NOVA SCHOOL OF SCIENCE & TECHNOLOGY

Dynamic Membership Management and Data Sharding in Edge-Enabled Publish/Subscribe Systems

Jaime Saramago, João A. Silva, ***Hervé Paulino*** *and João M. Lourenço*

NOVA LINCS & NOVA School of Science and Technology
NOVA University Lisbon
Portugal

NOVALINCS

Motivation

- Publish/Subscribe (Pub/Sub) systems are widely adopted
- Decentralized deployments used to improve **resilience** and **scalability**, and reduce **latency**
  - Well suited to **IoT** and **edge** computing scenarios
- Distributed Hash Tables (DHTs) are a common substrate solution for decentralized Pub/Sub systems

Problems

- IoT and content-sharing deployments rarely satisfy the uniformity assumptions of DHTs
  - Devices (nodes) may enter and leave the network
  - Some topics attract disproportionate publication and subscription demand
    - the number and size of values per topic (DHT key) can vary significantly
    - storage and query hot-spots where a few nodes become overloaded, which limits scalability

- IoT and content-sharing deployments rarely satisfy the uniformity assumptions of DHTs
  - **Devices (nodes) may enter and leave the network**
  - Some topics attract disproportionate publication and subscription demand
    - the number and size of values per topic (DHT key) can vary significantly
    - storage and query hot-spots where a few nodes become overloaded, which limits scalability

Proposal

- **DHT-Level: Group-based DHT named Parsley**
  - Supports standard operations
  - Each DHT node is a logical node composed of one or more physical nodes

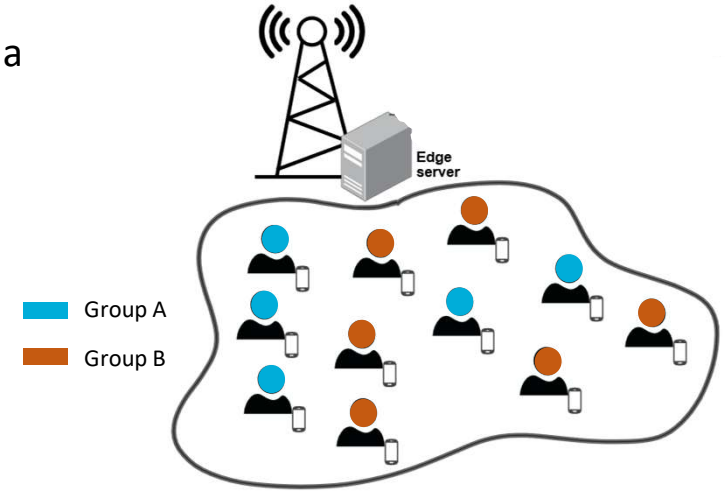
- IoT and content-sharing deployments rarely satisfy the uniformity assumptions of DHTs
  - Devices (nodes) may enter and leave the network
  - **Some topics attract disproportionate publication and subscription demand**
    - **the number and size of values per topic (DHT key) can vary significantly**
    - **storage and query hot-spots where a few nodes become overloaded, which limits scalability**

Proposal

- **DHT-Level: Group-based DHT named Parsley**
  - Supports the standard operations
  - Each DHT node is a logical node composed of one or more physical nodes
- **PubSub-Level: Dynamic data sharding of subscriptions and publications**

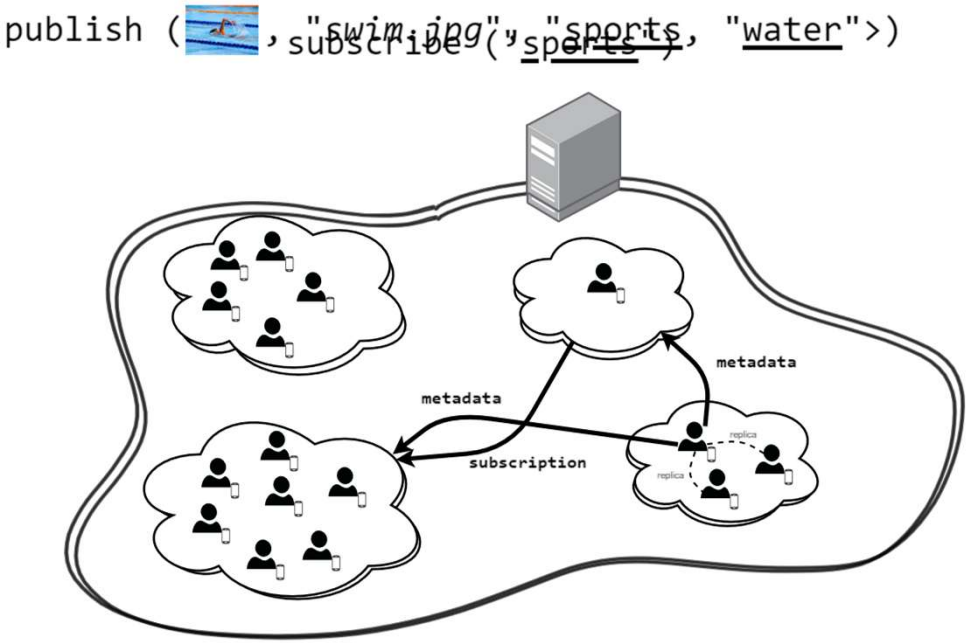
System Model

- Decentralized system of co-located (potentially mobile) devices
  - Connected wirelessly to an Access Point (AP) (e.g., via Wi-Fi).
- Data sharing via topic-based P/S service
  - Sustained by the devices distributed in a P2P network in the form of a group-based DHT
  - Aided by an edge server
- The failure model is crash-stop



PubSub

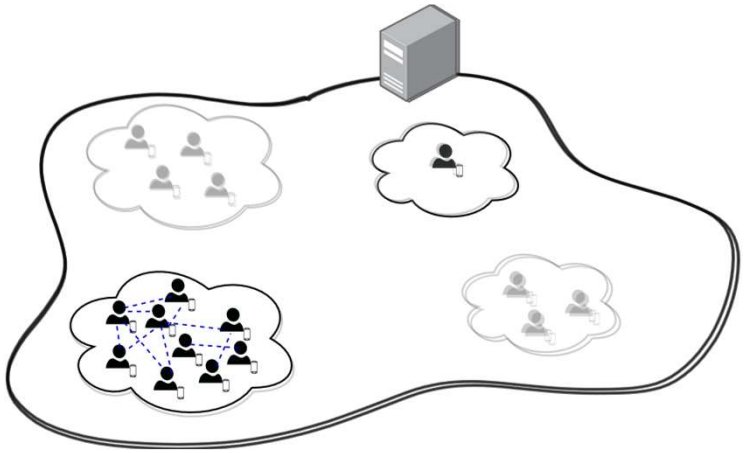
- Topic-based API
- Each group G manages at least one topic's data ($hash(topic) \rightarrow G$)
- Each group is responsible for:
  - Storing publications' metadata;
  - Store clients' subscriptions;
  - Match subscriptions with publications
  - Disseminate notifications
- The data itself is stored at the source
  - Each copy of the data becomes a new replica



Challenges

- Fixed group number limits the system scalability
- Dynamic nature of mobile devices disrupts groups' sizes

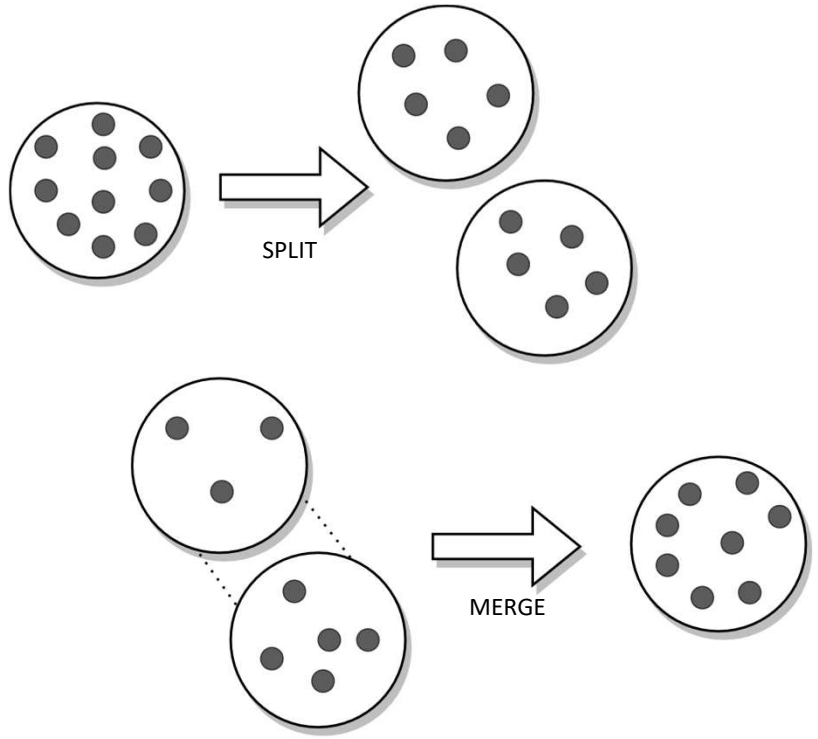
- Groups topology impact:
  - Underpopulated groups **leads to** permanent data loss
  - Overcrowded groups **leads to** traffic congestion



Dynamic group management

PARSLEY

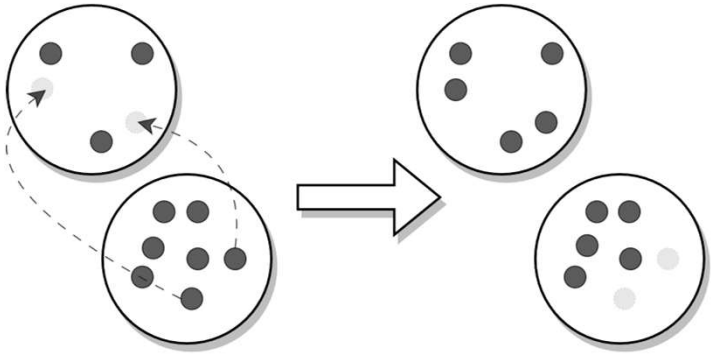
- Split groups
- Merge groups
- Maintain groups at an adequate size
- Prevent overloaded groups



Dynamic group management

PARSLEY

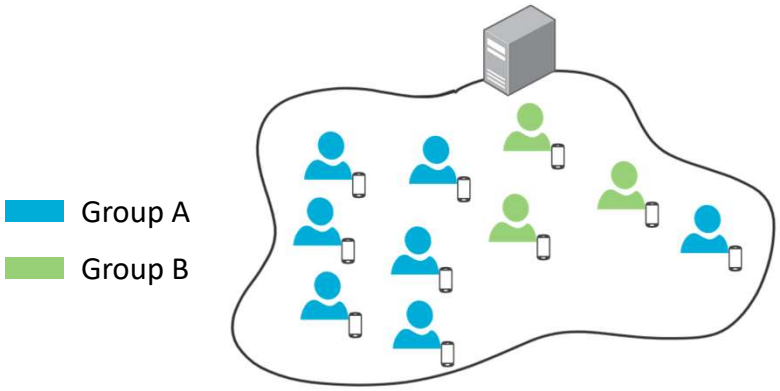
- Preemptive peer relocation
  - Push peers
  - Pull peers



PARSLEY

GROUP ASSIGNMENT

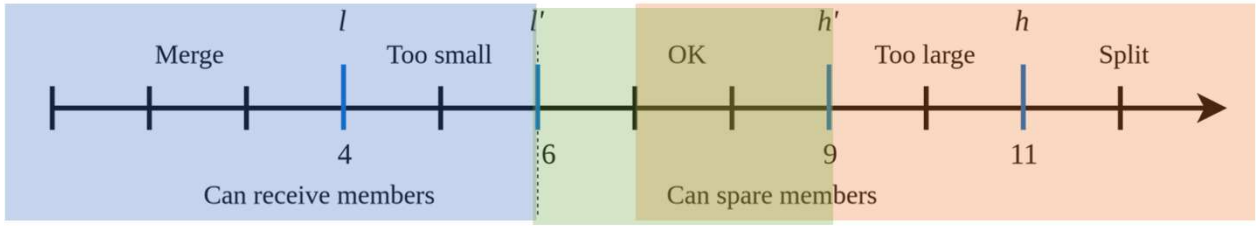
- Edge server responsible for initiating required protocols
- Group leader coordinates members during on-going operations
  - Device requests group to join
  - Edge assigns device to smallest known group
  - Presence announced to other members through *hearbeat* messages

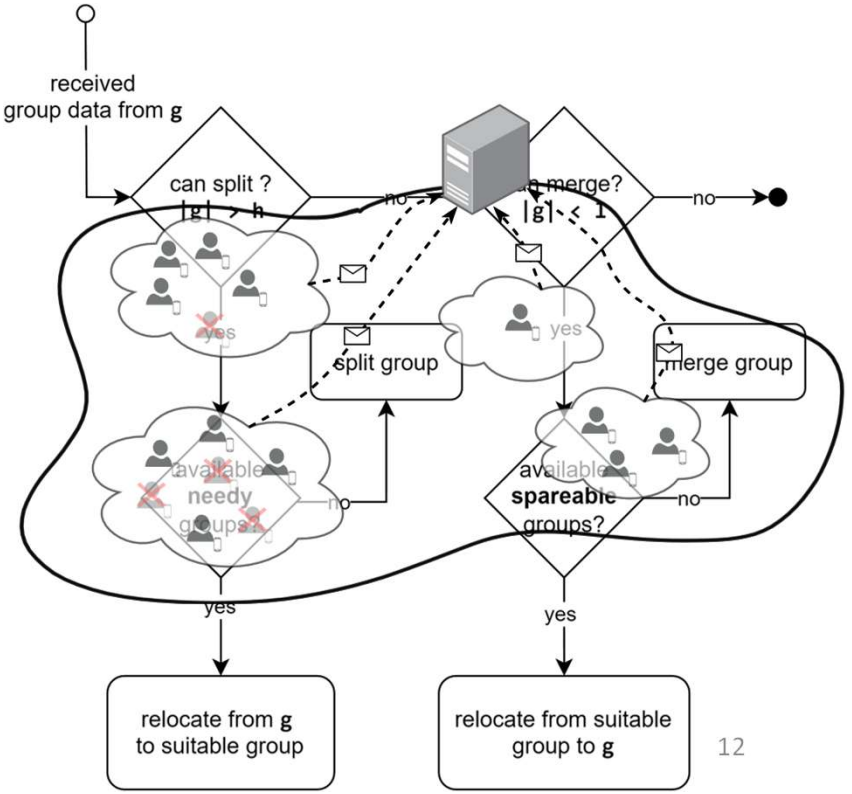


PARSLEY

GROUP MAINTENANCE

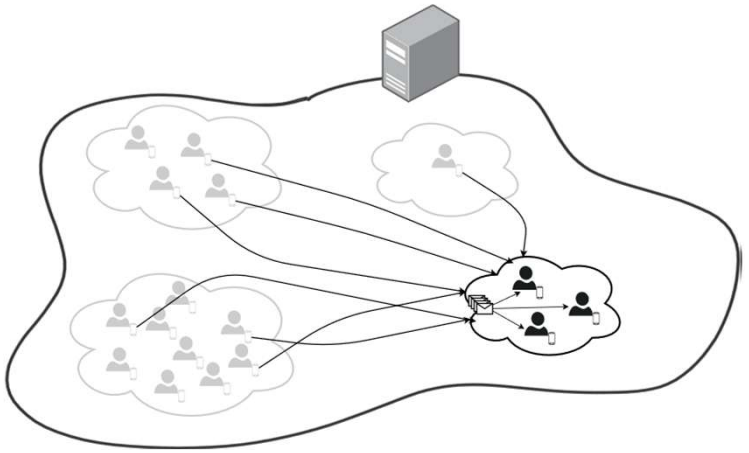
- Server aims to maintain groups within the ideal limits (**OK**)
- Groups periodically send relevant information to the server
  - e.g., recently left members
- Update groups' membership and their peer needs
  - **Needy** groups – min. and max. to receive
  - **Spareable** groups – min. and max. to spare
- Server checks for procedures every round of communication





Challenges

- Popular topics may accumulate excessive **publications' metadata** or **subscriptions**
- Excessive use of memory and inter-group communication in groups responsible for managing popular topics.
- The process of matching and notifying lots of subscribers may become a computationally heavy process

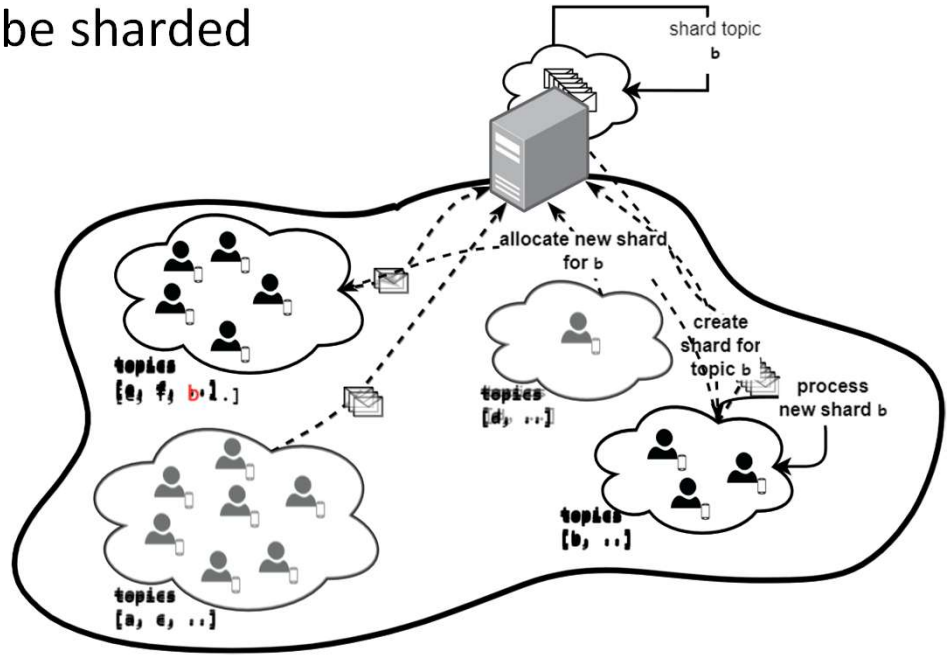


Dynamic data sharding

- Distribute popular topics' items storage across other groups
- Sharding policies to determine topics' popularity
- Configurable hotspot detectors
  - Absolute Amount
  - Absolute Size
  - Relative Size
  - Higher-Than-Average (HTA)
- Individual configuration of policies for publications and subscriptions

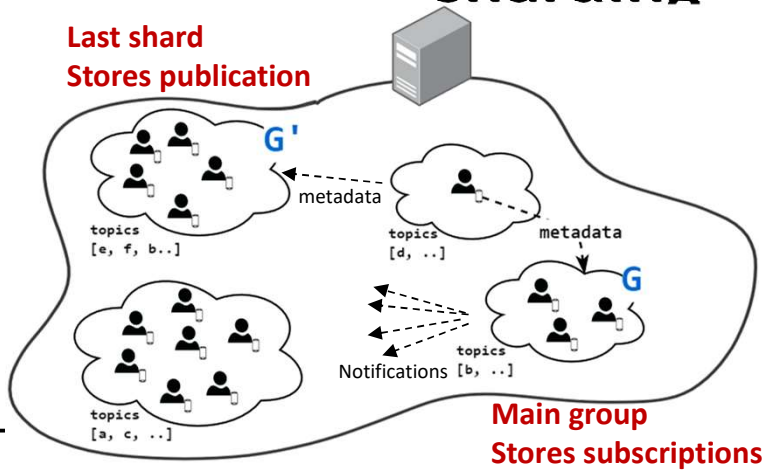
Dynamic data sharding

- Server collects groups' metrics per topic
  - **Publications' metadata**
  - **Subscriptions**
- Each round, the server checks if a topic can be sharded
- Assign another group for storing the topic's subsequent (popular) items until it is **full**



- Revision of client-side (and server-side) P/S operations
  - e.g.: **sharded** metadata and **single** subscriptions
  - Alternative **store** and **notify** destinations

Dynamic data sharding



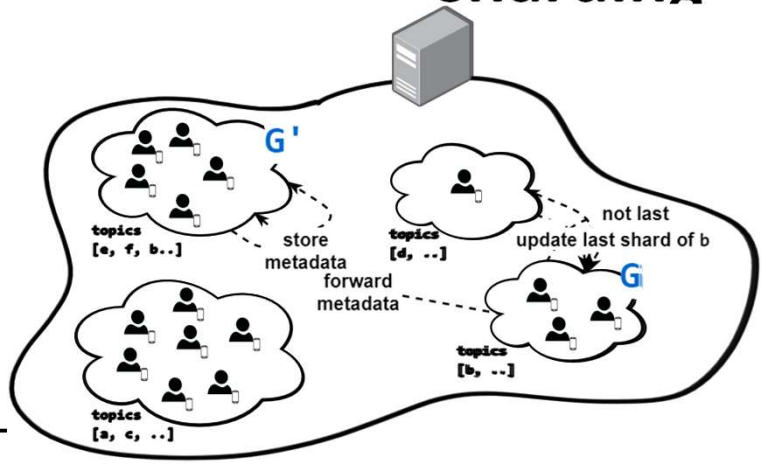
Topics state scenarios

metadata \ subscriptions	**Single**	**Sharded**
Single	Main* group stores and notifies	Main group stores Notify from shards of subscriptions
Sharded	Store in last shard of metadata Main group notifies	Store in last shard of metadata Notify from shards of subscriptions

* hash(k) → Main group

- Revision of client-side (and server-side) P/S operations
  - e.g.: **sharded** metadata and **single** subscriptions
  - Alternative **store** and **notify** destinations

Dynamic data sharding



Topics state scenarios

metadata \ subscriptions	**Single**	**Sharded**
Single	Main* group stores and notifies	Main group stores Notify from shards of subscriptions
Sharded	Store in last shard of metadata Main group notifies	Store in last shard of metadata Notify from shards of subscriptions

* hash(k) → Main group

Evaluation

- Group management
  - **Impact of topological operations** – How do the number of splits, merges, and relocations triggered by the mechanism vary under different churn rates?
  - **Group size variation** – How does the average group size change under different churn rates?

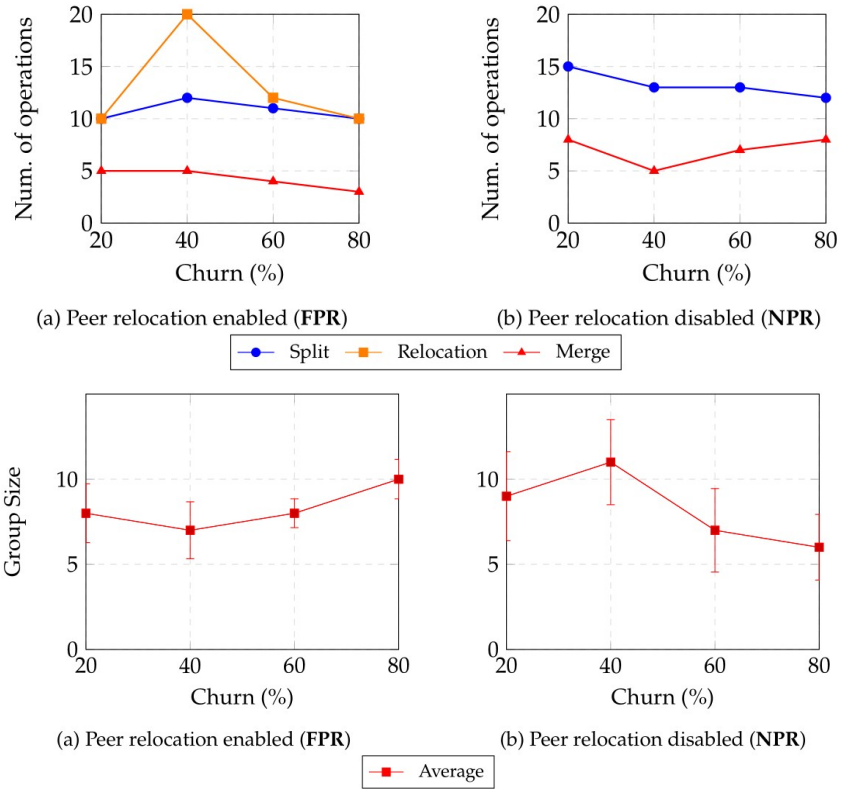
- Data sharding
  - **Load distribution** - How is memory usage affected in groups managing sharded topics under different sharding strategies and topic popularity skews?
  - **Network overhead** - How does the number of publish/subscribe messages vary across executions?

Dynamic group management

TOPOLOGICAL OPERATIONS VS. GROUP SIZE VARIATION

- Number of topological operations (split, relocate, merge) under varying churn rates
- Comparison between scenarios with and without the peer relocation mechanism
- **Peer relocation promotes efficient and minimal-disruption rebalancing, outperforming a rigid split/merge-only approach**
- **Average group size more stable with FPR in comparison with NPR**

EVALUATION

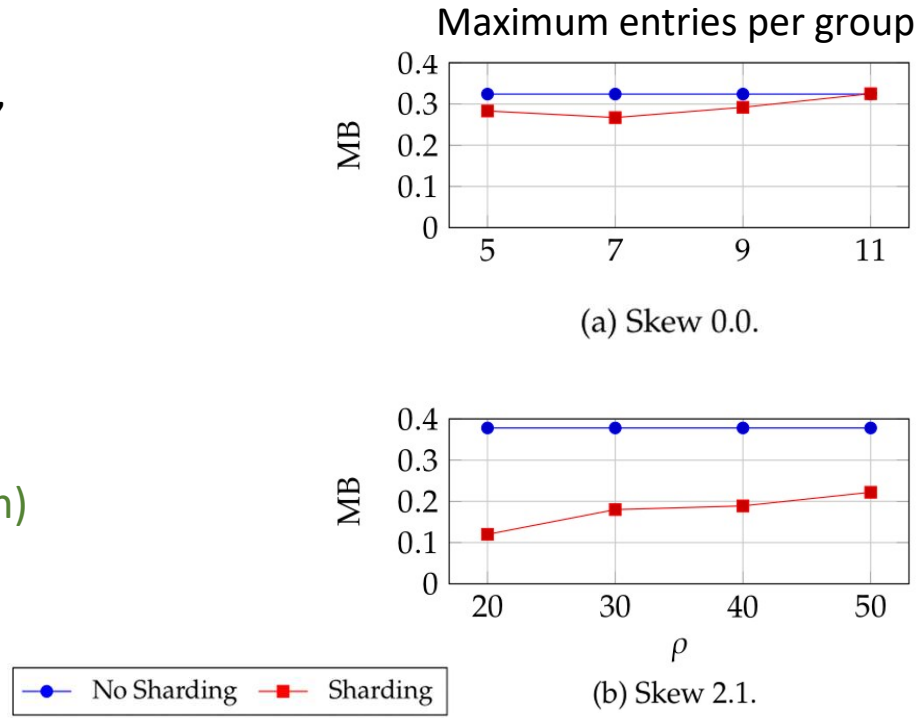


Dynamic data sharding

POPULAR TOPIC METATADATA
LOAD DISTRIBUTION

- Sharding policy configured for publications' metadata
  - Detector: **Absolute Amount** with parameter ρ **(number of entries)**
- Minimal impact with uniform distribution (i.e., $skew = 0$)
- Significant load balancing (~65% reduction) with trending topics (i.e., $skew = 2.1$)

Evaluation



CONCLUSION

- Proactive relocation
  - maintains stable group sizes
  - reduces merge frequency under churn
- Sharding mitigates storage hot-spots
  - lowering the maximum per-group load by more than half in skewed workloads.
- Although sharding increases message traffic, the overhead remains proportional to the number of popular topics and manageable.
- These results demonstrate that combining adaptive membership control with selective sharding yields a resilient and scalable solution for pubsub

FUTURE WORK

- Seamless transition between edge and edge-less scenarios
- Develop self-configurable sharding policies