Extension Framework

ONOS's northbound API has an extension capability which allows applications to use extended features of devices. The idea is to allow devices to expose non-standard features, and to support them in ONOS without polluting the API with device-specific features.

There are two main categories of APIs that can be extended: selectors and treatments.

Driver behaviors

Extensions are not supplied by the core; instead they are brought by the driver for the devices that implements them. Applications are not compiled directly against the extension types. This allows them to be added in a completely dynamic way without the ONOS core needing to have any knowledge of the extensions.

There are two behaviors that need to be implemented by drivers that want to provide extensions. They are ExtensionTreatmentResolver and ExtensionTreatmentInterpreter.

The ExtensionTreatmentResolver behavior is used by applications that want to get a reference to an extension object. Calls to the resolver should return instances of an extension if the driver supports that extension.

The ExtensionTreatmentInterpreter is used by the ONOS core to map extensions to the actual OF messages that need to be sent to switches.

These behaviours need to be added to the driver definition of the driver in onos-drivers.xml:

```
...  
<driver name="ovs" extends="default"
      manufacturer="Nicira, Inc\." hwVersion="Open vSwitch" swVersion="
      2\.\..">
  <!-- other behaviors -->
  <behaviour api="org.onosproject.openflow.controller.
               ExtensionTreatmentInterpreter"
            impl="org.onosproject.driver.extensions.
               NiciraExtensionInterpreter" />
  <behaviour api="org.onosproject.net.behaviour.
               ExtensionTreatmentResolver"
            impl="org.onosproject.driver.extensions.
               NiciraExtensionInterpreter" />
</driver>
...  
```

Treatment extensions

Treatment extensions are supported through the ExtensionTreatment class. The DefaultTreatmentBuilder has an extension() method where applications can supply the ExtensionTreatments that they have received from the driver's ExtensionTreatmentResolver behavior. There is a generic setProperty API on ExtensionTreatment that allows an application to add values that might be needed by the extension.

Selector extensions

Selector extensions are supported through the ExtensionSelector class. The extension selector framework is a mirror image of the treatment framework.

Sample application usage
// Get the ExtensionResolver behavior of the driver of the switch we're talking to
Driver driver = driverService.getDriver(deviceId);
DriverHandler handler = new DefaultDriverHandler(new DefaultDriverData (driver, deviceId));
ExtensionTreatmentResolver resolver = handler.behaviour
(ExtensionTreatmentResolver.class);

// Get an instance of the NICIRA_SET_TUNNEL_DST ExtensionInstruction from the driver
ExtensionTreatment extension = resolver.getExtensionInstruction(
    ExtensionTreatmentType.ExtensionTreatmentTypes.
    NICIRA_SET_TUNNEL_DST.type());

// Set the tunnelDst property of the extension instruction.
try {
    extension.setPropertyValue("tunnelDst", Ip4Address.valueOf(1));
} catch (ExtensionPropertyException e) {
    log.error("Error setting extension property", e);
}

// Build a treatment that includes the extension (of course we could add other instructions to this treatment as well)
TrafficTreatment treatment = DefaultTrafficTreatment.builder()
    .extension(extension, deviceId)
    .build();

// Now the treatment can be used in whatever object supports treatments:
FlowRules, FlowObjectives or Intents.