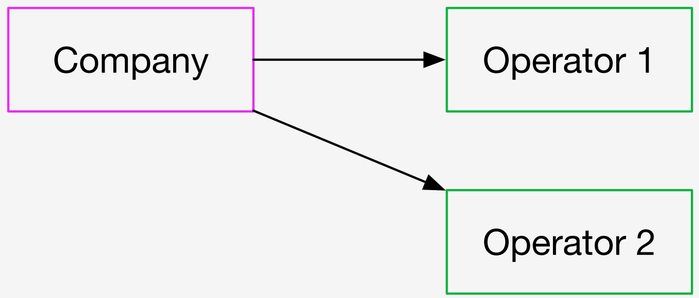
Leaderless Replication (and Balance Management) for Unordered SMS Messages

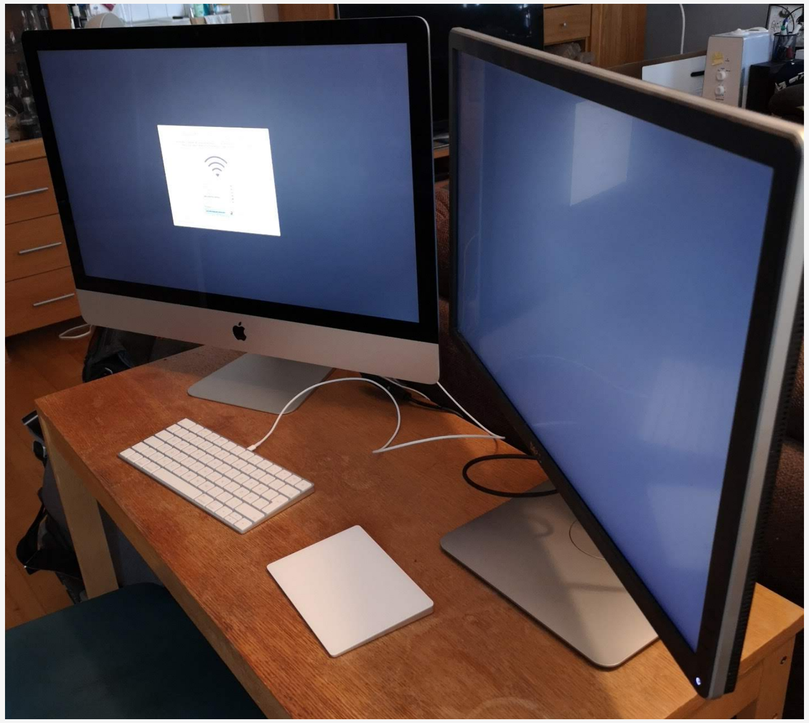
Daniel Brahneborg
Infoflex Connect AB
Mälardalens Högskola, Västerås, Sweden

@basic70, brahneborg@infoflexconnect.se

SMS / Mobile Text Messages

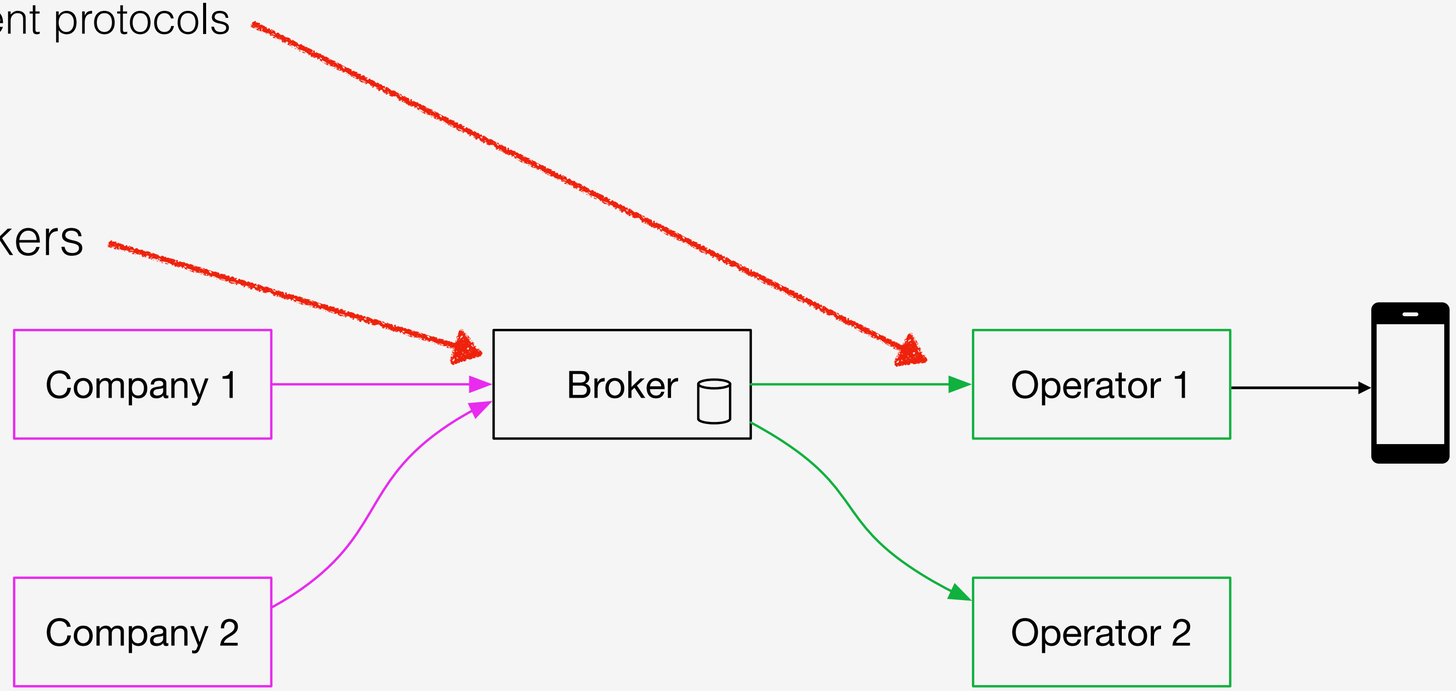
- Business-to-consumer communication
  - Two Factor Authentication (new computer etc)
  - Meeting reminders
- Version 1:





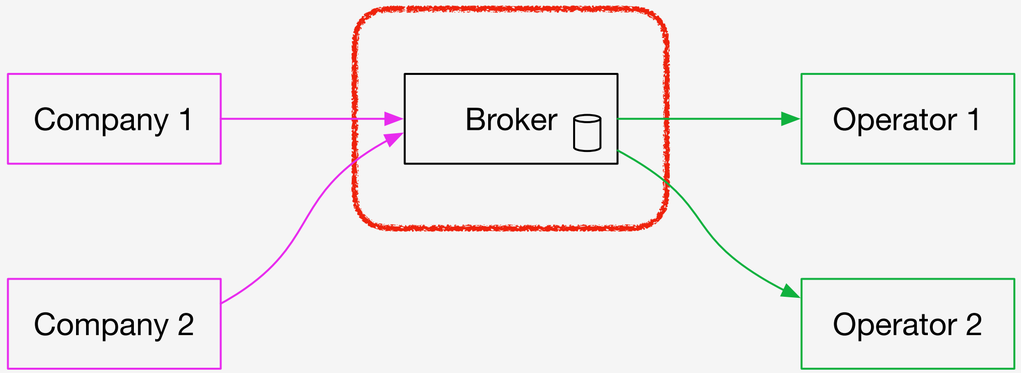
SMS, version 2

- Companies do not want to communicate directly with operators
  - Operators use different protocols
  - Too many operators
- Version 2: SMS Brokers

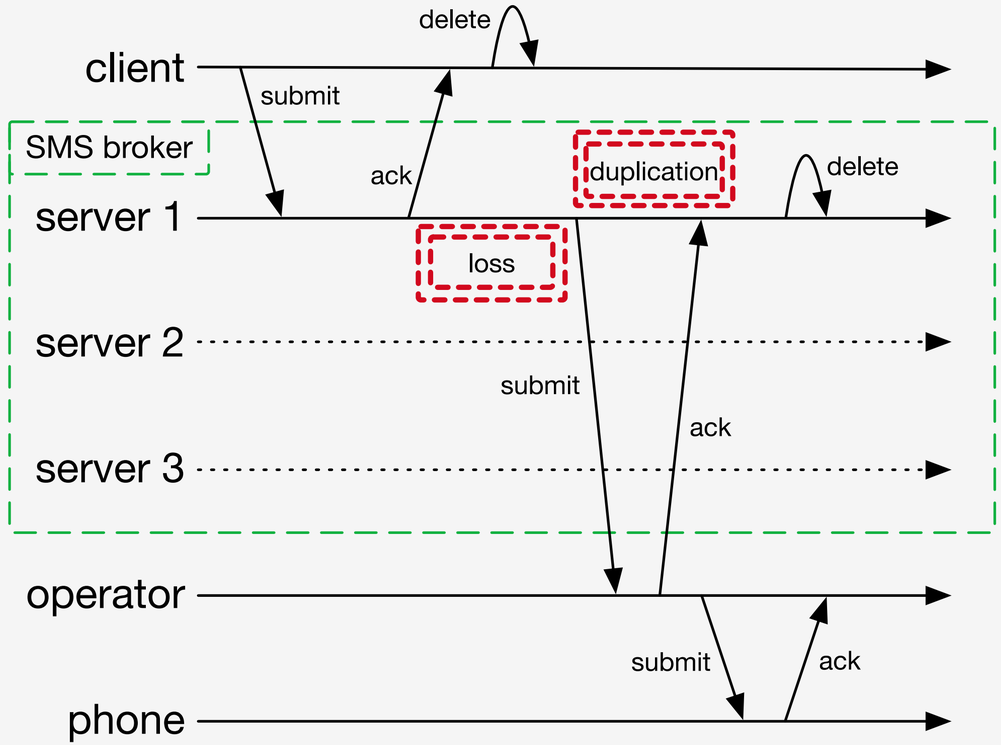


SMS Broker software = SMS Gateway

- Protocol conversion
- Routing
- Character set conversion
- My day job: Enterprise Messaging Gateway (EMG)



Problem: possible message loss



Obvious solutions

- Just store the messages in a replicated database?
- Just use Apache Kafka?

Requirements making this harder

- Between multiple Internet Providers
  - Minimal network traffic and round-trips
    - Carsten Binning: ”network communication is evil, must be avoided at all costs"
  - No master serialising server
- At least 1000 messages per second per node
- Can not change protocols towards clients or operators
- Deployed by customers, so preferably not JVM-based

Freedoms making this easier

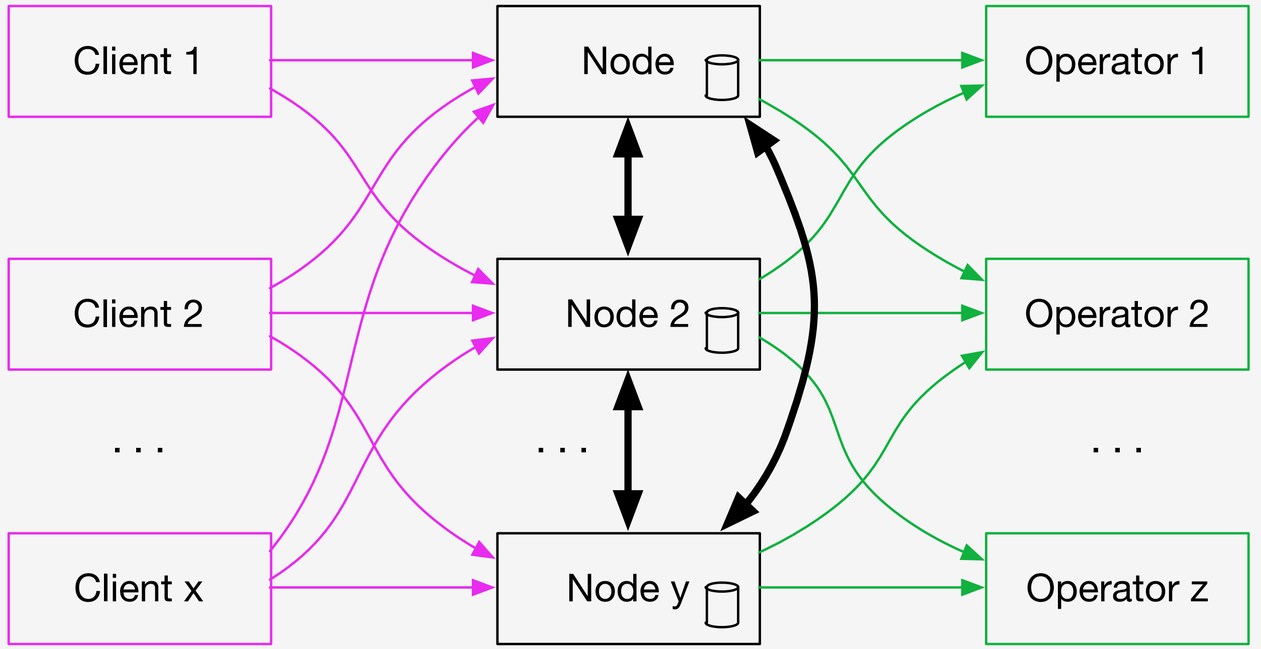
- We do not need “exactly once” delivery
  - 1 plus epsilon
- Messages have no relative order

Good or bad: Short lifetime

- Typically less than a second
- Never more than 72 hours

Target Architecture, version 3

- Multiple gateway nodes
- Replication between the nodes
- **So what are the black arrows?**

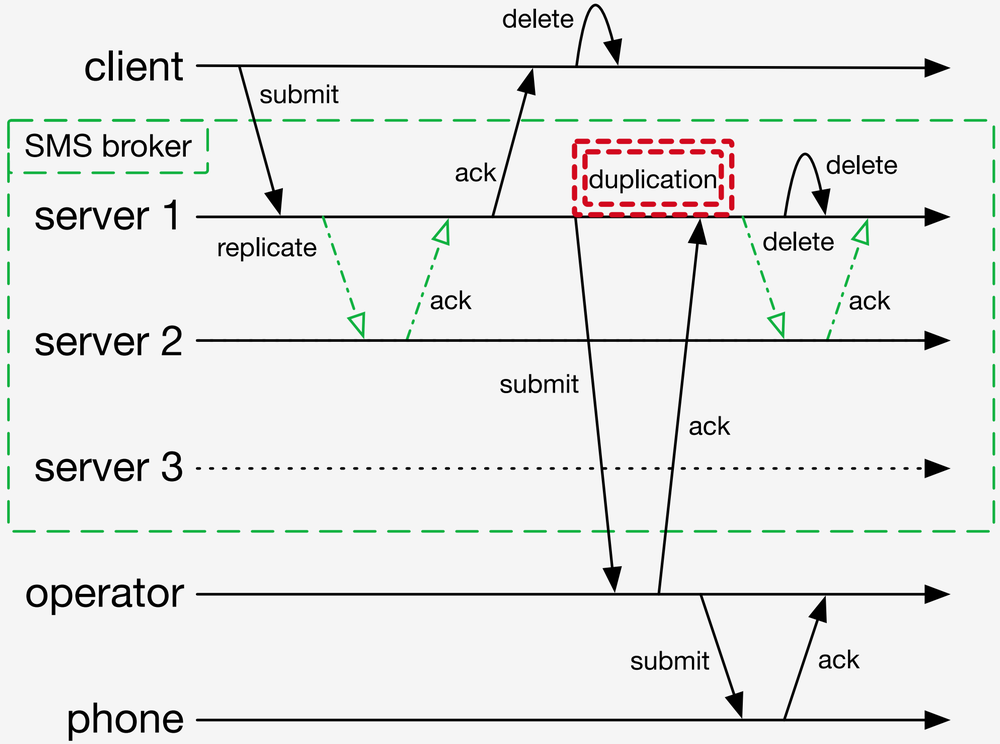


Alternatives

- SQL Database: MySQL/MariaDB already installed
  - Many and strong guarantees, making it too slow
- NoSQL Database
  - MongoDB too slow
- Event Queue: Apache Kafka, Spread
  - Model “everything to everybody” is a bad fit
- Replicated State Machine: Paxos, Raft
  - Too slow over WAN

Replicate with failover alternatives

- Replication contents
  - Message id
  - Message contents (recipient, body, etc)
  - “Owned by Server 1”
  - “First fallback is Server 2”



Proposal: GeoQueue library

- Small API: save(), delete(), adopt()
- Rare: Replicate to f, not n-1
- New: Include ordered list of failover alternatives

New: leaderless failover strategy

- When a message is received: pick f other random nodes
- Replicate message together with this list
  - Only to these f nodes
- When a node fails, message ownership goes to the next alive node
  - Can be done in parallel on all nodes
- If the new owner is me, trigger application callback adopt(message)
- If no remaining node, terminate the message

Method, for the rest of 2019

- Structured experiment, to evaluate replication methods for an unordered queue
  - SQL (MariaDB), NoSQL (Redis?), Event Queue (Spread), Bespoke (GeoQueue)
  - Spread: same failover logic as GeoQueue
  - Including verification of failover

Licentiate thesis

- Including an article on round-trip anomaly detection
- Some thoughts on sufficient guarantees versus necessary ones
- Looking for committee members

Future up to PhD, end of 2021

- Balance counters
  - Based on CRDT PN-counters
- Replicated message status database
- Various: fuzz testing EMG, replication over QUIC, etc