Zusammenfassung Soziotechnische Informationssysteme

Six degrees of separation:

Experiment von Milgram: Jeder Mensch ist über 6 Kanten (first-name basis), also Freunde/Bekannte mit jedem anderen verbunden.

Strong ties:

Starke überlappende Interessen, viel Kommunikation (gut Freunde/Familie)
Sind sozial sehr wichtig, für soziale Netze allerdings unwichtiger, da diese Bekanntschaften auch ohne Netze gepflegt werden.

Weak ties:

Grobe Bekanntschaften/ehemals Bekannte (z.B. alte Schulkammeraden etc.) zu denen man ohne soziale Netze fast keinen oder überhaupt keinen Kontakt mehr hätte.
Sehr wichtig zur Ausbildung sozialer Netze. Weak ties bieten Anhaltspunkt für Werbung/Kontaktknüpfung, die eine Person ohne „Empfehlung" von sozialen Netzen nicht erreichen würden.
Weak ties bilden Brücken zwischen den Clustern/Cliquen der strong ties.

Absent Ties:

Keine oder unwesentliche Beziehungen, können vernachlässigt werden.

Triadic closure:

Wenn A B kennt und B C, dann kennt A höchstwahrscheinlich auch C. Eine sehr häufige Struktur in sozialen Graphen

Clustering Koeffizient:

Geschlossene Triaden: Drei Knoten, die mit drei Kanten verbunden sind
Offene Triaden: Drei Knoten, die mit nur zwei Kanten verbunden sind
Clustering Koeffizient: $\frac{\#Geschlossene}{\#Geschlossene+\#Offene}$

Preferential Attachment:

„Reiche werden reicher" -> Ein Knoten mit vielen Kanten hat eine hohe Chance weitere Kanten auszubilden

Barabasi-Albert Graph (PA):

Skalenfrei: Kantengrade der Knoten entspricht Potenzgesetz
Gegeben: Initiale Knotenmenge >1 Kantengrad mind. 1
Generierung:

1. Füge einen neuen Knoten hinzu
2. Berechne für jeden Knoten die Chance eine neue Kante zu bekommen:
   $p_j = \frac{k_j}{\sum_i k_i}$ mit k=Kantengrad
   Knoten mit vielen Kanten haben also eine höhere Chance eine neue Kante zu bekommen
3. Ordne jedem Knoten einen Bereich zwischen 0 und 1 zu (abhängig von dessen p_j)

4. Würfle eine Zufallszahl zwischen 0 und 1 und erstelle eine Kante zwischen dem neuen und dem der Zufallszahl zugeordneten Knoten

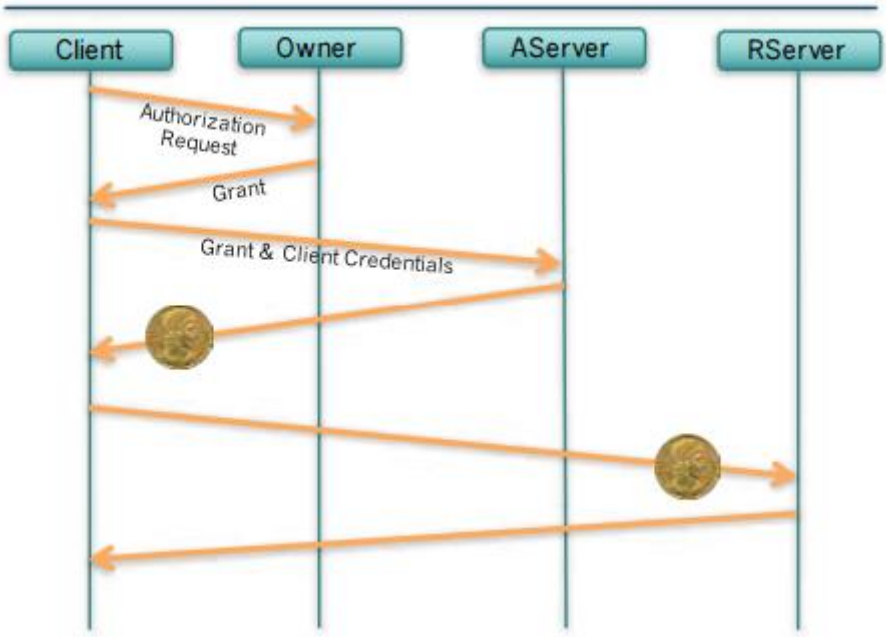
REST:

JSON basierte API (Representational state transfer)
Verwendet http Operationen
- GET für Anfragen
- POST für Erstellen
- PUT für Ändern
- DELETE für Löschen

Jedes Objekt/State hat eine eigene URL

OAuth:

Authorisationsprotokoll (darf ich das?)
Erlaubt Zugriff einer Ressource auf Daten/Funktionen eines anderen Service. Dabei wird ein externer Authorisationsservice genutzt, der ein Token für diese Ressource ausstellt.
Cross-Authentifizierung (Passwort antipattern)



Hadoop:

Name Node: Verwaltet Data Nodes, bearbeitet eingehende Anfragen, legt den Replikationsgrad fest
Data Node: Speichern die Daten, führen MapReduce aus
Job Tracker: Sammelt Ausgaben der Data Nodes und aggregiert sie
HFS (Hadoop Distributed File System) kennt den Ort von Daten -> es wird die nächste Node gewählt, ideal auf demselben Rechner

MapReduce:

Soll möglichst stark horizontal skalierbar sein
Funktioniert sehr gut bei semistrukturierten Daten (Key/Value)
Map Phase: Auf jeder Node werden Paare herausgefiltert, die einer Suchanfrage entsprechen
Reduce Phase: Auf einer oder wenigen Nodes werden diese gefundenen Paare vereint/verrechnet

HBase:

Basiert auf Hadoop
Erlaubt im Gegensatz zu Hadoop schnellen Zugriff auf einzelne Records
Matrix Schlüssel x Daten (Eine große Tabelle „Bigtable“)

SQL/NOSQL:

SQL: Tabellen, Transaktionen. Verteilung nur über durch Kopieren von Tabellen erreichbar.
Üblicherweise weniger als 10 Nodes, da JOINS schlecht horizontal Skalieren
NOSQL: Semistrukturierte Daten, leicht horizontal skalierbar
NOSQL besitzt kein Schema (Key-Value,Document,Column-Family,Graph)
Key-Value: Bekannt, ein Schlüssel und ein dazugehöriger Wert
Document: Teilstrukturiert, JSON etc.
Column-Family: Große Tabelle, BigTable etc.
Graph: Speichern Graphen als Datenmodell

BASE vs ACID:

ACID:
Atomicity: Unteilbarkeit von Transaktionen, entweder es wird alles ausgeführt oder nichts
Consistency: Datenkonsistenz bleibt auch bei fehlgeschlagenen Transaktionen erhalten
Isolation: Gleichzeitige Abfragen beeinflussen sich nicht gegenseitig (lock)
Durability: Nach Abschluss einer Transaktion sind die Daten dauerhaft gespeichert, nach Systemausfall reproduzierbar

BASE:
Basically Available: Sollte immer erreichbar sein, Ausfälle sind aber eingeplant
Soft-State: Eine Node muss die Daten nach einem Ausfall nicht selbst wiederherstellen können, es wird davon ausgegangen, dass andere Nodes diese Daten zur erneuten Replikation zur Verfügung stellen
Eventual Consistency: Es kann mehrere verschiedene Replikate geben, aber zu irgendeinem Zeitpunkt werden sie sich synchronisiert haben und konsistent sein

CAP:

Consistency: Falls es mehrere Kopien gibt, sind diese gleich
Availability: Alle Operationen liefern letztendlich ein Ergebnis
Partition Tolerance: Das System ist auch im Falle einer Partitionierung komplett einsatzbereit
Es können immer nur 2 dieser 3 Eigenschaften vorhanden sein.
CA: Vertikale Skalierung, Partitionierung heißt Ausfall – Klassische DBS
AP: Hauptsache verfügbar, hoher Replikationsgrad, Inkonsistenzen akzeptieren – DNS,NoSQL
CP: Haupsache Konsistent, es wird auf alle Knoten gewartet - Geldautomaten