



# Controlling flows execution

Kazuhito Yokoi

- We'd like to control the number of messages or priority in flows. For example, a flow use 50% CPU time and other flows share another 50% CPU time when flows handle a lot of messages.
- There will be a risk that Node-RED process is stopped when a user simply set low priority to a flow which retains a lot of messages in the queue.
- As another solution, incoming messages may have to be limited based on the size of queue in each node of the flow.

#	Methods	Challenge level	Priority setting using existing function	Risk of queue overflow
1	EventEmitter extension to support priority	Difficult (We need to discuss it with Node.js community)	None (implementation is needed)	High
2	Flow control using delay node	Easy	Available (Hitachi already implemented it)	High
3	Multi runtime	Middle (We need to define specifications)	Available (nice command in Linux)	Middle (It mitigates the risk but it cannot avoid the risk totally)
4	Incoming message control	Middle (We need to define specifications)	None	None

There're details in the next slides

# Flow control using delay node

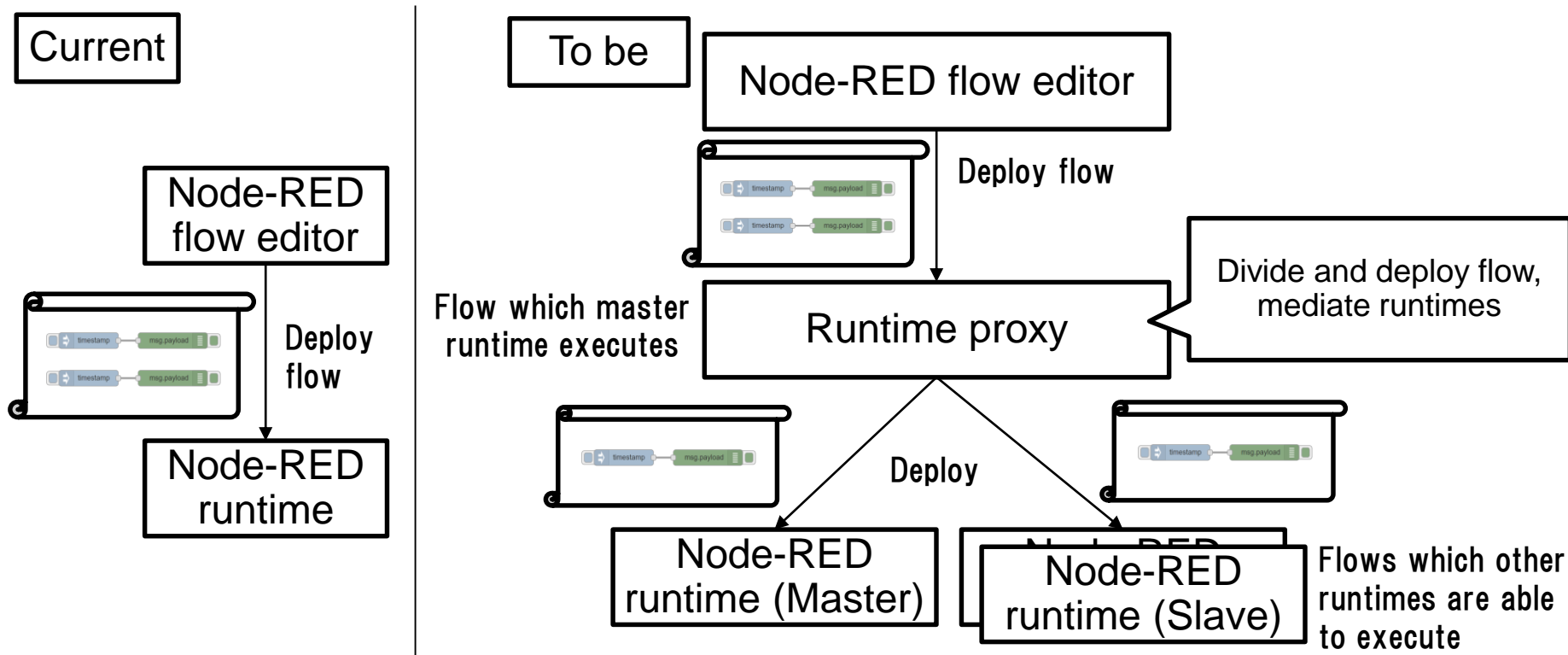
- We developed subflow node using delay nodes to set the relative limit between flows.
- While testing the subflow, we found that the Node-RED process is crashed when the message queue is full.

The screenshot displays the Node-RED web interface. At the top, there's a 'Node-RED' header with a 'Deploy' button. Below it, a tabbed interface shows 'Flow 1', 'Flow 2', 'Flow 3', 'Flow 4', and 'Flow 1'. The main workspace shows a flow with five parallel subflows. Each subflow starts with a 'timestamp' node, followed by a 'measure rate' node (showing rates like 93 msg/sec, 92 msg/sec, 92 msg/sec, and 92 msg/sec), then a 'limit rate' node (showing rates like 79%, 20%, 20%, and 20%), followed by another 'measure rate' node (showing rates like 86 msg/sec, 57 msg/sec, 55 msg/sec, and 55 msg/sec), and finally a 'msg.payload' node. A 'link' node is connected to the first 'timestamp' node. A dialog box titled 'Edit subflow instance: Subflow 1' is open, showing the 'Properties' section with 'Name' set to 'limit rate' and 'Environment Variables' set to 'rate' with a value of '79'. Below the main workspace, a detailed view of the subflow logic is shown, including 'Initialize context variables', 'Calculate the number of messages to all subflow', and a 'compare global.sum with rate' node leading to a series of 'limit' nodes (1 msg/s, 2 msg/s, 5 msg/s, 10 msg/s, 20 msg/s, 50 msg/s).

Flow inside subflow

Subflow property to set relative rate

- To control flow priority using nice command and utilize CPU resources efficiently, we'd like to use multiple runtimes to execute each flow in different processes.
- To manage multiple runtimes, runtime proxy divides flow data for each multi runtime from one flow JSON data and it mediates runtimes.
- This method mitigates the risk of queue overflow but it cannot avoid the risk totally.



- To avoid the risk of message queue overflow, the incoming messages may have to be limited at the first node in the flow.
- We'd like to suggest feedback functionality in Node-RED to limit the incoming message based on the size of queue in the following nodes.

