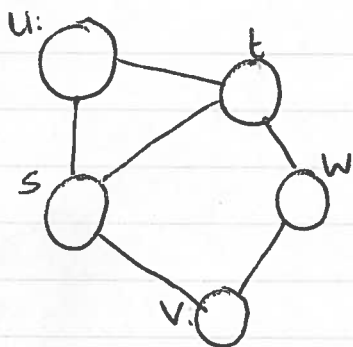


(3/10/12)

Net Change

- ① Doel, ontstaan
- ② Data per node
- ③ Updates
- ④ Node Set, Upcycling
- ⑤ Programmeren, Starten

① Doel: Routingstabel in netwerk.



Wat doet u met een bericht voor v?

Kan in 2 hops : distance via s : preferred neighbor

Node u heeft $Nb_u[v]$ nodig voor versturen, maar de afstandsschattingen zijn nodig om Nb

te kiezen. $D_u[v]$.

Ontstaan: Net Change is in 1977 bedacht door Tajbnapis. Uitgangspunten:

- Locale berekening (van alleen 1 link niet pad)
- Minimum Hop (gewogen kan ook, maar tricky)
- Behoud deelresultaat bij update ("Net change").

Nu in Internet: OSPF, houdt netwerk-status expliciet in elke knoop.

② Data in elke knoop

$d(u,v)$ - Afstand ("echt" dus: onbekend!)

$D_u[v]$ - Schatting in u van $d(u,v)$.

$Nb_u[v]$ - Node u 's Preferred Neighbor voor v .

$ndis_u[w,v]$ - Node u 's kennis over w 's afstand tot v .

$D_u[v]$ kan afwijken van $d(u,v)$

bv tijdens de berekening of vlak na een link-mutatie.

$ndis_u[w,v]$ kan afwijken van $D_w[v]$

als de laatste is gewijzigd, maar de wijziging nog niet is verwerkt door u .

Waarom een lokale kopie $ndis$ van D_w ?

Anders moet u , als hy zijn afstand wil updaten, de D_w opvragen als onderdeel van die berekening. Dat wilde T. niet: "doorgeven" van D_w en berekenen van D_u zijn gescheiden.

Waarom de $ndis$ van alle burenen?

en niet alleen de beste onthouden?

De beste kan wegvallen (failure uw) of met een hogere schatting komen.

Doe als voorbeeld: uitval van link (sv).

Recompute:

if ($v == u$) { $D_u[v] = 0$; $Nb_u[v] = local$ }

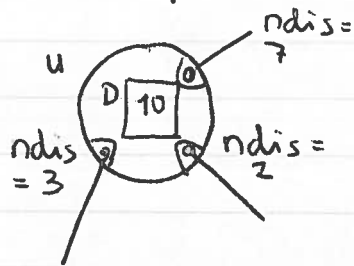
else { $Nb_u[v] = buur w met laagste ndis_u[w,v]$
 $D_u[v] = ndis_u[w,v] + 1$ }

// + iets over updaten!

③ Update Mechanismes

Wat wil je niet?

① D klopt niet met ndis:

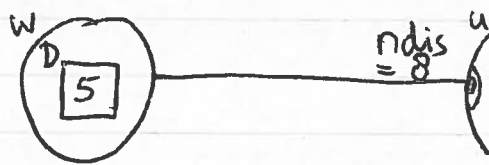


Kleinste ndis is 2 dus
D moet 3 zijn

Voorkom dit:

- Na elke wijziging ndis een Recompute.

② ndis klopt niet met D:



$D_w[v]=5$ dus
 $ndis_u[w,v]$ moet 5 zijn.

Voorkom dit:

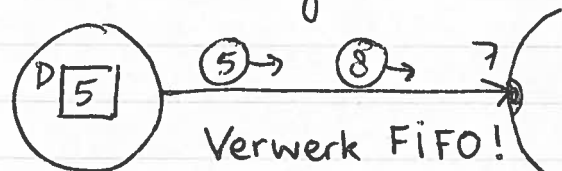
- Na elke wijziging D stuur je $\langle mydist, v, D \rangle$.

mydist: kan lyst zijn (alle v) of los bericht per v .
Beter los dan minder waarden doorgeven.

Updates na elkaar:

- By nieuwe link:

Alle D -waarden sturen!



Interactie tussen $Recompute(v)$ en $snd/rec \langle mydist, -- \rangle$:

ALS er na tijdstip t geen changes meer zijn

DAN ① Stopt de activiteit na een poosje

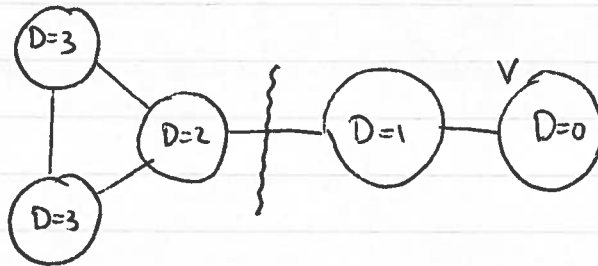
② Daarna kloppen de D/Nb tabellen.

④ Node set en Upcycling-probleem.

Het boek veronderstelt dat u weet welke bestemmingen v er allemaal zijn.

Hier ken je in 't begin alleen je eigen en de buur-nummers. Breid routingstabel uit als je hoort van een bestemming v uit een $\langle \text{mydist} \dots \rangle$ die je nog niet kent.

Bij het wegvallen van een link worden D -waarden stapsgewys naar boven aangepast. Probleem: Partitie!



Er worden eindeloos veel berichten rond gepompt voor één (niet bereikbare!) knoop.

Oplossing Tajibnapis: Je weet N , voor een pad is $N-1$ hops max (sliert). Gebruik N als "oneindig".

Hier: Je weet N niet, maar je kunt het "huidige" aantal "zichtbare" knopen als N gebruiken.
Minder mooi alternatief: gebruik 20 als bovengrens.
(Je moet nl. verstuurde schattingen tellen!)

⑤ Programmeer- en Opstartsuggesties.

Een socket heeft een "aanvrager" (Client) en een "aannemer" (Server) maar is in twee richtingen bruikbaar. Dus: tussen x en y is maar één socket nodig:



Je kunt beslissen, dat altijd de kleinste client is.

Dan:

- by opstarten een listener maken
- na opstarten, verbinding zoeken met nummers groter dan jij in je parameterlijst.
- listener houden voor latere connects.

En:

- Geef elke socket een thread om de info eruit af te handelen.
- Houd de Listener om nieuwe verbindingen aan te kunnen gaan.

Let op: In de opdracht wordt een $C \times$ of $D \times$

aan één kant gegeven. Boek stelt dat beide $\langle \text{repair}, w \rangle$ of $\langle \text{fail}, w \rangle$ krijgen.