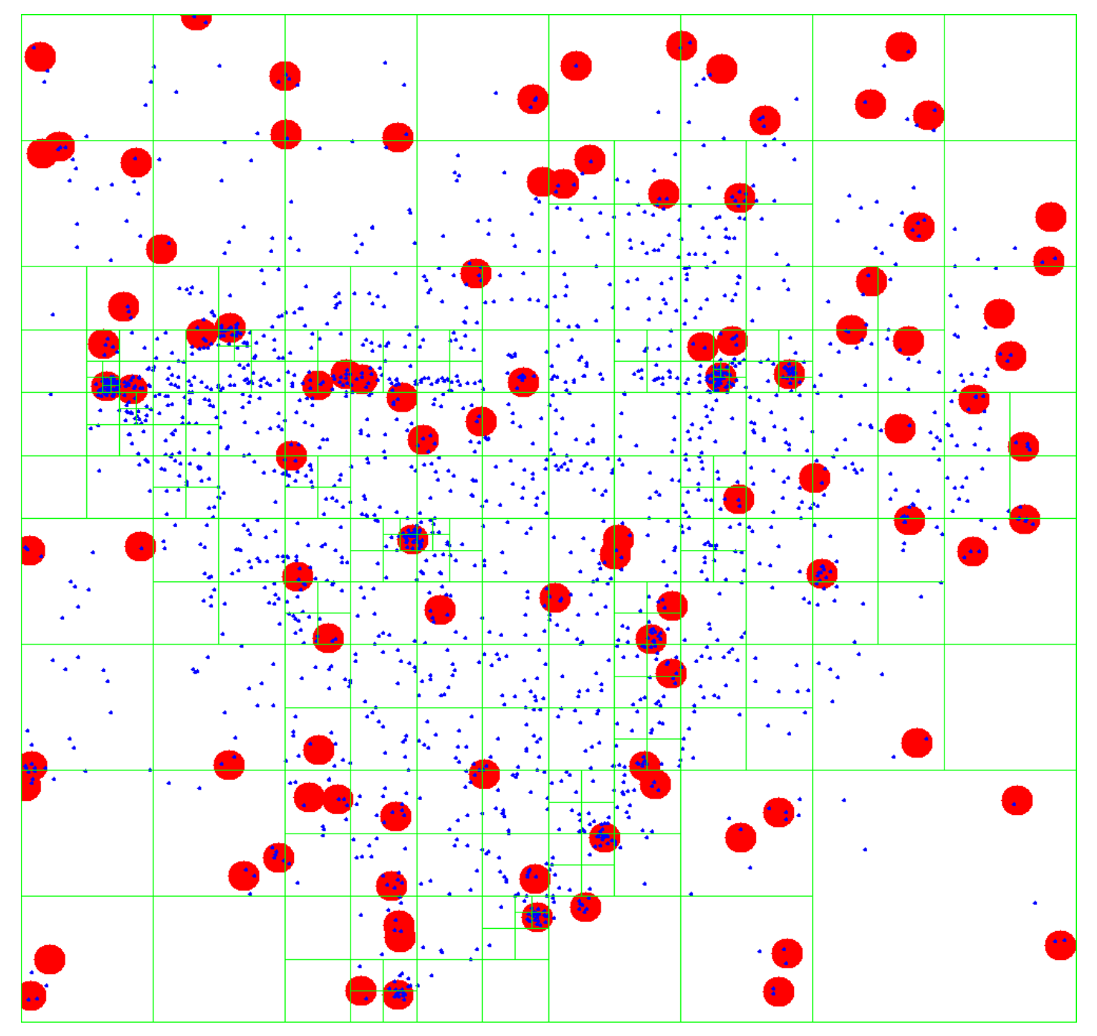
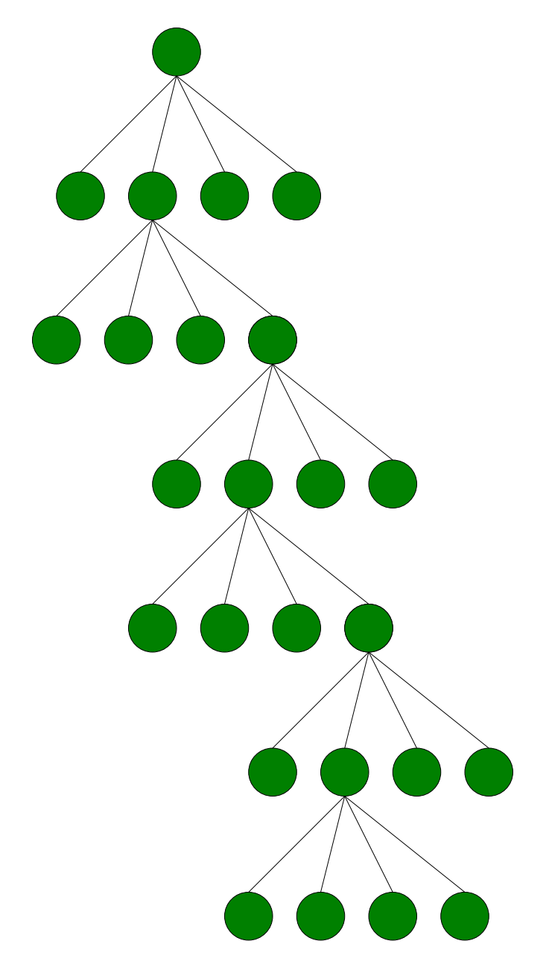
Scalable Peer-to-Peer Event Ordering

UNIVERSITY OF OREGON





Chris GauthierDickey and Virginia Lo

Is Scalable Event Ordering in Peer-to-Peer Systems Possible?

- Event ordering is a fundamental operation required in many distributed systems, for example:
  - Interactive, multi-player games
  - Distributed simulations
  - Online stock-trading
  - Gibson'esque virtual reality (The Matrix)
- The sheer scope of these distributed systems makes traditional event-ordering algorithms very hard
  - Paxos algorithm requires up to 5 rounds of communication
  - Naïve implementation would take $O(n^2)$ messages

Existing Research only provides Partial Solutions

- Peer-to-Peer structures, such as DHTs and other unstructured overlays, provide a mapping from the key space to the application space for efficient searching
  - Scalable in the number of nodes and searches are fast
  - However, this mapping does not directly help in event ordering
- Application-Layer Multicast (ALM) provides fast and efficient one-to-many communication
  - Totally-ordered, reliable multicast solves the event ordering problem, in fact, but it's not scalable

Our Solution

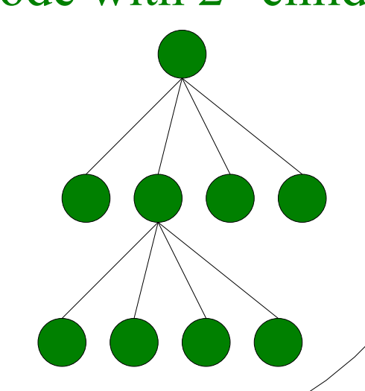
- Map the application state space to an N-dimensional Tree (called an N-Tree).
- Organize peers based on their *location*, or area of interest, in the state space.

What is an N-Tree?

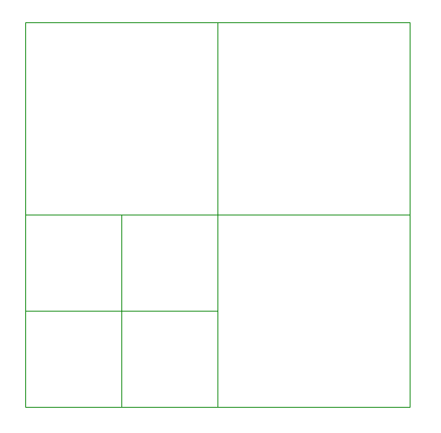
Basis: a Leaf

Inductive case: a Node with 2^N children, each an N-Tree

Example (a 2-tree):



The N-Tree maps to an N-dimensional space, where each dimension corresponds to an ordered set of values in the application. Peers locate themselves in the tree according to which values they are interested in. The 2-Tree above corresponds to this 2-dimensional cartesian grid.



Event Ordering with an N-Tree

- Tree operations:
  - Systems reside at the leaves of the N-Tree and use traditional event ordering protocols with other members of the leaf.
  - A leaf is subdivided whenever it has too many residents and branches are collapsed when the population is lower than a given threshold.
- Event Ordering:
  - Events include a scope, which is a function that defines the subset of the application state space affected by the event.
  - Events are propagated through the tree to the correct branches.
  - The larger the scope of an event, the longer it takes to totally order in the system.
  - We can bound the event-ordering delay by the longest path between two nodes in the N-Tree that an event must reach.

Advantages of N-Trees

1) N-Trees have reasonable asymptotic messaging costs:

Operation	*Distribution of Peers*	
	Uniform	*Pathalogical*
New Member Join	$O(\lg p) + O(h)$	$O(\lg p)$
Move to New Node	$O(h)$	$O(\log_d n)$
Amortized Movement	$O(1)$	$O(\log_d n)$
Leave	$O(1)$	$O(1)$
Collapse Branch	$O(1)$	$O(1)$
Subdivide Leaf	$O(1)$	$O(1)$
Event Propagation	$O(h)$	$O(\log_d n)$

p=number of peers, h=height of tree, n=number of nodes, $d=2^N$, or dimension of tree.

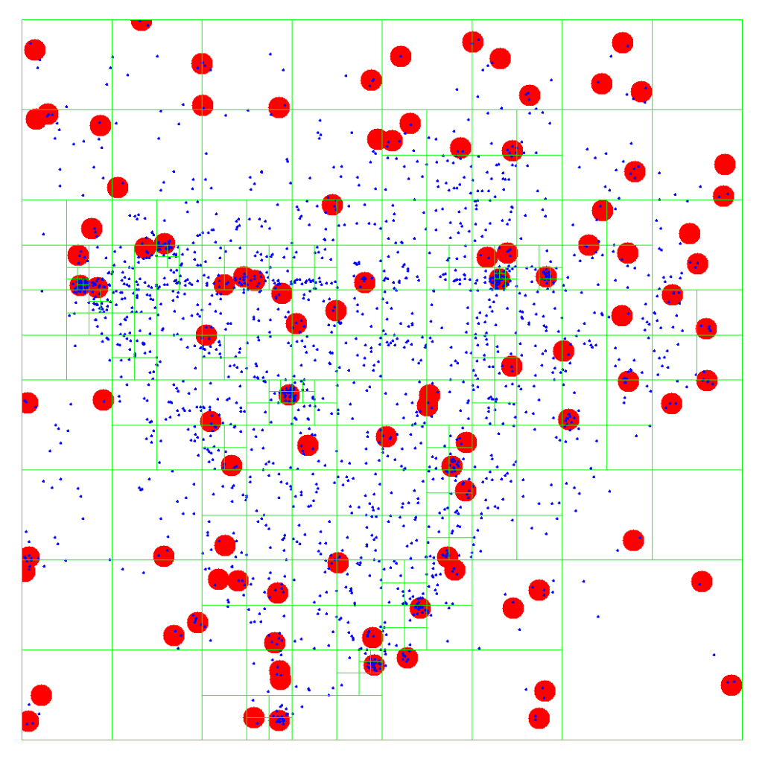
2) Peers are organized in the tree so that they exchange event messages with only those close by, avoiding needless messaging by event ordering protocols.

3) Events exceeding the scope of a leaf are quickly propagated through the tree, reducing event ordering delays.

How Well Do N-Trees Really Work?

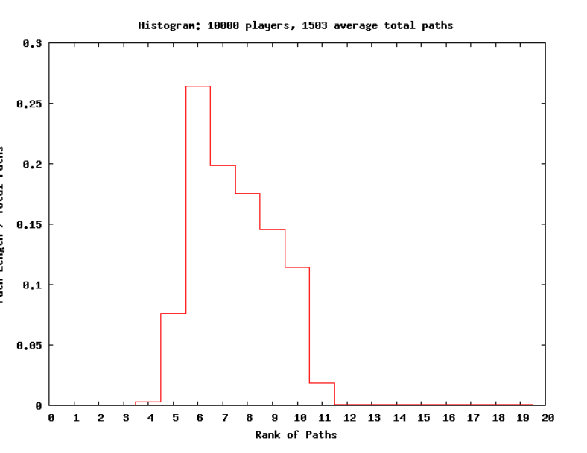
- The performance of N-Trees is based on the distribution of the areas of interest in an application state space.
  - Uniform distribution results in the best performance because the N-Tree is balanced.

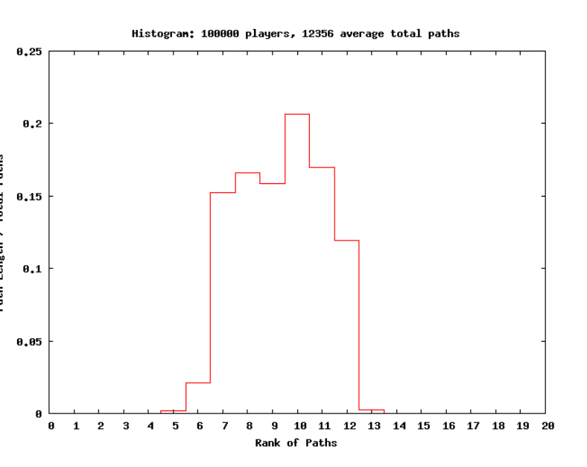
- In our experiments, we test the performance of N-Trees used for mutiplayer, interactive games. A visualization of the simulation is shown on the right.
  - **Red dots** represent *hot-spots*, which are places players are more likely to be located.
  - **Blue dots** are players.
  - **Green lines** are the divisions of the N-Tree (2 dimensional in this case).
- Players choose locations based on a Zipf (power-law) distribution and move to their chosen location.
- Over a period of time, the players wander around within the vicinity of their hot-spot and then choose a new hot-spot to travel to.
- We calculate the resulting N-Tree to determine how well it performs for event ordering.



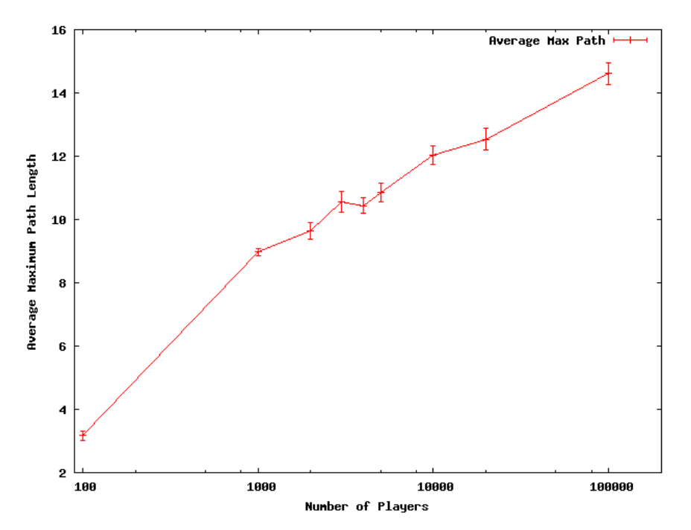
Simulation Results

- In order to determine the performance, we need to measure how long any path in the N-Tree is between two nodes.
  - This path determines how long it takes for events to propagate from one node to another, and therefore be ordered.
  - N-Trees are *not* balanced, therefore pathologic cases can be bad.





- Histograms show that the majority of path lengths are short, especially in comparison to the number of players. Even though some of the path lengths are several hops, we expect that most events will actually be local, and therefore totally ordered at the leaves without being propagated in the tree.



- Maximum average path lengths are also small, considering the number of players!

Summary and Conclusion

- Scalable Peer-to-Peer Event Ordering is possible by using hierarchy and event scoping.
- N-Trees efficiently map systems to their scope of interest in the application state space, allowing events to be propagated quickly between peers.
- N-Trees perform well theoretically, and optimally when systems are distributed in a uniform manner.
- Our simulation results verify that N-Trees work well for event ordering. In particular, they show that N-Trees also perform well when systems are distributed by power-laws.
- As future work, we plan to continue to study other metrics to measure the utility of N-Trees with event ordering.