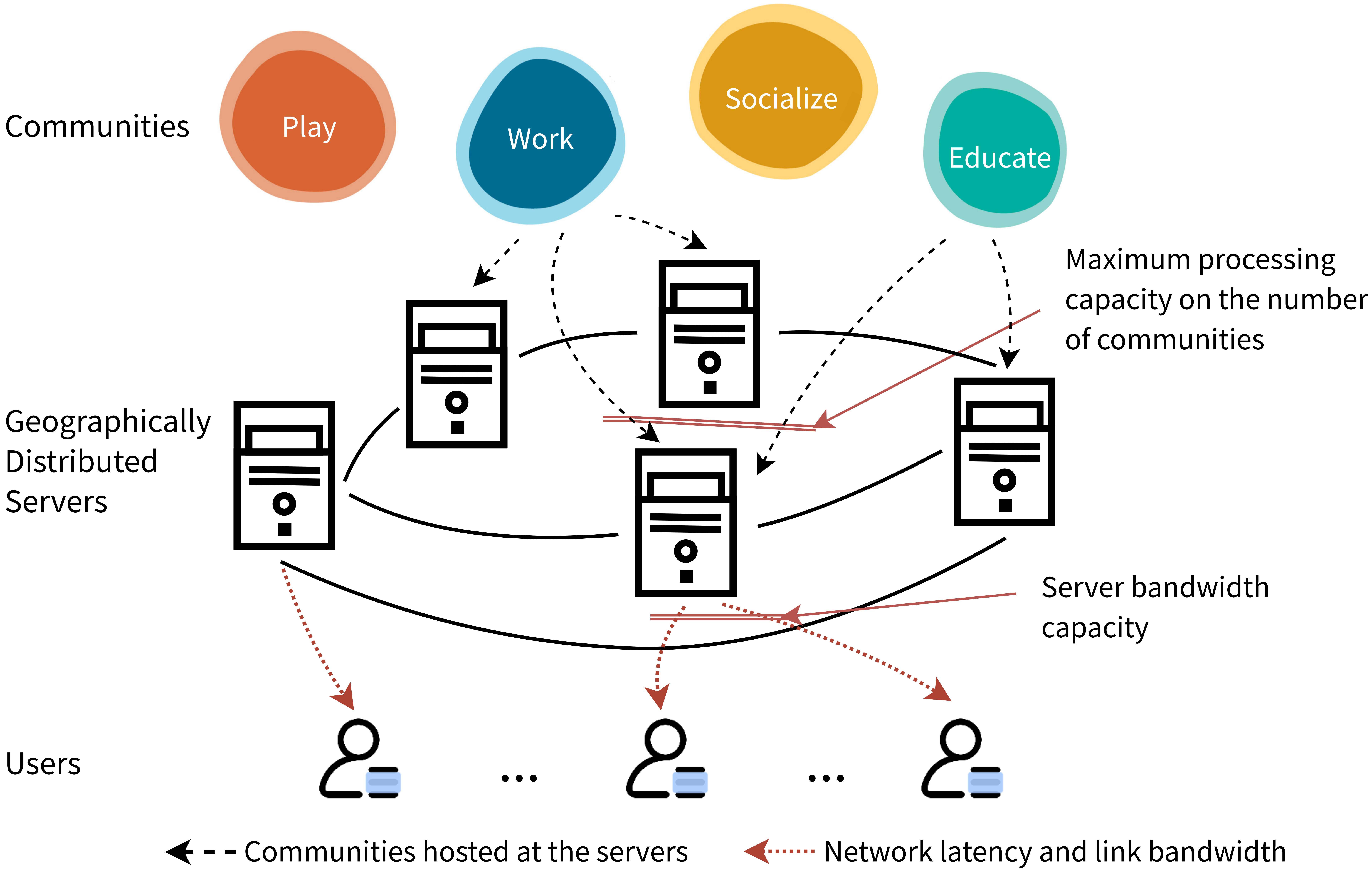
Multi-Server Stable Rendezvous for the Metaverse

Ningxin Su, Baochun Li
Department of Electrical and Computer Engineering
University of Toronto

Bo Li
Department of Computer Science and Engineering
Hong Kong University of Science and Technology

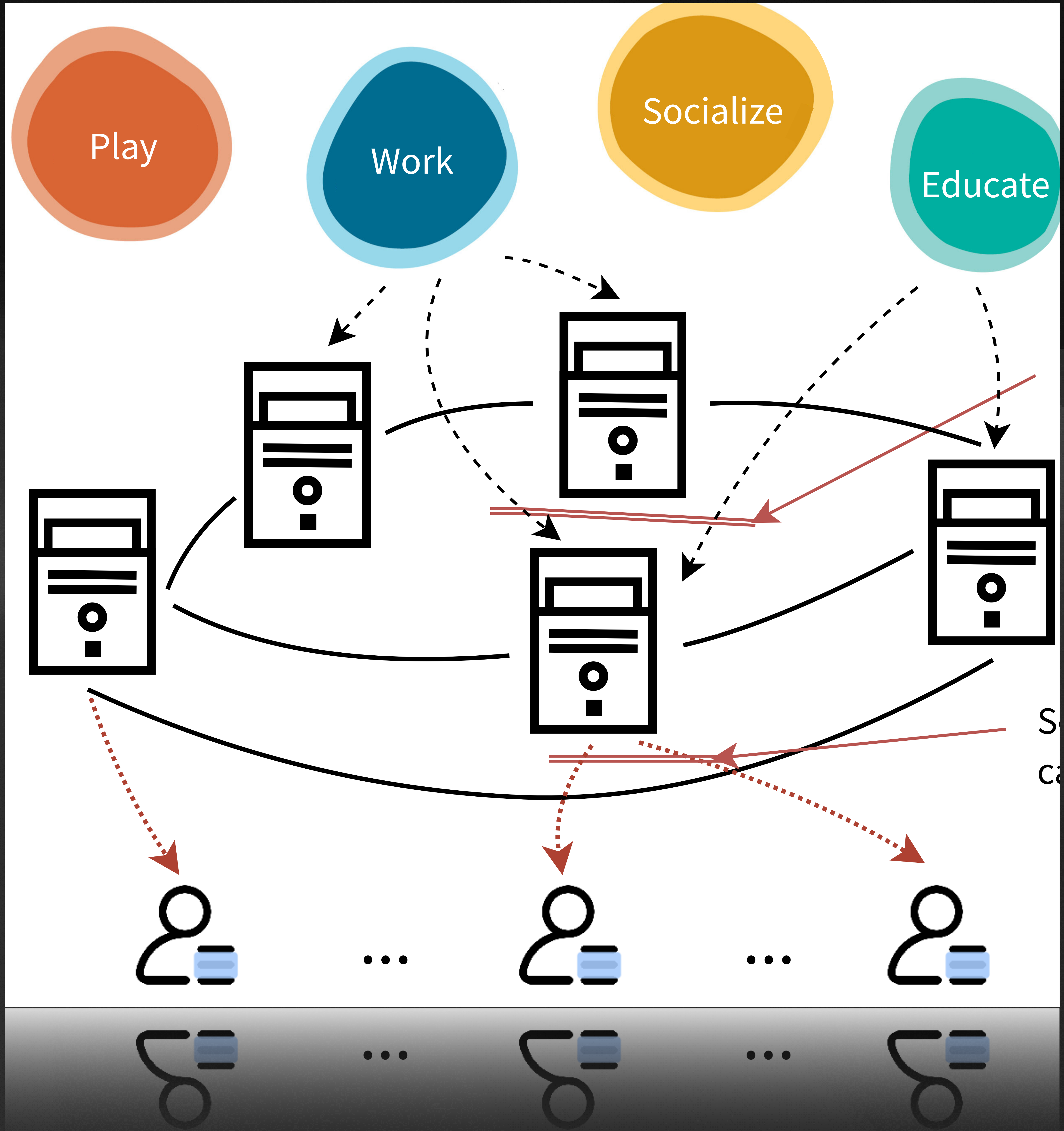


Communities
Play
Work
Socialize
Educate
Maximum processing capacity on the number of communities
Geographically Distributed Servers
Server bandwidth capacity
Users
...
...
Communities hosted at the servers
Network latency and link bandwidth

Objectives

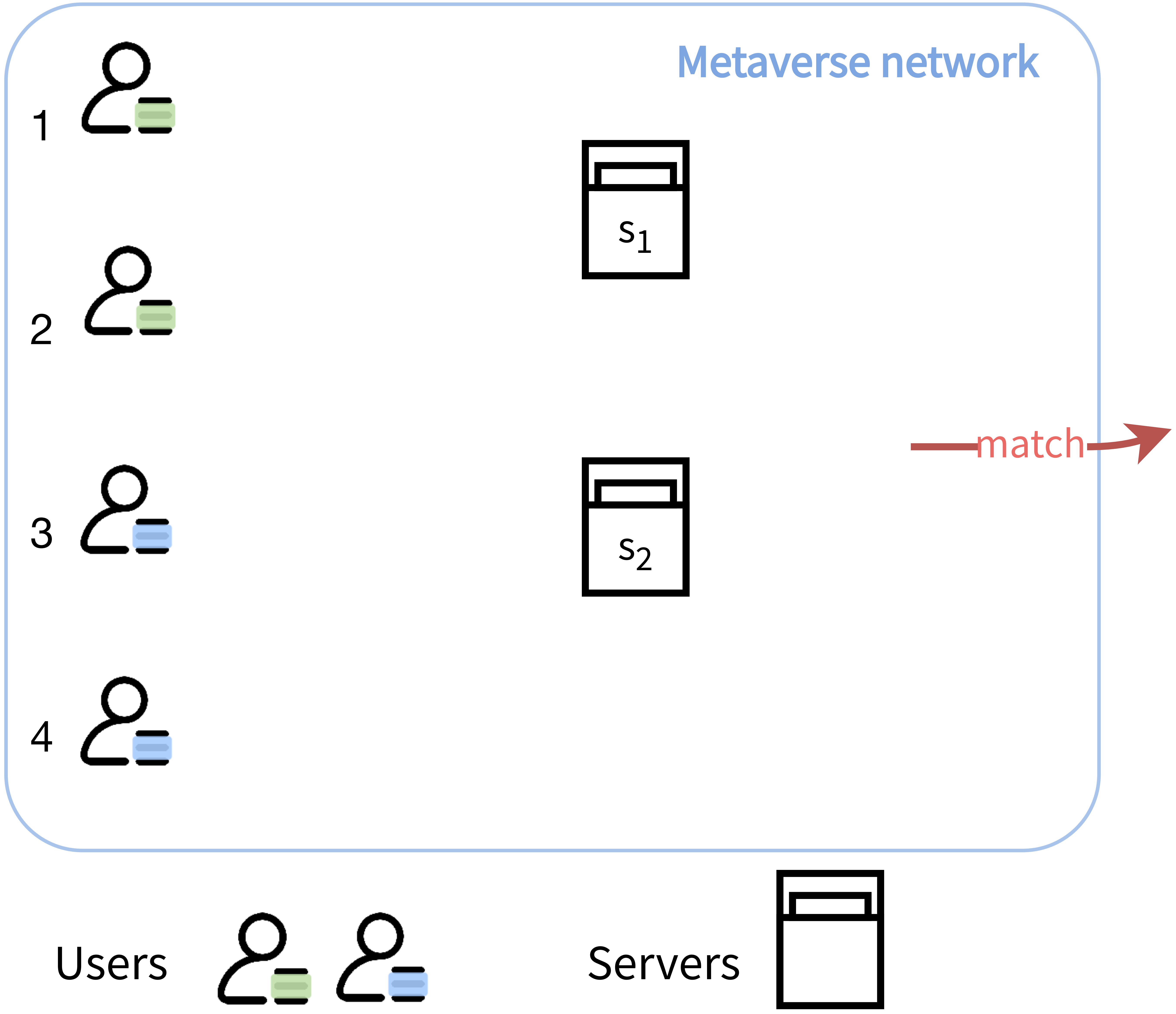
- fully decentralized
- community interests
- processing capacities on the servers
- latencies and bandwidth of links

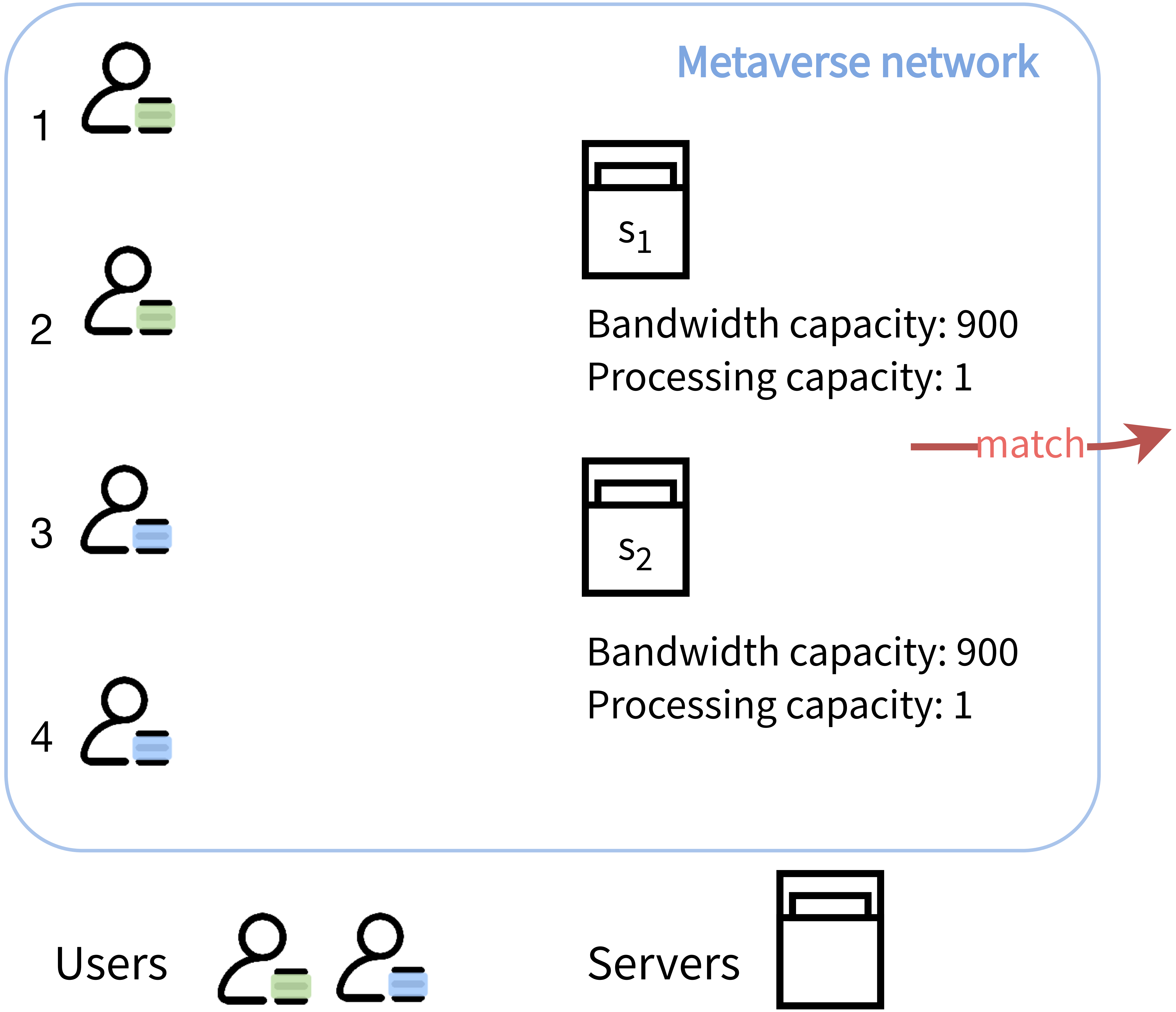
…

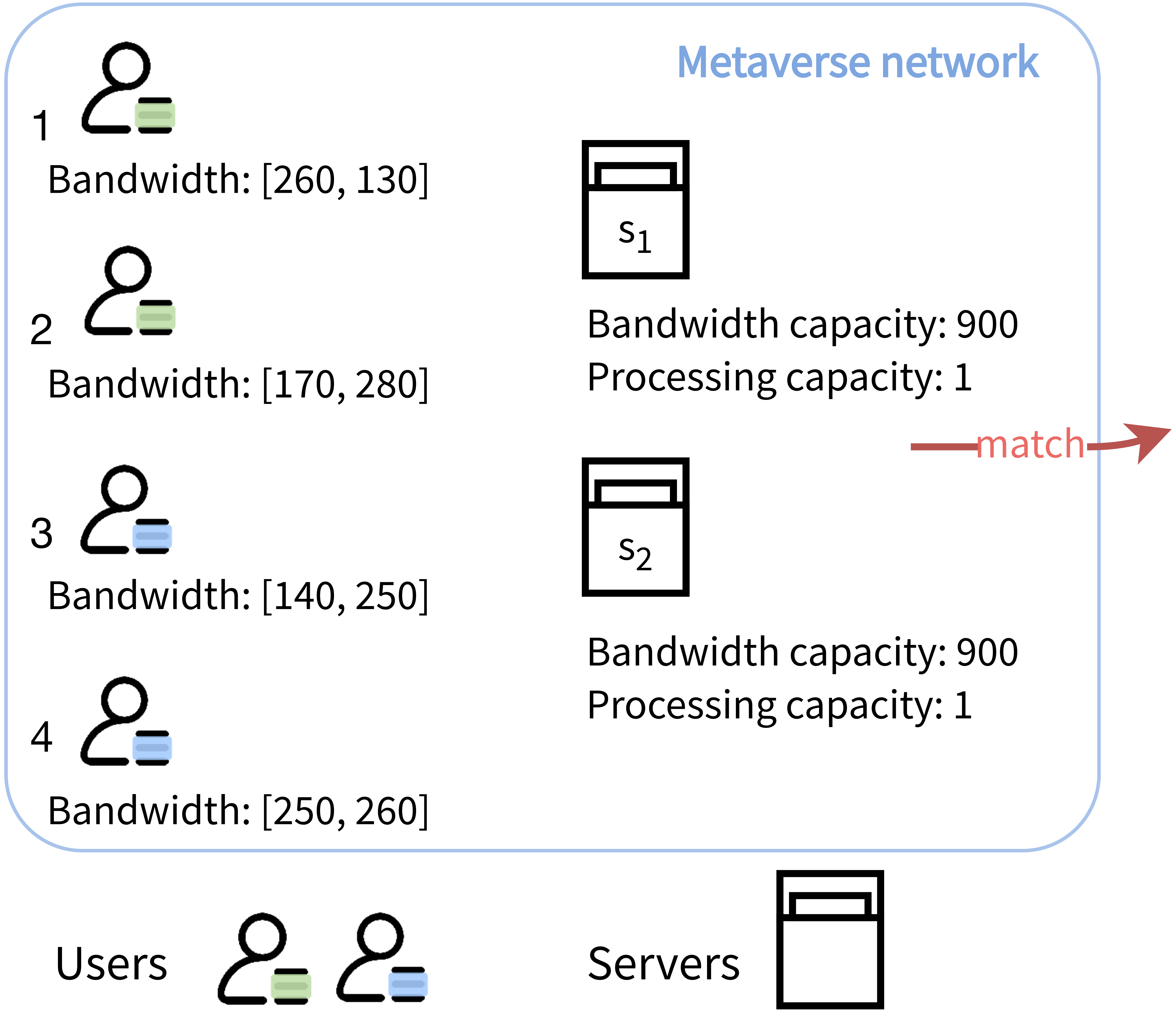


An optimization problem — *rendezvous* service in the Metaverse

A toy example


Metaverse network
1
2
3
4
s_1
s_2
match
Users
Servers

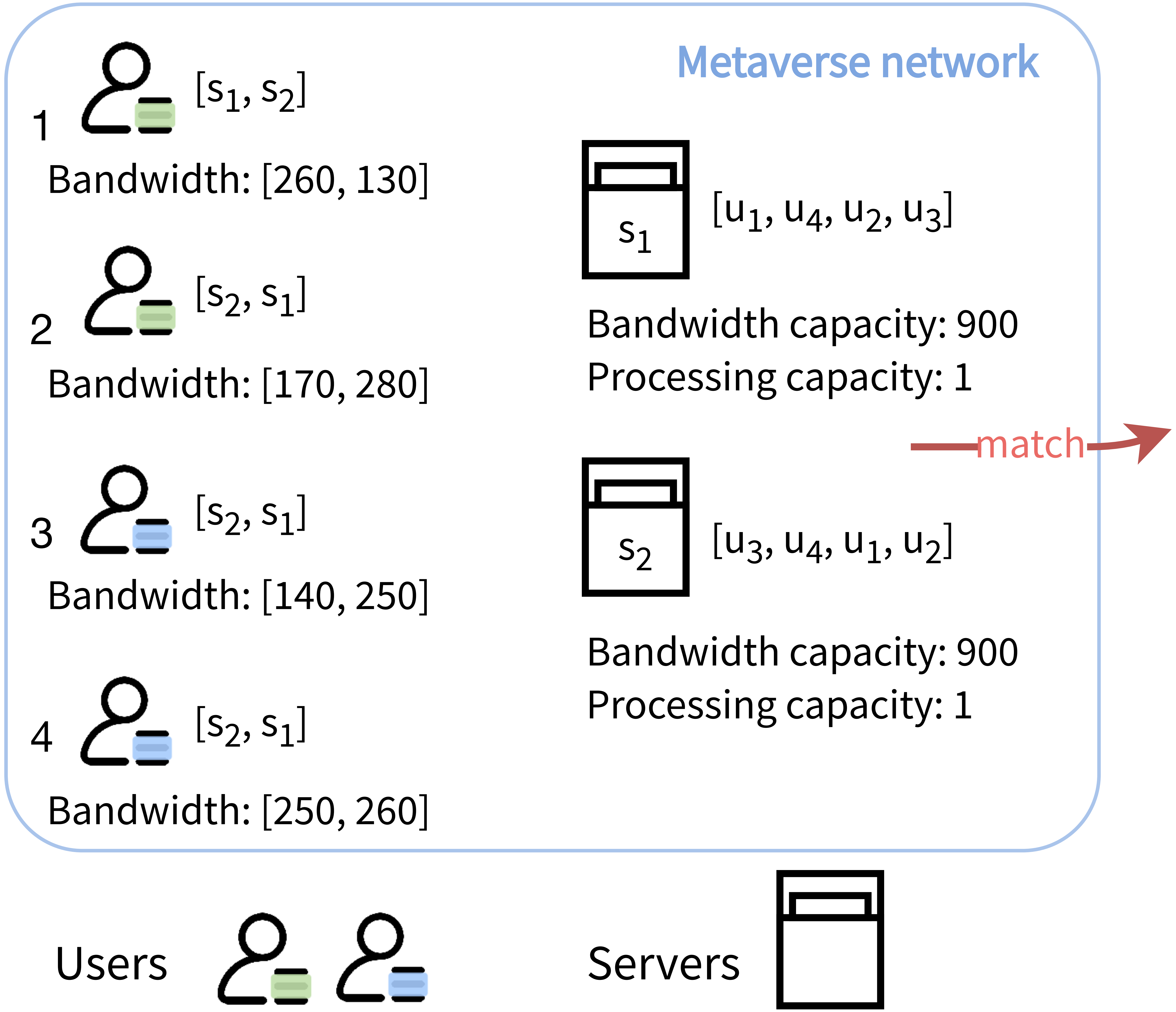

Metaverse network
1
2
3
4
s1
Bandwidth capacity: 900
Processing capacity: 1
match
s2
Bandwidth capacity: 900
Processing capacity: 1
Users
Servers


Metaverse network
1
Bandwidth: [260, 130]
2
Bandwidth: [170, 280]
3
Bandwidth: [140, 250]
4
Bandwidth: [250, 260]
s1
Bandwidth capacity: 900
Processing capacity: 1
match
s2
Bandwidth capacity: 900
Processing capacity: 1
Users
Servers

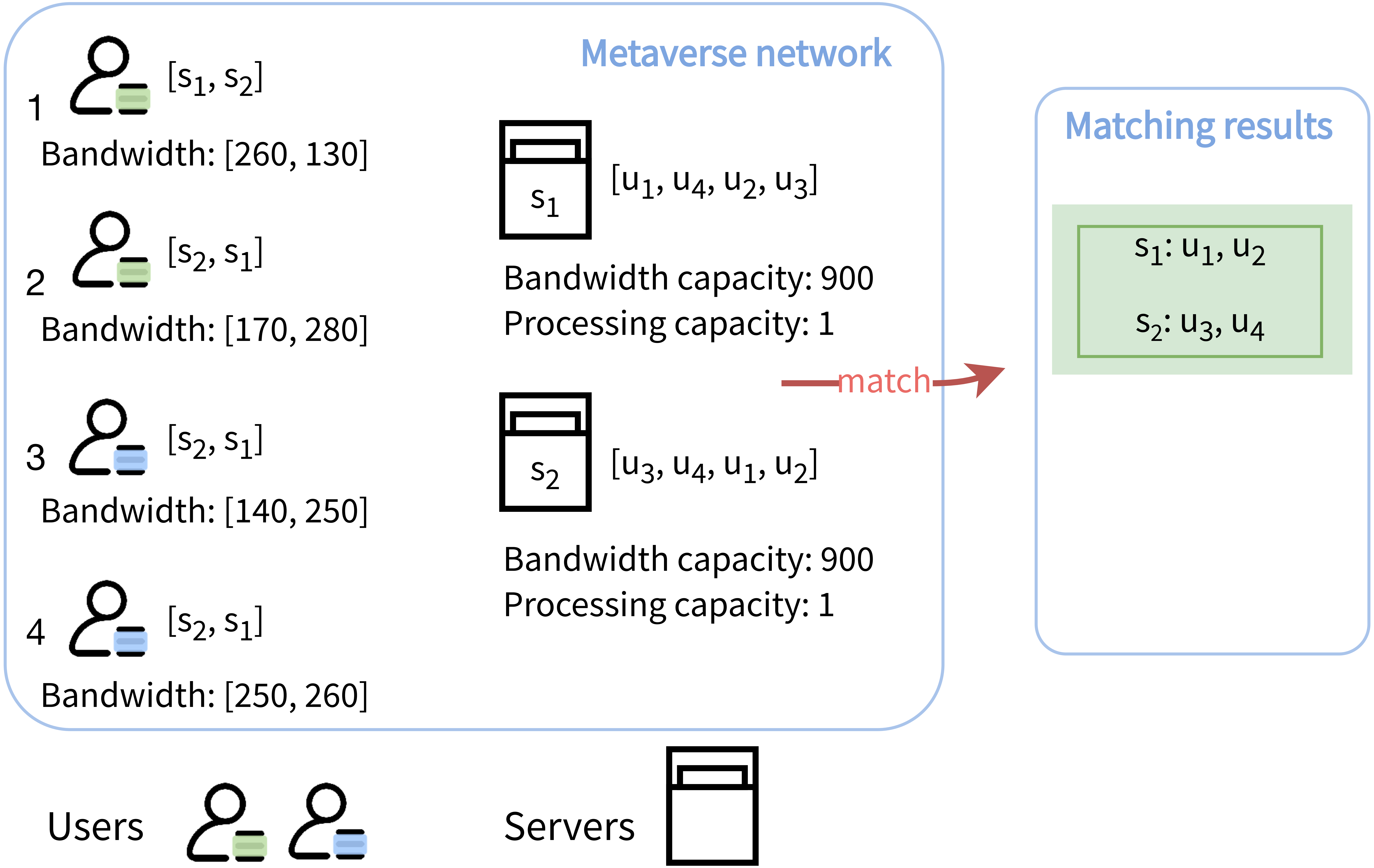
Metaverse rendezvous service

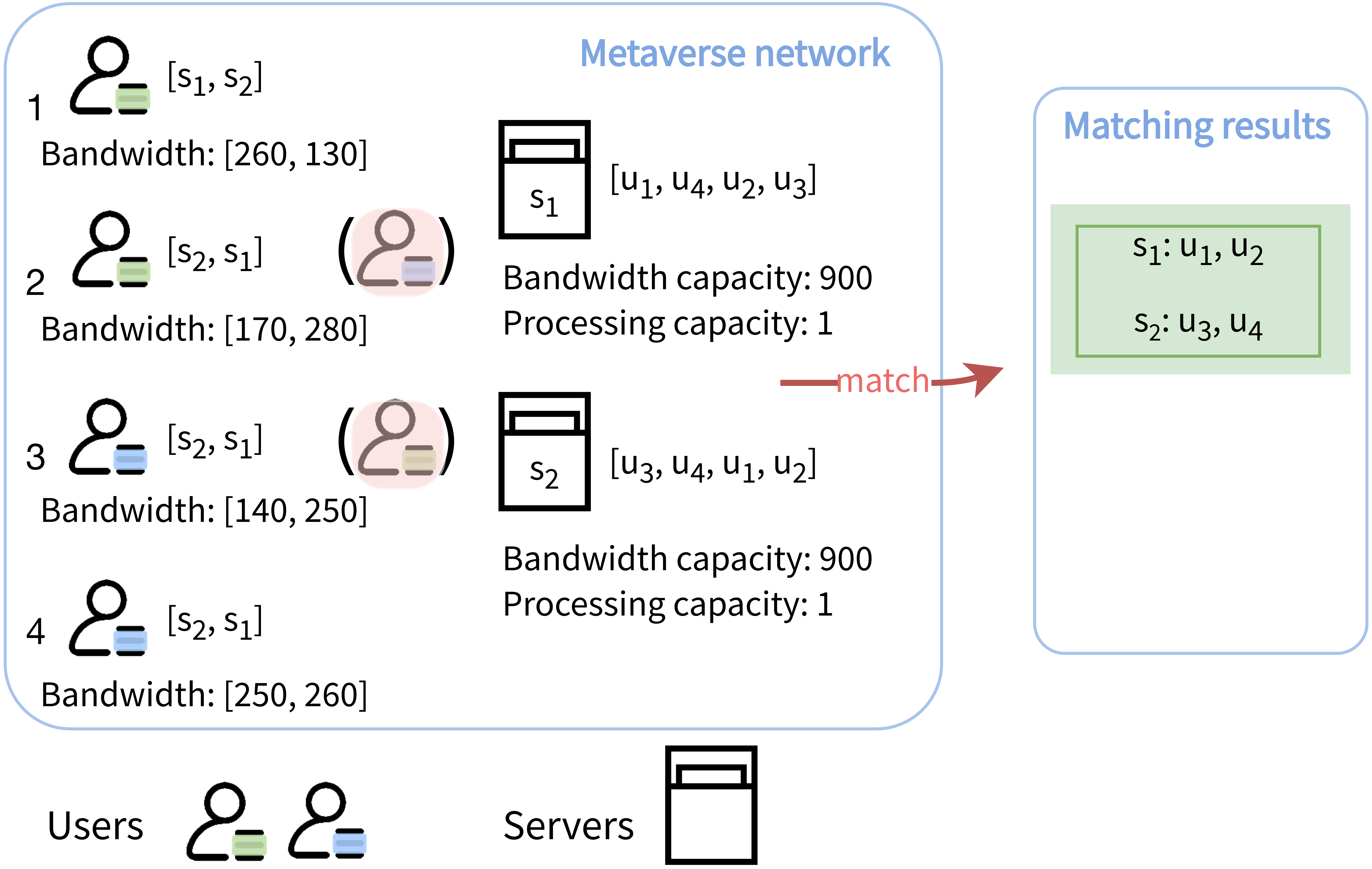
↘

Stable matching as the solution

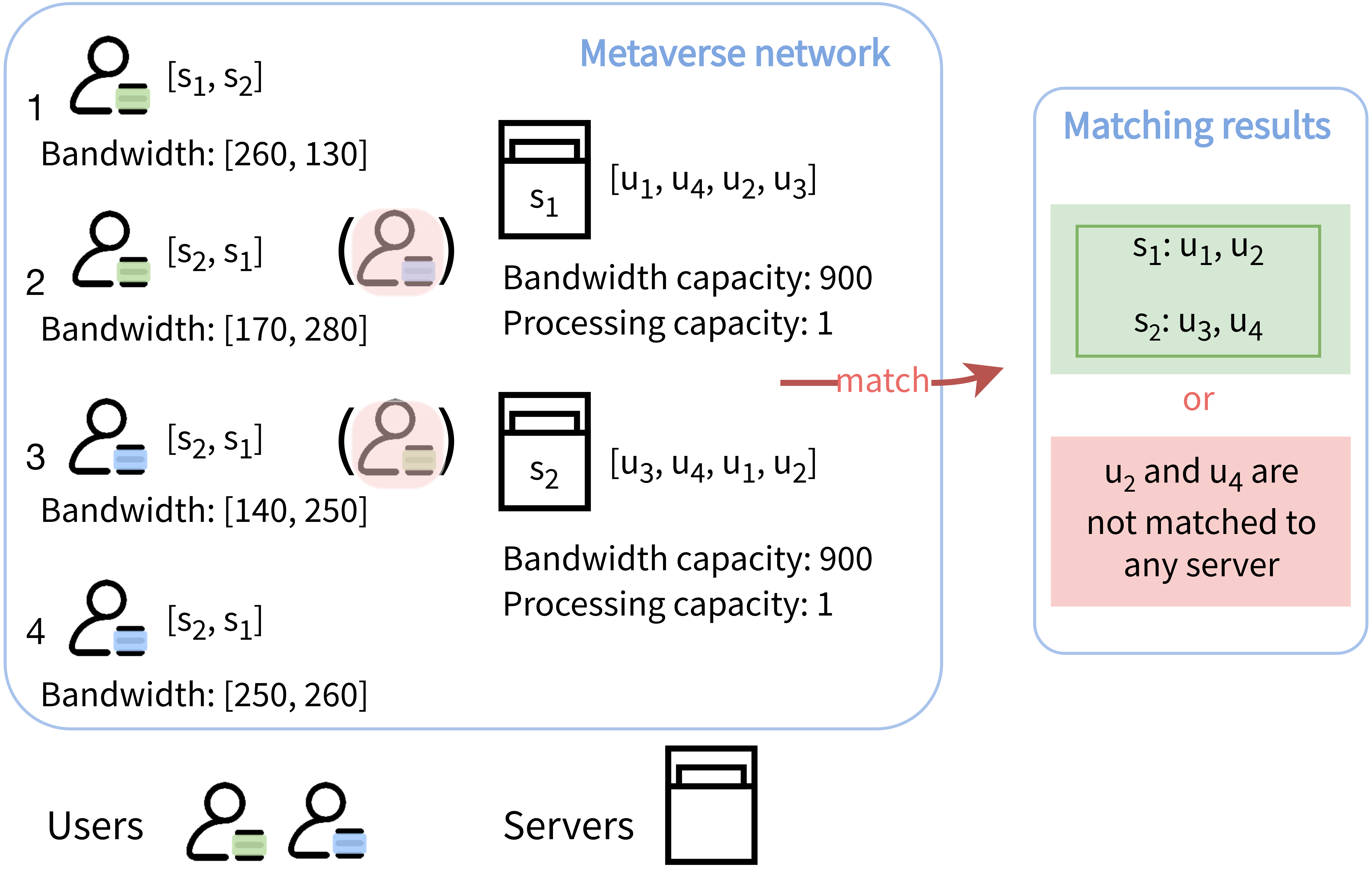


Metaverse network
1
[s1, s2]
Bandwidth: [260, 130]
2
[s2, s1]
Bandwidth: [170, 280]
3
[s2, s1]
Bandwidth: [140, 250]
4
[s2, s1]
Bandwidth: [250, 260]
s1
[u1, u4, u2, u3]
Bandwidth capacity: 900
Processing capacity: 1
match
s2
[u3, u4, u1, u2]
Bandwidth capacity: 900
Processing capacity: 1
Users
Servers


Metaverse network
1
[s_1, s_2]
Bandwidth: [260, 130]
2
[s_2, s_1]
Bandwidth: [170, 280]
3
[s_2, s_1]
Bandwidth: [140, 250]
4
[s_2, s_1]
Bandwidth: [250, 260]
s_1
[u_1, u_4, u_2, u_3]
Bandwidth capacity: 900
Processing capacity: 1
s_2
[u_3, u_4, u_1, u_2]
Bandwidth capacity: 900
Processing capacity: 1
match
Matching results
s_1: u_1, u_2
s_2: u_3, u_4
Users
Servers



Metaverse network
1
[s1, s2]
Bandwidth: [260, 130]
2
[s2, s1]
Bandwidth: [170, 280]
3
[s2, s1]
Bandwidth: [140, 250]
4
[s2, s1]
Bandwidth: [250, 260]
s1
[u1, u4, u2, u3]
Bandwidth capacity: 900
Processing capacity: 1
s2
[u3, u4, u1, u2]
Bandwidth capacity: 900
Processing capacity: 1
match
Matching results
s1: u1, u2
s2: u3, u4
Users
Servers


Metaverse network
1
[s1, s2]
Bandwidth: [260, 130]
2
[s2, s1]
Bandwidth: [170, 280]
3
[s2, s1]
Bandwidth: [140, 250]
4
[s2, s1]
Bandwidth: [250, 260]
s1
[u1, u4, u2, u3]
Bandwidth capacity: 900
Processing capacity: 1
s2
[u3, u4, u1, u2]
Bandwidth capacity: 900
Processing capacity: 1
match
Matching results
s1: u1, u2
s2: u3, u4
or
u2 and u4 are not matched to any server
Users
Servers

Let’s dive into the details

An existing solution of stable matching problem

Definition 1 (Matching). *An outcome of the college admissions problem is a* matching $\mu : \mathcal{A} \times \mathcal{C} \rightarrow \mathcal{A} \times \mathcal{C}$ *such that* $a \in \mu(c)$ *if and only if* $\mu(a) = c$, *and* $\mu(a) \in \mathcal{C} \cup \emptyset, \mu(c) \subseteq \mathcal{A} \cup \emptyset, \forall c, a.$

Definition 2 (Individual rationality)**.** *A matching is* individual rational *if and only if there does not exist an applicant* a *(or a college* c*) who prefers being unmatched to being matched with* $\mu(a)$ *(or* $\mu(c)$*), i.e.* $\emptyset \succ_a \mu(a)$ *(or* $\emptyset \succ_c \mu(c)$*) should not exist.*

Definition 3 (Blocking pair)**.** *A matching* μ *is blocked by an applicant-college pair* (a, c) *if they prefer each other to the match they receive at* μ*. That is,* $c \succ_a \mu(a)$ *and* $a \succ_c a', \exists\, a' \in \mu(c)$*.*

Definition 4 (Stablility)**.** *A matching* μ *is* stable *if and only if it is both* individually rational *and not* blocked *by any other pairing between applicants and colleges.*

How can we add multiple constraints in the existing matching problem?

while $\mathcal{N} \neq \emptyset$ **do** ▷ $\mathcal{N}$ denotes the set of unassigned users
    $i \leftarrow 0$
    for $u \in \mathcal{N}$ **do**
        $s \leftarrow$ the i^{th} item from $L(u)$
        Add u to the set of users on hold at server s, $\mathcal{H}(s)$
    end for
    $\mathcal{N} \leftarrow \emptyset$
    for $s \in \mathcal{S}$ **do**
        for $u \in \mathcal{H}(s)$ **do** ▷ iterate in the ranked order in $L(s)$
            Add $c(u)$ into the set of community IDs $\mathcal{C}(s)$
            if $b(s,u) \leq B(s)$ and $|\mathcal{C}(s)| \leq P(s)$ **then**
                $B(s) \leftarrow B(s) - B(s,u)$
            else
                Remove u from $\mathcal{H}(s)$
                Add u into the set of rejected users $\mathcal{R}(s)$
                Remove $c(u)$ from $\mathcal{C}(s)$
            end if
        end for
        $\mathcal{N} = \mathcal{N} \cup \mathcal{R}(s)$
    end for
    $i \leftarrow i + 1$
end while
return the final matching $\mathcal{H}(s)$

Conditions to satisfy constraints

Fully decentralized

Optimization solution

Stability

Scalability

Fully decentralized

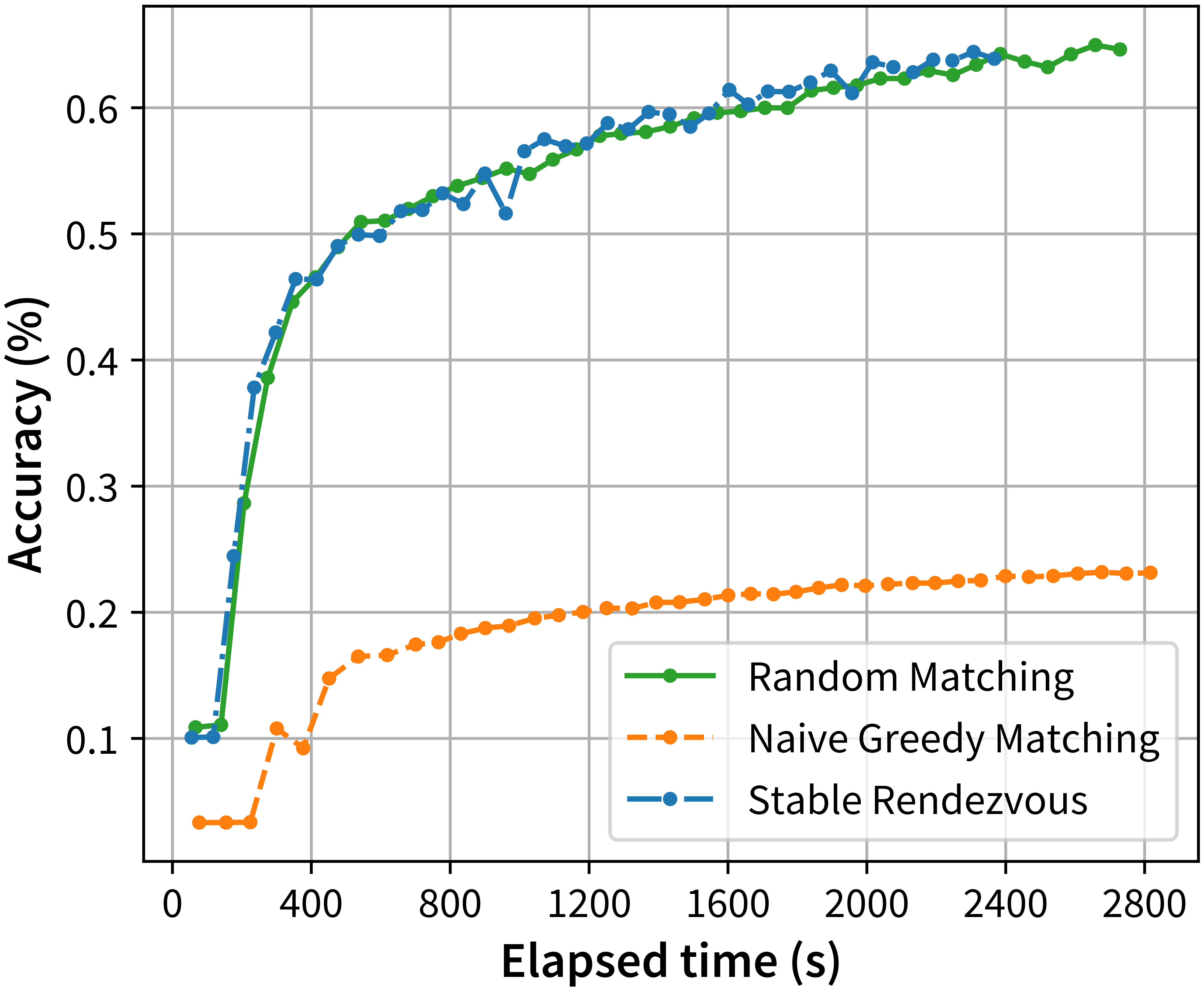
Optimization solution

Stability

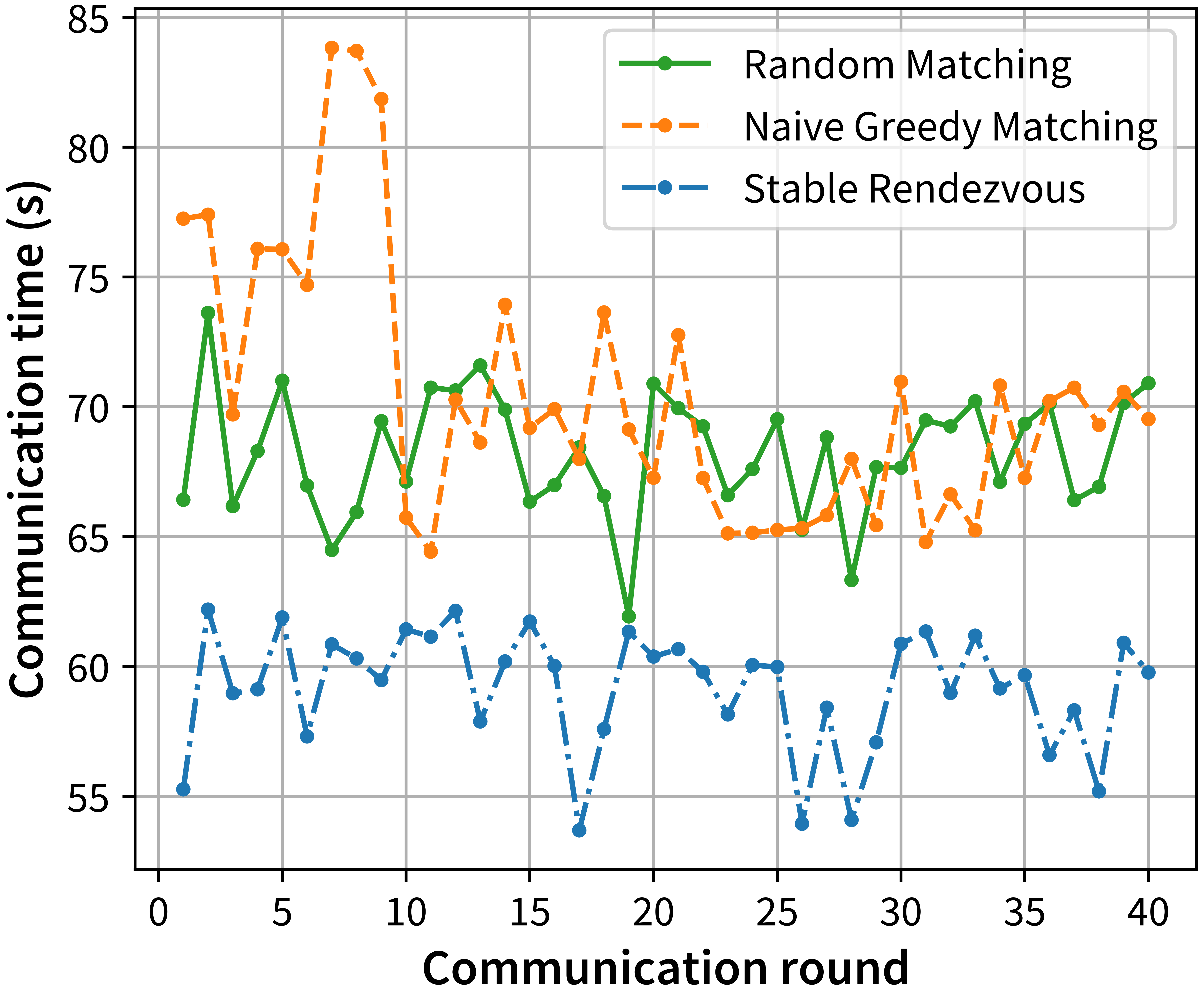
Scalability

Is it Safe and Effective in the real world?

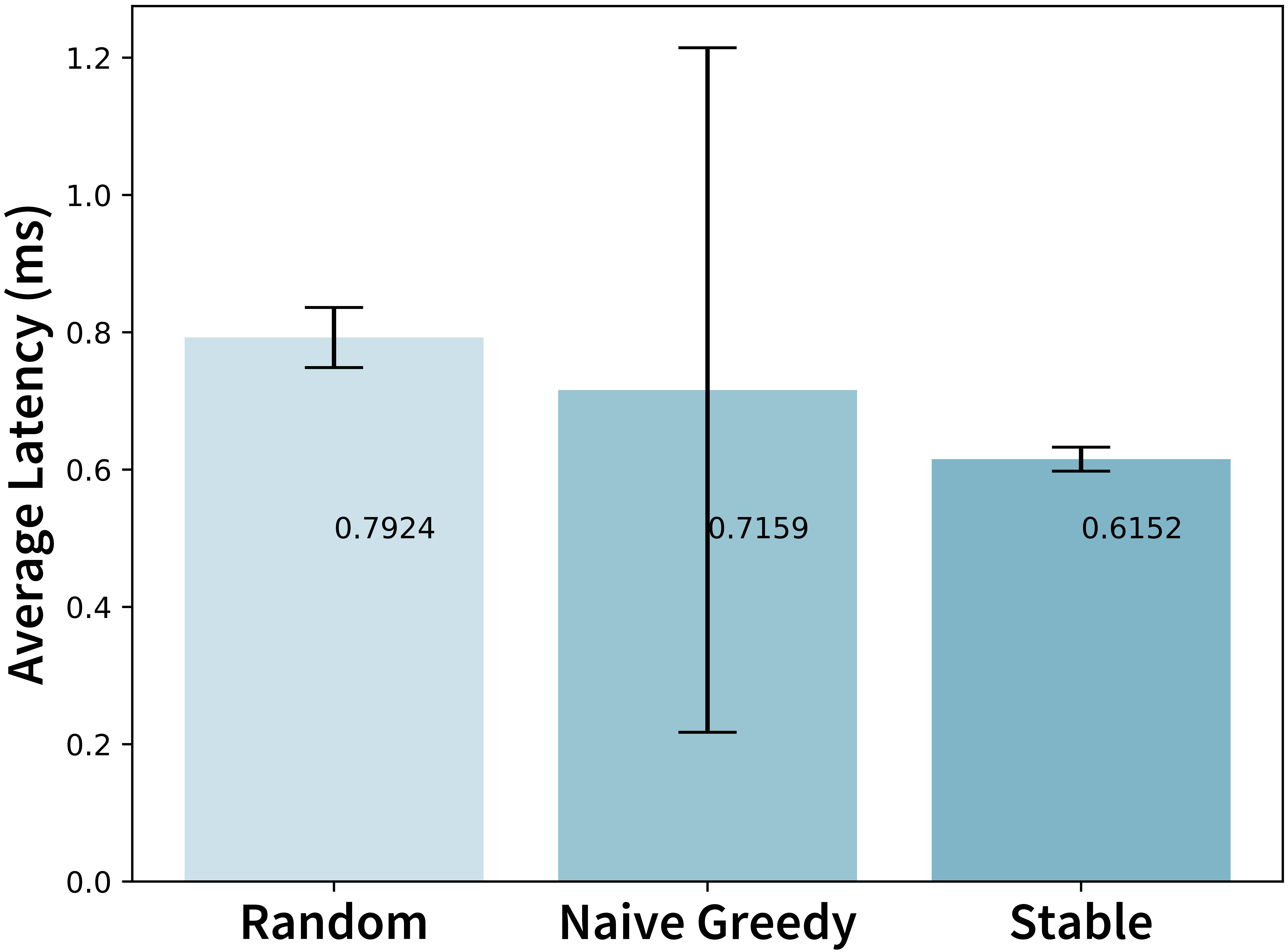
An experimental case study — Multi-server federated learning



— CIFAR-10&
ResNet18



— CIFAR-10&
ResNet18



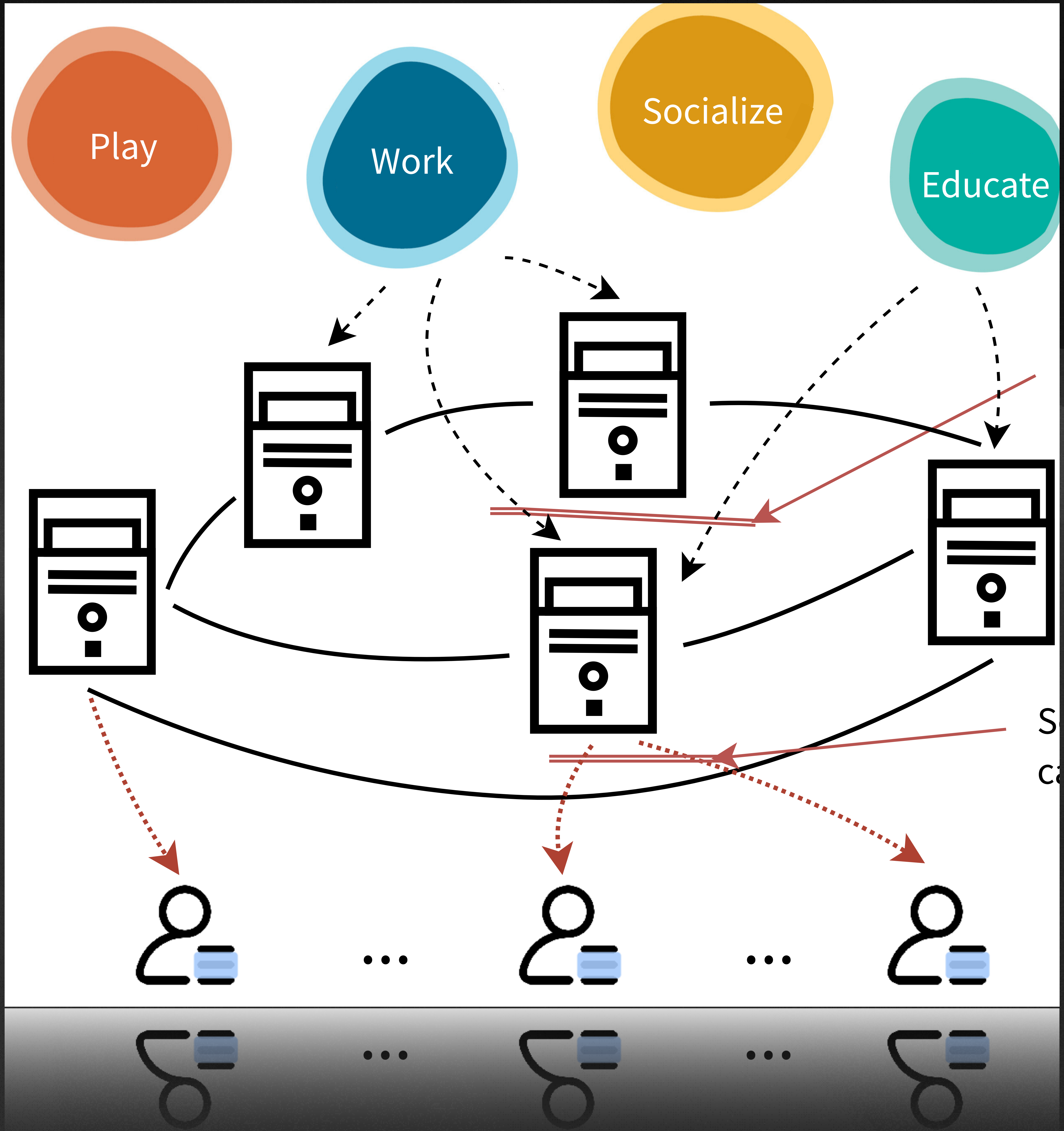
– CIFAR-10&
ResNet18

Privacy,
communication overhead,
performance and latency are guaranteed

Objectives Revisited

- fully decentralized
- community interests
- processing capacities on the servers
- latencies and bandwidth of links
- privacy

…



ningxinsu.github.io