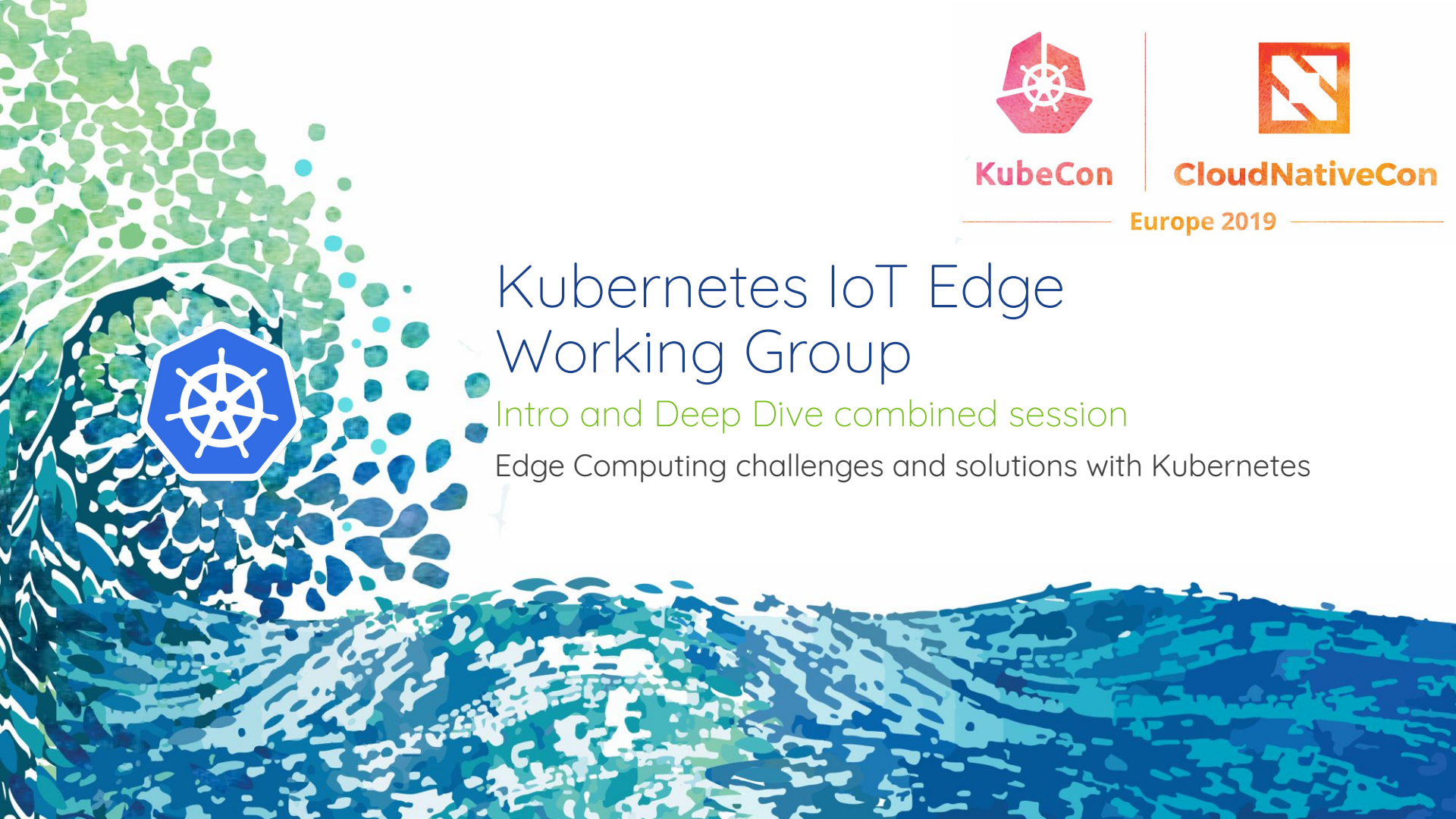


KubeCon | CloudNativeCon

Europe 2019

Kubernetes IoT Edge Working Group

Intro and Deep Dive combined session

Edge Computing challenges and solutions with Kubernetes

Abstract

Hidden slide during presentation – retain for published deck

This session will investigate and catalogue challenges encountered when Kubernetes is deployed in Edge and IoT applications.

We'll start by describing two basic approaches: deploying nodes to the Edge with a central control plane; and deploying whole clusters to the Edge.

This will be followed by a deep dive into Kubernetes architectural features and constraints in the context of both approaches. We'll see which course makes the most sense for some specific use cases.

Next we'll discuss some common challenges to successful deployments, such as resource limits and network availability, and provide some guidance on how to deal with them.

There are opportunities to contribute to the evolution of Kubernetes to better serve edge use cases. We will close with details on how you can get involved with the community effort to help this happen

Speakers

Cindy Xing
Huawei
@cindyxing

Dejan Bosanac
Redhat
@dejanb

Steve Wong
VMWare
@cantbewong

Kilton Hopkins
Edgeworx
@kiltonhopkins

Agenda

Approaches

1. Nodes to edge, with remote central control plane
2. Deploy whole clusters to the edge

Choosing an approach for some specific use cases

Dealing with some common challenges

Resource limits

Network Availability

Data Plane communication – edge to cloud services

Security

How you can get involved with community efforts

Approaches

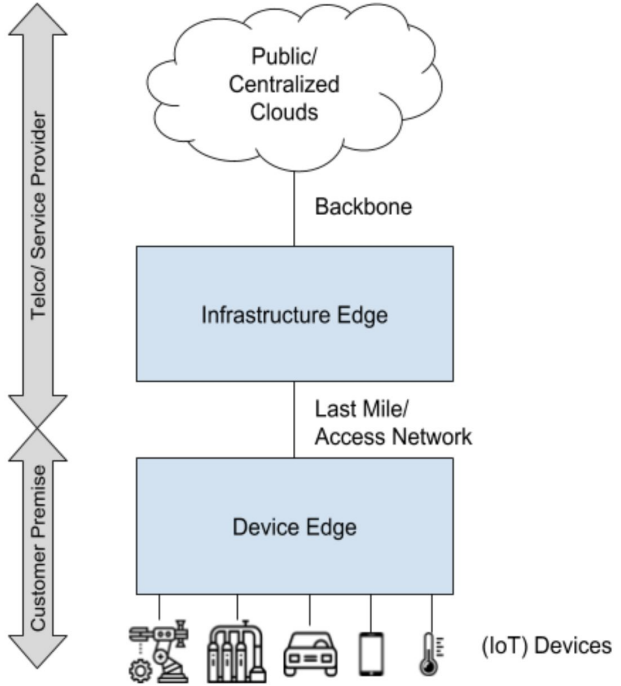
Available Approaches

1. Install and manage a Kubernetes cluster at edge locations
    - Reference architecture: K3S
    - Cluster at Edge
2. Manage edge nodes from cloud
    - Reference architecture: KubeEdge
    - Worker Node at Edge
3. Hierarchical cloud + edge
    - Reference architecture: Virtual Kubelet
    - Cluster at Edge and manage from Cloud

Edge types

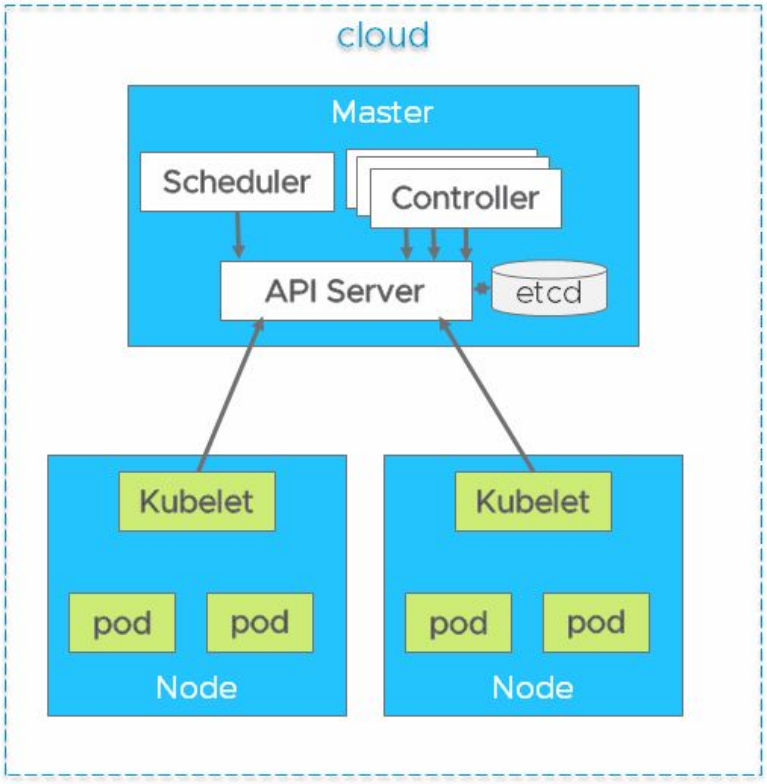
- Infrastructure Edge
  - Deploy whole clusters on Edge sites
  - Hybrid-clouds
  - Federated clusters

- Device Edge
  - Deploy cluster nodes outside of the cloud

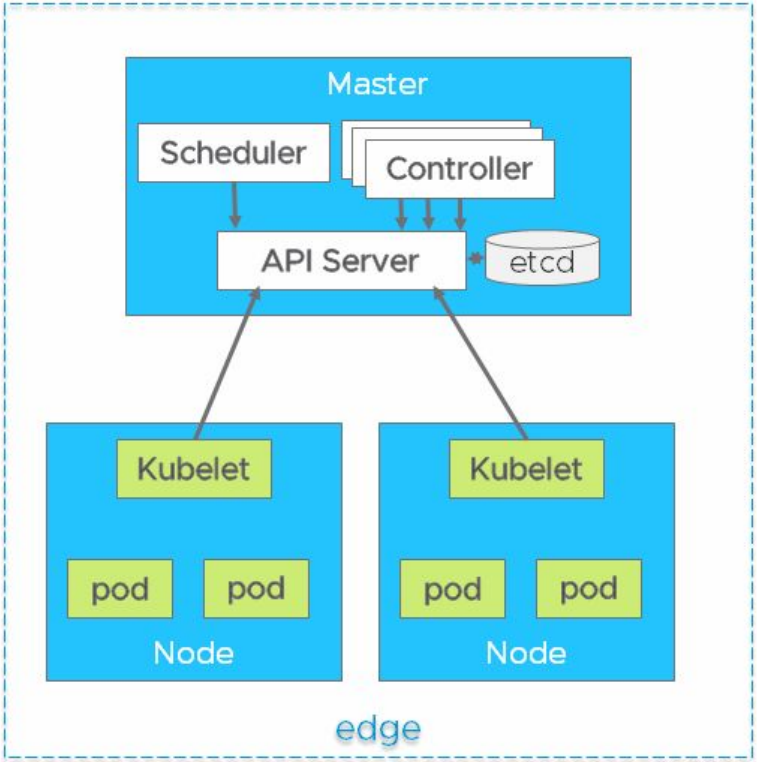


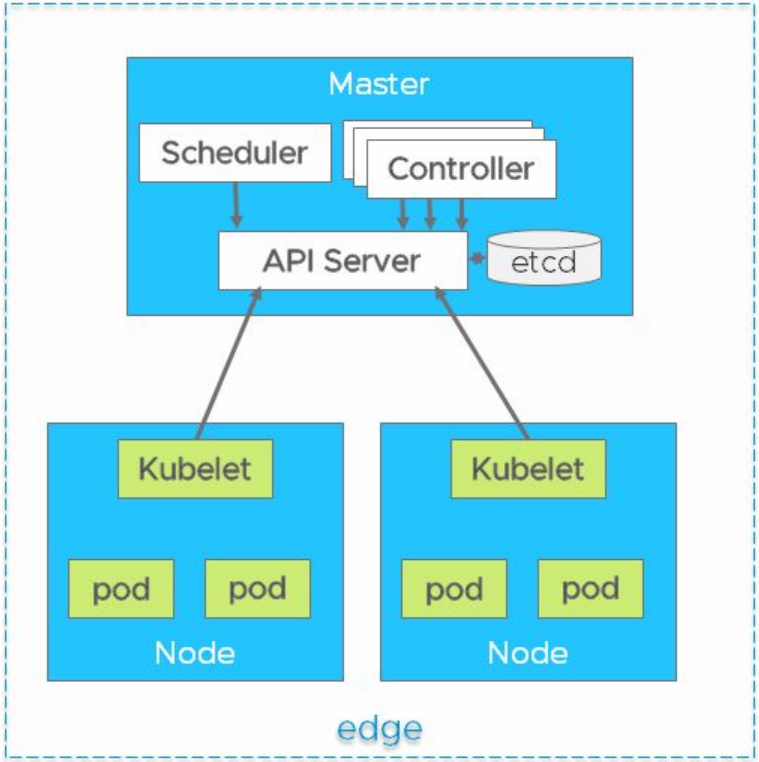
Source: Icons from https://www.flaticon.com/free-icon

Standard Kubernetes Architecture

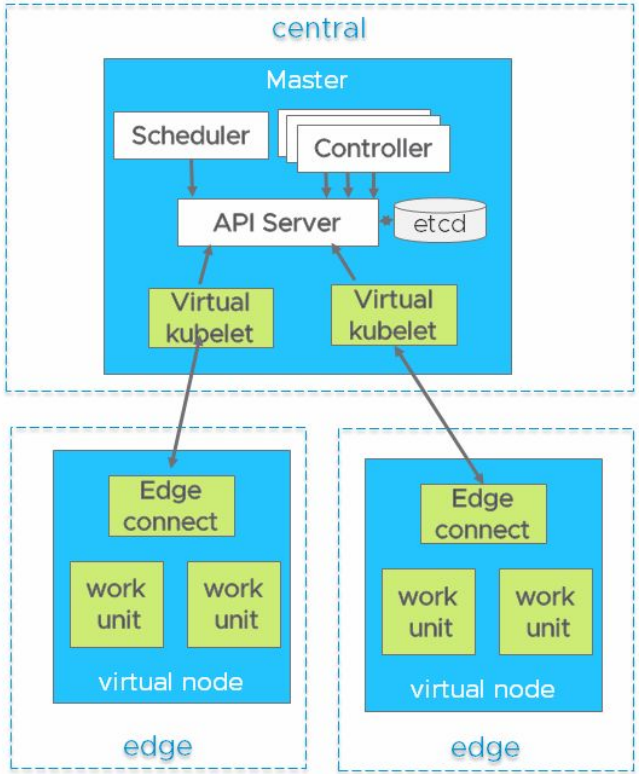


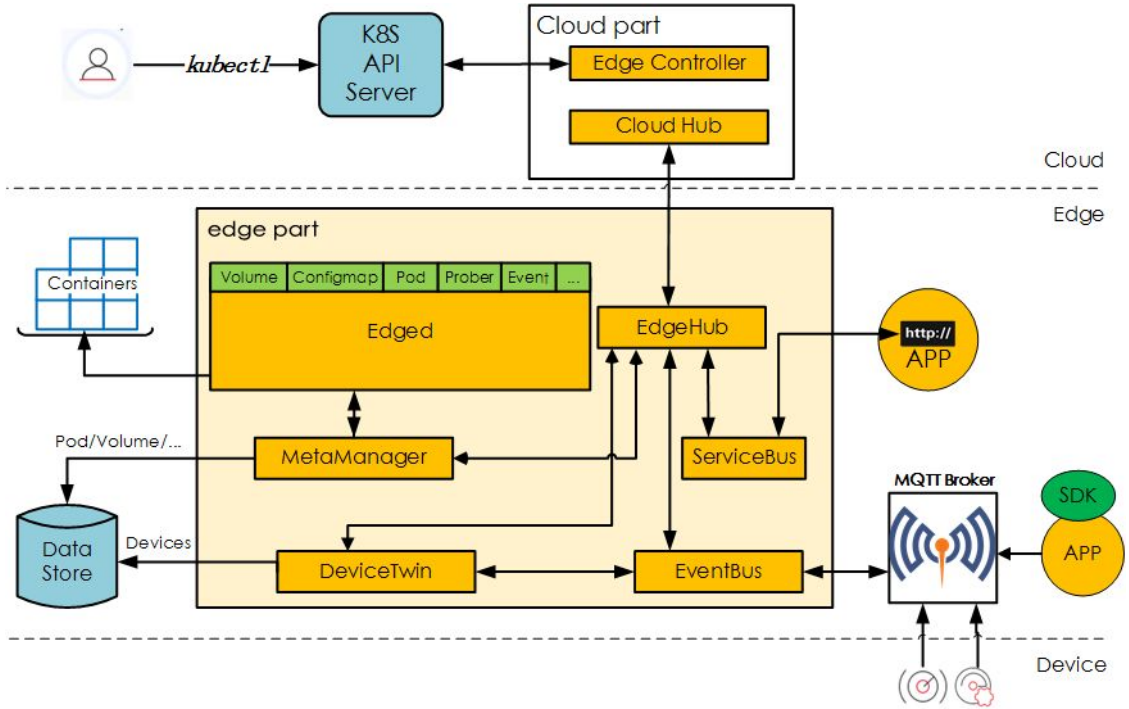
Option 1: whole clusters at edge



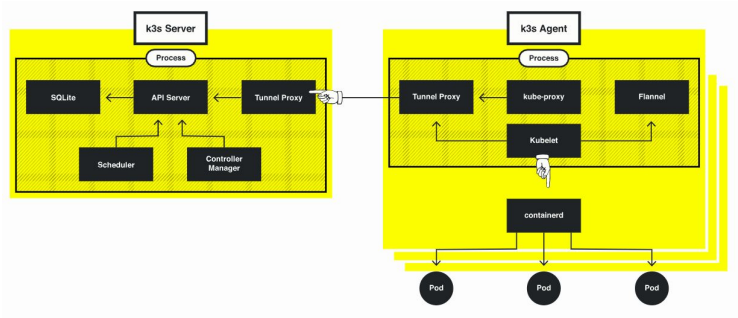


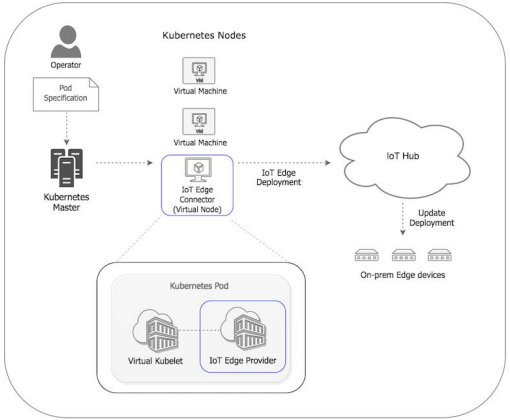
Option 2: central control managing edge

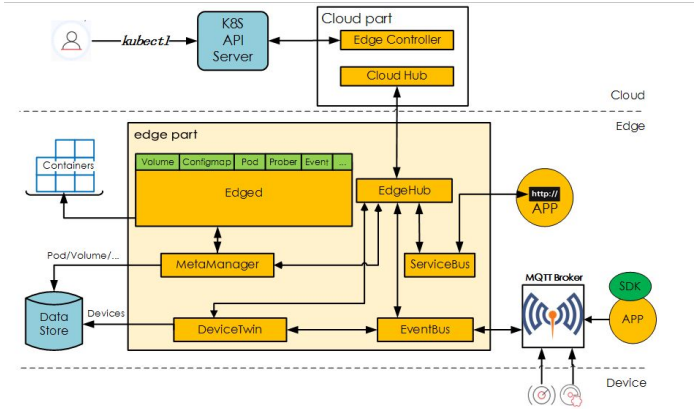




Comparisons







Area	K3S	KubeEdge	Virtual Kubelet
EdgeNode management	Yes	Yes	Indirect
App. Deployment & Orchestration	Yes	Yes	Yes
Device Management	No	Yes	No
EdgeNode registration	From client	From cloud	From cloud
Master Location	Edge	Cloud	Cloud and Edge
Pure K8s Native	Yes	Yes	With external provider
Extensibility	No	Yes	Yes
Module pluggable	No	Yes	Yes
Lightweight	Yes	Yes	No

Use cases

Common requirements for edge workloads

1. lowest latency between data and responses and decisions
2. pre-processing (reduction) before data moves to cloud,
3. remotely managed datasets for local access
4. remotely manage software deployment and updates
5. operate offline or with intermittent connectivity

	1	2	3	4	5
Remote office, retail			✓	✓	✓
Sensor data collection, analytics	✓	✓		✓	✓
Physical device control	✓			✓	✓
Gaming	✓	✓	✓	✓	
Telco edge cloud	✓	✓	✓	✓	

Edge Compute benefits

- Remote office, retail
- Telco Infrastructure
- Large scale IoT and IIoT
- Video games
- VR/AR
- AI/ML
- ….

IoT Edge Working Group

- Community to discuss all the topics around IoT and Edge Computing with K8S
- What kind of use cases, workloads and applications are common to running K8s on the Edge?
- Why and how Kubernetes can be a good fit to these Edge scenarios?
- What are the key challenges for deploying of Kubernetes to Edge today?

What is Edge Computing?

- Edge - everything that is not cloud
- Effort to deploy workloads closer to the users and devices

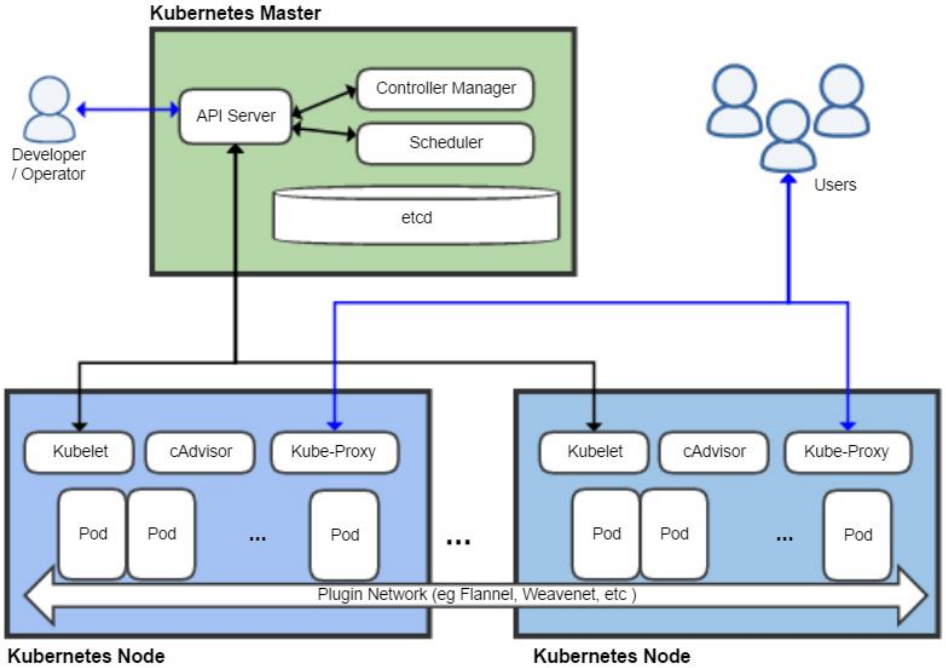
Why Edge Computing?

- Latency
- Availability
- Data locality
- ...

Why Kubernetes?

- Kubernetes API
  - Developers mindshare and tooling
  - Same workloads can move between Cloud and Edge
- Kubernetes Architecture
  - No need to reinvent it

Kubernetes Architecture



Edge Workloads

Edge workloads - Why?

- Data ingestion and processing
  - Protocol conversion
  - Data preprocessing
- Reliability and availability
  - Buffer and batch
  - Caching
- Latency
  - Edge functions
  - Compute offloading
  - Machine learning

Protocol conversion

- Network level
  - Converting non-IP protocols to TCP/IP based ones
    - Modbus in IIoT
    - Bluetooth in consumer IoT
  - Usually converting to some widely used messaging protocol
    - MQTT
    - AMQP
    - HTTP

- Kubernetes supports "device plugins"
- Taints and tolerances can be used for scheduling to appropriate nodes

Data preprocessing

- Convert data to general structured messages
- Normalize data structure
  - Vorto, LWM2M
- Data analytics
  - Send only relevant data
  - Combine multiple sources
- Add metadata
  - Location
  - Identity
  - Security

Generic Kubernetes workloads

Needs to be properly containerized and orchestrated on the Edge nodes

Reliability and high availability

- Buffer and batch
  - Store and forward
  - Brokers on Edge nodes
- Caching
  - Local databases on Edge nodes
  - Sync data with the cloud and other Edge nodes

Edge Nodes/Clusters may have limited storage volumes to hold data until it can be uploaded

Latency: Functions

- React locally on sensor or scheduled events

- Possible CNCF projects collaboration
  - Cloud Events - https://cloudevents.io/
  - Knative - https://github.com/knative/

Latency

- Compute offload
  - Schedule resource intensive tasks on the dedicated hardware on the Edge
  - Example AR/VR renderings
- Machine learning
  - Cloud trained models - executed on the Edge
  - Edge specific training (environment and data policies)

Taints and tolerances can be used for scheduling to appropriate nodes (e.g. GPU availability)

Challenges

Edge Challenges

- Infrastructure
  - How to manage resources (nodes and clusters) on the Edge?
- Control plane
  - How to manage workloads on the Edge?
- Data plane
  - How Edge sites communicates with the cloud and between themselves?

Challenges

- Resource constraints
- Network limitations
- Unattended operation
- Physical security

Network

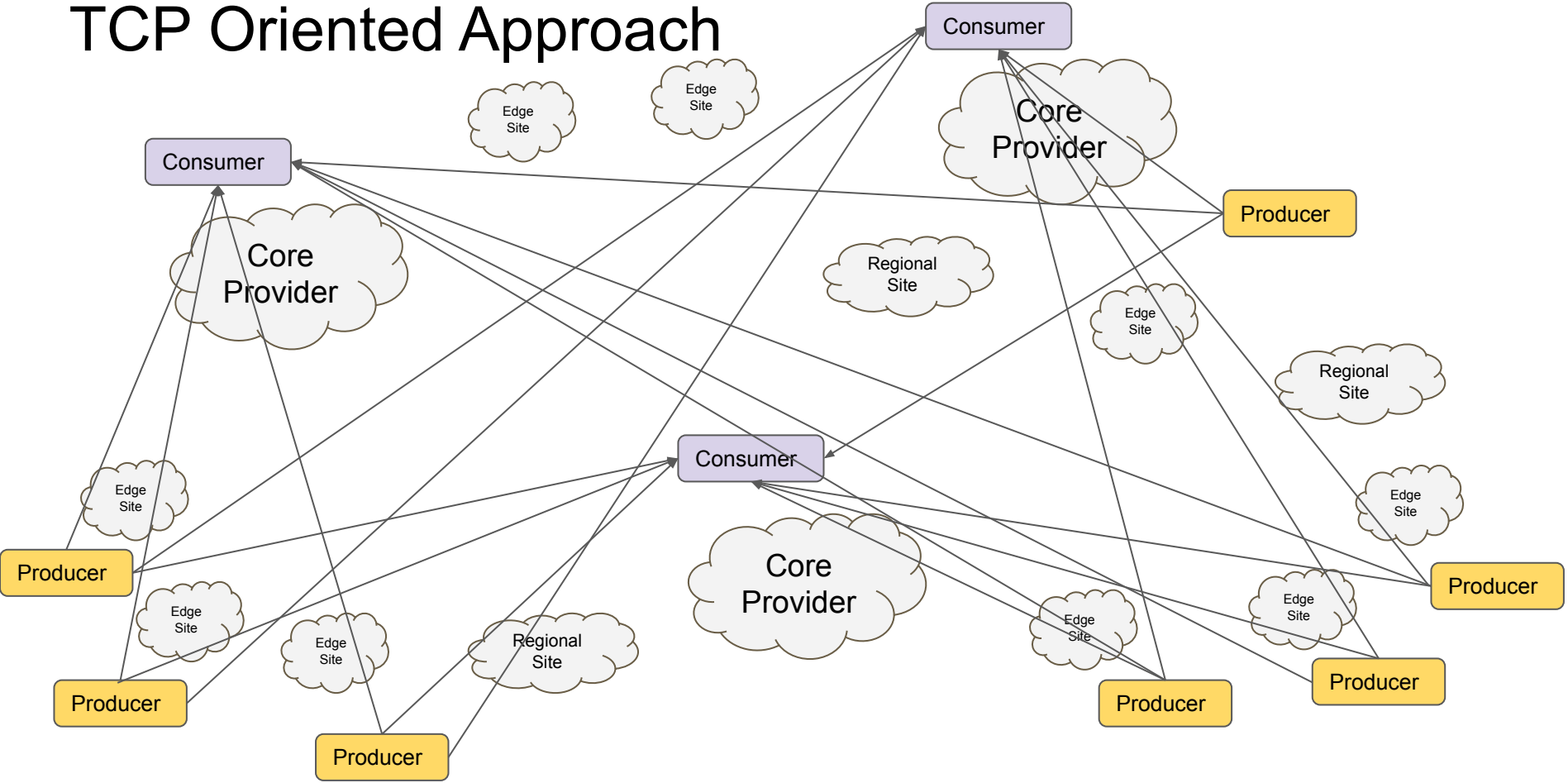
Challenges facing and not necessary having solutions

- Network bandwidth and reliability
- Connectivity
  - Between edge and cloud
  - Between edge nodes
- Network routing
  - North – South
  - East – West
- Discovery
- Network policy and access control
- K8s flat network requirement

Deep dive

Data plane

TCP Oriented Approach



AMQP 1.0 Features

- Middleware: application level protocol
  - Not O.S. dependent (aside from TCP)
- Support for common Messaging Patterns:
  - Request/Response (RPC)
  - Fan Out (Pub/Sub, Topics)
  - Queuing (Store and Forward)
- Strict Flow Control
- Peer to Peer protocol
  - Intermediaries NOT required (but allowed)

AMQP 1.0 Features

- Security at the Connection Level
  - Encryption (TLS)
  - Authentication (SASL)
- Common Type System (binary)
  - Integers, strings, floats, timestamps, UUID, …
  - Polymorphic Container Types
    - Maps (string and integer keys)
    - Lists
  - Self-describing
    - No need for pre-shared schema

AMQP 1.0 Features

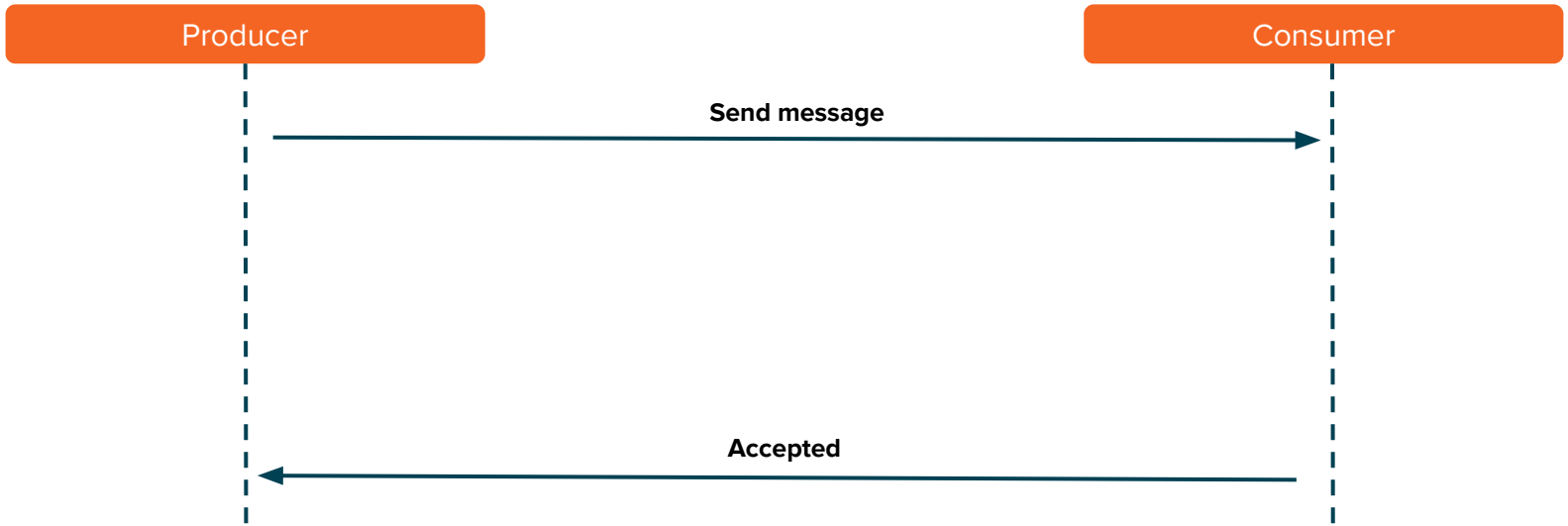
- Receipt Confirmation
  - End to End Acknowledgment
- Application defined Addressing (Layer 7)
  - Separate from Network Addressing
  - Simple UTF-8 Strings
- Efficient Use of Networking Resources
  - Multiplexing Multiple Message Flows per Socket
  - Async Message Transfer
- Per Message Priority

AMQP 1.0 Infrastructure Components

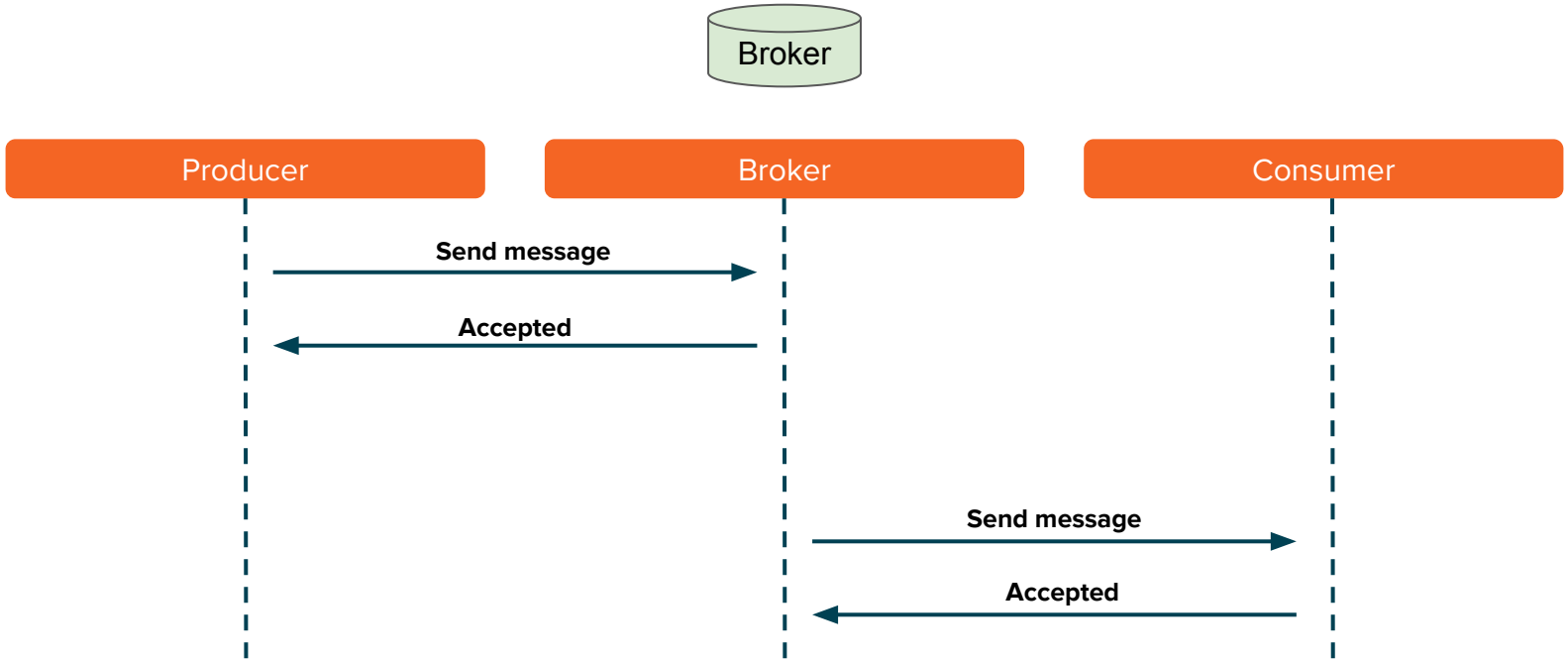
- Apache ActiveMQ Artemis (http://activemq.apache.org/artemis/)
  - Classical Broker provides queuing and pub/sub services
  - Best in class performance, rich feature set
  - AMQP 1.0, MQTT and JMS 2.0 support
- Apache Qpid-Dispatch-Router (https://qpid.apache.org/components/dispatch-router/)
  - Stateless “bump in the wire” AMQP 1.0 message router (no message queueing)
  - Dynamically learns addresses of messaging endpoints
  - Shortest-path messaging forwarding and end-to-end transfer of message ownership
    - High throughput, deterministic low latency
  - Common delivery patterns (anycast, multicast) and load balancing
- Apache Qpid-Proton (https://qpid.apache.org/proton/)
  - High performance messaging library
  - Go, java, c++, python, ruby

Routing vs Brokering

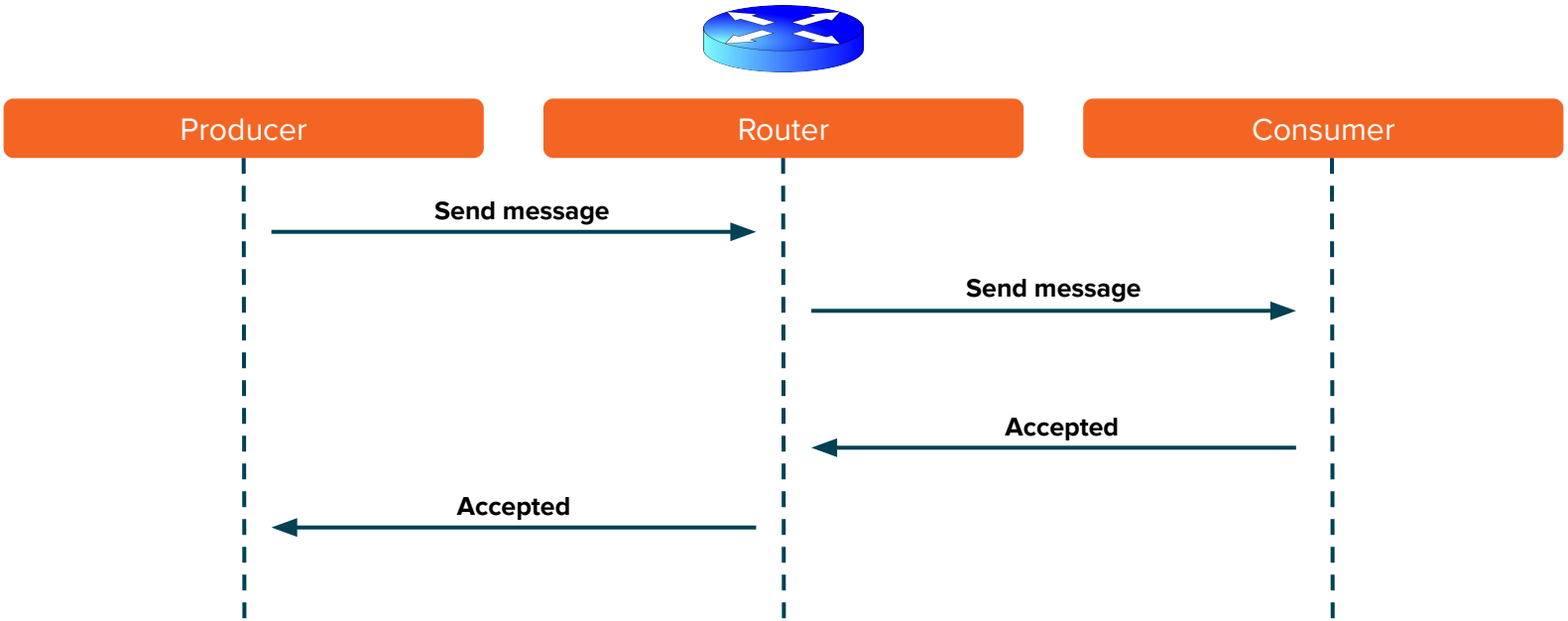
Direct



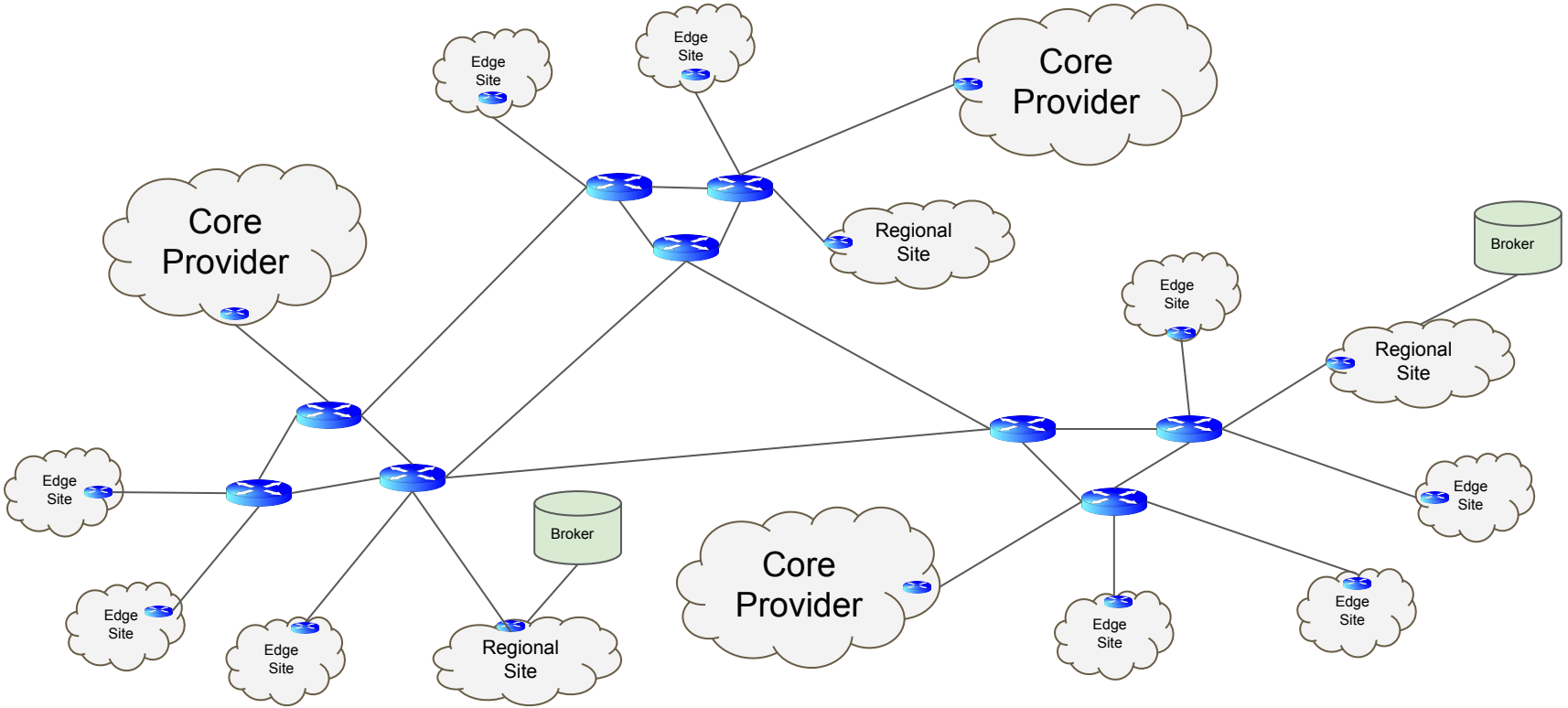
Routing vs Brokering



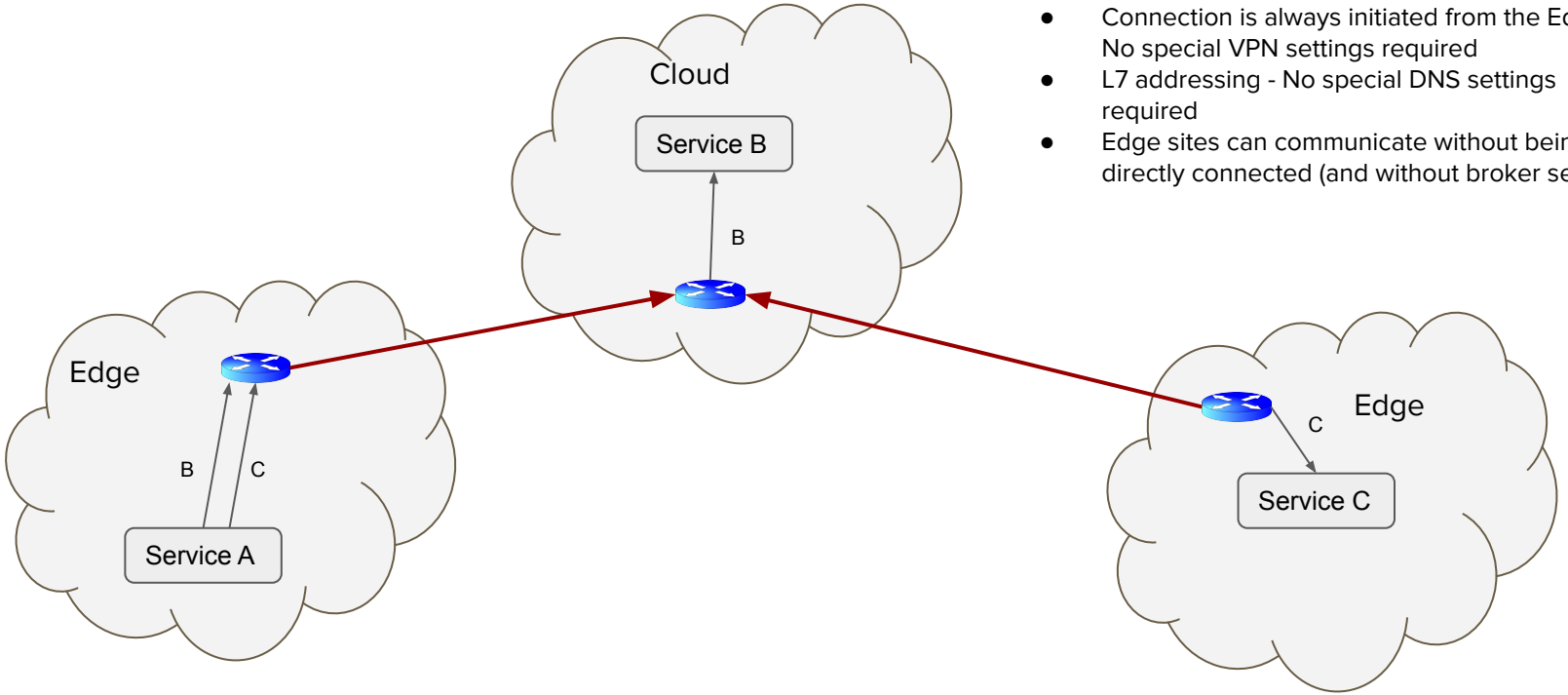
Routing vs Brokering



Router Based Multi Region Message Bus

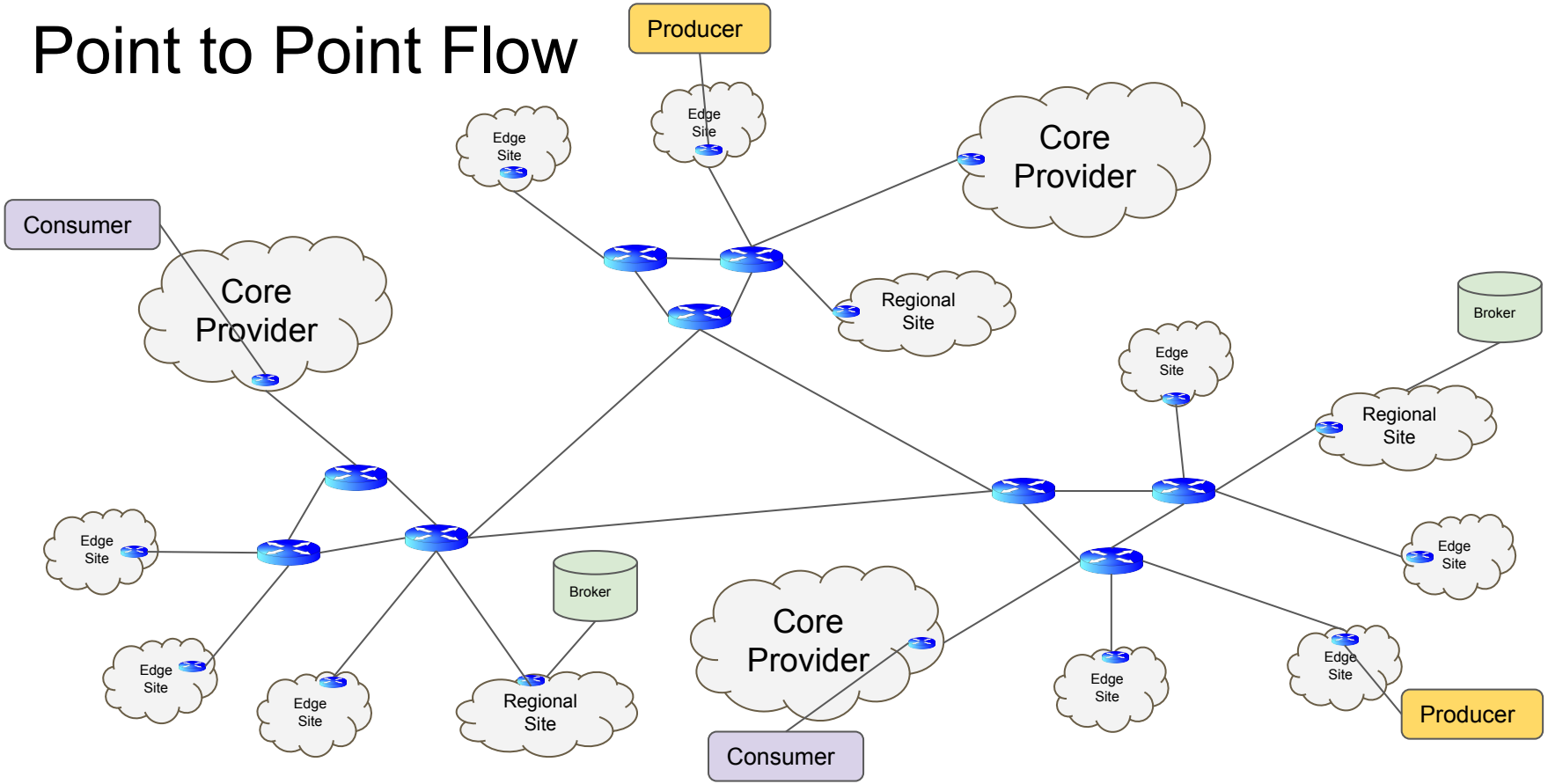


Secure multi-site communication

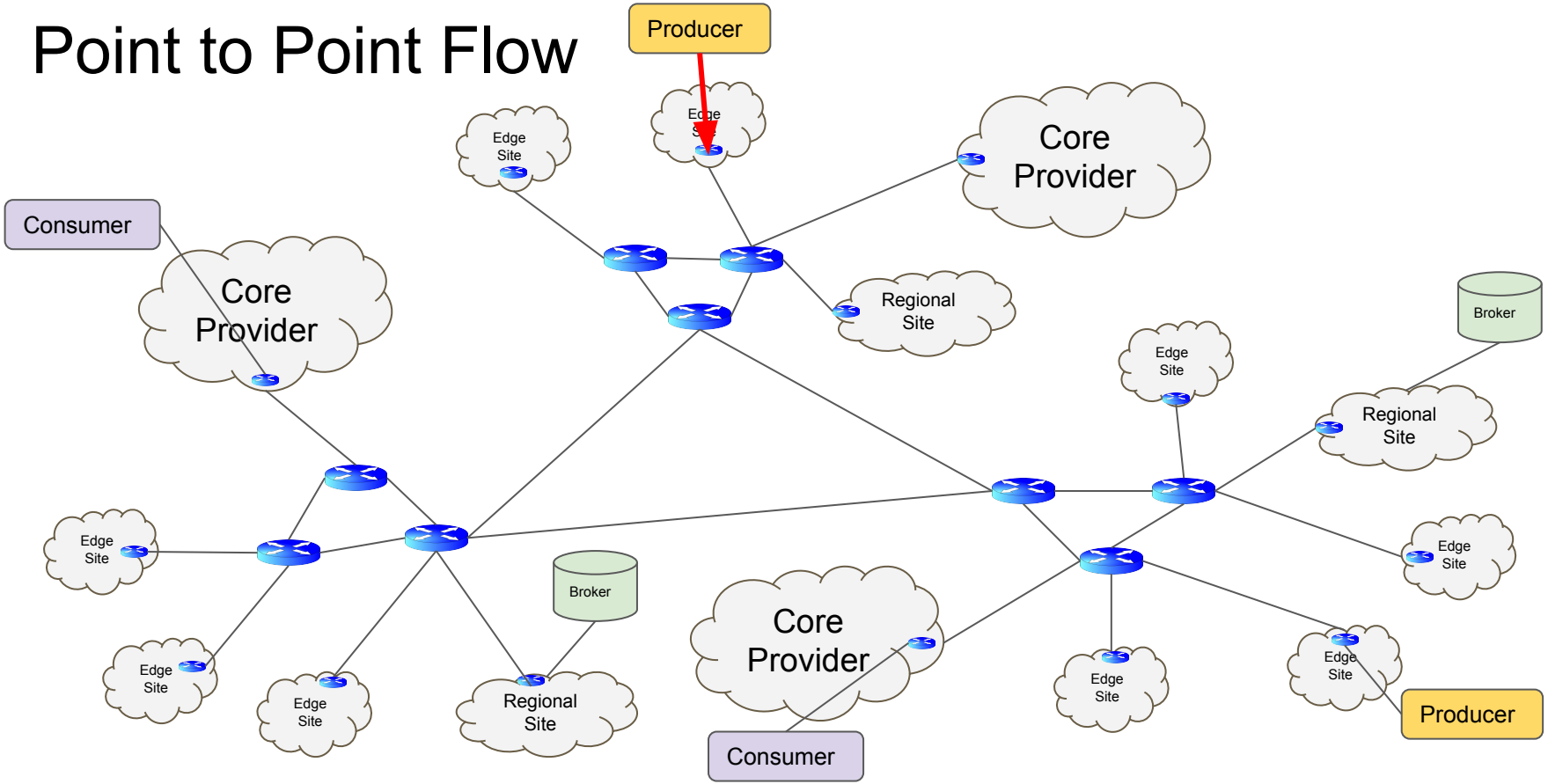


- Connection is always initiated from the Edge - No special VPN settings required
- L7 addressing - No special DNS settings required
- Edge sites can communicate without being directly connected (and without broker service)

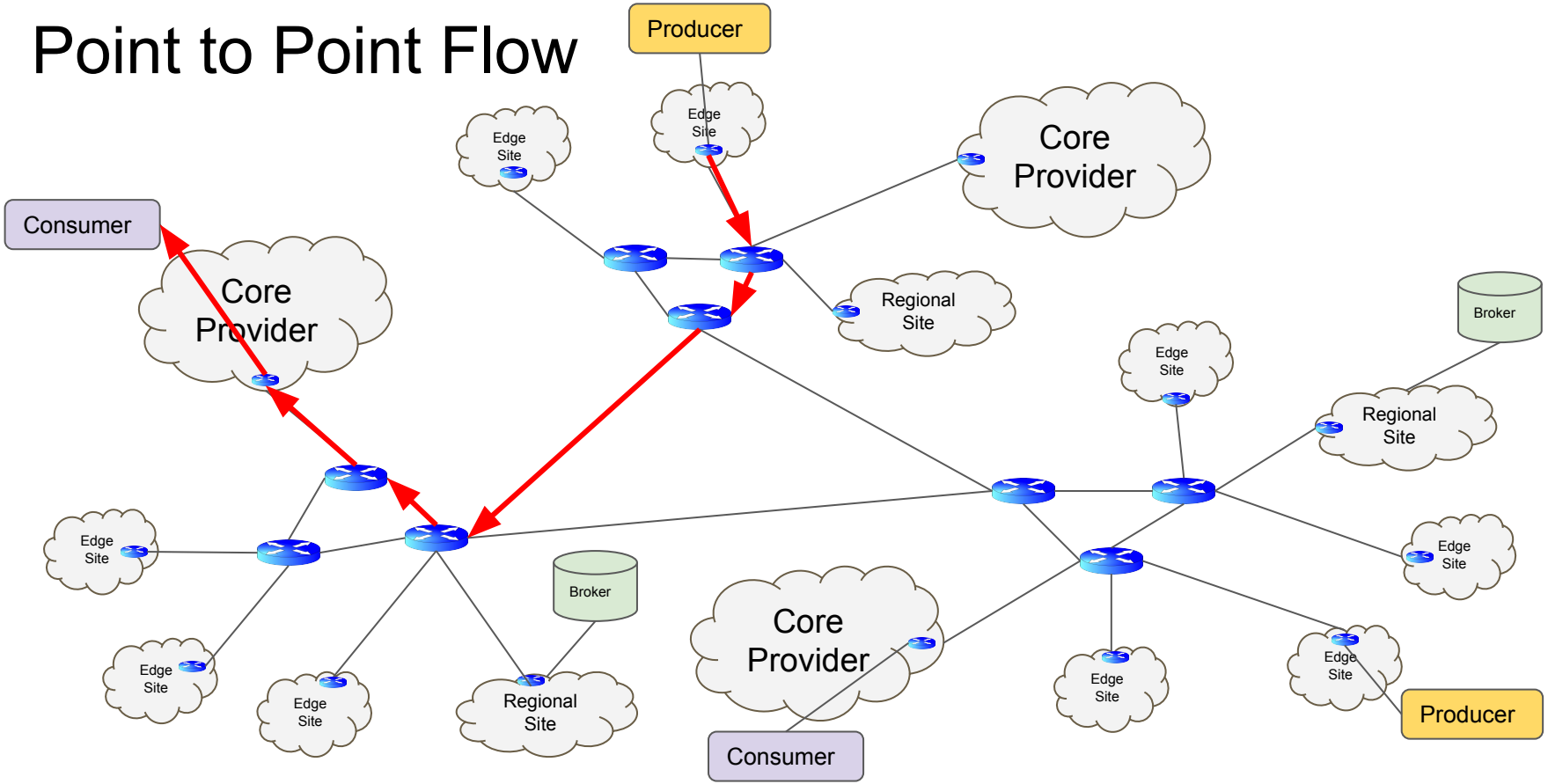
Point to Point Flow



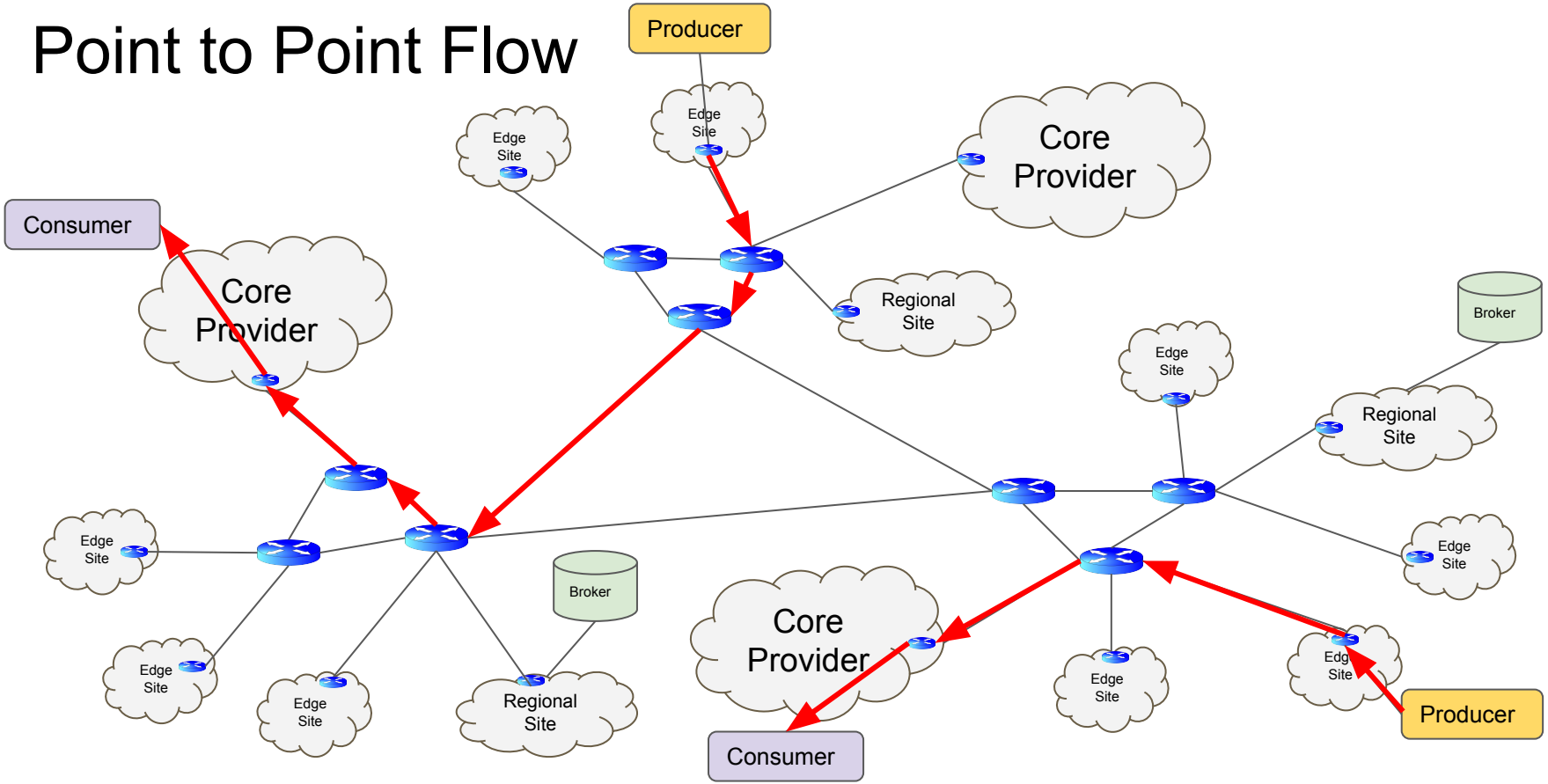
Point to Point Flow



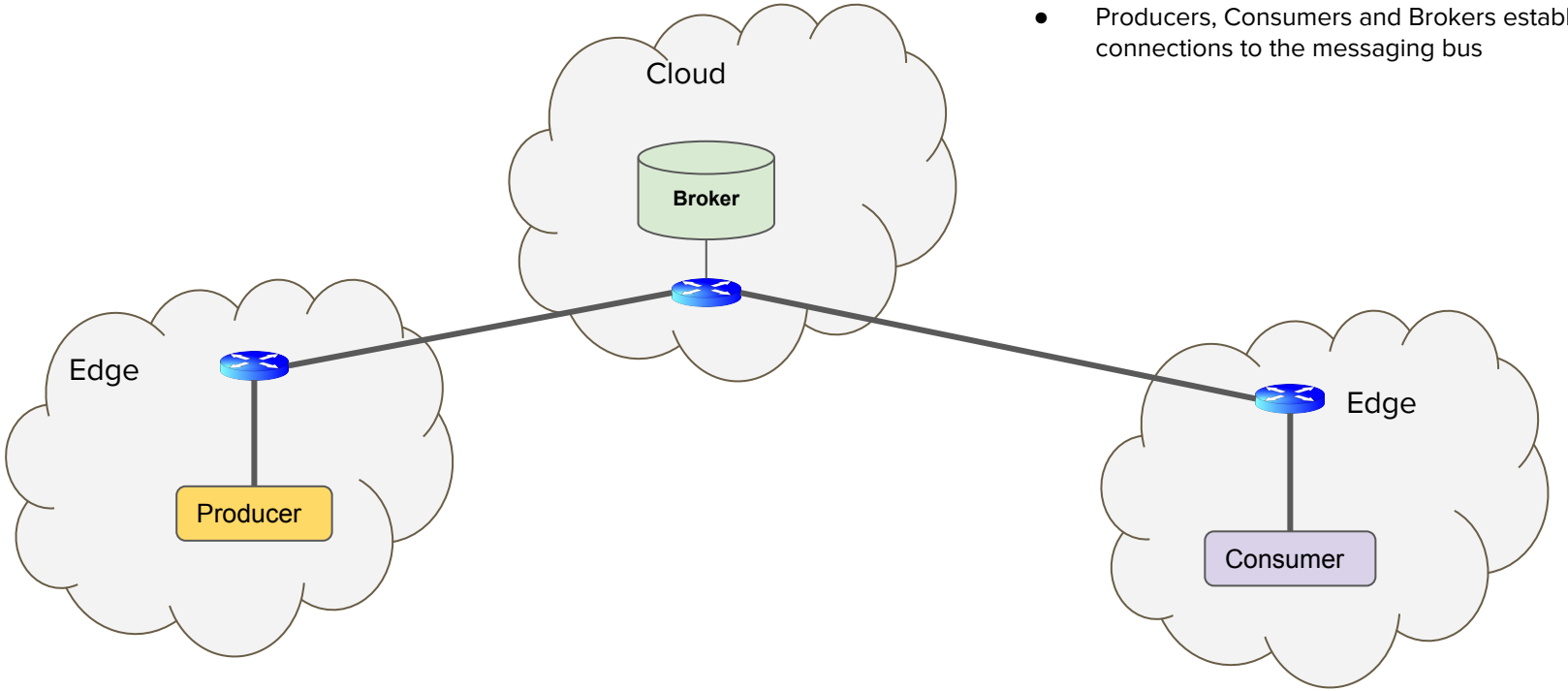
Point to Point Flow



Point to Point Flow

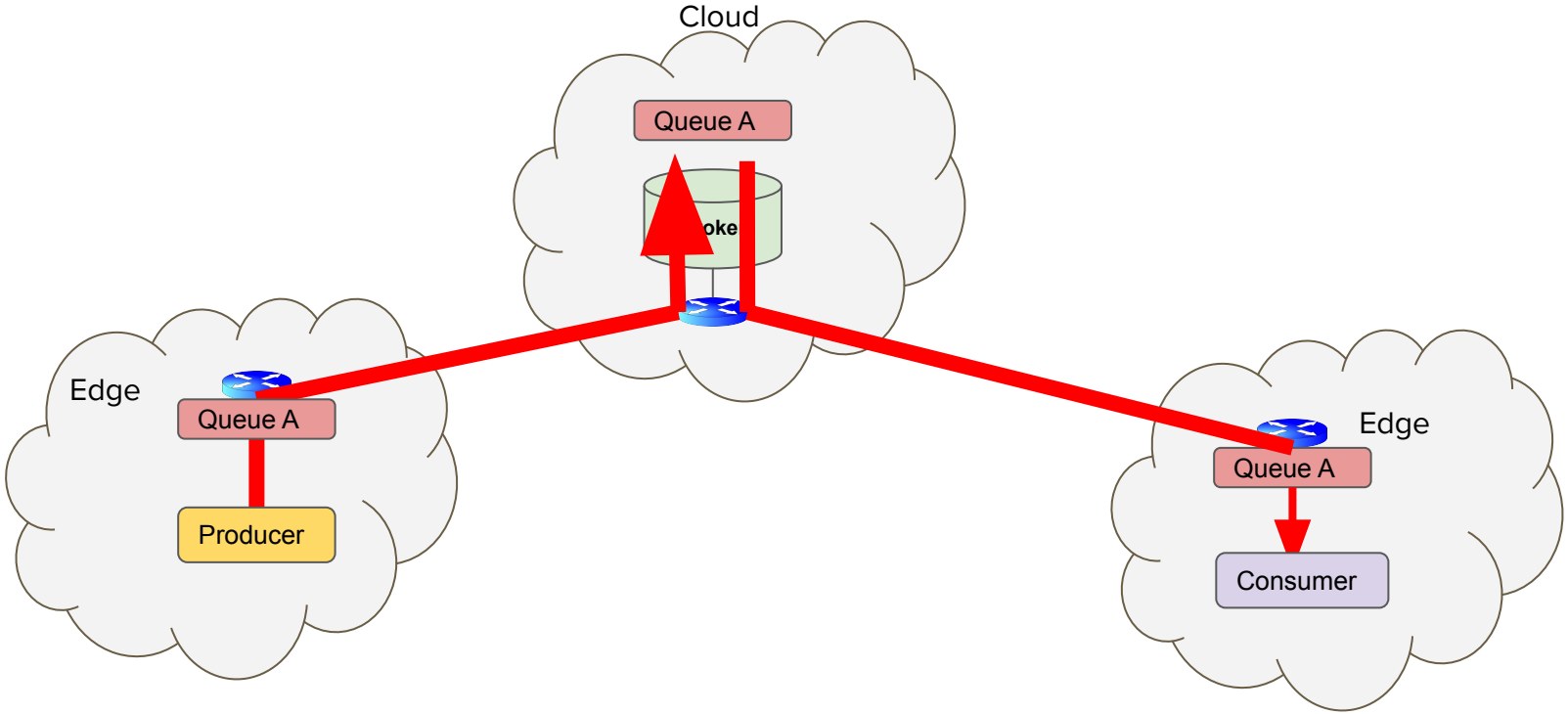


Message Queuing Services

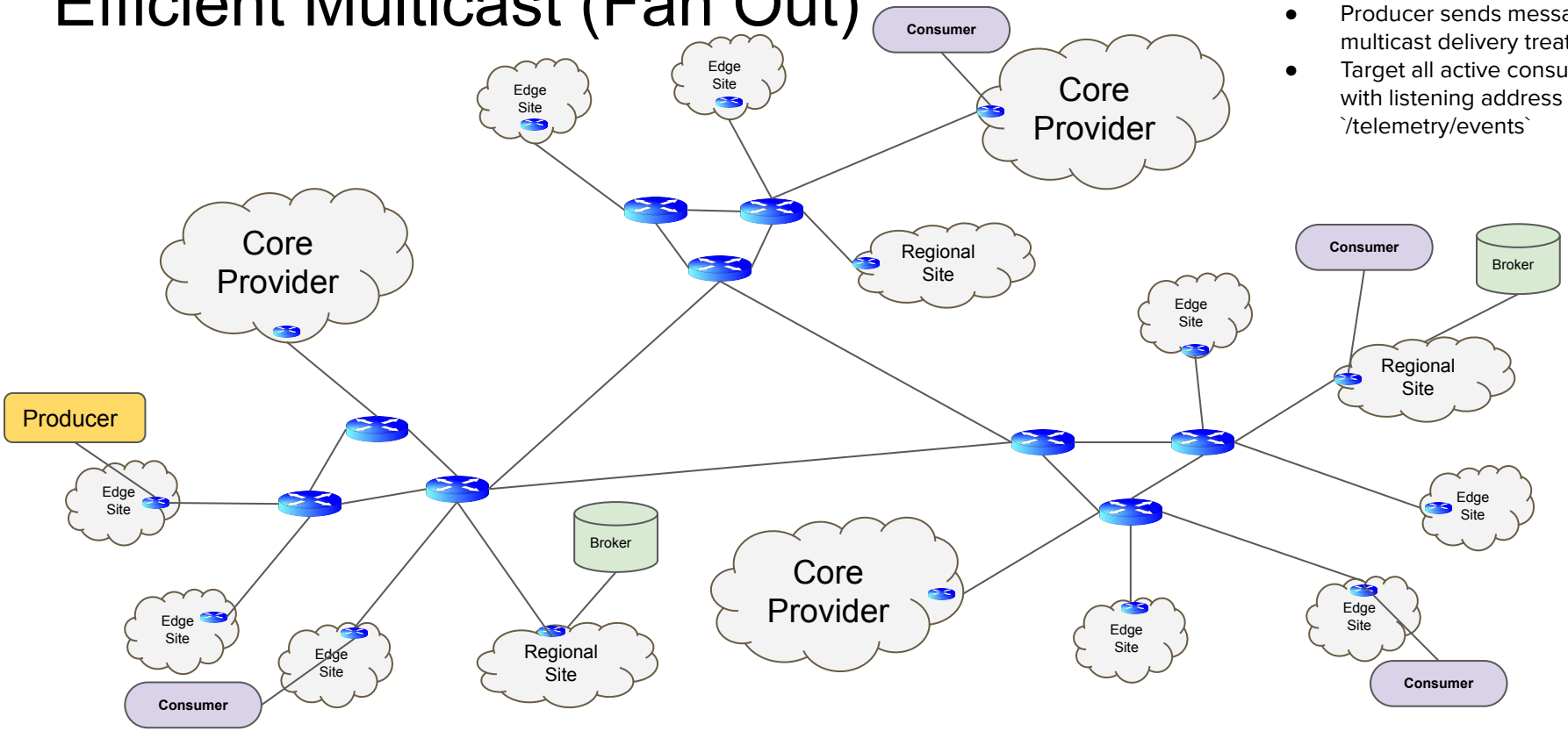


- Producers, Consumers and Brokers establish connections to the messaging bus

Message Queuing Services

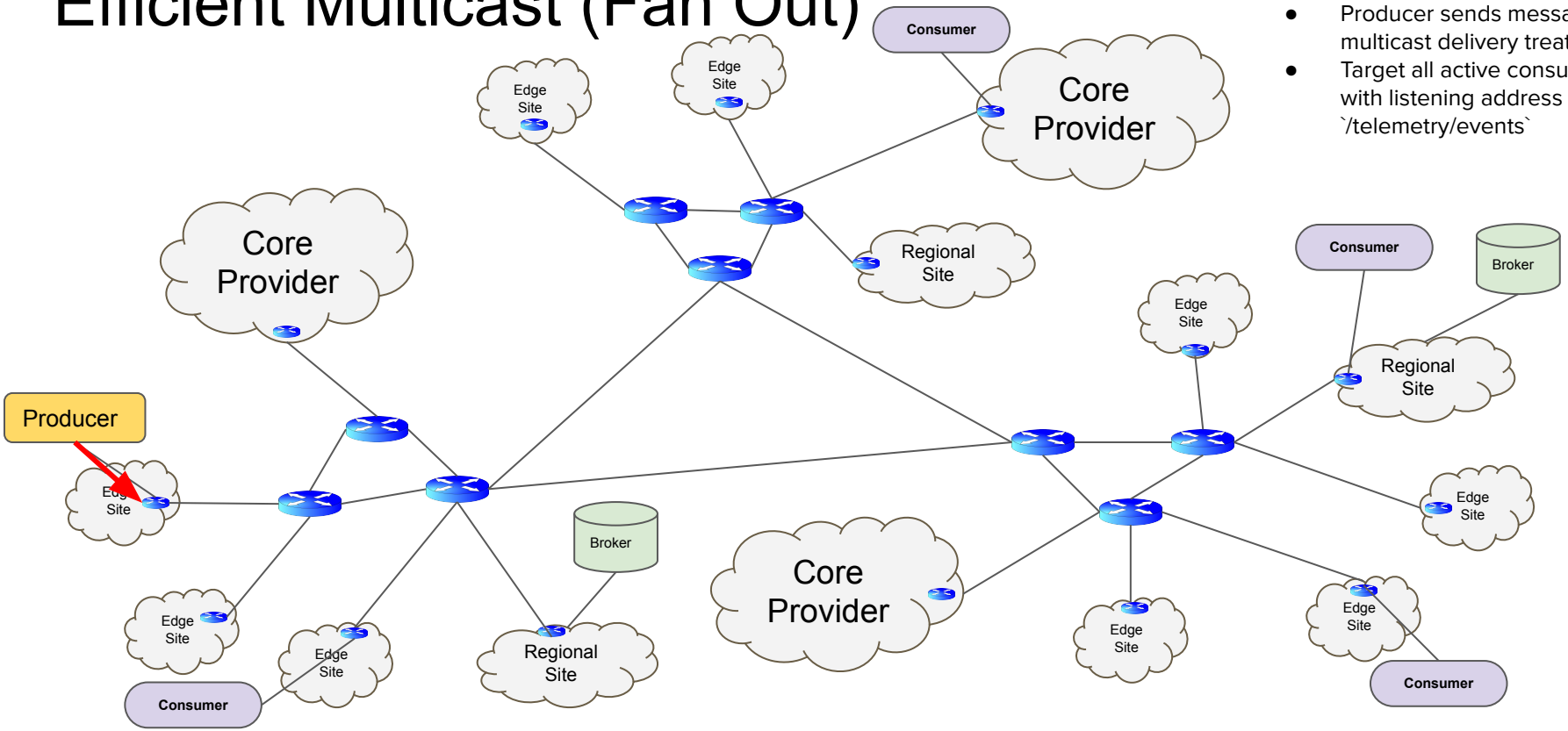


Efficient Multicast (Fan Out)



- Producer sends message with multicast delivery treatment
- Target all active consumers with listening address (e.g. `/telemetry/events`

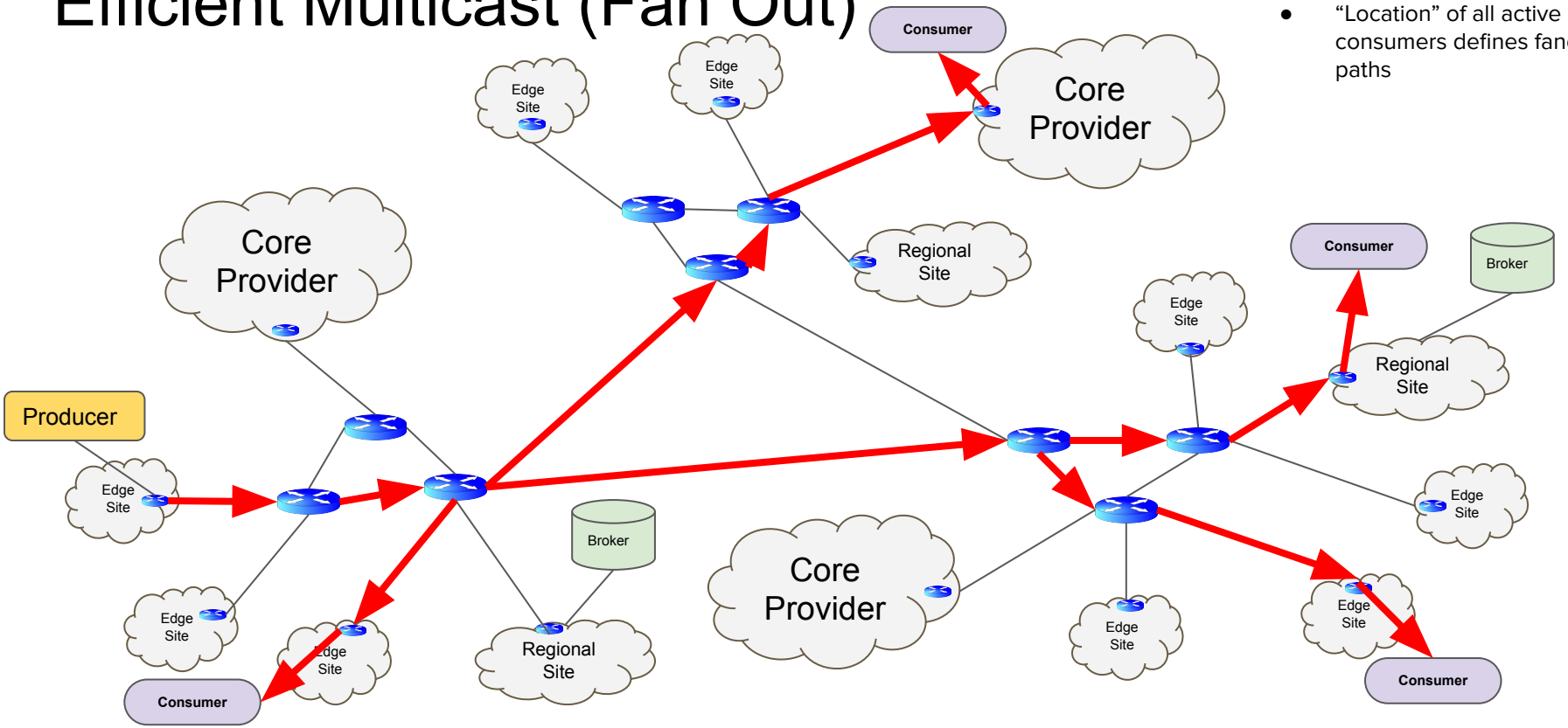
Efficient Multicast (Fan Out)



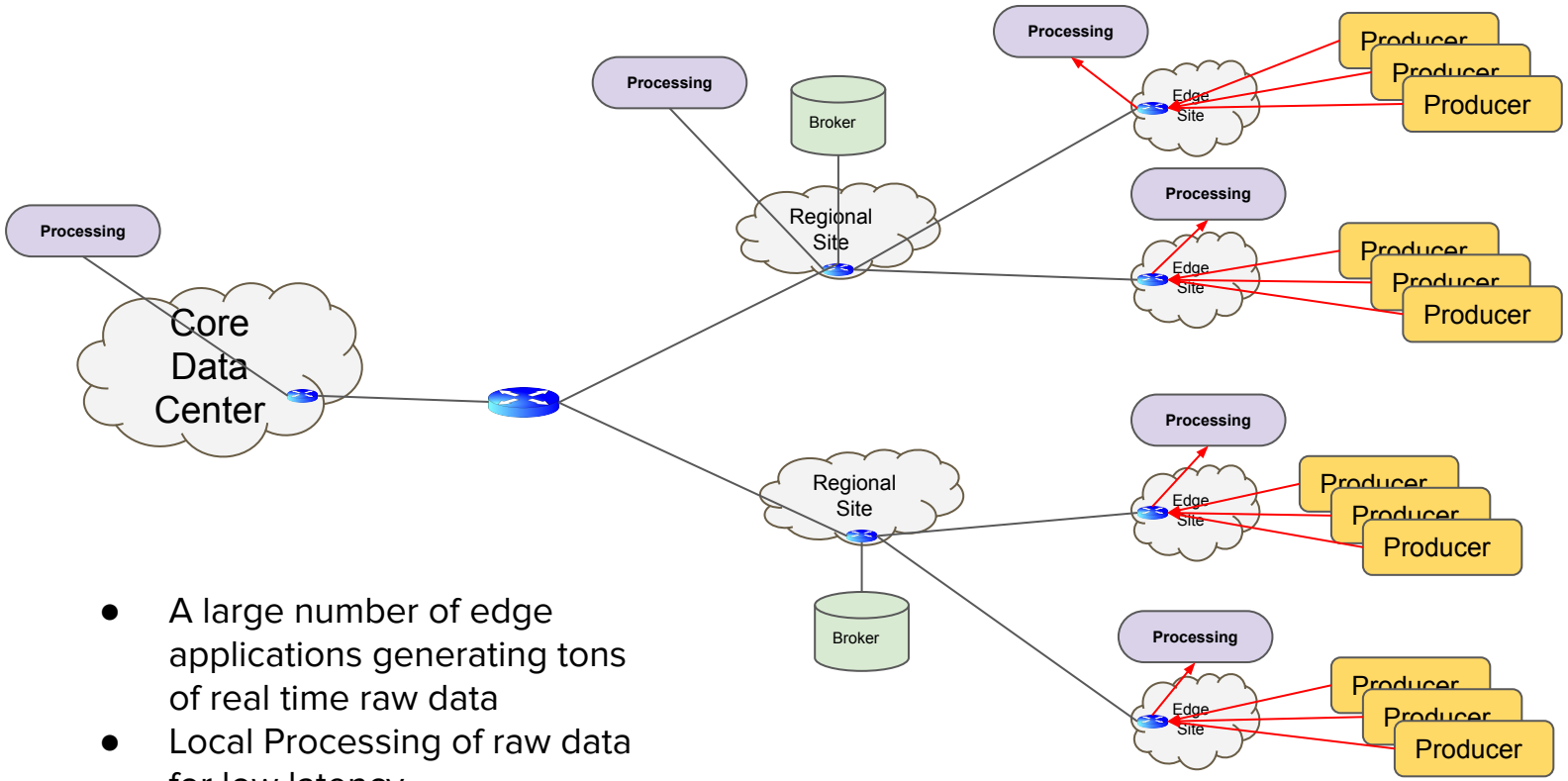
- Producer sends message with multicast delivery treatment
- Target all active consumers with listening address (e.g. `/telemetry/events`

Efficient Multicast (Fan Out)

- “Location” of all active consumers defines fanout paths

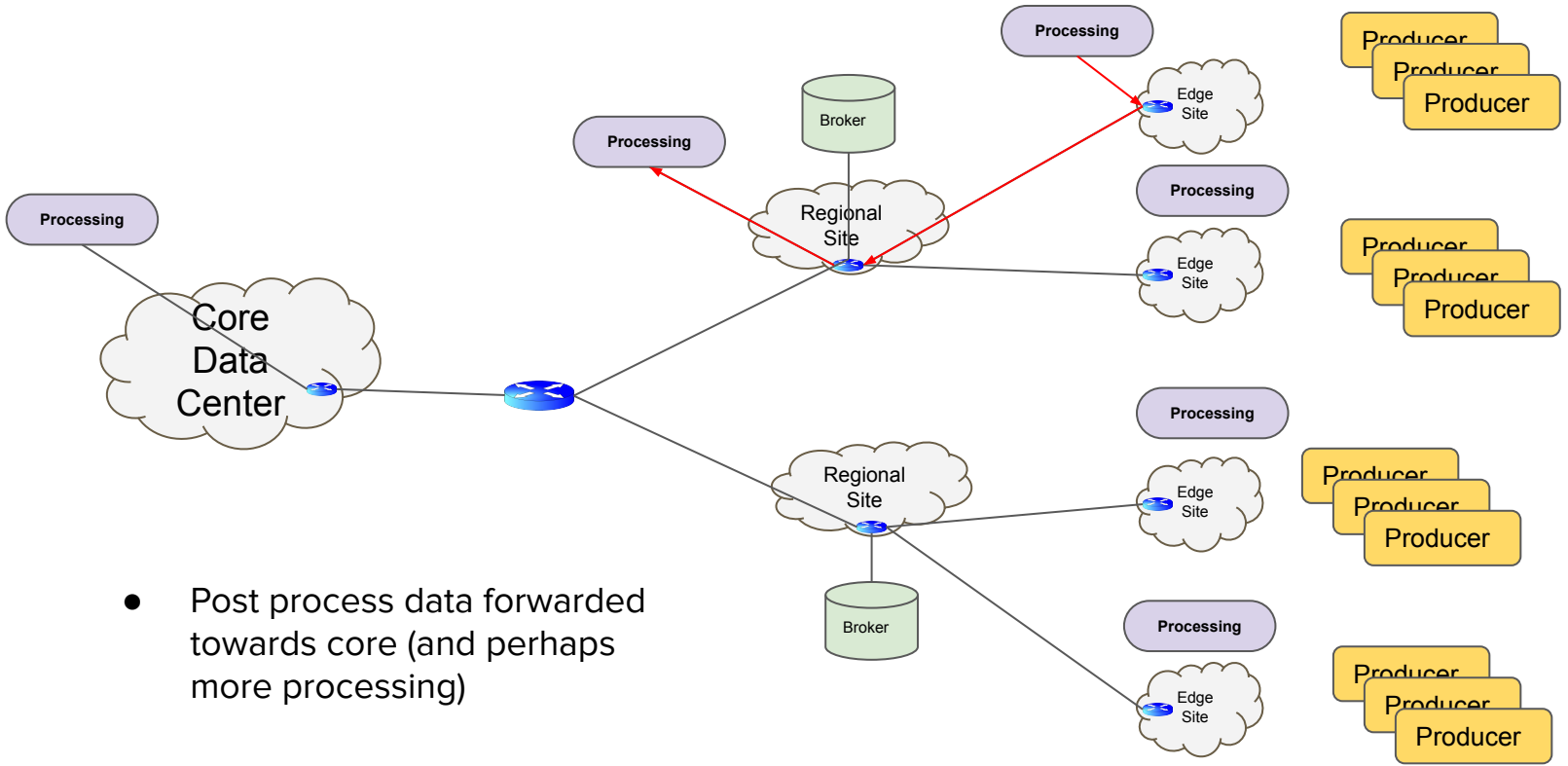


Fan In (Post Processing & Unreliable Uplinks)



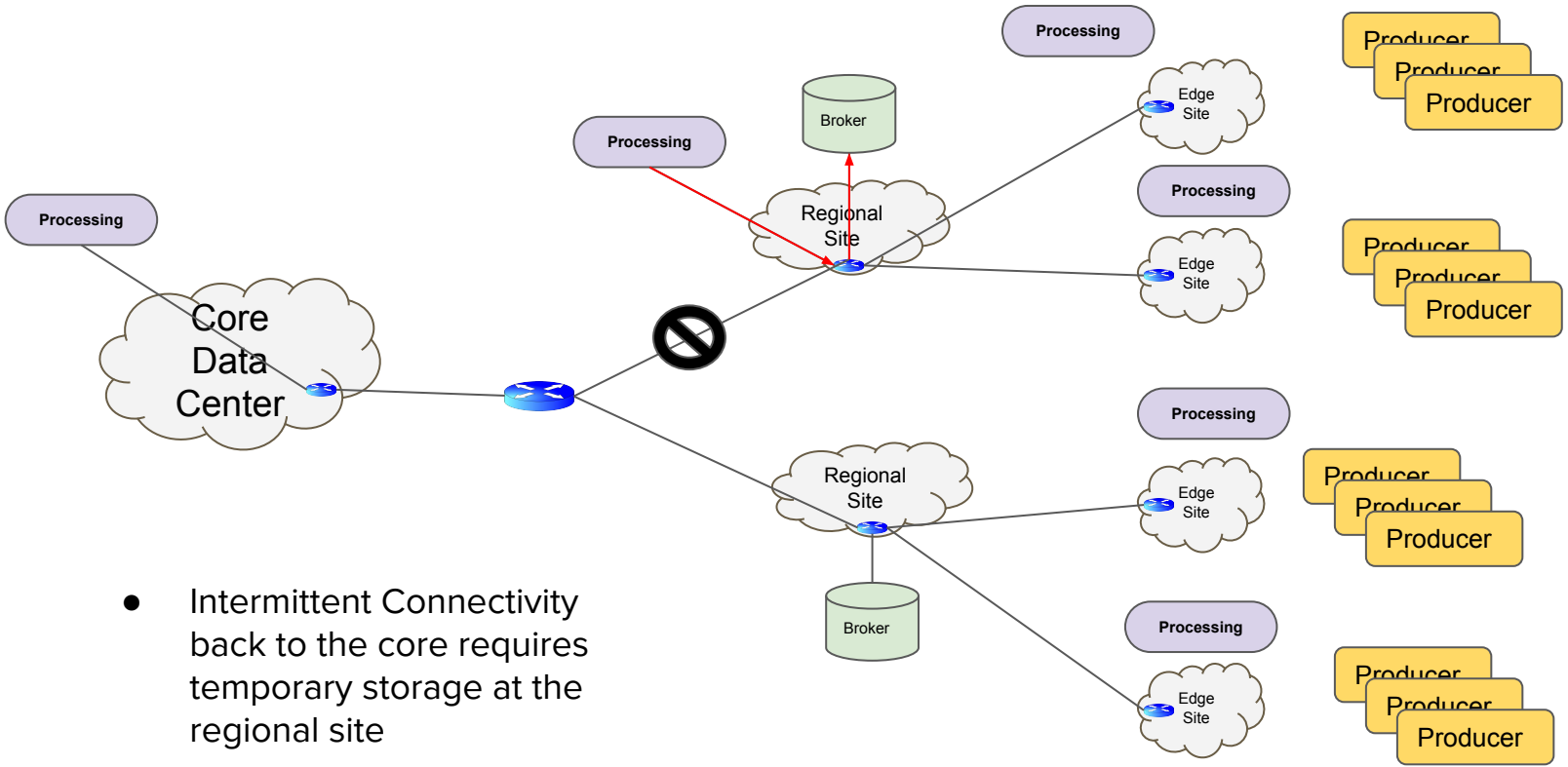
- A large number of edge applications generating tons of real time raw data
- Local Processing of raw data for low latency

Fan In (Post Processing & Unreliable Uplinks)



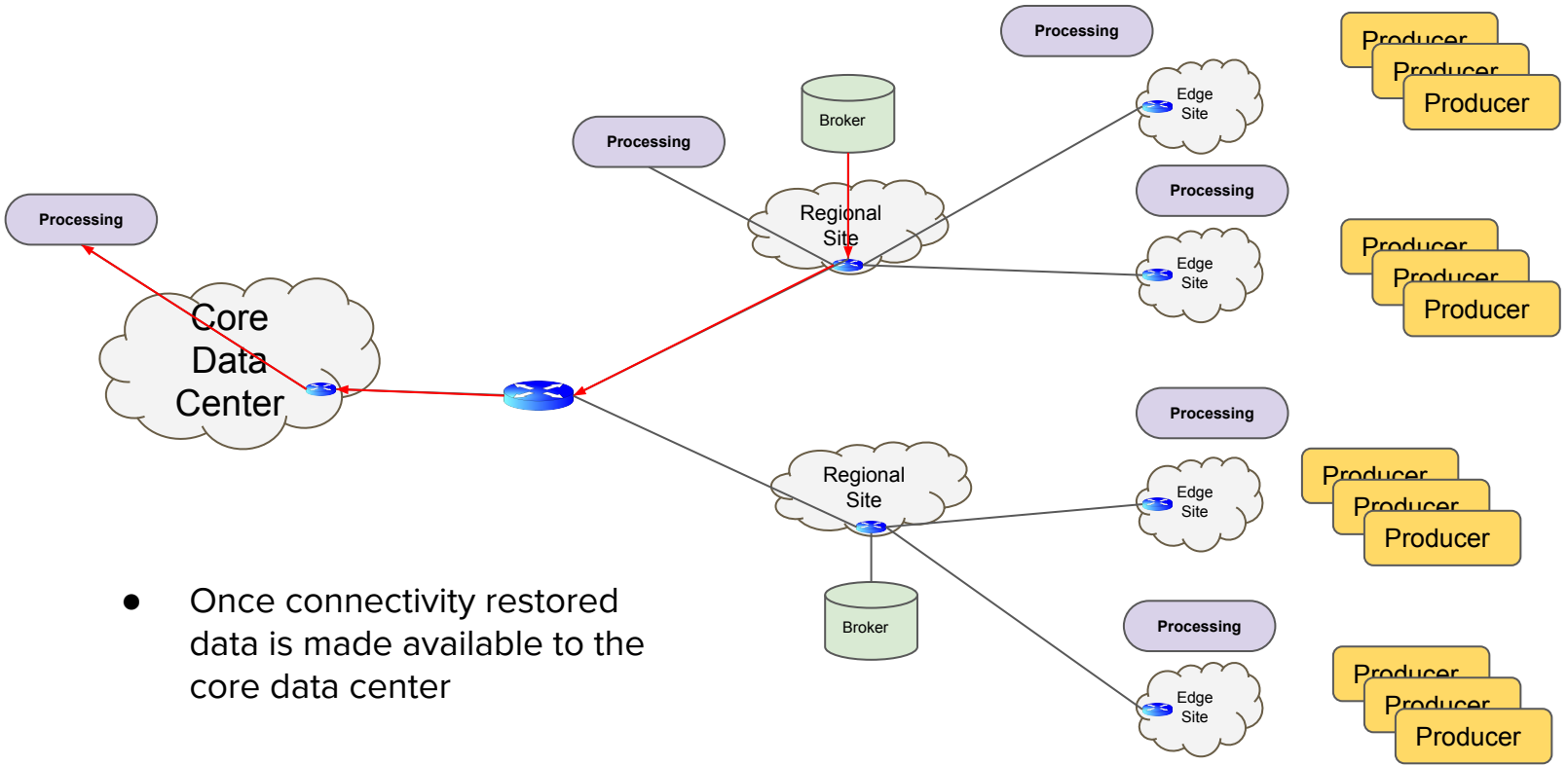
- Post process data forwarded towards core (and perhaps more processing)

Fan In (Post Processing & Unreliable Uplinks)



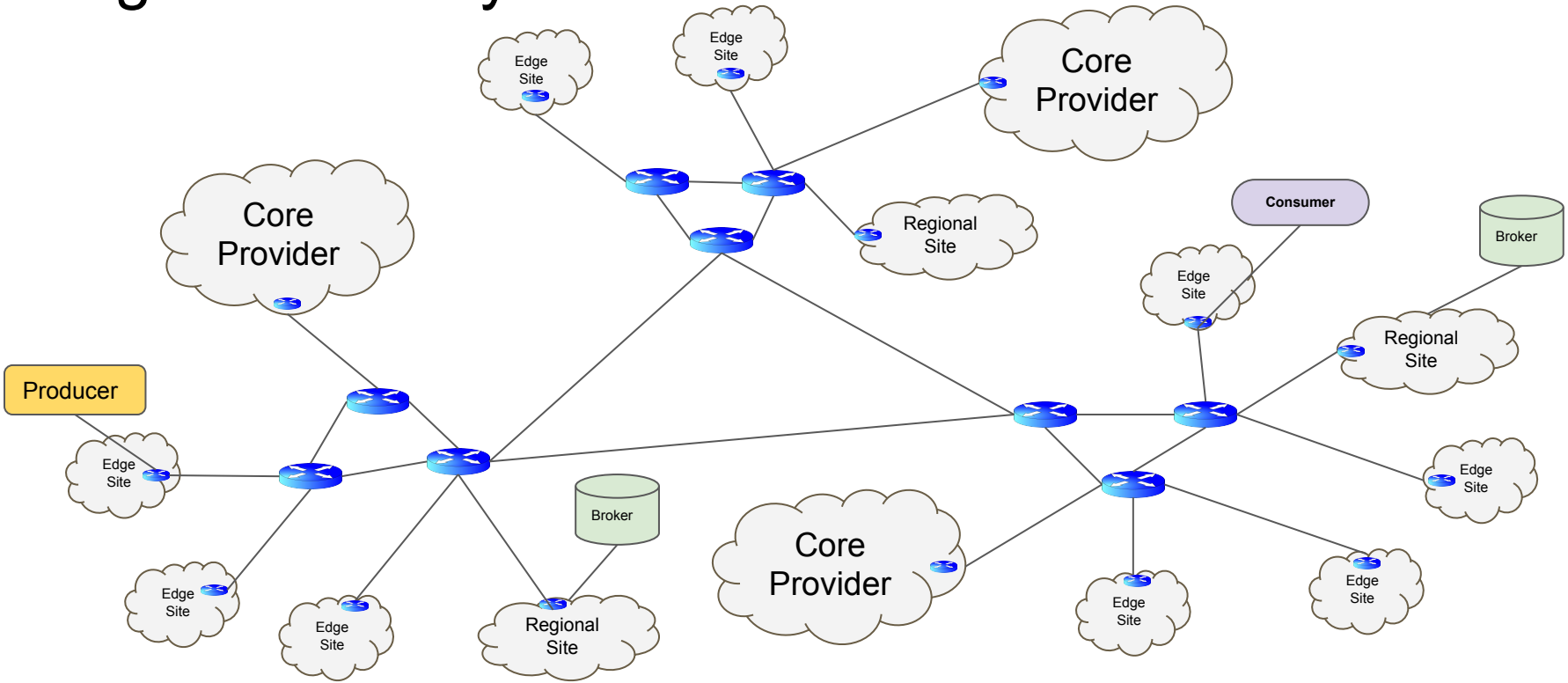
- Intermittent Connectivity back to the core requires temporary storage at the regional site

Fan In (Post Processing & Unreliable Uplinks)

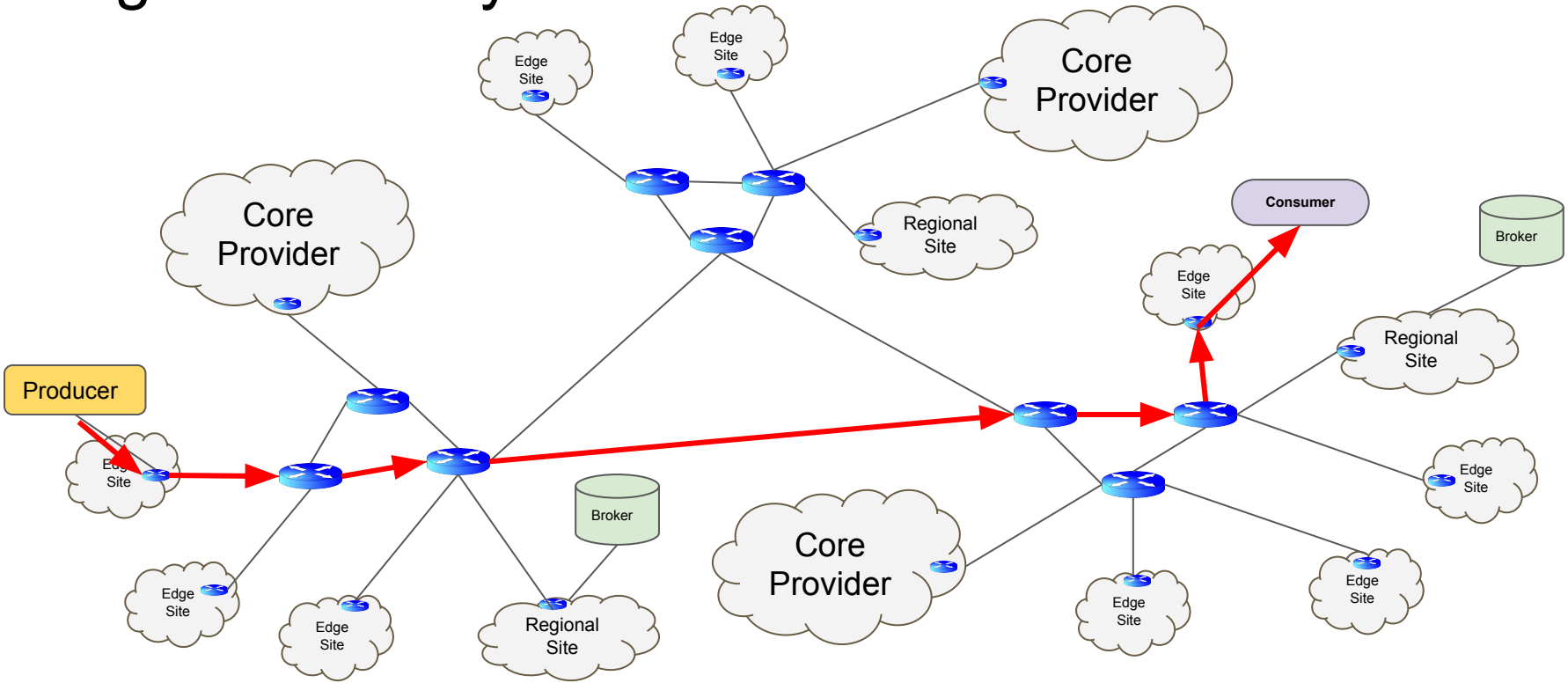


- Once connectivity restored data is made available to the core data center

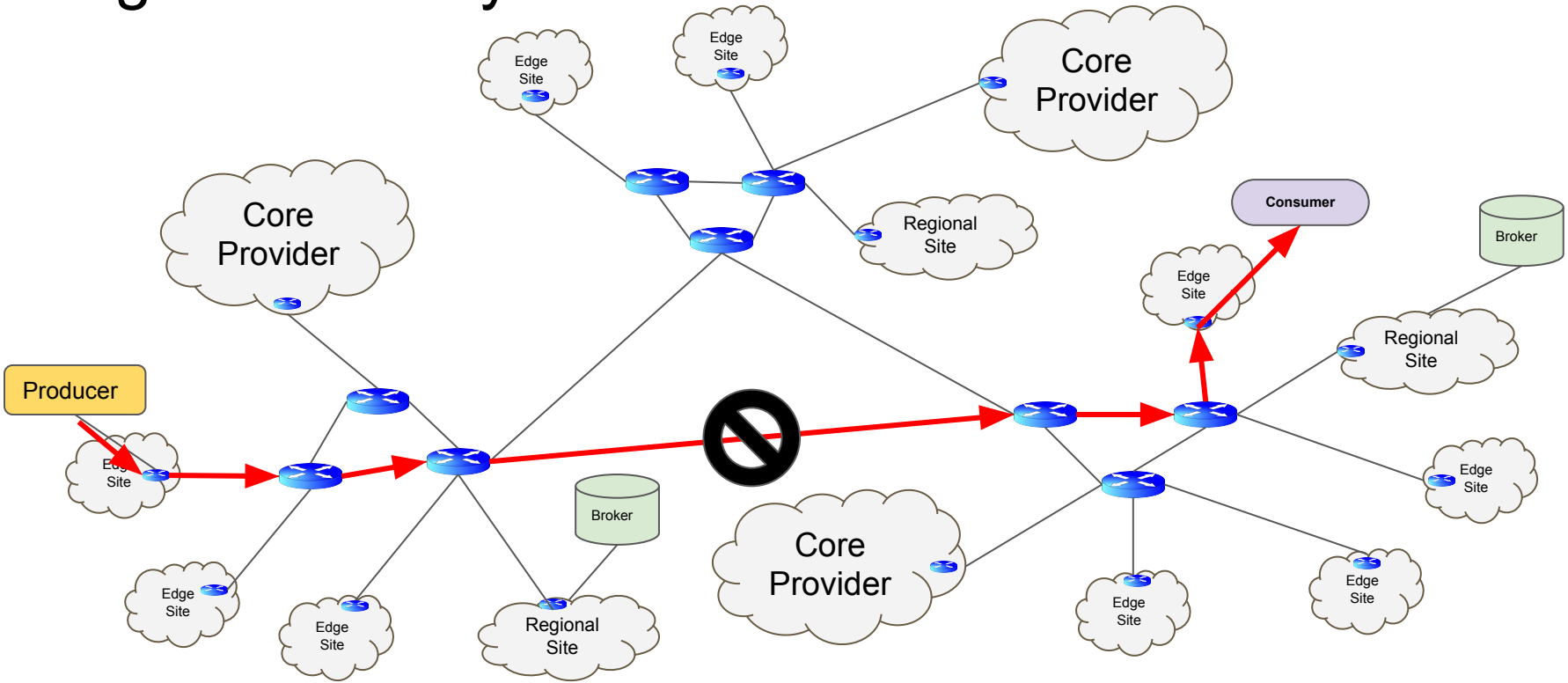
High Availability



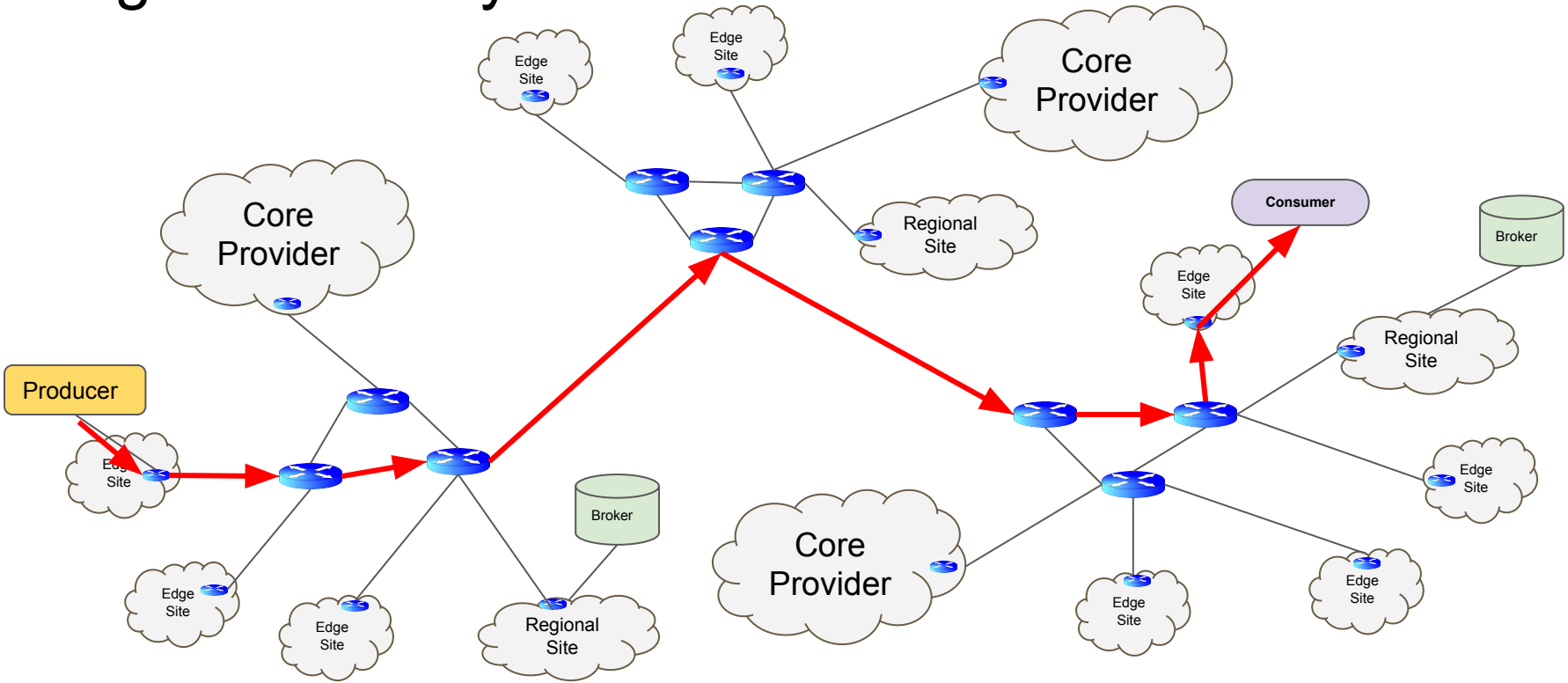
High Availability



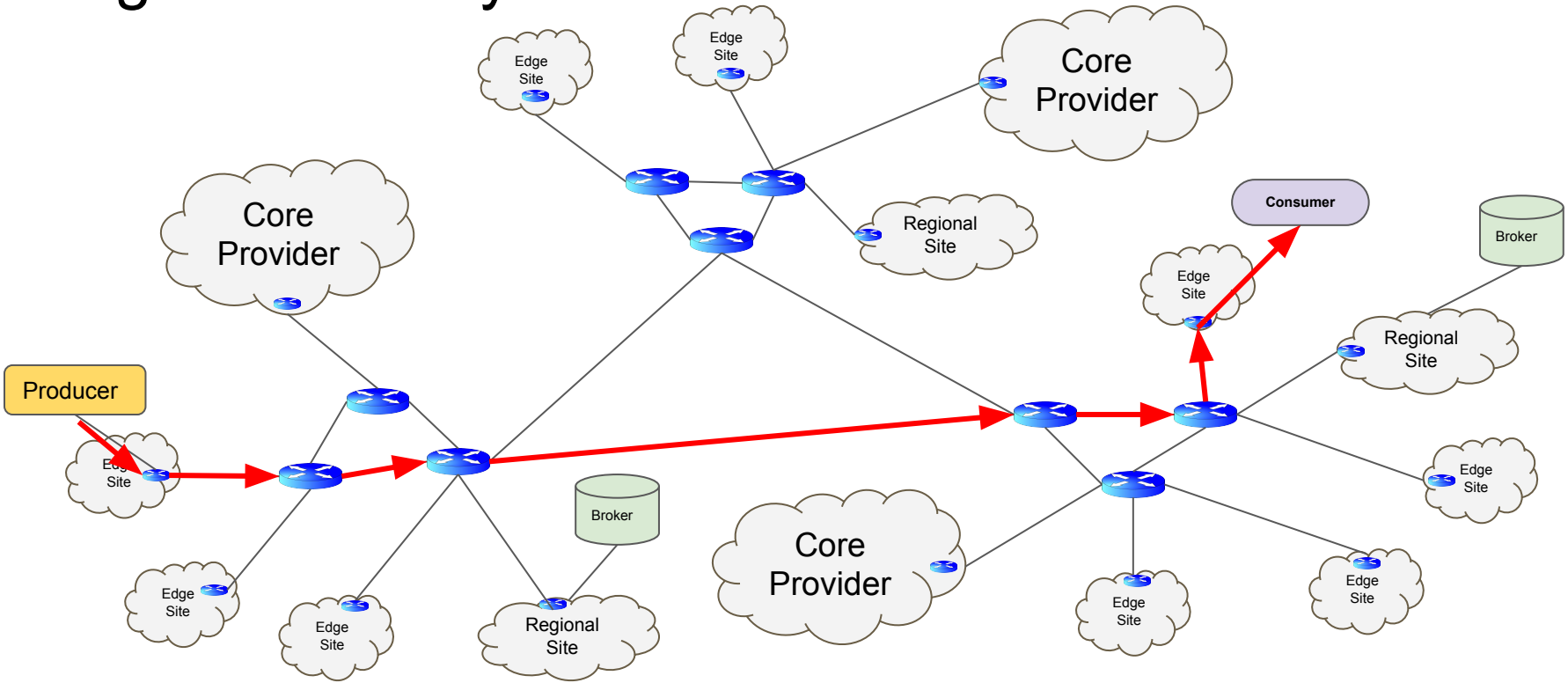
High Availability



High Availability



High Availability



Data plane demo

Security

Edge Environment Differences

- Diverse networks
- No guarantees of continuous power
- Intermittent connectivity
- Direct physical access to hardware
- Heterogeneous hardware
- Non-TCP/IP communication
- Multiple vendors in a single solution
- Need to handle security in “offline mode”
- Low latency locally, higher latency to cloud

Edge Security Challenges - Overview

- Trusting edge hardware
- Trusting connected devices
- Operating systems
- Network concerns
- Edge microservices

Trusting Edge Hardware

- Physical security is not guaranteed at the edge...

Questions?

How to get involved with the Working Group, learn more...

Regular Work Group Meeting:

USA WG Meeting Wednesday 9am PT, every 4 weeks, next on June 19

APAC WG meeting Wednesday 5 UTC every 4 weeks, next on June 5

- Meeting notes and agenda

Link to join the group

- https://groups.google.com/forum/#!forum/kubernetes-wg-iot-edge

Link to join Slack

- https://kubernetes.slack.com/messages/wg-iot-edge

White Paper

- http://bit.ly/iot-edge-whitepaper