KIP-559: Make the Kafka Protocol Friendlier with L7 Proxies

Status

Current state: *"Accepted"*

Discussion thread: *here*

JIRA: *here*

Please keep the discussion on the mailing list rather than commenting on the wiki (wiki discussions get unwieldy fast).

Motivation

There is a considerable interest in the Kafka community in the possibility of intercepting the Kafka protocol with L7 network proxies such as Envoy. The benefits of intercepting the protocol are huge. Amongst others, it includes the possibility to (1) observe the requests and the responses going to and coming from Kafka; (2) monitor then; (3) validate them; or (4) transform them (e.g. rename topics, encrypt payloads, filter records).

The number of L7 proxies supporting the Kafka protocol is extremely limited nowadays but there is an effort to bring support for the Kafka protocol in various proxies. Envoy has recently introduced support for the Kafka protocol [1][2], Linkerd has expressed its interest [3], and few people have implemented custom ones for specific use cases [4][5]. There is no doubt that this trend will continue and that intercepting the Kafka protocol will become easier and more common.

Overall, intercepting the Kafka protocol works well but we have noted few edge cases where few requests and/or responses can not be fully parsed without having kept informations about previous exchanges. This happens mainly when the requests or the responses contains "bytes" fields but without, explicitly or implicitly, indicating what those bytes represent. See Analysis of the Current API for details.

With this KIP, we would like to ensure that all the requests and the responses of the Kafka protocol are self-explanatory and thus could be processed independently. We feel like that it is the right time to tackle this as the adoption of those proxies is limited thus it gives us time to let the community adopt new releases of the protocol.

Analysis of the Current API

All the request and the response types have been analysis to identify the potential gaps in the API. Especially, we have looked at all the "bytes" fields in the API.

Type	Field (type)	Observation	Status
DelegationToken API			
CreateDelegationTokenResponse	▪ Hmac (bytes)	The `Hmac` field contains the HMAC of the delegation token. It is computed by the broker (HmacSHA512) when the delegation token is created, returned in the `CreateDelegationTokenResponse` and used in all the other requests/responses. There is not point in parsing it in a proxy.	**OK**
DescribeDelegationTokenResponse	▪ Hmac (bytes)		

ExpireDelegationTokenRequest	▪ Hmac (bytes)		
RenewDelegationTokenRequest	▪ Hmac (bytes)		
SaslAuthenticate API			
SaslAuthenticateRequest	▪ AuthBytes (bytes)	The `AuthBytes` field contains the SASL authentication bytes, as defined by the SASL protocol. It could be parsed independently by following the SASL protocol.	**OK**
SaslAuthenticateResponse	▪ AuthBytes (bytes)		
Produce/Fetch APIs			
ProduceRequest	▪ Records (bytes)	The `Records` field contains the so called Records Batches. They can be parsed using the Records Batched format as defined here.	OK
FetchResponse	▪ Records (bytes)		
Group API			
JoinGroupRequest	▪ Metadata (bytes)	The `Metadata` fields contain the protocol metadata of the member who would like to join the group. The request contains a `ProtocolType` field which indicate the protocol used to populate the `Metadata` fields. The group API has been designed to allow any protocol types. In practice, there are few well known types such as "consumer" or "connect" which are respectively used by Kafka Consumer and Kafka Connect. The request could be handled independently if the protocol type is known by the proxy. Note that there are cases where the broker also parses the metadata (e.g. KIP-496: Administrative API to delete consumer offsets).	**OK**
JoinGroupResponse	▪ Metadata (bytes)	The `Metadata` field contain the protocol metadata of a member of the group. Metadata of all the members are sent back to the group leader to let him compute the assignments for the group. The response does not contain any indication of the protocol type used. It was not put in the response because the members already have the information on their end. It means that the response could not be handled independently without prior knowledge captured from the JoinGroupRequest.	**GAP**
SyncGroupRequest	▪ Assignment (bytes)	The `Assignment` field contain the assignment for a member of the group. The group leader computes all the assignments foreach member and send them to the group coordinator. The request does not contain any indication of the protocol type used because both ends have already the information. It means that the request could not be handled independently without knowing the prior information exchanged between the members and the coordinator.	**GAP**
SyncGroupResponse	▪ Assignment (bytes)	The `Assignment` field contain the assignment for a member of the group. The group coordinator sends back the information to each member. The request does not contain any indication of the protocol type used because both ends have already the information. It means that the request could not be handled independently without knowing the prior information exchanged between the members and the coordinator.	**GAP**

DescribeGroupsResponse	<ul><li>MemberMetadata (bytes)</li><li>MemberAssignment (bytes)</li></ul>	The `MemberMetadata` and `MemberAssignment` contain the metadata and the assignment of a member. The response contains a `ProtocolType` field which indicate the protocol used by the group. The response can be handled independently.	**OK**

Proposed Changes

In order to fill in the gaps which have identified in the Group API, we propose to explicitly put the `ProtocolType` and the `ProtocolName` in all the requests and responses which contain the `Metadata` and/or the `Assignment`. Namely, we propose to add them in the `JoinGroupResponse`, the `SyncGroupRequest` and the `SyncGroupResponse`. See Public Interfaces for details.

With this information available, we propose to extend the `GroupCoordinator` and the `AbstractCoordinator` to verify them and to error out if they are not consistent with the Protocol Type and the Protocol Name of the group. This is not strictly required but contribute to increasing the robustness of the Group API. For instance, it could help to catch errors when a new client in implemented or when changes are made in the `GroupCoordinator`.

In the `GroupCoordinator`, during the handling of the `SyncGroupRequest`, we propose to return the `INCONSISTENT_GROUP_PROTOCOL` error if the consumer type provided by the client do not correspond to the consumer type and/or name of the group. For older version, the verification would be omitted.

In the `AbstractCoordinator`, during the handling of the `JoinGroupResponse` and the `SyncGroupResponse`, we propose to fail the future with the `INCONSISTENT_GROUP_PROTOCOL` error if the consumer type and/or name received is consistent with the consumer type known. It would behave similarly to failing to parse the metadata or the assignment.

Public Interfaces

JoinGroupRequest

The version is bumped to 7 without changing the fields.

```
{
  "apiKey": 11,
  "type": "request",
  "name": "JoinGroupRequest",
  // Version 1 adds RebalanceTimeoutMs.
  //
  // Version 2 and 3 are the same as version 1.
  //
  // Starting from version 4, the client needs to issue a second request to join group
  //
  // Starting from version 5, we add a new field called groupInstanceId to indicate member identity across 
restarts.
  // with assigned id.
  //
  // Version 6 is the first flexible version.
  //
  // Version 7 is the same as version 6.
  "validVersions": "0-7",
  "flexibleVersions": "6+",
  "fields": [
    { "name": "GroupId", "type": "string", "versions": "0+", "entityType": "groupId",
      "about": "The group identifier." },
    { "name": "SessionTimeoutMs", "type": "int32", "versions": "0+",
      "about": "The coordinator considers the consumer dead if it receives no heartbeat after this timeout in 
milliseconds." },
    // Note: if RebalanceTimeoutMs is not present, SessionTimeoutMs should be
    // used instead.  The default of -1 here is just intended as a placeholder.
    { "name": "RebalanceTimeoutMs", "type": "int32", "versions": "1+", "default": "-1", "ignorable": true,
      "about": "The maximum time in milliseconds that the coordinator will wait for each member to rejoin when 
rebalancing the group." },
    { "name": "MemberId", "type": "string", "versions": "0+",
      "about": "The member id assigned by the group coordinator." },
    { "name": "GroupInstanceId", "type": "string", "versions": "5+",
      "nullableVersions": "5+", "default": "null",
      "about": "The unique identifier of the consumer instance provided by end user." },
    { "name": "ProtocolType", "type": "string", "versions": "0+",
      "about": "The unique name the for class of protocols implemented by the group we want to join." },
    { "name": "Protocols", "type": "[]JoinGroupRequestProtocol", "versions": "0+",
      "about": "The list of protocols that the member supports.", "fields": [
      { "name": "Name", "type": "string", "versions": "0+", "mapKey": true,
        "about": "The protocol name." },
      { "name": "Metadata", "type": "bytes", "versions": "0+",
        "about": "The protocol metadata." }
    ]}
  ]
}
```

JoinGroupResponse

The version is bumped to 7 and the `ProtocolType` field is added.

```
{
  "apiKey": 11,
  "type": "response",
  "name": "JoinGroupResponse",
  // Version 1 is the same as version 0.
  //
  // Version 2 adds throttle time.
  //
  // Starting in version 3, on quota violation, brokers send out responses before throttling.
  //
  // Starting in version 4, the client needs to issue a second request to join group
  // with assigned id.
  //
  // Version 5 is bumped to apply group.instance.id to identify member across restarts.
  //
  // Version 6 is the first flexible version.
  //
  // Version 7 adds the Protocol Type.
  "validVersions": "0-7",
  "flexibleVersions": "6+",
  "fields": [
    { "name": "ThrottleTimeMs", "type": "int32", "versions": "2+", "ignorable": true,
      "about": "The duration in milliseconds for which the request was throttled due to a quota violation, or
zero if the request did not violate any quota." },
    { "name": "ErrorCode", "type": "int16", "versions": "0+",
      "about": "The error code, or 0 if there was no error." },
    { "name": "GenerationId", "type": "int32", "versions": "0+", "default": "-1",
      "about": "The generation ID of the group." },
    // New Field
    { "name": "ProtocolType", "type": "string", "versions": "7+",
      "nullableVersions": "7+", "default": "null", "ignorable": true,
      "about": "The unique name the for class of protocols implemented by the group we want to join." },
    // Nullable from version 7
    { "name": "ProtocolName", "type": "string", "versions": "0+", "nullableVersions": "7+",
      "about": "The group protocol selected by the coordinator." },
    { "name": "Leader", "type": "string", "versions": "0+",
      "about": "The leader of the group." },
    { "name": "MemberId", "type": "string", "versions": "0+",
      "about": "The member ID assigned by the group coordinator." },
    { "name": "Members", "type": "[]JoinGroupResponseMember", "versions": "0+", "fields": [
      { "name": "MemberId", "type": "string", "versions": "0+",
        "about": "The group member ID." },
      { "name": "GroupInstanceId", "type": "string", "versions": "5+",
        "nullableVersions": "5+", "default": "null",
        "about": "The unique identifier of the consumer instance provided by end user." },
      { "name": "Metadata", "type": "bytes", "versions": "0+",
        "about": "The group member metadata." }
    ]}
  ]
}
```

SyncGroupRequest

The version is bumped to 5 and the `ProtocolType` and `ProtocolName` fields are added.

```
{
  "apiKey": 14,
  "type": "request",
  "name": "SyncGroupRequest",
  // Versions 1 and 2 are the same as version 0.
  //
  // Starting from version 3, we add a new field called groupInstanceId to indicate member identity across
restarts.
  //
  // Version 4 is the first flexible version.
  //
  // Version 5 adds the Protocol Type.
  "validVersions": "0-5",
  "flexibleVersions": "4+",
  "fields": [
    { "name": "GroupId", "type": "string", "versions": "0+", "entityType": "groupId",
      "about": "The unique group identifier." },
    { "name": "GenerationId", "type": "int32", "versions": "0+",
      "about": "The generation of the group." },
    { "name": "MemberId", "type": "string", "versions": "0+",
      "about": "The member ID assigned by the group." },
    { "name": "GroupInstanceId", "type": "string", "versions": "3+",
      "nullableVersions": "3+", "default": "null",
      "about": "The unique identifier of the consumer instance provided by end user." },
    // New Field
    { "name": "ProtocolType", "type": "string", "versions": "5+",
      "nullableVersions": "5+", "default": "null", "ignorable": true,
      "about": "The unique name the for class of protocols implemented by the group we want to join." },
    // New Field
    { "name": "ProtocolName", "type": "string", "versions": "5+",
      "nullableVersions": "5+", "default": "null", "ignorable": true,
      "about": "The group protocol name." },
    { "name": "Assignments", "type": "[]SyncGroupRequestAssignment", "versions": "0+",
      "about": "Each assignment.", "fields": [
      { "name": "MemberId", "type": "string", "versions": "0+",
        "about": "The ID of the member to assign." },
      { "name": "Assignment", "type": "bytes", "versions": "0+",
        "about": "The member assignment." }
    ]}
  ]
}
```

SyncGroupResponse

The version is bumped to 5 and the `ProtocolType` and `ProtocolName` fields are added.

```
{
  "apiKey": 14,
  "type": "response",
  "name": "SyncGroupResponse",
  // Version 1 adds throttle time.
  //
  // Starting in version 2, on quota violation, brokers send out responses before throttling.
  //
  // Starting from version 3, syncGroupRequest supports a new field called groupInstanceId to indicate member
identity across restarts.
  //
  // Version 4 is the first flexible version.
  //
  // Version 5 adds the Protocol Type.
  "validVersions": "0-5",
  "flexibleVersions": "4+",
  "fields": [
    { "name": "ThrottleTimeMs", "type": "int32", "versions": "1+", "ignorable": true,
      "about": "The duration in milliseconds for which the request was throttled due to a quota violation, or
zero if the request did not violate any quota." },
    { "name": "ErrorCode", "type": "int16", "versions": "0+",
      "about": "The error code, or 0 if there was no error." },
    // New Fields
    { "name": "ProtocolType", "type": "string", "versions": "5+",
      "nullableVersions": "5+", "default": "null", "ignorable": true,
      "about": "The unique name the for class of protocols implemented by the group we want to join." },
    // New Field
    { "name": "ProtocolName", "type": "string", "versions": "5+",
      "nullableVersions": "5+", "default": "null", "ignorable": true,
      "about": "The group protocol name." },
    { "name": "Assignment", "type": "bytes", "versions": "0+",
      "about": "The member assignment." }
  ]
}
```

Compatibility, Deprecation, and Migration Plan

The changes are compatible with previous versions.

Rejected Alternatives

- An alternative to adding the protocol type in the various requests and responses would consist in not doing anything and letting the proxies either build complex logic to keep information about previous exchanges or to brute force the parsing of the fields. The former is hard because proxies are usually state less. The latter sounds very error prone.

References

- [1] https://github.com/envoyproxy/envoy/issues/2852
- [2] https://github.com/envoyproxy/envoy/pull/8188
- [3] https://github.com/linkerd/linkerd2/issues/2214
- [4] https://github.com/grepplabs/kafka-proxy
- [5] https://www.confluent.io/kafka-summit-lon19/handling-gdpr-apache-kafka-comply-freaking-out
- [6] https://www.kai-waehner.de/blog/2019/09/24/cloud-native-apache-kafka-kubernetes-envoy-istio-linkerd-service-mesh/