# Building Scalable Messaging Systems with Qpid

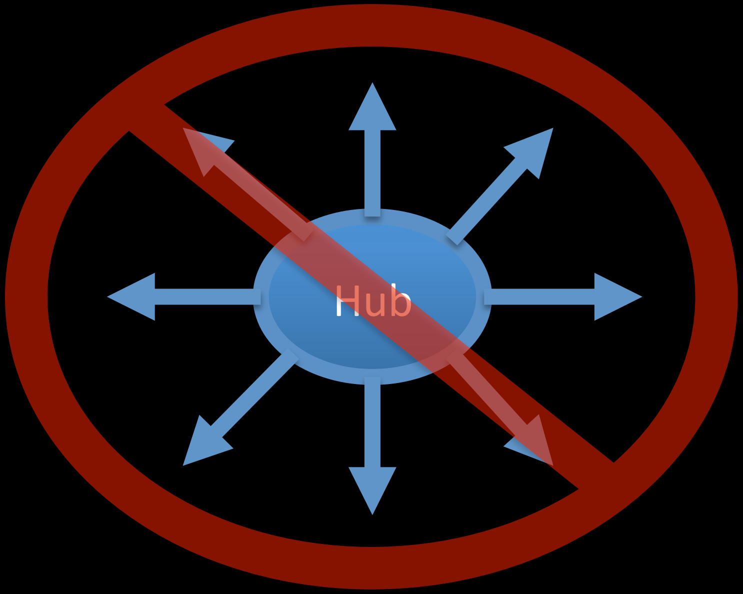
Lessons Learned from PayPal

## Background @ PayPal

- Handles 60% of all web transactions
- One of the largest Oracle instances
- Mix of proprietary systems
- Mix of 1000’s of stateless processes
- Traditional JEE applications
- Payments are generally asynchronous
- Payments are generally messages

# Basic Requirements

- Scaling
  - Highly Scalable
  - Partitionable
  - Cloud Friendly
- Failure
  - Continuously Available
  - No Avoid Single Point of Failure
  - Nothing Shared
- Latency
  - Near Real Time



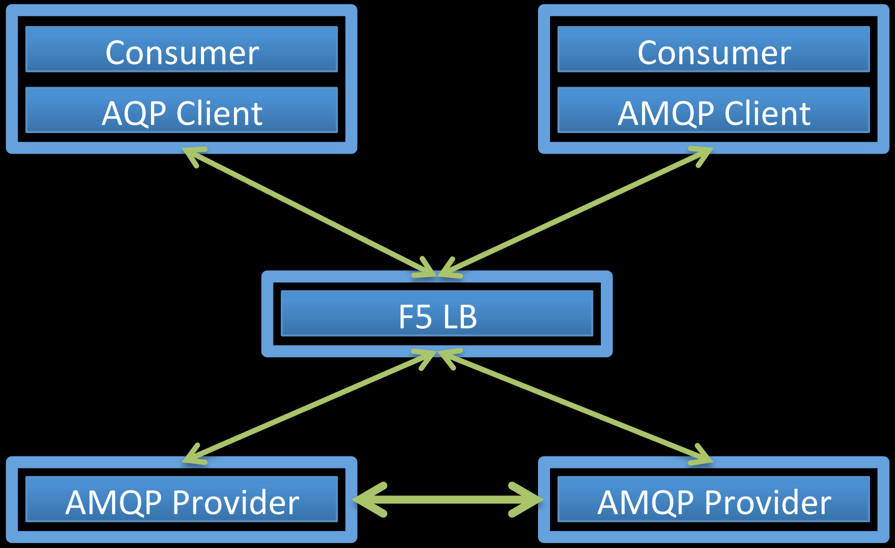
## Why

- Desired an open messaging protocol
- Cross platform interoperability (C++, Java, Python)
- Required very low latency
- Eventual interoperability with ActiveMQ
- Ability to influence the community

# Where We Started

- Simple Network of Brokers
- Load Balanced via L5 Switch
- Round Robin, Least/Min Rule
- Replicated

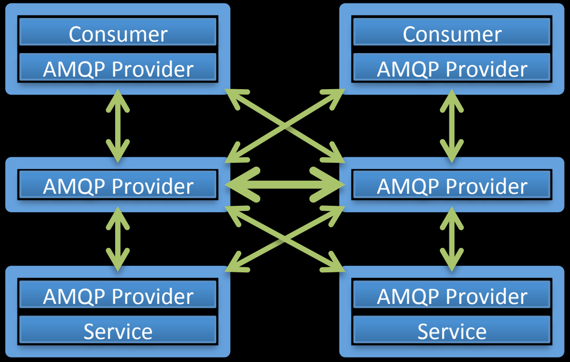
Point to Point



## What We Found

- Scale
  - 20 billion 2K Messages Per Day
  - Variation Message Size > Latency
- Connections
  - Short lived processes strain the broker
  - @5000-6500 broker begins to flail
- Routing Concerns
  - Distributing connections
  - Prohibited InVM messaging

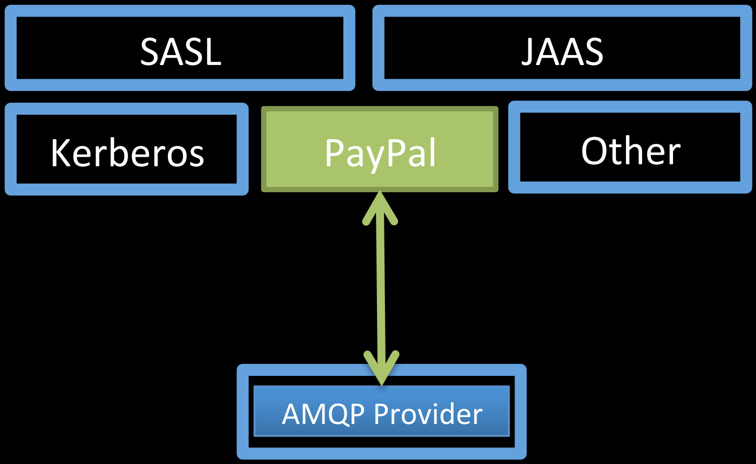
# Next Evolution

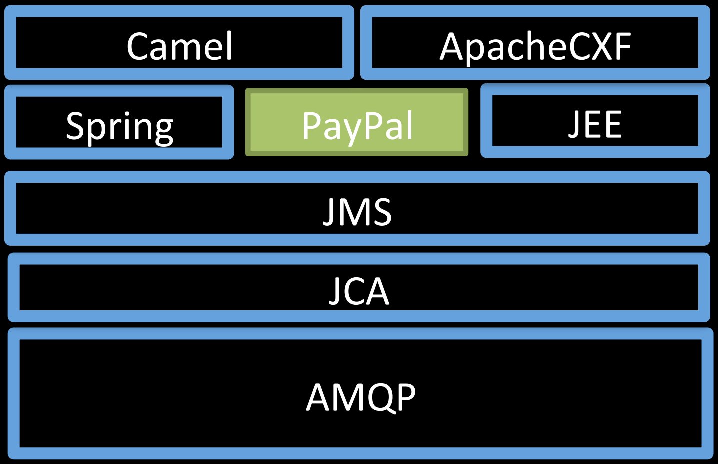


- Create Distinct Layers of Brokers
  - Front Tier
  - Mid Tier
  - Core Tier
- Partition Each Layer By Function or Actor
  - User Type (Consumers, Merchants, API)
  - Business Function (Risk, Payments, Account Servicing)
  - System Function (Events, Services, Logging)
  - Cloud Friendly
- Isolate Partitions within the Broker

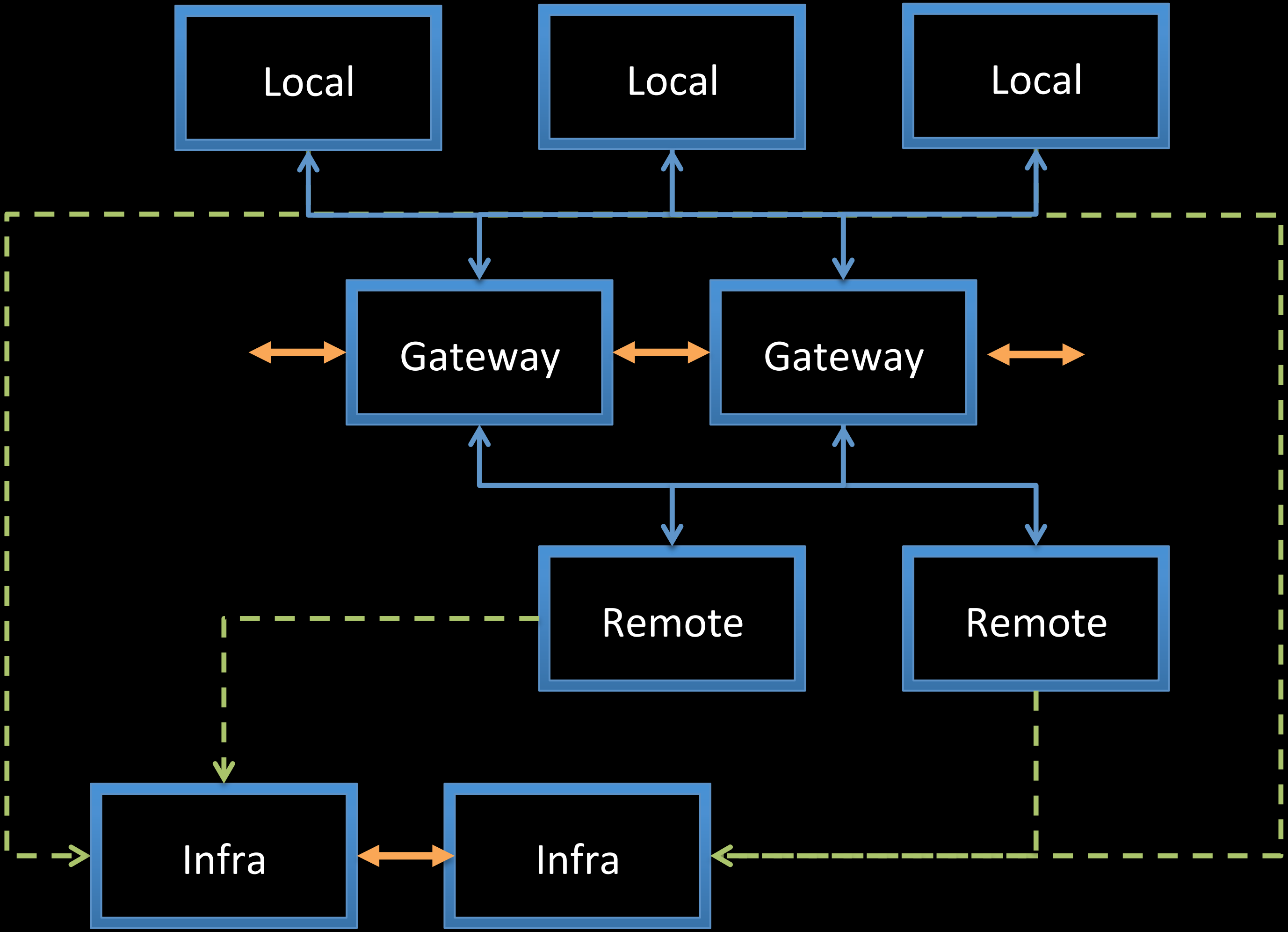
# Interfaces

- Federation Semantics Part of the Address
- Externalize Addressing
- Use “pure” AMQP or JMS wherever possible





# Network

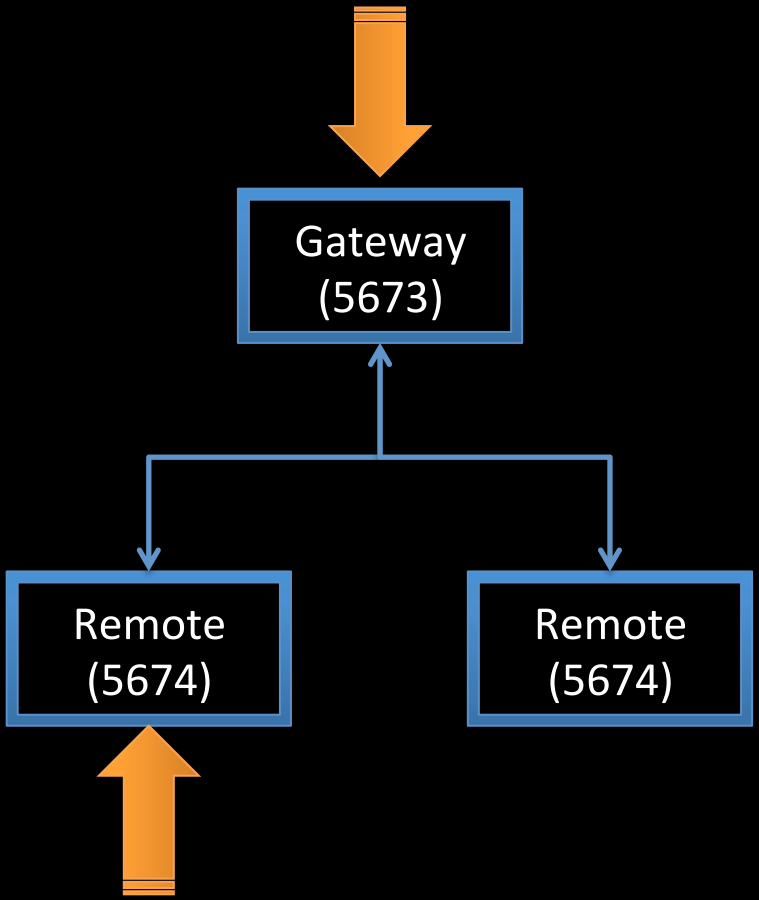

Local
Local
Local
Gateway
Gateway
Remote
Remote
Infra
Infra

## Federation

- Distinct Request & Response Exchanges
- Local & Remote Request Destinations
  - `qpid-config -a localhost:5673 add exchange direct service_gateway.direct`
  - `qpid-config -a localhost:5674 add exchange direct service_impl.direct`
  - `qpid-config -a localhost:5673 add queue user.lifecycle.request`
  - `qpid-config -a localhost:5674 add queue user.lifecycle.request.impl`
- Requests use queue routes
  - `qpid-route queue add localhost:5674 localhost:5673 service_impl.direct user.lifecycle.request`
- Responses use dynamic routes
  - `qpid-route -v dynamic add localhost:5674 localhost:5673 service_res.direct –durable`
- Responses use unique binding addresses for routing

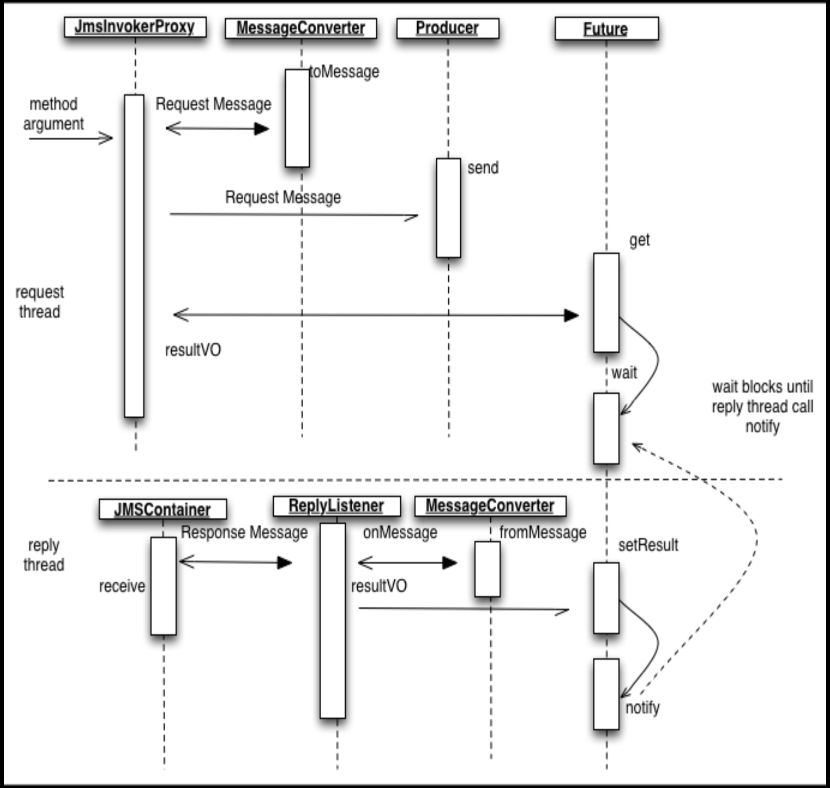
Example

# FEDERATION

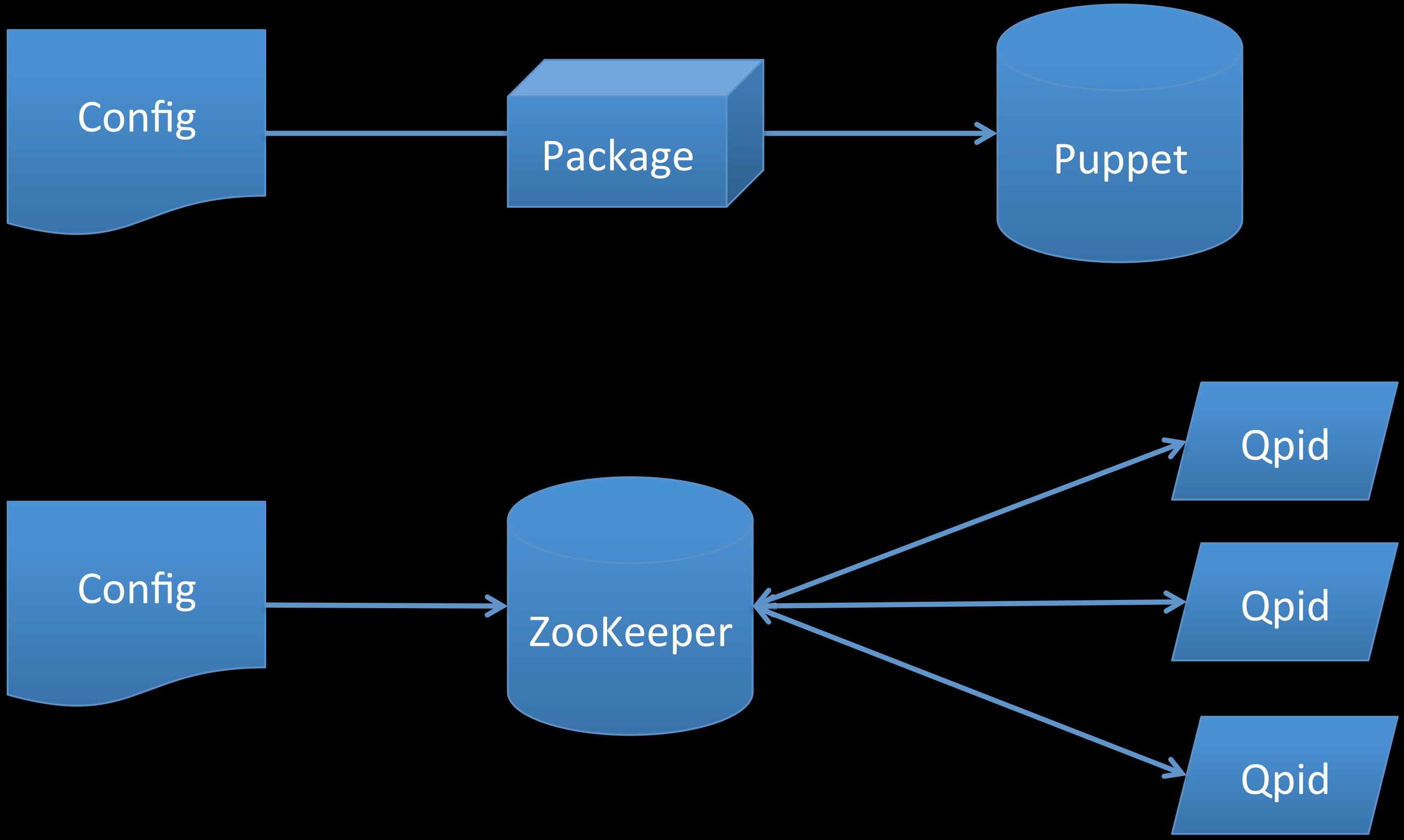


# Client/Publisher

- Message size is important
- Avoid default reply to implementations
- Pull configuration versus pre-packaging



# Managing the Config



# Monitoring

- Qpid Management Framework
  - Each Object In the Broker Publishes Event
  - Events are Messages (Topic) that are Routable
- Describe Interest In Events
- Listeners that subscribe & dispatch
  - SNMP
  - Nagios
  - Internal Logging

## What Can Be

- Agent
- Binding
- Bridge
- Broker Connection
- Exchange
- Link
- Queue
- Subscription
- System
- VHost

- Events
  - New Objects
  - Updates
  - Failures (Links)
- Configuration
  - Properties
  - States
- Statistics
  - TXN
  - Messages
  - Latency

Example

## QUEUE MONITORING

## Performance

Raw QPID average roundtrip message times in milliseconds over 100K messages.

| Configuration | 1 | 100 | 1024 | 2048 | 4096 | 8192 | 16384 | 32768 | 65356 | 130712 |
|---|---|---|---|---|---|---|---|---|---|---|
| Single Node TCP over 1gbE | 0.6 | 0.6 | 0.66 | 0.64 | 0.65 | 0.66 | 0.73 | 0.9 | 1.07 | 1.55 |
| Multi Node TCP over 1gbE | 0.9 | 0.9 | 0.98 | 1.08 | 1.13 | 1.25 | 1.49 | 2.14 | 3.35 | 5.72 |
| Multi Node RDMA over IB | 0.42 | 0.43 | 0.43 | 0.44 | 0.44 | 0.45 | 0.51 | 0.55 | 0.63 | 0.74 |

Use case comparison between MBSI architectures by Infrastructure. Times are average per message over 1K messages

| Use Case | Binary (1024) | XML (1024) | Binary (16384) | XML (16384) | Binary (65536) | XML (65536) | Binary (131072) | XML (131072) |
|---|---|---|---|---|---|---|---|---|
| QPID/SSL (Sync) | 2.66 | 3.33 | 6.11 | 7.17 | 11.73 | 16.07 | 19.71 | 27.86 |
| QPID/SSL (Fire-and-forget/Non-Ack'd) | 0.51 | 0.56 | 0.54 | 0.74 | 0.8 | 1.44 | 2.05 | 2.59 |
| QPID/SSL (Fire-and-forget/Ack'd) | 1.04 | 1.08 | 3.52 | 2.04 | 3.69 | 4.08 | 5.84 | |

Throughput:

- Single Node 141K Messages Per Second @ 1K Message Size & Single Producer
- I/O Bound on the Producer

## How We Can Use It

- PayPal Cloud
  - Openstack integration via AMQP
  - Dynamic Scaling via QMF events

- Possible Applications
  - Mobile via JavaScript Proton
  - Payment Devices

## Opportunities

- AMQP Links between Qpid & ActiveMQ
  - Heterogeneous messaging fabric

- Embedded Messaging Engines
  - In car devices
  - Point of Sale
  - Phones

- Replace proprietary service frameworks with Proton

- Replace Qpid Libraries with Proton