Exploiting Temporal and Spatial Constraints on Distributed Shared Objects

Richard West, Karsten Schwan, Ivan Tacic & Mustaque Ahamad

Georgia Institute of Technology

Introduction

- Distributed applications with shared state.
- Existing consistency protocols developed primarily for scientific applications.
- Better scalability & concurrency by exploiting application-level semantics.
- Problem: How to formulate & use application semantics to efficiently share state.

Approach

- Aim to support applications exhibiting:
  - Poor and unpredictable locality.
  - Symmetric data access.
  - Dynamic changes in sharing behavior.
  - Data races.
- Examples:
  - Multimedia video games.
  - Virtual environments.
  - Distributed interactive simulations.

Contributions

- Run-time support to efficiently maintain shared objects based on application semantics.
- Development of S-DSO:
  - Semantic Distributed Shared Object System.
  - Support for applications with spatial / ordered constraints on shared objects.

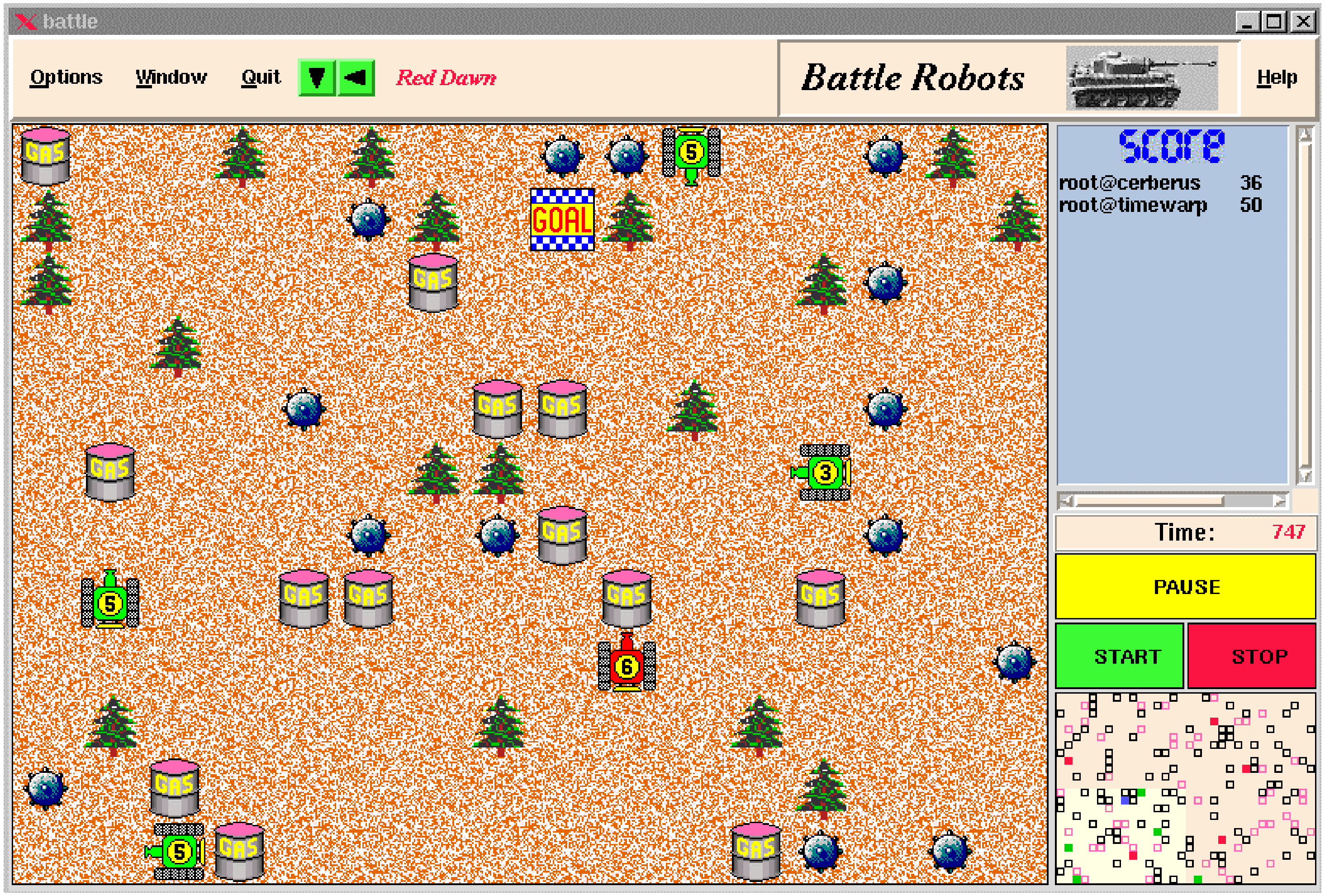
Overview

- Sample application.
- Semantics:
  - Definitions.
  - Temporal and spatial consistency.
- S-DSO overview.
- Experimental evaluation.
- Results.
- Conclusions.
- Future work.

Sample Application

- Multi-player combat game with shared environment.
- Derived from distributed interactive simulations.
- Maneuver team of tanks to known goal in presence of enemies.
- Exploit user-specified attributes to improve performance.

Video Application



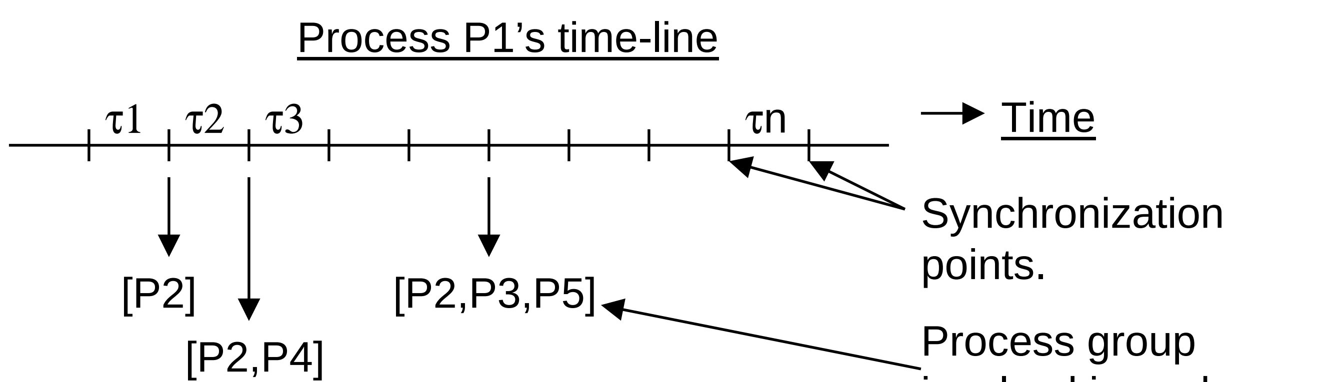
R.West, Georgia Tech (1997)

Semantics

- Application-level **spatial** & **temporal** semantics.
  - e.g. Exchange state info only when two tanks less than distance *d* apart.
- Lookahead consistency:
  - Ability to predict future times when process groups must exchange object modifications.
  - Processes synchronize if/when object's current state is required.

Temporal and Spatial Consistency

- **Temporal** $\Rightarrow$ *<u>when</u>* changes to shared objects become visible.
- **Spatial** $\Rightarrow$ *<u>which</u>* processes should be updated with changes based on locations in shared space.
- For any time interval τ_n processes P_i and P_j only consistent for those objects needed in interval τ_{n+1}

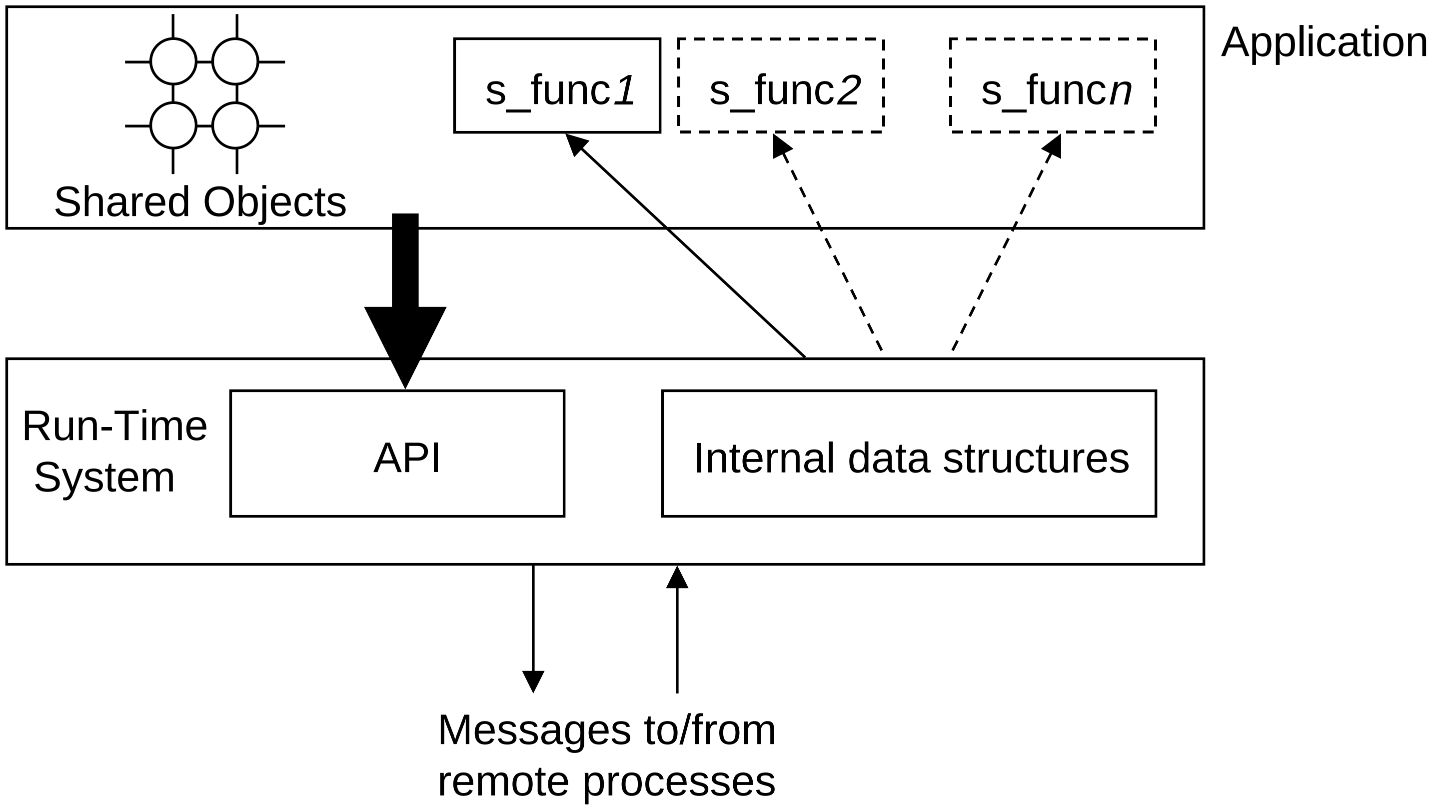


R.West, Georgia Tech (1997)

S-DSO: Semantic Distributed Shared Object System

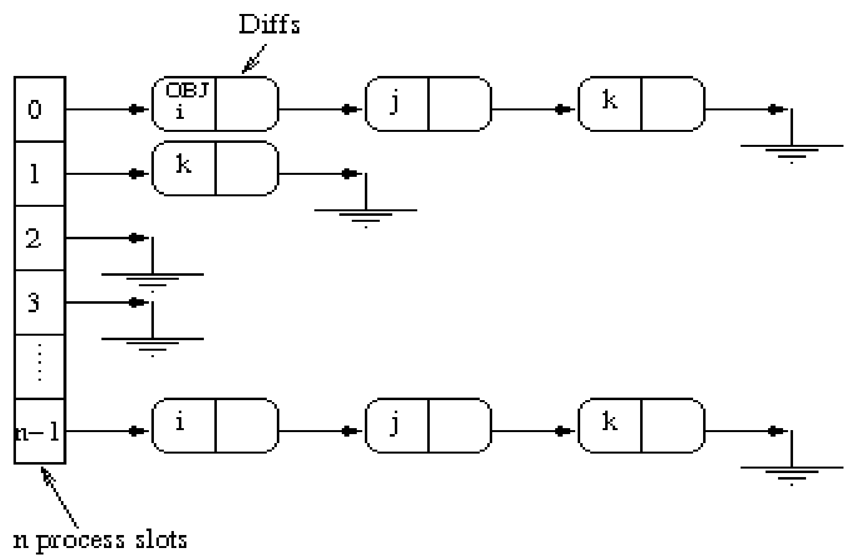
- **s_functions**:
  - Written by application programmer.
  - Used to dynamically determine:
    - *which* processes to send updates to *when*.
    - future synchronization times among process pairs.
- **exchange()** function:
  - Internal to S-DSO.
  - Controls synchronous exchange of info.
  - Uses s_function to calculate when and which processes to send updates.

S-DSO Overview



S-DSO Data Structures

- Time-ordered list of (exchange-time, process) pairs.
- Slotted buffer holding future exchanges with remote processes:



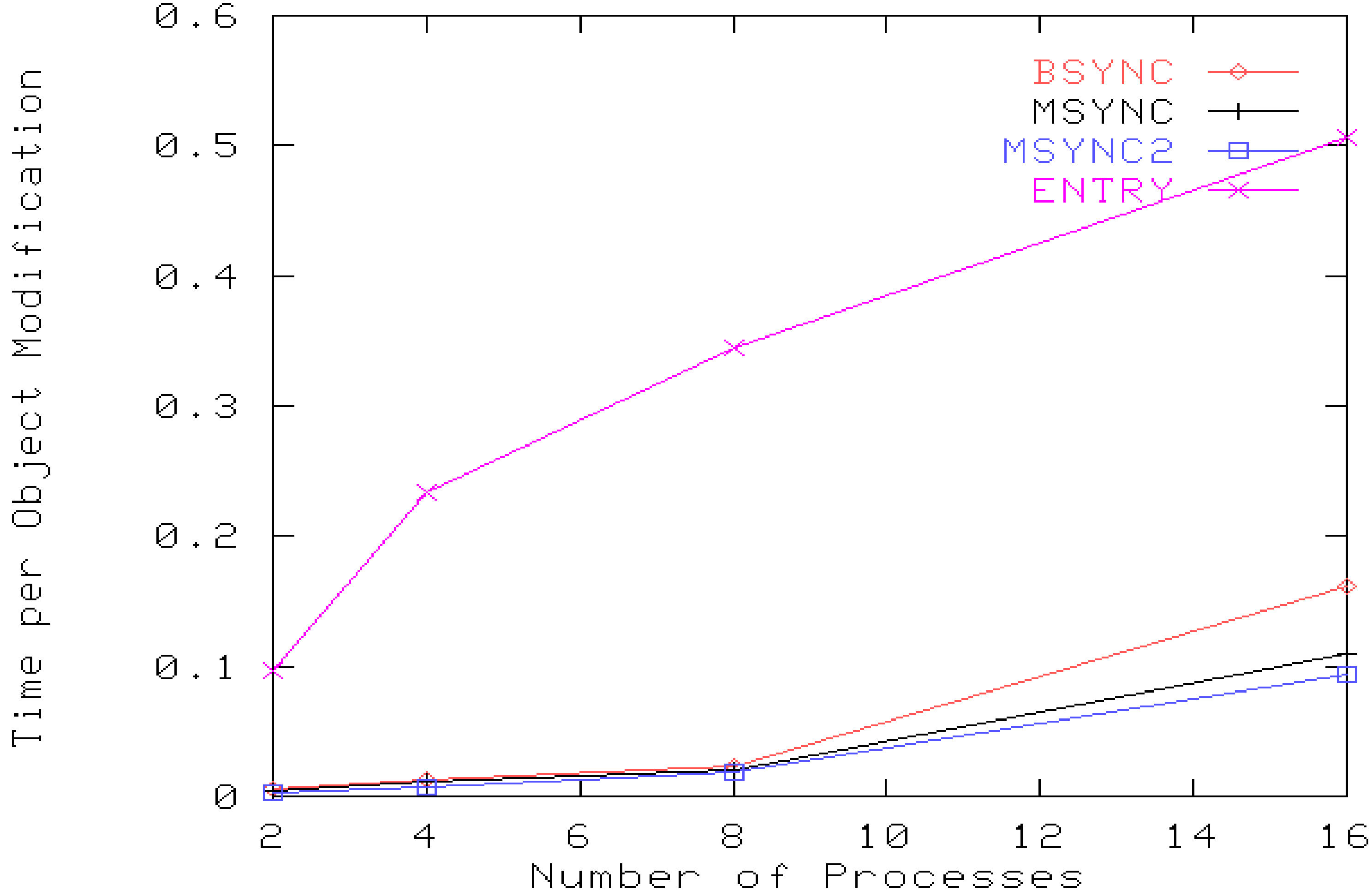
Semantic-Based Consistency Protocols

- Applied to our video game application.
- BSYNC:
  - broadcast updates after every update and await replies.
  - concurrent (phased) exchanges every τ time units.
- MSYNC:
  - Uses lookahead (s_function)
  - Synchronous exchanges based on position of process' tanks.
  - MSYNC2 reduces unnecessary exchanges.

S-DSO Experimental Evaluation

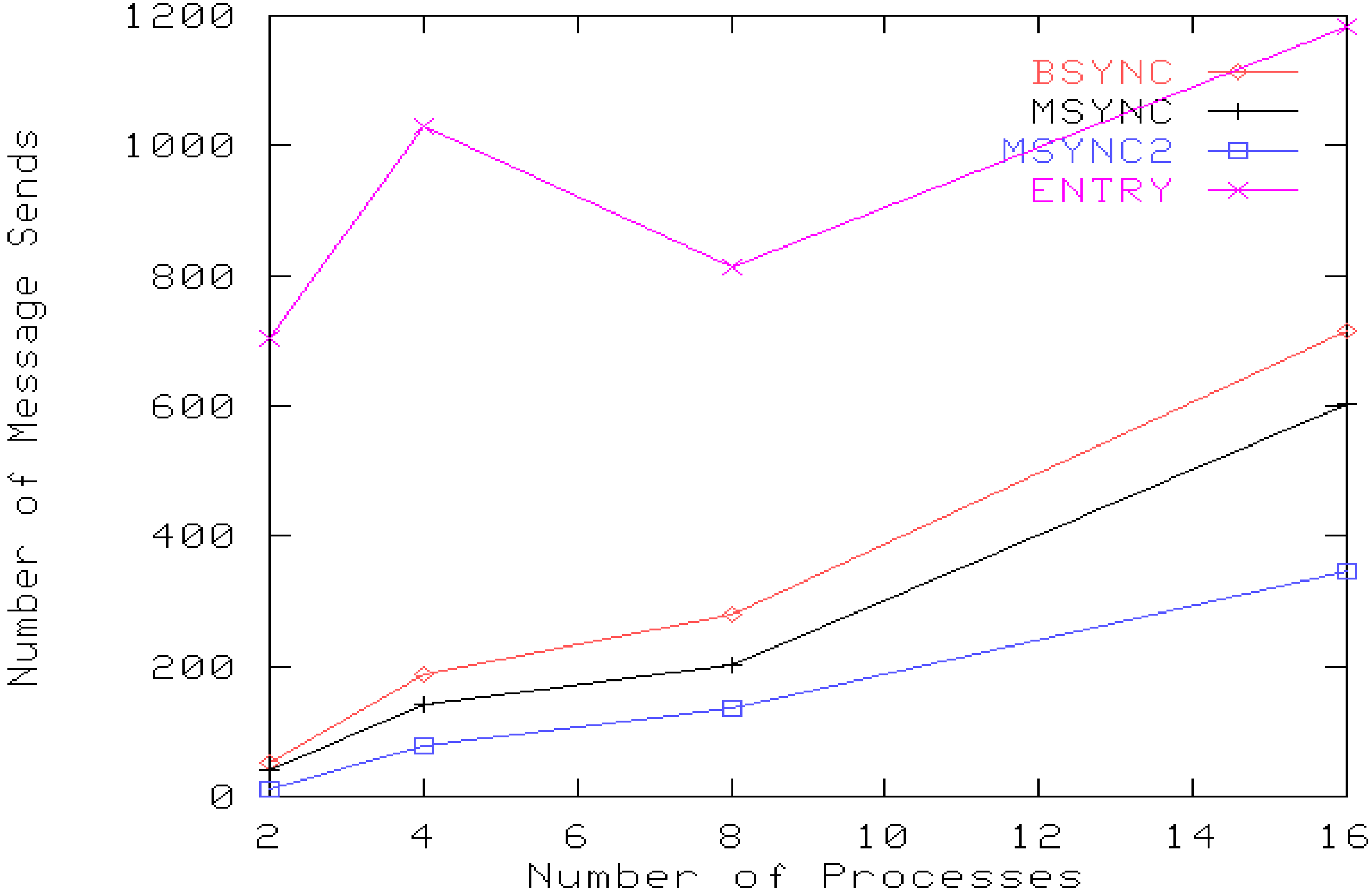
- 16 SGI workstations, 10 Mbps ethernet, TCP
- 2D shared environment (32x24 shared object blocks).
- One tank per process - one process per processor.
- Each tank tries to reach goal first.
- Objects in N,S,E,W direction and range of tank's location must be up-to-date.
- Compare BSYNC, MSYNC against Entry Consistency.

Time per Object Modification vs Number of Processes



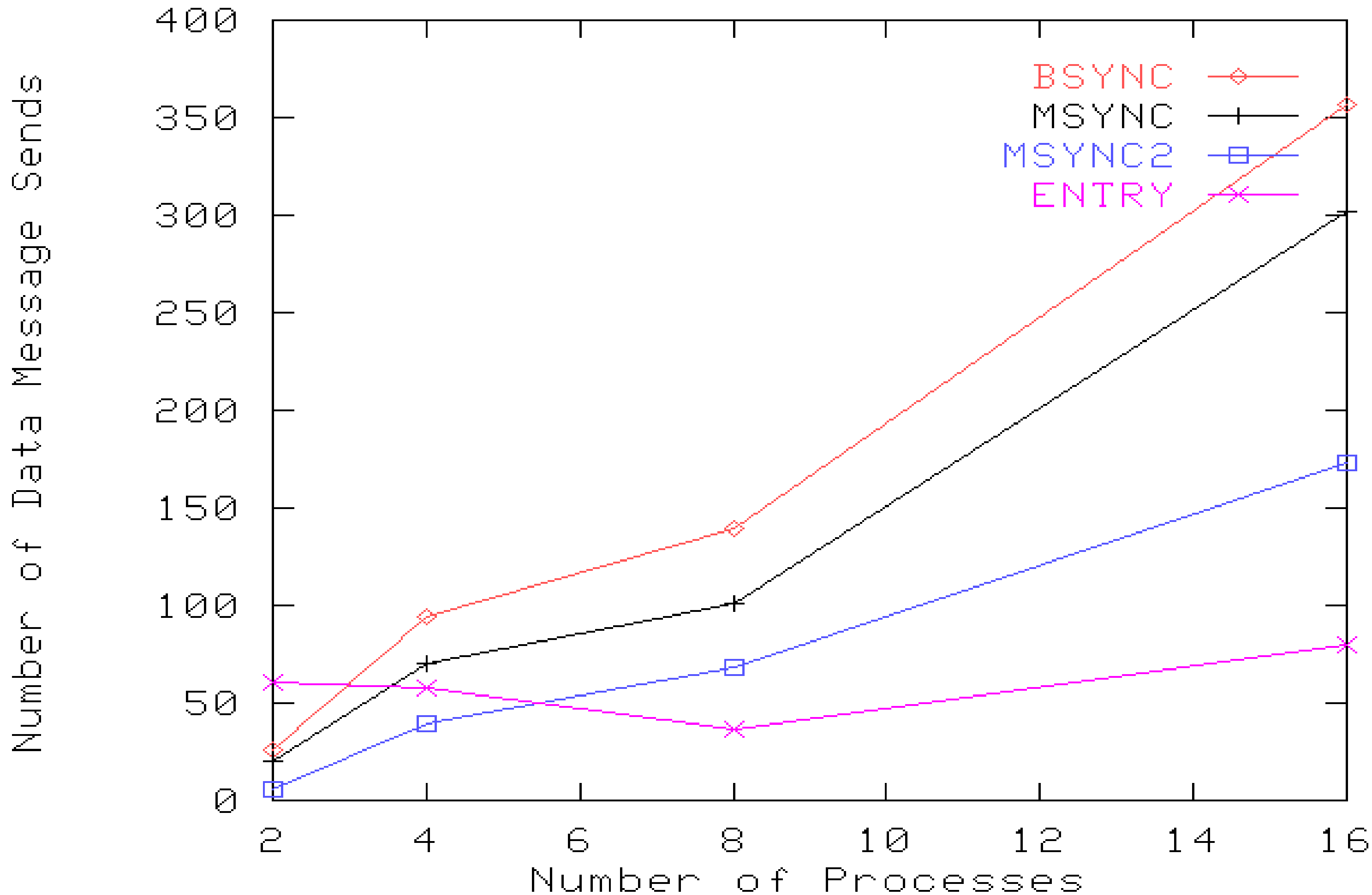
R.West, Georgia Tech (1997)

Number of Message Transfers as a Function of Number of Processes



R.West, Georgia Tech (1997)

Number of Data Message Transfers vs Number of Processes



R.West, Georgia Tech (1997)

Experimental Observations 1

- **Entry Consistency** exchanges fewer data messages.
  - Sends control messages to lock managers evenly distributed across nodes.
  - Suffers from blocking delays due to lock-acquisition.
- **Lookahead** protocols couple synchronization with data exchanges.
  - Can send unnecessary updates to processes.

Experimental Observations 2

- Lookahead consistency good for large numbers of fine-grained dynamically shared objects.
- Efficient s_functions ensure synchronization with fewer processes at any time.
- Problem with s_functions is how to avoid unnecessary exchanges.

Conclusions

- Implemented S(emantic)-DSO.
  - Supports application-specific consistency protocols.
- Lookahead consistency can effectively meet needs of applications with:
  - Dynamic sharing behavior.
  - Data races.
  - Symmetric object accesses.
- Assume ordered access or spatial relationships on objects.

Future Work

- Investigate use of graphs to represent relationships between objects.
- n-dimensional object spaces:
  - Explicit relationship between object name and location in space.
- Irregular object spaces:
  - Use graphs to capture:
    - Spatial relationships between objects.
    - Access-order to objects.
- Consistency of meta-level graph information.