Hedwig

A Shared-Nothing Message Broker

Hedwig Team
Yahoo! Research

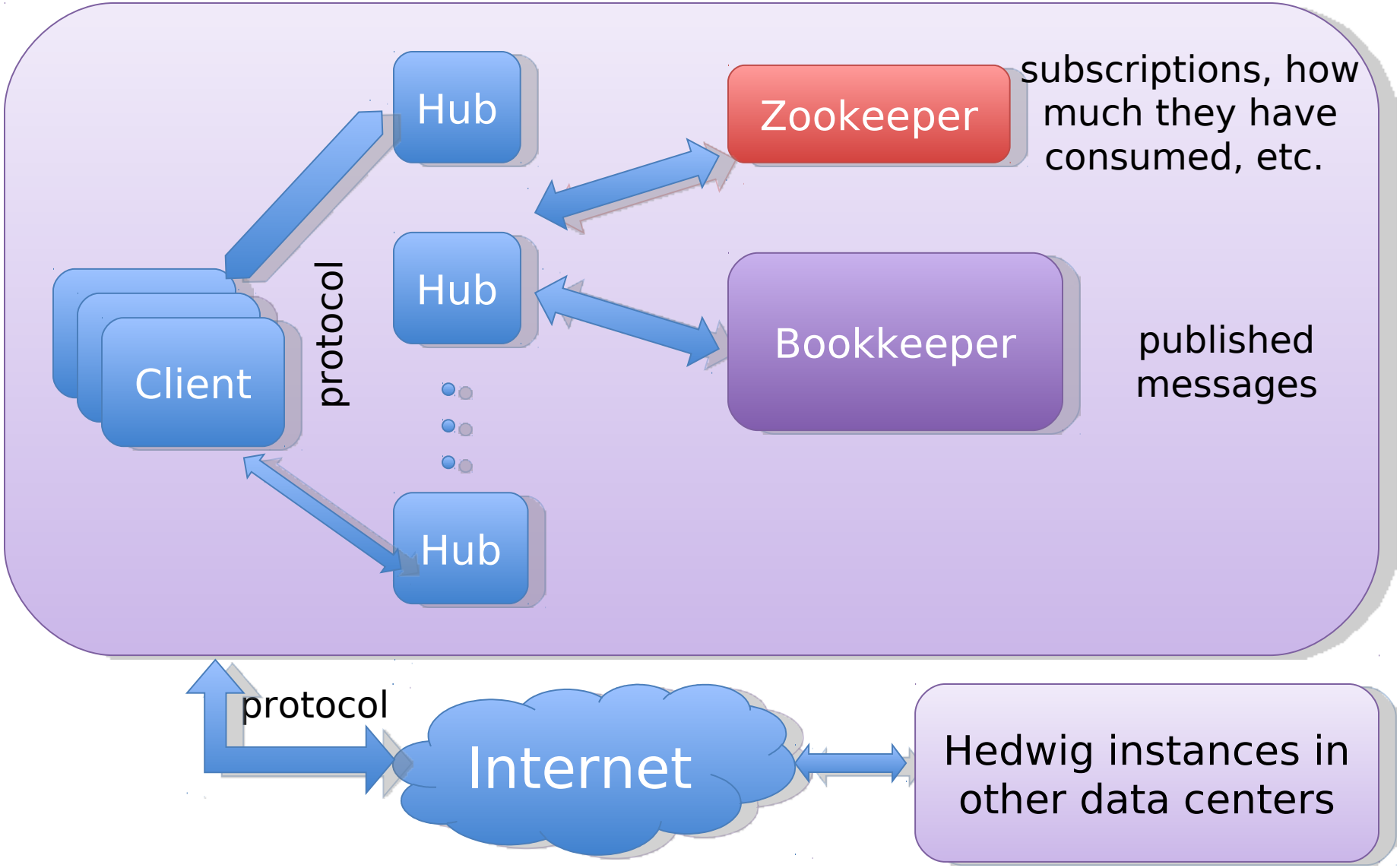
What Hedwig Offers

- Guaranteed-delivery topic-based pub-sub system
  - Durability: acknowledged published get delivered
  - Subscribers guaranteed to get all publishes after subscription (even if subscribers fail and come back)
  - Messages delivered in publisher order
- Elastically scalable
  - Deployed over commodity machines
  - Capacity can be added on-the-fly by adding machines
- Low Operational Complexity
  - Tolerate failures without manual intervention
  - Automatic load balancing
- Optimized for multiple data-centers

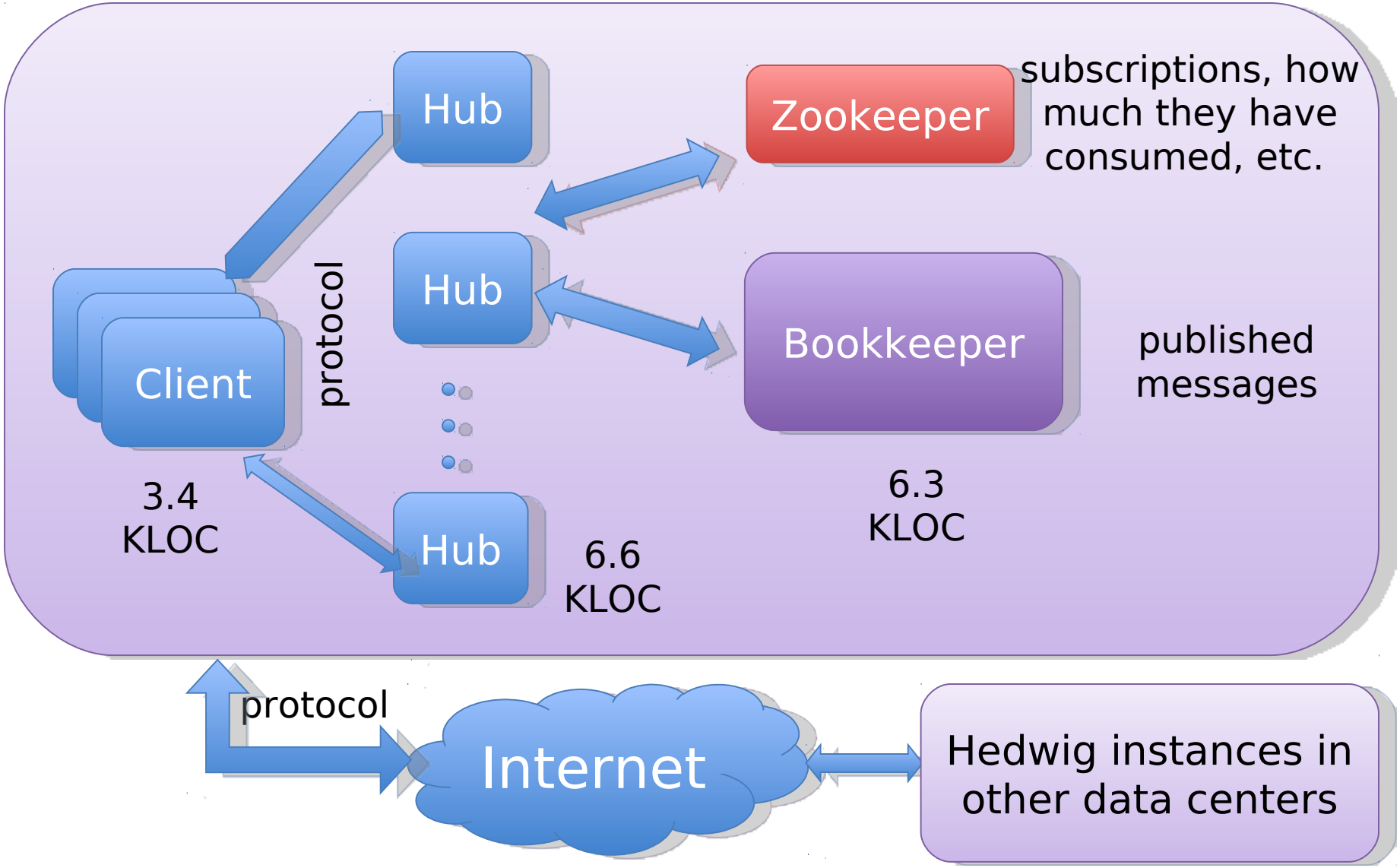
History

- Has been a relatively low-key effort
- Started gaining steam in June 2009.
- 2-2.5 engineers (+1 intern) over the summer.

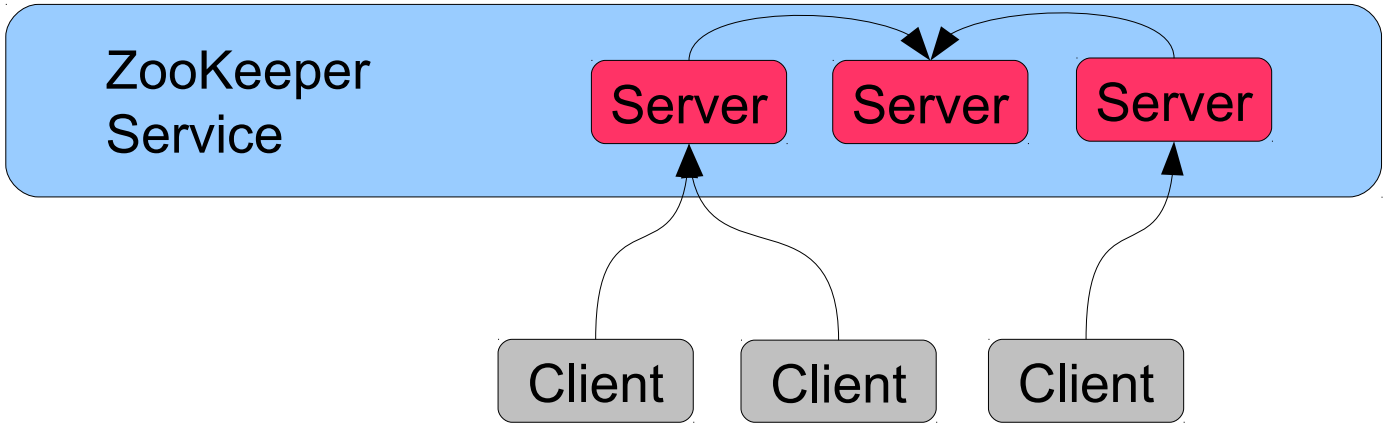
Architecture



Architecture



ZooKeeper coordination



- Data organized in a hierarchal namespace
- Nodes in the namespace, called znodes, can be persistent or empheral (znode will be deleted in service detects client failure)
- Strong durability guarantee
- Strong ordering guarantee
- API allows for clients to watch for changes
- Data stored in memory for low latency and consistent performance, but changes logged to disk for performance
- Reads processed using local server information, changes are linearalized through leader.

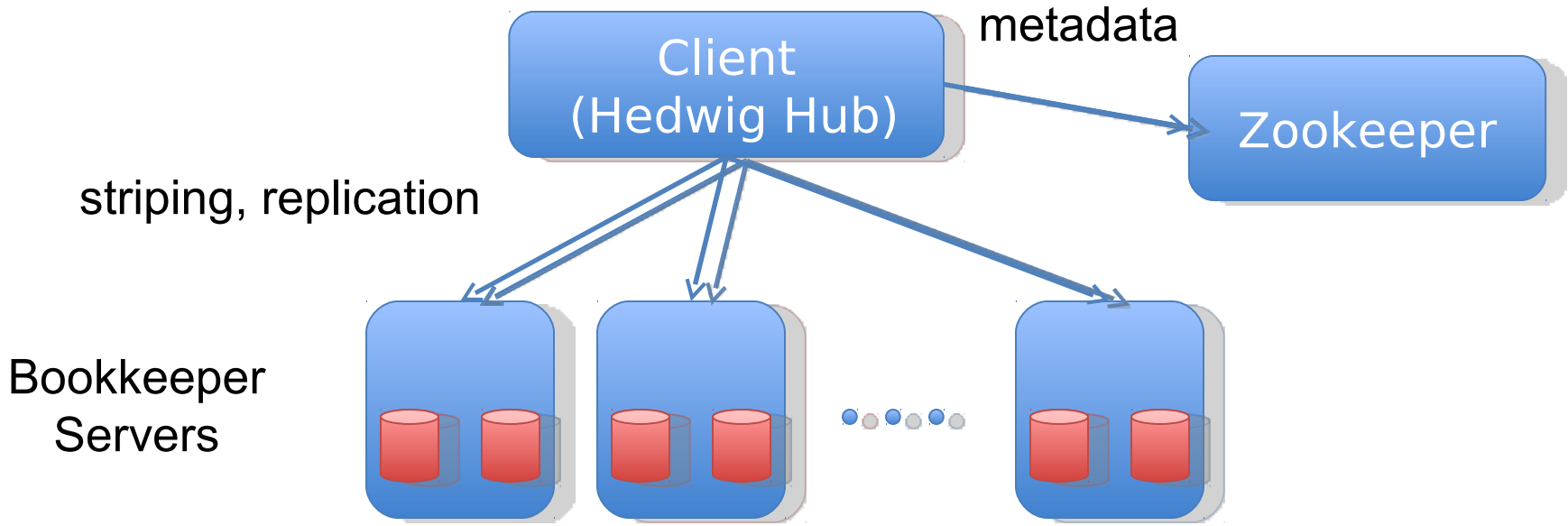
Zookeeper

- Used for configuration storage
  - Locations of ledgers
  - Subscriber information
- Topic leader election
  - Discovering topic leader
  - Ensuring single topic leader
  - Detecting topic leader failure
- Membership
  - Discovering available bookies

BookKeeper

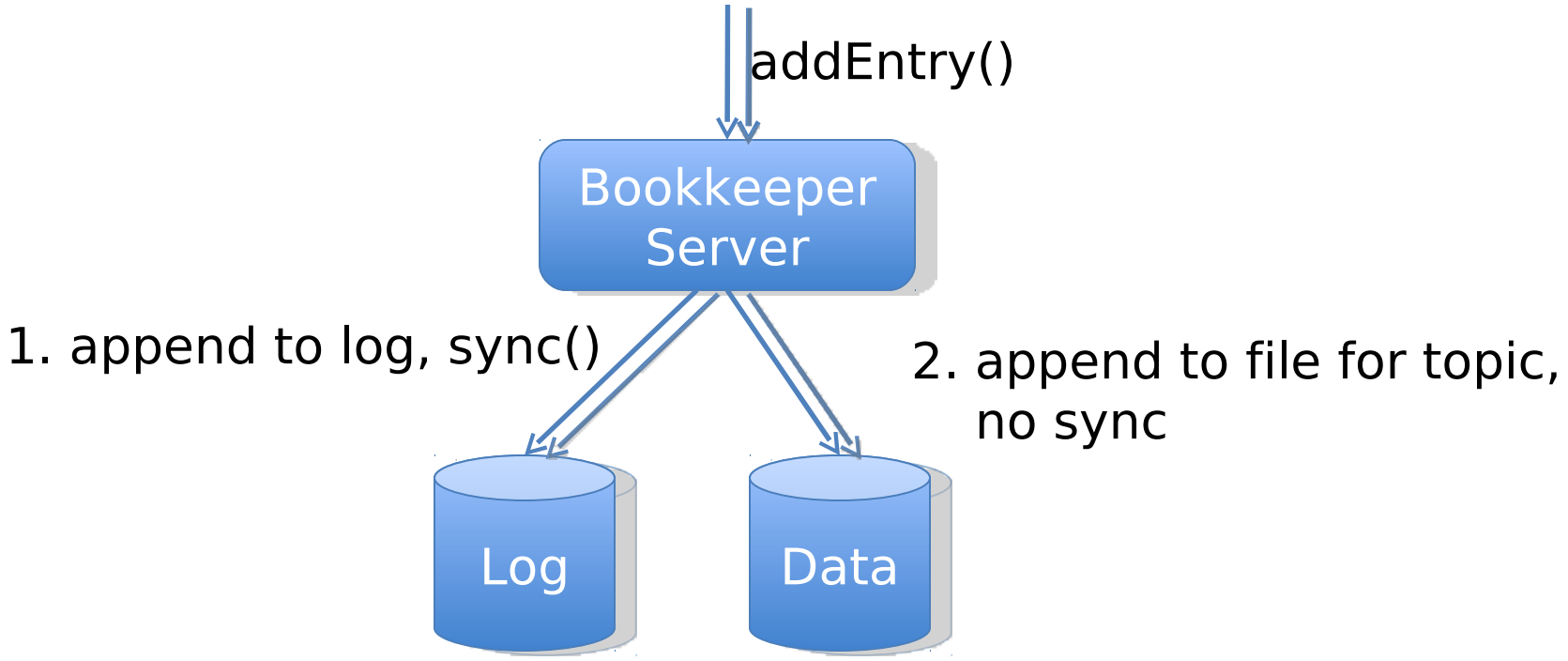
- Model a write-ahead log as an append-only sequence of entries, called a ledger
- Simple interface
  - create/openLedger
  - addEntry
  - readEntry
  - deleteLedger

Bookkeeper Architecture



- Like distributed RAID 1,0 (with append-only)
  - Configurable redundancy
  - Bookkeeper servers handle append-only, hence highly optimized.
- Open-source as contrib to Zookeeper

Performance

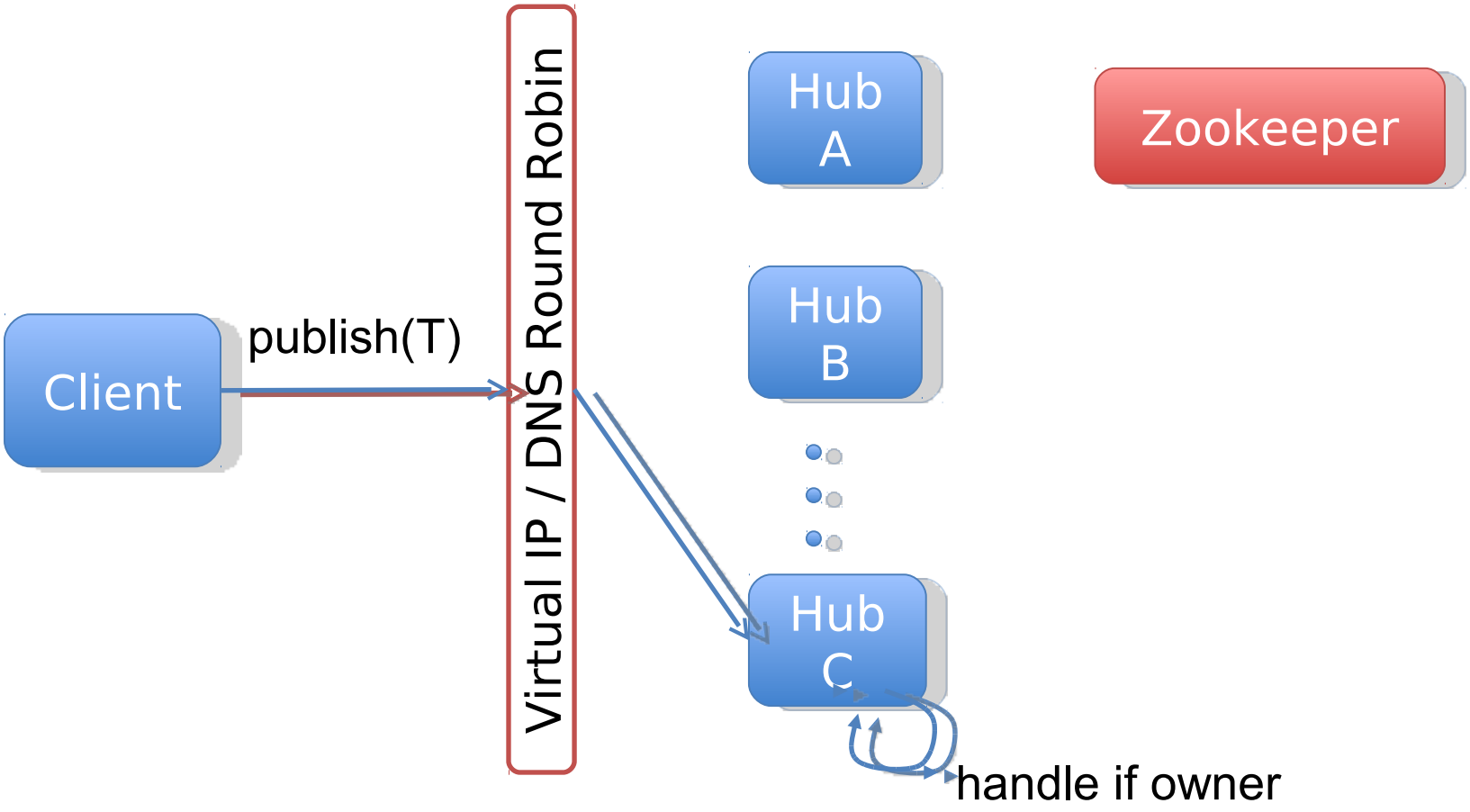


Can get close to sequential disk bandwidth due to group commit

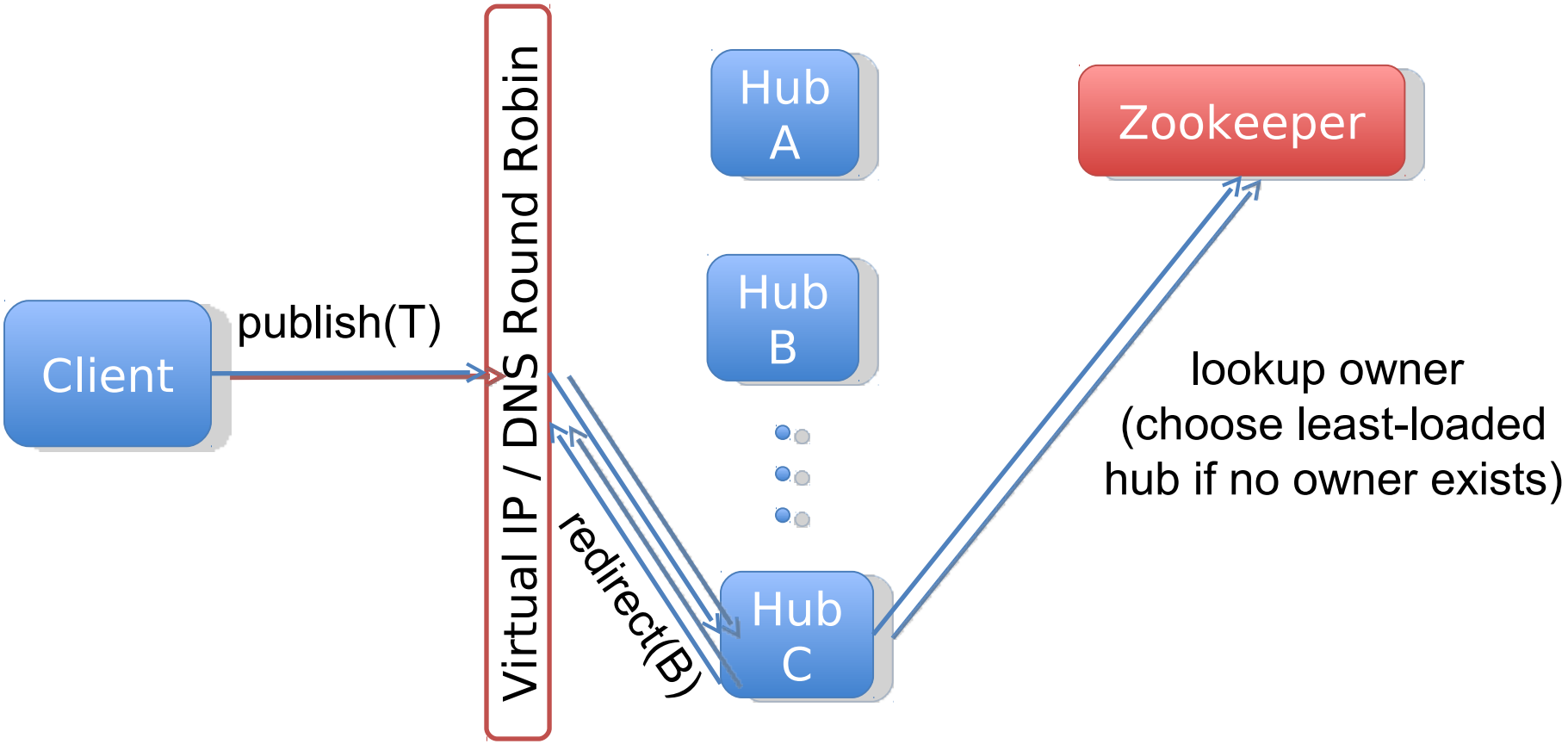
Hedwig Hubs

- Topics horizontally partitioned across hubs
  - A topics belong to only one hub (for per-topic ordering guarantees).
  - A hub can have multiple topics.
- Manages delivery to subscribers
  - Caches recently published data in process, to avoid trip to Bookkeeper.
- Subscribes to hubs in other data centers

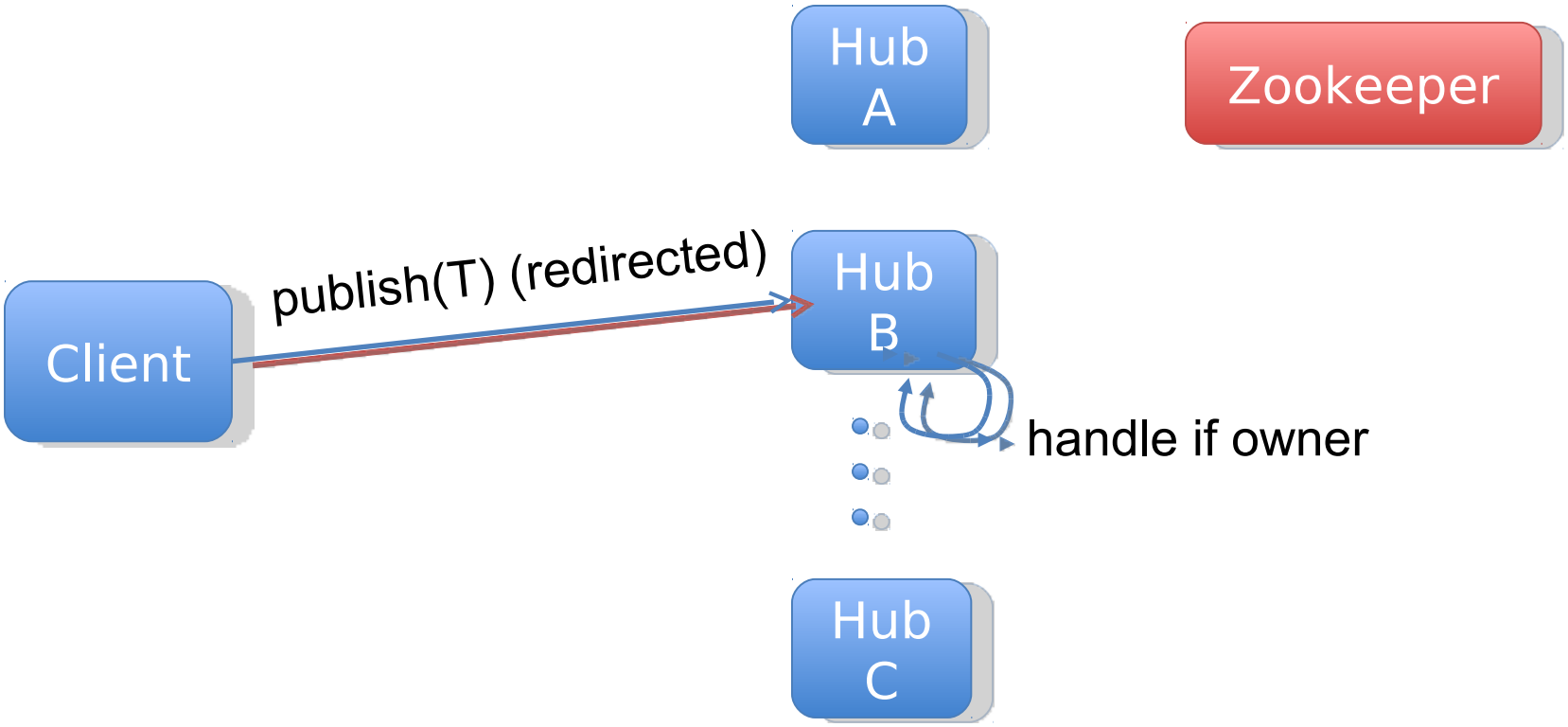
Automatic Failover



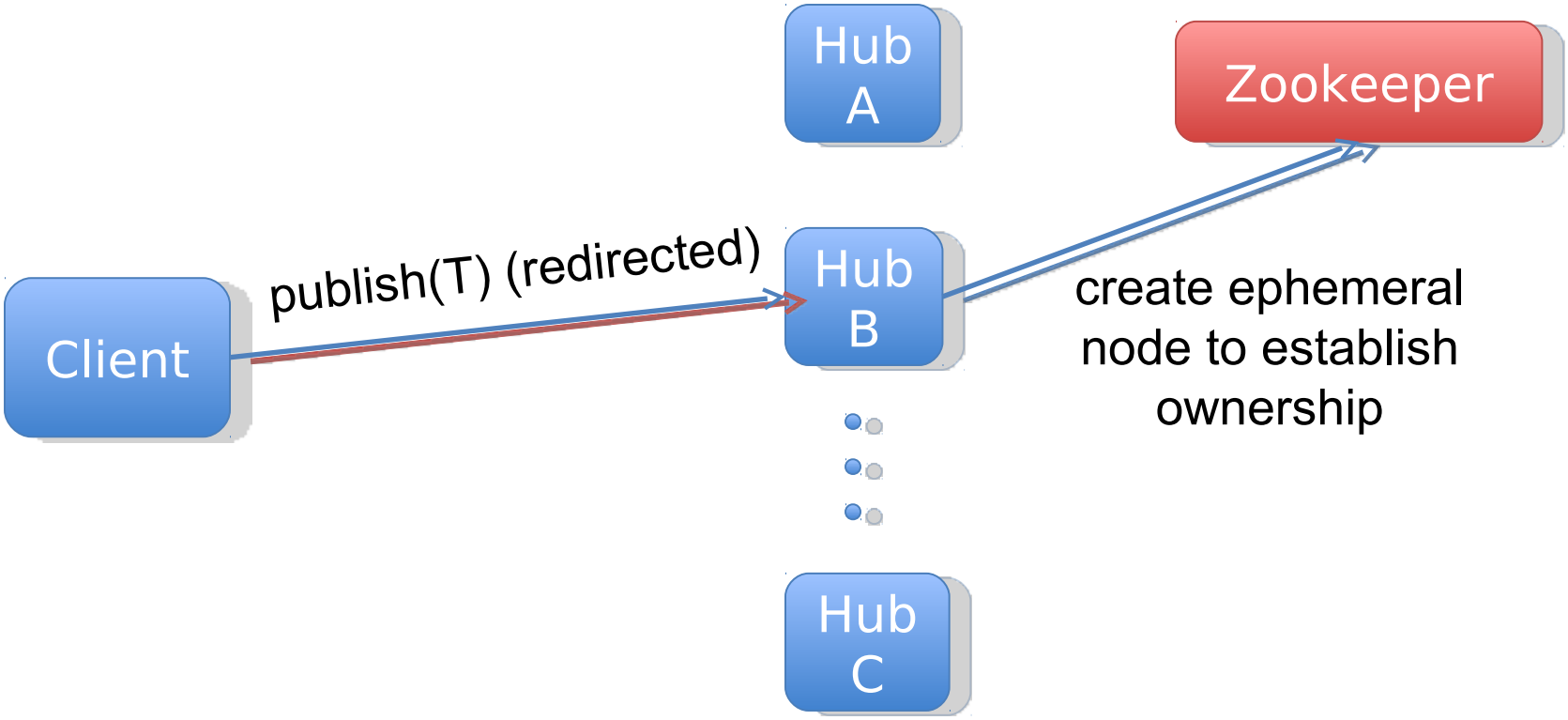
Automatic Failover



Automatic Failover

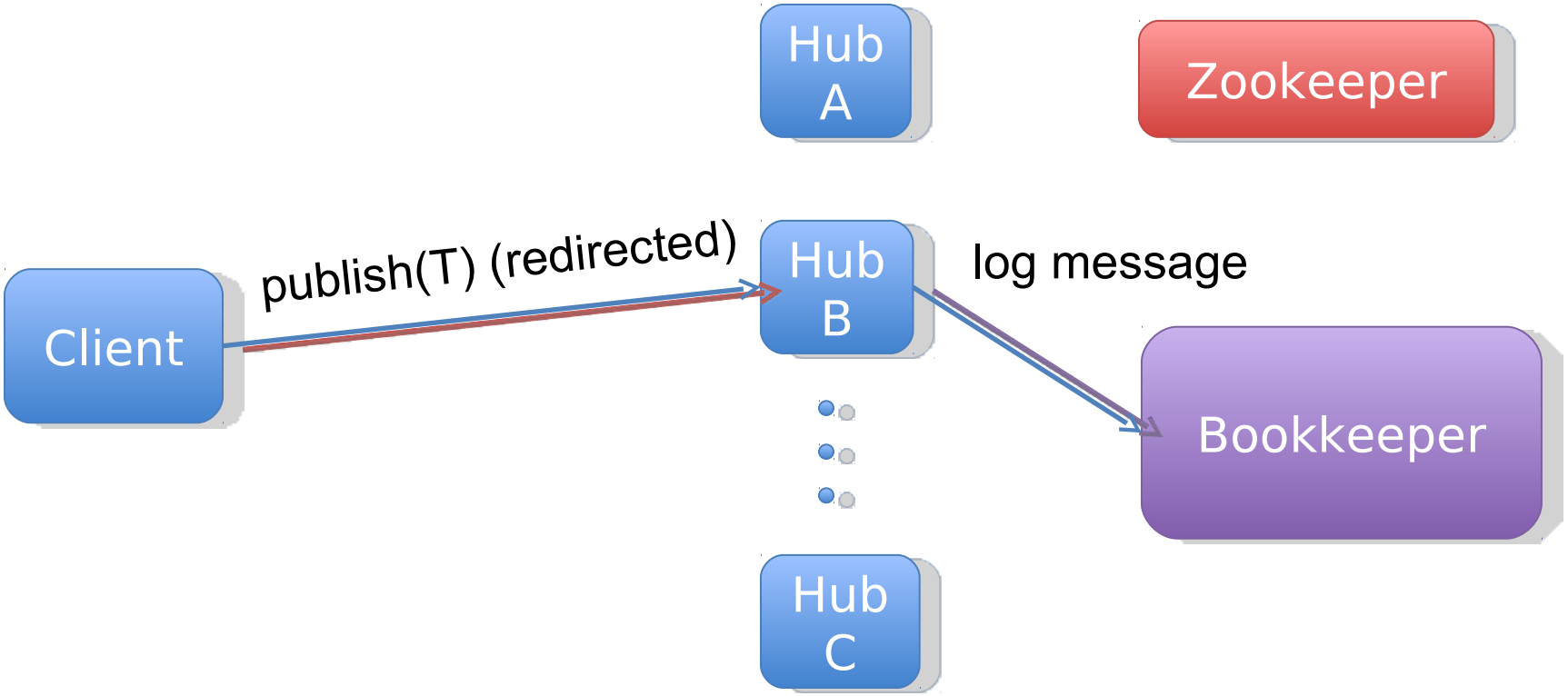


Automatic Failover

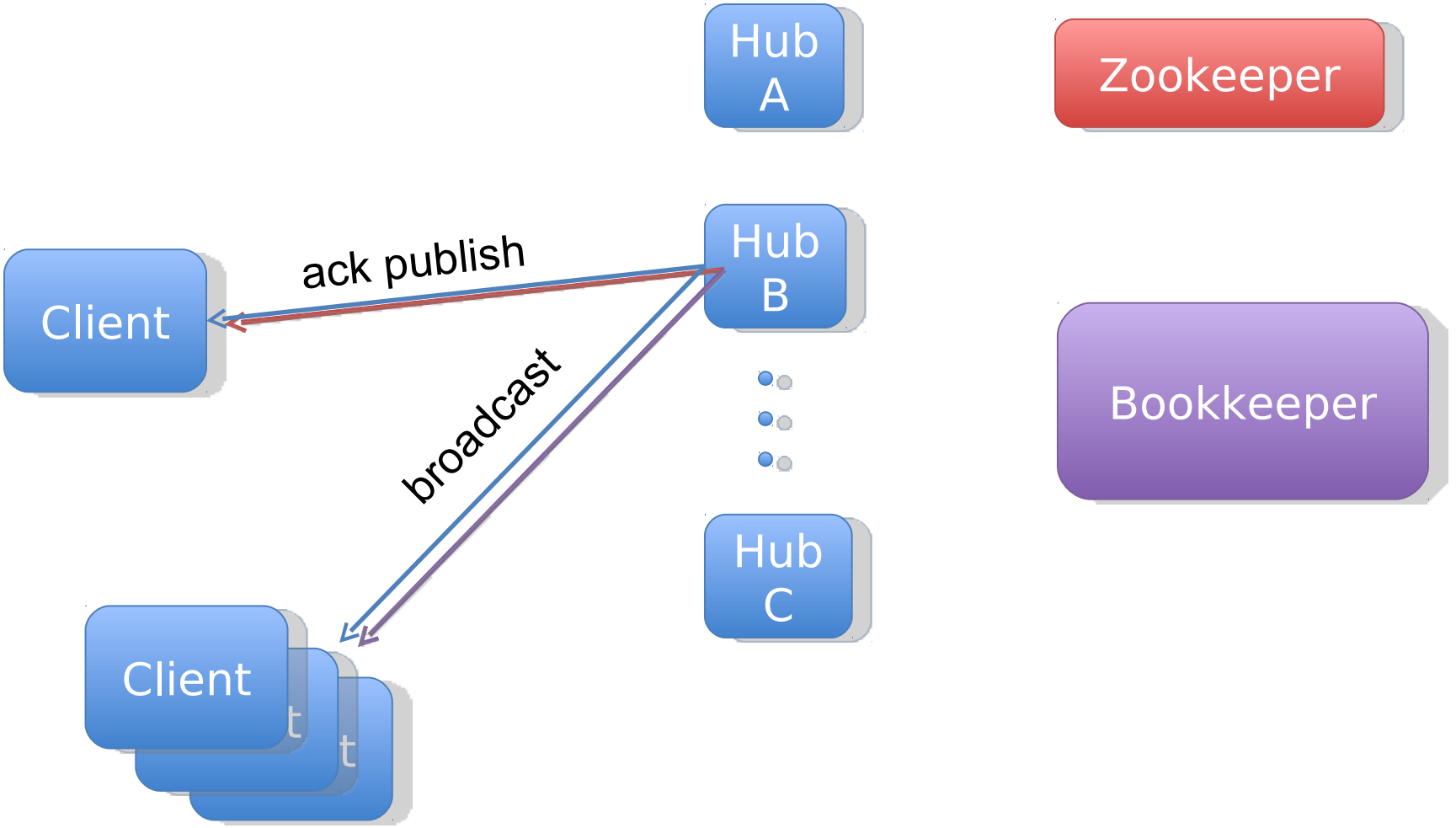


If B fails, ephemeral node disappears, and a new owner is chosen automatically

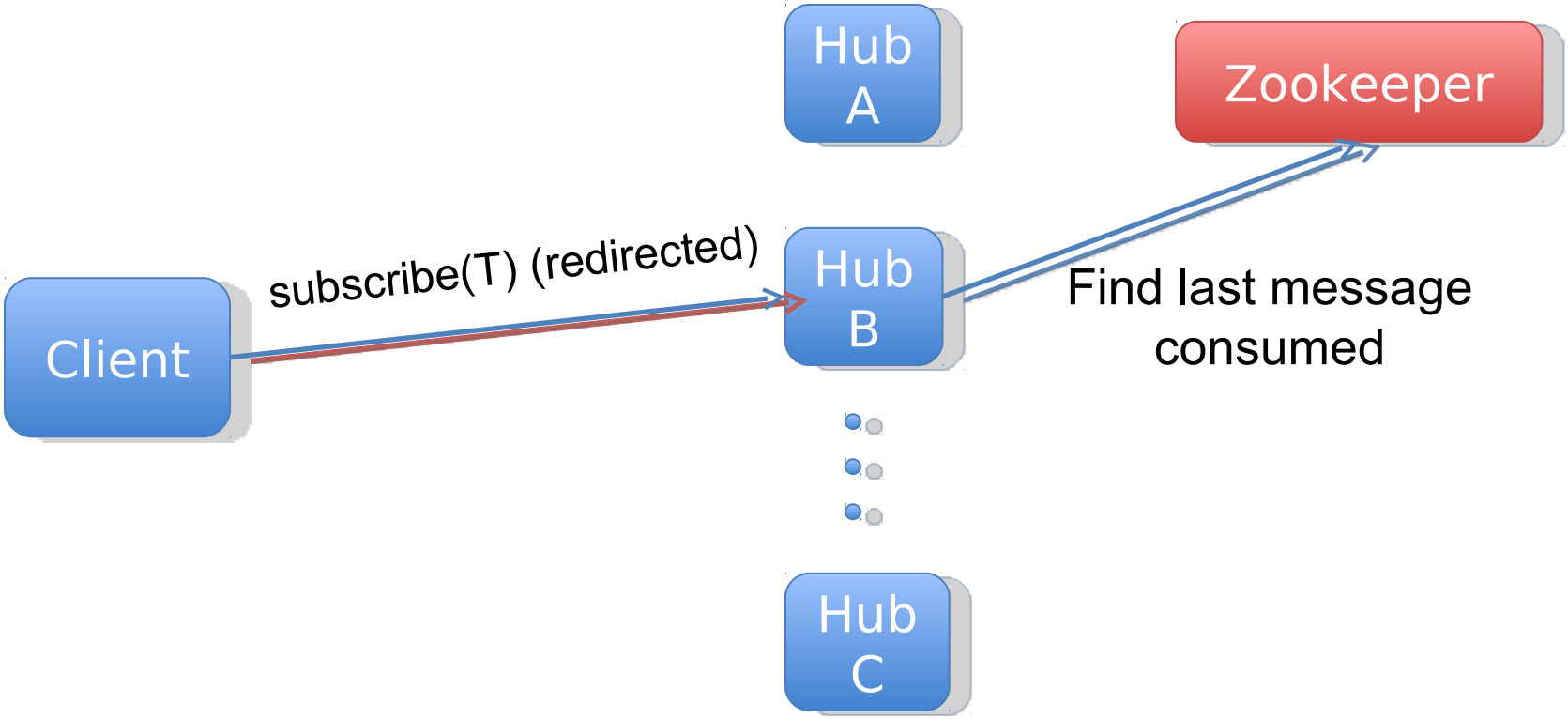
Publish


Hub
A
Zookeeper
Client
publish(T) (redirected)
Hub
B
log message
Bookkeeper
Hub
C

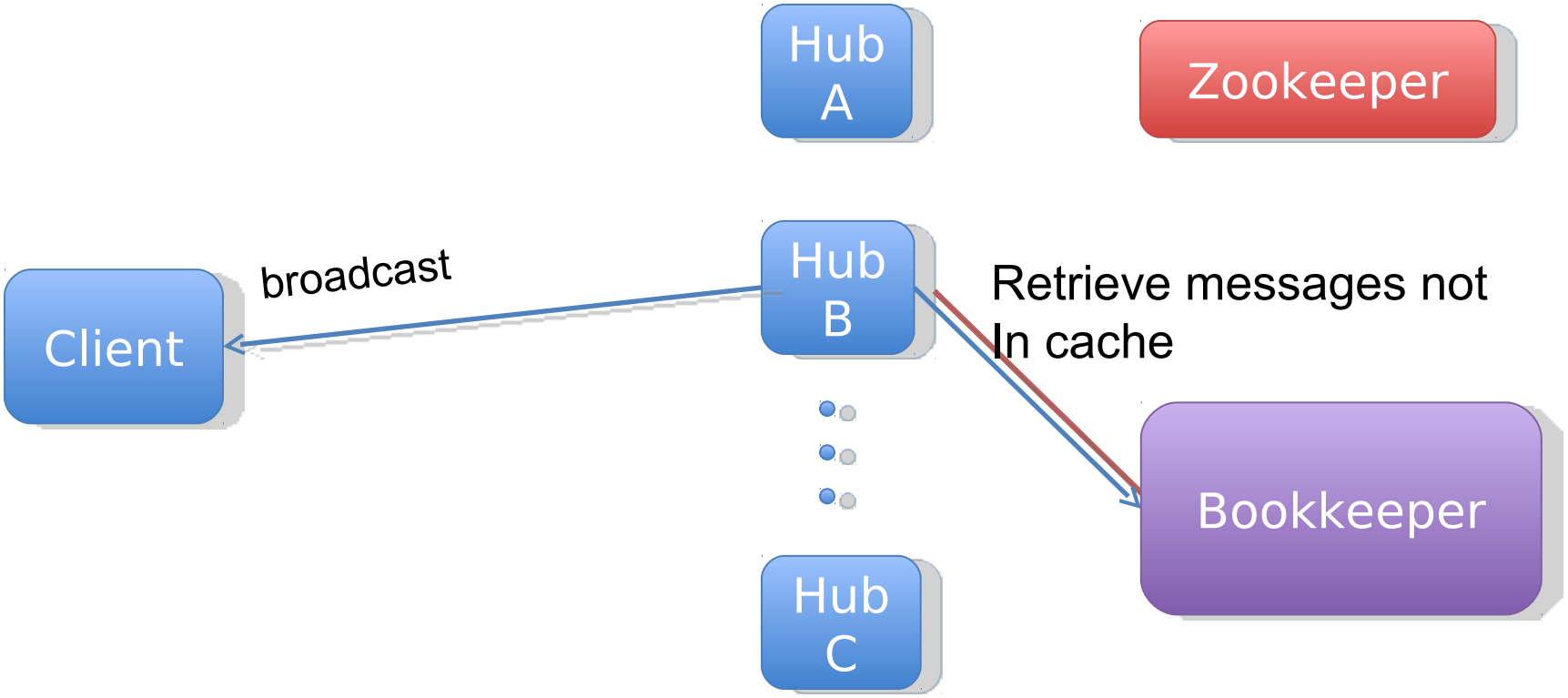
Publish


Hub A
Zookeeper
Hub B
ack publish
Client
broadcast
Bookkeeper
Hub C
Client

Subscribe


Hub A
Zookeeper
Client
subscribe(T) (redirected)
Hub B
Find last message consumed
Hub C

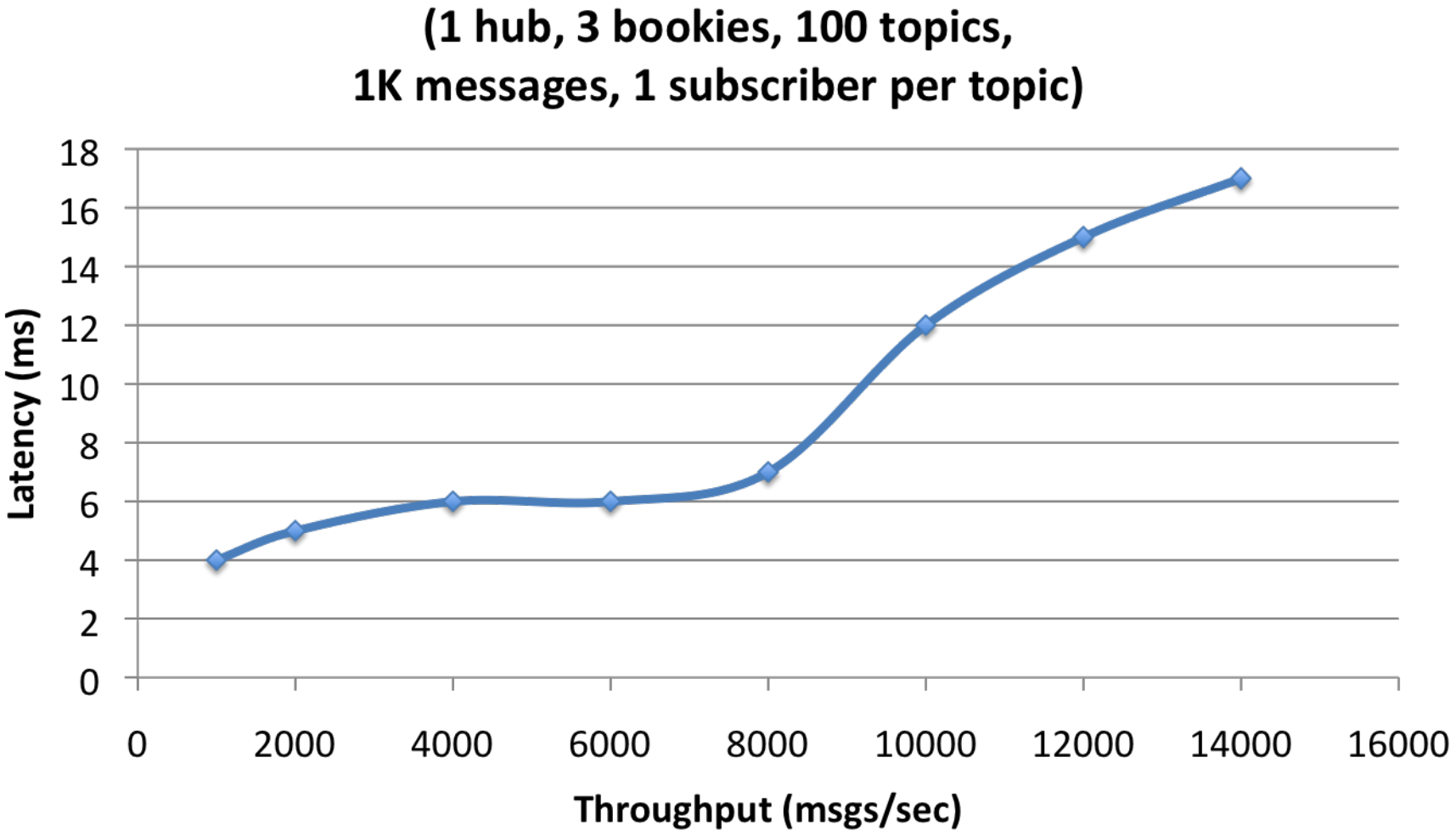
Subscribe


Hub A
Zookeeper
Client
broadcast
Hub B
Retrieve messages not
In cache
Bookkeeper
Hub C

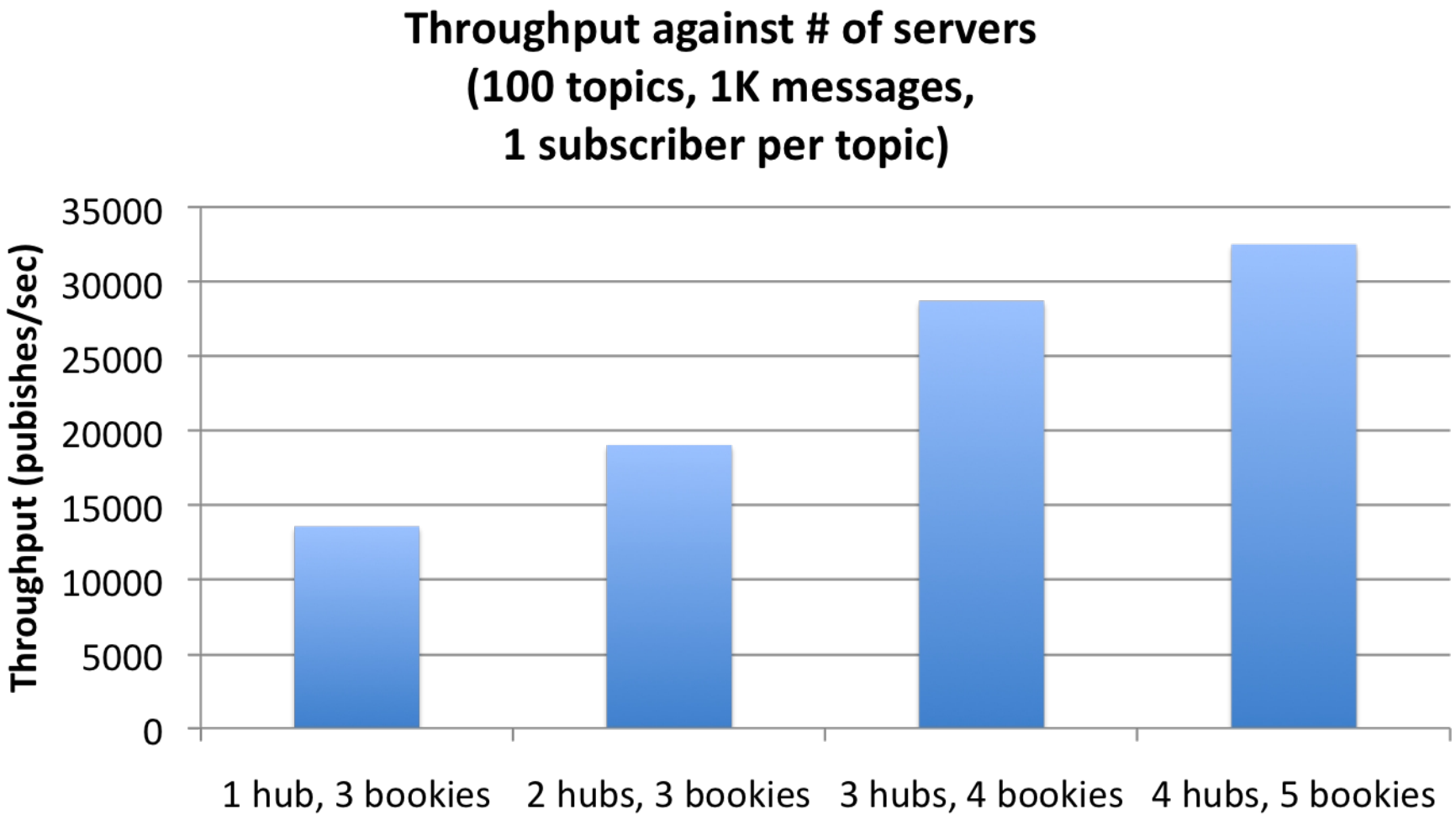
Testing

- Hardware
  - Old, relatively-crappy, commodity boxes
  - 2 cores, 2.13 GHz, 4GB RAM
  - 2 disks, 7.2K rpm SATA
- Most results on 4-box farm (1 hub, 3 bookies)
- Performance Tests
- Failure Tests
- Stability Tests

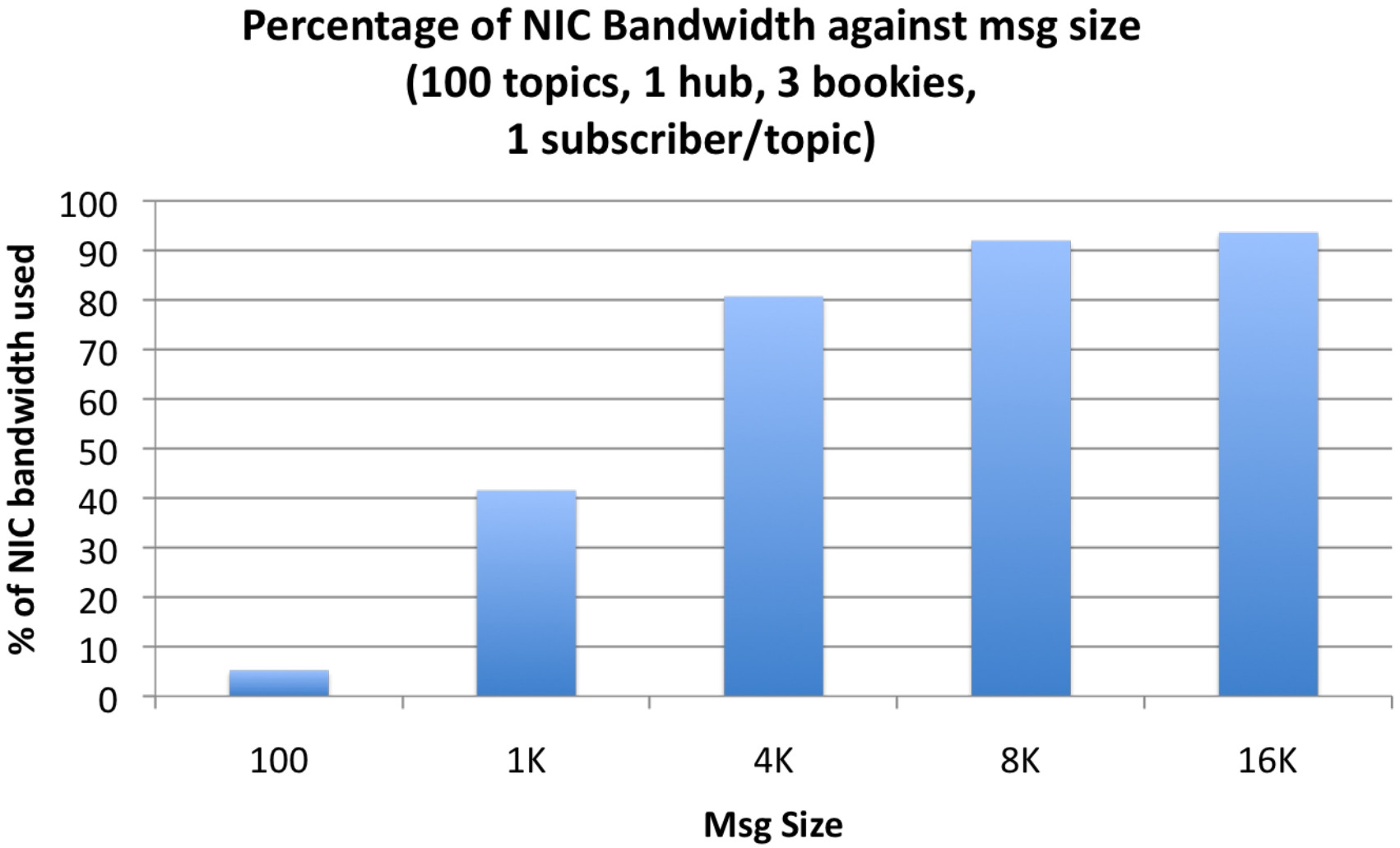
Performance (Latency v/s Throughput)



Scalability



Large message sizes



Failure Handling

- Able to shoot down a bookie
  - Operations continue without a single failure
- Able to shoot down a hub
  - Operations going to that hub fail, but only for a second (depending on our ZK timeout)
  - Topic gets taken up automatically by another hub

Stability

- Able to run the system for days without anything going wrong.
- Recovery tools done, but just started testing.

Recent improvements

- Scaling with number of topics.
  - Currently every topic gets its own file, which doesn’t scale.
  - Patch in progress to share files among topics
  - Preliminary numbers indicate scalability of up to 10s of thousands of topics per bookie

Recent improvements

- Collection of consumed logs
- Bookie recovery

What's Missing

- C++ Client Library
  - Have an initial implementation
- JMX binding
- Operational/Monitoring tools (1.5 months)
  - A promising approach is to write adapters so that existing tools just work.

Harder, Longer-term Things

- Notifications (3-4 months)
- Adaptive replication
  - Relatively easy with current design (3 weeks)
- Support for non-star topologies and changing data-center topology on the fly (4-5 months)

More information

http://zookeeper.apache.org