Lecture 7

Interest Management without Server

Point-to-Point Architecture

Problem: Communication between every pair of players

Idea: A player *p* only needs to communicate with another player *q* if *p* is relevant to *q*

Recall: In C/S Architecture, the server has global information and decides who is relevant to who.

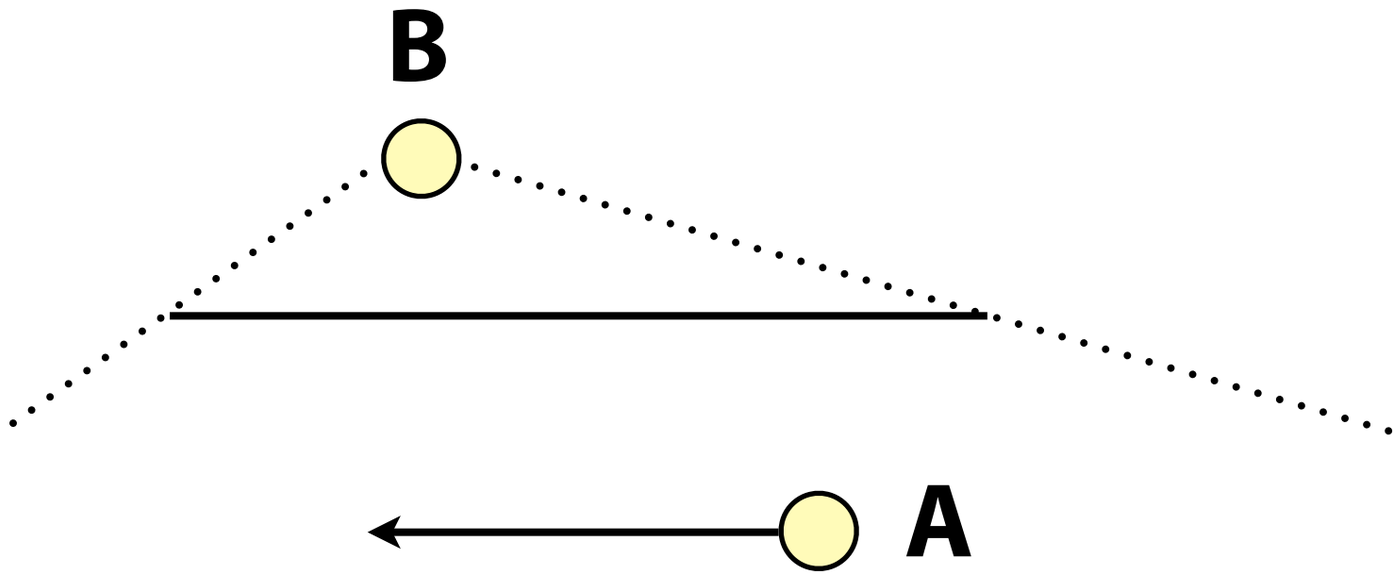
Challenge: No global information in P2P architecture.

Naive Solution: Every player keeps global information about all other player and makes individual decision.

Maintaining global information is expensive (and that's what we want to avoid in the first place!)

Smarter solution: exchange position, then decide when should the next position exchange be.

Idea: Assume *B* is static. If *A* knows *B*'s position, *A* can compute the region which is irrelevant to *B*. Need not update *B* if *A* moves within that region.



what if B moves?

It still works if *B* also knows *A* position and computes the region that is irrelevant to *A*.

Position exchanges occur once initially, and when a player moves outside of its irrelevant region wrt another player.

Frontier Sets

cell-based, visibility-based IM

Previously, we learnt how to compute cell-to-cell visibility.

Frontier for cells X and Y consists of two sets F_{XY} and F_{YX}

No cell in F_{XY} is visible from a cell in F_{YX}, and vice versa.

F_{XY} and F_{YX} are disjoint if X and Y are not mutually visible.

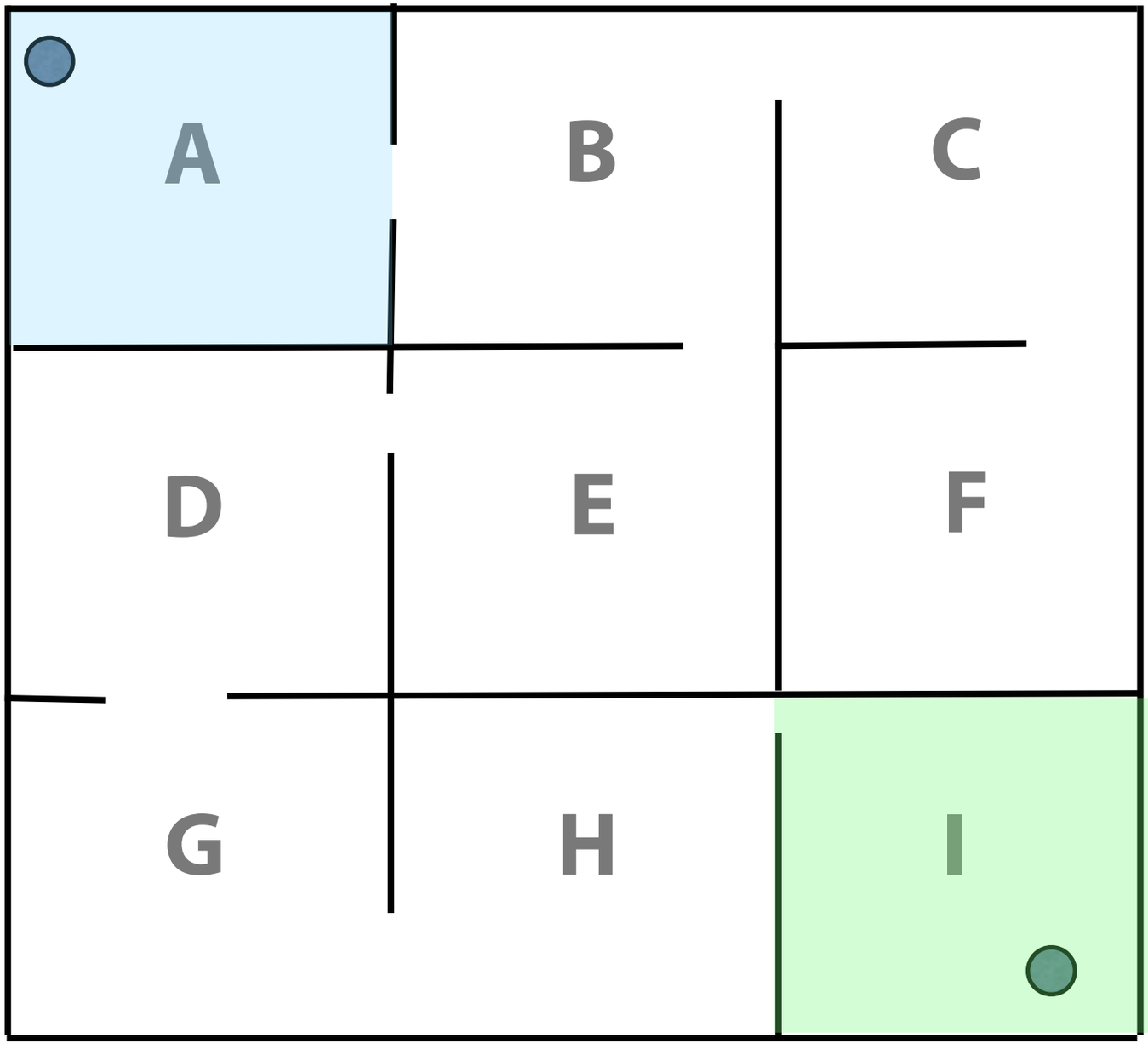
F_{XY} and F_{YX} are empty

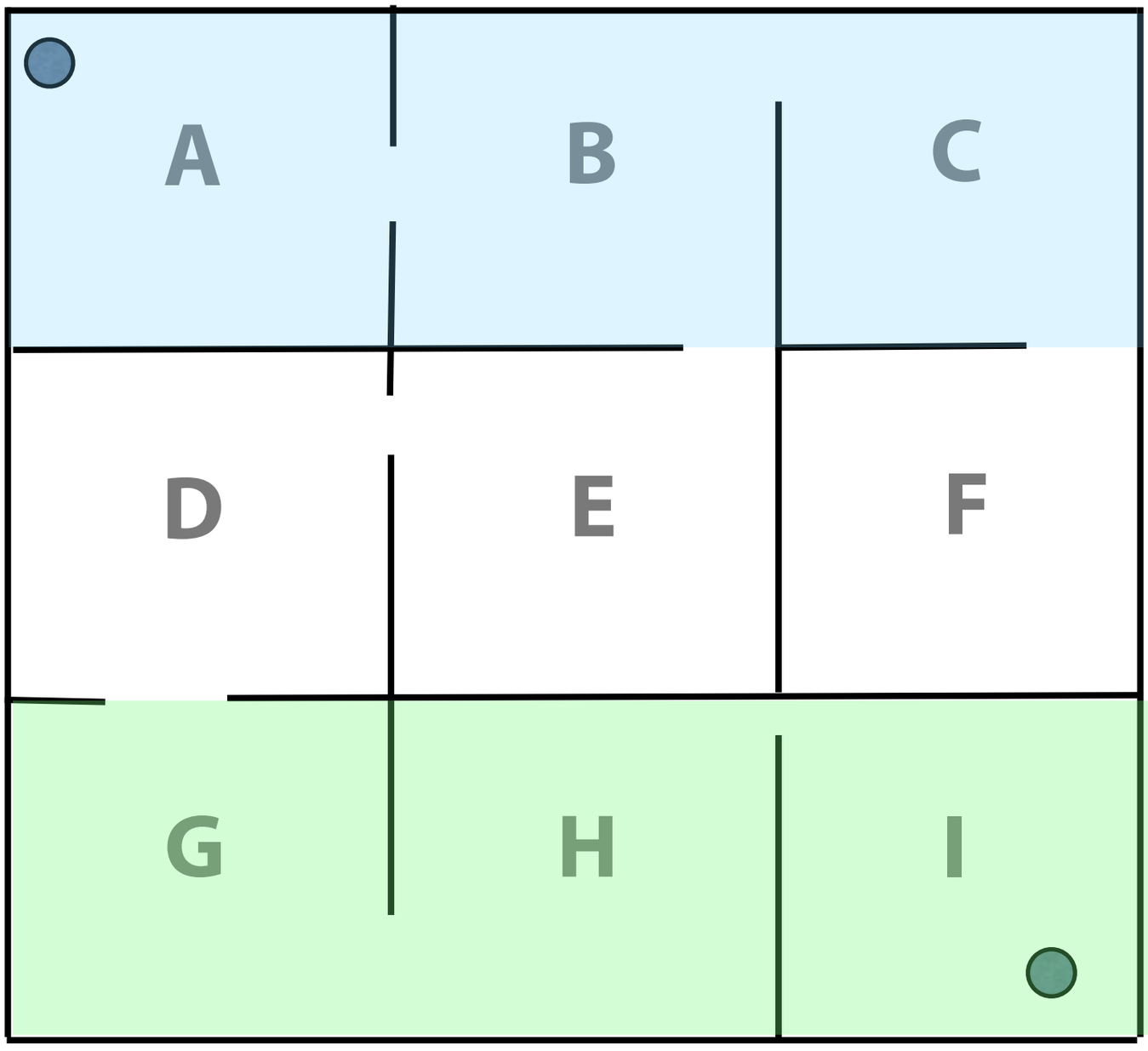
if X and Y are mutually visible.

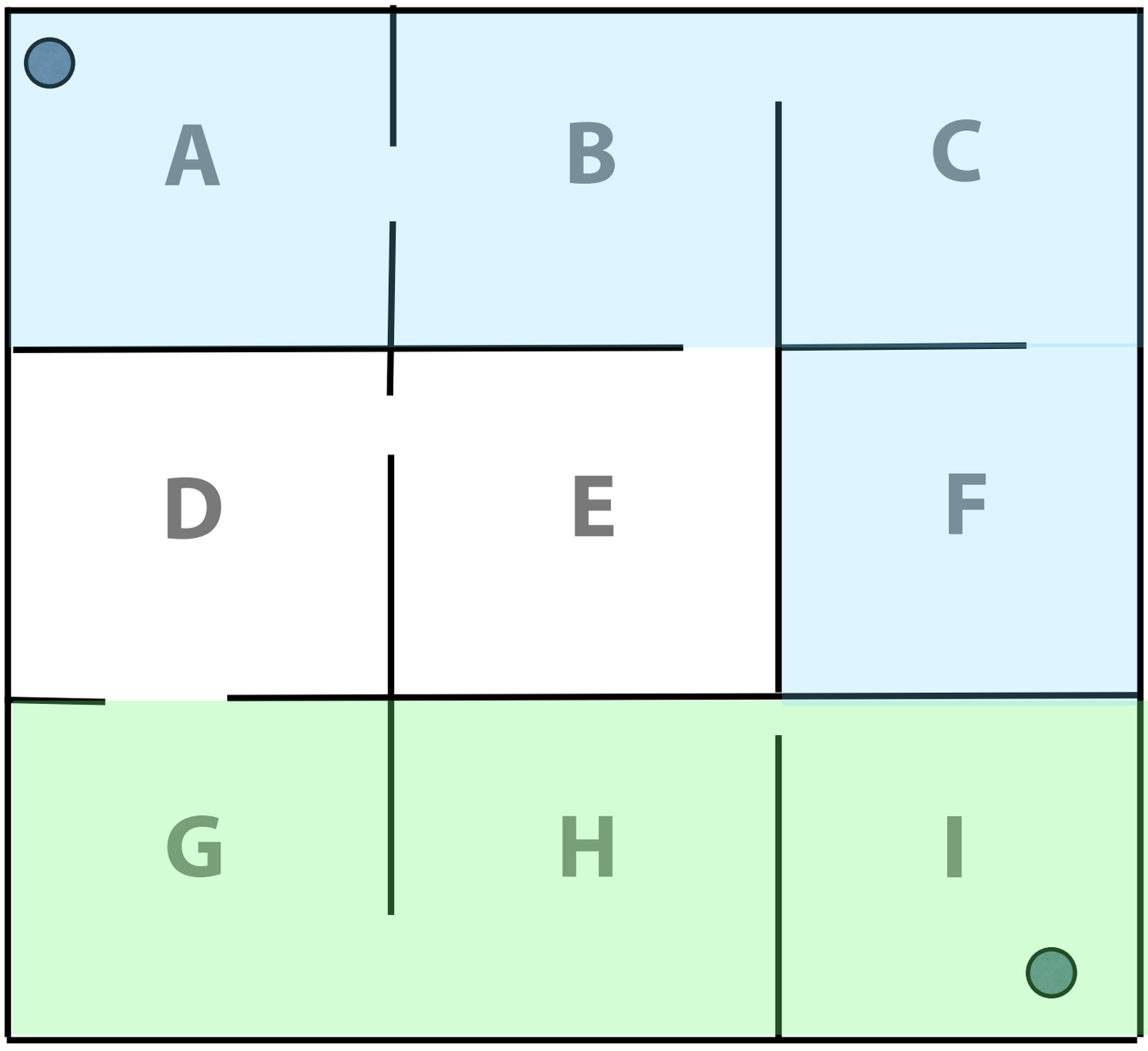
Suppose X and Y are not mutually visible, then a simple frontier is

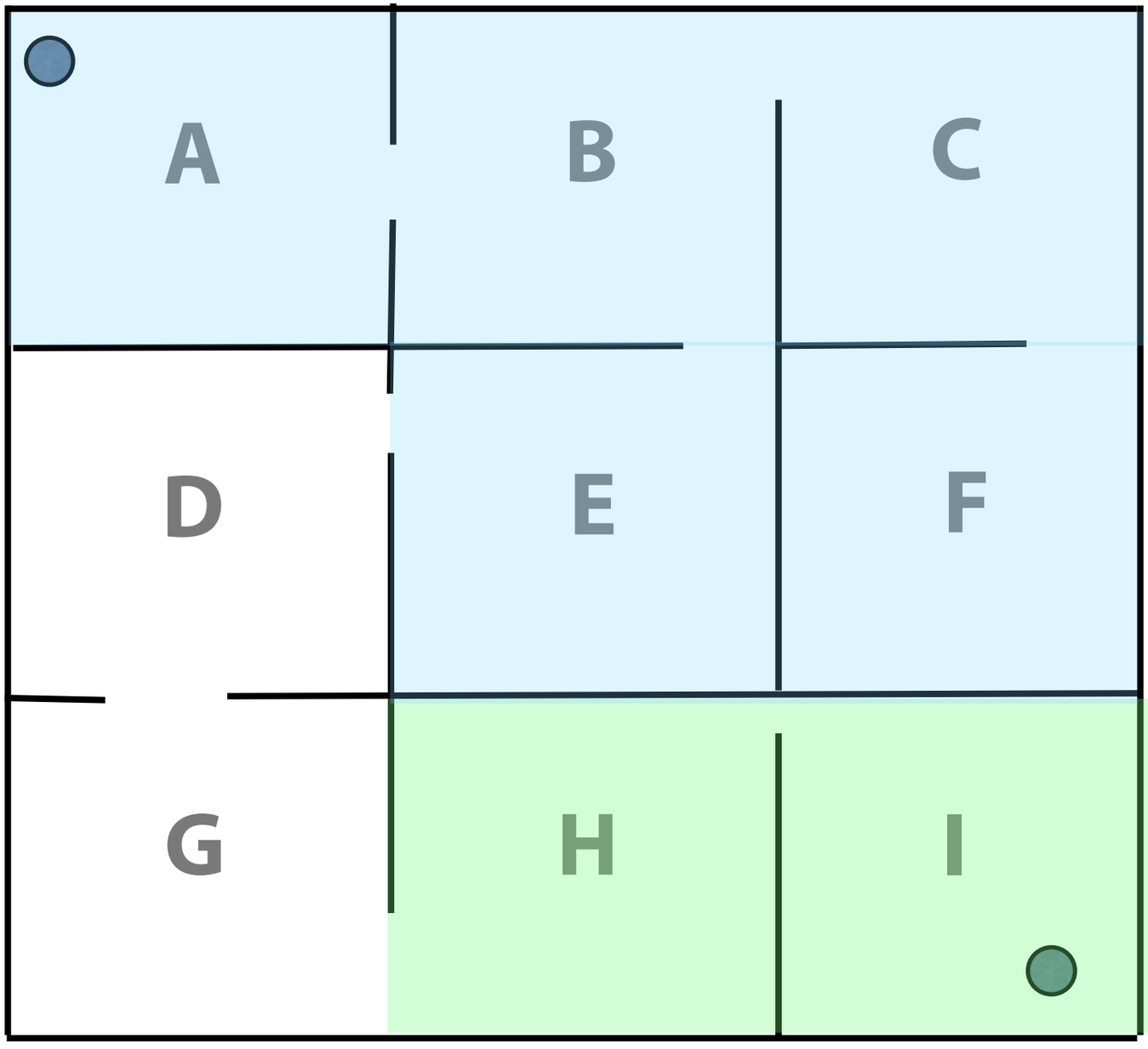
$$F_{XY} = \{X\} \quad F_{YX} = \{Y\}$$

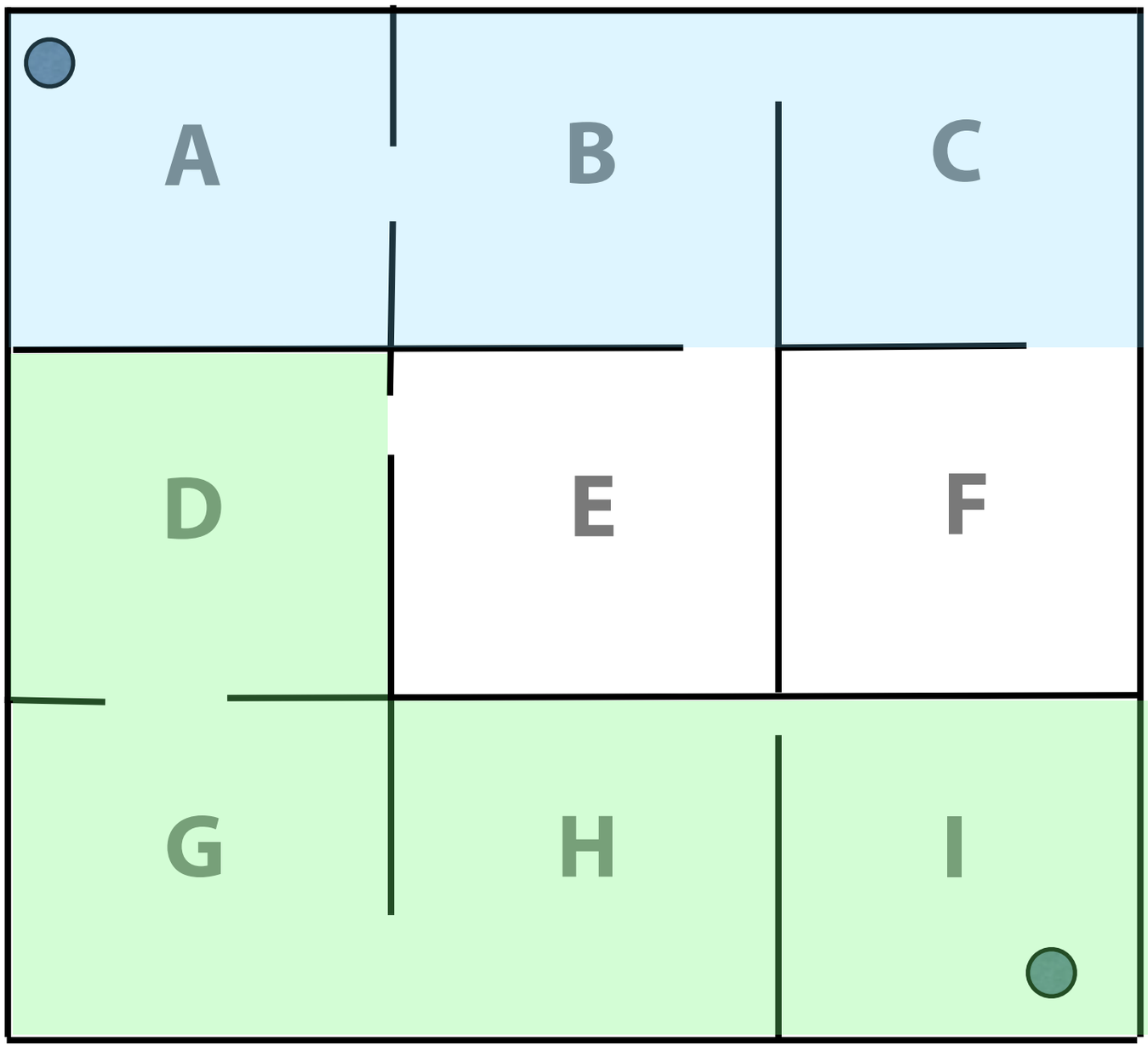
(many others are possible)


A
B
C
D
E
F
G
H
I


A
B
C
D
E
F
G
H
I


A
B
C
D
E
F
G
H
I


A
B
C
D
E
F
G
H
I


A
B
C
D
E
F
G
H
I

Position exchanges occur once initially, and when a player moves outside of its irrelevant region wrt another player.

Initialize:

Let player P be in cell X

For each player Q

Let cell of Q be Y

Compute F_{XY} (or simply F_Q)

Move to new cell:

Let X be new cell

For each player Q

If X not in F_Q

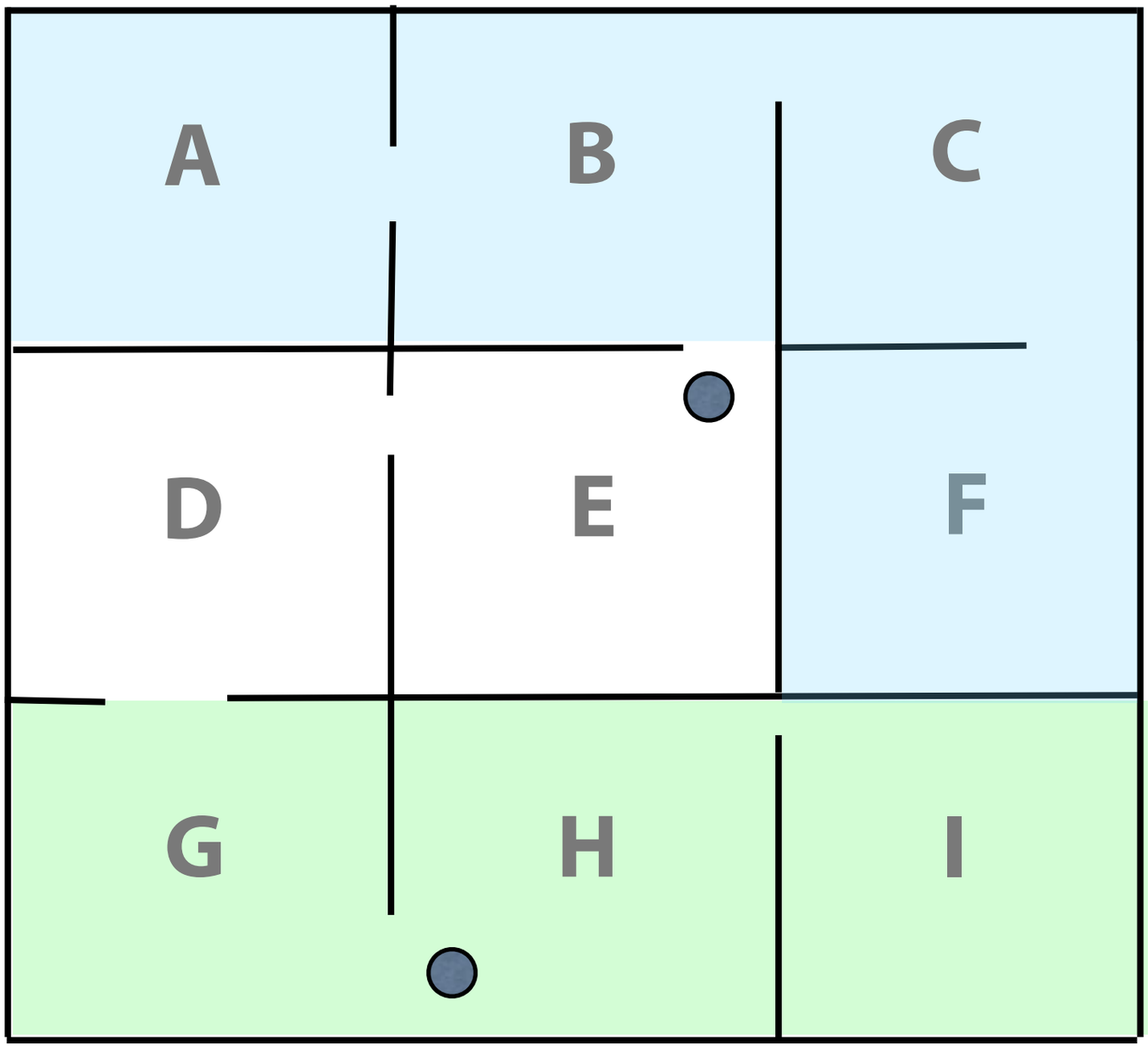
Send location to Q

Receive Update:
(location from Q)

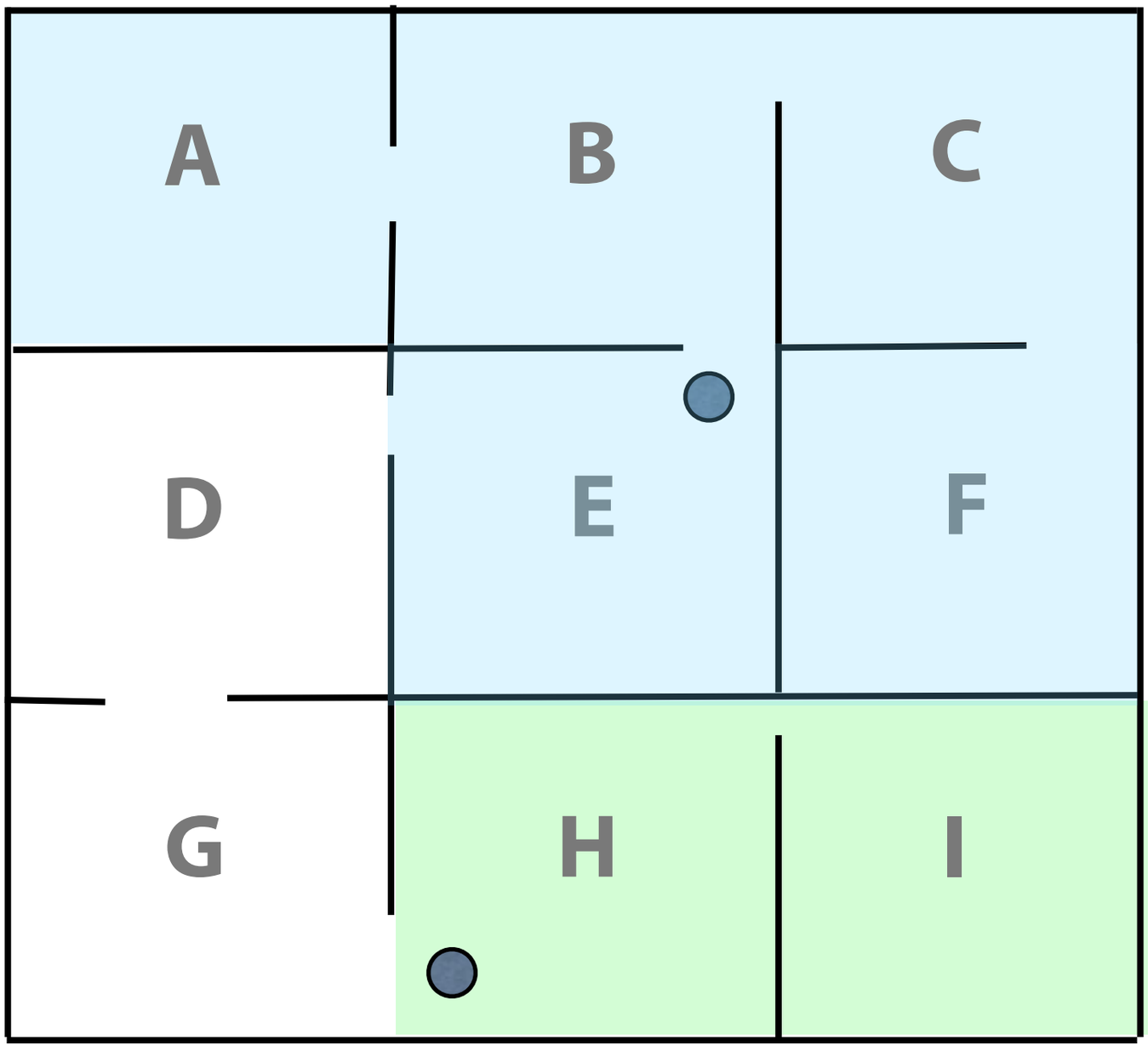
Send location to Q

Recompute F_Q

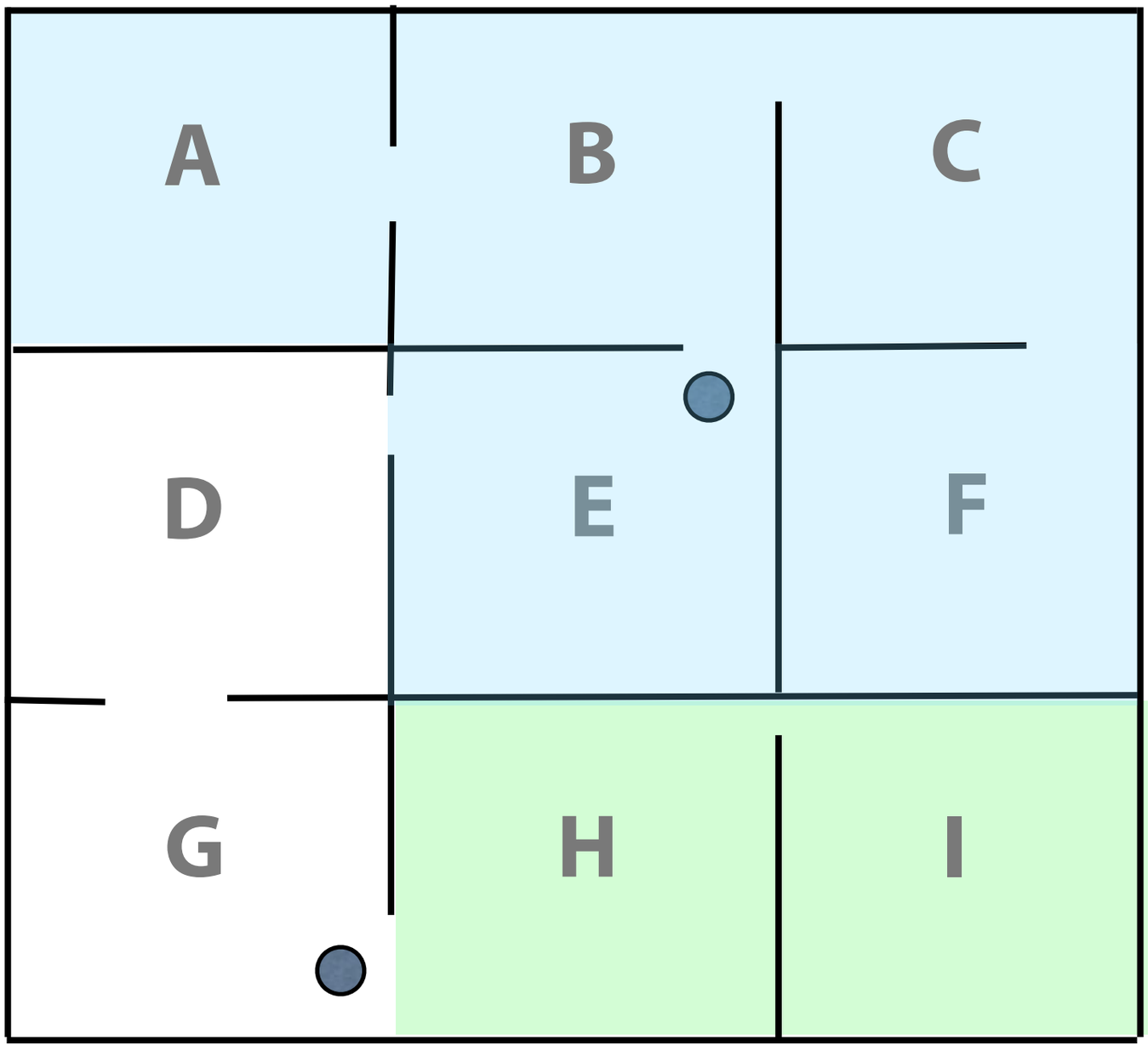
Update is triggered.



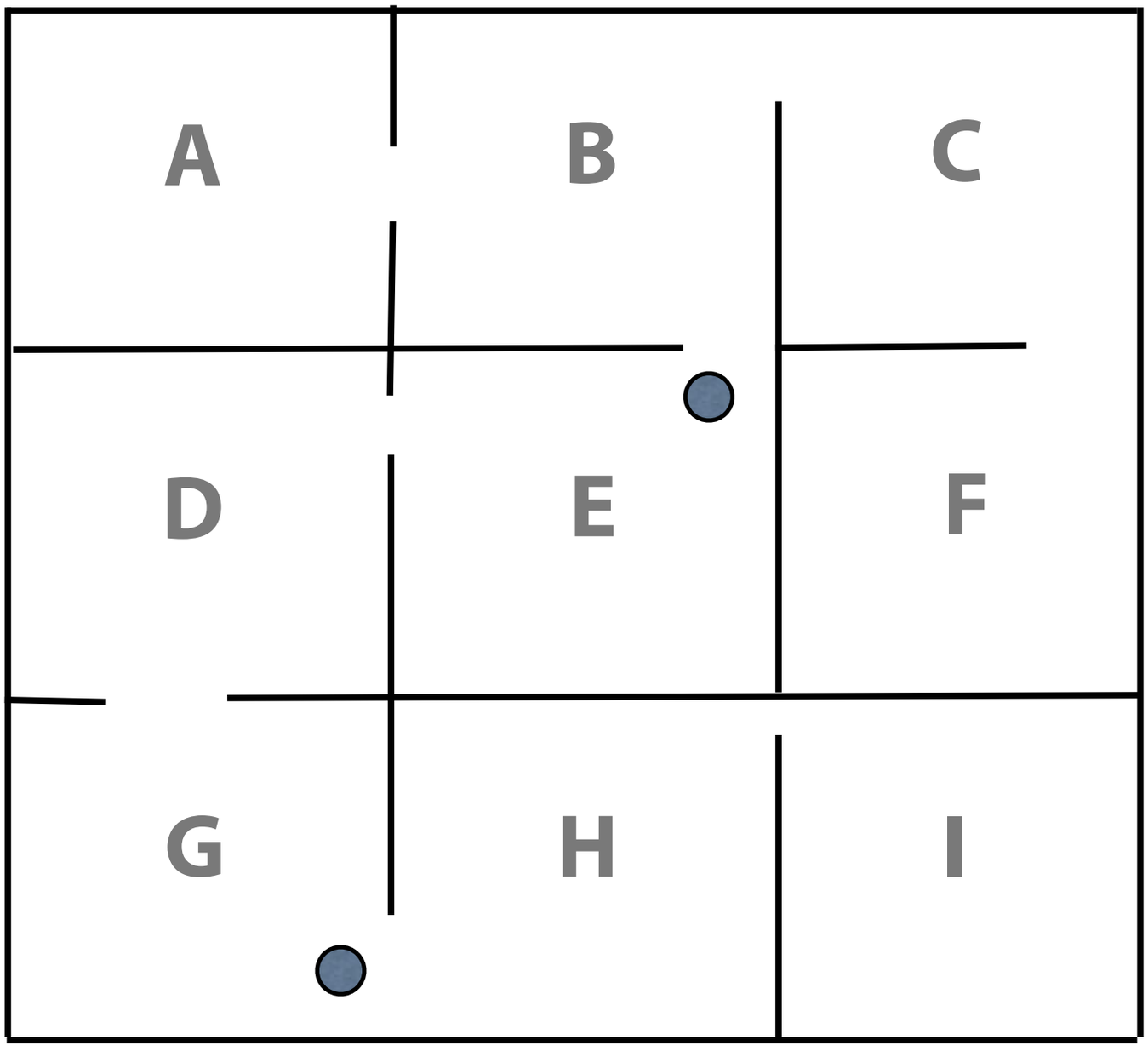
New Frontier.



Update triggered.



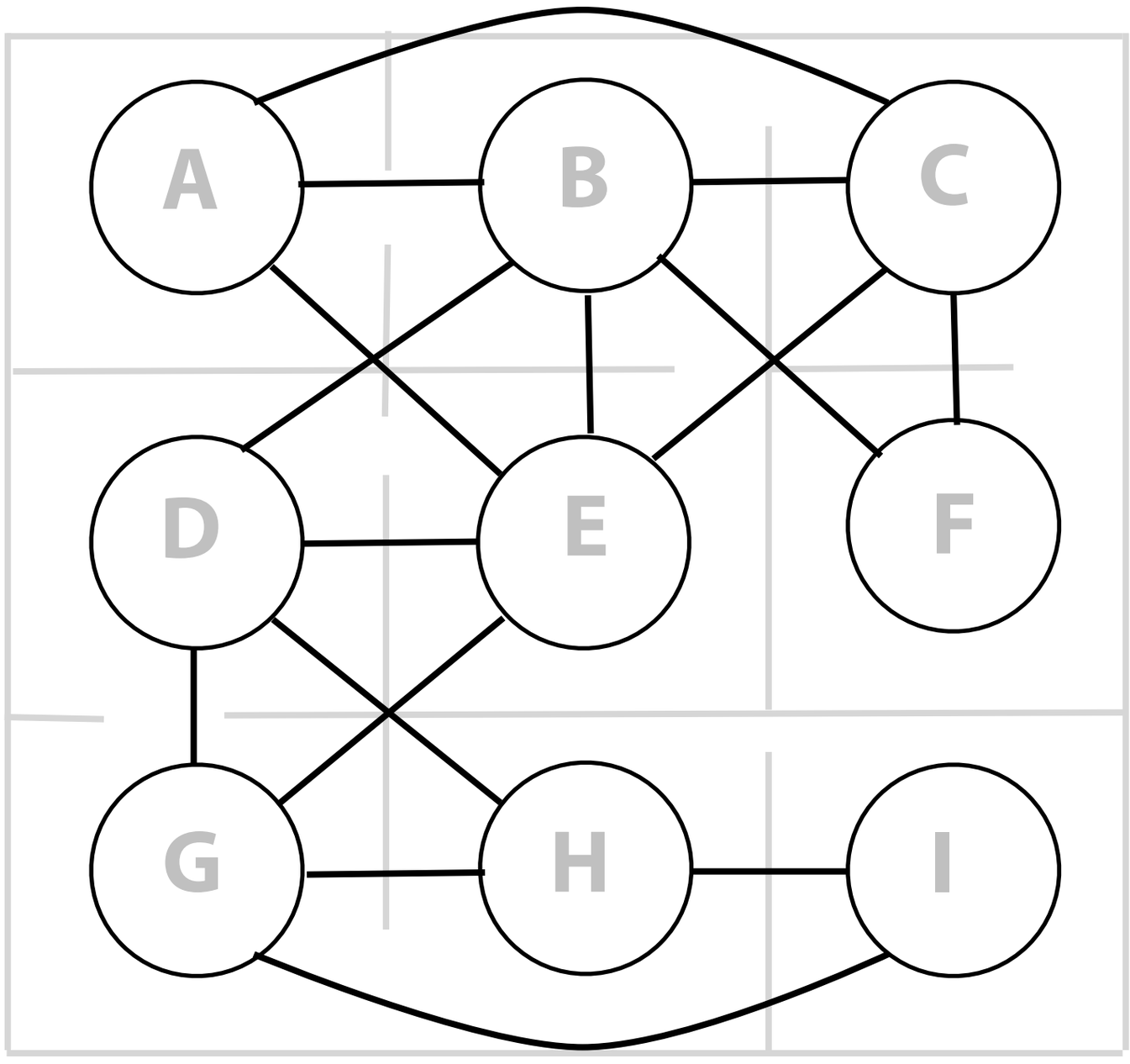
New frontier (empty since E can see G)



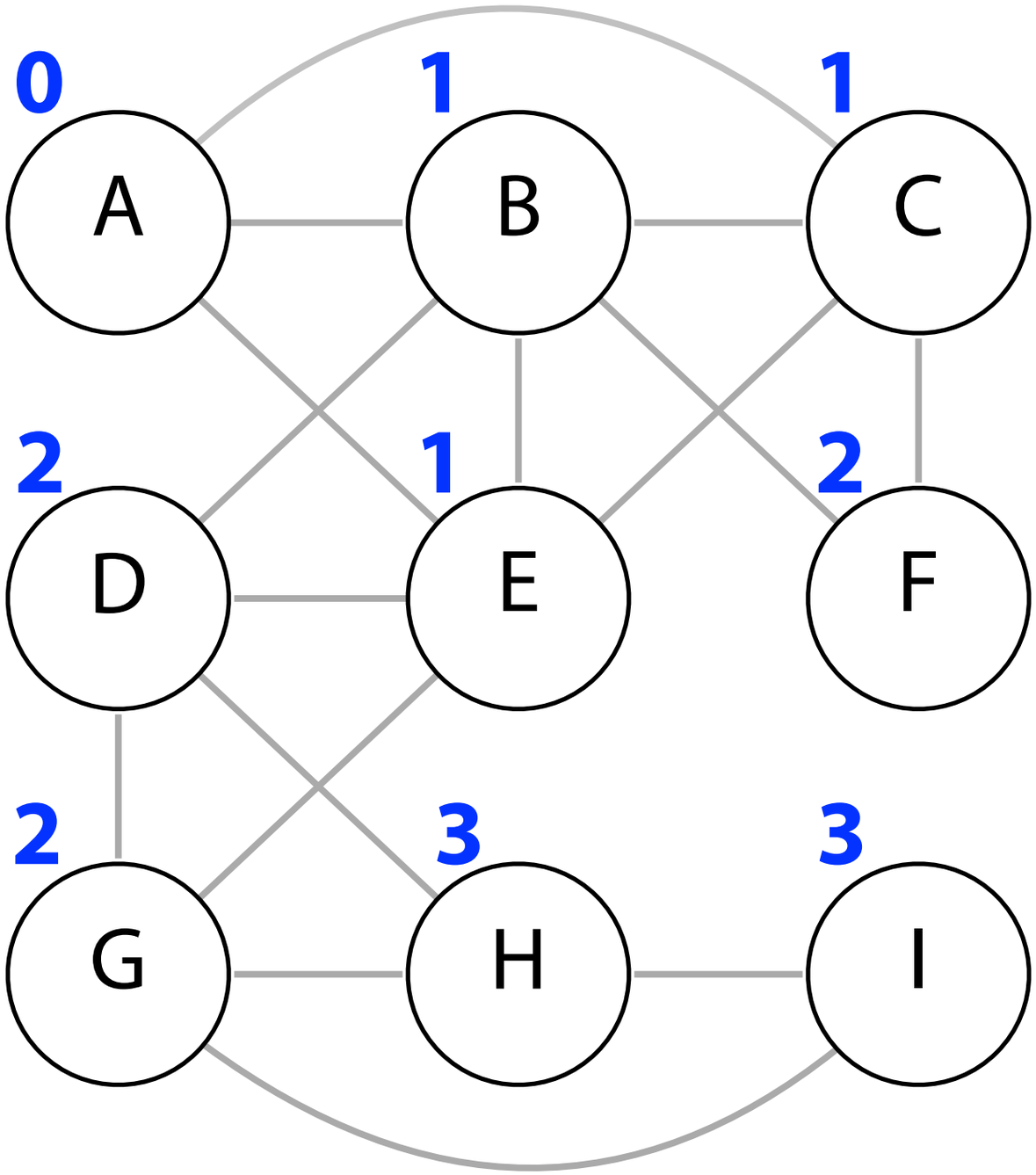
How to compute frontier?

A good frontier is as large as possible, with two almost equal-size sets.

Build a visibility graph. Cells are vertices. Two cells are connected by an edge if they are visible to each other (EVEN if they don't share a boundary)



Let dist(X,Y) be the shortest distance between two cells X and Y on the visibility graph.

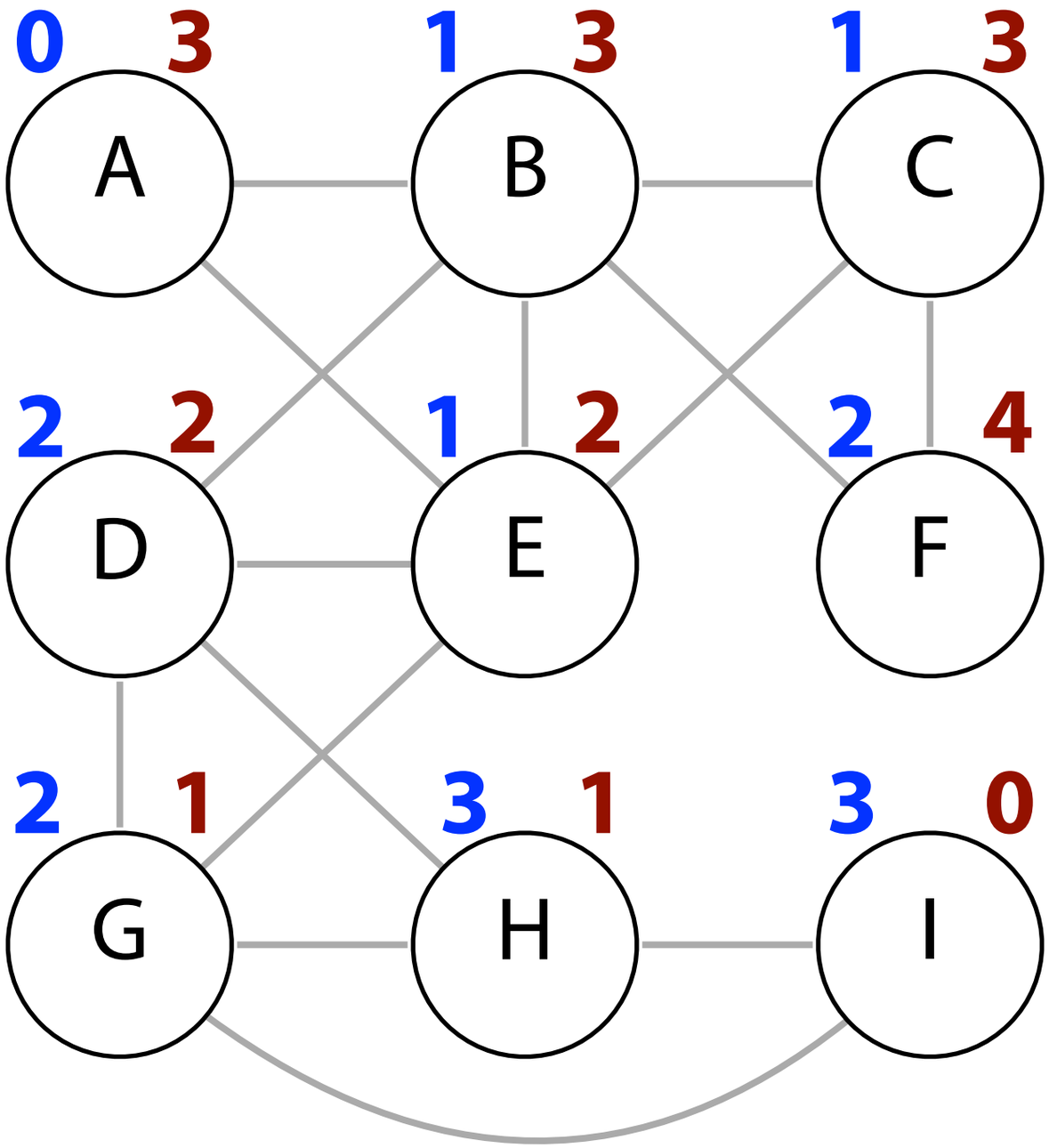

0
1
1
A
B
C
2
1
2
D
E
F
2
3
3
G
H
I

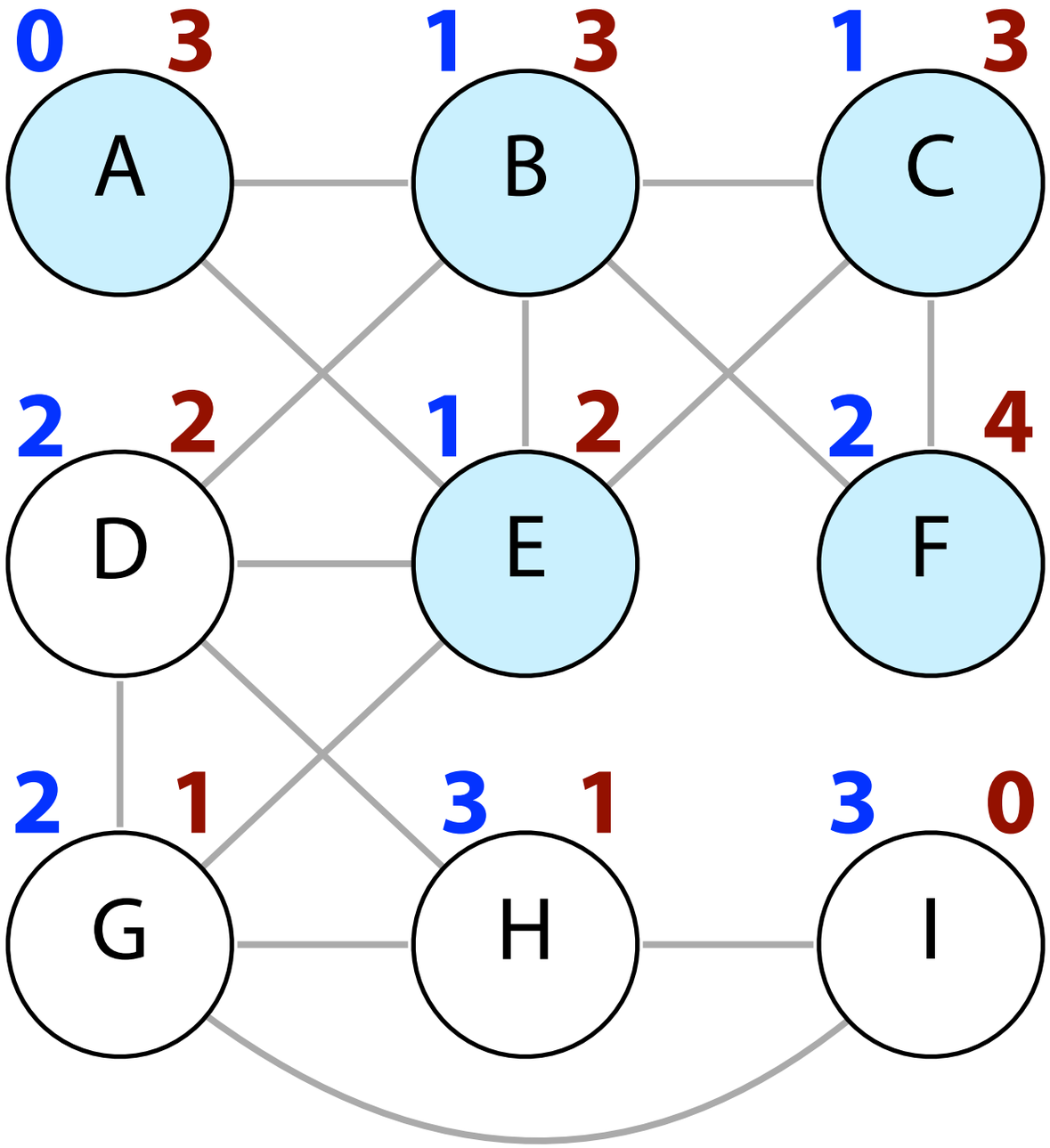
Theorem

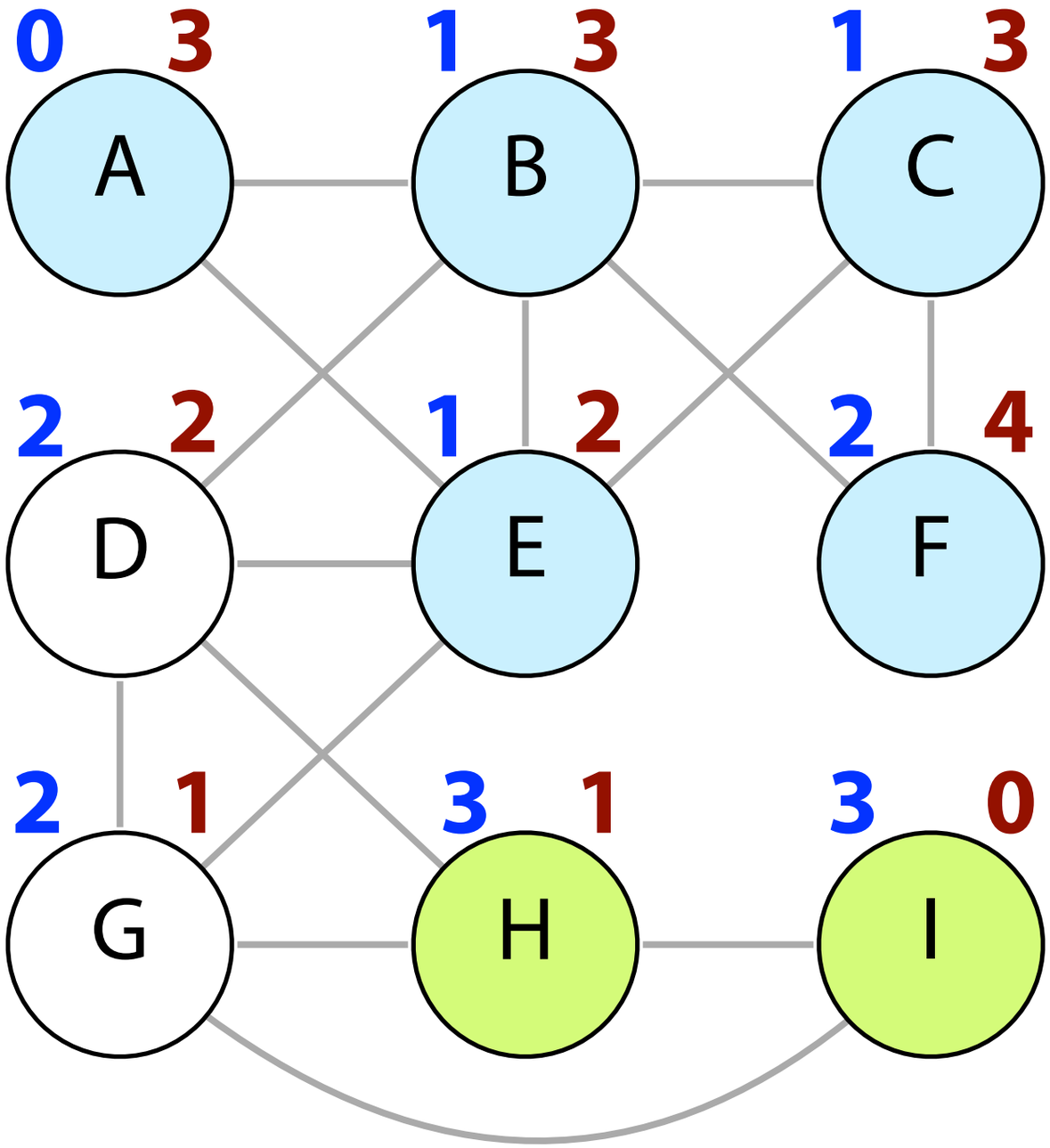
$F_{XY} = \{ i \mid dist(X,i) <= dist(Y,i) - 1\}$

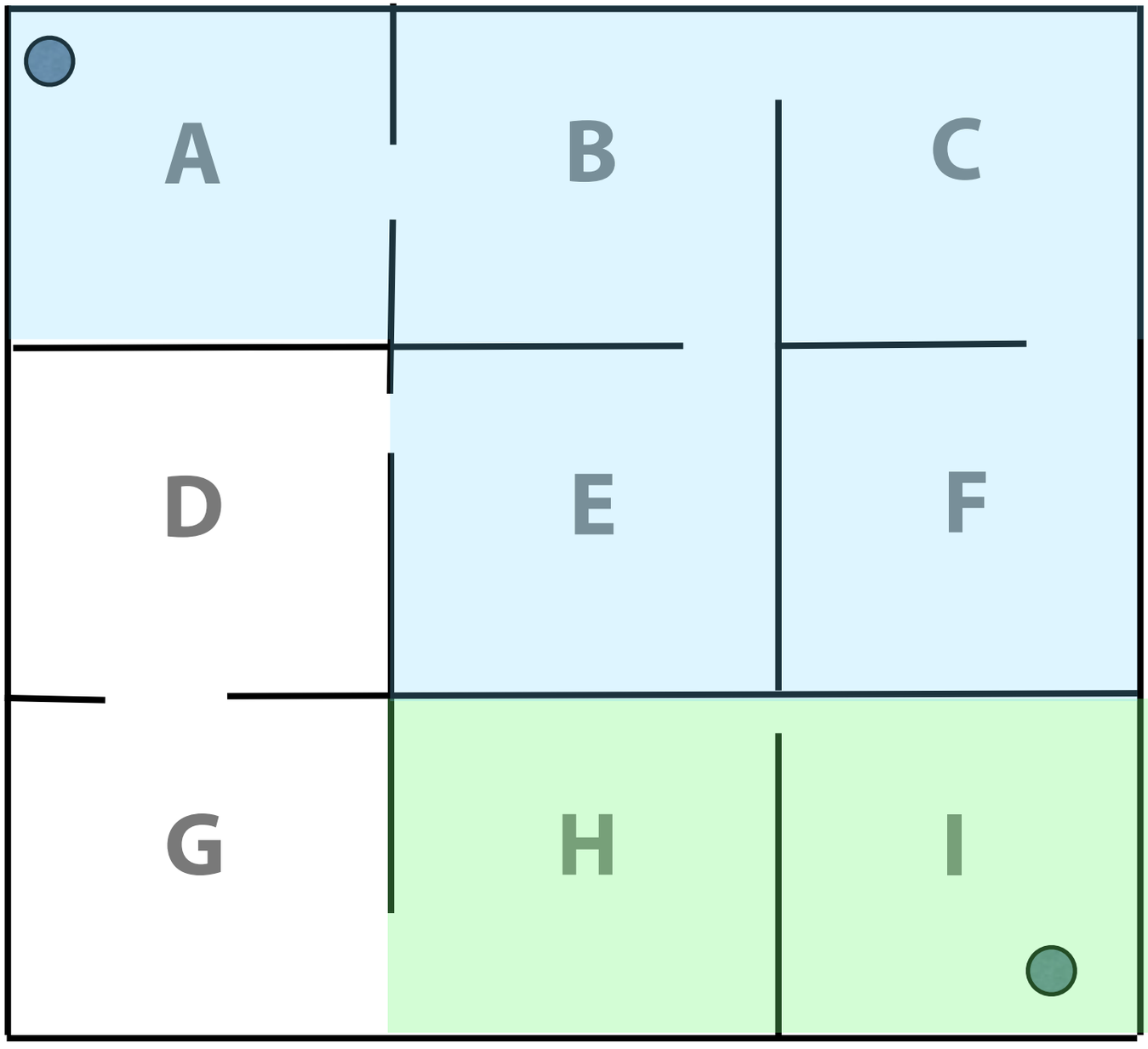
$F_{YX} = \{ j \mid dist(Y,j) < dist(X,j) - 1\}$

are valid frontiers.


0
3
1
3
1
3
A
B
C
2
2
1
2
2
4
D
E
F
2
1
3
1
3
0
G
H
I


0
3
1
3
1
3
A
B
C
2
2
1
2
2
4
D
E
F
2
1
3
1
3
0
G
H
I


0
3
1
3
1
3
A
B
C
2
2
1
2
2
4
D
E
F
2
1
3
1
3
0
G
H
I


A
B
C
D
E
F
G
H
I

Theorem

$F_{XY} = \{ i \mid dist(X,i) <= dist(Y,i) - 1\}$

$F_{YX} = \{ j \mid dist(Y,j) < dist(X,j) - 1\}$

are valid frontiers.

$$F_{XY} = \{ i \mid dist(X,i) <= dist(Y,i) - 1 \}$$
$$F_{YX} = \{ j \mid dist(Y,j) < dist(X,j) - 1 \}$$

Proof (by contradiction)

Suppose there are two cells, C in F_{XY} and D in F_{YX}, that can see each other.

$$F_{XY} = \{ i \mid dist(X,i) <= dist(Y,i) - 1 \}$$

$$F_{YX} = \{ j \mid dist(Y,j) < dist(X,j) - 1 \}$$

$$dist(X,C) <= dist(Y,C) - 1$$

$$dist(Y,D) < dist(X,D) - 1$$

$$dist(C,D) = dist(D,C) = 1$$

$$dist(X,C) <= dist(Y,C) - 1$$
$$dist(Y,D) < dist(X,D) - 1$$
$$dist(C,D) = dist(D,C) = 1$$

We also know that

$$dist(X,D) <= dist(X,C) + dist(C,D)$$
$$dist(Y,C) <= dist(Y,D) + dist(D,C)$$

1. dist(X,C) <= dist(Y,C) - 1
2. dist(Y,D) < dist(X,D) - 1
3. dist(C,D) = 1
4. dist(X,D) <= dist(X,C) + dist(C,D)
5. dist(Y,C) <= dist(Y,D) + dist(D,C)

From 4, 1, and 3:

dist(X,D) <= dist(Y,C) - 1 + 1

From 5:

dist(X,D) <= dist(Y,D) + 1

1. dist(X,C) <= dist(Y,C) - 1
2. dist(Y,D) < dist(X,D) - 1
3. dist(C,D) = 1
4. dist(X,D) <= dist(X,C) + dist(C,D)
5. dist(Y,C) <= dist(Y,D) + dist(D,C)

We have

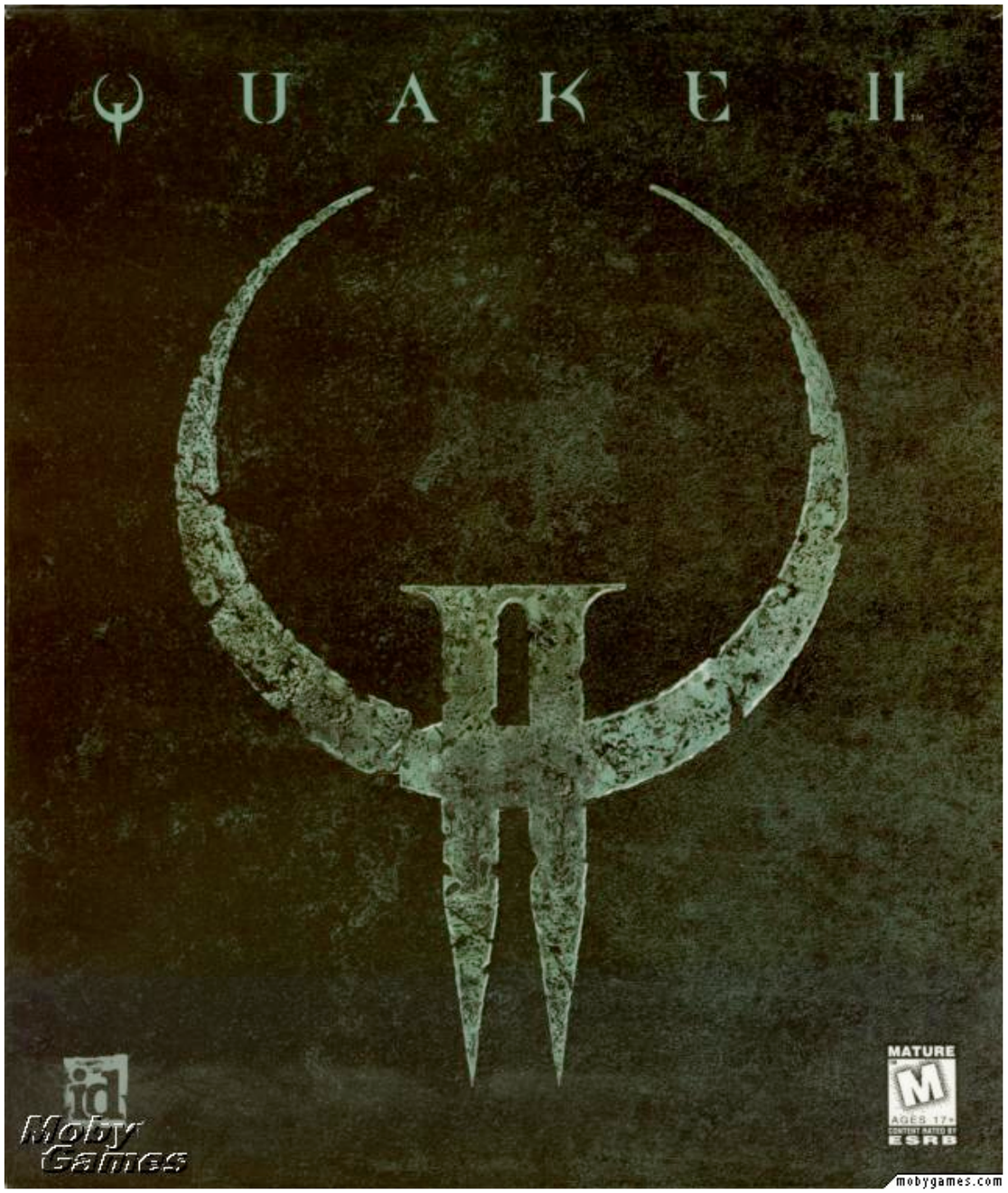
dist(X,D) <= dist(Y,D) + 1

Which contradict 2

dist(X,D) > dist(Y,D) + 1

How good is the idea?

(How many messages can we save by using Frontier Sets?)



https://github.com/id-Software/Quake-2

	q2dm3	q2dm4	q2dm8
Max dist()	4	5	8
Num of cells	666	1902	966

Frontier Density:
% of player-pairs with non-empty frontiers.

	q2dm3	q2dm4	q2dm8
Frontier Density	83.9	93	84.2

Frontier Size:

% of cells in the frontier on average

	q2dm3	q2dm4	q2dm8
Frontier Size	38.3%	67.3%	68.2%

Compare with

1. Naive P2P

2. Perfect P2P

Naive P2P

Always send update to 15 other players.

Perfect P2P

Hypothetical protocol that sends messages only to visible players.

Number of messages per frame per player.

	q2dm3	q2dm4	q2dm8
NPP	15	15.7	14.4
PPP	3.7	1.9	4.2
Frontier	5.4	2.6	5.9

Space Complexity

Let N be the number of cells. If we precompute Frontier for every pair of cells, we need

$$O(N^3)$$

space.

If we store visibility graph and compute frontier as needed, we only need

$O(N^2)$

space.

Frontier Sets

cell-based, visibility-based IM

Limitations

Works badly if there's little occlusion in the virtual world.

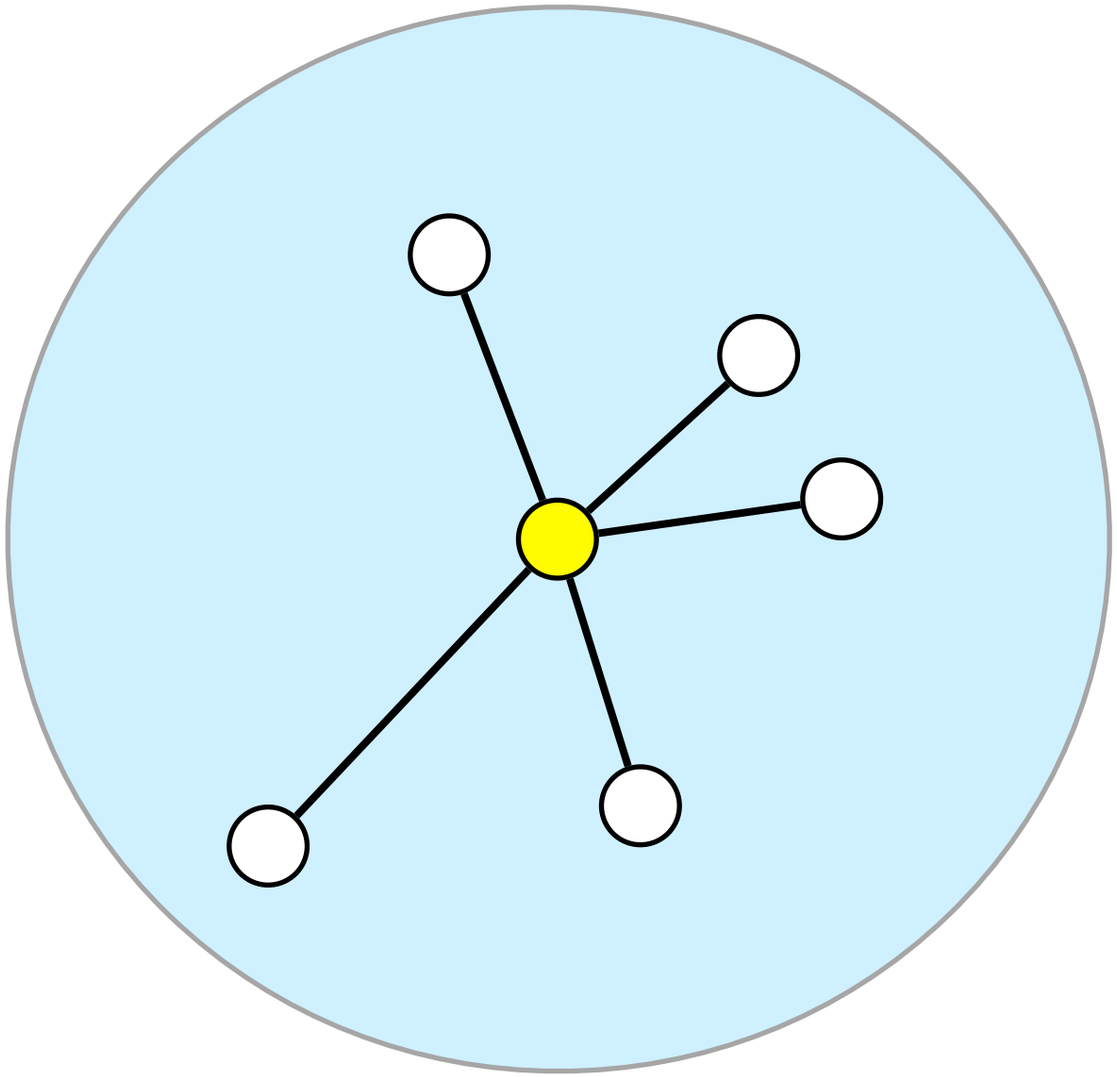
Still need to exchange locations with every other players occasionally.

Voronoi Overlay Network: Distance-based Interest Management

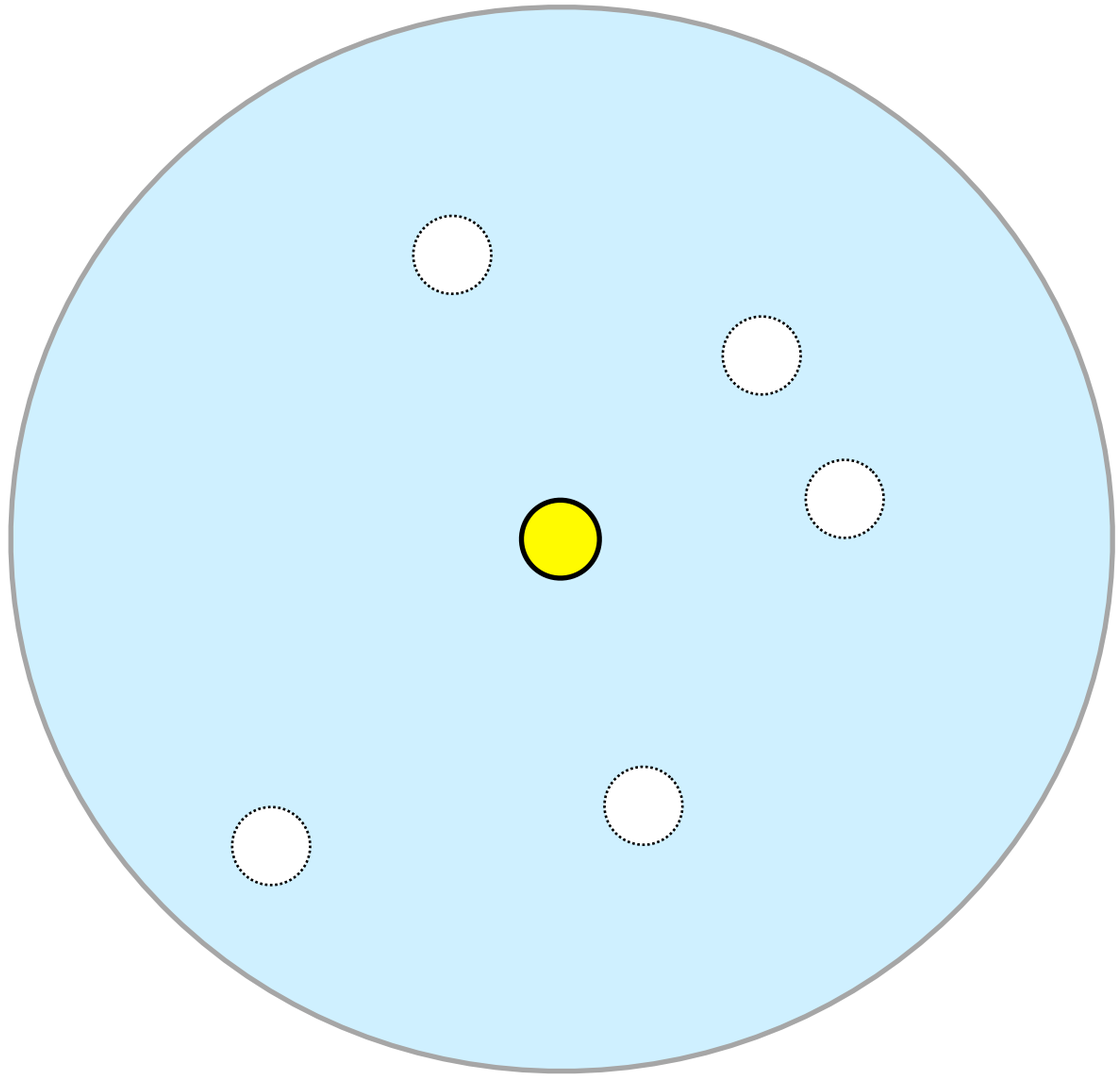
Diagrams and plots in the sections are taken from presentation slides by

Shun-yun Hu, available on http://vast.sf.net

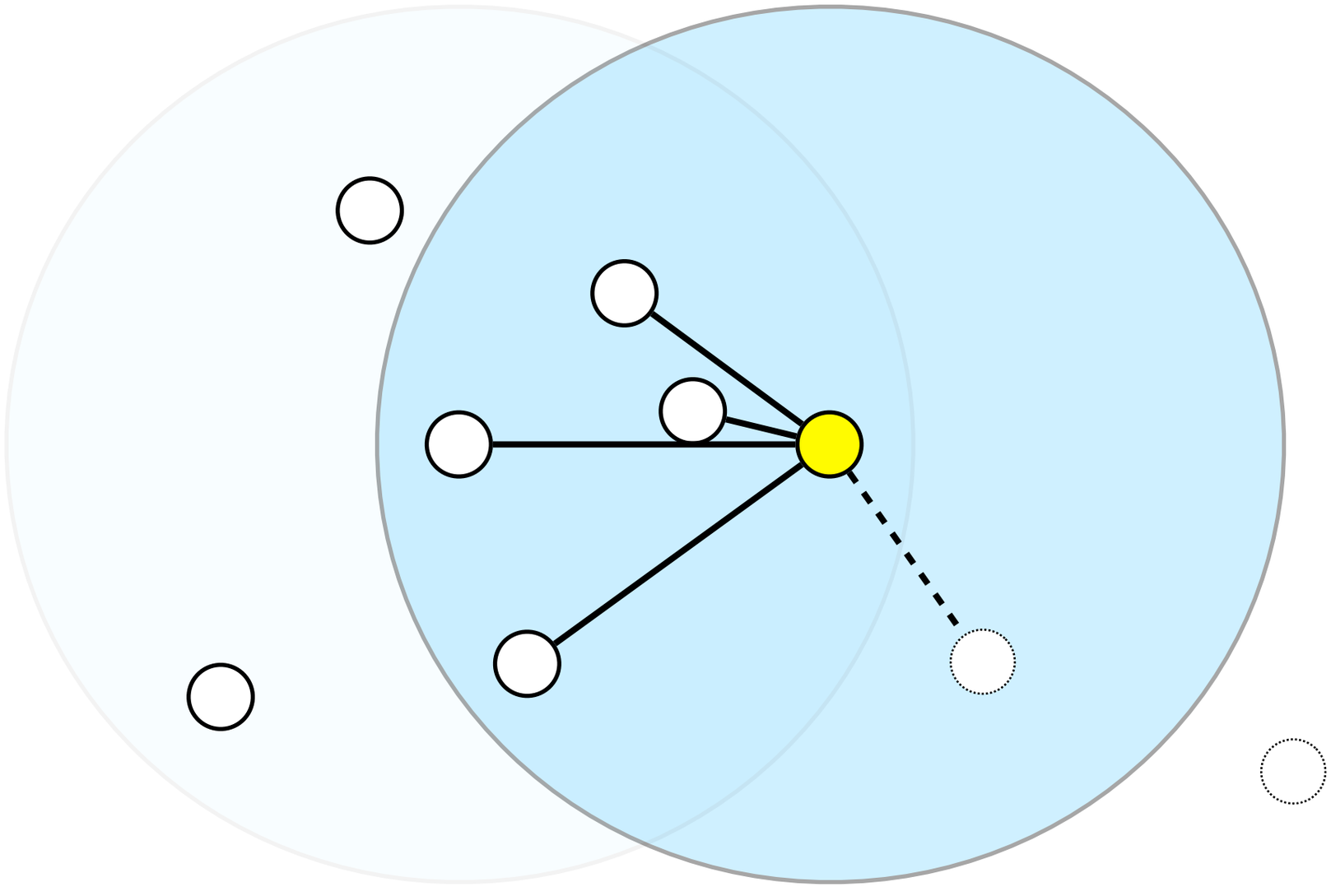
Keep a list of AOI-neighbors and exchange messages with AOI-neighbors.

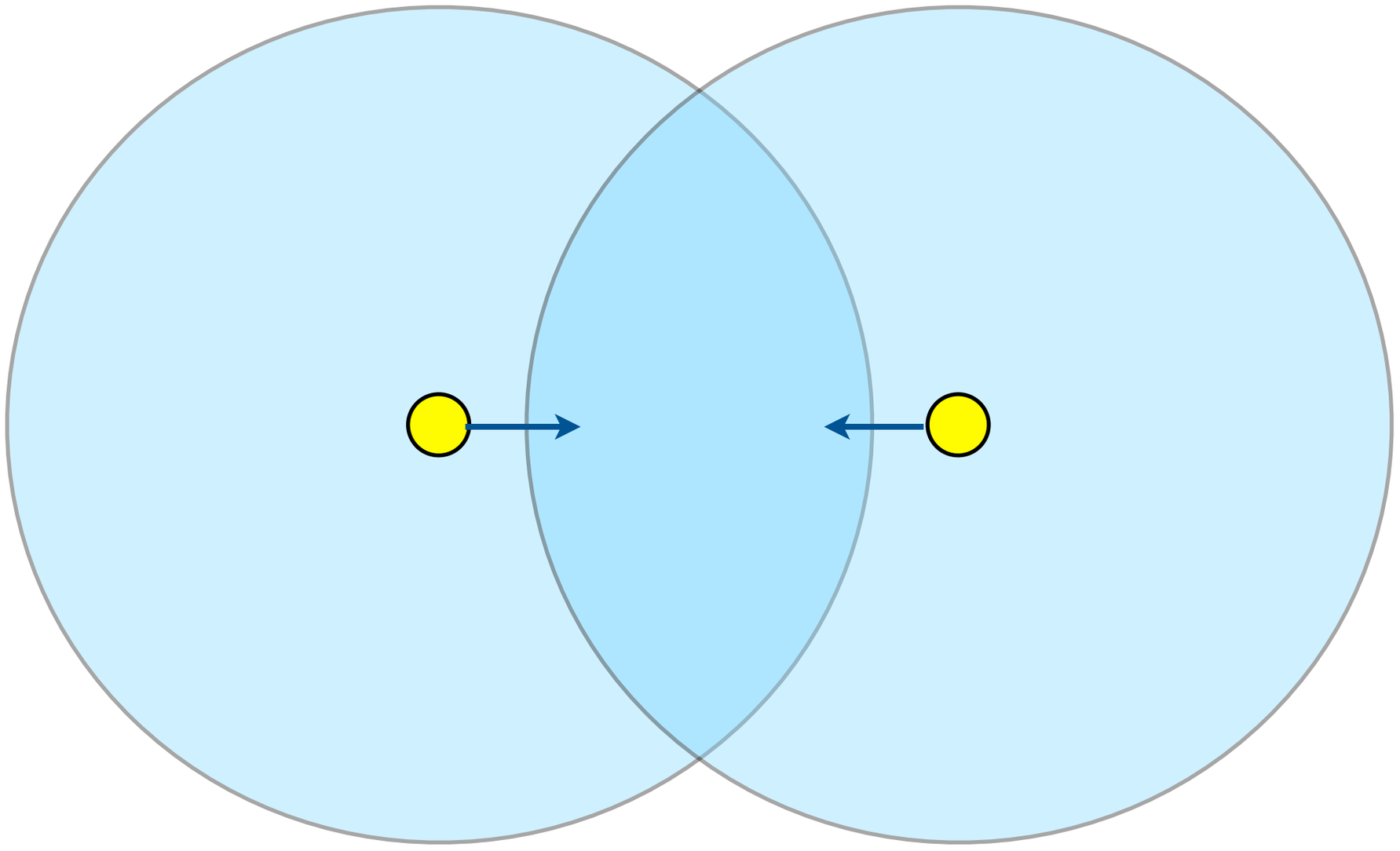


Q: How to initialize AOI-neighbors?

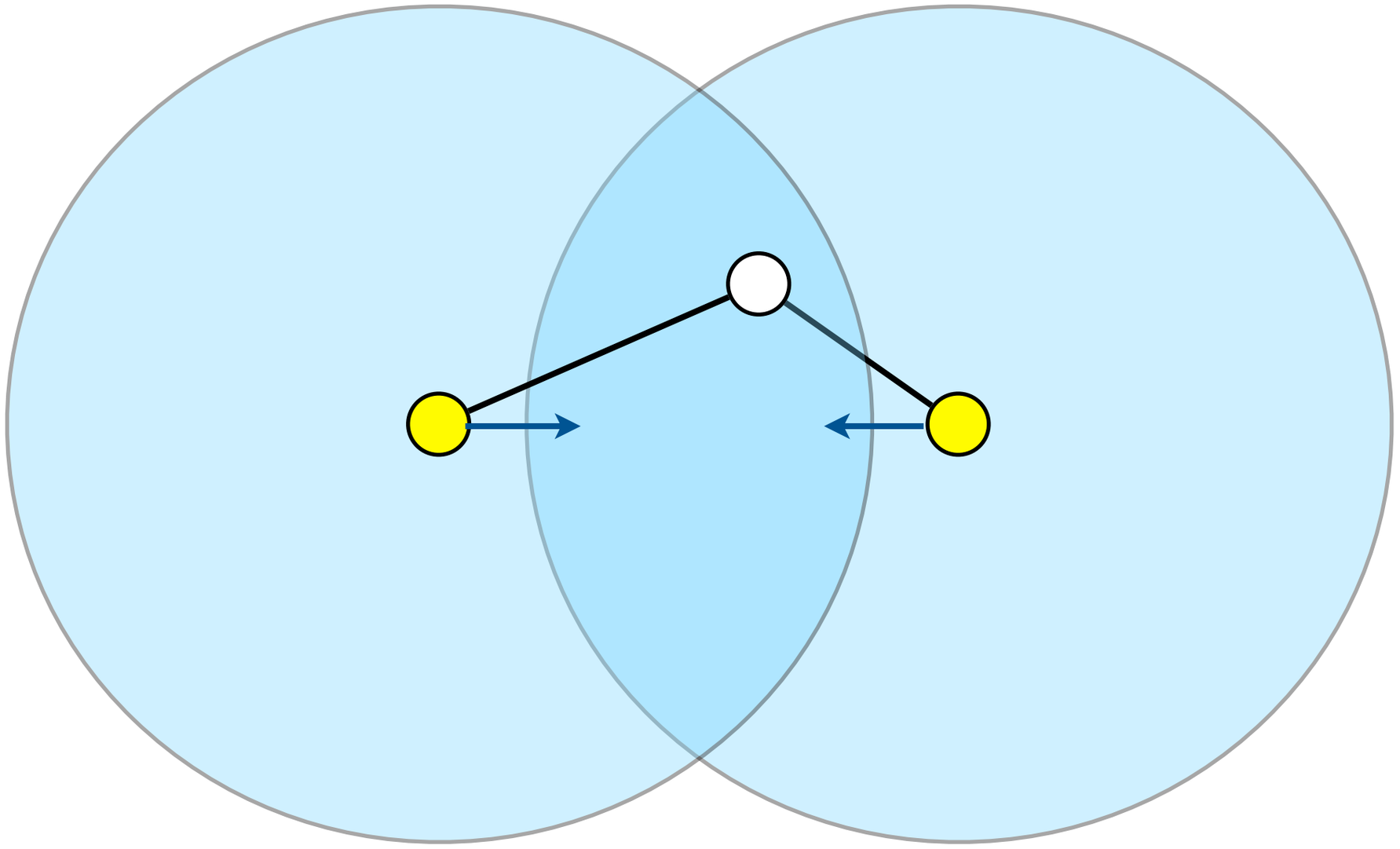


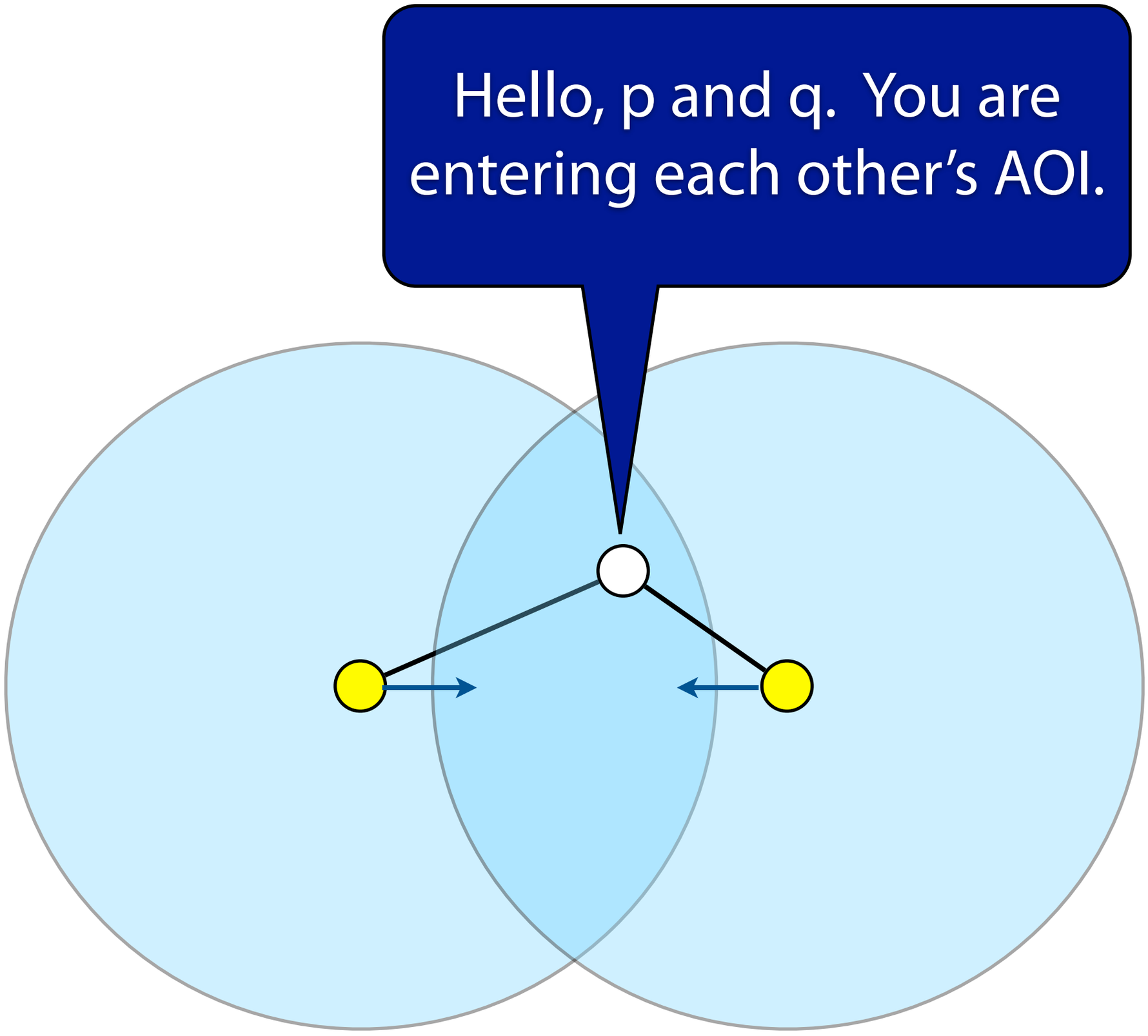
Q: How to update AOI-neighbors?



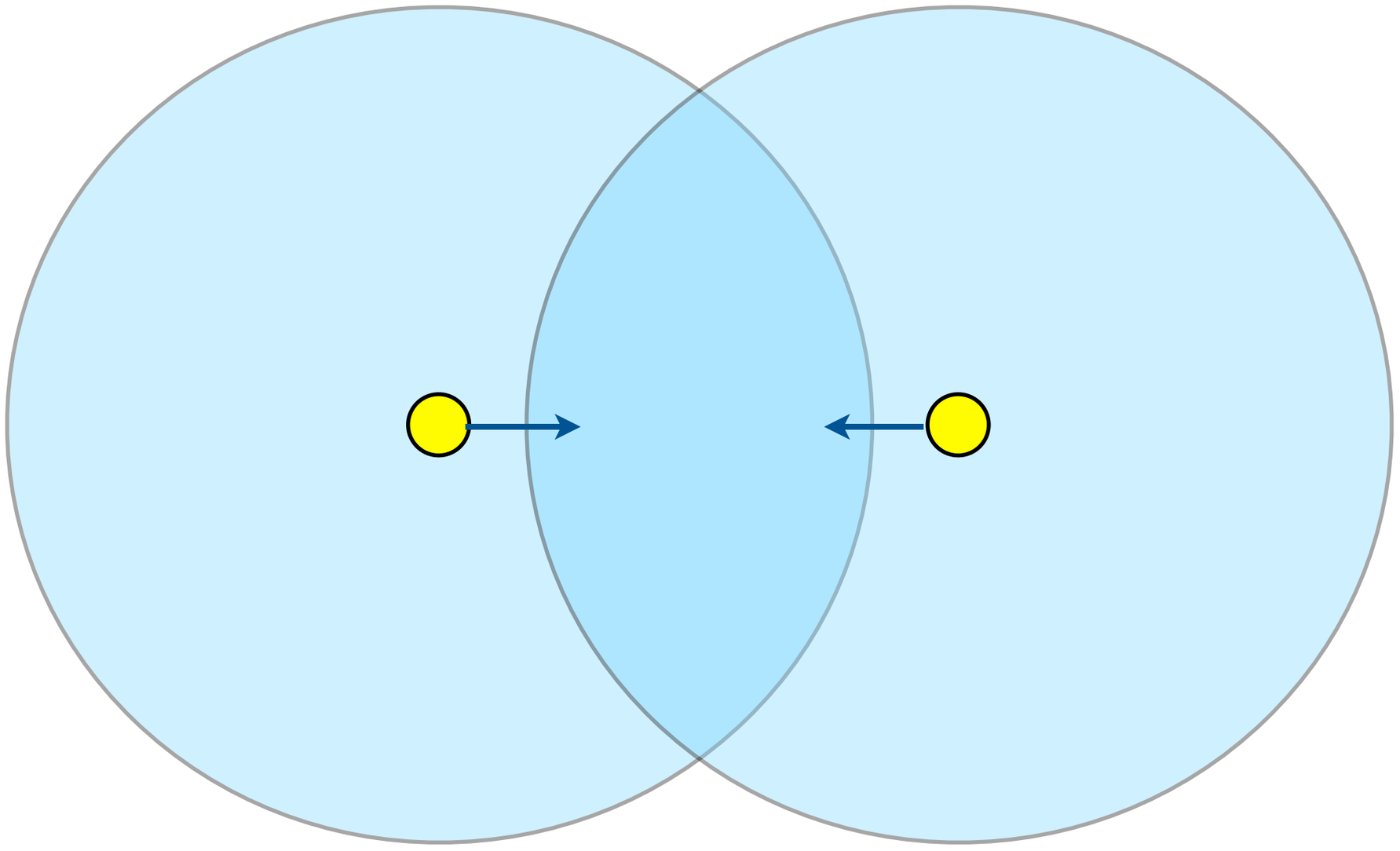


Case 1: Another player exists in the overlap area.

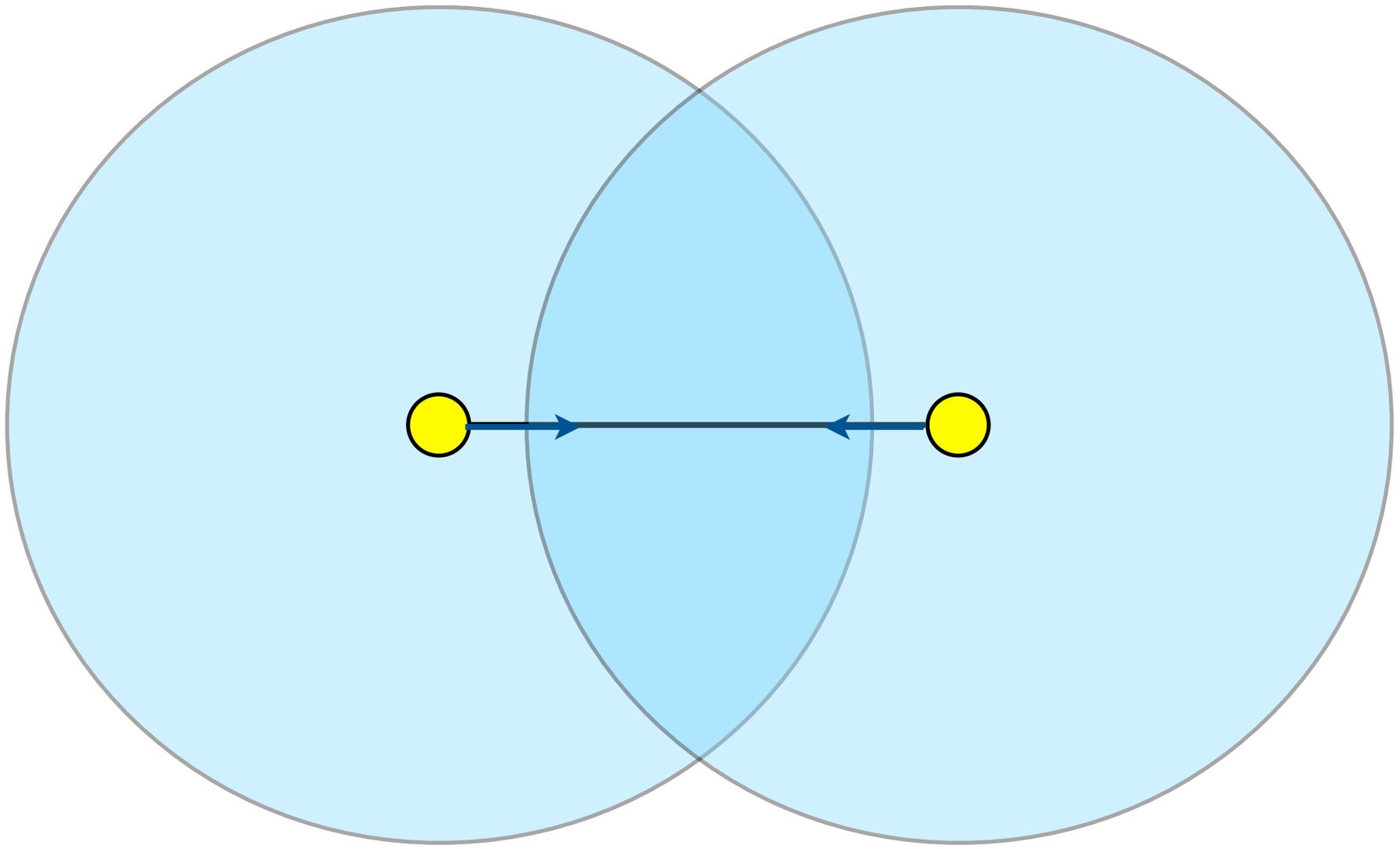



Hello, p and q. You are entering each other's AOI.

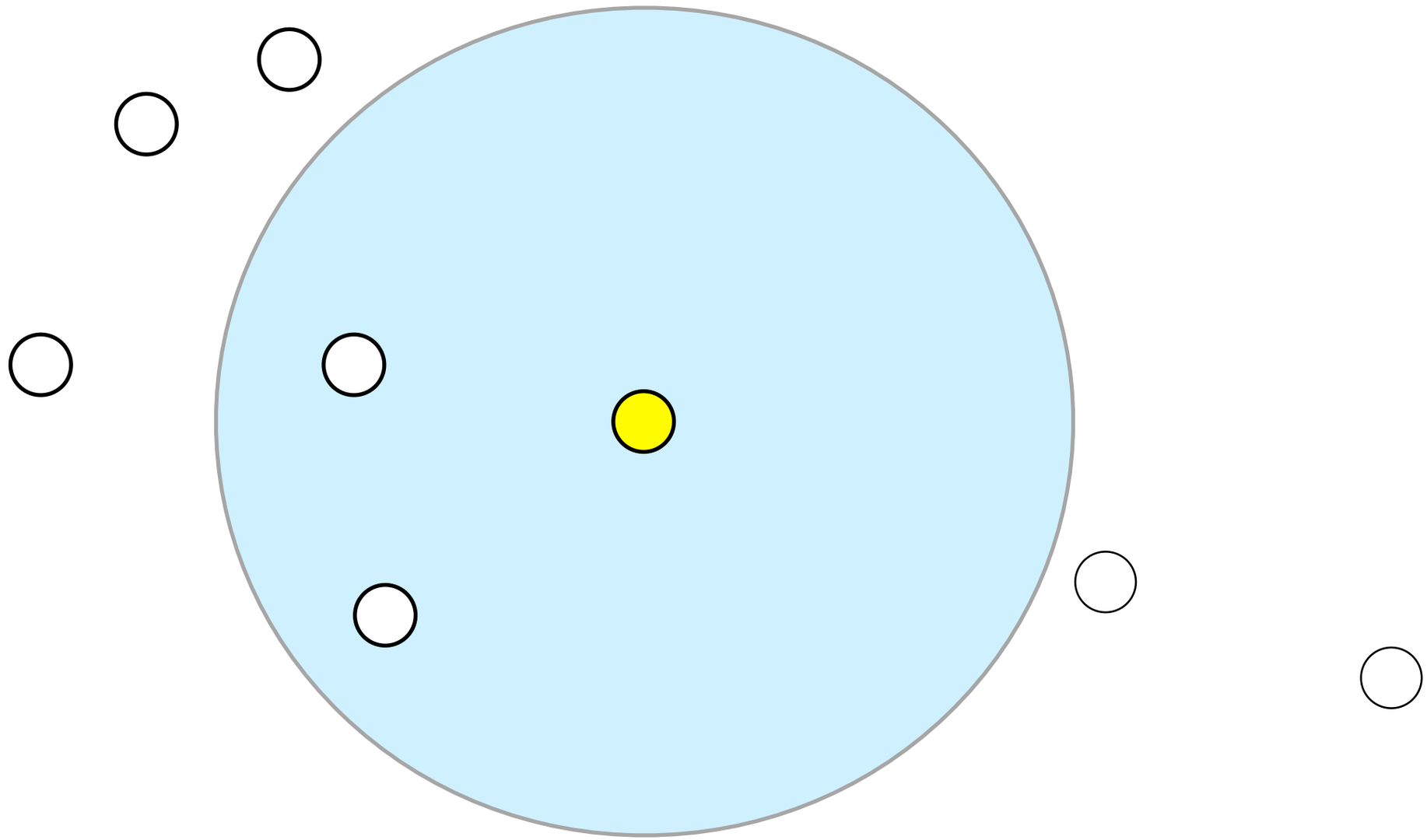
Case 2: No player exists in the overlap area.



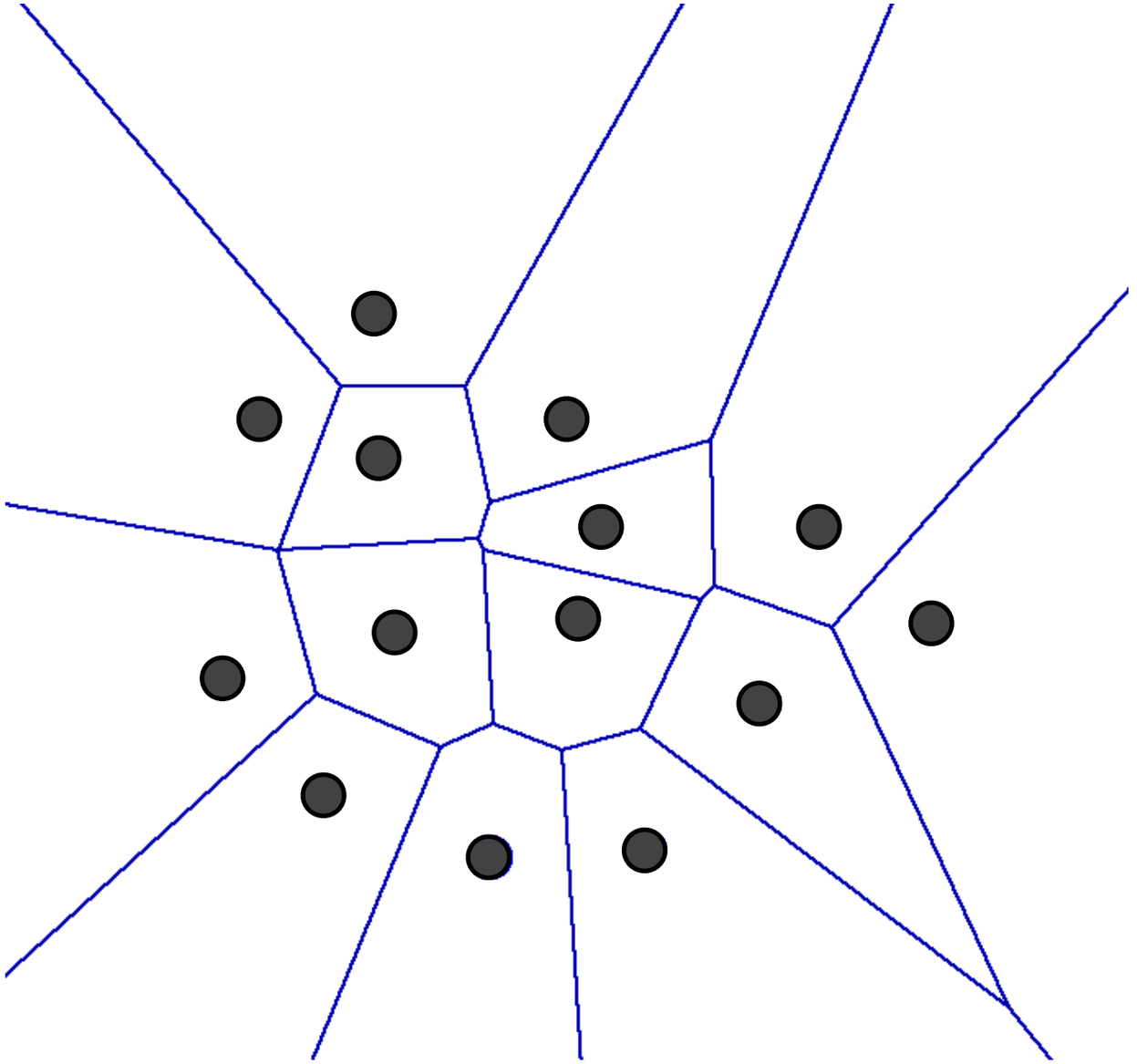
Players need to communicate with each other even if they are not in AOI



Challenge: Figure out who to communicate with without global information.



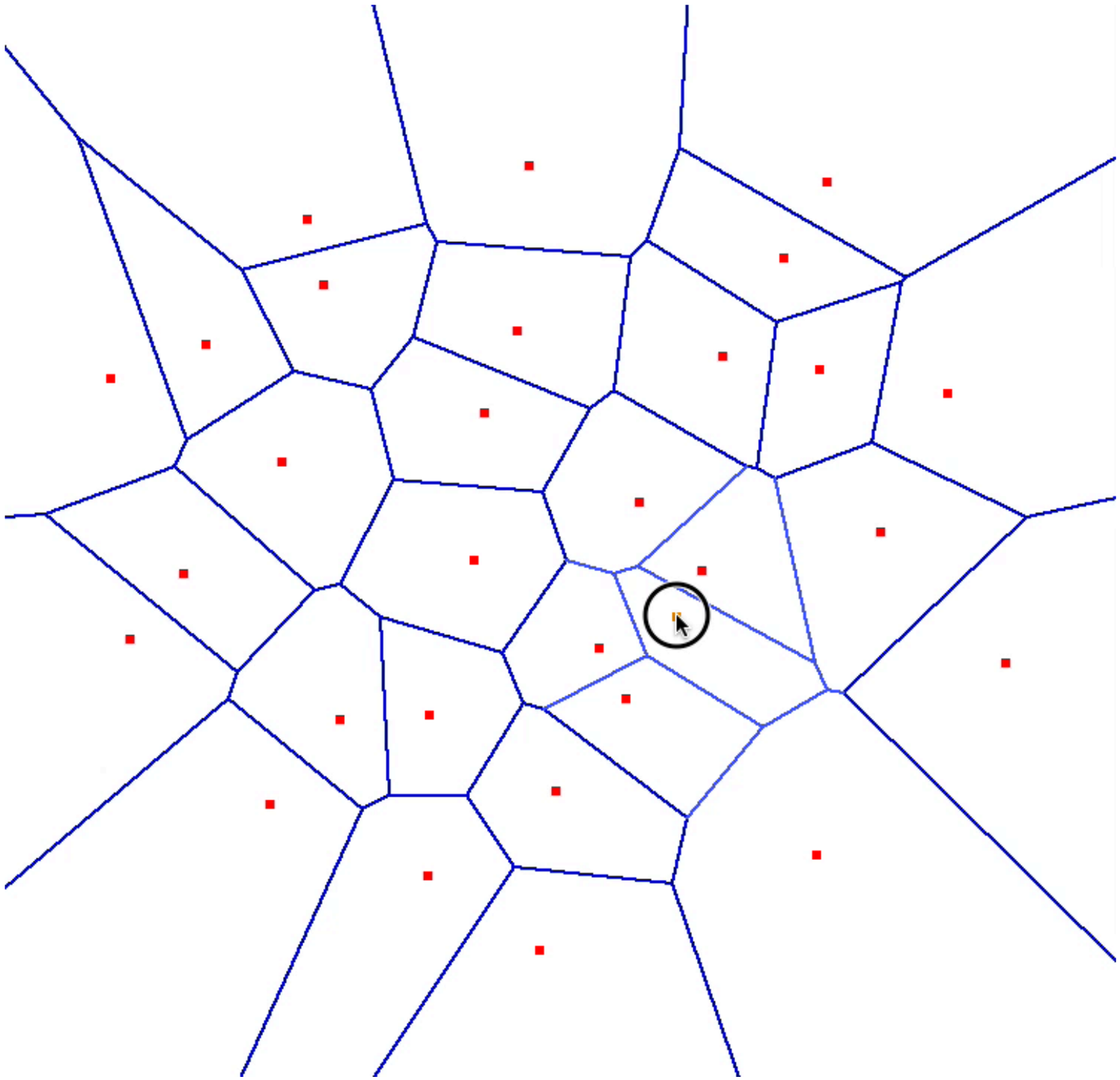
Voronoi Diagram

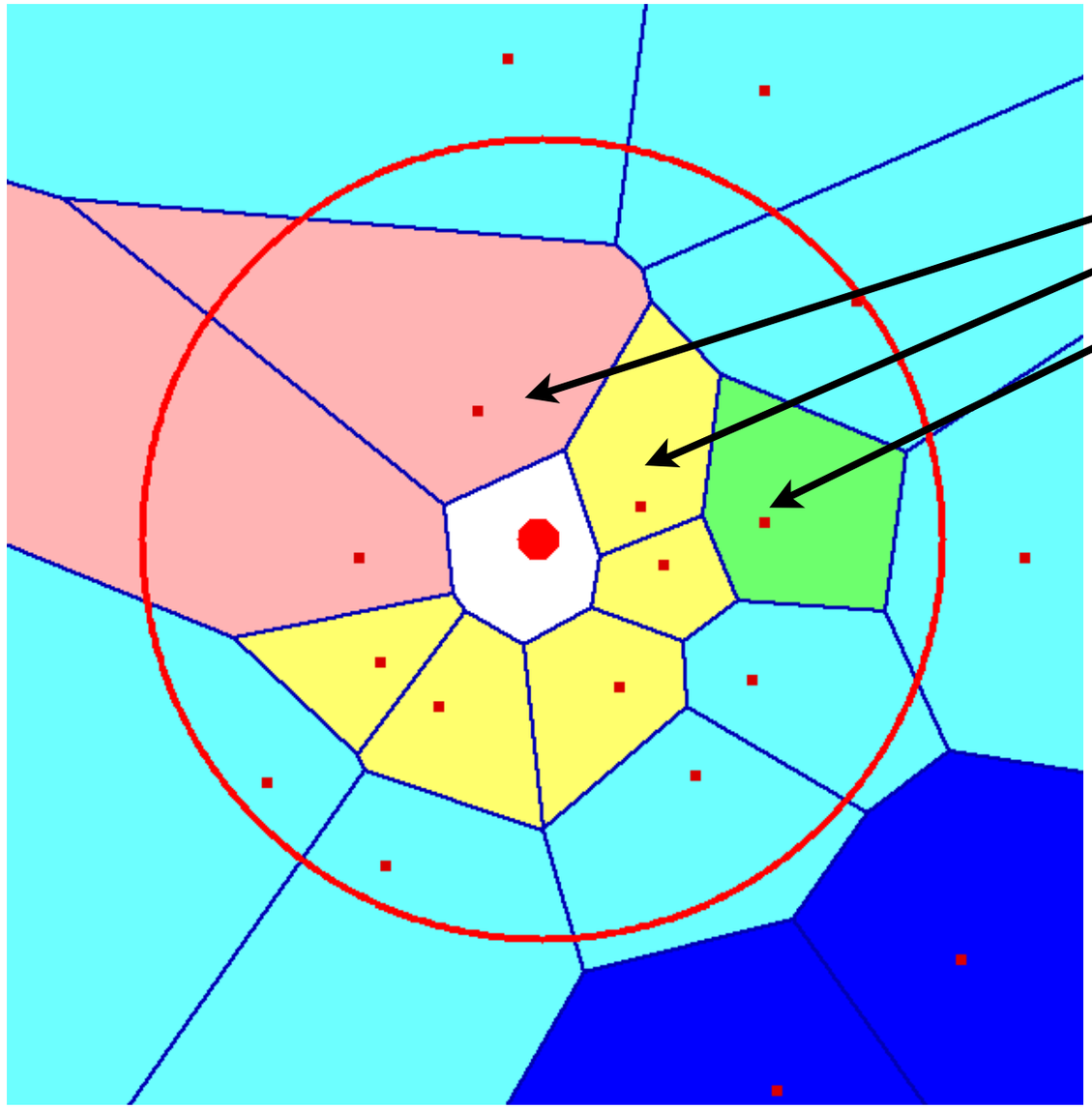


Every node is in charge of a region in the virtual world.

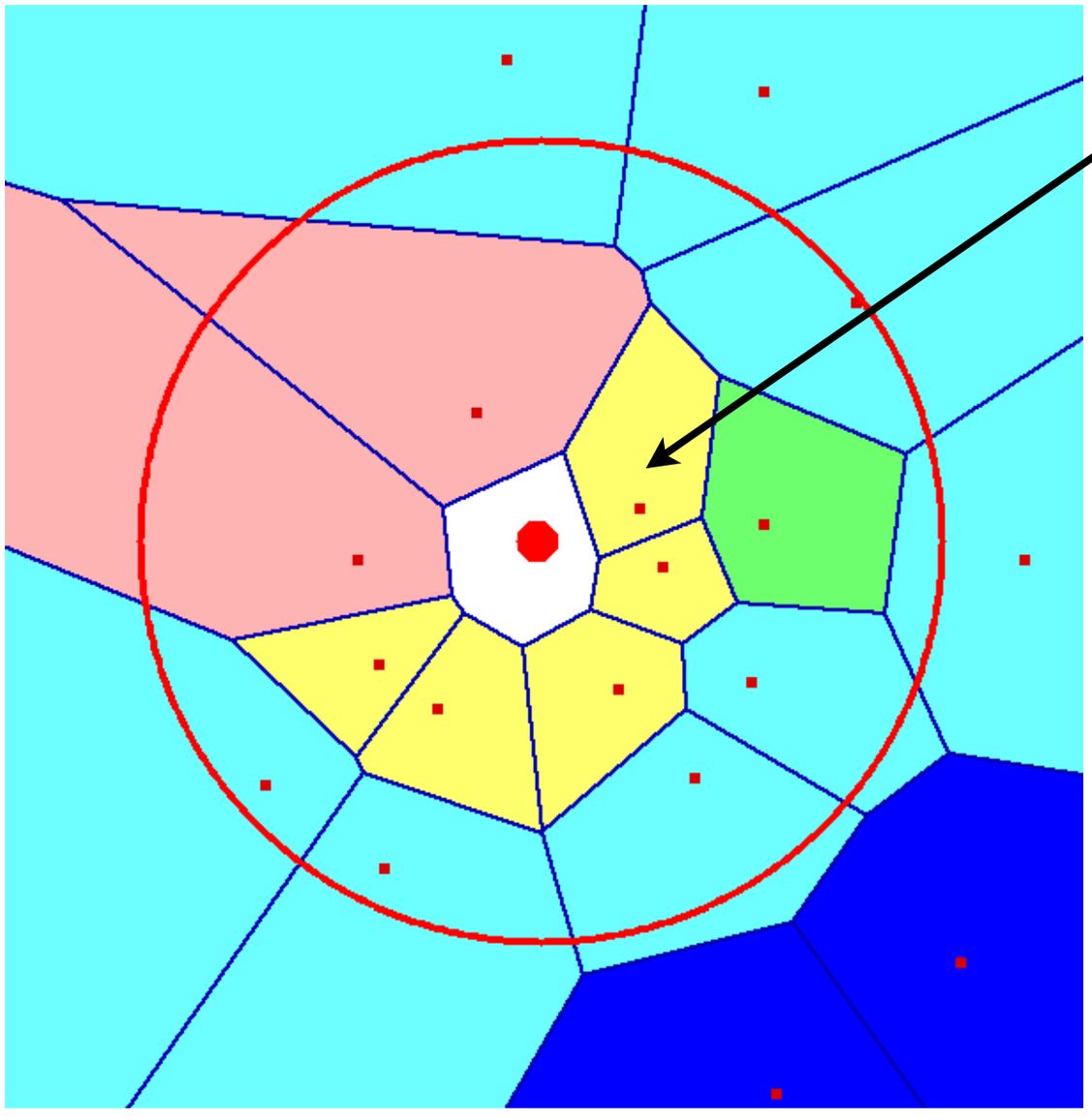
The region contains points closest to the node.

Voronoi Diagram



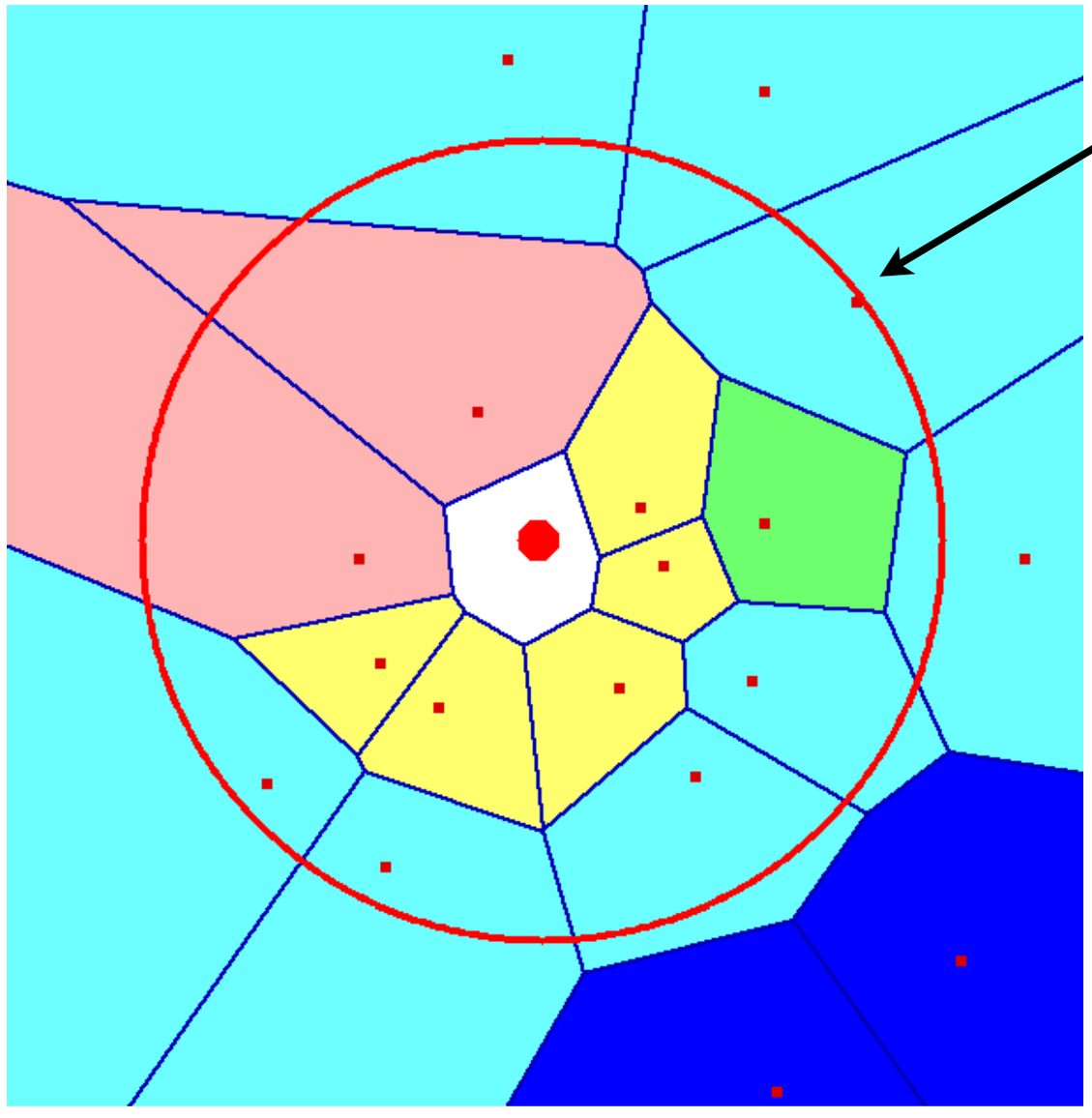


AOI Neighbors:
Neighbors in AOI



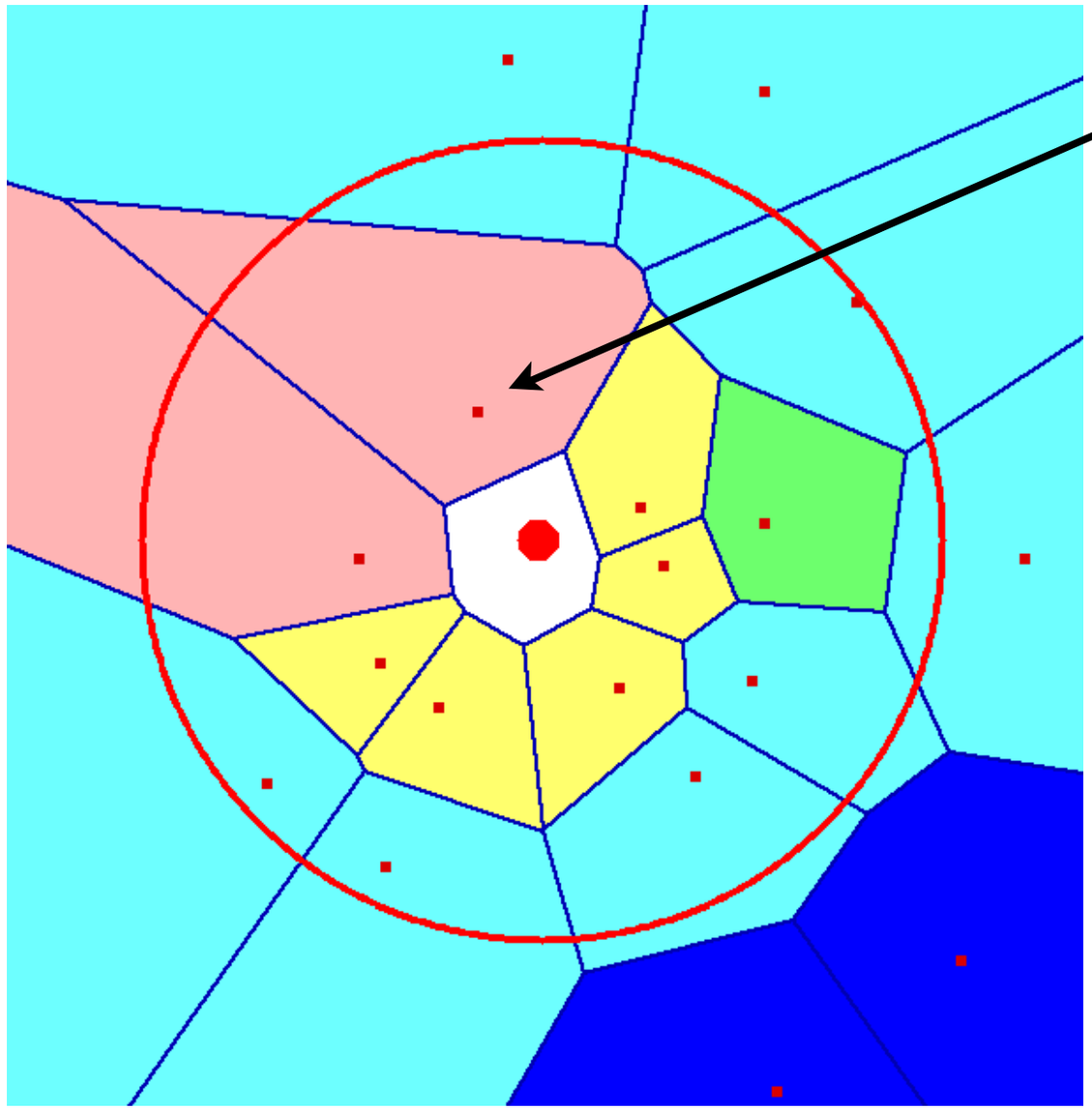
Enclosing Neighbors: Neighbors in adjacent region.

(may or may not be in AOI)

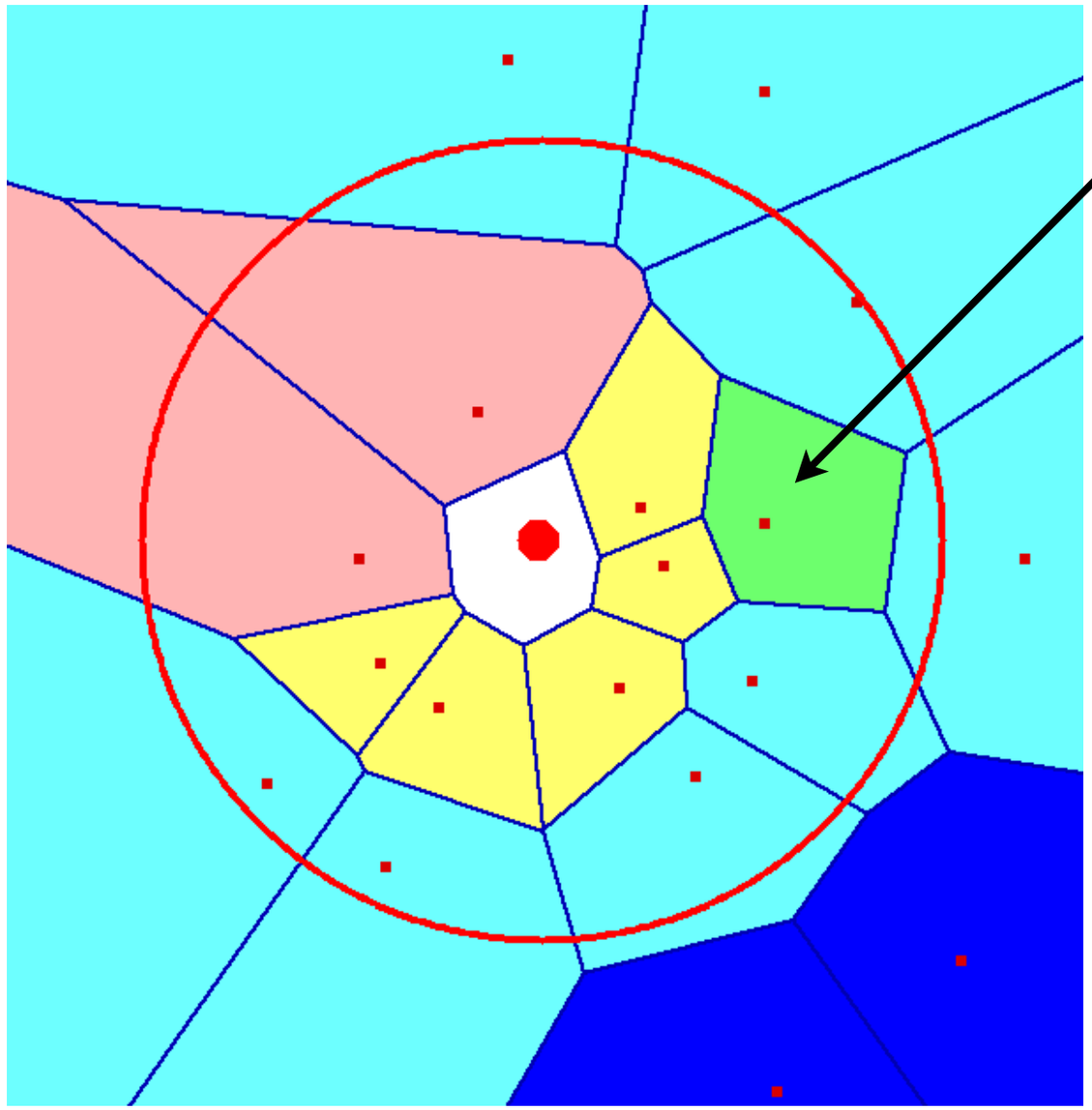


Boundary Neighbors:
Neighbors whose region intersect with AOI.

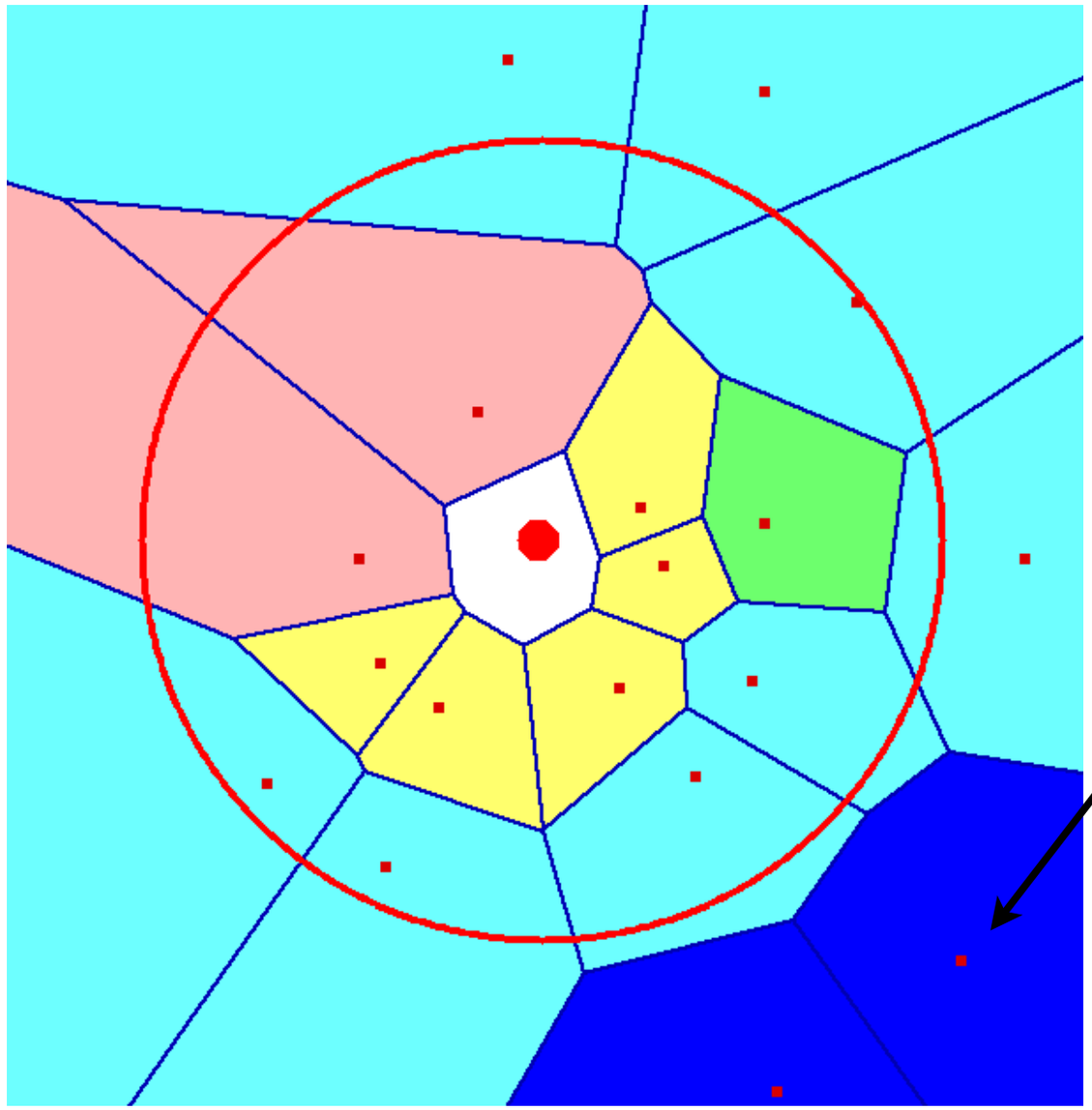
(may or may not be in AOI)



Boundary and Enclosing Neighbor

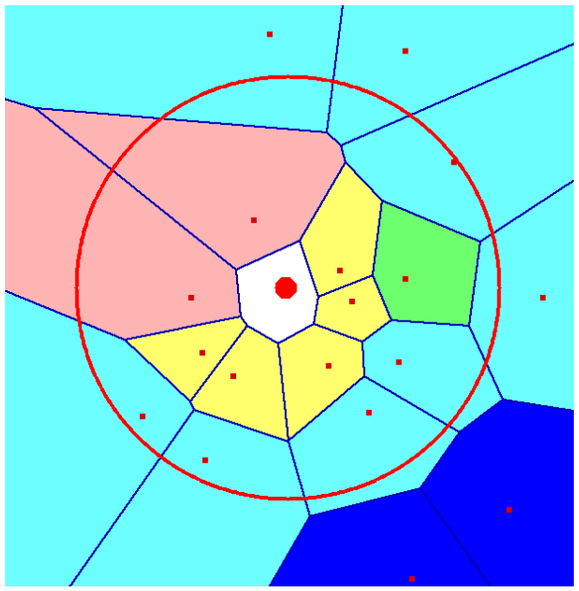


Regular AOI Neighbor: Non-boundary and non-enclosing neighbor in AOI

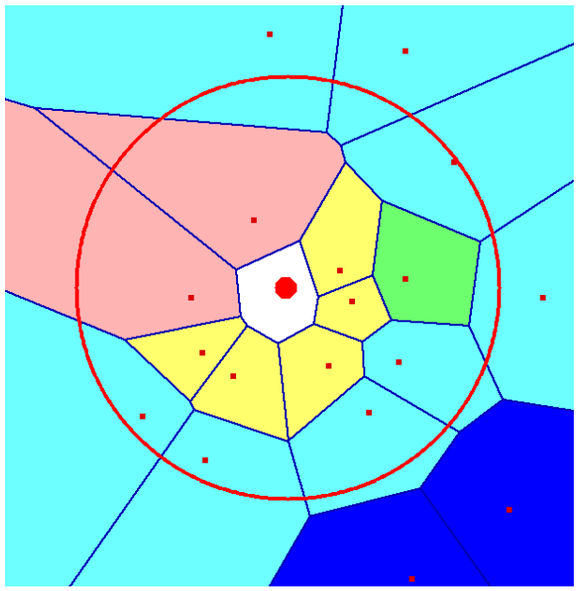


Unknown nodes
(not neighbors!)

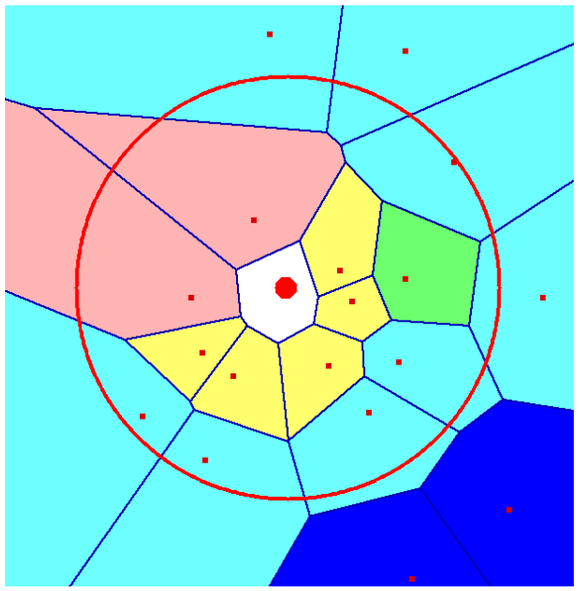
Type	in AOI?	intersect?	adjacent?
Regular	yes	no	no
Enclosing	maybe	no	yes
Boundary	maybe	yes	no
Enclosing +Boundary	maybe	yes	yes



A node always connects to its enclosing neighbors, regardless of whether they are in the AOI.



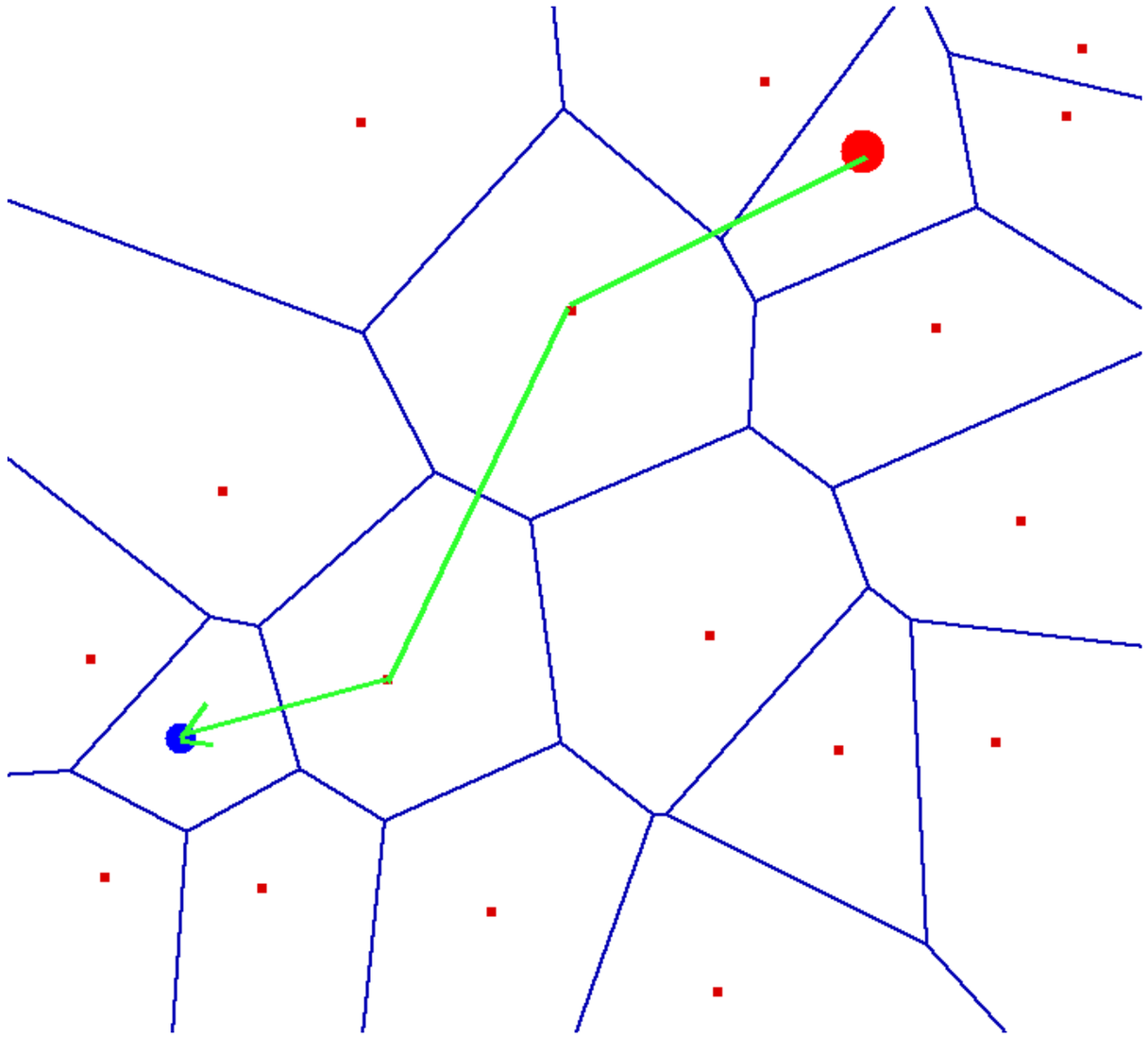
A node connects and exchanges updates with all neighbors.



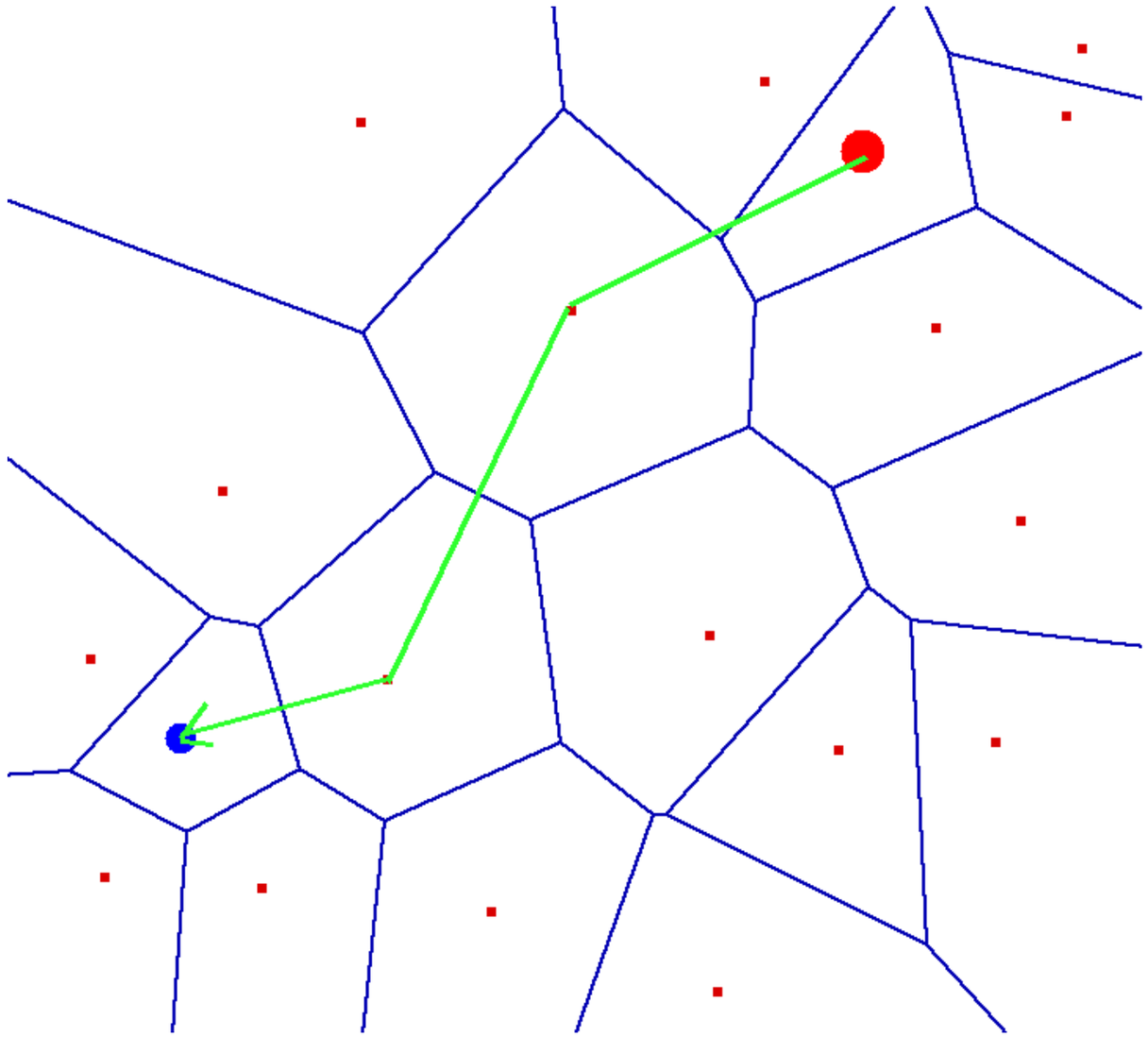
A node maintains Voronoi of all neighbors (regardless of inside AOI or not)

Suppose a player X wants to join. X sends its location to any node in the system.

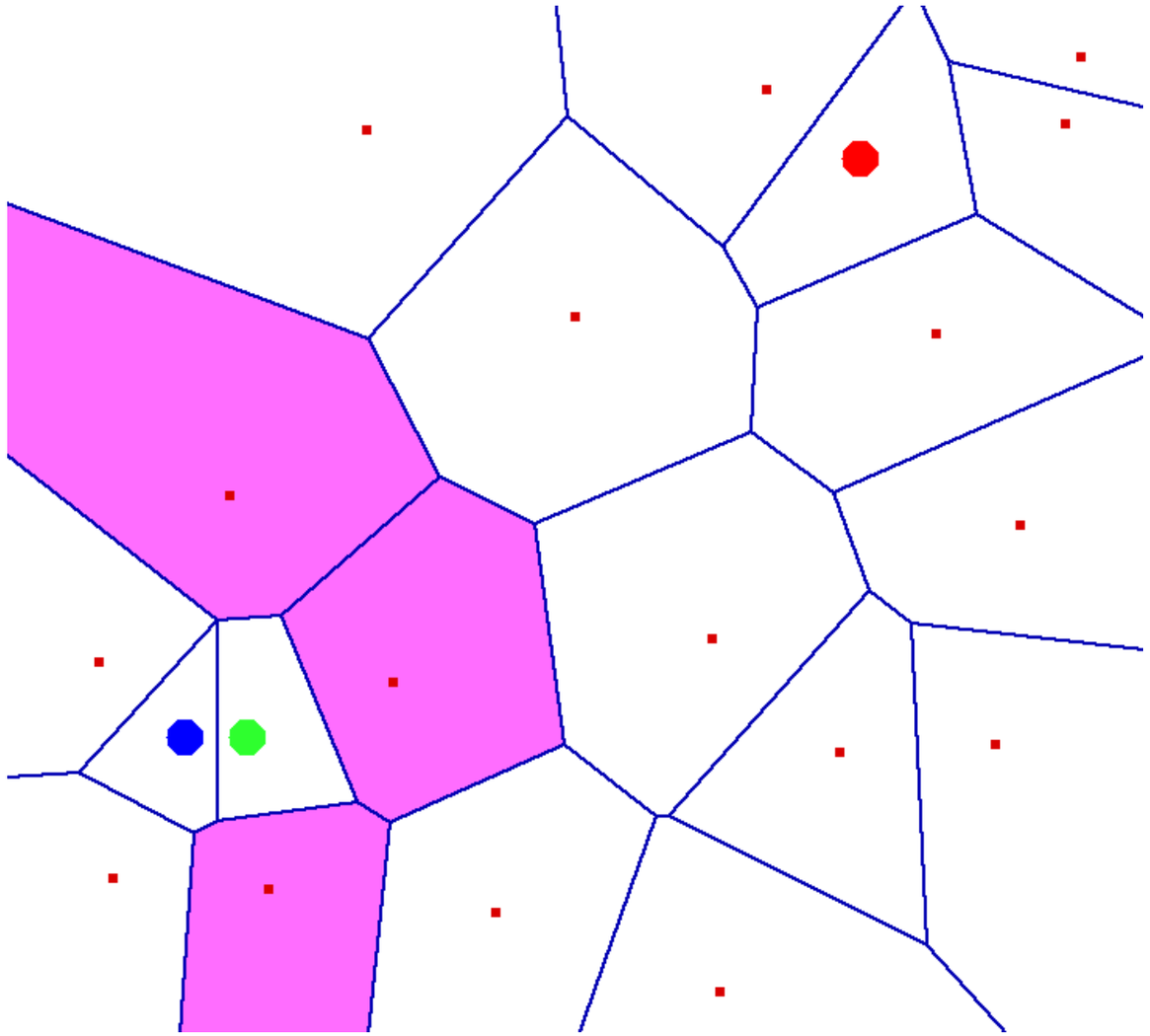
X join request is forwarded to the node in charge of the region (i.e., closest node to X), called acceptor.



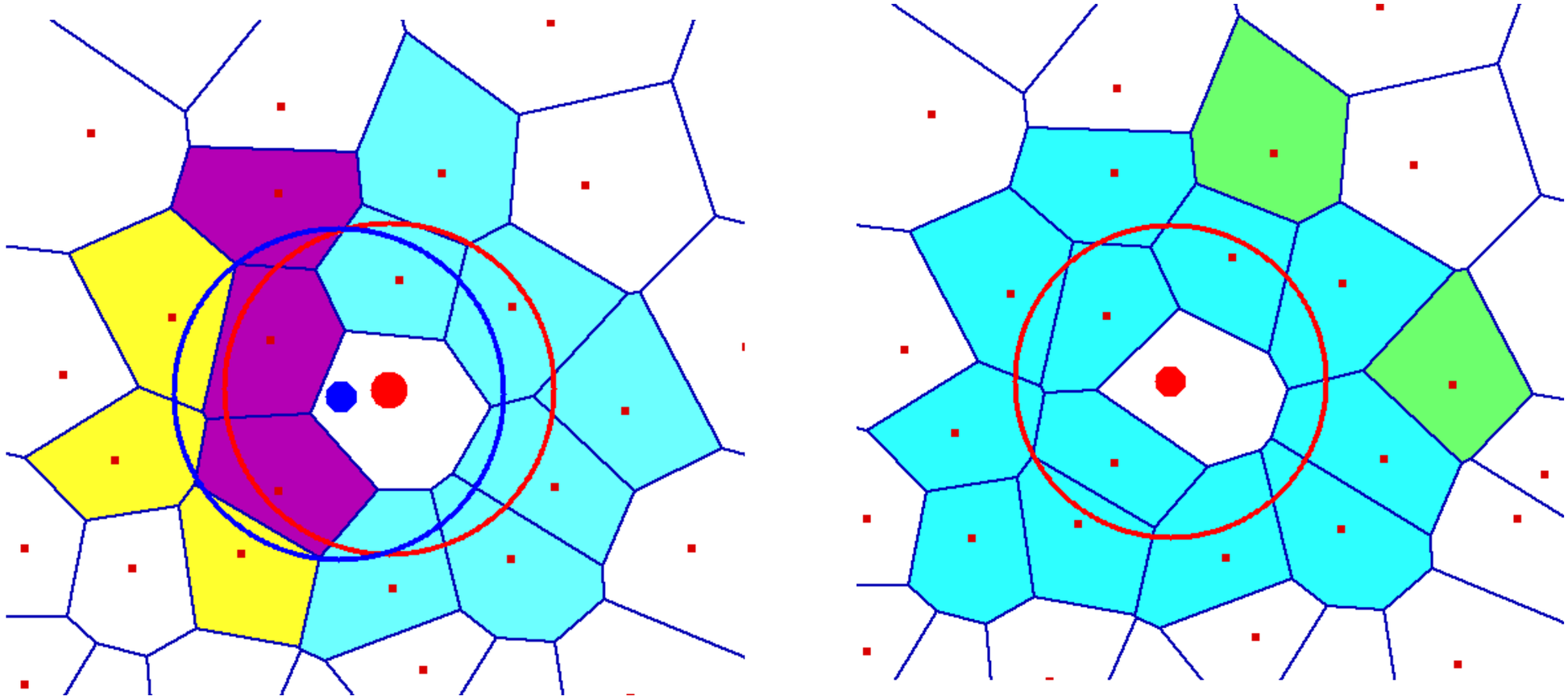
Forwarding is done greedily
(at every step, forward to neighbor closest to X)



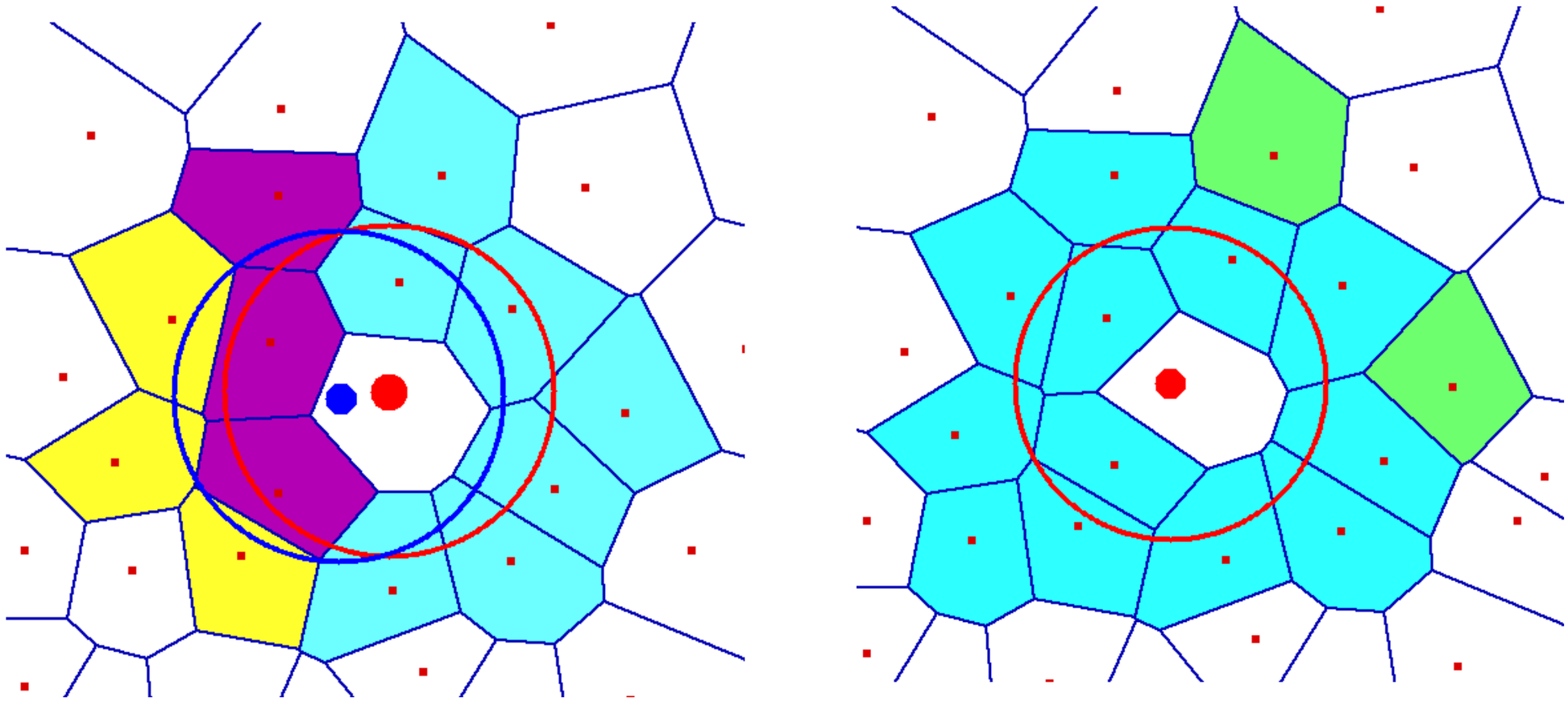
Acceptor informs the joining node X of its neighbors. Acceptor, X, and the neighbors update their Voronoi diagram to include the new node.



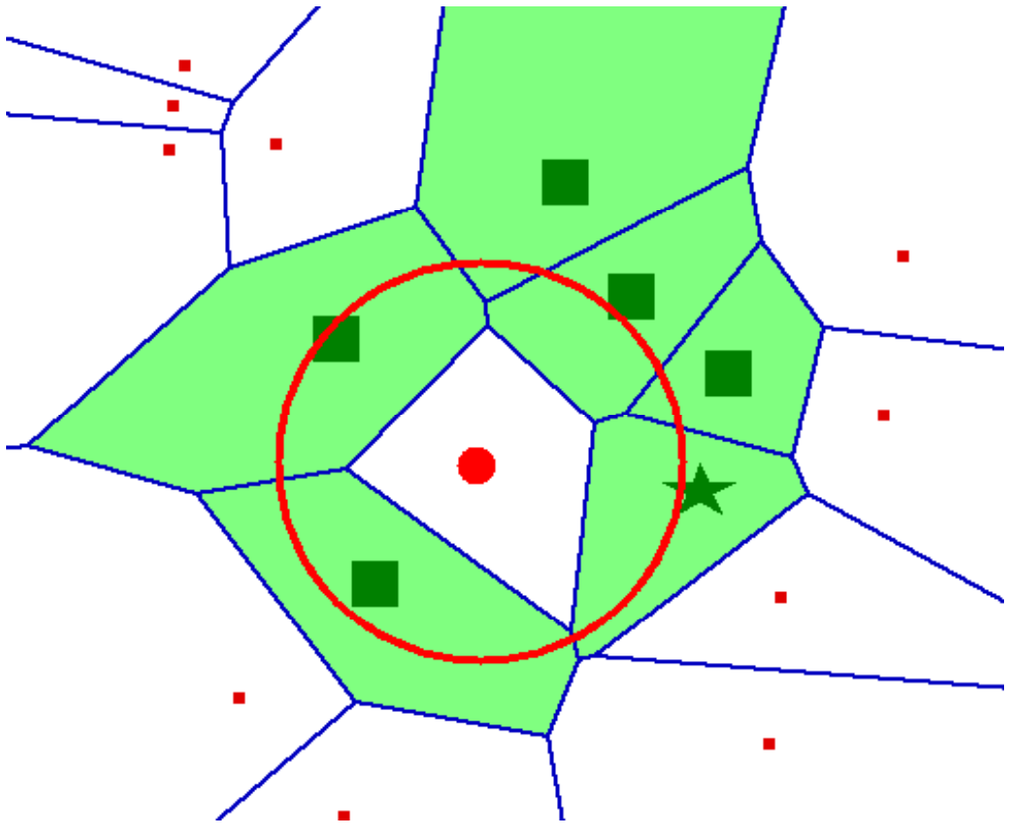
When X moves, X learns about new neighbors from the boundary neighbors.

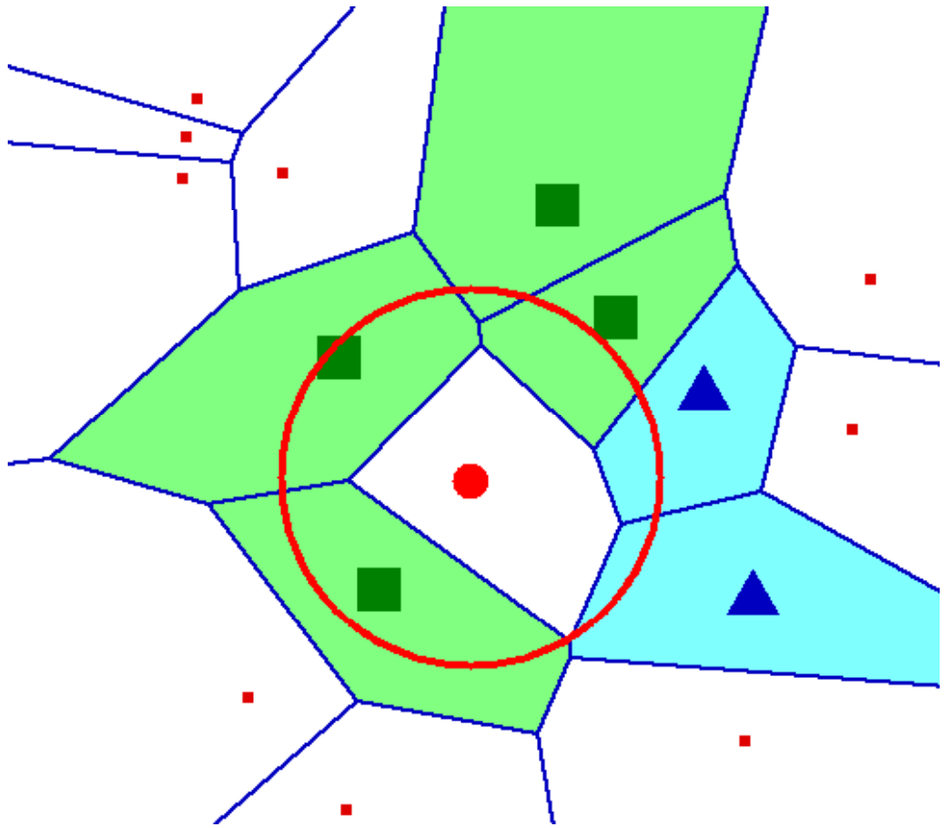


Boundary neighbors' enclosing neighbors may become new neighbors of X.



When a node disconnects, Voronoi diagrams are updated by the affected nodes. New boundary neighbors may be discovered.





VON Properties:

Number of connections depends on size of AOI, not size of virtual world

VON Properties:

Maintain a minimal number of enclosing neighbors when the world is sparse to ensure connectivity.

VON Properties:

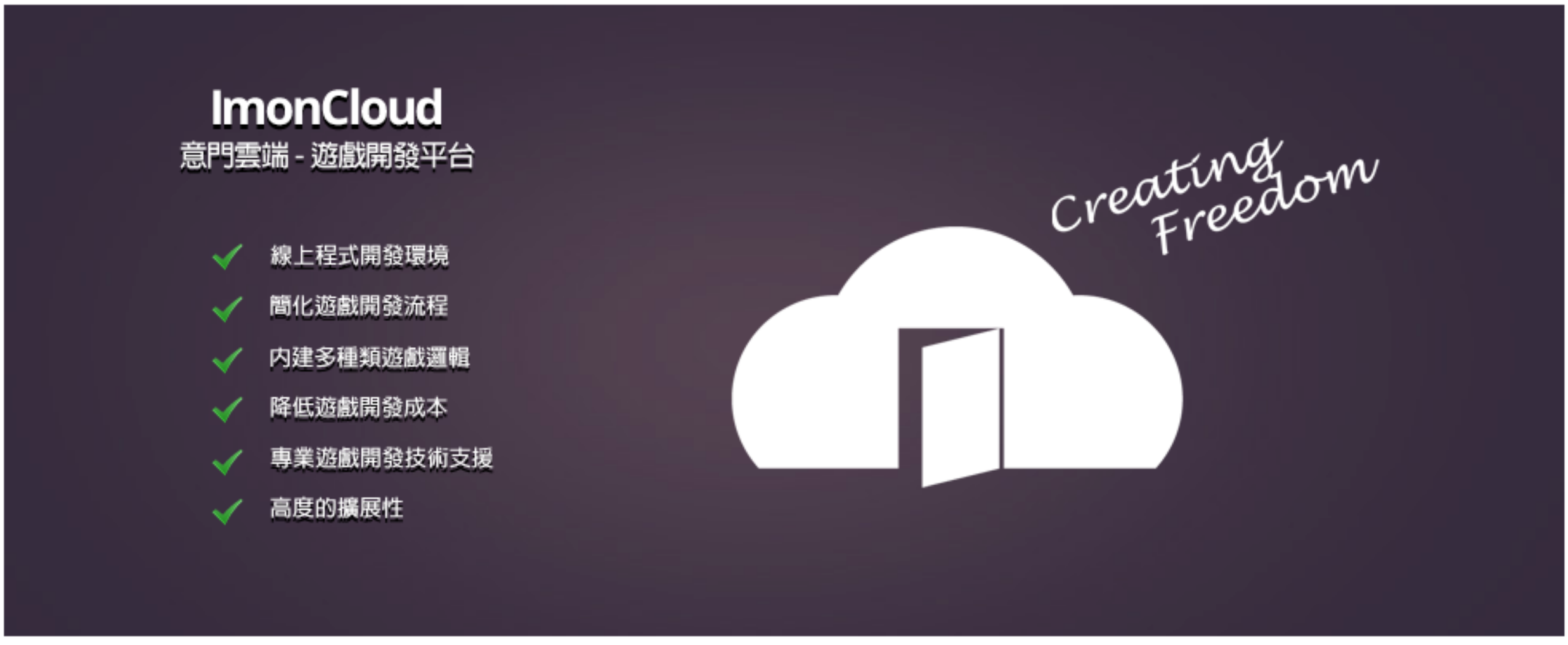
Boundary neighbors ensure that new neighbors are discovered.

VON Properties:

- Inconsistency may occur (e.g. with fast moving nodes)

ImonCloud*

Home | 線上開發 | 產品 | 價格 | 文件 | 討論區 | 聯絡我們



意門雲端 簡介

「意門雲端」(ImonCloud) 是一套可支援各類即時性遊戲或應用開發，且讓開發者在不修改伺服器程式的情況下，輕易支援大量連線人數的雲端平台 (Platform-as-a-Service, PaaS)。我們的目標為易用、低價、及可擴展，並期望降低開發 及維護虛擬世界的門檻，使小型公司或團隊亦可投入多人連線遊戲及應用的研發。