Zusatznotizen Vorlesung Verteilte Algorithmen vom 08. 11. 2001
Prof. Dr. R. Wattenhofer

Warum interessieren wir uns fuer Hypercubes?

Hat nette Eigenschaften. Wenn wir das Netzwerk selber waehlen koennten... manchmal koennen wir das sogar: Telefon Switches, Parallelrechner, etc.

- Kurze Pfade – kleiner Durchmesser D (fuer effizientes Routen)
- Kleiner Knotengrad g; am liebsten g konstant! (einfachere Hardware)

Grad $\leq g \Rightarrow n = O(g^D)$ Knoten $\Leftrightarrow D = \Omega(\log_g n)$

Was kennen wir fuer spezielle Graphen?

- Lineare Liste, Ring: Durchmesser riesig $\theta(n)$
- Mesh, Torus: Durchmesser $\theta(\sqrt{n})$
- Hypercube: Durchmesser $\theta(\log n)$, aber mehr auch $\theta(\log n)$ Knotengrad...
- Balancierter Binaerbaum? Durchmesser $\theta(\log n)$, Knotengrad 3! Aber der Binaerbaum hat eine Wurzel, durch die die Haelfte des Verkehrs muss...

Wir brauchen noch mehr Eigenschaften:

- Uniformitaet, Symmetrie (keine speziellen Knoten)... Denkuebung: formale Definition fuer Symmetrie/Uniformitaet?
- Redundanz, Fehlertoleranz
- Einfaches Addressieren und Routen
- Hohe „Parallelitaet“
- Simples Upgrading (nicht immer alle Knoten erweitern wie beim Hypercube!)

Butterfly

Dimension d, d+1 Level, 2^d Knoten auf jedem Level. Knoten ist definiert durch d-bit langen Bitstring plus Level k mit k = 0, ..., d.

Knoten auf Level k (k = 1, ..., d) mit Bitstring b_d, ... ,b_2, b_1 hat eine Verbindung mit:

- dem Knoten auf Level k-1 und dem gleichen Bitstring.
- dem Knoten auf Level k-1 und Bitstring b_d, ..., b_{k+1}, **1-b_k**, b_{k-1}, ..., b_1.

[Butterfly mit d=2 oder 3 zeigen]

Butterfly hat alles!! Knotengrad 4, einen logarithmischen Durchmesser, einfaches Routing, Broadcast, etc.

Butterfly vs. Hypercube: Fasse im Butterfly alle Knoten mit dem gleichen Bitstring in einen Superknoten zusammen, dann wird aus dem Butterfly ein Hypercube. Viele Algorithmen koennen deswegen einfach vom Butterfly auf den Hypercube oder umgekehrt uebertragen werden.

80/90er Jahre Vorlesung „Verteilte Algorithmen“ waere wahrscheinlich „Parallelrechnen Vorlesung“ und wuerde viele dieser Strukturen behandeln. [Siehe Liste Rechnerarchitekturen]

Permutation Routing auf Butterfly

(Bisher immer ein Sender und ein Empfaengerknoten. Im „wirklichen Leben“ wollen viele Paare gleichzeitig kommunizieren...)

Definition Permutation Routing: Jeder Knoten will genau eine Nachricht versenden und auch genau eine empfangen. Wir muessen also alle Permutationen im Kopf haben – wie effizient ist das im worst case?

Machen wirs uns einfacher und gucken nur auf die n=2^d Knoten im Level 0 (der Rest der Knoten ist passiv und routet einfach unseren Verkehr). Machen wir’s uns noch einfacher und nehmen mal an, dass die Nachrichten synchron im „Takt“ verschickt werden – zuerst einfach gerade nach rechts zu Level d und dann zurueck auf dem eindeutigen Pfad zur Destination auf Level 0.

Phase 1 stellt keine Problem dar, die Nachrichten stoeren sich nicht. Phase 2 ist schwieriger: Ein Knoten auf Level k hat noch 2^k moegliche Empfaenger auf Level 0, gleichzeitung kann die Nachricht von 2^(d-k) moeglichen Knoten von Level d stammen. D.h., wenn wir Pech haben (und das haben wir im worst case!), dann muessen durch einen Knoten auf Level k min(2^k,2^(d-k)) durchgeroutet werden. Fuer k=d/2 sind das 2^(d/2) Nachrichten. [siehe Beispiel]. Das sind √n Nachrichten durch einen einzigen Knoten. Kann man diesen Riesenstau abwenden?

Randomisierung: Wir haben nicht nur einen moeglichen Pfad von Sender zu Emfaenger sondern 2^d viele, die alle nur eine Laenge von 2d Kanten haben. Idee: Sende die Nachricht zu zufaelligem Knoten auf Level d (auf eindeutigem Pfad), und dann an den Empfaenger auf Level 0 (auch auf dem eindeutigen Pfad). [Dieser Trick ist von Leslie Valiant.]

Theorem: Mit hoher Wahrscheinlichkeit dauert das nur O(d) Schritte, ist also asymptotisch optimal.

Beweis: Nicht unkompliziert...

Beweisidee: Von wie vielen Startknoten geht Nachricht in Phase 1 durch einen speziellen Knoten u auf Level k? Maximal 2^k. Wieviele Zufallsentscheidungen hat bis zu diesem Zeitpunkt jede dieser 2^k Nachrichten gemacht? k. Nur nur wenn alle diese Entscheidungen exakt die „falschen“ waren kommt die Nachricht durch den Knoten u. Das heisst, dass die Wahrscheinlichkeit fuer jede dieser 2^k Nachrichten nur 2^{-k} ist, durch u geroutet zu werden. Alle 2^k Nachrichten haben ihre Entscheidungen unabhaengig getroffen. Die erwartete Anzahl Nachrichten durch Knoten u sind deshalb E[Nachrichten durch u] = 2^k*2^{-k} = 1. Das heisst wir haben keinen Stau! Beim Rueckweg ist es dasselbe, da wir im wesentlichen das invertierte Problem loesen.

Achtung: Eigentlich ist das alles viel schwieriger – ein Erwartungswert von 1 heisst noch lange nicht, dass bei 2^d Knoten auf Level k alle exakt eine Nachricht abkriegen. Auf Level d kriegt der meistbesuchte Knoten sogar O(log n) Nachrichten ab – das ist bekannt als das Balls-into-Bins Problem. [Verweis auf Vorlesung Web Algorithmen]