Network Coding in P2P live streaming

von Niklas Goby

Einleitung (1)

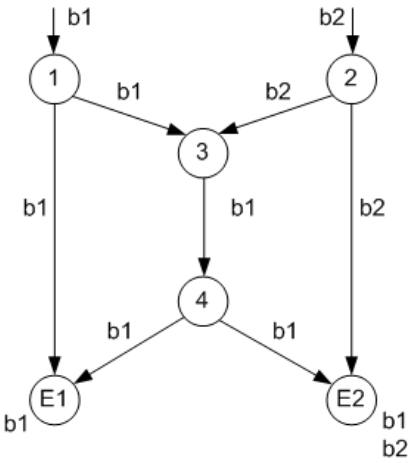
- Anforderungen an ein Live Stream Protokoll
  - Flüssige Wiedergabe
  - Skalierbarkeit
  - Auf Peer Dynamiken reagieren
  - Möglichst geringe Wiedergabeverzögerung

Einleitung (2)

- Bedingungen für eine faire Testumgebung
  - 1. eine Große Anzahl an TCP Verbindungen und UDP Traffic muss von jedem Peer effizient geleitet werden.
  - 2. eine optimierte Implementierung der Protokolle, mit Augenmerk auf maximale Performance.
  - 3. Live Streaming Protokolle mit und ohne Network Coding sollen fair vergleichbar und bewertbar sein
  - 4. Ankommende und gehende Peers sollen simuliert werden können.

Network Coding

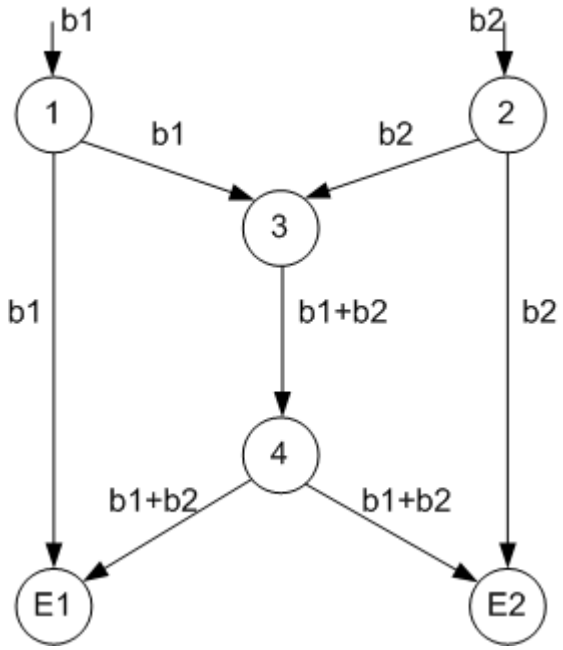
- Situation:
  - Sende Bit *b1* und *b2* an Empfänger E1 und E2
  - Jede Linie überträgt nur ein Bit
- Ohne Network Coding
  - Knoten 3: *b1* und *b2* speichern und nacheinander versenden



nach [1]

Network Coding

- Mit Network Coding
  - Knoten 3 codiert *b1* und *b2* zu *b1+b2*
  - E1 decodiert aus *b1* und *b1+b2* die gesendeten Bits *b1* und *b2*
    (E2 analog)



nach [1]

Einsatz in P2P live streaming (1)

Lava [2]

- Randomisiertes Network Coding
- Aufteilung jedes Segmentes in *n* Blöcke $[b_1, b_2, \ldots, b_n]$
- wähle zufällig eine Menge von Coding Koeffizienten $[c_1, c_2, \ldots, c_n]$
- Berechne codierten Block:

$$x = \sum_{x=1}^{n} c_i \cdot b_i$$

Einsatz in P2P live streaming (2)

Decodierung

- Sammle n codierte Blöcke, $\mathbf{x} = [x_1, x_2, \ldots, x_n]$
- Baue eine nxn Koeffizienten Matrix A
- Sei $\mathbf{b} = [b_1, b_2, \ldots, b_n]$

- Löse:

$$\mathbf{b} = \mathbf{A}^{-1}\mathbf{x}^{\mathbf{T}}$$

R^2

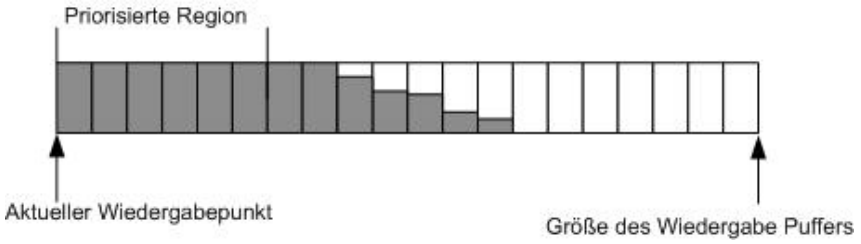
- „Random Push“ mit „Random Network Coding“ siehe [3]
- Random Network Coding
  - Unterschied zu Lava [2]: wähle *m* Coding Koeffizienten mit *m<= n*
  - Wähle zufällig *m* Blöcke aus den *n* original Blöcken
  - Generiere codierten Block:

$$x = \sum_{i=1}^{m} c_i b_i$$

- Decodierung identisch wie in Lava

Streaming in R^2 (1)

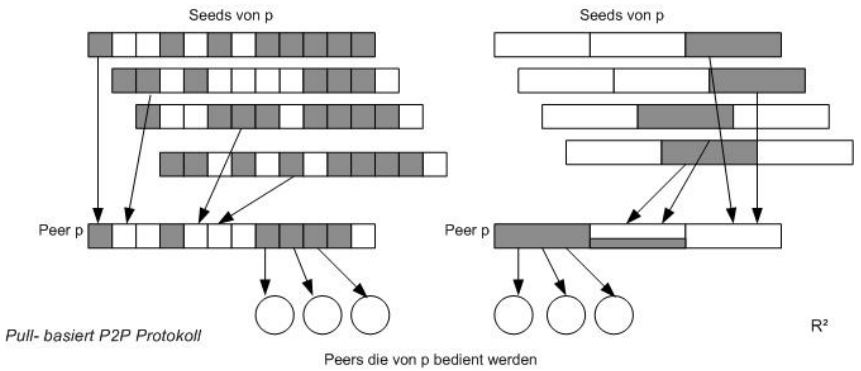
- Random Push:
  - Seeds senden vorrangig codierte Blöcke aus der „priority region“ ohne „request“
  - Seeds wählen zufällig zu sendende codierte Blöcke
  - Nur austausch von Buffer Maps notwendig (nicht periodisch sondern nach Zustandsänderung, wenn möglich in codierte Blöcke integriert)



nach [3]

Streaming in R² (2)

- Segmente können von mehreren Seeds bedient werden
  - Anzahl der bedienten Peers ist beschränkt



nach [3]

Streaming in R^2 (3)

- Eintritt neuer Peers
  - Austausch von Buffer Maps
  - Versand von Segmenten die nach der Wiedergabeverzögerungsdauer liegen (ermöglicht eine Synchronisierte Wiedergabe)
  - Wiedergabe Beginn nach Ablauf der Verzögerungsdauer

Test: Das Vergleichs Protokoll Vanilla

- Pull-Protokoll: Seeds senden Segmente nur nach request
- Verwendet kein Network Coding
- Ein Segment kann nur von einem Peer geladen werden
- Buffer Maps werden periodisch ausgetauscht
- Dient als Gerüst für das Network Coding Plugin

Die Testumgebung

- Ausrüstung
  - 44 (bzw. 48) Dual-Core Server
  - Über Gigabit Ethernet untereinander verbunden
  - Bestimmen der Upload Bandbreite auf der Anwendungsschicht.
- Parameter
  - Ein Segment repräsentiert vier Sekunden Wiedergabezeit
  - Ein Segment ist unterteil in 128 Blöcke
  - Jede Session dauert zehn Minuten
  - Puffer Größe: 32 Sekunden
  - Priority Region: acht Sekunden
  - Wiedergabeverzögerungsdauer: sechzehn Sekunden

  (siehe [3])

Vergleichswerte

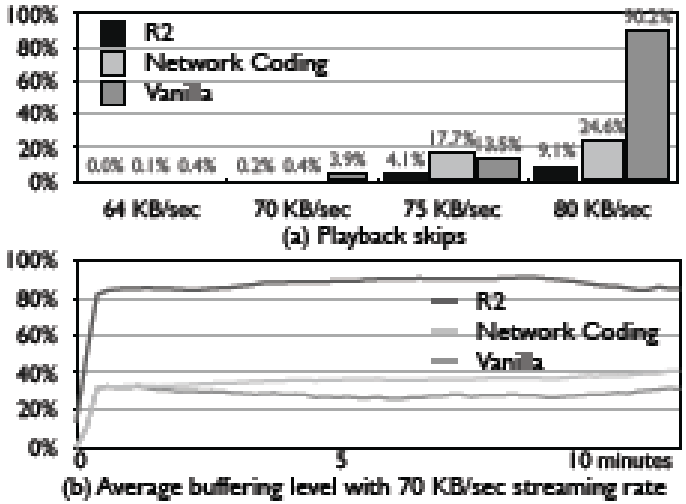
Siehe [3]:

- *Playback Skips*: Anzahl der ausgelassenen Segmente während der Wiedergabe. Ein Segment wird ausgelassen, wenn es zum Wiedergabezeitpunkt nicht vollständig geladen ist.
- *Bandbreiten Redundanz:* Prozentualer Anteil der verworfenen Segmente oder Blöcke aller empfangenen Segmente oder Blöcke.
- *Buffering Level*: Gemessener prozentualer Anteil aller empfangenen Segmente und Blöcke im Wiedergabe Puffer.

Alle Messungen sind Durchschnittswerte über alle Peers einer Session

Test: Bandbreiten Abhängigkeit

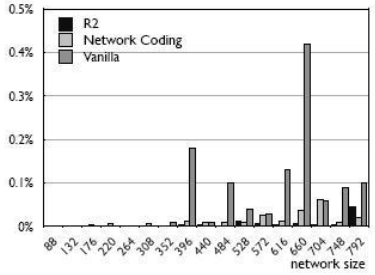
- Annahme:
  - 800 Peers
- Auswertung:
  - R^2 konstant die kleinsten Werte (Playback Skips)
  - R^2 garantiert eine flüssige Wiedergabe
  - Hohe Puffer füllstände
  - NC Protokolle fast immer besser



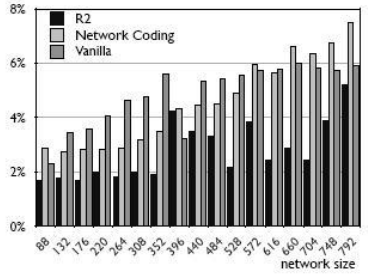
aus [3]

Test: Skalierbarkeit

- Schrittweise Erhöhung der Peer Anzahl von 88 auf 792
- Streaming Rate 64 KB/sec
- Auswertung:
  - Playback Skips:
  - NC Protokolle nahezu konstante, niedrigere Werte (unter 0.1%)
  - Dennoch alle Werte unter 0.45%
  - Bandbreiten Redundanz
  - NC Protokolle besser
  - Ab 660 Vanilla konstant, NC höher



(a) Average playback skips.

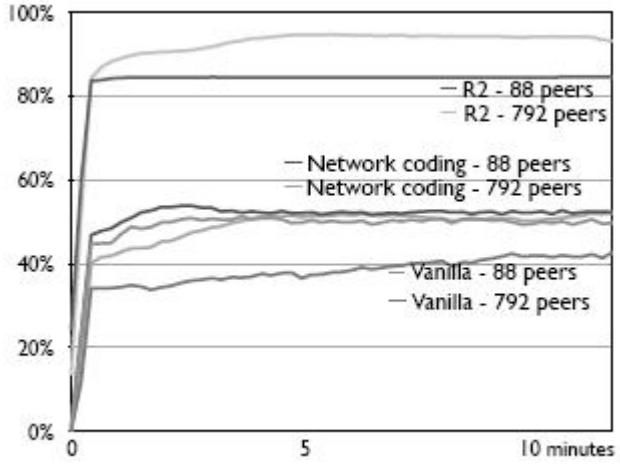


(b) Average bandwidth redundancy.

aus [3]

Test: Buffering Levels

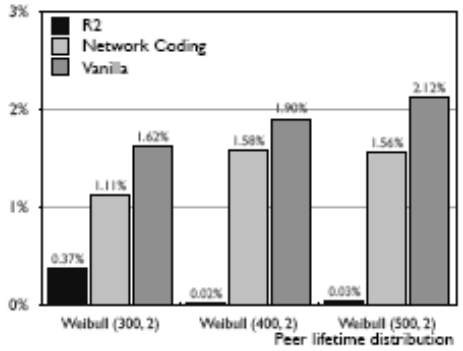
- Streaming Rate 64 KB/sec
- Alle drei Protokolle erreichen schnell ein ausreichend Hohes Level
- R^2 erreicht das höchste Level
- NC am konstantesten
- Fast kein Unterschied zwischen Vanilla und NC bei einer Peer Anzahl von 88.



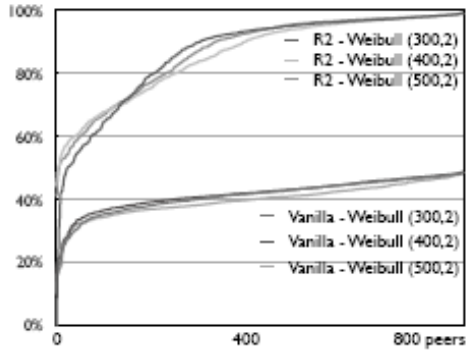
aus [3]

Test: Peer Dynamik

- Veränderungen innerhalb der Protokolle nur gering
- R^2 reagiert am Besten auf Peer Dynamiken
- R^2 fast gleichbleibend hoher Puffer Level



(b) The average playback skips.



(c) The average buffering level on each peer.

aus [3]

Fazit

Lava [2]

- Bandbreiten Angebot gerade noch größer als Nachfrage:
  - Weniger Playback Skips
  - Höherer und stabilerer Puffer Level
  - Weniger redundanter Bandbreiten gebrauch

R^2 [3]

- kürzere „initial buffering delays“
- Flüssigere Wiedergabe
- Bessere Elastizität bei Peer Dynamiken

Fazit

- Leistung der aktuellen Protokolle noch ausreichend gut
- Neue Protokolle wie R^2 zu aufwendig zum implementieren
- Kein Praxis Test mit wirklich Benutzern
  - Verhalten der User kann abweichen
  - Hardware Anforderungen könnten nicht erfüllt sein
  - Internet Anbindung können niedriger sein.

Referenzen

- [1] R. Ahlswede, N. Cai, S. R. Li, and R. W. Yeung, “Network Information Flow,” IEEE Transactions on Information Theory, vol. 46, no. 4, pp.1204–1216, July 2000.
- [2] M. Wang and B. Li, “Lava: A Reality Check of Network Coding in Peer-to-Peer Live Streaming,” in Proc. of IEEE INFOCOM, 2007.
- [3] M. Wang and B. Li, “R2: Random Push with Random Network Coding in Live Peer-to-Peer Streaming“Selected Areas in Communications, IEEE Journal on, vol.25, no. 9, pp.1655 – 1666, December 2007