KIP-647: Add ability to handle late messages in streams-aggregation

Status

Current state: *Under Discussion*

Discussion thread: *here [link to be updated]*

JIRA: *Unable to render Jira issues macro, execution error.*

Motivation

Some kafka-stream-applications aggregations might face out-of-order messages, thous which arrive when respective aggregation window (including grace period) is already closed.

Current implementation of stream-windowing-aggregation does not allow any handling for such messages, just drops them with a warning.

There is related SO-question, indicating a demand for user-defined handling, however suggested solutions are just workarounds for the problem.

Public Interfaces

- *Change to TimeWindowedKStream.aggregate - adding overloaded method with additional parameter.*

Proposed Changes

- Add overloaded aggregate method, which accepts additional **lateMessagesTopicName** as last parameter:

```
<VR> KTable<Windowed<K>, VR> aggregate(final Initializer<VR> initializer,
                                       final Aggregator<? super K, ? super V, VR> aggregator,
                                       final Named named,
                                       final Materialized<K, VR, WindowStore<Bytes, byte[]>>
materialized,
                                       final String lateMessagesTopicName);
```

- Optionally create additional SinkNode, if respective parameter is filled.
- Conditionally forward messages to the SinkNode
- Minor change to the forward-implementation: by default each message is sent to all sub-nodes, however new node-for-late-messages should be excluded from generic processing.

Compatibility, Deprecation, and Migration Plan

Suggested changes are fully backward-compatible, no migration needed.

Rejected Alternatives

Alternative approach would be providing api-users ability to define additional handler for late-messages, e.g.

```
<VR> KTable<Windowed<K>, VR> aggregate(final Initializer<VR> initializer,
                                       final Aggregator<? super K, ? super V, VR> aggregator,
                                       final Named named,
                                       final Materialized<K, VR, WindowStore<Bytes, byte[]>> materialized,
                                       final BiConsumer<? super K, ? super V> lateMessageConsumer);
```

However, common use-case of kafka-streams is writing data to topics.

Aggregation internally already uses topics, e.g. for state-storage, therefor writing late-messages to a dedicated topic better falls into api-style.