DISTRIBUTED MONOTONICITY

Kevin Cheek
Alex Maskovyak
Joseph Pecoraro

AGENDA

- Accomplishments
- Design
- Demonstration
- Future Work
- Lessons Learned

ACCOMPLISHMENTS

- Successfully built “RAIDS”, Redundant Array of Independent Distributed Storage Systems
  - Client application allows for network connections or multiple nodes per JVM.
- Successfully built “EVE”, our eavesdropping and network visualization server
  - Basic multithreaded server with network access.

RAIDS

- Offers RAID-like data redundancy and error recovery on a distributed system of computers.
- Is an application on FreePastry, an open-source peer-to-peer system framework to handle routing and maintenance chores
- Uses PAST's (a FreePastry app) DHT for storing file list and file division information
- Uses Scribe (a FreePastry app) for multicasting storage requests.

RAIDS FUNCTIONALITY

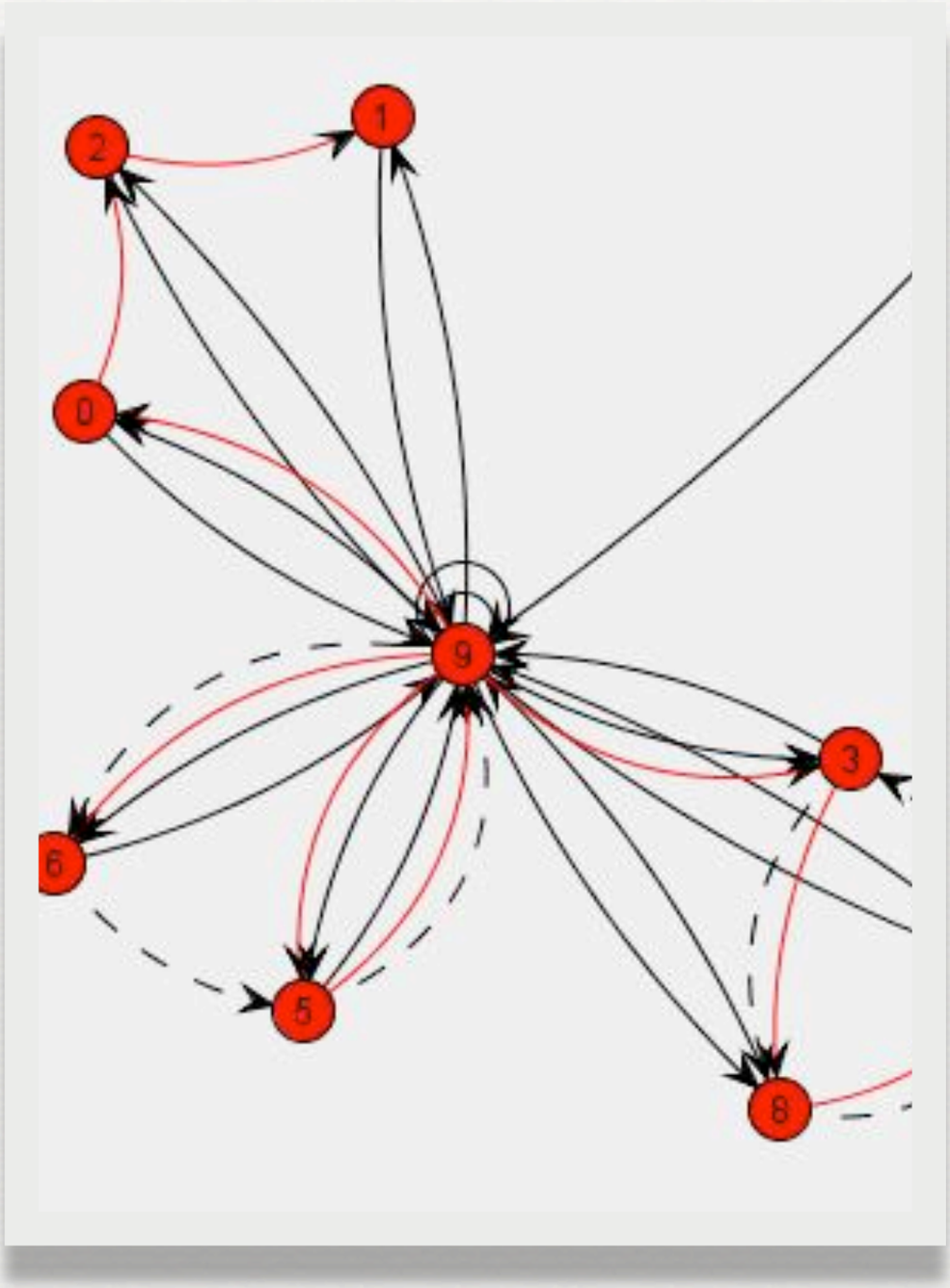
- Offers users a convenient command-line interface
- Chunks and reassembles files for upload and download requests
- Recovers from node failure and redistributes chunk copies to maintain a set replication factor
- Recovers from file corruption on a node

EVE

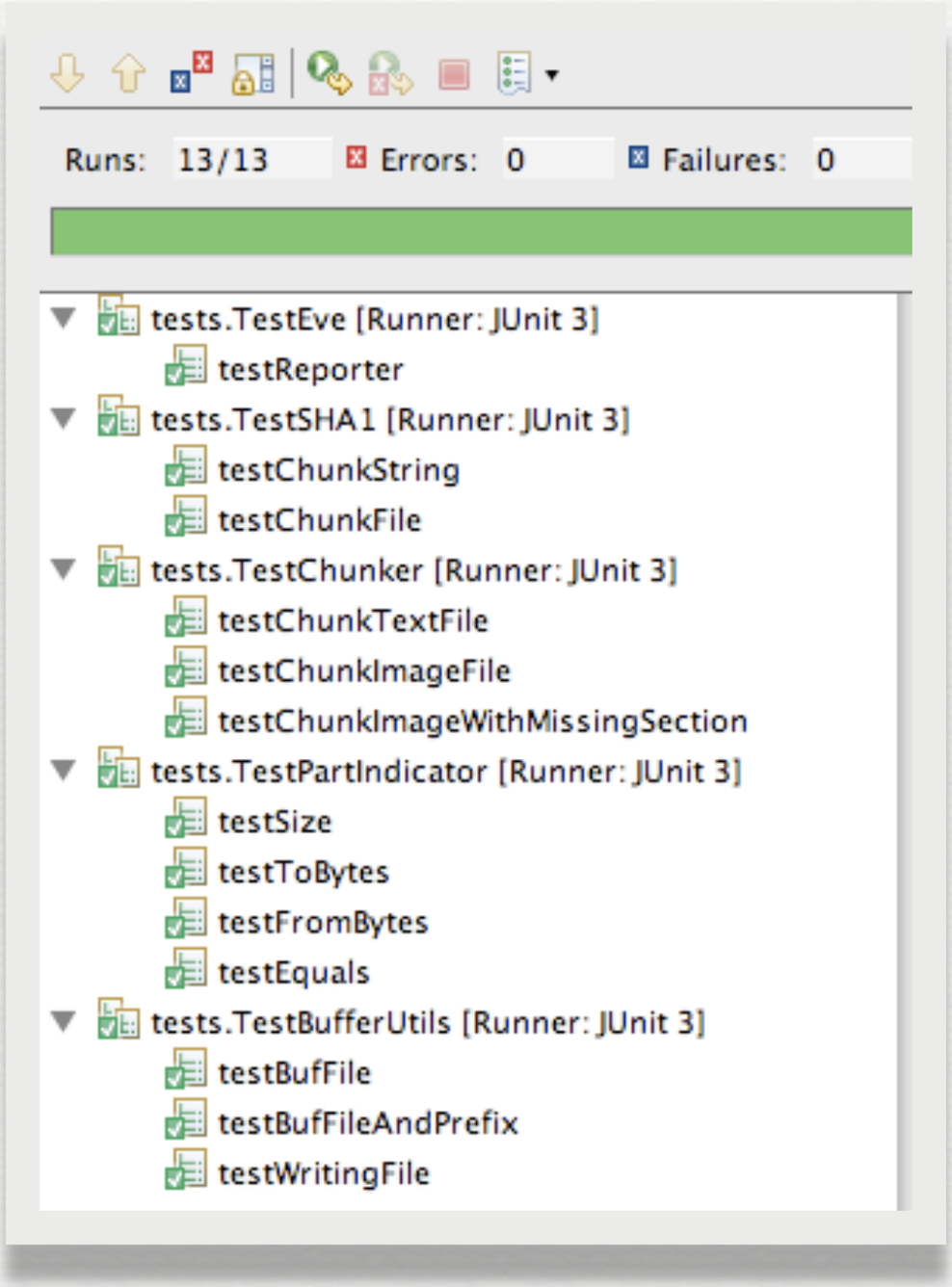
- Optional monitoring server set to listen on a socket.
  - Raw console output of messages.
  - Graph visualization of messages between nodes.
- RAIDS communicates message events to EVE.
  - Identical to logging, but with extra functionality.

EVE FUNCTIONALITY

- Provides a real-time visual of the network and message activity.
- Visualize Messages, Heartbeats, Uploads, and Downloads.



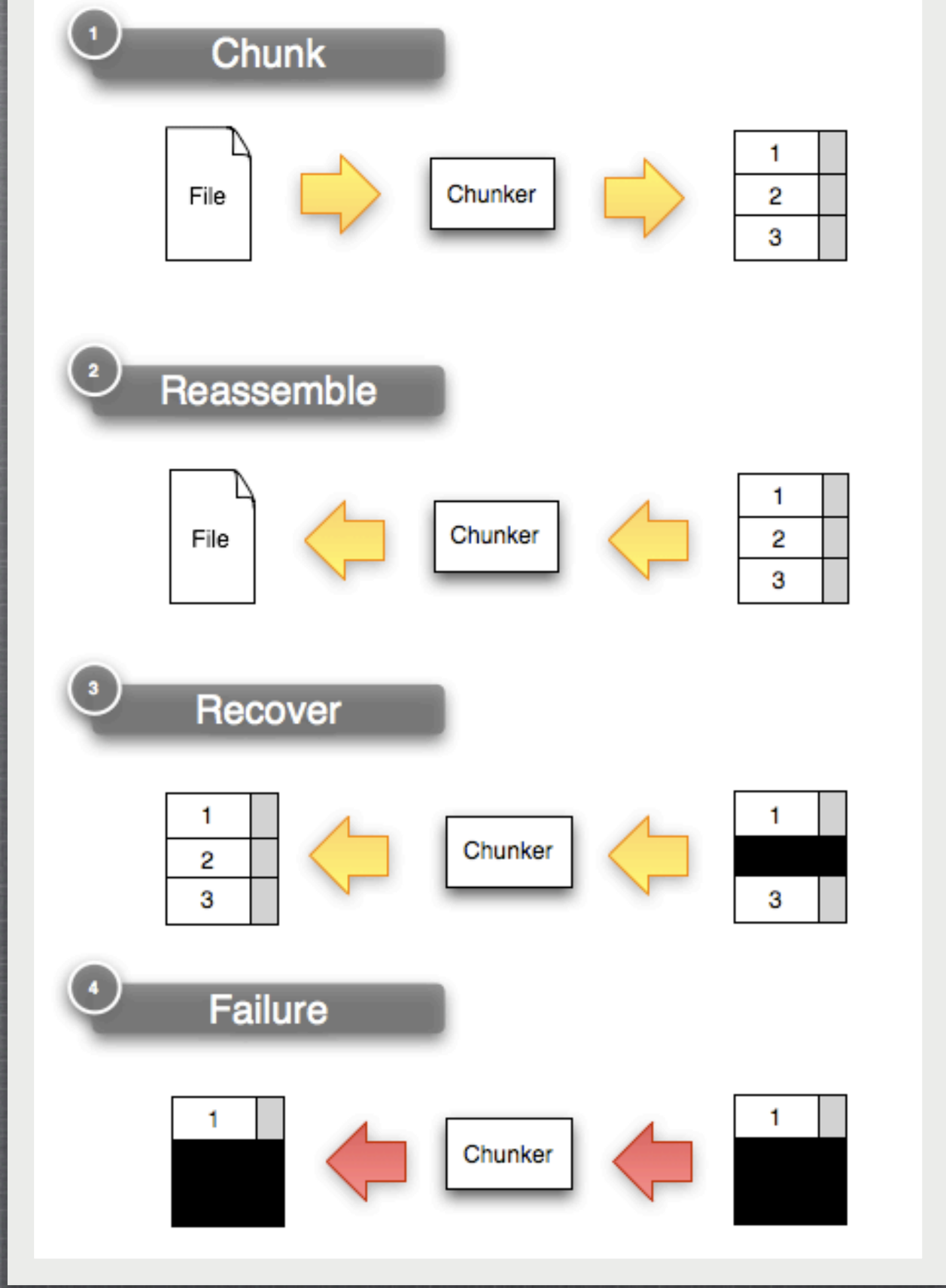
JUNIT TESTS

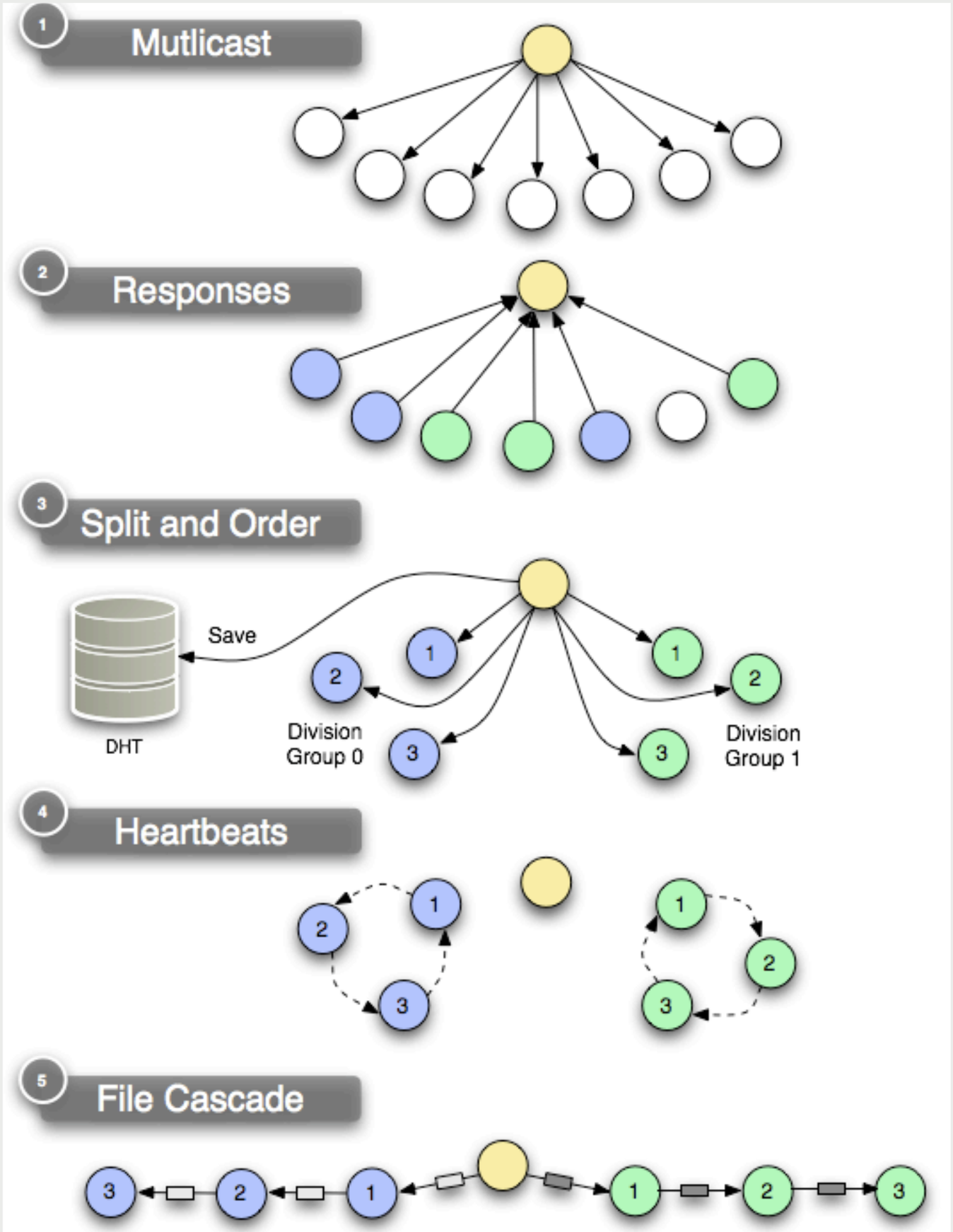


- Although testing the actual peer-to-peer application could not be done programmatically, we implemented JUnit Tests on our libraries and data structure transformations.

CHUNKER

File splitting, reassembly, and recovery in a raid like way by storing parity data.



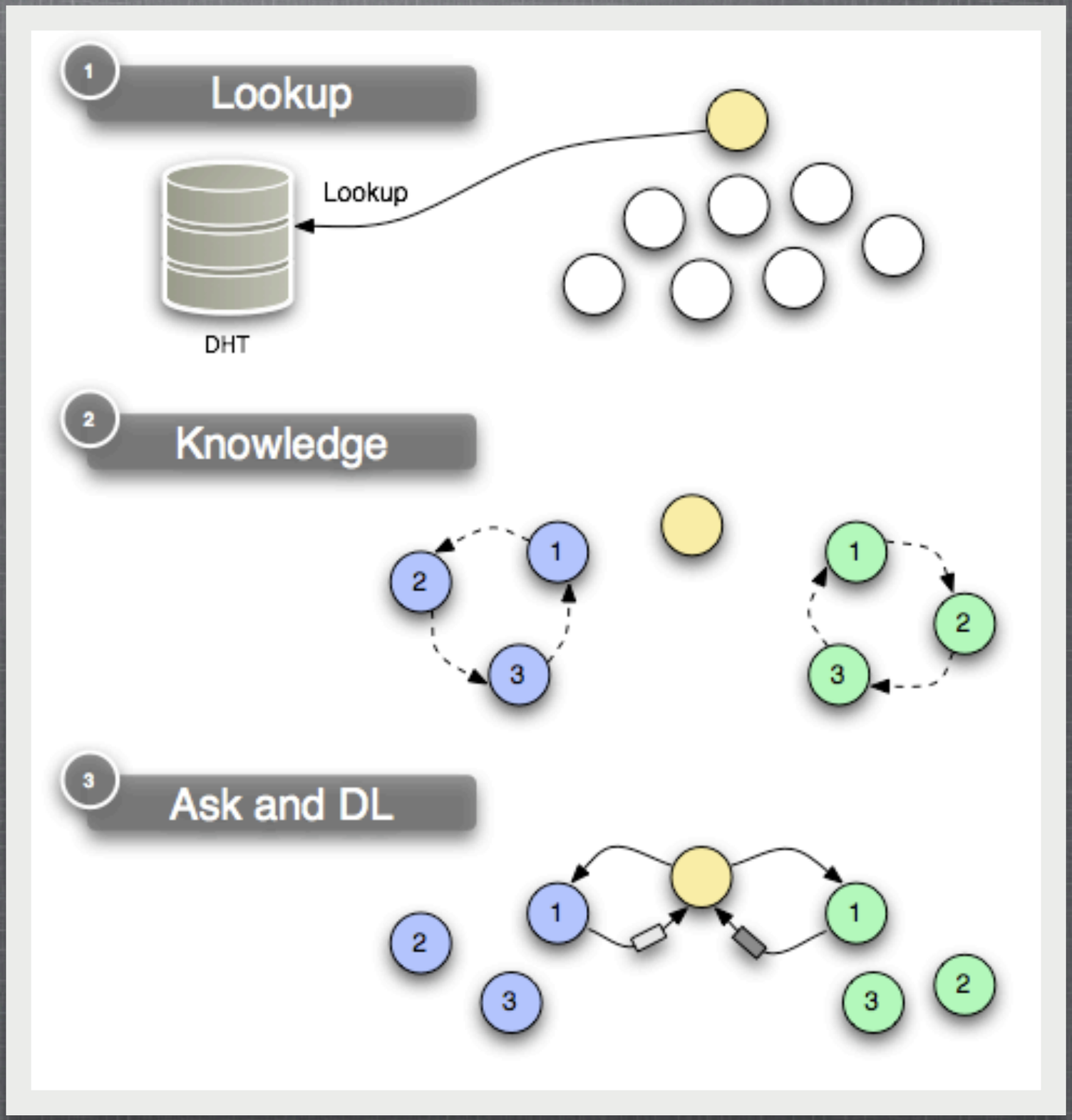


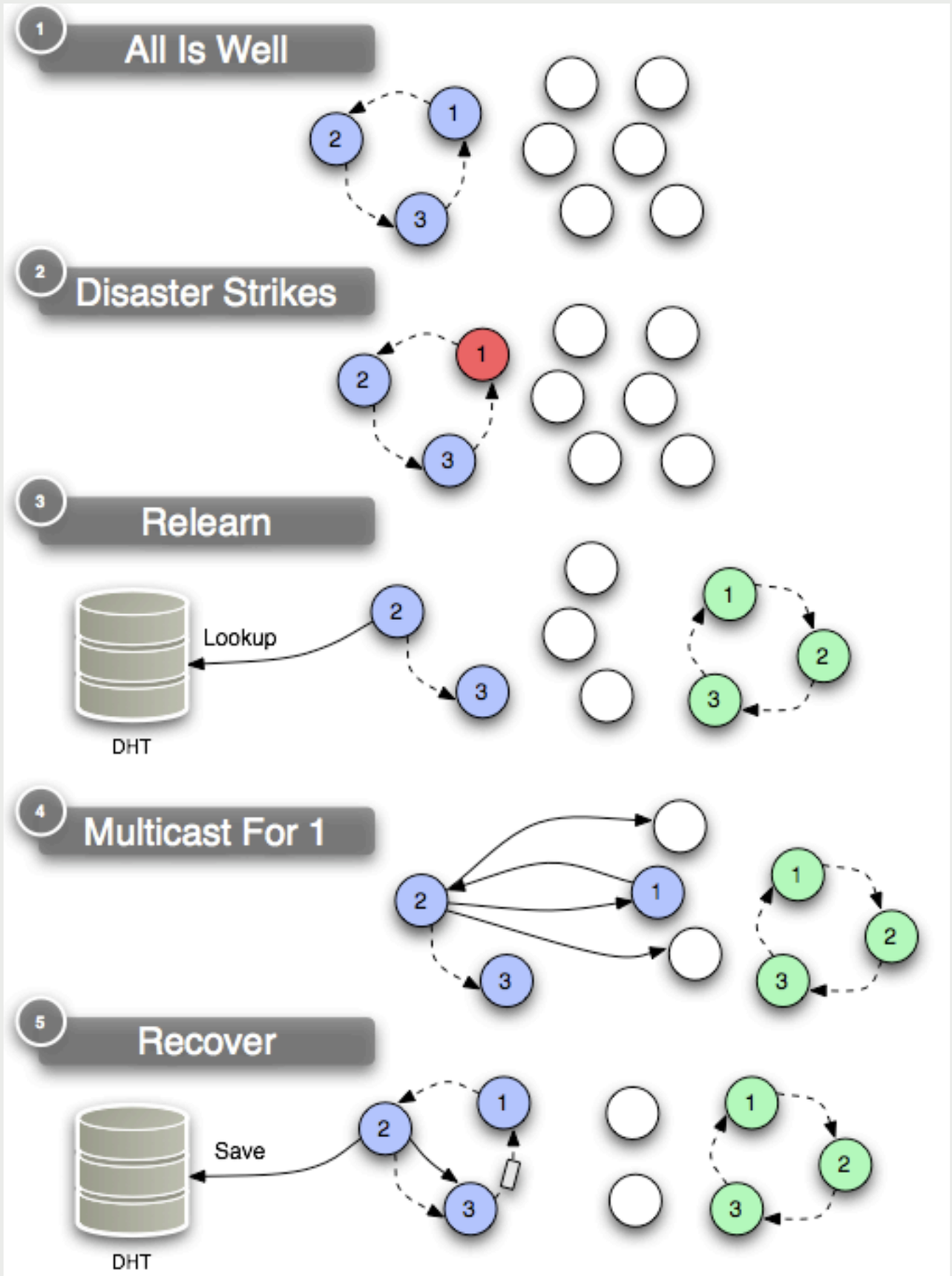
UPLOAD

The upload scenario's complex protocol of node election, ring formation, file distribution, and record keeping.

DOWNLOAD

The relative ease of downloading a previously uploaded file due to the complexity of the uploading protocol and maintenance.





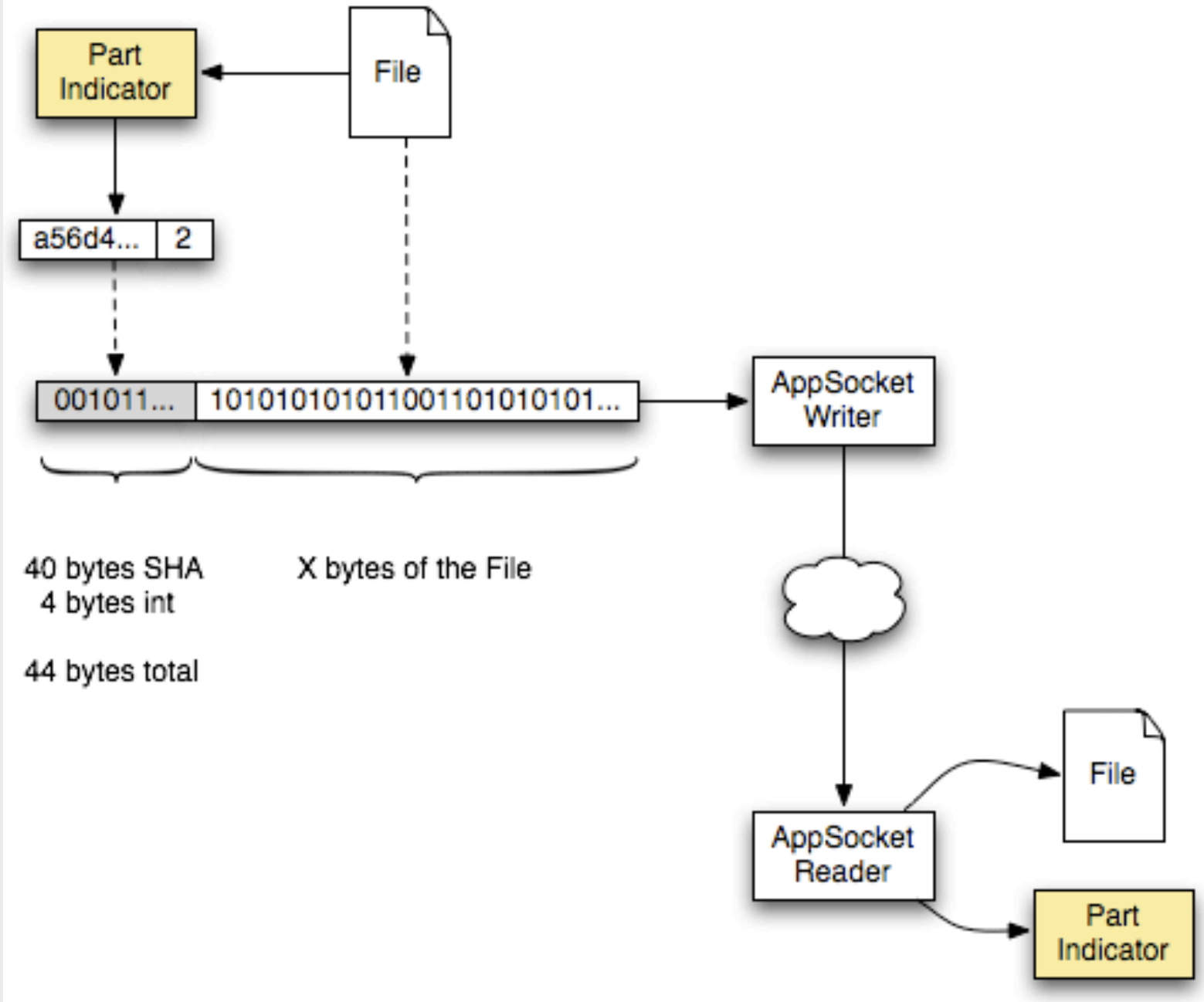
FAULT TOLERANCE

Our system is robust enough to recover from common node failure. The ring structure and heartbeats play a very important role.

FILE TRANSFERS

We make heavy use of PartIndicator objects instead of raw hashes in order to reserve the hashes for verification and trust building among nodes.

Efficient transfers in Pastry require some work on our part.



DEMONSTRATIONS

- Scenario: User enters interface and creates network
- Scenario: User uploads file
- Scenario: Storage node dies
- Scenario: User downloads file

DEMONSTRATION

LESSONS LEARNED

- Familiarity with a standard peer-to-peer substrate.
  - FreePastry claims to implement a standardized common API for general peer-to-peer overlays.
  - More experience with message passing techniques and the synchronization issues that arise.
- Sufficient thought and design pays off.
  - Every design concept was the result of hours of thought and experimentation. It paid off.

LESSONS CONT.

- Collaborative utilities make life easier.
  - Dropbox for file-sharing.
  - Google Documents for document construction.
  - GoogleCode's Subversion repository for shared development.
- Collaborative software is not a substitute for regular weekly meetings.
- Tutorials for open-source software can and will be outdated. Other sources, such as mailing lists help.
- Graphics are helpful for debugging.
- Graphics are hurtful for debugging.

FUTURE-WORK

- Provide public-key encryption for file storage and authentication for file requests.
- Permanent file storage rather then just in memory.
- Harden Fault Tolerance by handling known edge cases and race conditions.
- Modularize the RAID algorithm so any RAID-like algorithm could be put in its place. Swappable backup algorithms would be possible.

QUESTIONS?

LINKS

- Team Website:
  http://www.cs.rit.edu/~jjp1820/distributed/

- FreePastry:
  http://www.freepastry.org/