Verteilte Systeme SS2002:

Peer-to-Peer-Systeme

Marcel Waldvogel

Übersicht

- Definition
- Geschichte
- Beispiele
- Technologie
- Ausblick

Definition

- Bislang keine formale Definition
- Name
  - Kommunikation unter gleichberechtigten Partnern
- Verbindungsaufbau
  - Client-Server, besser: Initiator-Target
- Test
  - Könnte die Kommunikation bzw. Dienstnutzung auch umgekehrt erfolgen?

Geschichte

- Telefonie
  - Client-Server zu Zentrale
  - Peer-to-Peer zum Endbenutzer
- Internet
  - Jeder Rechner kann jeden Dienst anbieten
    - Mail, DNS
- WWW: Verlinkung
- Erste Generation
  - Eternity 1996, Napster 1999, Gnutella 2000
- Zweite Generation 2000/2001

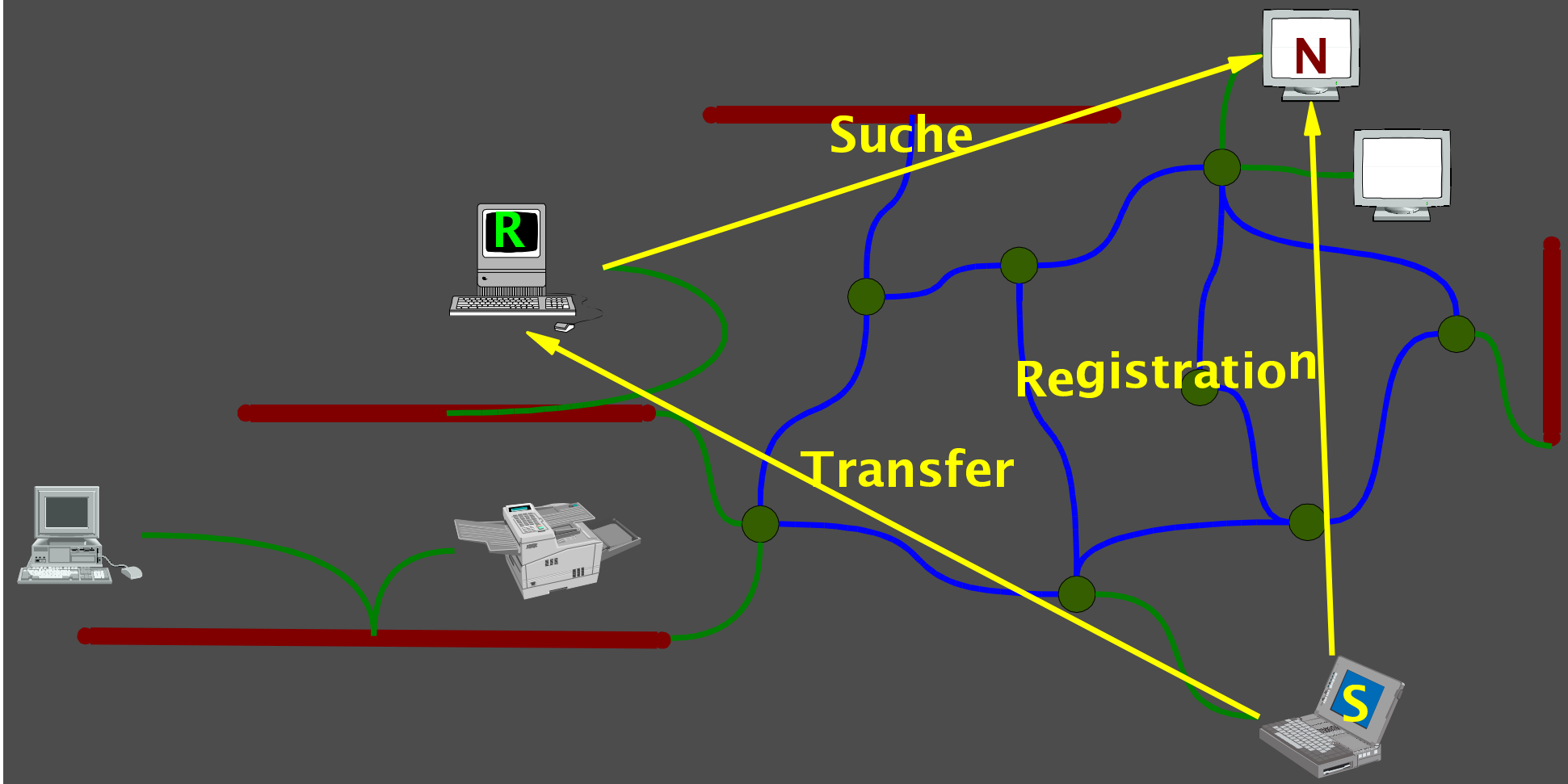
P2P-Applikationen und -Protokolle

- (Ver-)Teilen
  - Napster, Gnutella
- Konservation
  - Eternity/FreeNet, Publius
- Privatsphäre
  - Mixmaster (Onion Routing), Crowds
- Verteilte Hashtabelle
  - CAN, Pastry

Napster

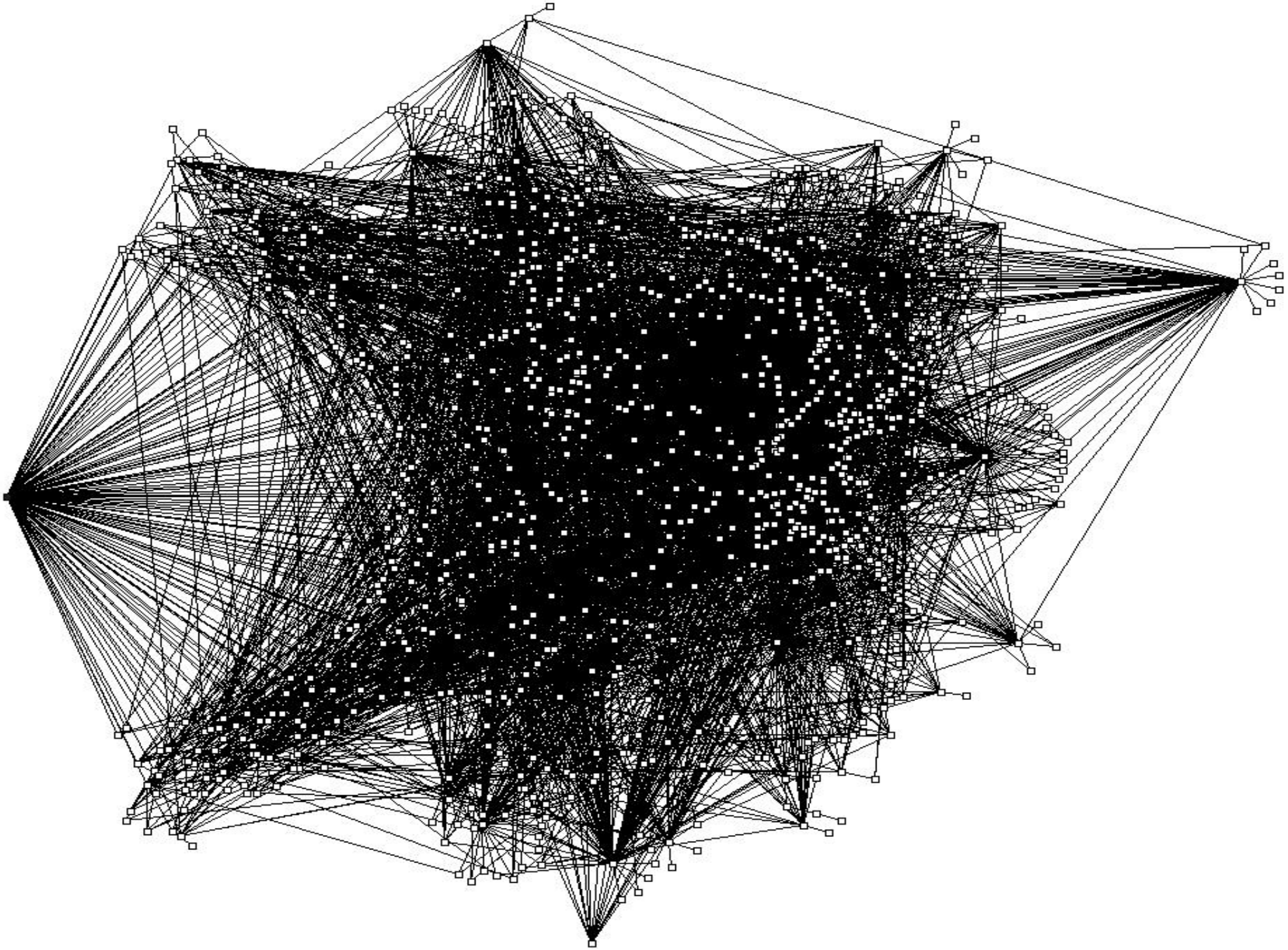
- Klassisches Beispiel
  - Legalität
  - Bandbreite
- Technologie
  - Zentraler Index-Server

Napster-Kommunikation



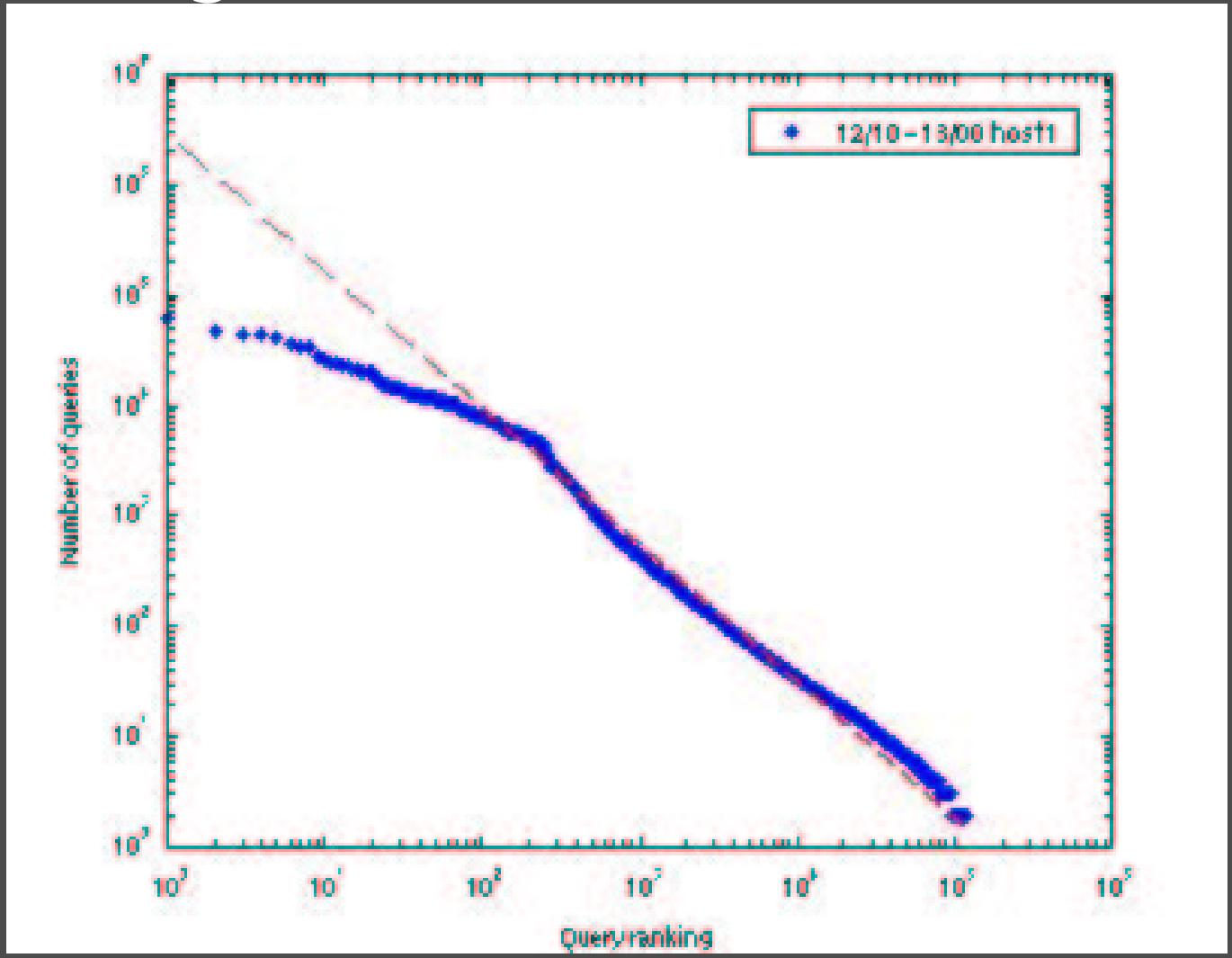
Gnutella

- Ohne zentralen Server
- Technologie
  - Topologie: Durchhangeln
    - Der Nachbar meines Nachbarn ist auch meiner
  - Anfrage: Fluten
    - Horizont
- Probleme
  - Stabilität
  - Skalierbarkeit



Anfragen

- Zipf-Verteilung
  - $1/i^a$



Eternity/FreeNet

- Idee
  - Beliebter Inhalt bleibt im Netz
  - (Kein) Vertrauen, dezentral
- Technologie
  - Keine Registrierung
  - Fixe Nachbarn
  - Iterative Suche
  - Iterative Rückgabe mit Kopie
  - Adressierung durch Inhalt

Mixmaster

- Idee
  - Anonymität
  - Antwort möglich
- Technologie
  - Onion Routing: Schichten aus Zwiebelschalen, die jede nur von einem bestimmten Server entfernt werden können
  - E-Mail

Probleme

- *Skalierbarkeit*
- *Effizienz*
- *Stabilität*
- *30% des Internetverkehrs ist P2P*

Inhaltsadressierbare Netze

- Idee
  - Gezielte Suche
  - Eindeutige ID eines Dokuments
- Technologie
  - Organisierte, überschaubare Topologien
    - Vor-/Nachteile?
    - Effiziente Adressierung

CAN

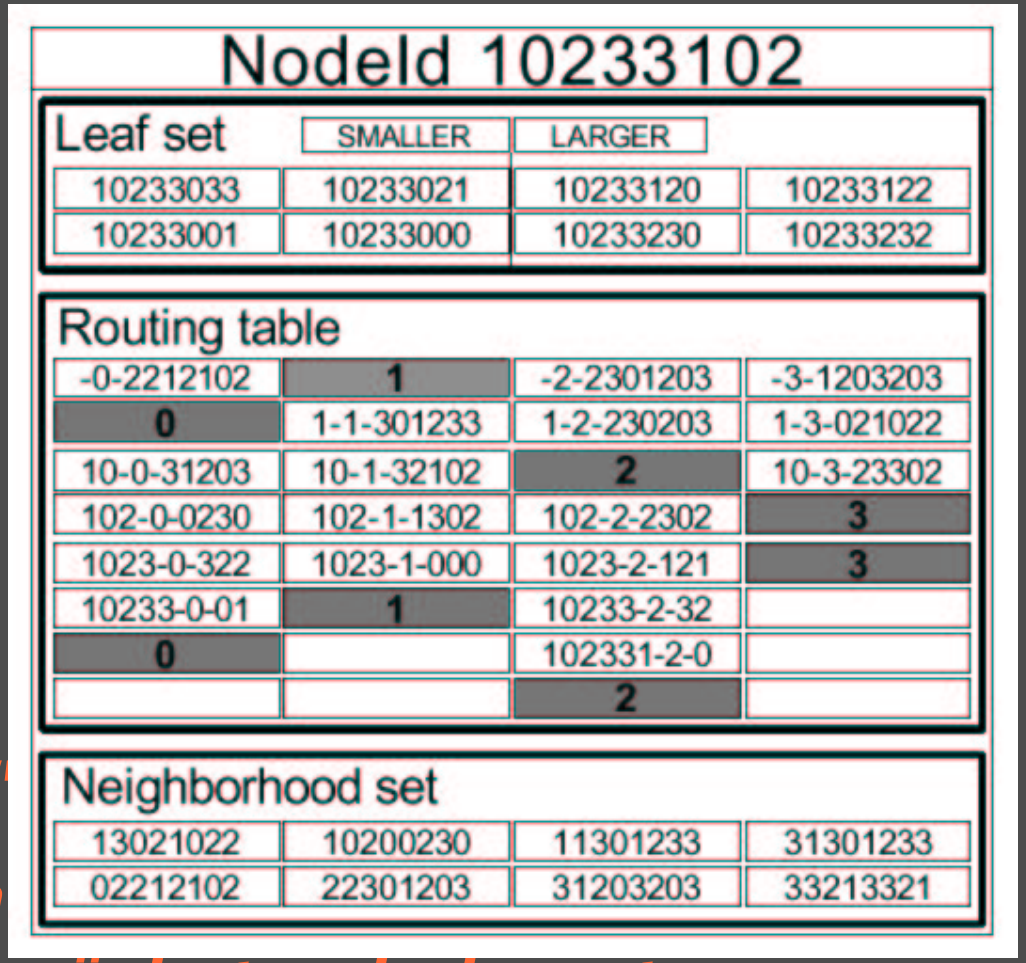
- Content-Adressable Network
- Idee
  - Rechteck-Tiling
- Technologie
  - Zufälliger Ort
  - Aufteilen des bestehenden Rechtecks

Pastry

- Nodes und Dokumente
  - "Zufällige" ID
  - Dokument beim Node mit nächster ID
- Baumsuche
  - "Baum mit vielen Wurzeln"

Pastry: Routing

- Nachbarn
  - Logisch (ID)
  - Lokal
- "Baum"
  - k breiter Pfad von "eigener Wurzel" zu "eigenem Blatt"
  - Weiterleitung zum jeweils der Ziel-ID nächsten bekannten Knoten



Nodeld 10233102

Leaf set

SMALLER		LARGER	
10233033	10233021	10233120	10233122
10233001	10233000	10233230	10233232

Routing table

-0-2212102	1	-2-2301203	-3-1203203
0	1-1-301233	1-2-230203	1-3-021022
10-0-31203	10-1-32102	2	10-3-23302
102-0-0230	102-1-1302	102-2-2302	3
1023-0-322	1023-1-000	1023-2-121	3
10233-0-01	1	10233-2-32	
0		102331-2-0	
		2	

Neighborhood set

13021022	10200230	11301233	31301233
02212102	22301203	31203203	33213321

Ausblick

- Eierlegende Wollmilchsau
  - Effizient adressierbare P2P-Systeme
  - Vertrauen und Vertrauensmasse
  - Caching
  - Suchmöglichkeiten
- Vergleich zu Client-Server
  - Vor-/Nachteile?

Beispiel: PGP-Keyserver

- *Globales Netz von Keyservern*
- *Synchronisation der Datenbanken*
- *Vertrauen*
- *Zuverlässigkeit*

Current Keyserver Infrastructure

- Single server per site
  - Monolithic
  - Integrated database, no RDBMS
  - Independent administration and policies
  - Replicated database with multi-masters
  - Sites may be down or disconnected
- Synchronization
  - Updates to be propagated to all sites - reliably
  - Add mostly nature of updates

Synchronization mechanisms

- Synchronization by e-mail
  - Unreliable
  - Overloads and overheads
  - Redundancy
  - Pksd, OKS
- LDAP
  - No state info kept - Inconsistency
  - Redundancy, Latency
  - Pgpcertd
- Synchronous and Asynchronous update tradeoff

Current Problems

- Catching up after a "down time"
- Keyservers Load
  - Mail queue
  - Processing overhead
  - Manual intervention - bounces
- Redundant messages for higher convergence
  - Flooding
  - Looping
- Manual dumping and feeding
- No means for state verification and correction

Goals

- Replicated Add mostly database synchronization
- Global consistency
- Efficient Transport
- No Redundancy and No Latency
- Scalability
- Outdated server -means to catch up

Heart beat Protocol

- Serial Numbers
  - Globally unique identifiers
- Periodic multicast of state information
- Waldvogel, Weiler, Baumer - ETHZ
- Modification of Marc's Code
- Problems
  - Heart beat takeover
  - Network partitioning
  - No RDBMS
  - Reliability

EKA

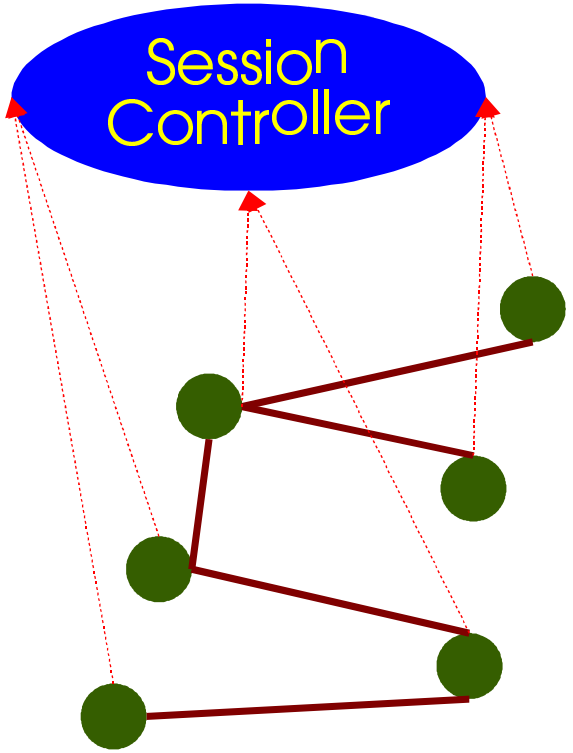
- Serial numbers
  - Globally unique identifiers (IP, local sequence)
  - Automatic and immediate loss detection
- No heart beats
- Instead Reliable multicast transport
- Add mostly nature of updates
  - Latency of asynchronous, consistency of synchronous
- Uses RDBMS - Oracle 8i
- Written in JAVA (from scratch)

Transport Evaluation

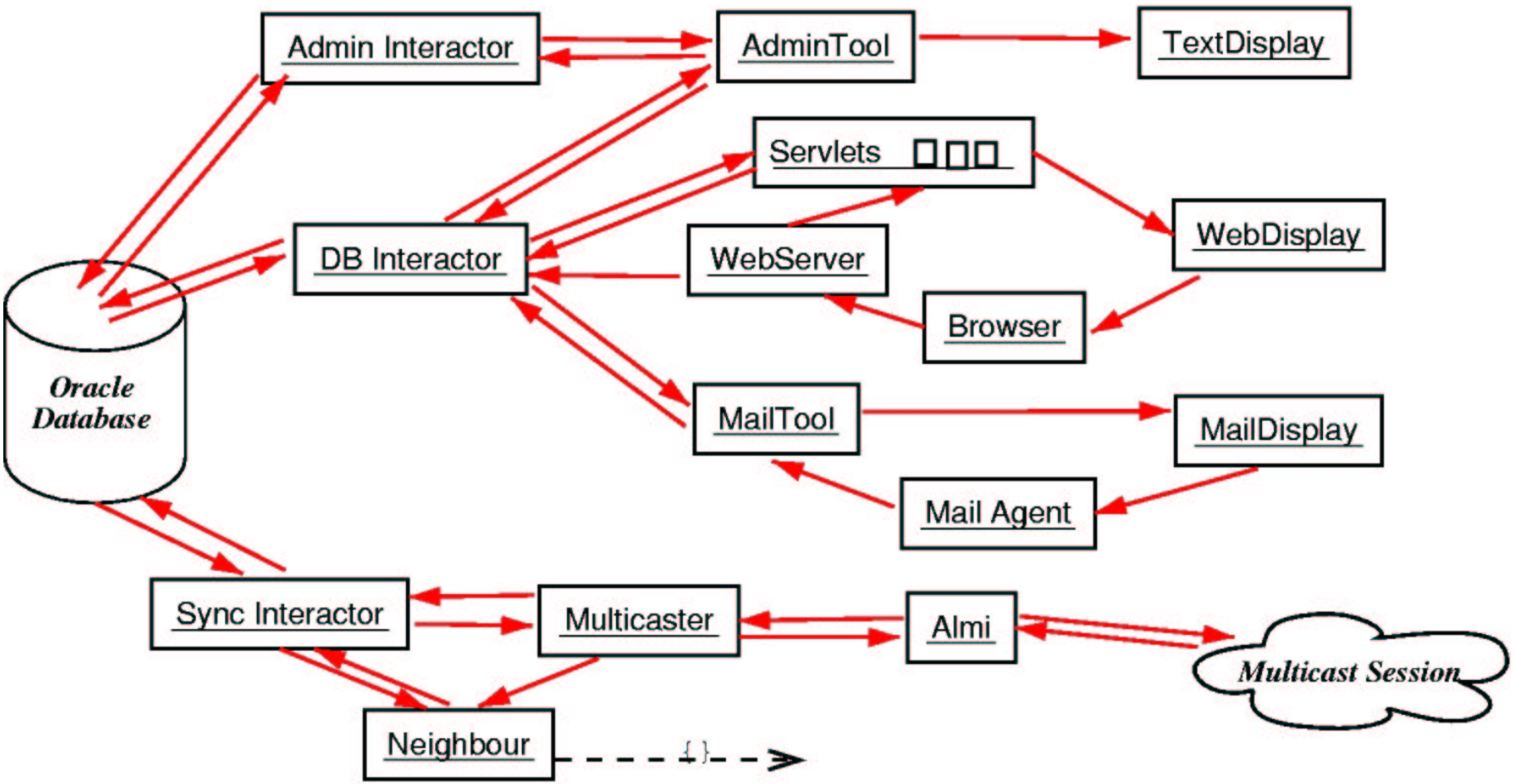
- Mbone
  - Limited access
  - Scalability issues (number of groups)
  - High data loss rate
  - Complex loss recovery
  - Congestion control
- ALMI
  - Dynamic Reconfiguration
  - Scalability issues (frequency of changes)
  - Currently centralized change management

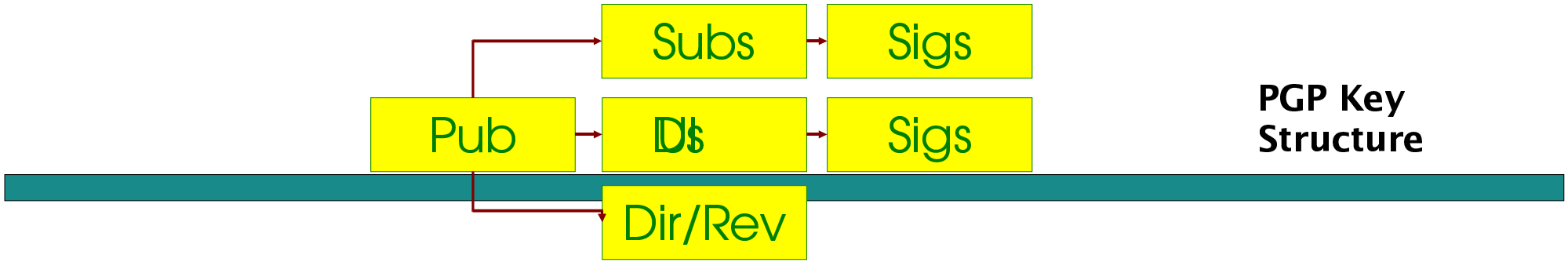
ALMI

- Session controller (SC)
  - Greeting point
  - Spanning tree topology
- Participants
  - Register with SC
  - Probe neighbors as told by SC
  - Report status to SC
- Middleware
  - Sherlia Shi

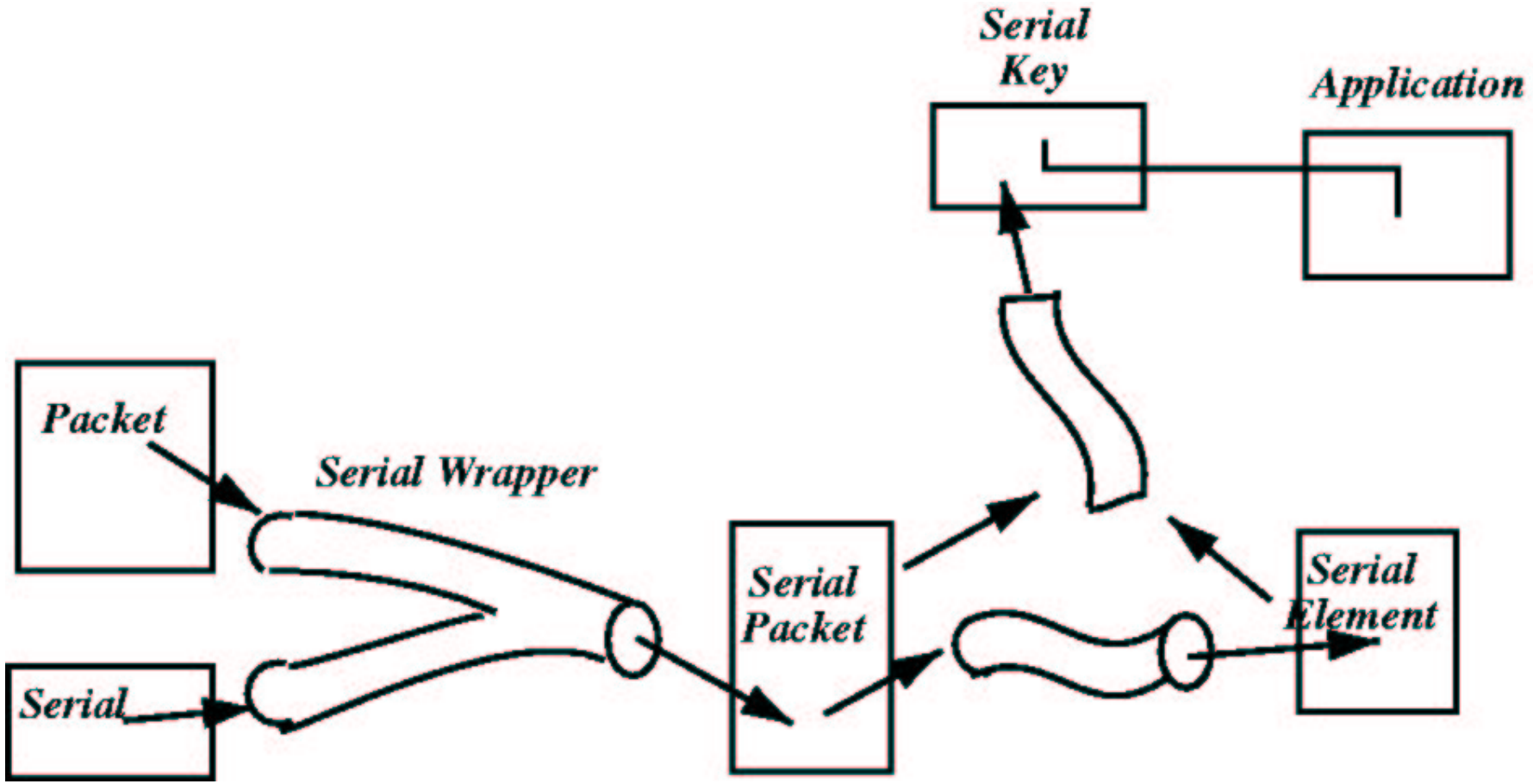


Architecture

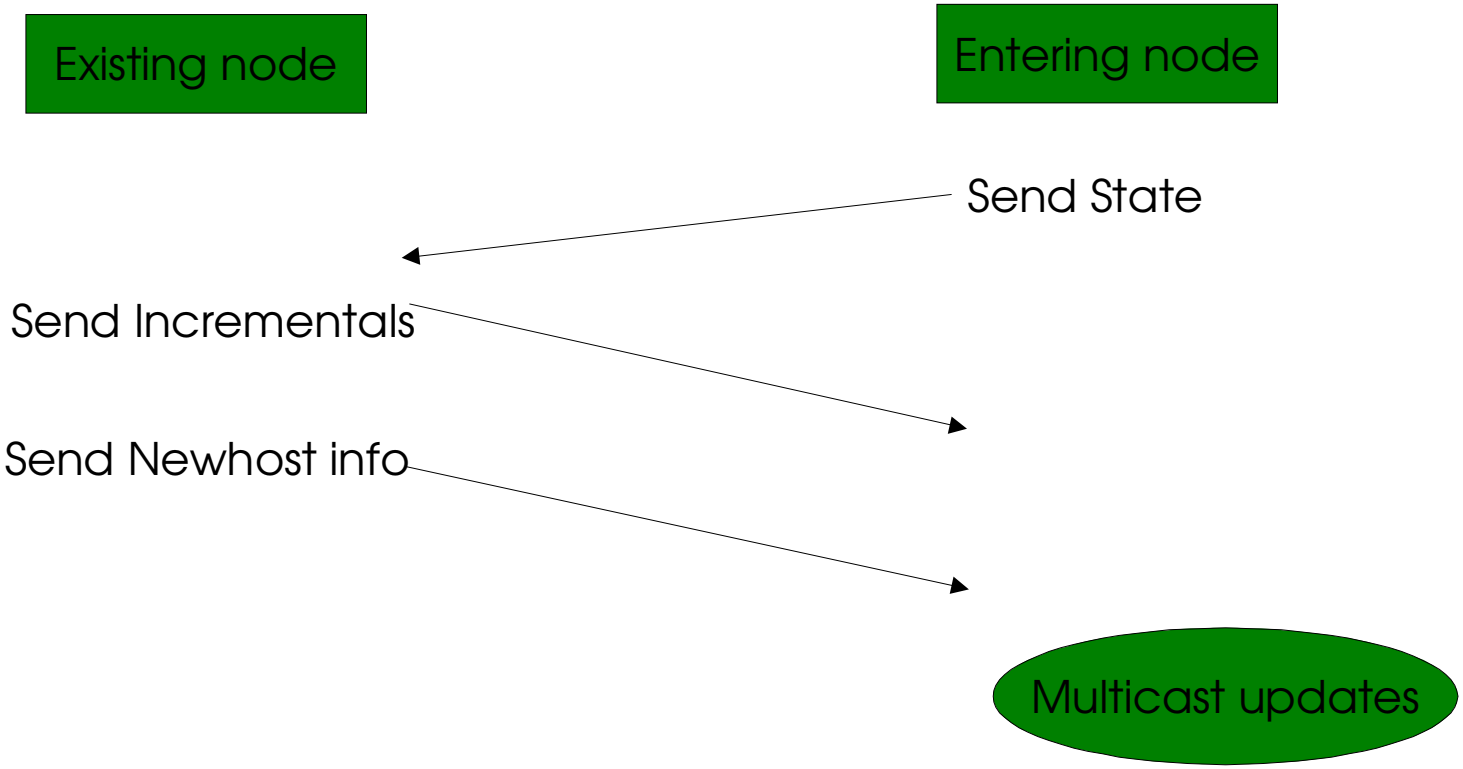




PGP Key Structure



OOB synchronization



Packet Format:

Type	Serial	Data

Ensuring Synchronization

- Short term Synchronization - ALMI
  - Reliable Transport - TCP
  - Recover from application - dynamic tree reconfiguration
- Long term Synchronization
  - Globally unique serial numbers
  - find lost updates during the down time

Recovery

- Almi recovery
  - Call back to application, query database
- Conflict Resolution
  - Keep the packet with smallest serial number
- Deletion
  - Mark as deleted