Abteilung Algorithmik **Sommer '10**
Institut für Betriebssysteme und Rechnerverbund
TU Braunschweig

Dr. Alexander Kröller
Christiane Schmidt

Verteilte Algorithmen
Übung 5 vom 23. 6. 2010

Abgabe der Lösungen am Montag, den 5. 7. 2010, entweder vor der Übung im IZ251, oder bis 16:40 im Hausaufgabenrückgabeschrank.

Aufgabe 1: Gegeben sei ein Netzwerk, in dem jeder Knoten einen Messwerte ermittelt hat. Ein Knoten $b \in V$ dient als Basisstation. Die Basisstation b möchte die k höchsten Messwerte ermitteln.
Wir gehen vom CONGEST-Modell aus, und nehmen an, dass ein Messwert in $\mathcal{O}(\log n)$ Bits codiert werden kann.
Verwende dafür den folgenden Algorithmus, der in Anlehnung an den Algorithmus aus Aufgabe 4.1 formuliert ist und ebenfalls die Idee des Pipeline-MST verwendet. $C_b = E_b \backslash S_b$

a) Solange v Kinder hat, von denen noch keine Nachricht empfangen hat: sammle Messwerte von Kindern, speichere in E_b

b) Solange $C_b \neq \emptyset$ und weniger als k Kanten gesendet

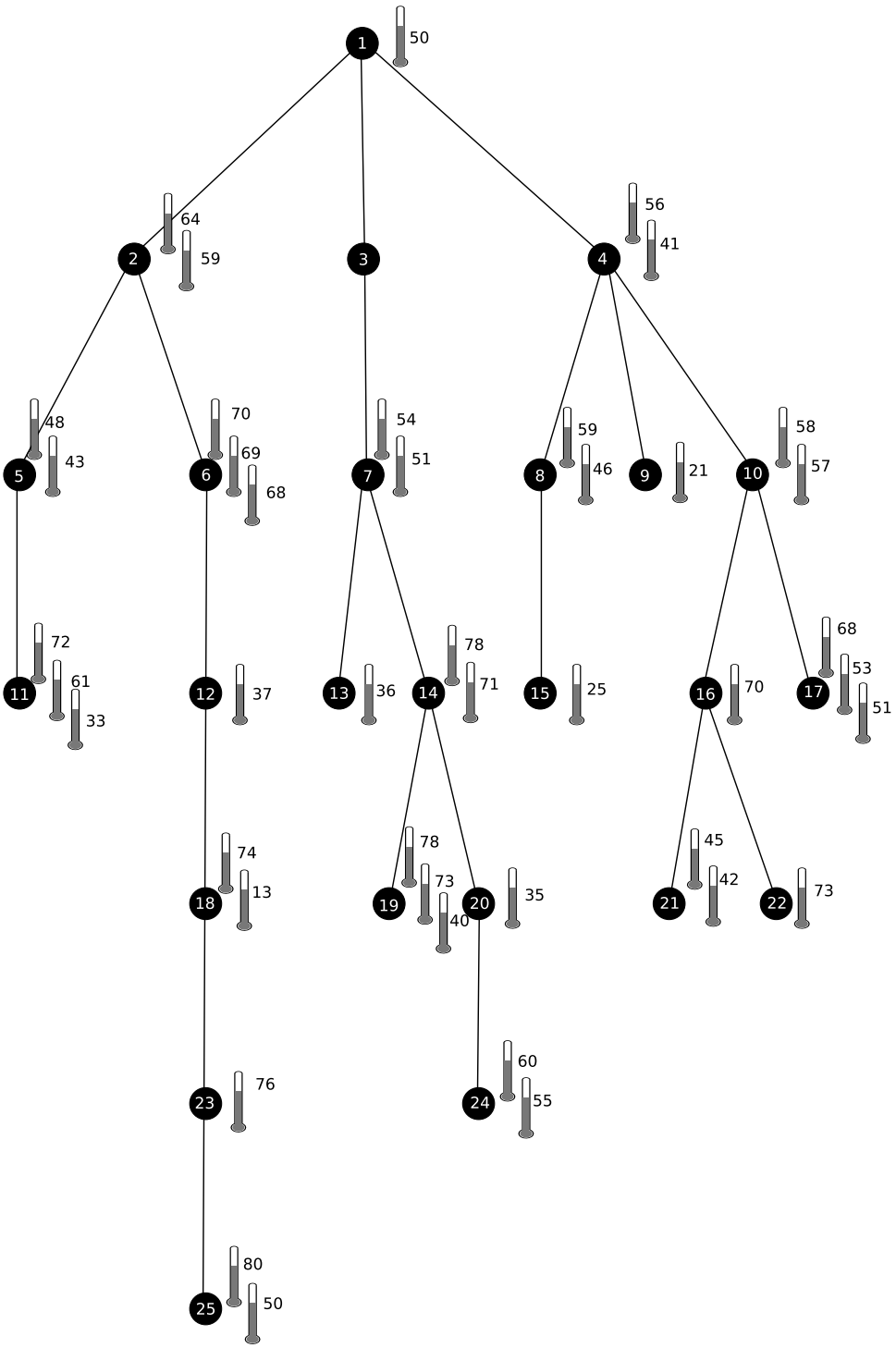
   - schicke den größten Messwert $m \in C_b$ an den Vater (falls ex.), füge w zu S_b hinzu
   - empfange Nachrichten von Kindern, füge Messwerte zu E_b hinzu

c) Schicke Ende an Vater, falls ex.

Gib für das Beispiel auf Seite 2 mit $b = 1$ und $k = 6$ in jeder Iteration an, welcher Knoten welche Werte wohin schickt. Wieviel Iterationen benötigt man? **(3 P.)**

Aufgabe 2: Angenommen, wir haben zwei Algorithmen $\mathcal{A}_1$ und $\mathcal{A}_2$ gegeben, die jeweils in Zeit θ_1 bzw. θ_2 stabilisieren. Allerdings benötigt $\mathcal{A}_2$ ein korrektes Ergebnis von $\mathcal{A}_1$ als Input (zum Beispiel könnte $\mathcal{A}_1$ einen Baum konstruieren und $\mathcal{A}_2$ darauf einen ConvergeCast durchführen).
Zeige, wie die beiden so ausgeführt werden können, dass ein kombinierter Algorithmus $\mathcal{A}$ entsteht, der in $\theta_1 + \theta_2$ stabilisiert. **(4 P.)**

Aufgabe 3: Betrachte Algorithmus 3.3 aus der VL (Dijkstras Token Ring). Zeige, dass der Algorithmus nicht stabilisiert, wenn v_0 nicht alle n Zustände $0, \ldots, n-1$ verwendet. Konkret: Nimm an, der Algorithmus hätte eine beliebige (!) zu kleine Schätzung $n' < n$. Erzeuge als Gegenspieler einen Fehlerzustand, aus dem der Algorithmus nicht zu einem korrekten Zustand konvergiert.
(Hinweis: Noch einmal ganz konkret :) : Gehe von einem Ring mit beliebig grossem $n > 2$ aus, und einem beliebigen $1 < n' < n$. Du darfst also **weder** für n **noch** für n' einen bestimmten Wert annehmen!) **(3 P.)**


1 50
2 64 59
3
4 56 41
5 48 43
6 70 69 68
7 54 51
8 59 46
9 21
10 58 57
11 72 61 33
12 37
13 36
14 78 71
15 25
16 70
17 68 53 51
18 74 13
19 78 73 40
20 35
21 45 42
22 73
23 76
24 60 55
25 80 50