故障に耐えるコンピュータ
～壊れても使えるシステム作りとは？～

平成24年度 市民講座 第8回（2月26日）

国立情報学研究所・アーキテクチャ科学研究系
米田友洋 yoneda@nii.ac.jp

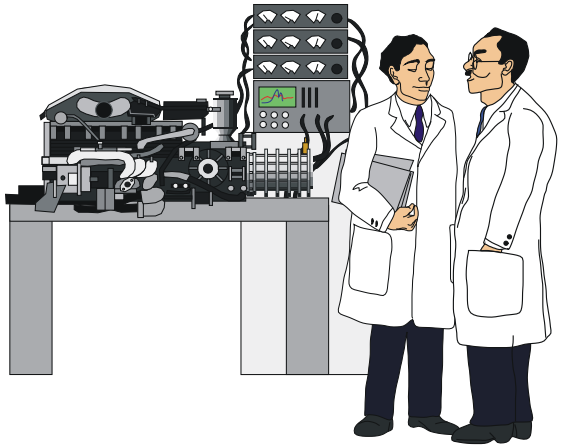
National Institute of Informatics
国立情報学研究所

あらまし

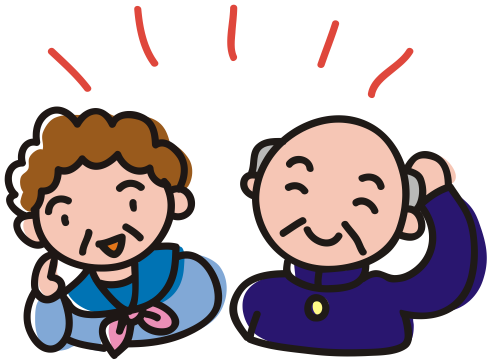
- 「故障」するとはどういうことか
- 「故障」に対する3つのアプローチ
- 「故障」に耐えるシステム作り
- 事例紹介
- 我々の取り組み
- まとめ

故障とは？

「壊れても使えるシステム」
作りをしている専門家



一般的な用法



「故障」

意味合いが違う

これが重要な意味を持つ

故障とは？

- （例）車が高速道路で止まってしまった！

修理屋さんで調べてもらうと，．．．
燃料噴射のための圧力を発生させるポンプが動かなくなっていた

「車が故障した」

- 専門用語的には，．．．

「故障」 → 燃料噴射ポンプの動作不良

車が動かない → システムとしての障害

この区別

ポイント1

故障とは？

- （例）車が高速道路で止まってしまった！

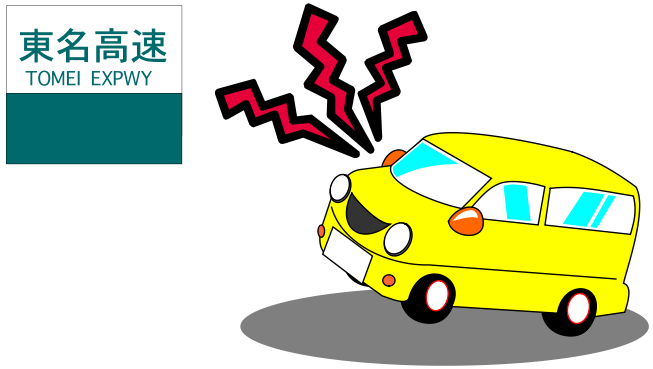
対処方法１

- 燃料噴射ポンプの故障原因を解明して，そのような故障が起こらないように設計し直す
- 定期的に検査して，必要なら部品交換をする

「故障」を防ぐ

故障とは？

- （例）車が高速道路で止まってしまった！



対処方法2

- 燃料ポンプを2つ用意しておく
- 通常は，それぞれのポンプの圧力の和で使用
- 一つが止まっても，残りのポンプの圧力だけでなんとか動かす

「故障」に耐える

故障とは？

- （例）車が高速道路で止まってしまった！
  - どちらの対処方法がいいか
    - 対処方法1の長所
      - コストが安い
      - 重量が軽い
    - 対処方法2の長所
      - 万が一，どちらかのポンプが止まってしまっても，車は止まらない

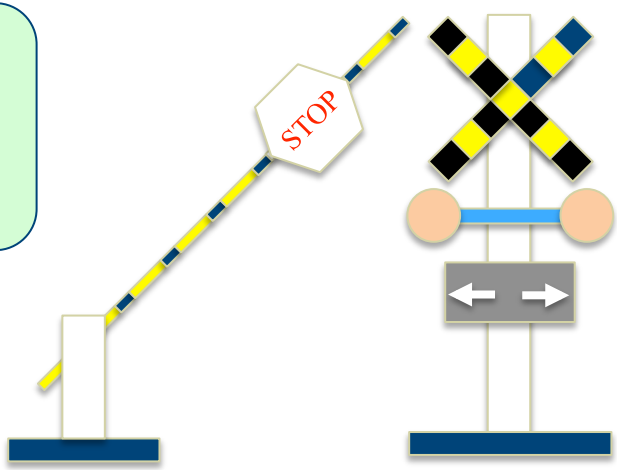
注意：コストに含まれるべきもの

- 実現するための費用
- （その故障により生じる損失・損害）×（発生率）

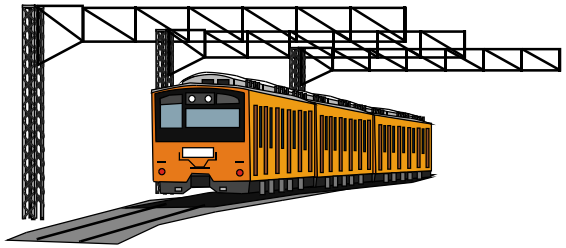
故障とは？

- （例）踏切に列車が近づいて来ても遮断機が降りない！

原因を調べてみると，．．．
周辺の工事で，踏切の電力ケーブルが切断されていた



「遮断機が故障した」



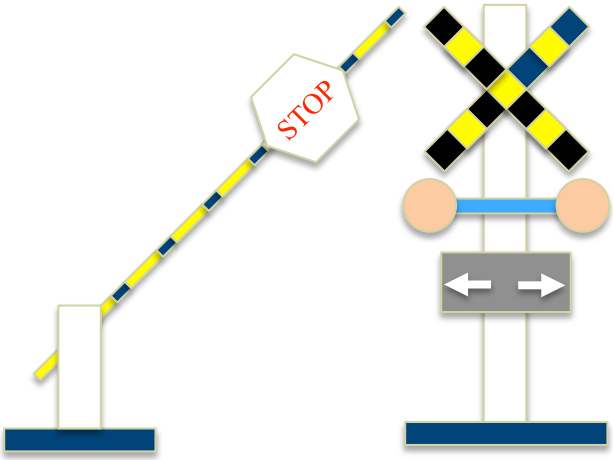
- 専門用語的には，．．．

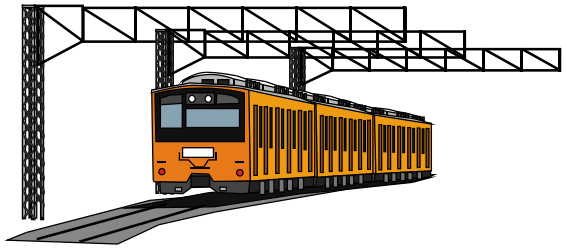
「故障」 ➡ 電力ケーブルの切断

遮断機が降りない ➡ システムとしての障害

故障とは？

- （例）踏切に列車が近づいて来ても遮断機が降りない！





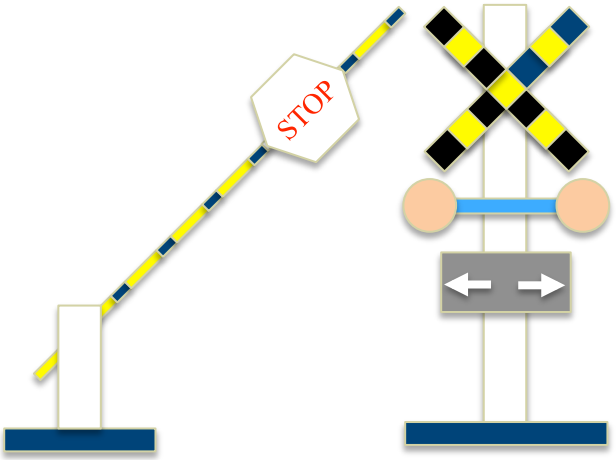
対処方法1

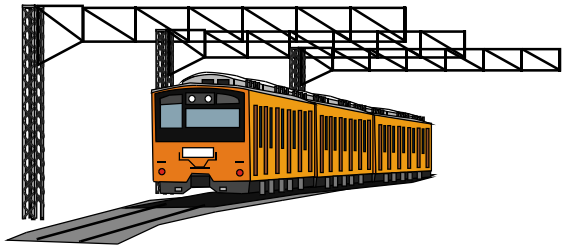
- 電力ケーブルのある場所には，それを明示し，工事による切断が起こらないように配慮する

「故障」を防ぐ

故障とは？

- （例）踏切に列車が近づいて来ても遮断機が降りない！





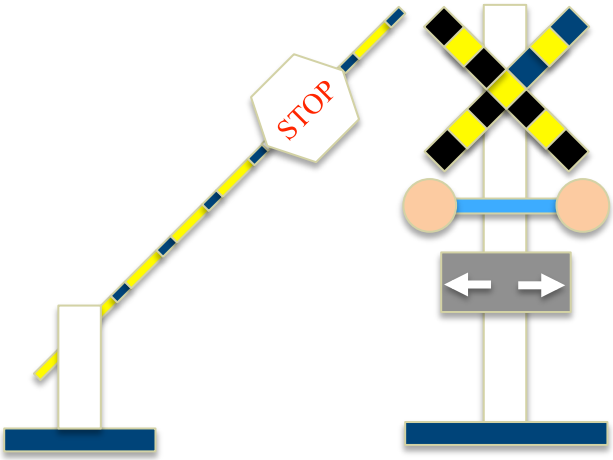
対処方法2

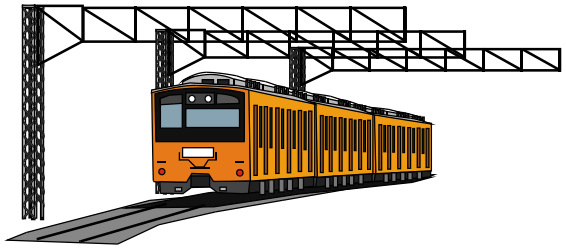
- 遮断機への電力供給を2系統とする

「故障」に耐える

故障とは？

- （例）踏切に列車が近づいて来ても遮断機が降りない！





対処方法3

- 遮断機への電力が絶たれたら，重力で遮断機が降りるように設計しておく

安全側に誤らせる

故障とは？

- 対処方法3の考え方
  - 出力（サービス）の分類
    - 安全側
    - 危険側
  - エネルギーの高い状態でのみ危険側出力が可能
    - 故障するとエネルギーは低い状態に移る
- 特徴
  - コストが安い
  - システムの障害は起こるが，常に安全側

故障とは？

- 安全側の概念の違い
  - 通常「止まること」が安全
    - 特に，鉄道系で長年用いられてきた概念
  - そうとは限らない場合
    - 自動車では「止まること」危険なことも，．．．
      - 高速道路で止まる
      - 街から遠く離れたところを走る場合
    - 鉄道系でも，．．．
      - トンネルの中で火災が発生した場合
      - 東日本大震災では，停電でさがりっぱなしになっている踏切のために渋滞が発生し，津波に襲われた

故障に対するアプローチ

- ３つのアプローチ
  - 対処方法１
    - 故障が起こらないようにする

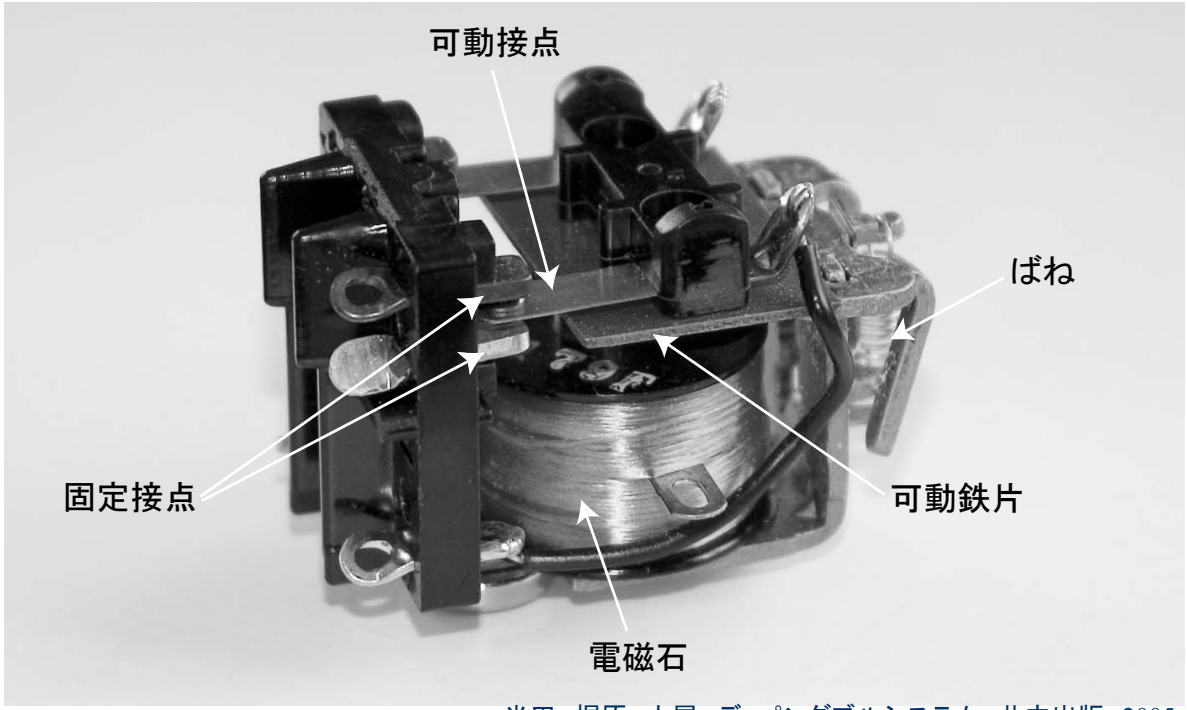
    フォールトアボイダンス
  - 対処方法２
    - 故障が起こっても，それに耐えるように設計しておく

    フォールトトレランス
  - 対処方法３
    - 故障が起こっても，システム障害が安全側になるように設計しておく

    フェイルセイフ

故障に対するアプローチ

- コンピュータの歴史の話
  - リレー
    - 信頼性低い、故障は避けられない
    - フォールトトレランス



米田，梶原，土屋，ディペンダブルシステム，共立出版，2005.

故障に対するアプローチ

- コンピュータの歴史の話
  - リレー
    - 信頼性低い、故障は避けられない
    - フォールトトレランス
  - 真空管
    - 信頼性やや向上
    - フォールトアボイダンス（ヒータ電圧を下げる）
  - トランジスタ, IC, VLSI：
    - 信頼性飛躍的に向上
    - フォールトアボイダンス（部品を選定する）
  - 大規模システム、超高信頼性システム
    - 故障は避け得ない
    - フォールトトレランス

故障に対するアプローチ

- ３つのアプローチ
  - 対処方法１
    - 故障が起こらないようにする

      フォールトアボイダンス
  - 対処方法２
    - 故障が起こっても，それに耐えるように設計しておく

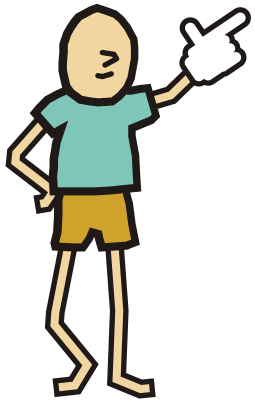
      フォールトトレランス
  - 対処方法３
    - 故障が起こっても，システム障害が安全側になるように設計しておく

      フェイルセイフ

今日のお話

「故障」に耐えるシステム作り

- 「故障は起こりうる」という前提がスタートポイント
- どうやって耐えるか
  - 先ほどの例
    - ポンプを2つ用意する
    - 2系統で遮断機の電力を供給する



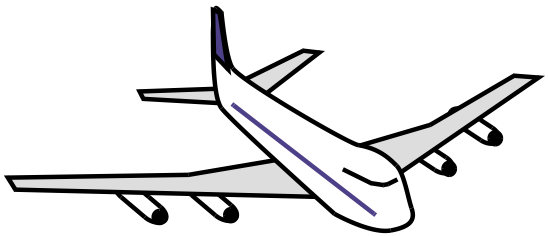
冗長性

ポイント2

根幹となる考え方

冗長性

- 身近な冗長性
  - 飛行機のエンジンは2つ以上ある
    - 1つのエンジンだけでもなんとか着陸できる



  - 二間続きの各部屋にあるエアコン

大体のものは性能向上が目的

冗長性

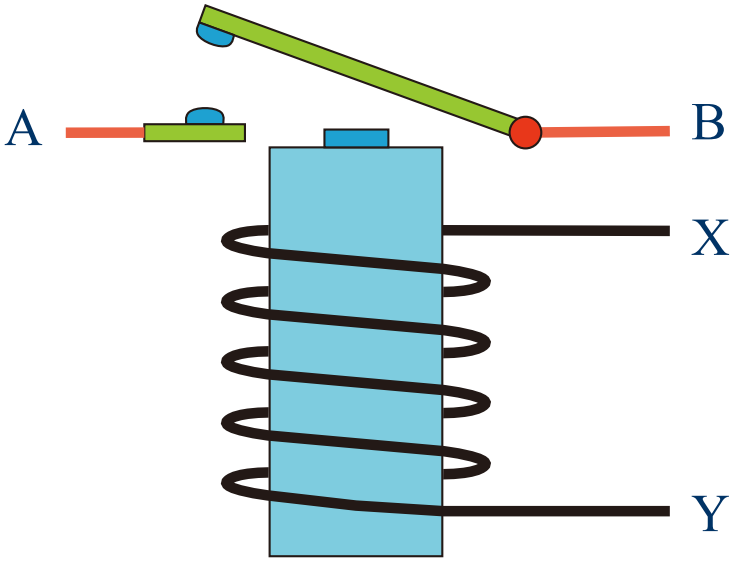
- 冗長度
  - 同じ機能を持ったユニットの個数

冗長度は大きければ大きいほどいいか？

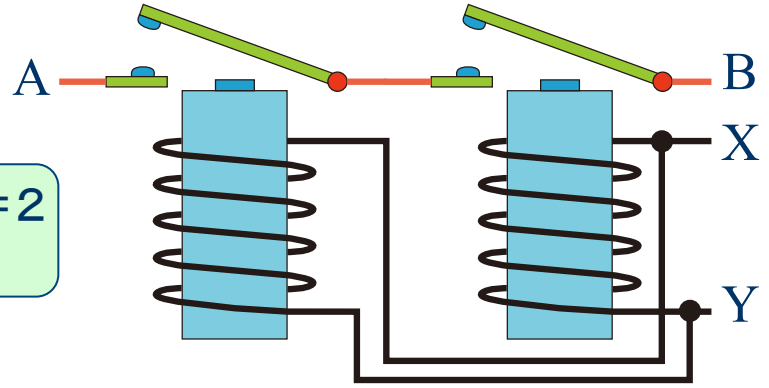
  - No!
    - コストがかかる
    - 場合によっては，故障の影響を受けやすくなる

冗長性

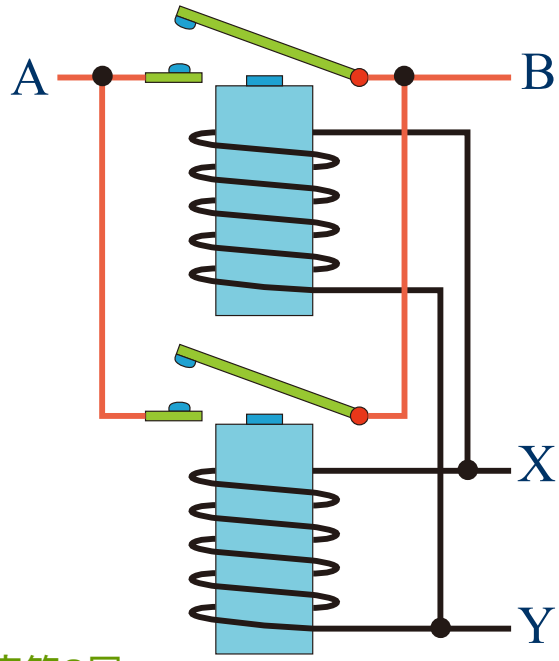
- リレーの接点の例



冗長度＝2
直列

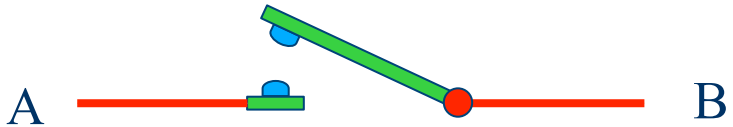


冗長度＝2
並列

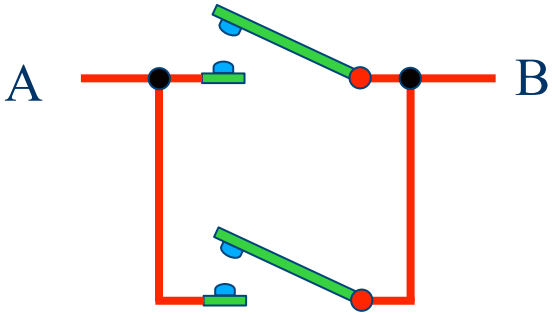


冗長性

- リレーの接点の例



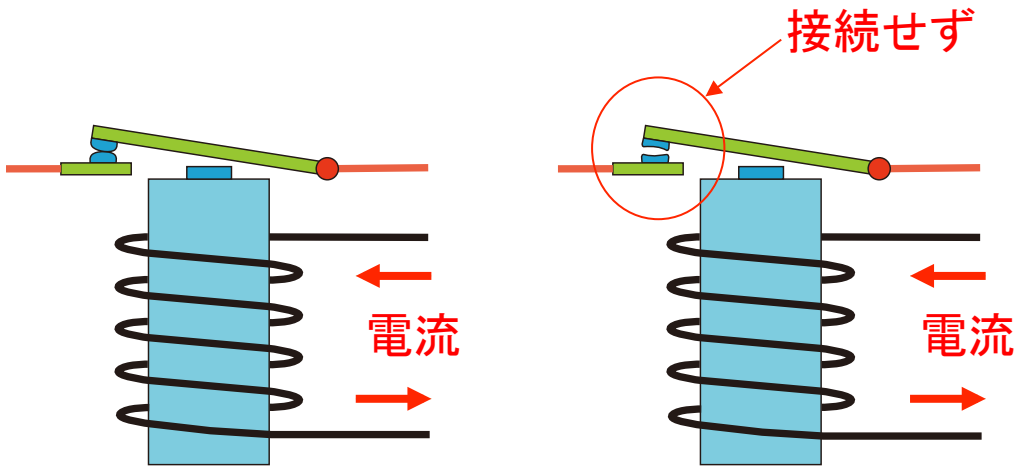
冗長度＝2
直列



冗長度＝2
並列

冗長性

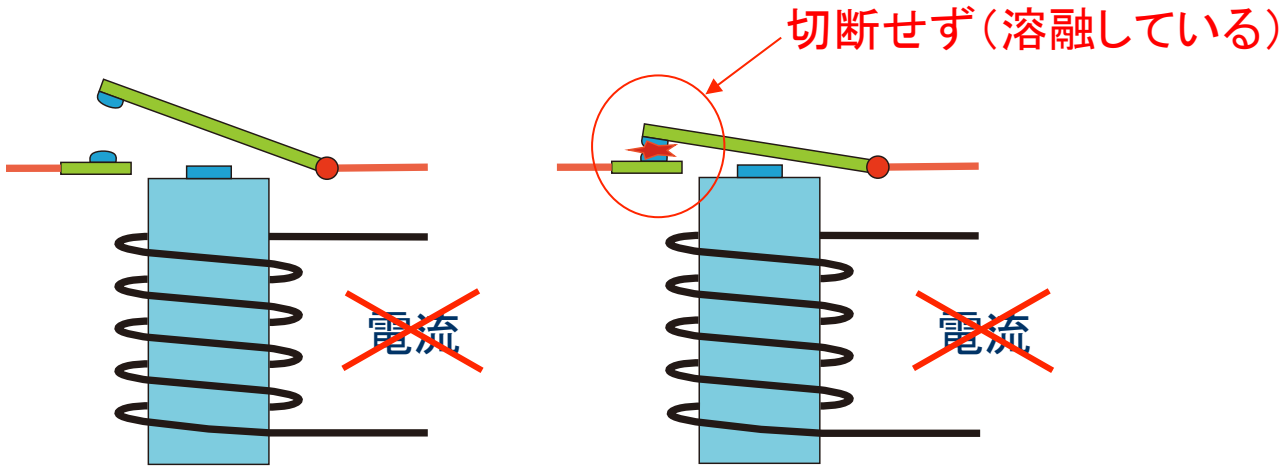
- リレーの接点の例
  - どのような故障が起こるか
    - 開放故障



正常時　　開放故障時

冗長性

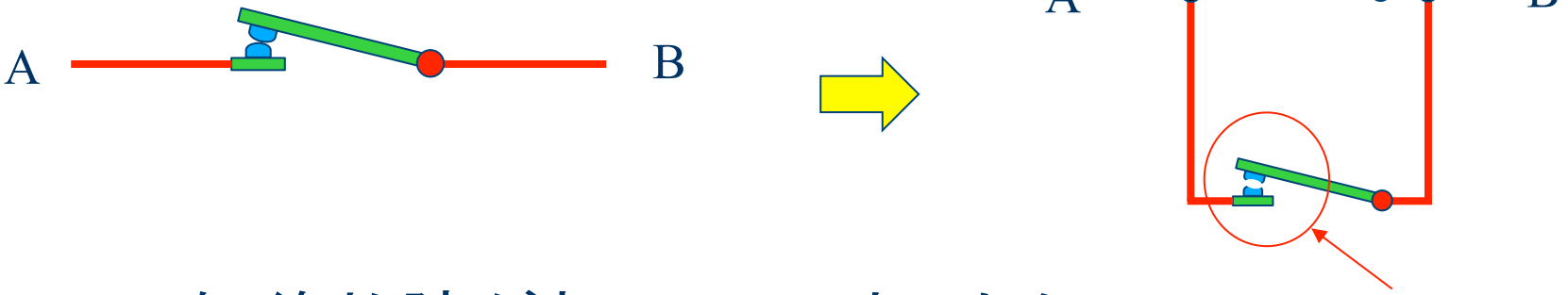
- リレーの接点の例
  - どのような故障が起こるか
    - 短絡故障



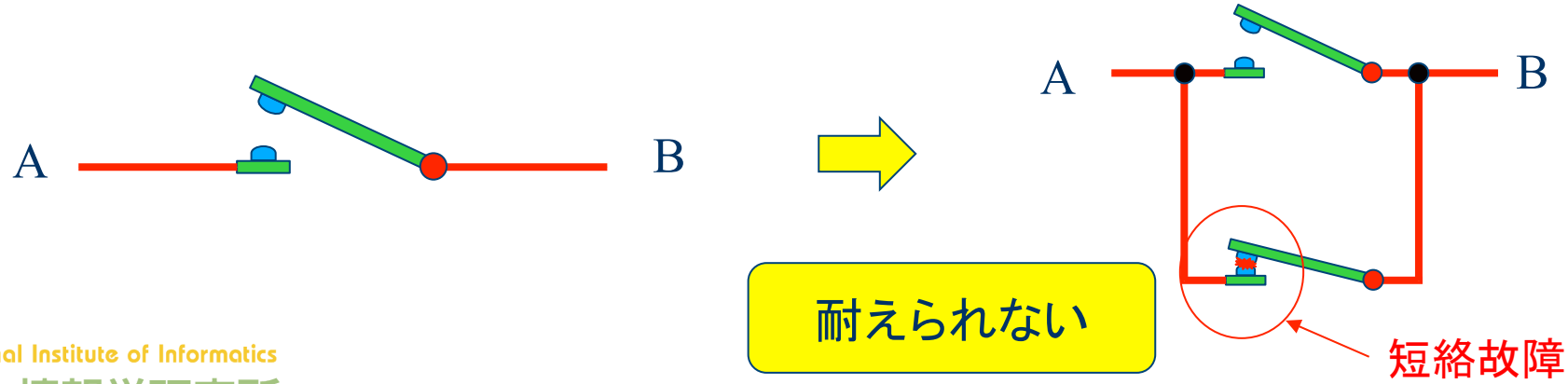
正常時　　短絡故障時

冗長性

- 冗長度＝2の場合
  - 開放故障に耐えるには

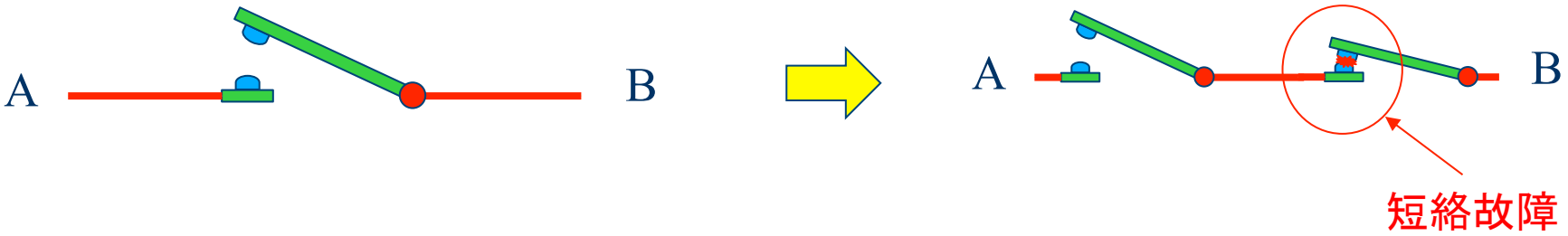


  - 短絡故障が起こってしまったら

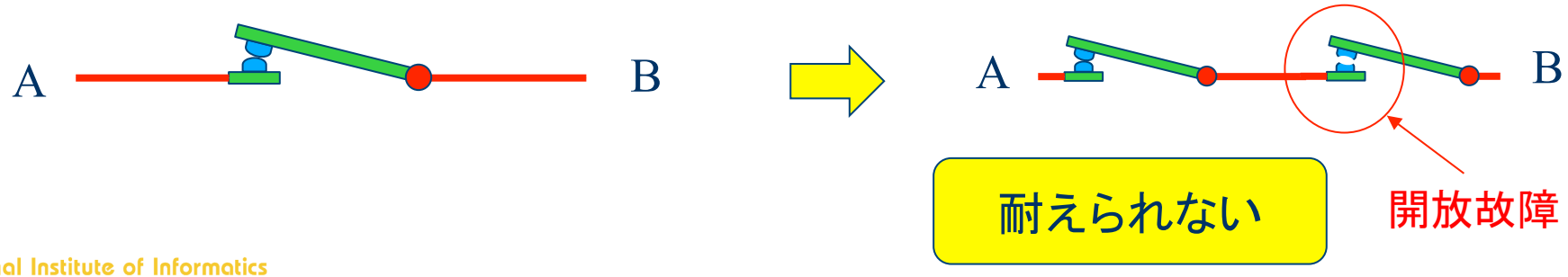


冗長性

- 冗長度＝2の場合
  - 短絡故障に耐えるには

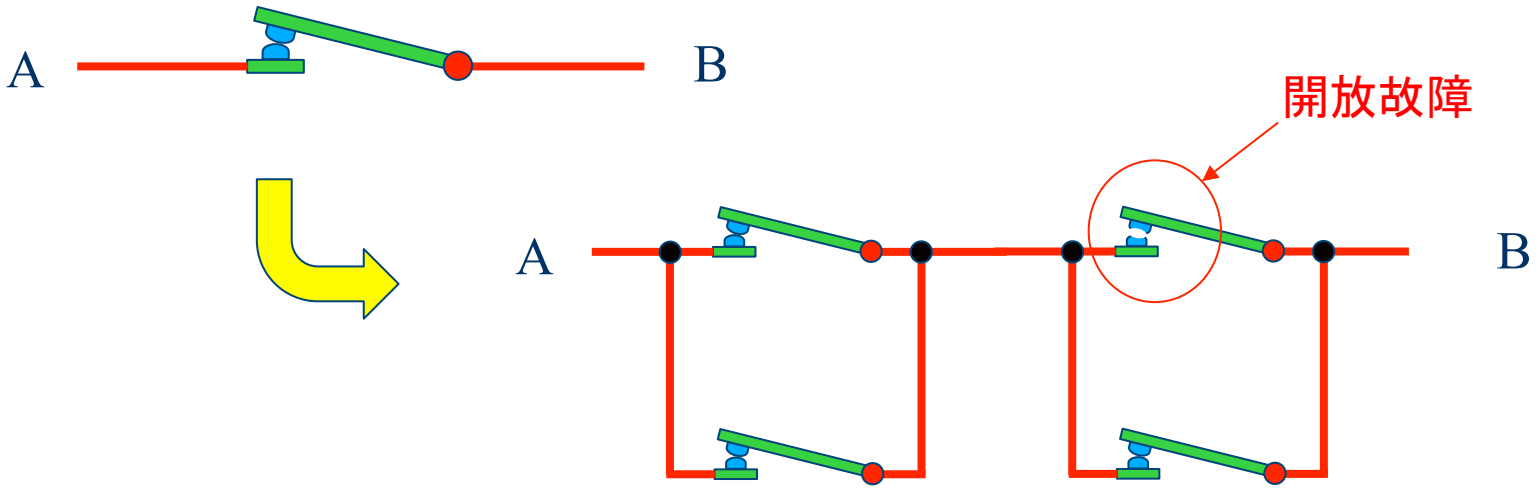


  - 開放故障が起こってしまったら



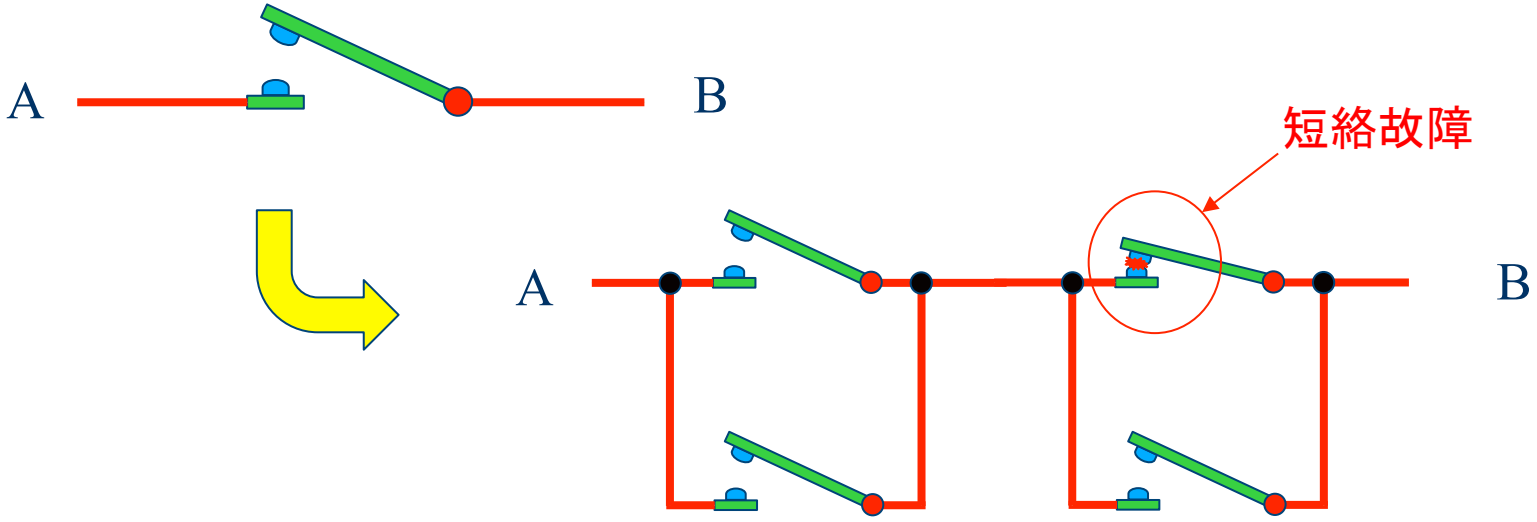
冗長性

- 開放故障と短絡故障のいずれにも耐えるには
  - 冗長度＝4とする
  - 開放故障が起こっても



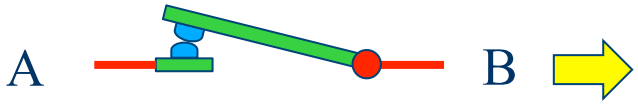
冗長性

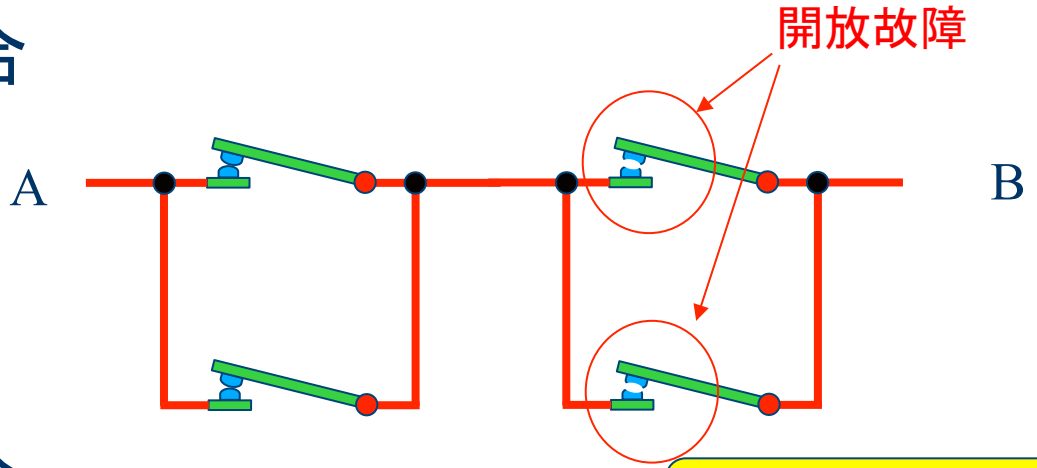
- 開放故障と短絡故障のいずれにも耐えるには
  - 冗長度＝4とする
  - 短絡故障が起こっても



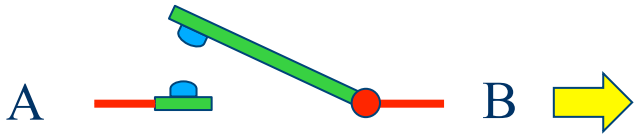
冗長性

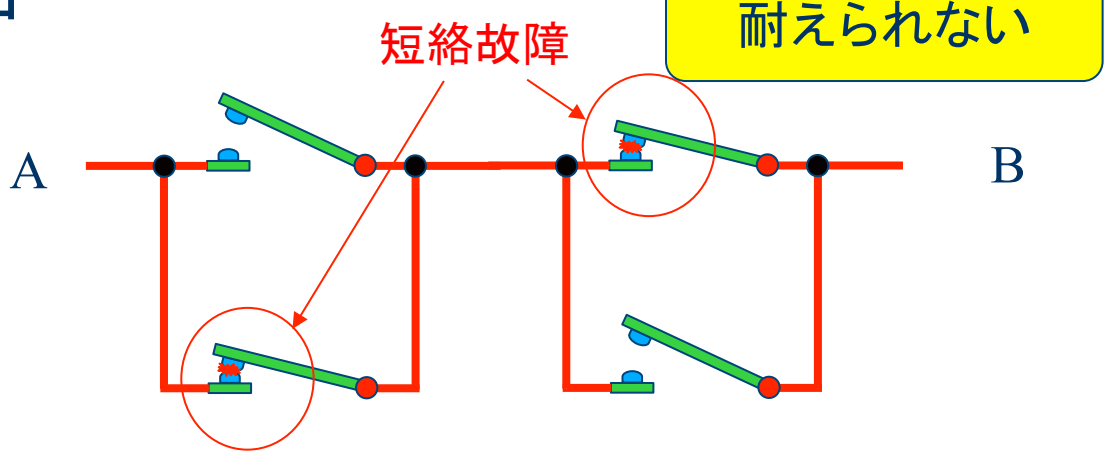
- 故障が2箇所に起こったら，．．．
  - 開放故障の場合





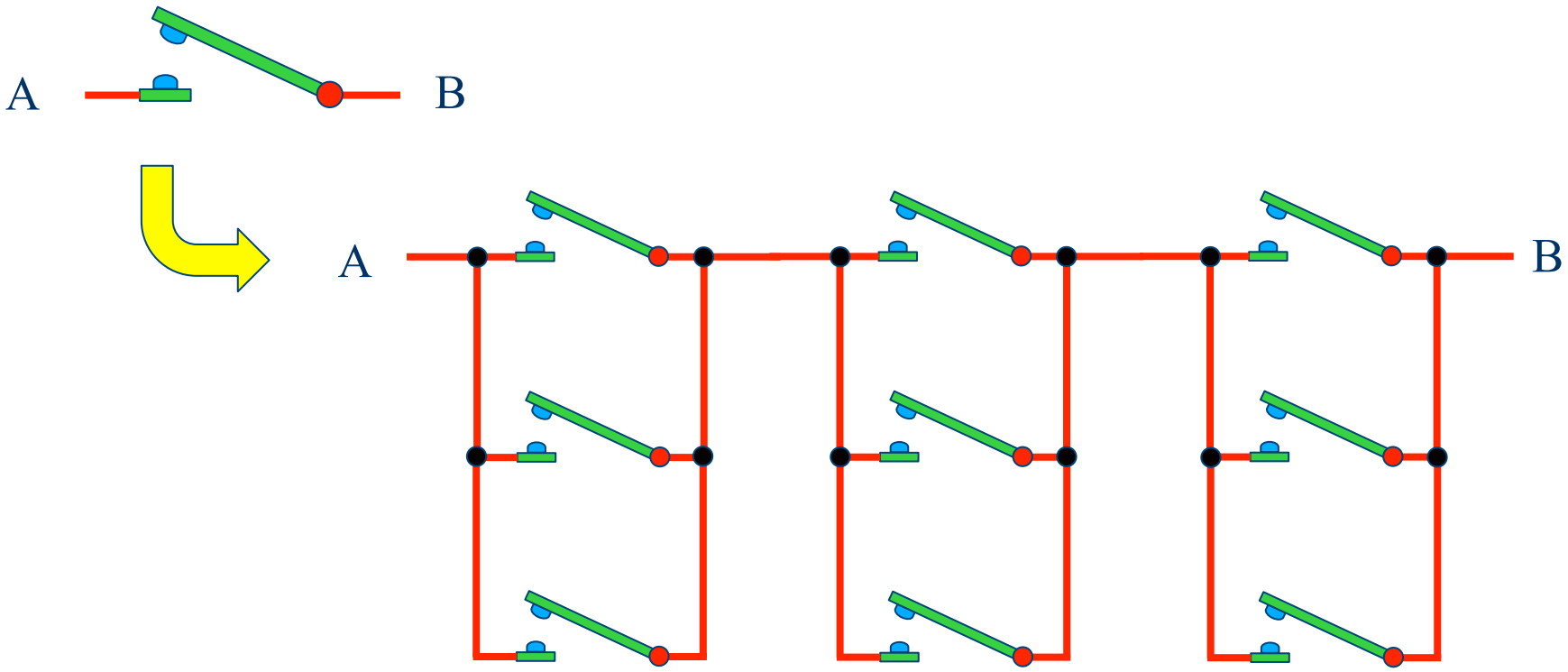
  - 短絡故障の場合





冗長性

- ２箇所の故障に耐えるには



冗長性

- 対象とする故障を決めないと耐故障設計は不可能
  - 想定外の故障に耐えるのはムリ！

ポイント3

- 冗長度を大きくする ⇒ 悪くなることもある

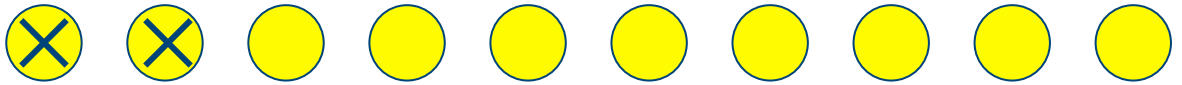
冗長性

- 疑問
  - たくさんリレーを使えば，故障もたくさん起こるのでは？
  - （例）2つのリレーに故障が起こる確率
    - 3つのリレーを使っている場合

1つのリレーが1年間に1回故障する確率(f)＝0.1

$${}_3\mathrm{C}_2 f^2(1-f) = 3f^2(1-f) = 0.027$$

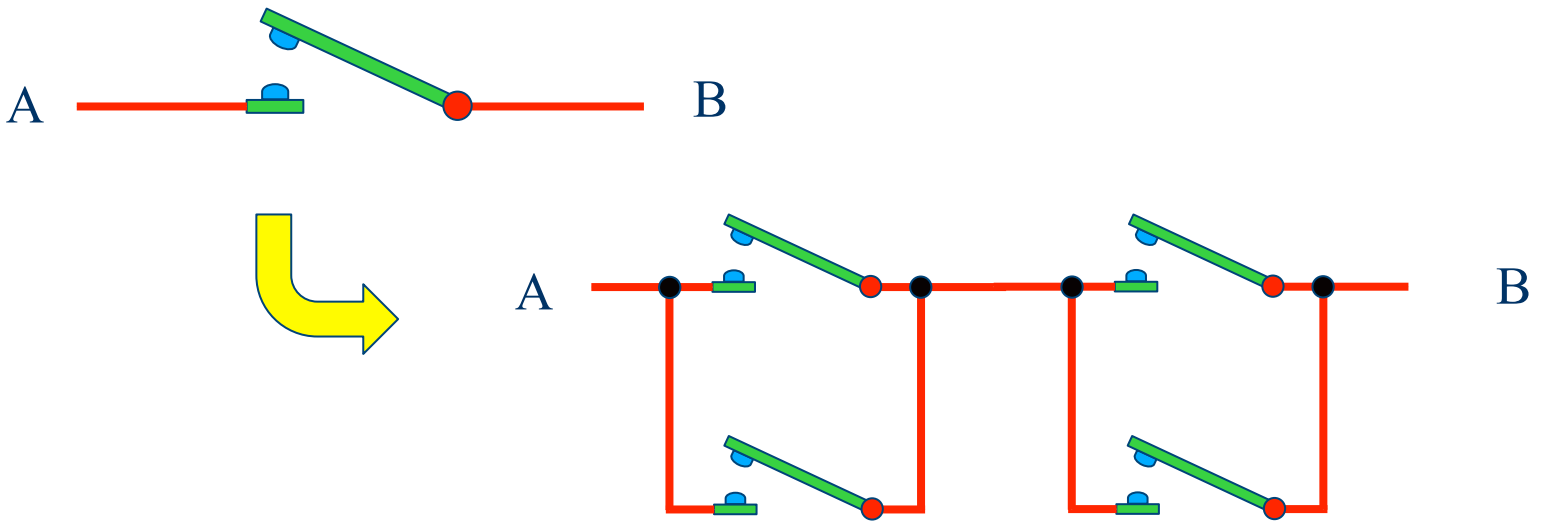
- 10個のリレーを使っている場合



$${}_{10}\mathrm{C}_2 f^2(1-f)^8 = 45f^2(1-f)^8 = 0.1937$$

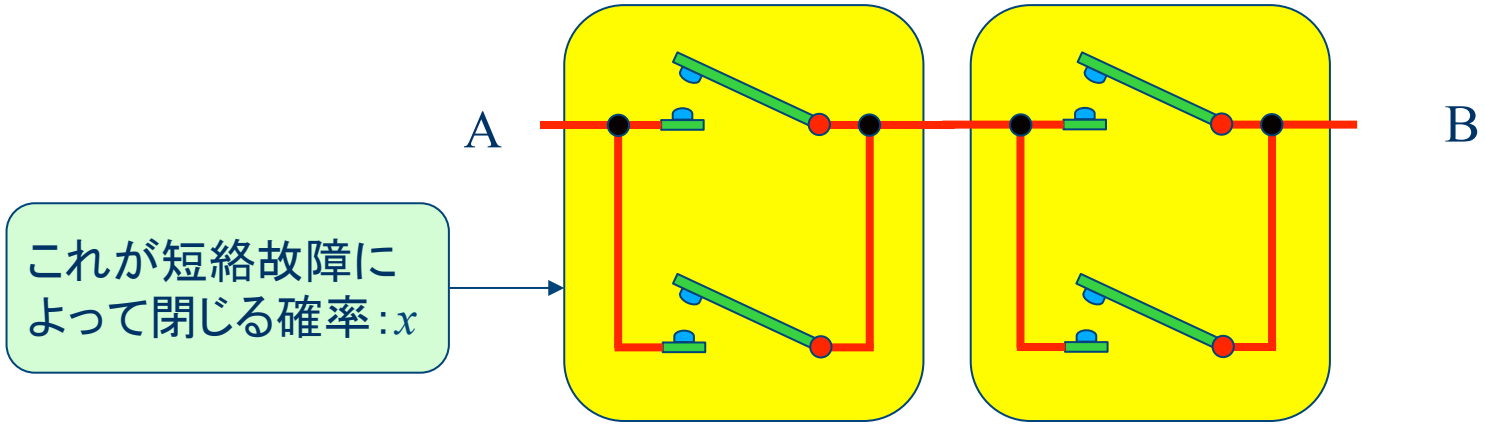
冗長性

- ほんとうに故障に強くなっているのか？
  - ε_{short} ：1つの接点が短絡故障する確率
  - P_{short} ：全体（A-B間）が短絡する確率



冗長性

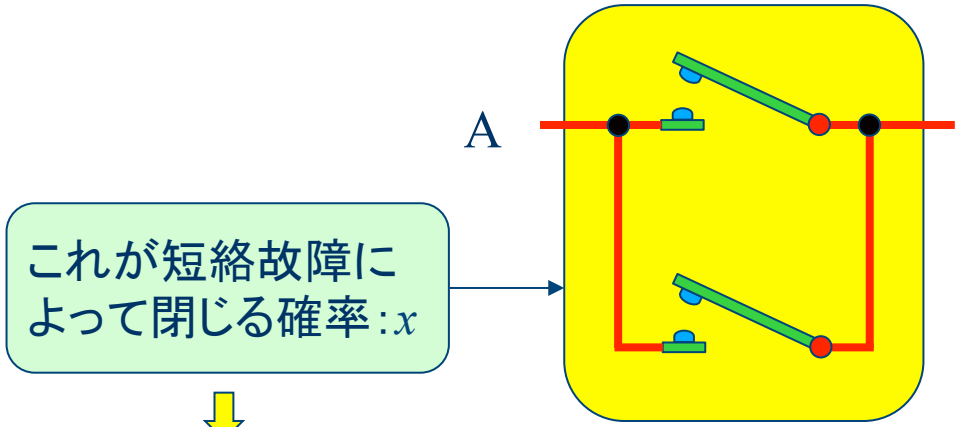
- ほんとうに故障に強くなっているのか？
  - ε_{short} ：1つの接点が短絡故障する確率
  - P_{short} ：全体（A-B間）が短絡する確率



$$P_{short} = x^2$$

冗長性

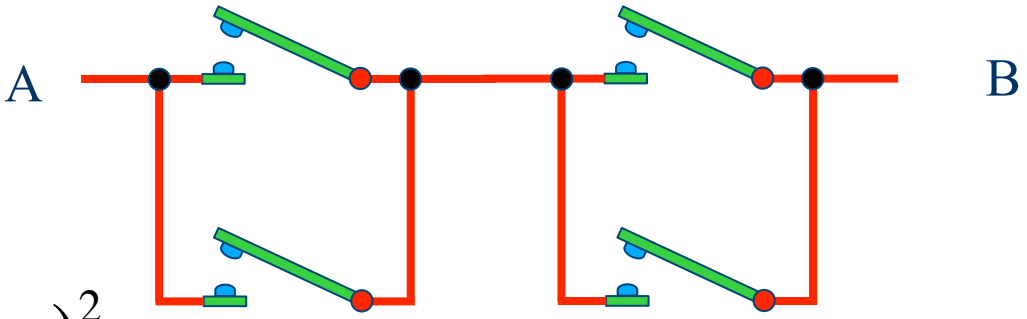
- ほんとうに故障に強くなっているのか？
  - ε_{short} ：1つの接点が短絡故障する確率
  - P_{short} ：全体（A-B間）が短絡する確率



$$1 - x = (1 - \varepsilon_{short})^2$$

冗長性

- ほんとうに故障に強くなっているのか？
  - ε_{short} ：1つの接点が短絡故障する確率
  - P_{short} ：全体（A-B間）が短絡する確率



$$P_{short} = x^2$$

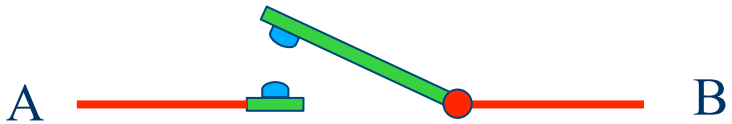
$$1 - x = (1 - \varepsilon_{short})^2$$

$$P_{short} = (1 - (1 - \varepsilon_{short})^2)^2$$

$$= \varepsilon_{short}^4 - 4\varepsilon_{short}^3 + 4\varepsilon_{short}^2$$

冗長性

- ほんとうに故障に強くなっているのか？
  - ε_{short} ：1つの接点が短絡故障する確率
  - P_{short} ：全体（A-B間）が短絡する確率

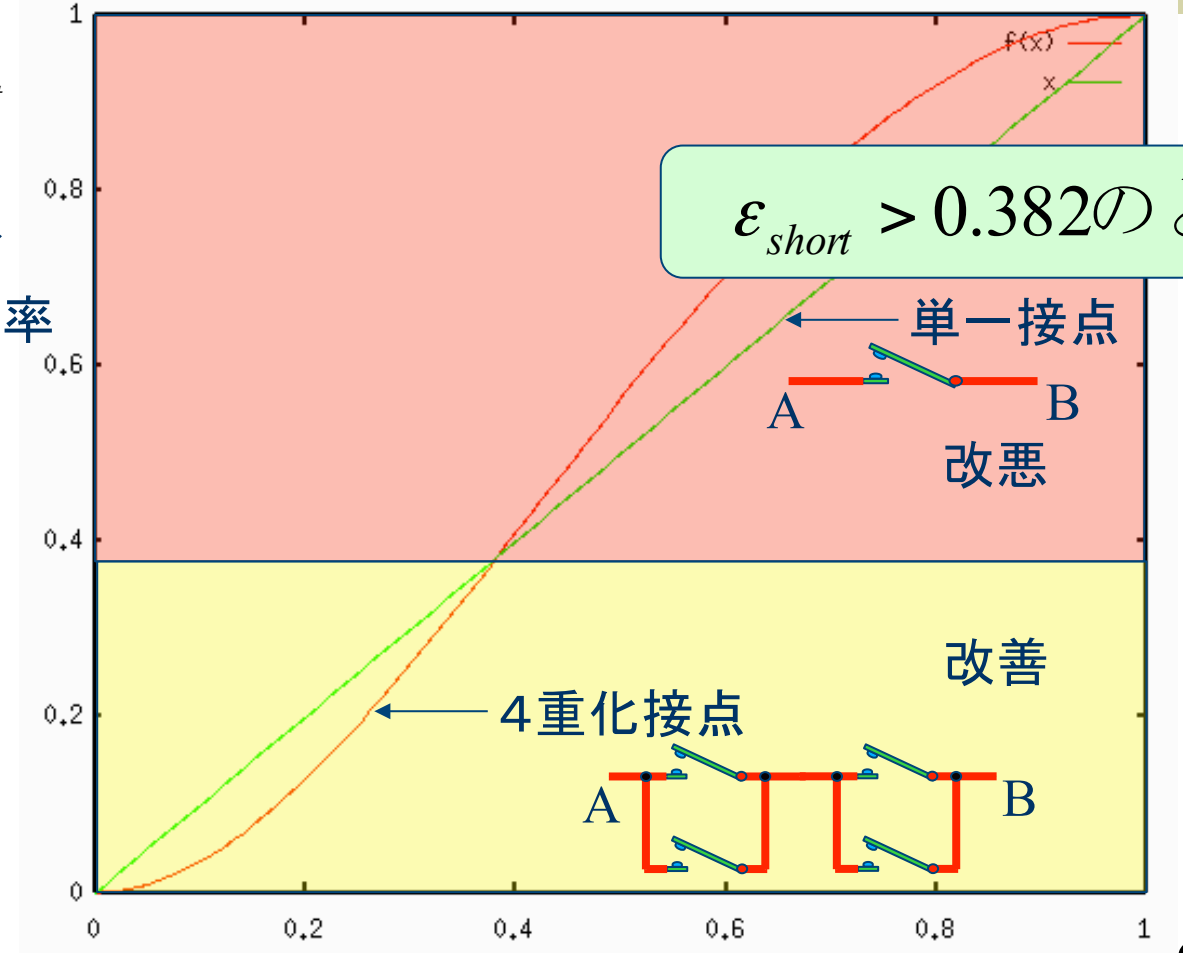


$$P_{short} = \varepsilon_{short}$$

冗長性

P_{short}

全体（A–B間）が
短絡故障する確率



ε_{short}

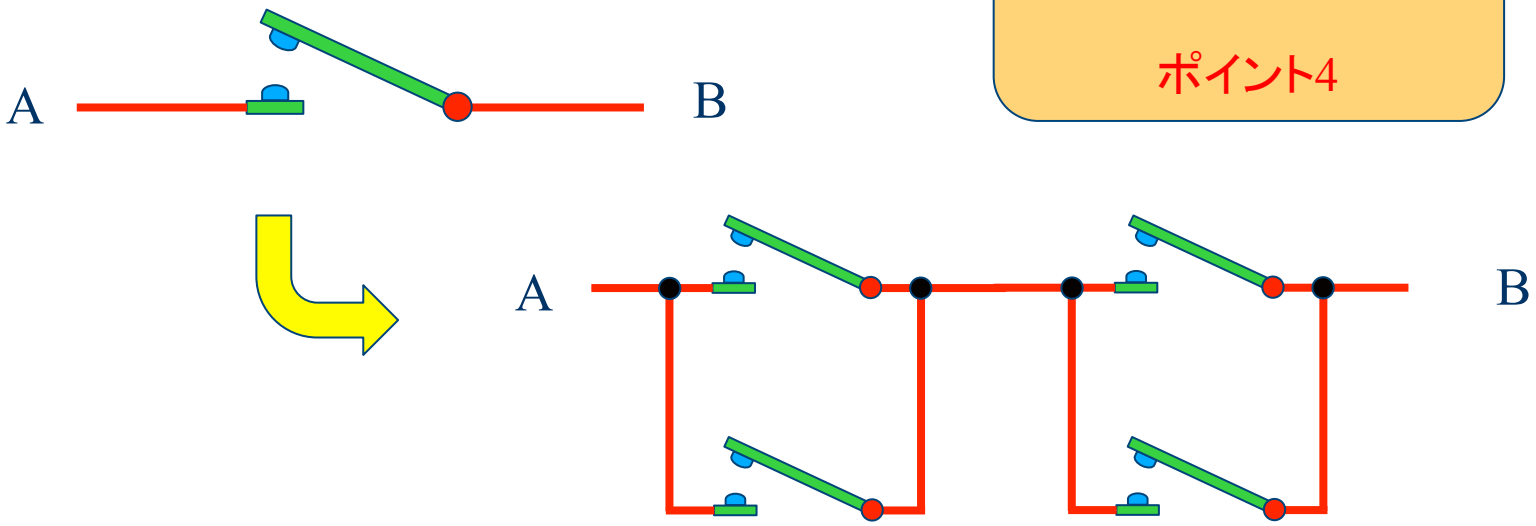
1つの接点が短絡故障する確率

冗長性

- ほんとうに故障に強くなっているのか？
  - （1つの接点が短絡故障する確率）＜0.382の場合に限って良くなる

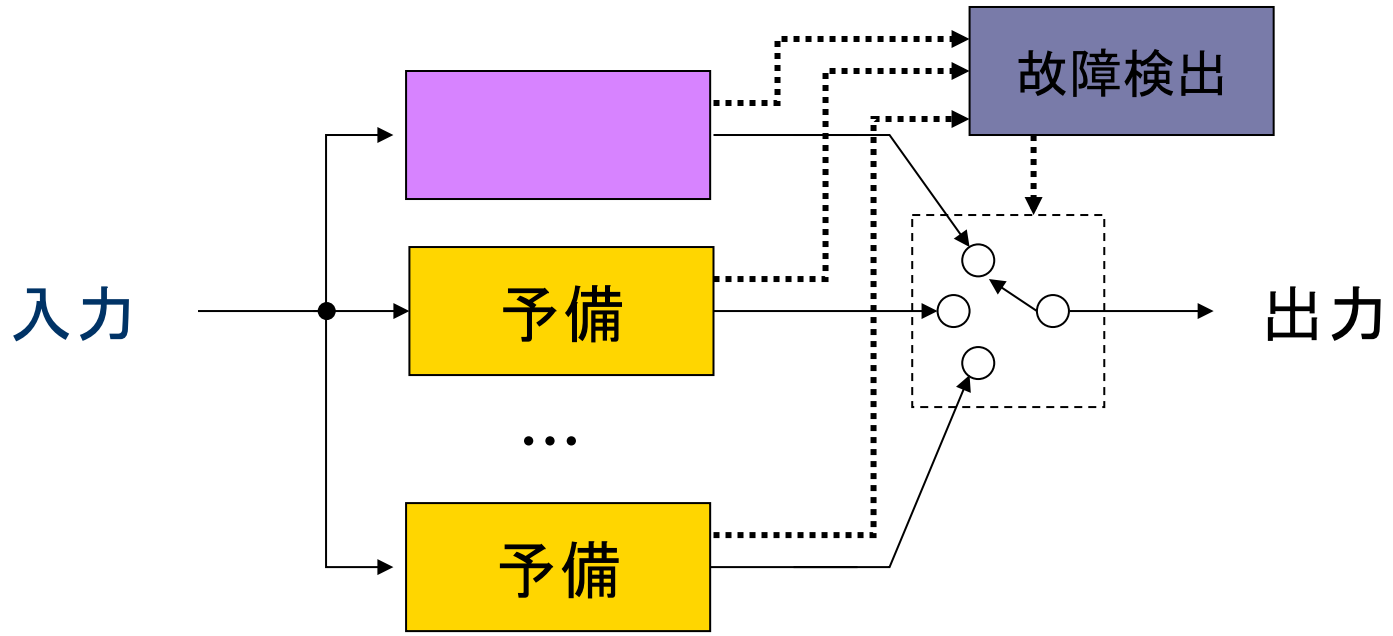
悪いものを冗長化
ても良くならない

ポイント4



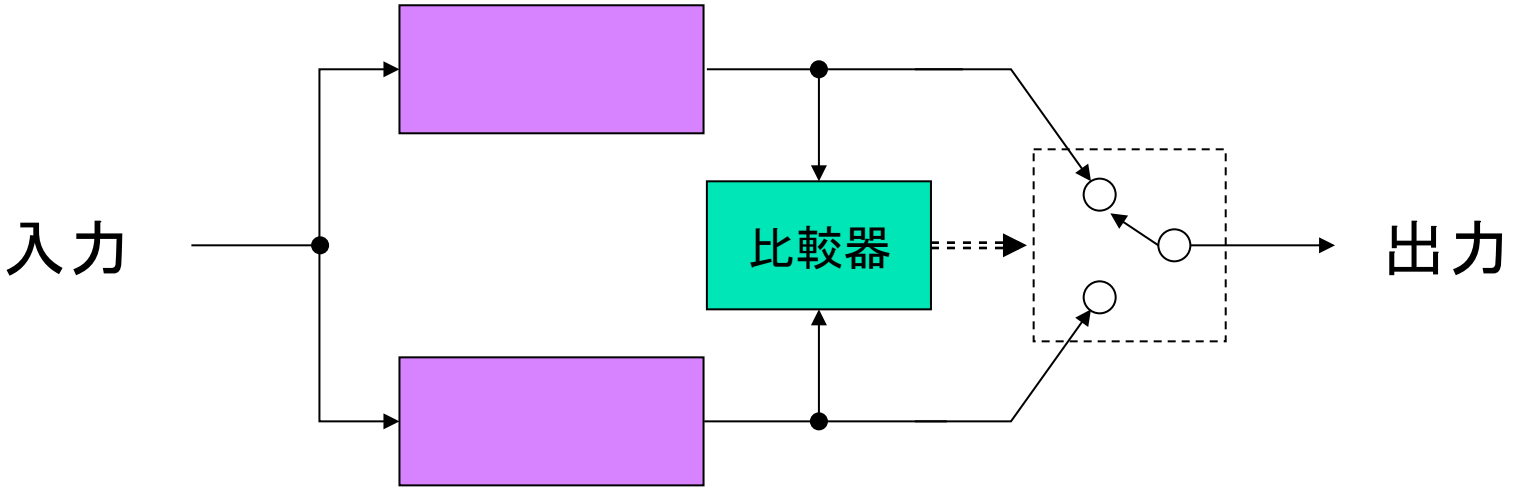
冗長性

- コンピュータシステムの場合（1）



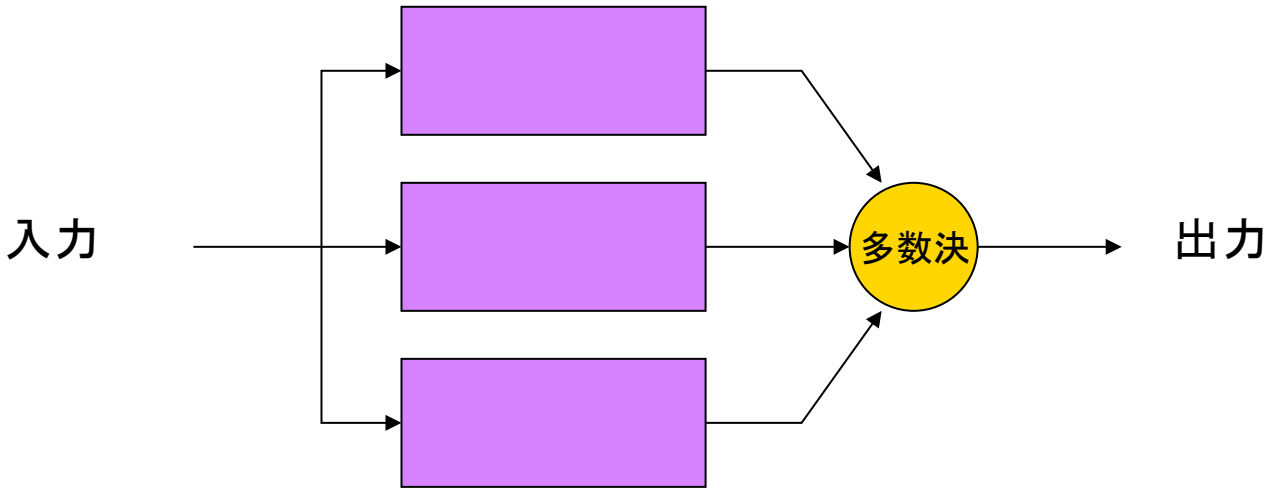
冗長性

- コンピュータシステムの場合（2）



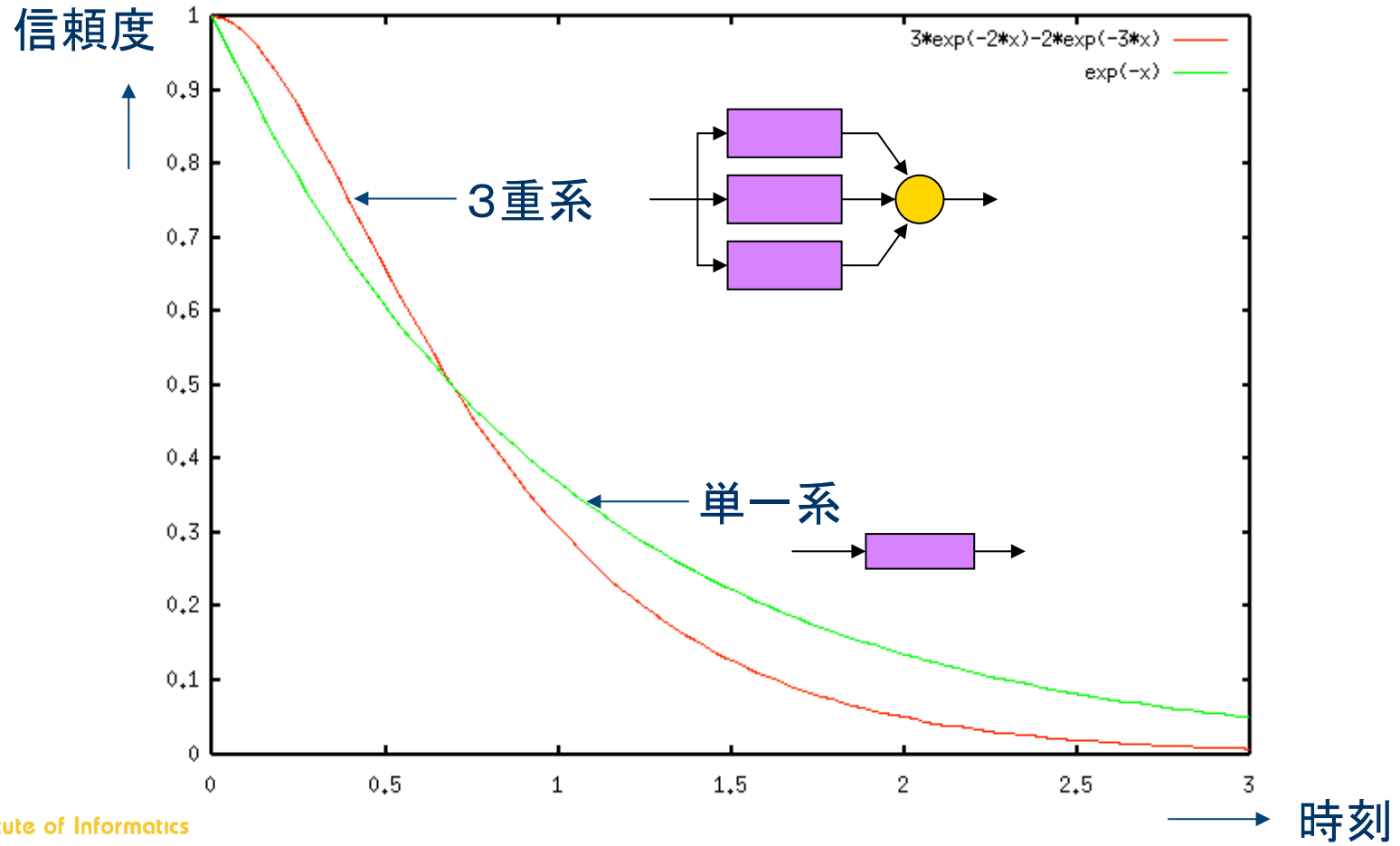
冗長性

- コンピュータシステムの場合（3）



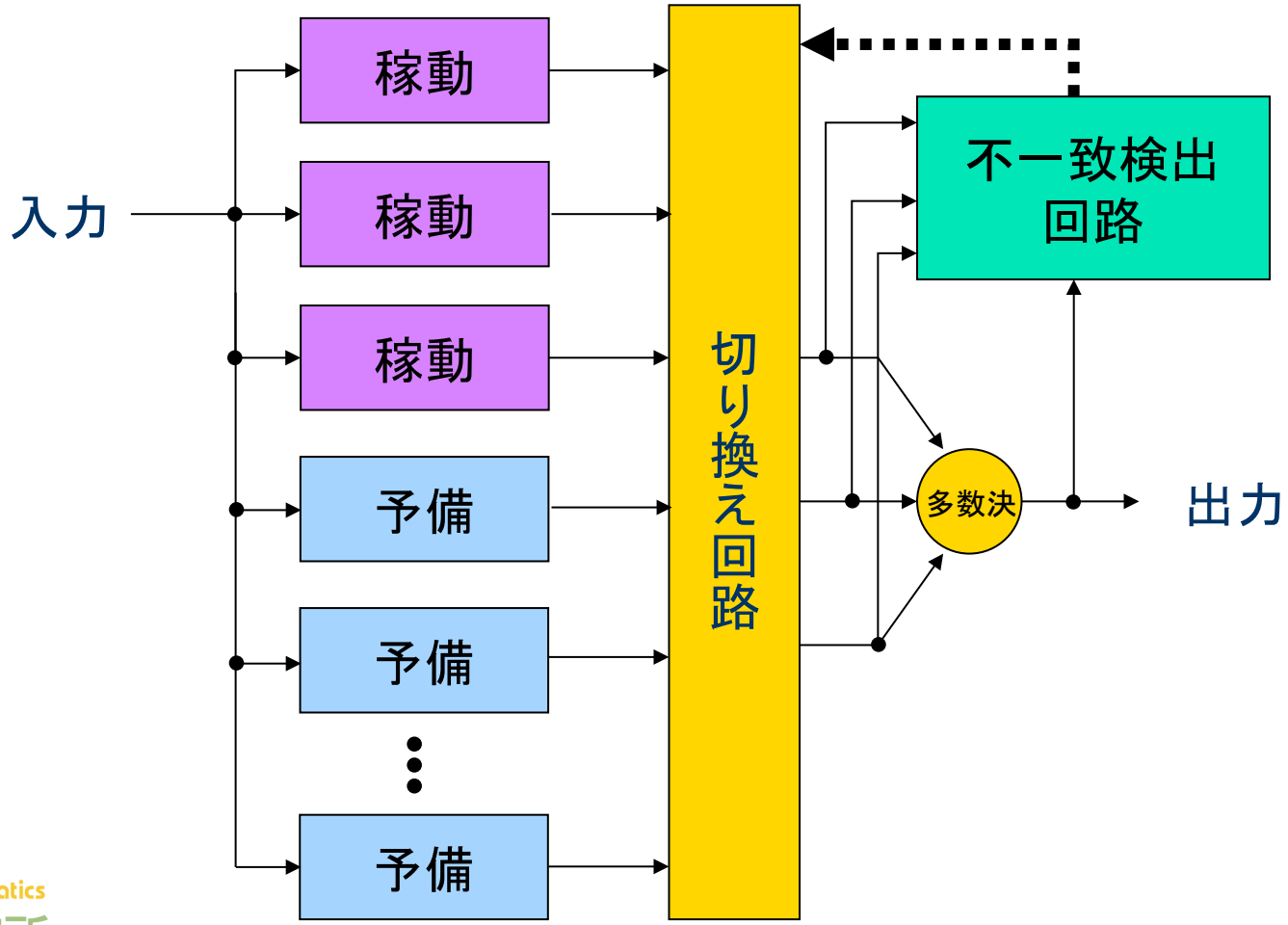
冗長性

- コンピュータシステムの場合（3）



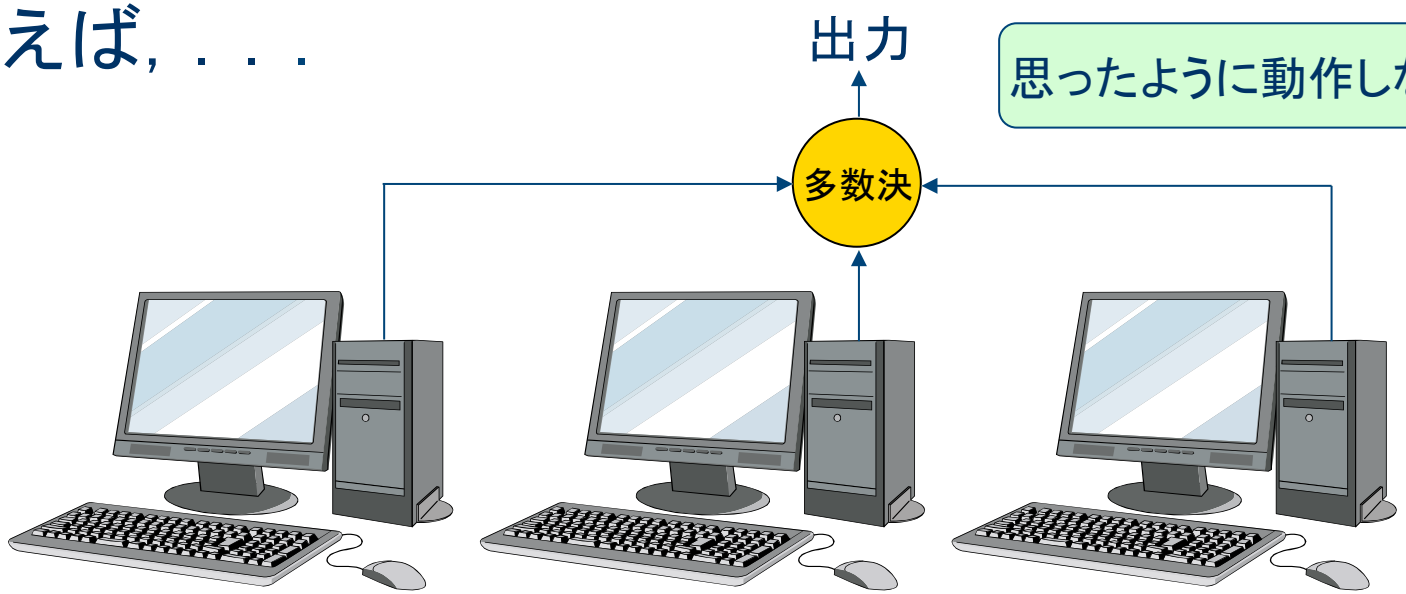
冗長性

- コンピュータシステムの場合（4）



冗長性

- 多数決を取るのは簡単ではない！
  - 例えば，．．．



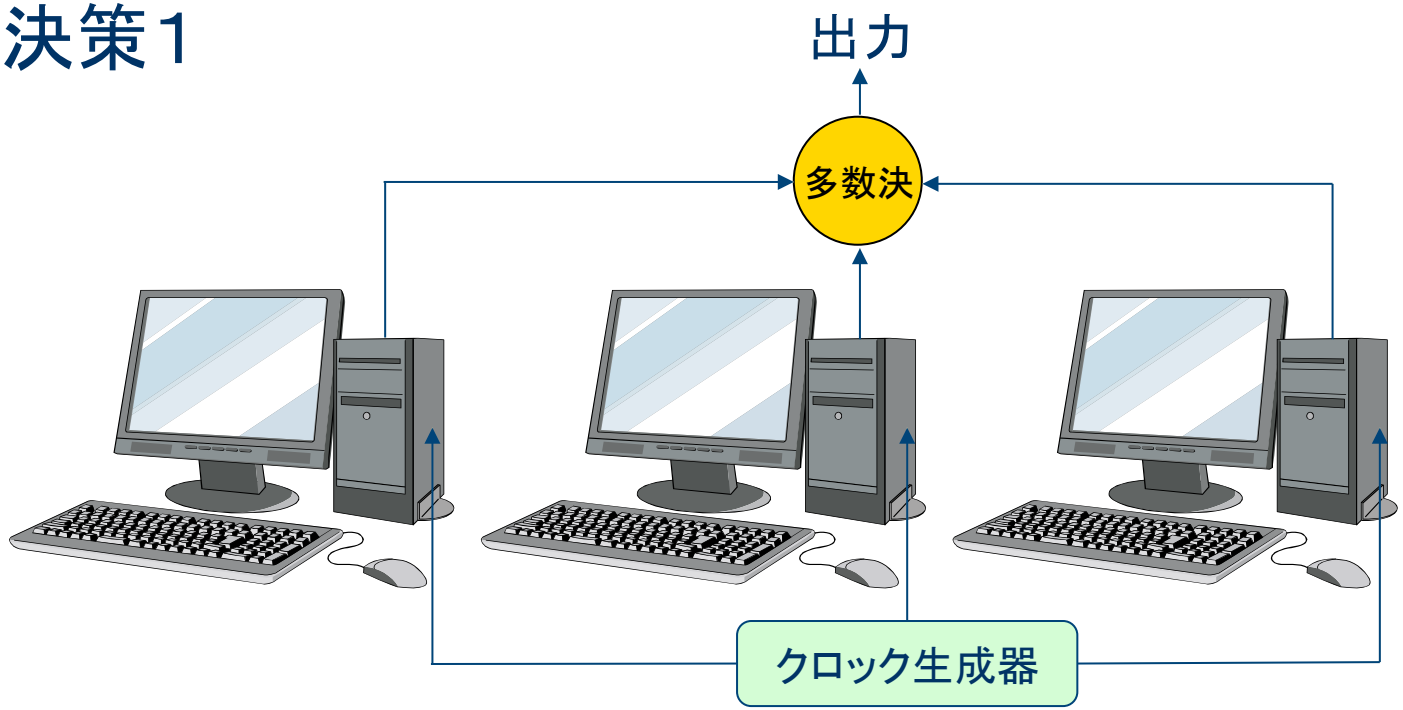
思ったように動作しない

  - だんだん処理がずれていく
    - クロックが正確に同じではない
    - ディスクアクセスなどでタイミングが狂う

同期という問題

冗長性

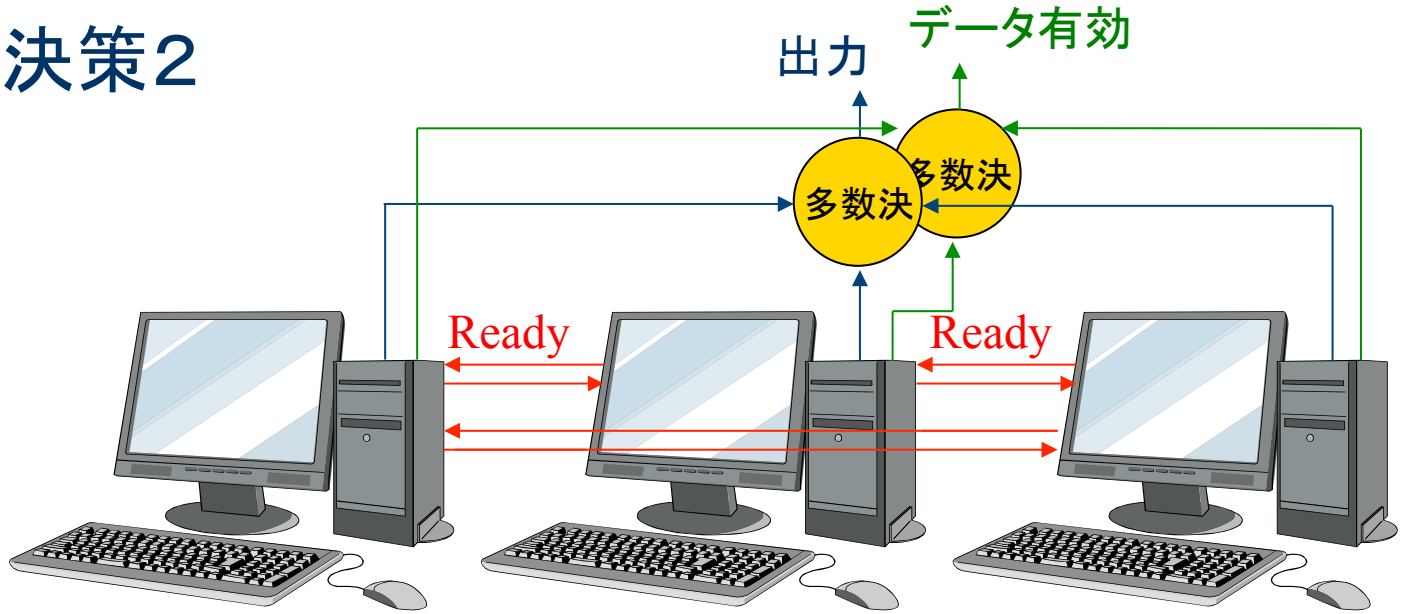
- 多数決を取るのは簡単ではない！
  - 解決策1



- 共通のクロック生成器を使う → 故障に弱い
- 特別なOSを使う

冗長性

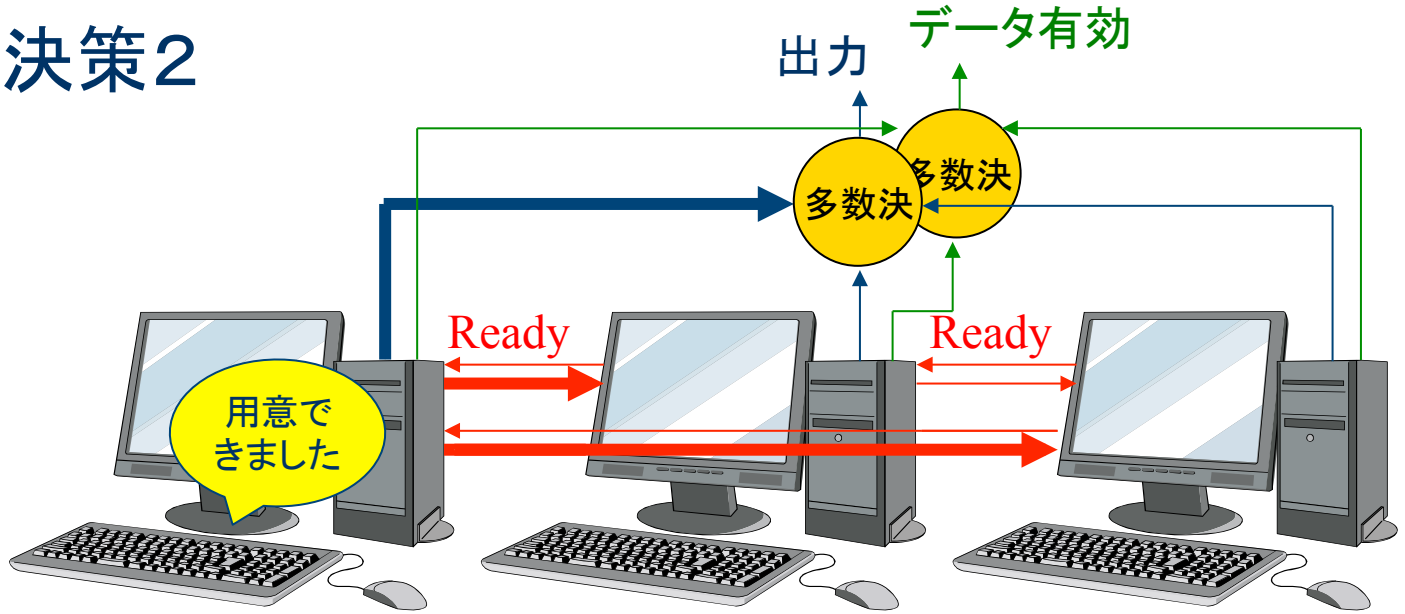
- 多数決を取るのは簡単ではない！
  - 解決策2



- 重要な出力を出すタイミングだけ合わせる
  - ゆるい同期
  - 「データ有効」信号を使う

冗長性

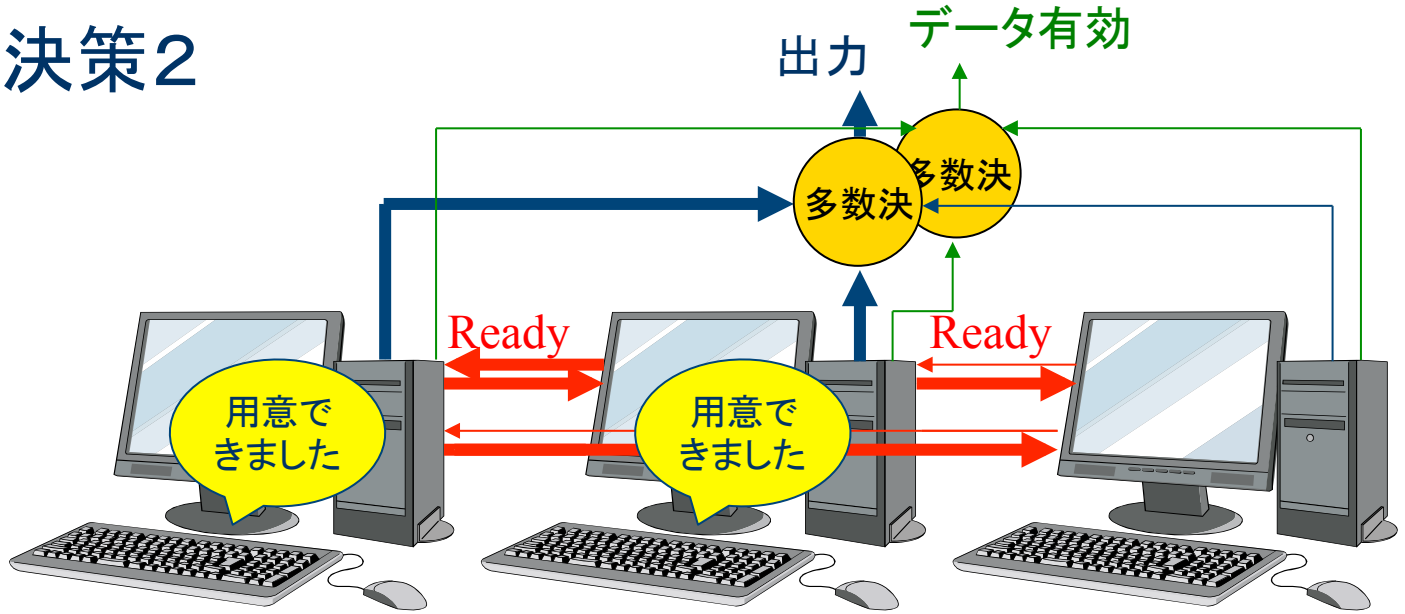
- 多数決を取るのは簡単ではない！
  - 解決策2



    - 基本的には，．．．

冗長性

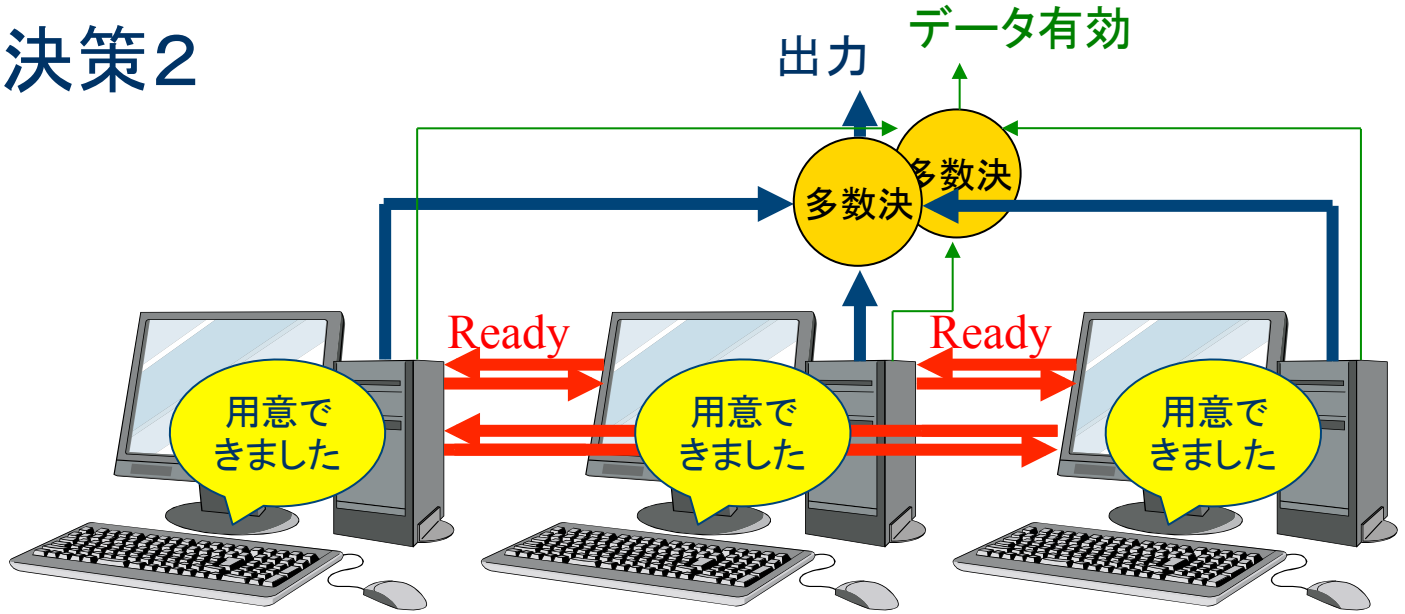
- 多数決を取るのは簡単ではない！
  - 解決策2



    - 基本的には，．．．

冗長性

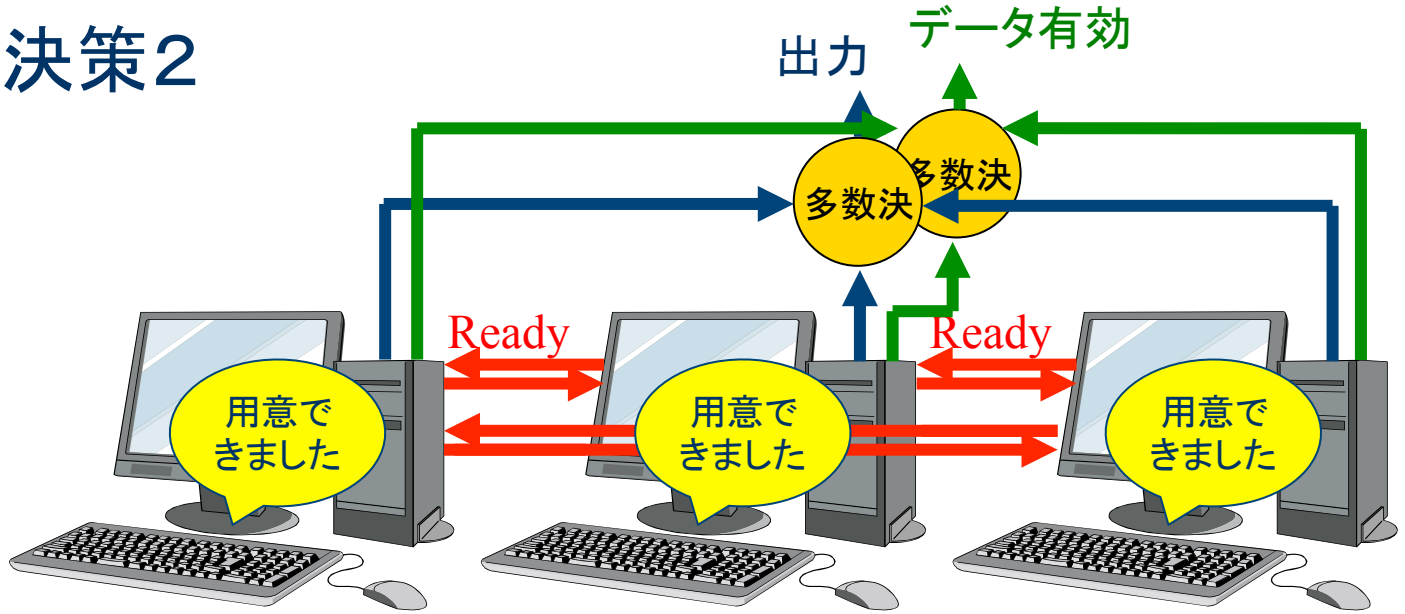
- 多数決を取るのは簡単ではない！
  - 解決策2



    - 基本的には，．．．

冗長性

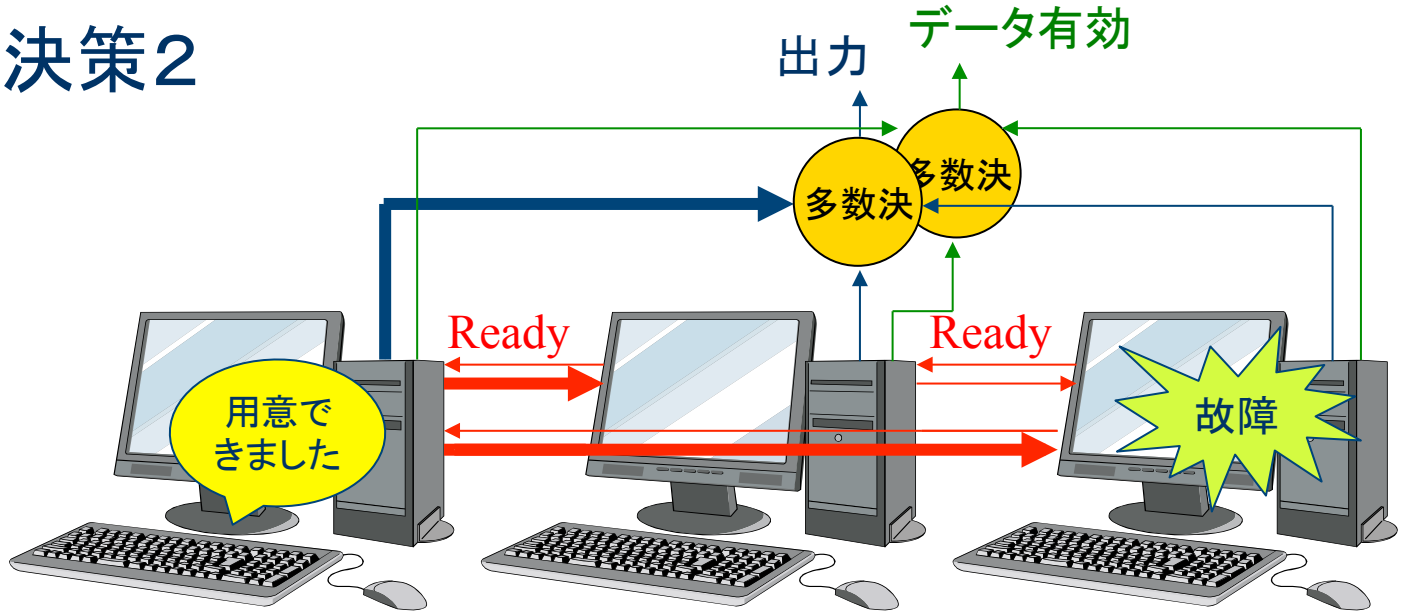
- 多数決を取るのは簡単ではない！
  - 解決策2



    - 基本的には，．．．

冗長性

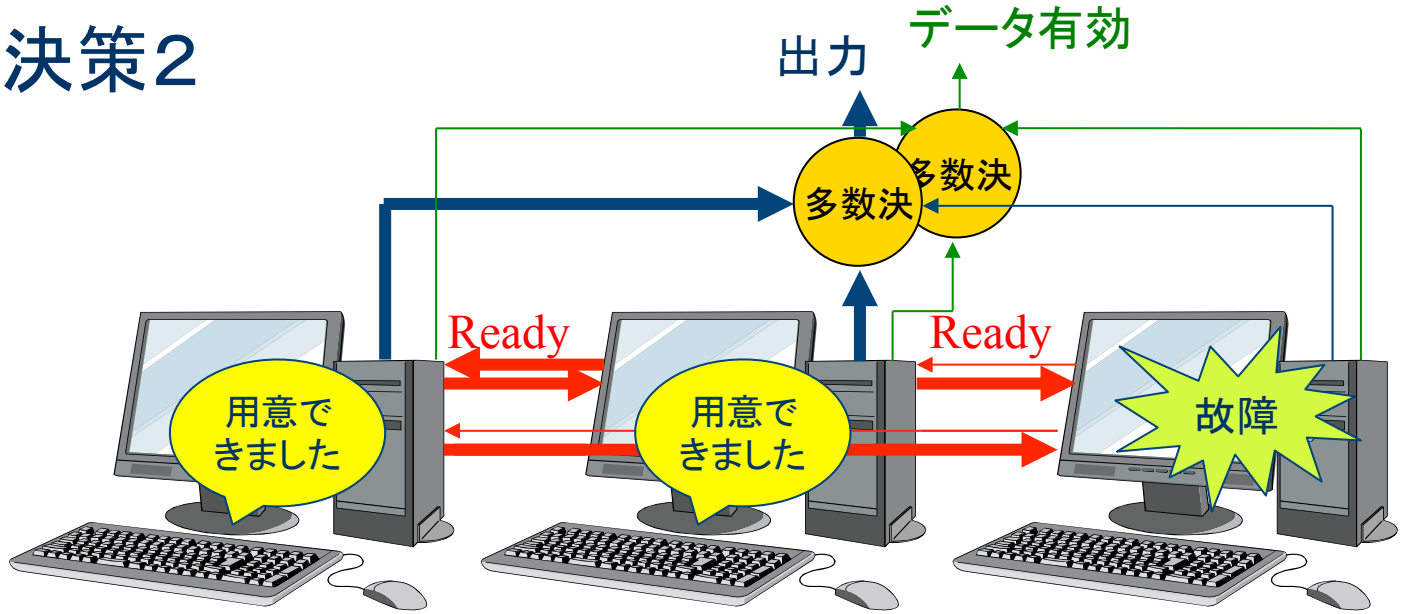
- 多数決を取るのは簡単ではない！
  - 解決策2



- コンピュータが故障していると，．．．

冗長性

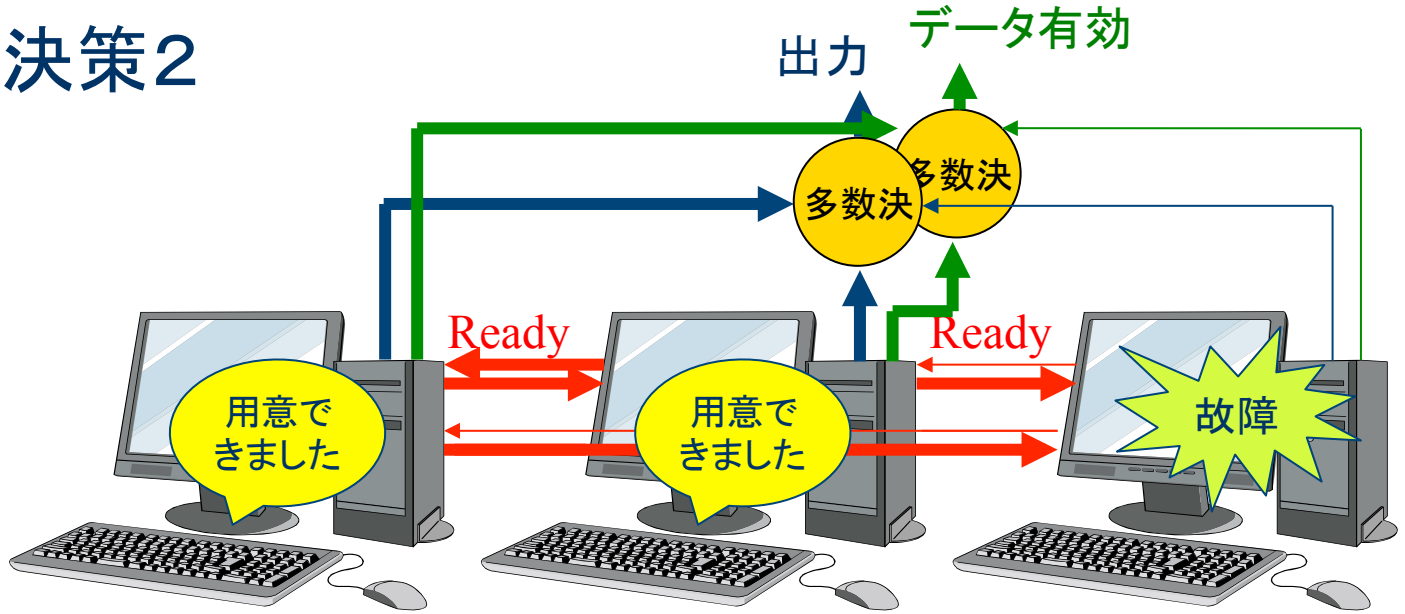
- 多数決を取るのは簡単ではない！
  - 解決策2



- コンピュータが故障していると，．．．
  - 「Ready」を出さないかもしれない

冗長性

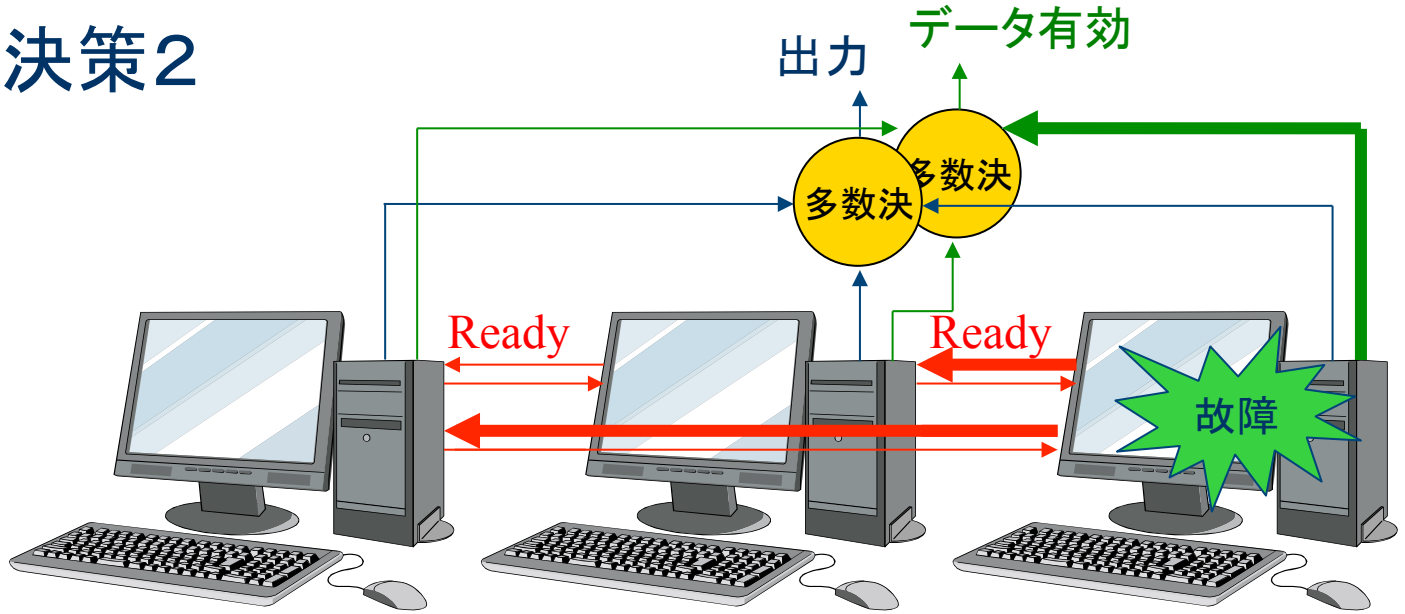
- 多数決を取るのは簡単ではない！
  - 解決策2



- コンピュータが故障していると，．．．
  - 「Ready」が2つで「データ有効」を出す必要がある

冗長性

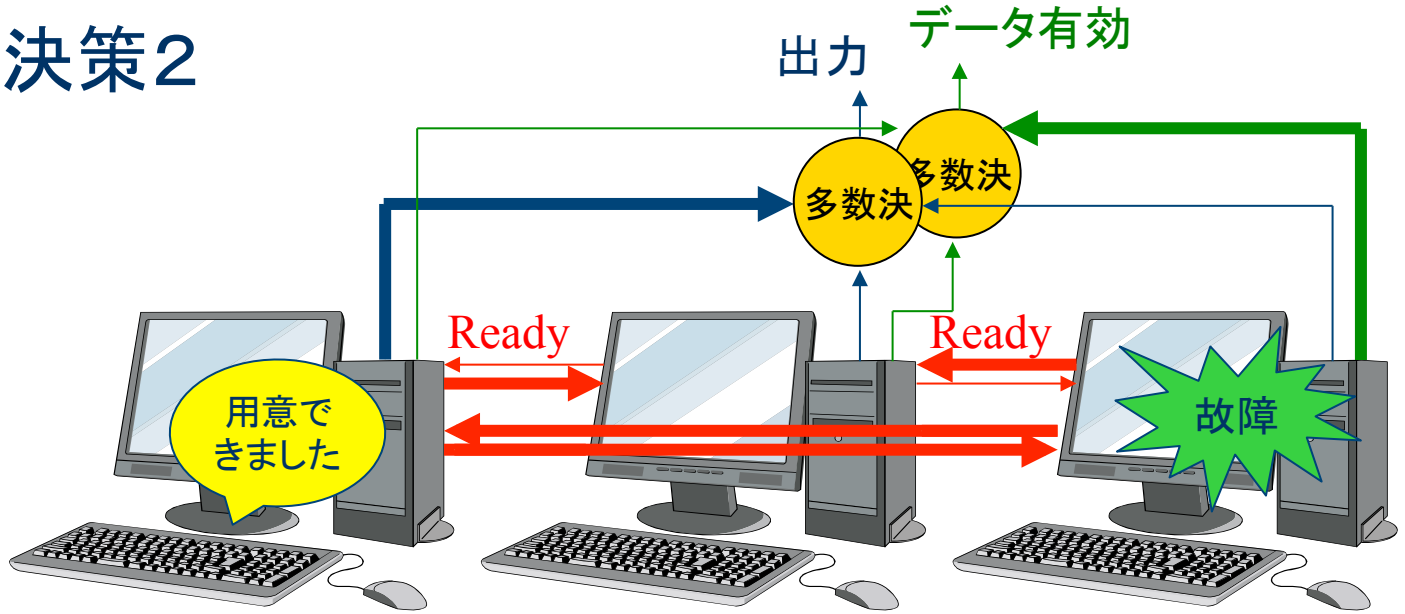
- 多数決を取るのは簡単ではない！
  - 解決策2



    - 常に「Ready」,「データ有効」を出している故障では，．．．

冗長性

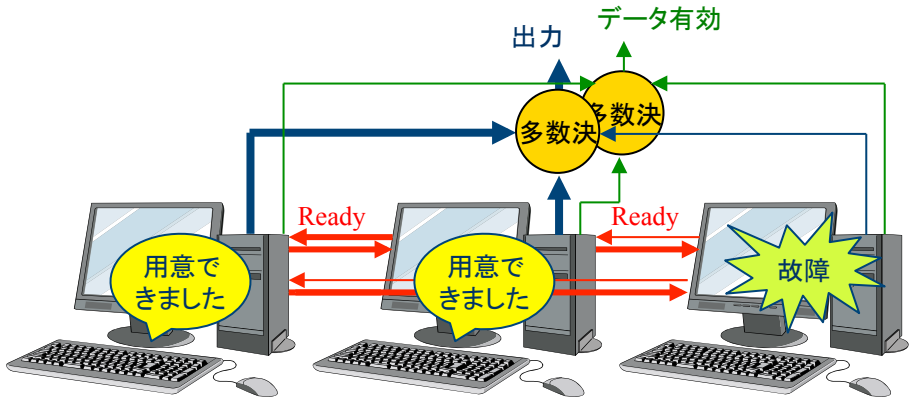
- 多数決を取るのは簡単ではない！
  - 解決策2



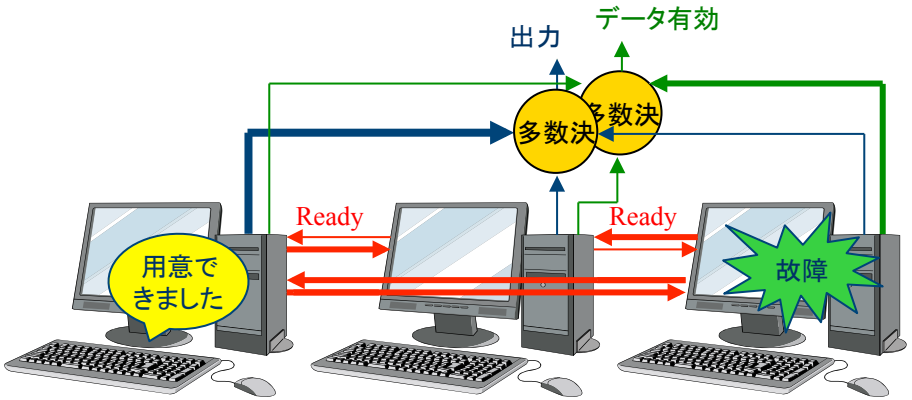
- 常に「Ready」を出している故障では，．．．
  - 「Ready」は2つ来ているが，「出力」はまだ有効ではない
    ⇒「データ有効」を出してはならない

冗長性

- 多数決を取るのは簡単ではない！
  - 解決策2



「Ready」が2つ来ている
⇒「データ有効」を出す必要がある

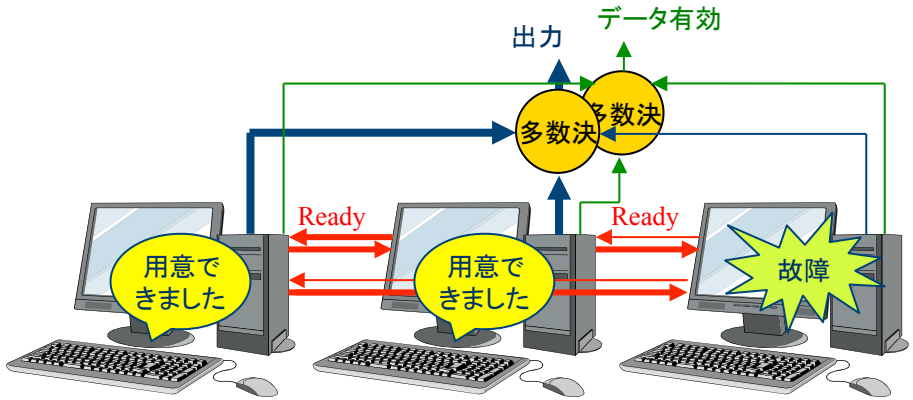


「Ready」が2つ来ている
⇒「データ有効」を出してはならない

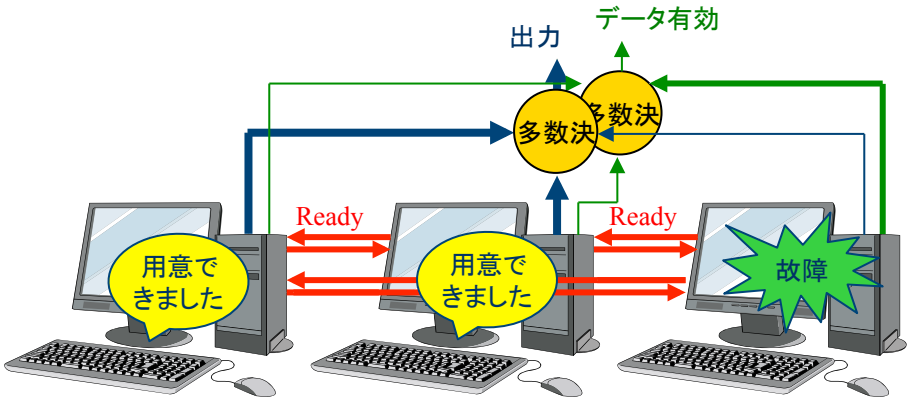
故障は1台のみ，
コンピュータはゆるく同期している
⇒ 少し待てば，もうひとつも「Ready」を出す

冗長性

- 多数決を取るのは簡単ではない！
  - 解決策2



「Ready」が2つ来ている
⇒「データ有効」を出す必要がある



「Ready」が2つ来ている
⇒「データ有効」を出してはならない

故障は1台のみ，
コンピュータはゆるく同期している
⇒ 少し待てば，もうひとつも「Ready」を出す

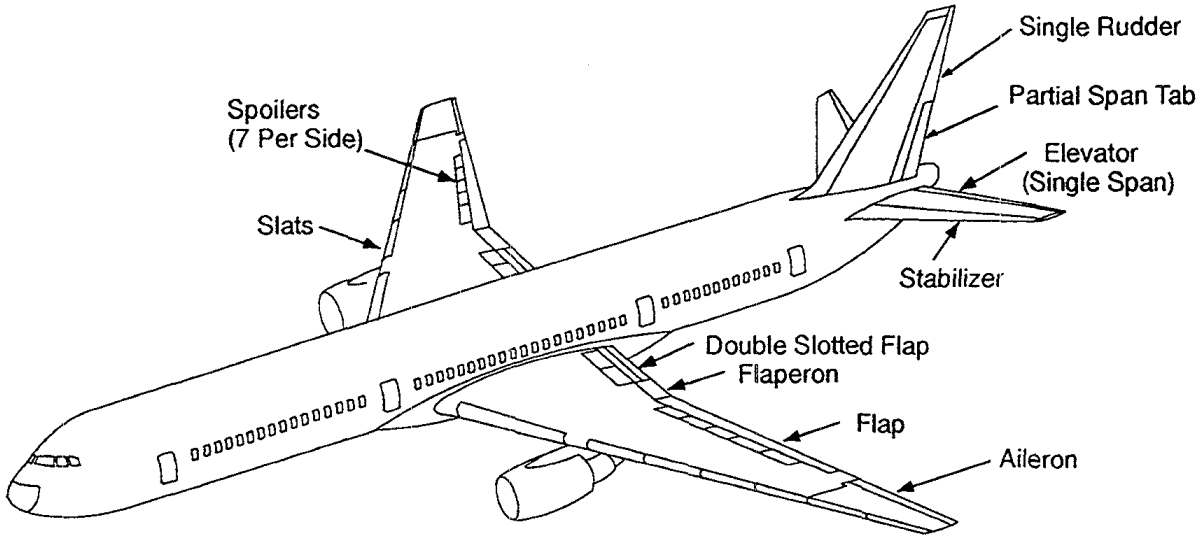
冗長性

- 多数決を取るのは簡単ではない！
  - 解決策2

- 「Ready」を（自分を含めて）2つ受け取ってから，しばらく待って「データ有効」を出す

事例紹介

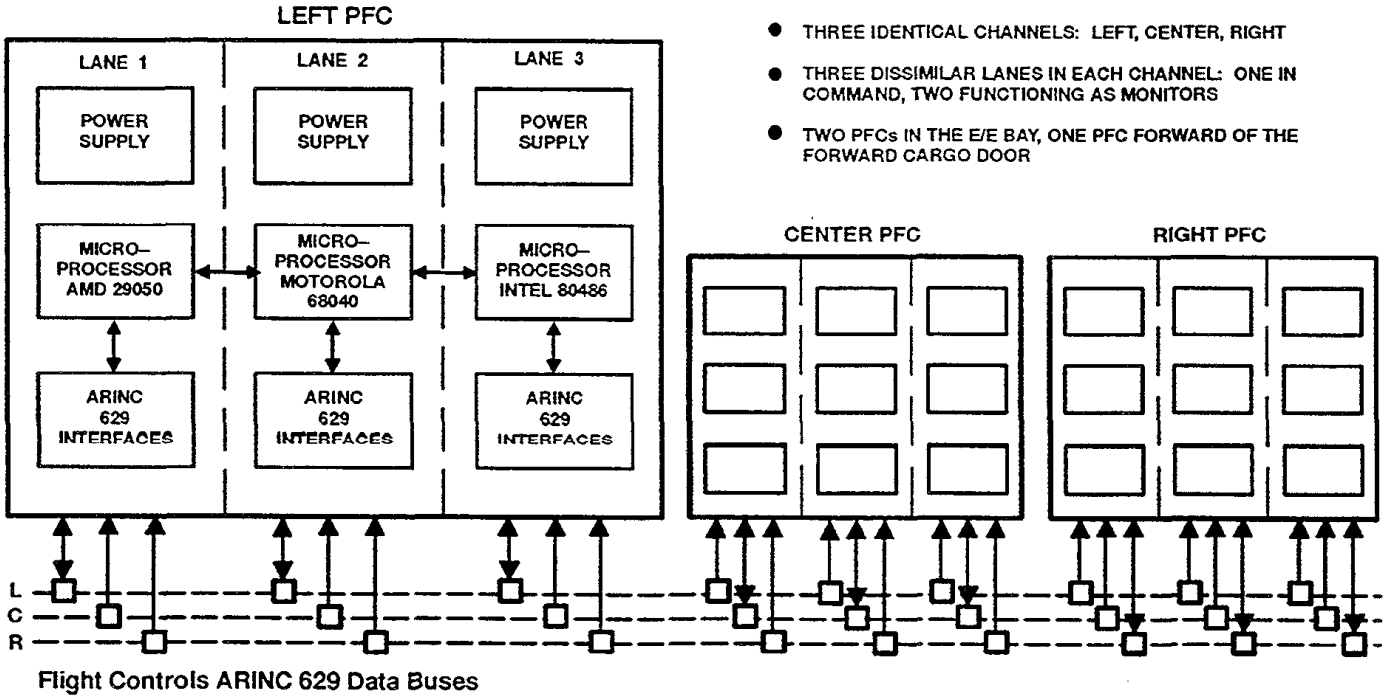
- ボーイング777



Yeh, Y.C., Triple-triple redundant 777 primary flight computer,
IEEE Aerospace Applications Conference, 1996. Proceedings., pp. 293 – 307, vol.1, 1996.

事例紹介

- ボーイング777
  - 3台のプライマリフライト制御システム（PFC）



Yeh, Y.C., Triple-triple redundant 777 primary flight computer,
IEEE Aerospace Applications Conference, 1996. Proceedings., pp. 293 – 307, vol.1, 1996.

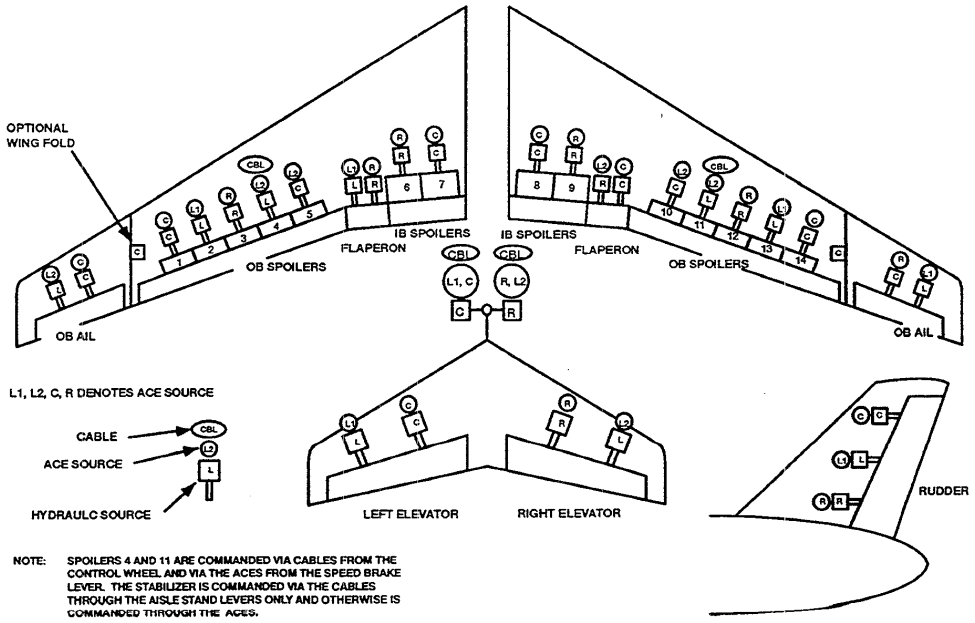
事例紹介

- ボーイング777
  - 3台のプライマリフライト制御システム（PFC）
    - 各プライマリフライト制御システムは3重化
      - 一つの計算要素がダウンしても，プライマリフライト制御システムは正常に処理を続行可
    - 1台のプライマリフライト制御システムが完全にダウンしても処理を続行可
      - 電源故障等を想定

Yeh, Y.C., Triple-triple redundant 777 primary flight computer,
IEEE Aerospace Applications Conference, 1996. Proceedings., pp. 293 – 307, vol.1, 1996.

事例紹介

- ボーイング777
  - 4台のアクチュエータ制御ユニット（ACE）
  - 3系統の電源，油圧，通信システム



Yeh, Y.C., Triple-triple redundant 777 primary flight computer,
IEEE Aerospace Applications Conference, 1996. Proceedings., pp. 293 – 307, vol.1, 1996.

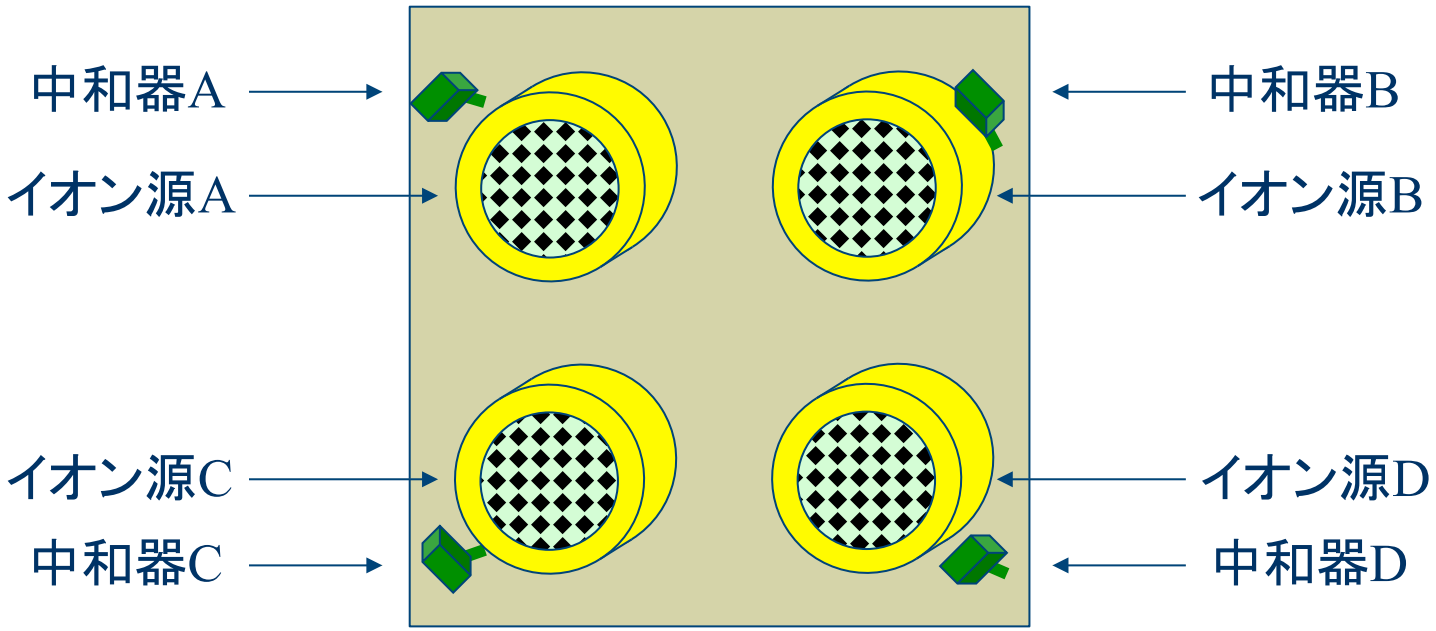
事例紹介

- はやぶさ
  - 冗長性のほんの例
    - イオンスラスタ制御装置：1（内部で3重化多数決）
    - イオンエンジン：4
    - イオンエンジン電源：3
    - リアクションホイール：3
    - キセノンタンクバルブ：2系統
    - アンテナ：3
  - 冗長度を増やせいない要因
    - 重量

川口淳一郎監修，小惑星探査機「はやぶさ」の超技術，講談社，2011.

事例紹介

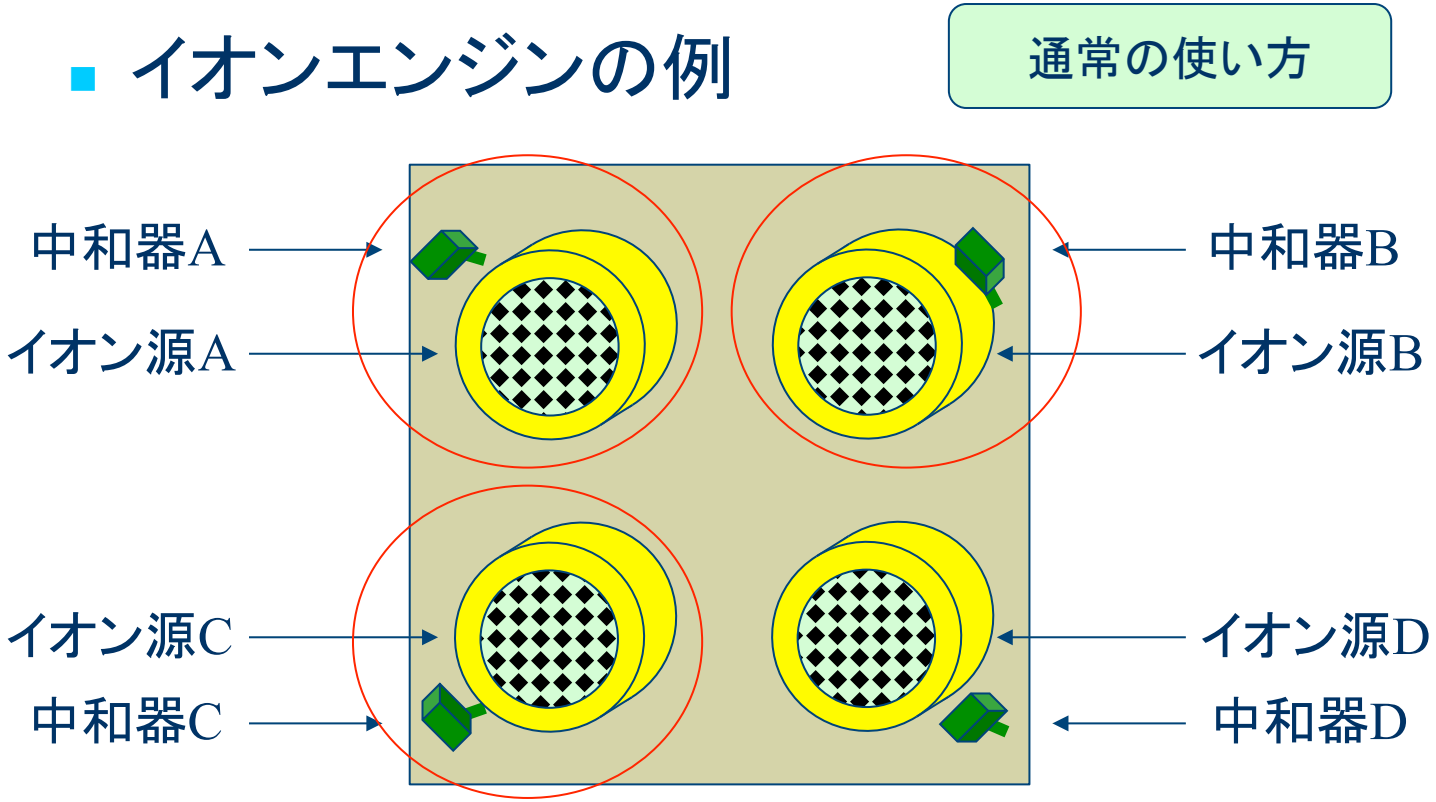
- はやぶさ
  - イオンエンジンの例



川口淳一郎監修，小惑星探査機「はやぶさ」の超技術，講談社，2011.

事例紹介

- はやぶさ
  - イオンエンジンの例

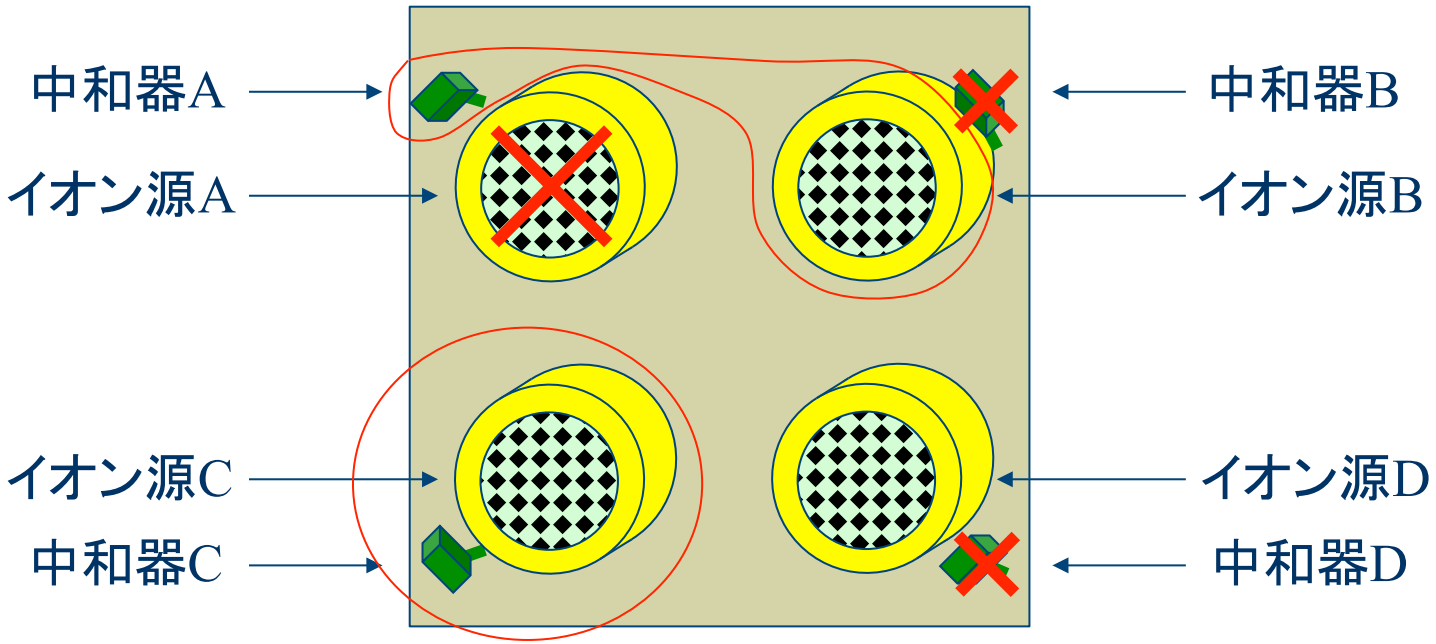


川口淳一郎監修，小惑星探査機「はやぶさ」の超技術，講談社，2011.

事例紹介

- はやぶさ
  - イオンエンジンの例

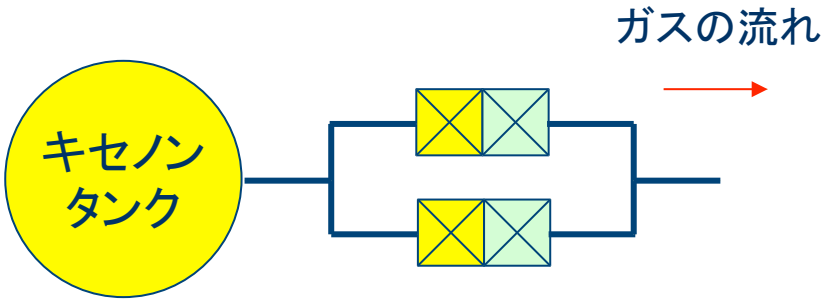
実際に生じた故障



川口淳一郎監修，小惑星探査機「はやぶさ」の超技術，講談社，2011.

事例紹介

- はやぶさ
  - 推進剤供給系のバルブ
    - 閉じない故障に耐える
      - 直列
    - 開かない故障に耐える
      - 並列
    - 同様な故障が同時に起こるのを回避
      - 2種類のバルブ
        - タイプ1：ラッチングバルブ
        - タイプ2：電磁弁



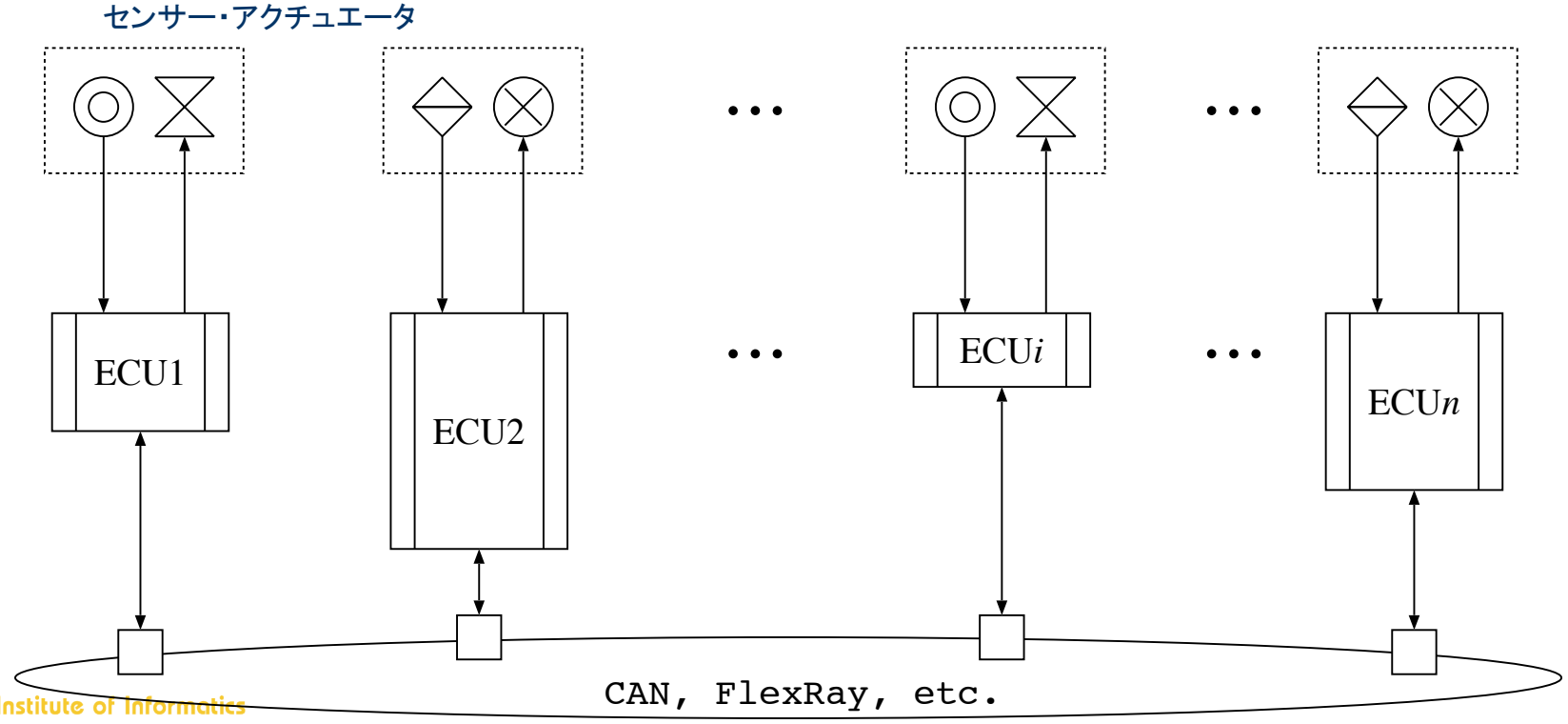
冗長性には
独立性が重要

ポイント5

川口淳一郎監修，小惑星探査機「はやぶさ」の超技術，講談社，2011.

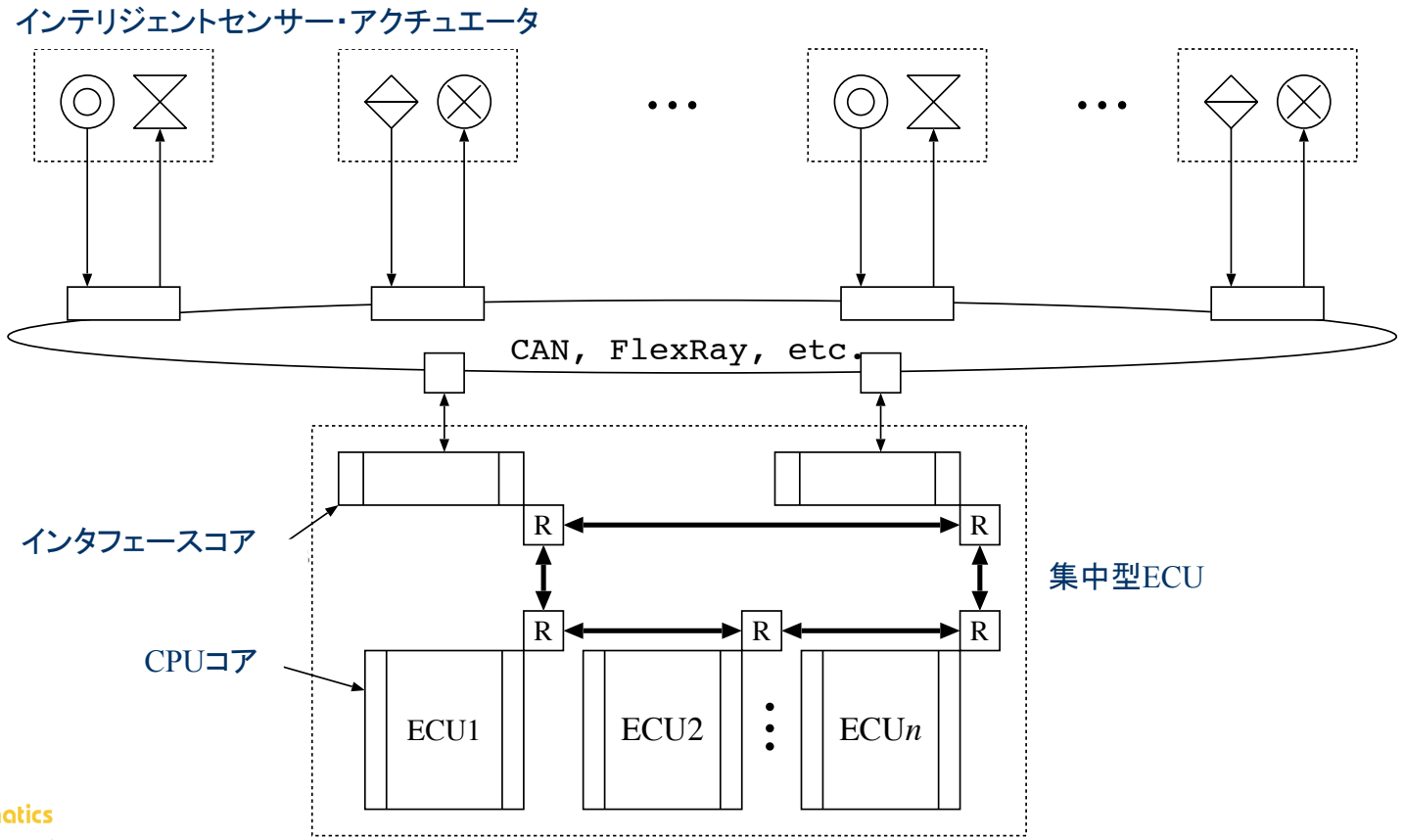
我々の取り組み

- 近年の車には多くのECU（制御用コンピュータ）が使われている
  - 従来のECU構成法



我々の取り組み

- 近年の車には多くのECUが使われている
  - 集中型ECUのアプローチ



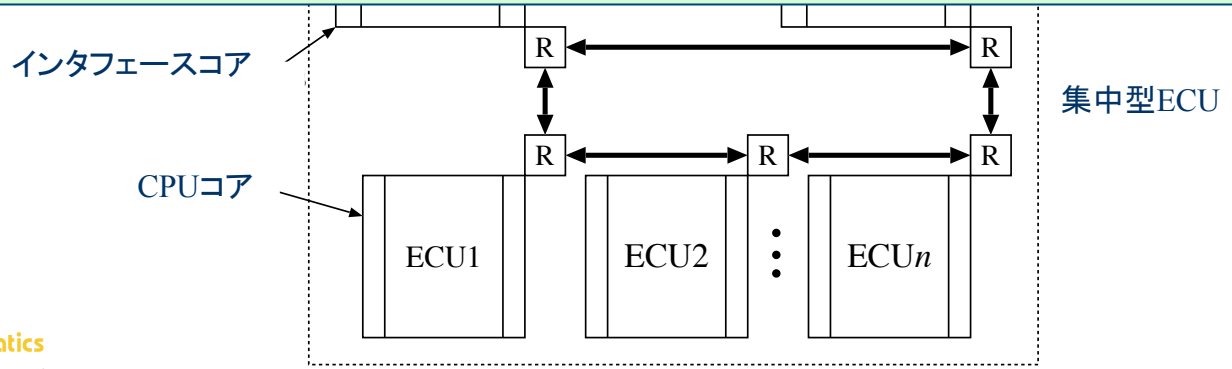
我々の取り組み

- 近年の車には多くのECUが使われている
  - 集中型ECUのアプローチ

インテリジェントセンサー・アクチュエータ

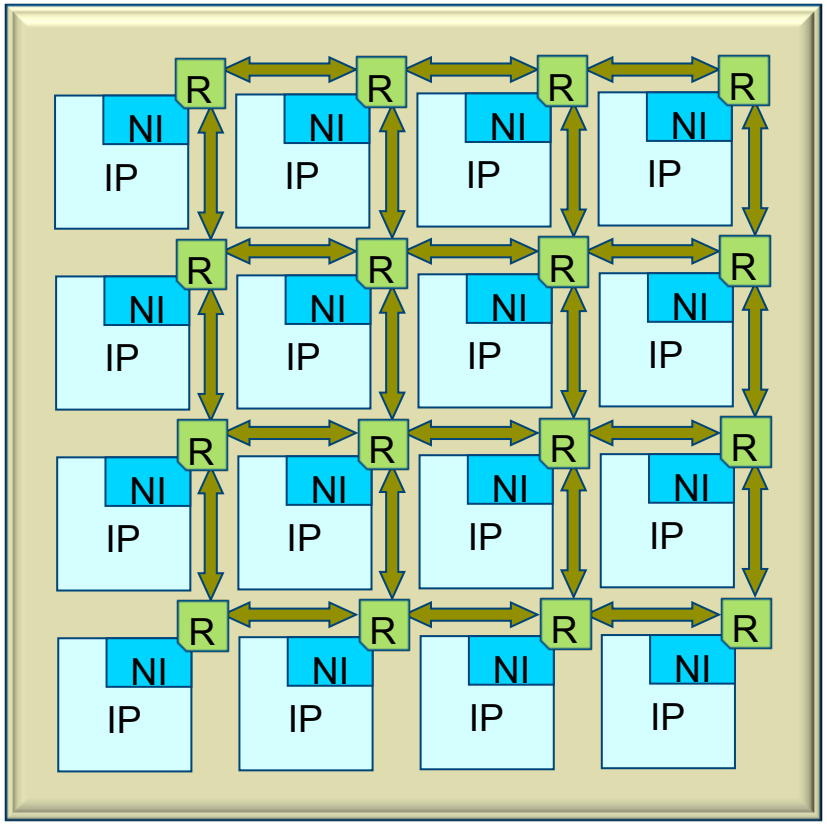
どのECUもすべてのセンサ・アクチュエータにアクセス可

各ECUに、負荷に応じて効率よくタスクを割当できる
あるECUが故障しても、他のECUにタスクを振り分けることで、処理を実行可
（ECU故障により、そのECUに割当てられていた機能が失われることを回避可）



我々の取り組み

- ネットワークオンチップ(NoC)

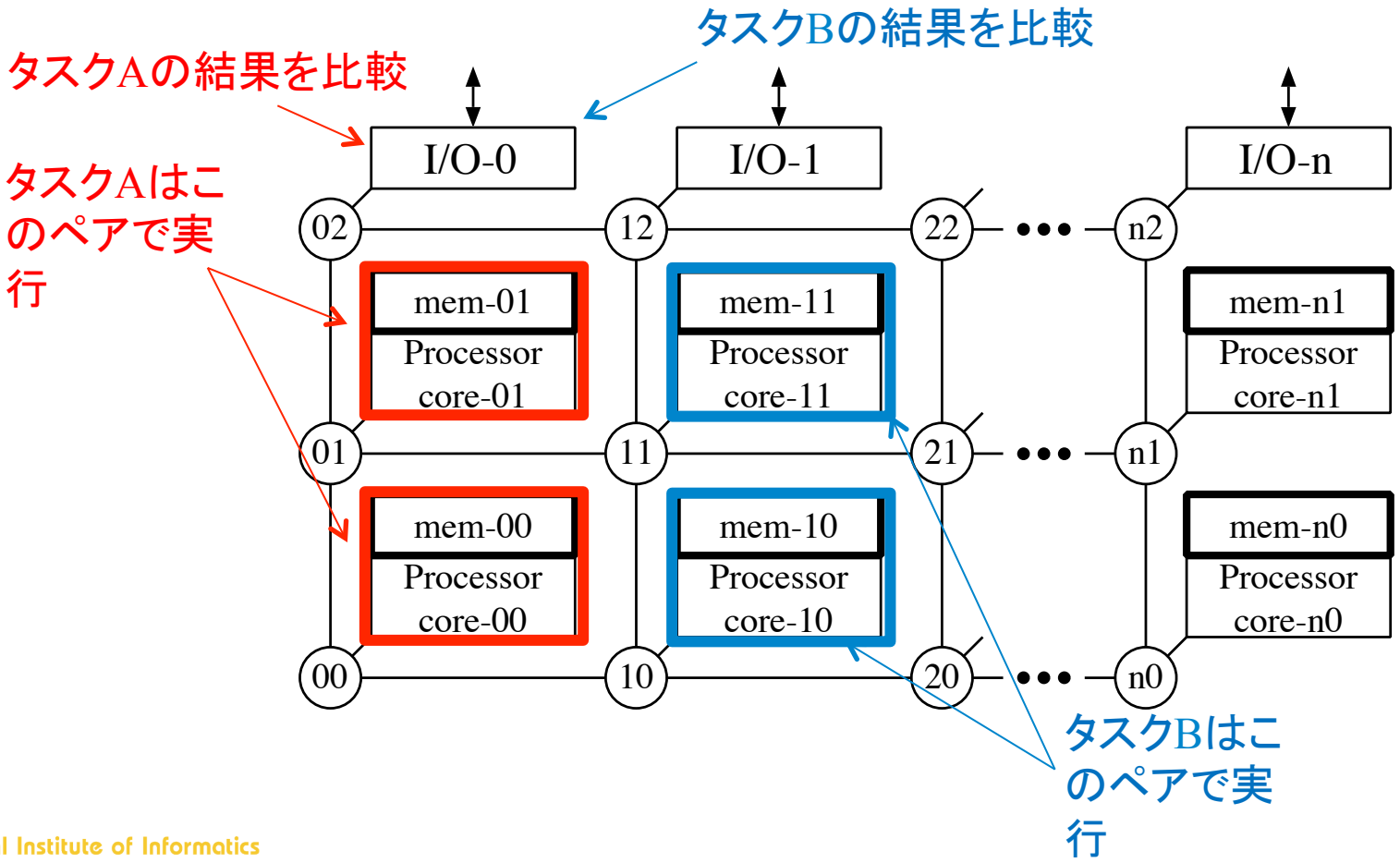


IP：CPUコアやアクセラレータ・メモリ等
NI: ネットワークインターフェース
R: ルータ

- 一つのチップの中にネットワークを作り、パケット交換により通信を行う
  - 計算コア等の数や種類にかかわらず、フレキシブルでスケーラブルなシステムを構築可能

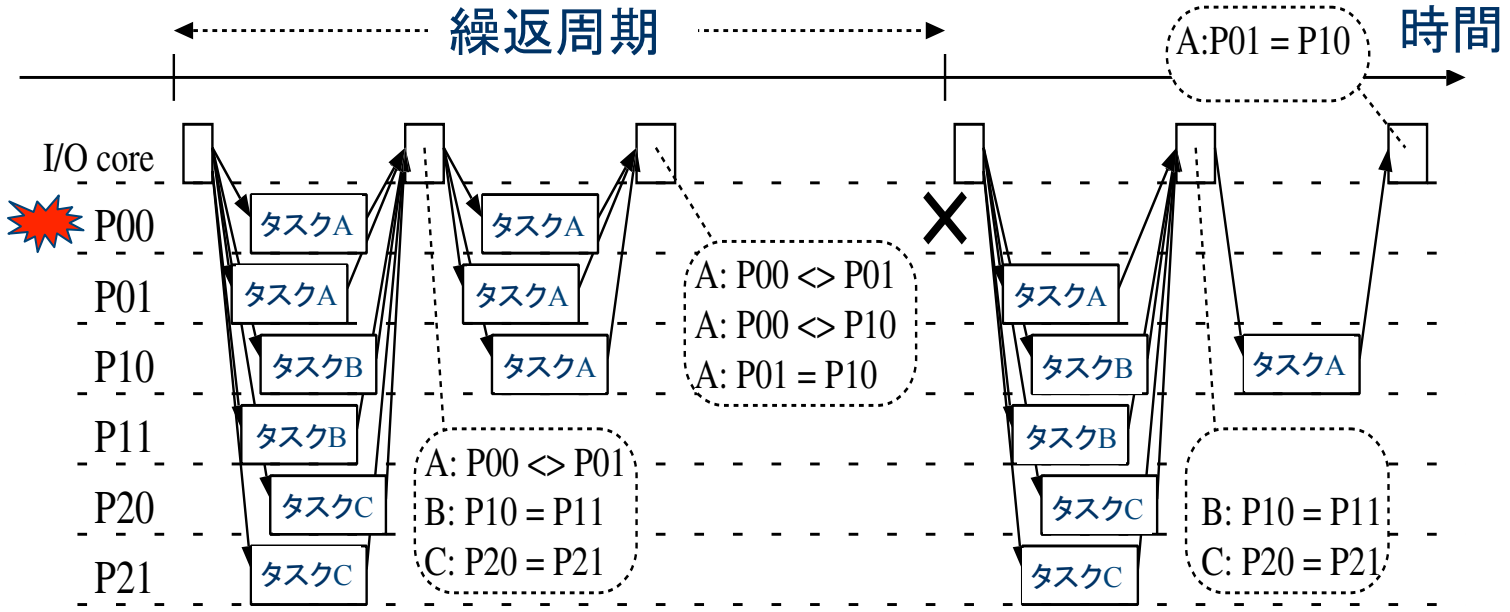
我々の取り組み

- 通常は各タスク(まとまった処理)を2重実行



我々の取り組み

- 故障発生時は，一時的に3重実行，ペアの再構成



タスクはいくつかのコアに冗長に格納しておく

制御し続けることが目的

まとめ

- 故障に耐えるシステム作り
  - ポイント1
    - 「故障」と「システム障害」を区別して考える
      - 「故障」は起こっても「システム障害」は防ぎたい
  - ポイント2
    - 「故障」に耐えるための基本的考え方⇒「冗長性」
  - ポイント3
    - 「想定外の故障」に耐えるようにするのは不可能
  - ポイント4
    - 悪いものを冗長化してもよくならない
  - ポイント5
    - 冗長性には独立性が重要