UNIVERSITAS SCIENTIARUM SZEGEDIENSIS

Research Group on Artificial Intelligence

Decentralized Management of Random Walks over a Mobile Phone Network

Árpád Berta and Márk Jelasity
University of Szeged
PDP17

Why decentralization?

- infinite scaleablity
  (IoT, telemedicina, smart devices)
- low budget (startup, commonweal, communities, NGO)
- privacy issue
  (not garanteed, but easier to reach)

Motivations

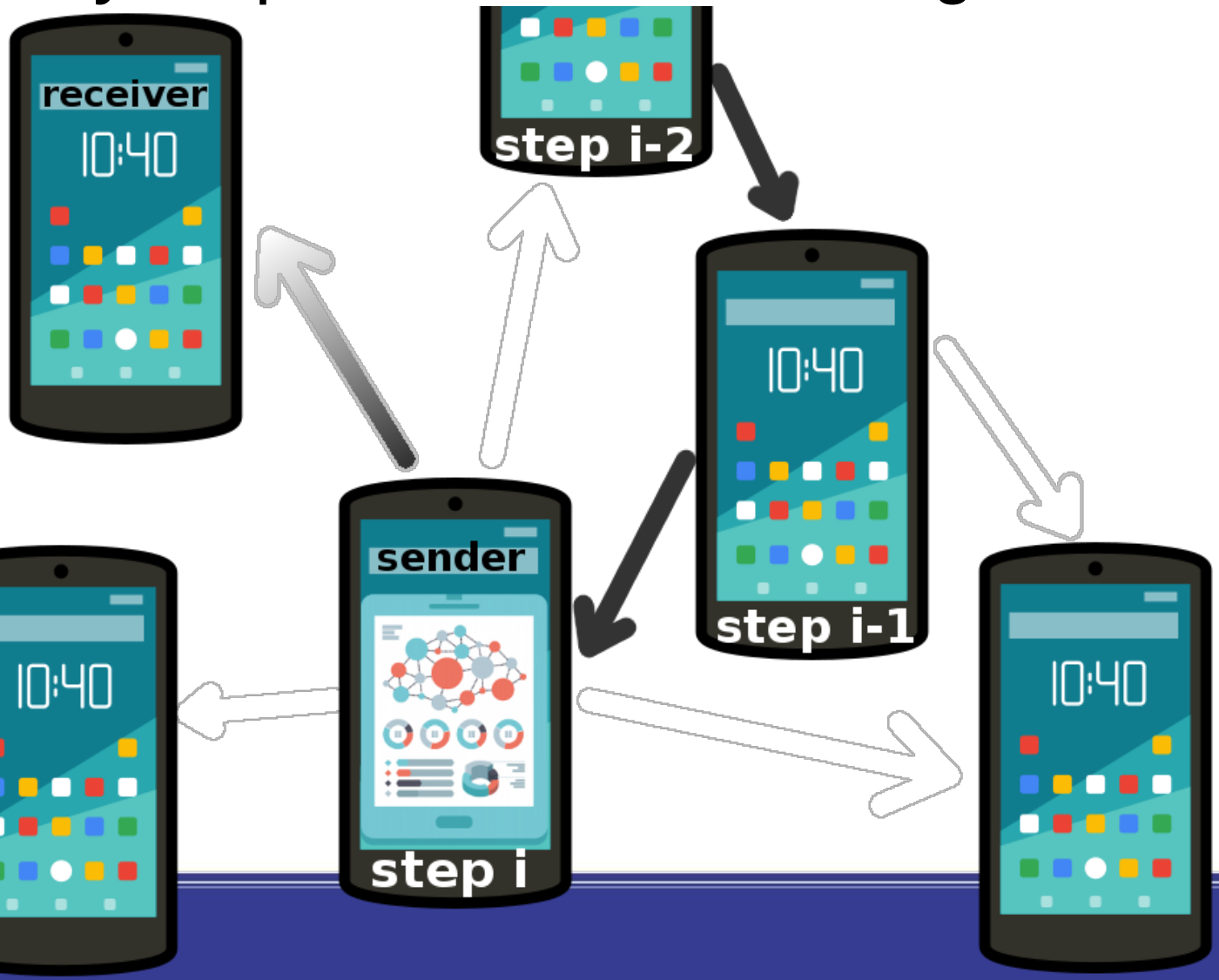
- fully open collaborative environment where those who provide data can also enjoy the benefit of mining the collective data of the community
- platform for independent decentralized data mining tasks that might belong to different users
- tasks: online learning from local data, where the private data never leave the nodes (privacy preserving)
- gossip learning → random walks
- **decentralized management of O(n) random walks**:
  - fault detection
  - restart failed random walks

System Model

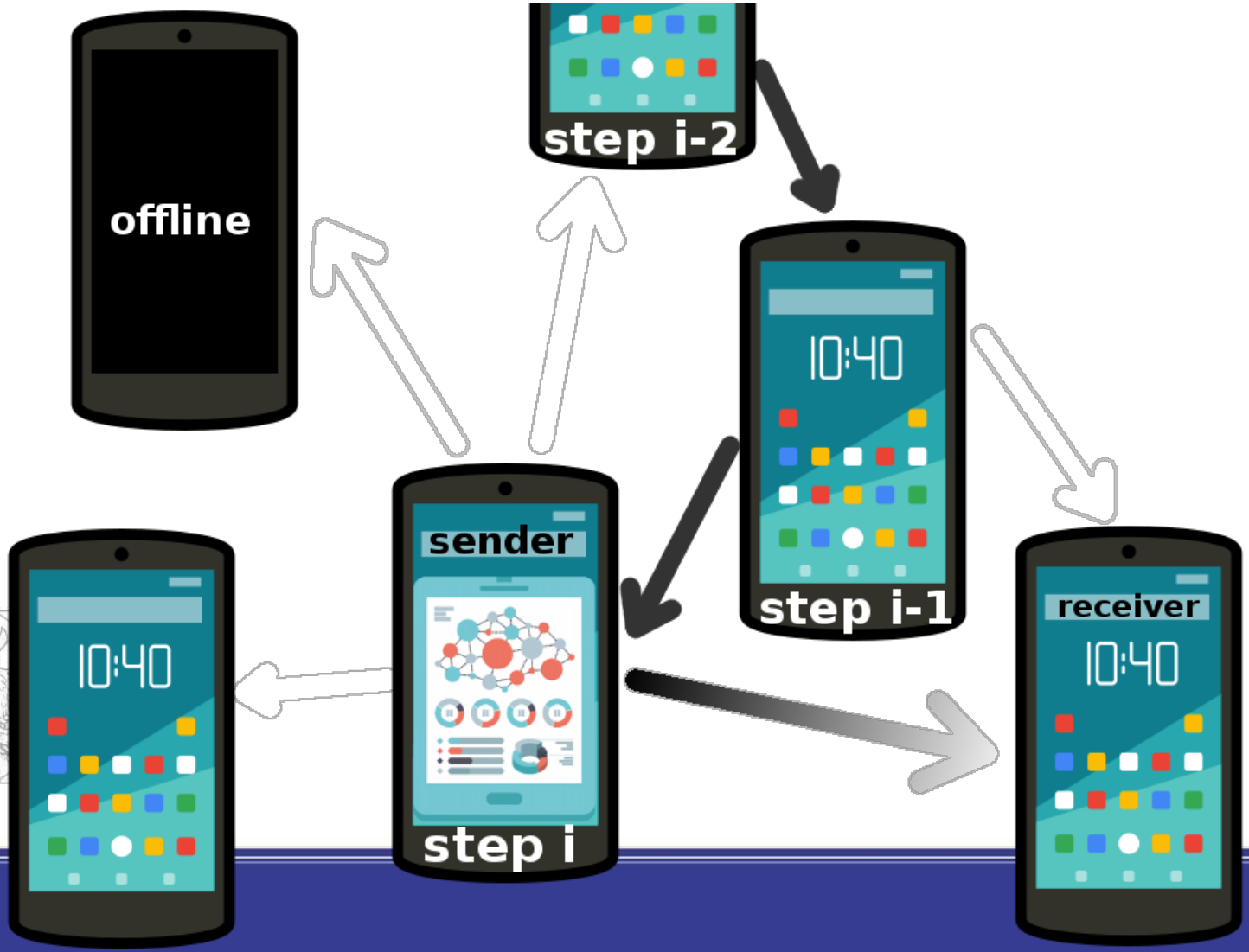
- large set of nodes that communicate with their neighbors via messaging
- reliable transfer protocol – communication fails only if the source or target node fails before transferring the full message
- set of neighbors is a uniform random sample from the network (peer sampling service)
- nodes can leave the network and join again without any prior notice
- no synchronized time

Random walks

step-by-step to next random neighbor

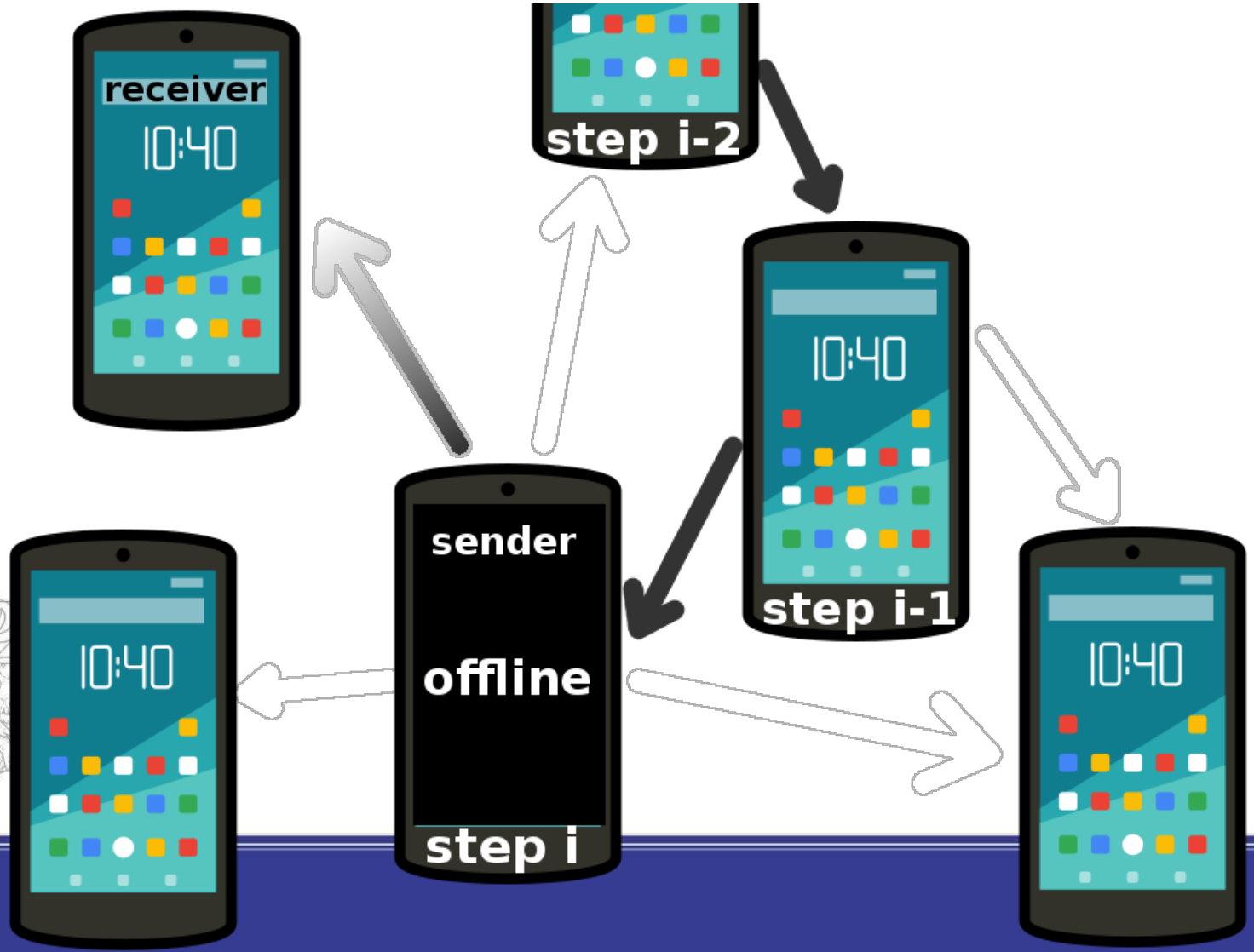


Receiver failed – start send to another online neighbor

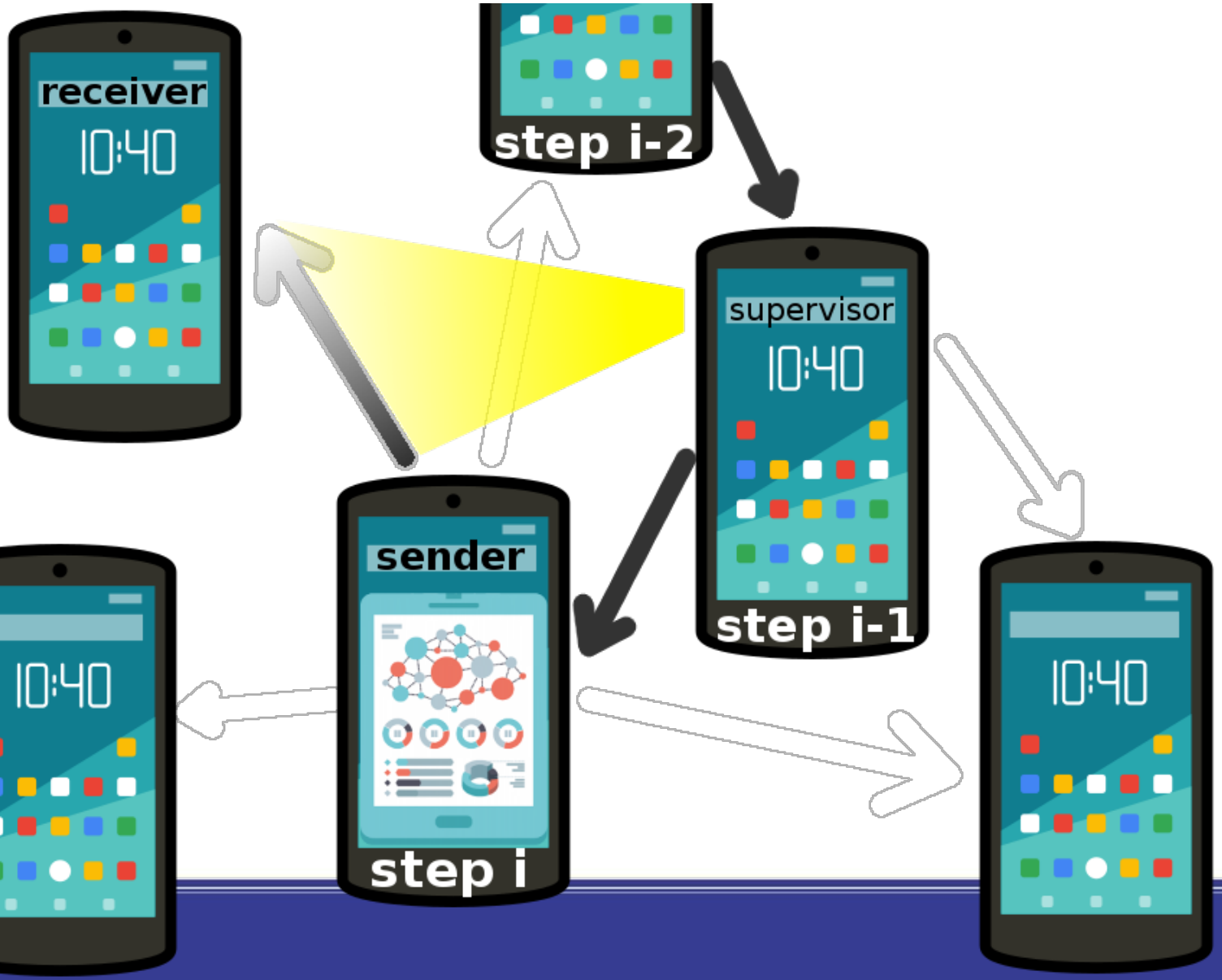


Multiple Random Walk Service

multilevel service for restarting failed random walks



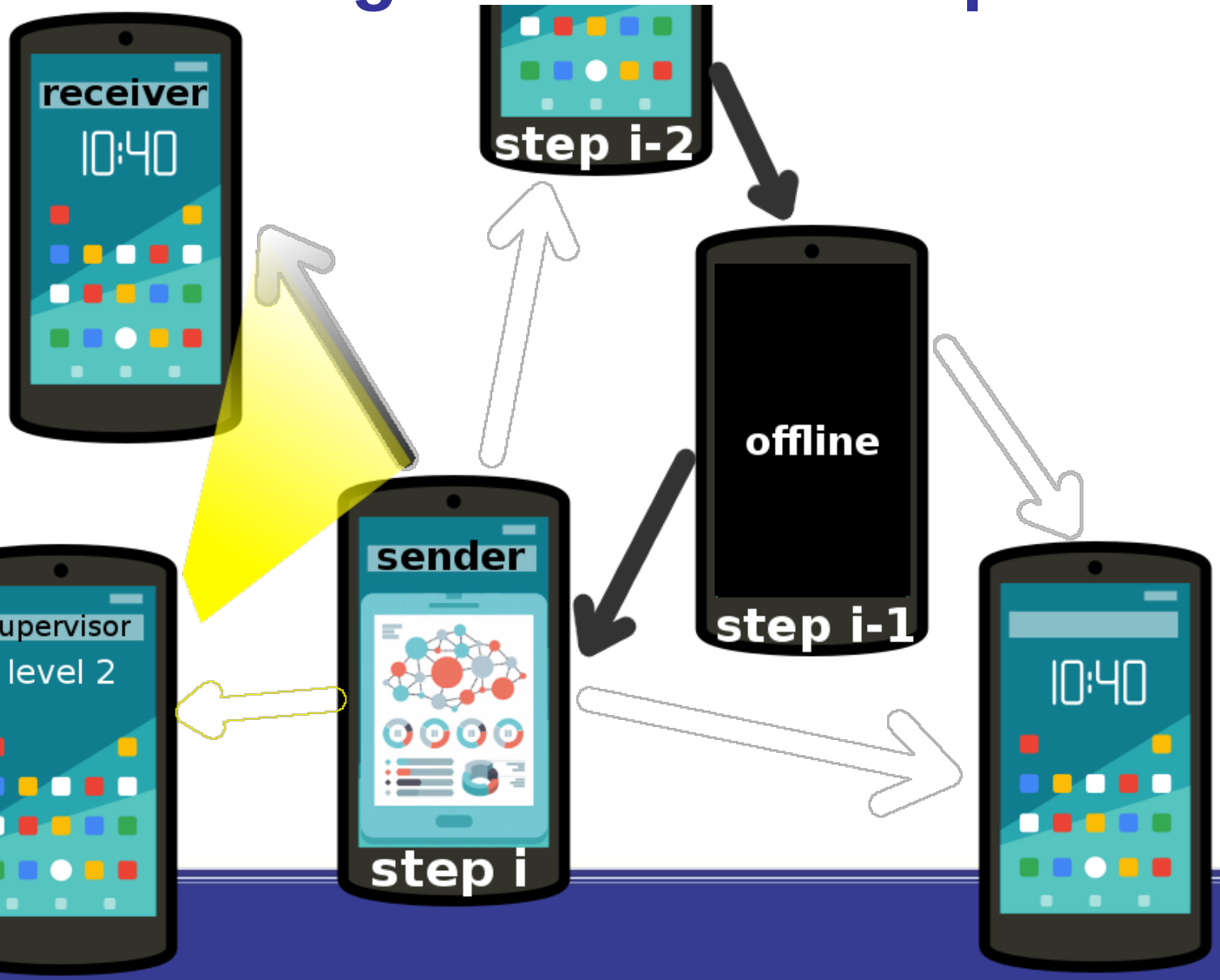
Level 1 – previous node is the supervisor



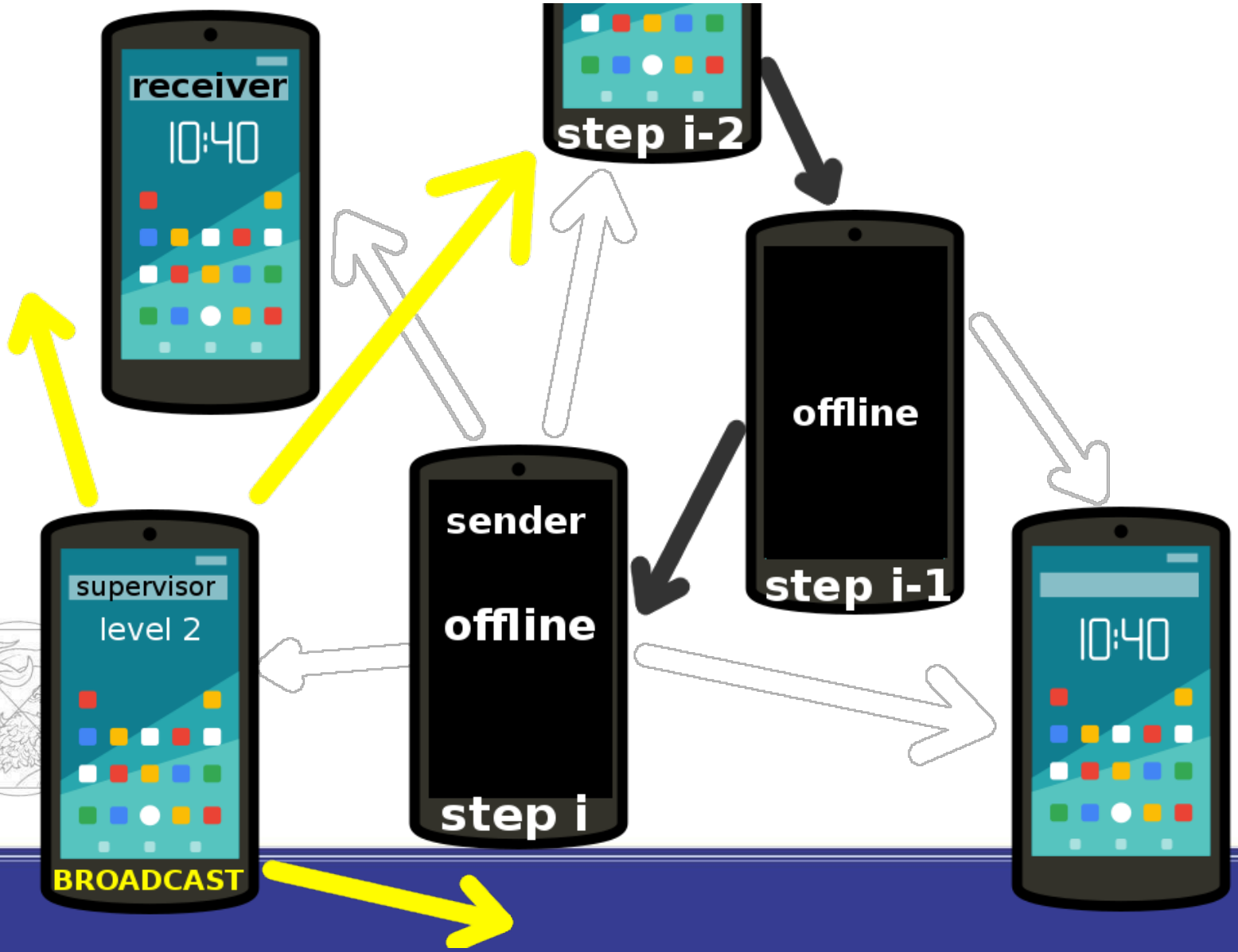
Level 1 restart – supervisor restarts the RW

Level 2 – Level 1 supervisor goes offline

Level 2 – sender node appoints a random neighbor as new supervisor



Level 2 restart – the supervisor broadcasts the failure event



Level 2 restart

- broadcast message has finite hop counter
- every node stores the visited random walk state for a finite time (storage queue)
- nodes obtain the broadcast message and if they have the previous state of the same random walk then switch to restart state
- restart state: use time window to avoid parallel restarts
- broadcast also about the restart decision and the collision of the restarts

Level 3 restart – a semi-centralized restart process

- random walk can report its state to owner regularly (online minimum 10 minutes per day or public cloud service)
- owner of the walk is responsible for restarts the random walk
- extremely rare event

Experimental setup

- peersim simulator
- simulated network sizes: 1000, 100000
- static network, 50 neighbors
- real churn – smart phone trace
  (one day long)
- number of tasks: based on the number of average online nodes (about 0.54n)
- message size: 1 Mbit, 10 Mbit

Intelligence

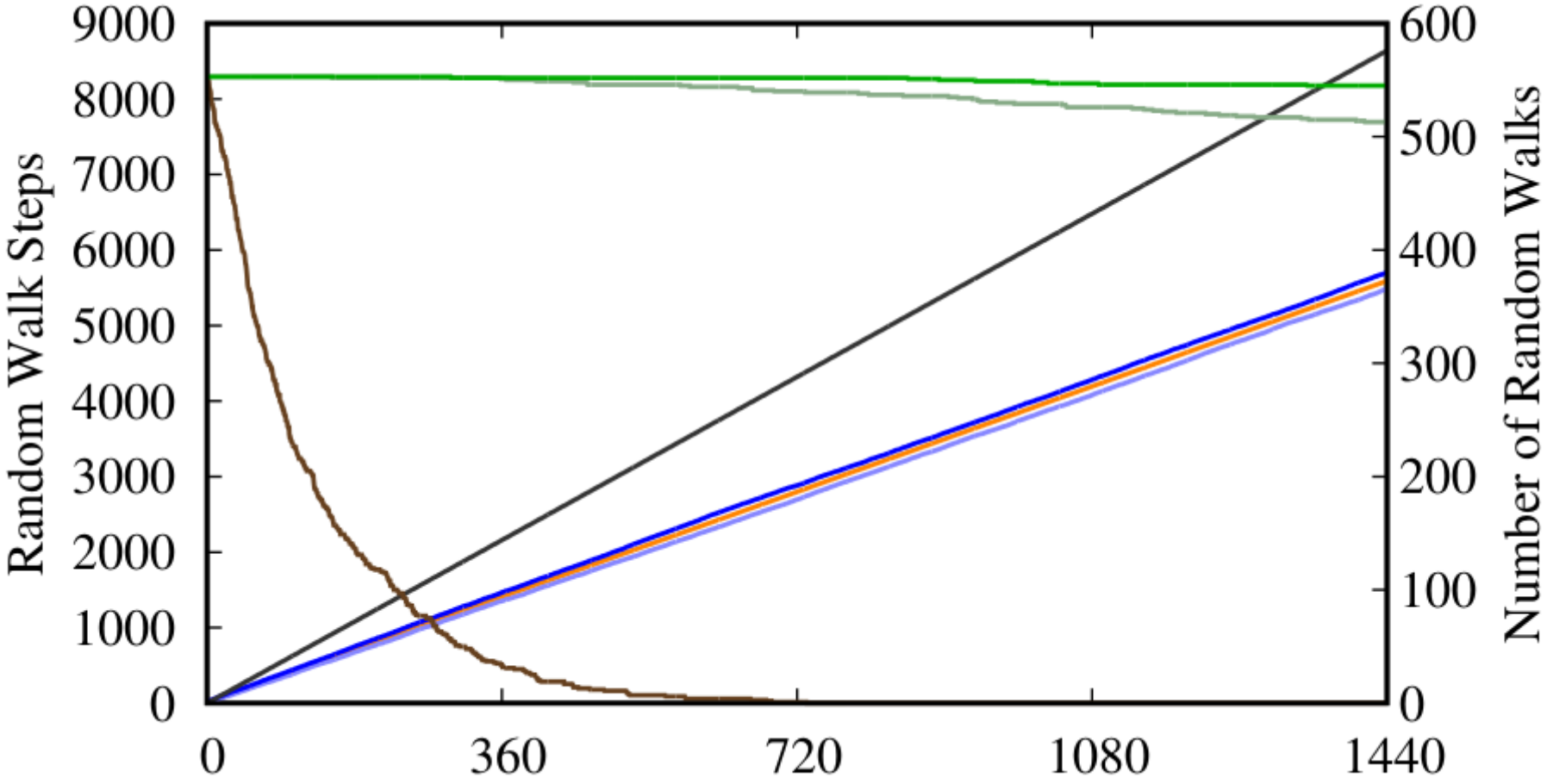
Results

Table II

Scenarios			Multiple Random Walk Protocol				without level 2 restarts		without restarts
network size	transmission time (s)	# random walks	# restarts at level 1	# restarts at level 2	lost random walks	max. # events broadcast	# restarts at level 1	lost random walks	lost random walks
10^3	1	40	672	0	0.00% (0)	0	672	0.00% (0)	100%
		553	9888	4	0.72% (4)	1	9922	1.44% (8)	100%
		5440	93620	253	0.62% (34)	3	90818	5.88% (320)	100%
	rand(1,10)	40	661	3	2.50% (1)	1	643	7.50% (3)	100%
		553	9512	20	1.98% (11)	1	9465	6.69% (37)	100%
		5440	75650	1083	0.58% (32)	7	67965	16.04% (873)	100%
	10	40	654	1	0.00% (0)	1	705	7.50% (3)	100%
		553	9011	46	1.44% (8)	1	8862	7.23% (40)	100%
		5440	79488	2055	0.75% (41)	7	65236	27.37% (1489)	100%
10^5	1	54383	923938	379	0.82% (449)	3	922568	1.43% (780)	100%
	rand(1,10)	54383	907904	2269	0.71% (391)	4	889612	4.85% (2642)	100%
	10	54383	887857	4183	0.74% (407)	5	858649	7.83% (4262)	100%

cial Intelligence

Results



Conclusions

- majority of restarts provided by a decentralized, scaleable supervison process (level 1)
- only small fraction of the problems get escaleted to level 2, rare broadcasting
- only extremely few centralized processes needed