Distributed Systems
Master of Science in Engineering in Computer Science

AA 2018/2019

BROADCAST IN PRESENCE OF BYZANTINE PROCESSES ON DYNAMIC DISTRIBUTED SYSTEMS

Distributed system := ABSTRACTION, modeling a set of spatially separate entities, each of these with a certain computational power, that are able to communicate and to coordinate among themselves for reaching a common goal.

Dynamic = continuously changing

What does it change continuously?

- The set of processes composing the systems (**Churns**), i.e. processes may leave or may enter
- <u>**The communication network**</u>

Vehicular Area Network, Sensor Network, P2P Network, Social Network etc.

Dynamic Communication Network main issues

- Processes can directly exchange messages with a subset of all processes
- This subset changes over the time
- Processes may be isolated for a while

Dynamic Network Model

How does the network change?

A **model** that characterize the evolution is required

Many models proposed in the literature, one of the most general is the **TVG** model

Dynamic network model - TVG [CFQS12]

Time Varying Graph (TVG)

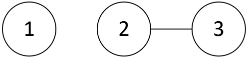
$\mathcal{G} := (V, E, \rho, \zeta)$

- V is the set of nodes;
- $E \subseteq V \times V$ is the set of edges (i.e., communication channels);
- $\rho : E \times \mathbb{N} \rightarrow \{0, 1\}$ is the *presence* function;
- $\zeta : E \times \mathbb{N} \rightarrow \mathbb{N}$ is the *latency* function;

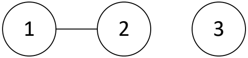
$\mathcal{G}$ can be alternatively described as a **sequence of static graphs (snapshots)** $\mathcal{S}_{\mathcal{G}} = G_0, G_1, \ldots G_T \,:\, G_i := (V, E_i)$, $E_i := \{e \in E \mid \rho(e, t_i) = 1$.

$G := (V, E)$ is called *underlying graph*

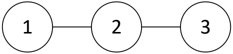
The problems that can be solved depend on the network evolution



$G1$



$G2$



G (underlying)

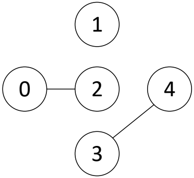
Problem: any process in G can broadcast (when all processes are correct)

In $\mathcal{G} = \{G1, G2\}$ ($\zeta(e,t) = 0 \; \forall t, e$) processes 2 and 3 can achieve broadcast, process 1 can't

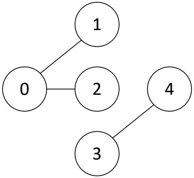
Journey (or Dynamic Path)

A sequence of distinct nodes $(p_1, ..., p_n)$ is a *Journey* (or *Dynamic Path*) from p_1 to p_n if there exists a sequence of dates $(t_1, ..., t_n)$ such that, $\forall i \in \{1, ..., n-1\}$ we have:

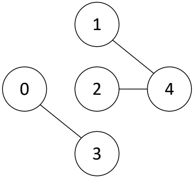
- $e_i = (p_i, p_{i+1}) \in E$, i.e. there exists an edge connecting p_i to p_{i+1}.
- $\forall t \in [t_i, t_i + \zeta(e_i, t_i)], \rho(e_i, t) = 1$, i.e. p_i can send a message to p_{i+1} at date t_i.
- $\zeta(e_i, t_i) \leq t_{i+1} - t_i$, i.e. the aforementioned message is received by date t_{i+1}.



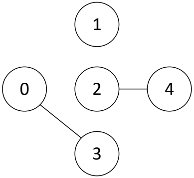
$G1$



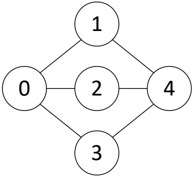
$G2$



$G3$



$G4$



G (underlying)

with $\zeta(e, t) = 1 \; \forall t, e$

$(0, 2, 4)$ is a journey

$(0, 1, 4)$ is not a journey

$(0, 3, 4)$ is not a journey

Classes of TVG have been defined to identify the minimal conditions that dynamic network has to satisfy in order to solve specific distributed systems problems

- Class 1. Temporal Source:
  $\exists u \in V : \forall v \in V, u \rightsquigarrow v$. (broadcast feasible from at least one node)

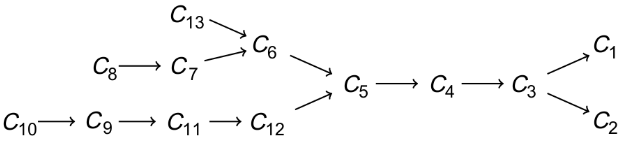
- Class 2. Temporal Sink:
  $\exists u \in V : \forall v \in V, v \rightsquigarrow u$. (compute a function whose input is spread over all the nodes)

- Class 3. Connectivity over time:
  $\forall v, u \in V, v \rightsquigarrow u$. (every node can reach all other once)

- Class 5. Recurrent connectivity:
  $\forall v, u \in V, \forall t \in \mathcal{T}, \exists \mathcal{J} \in \mathcal{J}^*_{(u,v)} : departure(J) > t$.
  (routing can always be achieved over time)
- Class 6. Recurrence of edges:
  $\forall e \in E, \forall t \in \mathcal{T}, \exists t' > t : \rho(e, t') = 1$ and G is connected.
- Class 7. Time-bounded recurrence of edges:
  $\forall e \in E,\ \forall t \in \mathcal{T},\ \exists t' \in [t, t + \Delta),\ \rho(e, t') = 1$ and G is connected.

- Class 8. Periodicity of edges:
  $\forall e \in E,\ \forall t \in \mathcal{T}, \forall k \in \mathbb{N},\ \rho(e,t) = \rho(e, t+kp)$ some $p \in \mathbb{T}$ and G is connected.

- Class 10. T-interval connectivity:
  $\forall i \in \mathbb{N}, \forall T \in \mathbb{N}, \exists G' \subseteq G : V_{G'} = V_G,\ G'$ is connected and $\forall j \in [i, i+T-1), G' \subseteq G_j$.



Reliable Delivery in Dynamic Networks

Issue: can we assume Perfect Point-to-Point Links? In particular, is the *Reliable Delivery* achievable?

A channel may be available for limited time and may fail in delivering a message due to the transmission latency

How implement a Reliable Channel?

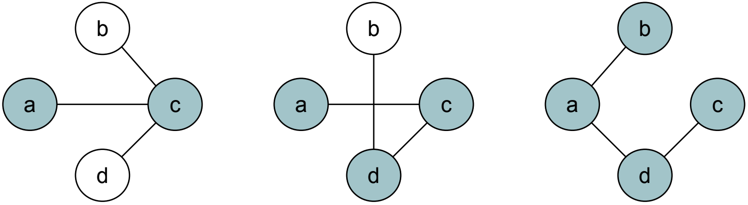
- a process knows exactly ρ and ζ of its channel;
- the channel is up enough to received an *ack* message;
- infinite message retransmission.

NOTE: the Journey definition guarantees *Reliable Delivery* from p to q

Broadcast Latency in 1-interval connected networks [KLO10]

Broadcast Latency on **static** Multi-Hop Networks
$O(n)$ (line topology)

Broadcast Latency on **1-interval connected networks**
$O(n)$ ($\forall t$ at least one not-informed node is connected to an informed node)



G_1 G_2 G_3

Byzantine Reliable Broadcast Specification

Module:

Name: Byzantine Reliable Broadcast, **instance** *brb*, with source *s*.

Events:

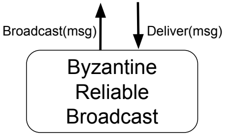
Request: $\langle brb, Broadcast|m\rangle$: Broadcasts a message m to all processes. Executed only by process s.

Indication: $\langle brb, Deliver|p, m\rangle$: Delivers a message m broadcast by p.

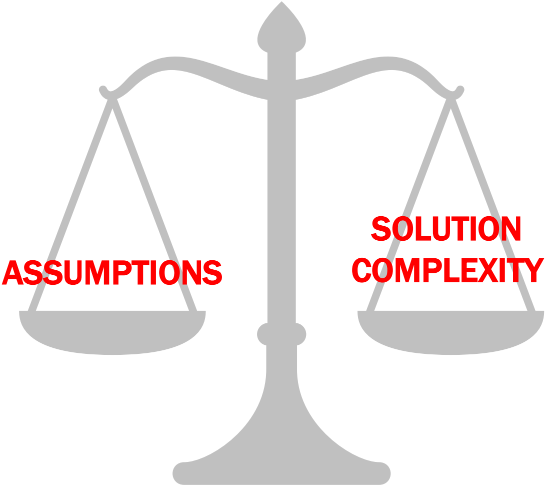
Properties:

RB1: *Safety*: If some correct process delivers a message m with source p and process p is correct, then m was previously broadcast by p.

RB2: *Liveness*: If a correct process p broadcasts a message m, then every correct process eventually delivers m.



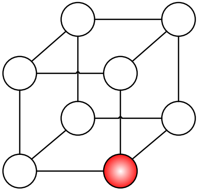
Consistency and *Totality* of Byzantine Reliable Broadcast for complete network are not considered.


ASSUMPTIONS
SOLUTION
COMPLEXITY

Failure Assumptions

Byzantine Failures

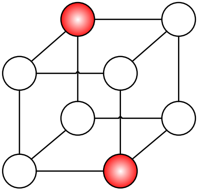
Globally Bounded



f=1

up to *f* faulty processes arbitrarily spread over the system

Locally Bounded



f=1

up to *f* faulty processes in the neighborhood of every process

Specific Spatial Distribution, Probabilistic Distribution, etc.

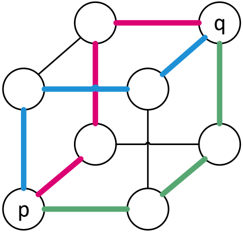
Dynamic Network
Globally Bounded Failure Model
[MTD15]

System Model

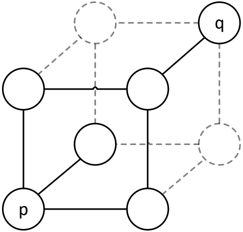
- n processes (each one with an unique identifier);
- **Dynamic Communication Network - TVG**
- processes can be correct or Byzantine faulty;
- up to f processes can be faulty (*globally bounded failures*);
- **processes have no global knowledge (except the value of f)**;
- Authenticated Channels

Graph Theory - Vertex Connectivity

Static Graphs: *Menger Theorem :* **Vertex Cut** = **Disjoint Paths**



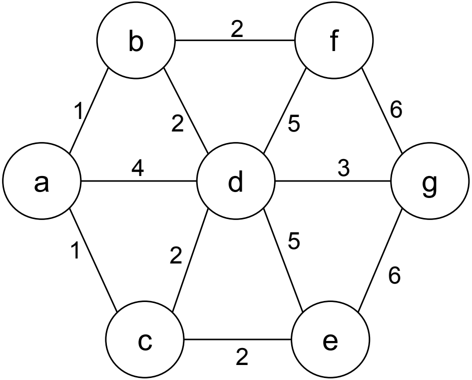
Disjoint Paths



Min Cut

Graph Theory - Vertex Connectivity on Dynamic Networks

Dynamic Graphs: **dynamic Vertex Cut** $\geq$ **dynamic Disjoint Paths** (between two endpoints)



2 disjoint dynamic paths between p and q

Min-Cut between p and q is 3.

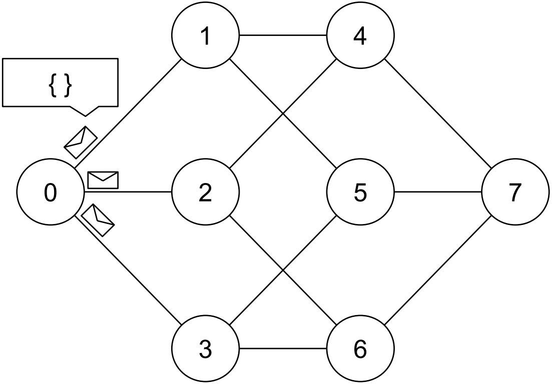
Maurer et al. Algorithm

Extends **Dolev's algorithm on Dynamic Networks**

Idea: leverage the authenticated channels to **collect the ID's of the processes traversed** by a messege

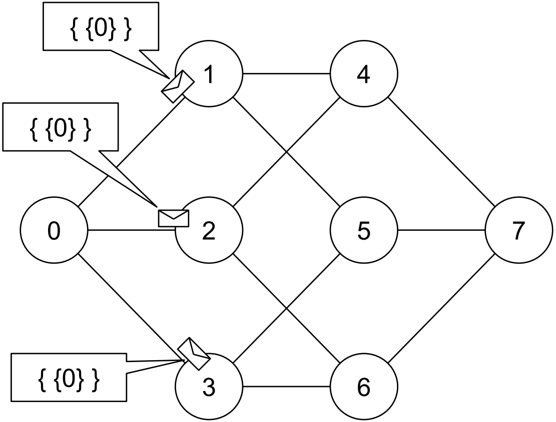
Message format

Dolev (Static DS) Algorithm Graphical Example



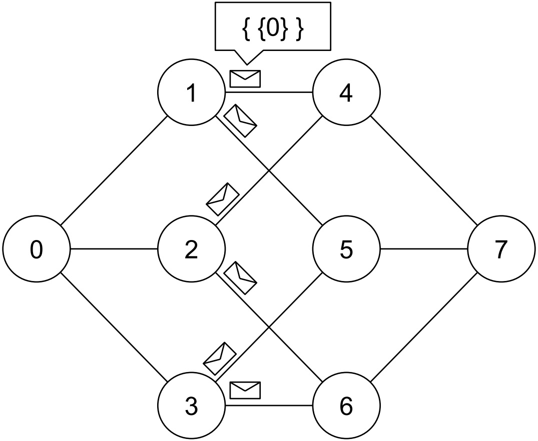
$f = 1$

Dolev (Static DS) Algorithm Graphical Example



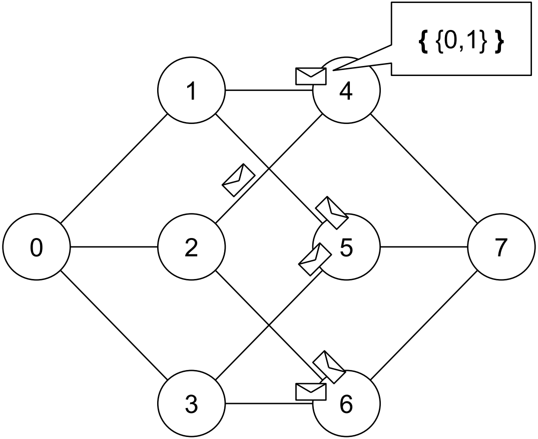
f = 1

Dolev (Static DS) Algorithm Graphical Example



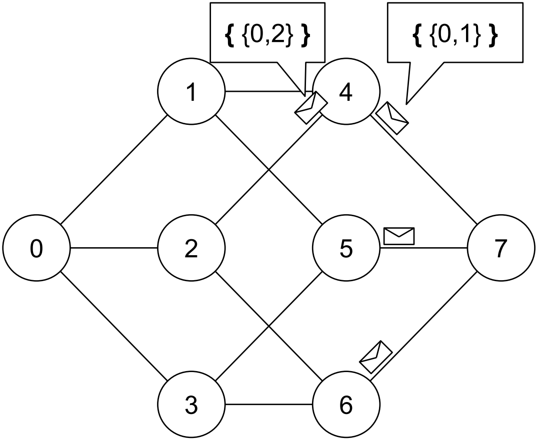
f = 1

Dolev (Static DS) Algorithm Graphical Example



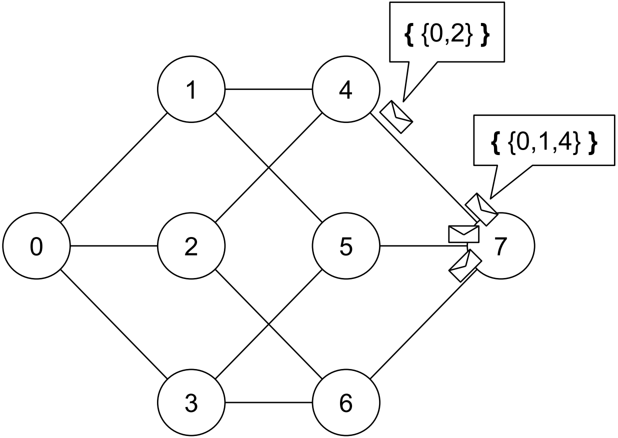
f = 1

Dolev (Static DS) Algorithm Graphical Example



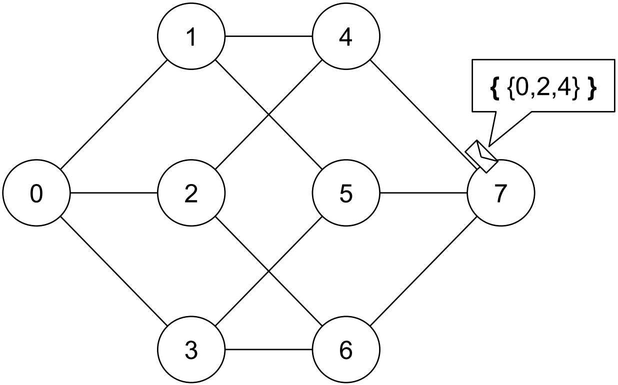
$f = 1$

Dolev (Static DS) Algorithm Graphical Example



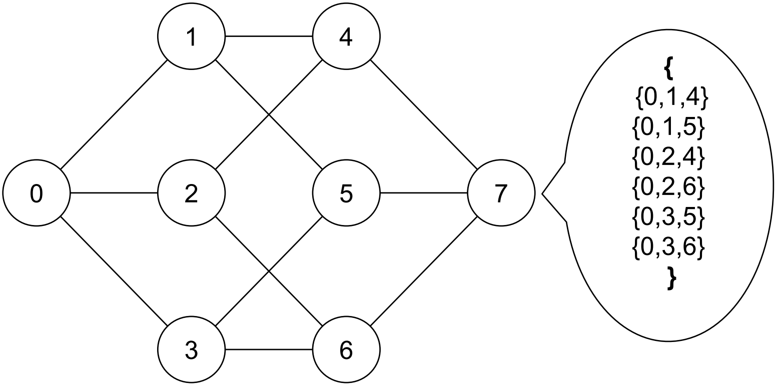
f = 1

Dolev (Static DS) Algorithm Graphical Example



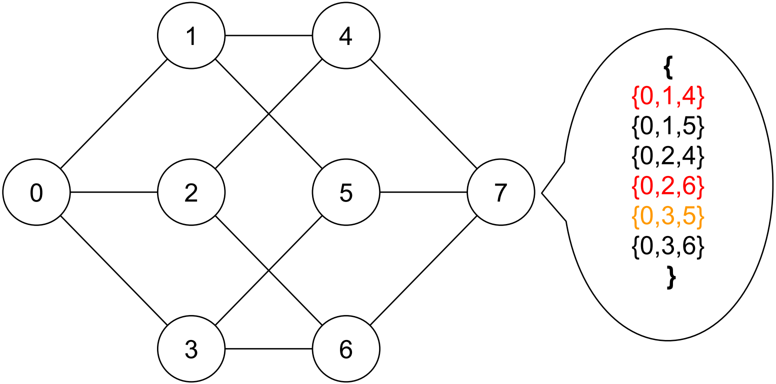
$$f = 1$$

Dolev (Static DS) Algorithm Graphical Example



f $= 1$

Dolev (Static DS) Algorithm Graphical Example



f = 1

Maurer et al. Algorithm

Same propagation algorithm

The **verification algorithm checks** for a **dynamic min-cut** with size $f+1$.

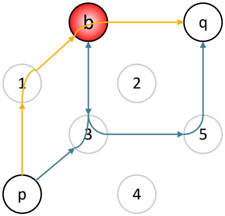
Every message is retransmitted every time a process detects a network change in its neighborhood.

Correctness - Safety

Same as Dolev:
All the messages generated by a Byzantine process are labeled with its ID

$\implies$ Byzantine processes are not able to generate Traversed_Processes with minimum cut lower than f.

$\implies$ Maurer et al. algorithm enforces *safety*



[0,1,b], [p,3,b,5]

Correctness - Liveness (Communication Between two Endpoints)

The Byzantine Reliable Communication (single source, single destination) from process p to q, considering at most f Byzantine processes, is achievable if and only if the **Dynamic Minimum Cut between p and q is at least 2f+1** (i.e. the minimum number of nodes to remove from the network in such a way that no dynamic path exist between p and q)

Maurer et al. algorithm Analysis

Same as Dolev:
Message Complexity: **Exponential in the number of processes** (considering only correct processes)

Delivery Complexity: Solve an **NP-Complete problem** (Min-Cut $\Rightarrow$ Minimum Hitting Set)

$\Rightarrow$ Not practically employable

Is it possible to do better? open research

Additional Issue

Static distributed systems

strict broadcast condition : $2f + 1$-connected network

The **vertex connectivity** of a graph can be **polynomially** verified through a max-flow algorithm

Dynamic distributed systems

strict communication condition : dynamic min-cut at least $2f + 1$

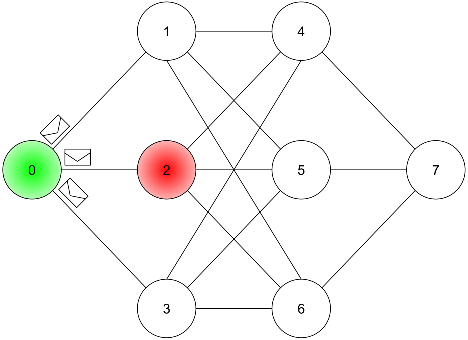
The computation of the **dynamic min-cut** is a **NP-Complete** problem.

Dynamic Network
Locally Bounded Failure Model
[BFT18]

System Model

- n processes (each one with an unique identifier);
- **Dynamic Communication Network - TVG**
- processes can be correct or Byzantine faulty;
- up to f processes can be faulty in the neighborhood of every process (*locally bounded failures*);
- **processes have no global knowledge (except the value of f)**;
- Authenticated Channels

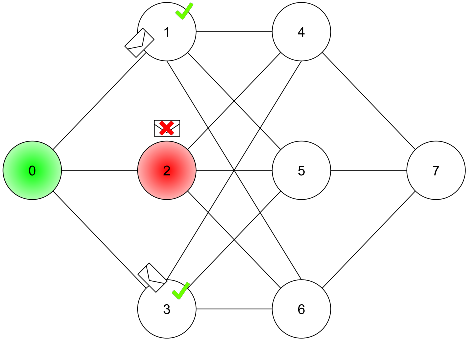
CPA Algorithm (Static DS)



f = 1

- **the source broadcasts the message**;
- a neighbor of the source directly accepts and relays the message;
- a process that receives the same message from $f + 1$ distinct neighbors accepts and relays the message.

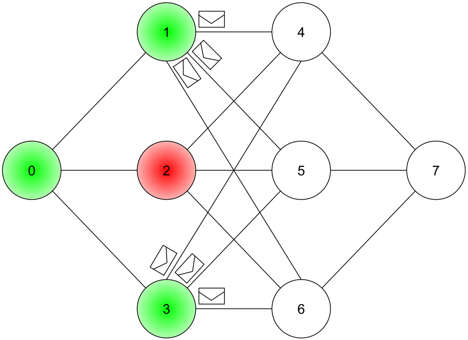
CPA Algorithm (Static DS)



f = 1

- the source broadcasts the message;
- **a neighbor of the source directly accepts and relays the message**;
- a process that receives the same message from $f + 1$ distinct neighbors accepts and relays the message.

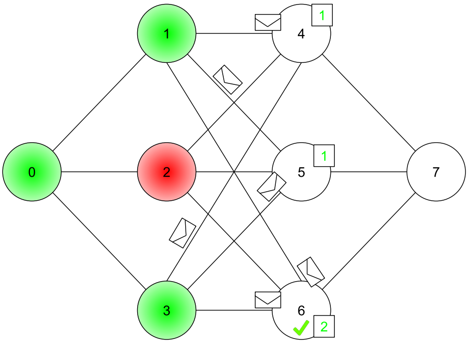
CPA Algorithm (Static DS)



f = 1

- the source broadcasts the message;
- **a neighbor of the source directly accepts and relays the message**;
- a process that receives the same message from $f + 1$ distinct neighbors accepts and relays the message.

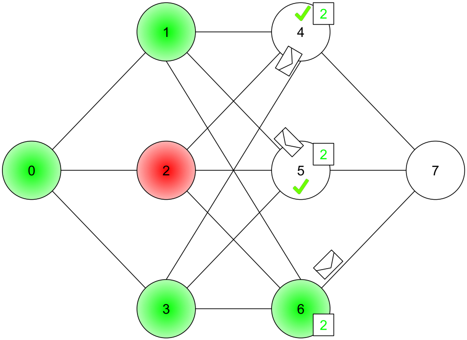
CPA Algorithm (Static DS)



f = 1

- the source broadcasts the message;
- a neighbor of the source directly accepts and relays the message;
- **a process that receives the same message from $f + 1$ distinct neighbors accepts and relays the message.**

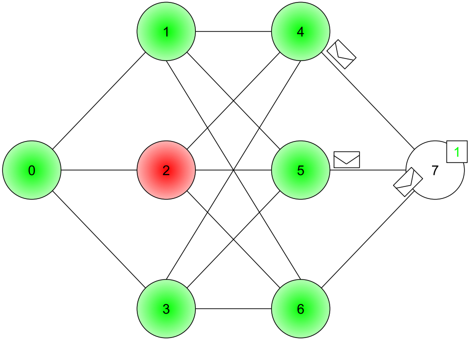
CPA Algorithm (Static DS)



f = 1

- the source broadcasts the message;
- a neighbor of the source directly accepts and relays the message;
- **a process that receives the same message from $f + 1$ distinct neighbors accepts and relays the message.**

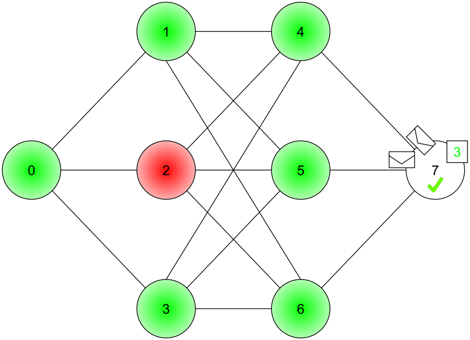
CPA Algorithm (Static DS)



f = 1

- the source broadcasts the message;
- a neighbor of the source directly accepts and relays the message;
- **a process that receives the same message from $f + 1$ distinct neighbors accepts and relays the message.**

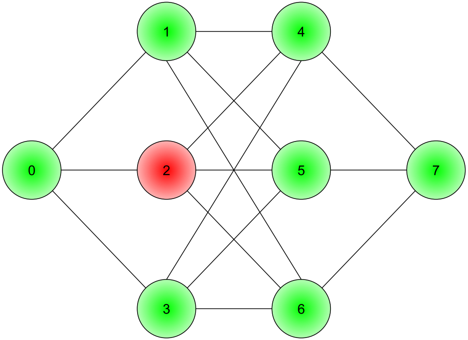
CPA Algorithm (Static DS)



f = 1

- the source broadcasts the message;
- a neighbor of the source directly accepts and relays the message;
- **a process that receives the same message from $f + 1$ distinct neighbors accepts and relays the message.**

CPA Algorithm (Static DS)



f $= 1$

- the source broadcasts the message;
- a neighbor of the source directly accepts and relays the message;
- **a process that receives the same message from $f + 1$ distinct neighbors accepts and relays the message.**

Does the CPA algorithm work on Dynamic Distributed Systems?

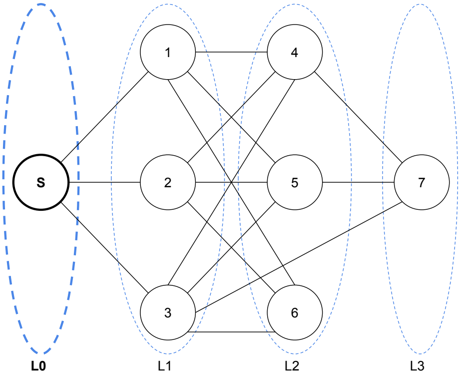
The **safety** property **is still guaranteed** for the same reasons of static systems:

(i) Every process relays a message only if it has been delivered

(i) At most f faulty process are present in the neighborhood

$\implies$ **CPA** algorithm **enforces safety** also on Dynamic Distributed Systems

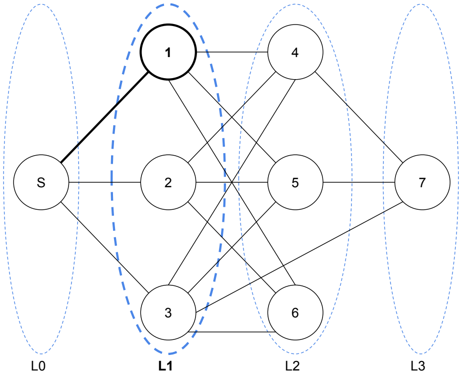
The **liveness** property on static DS requires the existence of a specific partition (MKLO) to be guaranteed



k = 3

- **The source is placed in** $L0$;

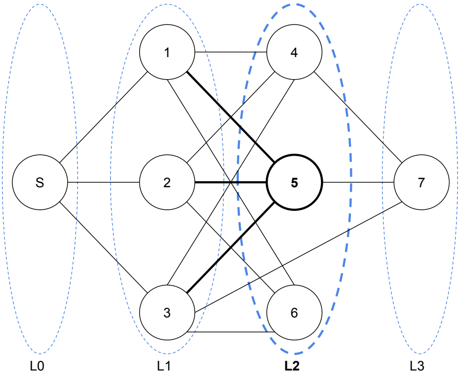
The **liveness** property on static DS requires the existence of a specific partition (MKLO) to be guaranteed



k = 3

- The source is placed in $L0$;
- **The neighbors of the source are placed in level $L1$;**

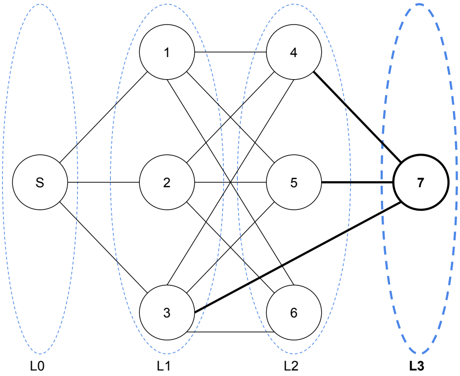
The **liveness** property on static DS requires the existence of a specific partition (MKLO) to be guaranteed



k = 3

- The source is placed in $L0$;
- The neighbors of the source are placed in level $L1$;
- **Any other node is places in the first level such that it has at least k neighbors in the previous levels**.

The **liveness** property on static DS requires the existence of a specific partition (MKLO) to be guaranteed



k = 3

- The source is placed in $L0$;
- The neighbors of the source are placed in level $L1$;
- **Any other node is places in the first level such that it has at least k neighbors in the previous levels**.

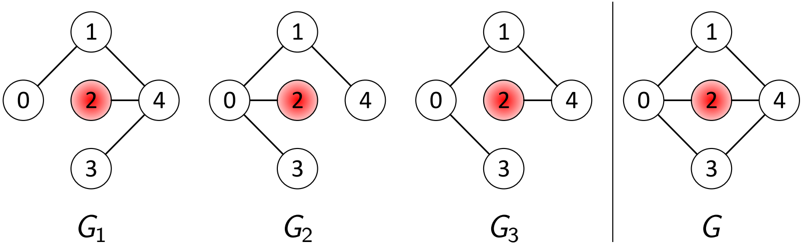
Correctness Necessary and Sufficient condition (Static DS)

Necessary condition: MKLO with $k = f{+}1$

Sufficient condition: MKLO with $k = 2f{+}1$

Strict condition: MKLO with $k = f{+}1$ removing any possible placement of the Byzantine processes (NP-Complete Problem)

On Dynamic DS a MKLO is not enough



1) An edge may disappear while transmitting a message

2) The order of appearance of edges matters

1) An edge may disappear while transmitting a message

A process has to face the unreliability of the channels:

- it knows exactly ρ and ζ of its channel;
- infinite message retransmission (if an ack is received it stops).

We are assuming no global knowledge, thus processes have to infinitively retransmit the delivered messages.

The **CPA algorithm can be ported on Dynamic Distributed Systems without further changes**

Liveness Conditions

Given the TVG, we identify through the predicate $\mathrm{RCD}(p_i, p_j, t')$ the **edge appearances that allow to reliably deliver** a message transmitted over a channel

$$\mathrm{RCD}(p_i, p_j, t') = \begin{cases} \text{true} & \textit{if } \rho(< p_i, p_j >, \tau) = 1, \ \forall \tau \in [t', t' + \zeta(e_{i,j}, t')]. \\ \text{false} & \textit{otherwise}. \end{cases}$$

Liveness Conditions

Liveness condition $\Rightarrow$ **RCD + MKLO**

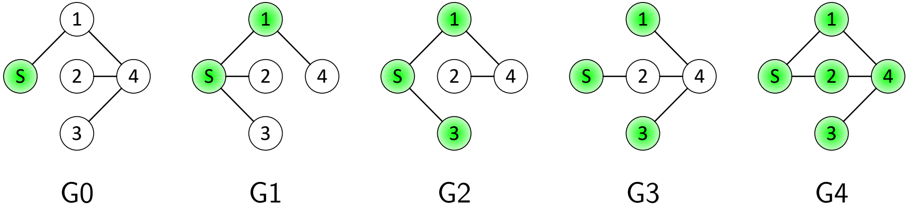
= Temporal Minimum K-level Ordering (TMKLO)

$$\mathcal{A}_k(p_j, t) = \begin{cases} 1 & \text{if } p_j = p_s \text{ with } t \geq t_{br} \qquad \text{(AK1)} \\ 1 & \text{if } \exists\ t' \geq t_{br} \ :\ \text{RCD}(p_s, p_j, t') = \text{true with } t \geq t' + \zeta(e_{s,j}, t') \qquad \text{(AK2)} \\ 1 & \text{if } \exists\ p_1, \ldots, p_k \ :\ \forall i \in [1,k],\ \mathcal{A}_k(p_i, t_i) = 1 \text{ and} \\ & \quad \exists\ t_i' \geq t_i \ :\ \text{RCD}(p_j, p_i, t_i') = \text{true with } t \geq t_i' + \zeta(e_{i,j}, t_i') \qquad \text{(AK3)} \\ 0 & \textit{otherwise} \end{cases}$$

TMKLO $:= p \in L_{t_i}$ iff $t_i = \min t \in \mathbb{N}$ such that $\mathcal{A}_k(p, t_i) = 1$

TMKLO example

constant latency $\delta = 2$



$$TMKLO_{k=2}: \quad L_0 = \{0\}, \quad L_1 = \{1\}, \quad L_2 = \{3\}, \\ L_4 = \{2, 4\}$$

Correctness - Liveness

Necessary condition: TMKLO with $k = f + 1$

Sufficient condition: TMKLO with $k = 2f + 1$

How verify the Liveness Conditions

Liveness Conditions $\implies$ TMKLO computation

TMKLO computation $\implies$ **full TVG knowledge**

$$\mathcal{G} := (V, E, \rho, \zeta)$$

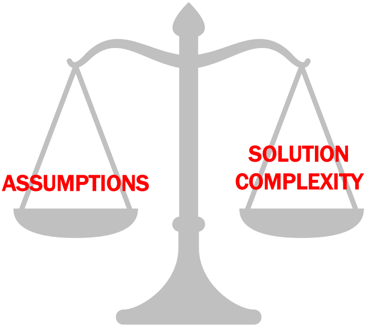
Complexity TMKLO computation: $= O(|V| + |T||E|)$

Dynamic Networks are rarely completely defined

Exceptions are transportation networks, satellite network, etc. ("constrained networks")

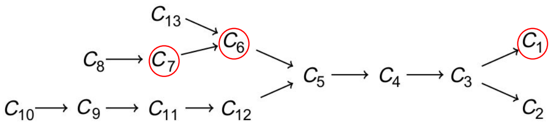
Most of the times just global features are available: connectivity, number of nodes, possible edges that may exists etc.

Liveness Conditions with weaker knowledge



A TMKLO is necessary to enfoce CPA liveness
$\implies$ further assumptions on the TVG are required

Liveness Conditions with weaker knowledge



C1. Temporal Source
C6. Recurrence of edges
C7. Time-bounded recurrence of edges

What happens moving in C6 or C7?

CPA Liveness Condition in C6 and C7

C6: every edge reapers infinitively often
C7. Time-bounded recurrence of edges

Assuming:

- on every edge the *RCD* predicate is true infinitively often
- the underlying graph G is known

MKLO(G) $\implies \exists$ TMKLO($\mathcal{G}$)

Broadcast Latency

How much it takes a broadcast to complete?

Broadcast Latency (BL) := the length of the period between the broadcast start and the last delivery of a correct process.

C1, full knowledge about $\mathcal{G}$, broadcast start at t_{br},

$\exists\ TMKLO_{2f+1}\ :\ P_{2f+1} = \{L_{t_0}, L_{t_1} \dots L_{t_x}\}$,

$t_{max}^{2f+1} = t_x$ (the time associated to the last level of P_{2f+1}).
$t_{max}^{f+1} = t_x$ (the time associated to the last level of P_{f+1}).

$$t_{max}^{f+1} - t_{br} \leq BL \leq t_{max}^{2f+1} - t_{br}$$

Broadcast Latency in C6

C6. every edge reapers infinitively often

assuming that: (i) on every edge the RCD predicate is true infinitively often

(ii) the underlying graph G is known

MKLO(G) $\implies \exists$ TMKLO($\mathcal{G}$)

but it is unknown, because only the knowledge about G is available

$\implies$ **no bounds for BL in C6**

Broadcast Latency in C7

C7. Time-bounded recurrence of edges,
reappearance bound Δ known,

assuming that:

(i) on every appearance makes the *RCD* predicate true,

(ii) the underlying graph G is known,

(iii) $\delta_{max} = \max(\zeta(e,t))$.

$\exists\ MKLO_{2f+1}\ :\ M_{2f+1} = \{L_0, L_1 \ldots L_x\}$,

$S_{2f+1} = |M_{2f+1}|$ (the number of levels in M_{2f+1})

$$\mathbf{BL} \leq \mathbf{S_{2f+1}}(\delta_{\mathbf{max}} + \boldsymbol{\Delta})$$

Final Remarks on Dynamic Distributed Systems

- The complete characterization about the evolution of the systems is seldom available (all the information about a TVG for example)
- Instead, most of the times global information are available (extracted by a network analysis)
- Solving problems becomes more challenging due to the dynamicity of the system
- More attention dedicated the approximate/weaker solution, otherwise too unrealistic assumption have to be considered.

References I

[BFT18] Silvia Bonomi, Giovanni Farina, and Sébastien Tixeuil.
Reliable broadcast in dynamic networks with locally bounded byzantine failures.
In Taisuke Izumi and Petr Kuznetsov, editors, *Stabilization, Safety, and Security of Distributed Systems - 20th International Symposium, SSS 2018, Tokyo, Japan, November 4-7, 2018, Proceedings*, volume 11201 of *Lecture Notes in Computer Science*, pages 170–185. Springer, 2018.
URL: `https://hal.archives-ouvertes.fr/hal-01712277`.

[CFQS12] Arnaud Casteigts, Paola Flocchini, Walter Quattrociocchi, and Nicola Santoro.
Time-varying graphs and dynamic networks.
IJPEDS, 27(5):387–408, 2012.
URL: `https://doi.org/10.1080/17445760.2012.668546`.

[KLO10] Fabian Kuhn, Nancy A. Lynch, and Rotem Oshman.
Distributed computation in dynamic networks.
In Leonard J. Schulman, editor, *Proceedings of the 42nd ACM Symposium on Theory of Computing, STOC 2010, Cambridge, Massachusetts, USA, 5-8 June 2010*, pages 513–522. ACM, 2010.
URL: `https://doi.org/10.1145/1806689.1806760`.

References II

[MTD15] Alexandre Maurer, Sébastien Tixeuil, and Xavier Défago.
Communicating reliably in multihop dynamic networks despite byzantine failures.
In *34th IEEE Symposium on Reliable Distributed Systems, SRDS 2015, Montreal, QC, Canada, September 28 - October 1, 2015*, pages 238–245.
IEEE Computer Society, 2015.
URL: https://doi.org/10.1109/SRDS.2015.10.