

DAS-P2P 2007, January 16, 2007

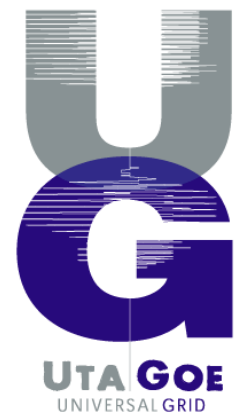


Overlay Weaver: An Overlay Construction Toolkit

Kazuyuki Shudo

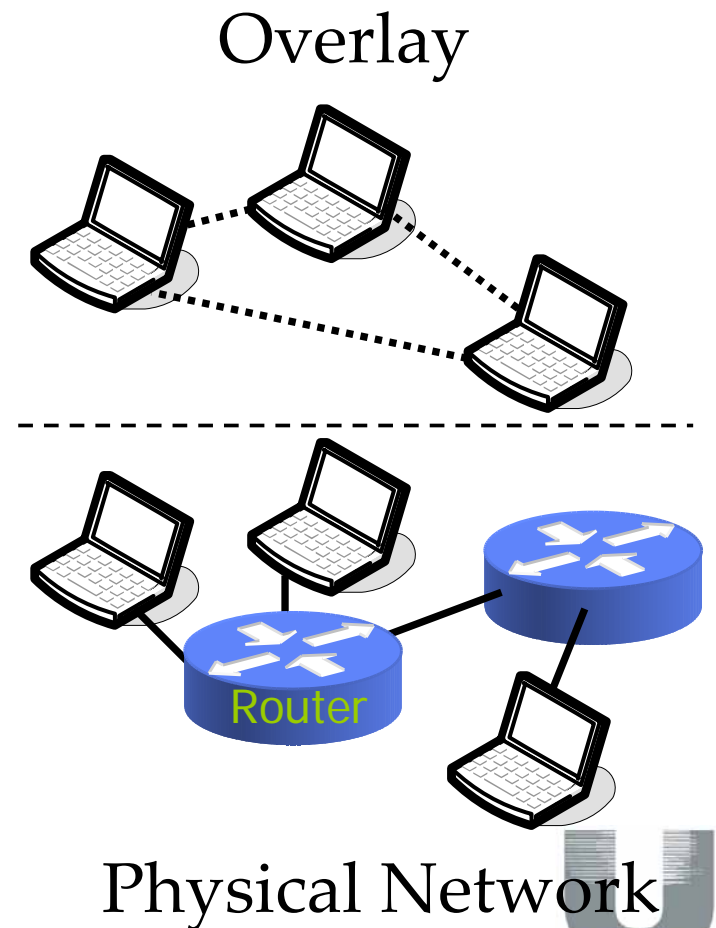
*Overlay
Weaver*

<http://overlayweaver.sf.net/>

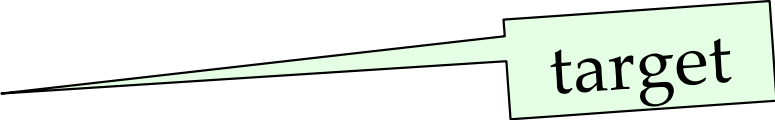


Overlay

- An network constructed over another network
 - e.g. Internet over a telephone network
 - e.g. Networks of file sharing programs
 - FastTrack, eDonkey2K, Gnutella, ...
 - with more than 1,000,000 nodes.
- Application-level network
 - constructed in a **autonomic and decentralized** way to keep **performance and fault-tolerance** with 1,000s and 1,000,000s nodes.
- Its topology is independent from the underlying network (i.e. Internet).
 - Then, called **Overlay Network** or **Network Overlay**.



Unstructured and Structured

- Unstructured overlay
 - e.g. **Gnutella** network, Winny network
 - **No (few) constraint** imposed on which node to be a neighbor (topology).
 - An existing object on it may be found.
 - Generally, less-efficient but supports flexible (e.g. range) search.
- Structured overlay 
 - e.g. Network for **distributed hashtable (DHT)**
 - **An algorithm-based constraints** imposed on which node to be a neighbor.
 - An existing object on it is (almost) certainly found.
 - Generally, efficient but it has been said to be weak in flexible search.

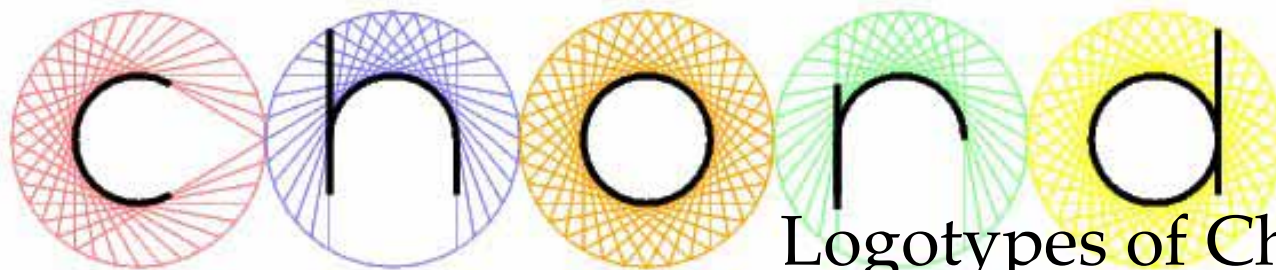
Overlay Weaver

Overlay Weaver

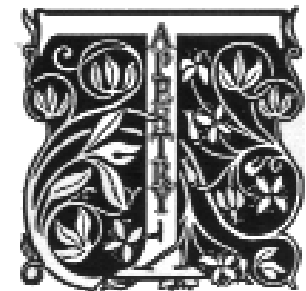
- We see analogies between structured overlays and weaving.
 - Chord, Tapestry, ...
- Overlay Weaver
 - A weaving device of (structured) overlays



Weaver



Logotypes of Chord and Tapestry

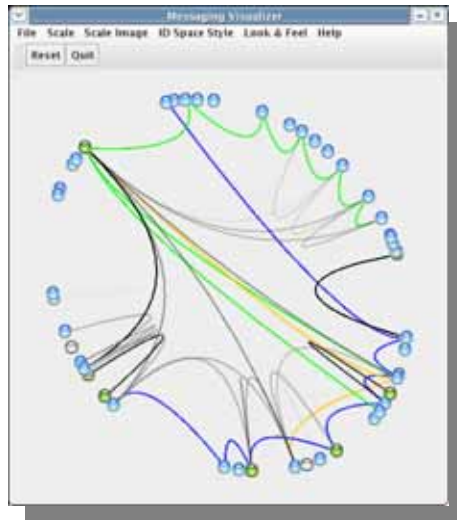


Overlay Weaver

- DHT Library

- in Java
- about 20,000 steps
- licensed under Apache License 2.0
 - ready to be applied to various purpose.

Overlay
Visualizer



- Properties

- supports application-layer/level multicast (ALM), not only DHT.
- supports multiple lookup algorithms:
 - Chord, Kademlia, Koorde, Pastry, Tapestry, ...
- We can conduct experiments without writing code.
 - Operation with sample tools such as DHT shell and Mcast shells.
 - Measurement of # of messages, # of hops and ... with Emulator.
- A DHT is accessible via XML-RPC-based protocol.
 - the same protocol as Bamboo and OpenDHT.

Overlay Weaver as an Open Source Software

- <http://overlayweaver.sf.net/> (SourceForge)

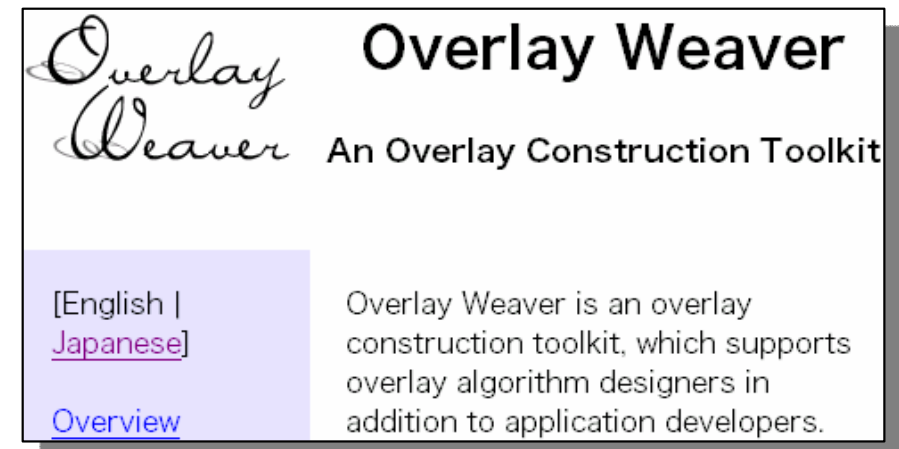
- released on 17th Jan, 2006.
- Apache License 2.0

- Statistics (as of 19:00, 14th Jan, 2007)

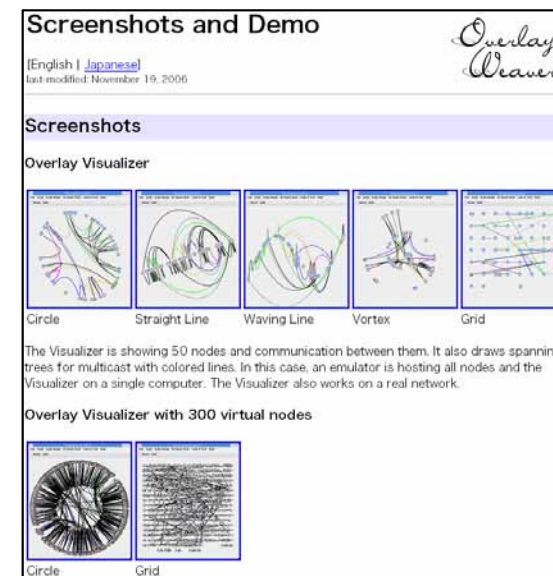
- 4,495 downloads
- # of subscribers of mail lists
 - English: 54, Japanese 71
- # of members of the mixi community: 122

- Expected users

- Algorithm designers
- Application developer
 - P2P storage of RDF documents by Dr. Matono (AIST)



The screenshot shows the top section of the Overlay Weaver website. On the left, the name 'Overlay Weaver' is written in a stylized, cursive font. To its right, the title 'Overlay Weaver' is in a bold, sans-serif font, followed by the subtitle 'An Overlay Construction Toolkit'. Below the title, there are links for '[English | Japanese]' and 'Overview'. On the right side of the header, a paragraph describes the toolkit: 'Overlay Weaver is an overlay construction toolkit, which supports overlay algorithm designers in addition to application developers.'



This screenshot shows the 'Screenshots and Demo' section of the website. It includes a sub-header 'Screenshots' and a section titled 'Overlay Visualizer'. Below this, there are five small images showing different network visualizations: 'Circle', 'Straight Line', 'Waving Line', 'Vortex', and 'Grid'. A paragraph explains that the visualizer shows 50 nodes and communication between them, drawing spanning trees for multicast. Below this, there is another section titled 'Overlay Visualizer with 300 virtual nodes' which shows two more images: 'Circle' and 'Grid'.

Web site

Problems of overlay research

- Distance between algorithm research and applications
 - Research implementations are often only for simulation to ensure its scalability.
- Much work to implement an algorithm
 - At least, 1,000s lines of code.
 - hard to debug because of the nature of a large-scale distributed system.
- Difficulty in fair and realistic comparison between algorithm implementations
 - often simulated. e.g. p2psim.
 - But, system behavior is heavily implicated in how it is implemented. Comparison of real impls is important.

Overlay construction toolkit

- Connects algorithm research to applications directly.
 - Once an algorithm is implemented, it **can be emulated** and also **works on a real network**.
 - In **Java**, in **natural programming**: can write new Thread()
 - 4,000 nodes emulations and 200 nodes experiments on a PC cluster have been conducted.
 - We can design, implement and verify an algorithm with Emulator and the resulted implementation works on Internet.
- Related work – different approaches
 - Bamboo, PIAX: supports both simulation and real-network, and needs special programming with an event-driven style.

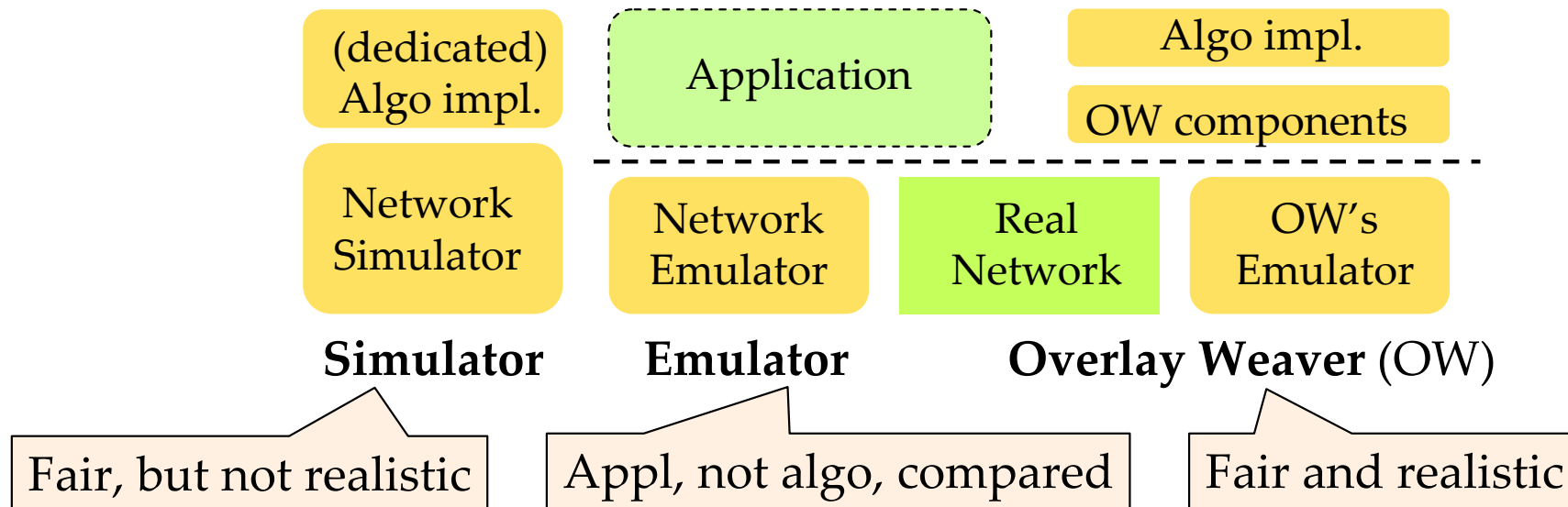
Overlay construction toolkit

- Facilitates algorithm implementation.
 - We could implement well-known 5 algorithms just in 100s steps.
 - Chord, Kademlia, Koorde, Pastry and Tapestry.
 - And 2 variations of Chord
 - Chord impl is 619 steps and Pastry impl is 872 steps in Java.
 - Rapid development and quality improvement with Emulator, which enables emulation with 1,000s nodes.
- Related work – different approaches
 - MACEDON: introduces an overlay describing language.
 - P2: introduces declarative overlay describing language OverLog.

Overlay Construction Toolkit

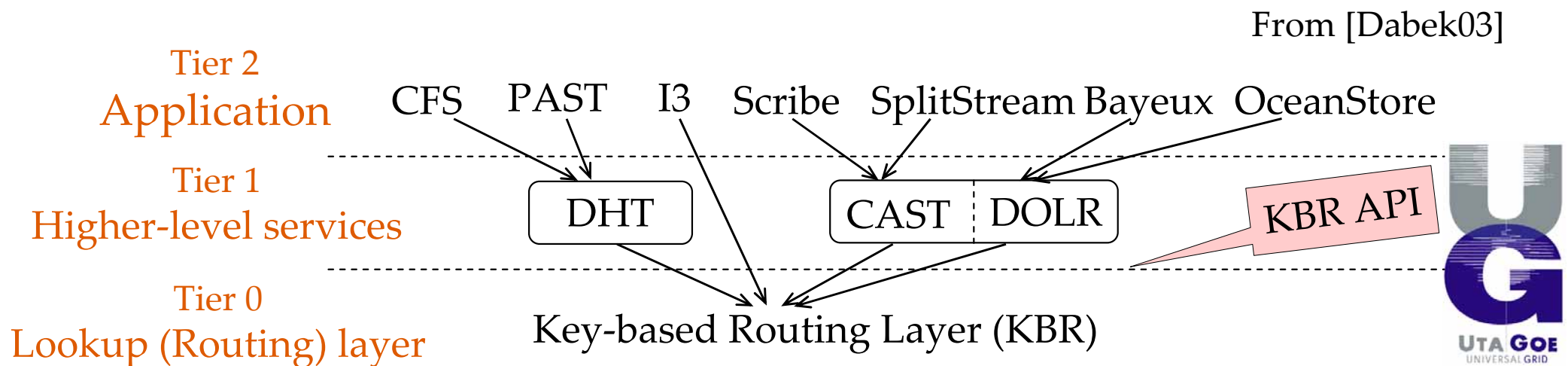
- Enables fair and realistic comparison between algorithm implementations.
 - Multiple algorithm implementations share OW components (e.g. communication).
 - Runs both on an emulator and on a real network.
- > fair and realistic comparison

Approaches to comparison between algorithm implementations




Overlay Weaver: Background

- Separation of lookup (routing) layer and higher-level services [Dabek03]
 - Frank Dabek et al., “Towards a common API for Structured Peer-to-Peer Overlays”, Proc. IPTPS’03.
- A lookup layer is called **key-based routing layer (KBR)**.
- Application (Tier 2) and higher-level services (Tier 1) can be independent from lookup algorithms (Chord, Pastry, ...)

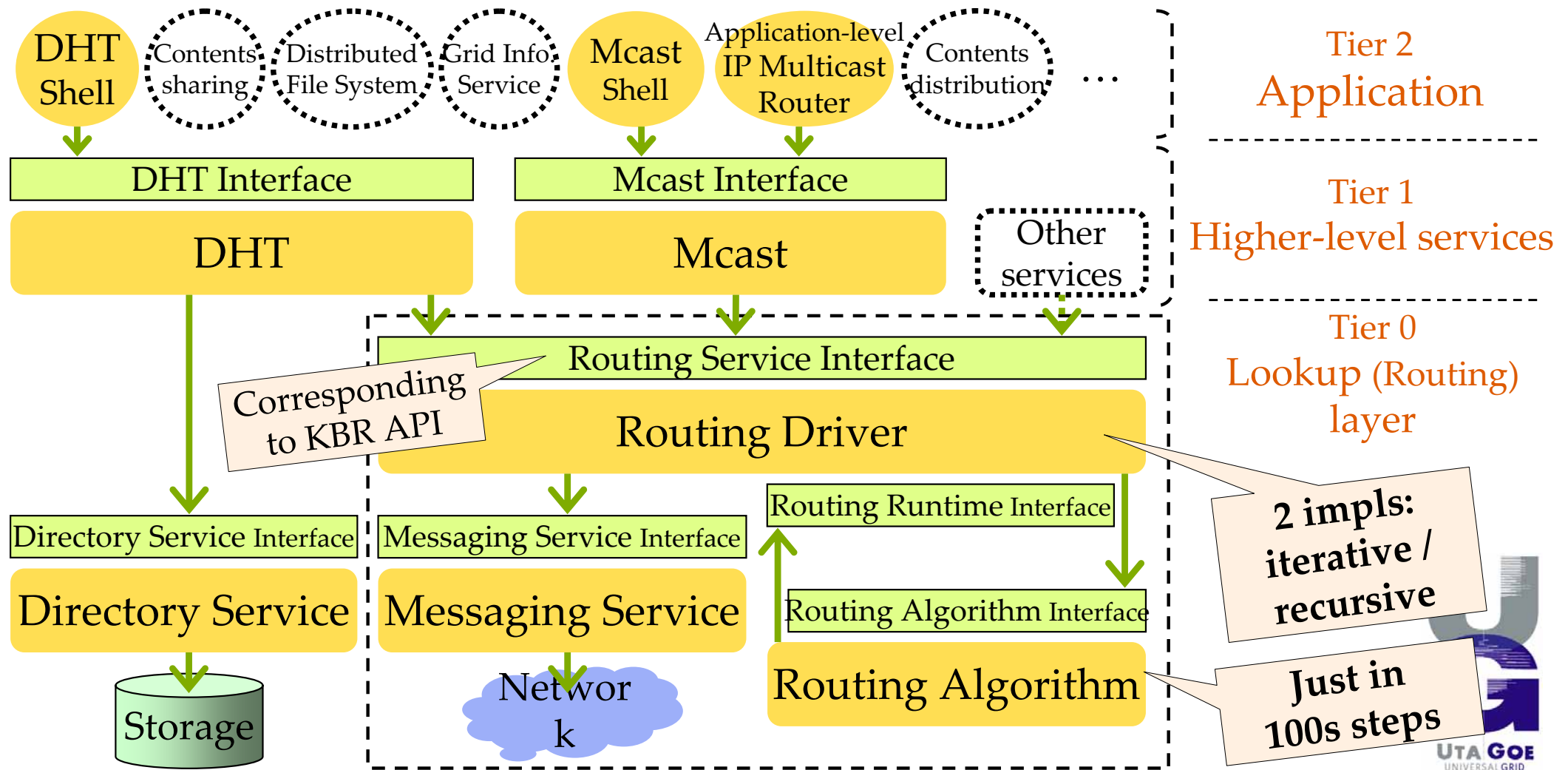


Decomposition of lookup layer

- Algorithm implementation still needs much work.
 - incl. communication and RPC.
 - e.g. Chord impl of p2psim has 2,835 steps.
 - To be easy
 - Not only your algorithm, but also other algos compared to with your one.
- 
- We factored out common processes to algorithms.
 - Further decomposition of KBR layer.

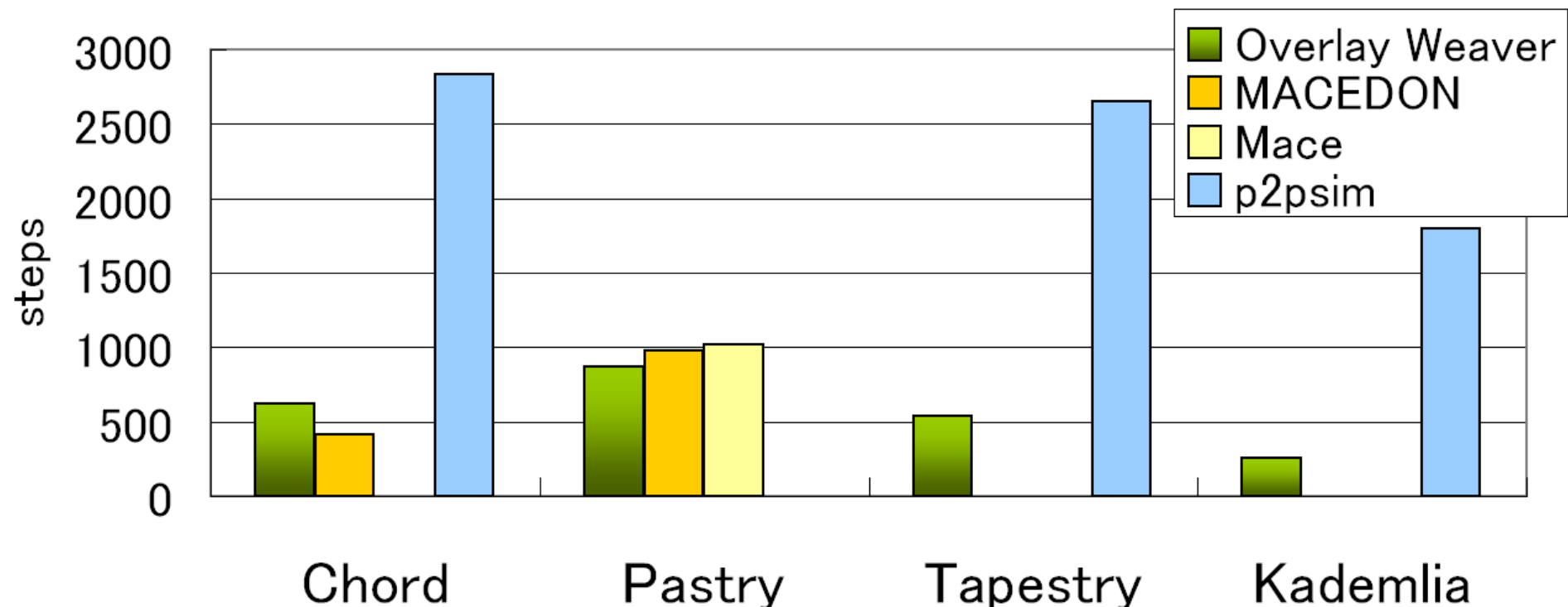
Decomposition of lookup layer

- Facilitates algorithm implementation
- 2 lookup styles are exchangeable: iterative and recursive



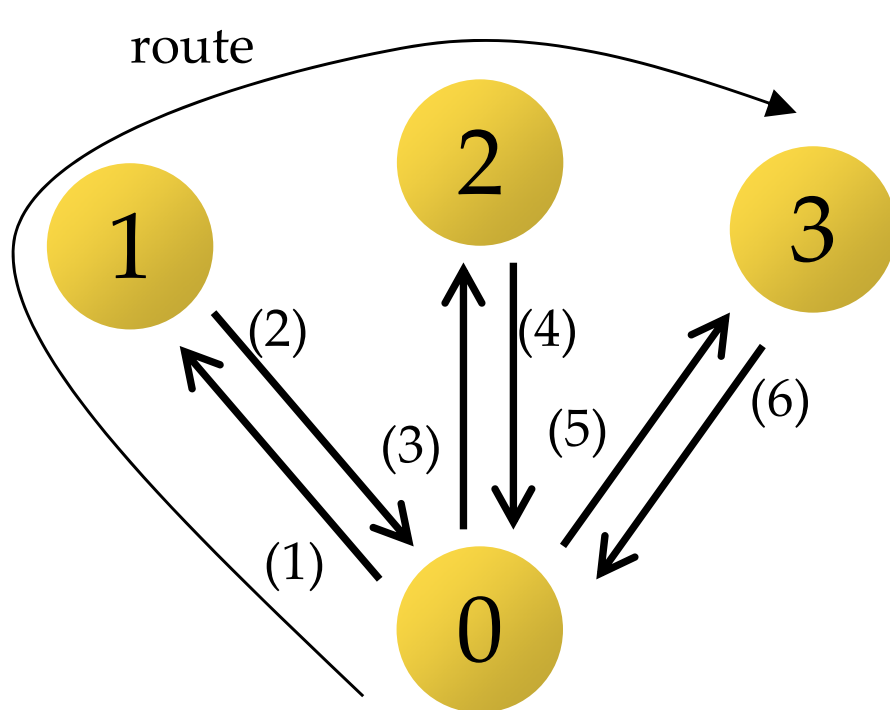
Facilitation of algorithm impl.

- We could implement well-known 5 lookup algorithms just in 100s lines of code.
- Compared with:
 - MACEDON: an algo described in the dedicated language and converted to C++.
 - p2psim: in C++. An algo impl does RPC itself -> more lines.

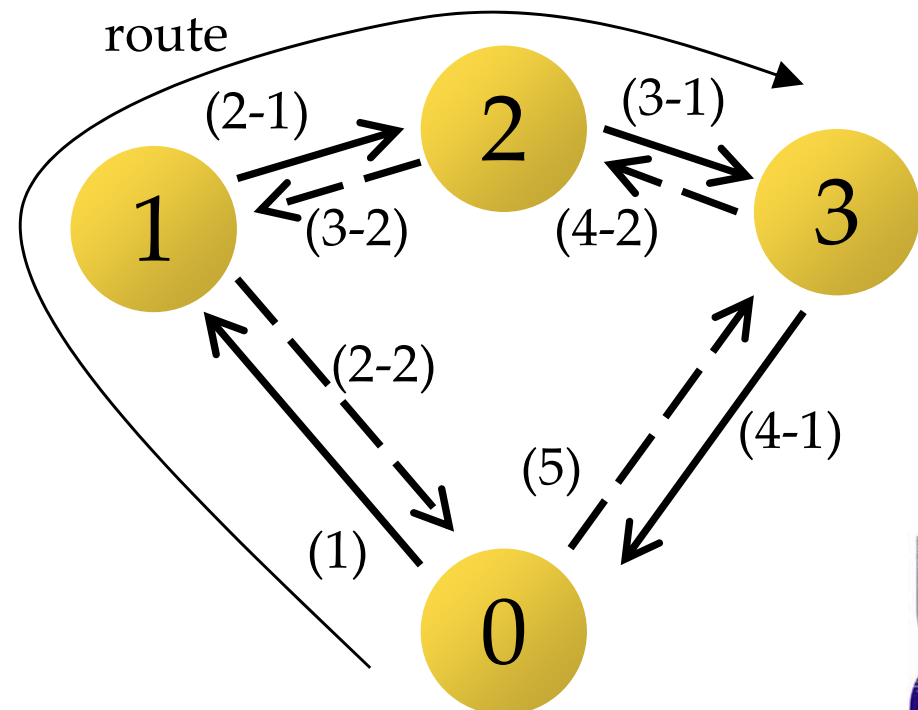


Exchangeable lookup styles

- Both Iterative / recursive lookup styles work with each lookup algorithms.
 - due to decomposition of lookup layer.
 - Each style has its own strengths and weaknesses.



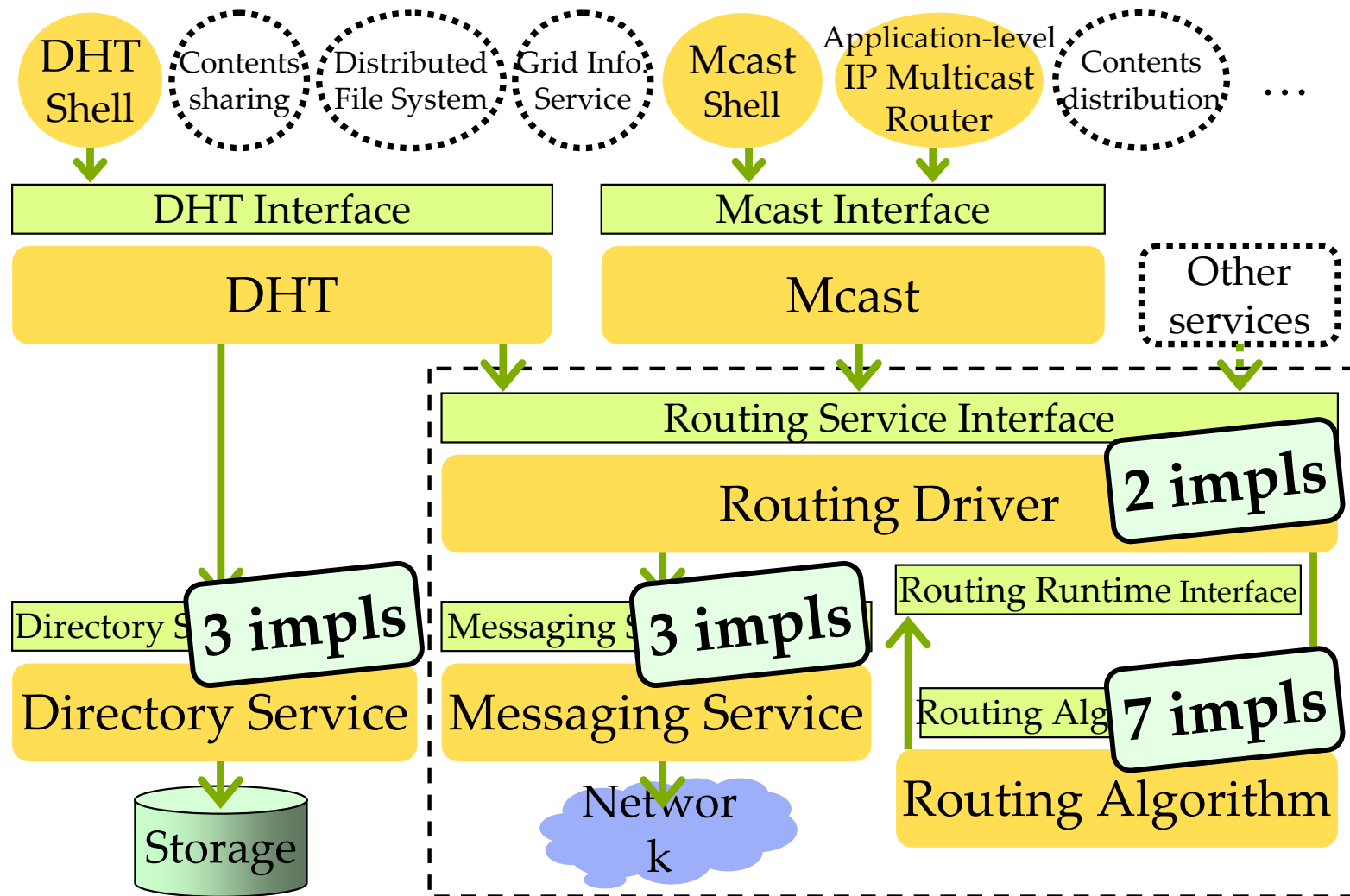
Iterative lookup



Recursive lookup

Components

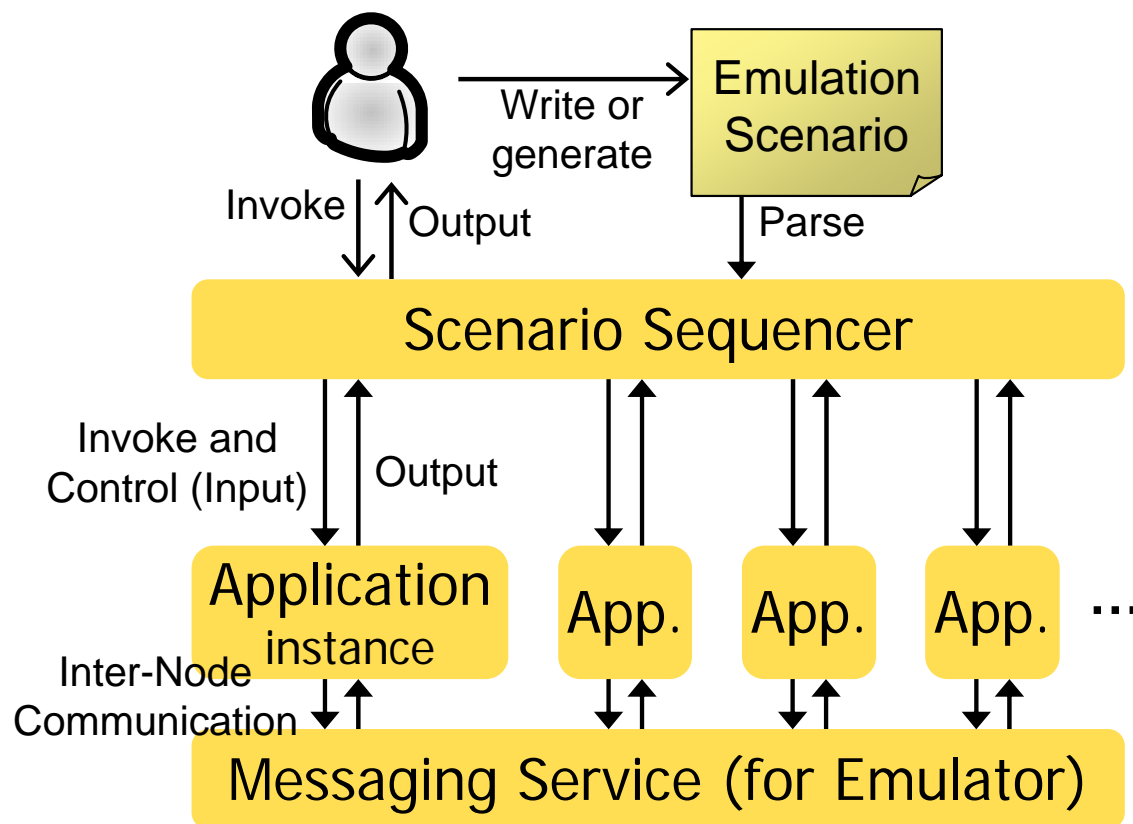
- Multiple implementations for each component, can be combined flexibly.



- Routing Driver
 - Iterative
 - Recursive
- Messaging
 - UDP
 - TCP
 - Emulator
- Directory Service
 - On-memory
 - checkpointing
 - Berkeley DB

Distributed Env Emulator

- It reads a scenario and invoke and control multiple application instances.
- Very useful for test and debug of algorithms.
- Invokes threads for each instances.

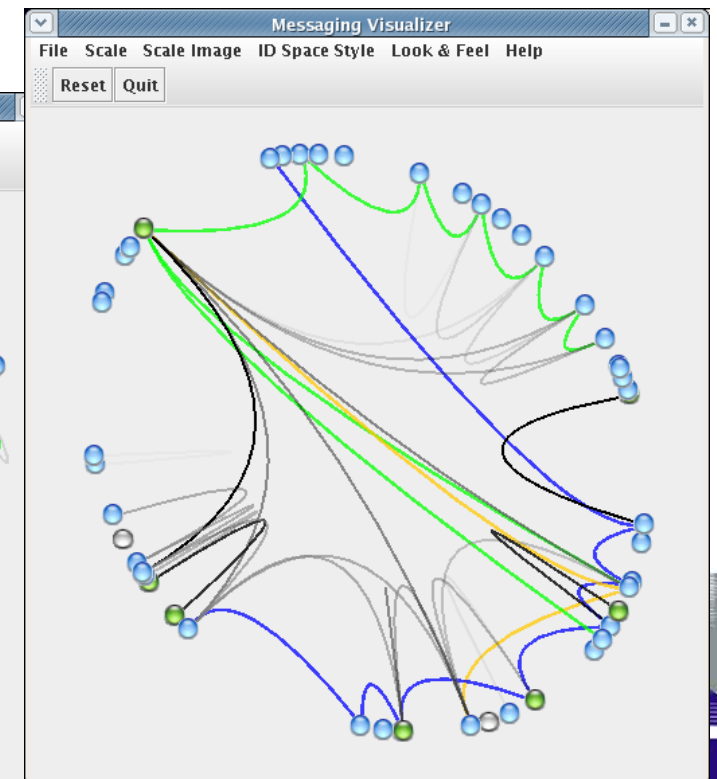
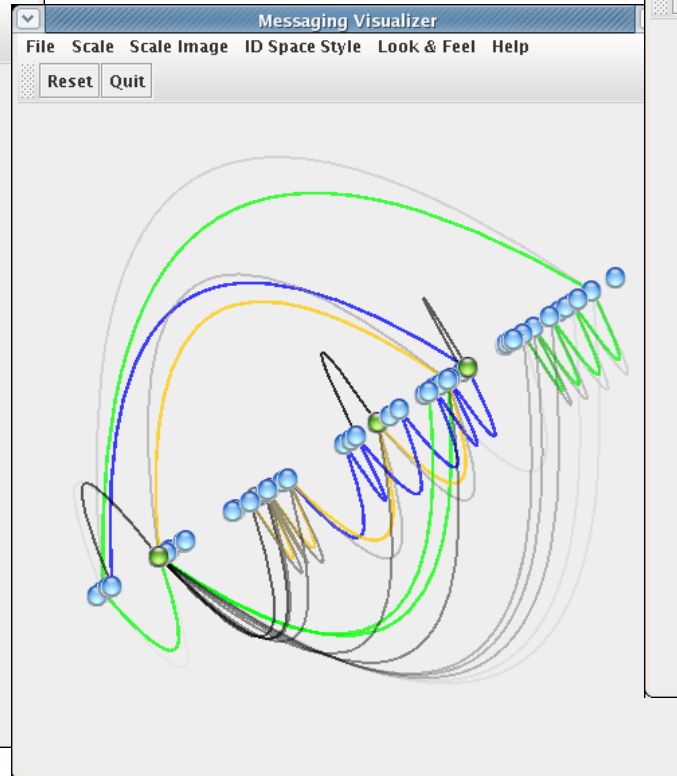
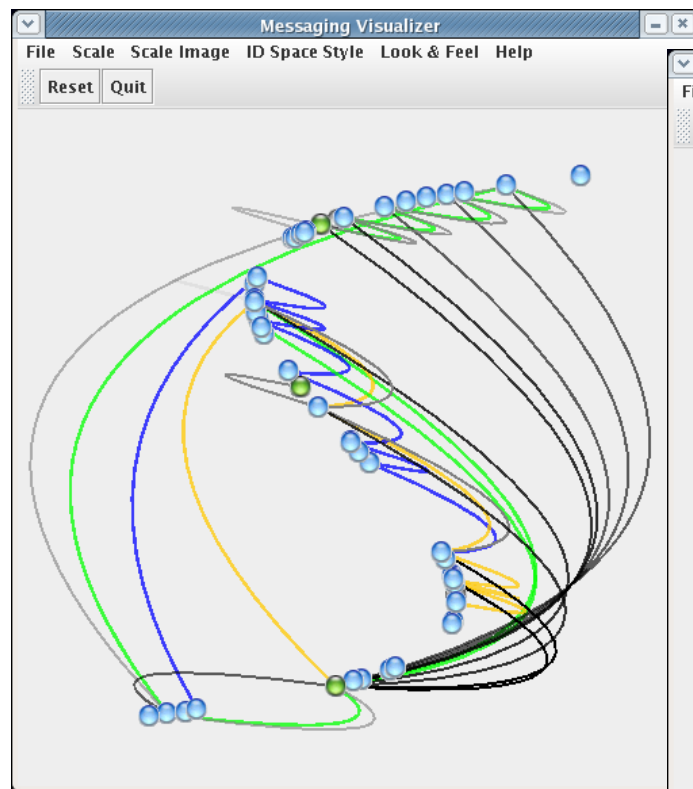


```
# i nvoke 1000 i nstances
class ow. tool . dhtshell . Mai n
arg -p 10000
schedule 0 i nvoke
arg ptp00. hpcc. jp
schedule 500, 500, 999 i nvoke
# put & get
schedule 510000 123 put foo bar 300
schedule 515000 234 get foo
```

A scenario

Overlay Visualizer

- Visualizes nodes, messages and delivery trees for ALM at runtime.
- Reports of messages are also delivered via Messaging Service.
 - -> Visualizer works both on an Emulator and a real network.

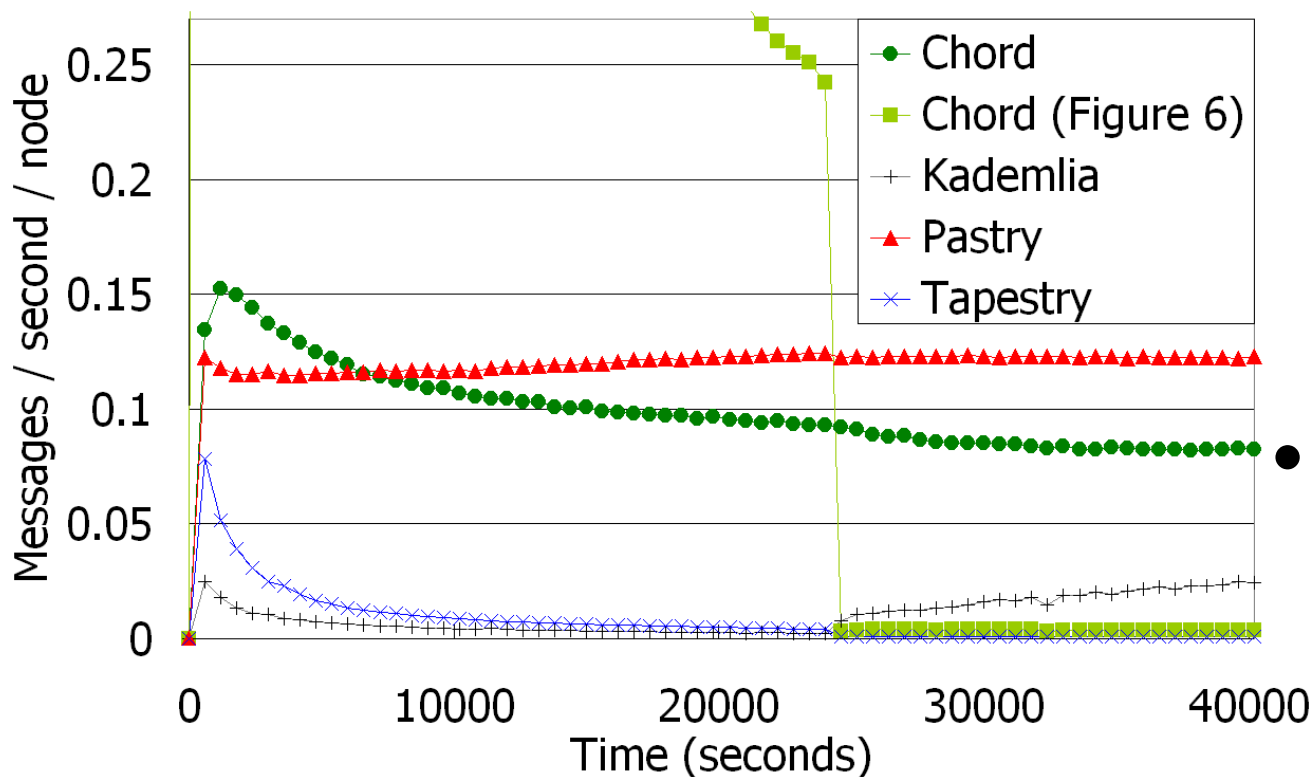


Evaluation

- Goals of Overlay Weaver
 - Connects algorithm research to applications directly.
 - Facilitates algorithm implementation.
 - Enables fair and realistic comparison between algorithm implementations.
- Goals broken down
 - Algorithm impl facilitated →
 - Supports test of algo impls with much nodes →
 - Enables fair comparison →
 - Runs on a real network →
- Means of confirmation
 - The amount of code (formerly shown)
 - 4,000 nodes emulation
 - 200 nodes experiments on a PC cluster

4,000 nodes emulation

- Possible.
- # of messages measured.
 - Scenario: put x 4,000, get x 4,000

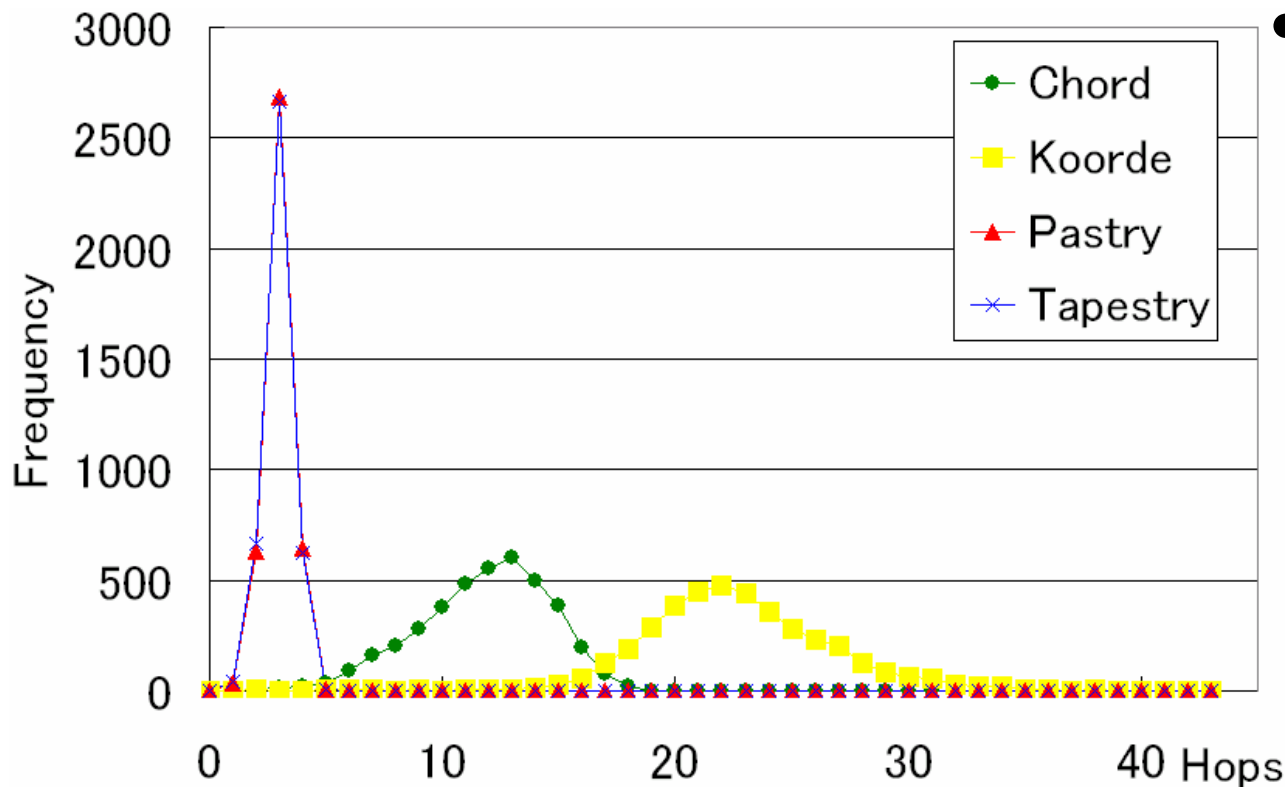


- Conditions – an commodity PC
 - 3.4 GHz Pentium 4
 - 1 GB RAM
 - Linux 2.6.15
 - Java 2 SE 5.0 Update 6

- Note
 - “Chord (Figure 6)” completes its routing table when joining.
 - Iterative routing.

4,000 nodes emulation

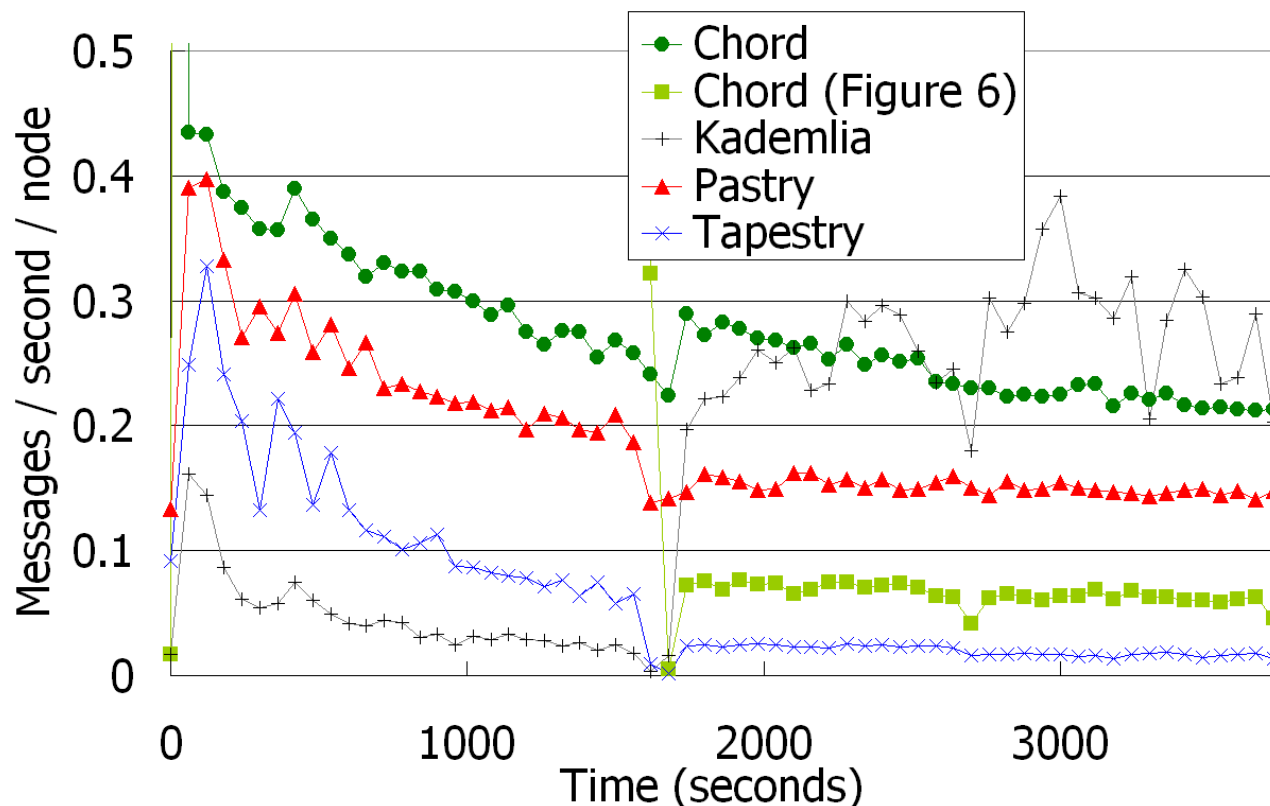
- # of hops in lookups measured.
Frequency of each # plotted.
 - Scenario: put x 4,000, get x 4,000



- Conditions – an commodity PC
 - 1.7 GHz Pentium M
 - 1 GB RAM
 - Linux 2.6.15
 - Java 2 SE 5.0 Update 6

200 nodes PC cluster

- Possible
- # of messages measured.
 - Scenario: put x 500, get x 500



- Conditions – 197 PCs in AIST Super Cluster
 - 3.06 GHz Xeon
 - Linux 2.4.24
 - Java 2 SE 5.0 Update 6
- Note
 - Graph sways because measurement interval is shorter.
 - Emu: 10 min
 - PC cluster: 1 min
 - Iterative routing.

Summary

- Overlay Weaver is an overlay construction toolkit ...
 - Connects algorithm research to applications directly.
 - Rapid development and tests with Emulator and the resulted implementation runs on a real network.
 - Facilitates algorithm implementation.
 - Just in 100s steps of code.
 - Impls of well-known 5 algorithm are provided.
 - Enables fair and realistic comparison between algorithm implementations.