Dokumentation: SpanningTree Simulation

1 Projektbeschreibung

Dieses Dokument enthält die Dokumentation, sowie Beispiele der Ausgabe des Programms, das für das Labor in der Vorlesung Kommunikations- und Netztechnik von Josia Scheerle (Matr.-Nr.: 9068864) entwickelt wurde.
Das Projekt wurde in der Programmiersprache Java geschrieben. Dafür wurde als Entwicklungsumgebung IntelliJ mit der OpenJDK23.0.1 verwendet. In dem Ordner `Code` liegt die gesamte Projektstruktur. Die einzelnen Java-Dateien liegen in `src`, sowie die Textdatei zur Eingabe des voliegenden Netzwerks, das simuliert werden soll. Die Ausgabe des Ergebnisses erfolgt in der Konsole, in der das Programm aufgerufen wurde.
Eine bereits kompilierte Version des Codes befindet sich im Ordner `out`
Diese Anwendung simuliert den Ablauf eines Spanning Tree Protocols (STP). Ziel ist es, aus einem gegebenen Netzwerk (bestehend aus Knoten und Verbindungen) einen minimalen Spannbaum zu erstellen, der sicherstellt, dass das Netzwerk ohne Zyklen bleibt.
Die Implementierung umfasst:

- **Knoten** (Nodes), die mit einer ID als Priorität des Knotens im Spanning Tree und einem Namen eindeutig identifiziert werden. Priorität und ID werden im folgenden synonymisch verwendet.
- **Verbindungen** (Edges), die die Kosten für die Kommunikation zwischen zwei Knoten angeben.
- **BPDU** (Bridge Protocol Data Unit), das Datenpaket, welches Informationen zur Optimierung des Spannbaums weiterleitet.

Die Simulation liest eine Eingabedatei ein, erstellt das Netzwerk und simuliert die Kommunikation zwischen den einzelnen Knoten, um den Graphen zu berechnen. Anschließend wird der fertige SpanningTree in der Konsole ausgegeben

2 Hauptklassen und ihre Funktionen

2.1 BPDU

Repräsentiert als Objekt der Klasse die Daten, die zwischen Knoten übertragen werden.

- Attribute: `source`, `rootID`, `totalCost`
- Methoden: `addCost(Integer costAdditive)`, `getSource()`, `getRootID()`, `getTotalCost()`

2.2 Node

Repräsentiert einen Knoten im Netzwerk.

- Attribute: `name`, `id`, `edges` (verbundene Kanten), `root` (zeigt an, ob der Knoten der Root-Knoten ist)
- Methoden:
  - `sendBroadcast()`: Sendet BPDU-Nachrichten an verbundene Knoten.
  - `receiveRequest(BPDU req)`: Empfängt BPDU-Nachrichten.
  - `processOffers()`: Verarbeitet empfangene BPDU-Nachrichten und aktualisiert den aktuellen Zustand des Knotens.
  - `printNextHop()`: Gibt den nächsten Hop aus.

2.3 Edge

Repräsentiert eine Verbindung zwischen zwei Knoten.

- Attribute: `nodes` (Liste der verbundenen Knoten), `cost`, `link`
- Methoden:
  - `transferRequest(BPDU bpdu)`: Überträgt BPDU-Nachrichten an den anderen Knoten.
  - `getElementByName(String name, List<Edge> list)`: Findet eine Verbindung basierend auf ihrem Namen.

2.4 Input

Liest die Netzwerkdaten aus einer Eingabedatei und erstellt Knoten und Verbindungen.

- Methoden:
  - `readInputFile()`: Liest die Datei und erstellt das Netzwerk.
  - `getSpanningTree()`: Gibt die Liste der Knoten zurück.

2.5 Main

Startet die Simulation und gibt das Ergebnis aus.

1. Liest die Eingabedatei ein.
2. Initialisiert und simuliert den Spanning Tree.
3. Gibt das Ergebnis aus.

3 Beispiele

3.1 Spanning Tree mit 7 Knoten

```
Graph LargeNetwork {
A=1
B=2
C=3
D=4
E=5
F=6
G=7
A-B:3
A-C:2
B-D:4
C-D:1
C-E:5
D-F:2
E-G:7
F-G:3
}
```

Ergebnis:

```
Spanning-Tree of LargeNetwork {
    Root: A
    B-A
    C-A
    D-C
    F-D
    E-C
    G-F
}
```

3.2 Spanning Tree mit 3 Knoten

```
Graph SimpleNetwork {
X=10
Y=20
Z=30
X-Y:10
Y-Z:15
X-Z:5
}
```

Ergebnis:

```
Spanning-Tree of SimpleNetwork {
    Root: X
    Y-X
    Z-X
}
```

4 Anweisungen zum Ausfüuhren

1. Erstellen Sie eine Eingabedatei `input.txt` mit dem gewünschten Netzwerk.
2. Kompilieren Sie das Projekt: `javac Main.java`.
3. Führen Sie die Simulation aus: `java Main`.

Das Programm gibt den berechneten Spannbaum in der Konsole aus.