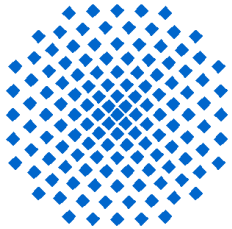


University of Stuttgart

Institute of Parallel and
Distributed Systems (IPVS)

Universitätsstraße 38
D-70569 Stuttgart

Replication of Data in MANETs with respect to the Global Temporal Ordering

Jörg Hähner

23.07.2003

Outline

- Motivation
- Consistency model
  - Definition
  - Examples
- Algorithm
  - Key features
  - Performance in a nutshell
- Summary

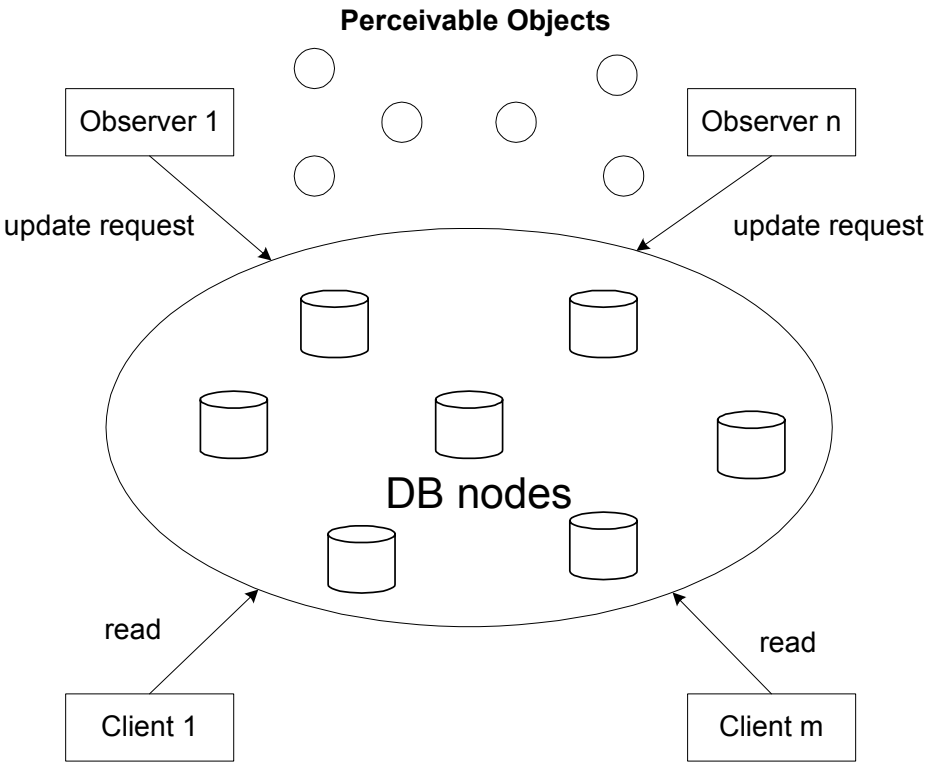
Motivation

- Task: monitoring of real-world objects
  - Sensors may be used to capture objects‘ state
  - Real-time order of observations important
  - Objects may be observed by independent sensor nodes
  - Typically highly distributed
- Many applications potentially need
  - Access to a global state of all aspects captured
  - High availability of state
- Spectrum of approaches
  - Complete replication on user devices → local queries
  - Partial replication
  - No replication

  } Remote queries

Consistency Model: Entities

- Perceivable objects
  - Unique identifier
  - State relevant to the application
- Observer nodes
  - Contain sensors that monitor the state of objects in their vicinity
  - Transmit state changes of objects → *update requests*
- DB nodes
  - Maintain a copy of the most recent state of each object
  - Perform synchronization among each other
- Clients
  - Perform read operations



Consistency Model: Definition of Ordering

- Real-time ordering only possible with limited accuracy
- Definition: *occurred-before*

  u ***occurred-before*** *(<) u**, iff

  $observation\_time(u^*) - observation\_time(u) > \delta > 0$

- System parameter δ
  - Difference less than δ → ***concurrent*** update requests
  - Describes precision of ordering

Consistency Model

- Definition: *update-linearizability*

  ∃ a **serialization** S for the executions of all clients and observers:

  C1: All ***read operations*** of a single client on a single object in *S* are ordered according to the program order of the client.

  For each **object *x*** and each **pair of update requests** *u*[*x*] and *u'* [*x*] on *x* in *S*:
  u' [*x*] is a (direct or indirect) successor of *u*[*x*] in *S* iff
  u[*x*] < *u'* [*x*] or *u*[*x*] || *u'* [*x*].

  C2: Each object o in the DB *S* meets the specification of a single copy of *o*

Consistency Model: Example

a)	O1: u[x]1				Correct
	O2:	u[x]2			
	C1:		r[x]1	r[x]2	
	C2:		r[x]2		
b)	O1: u[x]1	u[y]2			
	O2: u[y]1	u[x]2			
	C1:		r[x]2	r[y]1	
	C2:		r[y]2	r[x]1	
c)	O1: u[x]1				Not correct
	O2:	u[x]2			
	C1:		r[x]1	r[x]2	
	C2:		r[x]2	r[x]1	

Algorithm: Data Structures

- Update request $u^o_s[x]$ contains
  - Unique ID of observer *O*
  - Sequence number *s* of observer *O*
  - Unique ID of object *x*
  - State information
- Ordering graph for an object *x*
  - Reflects temporal ordering of update requests on *x*
  - Vertex: update request
  - Edge u[x] → u‘ [x]: (u[x] < u‘ [x]) or (u[x] || u‘ [x])

Algorithm: Key Features

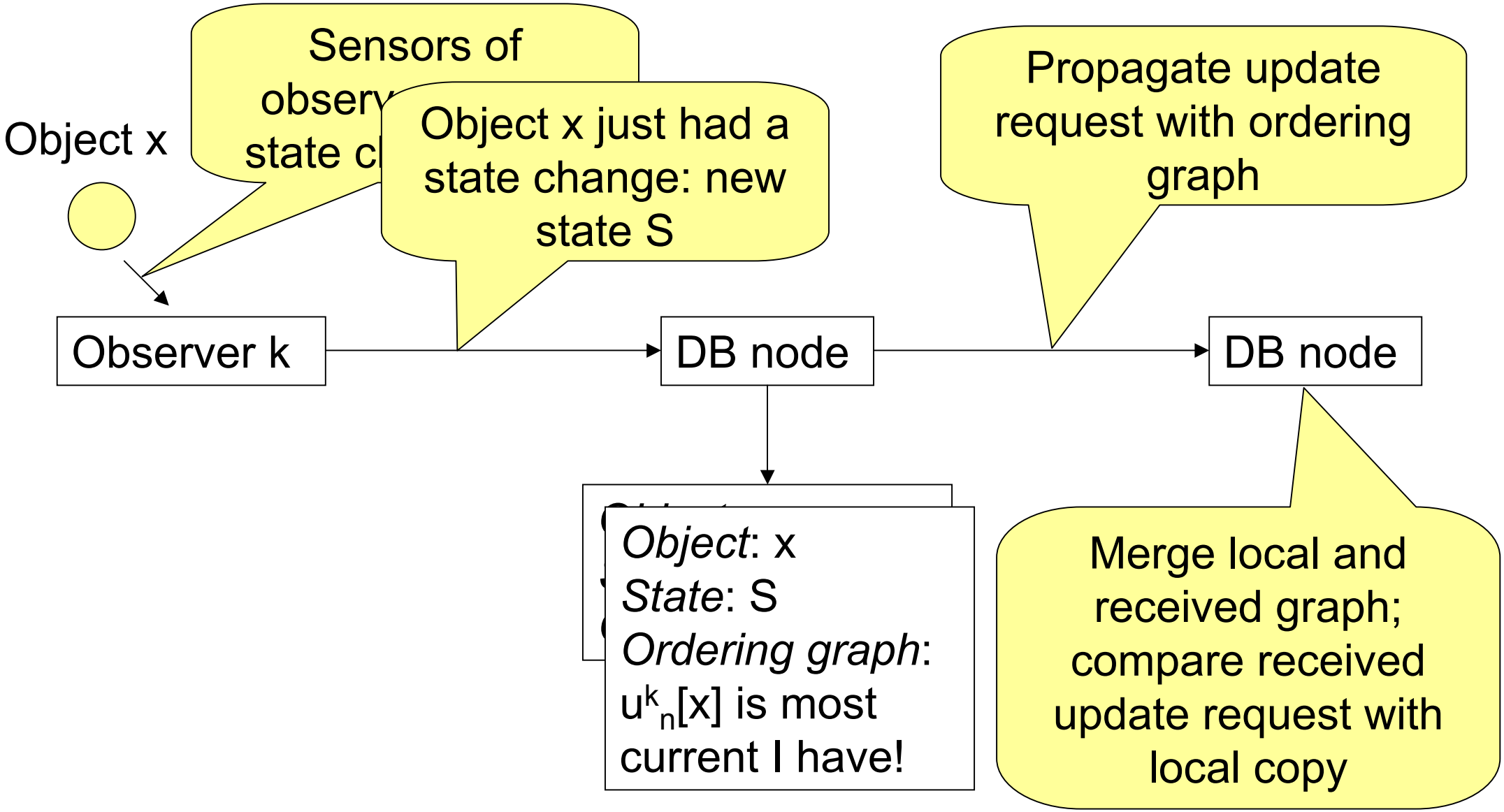
- Does not require synchronized clocks
- Observer nodes transmit update requests
  - Immediately as the corresponding state change occurrs
  - Only to DB nodes that are one-hop away
- Assumption: *state propagation delay* is bounded
  - Maximum jitter defines accuracy of ordering

Observer-node protocol

- DB nodes maintain ordering graph
  - For update requests of each object
- DB nodes synchronize with each other
  - Propagate update requests with ordering graph
  - Merge ordering information

Node-node protocol

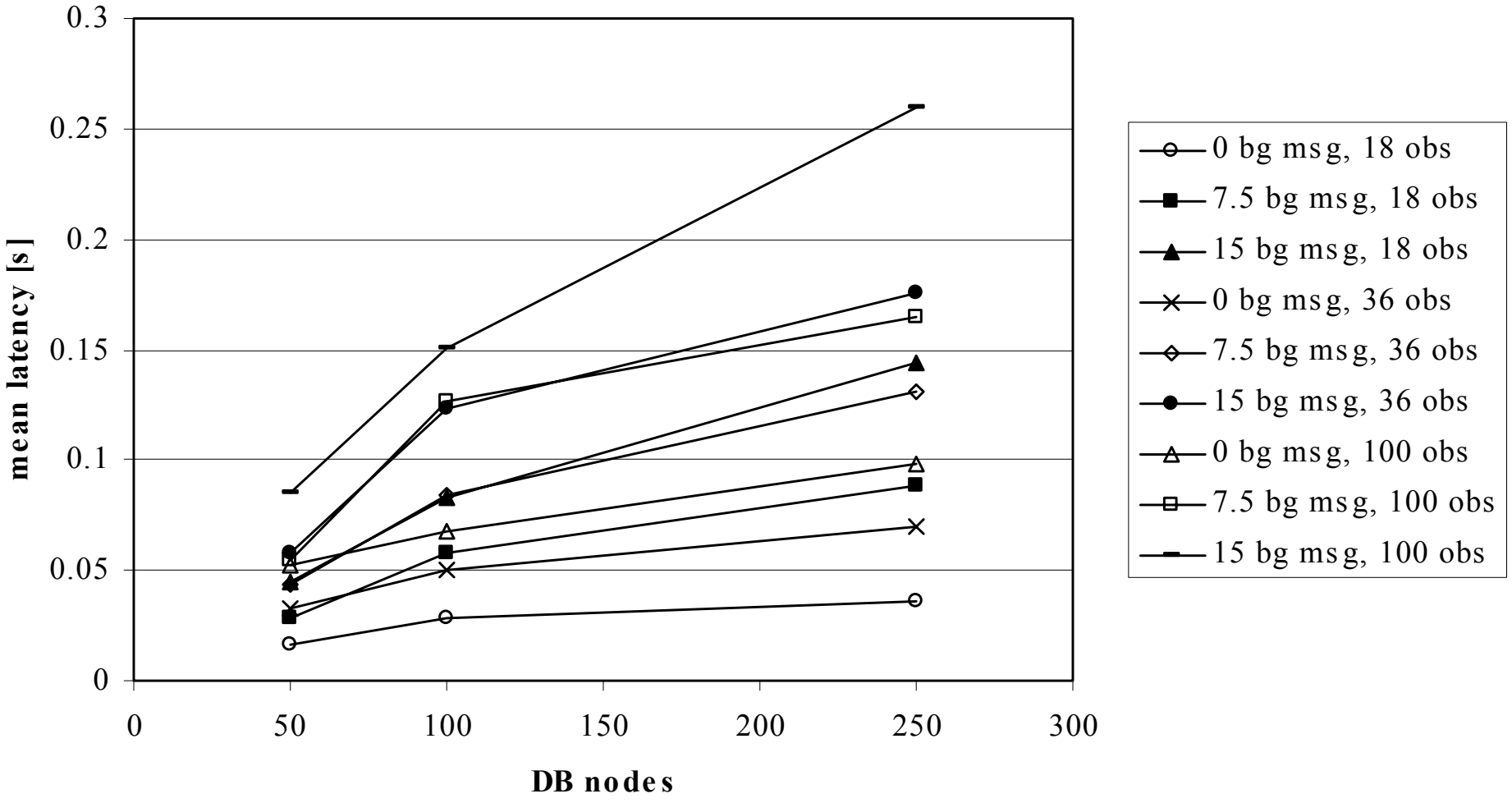
Algorithm: Key Features (2)



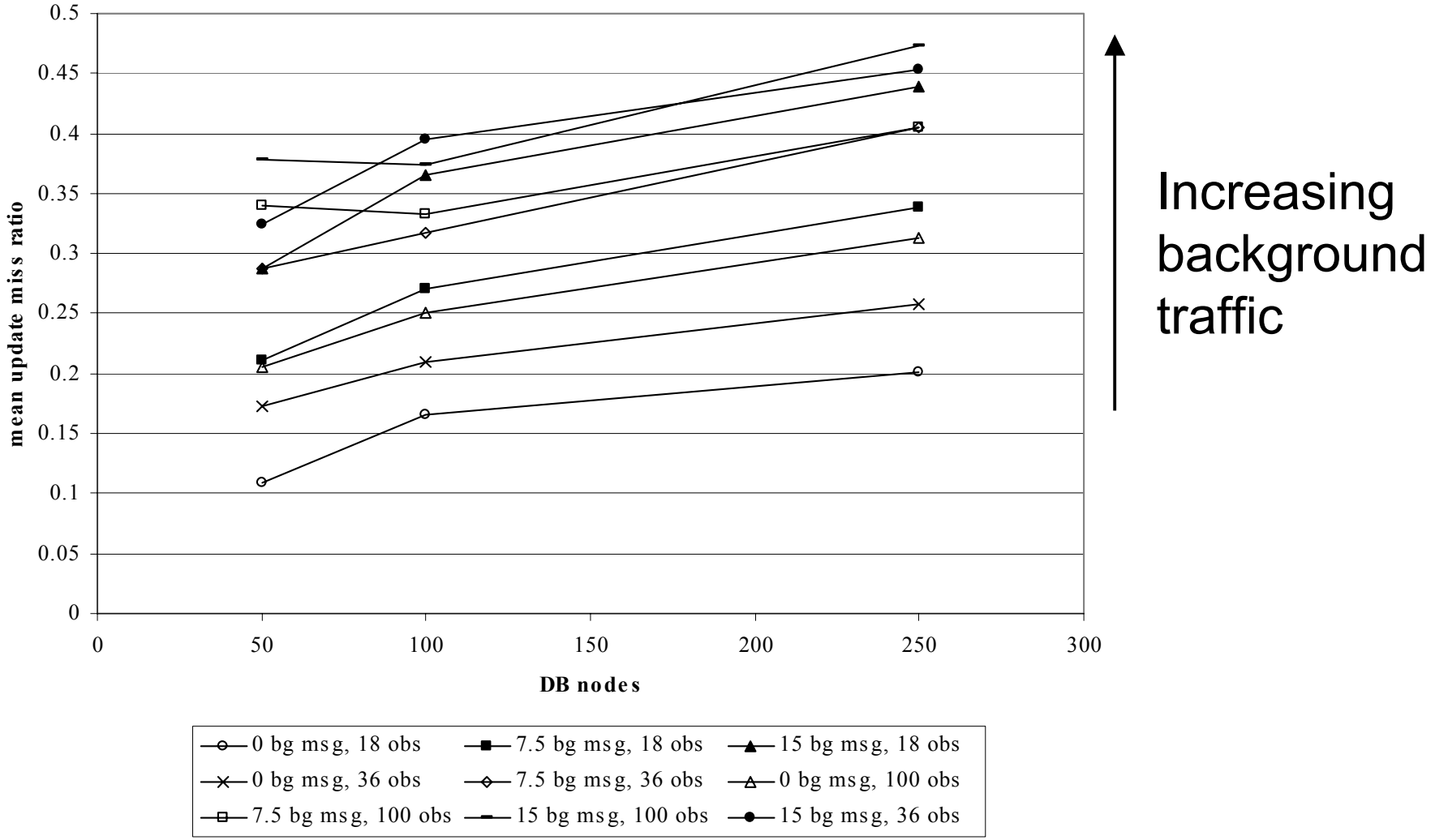
Algorithm: Performance in a Nutshell

- Varied system parameters
  - Number of DB nodes
  - Number of observers
  - Background traffic (messages per DB node)
- Measured quantities
  - Update latency
  - Update miss ratio for given object
  - Gap length → number of missed update requests between two accepted update requests
  - Number of messages sent per DB node and update request
  - Message length

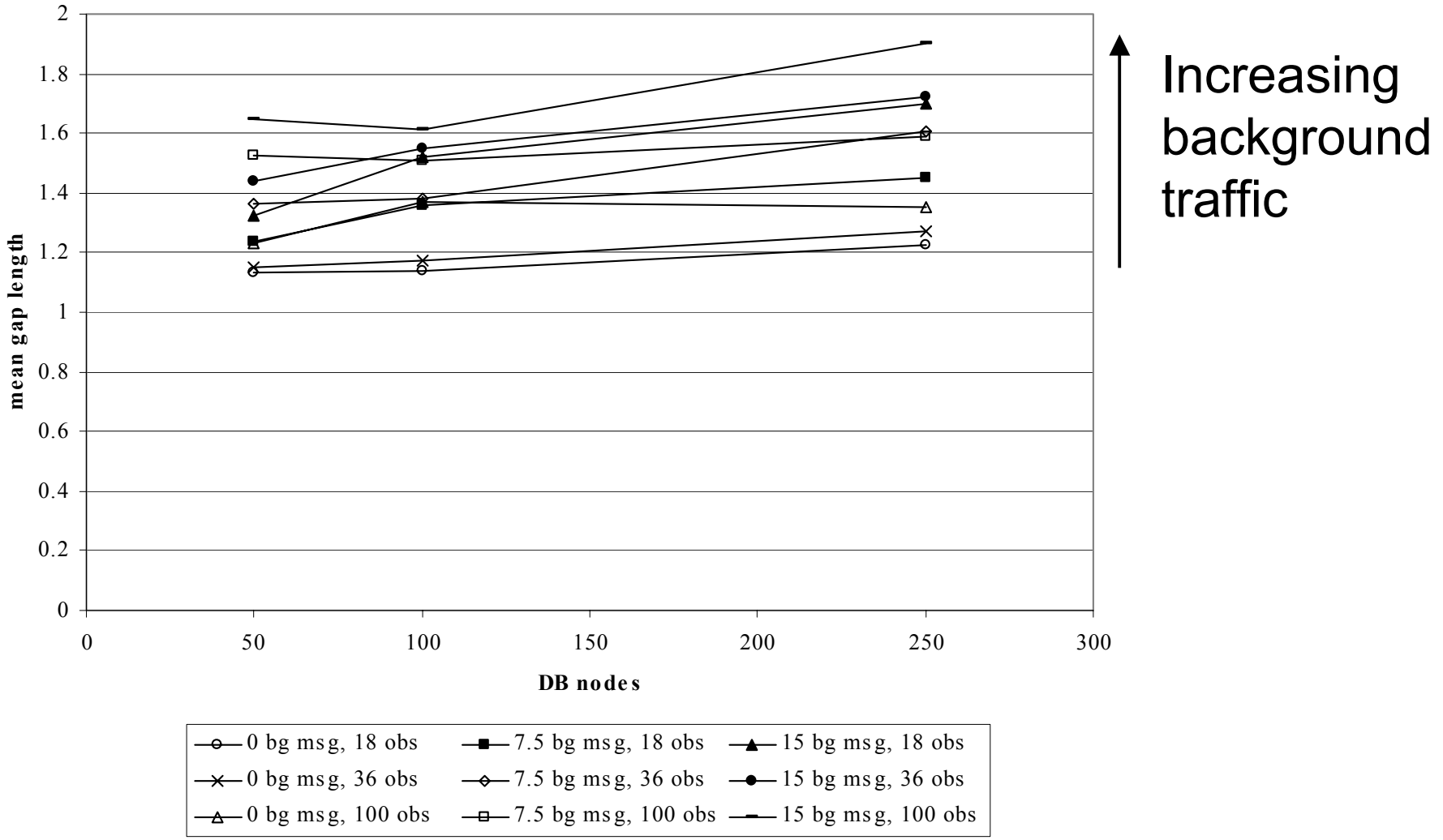
Algorithm: Performance in a Nutshell (2)



Algorithm: Performance in a Nutshell (3)



Algorithm: Performance in a Nutshell (4)



Summary

- Consistency model
  - Explicitly considers global temporal ordering of update operations
  - Provides explicit guarantees for application developers
  - Suitability & feasibility for applications in MANETs
- Proposed algorithm
  - Does not require synchronized clocks
  - Performance evaluation