

GoSyP: A Gossip-based Synchronization Protocol for State Consistency in Distributed Applications

Aaron M. Rosenfeld, Dara Kusic, William C. Regli, Joseph B.
Kopena

Department of Computer Science
College of Engineering
Drexel University
Philadelphia, PA, USA

May 16, 2011

- 1 Introduction
 - Terminology
 - Motivation
 - Proposed Approach
 - Related Work
- 2 The GoSyP Protocol
 - Contributions of GoSyP
 - Major Results
 - Architecture
 - Synchronization Process
- 3 Results
 - Experimental Procedure
 - Consistency Profiles
 - Varying Density
 - Varying Mobility
- 4 Conclusion
 - Summary
 - Future Work

Terminology

State Object: Singular piece of application data which is to be kept consistent across GoSyP instances. Comprised of an ID, version, and payload.

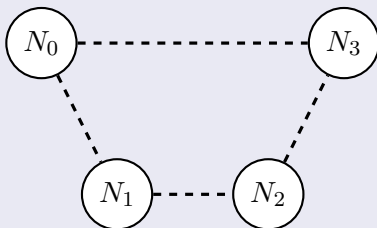
Datastore: Collection of State Objects which are maintained by GoSyP.

Metadata: Information about state objects used to locate discrepancies (e.g. ID & version).

Consistency: Percentage of nodes with identical datastores at a specific time.

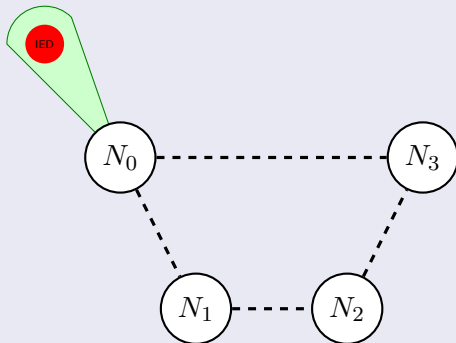
Motivation

Connected Network



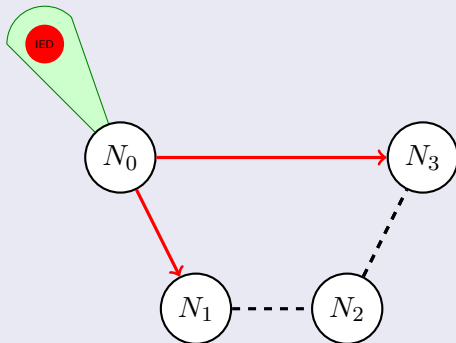
Motivation

Connected Network



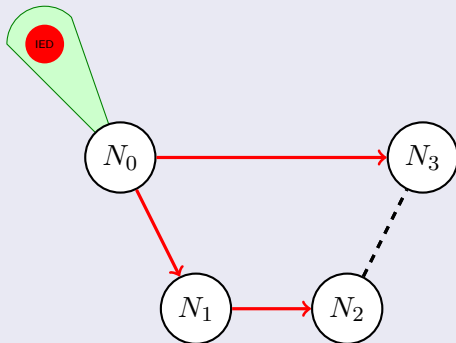
Motivation

Connected Network



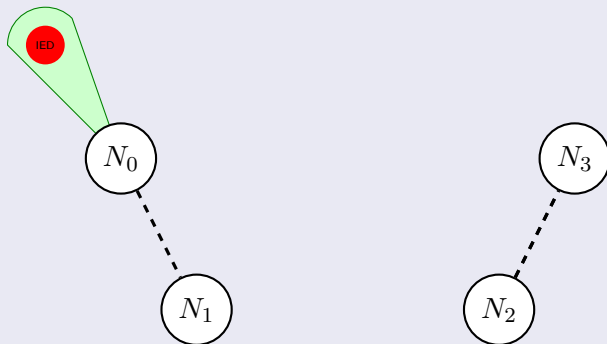
Motivation

Connected Network



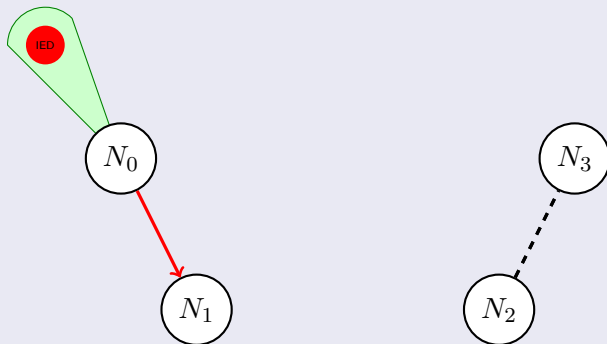
Motivation

Fragmented Network



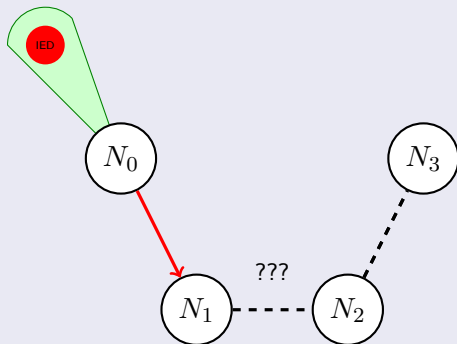
Motivation

Fragmented Network



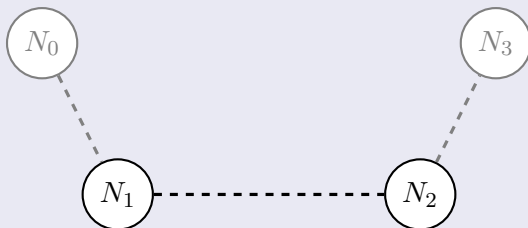
Motivation

Fragmented Network



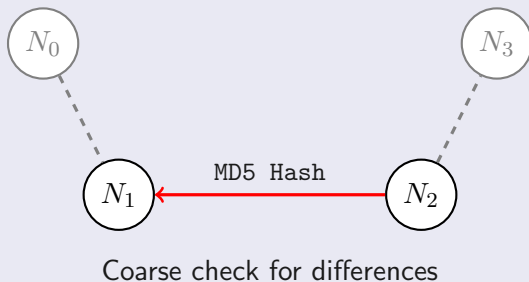
GoSyP Approach

GoSyP Reconciliation



GoSyP Approach

GoSyP Reconciliation



GoSyP Approach

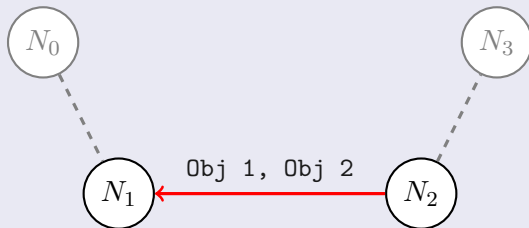
GoSyP Reconciliation



Pull information regarding remote datastore

GoSyP Approach

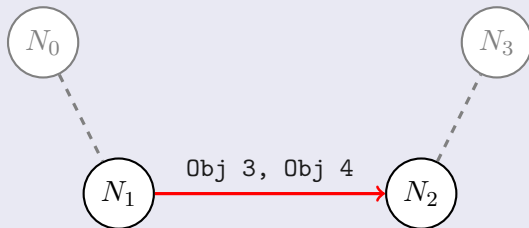
GoSyP Reconciliation



Pull data objects missing locally

GoSyP Approach

GoSyP Reconciliation



Push data objects needed remotely

Selected Related Work

Work	Contributions
Gossip (Demers)	Seminal work utilizing randomized messaging for weak datastore replication.
Polite Gossip (Levis)	Improves on gossip by intelligent metadata exchanges for locating datastore inconsistencies. May still involve randomness for transmitting payloads.
Trickle (Levis)	Uses periodic metadata exchanges, but uses suppressible payload messages to minimize redundant transmissions. Overhead scales $O(d)$ with d data items.
DIP (Lin)	Uses a binary search to locate inconsistencies between datastores. Overhead scales $O(\log d)$ with d data items.

Contributions

- Provides **generic middleware** architecture for maintaining weak consistency in disrupted environments.
- Proposes a **unicast, iterative metadata exchange process** for identifying inconsistencies between applications' state.
- Demonstrates the ability to maintain **minimal addressing information**, while allowing unicast to perform effectively.
- **Exploits wireless nature** of mobile networks.
- GoSyP has a worst-case of $O(d)$ **and best-case of $O(1)$** , with the in number of discrepancies & ordering method determining the exact network usage.

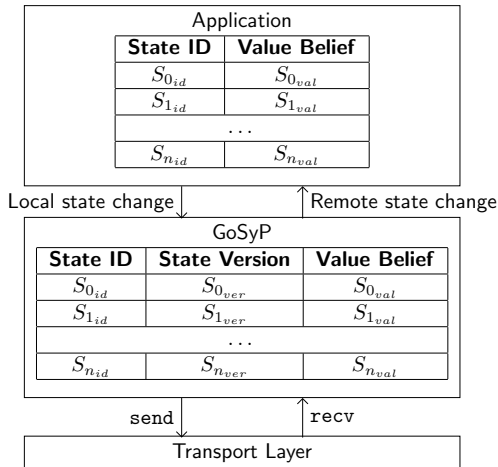
Major Results

- In all experiments, GoSyP transmitted less data than Simple Multicast Forwarding (SMF) and Polite Gossip.
- In nearly all experiments, GoSyP achieves state consistency between 90% of nodes in durations as good as, or better, than the alternative methods.
- GoSyP has a strong dependency on an underlying unicast routing protocol. GoSyP is best suited for networks where routes to nearby nodes are generally available and overhead is a concern.

Significance

- With the exception of Random Mobility, GoSyP achieved consistency 17% more slowly than the best method, but sent 25% of the data.
- Unicast can successfully route information in group mobility situations when address selection is intelligent.
- Sending state updates via unicast can be effective when nodes within one hop of the routing path can benefit from exchanges.

Architecture

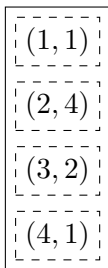


Synchronization Process

- 1 Choose a neighbor with which to synchronize.
- 2 Compare hashes of local state datastore.
- 3 If unequal, the datastores are iteratively compared.
- 4 Any differences are reconciled.

Synchronization Process Example

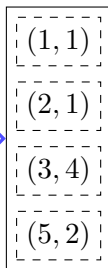
GoSyP Instance A



Hash of all state object ID/versions

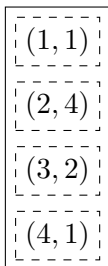


GoSyP Instance B

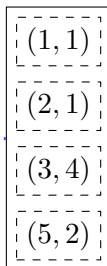


Synchronization Process Example

GoSyP Instance A



GoSyP Instance B

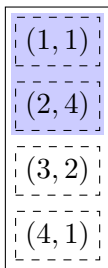


Hashes unequal



Synchronization Process Example

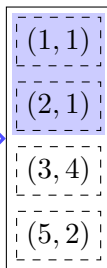
GoSyP Instance A



Send ID/version pairs for first sublist

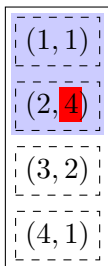
$[(1, 1), (2, 4)]$

GoSyP Instance B

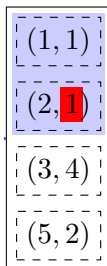


Synchronization Process Example

GoSyP Instance A



GoSyP Instance B

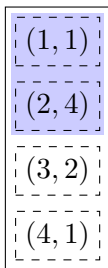


Request state 2 from instance 1.



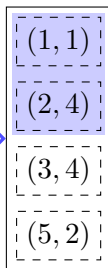
Synchronization Process Example

GoSyP Instance A



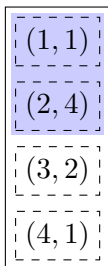
Payload of state 2.

GoSyP Instance B

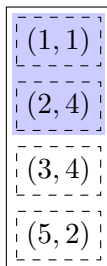


Synchronization Process Example

GoSyP Instance A



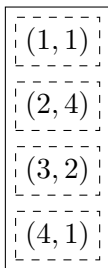
GoSyP Instance B



Sublist 1 now consistent

Synchronization Process Example

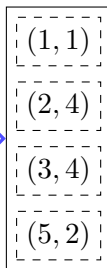
GoSyP Instance A



Hash of all state object ID/versions

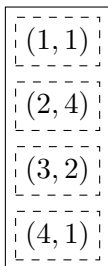


GoSyP Instance B

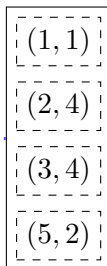


Synchronization Process Example

GoSyP Instance A



GoSyP Instance B

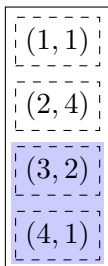


Hashes unequal



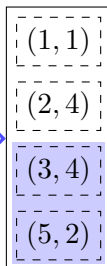
Synchronization Process Example

GoSyP Instance A



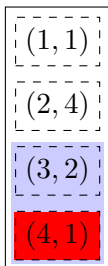
Send ID/version pairs for second sublist
→
[(3, 2), (4, 1)]

GoSyP Instance B

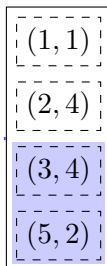


Synchronization Process Example

GoSyP Instance A



GoSyP Instance B

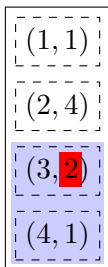


Request state 4 from instance 1.



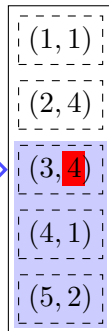
Synchronization Process Example

GoSyP Instance A



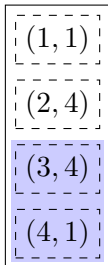
Payload of state 4.

GoSyP Instance B

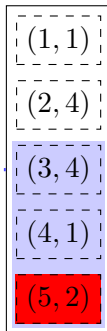


Synchronization Process Example

GoSyP Instance A



GoSyP Instance B

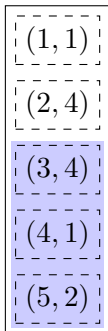


Push payload of state 3.

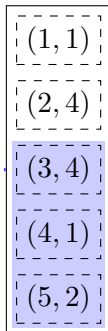


Synchronization Process Example

GoSyP Instance A



GoSyP Instance B



Push payload of state 5.

Synchronization Process Example

GoSyP Instance A

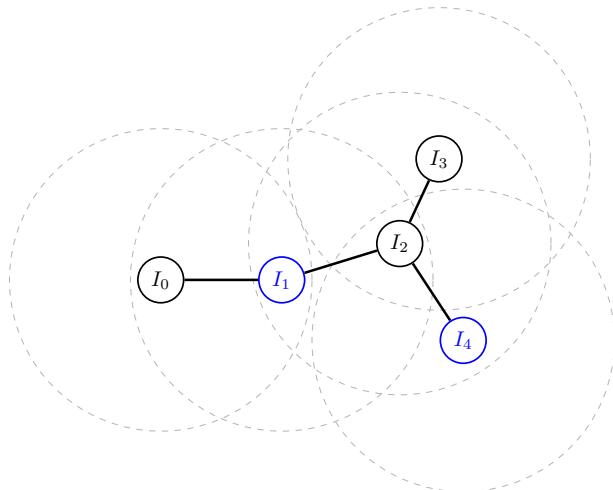
(1, 1)
(2, 4)
(3, 4)
(4, 1)
(5, 2)

GoSyP Instance B

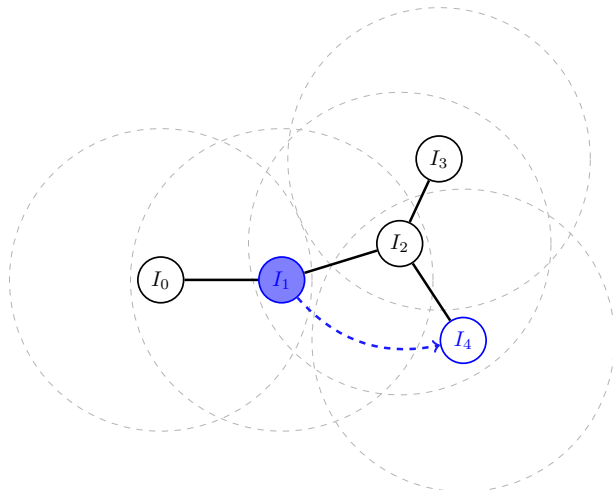
(1, 1)
(2, 4)
(3, 4)
(4, 1)
(5, 2)

Datastores are now equal

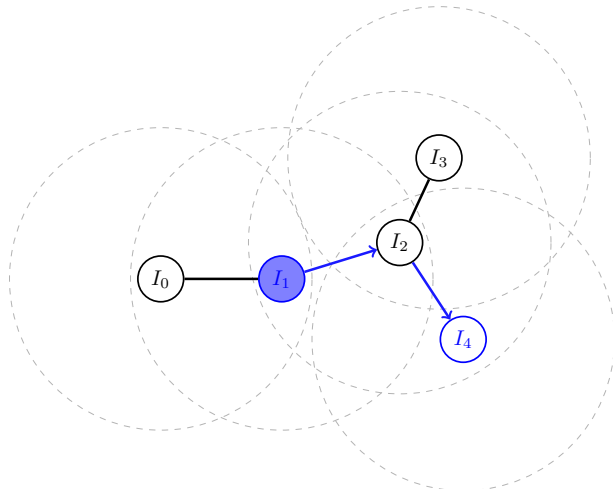
Passive Synchronization



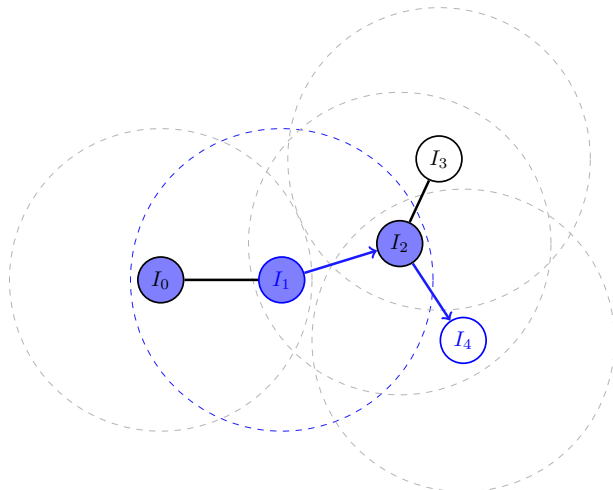
Passive Synchronization



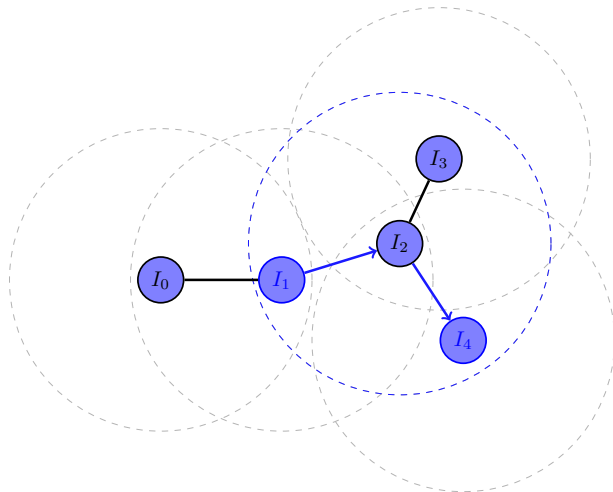
Passive Synchronization



Passive Synchronization



Passive Synchronization



Group Membership & Peer Selection

Only a subset of all nodes' addressing information, a **local view** \mathcal{V} , is maintained in each GoSyP instance.

View Maintenance

Every synchronization, \mathcal{V} is chosen as a random subset of:

- \mathcal{V}_{mix} : Views that the local GoSyP instance has requested from other instances. The number of instances that are queried is denoted f .
- \mathcal{V}_{req} : Addresses of instances which have requested the local instance's view.

Group Membership & Peer Selection

The size of \mathcal{V}_t (\mathcal{V} at time t) scales linearly with the **total** estimated network size, \hat{N} , given by:

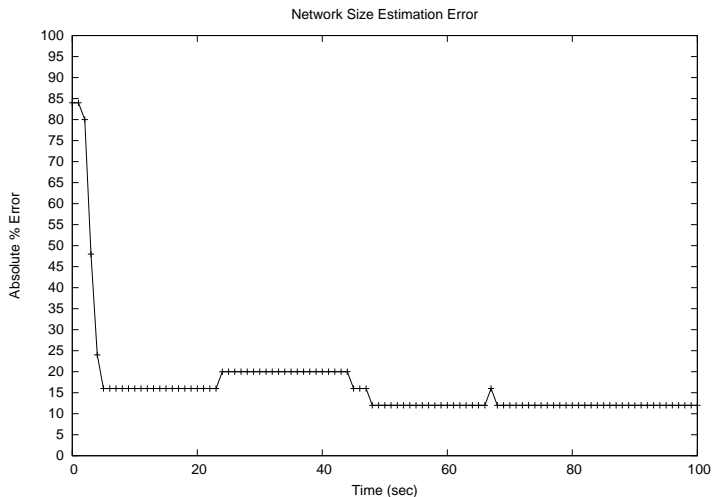
$$\hat{N} = \frac{(|\mathcal{V}_{t-1}| + 1) \cdot (|\mathcal{V}_t| + 1)}{|\mathcal{V}_{t-1} \cap \mathcal{V}_t| + 1}$$

Thus:

$$\mathcal{V} \subseteq (\mathcal{V}_{mix} \cup \mathcal{V}_{req}), |\mathcal{V}| \sim \hat{N}$$

During a synchronization at time t , a random element of \mathcal{V}_t is chosen as a peer.

Group Membership & Peer Selection



Experimental Procedure

Experiments were run in the **Common Open Research Emulator (CORE)**:

- 1 One instance of GoSyP, Polite Gossip, or SMF was run on all nodes.
- 2 A state application & the OSPF routing daemon were also run on each node.
- 3 A basic 802.11 model was used for the MAC layer.
- 4 25 nodes were scattered throughout a 2-dimensional area.
- 5 All applications were passed 5 unique state objects at startup.
- 6 The experiment was run until all 25 nodes had all 125 (25 nodes \times 5 objects) state objects.

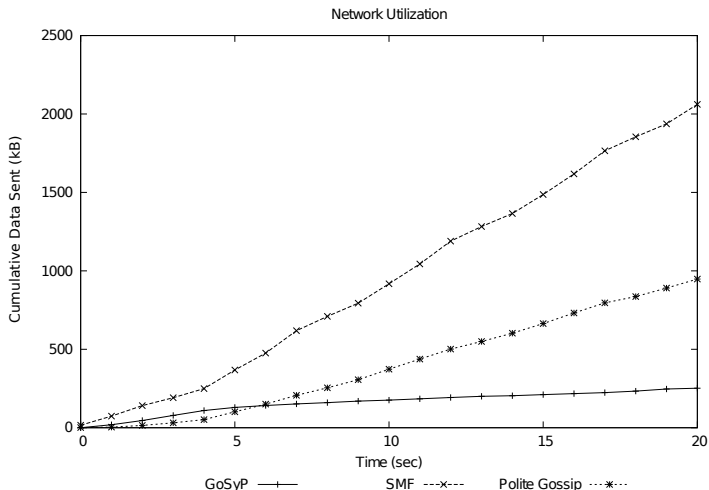
Experiment Parameters

Parameter	Value
Network size	25 nodes
State count per node	5
Max. Transmission range	250 m
Minimum latency	100 ms
Link Bandwidth	5.5 Mbps
Synch. interval	5 sec.
Number of runs per experiment	10

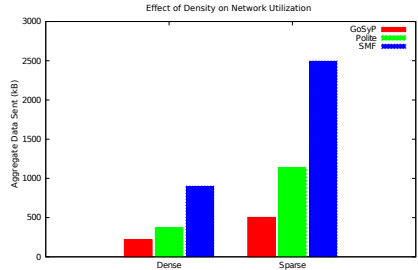
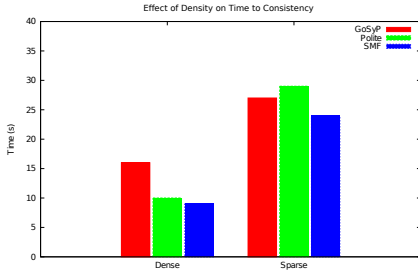
Experiment Metrics

Metric	Description
Time to Consistency	Time until all GoSyP instances which have received all state information.
Overhead	Cumulative quantity of network traffic at PHY layer (kB).

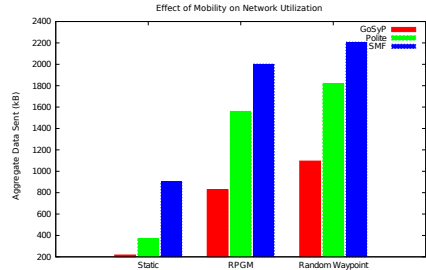
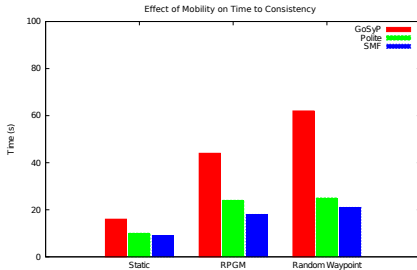
Consistency Profiles



Varying Density



Varying Mobility



Summary

- GoSyP provides a **generic middleware** for unicast datastore consistency.
- In group mobility situations, such as the tactical edge, data may generally be consistent within groups. GoSyP can **rectify issues arising from temporary fragmentation**.
- From experiments, GoSyP tends to **use significantly less overhead** than broadcast methods, even Polite Gossip which is considered “intelligent.”
- Like broadcast methods, GoSyP’s **transmissions generally reach more than one node**, allowing peers to benefit from foreign synchronization sessions.

Future Work

- Effect of other traffic on GoSyP performance.
- Alternative routing algorithms.
- State value errors during transmission.
- Other MAC layers.
- Item ordering.
- Bounding of overhead based on inconsistencies and ordering method.

Thank you!

Questions/Comments?

Aaron M. Rosenfeld
Department of Computer Science
College of Engineering
Drexel University
Philadelphia, PA, USA
ar374@drexel.edu