Policy Manager

INITIAL ALPHA OVERVIEW

NEAT properties

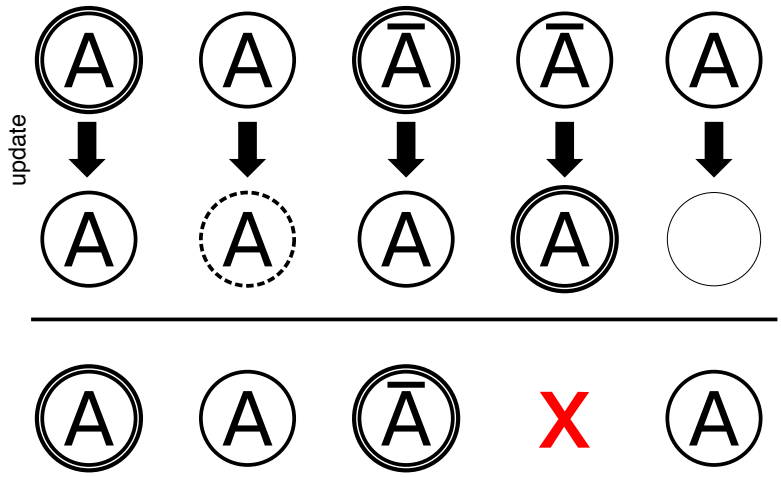
NEAT properties are `key|value` tuples describing attributes used within the NEAT Policy Manager. Properties can function as a constraint, e.g., in a policy, or as a statement, e.g., in the CIB.œ

Each property is associated with a `level` or type which identifies the "importance" of the property. Specifically the level indicates if the property may be modified by the Policy Manager logic. Currently three property levels are defined in order of decreasing precedence:

- `[immutable]` these are mandatory properties whose value cannot be changed.
- `(requested)` these are optional properties whose value may be overwritten. A mismatch of such an requested attribute will reduce the `score` of the property (see below).
- `<informational>` these are properties which have an informational nature. NEAT logic may choose to ignore these.

Two NEAT properties are considered 'equal' if their key and value attributes are identical, i.e., levels and scores are ignored when testing for equality. A comparison of two properties always yields a boolean result.

In the course of a lookup in the PM, properties from various sources will be compared and their values may be *updated*. A property's value may *only* be updated by another property whose level is greater or equal than itself. A property may only be updated by another property with the same key. When a property is updated it inherits the level of the updating property as illustrated below:



As an example, if an immutable property is requested by an application and this property clashes with the corresponding property in a connection candidate the candidate must be discarded.

In addition, each property is associated with a numeric `score` denoting whether, and how often, a property has been matched. Each time a property is updated its score is increased if its value matched the compared property, and decreased

otherwise. The property score is used to determine the most suitable NEAT connection candidate for a given request (see below).

Property ranges

In addition to single values (boolean, integer, float, ...) the value of a property may be specified as a numeric two-tuple to indicate a range range, e.g. `(5,30)` or `(100,inf)`. When a single value property is updated/compared to a property which includes a range, and the single value is within the given range, the range is replaced by the single value, and the properties are considered equal (e.g. the comparison returns true).

NEAT Policies

Policies are based around NEAT properties. Each policy contains the following entities:

- `match`: contains an object describing the properties which should trigger the policy. An empty match field will match *all* properties of a candidate. Match field properties are matched only against properties whose level is equal or higher than their own.
- `properties`: contains an object which lists a set of new properties which should be applied to the connection (if possible given the property levels).

NEAT Candidates

`NEATCandidate` objects are used to represent a candidate connections which will be passed to the NEAT logic.

NEAT Requests

A *NEAT Request* is an object containing a set of `NEATProperties` requested for a connection by an NEAT enabled application. In addition, the object includes a list of connection `candidates` (`NEATCandidate`) whose properties match a subset of the requested properties. The candidate list is populated during the CIB lookup phase and is ranked according to the associated property scores.

Each NEAT request is processed in two steps:

1. **CIB Lookup**: the request properties are compared against each entry in the CIB. The properties of a candidate are the union of the request and CIB entry property sets. Specifically, the properties are obtained by overlaying the request properties with the properties of a single CIB entry and updating the *intersection* of the two property sets with corresponding the values from the CIB entry properties.

The *N* entries with the larges aggregate score are appended to the candidate list.

1. **PIB Lookup**: For each candidate the PM iterates through all PIB policies and compares the match properties with the candidates properties. A policy is said to *match* a candidate whenever *all* of its match properties are found in the candidate properties. PIB entries are matched with a *shortest match first* strategy, i.e., policies with the smallest number of `match` properties are applied first. Subsequent, policies will *overwrite* any perviously applied policy properties. Conflicting policies must be identified by the NEAT logic.

CIB format

The CIB is made up of three repositories

- `local` (L)
- `connection` (C)
- `remote` (R)

TODO....

A lookup for a *request* currently involves the following steps:

1. Search the `remote` repository for specific DNS names or IP addresses.
2. If there is a match, obtain and match associated `connection` and `local` properties. If these searches fail, search and match only the `local` repository.

The following candidate cases are possible with n,m >= 1:

- n remote, n connection, n local
- n remote, m local
- m local

Example

Setup

Consider a host with three local interfaces `en0`, `en1`, `ra0`. Two of the interfaces, `en0`, `en1`, are wired while `ra0` is a 3G interface. The currently known network properties are stored in the following CIB entries:

```
A: {[local_ip: 10.2.0.1], [interface|en0], [is_wired: True], [MTU|9600], [remote_ip: 10.1.23.45],
<transport|TCP>, [capacity|10G], <dns_name|backup.example.com>}

B: {[local_ip: 192.168.1.2], [interface|en1], [is_wired: True], [MTU|1500], <transport|TCP>,
<transport|UDP>, [capacity|10G]}

C: {[local_ip: 10.10.2.2], [interface|ra0], [is_wired: False], [MTU|890], [remote_ip: 10.1.23.45]
[capacity|60M]}
```

An application would like to open a new TCP connection using NEAT to a destination host `d1` with the IP 10.1.23.45. Further the MTU should be 1500 bytes if possible. We denote this NEATRequest as follows:

```
Request: {[remote_ip|10.1.23.45], (MTU|1500), (transport|TCP)}
```

Further assume a "bulk transfer" policy is configured on the host. This policy is triggered by a specific destination IP, which is known to be the address of backup NFS share:

```
Policy "Bulk transfer": {(dst_ip|10.1.23.45)} ==> {[capacity|10G], (MTU|9600)}
```

Another configured policy is configured to enable TCP window scaling on 10G links, if possible:

```
Policy "TCP options": {(MTU|9600), (is_wired|true)} ==> {(TCP_window_scale|true)}
```

Outcome

The NEAT request yields three candidates after the CIB lookup:

```
Candidate 1: {[local_ip: 10.2.0.1], [interface|en0], [is_wired: True], [MTU|9600]+1, [remote_ip:
10.1.23.45]+1, (transport|TCP)+1, [capacity|10G], <dns_name|backup.example.com>}

Candidate 2: {[local_ip: 192.168.1.2], [interface|en1], [is_wired: True], [MTU|1500]+1,
(transport|TCP)+1, <transport|UDP>, [capacity|10G], [remote_ip: 10.1.23.24]}

Candidate 3: {[local_ip: 10.10.2.2], [interface|ra0], [is_wired: False], [MTU|890]-1, [remote_ip:
10.1.23.45]+1, [capacity|60M], (transport|TCP)}
```

The scores of the CIB properties which match the request properties have been increased

---

In the next step the policies are applied starting with the "Bulk transfer" policy which has the smallest number of match entries. Candidate 1 becomes:

```
Candidate 1: {[local_ip: 10.2.0.1], [interface|en0], [is_wired: True], [MTU|9600]+2, [remote_ip:
10.1.23.45]+1, (transport|TCP)+1, [capacity|10G]+1, <dns_name|backup.example.com>, [capacity|10G]}
```

and after the second policy:

```
Candidate 1: {[local_ip: 10.2.0.1], [interface|en0], [is_wired: True], [MTU|9600]+2, [remote_ip:
10.1.23.45]+1, (transport|TCP)+1, [capacity|10G]+1, <dns_name|backup.example.com>,
(TCP_window_scale|true)}
```

Next we examine Candidate 2. After the first policy the second candidate becomes:

```
Candidate 2: {[local_ip: 192.168.1.2], [interface|en1], [is_wired: True], [MTU|1500]+0,
(transport|TCP)+1, <transport|UDP>, [capacity|10G]+1, [remote_ip: 10.1.23.24]}
```

Note that the score of the MTU property was reduced, as it did not match the requested property of the "Bulk transfer" policy.

The "TCP options" policy is not applied as the candidate does not match the policy's MTU property.

The third candidate is invalidated because the "Bulk transfer" policy contains an immutable property requiring a capacity of 10G, which candidate 3 cannot fulfil.

The two candidates are passed on to the NEAT logic