Gossip Protocols

Part II - Peer sampling and Topology Management

Peer To Peer System 2016-2017
Emanuele Carlini - emanuele.carlini@isti.cnr.it

Gossip Midterm assignment

- Read the paper: Jelasity, M., Montresor, A., Babaoglu, O.: Gossip-based aggregation in large dynamic net-works. ACM Transactions on Computer Systems 23(3), 219–252 (2005).
- Write a report that describes:
  - 1) the gossip strategy to perfrom aggregation with the main charateristics of the protocol. Provide a short description of the functions that is possible to implement with gossip aggregation. (Optional bonus question: how would you estimate a distribution of values?)
  - 2) the impact of node failures, message losses, and message delay on the correctness of aggregation. Describe one way to cope with failures.

Gossip Java Libraries

incubator-gossip (https://github.com/apache/incubator-gossip)

Seems rather simple, recently updated and improving.

GossipLib (http://gossiplib.gforge.inria.fr/)

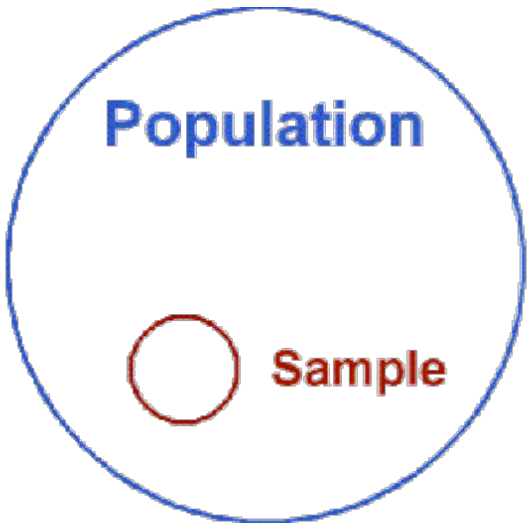
No documentation apart from Javadoc. Most of the popular protocols are already implemented.

java-gossip (https://github.com/jolira/java-gossip)

Last update 6 years ago. Some documentation on the old google code site (https://code.google.com/archive/p/java-gossip/).

Peer Sampling

select a sample of peers from the whole population



Peer sampling

A source node wants to choose another peer of the network that has certain characteristics

Node Selection

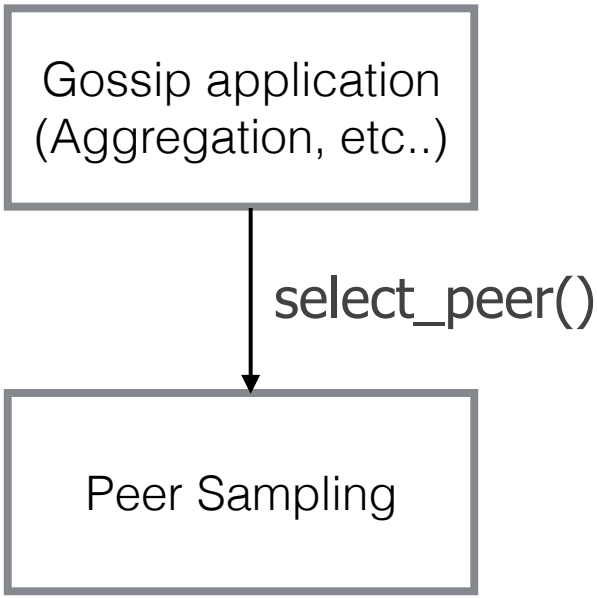
- Gossip algorithms (aggregation, dissemination, etc..) are based on the following assumption:
  - after a given interval, a node P may select a node Q chosen uniformly at **random** among the set of all the nodes participating to the protocol
- **Problem: How can we select a random peer in a fully distributed P2P system?**

Peer Sampling

- We need a **Peer Sampling Service**, i.e. a mechanism approximating a (random) choice on the whole set of nodes by exploiting only local information
  - we don't want a centralised server doing that
- The peer sampling service is built with gossip as well
- The basic idea: the nodes gossip with their neighbours and the topics of the gossip is.. the knowledge of the neighbours!

Two-layer architecture

- We have a two-layer architecture:
  - a gossip application layer (e.g. Aggregation)
  - an underlying peer sampling service
- The application layers exploit the peer sampling when it needs another node to communicate with



node's layered architecture

Ideal Peer Sampling

- In the ideal case nodes keep a full view of the network, and can sample from the entire node population
- This cannot be true in reality (it wouldn't scale!)
  - Too many nodes
  - Churn
- There is no point in having scalable gossip protocols if the peer sampling support does not

Actual Peer Sampling

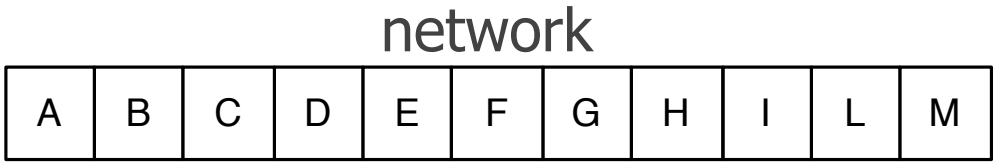
- Instead of the whole network, each node maintains a **partial view** of fixed size of the network
- When gossiping, nodes exchange such partial view and decide which peers to keep of the other's view
- The idea is to build a **dynamic unstructured overlay** through gossiping
- In other words, the peer sampling service dynamically changes the neighbours (hence the overlay) of a node

Peer Sampling

- Random Peer Sampling protocols (Newscast, Cyclon) provides an **always changing** subset of the view from where to pick

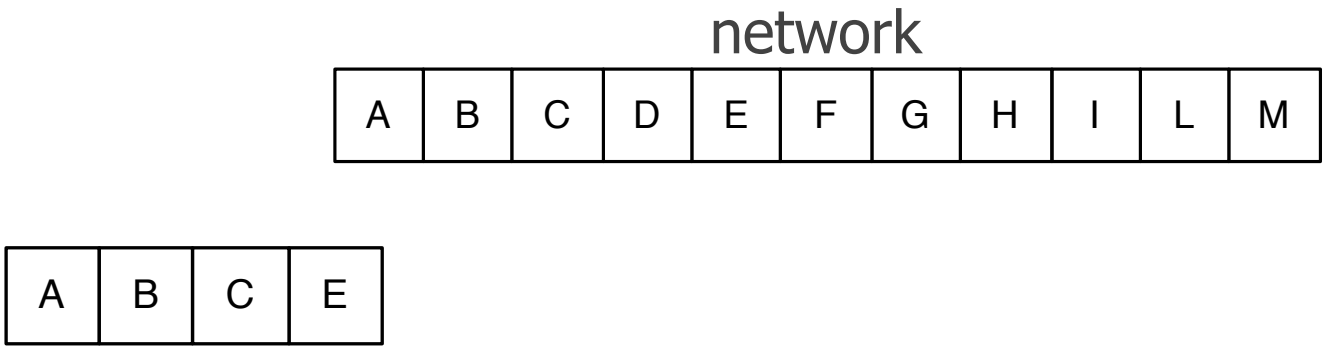
Peer Sampling

- Random Peer Sampling protocols (Newscast, Cyclon) provides an **always changing** subset of the view from where to pick



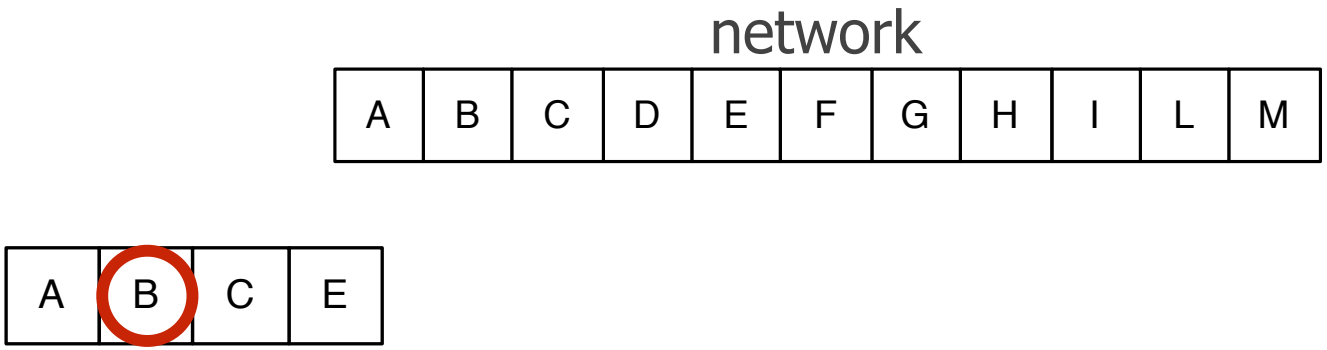
Peer Sampling

- Random Peer Sampling protocols (Newscast, Cyclon) provides an **always changing** subset of the view from where to pick



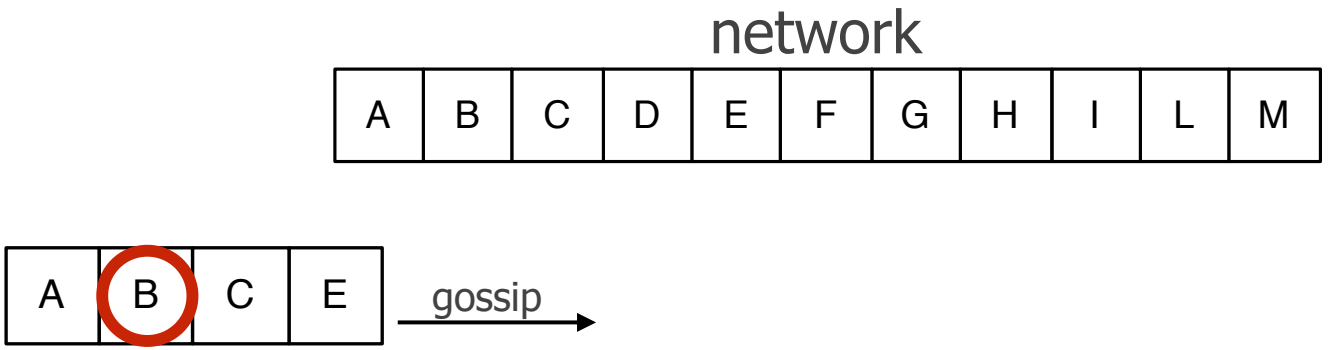
Peer Sampling

- Random Peer Sampling protocols (Newscast, Cyclon) provides an **always changing** subset of the view from where to pick



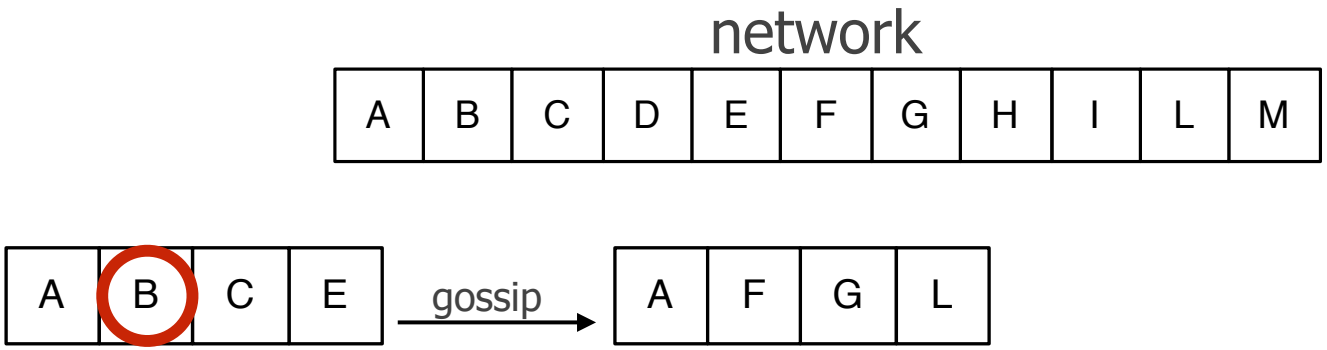
Peer Sampling

- Random Peer Sampling protocols (Newscast, Cyclon) provides an **always changing** subset of the view from where to pick



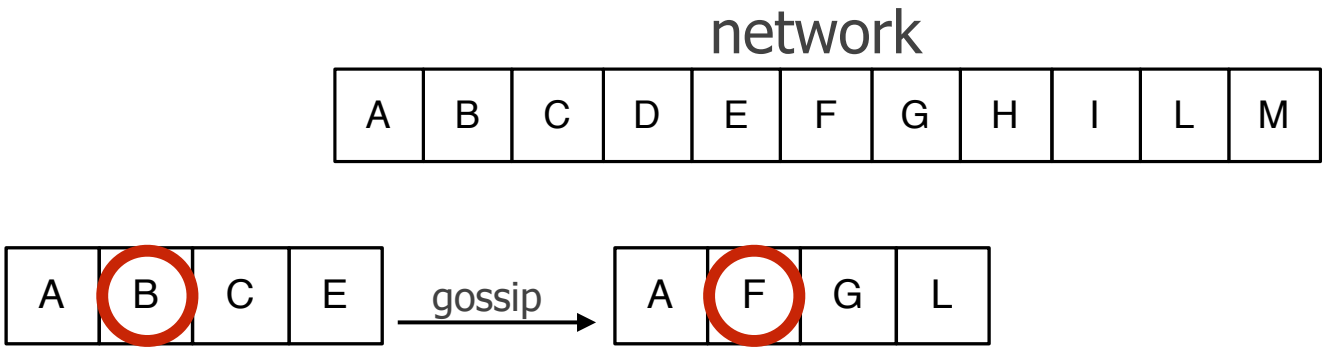
Peer Sampling

- Random Peer Sampling protocols (Newscast, Cyclon) provides an **always changing** subset of the view from where to pick



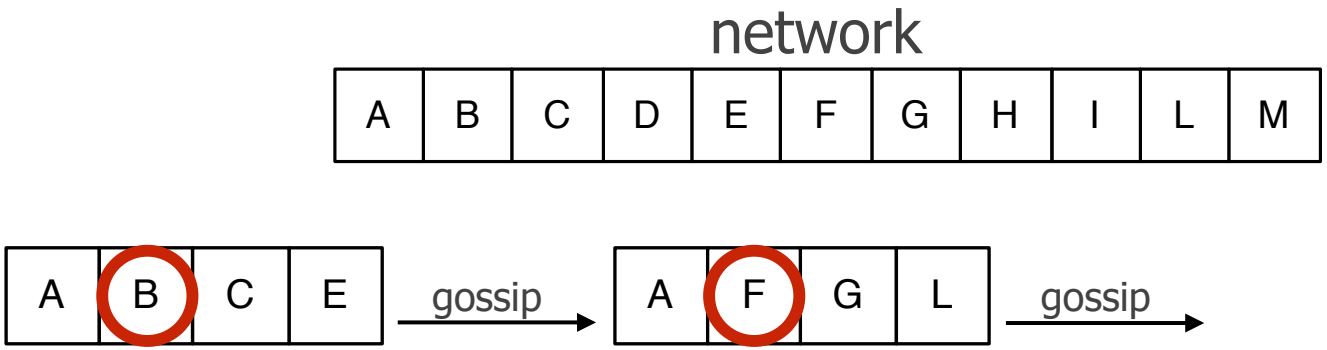
Peer Sampling

- Random Peer Sampling protocols (Newscast, Cyclon) provides an **always changing** subset of the view from where to pick



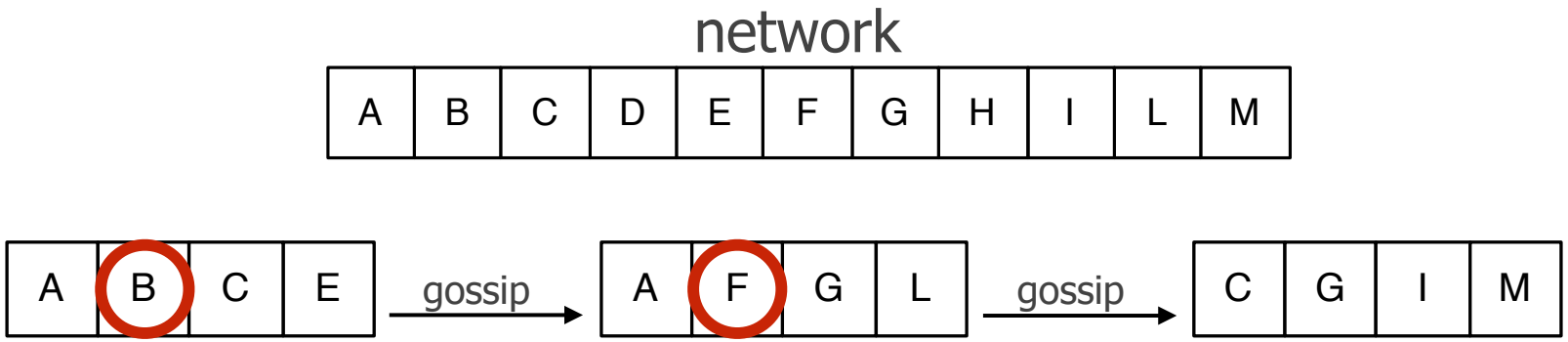
Peer Sampling

- Random Peer Sampling protocols (Newscast, Cyclon) provides an **always changing** subset of the view from where to pick



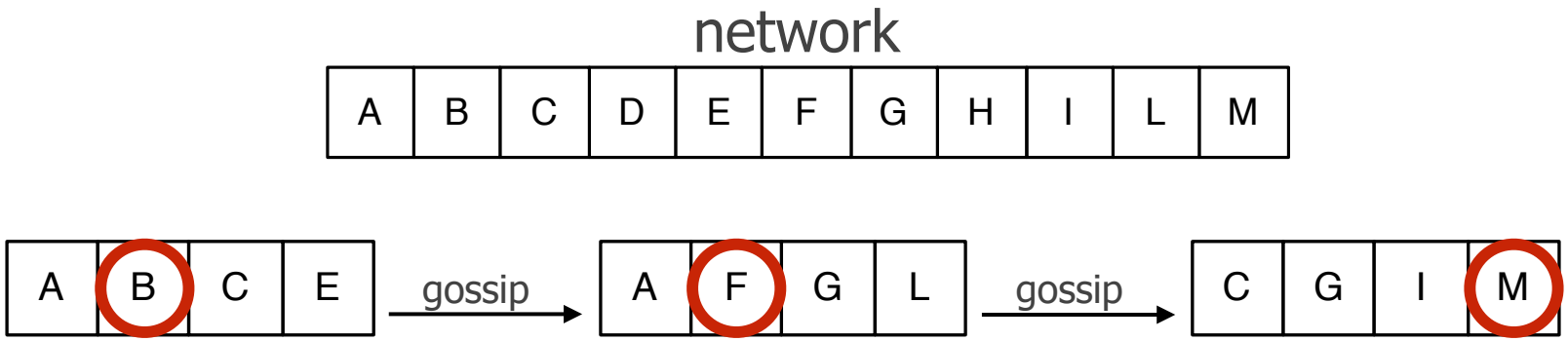
Peer Sampling

- Random Peer Sampling protocols (Newscast, Cyclon) provides an **always changing** subset of the view from where to pick



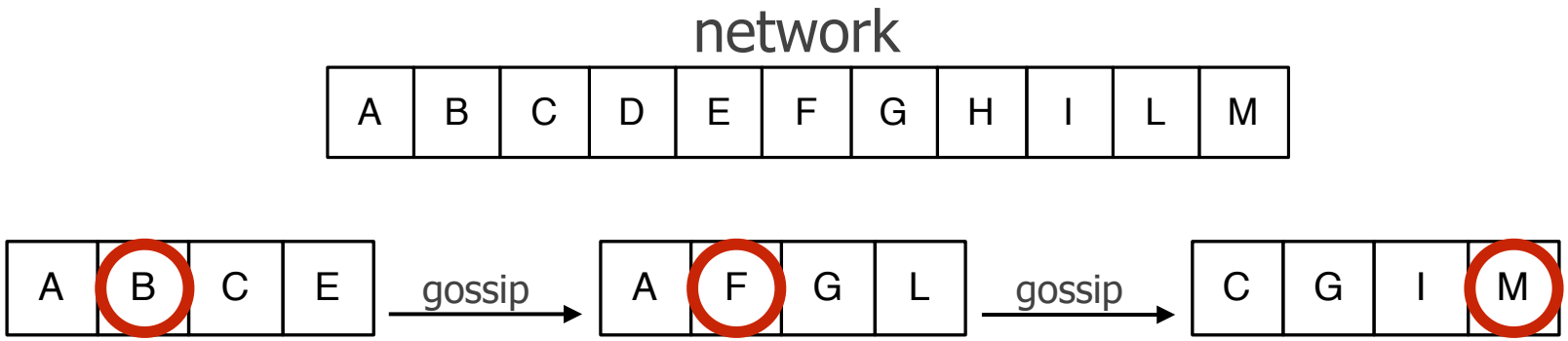
Peer Sampling

- Random Peer Sampling protocols (Newscast, Cyclon) provides an **always changing** subset of the view from where to pick



Peer Sampling

- Random Peer Sampling protocols (Newscast, Cyclon) provides an **always changing** subset of the view from where to pick



- Nodes keep a minimal (fixed!) subset of the node in the view
- The view adapts to churn

General Structure

- **select_peer**: selects a peer from the local view
- **select_to_send**: selects some entries from the local view
- **select_to_keep**: merges the received information to the local view, eliminates the duplicates and selects a subset of the resulting view which defines the new local view

active behaviour()

```
p = select_peer()
sent = select_to_send()
recv = send(p, sent)
view = select_to_keep(view, recv)
```

passive behaviour()

```
recv = receive(q)
sent = select_to_send()
view = select_to_keep(view, recv)
```

A peer sampling service is defined by specialising these operations

Partial view

- The peer sampling service keeps in the partial view information about the age and the contact point
- Age of the partial view is increased/decrease at every cycle

ID	Contact point (IP:port)	Age
1	192.168.0.44:4403	2
2	127.0.0.1:3463	5
3	8.9.0.42:5389	0

partial view of size 3

Random Peer Sampling

Choose a peer **at random** from the entire node population

- It's a core peer sampling service for almost any gossip applications
- Two popular protocols
  - Newscast — Tölgyesi, Norbert, and Márk Jelasity. "Adaptive peer sampling with newscast." European Conference on Parallel Processing. Springer Berlin Heidelberg, 2009.
  - Cyclon — Voulgaris, Spyros, Daniela Gavidia, and Maarten Van Steen. "Cyclon: Inexpensive membership management for unstructured p2p overlays." Journal of Network and Systems Management 13.2 (2005): 197-217.

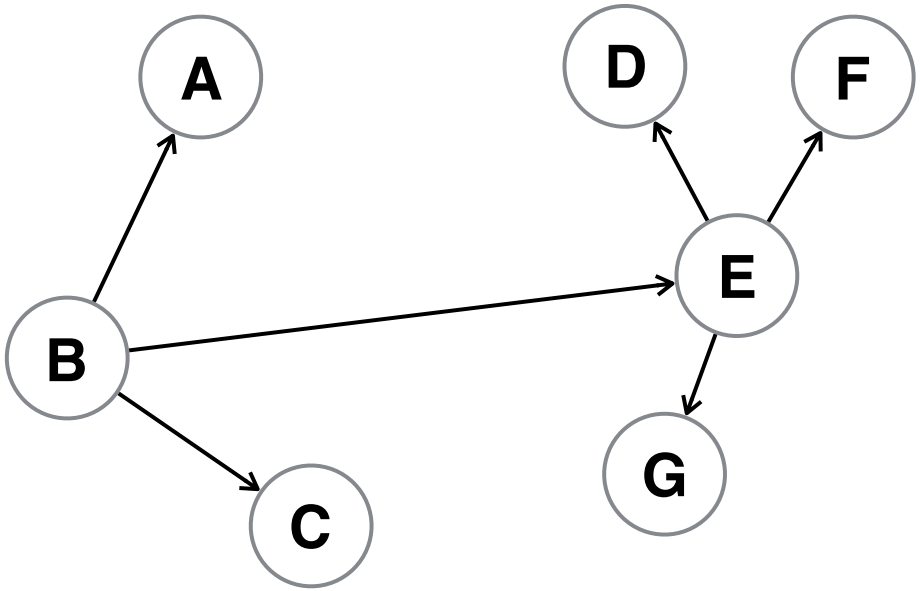
Newscast

- **SelectPeer**: selects a peer at random from the local view
- **SelectToSend**: selects all the descriptors in the local view + the descriptor of the actual peer
  - more recent descriptors have higher time-stamps
- **SelectToKeep**: keeps the most **c** recent descriptors

Newscast

B's view

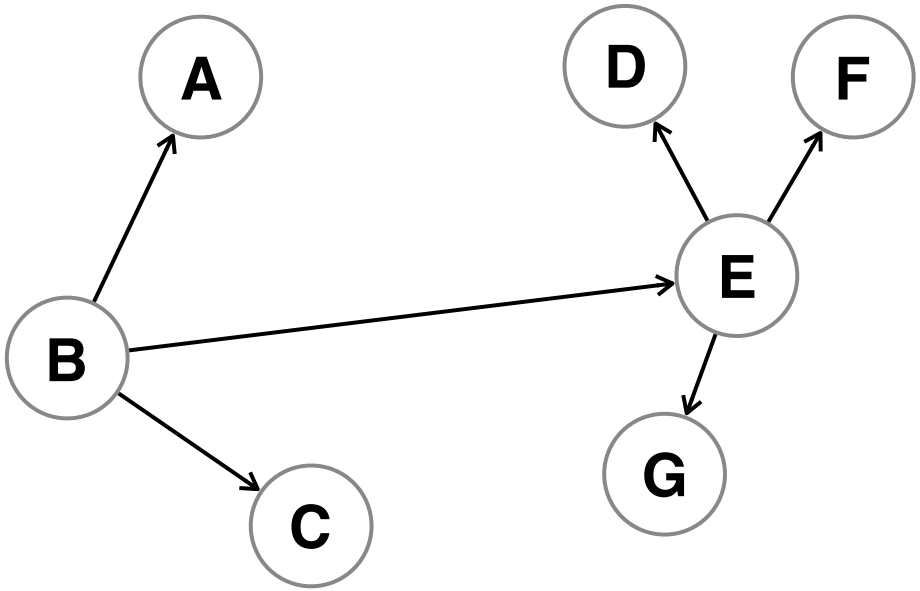
ID	Age
A	9
C	12
E	16



Newscast

B's view

ID	Age
A	9
C	12
E	16



E's view

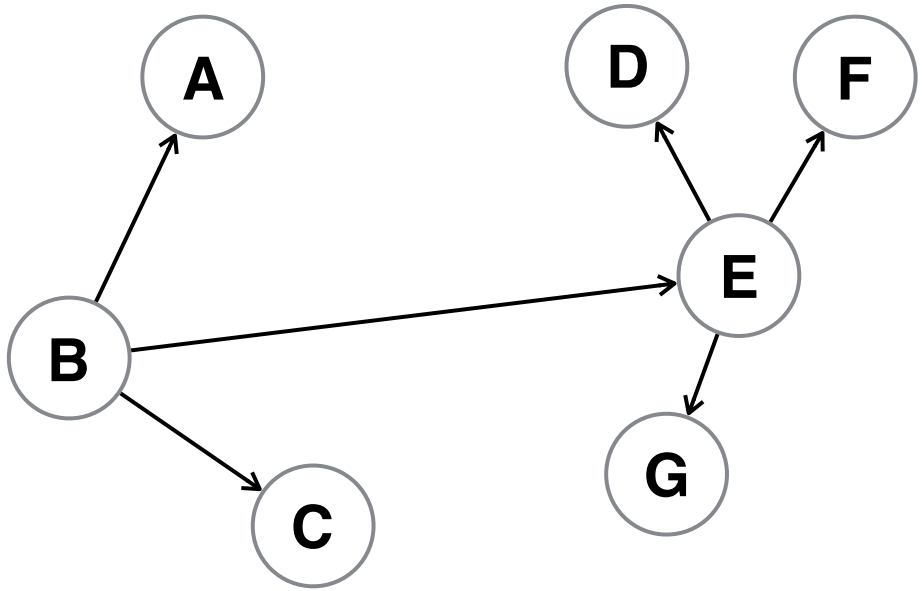
F	7
D	10
G	14

1. B picks a random node from the view

Newscast

B's view

ID	Age
A	9
C	12
E	16
F	7
D	10
G	14
E	20



E's view

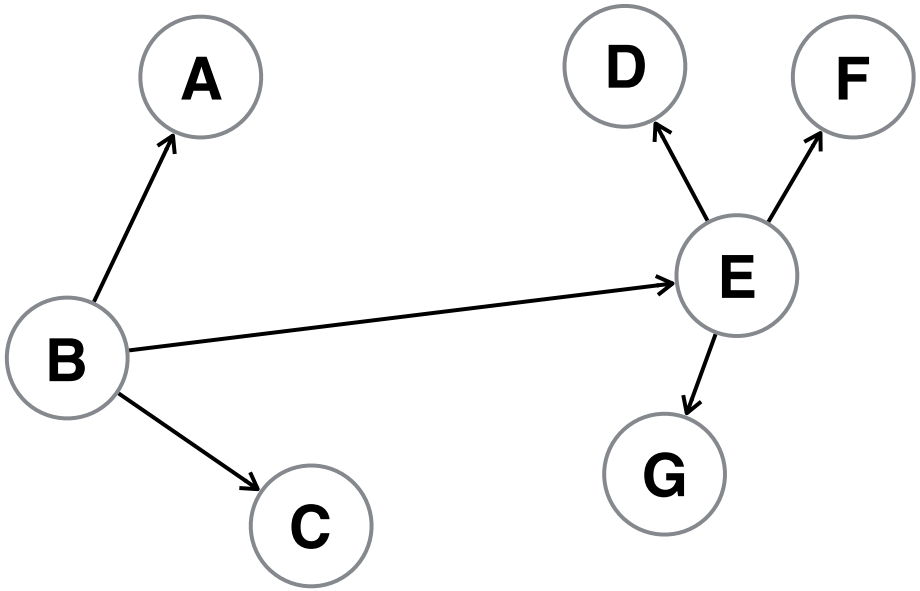
F	7
D	10
G	14
A	9
C	12
E	16
B	20

1. B picks a random node from the view
2. B and E exchanges views
   + their fresh descriptors

Newscast

B's view

ID	Age
[redacted]	
[redacted]	
E	16
[redacted]	
[redacted]	
G	14
E	20



E's view

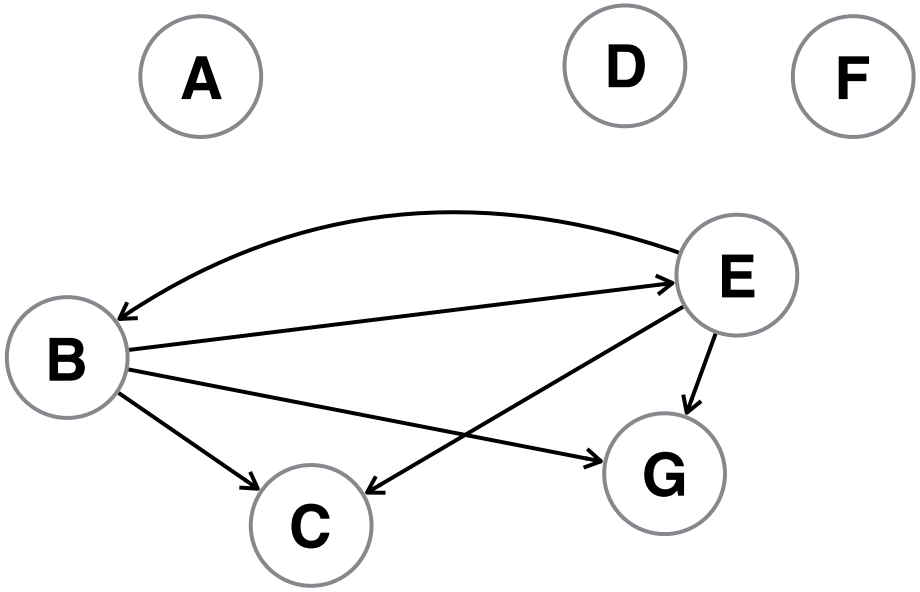
[redacted]	
[redacted]	
G	14
[redacted]	
C	12
[redacted]	
B	20

1. B picks a random node from the view
2. B and E exchanges views
   + their fresh descriptors
3. B and E keep the c=3 freshest links

Newscast

B's view

ID	Age
E	20
G	14
C	20



E's view

C	12
B	20
G	14

Resulting graph after the iteration

Newscast

- Robust to node and link failure and dynamism (churn) and scalable
- Newscast approximates **small-world** networks, with high clustering coefficient (CC) and small average path length
- High Clustering coefficient is bad for:
  - Flooding, due to the amount of redundant messages
  - Robustness, large clusters may be weakly connected to the rest of the network

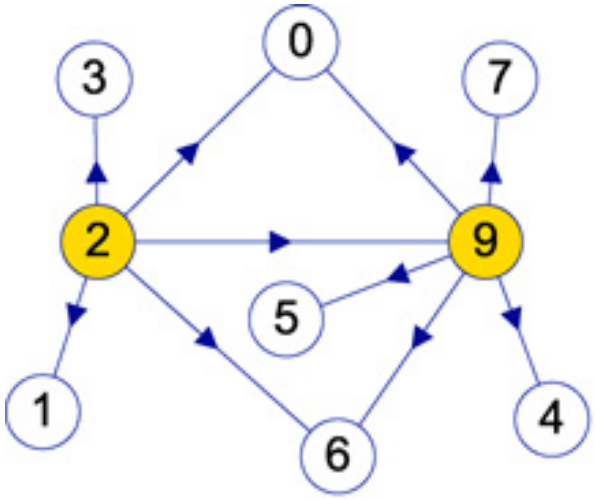
CC of a node = quantifies how close the neighbours are to being a clique

Cyclon

SelectPeer	Select a random peer p from view
SelectToSend	Select **L** entry of own view + peer descriptor
SelectToKeep	Replace peer's view with the one received from p (discard duplicates)

- Cyclon better approximates a random graph
  - Small clustering coefficient
  - Small average path length
- Connectivity is always guaranteed

Cyclon Example

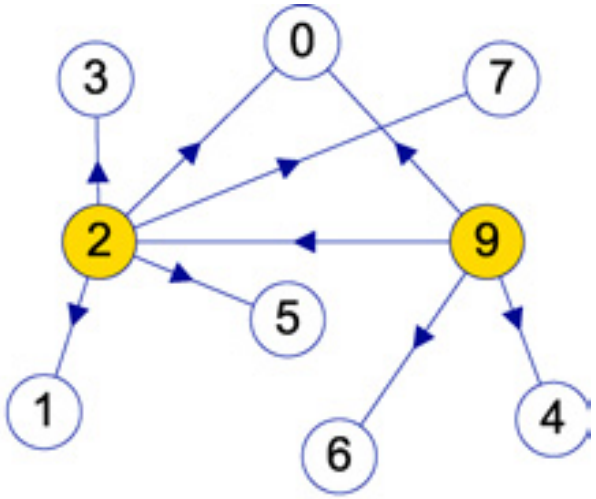


(a) Before shuffling

L=3

$2 \rightarrow 9 : \{2, 0, 6\}$

$2 \leftarrow 9 : \{0, 5, 7\}$



(b) After shuffling

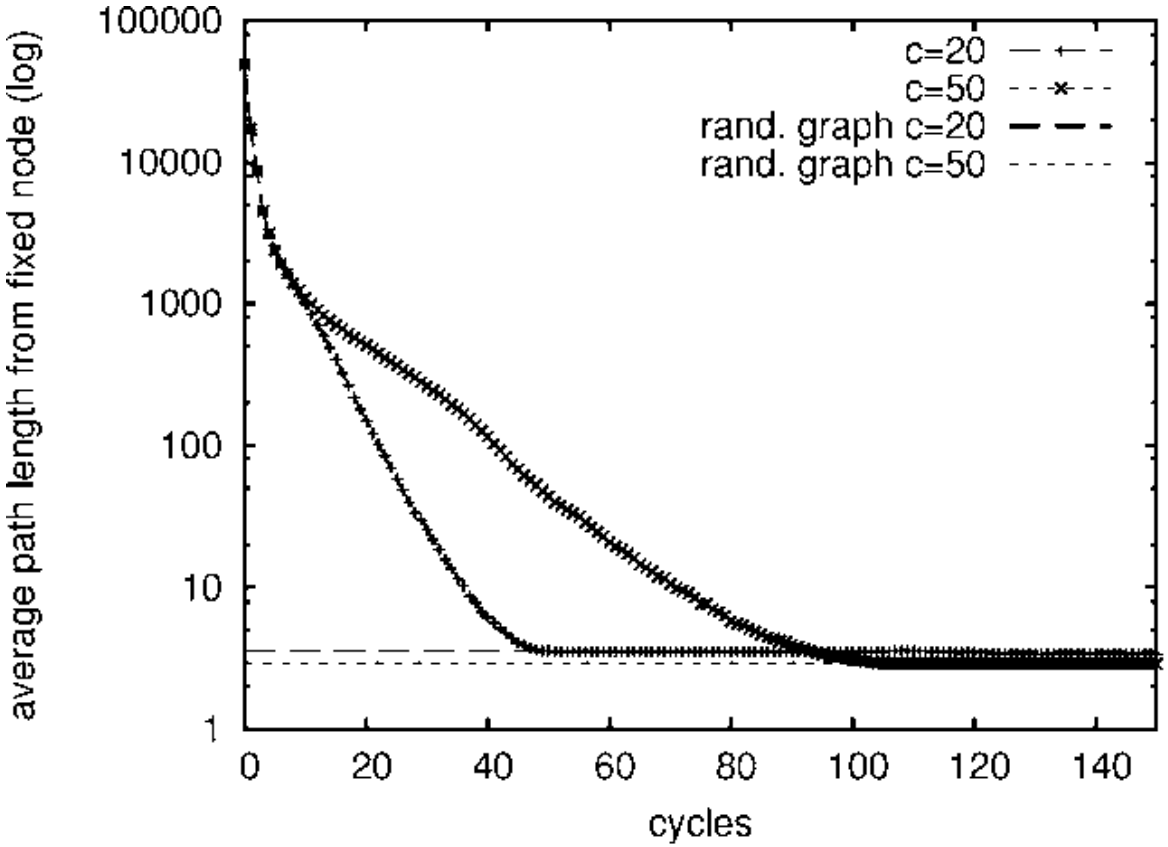
2:{0,1,3,6,9}

9:{0,4,6,5,7,}

2:{0,1,3,5,7}

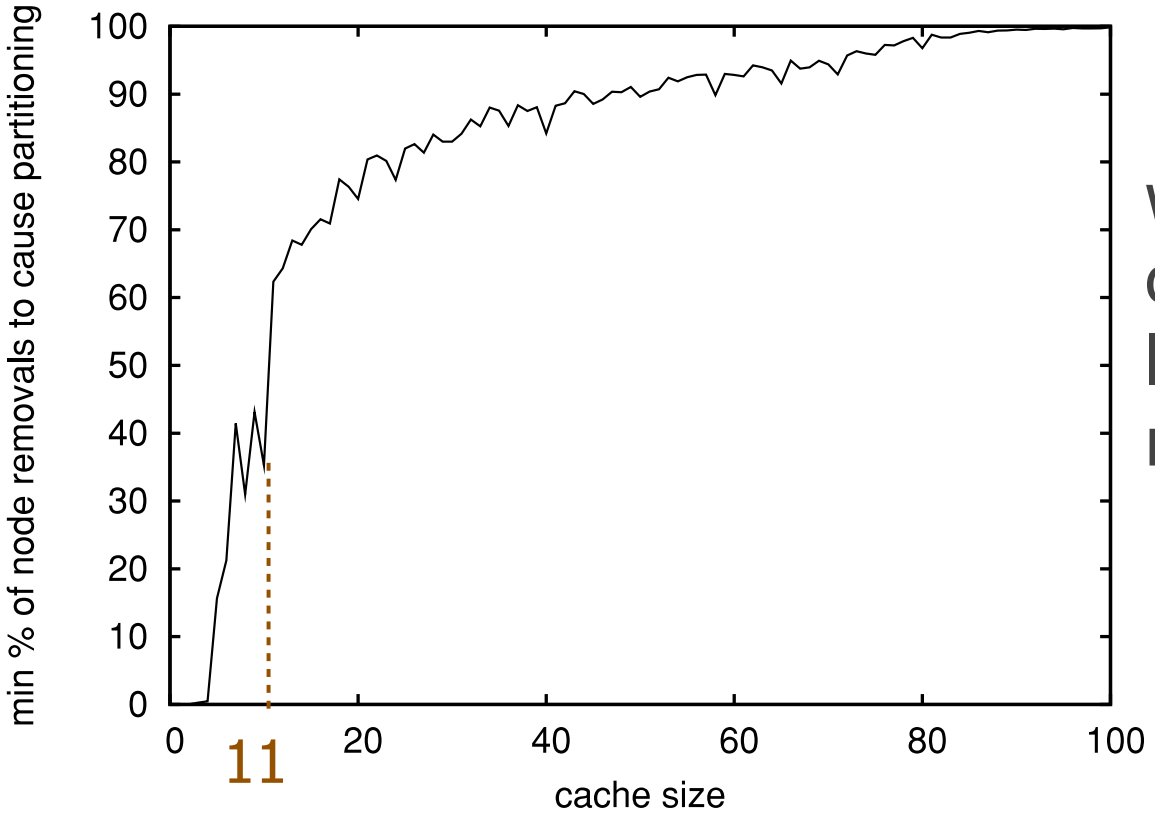
9:{0,2,4,6,5}

Cyclon Convergence



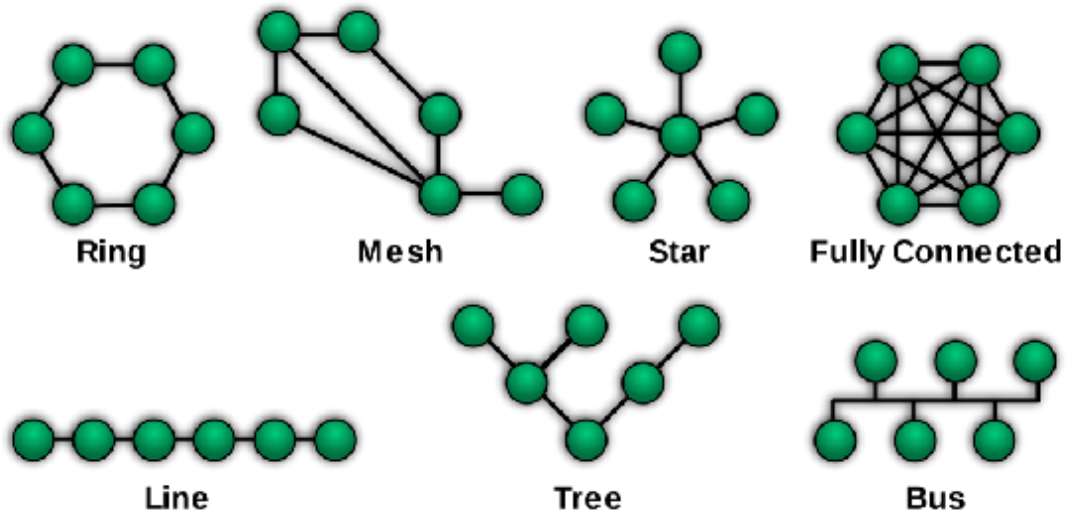
Starting from a 100k nodes chain graph, Cyclon converges to the average path length of a random graph in few cycles

Cyclon Robustness



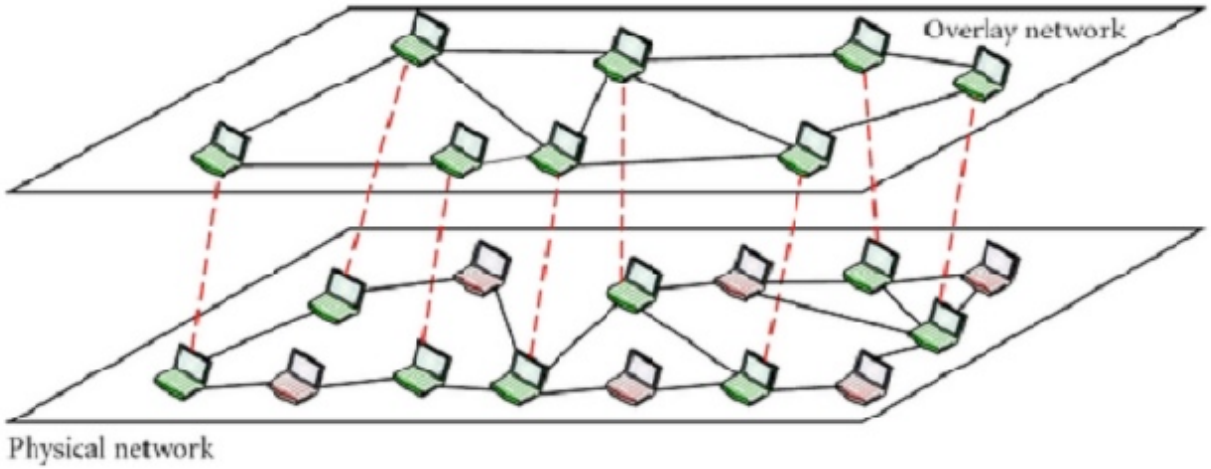
with a 100K nodes, cache size should be larger than 11 for good robustness

Topology Management



Overlay Network

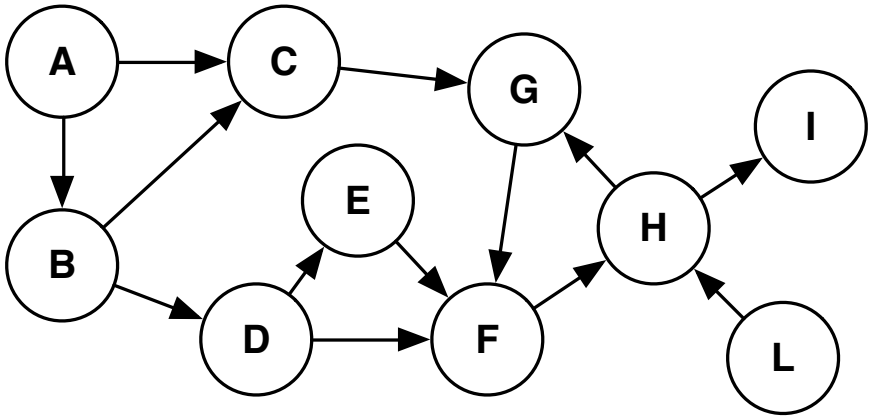
- **Overlay network**: a network built on top of another underlying network
  - Nodes in the overlay network are connected by logical links
  - Each overlay connection can corresponds to a path in the underlying network.
- Peer-to-peer networks are overlay networks built on top of the Internet
  - DHT



What is a Topology

- The set of all nodes and their neighbourhood represent a network with a given **topology**
- The best way to visualise a topology is to think to them as a graph
  - if node A has node B in its view, it exits an edge from A to B in the graph of the network
- The topology defines the properties of the network
  - random vs small scale
  - how information percolates in the network

Topology Example



A's view

ID
B
C

B's view

ID
C
D

H's view

ID
F
I

D's view

ID
E
F

With gossip protocols we can manipulate the view of the node, therefore the network topology

Topology management

A node dynamically organises its local connection, thereby affecting the topology of the whole overlay network

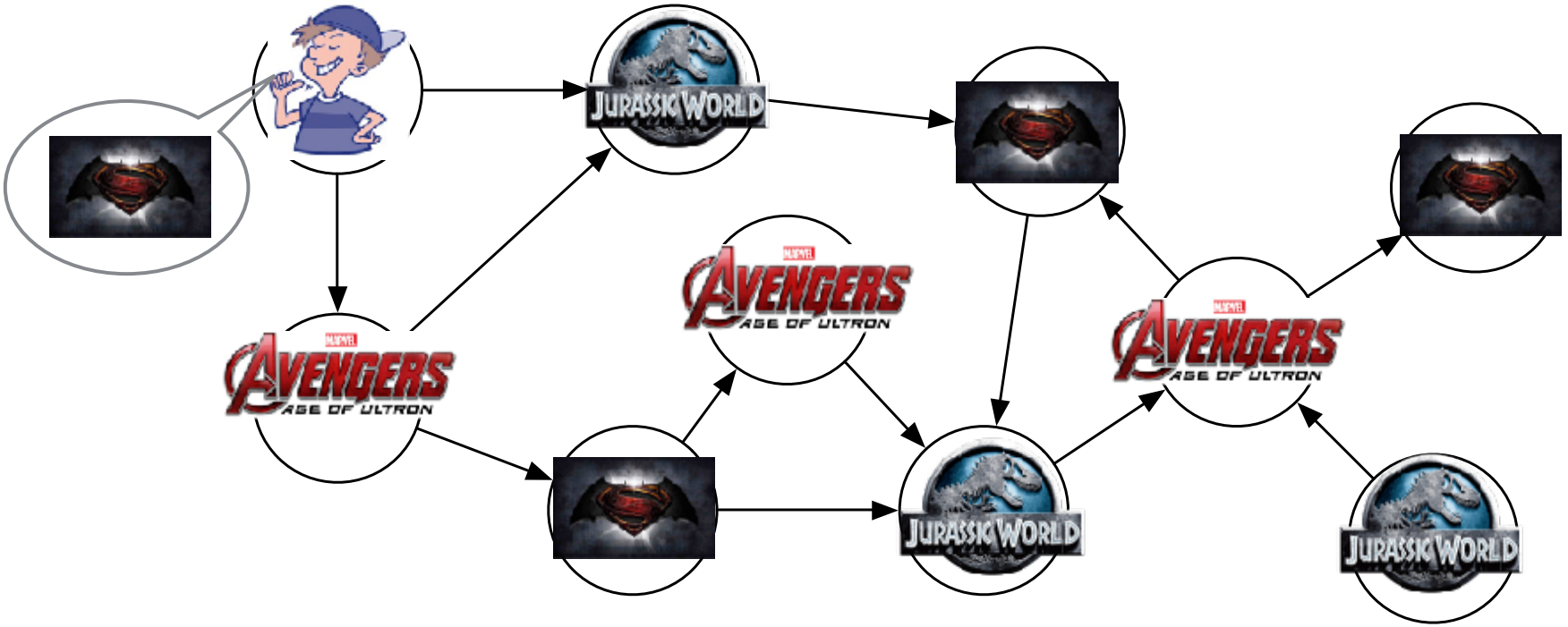
Fixed and variable topologies

- Dissemination and Aggregation considered fixed overlay topology, in which the connections of the network never changes
- In random peer sampling (especially with Cyclon) we aimed to a random graph topology

It is possible to modify the topology on demand, according to one's needs?

Pirate Illegal Movie Download Gossip Protocol

- Let's suppose I'm downloading Batman vs Superman and I want to connect to the nodes that are downloading the same movie



- Node reorganise the topology such that their neighbours are downloading the same movie

Pirate Illegal Movie Download Gossip Protocol

let's add a field to the node entry in the partial view

1	192.168.0.44:4403	2	

SelectPeer

SelectToSend

SelectToKeep

Pirate Illegal Movie Download Gossip Protocol

let's add a field to the node entry in the partial view

1	192.168.0.44:4403	2	

SelectPeer

select a peer (Q) with my very same movie from the partial view. If none available, select random

SelectToSend

SelectToKeep

Pirate Illegal Movie Download Gossip Protocol

let's add a field to the node entry in the partial view

1	192.168.0.44:4403	2	

SelectPeer

select a peer (Q) with my very same movie from the partial view. If none available, select random

SelectToSend

send at least size/2 entries from my partial view to Q. Preference to the entries having Q's movies

SelectToKeep

Pirate Illegal Movie Download Gossip Protocol

let's add a field to the node entry in the partial view

1	192.168.0.44:4403	2	

SelectPeer

select a peer (Q) with my very same movie from the partial view. If none available, select random

SelectToSend

send at least size/2 entries from my partial view to Q. Preference to the entries having Q's movies

SelectToKeep

keep the freshest entries that match my movie. If not enough to fill up my view, keep the freshest

Pirate Illegal Movie Download Gossip Protocol

- Now I'm happy and I can continue to download the movie..

Pirate Illegal Movie Download Gossip Protocol

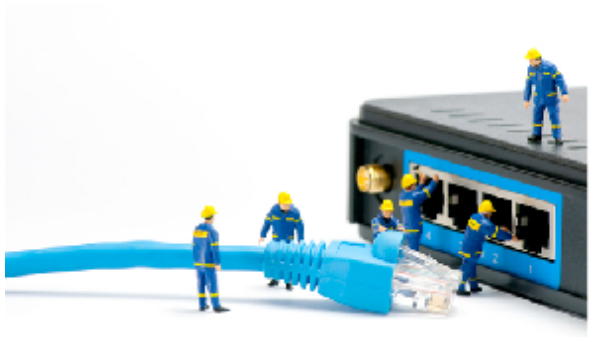
- Now I'm happy and I can continue to download the movie..
- .. but what if later I want to download Age of Ultron?

Pirate Illegal Movie Download Gossip Protocol

- Now I'm happy and I can continue to download the movie..
- .. but what if later I want to download Age of Ultron?

The network is disconnected and I cannot reach the other peers downloading AoU!

A Connectivity Problem

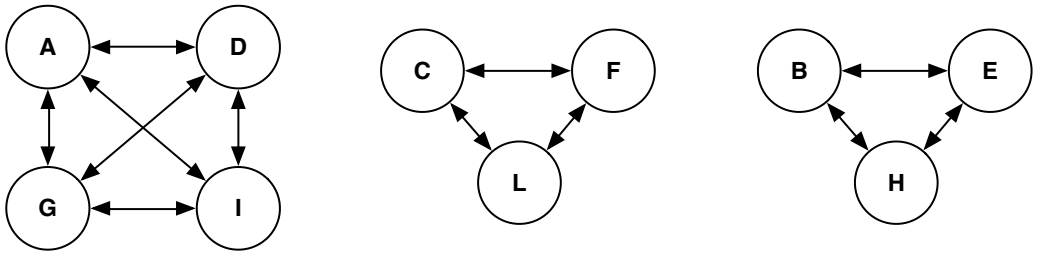


- Topology that connects nodes according to a criteria are useful, but also is maintaining connectivity
- The solution is that each node runs two ore more protocols
  - some to connect with nodes according to an “application” criteria (in our example, download the right movie)
  - the other to remain connected to the network
- For its characteristics **Cyclon** is often the best protocol to keep connectivity
  - realises a topology that maintains a random graph

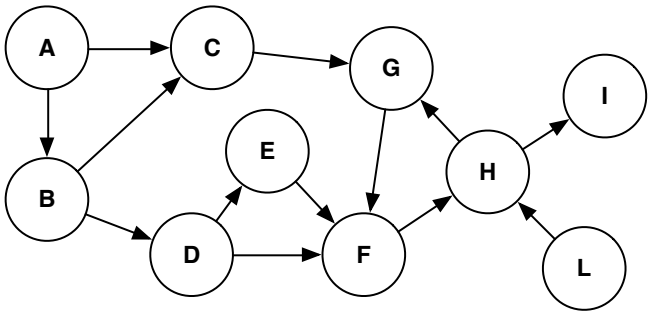
Layered Gossip

- Multiple protocols run on the same node
- Each protocol keeps its own view

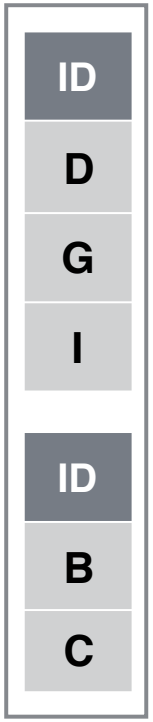
Layer 1



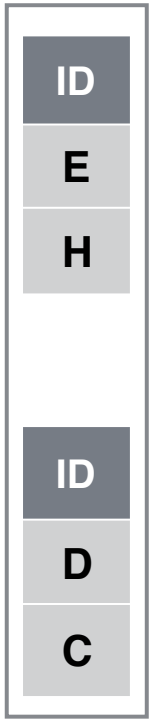
Layer 0



A's view

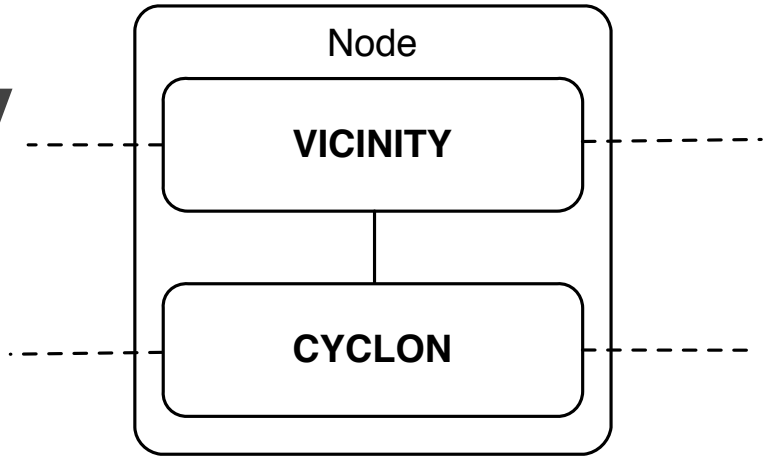


B's view



Vicinity

- Voulgaris, Spyros, and Maarten Van Steen. "Epidemic-style management of semantic overlays for content-based searching." Euro-Par 2005 Parallel Processing. Springer Berlin Heidelberg, 2005. 1143-1152.
- Layered approach
- Define a topology of **semantically close** neighbours
- Exploits Cyclon to maintain connectivity and for random peer sampling



Vicinity

let's define close according to proximity

1	192.168.0.44:4403	2	14,54

SelectPeer

SelectToSend

SelectToKeep

Vicinity

let's define close according to proximity

1	192.168.0.44:4403	2	14,54

SelectPeer

Select a random peer (P) from the underlying Cyclon protocol

SelectToSend

SelectToKeep

Vicinity

let's define close according to proximity

1	192.168.0.44:4403	2	14,54

SelectPeer

Select a random peer (P) from the underlying Cyclon protocol

SelectToSend

Order your own partial view and the cyclon view according to P position. Select the closest peers

SelectToKeep

Vicinity

let's define close according to proximity

1	192.168.0.44:4403	2	14,54

SelectPeer

Select a random peer (P) from the underlying Cyclon protocol

SelectToSend

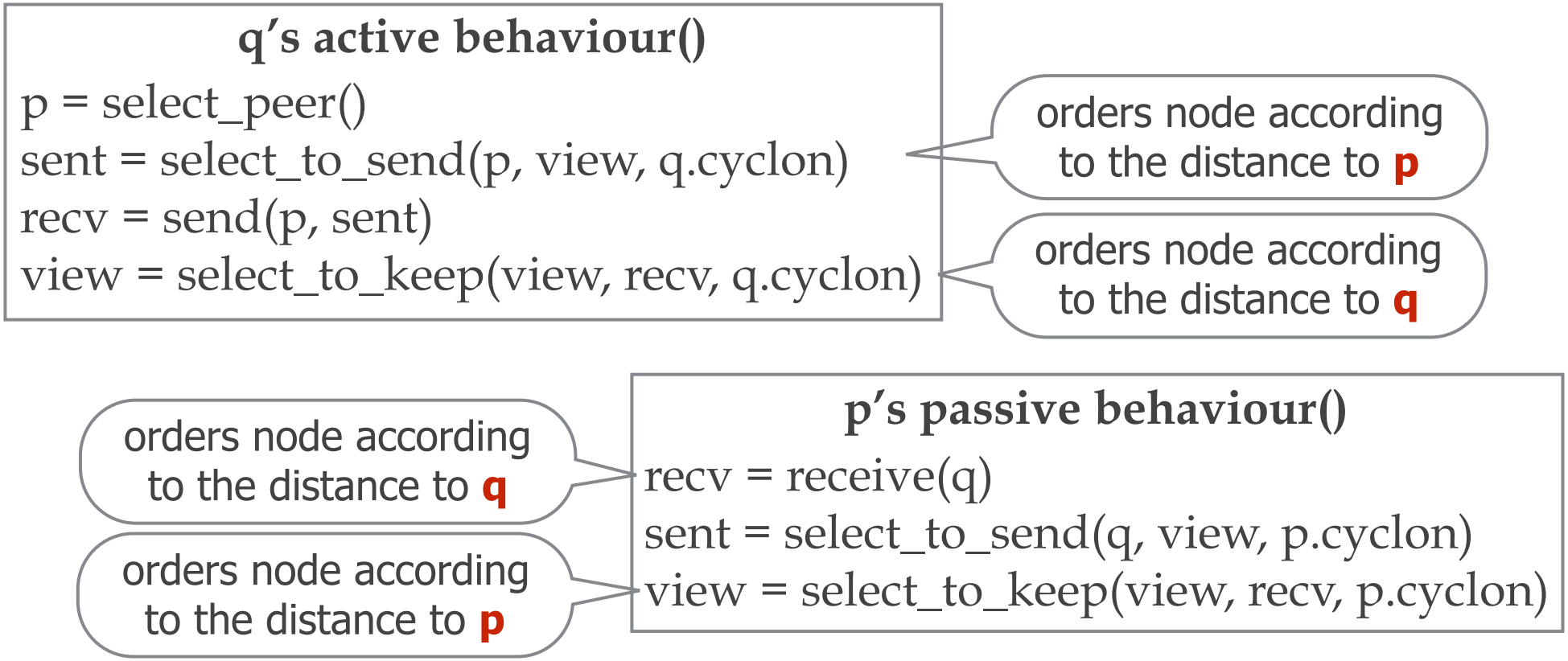
Order your own partial view and the cyclon view according to P position. Select the closest peers

SelectToKeep

Order the received peers according to your own position. Replace the most distant peers in the partial view

Vicinity

- Nodes are **ordered** according to their distance with respect to the target



Beyond Vicinity: T-man

- Vicinity exploits distance between items to provide a ranking of peers
- A generalisation of Vicinity would be to allow any function that provides a ranking
- Such generalisation exists and it's called T-man
  - Let the user define its own ranking systems
  - Vicinity can be seen as a special case of T-man

Jelasity, Márk, Alberto Montresor, and Ozalp Babaoglu. "T-Man: Gossip-based fast overlay topology construction." Computer networks 53.13 (2009): 2321-2339.

T-man

- Objective: build an overlay network (or graph) from scratch filling the partial views of nodes properly
  - It can be even a structured one, like a DHT (we see an example later)
- Initially the views of the nodes can be whatever (even empty). Only requirement is to have an underlying random peer sampling service
  - Exploits **Cyclon** as underlying random peer sampling service
- T-man exploits a ranking function that is a generalisation of the ordering by distance; can do what a simple distance cannot do (DHT)

T-man

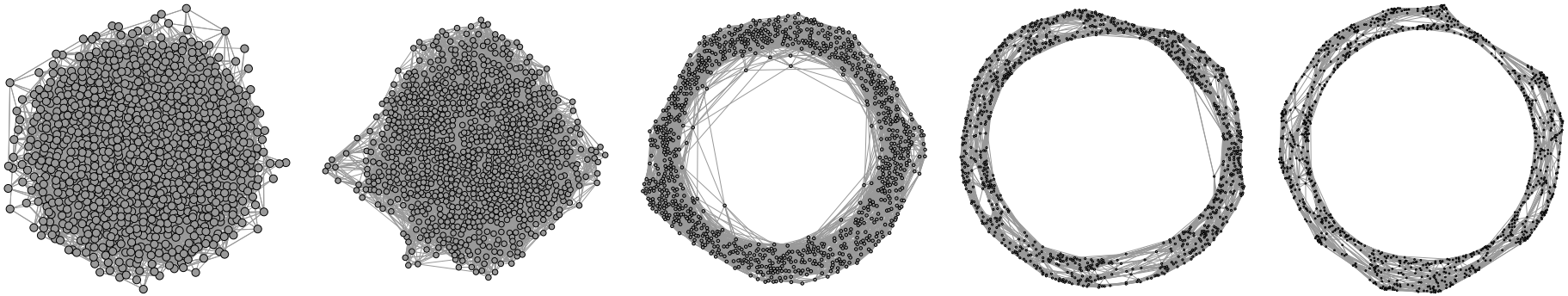
- T-man initial graph evolves to a given overlay, which is defined by means of the ranking function and a number **K**
  - Node[] rank(node [], internal_state)
  - Node exchanges their view and keep the first **K** peers that are ranking higher

T-man

- T-man initial graph evolves to a given overlay, which is defined by means of the ranking function and a number **K**
  - Node[] rank(node [], internal_state)
  - Node exchanges their view and keep the first **K** peers that are ranking higher

T-man

- T-man initial graph evolves to a given overlay, which is defined by means of the ranking function and a number **K**
  - Node[] rank(node [], internal_state)
  - Node exchanges their view and keep the first **K** peers that are ranking higher



T-Man

Node[] **rank**(node [], internal_state)

SelectPeer

SelectToSend

SelectToKeep

T-Man

Node[] **rank**(node [], internal_state)

SelectPeer Select p random among the first **Z** rank(view, my_state)

SelectToSend

SelectToKeep

T-Man

Node[] **rank**(node [], internal_state)

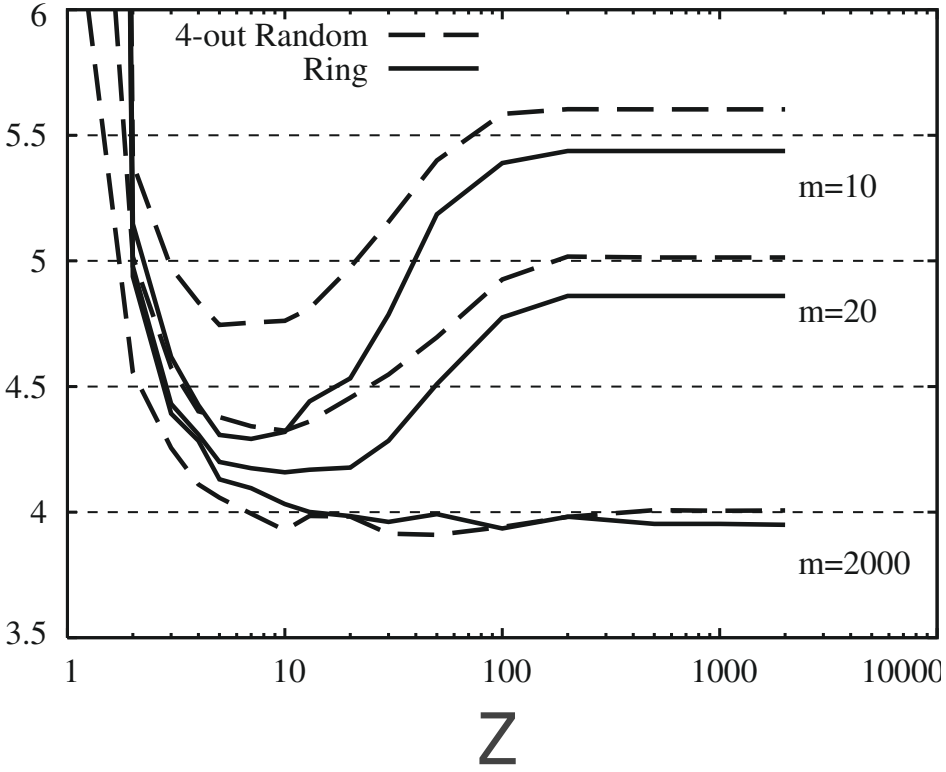
SelectPeer	Select p random among the first **Z** rank(view, my_state)
SelectToSend	send to p the first **m** elements of rank(view+myself, p.state)
SelectToKeep	

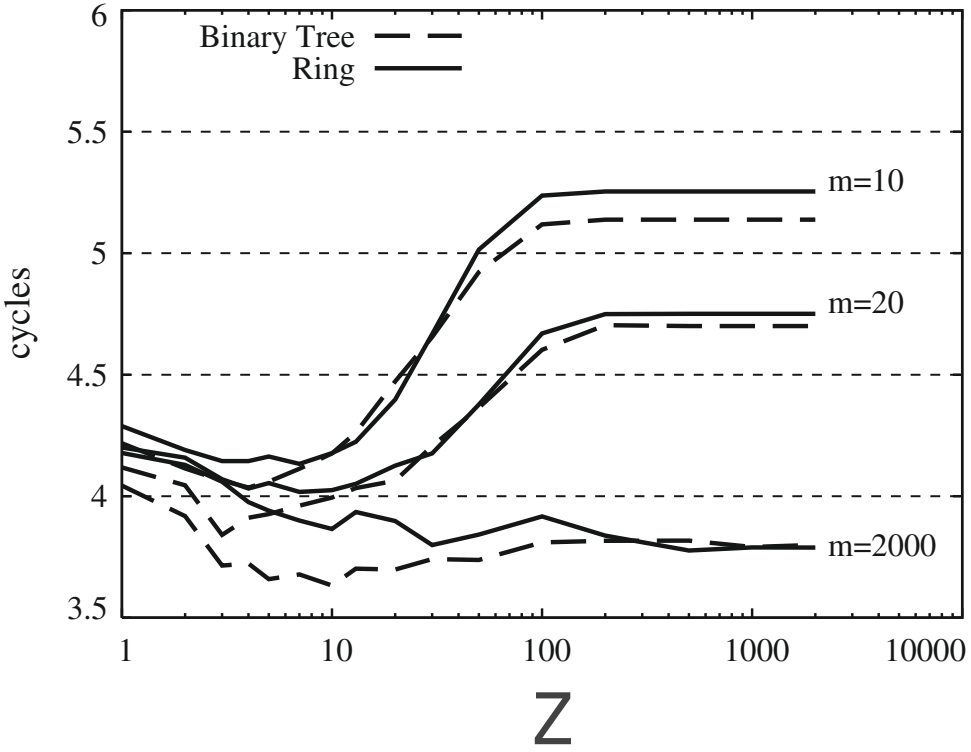
T-Man

Node[] **rank**(node [], internal_state)

SelectPeer	Select p random among the first **Z** rank(view, my_state)
SelectToSend	send to p the first **m** elements of rank(view+myself, p.state)
SelectToKeep	Replace local view with rank(view+p.view, my_state)

T-man performance



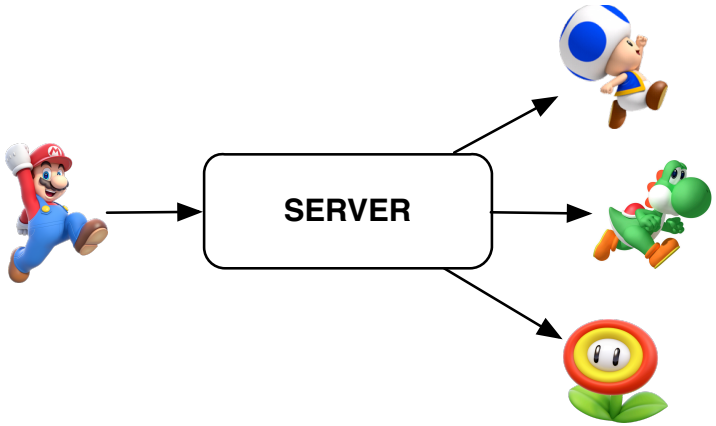


N=2000 nodes

T-man Case Study: Coverage Peer Sampling

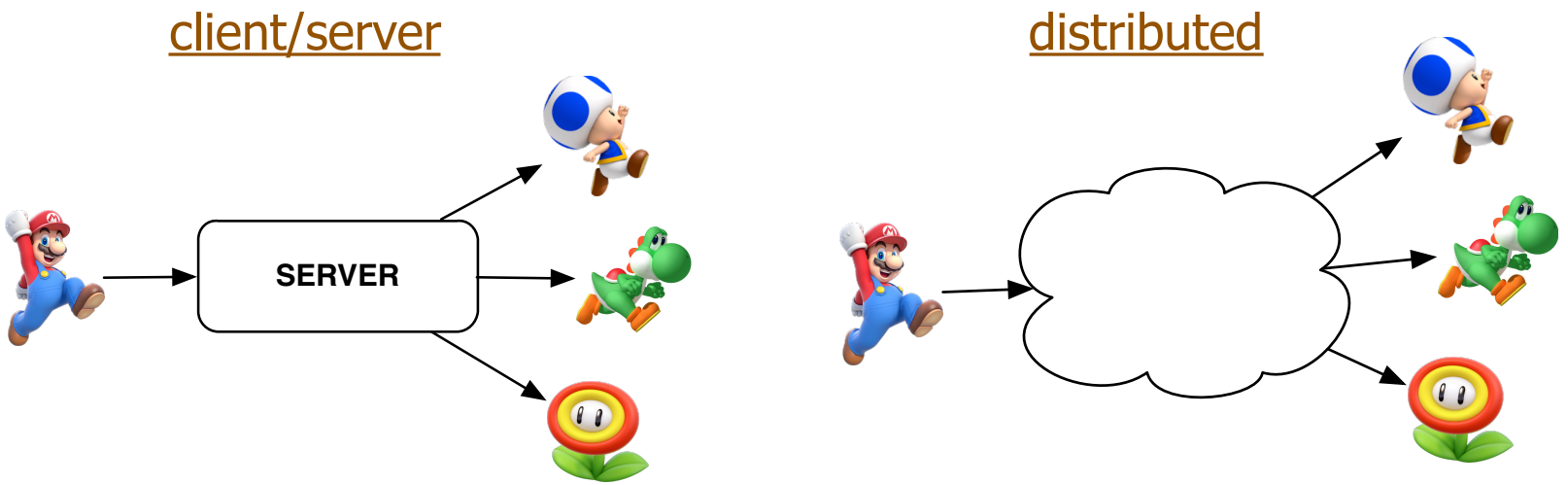
- Position dissemination in distributed online games.
- Players are usually interested in events that happens in their AOI
  - AOI: the area of the virtual world which is of interest for a generic participant
- Events are delivered to players according to the game architecture

client/server



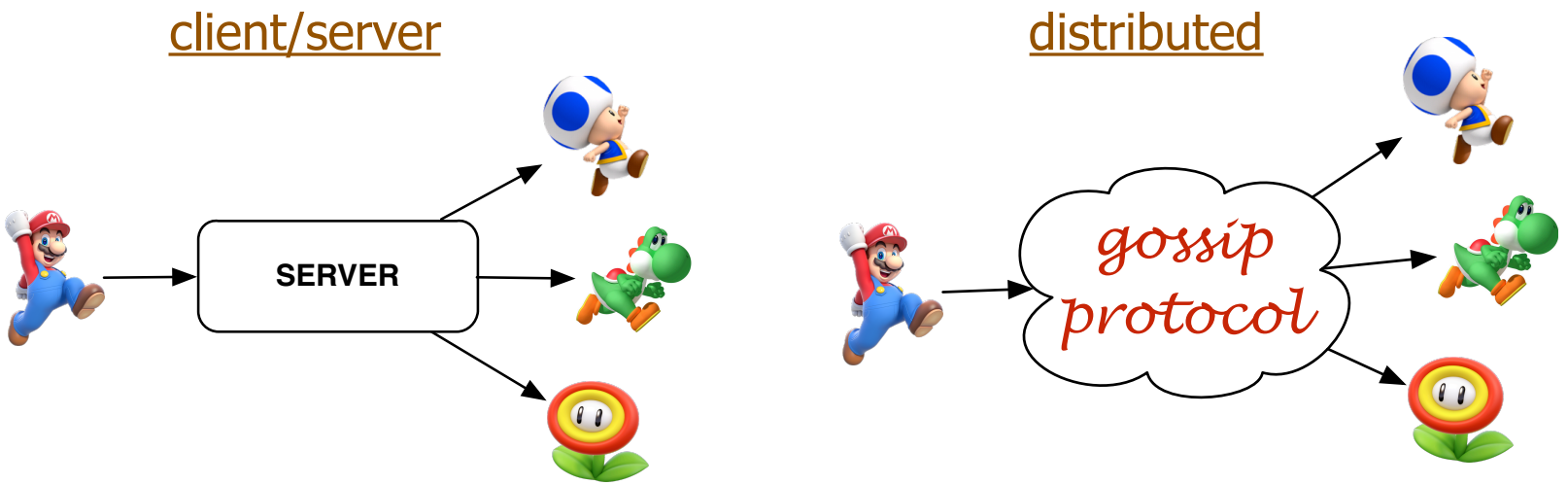
T-man Case Study: Coverage Peer Sampling

- Position dissemination in distributed online games.
- Players are usually interested in events that happens in their AOI
  - AOI: the area of the virtual world which is of interest for a generic participant
- Events are delivered to players according to the game architecture

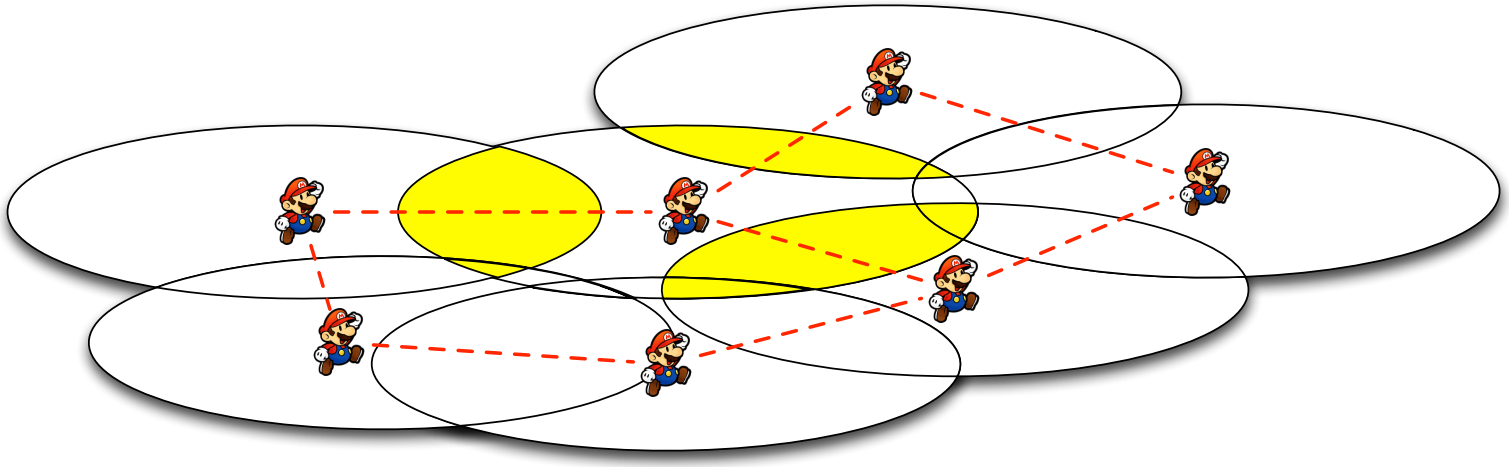


T-man Case Study: Coverage Peer Sampling

- Position dissemination in distributed online games.
- Players are usually interested in events that happens in their AOI
  - AOI: the area of the virtual world which is of interest for a generic participant
- Events are delivered to players according to the game architecture



The problem

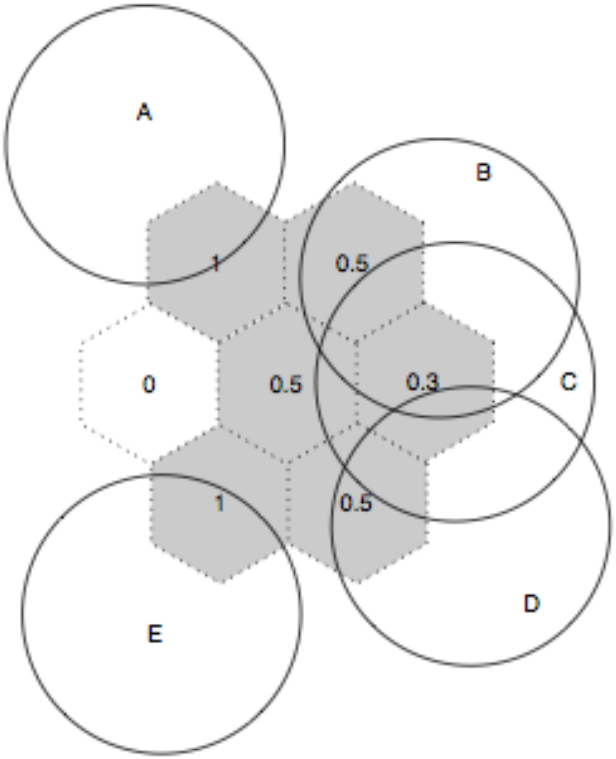


- When there are many players their AOI overlaps, and they can communicate each other the events that happens in the overlapping area. Two issues:
  - I want to keep connection with a **small, fixed** amount of other player (view)
  - Players are moving, so the view must be **updated** over time

Ranking function

- Definition of a function that rank the peers according to their the overlapping of the AOI

- Two functions
  - score-based
  - greedy-based



Score Heuristic with d=3

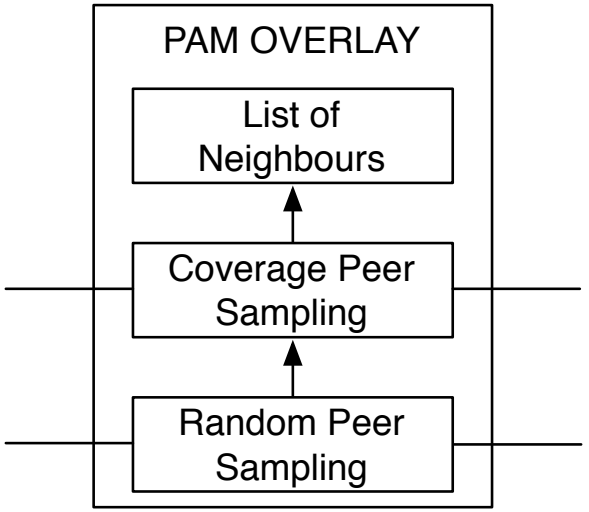
A	B	C	D	E
1	1.3	1.8	0.8	1

Greedy Heuristic with d=3

Step 1		Step 2		Step 3	
A	1	**CA**	5	CAB	5
B	3	CB	4	CAD	5
C	4	CD	4	**CAE**	6
D	2	CE	5		
E	1				

The protocol

- Layered architecture
  - CPS on the top
  - Cyclon on the bottom



- Each peer is identified by a descriptor that contains the position of the player in the virtual world
- SelectToSend
  - Rank peers according to the coverage of the receiver position
- SelectToKeep
  - Rank peers according to the coverage of my position

Lesson Take Away

- Gossip can be used to emulate the selection of a random peer in a completely decentralised network
- In decentralised gossip protocols connective is a huge issue
- Two-layer gossip architectures are very convenient. Bottom layer (i.e. Cyclon) provides connectivity and (random) peer sampling, upper layer focuses on the application

Lesson Take Away

- Definition of a specific ranking function allows to build from scratch complex overlays
- It is possible to combine overlay construction and peer sampling to support complex applications

Further Readings

- Jelasity, Márk, Alberto Montresor, and Ozalp Babaoglu. "**Gossip-based aggregation in large dynamic networks**." ACM Transactions on Computer Systems (TOCS) 23.3 (2005): 219-252.
- Voulgaris, Spyros, and Maarten Van Steen. "**Epidemic-style management of semantic overlays for content-based searching.**" Euro-Par 2005 Parallel Processing. Springer Berlin Heidelberg, 2005. 1143-1152.
- Jelasity, Márk, Alberto Montresor, and Ozalp Babaoglu. "**T-Man: Gossip-based fast overlay topology construction.**" Computer networks 53.13 (2009): 2321-2339.
- Voulgaris, Spyros, Daniela Gavidia, and Maarten Van Steen. "**Cyclon: Inexpensive membership management for unstructured p2p overlays.**" Journal of Network and Systems Management 13.2 (2005): 197-217.
- Jelasity, Márk, et al. "**Gossip-based peer sampling.**" ACM Transactions on Computer Systems (TOCS) 25.3 (2007): 8.
- Jelasity, Márk, et al. "**The peer sampling service: Experimental evaluation of unstructured gossip-based implementations.**" Proceedings of the 5th ACM/IFIP/USENIX international conference on Middleware. Springer-Verlag New York, Inc., 2004.
- Demers, Alan, et al. "**Epidemic algorithms for replicated database maintenance.**" Proceedings of the sixth annual ACM Symposium on Principles of distributed computing. ACM, 1987.
- Montresor, Alberto, Francesco De Pellegrini, and Daniele Miorandi. "**Distributed k-core decomposition.**" Parallel and Distributed Systems, IEEE Transactions on 24.2 (2013): 288-300.
- Jelasity, Márk, Wojtek Kowalczyk, and Maarten Van Steen. **Newscast computing.** Technical Report IR-CS-006, Vrije Universiteit Amsterdam, Department of Computer Science, Amsterdam, The Netherlands, 2003.