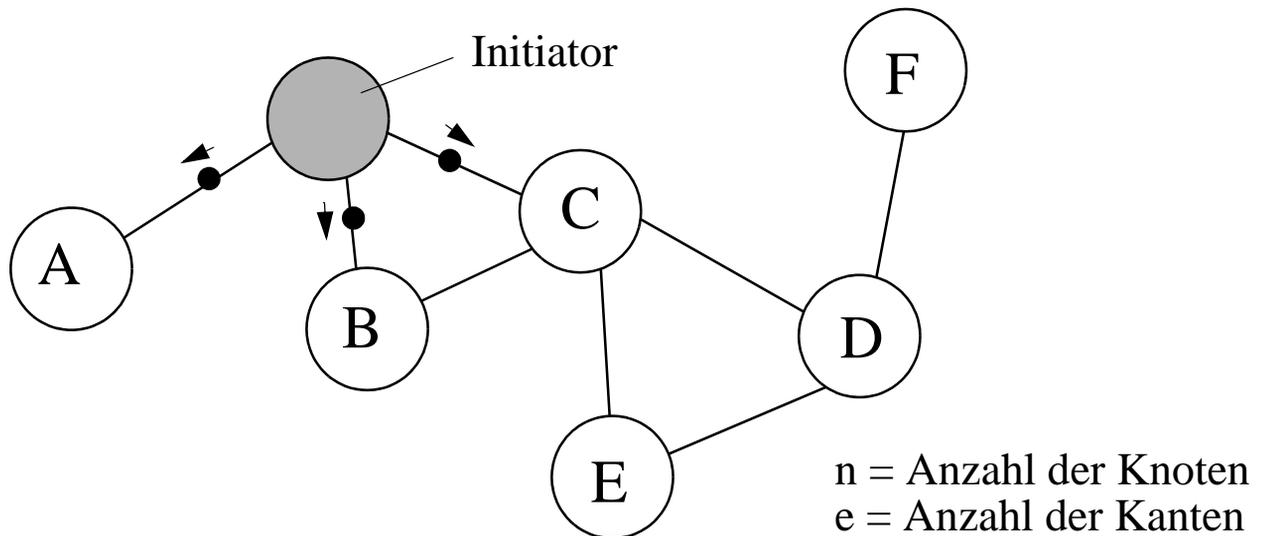


# Informationsverteilung durch ‘Fluten’

“flooding”



- Voraussetzung: zusammenhängende Topologie
- Prinzip: jeder erzählt *neues* Gerücht allen anderen Freunden
- Kein Routing etc. notwendig

---

## - Wieviele Nachrichten werden versendet?

- jeder Knoten sendet über alle seine inzidenten Kanten ( $\rightarrow 2e$ )
- jedoch nicht über seine Aktivierungskante zurück ( $\rightarrow -n$ )
- Ausnahme: Initiator ( $\rightarrow +1$ )

$\implies$  Also:  $2e - n + 1$

## - Frage: Wie *Terminierung* feststellen?

d.h.: wie erfährt der Sender (= Initiator), wann alle erreicht wurden?  
(das ist für “sicheren” oder “synchronen” Broadcast notwendig)

# Flooding-Algorithmus - eine etwas formale Spezifikation

- Zwei *atomare* Aktionen für jeden Prozess:

- wechselseitig ausgeschlossen
- "schlagartig"?
- ununterbrechbar?

Assertion (muss wahr sein,  
damit Aktion ausgeführt wird)

```
R: {Eine Nachricht <info> kommt an}
  if not informed then
    send <info> to all other neighbors;
    informed := true;
  fi
```

Natürlich auch  
"merken" der  
per Nachricht  
erhaltenen  
Information  
<info>

```
I: {not informed}
  send <info> to all neighbors;
  informed := true;
```

- initial sei informed=false
- Aktion R wird nur bei Erhalt einer Nachricht ausgeführt
  - "message driven"
- Aktion I wird vom Initiator *spontan* ausgeführt
  - darf es mehrere *konkurrente* Initiatoren geben?

# Terminierungserkennung von Flooding

1) Jeder Prozess informiert (ggf. indirekt) den Initiator (oder einen Beobachter) per *Kontrollnachricht*, wenn er eine *Basisnachricht* erhält; Initiator zählt bis  $2e-n+1$

- Nachteile?

- n und e müssen dem Initiator bekannt sein
- indirektes Informieren kostet ggf. viele Einzelnachrichten

- Nachrichtenkomplexität?

- Variante: Prozess sendet Kontrollnachricht, wenn er *erstmalig* eine Basisnachricht erhält; Initiator zählt bis n-1

- n muss dem Initiator bekannt sein
- Terminierung in dem Sinne, dass alle informiert sind - es können dann aber noch (an sich nutzlose) Basisnachrichten unterwegs sein

2) *Überlagerung* eines geeigneten Kontrollalgorithmus, der die Berechnung des Flooding-Verfahrens beobachtet und die Terminierung meldet

3) *Bestätigungsnachrichten* (acknowledgements)

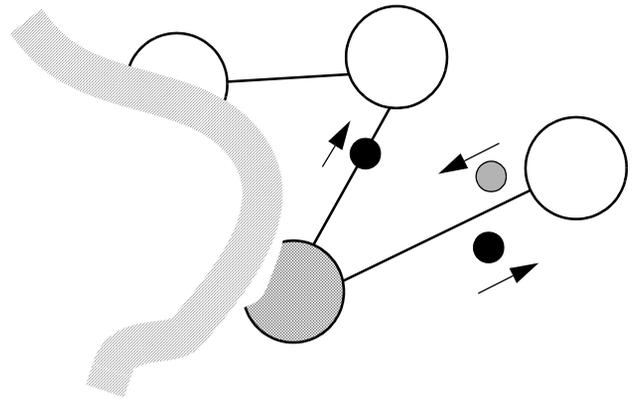
- direktes Bestätigen einer Nachricht funktioniert nicht
- indirekte Bestätigungsnachrichten: ein Knoten sendet erst dann ein ack, wenn er selbst für alle seine Nachrichten acks erhalten hat
- klappt diese Idee? auch wenn der Graph Zyklen enthält? wieso?

# Flooding mit Quittungsmeldungen

(Originalversion des *Echo-Algorithmus* von Chang '82)

*Prinzip:* Ein Prozess versendet eine Quittung für eine empfangene Nachricht erst dann, wenn er für alle von ihm selbst versendeten Nachrichten Quittungen erhalten hat

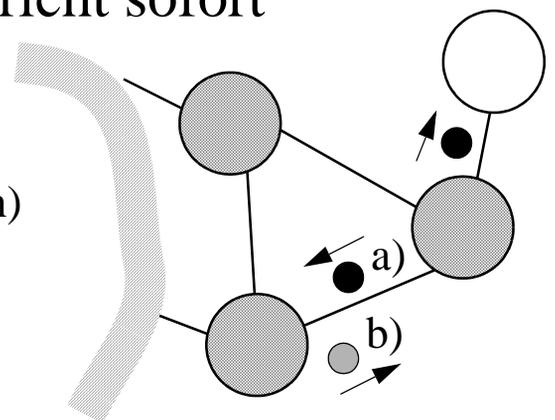
- Ein Knoten mit Grad 1 ("Blatt") sendet sofort eine Quittung zurück



Nachrichte des zugrundeliegenden (Flooding)-Algorithmus

- Ein Knoten, der bereits eine Basisnachricht erhalten hat, quittiert jede weitere Basisnachricht sofort

- *Prinzip:* "bin schon informiert"
- *Wirkung:* Zyklen werden aufgebrochen (als wäre die Kante gar nicht vorhanden)
- *Konsequenz:* es entsteht ein Baum



- Terminiert, wenn Initiator alle Quittungen erhalten hat

- Wieviele Quittungen / Nachrichten insgesamt?

# Der Echo-Algorithmus

(PIF-Variante von A. Segal, 1983)

Propagation of Information with Feedback

- Ähnliches Verfahren 1980 von Dijkstra/Scholten: "Diffusing Computations"

---

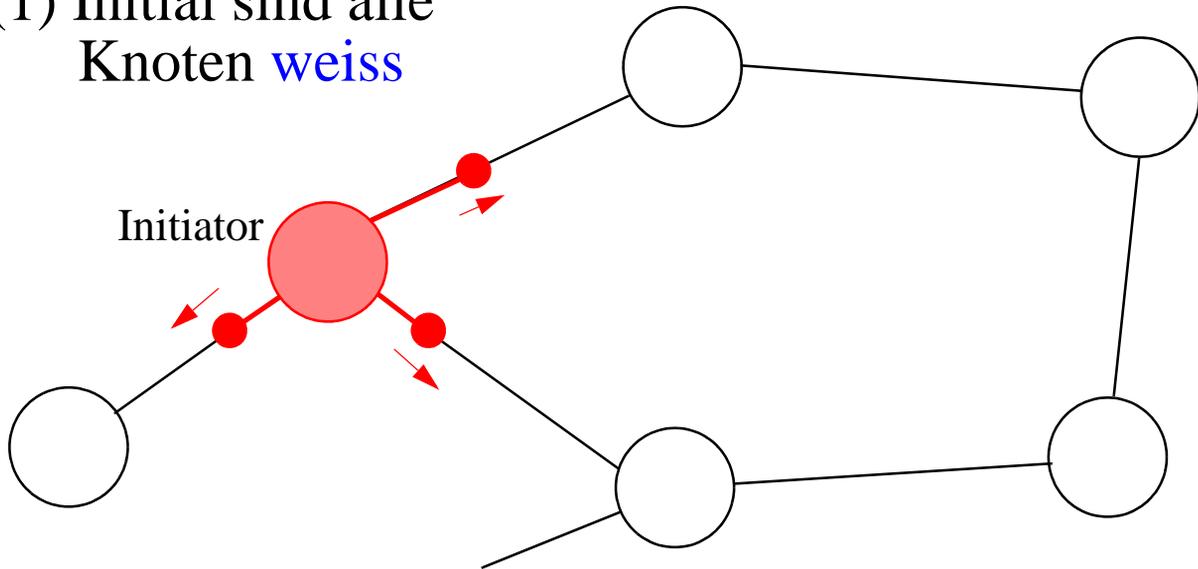
Ausgehend von einem einzigen Initiator:

- Paralleles *Traversieren* eines bel. (zusammenhängenden ungerichteten) Graphen mit 2e Nachrichten
  - Terminierung klar durch "Vollzugsmeldung"
  - Idee: *Indirektes* acknowledge
  - Hinwelle durch "*Explorer*": Verteilen von Information
  - Rückwelle durch "*Echos*": Einsammeln einer verteilten Information
  - Aufbau eines *spannenden Baumes* ("Echo Kanten": jeder Knoten sendet genau ein Echo)
- 

Paralleler *Wellenalgorithmus*:

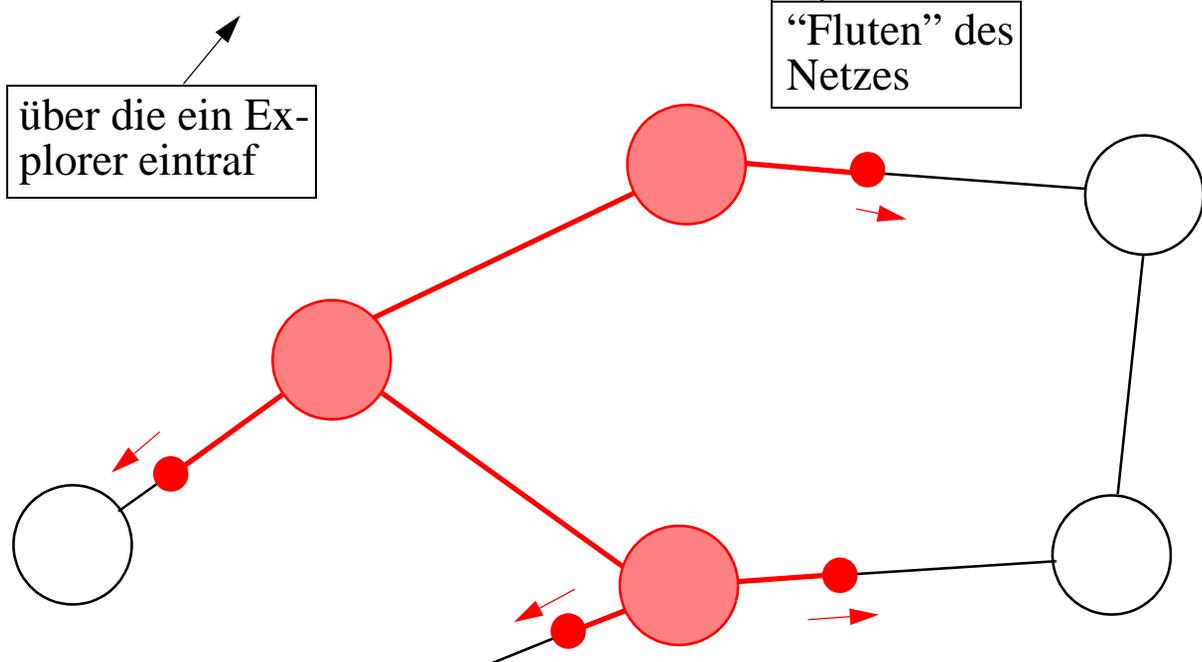
- virtueller broadcast
- *Basisalgorithmus* für andere Verfahren ("underlying algorithm"; "superposition")

(1) Initial sind alle Knoten **weiss**

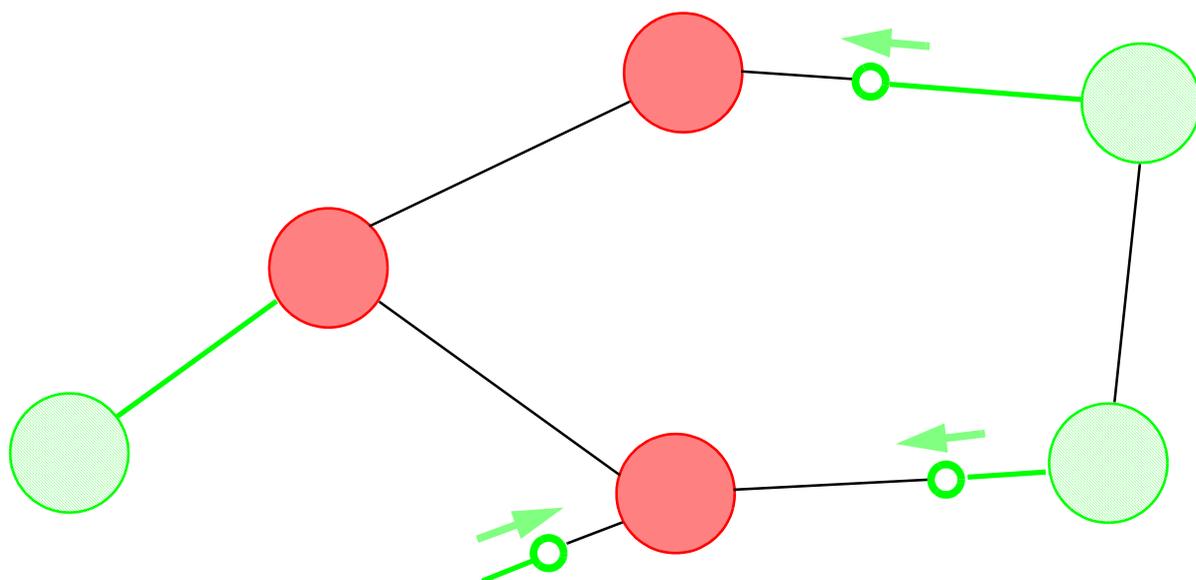
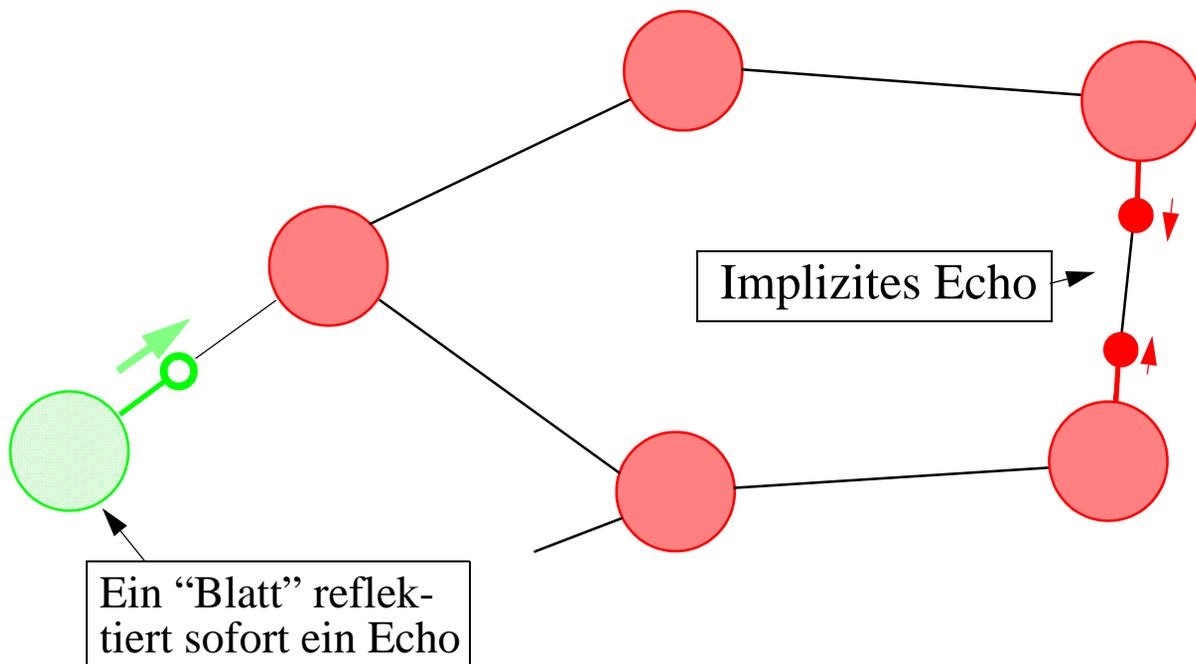


- Der (eindeutige) **Initiator** wird **rot** und sendet (rote) **Explorer** über alle seine Kanten

(2) Ein weisser Knoten, der einen Explorer bekommt, wird rot und sendet Explorer über alle seine anderen Kanten ("flooding") und merkt sich die **"erste" Kante**



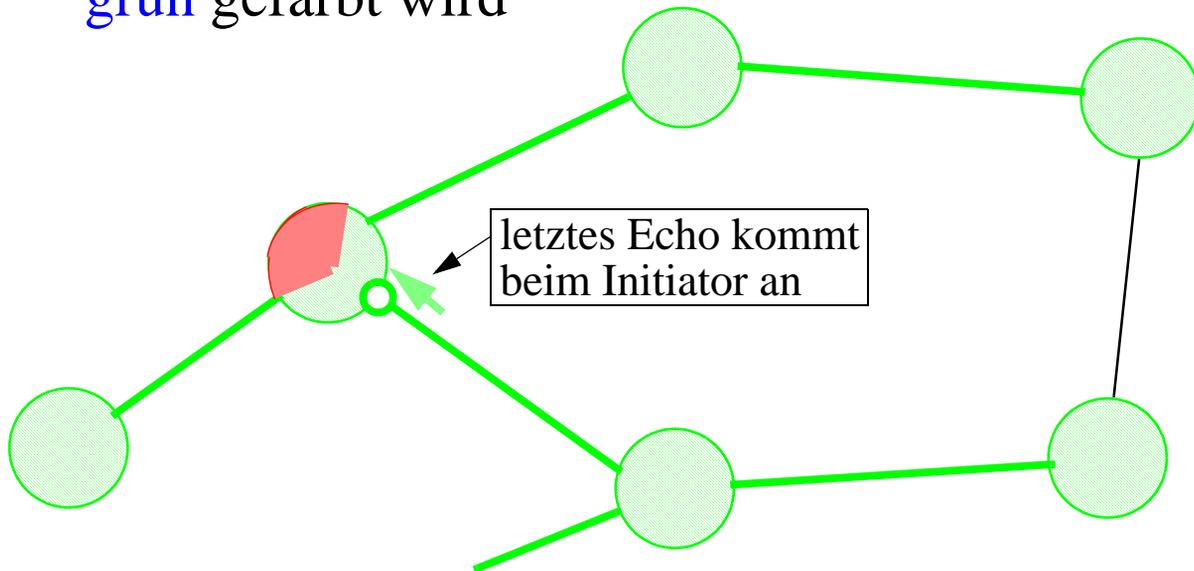
(3) Ein (roter) Knoten, der über alle seine Kanten einen Explorer *oder* ein Echo erhalten hat, wird **grün** und sendet ein (grünes) *Echo* über seine "erste" Kante



Beachte: *Atomare Aktionen*  
--> Explorer können sich höchstens auf Kanten begegnen, nicht aber bei Knoten!

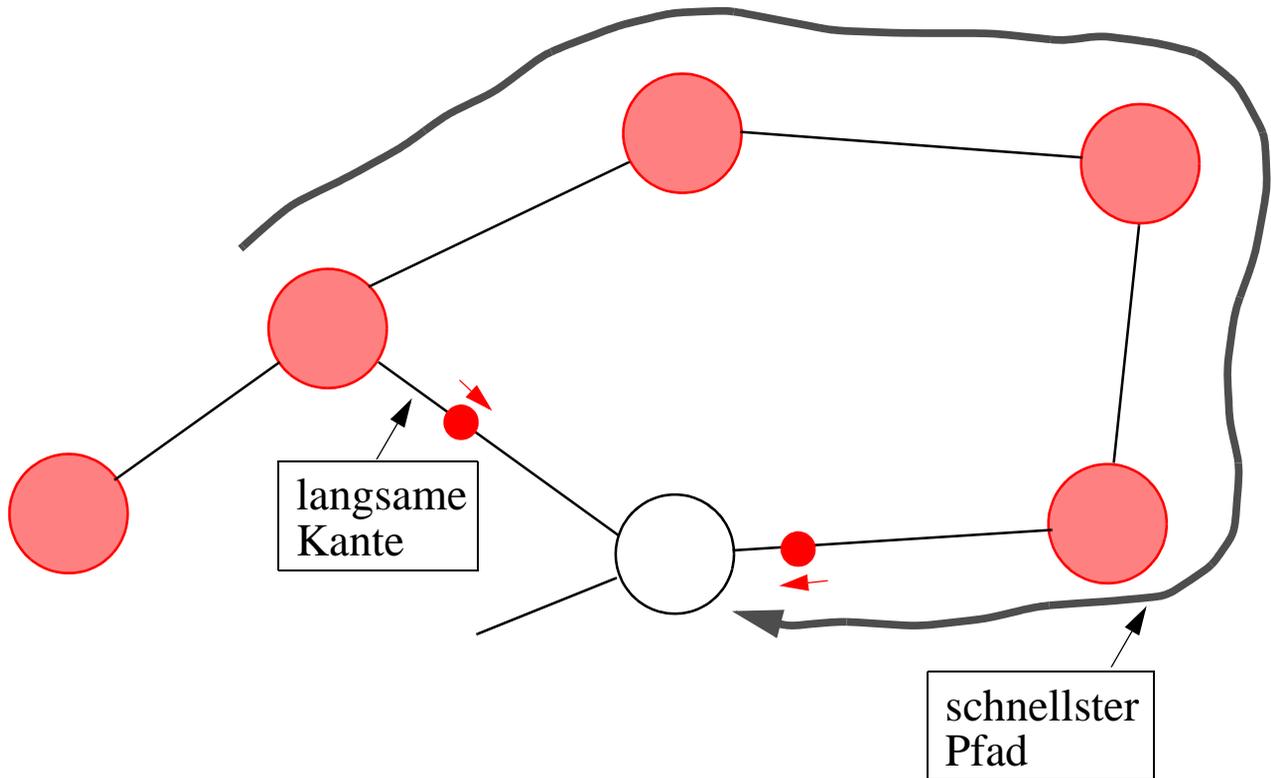
Auf einer Kante, wo sich zwei Explorer begegnen, wird der **Zyklus aufgebrochen**

(4) Das Verfahren ist **beendet**, wenn der **Initiator grün** gefärbt wird

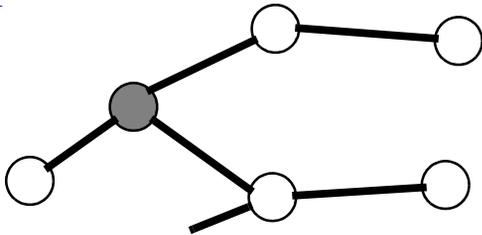


- **Grüne Kanten** bilden einen *Spannbaum*
  - alle Knoten bis auf den Initiator haben eine "erste" Kante
  - "grüner Graph" ist zusammenhängend
- Über jede Kante laufen genau 2 Nachrichten:
  - entweder ein Explorer und ein gegenläufiges Echo, oder zwei Explorer, die sich begegnen --> **Nachrichtenkomplexität =  $2e$**
- Verfahren ist *schnell*: parallel und "**bester**" (?) Baum
- Ereignis "**rot werden**" in jedem Prozess definiert eine Welle (*Hinwelle*)
- Ereignis "**grün werden**" in jedem Prozess --> *Rückwelle*
- Es darf nicht mehr als einen Initiator geben:
  - was geschieht sonst?
  - wie kann man dem Problem mehrerer Initiatoren begegnen?

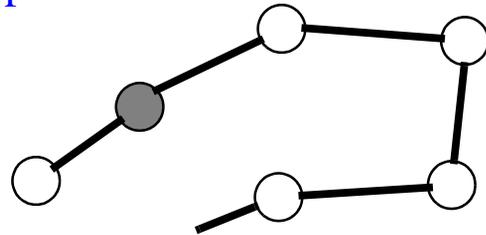
Echo-Algorithmus ist *nicht-deterministisch*, es können (je nach Geschwindigkeit einer "Leitung") *verschiedene Spannbäume* entstehen!



Spannbaum 1



Spannbaum 2



- Inwiefern ist die PIF-Variante besser als die Originalversion?

- Nachrichtenkomplexität
- Einfachheit / Eleganz

# Echo-Algorithmus...

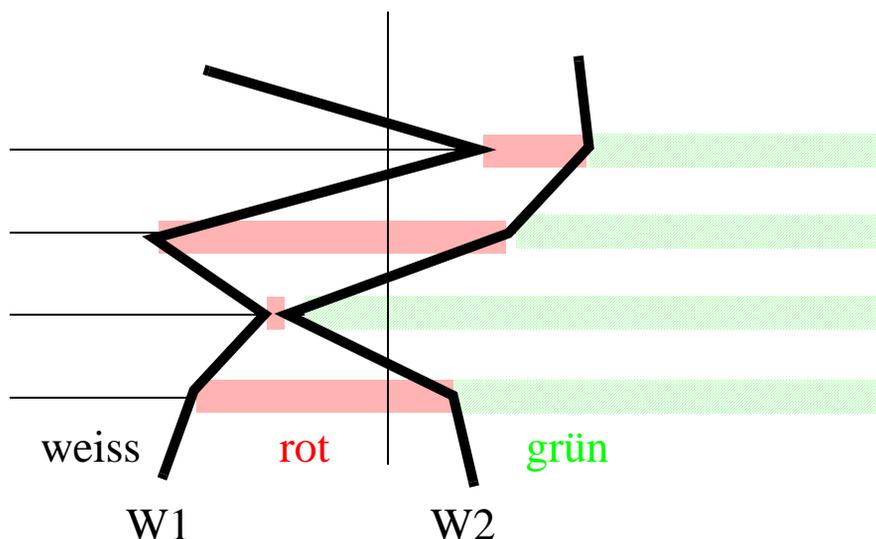
- Jeder Knoten wird *erst rot* und *dann grün*
- Rote Phase ist bei "Blättern" allerdings recht kurz
- Ein *grüner Knoten* hat *keine weisse Nachbarn*

==> Eine von einem grünen Knoten versendete Basisnachricht wird nicht von einem weissen Knoten empfangen

- beachte: gilt nur für *direkte* Nachrichten, nicht für *Nachrichtenketten!*

==> *Weisse und grüne Phase* sind in "gewisser Weise" *disjunkt*

- obwohl es globale Zeitpunkte geben kann, wo ein Knoten bereits grün ist, während ein anderer (nicht direkt benachbarter!) noch weiss ist!



- "Rot werden" und "grün werden" definieren zwei *Wellen*
- mit diesen Wellen kann *Information transportiert* werden (*verteilen* bzw. *akkumulieren*)
- Echo-Algorithmus wird daher oft als *Basis* für andere Verfahren verwendet

