

CRGC:



Dan Plyukhin
SDU

Fault-Recovering Actor GC in Apache Pekko



Gul Agha
UIUC

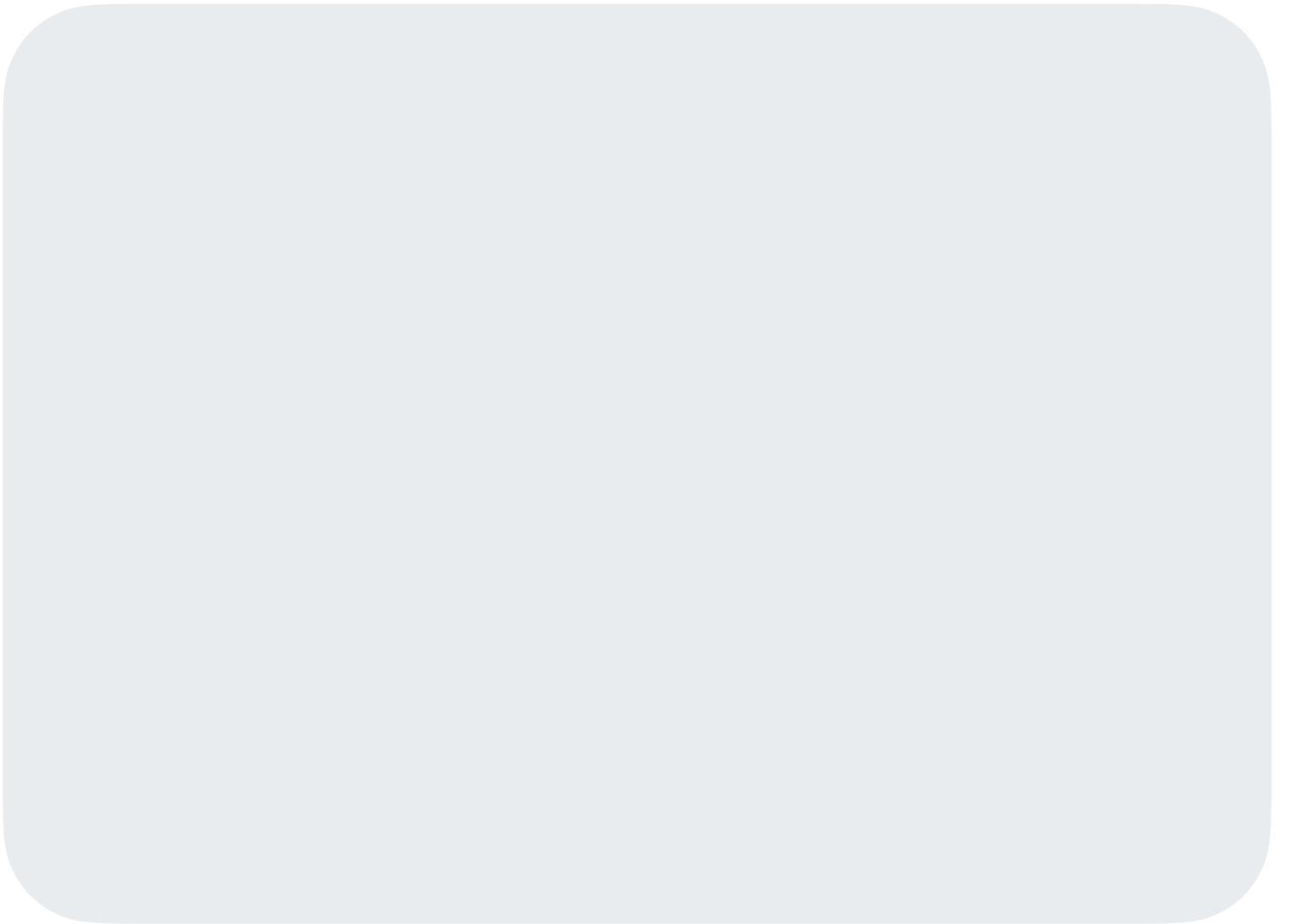


Fabrizio Montesi
SDU

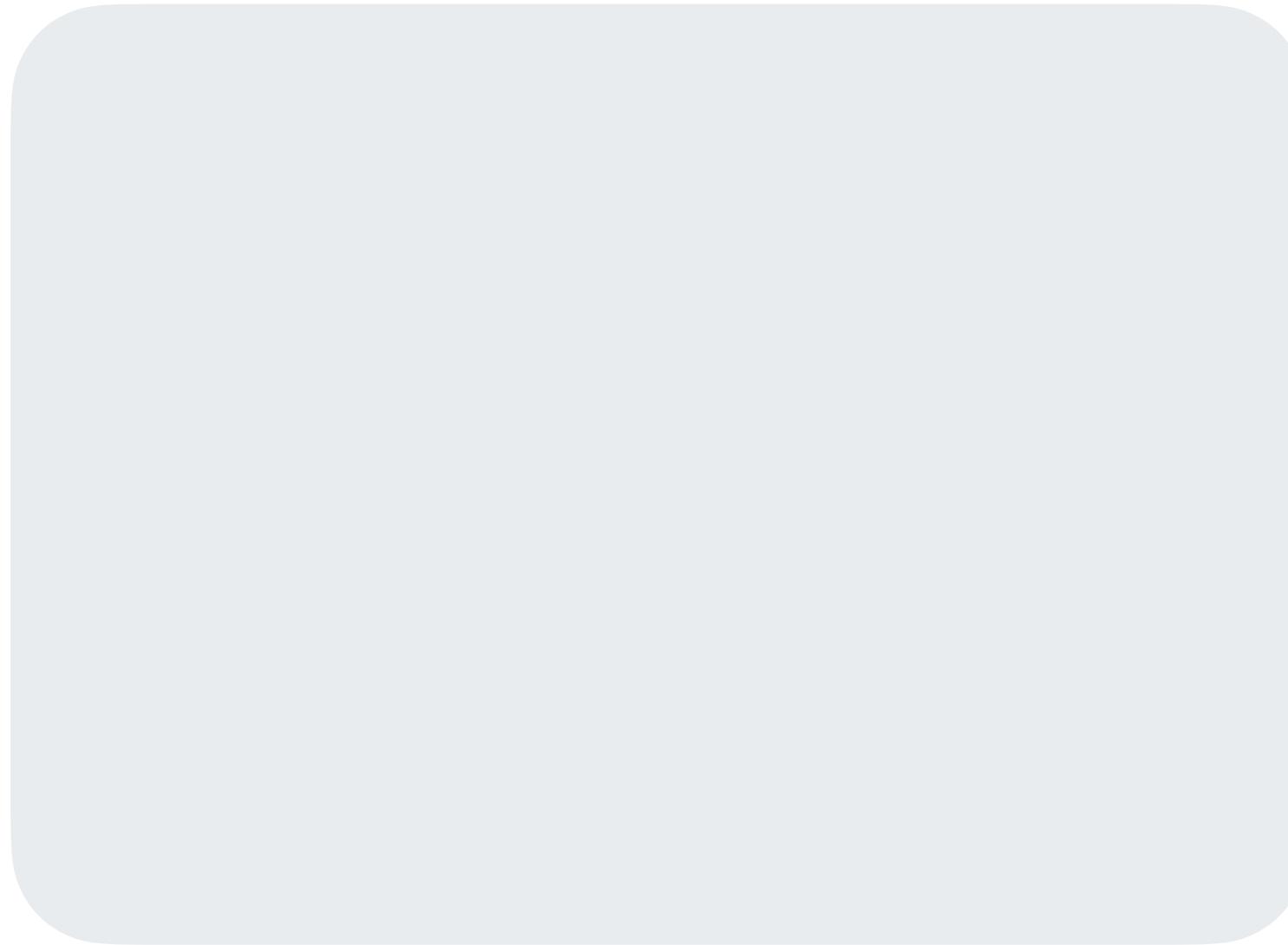
what are actors?

what are actors?

node 1



node 2

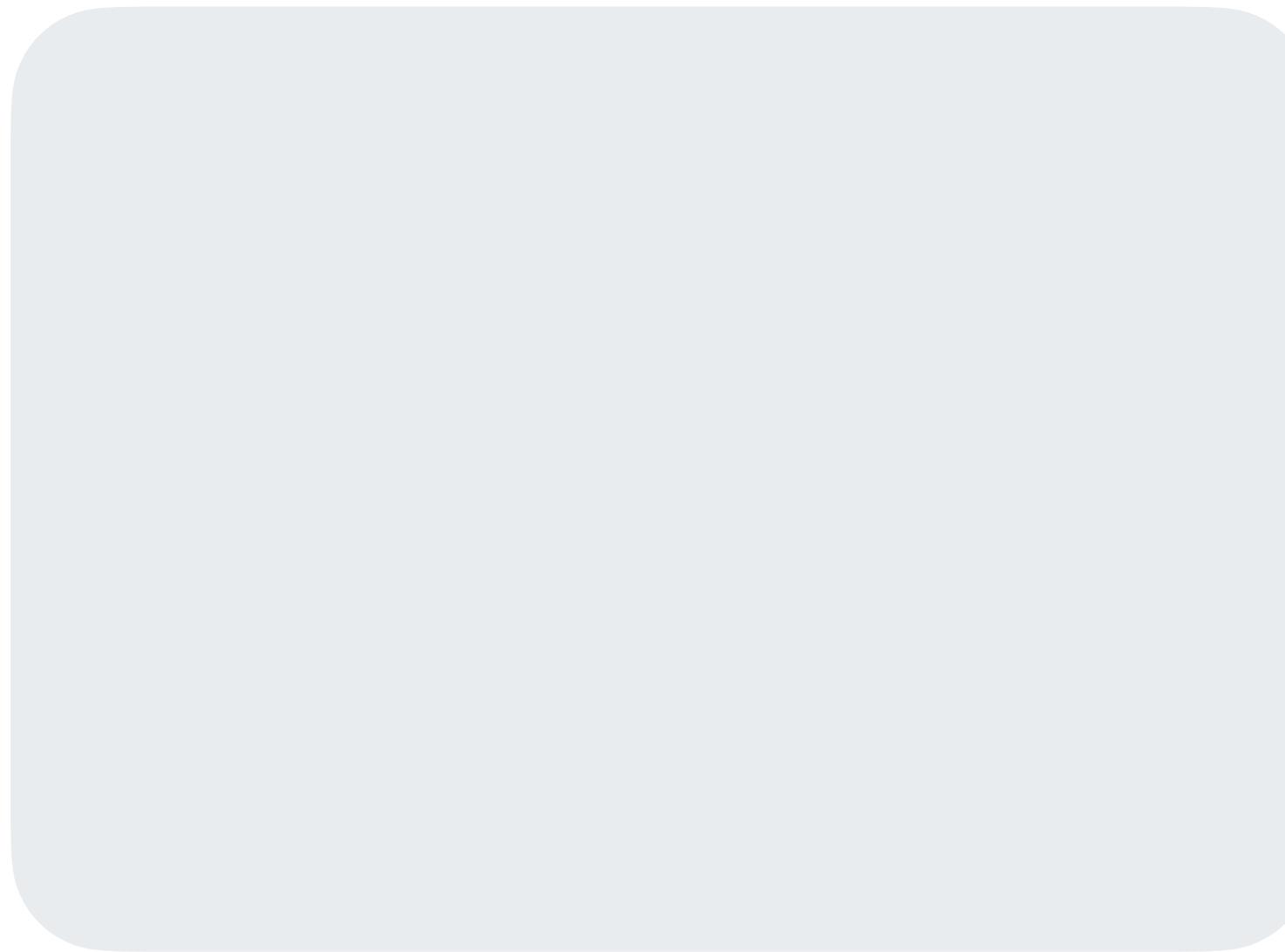


what are actors?

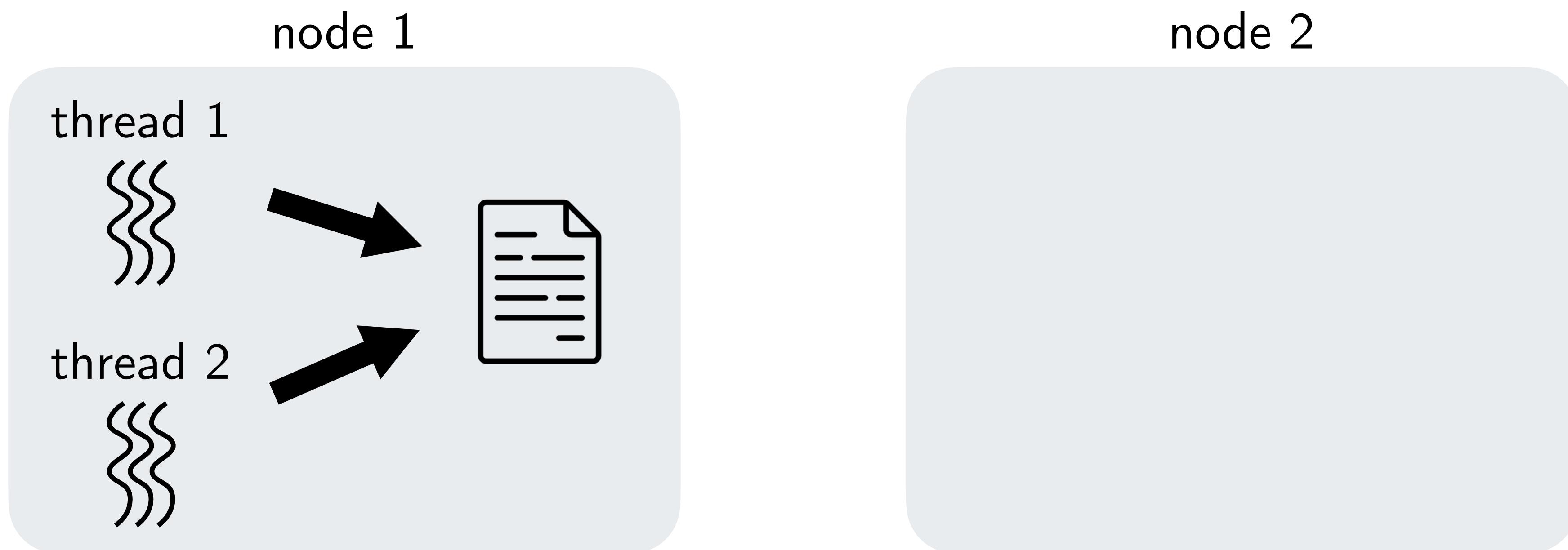
node 1



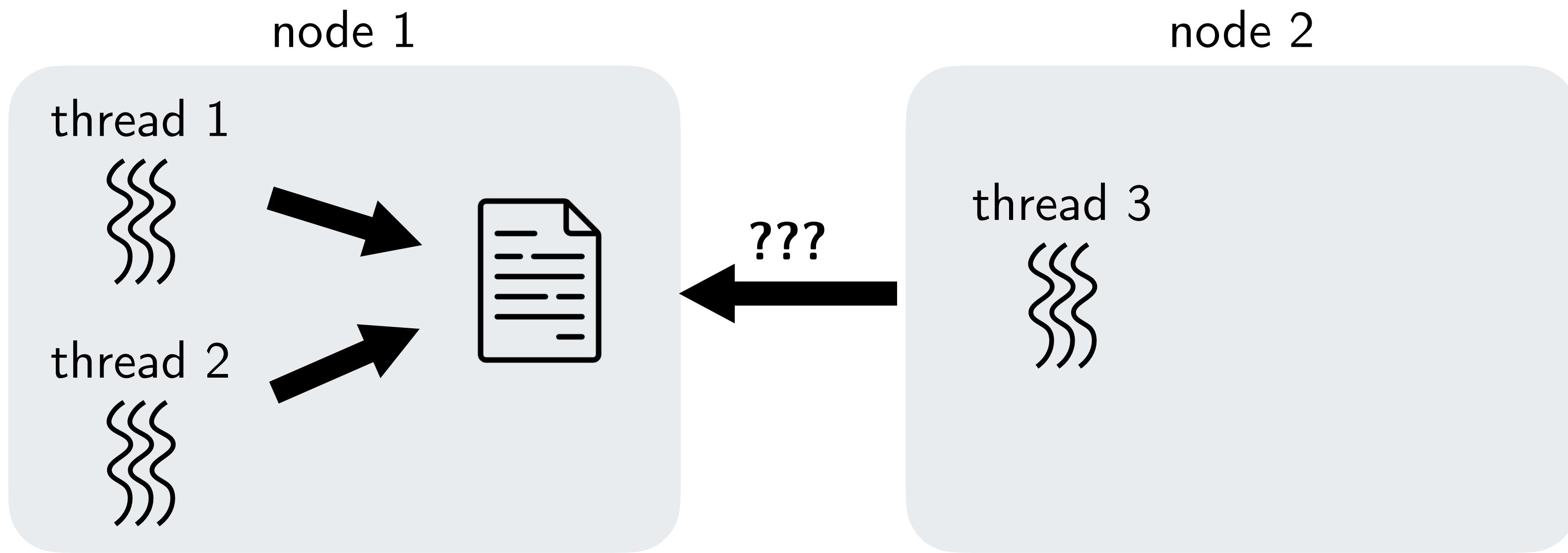
node 2



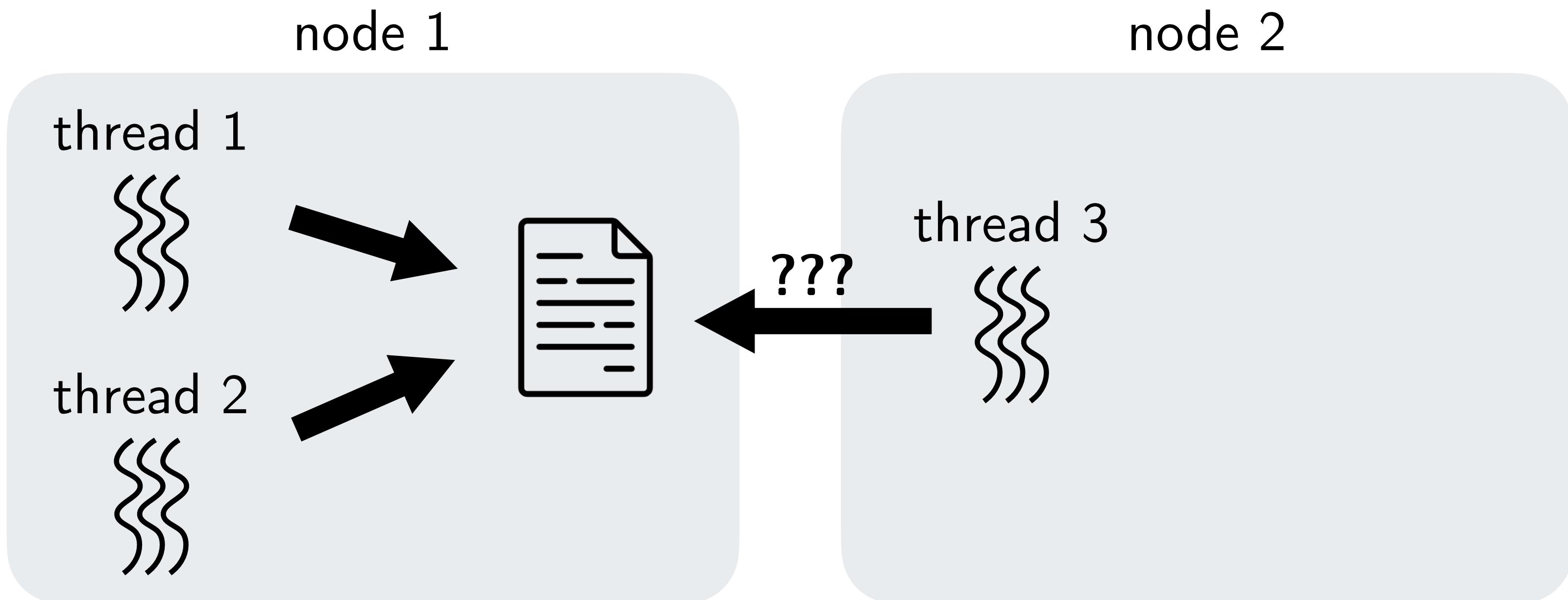
what are actors?



what are actors?

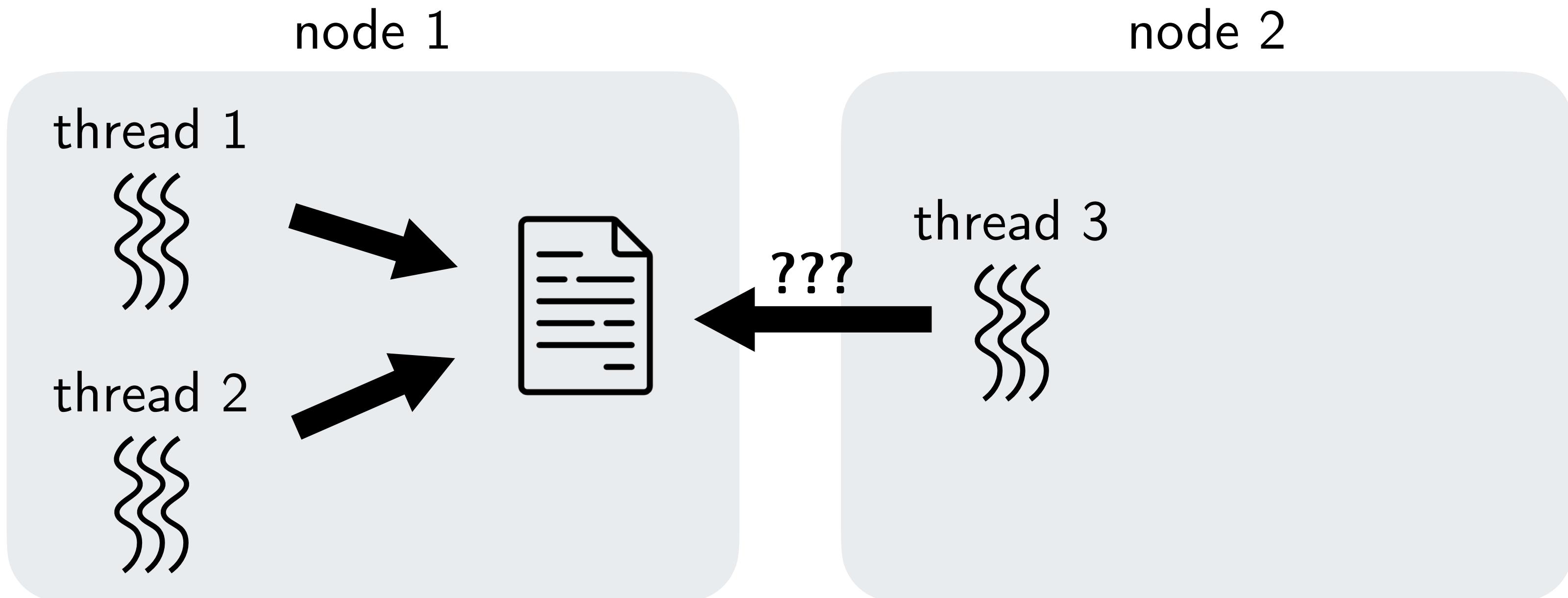


what are actors?



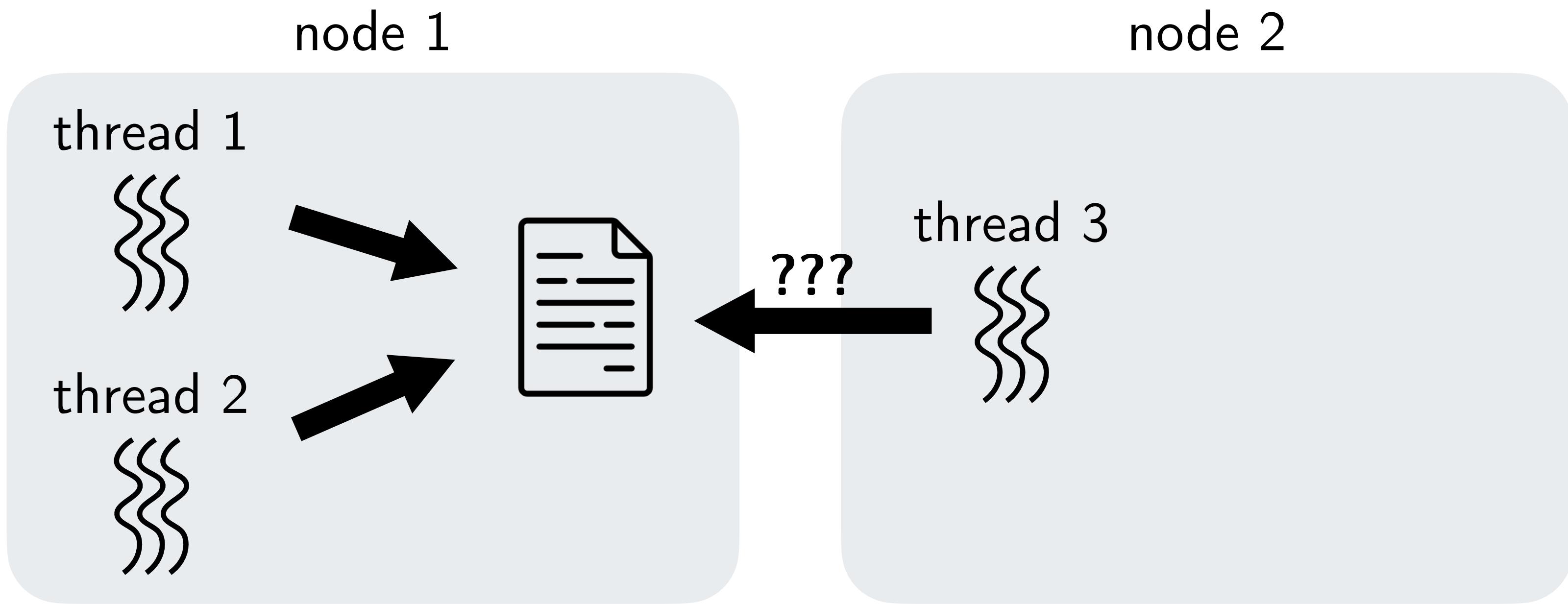
**concurrency
control?**

what are actors?



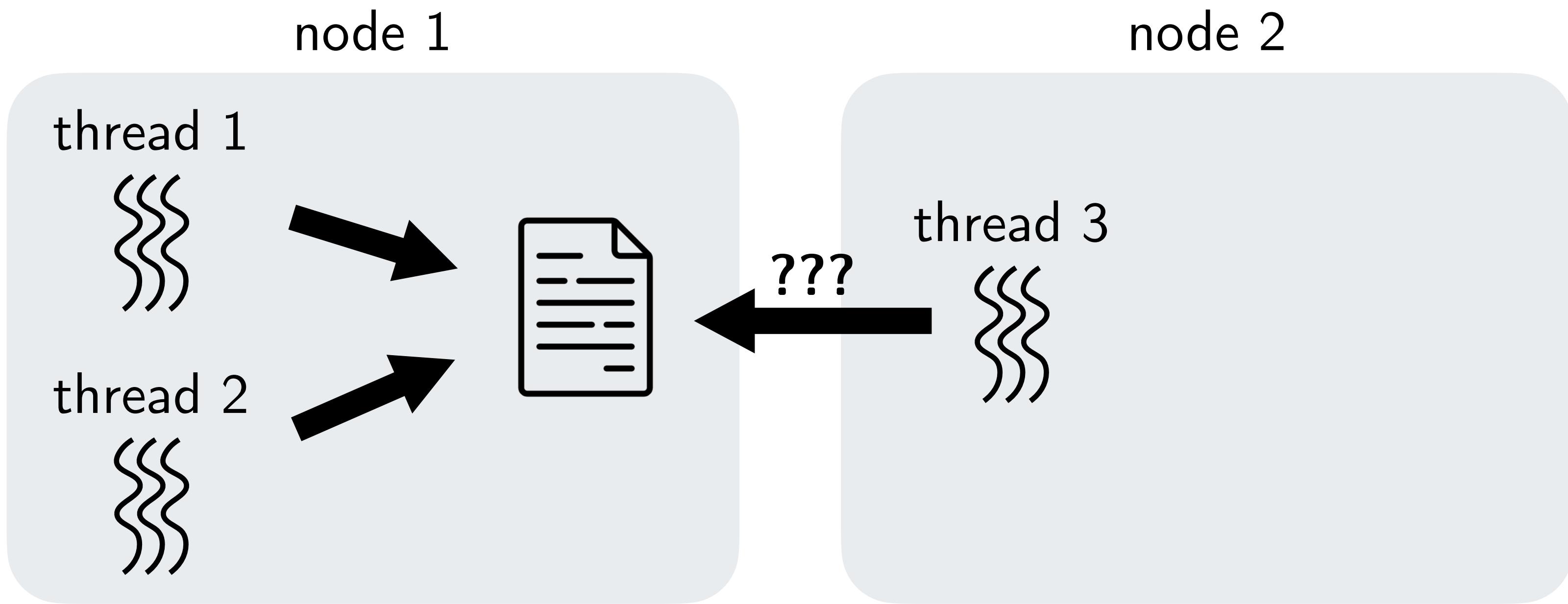
**concurrency
control?**
distribution?

what are actors?



**concurrency
control?**
distribution?
cleanup?

what are actors?

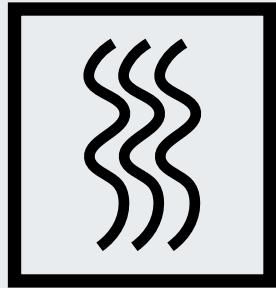


actors are
lightweight
processes

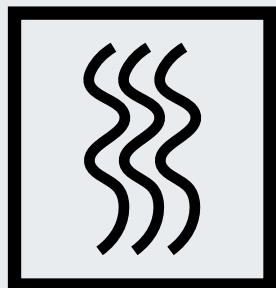
what are actors?

node 1

actor 1



actor 2

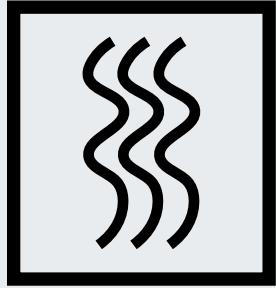


actor a



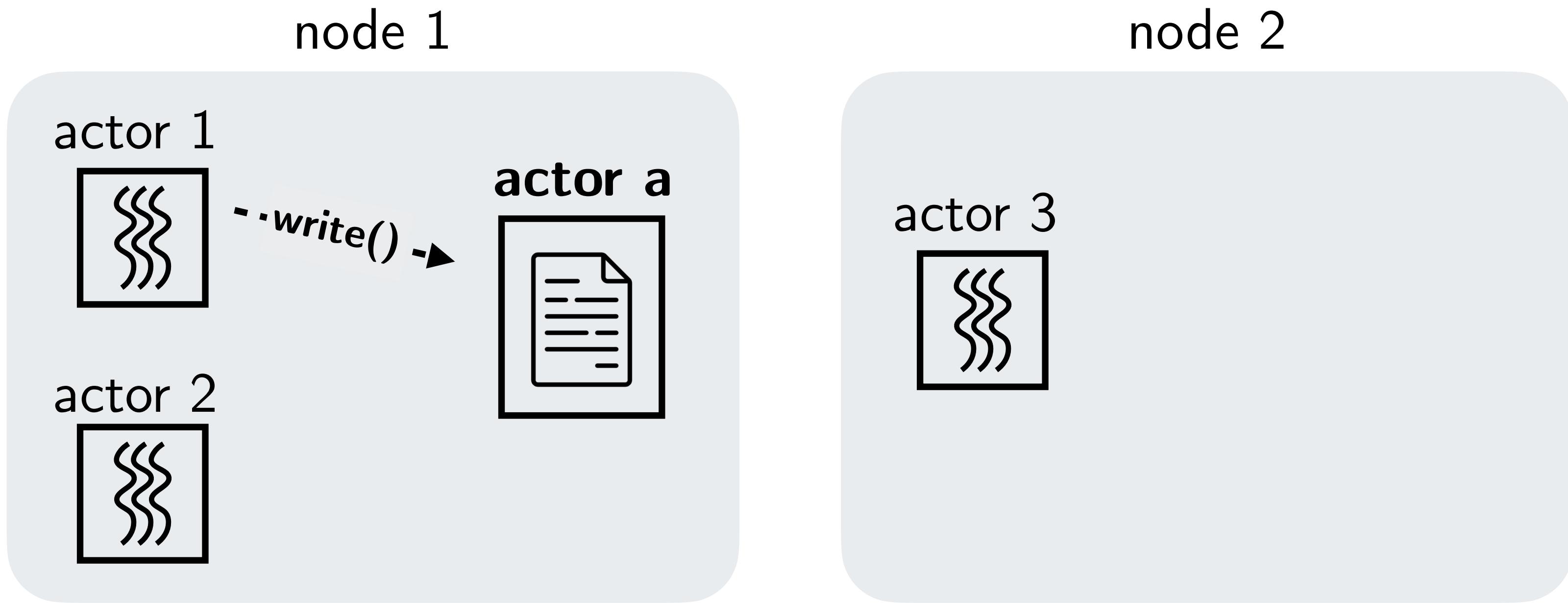
node 2

actor 3



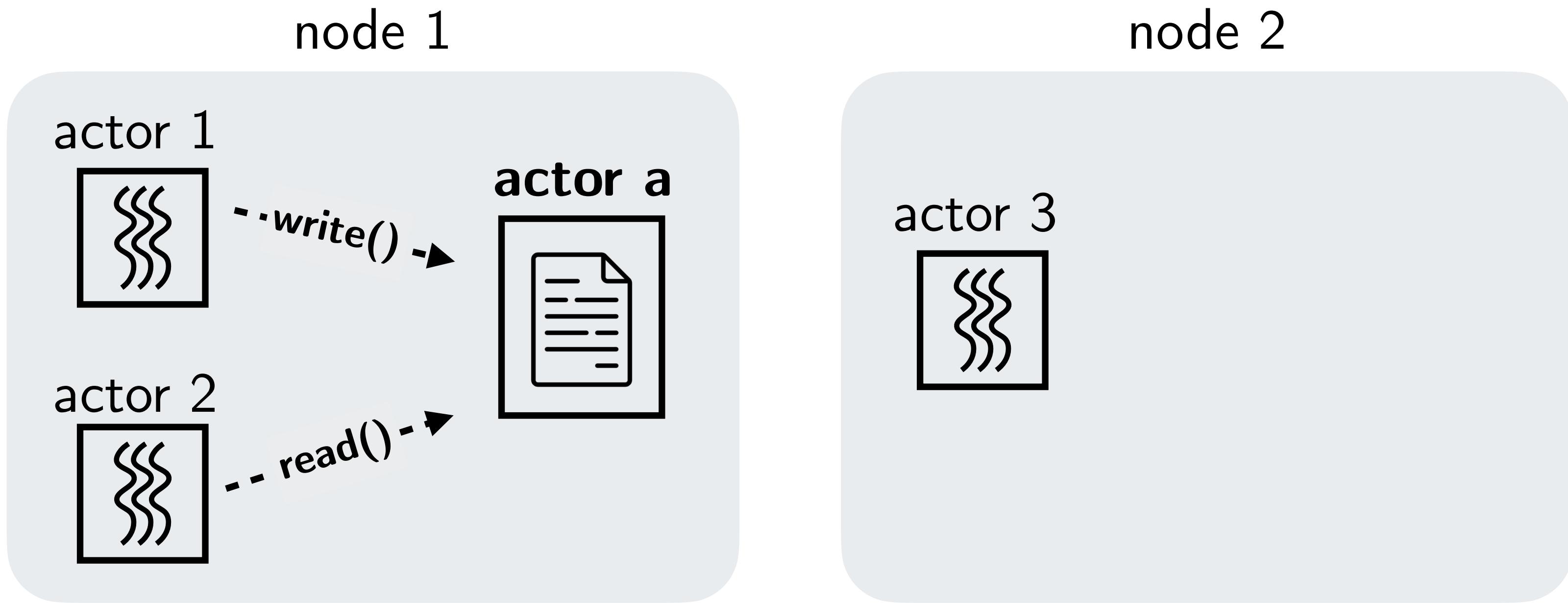
**actors are
lightweight
processes**

what are actors?



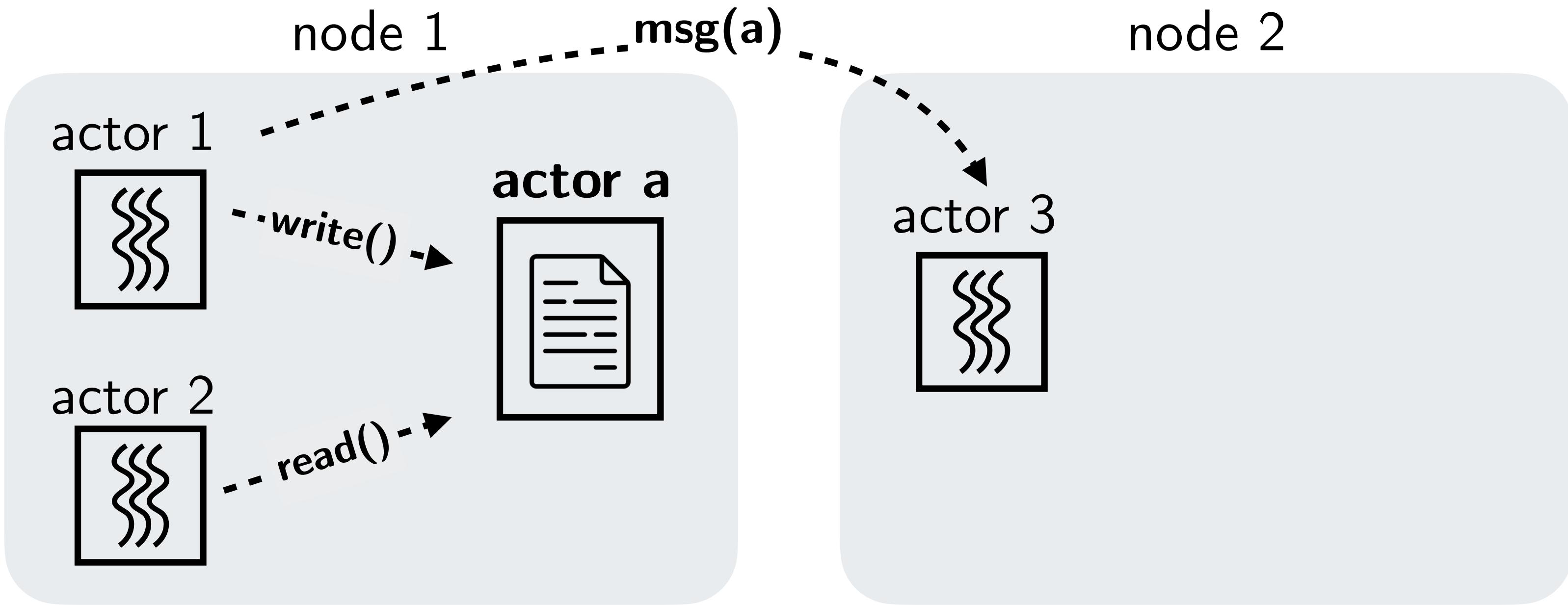
actors are
lightweight
processes

what are actors?



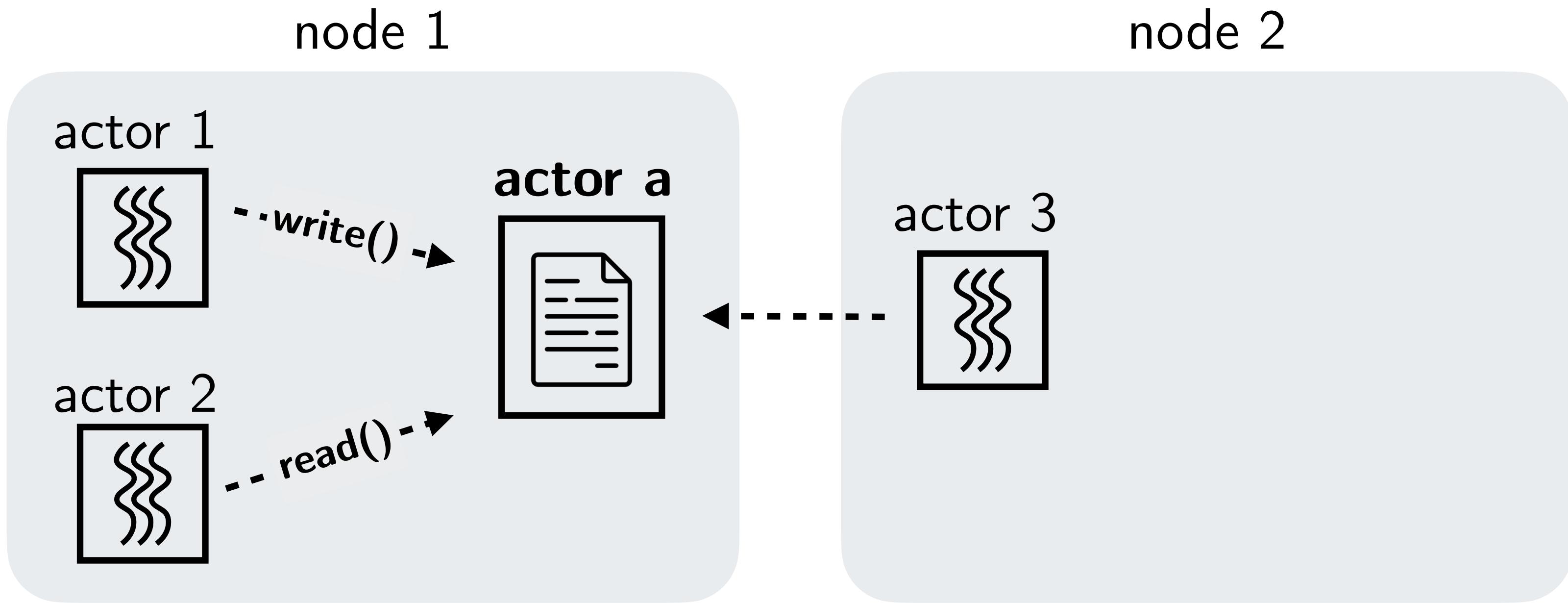
**actors are
lightweight
processes**

what are actors?



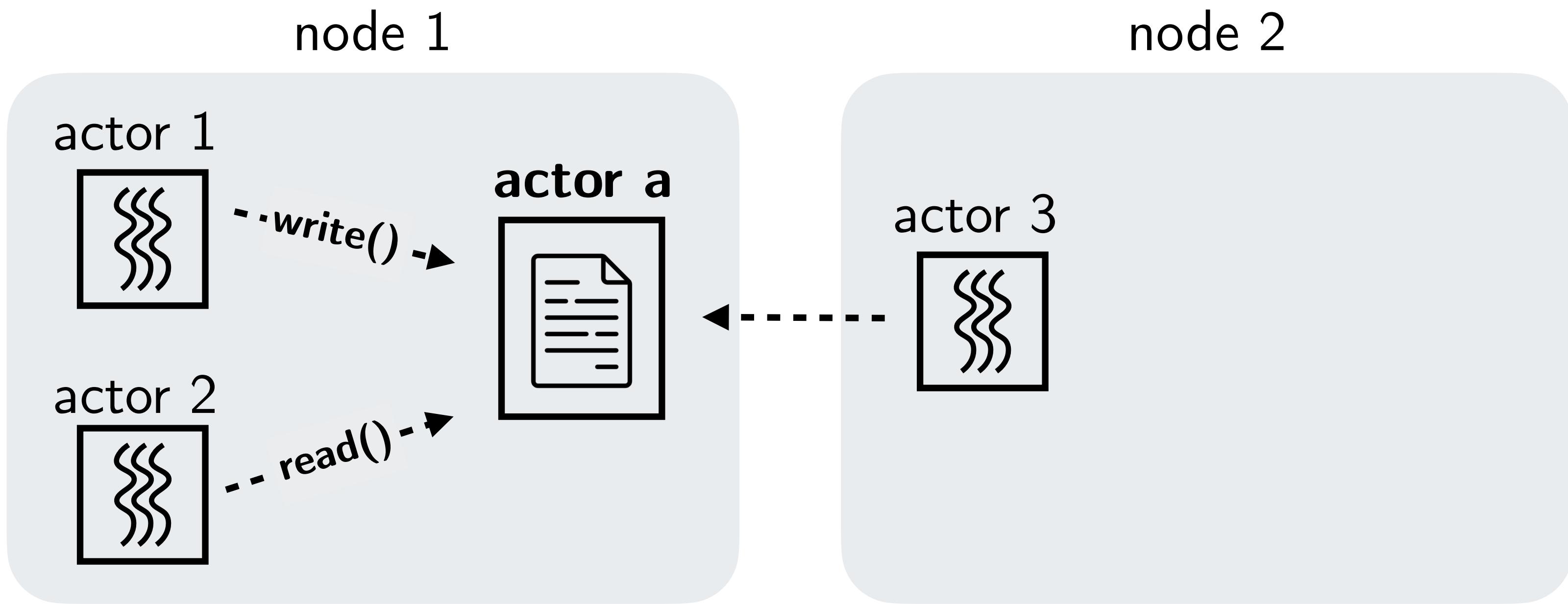
actors are
lightweight
processes

what are actors?



**actors are
lightweight
processes**

what are actors?



**messages are
handled
sequentially**

actor languages



actor languages



actor frameworks



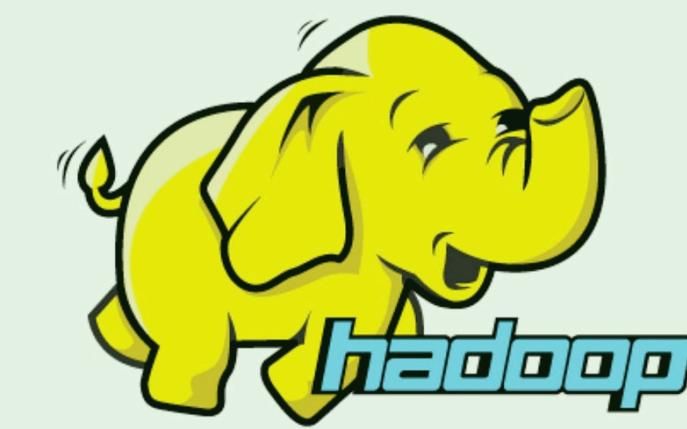
actor languages



actor frameworks



ad-hoc actors



actor languages

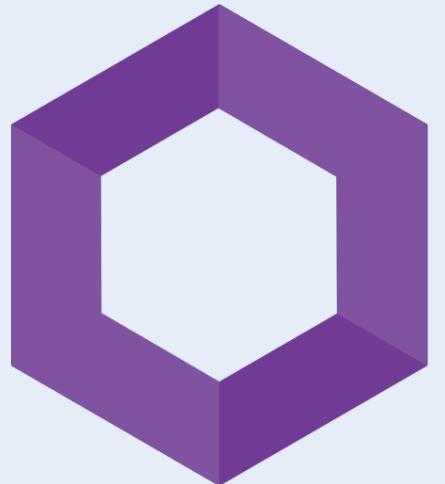


made with
actors

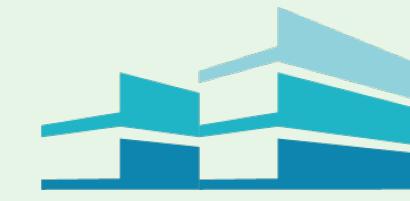
actor frameworks



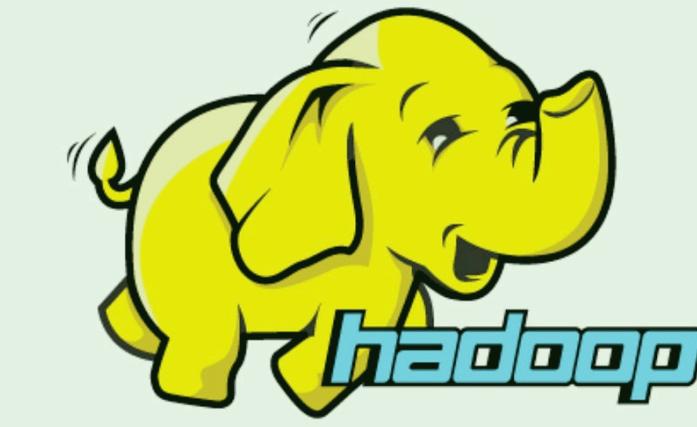
Pekko



RAY
Orleans



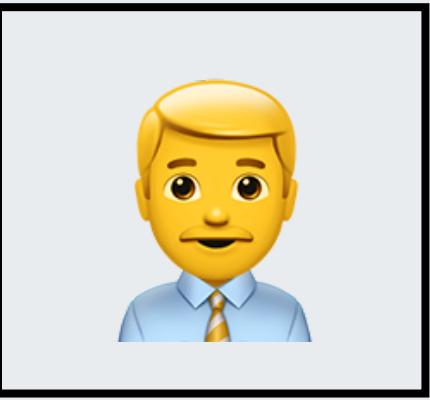
FOUNDATIONDB



the problem

node 1

manager



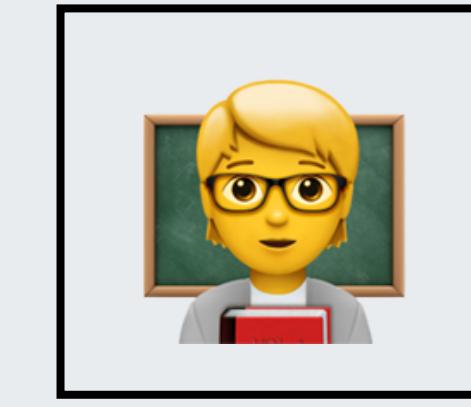
node 2

task

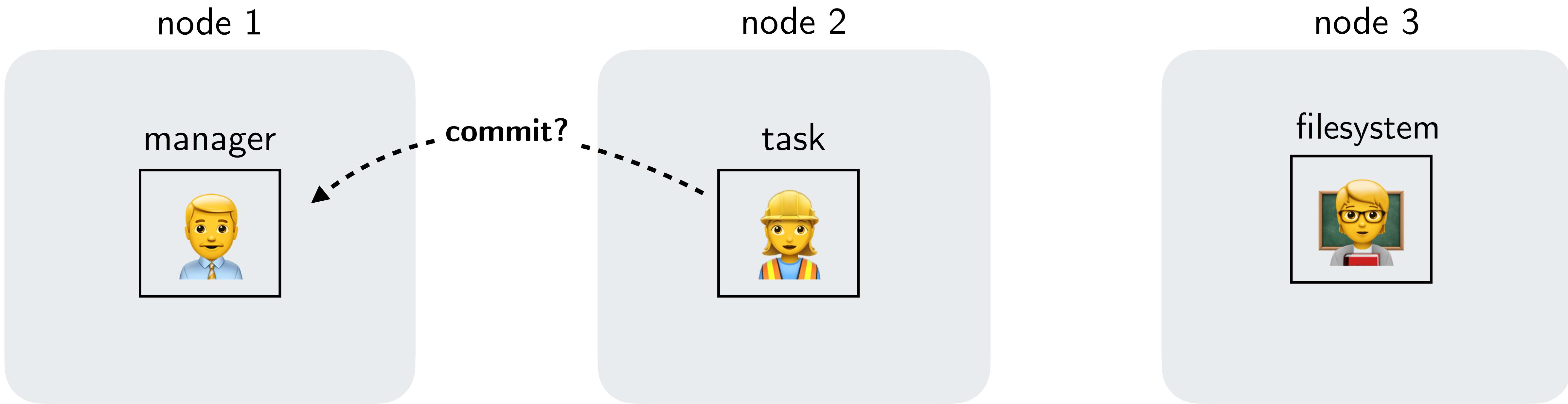


node 3

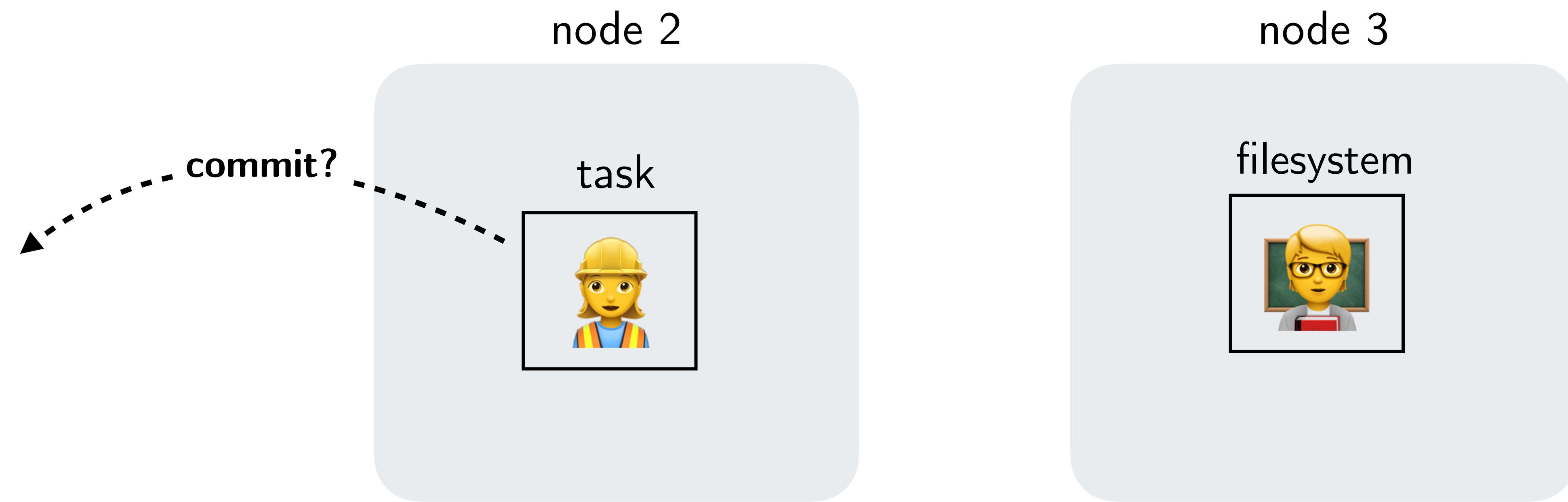
filesystem



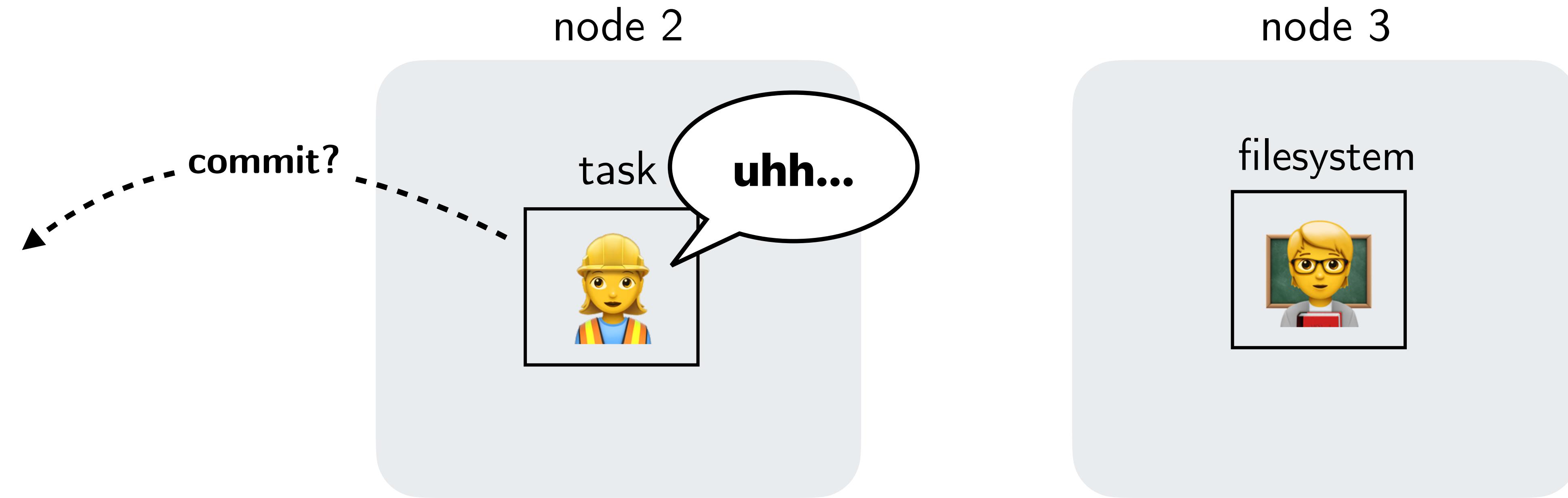
the problem



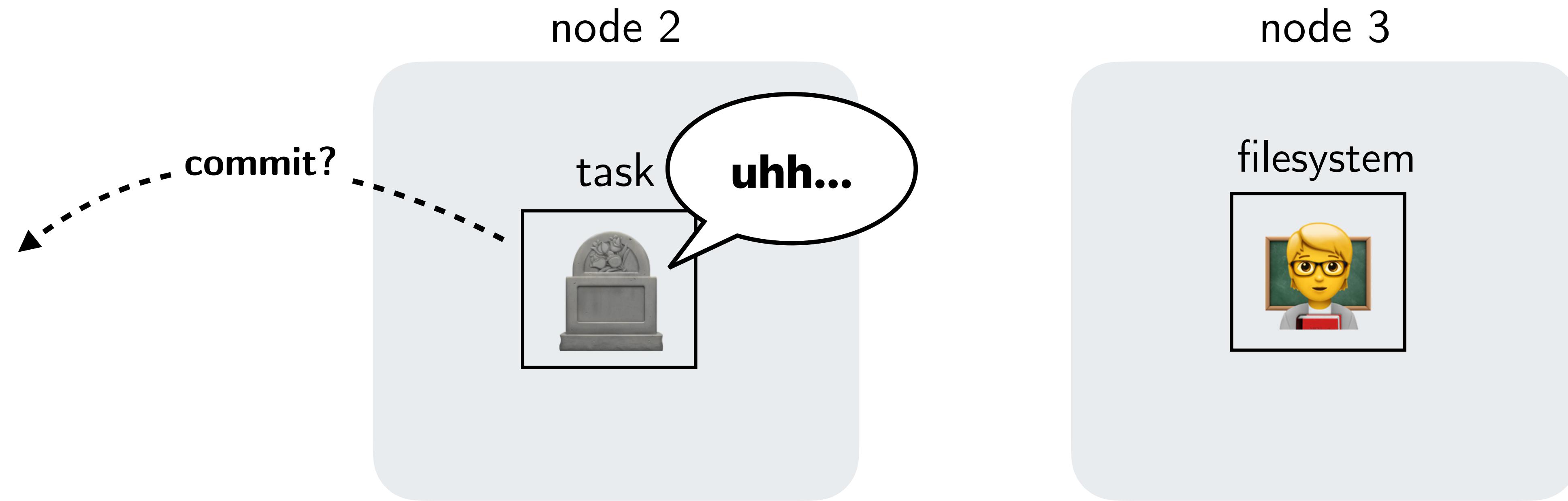
the problem



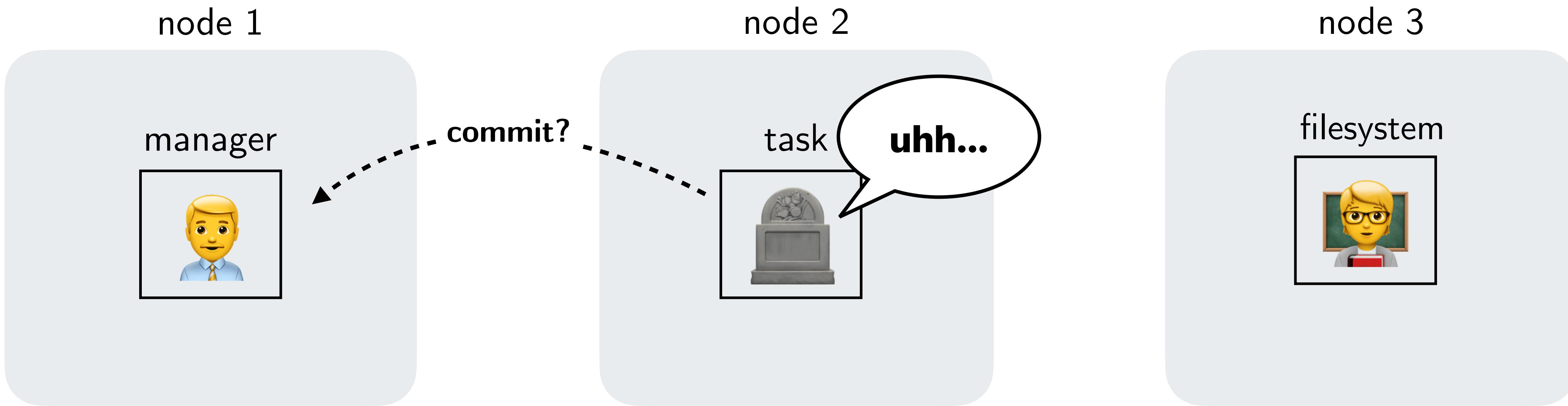
the problem



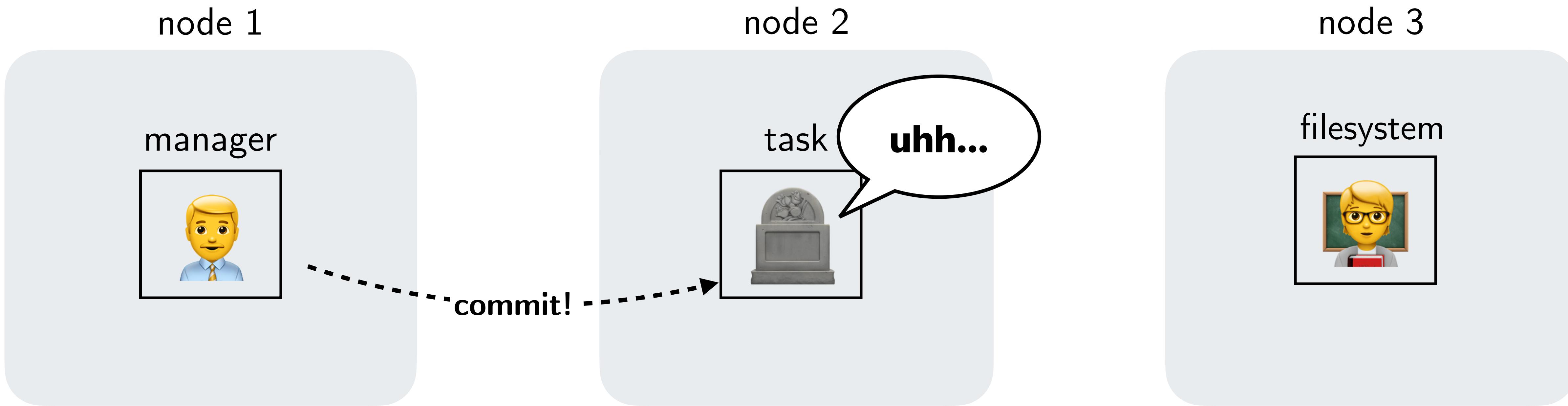
the problem



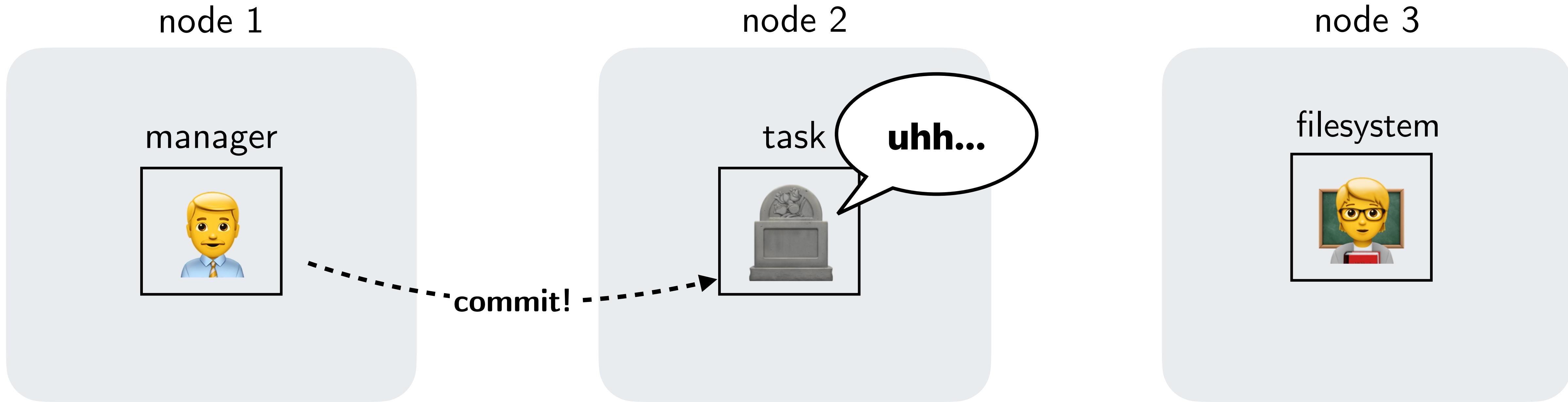
the problem



the problem



the problem



examples: see issues [#3006](#), [#4099](#), [#5009](#) in Hadoop MapReduce JIRA

the problem

the problem

clean up your actors!

the problem

clean up your actors!

the problem

clean up your actors!

...but not too early

the problem

clean up your actors!

...but not too early

the problem

clean up your actors!

...but not too early

...and not too late

the problem

clean up your actors!

...but not too early

...and not too late

the problem

clean up your actors!

...but not too early

...and not too late

...and predict all faults 

the problem

clean up your actors!

...but not too early

...and not too late

...and predict all faults 

our mission

the problem

clean up your actors!

...but not too early

...and not too late

...and predict all faults 

our mission

don't kill live actors

the problem

clean up your actors!

...but not too early

...and not too late

...and predict all faults 

our mission

don't kill live actors

kill all garbage actors

the problem

clean up your actors!

...but not too early

...and not too late

...and predict all faults



our mission

don't kill live actors

kill all garbage actors

prove it

the problem

clean up your actors!

...but not too early

...and not too late

...and predict all faults 

our mission

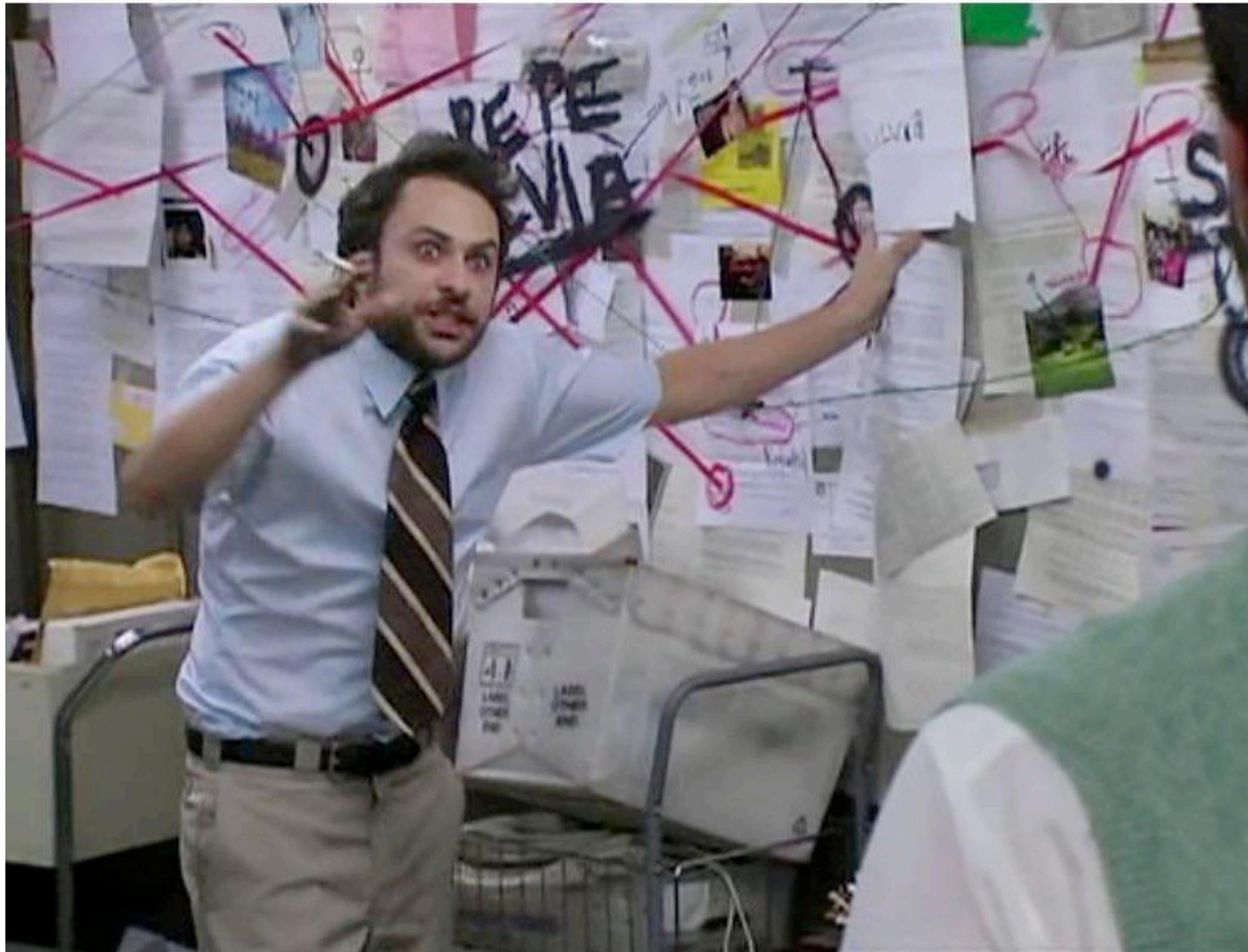
don't kill live actors

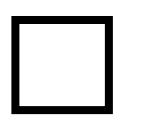
kill all garbage actors

prove it

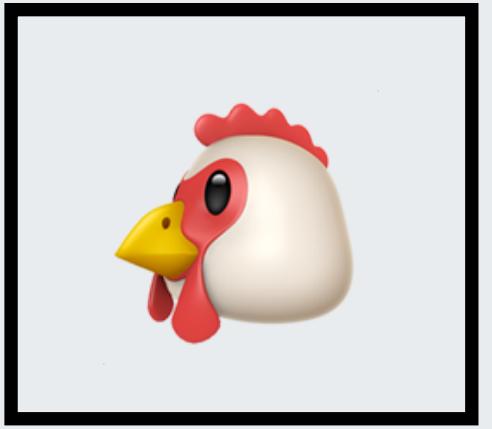
vroom vroom 

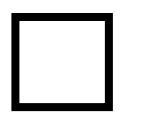
what is actor garbage?



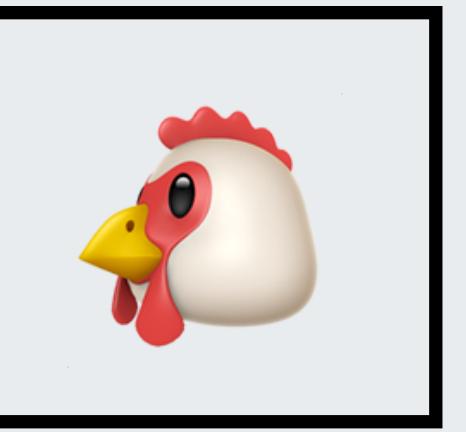


busy actor





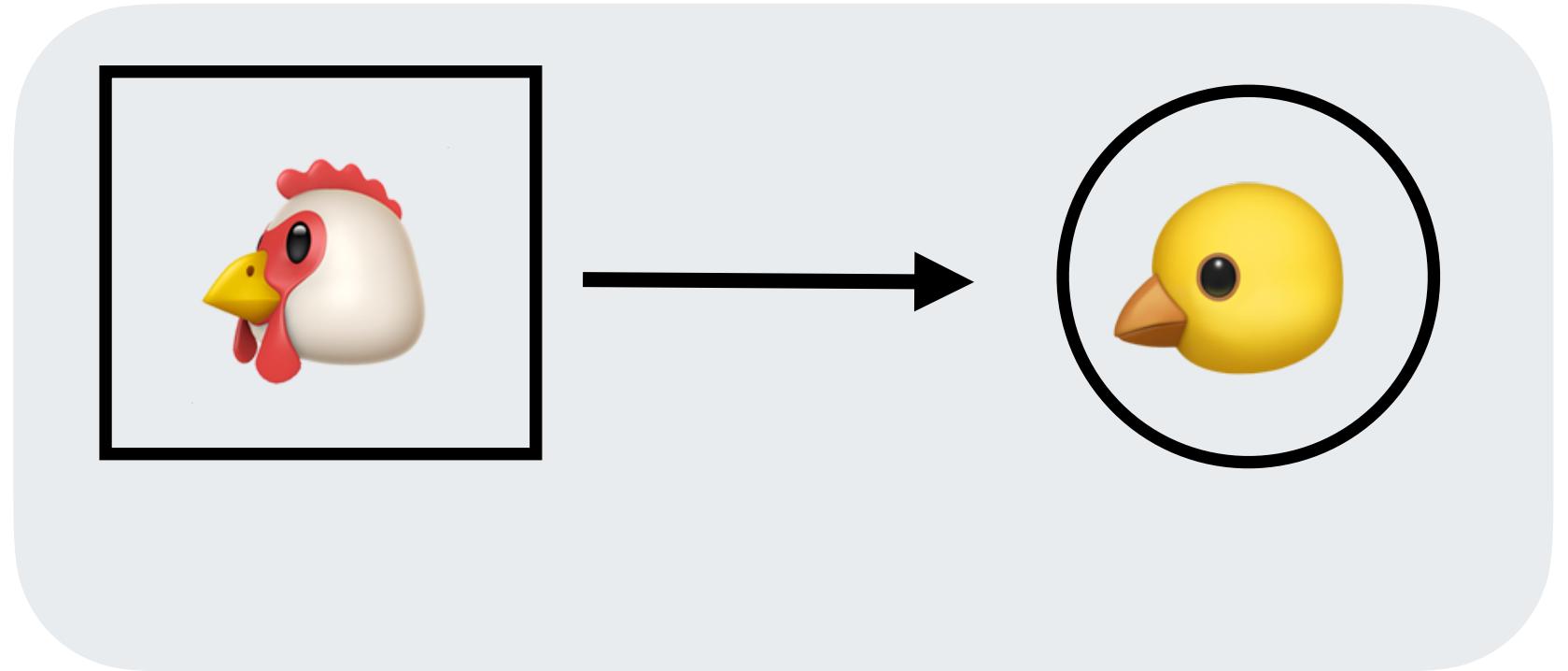
busy actor



actors can...

□ busy actor

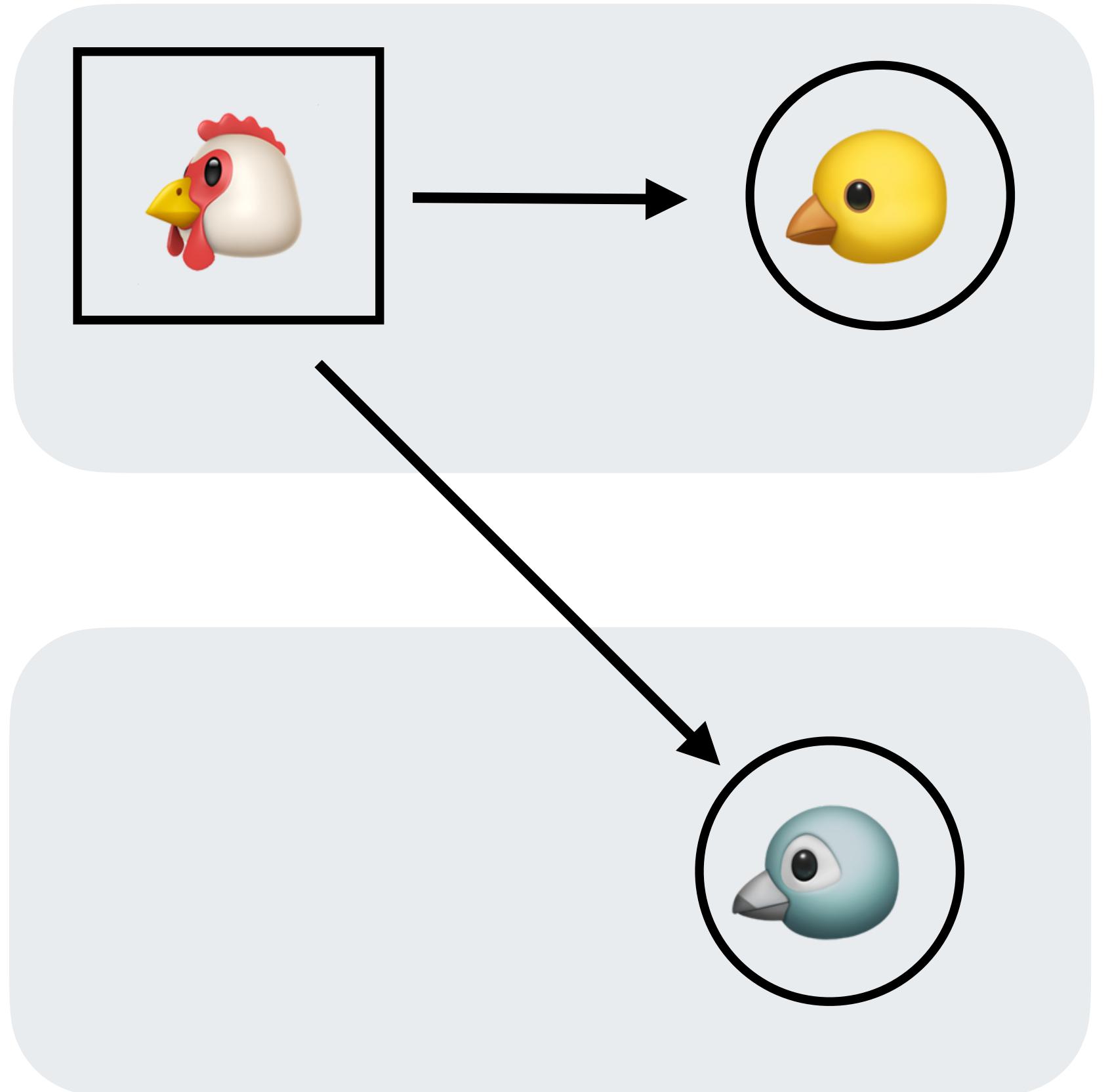
→ reference



actors can...
...spawn

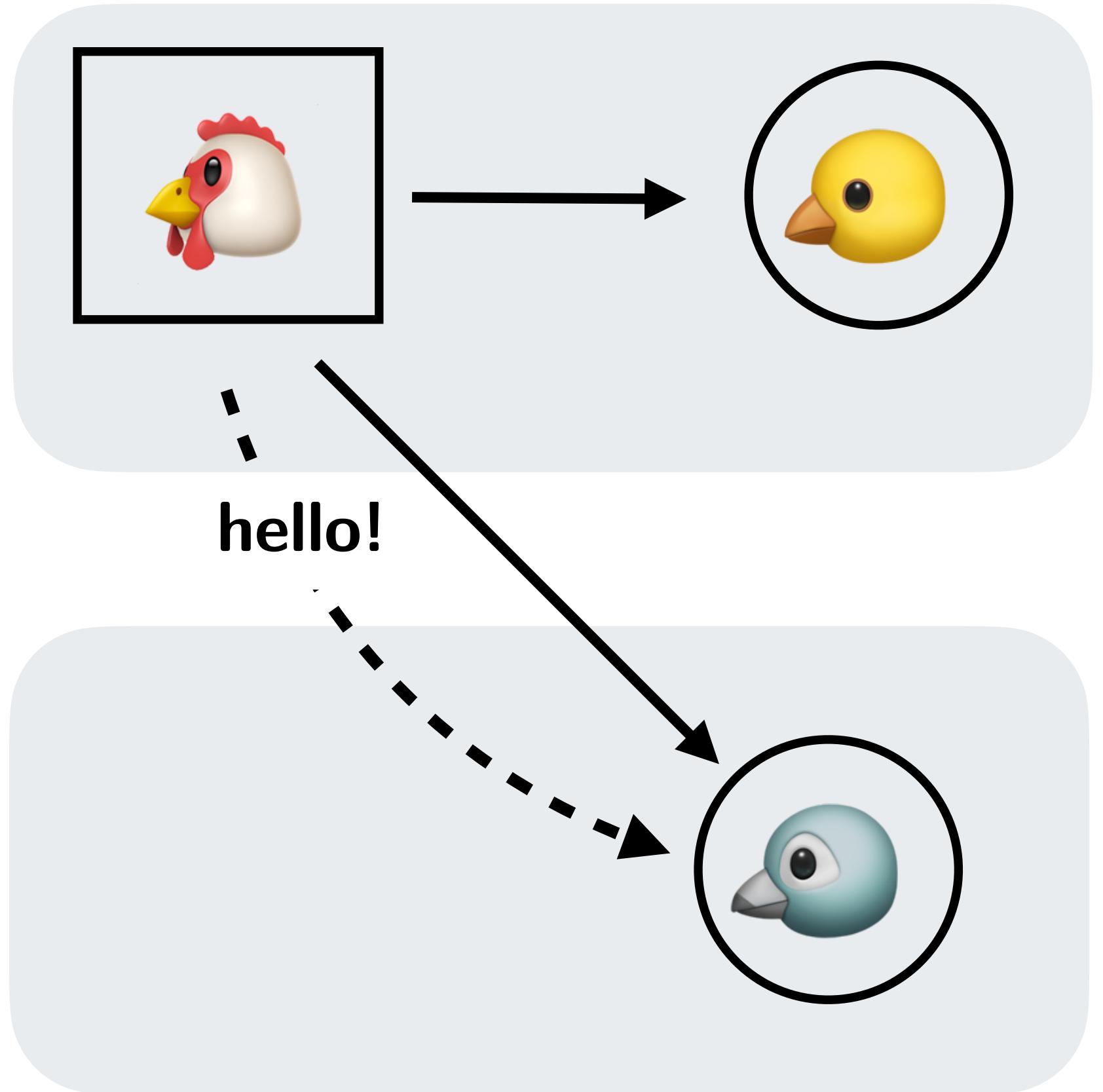
□ busy actor

→ reference



actors can...
...spawn

- busy actor
- reference
- > message

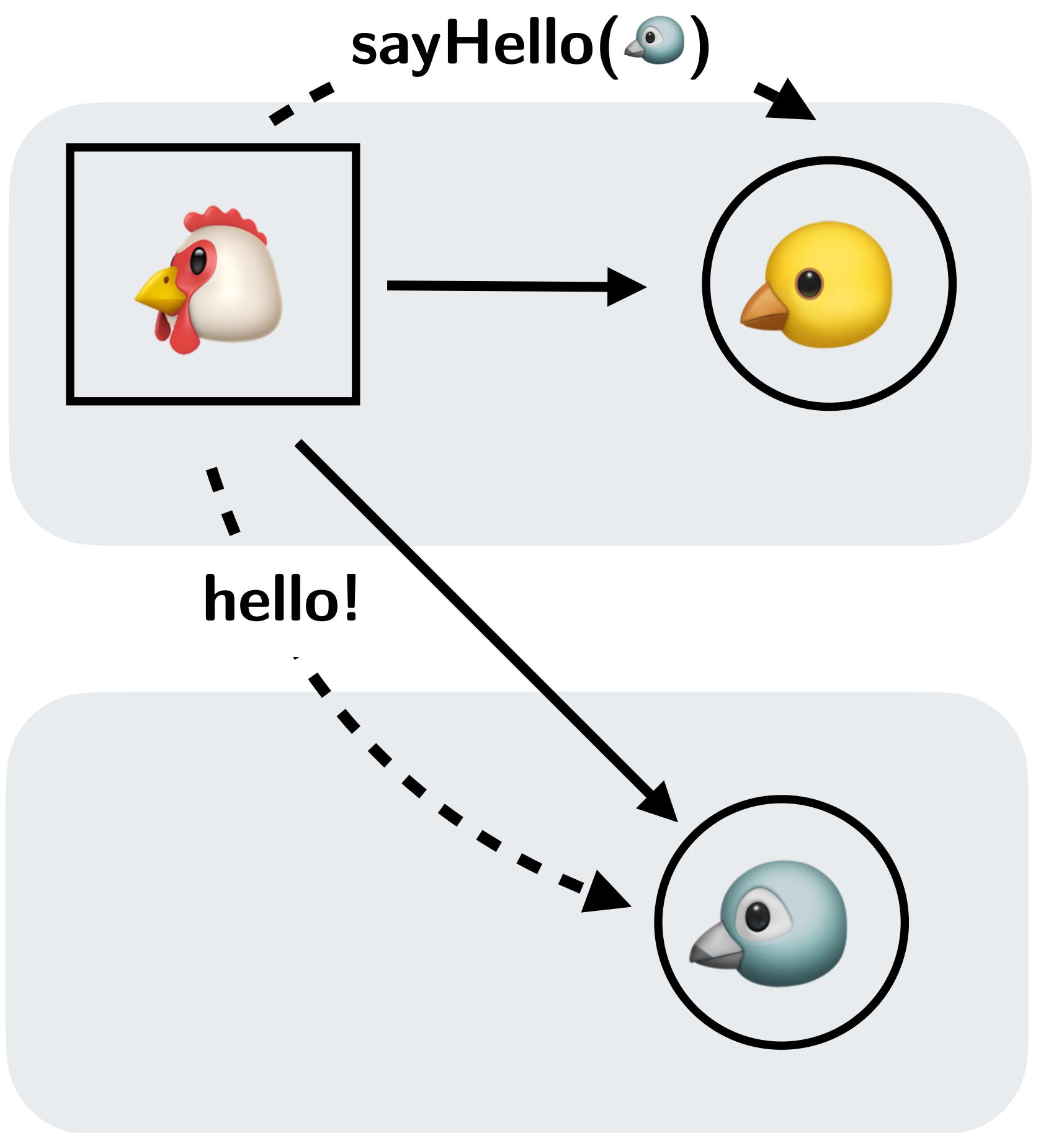


actors can...

...spawn

...send messages

- busy actor
- reference
- > message



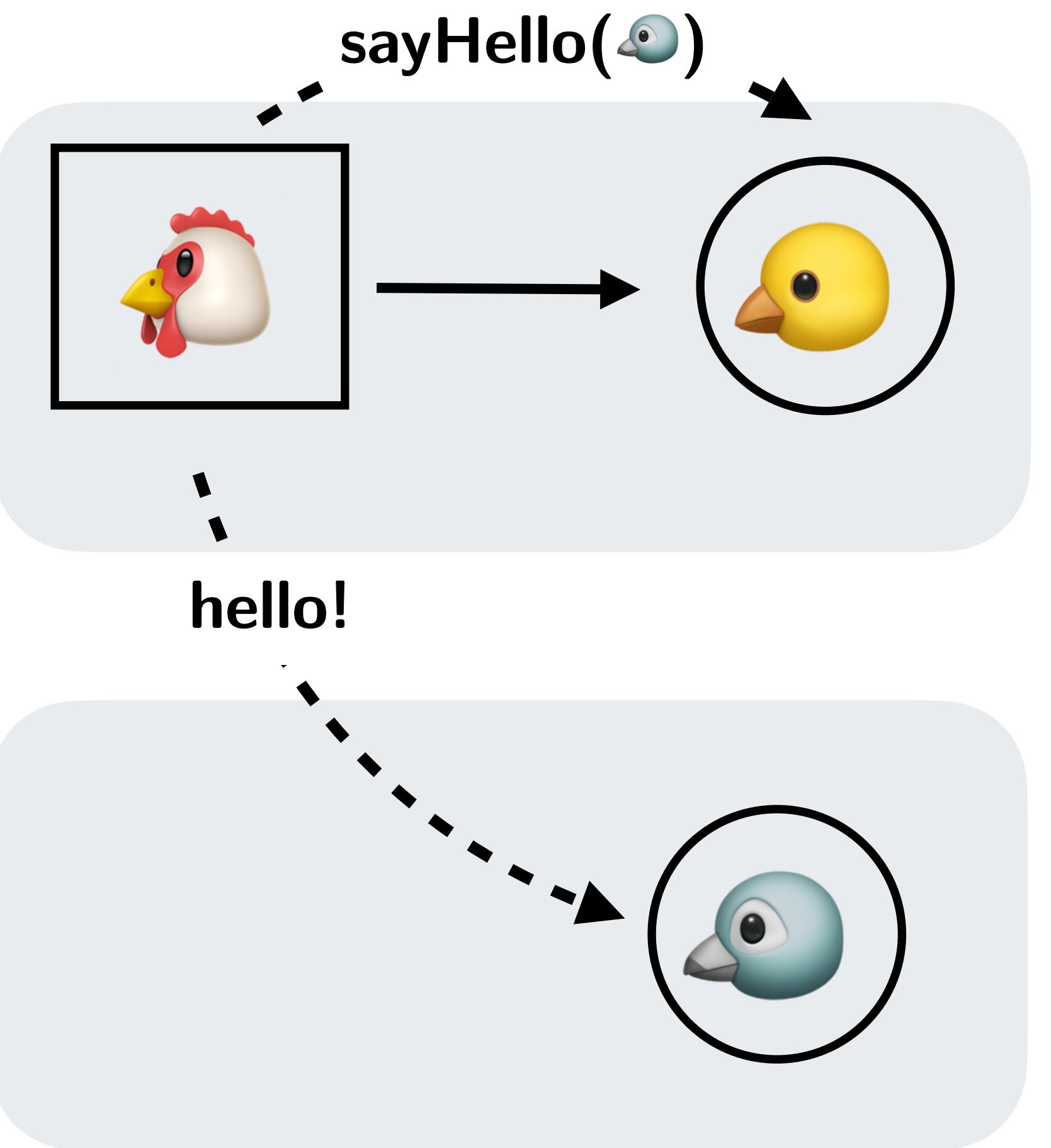
actors can...

...spawn

...send messages

...send references

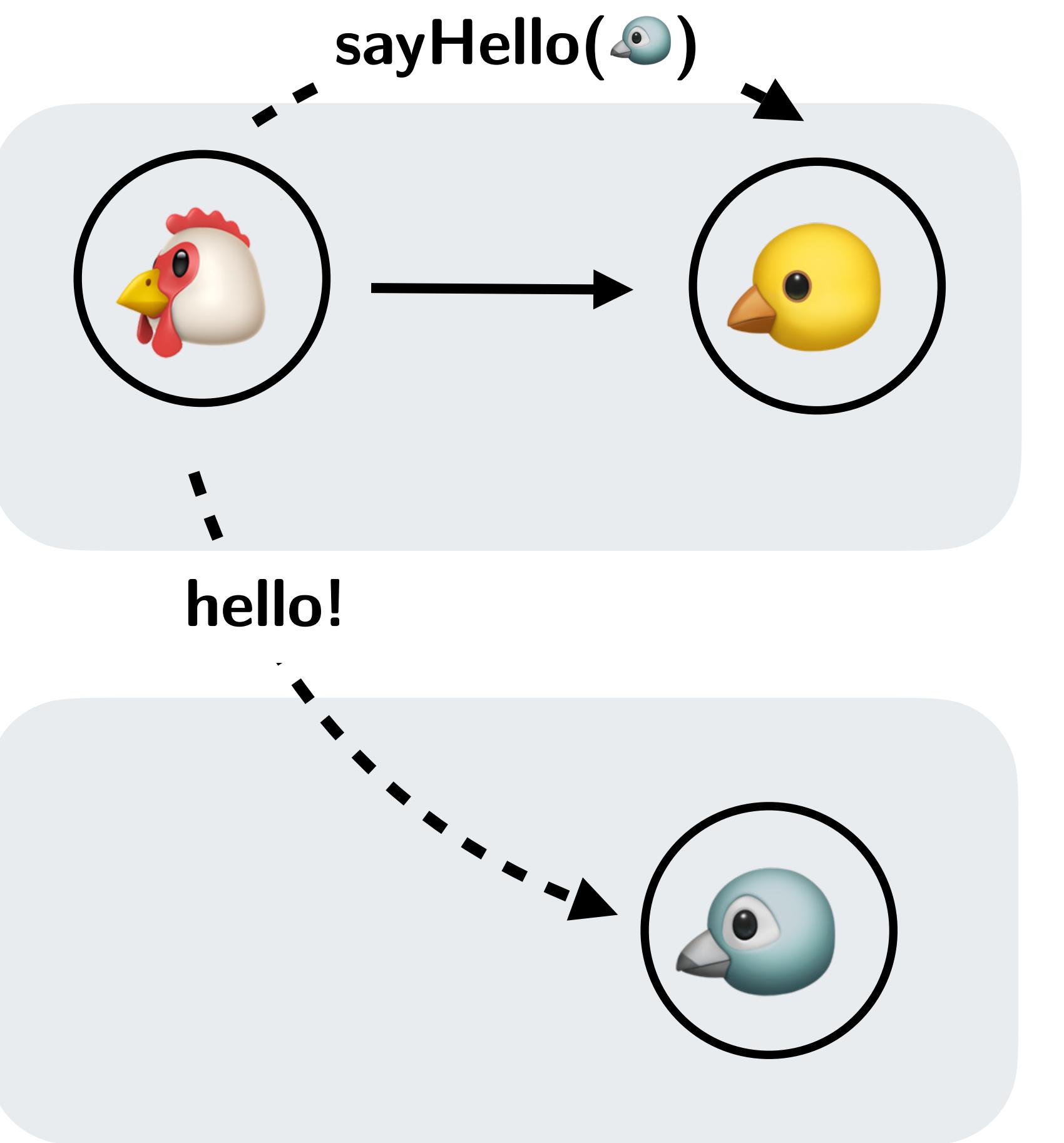
- busy actor
- reference
- > message



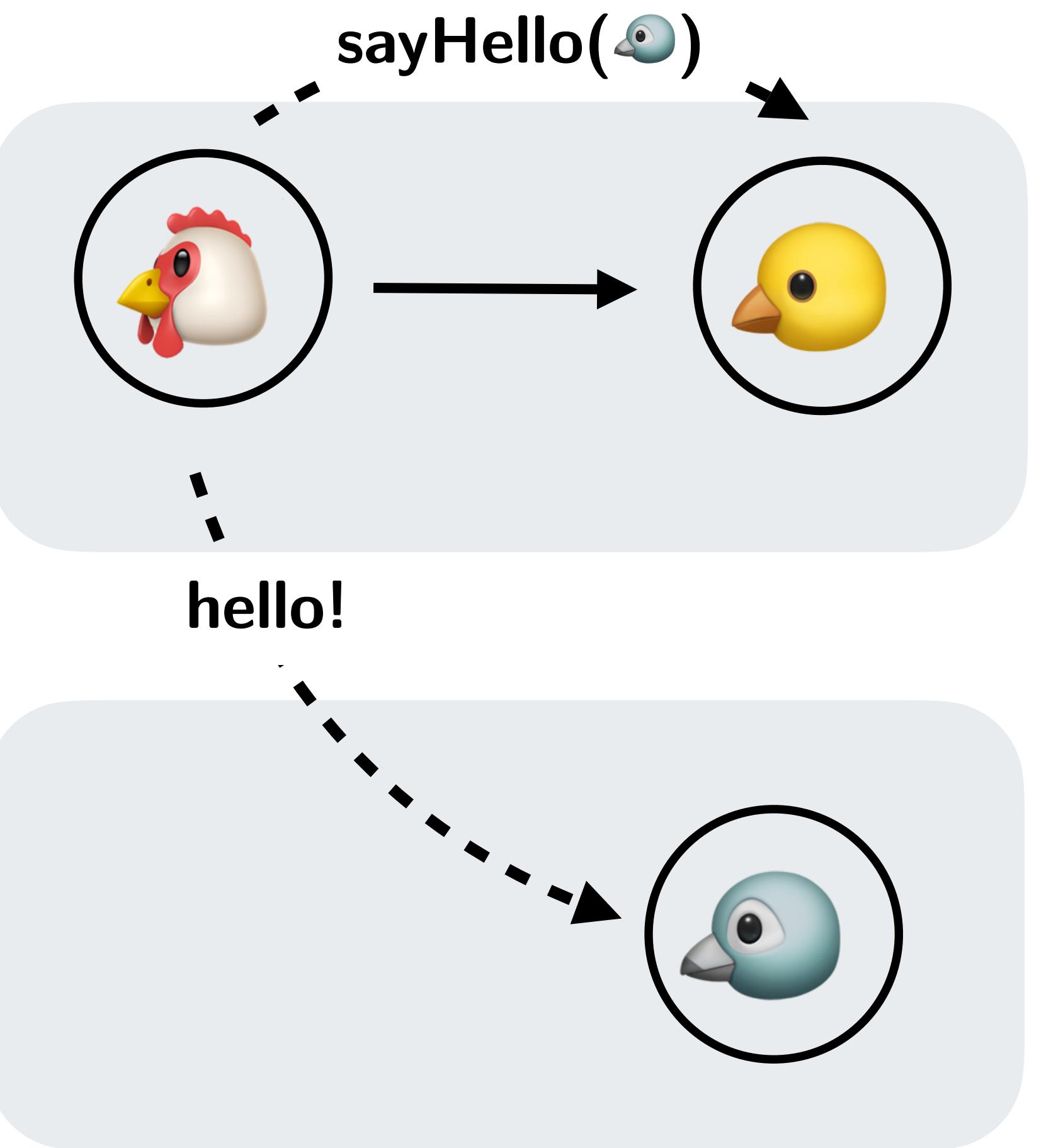
actors can...

- ...spawn
- ...send messages
- ...send references
- ...forget references

- busy actor
- idle actor
- reference
- -> message

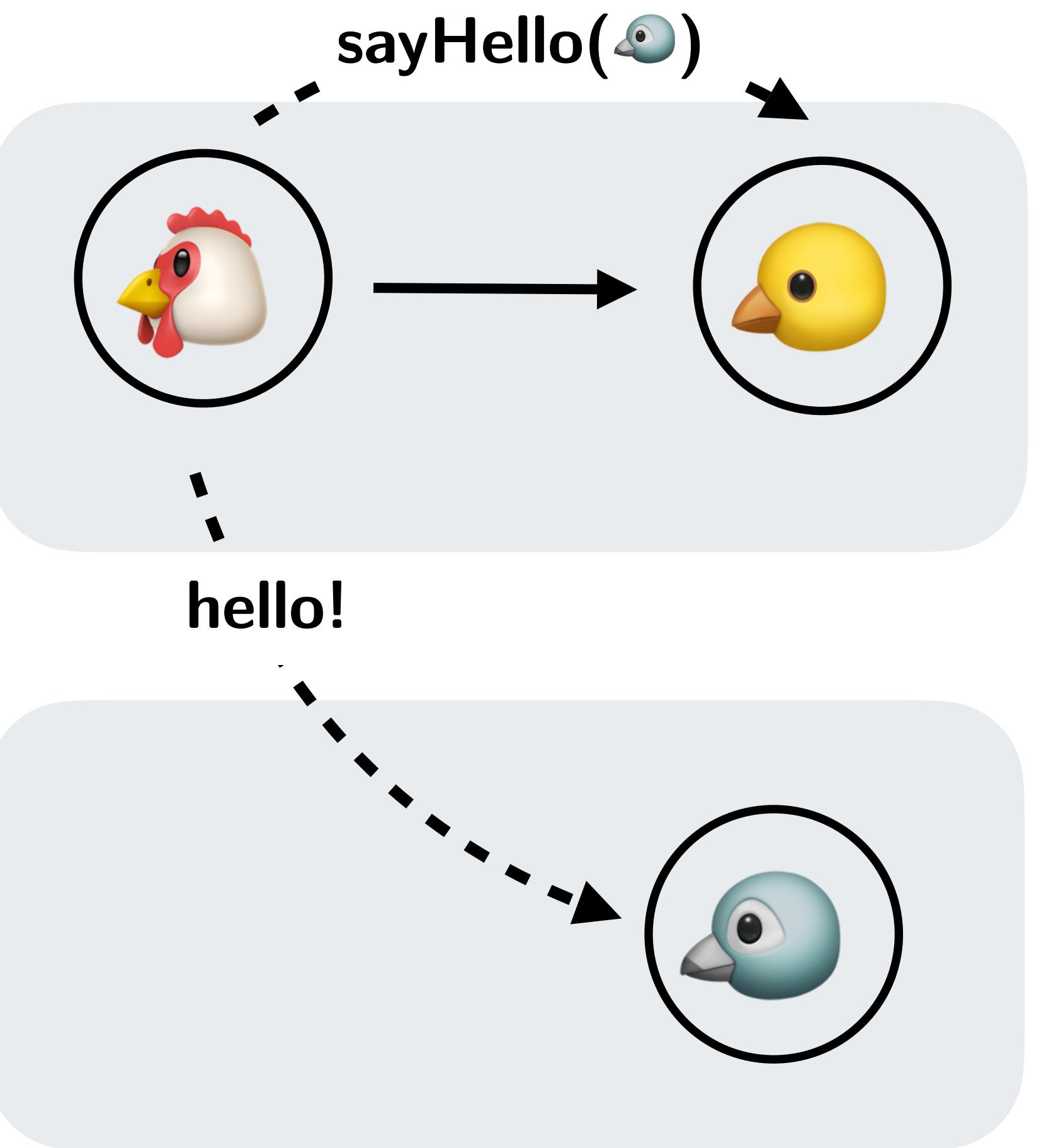


- busy actor
- idle actor
- reference
- message



who is garbage?

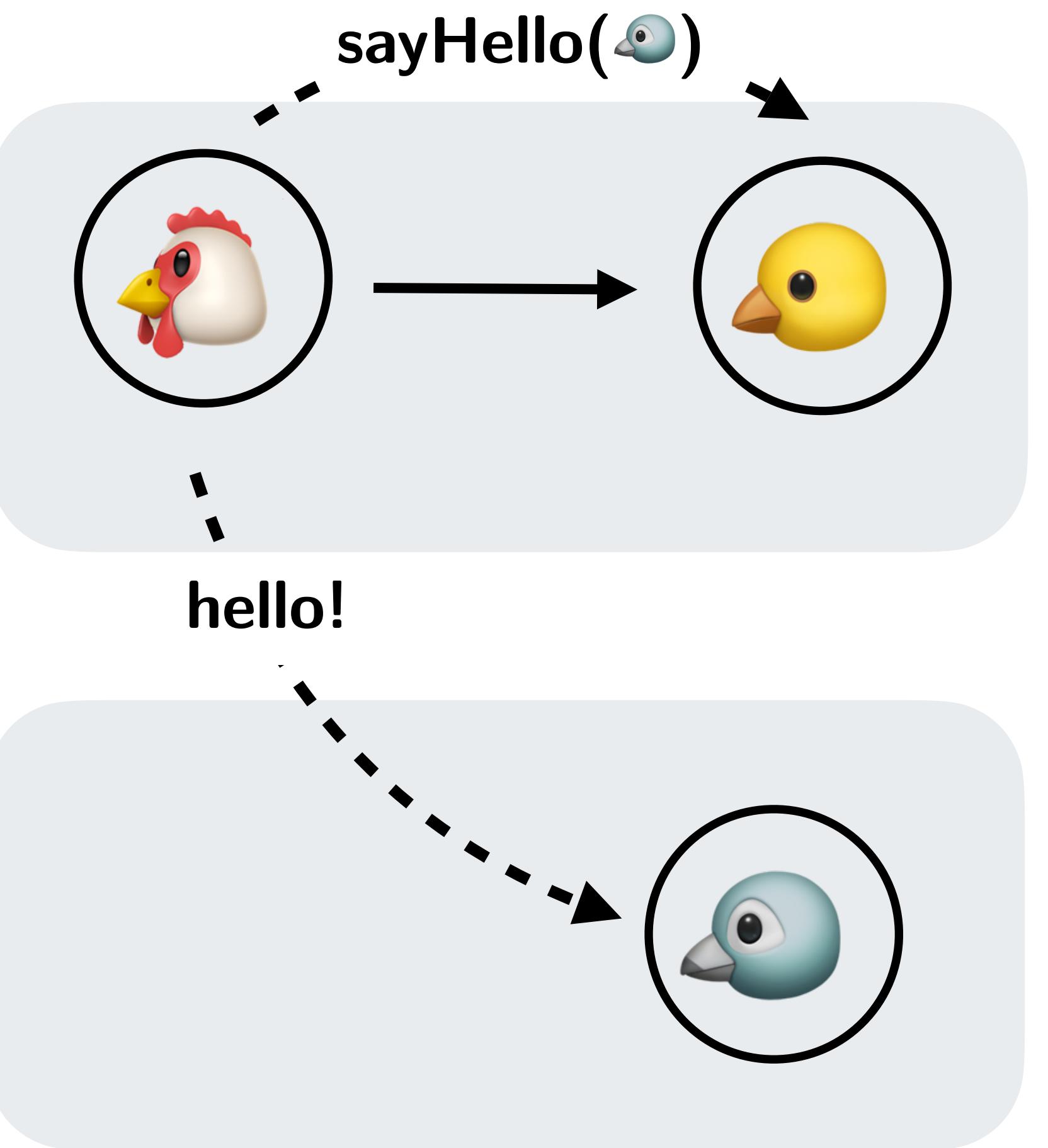
- busy actor
- idle actor
- reference
- message



who is garbage?



- busy actor
- idle actor
- reference
- message

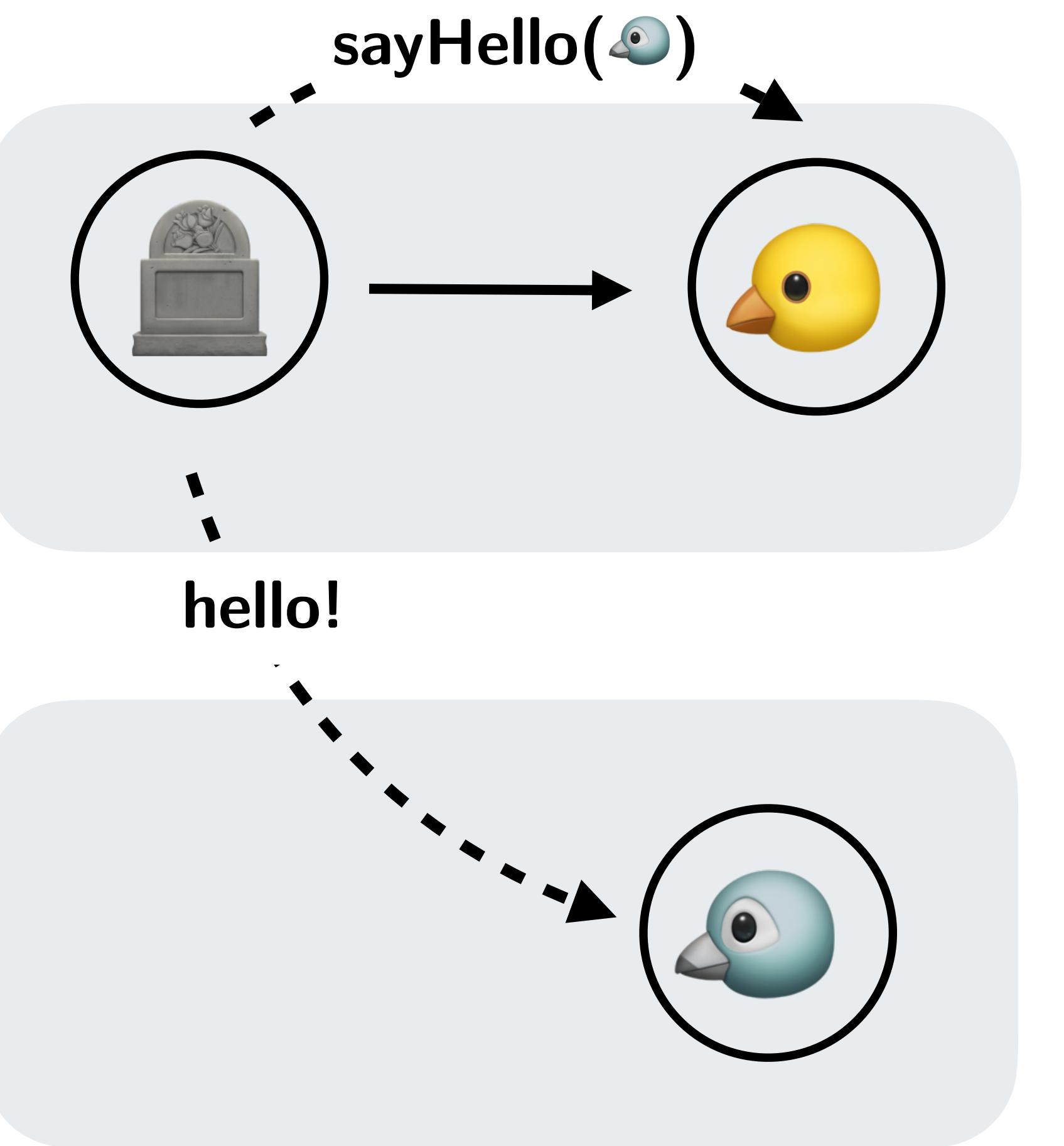


who is garbage?

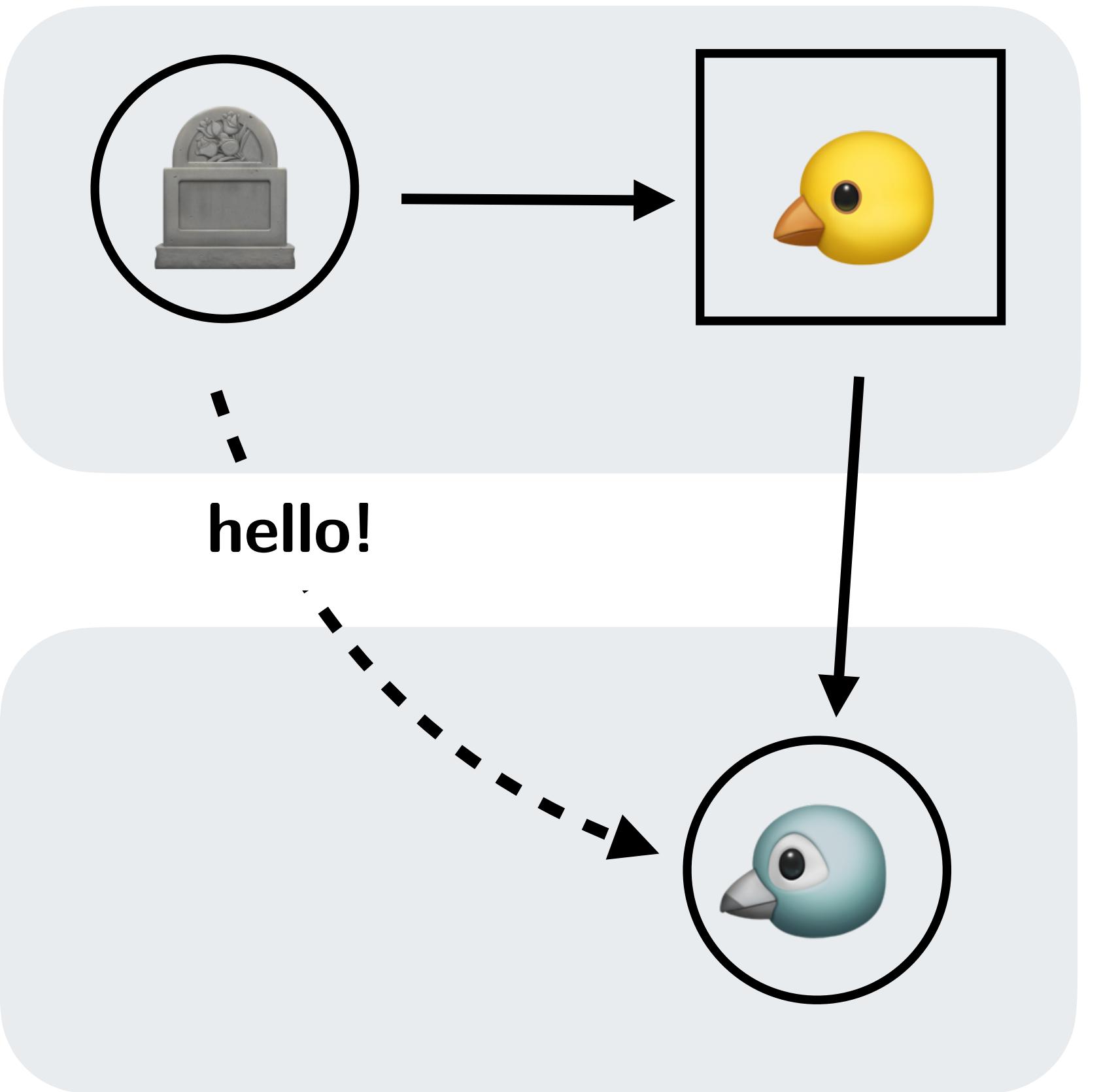


actors are **reactive**
and **capability-secure**

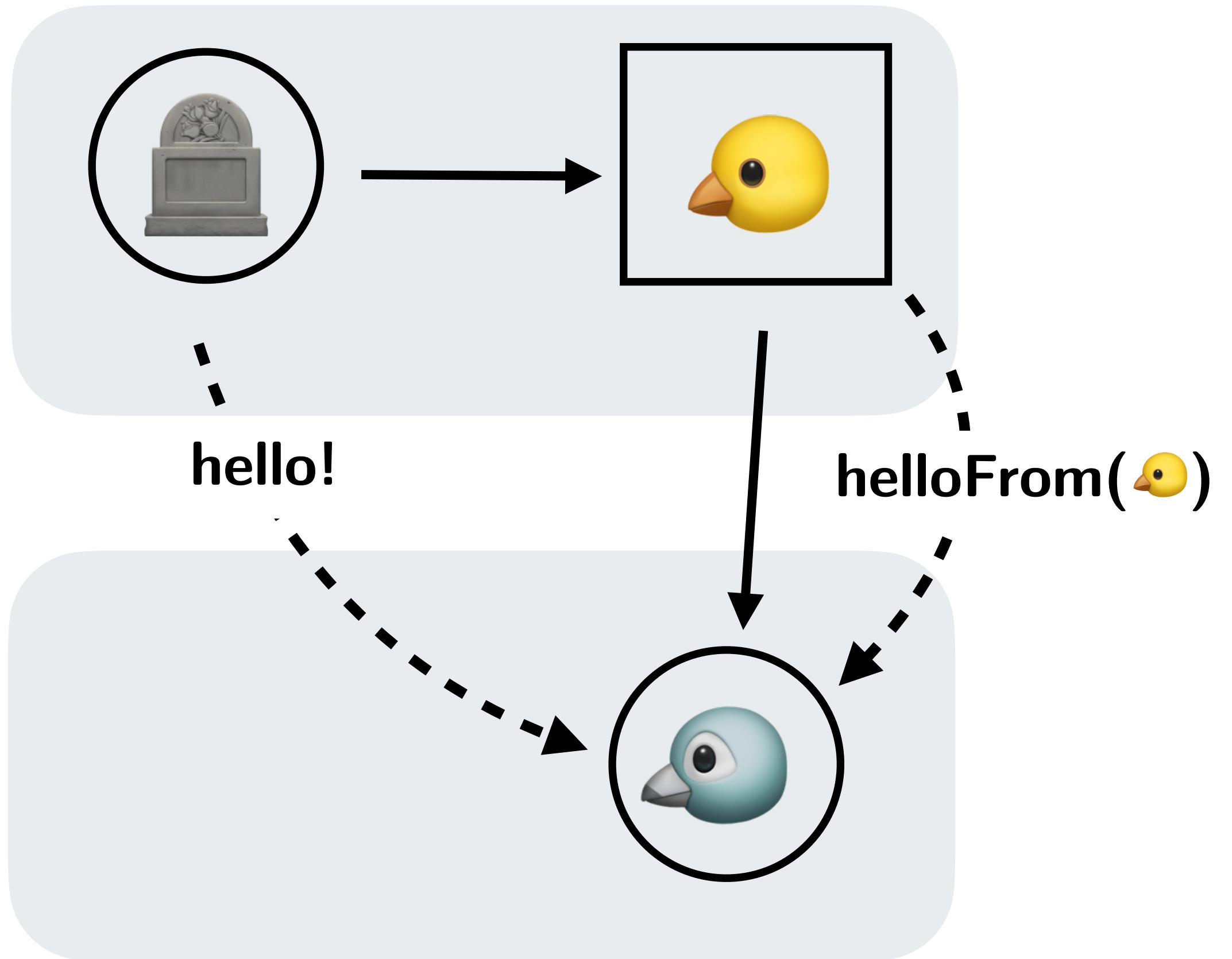
- busy actor
- idle actor
- reference
- -> message



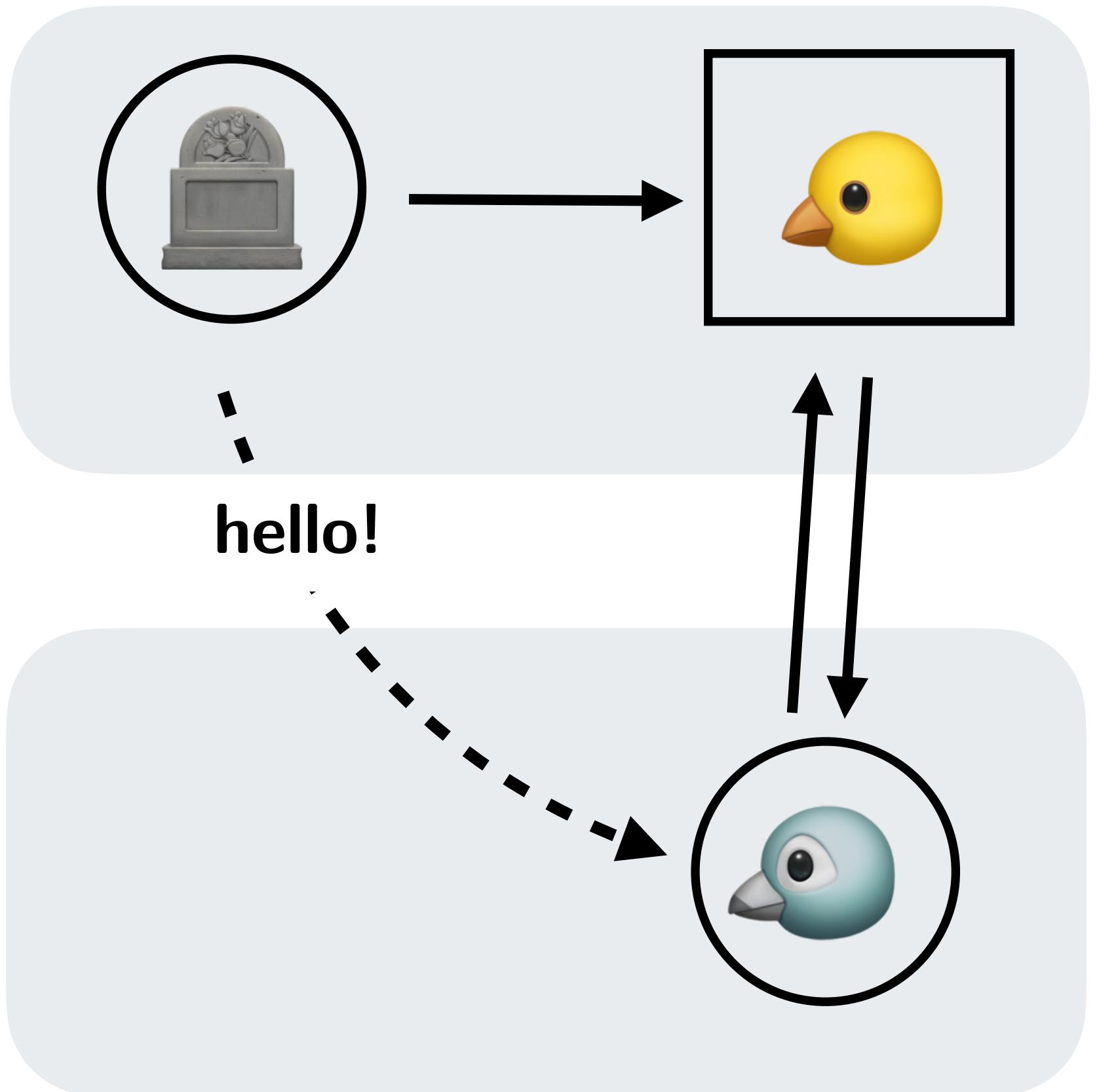
- busy actor
- idle actor
- reference
- message



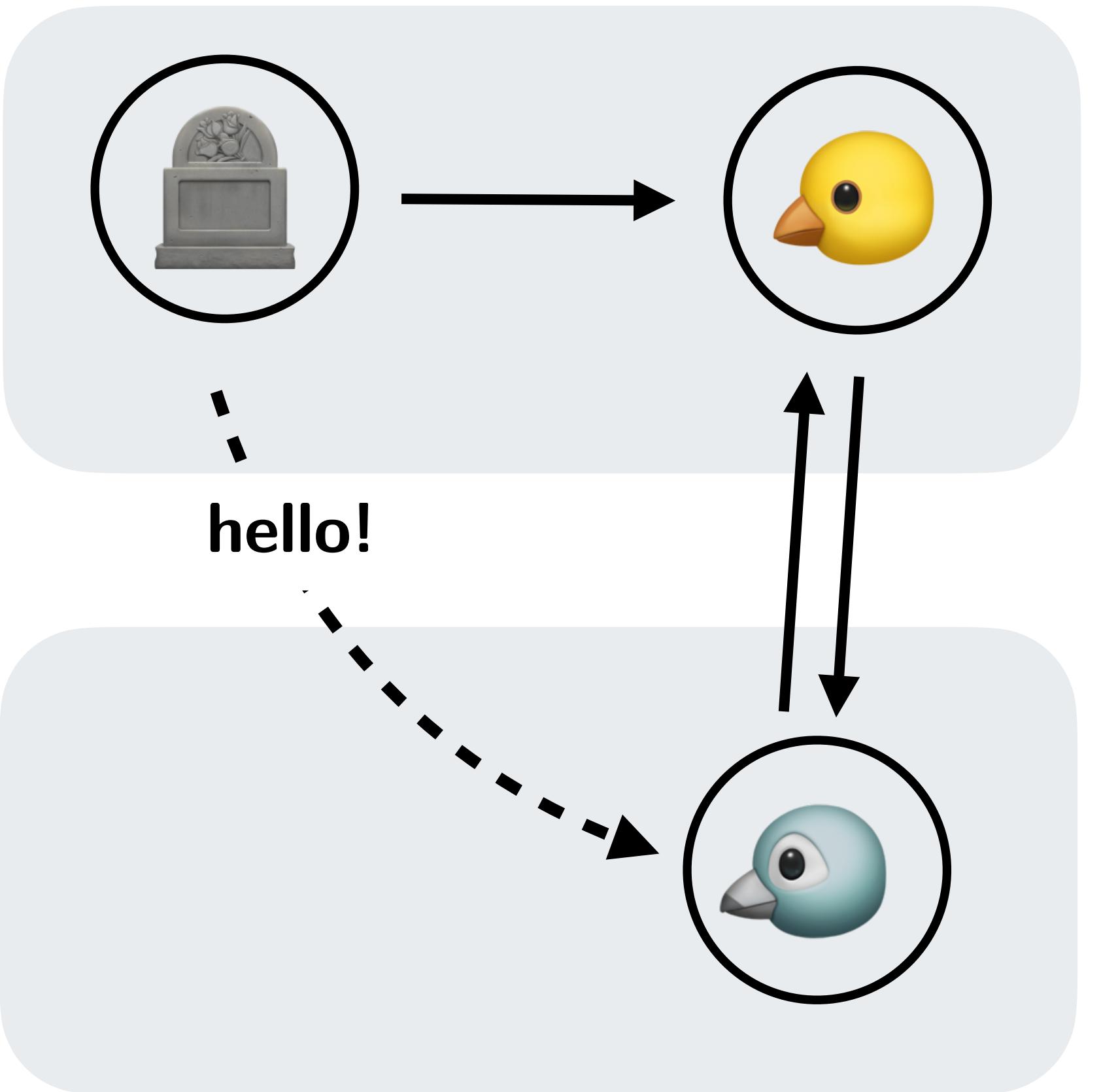
- busy actor
- idle actor
- reference
- message



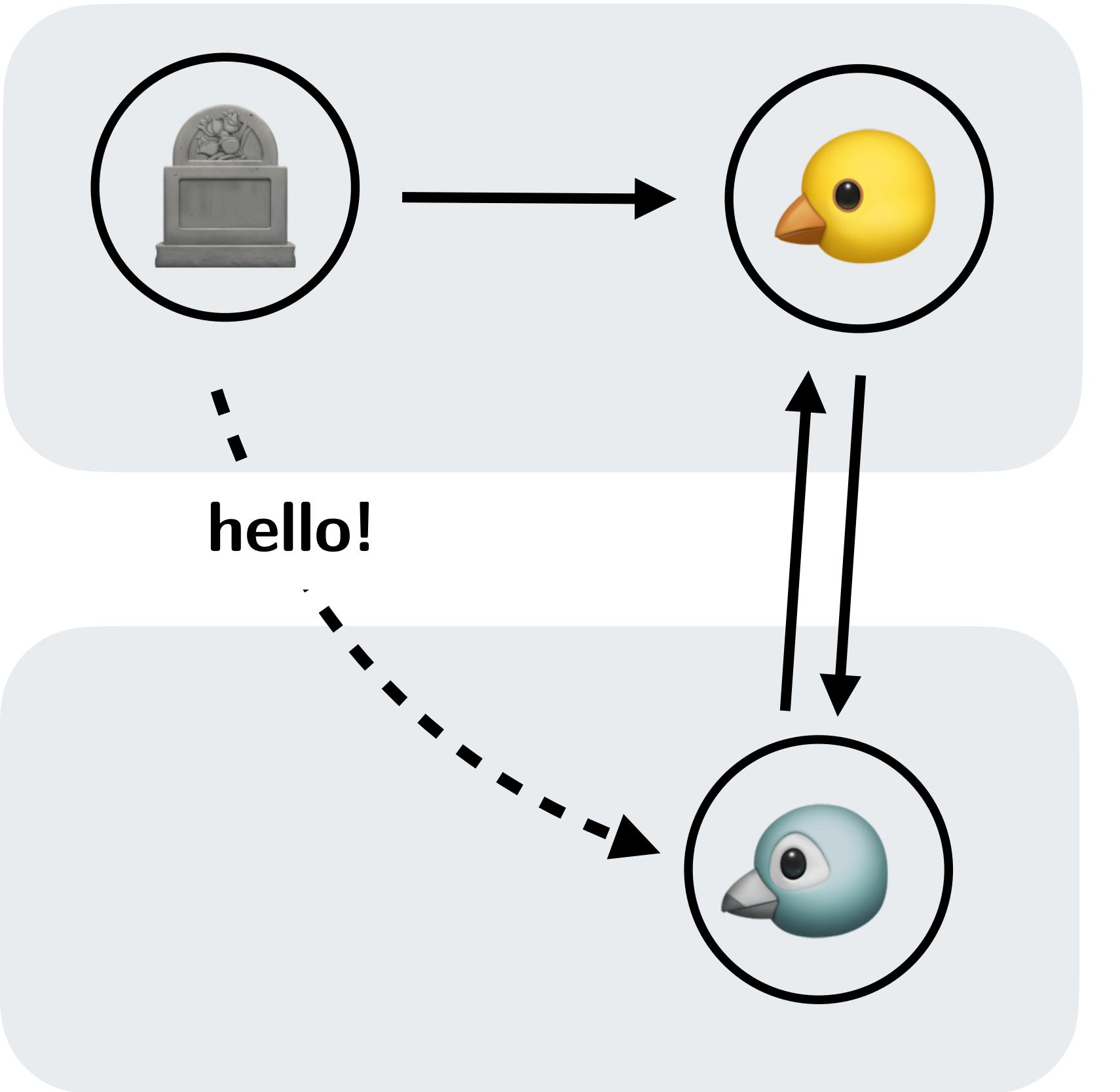
- busy actor
- idle actor
- reference
- message



- busy actor
- idle actor
- reference
- message

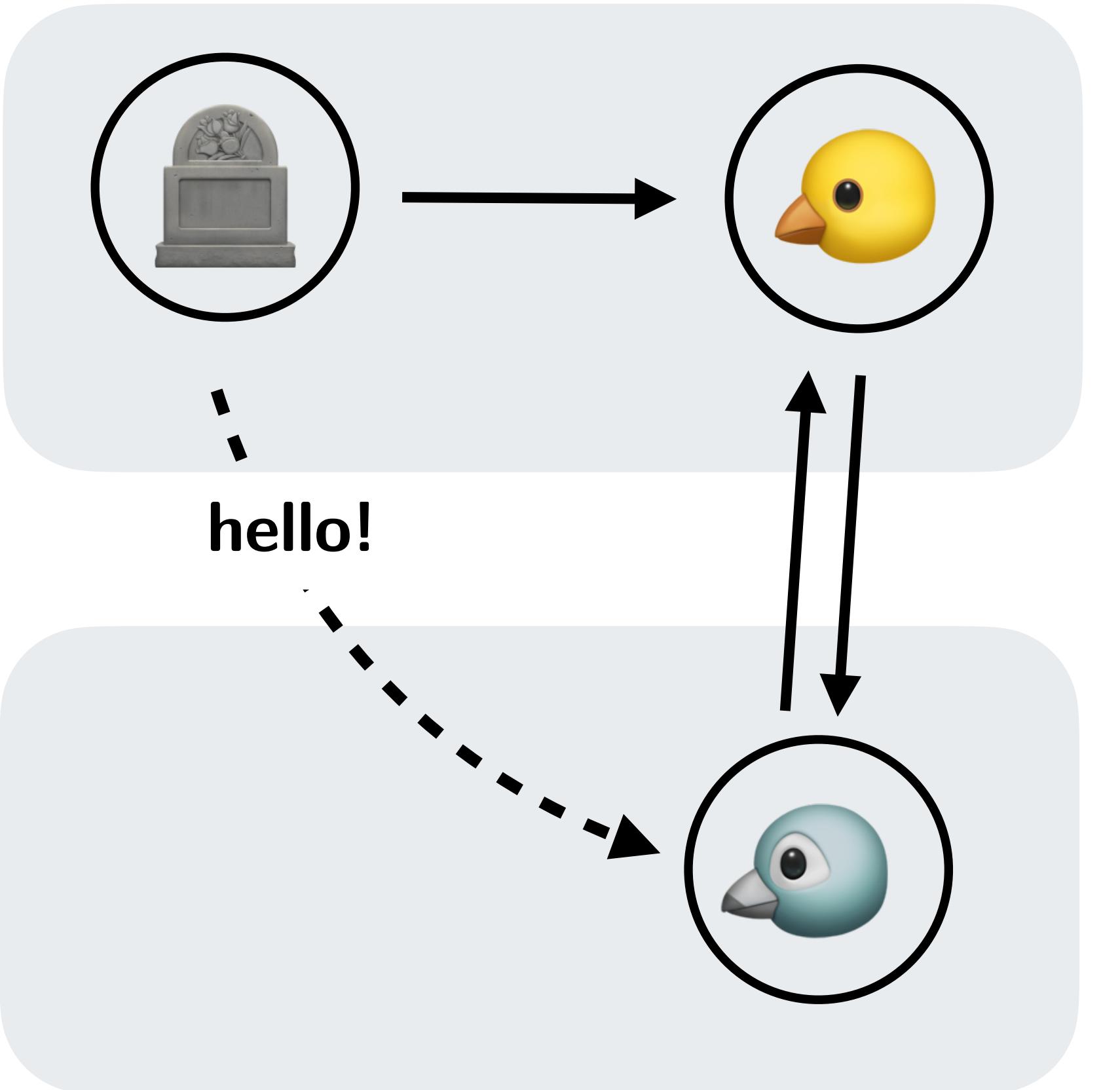


- busy actor
- idle actor
- reference
- > message



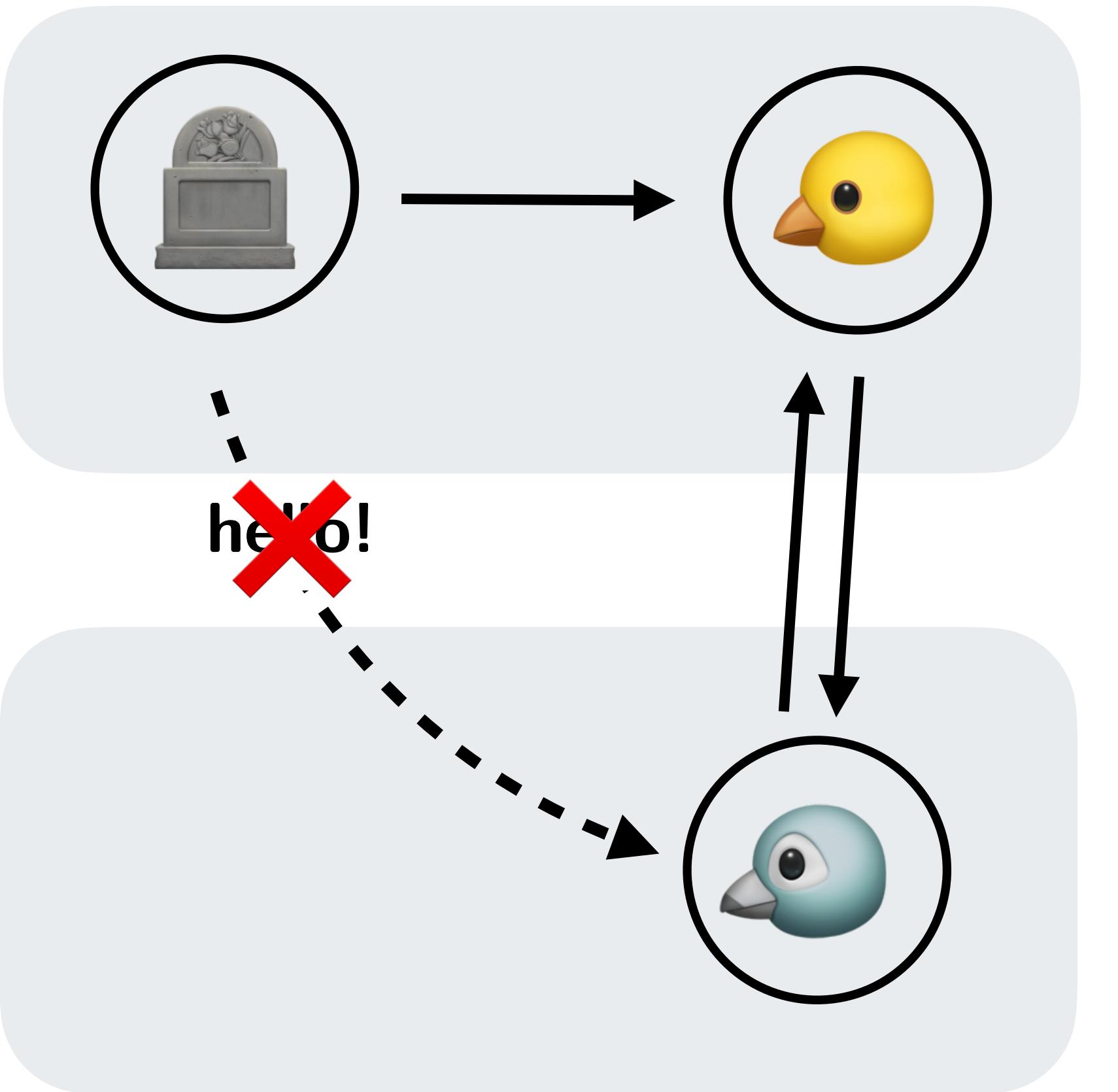
who is garbage?

- busy actor
- idle actor
- reference
- > message

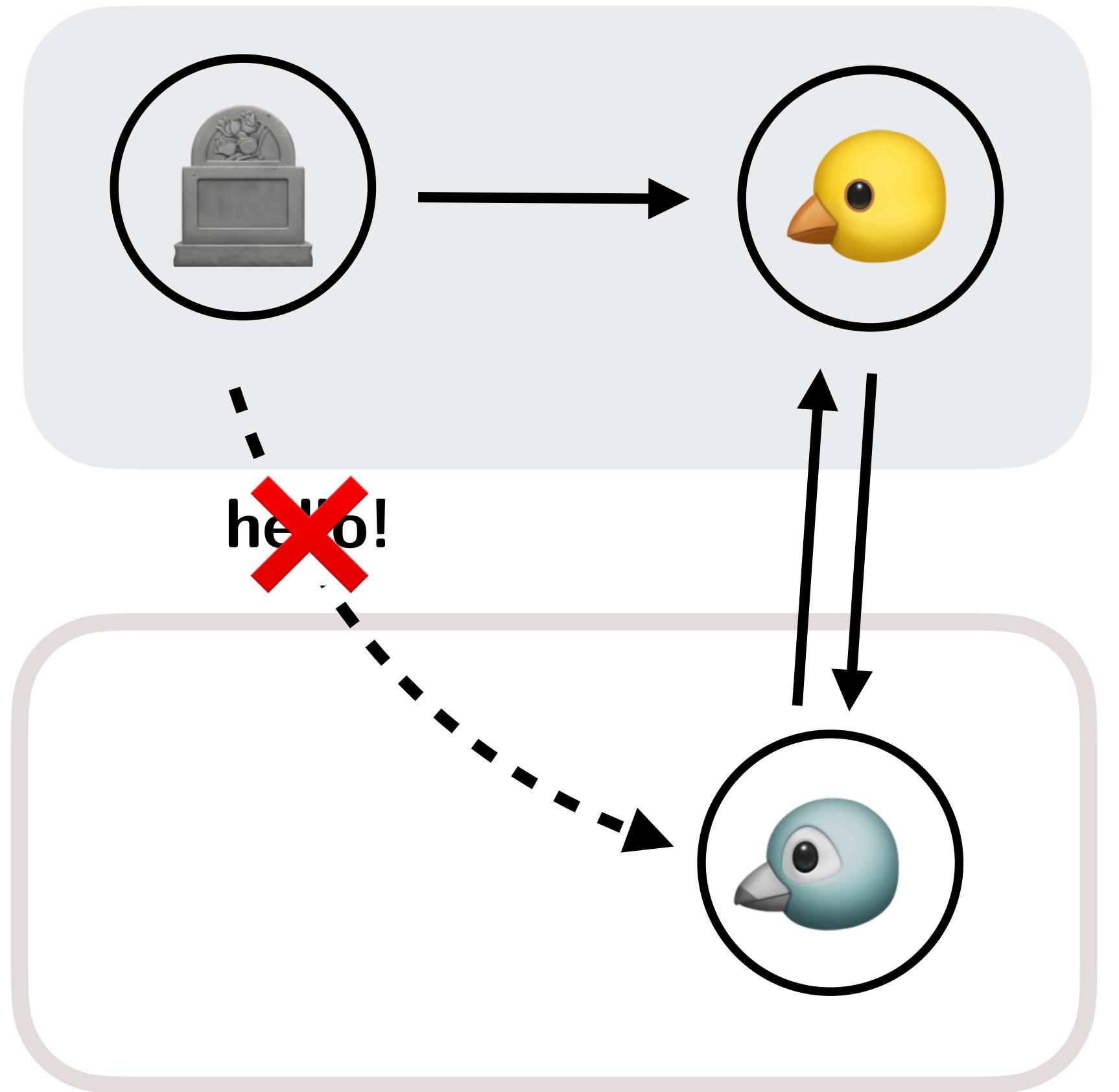


who is garbage?
nobody!

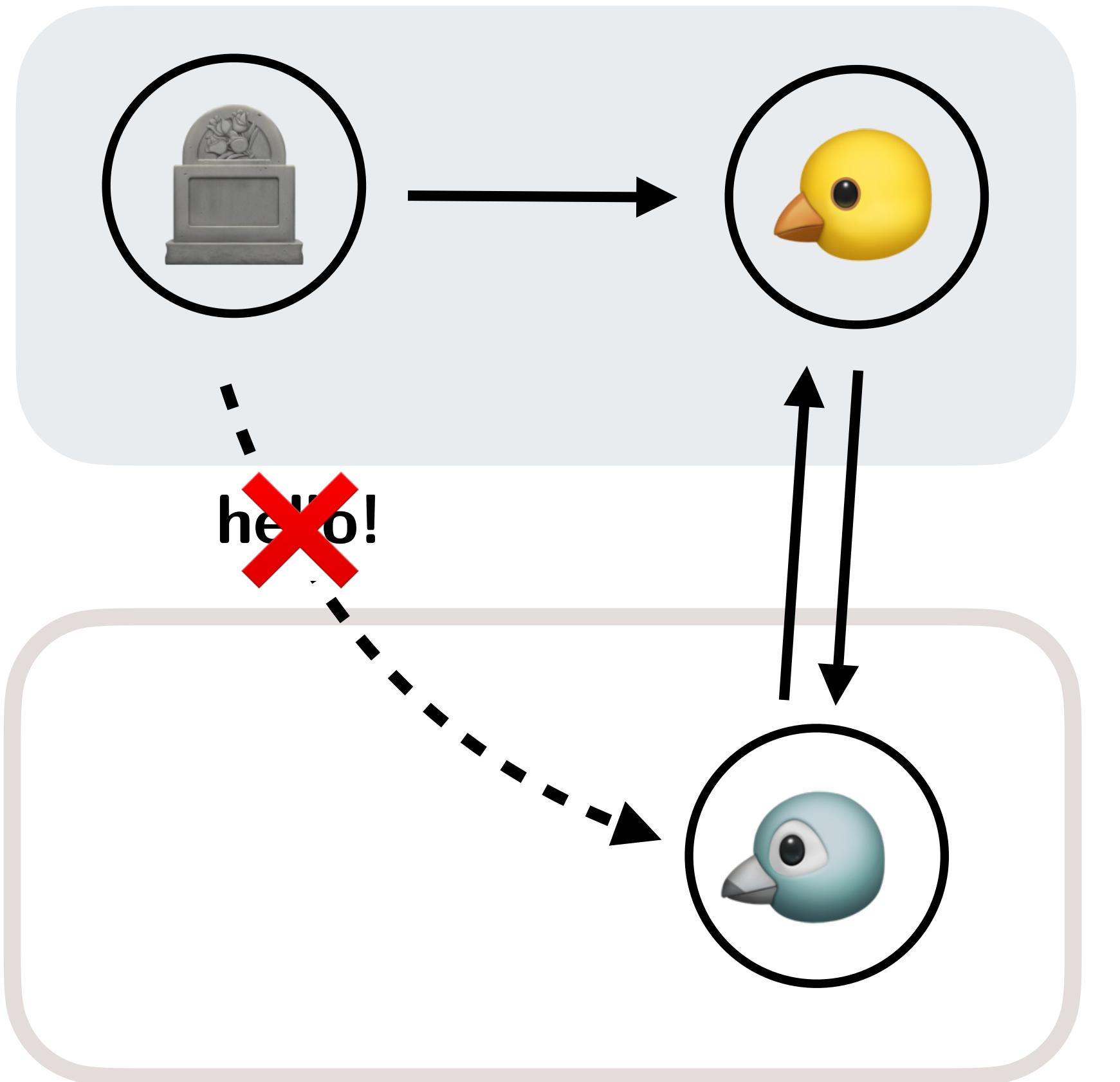
- busy actor
- idle actor
- reference
- message



- busy actor
- idle actor
- reference
- message
- healthy node
- crashed node

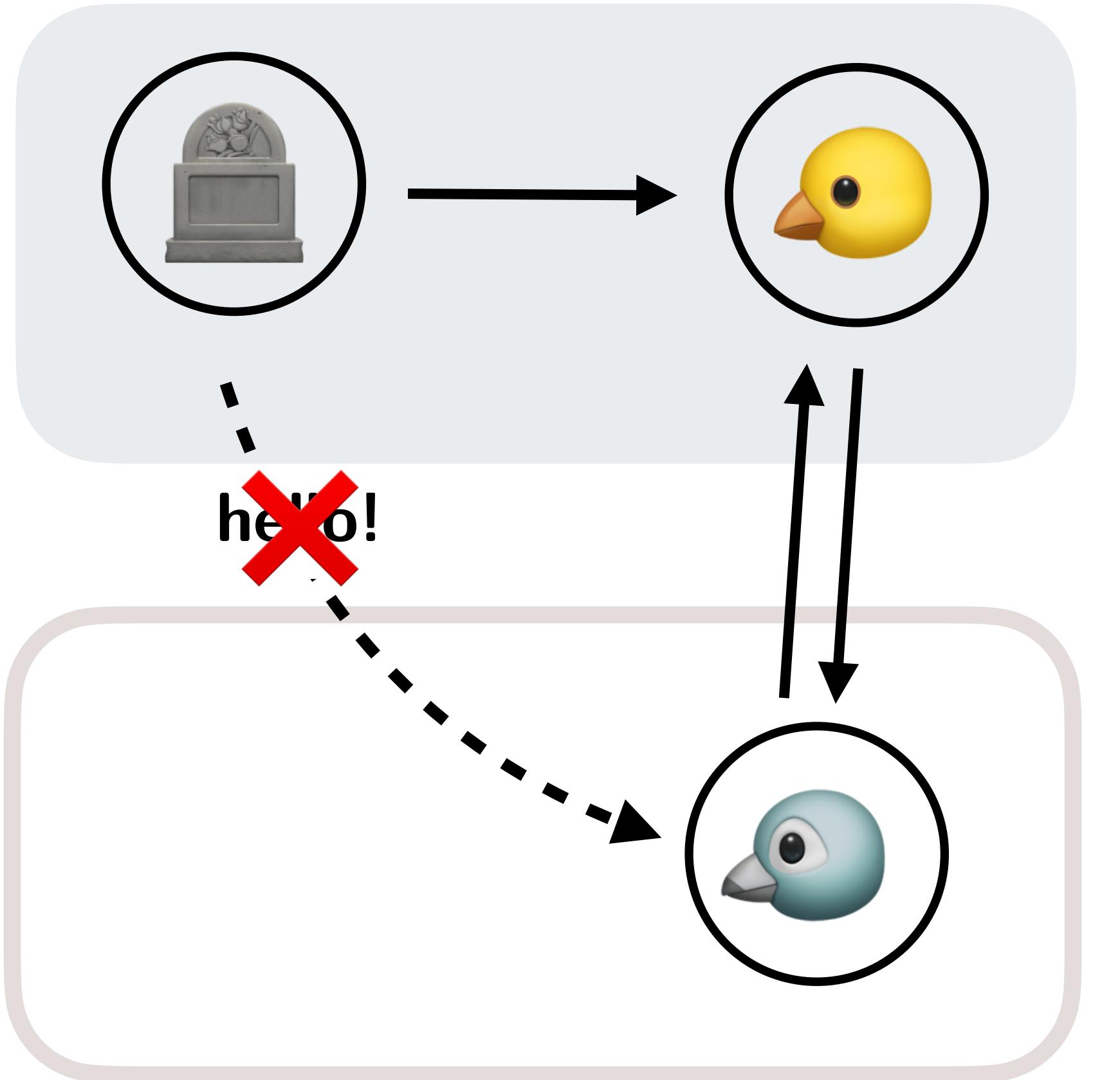


- busy actor
- idle actor
- reference
- message
- healthy node
- crashed node



who is garbage?

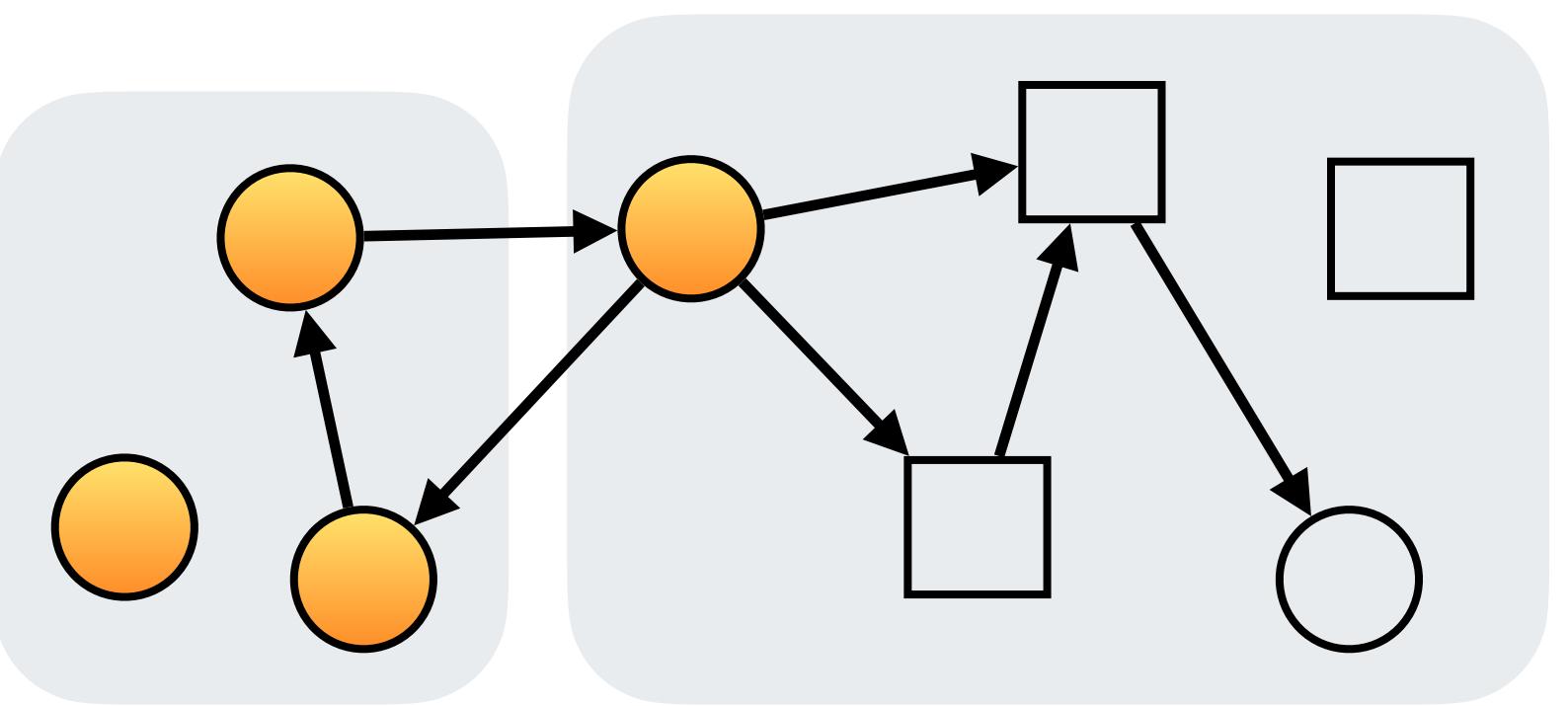
- busy actor
- idle actor
- reference
- message
- healthy node
- crashed node



who is garbage?
everybody!

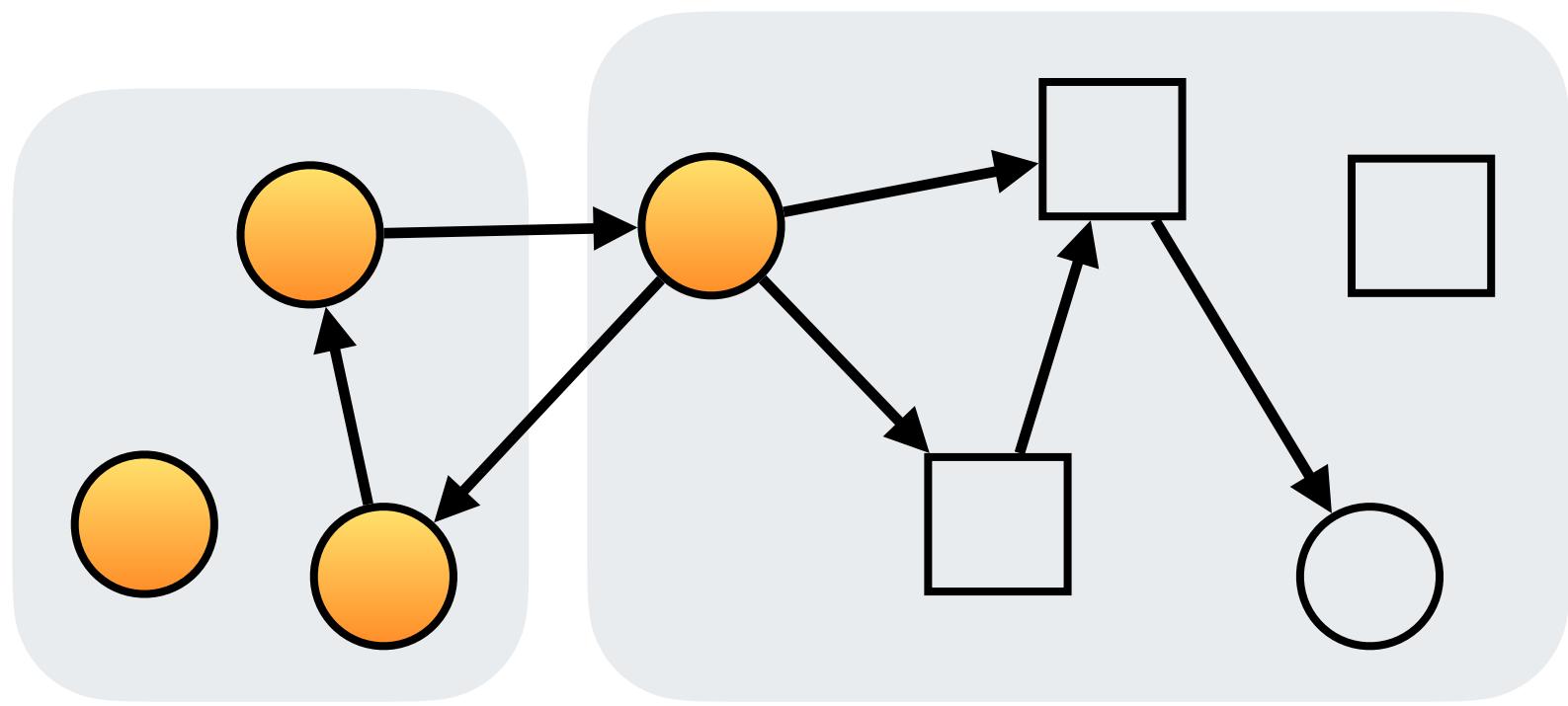
- busy actor
- idle actor
- reference
- message
- healthy node
- crashed node

garbage looks like:



- busy actor
- idle actor
- reference
- message
- healthy node
- crashed node

garbage looks like:



mark-and-sweep doesn't work!

Q: detecting crashed nodes?

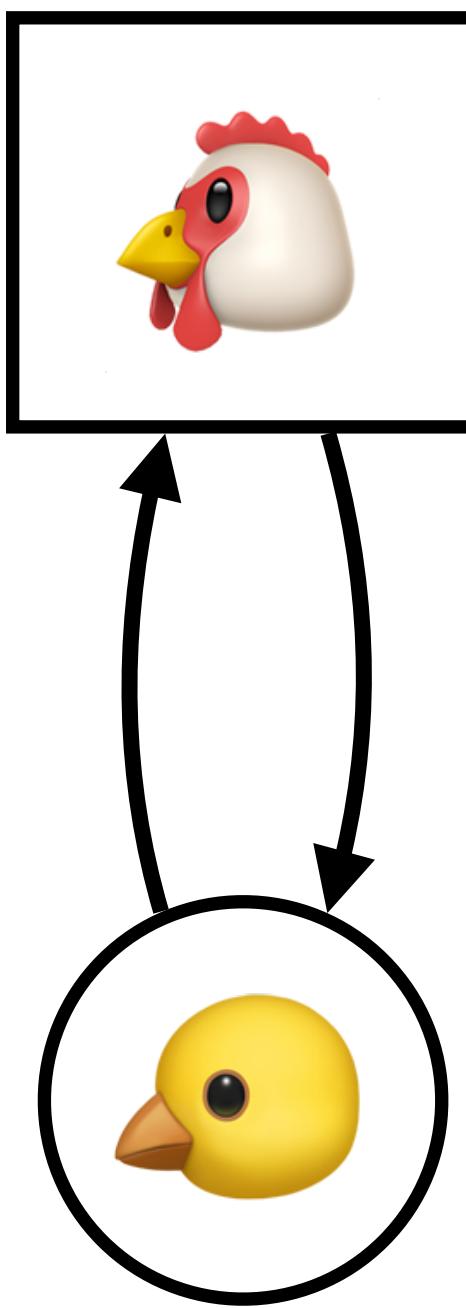
Q: detecting crashed nodes?

Q: detecting crashed nodes?

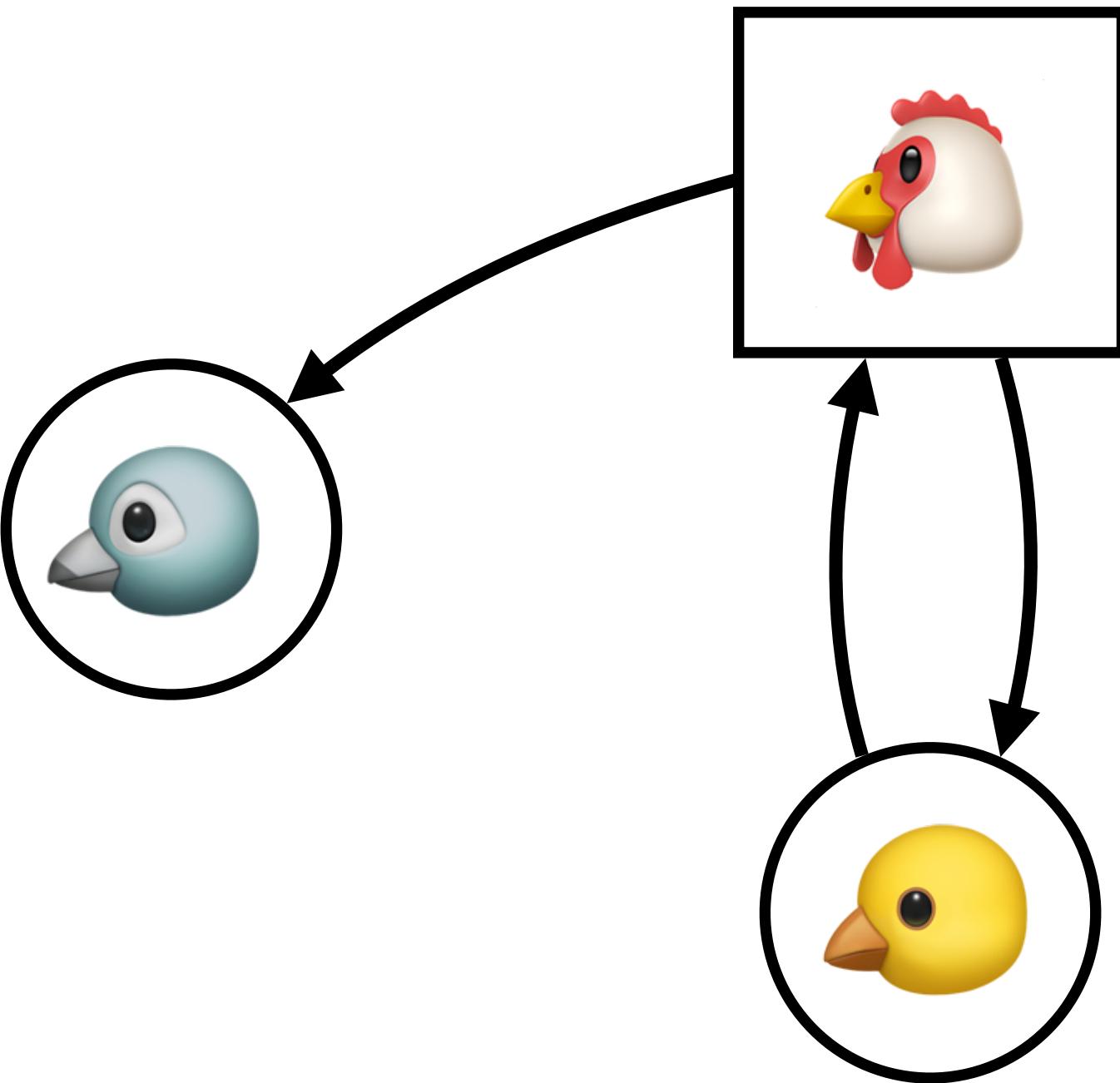
A: slow nodes are **kicked** from the cluster!

Challenge 1: Consistency

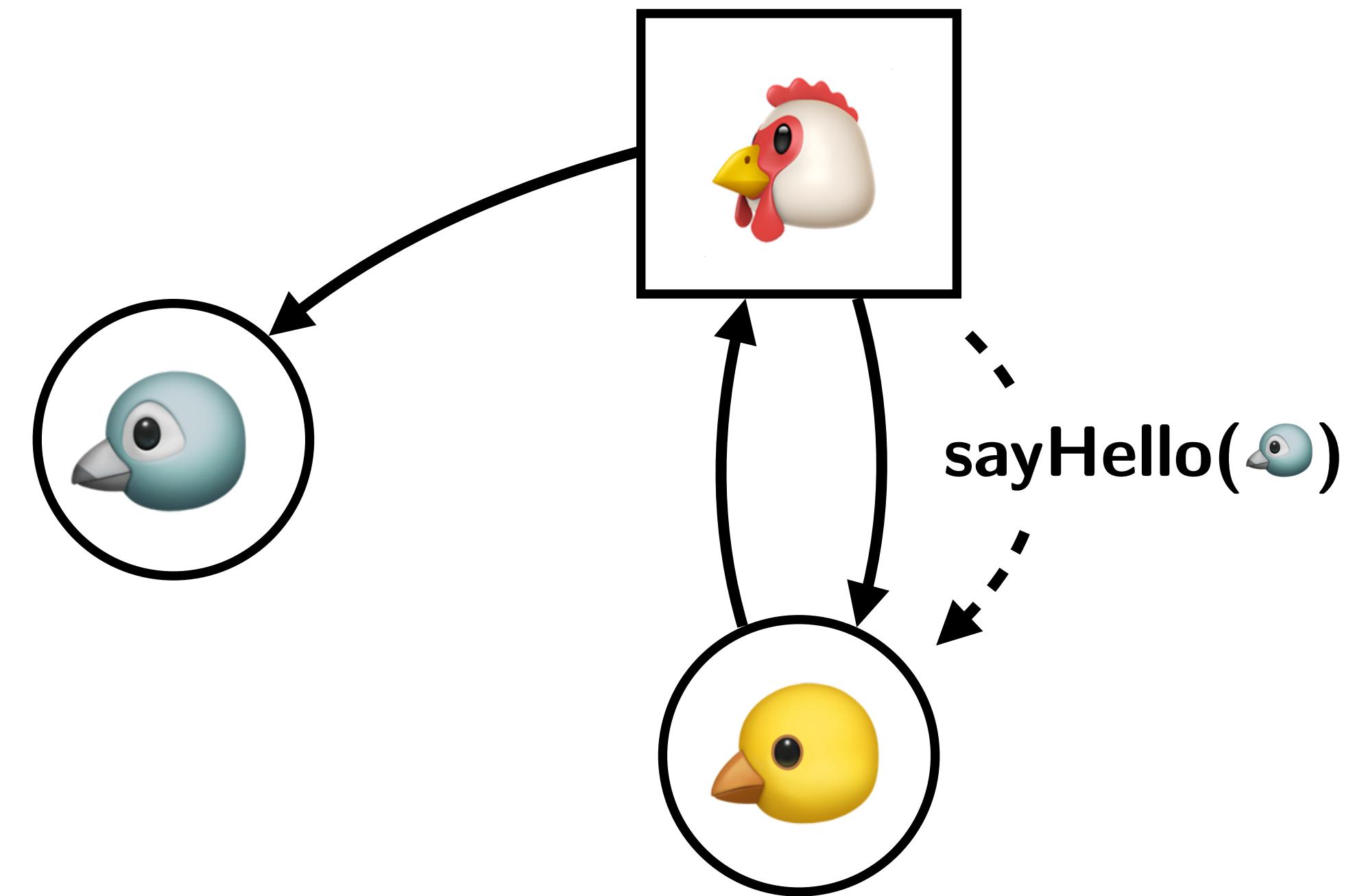
- busy actor
- idle actor
- reference
- message
- healthy node
- crashed node



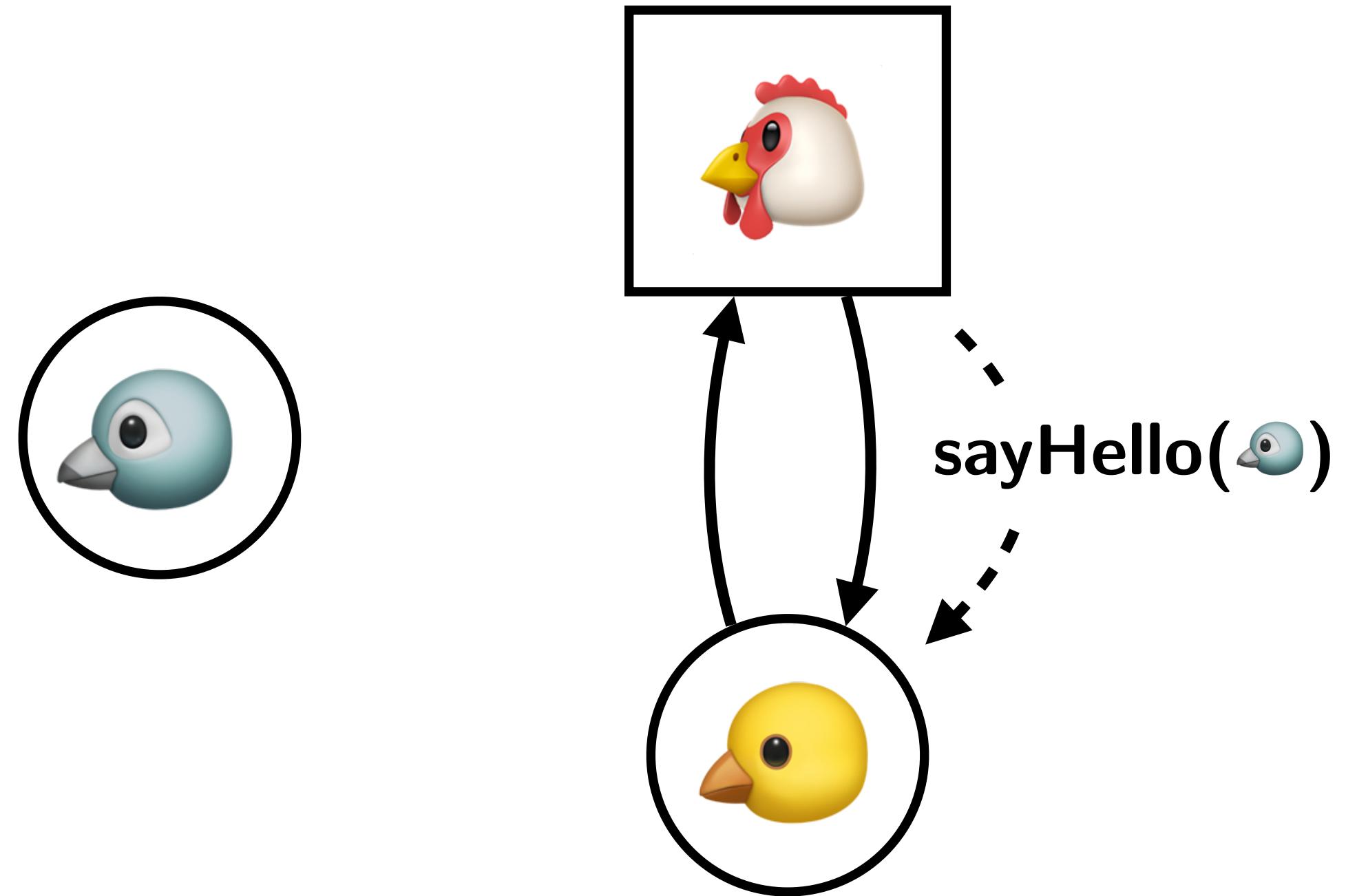
- busy actor
- idle actor
- reference
- message
- healthy node
- crashed node



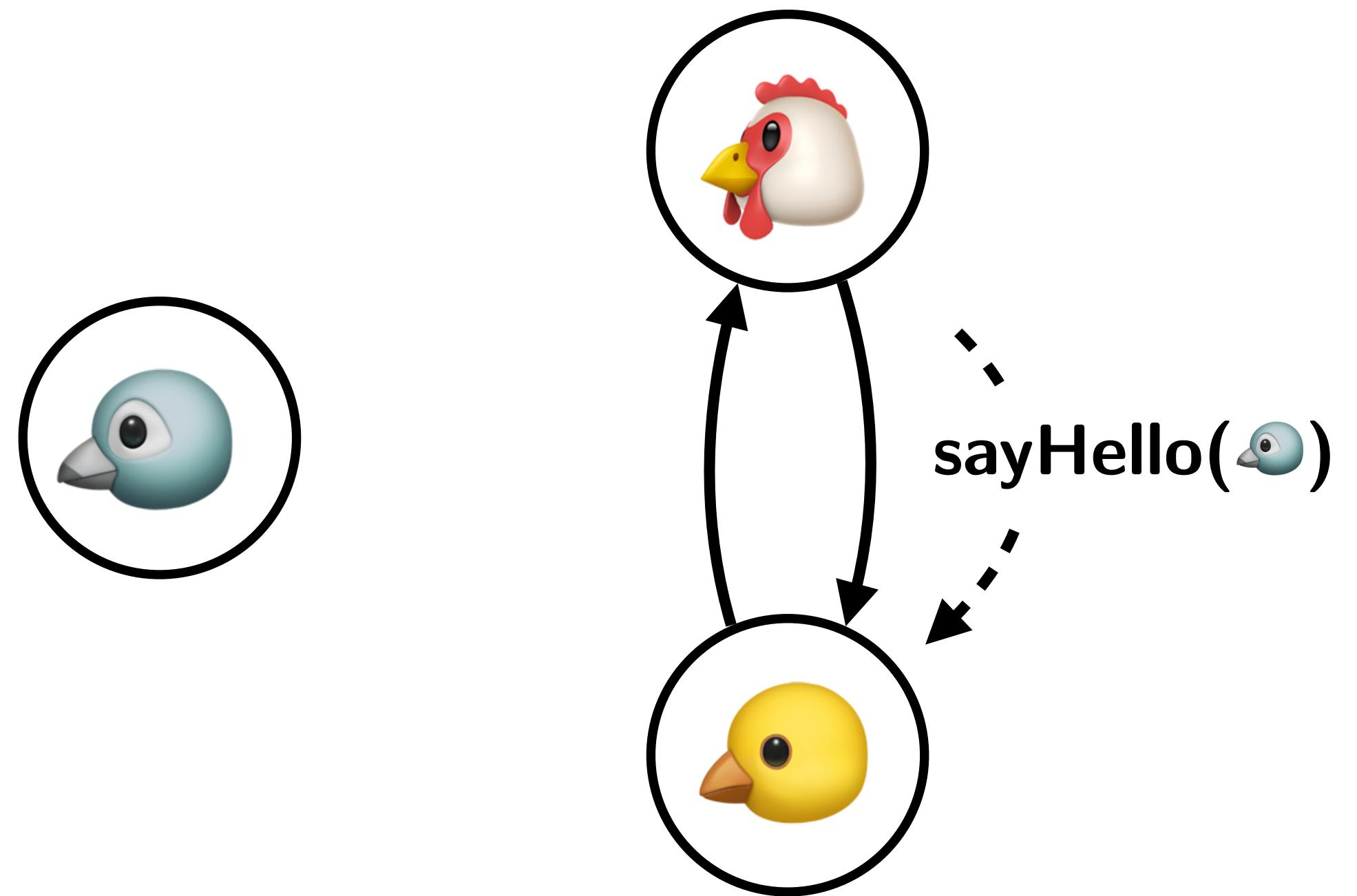
- busy actor
- idle actor
- reference
- message
- healthy node
- crashed node



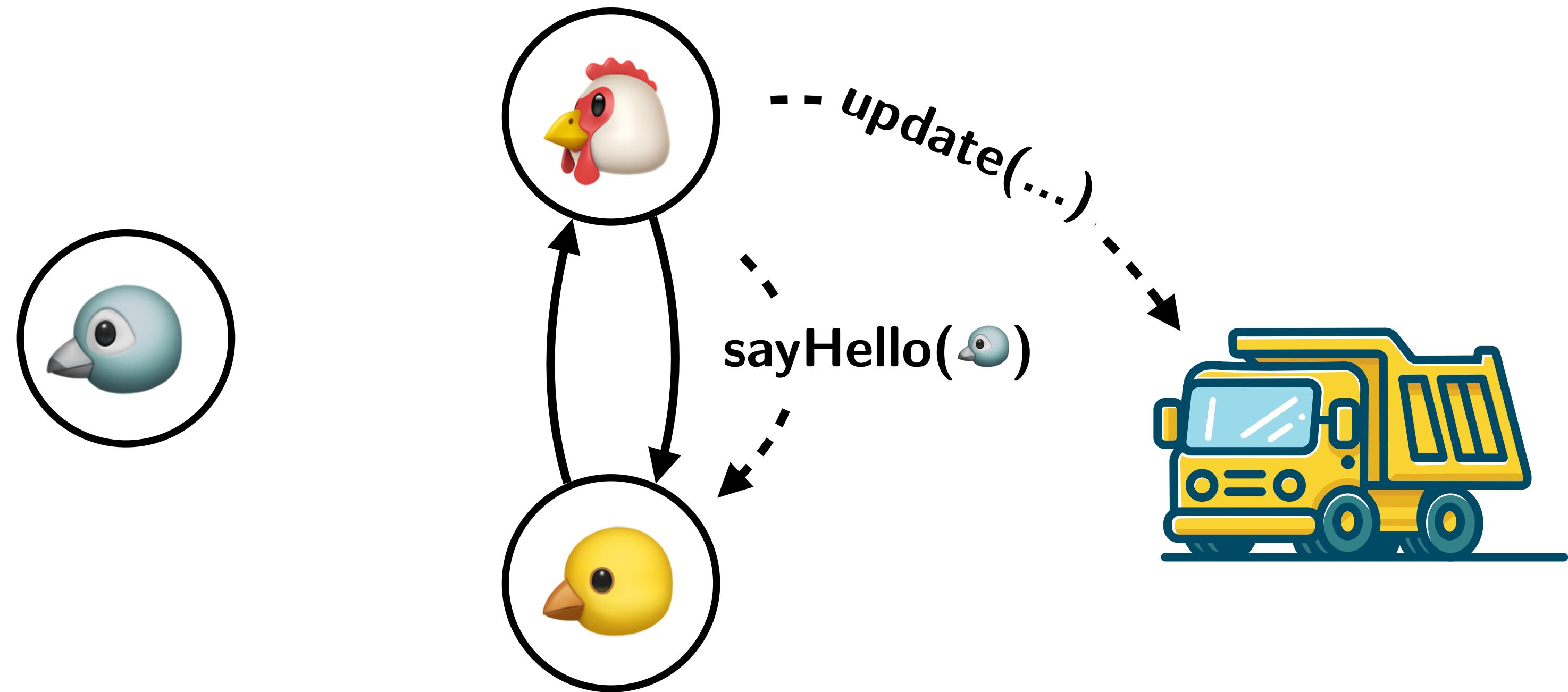
- busy actor
- idle actor
- reference
- message
- healthy node
- crashed node



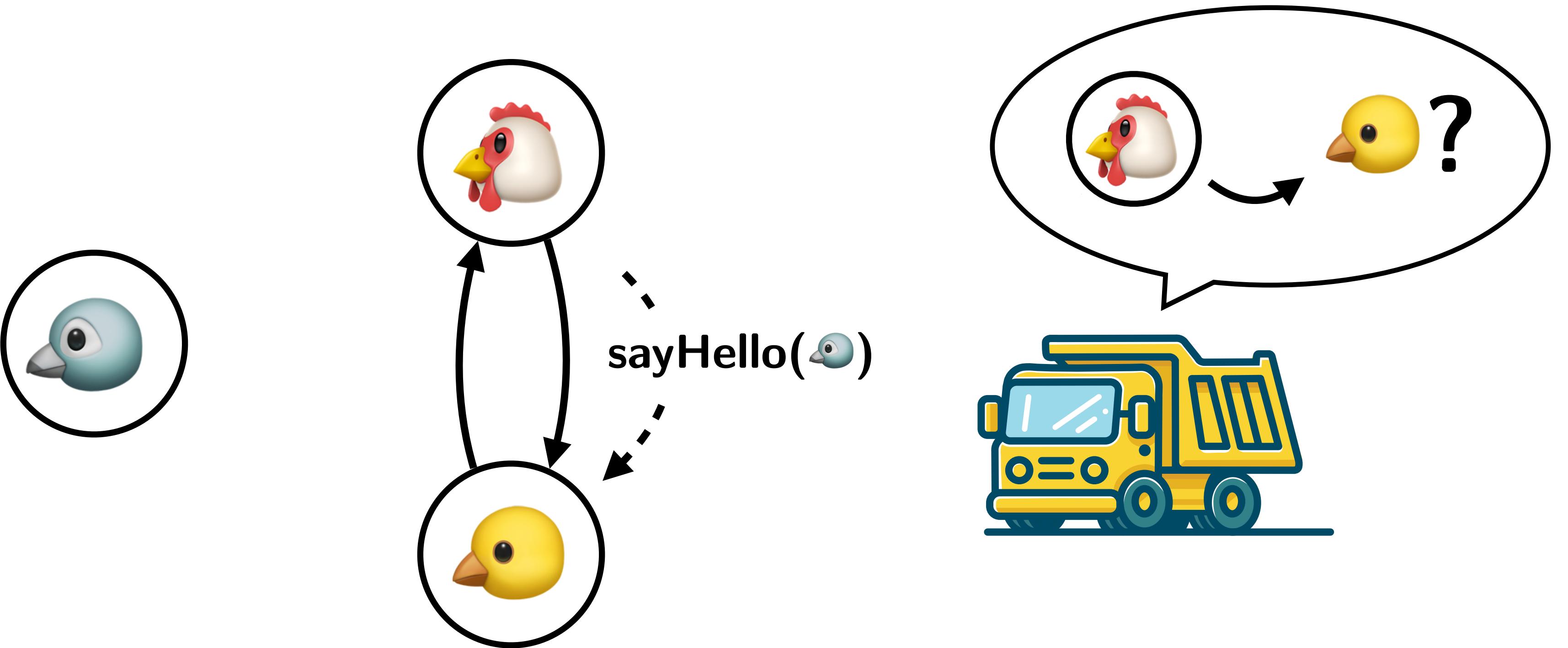
- busy actor
- idle actor
- reference
- message
- healthy node
- crashed node



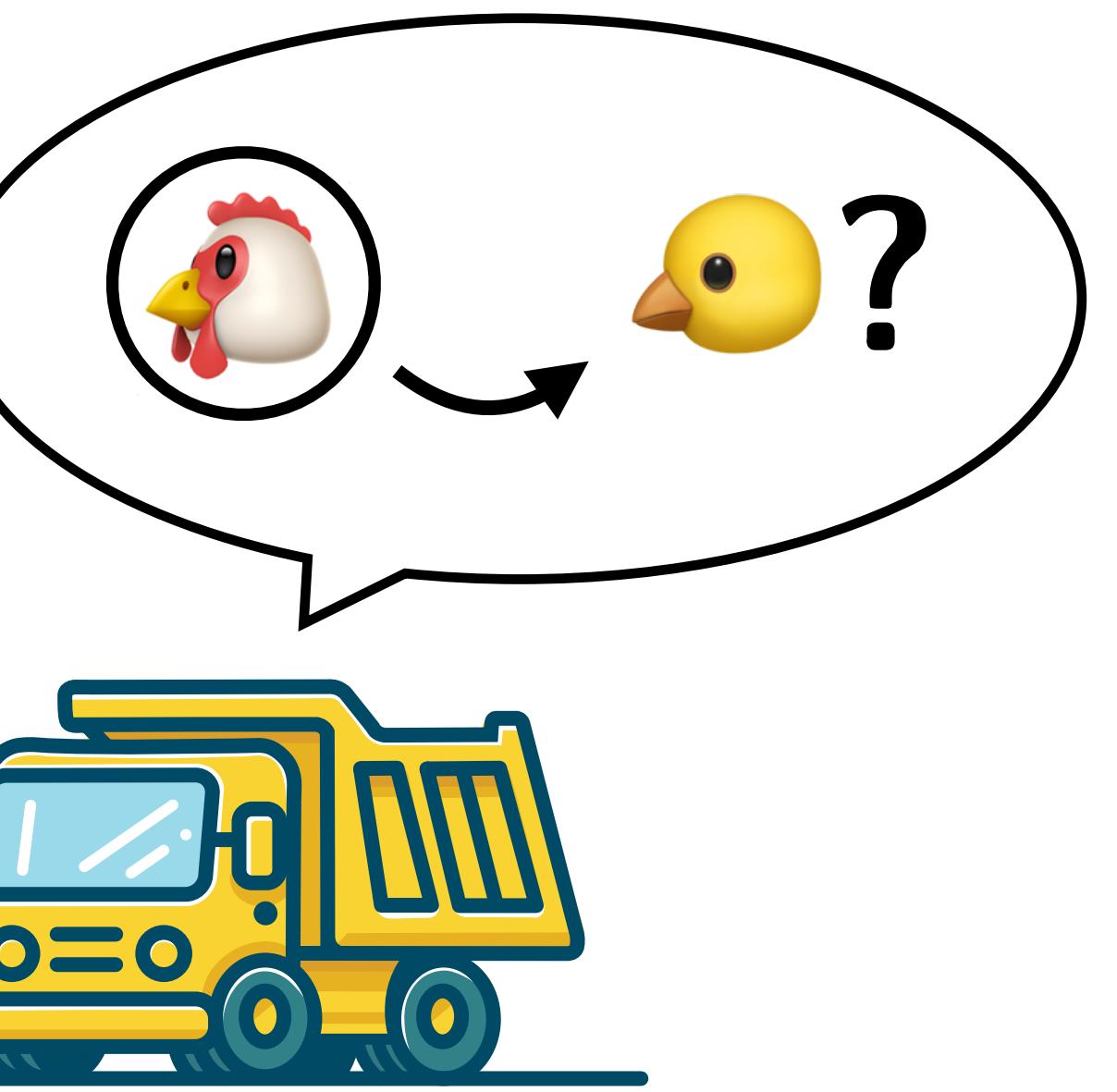
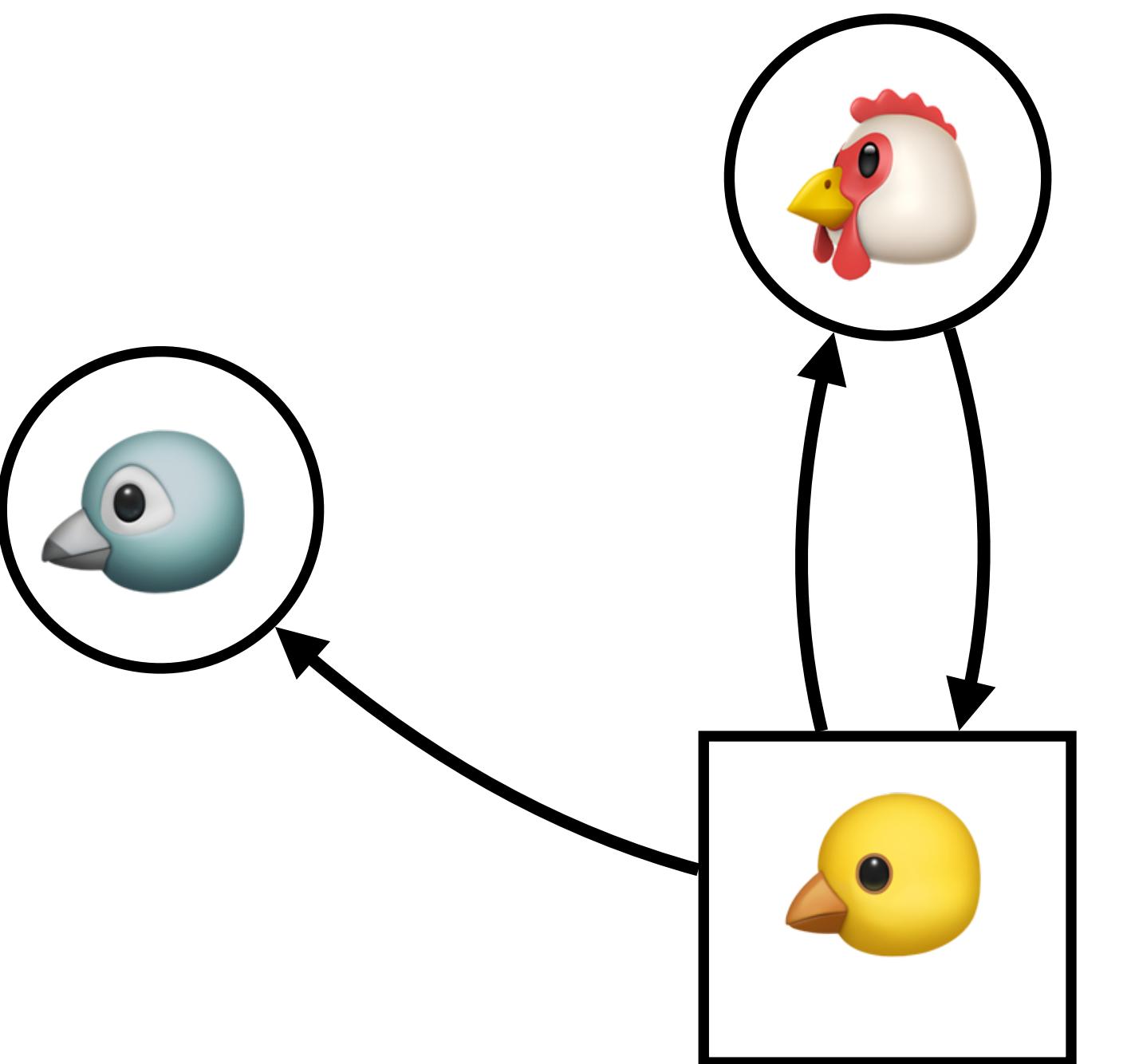
- busy actor
- idle actor
- reference
- message
- healthy node
- crashed node



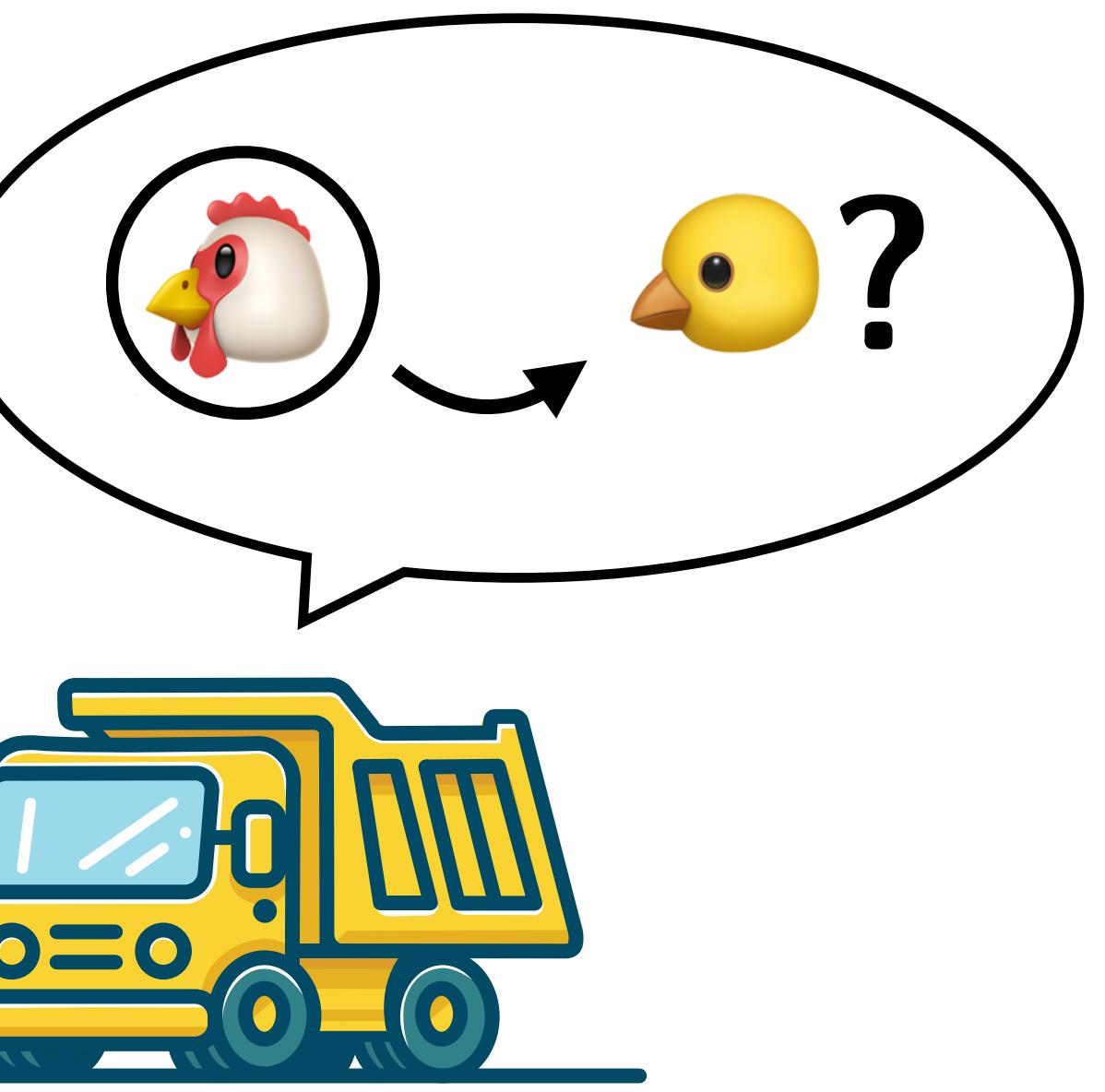
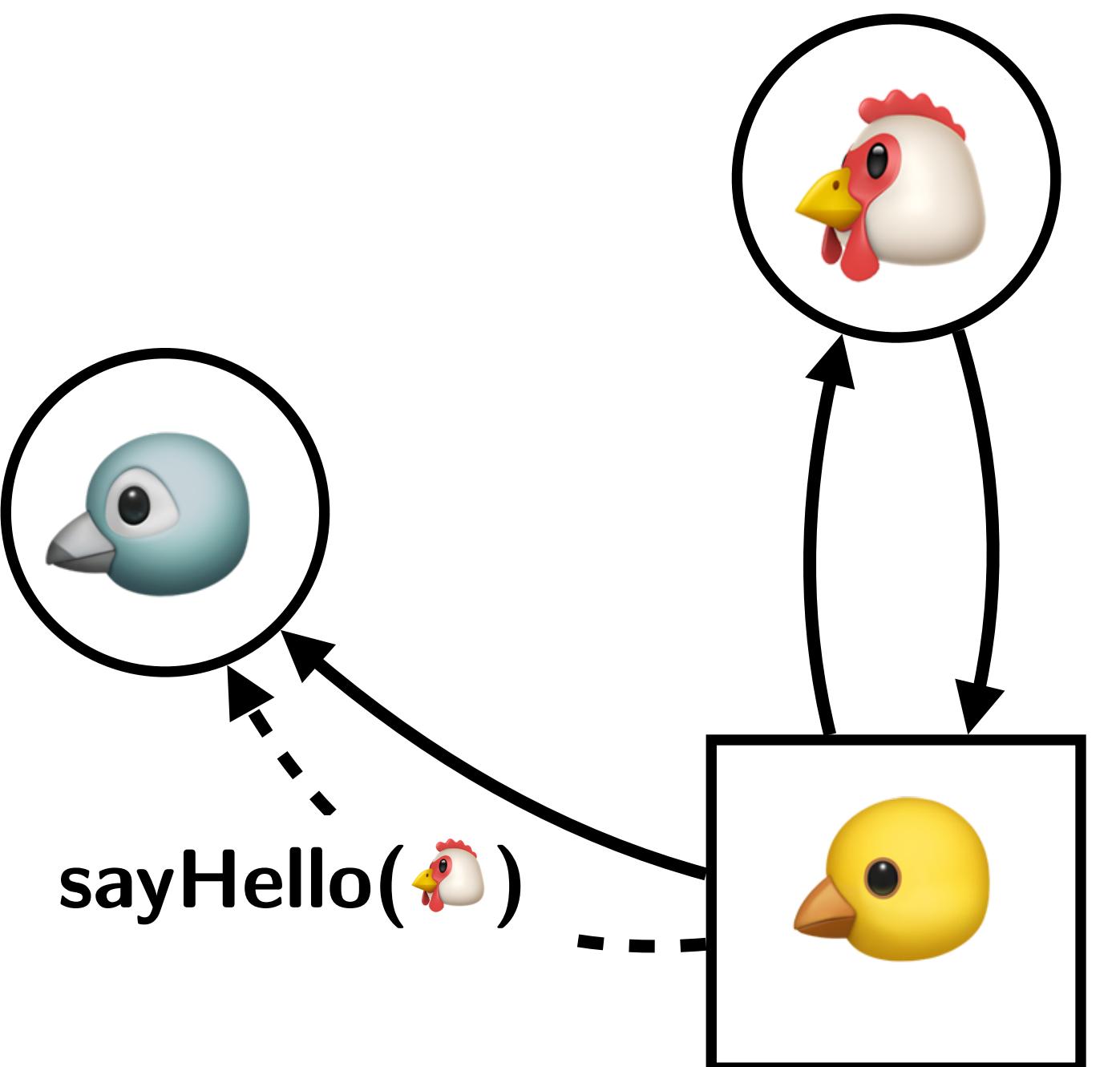
- busy actor
- idle actor
- reference
- message
- healthy node
- crashed node



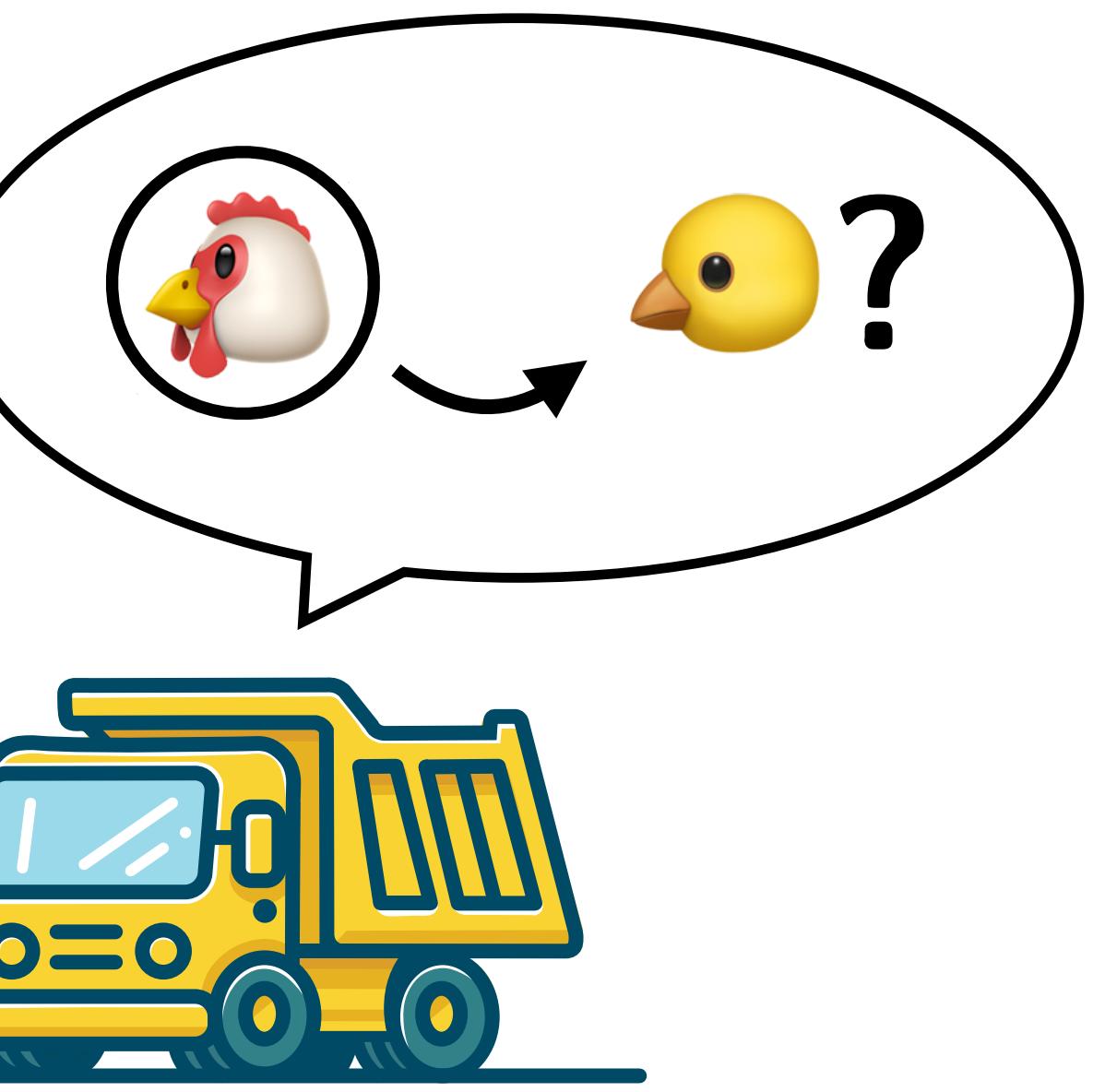
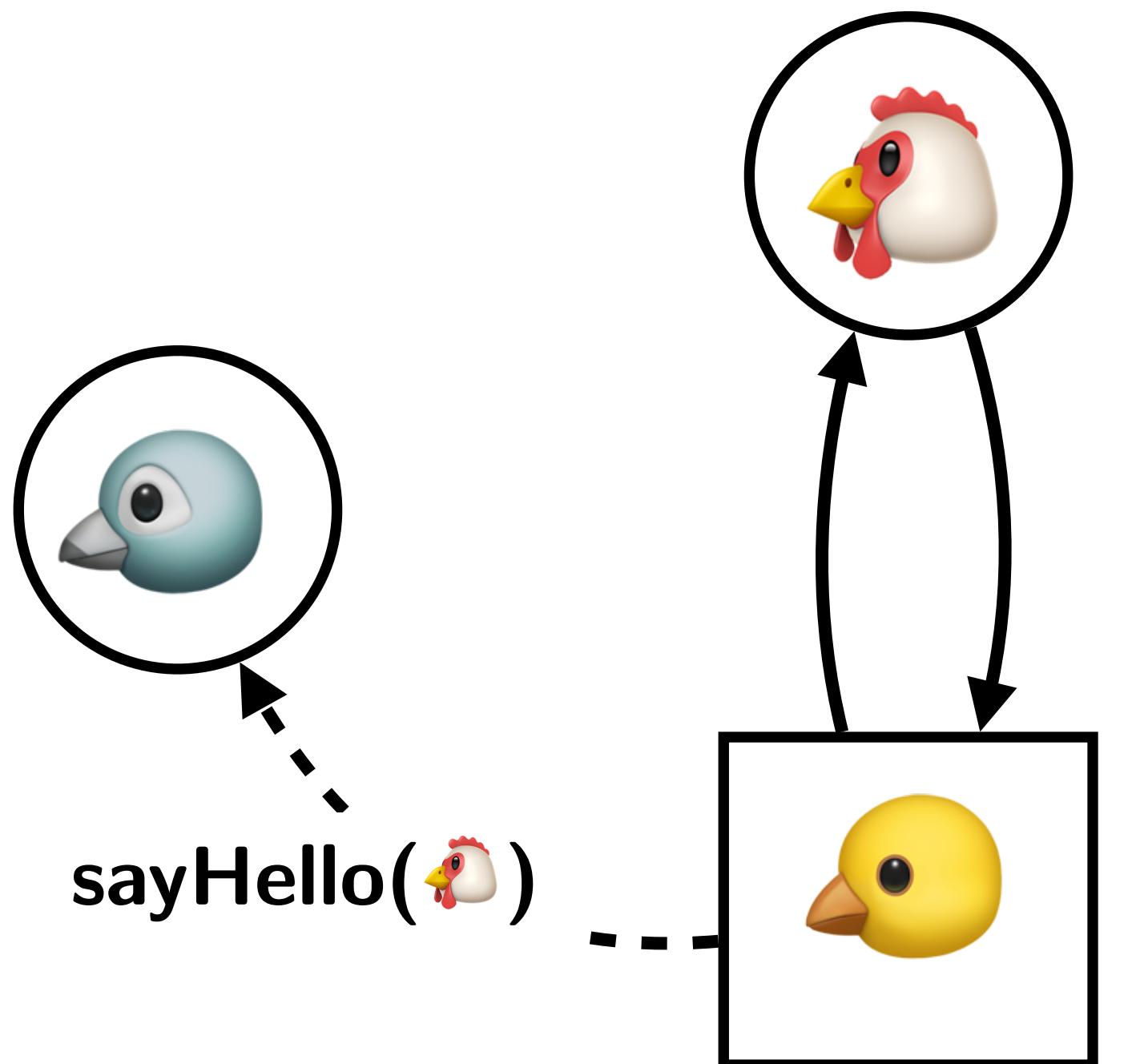
- busy actor
- idle actor
- reference
- message
- healthy node
- crashed node



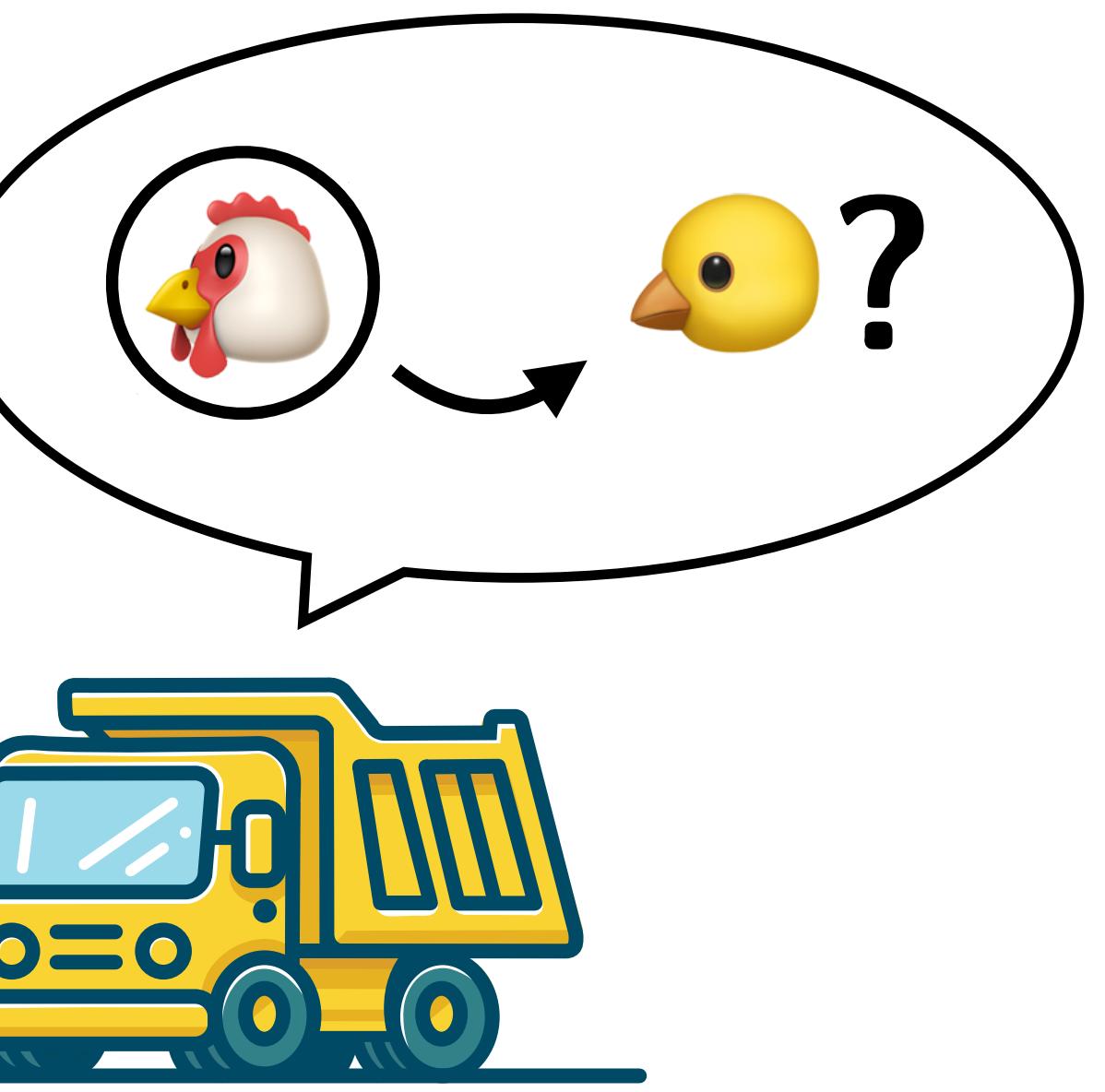
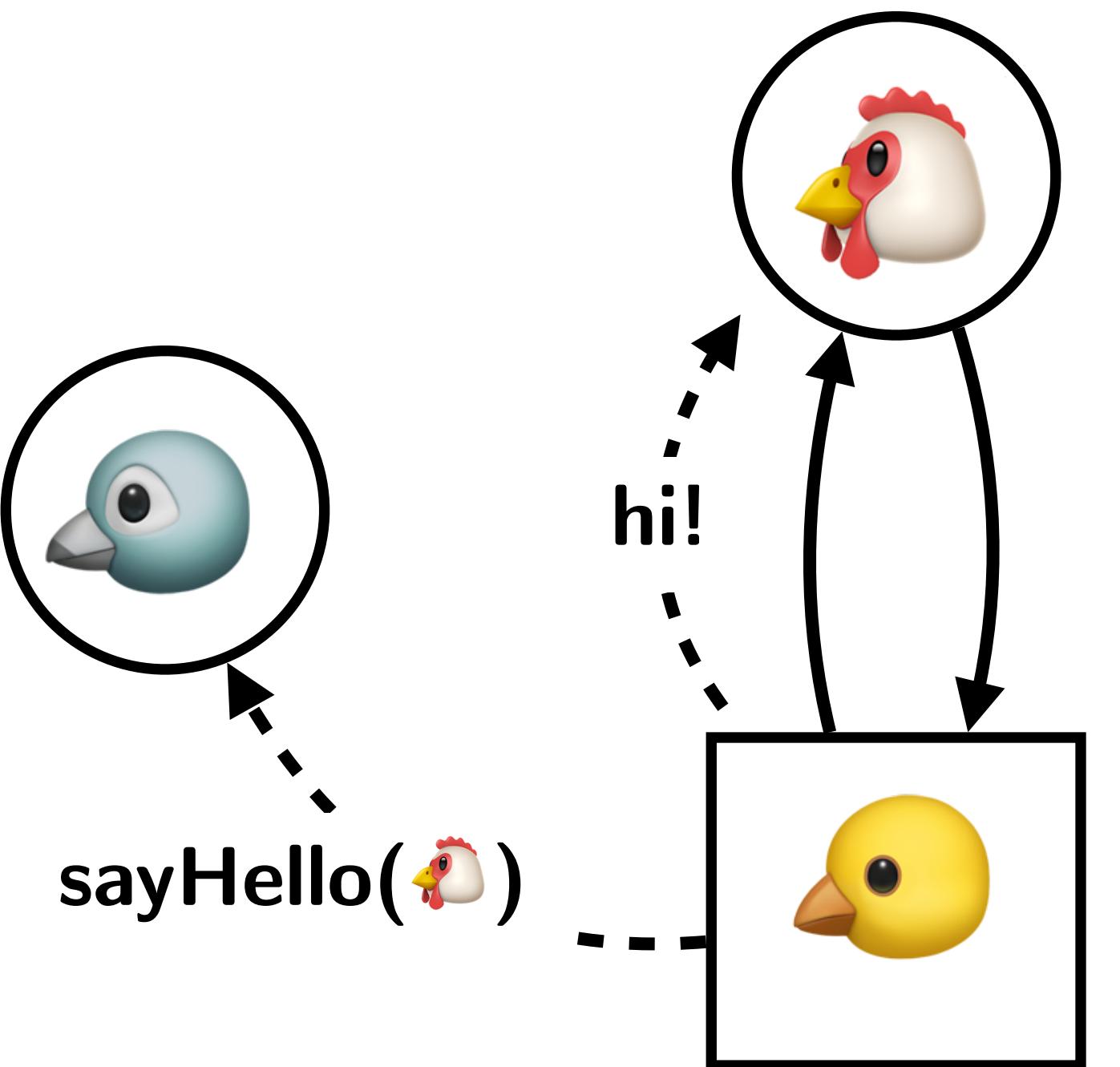
- busy actor
- idle actor
- reference
- message
- healthy node
- crashed node



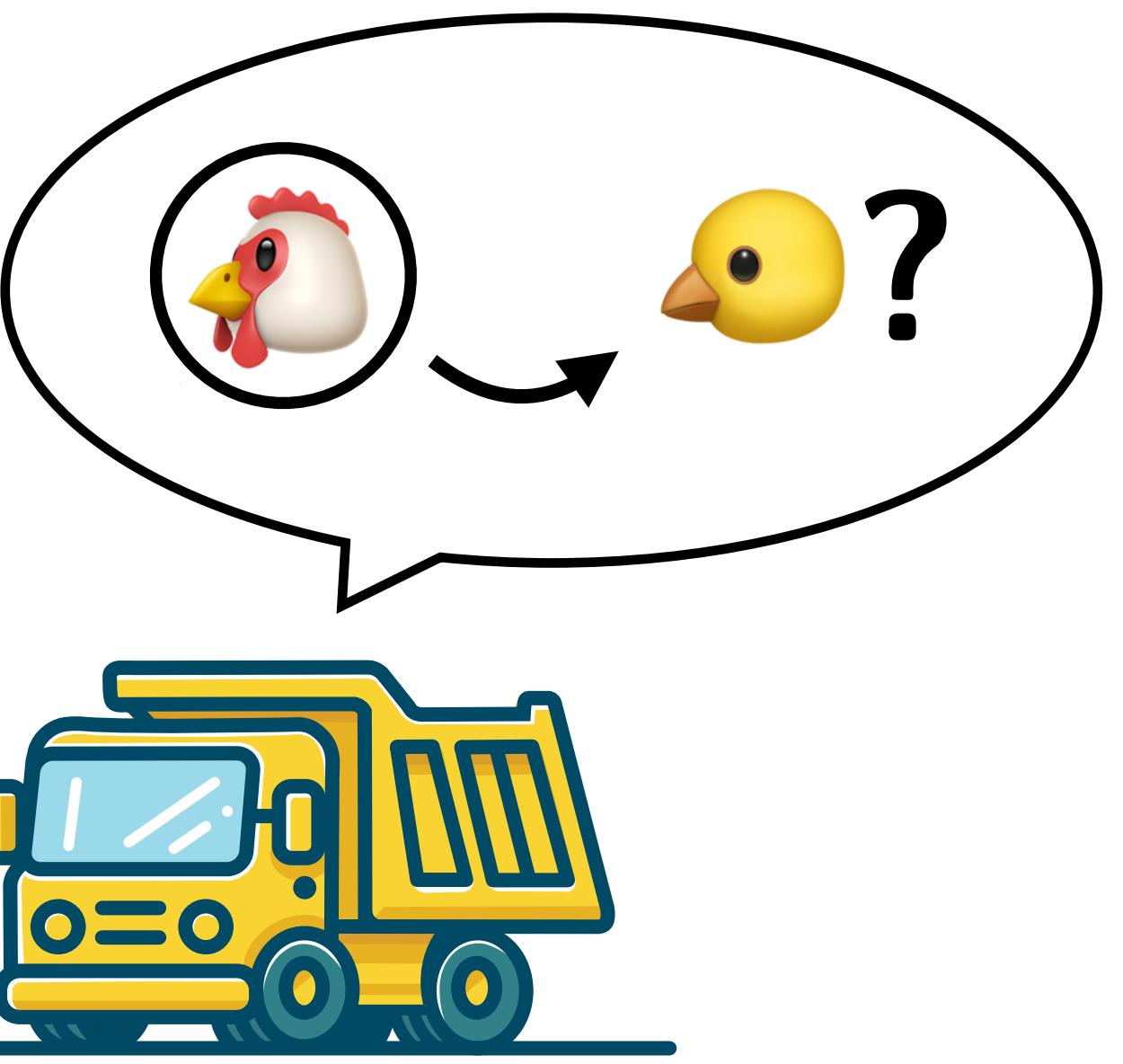
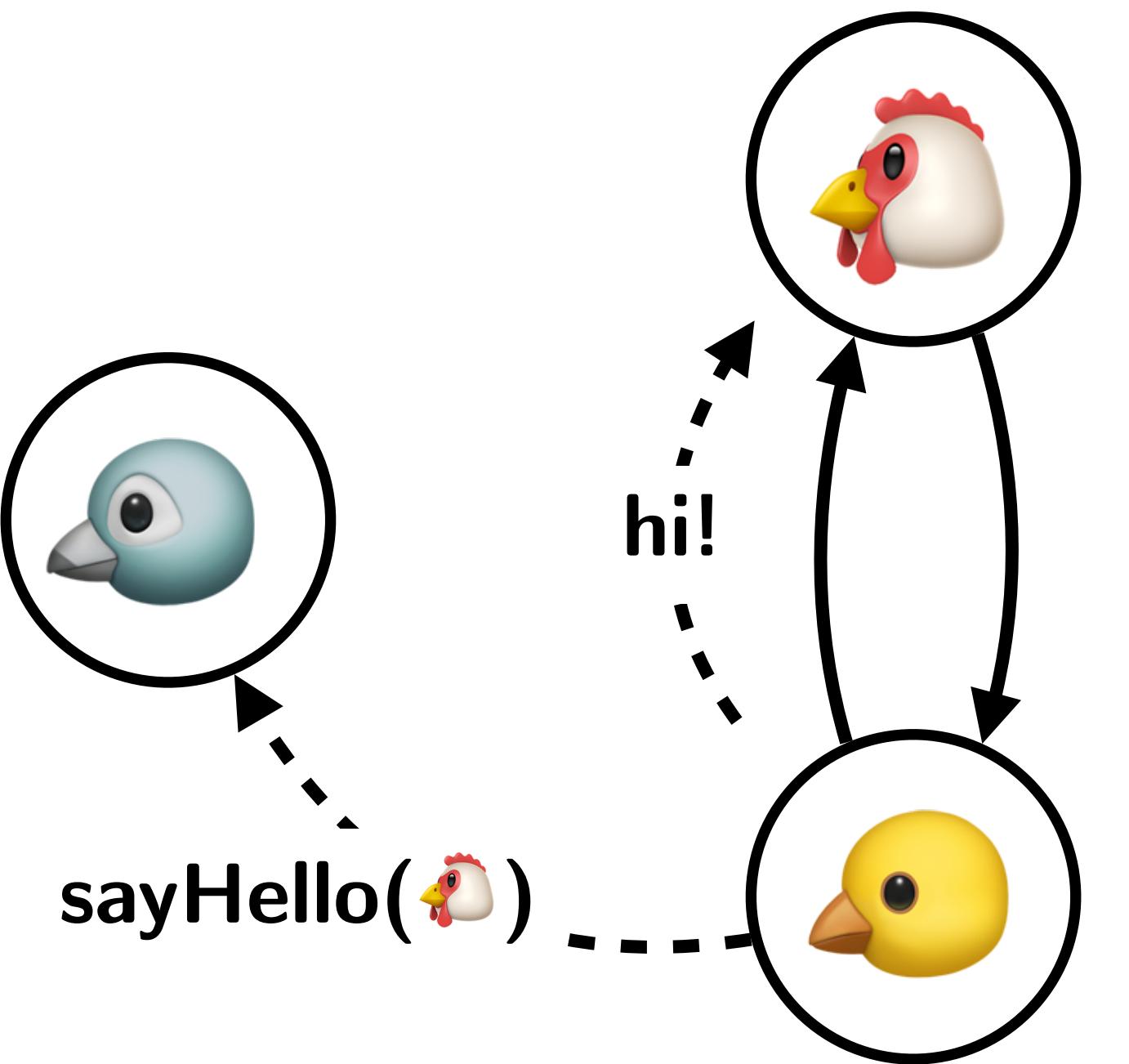
- busy actor
- idle actor
- reference
- message
- healthy node
- crashed node



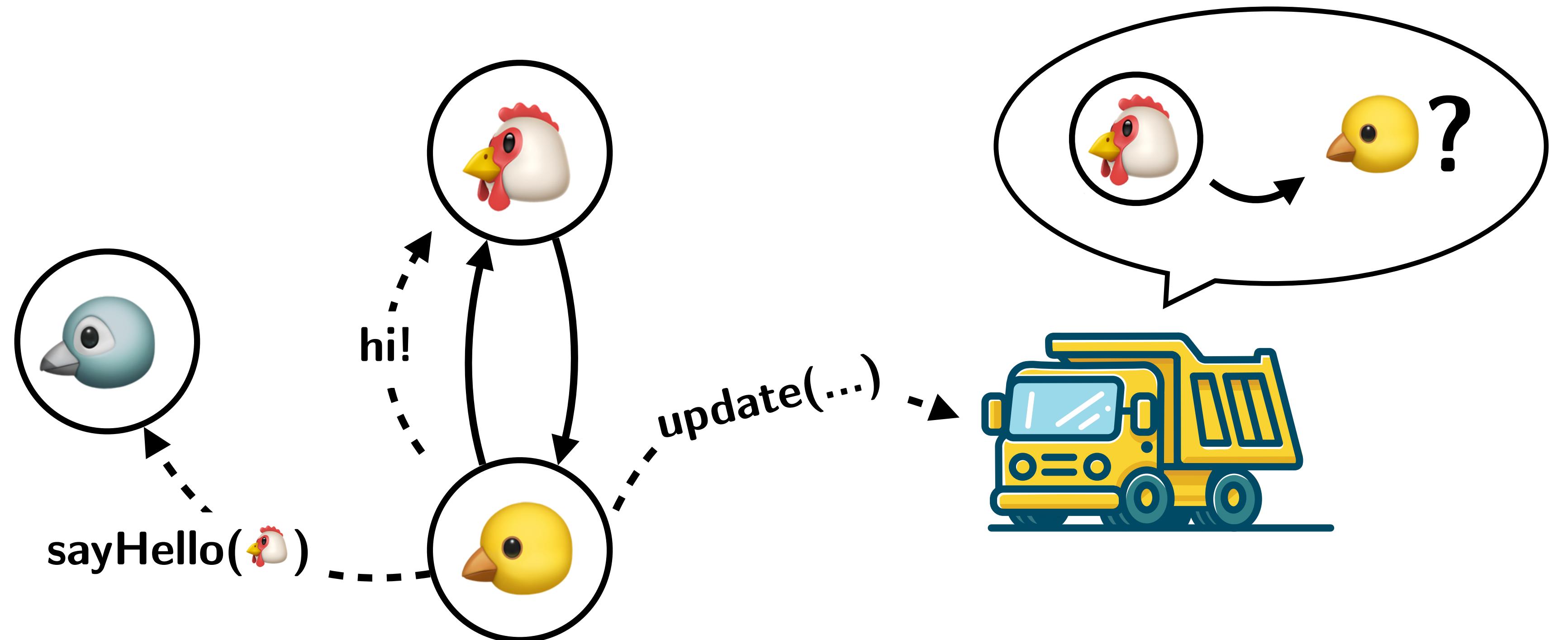
- busy actor
- idle actor
- reference
- message
- healthy node
- crashed node



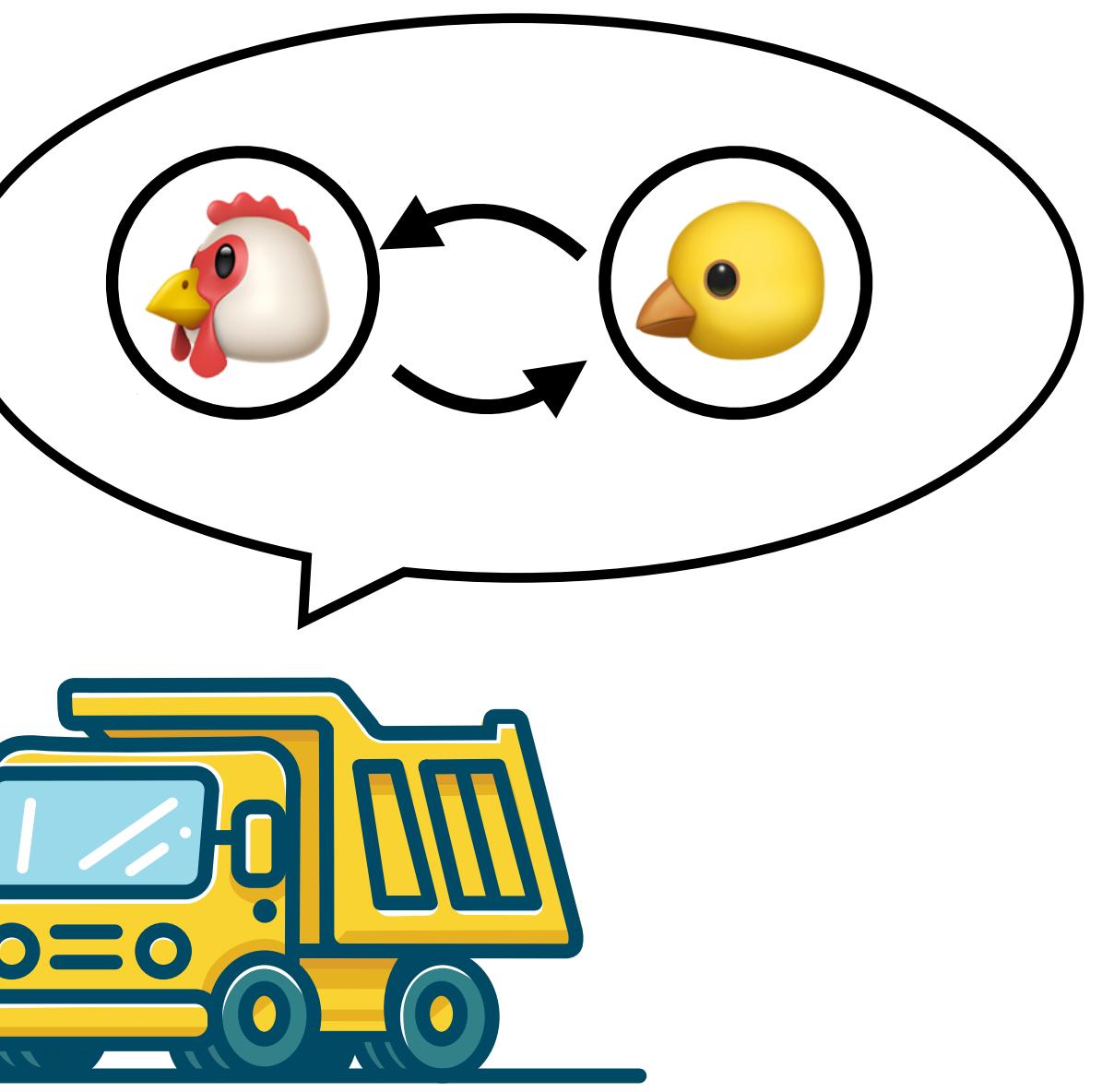
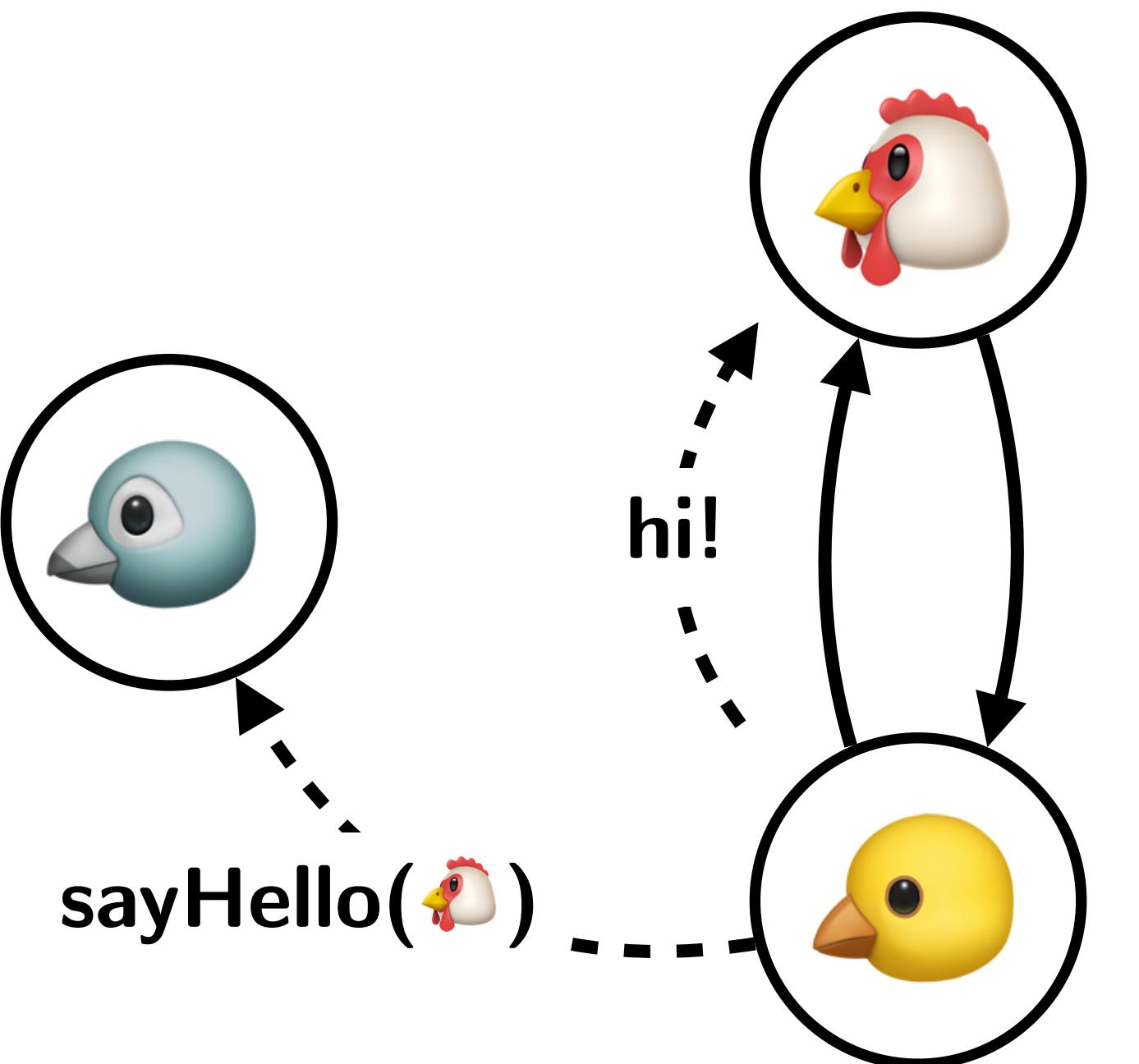
- busy actor
- idle actor
- reference
- message
- healthy node
- crashed node



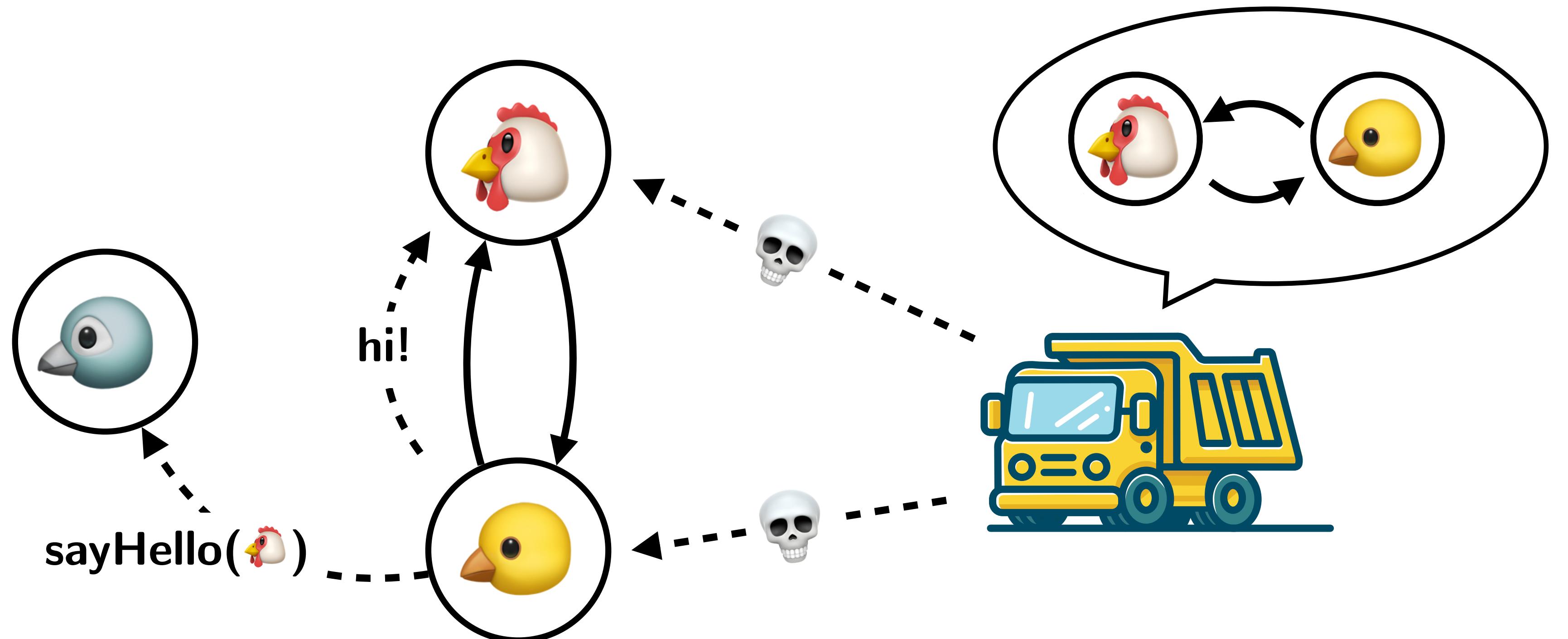
- busy actor
- idle actor
- reference
- message
- healthy node
- crashed node



- busy actor
- idle actor
- reference
- message
- healthy node
- crashed node



- busy actor
- idle actor
- reference
- message
- healthy node
- crashed node



problem 1: consistency requires careful timing

problem 1: consistency requires careful timing

problem 2: slow nodes block progress

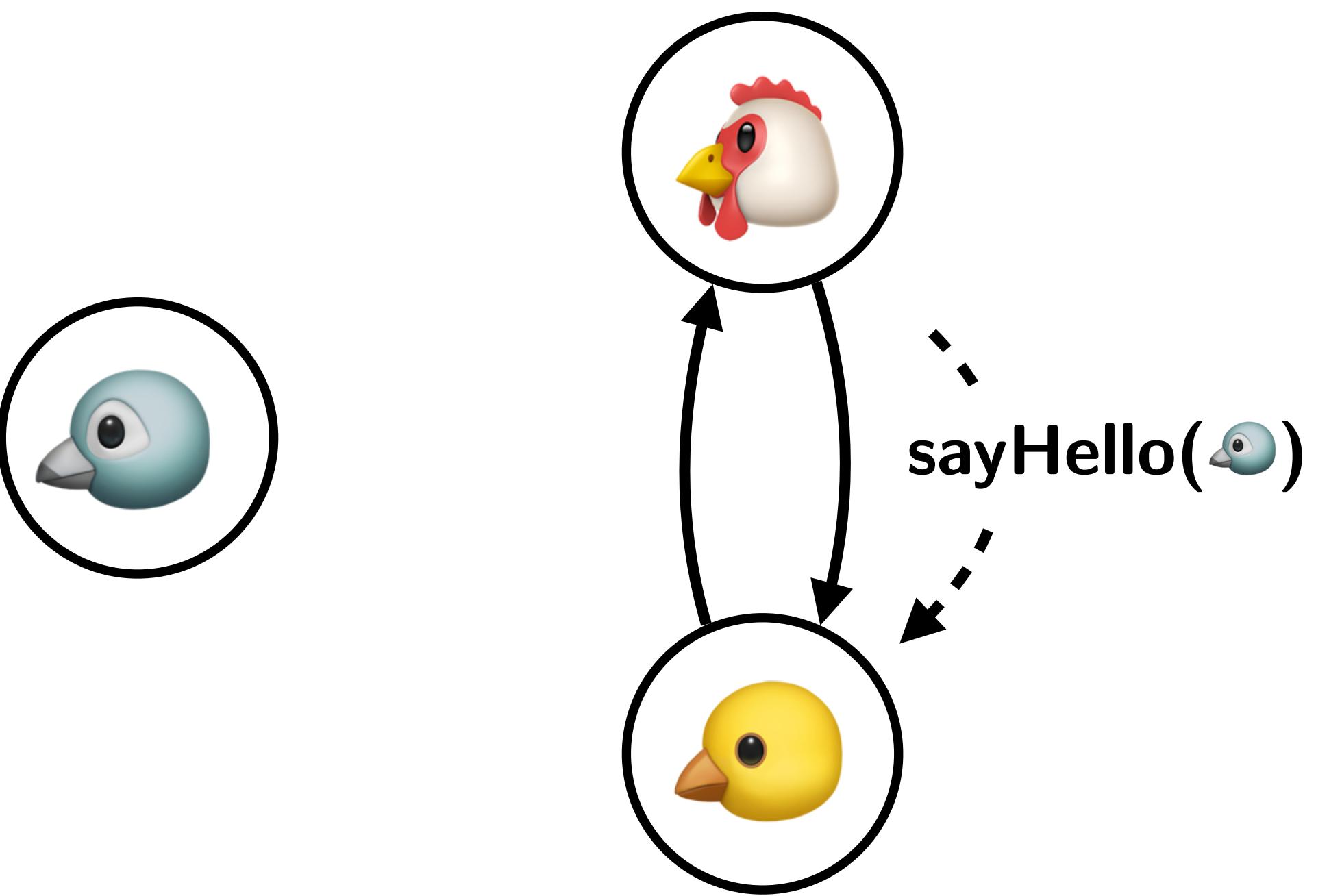
problem 1: consistency requires careful timing

problem 2: slow nodes block progress

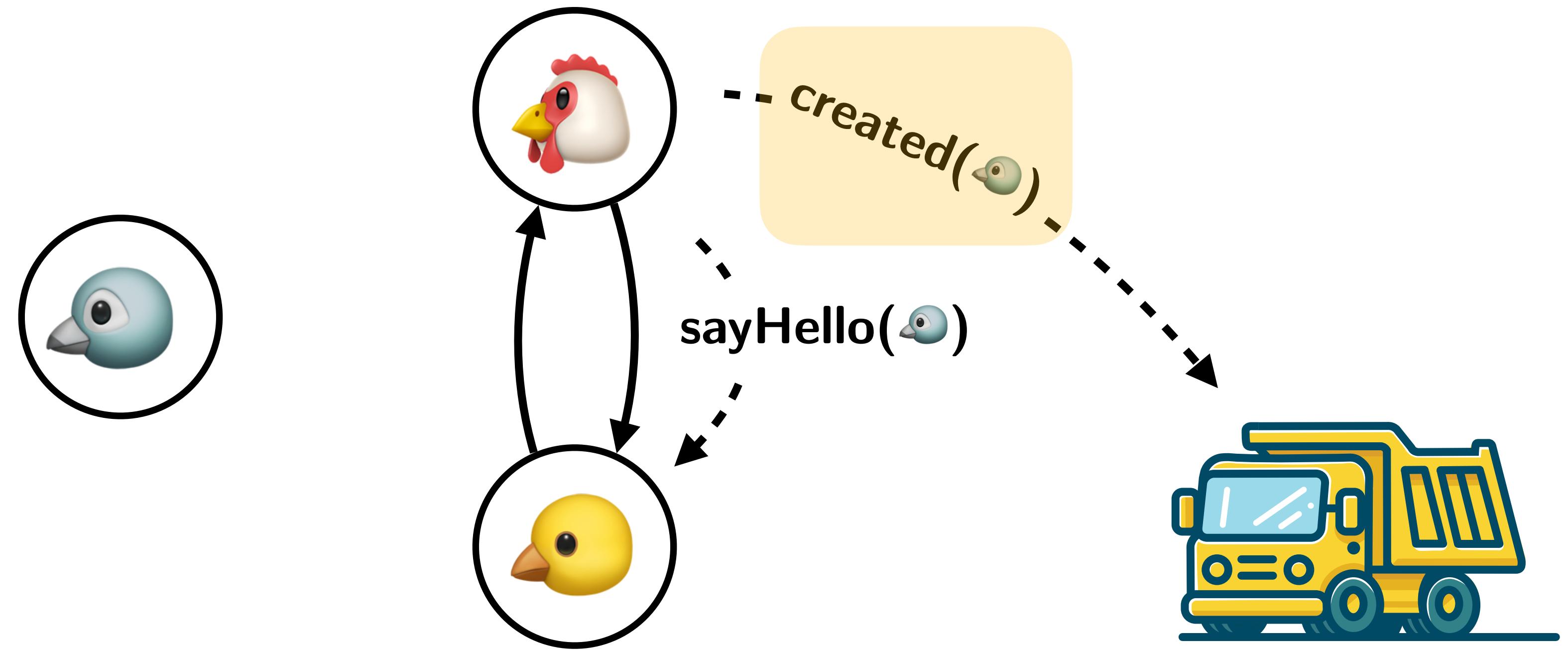
big idea #1

design actor's local state so that
"looking consistent" implies **"being consistent"**

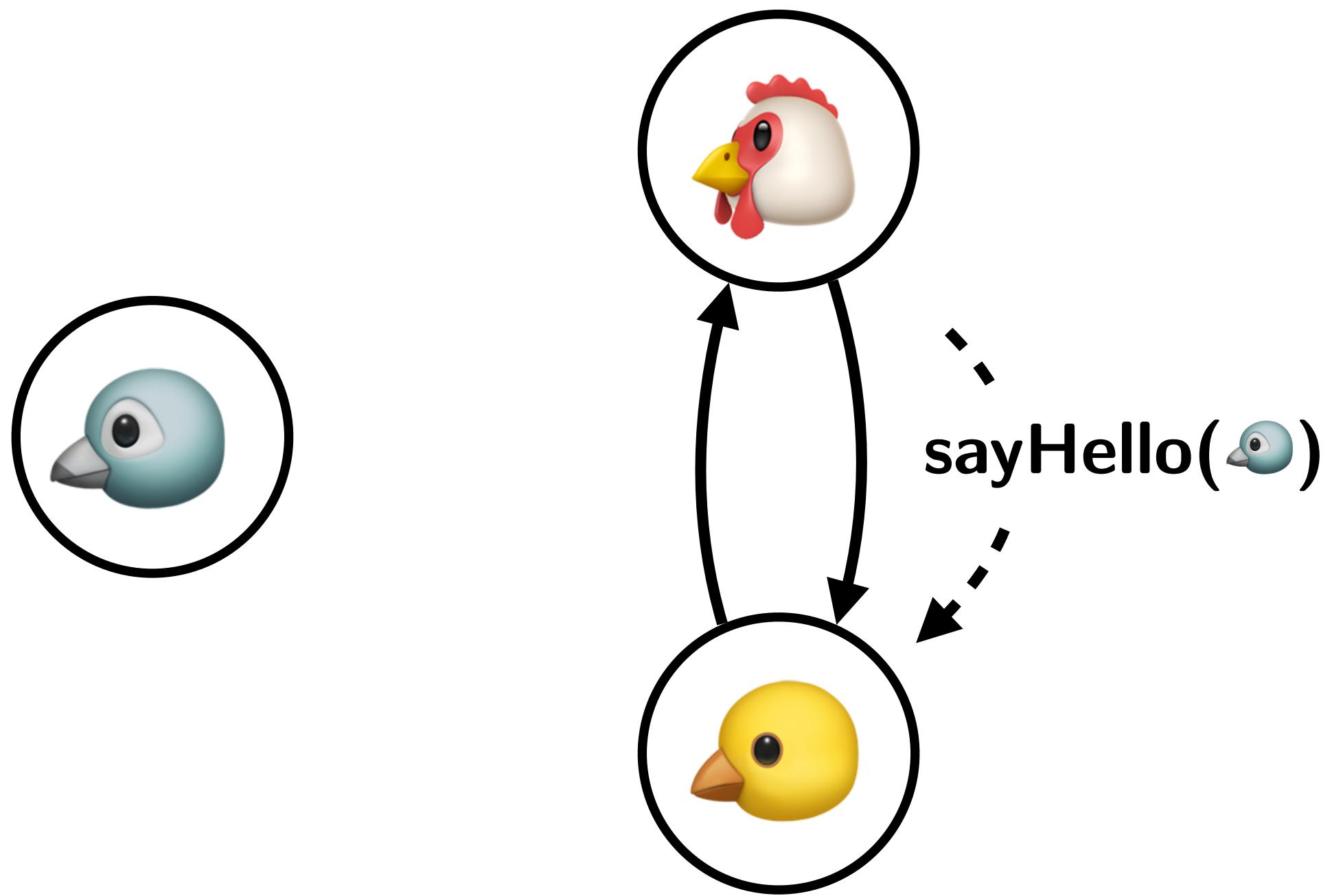
- busy actor
- idle actor
- reference
- message
- healthy node
- crashed node



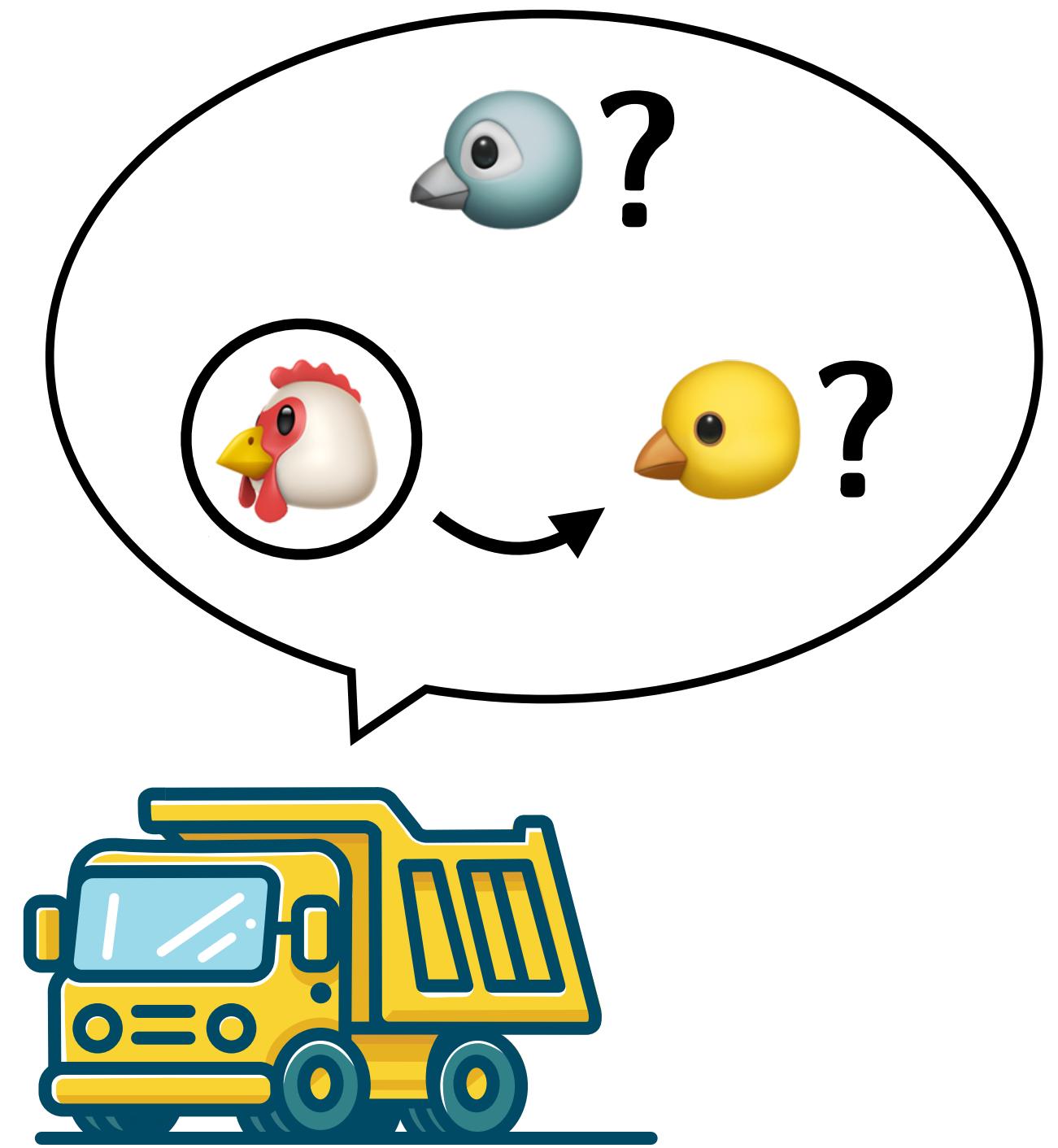
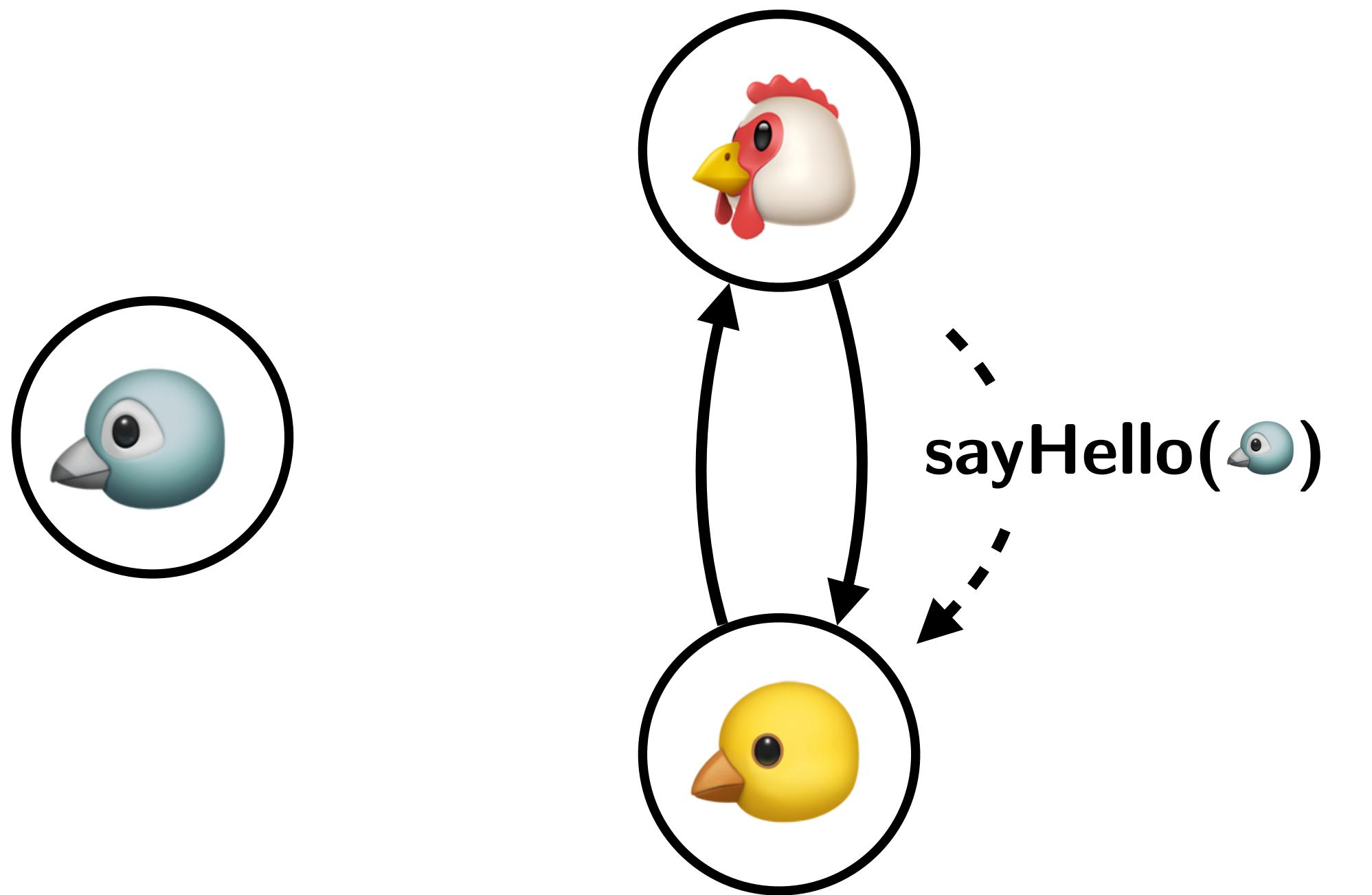
- busy actor
- idle actor
- reference
- message
- healthy node
- crashed node



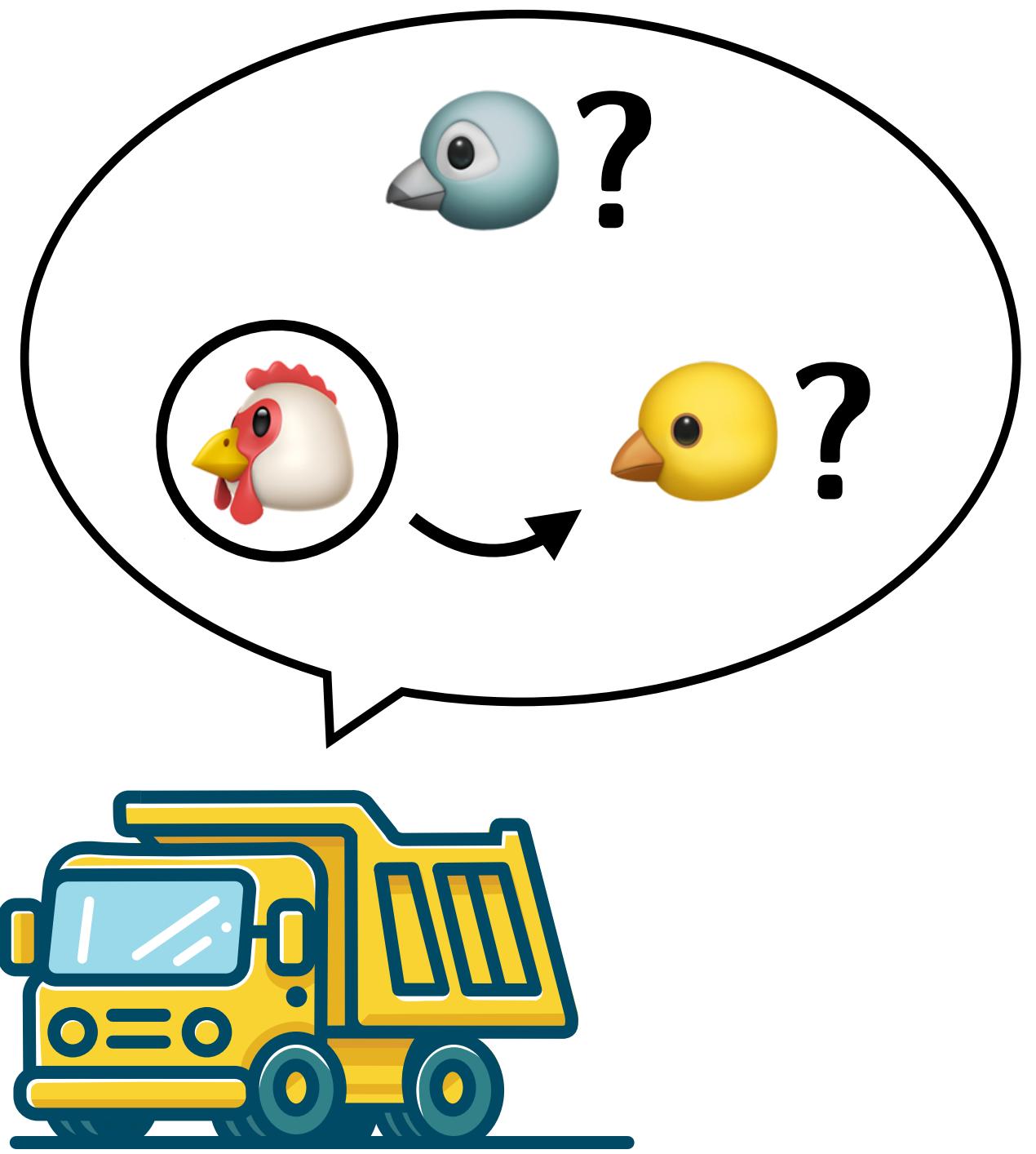
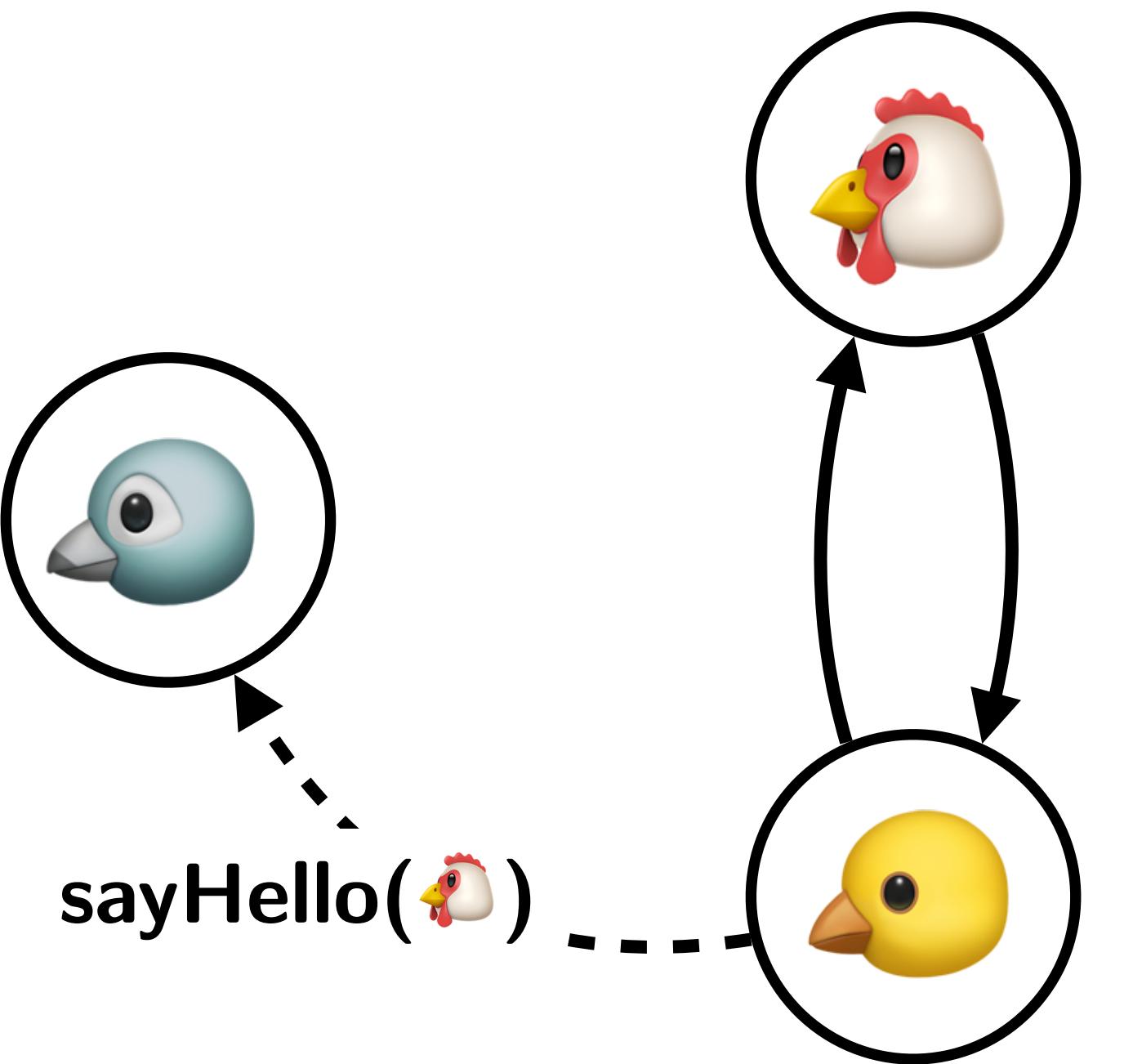
- busy actor
- idle actor
- reference
- message
- healthy node
- crashed node



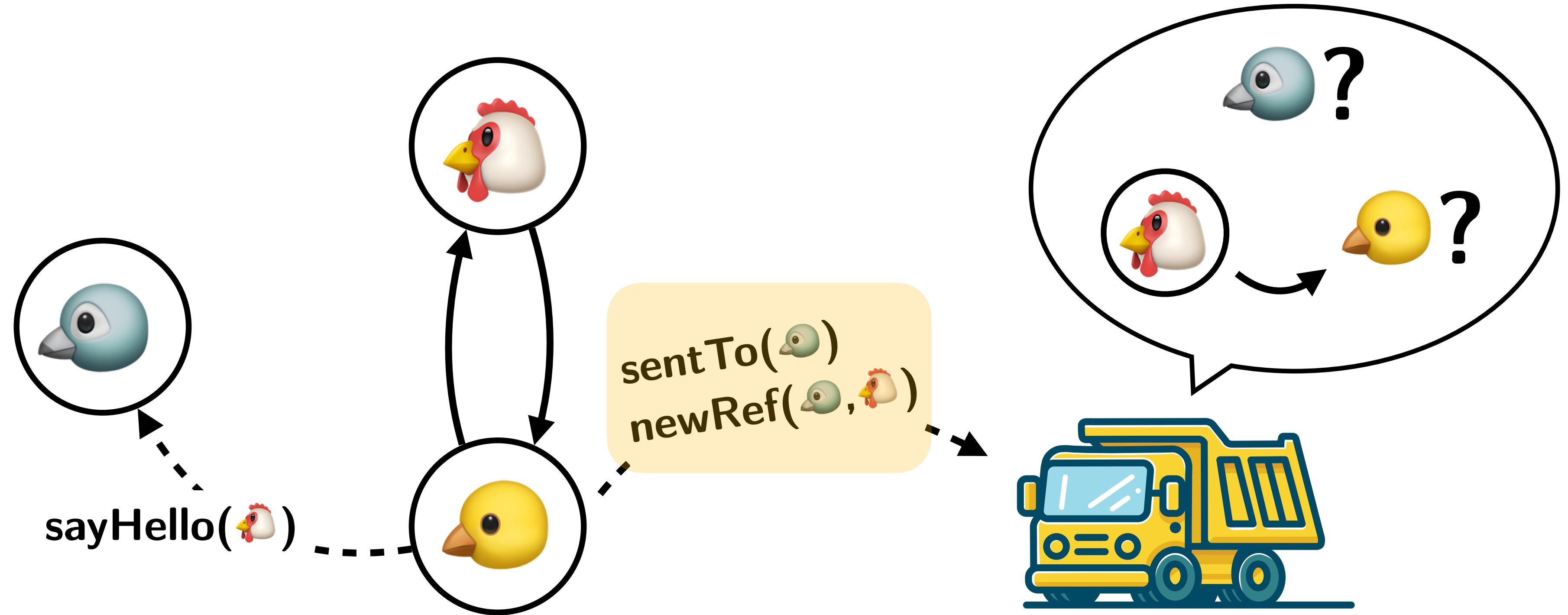
- busy actor
- idle actor
- reference
- message
- healthy node
- crashed node



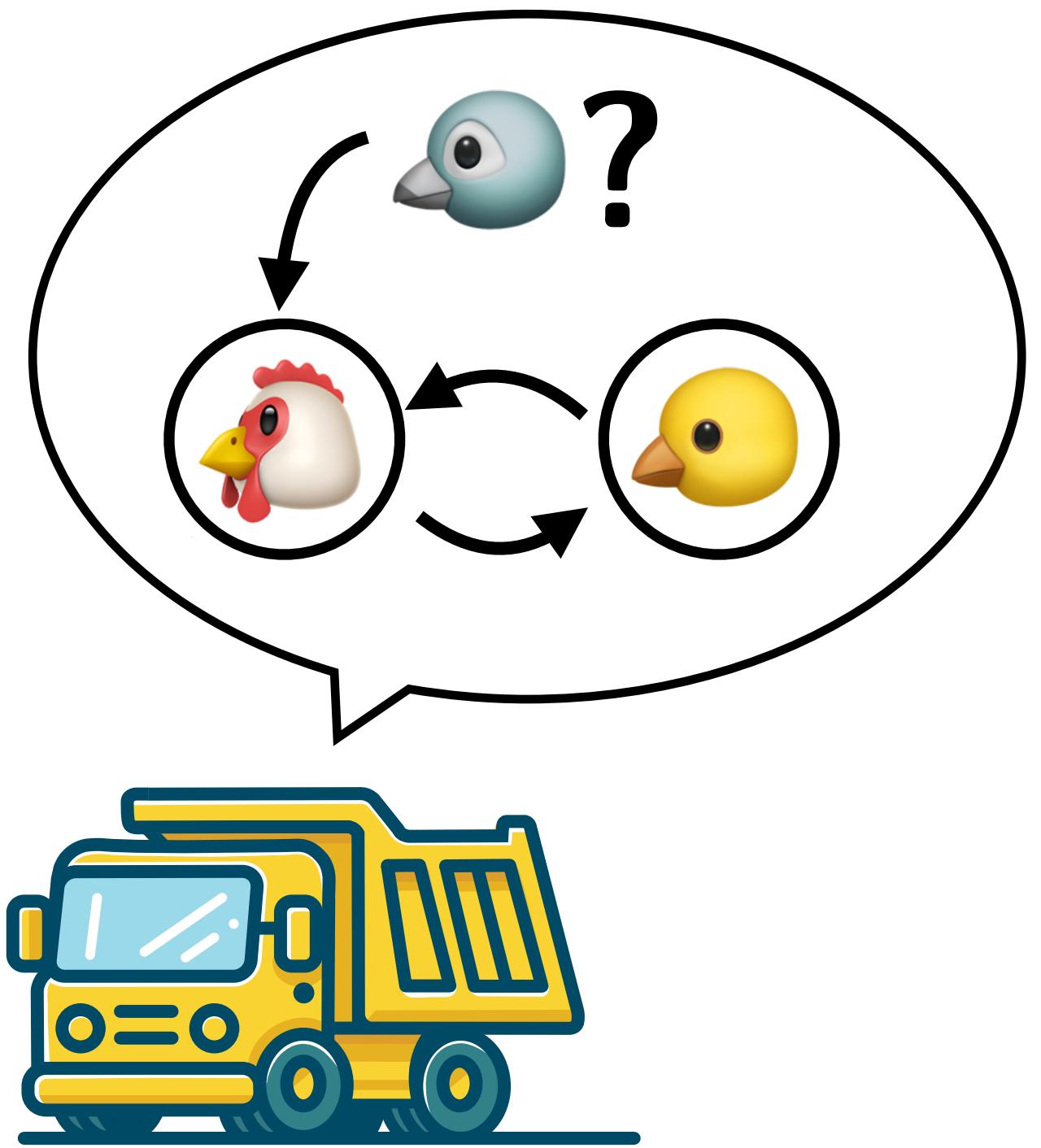
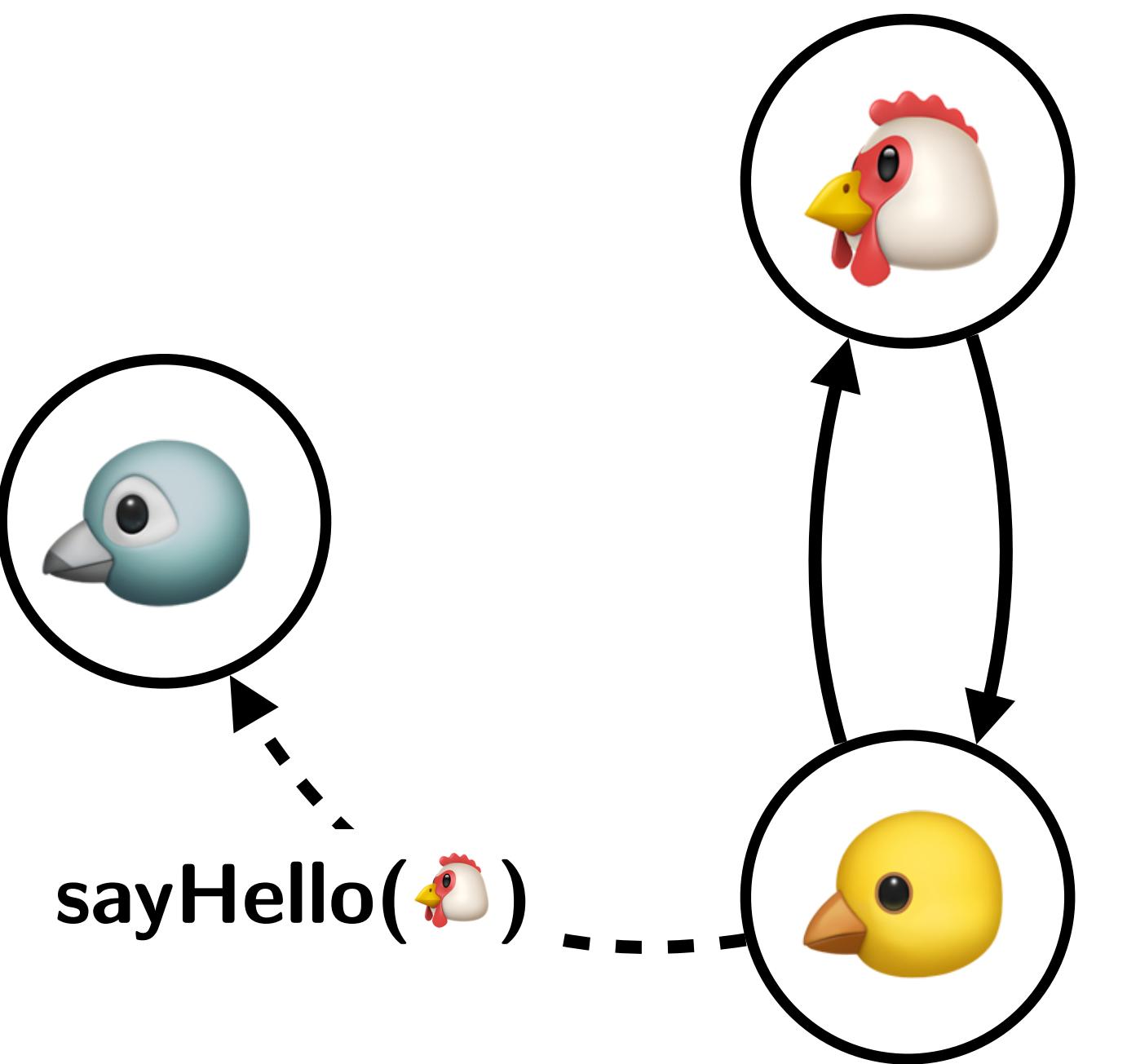
- busy actor
- idle actor
- reference
- message
- healthy node
- crashed node



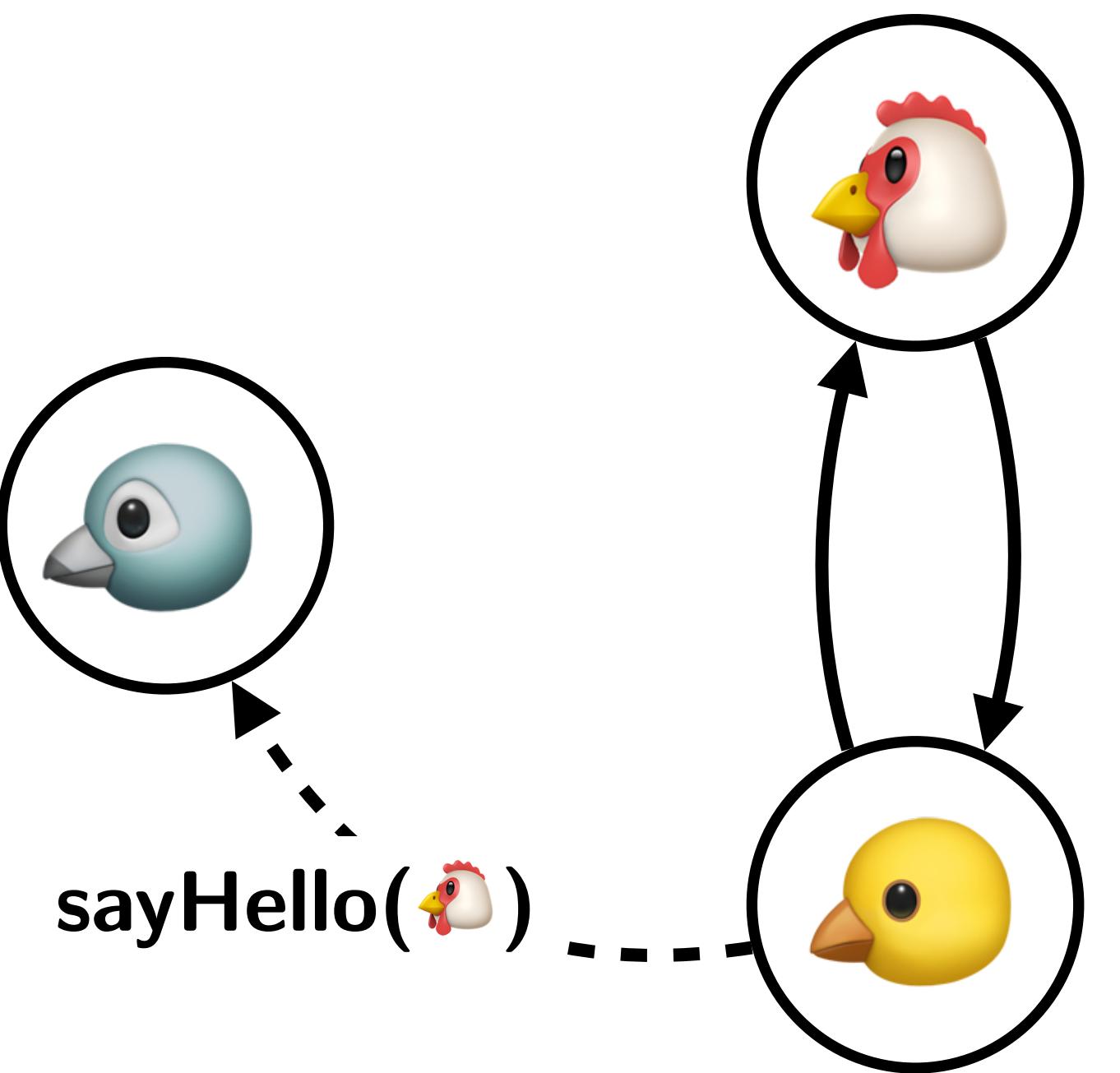
- busy actor
- idle actor
- reference
- message
- healthy node
- crashed node



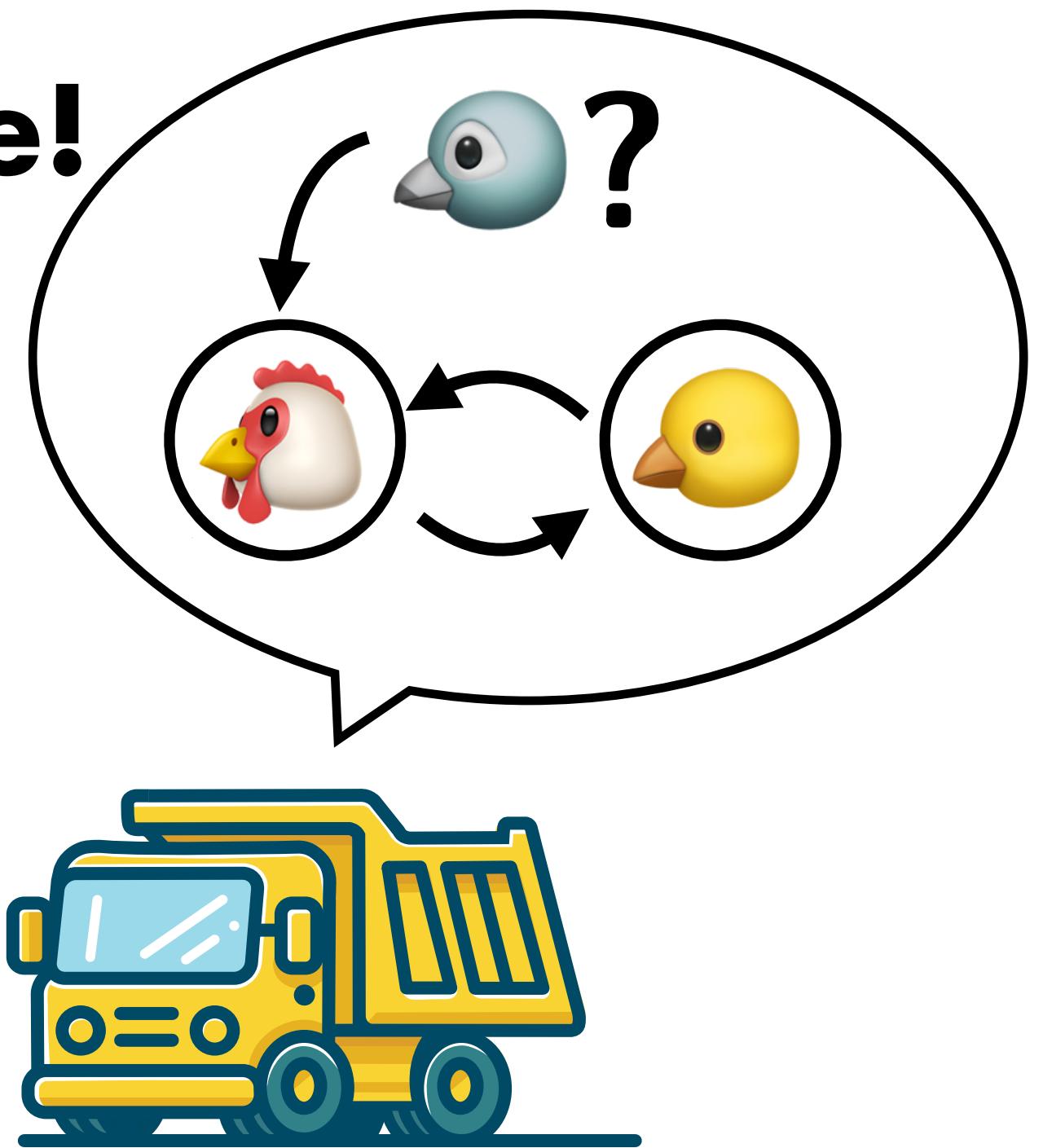
- busy actor
- idle actor
- reference
- message
- healthy node
- crashed node



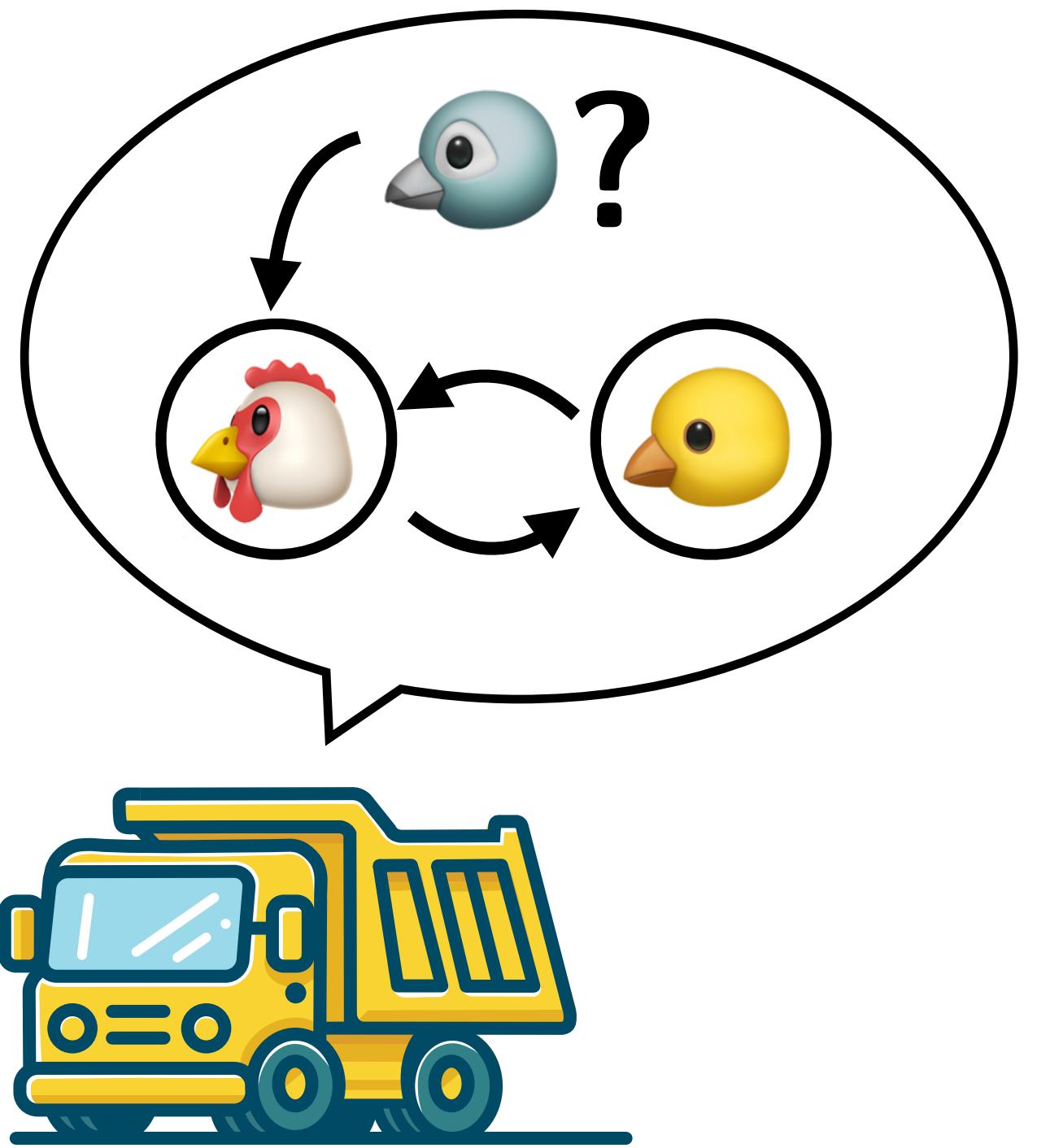
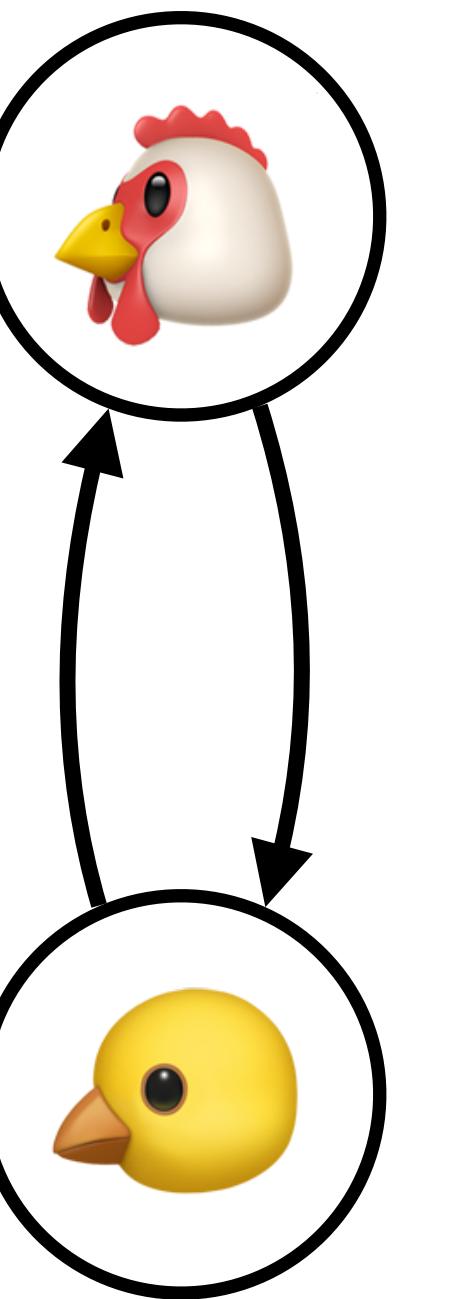
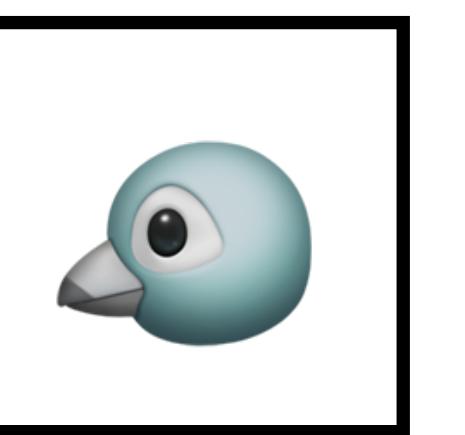
- busy actor
- idle actor
- reference
- message
- healthy node
- crashed node



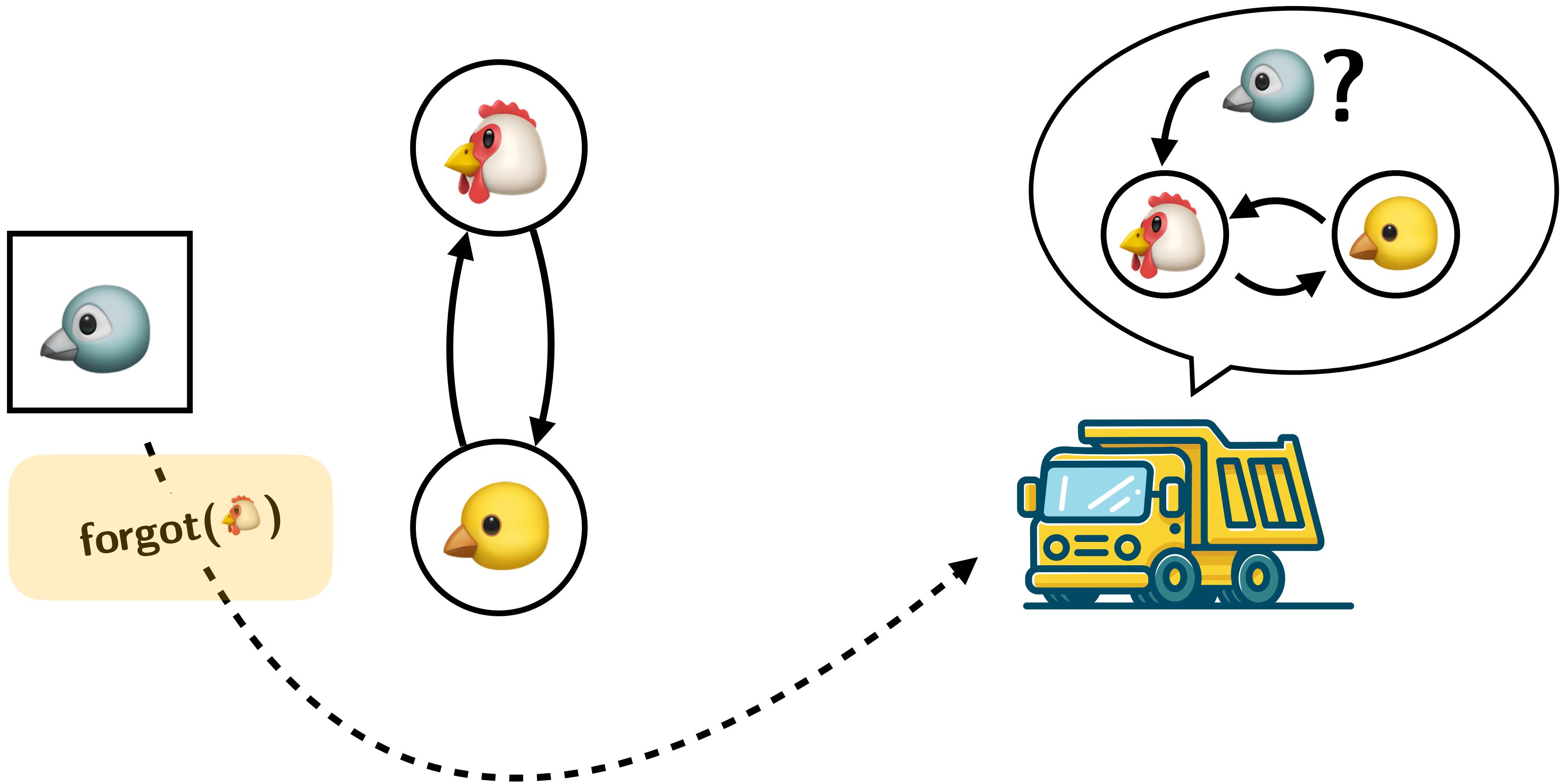
no garbage!



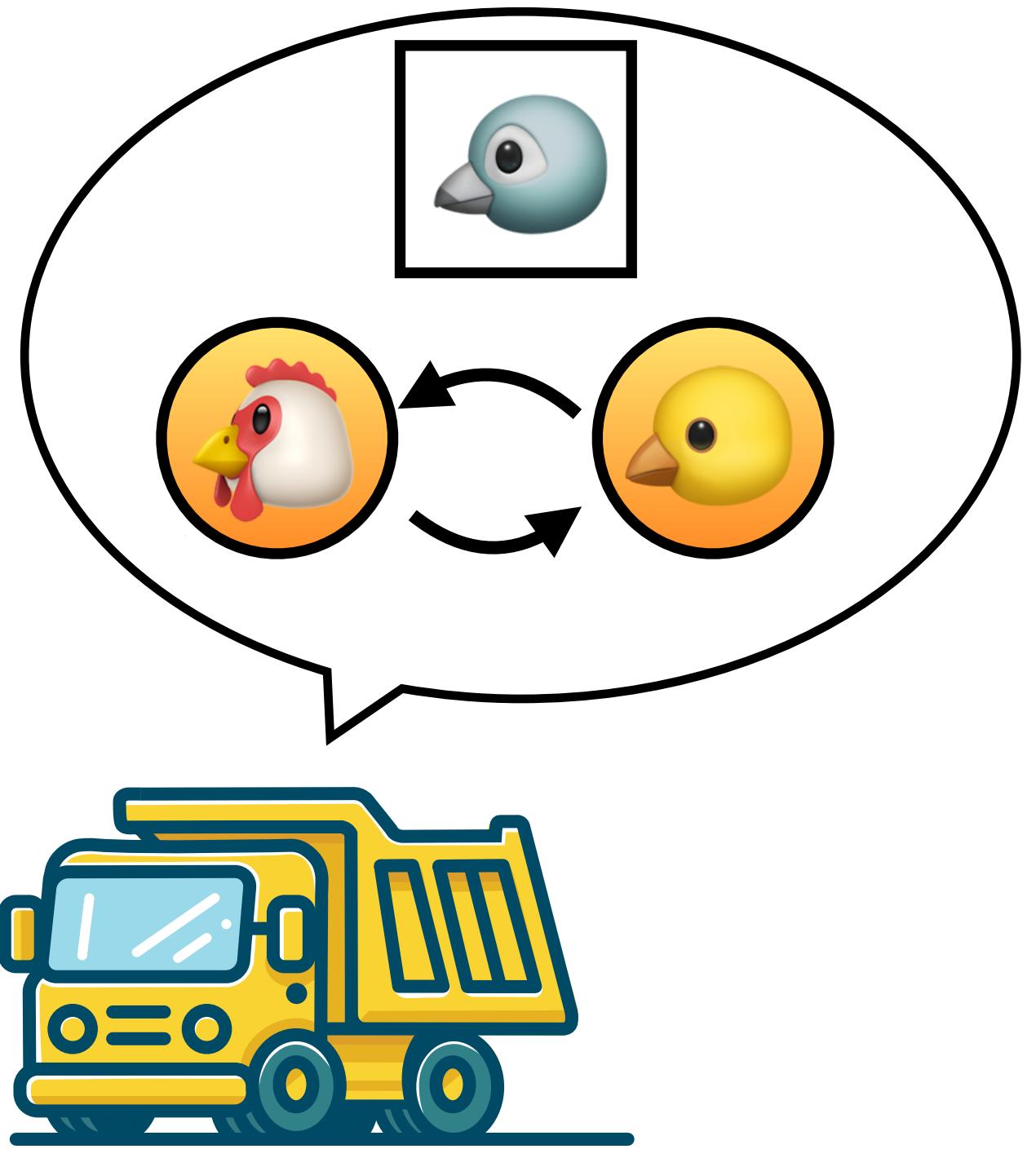
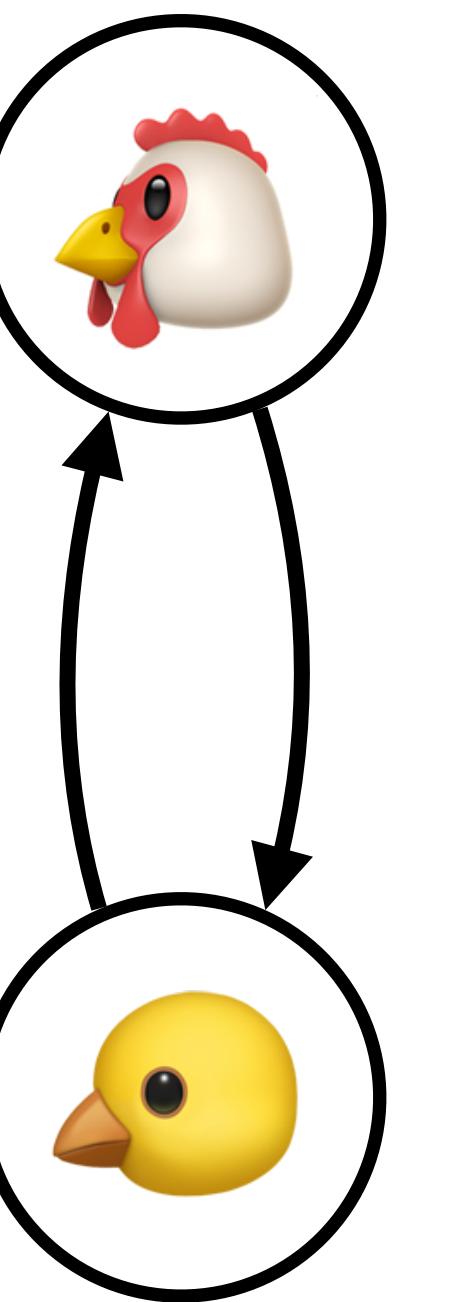
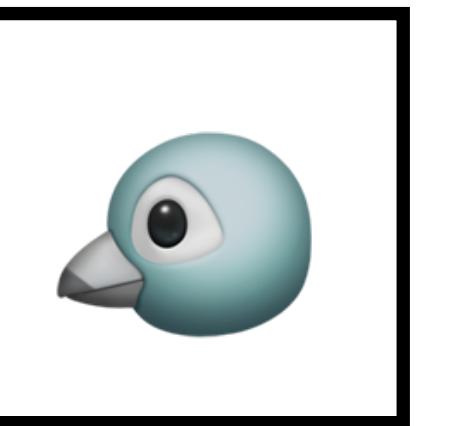
- busy actor
- idle actor
- reference
- message
- healthy node
- crashed node



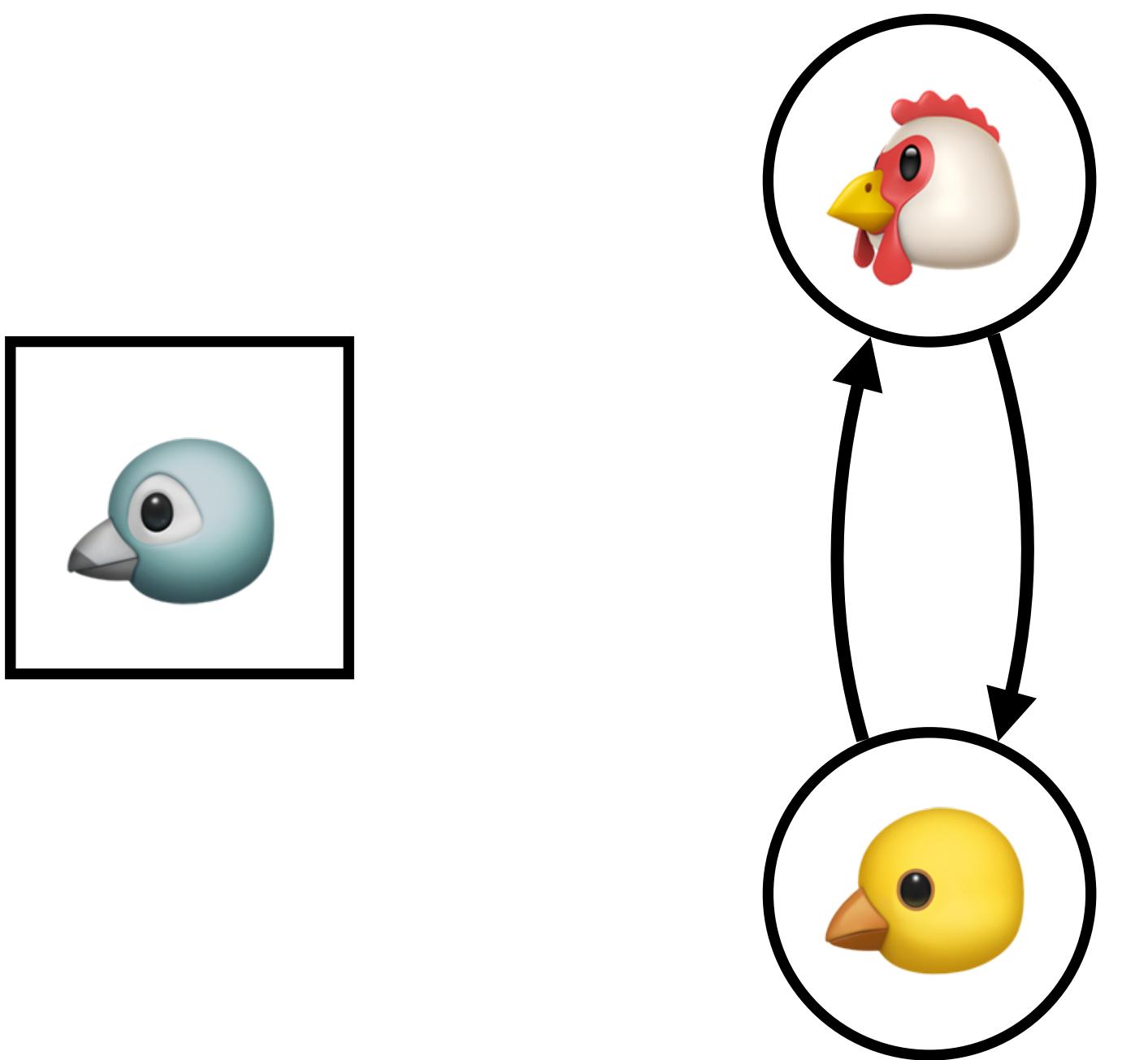
- busy actor
- idle actor
- reference
- message
- healthy node
- crashed node



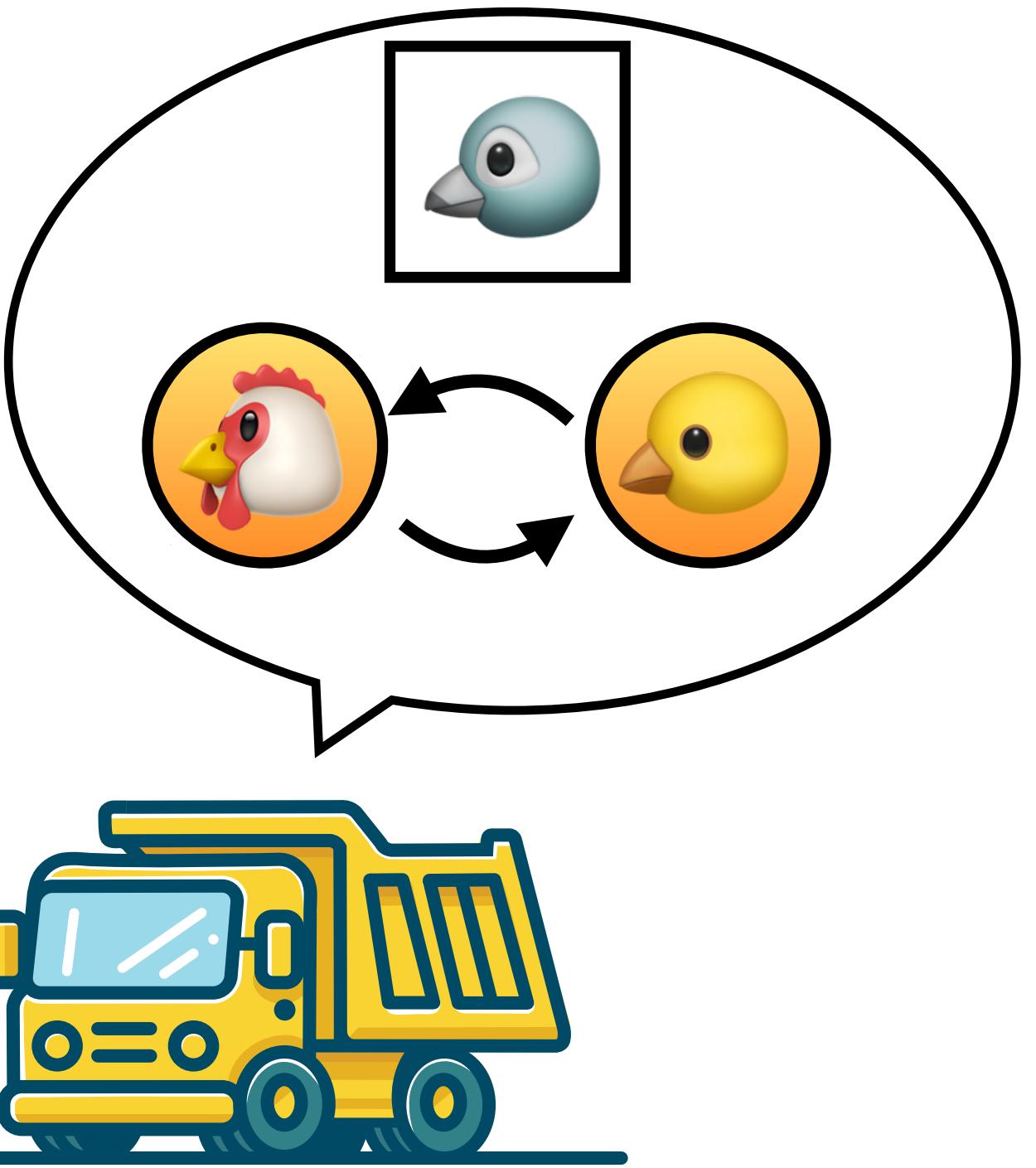
- busy actor
- idle actor
- reference
- message
- healthy node
- crashed node



- busy actor
- idle actor
- reference
- message
- healthy node
- crashed node

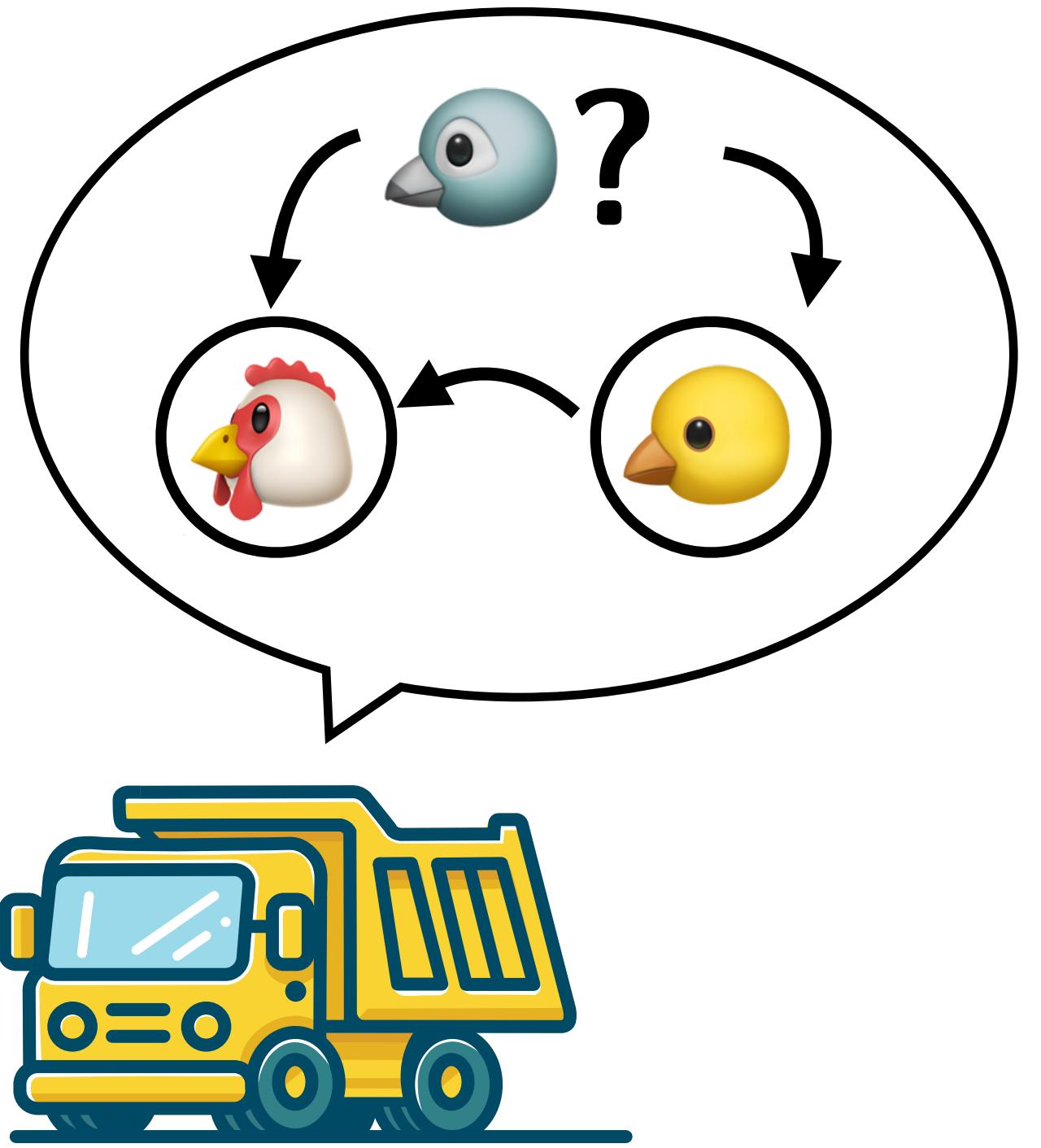
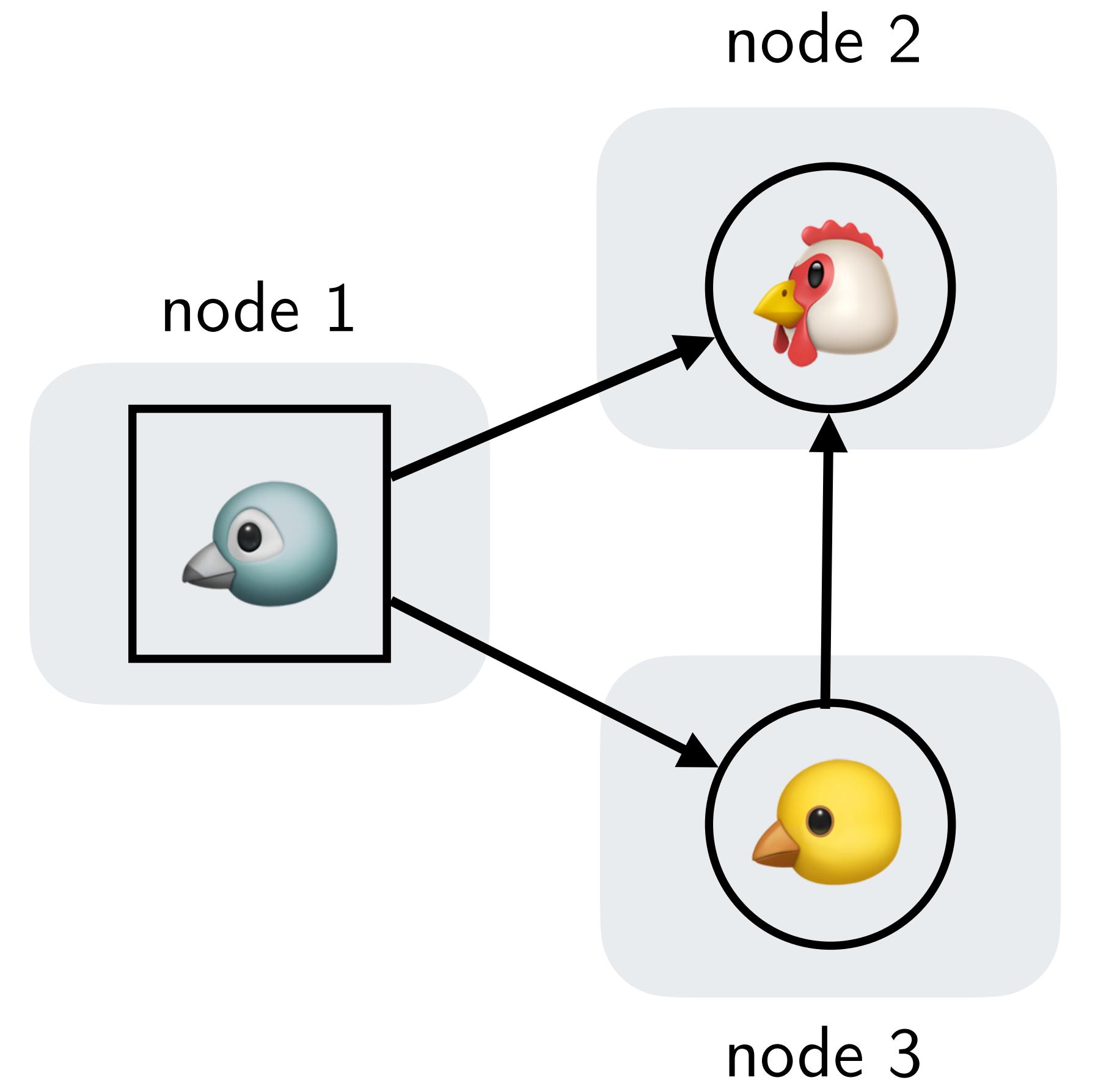


garbage!

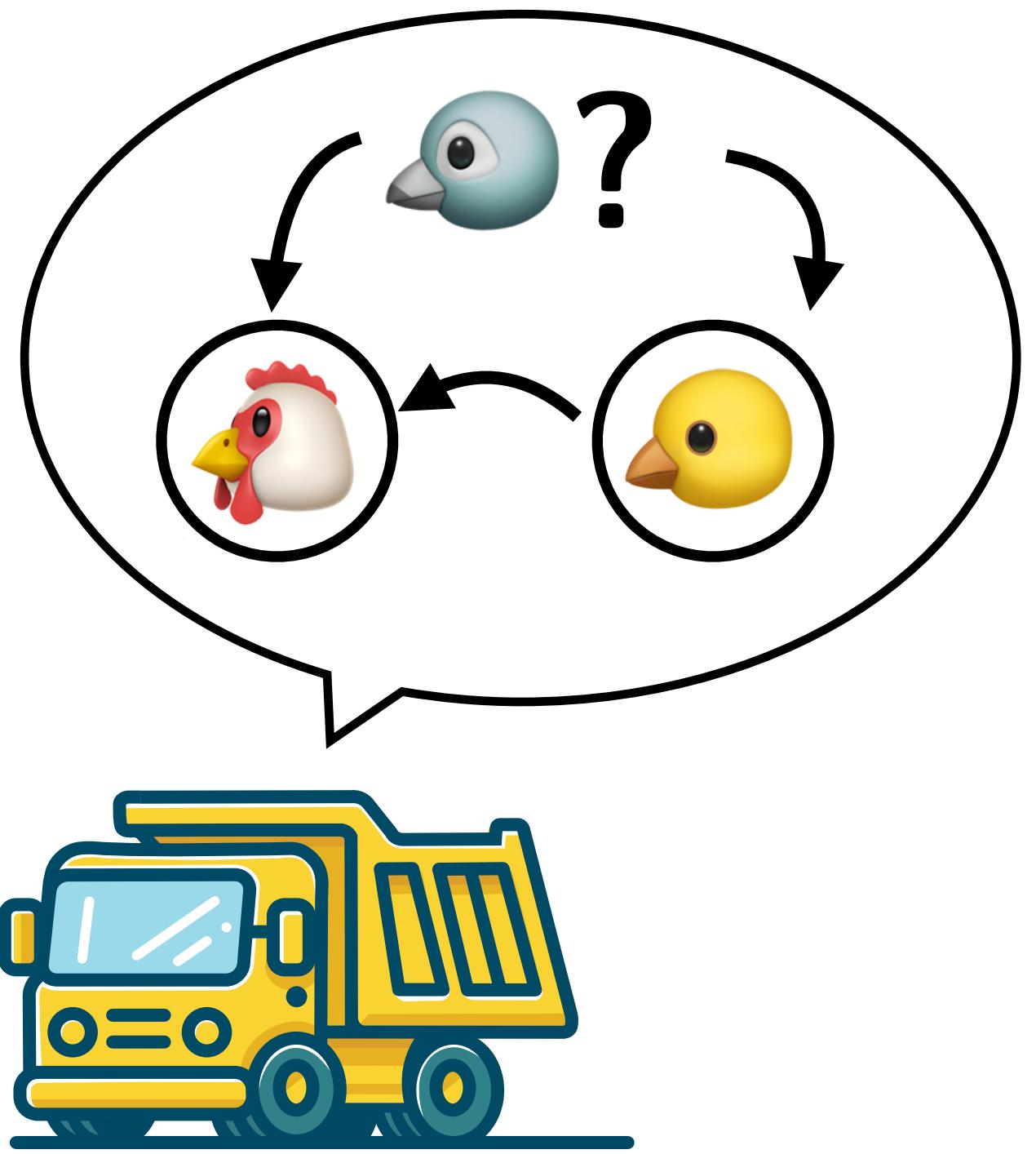
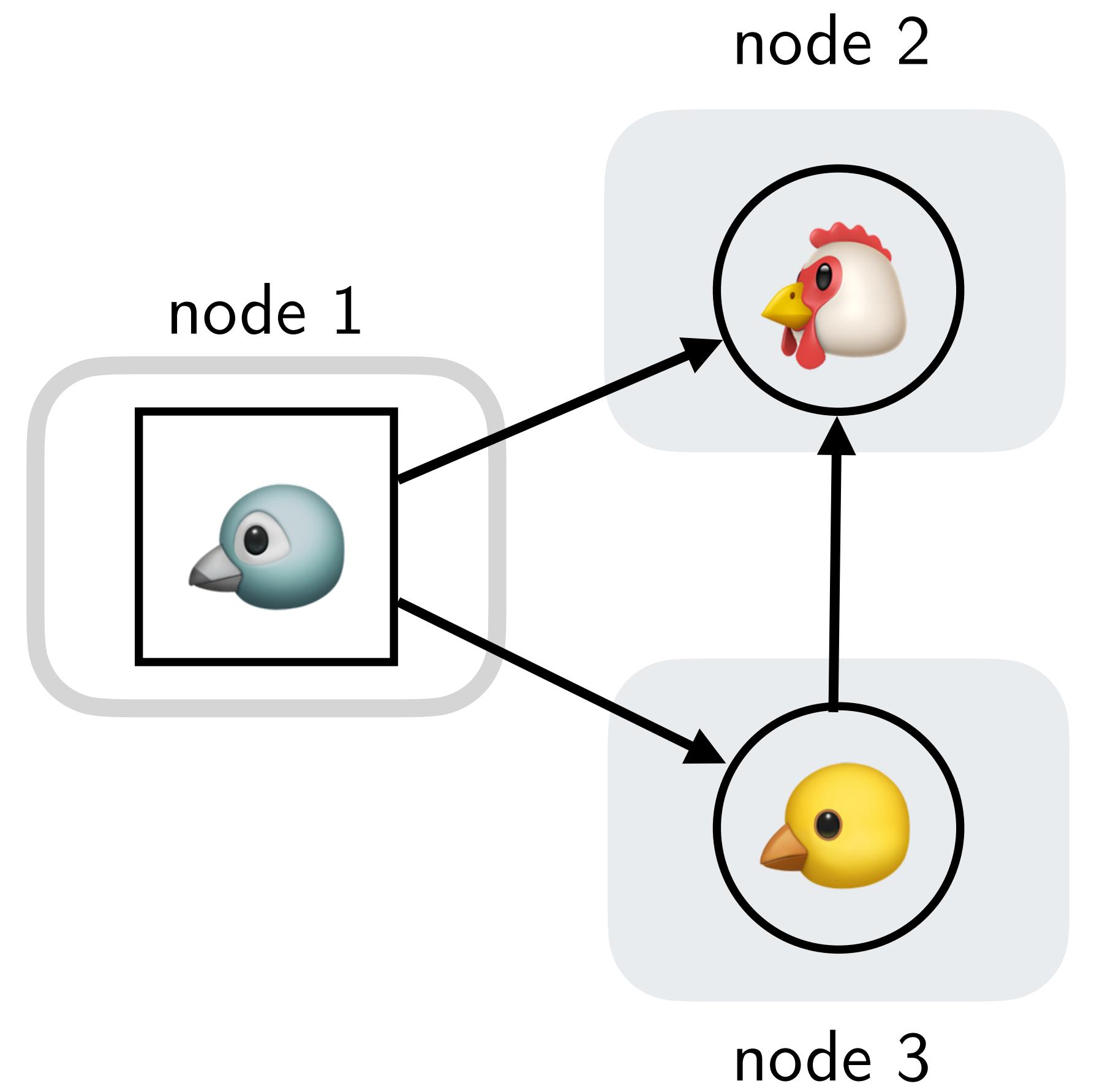


Challenge 2: Crashes

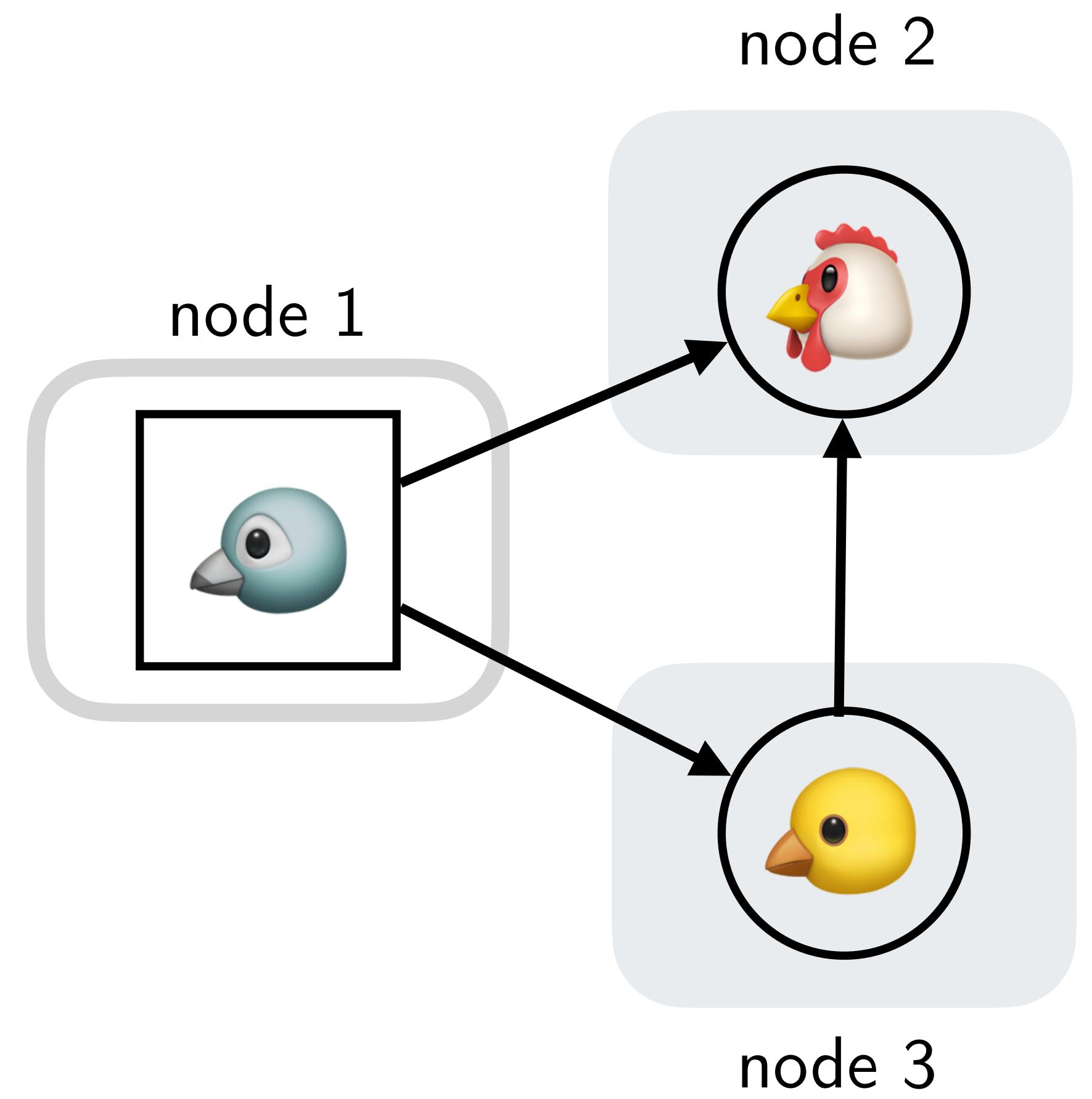
- busy actor
- idle actor
- reference
- message
- healthy node
- crashed node



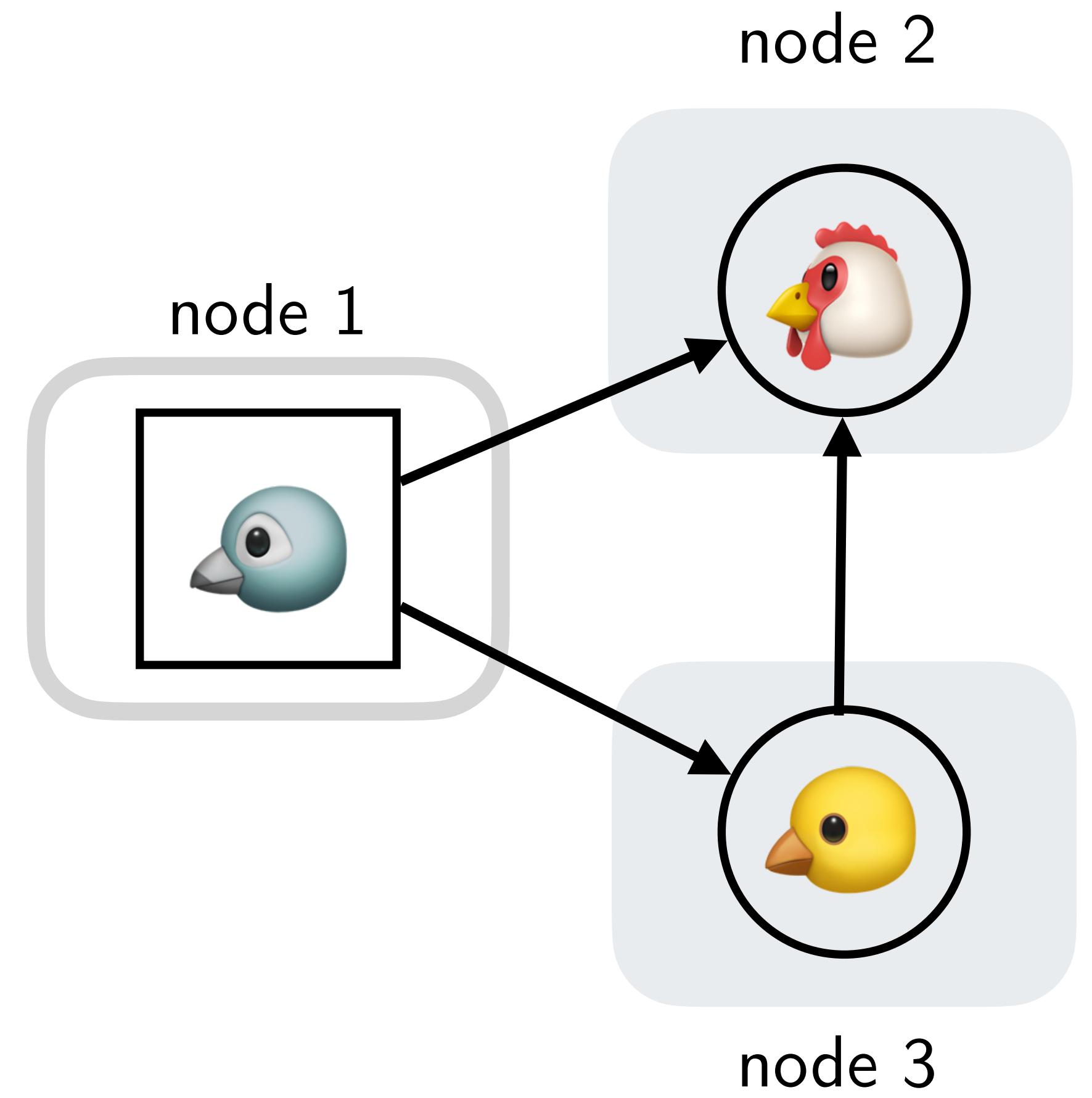
- busy actor
- idle actor
- reference
- message
- healthy node
- crashed node



- busy actor
- idle actor
- reference
- message
- healthy node
- crashed node

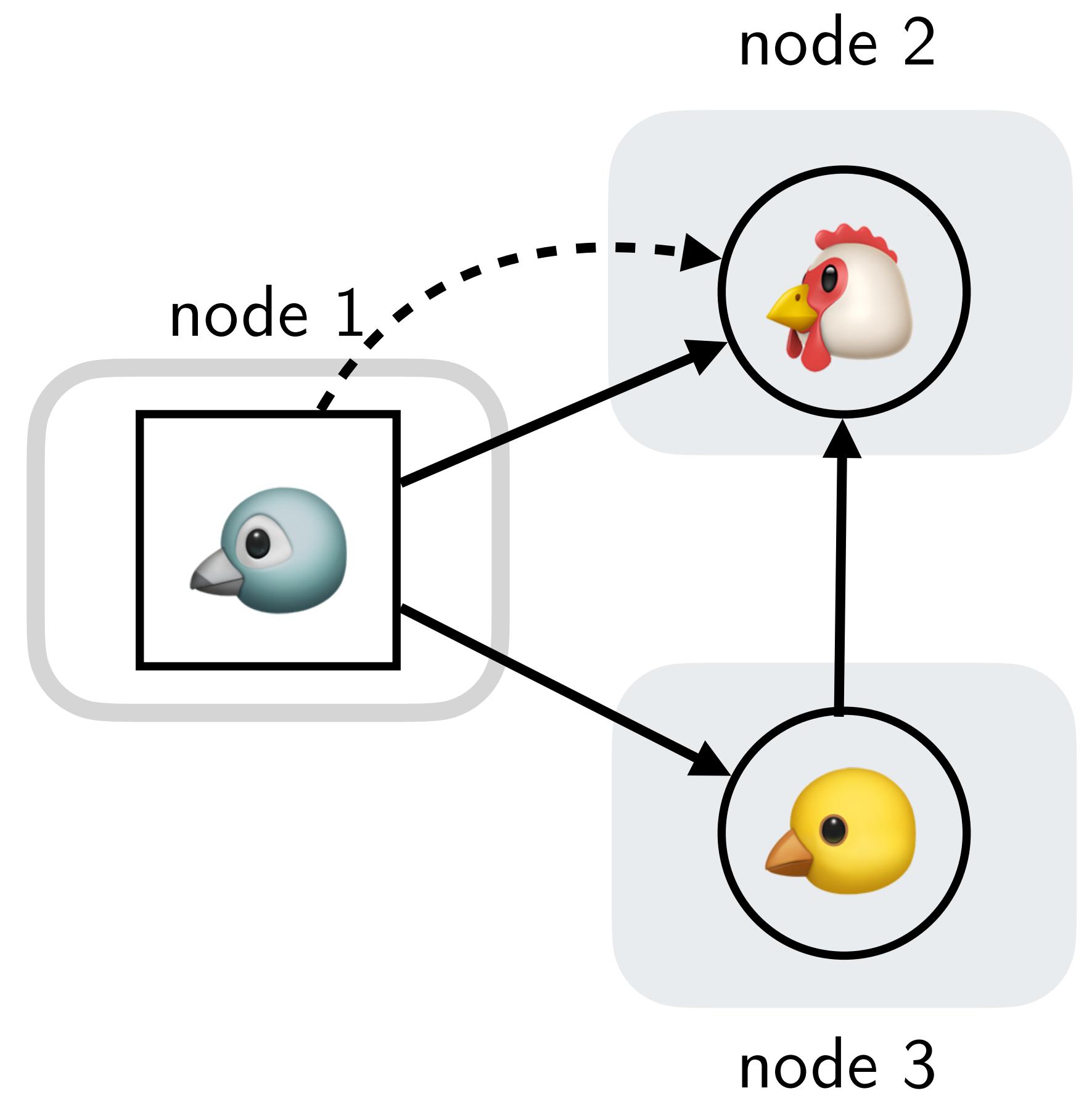


- busy actor
- idle actor
- reference
- message
- healthy node
- crashed node



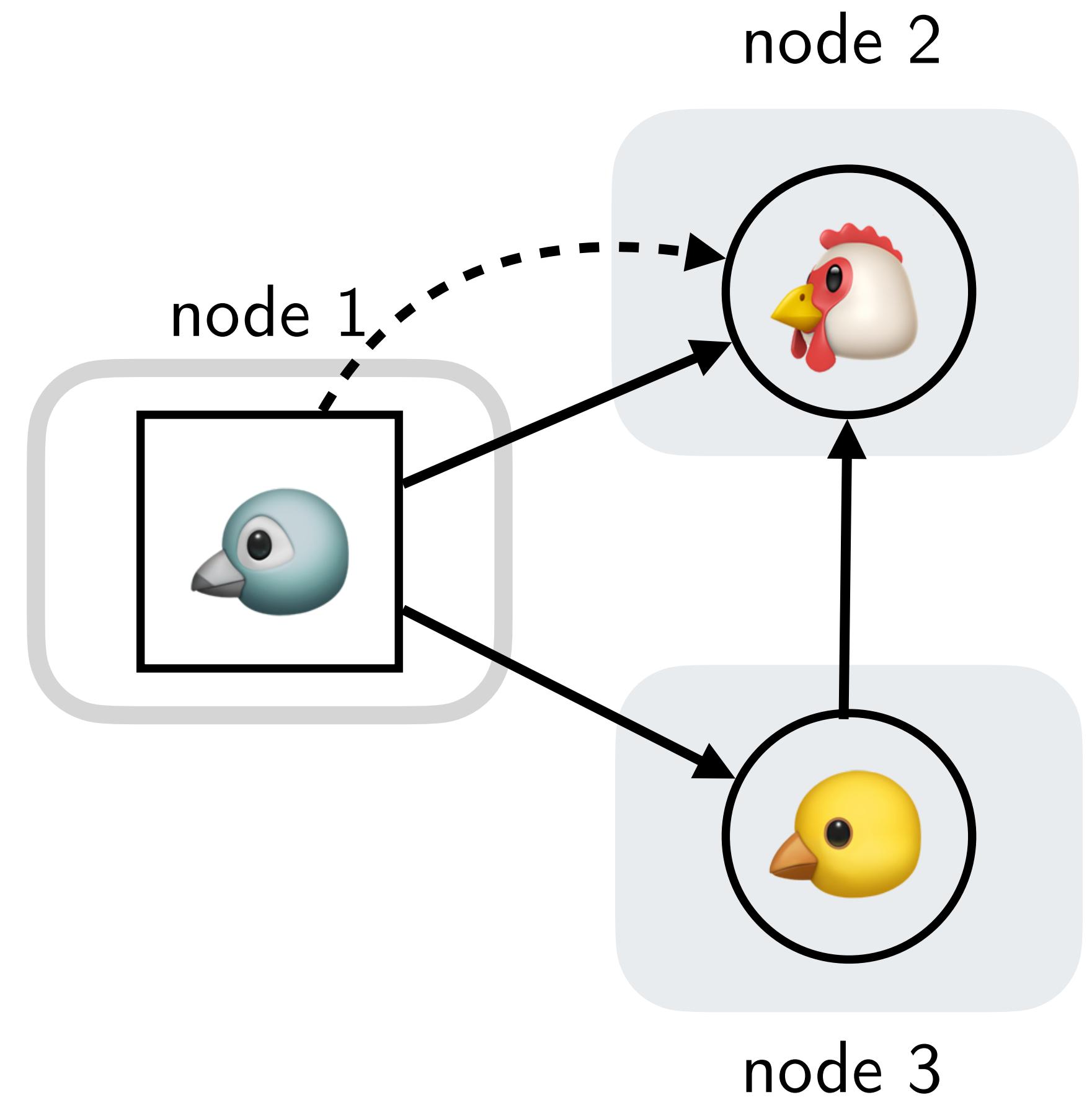
how many messages sent?

- busy actor
- idle actor
- reference
- message
- healthy node
- crashed node



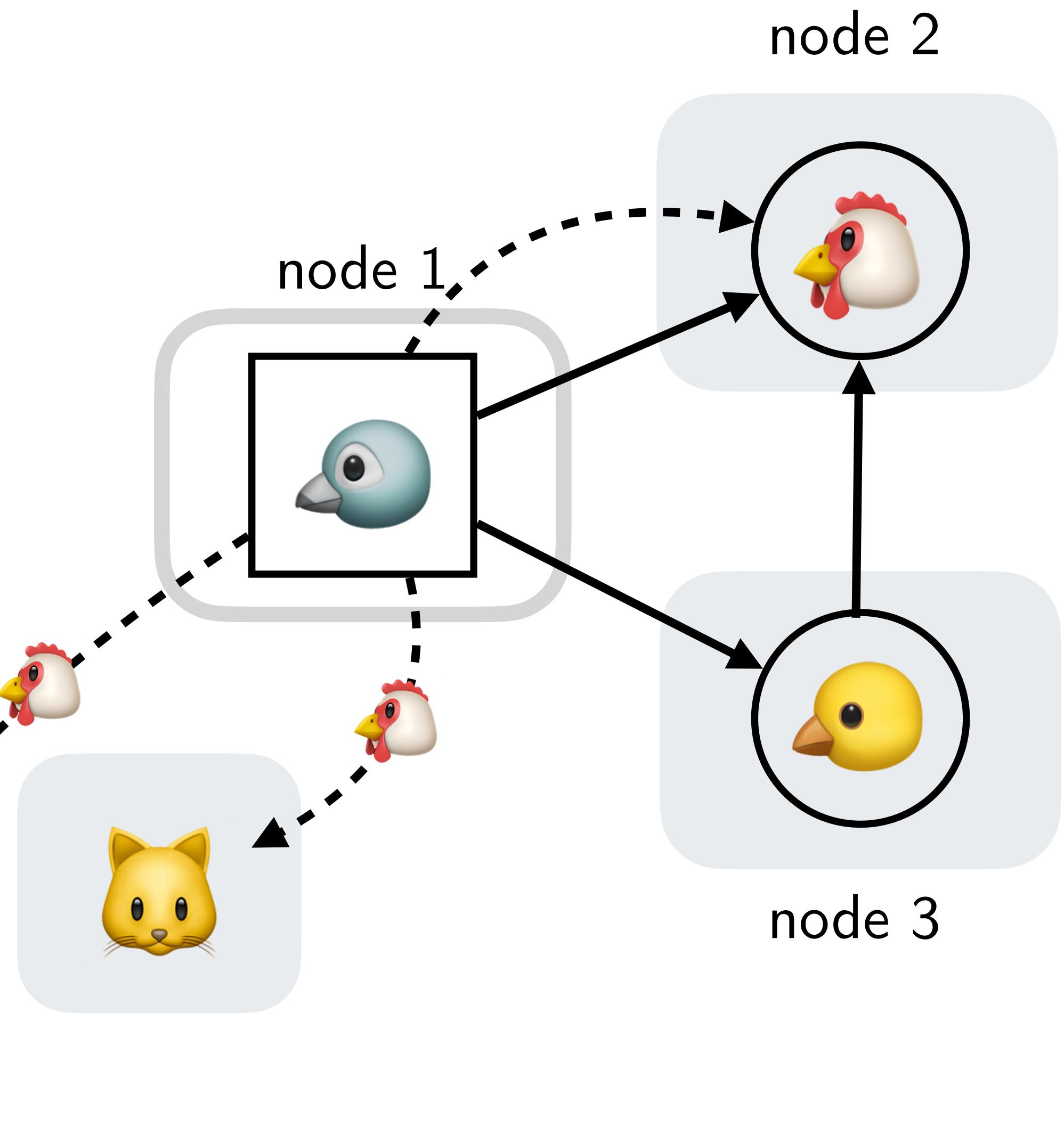
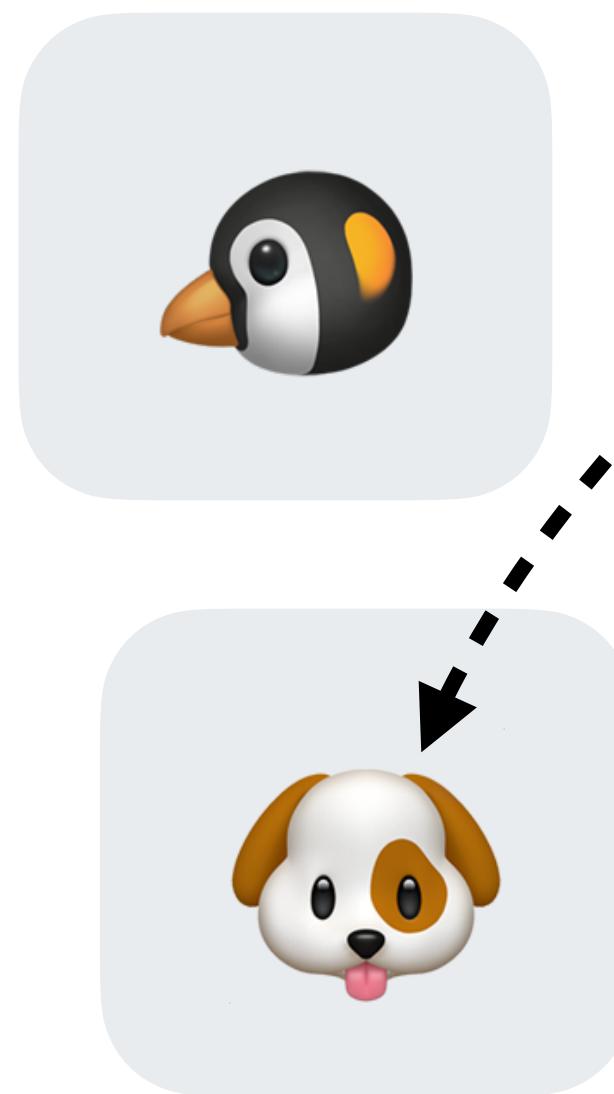
how many messages sent?

- busy actor
- idle actor
- reference
- message
- healthy node
- crashed node



how many messages sent?
did the reference leak?

- busy actor
- idle actor
- reference
- message
- healthy node
- crashed node



**how many messages sent?
did the reference leak?**

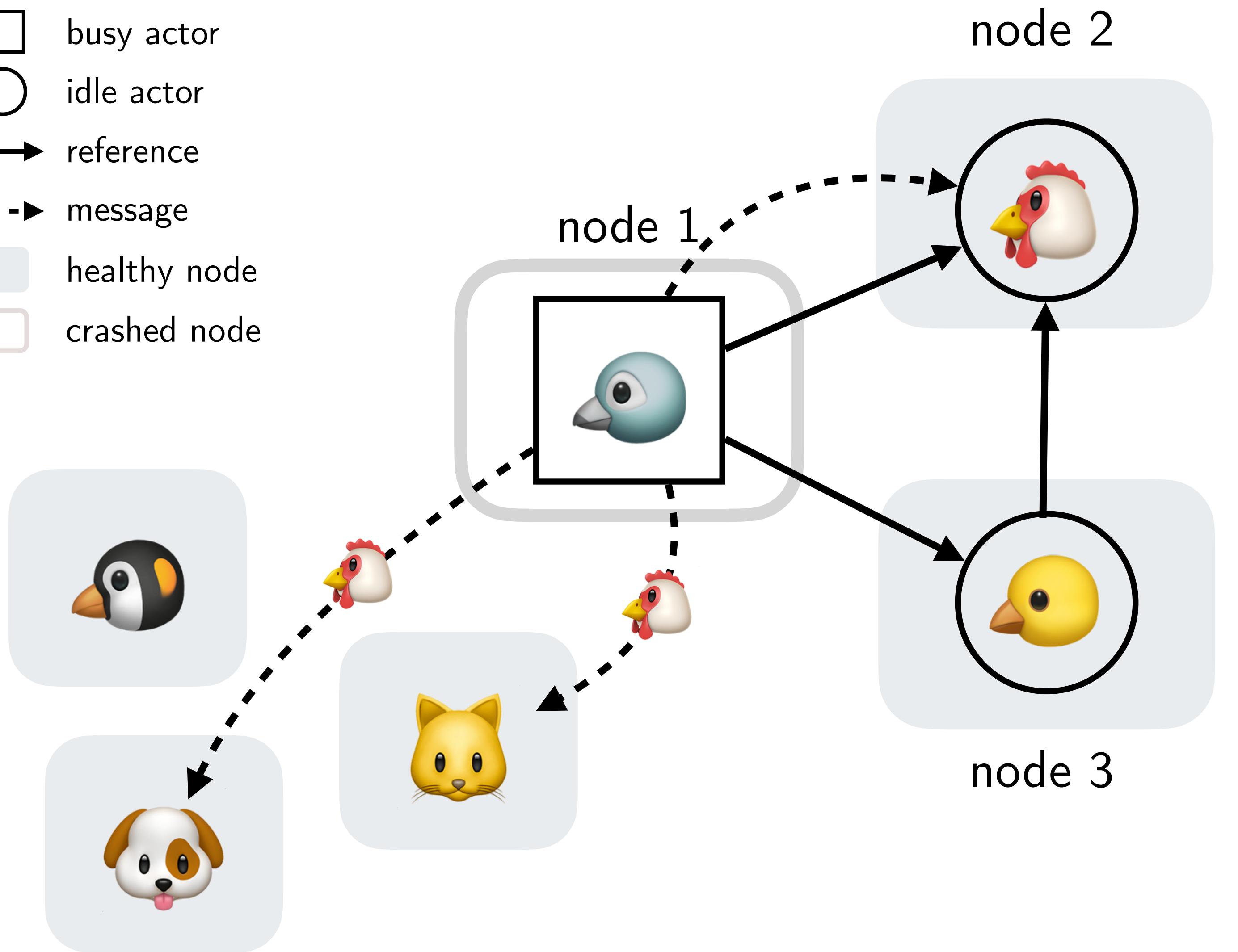
problem: how do we recover the data?

problem: how do we recover the data?

big idea #2

don't ask the crashed node—ask its **neighbors!**

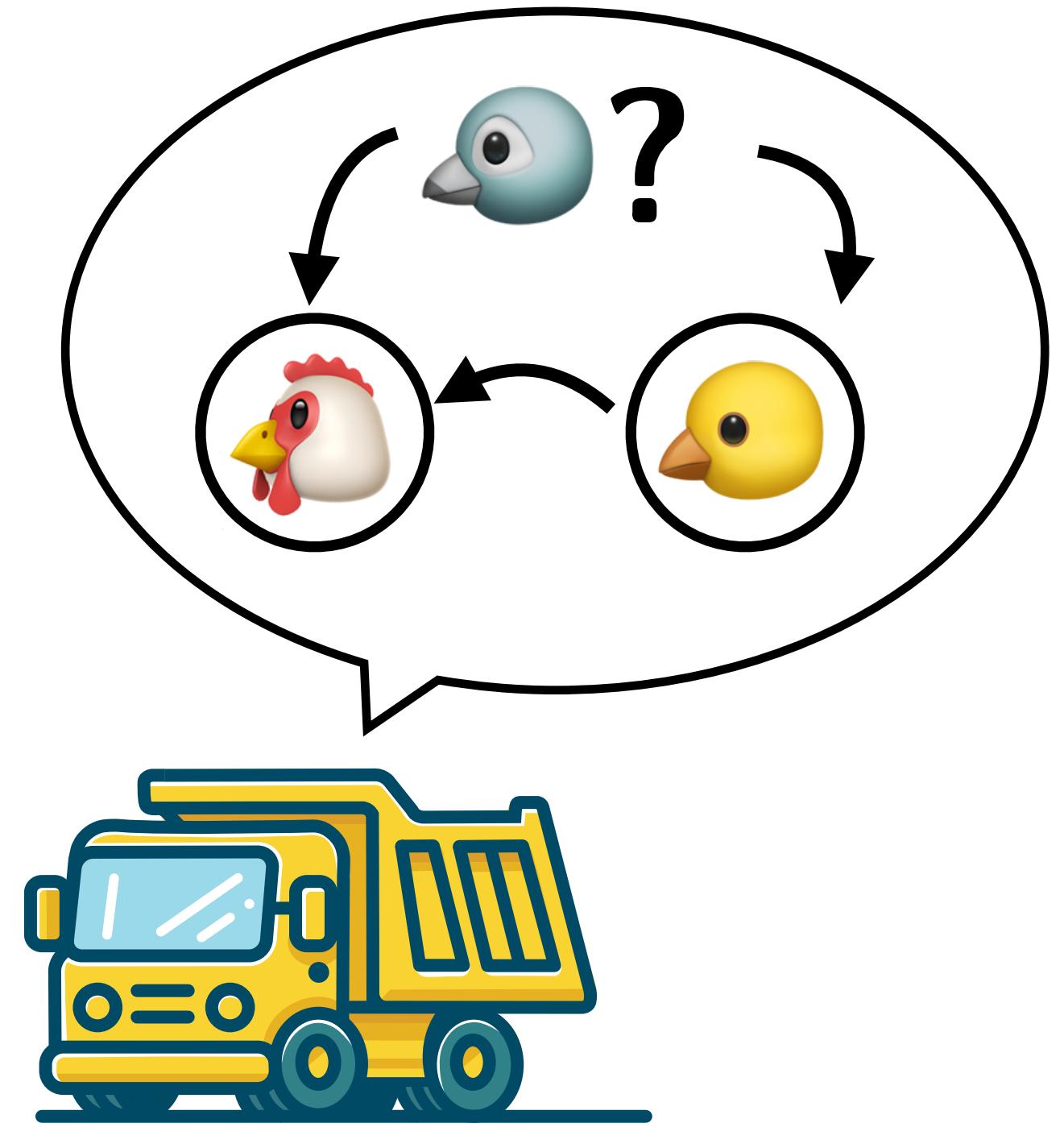
- busy actor
- idle actor
- reference
- message
- healthy node
- crashed node



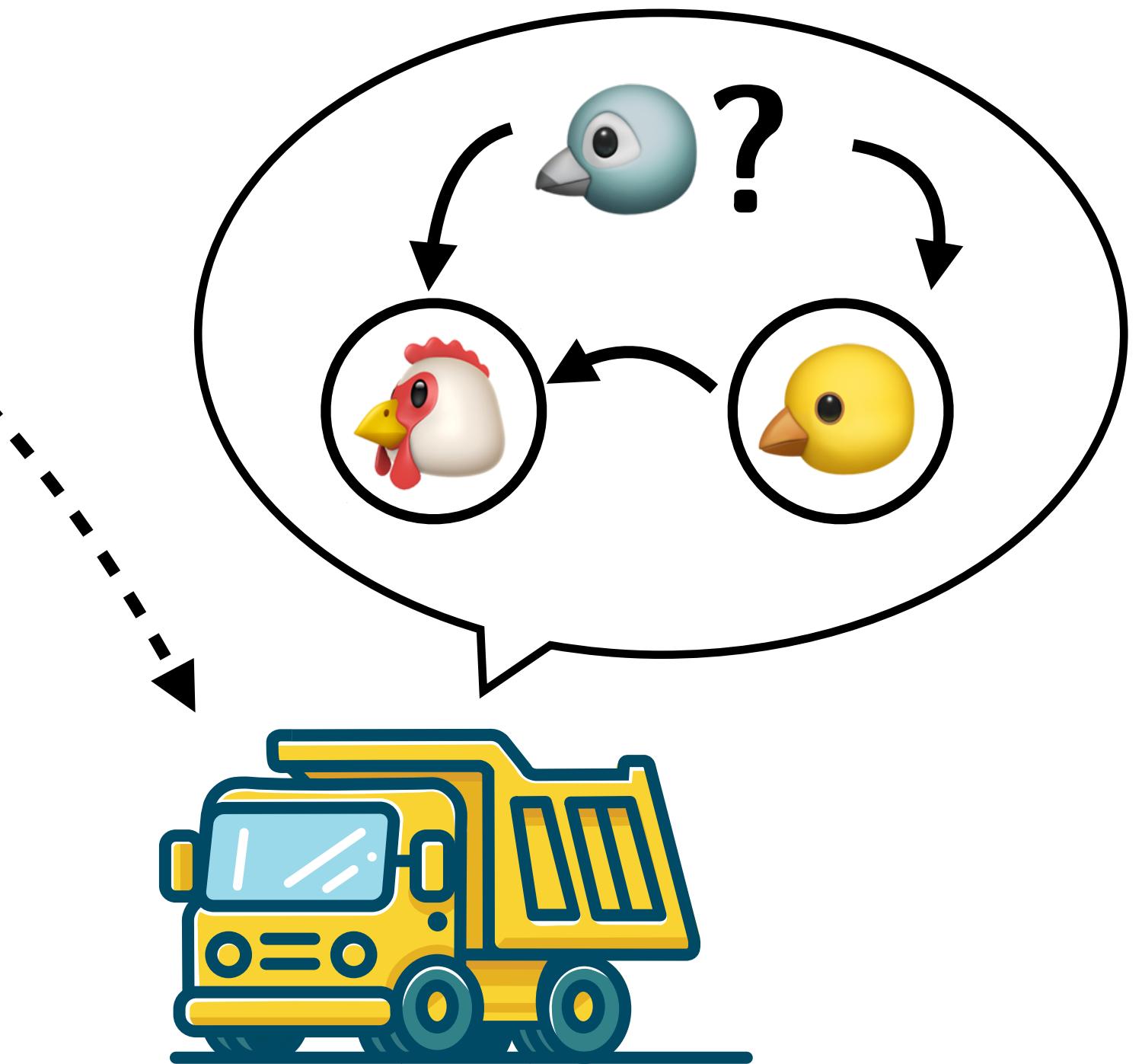
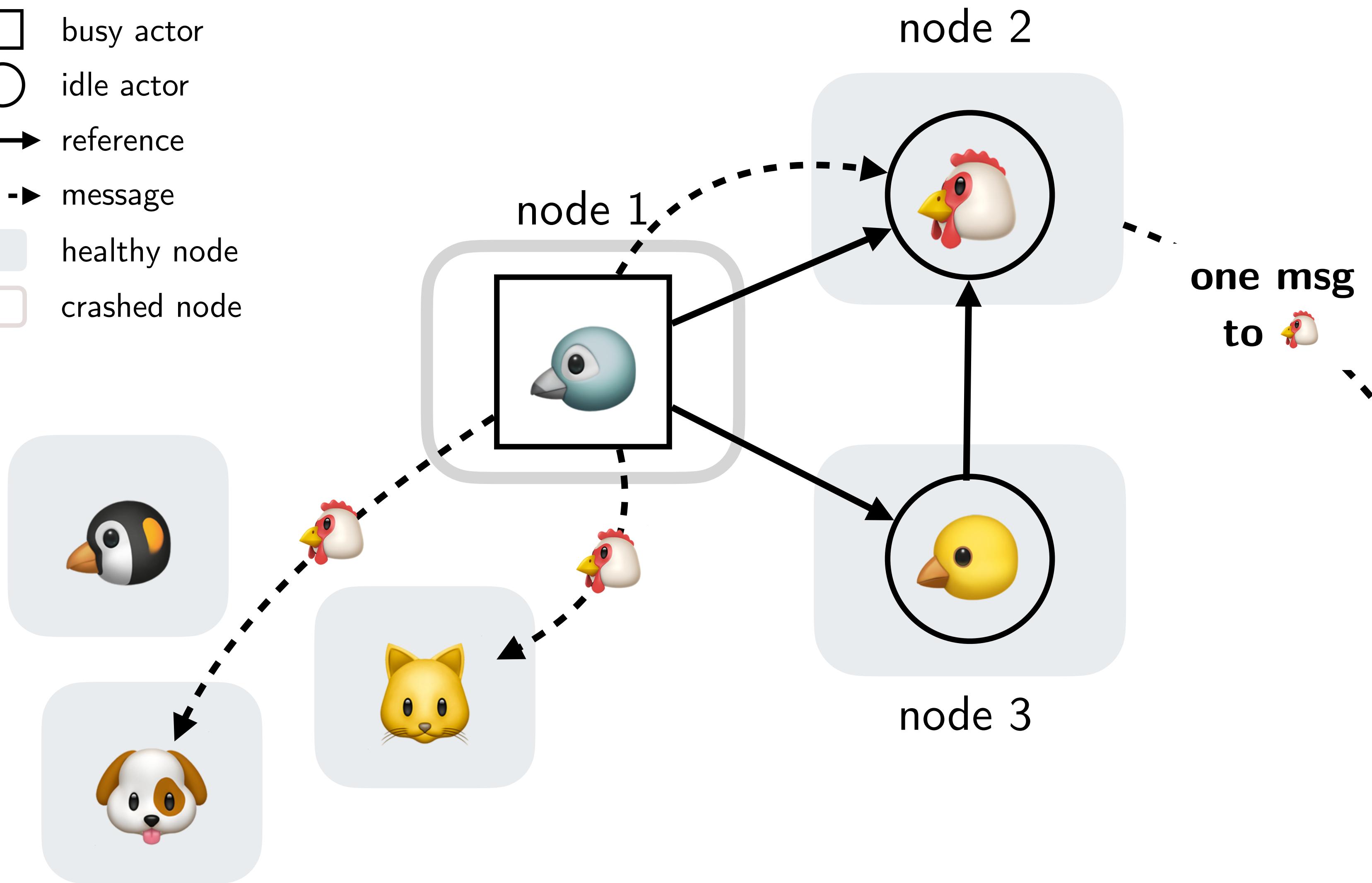
node 2

node 1

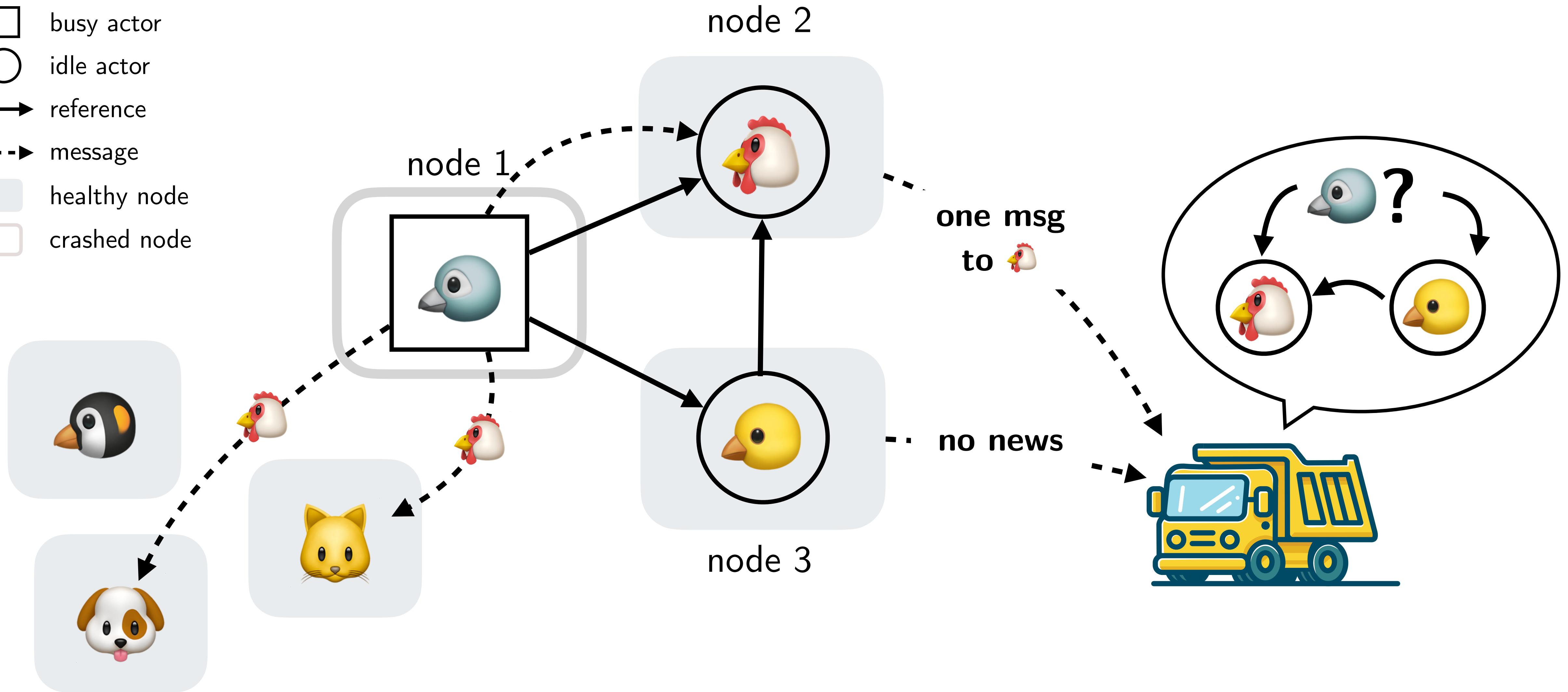
node 3



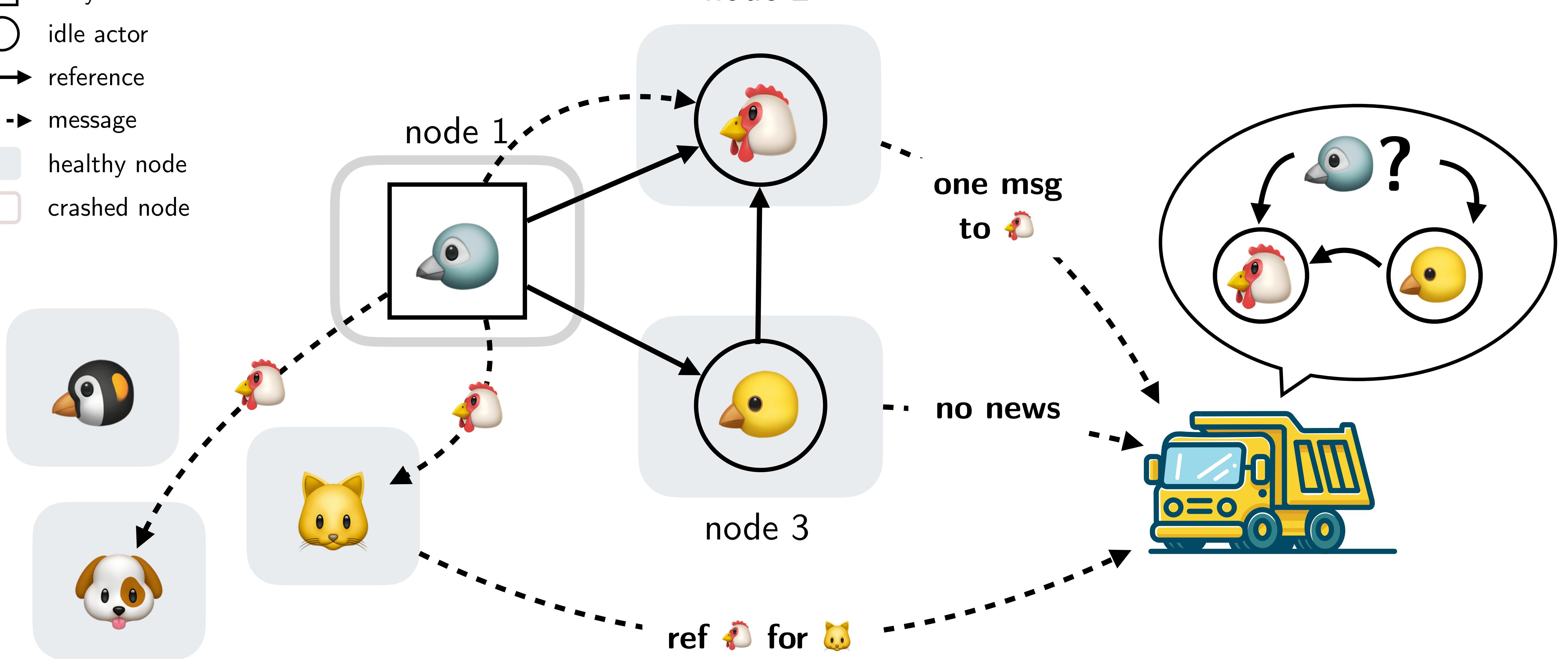
- busy actor
- idle actor
- reference
- message
- healthy node
- crashed node



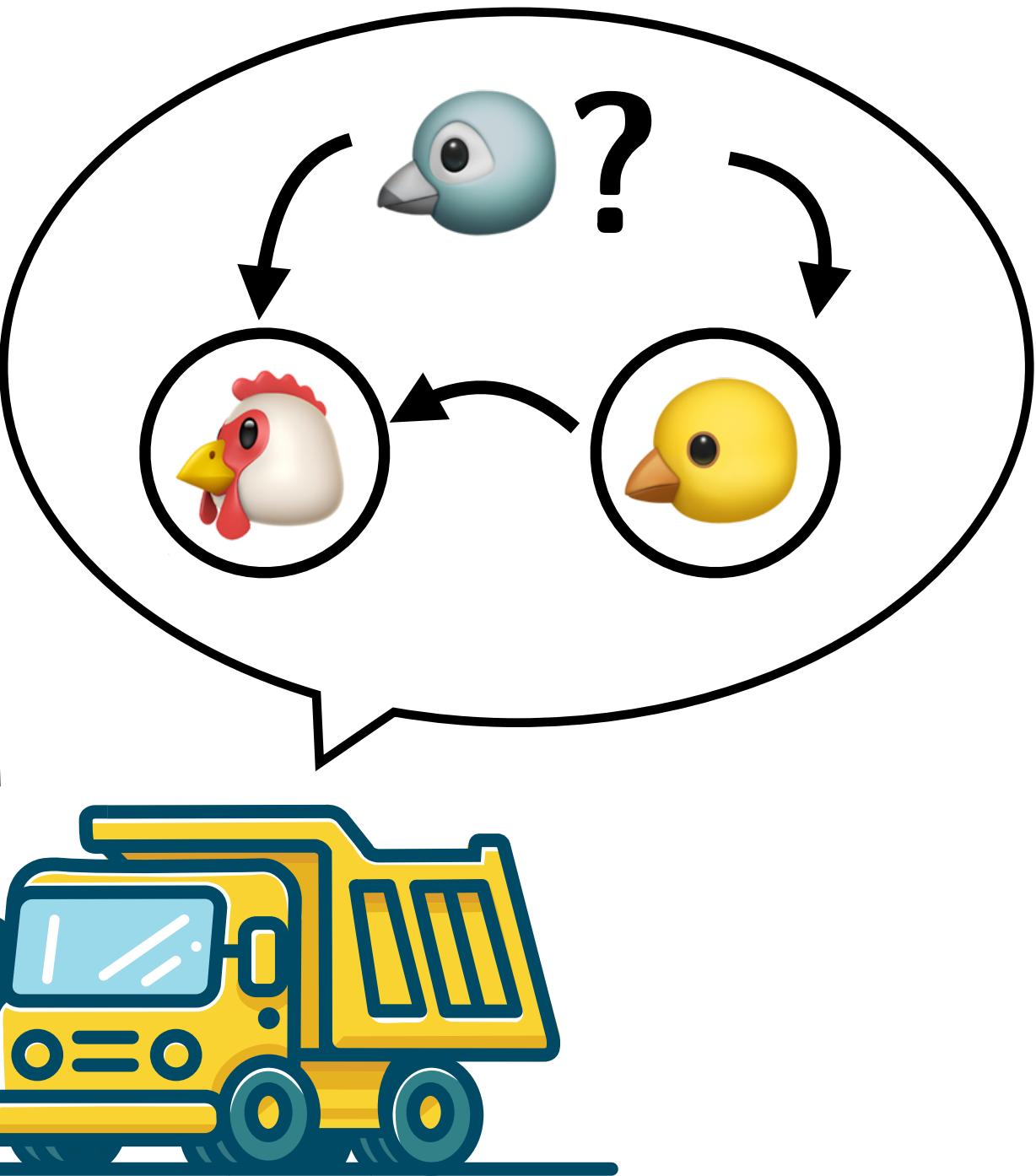
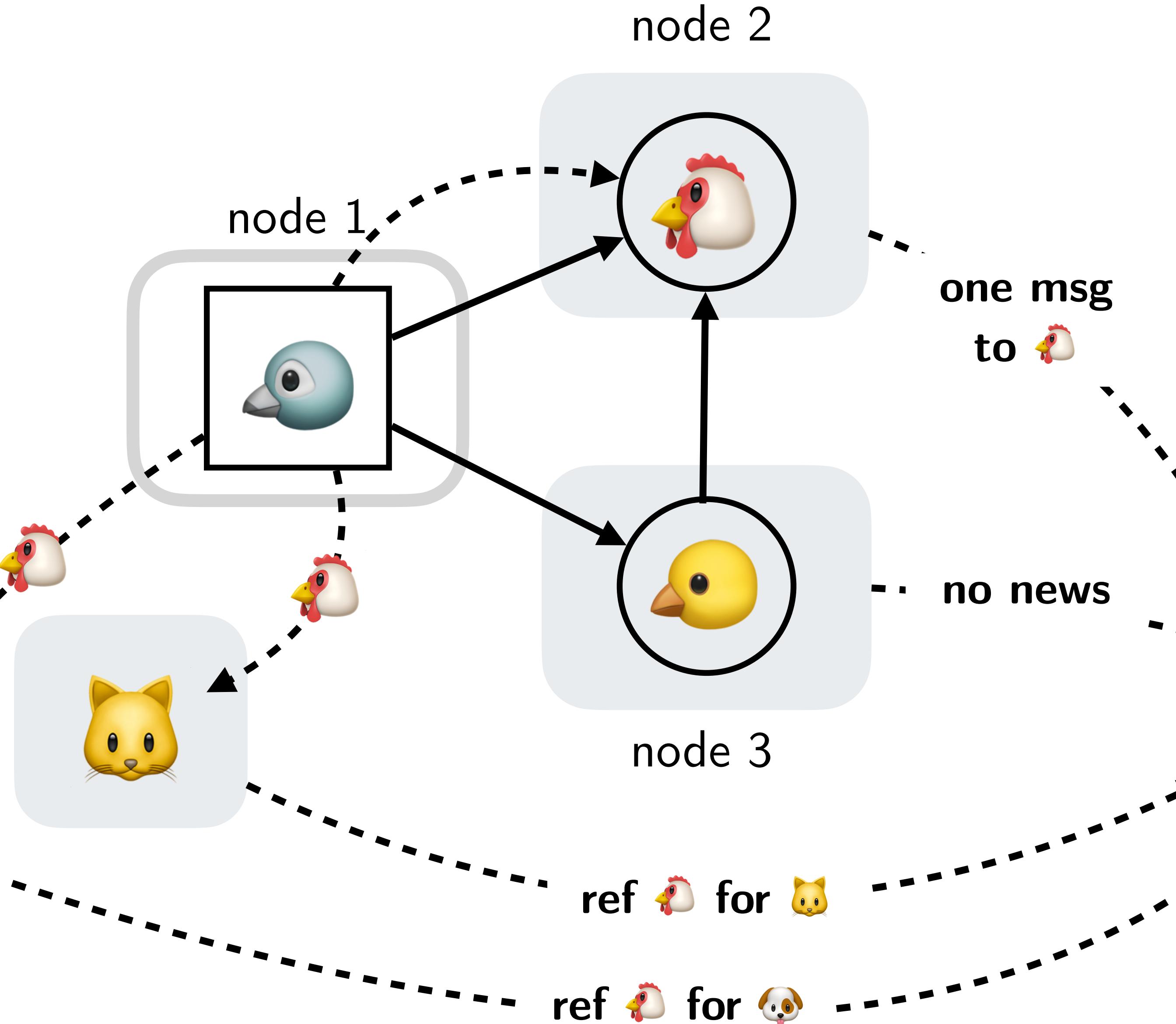
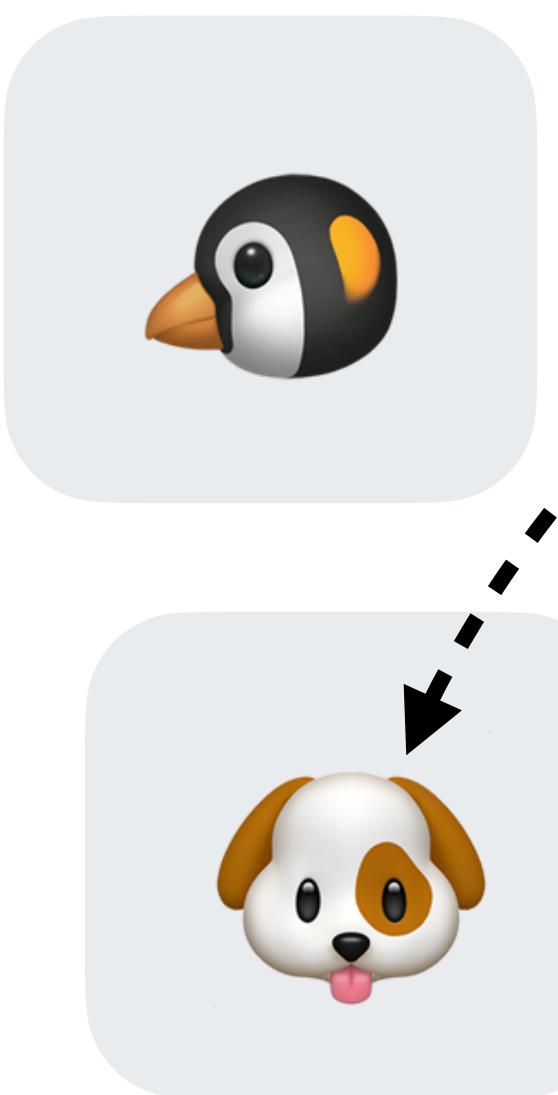
- busy actor
- idle actor
- reference
- message
- healthy node
- crashed node



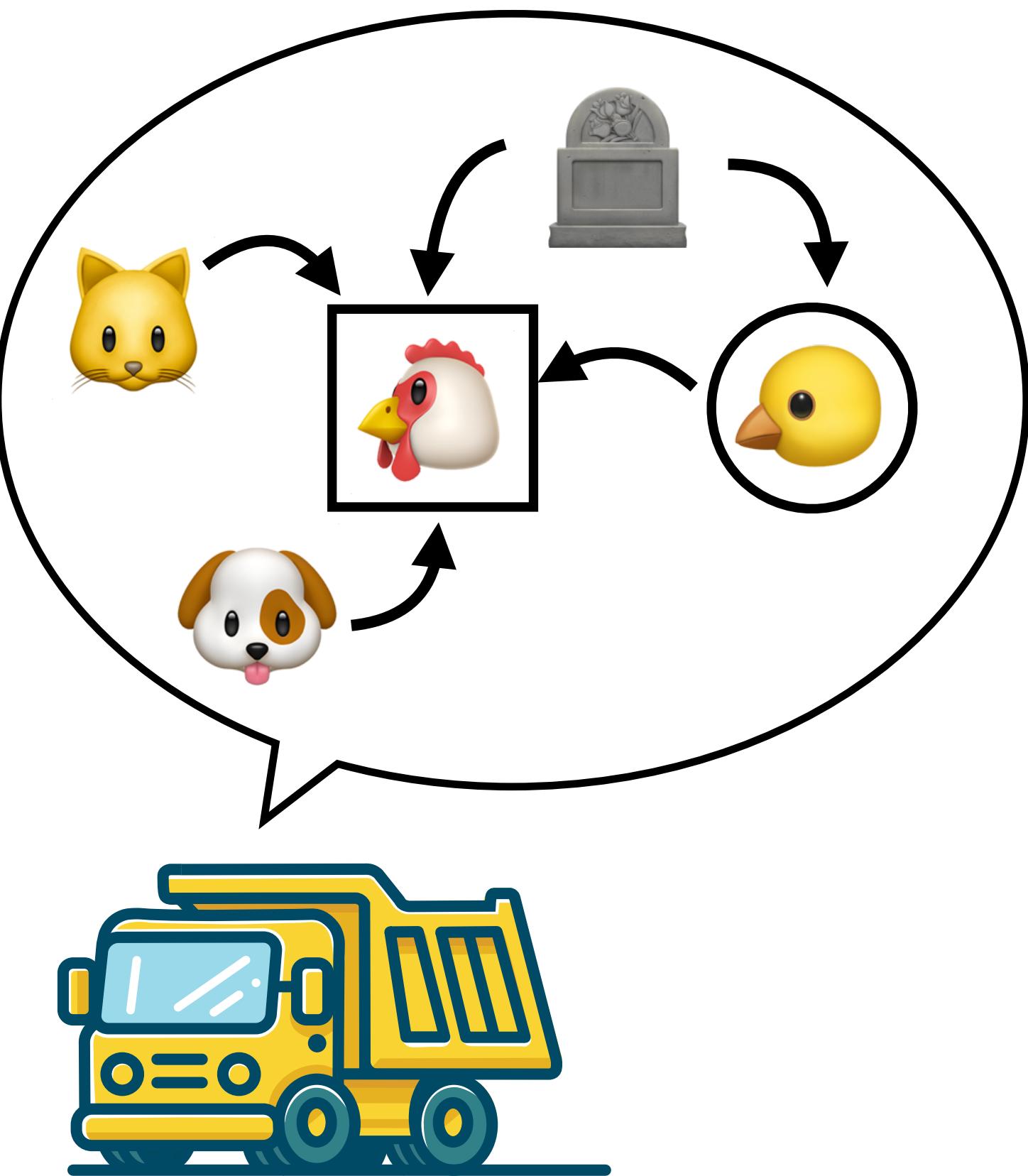
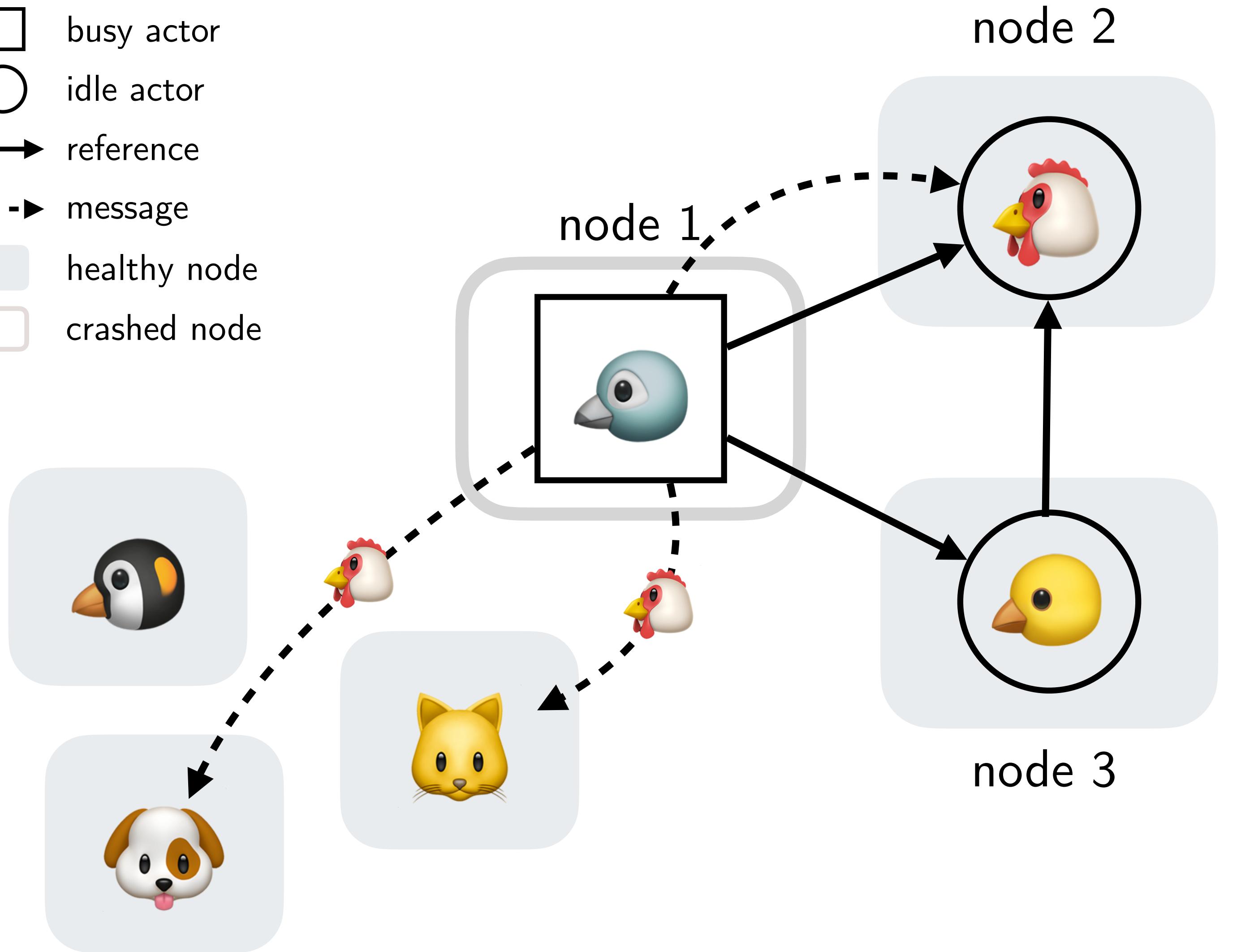
- busy actor
- idle actor
- reference
- message
- healthy node
- crashed node



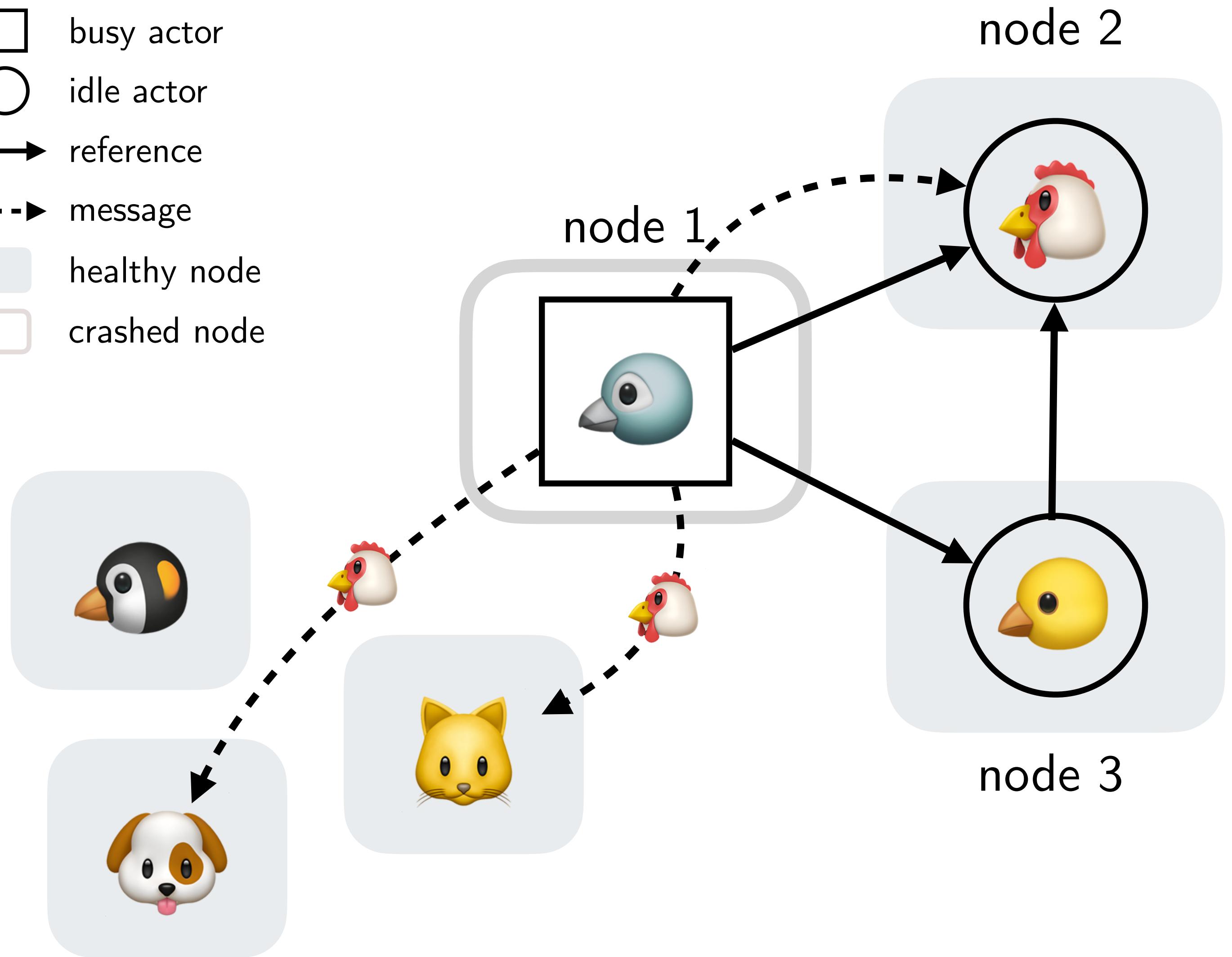
- busy actor
- idle actor
- reference
- message
- healthy node
- crashed node



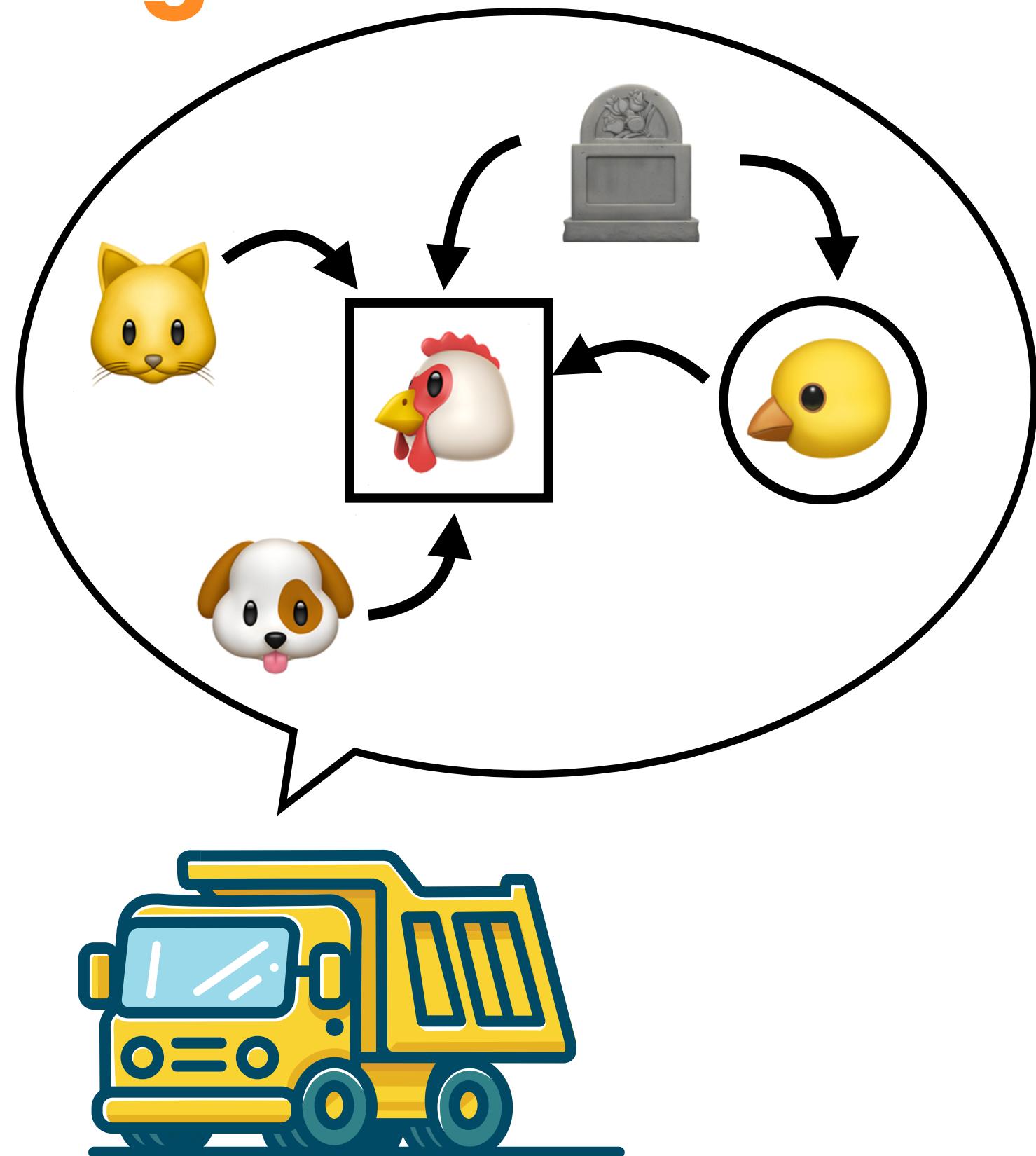
- busy actor
- idle actor
- reference
- message
- healthy node
- crashed node



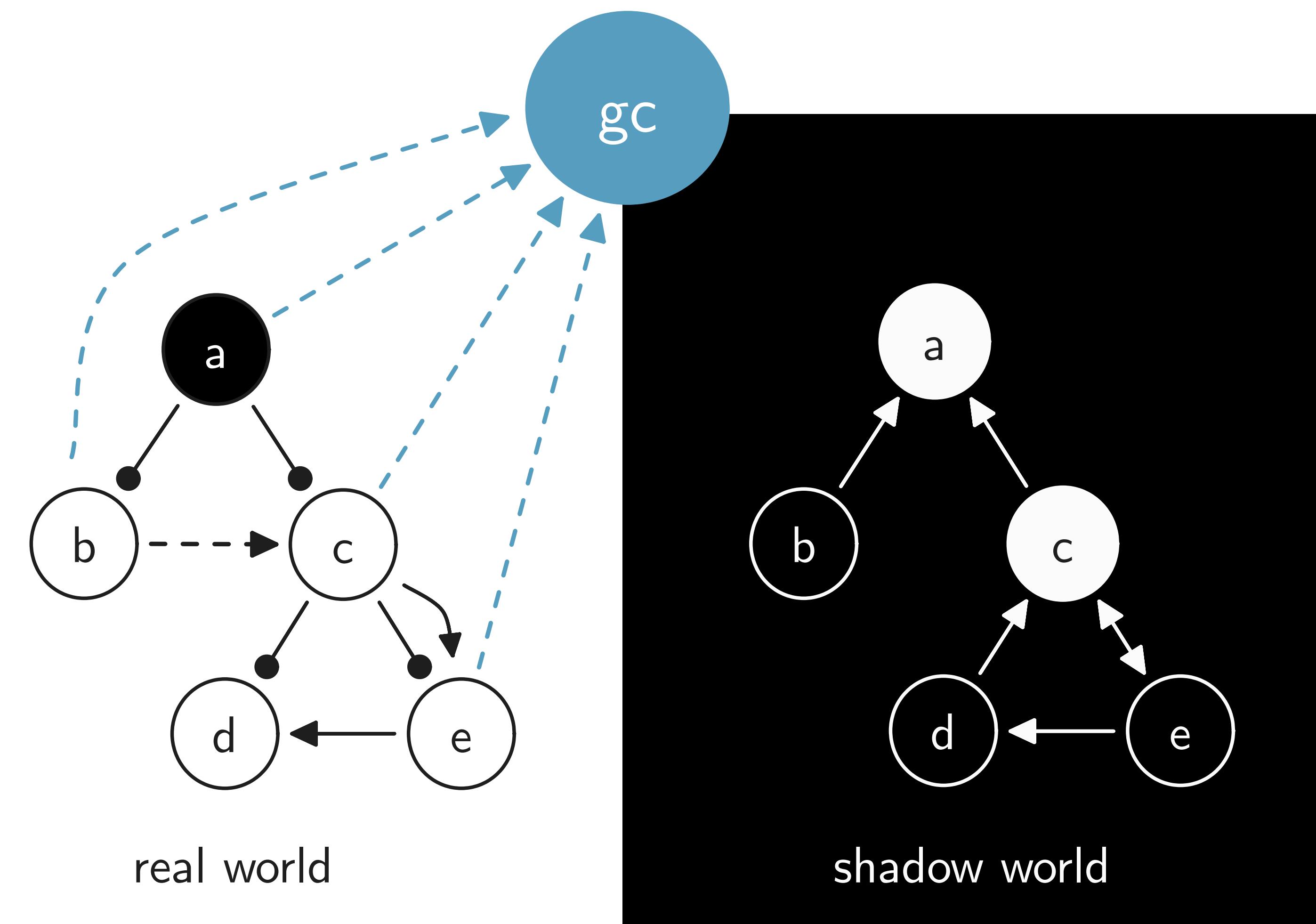
- busy actor
- idle actor
- reference
- message
- healthy node
- crashed node

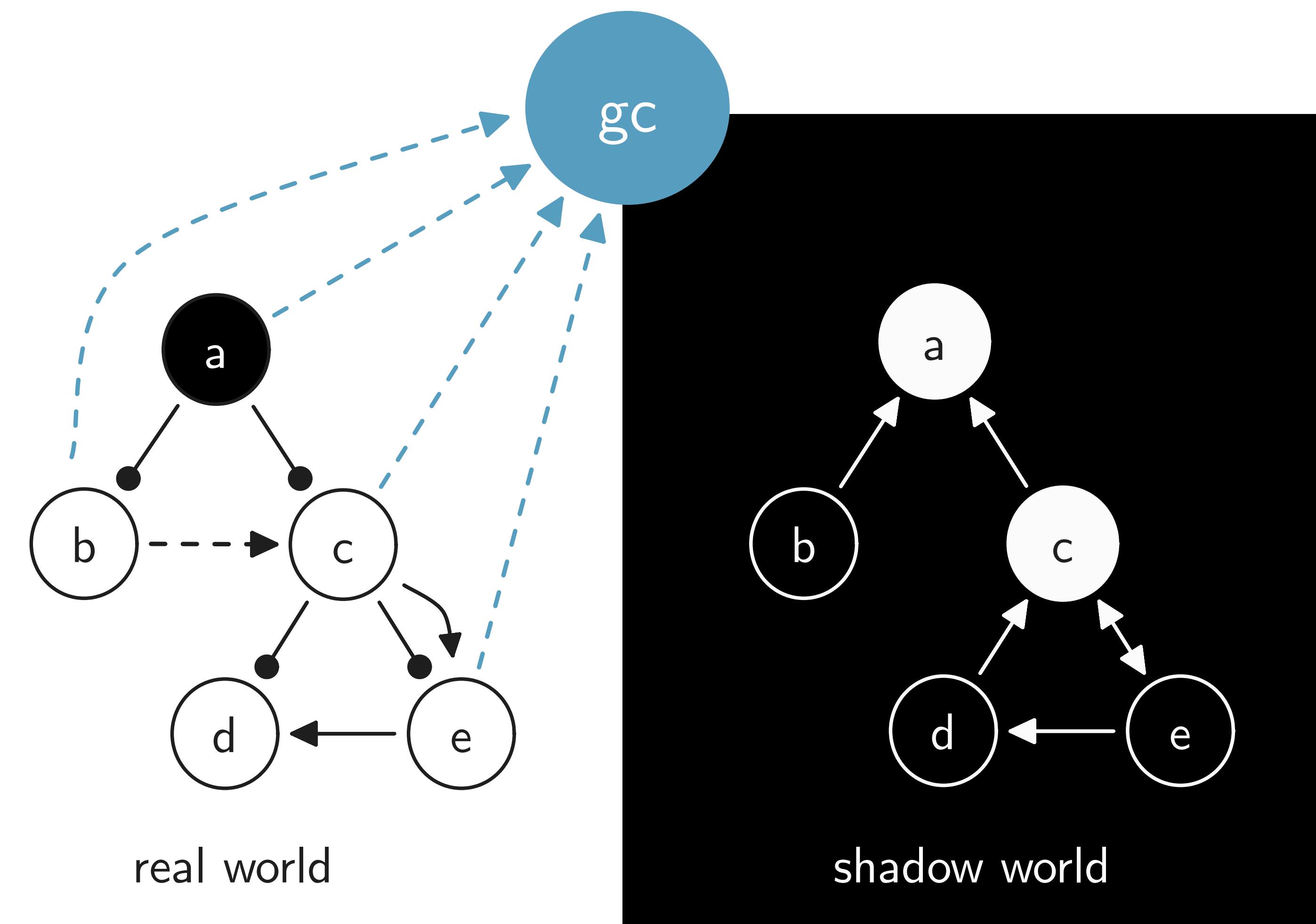


孵 is garbage!

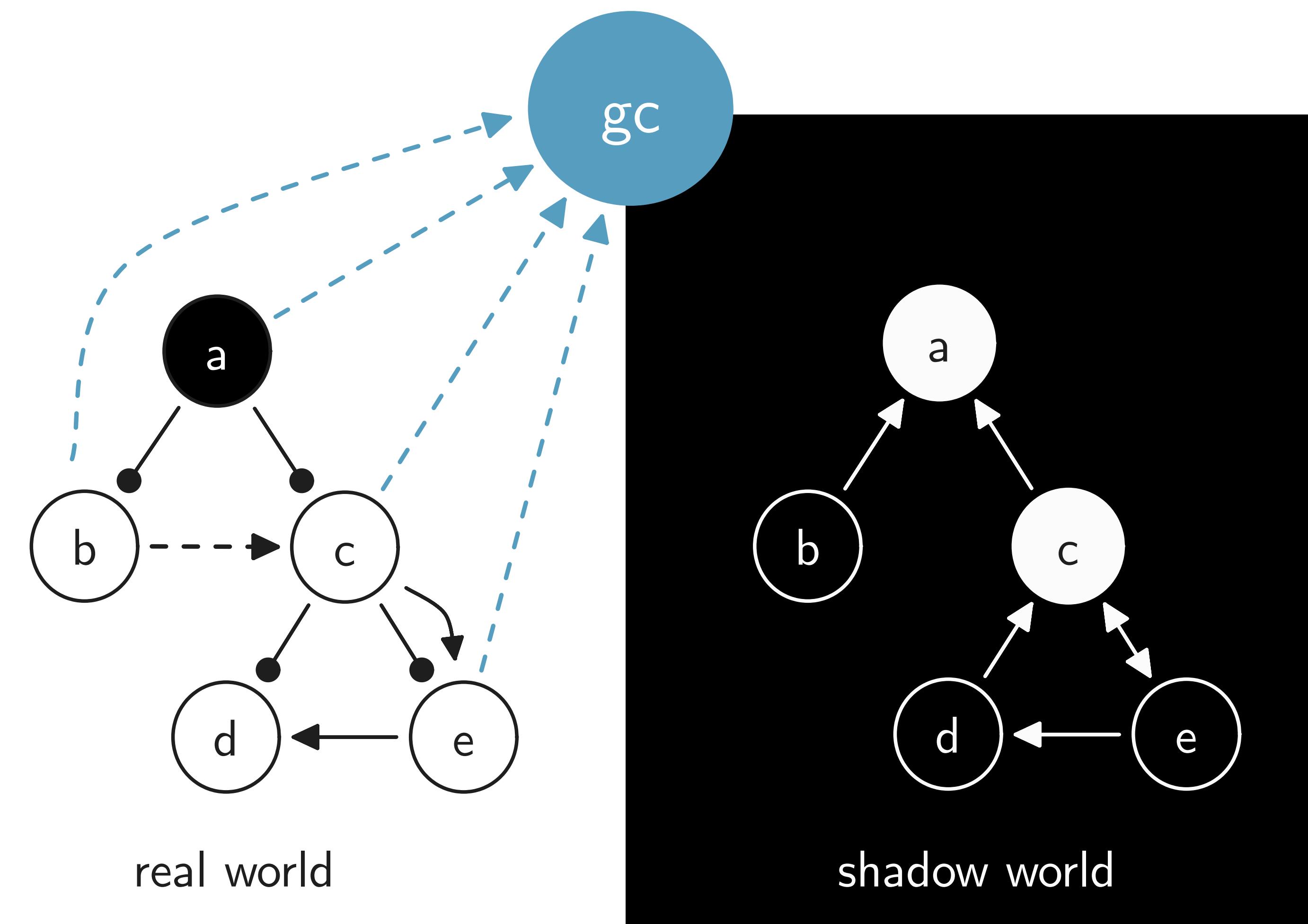


Implementation



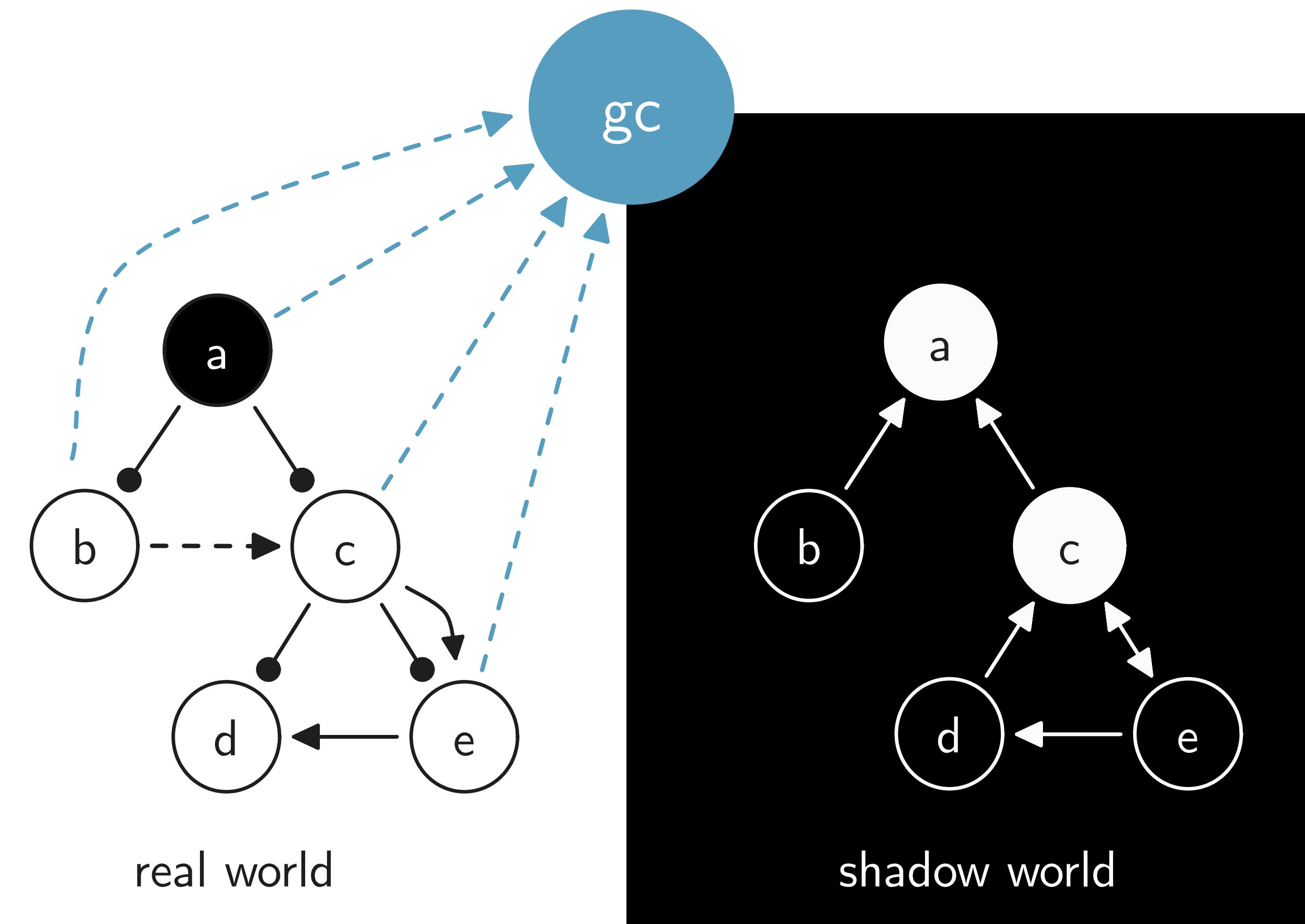


1) actors send updates to GC whenever they want



1) actors send updates to GC whenever they want

2) GC merges updates into a shadow graph

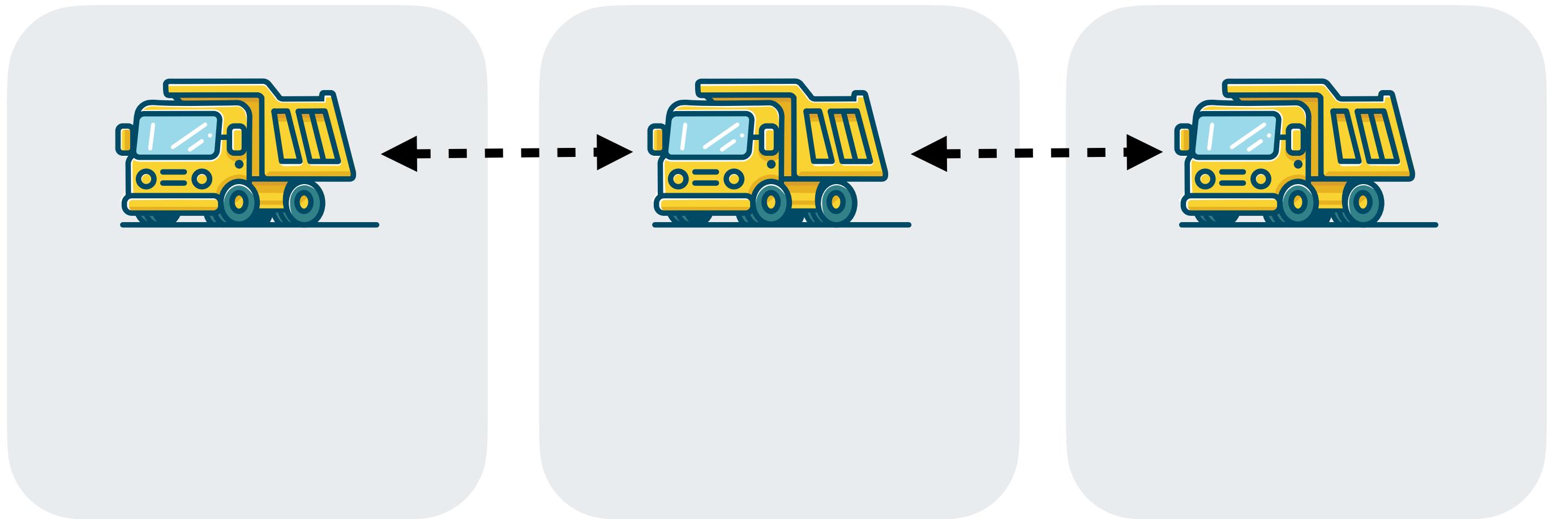


- 1) actors send updates to GC whenever they want**
- 2) GC merges updates into a **shadow graph****
- 3) GC **traces** its graph to find garbage**

node 1

node 2

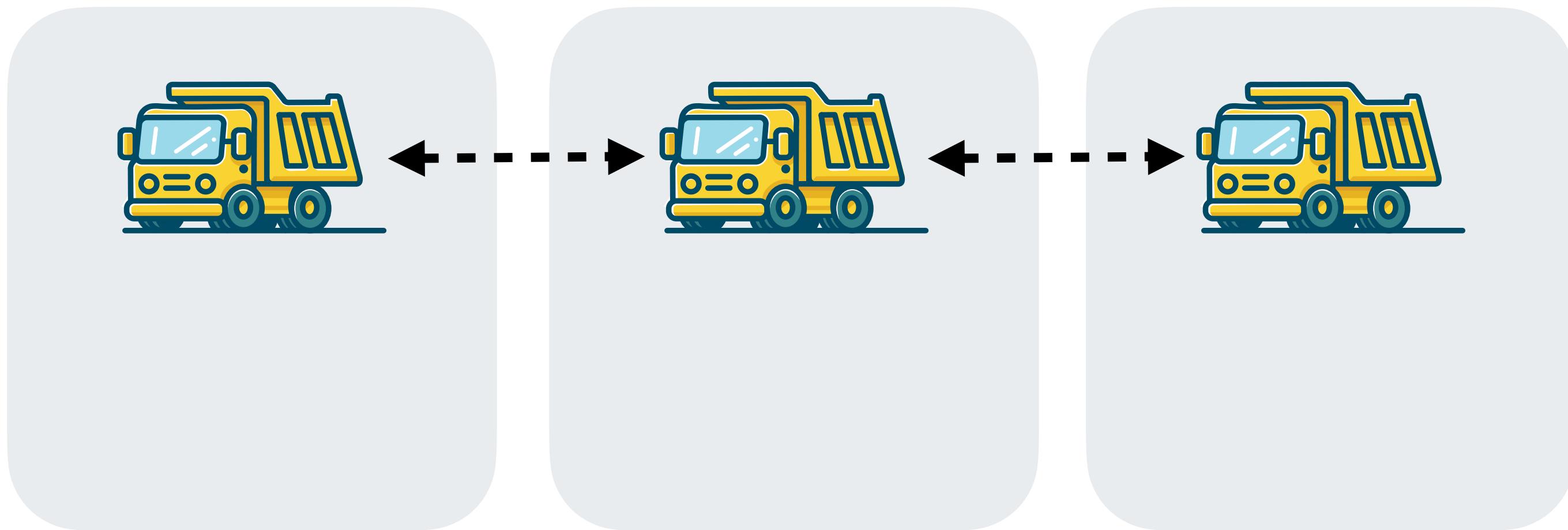
node 3



node 1

node 2

node 3

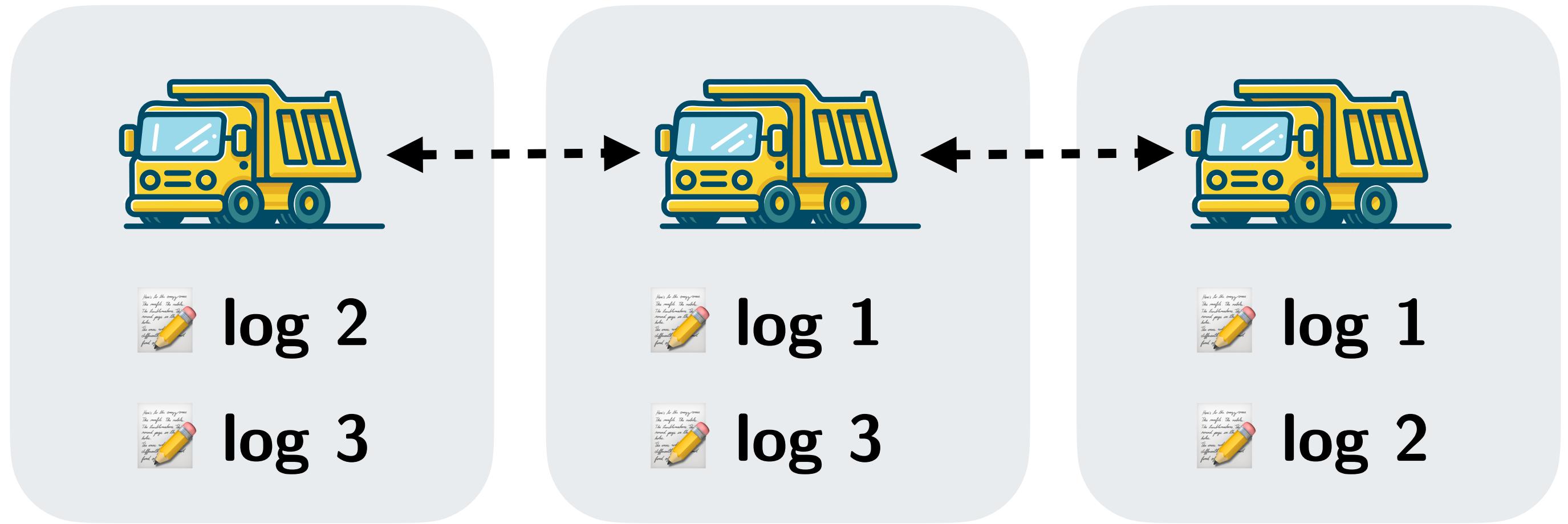


4) local GCs broadcast updates to remote GCs

node 1

node 2

node 3



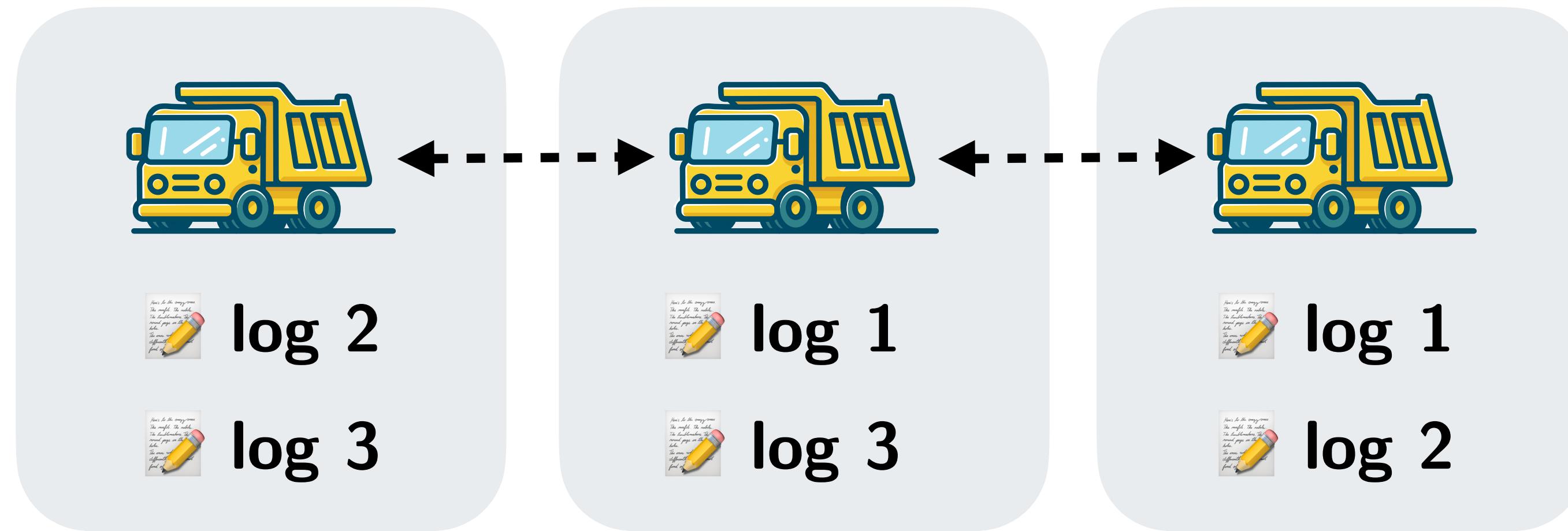
4) local GCs broadcast updates to remote GCs

5) remote GCs build undo logs

node 1

node 2

node 3

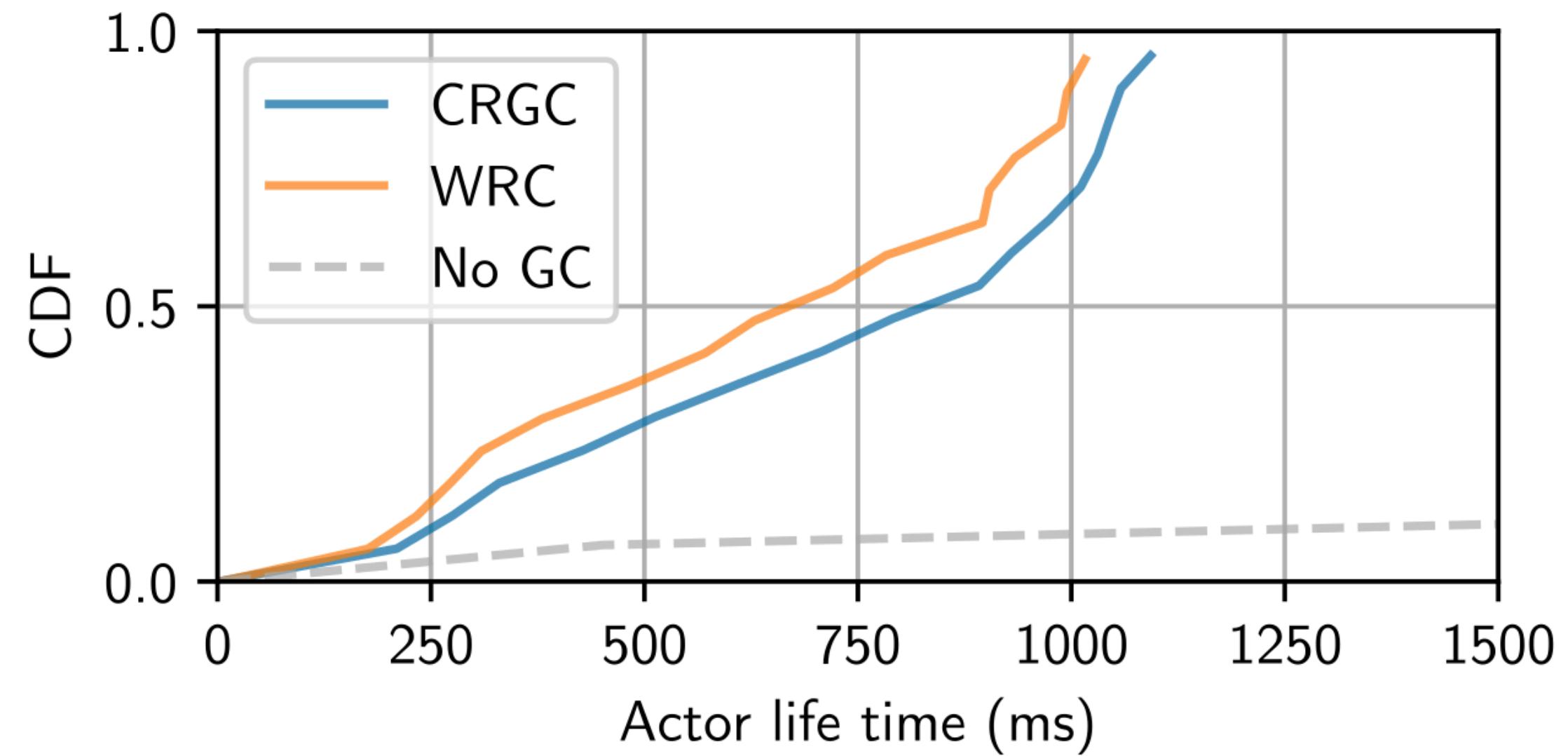


4) local GCs broadcast updates to remote GCs

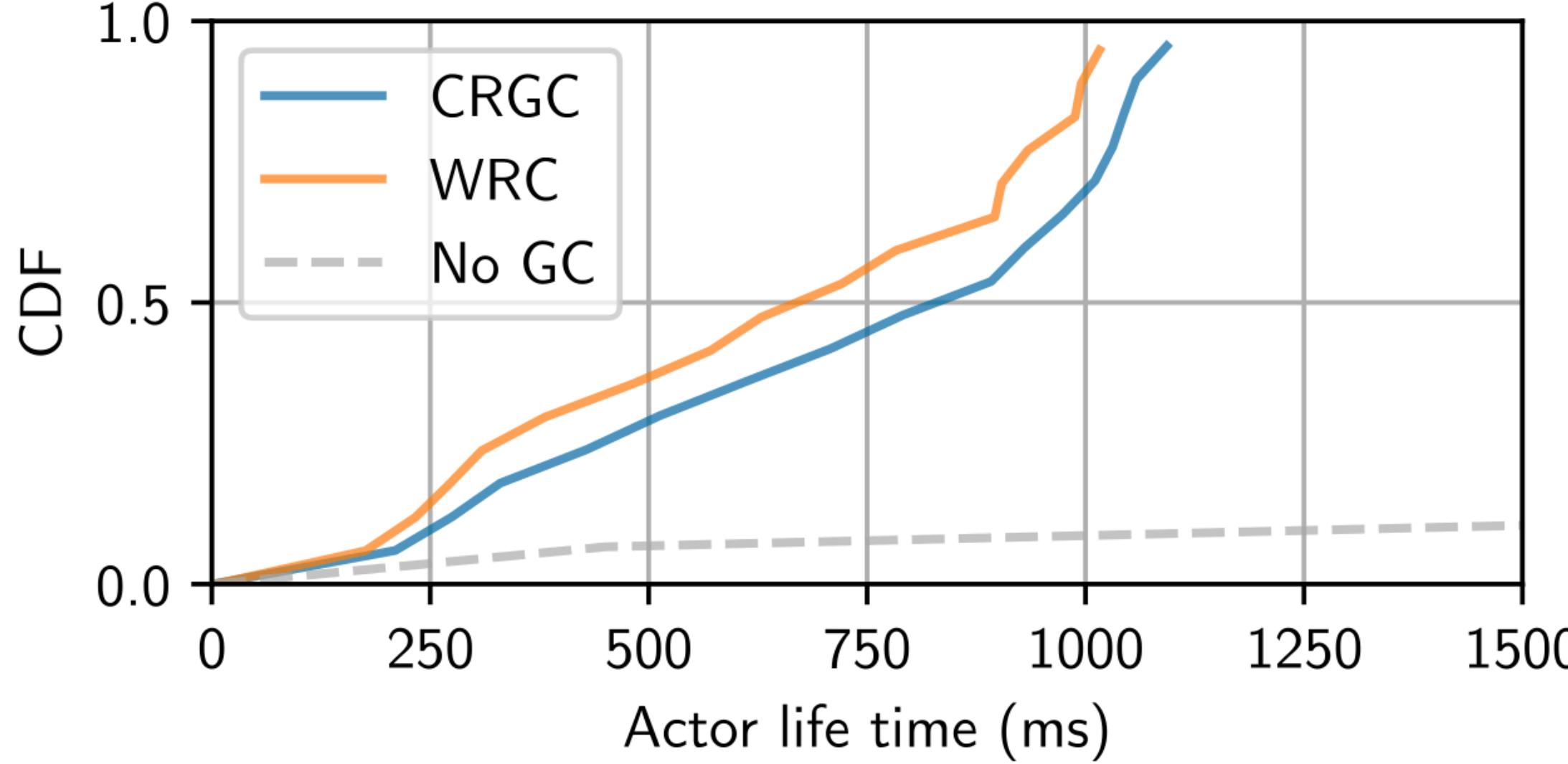
5) remote GCs build undo logs

6) if node X fails, merge the undo log into the shadow graph

acyclic garbage collection speed



acyclic garbage collection speed



average slowdown (%)

Benchmark	$No\ GC\ (stddev)$	Parallelism	
		WRC	CRGC-block
apsp	± 5	-1	3
astar	± 27	-12	-5
bitonicsort	± 5	39	4
facloc	± 2	4	51
nqueenk	± 1	5	5
piprecision	± 2	0	0
quicksort	± 1	0	-1
radixsort	± 8	1	2
repmatmul	± 1	0	0
sieve	± 2	0	1
trapezoid	± 1	0	0
uct	± 7	25	22
geomean		4	6

we need your help!

we need your help!

traditional GC tricks

we need your help!

traditional GC tricks

detecting shared
references

we need your help!

traditional GC tricks

bug study

**detecting shared
references**

we need your help!

traditional GC tricks

bug study

**detecting shared
references**

porting to BEAM

we need your help!

traditional GC tricks

bug study

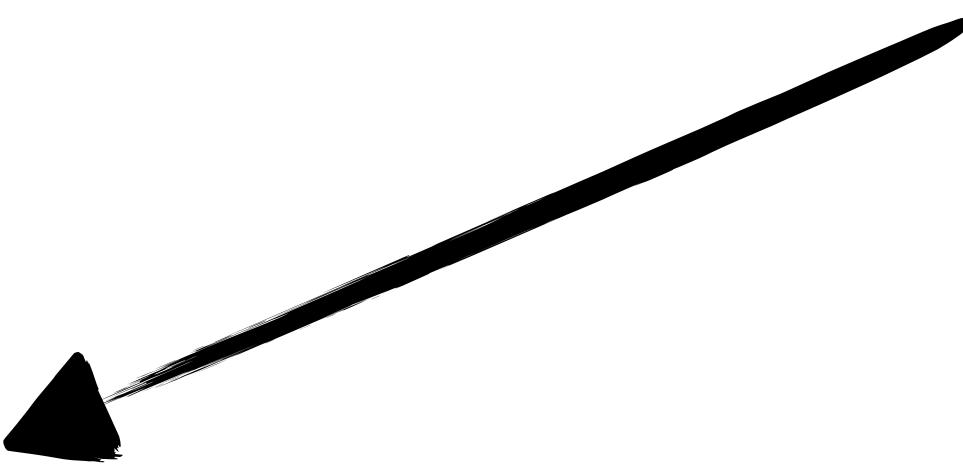
**detecting shared
references**

porting to BEAM

scaling to large clusters

thanks!

***on the job market!**



thanks!