PIP Make Message Dispatcher Pluggable

Motivation:
The community recently discussed about server side message filtering and seems reached consensus that we don't want to put business logic in our existing dispatcher so performance for basic messaging use case is not impacted but we'll make it possible for users to use customized dispatcher with business/filtering logic to support their advanced use cases.

This PIP is a proposal for making dispatcher pluggable and able to initialize customized dispatchers.

Proposed change:
Create a dispatcher factory interface with an implementation which create default dispatchers.

```
public interface DispatcherFactory {
        Dispatcher createPersistentExclusiveDispatcher(ManagedCursor cursor, SubType
subscriptionType, int partitionIndex, PersistentTopic topic, Subscription subscription);

        Dispatcher createPersistentSharedDispatcher(...);

        Dispatcher createNonPersistentFailOverDispatcher(...);

        Dispatcher createNonPersistentKeySharedDispatcher(...);

        …..
}

public class DispatcherFactoryImpl {
        Dispatcher createPersistentExclusiveDispatcher(ManagedCursor cursor, SubType
subscriptionType, int partitionIndex, PersistentTopic topic, Subscription subscription) {
                return new PersistentDispatcherSingleActiveConsumer(cursor,
                subscriptionType, partitionIndex, topic, subscription);
        }

        Dispatcher createPersistentSharedDispatcher(...);

        Dispatcher createNonPersistentFailOverDispatcher(...);

        Dispatcher createNonPersistentKeySharedDispatcher(...);

        …..
}
```

Then add configuration for factory class and nar path for customize factory implementation. In broker we'll load factory like

DispatcherFactory dispatcherFactory = (DispatcherFactory)
Class.forName("my.special.dispatcher").newInstance();

then in Subscription we'll have

```
switch (consumer.subType()) {
        case Exclusive:
           if (dispatcher == null || dispatcher.getType() != SubType.Exclusive) {
              previousDispatcher = dispatcher;
              dispatcher = dispatcherFactory.createPersistentSharedDispatcher(...);
           }
           break;
        case Shared:
        …..
```

And user can extend existing dispatcher and override filterEntriesForConsumer() method to add some customized business logic.