P2P Wissenswertes

Pro Contra P2P

Pro: kein Single Point of failure, keine zentrale Überwachung, Kostenreduktion

contra: Routing Probleme, Zuverlässigkeit, änderungsrate des Systems, muss Erreichbarkeit gewährleistet sein?

Bittorrent

Chunk selection Strategien in Bittorrent

Strict Priority

Erst alle Sub-pieces downloaden, erst dann das nächste Piece vom Peer

Rarest First

Die seltensten Pieces im Peer-Set downloaden, Ziel ist e seine möglichst schnelle verstreute Teilung zu erreichen. Dient dazu die Upload-Kapazität gut auszunutzen

First Random Piece

Solange zufällige pieces downloaden, bis der Download des ersten Piece abgeschlossen ist.

Endgame Mode

Die verbleibenden Sub-Pieces bei allen Peers im Peer-Set downloaden.

Pareto Effizienz in Bittorrent ist eine lokale Optimierung um ein globales Optimum bezüglich der Anzahl der handelnden Daten zu erreichen, basierend auf dem tit-for-tat Paradigma.

Choking Algorithmus in Bittorrent (mit rarest first ausreichend um bittorrent zu betreiben)

→unchoken der vier besten uploader im peer set

→basierend auf der downloadrate

→alle zehn sekunden neuberechnung welcher peer „geunchoked“ wird

- **Optimistic unchoking**: - alle 30 sekunden zufälligen peer unchoken, unabhängig von downloadrate
- **Anti snubbing**: choken der peers, die in den letzten 60 sekunden kein chunk hochgeladen haben
- **Upload only**: uploaden zu den peers mit den höchsten downloadraten, wenn man zum seeder geworden ist

Finger Table Information- Welche Informationen müssen gespeichert werden in einer Chord Finger Table, um korrekt zu routen?

IP adresse des betreffenden knoten und den genutzten Port speichern. Optional den identifier.

Duplicate Finger Table Entries-Szenario indem keine doppelten Einträge auftreten

Um das zu verhindern, muss ein Knoten für jede ID im identifier space vorhanden sein.

Kademlia

Vorteile von XOR in DHT

- Nodes erhalten requests von den gleichen „Knotenverteilungen“ die in ihren Routing tables enthalten sind. Nodes können von jedem querie, die Routing Informationen lernen.
- Es garantiert eine lookup Konvergenz
- Request können an jeden node im Intervall geschickt werden. Parallele, asynchrone queries können gesendet werden.

Update Policy

Der Sinn ist, aktive nodes nicht zu entfernen.

Es werden die Einträge im k-bucket angepingt. Wenn der gepingte node antwortet, werden die neuen node informationen am anfang der Liste abgelegt. Ansonsten wird der node entfernt und der neue node wird an das Ende der Liste angefügt.

Lookup algorithm

Ziel ist es die k nächsten nodes zu einer gegebenen ID zu finden. Der Inititator wählt alpha-nodes vom nächsten nicht leeren k-bucket.

Gleichzeitig sendet er FIND_NODE RPCs zu diesen alpha-nodes. Er wählt alpha nodes von resultierenden k nodes die am nächsten zu der target ID sind. Das wiederholt er bis keine neuen nodes gefunde wurden.

Das ist ein rekursiver algorithmus. Verglichen aber mit unserer Auffassung über rekursiv/iterativ von DNS ist es ein iterativer algorithmus, da der initiator alle anfragen sendet und antworten empfängt.

DHT in bittorrent

Bietet ein dezentralisiertes und verteiltes tracking der nodes.

CAN

Join prozedur im CAN Overlay

- Joining node A kontaktiert einen bootstrap-node B
- A wählt eine identifier ID
- B routet JOIN-Anfrage in Richtung C mit der ID
- C teilt eigene Zone in die Hälfte und weist A eine Hälfte zu
- A baut verbindung zu nachbarn auf
- A benachrichtigt nachbarn
- C updatet nachbarsliste

DHASH

Chord, eine DHT?

Nein, Chord ist kein DHT. Es bietet kein speichern von paaren (key, value) und es verbindet nicht values mit identifieres.

Chord ist ein skalierbarer P2P lookup service. Es verbindet key mit nodes und nicht mit values. Successor lookup ist die einzige implementierte funktionalität.

DHASH

Es ist eine distributet hash table. Es implementiert chord als lookup service. Es existieren die funktionen insert und lookup.

Es existieren die Layer Chord, DHASH und Application.

- Chord: mappt identifiers mit den successor nodes
- DHASH: verbindet values mit identifiers
- Application: bietet ein file system interface an

Arten DOS Attacks

- Entfernen von netzwerk links: ??
- Unmengen an nutzlosen daten einfügen: block/chunk quota einführen, damit ein einzelner node nur eine bestimme anzahl von blocks/chunks speichern kann

Load Balancing in DHASH

- Keine ganzen dokumente an einem node speichern
    - Dokumente in chunk aufteilen
    - Jeden block/chunk in den DHASH layer mit dem hash vom block einfügen
- Metaden benutzen um einen einzelnen namen für die datei zu bilden, eine art file descriptor
- Verteilen eines einzelnen files unter mehreren nodes

P2P Gaming

Neighbor Discovery ist eine große Herausforderung für die Entwicklung von P2P-basierten Gaming-Overlays

Man muss mit den naheliegendsten Peers in der virtuellen Welt verbunden sein und nicht mit den naheliegendsten im identifier-space eines Overlay-Netzwerkes.

Chord benötigt das mappen zwischen der virtuellen 2D oder 3D Welt und dem 1D chord identifier space. Desweiteren bewegen sich Spieler oft, was eine hohe änderungsrate in der Kommunikationsstruktur zur folge hätte. Es müsste andauernd die position auf dem chord ring geändert werden und die nachbarn der nodes.

Bei CAN hat man ein density problem, den dicht angesiedelte areale resultieren in kleine Can zellen. Spieler bewegen sich oft, gleiches problem wie bei chord.

Information Dissemination Overlay

Ist ein unstrukturiertes P2P overlay. Erlaubt schnelle und häufige Übertragung von update Nachrichten zu den anderen Peers im Vision/Interaction Range.

Basis Idee: definiere eine Area of Interest (AOI). Nur mit den peers in AOI kommunizieren.

pSense

- Spieler sind hauptsächlich mit den naheliegendsten in der virtuellen Welt verbunden (position based multicast)
- hält verbindung zu 2 verschiedenen arten von nodes
    - **near nodes**: peer im vision range
    - **sensor nodes**: peers außerhalb der vision range
- lokaler multicast
    - updates werden direkt zu den near nodes und sensor nodes gesendet
    - dann auch weitergeleitet an peers die weiter weg sind

Donnybrook

Hauptidee ist das nutzen der mesh topology. Die update rate wird für spieler die nicht im inetressanten bereich sind reduziert. Ein spieler ist interessant für einen wenn, in der virtuellen nah beieinander, er auf ihn schießt oder sie interagieren.

Unterschiede pSense und Donnybrook

- Anzahl Verbindungen
    - pSense: anzahl peers in vision range
    - Donnybrook: verbindung zu allen spielern
- Anzahl der gleichzeitig unterstützenden spieler
    - pSense: kein limit
    - Donnybrook: über 1000 speiler gleichzeitig

Wuala Erasure Coding

- Partitionierung eines files in m coding blocks gleicher größe
- Erstelle c extra blöcke
- M+c= n coding fragmente
- Redundanz erreicht, file kann rekonstruiert werden bei blockverlust

Social Networks

Goal of Cloning Attacks

- Persönliche informationen von usern
- Vertrauen der user gewinnen
- Entweder einen klon eines existierenden user profil krieren oder auf die freundesliste eines realen users gelangen

Kinds of cloning attacks

- Klonen eines profils im gleichen OSN oder in ein andrem OSN

Warum Dezentralisierte OSN?

- Umgehung des Problem zentralisierter Architekturen
    - Zentralisierte Kontrolle, Datenspeicherung, datenzugriff, Single Point of Failure
- Schutz der benutzer privatssphäre gegen
    - Intruders, crawlers und dritten parteien, Big Brother
- Performanz?
    - Daten sicher verwalten?,
- Sicherheit?
    - Availability of Service?, Key management?, fake accounts erkennen?

Safebook

Matryoshka-Pro Teilnehmer im Netzwerk eine Matryoshka

Sie dienen und unterstützen die privacy, basierend auf hop-by-hop trust, der Datenspeicherung und Replizierung und der communication privacySie werden folgendermassen gebildet:

→Für alle Kontake („freunde“) A von V:

- V sendet A: DHT lookup keys, TTL span
- Weiterleiten der anfrage zu #span von ihren kontakten mit dezimierter TTL

→TTL läuft bei node D ab

- D registriert die lookup keys mit referenz @D
- Dient als entry point für V’s matryoshka

- Core: der besitzer initiator einer matryoshka
- Mirror: kontakte eines nodes – replziert das profil
- Entrypoint: registriert die location bei V's keys in der DHT
- Inner shell: beinhaltet alle mirrors
- intermediate shell: shells/nodes zwischen mirrors und entrypoints
- outer shell: beinhaltet alle entrypoints

Profile Lookup von User A

- rekursiven lookup im DHT ausführen
    - lookup entry an position id(a)
- liste der entrypoints erhalten
    - beinhaltet all knoten im outer shell von A's matryoshka
- anfrage zu einem dieser entrypoints senden
    - anfrage wird durch A's matryoshka weitergeleitet bis mirror erreicht wird
- daten werden zurück gesendet
- es wird kein DHT benutzt, weil
    - adressen der user wären ungeschützt
    - online status wäre für jeden sichtbar
    - setzt hop-by-hop privacy aus

Profile replication stored?

- Abhängig von der Anzahl der Kontakte→ kontaktzahl=anzahl replicas
- Speicherung einer replica an jedem kontakt

Spannung Ratio

Der Span legt die anzahl der nodes fest an denen die matryoshka erstellungsanfrage weitergeleitet wird.

- TTL: tiefe des baumes der von der matryoshka erstellt wird
- Span: breite des baumes der von der matryoshka erstellt wird
- Bei erhöhung des span wird die anzahl der entrypoints der matryoshka erhöht
- Ein span von 1 würde nur wenige entrypoints bieten

Overlays in Safebook

- Matryoshaks
    - Mittiger ring von knoten der um benutzer knoten gebaut wird
    - vertrauenswürdige daten spicherung
    - profildaten abfrage
    - hop-by-hop privacy
- P2P substrate
    - Bietet einen Lookup-Service an um entrypoints der matryoshkas zu finden
    - KAD wird als DHT benutzt
    - Pseudonyme werden als node identifier benutzt
    - Registrierte keys sind „hashed properties“ von Usern

Multicast Paradigms

Different Mutlicast Paradigms

Unicast: Funktionalitäten an der Quelle implementiert--> Daten werden an der Quelle repliziert, mehrere kopien werden über den gleichen link gesendet

IP Multicast:funktionalitäten am router implementiert (und quelle)-->daten repliziert beim router, jeder link wird exakt einmal benutzt, router fordern status informationen an

End System Multicast: funktionalitäten an den kanten des netzwerkes implementiert--> P2P -based an den end hosts

Scenarios

Unicast: Keine infrastruktur, hoher user churn, sender besitzt genug recourcen

IP Multicast: bestehende Infrastruktur

End System Multicast: geringe recourcen der sender, große user gruppen

Application Multicast

Tree Based Topologies

- explizite Konstruktion des Streaming trees
- parents leiten packete an children (push)
- stream könnte gesplittet werden -->multiple trees
- Pro: geringer overhead/delays
- Contra: geringe Robustheit
- Application: live-streaming, broadcast-like streaming

Mesh Based topologies

- streams geteilt in chunks
- jedes peer fragt die chunks nach die es braucht (pull)
- mehr oder weniger willkürliche verbindung zwischen peers
- pro: sehr robust
- contra: langsam, hohe delays, signifikanter overhead
- application: video-on-demand streaming

Differences

- explizite vs. Implizite tree construction
- push vs. Pull
- QoS vs. Robustness
- Live streaming vs. VoD streaming

Pull based Streaming

- jeder part des stream explizitangefragt
- multiple connection erforderlich
- konstanter austausch der status informationen erforderlich

Push based streaming

- automatischer forward stream zu den predecessors
- daten nur von parents erhalten, weiterleiten an children
- kein state erforderlich, geringer maintenance overhead

Differences

- requesting vs. Automatic delivery
- bidrektionale vs unidirektionale daten übertragung
- state maintenance vs. Topology maintenance
- implizit vs. Explizite toplogy generation

P2P Netzwerktypen: Strukturiert/Unstrukturiert

Motivation für DHTs: Steigerung der effizienz be Such und Objektlokalisation in P2P-Netzwerken→Adressierung anstatt Suchen

„Stress“: Menge identischer Packete die den gleichen physischen link traversieren

Zwei Generelle Funktion von ALM (Application-Layer-Multicast): Lokalisation und Inhaltsverbreitung

Anforderungen an Online Games: Sicherheit, Kosten, Robustheit, Fairness

P2P für Games: Contra: Fair, Sicherheit, Robustheit

Pro: Single Point of Failure (Server), Kosten, Inhalt

Typische Eigenschaften von P2P Systemen:

1. Unzuverlässig, Unkoordiniert, Unverwaltet
2. Widerstandfähiger gegen Attacken
3. Große Anzahl an Ressourcen

Vergleich zwischen BitTorrent, Napster, Gnutella, FastTrack a.k.a KaZaA

BitTorrent:

- Ziel ist die schnelle und verlässliche Verteilung einer Datei an einer großen Anzahl Clients
- Bildet ein Netzwerk (Swarm) für jede verteilte Datei
- Vorteile von BitTorrent: Versenden von „links“ an Freunde und Links verweisen immer auf die gleiche Datei
- Datei wird in chunks aufgeteilt→Erlaubt paralleles herunterladen
- .torrent-datei beinhaltet Metadaten über die Datei, Tracker-URL und Größe der Chunks→Meistens hosted auf einem Web-Server
- Seeder ist der Server der die original Datei hosted bzw. Client mit einer vollständigen Kopie der Datei
- Leecher ist ein Client der die Datei runterlädt
- Client lädt .torrent Datei und kontaktiert Tracker, Client baut Verbindung zu 20-40 peers auf→Peer-Set
- Stärken: Funktioniert ganz gut, sobald peer genug chunks hat ist downloadrate gut
- Schwächen: Jeder muss mitmachen→problem für clients hinter Firewall, dateien müssen groß genug sein (chunk→256kb)

Napster

- Ein zentraler Index Server
- User registriert sich an zentralem Server
- Sendet liste der Dateien die geshared werden sollen
- Server kennt alle Peers und Files im Netzwerk
- Suchanfrage bringt als Resultat: Informationen über File und Peer und andere Metadaten
- Stärke: Überblick auf das ganze Netzwerk, schenlles und effizientes Suchen, Suchanfrage wird korrekt beantwortet
- Schwächen: downloaden von einzelnem peer, zentraler server ist single point of failure, index server braucht genug ressourcen um alle anfragen zu bearbeiten

Gnutella

- Dezentralisiert
- Basiert auf overlay netzwerk
- Hat nur peers→alle peers sind gleichberechtigt
- Peers werden servents genannt
- Um netzwerk beizutreten, braucht ein Peer die Adresse eines anderen Peers im Netzwerk
- Sobald Peer Netzwerk betritt lernt es über andere Peers und die Netzwerkstruktur
- Anfragen werden im Netzwerk geflutet. TTL kommt zum Einsatz, damit Anfarge nicht ständig existiert→nachrichten werden an alle nachbarn gesendet. Ein Query wird mit einem Queryhit beantwortet, das die IP und die Port nummer des Peer enthält.
- Download findet direkt zwischen Peers statt
- Für peers hinter firewalls gibt es die push message. Peer außerhal der firewall sendet Push zu peer innerhalb der firewall via „relay". Dabei gilt die annahme, das peer innerhalb der firewall eine TCP connection zu einem nachbar-peer im overlay hält
- Peer innerhalb firewall kontaktiert peer welches PUSH gesendet hat→download
- Stärken: völlig dezentralsiert Netzwerk, open protocol , robust gegenüber node failure
- Schwächen: flooding ist extrem ineffizient, QUERIES ebenfalls nicht

KaZaA

- Benutzt FastTrack Protocol
- Gibt ON, normaler peer von user, und SN, ebenfalls peer, aber mit mehr ressourcen und verantwortlichkeiten
- KaZaA formt eine zwei Schicht hierarchie→Top level nur SN, lower level nur ON
- Ein ON gehört zu einem SN
- ON senden Anfragen an ihrem SN→er kümmert sich um den Rest
- SN hat alle Informaionen über sein ON
- Ein peer kann es ablehnen ein SN zu werden (geringe Bandbreite z.B.)
- Stärken: effizientes suchen mit geringer ressourcen belastung durch SN Einsatz
- Schwächen: Suche nicht flächendeckend, singe Point of failure→SN