technische universität
dortmund

Projekt 2

Simon Buschmann, Yannik Buchner - Least Loaded Link First (LLLF)
Nikita Podibko, Jan Draeger - Avarage Link Utilization/Random Load Aware
Johannes Heinrich, Malek Haoues Rhaiem -

TU Dortmund University - Fakultät für Informatik

Inhaltsverzeichnis

Genereller Ablauf

Was gab es für Probleme?

Probleme:

Lösung:

Was gab es für Probleme?

Probleme:

- throughput.json wird nicht erstellt

Lösung:

Was gab es für Probleme?

Probleme:

- throughput.json wird nicht erstellt

Lösung:

- *.topo.sh excecutable machen

Was gab es für Probleme?

Probleme:

- throughput.json wird nicht erstellt
- flow.txt wird nicht erstellt

Lösung:

- *.topo.sh excecutable machen

Was gab es für Probleme?

Probleme:

- throughput.json wird nicht erstellt
- flow.txt wird nicht erstellt

Lösung:

- *.topo.sh excecutable machen
- als root das Progam ausführen

Was gab es für Probleme?

Probleme:

- throughput.json wird nicht erstellt
- flow.txt wird nicht erstellt
- Wie implementieren wir unseren Algorithmus?

Lösung:

- *.topo.sh excecutable machen
- als root das Progam ausführen

Was gab es für Probleme?

Probleme:

- throughput.json wird nicht erstellt
- flow.txt wird nicht erstellt
- Wie implementieren wir unseren Algorithmus?

Lösung:

- *.topo.sh excecutable machen
- als root das Progam ausführen
- nutze Nanonet um topologie aus dem Projekt 1 von *.json zu *.topo.py zu *.topo.sh konvertieren

Was gab es für Probleme?

Probleme:

- throughput.json wird nicht erstellt
- flow.txt wird nicht erstellt
- Wie implementieren wir unseren Algorithmus?
- Auch hier werden *.topo.py und *.topo.sh nicht erstellt

Lösung:

- *.topo.sh excecutable machen
- als root das Progam ausführen
- nutze Nanonet um topologie aus dem Projekt 1 von *.json zu *.topo.py zu *.topo.sh konvertieren

Was gab es für Probleme?

Probleme:

- throughput.json wird nicht erstellt
- flow.txt wird nicht erstellt
- Wie implementieren wir unseren Algorithmus?
- Auch hier werden *.topo.py und *.topo.sh nicht erstellt

Lösung:

- *.topo.sh excecutable machen
- als root das Progam ausführen
- nutze Nanonet um topologie aus dem Projekt 1 von *.json zu *.topo.py zu *.topo.sh konvertieren
- als root das Progam ausführen

Was gab es für Probleme?

Probleme:

Lösung:

Was gab es für Probleme?

Probleme:

- nuttcp not in Server/Client Mode error

Lösung:

Was gab es für Probleme?

Probleme:

- nuttcp not in Server/Client Mode error

Lösung:

- Zu geringe demands entfernen

Was gab es für Probleme?

Probleme:

- nuttcp not in Server/Client Mode error
- Wartezeit pro Test zu lange

Lösung:

- Zu geringe demands entfernen

Was gab es für Probleme?

Probleme:

- nuttcp not in Server/Client Mode error
- Wartezeit pro Test zu lange

Lösung:

- Zu geringe demands entfernen
- Abfrage, ob tests durchgelaufen sind

Least Loaded Link First

Topologie

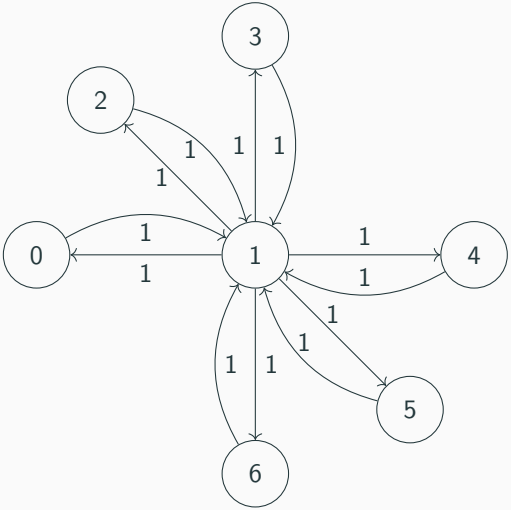


Figure 1: Basnet

Testweise haben wir diese Topologie implementiert.

Demands:

Topologie

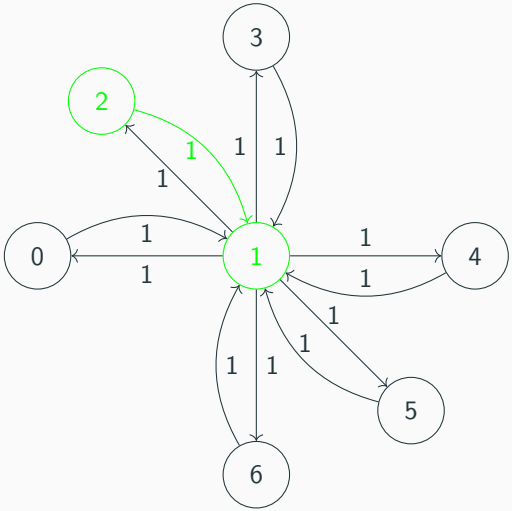


Figure 1: Basnet

Testweise haben wir diese Topologie implementiert.

Demands:

- $s = 2,\ t = 1,\ d = 1$

Topologie

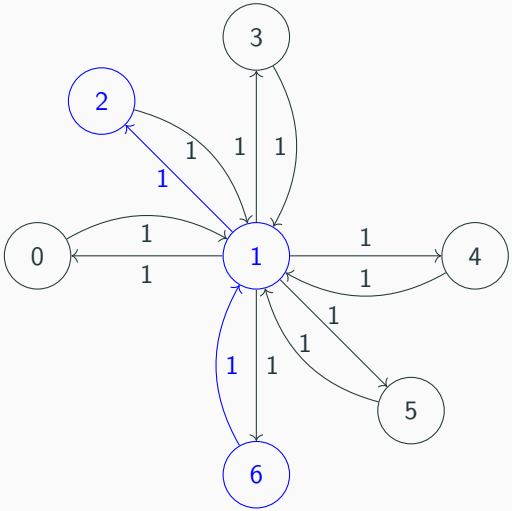


Figure 1: Basnet

Testweise haben wir diese Topologie implementiert.

Demands:

- $s = 2,\ t = 1,\ d = 1$
- $s = 6,\ t = 2,\ d = 1$

Topologie

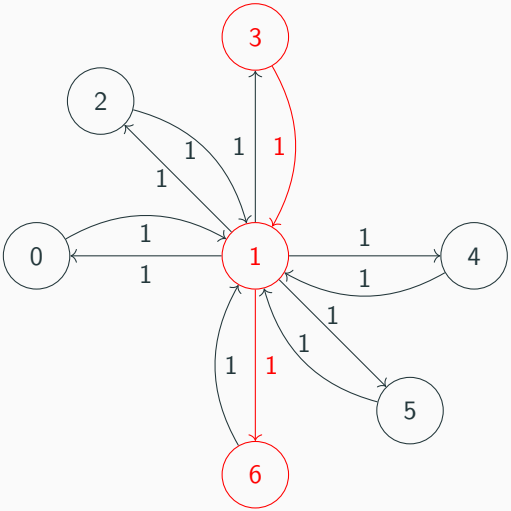


Figure 1: Basnet

Testweise haben wir diese Topologie implementiert.

Demands:

- $s = 2,\ t = 1,\ d = 1$
- $s = 6,\ t = 2,\ d = 1$
- $s = 3,\ t = 6,\ d = 1$

Topologie

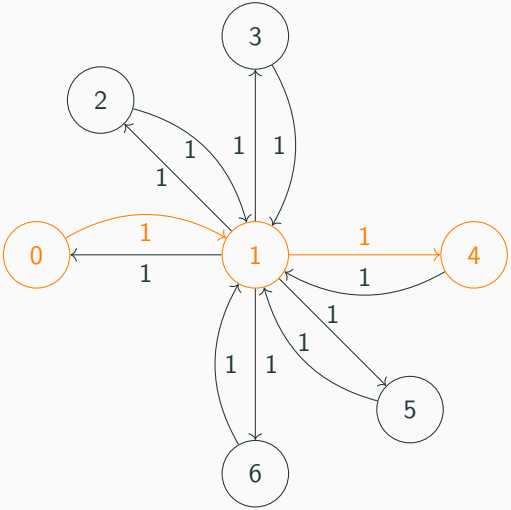


Figure 1: Basnet

Testweise haben wir diese Topologie implementiert.

Demands:

- $s = 2,\ t = 1,\ d = 1$
- $s = 6,\ t = 2,\ d = 1$
- $s = 3,\ t = 6,\ d = 1$
- $s = 0,\ t = 4,\ d = 1$

Topologie

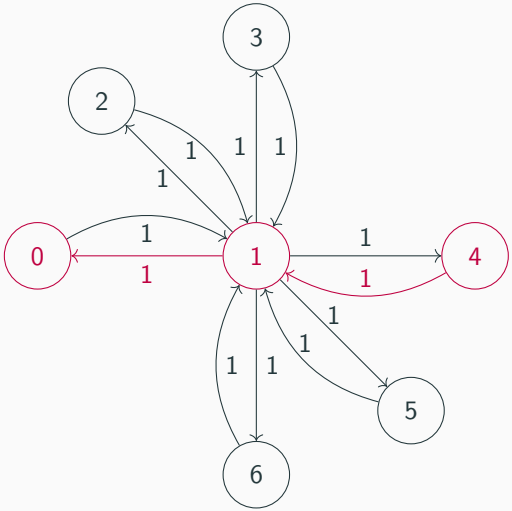


Figure 1: Basnet

Testweise haben wir diese Topologie implementiert.

Demands:

- $s = 2,\ t = 1,\ d = 1$
- $s = 6,\ t = 2,\ d = 1$
- $s = 3,\ t = 6,\ d = 1$
- $s = 0,\ t = 4,\ d = 1$
- $s = 4,\ t = 0,\ d = 1$

Topologie

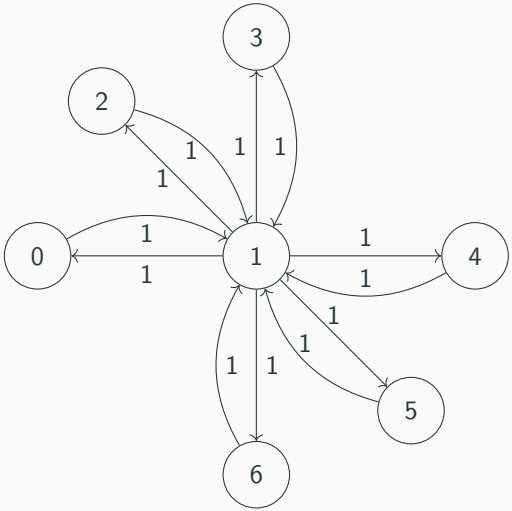


Figure 1: Basnet

Testweise haben wir diese Topologie implementiert.

Demands:

- $s = 2,\ t = 1,\ d = 1$
- $s = 6,\ t = 2,\ d = 1$
- $s = 3,\ t = 6,\ d = 1$
- $s = 0,\ t = 4,\ d = 1$
- $s = 4,\ t = 0,\ d = 1$

MLU: 1.0

Results

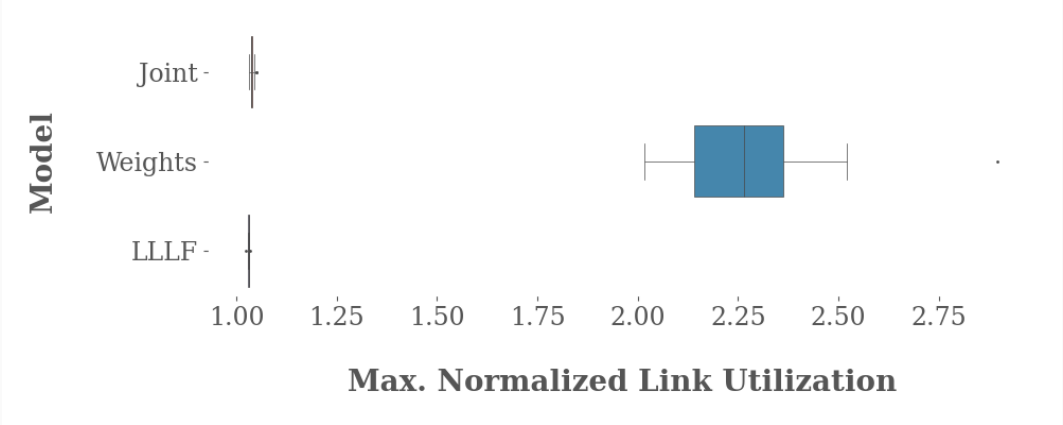


Figure 2: Results

Randomized Load Aware

RandomizedLoadAware

Topologie-Eigenschaft	**Warum schlechter für `RandomizedLoadAware`?**
Geringe Pfadvielfalt	Kein Auswahlspielraum für Pfadverteilung.
Zentrale Engpässe	Kann nicht umleiten, wenn es keine Alternativen gibt.
Ungenaue oder unvollständige Kapazitäten	Bewertungsfunktion wird verzerrt.
Zu hohe Netzgröße	Pfaderzeugung / Laufzeit steigt exponentiell.

Topologie

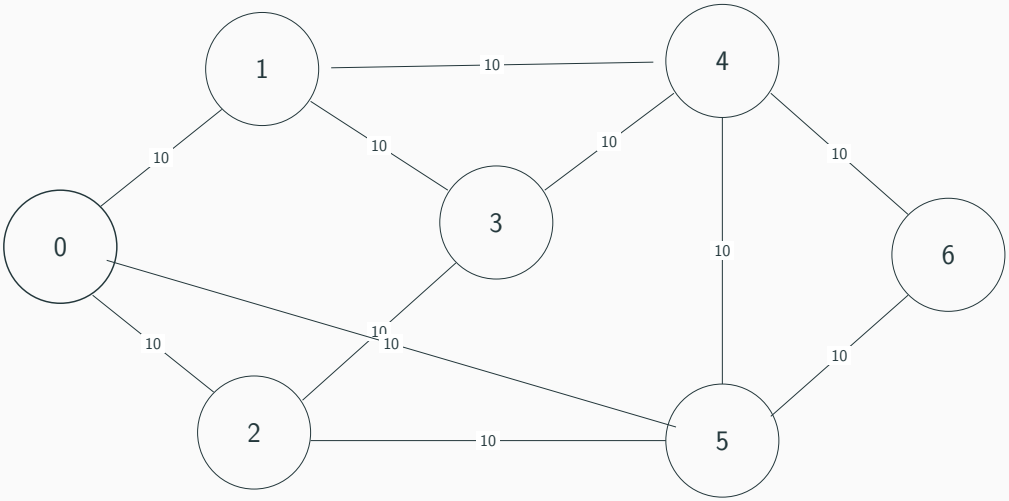
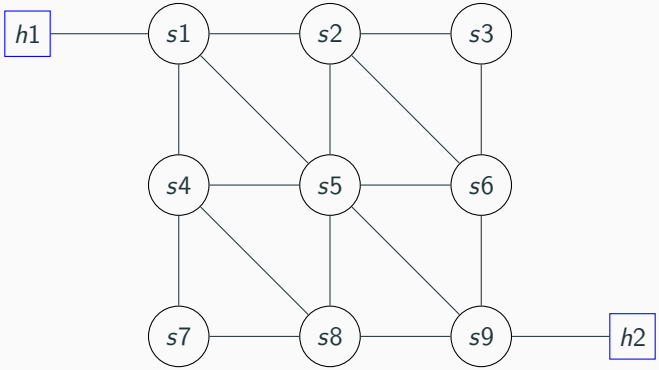


Figure 3: Beispiel Topologie

Average Path Length

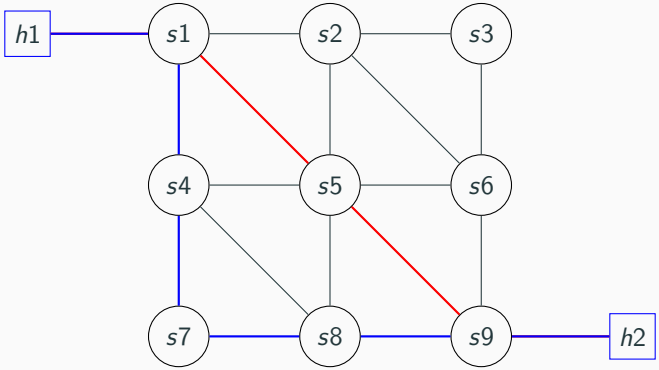
Topologie



Vorteile der Topologie

- **Pfad Diversität**: Durch viele Pfadmöglichkeiten
- **Tradeoff Visualisierung**: Visualisiert gut tradeoff zwischen Pfadlänge (Diagonale Wege) und MLU (Alternative Wege)
- **Waypoint Nutzwert**: Auch durch Pfad Diversität

Beispiel



Fragen?

Noch Fragen?