VOLTTRON-RabbitMQ Message Bus Overview

Shwetha Niddodi
Chandrika Sivaramakrishnan
Kyle Monson
Craig Allwardt
Jereme Haack

Outline

- Background
- Message Bus Plugin Framework
- RabbitMQ Overview
- RabbitMQ VOLTTRON
- Conclusion

Background

- VOLTTRON's ZeroMQ based message bus has been key for meeting the security and interoperability goals of the platform
- At the same time, RabbitMQ has become more mature as it has seen major investment by commercial companies.
  - Rabbit Technologies, now part of Pivotal Technologies (VMWARE spin-out). $105 million investment by GE in 2013.
  - Used by: Instagram, Indeed.com, Google Cloud Platform, Tesla ⋯
- Goals of the Refactor:
  - Maintain essential features of current message bus and minimize transition cost
  - Leverage an existing and growing community dedicated to the further development of RabbitMQ
  - Move services provided currently by VOLTTRON agents to services natively provided by RabbitMQ
  - Decrease VOLTTRON development time spent on supporting message bus which is now a commodity technology.
  - Address concerns from community about ZeroMQ
- View this effort as essential to the long-term future of the platform
  - Working with heavy users in the community to get feedback
  - Reduce long term costs of platform by moving message bus development out of core
  - Maintain support for ZMQ short term (3 – 5 years) as funding allows

Why RabbitMQ?

- Supports different messaging patterns, routing topologies
- Easy to use, has most of the features already built-in
- Well developed security feature
- Large scale deployment
- Flexibility in deployment
- Recommended by VOLTTRON community
- Other competitive message buses that were considered.
  - Kafka – Uses “dumb broker, smart consumer approach” which is opposite of our requirement.
  - MQTT – Limitation in security feature.
    - The payload can be secured but not the header

Message Bus Plugin Framework

- Decouple VOLTTRON from message bus implementation
  - Similar concept to Historian and Driver frameworks
  - Provide hooks to allow new solutions to be leveraged
- "Proxy Agent" that handles VOLTTRON specific code needed to interact with different types of message buses
- Must prevent fracturing of ecosystem
  - Ensure VOLTTRONs using different messaging systems can still communicate with each other

Upgrading to RabbitMQ Based Platform

- Minimize changes needed to agent code
  - No effect unless agent has ZMQ specific code (ForwardHistorian)
- No changes required to driver configuration
  - Data collection unaffected
- Backward compatibility
  - Allow ZMQ and RabbitMQ instances to communicate
- Will require new code to connect RabbitMQ based systems together

Message Bus Plugin Framework

- Consists of five components
  - New connection class per message bus
  - Extensions to router module functionality
  - Extensions to core agent functionality
  - Adding a proxy agent for each message bus
  - Authentication related changes

Platform Level Changes

- On startup, checks the type of message bus used.
  - Creates appropriate router module
- ZMQ Router – Handles actual routing since ZMQ is broker less protocol. Functionality is unchanged.
- RMQ Router – Light weight.
    - Maintains connection to RMQ message bus
    - Actual routing is handled by RMQ broker
    - Handles router specific subsystem messages
    - Handles unrouteable messages

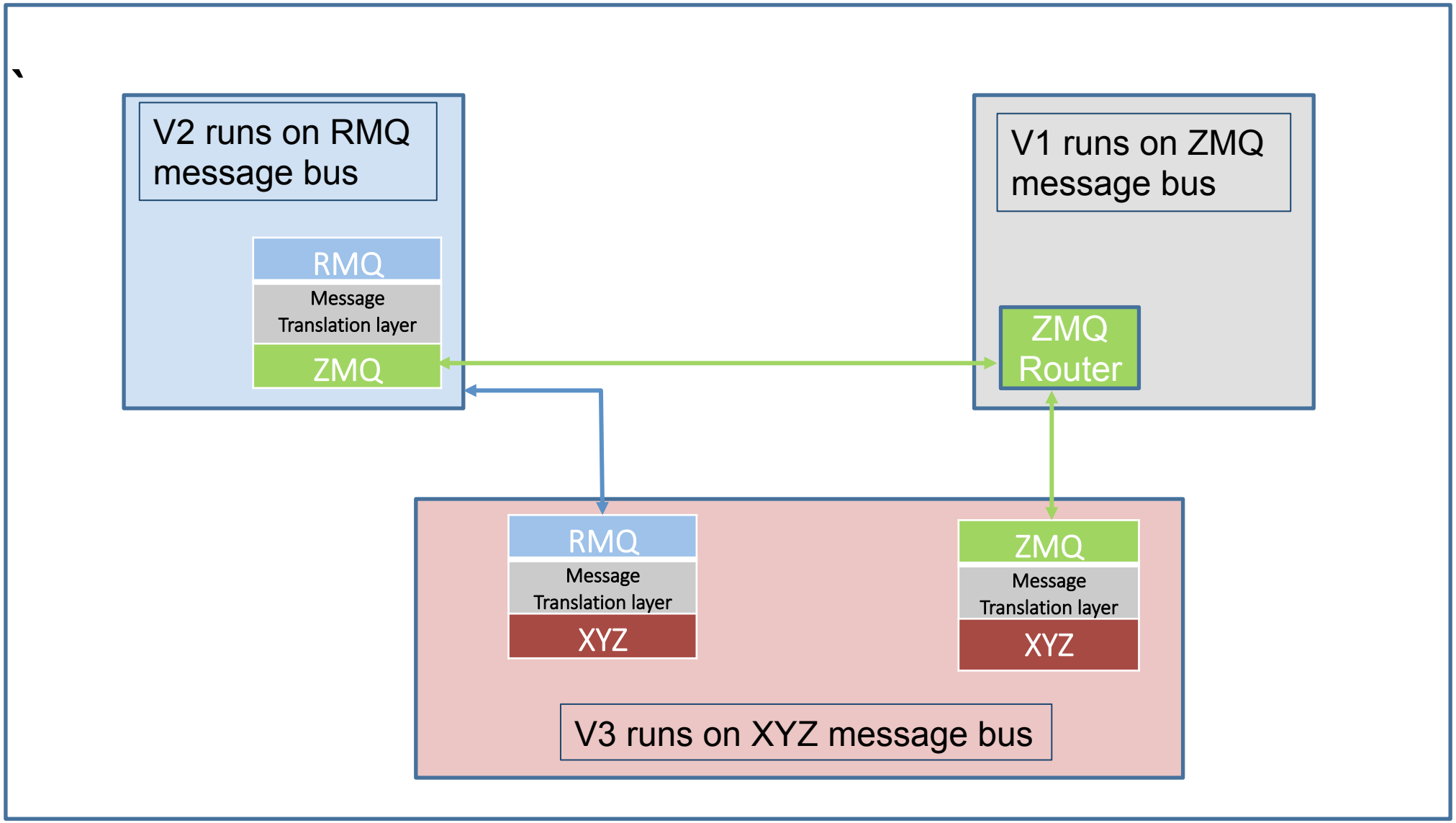
Agent Core Changes

- Application agent code remain unchanged
- Agent Core changes
  - On startup, checks the type of message bus used.
  - Maintains connection to appropriate message bus.
  - All subsystem messages are encapsulated inside a message bus agnostic VIP message object.

Compatibility between VOLTTRON instances running different message buses

- Proxy Agent – Acts like a bridge between local message bus and remote message bus
  - Different proxy agent for each type of remote message bus connection.
  - Maintains connections to internal and external message bus
  - Route messages from internal to external
  - Route messages from external to internal

Proxy Router Agent acting as bridge



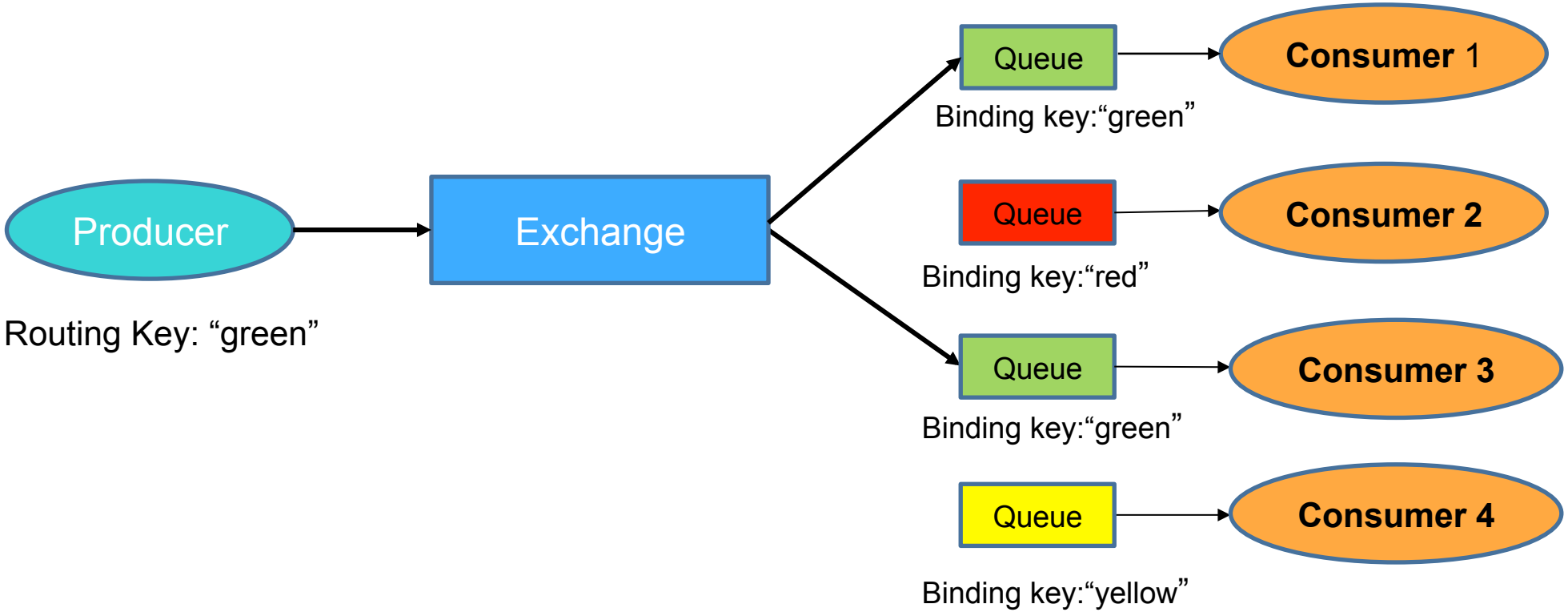
RabbitMQ Overview (contd.)

- RabbitMQ uses AMQP (Advanced Message Queuing Protocol)
- Exchanges – Responsible for routing of messages to Queues.
  - They look at the routing key in the message when deciding how to route messages to queues.

Exchange Type	Routing Type
Fanout	One to all
Direct	Based on exact match of routing key
Topic	Based on pattern match of routing key Example: “cars.Subaru.Outback”, “cars.Subaru.*”, “cars.#”
Headers	Based on attributes in message header. Routing key is ignored

RabbitMQ Overview

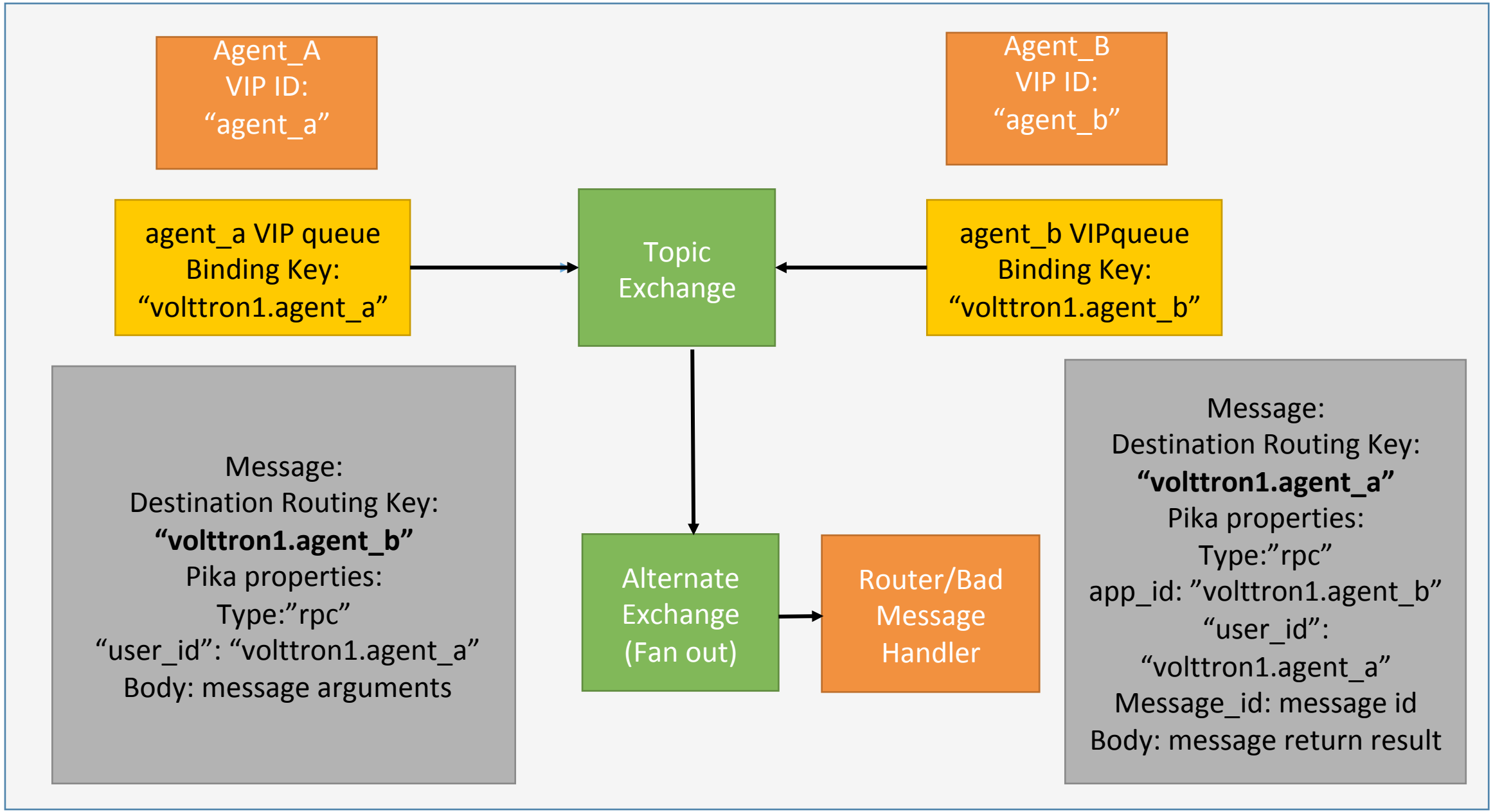
- Queues - Buffer that stores the messages until consumed by consumer.
- Bindings – Queues bind to the exchange with binding keys or routing pattern.
  - Messages are routed to one or many queues based on a matching between a message routing key and binding key.



RabbitMQ VOLTTRON

- Uses Pika library – Python library for RabbitMQ
- Each VOLTTRON instance creates a single topic exchange – “volttron”
- Each Agent Core
  - Connects to RabbitMQ broker
  - Creates VIP queue and binds to exchange with binding key “<instance name>.<agent identity>”
  - Send/Receive VIP messages using Pika methods
  - Routes incoming messages to appropriate subsystem
- Platform Router
  - Connects to RabbitMQ broker
  - Creates VIP queue and binds to exchange with binding key “<instance name>.<router>”
  - Handles messages intended to the router
  - Handles unrouteable messages

RPC Subsystem

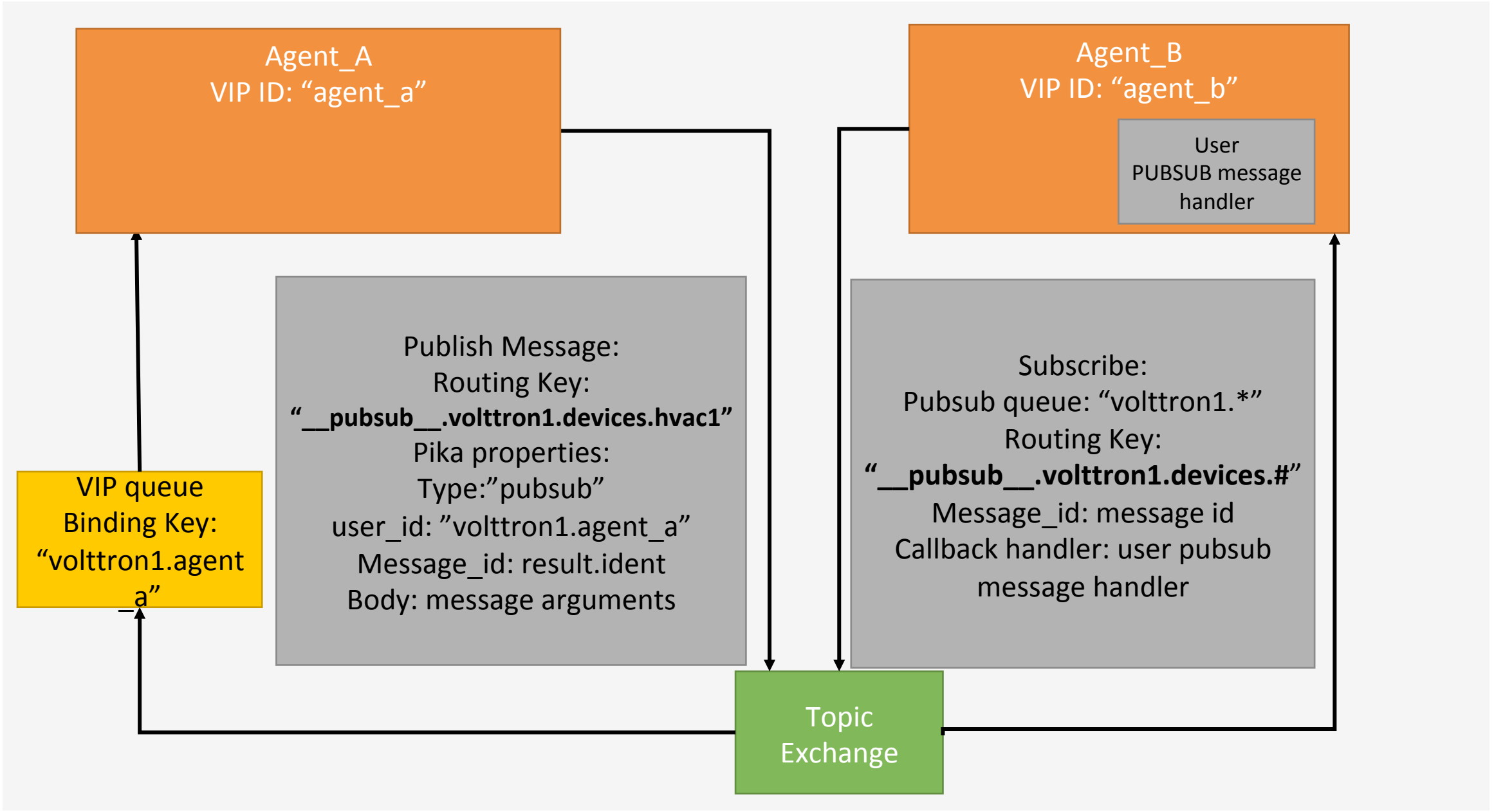


RabbitMQ PUBSUB

- Agent functionality remains unchanged.
  - Agents continue to use same PubSub interfaces
- User can continue to use topics delimited by “/”.
- RabbitMQ PUBSUB converts it to “.” internally.
- Agent topics are internally prefixed “__pubsub__.<platform_id>” to differentiate from main Agent binding.
- If agent wants to subscribe to topic from remote instances, it uses

  agent.vip.subscribe(“pubsub”, “devices.hvac1”, all_platforms=True”)
  - It is internally set to “__pubsub__.*.<remainder of topic>”

PubSub Subsystem

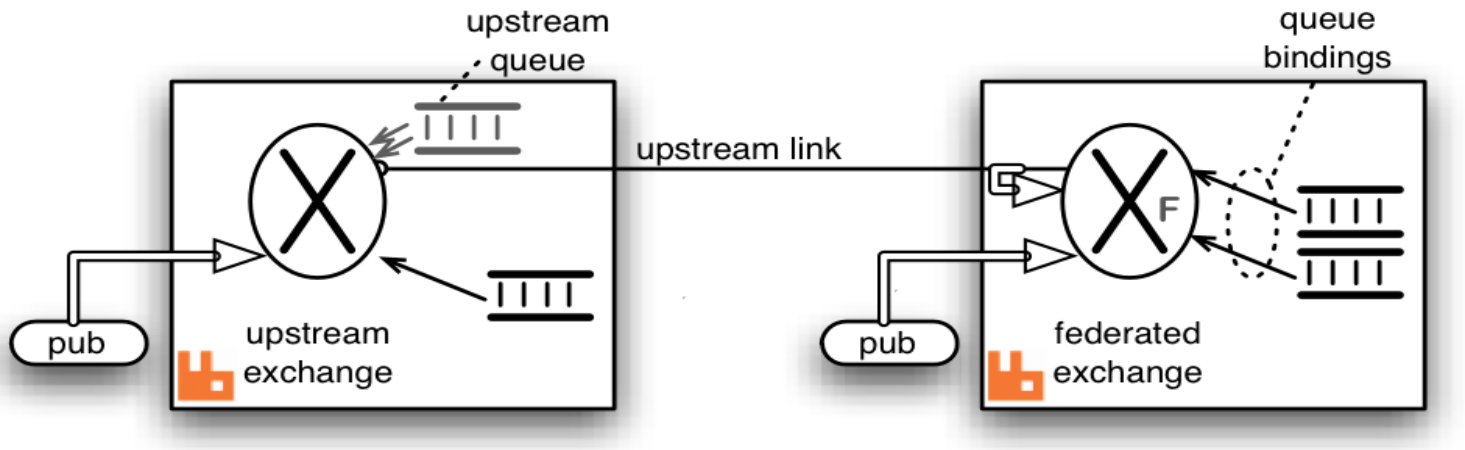


Multi-Platform Communication

- With ZMQ based VOLTTRON
  - Write an agent that would connect to remote instance directly.
  - Use special agents such as forwarder/data puller agents to forward/receive messages to/from remote instances.
    - Shovel Plugin can be used in RabbitMQ VOLTTRON
  - Configure VIP address of all remote instances in $VOLTTRON_HOME/external_discovery.json. Let the router module in each instance manage the connection and message routing for us.
    - Federation plugin can be used in RabbitMQ VOLTTRON

Federation Plugin

- Used for connecting multiple brokers
  - Loose coupling of nodes
  - WAN friendly – Tolerates network intermittency
  - Specificity – Not everything needs to be federated.
  - Scalability - Does not require $O(n^2)$ connections between *n* brokers
- Allows you to make exchanges and queues ***federated***.
- A federated exchange can route messages published upstream to a local queue.
- A federated queue lets a local consumer receive messages from an upstream queue

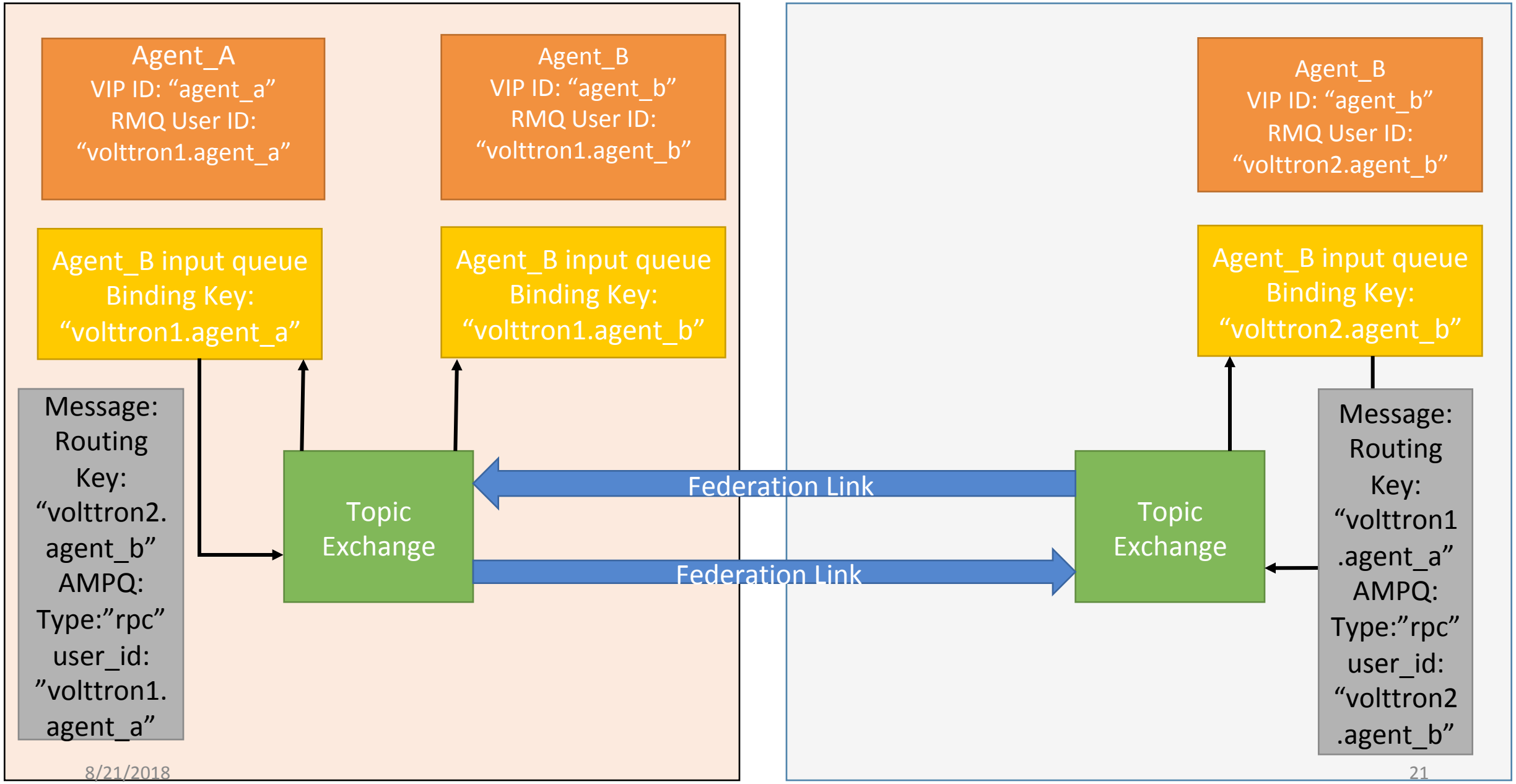


- Reference: https://www.rabbitmq.com/federation.html

Shovel Plugin

- Used to move messages from one broker to another broker
  - Loose coupling
  - WAN friendly - Tolerates network intermittency
  - Dynamic shovels do not need restart of broker
- Acts as well-written client application that
  - Connects to source and destination broker
  - Consumes messages from the queue
  - Re-publishes messages to the destination maintaining the same message format (and routing key if needed)
  - Messages forwarded based on routing key or pattern match
- Useful if one of the instance is behind NAT
- Shovel setup requires foreknowledge of topic subscriptions.
  - Limit unneeded traffic across the wire
  - Does not adapt to subscriptions automatically like a Federation link.
- Reference: https://www.rabbitmq.com/shovel.html

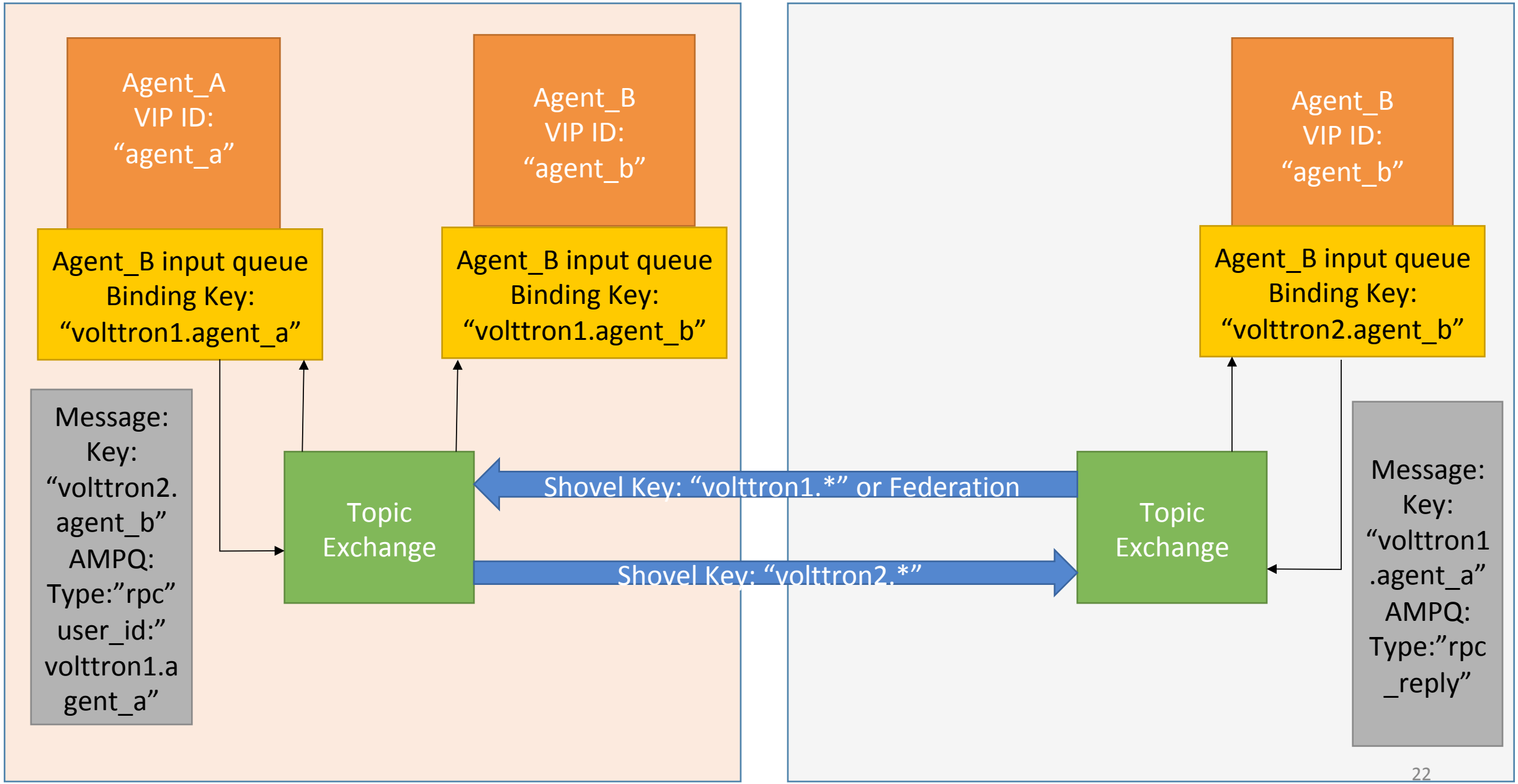
RPC Multiplatform



VOLTTRON – 1 (behind NAT)

RPC over NAT Multiplatform

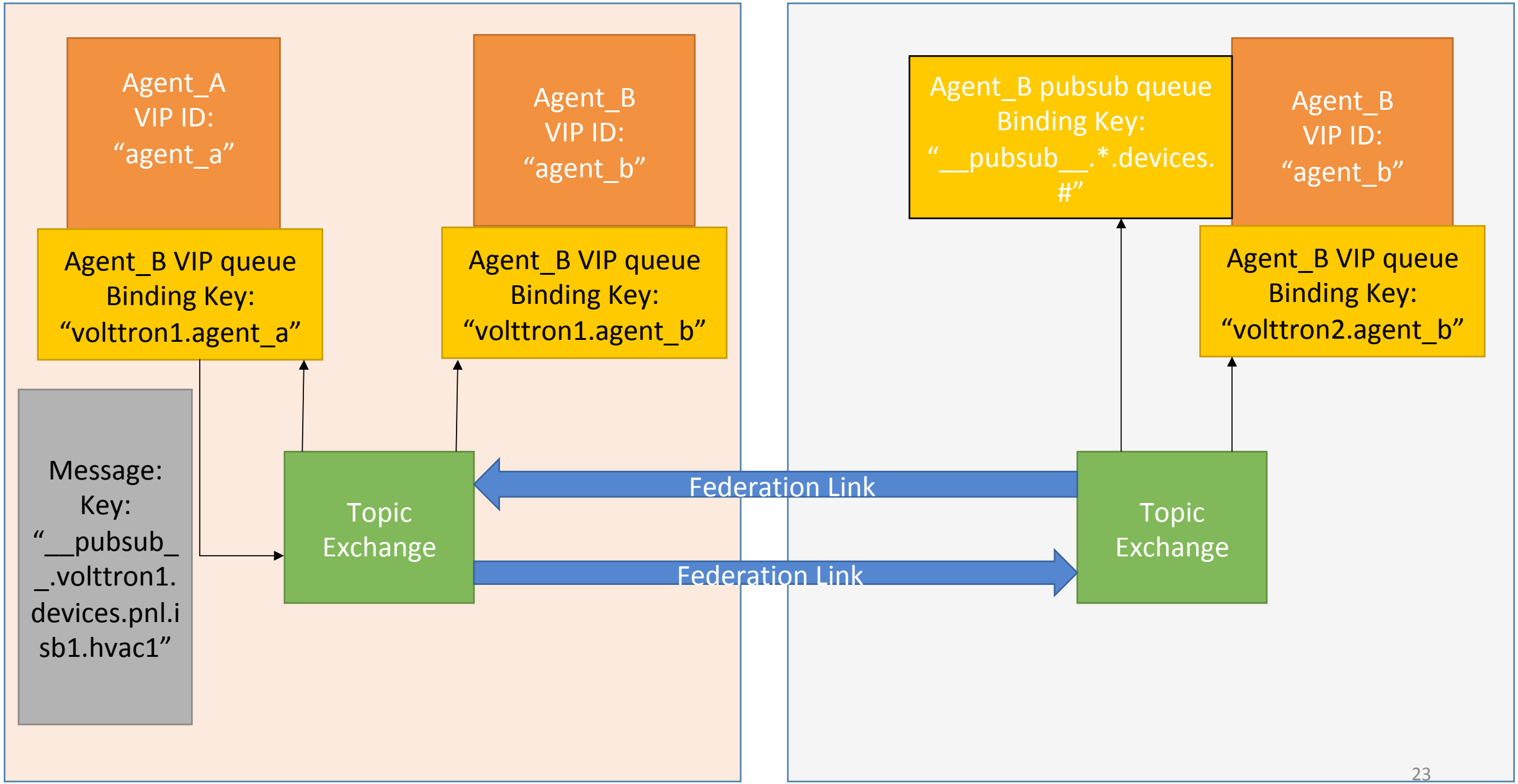
VOLTTRON - 2



VOLTTRON - 1

PUBSUB Multiplatform

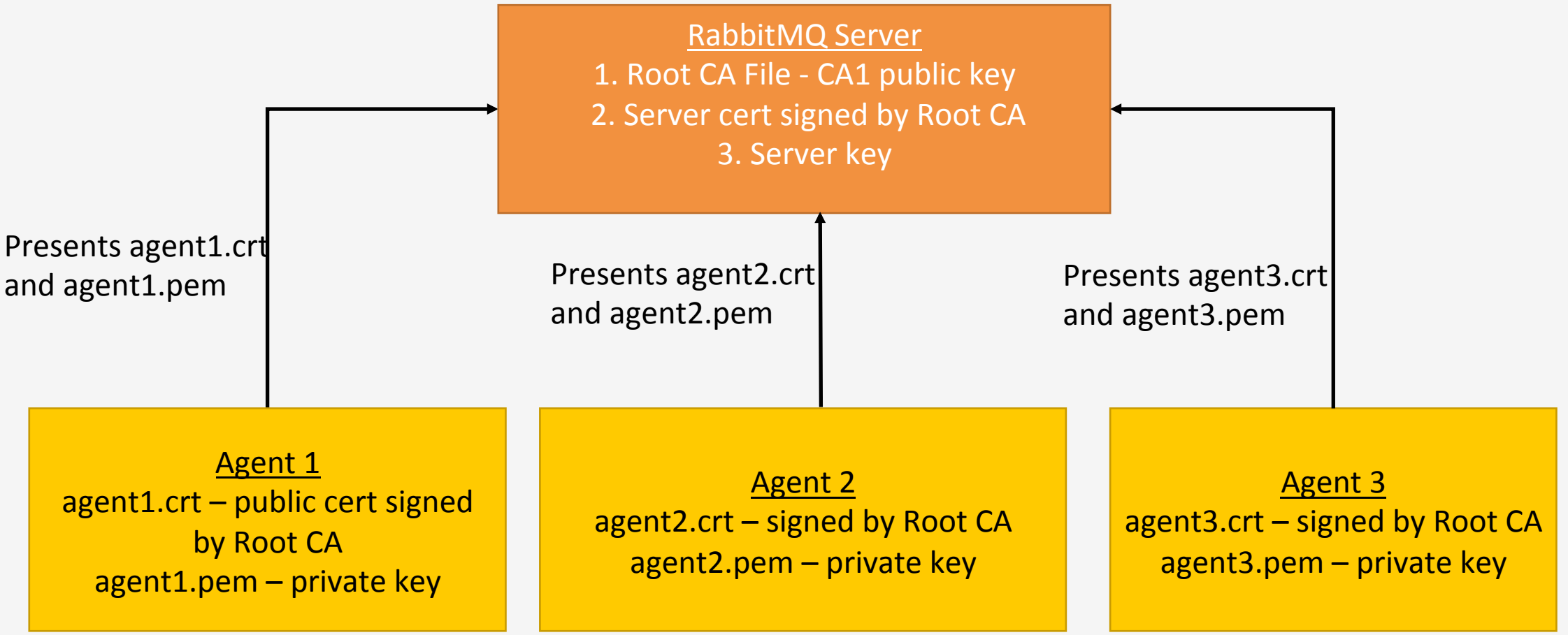
VOLTTRON - 2



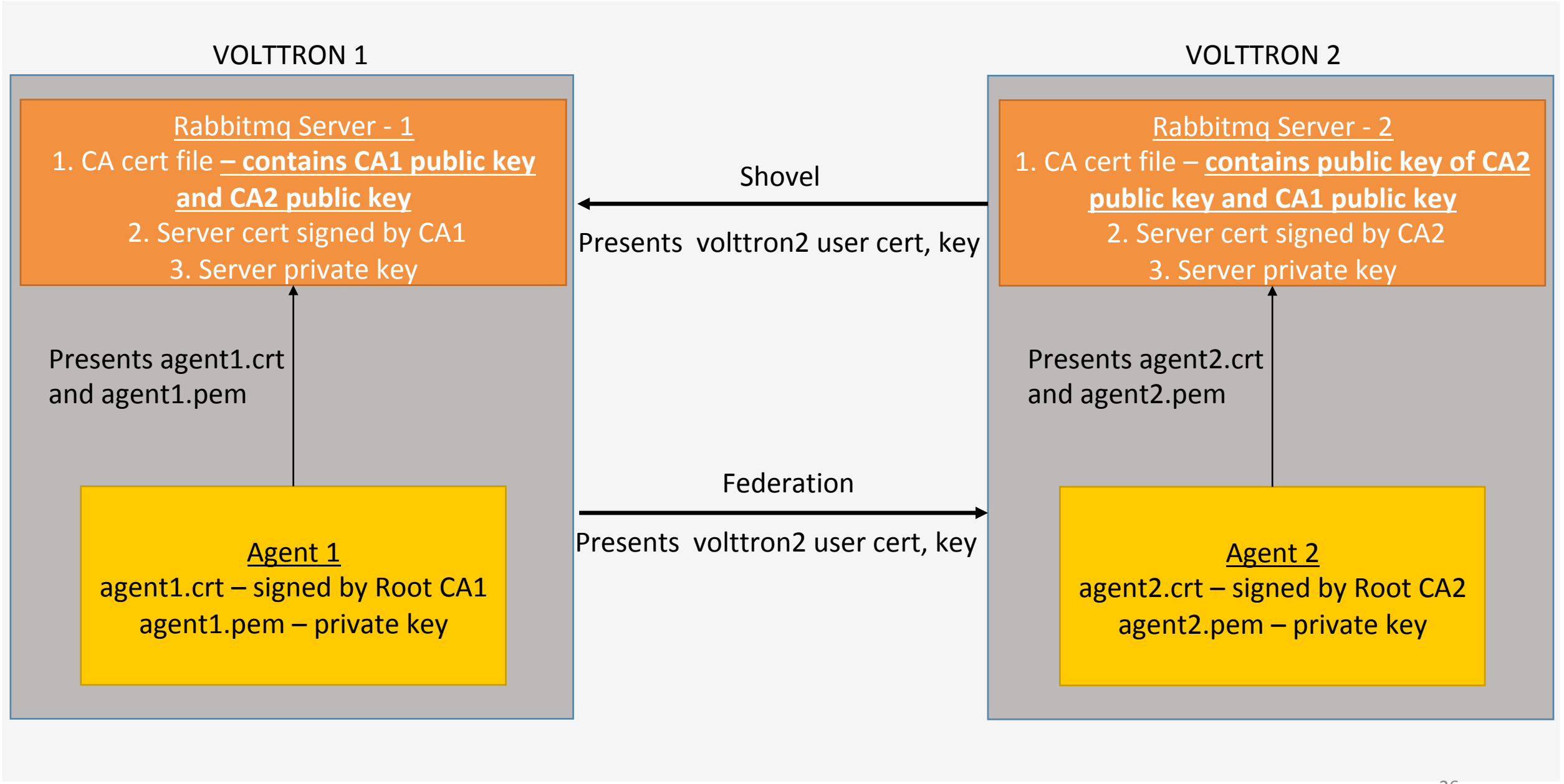
Authentication with RabbitMQ message bus

- RabbitMQ supports multiple authentication mechanisms
  - For VOLTTRON we use SSL peer verification using with x509 certificates
- SSL certificates of interest
  - Root CA
  - Server certificate signed by root
  - Client certificate signed by root

RabbitMQ Server SSL certificates



Multi-Platform Connection with SSL certificates



Authorization in RabbitMQ message bus

- For protected topics, topic permissions are set using RabbitMQ's management interface based on agent's user id
  - Allow publish rights – set read + write permission on the topic for the agent
  - Allow subscribe rights – Restrict read permission on the topic for the agent
- Capabilities on RPC methods – Remains unchanged
  - Specify required capabilities inside agent's code

```
@RPC.allow('SET_TEMP')
@RPC.export
def set_temperature():
    ....
```

  - Authorization entries (specified via volttron-ctl auth commands)

VOLTTRON RabbitMQ Management Utility

- Extend “volttron-ctl” utility to include RabbitMQ management commands.
- Uses RabbitMQ management http plugin
- Some of the functions
  - Create/Delete vhosts for each platform
  - Create/Delete unique user, password for each agent
  - Set permissions on the user
  - Create/Delete exchanges and queues
  - Create/Delete federation and shovel setup for multi-platform deployment.
  - Set topic permissions for protected topics
  - List the status of
    - Open Connections
    - Exchanges
    - Queues

Scalability Tests

- Single instance can support close to 1000 agents
- Tested with 100 instances forwarding messages to single root node connected together

- Latest Code in experiment branch:

https://github.com/VOLTTRON/volttron/tree/rabbitmq-volttron

Next Steps

- Integrate Volttron Central agent to use RabbitMQ message bus.
- Further improvements to installation steps and SSL based authentication.
- Scalability tests with large scale deployments.
- Complete testing of RabbitMQ VOLTTRON.

Conclusion

- Questions?
- Suggestions?
- Let us know if you would like to be in the message bus refactor working group: volttron@pnnl.gov

Backup

Proxy Router Agent acting as bridge