Algorithms used by CDNs

Stable Marriage Algorithm

Consistent Hashing

Dating Scenario

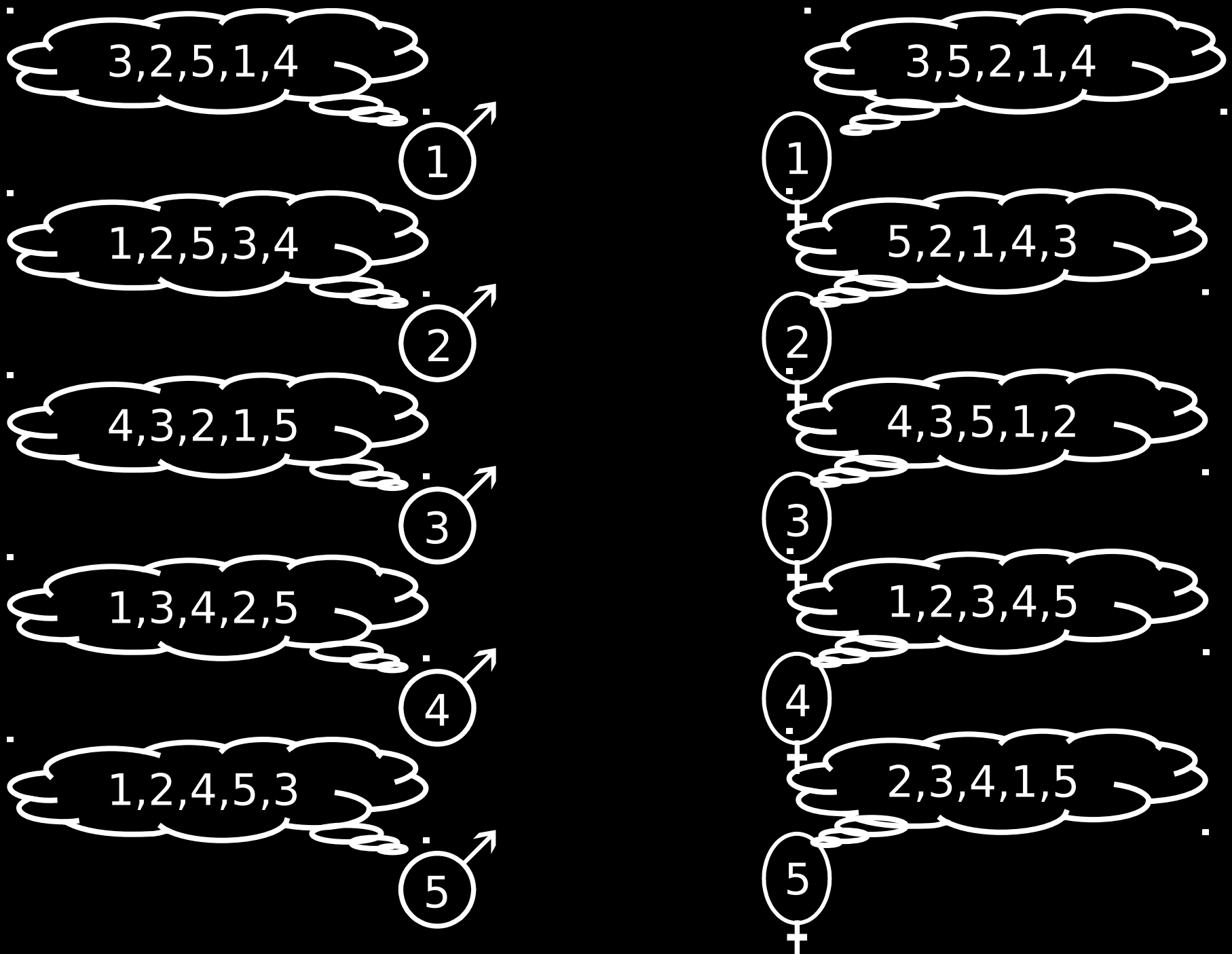
There are n boys and n girls

Each girl has her own ranked preference list of all the boys

Each boy has his own ranked preference list of the girls

The lists have no ties

Question: How do we pair them off?


3,2,5,1,4
1
1,2,5,3,4
2
4,3,2,1,5
3
1,3,4,2,5
4
1,2,4,5,3
5
3,5,2,1,4
1
5,2,1,4,3
2
4,3,5,1,2
3
1,2,3,4,5
4
2,3,4,1,5
5

Rogue Couples

Suppose we pair off all the boys and girls

Now suppose that some boy and some girl prefer each other to the people to whom they are paired

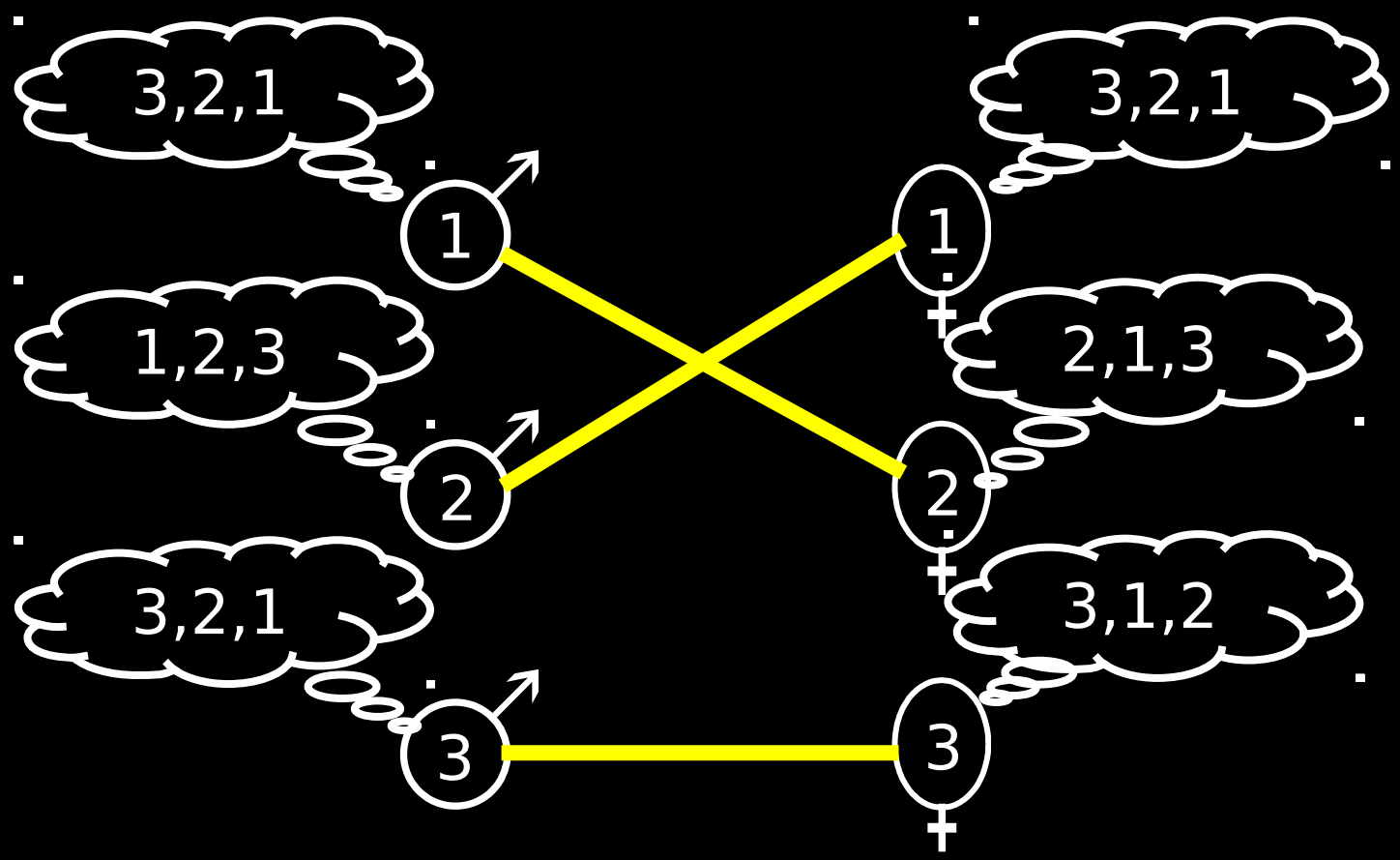
They will be called a rogue couple

Stable Pairings

A pairing of boys and girls is called stable if it contains no rogue couples

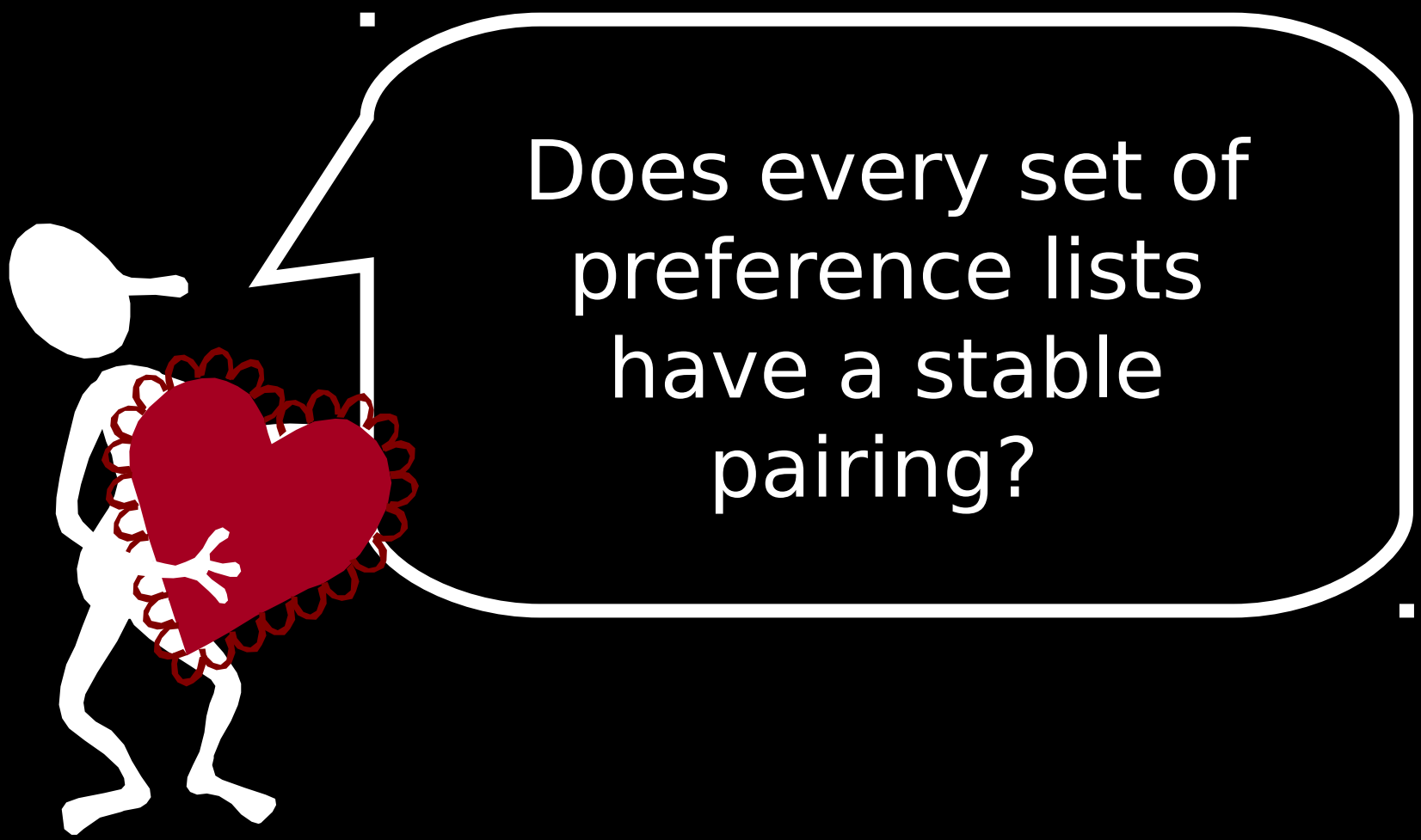
Stable Pairings

A pairing of boys and girls is called stable if it contains no rogue couples

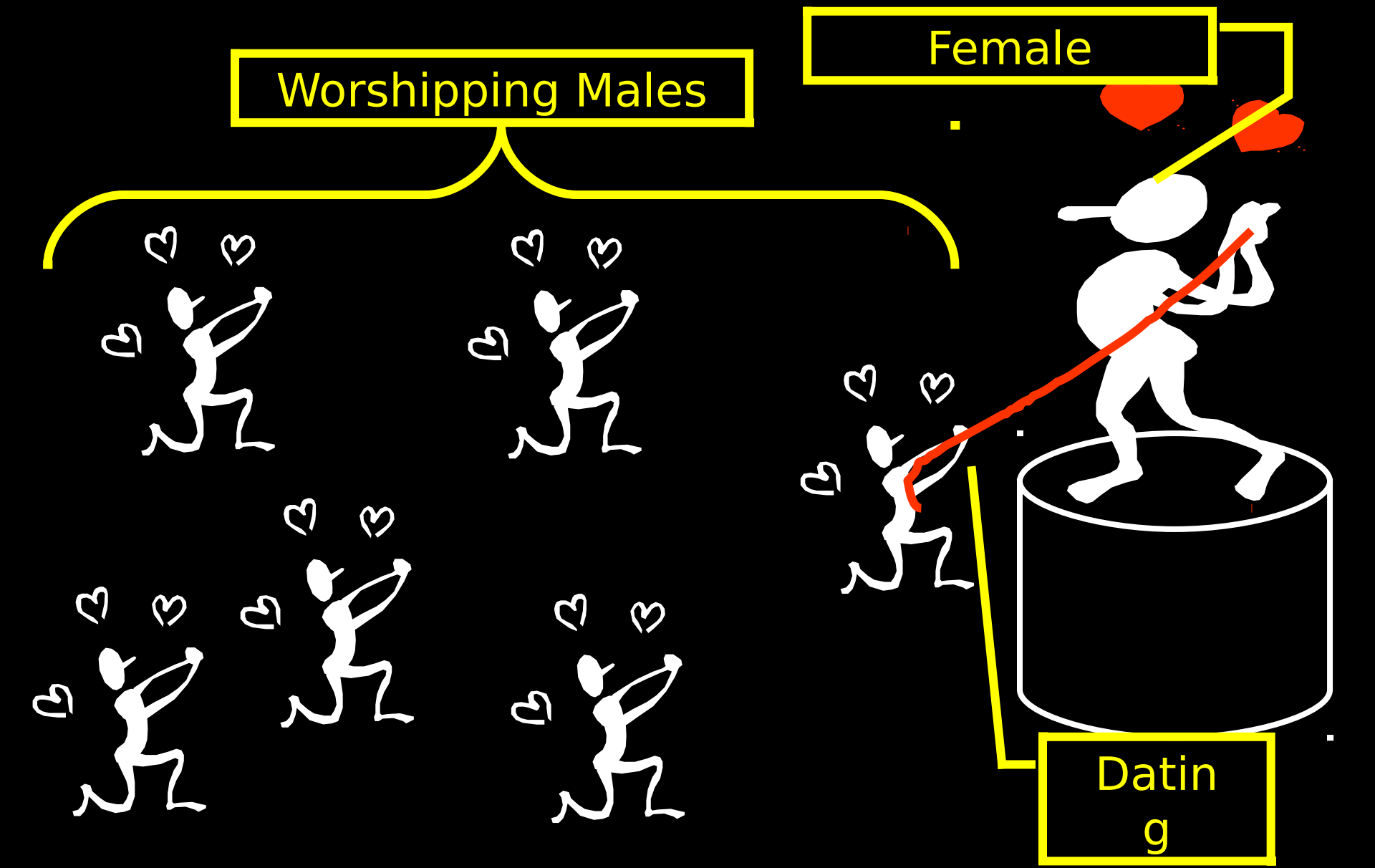


Given a set of preference lists, how do we find a stable pairing?

Better Question:



The Traditional Marriage Algorithm



The Traditional Marriage Algorithm

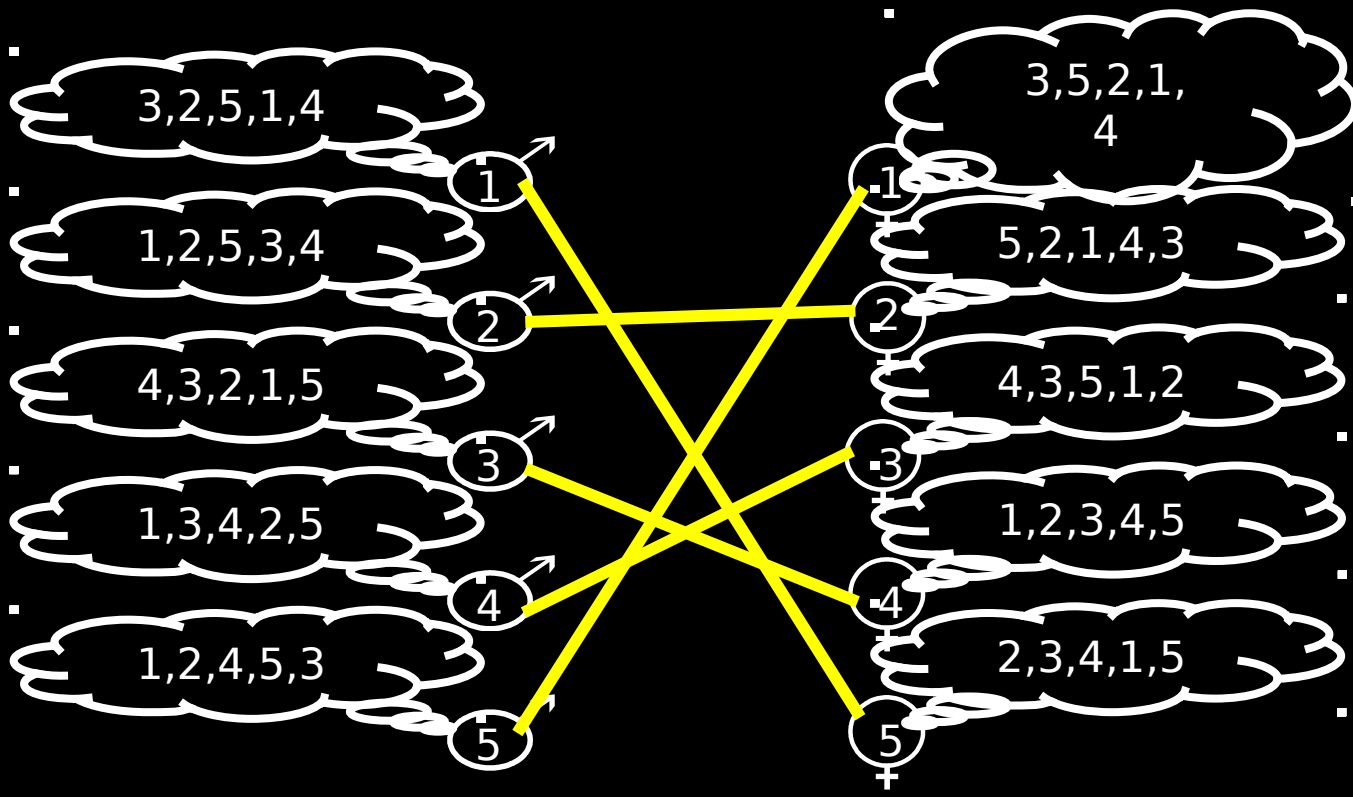
Morning

- Each girl stands on her balcony
- Each boy proposes to the best girl on his list who has not rejected him yet

Afternoon (for girls with at least one suitor)

- To today's best: "Let's just date for now, come back tomorrow"
- To any others: "No, I will never marry you, don't come back"

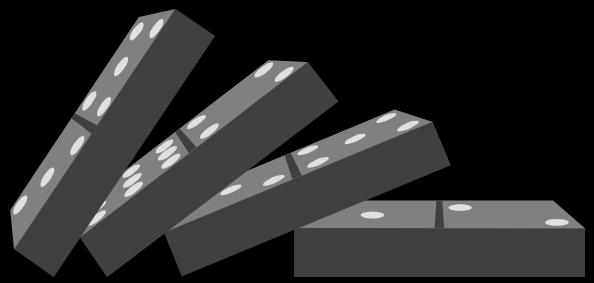
If no boys get a "No", each girl marries boy she is dating


3,2,5,1,4
1,2,5,3,4
4,3,2,1,5
1,3,4,2,5
1,2,4,5,3
1
2
3
4
5
3,5,2,1,
4
5,2,1,4,3
4,3,5,1,2
1,2,3,4,5
2,3,4,1,5
1
2
3
4
5

Improvement Lemma:

If a girl is dating someone, then she will always be dating someone at least as good until she is married

- She would only let go of him in order to date someone better
- She would only let go of that guy for someone even better
- She would only let go of that guy for someone even better
- AND SO ON…



Lemma: No boy can be rejected by all the girls

Proof (by contradiction):

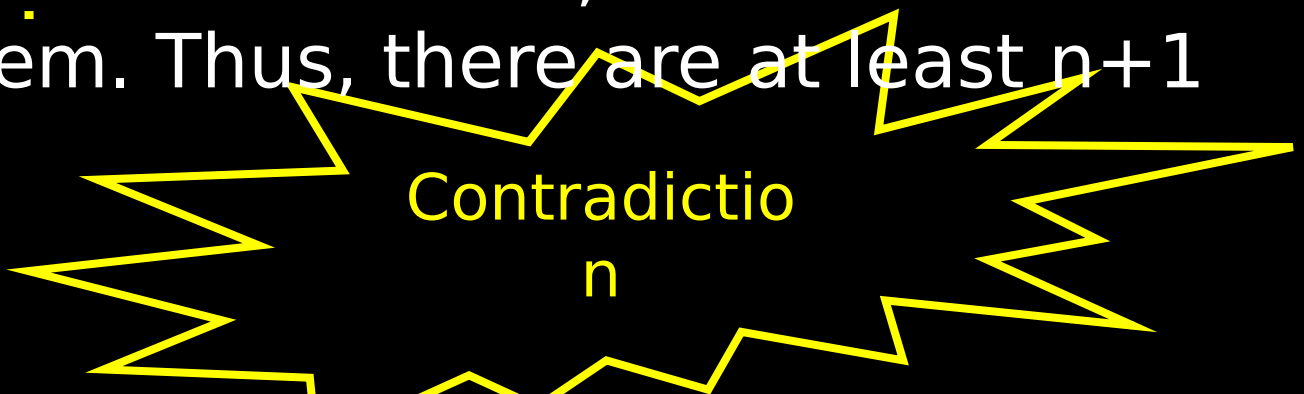
Suppose boy b is rejected by all the girls

At that point:

Each girl must have a suitor other than b

(By Improvement Lemma, once a girl has a suitor she will always have at least one)

The n girls have n suitors, and b is not among them. Thus, there are at least n+1 boys



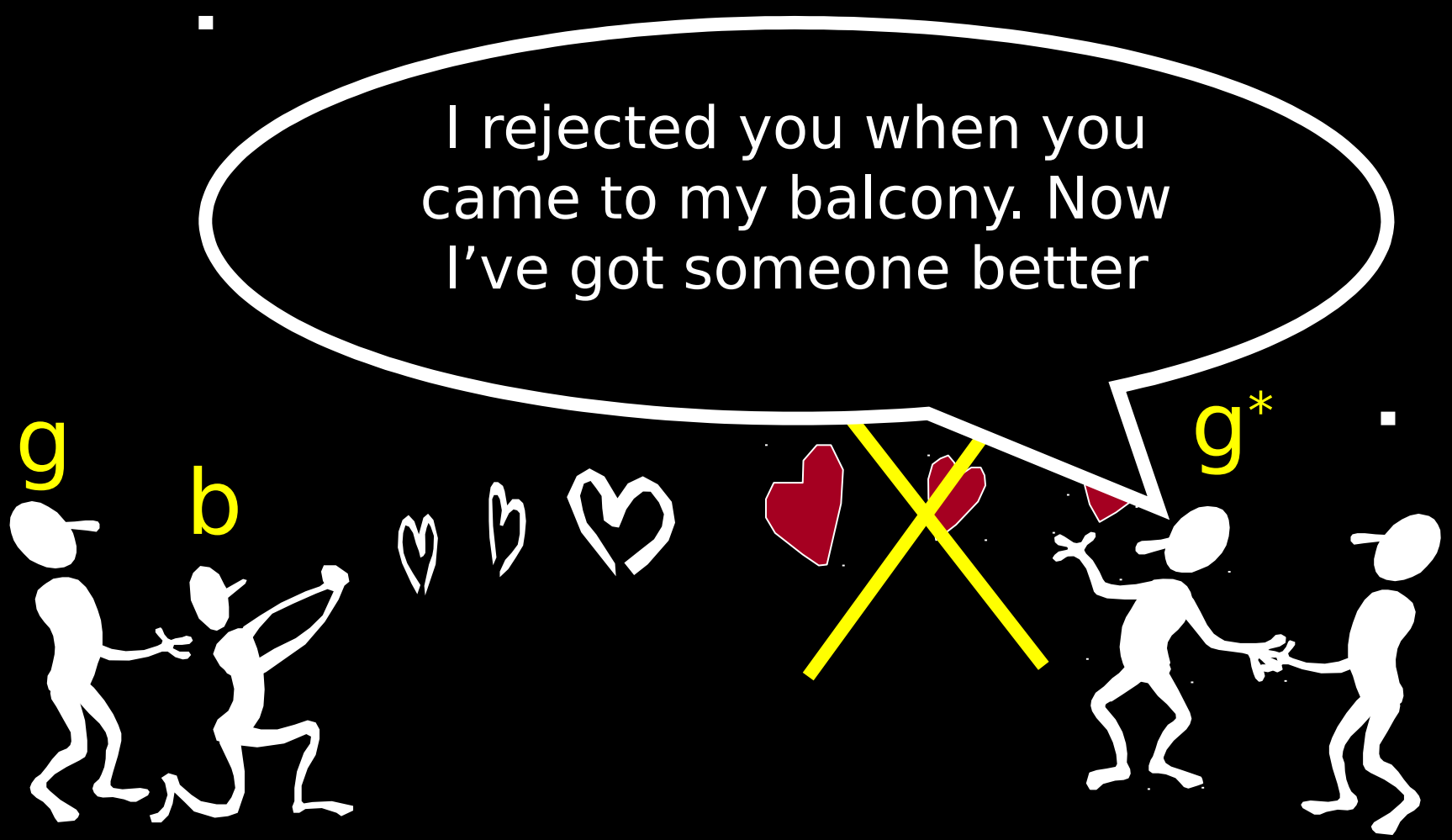
Theorem: The TMA always terminates in at most n^2 days

A “master list” of all n of the boys lists starts with a total of n x n = n^2 girls on it

Each day (until the last) at least one boy gets a “No”, so at least one girl gets crossed off the master list

Therefore, the number of days is bounded by the original size of the master list

Theorem: The pairing T produced by TMA is stable



Opinion Poll

The Optimal Girl

A boy's optimal girl is the highest ranked girl for whom there is some stable pairing in which the boy gets her

Presumably, she might be better than the girl he gets in the stable pairing output by TMA

The Pessimal Girl

A boy’s pessimal girl is the lowest ranked girl for whom there is some stable pairing in which the boy gets her

Dating Heaven and Hell

A pairing is male-optimal if every boy gets his optimal mate. This is the best of all possible stable worlds for every boy simultaneously

A pairing is male-pessimal if every boy gets his pessimal mate. This is the worst of all possible stable worlds for every boy simultaneously

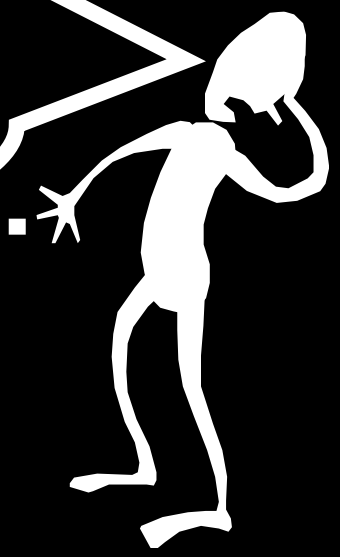
Dating Heaven and Hell

A pairing is female-optimal if every girl gets her optimal mate. This is the best of all possible stable worlds for every girl simultaneously

A pairing is female-pessimal if every girl gets her pessimal mate. This is the worst of all possible stable worlds for every girl simultaneously

The Naked Mathematical Truth!

The Traditional Marriage Algorithm always produces a male-optimal, female-pessimal pairing



Theorem: TMA produces a male-optimal pairing

Suppose, for a contradiction, that some boy gets rejected by his optimal girl during TMA

Let t be the earliest time at which this happened

At time t, boy b got rejected by his optimal girl g because she said “maybe” to a preferred b*

By the definition of t, b* had not yet been rejected by his optimal girl

Therefore, b* likes g at least as much as his optimal

Since g is the optimal girl for b, there must exist a stable pairing S in which b and g are married, but ...

b* wants g more than his wife in S:
g is at least as good as his best and he does not have her in stable pairing S

g wants b* more than her husband in S:
b is her husband in S and she rejects him for b* in TMA

Contradiction

Theorem: The TMA pairing, T, is female-pessimal

We know it is male-optimal. Suppose there is a stable pairing S where some girl g does worse than in T

Let b be her mate in T

Let b* be her mate in S

By assumption, g likes b better than her mate in S

b likes g better than his mate in S (we already know that g is his optimal girl)

Therefore, S is

Contradiction

The largest, most successful dating service in the world uses a computer to run TMA!

Assigning Name Servers to Clusters

- Every resolving (name server, traffic type) pair is a “boy”; each boy comes with a predicted load
- Every Akamai cluster is a “girl”
- Preferences are based on measured packet loss and latency to the name server’s core point and the cost to serve traffic from the cluster
- There are many more boys than girls, but each girl can marry multiple boys! (Up to the capacity of the cluster)
- Marriage dictates which clusters should serve which clients

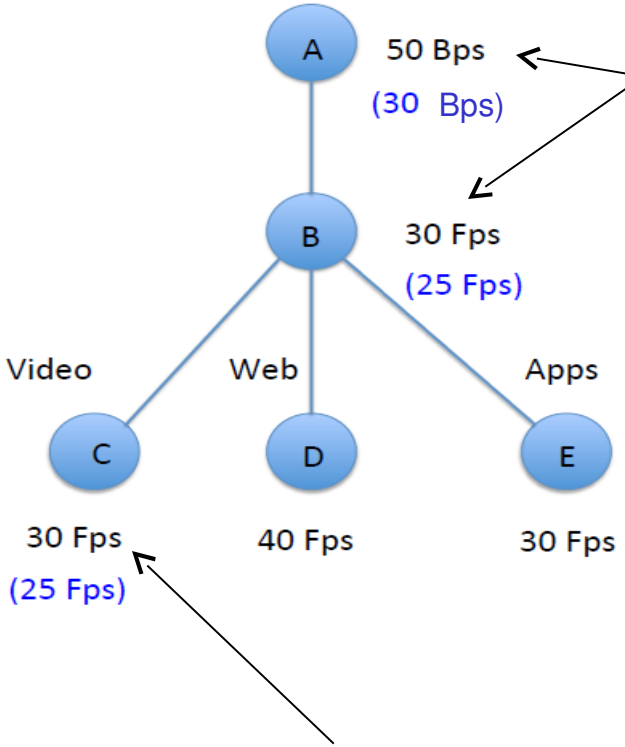
Why not find a maximum matching?

- The stable marriage algorithm is much faster in practice.
- But locally optimal as opposed to globally optimum.
- Warning: changing a **single** preference can cause ALL boys and girls to switch to new partners
- Before computing a new solution, boys increase preference for previous partner

Extension: Multiple Resources

- Each server has several resources (e.g., processor, disk access, network bandwidth) each with finite capacity.
- Each type of traffic (e.g., video, web, secure) puts different demands on each resource.
- Matching must not exceed the capacity of any of the individual resources.

Extension: Resource Trees



traffic class limit (in black)

Aggregate cluster capacity (in black)

- Assumption: resource constraints can be modeled hierarchically
- Each class of traffic puts load on a path to a leaf (video class load shown in blue)
- Stable marriage algorithm can be extended to respect these constraints.

Hashing

Universe U of all possible objects, set B of buckets.

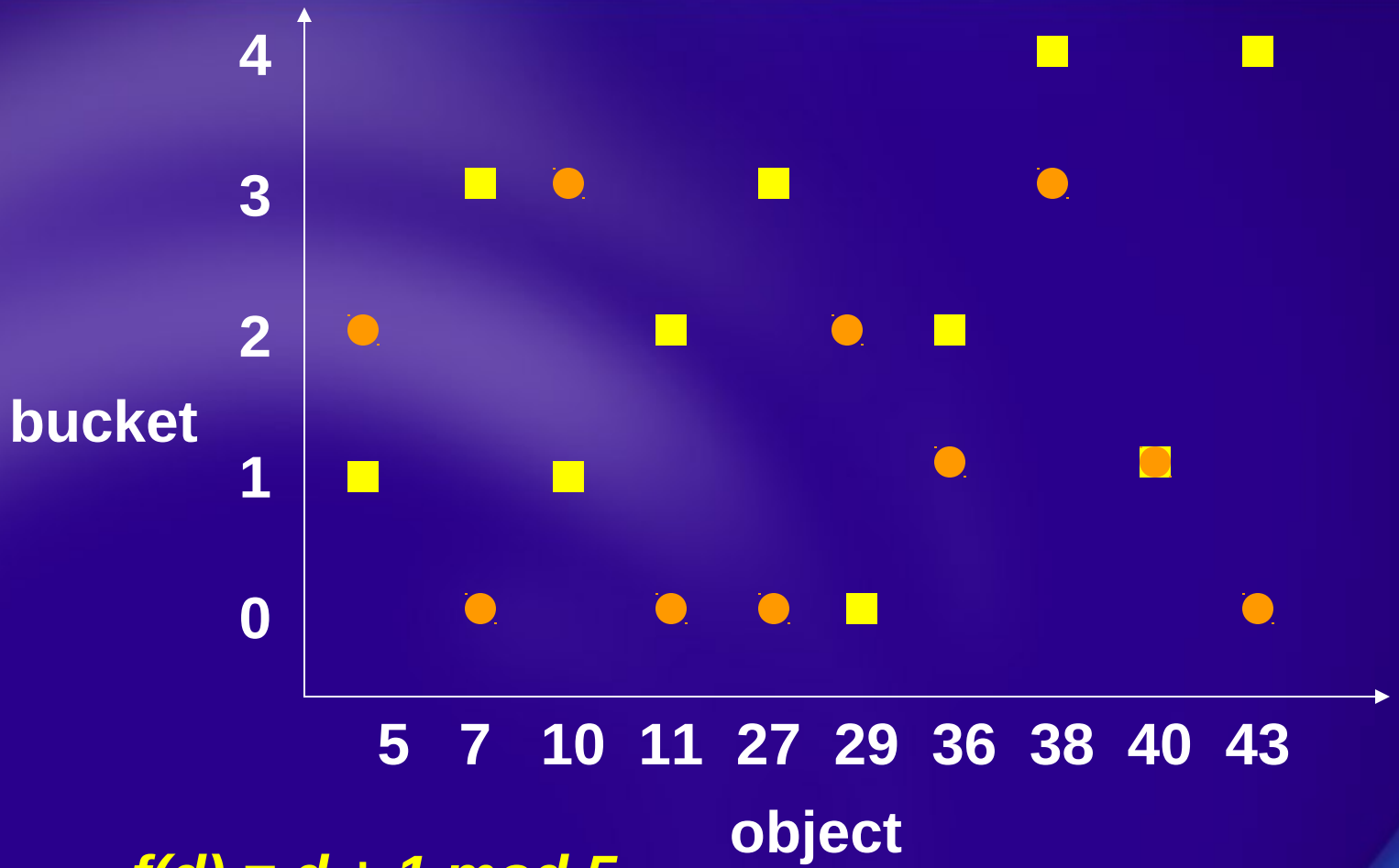
object: set of web objects with same serial number
bucket: web server

Hash function $h: U \rightarrow B$

Assigns objects to buckets

E.g., $h(x) = (((a\,x + b) \bmod P) \bmod |B|)$, where
P is prime, $P > |U|$
a,b chosen uniformly at random from Z_P
x is a serial number

Difficulty changing number of buckets

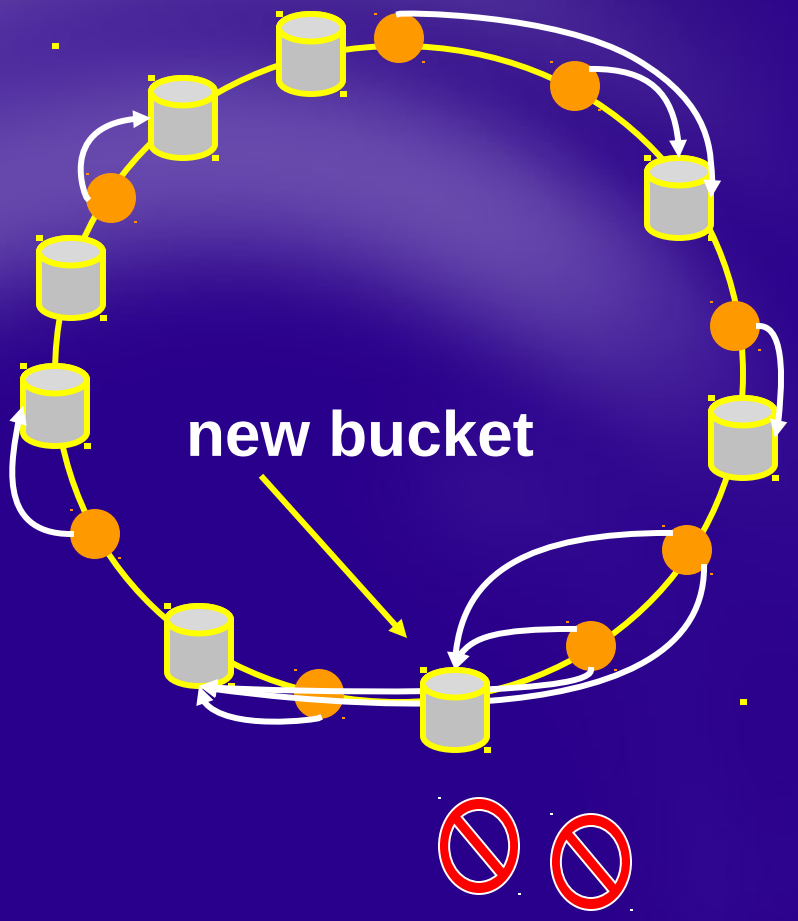


f(d) = d + 1 mod 5

f(d) = d + 1 mod 4

Consistent Hashing

Idea: Map both objects and buckets to unit circle.



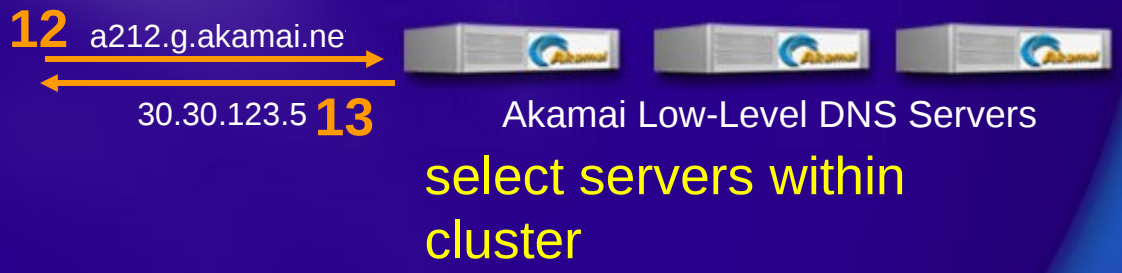
object

bucket

Assign object to next bucket on circle in clockwise order.

Complication – Different Views

Low-level DNS servers act independently and may have different ideas about how many and which servers are alive.



Properties of Consistent Hashing

Balance: Objects are assigned to buckets "randomly".

Monotonicity: When a bucket is added/removed, the only objects affected are those that are/were mapped to the bucket.

Load: Objects are assigned to buckets evenly, even over a set of views.

-- can be improved by mapping each bucket to multiple places on unit circle

Spread: An object should be mapped to a small number of buckets over a set of views.

How we really do it

random permutations of servers

Why? To spread load for one serial number.