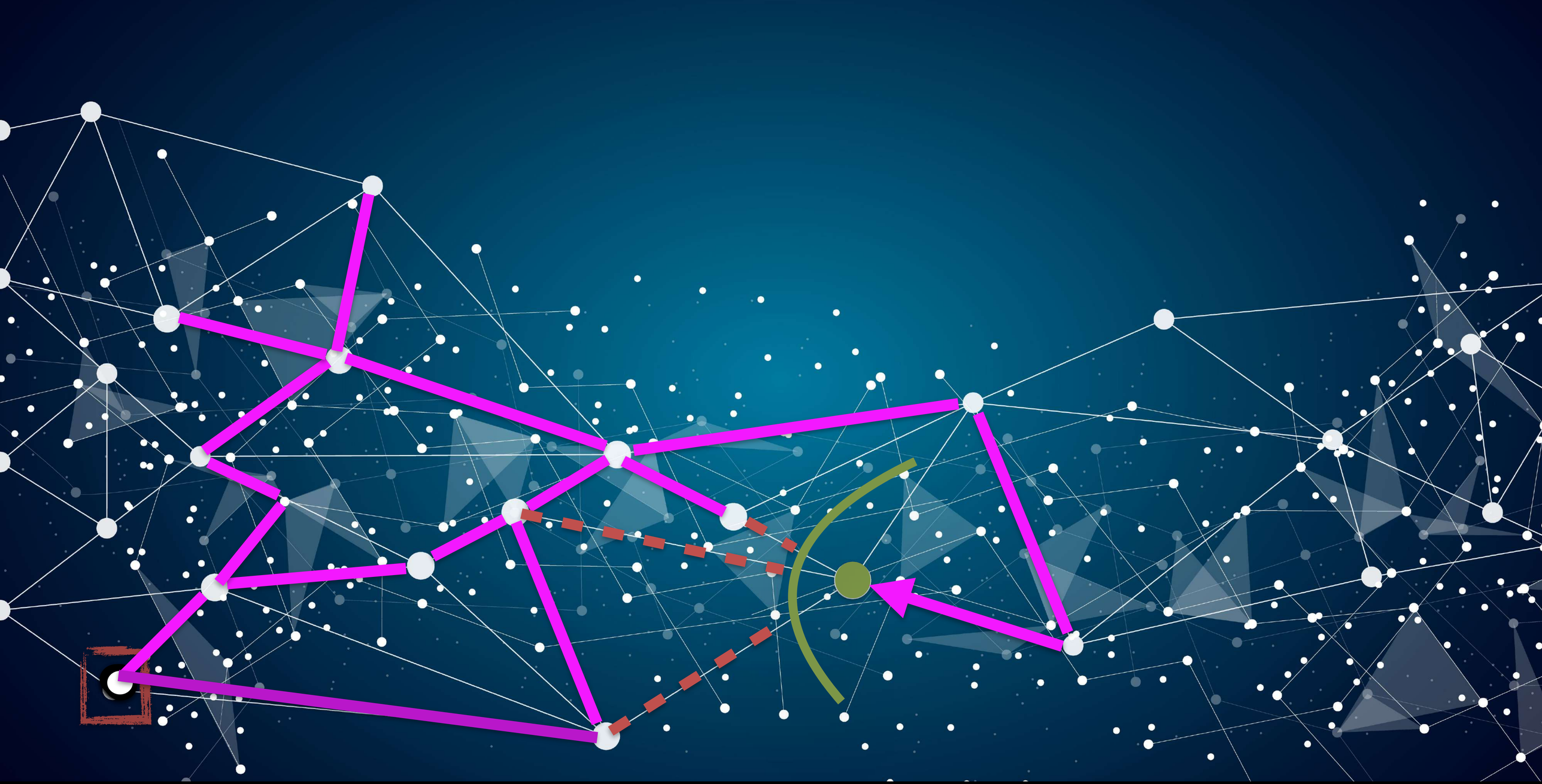
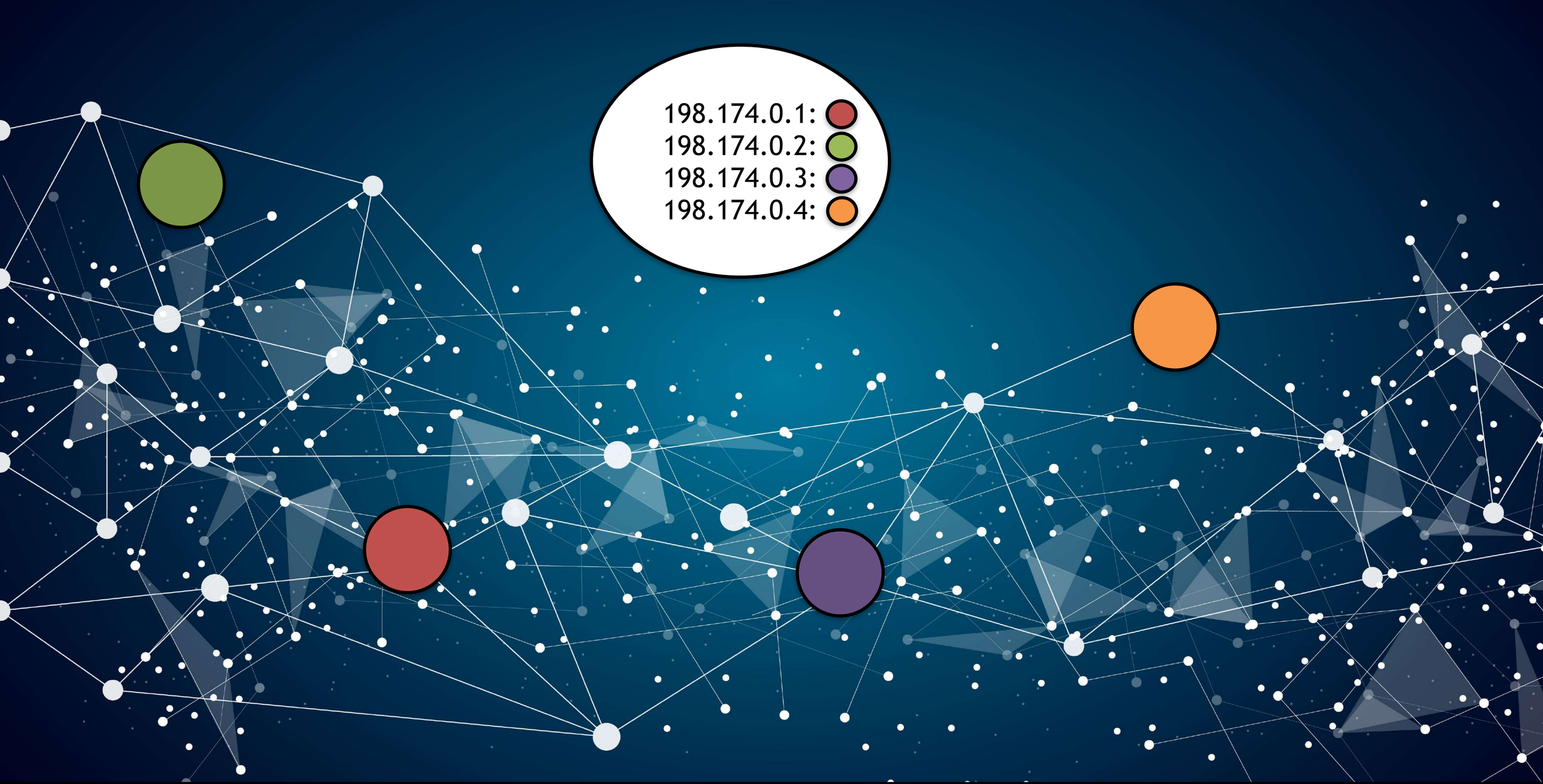
Understanding Gossip Protocols

Félix López Luis - @flopezluis

The best way to understand them is knowing the problems they solve



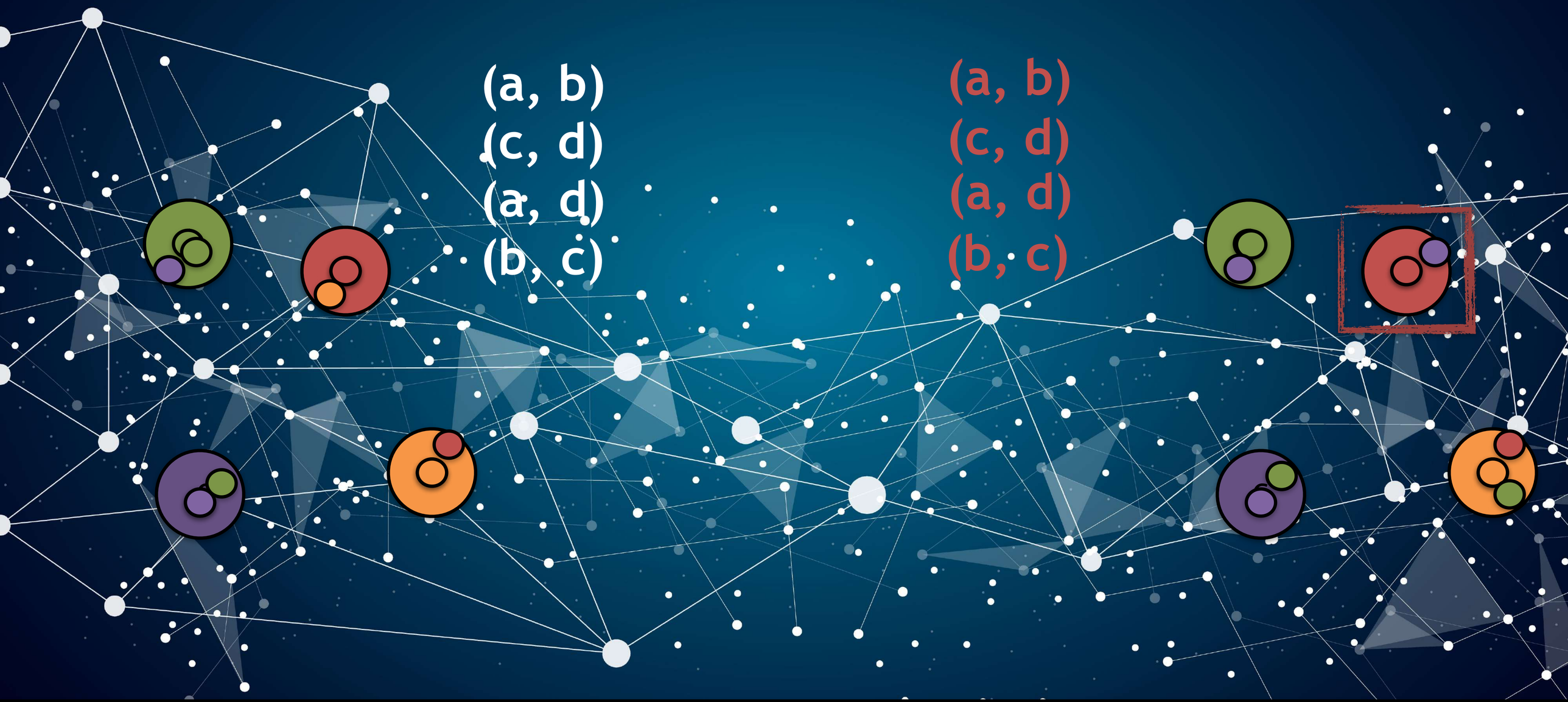
@flopezluis


198.174.0.1:
198.174.0.2:
198.174.0.3:
198.174.0.4:

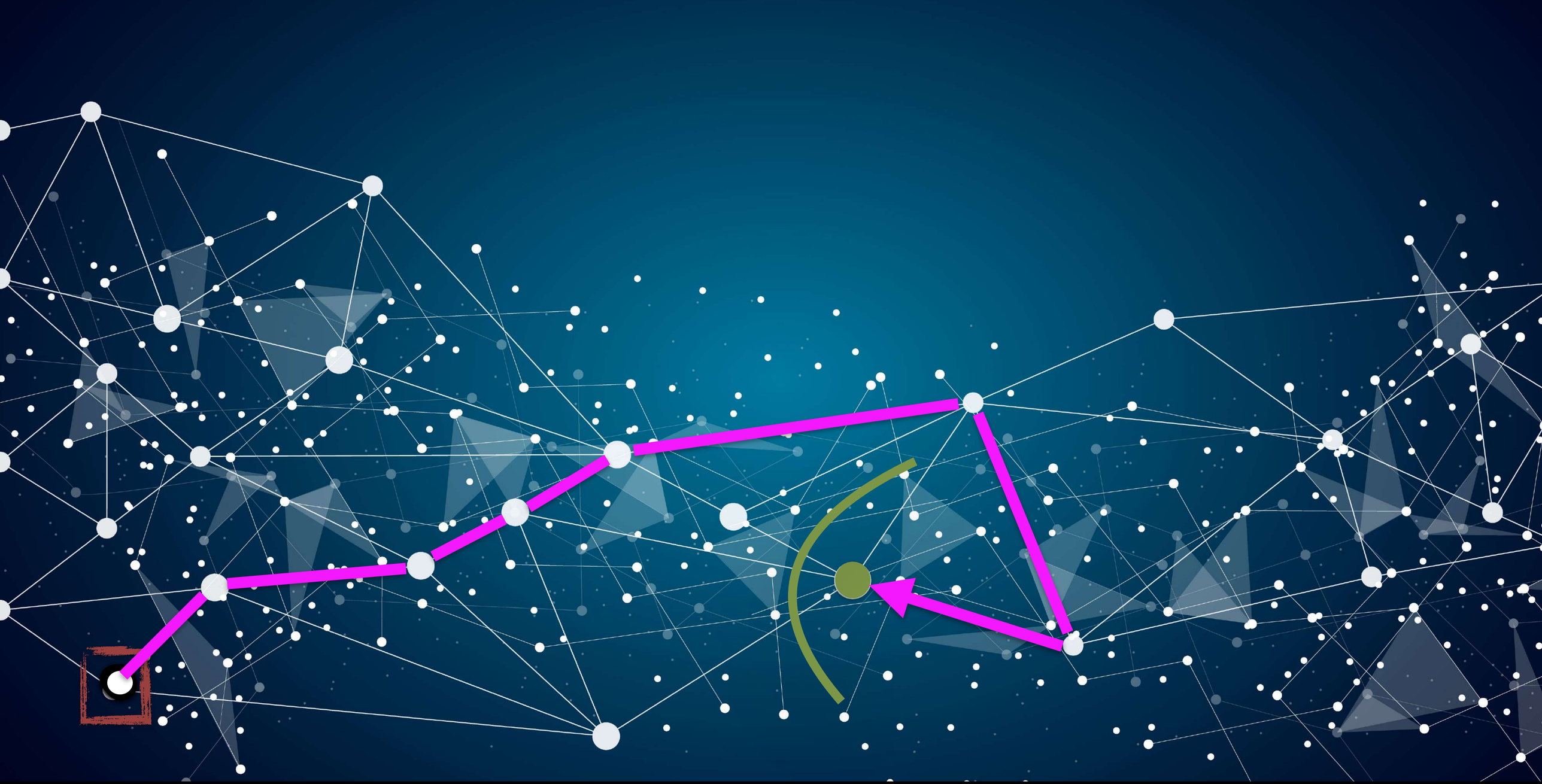
2n - 4

(a, b)
(c, d)
(a, d)
(b, c)

(a, b)
(c, d)
(a, d)
(b, c)



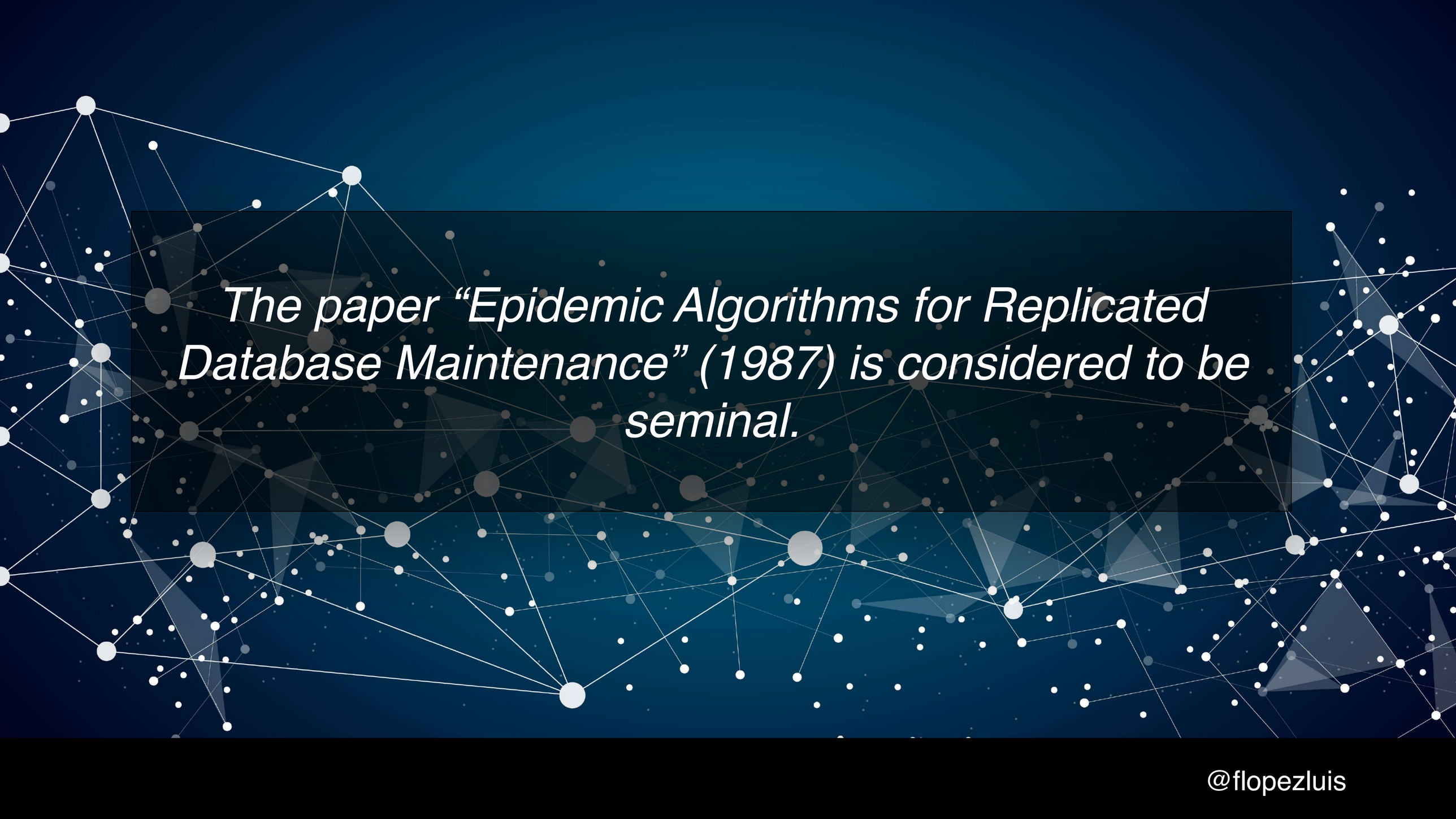
@flopezluis



@flopezluis

What are they?


They are a communication protocol, a broadcast protocol. We want to communicate a message to all the nodes in the network.

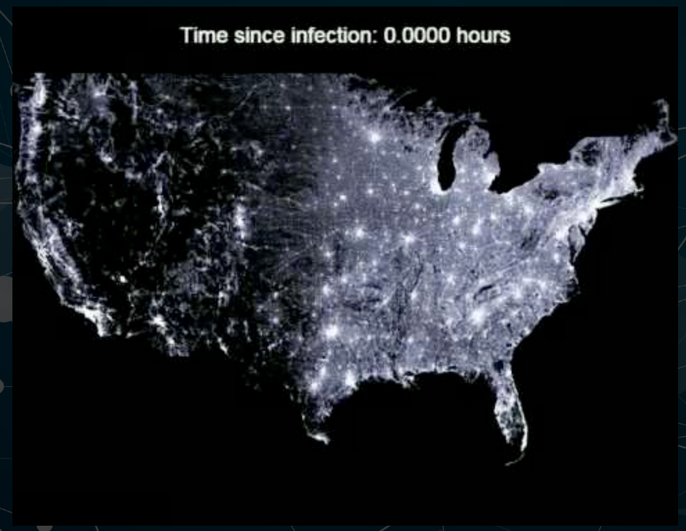

The paper “Epidemic Algorithms for Replicated Database Maintenance” (1987) is considered to be seminal.

- *They were trying to build a distributed K/V*
- *It lays the foundations of Gossip Protocols.*
- *Lots of concepts that we use today come from them.*
  - *Anti-entropy*
  - *Rumor Mongering*
  - *push / pull strategies*
  - *Fanout*
- *First analysis of reliability of these protocols.*


They’re inspired by:
Epidemics, human gossip, social networks


Anyone can start a rumor, but none can stop one.
~ American proverb

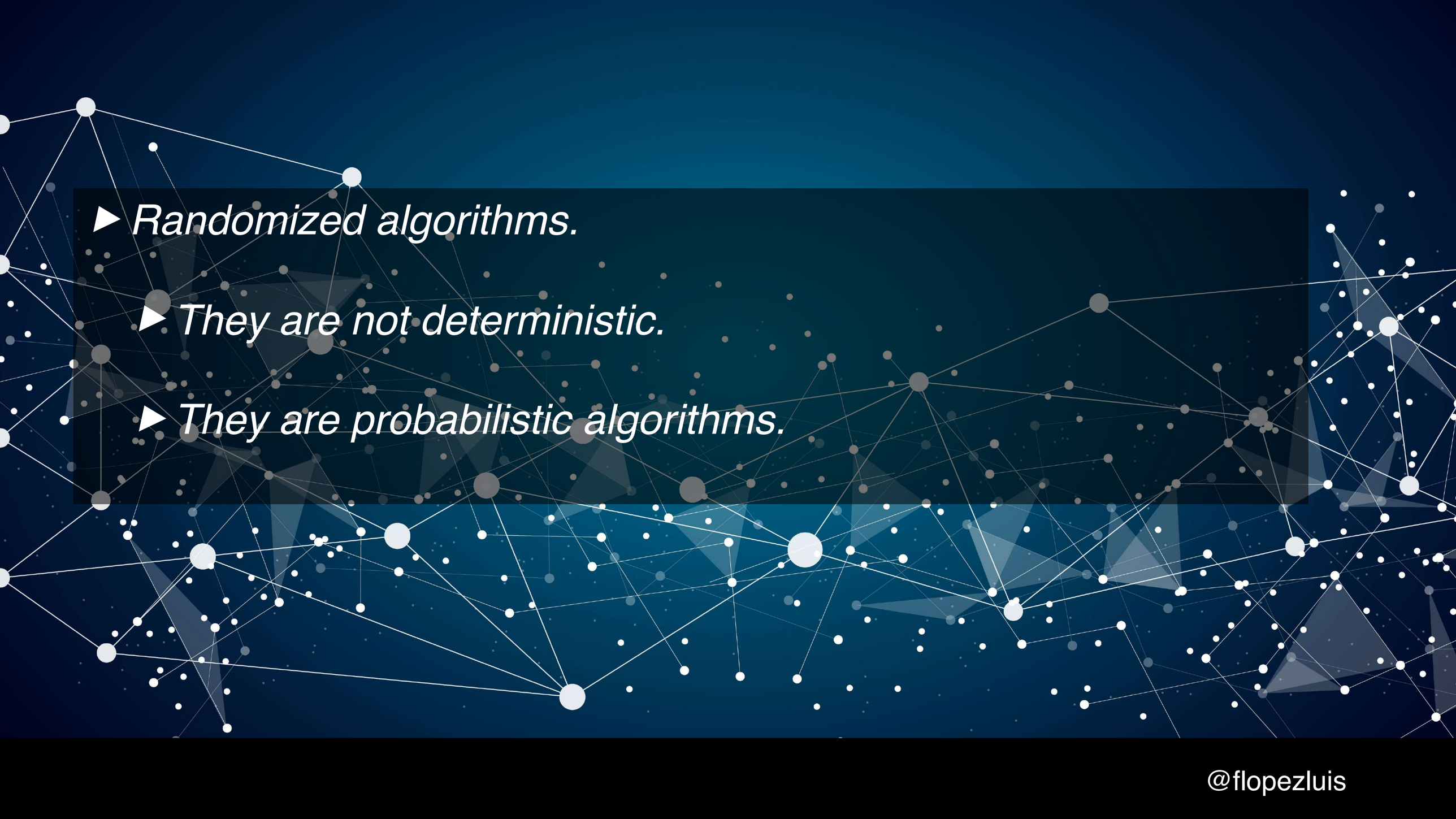
Epidemic theory shows that starting with a single infected the time to infect the entire population is proportional to the log of the population size.



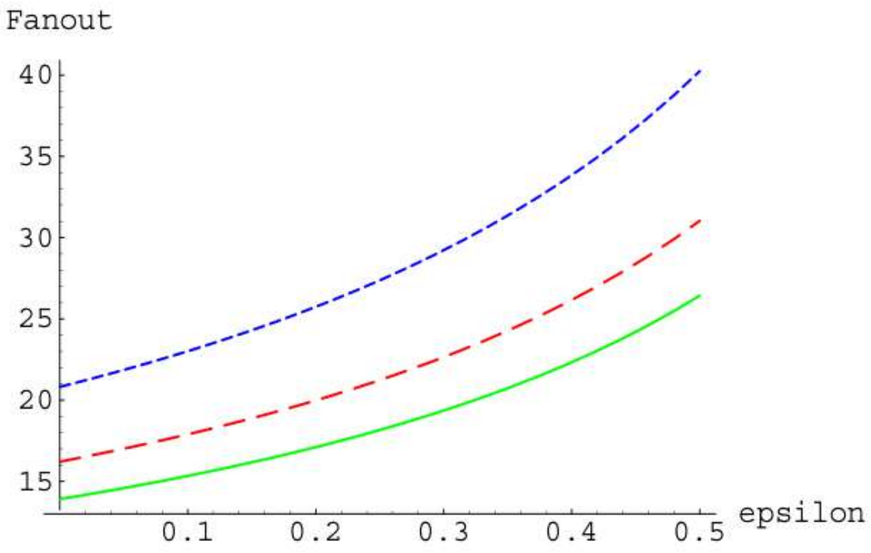
How do they work?

In general they have these properties:

- *Node selection must be random, or at least guarantee enough peer diversity*
- *Only local information is available at all nodes*
- *Communication is round-based (periodic)*
- *Transmission and processing capacity per round is limited*
- *All nodes run the same protocol*


- *Randomized algorithms.*
  - *They are not deterministic.*
  - *They are probabilistic algorithms.*

Probabilistic

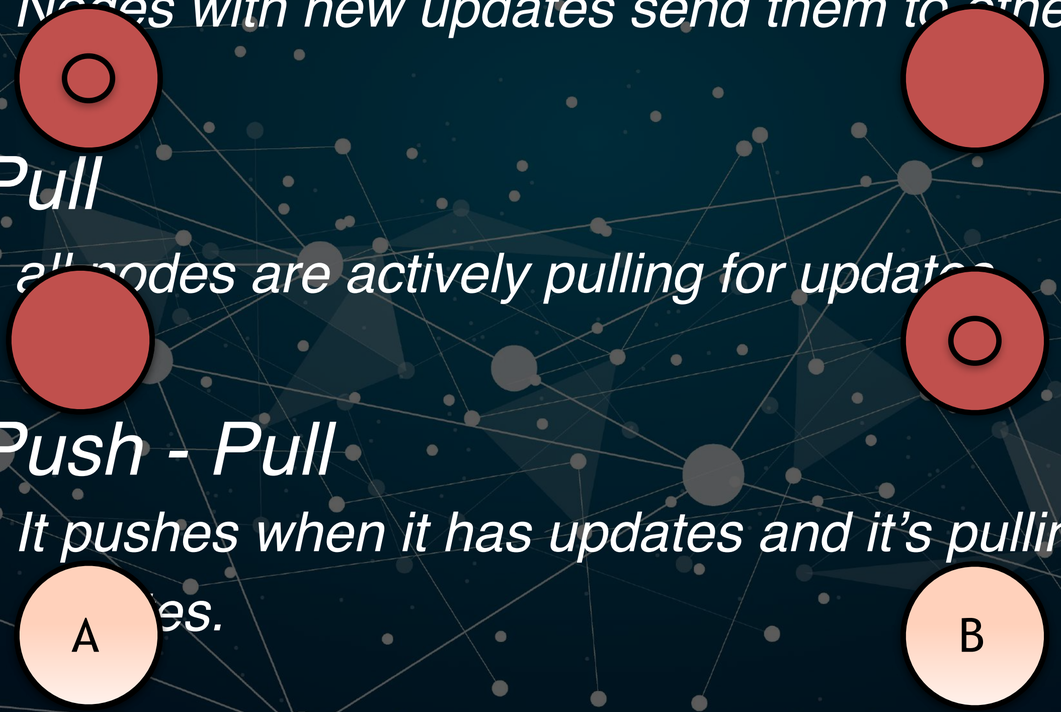


Fanout required versus probability of node failure, for n=1000,10000,100000.

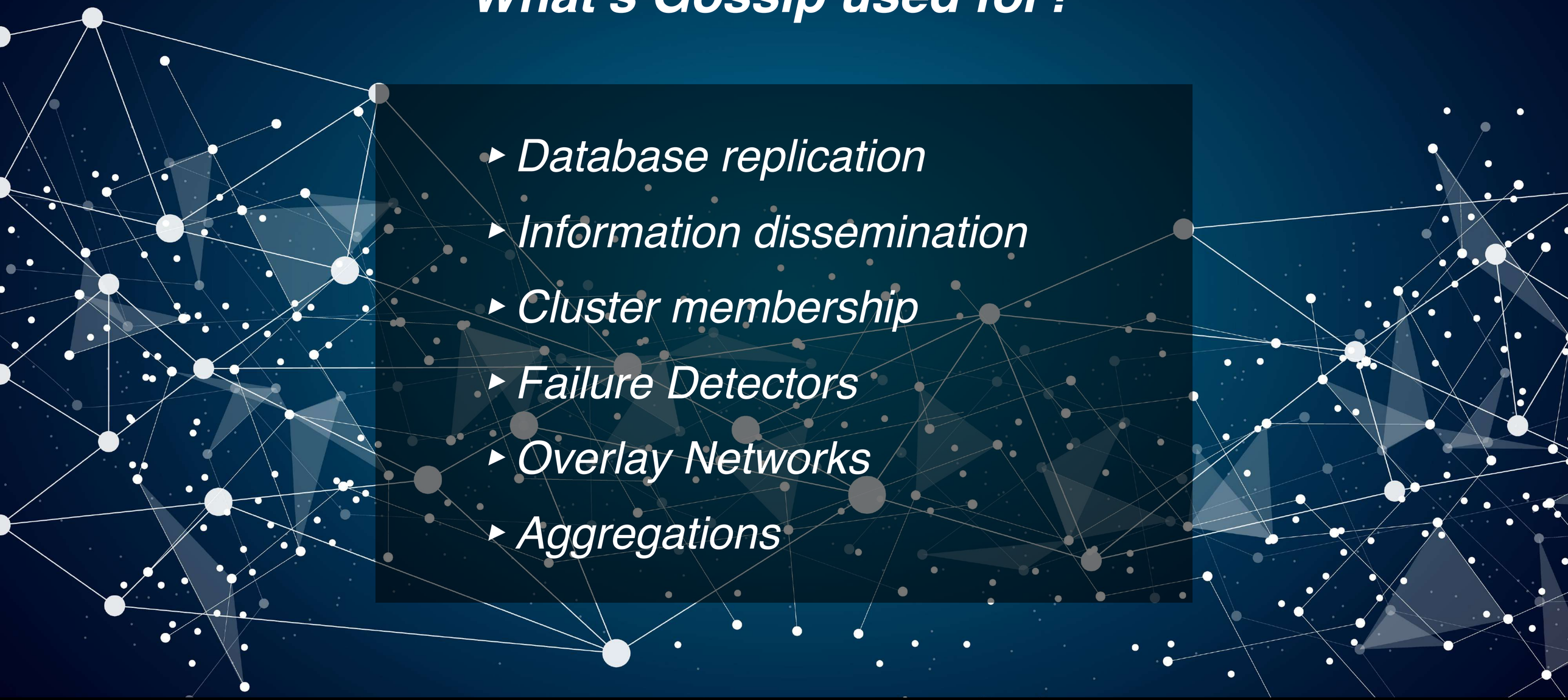
**From "Probabilistic reliable dissemination in large-scale system"*

Messaging strategies

- *Push*
  - *Nodes with new updates send them to other nodes.*
- *Pull*
  - *all nodes are actively pulling for updates*
- *Push - Pull*
  - *It pushes when it has updates and it's pulling for new updates.*



What's Gossip used for?



- *Database replication*
- *Information dissemination*
- *Cluster membership*
- *Failure Detectors*
- *Overlay Networks*
- *Aggregations*

Who using Gossip?

- *RIAK*
- *CASSANDRA*
- *DYNAMO*
- *CONSUL*
- *Amazon s3*
- *Docker Swarm*
- *ElasticSearch*

- *Hazelcast*
- *Redis Cluster*
- *AKKA*
- *Flume (cloudera)*
- *Blockchain*
- *Dynomite*
- *Tribler*

Strengths of Gossip

- *Scalable*
- *Fault-tolerance.*
- *Robust*
- *Convergent consistency.*
- *Extremely decentralized form of information discovery.*
- *Little code and complexity*

Strengths of Gossip

- *Little code and complexity*

Active thread (peer P):

```
(1) selectPeer(&Q);
(2) selectToSend(&bufs);
(3) sendTo(Q, bufs);                  ----->
(4)
(5) receiveFrom(Q, &bufr);            <-----
(6) selectToKeep(cache, bufr);
(7) processData(cache);
```

Passive thread (peer Q):

```
(1)
(2)
(3) receiveFromAny(&P, &bufr);
(4) selectToSend(&bufs);
(5) sendTo(P, bufs);
(6) selectToKeep(cache, bufr);
(7) processData(cache)
```

**From "Gossiping in Distributed Systems"*

Weaknesses of Gossip protocols

- *Not very efficient. Messages can arrive several times to a node.*
- *Gossip protocols are slow.*
- *Latency.*
- *Hard to reproduce and debug. Unexpected problems that arise at runtime*
- *Gossip protocols can't scale well in some situations.*
- *Most of them rely on non-scalable membership protocol.*

Peer Sampling Service

Active thread (peer P):

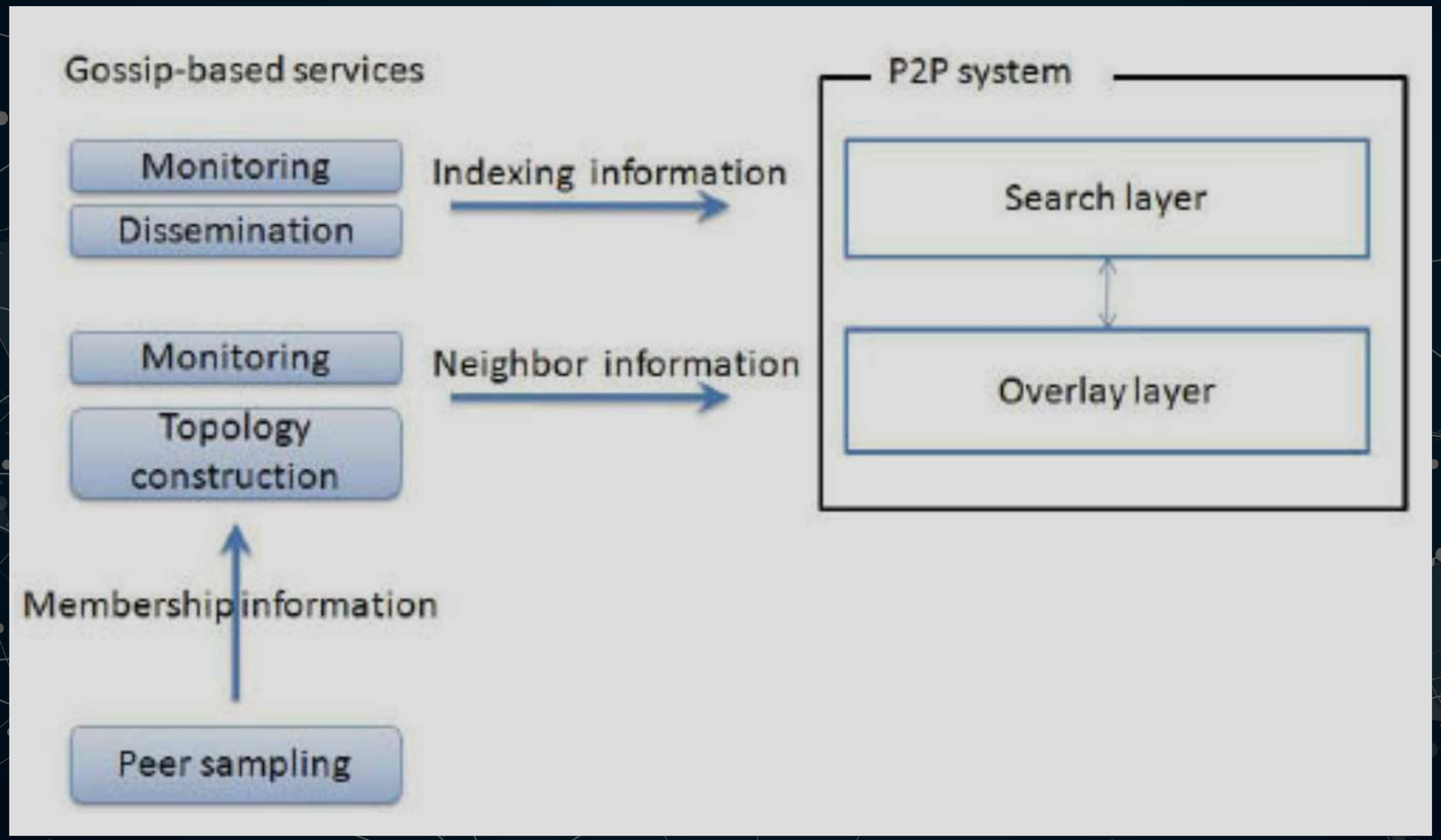
```
(1) selectPeer(&Q);
(2) selectToSend(&bufs);
(3) sendTo(Q, bufs);                    ----->
(4)
(5) receiveFrom(Q, &bufr);              <-----
(6) selectToKeep(cache, bufr);
(7) processData(cache);
```

Passive thread (peer Q):

```
(1)
(2)
(3) receiveFromAny(&P, &bufr);
(4) selectToSend(&bufs);
(5) sendTo(P, bufs);
(6) selectToKeep(cache, bufr);
(7) processData(cache)
```

**From "Gossiping in Distributed Systems"*

Peer Sampling Service



*From “P2P Techniques for decentralized applications”

FAQ

- *Partial view vs full view*
- *When nodes stop sharing an update/info?*
- *Gossip vs Raft/Paxos…*
- *Is it Gossip affected by partitions?*
- *Does Gossip tolerate Byzantine failures?*

Summary

- *Simple to implement, robust and resilient.*
- *Designed to deal with continuous changes.*
- *Reliable despite peer failures and message loss.*
- *Nodes are autonomous.*
- *They are PROBABILISTIC.*

“Success is not final, failure is not fatal: it is the courage to continue that counts”
Winston Churchill

Félix López Luis, @flopezluis

References

- *A. Demers, D. Greene, C. Hauser, W. Irish, J. Larson, S. Shenker, H. Sturgis, D. Swinehart, and D. Terry. “Epidemic Algorithms for R Database Maintenance.” In Proc. Sixth Symp. on Principles of Distributed Computing, pp. 1–12, Aug. 1987. ACM.*
- *Kermack, W. O.; McKendrick, A. G. (1927). "A Contribution to the Mathematical Theory of Epidemics". Proceedings of the Royal Soci Mathematical, Physical and Engineering Sciences 115 (772)*
- *Ken Birman. The Promise, and Limitations, of Gossip Protocols. SIGOPS Oper. Syst. Rev., 41(5):8–13, October 2007*
- *Gossip-based Protocols for Large-scale Distributed Systems. Márk Jelasity, 2013*
- *Anne-Marie Kermarrec and Maarten van Steen. 2007. Gossiping in distributed systems. SIGOPS Oper. Syst. Rev. 41, 5 (October 20 DOI: https://doi.org/10.1145/1317379.1317381*
- *Abhinandan Das, Indranil Gupta, and Ashish Motivala. 2002. SWIM: Scalable Weakly-consistent Infection-style Process Group Mem Protocol. In Proceedings of the 2002 International Conference on Dependable Systems and Networks (DSN ’02). IEEE Computer Sc Washington, DC, USA, 303-312.*
- *J. Leitão, J. Pereira, and L. Rodrigues. Epidemic broadcast trees. In Huai, J. and Baldoni, R. and Yen, I., editor, IEEE International S On Reliable Distributed Systems, pages 301–310. IEEE Computer Society, 2007*

References

- *P. Th. Eugster, R. Guerraoui, S. B. Handurukande, P. Kouznetsov, and A.-M. Kermarrec. 2003. Lightweight probabilistic broadcast. A Comput. Syst. 21, 4 (November 2003), 341-374. DOI=http://dx.doi.org/10.1145/945506.945507*
- *JELASITY, M., GUERRAOUI, R., KERMARREC, A.-M., AND VAN STEEN, M. 2004. The peer sampling service: Experimental evalua unstructured gossip-based implementations. In Middleware 2004, H.-A. Jacobsen, Ed. Lecture Notes in Computer Science, vol. 323 Springer-Verlag, 79–98.*
- *Lorenzo Alvisi, Jeroen Doumen, Rachid Guerraoui, Boris Koldehofe, Harry Li, Robbert van Renesse, and Gilles Tredan. 2007. How r gossip-based communication protocols?. SIGOPS Oper. Syst. Rev. 41, 5 (October 2007), 14-18. DOI: https://doi.org/10.1145/1317379.1317383*
- *Ali Saidi and Mojdeh Mohtashemi. Minimum-cost first-push-then-pull gossip algorithm. IEEE Wireless Communications and Networki Conference, WCNC, pages 2554–2559, 2012*
- *A Gossip-Style Failure Detection Service: Robbert van Renesse, Yaron Minsky, and Mark Hayden*; Dept. of Computer Science, Corn University; 4118 Upson Hall, Ithaca, NY 14853*
- *12] Gupta, Indranil, Chandra, Tushar D., and Goldszmidt, Germ´an S. On scalable and efficient distributed failure detectors. In Proce the Twentieth Annual ACM Symposium on Principles of Distributed Computing, PODC ’01, pp. 170–179,New York, NY, USA, 2001. A 1-58113-383-9. doi: 10.1145/383962.384010. URL http://doi.acm.org/10.1145/383962.384010*

References

- http://status.aws.amazon.com/s3-20080720.html
- http://docs.datastax.com/en/cassandra/3.0/cassandra/architecture/archGossipAbout.html
- https://www.consul.io/docs/internals/gossip.html
- Montresor, A.: Intelligent Gossip. In: Studies on Computational Inteligence, Intelligent Distributed Computing, Systems and Applicatio Springer, Heidelberg (2008)
- On disseminating information reliably without broadcasting. Proceedings of the International Conference on Distributed Computing S (1987), pp. 74–81
- [Brenda Baker and Robert Shostak. Gossips and telephones. Discrete Mathematics, 2(3):191–193, June 1972.
- http://www.inf.u-szeged.hu/~jelasity/ddm/gossip.pdf
- Kermarrec, Anne-Marie, and Steen, Maarten Van, “Gossiping in distributed systems”, ACM SIGOPS Operating Systems Review, Volu Issue 5, Pages: 2 – 7, 2007.
- S. Voulgaris, M. Jelasity, M. van Steen, A Robust and Scalable Peer-to-Peer Gossiping Protocol,Lecture Notes in Computer Science vol. 2872 (Springer, Berlin/Heidelberg, 2004), pp. 47–58. doi:10.1007/b104265