Distributed Systems

Gossip and Event-Based Coordination

WEEK 8: Gossip Protocols, Event-Based Coordination, Distributed Event Matching, Location Services, Epidemic Algorithms.

Online Lecture Series - 8

Felix Edesa

Addis Ababa Science and Technology University

AASTU

Gossip and Event-Based Coordination

Topics We Will Cover

- **Gossip Protocols** — Epidemic spreading and peer-sampling
- **Event-Based Coordination** — Asynchronous event handling
- **Distributed Event Matching** — Content-based structured filtering
- **Location Services** — GPS logical positioning for coordination
- **Routing Algorithms** — Flooding, cover-based, and merging approaches
- **Summary** — Principles, scalability, and fault tolerance

AASTU

Gossip Protocols: Introduction

Overview

- Gossip protocols are decentralized communication mechanisms.
- Nodes exchange information with randomly selected peers periodically.
- Provide scalable and fault-tolerant dissemination of events.
- Suitable for large distributed systems with dynamic membership.

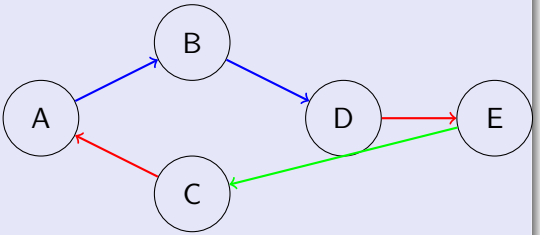
Industrial Example

- Amazon Dynamo uses gossip for:
  - Membership management
  - Failure detection
  - Metadata propagation
- Benefits: decentralization, resilience, scalability

AASTU

Gossip Protocol Mechanism: Visualization

Gossip Network Example



- Arrows represent message exchanges.
- Each node randomly selects peers per round.

Notes and Observations

- Nodes converge to the same state after multiple rounds.
- Gossip propagation is probabilistic but highly reliable.
- Push/Pull or Push-Pull variants determine message flow direction.
- Cassandra and DynamoDB are examples

AASTU

Epidemic Dissemination Model

Mathematical Model

- Probability a node receives an update after t rounds:

$$P(t) = 1 - (1 - p)^k$$

- p = probability of sending to a peer per round.
- k = number of contacts per round.
- $P(t)$ approaches 1 as t increases.

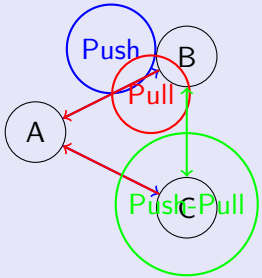
Industrial Insight

- Ensures high probability of delivery without central coordination.
- Used in distributed databases like Cassandra and DynamoDB.
- Robust against node failures and network partitions.

AASTU

Gossip Types: Push, Pull, Push-Pull

Diagram



- Arrows illustrate message direction.
- Colors indicate different gossip types.

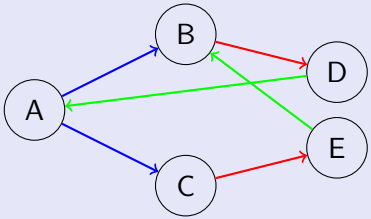
Notes and Industrial Examples

- **Push:** Sender actively sends updates to peers.
- **Pull:** Receiver requests updates from peers.
- **Push-Pull:** Combines push and pull for faster convergence.

AASTU

Node Interaction Diagram

Gossip Network Visualization



- Each arrow represents an exchange of updates between nodes.

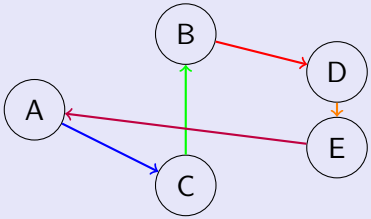
Notes and Explanation

- Nodes exchange updates with randomly selected neighbors.
- Gossip is probabilistic: each node doesn't need to communicate with all nodes.
- Over multiple rounds, all nodes eventually converge to the same state.

AASTU

Peer-Sampling in Gossip Protocols

Visualization



- Nodes periodically exchange neighbor lists.

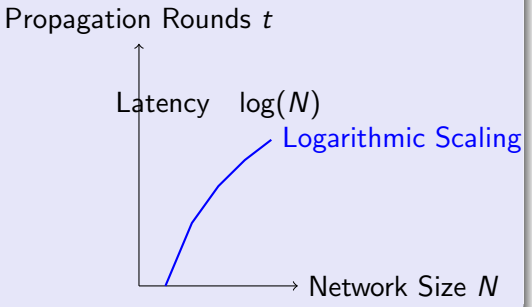
Concept and Industrial Example

- Maintains a **dynamic random view** of the network.
- Ensures **resilience and load balancing**.
- Avoids network hotspots and reduces latency.
- **Industrial Example:**
  - Cassandra uses peer-sampling for **efficient failure detection**.

AASTU

Scalability Considerations: Metrics

Network Growth vs Rounds



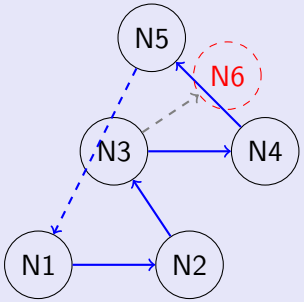
Observations and Industrial Example

- Gossip protocols scale efficiently with large N.
- **Latency grows logarithmically** instead of linearly.
- Probabilistic nature reduces message overhead.

AASTU

Fault Tolerance and Reliability

Gossip Network Fault Tolerance



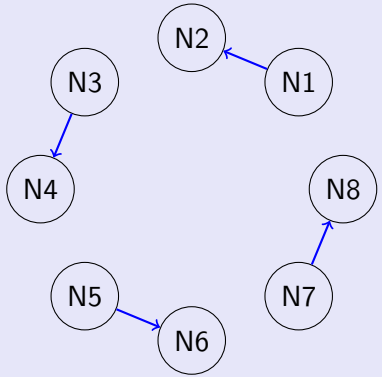
Key Observations

- Gossip protocols are **resilient to node failures**; updates find alternative paths.
- **Redundancy** ensures eventual delivery even if some nodes fail.
- Handles **network partitions**: delayed nodes synchronize when reconnected.
- **Example:** Riak and Cassandra

AASTU

Convergence Time: Basic Formula

Propagation Visualization



Key Formula

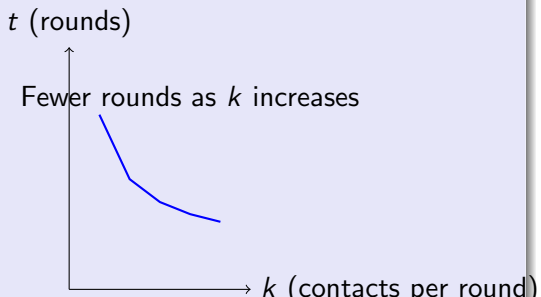
$$t \approx \frac{\ln n}{\ln(1+k)}$$

- n = number of nodes in the network
- k = number of contacts per node per round
- t = expected number of rounds for full coverage

AASTU

Convergence Time: Effect of Contacts

t vs k Graph



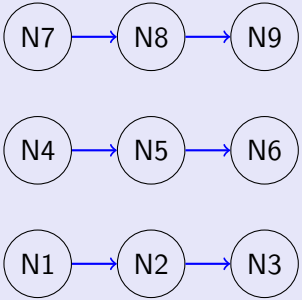
Observations

- Increasing k reduces t (faster convergence).
- Higher k increases bandwidth usage per round.
- **Trade-off:** latency vs network overhead.

AASTU

Convergence Time: Network Visualization

Example Network: 9 Nodes



Notes

- Smaller 3x3 network for clarity.
- Blue arrows show gossip propagation in **round 1**.
- Each node randomly selects peers for communication.
- Over a few rounds, all nodes converge to the same state.

AASTU

Convergence Time: Calculation

Expected Rounds Formula

$$t \approx \frac{\ln n}{\ln(1+k)}$$

- $n = 16$ nodes, $k = 2$ contacts per node per round
- $t \approx \frac{\ln 16}{\ln(1+2)} \approx 3$ rounds
- Demonstrates logarithmic scaling and rapid convergence

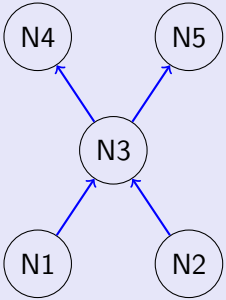
Industrial Insight

- Riak and Cassandra replicate cluster state using gossip.
- Full convergence occurs in just a few rounds.
- Engineers tune k to balance network overhead.

AASTU

Gossip Protocol Use Cases

Visual Example



Notes and Industrial Context

- **Cluster Membership:** Cassandra, Dynamo detect failed nodes dynamically.
- **Database Replication:** Riak Voldemort propagate updates to achieve eventual consistency.
- **IoT Event Dissemination**

AASTU

Gossip-Based Coordination: More Background

Concept

- Gossip protocols are used beyond simple updates.
- Enable aggregation, large-scale peer sampling, and overlay construction.
- Provide fault-tolerance, scalability, and decentralization.

Industrial Insight

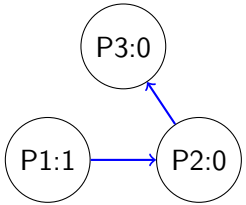
- Aggregation: Compute network-wide averages (e.g., monitoring metrics).
- Peer sampling: Maintain random network views for load balancing.
- Overlay construction: Build resilient communication layers.

AASTU

Aggregation Example

Mathematical Concept

- Each node P_i holds a value v_i.
- Upon contact with P_j: $v_i, v_j \leftarrow (v_i + v_j)/2$
- Repeated exchanges converge all nodes to the global average.



Notes

- Useful for network size estimation: if one node sets $v = 1$ and others 0, each node computes $1/N$ after convergence.
- Supports dynamic environments with nodes joining/leaving using epochs.

Randomized Aggregation for Initiator Election

Maximum Competition Method

- Each node P_i sets v_i randomly and keeps permanent m_i.
- On exchange: $v_i, v_j \leftarrow \max\{v_i, v_j\}$
- Nodes with $m_i < v_i$ lose the chance to initiate aggregation.

Industrial Insight

- Ensures a single node initiates network-wide operations.
- Optimistic approach: node assumes winner until proven otherwise.
- Practical in leader election for decentralized databases.

AASTU

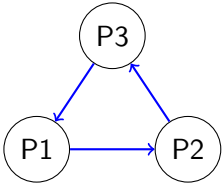
Peer-Sampling Service (PSS)

Mechanism

- Each node maintains a partial view of c neighbors.
- Nodes exchange entries to keep views updated.
- Randomized selection ensures uniform peer sampling without global knowledge.

Notes

- SelectPeer: pick a random neighbor.
- SelectToSend: choose entries to share.
- SelectToKeep: merge received entries, discard duplicates.



AASTU

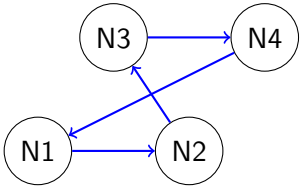
Overlay Construction with Gossip

Concept

- Use gossip to build resilient overlays on top of existing networks.
- Nodes maintain connections to a partial set of neighbors.
- Periodic exchanges optimize connectivity and reduce network diameter.

Notes

- Ensures robust communication paths even under node failures.
- Reduces overlay diameter for faster gossip propagation.
- Widely used in P2P systems, distributed databases, and content distribution networks.



AASTU

Event-Based Coordination: Introduction

What is Event-Based Coordination

- Components publish notifications when something interesting happens.
- Other components subscribe to events of interest (decoupled relationships).
- Coordination occurs via events rather than direct calls — enabling asynchronous, loosely-coupled systems. :contentReference[oaicite:0]index=0
- Supports reactive, scalable architectures (e.g., publish/subscribe, event buses).

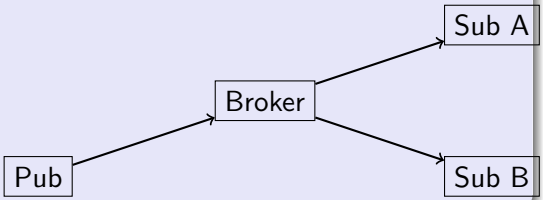
Why it matters

- Loose temporal referential coupling makes systems resilient to changes. :contentReference[oaicite:1]index=1
- Ideal for dynamic, distributed systems (IoT, microservices, event streams).

AASTU

Event-Based Model: Publish/Subscribe

Visual: Pub/Sub

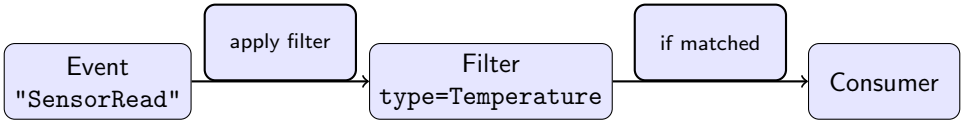


Key Principles

- Publisher doesn't need to know the subscribers — decoupling.
- Subscribers receive notifications of events they are interested in.
- Event broker or medium routes events based on topics, filters, or content.
- Enables scalable event propagation.

AASTU

Event Matching & Filtering — Diagram



The event passes through a filter that determines which consumer receives it.

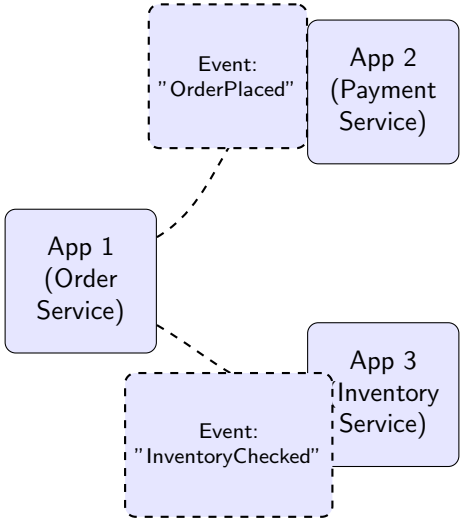
Event Matching & Filtering — Key Notes

Main Concepts

- Filters decide which consumers receive which events.
- Common filter types:
  - **Topic-based** – e.g. `sensor/temperature`
  - **Type-based** – e.g. `SensorReading`
  - **Content-based** – e.g. `value > 30`
- Prevents irrelevant data from flooding the system.
- Core in large-scale systems like IoT, finance, and monitoring networks.

AASTU

Coordination via Events — Use Case Diagram



Coordination via Events — Contextual Examples

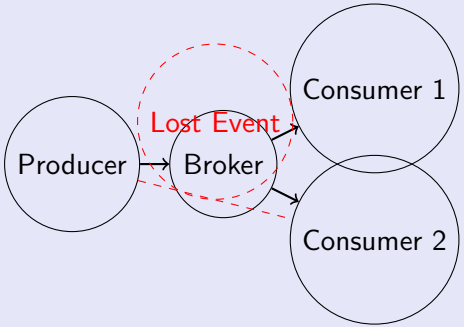
Real-World Scenarios

- **E-Commerce:** "OrderPlaced" event triggers:
  - Payment processing
  - Inventory deduction
  - Shipping updates
- **IoT Networks:** A "TemperatureExceeded" event triggers:
  - Alerts to monitoring dashboard
  - Automatic cooling system activation
  - Event logging for analytics

AASTU

Challenges in Event-Based Coordination

Challenges Visual

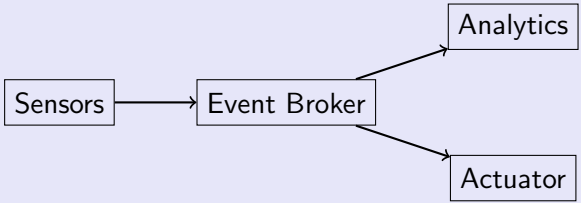


Major Practical Issues

- Event ordering causality — without care, consumers may see inconsistent sequences. :contentReference[oaicite:2]index=2
- Event loss, duplication and delivery semantics (at least once, exactly once).
- Scalability.
- Stateful coordination

Industrial Implementation Examples

Architecture Diagram



Real-World Insight

- Event-based coordination used in systems like Akka for distributed, asynchronous services. :contentReference[oaicite:4]index=4
- IoT platforms: sensors publish events, analytics subscribe.
- Financial trading platforms.

Distributed Event Matching — Overview

Core Idea

- Events are matched **dynamically** across multiple distributed brokers.
- Filtering rules are distributed to reduce central bottlenecks.
- Enables scalability for millions of concurrent subscriptions.

Why Content-based?

- Matches based on **event attributes** (not topics only).
- Example:

  Filter: $(\text{Type} = \text{"}\mathit{Temp}\text{"}) \wedge (\mathit{Value} > 30)$

- Consumers only receive relevant events.

AASTU

Event Structure and Filtering Model

Event Format

Each event E can be represented as:

$$E = \{(a_1, v_1), (a_2, v_2), \ldots, (a_n, v_n)\}$$

where a_i are attributes and v_i are their values.

A subscription S is defined as:

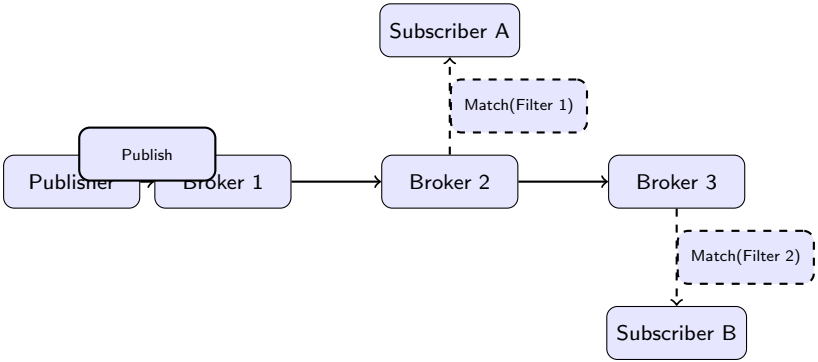
$$S = \{(a_i, op_i, c_i)\}$$

Matching Condition

$$match(E, S) = \begin{cases} 1, & \text{if } \forall (a_i, op_i, c_i) \in S, (a_i, \\ 0, & \text{otherwise} \end{cases}$$

Matching is boolean, but can be extended with confidence scores.

Distributed Matching Architecture



Brokers collaborate to evaluate filters and route only matching events.

Mathematical Analysis — Matching Cost (Part 1)

Complexity Model

Let:

- E = number of events
- S = number of subscriptions
- A = average attributes per event

The naive matching cost:

$$T_{\text{naive}} = O(E \times S \times A)$$

Distributed Optimization

Optimized distributed matching:

$$T_{\text{dist}} = O\left(\frac{E \times S}{k}\right)$$

where k = number of brokers performing parallel filtering.

Interpretation: The workload is divided across brokers.

Mathematical Analysis — Matching Cost (Part 2)

Analytical Comparison

Speedup ratio:

$$\eta = \frac{T_{\text{naive}}}{T_{\text{dist}}} = k$$

Scalability factor:

$$S_f = \frac{T_{\text{naive}} - T_{\text{dist}}}{T_{\text{naive}}} = 1 - \frac{1}{k}$$

Observations & Implications

Observation: When $k \to \infty$, $S_f \to 1$, meaning perfect scaling (ideal case).

Implication: Distributed event matching:

- Reduces latency linearly with broker count
- Improves throughput logarithmically with attribute complexity

AASTU

Content-based Routing Optimization

Routing Strategy

Use attribute-based indexes (e.g., R-trees, hash maps) to store filters.

$$\text{Route}(E) = \{S_i \mid \mathit{match}(E, S_i) = 1\}$$

Index efficiency:

$$C_{\mathit{match}} = O(\log S)$$

compared to linear $O(S)$ scanning.

Industrial Example

- **Apache Kafka Streams** — predicate-based filtering.
- **Amazon SNS** — message attributes for routing.
- **Esper CEP** — real-time event correlation.

Performance Metrics for Distributed Filtering

Quantitative Metrics

$$\text{Latency} = \frac{1}{E}\sum_{i=1}^{E}(t_{deliver,i} - t_{publish,i})$$

$$\text{Throughput} = \frac{E}{T_{obs}}$$

$$\text{MatchRate} = \frac{E_{matched}}{E_{total}}$$

Experimental Insights

- Latency grows logarithmically with broker hops.
- Increasing parallel brokers improves throughput.
- Filter selectivity strongly.

$$S_{eff} = 1 - \frac{E_{matched}}{E_{total}}$$

AASTU

Case Study: IoT Smart Grid — Filtering in Action

Scenario

- 10,000 IoT nodes publishing sensor data.
- Brokers filter based on:

$$(\text{type} = \text{"}\mathit{Temp}\text{"}) \wedge (\text{value} > 40) \wedge (\text{region} = \text{"}$$

- Only relevant alerts are sent to the control center.

Observed Metrics

$$T_{dist} = 1.3\,\text{s (for 10k events)}$$
$$\text{Throughput} = 7{,}600\,\text{events/s}$$
$$\text{Accuracy} = 98.7\%$$

Conclusion: Distributed content-based filtering scales efficiently with low latency.

AASTU

Location Services — Logical Positioning Concept

Core Idea

Location-based coordination uses logical positions rather than physical ones to manage event flow.

- Logical coordinates $\rightarrow$ Abstract representation of nodes.
- Events are routed based on logical proximity, not physical distance.
- Enables spatially-aware coordination in distributed systems.

Example

Scenario: In a fleet management system:

- Each vehicle publishes its GPS coordinates.
- A logical zone manager subscribes to events.

Ex: "Zone = (Lat 8.99, Long 38.78)"

AASTU

GPS Triangulation — Principle

Triangulation Principle

A GPS receiver estimates its position using signals from at least 3 satellites:

$$(x, y, z) = f(S_1, S_2, S_3)$$

- Each satellite signal provides a sphere of possible positions.
- Intersection of spheres determines receiver's logical position.
- Accuracy improves with more satellites.

Error Factor

$$\epsilon = \sqrt{(x - x_t)^2 + (y - y_t)^2 + (z - z_t)^2}$$

where (x_t, y_t, z_t) = true position of the receiver.

AASTU

Mathematical Model — Logical Zones

Zone Definition

Logical zones can be modeled as:

$$Z_i = \{(x, y) \in \mathbb{R}^2 \mid d((x, y), C_i) \leq r_i\}$$

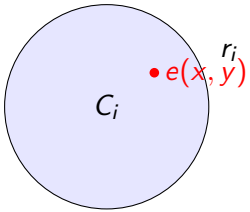
where:

- C_i — zone center, r_i — zone radius

Event Matching Rule

Event $e(x, y)$ belongs to Z_i if:

$$d((x, y), C_i) \leq r_i$$



Each zone corresponds to a coordination boundary.

Event Routing via Logical Zones

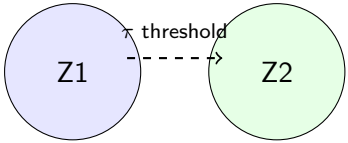
Routing Function

$$R(e) = \text{argmin}_{Z_i}\ d(e, C_i)$$

Each event is routed to the nearest logical zone.

Coordination Rule:

$$Z_i \to Z_j \text{ if } |C_i - C_j| < \tau$$



Zones communicate when within logical threshold τ.

Applications and Performance Metrics

Performance Metrics

$$L = \text{Average latency per event}$$

$$C = \text{Communication cost}$$

$$A_{acc} = \frac{\text{correct events}}{\text{total events}}$$

$$E_{gain} = \frac{1}{L \times C}$$

Goal: Maximize E_{gain} while maintaining $A_{acc} \geq 0.95$

Applications

- Vehicular ad-hoc networks (VANETs)
- Drone swarm coordination
- IoT-based logistics and delivery tracking
- Disaster response and rescue routing

AASTU

Routing Algorithms Overview

Overview of Routing Algorithms

Routing in distributed systems ensures events/messages reach target nodes efficiently.

- **Flooding:** Each node forwards messages to all neighbors.
- **Cover-based:** Messages are forwarded only to a subset of nodes (cover set).
- **Merging:** Messages from multiple sources are merged to reduce duplicates.

Notes

- Flooding is simple but can cause high redundancy.
- Cover-based routing improves scalability.
- Merging helps reduce network congestion.

Flooding Algorithm

Mechanism

- Node sends incoming message to all neighbors except the sender.
- Continues until all nodes are reached or TTL (Time-To-Live) expires.
- Ensures maximum coverage, but high message overhead.

Example

- Network of 5 nodes with degree 2.
- Each node forwards to 2 neighbors $\rightarrow$ exponential growth of messages.
- Good for small networks, poor for large-scale systems.

Formula for Message Count

For n nodes and average degree d:

$$M \approx d^t$$

where t = number of rounds.

Cover-based Routing

Mechanism

- Nodes maintain a cover set of neighbors.
- Reduces redundant message forwarding.

Coverage Probability

Let p_c = probability a node is in the cover set.
Then:

$$P(\text{message reaches node}) = 1 - (1 - p_c)^k$$

where k = number of neighbors contacted.

Industrial Example

- Large-scale IoT sensor networks.
- Reduces battery consumption and network congestion.
- Example: Wireless sensor networks forwarding environmental readings.

Merging-based Routing

Mechanism

- Nodes combine multiple incoming messages.
- Reduces network congestion and duplicates.
- Often used in content aggregation and distributed monitoring.

Message Reduction Formula

If m messages are merged:

$$M_{\text{merged}} = \frac{M_{\text{flooding}}}{m}$$

Example Use Case

- Sensor network measuring average temperature.
- Each node merges multiple readings $\rightarrow$ sends single message with combined info.
- Reduces network traffic significantly.

AASTU

Routing Algorithm Comparison

Comparison Table

Algorithm	Coverage	Message Overhead
Flooding	High	Very High
Cover-based	Medium	Medium
Merging	Medium-High	Low

Guidelines

- Small networks $\rightarrow$ Flooding acceptable.
- Large networks $\rightarrow$ Cover-based.
- Merging is ideal when content aggregation is needed.

Key Takeaways

- Trade-off between coverage, overhead, and scalability.
- Choice depends on network size and application requirements.
- Distributed coordination benefits from intelligent routing.

Chapter 8 Summary — Overview

Core Concepts

- **Gossip-based Coordination:** Nodes exchange information randomly with peers.
- **Event-based Coordination:** Nodes communicate via events, decoupling producers and consumers.
- **Decentralization:** Eliminates single points of failure; increases system robustness.

Key Takeaways

- Both methods enhance scalability and resilience.
- Gossip ensures eventual consistency and failure tolerance.
- Event-based coordination allows selective dissemination of information.

AASTU

Principles of Gossip Coordination

Gossip Principles

- Periodic, pairwise communication between nodes.
- Redundant paths ensure information reaches all nodes.
- Convergence is logarithmic in network size: $t \approx \frac{\ln n}{\ln(1+k)}$.

Fault Tolerance Reliability

- Naturally tolerates node failures.
- Handles network partitions; delayed nodes catch up.
- Industrial examples: Riak, Cassandra, Dynamo.

AASTU

Questions?

Thank you for your attention!

References

Suggested References

1. M. van Steen and A. S. Tanenbaum, *Distributed Systems*, 4th ed., Version 4.02, February 2024.
2. G. Coulouris, J. Dollimore, T. Kindberg, and G. Blair, *Distributed Systems: Concepts and Design*, 5th ed., Addison-Wesley, 2012.
3. N. Lynch, *Distributed Algorithms*, 2nd ed., Morgan Kaufmann, 1996.
4. R. E. Bryant and D. O'Hallaron, *Computer Systems: A Programmer's Perspective*, 4th ed., Pearson, 2024.

AASTU