並列分散コンピューティング
(12)分散スナップショット

大瀧保広

今日の内容

- 分散スナップショット
  - 分散システムの「状態」の記録
- 故障からの分散システムの復旧

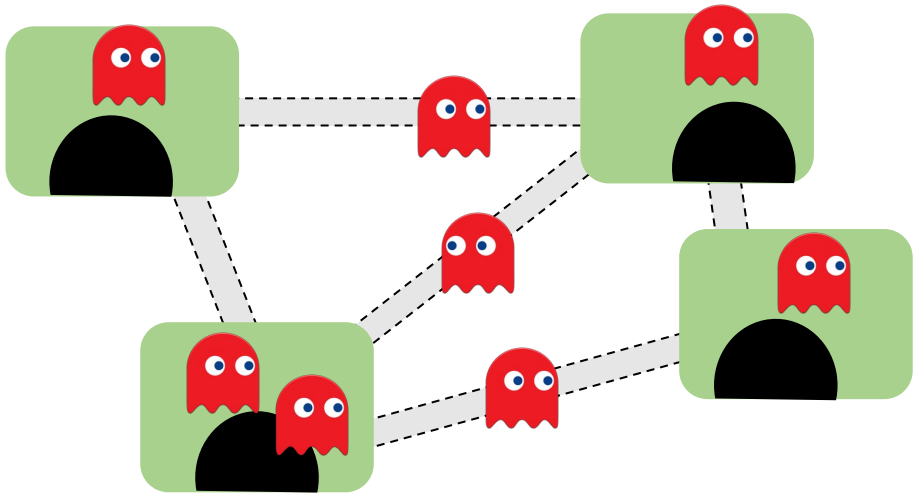
分散スナップショット

- 分散システム全体の状態を把握したい。
  - システム全体として正常なのか、異常が起きているのか

- しかし、分散システムでは
  各プロセスが独立に動作しているため
  「全体の状況」を知ることは容易ではない。

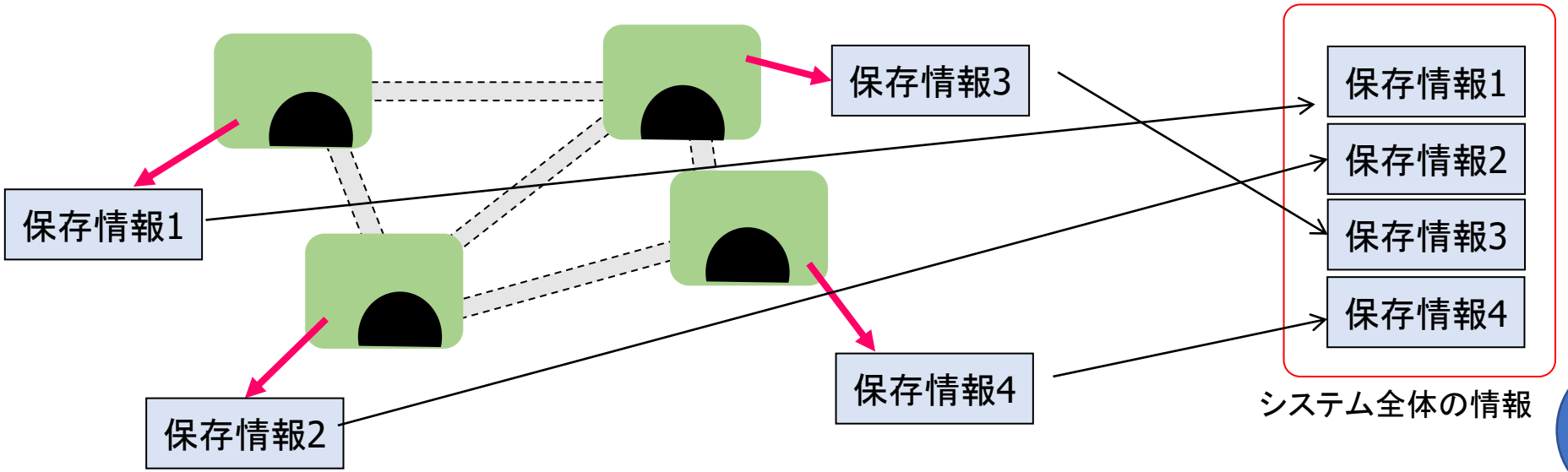
未確認生物の生息数調査

- いくつかの小島からなる国があり、
  島と島の間は地下トンネルで繋がっている。
- 国内に未確認生物が生息していることがわかった！
- 未確認生物が全部で何匹いるかを調べたい。
  （その生物にマーキングすることなく）
- 調査員は各島に一人ずつ配置できた。



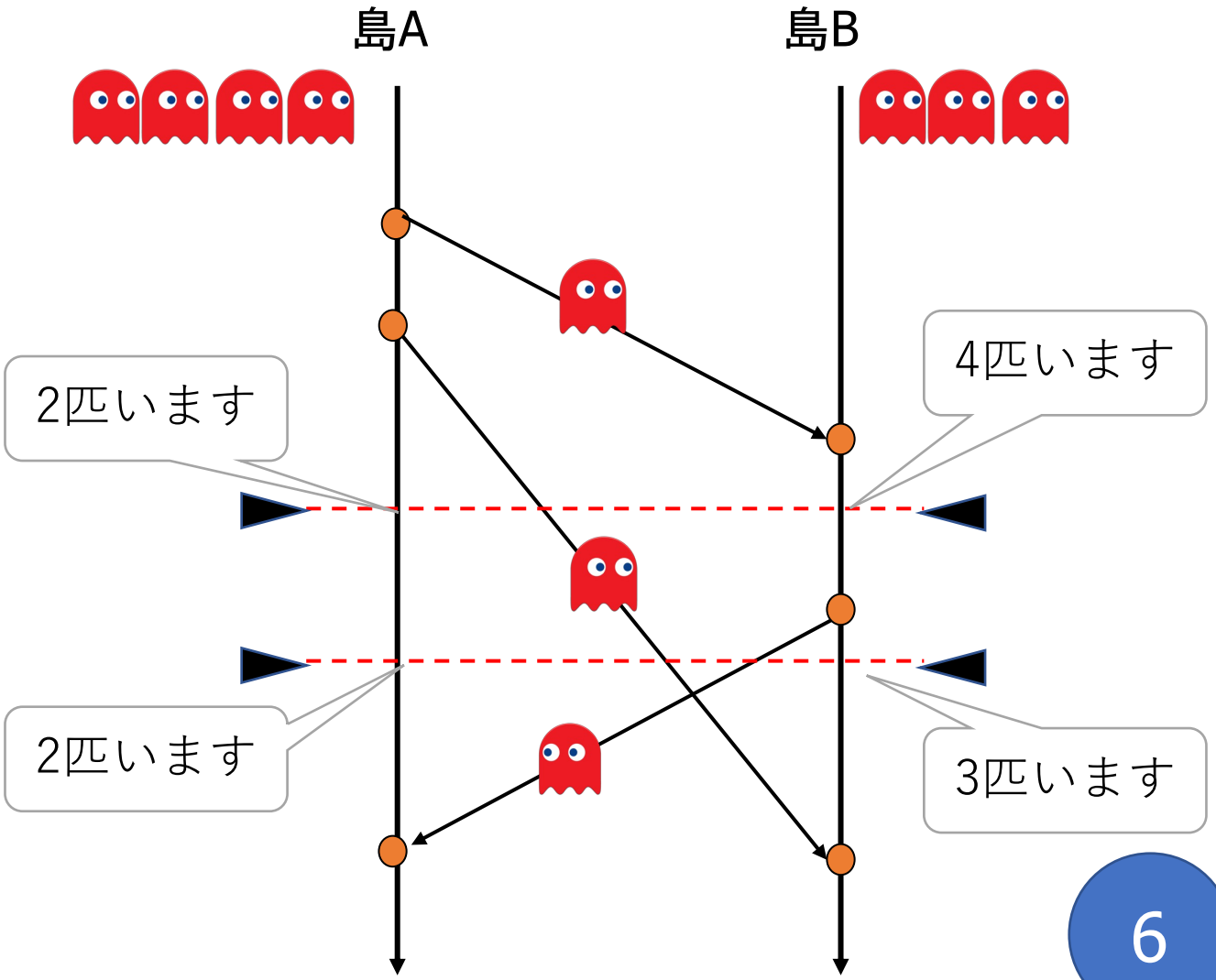
分散スナップショット

- 分散スナップショットとは
  分散システムのグローバル状態を知るためのアルゴリズム。
- 各プロセスの状態を保存し、それらの保存情報を集めることで、
  分散システムの全体状況を知る。



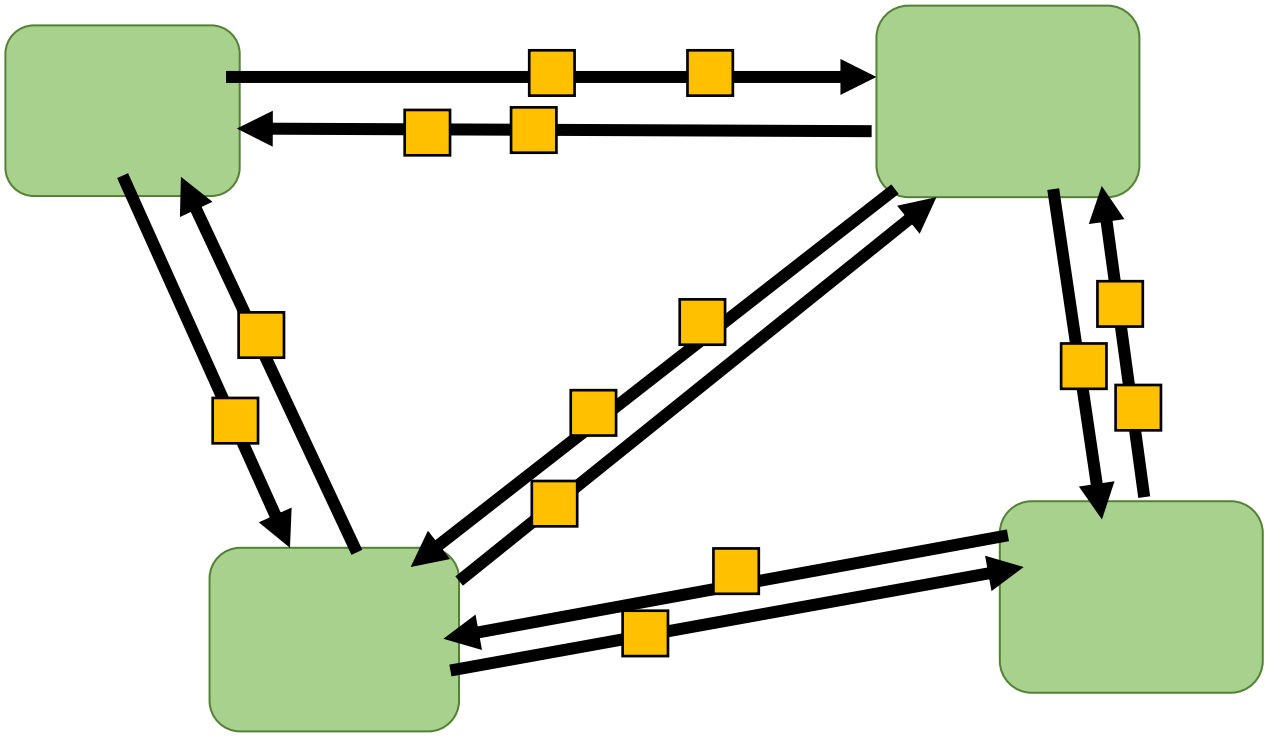
スナップショット　何が難しいか

- 単に時刻を決めて プロセスの局所状態を記録するだけでは、グローバル状態を正しく記録できない。
- 送信されたが まだ受信されていないメッセージの情報が足りない。

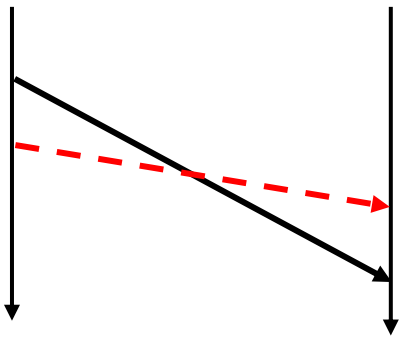


分散スナップショットの前提(1)

■通信路はすべてFIFO (First-In First-Out)である。
つまり、同じ通信路では、後から送ったメッセージが
先に送ったメッセージより先に届くことはない。



時空ダイアグラムでいうと
こういうことは起きない
ということ。



分散スナップショットの前提(2)

- プロセス間の通常のメッセージとは異なる特別なメッセージを送信することができる。
  - これをマーカーメッセージと呼ぶことにする。

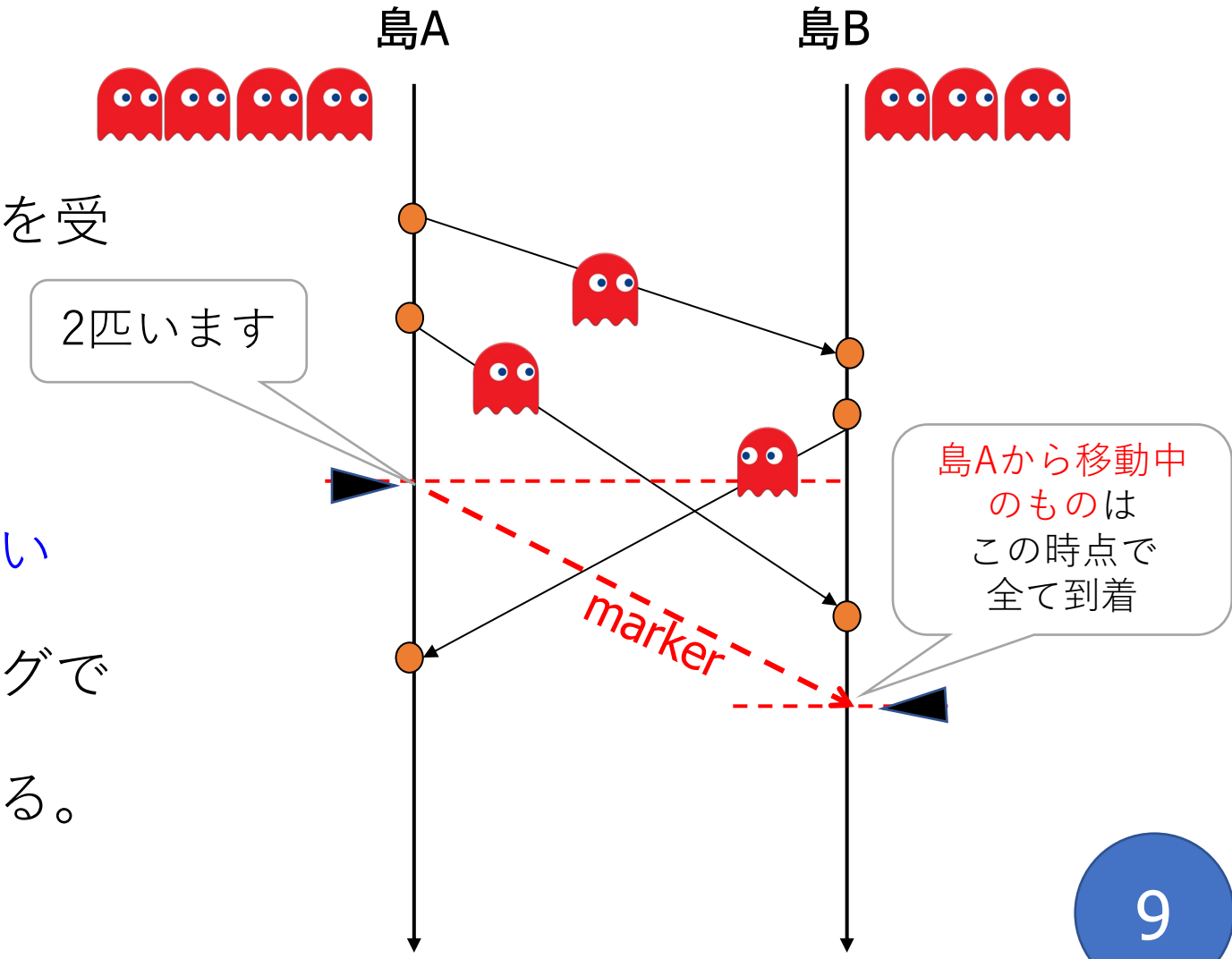
マーカーメッセージの実装は それほど難しくない。
識別方法の例：
「ヘッダの特定のビットが
1であればマーカーメッセージ、
0であれば通常メッセージである」など

マーカーメッセージで移動中のものを減らす

ポイント

■島Bでは
マーカーメッセージを受信したタイミングでスナップショットを取ることにする。

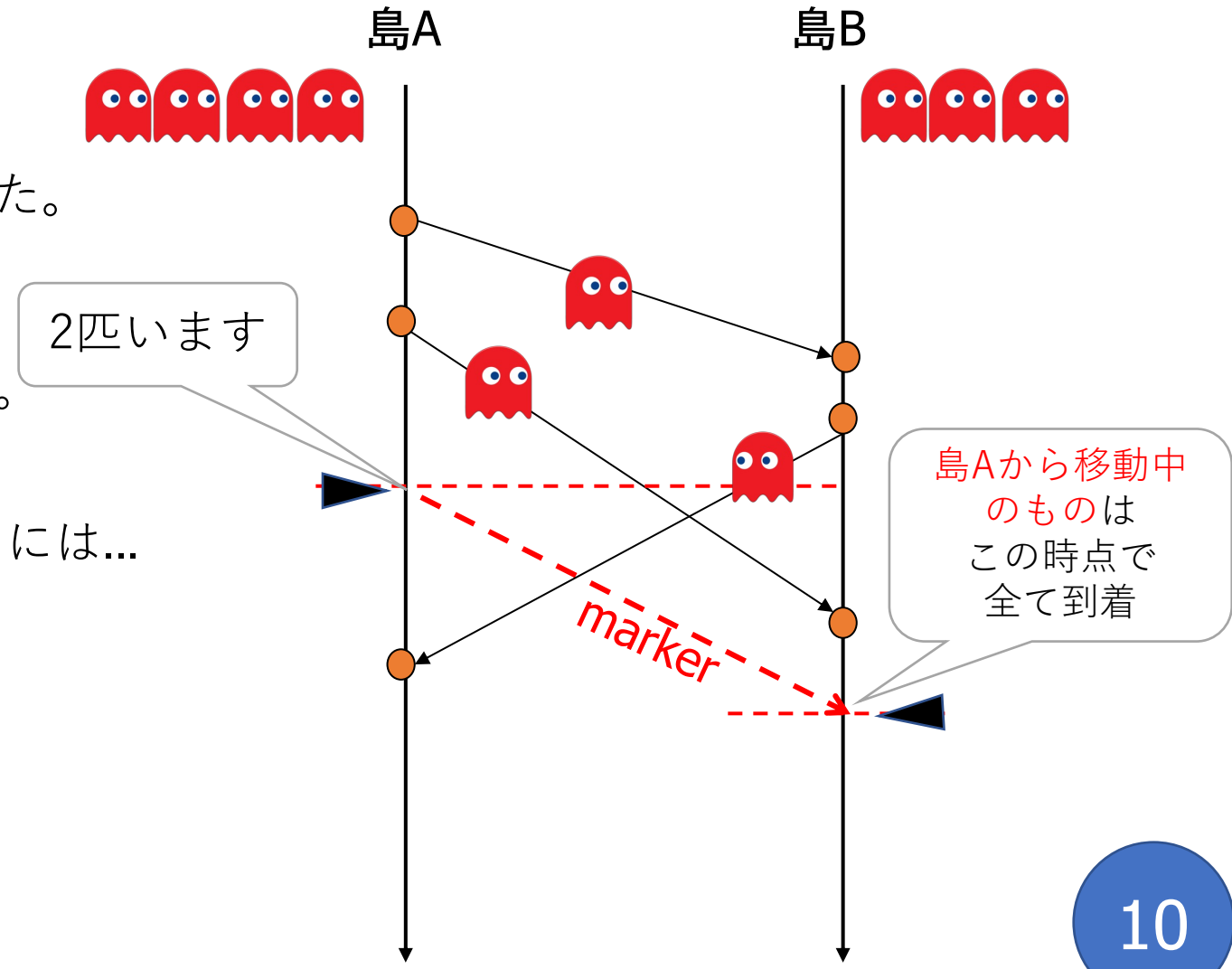
■送信されたがまだ受信されていないメッセージがより少ないタイミングでローカル状態を記録できるようになる。



逆方向に移動しているものが捕捉できていない

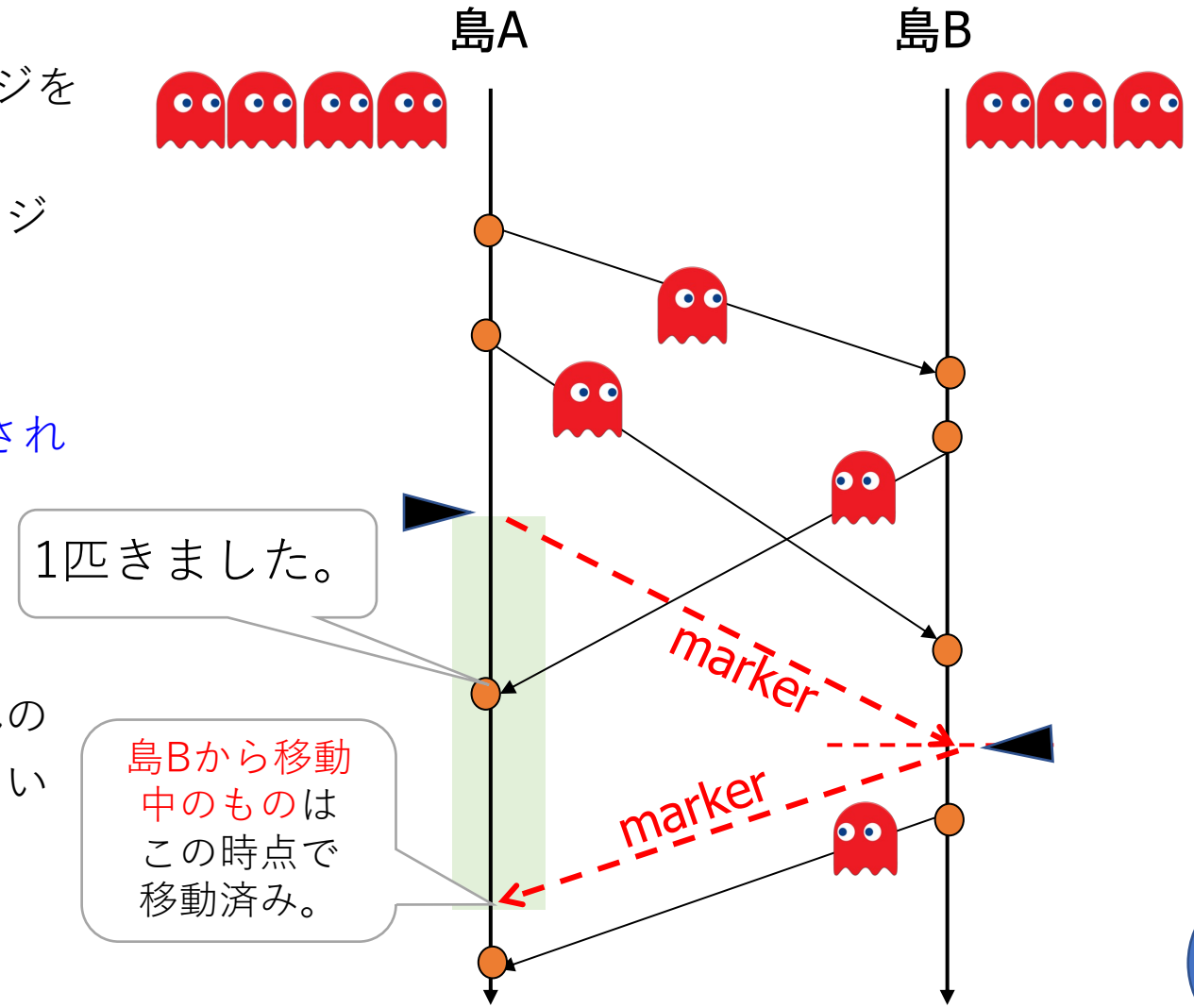
- 島A→Bの移動中のものは受信した状態でスナップショットがとれた。
- しかし島B→Aの方向に移動中のメッセージは記録し損ねたままである。

このメッセージを捕捉するには…



マーカーを送り返すことで捕捉漏れを検知

- 島Bがマーカーメッセージを受信したタイミングで即座にマーカーメッセージを送り返す。
- これにより、[送信されたがまだ受信されていないメッセージ]で捕捉しそこねたものを島Aで検知できる。
- このメッセージはいずれの局所状態にも反映されていないので、これは別途記録する必要がある。

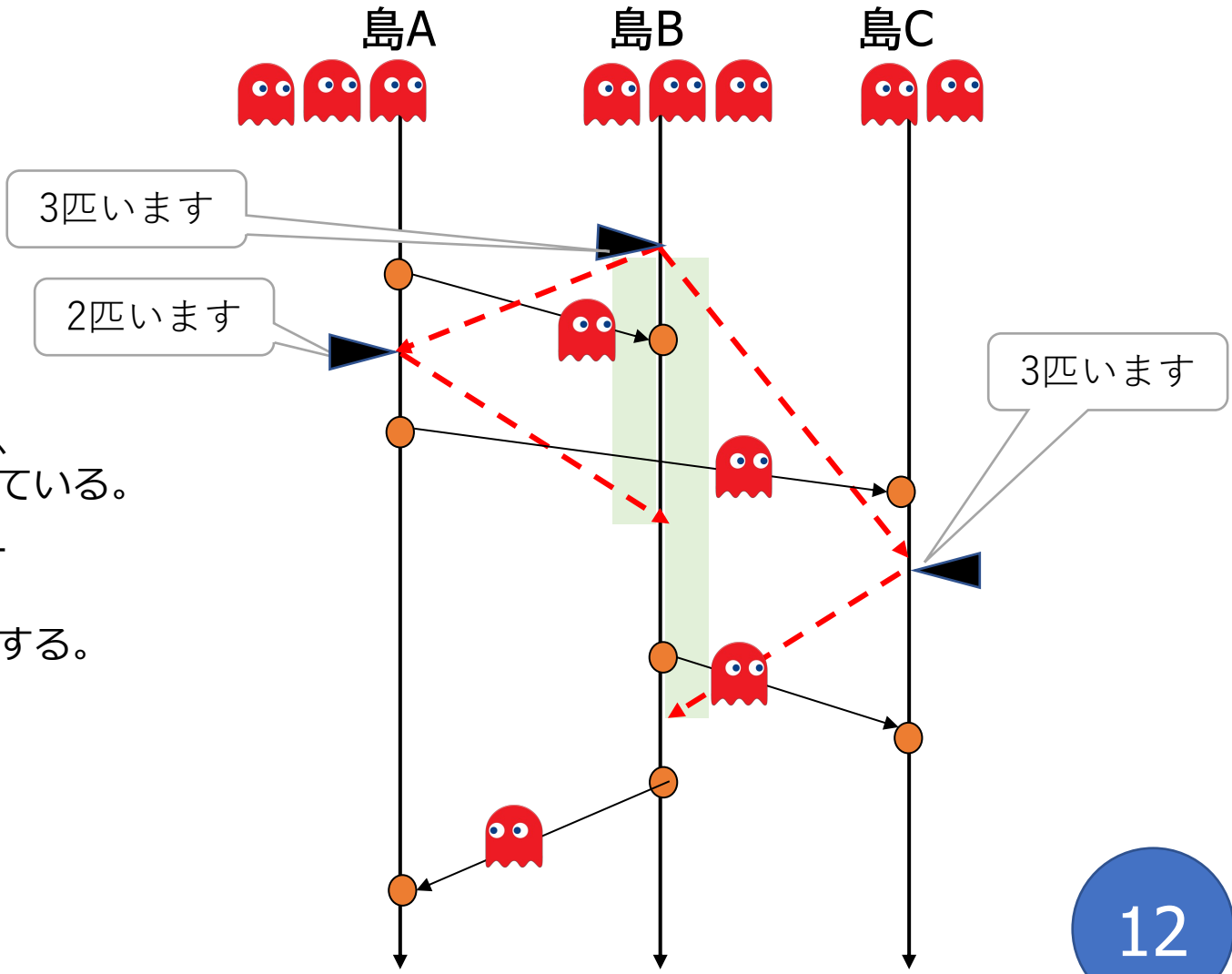


これで十分だろうか？

島が3つある例で考える。

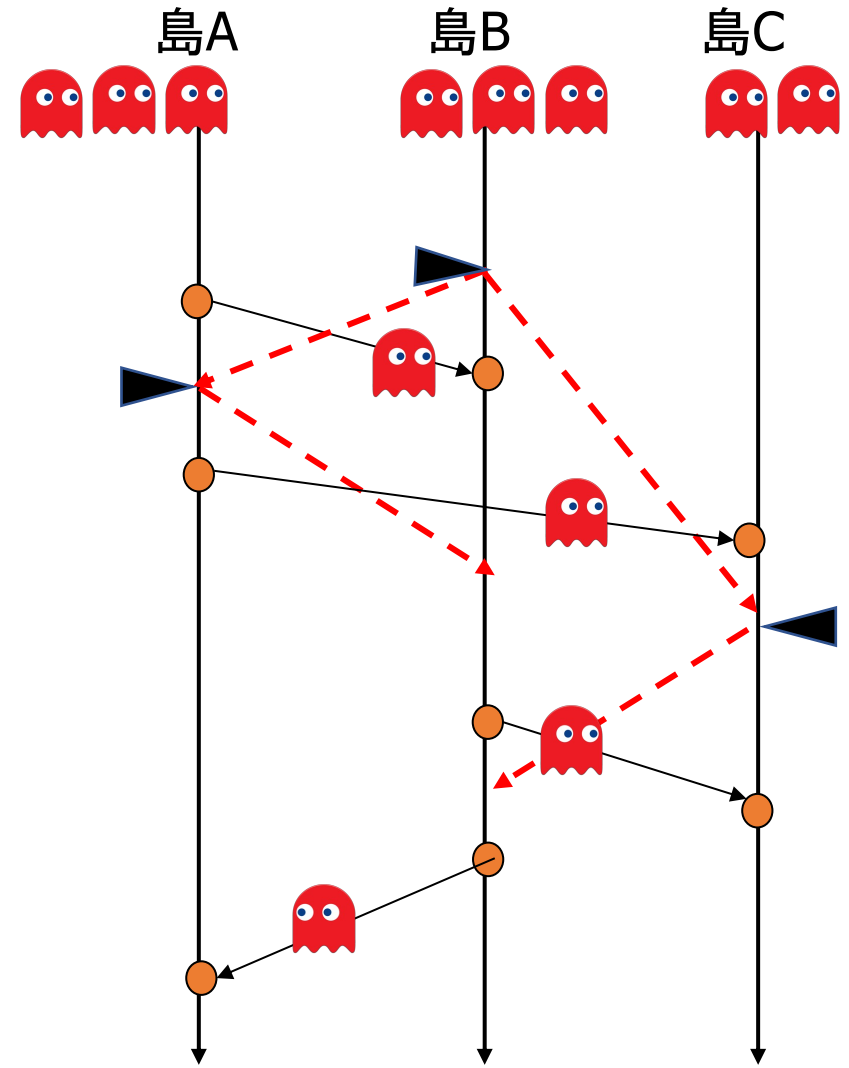
- 例えば、島Bが
  開始プロセスとして
  マーカーを送信する。
- 島A,島Cでは、
  マーカーメッセージを受信した
  時点でスナップショットをとり、
  マーカーメッセージを送り返している。
- 島Bでは、島A,Bからのマーカー
  メッセージの受信までに
  到着したメッセージを別途記録する。

よくみると、この例では
非常にまずいことが起きている。



こうなってしまうと おかしい

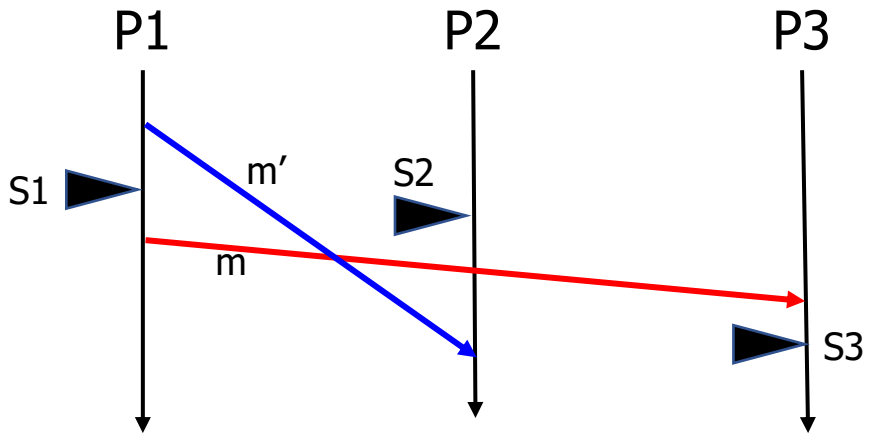
- スナップショットより下で出発し、スナップショットより上で到着しているメッセージがある。
- グローバル情報としてみると、メッセージをまだ送信していない状態とそのメッセージをすでに受信した状態が一緒に保存されていることになる。（整合性がない）



Orphanなメッセージ

- あるグローバル状態に保存されたローカル状態 Si , Sj があるとき、Siよりも後に送信され、Sj よりも前に受信されるような(通常)メッセージのことを Orphanなメッセージと呼ぶ。

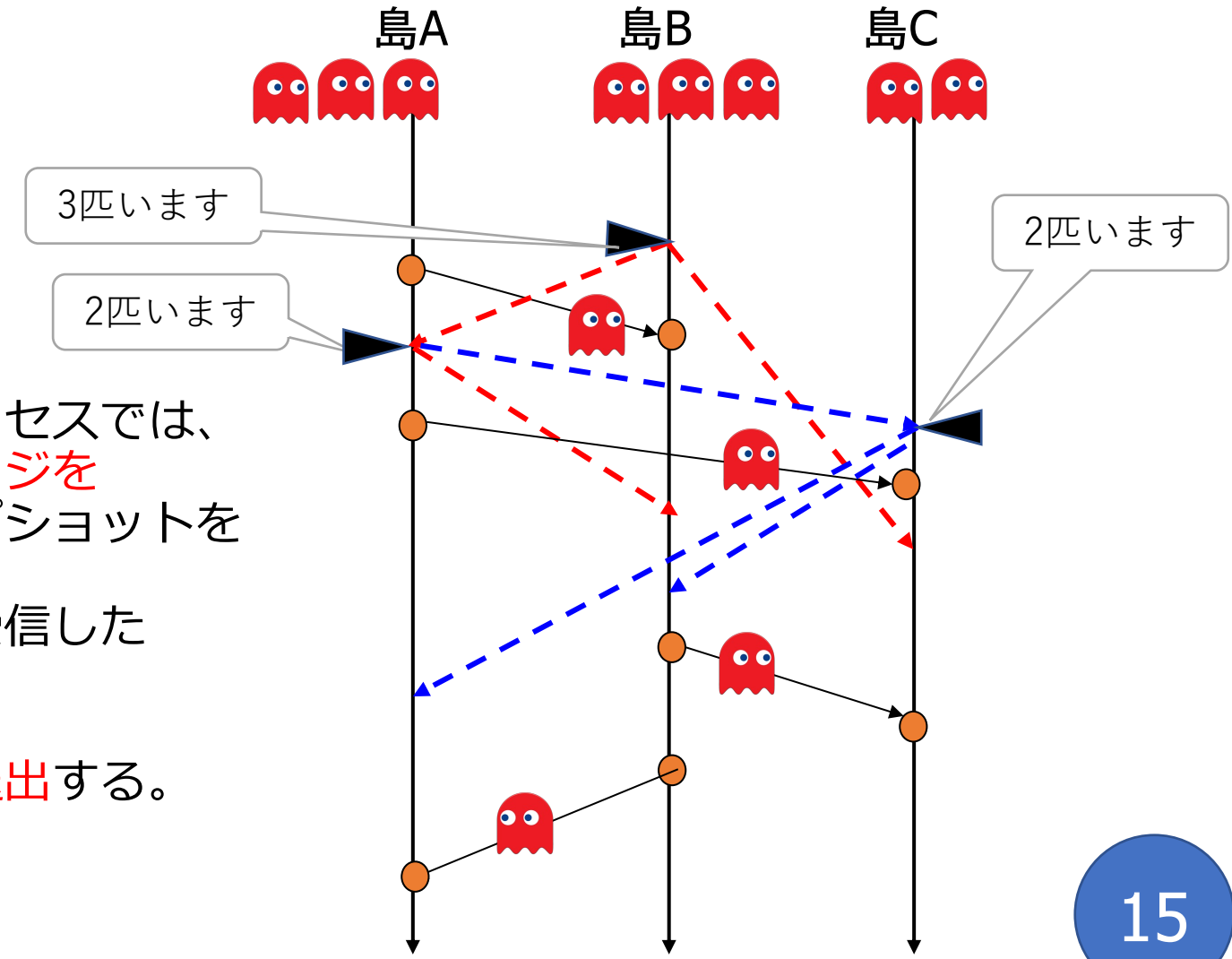
- 右の図において、メッセージ m はorphanなメッセージである。
- 一方、メッセージ m' はorphanなメッセージではない。
  - m'の送信は S1 の前であり、受信は S2の後である。このようなメッセージmは通信遅延によって普通に起こりうる。



どのように改良したら良いか

■開始プロセスが
マーカーメッセージを
送信するときには、
すべての通信路に送出
する。

■開始プロセス以外のプロセスでは、
最初にマーカーメッセージを
受信した時点でスナップショットを
とる。
マーカーメッセージを受信した
通信路以外も含めて、
すべての通信路に
マーカーメッセージを送出する。



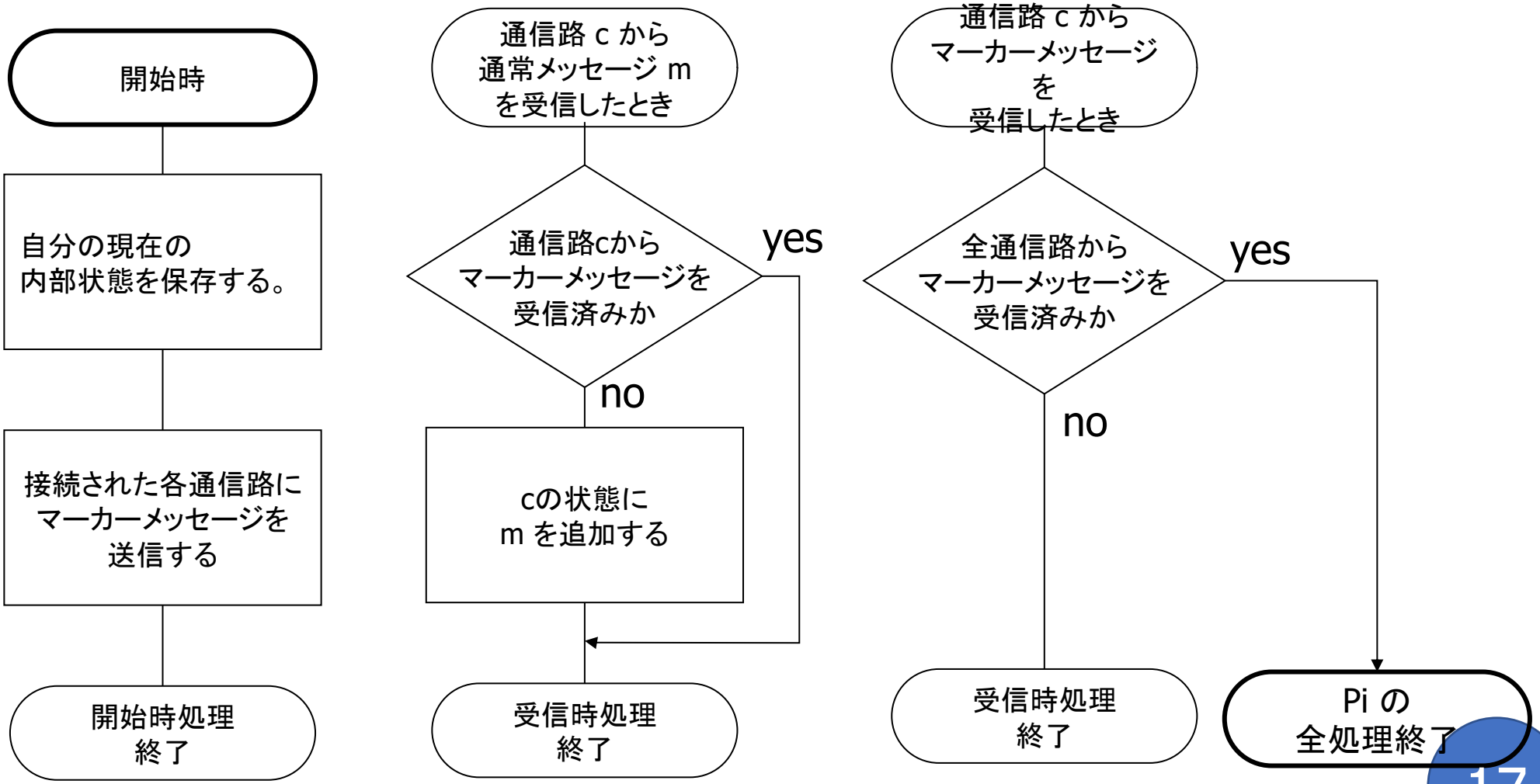
分散スナップショット アルゴリズム

/*開始プロセス Pi の動作 */
自分の現在の内部状態を保存する。

for all Pi に接続された各通信路 c について do
　　マーカーメッセージを c に送信する
end for

repeat
　　もしマーカーメッセージ未受信の通信路 c から
　　通常メッセージ m を受信したら、
　　m を通信路 c の状態として追加記録する。
until すべての接続通信路からマーカーメッセージを受信する。

開始プロセス Pi の処理フロー



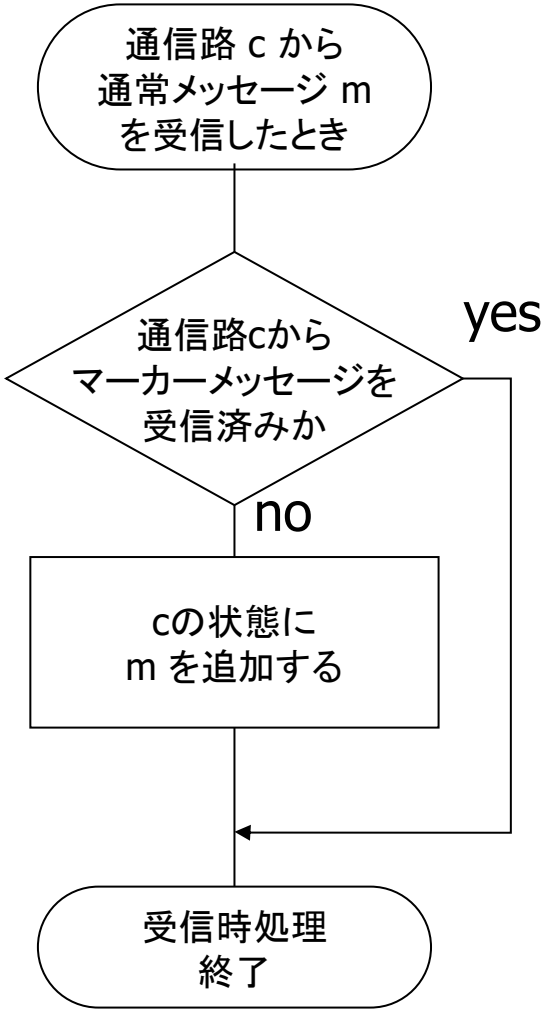
分散スナップショットアルゴリズム(他プロセス)

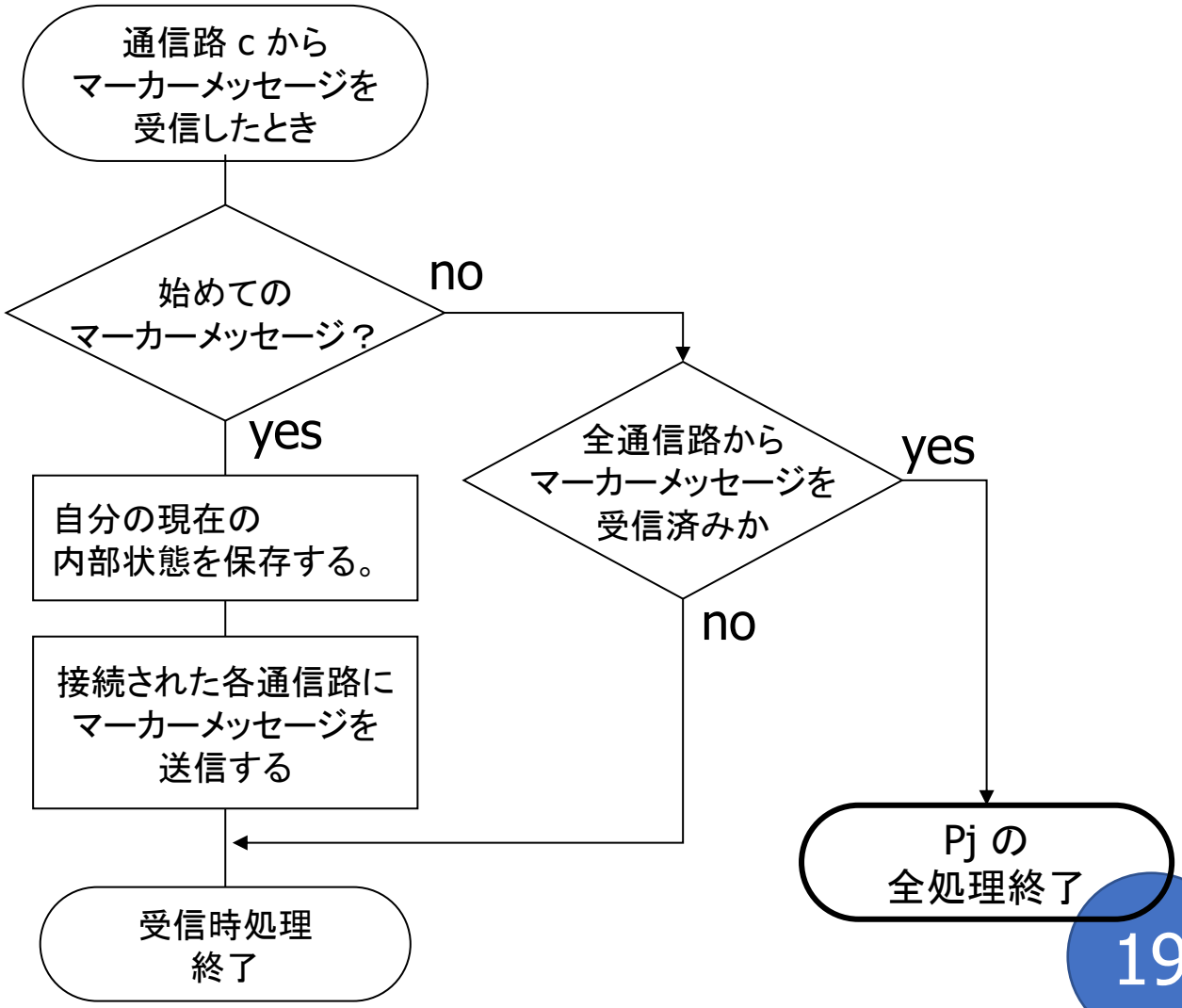
/*ほかのプロセス　Pj の動作 */
マーカーメッセージを最初に受信した時、
自分の内部状態を保存する。

for all Pj に接続された各通信路 c について do
　　マーカーメッセージを c に送信する
end for

repeat
　　マーカーメッセージ未受信の通信路 c から
　　通常メッセージ m を受信したとき、
　　m を通信路 c の状態として追加する。
until すべての接続通信路からマーカーメッセージを受信する。

その他のプロセス Pj の処理フロー





その他のプロセスの動き（別表現）

■開始プロセス以外のプロセスでは、
最初のマーカーメッセージの受信をもって、
分散スナップショットアルゴリズムの開始を知る。

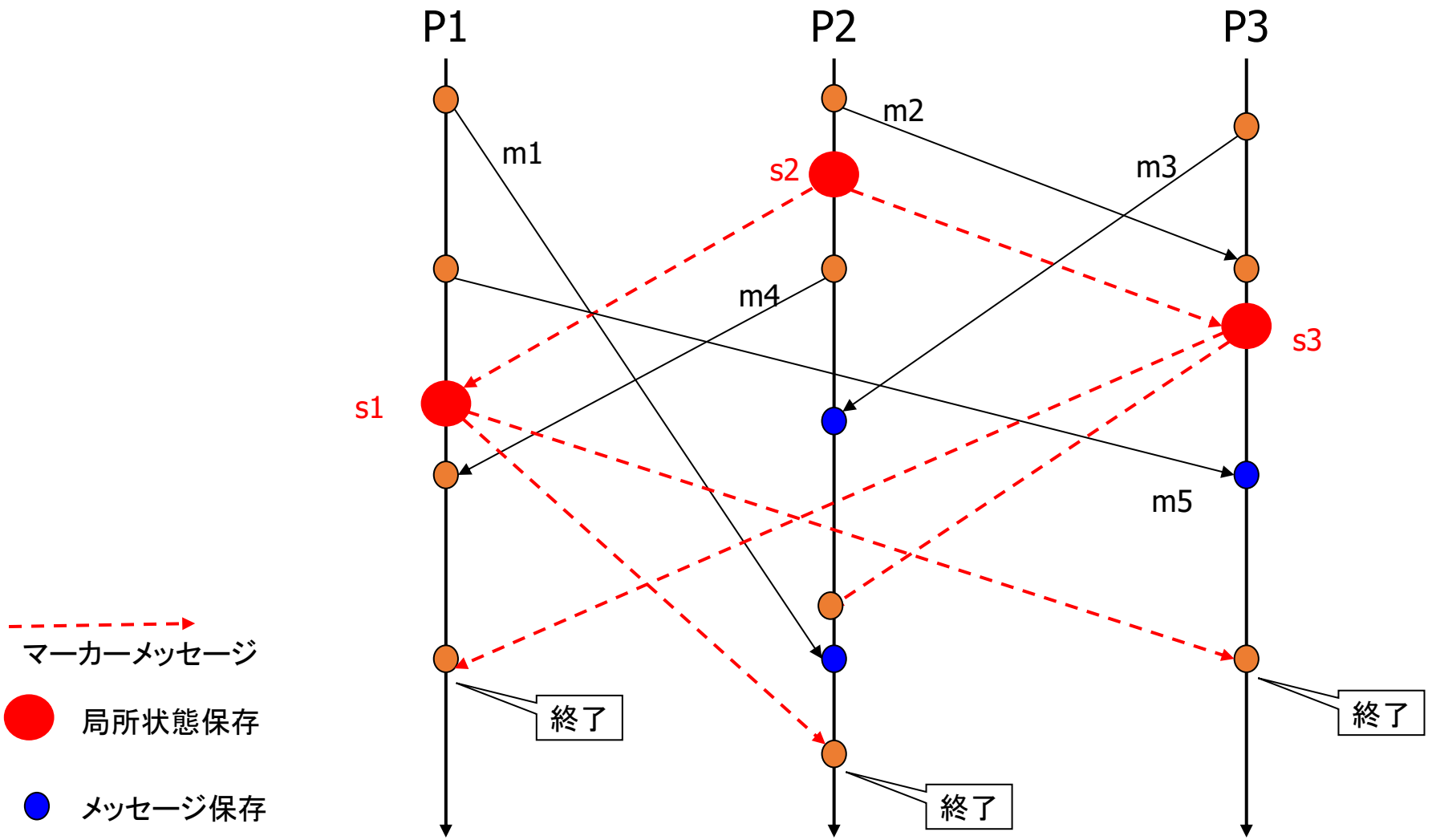
1. そのときの内部状態を保存する。
2. すべての通信路にマーカーメッセージを送信する。

■マーカーメッセージをまだ受信していない通信路から
通常メッセージを受信したら、
保存情報としてそのメッセージを追加する。

■すべての通信路からマーカーメッセージを受信したら、
分散スナップショットの処理を終了する。

■すべてのプロセスで処理が終了したとき、
各プロセスの保存情報（内部状態とメッセージ集合）の組が
分散スナップショットとなっている。

動作例(1)


P1
P2
P3
m1
m2
m3
m4
m5
s1
s2
s3
終了
終了
終了
マーカーメッセージ
局所状態保存
メッセージ保存

動作例（1）の説明

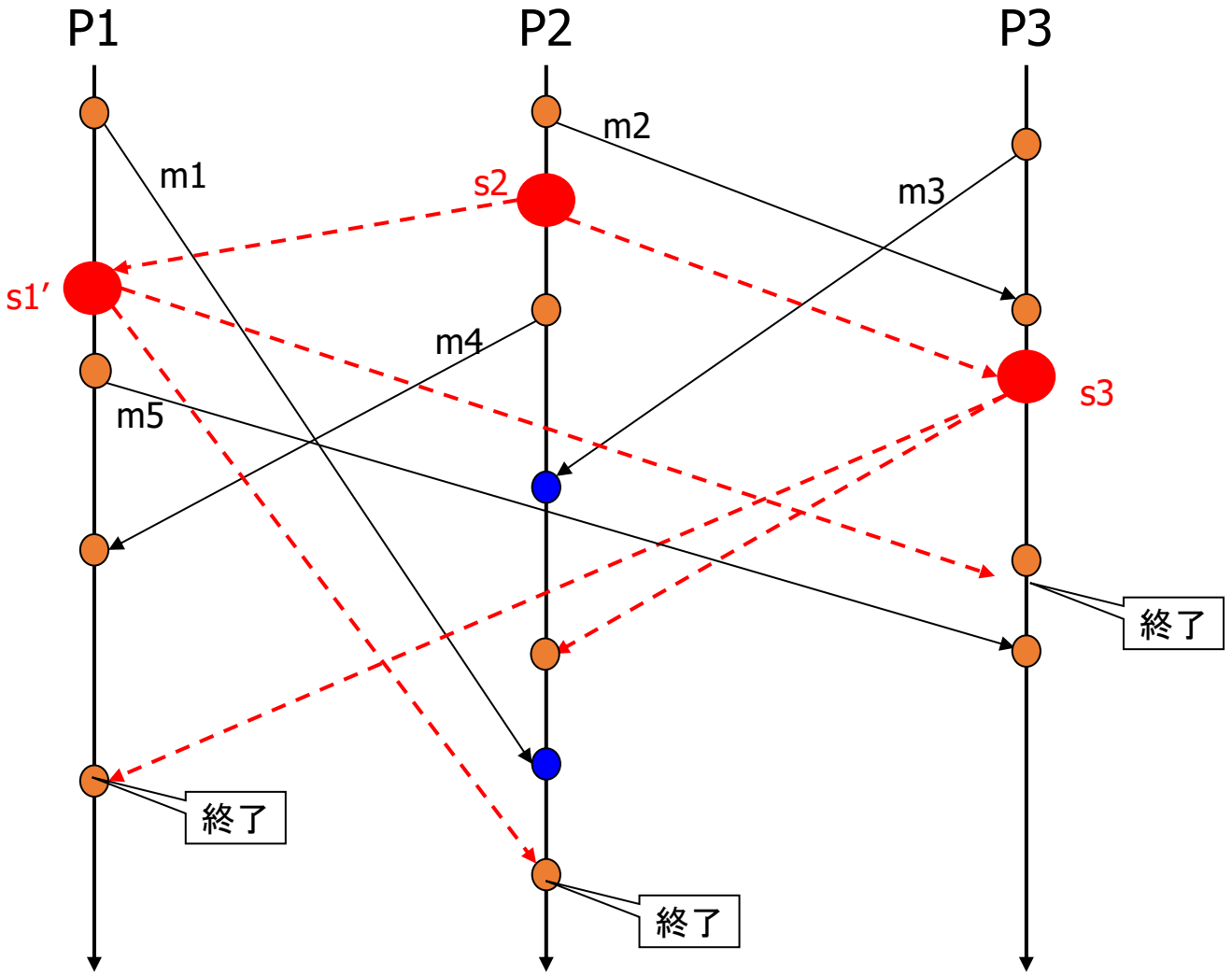
- 3つのプロセス P1, P2, P3
- すべてのプロセスの間に通信路があるものとする。
- 開始プロセスは P2

- 最終的な分散スナップショットは
  状態の組（s1, s2, s3）　と　メッセージの集合{m1, m3, m5}。

- 各プロセスの動作および通信の遅延時間は一定ではないので、
  同じようなタイミングで分散スナップショットアルゴリズム
  を走らせても、得られる結果（スナップショットの内容）は
  一意ではない。
  - 例えば、次の動作例（2）の場合の分散スナップショットは
    状態の組（s1', s2, s3）　と　メッセージの集合{m1, m3}。

動作例(2)

動作例(1)に
くらべると
早く到着した



マーカーメッセージ

局所状態保存

メッセージ保存

分散スナップショットの性質

- 分散スナップショットで得られた状態を (S1, S2, ... , Sn)とする。
- このとき、任意の状態の組 (Si, Sj)についてOrphanなメッセージは存在しない。
  - ある状態 Si より後に送信されて、
    別の状態 Sj よりも前に受信されるような
    通常メッセージ m
    は存在しない。

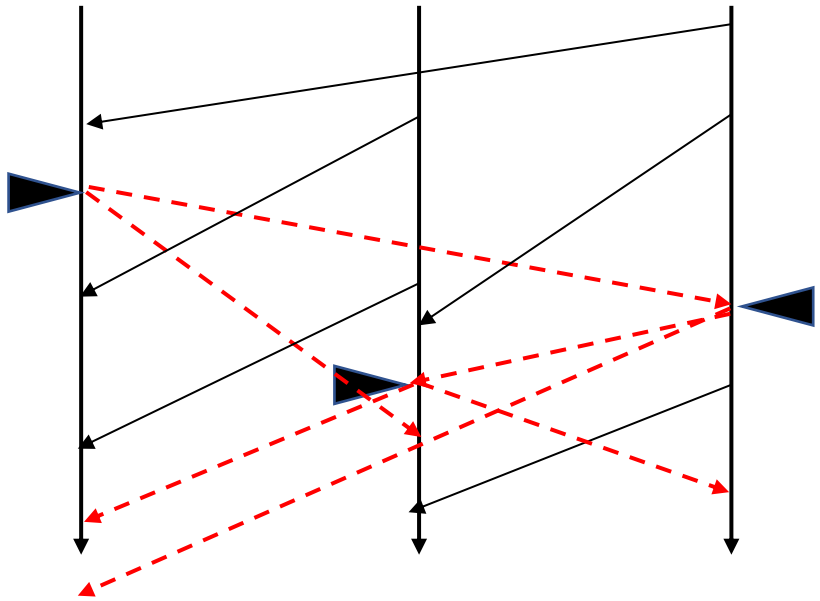
- メッセージ集合の性質

  分散スナップショットに含まれるある状態より**前に送信**され、
  まだ受信されていないメッセージは、
  すべて分散スナップショットのメッセージ集合に含まれている。

分散スナップショットの性質(つづき)

■分散スナップショットで局所状態を保存したイベントを
(S1, S2, … , Sn)とするとき、
任意の Si, Sj の組について、Si と Sj は並行である。

（ただしイベントの順序関係の判定ではマーカーメッセージは無視する）



分散スナップショットの応用

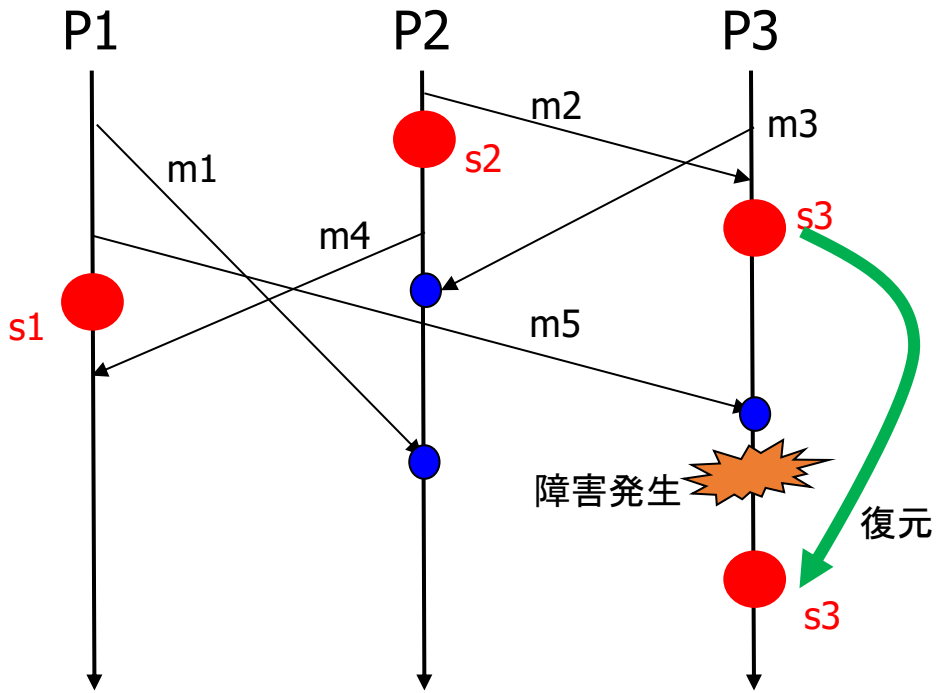
分散スナップショットによる障害復旧

- 障害復旧とは：
  計算機では、故障に備えて 作業の途中の状態を保存することはよく行われている。
  - 例：多くのエディタは、編集途中の文書を定期的に自動保存している。
    - 計算機が途中で故障した時には保存しておいた状態にロールバックすることができるようにしている。
    - この自動保存により、文書全体が失われることなく、被害を小さくすることができる。

- 分散スナップショットは、分散システムにおいて故障発生時の障害復旧に用いることができる。

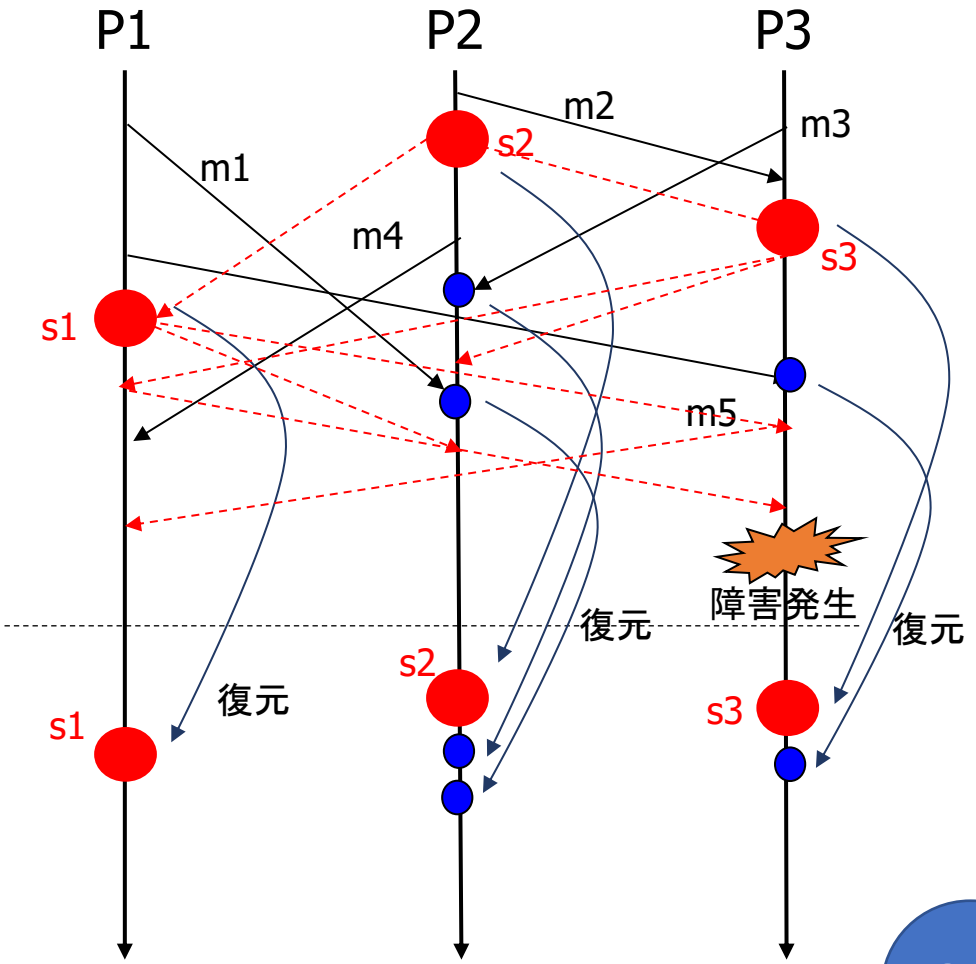
分散システムにおける復旧（ダメな例）

- 分散システムの場合には復旧は単純ではない。
- プロセスP3が故障した場合、復旧の方法として、「P3の状態を故障前の状態s3に戻す」だけでは不整合が生じる。
  - 本来ならば P3 は （状態S3の後に）P1からのメッセージ m5 を受け取る。P1はm5を送信済みと思っているため、P3には m5 は到着しない。



正しい復旧方法

- 故障したプロセスがP3だけであったとしても、すべてのプロセスと通信路の状態を分散スナップショットで保存していた状態に戻す必要がある。
- 「通信路の状態」の復元は、「記録されたメッセージ」を受信プロセスのバッファに配置するだけで良い。



今日のまとめ

- 分散システム全体の状況を把握する方法として、
  分散スナップショットアルゴリズムがある。
- 分散スナップショットアルゴリズムでは、
  あるプロセスから開始して、
  マーカーメッセージを送り合うことで、
  状態の集合　と　メッセージの集合　を求める。

- 分散スナップショットアルゴリズムを用いることで、
  分散システムが正常であることを確認したり、
  障害からの復旧を行うことができる。
- 復旧時には、全プロセスの状態をスナップショット時に
  セットし直す＋メッセージ集合のセットが必要。