stack but the scale and complexity of the Internet topology is, from our point of view, limited by the IP forwarding operations.

This work proposes CIDRarchy, an IPv4 routing protocol for ns-3 that uses CIDR as the base to create an hierarchical Internet-like network topology that enables (1) IP forwarding with constant time complexity and automatic IPv4 address assignment, and (2) the implementation of an ns-3 helper to ease network topology creation. We implemented CIDRarchy, evaluated its performance, and obtained simulation time reduction over existing ns-3 routing protocols implementations that can reach over one order of magnitude.

Categories and Subject Descriptors

I.6.5 [**Simulation and Modeling**]: Model Development-Modeling methodologies; I.6.m [**Simulation and Modeling**]: Miscellaneous

General Terms

Simulation, Large Scale, Networks

Keywords

ns-3, CIDR, Routing, Forwarding

1. INTRODUCTION

ns-3 [1] is a free and open-source discrete-event network simulator for Internet systems, mainly targeted for research and educational use. Despite being the successor of ns-2, which is the most popular network simulator for research [9], ns-3 is a replacement rather than an extension of ns-2 as its core was rewritten to improve upon ns-2 limitations.

Simulation plays a vital role on designing, building, understanding, and thoroughly evaluating large scale Internet systems. Large scale Internet systems, in particular those based on Peer-to-Peer (P2P) architecture, such as BitTorrent [4], usually involve thousands or even millions of peers. For systems on such a scale, with heterogeneous peers (e.g. heterogeneous Internet access, CPU, memory, and storage resources), it becomes impracticable to test accurately the designed protocols either analytically or using real, large scale implementations. Small scale tests, although feasible, may not be enough as some issues may only arise at the scale of thousands of peers or more [3].

Simulations are only as good as their models, and the simulation of large scale networks using accurate and complex models is a complex task. When simulating large scale Internet systems, two important models are required: the Internet topology, and the network stack and its protocols. Simulators are usually forced to trade off simulation accuracy for scale [5] because it is hard to evaluate Internet systems over a large and complex Internet topology while using complex and realistic network stacks, and its protocols.

An Internet topology model is a conceptual two-level hierarchical network that connects hosts, routers, and autonomous systems (ASes) to each other in a way that mimics the Internet topology. Given that Internet systems simulation typically does not consider inter-domain (AS-level) topology and its routing, this work targets the intra-domain topology and its routing. We refer the reader to [8] for further reading regarding AS-level topology. We follow the model provided in [2] that presents a conceptual intra-domain network topology model closely related to a real communication network of an Internet Service Provider (ISP). Figure 1 illustrates such model, and depicts the hierarchical structure of such communication networks, which typically consist on access, aggregation, and core sections. The network termination represents hosts.

ns-3 simulates realistically the network stack but the scale and complexity of the Internet topology is limited by the IP forwarding of existing ns-3 routing protocols implementations. Several solutions were proposed to improve ns-3 performance, e.g., resorting to parallel [12] and distributed [10] computing, and avoiding redundant computations [7], but the IP forwarding remains an open issue for large scale networks. ns-3 provides several routing protocols both for infrastructured and ad-hoc networks, the latter being the main subject of research in this area. The two main ns-3 routing

SIMUTOOLS 2015, August 24-26, Athens, Greece

DOI 10.4108/eai.24-8-2015.2261082

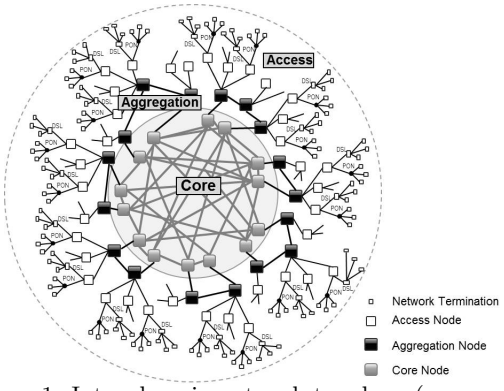


Figure 1: Intra-domain network topology (source: [2]).

protocols for infra-structured networks, `Ipv4GlobalRouting` and `Ipv4NixVectorRouting`, perform IP forwarding table lookup in linear and logarithmic time complexities, respectively. Additionally, ns-3 does not provide a straightforward way to construct a topology with asymmetric links, as Internet access usually is.

This work proposes the use of Classless Inter-Domain Routing (CIDR)[6] as the base to create an hierarchical Internet-like network topology that enables automatic IPv4 address assignment and IP forwarding with constant time complexity, i.e., it does not depend on the IP forwarding table size. Thus, the two main contributions of this work are (1) the implementation of a CIDR-based constant time ns-3 IPv4 routing protocol, and automatic IPv4 address assignment, and (2) the implementation of an ns-3 helper to ease network topology creation (with asymmetric links), and to enable large scale network simulation.

The remainder of this paper is structured as follows. Section 2 presents the main ns-3 routing protocols for infra-structured networks. Section 3 depicts the implementation of an ns-3 helper to ease Internet-like network topology creation. Section 4 describes the implementation of a CIDR-based constant time ns-3 IPv4 routing protocol for Internet systems simulation, which we name CIDRarchy, paired with automatic IPv4 address assignment to nodes. Section 5 provides examples of usage for topology creation, and CIDRarchy install. On Section 6 a performance comparison between CIDRarchy, other routing protocols is provided. Section 7 provides the main conclusions.

2. NS-3 MAIN ROUTING PROTOCOLS FOR INFRA-STRUCTURED NETWORKS

ns-3 simulated nodes are interconnected by one or more `NetDevices` communicating over their respective channels. Packets flow through these channels and, as they traverse the network stack, the actual packet formats are provided to each layer, mimicking real networks. The network stack is configurable per node and, at node creation, it only includes the layers below the IP layer.

`InternetStackHelper` helper provides a simple interface to install the IP stack, and to set the routing protocol that is to be used. Custom routing protocols can also be defined by deriving from `Ipv4RoutingProtocol` class or from `Ipv6RoutingProtocol` class, and must implement, at least, `RouteInput` and `RouteOutput` methods. `RouteInput` method returns a boolean indicating whether it takes responsibility for forwarding or delivering the packet; if so, one of four callbacks, with self-descriptive names, can be invoked to forward or deliver the packet: `UnicastForwardCallback`, `MulticastForwardCallback`, `LocalDeliverCallback`, and `ErrorCallback`. `RouteOutput` method is used by transport protocols for retrieving the route (`Ipv4Route` object) for outgoing packets, if any.

ns-3 provides already several routing protocols both for infra-structured and ad-hoc networks. Typical Internet systems consider only infra-structured networks. Therefore, next we only present the routing protocols provided by ns-3 for infra-structured networks: `Ipv4ListRouting`, `Ipv4StaticRouting`, `Ipv4GlobalRouting` and `Ipv4NixVectorRouting` routing protocols.

2.1 Ipv4ListRouting

`Ipv4ListRouting` is a meta routing protocol that serves the sole purpose of combining different routing protocols in a prioritized list. Routing protocols are invoked in order until an incoming packet is handled (`RouteInput`) or a route is returned for an outgoing packet (`RouteOutput`). I.e., the next routing protocol on the list is invoked only if previous routing protocol either does not handle the incoming packet or does not provide a route for the outgoing packet. The `InternetStackHelper` helper installs by default an `Ipv4ListRouting` enclosing both `Ipv4StaticRouting` and `Ipv4GlobalRouting`, having `Ipv4GlobalRouting` higher priority than `Ipv4Stati-cRouting`.

2.2 Ipv4StaticRouting

`Ipv4StaticRouting` provides a basic set of methods for setting static unicast and multicast routes. Though it can be used as a standalone routing protocol, `Ipv4StaticRouting` was designed to be inserted into an `Ipv4ListRouting` to complement other routing protocols. `Ipv4StaticRouting` is usually used as a lower priority routing protocol for setting default routes.

2.3 Ipv4GlobalRouting

`Ipv4GlobalRouting` is the actual default routing protocol installed by `InternetStackHelper` helper. It walks the simulation topology, and populates node's routing tables with the routes returned by a shortest path algorithm. `Ipv4GlobalRoutingHelper` helper provides two class methods, `PopulateRoutingTables` and `RecomputeRoutingTables`, to populate the routing tables from scratch, and to update routing tables, respectively. Thus, `Ipv4GlobalRouting` is easy to use and, given that both methods can be executed any time during simulation, also supports run-time topology changes.

2.4 Ipv4NixVectorRouting

`Ipv4NixVectorRouting` routing protocol [11] is a more efficient version of global routing that stores source routes in packets headers. The source node adds a set of bits to packet headers that indicate the interface that is to be used at each node to reach destination. The number of bits read by each node is the amount of bits required to represent all its interfaces, being its value the interface index within the node. `Ipv4NixVectorRouting` enables run-time topology changes by recomputing the source routes and flushing caches either implicitly, when interface changes occur, or explicitly, by invoking `FlushGlobalNixRoutingCache` class method.

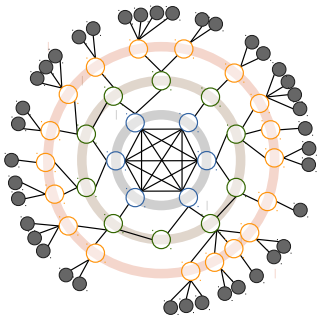


Figure 2: Example of a topology created using `AsymmetricHierarchicalTopologyHelper`. Routers are represented as empty circles; hosts are represented as filled circles.

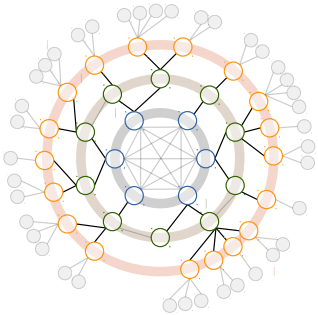


Figure 3: Hierarchy used for setting the subnetwork managed by each router (represented as empty circles).

3. NS-3 HELPER FOR TOPOLOGY CREATION

This section presents the implementation of a novel ns-3 helper, which we name `AsymmetricHierarchicalTopologyHelper`, that enables the creation of topologies akin to the model of ISP networks provided in [2]. All links between nodes are created using `PointToPointChannels` so that uplink and downlink bitrates, delay, and maximum transmission unit (MTU) can be configured per link (uplink and downlink bitrates are configured by setting the transmission bitrate of endpoints' `PointToPointNetDevices`). We describe how the access network is created, explain how the hosts connect to it, and depict how star, tree, and mesh network topologies can be created using `AsymmetricHierarchicalTopologyHelper`.

Following the referred network model, `AsymmetricHierarchicalTopologyHelper` helper enforces a node hierarchy per levels, in which links can only be created between nodes at the same level or at adjacent levels. Routers have only a single link to the router on the level above (parent router), except for core routers (first level) which have none, and there can be only one direct link between any pair of routers. Core routers form a fully connected mesh network so that packets are always forwarded through the minimum hop count route. Additionally, routers either connect hosts (access routers) or connect other routers (aggregation and core routers). Hosts access the network through a single link to an access router, and do not establish any other link with any other network node. Figure 2 depicts an example of a network topology that can be constructed using `AsymmetricHierarchicalTopologyHelper` helper.

Depending on the required level of accuracy, the ISP network can be instantiated as a star network, a tree network or a mesh network. Star network topology can be created using `AsymmetricHierarchicalTopologyHelper` helper by defining a single router to which all hosts connect to, i.e., there is a single level of routers with only one router. A tree topology can be obtained by creating one router at the first level, and adding additional levels with routers as required. Finally, a mesh topology can be obtained either implicitly, by creating several routers at the first level, or explicitly, by creating links between routers at the same level.

4. CIDRARCHY: CIDR-BASED CONSTANT TIME NS-3 IPV4 ROUTING PROTOCOL

This section presents CIDRarchy routing protocol, a novel CIDR-based ns-3 IPv4 routing protocol with constant time complexity. We start by describing how routers manage CIDR prefixes and assign IPv4 addresses to hosts, detail how the algorithm takes the forwarding decisions, and end discussing its implementation. We refer to nodes that have the same parent router as its children, and as siblings of each other.

CIDRarchy routing protocol requires a strict hierarchical assignment of the subnetworks managed by each router to achieve constant time complexity: the subnetworks managed by sibling routers have equal length prefix, and are aggregated in a single CIDR prefix at parent router. Thus, e.g., a router having three children routers, managing the subnetwork `27.2.13.0/24`, would assign the subnetworks `27.2.13.0/26`, `27.2.13.64/26`, and `27.2.13.128/26` to its child routers. This procedure is applied recursively from top to bottom, first by assigning equal CIDR prefixes to each core router, and then by treating each subnetwork as a tree topology: discarding the links between routers at the same level, each subnetwork becomes a tree. Figure 3 illustrates this approach. Access routers sequentially assign IPv4 addresses to hosts within the subnetwork they manage.

The IP packet forwarding decisions taken by CIDRarchy routing protocol can be seen as CIDR prefix adjustments to select an adjacent router that is one hop closer to destination host: the prefix widens (parent link) until the destination host is part of the subnetwork managed by the router, it narrows once it does (child link). Except for core routers, which have no parent link, links to routers at the same level are a shortcut to avoid going up and down the hierarchy. The index of the link to the child node is given by the bits that are part of child's subnetwork prefix but not of parent router's subnetwork prefix. For instance, a router managing the subnetwork `27.2.13.0/24`, that needs to forward a packet to host `27.2.13.66`, will select child with index 1 (the router managing subnetwork `27.2.13.64/26`). The same rationale is applied to routers at the same level, selecting the longest subnetwork prefix as the base (smallest subnetwork), and handling subnetworks with shorter prefix (larger subnetwork) as multiple subnetworks of equal size.

Given that the common behavior of ISP networks is for routers to never generate nor receive packets, and that the access network is static, we implemented CIDRarchy routing protocol in the `Ipv4CidrarchyRouting` class without deriving it from `Ipv4RoutingProtocol` class. `Ipv4L3Protocol`, the class that implements the IP layer, introduces a considerable overhead (1) in the way it processes the routes (`Ipv4Route`) passed by `RouteInput` and `RouteOutput` meth-

ods, and (2) in the way it checks if an IP packet is for local delivery (`IsDestinationAddress` method). `Ipv4Route` object contains the egress `NetDevice` but not the `Ipv4Interface`. For this reason, `Ipv4L3Protocol` iterates over the node's list of `Ipv4Interfaces` to perform a reverse lookup, which introduces overhead that varies with the interface list size. Given that every node is a potential host, when destination address differs from the one of ingress interface, by default, `IsDestinationAddress` iterates over the same list to check if there is a match with the destination address. Therefore, we chose to implement CIDRarchy through a callback registration at `PointToPointNetDevice`, using `SetReceiveCallback` method, and by snooping the IPv4 header to perform the IP packet forwarding.

The `Ipv4CidrarchyHelper` helper provides an `Install` method that receives, besides the network prefix, three node lists as input to perform all network setup operations: core routers, routers, and hosts lists. The list of core routers is required to bootstrap the assignment of the subnetwork managed by each router. The list of routers is required to complete network access setup. The list of hosts enables to automatically assign IPv4 addresses to them. CIDRarchy supports addition of hosts in run-time as long as there are IPv4 addresses available. Hosts can be added during run-time using the class methods `CreateHost` and `CreateNHosts`, which take the same parameters as their counterparts of class `AsymmetricHierarchicalTopologyHelper`.

5. EXAMPLES OF USAGE

This section provides examples of how `AsymmetricHierarchicalTopologyHelper` can be used to create topologies, and how CIDRarchy can be installed. The topologies created in this section are those that are used for comparing CIDRarchy with `Ipv4GlobalRouting`, and `Ipv4NixVectorRouting` in the next section. We first present the topology creation examples, and then the CIDRarchy install example.

5.1 Topology Creation

The topology creation examples show how a star topology, a balanced tree, an unbalanced tree, and a mesh topology can be created using `AsymmetricHierarchicalTopologyHelper`. Listing 1 provides the example for star topology creation, and depicts how default link parameters can be set and how routers and hosts can be created.

Listing 1: Star Topology Creation

```
AsymmetricHierarchicalTopologyHelper helper;

// downlink, uplink, link delay, and MTU
helper.SetRouterLinkDefaults(
  "100Gbps", "100Gbps", "5us", 1500
);
helper.SetHostLinkDefaults(
  "30Mbps", "3Mbps", "5ms", 1500
);

Ptr<Node> root = helper.CreateRouter(0);
helper.CreateNHosts(1000, root);
```

Listing 2 encloses the code for creating a balanced tree topology with two router levels and two branches (left and right). We omit the `AsymmetricHierarchicalTopologyHelper` declaration, and show that link creation functions can also receive the link parameters as arguments.

Listing 2: Balanced Tree Topology Creation

```
Ptr<Node> root = helper.CreateRouter(0);
Ptr<Node> left = helper.CreateRouter(
  root, "100Gbps", "100Gbps", "5us", 1500
);
Ptr<Node> right = helper.CreateRouter(root);

NodeContainer leftHosts = helper.CreateNHosts(
  500, left, "30Mbps", "3Mbps", "5ms", 1500
);
NodeContainer rightHosts = helper.CreateNHosts(
  500, right, "30Mbps", "3Mbps", "5ms", 1500
);
```

The code for creating an unbalanced tree topology with two levels of routers, two branches, and only two hosts on the right branch is depicted in Listing 3. This code also shows multiple router creation, and single host creation.

Listing 3: Unbalanced Tree Topology Creation

```
Ptr<Node> root = helper.CreateRouter(0);
NodeContainer access = helper.CreateNRouters(
  root, "100Gbps", "100Gbps", "5us", 1500
);

NodeContainer leftHosts =
  helper.CreateNHosts(998, access.Get(0));
NodeContainer rightHosts;

rightHosts.Add(helper.CreateHost(
  access.Get(1), "30Mbps", "3Mbps", "5ms", 1500
));
rightHosts.Add(
  helper.CreateHost(access.Get(1))
);
```

Listing 4 depicts the code used for creating a mesh topology with five core routers, two access routers per core router (10 access routers in total), and the same number of hosts per access router. Figure 4 illustrates all four topologies.

Listing 4: Mesh Topology Creation

```
NodeContainer core = helper.CreateNRouters(5, 0);
NodeContainer access;

for(uint32_t i = 0; i < core.GetN(); ++i)
 access.Add(helper.CreateNRouters(2,core.Get(i)));

for(uint32_t i = 0; i < access.GetN(); ++i)
 helper.CreateNHosts(10, access.Get(i));
```

5.2 CIDRarchy Routing Protocol Install

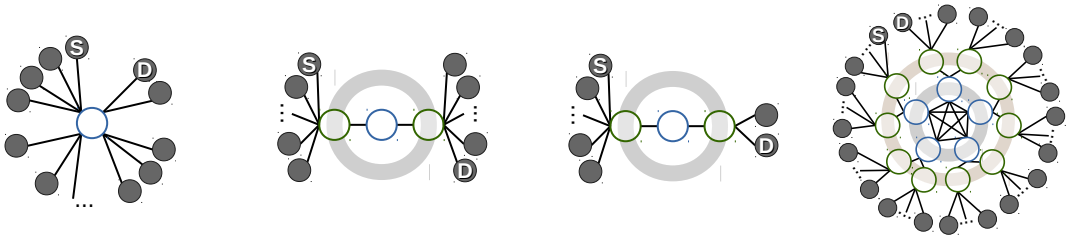
Installing CIDRarchy routing protocol on nodes is as simple as invoking the `Install` method provided by `Ipv4CidrarchyHelper`. The method takes a list of core routers, a list of routers, a list of hosts, a network prefix, and its length as shown in Listing 5.

Listing 5: CIDRarchy Install

```
NodeContainer core = helper.GetCoreRouterNodes();
NodeContainer routers = helper.GetRouterNodes();
NodeContainer hosts = helper.GetHostNodes();
Ipv4CidrarchyHelper cidr;
Ipv4Address netwAddr = "10.0.0.0";

cidr.Install(core, routers, hosts, netwAddr, 16);
```

The `Install` method fails if the subnetwork managed by any router is empty, i.e., contains less than two IPv4 addresses (/31 prefix).



(a) star (b) balanced tree (c) unbalanced tree (d) mesh

Figure 4: Topologies used for performance comparison. The nodes S, and D are the hosts used to install `OnOffApplication` and `PacketSink` applications, respectively.

6. RESULTS

In this section we compare CIDRarchy, `Ipv4GlobalRouting`, and `Ipv4NixVectorRouting` routing protocols with respect to their performance and scalability, with the main objective of evaluating the impact of the IP forwarding functionality on simulation time. For that purpose, first we measured the time required to simulate the transmission of a single traffic flow consisting of 150000 packets, each packet having a size of 1000 bytes, from a node running `OnOffApplication` application (host `S`) to a node running `PacketSink` application (host `D`), using an increasing number of hosts. Then, in a second study and in order estimate the overhead of ns-3 routing protocols bootstrap operations, we used a balanced tree topology with 1500 hosts on each branch (3000 hosts total) and measured the time required to simulate the transmission of multiple flows of 1500 packets, each packet having a size of 1000 bytes, from nodes on the left branch to nodes on the right branch, using an increasing number of flows. The results for single and multiple flows studies are shown, respectively, on Figures 5 and 6, and are the average of 5 simulation runs on an Intel Xeon E5-2650@2GHz CPU, running on Ubuntu 14.04, using ns-3.22 version.

For single flow scenarios, depicted on Figure 5, CIDRarchy presents similar simulation times as the number of hosts increases from 100 to 3000; the slight increase can be explained by the additional computations required for network creation. On the other hand, despite the amount of packets exchanged remains the same, `Ipv4GlobalRouting` and `Ipv4NixVectorRouting` show a significant increase on simulation times as the number of hosts increases. CIDRarchy simulation time increases slightly as the hop count increases showing best results for the star scenario (Figure 5a), which has a hop count of 2. `Ipv4GlobalRouting` and `Ipv4NixVectorRouting` simulation times decrease in the mesh scenario (Figure 5d), which has hop count of 5. This behavior shows that the performance of both routing protocols varies across different topologies, and also that IP forwarding has a significant impact on simulation time: simulation times of mesh scenario are three or four times lower than for other scenarios albeit having higher hop count. On all four scenarios, the simulation times of CIDRarchy routing protocol are much lower than those of other routing protocols, and the gains can reach over one order of magnitude.

Figure 6 shows the average simulation time per flow as the number of simulated flows increases, which is obtained by dividing the total simulation time by the number of transmitted flows. The average simulation time per flow remains nearly constant for both CIDRarchy and `Ipv4NixVectorRouting`, while it decreases for `Ipv4GlobalRouting` from

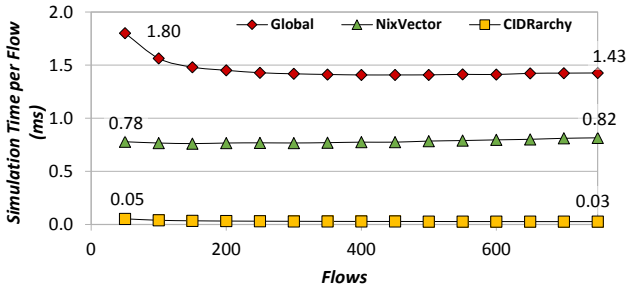


Figure 6: Simulation time per flow required by CIDRarchy, `Ipv4GlobalRouting` and `Ipv4NixVectorRouting` routing protocols to transmit flows of 1500 packets, each having a size of 1000 bytes, using `OnOffApplication` (left branch) and `PacketSink` (right branch) applications, in a balanced tree topology with 1500 hosts in each branch (3000 hosts total), for an increasing number of flows.

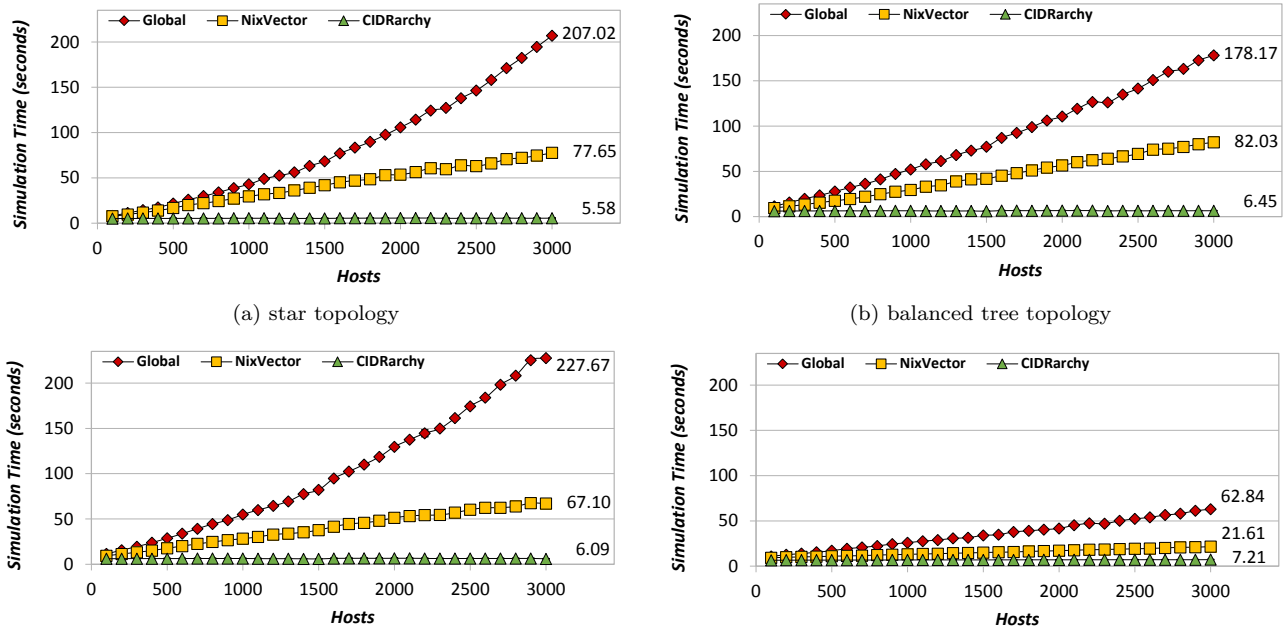
1.80 ms per flow (50 flows) until it stabilizes at around 1.43 ms per flow (250 flows). This shows that the initial cost of populating the `Ipv4GlobalRouting` routing tables is not negligible when considering short simulations. Despite considering only scenarios where each host communicates with no more than one host, which enables `Ipv4NixVectorRouting` to perform IP forwarding having only a single entry on its routing table and not to incur on any penalty that may arise from on-demand route calculation, CIDRarchy still outperforms it by at least one order of magnitude no matter the number of flows simulated.

7. CONCLUSIONS

CIDRarchy routing protocol enables ns-3 to simulate large Internet networks by reducing greatly the time spent in IPv4 forwarding. We implemented and evaluated the proposed routing protocol in ns-3 simulator, and demonstrated that the simulation time gains over existing routing protocols can reach over one order of magnitude. For instance, a simulation taking one month may be complete in less than four days using our proposed CIDRarchy. We provide also an helper that enables complex topology creation with a few lines of code. The implementation provided supports the addition of hosts during simulation time.

8. ACKNOWLEDGMENTS

The authors express their gratitude to the BEST CASE project (”NORTE-07-0124-FEDER-000056”, ”NORTE-07-01



(a) star topology

(b) balanced tree topology

(c) unbalanced tree topology

(d) mesh topology

Figure 5: Simulation time required by CIDRarchy, `Ipv4GlobalRouting` and `Ipv4NixVectorRouting` routing protocols to transmit 150000 packets, each packet having a size of 1000 bytes, using `OnOffApplication` and `PacketSink` applications, in star, balanced tree, unbalanced tree, and mesh scenarios with an increasing number of hosts.

24-FEDER-000058" and "NORTE-07-0124-FEDER-000060") financed by the North Portugal Regional Operational Programme (ON.2 – O Novo Norte), under the National Strategic Reference Framework (NSRF), through the European Regional Development Fund (ERDF).

This work was financed by the FCT – Fundação para a Ciência e a Tecnologia (Portuguese Foundation for Science and Technology) within project UID/EEA/50014/2013, and under the fellowship SFRH/BD/69388/2010.

9. REFERENCES

[1] The ns-3 network simulator. `http://www.nsnam.org/`.

[2] A. Betker, I. Gamrath, D. Kosiankowski, C. Lange, H. Lehmann, F. Pfeuffer, F. Simon, and A. Werner. Comprehensive topology and traffic model of a nationwide telecommunication network. *Optical Communications and Networking, IEEE/OSA Journal of*, 6(11):1038–1047, November 2014.

[3] L. Cheng, N. Hutchinson, and M. Ito. P2PNet: A simulation architecture for large-scale P2P systems. In H. Labiod and M. Badra, editors, *New Technologies, Mobility and Security*, pages 567–581. Springer Netherlands, 2007.

[4] B. Cohen. Incentives build robustness in BitTorrent, 2003.

[5] K. Eger, T. Hoßfeld, A. Binzenhöfer, and G. Kunzmann. Efficient simulation of large-scale P2P networks: Packet-level vs. flow-level simulations. In *Proceedings of the Second Workshop on Use of P2P, GRID and Agents for the Development of Content Networks*, UPGRADE '07, pages 9–16, New York, NY, USA, 2007. ACM.

[6] V. Fuller and T. Li. Classless Inter-domain Routing (CIDR): The Internet address assignment and aggregation plan, August 2006. RFC4632.

[7] K. Harrigan and G. Riley. Simulation speedup of ns-3 using checkpoint and restore. In *Proceedings of the 2014 Workshop on ns-3*, WNS3 '14, pages 7:1–7:7, New York, NY, USA, 2014. ACM.

[8] Y. He, G. Siganos, and M. Faloutsos. Internet topology. In R. A. Meyers, editor, *Encyclopedia of Complexity and Systems Science*, pages 4930–4947. Springer New York, 2009.

[9] S. Khan, B. Aziz, S. Najeeb, A. Ahmed, M. Usman, and S. Ullah. Reliability of network simulators and simulation based research. In *Personal Indoor and Mobile Radio Communications (PIMRC), 2013 IEEE 24th International Symposium on*, pages 180–185, Sept 2013.

[10] J. Pelkey and G. Riley. Distributed simulation with mpi in ns-3. In *Proceedings of the 4th International ICST Conference on Simulation Tools and Techniques*, SIMUTools '11, pages 410–414, ICST, Brussels, Belgium, 2011. ICST.

[11] G. F. Riley, M. H. Ammar, and R. Fujimoto. Stateless routing in network simulations. In *in Proceedings of the Eighth International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems*, 2000.

[12] B. P. Swenson and G. F. Riley. Simulating large topologies in ns-3 using BRITE and CUDA driven global routing. In *Proceedings of the 6^{th} International ICST Conference on Simulation Tools and Techniques*, SimuTools '13, pages 159–166, ICST, Brussels, Belgium, 2013. ICST.