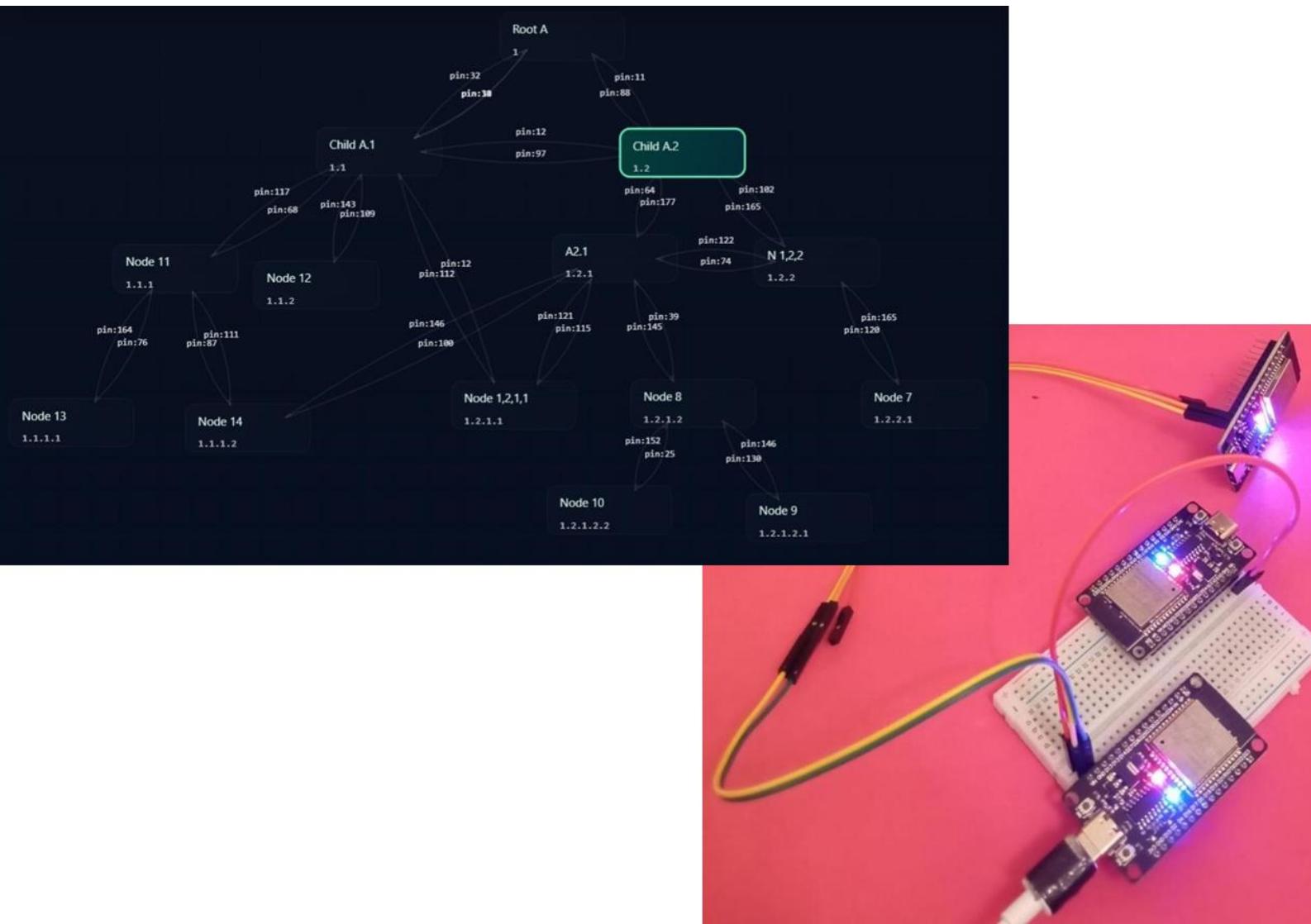


LHRP – Lightweight Hierarchical Routing Protocol



Von Manuel Westermeier im Jahr 25/26

Demo für die Logische Umsetzung des Routings:

<https://manuelwestermeier.github.io/LHRP/>

Inhaltsverzeichnis

1 Einleitung

- 1.1 Fachliche Kurzfassung
- 1.2 Motivation und Fragestellung

2 Idee

- 2.1 Hintergrund und theoretische Grundlagen
- 2.2 Vorgehensweise, Materialien und Methoden

3 Ergebnisse

- 3.1 Ergebnisse
- 3.2 Ergebnis Diskussion

4 Schluss

- 4.1 Ausblick
- 4.2 Fazit
- 4.3 Quellen- und Literaturverzeichnis
- 4.4 Umgang mit KI

1 Einleitung

1.1 Fachliche Kurzfassung

Das LHRP (Lightweight Hierarchical Routing Protocol) ist ein leichtgewichtiges Routing-Protokoll, das auf der Vermittlungsschicht des OSI-Modells arbeitet. Es übernimmt die effiziente Übermittlung von Daten zwischen Netzwerknoten, basierend auf einer baumstrukturierten, hierarchischen Adressverteilung. Durch die optimierten Abkürzungsmechanismen und minimalem Rechen- und Speicherbedarf erreicht LHRP eine hohe Effizienz auf ressourcenbeschränkten Systemen wie dem ESP32. Dabei habe ich auch einen großen Fokus auf Verschlüsselung und Sicherheit gesetzt.



https://www.techbuyer.com/media/wysiwyg/DE/DE_OSI_Refrence_Model_Diagram_.png

1.2 Motivation und Fragestellung

Ressourcenbeschränkte Mikrocontroller wie der ESP32 benötigen ein Routing-Verfahren, das trotz begrenzter Rechenleistung, geringem Speicher und instabilen Netzwerktopologien zuverlässig arbeitet. Klassische Routing-Protokolle sind für solche Umgebungen oft zu komplex oder ressourcenintensiv. Daher besteht die Motivation darin, ein minimalistisches, aber leistungsfähiges Routing-Protokoll zu entwickeln, das speziell auf kleine Embedded-Netzwerke und IoT-Strukturen zugeschnitten ist. LHRP soll eine robuste, skalierbare und extrem effiziente Datenübertragung ermöglichen, ohne die Hardware zu überfordern.

Wie kann ein Routing-Protokoll gestaltet werden, das auf dem ESP32 mit minimalem Speicher- und Rechenaufwand arbeitet, gleichzeitig aber eine zuverlässige und schnelle Datenübertragung gewährleistet?

Insbesondere stellt sich die Frage, inwiefern eine hierarchische, baumartige Netzwerkstruktur und optimierte Abkürzungsmechanismen genutzt werden können, um Routing-Effizienz, Stabilität und Skalierbarkeit in dynamischen, ressourcenarmen Netzwerken zu maximieren.

2 Idee

2.1 Hintergrund und theoretische Grundlagen

ESP-NOW ist ein leichtgewichtiges Protokoll für kurze, schnelle Paketübertragungen zwischen ESP-Geräten. Routing in Mesh-Netzen kann durch verschiedene Ansätze erfolgen (Link State, Distance Vector, Random Walk, Flooding, Centralized). LHRP verfolgt stattdessen ein hierarchisches Präfix-Matching: Adressen sind Vektoren (z. B. $\{1,1,1\}$), die Hierarchieebenen repräsentieren. Entscheidungen beruhen auf längstem gemeinsamem Präfix und Adresslängen, was zu der Größe Match-Index mit positiver und negativer Zusammensetzung führt, wodurch deterministische Next-Hop-Auswahl möglich wird.

2.2 Vorgehensweise, Materialien und Methoden

Materialien:

- ESP32-Module (Arduino-Framework, PlatformIO)
- ESP-NOW als Transport
- Entwicklungsumgebung: PlatformIO/VSCode

Protokoll-Entwicklung

1. Grundlegende Überlegungen

Das Ziel von LHRP ist der effiziente Datentransport mit minimalem Metadaten-Overhead bei gleichzeitiger Wahrung von Sicherheit, Dynamik und Skalierbarkeit.

In LHRP sind alle Nodes grundsätzlich gleichgestellt (keine festen Rollen). Da klassische Netz-Topologien spezifische Nachteile für dieses Szenario aufweisen, wurde ein hybrider Ansatz gewählt:

Vergleich der Netzwerk-Topologien

Topologie	Nachteile im LHRP-Kontext
Vollvermascht	Hoher Speicherbedarf und ineffizientes Routing, da jede Node die gesamte Struktur kennen muss.
Bus	Zu unflexibel und schlecht skalierbar bei dynamischen Änderungen.
Stern	Zentraler Flaschenhals und ein kritischer Single Point of Failure.

Die LHRP-Lösung

LHRP nutzt eine **baumartige bzw. hierarchische Topologie**, die Folgendes kombiniert:

- **Strikt hierarchische Node-Adressen** (logische Baumstruktur).
- **Frei vernetzte physische Verbindungen** (z. B. Shortcut-Links).

Wesentliche Regel: Jede Node muss mindestens eine Verbindung zu ihrer Parent-Adresse besitzen (Ausnahme: Root-Node). Dadurch entsteht eine logische Hierarchie, ohne die physische Vernetzung einzuschränken.

2. Match-Index-Berechnung (Routing-Entscheidung)

LHRP fungiert als Forward-Protokoll. Beim Eintreffen eines Pakets wird der beste **Next Hop** anhand folgender Logik ermittelt:

1. **Zielprüfung:** Ist das Paket an die eigene Node adressiert?
 - **Ja:** Übergabe an die Applikation (kein Forwarding).
 - **Nein:** Vergleich aller verbundenen Nodes.
2. **Bewertung:** Die Entscheidung basiert auf dem **Match Index**.

3. Der Match Index

Der Match Index bewertet die Übereinstimmung einer potenziellen Next-Hop-Adresse mit der Zieladresse des Pakets. Er setzt sich aus zwei Komponenten zusammen:

- **Positive:** Anzahl der identischen Bytes ab Beginn der Adresse (Adressen sind in Layers gegliedert, die jeweils in bytes representiert werden Bsp: vector{2, 255, 4, 5}. Layer 3 = 4) (Prefix-Match).
- **Negative:** Anzahl der verbleibenden Bytes der Next-Hop-Adresse (negative = nextHopAddress.length - positive).

Formel

$$\text{MatchIndex} = \text{positive} - \text{negative}$$

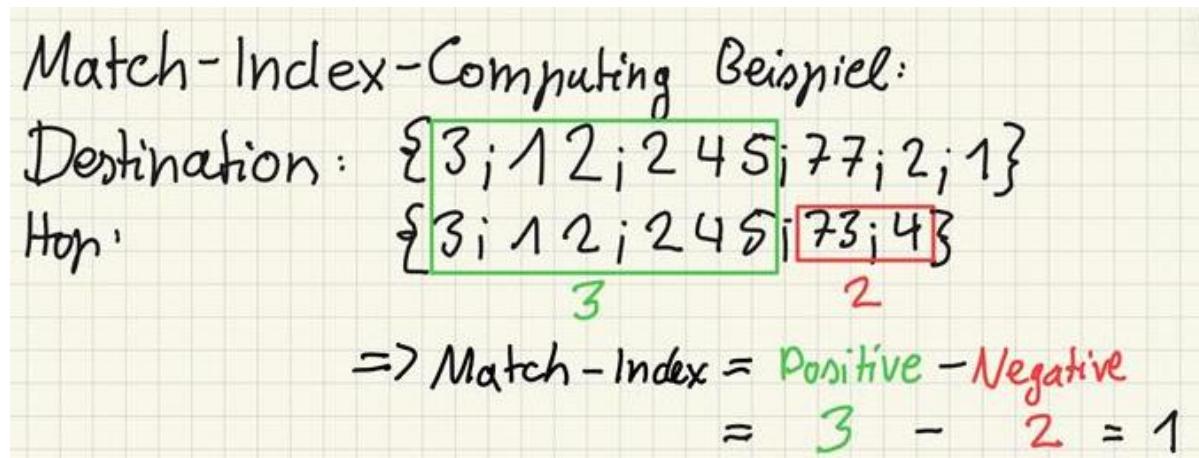
Interpretation

- **Maximaler Match Index:** Vollständiger Match mit der Zieladresse (positive = Zieladresse.length).
- **Minimaler Match Index:** Kein gemeinsames Prefix (- nextHopAddress.length).

Routing-Verhalten:

- **Negative Indizes:** Bevorzugen Parent-Nodes (Aufstieg im Baum).
- **Positive Indizes:** Bevorzugen Child-Nodes (Abstieg im Baum).
- **Shortcut-Verbindungen:** Werden durch den Index automatisch gewichtet und bevorzugt, wenn sie den Weg verkürzen.

Beispiel



4. Auswahl des besten Next Hops

Die Auswahl erfolgt nach strikten Prioritäten:

1. **Höchster Index:** Der Next Hop mit dem höchsten Match Index gewinnt.
2. **Gleichstand (Tie-Break):** Bei identischem Index gewinnt die Verbindung mit der längeren Adresse verwendet (entlastet Root-Nodes, fördert tieferes Routing).
3. **Lokaler Stopp:** Hat die eigene Node den besten Match Index, erfolgt die lokale Zustellung.
4. **Hierarchie-Schutz:** Wenn das Ziel ein direktes Child der eigenen Node ist, darf das Paket nicht an einen Parent oder Neighbor gesendet werden.

5. Codebeispiel (C++)

Datenstrukturen und Match-Logik

```
struct Match {  
    uint16_t positive;  
    uint16_t negative;  
};  
  
inline Match match(const Address &connection, const Address &pocket) {  
    size_t minLen = min(connection.size(), pocket.size());  
    Match m{0, 0};  
  
    while (m.positive < minLen && connection[m.positive] == pocket[m.positive]) {  
        m.positive++;  
    }  
  
    m.negative = connection.size() - m.positive;  
    return m;  
}  
  
inline int matchIndex(const Match &m) {  
    return (int)m.positive - (int)m.negative;  
}
```

Adressvergleiche

```
inline bool eq(const Address &a1, const Address &a2) {  
    return a1.size() == a2.size() &&  
           equal(a1.begin(), a1.end(), a2.begin());  
}  
  
inline bool isChildren(const Address &other, const Address &you) {  
    if (other.size() <= you.size())  
        return false;  
  
    for (size_t i = 0; i < you.size(); i++)  
        if (other[i] != you[i])  
            return false;  
  
    return true;  
}
```

Routing-Entscheidung

```
uint8_t send(const Pocket &p) {
    if (connections.empty()) return 0;
    if (eq(you, p.address)) return 0;

    Connection best = connections[0];
    int bestIdx = matchIndex(match(best.address, p.address));
    size_t bestLen = best.address.size();

    bool directChild = isChildren(p.address, you);

    for (size_t i = 1; i < connections.size(); i++) {
        int idx = matchIndex(match(connections[i].address, p.address));
        size_t len = connections[i].address.size();

        if (idx > bestIdx || (idx == bestIdx && len > bestLen)) {
            best = connections[i];
            bestIdx = idx;
            bestLen = len;
        }
    }

    // Sicherheitsprüfung für direkte Childs
    if (directChild && !isChildren(best.address, you))
        return 0;

    return best.pin;
}

1. Die Formatierung stammt nicht von ChatGPT, sondern aus einer Markdown-Datei und wurde per Screenshot übernommen.
2. Der ganze Code vom Projekt besser dargestellt:  

https://github.com/ManuelWestermeier/LHRP/blob/main/src/LHRP/protocol.hpp
)
```

3 Ergebnisse

3.1 Ergebnisse

Im Rahmen dieser Arbeit ist es gelungen, zwischen mehreren ESP32- und anderen Microcontroller-Nodes ein **stabiles und zuverlässiges Netzwerk** aufzubauen. Der Datenaustausch zwischen den Nodes funktionierte konsistent und fehlerfrei.

Die entwickelte Kommunikation wurde als **eigenständige Library** implementiert und erfolgreich in **kleinen Netzwerken** getestet. Ein typisches Testszenario bestand darin, dass eine Node Pakete mit Zufallswerten an mehrere andere Nodes sendete, welche diese Daten zur Anpassung der LED-Helligkeit verwendeten.

Aufgrund der hohen Stabilität des Systems wurde die Kommunikation anschließend **vollständig kryptografisch abgesichert**. Dabei kamen **AES-GCM mit Initialisierungsvektor (IV)**, GCM-Tag **Authentifizierung** sowie ein **Replay-Schutz** zum Einsatz. Auch unter diesen Bedingungen blieb die Funktionalität des Netzwerks uneingeschränkt erhalten. Dieses Protokoll funktioniert auch sehr gut für Ring, Bus, Stern, vermaschte (Mesh, eingeschränkt), normale Bäume, Linien und vollvermaschte Topologien. Jedoch können esp32 nur begrenzt - max. 20 ESP-NOW Verbindungen - gleichzeitig stabil halten.

Frames und Erklärung

Die Klasse `Pocket` beinhaltet die Quell-Adresse, die Ziel-Adresse, die Daten und die Replayschutz-Sequenznummer. Die "errored" Flag, existiert, um den Nutzer auf Fehler bei der Übertragung hinzuweisen, wird jedoch nicht übertragen. Die Daten in `Pocket` werden im verschlüsselt und im Feld "rawData" des `RawPockets` gespeichert.

```
struct Pocket
{
    Address destAddress;
    Address srcAddress;
    vector<uint8_t> payload;
    bool errored;
    uint32_t seq; // neu: Sequenznummer (32-bit), wird beim Deserialisieren gesetzt
};
```

`RawPacket` wird tatsächlich übertragen. Es darf nicht größer als 250 bytes sein, da es in den ESP-NOW Payload passen muss. Es beinhaltet die Netzwerk-Id, die Längen von Dest -Adresse und Source-Adresse, die Datenlänge, den AES-Initialisierungsvektor und den AES-GCM-Tag. Am Ende wird das Verschlüsselte "Packet" in `rawData` eingefügt, und `RawPacket` versendet.

```
struct __attribute__((packed)) RawPacket
{
    uint8_t netId;                                // 1
    uint8_t lengths;                             // 1 (destLen << 4 | srcLen)
    uint8_t dataLen;                            // 1 (authenticated!)
    uint8_t iv[12];                               // 12
    uint8_t tag[16];                             // 16
    uint8_t rawData[RAWPACKET_SIZE - 1 - 1 - 1 - 12 - 16]; // 219
};
```

3.2 Ergebnis Diskussionen

Die im Rahmen dieser Arbeit erzielten Ergebnisse zeigen, dass das entwickelte **Lightweight Hierarchical Routing Protocol (LHRP)** die gesteckten Ziele hinsichtlich **Stabilität, Effizienz und Sicherheit** erfüllt. Das Protokoll ermöglichte einen zuverlässigen Datenaustausch zwischen mehreren ESP32- und Microcontroller-Nodes, selbst in dynamischen und verteilten Netzstrukturen. Dabei ist zu beachten, dass das Konzept auch ohne Paket-Counter oder Paket-ID ein zyklusfreies Routing zuverlässig ermöglicht. Dies wird durch den Match-Index ermöglicht. Zyklen können nur entstehen, wenn das Paket an eine andere Node mit Match-Index \leq Eigenem-Match index gesendet wird. Dieses Szenario wird jedoch durch Fail-Checks im Code blockiert, und die Pakete werden sobald der Fehler auftritt nicht mehr weitergeleitet (dies passiert nur, wenn die Hierarchische Struktur nicht berücksichtigt würde).

Besonders hervorzuheben ist die Kombination aus **hierarchischer Adressierung** und **frei gestaltbarer physischer Vernetzung**. Dieser Ansatz vereint die Vorteile klassischer Baumtopologien mit der Flexibilität von Mesh-ähnlichen Verbindungen, ohne deren typische Nachteile wie hohen Speicherbedarf oder komplexe Routingtabellen zu übernehmen.

Die Match-Index-basierte Routing-Entscheidung erwies sich als effizient und robust. Durch die Bewertung von Prefix-Übereinstimmung und Adresstiefe konnte in den meisten Fällen ein optimaler Next Hop bestimmt werden, ohne globale Netzwerkinformationen zu benötigen. Shortcut-Verbindungen wurden automatisch bevorzugt, sofern sie den Routingpfad verkürzten, was die Latenz weiter reduzierte.

Auch im Hinblick auf die Sicherheit zeigte sich das System als praxistauglich. Die Implementierung von **AES-Verschlüsselung mit Initialisierungsvektor, Authentifizierung sowie Replay-Schutz** führte zu keiner messbaren Beeinträchtigung der Stabilität. Damit eignet sich LHRP auch für sicherheitskritische Embedded-Anwendungen.

Einschränkend ist anzumerken, dass die Tests bislang hauptsächlich in **kleinen bis mittelgroßen Netzwerken** durchgeführt wurden (max. 5 ESP-32s). Aussagen über das Verhalten in sehr großen oder hochdynamischen Netzen können daher nur eingeschränkt getroffen werden. Jedoch sollte sich wie in der Simulation gezeigt theoretisch nichts am Verhalten und der Stabilität des Netzwerkes bei normaler Nutzlast ändern.

4 Schluss

4.1 Ausblick

Aufbauend auf den vorliegenden Ergebnissen ergeben sich mehrere Ansätze für zukünftige Erweiterungen und Verbesserungen des LHRP:

- Skalierungstests in größeren Netzwerken, um das Verhalten bei hoher Knotenzahl und tieferen Hierarchien zu analysieren
- Automatische Reorganisation der Baumstruktur, beispielsweise bei Node-Ausfällen oder mobilen Nodes
- Optimierung des Match Index, etwa durch Gewichtung einzelner Layer oder dynamische Anpassung der Negativkomponente
- Energieoptimierungen, um den Einsatz in batteriebetriebenen oder energieautarken Systemen weiter zu verbessern
- Integration zusätzlicher Transportmechanismen, wie Fragmentierung oder Priorisierung von Paketen für zeitkritische Anwendungen

Darüber hinaus könnte LHRP auf andere Hardware-Plattformen oder Funktechnologien übertragen werden, um die allgemeine Einsetzbarkeit des Protokolls weiter zu erhöhen.

4.2 Fazit

In dieser Arbeit wurde mit dem Lightweight Hierarchical Routing Protocol (LHRP) ein neuartiger Routing-Ansatz für ressourcenbeschränkte Embedded-Systeme entwickelt und erfolgreich implementiert. Die Ergebnisse zeigen, dass LHRP einen effizienten, sicheren und skalierbaren Datentransport ermöglicht, ohne den Speicher- und Rechenaufwand klassischer Routing-Protokolle zu verursachen.

Durch die klare hierarchische Adressierung, den Match-Index-Mechanismus und die konsequente Sicherheitsarchitektur stellt LHRP eine praktikable Lösung für verteilte Microcontroller-Netzwerke dar. Die erfolgreiche Umsetzung als Library sowie die stabilen Testergebnisse bestätigen die Praxistauglichkeit des Konzepts.

Abschließend lässt sich festhalten, dass LHRP eine solide Grundlage für zukünftige Weiterentwicklungen bietet und insbesondere für IoT- und Embedded-Anwendungen mit hohen Anforderungen an Effizienz und Sicherheit geeignet ist.

4.3 Quellen- und Literaturverzeichnis

Source-Code:

<https://github.com/ManuelWestermeier/LHRP>

<https://github.com/ManuelWestermeier/LHRP/blob/main/src/LHRP/protocol.hpp>

LHRP Tree Simulator — Self-contained (mit Zwischenverbindungen):

V2: <https://manuelwestermeier.github.io/LHRP/> (besser)

V1: <https://manuelwestermeier.github.io/tree-networking-protocol/network-sim> (alt)

4.4 Umgang mit KI

Teile der sprachlichen Formulierung dieses Textes wurden mithilfe eines KI-Modells (ChatGPT) überarbeitet. Die inhaltliche Konzeption, der Quellcode, die Experimente und die Ergebnisse stammen vom mir.