KIP-53 - Add custom policies for reconnect attempts to NetworkdClient

Status

Current state: *Discussion*

Discussion thread:

JIRA: *KAFKA-3496*

Motivation

Currently if a Kafka client loses a connection with brokers it will wait for ‘reconnect.backoff.ms’ milliseconds before attempting to reconnect.

While this strategy works well when a client is disconnected for a short time if a single broker or the entire cluster become unavailable for a long time all clients will quickly generate a lot of connections.

In addition, developers have limited control over a client which constantly loses its connections with the cluster.

In this KIP, we propose to add a new public interface to be able to add custom policies for the reconnect attempts .

Additionally, this feature may be a first step to support active/passive clusters. For instance, a kafka producer may switch to a backup cluster if it loses all its connections from its active cluster for a configured time or after a number of failed connection attempts.

Public Interfaces

We propose to add a new public interface

```
public interface ReconnectAttemptPolicy {
        void configure(AbstractConfig configs);

        ReconnectAttemptScheduler newScheduler();

        interface ReconnectAttemptScheduler {
                long nextReconnectBackoffMs();
    }
}
```

Briefly, the class ReconnectAttemptPolicy is a factory to create a ReconnectAttemptScheduler. The ReconnectAttemptScheduler is used to get the amount of time to wait before attempting a new connection.

N.B : This design is inspired from : https://github.com/datastax/java-driver/blob/3.0/driver-core/src/main/java/com/datastax/driver/core/policies/ReconnectionPolicy.java

Below are the mains classes that changed for this proposal this list is not exhaustive) :

Connection

Currently these classes use directly the property : 'reconnect.backoff.ms'. They must be changed to accept a ReconnectAttemptPolicy as constructor argument.The ReconnectAttemptPolicy

- kafka/clients/ClusterConnectionStates
- kafka/clients/NetworkClient

Configuration

- kafka/clients/CommonClientConfigs

Consumer/Producer

- kafka/clients/producer/KafkaProducer
- kafka/clients/producer/ProducerConfig
- kafka/clients/consumer/KafkaConsumer
- kafka/clients/consumer/KafkaConfig

This new features don't break any existing users.

Proposed Changes

The ReconnectAttemptPolicy is provided by the user configuration. A kafka client should hold a single instance of ReconnectAttemptPolicy which is configured by either a ProducerConfig or ConsumerConfig.

The instance is passed to the ClusterConnectionStates through the NetworkClient.

A new ReconnectAttemptScheduler instance is created each time a connection to a broker is lost. Only one instance is maintained for each broker. This instance can be discarded when the client connection gets back.

Out-of-box, we could provide this following two implementations :

- **ConstantReconnectAttemptPolicy** : A policy which uses a constant delay between each reconnection attempt (default implementation which uses *‘reconnect.backoff.ms’*)
- **ExponentialReconnectAttemptPolicy** : A policy which exponentially grows the delay between attempts.

The ReconnectAttemptPolicy will be provided by the user configuration :

This is an exemple :

```
Properties config = new Properties();
config.put(ConsumerConfig.RECONNECT_ATTEMPTS_POLICY_CLASS_CONFIG, "org.apache.kafka.clients.
ExponentialReconnectAttemptPolicy");
config.put(ConsumerConfig.RECONNECT_EXPONENTIAL_MAX_DELAY_MS_CONFIG, 5000);
config.put(ConsumerConfig.RECONNECT_EXPONENTIAL_BASE_DELAY_MS_CONFIG, 50);
```

Compatibility, Deprecation, and Migration Plan

This change does not affect the behavior of existing client because we should set the ConstantReconnectAttemptPolicy as the default value for the 'reconnect.attempts.policy.class' property.

This default implementation must use the ‘reconnect.backoff.ms’ to provide a constant policy. In addition, to don't change the behaviour of all command-line tools this default policy must be used.

Rejected Alternatives

If there are alternative ways of accomplishing the same thing, what were they? The purpose of this section is to motivate why the design is the way it is and not some other way.