Robust Machine Learning-enabled Routing for Highly Mobile Vehicular Networks with PARRoT in ns-3

Cedrik Schüler, Manuel Patchou, Benjamin Sliwa, Christian Wietfeld
Communication Networks Institute
TU Dortmund University, Germany
firstname.lastname@tu-dortmund.de

ABSTRACT

As implied by the high grade of relative mobility, the inherent network topology dynamics render aerial and ground-based vehicular mesh routing a highly challenging task. Since existing protocols are often not able to timely adopt their decision-making to the actual network conditions, they fail to provide reliable and efficient data delivery mechanisms. In this paper, we present the ns-3 integration of Predictive Ad-hoc Routing fueled by Reinforcement learning and Trajectory knowledge (PARRoT), a novel reinforcement learning-enabled routing protocol that integrates knowledge about the future motion of the mobile agents into the routing process.

CCS CONCEPTS

• **Networks** → *Routing protocols*; *Network simulations*; • **Computing methodologies** → *Machine learning*.

KEYWORDS

Mobile vehicular networks, routing protocol, reinforcement learning

1 INTRODUCTION

Infrastructure-less communication presents a research field of keen interest. Although the Wireless Fidelity (WiFi)-based ad-hoc communication is a long encountered example and is also anchored in the 802.11s [5] standard, upcoming wireless communications also demand a decentralized approach, as shown in the efforts of the 3rd Generation Partnership Project (3GPP) in their Cellular Vehicle-to-Everything (C-V2X) Mode 4 Rel. 14 [1]. However, the opportunities given by spontaneously deployable Mobile Ad-hoc Networks (MANETs) in terms of network provisioning and remote sensing in disaster situations [4][27], the integration of Unmanned Aerial Vehicles (UAVs) in hybrid Intelligent Transportation Systems (ITSs), and, future applications, such as 5th Generation of Mobile Networks (5G) ad-hoc campus networks, motivate further research. With the lack of a centralized coordination instance, robust and reliable routing becomes one of the most important tasks in MANETs, as all nodes in the network need to organize themselves, collaborate for a common goal, and face challenging conditions. As high grades of mobility and hybrid types of vehicles lead to a dynamically changing network topology with small coherence times, a fast adaption is needed. Reinforcement learning-based routing, as initially proposed in [8], omits on complex modeling of the various influxes of routing, but, rather learns from (multiple) abstract metrics [18] and context information, gained by following an anticipatory networking paradigm [9].

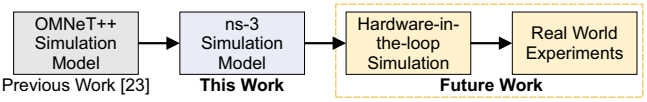

Figure 1: Overall Vision of the PARRoT Evaluation Methods

In [23], we initially presented PARRoT as a novel machine learning-enabled routing method which takes into account information about the predicted future relative mobility of the network nodes. In an in-depth simulation campaign in Objective Modular Testbed in C++ (OMNeT++), PARRoT was able to provide significantly more reliable communication paths than established routing protocols even in challenging channel situations. However, for realizing our overall methodological vision (see Figure 1), it was found that ns-3 provided a better pathway to the planned Hardware-in-the-Loop (HIL) simulations and future real-world experiments than the OMNeT++ methodology. The scope of this paper is therefore to migrate the simulation model to the ns-3 environment, discuss the required changes, and provide a behavior analysis of PARRoT in ns-3.

The contributions are summarized as follows:

- Presentation of the **ns-3 implementation** of the PARRoT routing protocol[1]
- **Parameter optimization** for PARRoT within ns-3
- **Performance analysis** of PARRoT networks in ns-3 and OMNeT++

The remainder of the paper is structured as follows. After discussing the related work in Section 2, we present the implementation of the routing protocol in ns-3 and point out structural differences to OMNeT++, considered in the migration process, in Section 3. Afterwards, an overview about the methodological aspects, the simulation setup and constraints is given in Section 4. Finally, detailed results of the parameter optimization and performance analysis are provided in Section 5.

2 RELATED WORK

Topology-based MANET routing protocols are classified into reactive and proactive approaches. Ns-3 provides implementations of Ad-hoc On-demand Distance Vector (AODV)[20] and Dynamic Source Routing (DSR)[15] as reactive, and Optimized Link State Routing (OLSR)[11] and Destination-Sequenced Distance Vector (DSDV)[19] as proactive implementations. Greedy Perimeter Stateless Routing in Wireless Networks (GPSR)[16] is a geo-based protocol, which ns-3 interpretation is presented in [13].

[1] ns-3 simulation model: https://github.com/cedrikschueler/PARRoT_ns3

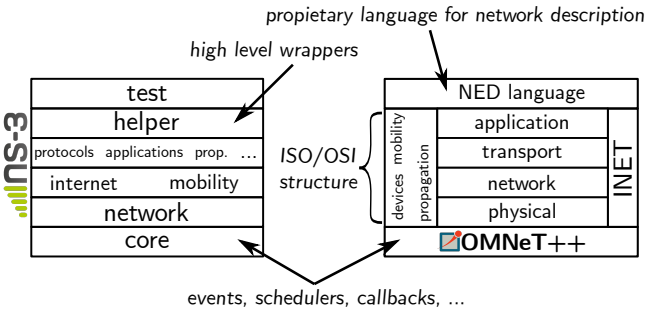

Figure 2: Comparison of the ns-3 and the OMNeT++ Stacks

Reinforcement learning-based routing protocols have evolved from classical Q-routing [8] to deep learning-based concepts, derived from current developments in artificial intelligence research, as [14], [2], and [24]. [25] proposes an approach to feed Q-learning with mobility, link, and energy information. A comprehension of Machine Learning (ML)-enabled networking is given in [7]. [3] presents a survey on routing protocols for UAVs networks. Ad-hoc communications for upcoming 5G and beyond technologies are discussed in [12] and [17].

3 IMPLEMENTATION OF REINFORCEMENT LEARNING-BASED PARROT ROUTING IN NS-3

In this section, the structure of routing protocols in ns-3 and implementation aspects of PARRoT will be discussed. Afterwards, we briefly compare ns-3 and the *INET* stack in OMNeT++ as depicted in Figure 2.

3.1 Routing Protocol Structure in ns-3

Existing MANET routing protocols in ns-3, such as AODV, DSDV, and OLSR, are subclasses of `Ipv4RoutingProtocol` in their corresponding namespaces. Although an implementation of an `Ipv6RoutingProtocol` is located in the ns-3 main branch, none of the established MANET protocols has been transferred to support *Ipv6* yet. However, `Ipv4RoutingProtocol` serves as a abstract base class, closely related to the Linux' equivalent routing functions, and requires interfaces for:

- `RouteOutput` – A function that is responsible for the route lookup for an outgoing packet.
- `RouteInput` – A function that is signalling, if an incoming packet will be processed or not.
- `Notify{InterfaceUp, InterfaceDown, NotifyAddAddress, NotifyRemoveAddress}` – Interfaces, to notify the routing protocol about changes of the handled interfaces and addresses.

Routing protocols in ns-3 model an entity within the Internet Protocol (IP) stack, which is installed onto a `Node` and utilizes sockets, to approach the underlying `NetDevices`, representing Network Interface Cards (NICs). A node is further complemented by a mobility algorithm, additional applications, and protocol stacks. The nodes are connected via channels, on which physical effects as delay, noise, and propagation are modeled, and thereby, form the network. Figure 3 shows the structure of a node and the encapsulation of the routing protocol within the internet stack.

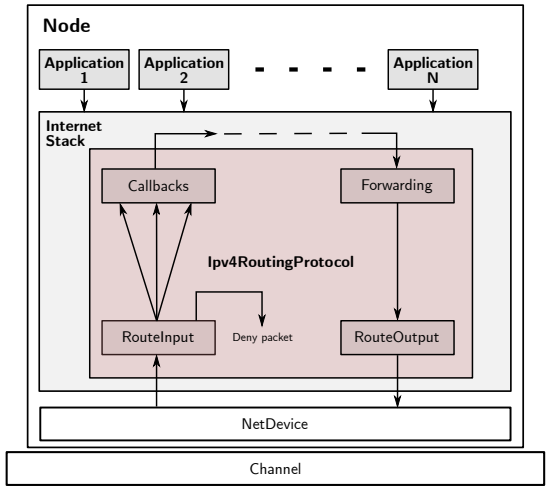

Figure 3: A Network Node with a `Ipv4RoutingProtocol`

The interface function `RouteOuput` provides all routing protocols a certain degree of freedom in terms of route selection algorithms, but, however, requires the protocol to return a generic representation of an `Ipv4Route`, which is described by a tuple < destination address `dst`, source address `src`, gateway `gw`, device `dev` >. Usually, the routing protocols maintain a routing table, in which knowledge about the network topology is stored, and, used to look up the best available route to the packet's destination. Additionally, the protocols can trigger actions like queuing or route discovery at this point, if no route is available. The interface function `RouteInput` comes along with different callback functions for uni- and multicast forwarding, local delivery, and errors that are passed in. The protocol decides, whether it is capable and responsible for processing the incoming packet, and upon its decision, the corresponding callback is invoked and the packet is accepted or declined.

While these functions accomplish the forwarding mechanism of packets, all routing protocols demand the periodic and/or aperiodic exchange of information about the current network topology. For this purpose, a dedicated socket is bound to a *protocol port*, listens to all incoming messages, and redirects them to a callback function. If a node receives a routing broadcast from another node, it evaluates the transmitted metrics within the callback function and triggers further processing such as forwarding the routing message. An example propagation of routing messages is given on the example of PARRoT in Figure 5.

3.2 PARRoT Setup

The initial proposal of PARRoT [23] was developed in OMNeT++ and, besides rich parameterization options, consists of three core components, whose implementation aspects are discussed in the following. The overall architecture adapted to ns-3 is shown in Figure 4.

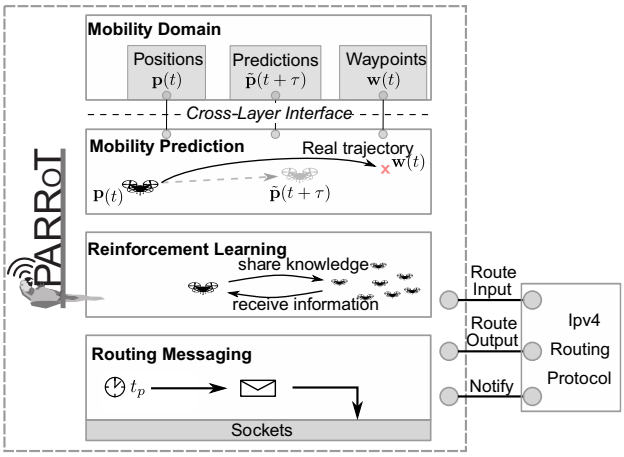

Figure 4: Architecture of PARRoT's ns-3 Implementation

3.2.1 Parameterization and Cross-layer Referencing in PARRoT. As introduced in Section 3.1, PARRoT is also implemented in the IP stack. The routing interfaces `RouteOutput` and `RouteInput` trigger a lookup on the routing table that is stored by the node and fed by a greedy selection from a corresponding Q-table. The Q-table holds route-dependent metrics in a PARRoT Destination Element (PDE) struct for each concerned (destination, gateway)-pair. Additionally, as required by the used metrics for the Q-learning algorithm, the node keeps track of its neighbors and stores its sequence number, position, predicted position, and a timestamp of last contact, in a PARRoT Neighbor Element (PNE). PARRoT is implemented as a proactive protocol. Its main parameter regarding the communication behavior is the periodic update interval t_p for broadcasting routing messages, which we refer to as chirps. The `PredictionWidth` τ describes not only the scope of the mobility prediction but, also the maximum hold time for local routing entries. This parameter is usually a multiple of t_p. Regarding the mobility prediction, an estimated communication range r_c needs to be specified. It can be calculated according to a free space path loss, respecting the radio properties, an attenuation coefficient η, and the receiver sensitivity. While this parameter describes the outer edge of the communication range, the range budget r_b can adjust the estimation with an offset. The last parameter of the mobility prediction is the prediction method, where PARRoT supports a history-based and a waypoint-aware method. However, PARRoT utilizes a cross-layer reference to the mobility application, to make predictions. Finally, the reinforcement learning update rule

$$Q(d,j) = Q(d,j) + \alpha \left[\gamma_0 \cdot \Phi_{\mathrm{LET}}(i,j) \cdot \Phi_{\mathrm{Coh}(j)} \cdot V_j - Q(d,j)\right], \quad (1)$$

that depends on the destination d, the value V_j for a neighbor j, the cohesion $\Phi_{\mathrm{Coh}(j)}$ and the Link Expiry Time (LET) $\Phi_{\mathrm{LET}}(i,j)$ between the current node i and j, can be fine-tuned by the learning rate α and a constant discount factor γ_0. A default parameterization will be carried out in Section 5.1.

3.2.2 Utilization of Trajectory Knowledge. PARRoT agents share knowledge about their predicted position $\tilde{\mathbf{p}}(t+\tau)$ to anticipate the future topology in their local scope. To achieve accurate predictions, knowledge from the mobility domain is leveraged. We introduce three different stages of cross-layer interaction with the mobility control algorithm:

- Low: Only position data is provided. The agent stores the position history independently and extrapolates the future position.
- Medium: Besides the historical positions, that are still maintained as a backup, the agent receives information about the next planned waypoints, enabling an iterative prediction process with higher accuracy.
- High: We also introduce the concept of modern mobility algorithms, which are considered experts for their domain. Therefore, we aim to remove the prediction process from the routing agent and hand it back to the mobility domain. This enables the prediction to take into account more advanced behavior influences, such as swarm coherence [22] for UAVs and platooning and intersection-awareness for vehicles. The predicted position $\tilde{\mathbf{p}}(t+\tau)$ is then provided by an interface for the routing algorithm.

3.2.3 Distribution of Routing Messages. To keep the agent's knowledge up-to-date, all agents schedule chirps, which are the routing messages in PARRoT, and are broadcasted after every expired time interval t_p. The messages contain the address of their originator, the current and predicted position, two metrics, a Time to Live (TTL), and a sequence number, which in total produce 40 Byte of routing overhead to be shared among the agents. Messages in ns-3 are implemented as `Header` subclasses, that require an interface for serialization and deserialization into and from a Byte `Buffer` respectively.

Initially, the reward metric of the chirp is set to the value of 1.0. The sequence number is a counter, managed by the agent, and incremented after every periodic chirp. The sequence number and the originator's address are never changed when other agents forward the message.

Upon the reception of a chirp, the neighbor information is updated, the actuality and information value is assessed, and reinforcement learning is performed. Afterward, the agent checks, whether the gateway over which the chirp was received, is considered as the best gateway to reach the originator d. If this is not the case, we assume that the other (better) route has already been propagated earlier. Otherwise, the agent learned a better routing policy and needs to share its knowledge. Though, the information of the received chirp is updated, setting the reward field V to the maximal value at the agent

$$V = \arg\max_j Q(d,j). \quad (2)$$

The position fields and $\Phi_{\mathrm{Coh}(j)}$ metric are overwritten by the agent's data and the TTL is decremented before forwarding the chirp. This algorithm produces the propagation of chirps through the network, sharing the best effort estimation as a reward metric V. Based on the received information and learned knowledge, routes are built based on the reverse paths of incoming chirps, as depicted in Figure 5 for a route from node S to node D.

3.2.4 Autonomous Routing Decisions. PARRoTs reinforcement learning system relies on Q-learning and yields an abstract metric $Q(d,j)$,

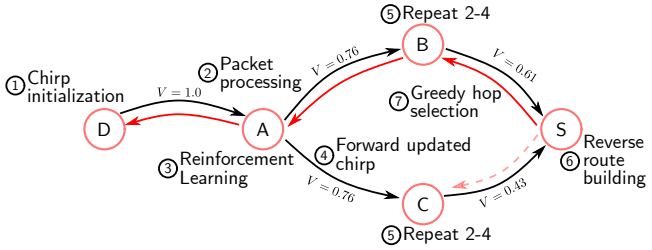

Figure 5: Example Route Building Process

according to Equation 1, that rates the quality of a neighbor j for a requested destination d and is updated in online learning. Agents estimate a LET for a forwarding candidate, by spectating their own and the candidate's trajectory, which is known from received chirps, and, thus, getting the metric Φ_{LET}. Besides a contribution to the reinforcement learning, the assumed availability of all entries, concerning this neighbor is expressed by $\min(\tau, \Phi_{\text{LET}})$. Therefore, a longer persisting valid route to a destination via a specific forwarder needs to be re-acknowledged by another incoming update. The further quality of a forwarder is described by Φ_{Coh} [18], which is a representation of the agents' cohesion. Sets of current and previous neighbors are maintained in PNEs by each agent and evaluated for their quantity ratio. The learning process includes the hyperparameters α and γ_0. The former describes the rate of adaption and implicitly weights the importance of newly incoming packets. The latter is of special interest for situations, in which all incorporated metrics become 1.0, as is guarantees the degradation of multiple hops along a path, and, thus, prevents the creation of routing loops.

3.3 Routing Protocol Migration Among Simulators

While OMNeT++ represents a general purpose simulation environment, the IP stack as well as the routing protocol implementations are provided by the *INET*-based [6] *INETMANET* [21] framework. For better readability, the term *OMNeT++* is used in the remainder of this manuscript for describing the resulting overall simulation system.

A major difference is the usage of proprietary description languages in OMNeT++, while ns-3 omits those and is a pure `C++` framework. The used Network Description (NED) language addresses a scalable way of describing components, interfaces, modules, and networks while giving full support for inheritance and hierarchical nesting of submodules. The so assembled modules retain their functionality by underlying `C++` implementations. Another difference to ns-3 is the `.msg`-language, which also allows an abstract description of packets and provides automatic (de-)serialization functions, and generates corresponding source files. The third difference is the way of study parameterization, which is realized through `.ini` initialization files. OMNeT++ allows the definition of extensible scenarios, easy-to-use parameter studies, and an automatic run configuration for parallel execution.

Another difference in structure is the strict adherence to the layer ordering in the *INET* framework, which is based on the ISO/OSI reference model. The routing protocol works directly with the network layer and is only responsible for handling the referenced routing table. This routing table is in turn connected to the forwarding process. Thus, compared to ns-3, an abstract `Ipv4RoutingTable` also exists, but it is only modified with entries from the routing protocol and does not require the provision of interfaces such as `RouteInput` and `RouteOutput`.

4 METHODOLOGY

In this section, we describe the simulation setup that was used for the evaluation. As we aim to compare the PARRoT implementation in both simulators, the setup was oriented to the parameterization that we used in [23]. As a reference scenario, we look at a three-dimensional UAV scenario with 10 agents and a 2 Mbps constant bit rate User Datagram Protocol (UDP) stream between two agents. A start phase of 5 s passes before the traffic flow starts, to allow the routing protocols to build up their routes. The mobility pattern is a controlled waypoint model, which equals the well known random waypoint mobility, but, with the extension, that the next targeted waypoint can be provided, which is utilized by the mobility prediction of PARRoT as described in Section 3.2.2. The available ns-3 `FriisPropagationLossModel` is modified to calculate the path loss L with an exponential loss coefficient of $\eta = 2.75$. In the error model, the transmission power, and receiver sensitivities are aligned to OMNeT++'s default parameters. All parameters are summarized in Table 1.

Table 1: Default Parameters of the Evaluation Setup

Parameter	Value
Runs	25
Simulation time	900 s
Start phase duration	5 s
MAC	802.11g
Bit Error model	NistErrorModel
Noise Figure	0 dB
Rate Control	IdealRateControlManager
Transmission power	20 dBm
Receiver sensitivity	-85 dBm
Channel model	Friis ($\eta = 2.75$)
Mobility model	Controlled waypoint
Playground size	500 m x 500 m x 250 m
Number of hosts	10
Speed	$50 \frac{km}{h}$
Traffic	UDP constant bit rate (2 Mbps)

For data collection, we use ns-3's `FlowMonitor` to track sent and received packets, and gather the mean end-to-end latency and Packet Delivery Ratio (PDR) as Key Performance Indicators (KPIs) according to [10].

In Section 5.2, we inspect the memory usage and execution time for both simulation tools and an increasing number of agents. For this purpose, we created the evaluation setup according to Table 1

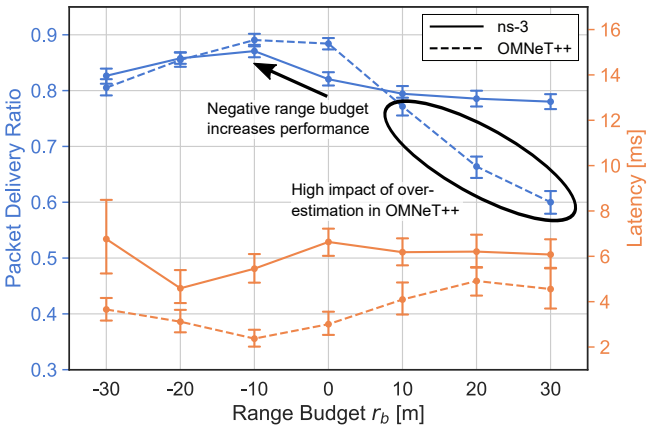

Figure 6: Application of Different Range Budgets

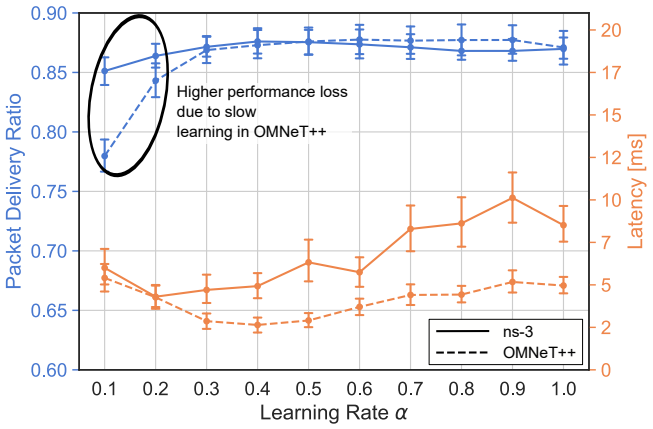

Figure 7: Performance for Different Learning Rates

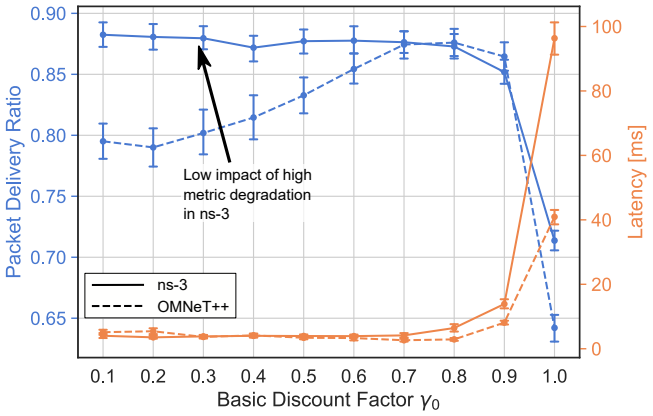

Figure 8: Performance for Different Basic Discount Factors

in both simulators. Console output and result logging were disabled to minimize external influences. The results were monitored for 15 runs by an external tool.

5 RESULTS

In this section, we present the simulation results obtained by ns-3. The error bars show the 95 % confidence interval of the mean values over all runs. We refer to the corresponding results of our work [23] when taking OMNeT++ results into account. In general, we utilize the default parameterization of our previous work as an entry point for this work. We then carry out a new parameter optimization, to gather the ns-3 configuration of PARRoT. Also, we evaluate the scalability of different scaled PARRoT networks in terms of consumption in both simulators.

5.1 Stepwise Parameter Optimization of PARRoT

In the following, we optimize the parameters r_b, α, γ_0, and τ with the default scenario setup in three steps. First, we vary the range budget r_b in the interval of $[-30, 30]$ m. As seen in Figure 6, a change of the estimated communication range has a significant impact on the end-to-end routing performance. For the simulation setup and its parameters, a range of $r_c \approx 230$ m is assumed. Applying a range budget of $r_b = -30$ m therefore leads to an estimation of $r = 200$ m. We observe that an overestimation of communication range leads to performance drains, as links are falsely assumed to be available. This effect is significantly more distinct in OMNeT++. However, while the latency remains within a quite consistent range, a negative range budget has a positive effect on the PDR. This is explained by the fact, that links are not considered available up to the outer edge of the real range, where low Signal-to-Interference-plus-Noise-Ratio (SNIR) conditions may lead to packet errors, even though the reception is possible. Based on this funding, we specify $r_b = -12.5$ m as default value for the following simulations.

In Figures 7 and 8, we see the performance evaluation of the learning rate α and the basic discount factor γ_0 respectively. Concerning the learning rate, we notice a performance drop for parameterization with lower values due to a slower learning phase and the resulting inability to adapt adequately fast to topology changes. For higher α, the PDR again slightly drops, as incoming packets are weighted quite high, and, therefore, the stored knowledge becomes volatile. We observe a local maximum for $\alpha = 0.4$, which we define as the default value. However, it is remarkable, that although the characteristics of the curve are similar to our previous results in OMNeT++, the effect of slower learning is far less distinct in ns-3. Overall, the influence of α is weaker than in OMNeT++. The basic discount factor γ_0 is responsible to guarantee a path degradation along multiple hops. Thus, a value of 1.0 is invalid, as long routes, and potentially routing loops, can occur, which we can identify in Figure 8. In contrast to the OMNeT++ results, the PDR does not show an impact of a higher metric degradation, which means the effect of punishing the path score for every hop too high, and, implicitly limiting the routing to short routes. A value of $\gamma_0 = 0.7$ is set as default.

The last parameter to be optimized is the prediction width τ. As described in Section 3.2.1, this is not only the width of the mobility

prediction but also a time constraint for routing entries. Figure 9 shows the PDR for different widths and different speed profiles of 50 and 250 $\frac{km}{h}$. Correspondingly to OMNeT++, the performance shows a high gain through the application of mobility prediction, compared to $\tau = 0.0$. Also, the overall performance drops with increased agent speed. Another funding we can verify here is that the best choice of τ is delimited to a peak value for higher speeds, while lower speeds show a plateau of equivalent choices of τ. Also, the best choice τ is described by a smaller value for higher speeds. Therefore, the parameterization does not only require a smaller value but becomes more sensitive. The effect of plateaus is a little more expressive in the ns-3 results due to the applied range budget.

As we conclude our parameter optimization and discussed some differences to our results in OMNeT++, Table 2 lists the resulting parameterization.

Table 2: Parameterization of PARRoT in ns-3

Protocol	Parameter	Value
PARRoT	Chirp Interval	0.5 s
	Learning Rate	0.4
	Basic Discount Factor	0.7
	Range Budget	-12.5 m
	Prediction Width	2.0 s
	Prediction Method	Waypoint

5.2 Analysis of Time and Memory Consumption for PARRoT Scenarios

In Figure 10, we show the execution time and memory consumption for an increasing number of PARRoT agents participating in the network for a simulated time of 900 s. With execution time, we refer to the time between starting and ending the simulation, not including the build process. In terms of memory usage, we used `pidstat` to capture the corresponding process id and measured

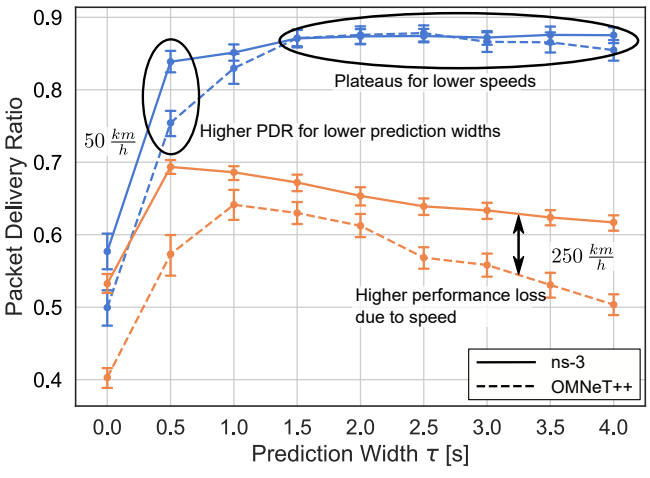

Figure 9: Prediction Width for Low and High Speeds

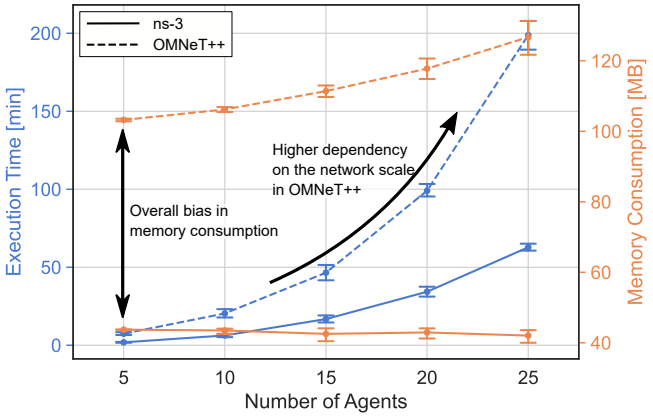

Figure 10: Analysis of Time and Memory Consumption

the resident set size, which is described as the occupied physical memory of the task. The simulations were carried out on the same server in order to provide comparable conditions.

Ns-3 shows a lower and nearly constant memory occupation, that is not significantly affected by the number of agents. OMNeT++ on the other hand requires more than double the amount of memory for a five node network simulation. With increasing agents, more memory is consumed by the simulation.

In addition to the study of memory efficiency, time consumption is of particular interest to users of network simulators. While both simulators can perform the simulation for a five agent network in less than 10 minutes, a growth in acquired time can be observed for both. Although ns-3 is also affected by the increased network complexity, the execution time remains at a considerable low level, while the OMNeT++ setup shows a higher dependency on the network size and makes it unsuitable for extensive studies. [26] also compares the performance of network simulators for a more basic static network with less traffic load and also indicated ns-3 to be the more efficient simulator compared to OMNeT++.

6 CONCLUSION

In this work, we presented the novel PARRoT simulation model for ns-3 which was derived from an existing implementation in OMNeT++. Although there are differences between the two simulation setups that demand further investigation — the parameter optimization has revealed a higher impact of the range budget and a smaller effect of the reinforcement learning hyperparameters — the general behavior of PARRoT could be replicated. The simulation efficiency analysis has shown ns-3 to be the faster and more efficient simulator in terms of the considered MANET routing context. In future work, we will leverage the new ns-3 evaluation setup for performing hardware-in-the-loop simulations of PARRoT as a preparatory step before a comprehensive real-world performance evaluation is performed.

ACKNOWLEDGMENT

This work has been supported by the German Research Foundation (DFG) within the Collaborative Research Center SFB 876 "Providing Information by Resource-Constrained Analysis", projects A4 and B4 as well as by the German Federal Ministry

of Education and Research (BMBF) in the project A-DRZ (13N14857) and the Ministry of Economic Affairs, Innovation, Digitalization and Energy of the state of North Rhine-Westphalia in the course of the Competence Center 5G.NRW under grant number 005-01903-0047, and in the course of the project Plan & Play under grant number 005-2008-0047.

REFERENCES

[1] 3GPP. 2018. *Service Requirements for V2X Services.* Technical Report 22.185. 3rd Generation Partnership Project (3GPP). Version 14.4.0.

[2] Ramy Ali, Bilgehan Erman, Ejder Baştuğ, and Bruce Cilli. 2020. Hierarchical Deep Double Q-routing. In *ICC 2020 - 2020 IEEE International Conference on Communications (ICC)*. Dublin, Ireland, 1–7.

[3] Muhammad Arafat and Sangman Moh. 2019. Routing Protocols for Unmanned Aerial Vehicle Networks: A Survey. *IEEE Access* 7 (2019), 99694–99720.

[4] Usman Ashraf, Amir Khwaja, Junaid Qadir, Stefano Avallone, and Chau Yuen. 2021. WiMesh: Leveraging Mesh Networking For Disaster Communication in Poor Regions of the World. *arXiv preprint arXiv:2101.00573* (2021).

[5] Michael Bahr. 2006. Proposed Routing for IEEE 802.11s WLAN Mesh Networks *(WICON '06)*. Association for Computing Machinery, New York, NY, USA, 5âĂŞes.

[6] Zoltan Bojthe, Levente Meszaros, GyÃűrgy Százkő, Rudolf Hornig, Andras Varga, and Attila TÃűrÃűk. 2020. INET Framework. https://github.com/inet-framework/inet.

[7] Raouf Boutaba, Mohammad Salahuddin, Noura Limam, Sara Ayoubi, Nashid Shahriar, Felipe Estrada-Solano, and Oscar Caicedo. 2018. A Comprehensive Survey on Machine Learning for Networking: Evolution, Applications and Research Opportunities. *Journal of Internet Services and Applications* 9, 1 (2018), 16.

[8] Justin Boyan and Michael Littman. 1994. Packet Routing in Dynamically Changing Networks: A Reinforcement Learning Approach. In *Advances in neural information processing systems*. San Francisco, CA, USA, 671–678.

[9] Nicola Bui, Matteo Cesana, S Amir Hosseini, Qi Liao, Ilaria Malanchini, and Joerg Widmer. 2017. A survey of Anticipatory Mobile Networking: Context-based Classification, Prediction Methodologies, and Optimization Techniques. *IEEE Communications Surveys & Tutorials* (2017).

[10] Gustavo Carneiro, Pedro Fortuna, and Manuel Ricardo. 2009. Flowmonitor: A Network Monitoring Framework for the Network Simulator 3 (ns-3). In *Proceedings of the Fourth International ICST Conference on Performance Evaluation Methodologies and Tools*. Brussels, BEL, 1–10.

[11] Thomas Clausen, Philippe Jacquet, Cédric Adjih, Anis Laouiti, Pascale Minet, Paul Muhlethaler, Amir Qayyum, and Laurent Viennot. 2003. Optimized Link State Routing Protocol (OLSR). (2003).

[12] Baldomero Coll-Perales, Javier Gozalvez, and Juan Maestre. 2019. 5G and beyond: Smart Devices as Part of the Network Fabric. *IEEE Network* 33, 4 (2019), 170–177.

[13] António Fonseca, André Camões, and Teresa Vazão. 2012. Geographical Routing Implementation in ns3 *(SIMUTOOLS '12)*. ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering), Brussels, BEL, 353–358.

[14] Syed Jalil, Mubashir Husain Rehmani, and Stephan Chalup. 2020. DQR: Deep Q-routing in Software Defined Networks. In *2020 International Joint Conference on Neural Networks (IJCNN)*. Glasgow, UK, 1–8.

[15] David Johnson, Yih-Chun Hu, and David Maltz. 2007. *The Dynamic Source Routing Protocol (DSR) for Mobile Ad hoc Networks for IPv4.* Technical Report. RFC 4728.

[16] Brad Karp and Hsiang-Tsung Kung. 2000. GPSR: Greedy Perimeter Stateless Routing for Wireless Networks. In *Proceedings of the 6th annual international conference on Mobile computing and networking*. New York, NY, USA, 243–254.

[17] Muhammad Khan and Kok-Lim Yau. 2020. Route Selection in 5G-based Flying Ad-hoc Networks using Reinforcement Learning. In *2020 10th IEEE International Conference on Control System, Computing and Engineering (ICCSCE)*. Penang, Malaysia, 23–28.

[18] Guido Oddi, Donato Macone, Antonio Pietrabissa, and Francesco Liberati. 2012. A Proactive Link-failure Resilient Routing Protocol for MANETs based on Reinforcement Learning. In *2012 20th Mediterranean Conference on Control Automation (MED)*. Barcelona, Spain, 1259–1264.

[19] Charles Perkins and Pravin Bhagwat. 1994. Highly Dynamic Destination-sequenced Distance-vector Routing (DSDV) for Mobile Computers. *ACM SIGCOMM computer communication review* 24, 4 (1994), 234–244.

[20] Charles Perkins and Elizabeth Royer. 1999. Ad-hoc On-demand Distance Vector Routing. In *Proceedings WMCSA'99. Second IEEE Workshop on Mobile Computing Systems and Applications*. New Orleans, LA, USA, 90–100.

[21] Alfonso Quintana. 2020. INETMANET Branch of INET Framework. https://github.com/aarizaq/inetmanet-4.x.

[22] Craig Reynolds. 1987. Flocks, Herds and Schools: A Distributed Behavioral Model. In *Proceedings of the 14th annual conference on Computer graphics and interactive techniques*. New York, NY, USA, 25–34.

[23] Benjamin Sliwa, Cedrik Schüler, Manuel Patchou, and Christian Wietfeld. 2021. PARRoT: Predictive Ad-hoc Routing Fueled by Reinforcement Learning and Trajectory Knowledge. In *2021 IEEE 93rd Vehicular Technology Conference (VTC-Spring)*. Helsinki, Finland.

[24] Fengxiao Tang, Bomin Mao, Zubair Fadlullah, Nei Kato, Osamu Akashi, Takeru Inoue, and Kimuhiro Mizutani. 2018. On Removing Routing Protocol from Future Wireless Networks: A Real-time Deep Learning Approach for Intelligent Traffic Control. *IEEE Wireless Communications* 25, 1 (2018), 154–160.

[25] Valmik Tilwari, Kaharudin Dimyati, Mohammad Hindia, Anas Fattouh, and Iraj Amiri. 2019. Mobility, Residual Energy, and Link Quality Aware Multipath Routing in MANETs with Q-learning Algorithm. *Applied Sciences* 9, 8 (Apr 2019), 1582.

[26] Elias Weingartner, Hendrik Vom Lehn, and Klaus Wehrle. 2009. A Performance Comparison of Recent Network Simulators. In *2009 IEEE International Conference on Communications*. IEEE, Dresden, Germany, 1–5.

[27] Yoong Zeng, Qingqing Wu, and Rui Zhang. 2019. Accessing from the Sky: A Tutorial on UAV Communications for 5G and Beyond. *Proc. IEEE* 107, 12 (2019), 2327–2375.