Lecture 12

Overlay Applications and Mechanisms (2)

Universität Zürich[UZH]

Communication Systems Group

0. Lecture Overview

- **Hot Topics / P2P in the News / Interesting Developments**
- **Application Layer Broadcasting with Kademlia**
  - Broadcasting / Flooding
  - Structured Broadcasting
- **MapReduce**
  - Introduction
  - Examples, Demo
- **WebRTC**
- **NoSQL**
- **Tor**

Hot Topics / P2P in the News / Interesting Developments

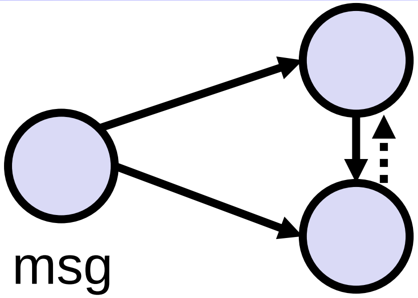
- **01.05.2015 - Ring: Chat. Talk. Share.**
  - Decentralized communication
  - OpenDHT protocol
    - Light and robust network project DHT in C++11 (30.5MB download!)
    - Based on Mainline DHT, but now incompatible
- **08.05.2015 - Bitcoin Extortion Group DD4BC Prompts Warning from Swiss Government**
  - Extortion group DD4BC – demanding 25BTC
  - "several high profile targets in Switzerland that have recently received a blackmail from DD4BC and have consequently suffered from DDoS attacks" (ref)
  - DDoS attacks consuming a bandwidth of 4 - 30 Gbit/s (observed)
    - Amplification attacks: NTP, SSDP or DNS, TCP SYN flooding, and layer 7 attacks

Hot Topics / P2P in the News / Interesting Developments

- **11.05.2015 - Bitcoin faces a crossroads, needs an effective decision-making process**
  - Raise block size from 1MB?
  - Bitcoin is decentralized – 5 core developers have lots of power
  - 1MB block size = more expensive transactions
  - Raising limit - block chain grows
  - → "Bitcoin's current governance model is seriously inadequate"
- **11.05.2015 - Please stop calling databases CP or AP**
  - CAP, will be discussed today
  - Imprecise terminology, one software – different consistence
- **14.05.2015**
  - Ascension Day / Auffahrt – No Excercises / Scrum
- **RB-Horst?**

1. Application Layer Broadcasting with Kademlia

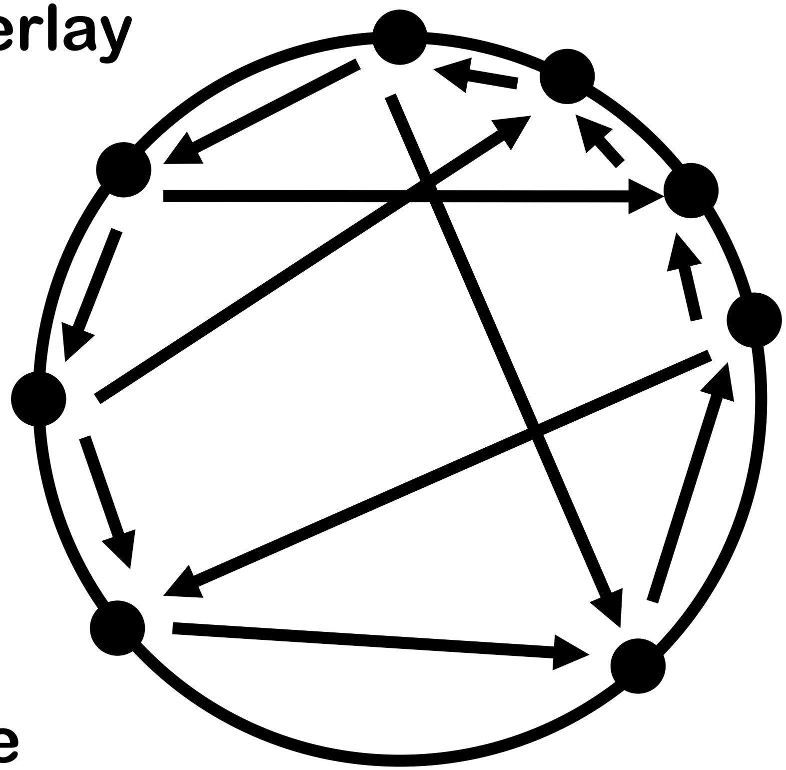
Broadcasting / Flooding in P2P



- **Flooding scales poorly**
  - Unnecessary traffic
  - 5 peers, fully meshed: 4 msg, 4 x 3 msg → 16 msg
- **Reduce Messages**
  - Don’t send to all neighbors ([modified BFS](), [random walk]())
  - [Coupon collector's problem]()
    - Number of trials grow O(n log n),
    - Reach 50 peers, it takes about 225 trials to reach all 50 peers
    - Tradeoff: all peers vs. most peers / many msgs vs. reduce msgs
- **Broadcasting use-cases**
  - Broadcast global parameters, global events (maintenance)
  - Hierarchical system: broadcast root inode (e.g. / in a file system)

Broadcasting in Structured Systems

- **5 peers fully meshed: How to send only 4 messages?**
- **Structured vs. unstructured:**
  - Knowing the direction
- **Broadcasting in Ring-based Overlay**
  - Main idea: tell other peers in what direction so send
- **Example:**
  - Send to neighbor (successor)
  - For faster spreading / redundancy send to far away peer (overhead)
  - Overhead: 4 messages
- **Kademlia? Tree-based structure**



Broadcasting in Kademlia

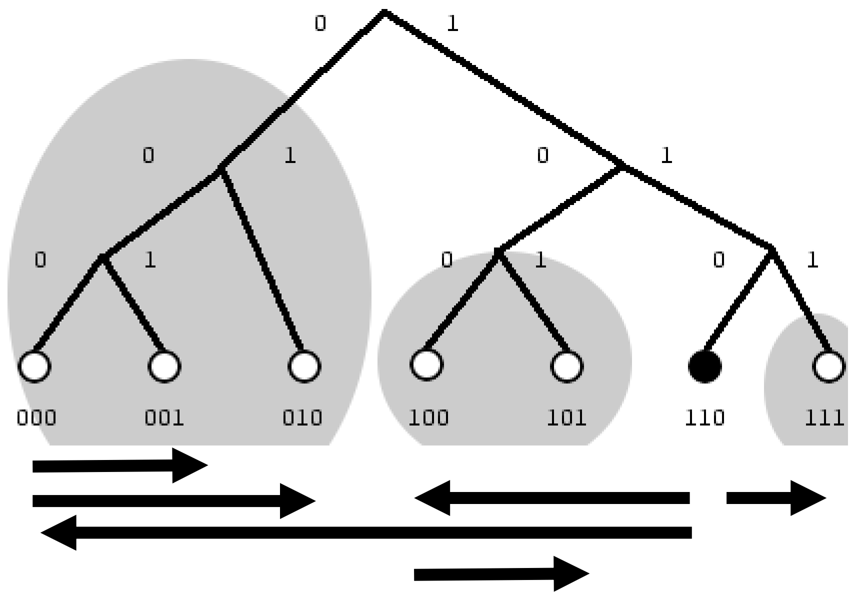
- **Tree structure – no direction, example 7 peers**
- **Main idea: send to random peer in bag X**
  - Send along in which bag this peer was, only send to smaller bags

Peer 6 (110) broadcast to b1,b2,b3		
7 (b1)	4 (or 5) (b2)	0 (or 1, 2) (b3)

Peer 4 (100) broadcast to b1		
5 (b1)	-	-

Peer 0 (000) broadcast to b1,b2		
1 (b1)	2 (b2)	-

Routing with XOR, with 3-bits



- **Peer 4 and 0 send simultaneously**

Broadcasting in Kademlia

- **7 peers, 6 messages**
  - Works only with good routing, churn?
- **Redundancy: send to R random peer in bag X**
  - Previous example: R=1, TomP2P: R=3
- **Demo: StructuredBroadcastHandler**
  - 1000 peers
  - FROM_EACH_BAG = 3, ~3800 messages
  - No redundancy: 999 messages
- **Testcase/showcase:**
  - net.tomp2p.p2p.TestBroadcast.testBroadcastTCP()

2. MapReduce

Introduction, Idea, Examples, Demo

Introduction

- **Inspired by functional programming (Lisp, Haskell, …)**
  - map square [1,2,3,4,5] → [1,4,9,16,15]
  - reduce (fold) with addition [1,2,3,4,5] → 15
- **MapReduce for distributed systems different purpose**
  - Scalability, fault tolerance
- **Main idea**
  - Map operates on a list of values to produce a new list of values, applying the same computation
  - Reduce operates on a list of values to combine those values into a single / few values, applying the same computation
- Map Reduce advantages in distributed systems, on single machine / single thread → not much gain

Map Reduce

- **Filtering / sorting**
  - `Map(k1, v1)` → `k2, list(v2)`
- **Summary operation**
  - `Reduce(k2, list (v2))` → `k3, list(v3)`

- **Used to deal with large data-sets in-parallel on large clusters in a reliable, fault-tolerant manner**
  - Parallelizable problems
  - Example: sorting petabytes in 33 minutes
- **Apache Hadoop, mongoDB, Google MapReduce, riak, (TomP2P), …**

Example

- **Hello World of Map Reduce: WordCount**
  - Input text

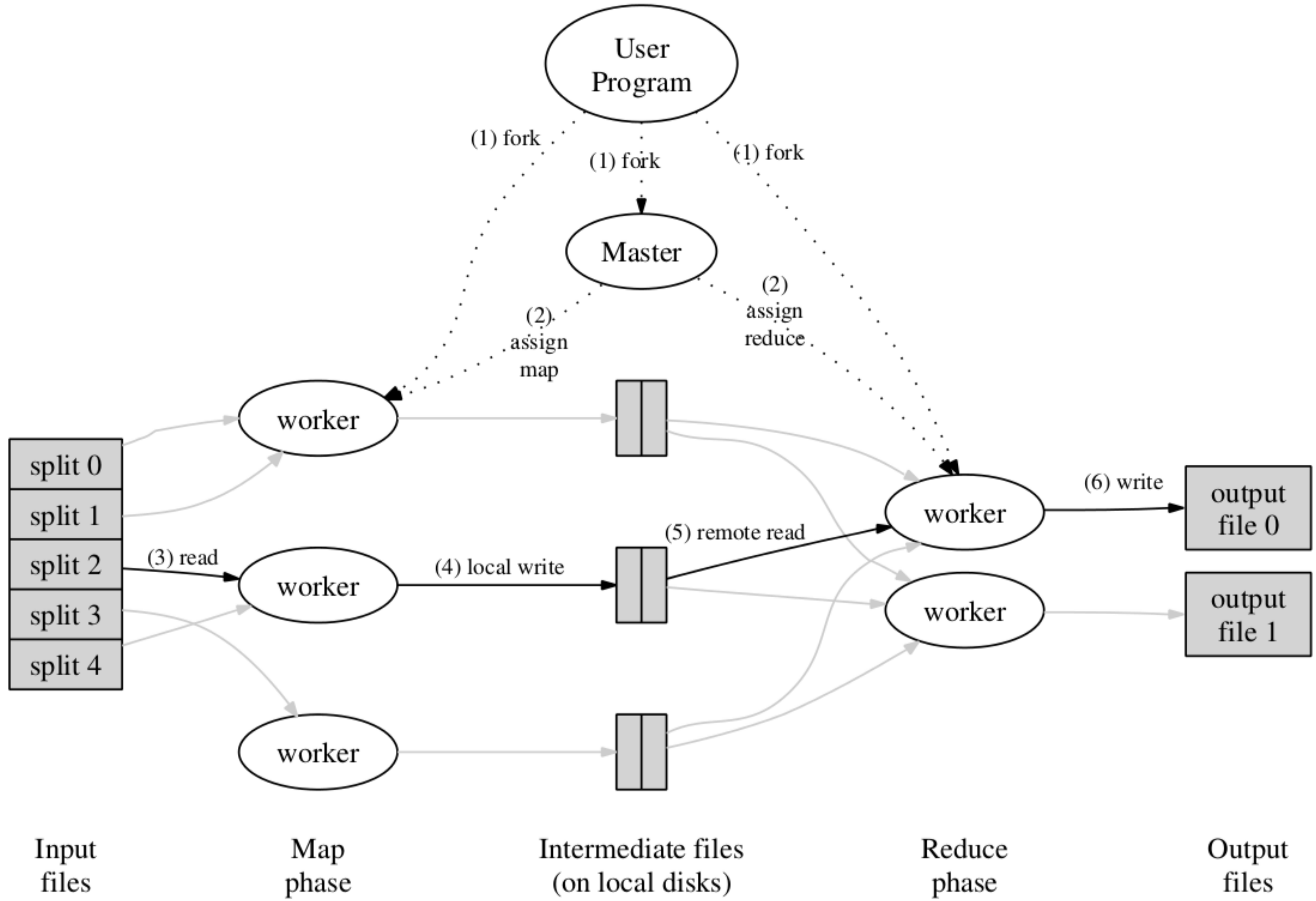
```
map (String value) {
  for each word w in values {
    store(w, 1);
}}

reduce (List intermediate_values) {
  int result = 0;
  for each v in intermediate_values {
    result += v;
    store(result);
}}
```

Example

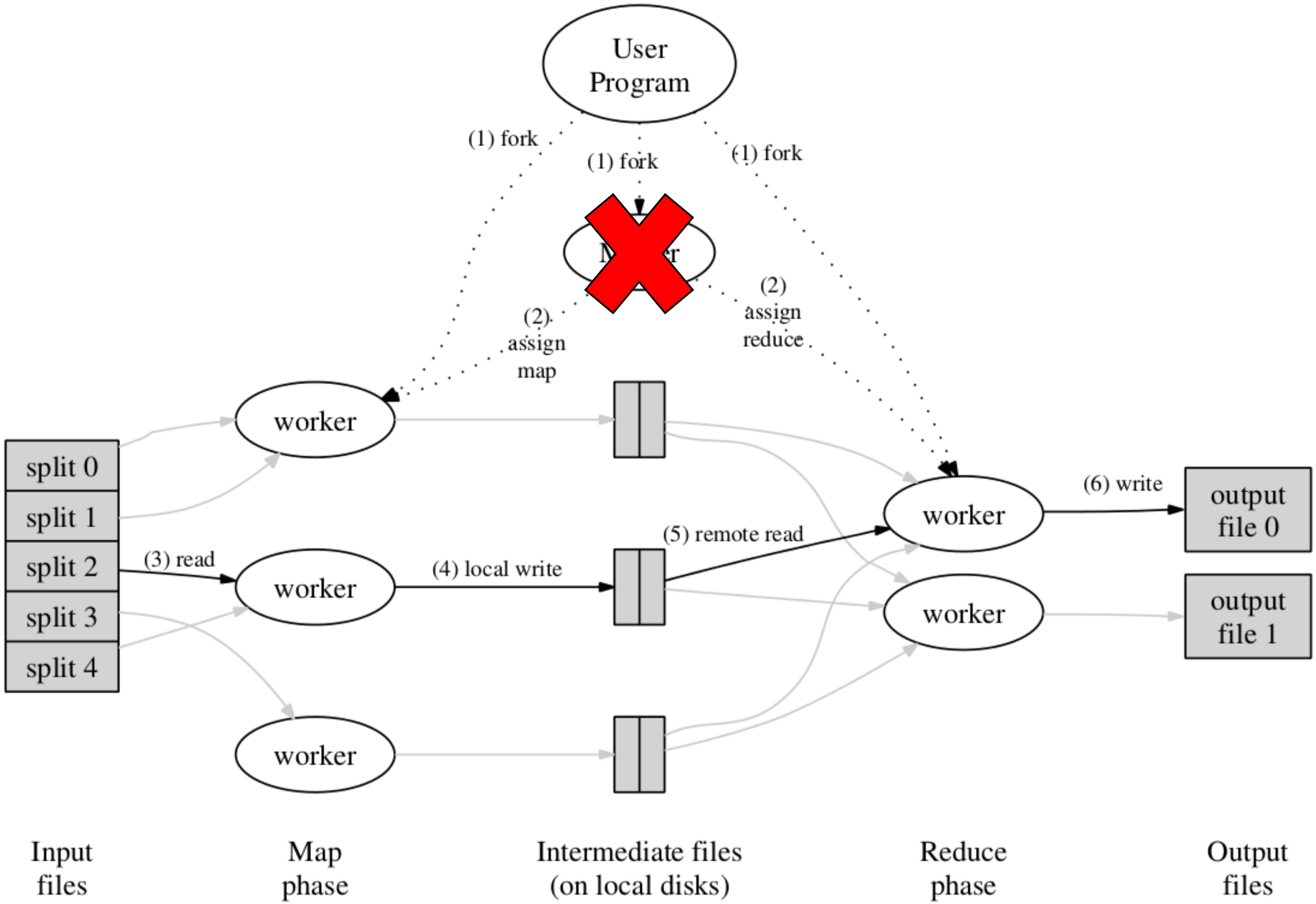
- **MapReduce example**
  - Text1: “to be or not to be”
  - Text2: “to do is to be”
  - Text3: “to be is to do”
- **Map phase**
  - store (to, 1), store (be, 1), store (or, 1), store (not, 1), store (to, 1), store (be, 1)
  - store (to, 1), store (do, 1), store (is, 1), store (to, 1), store (be, 1)
  - store (to, 1), store (be, 1), store (is, 1), store (to, 1), store (do, 1)
- **Reduce phase**
  - On node that stores “to” → add it → 6 → store it on node X
  - On node that stores “be” → add it → 4 → store it on node X
  - On node that stores “or” → add it → 1 → store it on node X
  - On node that stores “not” → add it → 1 → store it on node X
  - On node that stores “do” → add it → 2 → store it on node X
  - On node that stores “is” → add it → 2 → store it on node X
- **Result phase**
  - Get data from node X

Mechanism



Jeffrey Dean and Sanjay Ghemawat. 2008. MapReduce: simplified data processing on large clusters. Commun. ACM 51, 1 (January 2008), 107-113.

Mechanism



Jeffrey Dean and Sanjay Ghemawat. 2008. MapReduce: simplified data processing on large clusters. Commun. ACM 51, 1 (January 2008), 107-113.

Demo

- **For TomP2P 4 we need to map this to the key-value concept** → submitTask()
  - OverDHT (same category as FastSS, DST)
- **Redundancy**
  - Always record from which peer the result came
- ~~**Demo with TomP2P**~~
  - ~~Redundancy not supported yet~~
  - ~~Load balancing not supported yet~~
- tomp2p-task not yet ported
  - Low priority / documentation!

```
ReduceTask reduce = new Task() {
 exec(key, data) {
  int r[] = get(key)
  a=sumAll(r);
  DHT.addList(result, a);
}}
MapTask map = new Task(){
 exec(key, data) {
  String text = data.getText();
  for each word w in text {
    DHT.addList(w, 1)
}}}
DHT.submit(key1, map, text1);
DHT.submit(key2, map, text2);
DHT.submit(key3, map, text3);
//start reducer once all are done
for each word w from futureTasks {
  DHT.submit(w, reduce);
}
DHT.get(result);
```

WebRTC

Mechanism

Example

WebRTC – Introduction (1)

- **WebRTC for browser to browser communication**
  - P2P, no server involved (~mostly)
- **Google bought in 2010 GIPS and open sourced WebRTC**
- **Protocol standardized by IETF (codec requirements, media protocol), JavaScript API by W3C**
- **Supported by Chrome, Firefox (and others)**
- **Compatibility**
  - WebRTC support since Chrome 26+, Firefox 23+
  - SCTP: supported by Firefox, Chrome 31+
  - Binary data: supported by Firefox, Chrome 31+

WebRTC – Introduction (2)

- **Filling gap in the Web-Experience**

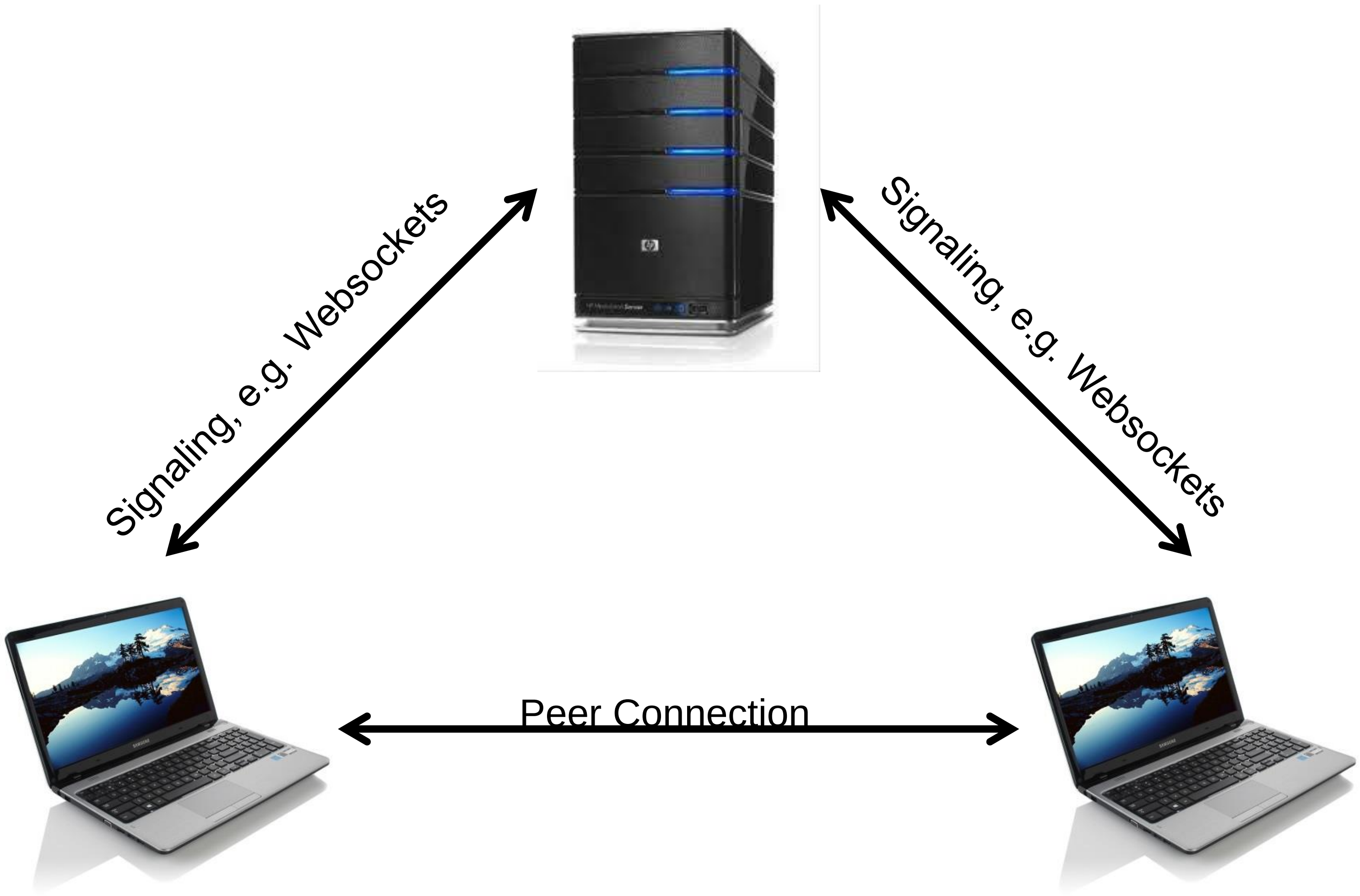
  - Video Chat → Google Hangouts Plugin, Flash, Java
  - Multimedia / Confernces → Expensive, proprietary 3rd party apps
  - Customer Service → Chat only 3rd party plugins/apps
  - Online Games → Flash
  - Real-time Feeds → Proprietary software
  - File Sharing → Requires Server / BitTorrent

- **WebRTC widely deployed, no client necessary!**

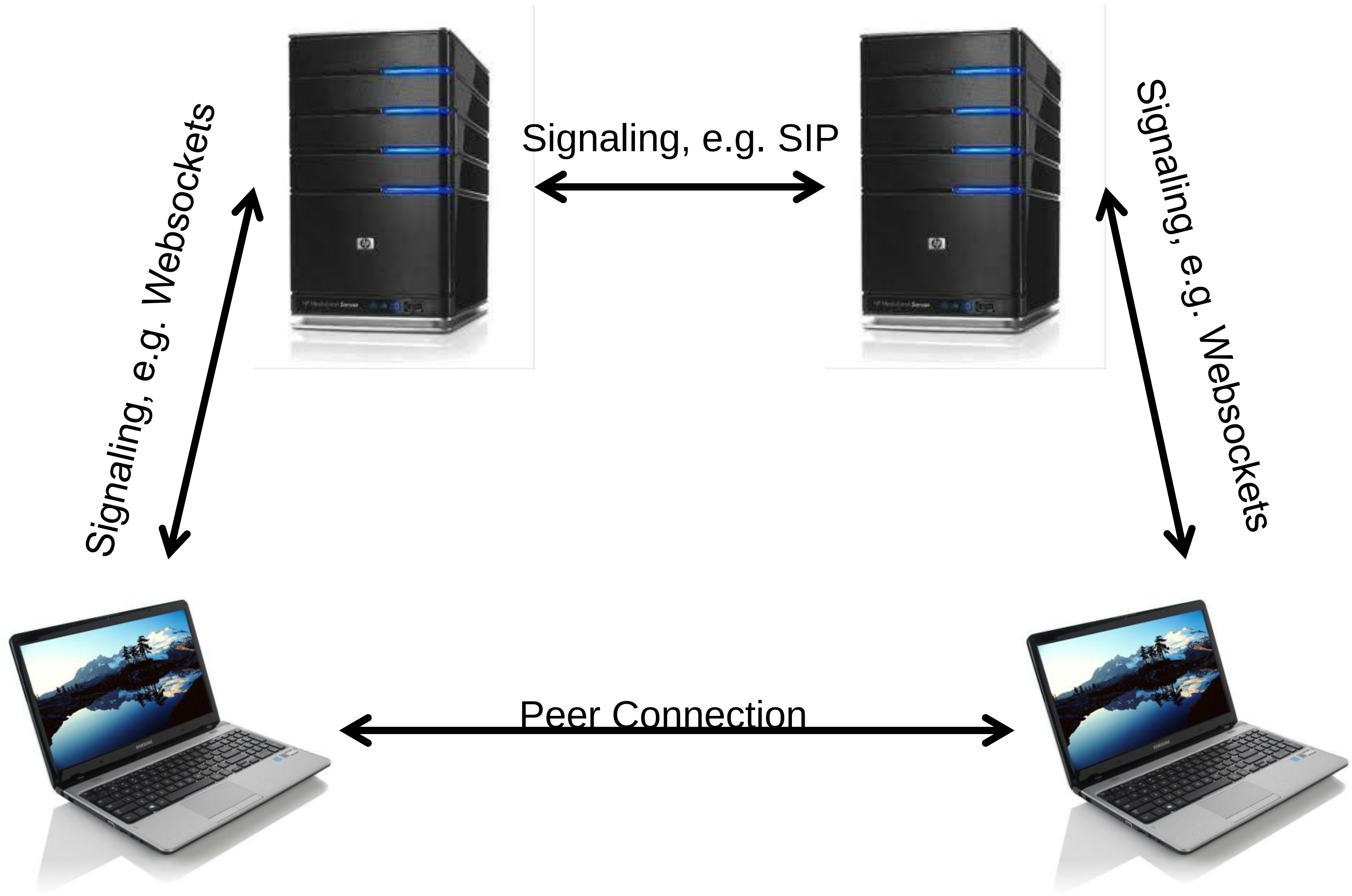
WebRTC – Introduction (2)

- **Developer does not need to care about NAT**
  - Abstraction, using STUN, ICE, TURN
  - STUN: session traversal utilities for NAT (detect which kind of NAT)
  - TURN: traversal using relays around NAT (relay)
  - ICE: interactive connectivity establishment , uses STUN and TURN
  - UPnP / NAT-PMP setup by the browser optional?
    - Bugzilla@Mozilla – Bug 860045
- **Once connection is established – easy API**
  - `sendChannel.send("hallo")`
  - `sendChannel.onmessage = function …`
- **Mandatory AES encryption**
  - **SRTP for Media, DTLS for Data, HTTPS for Signalling**

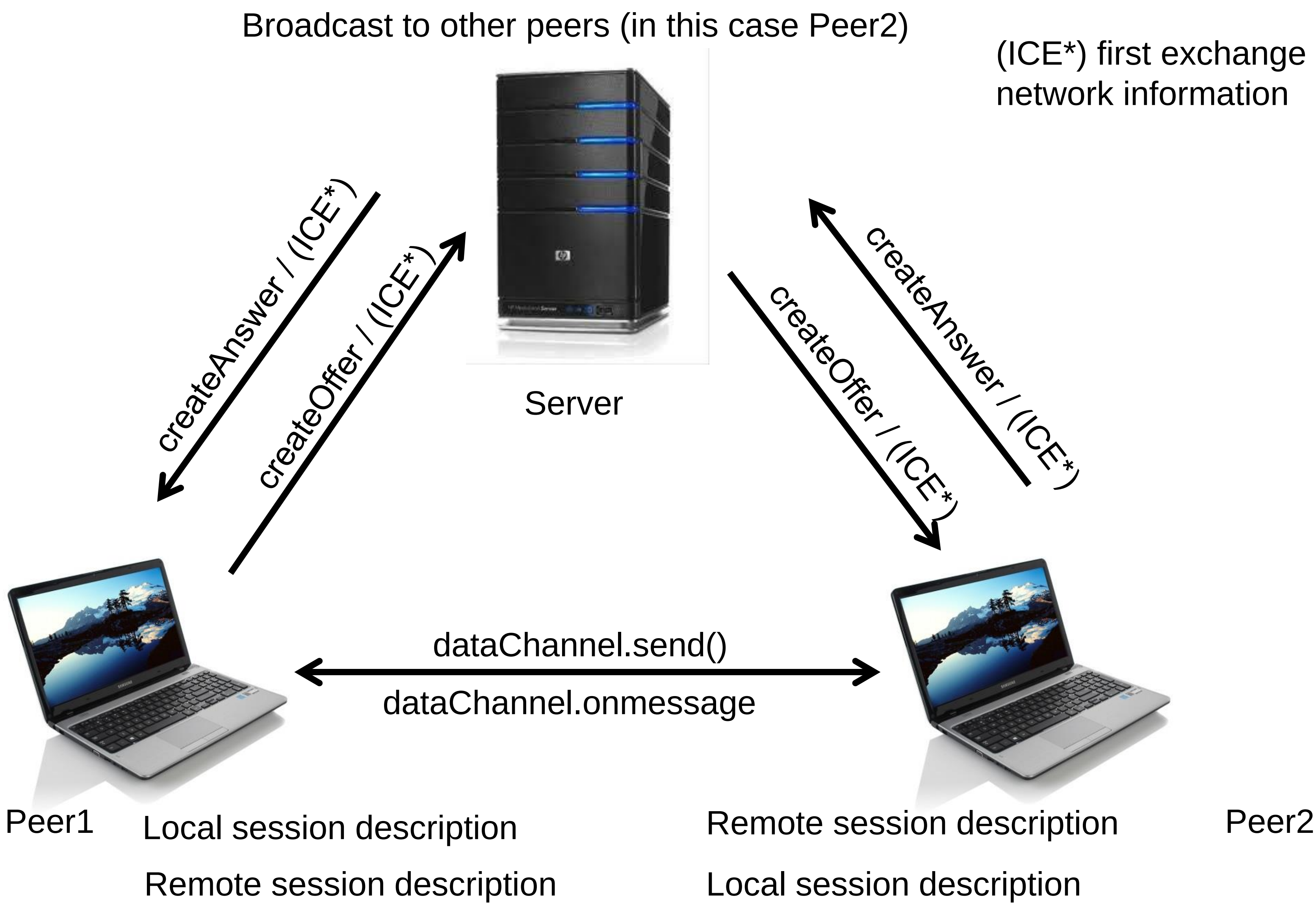
WebRTC Architecture - Triangle



WebRTC Architecture - Trapezoid



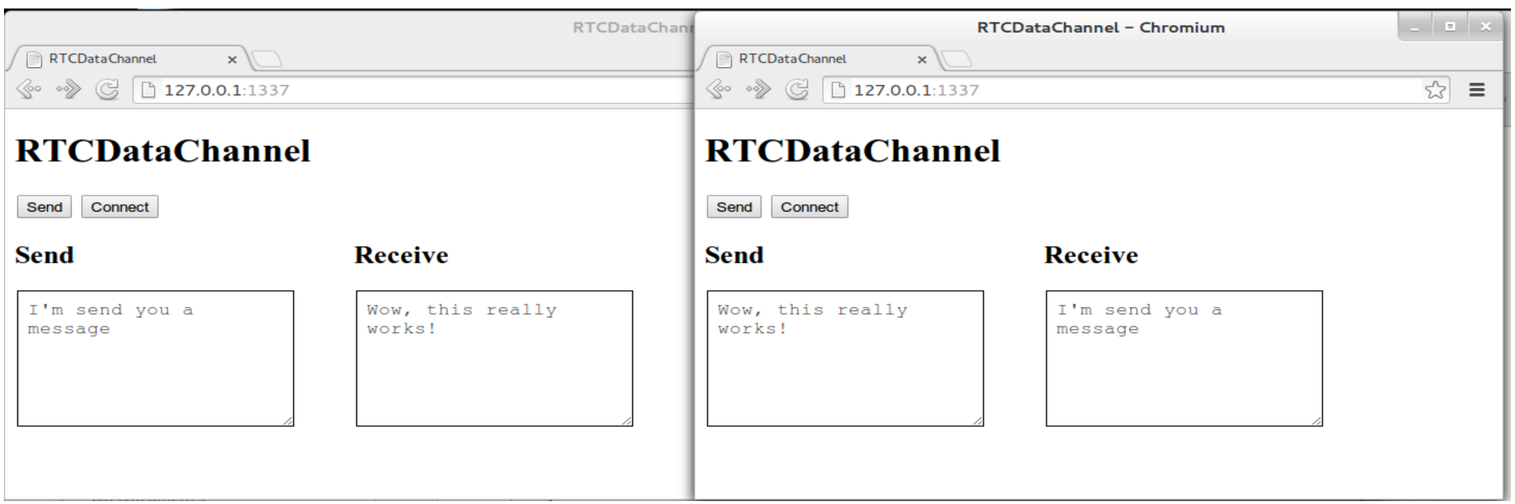
WebRTC Architecture - Demo



WebRTC - Demo

- **Server - tracker.js**
  - Node.js server
  - Broadcast messages
  - Seen as a “tracker”
- **Lets try a real setup!**

- **Client – index.html**
  - Client side JavaScript
  - One connection per peer
  - index.html also served by tracker.js



Example: https://www.webrtc-experiment.com/Pluginfree-Screen-Sharing/#CA134CZY-1FQD7VI

WebRTC - Outlook

- **Strong focus on VoIP [demo]**
  - Skype competitor?
  - Microsoft / IE and WebRTC?
  - SDP / signaling overhead if using with raw P2P data
- **Fewer plugins (flash, java), fewer registrations**
- **Mandatory Codecs? - VP8 and H.264 are MTI**
  - Video codec in JavaScript
- **Web-based P2P frameworks**
  - http://peerjs.com - make the API simpler
- **New types of applications – Conferencing, gaming, P2P file sharing**
  - PeerCDN, serve static content from browsers of other visitors

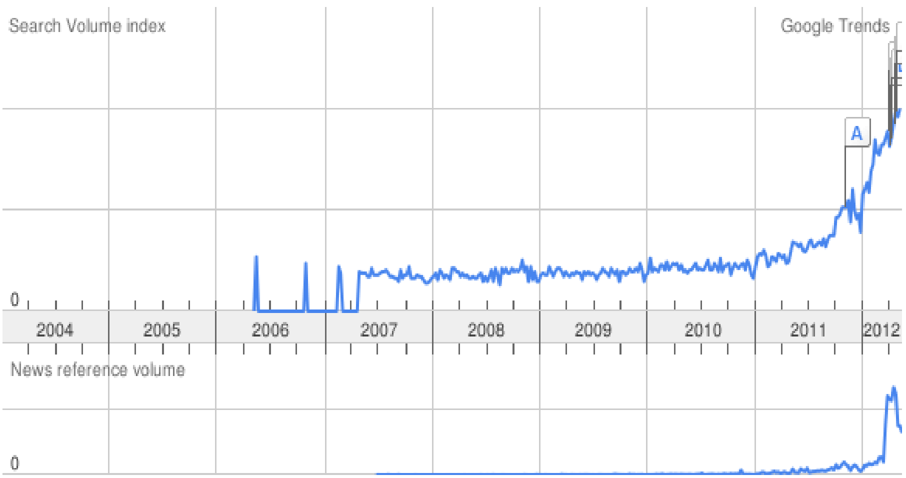
4. NoSQL

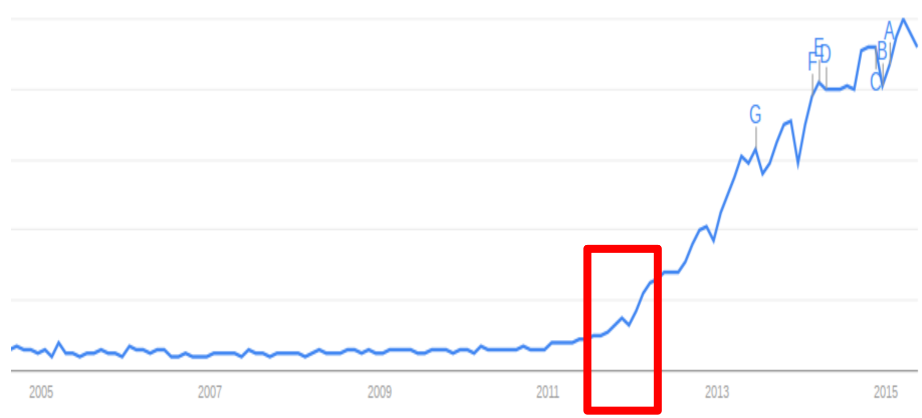
Introduction, Idea, Examples

Concept

Introduction

- **NoSQL: Not only SQL (http://nosql-database.org/)**
  - Non-relational, distributed, scalable (adding more nodes), fault-tolerant, availability
  - Used often for big data
  - No ACID, **NewSQL** ([Google Spanner]()) – Scalable like NoSQL, but with ACID
- **Google trend for "Big Data" 2012 vs 2015**





NoSQL Types

- **Key/Value Storage**
  - Cache
  - Store (consistent)
- **Document Storage**
  - Specialized key/value storage
- **Column Storage**
  - Key/value: rows -- column-family / columns
- **Graph Storage**
- **Others: Geospatial, Object (not much difference to document storage)**

ACID and Distributed Transactions

- **ACID: guarantee for database transactions**
  - Atomic: Everything in a transaction succeeds or is rolled back
  - Consistent: Before or after a transaction, the database is in a consistent state.
  - Isolated: Transactions cannot interfere with each other.
  - Durable: Completed transactions persist.
- **Transactions in distributed systems**
  - 2 phase commit
    - commit-request phase (or voting phase): prepare and report
    - commit phase: coordinator decides if commit or roll back
  - But: decreases availability (coordinator)
  - Paxos, Raft consensus protocol: quorum

Eventually consistent

- **BASE ↔ ACID**
  - Basically Available: Fast response even if replicas are slow or lost
  - Soft state: Cannot store permanent data, restarts in clean (in-memory)
  - Eventual consistency: Waiting long enough → get consistent data. May return inconsistent data in the meantime
- **BASE faster in distributed systems, but less guarantees**
  - E.g. web search vs. financial transactions
- **CAP Theorem by Eric Brewer (pick two)**
  - Consistency: Everyone sees the same data at the same time
  - Availability: All operations result in a response.
  - Tolerance to network partitions: Service will succeed, even if some components are unavailable.
- **At most two of them, focus on AP**

Key / Value

- **Types of consistency (Cassandra) write / read**
  - One – if one node saves the data, it is reported to the client
  - Quorum – if a majority reports to save the data, report it
  - All, if a nodes save the data, report it
  - TomP2P: voting schemes
- **Key / Value: collection of Key / Values**
  - Find fast content with given key / Big Data
    - Memcache (memory only), Redis, SimpleDB, …
- **Demo TomP2P: inconsistency join / leave**
  - DHT attack – countermeasures – voting schemes

Document Storage

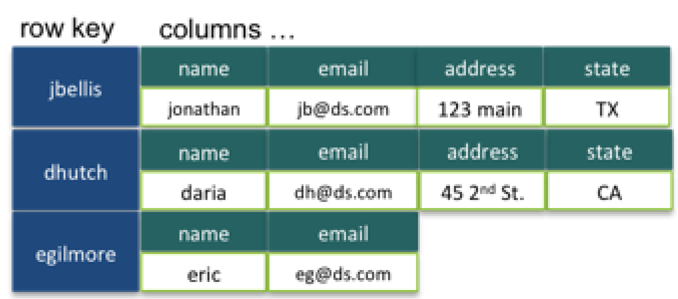
- **Similar to Key / Value, Value is document**
  - CouchDB, MongoDB
  - Access mostly through HTTP
  - Store data as JSON / BSON
- **Schema-less**
  - Types determined by the application

```
"name" : "Jim",
"number" : 052435345


"name" : "Bob",
"number" : 091454353,
"lectures" : "612 - Peer-to-Peer Systems and Applications"
```

Column Storage

- **Key associated with multiple attributes**
  - TomP2P similar (locationKey, domainKey, contentKey, value)
  - Can “emulate” Key / Value
  - HBase, Cassandra, Google BigTable
  - No constraints / foreign key
  - Denormalization (avoid in RDBMS)
- **How to make TomP2P a column storage**
  - Split locationKey or domainKey or contentKey (64bit / 96bit)
  - Keyspace, column family objects,
    row key, columns (name, value)

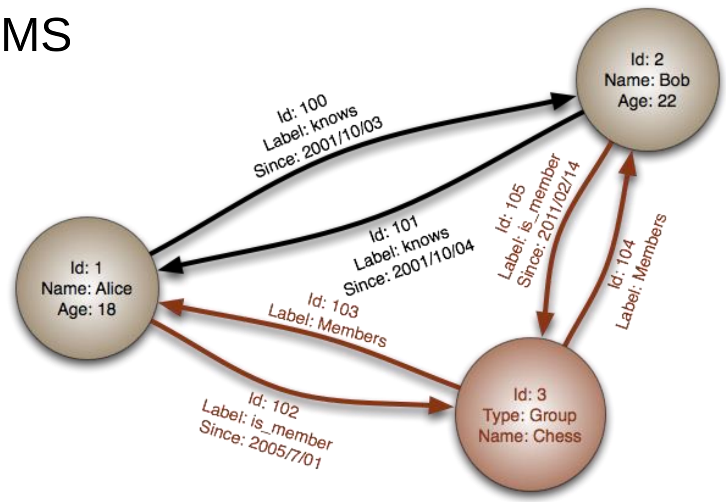


- **Demo TomP2P: multiple columns**

http://www.datastax.com/docs/1.0/ddl/column_family

Graph Databases

- **Focus on structure of data**
  - Nodes (similar to objects in OO languages), edges, and properties
  - Neo4j, InfiniteGraph, and more
  - From http://neo4j.org: “For many applications, Neo4j offers performance improvements on the order of 1000x or more compared to relational DBs.”
    - Can be more efficient than joins in RDBMS
- **Graph database with TomP2P?**
  - Friend recommendation system with graphs



http://en.wikipedia.org/wiki/Graph_database

Other Types / Conclusions

- **Geospatial databases**
  - Support: MongoDB, BigTable, Cassandra, CouchDB
  - MongoDB: supports two-dimensional geospatial indexes
    - Find closest N items (e.g. airports) to location X
  - Map this to a DHT – SpatialP2P

    Storing and Indexing Spatial Data in P2P Systems; V. Kantere, S. Skiadopoulos, T. Sellis, IEEE Transactions on Knowledge and Data Engineering
- **Conclusions: NoSQL vs. SQL**
  - Choose the right tool for the job!
  - high-volume, high-availability use case
  - There is no free lunch, SQL is very powerful
  - SQL on top of NoSQL → is it possible? How well does it work?
- Query Language! **UnQL (dead?) - Demo TomP2P (subset)**

5. Tor

Tor

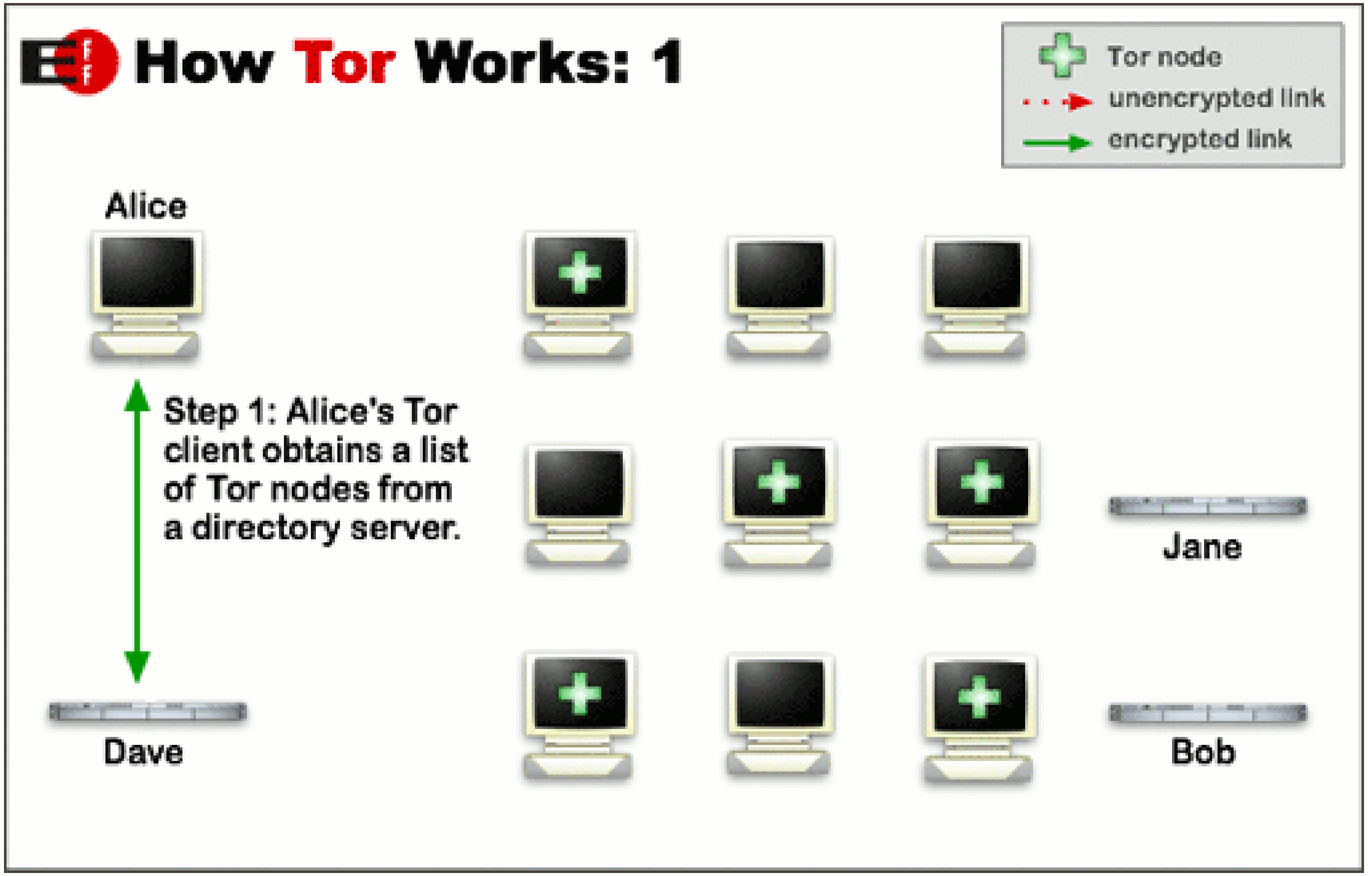
- The Onion Router
- **Started ~1995 by U.S. Naval Research Laboratory**
  - protect US intelligence communication
  - Further development by DARPA, 2004 free license
  - EFF funding development
- **Anonymous internet communication system**
  - SSL / TLS is not enough
  - protect privacy of users
- **Encrypts all messages (also header, IP)**
- **Sends data through virtual circuit (3 random relays)**
  - Use SOCKS proxy to connect via Tor
  - Example Tor Browser accessing http://tomp2p.net

Tor

- **Hidden services**:
  - Servers configured to receive connections only through Tor
  - Silk road was using hidden services + Bitcoins → bad press
- **Main use-case: journalists, whistleblowers, and dissidents**
  - Can be used against price discrimination
- **Attention**
  - Tor exit node can see traffic! – Use SSL/TLS
  - Services block Tor exit nodes, e.g., example: Wikipedia
  - Tor exit node operator facing copyright claims - template
- **Large project: 152’000 LoC – TCP only**

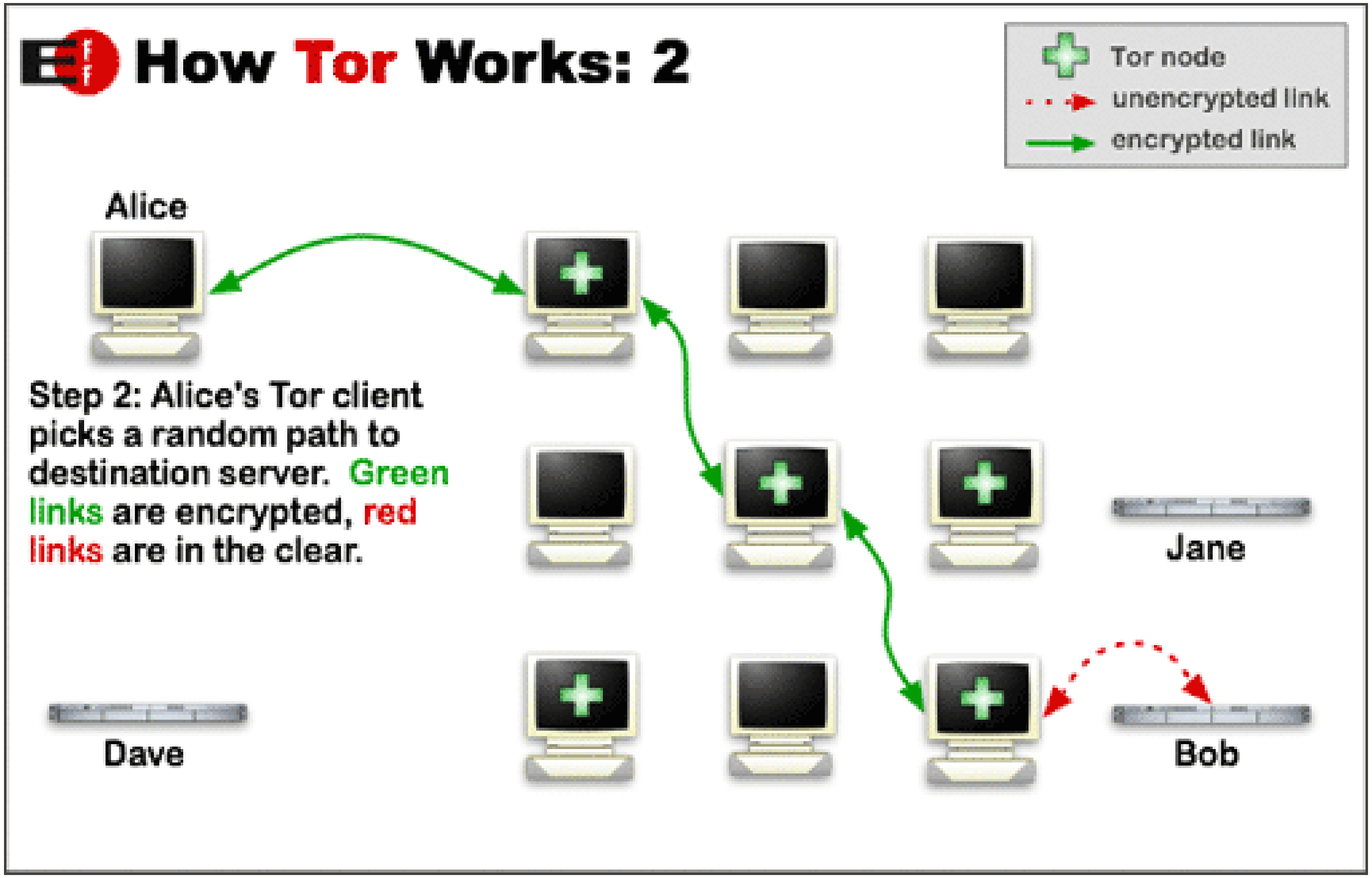
Tor

- How it works



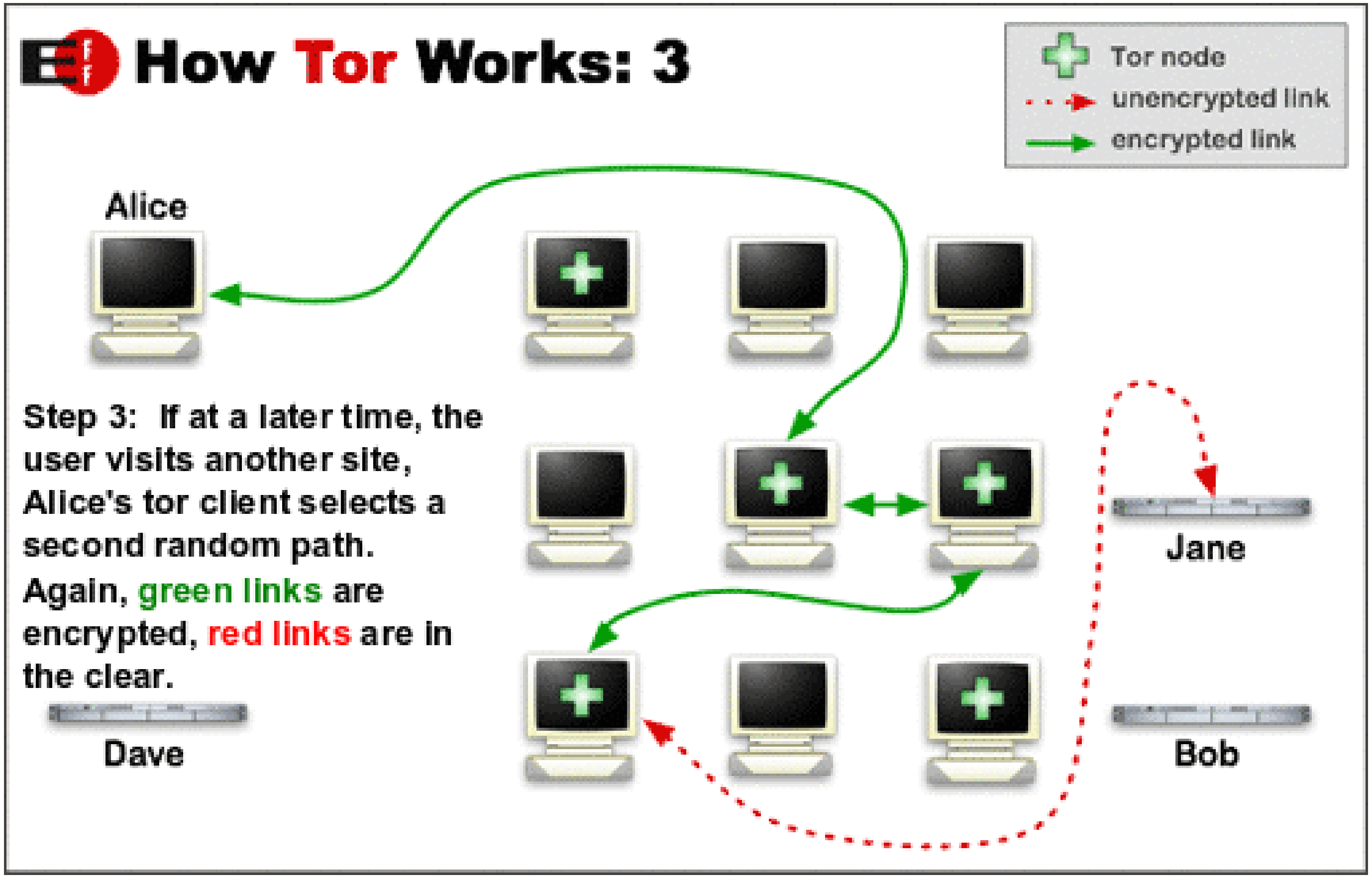
Tor

- Alice to Bob



Tor

- Alice to Jane



Tor

- **Be smart: Don't provide your name or other revealing information in web forms**
- **Bomb threats at Harvard**
  - FBI figured out, Tor was used – check who was using Tor in the Harvard network → 2 days later student was caught.
- **Language may be different**