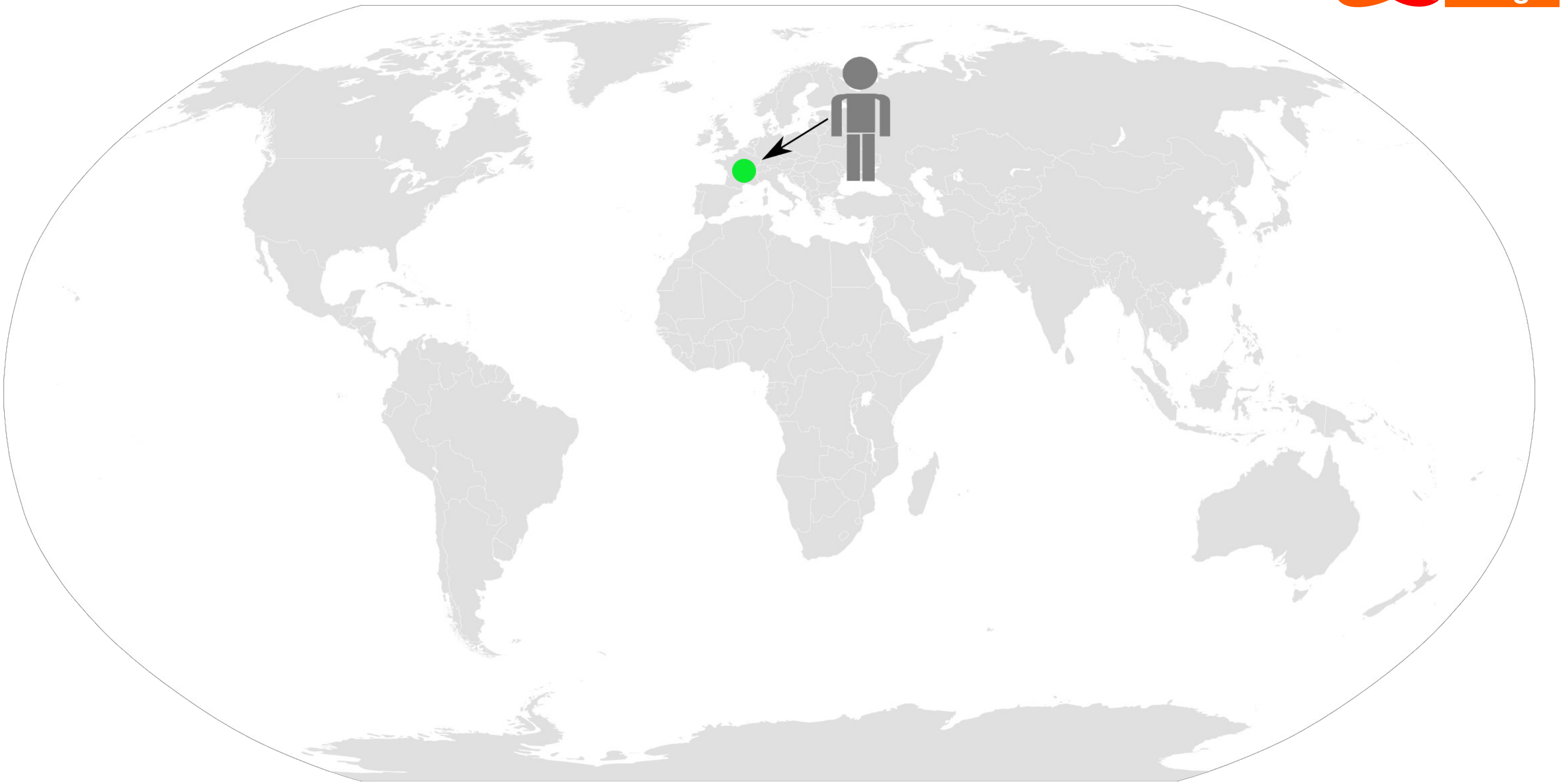
Caju: a content distribution system for edge networks

Guthemberg Silvestre[1,2], Sébastien Monnet[1],
Ruby Krishnaswamy[2] , and Pierre Sens[1]

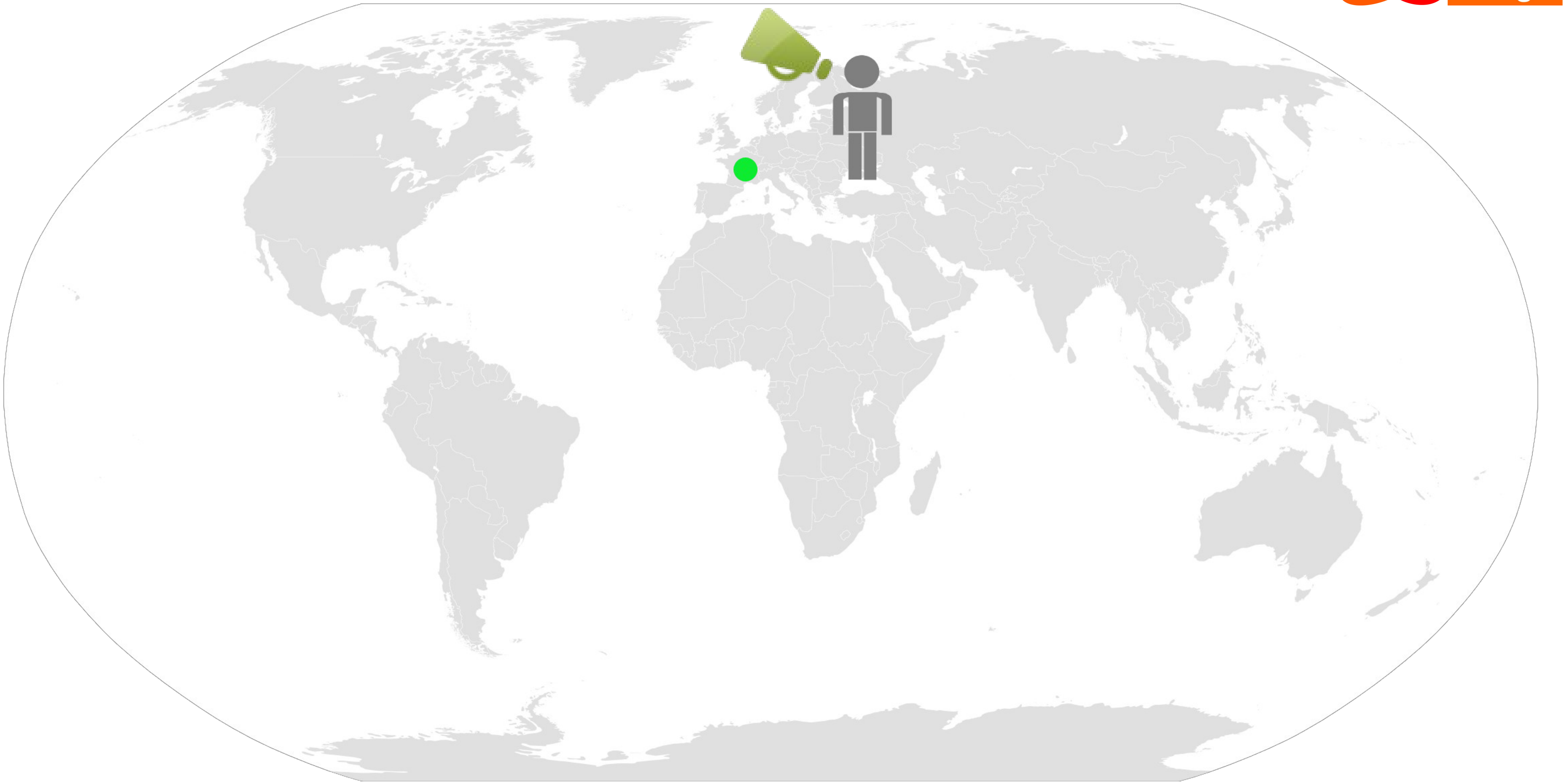
[1]LIP6/UPMC/CNRS/INRIA
[2]Orange Labs

BDMC 2012, Rhodes Island, Greece
27 August 2012

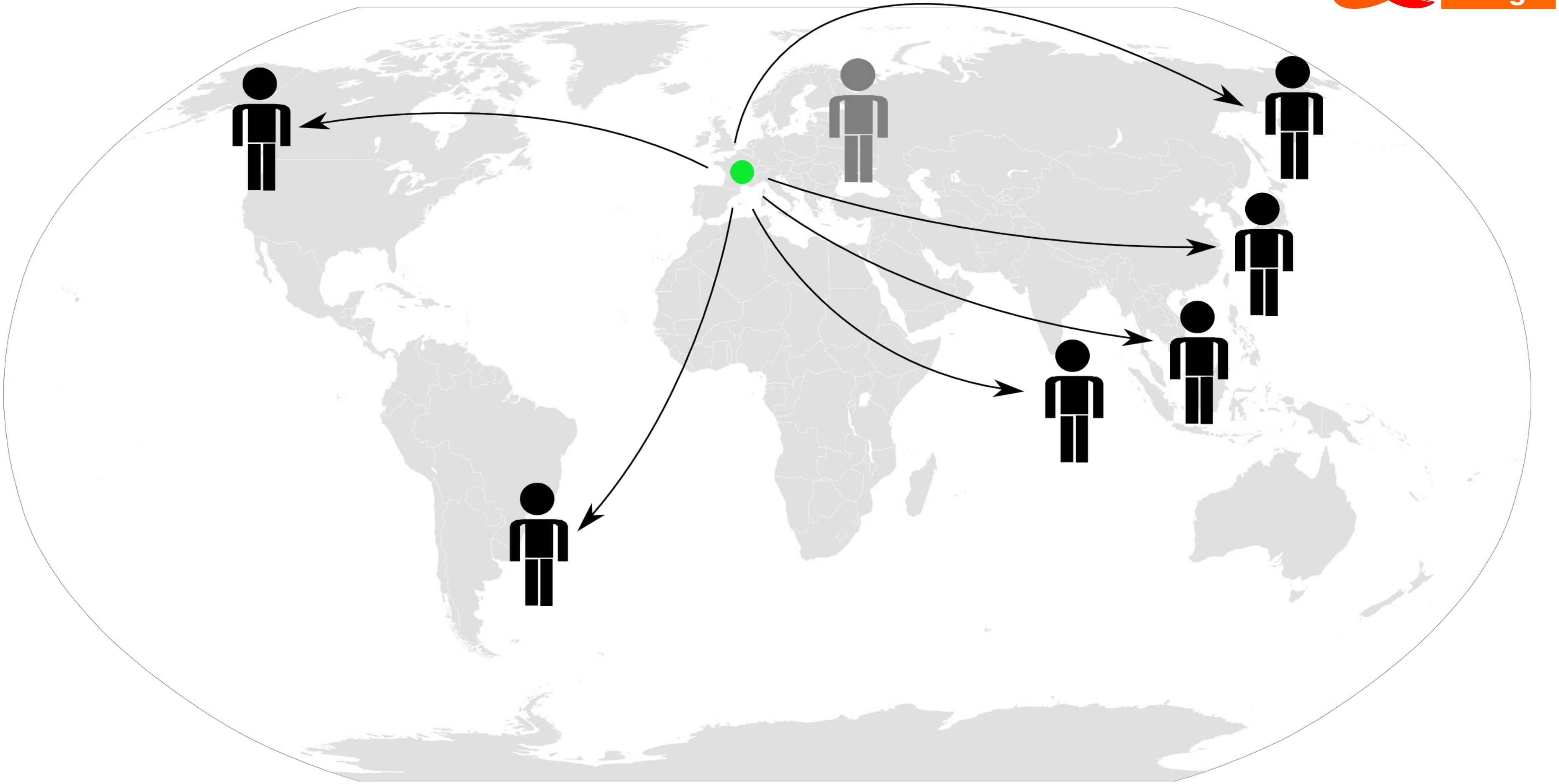
Let me store and share...



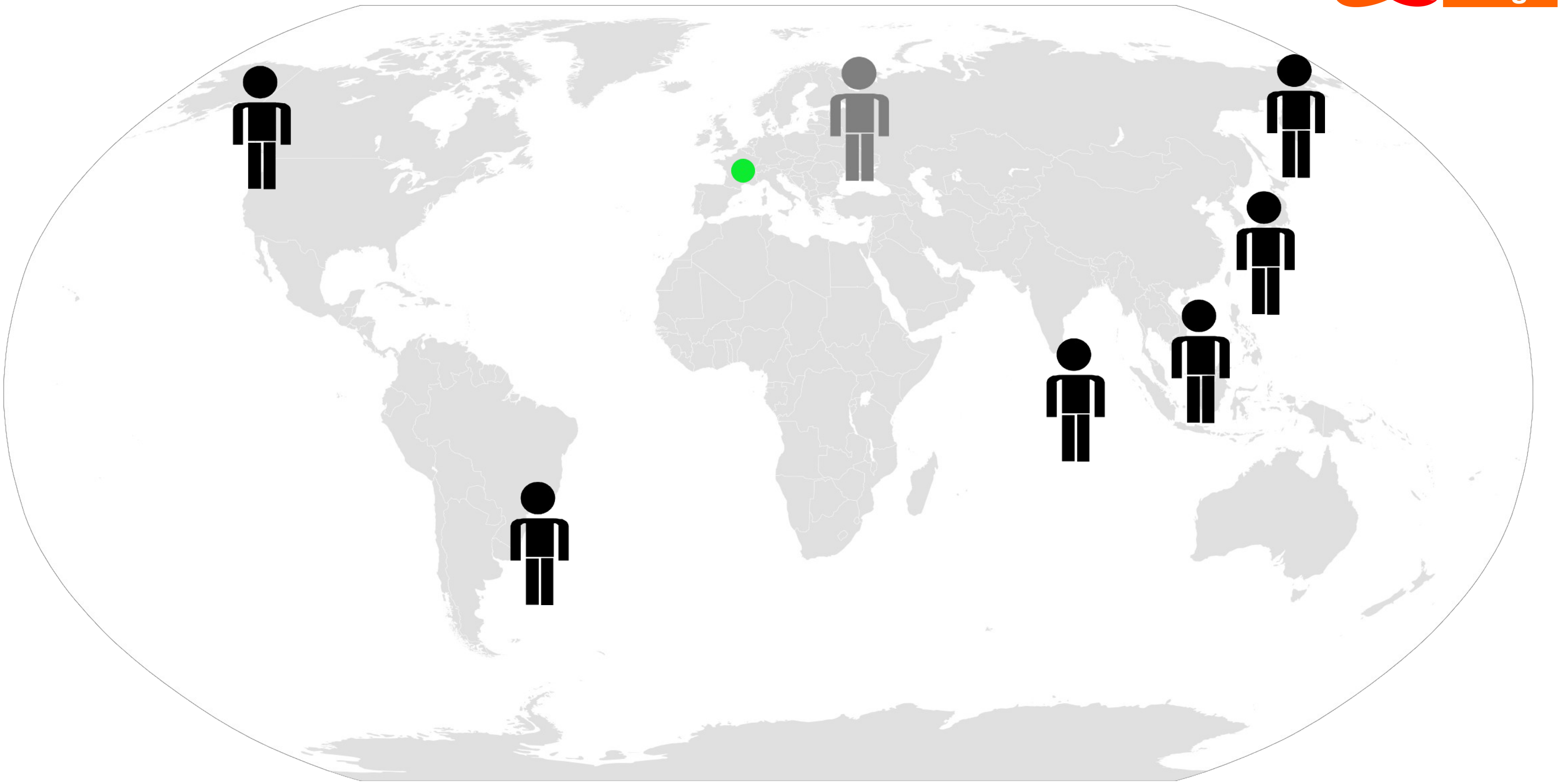
Let me store and share...



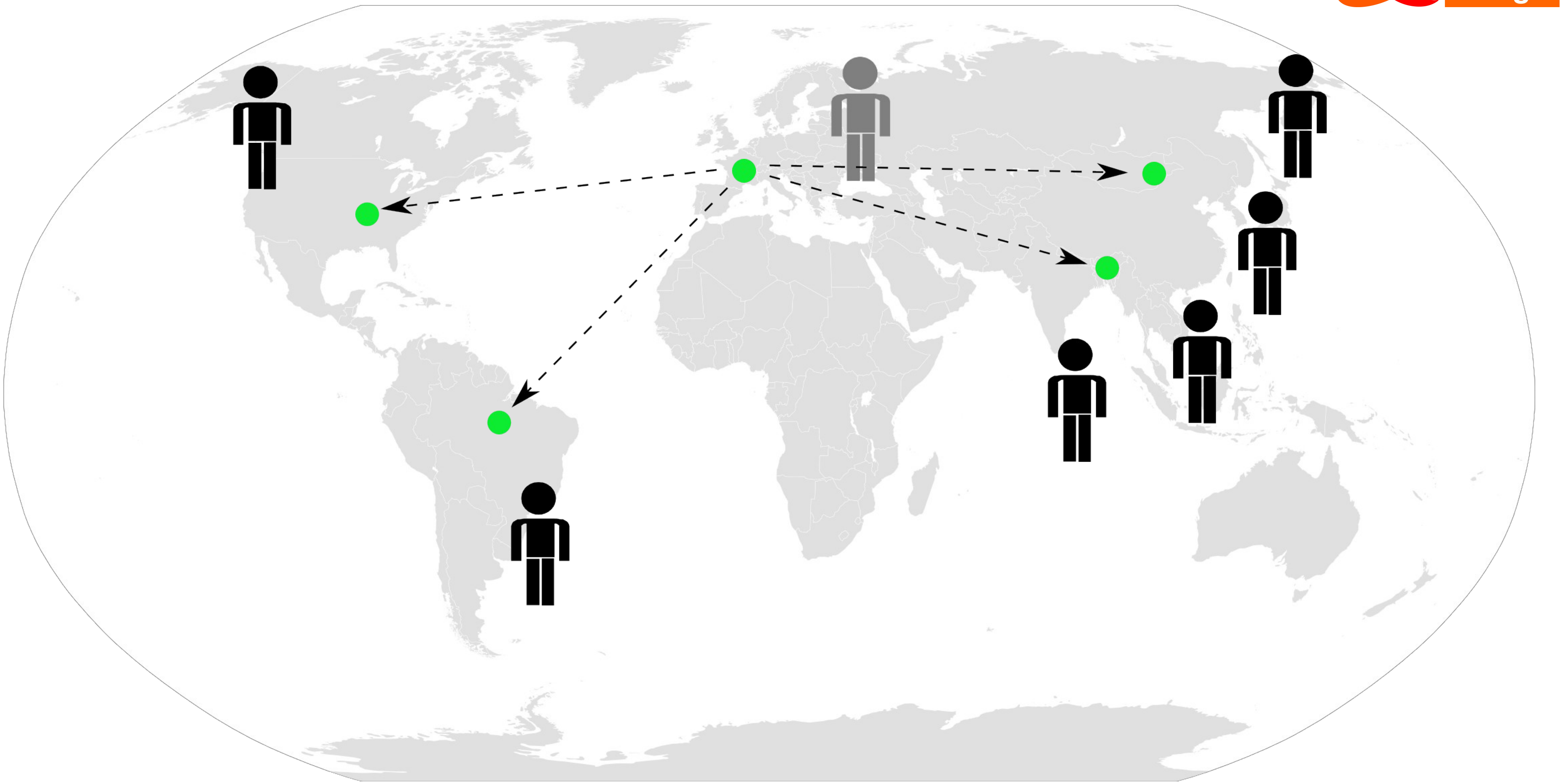
Let me store and share...



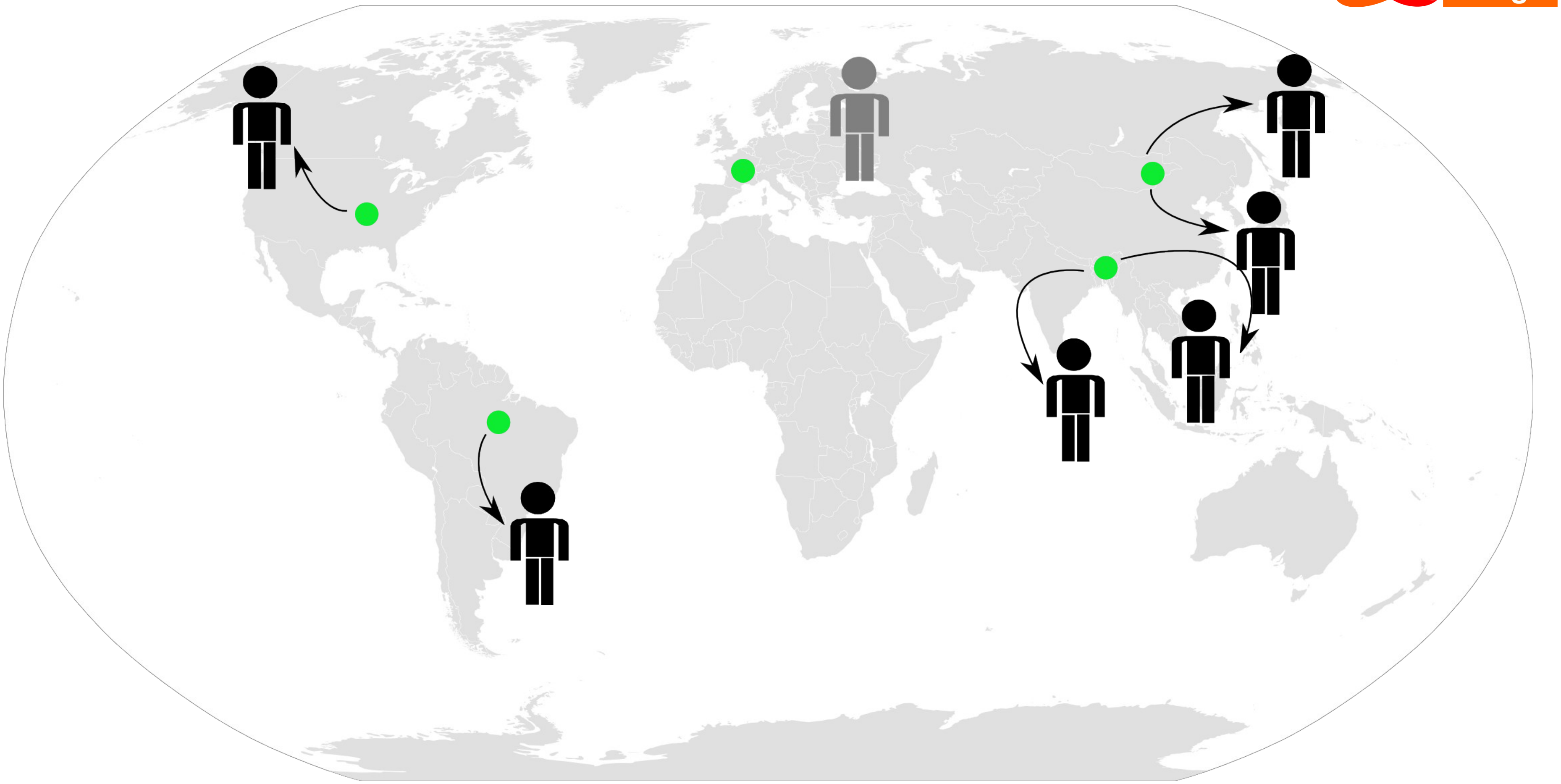
Let me store and share...



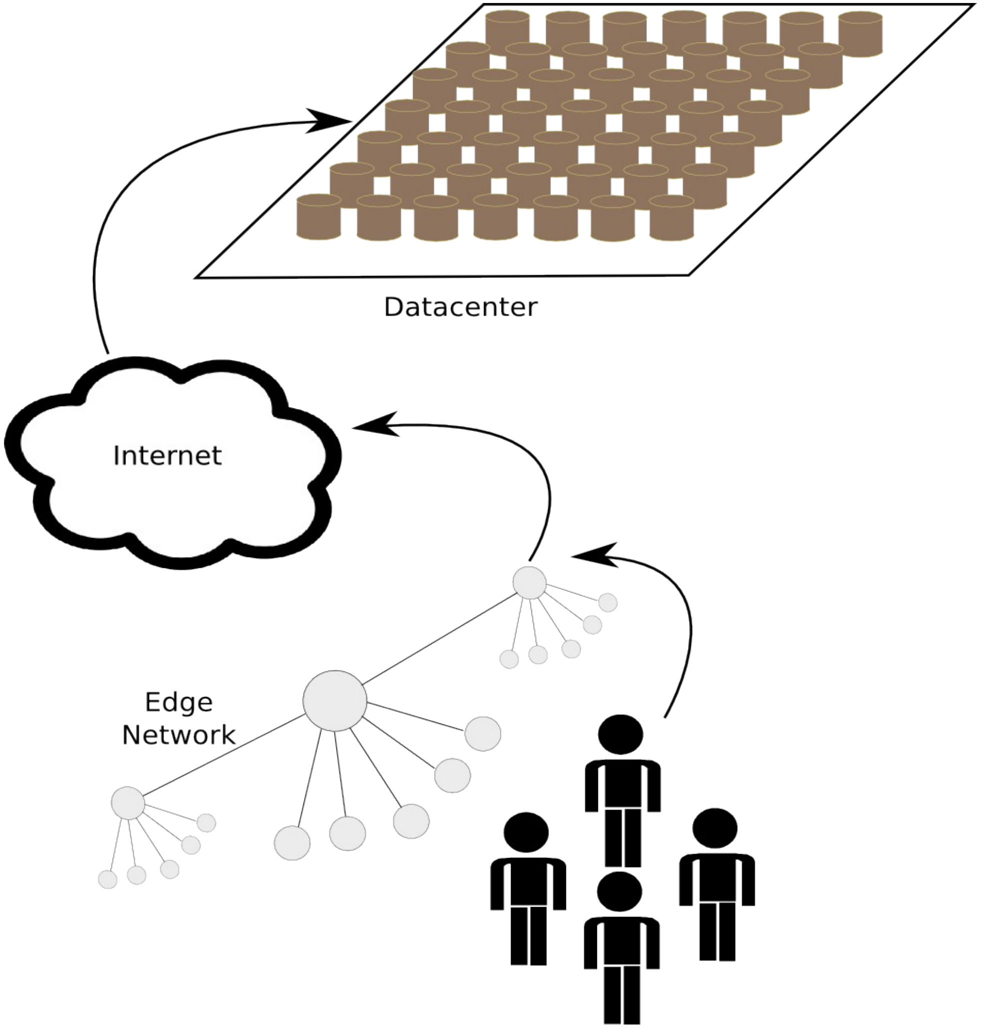
Let me store and share...



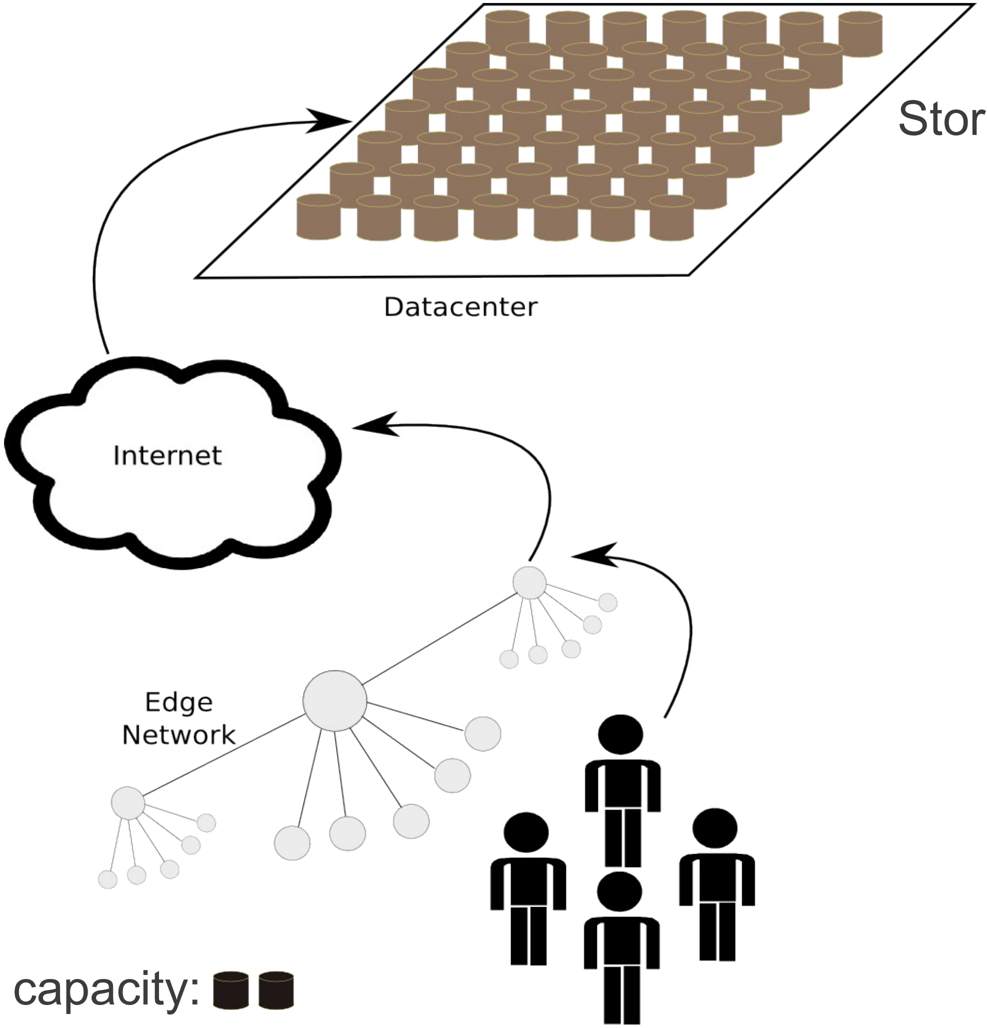
Let me store and share...



… at the edge of the network



… at the edge of the network

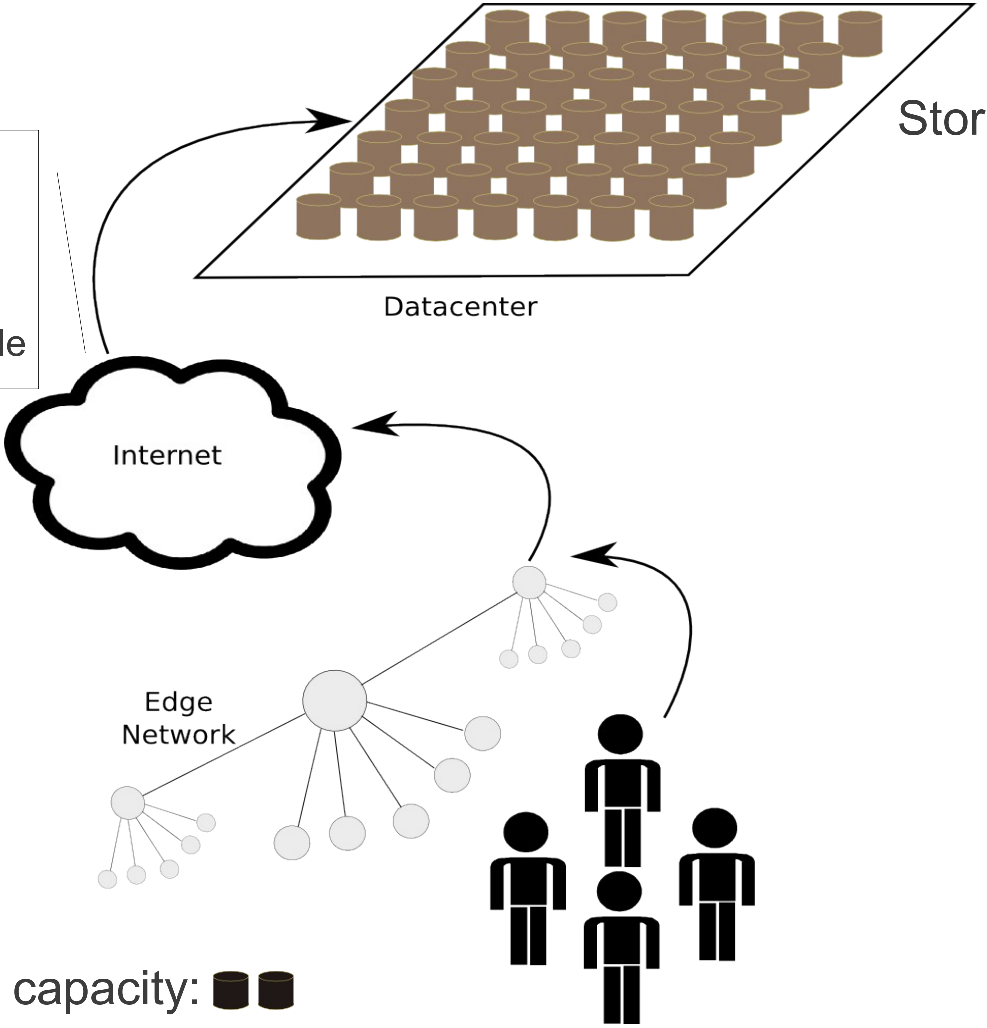


Storage capacity: ■■■■■

Storage capacity: ■■

… at the edge of the network

Proximity: ???
Network capacity: ???
Energy consumption: $$$
Load: potentially heavy
Spare resources: unpredictable



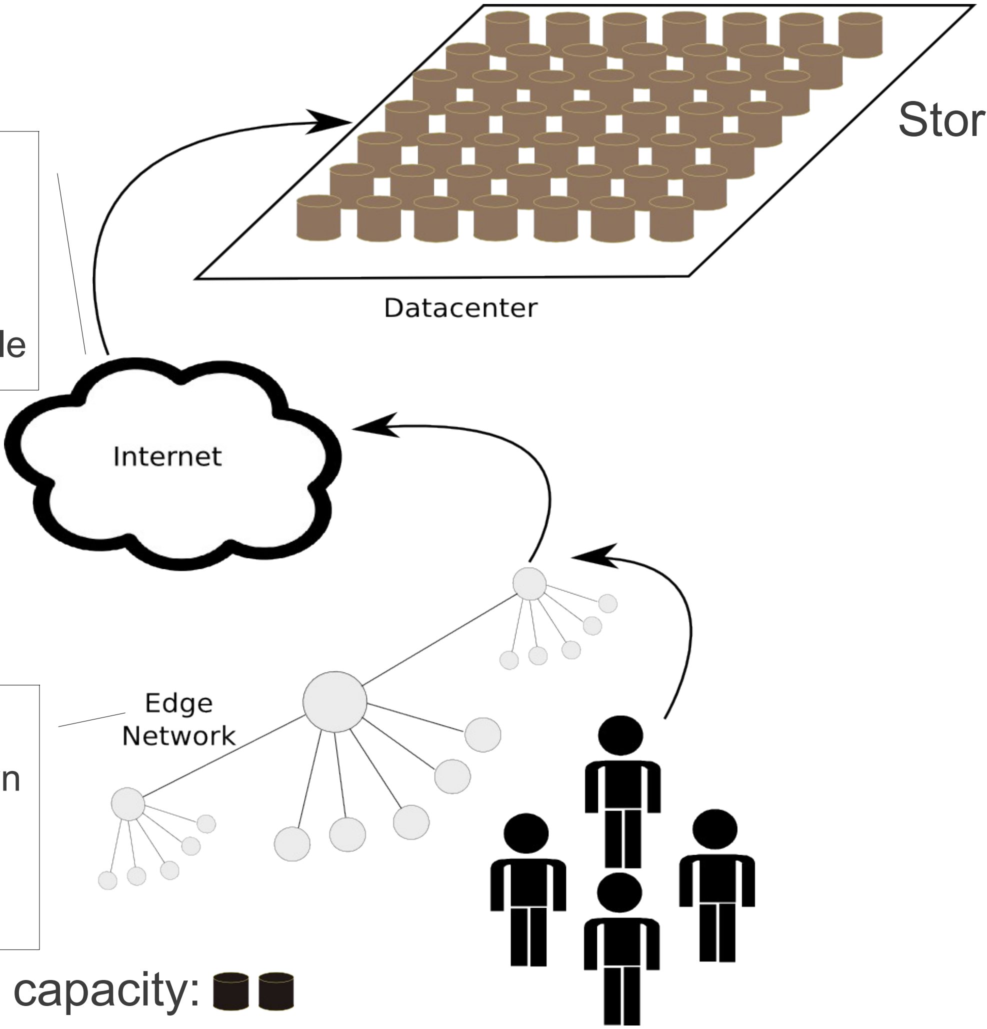
Storage capacity: ■■■■■

Storage capacity: ■■

… at the edge of the network

Proximity: ???
Network capacity: ???
Energy consumption: $$$
Load: potentially heavy
Spare resources: unpredictable

Storage capacity: ⛁⛁⛁⛁⛁

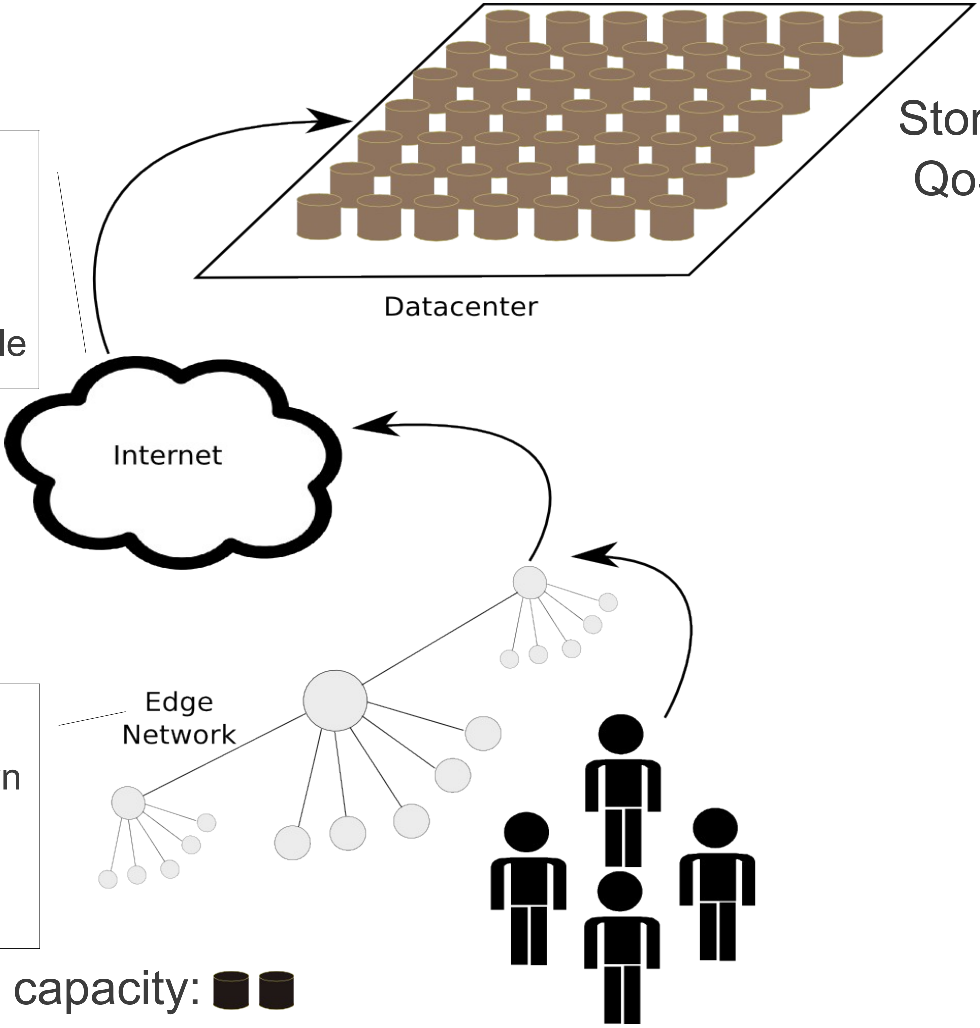


Proximity: highest
Network capacity: well-known
Energy consumption: $
Load: light
Spare resources: available

Storage capacity: ⛁⛁

… at the edge of the network

Proximity: ???
Network capacity: ???
Energy consumption: $$$
Load: potentially heavy
Spare resources: unpredictable

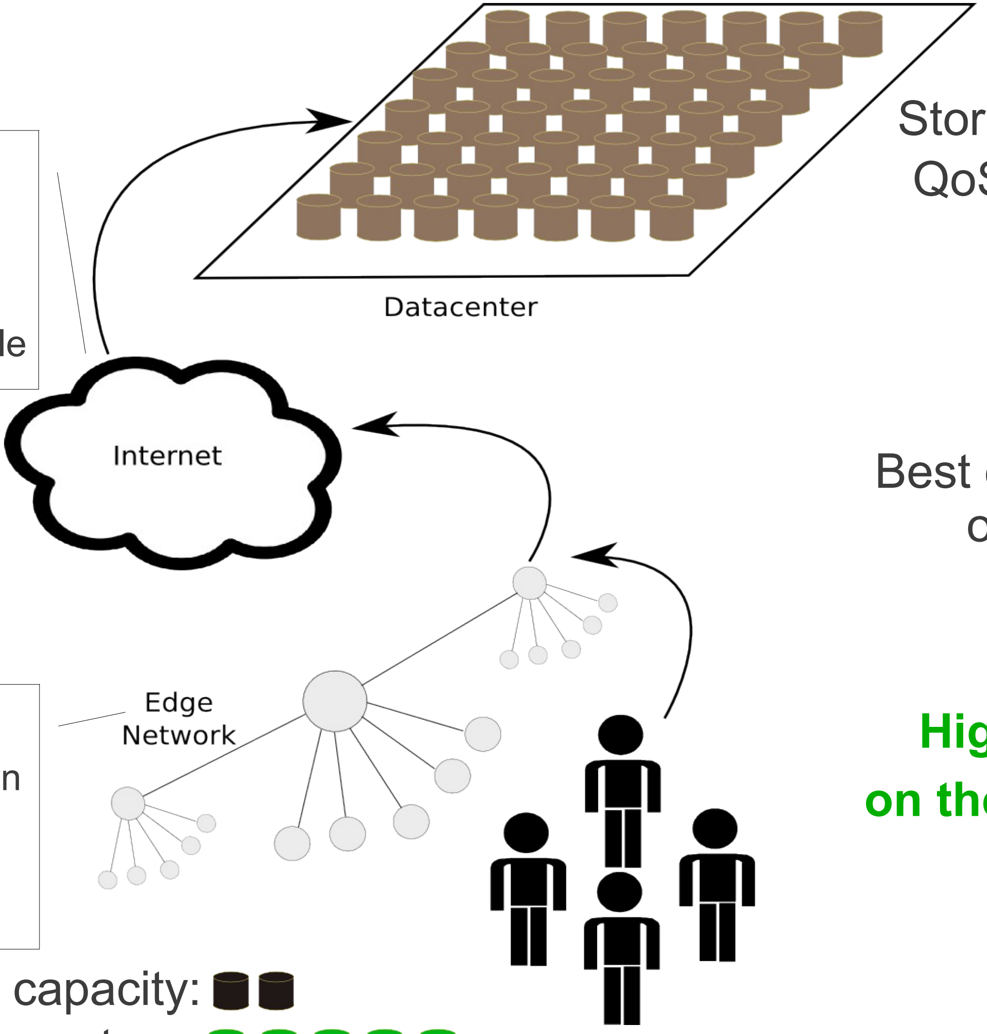


Storage capacity: ■ ■ ■ ■ ■
QoS guarantees: 🟩 🟩

Proximity: highest
Network capacity: well-known
Energy consumption: $
Load: light
Spare resources: available

Storage capacity: ■ ■
QoS guarantees: 🟩 🟩 🟩 🟩 🟩

… at the edge of the network



Proximity: ???
Network capacity: ???
Energy consumption: $$$
Load: potentially heavy
Spare resources: unpredictable

Storage capacity: ⬛⬛⬛⬛⬛
QoS guarantees: 🟩🟩

Proximity: highest
Network capacity: well-known
Energy consumption: $
Load: light
Spare resources: available

Storage capacity: ⬛⬛
QoS guarantees: 🟩🟩🟩🟩🟩

Best effort storage service
on the datacenter

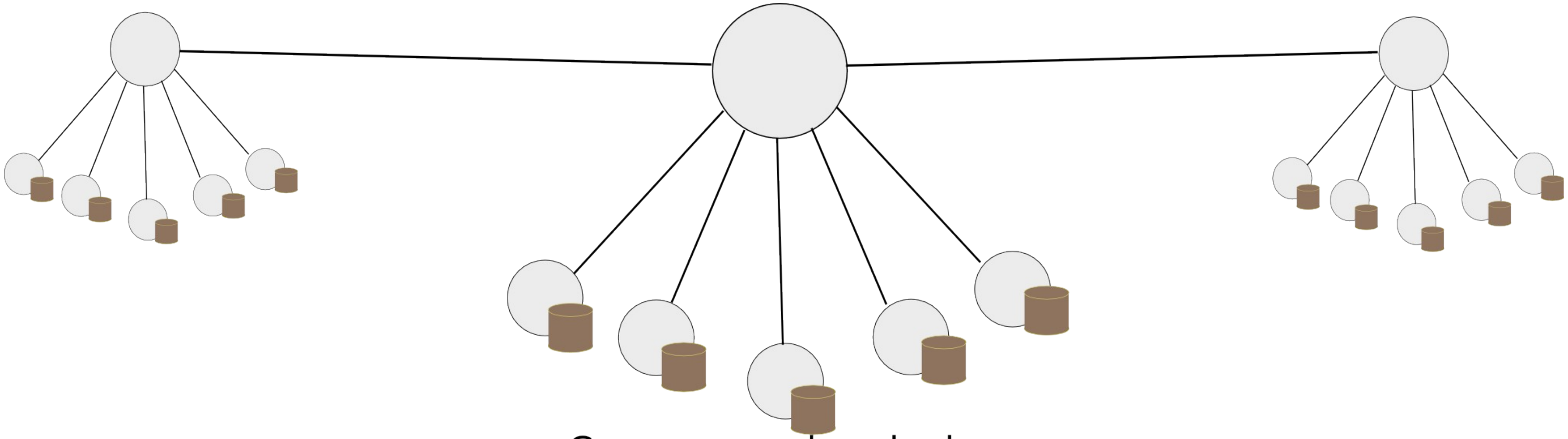
X

High quality storage
on the Edge Network

Hot questions

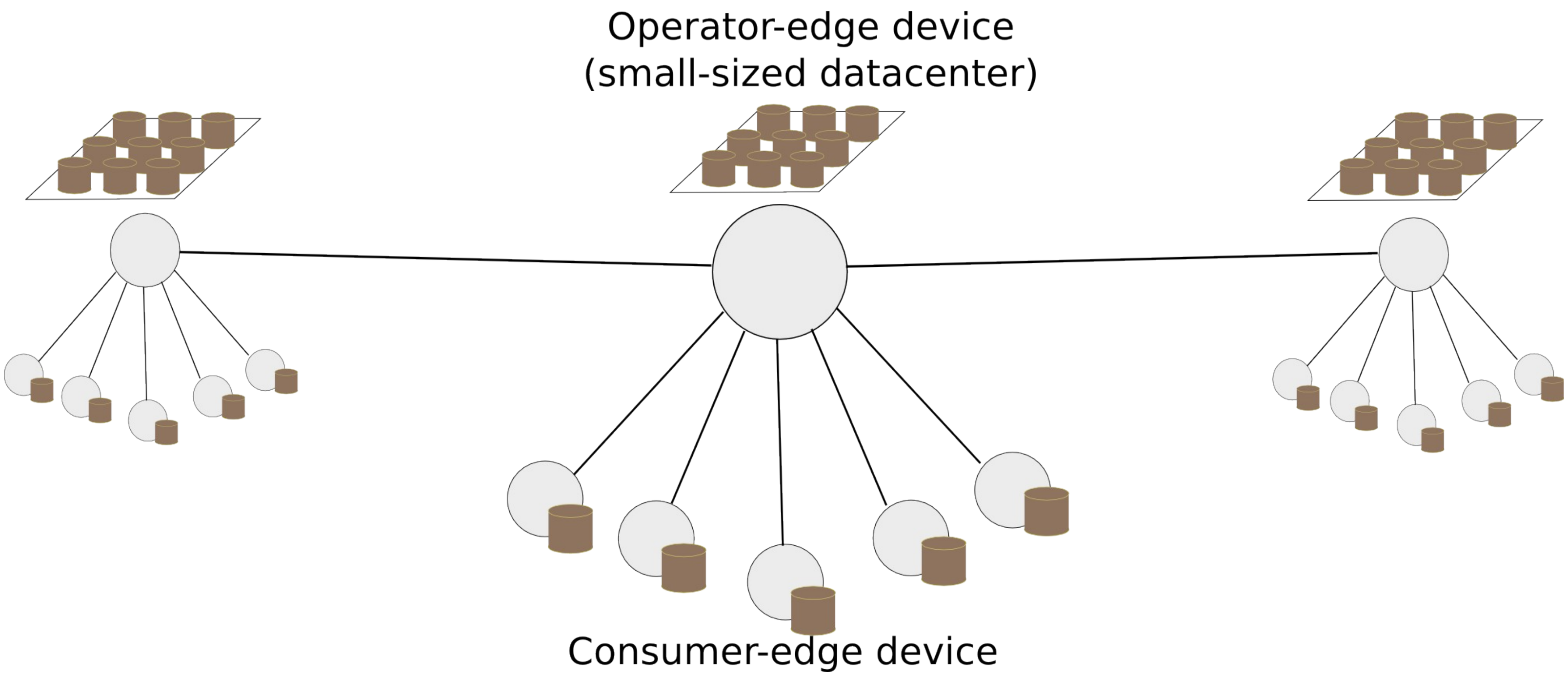
- How do we handle edge network devices for object-based storage systems?
- Where do we place clients' objects?
- How many replicas per object should the system create?
- How could we prevent SLA violations and optimize edge resources utilization?

Caju

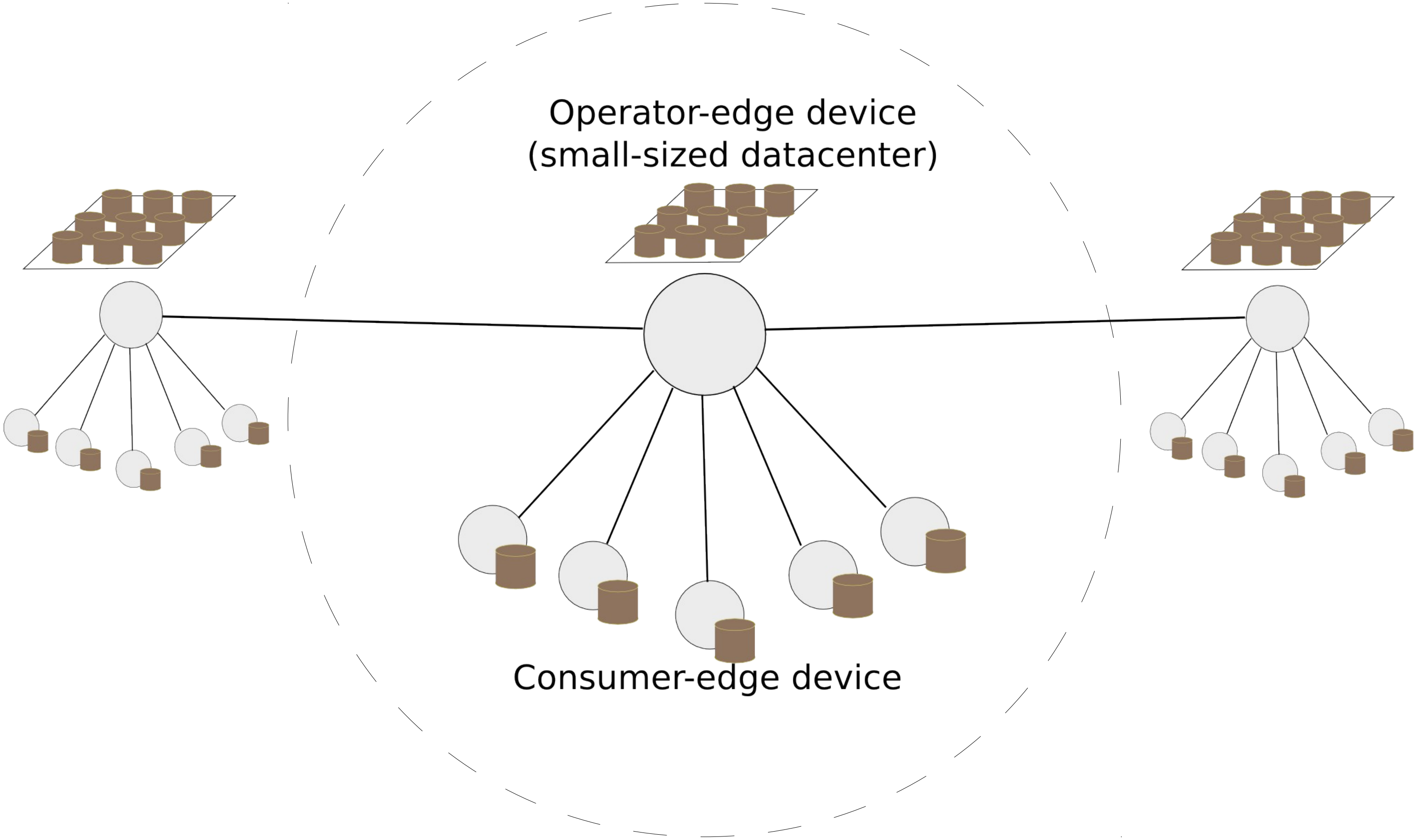


Consumer-edge device

Caju


Operator-edge device
(small-sized datacenter)
Consumer-edge device

Caju

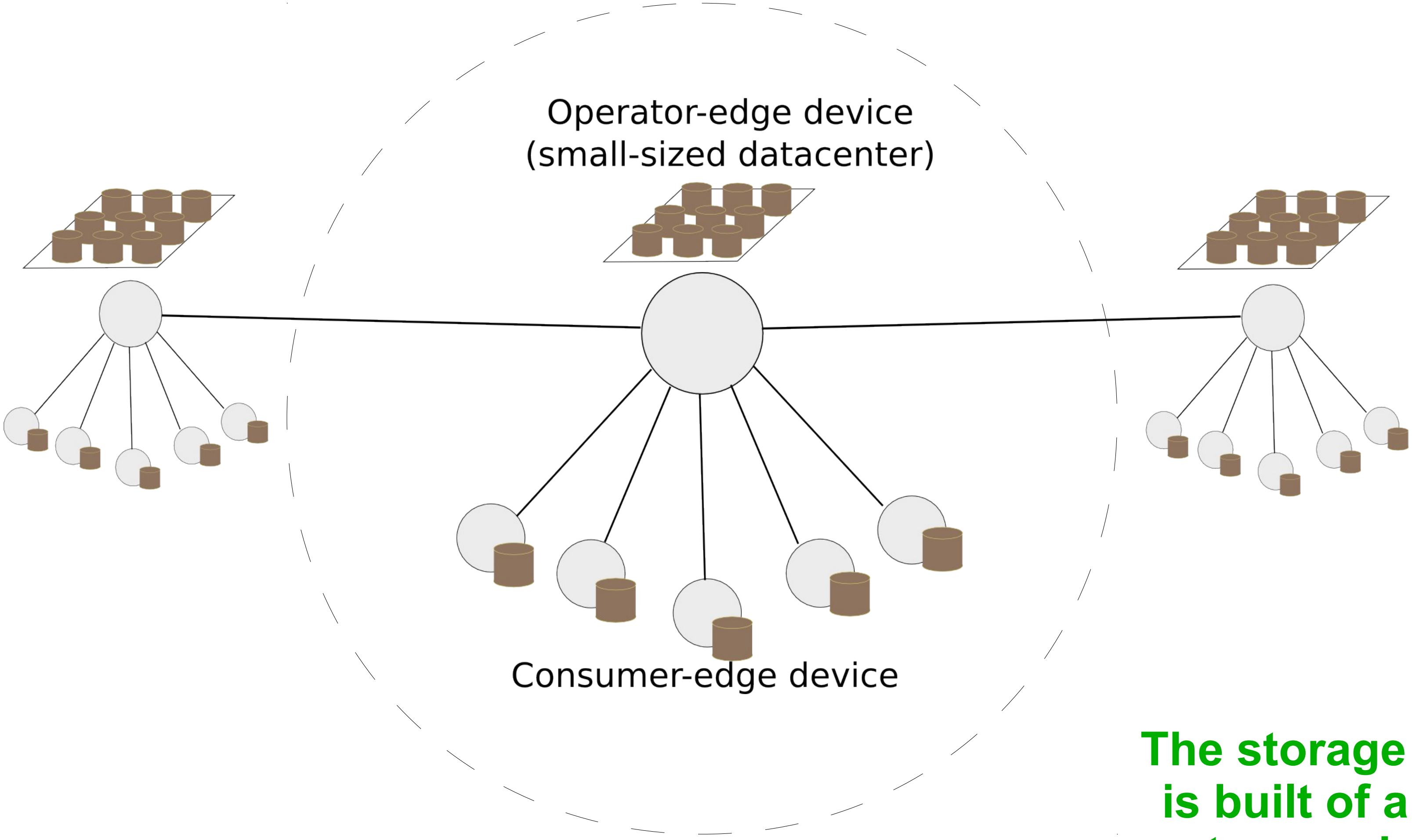


Operator-edge device (small-sized datacenter)

Consumer-edge device

Storage domain

Caju



Storage domain

The storage system is built of a set of storage domains

Caju: Model

- Storage domain, storage nodes and objects:
  - A storage domain *i*, *i* ∈ {1, 2, . . . , I} has storage capacity of S_i and throughput T_i . Each storage domain has a set J_i of J storage elements, *j* ∈ {1, 2, . . . , J}, partitioned in two distinct classes: C_0 for operator-edge class, and consumer-edge C_c for consumer-edge class, where $|C_0| >> |C_c|$
  - Storage capacity:

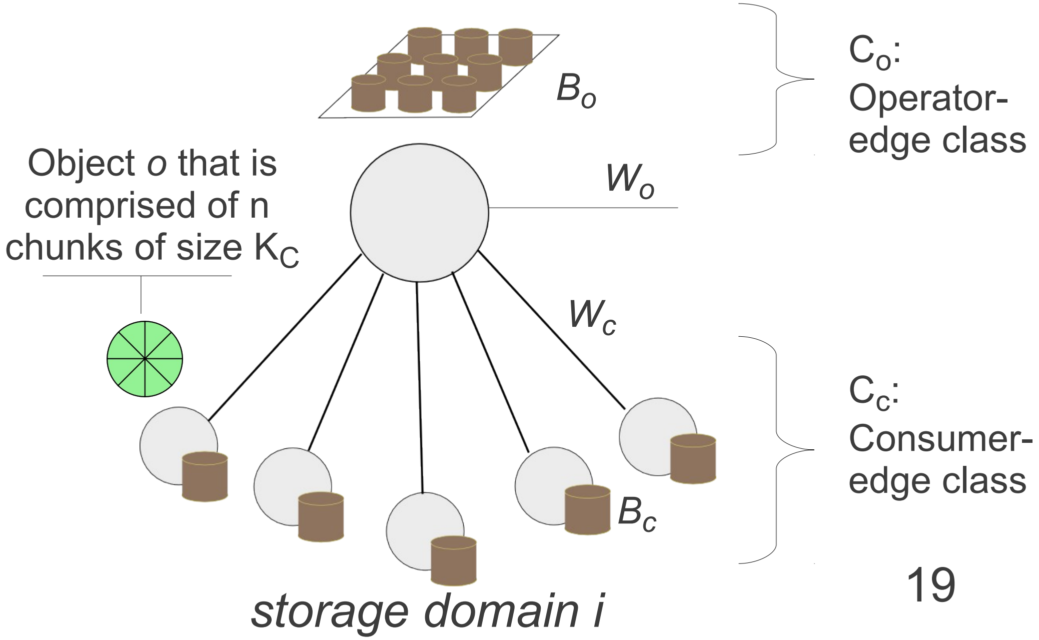
$$s_{ij} = \begin{cases} B_o & \text{if } j \in \mathcal{C}_o; \\ B_c & \text{if } j \in \mathcal{C}_c. \end{cases}$$

$$S_i = \sum_{j=1}^{J} s_{ij} \quad i \in \{1, 2, \ldots, I\}$$

  - Network capacity

$$b_{ij} = \begin{cases} W_o & \text{if } j \in \mathcal{C}_o; \\ W_c & \text{if } j \in \mathcal{C}_c. \end{cases}$$

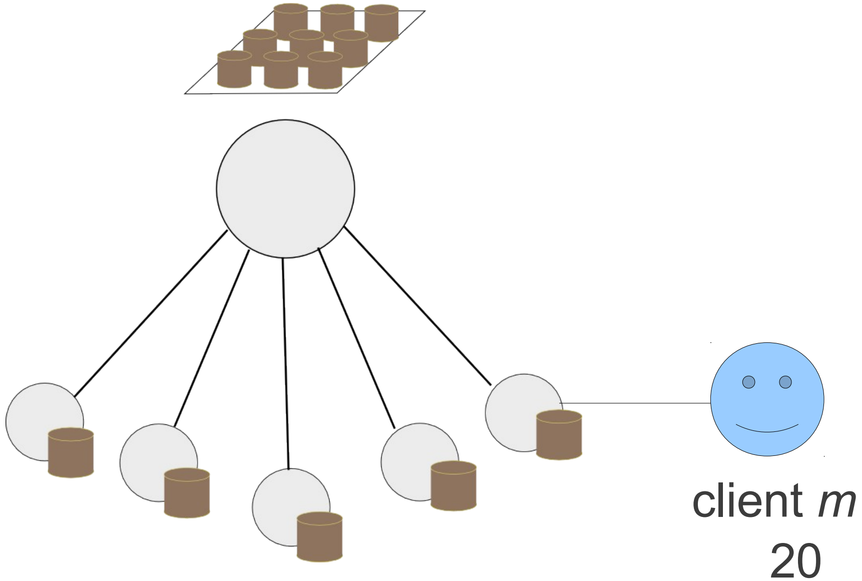
$$\sum_{j \in \mathcal{C}_c} b^u_{ij} \leq W_l$$


B_o
C_o: Operator-edge class
Object *o* that is comprised of n chunks of size K_C
W_o
W_c
B_c
C_c: Consumer-edge class
storage domain i

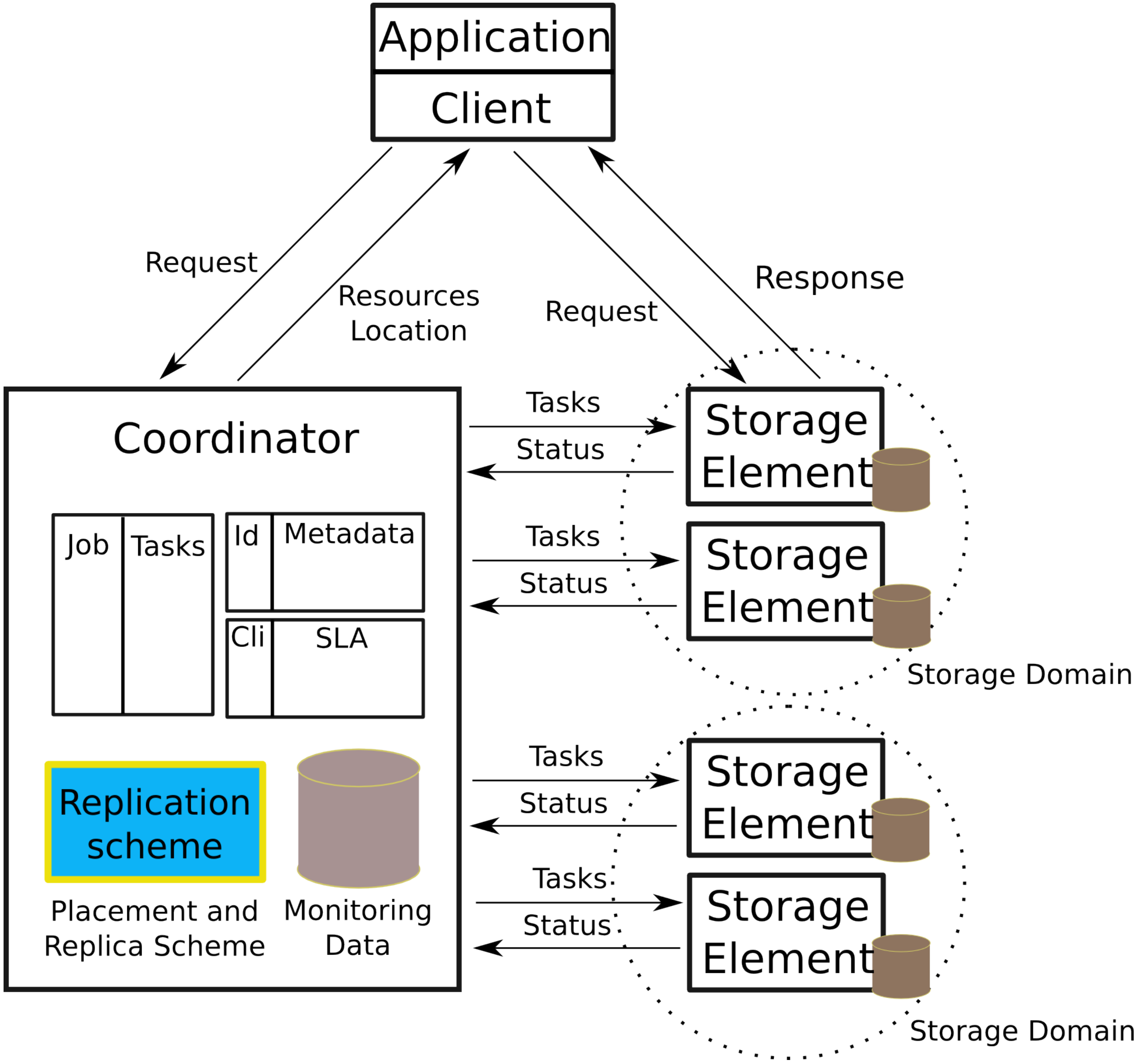
Caju: Model

- Clients and requests:
  - Any client *m* is connected to the system through a consumer-edge device, and assigned to a home storage domain
  - The set of all request R, where R = [R_G, R_P, R_R, R_D], meaning GET, PUT, REPLICATE, and DELETE respectively. Clients are able to do any number of request r^c from [R_G, R_P, R_D] towards objects.

  - Clients' SLA
    - Transfer rate: λ_m^s
    - Minimum percentage of successful requests: P_m^a

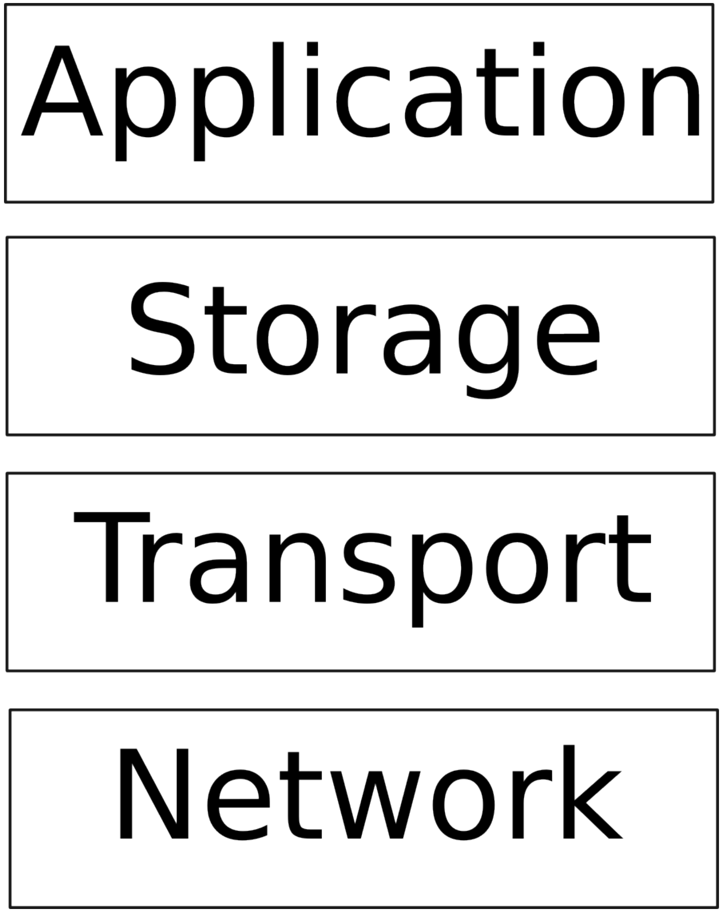


Caju: Main functional blocks



Simulations

- Protocol stack (on PeerSim)



Application

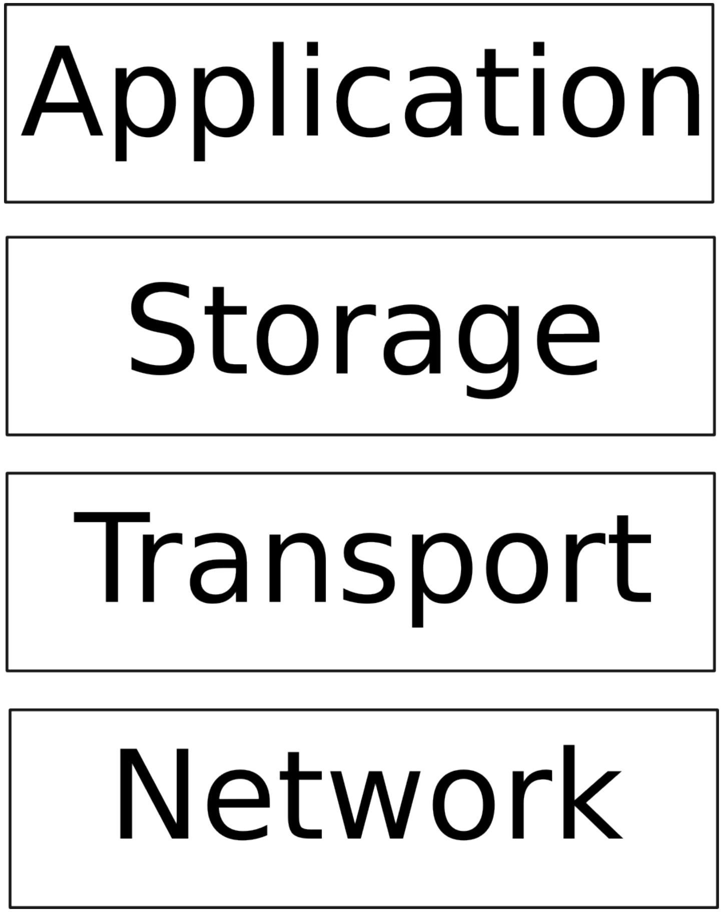
Storage

Transport

Network

Simulations

- Protocol stack (on PeerSim)



Application

Target service: Multi-purpose object-based storage, and sharing
Workload: Users, objects, interactions, and SLA

Storage

Handle requests: Jobs and tasks, storage resources
Quality Control: Content popularity, SLA, and storage/network resource allocation

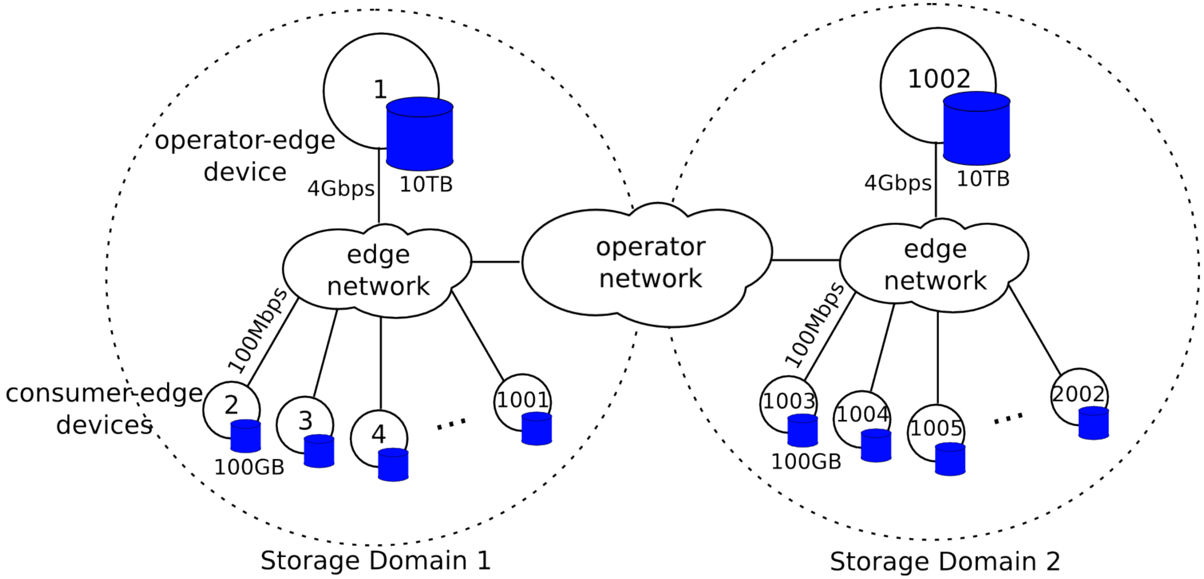
Transport

Communication interface: keep source-destination map, collect and export network flow information, handle network connection events

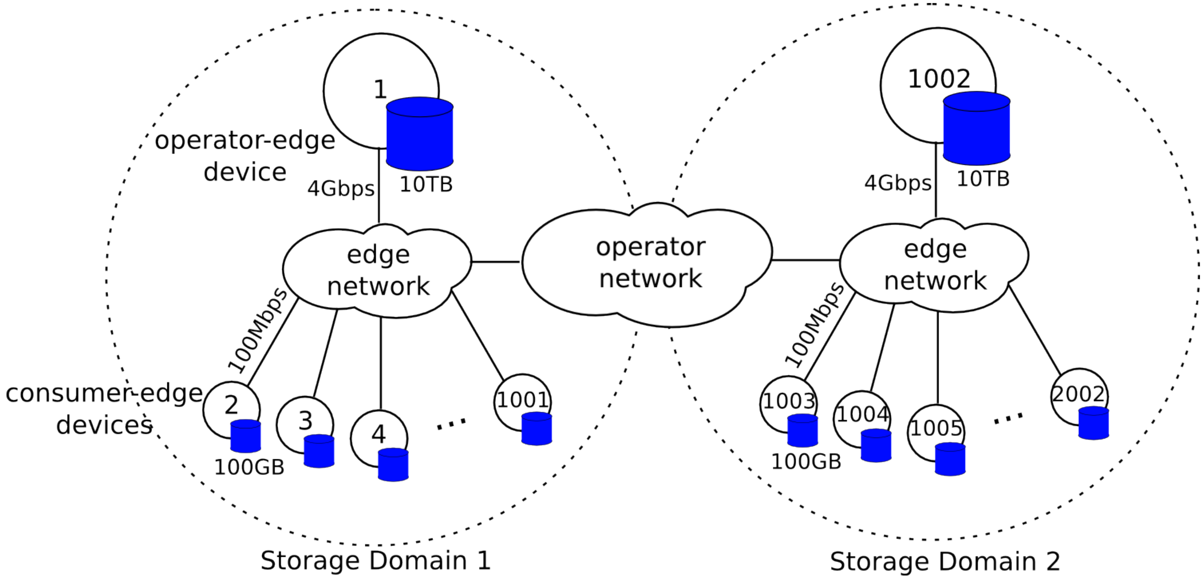
Network

Network functionalities: data transfer between nodes, network resources, fair-sharing bandwidth, reservation, deadline enforcement

Evaluation: Scenario



Evaluation: Scenario



Three SLA contracts:

Rate (chunks/second)	%clients
41	40
21	40
14	20

Workload	
Requests per client	uniform
Experiment duration	1h 12min
Object size (follows Pareto)	shape=5 lower bound=70MB upper bound=1GB (mean 93MB)
Mean requests per second	50
Requests division	5% for PUTs 95% for GETs
Popularity growth (follows Weibull)	shape=2 scale $\propto$ duration

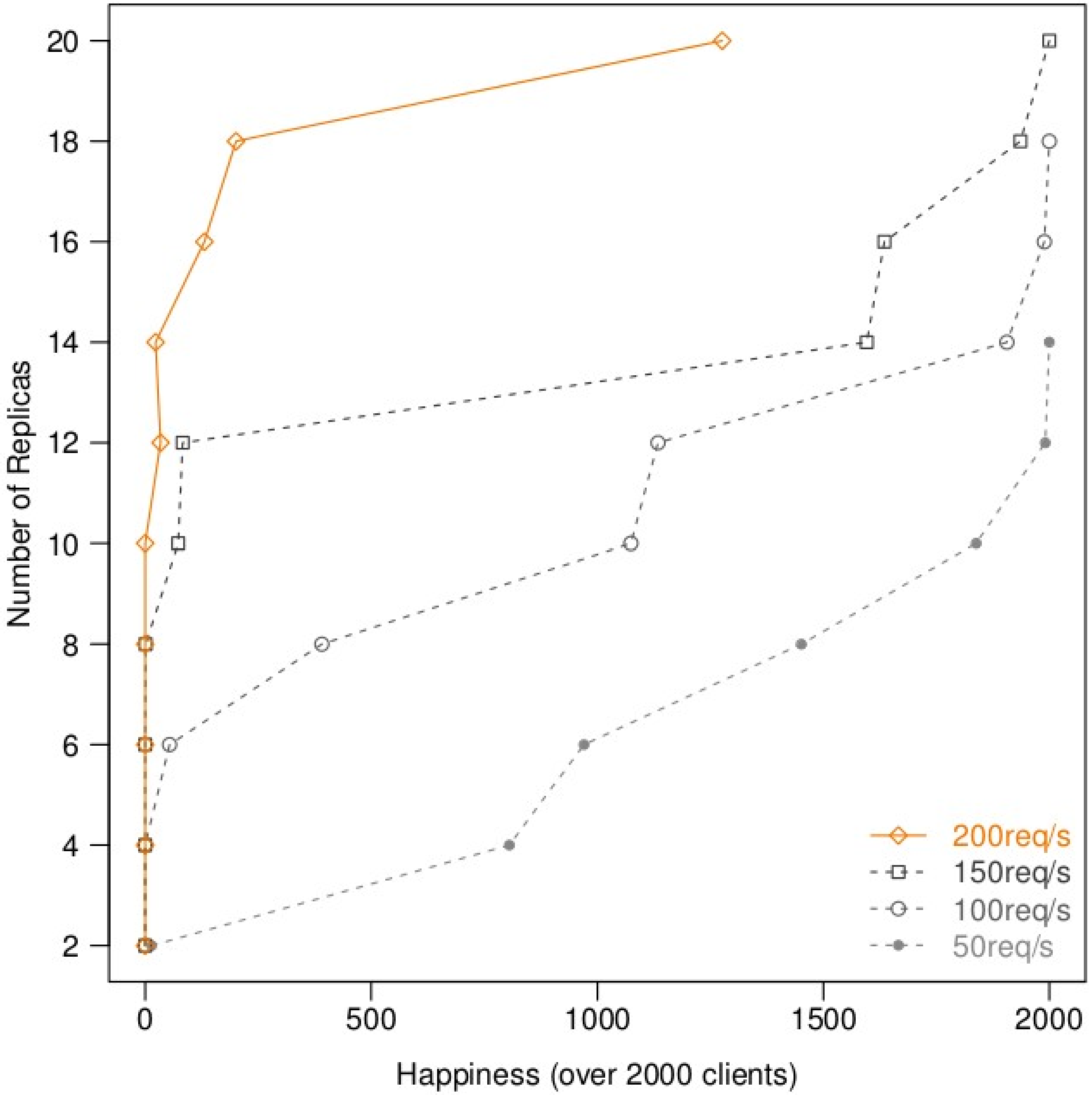
Content popularity (Zipf-Mandelbrot)	shape=0.8
	cutoff=# of objects

PUTs (Poisson)	λ=PUTs/s

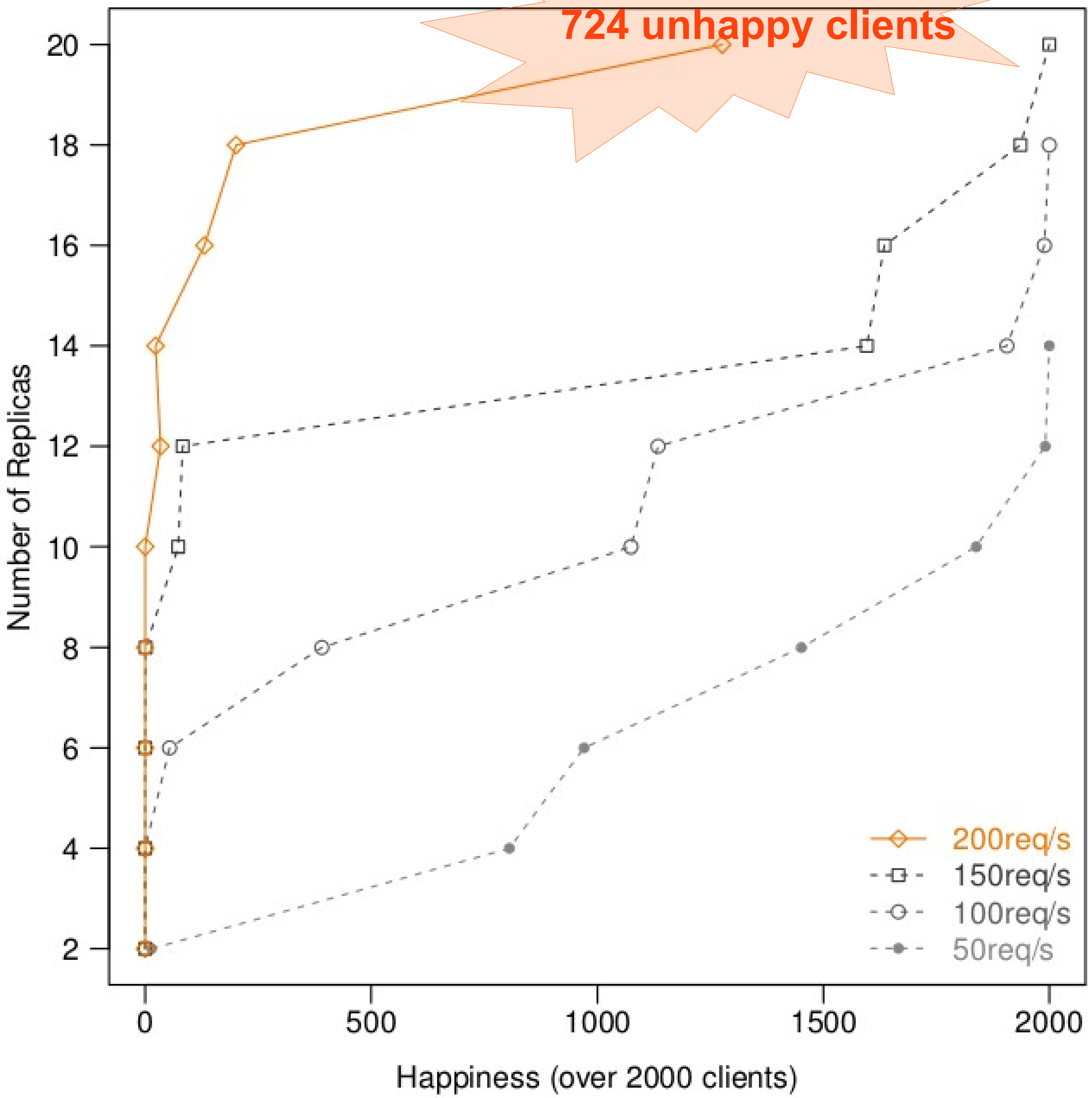
Evaluation: replication schemes

- Uniform replication scheme with fixed number of replicas
  - Replication: fixed number of replicas *n*
  - Request scheduling: request might be served by at most *K* nodes with equal load; $r = \min(n, K)$
- Non-collaborative LRU caching
  - Replication:
    - a new replica is created whenever a client, connected to a consumer-edge device, performs a GET
    - LRU replacement is enforced according to a static percentage of the local storage capacity for caching
  - Scheduling similar to that used in uniform approach

Evaluation

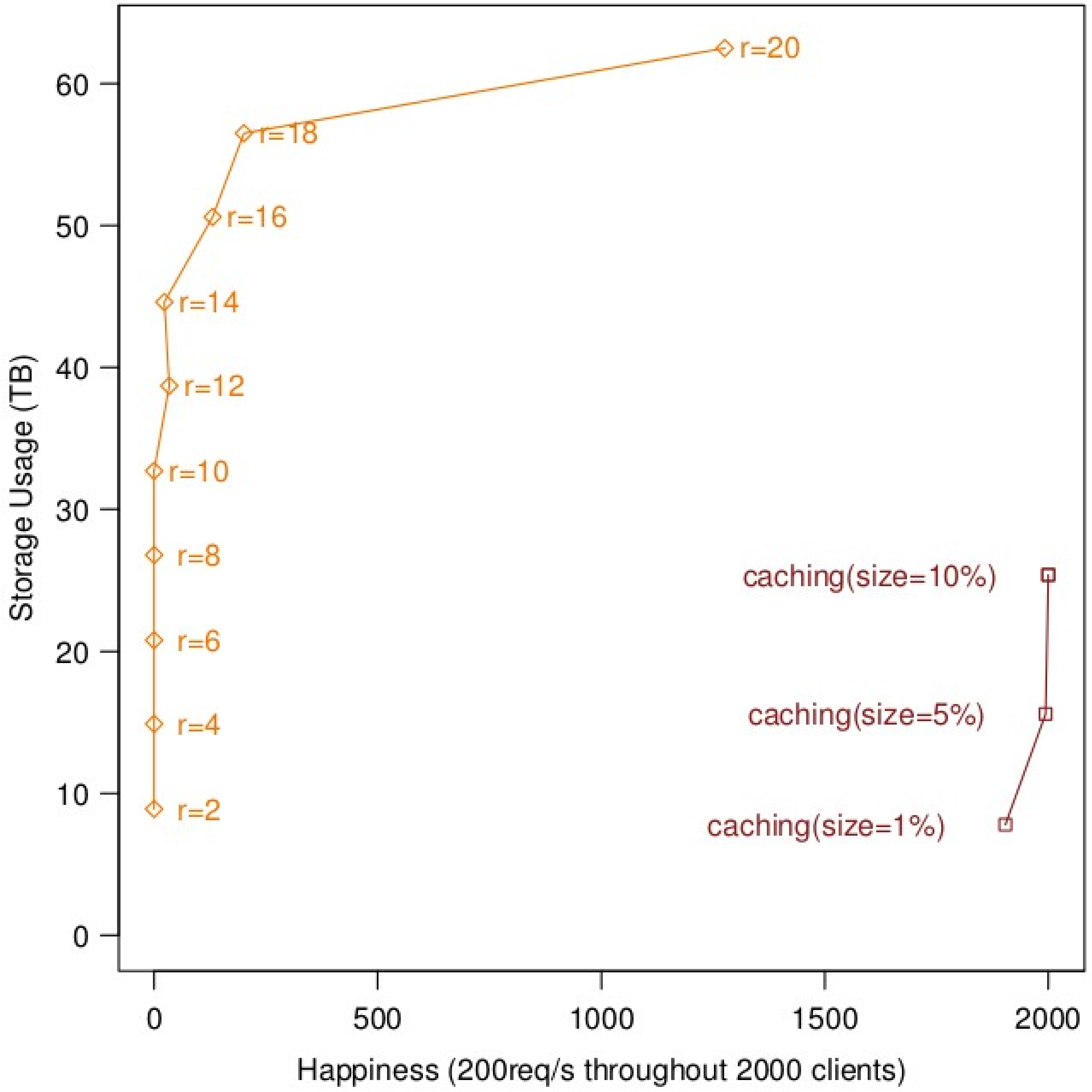

Number of Replicas
20
18
16
14
12
10
8
6
4
2
0
500
1000
1500
2000
Happiness (over 2000 clients)
200req/s
150req/s
100req/s
50req/s

Evaluation



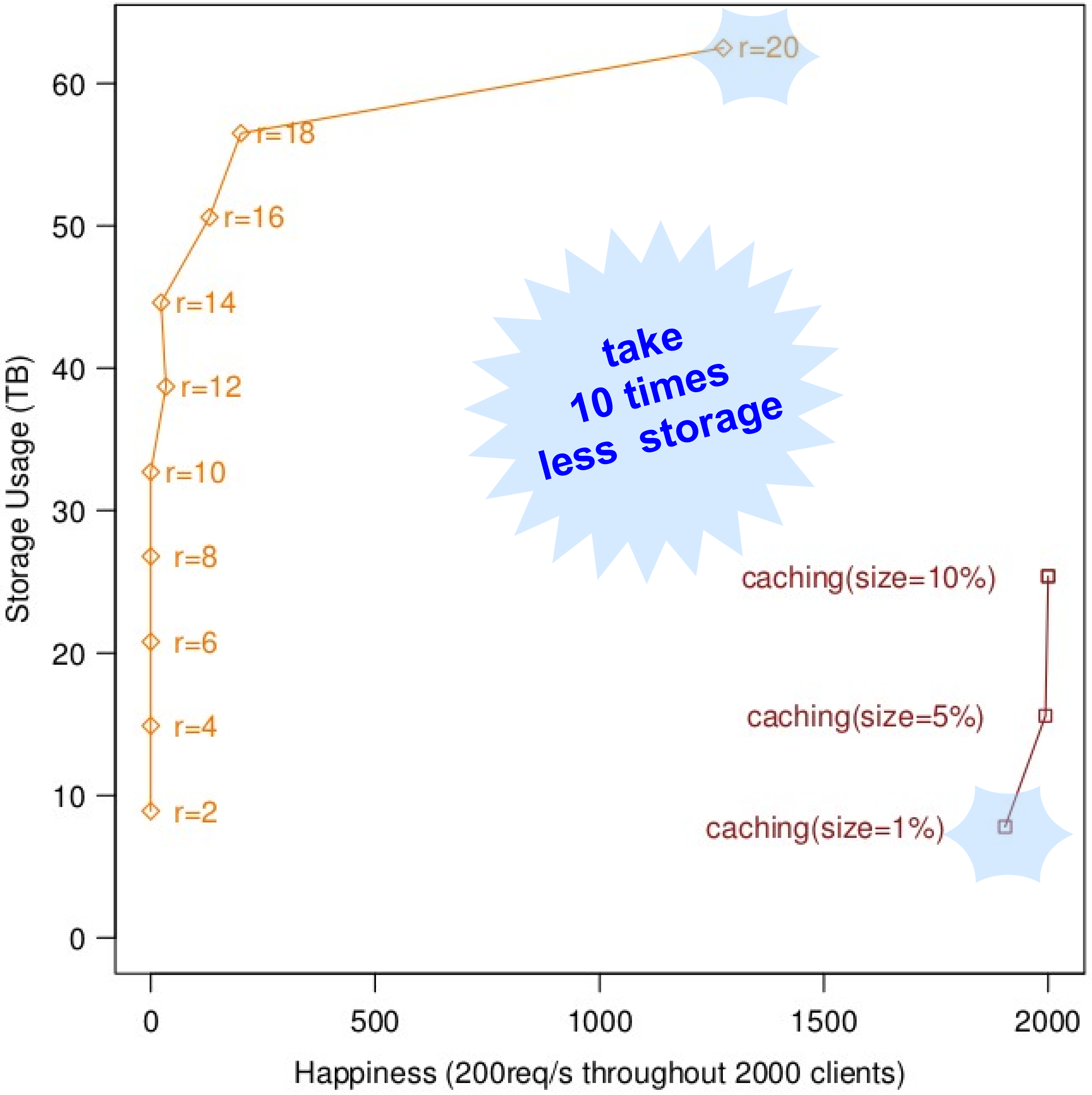
799 violations
724 unhappy clients

Evaluation



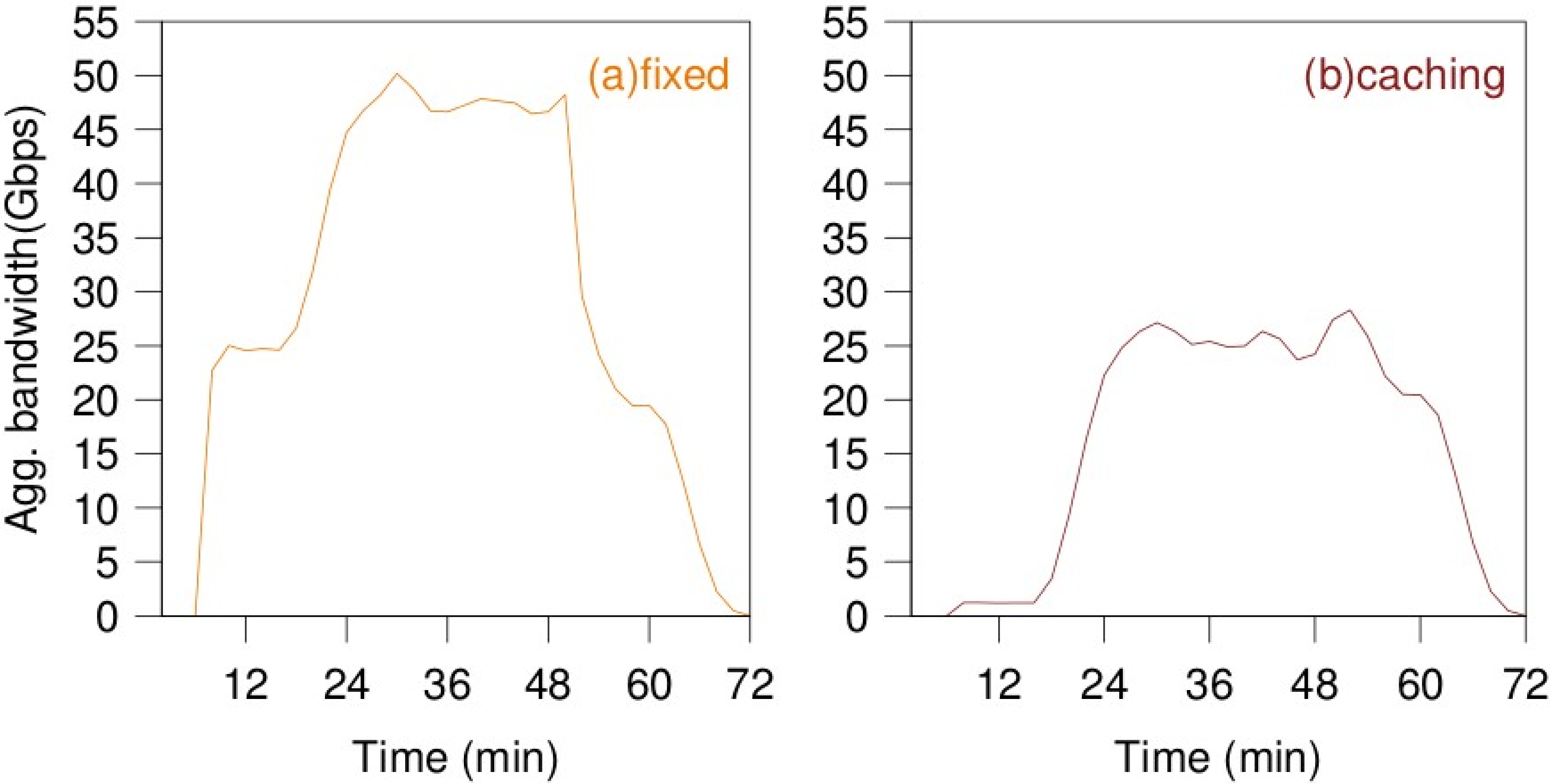
Storage Usage (TB)
0
10
20
30
40
50
60
Happiness (200req/s throughout 2000 clients)
0
500
1000
1500
2000
r=2
r=4
r=6
r=8
r=10
r=12
r=14
r=16
r=18
r=20
caching(size=10%)
caching(size=5%)
caching(size=1%)

Evaluation

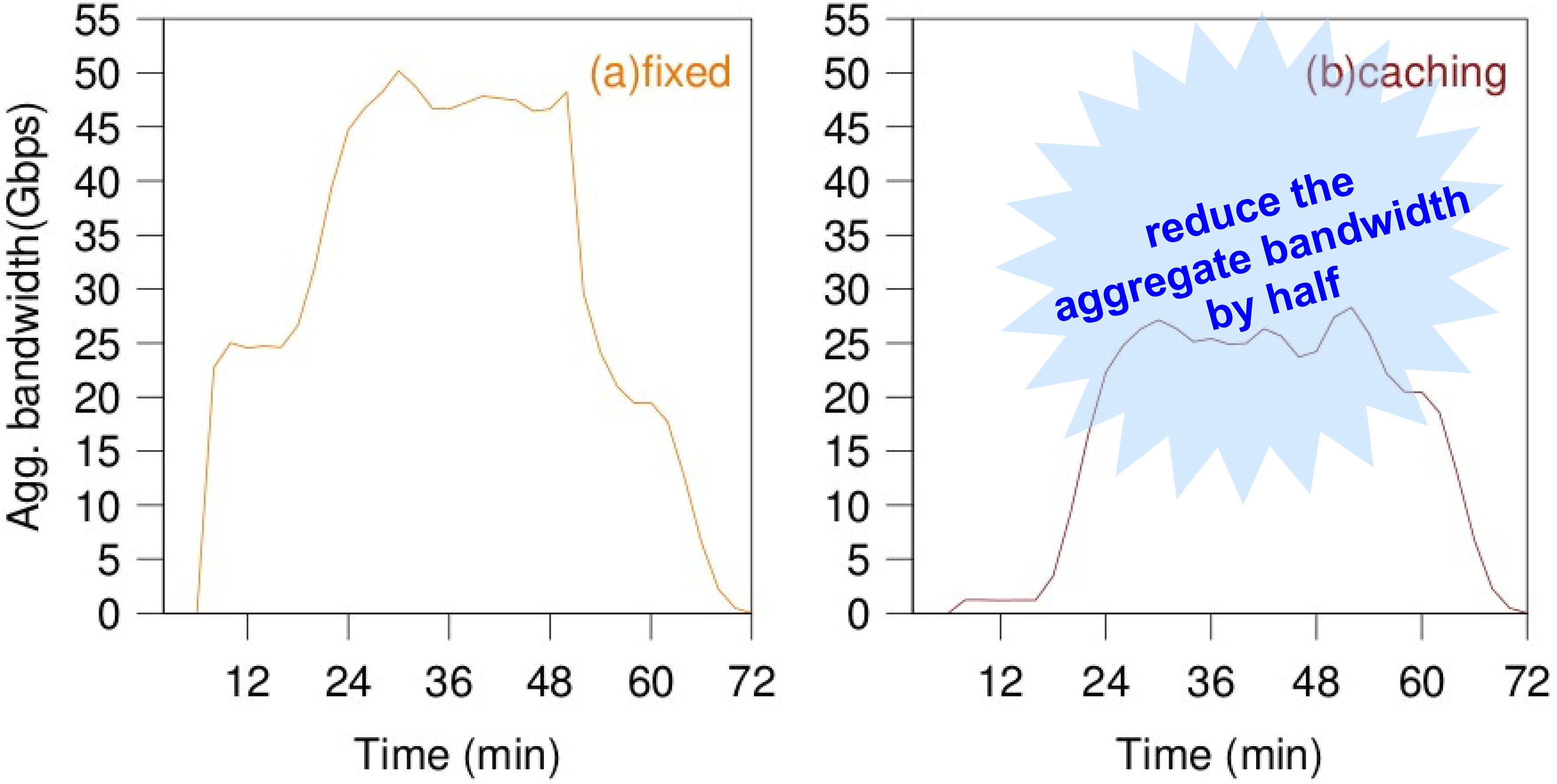


take 10 times less storage

Evaluation

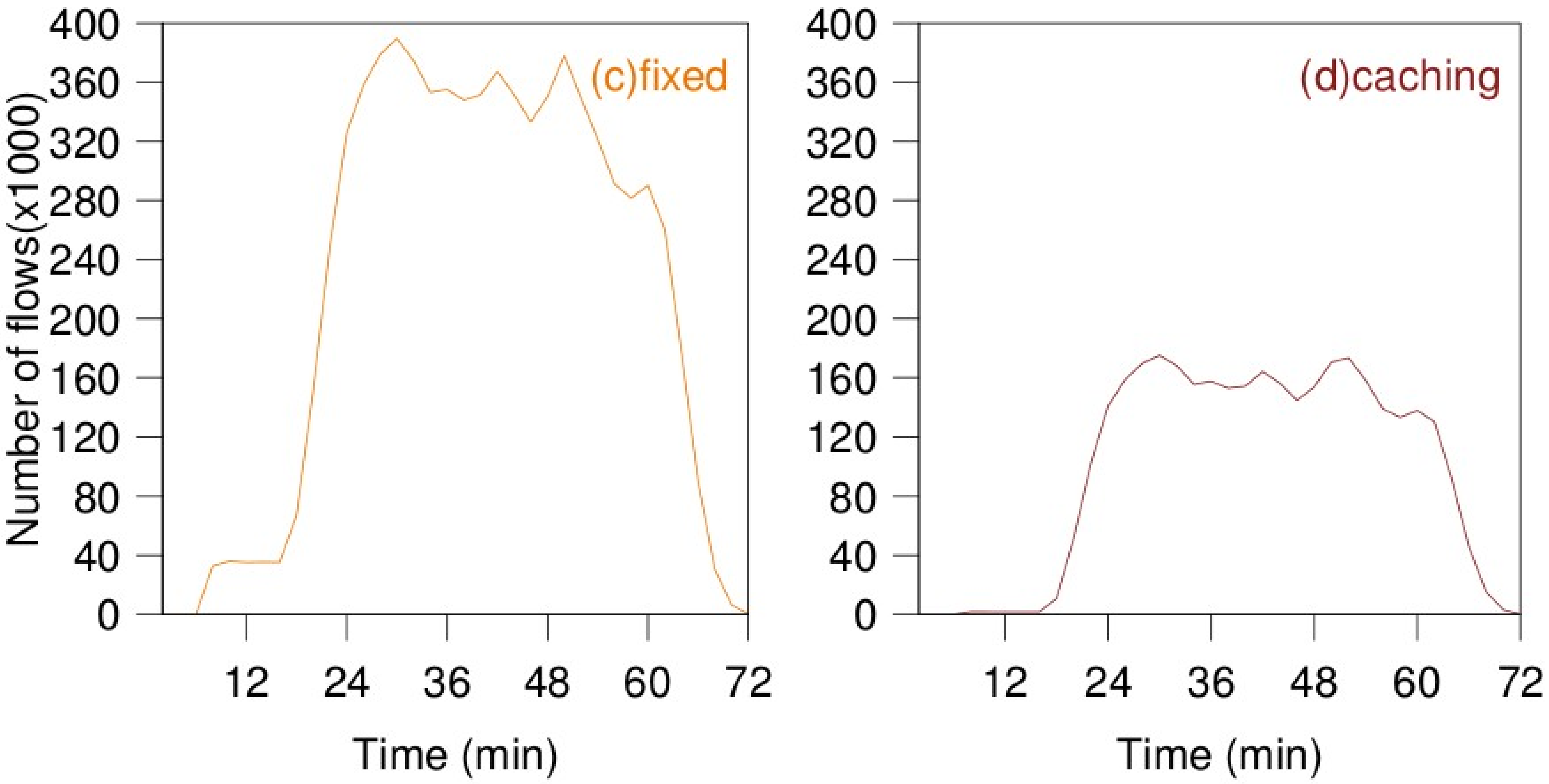

Agg. bandwidth(Gbps)
0
5
10
15
20
25
30
35
40
45
50
55
(a)fixed
(b)caching
12
24
36
48
60
72
Time (min)
Time (min)

Evaluation

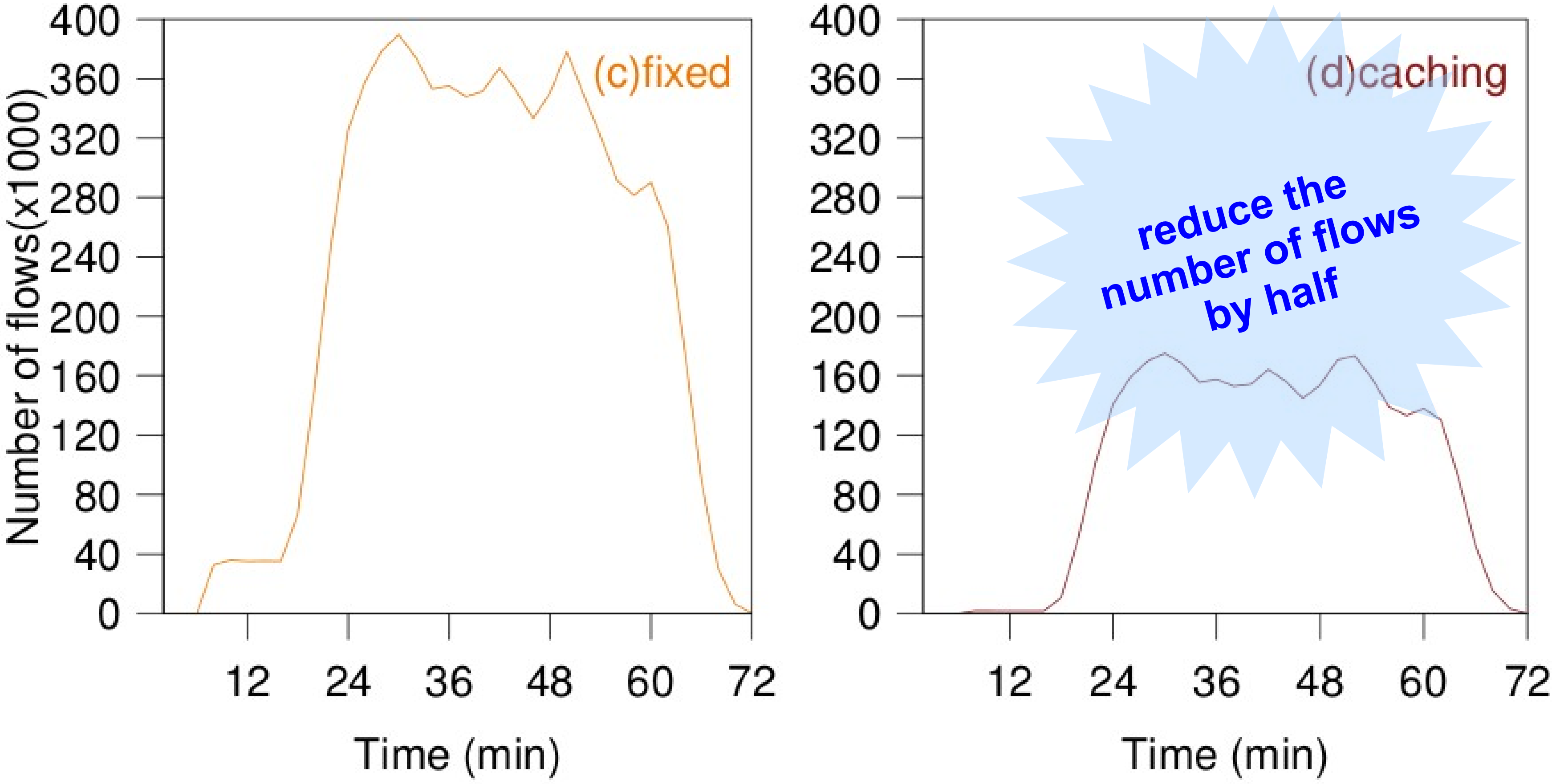


reduce the aggregate bandwidth by half

Evaluation

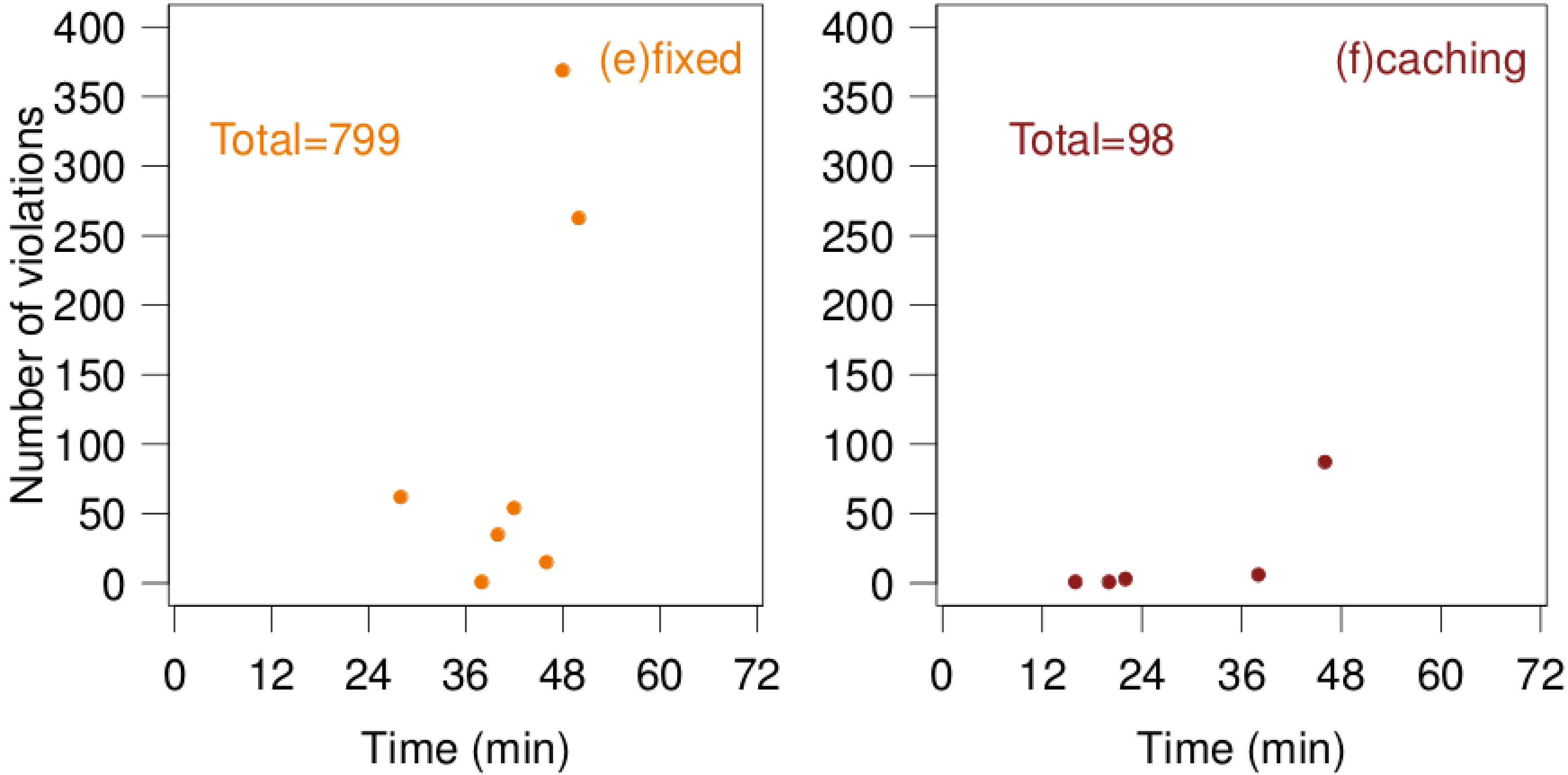

400
360
320
280
240
200
160
120
80
40
0
Number of flows(x1000)
(c)fixed
12
24
36
48
60
72
Time (min)
400
360
320
280
240
200
160
120
80
40
0
(d)caching
12
24
36
48
60
72
Time (min)

Evaluation

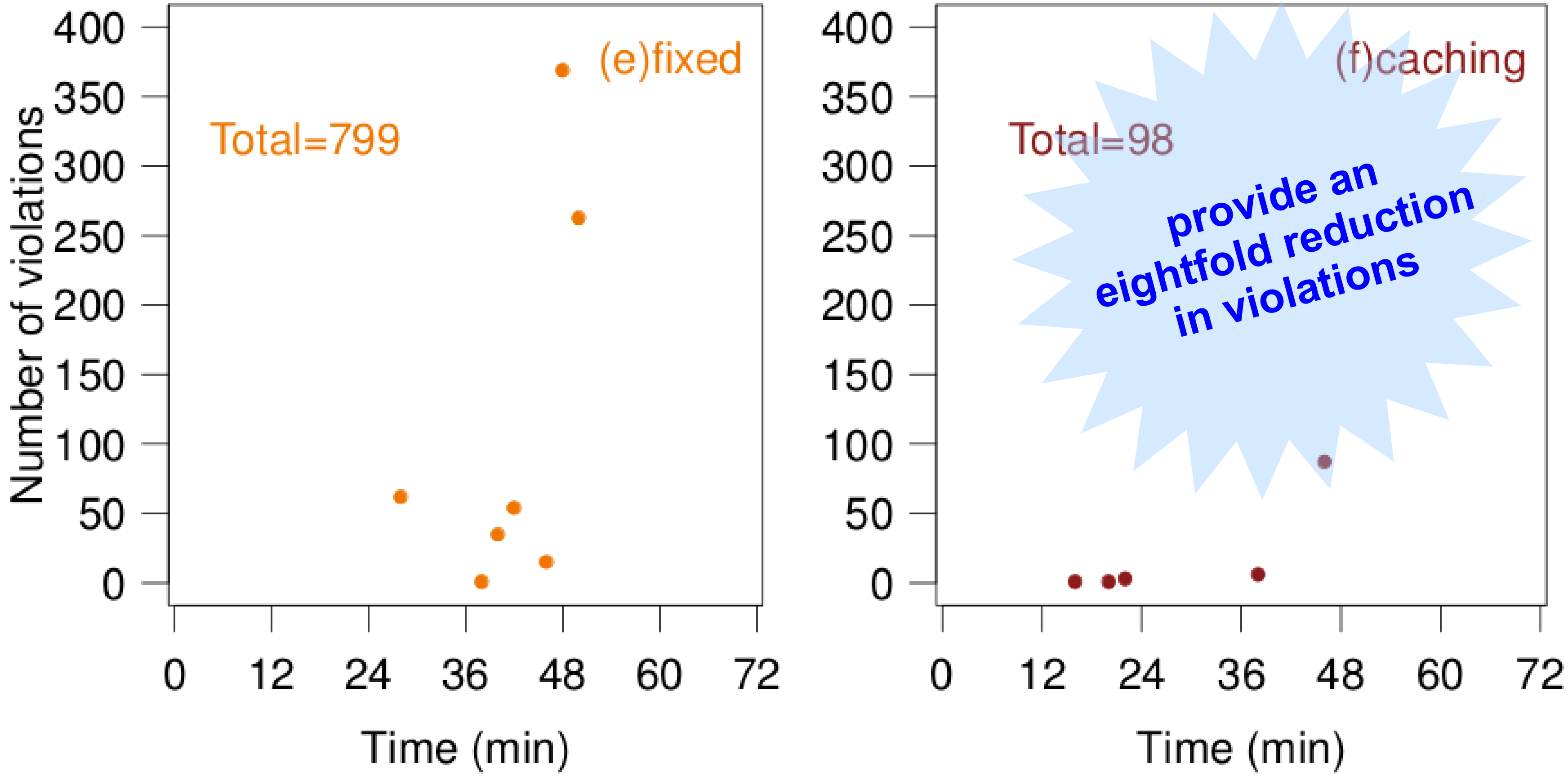


reduce the number of flows by half

Evaluation


Number of violations
400
350
300
250
200
150
100
50
0
(e)fixed
Total=799
(f)caching
Total=98
0
12
24
36
48
60
72
Time (min)
Time (min)

Evaluation

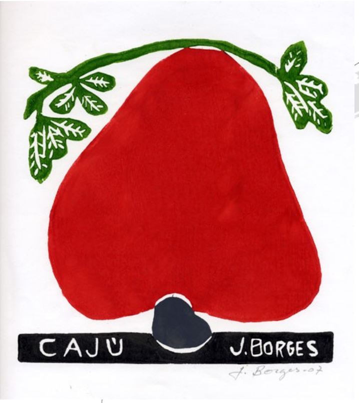


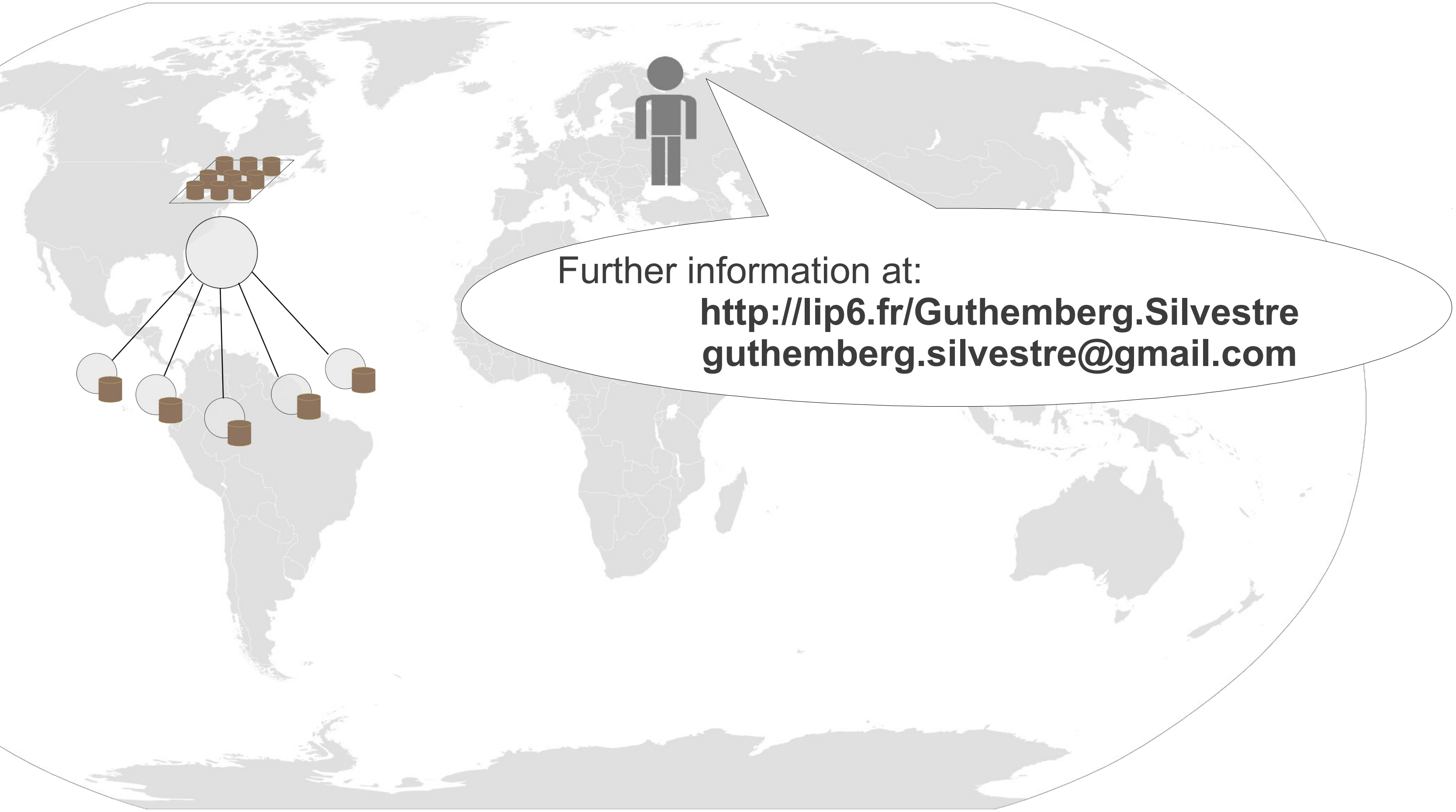
provide an eightfold reduction in violations

Conclusions and perspectives

- Cloud storage has becomes very popular
- Content popularity matters to the efficiency of replication schemes (SLA, storage, network)
- Non-collaborative LRU caching outperforms fixed replication:
  - eightfold reduction in SLA violations
  - requires up to 10 times less storage capacity for replicas
  - reduces aggregate bandwidth and number of flows by half
- Enhance adaptive replication for popular content delivery

Caju: a content distribution system for edge networks





Further information at:
http://lip6.fr/Guthemberg.Silvestre
guthemberg.silvestre@gmail.com