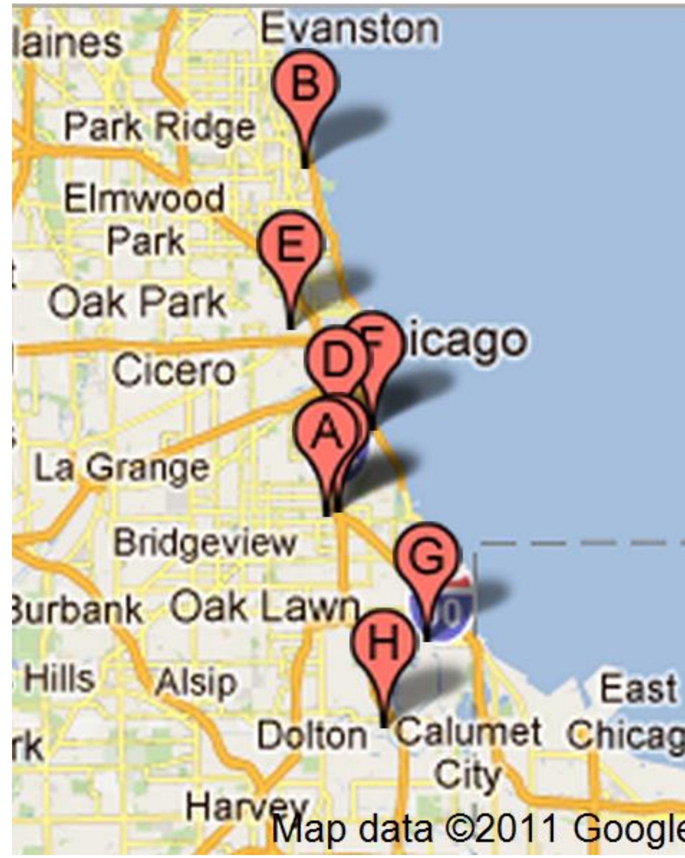
Dynamic Matching in Overloaded Waiting Lists

Jacob D. Leshno
Columbia Business School

Chicago Public Housing

- The Chicago public housing authority runs approx 20,000 apartments, spread throughout the city
- 60,000 applicants wait to be assigned on the capped waiting list
- Apartments become available stochastically over time as current tenants move out



Example: private vs. social

- Two agents a_1, a_2 with equal waiting costs,

  a_1 prefers N, a_2 prefers S

- S item arrives in period 1 and N item arrives in period 2

  Possible allocations:

  I. a_1 gets N, a_2 gets S and a_1 waits
  II. a_1 gets S, a_2 gets N and a_2 waits

- $I.$ is socially optimal, but a_1 may prefer $II.$

Dynamic Allocation

- Waiting lists, items arrive stochastically over time
  - Public housing
  - Organs for transplant
  - Nursing home spots
  - Daycare centers,…
- Welfare depends on the matching of items to agents
  - Agents have different preferences, which are private information
    - For example: location of family, work, school
  - Overloaded system – a matching agent is in the system, but need to ask agents to search for one
  - Impatient agents may misreport preferences to get assigned earlier
- How should we assign items dynamically to maximize welfare?

Talk Outline

- Model
  - 2 types of agent, 2 kinds of items
- Analyze a benchmark policy
  - Tractable formula for welfare
- Derive optimal policy
  - New queueing policy: the uniform-wait (UW) queue
- Simple robust policy
  - Give priority to agents who decline an item, and run a uniform lottery between them (SIRO)
- Extensions

Model

- Infinite pool of agents
  - Private types: α w.p. P_α, or β w.p. $P_\beta = 1 - P_\alpha$
  - Identical per period waiting cost c
  - Item valuations:

Value:	A	B
α	1	v
β	v	1

- One item, A or B, arriving each period
  - A with probability P_A, or B with probability $P_B = 1 - P_A$
  - Items must be assigned in the period they arrive
- For simplicity, no structural imbalance: $P_A = P_\alpha = p$

Welfare

We aim to maximize welfare, defined as the sum of agent utility gains

Lemma: Maximizing welfare is equivalent to minimizing misallocation

Intuition:

Overloaded system

$\Rightarrow$ One agent assigned, all others must wait

$\Rightarrow$ Can only shift waiting time between agents

Literature

- **Queueing:**
  - **Congestion costs:** Naor (1969), Hassin and Haviv (2006)
  - **Organs:** Zenios (1999), Su and Zenios (2004,2005), Alagoz, Mailllart, Schaefer, Roberts (2007)
  - **Public housing:** Kaplan (1986,8), Talreja and Whitt (2008), Caldentey, Kaplan, Weiss (2009)
- **Dynamic market design:** Unver (2010), Abdulkadiroglu and Loertscher (2007)
- **Dynamic mechanism design:** Bergermann and Said (2010), Gershkov and Moldovanu (2008,10), Lavi, Nisan (2005), Pavan, Segal, Toikka (2010)
- **Rationing and Misallocation:** Barzel (1974), Glaeser and Luttmer (2003)

No Choice Policy

- A mechanism that does not allow agents to express their preferences will result in random assignment
  - For example, not allowing to decline apartments

- The probability of mismatch:

$$\xi^{Rand} = P_A P_\beta + P_B\, P_\alpha$$
$$= 2p(1-p)$$

Benchmark Policy – FCFS

Single line for both goods, agents can decline and keep place in line

- Agents exogenously join and wait for both A and B
- Offer items according to First Come First Served (FCFS) order
- When offered, agents can choose :
  - Take offered item
  - Decline item and keep position
- Agents know their position in both waiting lists

FCFS - α's Choice

- Take current mismatched (B) item:

$$U_\alpha(Current\ B) = v$$

- Decline B, and stay first:

$$\mathrm{E}[U_\alpha(wait\ for\ A)] = 1 - c \times \frac{1}{p}$$

Animation

FCFS - α's Choice

- Take current mismatched item:

$$U_\alpha(Current\ B) = v$$

- Decline B, keep k-th position:

$$\mathrm{E}[U_\alpha(wait\ for\ A)|k\text{–th}\ position] = 1 - c \times \frac{k}{p}$$

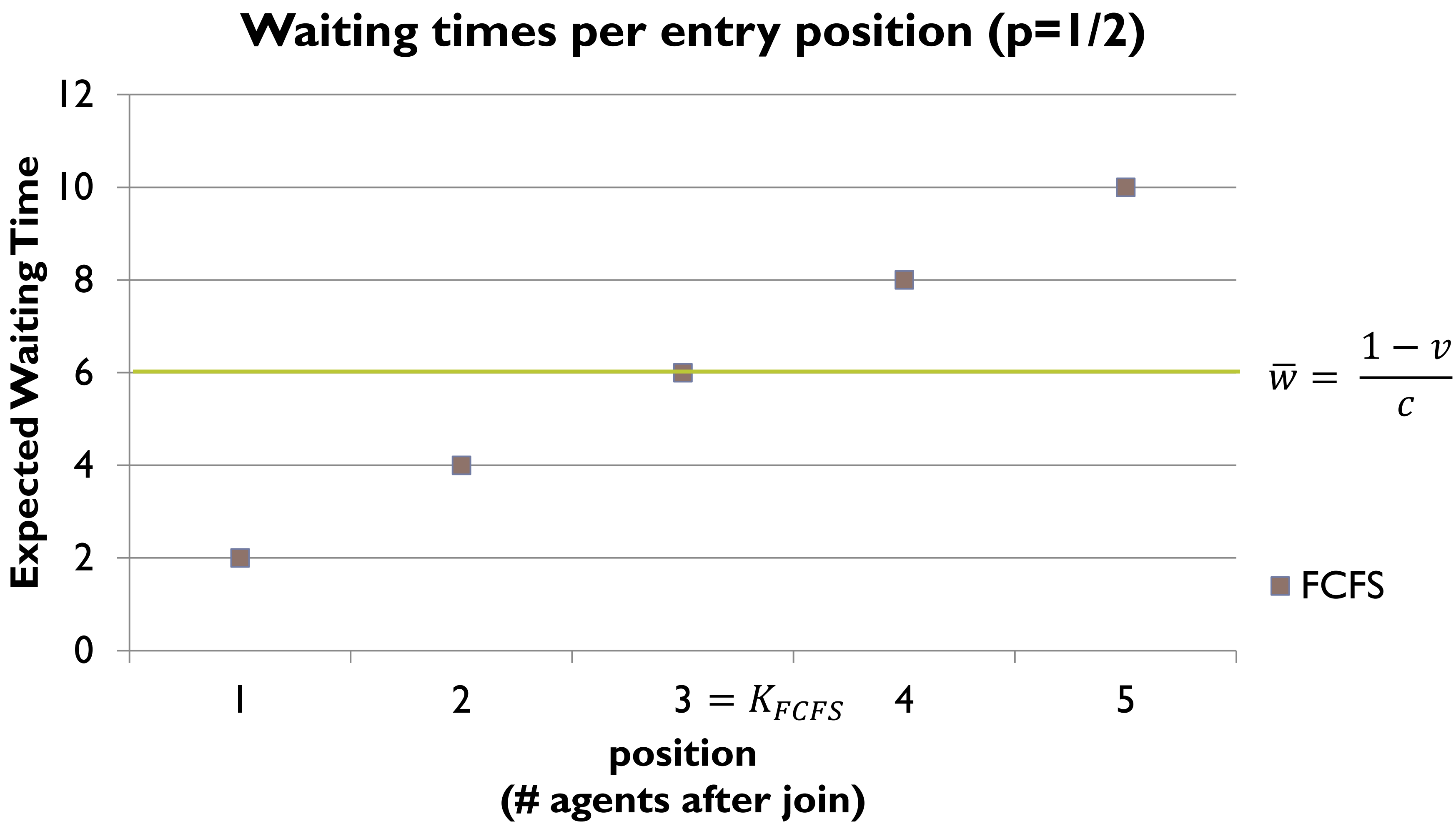
  - ≈ join the k-th position of a buffer-queue for A

- Decline and avoid mismatch only if

$$k \le K_\alpha = \left\lfloor p\frac{1-v}{c} \right\rfloor = \lfloor p\,\overline{w} \rfloor$$

Animation

Maximal Size of the Buffer-Queue



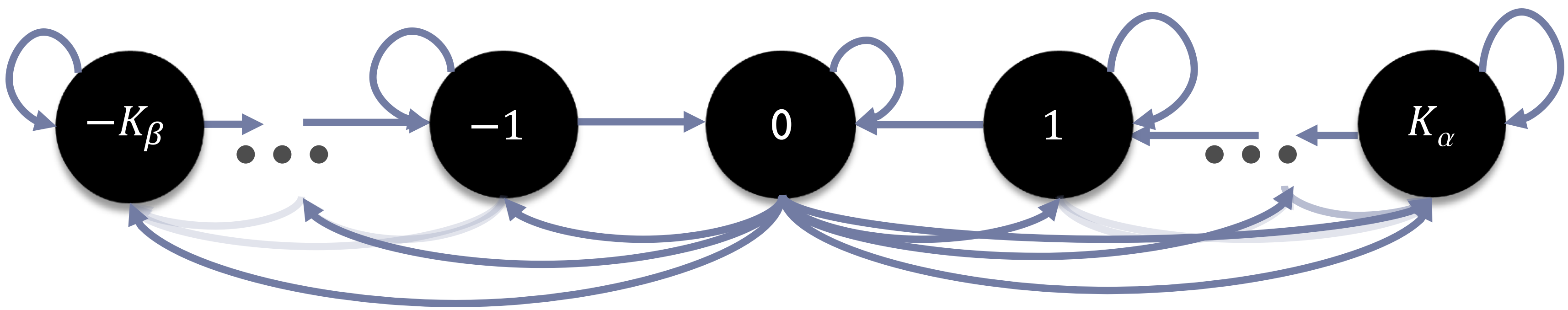
FCFS – System Dynamics

- *Proposition:* The dynamic behavior of the system can be captured by the state of the *buffer-queue* (agents who declined items)
  - Waiting agents who previously declined are of a single type
  - Number of α agents $\leq K_\alpha$
  - Number of β agents $\leq K_\beta$
  - System is Markovian with the state space:

$$S = \{-K_\beta , \dots , -1 , 0, 1, 2, \dots , K_\alpha\}$$

    - State *k>0 indicates* k α-agents declined and are waiting for an A

System Dynamics



Animation

Welfare under FCFS

- Expected welfare loss under FCFS buffer-queue:

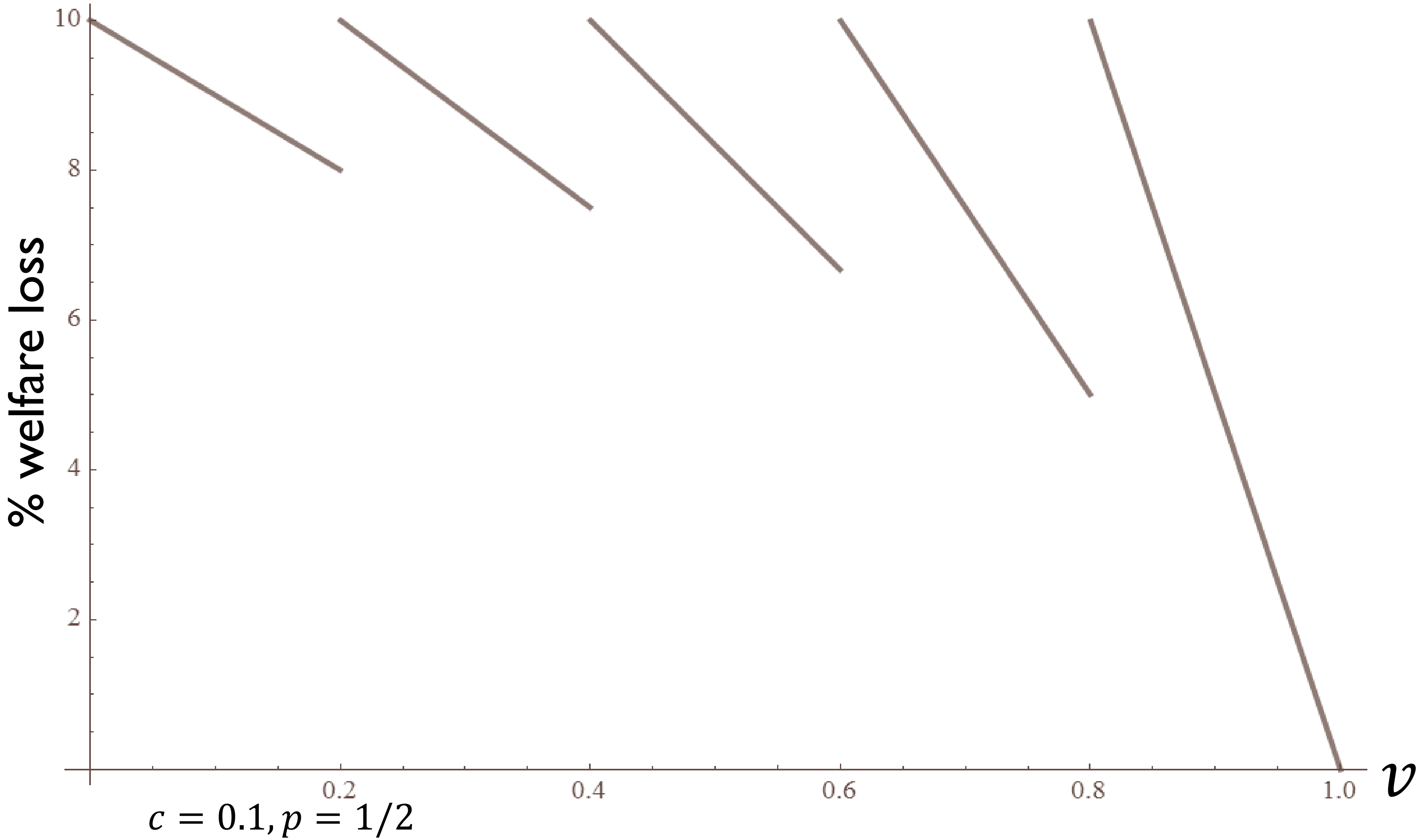
$$WFL_{FCFS} = (1-v)\xi^{FCFS}$$

$$= (1-v)\frac{2p(1-p)}{K_\alpha(1-p)+K_\beta p+1}$$

where by the IC

$$K_\alpha = \left\lfloor p\frac{1-v}{c}\right\rfloor, \qquad K_\beta = \left\lfloor (1-p)\frac{1-v}{c}\right\rfloor$$

FCFS – Welfare Loss from Mismatch



Optimal Buffer-Queue

Can we do better?

- Stationary mismatch probability depends only on the buffer-queue lengths:

$$\xi = \frac{2p(1-p)}{K_\alpha(1-p) + K_\beta p + 1}$$

- Design a buffer-queue policy to get higher K_α, K_β
  - Avoid mismatch by placing up to K_α or K_β agents in buffer-queue
  - Search for matching agents until buffer-queue is full
  - Size limited by incentive constraints - agents must be willing to join

Optimal Buffer-Queue

- The buffer-queue is *Incentive Compatible* if the expected wait w_k at position $k \leq K$ satisfies:

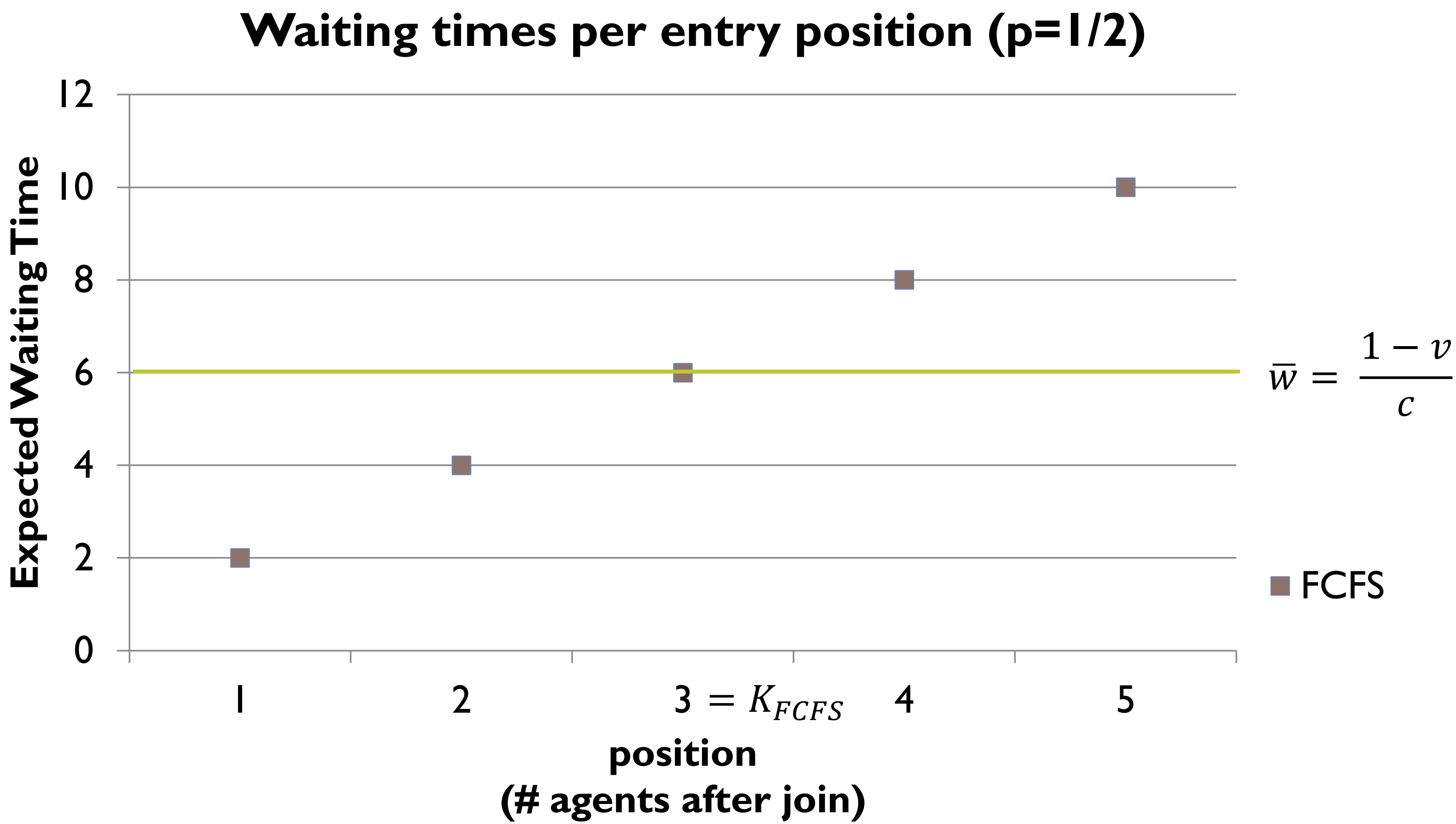
$$1 - c \cdot w_k \geq v$$

or

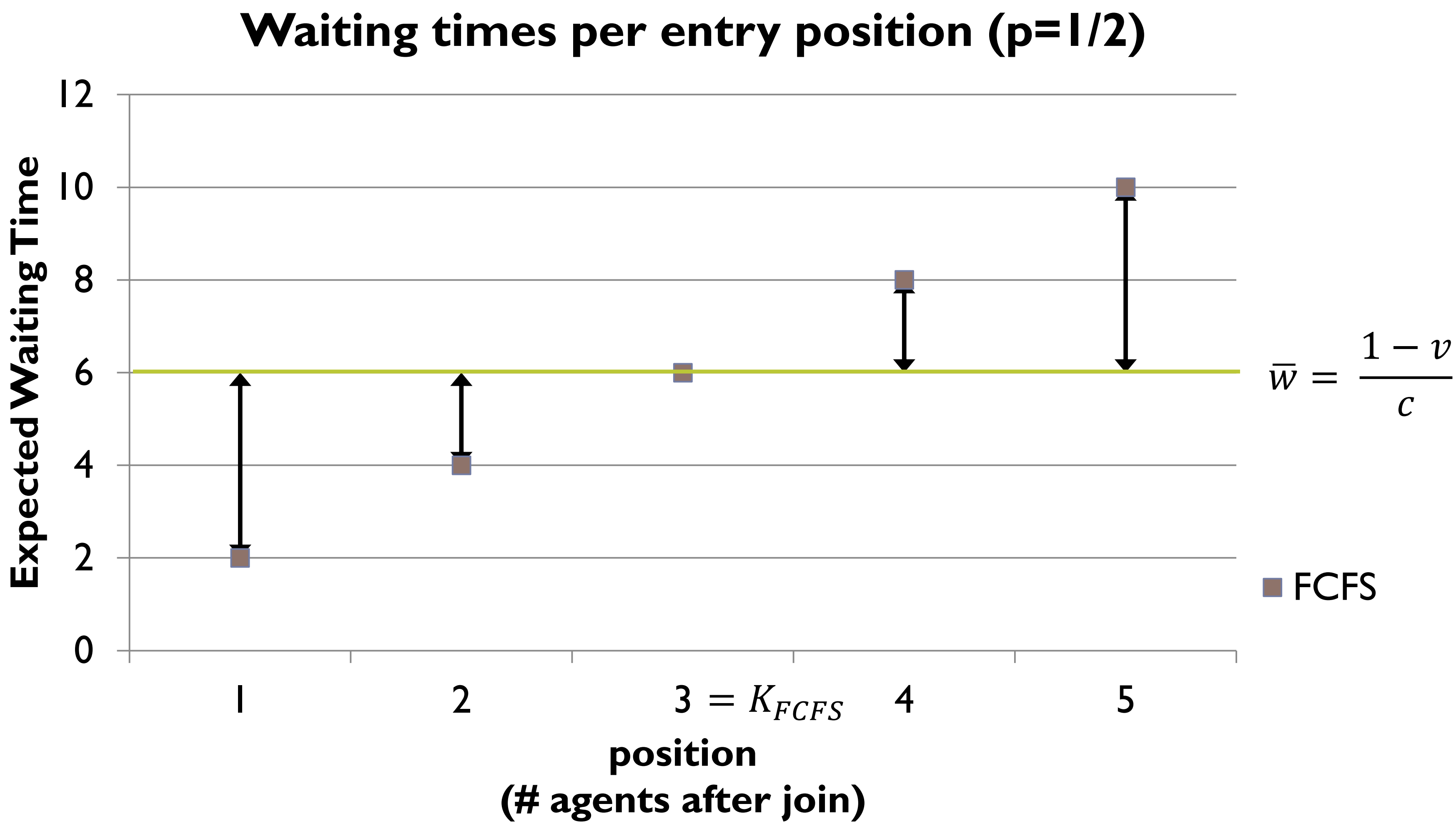
$$w_k \leq \bar{w} = \frac{1-v}{c}$$

- *Can optimize the A buffer-queue independently*
- *We reduced the problem to finding a policy for the buffer-queue that minimizes balking (i.e. minimizes taking mismatch instead of waiting)*

Can we do better?



Can we do better?



Buffer-Queue Policies

- $\langle K, \varphi \rangle$ queue policy
  - Up to K agents on the buffer-queue
  - Assign item w.p. $\varphi(k, i)$ to agent in position i when k agents are on the buffer-queue
  - Examples:
    - $\varphi(k, 1) = 1 \implies FCFS$
    - $\varphi(k, k) = 1 \implies LCFS$

Upper Bound

- *Lemma:* Expected wait for a random position is independent of assignment probabilities:

$$\mathrm{E}[w_{\tilde{k}}] = \frac{K+1}{2p}$$

- *Proposition:* There is no incentive compatible policy with

$$K > K^* = \lfloor 2p\bar{w} \rfloor - 1$$

Proof of Lemma:

- Limit attention to when the A-queue is not empty
- Conditional probability of having k agents waiting is $1/K$
- By Little's law

$$\mathrm{E}[w_{\tilde{k}}] = \frac{E[\#agents\ waiting]}{arrival\ rate} = \frac{1}{p} \cdot \sum_{k=1}^{K} \frac{k}{K}$$

$$= \frac{K+1}{2p}$$

Proof of proposition:

- For any IC policy $\langle K, \varphi \rangle$ we have for every k:

$$w_k \leq \bar{w}$$

Therefore, expected IC holds:

$$\mathrm{E}[w_{\tilde{k}}] = \frac{K+1}{2p} \leq \bar{w}$$

giving

$$K \leq 2p\bar{w} - 1$$

Optimal Policy

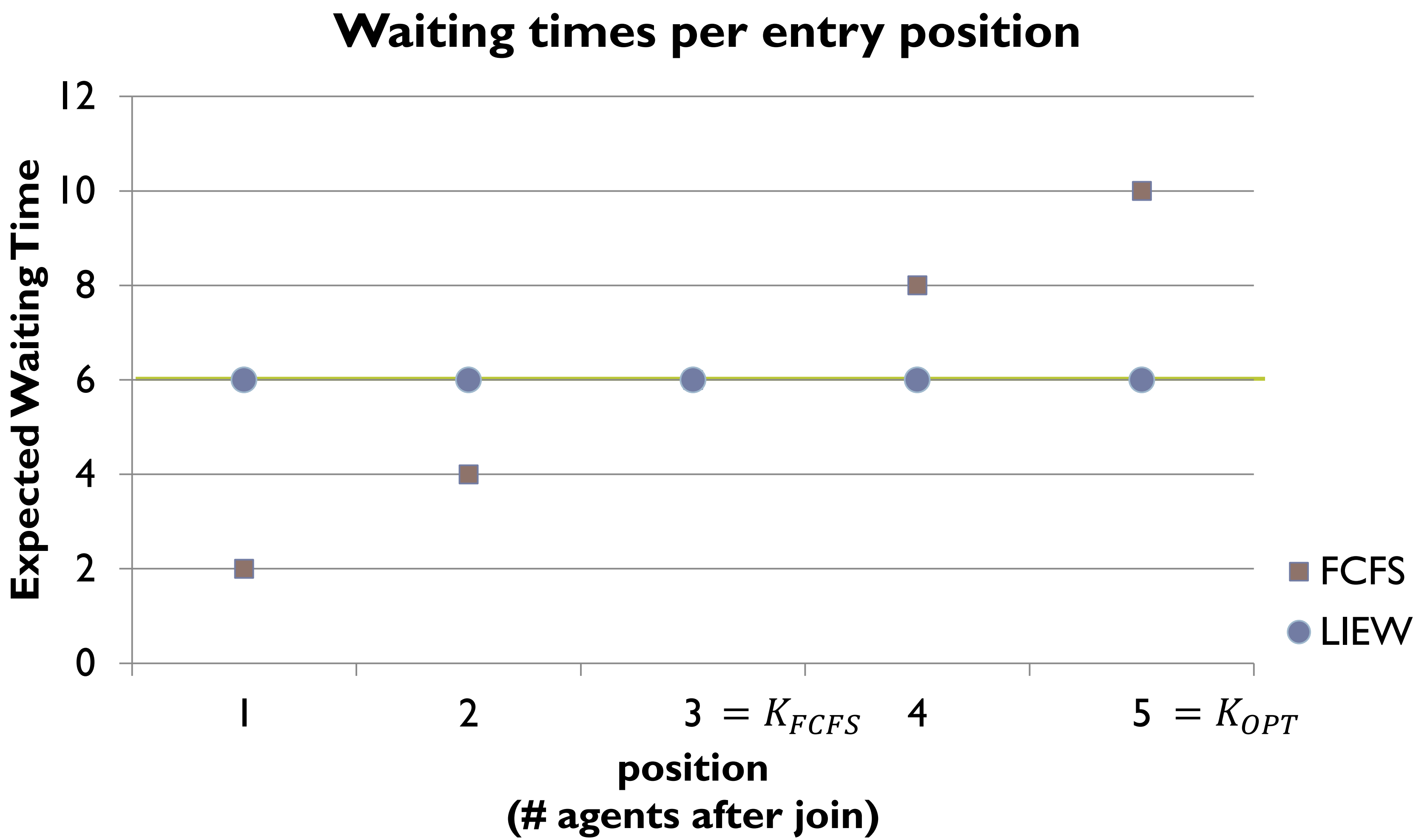
- To achieve the upper bound, no position can have an expected wait that is greater than average (expected for random position)
- Therefore, to get the optimal incentive compatible buffer-queue policy we need all position to have the same expected wait
- This requires a new queueing policy, which we name the Load Independent Expected Wait (LIEW) queueing policy

A New Queueing Policy

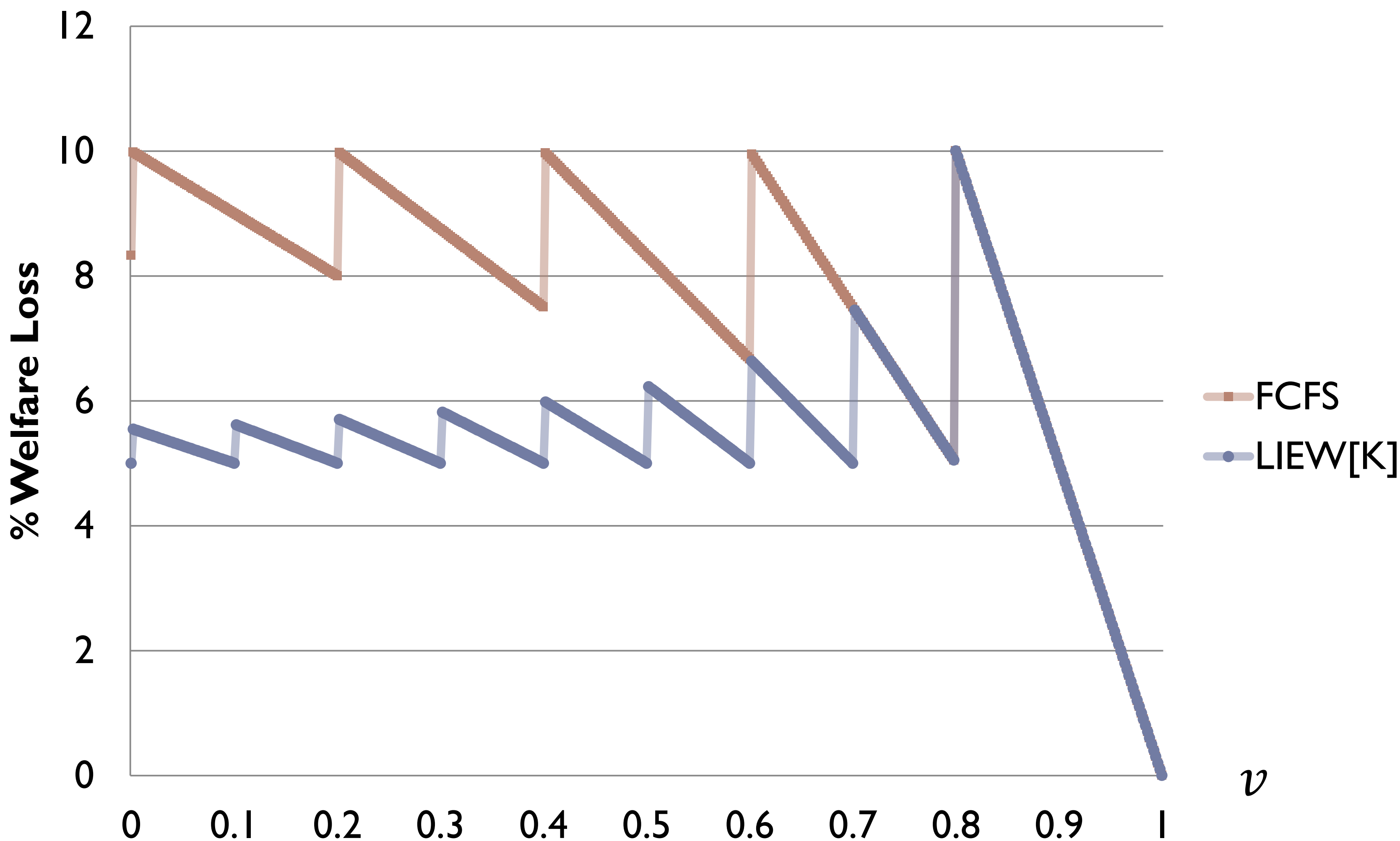
Definition: A $\langle K, \varphi \rangle$ queue policy is a $LIEW[K]$ policy if for all $k \le K$ the expected wait at join is $\mathrm{w}_k = \frac{K+1}{2p}$.

Theorem: The LIEW policy is the optimal buffer-queue policy.

LIEW Buffer-Queue



Mechanisms – Welfare Loss Comparison



$c = 0.1, p = 1/2$

Designing the LIEW policy

Which assignment probabilities generate a LIEW queue?

- Let $\vec{w} = (w_1, \dots w_K)$
- Setting $\varphi = \begin{pmatrix} 1 & & & \\ 1 & 0 & & \\ \vdots & & \ddots & \\ 1 & 0 & \cdots & 0 \end{pmatrix}$ gives FCFS and $\vec{w} \cdot p = (1,2,..,K)$
- Setting $\varphi = \begin{pmatrix} 1 & & & \\ 0 & 1 & & \\ \vdots & & \ddots & \\ 0 & 0 & \cdots & 1 \end{pmatrix}$ gives LCFS and $\vec{w} \cdot p = (K,..,2,1)$
- Take φ to be "in between"

Designing the LIEW Queue

Start with w_1:

$$\varphi = \begin{pmatrix} 1 & & \\ 1 & 0 & \\ 1 & 0 & 0 \end{pmatrix}$$

$$\vec{w} \cdot p = (1, 2, 3)$$

Designing the LIEW Queue

Start with w_1:

$$\varphi = \begin{pmatrix} 1 & & \\ 0 & 1 & \\ 0 & 1 & 0 \end{pmatrix}$$

$$\vec{w} \cdot p \; = (3, 1, 2)$$

Designing the LIEW Queue

Start with w_1:

$$\varphi = \begin{pmatrix} 1 & & \\ \frac{1}{2-p} & \frac{1-p}{2-p} & \\ 0 & 1 & 0 \end{pmatrix}$$

$$\vec{w} \cdot p = (2, 2 - \frac{1}{2-p}, 2 + \frac{1}{2-p})$$

Designing the LIEW Queue

Shifting continuously from FCFS to LCFS to get the LIEW assignment probabilities:

$$\varphi^{LIEW[3]} = \begin{pmatrix} 1 & & \\ \dfrac{1}{2-p} & \dfrac{1-p}{2-p} & \\ 0 & \dfrac{1}{2} & \dfrac{1}{2} \end{pmatrix}$$

$$\vec{w} \cdot p \ = (2, 2, 2)$$

- Achieves $K = 3$ when agents are willing to wait $\overline{\mathrm{w}} = \frac{1-v}{c} = \frac{2}{p}$

LIEW Policy - Issues

- Complicated
- Agent's belief matters
  - An agent will not join the queue if his belief is that following agents will join as well
- Parameter dependent
  - Designer needs to know p and set K^*
  - Performs poorly when if parameters are wrong

Robust Buffer-Queue

Optimize the buffer-queue while maintaining robustness

- *Safe for agents:* incentive compatible for agents to join, regardless of their belief about the waiting list
  - Implies that agents do not regret joining if other agents join after them

- *Simple for designer:* a single scalable mechanism that applies to multiple environments

A Policy for any Buffer-Queue Size

- *Scalable policy*: A scalable policy $\langle\varphi\rangle$ is given by weights

  $\{v_i\}_{i\leq\infty}$ such that $\varphi(k,i) = \frac{v_i}{\sum_{j\leq k} v_j}$

- "Same" randomization for any queue size.
- The mechanism can react to the parameters of the

envioronment (v, c) only by adjusting the maximal size of the buffer-queue K.

Belief Free IC

- A policy $\langle\varphi\rangle$ with maximal size K *is Belief free IC* if for any belief σ on following types

$$w_k^\sigma \leq \bar{w}$$

  - Dominant strategy to report truthfully – agents are willing to decline mismatch regardless of their belief on the joining of future agents
  - Satisfied by FCFS
- *Lemma:* a policy is BF-IC if and only if it is ex-post BF-IC, that is the expected wait for an agent in position i out of k is

$$w_{[i,k]}^\sigma \leq \bar{w}$$

Comparing Mechanisms

- We want to compare mechanisms without assumptions on the environment
- Let $\mathcal{K}_\varphi(\overline{w})$ be the maximal K for which φ is BF-IC
- *Definition:* $\langle\varphi\rangle$ dominates $\langle\psi\rangle$ if there is w_0 such that for every $w \leq w_0$ we have $\mathcal{K}_\varphi(w) \geq \mathcal{K}_\varphi(\psi)$ with strict inequality for some $w \leq w_0$.
  - Better, or at least of length $\mathcal{K}_\varphi(w_0)$
  - Increasing buffer size is most important when buffer is small
  - Need to be optimal for $K + 1$ conditional on being optimal for $1, \ldots, K$

SIRO - Service In Random Order

- Equal probability to each waiting agent: $\varphi(k,i) = \frac{1}{K}$

$$\varphi = \begin{pmatrix} 1 & & & & \\ 1/2 & 1/2 & & & \\ 1/3 & 1/3 & 1/3 & & \\ 1/4 & 1/4 & 1/4 & 1/4 & \\ \vdots & \vdots & \vdots & \vdots & \ddots \end{pmatrix}$$

- Maximizing incentives for last agent to join, while minimizing regret for agents already on the queue

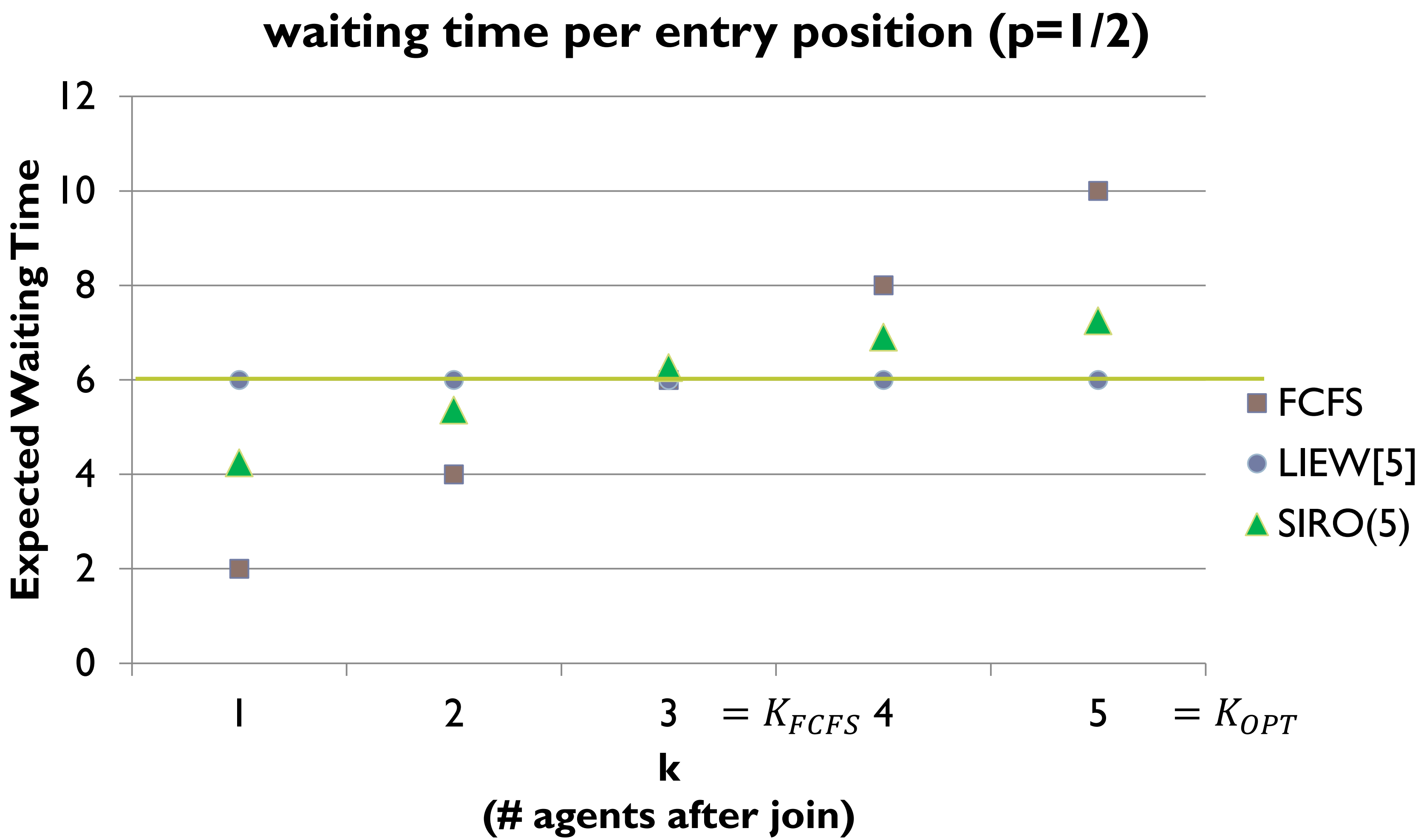
SIRO – Scalable Optimal

Theorem: SIRO is the unique undominated scalable policy

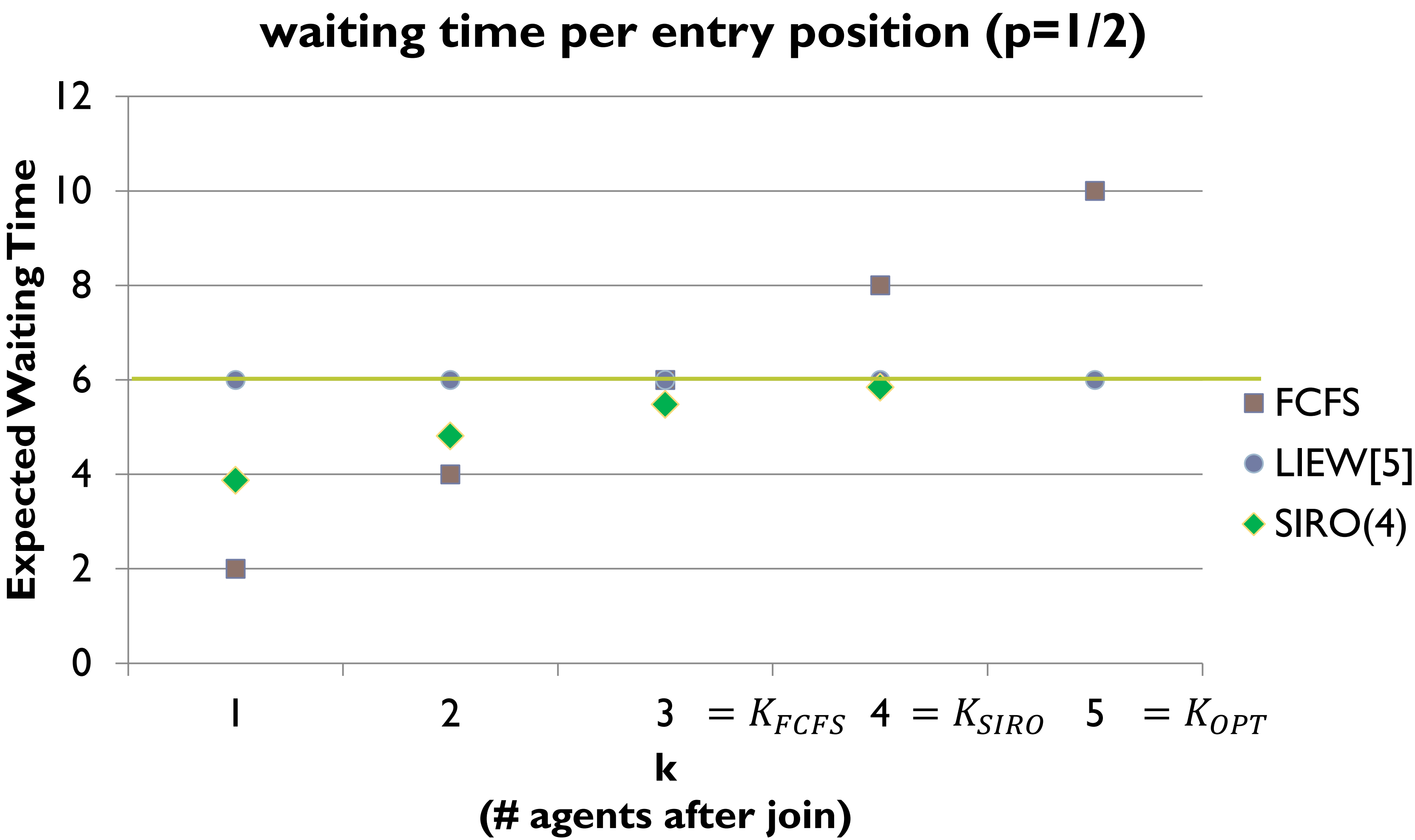
Proof sketch:

- Optimize for $K+1$ conditional on being optimal for K
- Minimize wait for new $K+1$ position $w^{\sigma}_{[K+1,K+1]} \leq \bar{w}$
- For Bf-IC, wait for other positions is $w^{\sigma}_{[i,K+1]} \leq \bar{w}$
  $\Rightarrow$ treat all agents in the buffer-queue the same (ex-post)

SIRO with $K = 5$

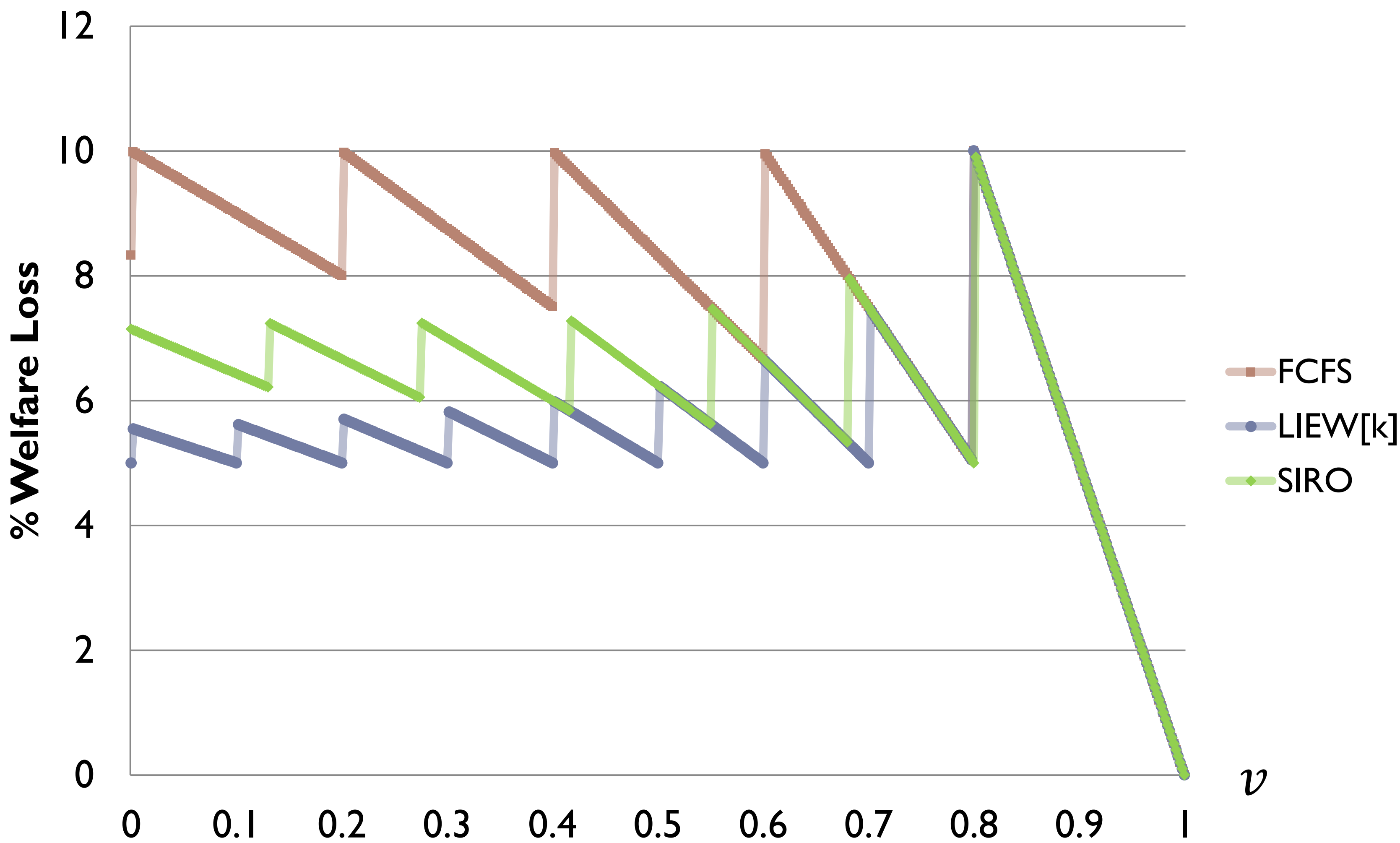

waiting time per entry position (p=1/2)
Expected Waiting Time
12
10
8
6
4
2
0
1
2
3
$= K_{FCFS}$
4
5
$= K_{OPT}$
k
(# agents after join)
FCFS
LIEW[5]
SIRO(5)

SIRO with $K = 4$



waiting time per entry position (p=1/2)
Expected Waiting Time
12
10
8
6
4
2
0
1
2
3
4
5
$= K_{FCFS}$
$= K_{SIRO}$
$= K_{OPT}$
k
(# agents after join)
FCFS
LIEW[5]
SIRO(4)

Mechanism Comparison



$c = 0.1, p = 1/2$

SIRO - properties

- Parameter free
- Simple
  - No positions in the queue
- No need to restrict joining when agents are symmetric
- Ex post BF-IC – waiting agents can be offered a B
- SIRO achieves better welfare that FIFO, for any parameters and beliefs

Heterogeneous values

- % Welfare loss under mechanisms when $v \sim U[0,1]$:

c	FCFS	LIEW[4]	SIRO
.05	2.50	3.82	2.25
.10	5.00	4.67	4.37
.15	7.43	6.66	6.53
.20	9.65	9.29	8.69
.25	12.50	11.74	10.86
.30	13.57	13.57	13.05
.35	15.19	15.19	15.19
.40	17.50	17.50	17.50
.45	20.68	20.68	20.68
.50	25.00	25.00	25.00

Conclusion

- Welfare in congested waiting lists
  - Match quality matters, waiting time cancels out
  - Need to incentivize agents to decline mismatched items

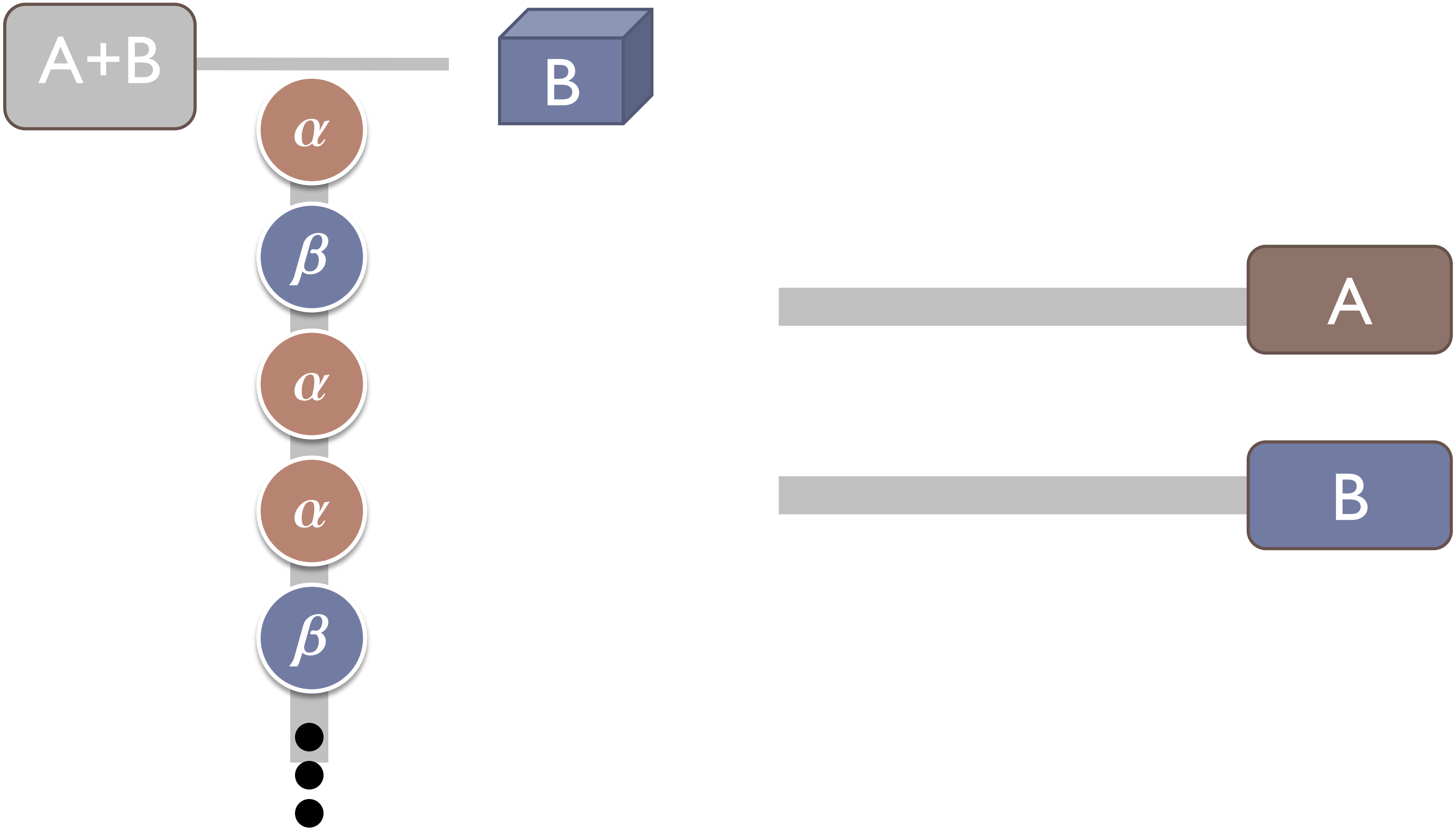
- Tractable analysis by tracking only agents who declined items
  - Closed form solutions for the stylized model
  - Solving for richer environments by simulation methods

Conclusion

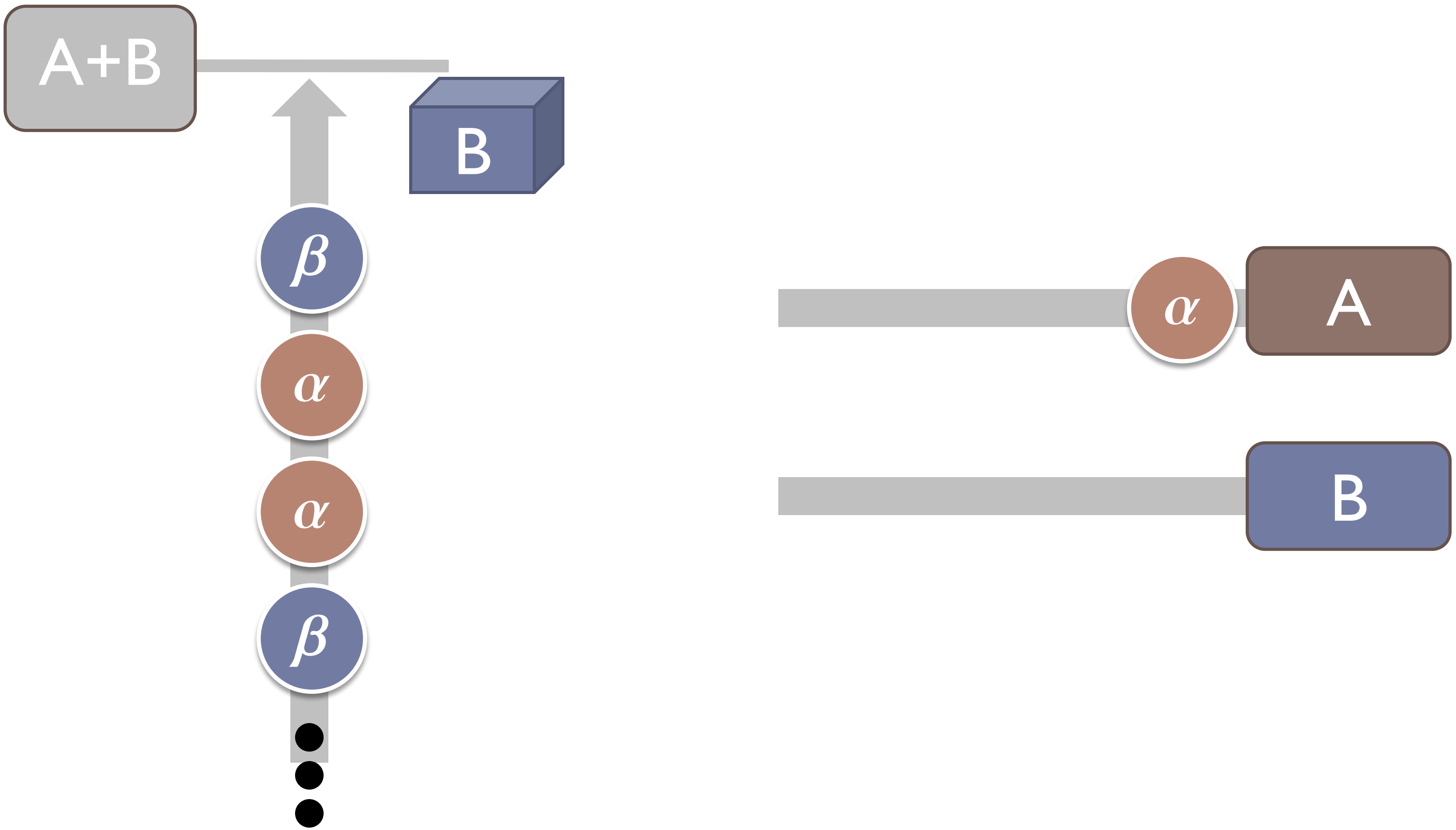
- The SIRO buffer-queue policy
  - Lottery gives higher incentives for the marginal agent to decline a mismatched item,
    reducing waiting time fluctuation and misallocation
  - Safe and parameter free

  - Thank you.

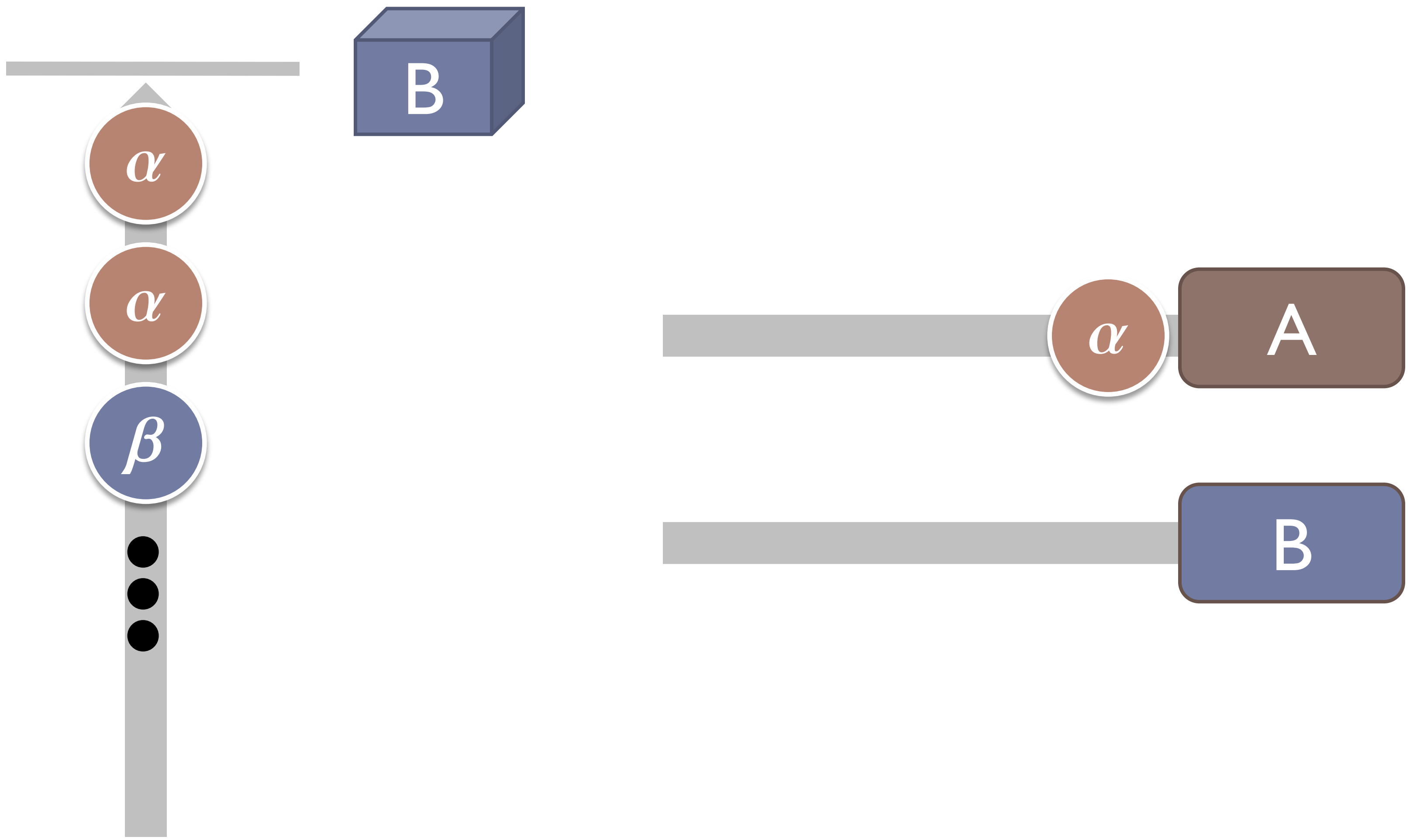
FCFS, t=1



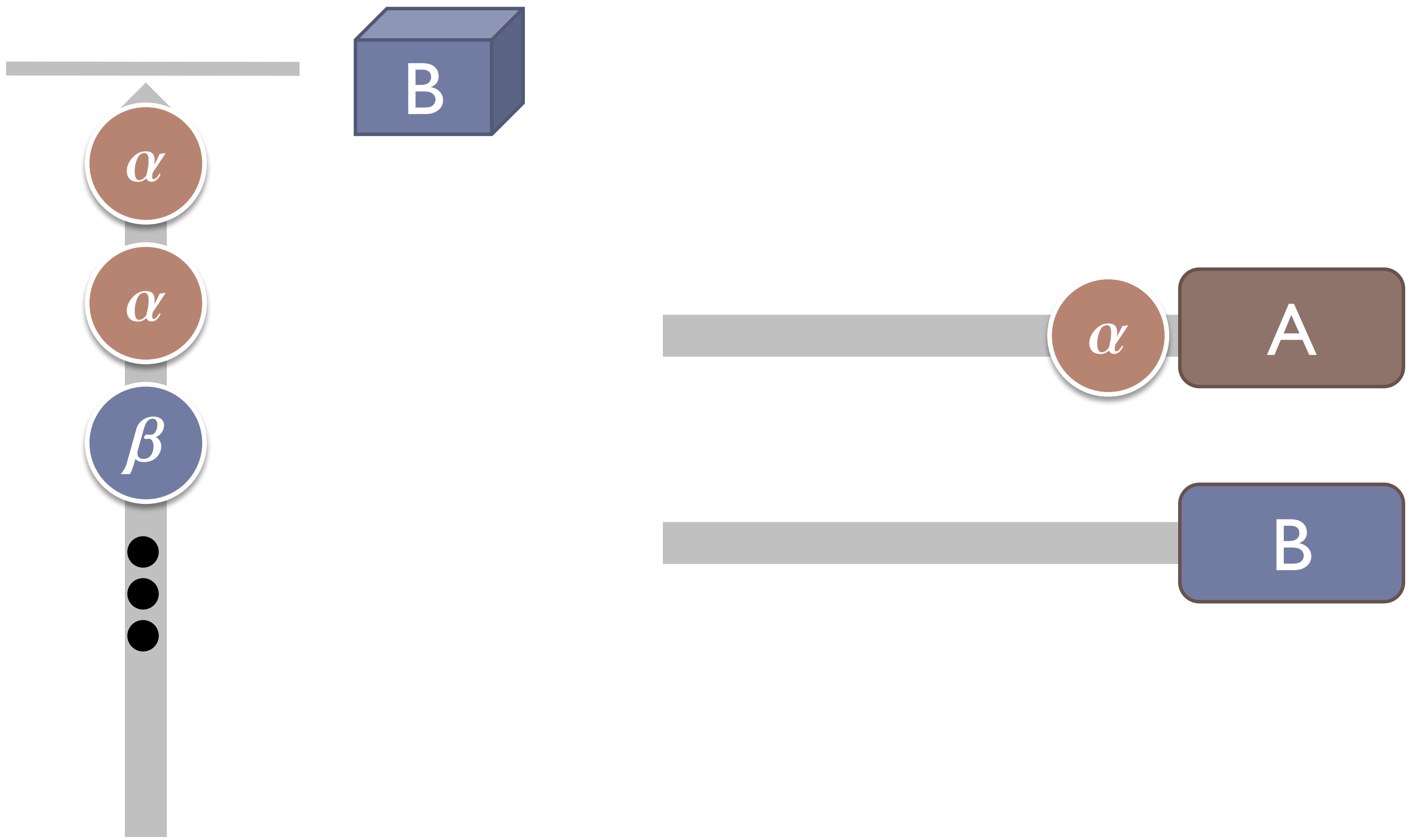
FCFS, t=1



FCFS, t=2

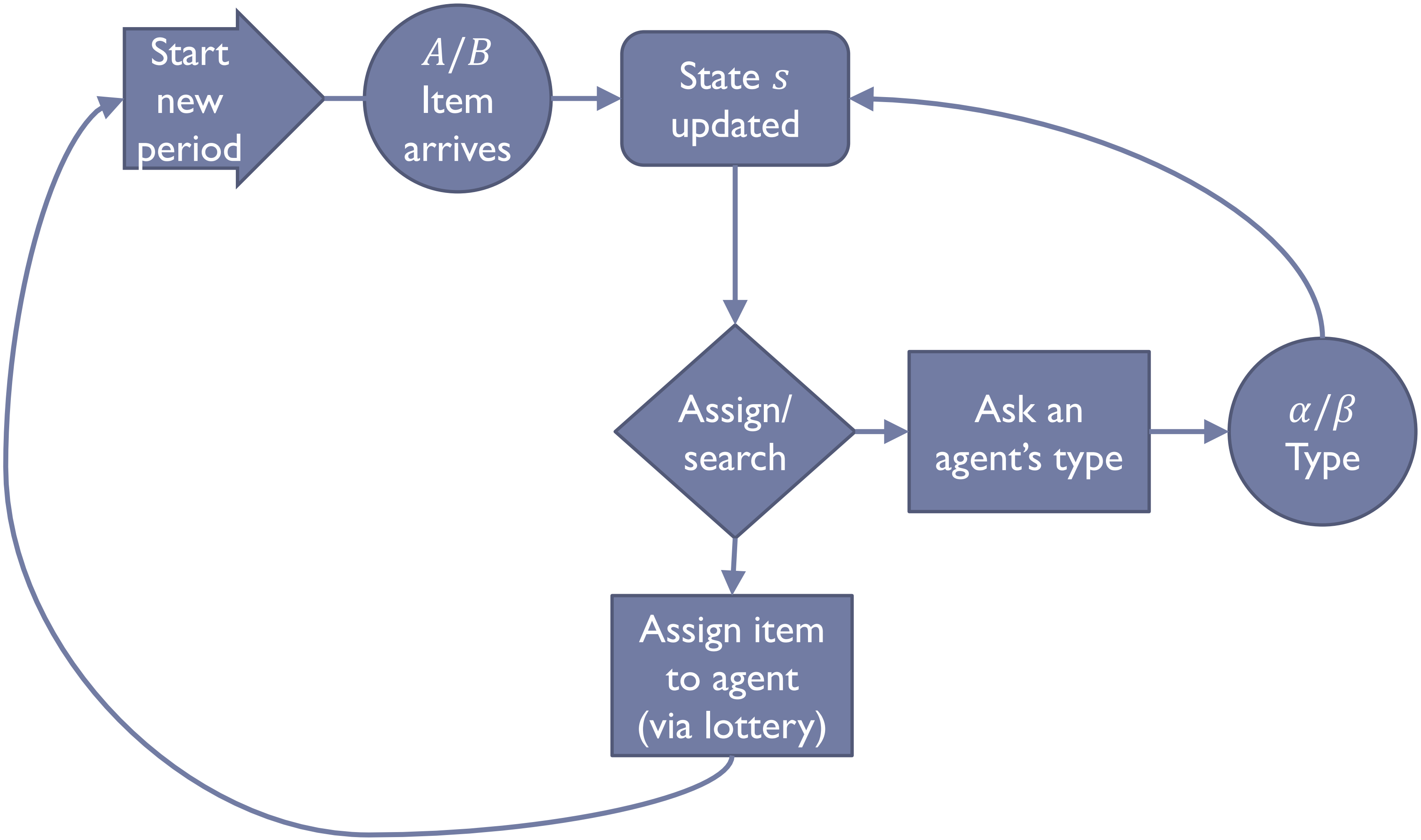


FCFS, t=2



Back

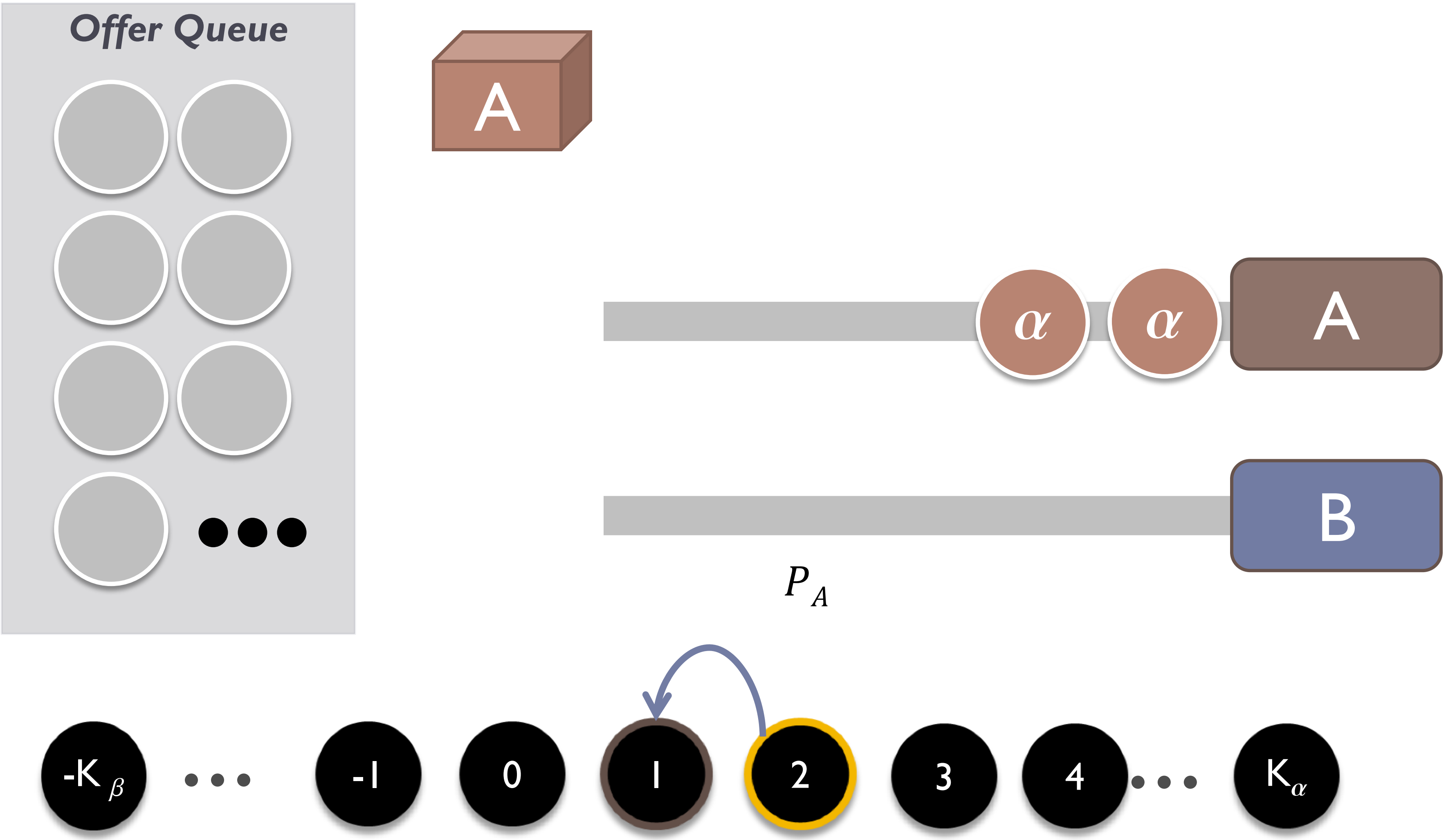
Dynamic Direct Mechanism



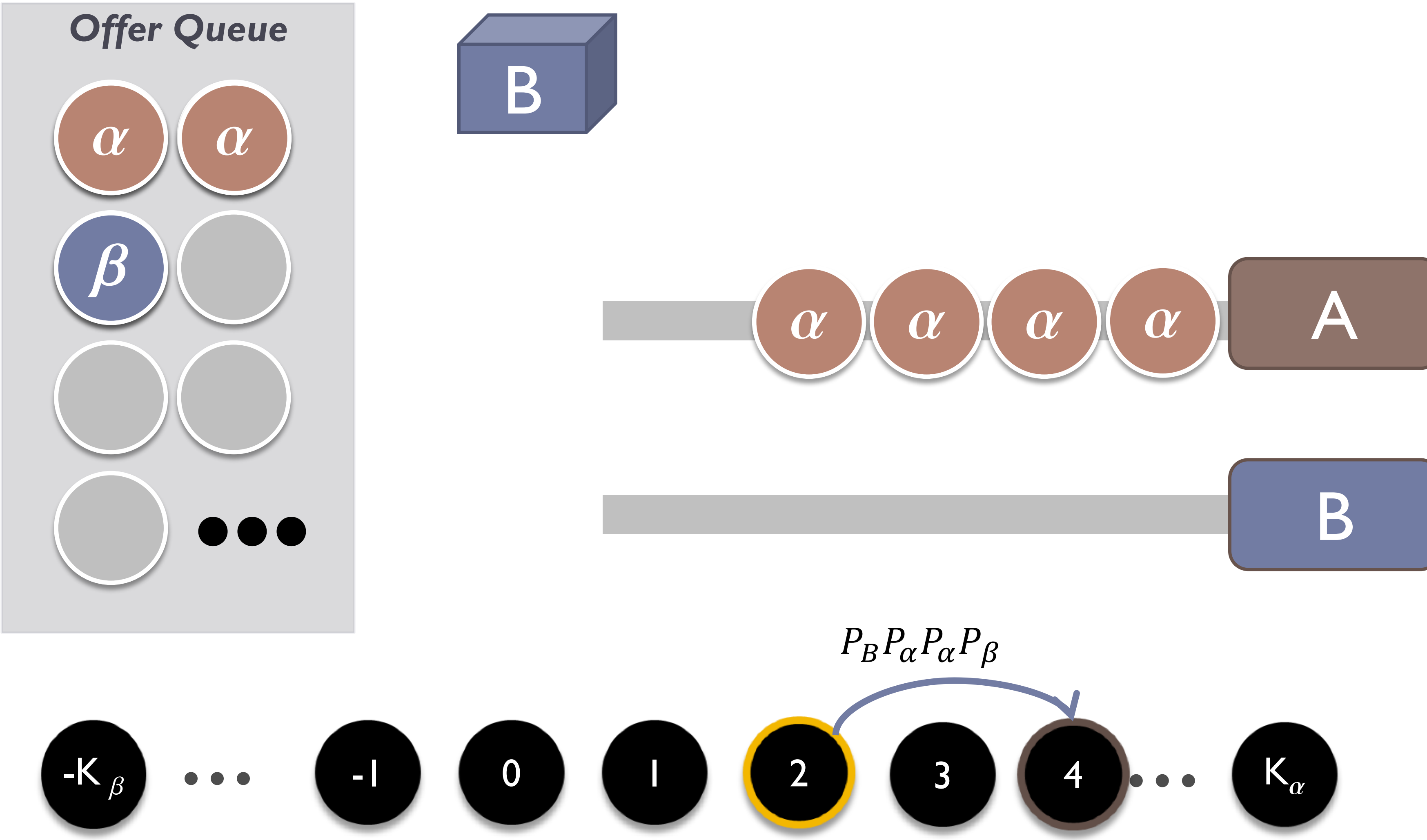
Buffer-Queue Mechanism

- A dynamic direct mechanism such that:
  - Ask agents only if there is no matching agent for the current item (no asking in advanced)
  - Always assign waiting agents to their preferred item
  - Queue based: offers are generated by positions on a queue

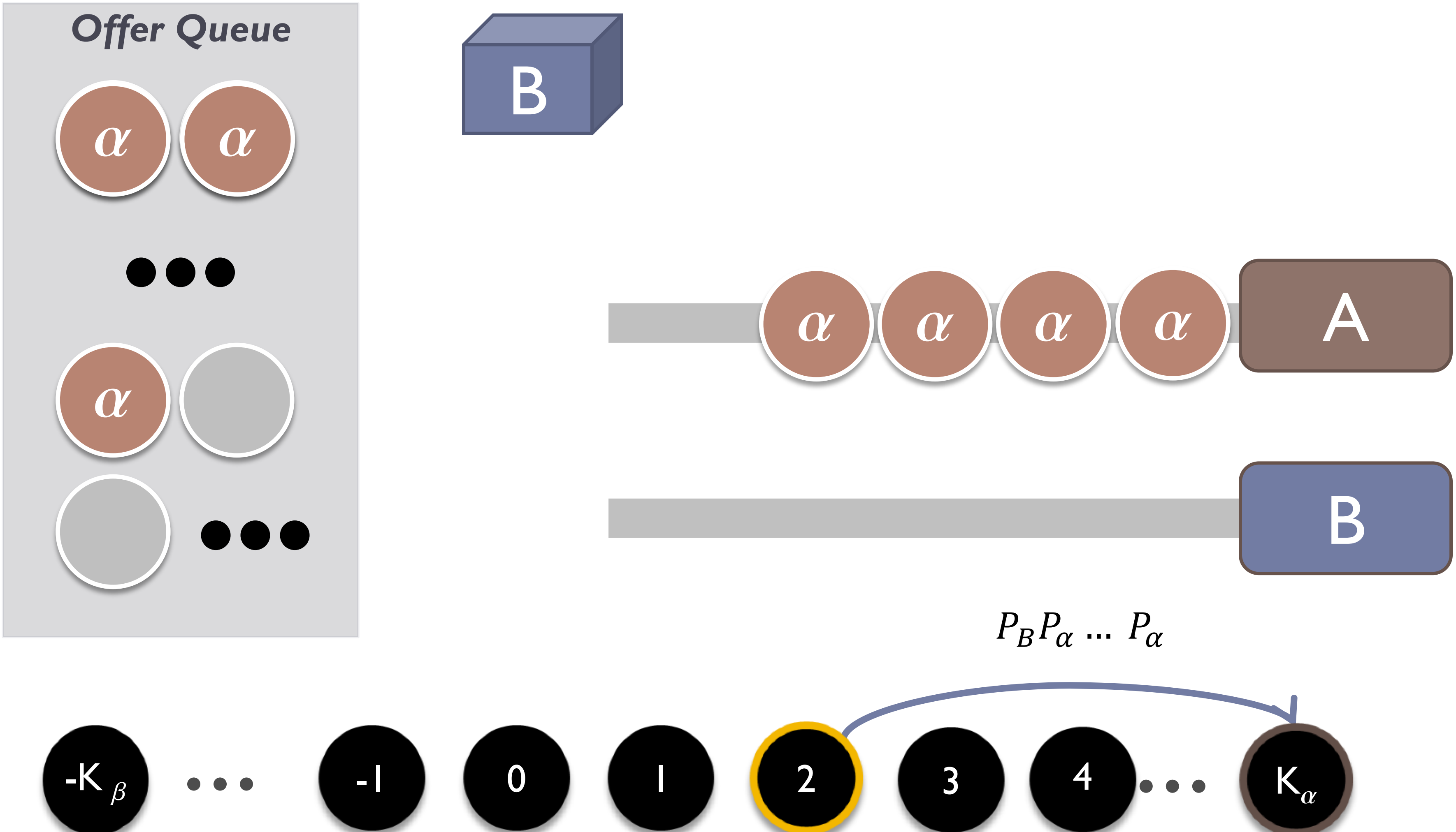
System Dynamics -transitions



System Dynamics -transitions



System Dynamics -transitions



Back