GETTING THE WORD OUT:

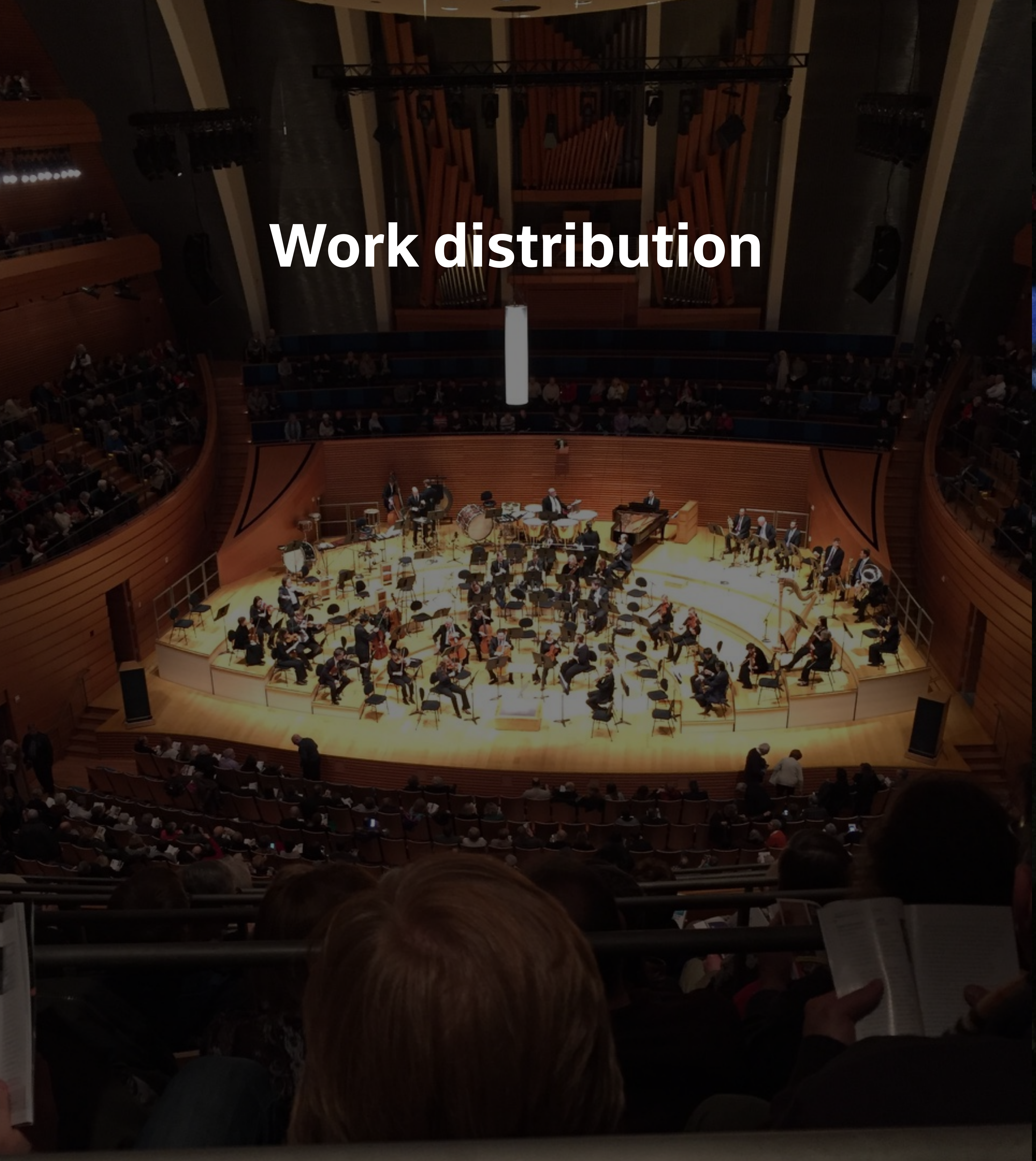
MEMBERSHIP, DISSEMINATION & POPULATION PROTOCOLS

SEAN CRIBBS
SENIOR PRINCIPAL ENGINEER
COMCAST

All photos are my own unless attributed.

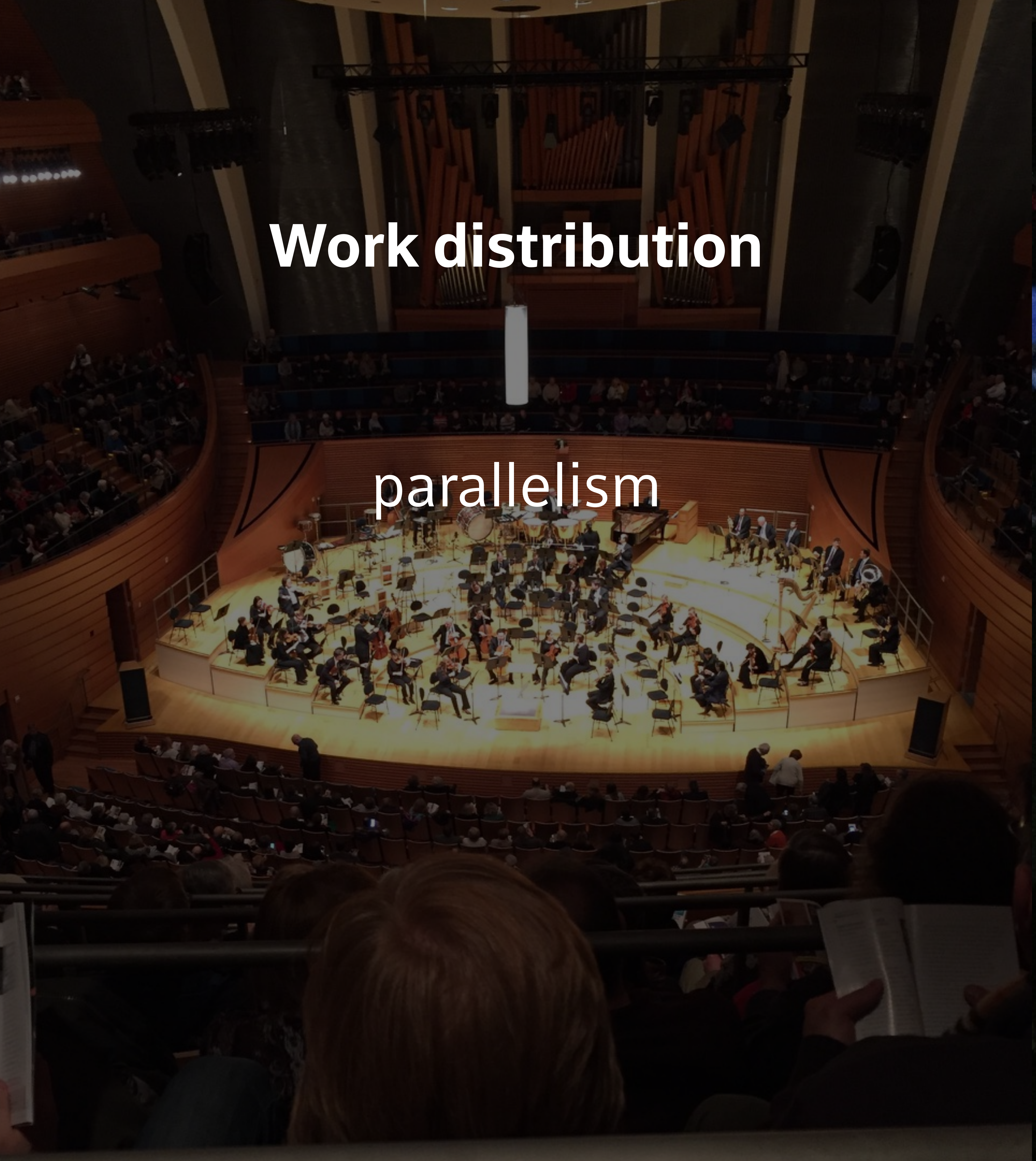
WHY BUILD PEER-TO-PEER SYSTEMS?

Work distribution



Work distribution

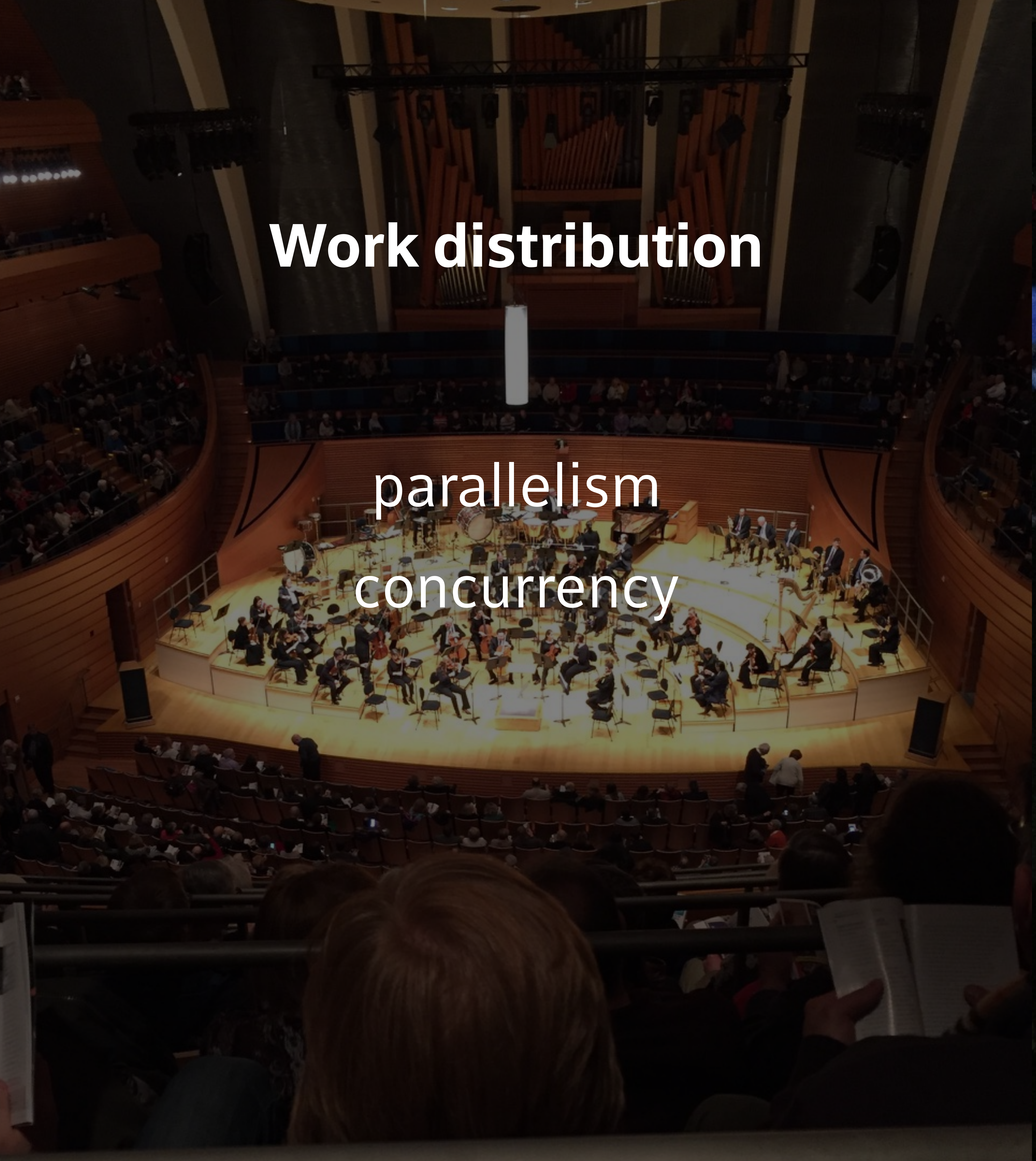
parallelism



Work distribution

parallelism

concurrency



Work distribution

parallelism

concurrency

independence

Work distribution

parallelism

concurrency

independence

Fault Tolerance

Work distribution

parallelism

concurrency

independence

Fault Tolerance

detection

Work distribution

parallelism

concurrency

independence

Fault Tolerance

detection

recovery

Work distribution

parallelism

concurrency

independence

Fault Tolerance

detection

recovery

redundancy

Work distribution

parallelism

concurrency

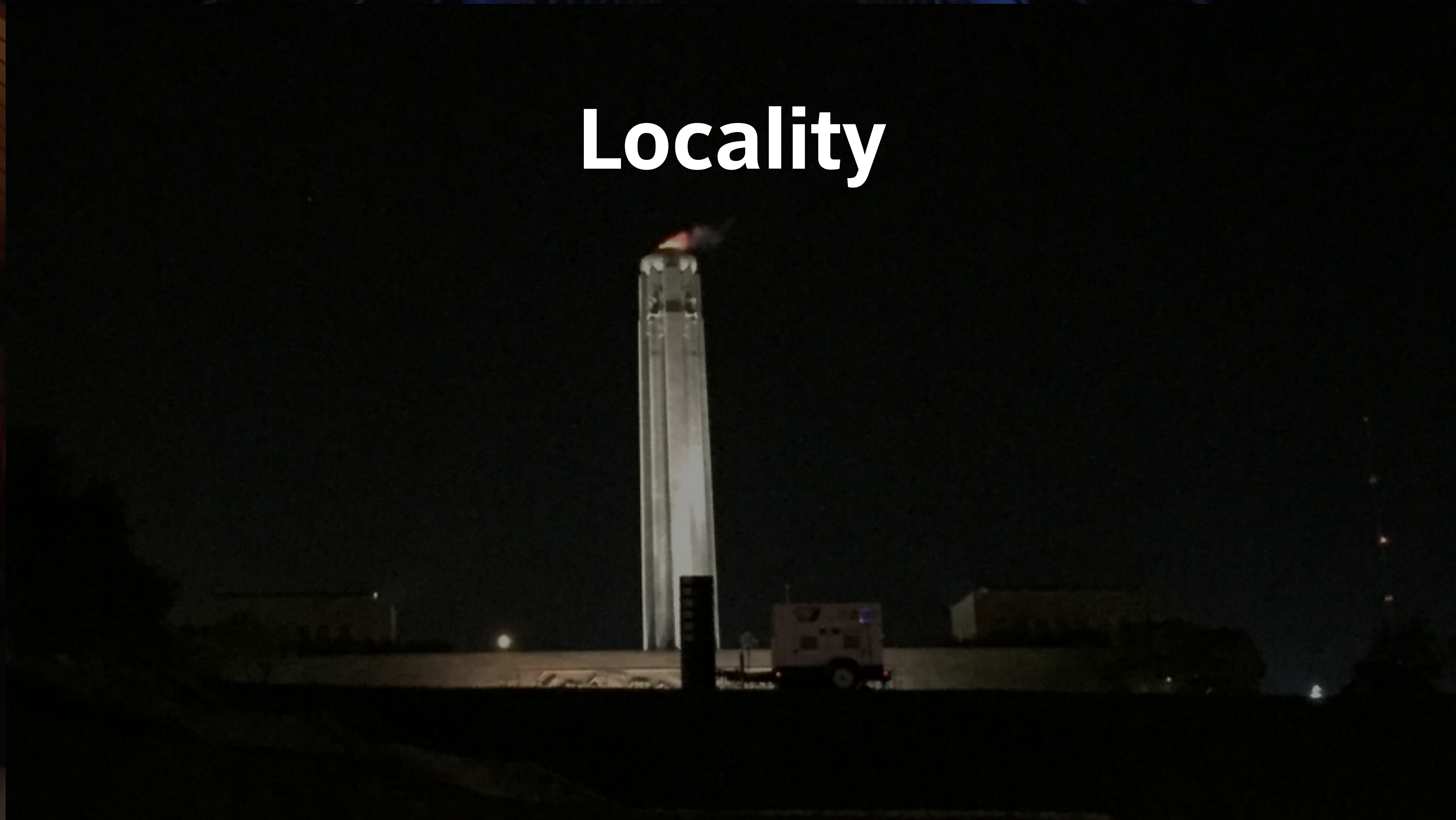
independence

Fault Tolerance

detection

recovery

redundancy



Locality

Work distribution

parallelism

concurrency

independence

Fault Tolerance

detection

recovery

redundancy

Locality

regional presence

Work distribution

parallelism

concurrency

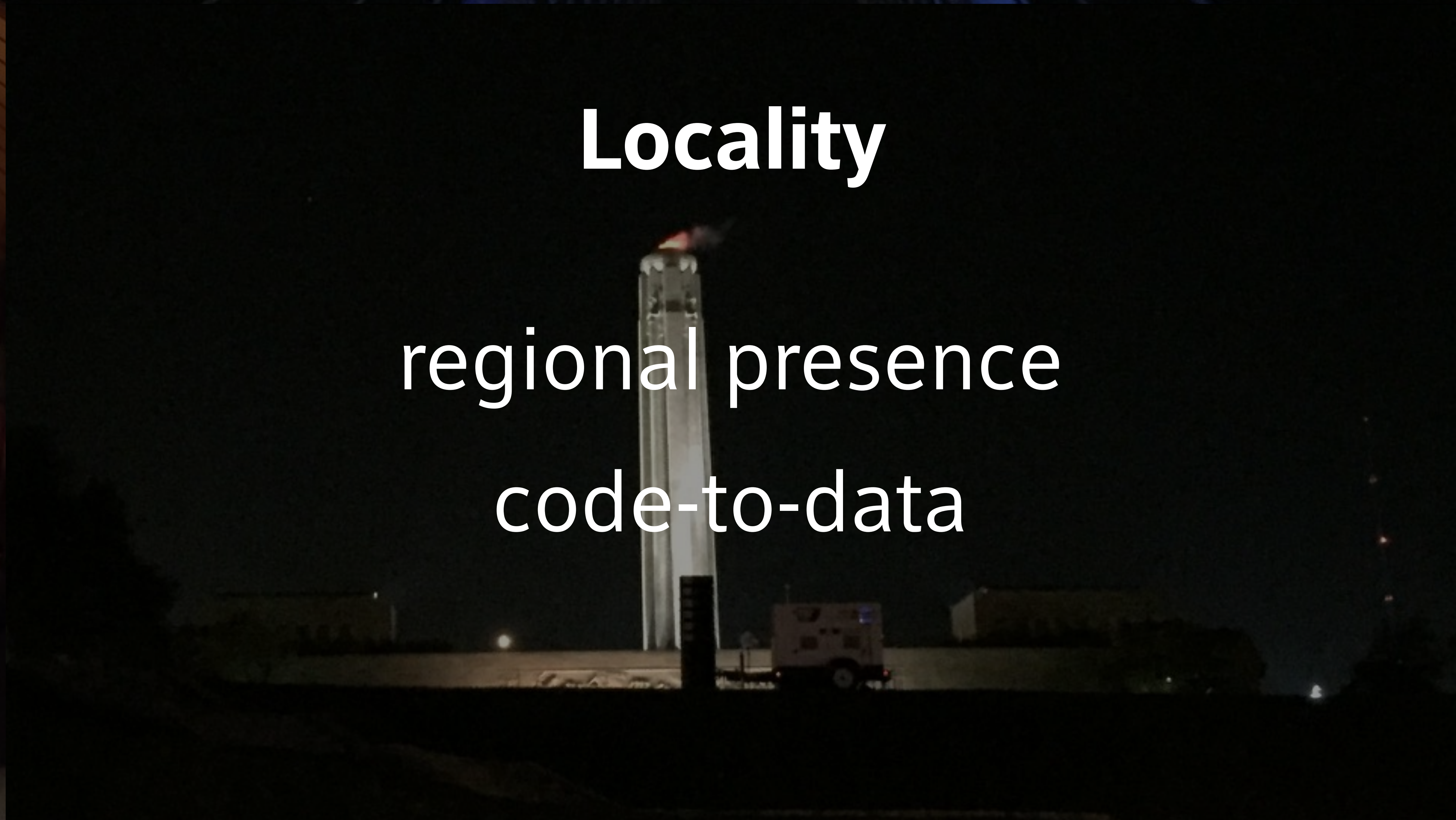
independence

Fault Tolerance

detection

recovery

redundancy



Locality

regional presence

code-to-data

WHY **NOT** PEER-TO-PEER SYSTEMS?

WHY **NOT** PEER-TO-PEER SYSTEMS?

Now you've got **N** problems

WHY **NOT** PEER-TO-PEER SYSTEMS?

Now you've got **N** problems

N = 8 probably

COR JESU SACRATISSIMUM
DEPUIS PLUS DE 125 ANS ICI JOUR ET NUIT
FOR OVER 125 YEARS HERE NIGHT AND DAY SOMEONE IS PRAYING TO THE LORD
SANCTUAIRE DE L'ADORATION

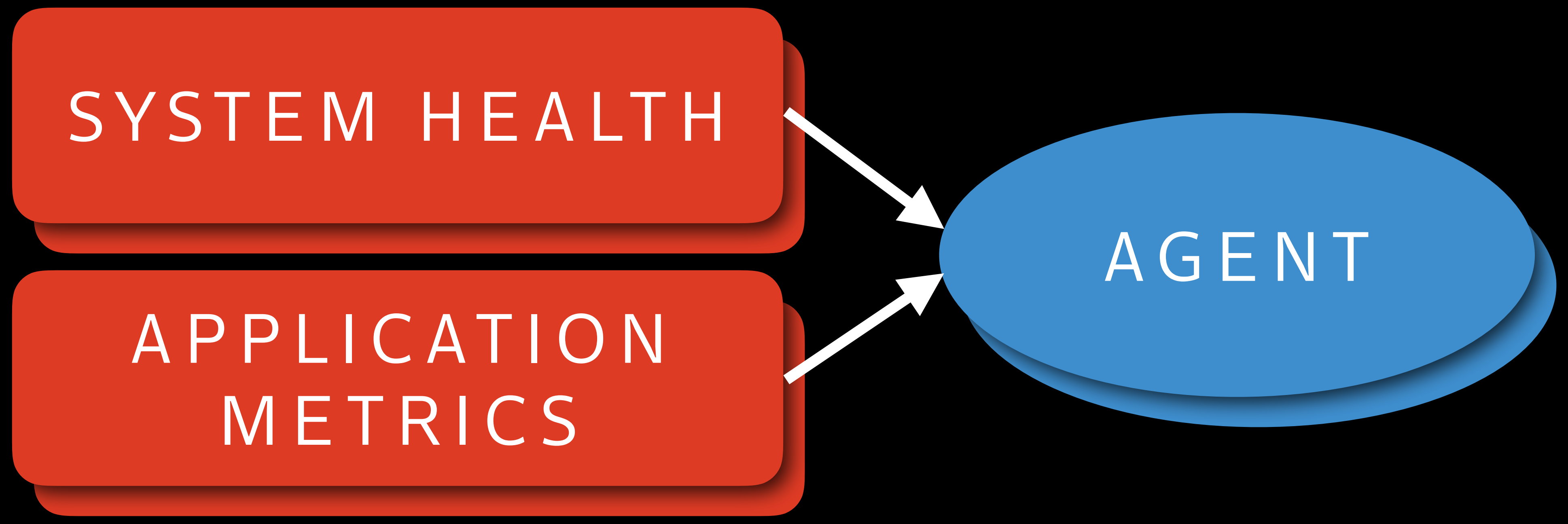
WHAT ARE WE BUILDING?

ARGUS OPERATIONAL VISIBILITY PROJECT

SYSTEM HEALTH

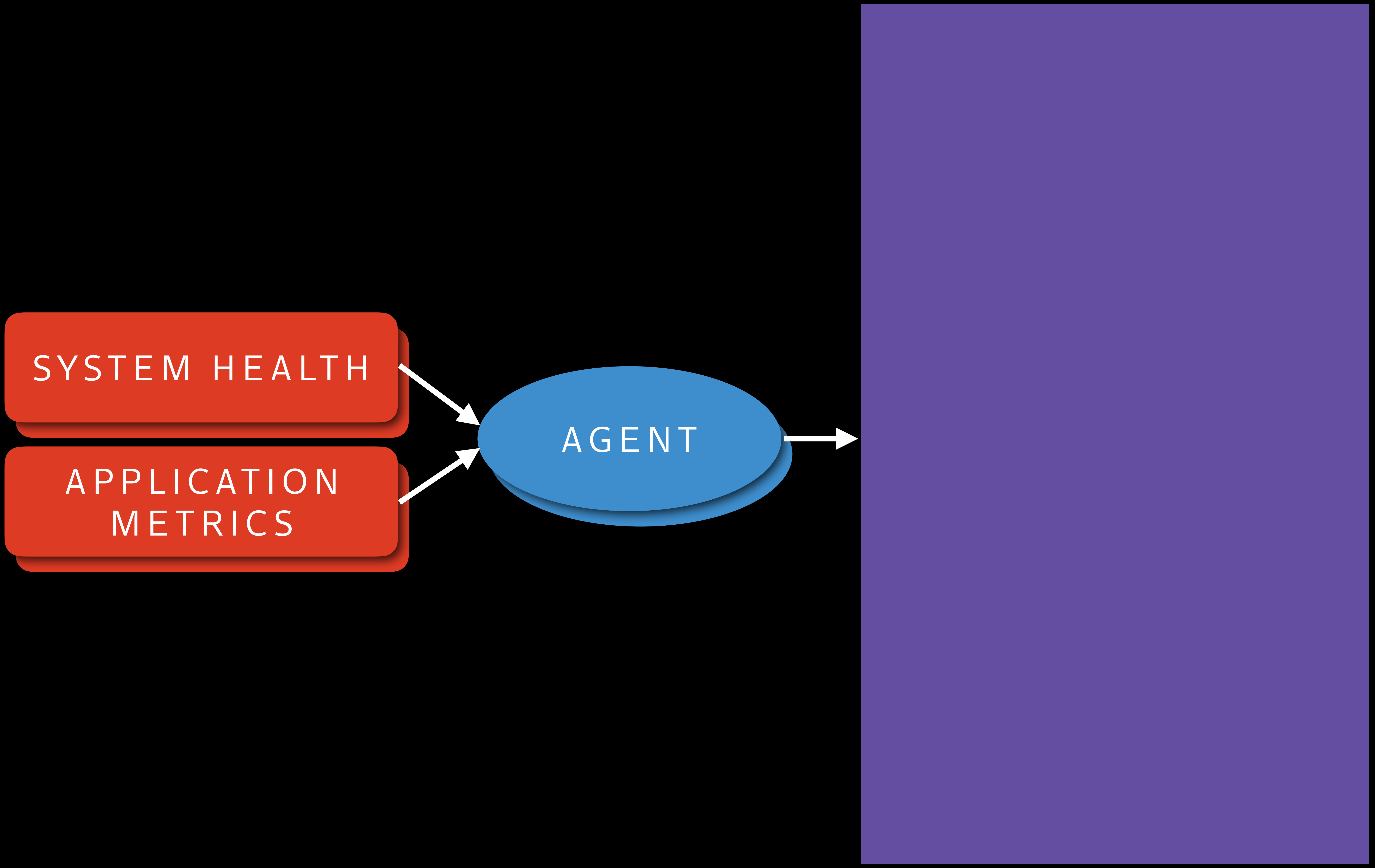
APPLICATION METRICS

ARGUS OPERATIONAL VISIBILITY PROJECT



Emoji provided free by Emoji One

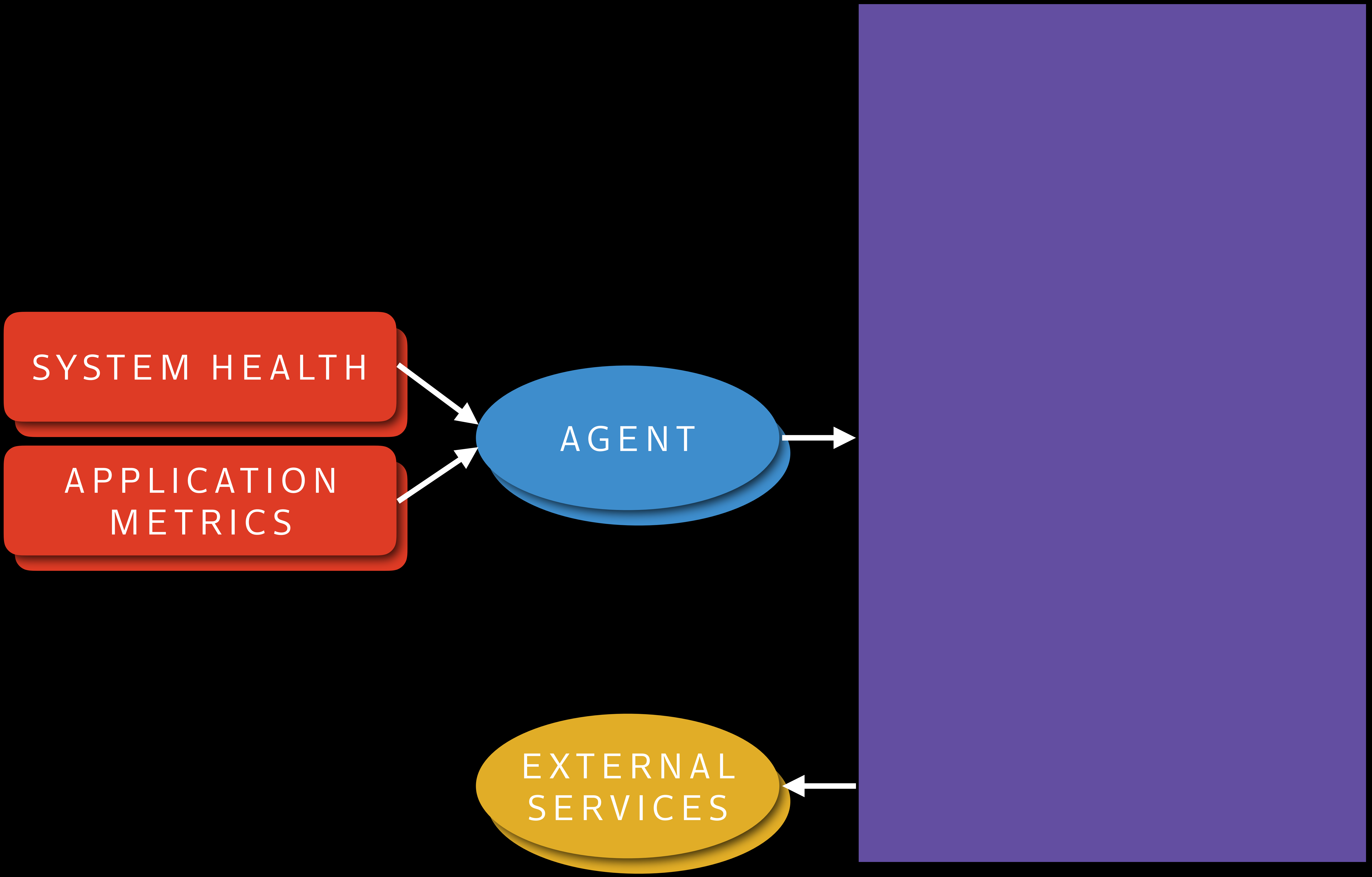
ARGUS OPERATIONAL VISIBILITY PROJECT



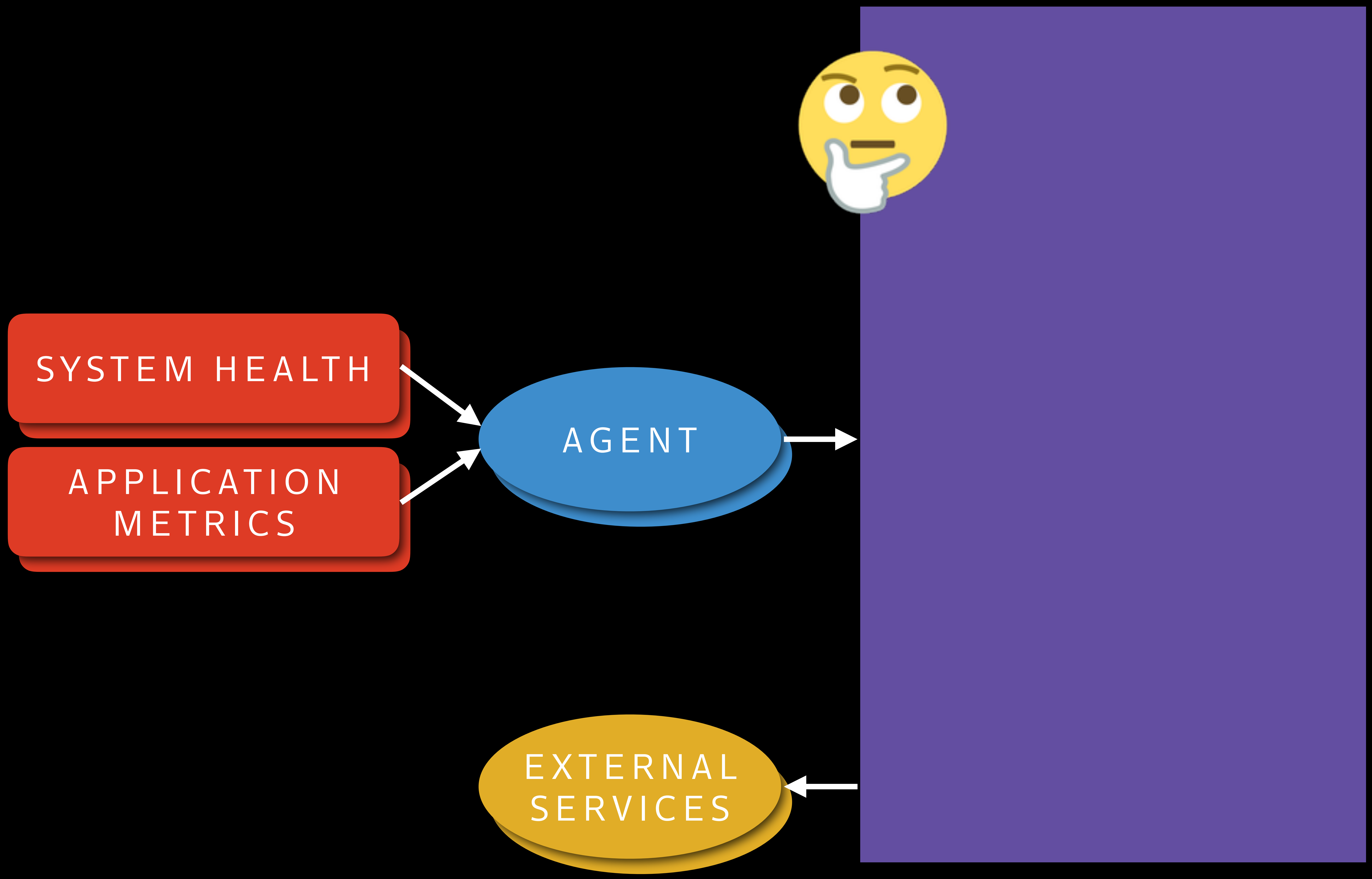
Emoji provided free by Emoji One

COMCAST

ARGUS OPERATIONAL VISIBILITY PROJECT

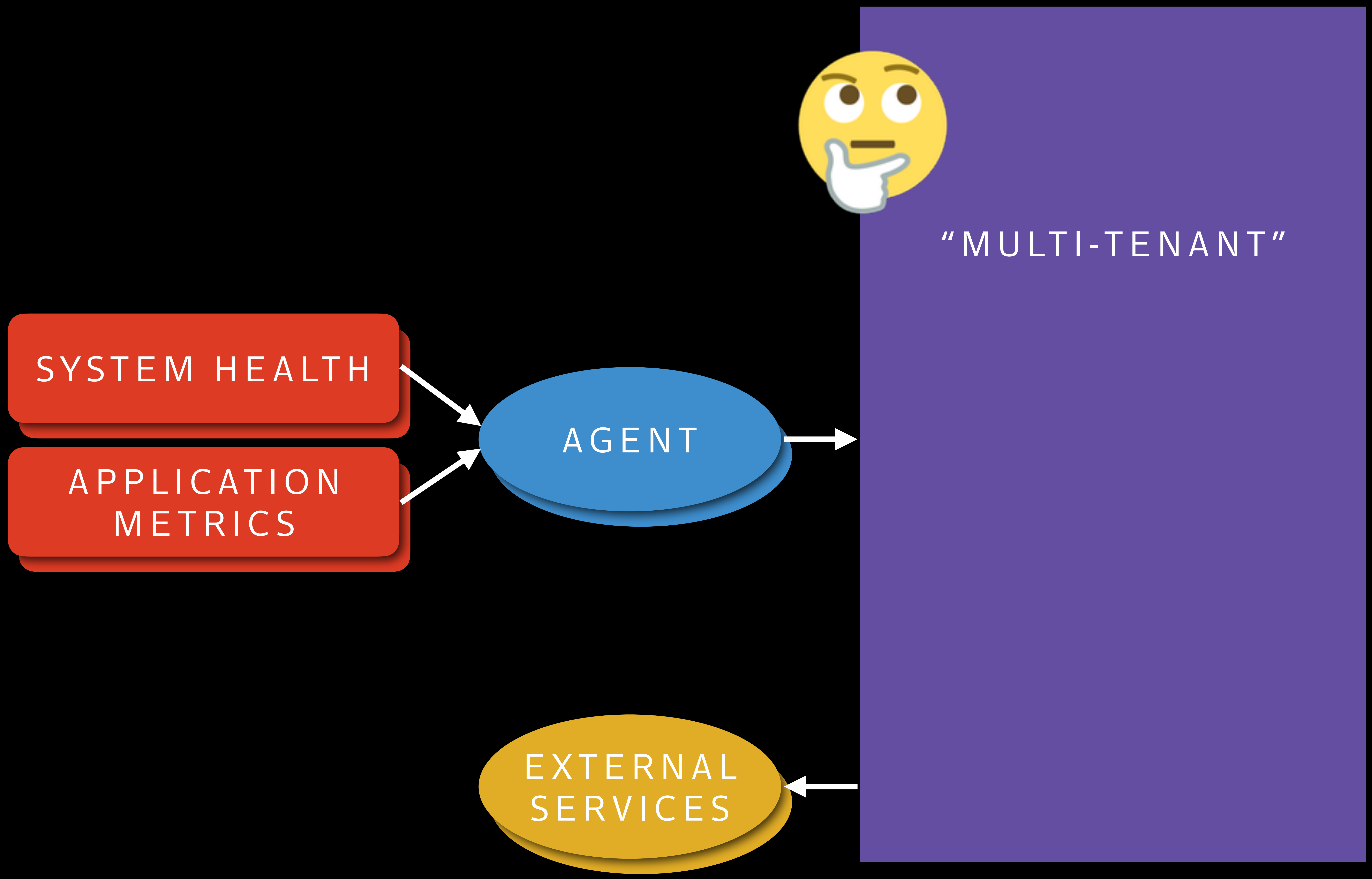


ARGUS OPERATIONAL VISIBILITY PROJECT



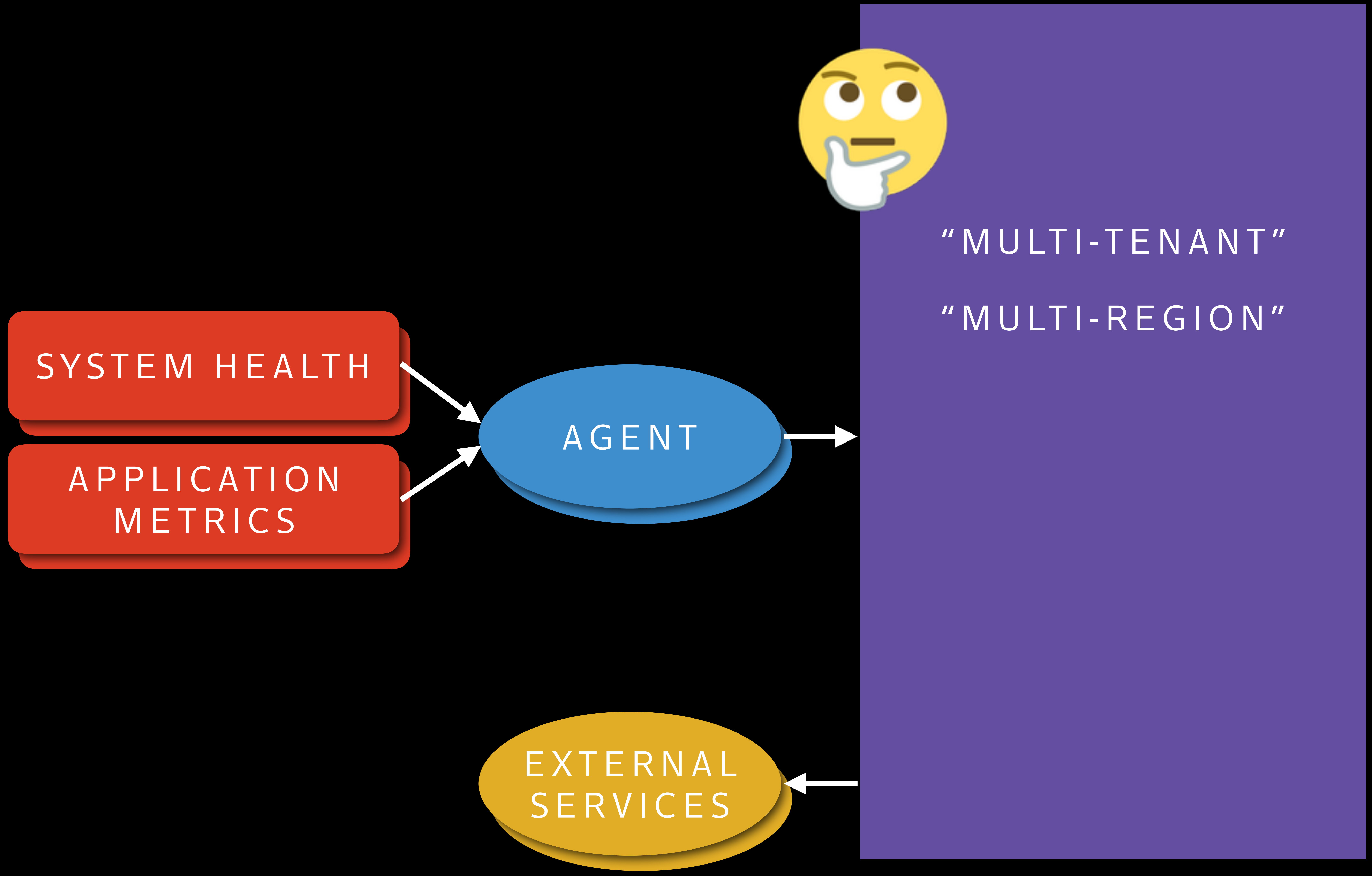
Emoji provided free by Emoji One

ARGUS OPERATIONAL VISIBILITY PROJECT



Emoji provided free by Emoji One

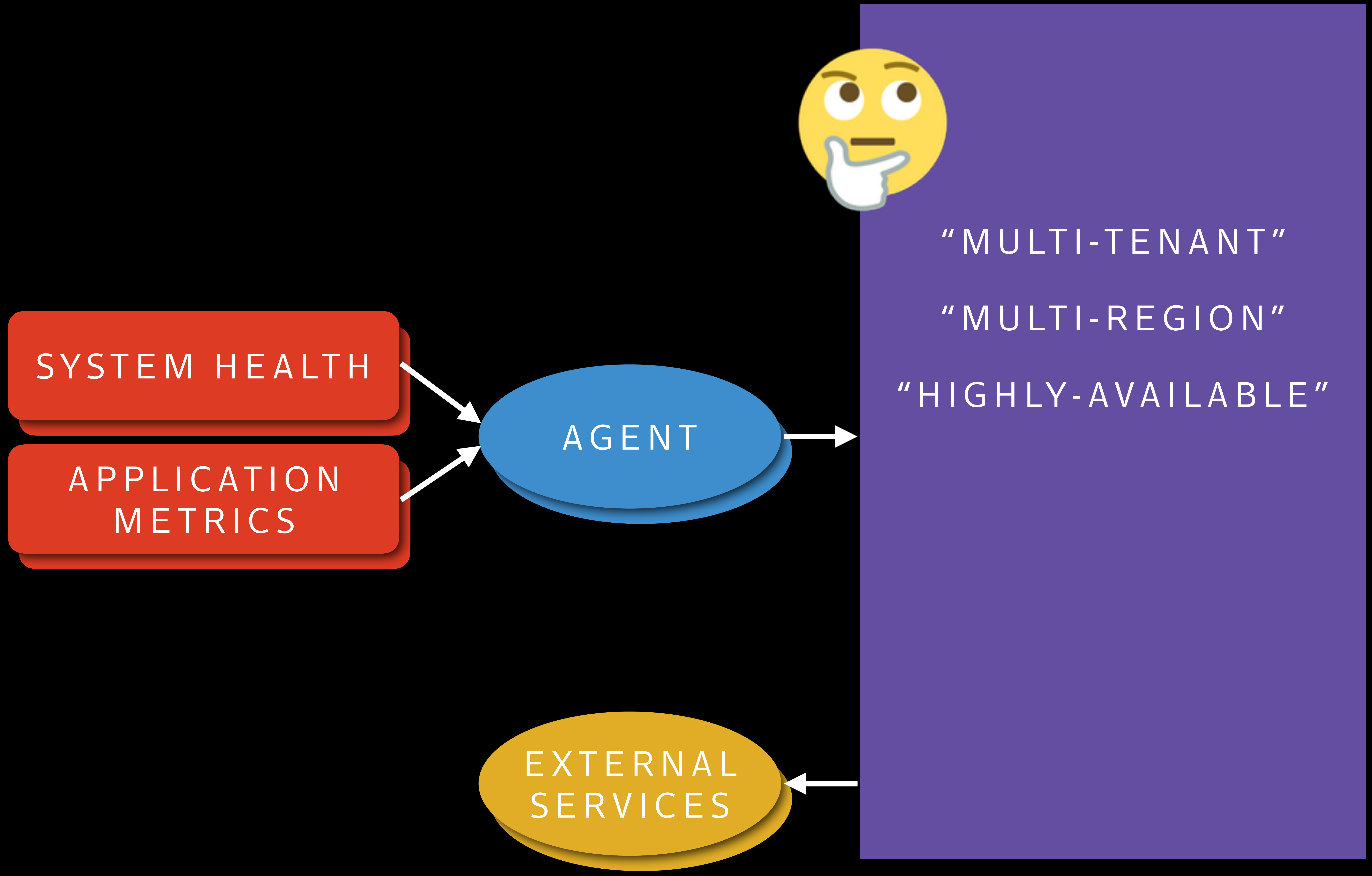
ARGUS OPERATIONAL VISIBILITY PROJECT



Emoji provided free by Emoji One

COMCAST

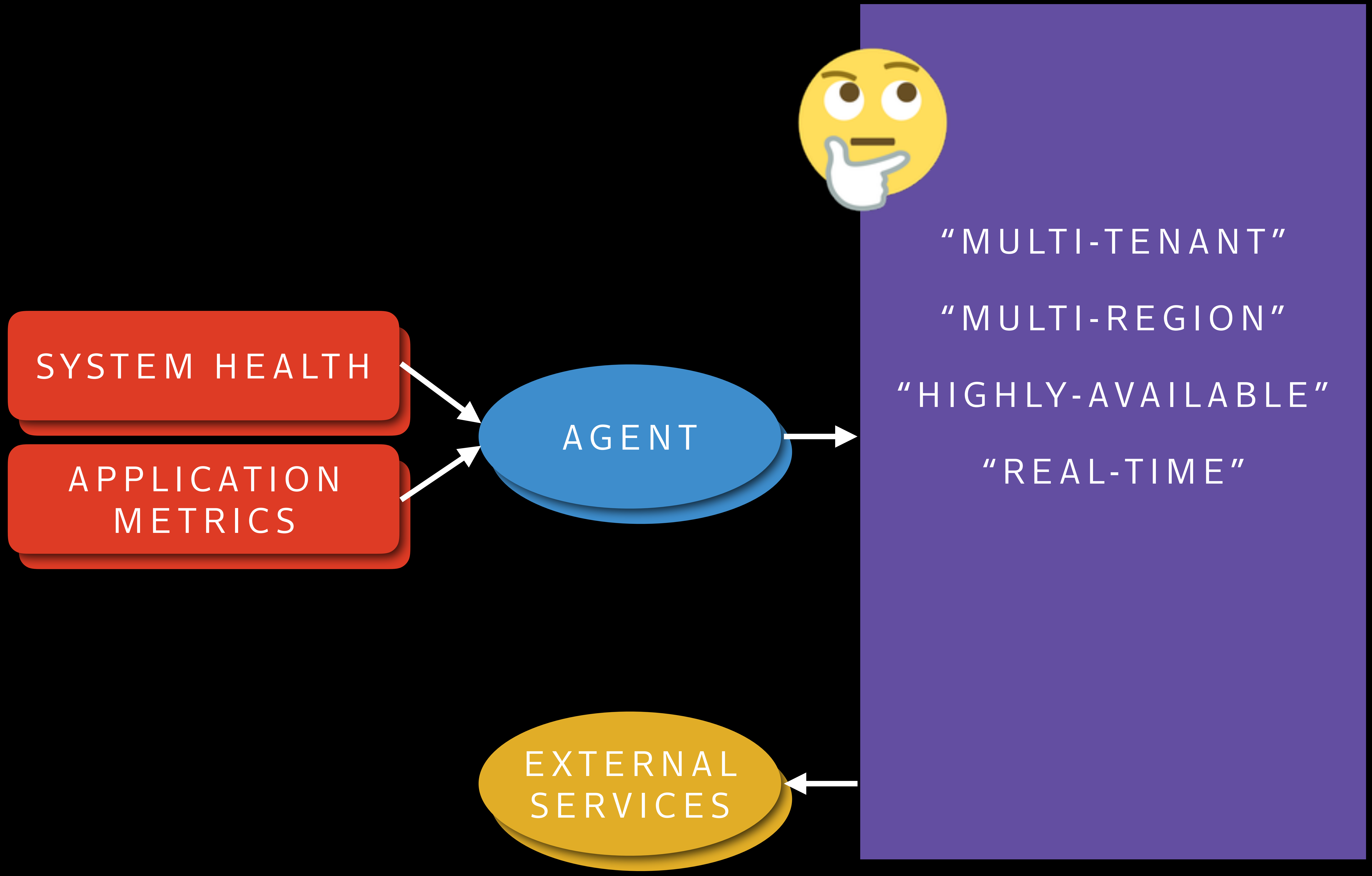
ARGUS OPERATIONAL VISIBILITY PROJECT



Emoji provided free by Emoji One

COMCAST

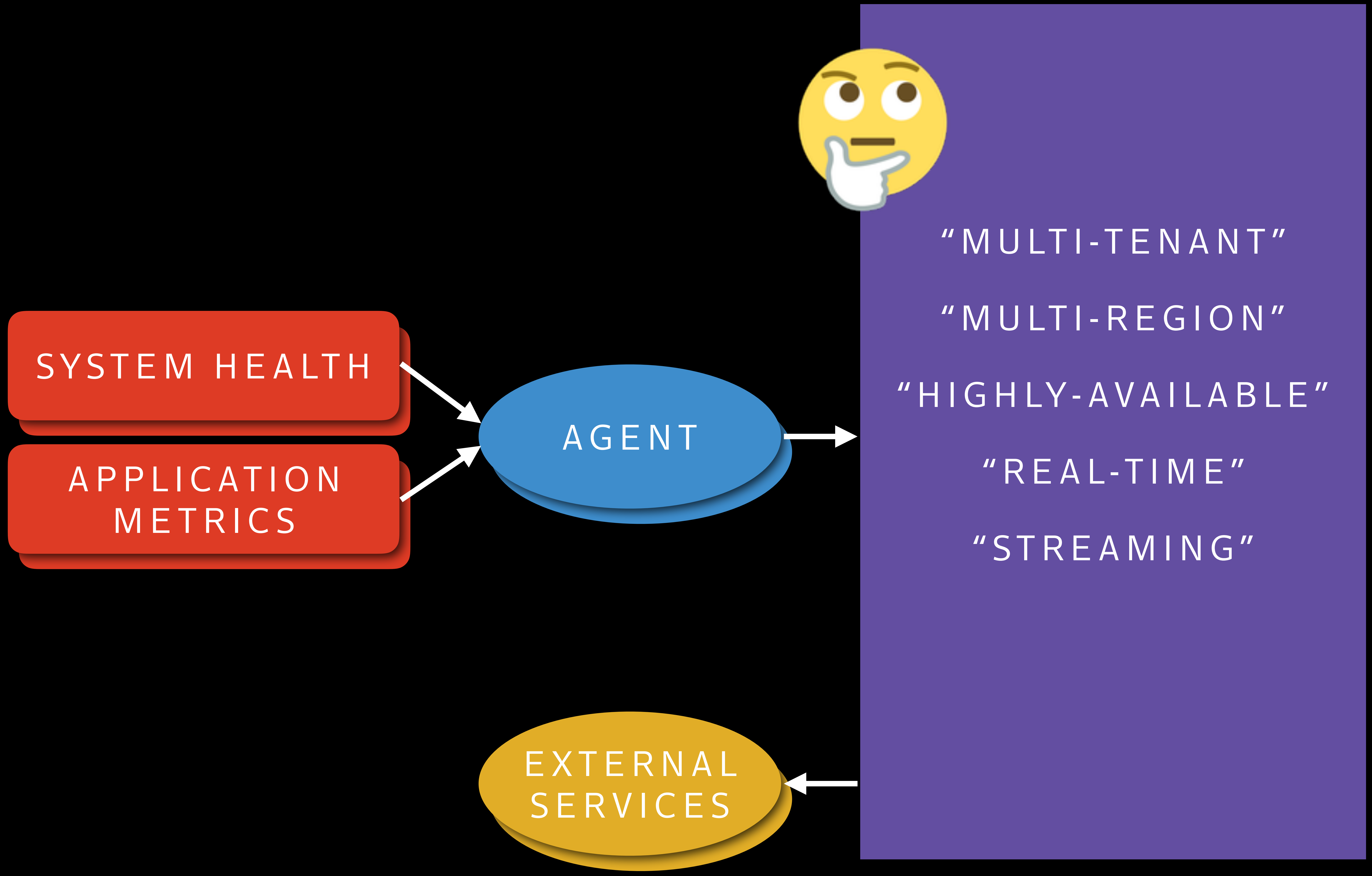
ARGUS OPERATIONAL VISIBILITY PROJECT



Emoji provided free by Emoji One

COMCAST

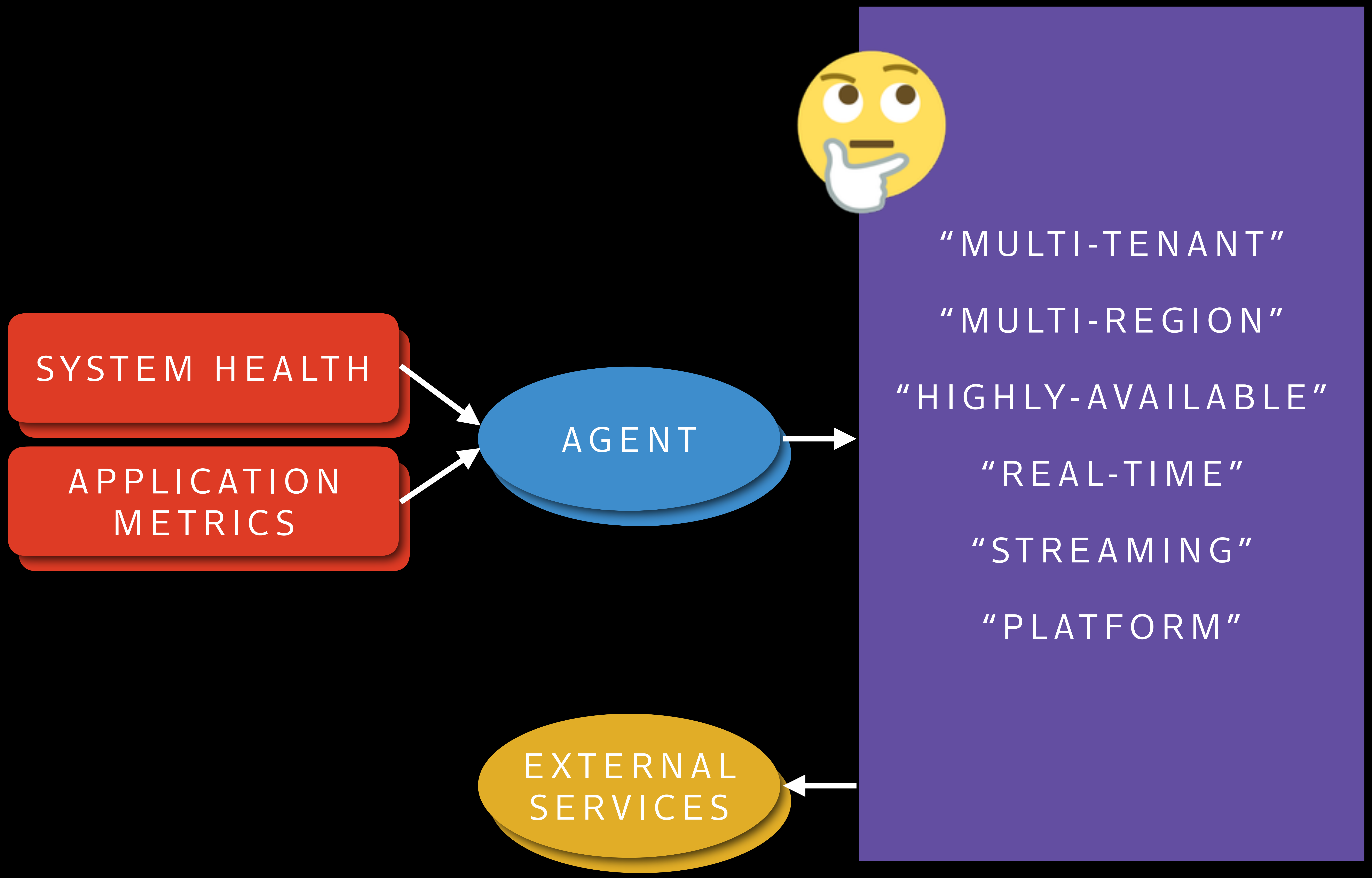
ARGUS OPERATIONAL VISIBILITY PROJECT



Emoji provided free by Emoji One

COMCAST

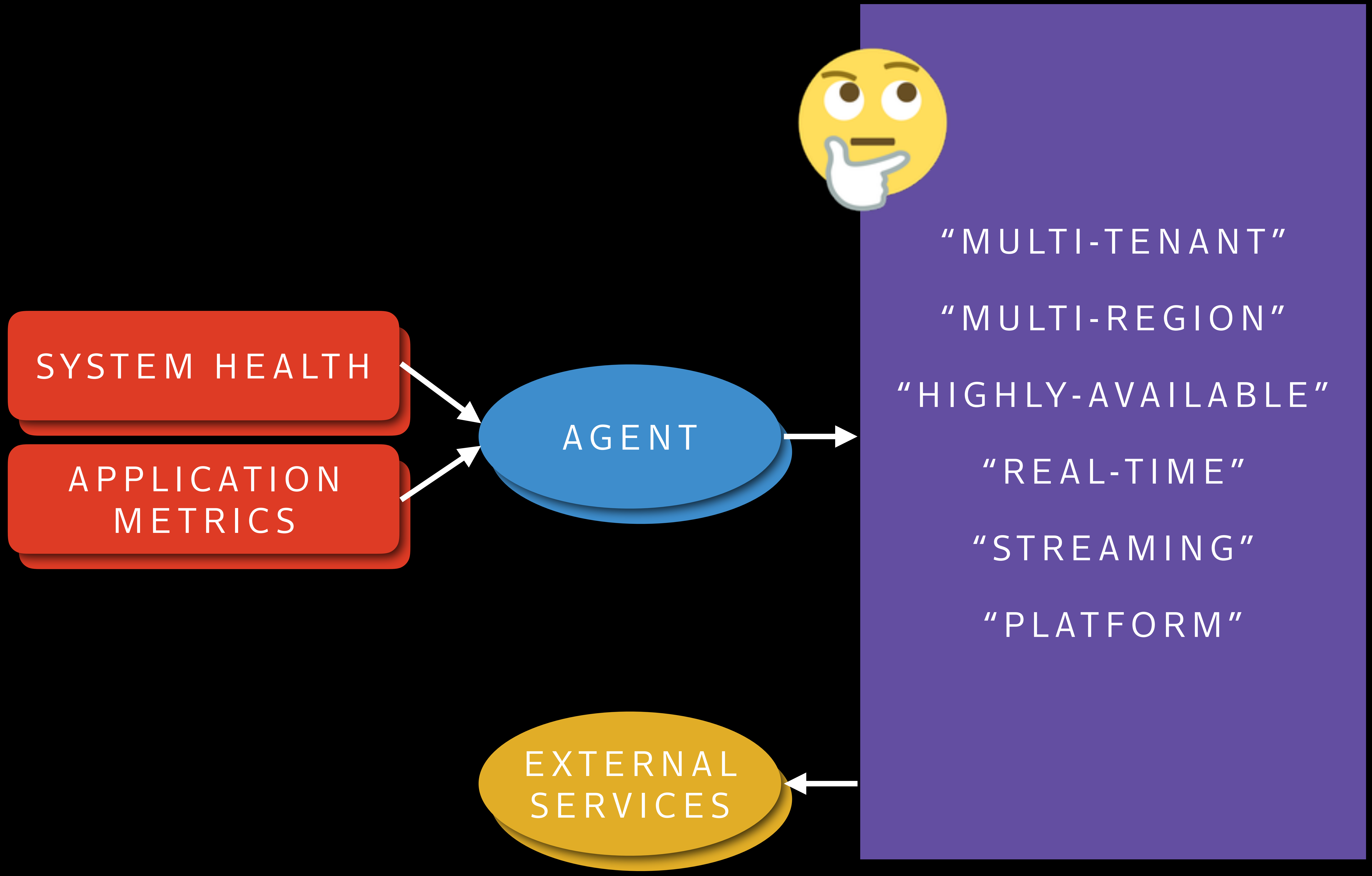
ARGUS OPERATIONAL VISIBILITY PROJECT



Emoji provided free by Emoji One

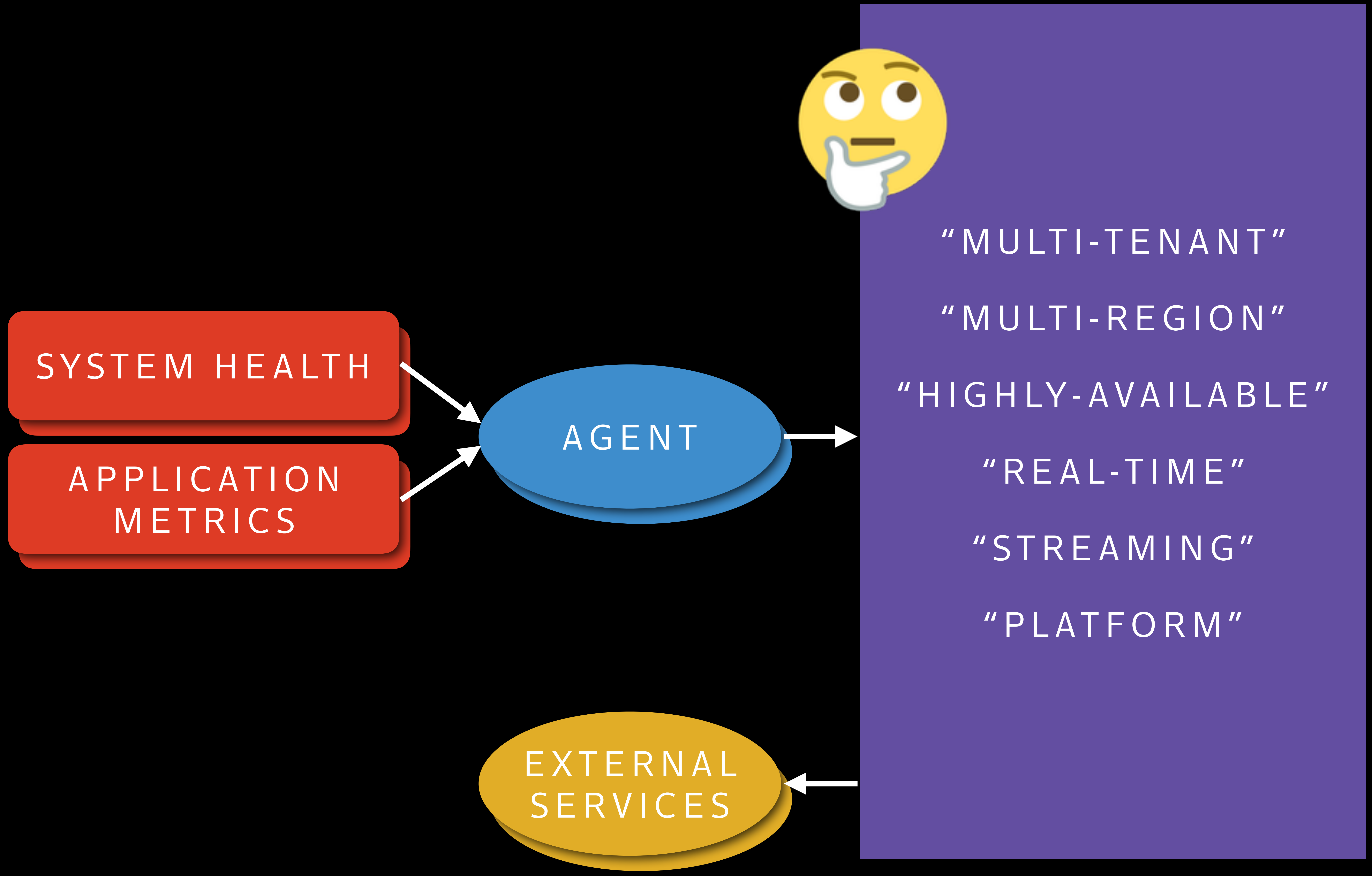
COMCAST

ARGUS OPERATIONAL VISIBILITY PROJECT



➡ Work distribution

ARGUS OPERATIONAL VISIBILITY PROJECT

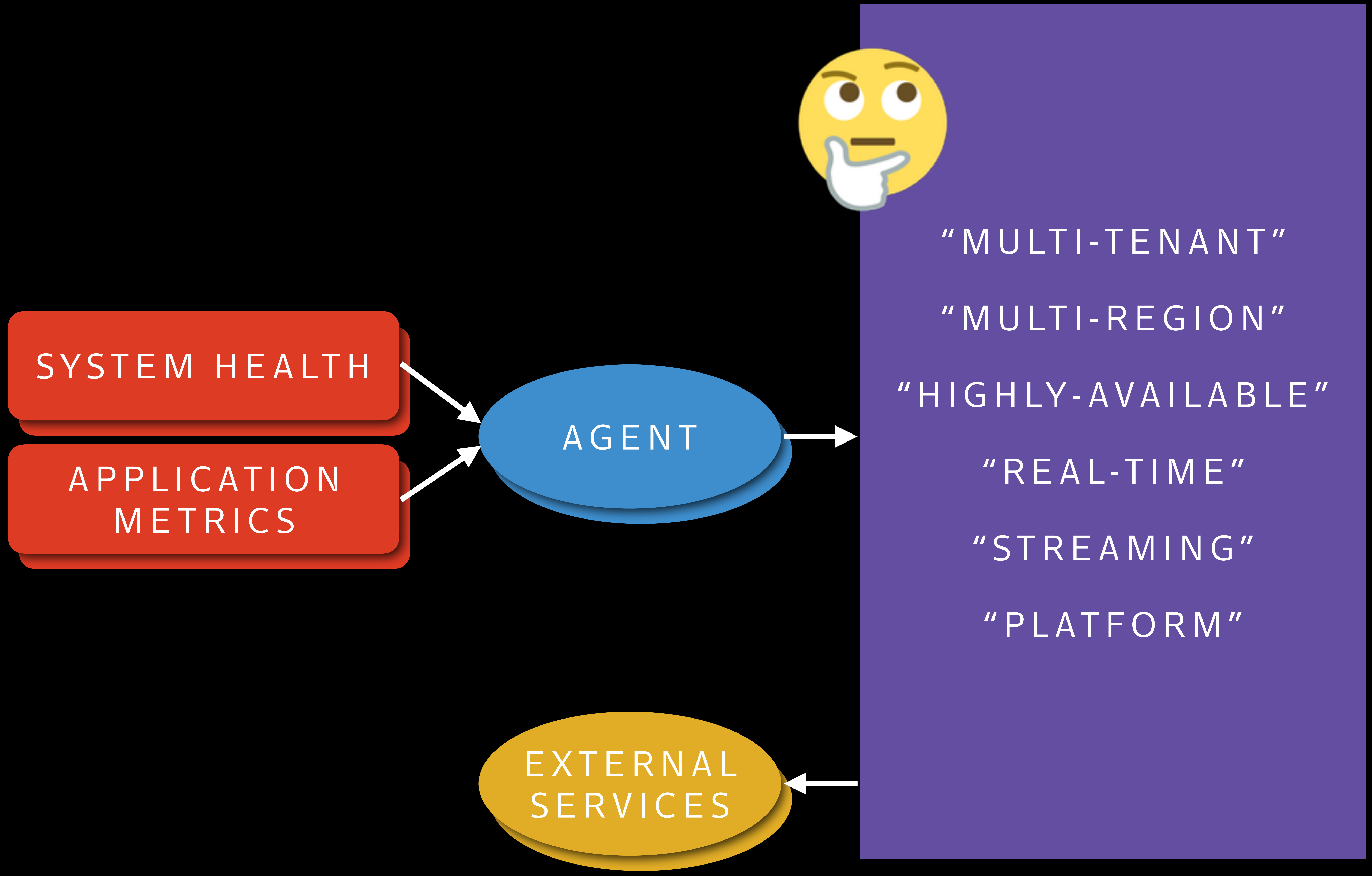


- ➡ Work distribution
- ➡ Fault-tolerance

Emoji provided free by Emoji One

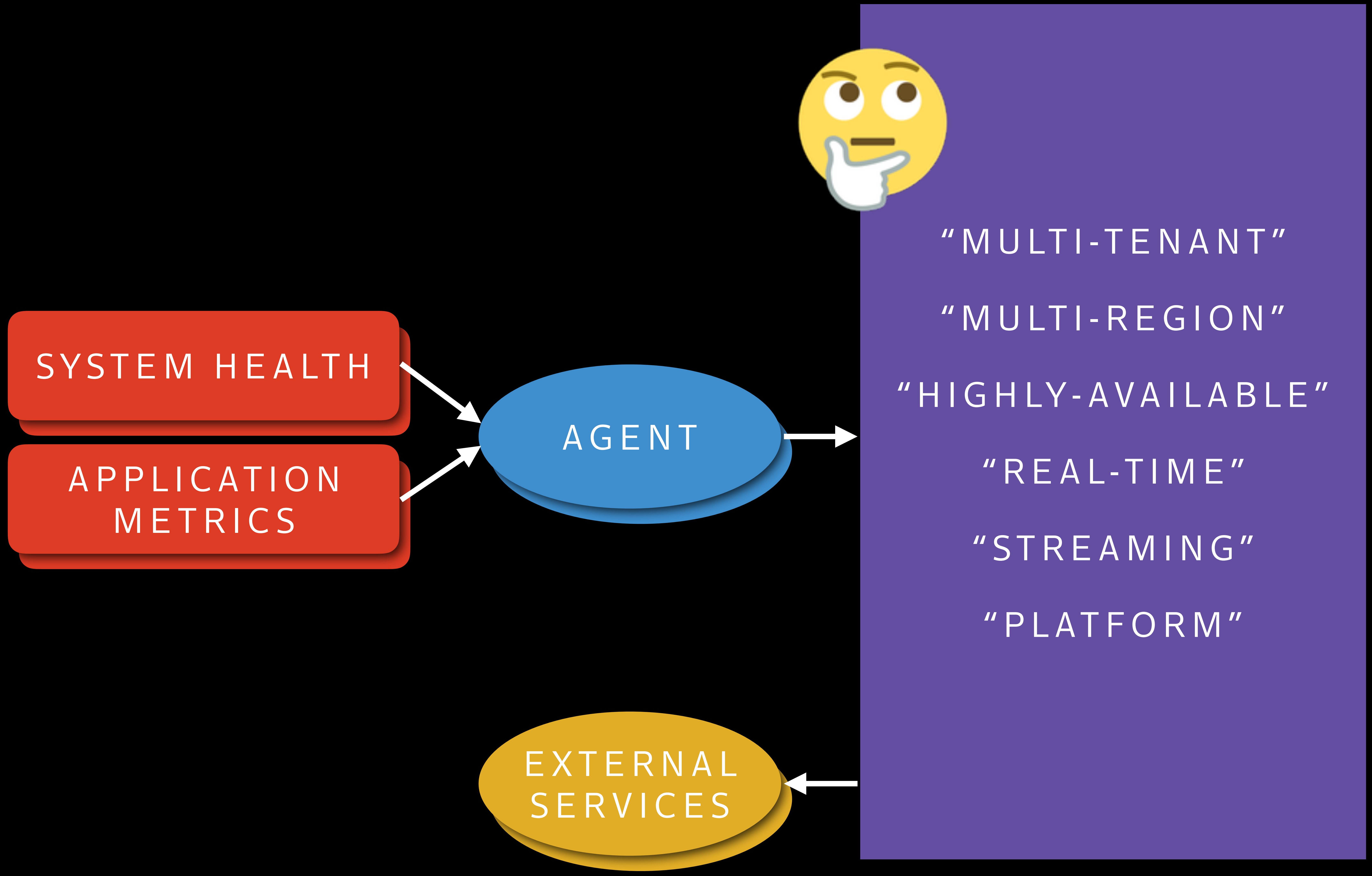
COMCAST

ARGUS OPERATIONAL VISIBILITY PROJECT



- Work distribution
- Fault-tolerance
- Locality

ARGUS OPERATIONAL VISIBILITY PROJECT

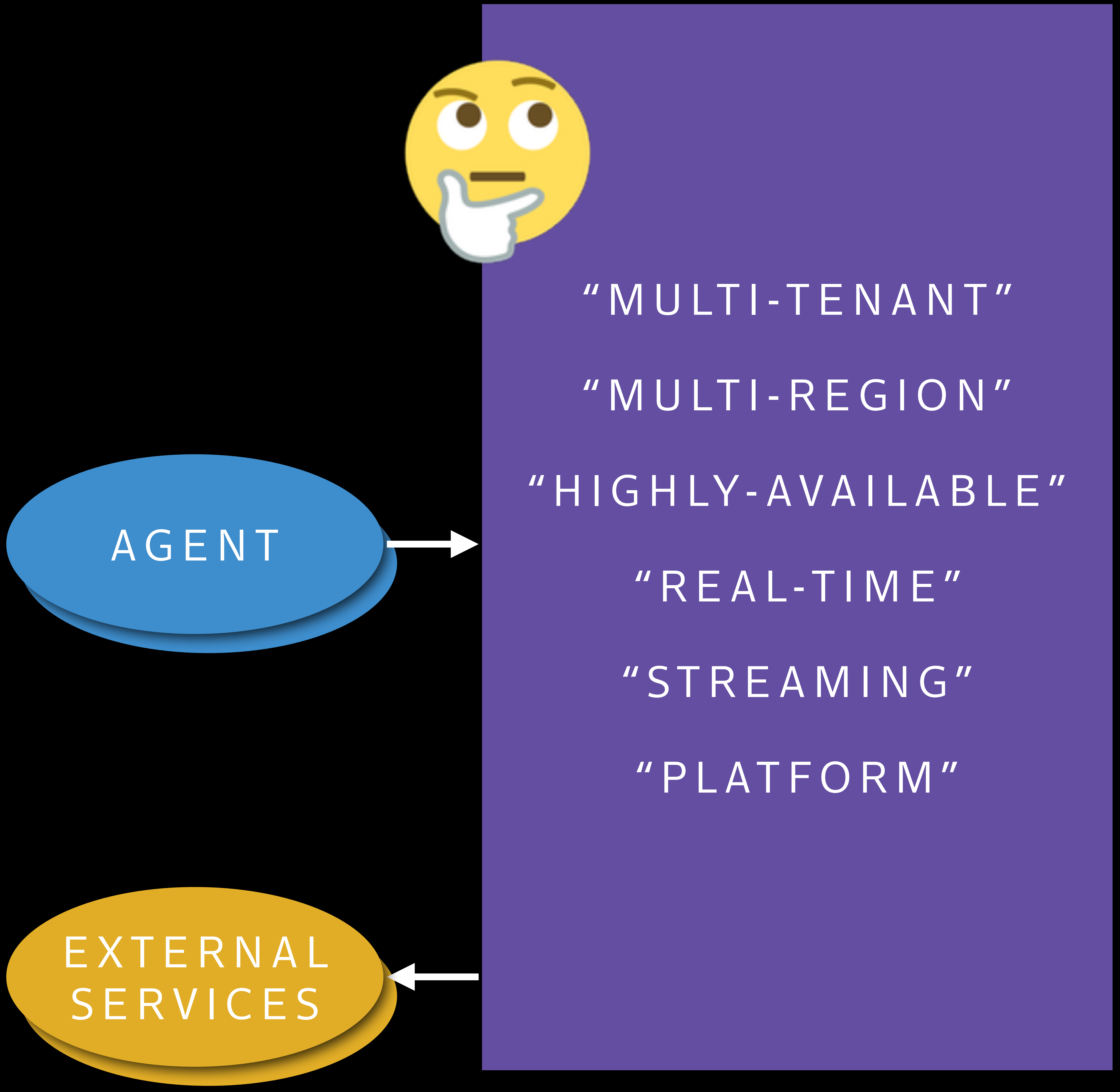


- ➡ Work distribution
- ➡ Fault-tolerance
- ➡ Locality
- ✓ Peer to Peer!

Emoji provided free by Emoji One

COMCAST

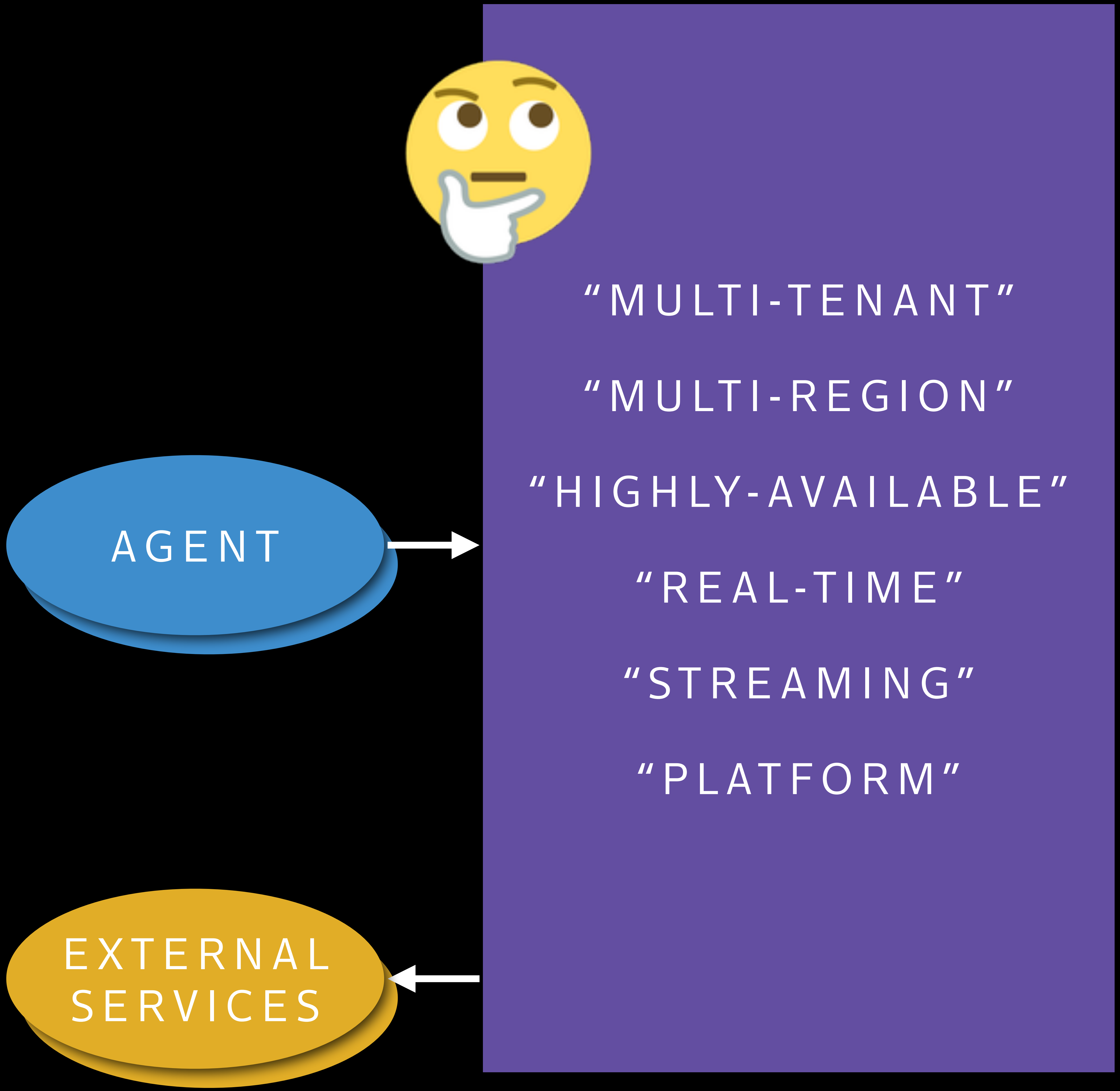
ARGUS OPERATIONAL VISIBILITY PROJECT



Emoji provided free by Emoji One

COMCAST

ARGUS OPERATIONAL VISIBILITY PROJECT

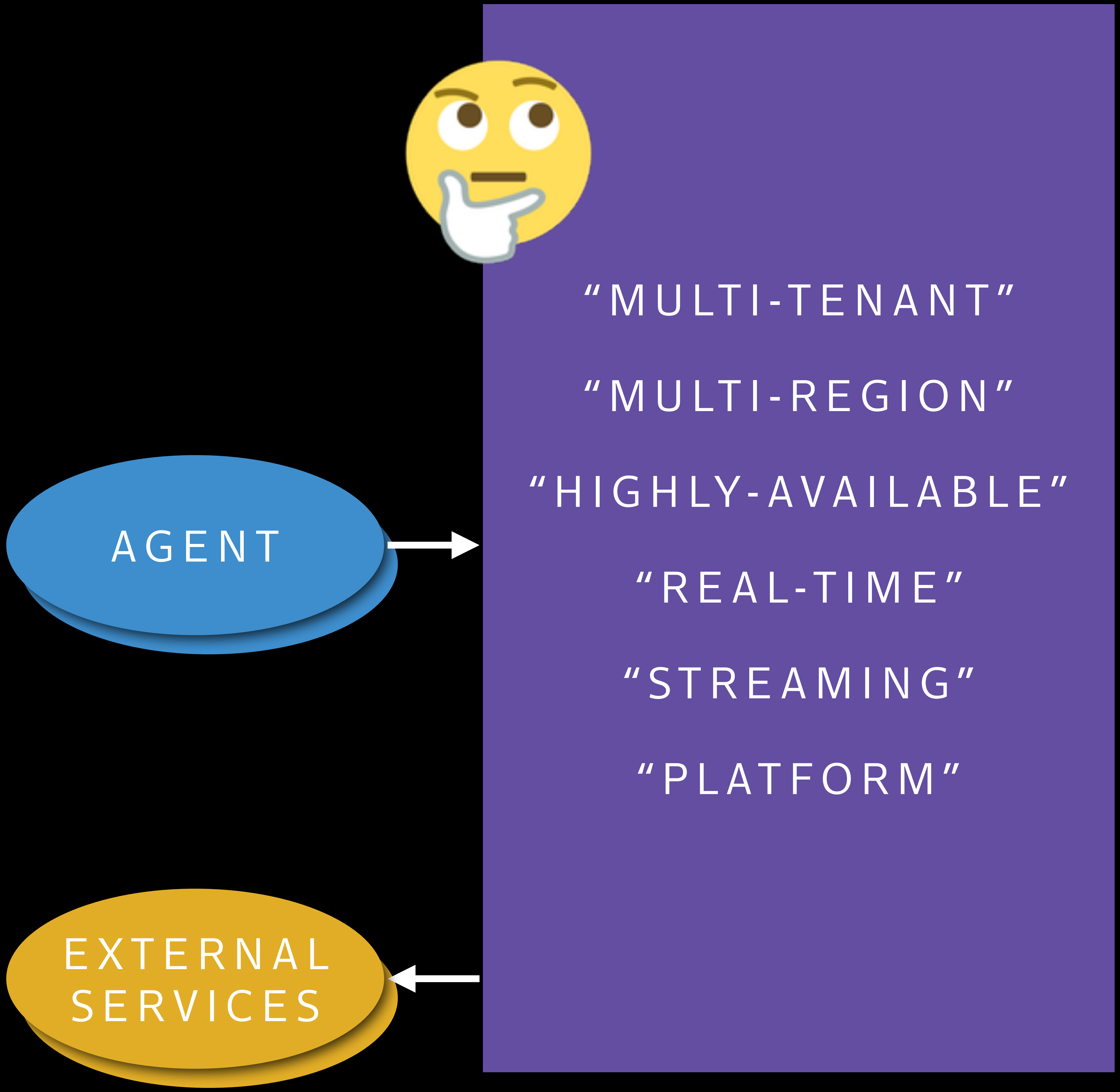


How do cluster nodes find each other?

Emoji provided free by Emoji One

COMCAST

ARGUS OPERATIONAL VISIBILITY PROJECT



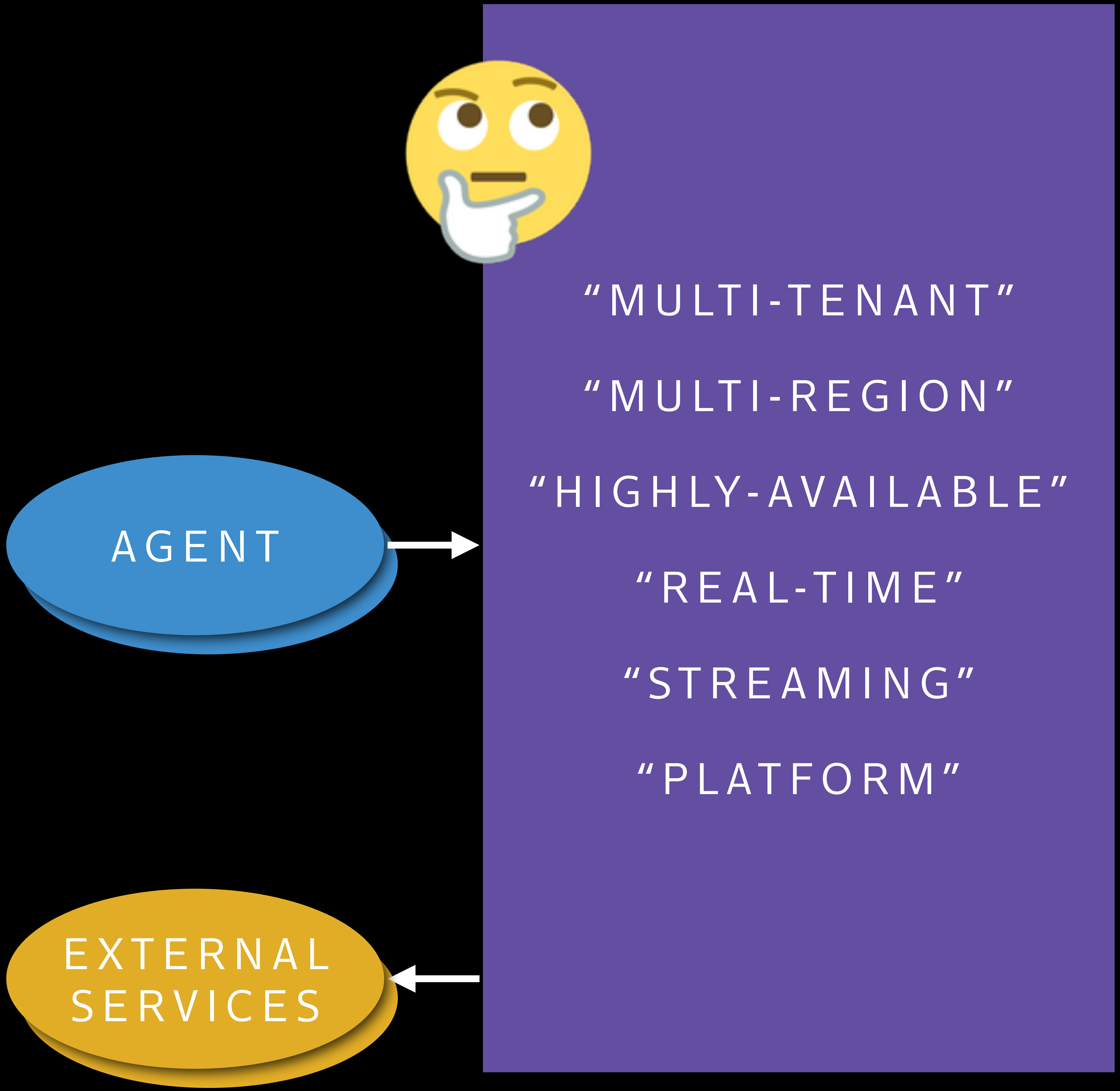
How do cluster nodes find each other?

Distribute code and configuration?

Emoji provided free by Emoji One

COMCAST

ARGUS OPERATIONAL VISIBILITY PROJECT



How do cluster nodes find each other?

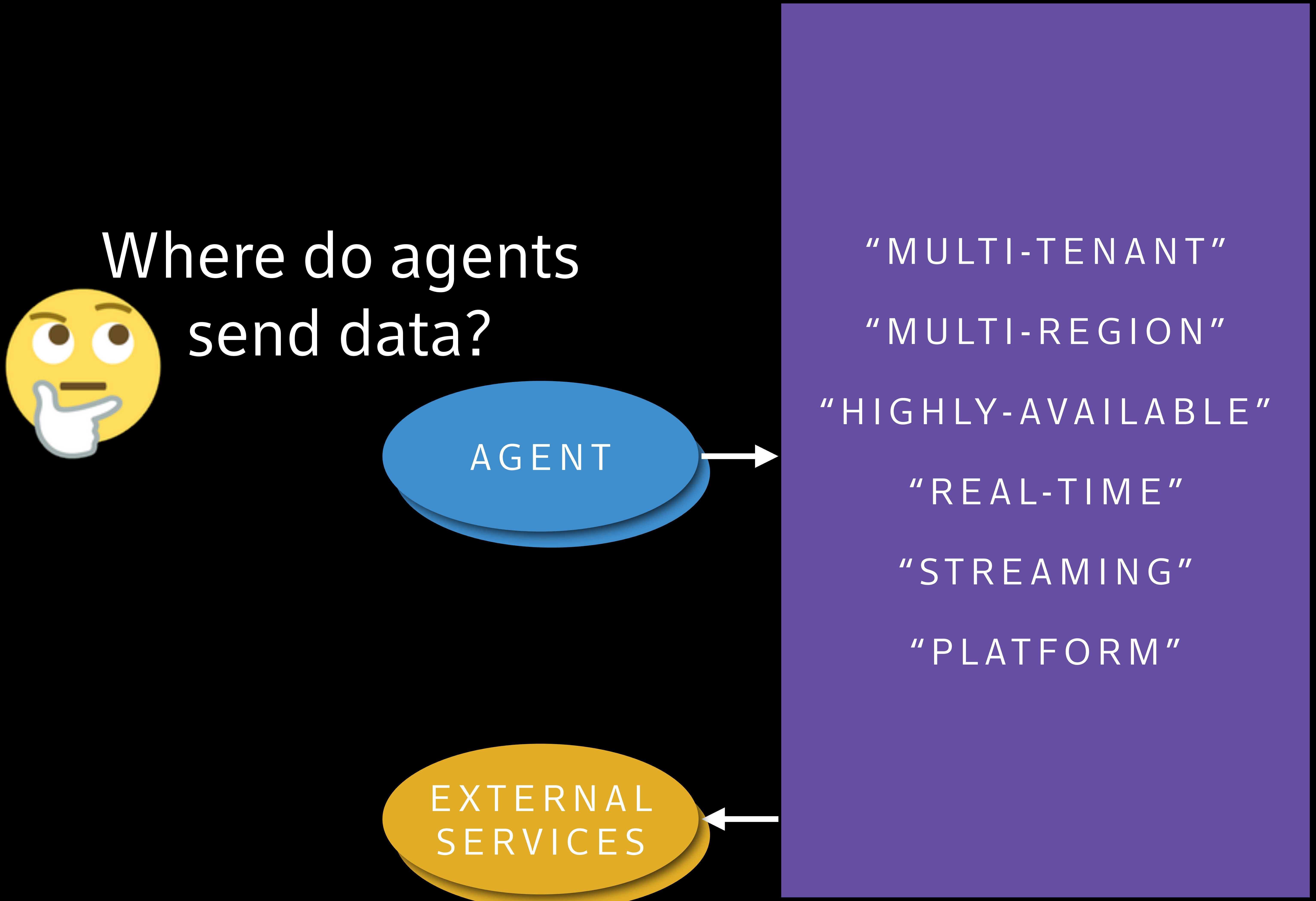
Distribute code and configuration?

Know what happened when?

Emoji provided free by Emoji One

COMCAST

ARGUS OPERATIONAL VISIBILITY PROJECT



How do cluster nodes find each other?

Distribute code and configuration?

Know what happened when?

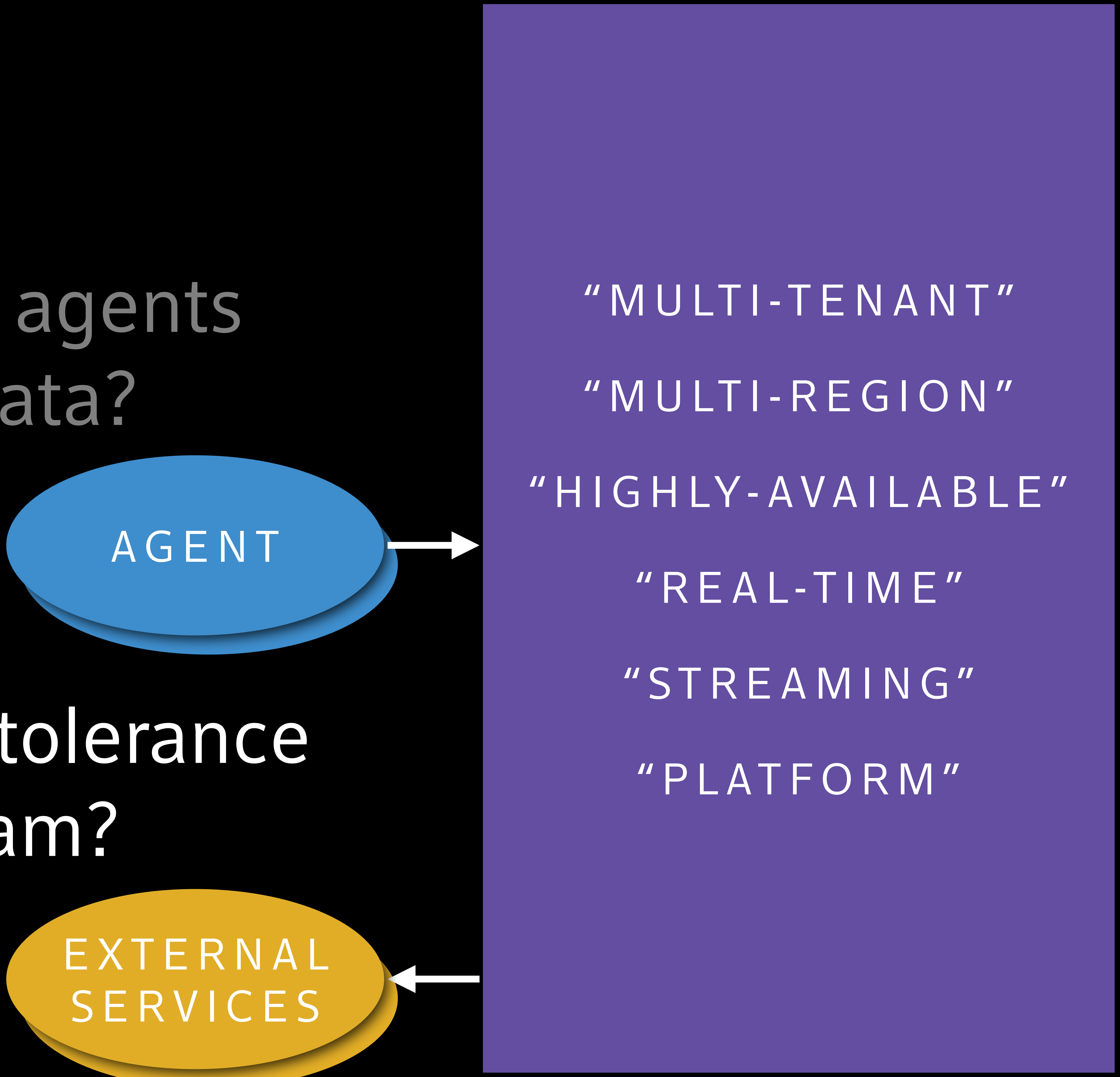
Emoji provided free by Emoji One

COMCAST

ARGUS OPERATIONAL VISIBILITY PROJECT

Where do agents send data?

How to get fault-tolerance without spam?



How do cluster nodes find each other?

Distribute code and configuration?

Know what happened when?

Emoji provided free by Emoji One

COMCAST

ARGUS OPERATIONAL VISIBILITY PROJECT

COMCAST

ARGUS OPERATIONAL VISIBILITY PROJECT

➡ Cluster membership and discovery

COMCAST

ARGUS OPERATIONAL VISIBILITY PROJECT

- ➡ Cluster membership and discovery
- ➡ Code and configuration dissemination

COMCAST

ARGUS OPERATIONAL VISIBILITY PROJECT

- ➡ Cluster membership and discovery
- ➡ Code and configuration dissemination
- ➡ Relative and convergent time

MEMBERSHIP PROTOCOLS

JUST RUB SOME CONSENSUS ON IT

WHY NOT ZOOKEEPER/CONSUL/ETCD?

MEMBERSHIP: DESIRABLE PROPERTIES

MEMBERSHIP: DESIRABLE PROPERTIES

- ➡ Connectedness

COMCAST

MEMBERSHIP: DESIRABLE PROPERTIES

- Connectedness
- Balance

COMCAST

MEMBERSHIP: DESIRABLE PROPERTIES

- ➡ Connectedness
- ➡ Balance
- ➡ Short path-length

COMCAST

MEMBERSHIP: DESIRABLE PROPERTIES

- Connectedness
- Balance
- Short path-length
- Low clustering

COMCAST

MEMBERSHIP: DESIRABLE PROPERTIES

- Connectedness
- Balance
- Short path-length
- Low clustering
- Scalability

COMCAST

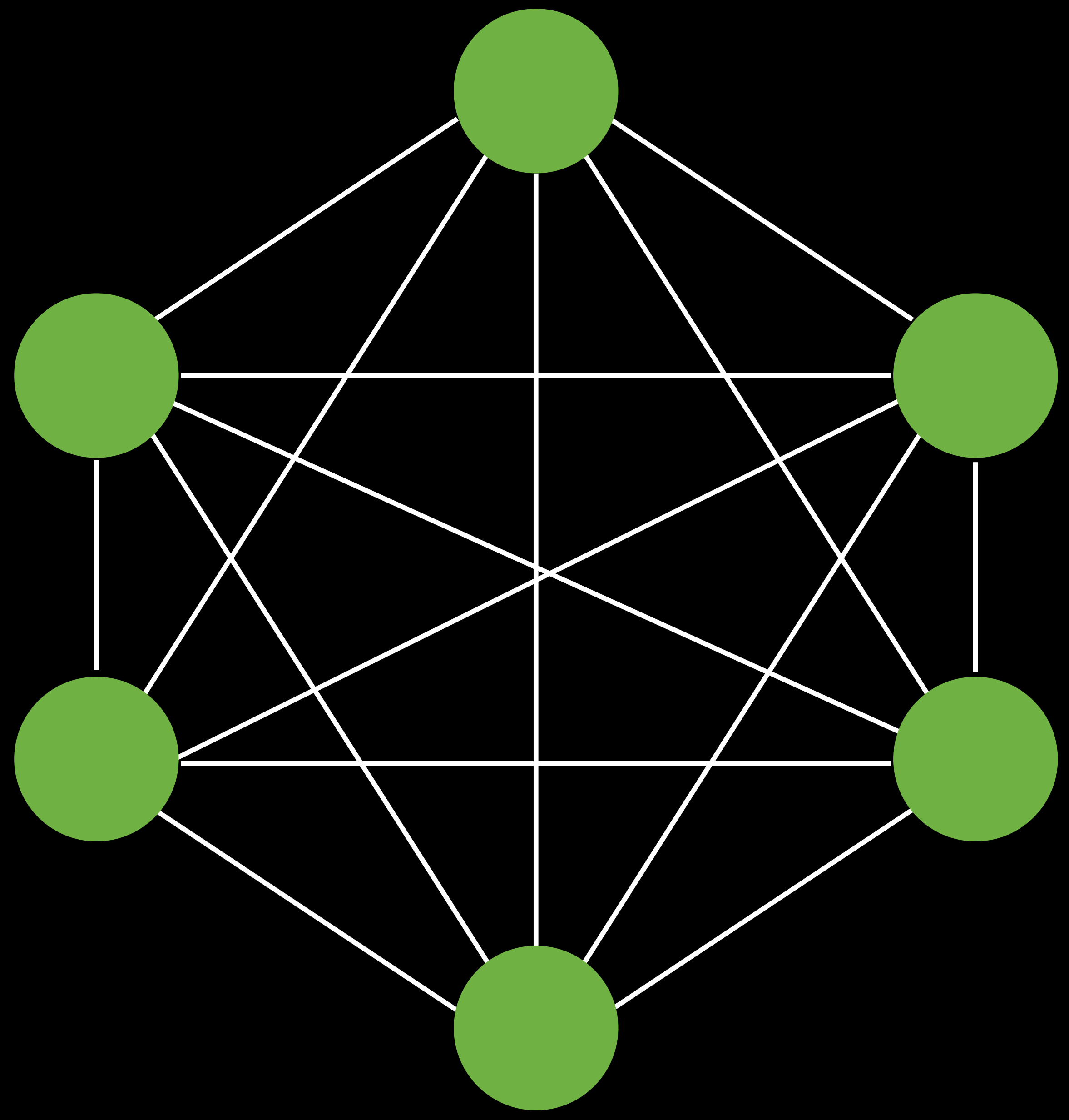
MEMBERSHIP: DESIRABLE PROPERTIES

- Connectedness
- Balance
- Short path-length
- Low clustering
- Scalability
- Accuracy

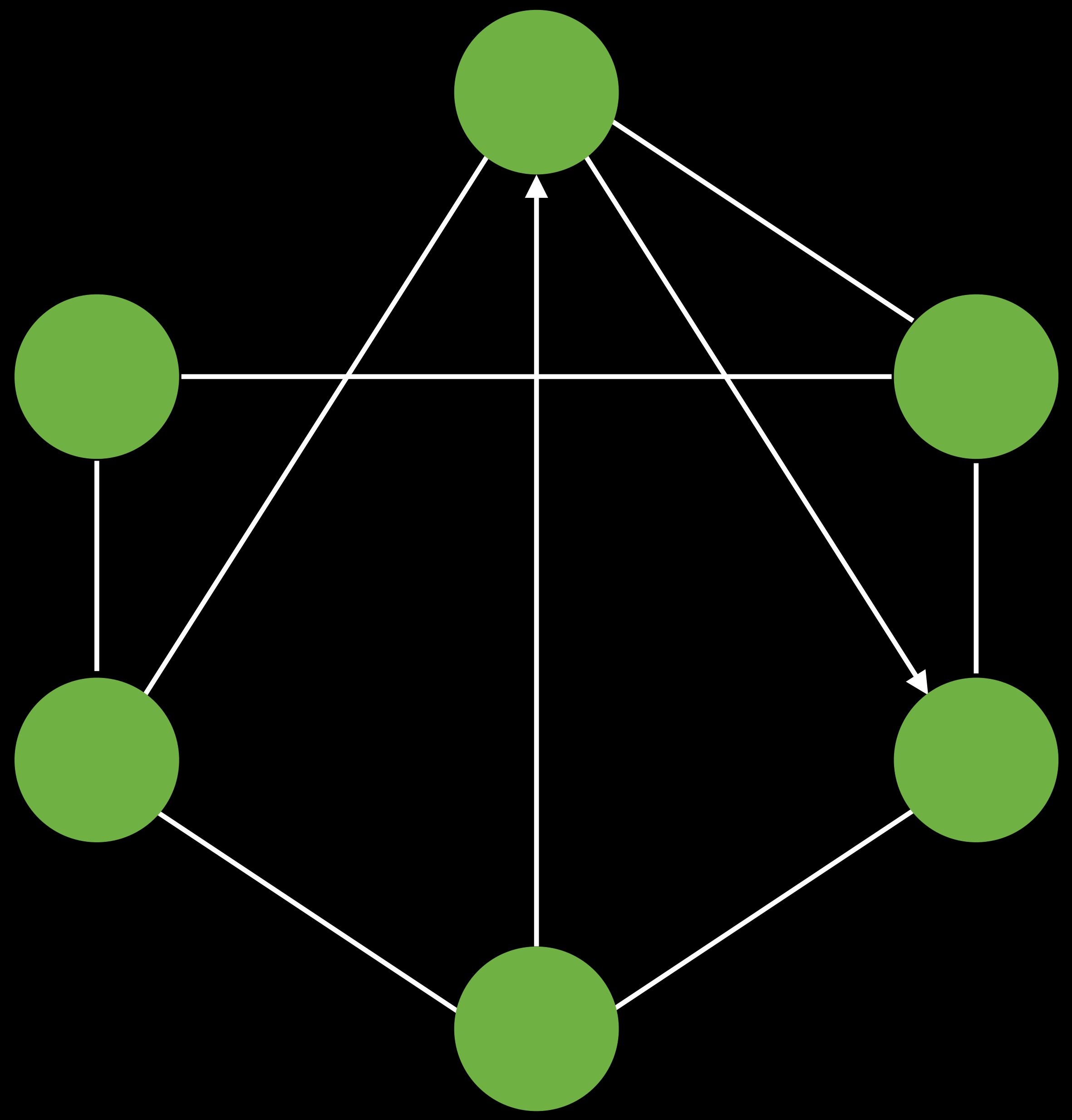
COMCAST

MEMBERSHIP: "VIEW" FLAVORS

Full

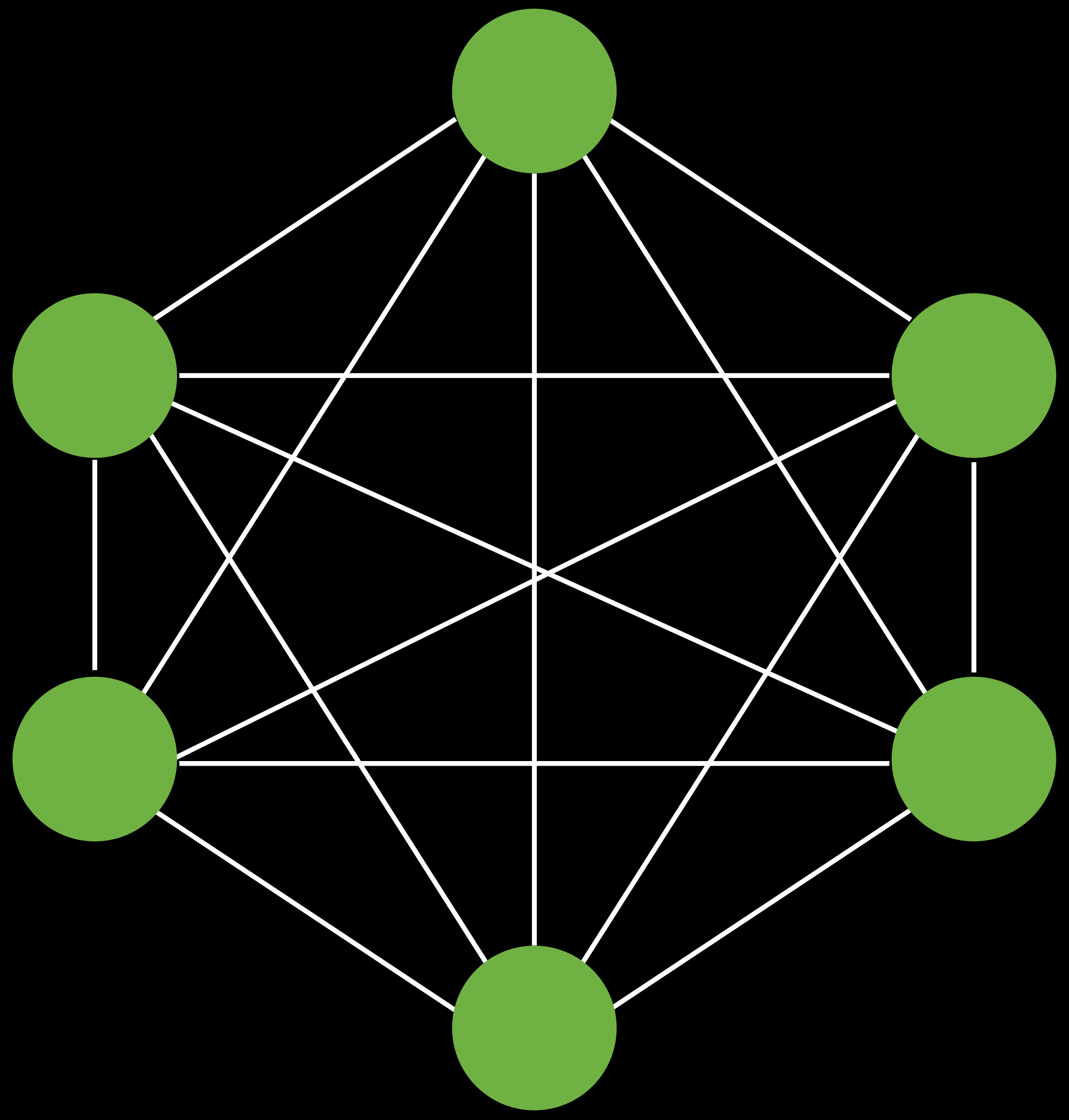


Partial



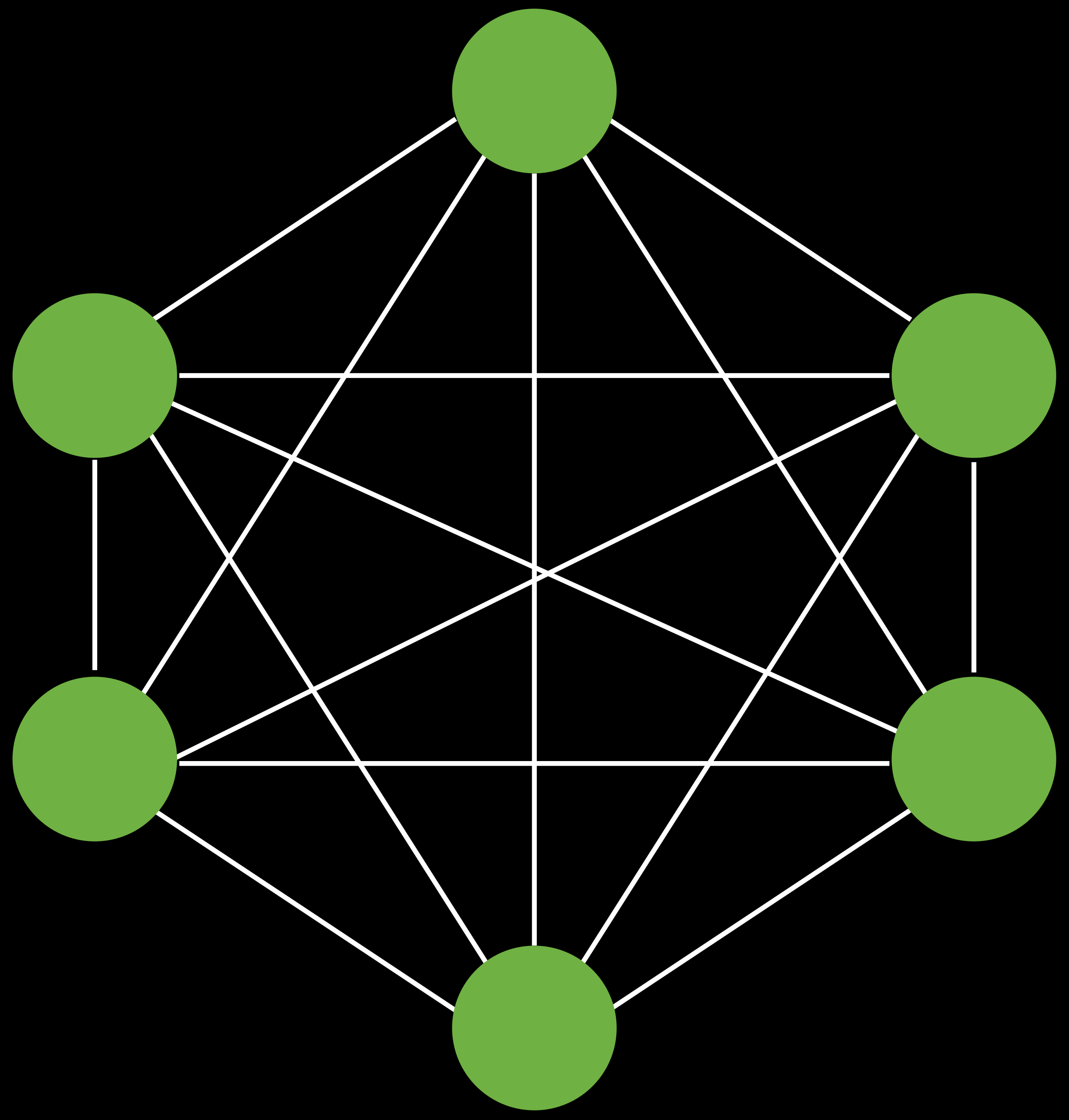
COMCAST

Full



COMCAST

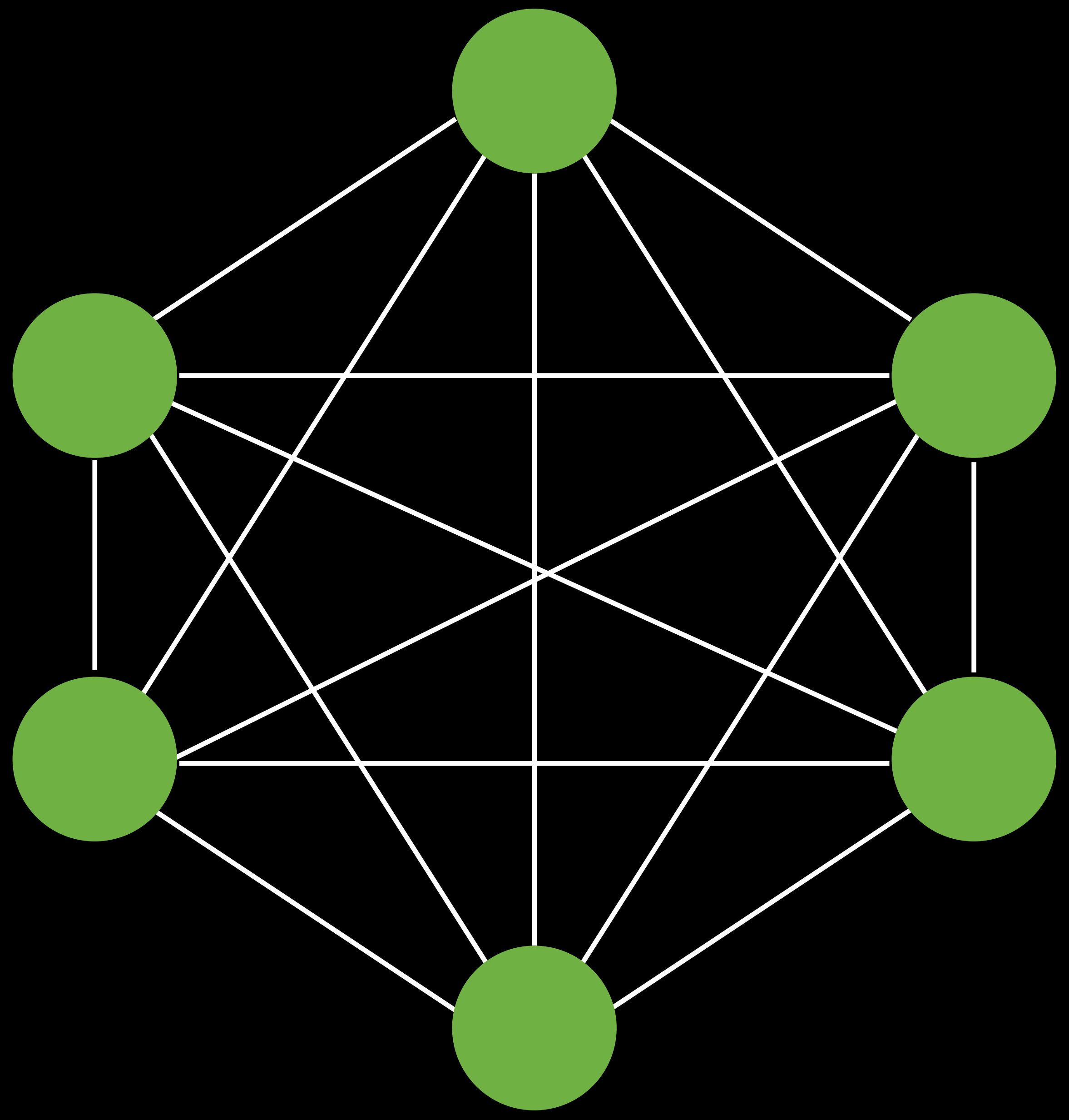
Full



✓ Connectedness

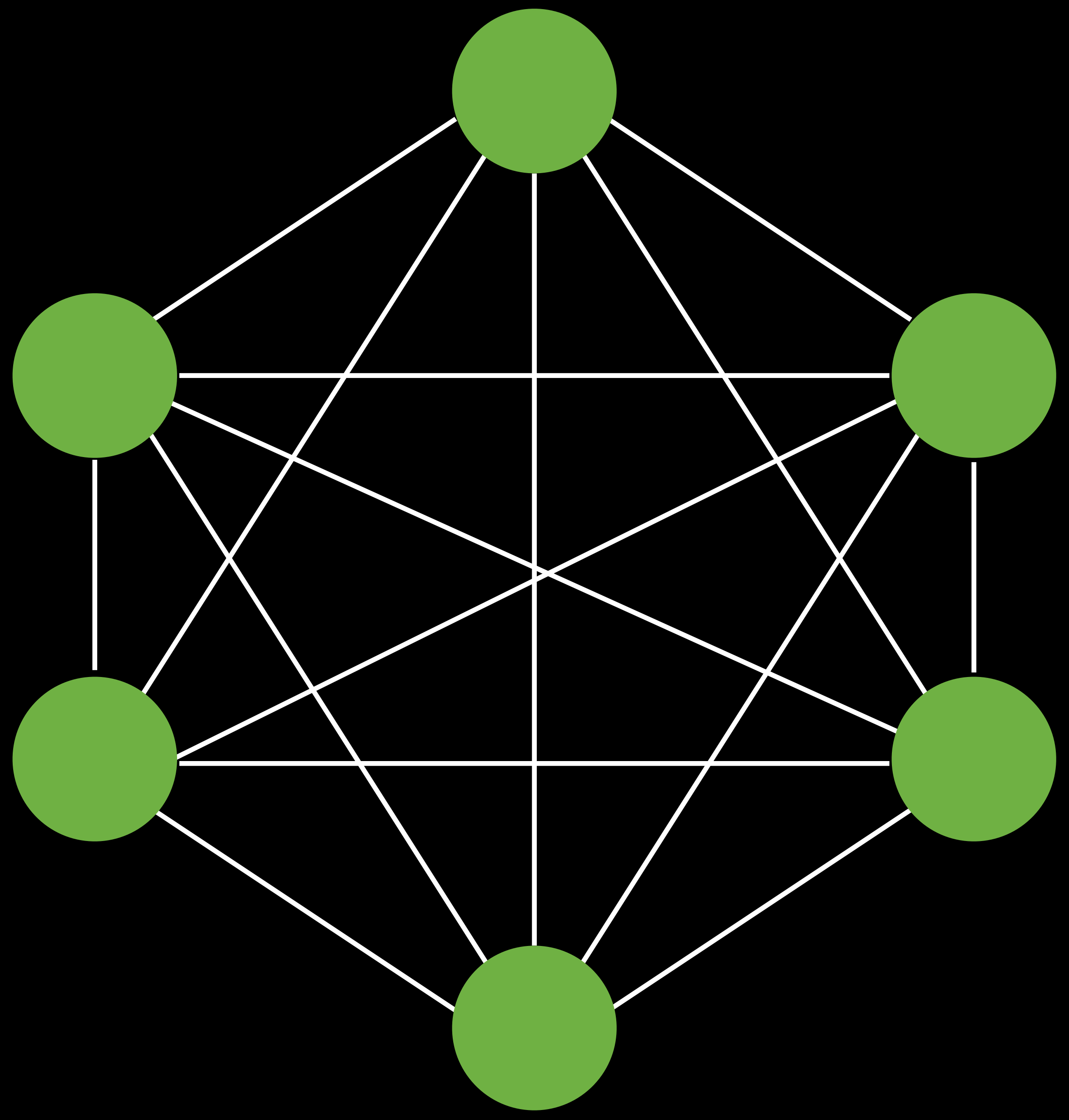
COMCAST

Full



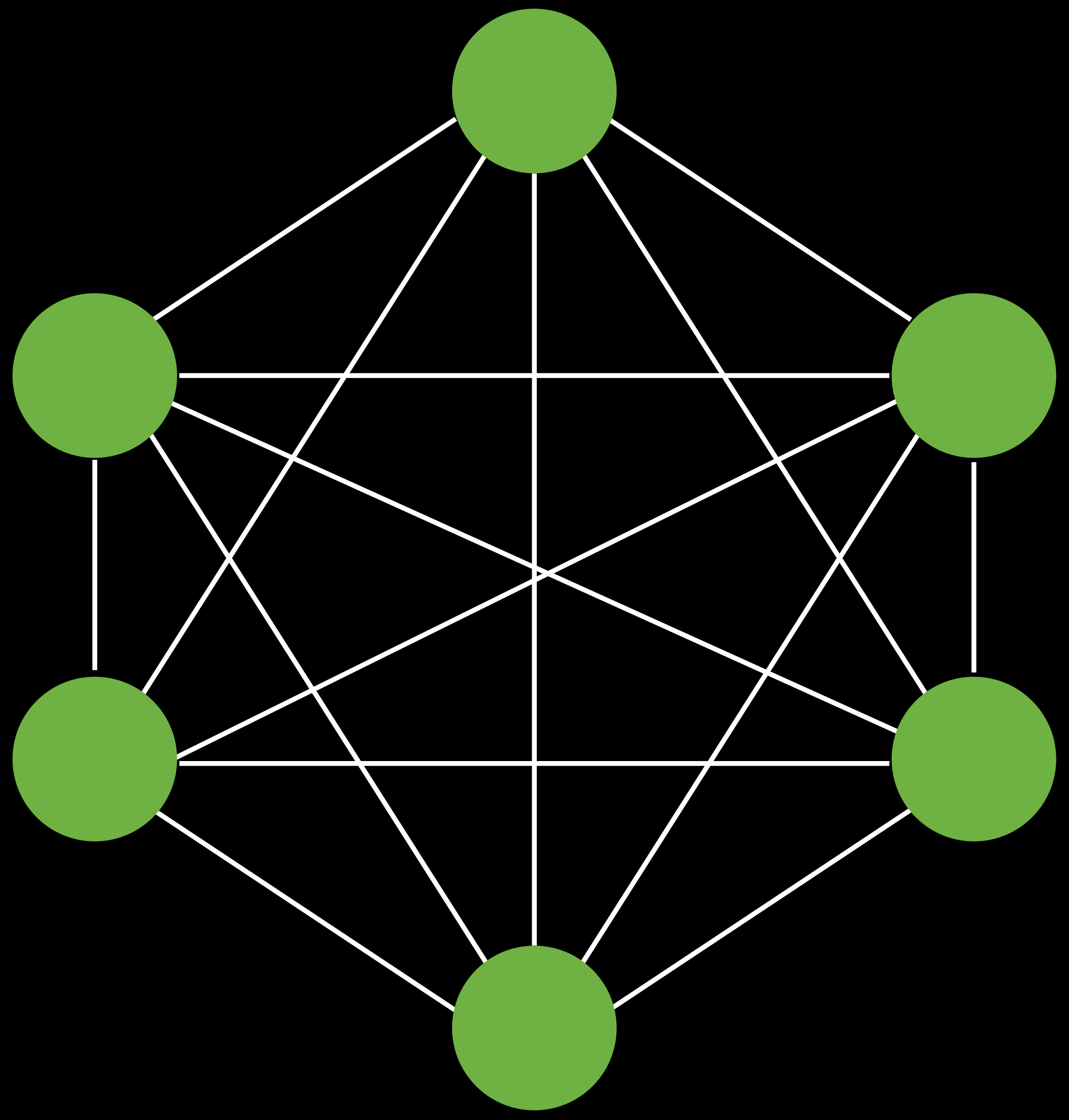
- ✓ Connectedness
- ✓ Short path-length

Full



- ✓ Connectedness
- ✓ Short path-length
- - Accuracy

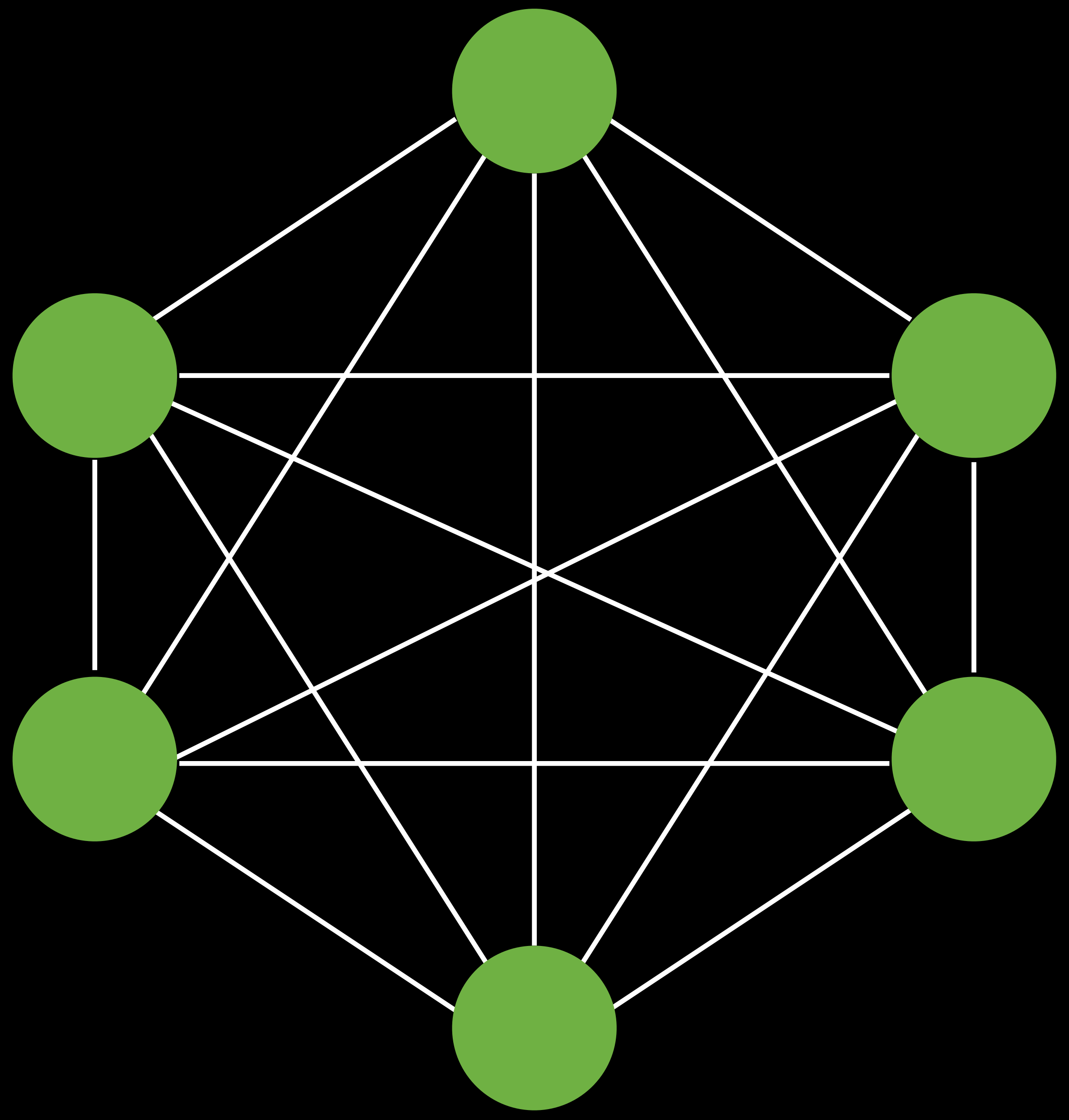
Full



- ✓ Connectedness
- ✓ Short path-length
- - Accuracy
- - Balance

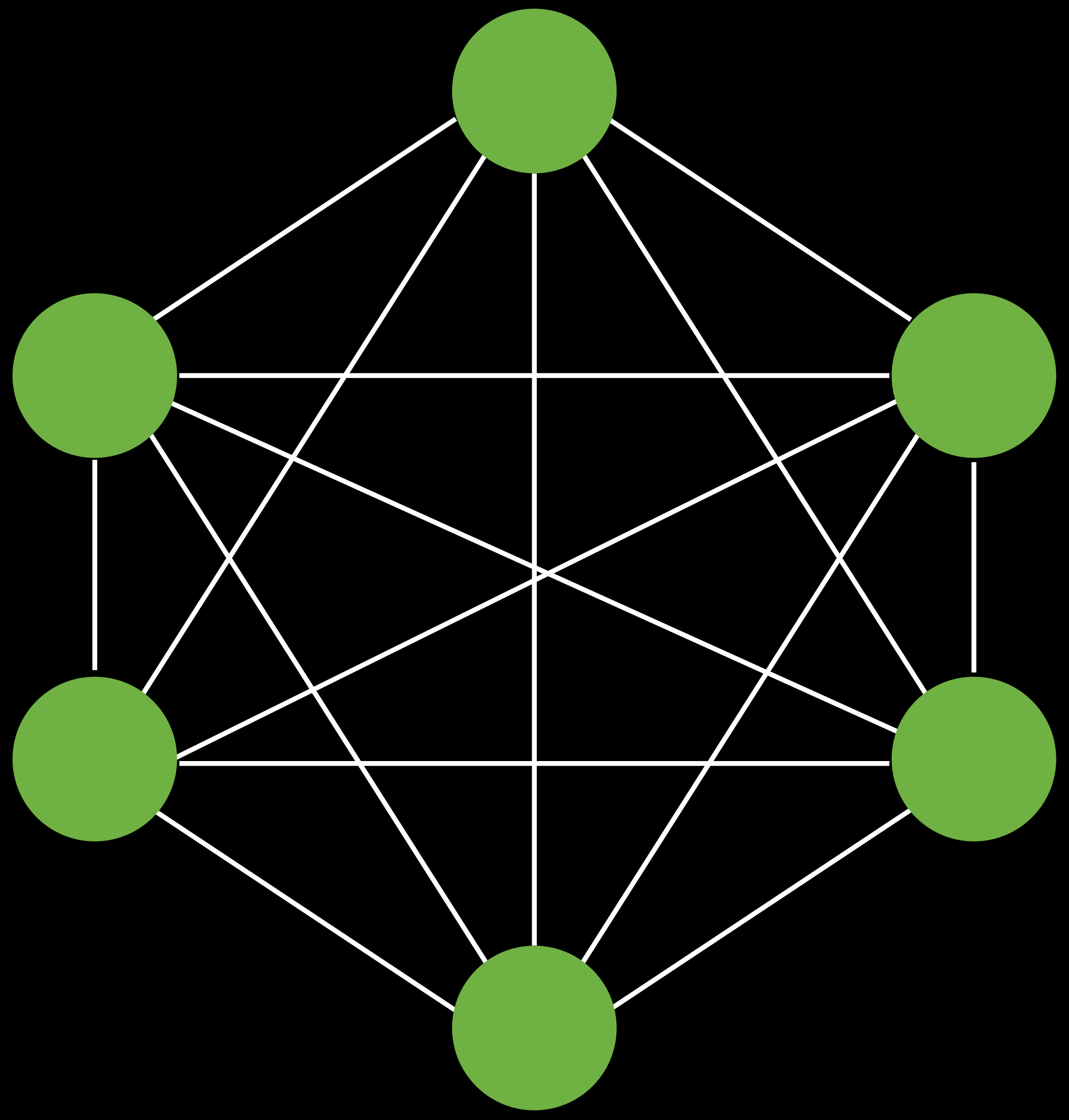
COMCAST

Full



- ✓ Connectedness
- ✓ Short path-length
- - Accuracy
- - Balance
- ● High Clustering

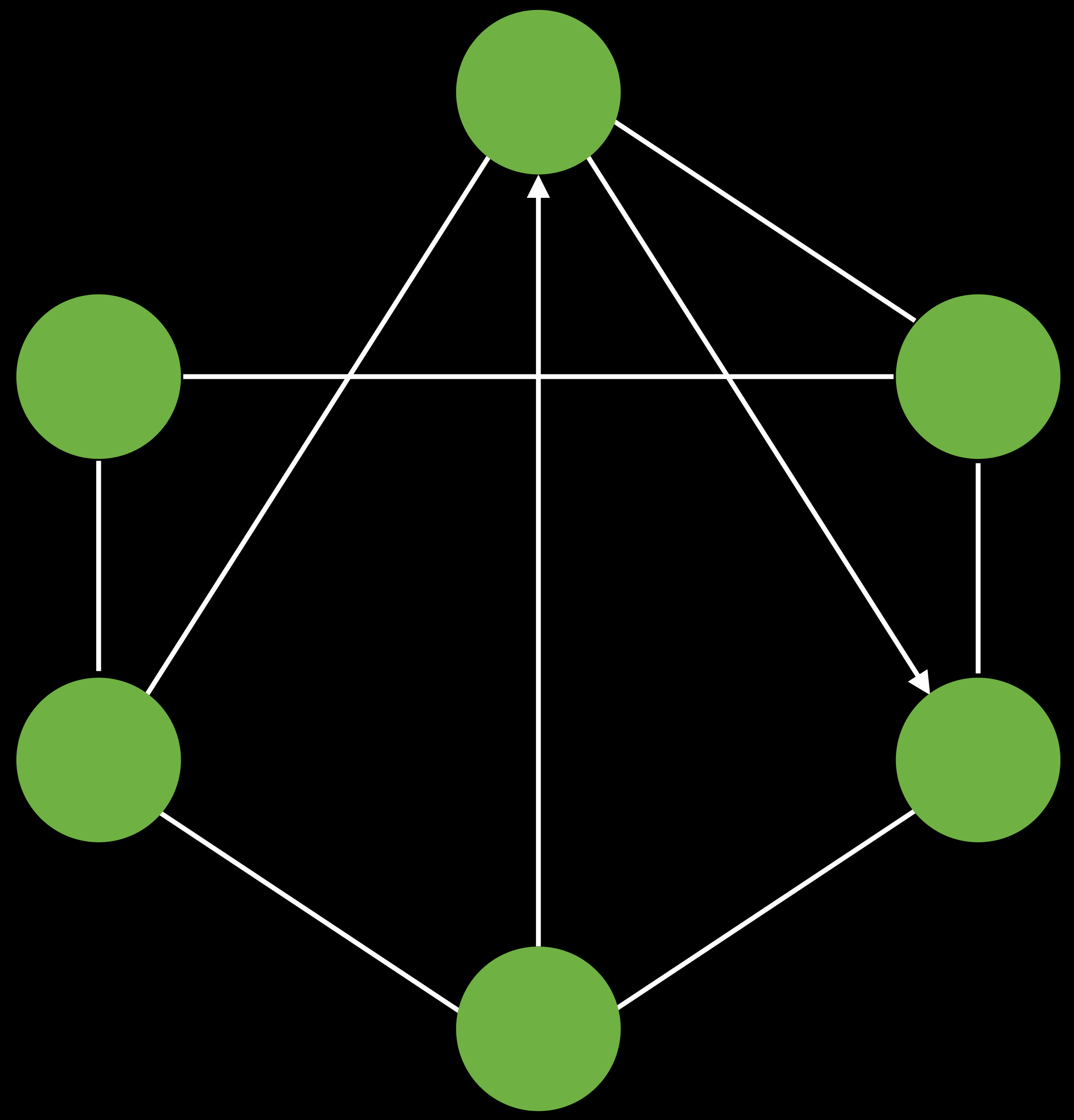
Full



- ✓ Connectedness
- ✓ Short path-length
- - Accuracy
- - Balance
- ● High Clustering
- ● Low Scalability

COMCAST

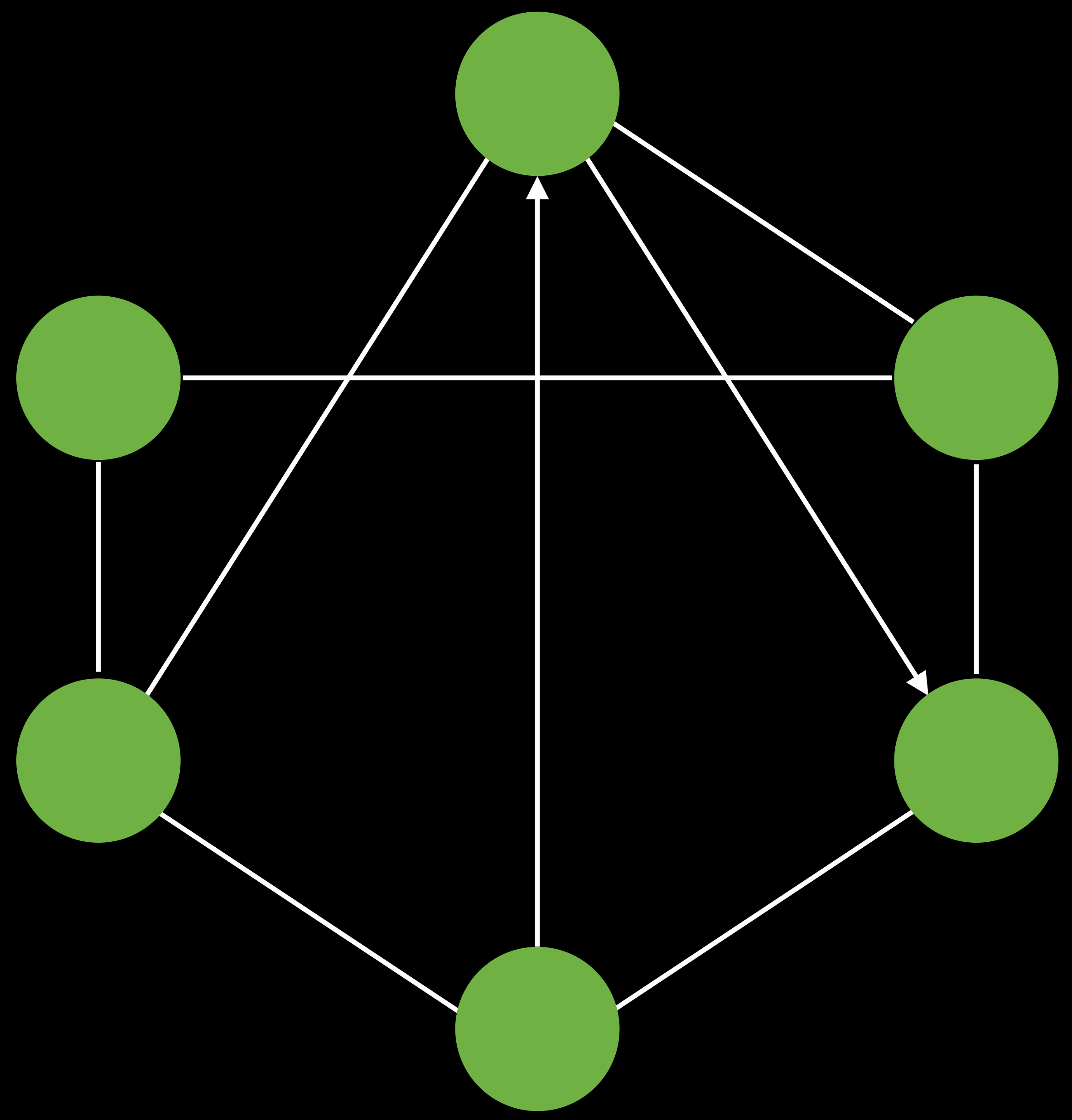
Partial



COMCAST

✓ Low Clustering

Partial

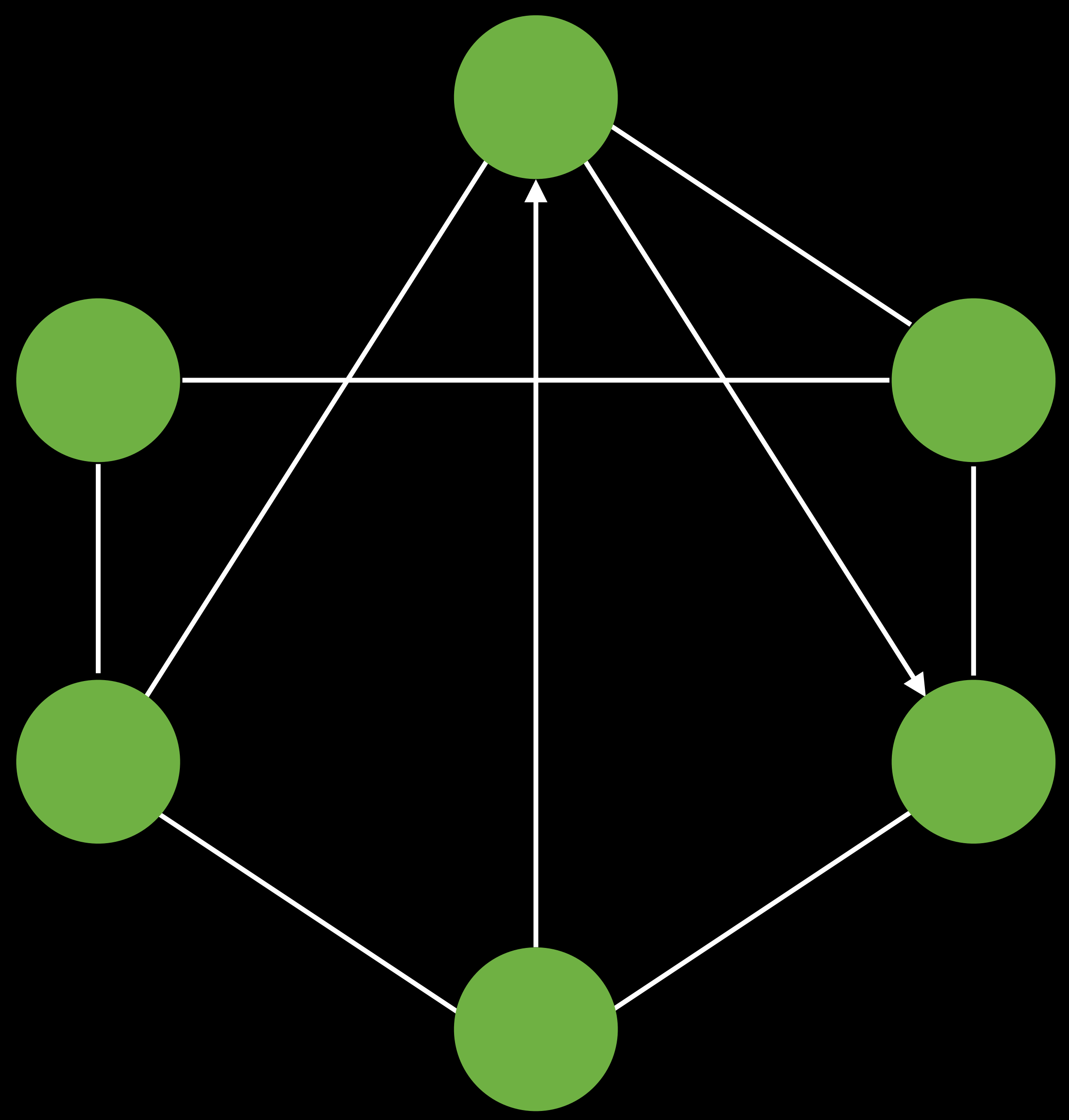


COMCAST

✓ Low Clustering

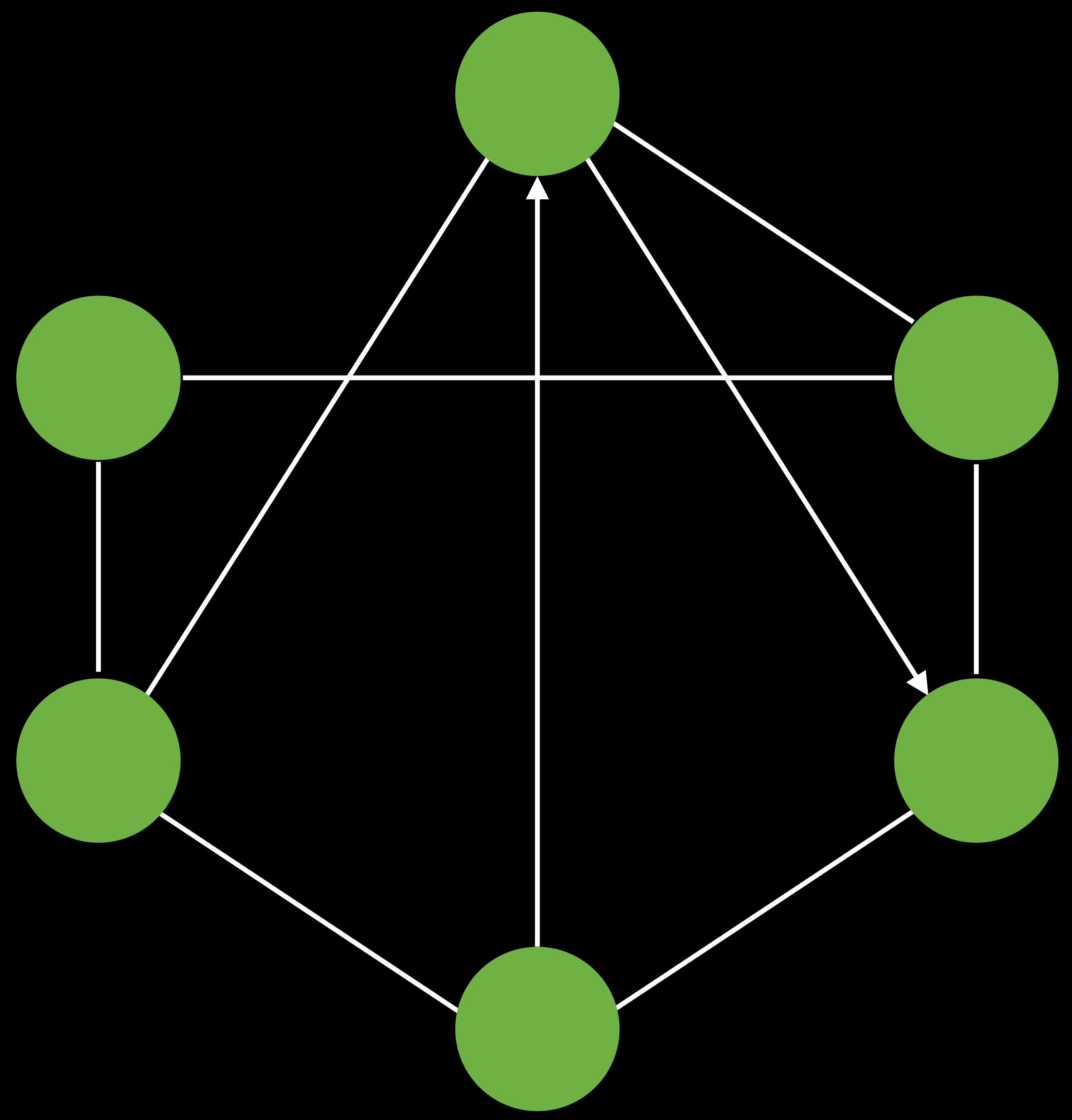
✓ High scalability

Partial



- ✓ Low Clustering
- ✓ High scalability
- - Connectedness

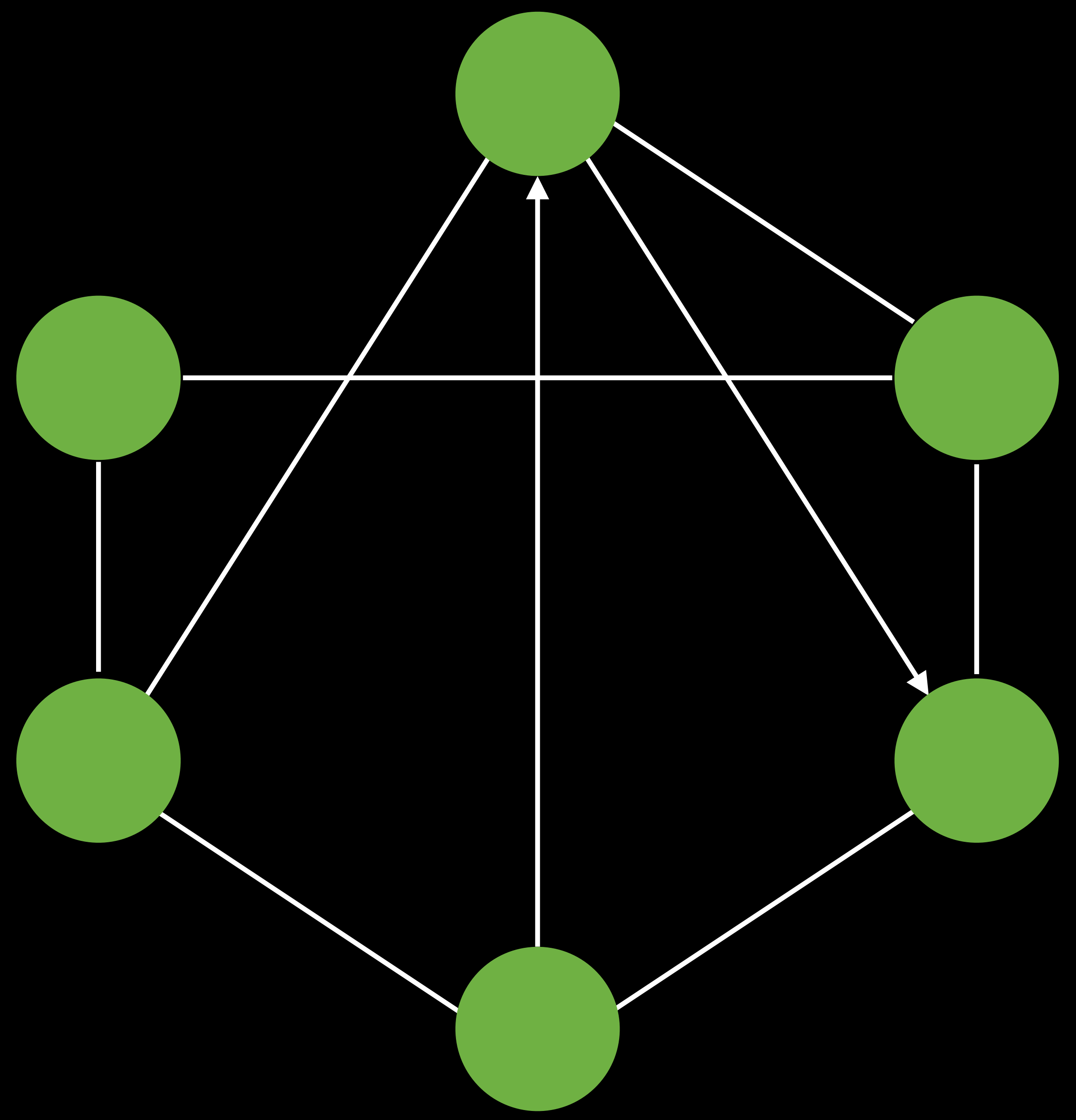
Partial



COMCAST

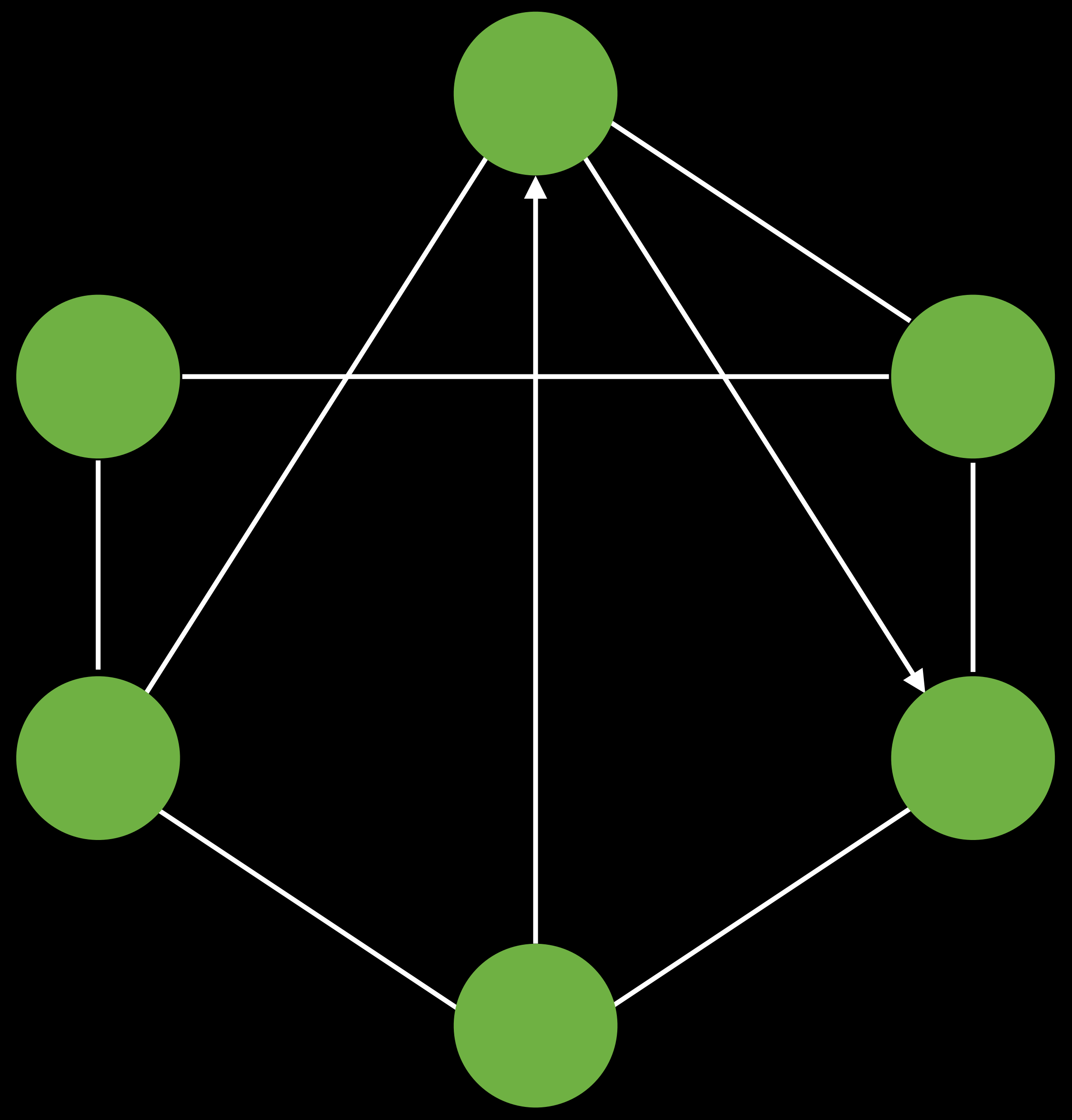
- ✓ Low Clustering
- ✓ High scalability
- - Connectedness
- - Balance

Partial



- ✓ Low Clustering
- ✓ High scalability
- - Connectedness
- - Balance
- - Path-length

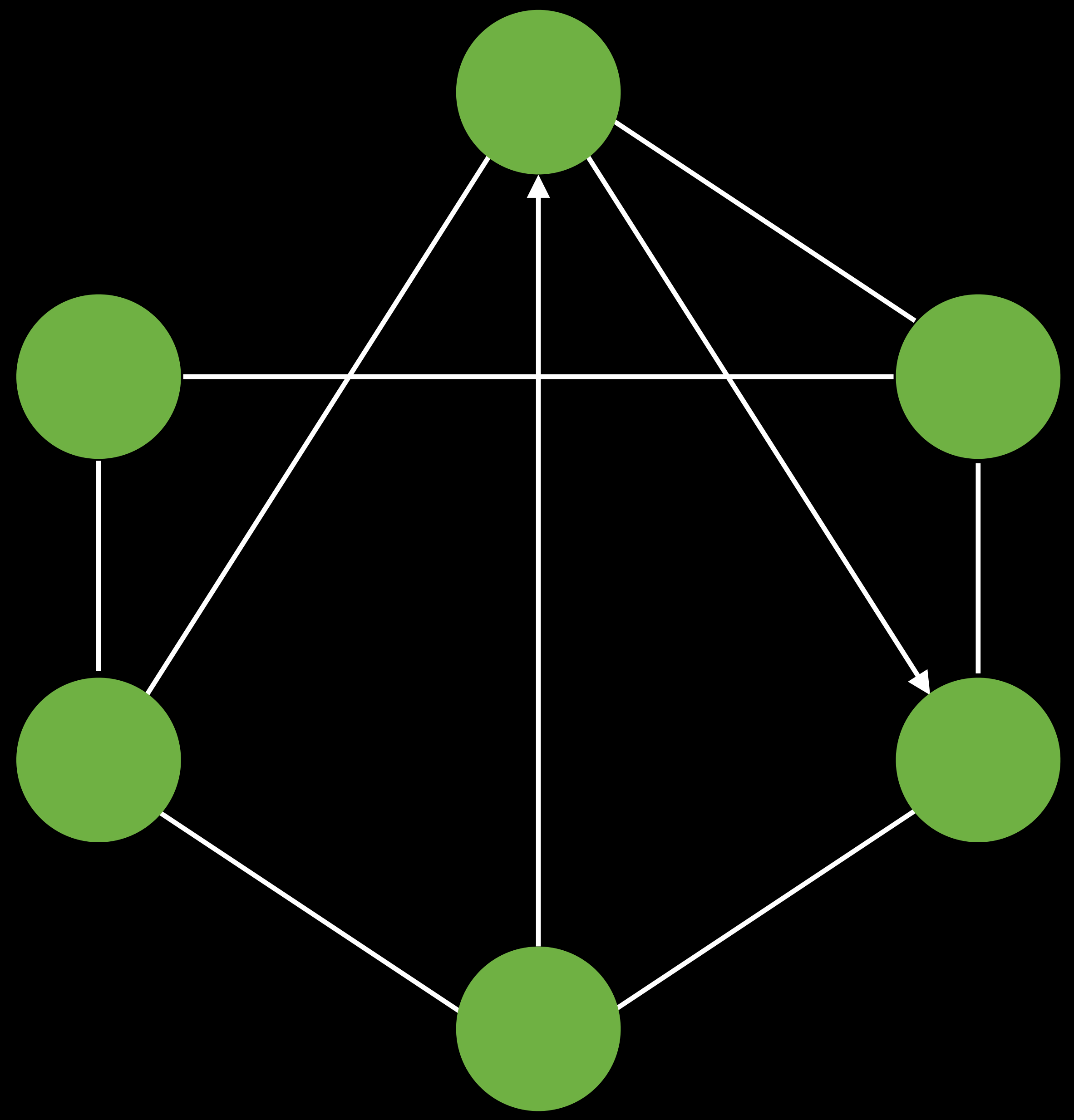
Partial



COMCAST

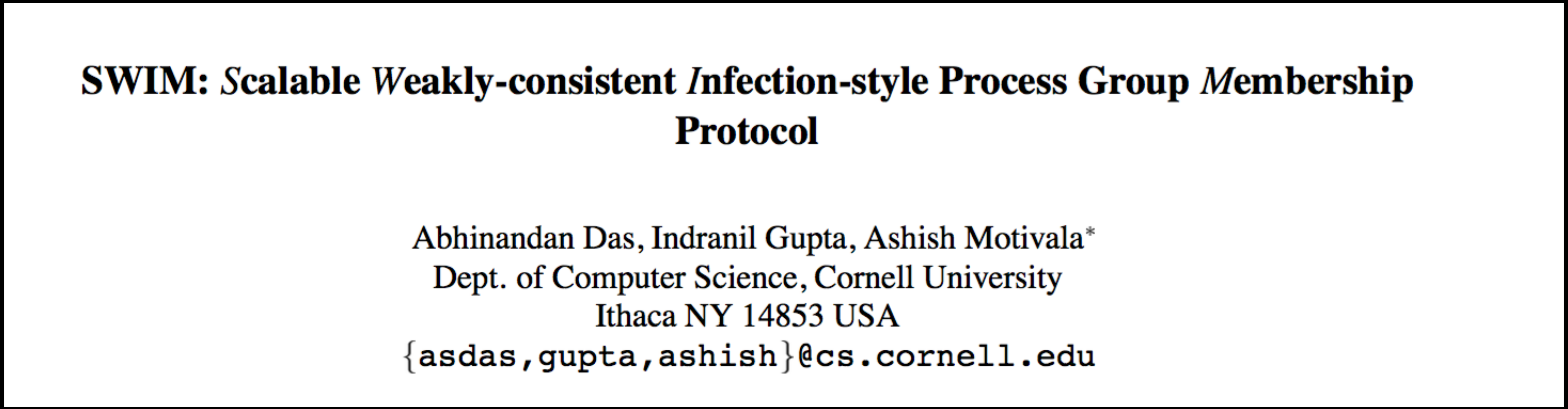
- ✓ Low Clustering
- ✓ High scalability
- - Connectedness
- - Balance
- - Path-length
- - Accuracy

Partial



COMCAST

SWIM - 2002



SWIM: *S*calable *W*eakly-consistent *I*nfection-style Process Group *M*embership Protocol

Abhinandan Das, Indranil Gupta, Ashish Motivala*
Dept. of Computer Science, Cornell University
Ithaca NY 14853 USA
{asdas,gupta,ashish}@cs.cornell.edu

SWIM - 2002

SWIM - 2002

Heartbeat protocols

SWIM - 2002

- Heartbeat protocols
- ● Quadratic load

SWIM - 2002

Heartbeat protocols

- Quadratic load
- Failure detection

Emoji provided free by Emoji One

COMCAST

SWIM - 2002

Heartbeat protocols

- Quadratic load
- Failure detection
  - Response times

Emoji provided free by Emoji One

COMCAST

SWIM - 2002

Heartbeat protocols

- Quadratic load
- Failure detection
  - Response times
  - False positives

Emoji provided free by Emoji One

COMCAST

SWIM - 2002

Heartbeat protocols

- Quadratic load
- Failure detection
  - Response times
  - False positives

SWIM solutions

Emoji provided free by Emoji One

COMCAST

SWIM - 2002

Heartbeat protocols

- Quadratic load
- Failure detection
  - Response times
  - False positives

SWIM solutions

➡ Separate membership and failure detection

Emoji provided free by Emoji One

COMCAST

SWIM - 2002

Heartbeat protocols

- Quadratic load
- Failure detection
  - Response times
  - False positives

SWIM solutions

- ➡ Separate membership and failure detection
- ➡ Randomized probing

Emoji provided free by Emoji One

COMCAST

SWIM - 2002

Heartbeat protocols

- Quadratic load
- Failure detection
  - Response times
  - False positives

SWIM solutions

- ➡ Separate membership and failure detection
- ➡ Randomized probing
- ➡ Piggyback membership on probes

Emoji provided free by Emoji One

COMCAST

SWIM - 2002

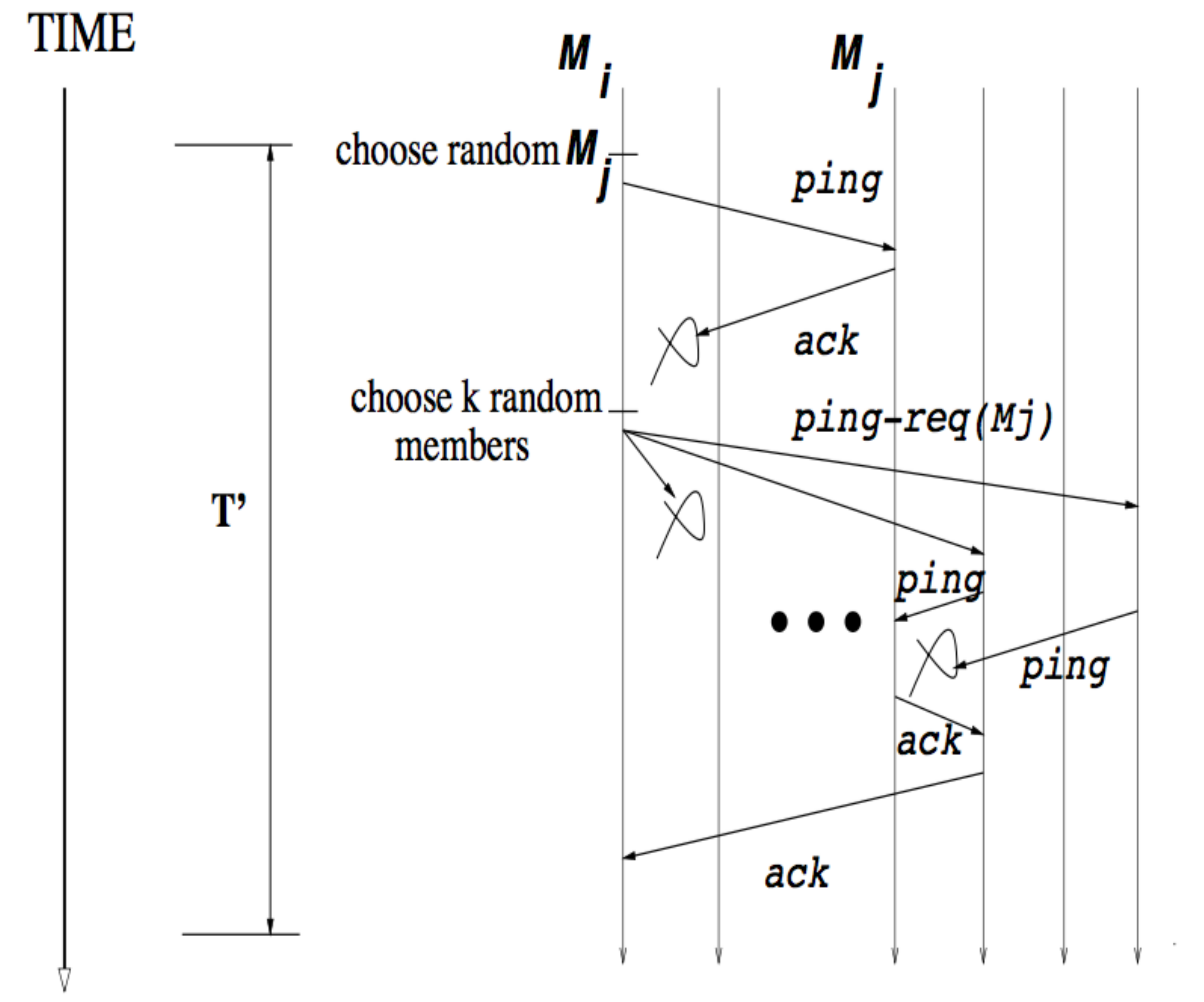
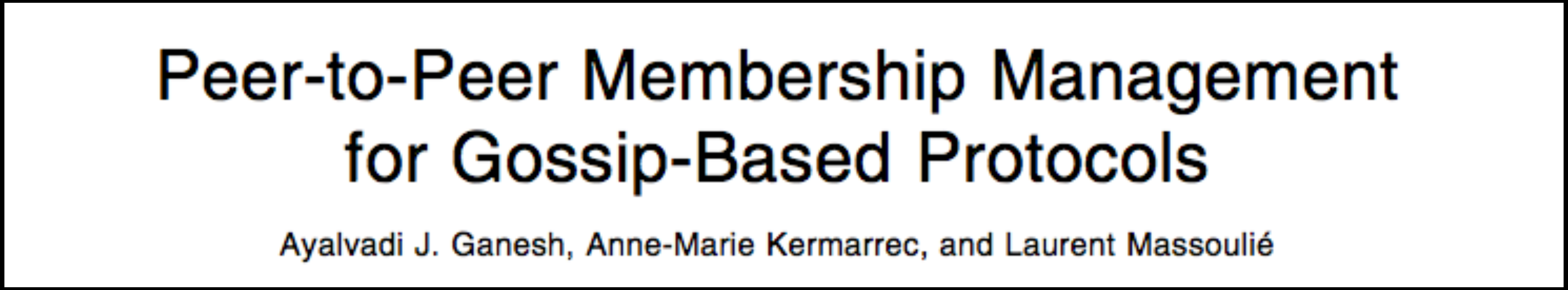


Figure 1. SWIM failure detection: Example protocol period at M_i. This shows all the possible messages that a protocol period may initiate. Some message contents excluded for simplicity.

SCAMP - 2003



Peer-to-Peer Membership Management for Gossip-Based Protocols

Ayalvadi J. Ganesh, Anne-Marie Kermarrec, and Laurent Massoulié

SCAMP - 2003

SCAMP - 2003

- Full views limit scalability

SCAMP - 2003

◉ Full views limit scalability

➡ Flexible partial-view size, asymmetric

SCAMP - 2003

- Full views limit scalability
- Flexible partial-view size, asymmetric
- Reactive view management

COMCAST

SCAMP - 2003

- Full views limit scalability
- ➡ Flexible partial-view size, asymmetric
- ➡ Reactive view management
- ➡ Join (“subscribe”) via random walk

SCAMP - 2003

- ◉ Full views limit scalability
- ➡ Flexible partial-view size, asymmetric
- ➡ Reactive view management
- ➡ Join (“subscribe”) via random walk
- ➡ Automatic balancing via indirection and leases

SCAMP - 2003

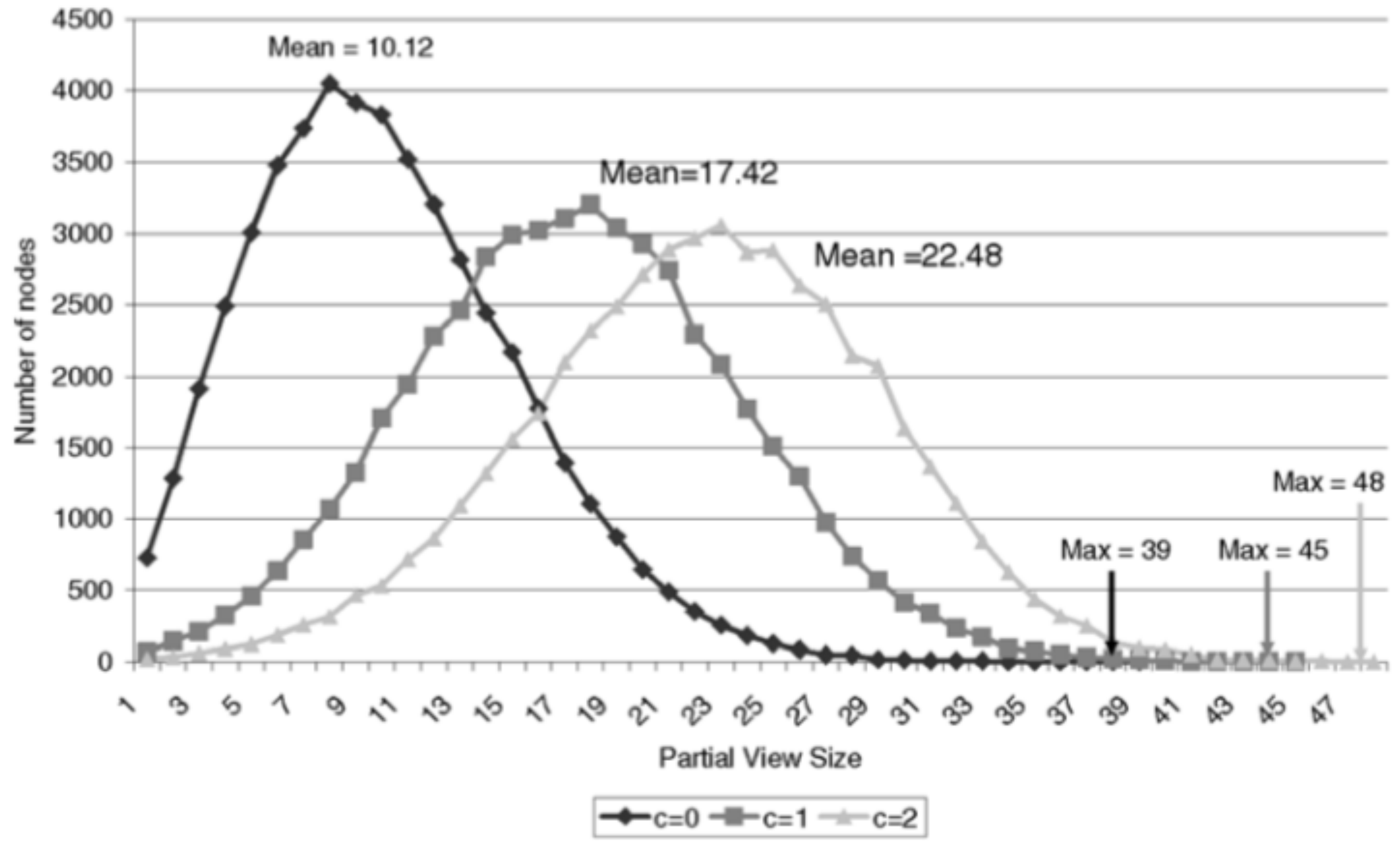


Fig. 5. Impact of c on the partial view size distribution in a 50,000 node group.

SCAMP - 2003

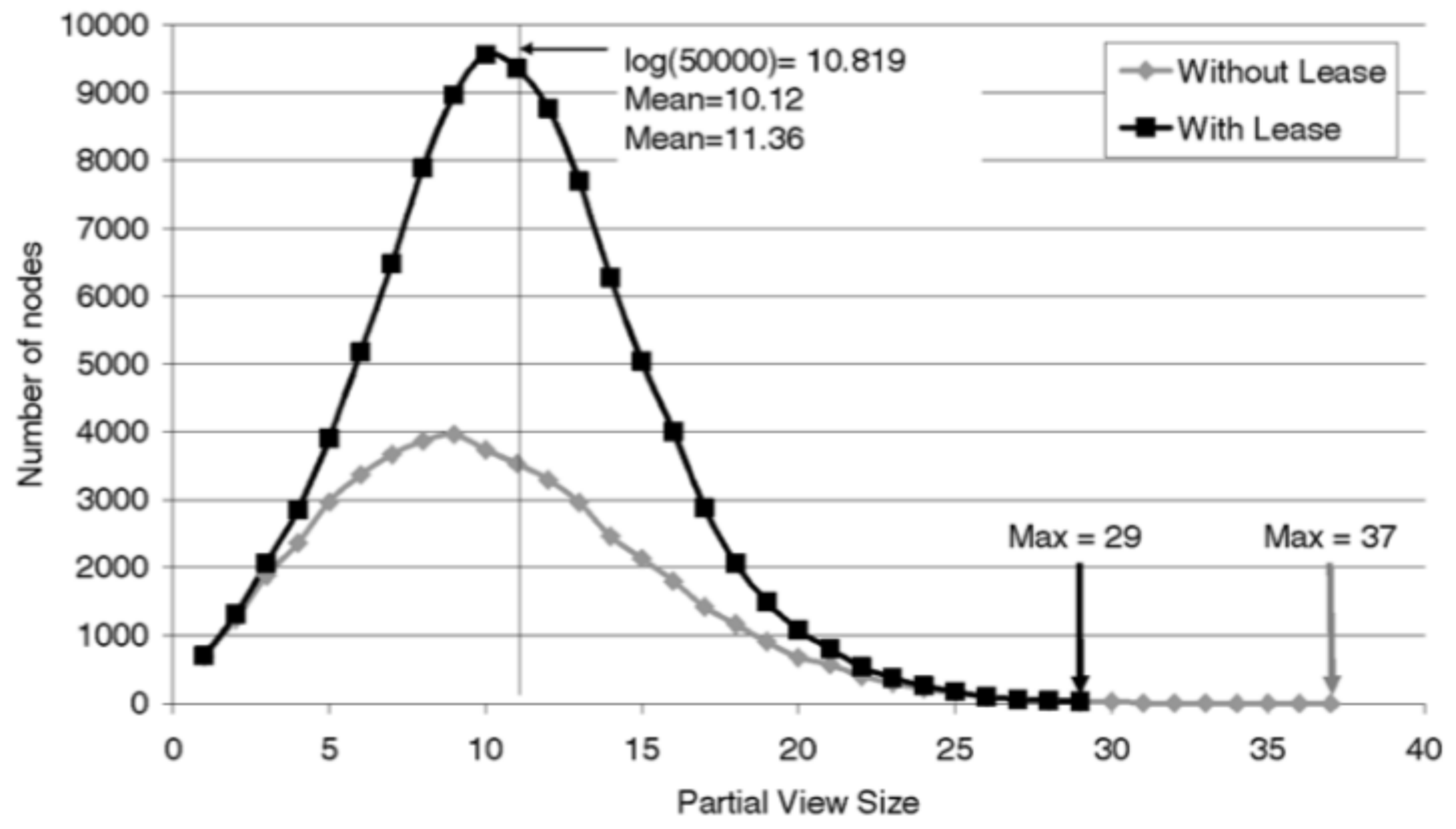
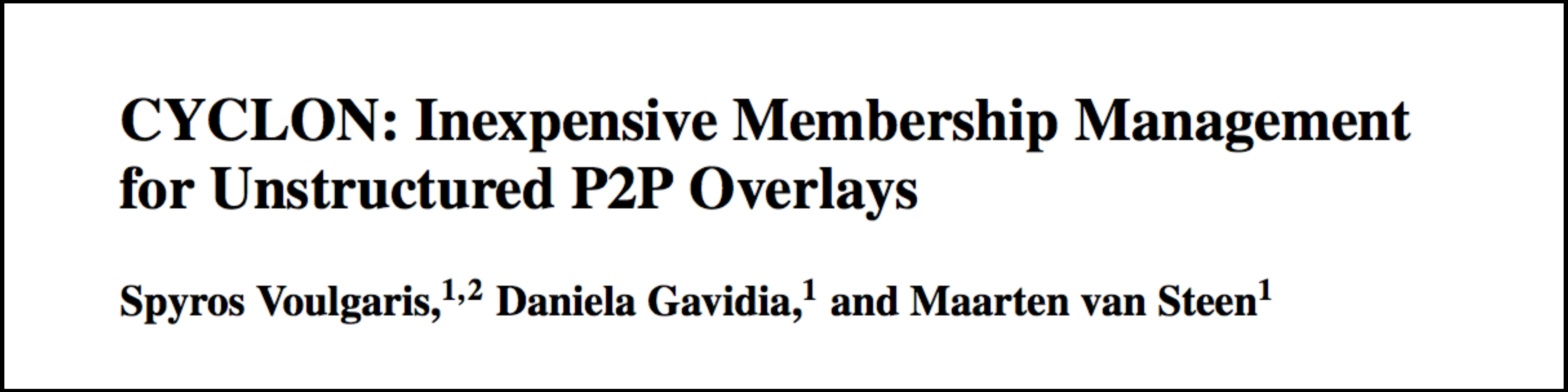


Fig. 8. Distribution of the partial views by size in a 50,000 node group with and without the lease mechanism.

COMCAST

CYCLON - 2005



CYCLON: Inexpensive Membership Management for Unstructured P2P Overlays

Spyros Voulgaris,[1,2] Daniela Gavidia,[1] and Maarten van Steen[1]

CYCLON - 2005

CYCLON - 2005

- Random shuffling doesn’t create good balance

CYCLON - 2005

◉ Random shuffling doesn’t create good balance

➡ Fixed partial-view size, symmetric

CYCLON - 2005

◉ Random shuffling doesn’t create good balance

➡ Fixed partial-view size, symmetric

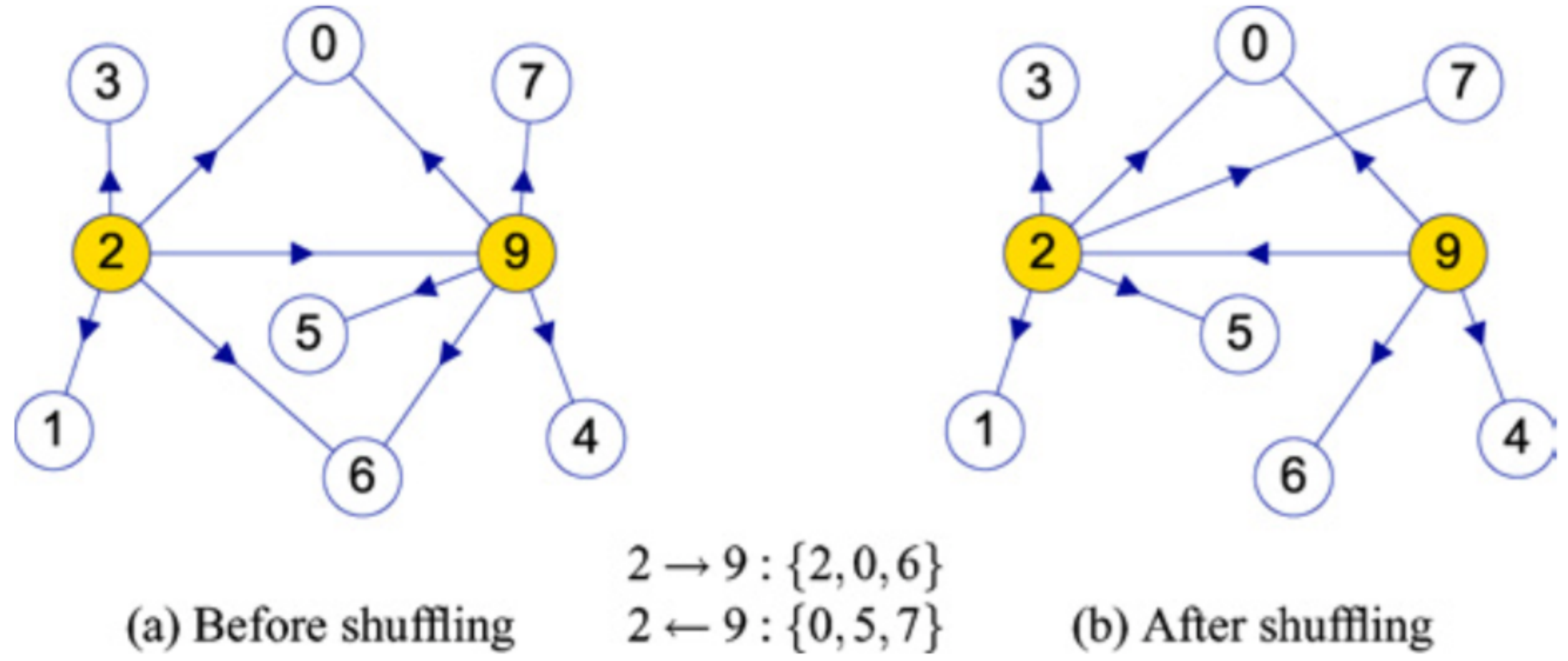
➡ Cyclic view management

COMCAST

CYCLON - 2005

- ◉ Random shuffling doesn’t create good balance
- ➡ Fixed partial-view size, symmetric
- ➡ Cyclic view management
- ➡ Join via random walk

CYCLON - 2005



(a) Before shuffling
$2 \rightarrow 9 : \{2, 0, 6\}$
$2 \leftarrow 9 : \{0, 5, 7\}$
(b) After shuffling

Fig. 1. An example of shuffling between nodes 2 and 9. Note that, among other changes, the link between 2 and 9 reverses direction.

CYCLON - 2005

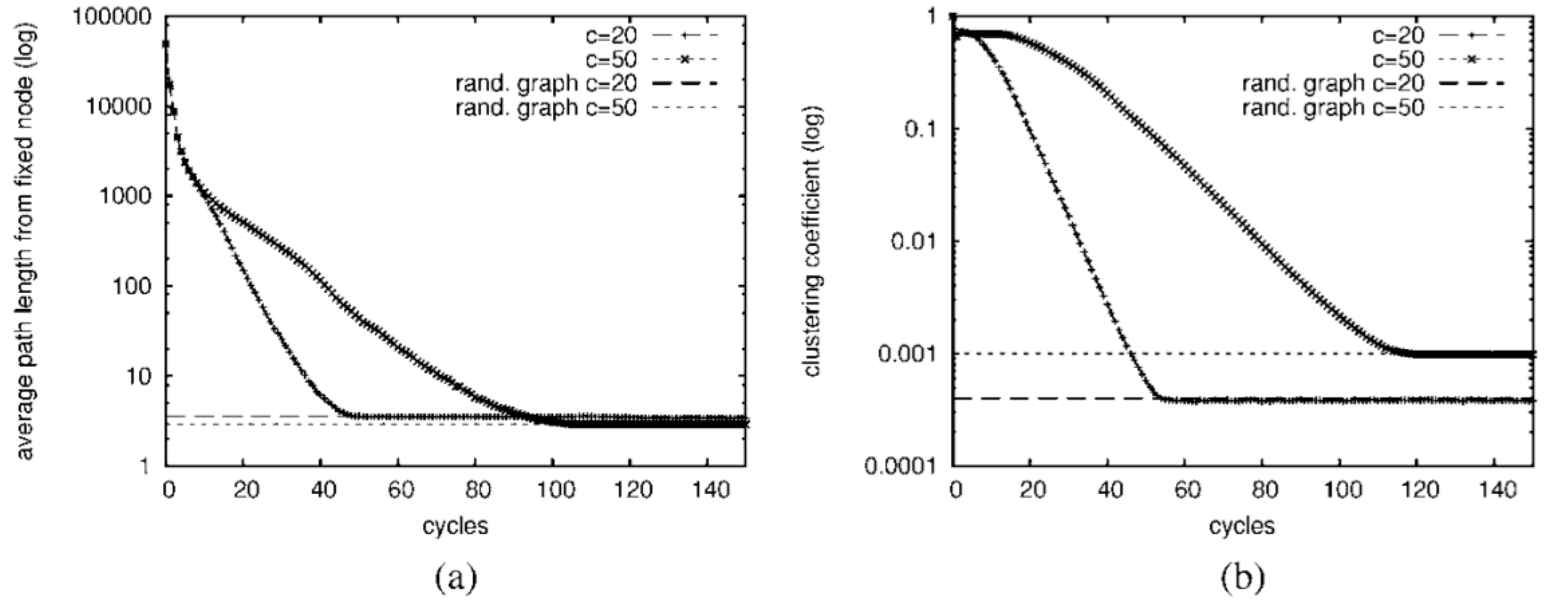


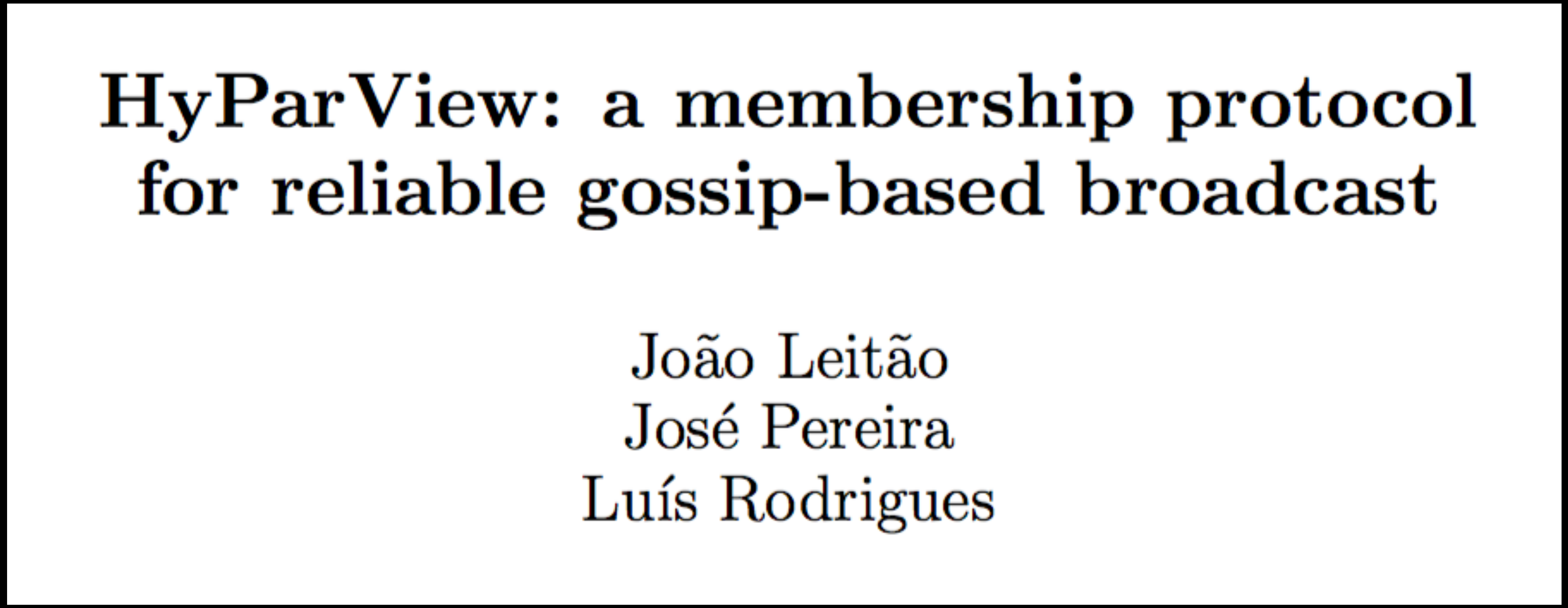
Fig. 2. (a) Average shortest path length between two nodes for different cache sizes. (b) Average clustering coefficient taken over all nodes.

PROBLEMS WITH SCAMP & CYCLON

- No failure detectors
- SCAMP: asymmetric views $\Longrightarrow$ disconnection
- SCAMP: unbounded view size $\Longrightarrow$ imbalance

COMCAST

HYPARVIEW - 2007



HyParView: a membership protocol for reliable gossip-based broadcast

João Leitão
José Pereira
Luís Rodrigues

HYPARVIEW - 2007

COMCAST

HYPARVIEW - 2007

- Fanout is related to reliability

HYPARVIEW - 2007

- Fanout is related to reliability
- High failure rates decrease quality

COMCAST

HYPARVIEW - 2007

- Fanout is related to reliability
- High failure rates decrease quality

HYPARVIEW - 2007

- Fanout is related to reliability
- High failure rates decrease quality

➡ TCP for transport and failure detector

COMCAST

HYPARVIEW - 2007

- Fanout is related to reliability
- High failure rates decrease quality

➡ TCP for transport and failure detector

➡ Small reactive view (“active”)

COMCAST

HYPARVIEW - 2007

- Fanout is related to reliability
- High failure rates decrease quality

- ➡ TCP for transport and failure detector
- ➡ Small reactive view (“active”)
- ➡ Larger cyclic view (“passive”)

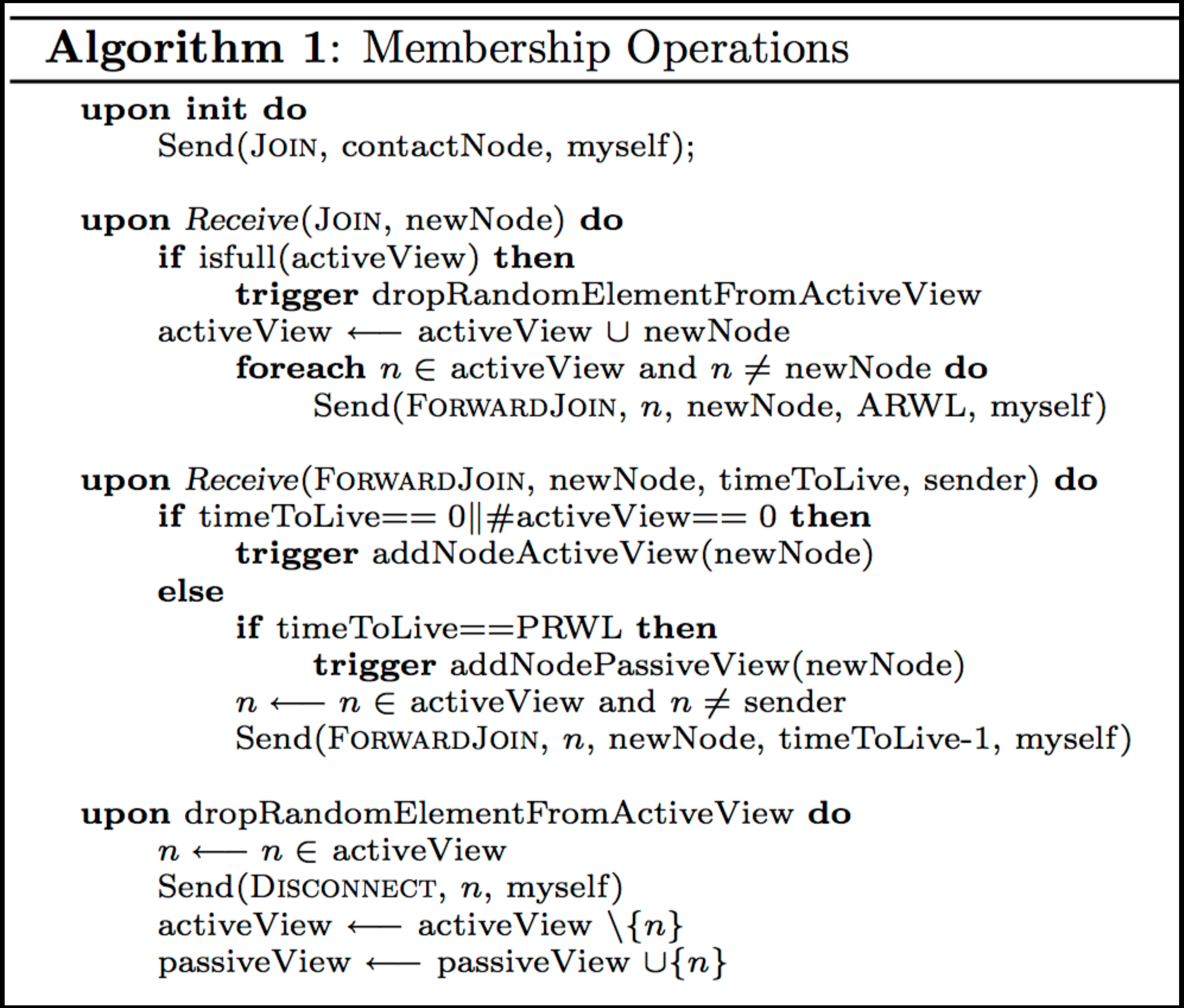
HYPARVIEW - 2007

- Fanout is related to reliability
- High failure rates decrease quality

- ➡ TCP for transport and failure detector
- ➡ Small reactive view (“active”)
- ➡ Larger cyclic view (“passive”)
- ➡ Join and shuffle via random walk

COMCAST

HYPARVIEW - 2007

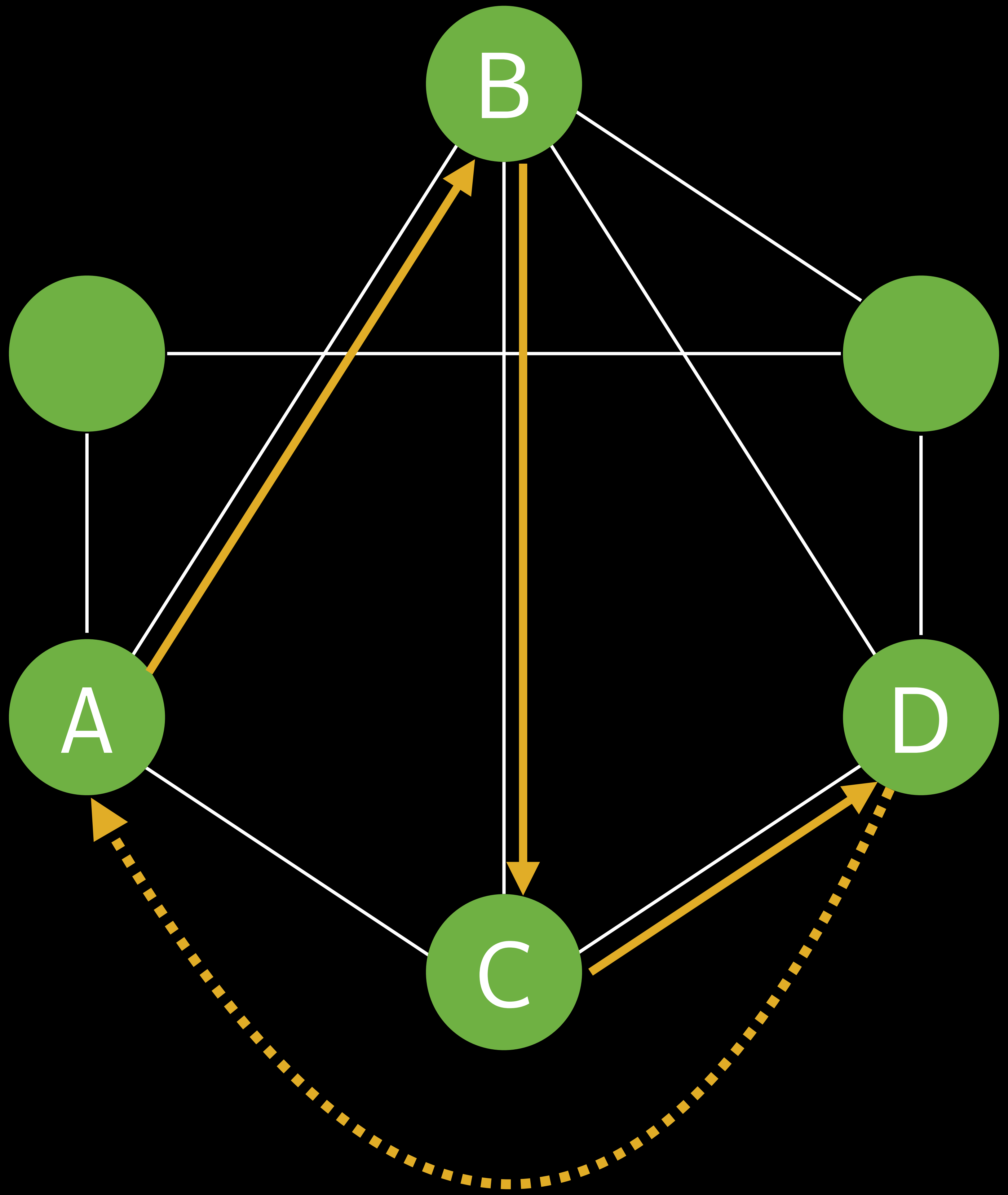


Algorithm 1: Membership Operations

```
upon init do
    Send(JOIN, contactNode, myself);

upon Receive(JOIN, newNode) do
    if isfull(activeView) then
        trigger dropRandomElementFromActiveView
    activeView ⟵ activeView $\cup$ newNode
        foreach $n \in$ activeView and $n \neq$ newNode do
            Send(FORWARDJOIN, $n$, newNode, ARWL, myself)

upon Receive(FORWARDJOIN, newNode, timeToLive, sender) do
    if timeToLive== 0||#activeView== 0 then
        trigger addNodeActiveView(newNode)
    else
        if timeToLive==PRWL then
            trigger addNodePassiveView(newNode)
        $n \longleftarrow n \in$ activeView and $n \neq$ sender
        Send(FORWARDJOIN, $n$, newNode, timeToLive-1, myself)

upon dropRandomElementFromActiveView do
    $n \longleftarrow n \in$ activeView
    Send(DISCONNECT, $n$, myself)
    activeView ⟵ activeView $\setminus\{n\}$
    passiveView ⟵ passiveView $\cup\{n\}$
```

HYPARVIEW - 2007

Passive view maintenance



HYPARVIEW - 2007

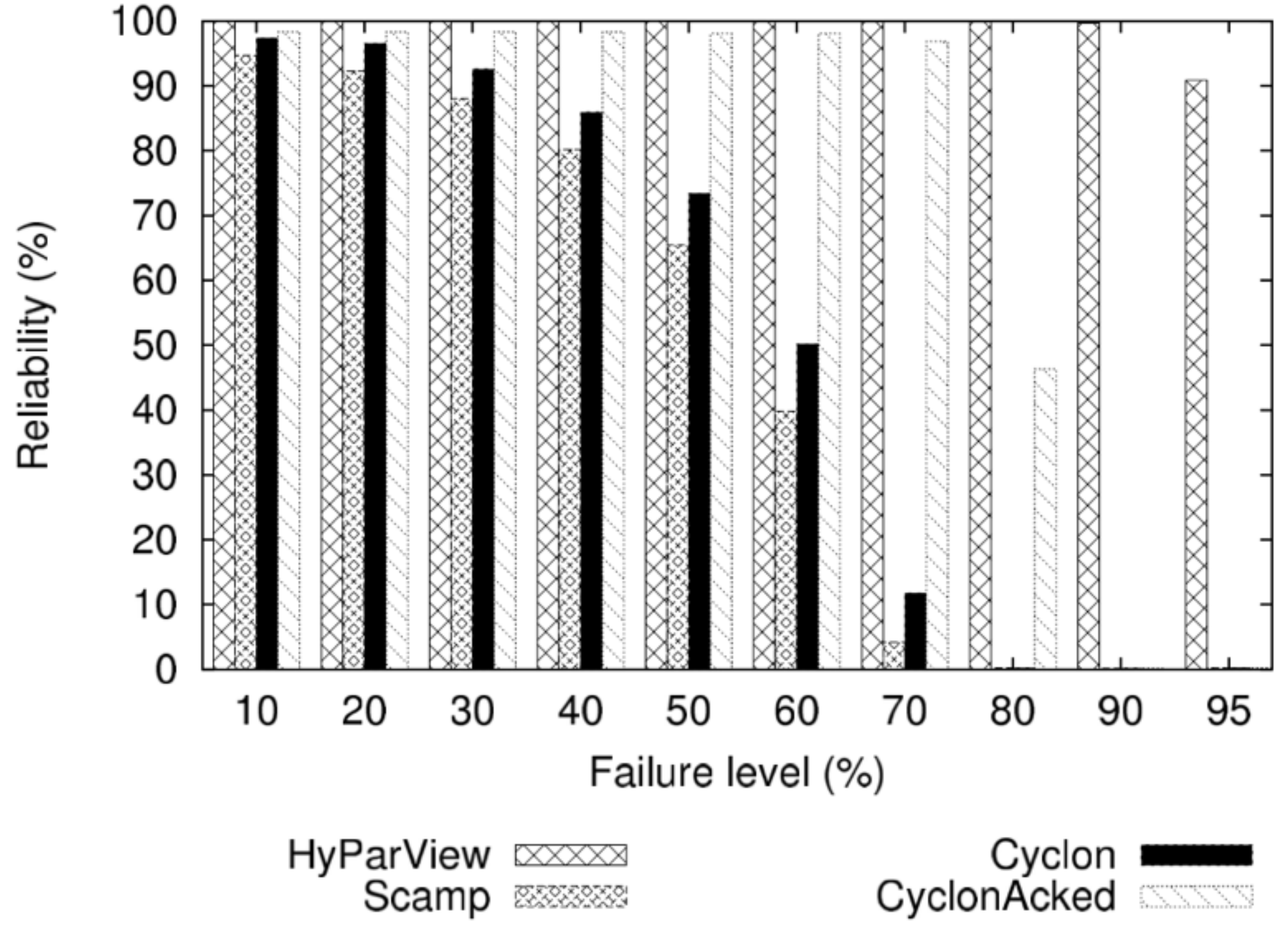


Figure 2: Reliability for 1000 messages

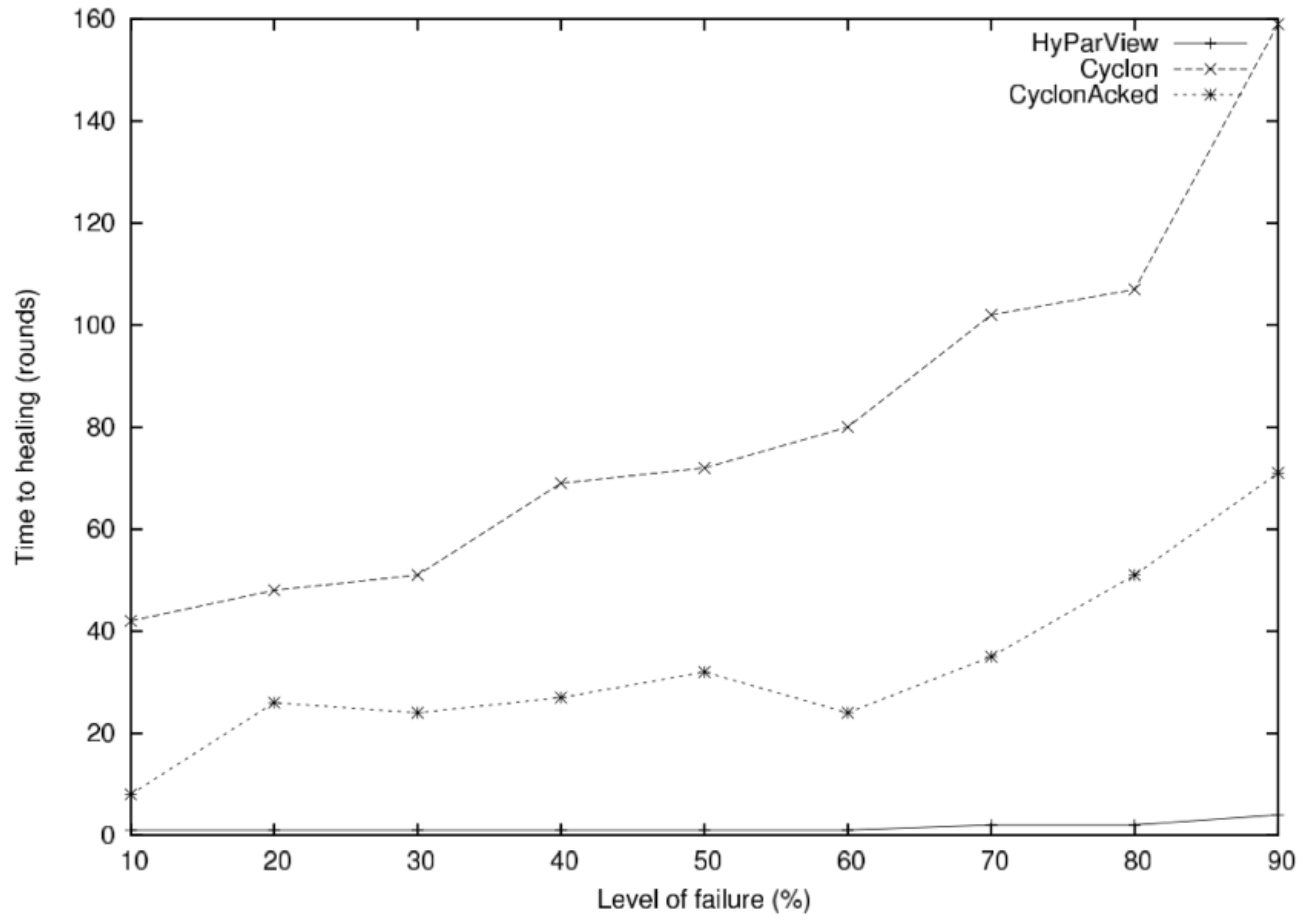


Figure 4: Healing time

WE CHOSE HYPARVIEW

- **Only** active view maintenance
- Passive view maintains **full membership** (unbounded)
- **Later**: switch to complete passive maintenance

COMCAST

Fulmine afflata
Magno hominum pavore, ſed vel nu
Pio Regis glorioſ: mem: GUSTAVI I.
A Superſtitionibus Papiſticis A: MDXXVII repurgata.
Iuſſu Ejusdem Regis & Regni Statoris,
Ne nimia propinquitate arci officeret,
A parte hac orientali truncata.
dini demtum, latitudine penſatum
utrinque extenſus.
Tandem juſſu
FRIDERICI,
Turris ſtupendæ altitudinis cum vetuſtate fatiſceret,
diruta,
Et ipsa ædes una cum tecto renovata.
Opus inchoatum A MDCCXXXVI, conſum̄atum MDCCXLIII,
hominis diſpendio ſtetit

DISSEMINATION PROTOCOLS

DISSEMINATION: DESIRABLE PROPERTIES

COMCAST

DISSEMINATION: DESIRABLE PROPERTIES

➡ Reliability

DISSEMINATION: DESIRABLE PROPERTIES

- ➡ Reliability
- ➡ Scalability

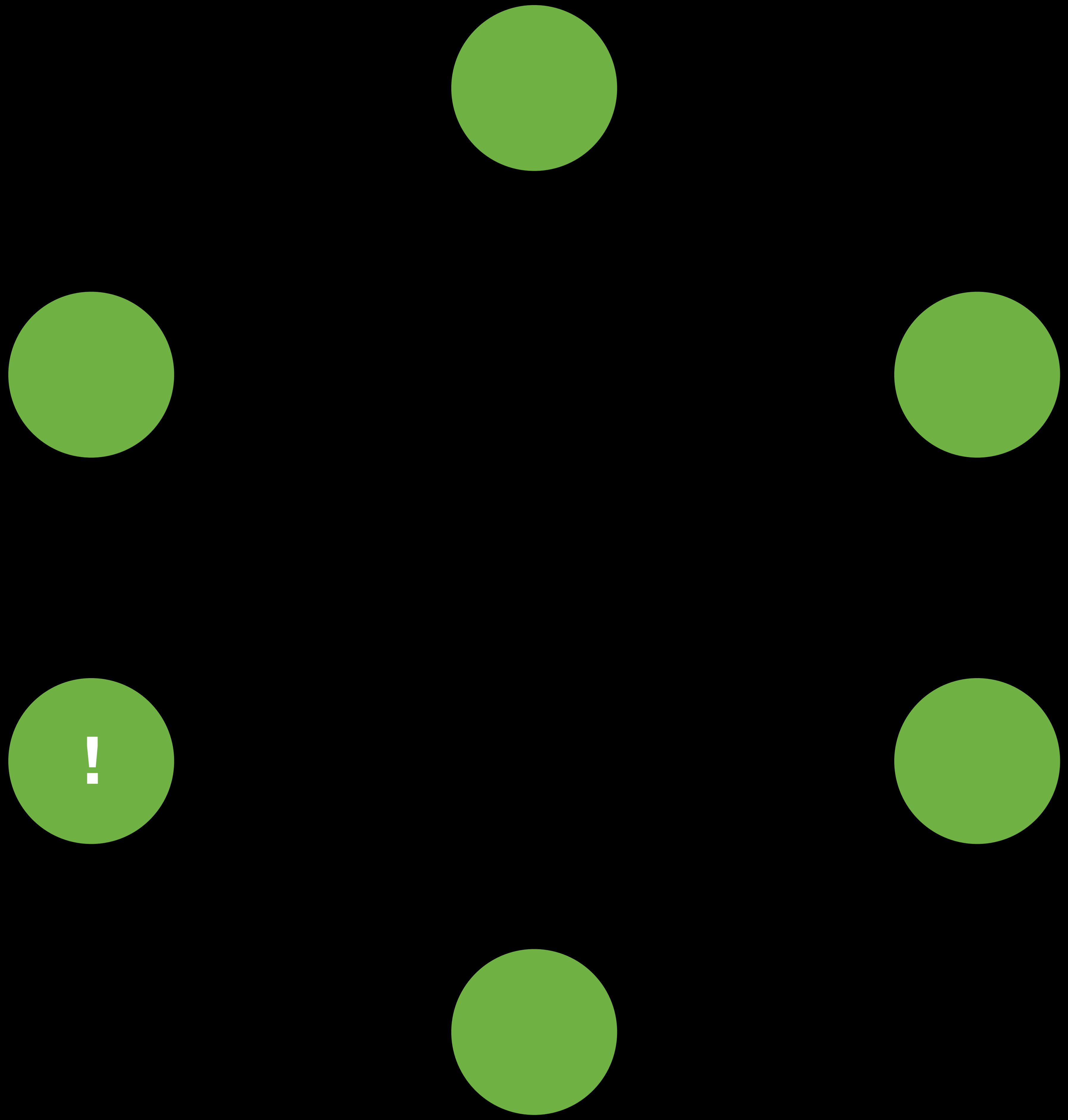
COMCAST

DISSEMINATION: DESIRABLE PROPERTIES

- ➡ Reliability
- ➡ Scalability
- ➡ Efficiency

COMCAST

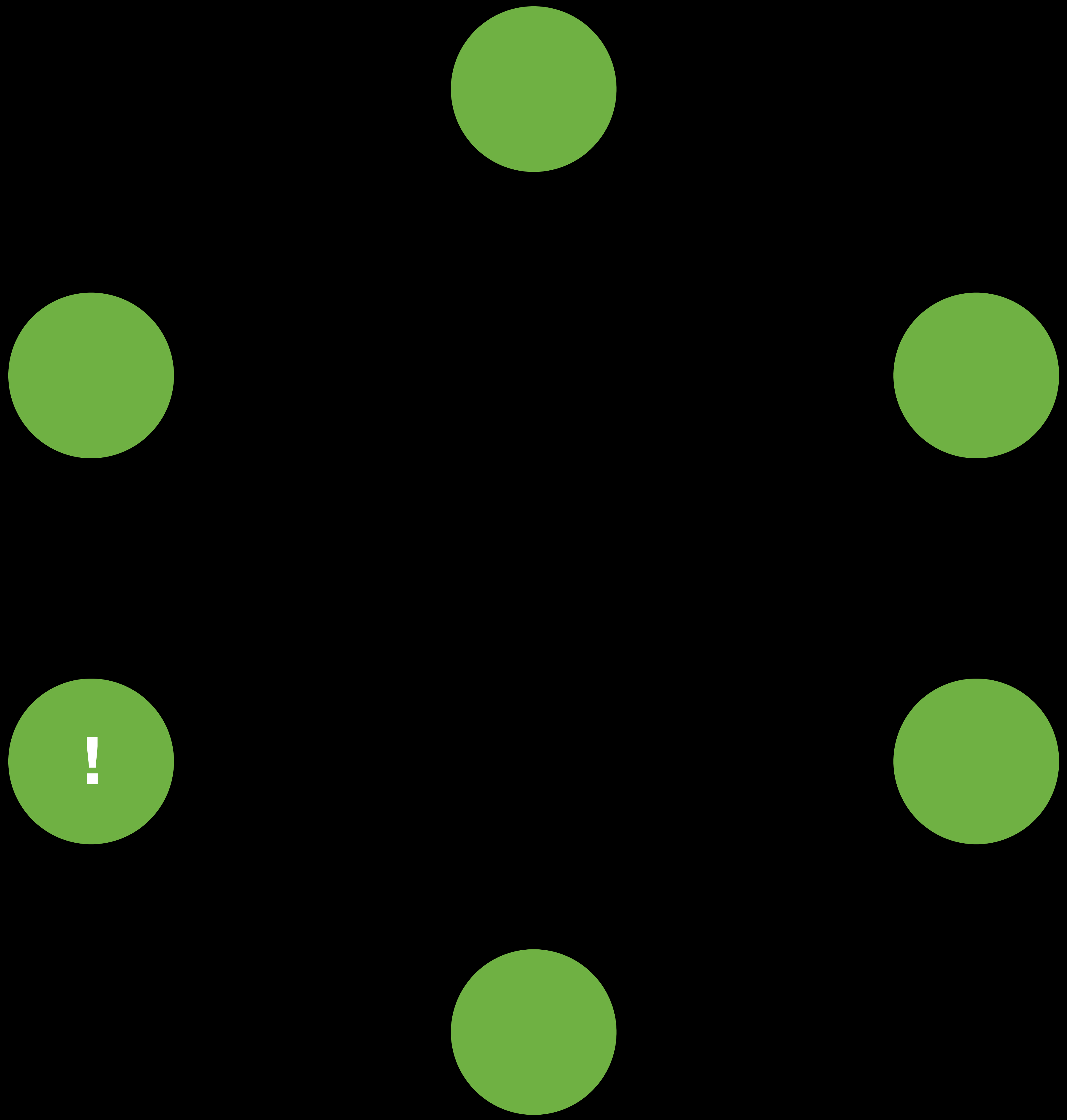
EPIDEMIC BROADCAST (GOSSIP)



COMCAST

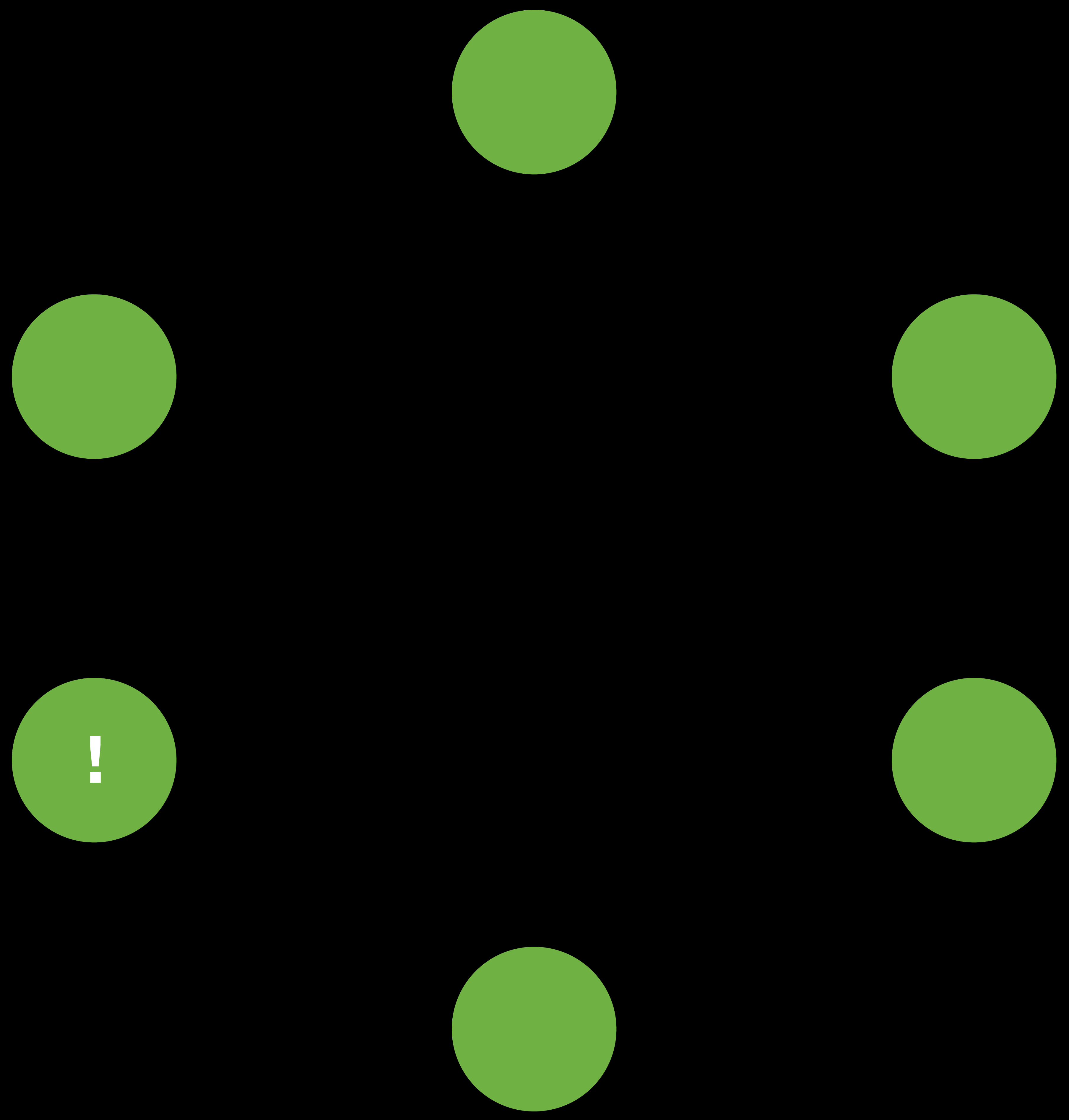
EPIDEMIC BROADCAST (GOSSIP)

- Send to random peers



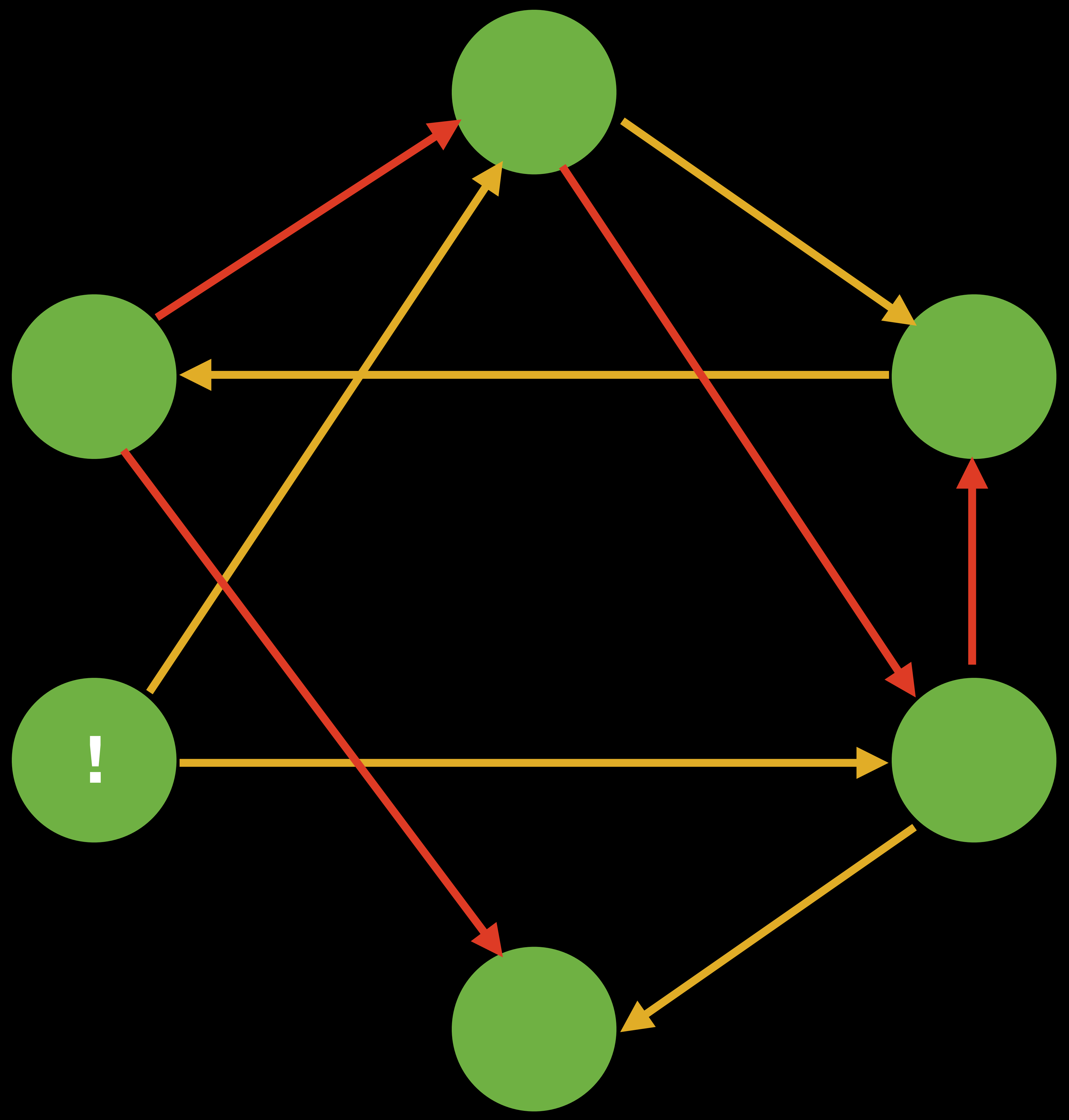
EPIDEMIC BROADCAST (GOSSIP)

- Send to random peers
- Messages rebroadcast by recipients



EPIDEMIC BROADCAST (GOSSIP)

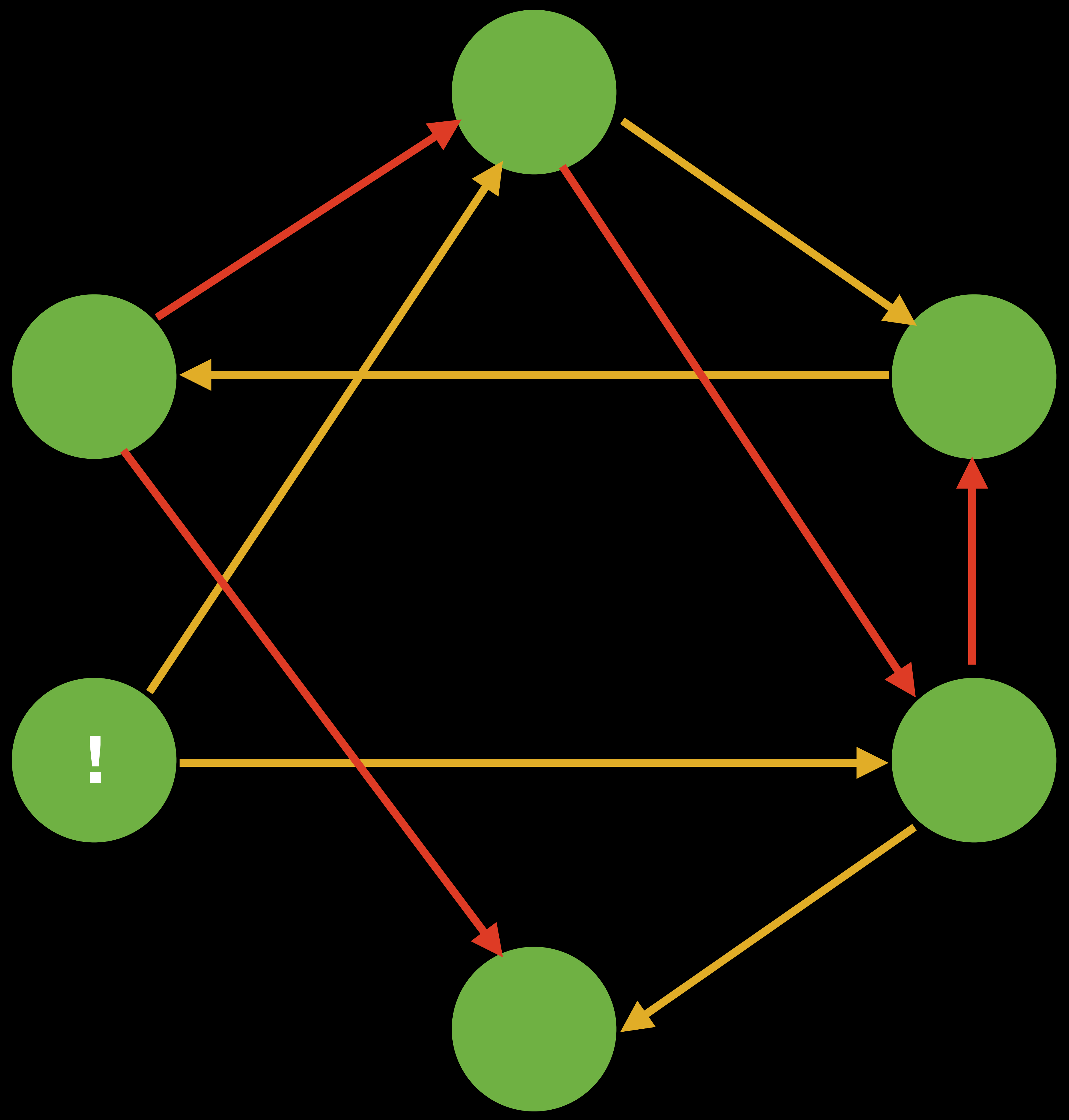
- Send to random peers
- Messages rebroadcast by recipients



COMCAST

EPIDEMIC BROADCAST (GOSSIP)

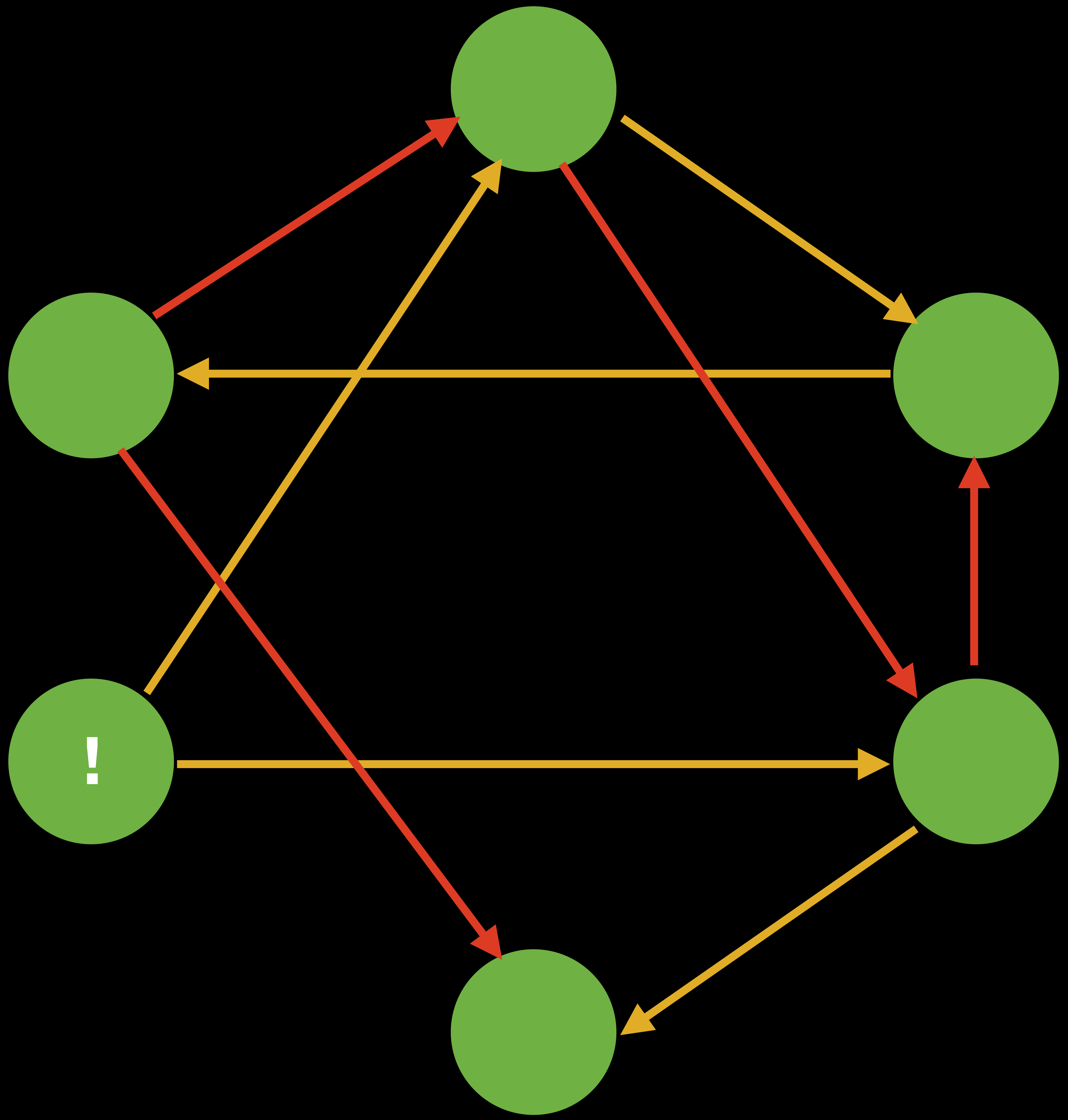
- Send to random peers
- Messages rebroadcast by recipients
- High redundancy



COMCAST

EPIDEMIC BROADCAST (GOSSIP)

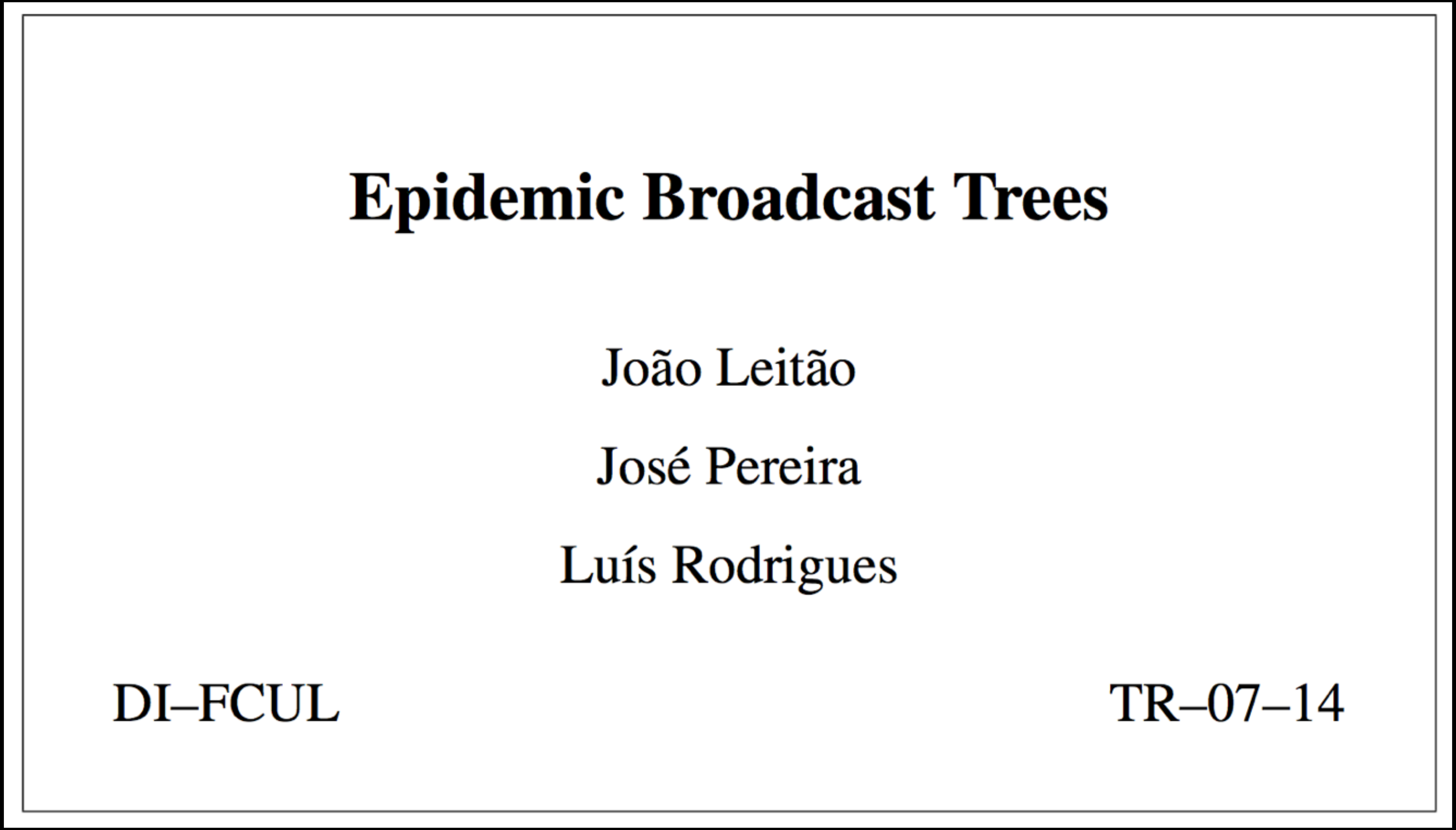
- Send to random peers
- Messages rebroadcast by recipients
- High redundancy
- Low scalability



COMCAST

WITHOUT REDUCING DELIVERY GUARANTEES, WE NEED

INCREASED EFFICIENCY



Epidemic Broadcast Trees

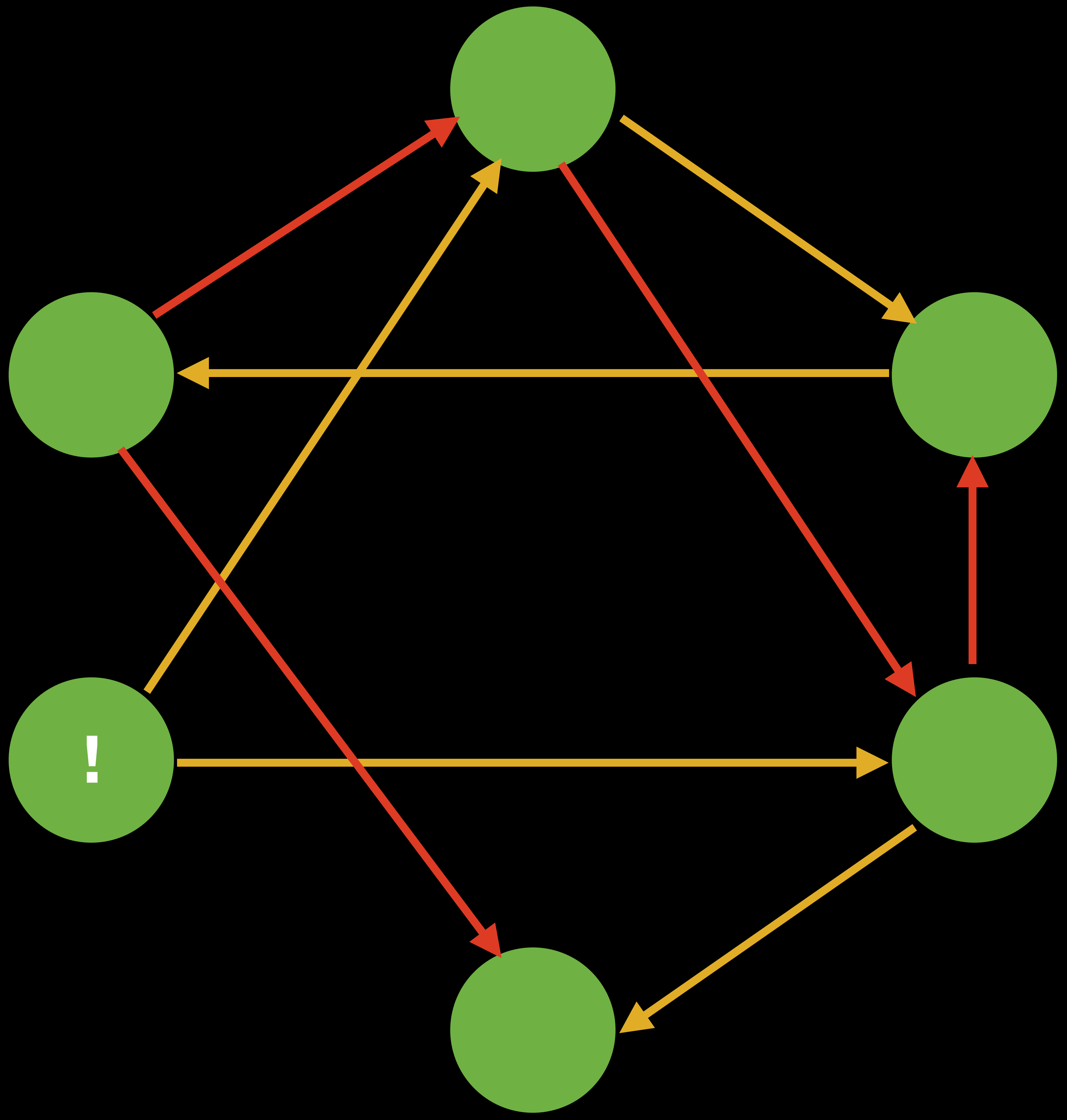
João Leitão

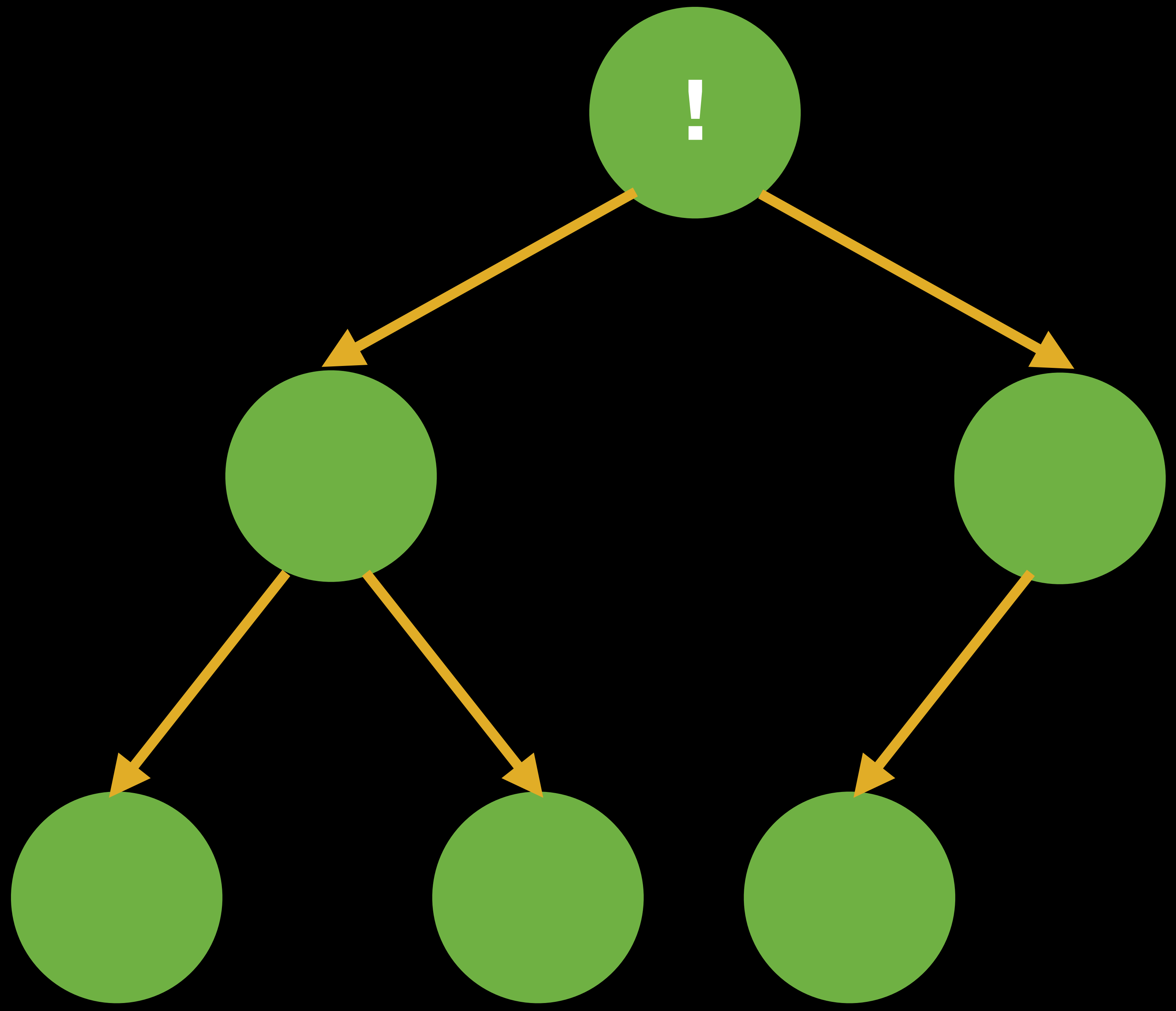
José Pereira

Luís Rodrigues

DI–FCUL TR–07–14

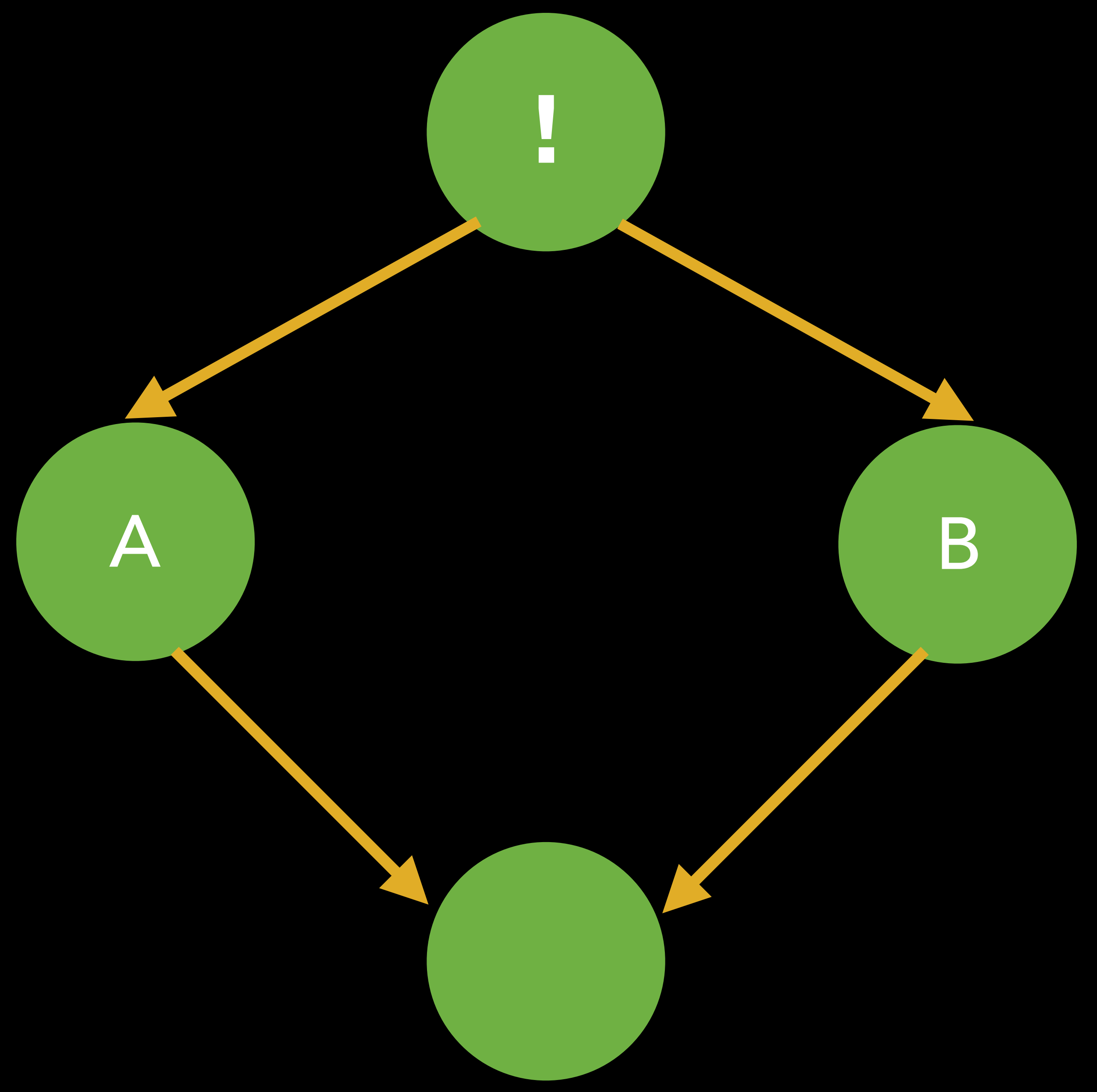
PLUMTREE - 2009





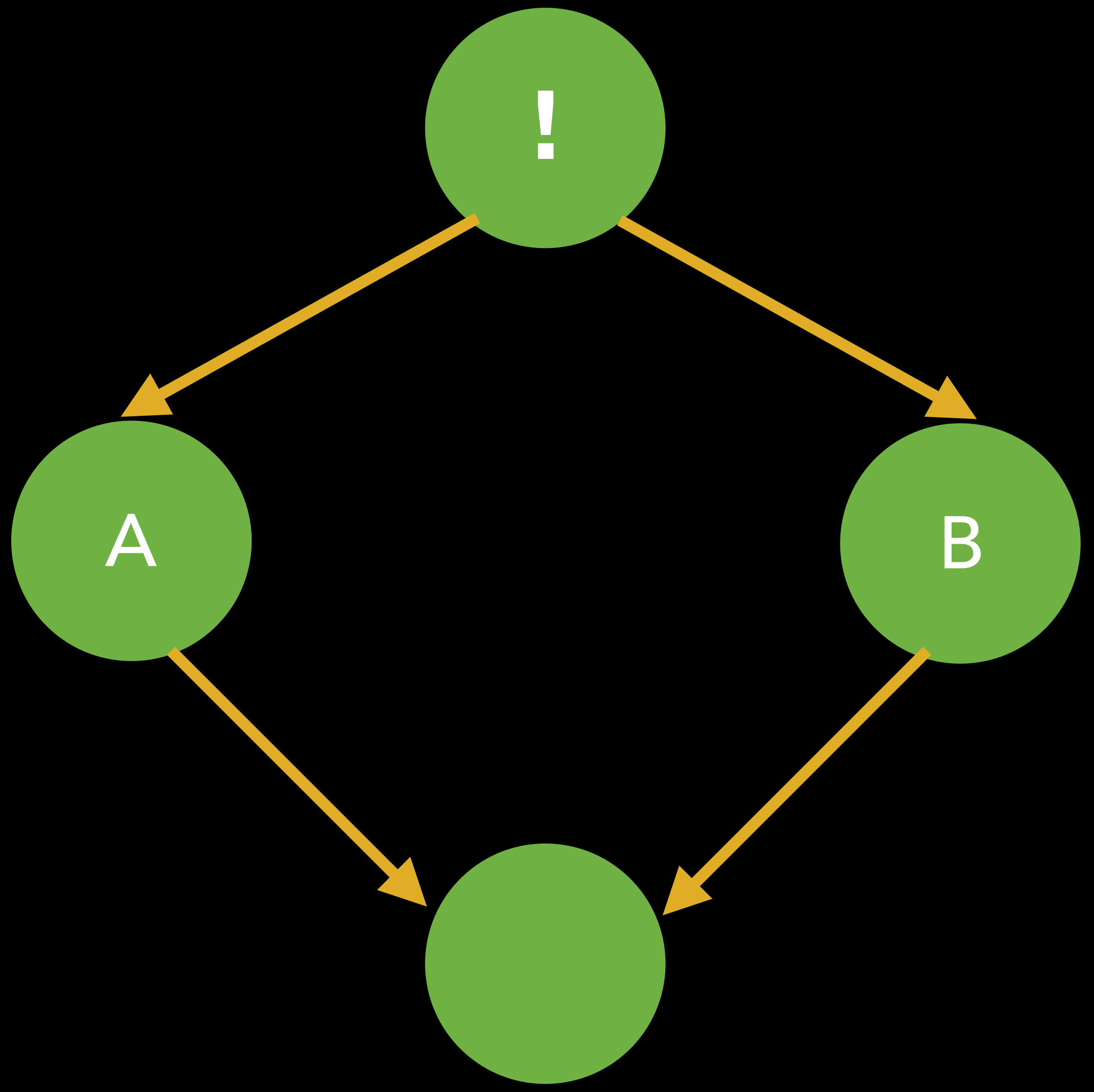
COMCAST

PLUMTREE - 2009 CONSTRUCTION



PLUMTREE - 2009 CONSTRUCTION

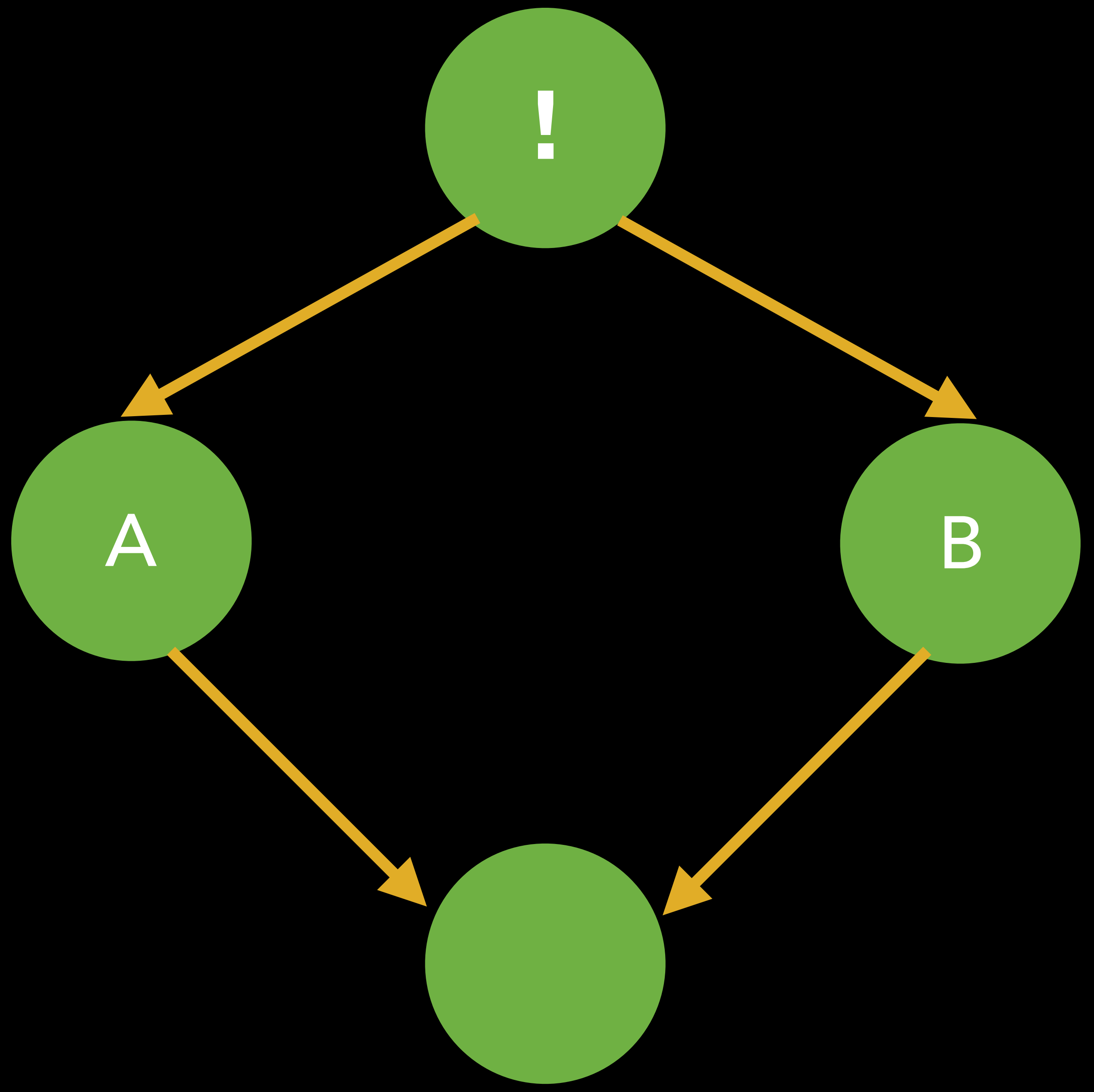
- All nodes start with full “eager” set



COMCAST

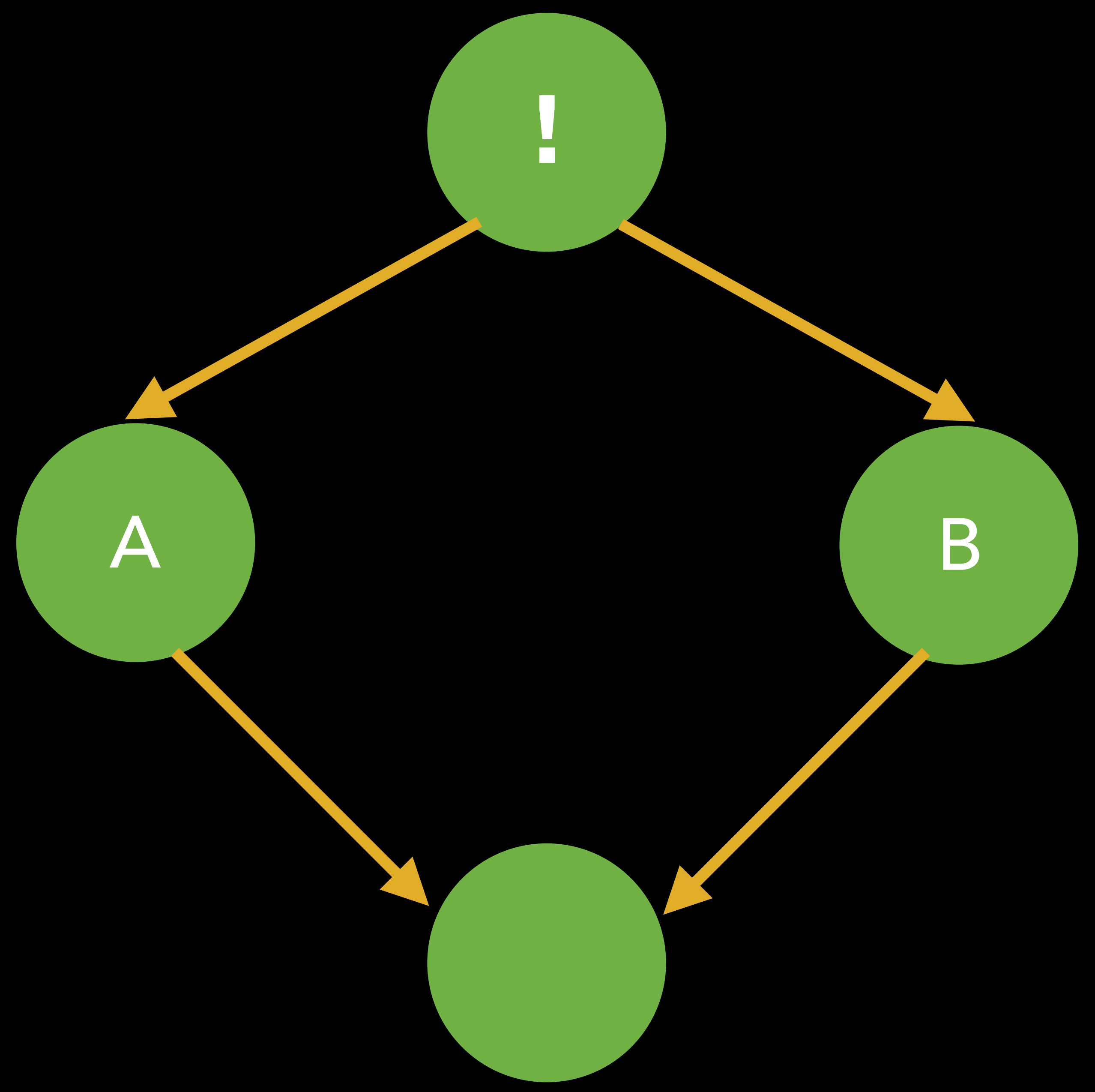
PLUMTREE - 2009 CONSTRUCTION

- All nodes start with full “eager” set
- Broadcast triggers eager-push



PLUMTREE - 2009 CONSTRUCTION

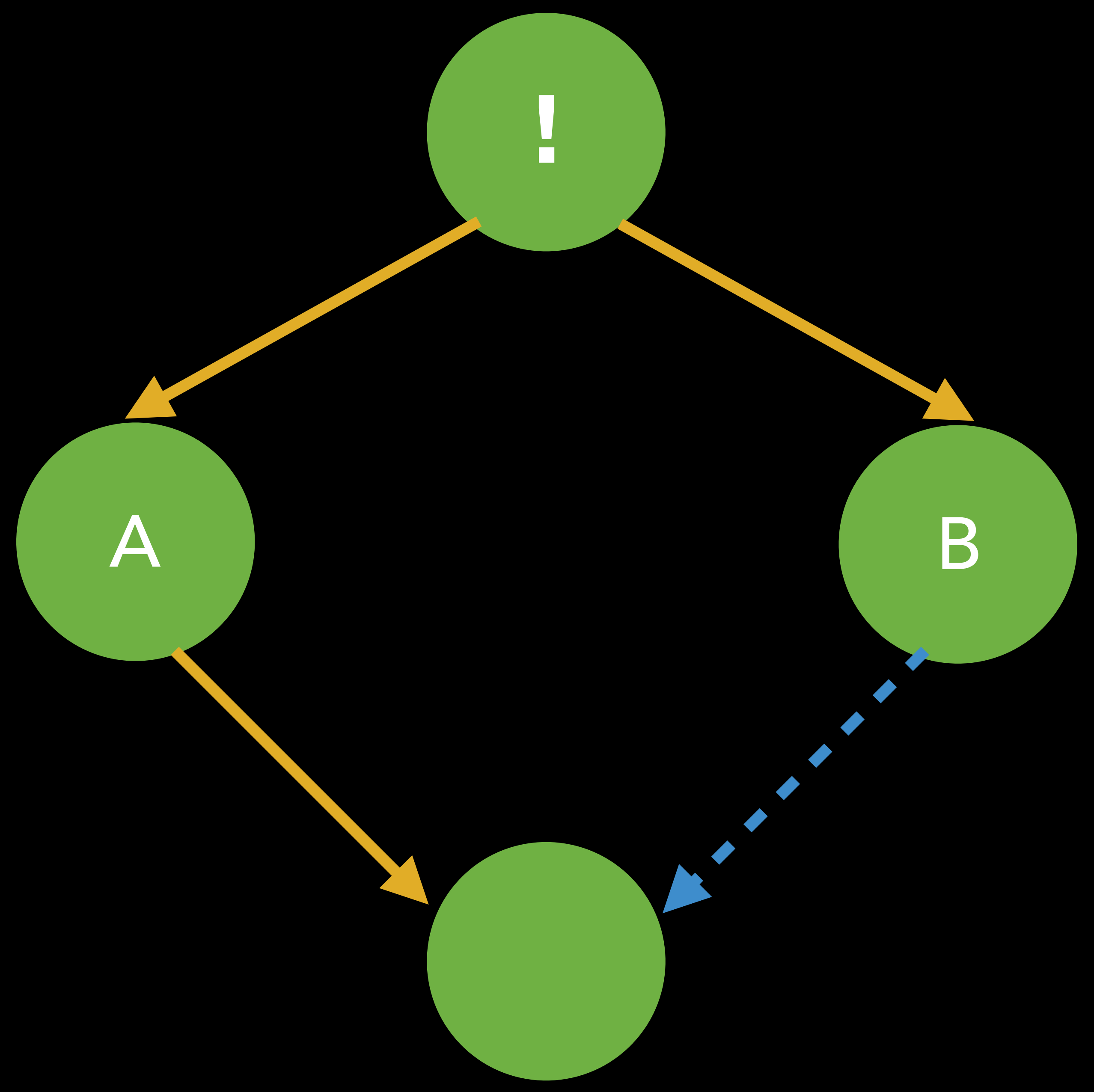
- All nodes start with full “eager” set
- Broadcast triggers eager-push
- Duplicate messages cause “pruning” (move to “lazy”)



COMCAST

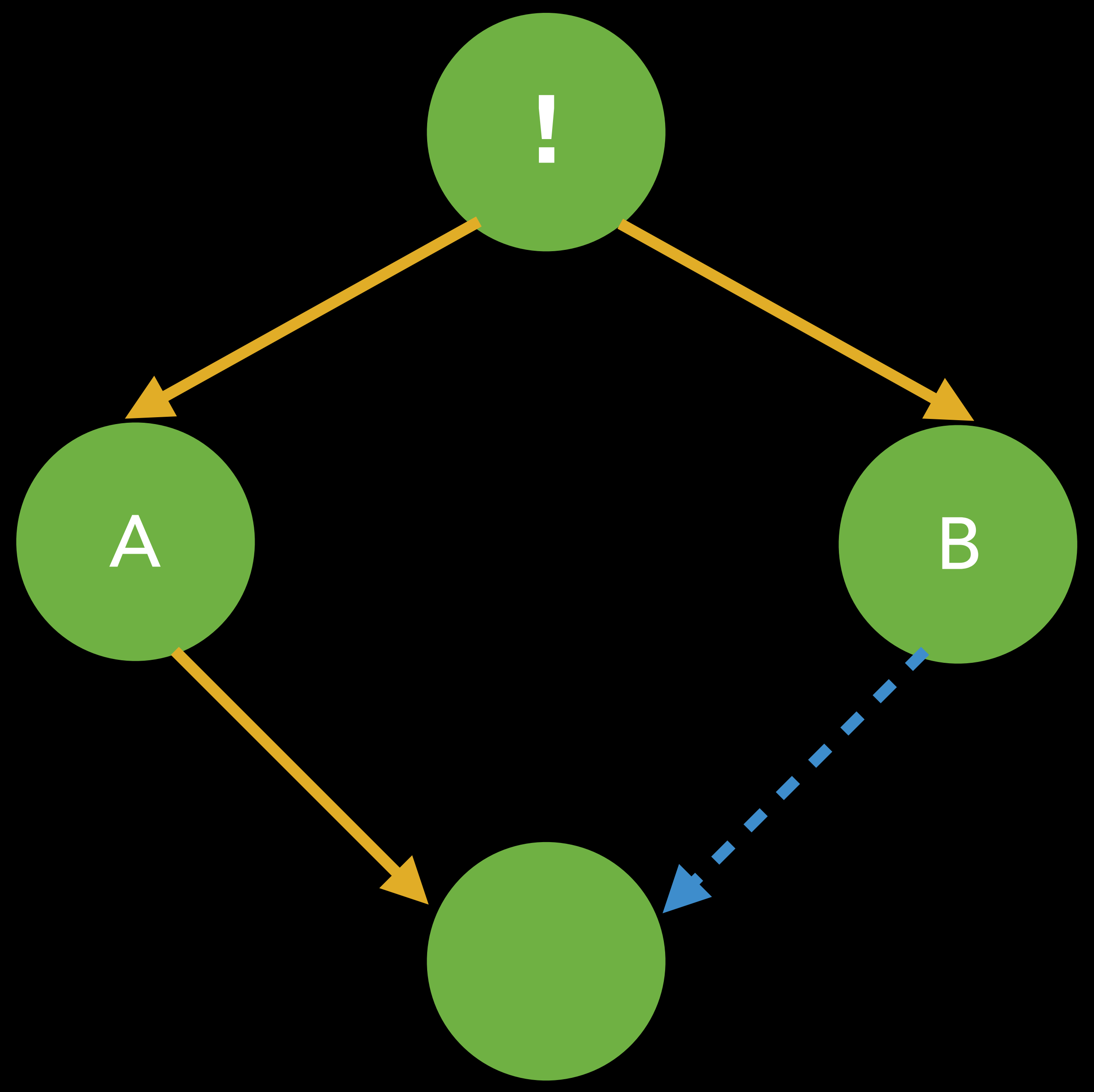
PLUMTREE - 2009 CONSTRUCTION

- All nodes start with full “eager” set
- Broadcast triggers eager-push
- Duplicate messages cause “pruning” (move to “lazy”)



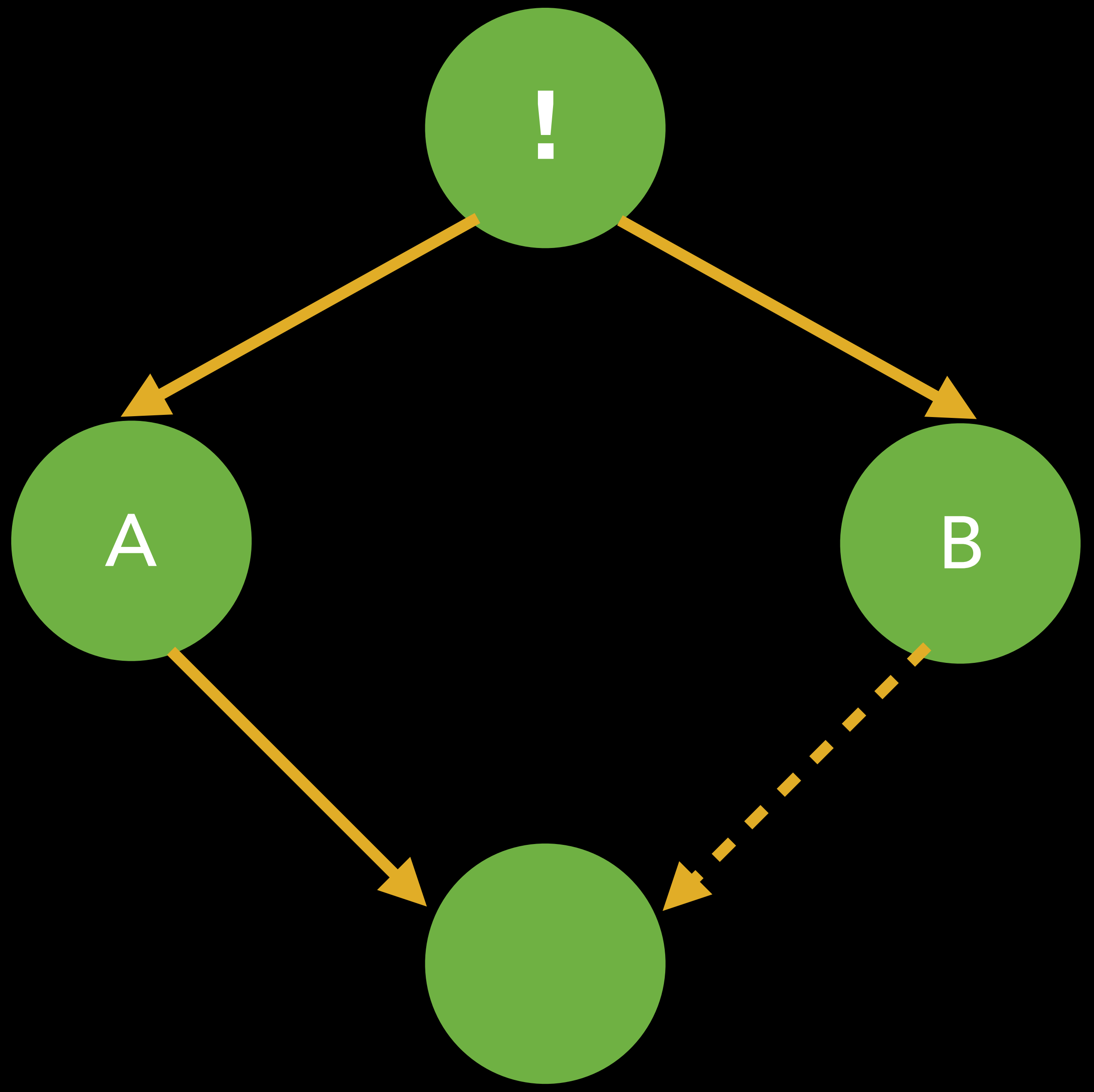
PLUMTREE - 2009 CONSTRUCTION

- All nodes start with full “eager” set
- Broadcast triggers eager-push
- Duplicate messages cause “pruning” (move to “lazy”)
- Regular broadcasts proceed with new “eager” sets



COMCAST

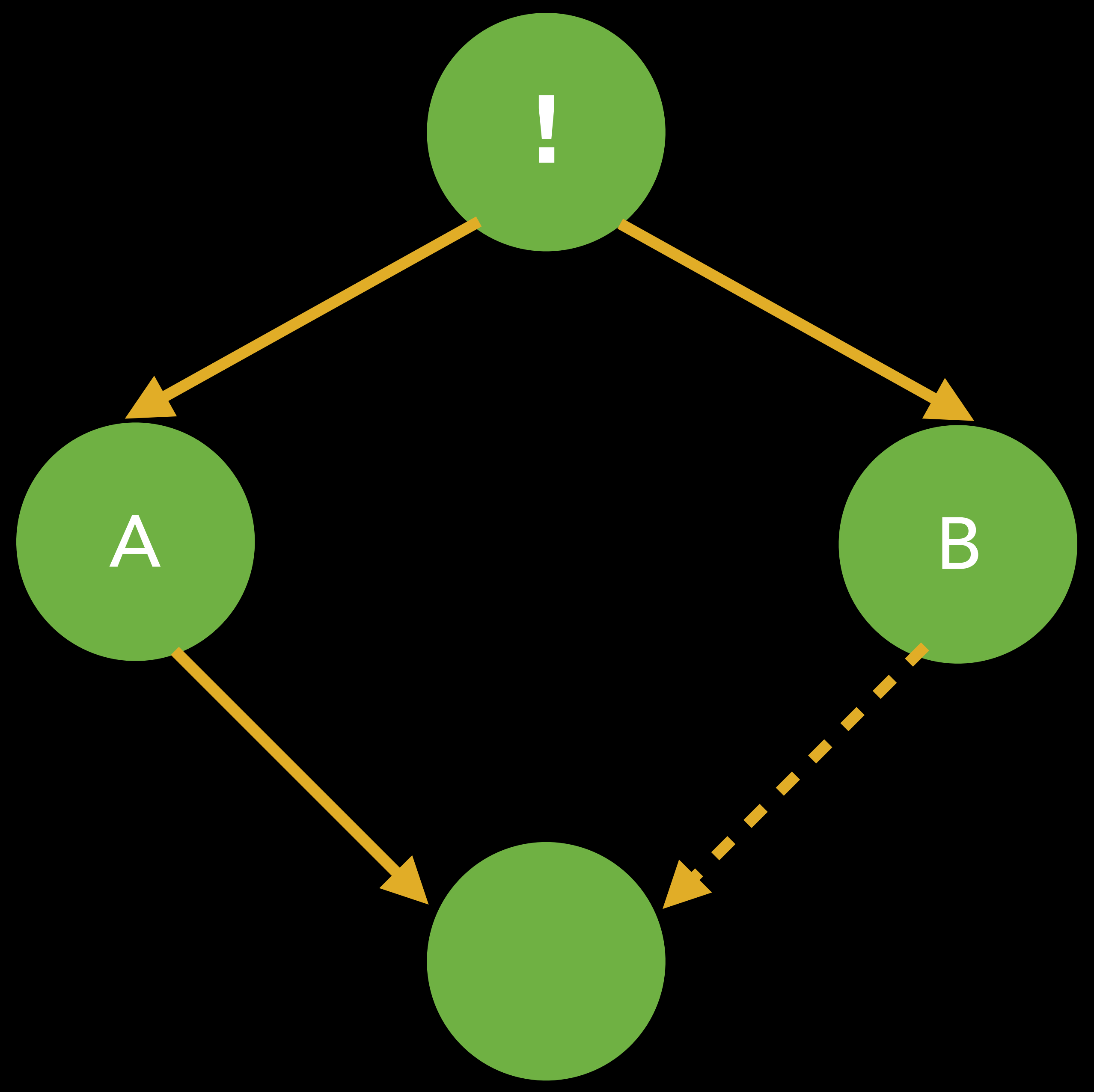
PLUMTREE - 2009 REPAIR



COMCAST

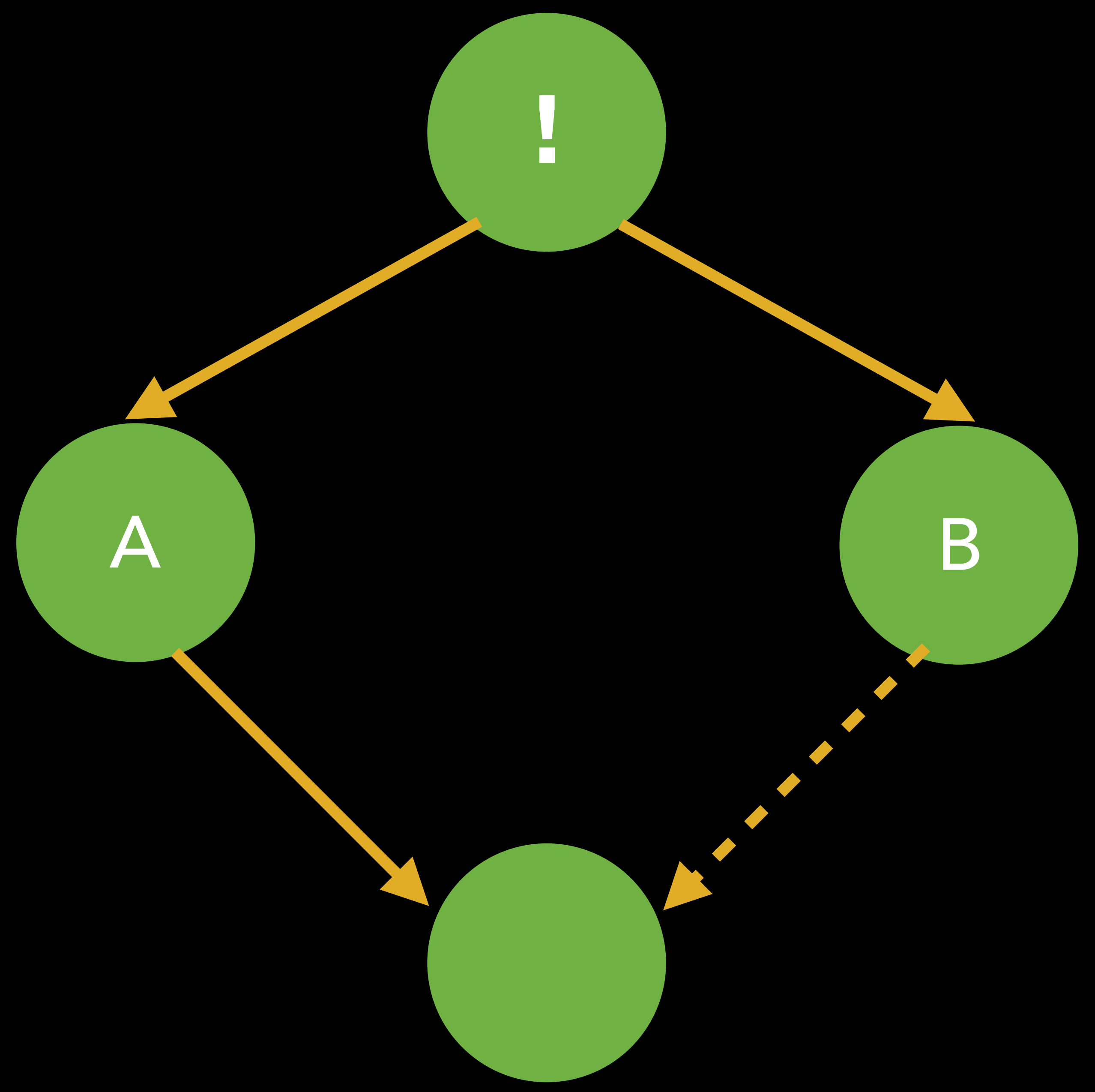
PLUMTREE - 2009 REPAIR

- Lazy-push sends “I Have” messages



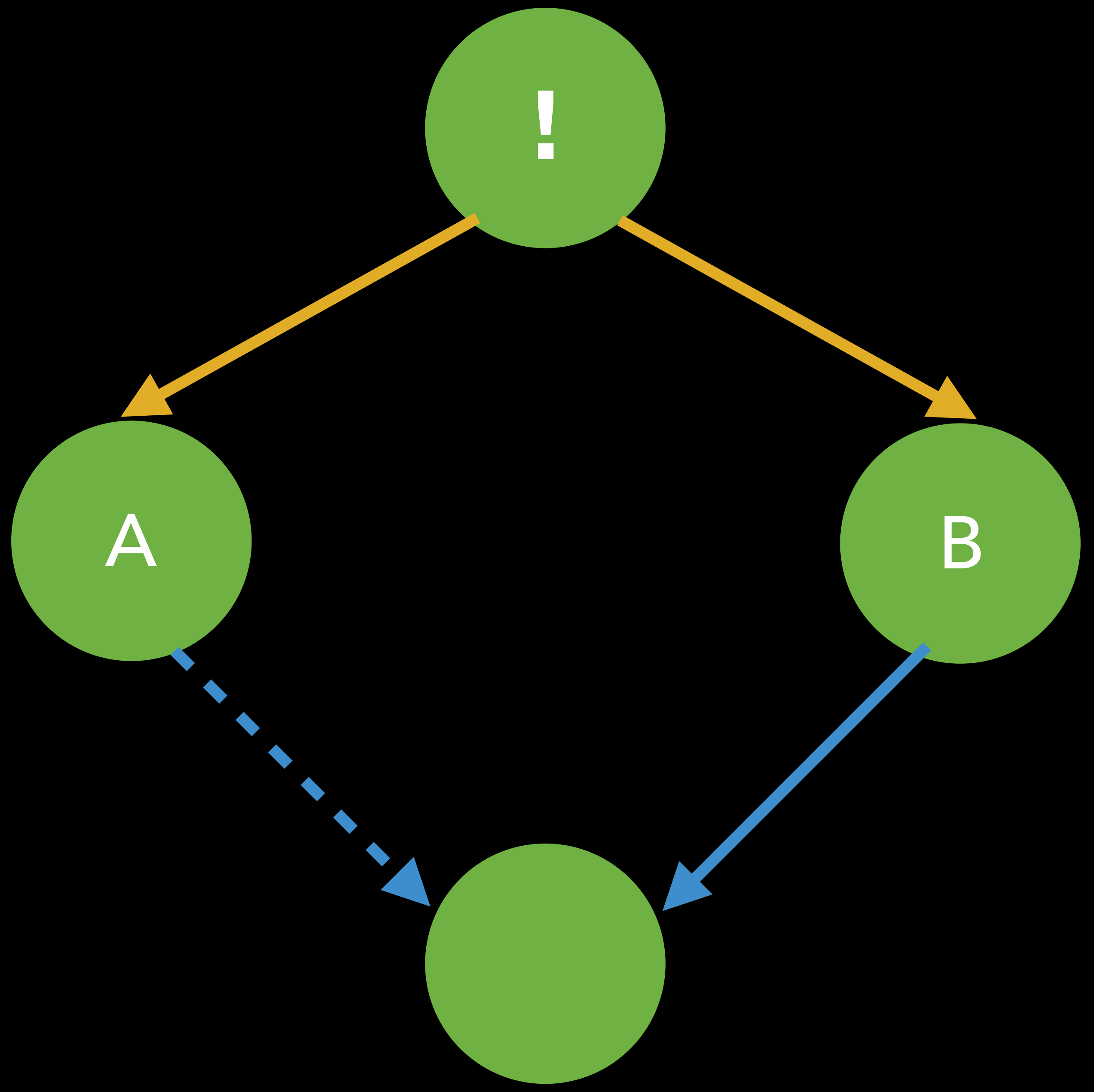
PLUMTREE - 2009 REPAIR

- Lazy-push sends “I Have” messages
- Timeout triggers “grafting” (move to “eager”)



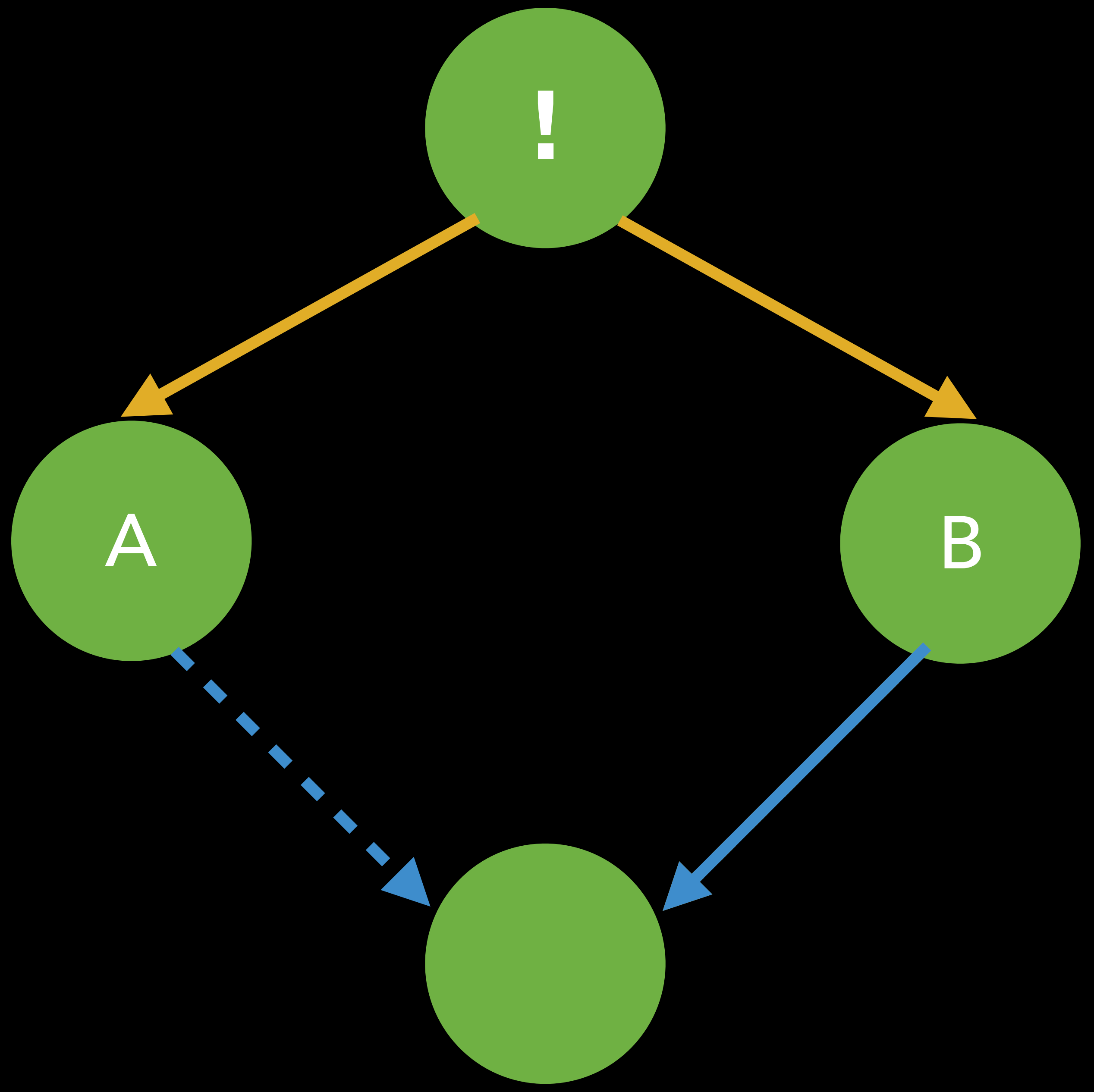
PLUMTREE - 2009 REPAIR

- Lazy-push sends “I Have” messages
- Timeout triggers “grafting” (move to “eager”)



PLUMTREE - 2009 REPAIR

- Lazy-push sends “I Have” messages
- Timeout triggers “grafting” (move to “eager”)
- Lazy-push batched to reduce overhead



WE CHOSE PLUMTREE

COMCAST

WE CHOSE PLUMTREE

- **Good tradeoff** between reliability and redundancy

WE CHOSE PLUMTREE

- **Good tradeoff** between reliability and redundancy
- Optimizes for **lowest-latency paths**

WE CHOSE PLUMTREE

- **Good tradeoff** between reliability and redundancy
- Optimizes for **lowest-latency paths**
- **Existing** open-source implementations

COMCAST

WE CHOSE PLUMTREE

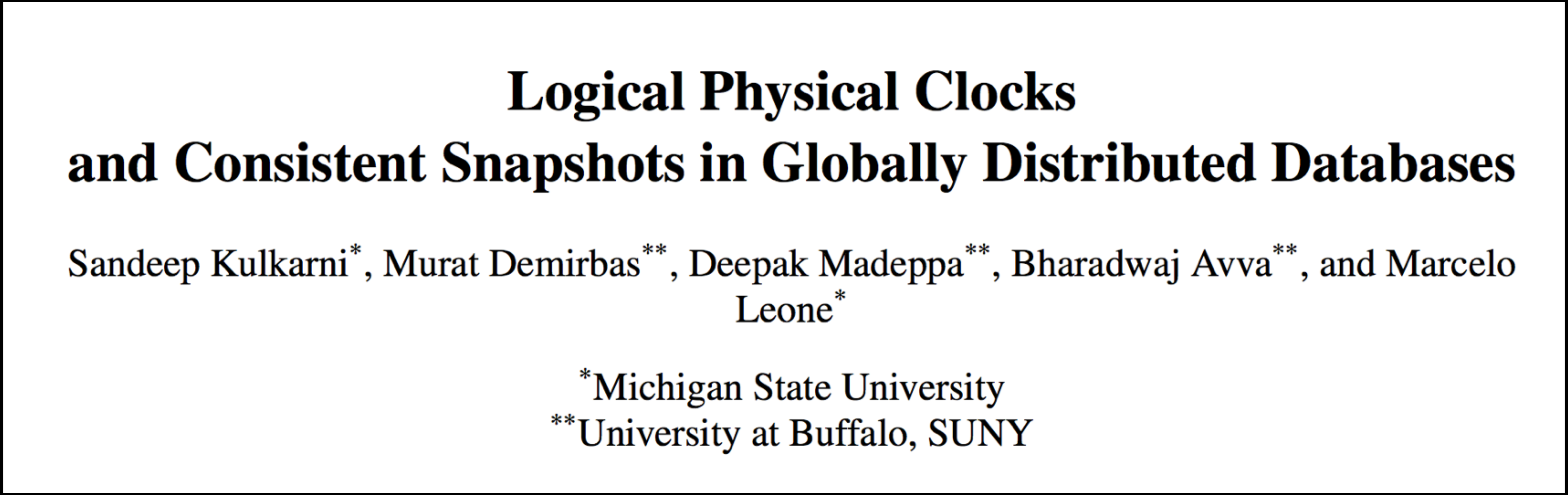
- **Good tradeoff** between reliability and redundancy
- Optimizes for **lowest-latency paths**
- **Existing** open-source implementations
- **Excellent fit** with HyParView

COMCAST

POPULATION PROTOCOLS

POPULATION PROTOCOLS USE

RANDOMIZED INTERACTIONS



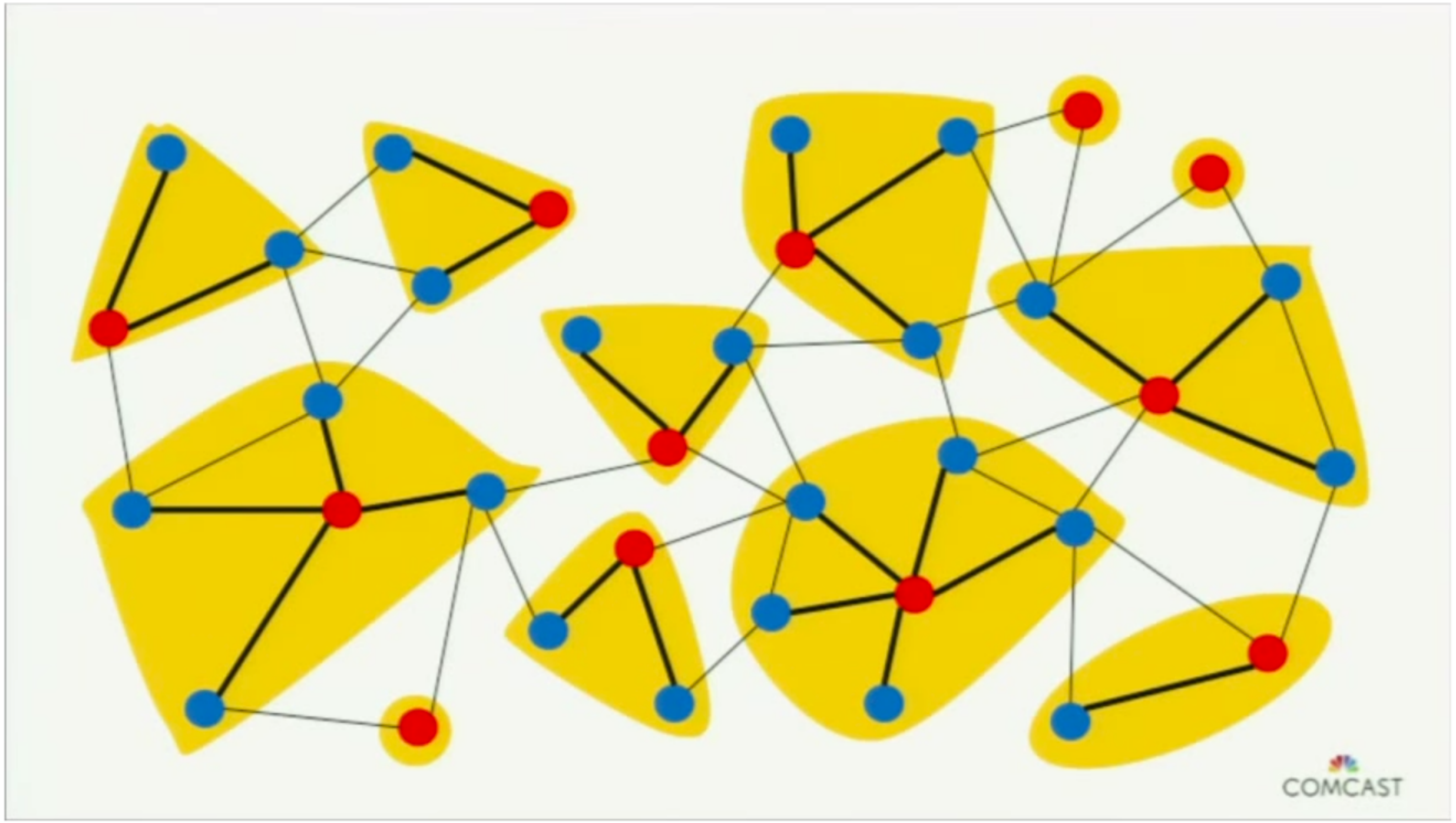
Logical Physical Clocks and Consistent Snapshots in Globally Distributed Databases

Sandeep Kulkarni[*], Murat Demirbas[**], Deepak Madeppa[**], Bharadwaj Avva[**], and Marcelo Leone[*]

[*]Michigan State University
[**]University at Buffalo, SUNY

JON MOORE

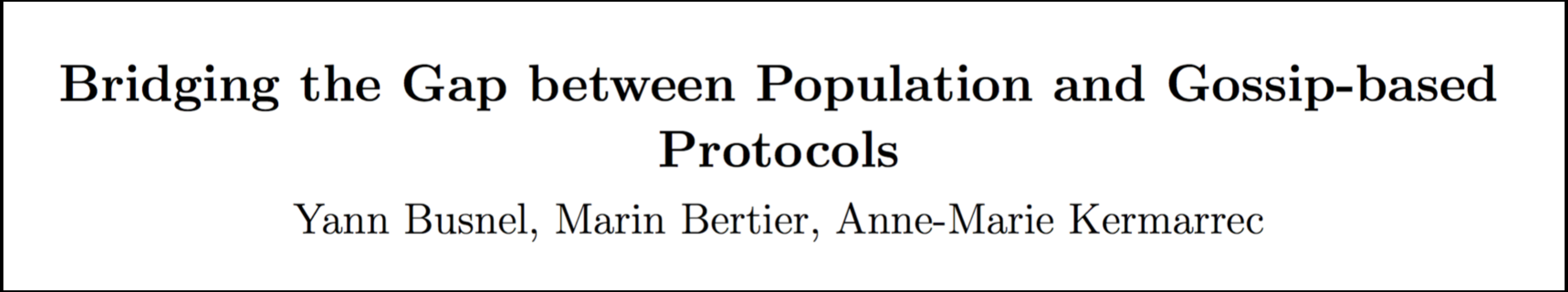
DISTRIBUTED MONOTONIC CLOCKS



Vidcap from StrangeLoop 2015: https://youtu.be/YqNGbvFHoKM

DMC PROBLEMS

- “Wacky clock mode”
- Hierarchy imbalances load
- Long-lived partitions
- No convergence proof



Bridging the Gap between Population and Gossip-based Protocols

Yann Busnel, Marin Bertier, Anne-Marie Kermarrec

COMCAST

APPLYING DMC

- Use **existing dissemination** with DMC
- Transmit clocks **along with other messages**
- Use monotonic clocks as a **drift-detection mechanism**

LESSONS LEARNED

Photo by Chris Meiklejohn

Vidcap from RICON West: https://youtu.be/s4cCUTPU8GI

Photo by Comcast

THANK YOU!

@SEANCRIBBS

COMCAST