Network Working Group
Internet-Draft
Intended status: Informational
Expires: 27 April 2023

S. Nandakumar
C. Jennings
Cisco

24 October 2022

Cached and Async meSsage Transport (CAST) draft-nandakumar-mimi-transport-00

Abstract

This specification defines message transport, called CAST, based on publish/subscribe semantics for interoperable inter-server messaging that is rooted in modern, cloud friendly and scalable architectural underpinnings.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at https://datatracker.ietf.org/drafts/current/.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on 27 April 2023.

Table of Contents

1. Motivation

When designing federation systems for message exchange, one common way to implement a messaging transport would follow architectural patterns like "Send and Forget Architecture".

In this type of architecture, the requirement of the message transport is to attempt to send the message from the source domain to the target domain with no additional protections against losses in transit or causes that render messages not being delivered.

As depicted in the diagram below, the happy path flow results in the message being from alice@abc.com to be delivered to bob@xyz.com.

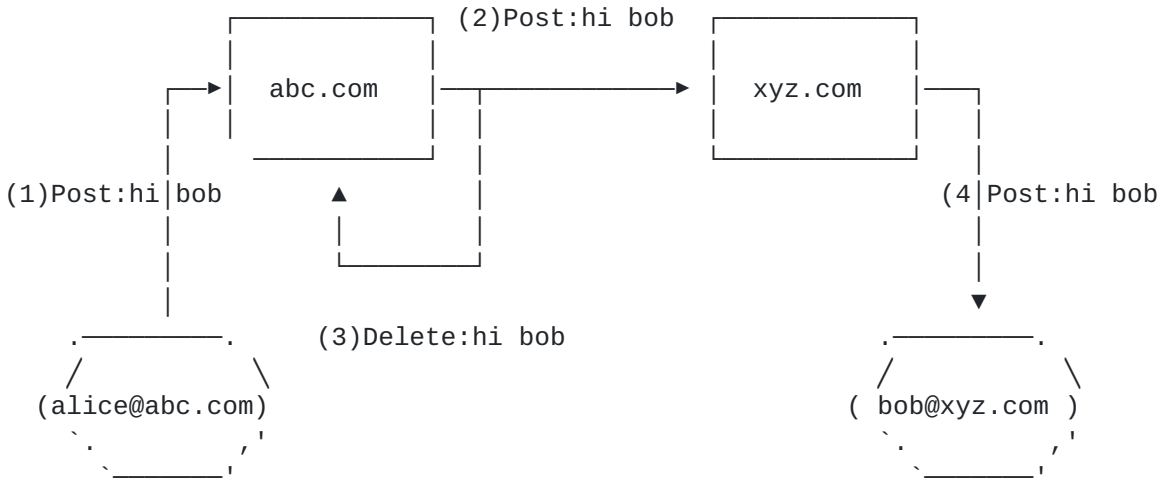
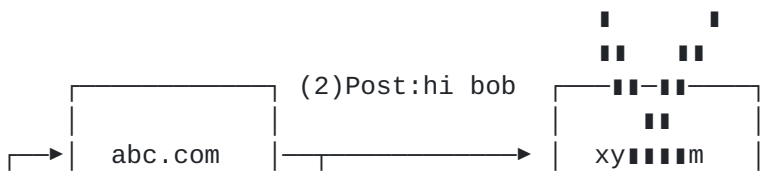


Figure 1: Send and Forget Pattern

One of the shortcomings of this architecture is the federation message transport doesn't provide protections or options to recover from reasons that might result in messages being undelivered. As shown in the below depiction, when one/more resources at the target domain becomes unavailable (due to server(s) failure, server down due to maintenance), the message from alice@abc.com misses its delivery to bob@xyz.com. Also to note, mechanisms to retry may be unsuccessful under these circumstances. In such cases, the onerous task is on the message sender to realize and retry/resend the messages. This is either done out of band with human intervention typically.



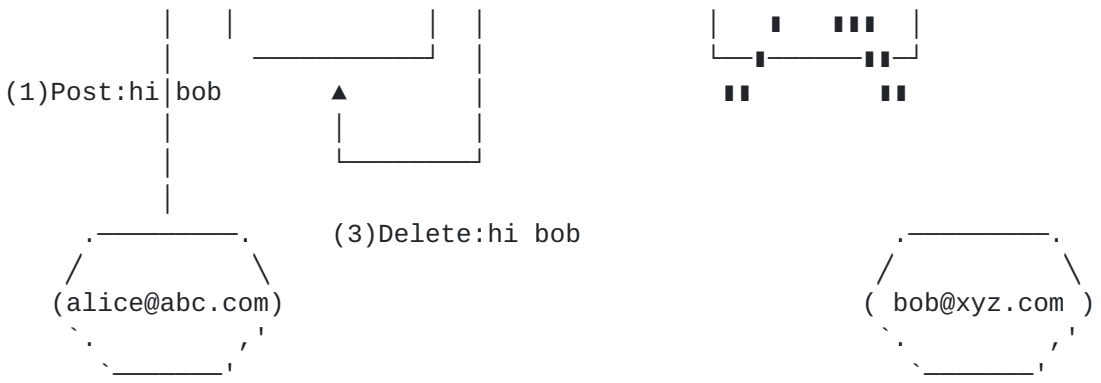


Figure 2: Send and Forget Pattern, Failure Case

SMTP and XMPP based federation transports are typical examples where "Send and Forget" architecture is employed.

This specification proposes a message interop transport suited for modern messaging and cloud friendly architectures which intends to address shortcoming with existing transports.

2. Modern Messaging Federation Transport

Any considerations for building interop messaging protocol for modern messaging workloads needs to meet certain functional requirements as listed below:

* Leverage modern cloud native architectures

* Scale to large number of messages and consumers

* Be less onerous on the message senders to ensure the delivery

* Easy to build high reliability cloud design

* Easy to build horizontal scalability cloud design

* Better aligned with internal architecture of some existing solutions

* Easy to build gateways to existing APIs

CAST is publish/subscribe based interoperable messaging transport for inter-server message delivery. Such a transport can be used for delivering messages between servers within the same domain of operation or for cross domain message delivery.

An example set of CAST messages exchanged for delivering messages for the flow depicted are given below

(1)Add Subscription
messages/xyz.com/* -> xyz.com

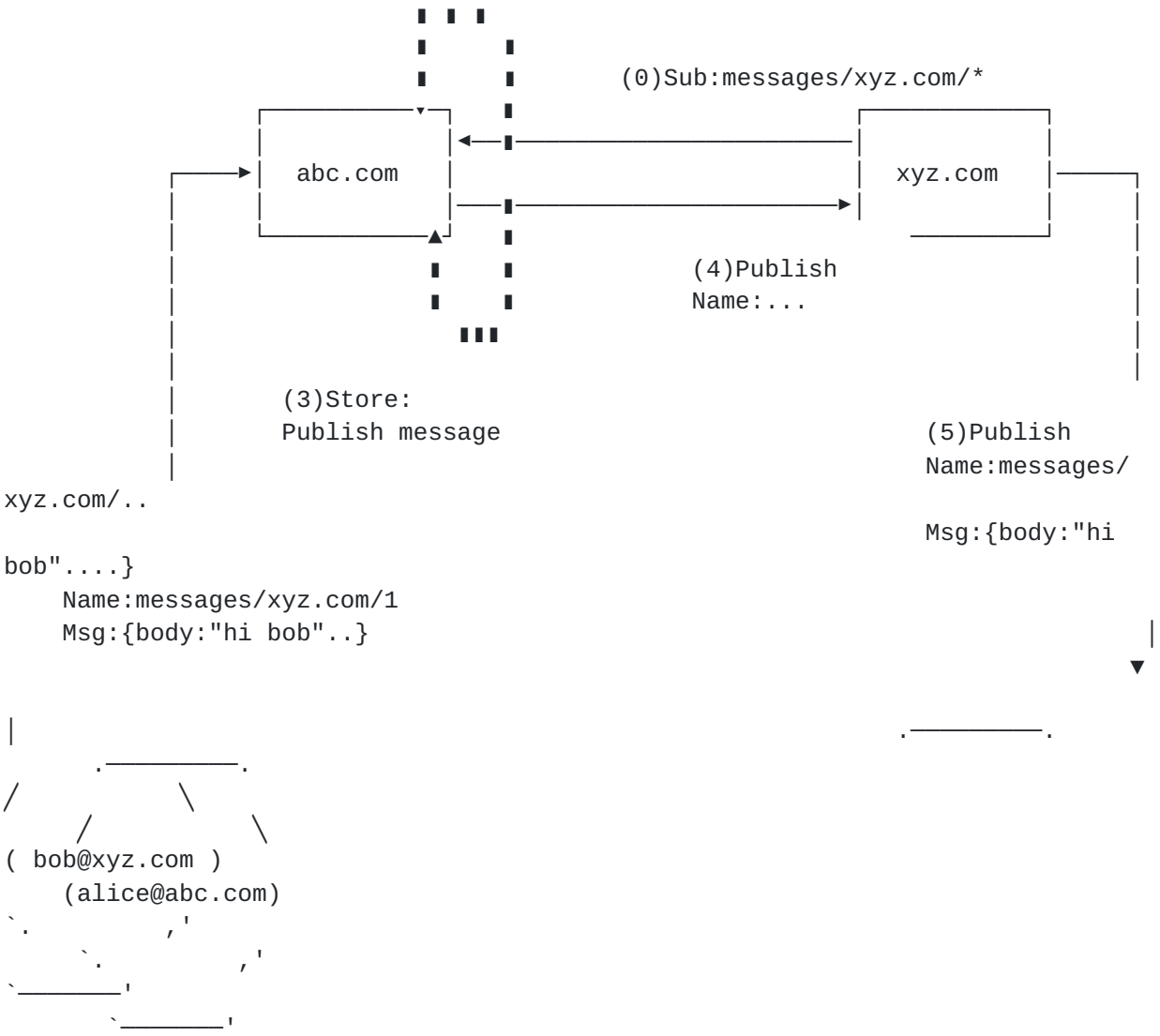


Figure 3: CAST Message Flow

In the example, the target domain, xyz.com, is interested in having messages from the domain abc.com to be delivered. Such a setup may be possible due to business relationships between the domains or any entity within the target domain expresses such an interest.

1. The CAST endpoint in the domain xyz.com sends "Subscribe" message to CAST endpoint in the "abc.com" indicating its interest to receive any message that matches the name "messages/xyz.com/*", i.e any message targetted to the domain "xyz.com"

2. On receiving the "Subscribe" message, the CAST entity at the domain "abc.com" creates an active subscription entry for the name "messages/xyz.com/*" against the domain "xyz.com". Thus created subscriptions remain active until they expire or are canceled. Subscriptions can be renewed periodically to keep them active as well.

3. When Alice from abc.com publishes message to Bob at xyz.com, by sending "Publish" message, the CAST entity (within abc.com) performs the following steps on receiving the message: - Store the message (with name messages/xyz.com/1) for at least for 24 hours. - Look up for any active subscriptions that matches the name - Forward the message to the CAST endpoint that matches the name from the previous step - In this example, since there exists an active subscription for the pattern "messages/xyz.com/*", Alice's message will be delivered to the CAST entity at "xyz.com" based on the lookup result

4. On receiving the CAST Publish message, the CAST entity at the domain "xyz.com" will have sufficient information in the message to forward it to the right target within its domain, in this case, to Bob

Messages within CAST are cached for at least 24 hours by default, regardless of the status of message delivery. This allows, for example, "xyz.com" to ask for the message again if the transaction to store it is unsuccessful.

In scenarios where subscriber CAST entity is unavailable at the time of the message delivery, the CAST entity resyncs its state by reissuing a "Subscribe" message, when it's back in operation and thus retrieve any cached messages as well as stream new messages that get published in the future.

2.1. Load Balancing and Reliability

To horizontally scale to a huge volume of messages, the servers in the abc.com domain can use any of the traditional techniques for load balancing requests across a cluster of servers including DNS, IP hashing, and others. Allowing the receiving domain, xyz.com, to load balance the incoming message across many servers can be slightly more complicated. The technique proposed here is to allow the subscription to filter a subset of message and make sure there is a sperate subscription for each incoming set. The filter criteria is is done by taking the hash of the UUID for the message and computing a modulo for it and checking it that matches a constant provide for each subscription. For example if the xyz.com wanted to split the data across 3 connection, it would specify a module of 3 and then each connection would specify 0,1,2 respectively as the constant.

The fact that the abc.com domain retains the message for some period of time and they can be requested a second time if needed allows the messages processing to be pipelined and batched with no acknowledgement which can greatly increase the speed of processing some sort of transaction where the sender can discard the message once the receivers acknowledge having it. The UUID in the messages allow the receiver to easily deal with receiving the same message

more than once.

3. Security Considerations

The assumption is all messages are end to end encrypted and neither domain can read the contents of any message between alice and bob.

The assumption is that a major mitigation of SPAM will be that alice sends a connection request to bob and bob accepts that before any messages with user generated content can be sent between alice and bob.

Within the CAST architecture, the interacting domains are trusted to deliver each other messages for their users and are bound by business agreements that further constrain the rules related to use of messages exchanged, dealing with spam and any other policies that govern the successful federation. This is meant for major services to connect to other major services and not designed to deal with the issue of a domain with no business relationship to another domain connected to it.

A given domain like abc.com does not reveal to xyz.com all the users it has but if alice in abc sends a message to bob in xyz, it does reveal to abc the existence of bob, and to xyz the existence of alice.

Appendix A. Acknowledgements

TODO

Authors' Addresses

Suhas Nandakumar
Cisco
Email: snandaku@cisco.com

Cullen Jennings
Cisco
Email: fluffy@iii.ca