Chương 7:

Câu 1:

Định nghĩa tính chịu lỗi trong hệ thống phân tán

Tính chịu lỗi (Fault Tolerance) là khả năng của hệ thống tiếp tục hoạt động bình thường hoặc trong phạm vi chấp nhận được ngay cả khi một số thành phần của nó bị lỗi. Mục tiêu là tránh gián đoạn dịch vụ và duy trì tính toàn vẹn dữ liệu dù có lỗi xảy ra.

Ví dụ: Nếu một nút trong hệ thống phân tán bị sập, các nút khác có thể tiếp tục xử lý công việc và đảm bảo người dùng không bị gián đoạn dịch vụ.

Bốn yêu cầu cơ bản của một hệ thống chịu lỗi

1. Phát hiện lỗi (Fault Detection)
   Hệ thống phải có khả năng phát hiện khi có thành phần hoạt động bất thường hoặc ngừng hoạt động.
2. Ngăn chặn lan truyền lỗi (Fault Containment)
   Khi lỗi xảy ra, cần giới hạn ảnh hưởng để không lan sang các phần khác của hệ thống.
3. Khôi phục lỗi (Fault Recovery)
   Khả năng khôi phục hệ thống về trạng thái ổn định (ví dụ: thông qua rollback, backup hoặc restart).
4. Dự phòng lỗi (Redundancy)
   Dự phòng tài nguyên (nút, dữ liệu, kênh truyền...) để thay thế khi một thành phần bị lỗi.

Phân biệt tính sẵn sàng và tính đáng tin cậy

Tiêu chí	Tính sẵn sàng (Availability)	Tính đáng tin cậy (Reliability)
Định nghĩa	Khả năng hệ thống phản hồi và cung cấp dịch vụ liên tục	Khả năng hệ thống duy trì hoạt động mà không xảy ra lỗi
Thời điểm đo lường	Tức thời hoặc thời gian thực	Trong một khoảng thời gian dài
Mức độ chấp nhận lỗi	Có thể chấp nhận một số lỗi nhỏ nhưng phục hồi nhanh	Không được lỗi trong thời gian xác định

Ví dụ minh họa:

- Tính sẵn sàng: Một website thương mại điện tử có hệ thống failover. Khi server chính gặp sự cố, server phụ thay thế trong vài giây. Người dùng chỉ bị chậm một chút, hệ thống có tính sẵn sàng cao.
- Tính đáng tin cậy: Hệ thống điều khiển động cơ máy bay phải hoạt động liên tục và chính xác trong suốt chuyến bay, yêu cầu độ đáng tin cậy cực cao, không được lỗi.

Ví dụ lỗi mạng và cách khắc phục bằng dư thừa

Kịch bản lỗi:
Giả sử trong một hệ thống truyền thông giữa các máy chủ, gói tin bị mất trong quá trình truyền (do nhiễu, lỗi mạng), hoặc bị tràn bộ đệm dẫn đến mất dữ liệu.

Cách khắc phục bằng phương pháp dư thừa:

a. Dư thừa dữ liệu:
Gửi nhiều bản sao của cùng một gói tin qua các kênh khác nhau. Nếu một bản bị mất, bản còn lại vẫn đến được, không gián đoạn dịch vụ.

b. Dư thừa thời gian:
Sử dụng cơ chế xác nhận (ACK). Nếu không nhận được phản hồi sau một thời gian nhất định, hệ thống sẽ gửi lại gói tin.

c. Dư thừa phần cứng:
Cấu hình nhiều máy chủ chạy song song. Nếu một máy chủ không phản hồi, bộ cân bằng tải sẽ chuyển yêu cầu sang máy chủ khác.

Ví dụ cụ thể: Hệ thống email lưu bản nháp tạm thời. Nếu mất kết nối khi gửi, khi mạng khôi phục, hệ thống sẽ tự động gửi lại mà không yêu cầu người dùng thao tác lại.

Phân loại và phân tích ba loại lỗi trong hệ thống phân tán

Loại lỗi	Đặc điểm	Nguyên nhân gốc rễ	Tác động đến hệ thống phân tán
Nhất thời (Transient)	Xuất hiện trong thời gian ngắn rồi tự biến mất	Nhiễu mạng, tắc nghẽn gói tin, lỗi bộ nhớ tạm thời	Tạm thời gián đoạn kết nối hoặc sai lệch dữ liệu nhỏ
Chập chờn (Intermittent)	Xảy ra không đều, lặp lại không đoán trước được	Lỗi phần cứng không ổn định, tiếp xúc không tốt	Khó tái hiện, gây mất ổn định, làm tăng độ trễ hoặc mất đồng bộ
Vĩnh cửu (Permanent)	Không tự phục hồi, tồn tại đến khi có can thiệp	Hỏng phần cứng, lỗi logic phần mềm, mất điện	Mất chức năng hoàn toàn, ảnh hưởng trực

			tiếp đến độ tin cậy và tính sẵn sàng

Đánh giá các mô hình thất bại trong hệ thống phân tán

a. Crash failure (Sụp đổ)

- Ưu điểm:
  - Dễ phát hiện (qua timeout hoặc heartbeat).
  - Có thể khôi phục bằng dự phòng (failover).
- Nhược điểm:
  - Nếu không có cơ chế phục hồi, mất dịch vụ.
  - Mất trạng thái nếu không lưu trữ liên tục.

b. Omission failure (Bỏ sót)

- Ưu điểm:
  - Phát hiện tương đối đơn giản (qua retry, timeout).
- Nhược điểm:
  - Có thể gây mất dữ liệu hoặc mất thông điệp.
  - Dễ bị nhầm với lỗi mạng tạm thời.

c. Timing failure (Thất bại về thời gian)

- Ưu điểm:
  - Xảy ra khi hệ thống real-time yêu cầu độ chính xác cao, giúp đánh giá khả năng đáp ứng.
- Nhược điểm:
  - Khó phân biệt với chậm trễ mạng.
  - Phát hiện và phục hồi cần đồng hồ chính xác và đồng bộ tốt.

d. Byzantine failure

- Ưu điểm:
  - Mô hình tổng quát và mạnh nhất, bao quát nhiều loại lỗi khác.
- Nhược điểm:
  - Rất khó phát hiện và xử lý.
  - Cần giải pháp phức tạp, tiêu tốn tài nguyên (ví dụ: BFT, consensus protocols như PBFT, Raft).

Chiến lược chịu lỗi tổng thể cho hệ thống phân tán phức tạp

Mục tiêu:

- Tối ưu cả: Tính sẵn sàng, Đáng tin cậy, An toàn, Dễ bảo trì

a. Phát hiện lỗi

Loại lỗi	Cách phát hiện
Crash	Heartbeat, timeout
Omission	ACK timeout, kiểm tra toàn vẹn message
Timing	Giám sát thời gian phản hồi, đồng hồ logic
Byzantine	So sánh kết quả từ nhiều nút (voting)

b. Ngăn ngừa lỗi

- Dự phòng phần cứng và phần mềm: sử dụng hệ thống đa nút (replication).
- Giới hạn quyền truy cập và kiểm tra đầu vào: ngăn chặn lỗi logic hoặc khai thác bảo mật.
- Kiểm thử đều đặn: kiểm tra hồi quy, stress test, chaos testing.

c. Phục hồi lỗi

Loại lỗi	Cách phục hồi
Crash/Omission	Sử dụng checkpoint và restart từ trạng thái gần nhất
Timing	Cân bằng tải, điều chỉnh timeout linh hoạt
Byzantine	Áp dụng thuật toán đồng thuận (PBFT, Raft)

d. Công nghệ gợi ý triển khai

- Replication: Dữ liệu được sao lưu theo kiểu chủ-tớ (master-slave) hoặc đồng đẳng (peer-to-peer).
- Load balancing + Health check: tự động chuyển hướng sang nút hoạt động.
- Distributed Logging: để theo dõi và phân tích nguyên nhân lỗi.
- Consensus protocols: để đảm bảo tính đúng đắn dù có lỗi Byzantine.
- Chaos Engineering: để chủ động gây lỗi và kiểm tra khả năng chịu lỗi của hệ thống.

Câu 2:

Ba loại dư thừa trong che giấu lỗi

Loại dư thừa	Giải thích
Dư thừa thông tin	Lưu trữ thêm thông tin để kiểm tra, phát hiện, hoặc khôi phục lỗi
Dư thừa thời gian	Lặp lại thao tác để tăng khả năng thành công
Dư thừa vật lý	Bổ sung phần cứng độc lập để thay thế khi có hỏng hóc

Tiến trình bền bỉ (Persistent Process)

Khái niệm:
Tiến trình bền bỉ là tiến trình có thể khôi phục lại trạng thái trước khi gặp lỗi hoặc được thay thế bằng tiến trình khác giữ nguyên chức năng. Mục tiêu là duy trì liên tục dịch vụ dù có sự cố phần mềm hoặc phần cứng.

Cơ chế hoạt động khi tiến trình chính bị lỗi:

- Một hoặc nhiều tiến trình phụ (backup) được chạy song song hoặc theo dõi tiến trình chính.
- Khi tiến trình chính bị lỗi (mất phản hồi, crash), tiến trình phụ phát hiện lỗi qua timeout/"alive signal", sau đó:
    - Kế thừa trạng thái từ tiến trình chính nếu được checkpoint định kỳ.
    - Tiếp quản vai trò xử lý, đảm bảo dịch vụ không bị gián đoạn.

Đề xuất cho hệ thống dịch vụ trực tuyến lưu dữ liệu quan trọng

Kết hợp dư thừa thông tin và vật lý để che giấu lỗi:

- Dư thừa thông tin:
    - Sử dụng CRC + mã hóa checksum để phát hiện lỗi trong truyền và lưu trữ dữ liệu.
    - Nhân bản dữ liệu (replication) theo mô hình leader-follower (ví dụ: Master-Slave PostgreSQL hoặc Replica Set MongoDB).
    - Kết hợp với snapshot + logging (WAL) để có thể rollback hoặc phục hồi.
- Dư thừa vật lý:
    - Cụm máy chủ dự phòng (active-passive hoặc active-active): khi một server chính gặp sự cố, server phụ tự động tiếp quản.
    - Sử dụng các vùng khả dụng (Availability Zones) trong dịch vụ đám mây để chống mất điện/toàn vùng.

Tổng thể: Khi một node gặp lỗi, hệ thống vẫn truy cập được dữ liệu nhờ replication và tự động failover, đồng thời kiểm tra CRC đảm bảo dữ liệu không bị hỏng.

So sánh kiến trúc nhóm tiến trình: ngang hàng và phân cấp

Tiêu chí	Ngang hàng (Peer-to-Peer)	Phân cấp (Hierarchical)
Độ trễ	Cao hơn (do phải đồng bộ giữa nhiều nút)	Thấp hơn (quản lý tập trung)
Quản lý	Phức tạp (mỗi nút ngang hàng nhau)	Dễ quản lý hơn, nhưng phụ thuộc vào nút chính
Khả năng chịu lỗi	Tốt hơn nếu có nhiều tiến trình backup (không có điểm lỗi đơn)	Dễ bị ảnh hưởng nếu leader sập (cần cơ chế bầu chọn leader mới)
Ứng dụng thực tế	BitTorrent, blockchain	Hệ thống Master-Slave, Zookeeper

Kết luận: Ngang hàng phù hợp với hệ thống yêu cầu tính sẵn sàng cao và không phụ thuộc trung tâm; phân cấp phù hợp với hệ thống cần kiểm soát chặt và độ trễ thấp.

Thiết kế chiến lược chịu lỗi tổng thể cho hệ thống thanh toán phân tán

a. Phát hiện lỗi:

- Timeout heartbeat + "IS ALIVE": tiến trình gửi định kỳ tín hiệu sống (heartbeat) cho các node khác.
- Nếu không nhận được phản hồi trong một thời gian định trước → xác định là lỗi.

b. Dư thừa đa tầng:

- Thông tin: checksum, replication, WAL (Write Ahead Log).
- Thời gian: thực hiện lại (retry) nếu timeout, song giới hạn số lần thử để tránh gây lỗi logic.
- Vật lý: cụm máy chủ đa vùng, cân bằng tải, failover DNS.

c. Giao thức truyền thông tin cậy:

- Point-to-point: TCP hoặc UDP + ACK/timeout.
- Nhóm (group communication): sử dụng multicast kèm giao thức đảm bảo thứ tự (Total Order Broadcast).

d. Cơ chế xử lý thất bại RPC:

Cơ chế	Đặc điểm	Áp dụng khi
At most once	Tránh lặp (idempotent); đảm bảo không thực hiện thừa	Thanh toán hoặc trừ tiền
At least once	Chấp nhận lặp nhưng đảm bảo kết quả được thực hiện	Gửi cảnh báo, logging
Exactly once	Cần đảm bảo không mất, không lặp	Rất tốn kém, cần kết hợp xác nhận và lưu trạng thái

e. Phối hợp tổng thể:

1. Tầng giao tiếp:
    - RPC dùng at most once, kiểm tra timeout và gắn mã định danh yêu cầu để phát hiện lặp.
2. Tầng phát hiện lỗi:
    - Heartbeat giữa các node, khi mất kết nối thì kích hoạt failover hoặc khôi phục.
3. Tầng lưu trữ:
    - Dùng replication kèm checkpoint + logging.
    - Kết hợp với cơ chế nhất quán eventual hoặc quorum consensus (ví dụ: Paxos, Raft).
4. Tầng điều phối:
    - Sử dụng leader election (Zookeeper hoặc Raft).
    - Khi leader chết, node khác tự động thay thế.

Câu 3:

Định nghĩa cam kết nguyên tử và ba kiểu giao thức

Cam kết nguyên tử (Atomic Commit):

Là đảm bảo tính toàn vẹn của giao dịch phân tán: tất cả các node (participant) trong giao dịch hoặc cùng commit hoặc cùng rollback – không có trạng thái trung gian.

Mục tiêu: đảm bảo tính nguyên tử (Atomicity) trong mô hình ACID của giao dịch.

Ba kiểu giao thức cam kết:

1. One-Phase Commit (1PC) – Cam kết một pha
2. Two-Phase Commit (2PC) – Cam kết hai pha
3. Three-Phase Commit (3PC) – Cam kết ba pha

Cơ chế hoạt động của giao thức cam kết hai pha (2PC)

Thành phần:

- Coordinator (Điều phối viên): điều hành quá trình giao dịch.
- Participants (Người tham gia): thực thi phần giao dịch.

Pha 1 – Biểu quyết (Voting phase):

1. Coordinator gửi PREPARE đến tất cả participants.
2. Mỗi participant thực hiện kiểm tra (validate, log lại) → phản hồi:
    - YES: nếu sẵn sàng commit.
    - NO: nếu không thể thực hiện.

Pha 2 – Quyết định (Commit phase):

- Nếu tất cả đều YES, coordinator gửi COMMIT → participants commit thật.
- Nếu có ít nhất 1 NO, coordinator gửi ABORT → tất cả rollback.

Phục hồi khi participant không phản hồi:

- Coordinator timeout → quyết định ABORT toàn bộ.
- Participant restart sẽ kiểm tra nhật ký để xác định:
    - Nếu đã ghi "YES", sẽ chờ coordinator gửi quyết định.
    - Nếu không có gì, rollback.

Ứng dụng 2PC/3PC trong hệ thống đặt vé máy bay phân tán

Bối cảnh:

- Các bước độc lập: Đặt chỗ (Reservation), Thanh toán (Payment), Phát vé (Ticketing).
- Mỗi bước thực hiện bởi một service riêng biệt.

Chọn giao thức: 3PC (Three-Phase Commit)

Lý do:

- Hệ thống đặt vé có yêu cầu cao về không phong tỏa, do xử lý người dùng lớn.
- 2PC có nguy cơ blocking nếu coordinator sập, khiến vé bị "treo".
- 3PC tránh blocking bằng cách chia nhỏ pha 1 thành:
    - Can Commit?
    - Pre-Commit
    - Commit/Abort

Xử lý khi coordinator lỗi:

- Trong 3PC, các participant khi ở trạng thái "Pre-Commit" mà không thấy phản hồi sau timeout sẽ tự quyết định commit/abort dựa vào trạng thái nhóm → giảm blocking.
- Nếu có logging, participant có thể phục hồi trạng thái trước và tiếp tục.

So sánh ưu – nhược của 2PC vs 3PC

Tiêu chí	2PC	3PC
Phong tỏa (Blocking)	Có thể blocking nếu coordinator chết sau khi nhận “YES”	Không blocking (participants có thể tiến hành commit/abort độc lập sau timeout)
Hiệu năng	Tốt hơn (ít message: ~3N)	Tốn hơn (thêm 1 round: ~5N)
Logging	Đơn giản: mỗi participant ghi lại “YES”	Phức tạp hơn: cần ghi cả trạng thái trung gian (Pre-Commit)
Phục hồi	Dễ gây không rõ trạng thái nếu coordinator chết	Phục hồi an toàn hơn nhờ trạng thái tường minh
Tính nhất quán	Mạnh	Mạnh
Tính sẵn sàng	Thấp nếu xảy ra lỗi	Cao hơn

Thiết kế giải pháp cam kết phân tán cho hệ thống ngân hàng trực tuyến

Yêu cầu:

- Tính sẵn sàng cao (liveness).
- Không phong tỏa.
- Không mất tiền hoặc thực hiện sai giao dịch.

(1) Giao thức: Kết hợp 2PC + pha tăng cường (Enhanced 2PC)

- Sử dụng 2PC làm nền, nhưng:
  - Dùng consensus (Paxos hoặc Raft) để bầu lại coordinator khi lỗi.
  - Có thể mở rộng thêm 1 pha xác nhận giống 3PC nếu hệ thống cần không blocking tuyệt đối.

(2) Cơ chế Logging và phục hồi:

Participant:

- Ghi vào write-ahead log (WAL) trạng thái:
  - "INIT" → "VOTE YES" → "PRE-COMMIT" → "COMMIT".
- Sau crash, đọc log để xác định:
  - Nếu "VOTE YES" chưa có kết quả → chờ.
  - Nếu "COMMIT" rồi → hoàn tất commit lại.

Coordinator:

- Ghi nhật ký:
  - "PREPARE_SENT"
  - "ALL_YES → COMMIT"
  - "ABORT_REASON"

Replication log theo quorum (ví dụ: ZooKeeper, etcd) để bảo vệ log khi coordinator chết.

(3) Cân bằng giữa atomicity, liveness và throughput:

Khía cạnh	Giải pháp
Tính nhất quán (Atomicity)	Giao thức 2PC + log mạnh
Tính sẵn sàng (Liveness)	Không blocking (3PC/timeout), quorum consensus, failover
Hiệu năng (Throughput)	Tối ưu batch prepare/commit + log không đồng bộ + sử dụng Kafka hoặc gRPC

(4) Mô tả chi tiết các bước, trạng thái và phục hồi

Trạng thái của Participant:

- INIT → Nhận prepare
- VOTE YES / NO
- PRE-COMMIT (nếu 3PC)
- COMMIT / ABORT

Trạng thái Coordinator:

- PREPARE → Gửi đến các node
- Thu thập VOTE
- Nếu ALL YES → COMMIT
- Nếu có NO → ABORT

Sơ đồ lỗi:

Kịch bản	Xử lý
Coordinator chết trước khi gửi quyết định	Participant sẽ chờ timeout → hỏi lẫn nhau (giao thức "termination") hoặc leader mới được bầu
Participant chết sau khi gửi YES	Phục hồi lại log và tiếp tục từ trạng thái trước
Coordinator và tất cả participant cùng chết	Khi khởi động lại, dựa vào log để rollback hoặc hỏi trạng thái quyết định cuối cùng từ quorum

Câu 4:

Định nghĩa: Phục hồi lùi và phục hồi tiến

Phục hồi lùi (Rollback Recovery)	Phục hồi tiến (Forward Recovery)
Khôi phục về trạng thái trước lỗi đã lưu lại (checkpoint), sau đó thực hiện lại các hành động	Sửa lỗi tại chỗ hoặc chuyển sang trạng thái an toàn kế tiếp mà không cần quay lại quá khứ
Phổ biến trong các hệ thống có khả năng checkpoint	Ít phổ biến, chủ yếu dùng trong hệ thống an toàn, hệ thống thời gian thực, AI tự học

Ưu – Nhược điểm:

Tiêu chí	Phục hồi lùi	Phục hồi tiến
Ưu điểm	Dễ triển khai, phù hợp với các giao dịch phân tán (database, dịch vụ web)	Không mất trạng thái hiện tại, phục hồi nhanh
Nhược điểm	Có thể mất thông tin nếu checkpoint quá cũ; cần nhiều dữ liệu log	Khó thực hiện; cần biết rõ lỗi là gì và cách sửa tự động

Vai trò của checkpoint và message logging trong phục hồi lùi

Checkpoint:

- Ghi lại toàn bộ trạng thái tiến trình tại một thời điểm → dùng làm mốc phục hồi.
- Giúp giảm chi phí phục hồi vì không cần chạy lại từ đầu.

Message Logging:

- Ghi lại các thông điệp đã nhận và xử lý sau checkpoint.
- Giúp phát lại thông điệp sau khi phục hồi checkpoint để quay lại đúng trạng thái trước khi lỗi xảy ra.

Kết hợp cả hai để giảm chi phí lưu trạng thái:

- Checkpoint không cần quá thường xuyên nếu có logging.
- Logging chỉ cần lưu sau checkpoint → tránh lưu trữ dư thừa toàn bộ thông điệp.
- Tối ưu chi phí lưu trữ và thời gian phục hồi.

Đề xuất chính sách checkpoint và logging cho hệ thống giao dịch ngân hàng

Yêu cầu: Không mất dữ liệu, phục hồi nhanh, hiệu năng cao.

Chính sách checkpoint:

- Chu kỳ checkpoint định kỳ (ví dụ mỗi 10 phút) hoặc khi có giao dịch lớn/nhạy cảm kết thúc.
- Có thể kết hợp checkpoint vi sai (incremental) nếu dữ liệu lớn.

Chiến lược message logging:

- Ghi log đồng bộ (synchronous) khi nhận hoặc xử lý yêu cầu tài chính.
- Sử dụng logging kiểu hồi phục có thể xác định (deterministic replay) hoặc có thứ tự (FIFO log) để tái phát dễ dàng.

Đảm bảo phục hồi nhanh – hiệu năng cao:

- Áp dụng giao thức non-blocking recovery (với message logging causal).
- Ghi log song song với xử lý giao dịch, tránh làm chậm.
- Dùng SSD hoặc bộ nhớ tạm RAM + flush định kỳ để tăng tốc.

So sánh checkpoint độc lập vs checkpoint phối hợp

Tiêu chí	Checkpoint độc lập	Checkpoint phối hợp
Recovery line	Khó xác định, có thể gặp vấn đề "domino"	Dễ xác định, nhất quán toàn hệ thống
Hiệu năng khi hoạt động	Không bị gián đoạn; nhẹ	Bị chặn toàn bộ hệ thống để đồng bộ checkpoint
Độ phức tạp triển khai	Dễ hơn (mỗi tiến trình tự checkpoint)	Phức tạp hơn (cần đồng bộ, quản lý tập trung hoặc thông qua marker)

Checkpoint phối hợp phù hợp với hệ thống real-time cần khôi phục chính xác, còn độc lập phù hợp với hệ thống nhẹ, truyền thông không thường xuyên, như mạng cảm biến.

Cơ chế phục hồi tổng thể cho hệ thống IoT phân tán

Bối cảnh:

Mạng cảm biến môi trường (ví dụ: đo nhiệt độ, độ ẩm, chất lượng không khí), với:

- Nút cảm biến (sensor node)
- Gateway thu thập
- Server trung tâm lưu trữ/hiển thị

(1) Sao lưu trạng thái và message logging

Checkpoint:

- Cảm biến cá nhân: checkpoint nhẹ (trạng thái cảm biến, chỉ số pin, cấu hình mạng).
- Gateway: checkpoint toàn bộ bảng routing, cache dữ liệu cảm biến.
- Checkpoint định kỳ (mỗi 30 phút) hoặc khi có sự kiện cấu hình mới.

Message Logging:

- Gateway log thông điệp dữ liệu đã nhận từ cảm biến (chỉ lưu ID + timestamp).
- Server log tất cả kết quả xử lý/giản đồ cảnh báo → dùng để khôi phục phân tích.

(2) Cơ chế phát hiện lỗi và kích hoạt phục hồi

- Cảm biến không phản hồi → gateway gửi ping + timeout.
- Gateway có thể dùng WDT (watchdog timer) hoặc heartbeat định kỳ.
- Khi phát hiện lỗi:
    - Gửi cảnh báo về server.
    - Kích hoạt tiến trình khôi phục nút: gửi cấu hình lại, gửi yêu cầu dữ liệu lại.

(3) Quy trình khôi phục khi một nút hoặc vùng mạng bị mất

- Nếu cảm biến chết → cấu hình lại từ gateway (nếu không chết vật lý).
- Nếu gateway chết:
    - Cảm biến tự động kết nối gateway khác (nếu có).
    - Server phục hồi từ checkpoint gần nhất + message log từ gateway khác (nếu dùng phân tán).

(4) Cân bằng giữa tính sẵn sàng, nhất quán, và chi phí

Mục tiêu	Giải pháp
Sẵn sàng	Nhiều gateway; cảm biến kết nối động
Nhất quán	Checkpoint đồng bộ giữa gateway và server
Chi phí mạng/thời gian	Checkpoint phân tán (phi phối hợp); log nén và gửi theo lô

Chi tiết điều kiện checkpoint, phục hồi và resend thông điệp

Bước	Mô tả

Khởi tạo checkpoint	Khi cấu hình thay đổi, pin yếu, theo chu kỳ, hoặc khi server yêu cầu
Tái gửi thông điệp	Gateway giữ log ID các thông điệp đã nhận. Khi khôi phục → yêu cầu cảm biến resend nếu thấy thiếu
Đường phục hồi	Gateway xác định bằng checkpoint + log gần nhất. Nếu cảm biến không log được, chỉ dựa vào buffer của gateway/server