Exploiting External Event Detectors to Anticipate Resource Requirements for the Elastic Adaptation of SDN/NFV Systems

I-D Updates & Others

Pedro Martinez-Julia
Network Science and Convergence Device Technology Laboratory, Network System Research Institute
National Institute of Information and Communications Technology
pedro@nict.go.jp

NMRG Meeting @ IETF 102
19 July 2018
Context: Management Targets

- **Adapt** the resources assigned to a system to its dynamic demands:
  - Attend clients with less **resources**: *Reduce cost*.
  - Attend more **clients** with the same resources: *Increase revenue*.

- **Avoid** discarding requests:
  - Important for meeting **service quality**.
  - **Essential** in emergency scenarios.

- Estimate and **anticipate** resource **demands** by considering internal telemetry and external event notifications:
  - Avoid the need for **overallocation** because of **conservative thresholds**.
  - Achieve **fast adaptation** to bursts (**flash crowd**).
while TRUE do
    event = GetExternalEventInformation()
    if event != NONE then
        anticipated_resource_amount = Anticipator.Get(event)
        if IsPolicyCompliant(anticipated_resource_amount) then
            current_resource_amount = anticipated_resource_amount
            anticipate_time = NOW
        end if
    end if
    anticipated_event = event
    if anticipated_event != NONE and
        (NOW - anticipation_time) > EXPIRATION_TIME then
        current_resource_amount = DEFAULT_Resource_AMOUNT
        anticipated_event = NONE
    end if
    state = GetSystemState()
    if not IsAcceptable(state, current_resource_amount) then
        current_resource_amount = GetResourceAmountForState(state)
        if anticipated_event is not NONE then
            Anticipator.Set(anticipated_event, current_resource_amount)
            anticipated_event = NONE
        end if
    end if
end while
Information Model: Tree Structure (I)

```plaintext
module: ietf-nmrg-nict-resource-anticipation
  +--rw events
    +--rw event-payloads
    +--rw external-events

notifications:
  +----n event

• Two **main** models:
  - Events are structured in payloads and the content of events itself (external-events).
  - For the time being, there is only one notification, which is the event itself.
```
Information Model: Tree Structure (II)

```
+-rw event-payloads
  +--rw event-payloads-basic
  +--rw event-payloads-seismometer
  +--rw event-payloads-bigdata
```

• The **event payloads** are, for the time being, composed of three types:
  - Basic: Intended to carry any arbitrary data.
  - Seismometer: Carry information about seisms.
  - Big Data: Carries notifications coming from BigData sources.
Information Model: Tree Structure (III)

---rw event-payloads-basic* [plid]
  +--rw plid string
  +--rw data? union

• The **basic payload** is able to hold any data type, so it has a union of several types.

• It is intended to be used by any source of events that is (still) not covered by other model.
  - Any source of telemetry information (e.g. OpenStack controllers)

• Is tightly interrelated to a framework to retrieve network telemetry:
  - draft-song-ntf
The seismometer payload includes the relevant information to a seism:

- Location of the incident.
- Magnitude of the incident (severity).

Other context information can be attached to the main event model (detailed below).

Additional fields can be defined in the future by extending this model.
The bigdata payload includes:

- A description of an event (or incident):
  - Arbitrary string of characters that describes the event using some higher level format (e.g. Turtle or N3 for carrying RDF knowledge items).
  - Its estimated general severity (similar to the magnitude of a seism).
Information Model: Tree Structure (VI)

```yang
++-rw external-events* [id]
   ++-rw id               string
   ++-rw source?          string
   ++-rw context?         string
   ++-rw sequence?        int64
   ++-rw timestamp?       yang:date-and-time
   ++-rw payload?         binary
```

- **Format of external events:**
  - Encapsulates the payloads introduced above.
  - Is complemented with:
    - an identifier of the message,
    - a string describing the source of the event,
    - a sequence number, and
    - a timestamp.
  - It includes a string describing the context of the event:
    - Intended to communicate the required information about the system that detected the event, its location, etc.
    - This field can be formatted with a high level format, such as RDF.
The **event notification** inherits all the fields from the model of external events defined above:

- *It is intended to allow software and hardware elements to send, receive, and interpret not just the events that have been detected and notified by, for instance, a sensor, but also the notifications issued by the underlying infrastructure controllers, such as the OpenStack Controller.*
Additional Topic:

Essential Artifacts for Intelligence Driven Networks
Clarifying some concepts...

• **AI ≠ ML:**
  - AI has a broader spectrum of methods, some of them are already exploited in the network for a long time.
  - **Perception, reasoning, and planning** are still not fully exploited in the network.

• **Intelligence ≠ Intelligent:**
  - **Intelligence** emphasizes data gathering and management:
    • Which can be processed by systematic methods or intelligent methods...
  - **Intelligent** emphasizes the reasoning and understanding of data to actually “posses” the intelligence.
Why AI in Network (and) Management?

- **Management** decisions are more and more **complex**:
  - From: Is there a problem in my system?
  - To: Where should I migrate this VM to accomplish my goals?

- **Operation environments** are more and more **dynamic**:
  - Softwarization and programmability elevate flexibility and allow networks to be totally adapted to their static and/or dynamic requirements.
  - Network virtualization enabling **network automation**.

- Network **devices** become **autonomic**:
  - They must take **complex decisions** without human intervention.
  - Zero-Touch networks exploiting fully programmable elements and advanced automation methods (ETSI ZSM).

- Why not?
  - **AI** methods are just **resources, not solutions**!
Exploiting AI in IDNET

- AI methods in IDNET will have access to a huge amount of (intelligence) data from the systems they manage.

- The knowledge derived from such data can be used to decide the strategic response to any event or situation of such networks.

- Constantly evolving model:
  - Knowledge (and Intelligence) Driven Network.
Intelligence Driven Network

• The **structure** of the network results from **reasoning** on intelligence data:
  - The network **adapts** to new situations without requiring human involvement.
  - Administrative **policies** are still enforced to **decisions**.

• Intelligence data is **managed** properly to exploit all its potential:
  - Data with high accuracy and high frequency will be processed in **real-time**.
  - Fast and **scalable** methods are essential to the objectives of the network.

• **AI algorithms** must be **adapted** to work on **network problems**:
  - Joint **physical** and **virtual** network elements form a **MAS** to achieve system goals.

• **Use cases**:
  - Predicting traffic behaviour.
  - Iterative **network optimization**.
  - Assessment of administrative policies.
Standardization Issues

To facilitate the coexistence of methods from different providers/vendors...

- The **methods** used to **retrieve** the information must be **quality assured** (assessment).

- The **types and qualities** of **information** that is retrieved from a system or object must be **consistent**.

- The **format** and **ontology** used to **represent** the information must be **compatible** (or easily translatable) across all systems.

- The **protocols** used to **communicate** (or disseminate, or publish) the information must respond to the **constraints** of their target usage.
Thanks for Your Attention

Questions?