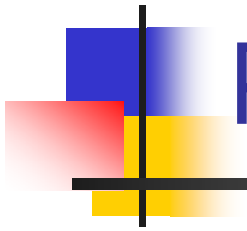


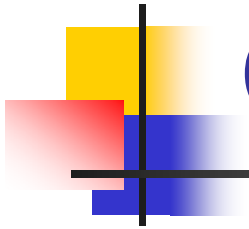
GALANX: An Efficient Peer-to-Peer Search Engine System

Yuan Wang David J. DeWitt

Leonidas Galanis

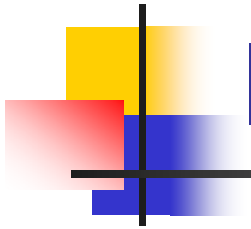
Proseminar
Peer-to-Peer Information Systems

Tutor: Christian Zimmer
Speaker: Steffen Metzger



Overview

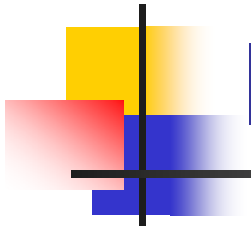
- Introduction: What is GALANX
- The Key Idea
- State of the Art
- A first DHT-Solution
- GALANX in Detail
- Implementation & Tests
- Conclusions & Future Work



Introduction: What is GALANX

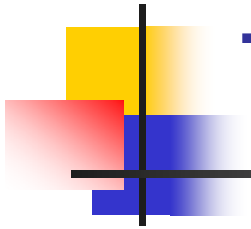
- Looking at large data networks (e.g. “the Web”, corporate LANs or perhaps file-sharing networks) we have a lot of data providers and even more clients searching for specific data

- How to route search queries efficient?
  - Google: centralised, fast lookups, not scalable, expensive, vulnerable, not up to date, not covering the whole web



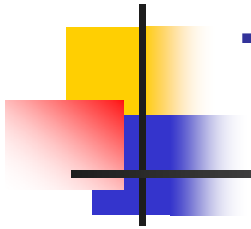
Introduction: What is GALANX

- **Scanning** the whole web **instantly** with a **centralized** approach (Google-like) seems to be **impossible** looking at the cost
- Peer-to-Peer looks like a good option, but several open questions how to do this right
- **GALANX focuses on** the core of **how to route a query fast and efficient** to the appropriate data provider



The Key Idea

- **Each data provider is a node** in the Peer-to-Peer network
- **Each node** publishes an **Index** which holds information about the data it shares



The Key Idea - Properties

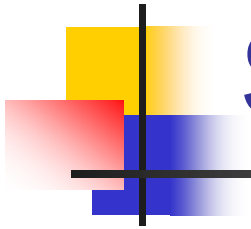
- **The system is adaptable to complicated queries** (e.g. full text search)
- **Ownership of data is respected** (no data caching or moving between nodes)
- **Every node has as much freedom as possible**

The Key Idea – Simplify

- We assume:
  - Data objects are simple text documents
  - Requests are lists of keywords, we have a hit if all keywords appear in a file
  - Data sharing nodes (providers) are stable

$Keyword_1$	$Doc_{1,1}, Doc_{1,2}, \ldots, Doc_{1,k1}$
$Keyword_2$	$Doc_{2,1}, Doc_{2,2}, \ldots, Doc_{2,k2}$
…	…
$Keyword_n$	$Doc_{n,1}, Doc_{n,2}, \ldots, Doc_{n,kn}$

Fig.1 - The Local Data Index on an arbitrary node



State of the Art in P2P routing

- Centralized Index (Napster)
  - Not scalable, vulnerable, “no real P2P”
- Flooding (Gnutella)
  - wastes bandwidth, not reliable (TTL)
- DHT Indices
  - Scalable and reliable, distributed load
  - BUT **distributes data deterministic** over the network by a special key-value which would imply **moving of data**
  - A good starting point ?

A first DHT-Solution - Idea

- Idea: Put a “DHT routing layer” above the real data network

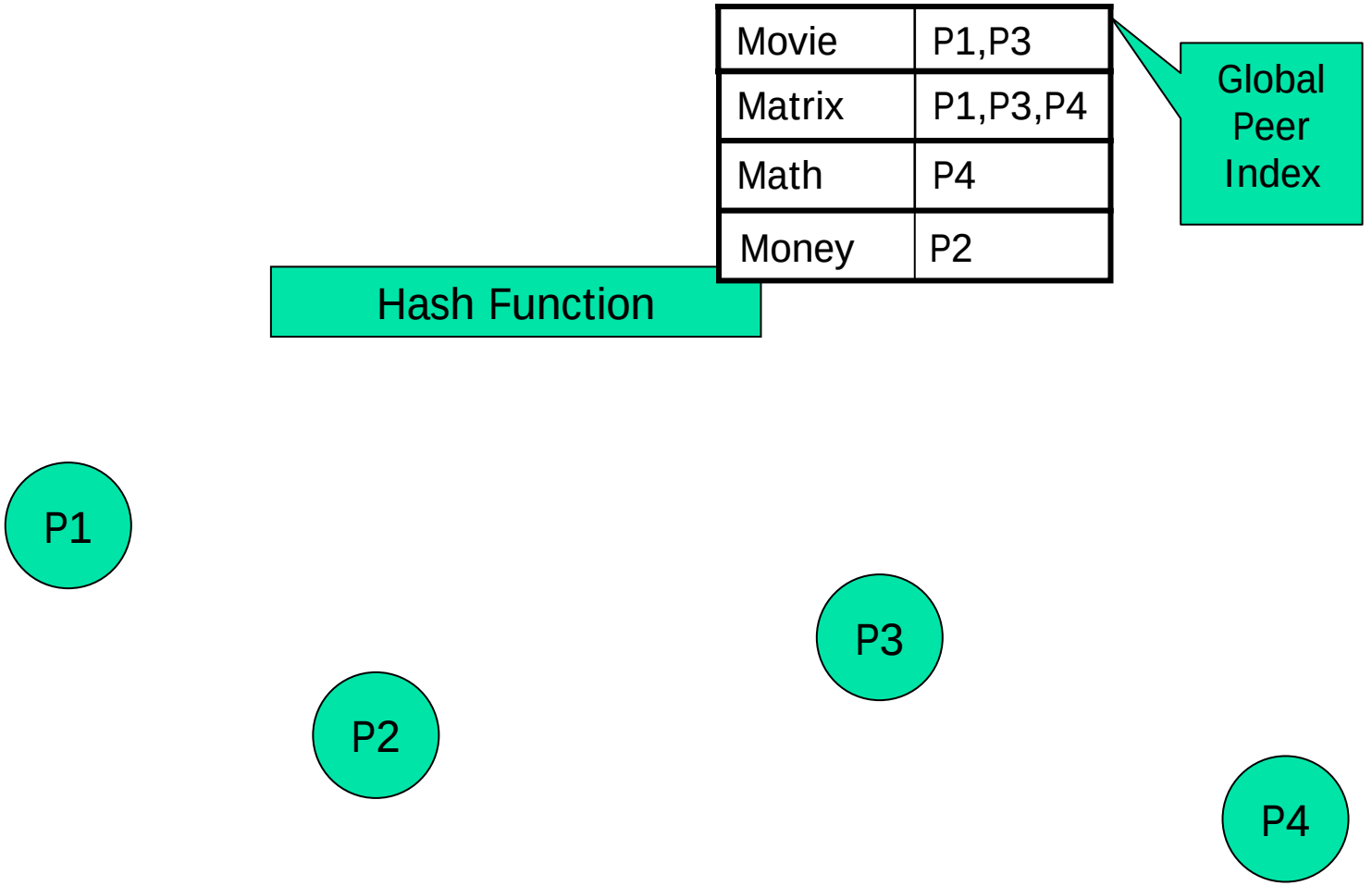
- For that we need **Peer Indices**:

$Keyword_1$	$Peer_{1,1}, Peer_{1,2}, \ldots, Peer_{1,k1}$
$Keyword_2$	$Peer_{2,1}, Peer_{2,2}, \ldots, Peer_{2,k2}$
…	…
$Keyword_n$	$Peer_{n,1}, Peer_{n,2}, \ldots, Peer_{n,kn}$

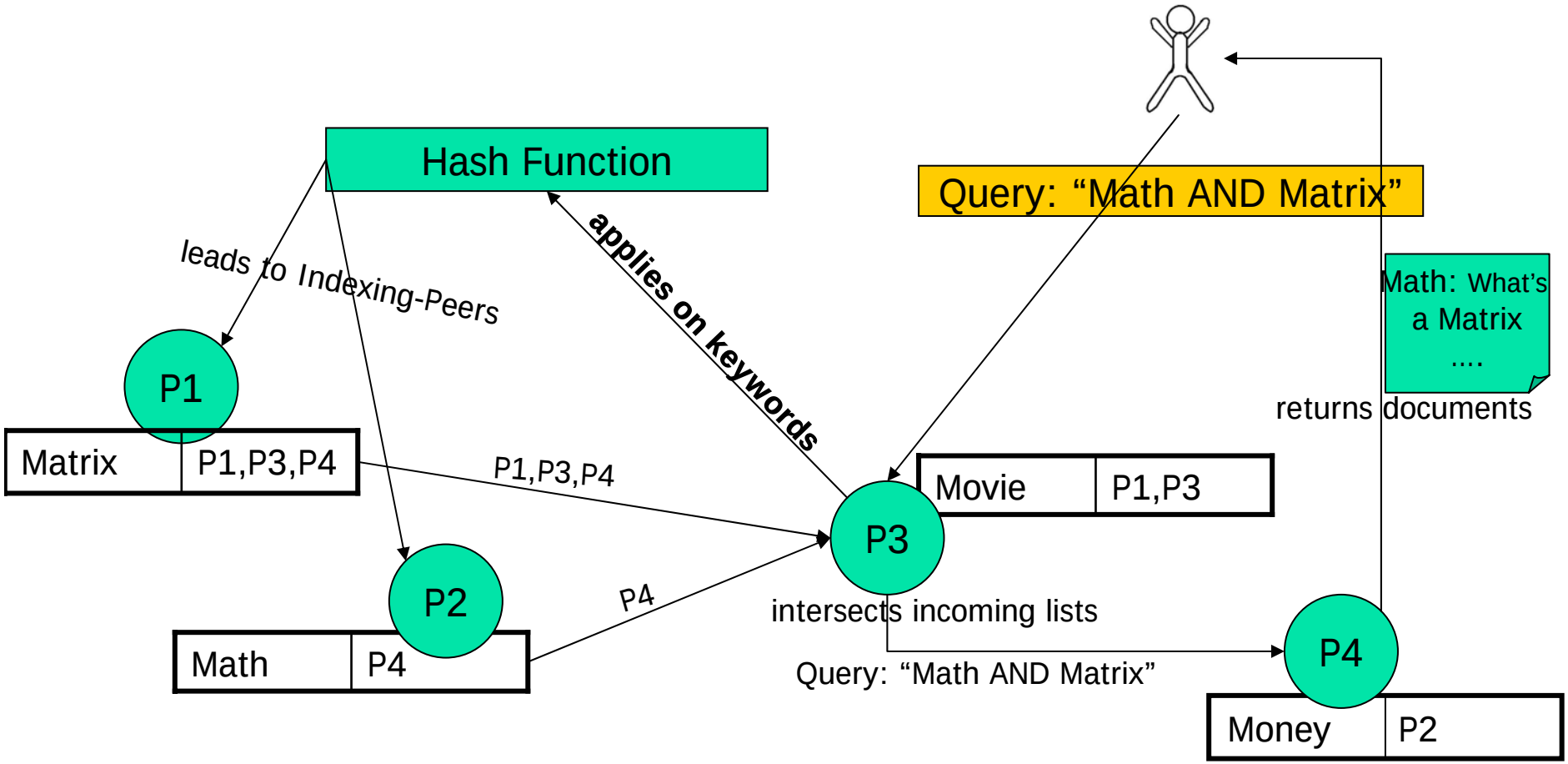
Fig.2 - A Peer Index

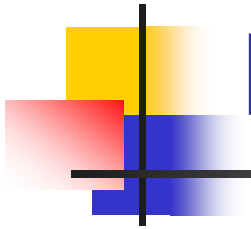
If such **a Peer Index covers every Keyword** in the network, we call it a **Global Peer Index**.

A first DHT-Solution - Example



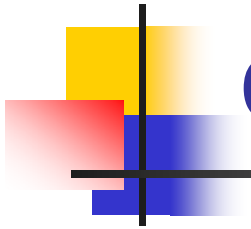
A first DHT-Solution - Example





DHT-Solution – Good Enough?

- It is reliable, theoretically scalable, Ownership respecting and adaptable
- **BUT nodes are not free** to decide themselves what indices to maintain
- **Performance Loss** through “Extra”-Lookup
- Deterministic Distribution increases Overhead



GALANX in Detail – A perfect Solution?

- **Imagine** a configuration where **every node maintains the complete Global Peer Index**
- **In terms of query routing** that would be a **perfect situation**, because every query could be routed directly to the correct node(s)
- **BUT Updates** would be **extremely expensive** and **Peer Indices** would be **very large**

In Detail – GALANX Peer Indices

$Keyword_1$	$Peer_{1,1}, Peer_{1,2}, \ldots, Peer_{1,k1}$
$Keyword_2$	$Peer_{2,1}, Peer_{2,2}, \ldots, Peer_{2,k2}$
…	…
$Keyword_j$	$Peer_{j,1}, Peer_{j,2}, \ldots, Peer_{j,kj}$

Direct Peer Index

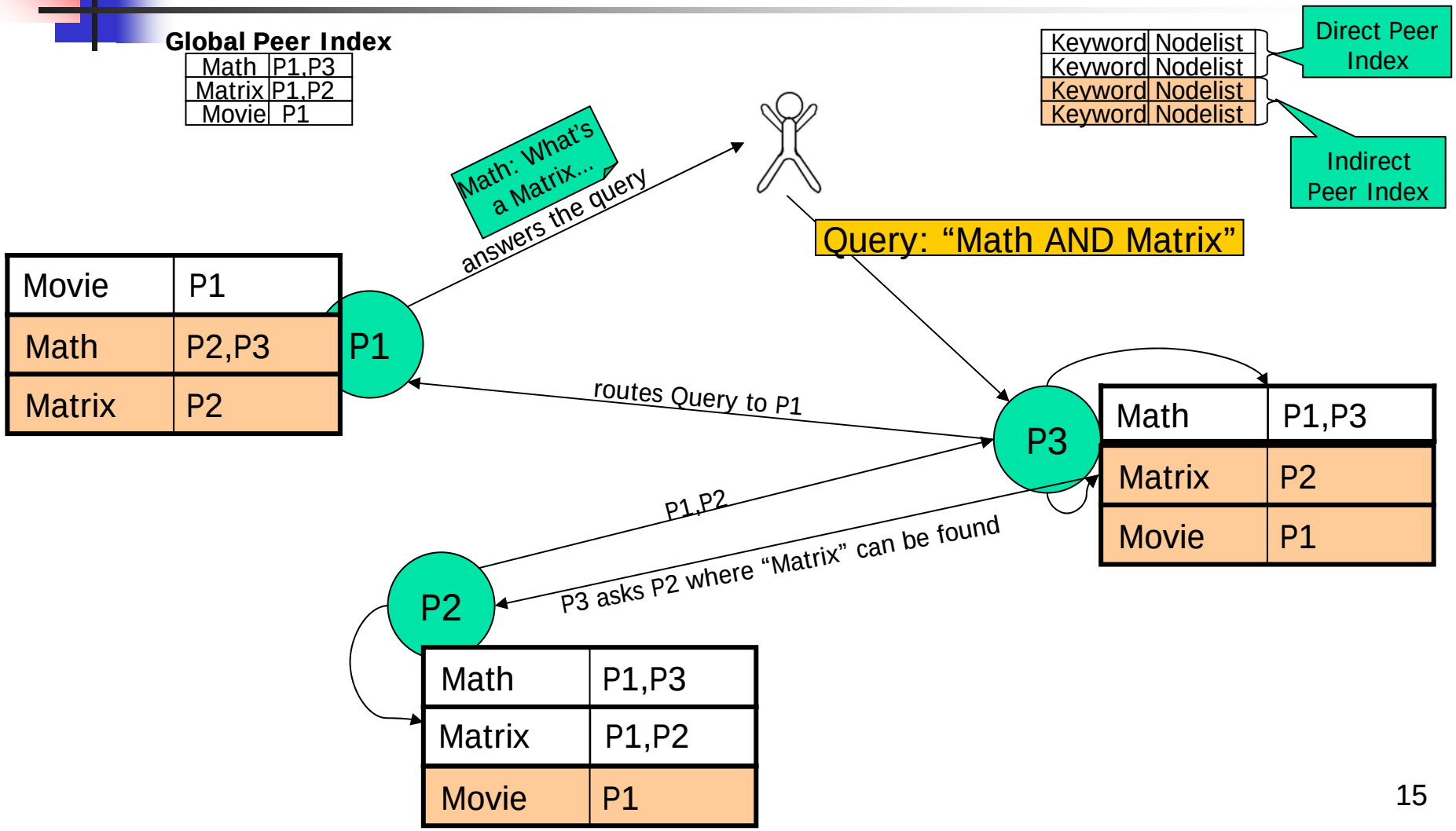
$Keyword_{j+1}$	$Peer_{j+1,1}, Peer_{j+1,2}$
$Keyword_{j+2}$	$Peer_{j+2,1}, Peer_{j+2,2}, Peer_{j+2,3}$
…	…
$Keyword_n$	$Peer_{n,1}, Peer_{n,2}$

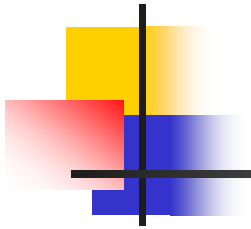
Indirect Peer Index

Fig.3 – GALANX Peer Index

- The **Direct Peer Index** is a Peer Index as known
  - covering several keywords with full list of data holding peers
- The **Indirect Peer Index** has for each of its keywords a list of peers maintaining a Direct Peer Index for this keyword
- Both together cover all keywords in the network

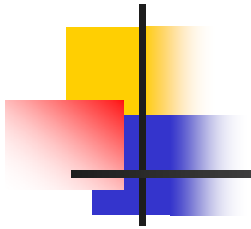
In Detail – Routing Example





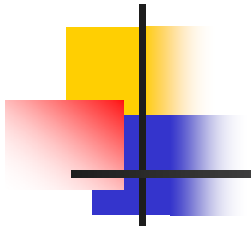
In Detail – Properties?

- If **all keywords** of a query are **in the Direct Peer Index**, that's **perfect**
- If not, the **remaining keywords** can probably **be found at** just **one other node** or at least at **less than with pure DHT** distribution (by intersecting the Peer lists wisely)
- If a **node's strategy is good**, most **popular words** can be found **in Direct Peer Index**



In Detail – Direct Index Construction

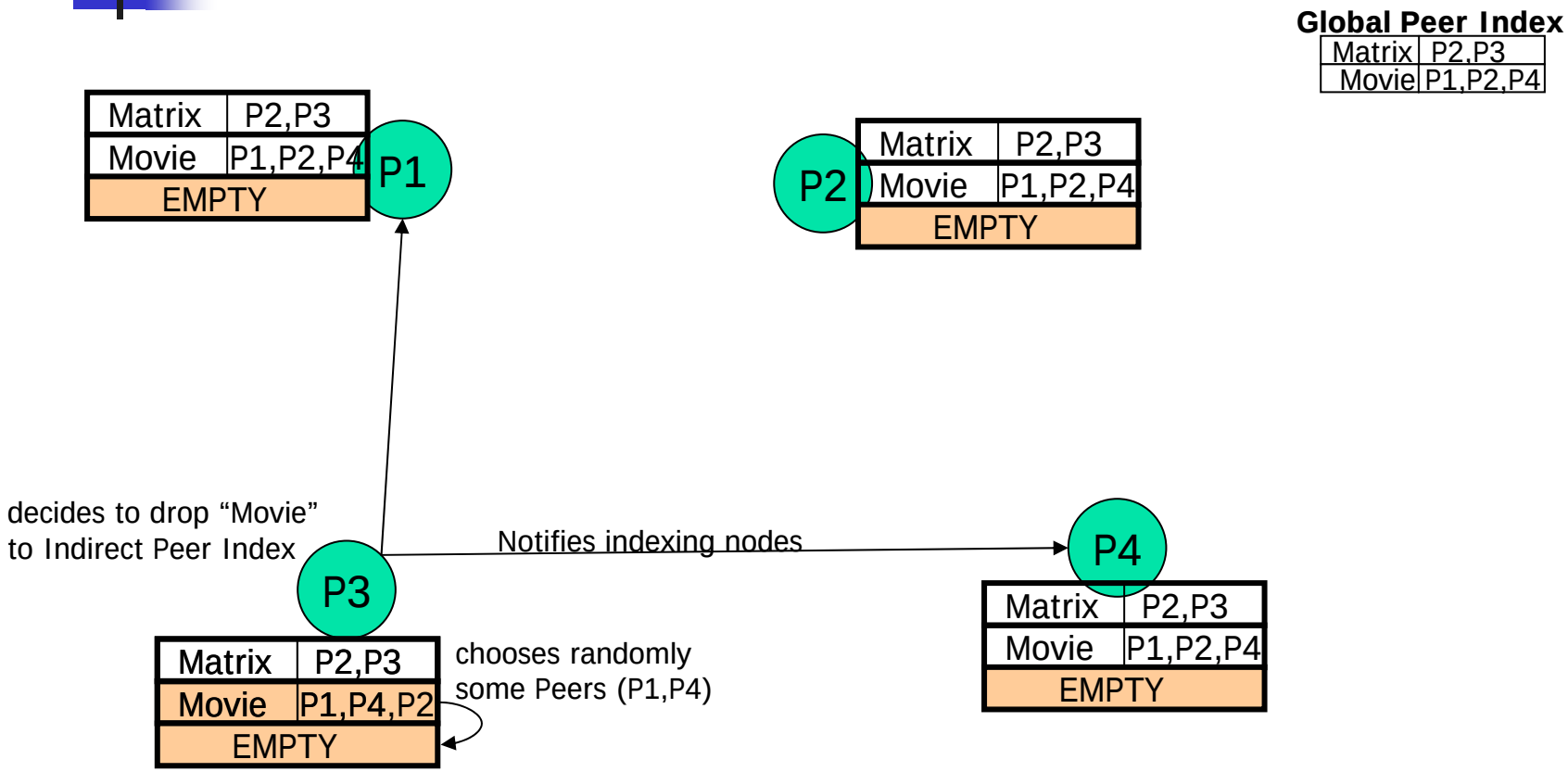
- Starting with few (or one) nodes
  - every node holds the complete Global Peer Index
- Joining nodes copy an arbitrary Peer Index
  - update that with own data
- Nodes can drop keywords from Direct to Indirect Peer Index by:
  - Randomly **pick a few "keyword-data" holding peers**
  - **Put them in the Indirect Peer Index** as indexing nodes
  - **Notify them**, so they can notify you if they drop that word



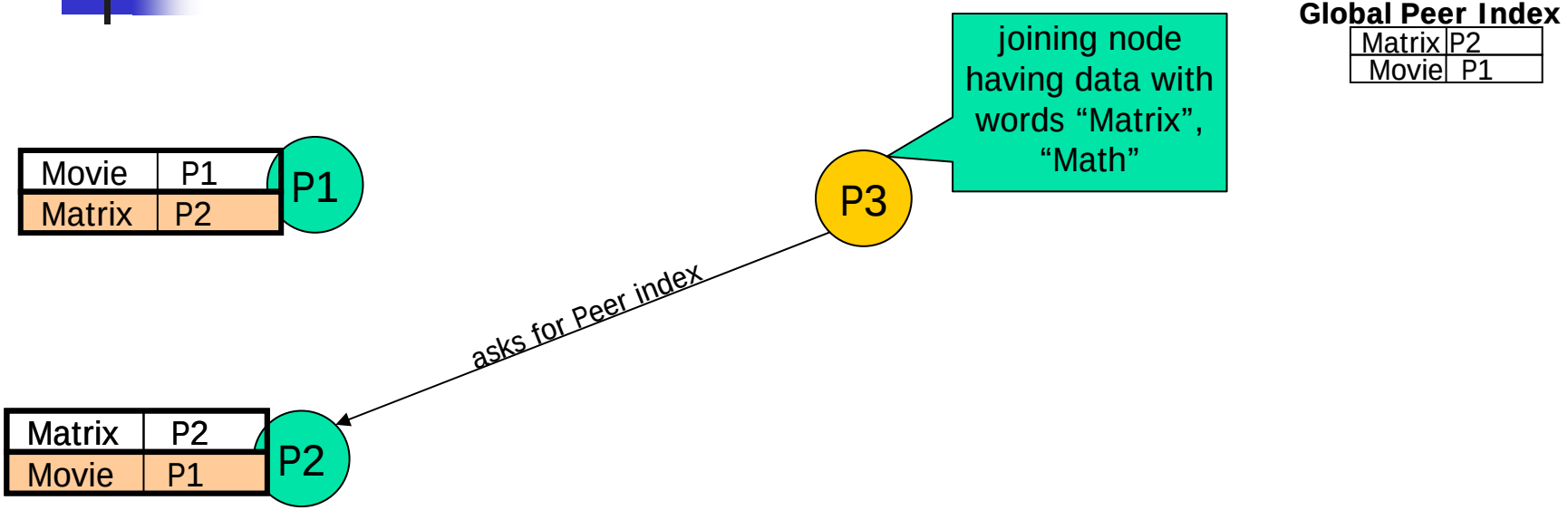
In Detail – Keyword Updates

- Keyword already exists in the network:
  - Entry in the Peer Index -> Update sent to all indexing nodes
- Keyword is completely new:
  - Update is sent to all nodes in the network
- Possible: lazy Update-strategy with wildcards

In Detail – Direct Construction



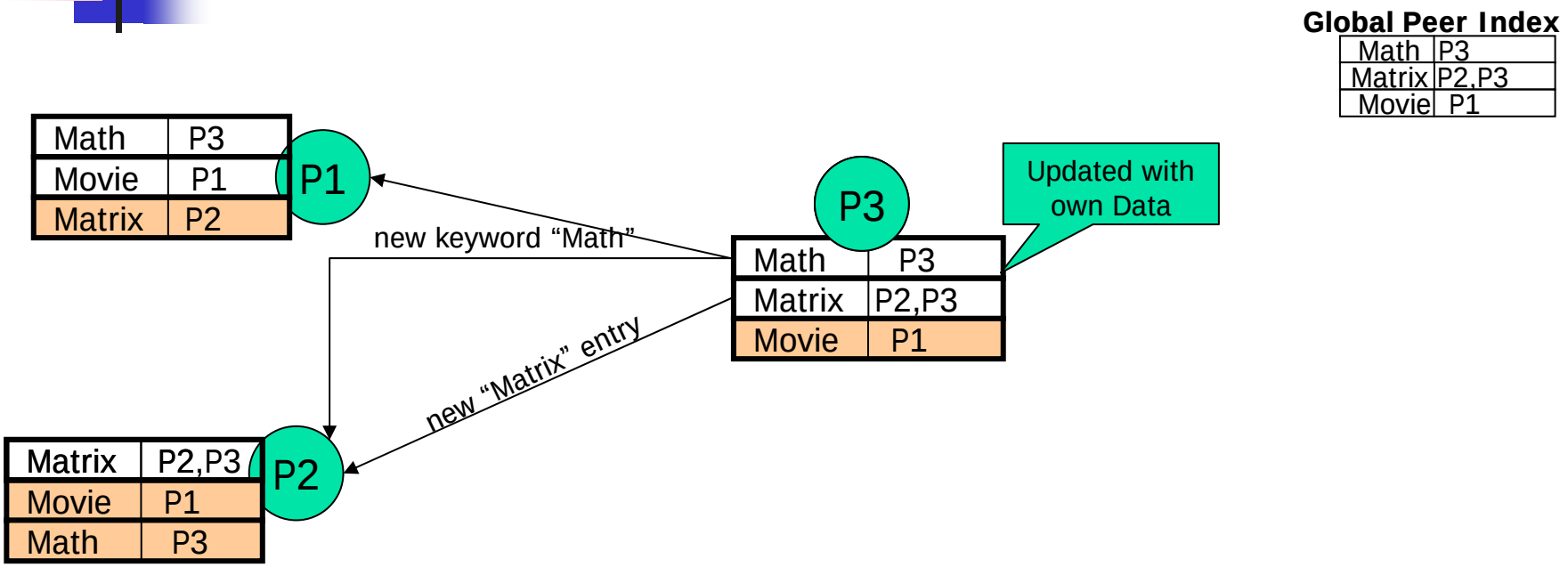
In Detail – Update Example



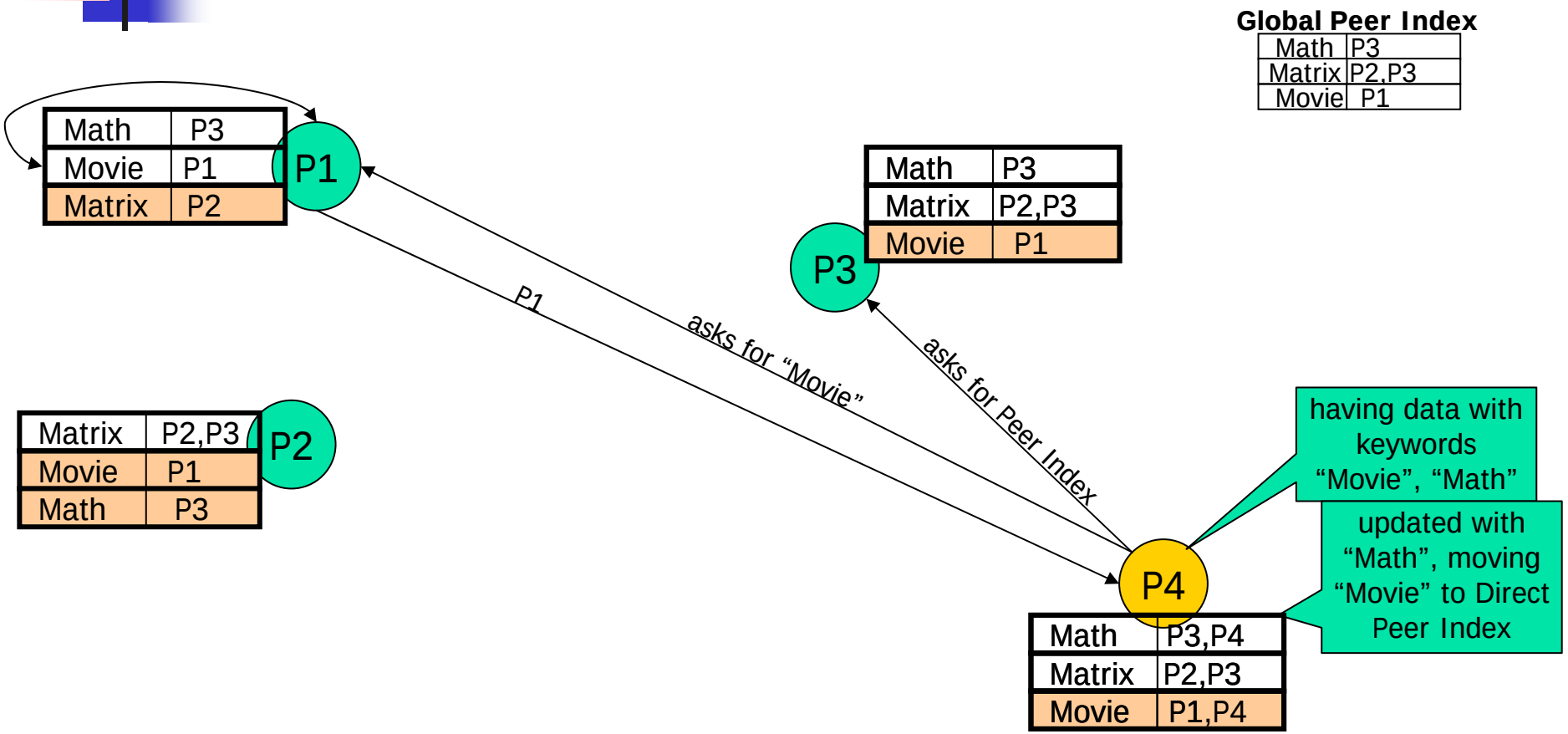
In Detail – Update Example

Global Peer Index

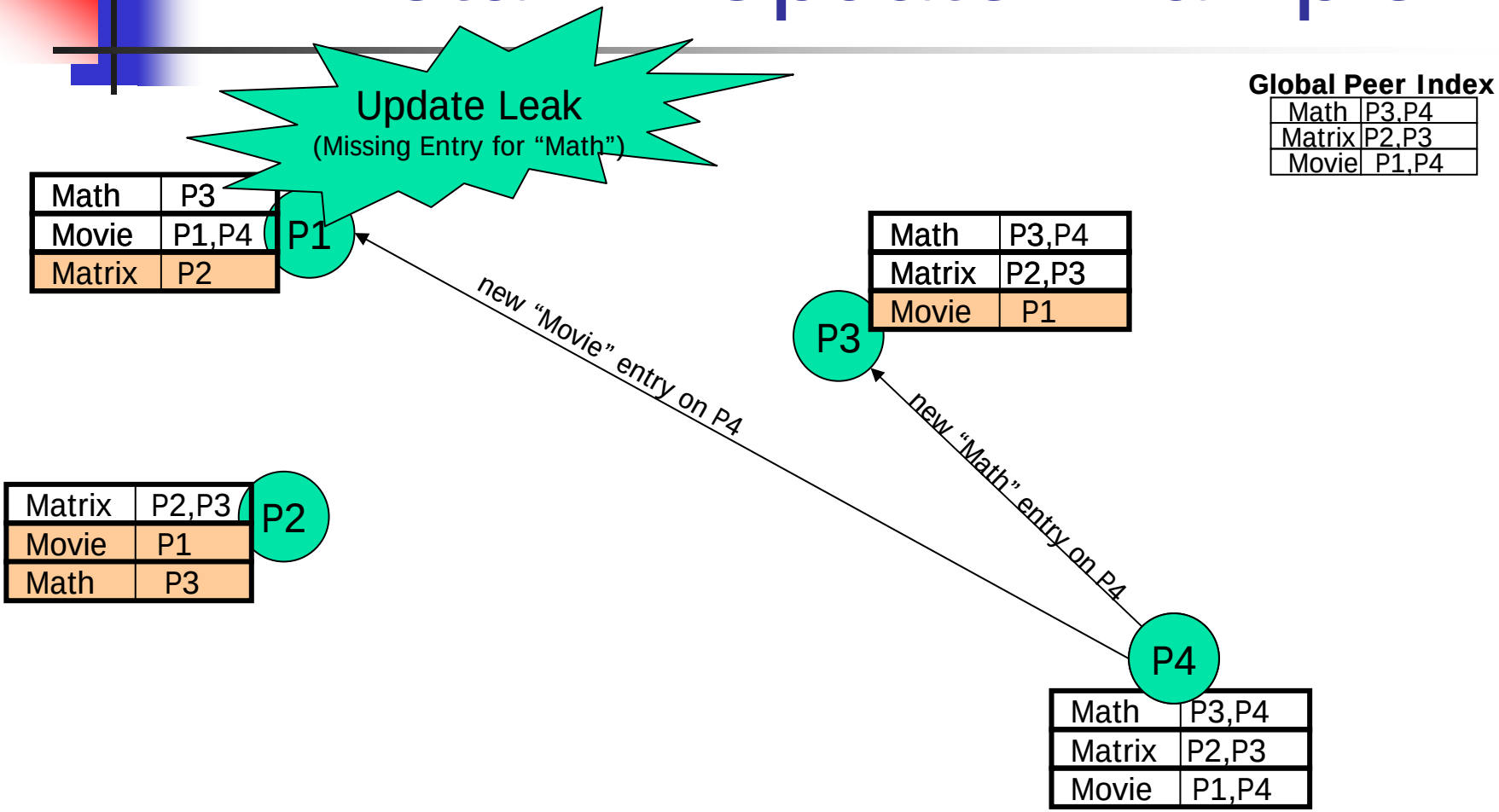
Math	P3
Matrix	P2,P3
Movie	P1

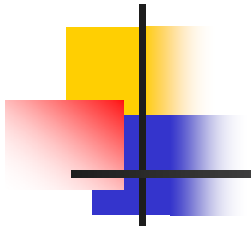


In Detail – Update Example



In Detail - Update Example

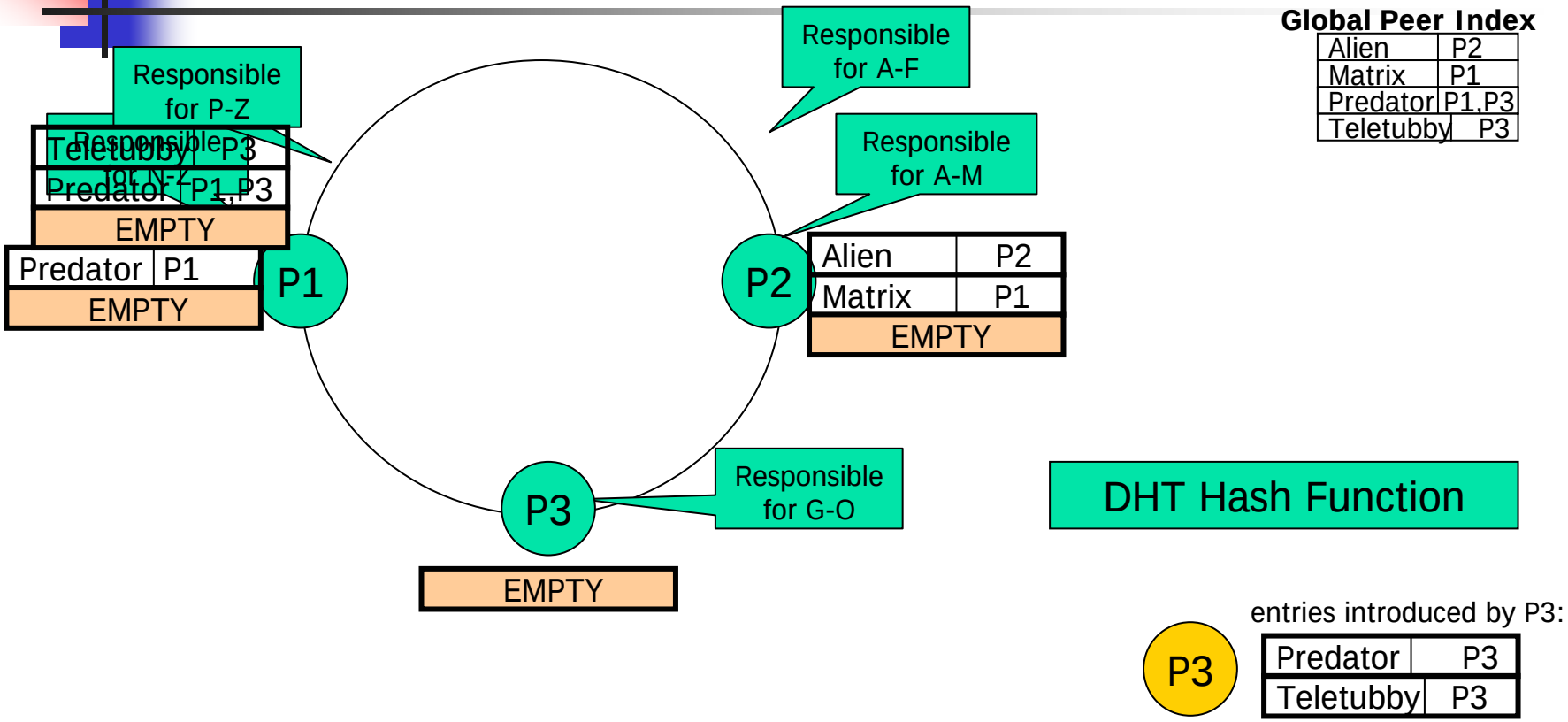




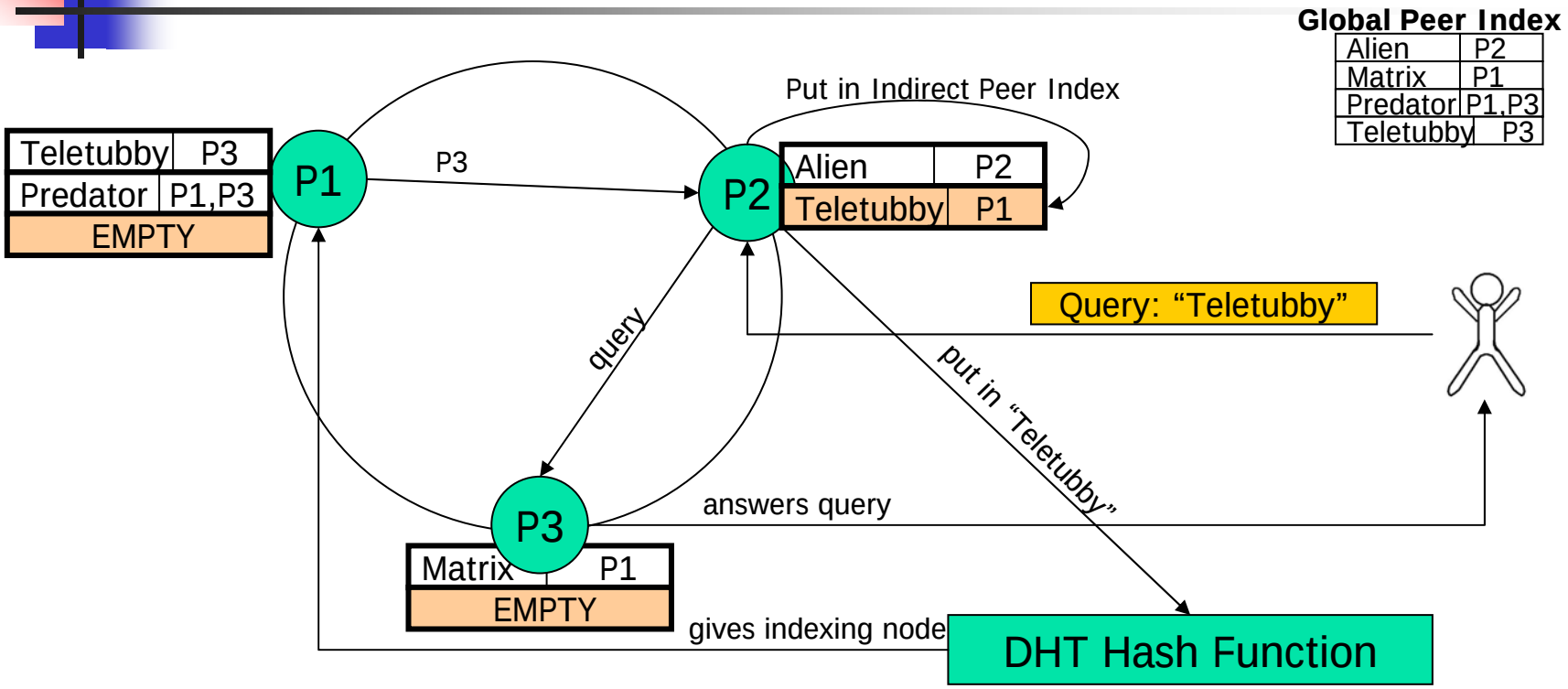
In Detail – DHT Index Construction

- Starting again with few nodes
  - The **Global Peer Index is distributed** over all peers **via a Hash function**
  - Every Peer is responsible for indexing a part of the Keyspace
- Joining nodes get “their” Keyspace by the DHT mechanism, corresponding Peer Index entries are moved
- Updates work similar as seen using DHT mechanism
- Nodes trained by Queries
  - Keywords taken in Indirect (or Direct) Peer Index if not already in Direct Peer Index)

In Detail – DHT Construction



In Detail – DHT Training



Implementation

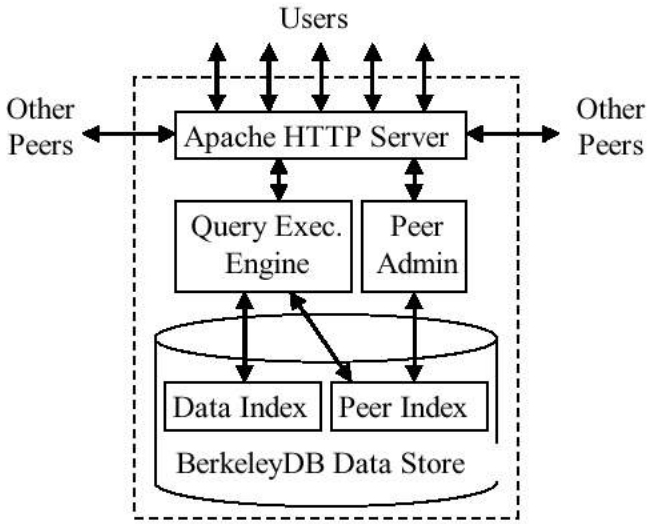
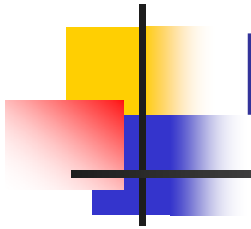


Fig.4 - GALANX Prototype Architecture

- Apache Server handles all communication
- Query Execution Engine, Peer Admin, User Interface embedded as modules, written in C
- BerkeleyDB as Database holding Data & Peer Index

Apache Version multi-process mode doesn't work well with BerkeleyDB => **lower Performance as possible** because queries are queued in some cases



Performance Tests

- Experimental Setup:
  - 100-node computer LAN-cluster, 933/550MHz, 1GB memory
  - 20million+ HTML-pages distributed by category to nodes
  - 2.49 million keywords, average 125.000 keywords per node
  - Simulated user queries with dynamic expiration time
  - Initial expiration time 10 sec, min 5 sec, max 30 sec
  - Only queries which can be answered

Performance Tests

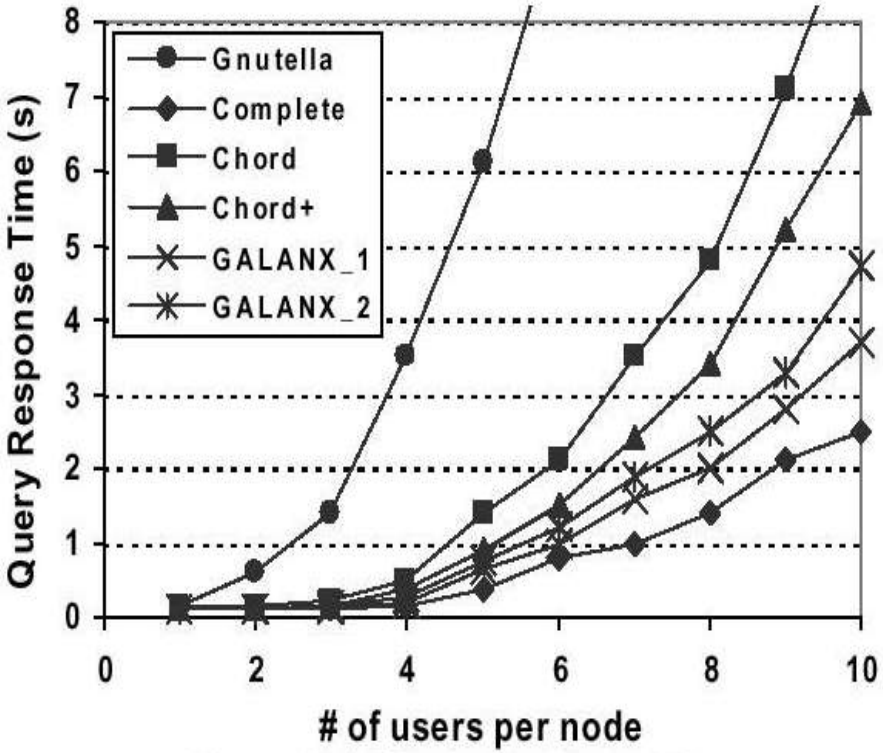


Figure 5.1 Query Response Time

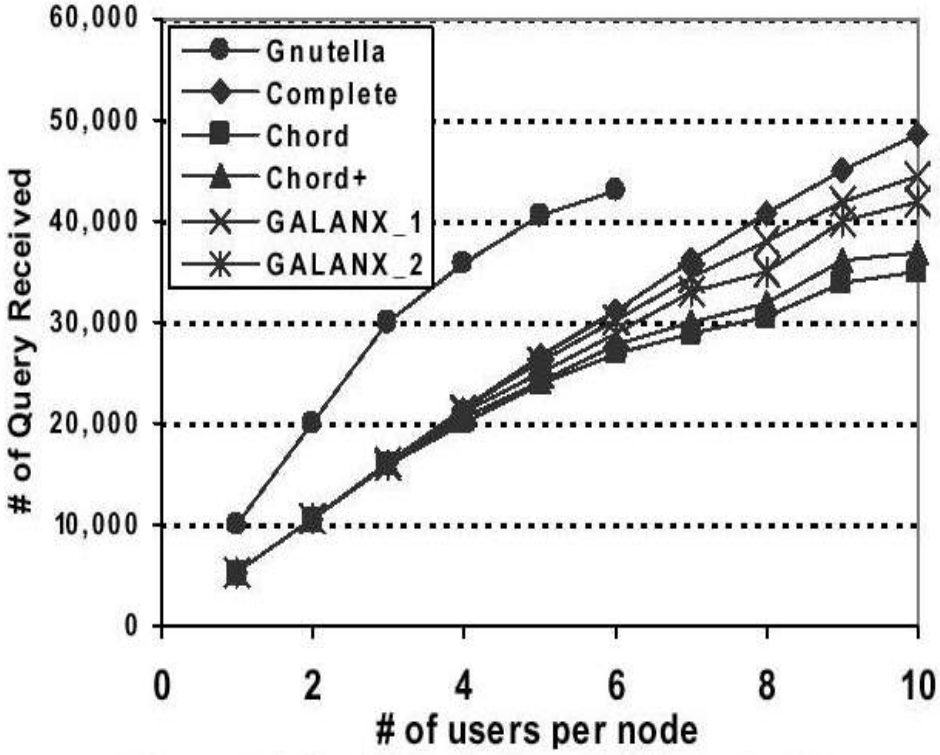


Figure 5.2 # of Queries received on Each Peer

Performance Tests

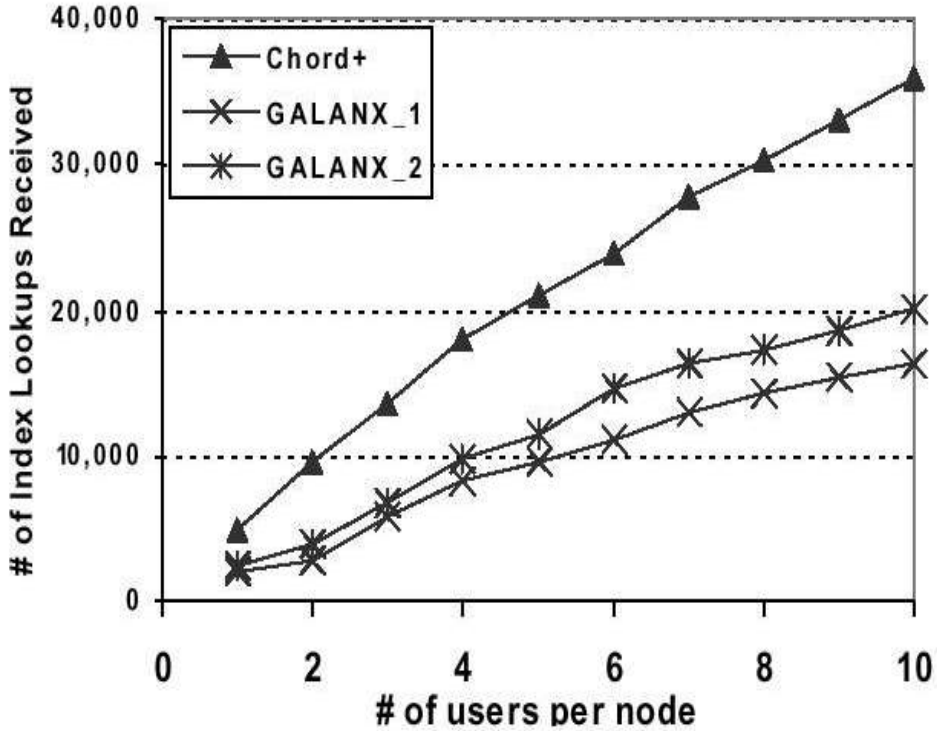


Figure 5.3 # of Index Lookup Requests Received

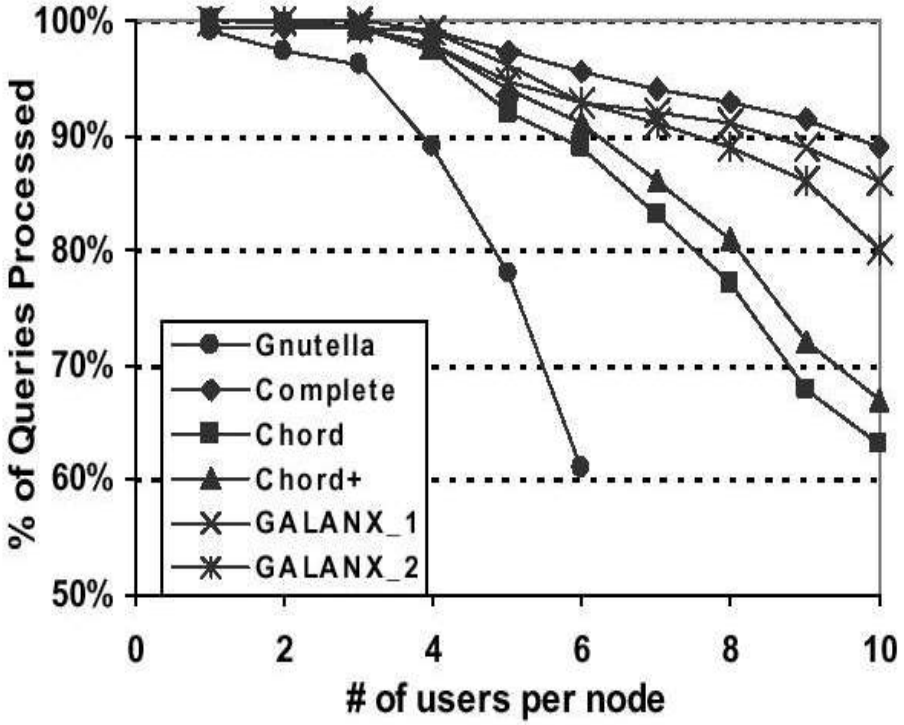


Figure 5.4 Query Execution Ratio

Performance Tests

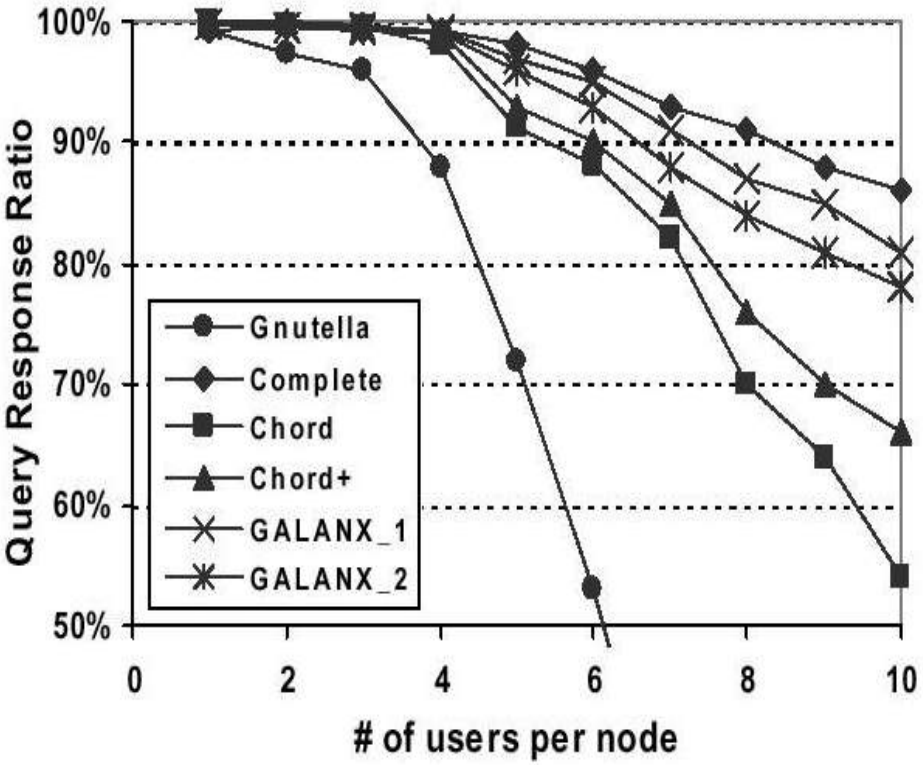


Figure 5.5 Query Response Ratio

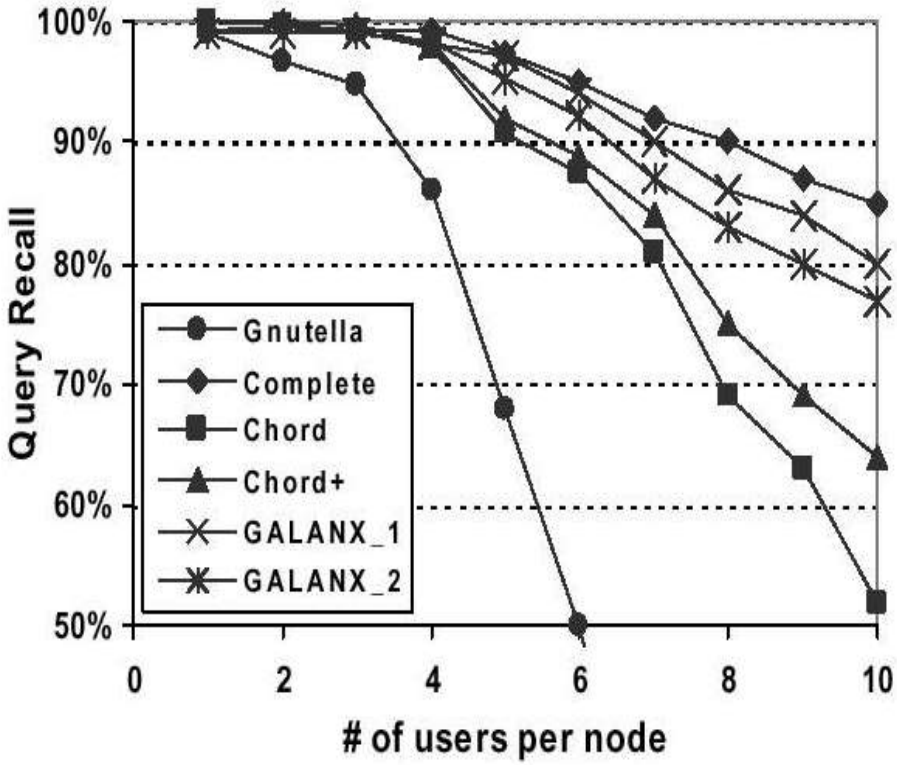


Figure 5.6 Query Recall

Performance Tests

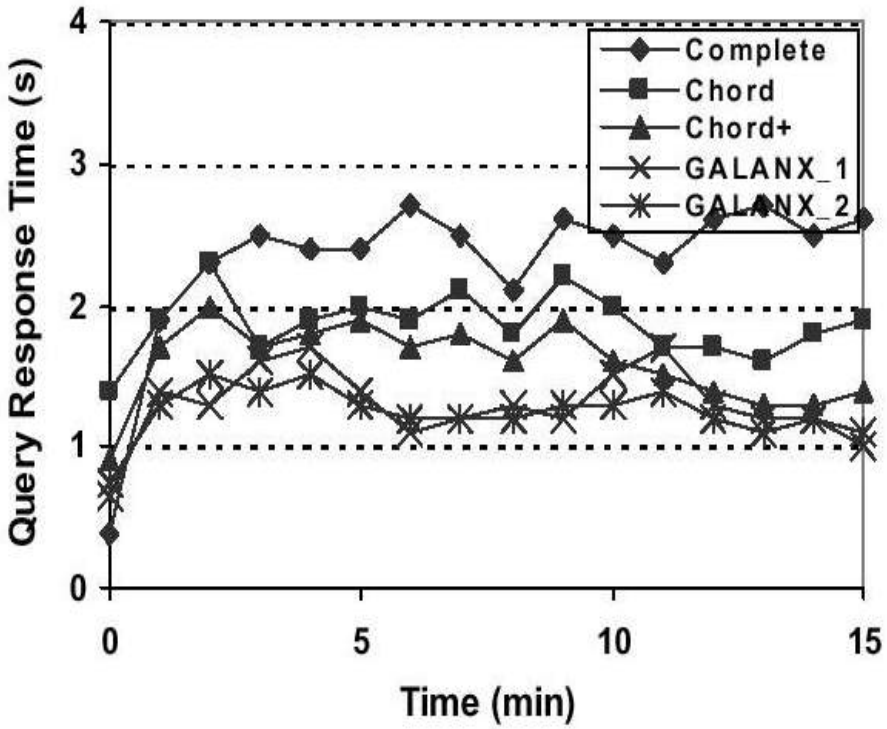


Figure 5.7 Query Response Time (with new data)

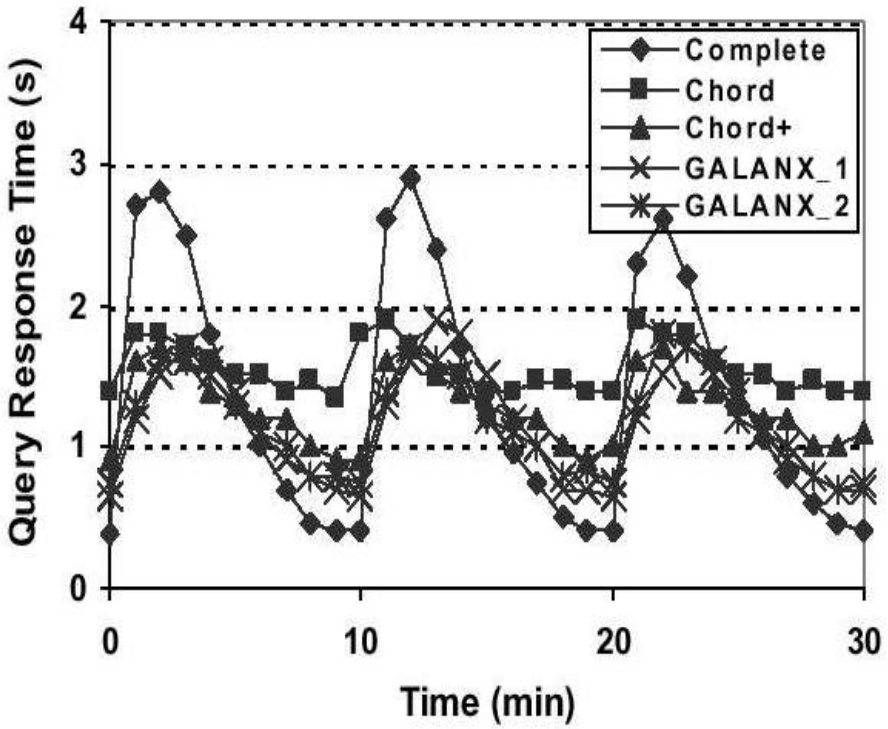
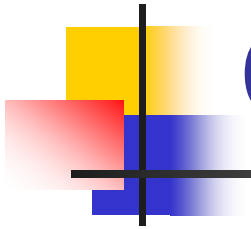
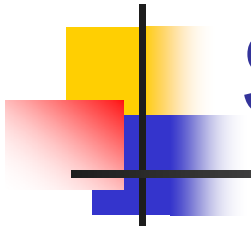


Figure 5.8 Query Response Time (with new nodes)



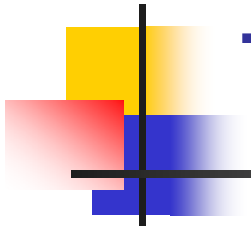
Conclusions & Future Work

- Fast routing Approach, IF Assumptions hold
- Just one piece of the whole; to get a really applicable system lot has to be added
- What about reliability in really dynamic Environment?
- Authors next aim:
  - Extend to more complex forms of queries
  - Introduce Information Retrieval Techniques like relevance ranking



Summary

- Want to have fast query routing system
  - Existing approaches expensive, not fast or not reliable->GALANX
- DHT-Approach
  - Global Peer Index distributed over all nodes
  - Not good enough (extra lookup, nodes not free)
- GALANX Peer Indices
  - Direct & Indirect Peer Index, Design delivers peers freedom
  - Direct Construction: complete global peer index on every node, updates sent to indexing nodes or complete network, dropping keywords to indirect Peer Index
  - DHT Construction: Having DHT-Distribution of Keyspace, nodes responsible (direct indexing) for certain Keyspace, training with queries
  - Experiments look good, although system not perfect



Thanks

- for your attendance
- Questions ?...