QOS_AGGREGATE V1.4

Release history

Version	Author	Comments
1.0	Gijsbert Wiesenekker	Initial release.
1.1	Gijsbert Wiesenekker	Supports multiple monitors, source and alarm expressions.
1.2	Gijsbert Wiesenekker	Supports delta qos, delta alarms, speed QoS, speed alarms and receive alarms.
1.3	Gijsbert Wiesenekker	Code cleanup. QoS messages are now prefixed with the name of the probe.
1.4	Gijsbert Wiesenekker	The documentation is now in PDF format.

Description
This probe subscribes to certain QoS messages and performs calculations on the subscribed QoS messages. You can for example take the average of three QoS values. The aggregate will be published as a QoS.

Installation
Import the probe into the archive.
Install the corresponding `Perl_` package from the `Archive` on the robot that will run `qos_aggregate` if it is a Unix robot.
Install `ActiveState Perl 5.14` on the robot that will run `qos_aggregate` if it is a Windows robot. Note that `ActiveState Perl` is not available in the `Archive`.
Install the `SDK_Perl` package from the `Archive` on the robot that will run `qos_aggregate`.
Install the probe from the `Archive`.

Usage
Double click the probe in Infrastructure Manager to raw configure it or edit the configuration file with a text-editor (recommended):

Name	Optional/Required	Description
`interval`	Optional. The default is `300`.	The interval at which the probe should check if all QoS messages have been received for the defined monitors. The interval should be smaller than any of the intervals for all QoS watchers (see below).

The `<monitors>` section defines the monitors. For each monitor you specify:

Name	Optional or required	Description
`description`	Optional	A description of the monitor. If description is not defined no aggregated QoS will be published and no aggregated alarm will be generated. You can still specify QoS messages to subcribe to. This allows you to calculate delta QoS, delta alarms, speed QoS and speed alarms without having to generate an aggregate QoS.
`name`	Required	The name for the aggregate QoS. THE NAME OF THE PROBE WILL BE PREFIXED TO IT.
`source`	Required	The source of the QoS. You can use a static value but also any valid Perl `eval()` expression as described below.
`target`	Required	The target of the QoS. You can use a static value but also any valid Perl `eval()` expression as described below.
`expression`	Optional	Any valid Perl `eval()` expression to calculate the aggregated QoS as described below.
`alarm`	Optional	A valid Perl boolean expression to determine if an alarm has to be sent for the aggregated QoS as described below.
`interval`	Optional	The interval in which all of the subscribed QoS messages should occur.
`delta_target`	Optional	If specified the difference between successive aggregated QoS values will be published as a QoS. The target of the QoS will be equal to the `target` of the aggregated QoS postfixed by `delta_target`. So if the `target` is ‘average’ and `delta_target` is ‘_delta’, the delta target will be ‘average_delta’.
`delta_alarm`	Optional	A valid Perl boolean expression to determine if an alarm has to be sent for the aggregated QoS delta as described below.

Name	Optional or required	Description
`speed_target`	Optional	If specified the difference between successive aggregated QoS values divided by the difference between successive aggregated QoS sample times will be published as a QoS. The target of the QoS will be equal to the `target` of the aggregated QoS postfixed by `speed_target`. So if the `target` is 'average' and speed_target is '_speed', the speed target will be 'average_speed'.
`speed_alarm`	Optional	A valid Perl boolean expression to determine if an alarm has to be sent for the aggregated QoS speed as described below.

The qos section specifies the QoS messages to subscribe to as follows:

Name	Optional or required	Description
`name`	Required	The name of the QoS to subscribe to.
`source`	Required	The source of the QoS to subscribe to.
`target`	Required	The target of the QoS to subscribe to.
`delta_target`	Optional	If specified the difference between successive QoS values will be published as a QoS. The `target` of the QoS will be equal to the `target` of the QoS postfixed by `delta_target`. So if the `target` is 'System' and `delta_target` is '_delta', the delta target will be 'system_delta'.
`delta_alarm`	Optional	A valid Perl boolean expression to determine if an alarm has to be sent for the QoS delta as described below.
`speed_target`	Optional	If specified the difference between successive QoS values divided by the difference between successive QoS sample times will be published as a QoS. The target will be equal to the target of the QoS postfixed by `speed_target`. So if the `target` is 'System' and `speed_target` is '_speed', the speed target will be 'System_speed'.
`speed_alarm`		A valid Perl boolean expression to determine if an alarm has to be sent for the QoS speed as described below.

Name	Optional or required	Description
receive_alarm	Optional	If specified an alert will be generated if a QoS has not been received during (receive_alarm * samplerate) seconds.

The source of the subscribed QoS messages is stored in a Perl array @s in the order specified in the configuration file, so
$s[0] will contain the source of the first subscribed QoS message,
$s[1] will contain the source of the second subscribed QoS message,
$s[2] will contain the source of the third subscribed QoS message,
etc.
This allows you to construct a value for the source of the aggregated QoS using a Perl string expression like $s[0] . $[s1] . $s[2]

The subscribed QoS messages are collected in a buffer. The buffer contains the sampletime and the samplevalue of the QoS messages.
When all slots in the buffer have been filled, the probe checks if the time between the oldest and newest message is less than or equal to the interval time. If so, the aggregate value is calculated as follows:

The samplevalues of the subscribed QoS messages are stored in a Perl array @v in the order specified in the configuration file, so
$v[0] will contain the samplevalue of the first subscribed QoS message,
$v[1] will contain the samplevalue of the second subscribed QoS message,
$v[2] will contain the samplevalue of the third subscribed QoS message,
etc.
The samplevalues of the subscribed QoS messages sorted in numerical order from low to high are stored in a Perl array @w.

The QoS expression and alarm expression can refer to the arrays @v and @w. In addition you can use the variable $e that will be set to the value of the aggregated QoS in an alarm expression.
Two examples:
If you want to alert if the aggregated QoS value is below 10 you use the alarm expression:
alarm = $e < 10
If you want to alert if one of the three QoS metrics is 50% below the average of the other two you can use the alarm expression:
alarm = $w[0] < ($w[1] + $w[2])/2.0 * 0.50

The Perl variable $d will be set to the value of the difference between successive (aggregated) QoS values. You can use that variable in an alarm expression, for example if you want to alert if the difference is larger than 10 you use the alarm expression:

```
delta_alarm = $d > 10
```

The Perl variable $s will contain the value of the speed between successive (aggregated) QoS speeds. You can use that variable in an alarm expression. For example, if you want to alert if the speed lies between 1 and 10 you use the alarm expression:

```
speed_alarm = ($s > 1) && ($s < 10)
```

The following configuration file gives an example on how to subscribe to three QoS messages and publish the average of those.

```
<setup>
   loglevel = 2
   logfile = qos_aggregate.log
   interval = 300
   <monitors>
      <0>
         description = take the average of 3 CPU values on Linux
         name = CPU_USAGE_AVERAGE
         source = $s[0]
         target = average
         expression = ($v[0] + $v[1] + $v[2])/3.0
         alarm = $e > 2
         interval = 600
         <qos>
            <0>
               name = QOS_CPU_USAGE
               source = centos64int123
               target = User
               receive_alarm = 3.5
            </0>
            <1>
               name = QOS_CPU_USAGE
               source = centos64int123
               target = System
               receive_alarm = 3.5
            </1>
            <2>
               name = QOS_CPU_USAGE
               source = centos64int123
               target = Idle
               receive_alarm = 3.5
            </2>
         </qos>
      </0>
...</monitors>
</setup>
```