Simulation of a

Spanning Tree Protocol

STG-TINF24B

Merlin Bürkle Mat-Nr: 5316281, Tom Schneider Mat-Nr: 9061251

Inhaltsverzeichnis

Ziel und Zweck des Projekts

Das Ziel dieses Projekts ist die Simulation des Spanning Tree Protocols (STP) in einem Netzwerk.

Das STP wird in Computernetzwerken – insbesondere bei Switches – eingesetzt, um Schleifen (Loops) zu verhindern, die durch redundante Verbindungen entstehen können.

Mit Hilfe dieses Python-Programms lässt sich nachvollziehen, wie ein Netzwerk auf Basis von Root-IDs und Pfadkosten einen spannenden Baum (Spanning Tree) bildet. Dadurch bleibt das Netz vollständig verbunden, aber ohne Kreisläufe.

Das Programm dient also der Veranschaulichung der Grundidee des STP und ermöglicht es, die Entscheidungslogik der Knoten in einzelnen Schritten zu beobachten.

Da es sich um eine Simulation handelt, wird das reale Verhalten des STP bewusst vereinfacht dargestellt. In echten Netzwerken läuft der Austausch der Informationen nicht streng in Runden ab, sondern asynchron und kontinuierlich: Jeder Switch reagiert sofort, wenn er eine Nachricht empfängt, und sendet bei Bedarf neue Informationen weiter. In dieser Simulation hingegen wird der Ablauf in klar getrennte Phasen unterteilt – alle Knoten senden zunächst ihre Nachrichten, empfangen anschließend alle Antworten und aktualisieren danach ihren Zustand. Diese Vereinfachung wurde gewählt, um den Prozess besser nachvollziehbar und reproduzierbar zu machen.

Funktionsweise

Grundidee:

Jeder Knoten (Node) im Netzwerk kennt zunächst nur sich selbst als Root, tauscht regelmäßig Informationen über Root-IDs und Pfadkosten mit seinen Nachbarn aus, passt seine eigene Sicht auf das Netzwerk an, bis alle Knoten dieselbe Root kennen und stabile Pfade existieren.

Das Programm simuliert diesen Ablauf vollständig, ohne echte Netzwerkhardware.

Aufbau des Programms

Das Projekt besteht aus einer Python-Datei, z. B. stp_simulation.py, und einer Topologie-Datei, z. B. input.txt.

Klasse Node

Repräsentiert einen Knoten im Netzwerk (z. B. einen Switch oder Router).

Wichtige Attribute:

- *id*: eindeutige numerische ID
- *name*: Name (z. B. „A“)
- *root_id*: aktuell bekannte Root-ID
- *cost*: bisherige Kosten zum Root-Knoten
- *next_hop*: Nachbar, über den der beste Pfad verläuft
- *neighbors*: Liste direkter Nachbarn
- *received_messages*: Speicher für empfangene Nachrichten

Wichtige Methoden:

- *add_neighbor*(): fügt einen Nachbarn hinzu
- *send_message*(): sendet Root-Informationen an alle Nachbarn
- *receive_message*(): empfängt Nachrichten

- *update_state*(): verarbeitet eingegangene Nachrichten und passt Zustand an

Funktion read_topology_from_file(filename)

Liest eine Netzwerktopologie aus einer Datei im definierten Format ein (siehe Abschnitt 6).

Erzeugt die entsprechenden Node-Objekte und verknüpft sie über ihre Nachbarn.

Funktion stp_protocol(all_nodes)

Führt die eigentliche Simulation des Spanning Tree Protocols aus:

Jeder Knoten sendet Nachrichten.

Jeder Knoten aktualisiert seinen Zustand.

Der Prozess wiederholt sich, bis Konvergenz erreicht ist (d. h. keine Änderungen mehr auftreten).

Funktion print_out(all_nodes)

Gibt nach der Simulation den aktuellen Routing-Zustand jedes Knotens aus.

Programmablauf

Das Programm liest zunächst die Netzwerktopologie aus der Datei input.txt ein und erstellt auf dieser Grundlage alle Knoten sowie deren jeweilige Nachbarn.

Zu Beginn der Simulation betrachtet sich jeder Knoten selbst als Root, da er noch keine Informationen über andere Knoten besitzt. Anschließend tauschen die Knoten untereinander ihre aktuellen Root-Informationen aus und passen ihren Zustand entsprechend an.

Durch diesen wiederholten Austausch in mehreren Runden verbreiten sich die Root-Informationen im gesamten Netzwerk, bis schließlich ein stabiler Zustand erreicht ist.

Am Ende der Simulation ist ein vollständiger Spanning Tree entstanden, der alle Knoten miteinander verbindet und keine Schleifen enthält. Die endgültigen Ergebnisse, also welcher Knoten über welchen Nachbarn mit der Root verbunden ist, werden abschließend in der Konsole ausgegeben.

Format der Eingabedatei (input.txt)

Die Datei beschreibt Knoten und Kanten des Netzwerks.

Beispiel:

```
Graph {
   A = 1;
   B = 2;
   C = 3;
   D = 4;

   A-B: 4;
   B-C: 2;
   C-D: 3;
   A-D: 5;
}
```

Bedeutung:

A = 1; → Knoten A hat die ID 1

A-B: 4; → Verbindung zwischen A und B mit Gewicht (Kosten) 4

Ausführen des Programms

Voraussetzungen:

Python 3.8 oder höher

Datei input.txt im gleichen Verzeichnis wie das Skript

Schritt-für-Schritt:

Öffne ein Terminal oder eine Eingabeaufforderung.

Navigiere in das Projektverzeichnis:

cd pfad/zum/projekt

Führe das Skript aus:

python stp_simulation.py

Beobachte die Ausgabe in der Konsole:

A->Root

B->A

C->B

D->C

Beispielergebnis

Eingabe:

Graph {

A = 1;

B = 2;

C = 3;

A-B: 4;

B-C: 2;

A-C: 5;

}

Ausgabe:

A->Root

B->A

C->B

Interpretation:

A ist Root (kleinste ID).

B erreicht A direkt.

C erreicht A über B mit geringeren Gesamtkosten als direkt.

Fazit

Dieses Projekt zeigt, wie das Spanning Tree Protocol in einem Netzwerk funktioniert – auf einfache, nachvollziehbare Weise ohne physische Netzwerkgeräte.

Es bietet eine wertvolle Grundlage für das Verständnis von Routing-Protokollen, Pfadfindung und verteilten Netzwerkalgorithmen.