Lamport's Scalar Clocks
vs
Singhal & Kshemkalyani's Vector Clocks

Presenter: Marling Engle

Background – Scalar Clocks

- Lamport's Scalar Clocks
  - Outgoing messages tagged with process's clock value
  - On receive of message:
    - myclock = MAX(incoming, myclock)

Background – Vector Clocks

- SK Vector Clocks
  - Outgoing messages tagged with a vector of {id,clock}
    - Represents the last known clock value for id (that the originating process is aware of)
  - On receive message(incoming[])
    - Update my known values for each clock
    - current = MAX(incoming[],current)

Motivation

- Functionality of Vector Clocks supersedes that of Scalar Clocks
  - consistency: if a>b, then C(a) > C(b)
  - strong consistency: consistency and if C(a)>C(b) then a>b
- Vector Clocks can be reduced to Scalar Clocks
  - Simply send the vector containing nothing but your own clock value
- There must be a prohibitive cost for using vector over scalar clocks
  - What is it? How much does it cost?

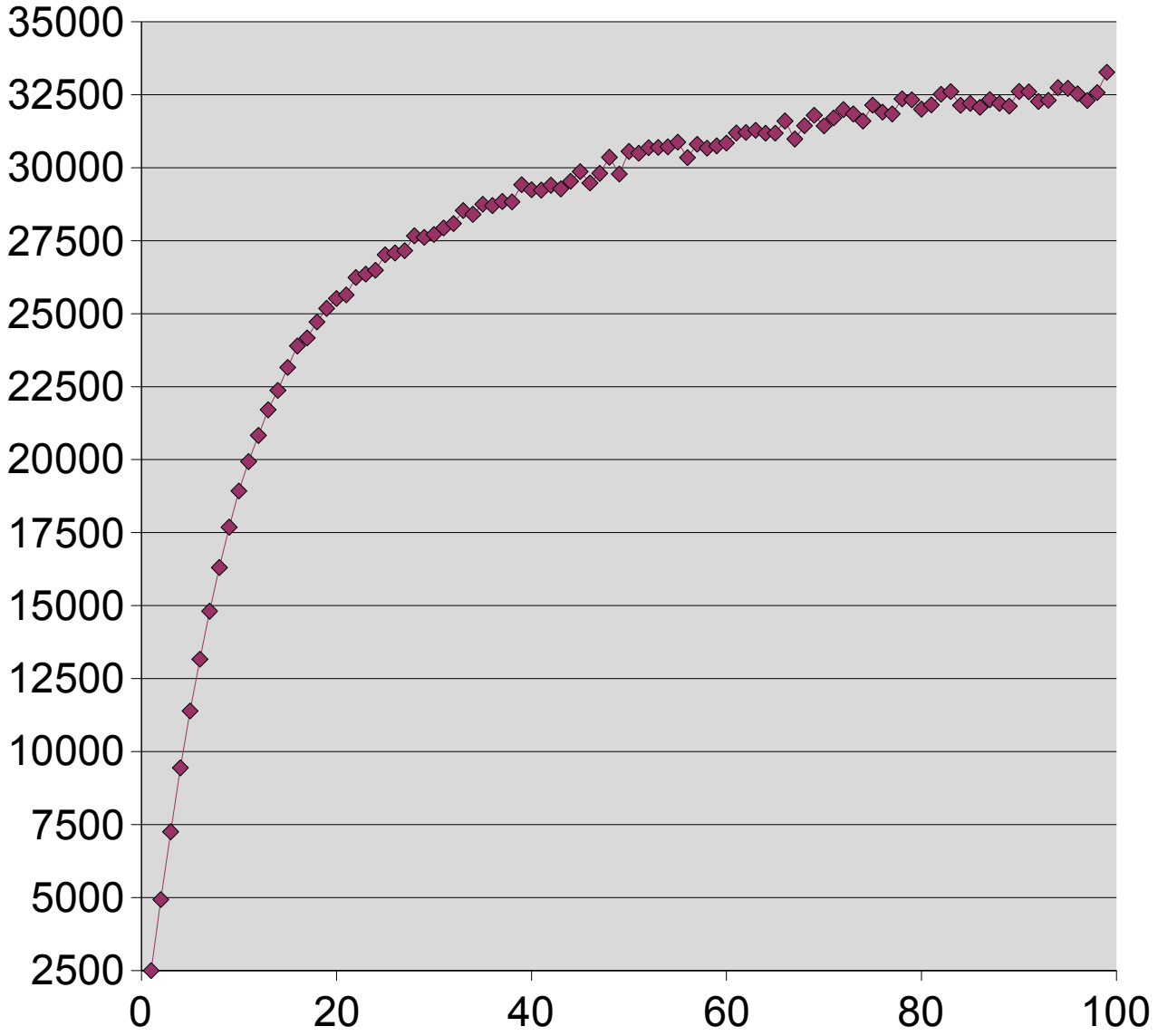
Experiment – Base Algorithm

- There are N nodes in the system:
  - Initiator sends T tokens, each with a random destination
  - Upon receipt of the token, the process forwards it to a random destination
- Base algorithm continually generates message traffic – The overall message complexity is determined by T (more tokens being forwarded = higher complexity)

Experiment - Setup

- Measure the effect of message complexity on average message length
  - Fixed number of nodes – 11
- Measure the effect of number of nodes on average message length
  - Fixed number of tokens – 3
- Experiment terminated when the first process reaches a clock of 5000

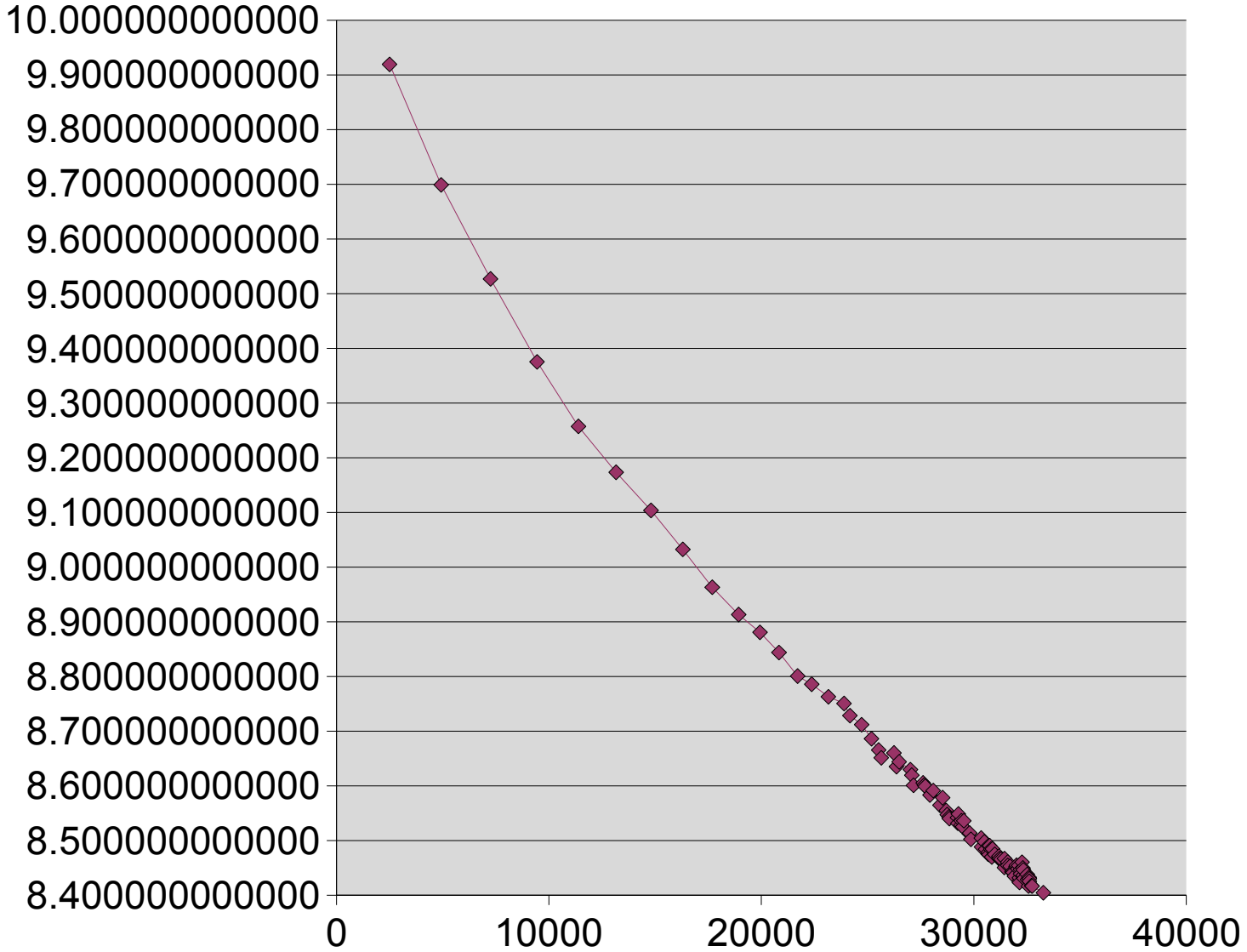
Message Complexity vs Tokens



Results: Demonstrates that T (number of tokens) is sufficient to directly influence message complexity

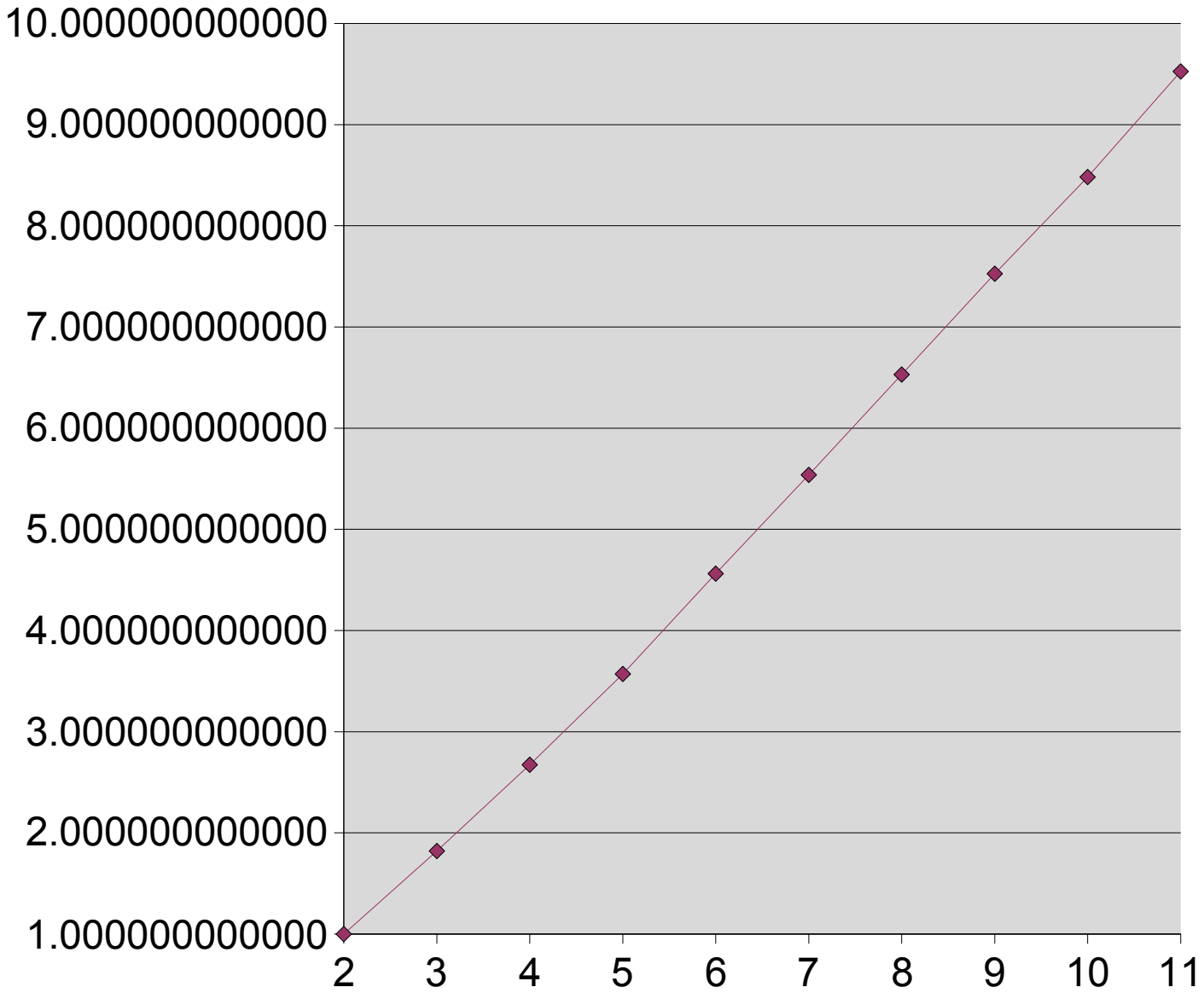
Experiment - Results

Message Length vs Message Complexity



Experiment - Results

Message Length vs Nodes



Conclusions

- Increasing message complexity decreases message length
  - The more messages circulating the network, the less likely a node A will have to send it's entire vector to node B
- Increasing the number of nodes increases message length
  - More nodes to keep track of, and larger vectors to send

Future Work

- Explore less 'random' base algorithms
  - Example: wave algorithms
    - In algorithms where the messages share some more defined dependency, it may be less necessary to send updates
    - Subgroup activity clusters
- Real time
  - Experiment limited to 5000 logical clock ticks
  - Real time limited – would allow for greater variance in messages sent
    - More messages means more logical clock ticks, which means faster (real time) exit

Problems

- Base Algorithm
  - Where to count sent tokens
  - Tracing tokens
- TinyOS limits
  - Packet size
    - Forced reimplementation of SK Vector clocks to get worthwhile results
  - Messaging Speed
  - Token Loss
- Implementation
  - Command line arguments