SPP Übung 2 Quang Dui Nguyen, Egemen Ulutürk, Leonard Bongard

1)

A)

1) Diameter = 9, kürzester kritischer Pfad von 0-4 -> 21

2) Bandbreite 2,3 Gb/s Teilung bei 14 cut 10, 11, 12, 13, 15

3) Network Degree= 7, Knoten 5, 14 sind die Verursacher

4) Node connectivity von 1, Knoten 11, 24, 23, 22, 20

B)

Schwäche 1: Der Diameter ist zu hoch. Dadurch verlangsamt sich der Verkehr und es bedarf einer menge Sprünge um bspw. von dem Anfang zum Ende des kritischen Pfades zu kommen.

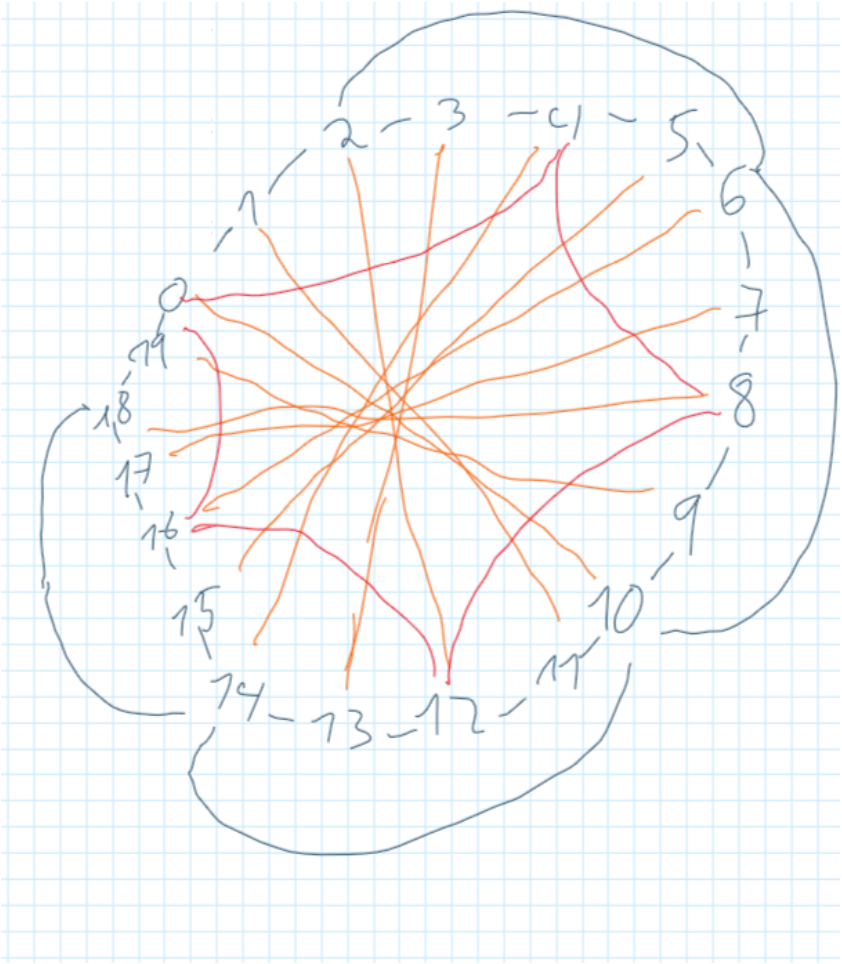
Schwäche 2: Die geringe Bandbreite ist eine Schwäche für das Netzwerk. Darunter leidet die Geschwindigkeit im Netzwerk und es kann so zu Verstopfungen bei kleineren Kanten kommen.

Schwäche 3: Die geringe **Node-Connectivity** sorgt dafür, dass, im ungünstigsten Fall (in diesem Netzwerk: 11, 20, 22, 23, 24, 21), beim Ausfall eines Knotens auch andere Knoten vom Netzwerk getrennt werden.

C) FETT = Bunt; DÜNN = Blau

Diameter = 4; von 18 zu 2: (18)-14-4-3-2

Bandbreite= 11,1 Gb/s



2)

A) kohärenter Speicher: Writes to the same location are serialized, that is, two writes to the same location by any two processors are seen in the same order by all processors.

B) Sequentiel konsistenter Speicher: Every thread sees the effects of every operation in the same order

C) Nichts: Die folgende Bedingung ist verletzt also nicht kohär und die Befehle sind nicht in der richtigen Reihenfolge, alo auch nicht seq. ; Writes to the same location are serialized, that is, two writes to the same location by any two processors are seen in the same order by all processors.

D)kohärenter Speicher: Gleiche Begründung wie A

E) Sequentiel konsistenter Speicher: Alle Befhel sind in der richtigen Reihenfolge.

3)

A)

- Thread ID
- Program counter
- Register set

- Stack

B)

Threads teilen sich den gleichen Speicher und damit die gleichen Ressourcen. Damit können Threads viel schneller write und read Befehler ausführen.

Verschiedene Prozesse haben nicht den gleichen Speicher. Also müssen sie um anderen Prozessen die Speicheradresse mitzuteilen, über ein Network interagieren.

C)

Die Parallelisierung mit Threads macht Sinn, wenn häufiger mit “stalls” zu rechnen ist. Mit genug nebenläufigen Threads kann man so die Aufgaben verteilen und so die Stalls umgehen. Dadurch steigt die Geschwindigkeit zum Bearbeiten deutlich.

Außerdem erlauben uns Threads die die gleichen Speicherstellen benutzen eine Aufgabe unter den Threads aufzuteilen.

Prozesse, die verschiedene Speicherstellen haben, erlauben uns die gleiche Aufgabe oder den gleichen Algorithmus gleichzeitig auf deren Speicherstelle auszuführen.