Christian-Albrechts-Universität zu Kiel Lehrstuhl für Nachrichten-übertragungstechnik Kaiserstr. 2 24143 Kiel		BMBF-Verbundprojekt QuNet+ML Optimierung von Quantenkommunikations-netzen mittels maschinellen Lernens

Schlussbericht

Zuwendungsempfänger: Christian-Albrechts-Universität zu Kiel Prof. Dr.-Ing. Stephan Pachnicke (stephan.pachnicke@tf.uni-kiel.de)	**Förderkennzeichen:** 16KISQ069
Vorhabenbezeichnung: Digital. Sicher. Souverän. Verbundprojekt: Optimierung von QKD-Netzen mittels maschinellen Lernens Teilvorhaben: Systemoptimierung und Pfadwahl in QKD-Netzen unter Einsatz von maschinellem Lernen	
Laufzeit des Vorhabens: 01.01.2022 – 30.06.2025	

Inhalt

2 Kurzdarstellung ... 3
2.1 Aufgabenstellung ... 3
2.2 Wissenschaftlich-technischer Stand bei Projektbeginn ... 3
2.3 Ergebnisse ... 4
2.4 Zusammenarbeit ... 4
3 Eingehende Darstellung ... 5
3.1 Erzielte Ergebnisse ... 5
3.1.1 Projektziele ... 5
3.1.2 Projektergebnisse ... 6
3.2 Zahlenmäßiger Nachweis ... 23
3.3 Notwendigkeit und Angemessenheit der Arbeit ... 24
3.4 Nutzen ... 24
3.5 Veröffentlichungen ... 24

2 Kurzdarstellung

2.1 Aufgabenstellung

Das Projekt QuNET+ML ist Teil der QuNET-Initiative. Die verschiedenen Plus-Projekte adressieren hierbei die Fehlbedarfe des Kernprojektes. Ziel von QuNET+ML war es, den Einsatz von Quantum Key Distribution (kurz: QKD) in realistischen Netzszenarien zu untersuchen. Das Interesse lag insbesondere auf vermaschten Netzstrukturen mit dynamischen Lastanforderungen. Bei der Realisierung eines vermaschten QKD-Netzes ist, für eine erhöhte Ausfallsicherheit, die Berücksichtigung von Resilienzverfahren besonders wichtig. Darüber hinaus muss in großen Netzen ebenfalls von heterogenen Systemen ausgegangen werden, wodurch unter anderem unterschiedliche QKD-Verfahren, Protokolle, etc. innerhalb eines Netzes eingesetzt werden. Hieraus folgt, dass die Optimierung der verschiedenen Systemparameter äußerst herausfordernd sein wird und mit klassischen Optimierungsverfahren nicht gelöst werden kann. Daher sollen verschiedene Methoden des maschinellen Lernens (ML) untersucht werden. Diese Methoden sollen in einem Demonstrator integriert werden, um mit Hilfe der Industriepartner eine zügige Kommerzialisierung voranzutreiben.

2.2 Wissenschaftlich-technischer Stand bei Projektbeginn

QKD ist eine bereits weit erforschte Anwendung im Bereich der Quantenkommunikation und konnte praktisch realisiert werden. Im Softwarebereich gibt es eine Vielzahl von internationalen Bestrebungen zur Untersuchung verschiedener Protokolle. Hierbei zeigen die Kodierung und Übertragung von nichtklassischen Phasen-/Amplituden-Zuständen oder auch Polarisationsverschränkungen jeweils ihre Vorteile, wobei jedes Verfahren individuelle Stärken aufweist, je nach Anwendung bzw. zu überbrückender Distanz.

Auf der Hardwareseite werden zumeist homogene Architekturen betrachtet (Faser-, Freistrahl- oder Satelliten-Verbindungen). Jedoch werden zukünftige Quantenkommunikationsnetze sehr wahrscheinlich aus hybriden Strukturen bestehen, so dass ein reibungsloser Wechsel zwischen den Technologien möglich sein wird. So können z.B. Quanten-Repeater in lokalen Glasfasernetzen genutzt werden oder auch sogenannte „Trusted Nodes“ als Repeater sowie Satellitenverbindungen für Langstreckenübertragungen.

Weitere Fortschritte in der Forschung führten zur Realisierung kompakter Quellen für verschränkte Photonen, welche vielseitiger und leistungsstärker geworden sind. Es konnten bereits mittels mobiler Bodenstationen und Fluggeräten (wie z.B. Flugzeugen, Drohnen, Ballons) Freistrahlverbindungen realisiert werden. Die Überbrückungsdistanz lag hier zwischen einigen Kilometern bis zu 100 km, war jedoch von guten Sichtbedingungen zur Übertragung der Quantenzuständen abhängig. Die Übertragungsraten waren ebenfalls beschränkt und konnten nur für ein zwei User-Szenario realisiert werden. Im Bereich der faser-basierten QKD-Tests konnten bereits diverse zeitlich begrenzte Forschungsstrecken realisiert werden. Hierbei lagen die Übertragungsstrecken bei ca. 100 km. Auf diese Weise konnte somit die allgemeine technische Realisierbarkeit von QKD gezeigt werden. Für größere Übertragungsstrecken (ca. 1000 km) kann Satelliten-QKD mittels DV-QKD (discrete-variable QKD) genutzt werden, was erstmals mit dem MICIUS Forschungssatelliten von chinesischen Wissenschaftlern 2017 demonstriert wurde. Parallel dazu zeigten Forscher des Max-Planck Instituts für die Physik des Lichtes in Erlangen eine Realisierung von CV-QKD mittels eines TESAT-Satelliten. Darüber hinaus wird in weiteren Projekten an anderen Satelliten-Link-Technologien für P2P-Verbindungen geforscht.

Viele Start-Ups bieten bereits QKD-Hardware und Expertise weltweit kommerziell an, wohingegen in Deutschland das Thema QKD vorwiegend in der Forschung adressiert wurde. Jedoch zeigen größere deutsche Unternehmen ein steigendes Interesse an QKD und arbeiten dabei an langfristigen Projekten. So sind die Deutsche Telekom AG und die ADVA Optical Networking SE (jetzt: Adva Networking Security GmbH) in einigen Konsortien und Projekten zum Thema QKD aktiv.

Die Fraunhofer-Gesellschaft, die Max-Planck-Gesellschaft und das Deutsche Zentrum für Luft- und Raumfahrt (DLR) (allesamt an der QuNET Initiative beteiligt) konnten 2021 wichtige Grundsteine für moderne und sichere Kommunikationsstandards legen. Die Forscher/-innen haben die Gesamtarchitektur für Systeme zur quantensicheren Kommunikation weiterentwickelt und ebenfalls Möglichkeiten zum Austausch von Quantenschlüsseln über kurze, mittlere und lange Strecken mittels Faser- und Freistrahlsystemen entwickelt. Für eine erste Demonstration wurde hierfür eine Videokonferenz zwischen BMFTR und BSI quantenbasiert übertragen. Hierfür wurden mehrere Freistrahl- und Fasersysteme genutzt. Bei dieser Demonstration handelte es sich um ein komplexes Verbindungsszenario über einen einzelnen Kanal. Dieser Versuchsaufbau diente zusätzlich als wissenschaftliches Experiment, mit dem wichtige Erkenntnisse für die Kommunikation in komplexen Netzen gewonnen werden konnten. Das QuNET+ML Projekt soll auf ebendiesen Erfahrungen und Erkenntnissen aufbauen.

Aus Mangel an Quantenrepeatern sind die Übertragungsreichweiten auf Faserstrecken für QKD zurzeit noch auf ungefähr 100 km begrenzt. Aus diesem Grund muss für eine Weitstreckenübertragung derzeit auf sogenannte Trusted Nodes als Repeater zurückgegriffen werden, damit ein sicherer Ende-zu-Ende Schlüssel realisiert werden kann. Aus diesem Grund ist es für vermaschte QKD-Netze unabdingbar, dass stets zwischen allen Knotenpunkten ausreichend Schlüssel verfügbar sind, um den anfallenden Datenverkehr zu verschlüsseln. Zu diesem Zweck können Verfahren des maschinellen Lernens genutzt werden, um das Schlüsselmanagement und die Schlüsselverteilung in großen vermaschten Netzen zu realisieren. Analog kann hierzu die ML-basierte Automatisierung klassischer Netze betrachtet werden.

2.3 Ergebnisse

Die wesentlichen Projekt-Ergebnisse der Christian-Albrechts-Universität (CAU) sind wie folgt zusammenzufassen:

- Mitwirkung an der Definition der Netzarchitektur
- Ausarbeitung von Vergleichsalgorithmen, um die Ergebnisse der ML-Algorithmen zu validieren
- Entwicklung eines ML-Algorithmus zu einer vorhersagebasierten Routingoptimierung unter der Nutzung klassischer Shortest-Path-Algorithmen (SP-Algorithmen)
- Entwicklung eines Deep Reinforcement Learning (DRL) Agenten für dezentrales Routing ohne die Verwendung klassischer SP-Algorithmen
- Entwicklung eines DRL Agenten für zentrales Routing ohne die Verwendung klassischer SP-Algorithmen
- Bewertung der Sicherheit der verschiedenen ML-Algorithmen

2.4 Zusammenarbeit

Um eine enge Zusammenarbeit innerhalb des Projektes zu gewährleisten wurde alle 4 Wochen ein Status Meeting mit allen Projektpartnern durchgeführt. Die Teilnahme an vorort- und Online-Treffen des QuNET-Kernprojektes sind im Folgenden aufgelistet:

18.03.2022:	Virtual Kickoff Meeting (online)
19.05.2022-20.05.2022:	Projekt Meeting (Berlin)
23.06.2022:	Fachgruppentreffen Faser (online)
04.10.2022-05.10.2022:	Projekt Meeting (Nürnberg/Erlangen)
15.11.2022:	Mitgliedervollversammlung QuNET (online)
28.02.2023-01.03.2023:	QuNET-Fachgruppentreffen (Jena)
06.11.2023-10.11.2023:	SQuadD-QuNET-Workshop + Mitgliedervollversammlung QuNET (Jena)
07.12.2023:	Fachgruppentreffen Schnittstellen & Architektur (online)
04.03.2024-05.03.2024:	QuNET-Fachgruppentreffen (Berlin)
10.10.2024-11.10.2024:	Mitgliedervollversammlung QuNET (Oberpfaffenhofen)
05.12.2024:	QuNET-Fachgruppentreffen (online)
05.05.2023-07.05.2025:	QuNET+ML Abschluss-Meeting + QuNET-Fachgruppentreffen (Oberpfaffenhofen)

3 Eingehende Darstellung

3.1 Erzielte Ergebnisse

3.1.1 Projektziele

Die Ziele des Projektes QuNET+ML sind die Definition von Netzparametern und die Entwicklung von ML-Algorithmen zur Optimierung der Schlüsselnutzung in einem vermaschten QKD-Netz, welches auf der Nutzung von Trusted Nodes basiert. Gleichzeitig soll die Sicherheit der KI gegenüber dem Einfluss Dritter betrachtet werden. Die gesammelten Erkenntnisse sollen für die weiteren Projektphasen der QuNET-Initiative genutzt werden.

Der Fokus der Christian-Albrechts-Universität zu Kiel richtet sich hierbei im Besonderen auf die Entwicklung von ML-Algorithmen zur Optimierung des Routings zur Etablierung eines sicheren Ende-zu-Ende Schlüssels zwischen zwei Kommunikationsparteien. Der Schwerpunkt der Arbeit lag somit auf den Arbeitspaketen 3.1, 3.2 und 3.3.

3.1.1.1 Erzeugung von Referenzdaten

Basierend auf der im Projekt definierten Netzarchitektur werden Referenzdaten auf Netzebene modelliert, die für das maschinelle Lernen genutzt werden können. Hierbei liegt der Fokus auf der Generierung einer realistischen Simulationsumgebung eines vermaschten QKD-Netzes. Die Modellierung der Schlüsselraten sollen hierbei auf dem Testbett des Fraunhofer Heinrich-Hertz-Instituts (HHI) basieren.

3.1.1.2 Entwicklung von ML-Modellen

Der Fokus der ML-Modelle soll auf der Systemoptimierung und dem Routing liegen. Hierbei soll u.a. eine zentrale Datenbank angenommen werden, die die ML-Modelle mit Informationen versorgt. Des Weiteren sollen die Routingalgorithmen derart konzipiert werden, dass sie in vermaschten Netzen funktionsfähig sind. In einer Simulation sollen die entwickelten ML-Modelle anhand verschiedener Referenzszenarien angewendet und die Leistungsfähigkeit simulativ evaluiert werden. Darüber hinaus werden die ML-Algorithmen mit einem Basisszenario, basierend auf klassischen Shortest-Path Algorithmen, verglichen. Abschließend sollen ebenfalls Resilienzverfahren untersucht werden, damit die

Reaktion der Algorithmen auf mögliche Fehler in einer vermaschten QKD-Netzstruktur geprüft werden kann.

3.1.1.3 Sicherheit der KI

Es erfolgte eine Bewertung der Angreifbarkeit der ML-Algorithmen. Dies bezieht sich z.B. auf mögliche Attacken auf die zentrale Datenbank, die genutzt wird, um einen möglichst idealen Pfad für das Schlüssel-Routing zu finden.

3.1.2 Projektergebnisse

3.1.2.1 Erzeugung von Referenzdaten

Um die Routingalgorithmen, die innerhalb dieses Projektes entwickelt werden, zu testen, muss ein vermaschtes QKD-Netz simuliert werden. Aus Unkenntnis einer realen QKD-Topologie, wurde für eine realistische Simulation auf die Nobel-Germany Topologie des DFN Vereins zurückgegriffen, welche in Abb. 1 dargestellt ist. Es handelt sich hierbei um eine aggregierte Version des realen DFN-Netzwerkes. Da die Topologie inklusive der Geokoordinaten als open-source zur Verfügung steht, können die Entfernungen der jeweiligen benachbarten Knoten voneinander berechnet werden. Diese Wegstrecken können genutzt werden, um die distanz-abhängigen Schlüsselraten möglichst realistisch zu berechnen. Des Weiteren stehen für diese Topologie Datenverkehrsmatrizen zur Verfügung, welche exemplarisch in Tab. 1 dargestellt sind. Diese können für Datenverkehrsvorhersagen mittels maschinellen Lernens genutzt werden. Anhand des Datenverkehrs wird die benötigte Menge an QKD-Schlüsseln berechnet.

Abb. 1: Nobel-Germany Topologie erweitert mit Trusted Nodes

Die QKD-Schlüssel werden genutzt, um Datenverkehr mittels Algorithmen, wie z.B. AES, zu verschlüsseln.

Tab. 1: Exemplarische Darstellung einer Verkehrsmatrix

Start	Ziel	Nachgefragte Datenmenge
'Hamburg'	'München'	100 GByte
⋮	⋮	⋮
'Berlin'	'Dortmund'	35 GByte

Um ein realistisches QKD Szenario zu simulieren, werden die Teilstrecken der einzelnen Knotenverbindung auf maximal 80 km beschränkt, um auf diese Weise eine ausreichende Schlüsselrate in den Simulationen zu erreichen. Überschreitet eine Teilstrecke 80 km wird sie in Abschnitte gleicher Länge unterteilt, um die Schlüsselraten auf den Teilstrecken auf einem vergleichbaren Niveau zu halten. Die Berechnungen der Schlüsselraten basieren auf einem Modell des Fraunhofer HHI und werden im Folgenden erläutert. Die Grundlage für das Modell bildet das BB84-Protokoll. Es werden in diesem Fall also Einzelphotonenquellen und Einzelphotonendetektoren angenommen. In diesem Modell werden die sicheren Schlüsselbits in Blöcken N_{sec} nach einem gewissen Intervall ausgegeben. Es handelt sich daher nicht um einen konstanten Fluss von Bits, die für die Generierung der Schlüssel zur Verfügung stehen. Zur Berechnung von N_{sec} wird in einem ersten Schritt die Quantum Bit Error Rate (QBER) berechnet:

$$\mathrm{QBER} = \frac{1}{2} \cdot \frac{p_{dc} + p_{ap} \cdot (p_{det} + p_{dc}) + q_{TX} \cdot p_{det}}{(1 + p_{ap}) \cdot (p_{det} + p_{dc})}.$$

Die QBER gibt Aufschluss über die potentielle Existenz einer dritten Partei, die den Schlüsselaustausch möglicherweise unerlaubt überwacht. Überschreitet die QBER einen gewissen Schwellwert, können die Schlüsselbits nicht mehr genutzt werden, da der Austauschprozess möglicherweise korrumpiert wurde. Die Berechnung der QBER beinhaltet verschiedene Wahrscheinlichkeiten, wie p_{dc} als die Wahrscheinlichkeit für die Dunkelzählrate pro Qubit. Außerdem werden die Wahrscheinlichkeit für das Nachpulsieren p_{ap} und die Wahrscheinlichkeit für einen Fehler bei der Zustandsvorbereitung q_{TX} verwendet. Die Wahrscheinlichkeit zur Detektion wird mit p_{det} angegeben und leitet sich von der durchschnittlichen Photonenzahl pro Qubit ab. Anschließend kann mittels der QBER und der binären Shannon Entropie der Kompressionsfaktor f_{PA} berechnet werden.

$$f_{\mathrm{PA}} = 1 - h(\mathrm{QBER})$$

Mittels des Kompressionsfaktors und der Blocklänge nach dem Post-Processing $\boldsymbol{N_{PP}}$ kann abschließend die Blocklänge des sicheren Schlüssels nach einem gewissen Zeitintervall berechnet werden und ergibt sich aus:

$$N_{\mathrm{sec}} = N_{\mathrm{PP}} \cdot f_{\mathrm{PA}}.$$

Mittels dieser Formeln können für die Topologie aus Abb. 1 längenabhängige Schlüsselraten berechnet werden. Des Weiteren können verschiedene Parameter ebenfalls stochastisch variiert werden, um bei Bedarf ein dynamischeres Verhalten bei der Schlüsselerzeugung zu erhalten. In Abb. 2 ist exemplarisch eine entfernungsabhängige Schlüsselrate aus dem Modell abgebildet.

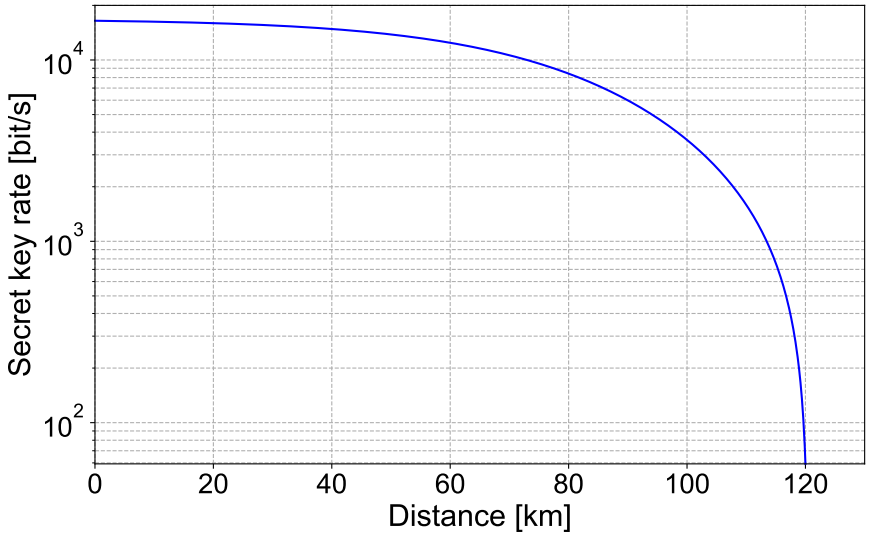

Abb. 2: Entfernungsabhängige Schlüsselrate

3.1.2.2 Entwicklung von ML-Modellen

Bisher bestehen noch keine einheitlichen Standards für den Betrieb eines QKD-Netzes. Darüber hinaus sind die Anforderungen an ein solches Netz, z.B. ob eine zentrale oder dezentrale Kontrolle des Netzes erwünscht ist, stark von den Nutzern abhängig. Aus diesem Grund werden innerhalb dieses Projektes verschiedene Szenarien für das Routing von Ende-zu-Ende Schlüsseln betrachtet. Auf diese Weise ist es möglich, die verschiedenen Vor- und Nachteile der ML-basierten Algorithmen zu vergleichen. Im folgenden Abschnitt werden zu Beginn das Referenzszenario und anschließend drei verschiedene ML-Algorithmen vorgestellt.

3.1.2.2.1 Definition und Entwicklung von Referenzszenarien

Für eine qualitative Einschätzung der Qualität der entwickelten ML-Algorithmen ist die Definition und Entwicklung von Referenzalgorithmen in einem Vergleichsszenario entscheidend. Zu diesem Zweck wurden, in enger Zusammenarbeit mit den Projektpartnern (Deutsche Telekom und Adva Networking Security), verschiedene Vergleichsalgorithmen für das Routing zur Etablierung von Ende-zu-Ende-Schlüsseln entwickelt. Bei der Definition der Architektur eines vermaschten QKD-Netzes wurde festgelegt, das die Schlüssel On-Demand erstellt werden, um lange Speicherzeiten von Ende-zu-Ende-Schlüsseln zu verhindern. Diese Speicherzeiten könnten zu einer Minimierung des Sicherheitslevels und zur Verschwendung wichtiger Schlüsselressourcen führen. Die ausgetauschten Schlüsselbits werden in

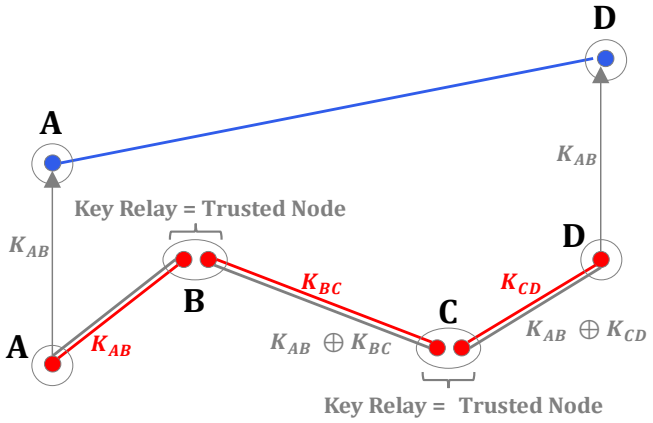

Abb. 3: XOR-basierte Weiterleitung zur Generierung eines Ende-zu-Ende Schlüssels

Schlüsselspeichern gesichert und bei Bedarf zur Erstellung eines Ende-zu-Ende-Schlüssels verwendet. In Abb. 3 ist gezeigt, wie der Schlüssel schrittweise zwischen Start- und Zielknoten mittels einer bitweisen XOR-Operation verschlüsselt wird. Auf diese Weise haben Start- und Zielknoten am Ende einen gemeinsamen geheimen Schlüssel, jedoch wird das Schlüsselmaterial aus den Schlüsselspeichern entlang des kompletten Pfades konsumiert. Eine Alternative wäre hierfür in der Zukunft der Einsatz von Quantenrepeater. Da diese sich jedoch noch in der Entwicklungsphase befinden, wurden sie im Rahmen dieses Projektes nicht berücksichtigt.

Das Schlüsselmanagementsystem an jedem Knotenpunkt verwaltet die einzelnen virtuellen Subspeicher. Eine exemplarische Darstellung der Netzarchitektur ist in Abb. 4 dargestellt. Das übergeordnete Ziel für den Netzkontroller muss daher sein, die Schlüsselspeicher auf einem möglichst hohen Niveau zu halten. Auf diese Weise können Engpässe verhindert werden und somit ein reibungsloser Betrieb des Netzes sichergestellt werden. Untersuchungen der Deutschen Telekom haben den Schlüsselspeicherstand zwischen benachbarten QKD Geräten als den zentralen Parameter identifiziert. Bei der Entwicklung der ML-Algorithmen wird sowohl ein zentrales als auch dezentrales Schlüsselmanagement betrachtet.

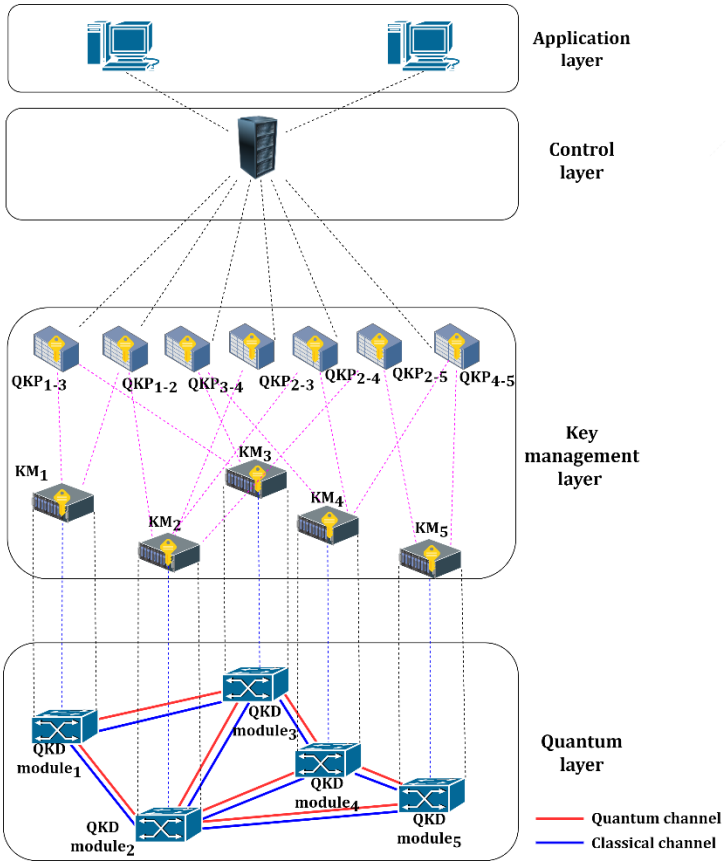

Abb. 4: Darstellung der Netzarchitektur

Für den Vergleich zwischen ML-basiertem und klassischem Routing ist die anfängliche Wahl auf den Dijkstra-Algorithmus für die Berechnung eines gewichteten Pfades zwischen Start- und Zielknoten gefallen. Die Kantengewichte des Graphen spiegeln hierbei die Kosten wider, die entstehen, wenn eine bestimmte Verbindung genutzt wird. Mittels des Dijkstra-Algorithmus wird der entsprechende günstigste Pfad berechnet, der vom Start zum Ziel führt. Für die Gewichtung wird folgende Funktion verwendet:

$$W = e^{a \cdot E}$$

Wobei a ein Skalierungsfaktor ist und E das Verhältnis des aktuellen Füllstandes des Schlüsselspeichers an einer Kante zum maximalen Speicherfüllstand darstellt. Mittels des Skalierungsfaktors kann diese Gewichtung verändert und an die aktuelle Topologie angepasst werden. Im Zuge unserer Untersuchung wurde $a = 1$ verwendet.

3.1.2.2.2 Entwicklung von Algorithmen zur Routingoptimierung mittels maschinellen Lernens

3.1.2.2.2.1 Datenverkehrsvorhersage mittels Long Short-Term Memory

Problematisch bei einem traditionellen Algorithmus, wie dem im vorherigen Abschnitt beschriebenen gewichteten Dijkstra-Algorithmus, ist die Tatsache, dass dieser alle aktuellen Speicherfüllstände des kompletten Netzes in Echtzeit kennen muss, um optimale Entscheidungen treffen zu können. Da QKD informationstheoretische Sicherheit bietet, muss auch der Betrieb des Netzes möglichst widerstandsfähig gegen Angriffe sein bzw. die Angriffsfläche beim Netzwerkmanagement sollte möglichst gering sein. Es wurde daher ein vorhersagebasierter Algorithmus entwickelt, der die Speicherstände lediglich zu Beginn eines Zeitintervalls kennen muss. Auf diese Weise können die Kantengewichtungen vorausschauend angepasst werden und gleichzeitig der Datenverkehr zum Netz-Controller reduziert werden, um die Resilienz des Netzes zu erhöhen (z.B. auf Attacken, die bewusst falsche Daten in den Netz-Controller induzieren).

Simulationssetup

In unserem Simulationsszenario werden, wie zuvor beschrieben, die Schlüsselspeicher in festen Zeit-Intervallen mit neuen Schlüsseln versorgt. Innerhalb eines solchen Intervalls findet parallel der Datenaustausch, welcher mittels der Schlüssel verschlüsselt wird, statt. Abhängig von der Menge des Datenverkehrs werden für jede Anfrage unterschiedlich viele Schlüssel benötigt, um den Datenverkehr zu verschlüsseln. Der Datensatz aus der SNDlib verfügt über 288 Datenmatrizen, die den dynamischen Datenverkehr eines Tages in einer Granularität von 5 Minuten abbilden. Insgesamt handelt es sich um 73.512 Datenverkehrsströme. Mittels eines auf einem Long Short-Term Memory (LSTM) basierenden Algorithmus, sollen diese Datenverkehrsströme vorhergesagt werden. Anschließend werden immer die letzten 3 Datenmatrizen genutzt, um die Matrix im nächsten Zeitschritt vorherzusagen. Zu Beginn eines Zeitintervalls (60 Sekunden) werden die Vorhersagen benötigt, um die Kantengewichtungen auf Basis der Vorhersage zu optimieren. Die ersten 50 Datenmatrizen wurden genutzt, um das LSTM zu trainieren. Die vorhersagebasierten Kantengewichtungen werden verwendet, um mittels eines Dijkstra-Algorithmus die besten Pfade für die verschiedenen Ende-zu-Ende Schlüssel zu finden. Dies hat den Vorteil, dass der Netzcontroller die Speicherstände nur einmal zu Beginn eines jeden Intervalls abfragen muss und nicht vor jeder Schlüsselanfrage. Dies hat zum Vorteil, dass der Managementdatenverkehr reduziert werden kann, was die potentielle Angriffsfläche für Drittparteien minimiert.

Der ML-Algorithmus wird mit vier weiteren Algorithmen verglichen, um die Qualität der Routingentscheidungen bewerten zu können. Der einfachste Algorithmus ist der sogenannte „Hop-basierte“ kürzeste Pfad. Dieser Algorithmus berechnet die fünf kürzesten Pfade auf Basis der passierten Knoten zwischen Start und Ziel. Des Weiteren wird ein gewichteter Dijkstra-Algorithmus genutzt, der vor jeder Pfadberechnung die Schlüsselspeicherstände im Netz abfragt, wie im vorherigen Abschnitt bereits erläutert wurde. Auch hier werden auf Basis der Kantengewichtungen die fünf kürzesten Pfade berechnet. Des Weiteren wird ein vom Projektpartner ANS entwickelter ILP-Algorithmus genutzt, der die hypothetische Annahme nutzt, dass alle Anfragen eines Zeitschrittes bereits vorher bekannt sind. Anschließend berechnet er die optimalen Pfade für alle Anfragen des Zeitschritts auf Basis der Speicherfüllstände zu Beginn des Intervalls. Zuletzt nehmen wir, um die Qualität der Vorhersage bewerten zu

können, das hypothetische Szenario an, dass die Matrizen mit perfekter Genauigkeit vorhergesagt werden können. Anschließend werden die Kantengewichte analog zu dem ML-Algorithmus auf Basis der Vorhersage angepasst und die Schlüsselanfragen des Intervalls auf Basis dieser Gewichtungen geroutet.

Der Ablauf einer Simulation ist in Abb. 5 dargestellt. Zu Beginn werden die Datenmengen zwischen den verschiedenen Knoten mit einem Skalierungsfaktor multipliziert. Auf diese Weise soll untersucht werden, wie gut die verschiedenen Routingalgorithmen in der Lage sind, mit einem Anstieg der zu verschlüsselnden Datenmengen umzugehen. Zu Beginn der Simulation werden komplett gefüllte Schlüsselspeicher angenommen. Anschließend werden die Nachfragematrizen zufällig umsortiert, um einen Einfluss der Reihenfolge der Anfragen auf das Routingergebnis zu verhindern. Anschließend werden nacheinander die fünf kürzesten Pfade für die jeweiligen Anfragen berechnet. Im Folgenden wird für alle Pfade geprüft, ob genügend Schlüssel entlang des Pfades vorhanden sind. Falls dies der Fall ist, wird der Pfad genutzt und die Schlüssel entlang des Weges konsumiert, und die Anfrage ist demzufolge erfolgreich bearbeitet. Sollte keiner der Pfade ausreichend Schlüssel vorrätig haben, gilt die Anfrage als blockiert und die nächste Anfrage wird bearbeitet. Dieses Vorgehen wird iterativ für jeden Algorithmus und für jede Anfrage wiederholt.

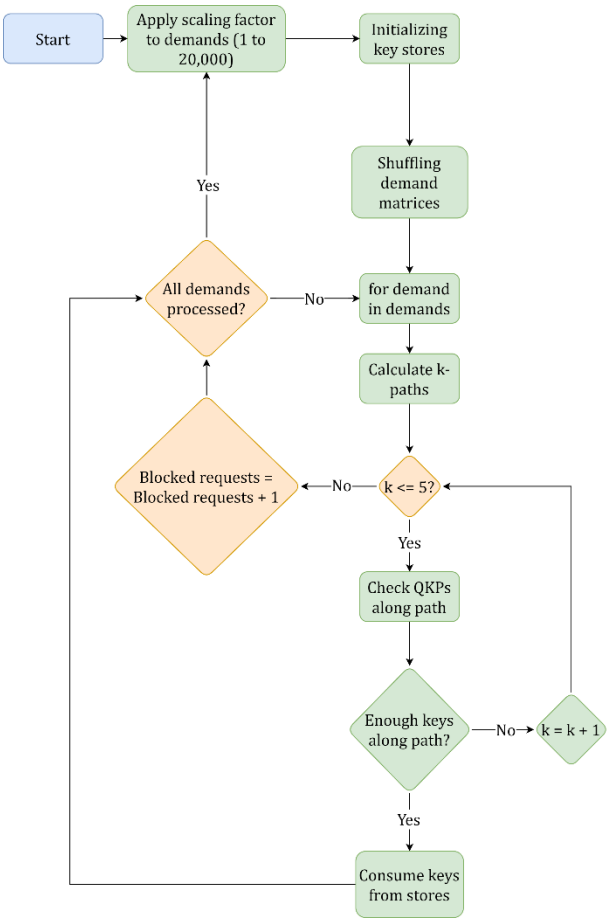

Abb. 5: Ablaufdiagramm der Simulation

Ergebnisse

Als Benchmark dient der ILP-Algorithmus, da dieser basierend auf mathematischen Zielfunktionen die Pfade nahezu ideal berechnen und auswählen kann. Er basiert aber auf der idealisierten Annahme, dass die Schlüsselanfragen vorher bekannt sind. In einem realen Netz wird dies nicht der Fall sein. Das Ziel der übrigen Algorithmen ist es aber, möglichst nah an der Performance des ILP zu arbeiten. Abb. 6

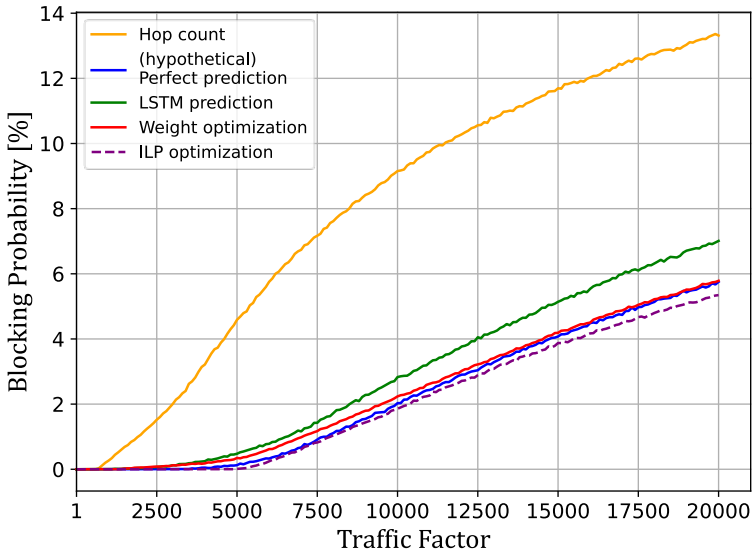

Abb. 6. Blockierungswahrscheinlichkeit von Schlüsselanfragen in Abhängigkeit zum Skalierungsfaktor.

zeigt, dass der „Hop-basierte“ Algorithmus schnell die ersten Anfragen blockiert, da auf den gewählten Pfaden nicht mehr genügend Schlüssel in den Speichern vorhanden sind. Dies liegt an der simplen Kantengewichtung als Kriterium und der daraus resultierenden Ignorierung der Schlüsselspeicherstände. Die hypothetische, perfekte Vorhersage arbeitet am dichtesten an dem ILP-Algorithmus und damit ebenfalls etwas besser als der gewichtungsoptimierte Dijkstra-Algorithmus, der vor jeder Routingentscheidung die Speicherfüllstände abfragen muss. Hierdurch zeigt sich auch, welche potentiellen Ergebnisse die LSTM-basierte Vorhersage erzielen könnte, wenn die Vorhersage eine höhere Genauigkeit aufweisen würde. Die Performance der LSTM-Vorhersage ließe sich verbessern, wenn eine größere Menge an realistischen Datenmatrizen zur Verfügung stehen würde. Somit handelt es sich hierbei um einen Trade-off zwischen geringeren Blockierungsraten und einer Minimierung des Managementdatenverkehrs zwischen Netzcontroller und den einzelnen QKD-Knoten im Netz.

Abb. 7 zeigt den Speicherfüllstand der stark beanspruchten Verbindung zwischen Frankfurt und Köln. Es ist klar zu sehen, dass der Hop-basierte Algorithmus den Speicher sehr schnell leert und nicht in der Lage ist, für eine Entlastung zu sorgen. Die hypothetische, perfekte Vorhersage ermöglicht einen deutlich höheres mittleres Füllstandsniveau. Bei der LSTM-Vorhersage ist zu berücksichtigen, dass die ersten 50 Matrizen zum Training verwendet werden. Daher agiert der Algorithmus zu Beginn analog zu dem Hop-basierten Algorithmus, da dieser während der Trainingszeit angewendet wird. Anschließend ist jedoch eine erhebliche Verbesserung des Füllstandes zu erkennen und ein höheres Niveau stellt sich ein. Der gewichtungsbasierte Algorithmus leistet insbesondere zu Beginn sehr gute Arbeit, auch wenn er das Niveau des ILP-Algorithmus nicht erreichen kann.

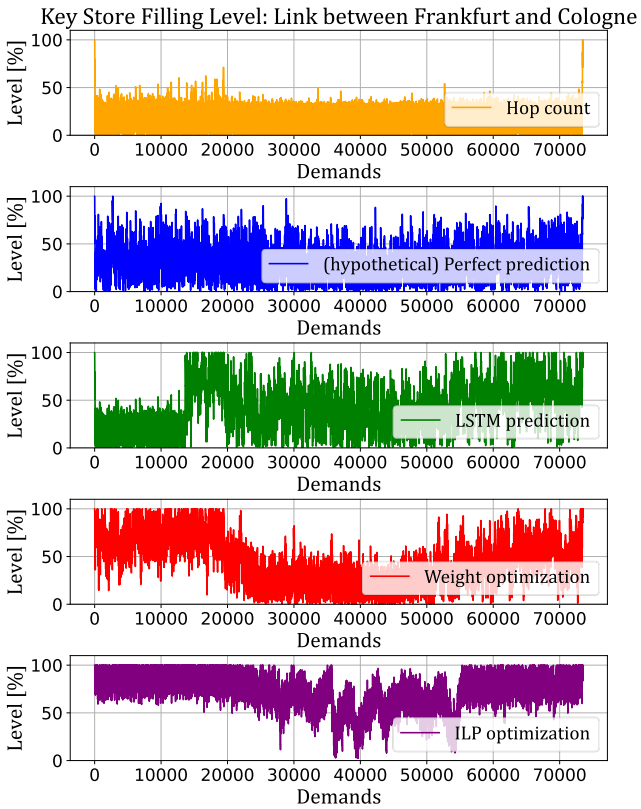

Abb. 7: Verlauf des Schlüsselspeicherstandes der Verbindung zwischen Frankfurt und Köln in Abhängigkeit zur Anzahl der Schlüsselanfragen.

Zusammenfassend kann gesagt werden, dass eine vorhersagebasierte Optimierung des Routings erhebliches Potential bietet.

Der Managementdatenverkehr kann durch den vorhersagebasierten ML-Algorithmus erheblich minimiert werden, da die Speicherfüllstände lediglich zu Beginn eines Zeitschrittes abgefragt werden müssen und zusätzlich noch einmal bei der Auswahl des finalen Pfades. Dies ist in Abb. 8 dargestellt. Es ist klar zu erkennen, dass der gewichtungsbasierte Algorithmus einen deutlich höheren

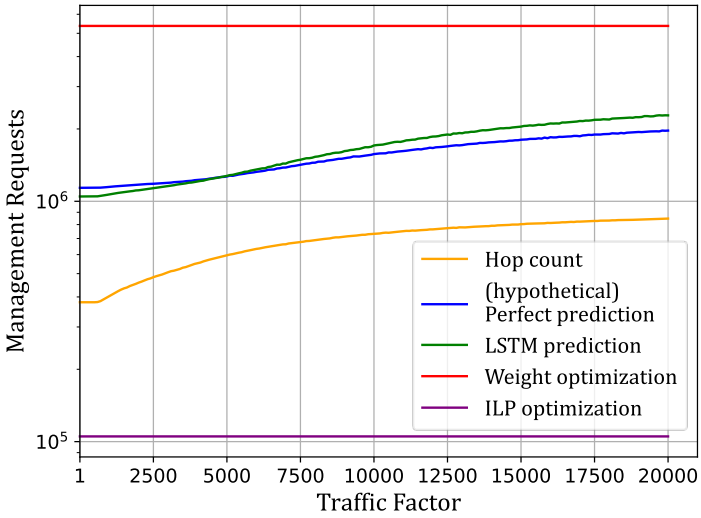

Abb. 8: Anzahl der Managementanfragen an die verschiedenen Schlüsselspeicher durch den Netzcontroller in Abhängigkeit zum Skalierungsfaktoren.

Managementverkehr aufweist, da die Speicherstände für jede Routingentscheidung abgefragt werden müssen. Auf diese Weise kann die Angriffsfläche minimiert werden und somit die Resilienz des gesamten Netzes erhöht werden. Gleichzeitig kann ein besseres Ergebnis mit Bezug auf die Anzahl der blockierten Anfragen erzielt werden, falls eine größere Datengrundlage zur Verfügung stehen würde.

3.1.2.2.2.2 Dezentrales Routing und Lastverteilung mittels Deep Reinforcement Learning

Im vorherigen Abschnitt wurde ein klassischer Routingalgorithmus mit Methoden des maschinellen Lernens verbunden. Mittels Deep Reinforcement Learning (DRL) wurde ein Routingagent entwickelt, welcher ohne konventionelle Algorithmen wie den Dijkstra-Algorithmus, Routingentscheidungen treffen kann.

Wie bereits im vorherigen Abschnitt beschrieben, kann der Netzcontroller ebenfalls das Ziel von Angriffen werden, um ein QKD-Netz zu manipulieren. Eine Alternative zu einer zentralen Kontrollinstanz bietet das sogenannte dezentrale Routing. Hier werden Entscheidungen direkt in den jeweiligen Knoten getroffen, ohne über eine zentrale Datenbank mit Netzparametern wie den Speicherfüllständen zu verfügen. Die Entscheidungen basieren daher lediglich auf lokalen Parametern, um die Angriffsfläche auf das QKD-Netz zu minimieren. Traditionelle gewichtete kürzeste Pfad-Algorithmen, wie der Dijkstra-Algorithmus, sind aber stark von der Kenntnis der Netzparameter abhängig, da ansonsten der Pfad iterativ nach jedem Schritt zum nächsten Knoten neu berechnet werden muss. DRL kann hier Abhilfe schaffen.

Simulationssetup

Das Reinforcement Learning basiert auf der Umwelt (engl. Environment) und dem Agenten. Der Agent macht Beobachtungen von der Umwelt und führt auf Basis der Beobachtungen Aktionen aus. Diese werden anschließend belohnt oder bestraft, wodurch der Agent nach einer Vielzahl von Iterationen sich auf eine bestimmte Art und Weise verhält. Folglich lernt er selbständig, Entscheidungen zu treffen. Auf diese Weise kann durch DRL das Routing ohne konventionelle Algorithmen durchgeführt werden.

Der Zustand der Umwelt wird an den Agenten übermittelt. In diesem Fall handelt es sich hierbei um eine gewichtete Nachbarschaftsmatrix der Nobel-Germany Topologie, wobei die Gewichtungen auf den prozentualen Speicherfüllständen der verschiedenen Verbindungen basieren. Des Weiteren bekommt der Agent Informationen über die derzeitige Position im Netz (z.B. Startknoten) und das Ziel der Schlüsselanfrage. Außerdem enthält der Zustand die Informationen, wie viel Prozent des maximalen Schlüsselspeichers von der aktuellen Anfrage konsumiert wird. Während des Trainings kennt der Agent alle Speicherfüllstände des gesamten Netzes und wird erst in der Anwendung auf die lokalen Informationen der jeweiligen Knoten beschränkt. Auf diese Weise lernt der Agent wichtige Zusammenhänge über das Netz. Basierend auf dem Zustand der Umwelt führt der Agent eine Aktion aus, was in diesem Fall die Wahl des nächsten Knotens in Richtung des Zielknotens ist. Zentral für das DRL ist die Entwicklung der Belohnungsfunktion, da diese das Verhalten des Agenten steuert. Das oberste Ziel ist es, für den Agenten die zu erwartende Belohnung für eine Aktion zu maximieren. Im Folgenden wird die Zusammensetzung der Belohnungsfunktion hergeleitet.

$$KL = \frac{\sum_E \left(p_E \cdot log_2 \frac{p_E}{q_E} \right)}{log\, 2}$$

Bei *KL* handelt es sich um die Kullback-Leibler Divergenz, welche die relative Entropie zwischen zwei Verteilungen berechnet, um den Unterschied zwischen den beiden Verteilungen zu bestimmen. Bei p_E

handelt es sich um einen Vektor, der die Füllstände aller Speicher beinhaltet. Die komplett gefüllten Speicher werden in Vektorform von q_E repräsentiert. Beide Vektoren haben die Länge E, was der Anzahl der Schlüsselspeicher entspricht. Zum Start eines Routingprozesses für einen Ende-zu-Ende Schlüssel wird eine initiale Belohnung berechnet, die den aktuellen Zustand des Netzes beschreibt. Sie berechnet sich aus:

$$B_{init} = (1 - KL) \cdot \sum_E p_E.$$

Die Berechnung des initialen Zustandes des Netzes hilft dem Agenten, die Auswirkungen seiner Entscheidungen besser zu verstehen. Sobald der Agent das Ziel erreicht, berechnet sich der zweite Teil der Belohnung wie folgt:

$$B = (1 - 1.1 \cdot \ KL) \cdot \sum_E p_E.$$

Durch den Vorfaktor wird der Einfluss der, nach dem Routingprozess vorhandenen, Entropie erhöht, was sich positiv auf die Lastverteilung bei den Routingentscheidungen auswirkt.

$$R = \begin{cases} -1, & \text{wenn Aktion invalide} \\ 0, & \text{wenn Aktion valide} \\ B - 0.9 \cdot B_{init}, & \text{wenn Ziel erreicht} \end{cases}$$

Die Belohnung des Agenten lässt sich abschließend in drei unterschiedliche Fälle aufteilen. Falls der Agent eine unerlaubte Entscheidung trifft (z.B. direkt zu einem nicht benachbarten Knoten springt oder eine Kante ohne ausreichende Schlüssel nutzt), wird der Agent bestraft und die Trainingsepisode endet. Trifft der Agent eine valide Entscheidung, die ihn nicht zum Zielknoten bringt, wird er weder belohnt noch bestraft. Die Belohnung wird lediglich beim Erreichen des Zielknotens ausgezahlt. Auf diese Weise lernt der Agent, sich auf die gewünschte Art und Weise zu verhalten, ohne die Aktionen vorgegeben zu bekommen.

Während des Trainings werden die Füllstände zufällig zwischen 0,1% und 100% initialisiert, um dem Agenten eine Vielzahl von Netzzuständen zu zeigen. Es werden über 10 Millionen Zeitschritte in 32 parallelen Umgebungen trainiert. In jeder Trainingsepisode wird zufällig eine Anfrage für einen Ende-zu-Ende Schlüssel zwischen einem Start- und einem Zielknoten erzeugt. Eine solche Anfrage benötigt zwischen 0,1% und 1% des maximalen Speicherfüllstandes einer Kante.

Für den Vergleich mit traditionellen Routingalgorithmen werden zwei kürzeste Pfad Algorithmen implementiert. Als erster Algorithmus wird erneut ein simpler „hop-basierter“ Algorithmus genutzt, der lediglich die Anzahl der passierten Knoten zwischen Start und Ziel minimiert. Der zweite Algorithmus ist eine dezentralisierte Variante des Dijkstra-Algorithmus. Da lediglich die Speicherfüllstände der direkten Nachbarschaft für den Controller bekannt sind, muss der Pfad nach jedem Schritt neu berechnet werden, da sich die Nachbarschaft mit jedem Schritt verändert. Die Kantengewichtungen sind erneut basierend auf den aktuellen Speicherfüllständen gewählt.

Ergebnisse

Für die Evaluierung werden in jedem Zeitschritt zufällige Schlüsselanfragen zwischen allen 17 Knoten des Netzes generiert, wobei jede Anfrage zwischen 0% und 1% des maximalen Schlüsselspeicherstandes benötigt. Dies resultiert in maximal 272 Anfragen pro Zeitschritt. Im Anschluss an die Bearbeitung der Anfragen eines Zeitschrittes werden neue Schlüssel generiert und die Speicher aufgefüllt. Auf diese

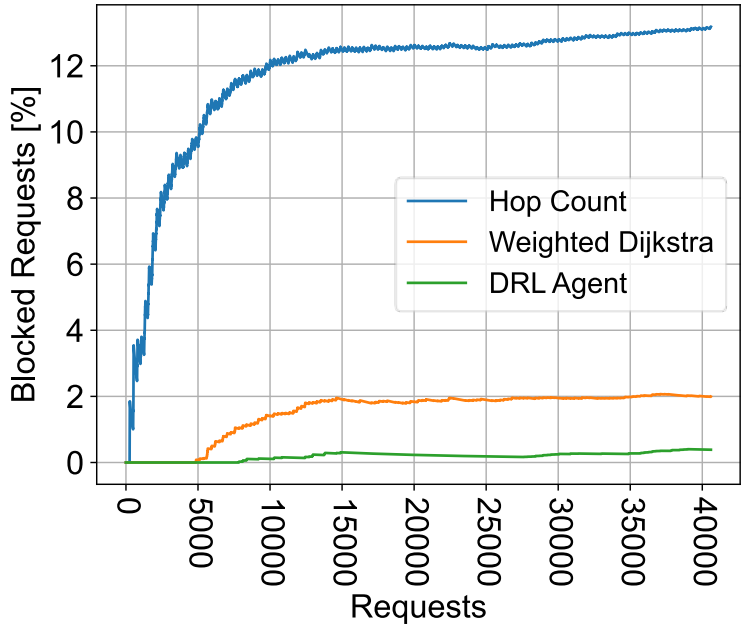

Abb. 9: Blockierte Anfragen in Relation zu den gesamten bearbeiteten Anfragen

Weise soll, bei einer ausgewogenen Verteilung der Last auf die Kanten, eine Regeneration der Schlüssel-Speicher ermöglicht werden.

Abb. 9 zeigt, dass der „Hop-basierte“ Algorithmus sehr schnell Anfragen blockiert, da die Schlüssel-Speicher durch eine häufige Benutzung schnell überlastet werden. Dies impliziert, dass einige Knoten nicht mehr erreichbar sind. Der gewichtete Dijkstra-Algorithmus arbeitet deutlich besser, aber durch die limitierten Kenntnisse über die Schlüsselspeicher von Knoten, die sich nicht in direkter Nachbarschaft befinden, werden ebenfalls Anfragen blockiert. Der DRL-Agent ist durch sein zentrales Training in der Lage, potentielle Engpässe in der Topologie zu erkennen und verteilt die Last vorausschauend auf andere Kanten, so dass nur sehr wenige Anfrage blockiert werden.

In Abb. 10 sind die Speicherauslastungen der wichtigsten Verbindungen der Topologie über die komplette Simulationsdauer für eine Verdeutlichung dargestellt. Es ist deutlich zu erkennen, dass der Hop-basierte Algorithmus Kanten, die oft auf dem kürzesten Pfad liegen, stark auslastet und andere Kanten nahezu ignoriert. Auf diese Weise entstehen Engpässe, und die Ressourcen werden nicht gleichmäßig genutzt. Der gewichtete Dijkstra-Algorithmus verteilt die Last deutlich besser, aber ist ebenfalls nicht in der Lage Engpässe zu verhindern, da er sich strikt an seine Gewichtungsfunktion hält. Außerdem

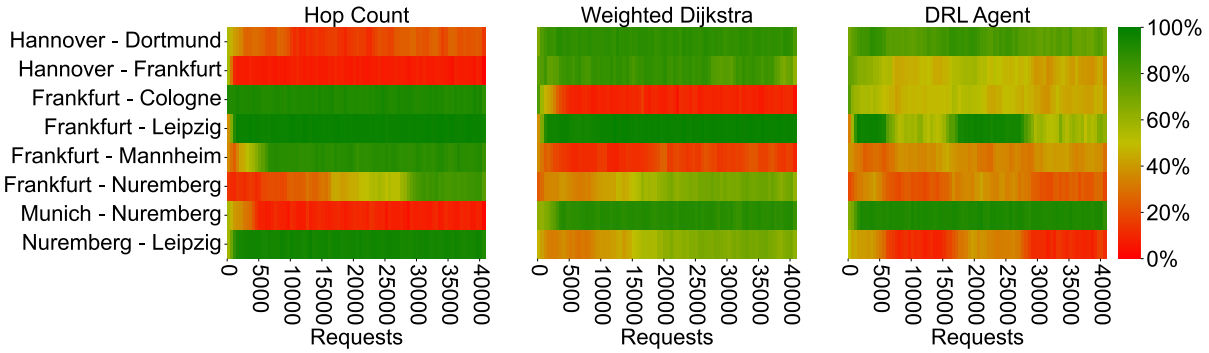

Abb. 10: Heatmap über die Speicherfüllstände der interessantesten Kanten im Netz in Relation zu den Anfragen.

wirkt sich die iterative Neuberechnung des Pfades negativ auf die Leistungsfähigkeit aus. Der DRL-Agent profitiert bei der dezentralen Anwendung von seinem zentralen Training. So können die Agenten potentielle Engstellen identifizieren und präventiv andere Routen wählen. Dies führt zu einer deutlichen Entlastung wichtiger Kanten, wodurch diese mit einer geringeren Wahrscheinlichkeit blockiert werden.

3.1.2.2.2.3 Zentrales Topologie-unabhängiges Routing mittels Deep Reinforcement Learning und Graph Neural Networks

Der Nachteil des zuvor vorgestellten Algorithmus liegt in seiner Abhängigkeit von der Trainingstopologie. Der Agent kann zwar dezentral Routingentscheidungen treffen, aber er ist auf eine einzelne Topologie beschränkt. Für den Fall, dass eine größere Flexibilität und eine zentrale Routinginstanz gewünscht sind, wurde ein weiterer DRL Agent entwickelt, der zentrales Routing unabhängig von der Topologie durchführen kann. Auf diese Weise kann der Agent auf Veränderungen der Topologie reagieren, die z.B. durch Verbindungsausfälle auftreten oder falls einige Knoten lediglich temporär zur Verfügung stehen (z.B.: Satelliten, Flugzeuge). In diesem Fall wird ein zentraler SDN Controller angenommen, der neben der Topologie auch Kenntnisse über die Speicherfüllstände in Echtzeit besitzt. Die Gewichtungsfunktion bei traditionellen Routingalgorithmen sind teilweise stark von der Topologie abhängig und müssen bei Veränderungen in der Topologie manuell angepasst werden. In diesen ersten Betrachtungen wird sich auf Topologien mit 10 Knoten beschränkt.

Simulationssetup

Das Training eines Agenten mit Daten, die als Graph dargestellt werden, ist äußerst komplex. Die Darstellungsweise als Nachbarschaftsmatrix ist, wie zuvor beschrieben, zwar möglich aber nicht sehr effizient, da viele redundante bzw. unwichtige Informationen enthalten sind. Hierdurch ist es für einen Agenten schwer, über zufällige Topologien zu generalisieren. Trotzdem werden von den konventionellen Python-Bibliotheken, wie z.B. stable-baselines3, Graphen als Beobachtungsraum nicht nativ unterstützt. Aus diesem Grund wurde ein eigener Features Extractor geschrieben, der die Beobachtungen aus der Umgebung des Agenten vorverarbeitet. Das neuronale Netz, welches als Features Extractor genutzt wird, wird ebenfalls im Trainingsprozess trainiert.

Für den Features Extractor wird ein Graph Attention Network (GAT) genutzt, welches zu den Graph Neural Networks gehört. Diese neuronalen Netze sind in der Lage, Daten zu verarbeiten, die wie Graphen strukturiert sind, da sie die Verbindungen (Kanten) der verschiedenen Knoten berücksichtigen können. Insbesondere kann das GAT Kantenfeatures direkt verarbeiten. Auf diese Weise können die Schlüsselspeicherstände zwischen den jeweils benachbarten Knoten direkt als Kantenfeature verarbeitet werden.

Aus der Umwelt des Agenten können daher die Beobachtungen in Form von Knoten- und Kantenfeatures und einer Liste der Kanten direkt vom Features Extractor verarbeitet werden. Insgesamt werden drei GAT-Layer genutzt. In dem ersten Layer kommunizieren lediglich direkt benachbarte Knoten miteinander. Mit einem weiteren Layer wird dieser Informationsaustausch um diejenigen Knoten erweitert, die sich in der direkten Nachbarschaft der aktuellen Nachbarn befinden. Dies ist exemplarisch in Abb. 11 dargestellt. In einem Netz im zehn Knoten sollten daher in der Regel drei Layer reichen, um den Informationsaustausch auf das gesamte Netz auszuweiten.

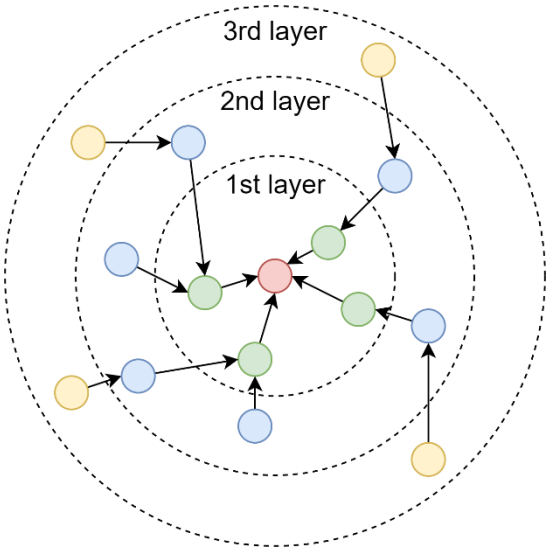

Abb. 11: Exemplarische Darstellung des Einflusses mehrerer Schichten auf den Informationsaustausch in einem Graph Neural Network

Da stable-baseline3 einen konstanten Beobachtungsraum erwartet, müssen die zufällig generierte Trainingsgraphen einige Voraussetzungen erfüllen. Zuerst werden die Graphen, wie eingangs bereits erwähnt, auf zehn Knoten beschränkt, um die Problemstellung übersichtlich zu halten. Da auch die Vektoren für die Kantenfeatures eine konstante Dimension aufweisen muss, wird der minimale Grad auf zwei und maximale Grad eines Knotens auf fünf Verbindungen festgelegt. Daraus ergeben sich für jeden zufälligen Graphen maximal 50 Kanten. Sollte ein Graph weniger Kanten besitzen, werden die Vektoren für die Features und die Kantenliste mit Nullen aufgefüllt. Dies gewährleistet die Kompatibilität mit der stable-baseline3 Bibliothek aus Python.

Während des Trainings ist es die Aufgabe des Agenten, zufällige Anfragen für Ende-zu-Ende Schlüssel zu routen und dies in ebenfalls zufälligen Topologien (siehe exemplarisch Abb. 12). Auf diese Weise soll der Agent eine robuste Regel für seine Entscheidungen entwickeln und ihn so flexibel einsetzbar machen. Die Füllstände werden zufällig zwischen 0,1% und 100% initialisiert. Eine einzelne Anfrage benötigt zwischen 1% und 2% des maximalen Speicherfüllstandes. Der Einfachheit halber wird eine einheitliche Größe der Schlüsselspeicher angenommen.

Die Belohnungsfunktion wurde im Vergleich zum dezentralen Ansatz weiter optimiert, um ein besseres Routingergebnis zu erzielen. Grundsätzlich kann die Belohnung auch hier in drei unterschiedliche Fälle aufgeteilt werden.

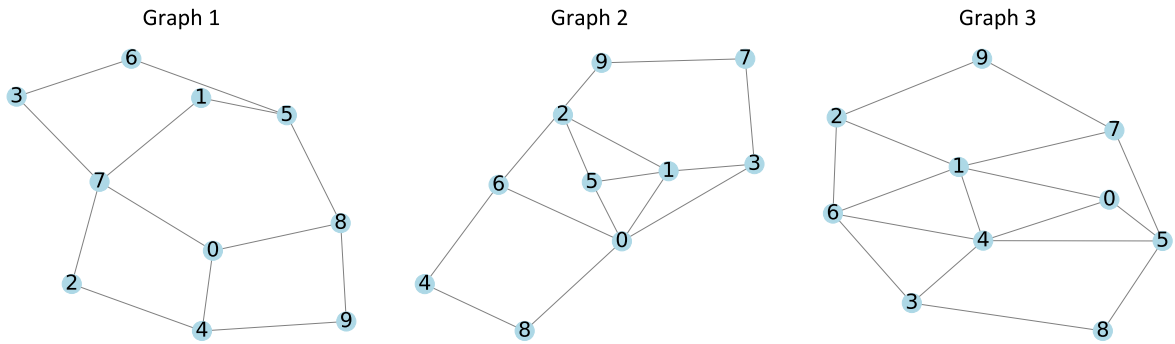

Abb. 12: Exemplarische Darstellung von drei zufälligen Trainingstopologien.

$$R_a = \begin{cases} -1, & wenn\ a\ invalide \\ 0, & wenn\ a\ \text{valide} \\ R', & wenn\ \text{Ziel erreicht} \end{cases}$$

1.) Wenn eine invalide Aktion ausgeführt wird, wie die Nutzung einer Kante mit unzureichender Schlüsselmenge oder der direkte Sprung zu einem nicht benachbarten Knoten, wird der Agent bestraft.
2.) Für valide Aktionen, die nicht zum Zielknoten führen gibt es weder eine Belohnung noch eine Bestrafung.
3.) Falls das Ziel erreicht wird, wird die Belohnung R' ausgezahlt.

$$R' = 1 - \left(a\big(D(f_{t-1}) - D(f_t)\big) - b\left(\sum_i f_{t,i} - \sum_i f_{t-1,i}\right)\right)$$

Die Belohnung setzt sich prinzipiell aus zwei Teilen zusammen. Der erste Teil, der sich mit dem Skalierungsfaktoren a gewichten lassen kann, belohnt die gleichmäßige Auslastung der Schlüsselspeicher. Zu diesem Zweck wird die Kullback-Leibler-Divergenz (KL-Divergenz) der Speicherfüllstände vor dem Routingprozess einer Anfrage berechnet und davon wird die KL-Divergenz nach Abschluss des Routingvorganges abgezogen. Auf diese Weise lernt der Agent, wie sich die Routingentscheidung auf die Verteilung der Last im Netz auswirkt. Der zweite Teil der Gleichung, skaliert mit dem Faktor b, berechnet die konsumierte Schlüsselmenge für den gewählten Pfad. Dies verhindert die Verschwendung wichtiger Schlüsselressourcen durch unnötig lange Pfade.

Für die Evaluierung wurde die NOBEL-Germany Topologie um einige Knoten reduziert, um den Anforderungen des Agenten zu entsprechend. Die reduzierte Variante der Topologie ist in Abb. 13

Abb. 13: Reduzierte Nobel-Germany Topologie

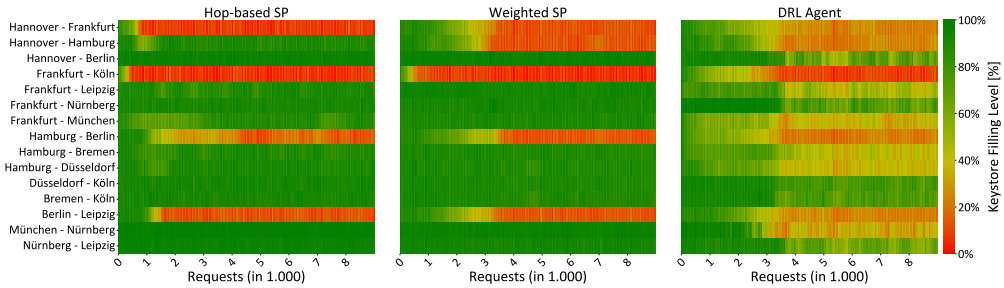

Abb. 14: Heatmap der verschiedenen Kanten im Netz über die Dauer der Simulation.

dargestellt. Diese Topologie wurde während des Trainings nicht verwendet und ist für den Agenten somit unbekannt. Für den Vergleich der Routingperformance wurden wieder ein gewichteter Dijkstra-Algorithmus und die „hop-basierte“ Metrik verwendet. Während des Routings sind die Speicherfüllstände und die Konnektivität des Graphen für jeden Algorithmus bekannt.

Ergebnisse

Abb. 14 zeigt die Auslastungen der verschiedenen Schlüsselspeicher im Netz. Es zeigt sich deutlich, dass der hop-basierte Algorithmus schnell vier Schlüsselspeicher leert und erst anschließend auf alternative Routen zurückgreift. Auf diese Weise sind diese Kanten kurzzeitig blockiert und können erst nach einer Regeneration des Schlüsselspeichers wieder genutzt werden. Der gewichtete Shortest-Path-Algorithmus kann dies länger herauszögern, aber auch hier werden einige Verbindungen bis zur Blockierung genutzt. Dies impliziert, dass die Kantengewichtungsfunktion in diesem Graphen die Last nicht ausreichend balancieren kann und somit manuell angepasst werden müsste. Auf diese Weise kann gezeigt werden, dass sich Veränderungen in der Topologie stark auf die Qualität der Gewichtungsfunktion auswirken können. Der DRL-Agent lastet zwar Speicher stärker aus, die von den anderen Algorithmen weniger Beachtung fanden, aber sie weisen meist einen Speicherstand von ca. 40% aus, was einen reibungslosen Betrieb des Netzes trotzdem gewährleistet. Überlastungen können deutlich verzögert werden bzw. weisen stark ausgelastete Kanten trotzdem einen höheren Restspeicherstand auf als die beiden Vergleichsalgorithmen. Dies wird durch Abb. 15 verdeutlich, in der vier der kritischsten Kanten aufgeführt sind. Es zeigt sich hier, dass der „hop-basierte“ Algorithmus die Speicher entweder sehr schnell leert oder kaum benutzt. Der gewichtete Algorithmus verzögert die Vollauslastung deutlich, aber erreicht ebenfalls eine starke Auslastung der Speicher. Der DRL-Agent schafft es in den meisten Fällen, den minimalen Füllstand der Speicher auf einem höheren Niveau (ca. 25%) zu halten und somit eine vollständige Blockierung der Verbindung zu verhindern. Bei der Verbindung zwischen

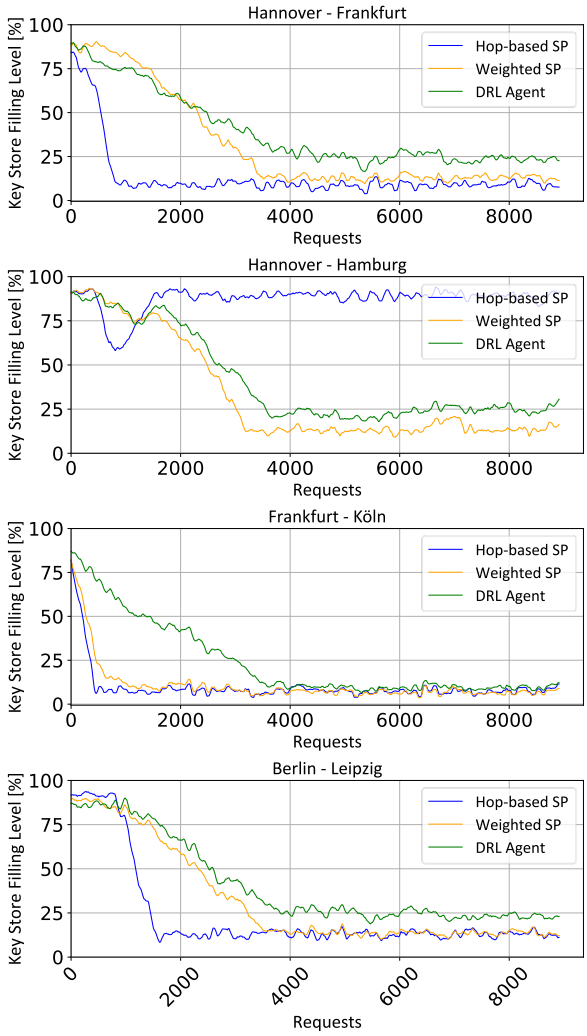

Abb. 15: Detaillierte Betrachtung der viel kritischsten Speicher über die Anzahl der Gesamtzahl der Anfragen.

Frankfurt und Köln erreicht der Agent eine erhebliche Verzögerung der Vollauslastung im Vergleich zu den beiden anderen Algorithmen.

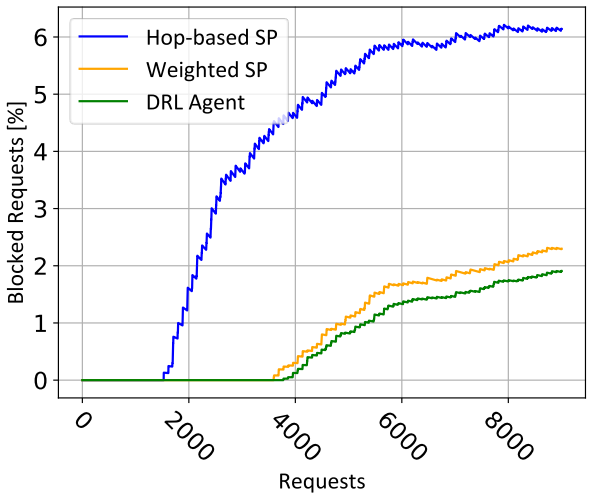

Abb. 16: Anzahl der blockierten Anfragen in Relation zu den gesamten Anfragen.

Abschließend zeigt Abb. 16 die prozentuale Anzahl der blockierten Anfragen in Relation zu der Gesamtzahl der Anfragen von Ende-zu-Ende Schlüsseln. Es zeigt sich klar, dass durch die fehlende Berücksichtigung der Speicherfüllstände, der „hop-basierte“ Algorithmus, wie bereits zuvor erläutert, schnell Engpässe verursacht und einige Knoten auf diese Weise nicht mehr erreichbar sind, bis ausreichend Schlüssel nachproduziert wurden. Die bessere Lastverteilung des DRL-Agenten spiegelt sich auch in der geringeren Zahl der blockierten Anfragen wider. Des Weiteren treten die Blockierungen auch erst zu einem späteren Zeitpunkt auf.

3.1.2.2.3 Sicherheit der künstlichen Intelligenz

Die Sicherheit der KI muss für jeden der entwickelten Algorithmen separat betrachtet werden, da sowohl für das Training als auch für die Anwendungen unterschiedliche Informationen über das Netz, in dem sie arbeiten, benötigt werden.

3.1.2.2.3.1 Datenverkehrsvorhersage mittels Long Short-Term Memory

Dieser Algorithmus benötigt Zugriff auf eine Datenbank, die die verschiedenen Datenverkehrsströme beinhaltet. In diesem Fall ist die Aktualität der Daten ebenfalls von Relevanz. Wird diese Datenbank korrumpiert, kann dies zu erheblichen Beeinträchtigungen des Algorithmus führen, da die Kantengewichtung auf Basis von fehlerhaften Vorhersagen berechnet würde. Aus diesem Grund ist ein Schutz dieses Datenverkehrs von erheblicher Bedeutung.

3.1.2.2.3.2 Dezentrales Routing und Lastverteilung mittels Deep Reinforcement Learning

Obwohl das Training des Agenten zentral erfolgt, ist eine Beeinflussung durch eine dritte Partei stark erschwert, da das Training des Agenten offline passiert und die Routingentscheidung lediglich auf Basis der Topologie und der Füllstände erfolgt. Während des Betriebes des Netzes würde sich in jedem Knoten ein Agent befinden, der unabhängig von den anderen Agenten eine Routingentscheidung trifft. Durch das Training auf eine explizite Topologie könnte lediglich eine dauerhafte Beeinflussung einzelner Verbindung zu Problemen führen, jedoch würde sich eine solche Attacke nicht auf den Agenten oder eine zentrale Datenbank beziehen. Es sei denn, es würde sich um einen allgemeinen Angriff auf die Netzstruktur handeln, der auch andere konventionelle Algorithmen in ähnlicher Weise betreffen würde.

3.1.2.2.3.3 Zentrales Topologie-unabhängiges Routing mittels Deep Reinforcement Learning und Graph Neural Networks

Der Agent wurde, wie auch der dezentrale Agent, offline trainiert und trifft Entscheidungen lediglich auf Basis der aktuellen Speicherfüllstände. Während des Trainings werden keine Speicherhistorien betrachtet und auch keine zeitabhängigen Nachfragematrizen. Daher ist das Training des Agenten nicht sonderlich kritisch und das Risiko der Manipulation des Trainings eher gering. Durch die Generalisierungsfähigkeit über zufällige Topologien können auch Verbindungsausfälle und die damit einhergehende temporäre Veränderung der Netztopologie durch den Agenten kompensiert werden. Der Agent trifft jedoch seine Routingentscheidung auf Basis von Echtzeitfüllständen der verschiedenen Speicher im Netz. Sollten diese Informationen korrumpiert werden, beeinflusst dies auch die Qualität der Routingentscheidungen des Agenten.

3.2 Zahlenmäßiger Nachweis

Tabelle 2 gibt einen Überblick über die geplanten Kosten und die tatsächlich entstandenen Kosten des Projekts.

Tab. 2: Vergleich der geplanten und angefallenen des Projekts (ohne Projektpauschale)

	Gesamt kosten (geplant)	Angefallene Kosten	Differenz	Differenz in %
0812 Beschäftige E12-E15	233.588,04	261.486,96	27.898,92	11,9
0822 Beschäftigungsentgelte	24.024,00	2.891,46	-21.132,54	-88
0843 Sonst. allg. Verw.-ausgaben		2.345,24	2.345,24	100
0846 Dienstreisen	14.250,00	4.373,69	-9.876,31	-69,3
0850 Gegenstände & andere Investitionen	7.245,91	6.889,94	-355,97	-4,9
gesamt	**279.107,95**	**277.987,29**	**-1.120,66**	**-0,4**

Tab. 3: Vergleich des geplanten und angefallenen Arbeitsaufwands im Verlauf des Projekts.

	Arbeitsaufwand in Personenmonaten		
Quartal	Plan zu Projektbeginn	Kostenneutrale Verlängerung ab 01/2025	Tatsächlicher Aufwand
Q1/2022	3	39	0
Q2/2022	3		2
Q3/2022	3		3
Q4/2022	3		3
Q1/2023	3		3
Q2/2023	3		3
Q3/2023	3		3
Q4/2023	3		5
Q1/2024	3		5
Q2/2024	3		6
Q3/2024	3		6
Q4/2024	6		3
Q1/2025		3	3
Q2/2025		3	3
gesamt	**39**	**45**	**48**

3.3 Notwendigkeit und Angemessenheit der Arbeit

Die im Projekt geleisteten Arbeiten waren auf die Entwicklung neuartiger KI-basierter Verfahren für das Routing in vermaschten QKD-Netzen ausgerichtet. Es wurden verschiedene Verfahren implementiert und evaluiert, sowohl in Hinblick auf ihre Blockierungswahrscheinlichkeit (bzw. Lastverteilung) als auch auf die Sicherheit gegen externe Attacken. Die Untersuchungen und Entwicklungen unterstützten die Gesamtheit des Projektziels und ermöglichten durch die entwickelten Verbesserungen die Erreichung des Projektziels.

Der Aufwand, der im Projekt geleisteten Arbeit, war unter Berücksichtigung des komplexen Themas angemessen.

3.4 Nutzen

Die im Verlaufe des Projektes gewonnen Ergebnisse wurden zur Ausbildung benötigter Fachkräfte im Hinblick auf Seminararbeiten, Bachelorabschlüssen und für eine Promotion verwendet. Die Promotion ist in Vorbereitung.

3.5 Veröffentlichungen

Folgende Ergebnisse des Projektes wurden in Zeitschriften und Tagungsbänden veröffentlicht:

[1] Johann, T., Kühl, S., & Pachnicke, S. (2025). (Invited) Deep Reinforcement Learning Based Topology Independent Routing and Load-Balancing in QKD Networks using Graph Neural Networks. Beitrag in International Conference on Transparent Optical Networks (ICTON).

[2] Johann, T., Wenning, M., Giemsa, D., Dochhan, A., Gunkel, M., Fehenberger, T., & Pachnicke, S. (2024). Comparison and optimization of different routing methods for meshed QKD networks using trusted nodes. Journal of Optical Communications and Networking, 16(3), 382-391. https://doi.org/10.1364/JOCN.509269

[3] Johann, T., Kühl, S., & Pachnicke, S. (2024). Deep Reinforcement Learning based Decentralized Routing and Load-Balancing in Meshed QKD-Networks. In J.-P. Elbers, R. Freund, & C. Mas-Machuca (Hrsg.), ECOC 2024: 50th European Conference on Optical Communication 22.-26. September 2024, Congress Center Messe Frankfurt Artikel W1A.3 (ITG-Fachbericht; Band 317). VDE Verlag GmbH.

[4] Johann, T., Giemsa, D., & Pachnicke, S. (2023). Routing Optimization of QKD-Networks using Machine-Learning Based Prediction. In ITG Conference "Photonic Networks" (S. 1-5). VDE.

Berichtsblatt

1. ISBN oder ISSN	2. Berichtsart (Schlussbericht oder Veröffentlichung) Schlussbericht
3. Titel Schlussbericht zu QuNET+ML Verbundprojekt: Optimierung von QKD-Netzen mittels maschinellen Lernens Teilvorhaben: Systemoptimierung und Pfadwahl in QKD-Netzen unter Einsatz von maschinellem Lernen	
4. Autor(en) [Name(n), Vorname(n)] Pachnicke, Stephan	5. Abschlussdatum des Vorhabens 30.06.2025
	6. Veröffentlichungsdatum
	7. Form der Publikation
8. Durchführende Institution(en) (Name, Adresse) Christian-Albrechts-Universität zu Kiel Lehrstuhl für Nachrichtenübertragungstechnik Kaiserstr. 2, 24143 Kiel	9. Ber. Nr. Durchführende Institution
	10. Förderkennzeichen 16KIS069
	11. Seitenzahl 24
12. Fördernde Institution (Name, Adresse) Bundesministerium für Forschung, Technologie und Raumfahrt (BMFTR) 53170 Bonn	13. Literaturangaben 4
	14. Tabellen 3
	15. Abbildungen 16
16. Zusätzliche Angaben Eine Veröffentlichung des Schlussberichts in der vorgelegten Form ist nicht geplant. Jedoch wurden die Ergebnisse des Projekts in zahlreichen Veröffentlichungen publiziert, die im Bericht referenziert sind.	
17. Vorgelegt bei (Titel, Ort, Datum)	
18. Kurzfassung Das Projekt QuNET+ML ist Teil der QuNET-Initiative. Die verschiedenen Plus-Projekte adressieren hierbei die Fehlbedarfe des QuNET-Kernprojektes. Ziel von QuNET+ML war es, den Einsatz von Quantum Key Distribution (kurz: QKD) in realistischen Netzszenarien zu untersuchen in Verbindung mit maschinellem Lernen für das Routing. Das Interesse lag insbesondere auf vermaschten Netzstrukturen mit dynamischen Lastanforderungen. Bei der Realisierung eines vermaschten QKD-Netzes ist, für eine erhöhte Ausfallsicherheit, die Berücksichtigung von Resilienzverfahren besonders wichtig. Darüber hinaus muss in großen Netzen ebenfalls von heterogenen Systemen ausgegangen werden, wodurch unter anderem unterschiedliche QKD-Verfahren, Protokolle, etc. innerhalb eines Netzes eingesetzt werden. Hieraus folgt, dass die Optimierung der verschiedenen Systemparameter äußerst herausfordernd sein wird und mit klassischen Optimierungsverfahren nicht gelöst werden können. Daher wurden verschiedene Methoden des maschinellen Lernens (ML) untersucht. Diese Methoden wurden in einem Demonstrator integriert, um mit Hilfe der Industriepartner eine zügige Kommerzialisierung voranzutreiben. Die wesentlichen Ergebnisse der Christian-Albrechts-Universität (CAU) sind wie folgt zusammenzufassen: • Mitwirkung an der Definition der Netzarchitektur • Ausarbeitung von Vergleichsalgorithmen, um die Ergebnisse der ML-Algorithmen zu validieren • Entwicklung eines ML-Algorithmus zu einer vorhersagebasierten Routingoptimierung unter der Nutzung klassischer Shortest-Path-Algorithmen (SP-Algorithmen) • Entwicklung eines Deep Reinforcement Learning (DRL) Agenten für dezentrales Routing ohne die Verwendung klassischer SP-Algorithmen • Entwicklung eines DRL Agenten für zentrales Routing ohne die Verwendung klassischer SP-Algorithmen • Bewertung der Sicherheit der verschiedenen ML-Algorithmen	
19. Schlagwörter Sichere Kommunikation, Quantum Key Distribution, maschinelles Lernen, Routingoptimierung	
20. Verlag	21. Preis