CS 60002: Distributed Systems

T2: Models of Distributed Systems

Department of Computer Science and Engineering

INDIAN INSTITUTE OF TECHNOLOGY KHARAGPUR

Sandip Chakraborty
sandipc@cse.iitkgp.ac.in

Communication Abstraction

- We use a simple mode of abstraction as follows.

Communication Abstraction

- We use a simple mode of abstraction as follows.

- The underlying network can be much more complex
  - Heterogeneous network technologies
  - Complex message queueing (RED, WRED, …)
  - Heterogeneous network protocols (Connection-oriented or Connection-less, lossy or lossless)
  - Network performance counters may vary widely (low BDP low loss, low BDP high loss, high BDP low loss, high BDP high loss)

Latency vs Bandwidth

- **Latency**:
  - In the same datacenter ~1 ms
  - Two datacenters over same continent ~10ms
  - Geo-distributed datacenters over different continent ~100ms

- **Bandwidth**:
  - 4G Cellular data ~30Mbps
  - 5G Cellular data ~1 Gbps
  - Home broadband ~100Mbps
  - Inter-DC network ~10Gbps

AWS Snowball

- Use physical storage devices to transfer large amount of data between Amazon S3 and Local Datacenter Storage

- Faster-than-Internet !

- **High Latency High Bandwidth**
  - Latency ~1 Day
  - Bandwidth ~50 TB

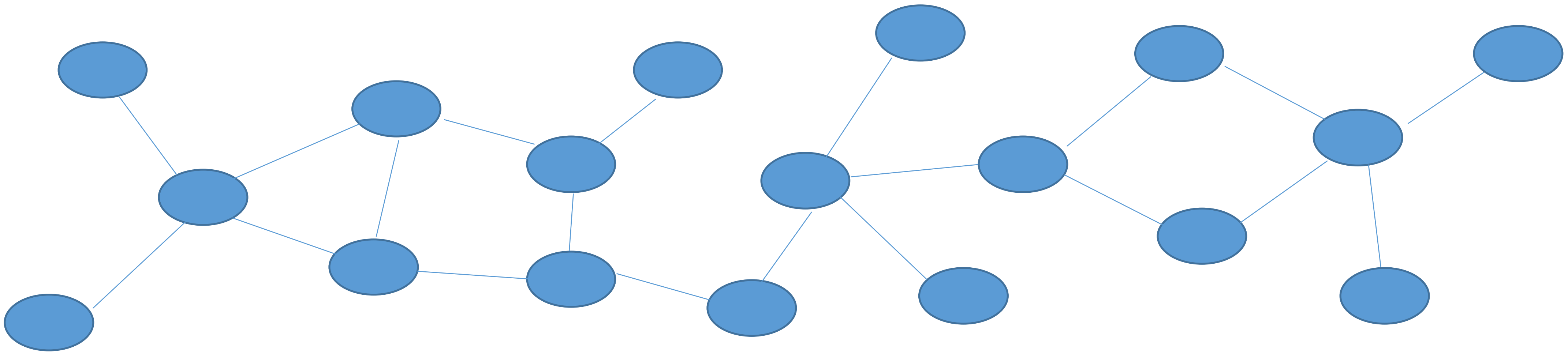
Architecture of the Communication Network

- The underlying physical network is like the Internet; you have switches, routers, gateways – but we are not concerned about them.

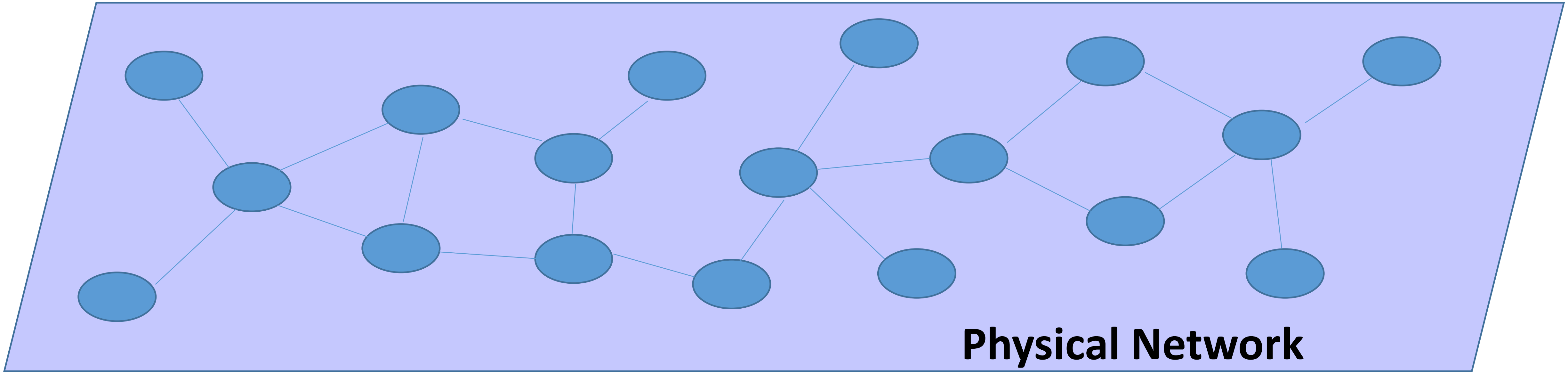
- We are more concerned of the application-level logical network; often termed as the **overlay network.**
  - **May be a Peer-to-Peer (P2P) overlay network**

- **Example:** BitTorrent network
  - Your BitTorrent client is connected to few other peers from where you download part of the files
  - The peers are directly connected to you over the overlay network
  - However, there might be multiple other physical network nodes in-between

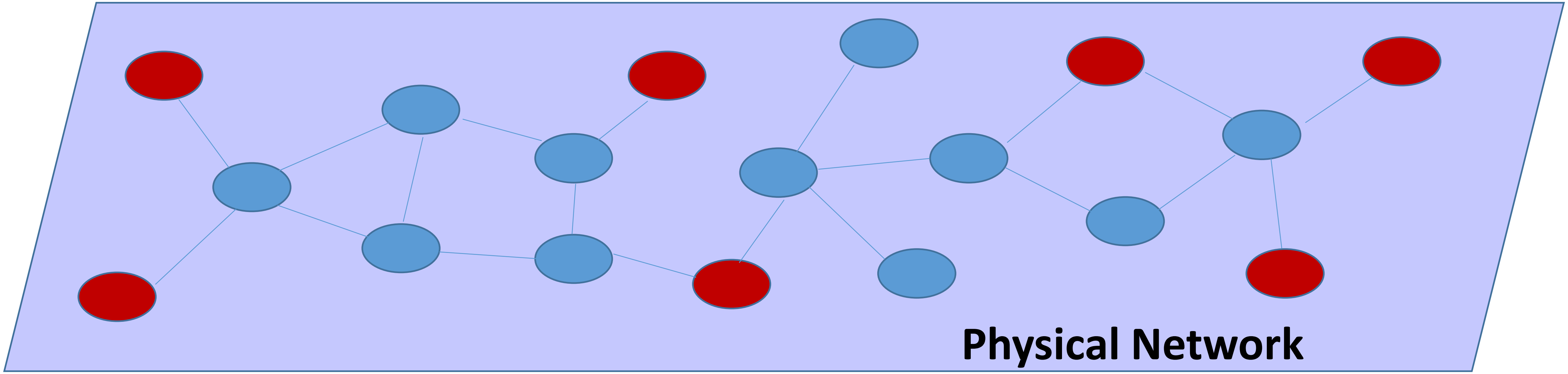
Overlay Network



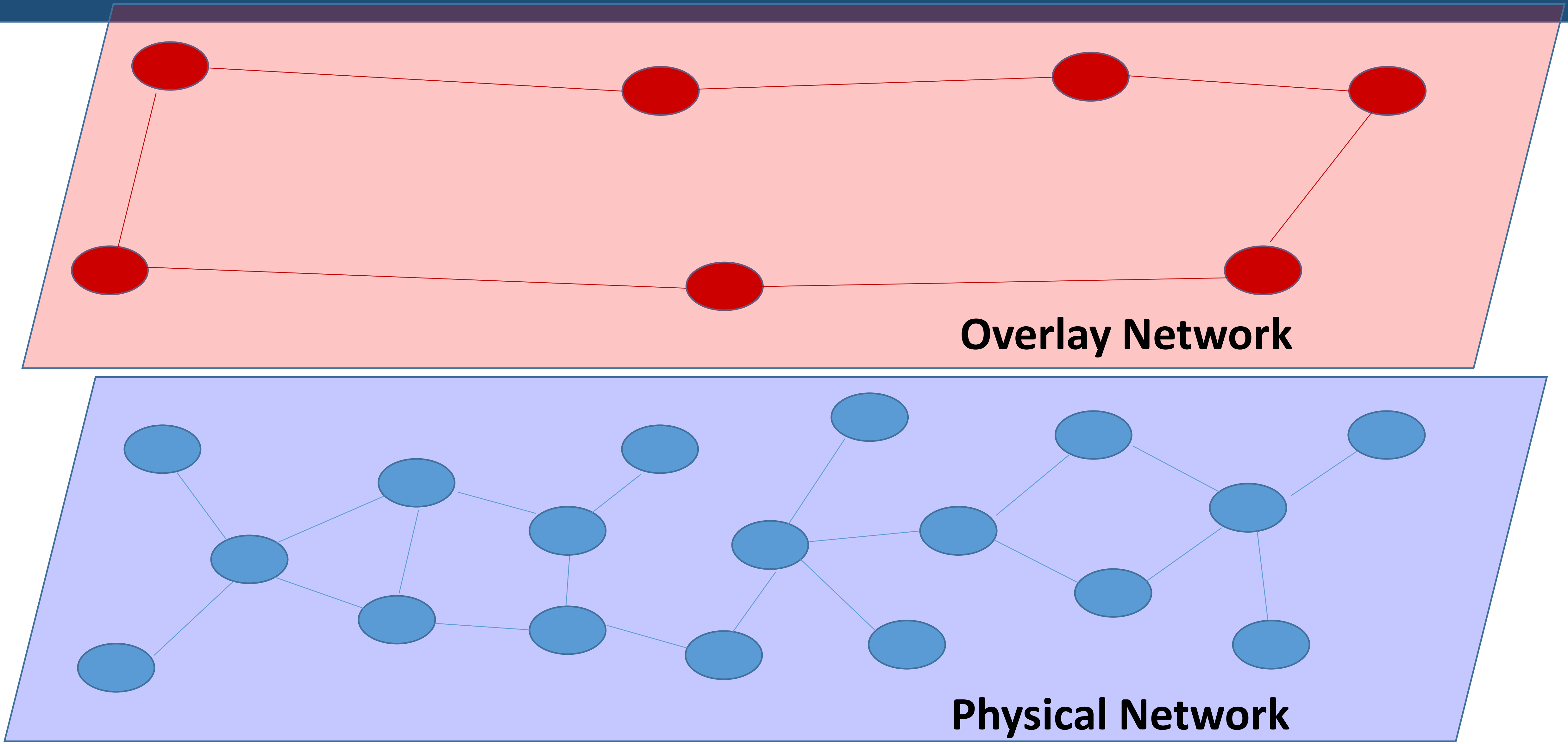
Overlay Network



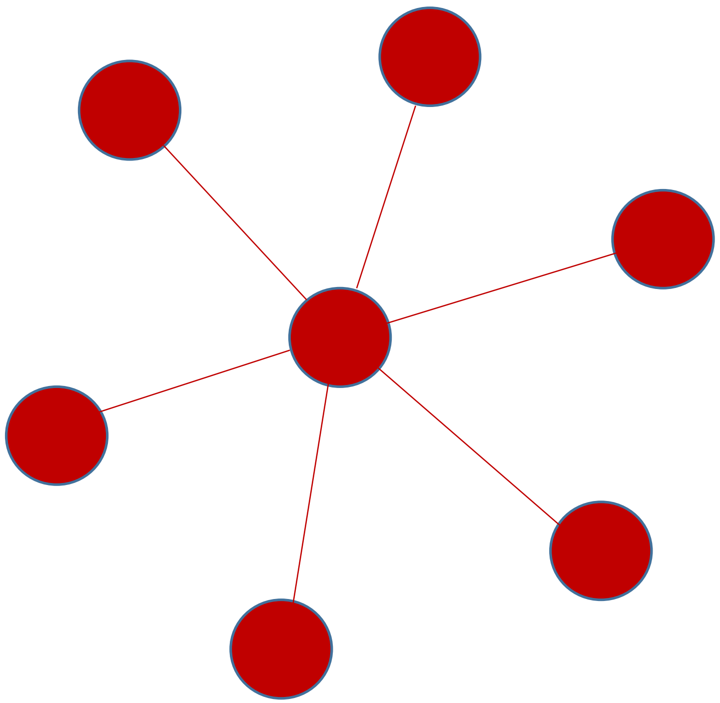
Overlay Network



Overlay Network

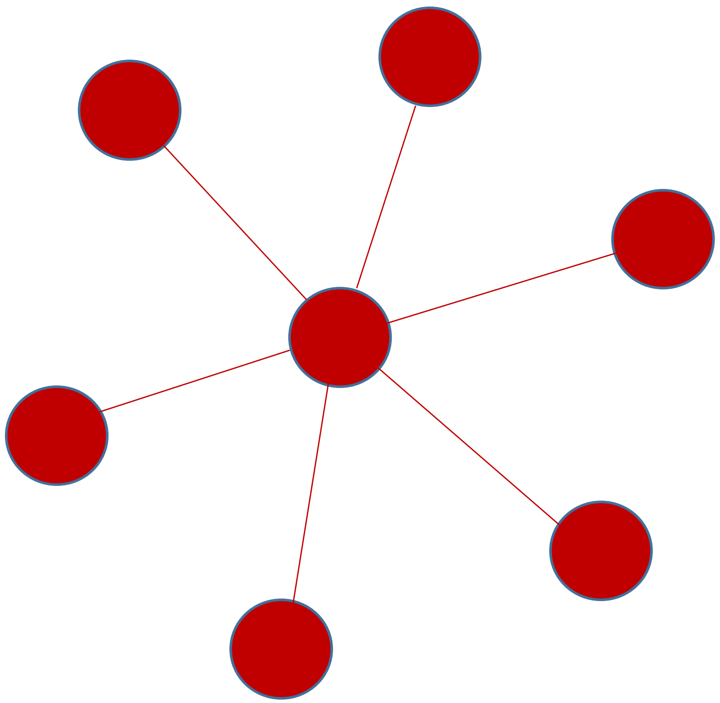


Typical Topology over an Overlay Network

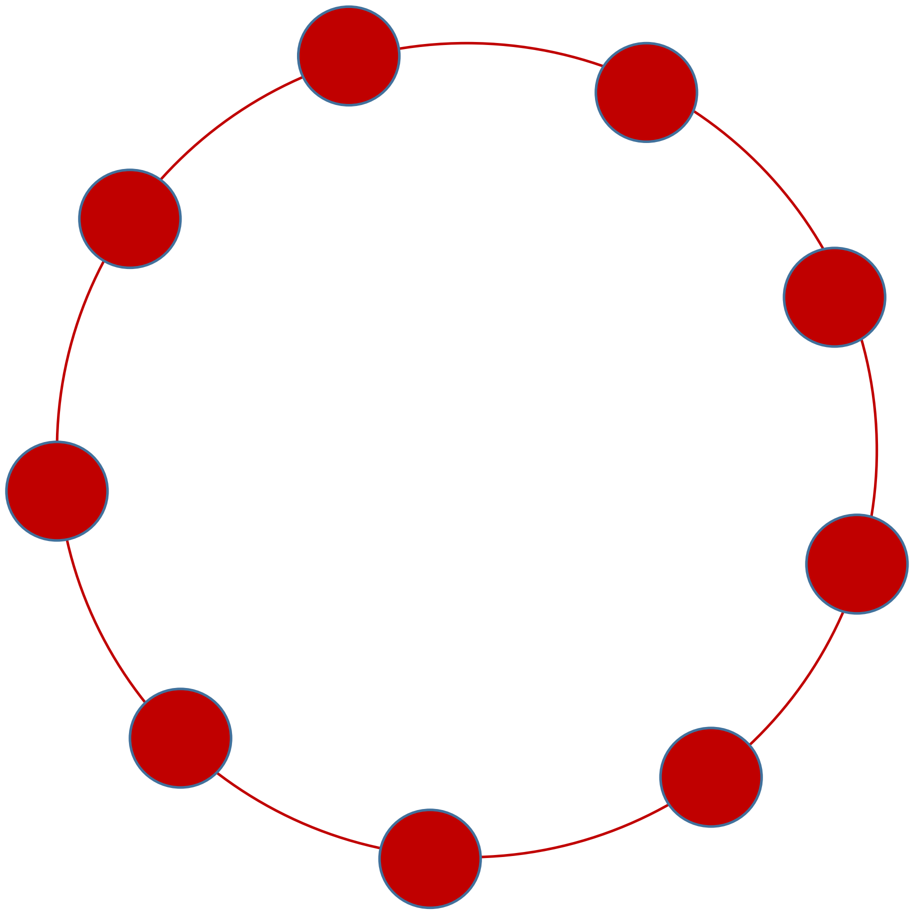


Star

Typical Topology over an Overlay Network

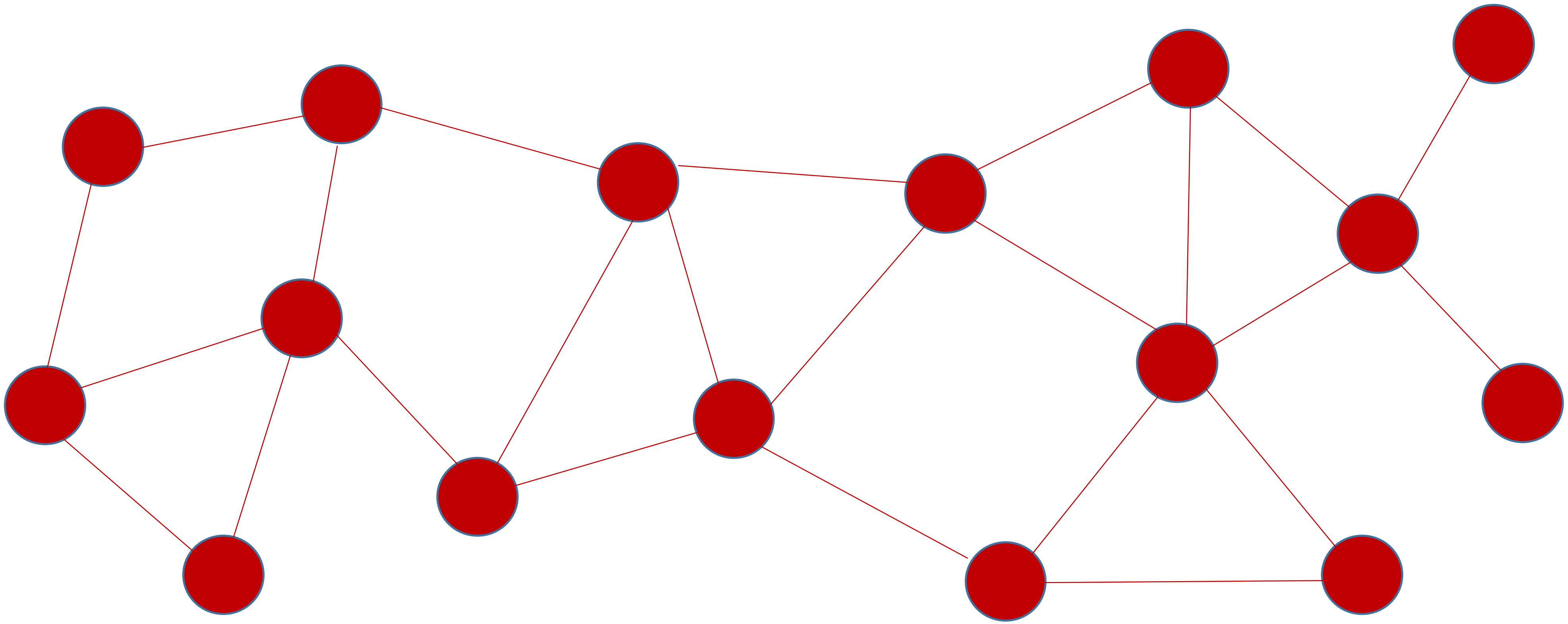


Star



Ring

Typical Topology over an Overlay Network



Mesh

Communication Paradigm – The Gossip Protocol

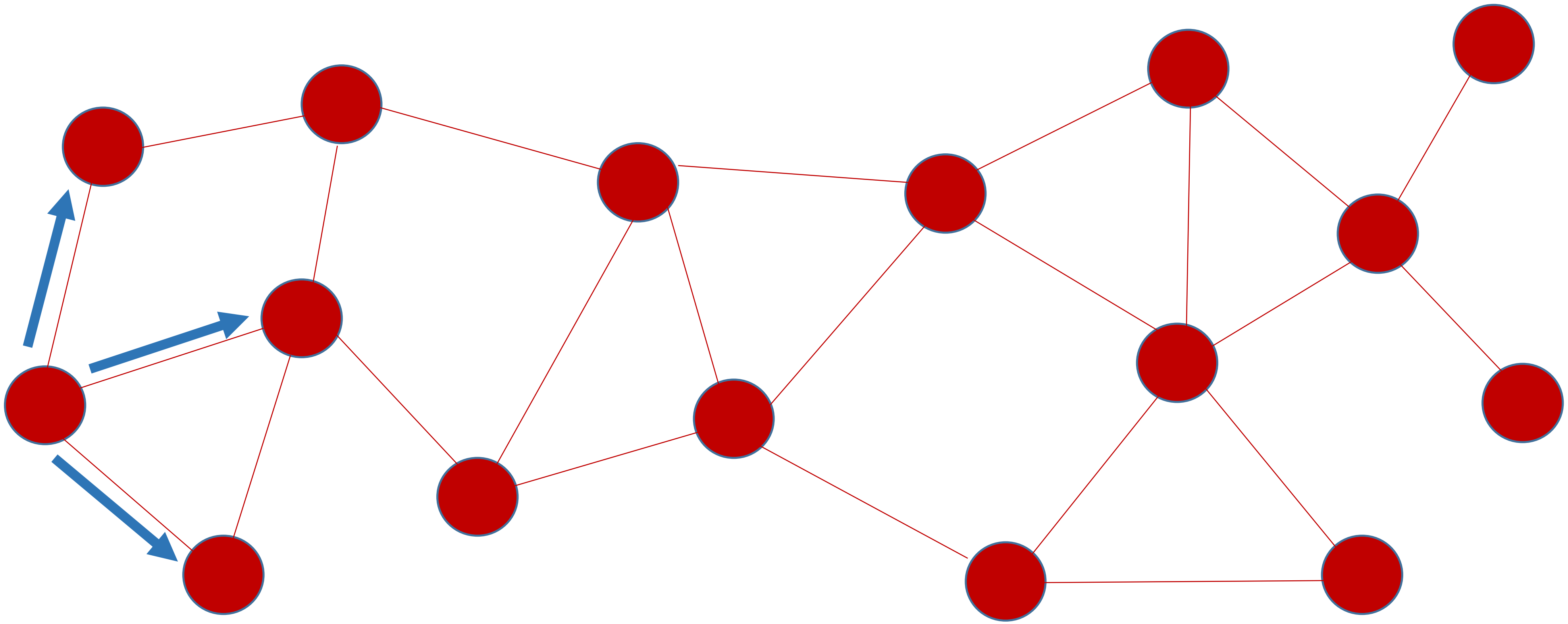
- In a distributed system, the typical model of communication is through message broadcast (an <u>overlay broadcast</u>).
  - How do you broadcast the message when all the nodes are not directly connected to each other over the overlay network?

Communication Paradigm – The Gossip Protocol

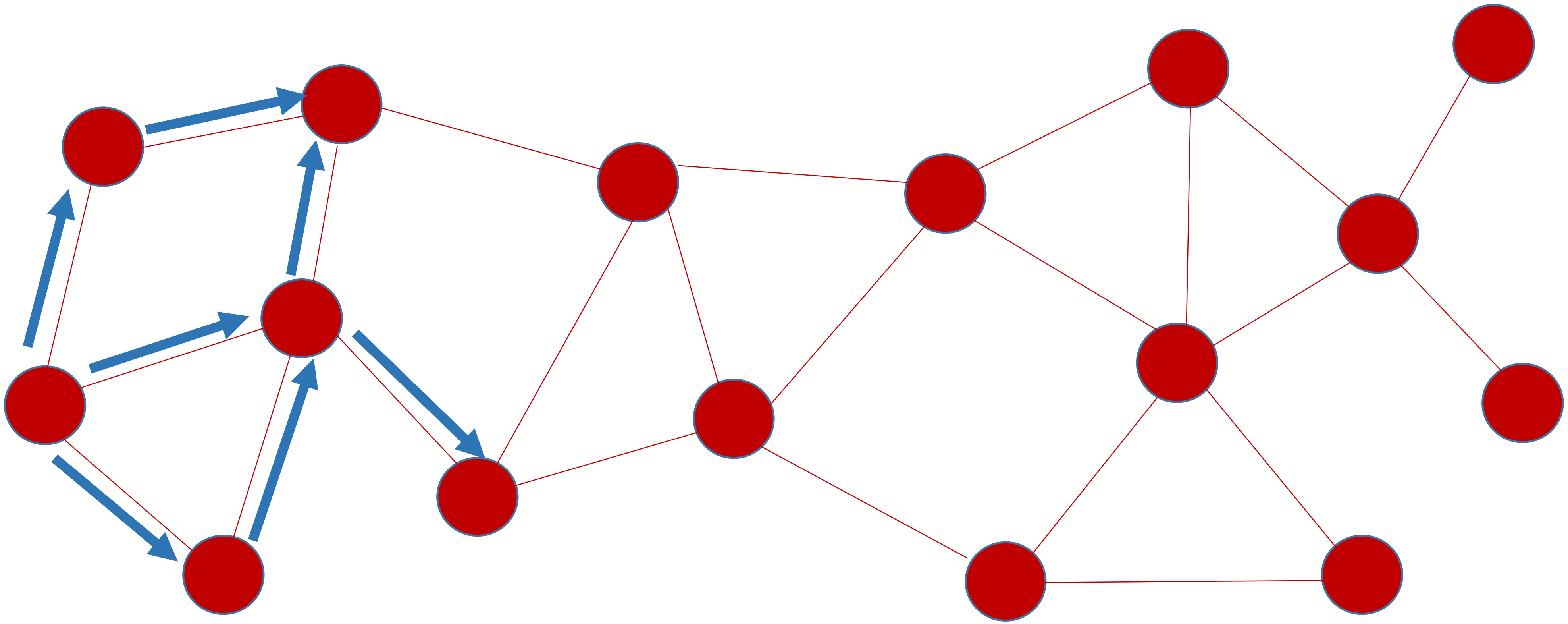
- In a distributed system, the typical model of communication is through message broadcast (an overlay broadcast).
  - How do you broadcast the message when all the nodes are not directly connected to each other over the overlay network?

- **The Gossip Protocol:**
  - Propagate messages based on the way epidemics spread
  - **Gossip**: Think of the office workers spreading rumors

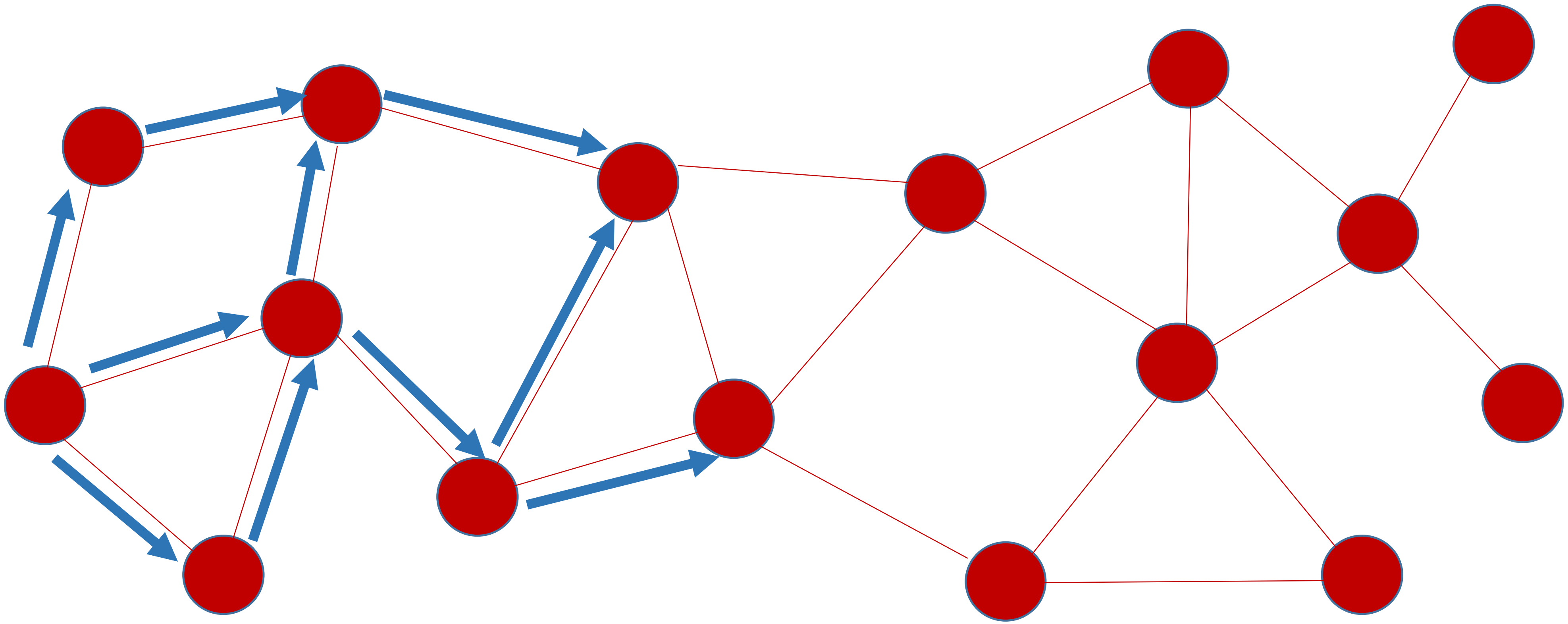
The Gossip Protocol



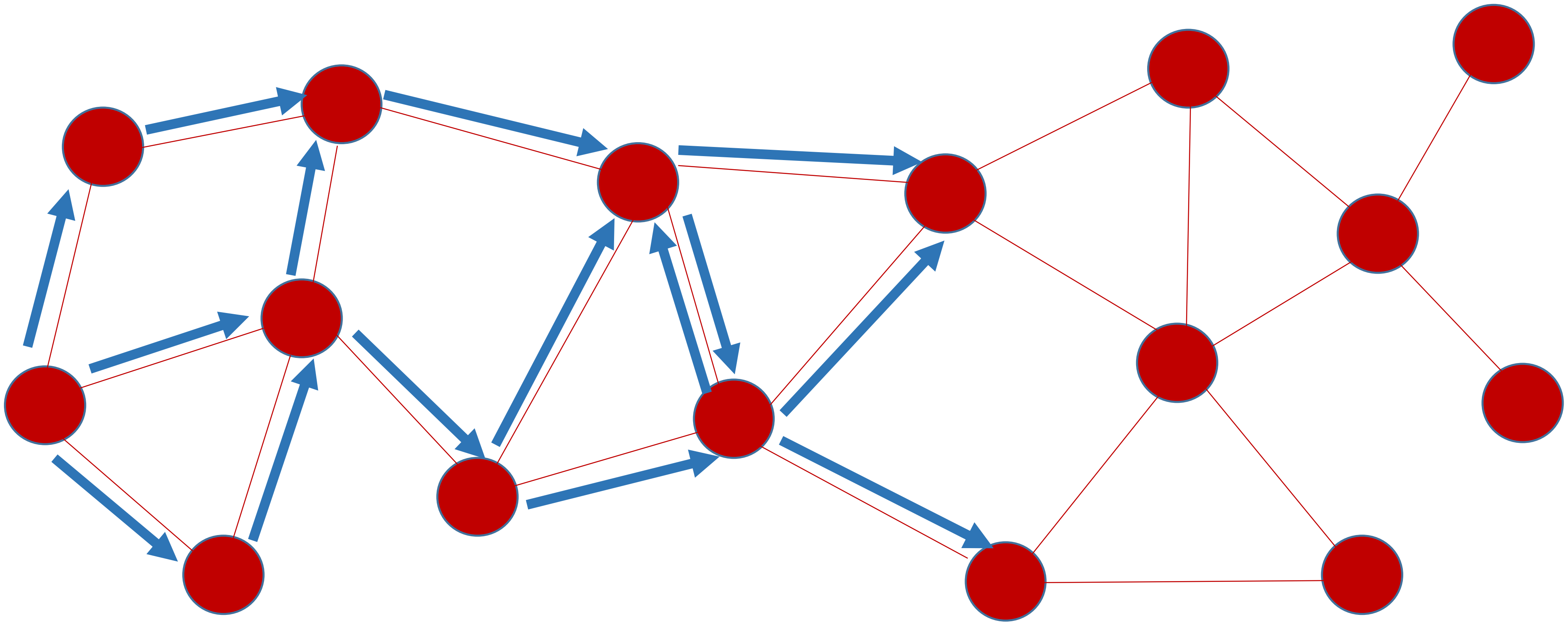
The Gossip Protocol



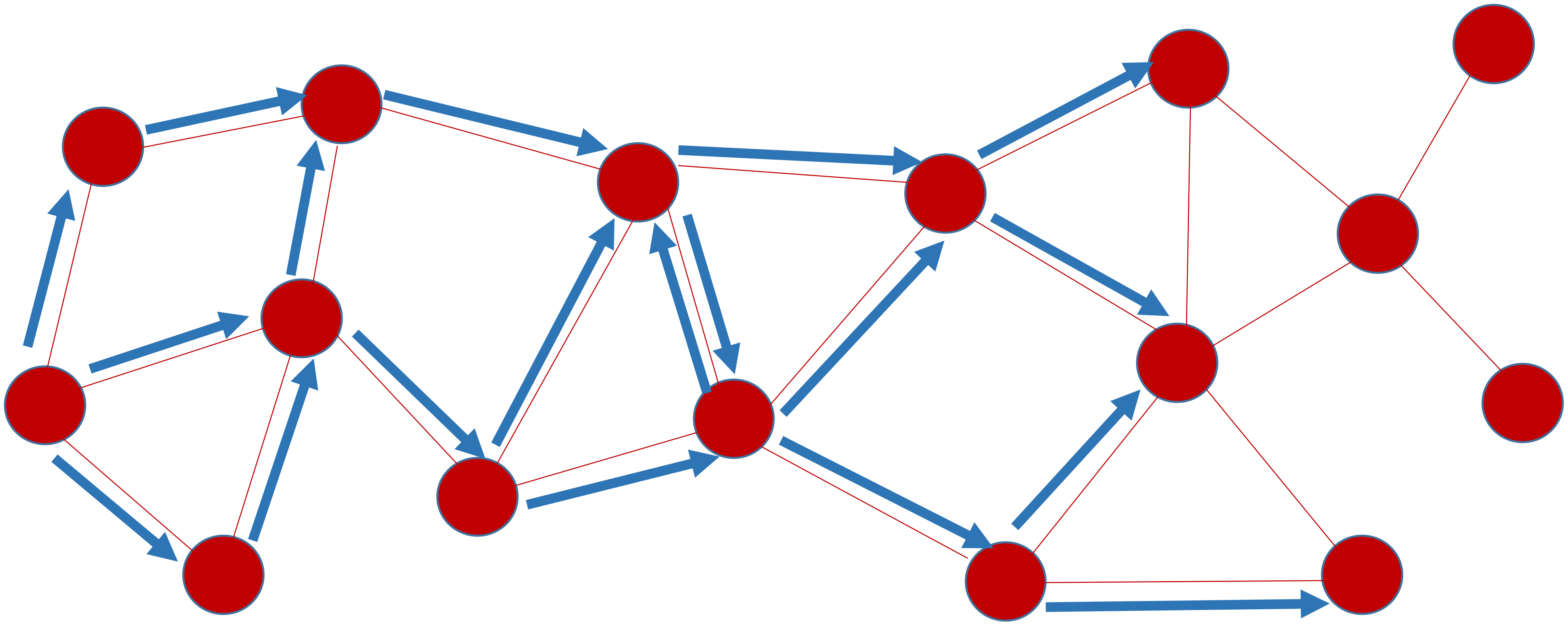
The Gossip Protocol



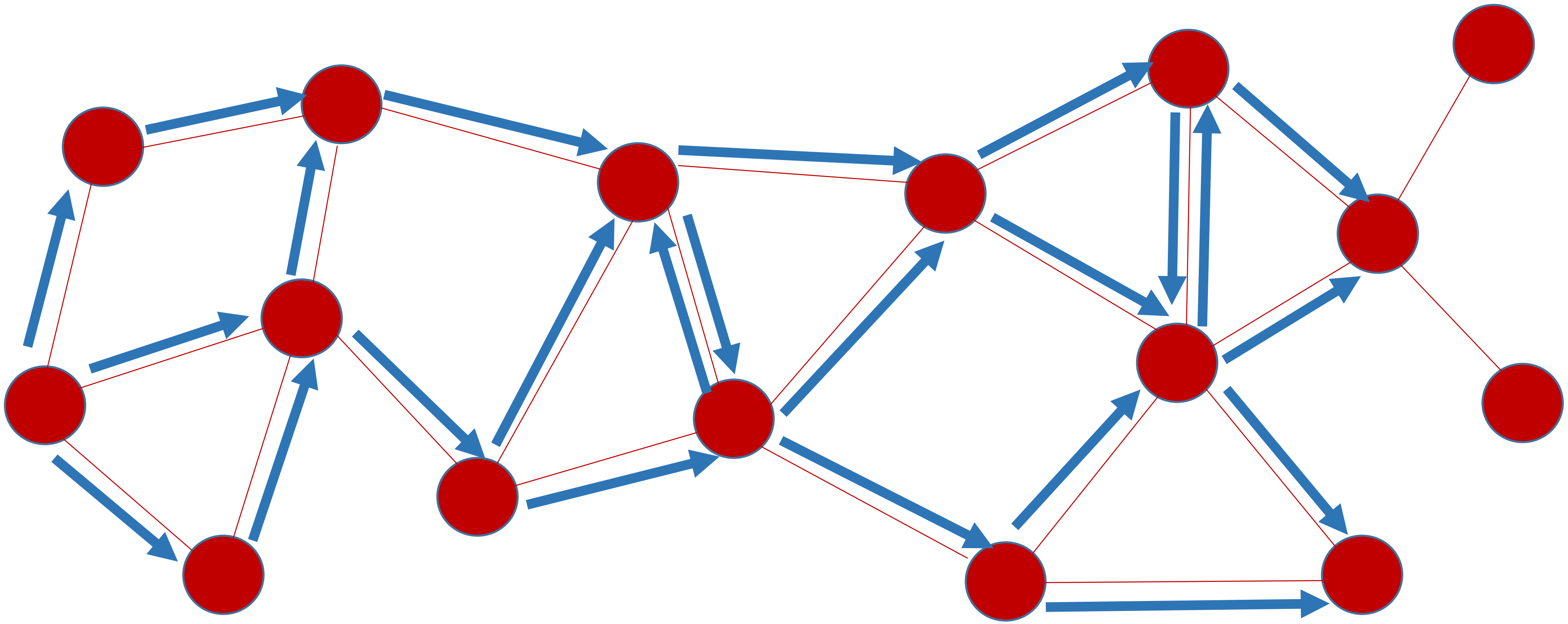
The Gossip Protocol



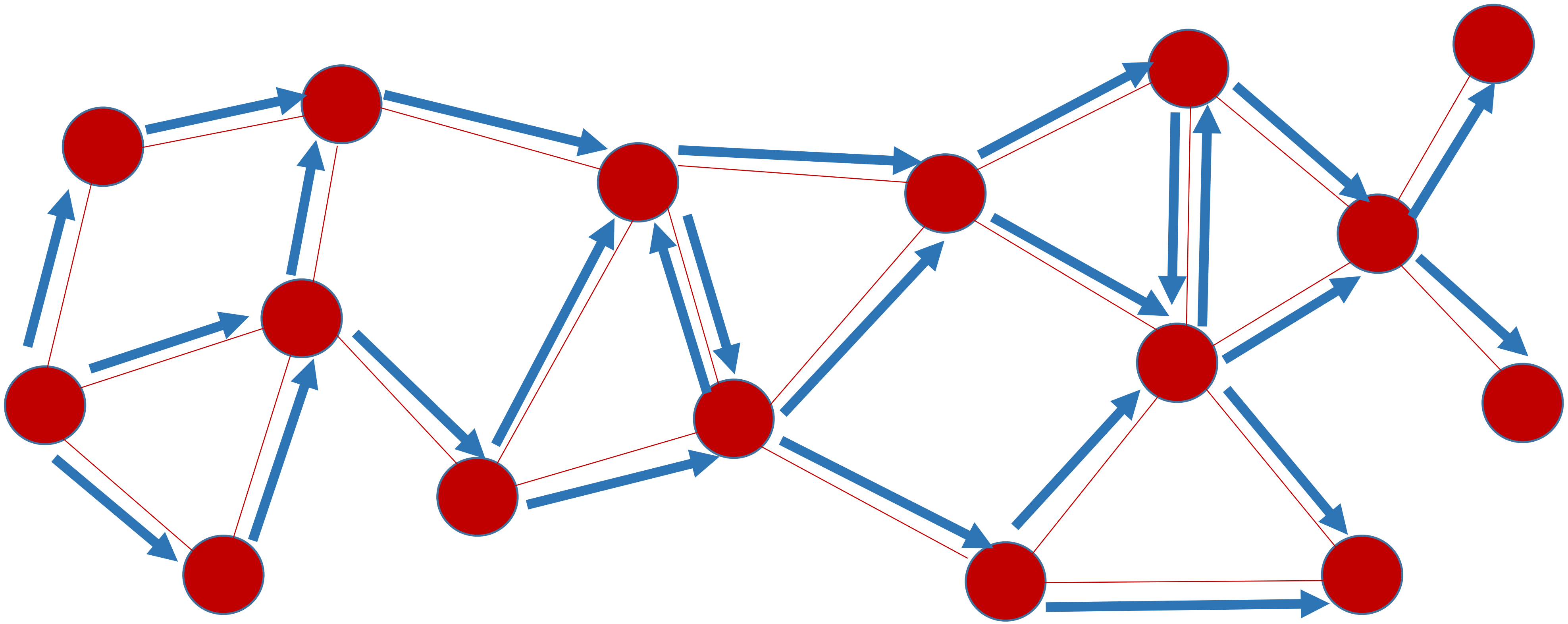
The Gossip Protocol



The Gossip Protocol



The Gossip Protocol



Communication Paradigm – The Gossip Protocol

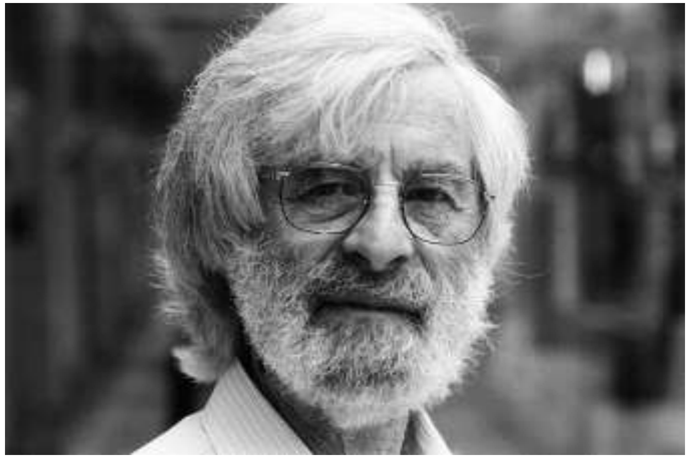
- In a distributed system, the typical model of communication is through message broadcast (an overlay broadcast).
  - How do you broadcast the message when all the nodes are not directly connected to each other over the overlay network?

- **The Gossip Protocol:**
  - Propagate messages based on the way epidemics spread
  - **Gossip**: Think of the office workers spreading rumors
  - More like a flooding, possibility of receiving duplicate information
  - Widely used in several distributed systems – distributed database consistency, distributed consensus protocols (state machine replication), Internet routing (think of the way routing messages are disseminated)

What is a Distributed System?

"*A distributed system is one in which the failure of a computer you didn't even know existed can render your own computer unusable*"

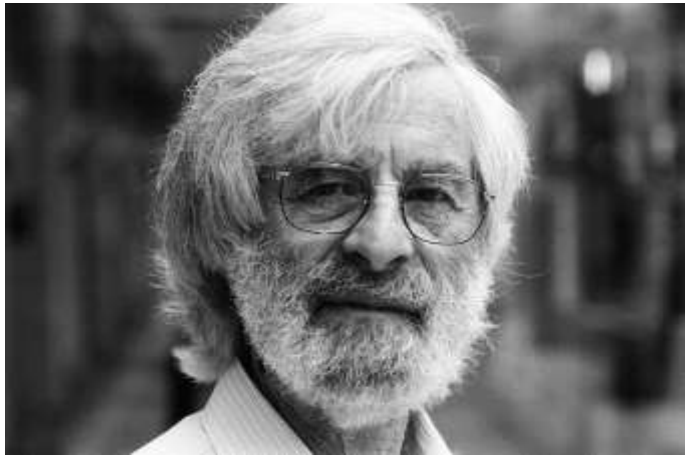
-- Leslie Lamport



What is a Distributed System?

*"A distributed system is one in which the **failure of a computer** you didn't even know existed can render your own computer unusable"*

-- Leslie Lamport

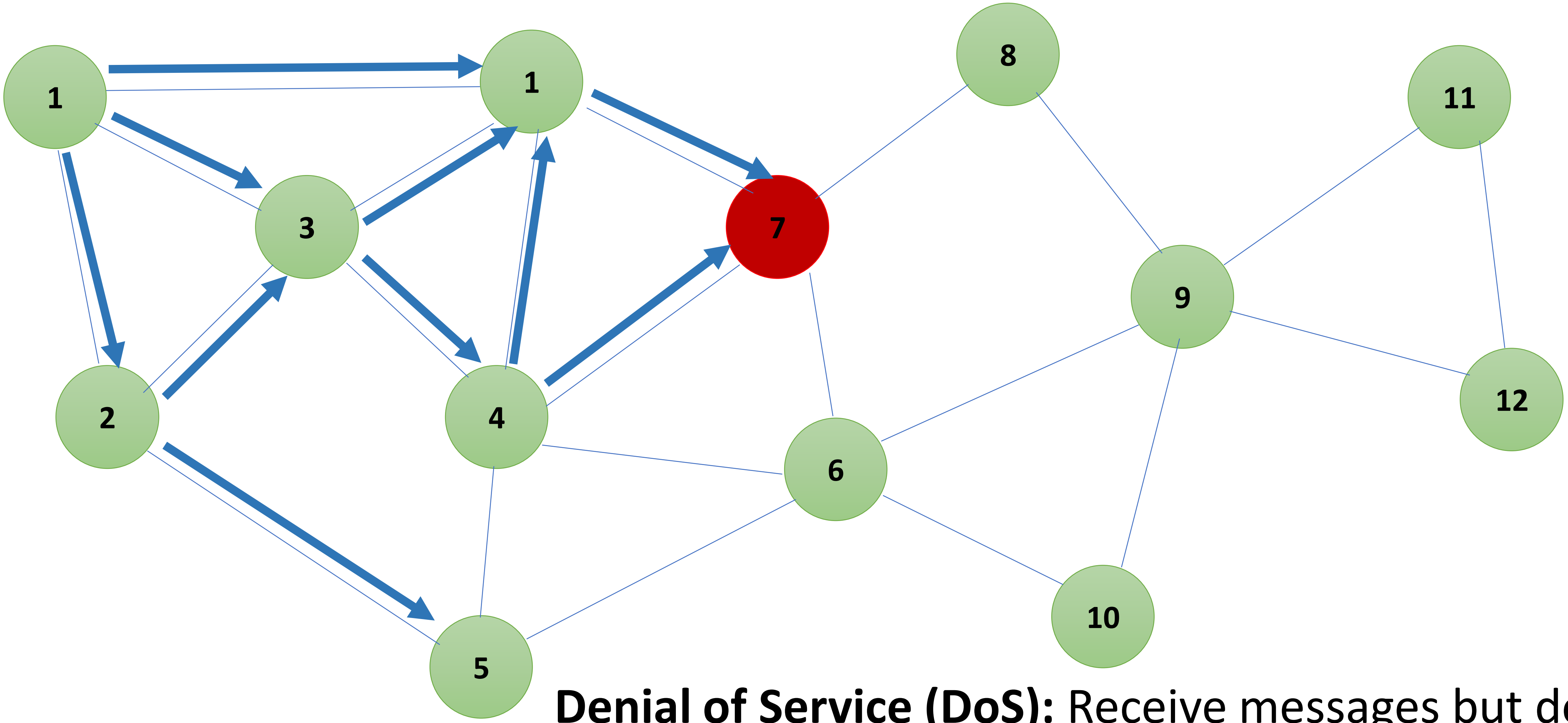


Type of Faults in a Distributed System

- **Crash Faults:**
  - A device stops responding suddenly
  - Hardware or software faults
  - May or may not recover later on (Crash-stop or Crash-recover)
  - (Network faults) A link fails instead of the entire node

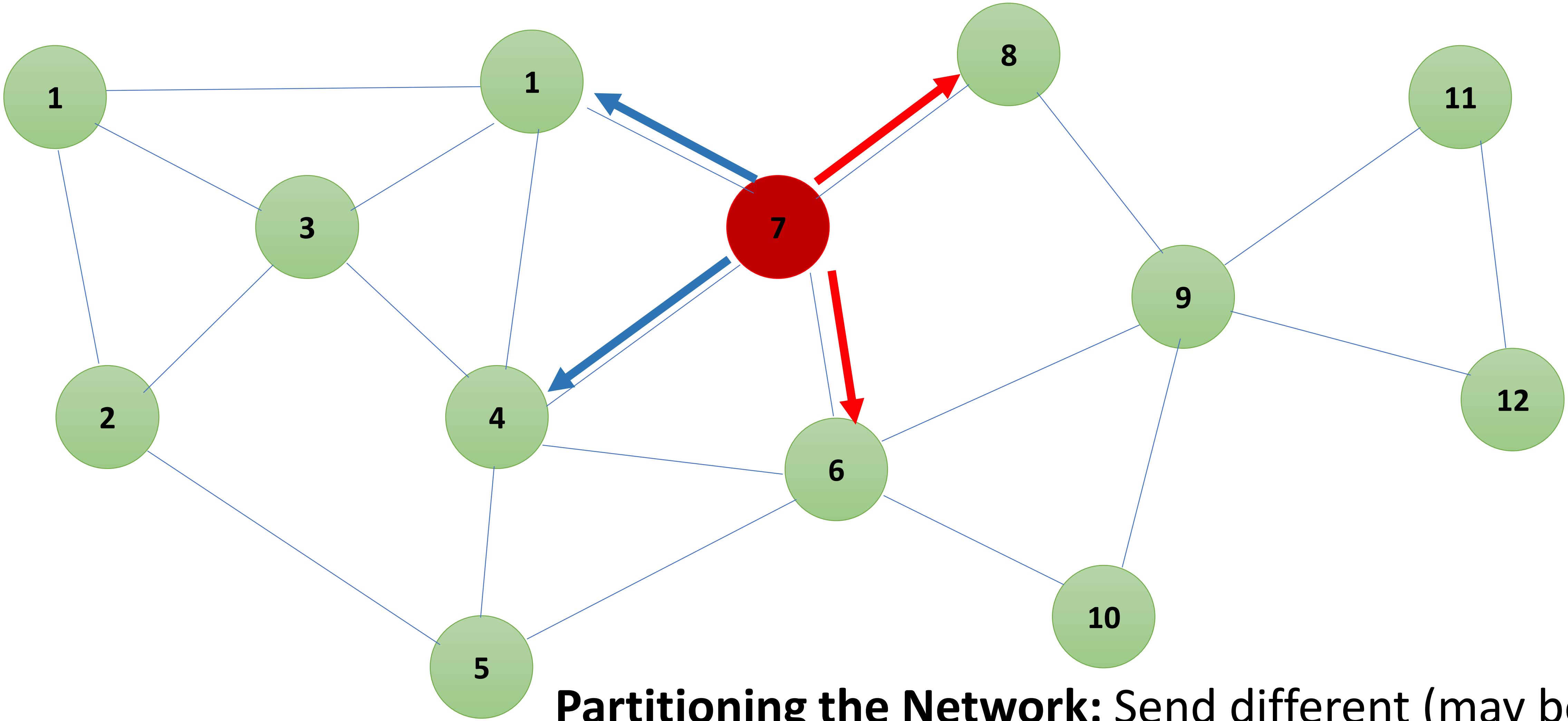
- **Byzantine Faults**
  - Nodes start misbehaving – forward different information to different peers
  - Internal or external attacks
  - Much more severe to handle compared to crash faults

Byzantine Faults



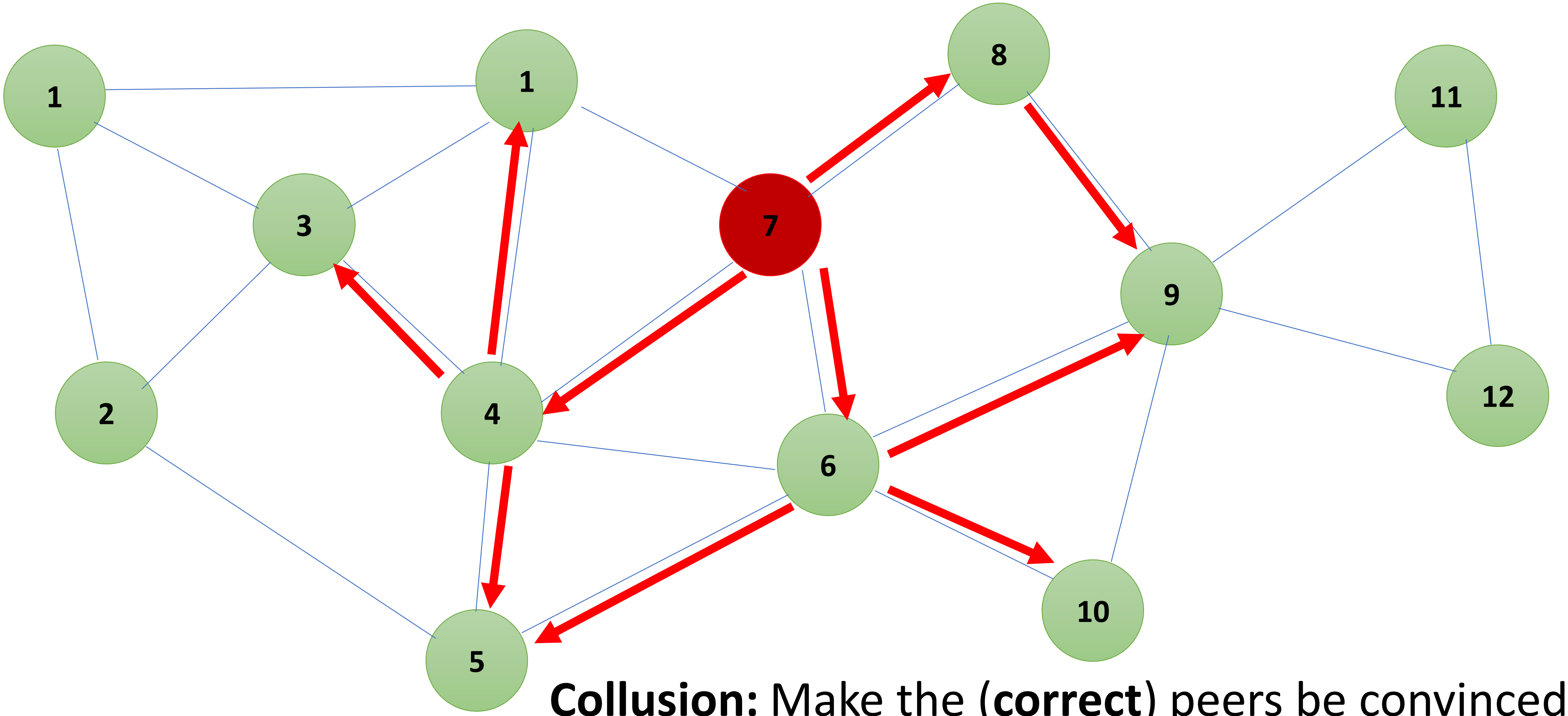
Denial of Service (DoS): Receive messages but do not propagate them further

Byzantine Faults



Partitioning the Network: Send different (may be conflicting) information to different peers

Byzantine Faults



Collusion: Make the (**correct**) peers be convinced on a misinformation, that they start spreading

Network Behavior

- **Reliable (Perfect) Links:**
  - A message is received if and only if it is sent
  - Reordering of messages is possible
- **Fair-loss Links:**
  - Message may be lost, reordered, or duplicated
  - A message is received eventually if retries are allowed
- **Arbitrary Links:**
  - Links under active adversary
  - A malicious adversary can interfere with the message (eavesdrop, modify, replay, spoof)

Network Behavior

- **Reliable (Perfect) Links:**
  - A message is received if and only if it is sent
  - Reordering of messages is possible
- **Fair-loss Links:**
  - Message may be lost, reordered, or duplicated
  - A message is received eventually if retries are allowed
- **Arbitrary Links:**
  - Links under active adversary
  - A malicious adversary can interfere with the message (eavesdrop, modify, replay, spoof)

SSL/TLS

Network Behavior

- **Reliable (Perfect) Links:**
  - A message is received if and only if it is sent
  - Reordering of messages is possible
- **Fair-loss Links:**
  - Message may be lost, reordered, or duplicated
  - A message is received eventually if retries are allowed
- **Arbitrary Links:**
  - Links under active adversary
  - A malicious adversary can interfere with the message (eavesdrop, modify, replay, spoof)

Retry (at network level), deDUP

SSL/TLS

Mode of Communication – Synchronous vs Asynchronous

- **The "Tom and Jerry" of distributed computing :**

- **Synchronous:**
  - **"Instantaneous"** message delivery, asymptotic upper bound on message delay
  - Sender is **blocked** until the message is delivered
  - There exists a global clock with a consistent clock rate
- **Asynchronous:**
  - No asymptotic bound on the message delay
  - Sender does not get blocked – messages are received eventually
  - No global clock or consistent clock rate

Impact of Failures based on the Mode of Communication

- **Failures over Synchronous Channels**
  - You can put a threshold on the message delay
  - You do not receive a message within the threshold ->
    You know that something has happened – may be a failure!

Impact of Failures based on the Mode of Communication

- **Failures over Synchronous Channels**
  - You can put a threshold on the message delay
  - You do not receive a message within the threshold -> You know that something has happened – may be a failure!

- **Failures over Asynchronous Channels**
  - There is no bound on the message delay!
  - You do not receive a message -> You do not know whether there is a failure, or the message is in transit ...

Synchronous and Asynchronous Nodes

- **Synchronous Nodes:**
  - Nodes execute algorithms at a known speed
  - Asymptotic upper bound on the execution time
- **Asynchronous Nodes:**
  - Nodes can pause algorithm execution arbitrarily
  - No timing guarantees on the execution of the code
- All combinations are possible
  - (synchronous/asynchronous) nodes over (synchronous/asynchronous) communication channels

Partially Synchronous

- **Complete Synchronous** and **Complete Asynchronous** are more of a theoretical concept
  - Practical networks have a predictable upper bound, the network occasionally deviates from that bound (congestion, retransmission of a lost packet, network or route reconfiguration)
  - Nodes typically execute code with a predictable speed with occasional deviations (such as OS scheduling problem, pause of garbage collection, thrashing, etc.)

Partially Synchronous

- **Complete Synchronous** and **Complete Asynchronous** are more of a theoretical concept
  - Practical networks have a predictable upper bound, the network occasionally deviates from that bound (congestion, retransmission of a lost packet, network or route reconfiguration)
  - Nodes typically execute code with a predictable speed with occasional deviations (such as OS scheduling problem, pause of garbage collection, thrashing, etc.)

- **Partially Synchronous**
  - The system is asynchronous for some finite (but unknown) periods of time, and synchronous for other times

Partially Synchronous

- **Complete Synchronous** and **Complete Asynchronous** are more of a theoretical concept
  - Practical networks have a predictable upper bound, the network occasionally deviates from that bound (congestion, retransmission of a lost packet, network or route reconfiguration)
  - Nodes typically execute code with a predictable speed with occasional deviations (such as OS scheduling problem, pause of garbage collection, thrashing, etc.)

- **Partially Synchronous**
  - The system is asynchronous for some finite (but unknown) periods of time, and synchronous for other times

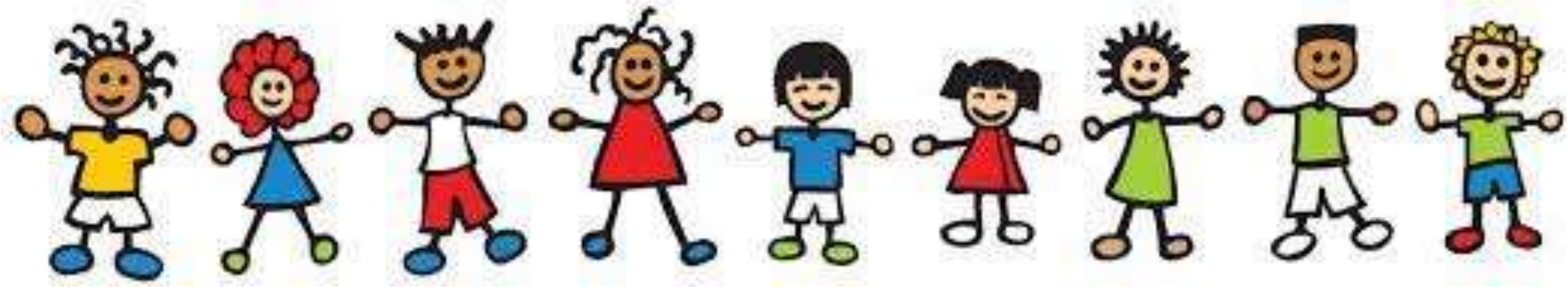
Other parts of Computer Science use the terminologies "Synchronous" and "Asynchronous" differently

System Model for Any Distributed Algorithm

- **Network**
  - Reliable
  - Fair-loss
  - Arbitrary
- **Timing**
  - Synchronous
  - Partially Synchronous
  - Asynchronous
- **Failures**
  - Crash-stop
  - Crash-recovery
  - Byzantine

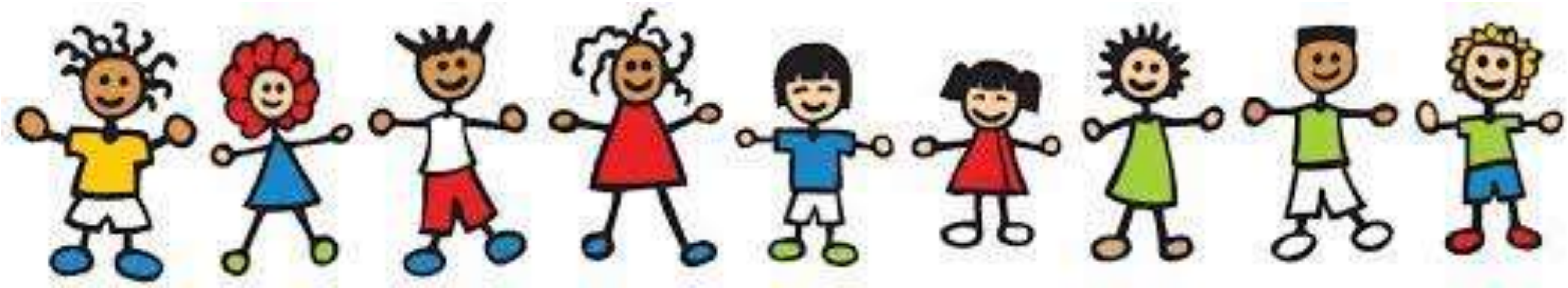
- **Pick one for each of the three parts**
- **All bets are off if your assumptions are wrong, so be careful!**

Muddy Children Puzzle



N number of children playing, k gets muddy

Muddy Children Puzzle

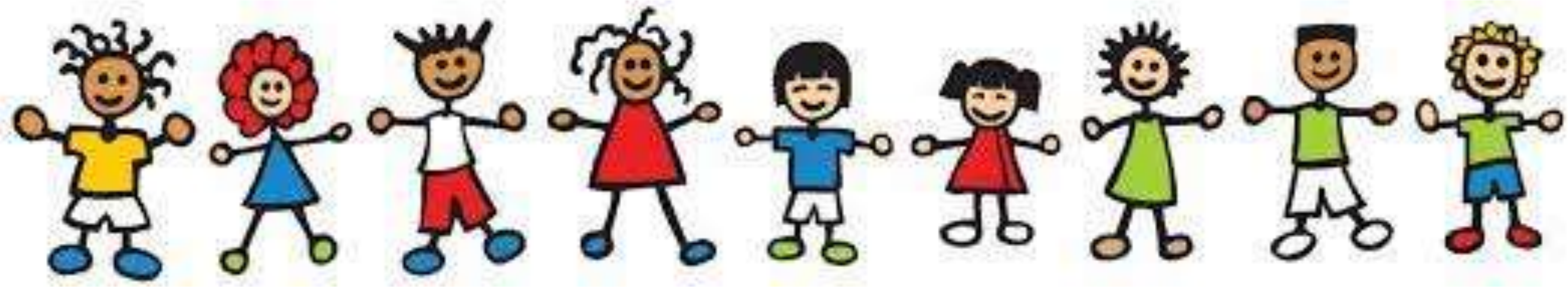


N number of children playing, k gets muddy

Each can observe all others, do not know their own states

- **Network:** Reliable (Children and intelligent and truthful)
- **Failures:** None
- **Timing:** Synchronous (They answer simultaneously based on the observations from the immediate past rounds)

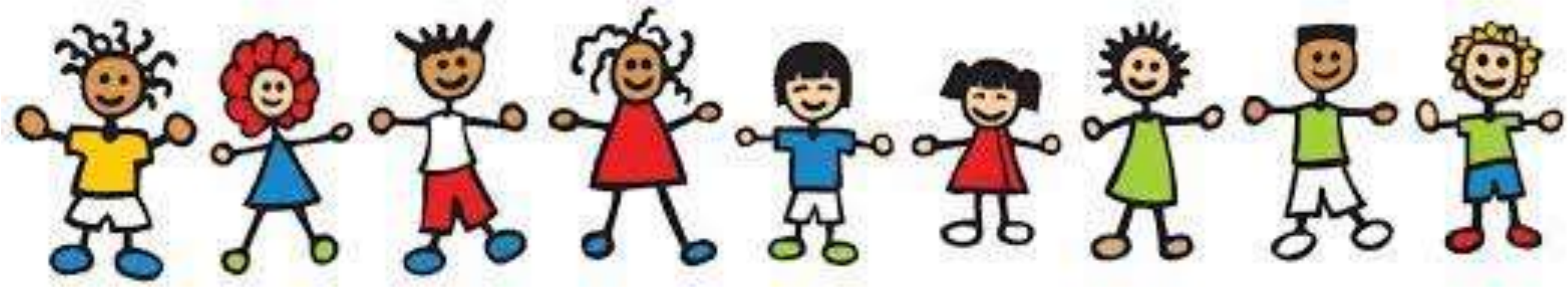
Muddy Children Puzzle



N number of children playing, k gets muddy

At least one of you is muddy. Do you know if you are muddy?

Muddy Children Puzzle



N number of children playing, k gets muddy

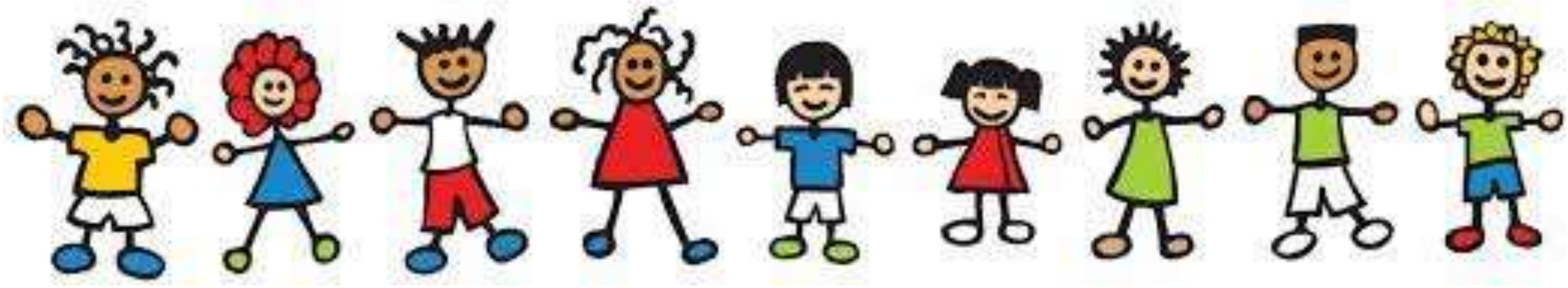
Information sources for the children

- Observing others (cannot observe himself or herself)
- Hearing what others say

Can infer based on previous rounds

- Only answers "**Yes**" (I know) / "No" (I do not know)

Muddy Children Puzzle

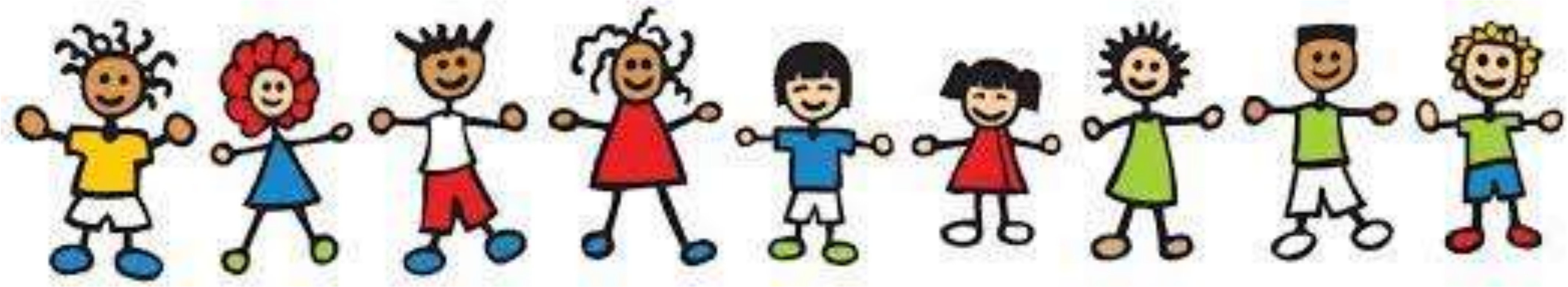


N number of children playing, k gets muddy

The protocol works at rounds. At every round, the children answer **"Yes"/"No" (Synchronous)**

How many rounds do you need to get the correct answers from all muddy children?

Muddy Children Puzzle



N number of children playing, k gets muddy

The protocol works at rounds. At every round, the children answer **"Yes"/"No" (Synchronous)**

How many rounds do you need to get the correct answers from all muddy children?

k

Muddy Children Puzzle, k = 1

N number of children playing, k gets muddy

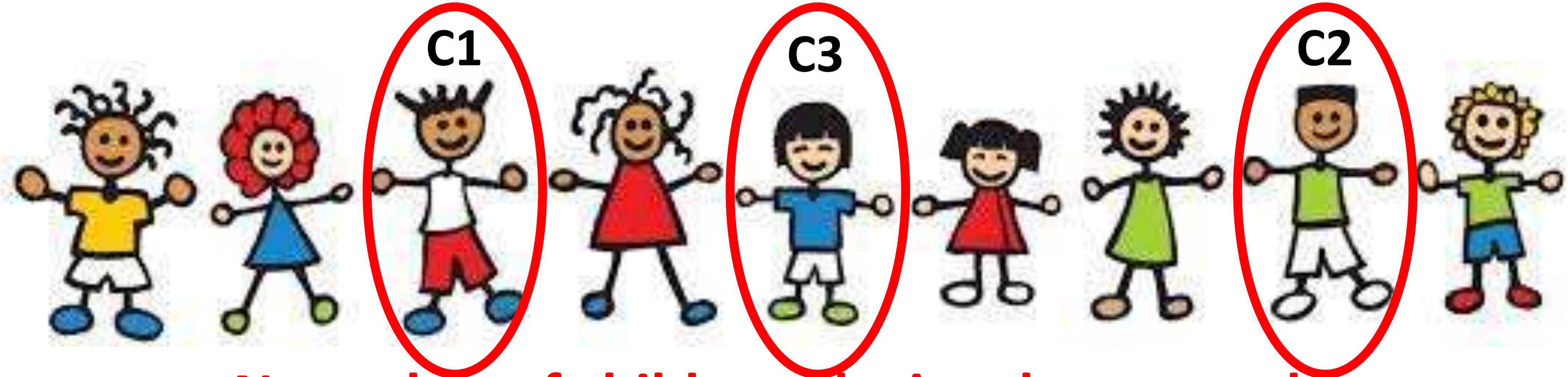
- Muddy child observes all others are clean
- Father said someone is muddy
- The child answers "Yes"
- Others hear the muddy child, confirms

Muddy Children Puzzle, k = 2

N number of children playing, k gets muddy

- <u>Round 1</u>: All says "No" as they are unsure as each muddy child sees one other muddy child (father said at least one)
- The two muddy children realizes that they are muddy, as others said no so they must have seen another muddy child
- <u>Round 2</u>: Muddy children answer "Yes", others also follow.

Muddy Children Puzzle, k = 3



N number of children playing, k gets muddy

- All says "No" up to Round 2.
- If C1 was clean, C2 and C3 must have answered "Yes" after Round 2. C1 understand that (s)he is not clean. So, in Round 3, (s)he replies with a "Yes" at Round 3. Same for C2 and C3.

Common Knowledge

N number of children playing, k gets muddy

- "At least one of you is muddy" - Do the children need this information for k > 1 ?

Common Knowledge

N number of children playing, k gets muddy

- **"At least one of you is muddy" - Do the children need this information for k > 1 ?**
    - Say, for k = 2, C1 observes C2, but does not know whether C2 observed C1, thus can understand k ≥ 1 (k = 1 or k = 2)

Do We Need the Common Knowledge?

N number of children playing, k gets muddy

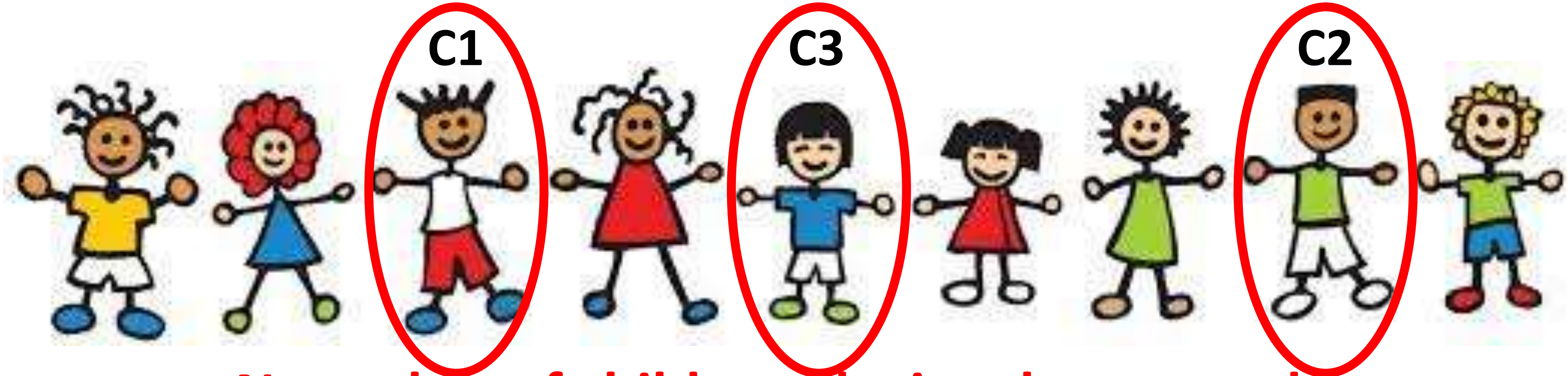
- **Common knowledge**: $k \geq 1$
- Without common knowledge
  - C1 can observe C2 muddy
  - But C2 can still say "No" even if (s)he observes C1 to be clean, as C2 does not know whether $k = 0$ or $k = 1$

Valid sequence with the common knowledge …

N number of children playing, k gets muddy

- **Common knowledge**: $k \geq 1$
- R1: Cannot decide whether k=1 or k=2; I see at least one other muddy, does not know my state **=>** all says "No" **=>** $k \neq 1$ as everyone (including the muddy child) sees at least one other muddy

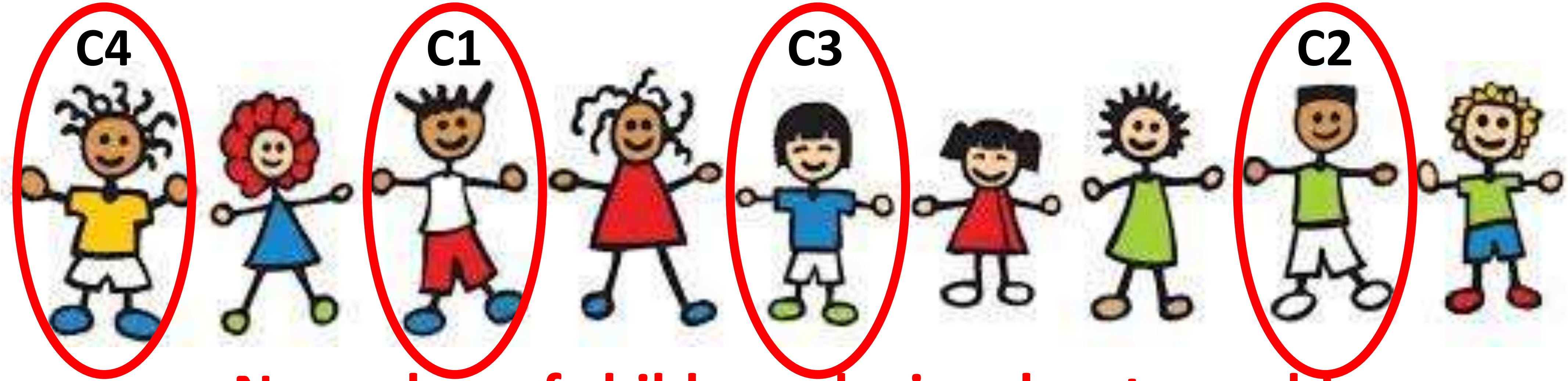
Valid sequence with the common knowledge ...



N number of children playing, k gets muddy

- R2: Cannot decide whether k=2 or k=3; I see at least two others muddy, does not know my state **=>** all says "No" **=>** k ≠ 2 as everyone (including the muddy children) sees at least two other muddy

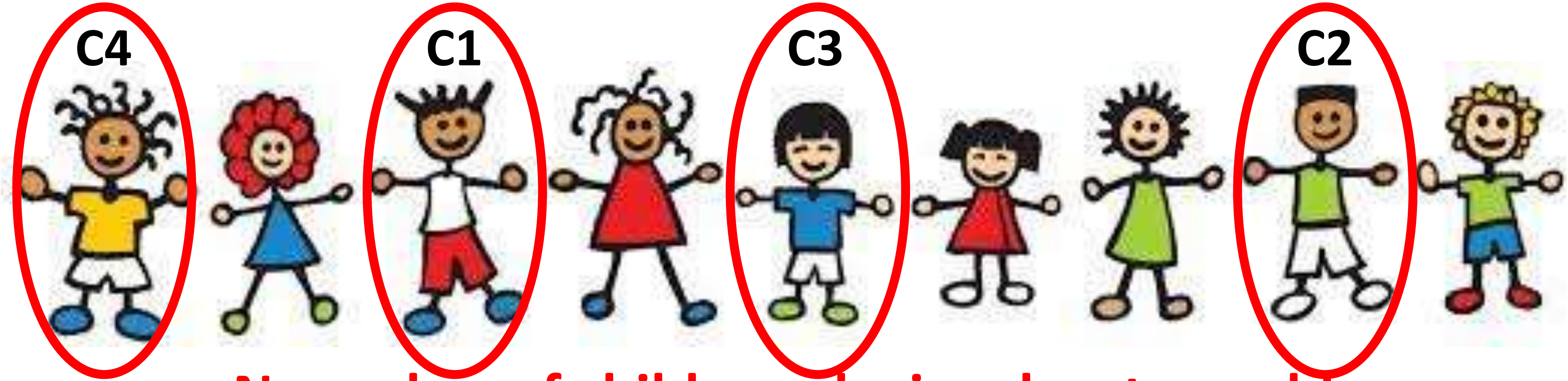
Valid sequence with the common knowledge ...



N number of children playing, k gets muddy

- R3: Cannot decide whether k=3 or k=4; I see at least three others muddy, does not know my state => all says "No" => k ≠ 3 as everyone (including the muddy children) sees at least two other muddy

Valid sequence with the common knowledge …



N number of children playing, k gets muddy

- The sequence progresses until there is a "Yes", i.e., some muddy children confirm themselves as muddy while observing k-1 other muddy children

Knowledge Hierarchy in Distributed System

- The knowledge of an agent (who runs the protocol) depends on
  - The starting / initial knowledge (may be the common knowledge)
  - The history of knowledge that it observed from the start

- **Common knowledge:** The "Publicly known" fact
  - Each can assume **others know it**
  - $k \geq 1$ is the common knowledge when the father mentions that "*at least one of you are muddy*"

Knowledge Hierarchy in Distributed System

- **Distributed knowledge:** The knowledge that is "distributed" among the members of the group, known by someone in the group
  - A node **cannot assume** that others know it
  - $k \geq 1$ is the distributed knowledge when the father does not mention anything, and the knowledge is purely based on the observation

- We'll see next how, starting from a common knowledge, an agent can uplift in the knowledge hierarchy with the help of distributed knowledge based on observations

- **Reference**: *Joseph Y. Halpern and Yoram Moses. 1990.* ***Knowledge and common knowledge in a distributed environment****. J. ACM 37, 3 (July 1990)*

Happy Learning