ネットワークエンジニアが知るべきサービスメッシュ

Why Network Engineers should know about Servicemesh

26 Jan. 2023

Miya Kohno, Cisco Systems (mkohno@cisco.com)

Abstract

サービスメッシュは、現在クラウド界隈で注目されている言葉のひとつで、マイクロサービス間の接続、制御、監視を可能にし、マイクロサービスアーキテクチャの一貫した開発・展開、ヘルスチェック、セキュリティ、およびスケーラビリティを提供します。

しかし、サービスメッシュが必要不可欠という訳でもなく、単なるバズワードのようにも思えます。そもそもマイクロサービス自体が、多くのネットワークエンジニアにとってさほど気にかけるべき存在でもないかもしれません。

本セッションでは、「ネットワークエンジニアが知るべきサービスメッシュ」と題し、我々がサービスメッシュに注目すべき3つのポイントについて議論します。

1）マイクロサービスはデザインパターンの集合であり、サービスメッシュもデザインパターンである

2）マイクロサービスのデザインパターンは、ネットワークシステムにも有効である

3）システムの境界が変化している

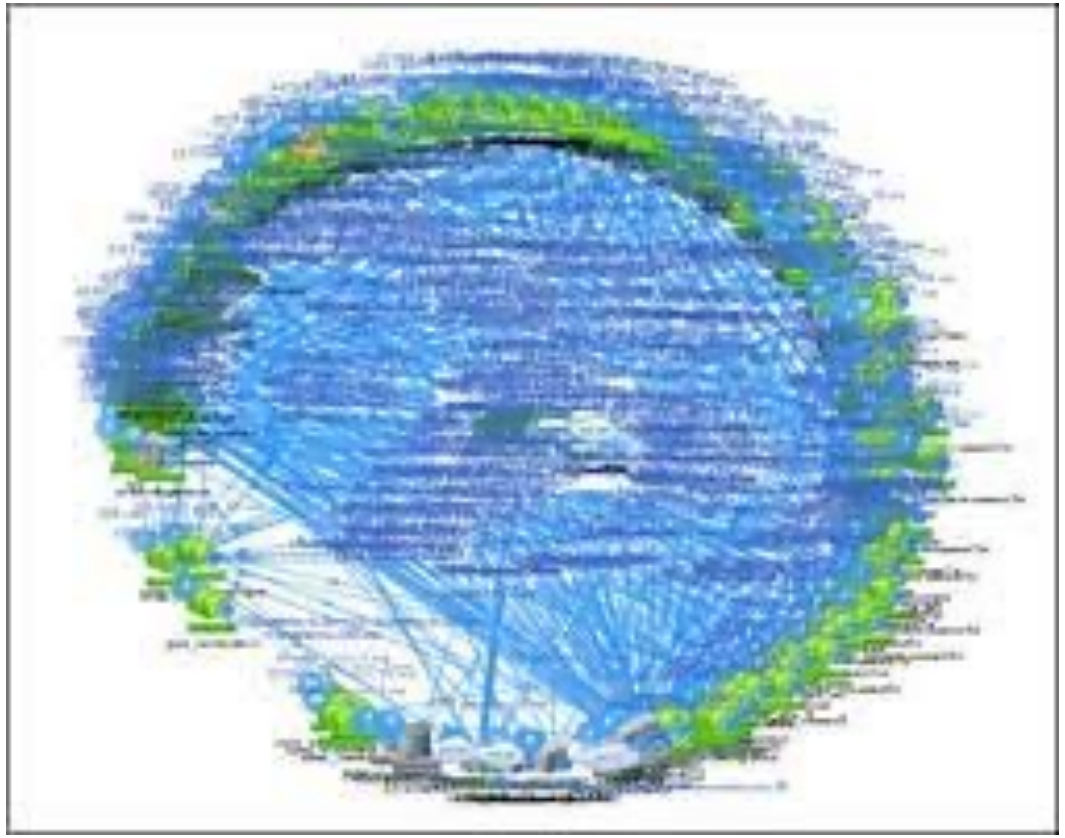
サービスメッシュに注目すべき3つのポイント (1)

マイクロサービスはデザインパターンの集合であり、
サービスメッシュもデザインパターンである

マイクロサービス：疎結合 (loosely coupled) システム

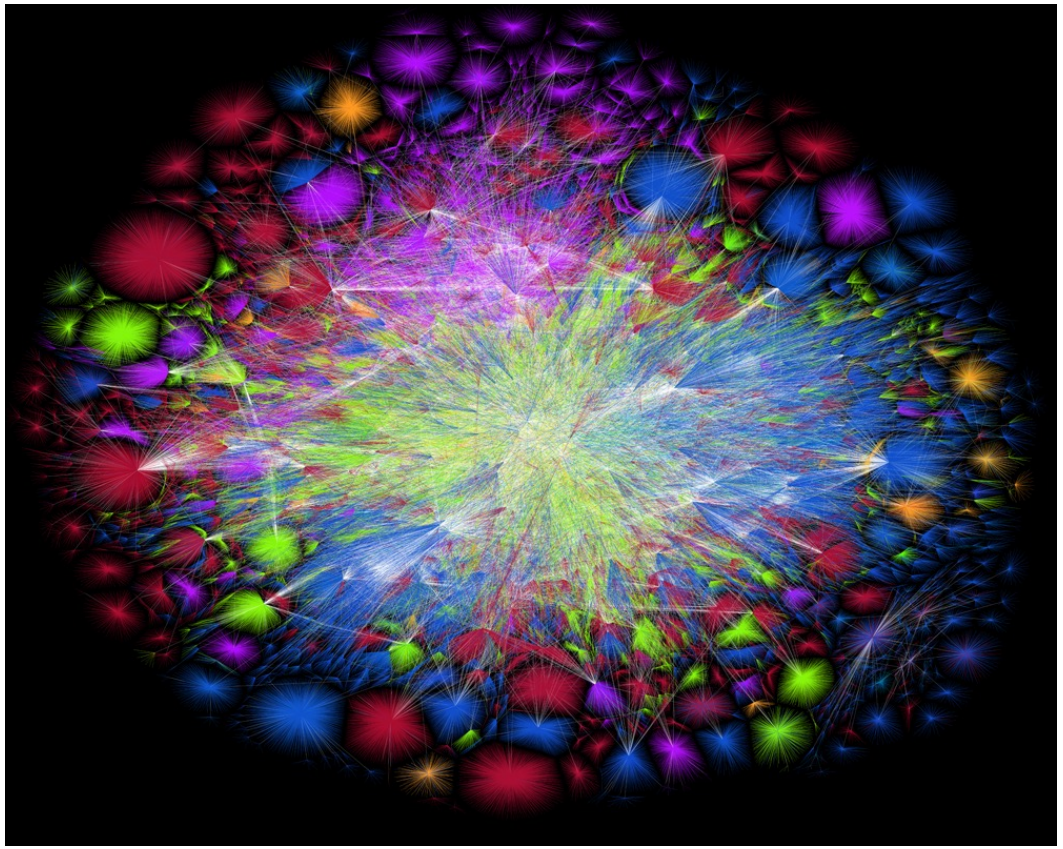
- 複数の独立した小規模なサービスを組み合わせて、一つの大きなアプリケーションを構築する開発手法
- 各コンポーネントの独立性が高く、緩やかに結合されている

Micro Service (Netflix)



https://www.slideshare.net/BruceWong3/the-case-for-chaos

The Internet



http://www.opte.org/the-internet/

デザインパターンとは

アーキテクチャ設計方法の分類

分類	特徴	例	どちらかというと
規範的 (Normative)	ソリューションに基づく	建築基準法 通信規格	Science
合理的 (Rational)	方法論に基づく	システム分析	Science
参加型 (Participative)	システムのステークホルダーに基づく	コンカレントエンジニアリング ブレーンストーミング	Art
発見的 (Heuristic)	教訓や経験から学ぶ	格言 デザインパターン	Art

Mark W.Maier, Eberhardt Rechtin “The Art of Systems Architecting” を参考に作成

- システムのスコープを定義する規範的・合理的な方法だけでは、前例のない未知のシステムや、状況の変化に対応しにくい
- 発見的方法 (heuristics) は、規範的・合理的方法を補い、知識や経験をもとにして問題を解決するための方法論である
- デザインパターンは、パターンランゲージに由来し、アーキテクチャ指針を共有するための共通言語となる

デザインパターン @ JANOG

- 発見的 (Heuristic) なデザイン手法 Source : P. Alexander, “Pattern Language”

- 今から10年前の昨日！（2013年1月25日）JANOG31でセッションしました。
  https://www.janog.gr.jp/meeting/janog31/program/design.html
- そしてその3年後（2016年1月21日）JANOG37でもセッションしました。
  https://www.janog.gr.jp/meeting/janog37/program/pattern.html

- JANOGにおいて随時、方法論として活用されています。
  - パスワードのデザインパターン
    https://www.janog.gr.jp/meeting/janog31.5/program/passwd-system-design-pattern.html
  - ネットワークシステムデザインにおけるエキスパート思考パターン
    https://www.janog.gr.jp/meeting/janog37/download_file/pattern_title.pdf
  - Telemetry システムのためのデザインパターン
    https://www.janog.gr.jp/meeting/janog43/application/files/9615/4815/7538/Telemetry_.pdf

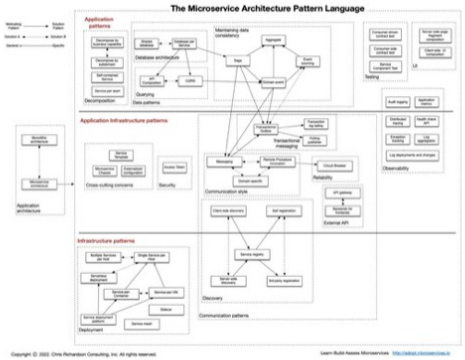
マイクロサービスはデザインパターンの集合

What are microservices?

Microservices - also known as the microservice architecture - architectural style that structures an application as a collectio are

- Highly maintainable and testable
- Loosely coupled
- Independently deployable
- Organized around business capabilities
- Owned by a small team

The microservice architecture enables the rapid, frequent and of large, complex applications. It also enables an organizatior technology stack.



The pat guide

The microservice ar Moreover, when usi must address.

The microservice ar applying the micros

1. The pattern l a good fit for
2. The pattern l successfully.

https://microservices.io/patterns/

Pattern: Microservice Architecture

Context

You are developing a server-side enterprise application. It must support a variety of different clients including desktop browsers, mobile browsers and native mobile applications. The application might also expose an API for 3rd parties to consume. It might also integrate with other applications via either web services or a message broker. The application handles requests (HTTP requests and messages) by executing business logic; accessing a database; exchanging messages with other systems; and returning a HTML/JSON/XML response. There are logical components corresponding to different functional areas of the application.

Problem

What's the application's deployment architecture?

Forces

- There is a team of developers working on the application
- New team members must quickly become productive
- The application must be easy to understand and modify
- You want to practice continuous deployment of the application
- You must run multiple instances of the application on multiple machines in order to satisfy scalability and availability requirements
- You want to take advantage of emerging technologies (frameworks, programming languages, etc)

Solution

Define an architecture that structures the application as a set of loosely coupled, collaborating services. This approach corresponds to the Y-axis of the Scale Cube. Each service is:

- Highly maintainable and testable - enables rapid and frequent development and deployment
- Loosely coupled with other services - enables a team to work independently the majority of time on their service(s) without being impacted by changes to other services and without affecting other services

サービスメッシュもデザインパターン

Pattern: Service mesh

Context

You have applied the Microservice architecture pattern and architected your system as a set of services.

Problem

You must implement numerous cross-cutting concerns including:

- Externalized configuration - includes credentials, and network locations of external services such as databases
- Logging - configuring of a logging framework such as log4j or logback
- Health checks - a url that a monitoring service can “ping” to determine the health of the application
- Metrics - measurements that provide insight into what the application is doing and how it is performing
- Distributed tracing - instrument services with code that assigns each external request an unique identifier that is
  services.

Forces

Solution

Use a service mesh that mediates all communication in and out of each service.

https://microservices.io/patterns/deployment/service-mesh.html

- 独立性を高めるパターン
- メインの業務ロジックと周辺処理を分離することにより、メインの処理実装をシンプルにし、柔軟性を高める （多言語対応(Polyglot)など）
- 周辺処理の例：
  - マイクロサービス間の通信・トラフィック制御
  - ロギング
  - ヘルスチェック
  - …

サービスメッシュに注目すべき3つのポイント (2)

マイクロサービスのデザインパターンは、

ネットワークシステムにも有効である

マイクロサービスデザインパターンの基本原則

- Scalability (スケーラビリティ)
- Availability (可用性)
- Resiliency (回復力)
- Independent, autonomous (独立性、自律性)
- Decentralized governance (分散型ガバナンス)
- Failure isolation (故障の隔離)
- Auto-Provisioning (自動プロビジョニング)
- Continuous delivery through DevOps (DevOpsによる継続的なデリバリー)

Reference： https://dzone.com/articles/design-patterns-for-microservices

→ この基本原則は、ネットワークシステムにとっても重要！

最近の通信事故

総務省 電気通信事故検証会議 公開資料より https://www.soumu.go.jp/main_content/000852486.pdf

- 最近、通信障害が多発。その背景には、**リスク評価やリスクの洗い出しの不足、ヒューマンエラー防止や訓練面での課題、保守運用態勢に対するガバナンスの不足等、共通する課題も多いのではないか。**
- こうした個別の事故の背景にある組織・態勢面等の**構造的問題について、検証していくことが必要ではないか。**

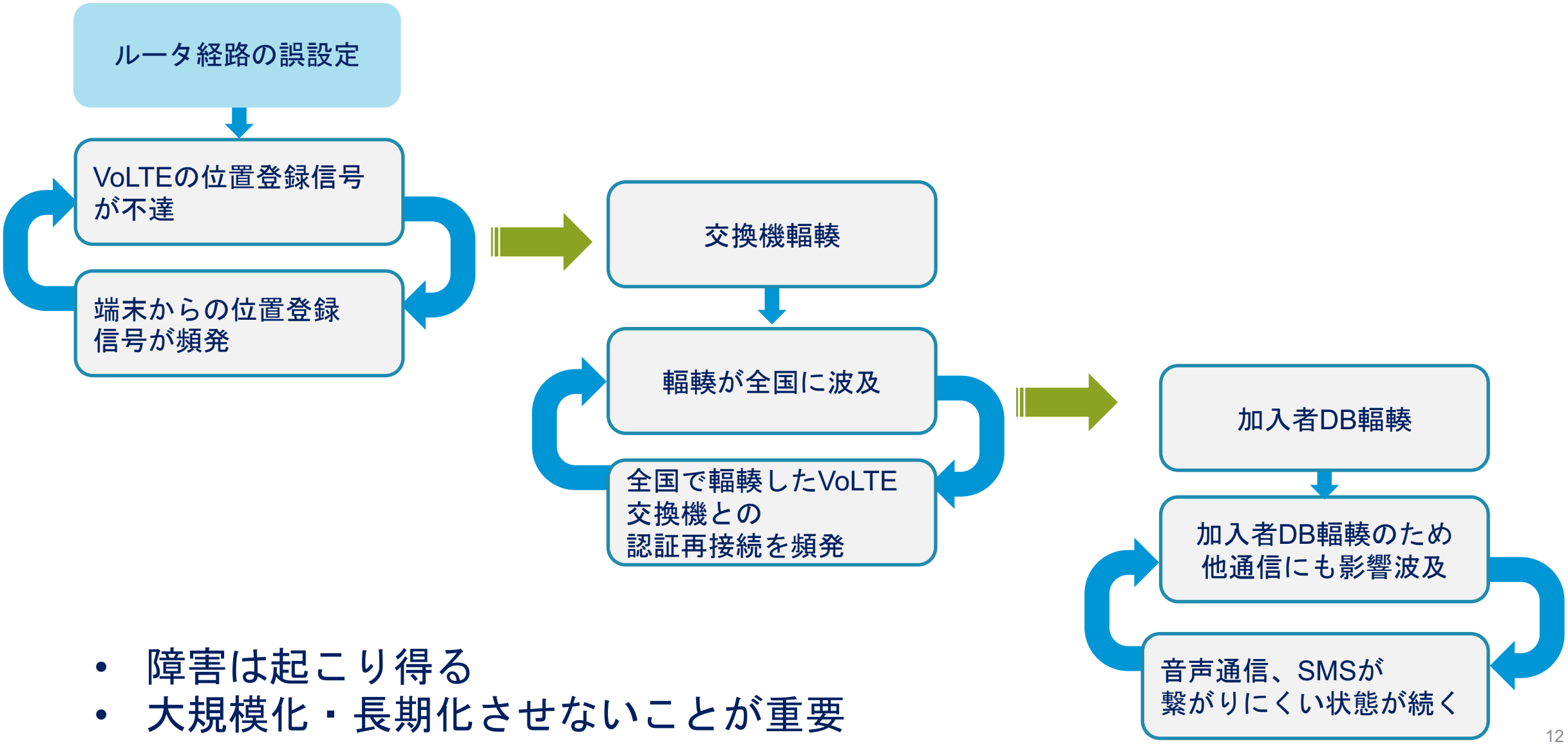
発生日時（継続時間）	通信事業者	影響サービス	影響範囲	発生原因
7月2日(土)（61時間25分）	KDDI	音声通話、SMS、ホーム電話、データ通信	全国 音声通話：約2,278万人 データ通信：765万人以上 【重大事故に該当】	人為的ミス コアネットワークの障害
8月24日(水)（45分間）	KDDI	音声通話、SMS、ホーム電話、データ通信	東日本エリア 最大8.3万人	設備異常 コアネットワークの障害
8月25日(木)（5時間47分）	NTT西日本	インターネットサービス（フレッツ光）	西日本エリア 最大211万回線(品質低下) 【重大事故に該当】	設備異常 コアネットワークの障害
9月4日(日)（2時間6分）	楽天モバイル	音声通話、データ通信	全国エリア 最大130万回線 【重大事故に該当】	設備異常 コアネットワークの障害
9月4日(日)（37分間）	ソフトバンク	音声通話、データ通信	中国・四国・九州地方 4G回線：最大約105万回線等	人為的ミス コアネットワークの障害
9月11日(日)（2分間）	KDDI	音声通話、データ通信	東日本エリア	設備異常 コアネットワークの障害
11月10日(木)（1時間56分）	NTTドコモ	音声通話、データ通信	熊本県内の一部エリア 音声通話：約2万人 データ通信：約17.9万人	設備異常 コアネットワークの障害

→ 必要な対策 (?!)

- 技術基準の見直し
- 管理規定内容の見直し
- ガバナンス強化やモニタリングの見直し

障害機序– 2022年7月の例

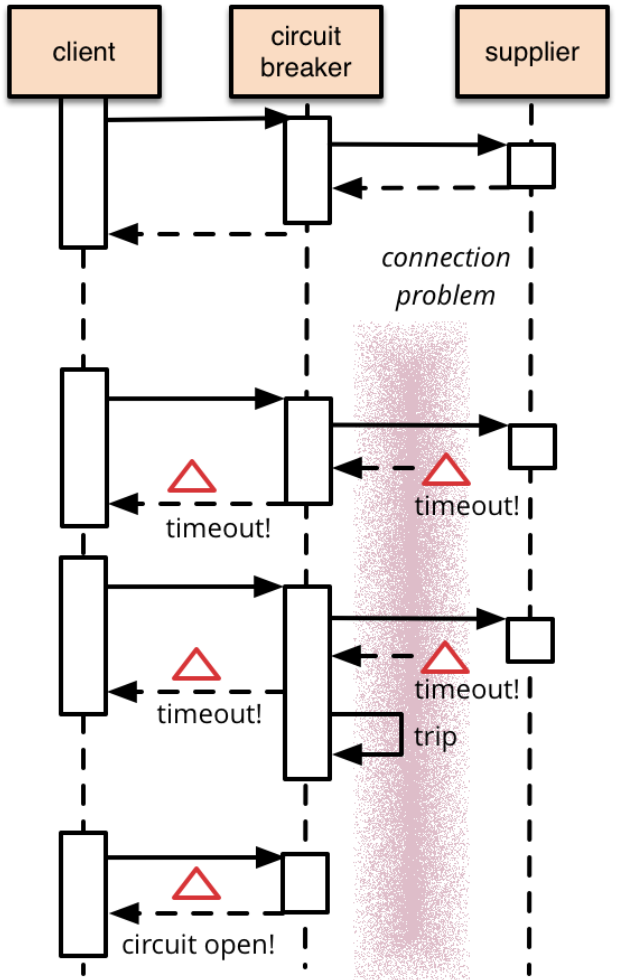
https://www.soumu.go.jp/main_content/000839638.pdf



- 障害は起こり得る
- 大規模化・長期化させないことが重要

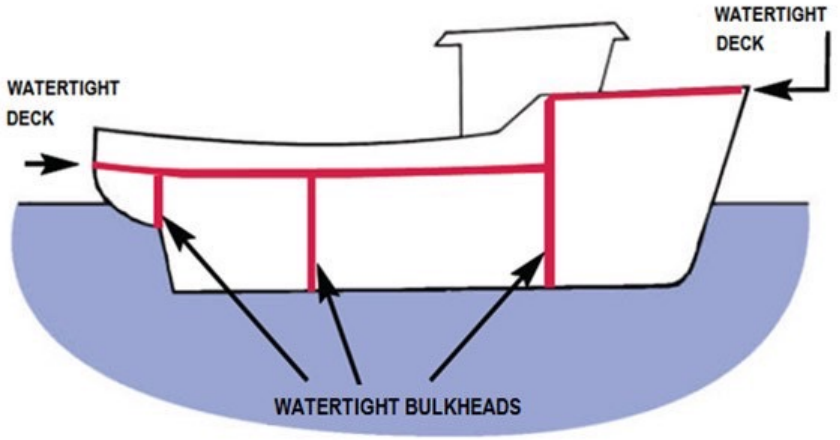
デザインパターン – Failure islolation (故障の隔離)

Circuit Breaker （サーキットブレーカー）デザインパターン



https://martinfowler.com/bliki/CircuitBreaker.html

Bulkhead（バルクヘッド）デザインパターン



https://medium.com/@abhisheksharma_49625/bulkhead-pattern-for-micro-services-85d5f9e3215f

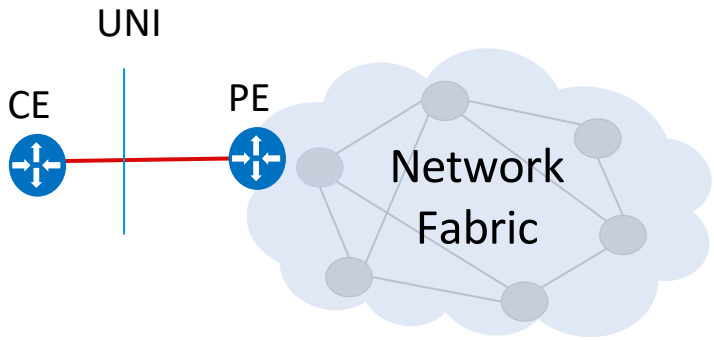
サービスメッシュに注目すべき3つのポイント (3)

システムの境界が変化している

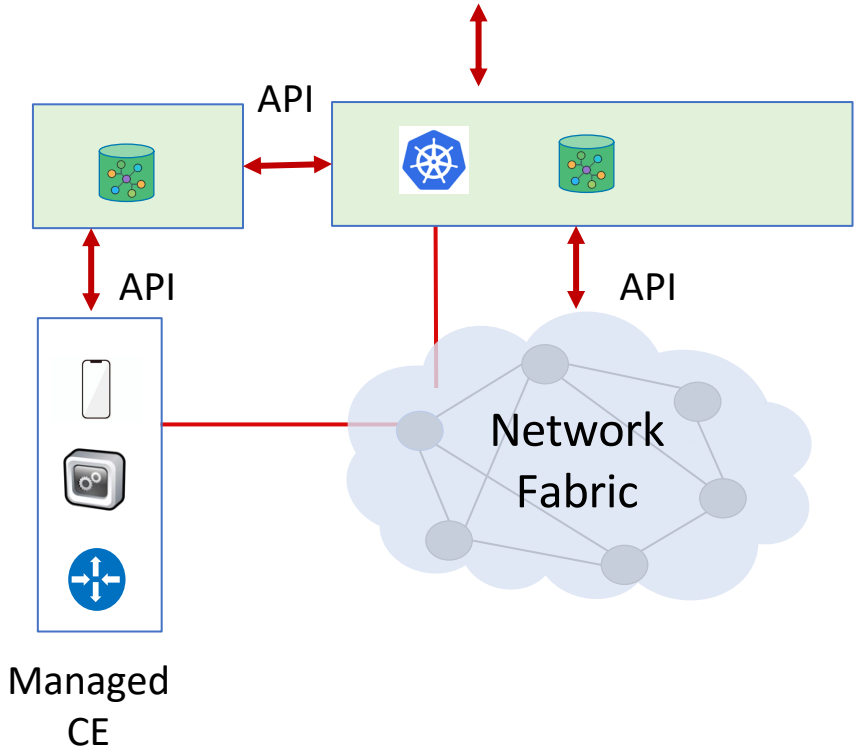
システム境界の変化

UNI から API へ

Traditional SP Network



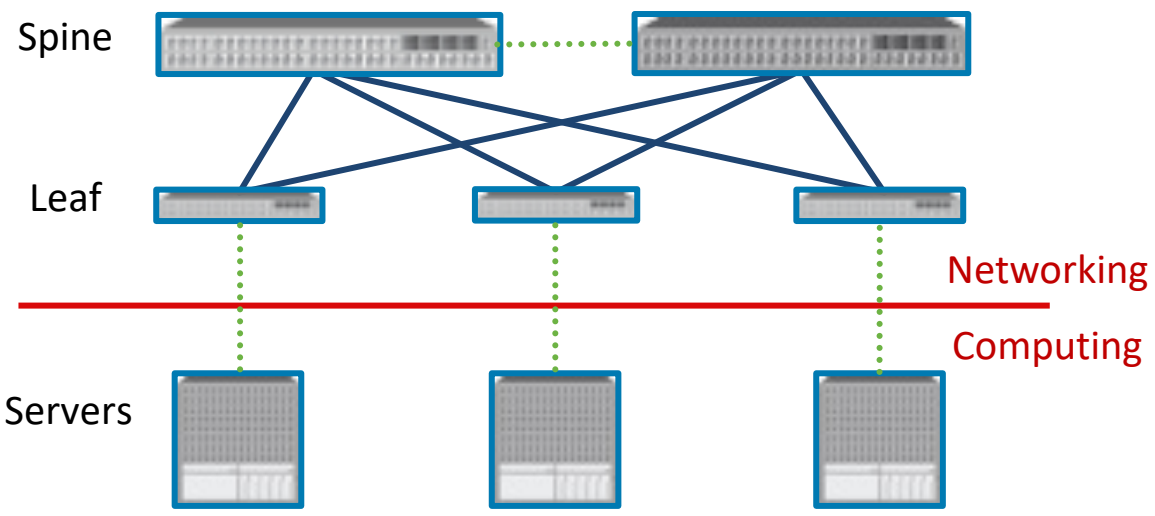
SP Network as a Platform for Managed NaaS



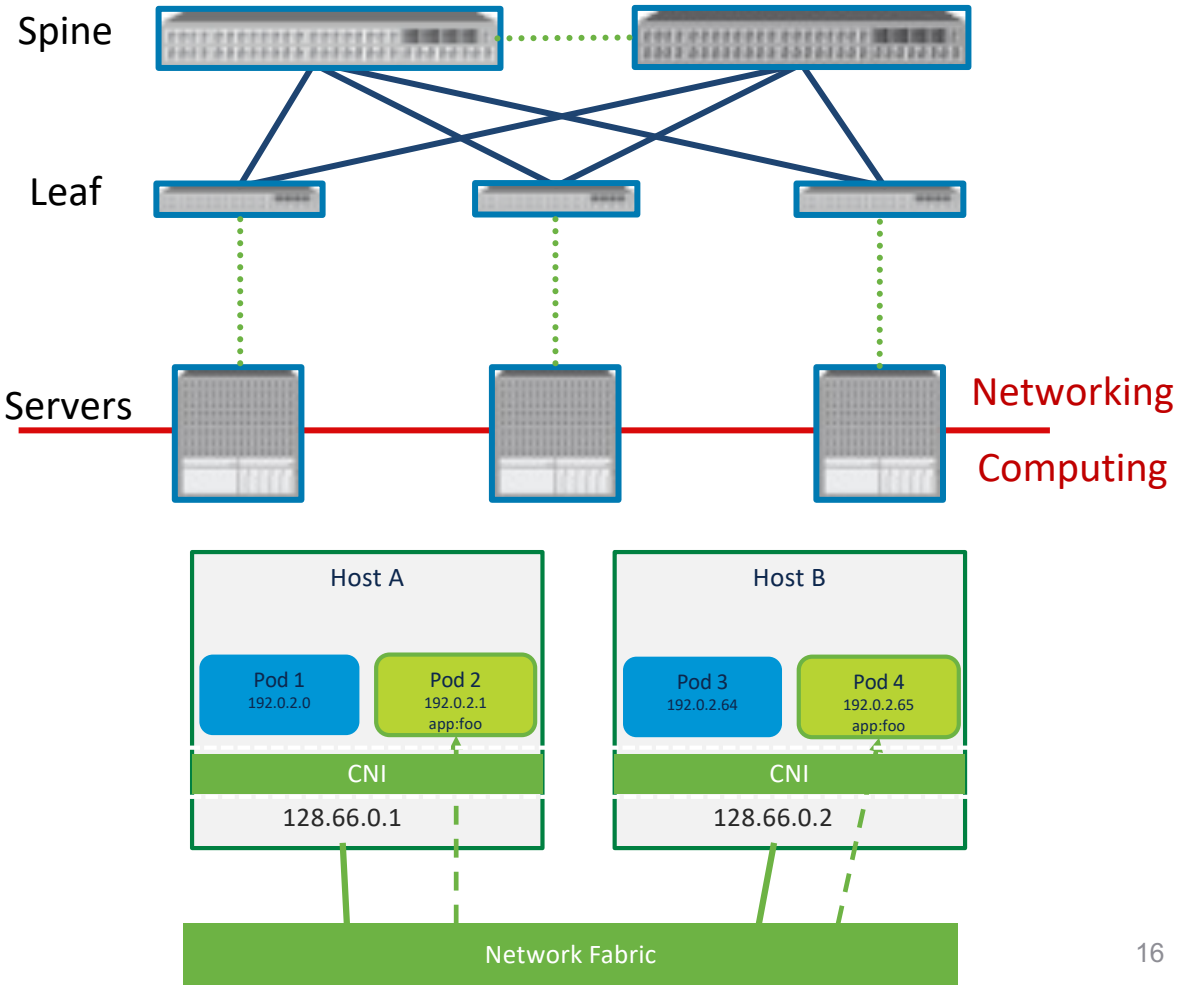
システム境界の変化

Computing とNetworking 境界の変化

Traditional DC Fabric



Cloud Native Networking



Conclusion

「ネットワークエンジニアが知るべきサービスメッシュ」と題し、サービスメッシュに注目すべき3つのポイントについて議論した

1）マイクロサービスはデザインパターンの集合であり、サービスメッシュもデザインパターンである

2）マイクロサービスのデザインパターンは、ネットワークシステムにも有効である

3）システムの境界が変化している

関連記事：「システム理論の続き – デザインパターンとサービスメッシュ」
https://qiita.com/mkohno/items/f04e4037a8ae78750f54

The bridge to possible