Seeking the Source

Software Source Code as a Social and Technical Artifact

Cleidson de Souza, ***Jon Froehlich***, and Paul Dourish

Overview

- Uncover structures of software projects
  - Uniform treatment of artifacts and activities
  - Use code analysis and development metadata to derive views of social structure
- Qualitative study of open source software

Background

- Software architecture defines a relationship between components
  - Functional/technical influences
  - Social and organizational influences also at play
- Conway [1968]
  - "The structure of a system mirrors the structure of the organization that designed it"
- Also, Parnas' [1972]
  - Principle of information hiding

Background Continued

- What happens when there is no “formal” organization
  - OSS might be an example…
- We created a tool to investigate:
  - how the relationships between software modules expose one view of the underlying social structure

Outline

- Augur, A Tool
- Visualizing Software Development
- Evolutionary Patterns
- Conclusion & Future Work

Augur, A Tool

Augur

- Visualization system for CM repositories of source code
  - Attempts to unify views of “activities” and “artifacts”
  - Contains multiple, dynamic views of data

Augur's Data Source

- Configuration Management (CM) Systems
  - Repository for both source code and meta-data about development
    - Author information
    - Checkin information
    - Use this as a basis for analysis
  - Augur connects to existing CVS servers with no additional overhead


Augur: Viewing jung from :pserver:anonymous@cvs.sourceforge.net:/cvsroot/jung
Display By: Age
Indented:
Size:
Time:
Past 2 months
05/06/04
0.6 weeks
1.1 weeks
1.7 weeks
2.3 weeks
2.9 weeks
3.4 weeks
1 month
4.6 weeks
5.1 weeks
5.7 weeks
6.3 weeks
6.9 weeks
7.4 weeks
2 months
jung
src
Sun.
Mon.
Tue.
Wed.
Thu.
Friday
Sat.
12 AM
6 AM
12 PM
6 PM
jung/
Lines of Code
60,000
50,000
40,000
30,000
20,000
10,000
0
4/1/03
7/1/03
10/1/03
1/1/04
4/1/04
Time of Checkin
Num Lines Added by All Authors
439
0
03/31/04
04/21/04
Display:
Num Lines Added
Author:
All Authors
Num Check-ins by All Authors

Simple Analysis

- Analyze which developers in the system work on which files or lines
  - Broad temporal patterns
    - e.g. clear distinctions between workday and weekend, individual contribution patterns, etc.
  - Broad collaborative patterns
    - e.g. certain authors only work alone, certain files contain many authors

Static Analysis

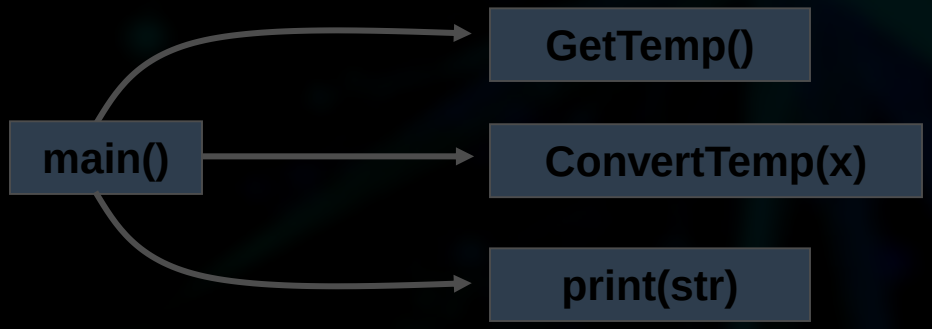
- Static analysis enables us to explore richer, more meaningful relationships
  - Line types
  - Containment structures
- Dependency relationships begin to emerge
  - Object extension
  - Interface implementation and extension

Dependency Analysis

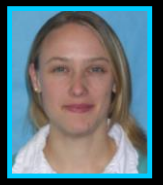
- Augur extended to conduct Call Graph Analysis
  - A window into the dynamic structures of the code
    - As Parnas and Conway suggest, this structure influences the coordination of the work

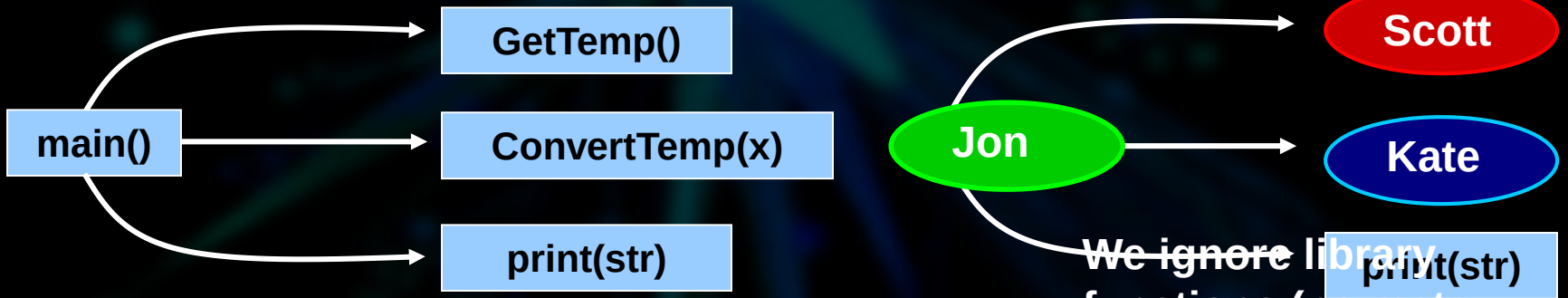
```
int main(){
   int temp = GetTemp();
   double fahr = ConvertTemp(temp);
   print("Temperature: " + fahr);
}
```

An Example



```
int main(){
   int temp = GetTemp();
   double fahr = ConvertTemp(temp);
   print("Temperature: " + fahr);
}
```





~~We ignore~~ library functions (no meta data available)

Visualizing Software Development

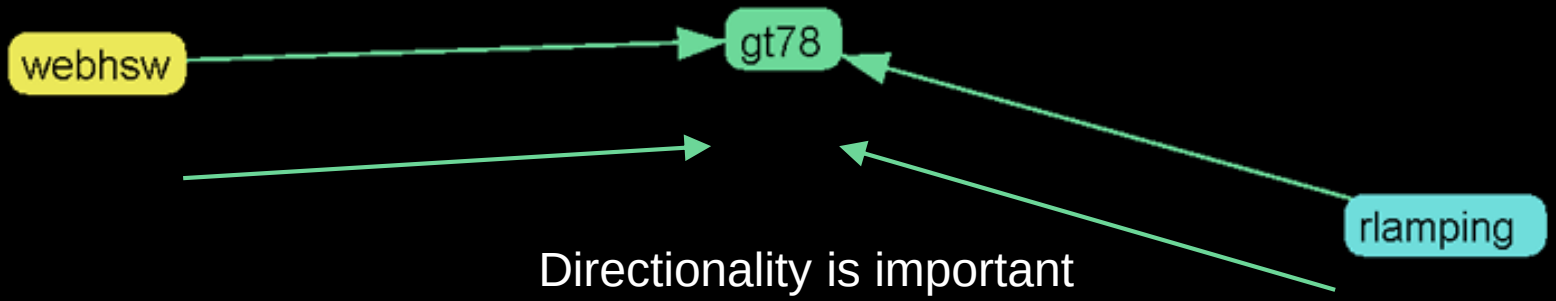
Types of Projects

- Network relationships between developers
  - As derived from code dependencies
- Many different types, we'll focus on three emergent patterns
  - Centralized
  - Densely Networked
  - Core and Periphery

Centralized

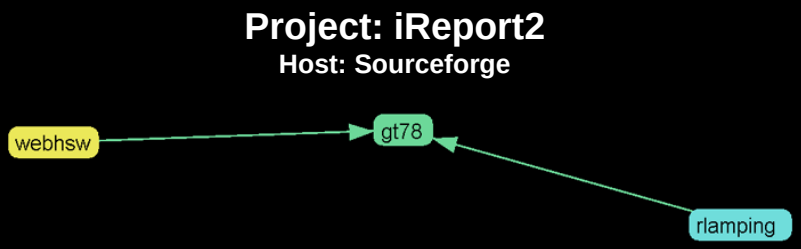
Project: iReport2

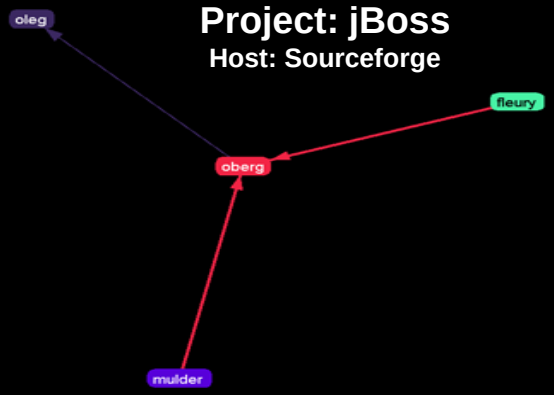
Host: Sourceforge

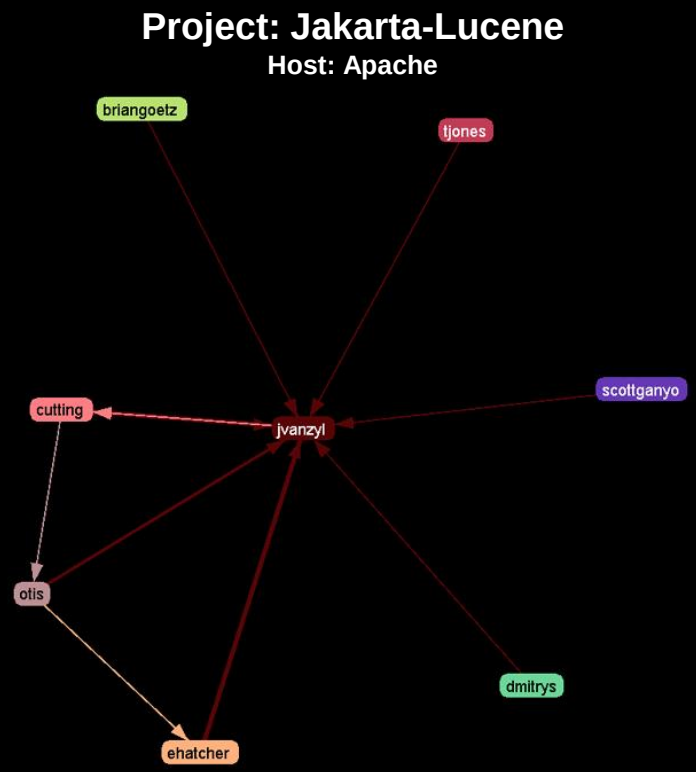


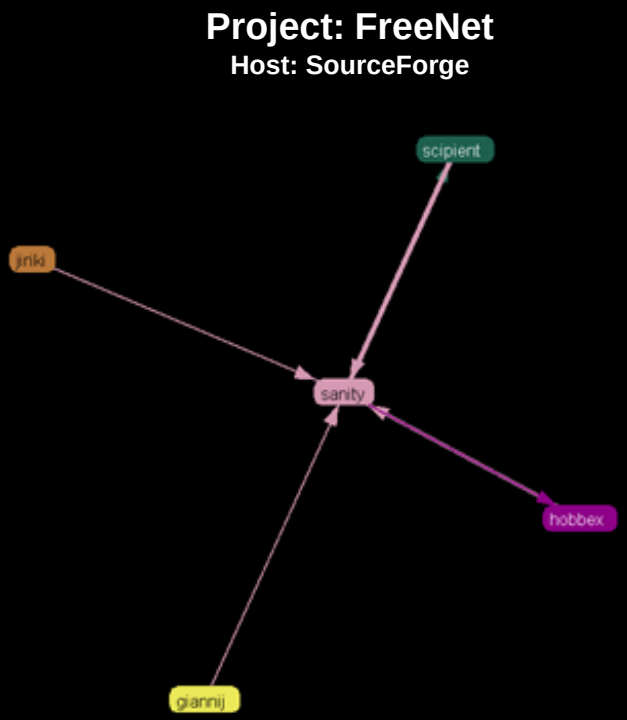
Directionality is important

Centralized





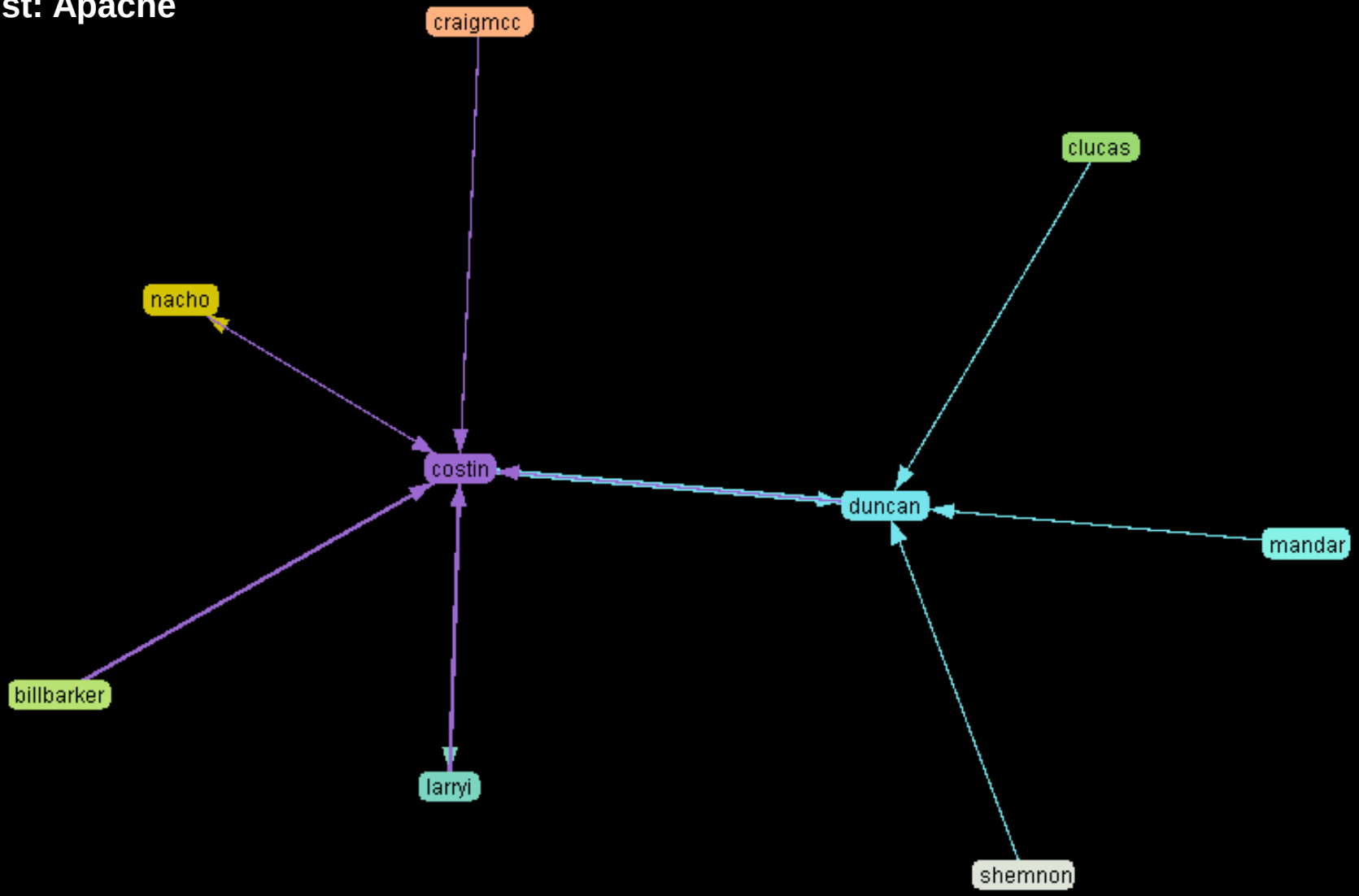




Centralized

Project: Jakarta-Tomcat

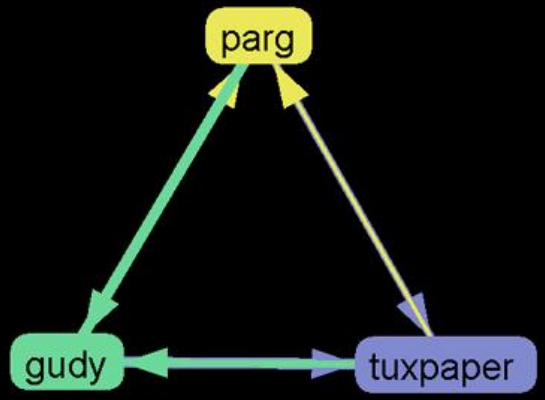
Host: Apache



Densely Networked

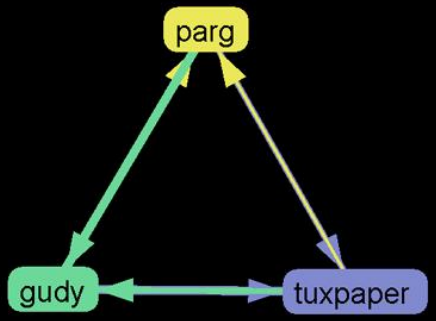
Project: Azureus

Host: Sourceforge

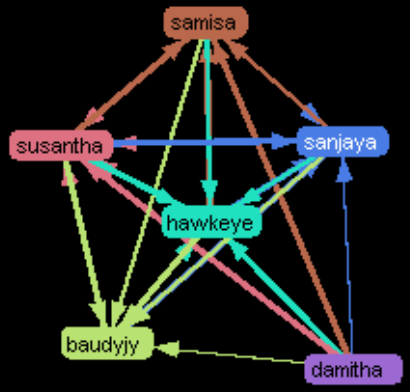


Densely Networked

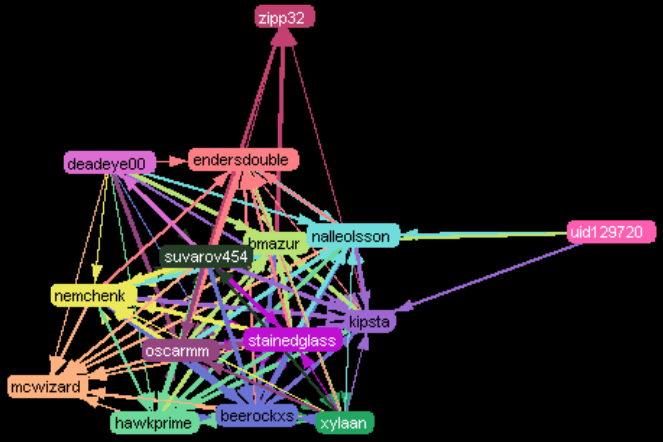
Project: Azureus
Host: Sourceforge



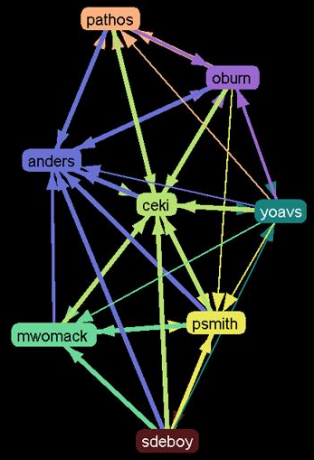
Project: Apache WS-AXIS
Host: Apache



Project: Megamek
Host: Sourceforge

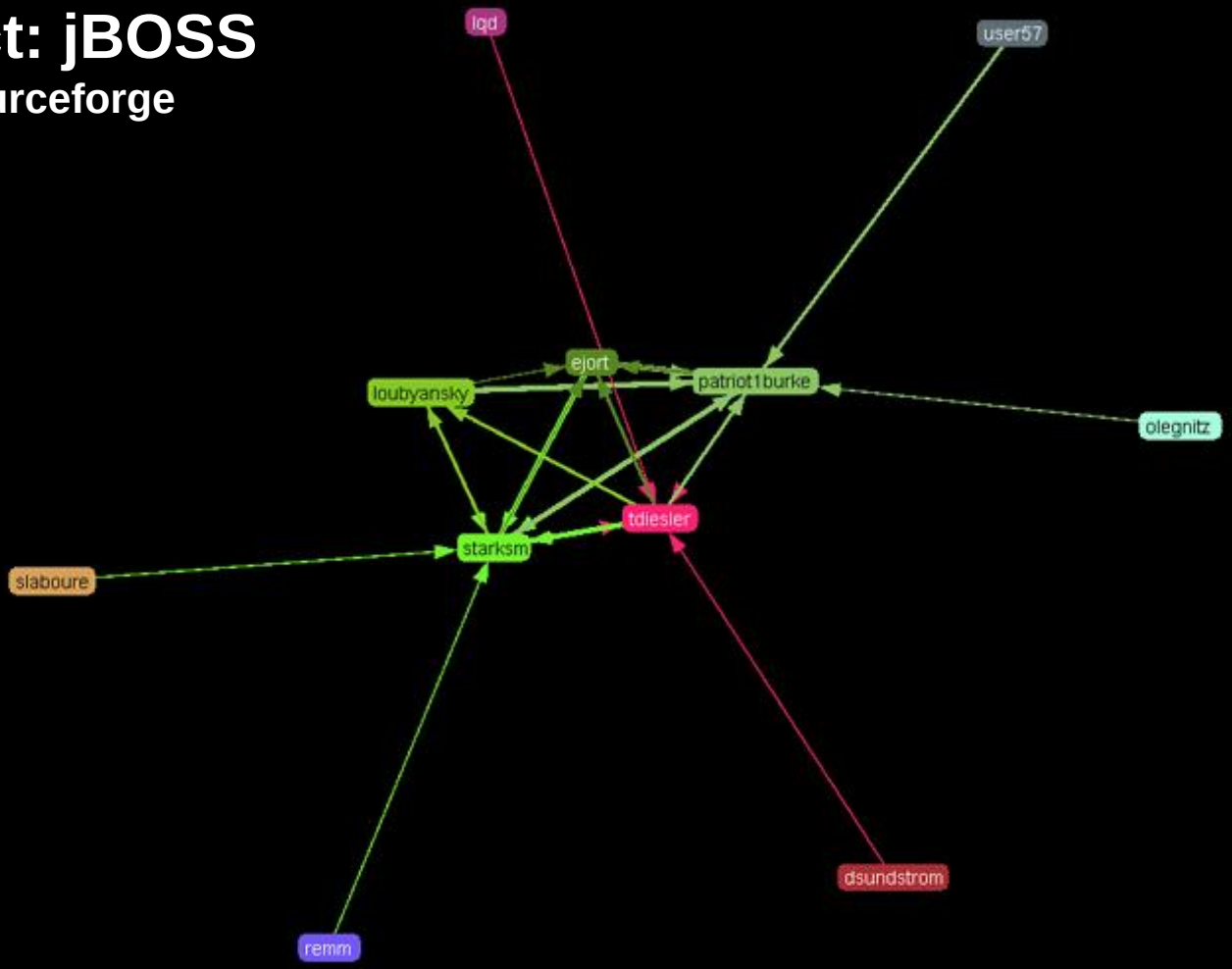


Project: Apache-Log4j
Host: Apache



Core & Periphery

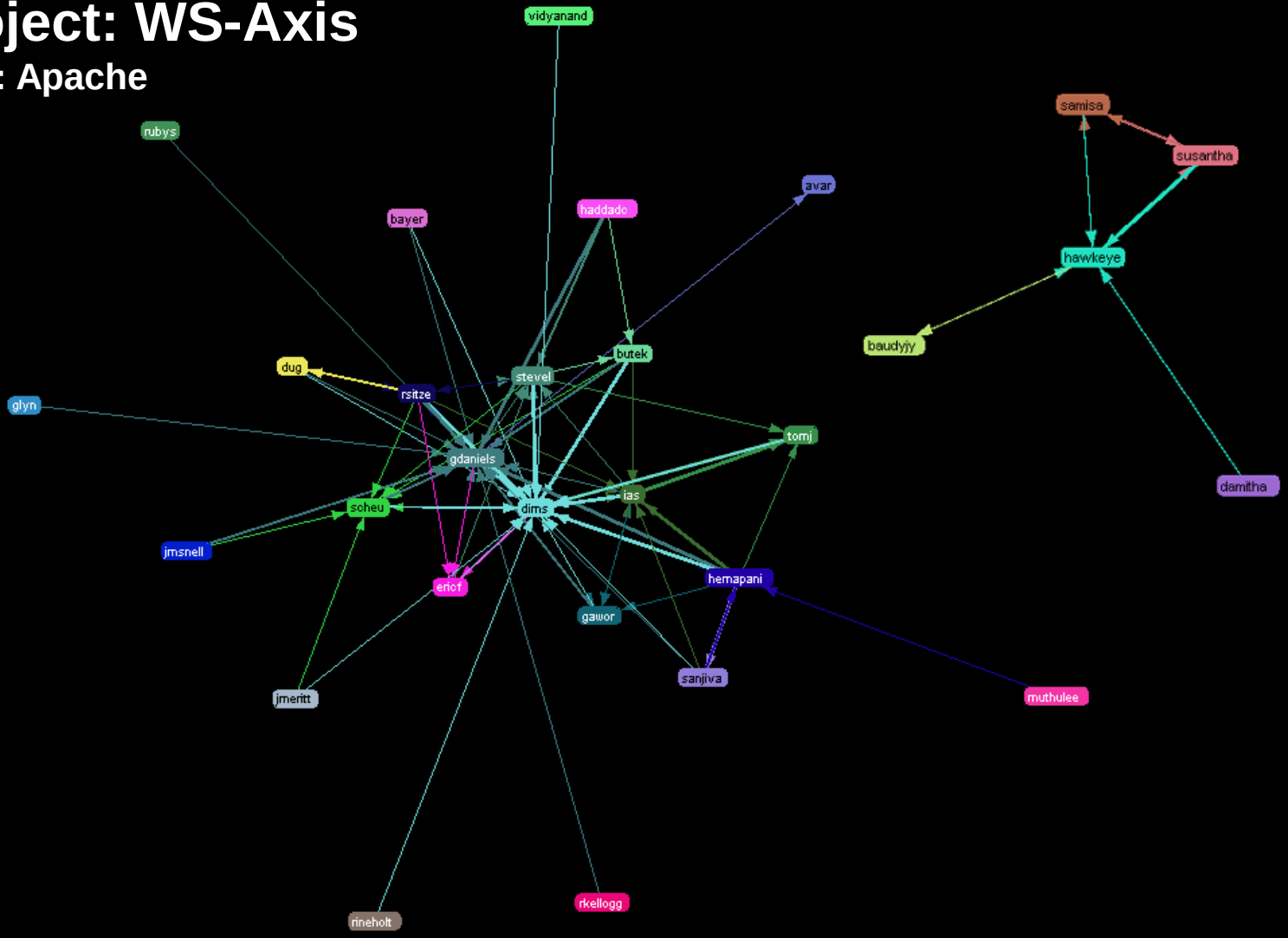
Project: jBOSS
Host: Sourceforge

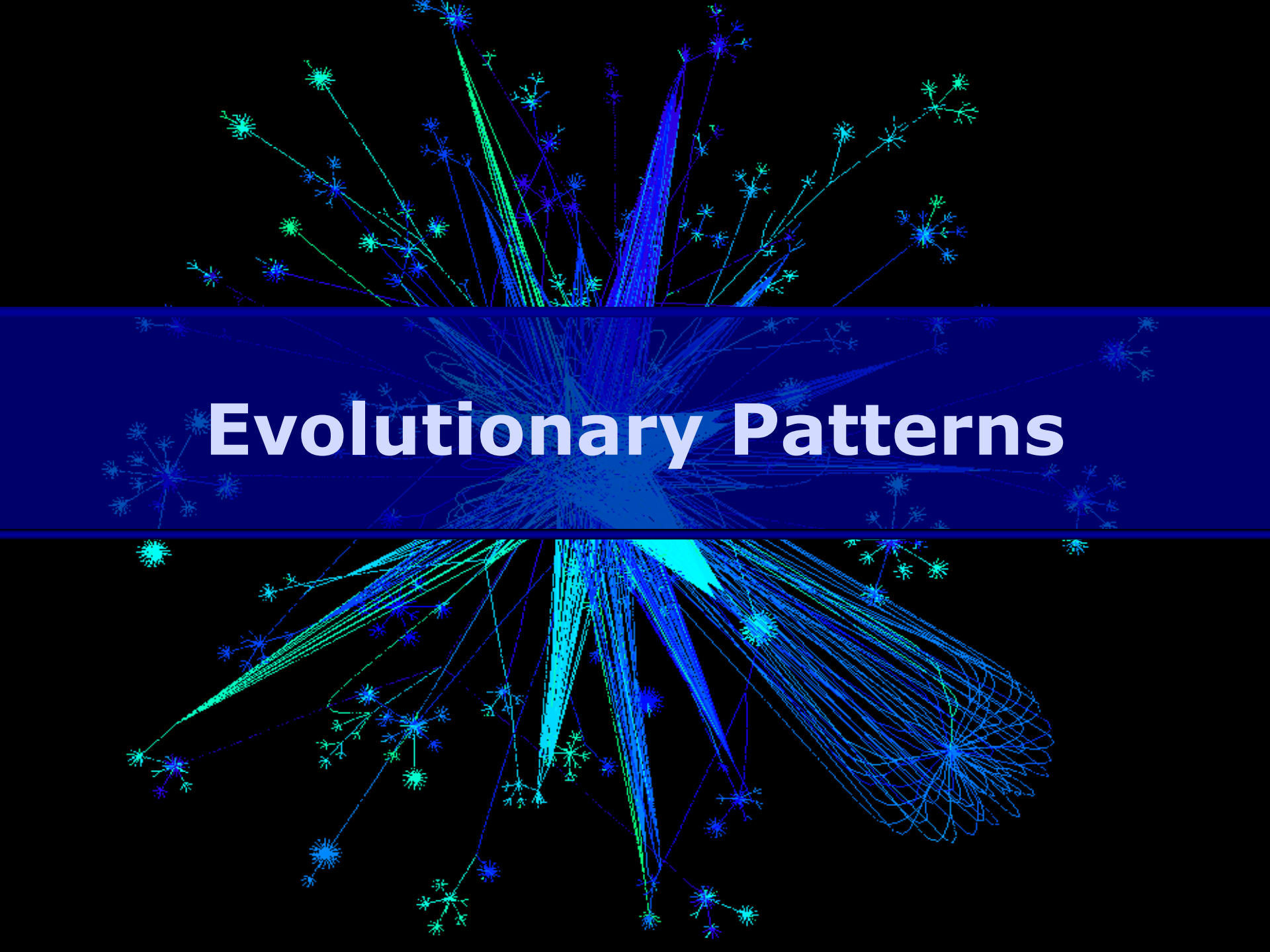


Core & Periphery

Project: WS-Axis

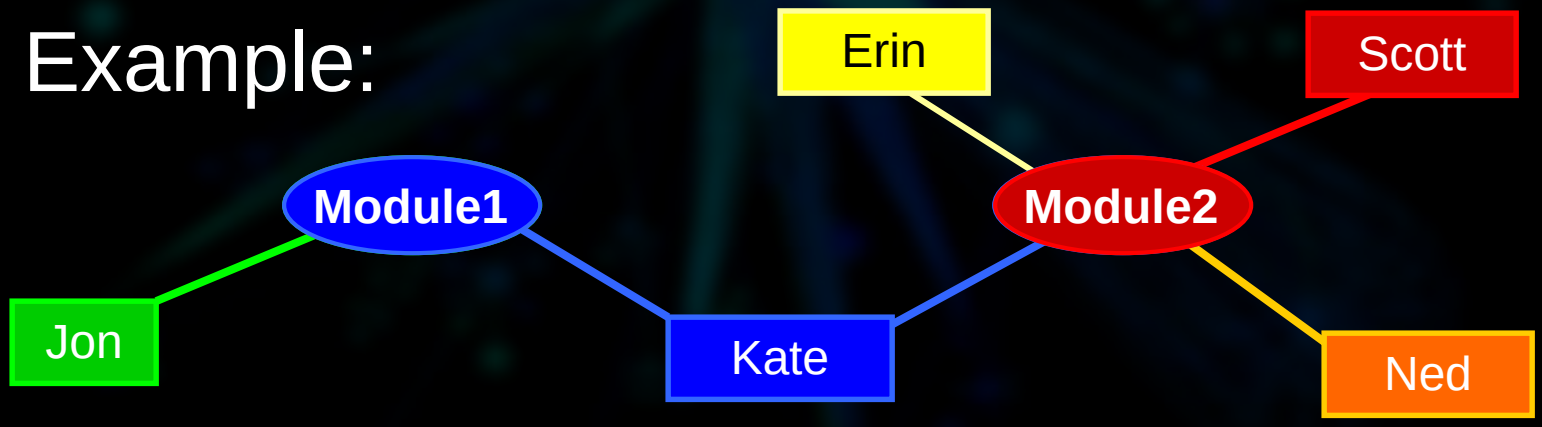
Host: Apache

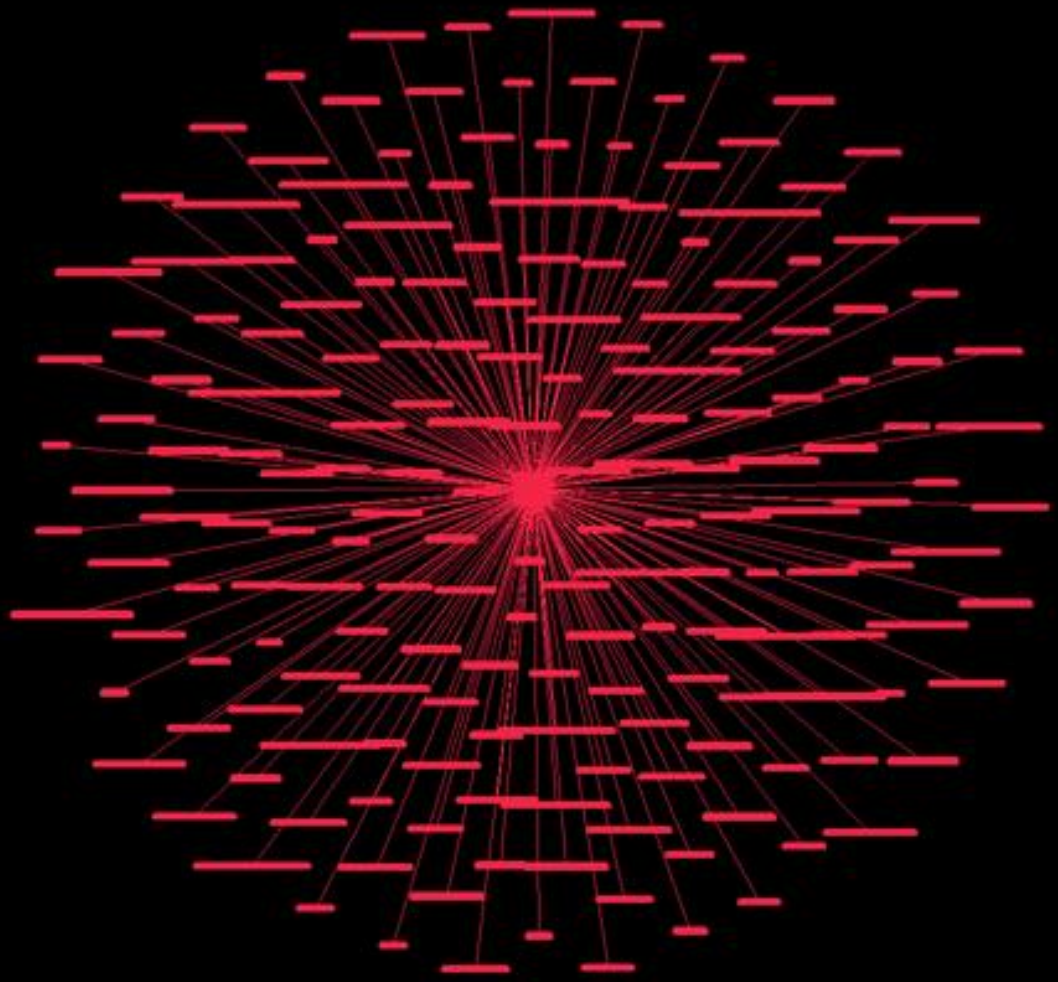



Evolutionary Patterns

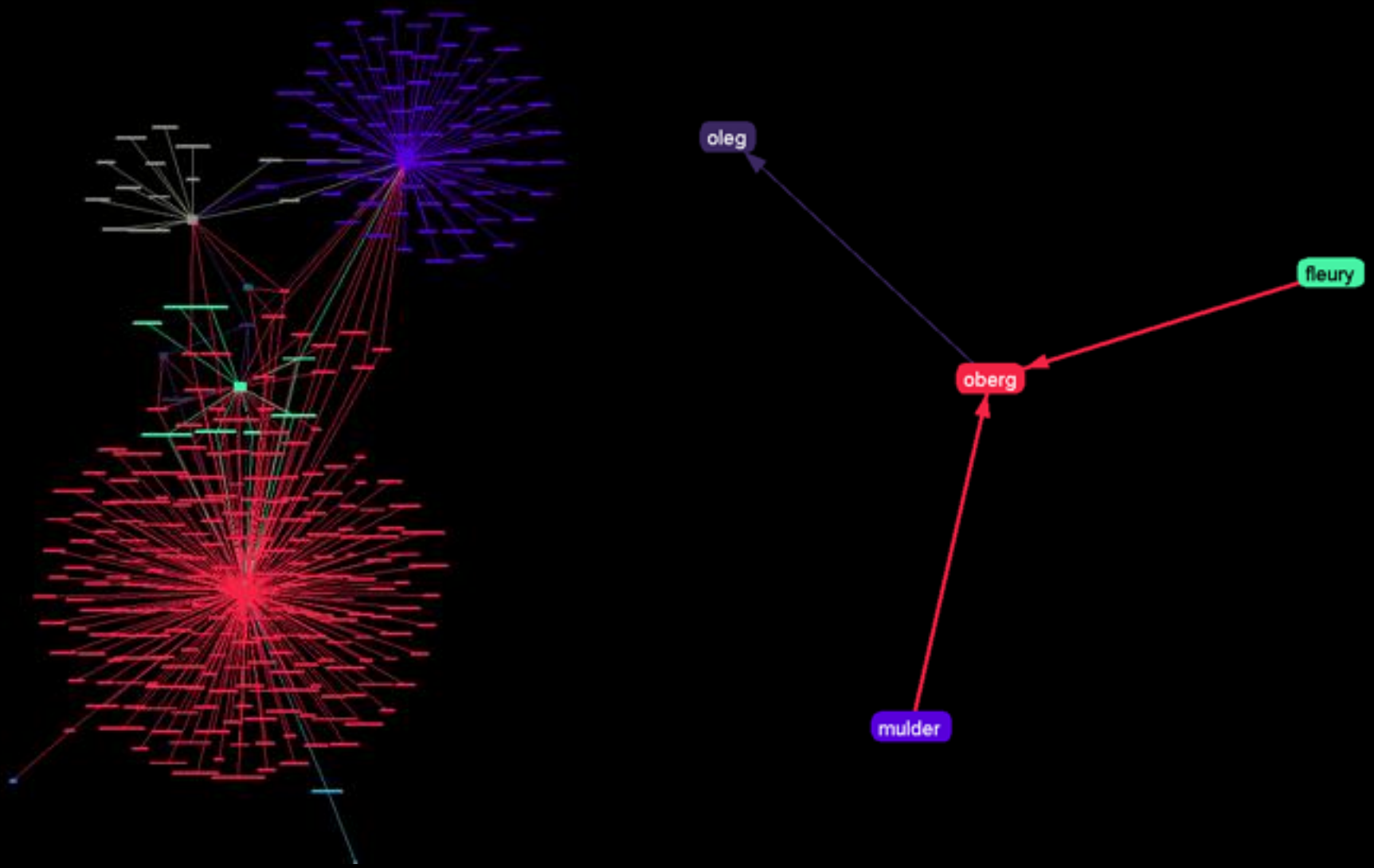
Bipartite Graphs

- Quick digression…
- Useful to see both artifact *and* author in the same frame
  - The connection between the two becomes explicit – Bipartite Graphs
- Example:

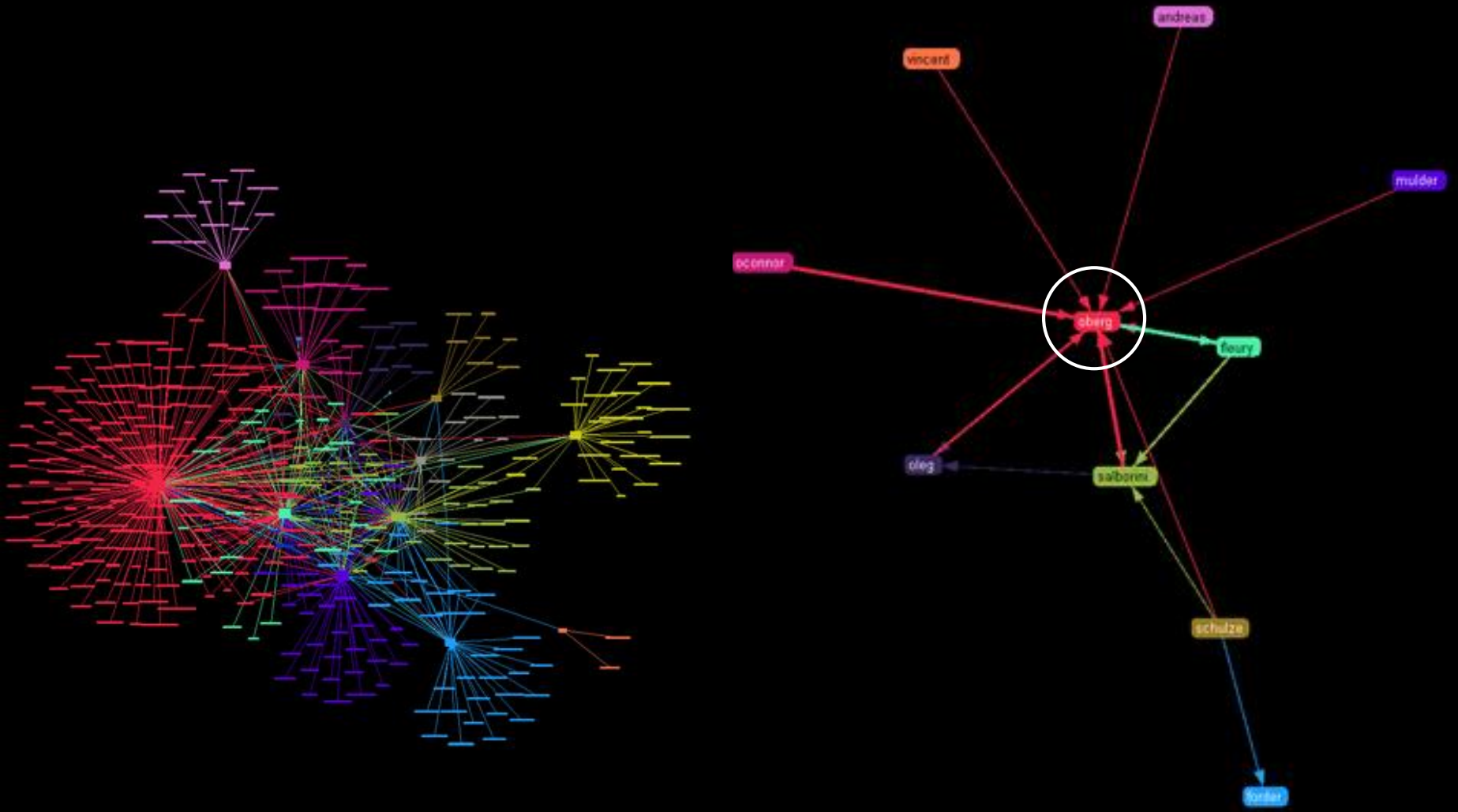




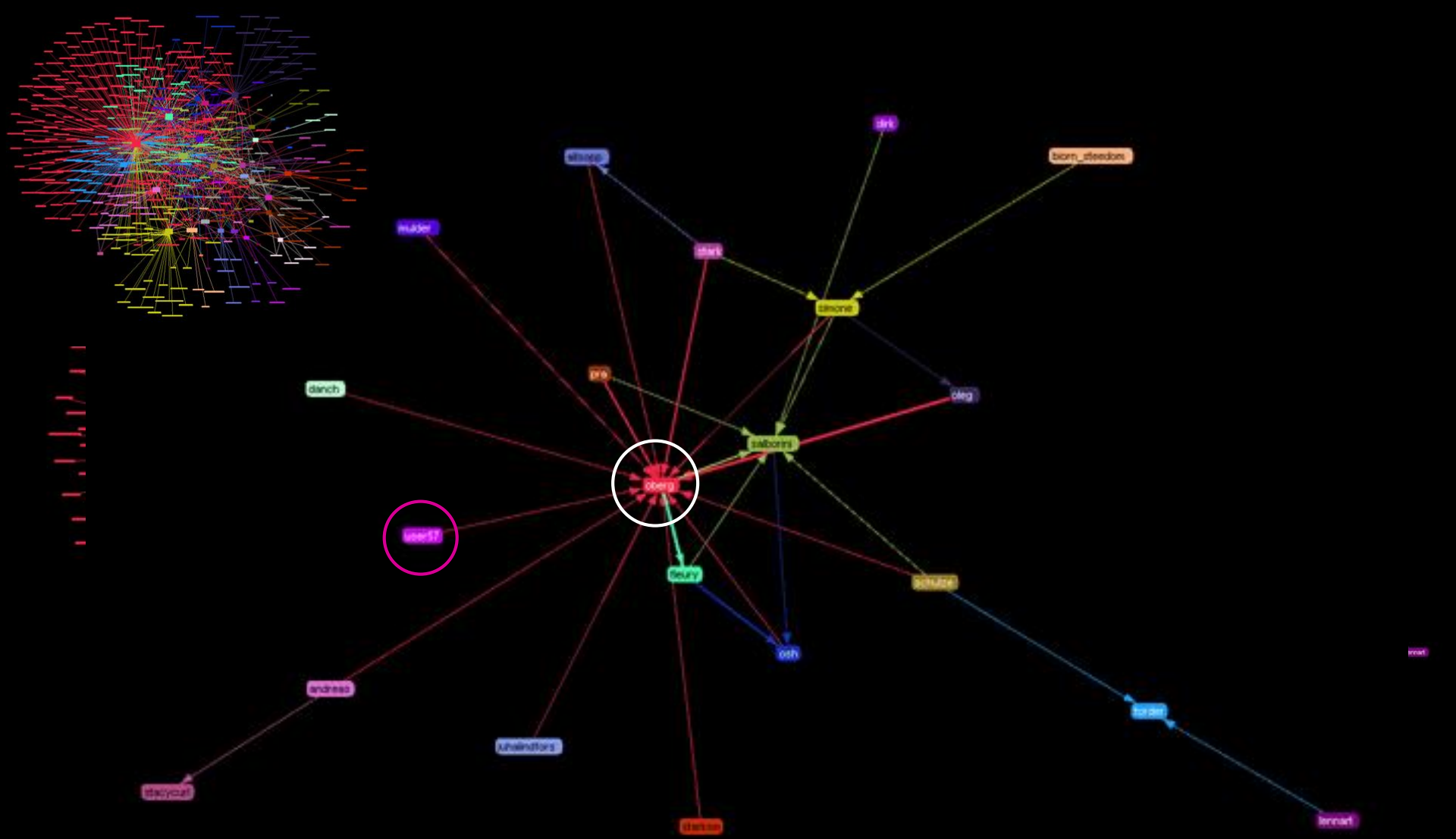
04-23-2000
One Day Old



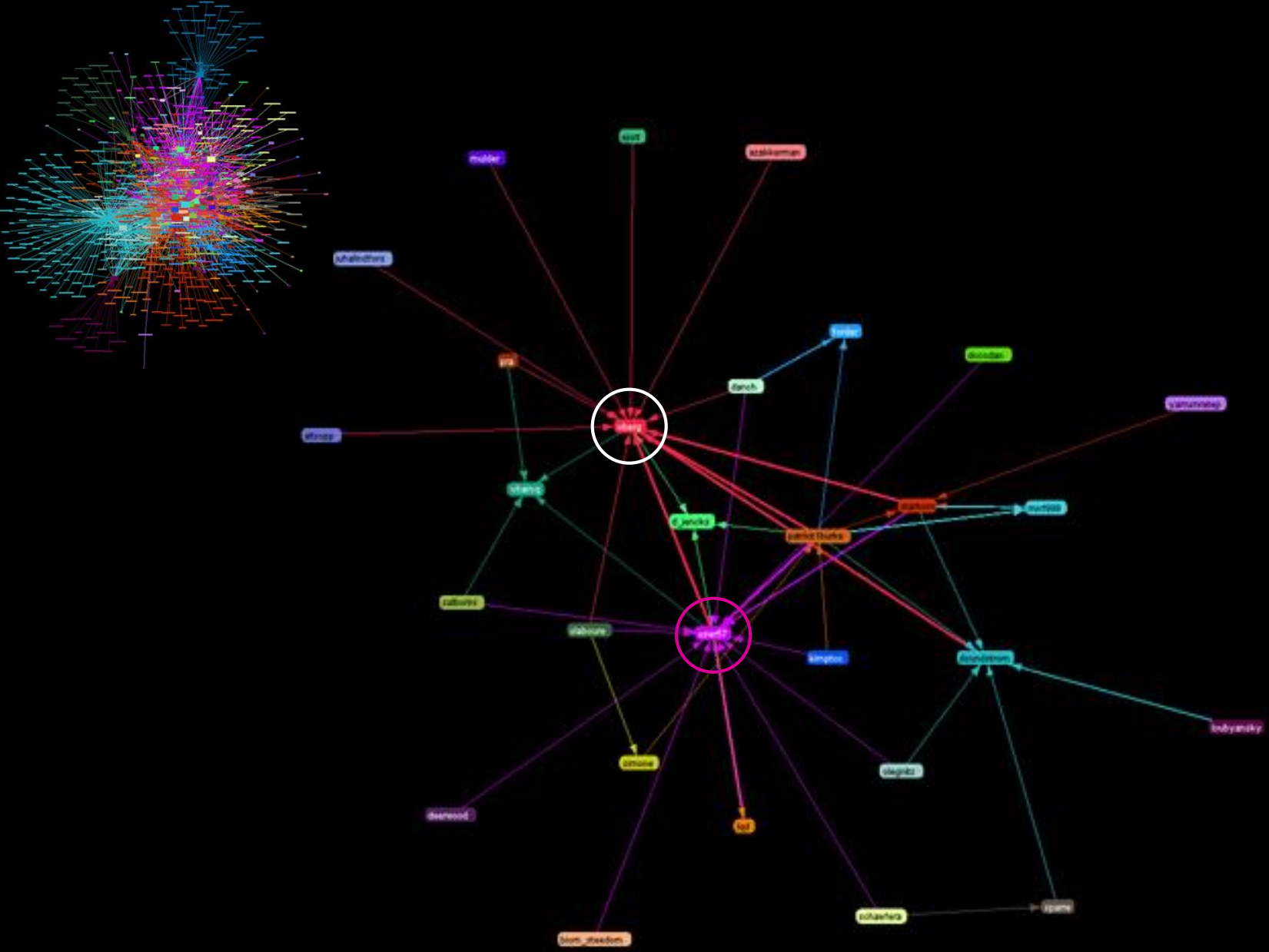
07-23-2000
Three Months Old



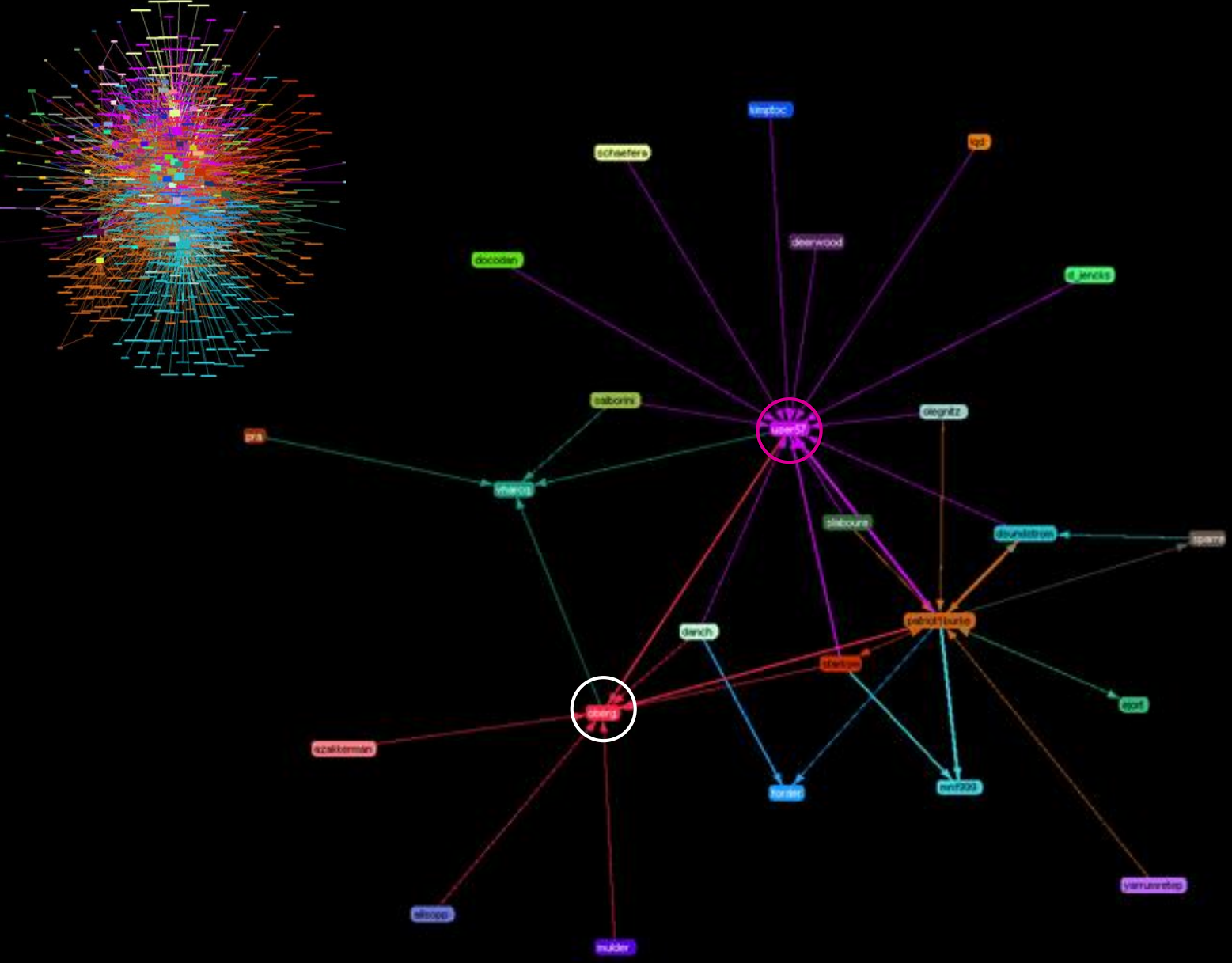
10-23-2000
Six Months Old



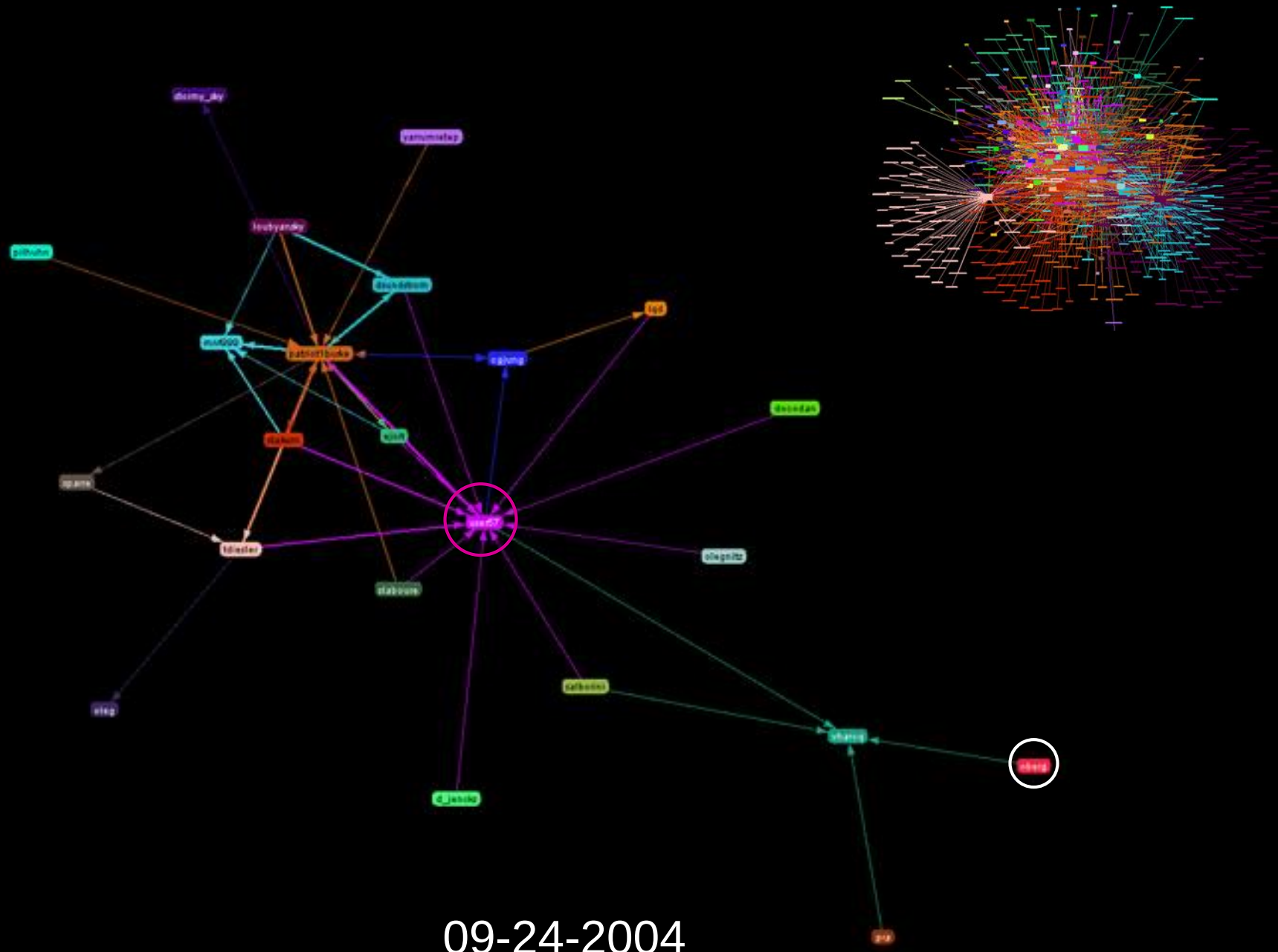
04-23-2001
One Year Old



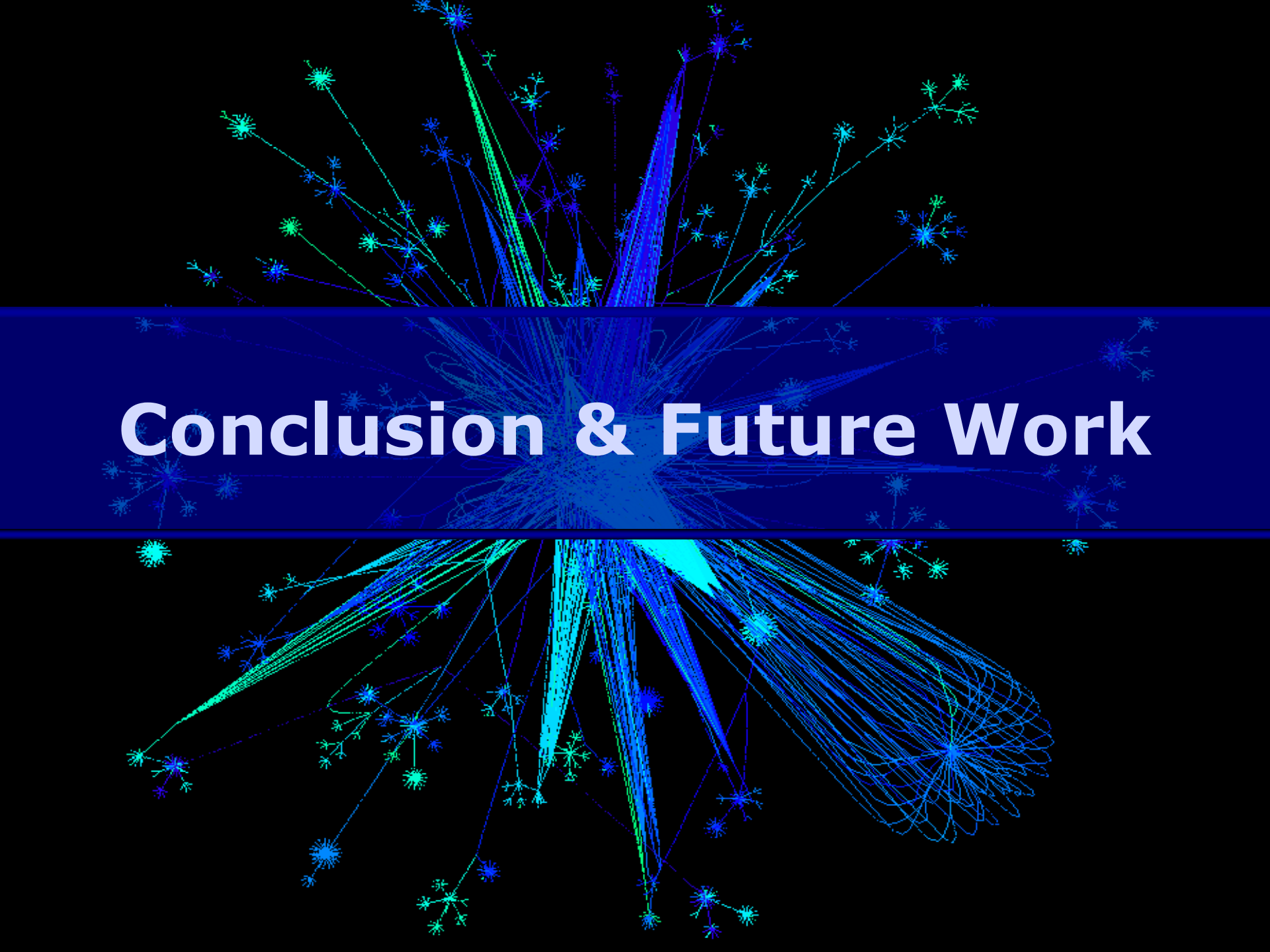
10-23-2002
2.5 Years Old



10-23-2003
3.5 Years Old



09-24-2004
~4.5 Years Old



Conclusion & Future Work

Conclusion

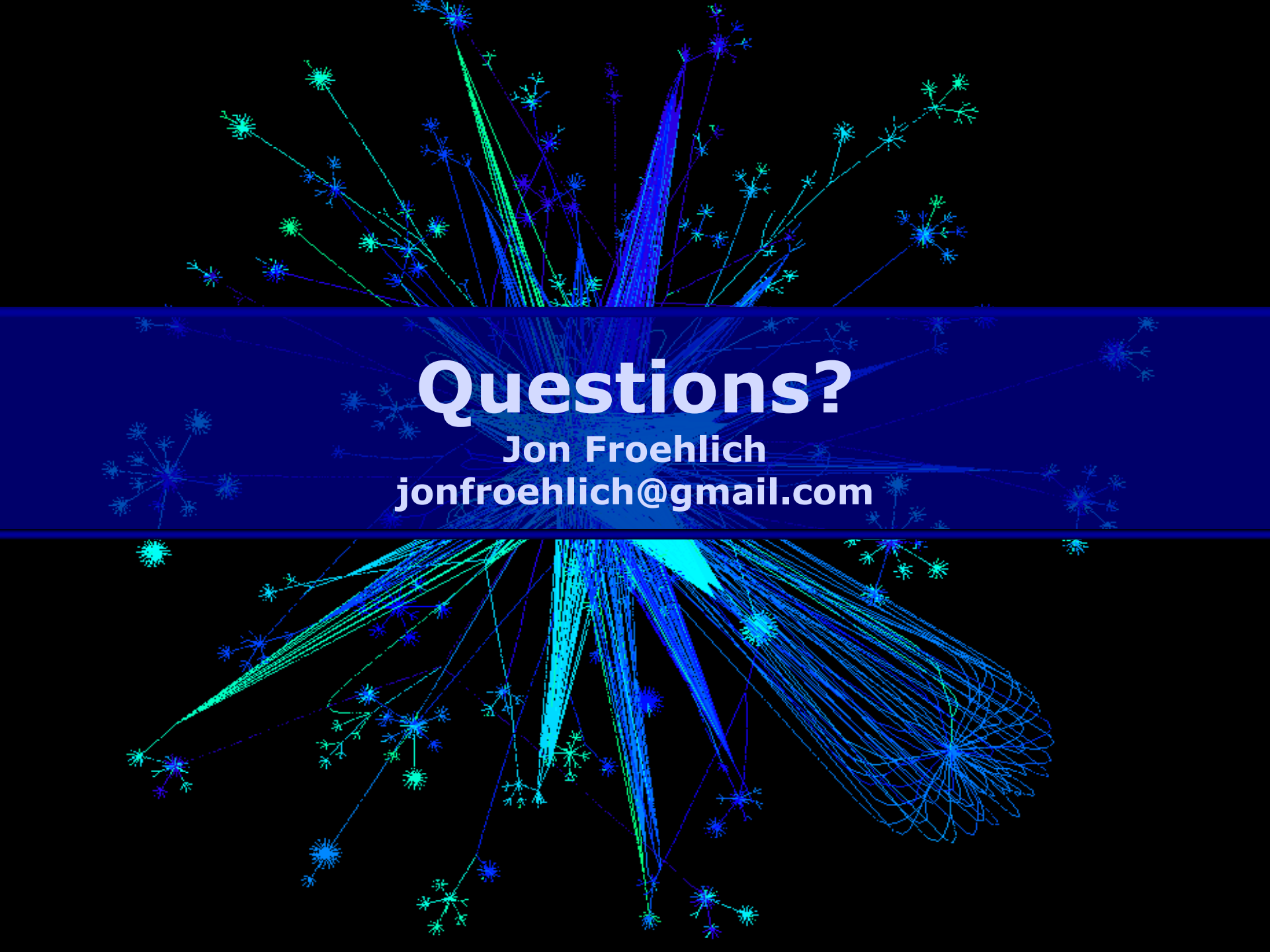
- We explored emergent social and technical structures through artifacts
- Our preliminary empirical examinations have shown that:
  - Social patterns can be revealed through CM analysis combined with artifact analysis
    - E.g. Certain modules and certain “authors” become obligatory passage points in the system
  - Suggest that tools can be built to support both technical and social structures in the system

Future Work

- Deeper investigations…
  - Interviews with project developers (compare/contrast)
- Automatic recognition…
  - Similar to Soylent e-mail system [Fisher 2004]
- Is Augur a useful tool to OSS research?
  - Is there value to the community?
- Investigate different communities
  - Commercial systems
  - Co-located development teams
- Integrate other sources of information
  - E.g. mine bug-tracking systems [Crowston 2004]

Acknowledgements

- My two co-authors:
  - Cleidson de Souza
    - Departmento de Informatica
      Universidad Federal do Para
  - Paul Dourish
    - Donald Bren School of Information and Computer Sciences
      University of California, Irvine
- Jeffrey Heer for Prefuse
- Danyel Fisher for JUNG



Questions?

Jon Froehlich
jonfroehlich@gmail.com