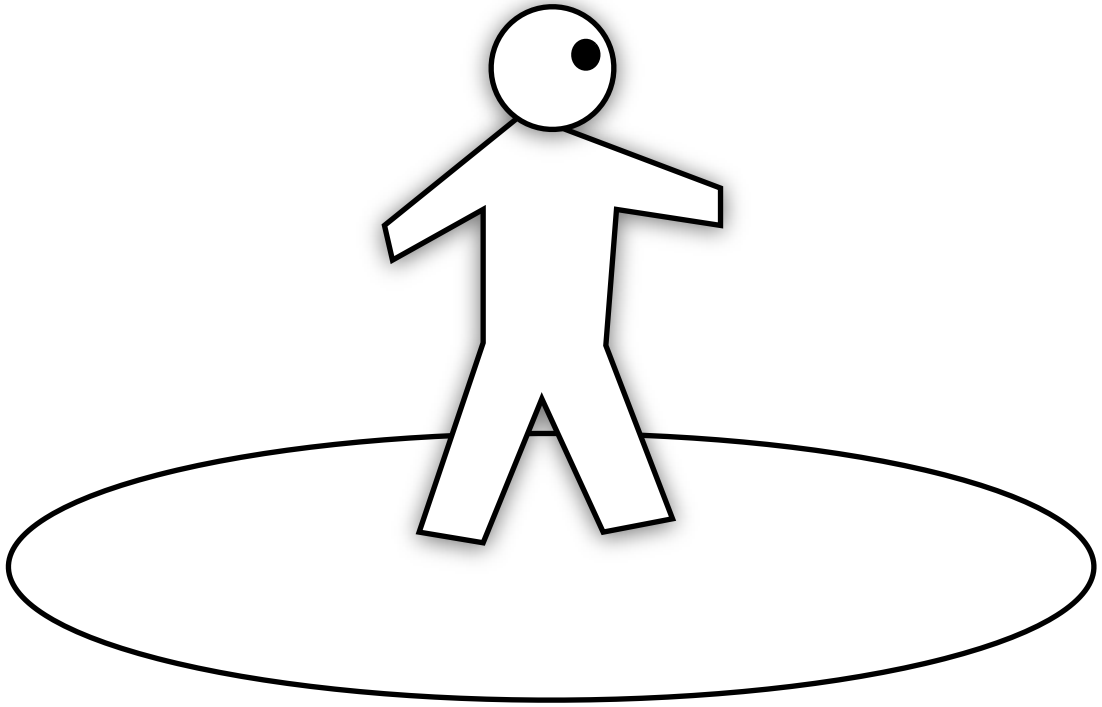
Interest Management

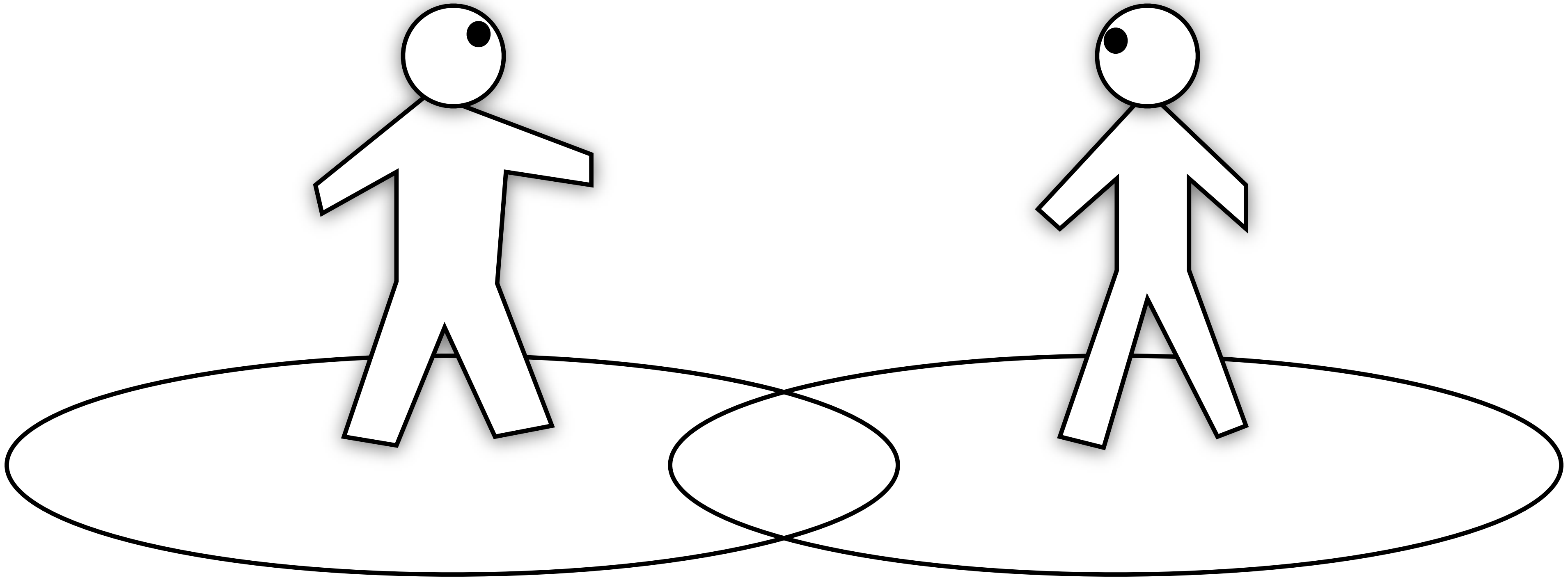
Relevance Filtering

Idea: only need to update another player p if the update matters to p.

Aura / Area-of-Interest

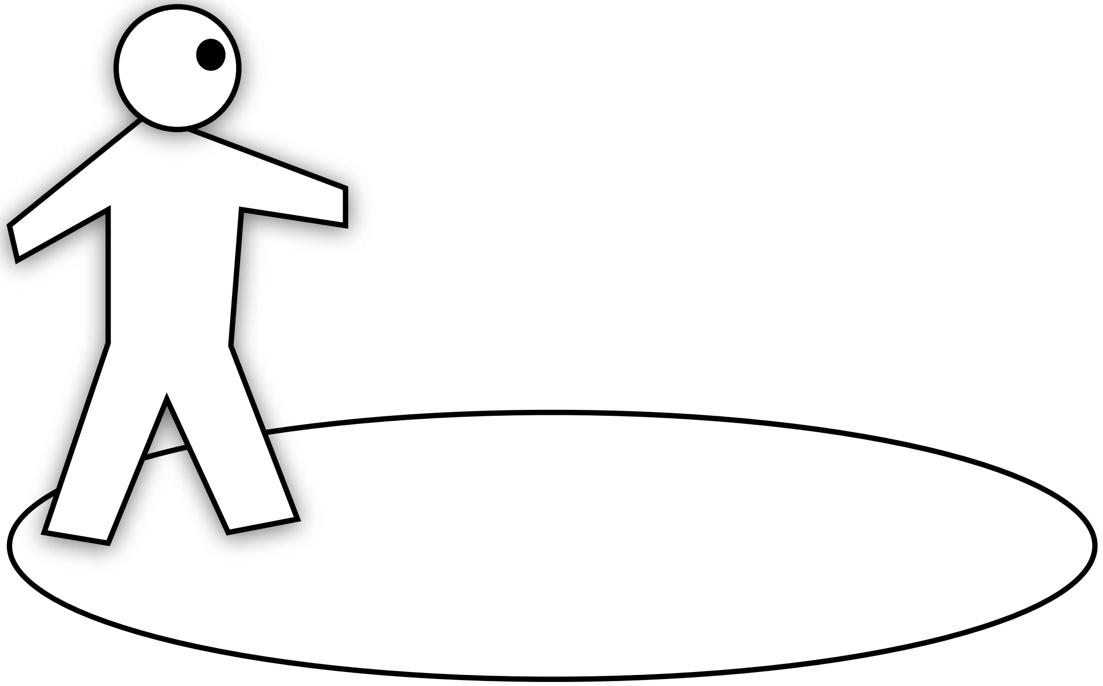


Update of *p* matters to *q* if the auras of *p* and *q* intersect.

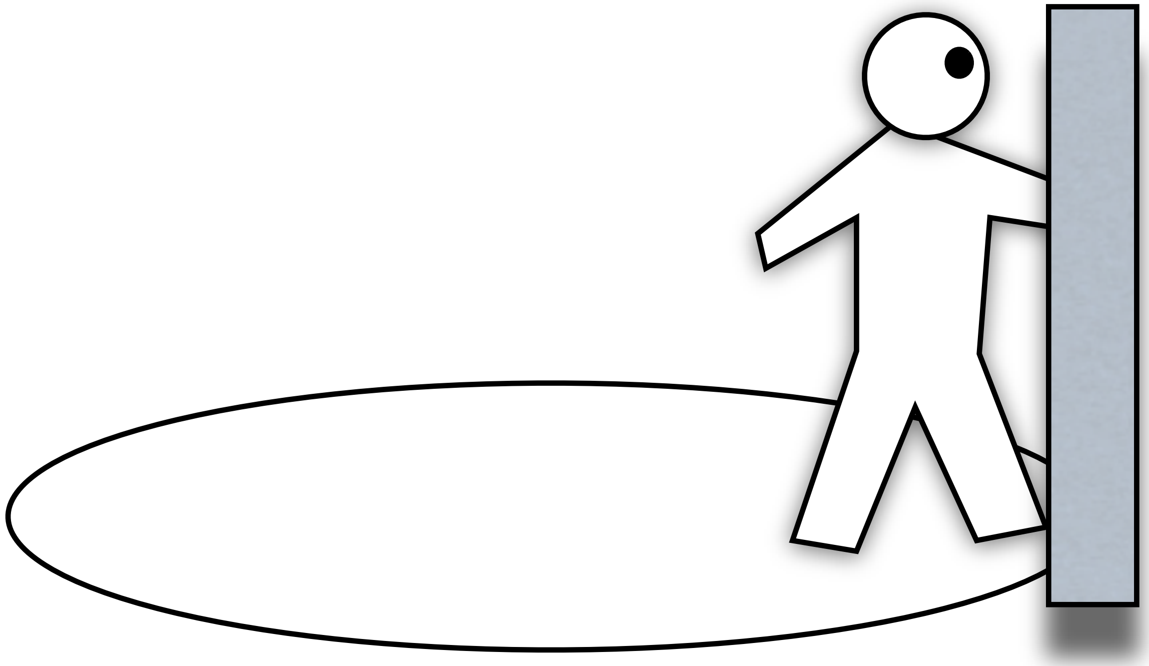


Foci

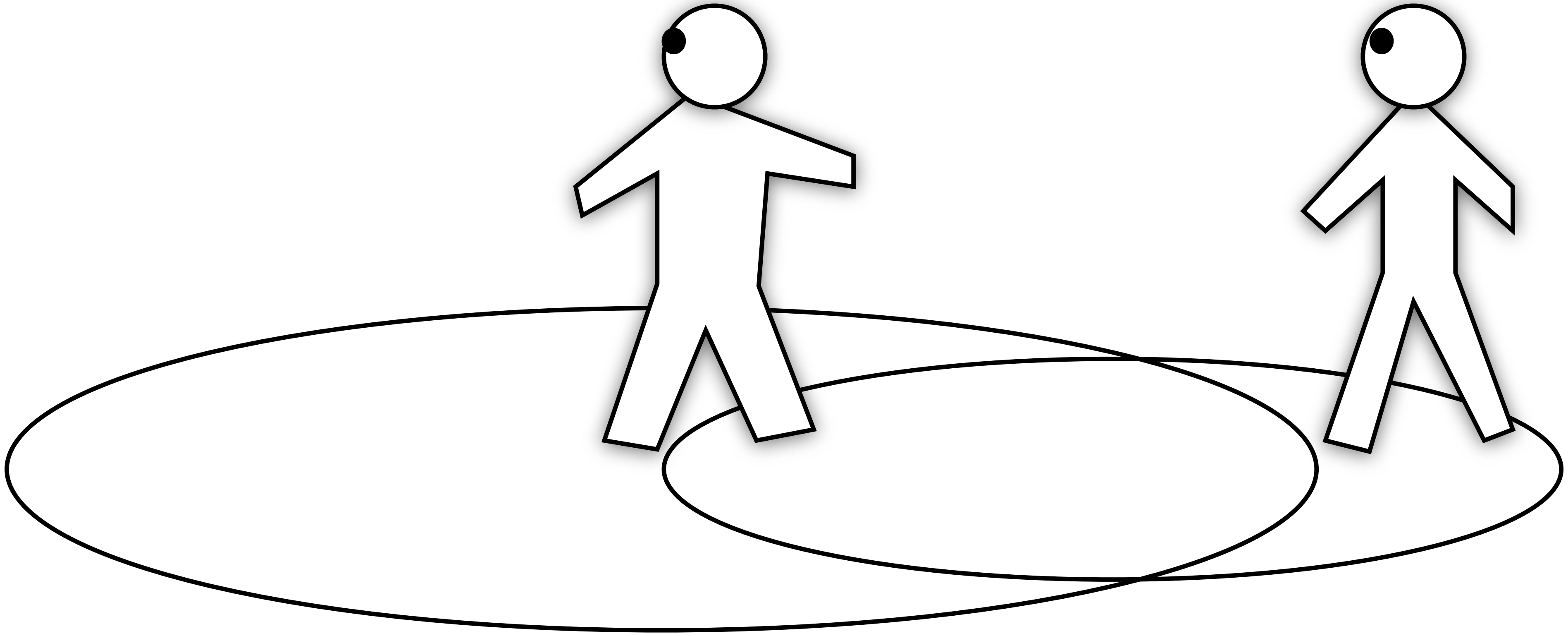
(what a player can see)



Nimbi
(where a player can be seen)

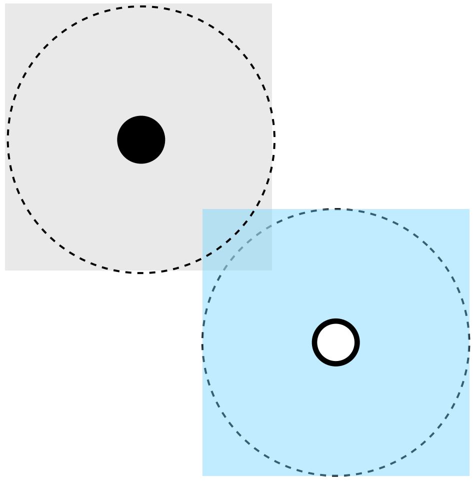


Update of *p* matters to *q* if the foci of *p* intersects nimbi of *q*.

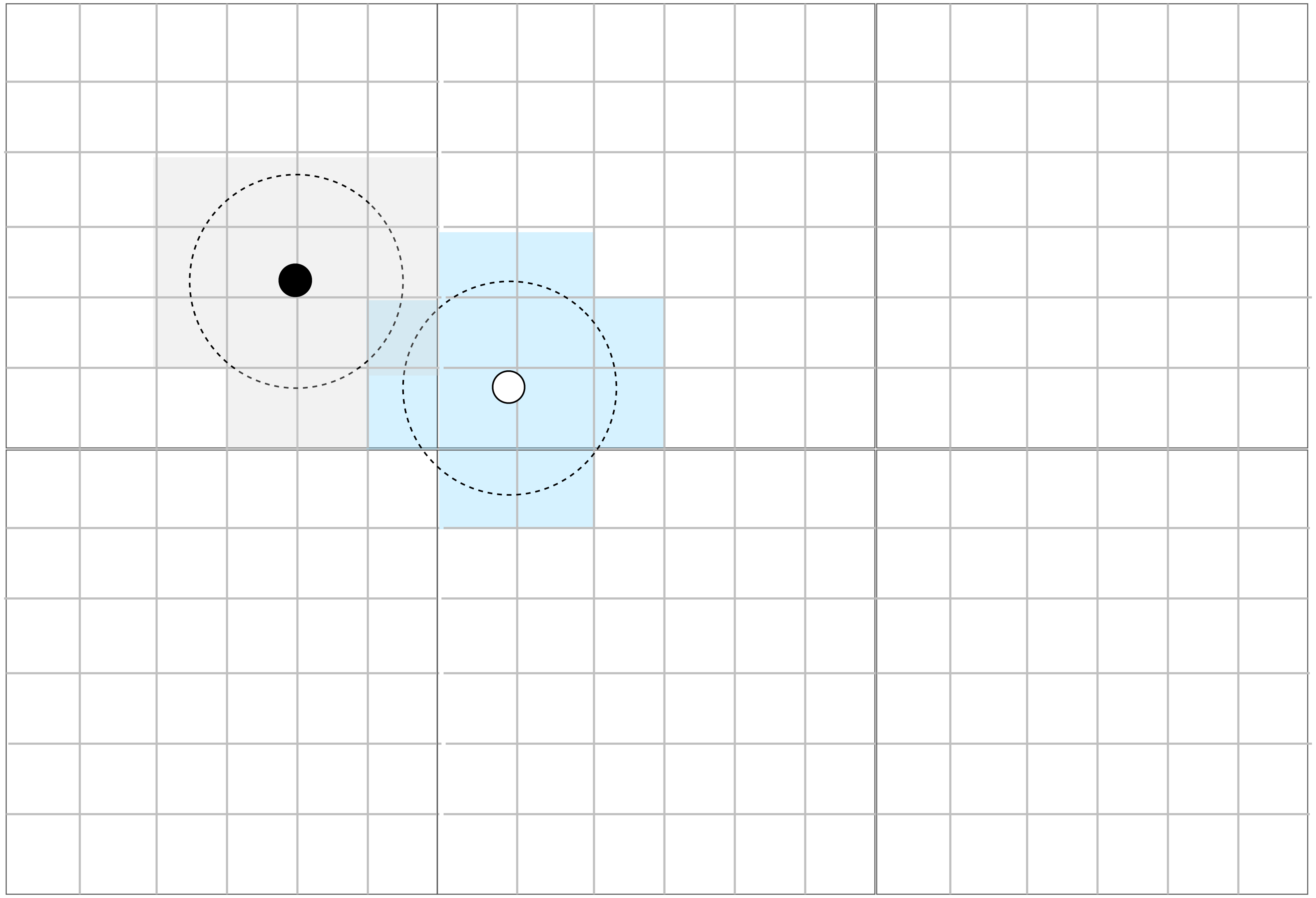


Calculating aura/foci/nimbi can be costly.

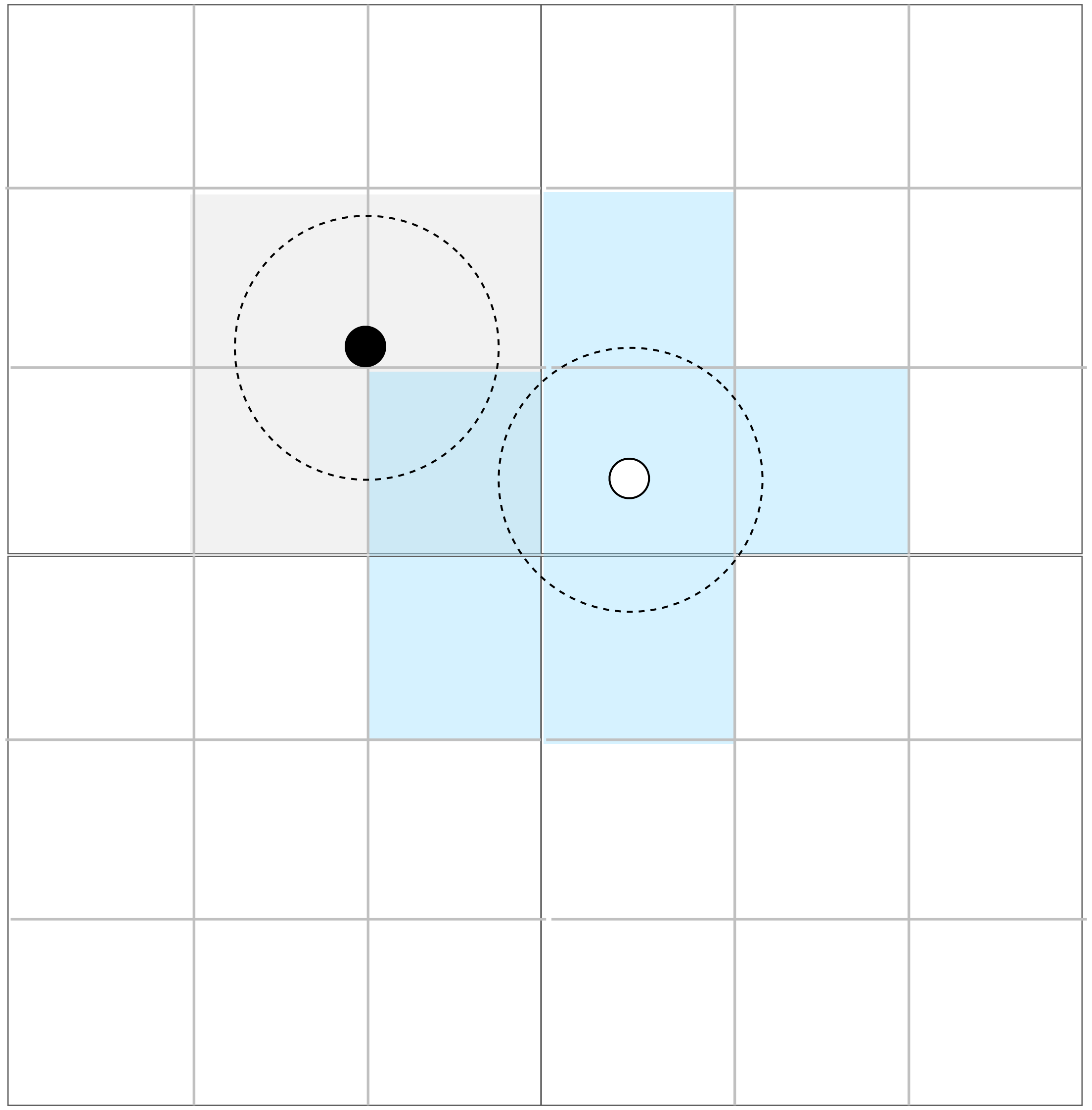
Idea: approximate use bounding boxes



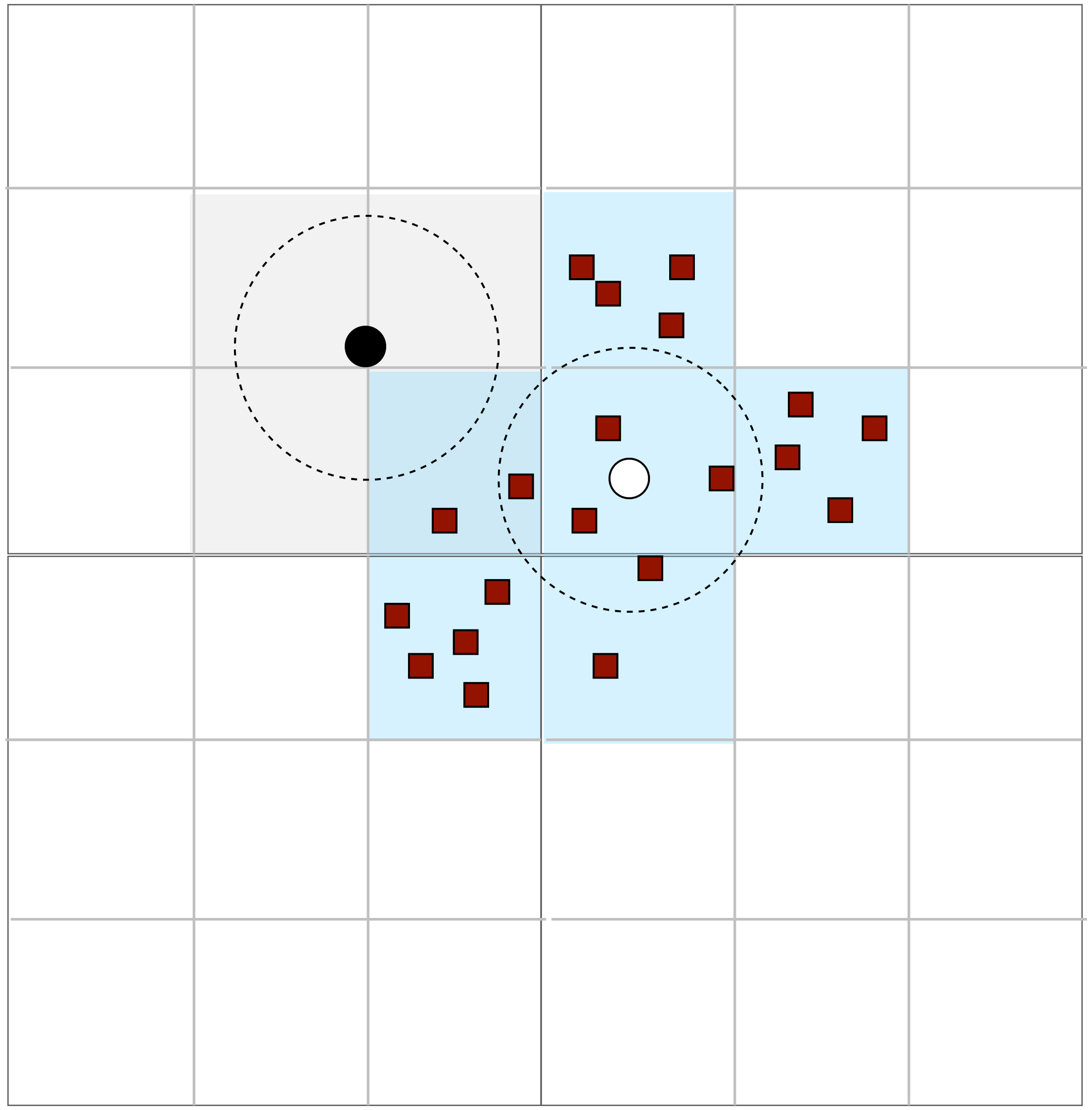
or approximate using cells



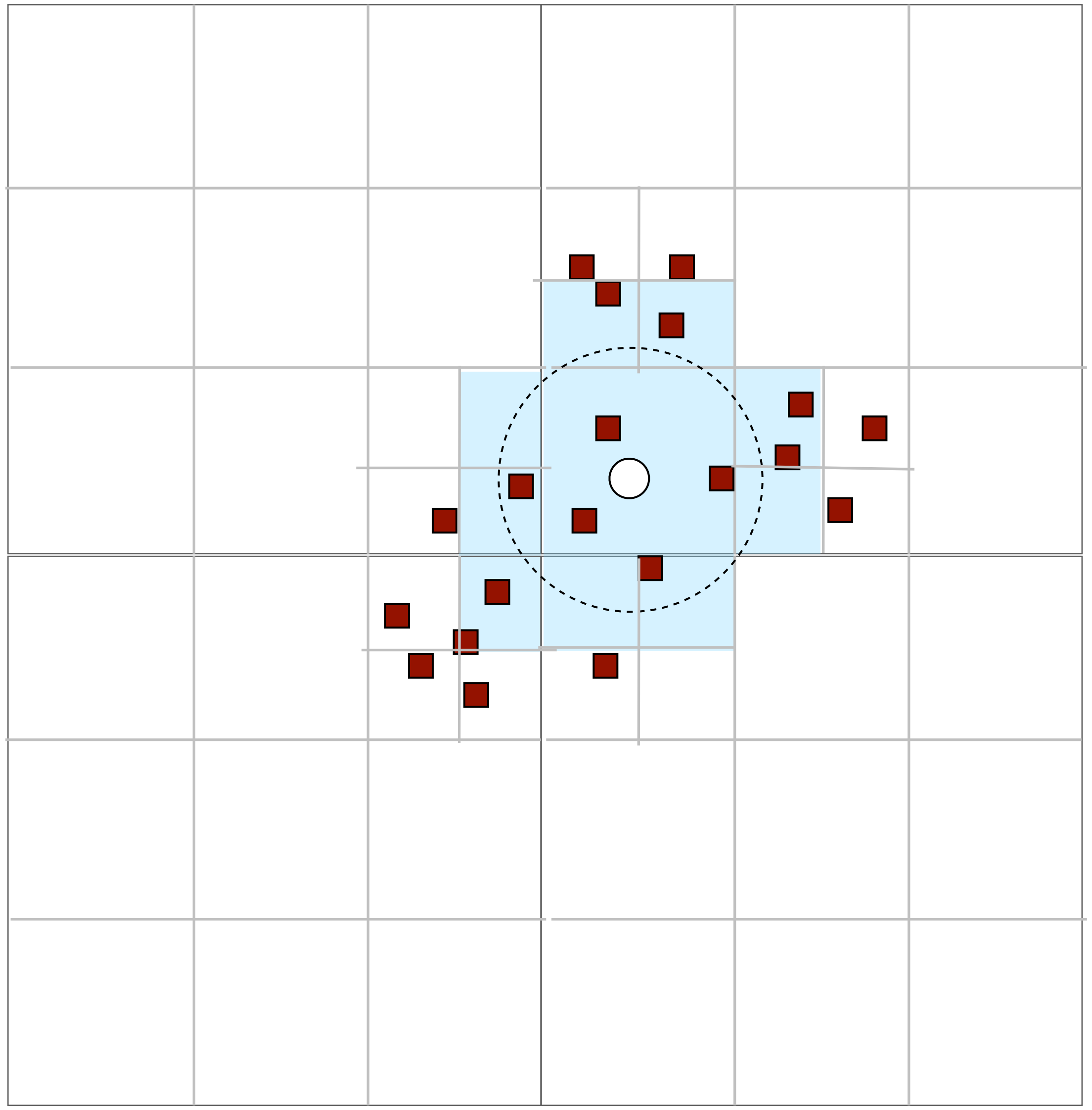
Large cell: Redundant messages.
Small cell: Large management overhead.



The white player will receive many messages he/she is not interested in.



Idea: we can dynamically partition the cells into smaller ones as needed.



Generalization: an entity may specify any other events/ entity it is interested in.

Communication Abstraction

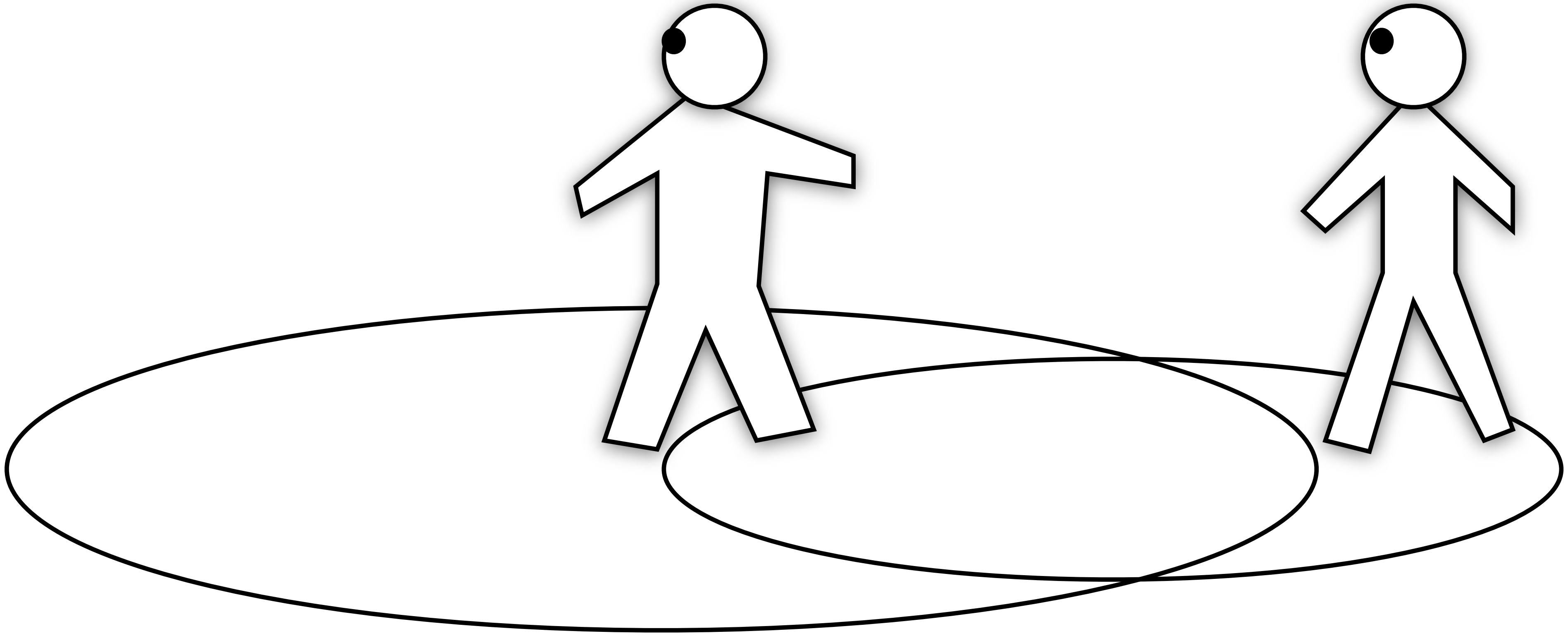
Multicast: send a message to a set of subscribers

Group: a channel to **publish** messages

A client can **subscribe** to/ **join** a group to start receiving messages from that group.

A client can **unsubscribe** from/**leave** a group to stop receiving messages from that group.

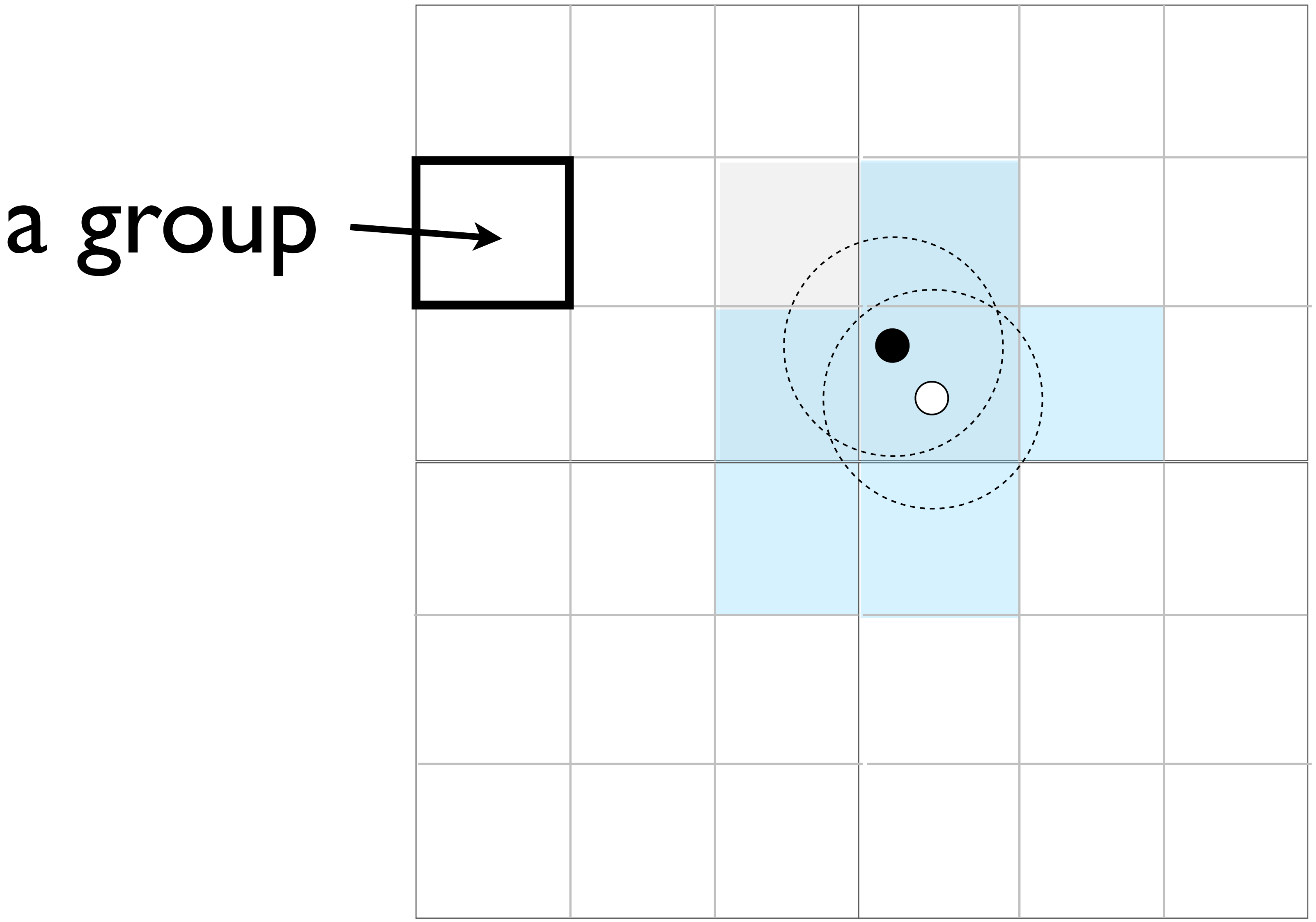
Anyone can send a message to a group (need not be a subscriber).



a group

a subscriber

Each cell is a group. A subscriber can subscribe to multiple cells. A group can have multiple publishers.



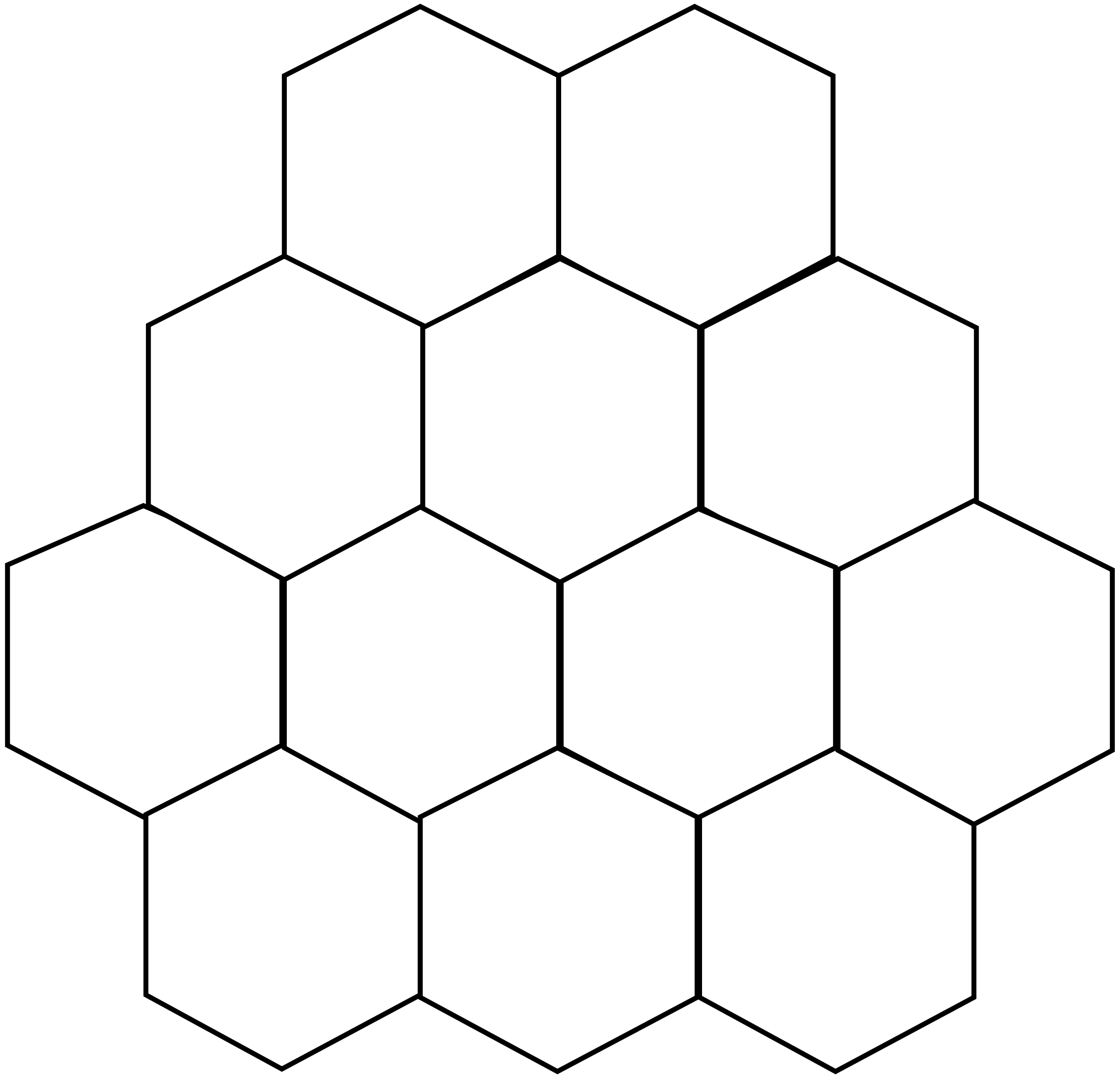
Previously

- Motivation for Interest Management
- Aura-based / Cell-based / General IM
- Publish / Subscribe Abstractions
- IP Multicast

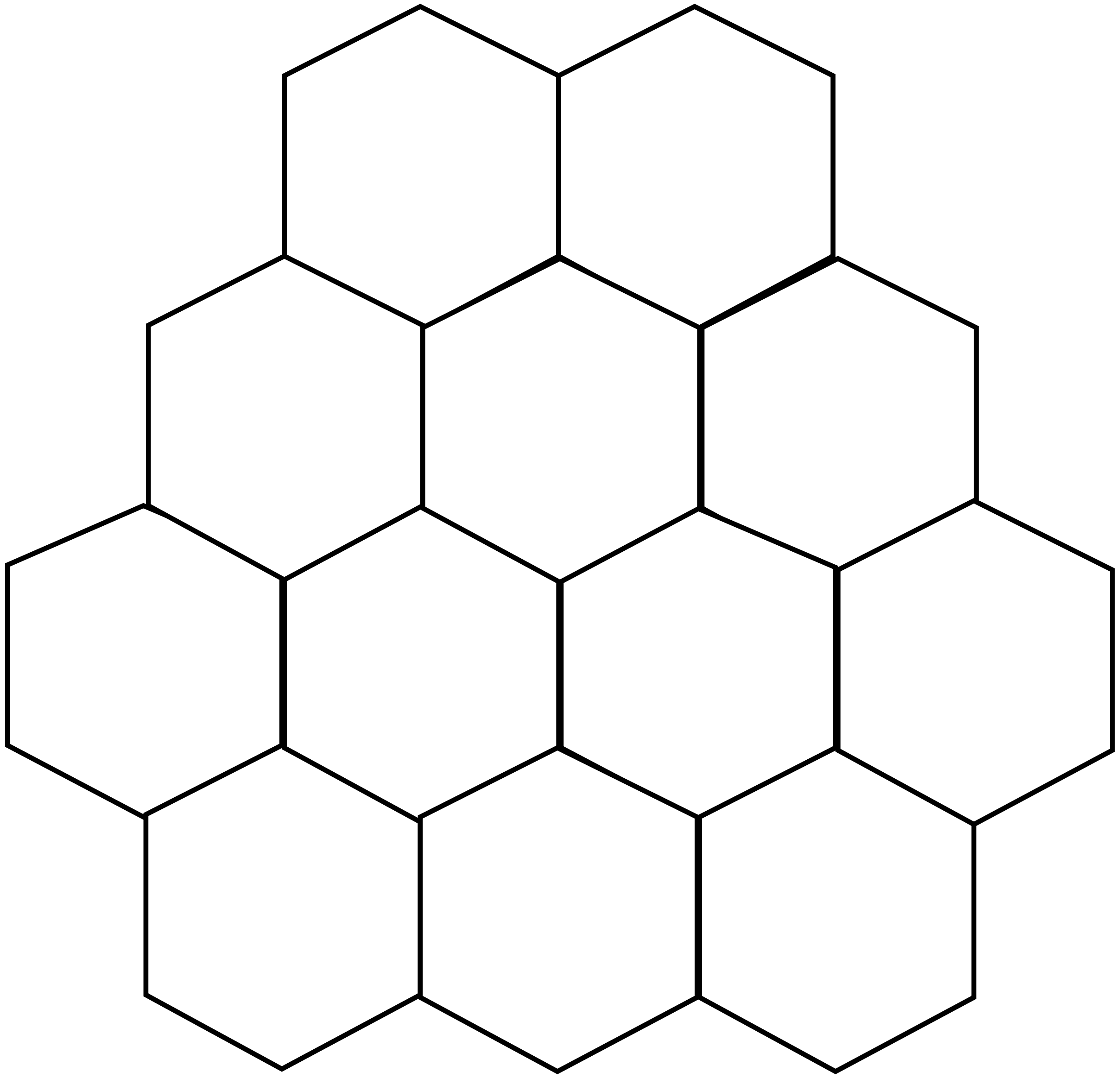
Cell-based

Is rectangle the best shape for a cell?

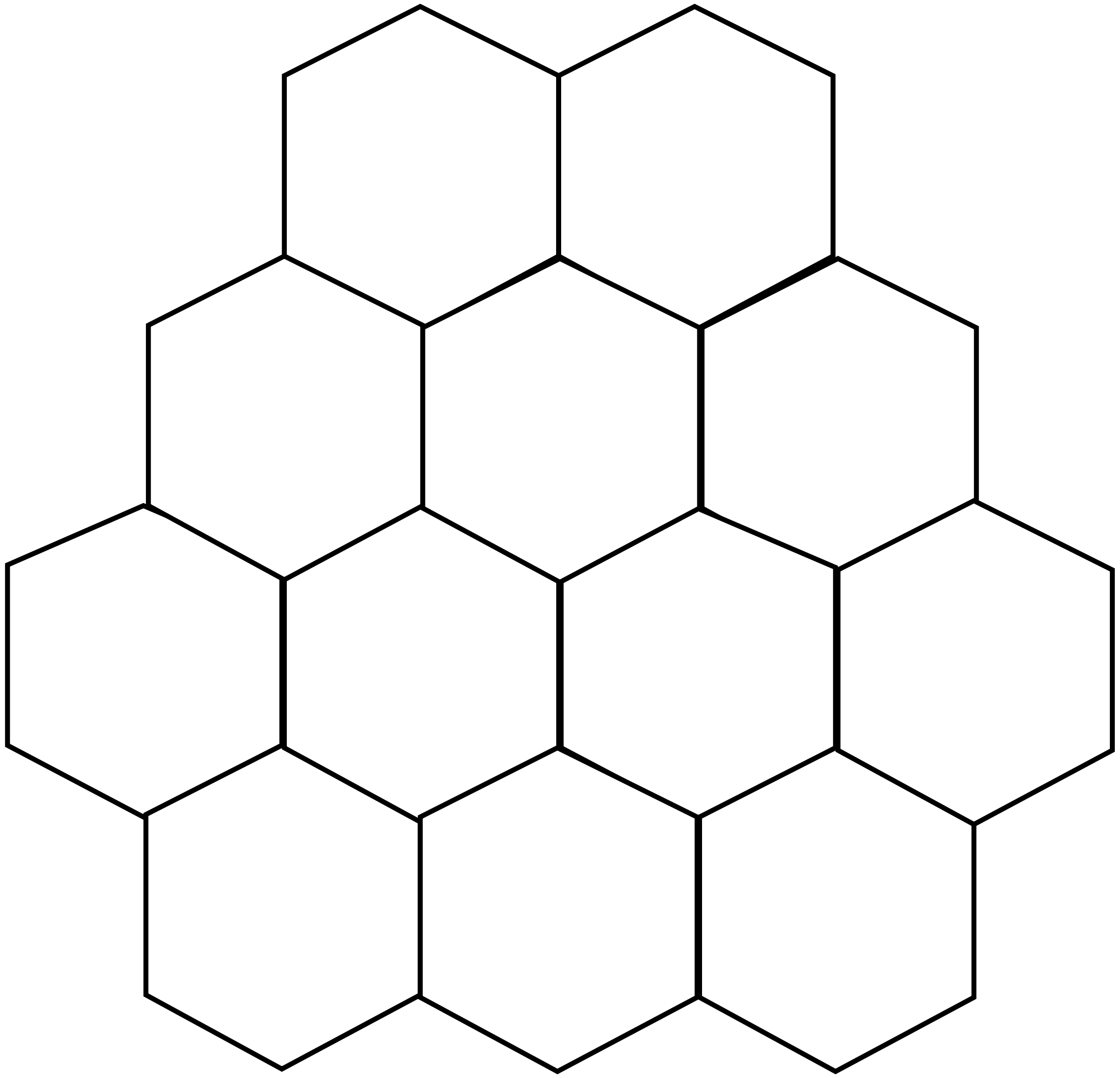
Hexagonal cells approximate a circle better.



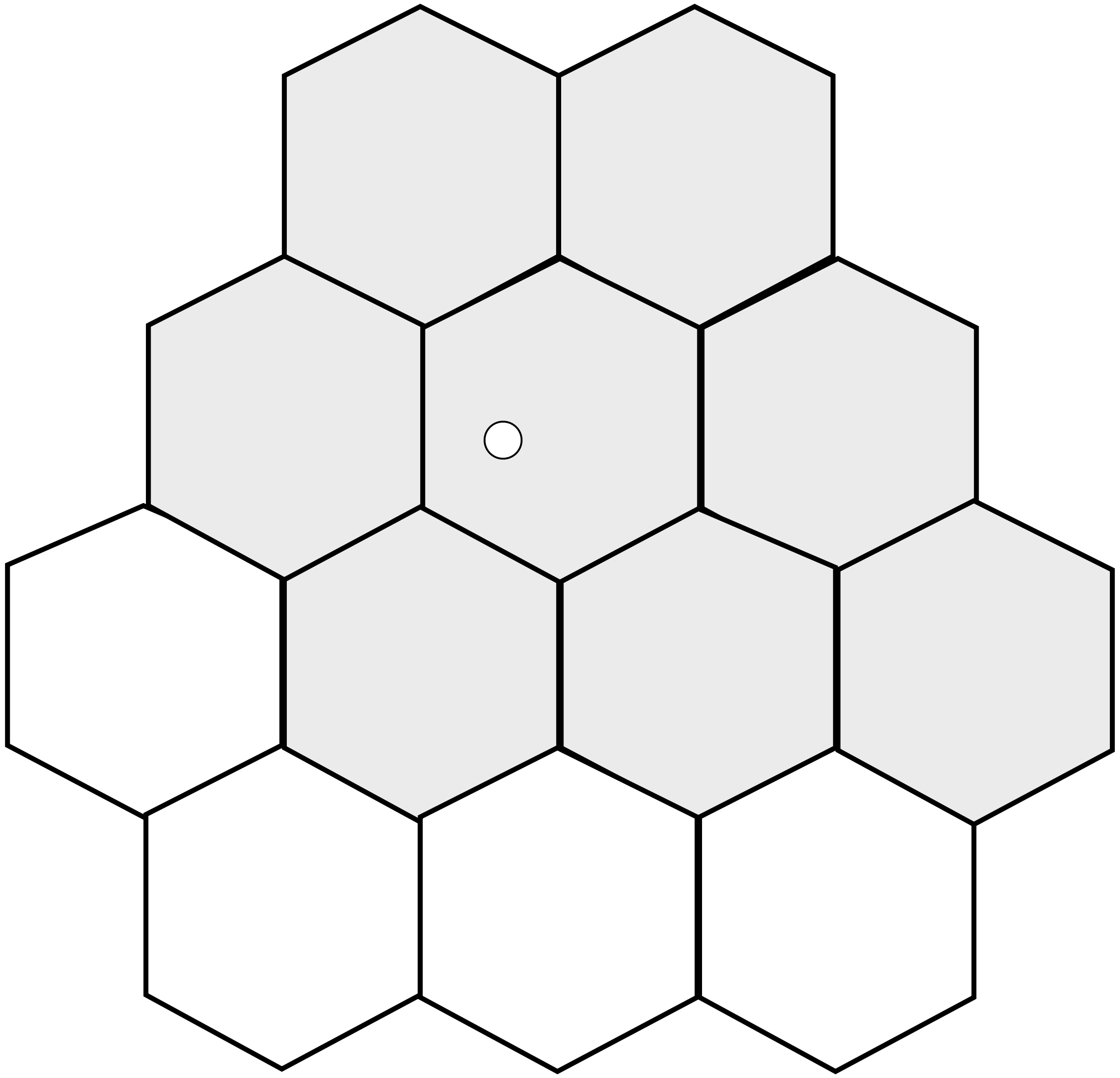
Require less subscribe/unsubscribe when moving.



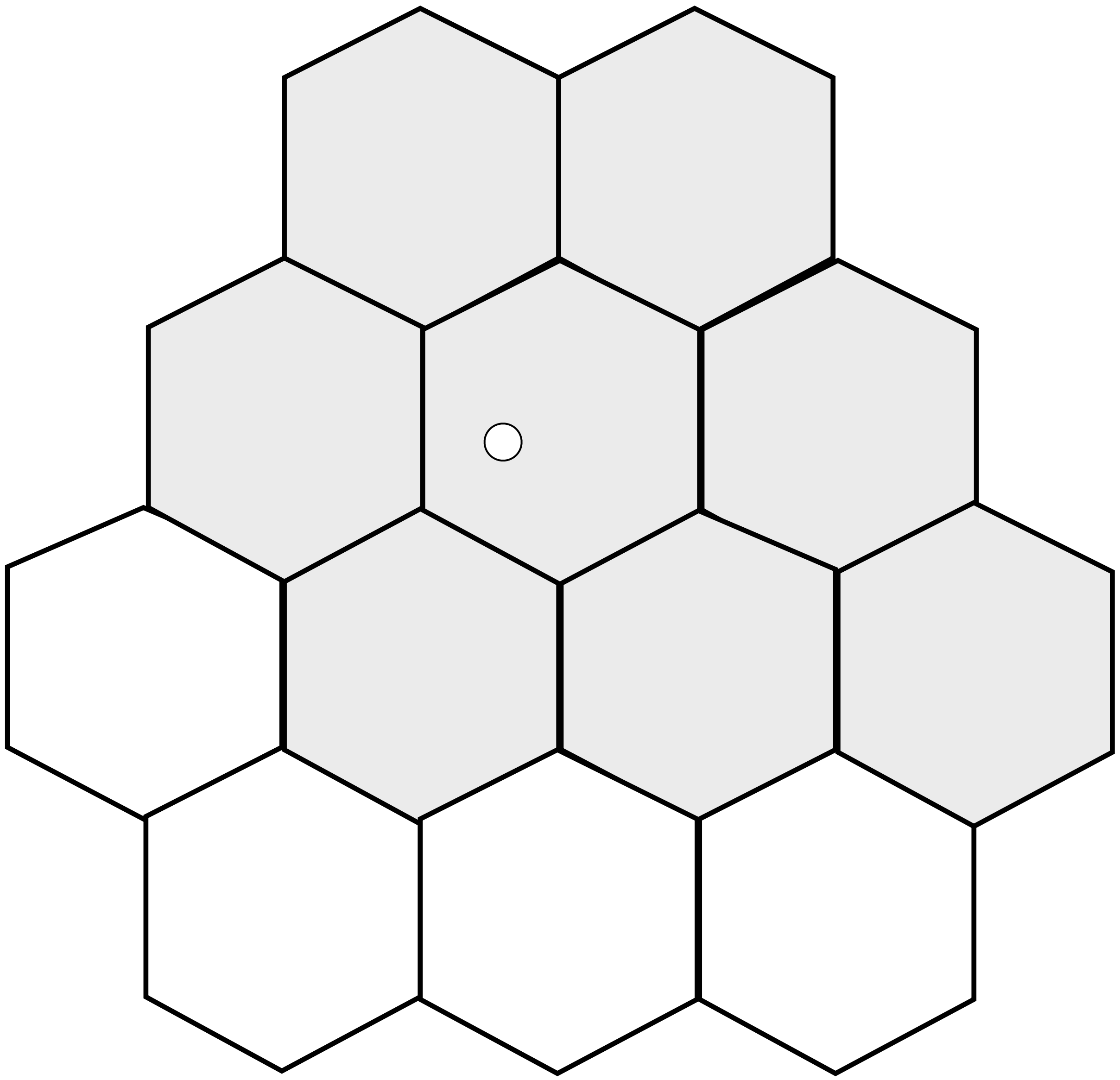
Assume a player is interested in it's current cell and surrounding cell.



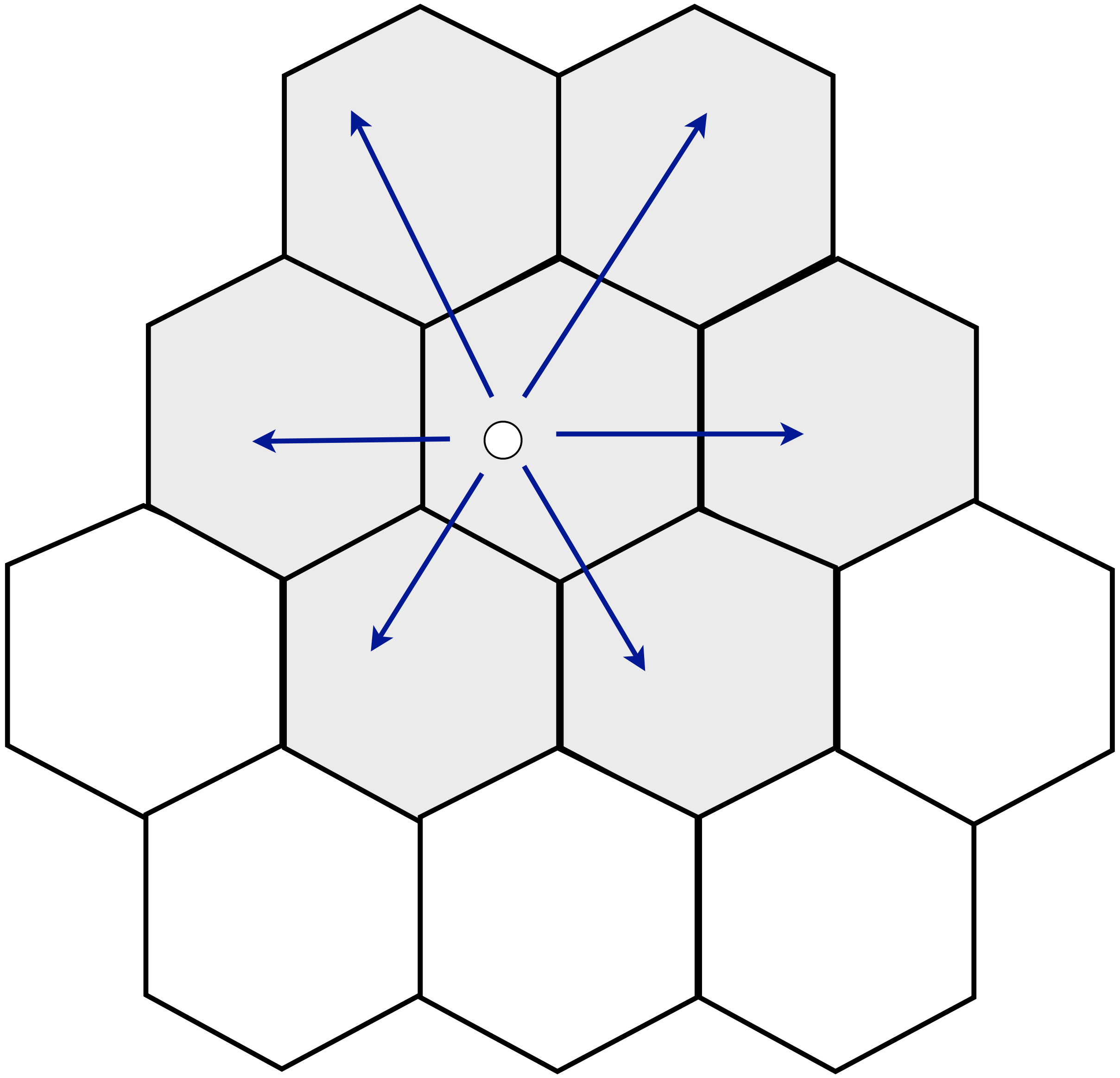
Assume a player is interested in it’s current cell and surrounding cell.



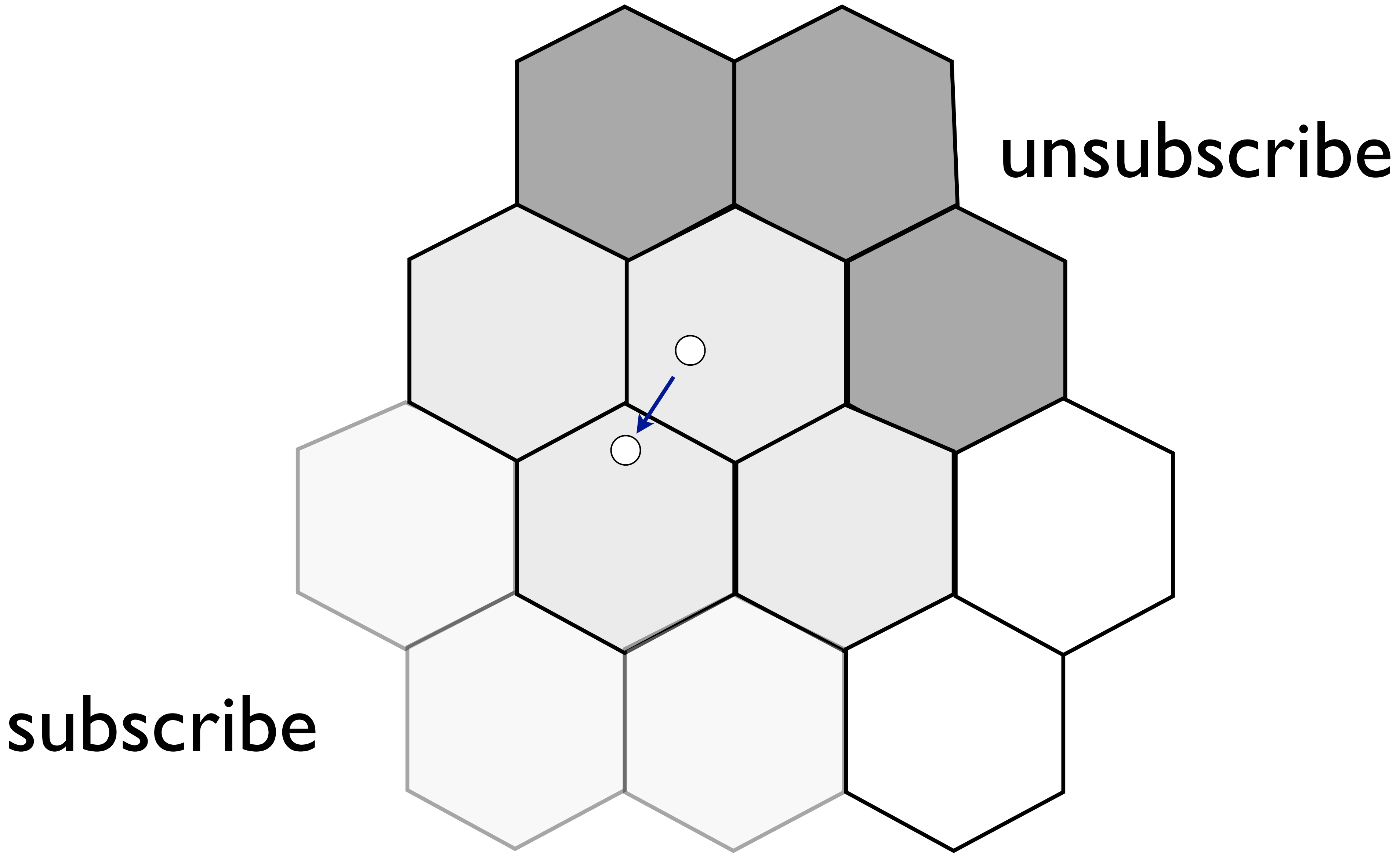
and moves to a neighboring cell with equal probability.



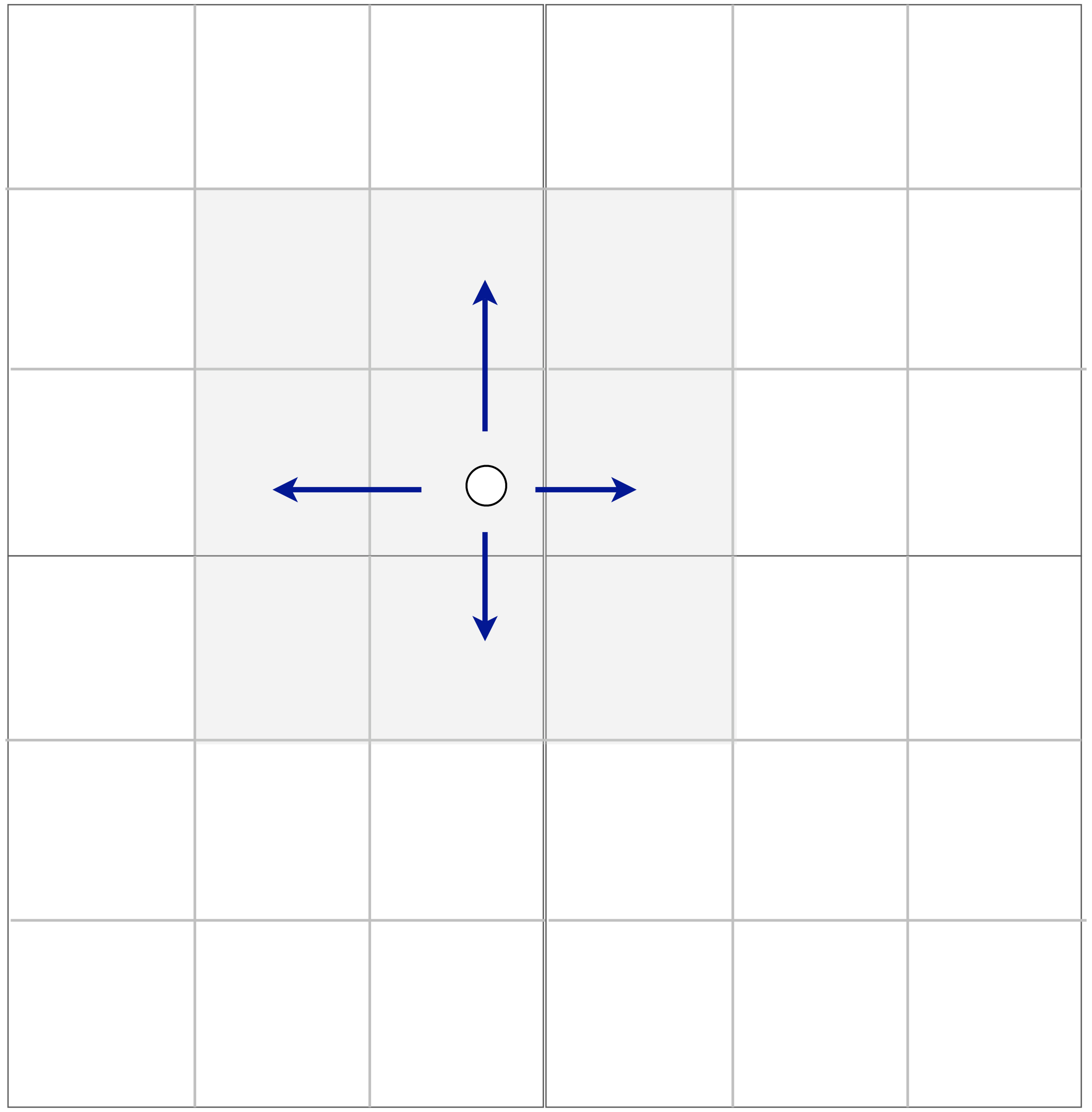
and moves to a neighboring cell with equal probability.



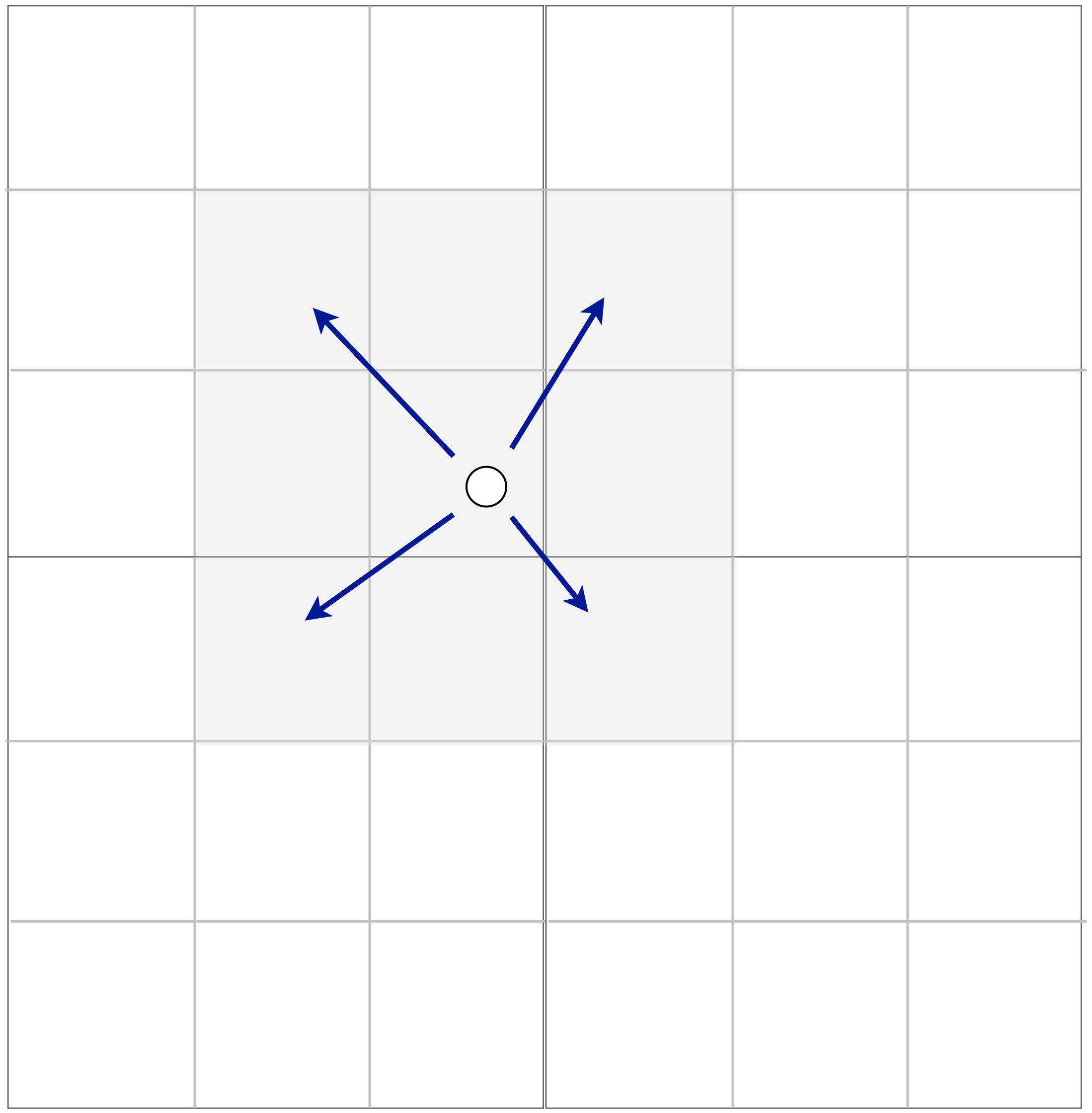
Every move requires **3** new subscriptions and **3** un-subscriptions.



Moving horizontally/vertically requires **3** new subscription and **3** unsubscriptions.



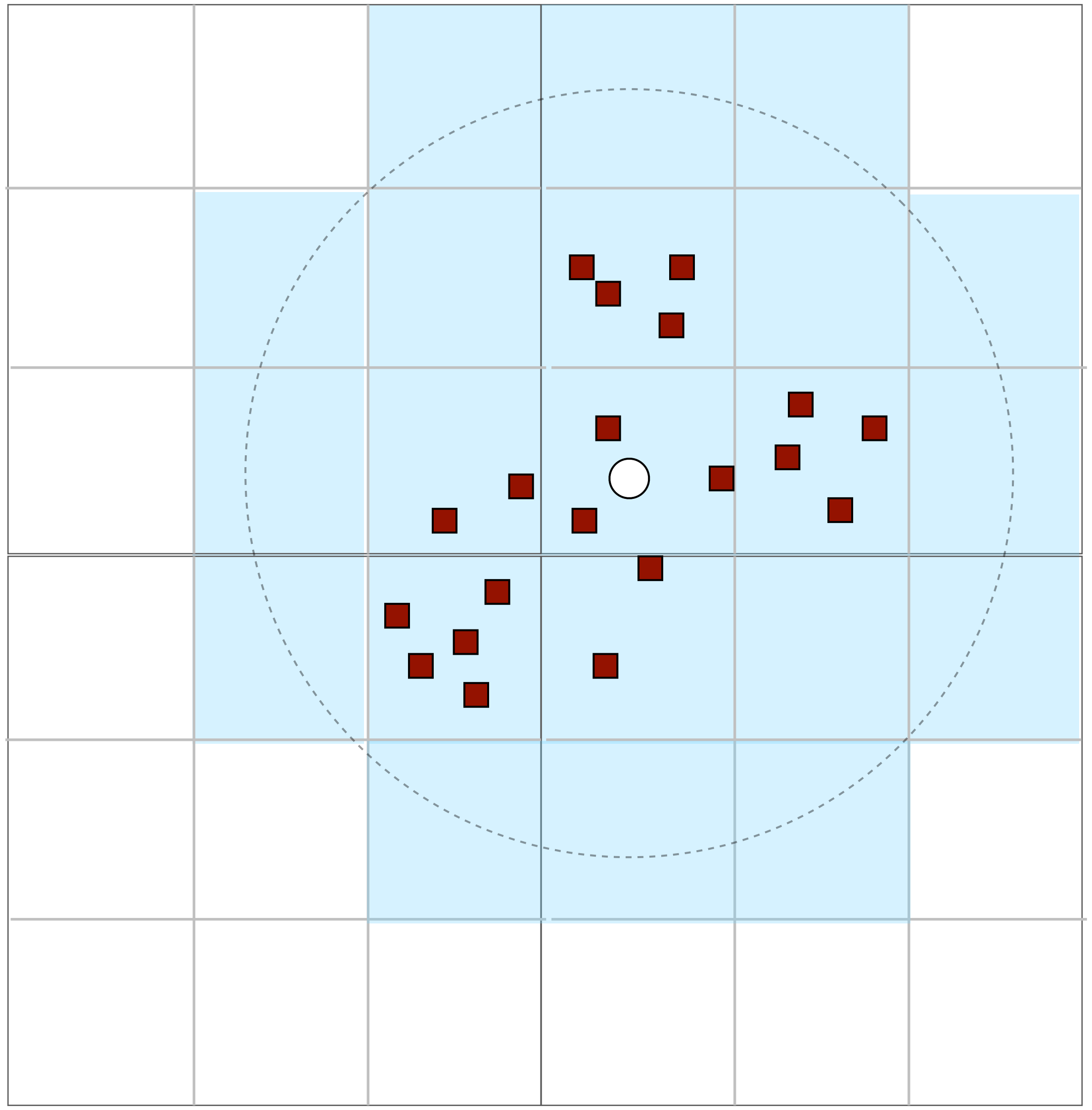
Moving diagonally requires **5** new subscription and **5** unsubscriptions.



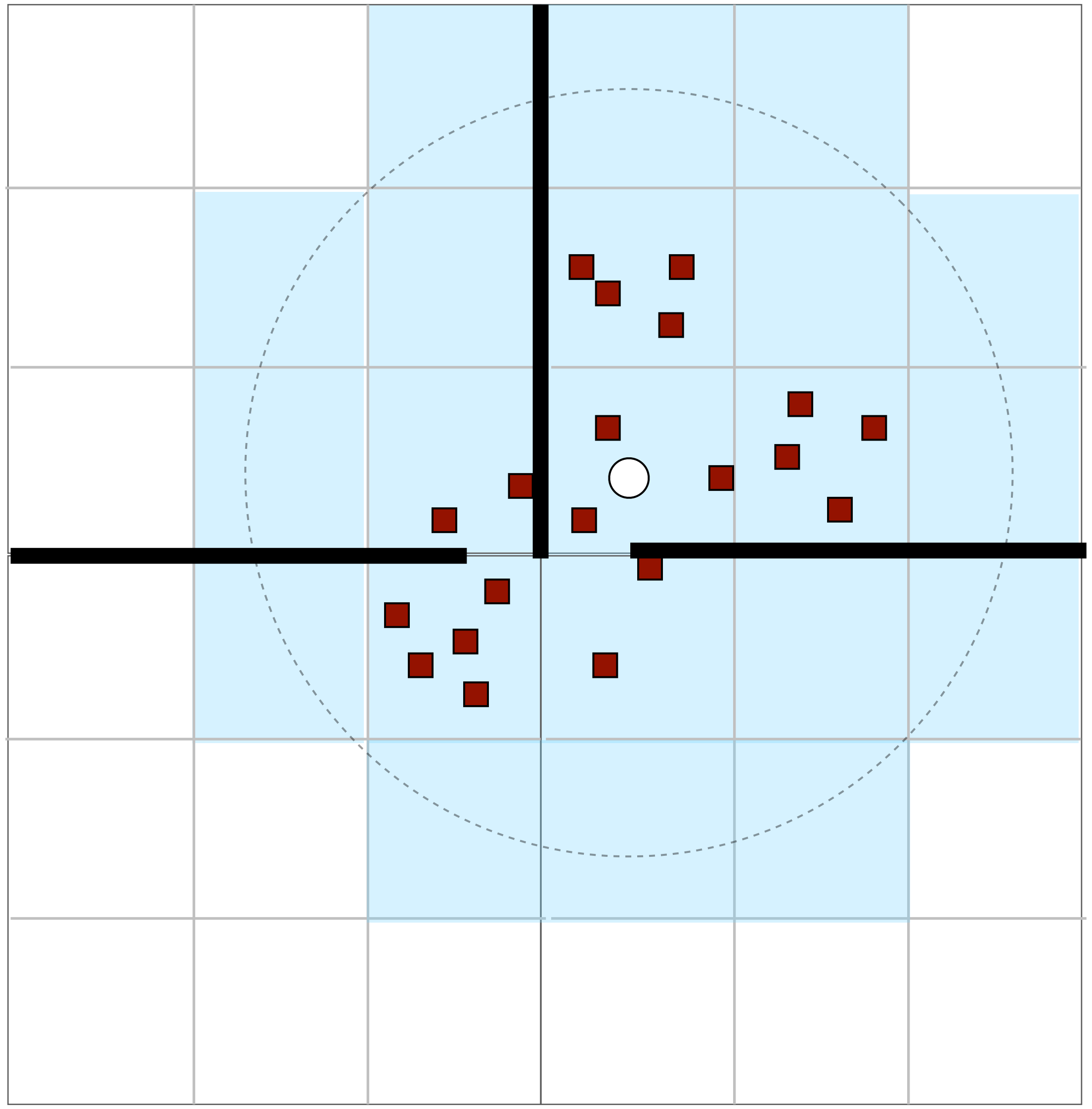
Hexgonal cells is better

1. rounder
2. less group join/leave

Visibility-Based Interest Management

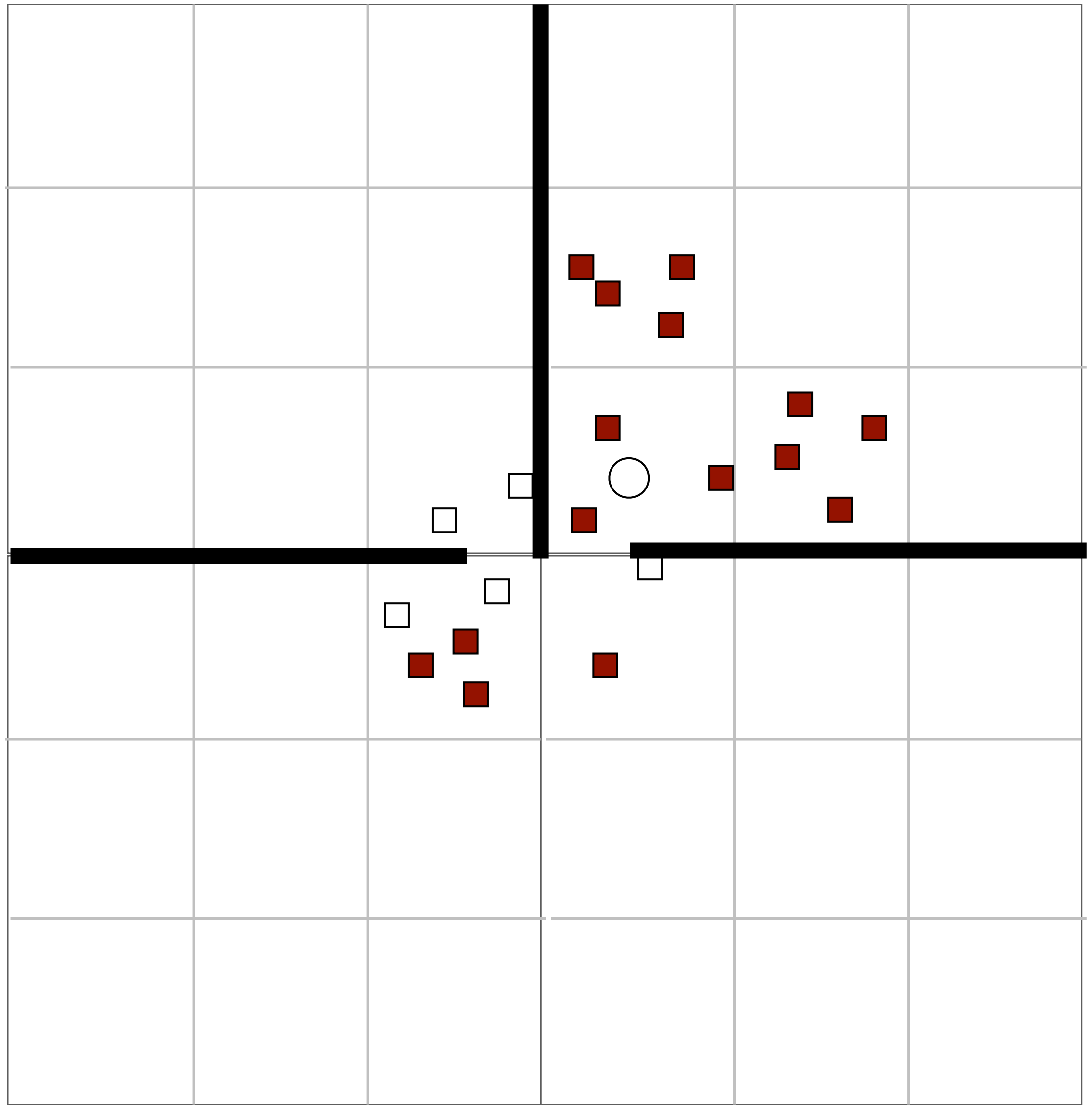


Ideally one should consider occlusion (we focus on visual occlusion)



A player *P* is interested in (events generated by) an entity *Q* if *P* can see *Q*, and *Q* is near *P*.

Ideally one should consider occlusion (we focus on visual occlusion)



need not be binary: can generalize to multi-level of interest depending on distance

Ray Visibility

Interest Management

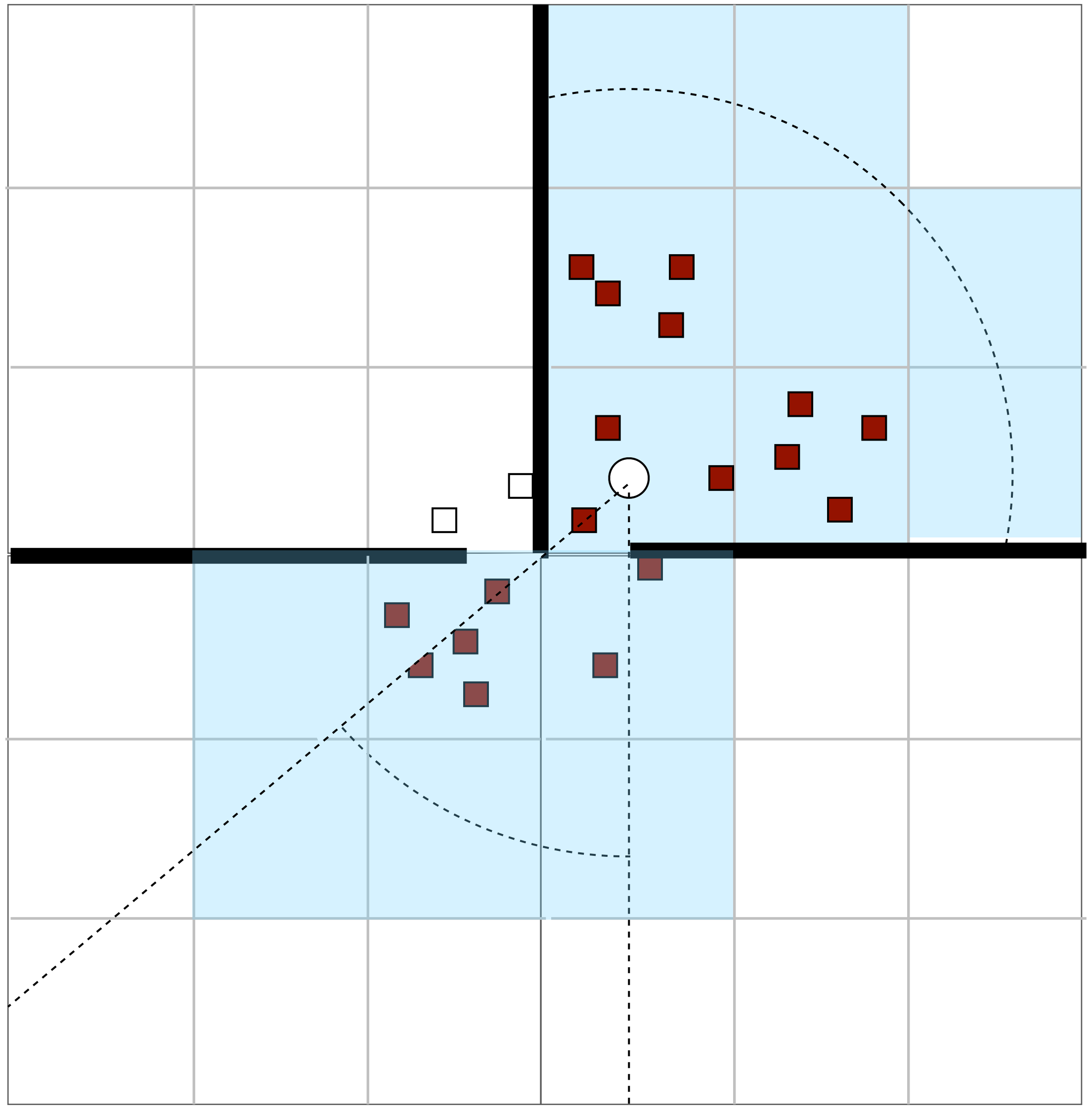
Object-to-Object Visibility

1. Expensive
2. Frequent re-calculations.

but gives exact visibility.

A player *P* is interested in (events generated by) an entity *Q* if *P* can see *Q*’s cell, and *Q* is near *P*.

Object-to-Cell Visibility



Object-to-Cell Visibility

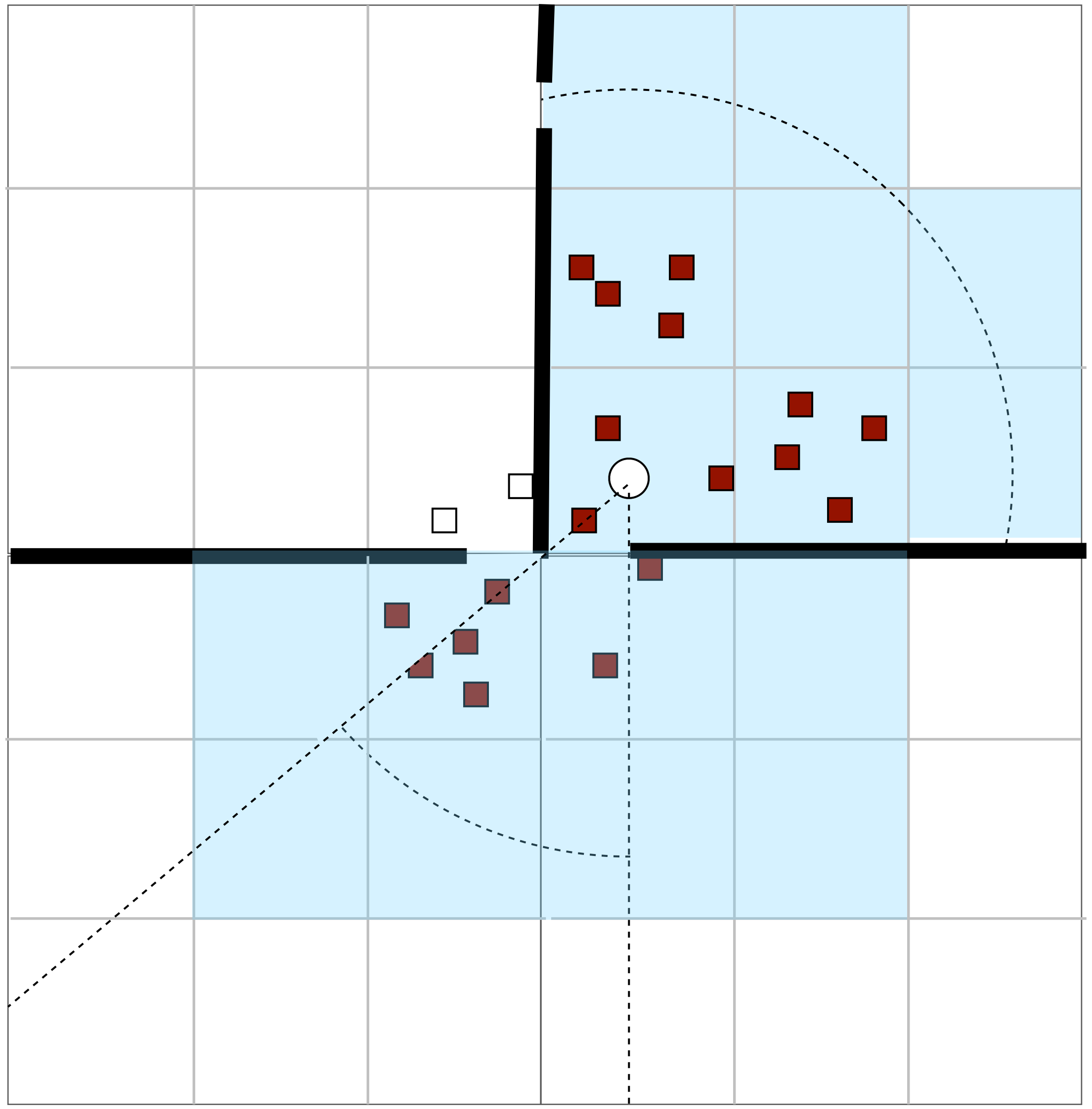
1. Less expensive
2. Less frequent re-calculations
3. Less accurate

When player moves, still need to recompute visible cells.

A player *P* is interested in (events generated by) an entity *Q* if *P*'s cell can "see" *Q*'s cell, and *Q* is near *P*.

i.e., there exists in a point in *P*’s cell that can see a point in *Q*’s cell, and *Q* is near *P*.

Cell-to-Cell Visibility



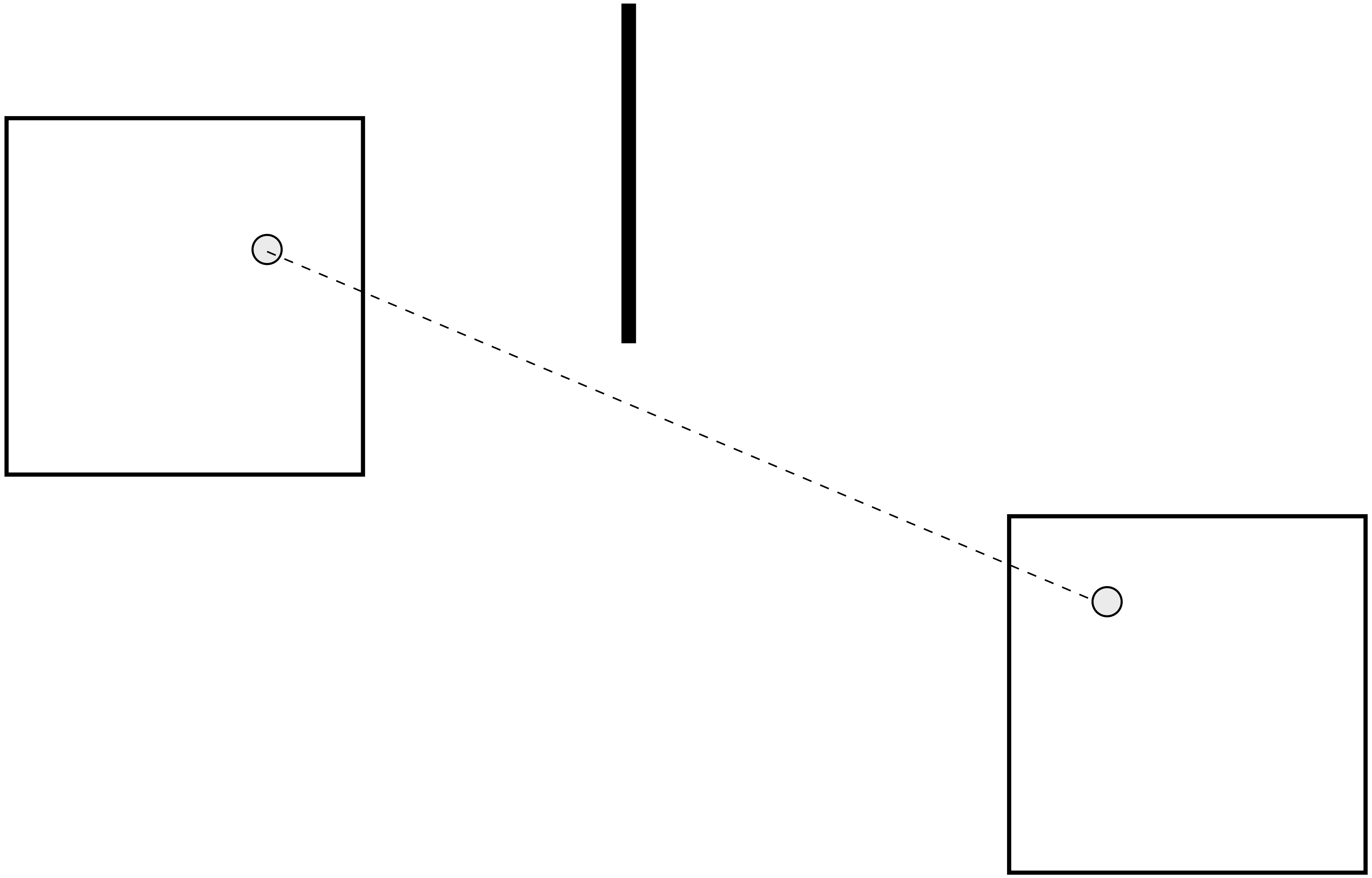
Cell-to-Cell Visibility

1. Much Less expensive
2. Calculate once!

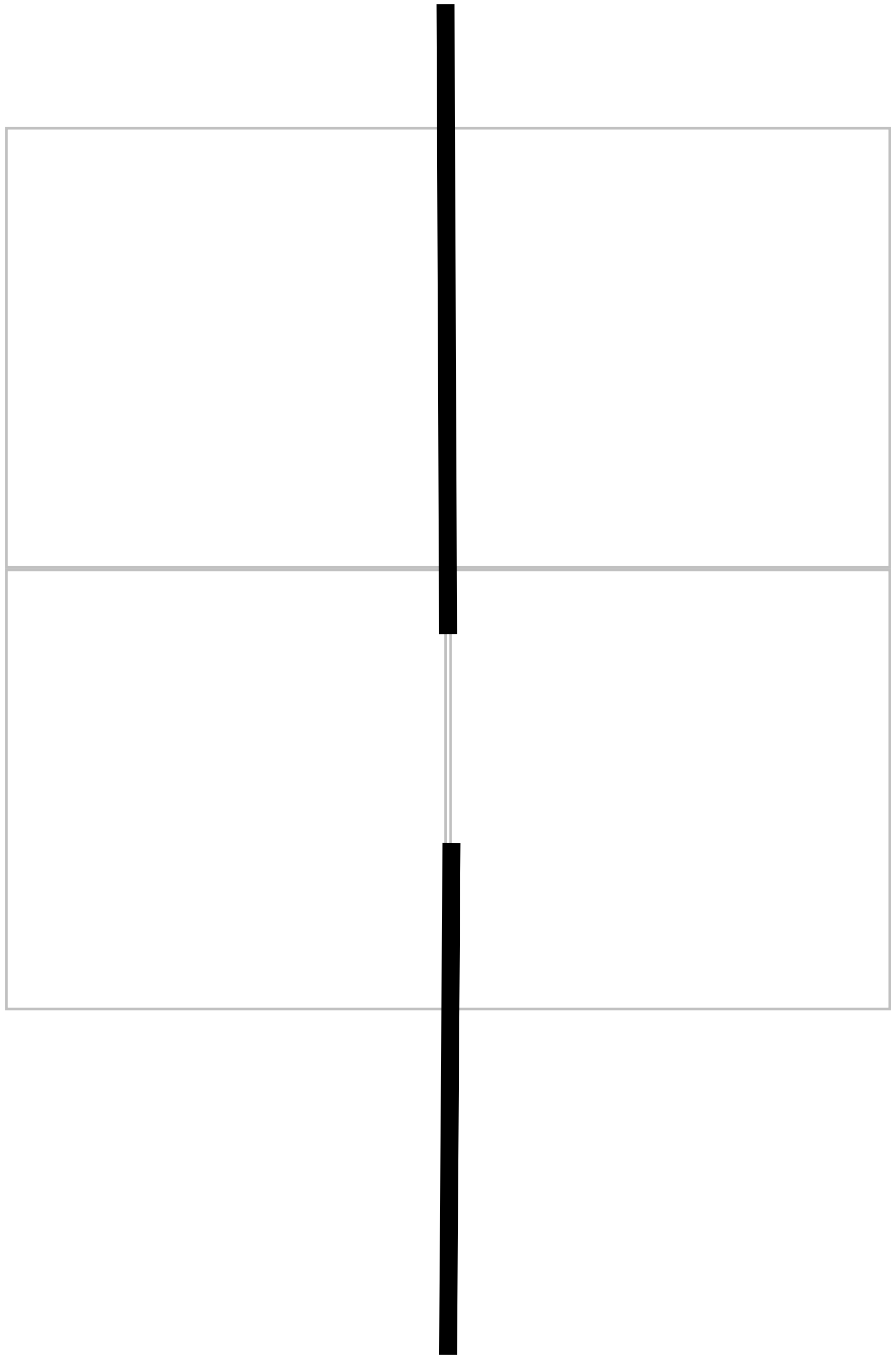
but even less accurate.

Computing
Cell-to-Cell Visibility

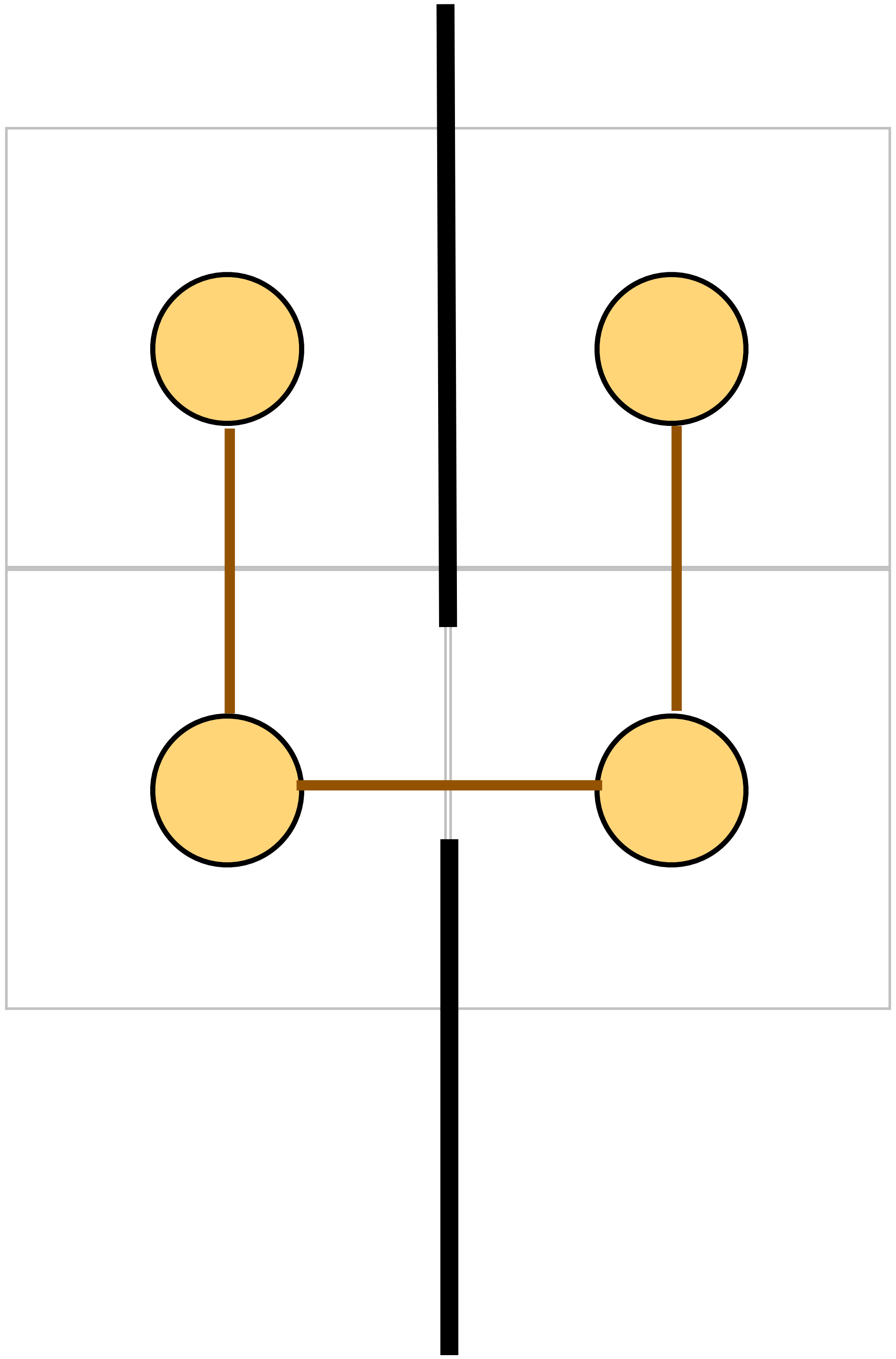
Check if there exist two points, one in each cell, that can see each other (can draw a line without passing through occlusion)



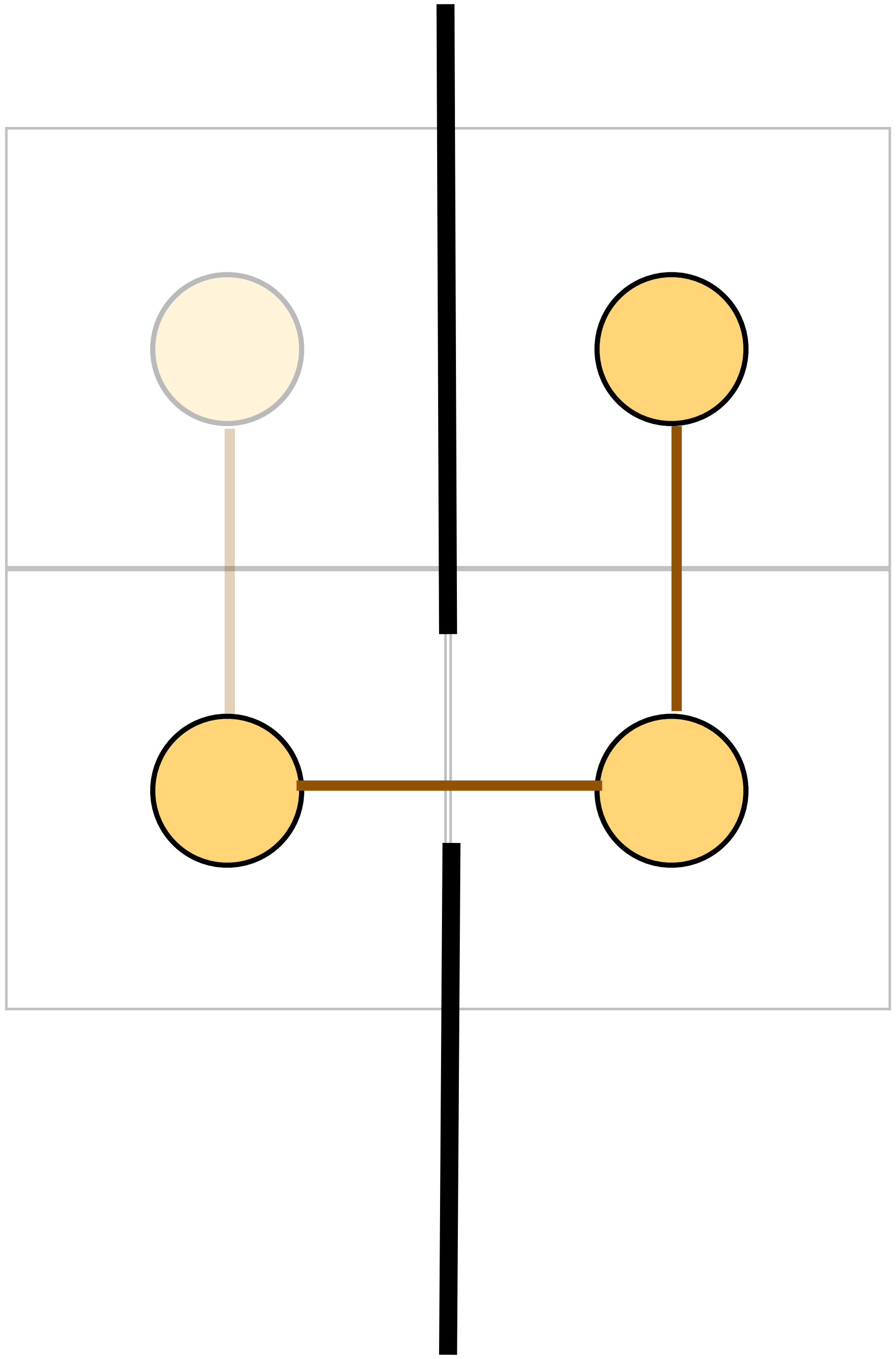
Trivial case: if two cells are adjacent and the boundary is not completely occluded.



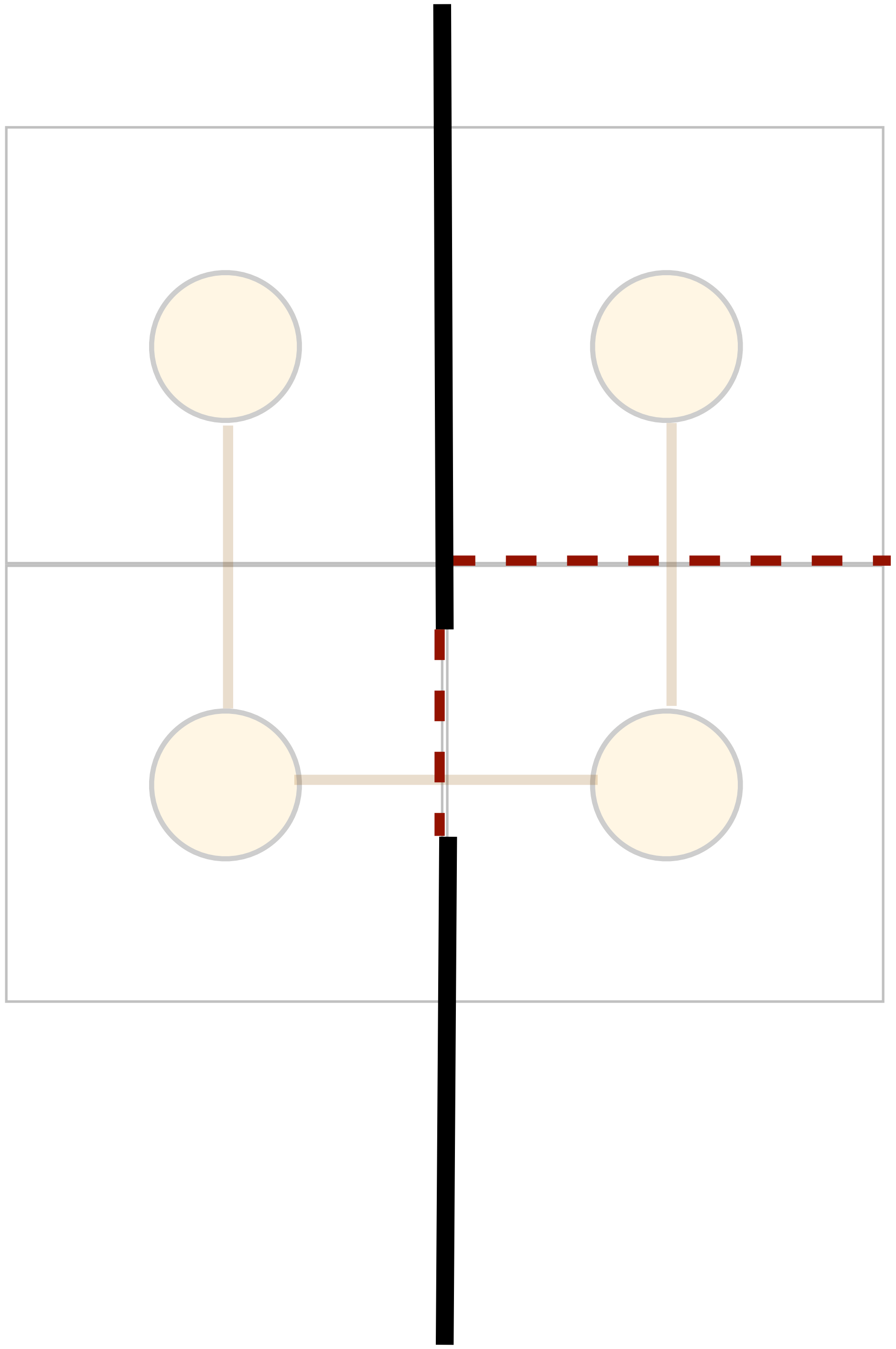
Build a graph of cells -- connect two vertices
if they share a boundary and is visible to each other.



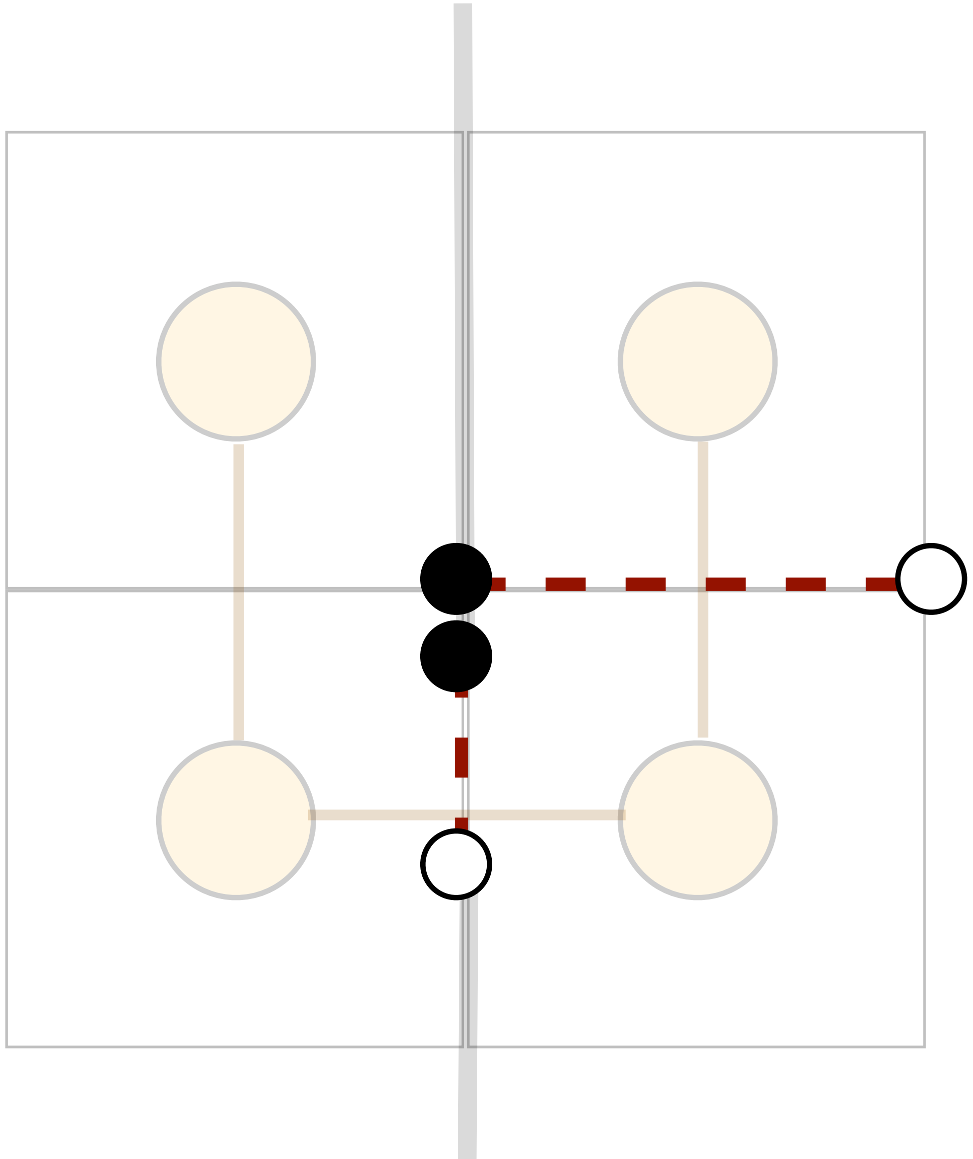
if two cells are not-adjacent, then for them to be visible to each other, there should exists a path between them, and ...



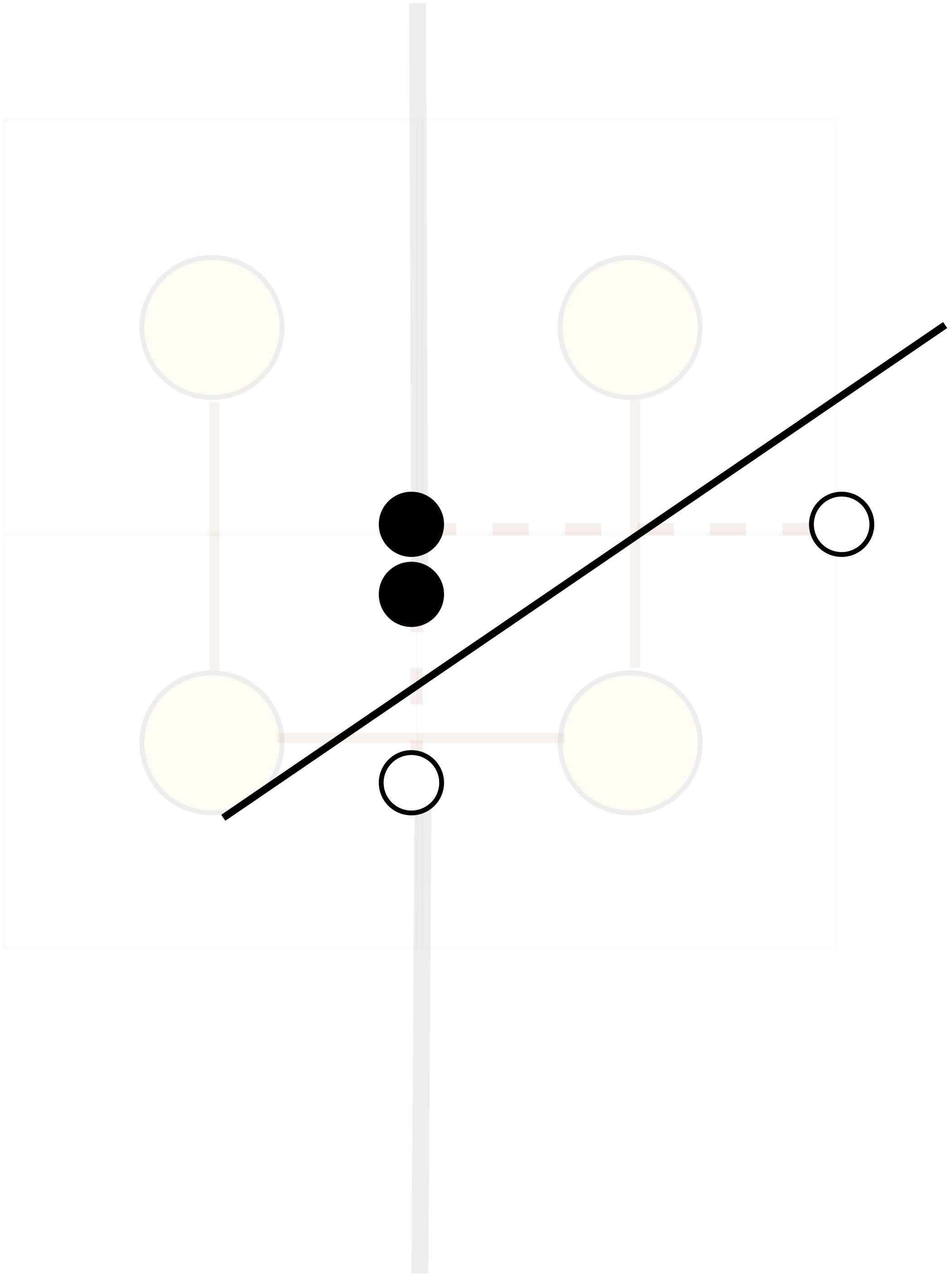
consider the non-occluded boundaries along path..



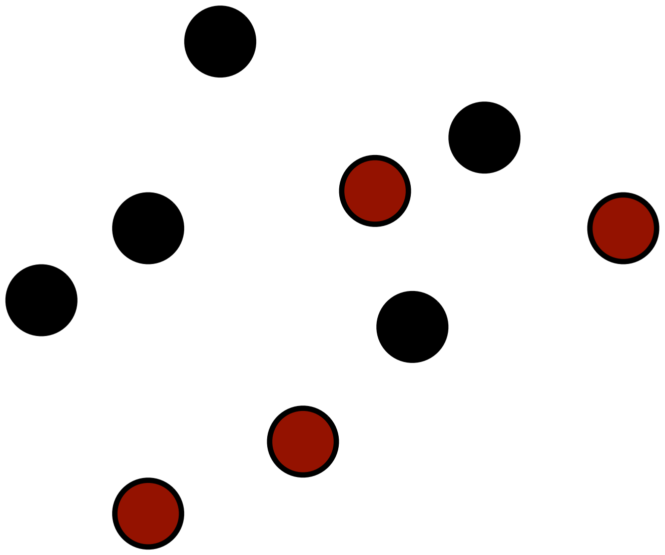
The set of points on the left L and right R can be separated by a line.



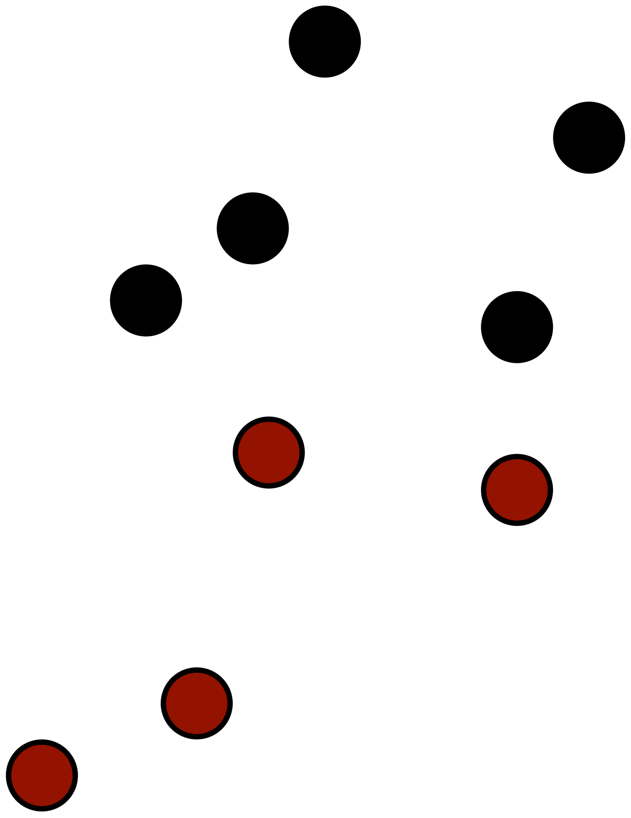
The set of points on the left L and right R can be separated by a line.



Linearly Separable Point Sets

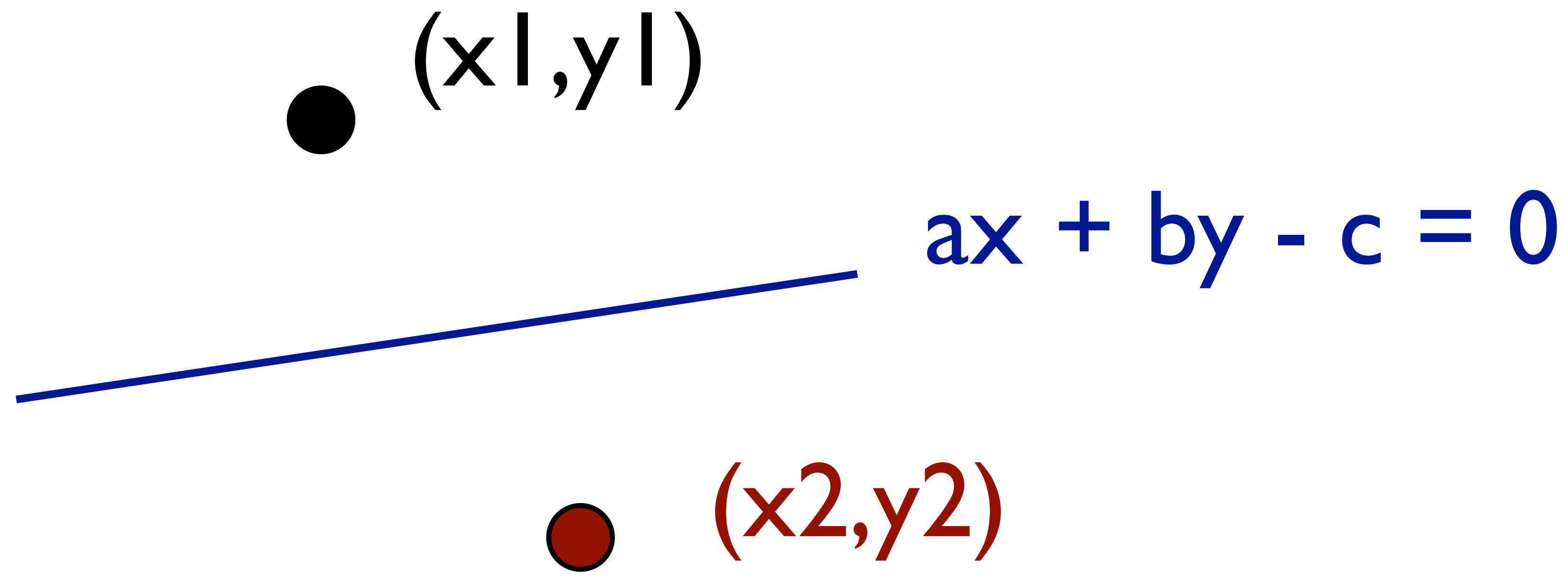


no



yes

We can model this problem as a set of linear equations.

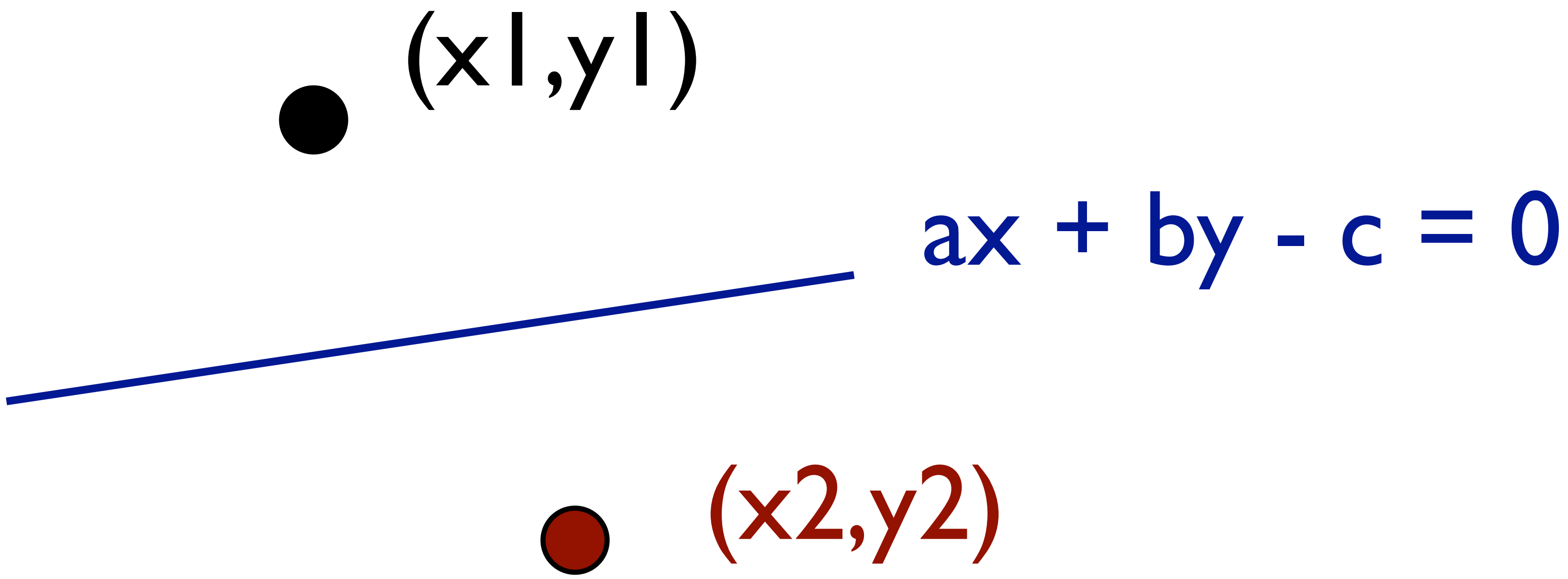


Find a solution (a, b, c) for the following:

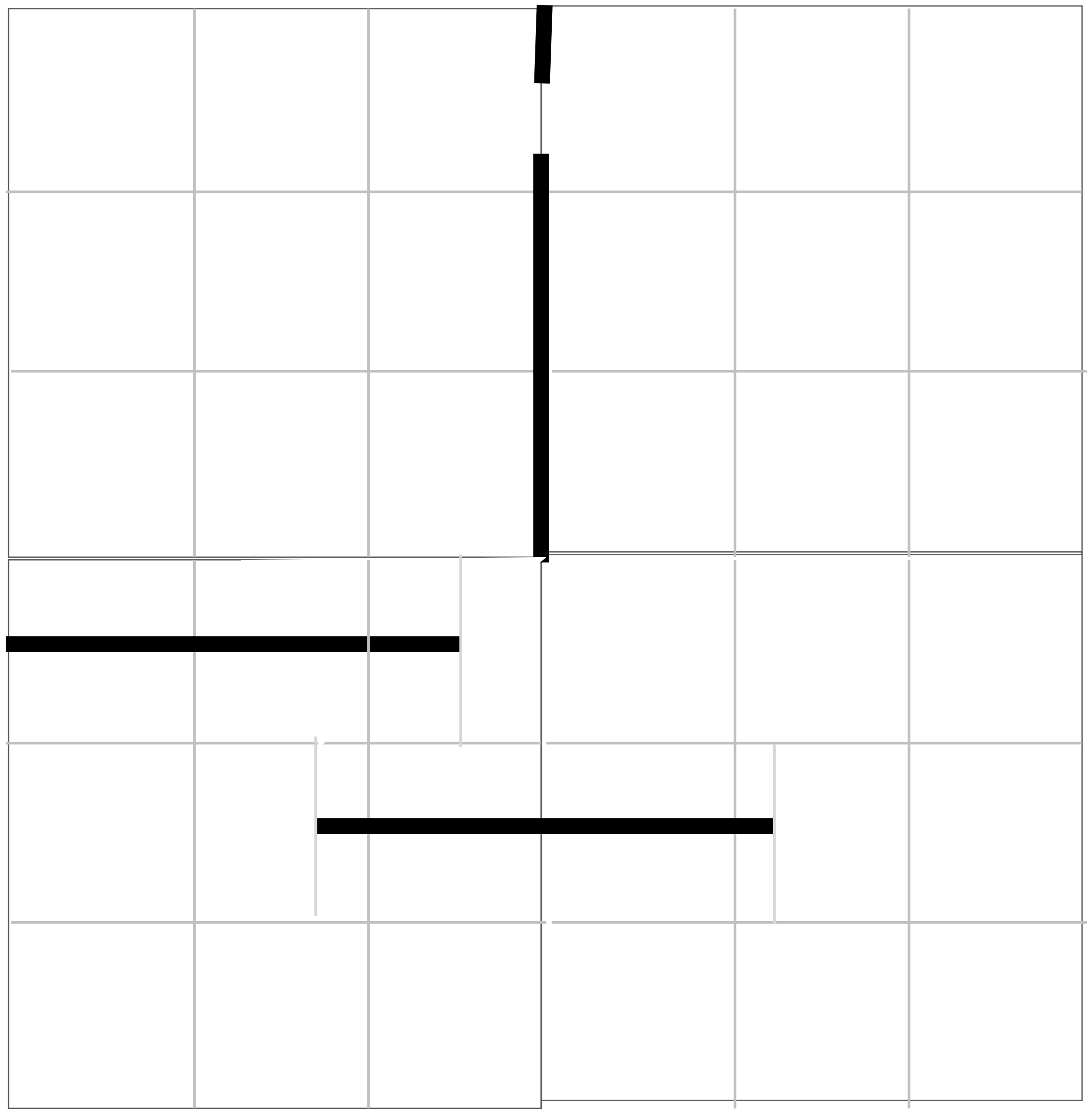
$ax + by - c = 0$

$ax1 + by1 - c > 0$ for all $(x1,y1)$ in L

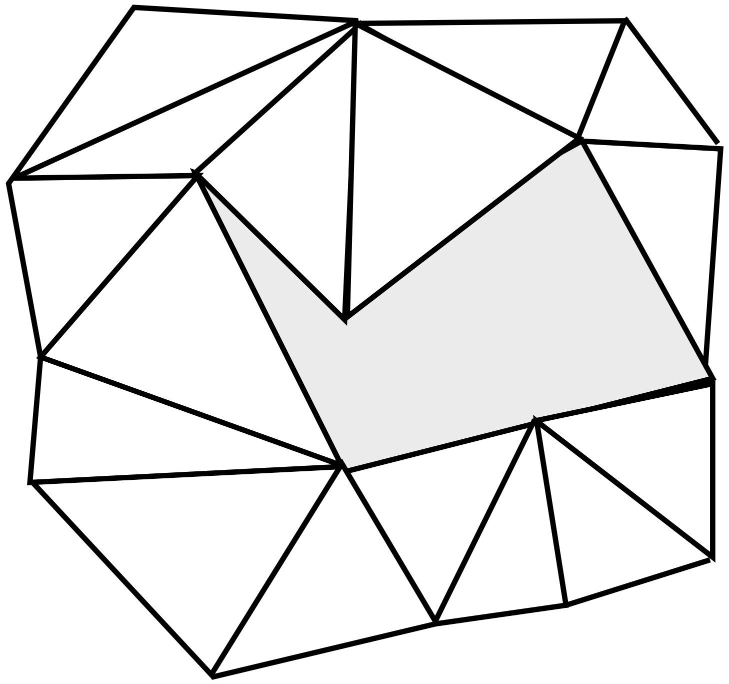
$ax2 + by2 - c < 0$ for all $(x2,y2)$ in R



We can break into smaller cells if occlusion is not aligned with boundary of cells.



(Irregular) triangular cells can adapt to any polygonal occlusions.



Note: Rendering engine usually compute visibility information which we may be able to reuse in the Interest Management module.

Recap:

Shape of cells

Visibility-based IM

Pre-computing C2C Visibility

Generalized Interest Management

Example: Interested in
(i) objects around avatar
(ii) buildings in a region
(iii) the opponent’s avatar

Subscription can be based on any attributes (not just position)

We can view each object as occupying a multidimensional space (each attribute is a dimension)

A subscription specify a region in the same space.

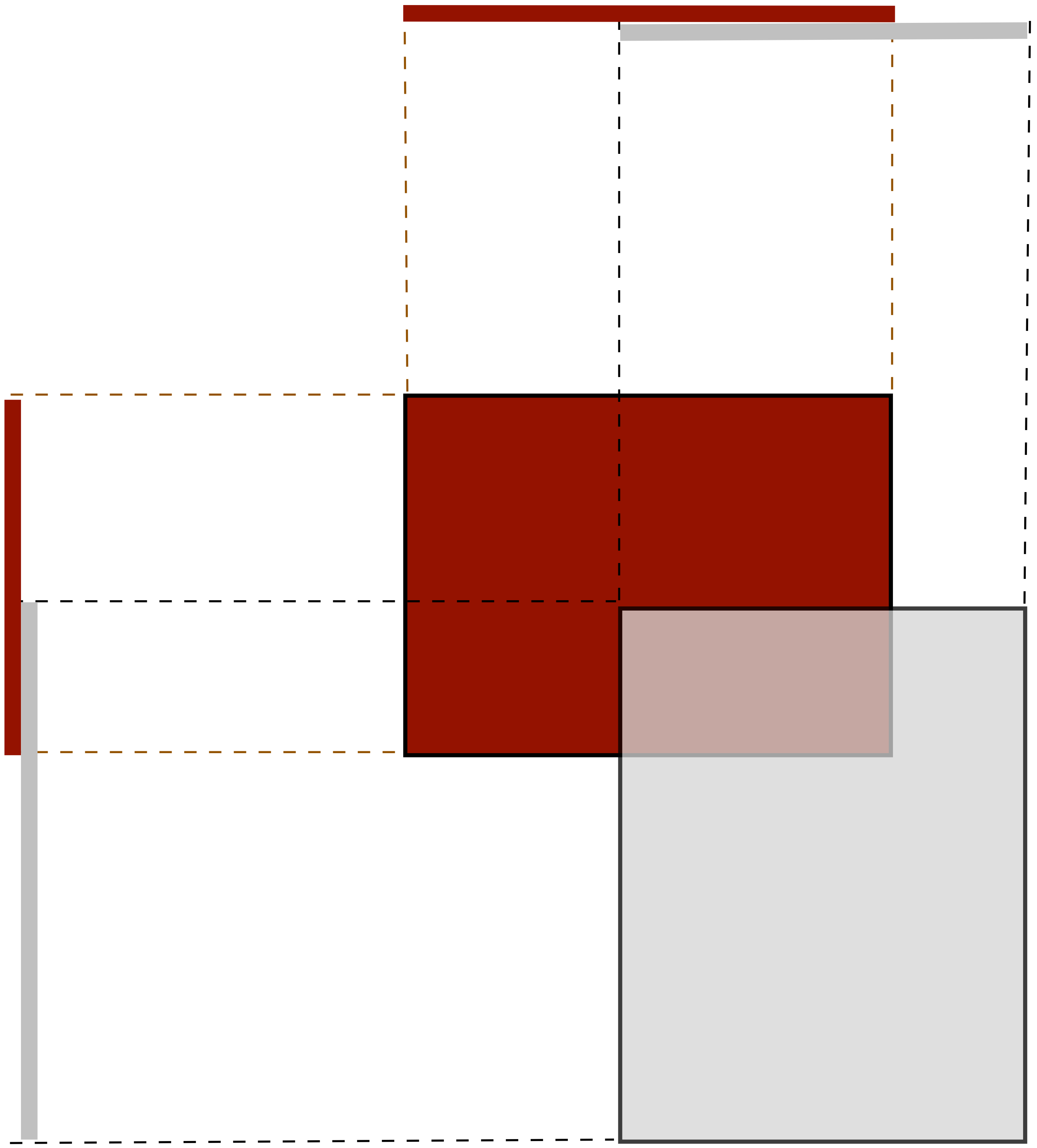
When an **update region** of an entity *P* **intersects** the **subscription region** of entity *Q*, updates of *P* is sent to *Q*.

How to test if two regions overlap in k-dimensional space?

Naive approach: O(nm) for *n* update region and *m* subscription region.

Dimensional Reduction

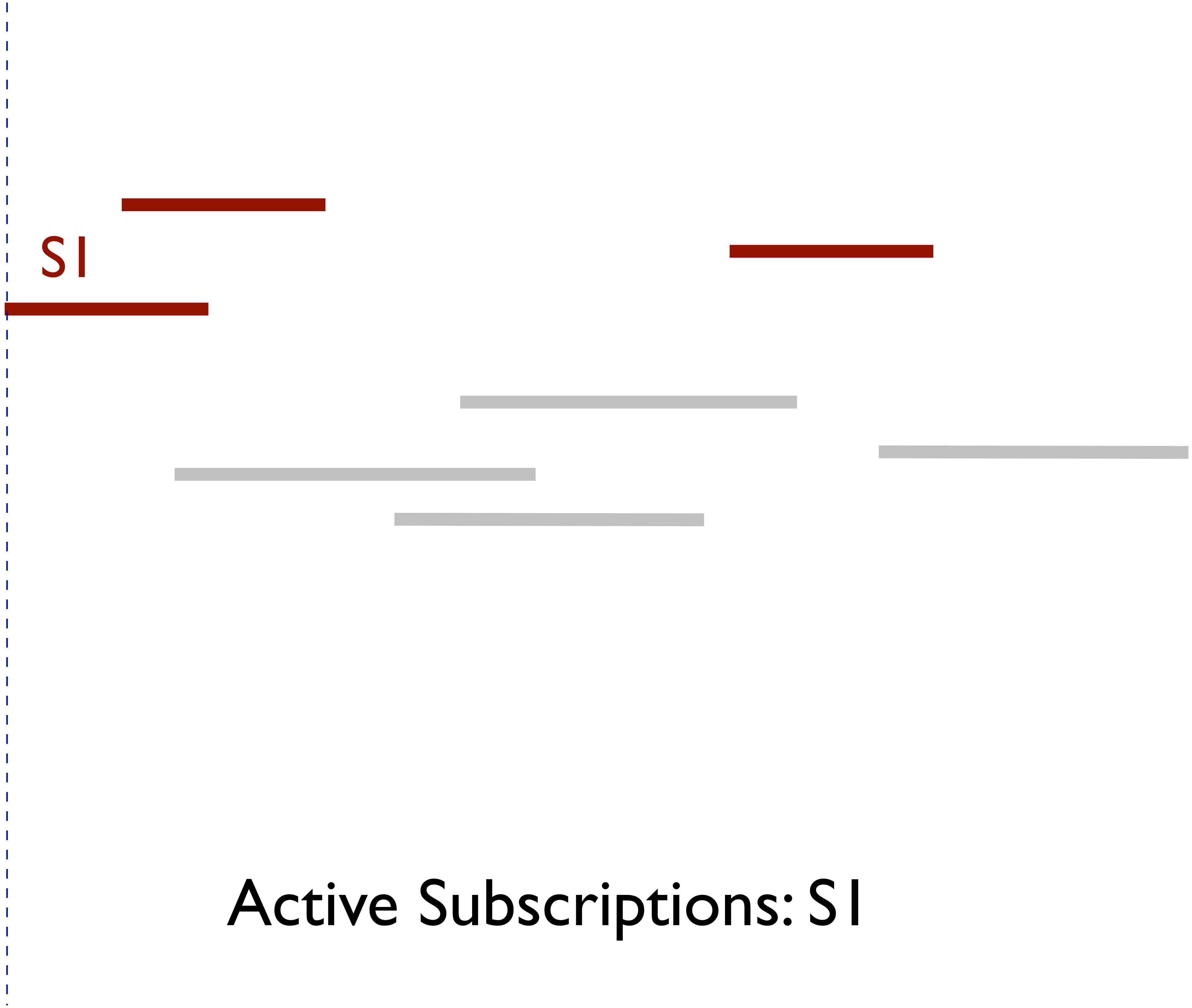
If 2 regions overlap, then they overlap in each of the individual *k* dimension.



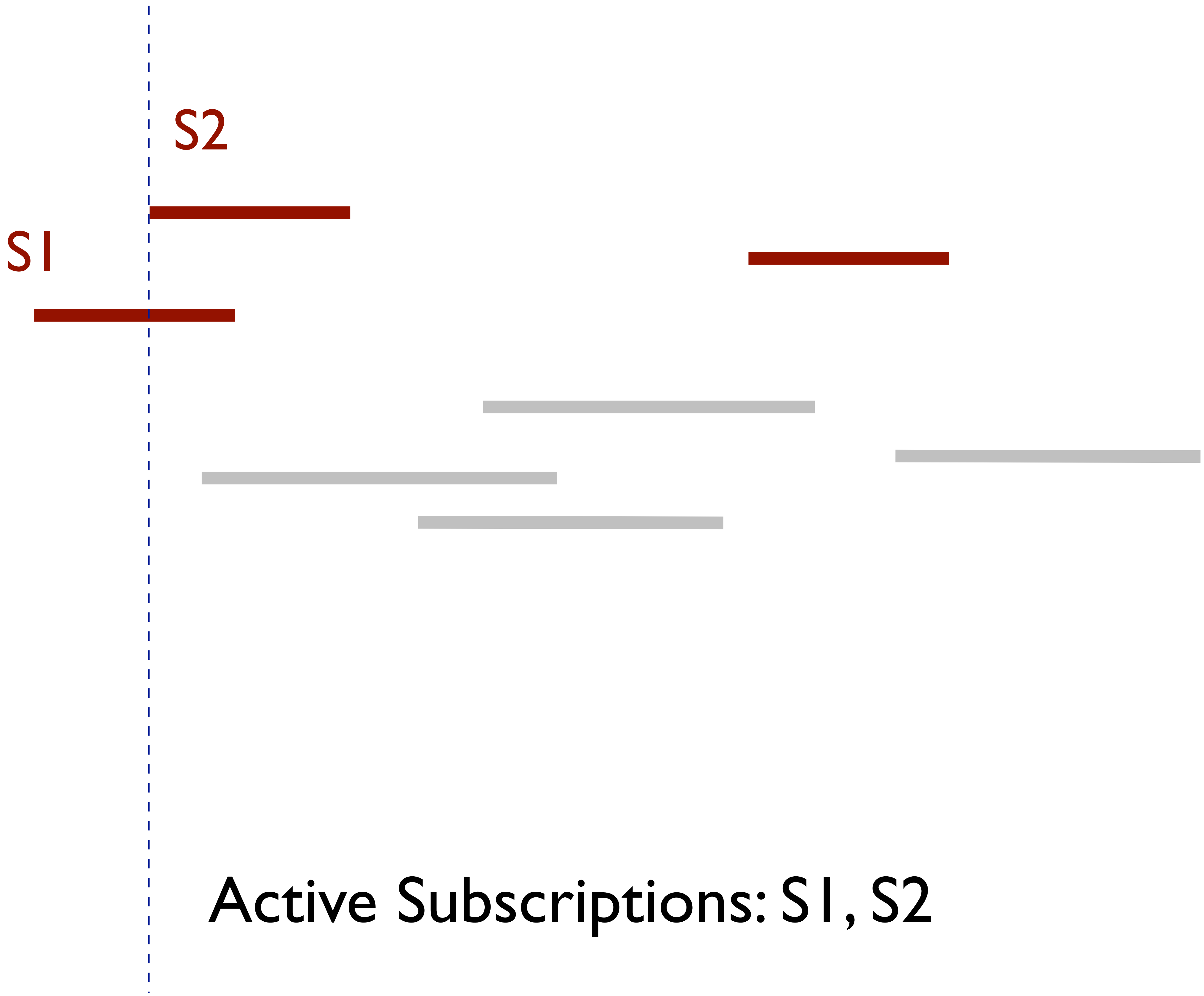
How to test if two intervals overlap?

Step 1: Sort all end points and put into a list L

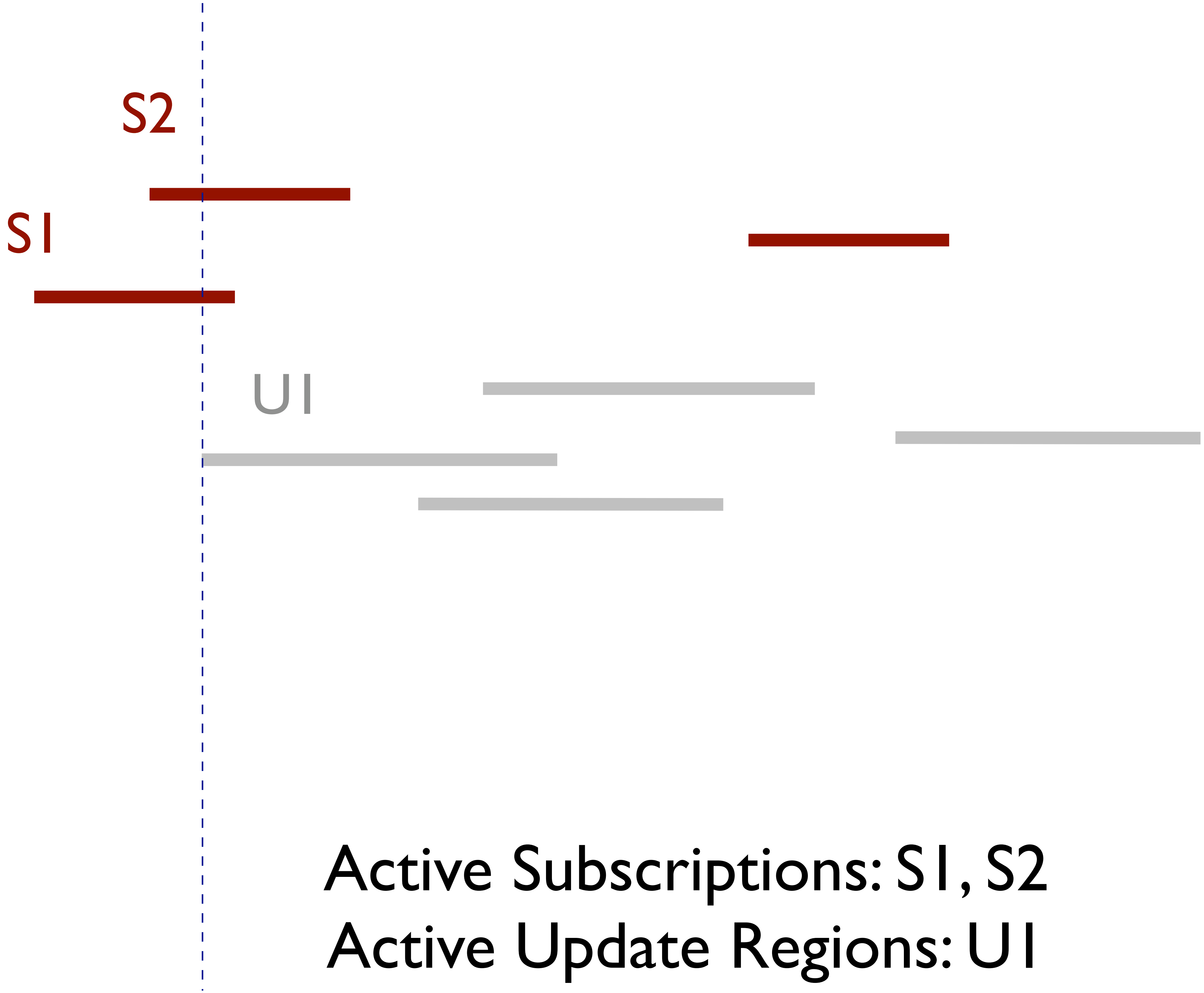
Step 2: Scan from left to right. Remember all active subscription regions S and all active update regions U.



Active Subscriptions: S1

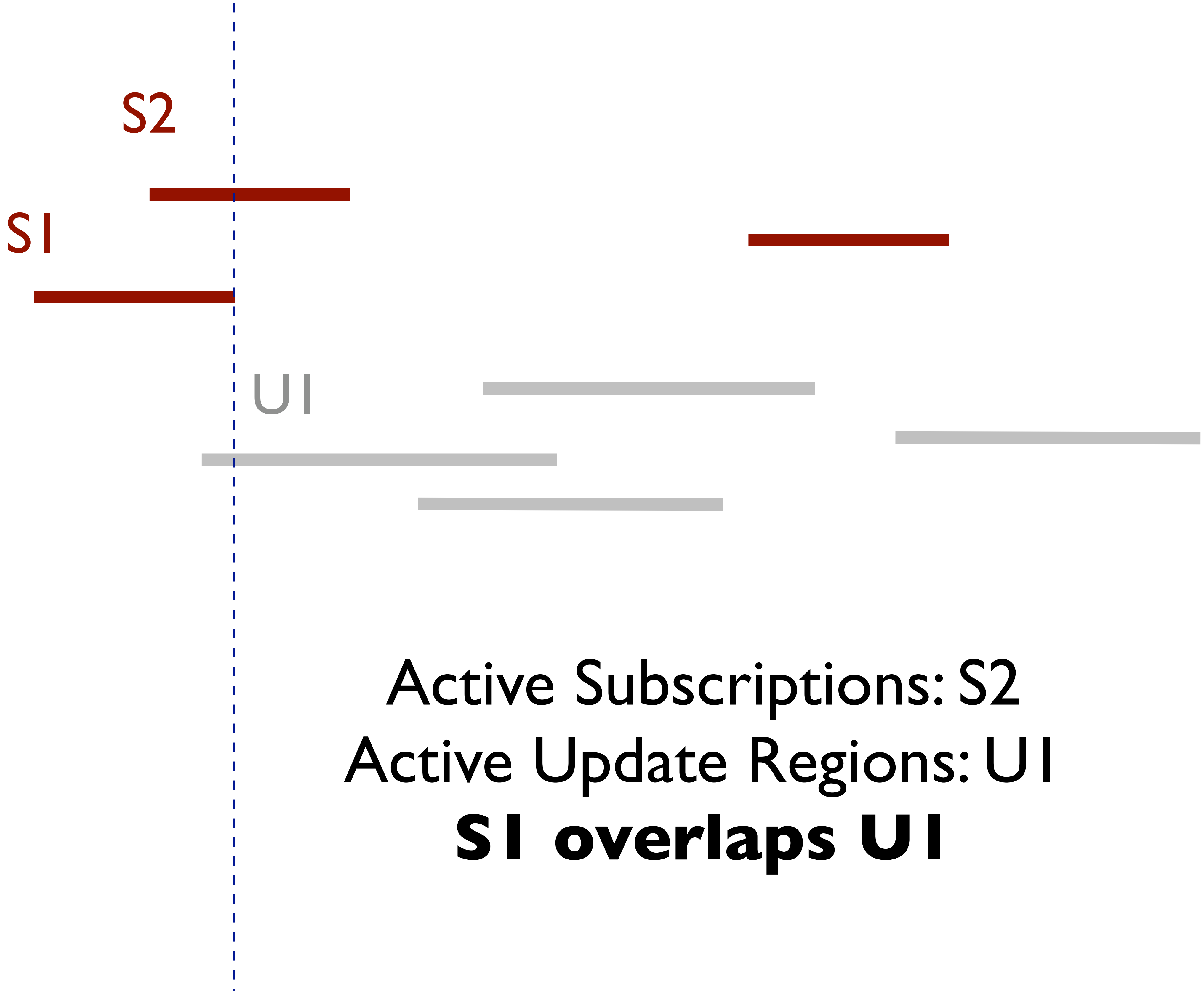


Active Subscriptions: S1, S2

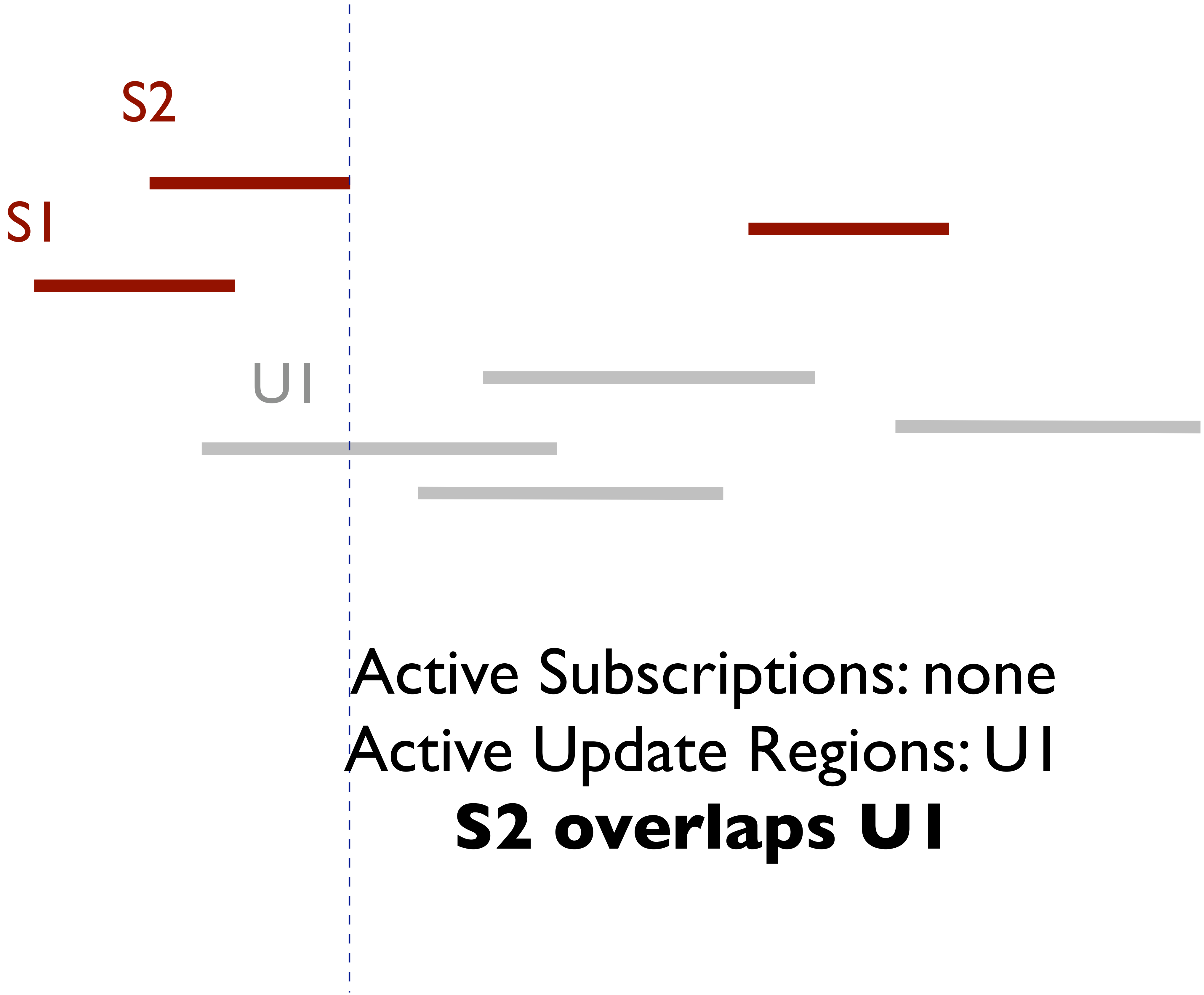


Active Subscriptions: S1, S2
Active Update Regions: U1

We can determine the overlaps when we process the endpoint of a region.



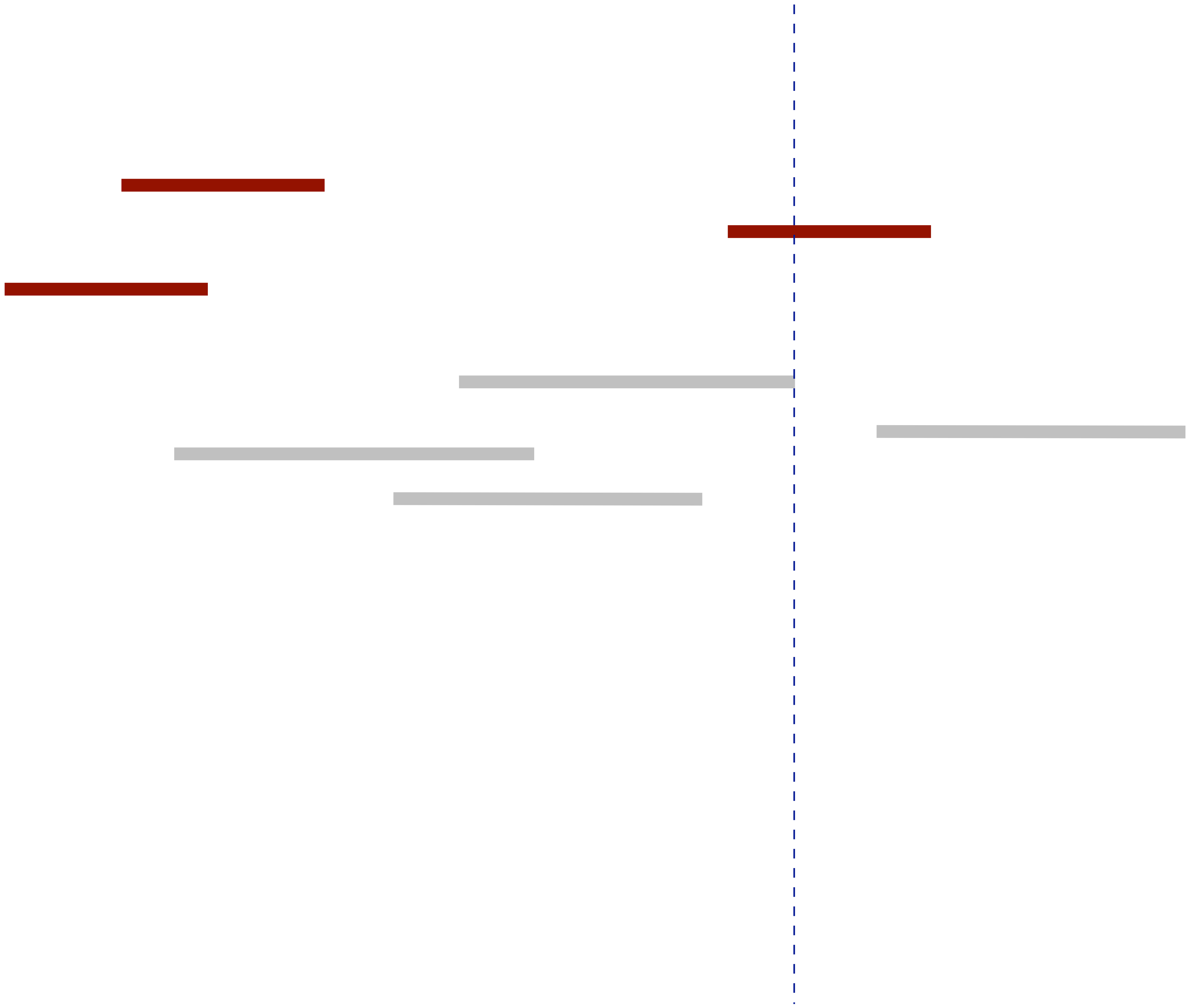
Active Subscriptions: S2
Active Update Regions: U1
S1 overlaps U1



Active Subscriptions: none

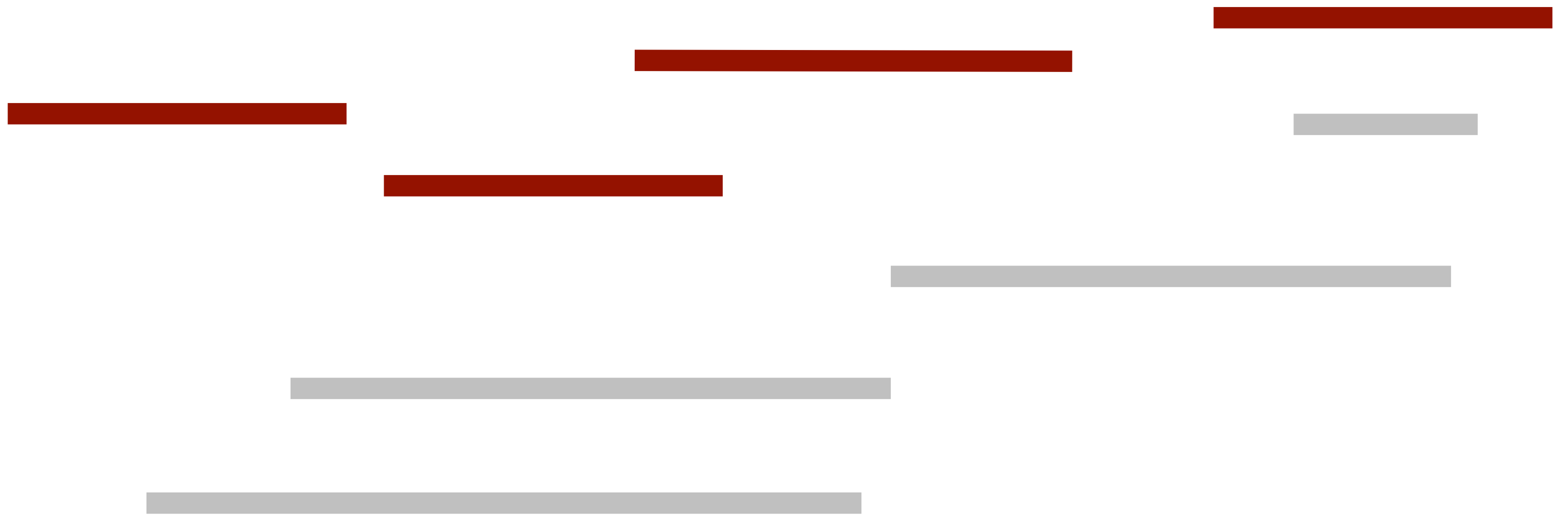
Active Update Regions: U1

S2 overlaps U1



If we encounter the endpoint of a subscription region, then it overlaps with all active update regions.

If it is the endpoint of an update region, then it overlaps with all active subscription region.



Exercise: trace through the small example and convince yourself that it works..

Sort-based approach:

O(n log n + m log m)
 for sorting

O(n + m)
 to scan

Note: storing overlap information still takes O(nm) since in the worst case there are O(nm) overlaps.

Temporal Coherence

Changes to value of an attribute is small between two consecutive time steps.

Sort-based approach:

O(n log n + m log m)

to pre-sort the data

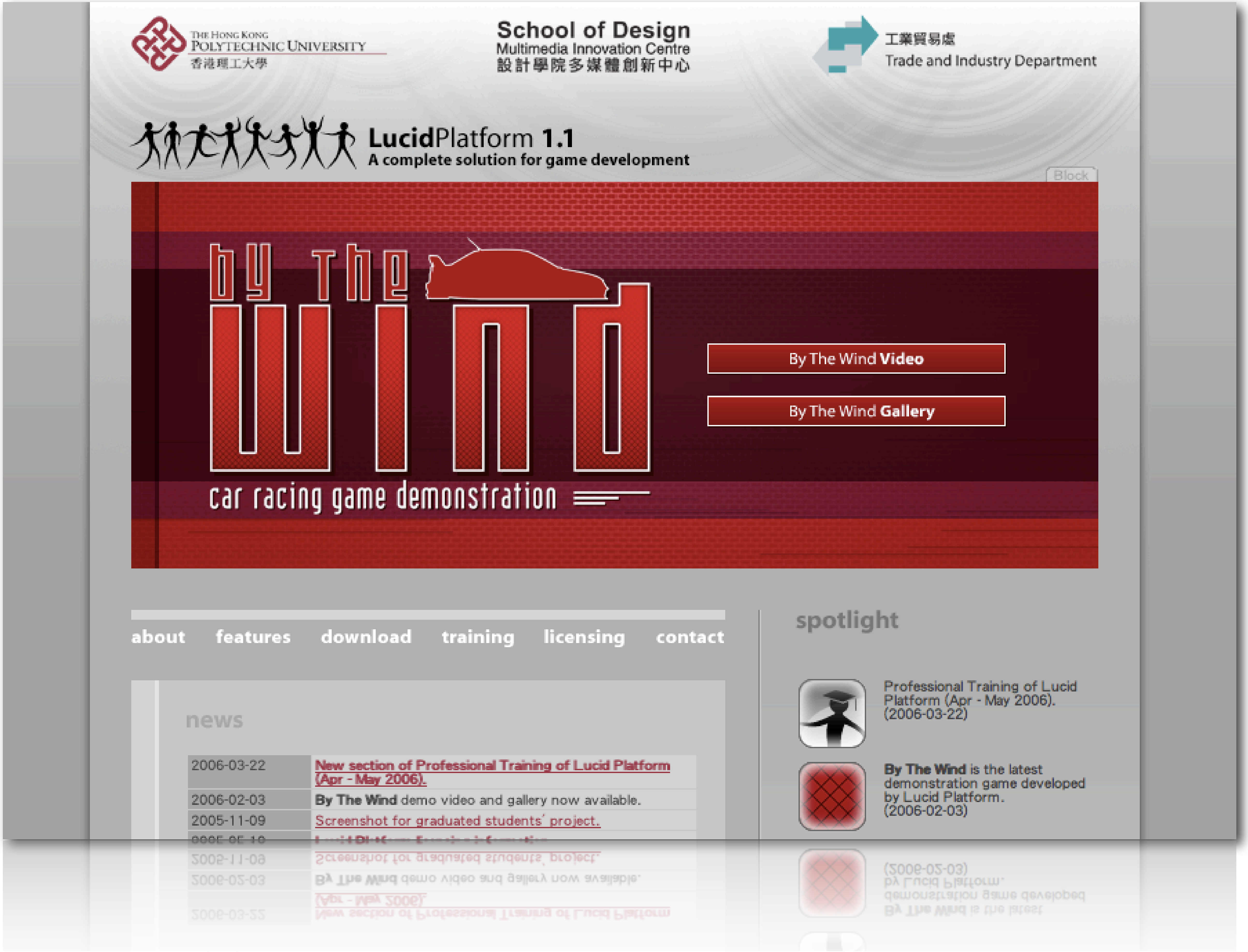
O(n + m)

for sorting (insertion sort)

O(n + m)

to scan

In fact, only regions which are swapped during insertion sort need to update their overlap set.



2006-03-22	New section of Professional Training of Lucid Platform (Apr - May 2006).
2006-02-03	**By The Wind** demo video and gallery now available.
2005-11-09	Screenshot for graduated students' project.

Professional Training of Lucid Platform (Apr - May 2006). (2006-03-22)

By The Wind is the latest demonstration game developed by Lucid Platform. (2006-02-03)