

# Gossip & CYCLON

**Vorlesung: Verteilte Systeme 2013**

# Gliederung

## 1. Gossip

1. Überblick
2. Struktur
3. Anwendungsbeispiele
  1. Datenaustausch
  2. Peerauswahl
  3. Topologiekonstruktion
  4. Ressourcenverwaltung
  5. Berechnungen

## 2. CYCLON

1. Allgemein
2. Algorithmus
3. Eigenschaften
  1. Konnektivität
  2. Konvergenz
  3. Ingradverteilung
  4. Robustheit
  5. Selbstheilung
4. Fazit

# Gossip: Überblick

- probabilistischer Informationsaustausch
- Wiederholung der einzelnen Arbeitsschritte (endlos)
- analog zur Gerüchteverbreitung oder zu Krankheitsepidemien
- historisch zur Sicherung der Konsistenz verteilter Datenbanken

# Gossip: Struktur

## Begriffserklärung Peers

- Prozesse
- haben Cache mit Referenzen zu anderen Peers
- ggf. auch peer-spezifische Informationen im Cache

# Gossip Struktur: Peerauswahl

- verschiedene Auswahlkriterien je nach Anwendung
- Unterschiede bei Auswahl über kabellose oder kabelgebundene Verbindungen
- Simulation eines anderen Verbindungstyps möglich, häufig teuer und unnötig
- kaum Unterschiede auf Applikationsschicht zwischen synchron und asynchron
- asynchron ist kein "richtiges" Gossiping

## Gossip Struktur: Datenaustausch

- Peers entscheiden, welche Daten sie austauschen
- entweder Applikationsdaten oder Referenzen zu anderen Peers werden ausgetauscht

## Gossip Struktur: Datenverarbeitung

- stark anwendungsabhängig

# Gossip Anwendung: Verteilung 1/2

- Nachrichten/Daten in einem Netzwerk (möglichst gleichmäßig) verteilen
- Jeder Knoten hat lokalen Cache, in welchem die Nachrichten/Datensätze abgelegt werden

## Peerauswahl

Zufällige Auswahl einer bestimmten Anzahl von Kommunikationspartnern

## Datenaustausch

Eine Nachricht aus dem lokalen Cache eines Peers in den lokalen Cache eines anderen Peers kopiert

# Gossip Anwendung: Verteilung 2/2

## Datenverarbeitung

Eventuell Weiterleitung neuer Nachrichten an höhere Schichten, Löschung veralteter Nachrichten

- push/pull/hybrid Modus
- Durchschnittliche Verbreitungsgeschwindigkeit:  $O(\log N)$  mit  $N$  = Anzahl der Knoten



# Gossip Anwendung: Partnerfindung 1/2

## Peerauswahl

Austauschpartner werden zufällig aus einem lokalen Cache (Nachbarliste) ausgewählt

## Datenaustausch

Weitergabe der lokalen Liste

## Datenverarbeitung

Empfangene Nachbarlisten werden in lokale Liste eingefügt

## Gossip Anwendung: Partnerfindung 2/2

- Grundlage für viele andere Gossipingsysteme
- Anzahl übernommener Nachrichten ausschlaggebend für Diversität des lokalen Cache
- In dynamischen Netzwerken Mechanismus zur Löschung veralteter(inaktiver) Knoten notwendig
- Annahme eines homogen strukturierten zugrundeliegenden Netzes

# Gossip Anwendung: Topologie Konstruktion

- Manchmal striktere Kontrolle über Overlay-Netzkonstruktion notwendig

## Peerauswahl

Zufällige Auswahl aus dem lokalen Cache

## Datenaustausch

Listen von Peers

## Datenverarbeitung

Einfügen der empfangenen Liste in den lokalen Cache und Bewertung der neuen Peers (evtl. Löschung von Peers)

# Gossip Anwendung: Topologie Konstruktion

- Jeder Peer hat nur partielle Sicht auf das Gesamtsystem
- Einführung einer Bewertungsfunktion für Nachbarn
- Bewertungsfunktionen
  - Anbindungsgeschwindigkeit
  - Verfügbarkeit
  - ID-abhängige Kriterien (z.B. Ringkonstruktion)

# Gossip Anwendung: Ressourcenverwaltung

## Peerauswahl

Zufällige Auswahl aus dem lokalen Cache

## Datenaustausch

Statusinformationen über benachbarte Peers

## Datenverarbeitung

Updaten des lokalen Cache mit neuen  
Statusinformationen

- Fehlererkennung
- Verwerfen von Statusinformationen fehlerhafter Peers

# Gossip Anwendung: Berechnungen

- Aggregationen wie Durchschnittsfindung, Extremwertbestimmung
- Einsatz z.B. in Sensornetzen

## Peerauswahl

Zufällige Auswahl aus dem lokalen Cache

## Datenaustausch

Anwendungsabhängiges Datum wird kopiert

## Datenverarbeitung

Neues Datum wird aus dem empfangenen und dem lokalen Datum berechnet

# Gliederung

## 1. Gossip

1. Überblick
2. Struktur
3. Anwendungsbeispiele
  1. Datenaustausch
  2. Peerauswahl
  3. Topologiekonstruktion
  4. Ressourcenverwaltung
  5. Berechnung

## 2. CYCLON

1. Allgemein
2. Algorithmus
3. Eigenschaften
  1. Konnektivität
  2. Konvergenz
  3. Ingradverteilung
  4. Robustheit
  5. Selbstheilung
4. Fazit

# CYCLON: Allgemein

- Algorithmus zur Peerauswahl
- Zufällig mit gleicher Wahrscheinlichkeit aus gesamten Netzwerk
- Nur lokale Sicht
- Oft als Service für höhere Schichten verwendet



# CYCLON: Enhanced Shuffling 1/2

Für Knoten  $P$

1. Erhöhe Alter um eins für alle Nachbarn
2. Wähle ältesten Nachbar  $Q$  und  $l - 1$  zufällige Nachbarn
3. Ersetze  $Q$ 's Eintrag mit dem Alter 0 und Adresse von  $P$
4. Sende aktualisierte Teilmenge zu  $Q$
5. Empfange eine Teilmenge von  $Q$  mit  $i$  eignen Einträgen
6. Verwerfe Einträge die auf  $P$  zeigen und in  $P$ 's Cache liegen
7. Aktualisiere  $P$ 's Cache und füge alle verbleibenden Einträge hinzu. (erst die leeren Cacheeinträge nutzen, dann ersetze die Einträge, die man zu  $Q$  geschickt hat)

# CYCLON: Enhanced Shuffling 2/2

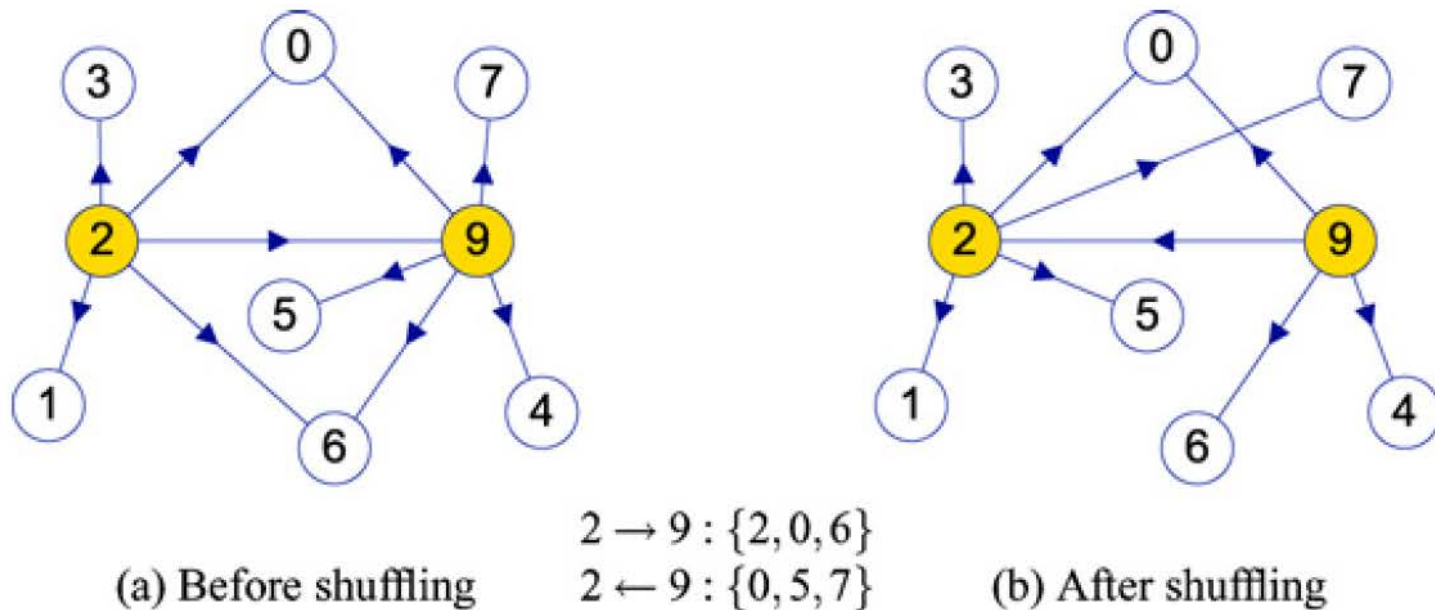
Für Knoten  $Q$ :

1. Wähle  $l$  zufällige Nachbarn, sende an  $P$
2. Verwerfe Einträge die auf  $Q$  zeigen und in  $Q$ 's Cache liegen
3. Aktualisiere  $Q$ 's Cache und füge alle verbleibenden Einträge hinzu. (erst die leeren Cacheeinträge nutzen, dann ersetze die Einträge, die man zu  $P$  geschickt hat)

Parameter

- $c$  Cachegröße
- $l$  Anzahl d. ausgetauschten Nachbarn)
- $\Delta T$  Zeitintervall zwischen den Nachrichten

# CYCLON: Enhanced Shuffling cont'd



**Fig. 1.** An example of shuffling between nodes 2 and 9. Note that, among other changes, the link between 2 and 9 reverses direction.

# CYCLON: Grundlegende Eigenschaften

- Konnektivität
- Konvergenz
- Ingradverteilung
- Robustheit - Selbstheilung

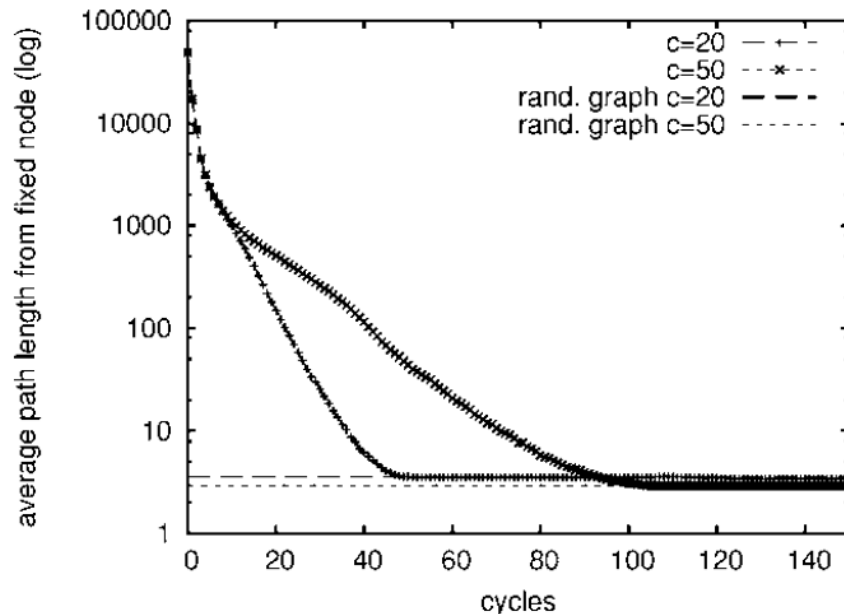
# CYCLON: Konnektivität

- Shuffling stellt sicher:
  - Kein Knoten wird aus dem Netzwerk entfernt

→ Netzwerk wird nicht partitioniert

(Annahme: Zuverlässige Übertragung)

# CYCLON: Konvergenz



- Durchschnittlicher kürzester Weg im Netzwerk
- Enhanced shuffling konvergiert zu Zufallsgraphen

# CYCLON: Ingradverteilung

## Ausgrad

Nachbarn eines Knoten

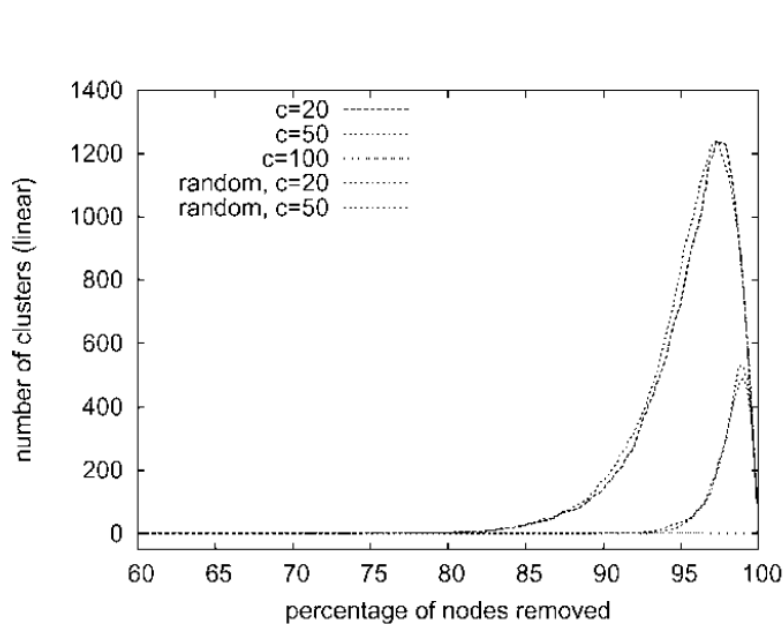
- Ausgrad ist durch Cachegröße gegeben

## Ingrad

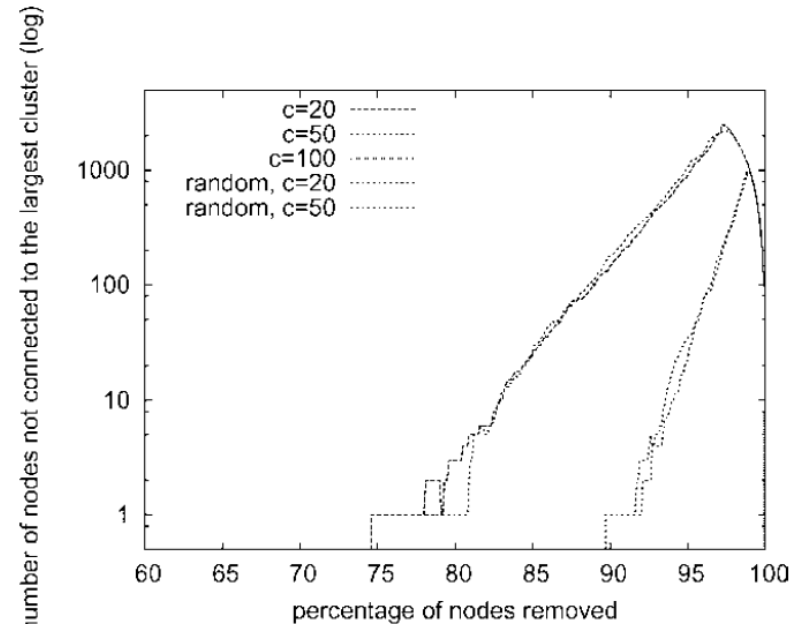
Nachbarn die den Knoten kennen

- Ideal ist gleichmässige Verteilung des Ingrad
- z.B. Überlastung einzelner Knoten verhindern

# CYCLON: Robustheit



(a)

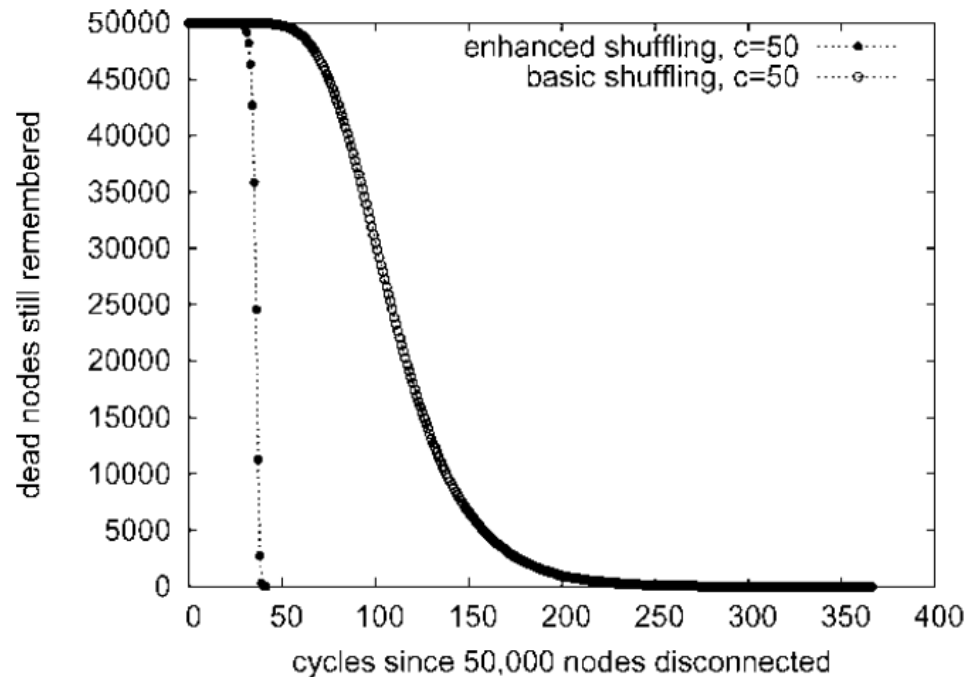


(b)

- 100.000 Knoten
- 80% der Knoten können entfernt werden bevor Partitionierung einsetzt



# CYCLON: Selbstheilung



- Anzahl Zyklen bis Deadlink aus Netzwerk verschwindet
- enhanced deutlich besser als basic

# CYCLON: Fazit

Dargestellte Eigenschaften (Robustheit, Ingradverteilung, kurze durchschnittliche kürzeste Wege) sind Eigenschaften des Gesamtsystems.

CYCLON schafft es diese dezentral, mit minimalen Aufwand und ausschließlich lokaler Sicht sehr gut zu approximieren.

# Quellen

- Spyros Voulgaris, Daniela Gavidia, Maarten van Steen, **CYCLON: Inexpensive Membership Management for Unstructured P2P Overlays**. Journal of Network and Systems Management, Vol. 13, No. 2, June 2005
- Anne-Marie Kermarrec, Maarten van Steen, **Gossiping in Distributed Systems**. ACM SIGOPS Operating Systems Review - Gossip-based computer networking, Volume 41 Issue 5, October 2007