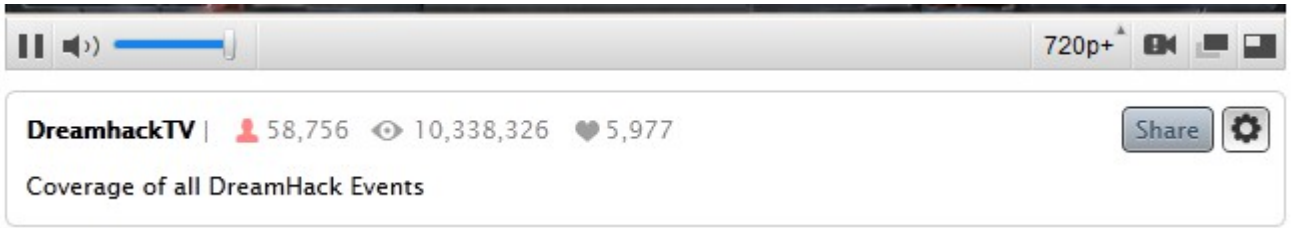
Monitoring Server für ein P2P-basiertes Live-Streaming-System

TECHNISCHE UNIVERSITÄT DARMSTADT

Motivation – Warum P2P?

TECHNISCHE UNIVERSITÄT DARMSTADT

- Live-Streaming immer beliebter
  - Besonders große Events
    - Dreamhack Winter 2011 → 94.932 gleichzeitige Zuschauer(1)
    - Global Challenge Kiew 2012 → ~250.000(2)
- Qualitätsansprüche steigen



  - 720p ~ 2,4 Mbit/s → 141Gbit/s

(1) http://www.dreamhack.se/dhw11/2011/12/06/dreamhack-and-twitch-tv-announce-record-breaking-online-viewership/

(2) http://www.esb-online.com/start/newsdetails/article/3284/

Motivation – Warum ein Monitor?

TECHNISCHE UNIVERSITÄT DARMSTADT

- Vorteile P2P-basierter Systeme
  - Nutzung der Bandbreite der Nutzer
  - Weniger Infrastruktur benötigt
  - Skalierbar
- Nachteile P2P-basierter Systeme
  - Komplexer
  - Authentifizierung
- Monitor
  - Übersichtlichkeit
  - Manipulation

Übersicht

TECHNISCHE UNIVERSITÄT DARMSTADT

- Motivation
- Grundlagen
- Existierendes System
- Anforderungen
- Realisierung der Anforderungen
- Implementierung
- Live-Demo
- Zusammenfassung und Ausblick

Grundlagen

TECHNISCHE UNIVERSITÄT DARMSTADT

- Client-Server
  - Anfang des Live-Streamings
  - Clients verbinden sich mit Server
  - Schlechte Skalierbarkeit

- P2P-Systeme
  - Provider und Konsumenten sind Peers
  - Peers empfangen und senden
  - Probleme:
    - Heterogenität der Hardware
    - Positionsfindung im Overlay
    - Verzögerung des Streams

Grundlagen - Streaming

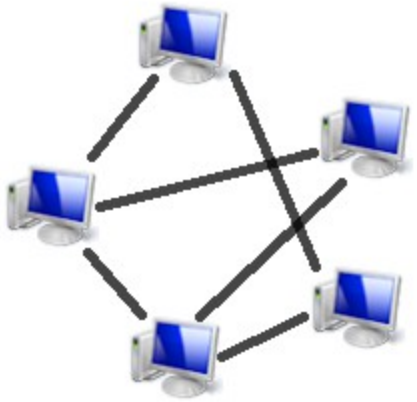
TECHNISCHE UNIVERSITÄT DARMSTADT

- Push-basierte Systeme
  - Peers registrieren sich
  - Bekommen Daten automatisch gesendet
  - Wenig Overhead und Verzögerung
  - Ausfall von Sender müssen erkannt werden
- Pull-basierte Systeme
  - Peers fragen Daten an
  - Schnelle Reaktion bei Ausfall eines Peers
  - Mehr Overhead als Push-basierte

Grundlagen - Overlays

TECHNISCHE UNIVERSITÄT DARMSTADT

- Mesh-basiert
  - Keine feste Struktur
  - Peers verbinden sich dynamisch mit anderen Peers
  - Peers werden meist mit Metriken (z.B. Delay) ausgewählt
  - Pull-basiert
  - Robust gegen Ausfall von Peers



Grundlagen - Overlays

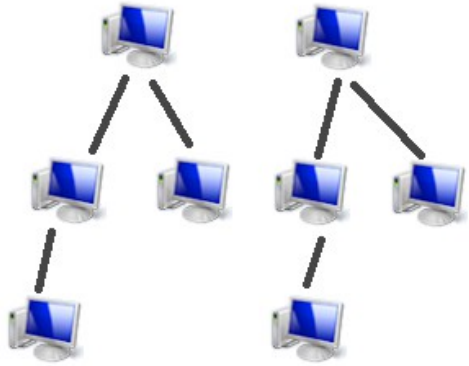
TECHNISCHE UNIVERSITÄT DARMSTADT

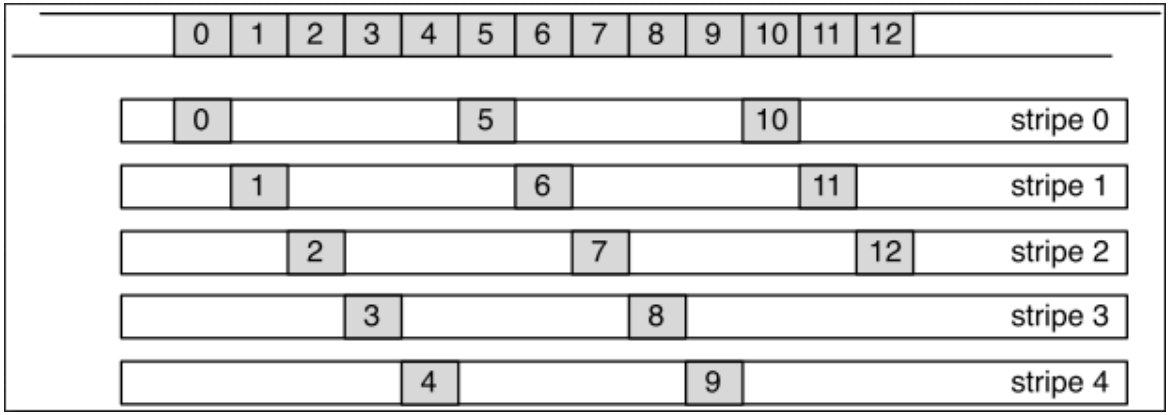
- Tree-basiert
  - Peers verbinden sich mit Elternknoten
  - Daten werden von Eltern an Kinder übertragen
  - Push-basiert
  - Probleme
    - Nicht robust
    - Blatt-Peers nutzen nicht ihre Bandbreite

Grundlagen - Overlays

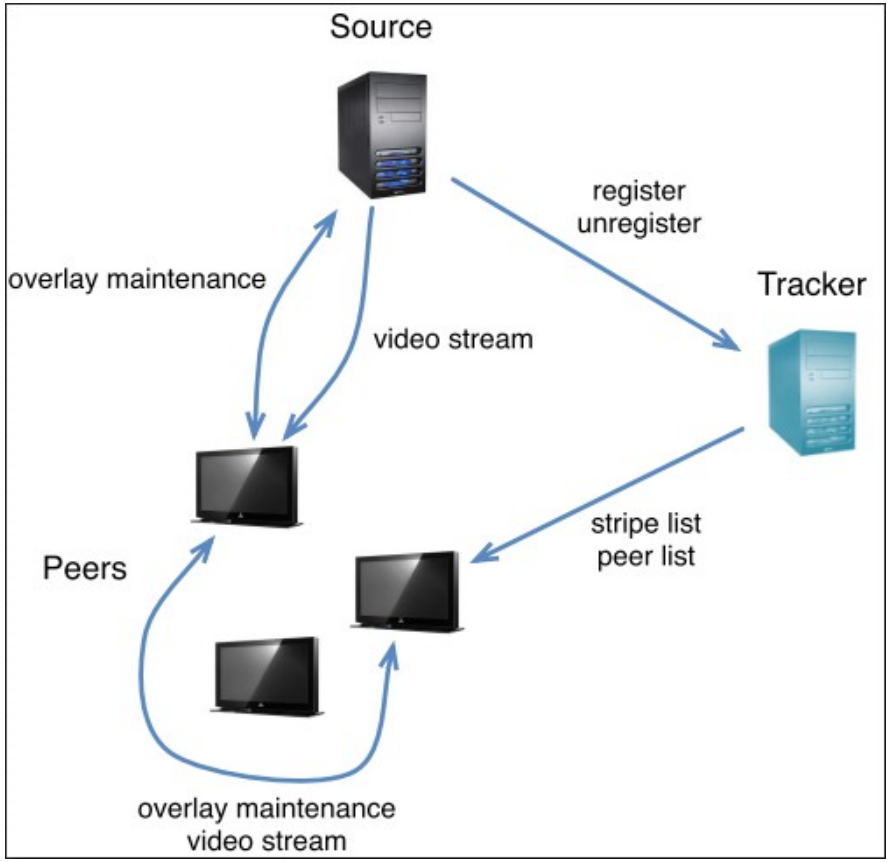
TECHNISCHE UNIVERSITÄT DARMSTADT

- Lösung: Multi-Tree
  - Aufteilung des Streams in Stripes
  - Für jeden Stripe wird eine Baumstruktur erstellt





Existierendes System (1/3)



Existierendes System (2/3)

TECHNISCHE UNIVERSITÄT DARMSTADT

- Tracker:
  - Pflegt Liste der Streams
  - Sowie zugehöriger Source und Peers
  - Dient als Einstiegpunkt für neue Peers
- Source:
  - Quelle → Wurzel aller Bäume
  - Aufteilen des Streams in Stripes
  - Registriert Stream beim Tracker
  - Verhält sich ansonsten wie ein normaler Peer

Existierendes System (3/3)

TECHNISCHE UNIVERSITÄT DARMSTADT

- Peer:
  - Fragt Streamliste von Tracker an
  - Fragt Peerliste von Tracker an
  - Verbindet sich für jeden Stripe mit einem Peer
  - Empfängt Stream und sendet ihn an seine Kinder
  - Versucht Overlay zu optimieren
    - z.B. anhand der verfügbaren Bandbreite
      - Weniger Bandbreite verfügbar
        - Leitet Kinder an seine Kinder weiter
      - Mehr Bandbreite verfügbar
        - Macht Kinder von Kindern zu seinen Kindern
  - Fügt Stripes zu Stream zusammen

Anforderungen (1/2)

TECHNISCHE UNIVERSITÄT DARMSTADT

- Visualisierung der Informationen
  - Baum-Struktur der Stripes
  - Nicht-verbundene Peers getrennt anzeigen
  - Eigenschaften der Peers
- Sammlung von Informationen
  - Streams
  - Peers
  - Verbindungen zwischen Peers

Anforderungen (2/2)

TECHNISCHE UNIVERSITÄT DARMSTADT

- Manipulation des Netzwerks
  - Beenden von Peers
  - Peers zum „Absturz“ zwingen
  - Eigenschaften von Peers ändern
    - Bandbreite
- Konfigurationsmöglichkeit
  - Anordnung von Stripes und Peers
  - Bilder für Peertypen
  - Definition von Peer Eigenschaften

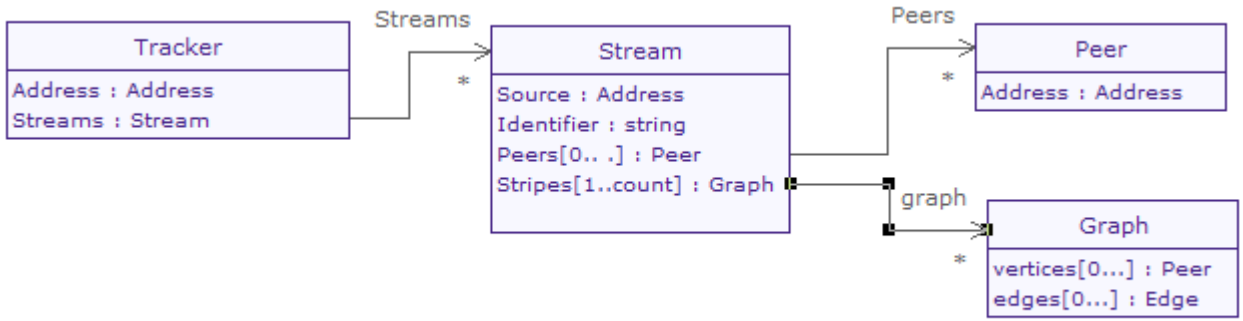
Realisierung (1/3)

- Visualisierung der Informationen
  - Graphen Framework
  - Layouts für die Anordnung von ...
    - Stripes
    - Peers
      - Baum-Struktur
      - Ebenen-Struktur
    - Während der Laufzeit einstellbar

Realisierung (2/3)

TECHNISCHE UNIVERSITÄT DARMSTADT

- Sammlung von Informationen
  - Monitor
    - Streams → „MGetStreams“
    - Peers → „MGetAllPeersForStream“
  - Peers/Source
    - Verbindungsinformationen
      - „GetChildren“
      - „GetParents“
  - Datenstruktur:



Realisierung (3/3)

TECHNISCHE UNIVERSITÄT DARMSTADT

- Manipulation des Netzwerks
  - Beenden von Peer
    - Normales → „Term“
    - Absturzsimulation → „Kill“
  - Eigenschaften von Peers
    - abrufen → „GetInfo“
    - setzen → „SetProperties“
- Konfigurationsmöglichkeit
  - XML-Format
    - Layouts
    - Bilder für Peertypen
    - Eigenschaftsspezifikationen

Implementierung (1/2)

TECHNISCHE UNIVERSITÄT DARMSTADT

- Verwendete Bibliotheken
  - Graphen Framework „JUNG“(1)
    - Strukturen für Graphen
    - Visualisierungs-Engine
      - Layouts
      - Darstellung von Knoten und Kanten
      - Ereignisse

(1) http://jung.sourceforge.net/

Implementierung (2/2)

TECHNISCHE UNIVERSITÄT DARMSTADT

- Netzwerkkommunikation
  - Nutzt lso.MaintenanceNode vom bestehen System
  - Diese nutzt Google Protocol Buffers
- Graphen Framework
  - Visualisierung mit Hilfe von VisualizationViewer
  - Plugins
    - Layout
    - Ereignisse
    - Graphische Darstellung
- Benutzerinterface
  - Swing

Live-Demo

Zusammenfassung & Ausblick

TECHNISCHE UNIVERSITÄT DARMSTADT

- Netzwerk wird visualisiert
- Strukturfehler werden gemeldet
- Manipulationsmöglichkeiten

- Erweiterungsmöglichkeiten
  - Android
  - Timeline

Vielen Dank für Ihre Aufmerksamkeit!