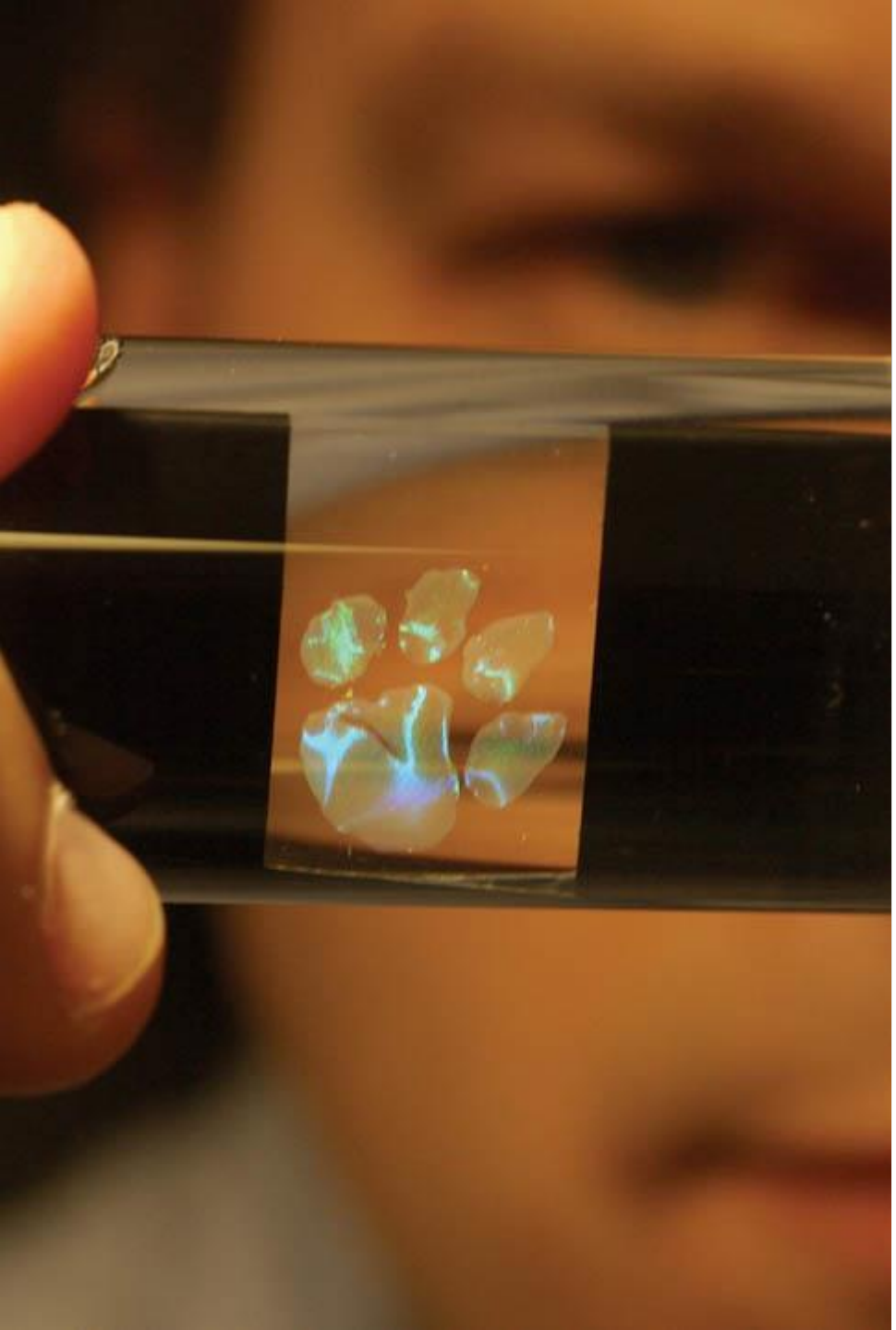


CLEMSON UNIVERSITY

MobileCopy: Resisting Correlated Node Failures to Enhance Data Availability in DTNs

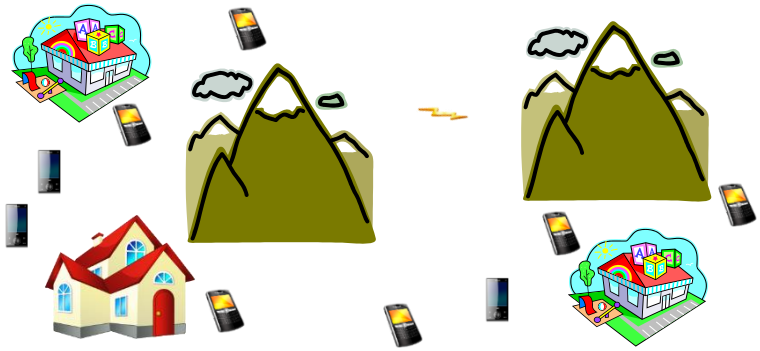
Authors: Li Yan, Kang Chen, Haiying Shen and Guoxin Liu
Dept. of Electrical and Computer Engineering
Clemson University, SC, USA

Outline

- Introduction
- Related work
- Problem depiction
- Design of MobileCopy
- Evaluation
- Conclusion

Introduction

- Nodes form delay tolerant networks in distributed manner
  - Without infrastructure for communication
- Nodes move autonomously in the network
  - Example 1: probing sensors on battlefield
  - Example 2: rescue devices in disaster area
  - Example 3: mobile devices held by people on campus



Introduction (cont.)

- File sharing in DTN is **important**
  - Nodes need to share their captured events
  - Locate malicious nodes timely
  - Nodes need to efficiently find the interested file

- Cascaded node failure in DTN is **common**
  - Failure of one node may result in overload to nearby nodes
  - Malicious nodes can infect nearby nodes
  - Power outage leads to correlated node failure

Related Work

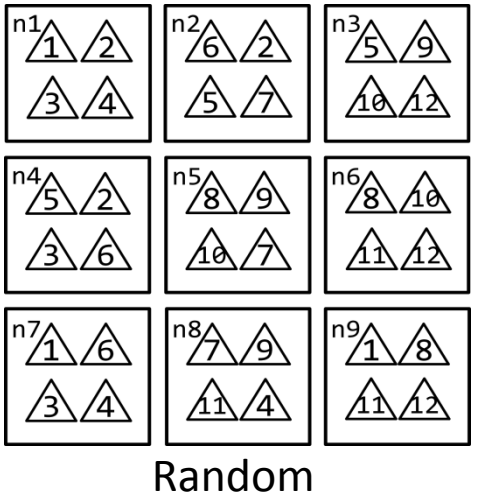
- Increase file availability by replication [TMC'06, MSN'13]
  - Create or delete file replicas to prevent data loss with reasonable cost
  - Differentiate nodes by carrying ability
- Popularity based methods [ICNP'11, Mobilware'09]
  - Determine file replicas based on access frequency of files
  - Consider storage and mobility of nodes
- *There is no work that tries to reduce data loss in correlated node failures, which is common in wireless networks*

Problem Depiction

- Previous methods: random replication
  - A failure node set (FNS) is a set of nodes whose simultaneous failures cause a file loss
  - The probability that a file is replicated on any FNS is uniformly distributed
- Drawback of random replication
  - When the number of files is large enough, the failure of any FNS will result in the loss of some files
  - Distributed nature of DTNs makes file status update difficult
  - Invalid requests for lost files degrade the efficiency of file sharing in DTNs

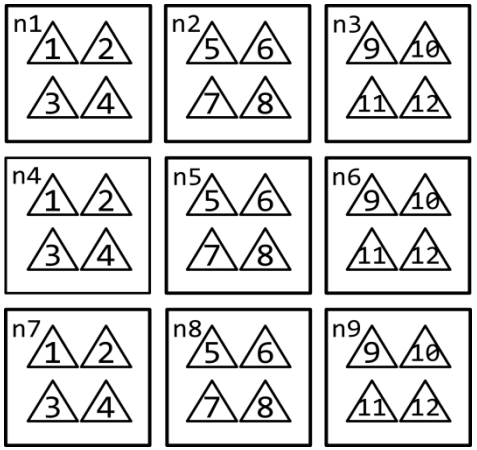
Problem Depiction

- Random replication
  - $File_1:(n_1,n_7,n_9)$, $File_2:(n_1,n_2,n_4)$, $File_3:(n_1,n_4,n_7)$,
  - $File_4:(n_1,n_7,n_8)$, $File_5:(n_2,n_3,n_4)$, $File_6:(n_2,n_4,n_7)$,
  - $File_7:(n_2,n_5,n_8)$, $File_8:(n_5,n_6,n_9)$, $File_9:(n_3,n_5,n_8)$,
  - $File_{10}:(n_3,n_5,n_6)$, $File_{11}:(n_6,n_8,n_9)$, $File_{12}:(n_3,n_6,n_9)$
  - Probability of data loss: 12/84=14.3%



Random

- Constrained replication
  - $<File_1,File_2,File_3,File_4>:(n_1,n_4,n_7)$,
  - $<File_5,File_6,File_7,File_8>:(n_2,n_5,n_8)$,
  - $<File_9,File_{10},File_{11},File_{12}>:(n_3,n_6,n_9)$
  - Probability of data loss: 3/84=3.6%



Constrained

Problem Depiction

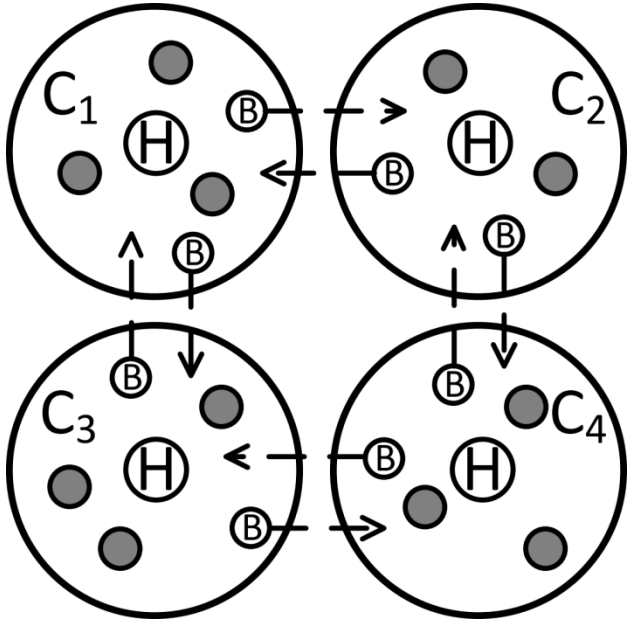
- Constrained replication in DTN is non-trivial
  - No central server for determining candidate replica holder combinations
  - Delivering replicas to determined holders takes time in DTNs
  - Accessing replica placement information is not easy
  - Jointly considering file popularity and data loss reduction to enhance data availability is challenging

Design: Network Model

- A DTN with n nodes
  - N_i, $i = 1,2,3,\cdots,n$
- File sharing is based on community
  - Nodes may have common mobility patterns
  - Nodes with high meeting probability form one community
- Focus of MobileCopy
  - File replication
  - File replica indexing

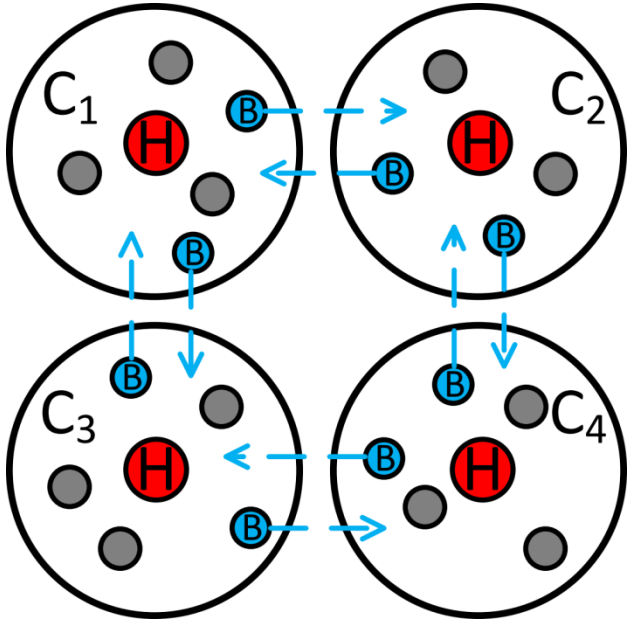
Design: File Replication

- Different roles in a community
  - Head: the node with the highest centrality
    - Maintain information of the community distribution
    - Conduct file replication
    - Maintain replica placement information in its community
  - Broker: the node that frequently visits other communities
    - Transfer information between communities



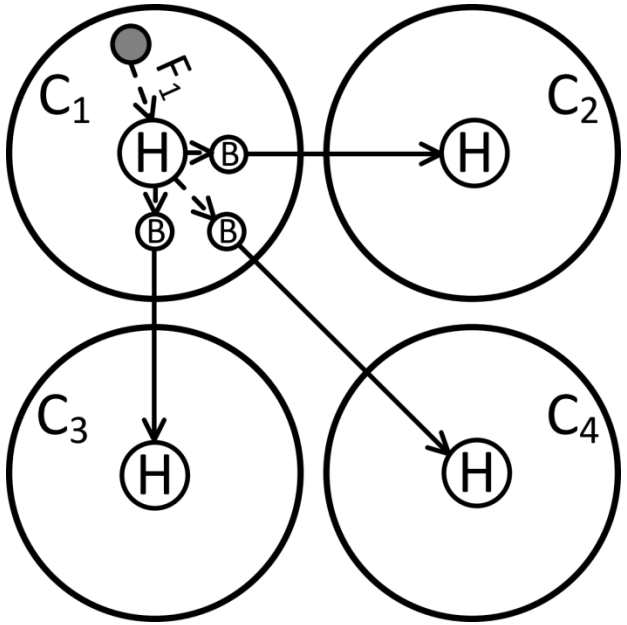
Design: File Replication

- Different roles in a community
  - Head: the node with the highest centrality
    - Maintain information of the community distribution
    - Conduct file replication
    - Maintain replica placement information in its community
  - Broker: the node that frequently visits other communities
    - Transfer information between communities



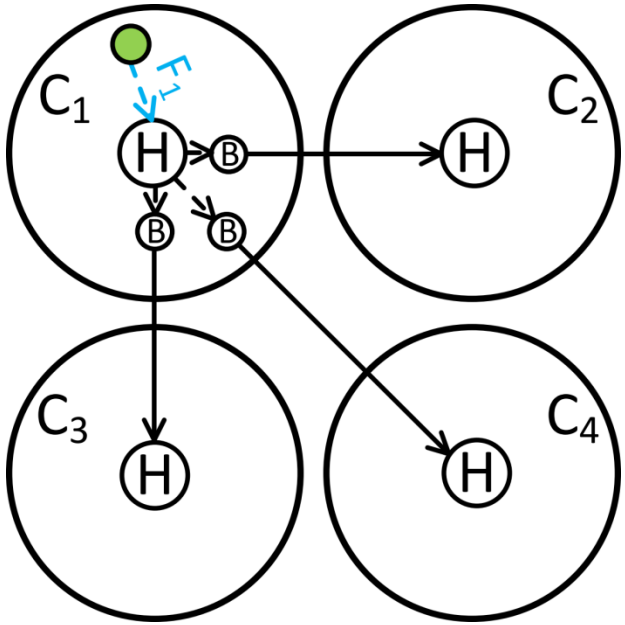
Design: File Replication

- Inter-community file replication
  - The head keeps track of the querying frequency of each file
  - The head determines which community needs replica dissemination
  - The head asks broker for each community to disseminate files



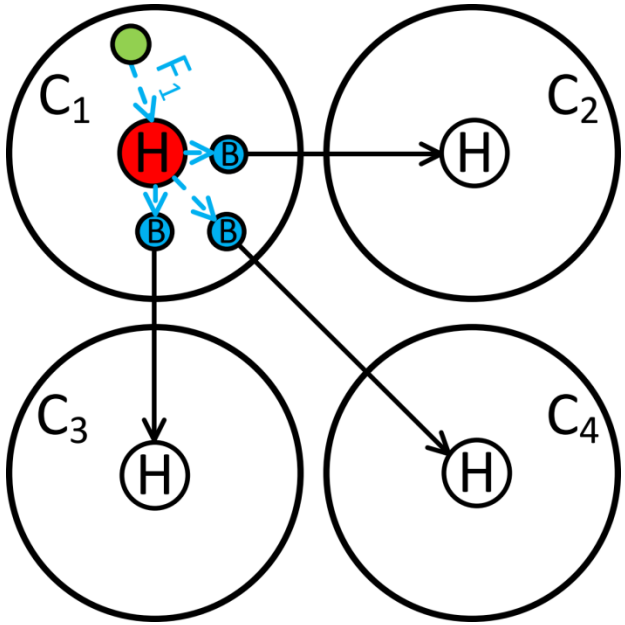
Design: File Replication

- Inter-community file replication
  - The head keeps track of the querying frequency of each file
  - The head determines which community needs replica dissemination
  - The head asks broker for each community to disseminate files



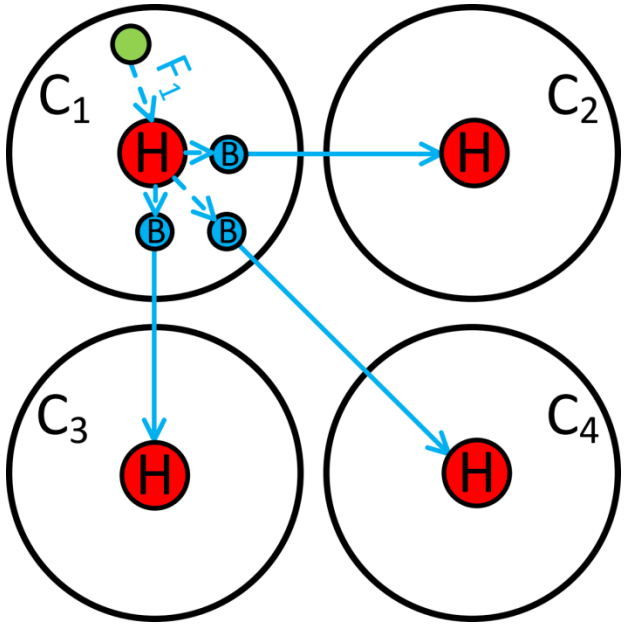
Design: File Replication

- Inter-community file replication
  - The head keeps track of the querying frequency of each file
  - The head determines which community needs replica dissemination
  - The head asks broker for each community to disseminate files



Design: File Replication

- Inter-community file replication
  - The head keeps track of the querying frequency of each file
  - The head determines which community needs replica dissemination
  - The head asks broker for each community to disseminate files



Design: File Replication

- Popularity-aware replica number determination
  - The head determines the number of replicas (R_f) for a file (F_f) based on its popularity (P_f)
  - Specifically, the head determines the maximal number of replicas allowed for a file (M). The head selects M thresholds to evenly split popularity into $M+1$ ranges
  - The number of replicas in each range is r ($r=0,1,\ldots,M$), then M should satisfy

$$\sum_{r=0}^{M} r * \frac{N_c}{M+1} * s_a \approx \alpha S$$

  - Where N_c is the total number of nodes in the community. s_a is the average file size in the community. α determines the percentage of the storage for replicas. S is the size of available storage in the community.

Design: File Replication

- Limiting replica holder combinations
  - For each file, we limit the number of candidate replica holders to $M+t$ (t is a small integer)
  - Specifically, the head splits N_c community nodes into $\left\lceil \frac{N_c}{M+t} \right\rceil$ groups. The replica holders of each file can only be selected from one group.
  - Suppose M=3, t=1, N_c=12, the nodes are split as below

$$< N_1, N_2, N_3, N_4 \| N_5, N_6, N_7, N_8 \| N_9, N_{10}, N_{11}, N_{12} >$$

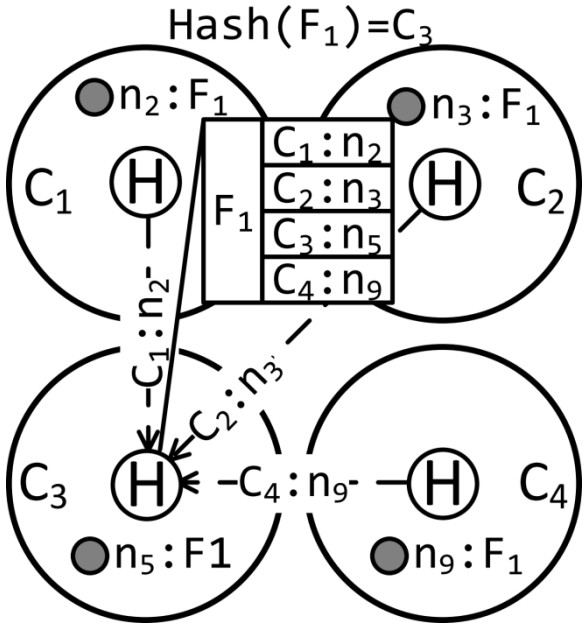
  - Random replication has $\binom{12}{3} = 220$ possible combinations. While MobileCopy only has $3 * \binom{4}{3} = 12$ possible combinations.

Design: File Replication

- Storage limitation consideration
  - MobileCopy lets each node report to the head when its storage becomes full
  - The head excludes the nodes without available storage when selecting holders for replicas
  - When the storage of all nodes in a community is full, the target node of the new replica randomly drops some replicas
  - Dropping replicas randomly does not break the rule that more popular files have more replicas

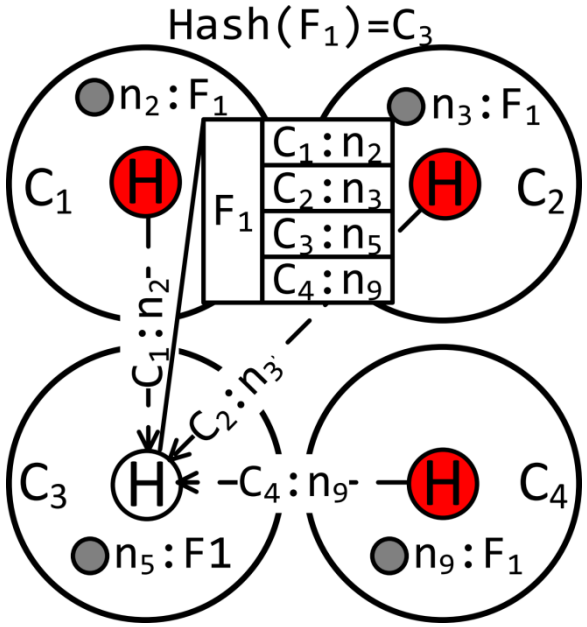
Design: Replica Indexing

- Distributing replica information
  - MobileCopy uses the distributed hash table (DHT) to distribute replica placement information
  - By hashing the file name, the head identifies the community to report the determined replica placement information
  - The replica placement information will be stored in multiple nodes that often stay in the community



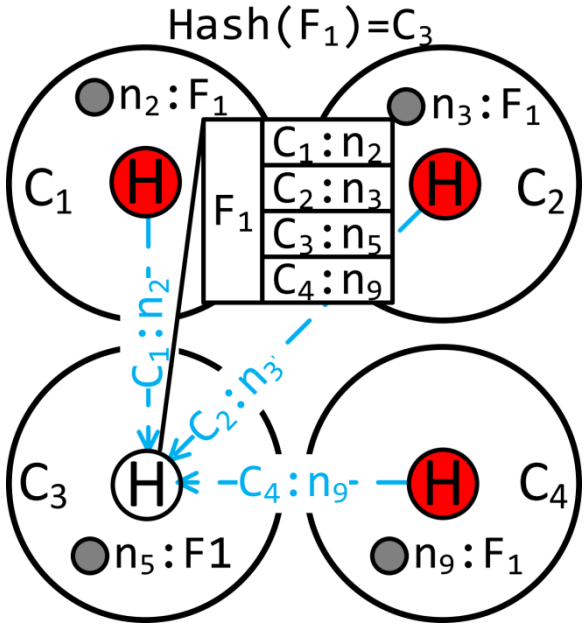
Design: Replica Indexing

- Distributing replica information
  - MobileCopy uses the distributed hash table (DHT) to distribute replica placement information
  - By hashing the file name, the head identifies the community to report the determined replica placement information
  - The replica placement information will be stored in multiple nodes that often stay in the community



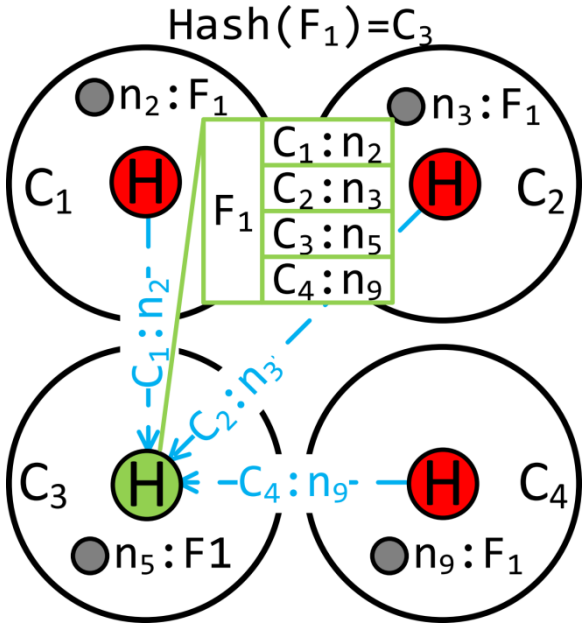
Design: Replica Indexing

- Distributing replica information
  - MobileCopy uses the distributed hash table (DHT) to distribute replica placement information
  - By hashing the file name, the head identifies the community to report the determined replica placement information
  - The replica placement information will be stored in multiple nodes that often stay in the community



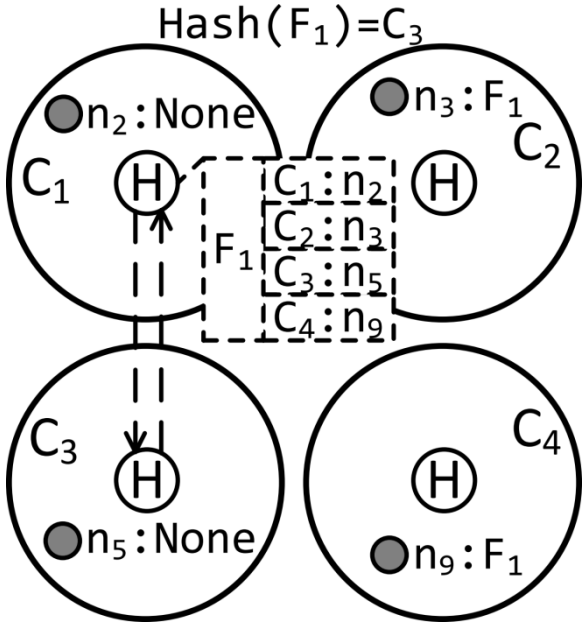
Design: Replica Indexing

- Distributing replica information
  - MobileCopy uses the distributed hash table (DHT) to distribute replica placement information
  - By hashing the file name, the head identifies the community to report the determined replica placement information
  - The replica placement information will be stored in multiple nodes that often stay in the community



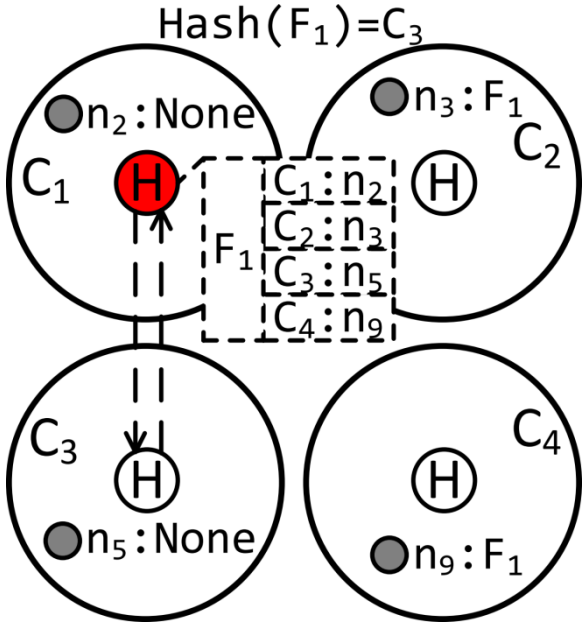
Design: Replica Indexing

- Finding placement information
  - Upon the request for a file, the head first checks whether the replica holder of the file exists in local community
  - Otherwise, the head hashes the file ID to know the community with the replica placement information
  - Through brokers, the placement information is returned back



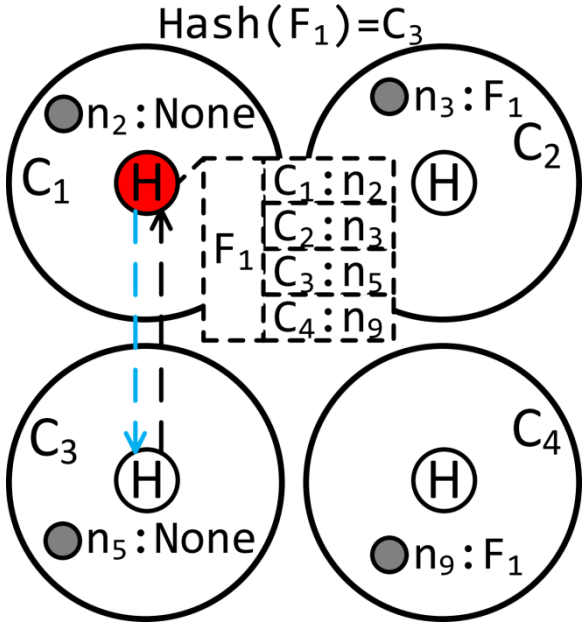
Design: Replica Indexing

- Finding placement information
  - Upon the request for a file, the head first checks whether the replica holder of the file exists in local community
  - Otherwise, the head hashes the file ID to know the community with the replica placement information
  - Through brokers, the placement information is returned back



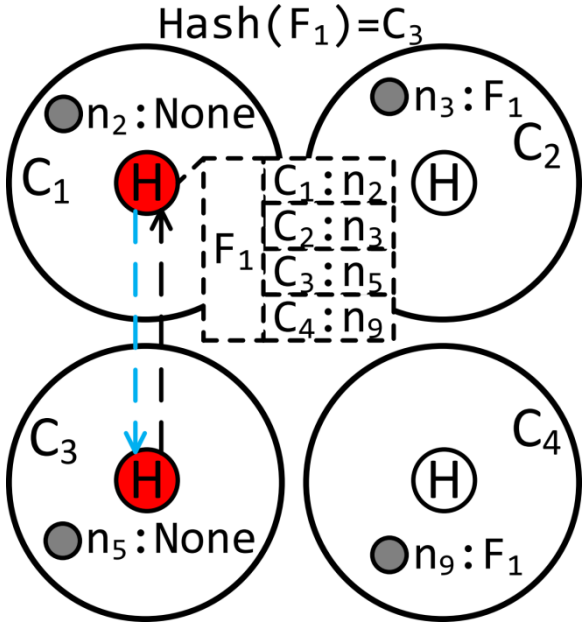
Design: Replica Indexing

- Finding placement information
  - Upon the request for a file, the head first checks whether the replica holder of the file exists in local community
  - Otherwise, the head hashes the file ID to know the community with the replica placement information
  - Through brokers, the placement information is returned back



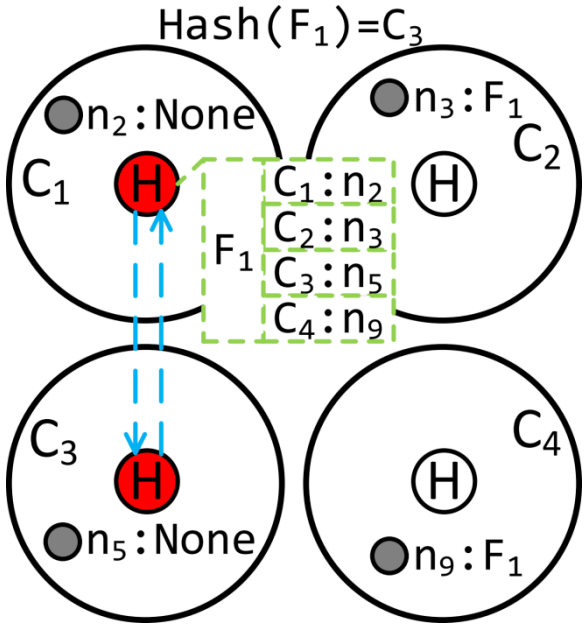
Design: Replica Indexing

- Finding placement information
  - Upon the request for a file, the head first checks whether the replica holder of the file exists in local community
  - Otherwise, the head hashes the file ID to know the community with the replica placement information
  - Through brokers, the placement information is returned back



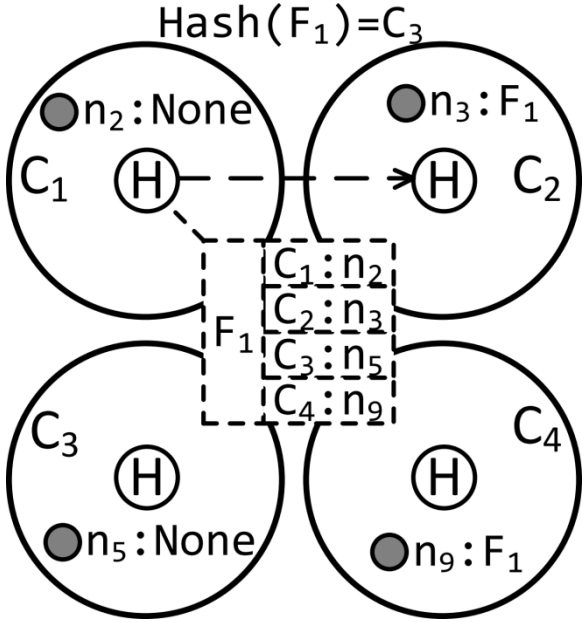
Design: Replica Indexing

- Finding placement information
  - Upon the request for a file, the head first checks whether the replica holder of the file exists in local community
  - Otherwise, the head hashes the file ID to know the community with the replica placement information
  - Through brokers, the placement information is returned back



Design: Replica Indexing

- Locating the requested file
  - From the replica placement information, the head schedules searches in the communities with replicas
  - The head uses DTN routing algorithm to send the request to the target node
  - The procedures repeat until a replica is found



Performance Evaluation

- Simulator
  - Event driven simulator
- Node mobility traces
  - Dartmouth trace (DART) [1]:
    - A 119-day record for wireless devices carried by students on Dartmouth College campus
    - 22 communities, 15 nodes per community in average
    - Initial period: 30 days
  - DieselNet trace (DNET) [2]:
    - A 20-day record for WiFi nodes attached to the buses in the downtown area of UMass college town
    - 3 communities, 12 nodes per community in average
    - Initial period: 2.5 days

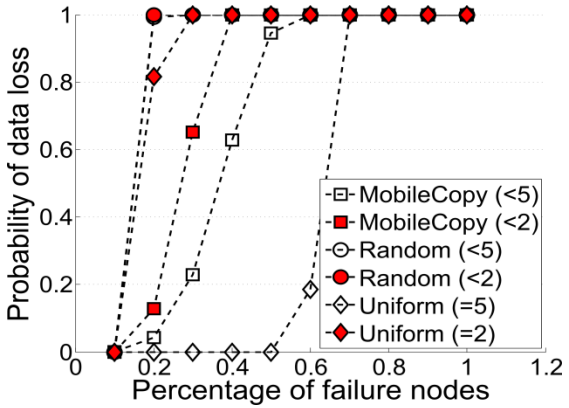
[1] T. Henderson, etc. "The changing usage of a mature campus-wide wireless network," in Proc. of MobiCom, 2004.
[2] X. Zhang, etc. "Study of a bus-based disruption-tolerant network: mobility modeling and impact on routing," in Proc. of MobiCom, 2007.

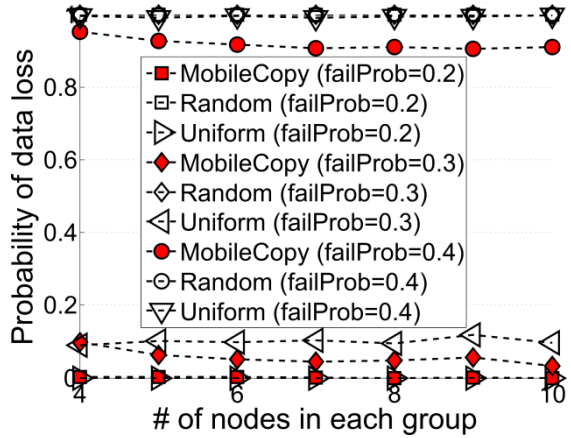
Performance Evaluation (cont.)

- Experiment settings
  - File properties
    - Each node originally holds 100 files and has 300KB available storage
    - Each file has the same size: 1KB
    - File popularity follows Zipf distribution with parameter 0.7
  - Search requirements
    - File requests were generated periodically
    - Generation period: 1 day for DART and 4 hours for DNET
    - Search rate: number of file requests generated in each period
    - Expiration TTL: 4 hours in DART and 2 hours in DNET

Performance Evaluation (cont.)

- Data loss resistance
  - Comparison methods
    - Random: randomly places replica considering popularity
    - Uniform: randomly places replica without considering popularity
  - Metrics
    - Data loss probability under various percentages of concurrent failure
    - Data loss probability under various sizes of nodes in a group



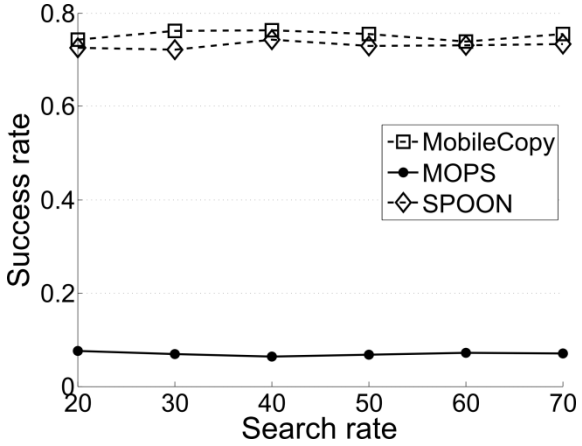


Performance Evaluation (cont.)

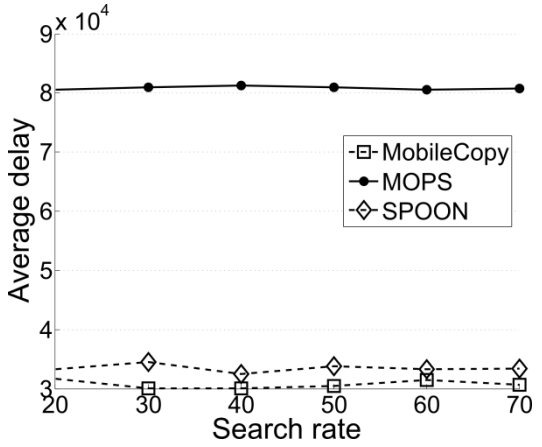
- File search efficiency
  - Comparison methods
    - MOPS publish/subscribe system [1]
    - SPOON file sharing system [2]
  - Metrics
    - **Success rate**: percentage of file requests that successfully reach their target files within TTL
    - **Average delay**: average time (in seconds) spent by file requests to reach their target files
    - **Average search length**: average number of forwarding hops experienced by a file request

[1] F. Li and J. Wu, "MOPS: Providing content-based service in disruption-tolerant networks," in Proc. of ICDCS, 2009.
[2] K. Chen, H. Shen, and H. Zhang, "Leveraging social networks for p2p content-based file sharing in mobile ad hoc networks," in Proc. of MASS, 2011.

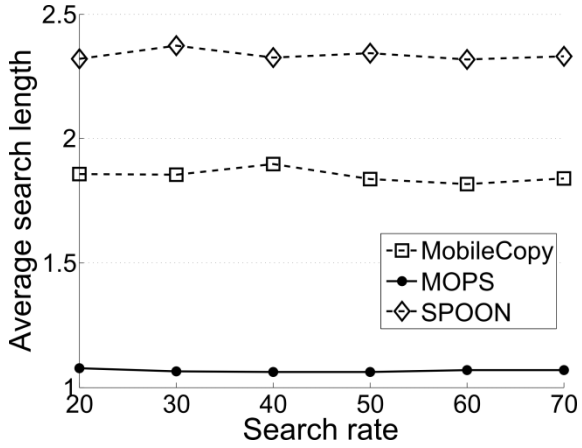
Experiment with Different Search Rates (DART)



Success rate: MobileCopy>SPOON>>MOPS

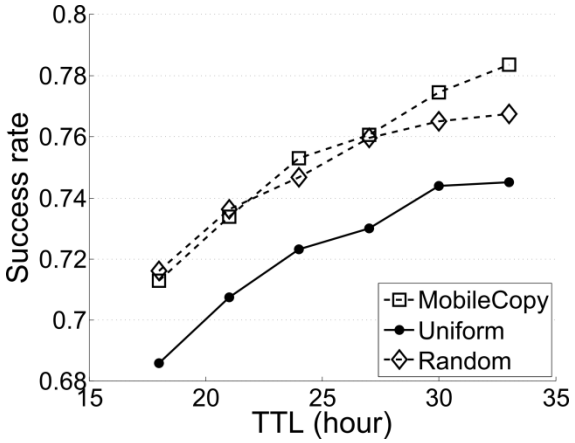


Ave. delay: MOPS>>SPOON>MobileCopy

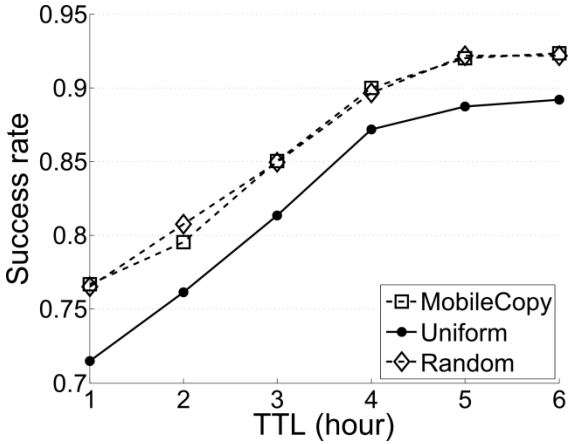


Ave. search length: MOPS<MobileCopy<SPOON

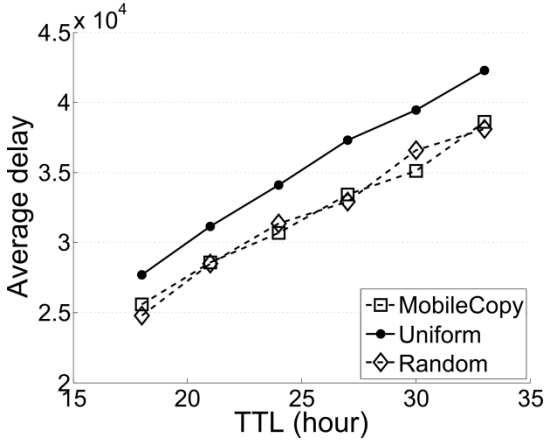
Data Availability



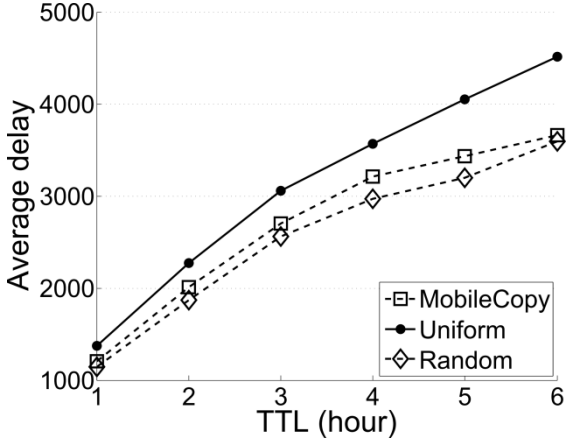
Success rate with DART trace



Success rate with DNET trace



Ave. delay with DART trace



Ave. delay with DNET trace

Conclusions

- We proposed MobileCopy, it
  - reduces the probability of data loss under correlated node failures
  - determines replication of a file based on its popularity
  - uses a DHT-based file replica indexing method for efficiently updating and fetching replicas
- In our future work, we plan to further study how to constrain the number of FNSs to reduce data loss.

Thank you!

Questions & Comments?

Li Yan, PhD Candidate

lyan@clemson.edu

Pervasive Communication Laboratory

Clemson University