|  |
| --- |
| 软件工程实验I组 |
| 设计实现方案v1.0.0 |
| 对Node-RED工具的消息聚集和图形可视化拓展 |

|  |
| --- |
| 暴明坤、夏欣怡、胡俊涛、张雨濛、叶柏威、刘子渊  2020-6-12 |

版本变更历史

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 版本 | 提交日期 | 编制人 | 修改说明 | 审核人 | 版本说明 |
| V1.0.0 | 2020/6/12 | 张雨濛 | 整合各部分内容，调整格式 | 暴明坤 | 一稿 |
| V0.5.0 | 2020/6/12 | 暴明坤 | 撰写第三章部分内容及其他飞书相关节点设计说明 | 张雨濛 | 一稿 |
| V0.4.0 | 2020/6/11 | 叶柏威 | 撰写第三章部分内容及图表渲染节点设计说明 | 暴明坤 | 一稿 |
| V0.3.0 | 2020/6/11 | 刘子渊 | 撰写飞书添加目标节点、飞书删除目标节点设计说明 | 叶柏威 | 一稿 |
| V0.2.0 | 2020/6/5 | 胡俊涛 | 撰写第三章部分内容及模板渲染节点、窗口节点设计说明 | 刘子渊 | 一稿 |
| V0.1.0 | 2020/3/22 | 张雨濛 | 搭建设计实现方案框架，完成第一、二章 | 暴明坤，胡俊涛，叶柏威，刘子渊 | 初稿 |

目录

[1 概述 1](#_Toc42872034)

[1.1 编写目的 1](#_Toc42872035)

[1.2 项目背景 1](#_Toc42872036)

[1.3 项目目标 2](#_Toc42872037)

[1.4 术语和缩略词 2](#_Toc42872038)

[1.5 依从标准 4](#_Toc42872039)

[2 需求概述 4](#_Toc42872040)

[3 结构设计 5](#_Toc42872041)

[3.1 Node-RED架构 5](#_Toc42872042)

[3.2 总体设计 6](#_Toc42872043)

[3.3 数据结构设计 7](#_Toc42872044)

[3.4 出错处理设计 8](#_Toc42872045)

[4 模块详细设计 8](#_Toc42872046)

[4.1 消息聚集拓展包设计 8](#_Toc42872047)

[4.1.1 飞书配置节点 8](#_Toc42872048)

[4.1.2 飞书获取群列表节点 8](#_Toc42872049)

[4.1.3 飞书获取群信息节点 9](#_Toc42872050)

[4.1.4 飞书获取图片节点 9](#_Toc42872051)

[4.1.5 飞书上传图片节点 9](#_Toc42872052)

[4.1.6 飞书发送消息节点 9](#_Toc42872053)

[4.1.7 飞书接收消息节点 9](#_Toc42872054)

[4.1.8 飞书添加目标节点 9](#_Toc42872055)

[4.1.9 飞书删除目标节点 10](#_Toc42872056)

[4.2 图表绘制及渲染拓展包设计 10](#_Toc42872057)

[4.2.1 图表渲染节点 10](#_Toc42872058)

[4.2.2 模板渲染节点 11](#_Toc42872059)

[4.2.3 窗口节点 12](#_Toc42872060)

[5 参考资料 15](#_Toc42872061)

# 概述

## 编写目的

**本文档的编写目的是说明项目系统各个层次中的每一个模块（或程序）的具体设计，为软件开发、编码、测试提供依据。本文档是项目开发团队在开发阶段的重要依据，是明确所开发产品的具体功能及各个部分之间关系的具体参考。**

## **项目背景**

物联网（Internet of Things，简称IoT）是上个世纪就已经开始发展的互联网趋势，其主要目标是让尽可能多的设备连入互联网，让更多的数据通过网络连接流动，而不是低效的人-人、人-电脑传递。为解决传统命令式编程（Imperative Programming）不能适应物联网开发过程中需求的复杂性和灵活性，IBM公司带来了基于流编程（Flow-Based Programming）范式的事件驱动（event-driven）流程编排工具Node-RED，其将每个数据处理的步骤抽象为节点（node），通过拖拽创建节点、改变节点位置，以及连线表示节点之间的数据流动方向，给用户以方便且动态地编排并部署工作流（flow）的可能，而部署后的工作流则由事件触发执行。而Node-RED工具另一个顺应物联网开发需求的特性是模块化，其通过良好的代码结构设计实现了节点、流程与工具本身的解耦，并集成了可供开发人员共享节点和流程的社区，因此用户可以一键下载并热插拔节点和流程，尽可能地复用他人已经实现的数据处理需求。

目前，Node-RED在国外社区较为繁荣，但一些实用功能并没有国内开发者进行本土化，比如消息聚集拓展包等。消息聚集拓展包可以通过即时通信软件如Telegram、Facebook Messenger、Slack等提供的机器人机制，将流程执行中的信息实时主动地反馈给用户，也可以让用户从移动端发消息给Node-RED作为事件来触发流程的执行，极大地拓展和丰富了Node-RED的使用场景。而国内即时通信软件，如阿里的钉钉，字节跳动的飞书等均未被Node-RED支持。除此以外，在这个愈发强调数据可视化的时代，现有的 Node-RED 插件中缺少通用的内容渲染模块，该模块的缺失使Node-RED在很多情况下使用受限。

## 项目目标

本项目计划完成两个方面的内容，一是借助本土办公通信软件例如飞书或钉钉等，开发Node-RED工具的本土化消息聚集拓展包，方便国内物联网开发人员的使用，填补Node-RED在中国本土化的空白；二则是为满足Node-RED中数据可视化的要求，实现推送消息的图文并茂，开发一个将数据绘制为图表和进行Markdown模板渲染的可视化拓展包。最终通过一些使用场景示例的搭建展示这两个拓展包的实用价值。

## 术语和缩略词

关于本项目中的一些专业术语如下表1所示。

表1专业术语表

|  |  |  |
| --- | --- | --- |
| 编号 | 术语/缩略语 | 描述 |
| 1 | JavaScript | JavaScript（简称JS）是一种基于原型编程、解释型或即时编译型的编程语言，支持面向对象、命令式和函数式编程范式，最开始用作开发Web页面的脚本语言，后来也运用到一些非浏览器环境中。 |
| 2 | HTML | 超文本标记语言（HyperText Mark-up Language）的缩写，HTML文本是由结构嵌套的HTML标签组成的描述性文本，HTML标签可以说明文字、图形、动画、声音、表格、链接等，可以引用其他位置上的资源，是目前用于网页内容编写的唯一语言。 |
| 3 | Node.js | Node.js是基于Chrome V8引擎开发、跨平台的JavaScript运行环境，采用事件驱动、非阻塞式I/O的模型，可以让JavaScript运行在服务器端。 |
| 4 | node | Node-RED流程中的一个节点，它们可以在流程执行时实例化，并接收、处理和发送消息。编码上node由一组定义节点功能的JavaScript文件和用于节点配置编辑页面及帮助文本页面的HTML文件构成。 |
| 5 | flow | Node-RED中的流程，一个流程由多个node组成，node之间有数据流的依赖关系，流程的起始节点一般为事件监听相关node。流程部署后，由事件（例如手动、定时器、HTTP请求等）触发流程的执行。 |
| 6 | Markdown | Markdown是一种轻量级标记语言，它允许人们使用易读易写的纯文本格式编写文档，然后转换成有效的XHTML（或者HTML）文档。 |
| 7 | 消息聚集 | 通过技术手段，将来自不同信息源的消息聚集到一个单一的用户界面上，方便用户进行查阅。 |
| 8 | npm | npm 是 Node.js 的包管理工具，用来安装各种 Node.js 的扩展.同时npm也用来代指npm的在线开源软件仓库。 |
| 9 | Node-RED | Node-RED是一个用来连接各种硬件功能的可视化物联网编程套件，使用javascript开发，基于node.js平台。 |

## 依从标准

本项目将以下面这些标准作为参考：

1. GB/T 13702-1992 计算机软件分类与代码。

2. GB/T 19003-2008 软件工程。

3. GB/T 5538-1995 软件工程标准分类法。

4. GB/T 9385-2008 计算机软件需求规格说明。

5. GB/T 5532-2008 计算机软件测试规范。

6. GB/T 18221-2000 信息技术程序设计语言。

7. GB/T 8567-2006 计算机软件文档编制规范。

# 需求概述

本项目将基于Node-RED平台进行开发，在该平台基础上拓展消息聚集和图形可视化功能。

其中消息聚集部分的功能是打通Node-RED平台与本土办公通信软件之间的壁垒，使得用户可以以低代码（可视化应用开发方法）方式使用通信软件API。在这里，我们选取了飞书作为本土办公通信软件的代表进行开发，因此消息聚集模块主要包含飞书配置节点、飞书获取群列表节点等10个节点的工作内容。

图形可视化部分则提供将数据绘制为图表和进行Markdown模板渲染的可视化的功能，主要包含图表渲染节点、富文本渲染节点和窗口节点。

（详细说明见《需求规格说明书\_v2.6.1》）

# 结构设计

## Node-RED架构

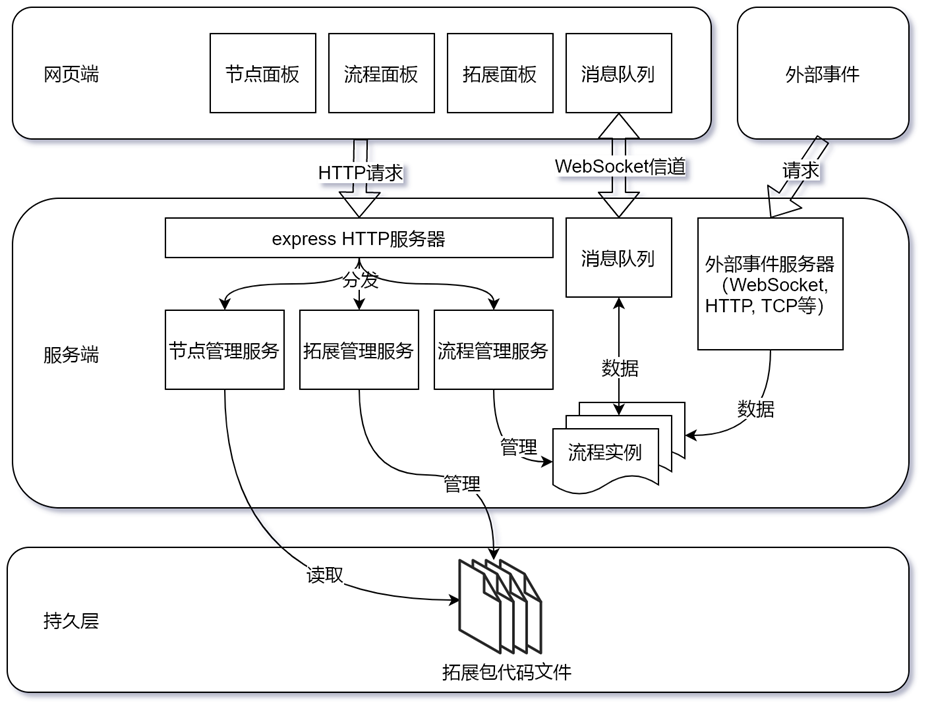


图1 Node-RED架构图

**需求规格说明书v2.6.1中指出，本项目的工作是基于Node-RED平台展开的。因此在开始拓展部分的设计之前先给出了Node-RED平台的架构图，如图1所示。Node-RED平台主要采用了较为典型的三层结构，即表示层、服务层和持久层。**

**表示层为Node-RED的网页端部分，使用常见的前后端分离、通过API与服务层交互的模式，根据功能主要可以划分为面向用户的节点面板、流程面板、拓展面板和起支持作用的消息队列。用户在节点面板、流程面板和拓展面板上的操作均使用HTTP请求发送至服务端进行处理，而消息队列通过WebSocket信道与服务端的消息队列部分连接，形成贯穿前后端的集成消息队列，可以完成通过网页向服务端运行的流程实例发送数据，以及流程实例向网页反馈数据的功能，用于诸如流程执行信息反馈等功能的实现。**

**服务层为Node-RED的服务端部分，使用Node.js语言实现。对于网页端的HTTP请求，使用Express框架作为服务中间件，分别路由到节点管理服务、拓展管理服务和流程管理服务，对于WebSocket信道则使用单独的WebSocket服务，有网页端主动发起连接并建立长连接，形成集成消息队列。节点管理服务和拓展管理服务读写持久层中的拓展包代码文件，流程管理服务根据网页端的请求创建流程实例，流程实例的执行过程中可以通过消息队列从网页端获取消息或者向网页端发送消息，或者从外部事件服务器（诸如WebSocket、HTTP、TCP等）中获得外部事件的数据。**

## **总体设计**

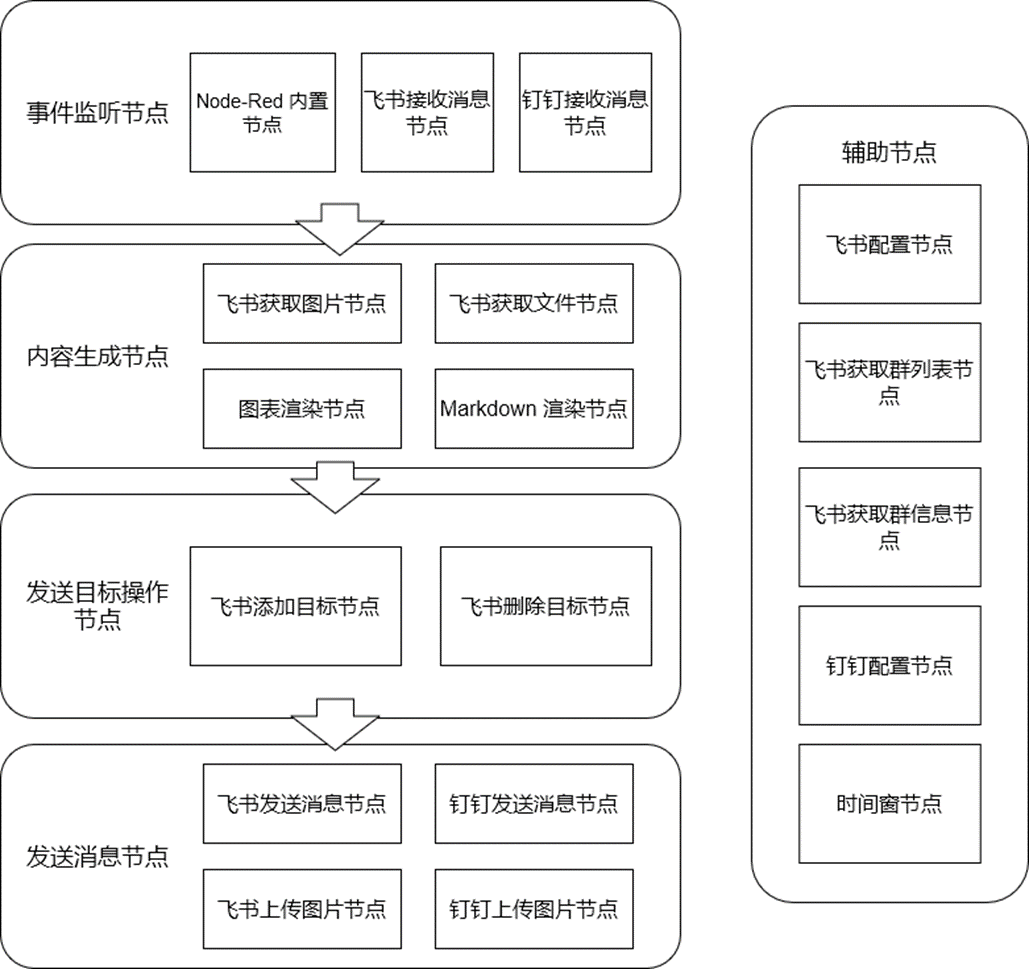


图2 Node-RED拓展结构图

本项目的工作是基于Node-RED完成消息聚集和图形可视化功能的拓展，其中拓展部分的结构如图2所示。项目共计开发17个拓展节点，这17个拓展节点根据其功能分类可以分为五类，分别是事件监听节点、内容生成节点、发送目标操作节点、发送消息节点和辅助节点。

## 数据结构设计

**该部分主要是说明本组对消息结构的设计，其中消息主要的负载放在msg.payload属性下，而关于飞书目标相关的数据放置在msg.feishu\_meta\_data属性下。具体的定义如下:**

**export interface MetaInfo {**

**target: {**

**chat: (string | Chat)[],**

**user: string[]**

**},**

**chat: (string | Chat)[],**

**user: string[]**

**}**

**export interface Chat {**

**avatar: string,**

**chat\_id: string,**

**description: string,**

**i18n\_names?: {**

**en\_us: string,**

**ja\_jp: string,**

**zh\_cn: string,**

**},**

**members: {**

**open\_id: string,**

**user\_id: string,**

**}[],**

**name: string,**

**type: "group"**

**owner\_open\_id: string,**

**owner\_user\_id: string**

**}**

## 出错处理设计

**Node-RED平台为节点报告错误提供了两种方法，一种是向日志中写入一条错误消息，另一种是将错误抛出给下一个节点，被捕获后触发错误处理流程。第一种错误处理方式的缺点是只能在日志输出中看到，无法修复或处理这些错误，所以我们选择了第二种错误处理方式，在当前节点内捕获到错误后，将错误写入到msg.error告知下一个节点。**

# 模块详细设计

## **消息聚集拓展包设计**

* + 1. 飞书配置节点

飞书配置节点的主要设计是通过Node-RED的前端页面获得飞书APP认证所需的APP\_ID和APP\_SECRET。之后使用node.js提供的setInterval 周期性向飞书API服务器发起请求，获得或刷新当前有效的tenantToken。当节点删除时，通过setInterval返回的interval id 取消周期性任务。

* + 1. 飞书获取群列表节点

飞书获取群列表节点，对于每个接收到的消息。通过调用飞书获取机器人所在群列表API来获得机器人所在的群列表，之后将群列表更新到 msg.feishu\_meta\_info.chat属性上。

* + 1. 飞书获取群信息节点

飞书获取群信息节点，对于每个接收到的消息。通过调用飞书API获取群信息。将msg.feishu\_meta\_info.chat中的每个群拓展成包含群信息的完整对象。

* + 1. 飞书获取图片节点

飞书获取图片节点，接收消息的msg.payload应该是上传图片的image\_key。

该节点通过调用飞书的获取图片API来获得之前上传的图片。返回的图片使用Node.js Buffer。

* + 1. 飞书上传图片节点

飞书上传图片节点，接收消息的msg.payload应该是一个包含图像二进制内容的Buffer。

该节点通过调用飞书的上传图片API来将图片上传到飞书服务器上，输出的消息的msg.payload是飞书服务器返回的image\_key。

* + 1. 飞书发送消息节点

飞书发送消息节点。使用飞书的发送消息API进行消息的发送，发送的对象可以是用户私聊或者群聊。发送的内容可以是图片或文本。发送内容通过msg.pyaload.msg对象进行控制。如果是一个含有text域的对象，则发送文本消息。如果是一个含有image\_key域的对象，则发送图片消息。发送对象则通过msg.feishu\_meta\_info.target对象中的chat和user属性来进行控制。

* + 1. 飞书接收消息节点

飞书接收消息节点。首先需要通过飞书的前端，获取监听端口和URL，端口和URL应该与在飞书开放平台设置的接收URL相匹配。在获取参数后，节点启动一个express server对对应的端口和URL进行监听。当接收到飞书平台发送的HTTP请求之后，将请求打包成Node-RED message，发送到流中。

* + 1. 飞书添加目标节点

飞书添加目标节点。该节点的主要作用为对飞书元信息中的群聊/用户按照配置的筛选模式进行筛选，并将筛选后的对象添加至消息发送目标中。其中所筛选的域和筛选模式为两个正交的选项，互不干扰。

支持的筛选域为群聊的 open id，群聊的名称；用户的 open id。支持的筛选模式为列表筛选和正则表达式筛选。其中列表筛选的具体使用方式为用户提供一系列候选值，如果待筛选对象的筛选域和列表中的候选值完全匹配，则认为该候选对象被选中。正则表达式筛选模式出于效率的考虑，只允许使用一个正则表达式。正则表达式筛选模式可以额外配置是否忽略大小写的选项。

* + 1. 飞书删除目标节点

飞书删除目标节点。该节点的主要作用为对飞书消息发送目标中的群聊/用户按照配置的筛选模式进行筛选，并将筛选后的对象从发送目标中移除。其中所筛选的域和筛选模式为两个正交的选项，互不干扰。

支持的筛选域为群聊的 open id，群聊的名称；用户的 open id。支持的筛选模式为列表筛选和正则表达式筛选。其中列表筛选的具体使用方式为用户提供一系列候选值，如果待筛选对象的筛选域和列表中的候选值完全匹配，则认为该候选对象被选中。正则表达式筛选模式出于效率的考虑，只允许使用一个正则表达式。正则表达式筛选模式可以额外配置是否忽略大小写的选项。

## **图表绘制及渲染拓展包设计**

* + 1. 图表渲染节点

图表渲染节点用于对接收到的数据进行图表绘制，图形渲染基于vega，支持Bar chart, Line chart, Pie chart, Radar chart, Polar area chart五种图表类型。在使用该节点前需要对节点进行配置，需要定义图表类型，图表标题，图表坐标轴，数据窗口大小，以及选择将渲染好的png图像输出为base64字符串还是二进制格式。

图表渲染节点的输入有两种模式，每种模式都支持单个数据或Array类型的数据。若为单个数据，则数据会在节点内部累积（可以由数据窗口大小来控制节点内存储的最大数据个数）；若为Array类型的数据，则只对当前输入进行图表绘制，同时节点内部的数据存储状态被更新为当前输入。

第一种模式是将数据存放在默认的payload域，该种模式无法设置横坐标值（或者其他图表中的标签值），采用的是默认值。

第二种模式是将数据存放在x域和y域，这种模式需要将payload域置为"null"。x数据域作为横坐标值（或者其他图表中的标签值），y数据域作为纵坐标值（或者其他图表中的数据值）。同时，当x数据域存储的是date string时，可以在format域存放格式化字符串，将时间信息进一步格式化（详见d3-time-format syntax: https://github.com/d3/d3-time-format#locale\_format）

为了让图表渲染节点使用起来更加灵活，节点提供了config域来对节点配置进行覆盖。例如当config.chartType不为空时，节点使用config.chartType作为表格类型。

* + 1. 模板渲染节点

模板渲染的功能是用户预先设定一个包含若干待填字段的字符串模板，在流程实例运行的过程中根据输入数据填充这些字段，并根据用户的设置输出填充后的字符串。模板渲染默认使用了Markdown的语法，并借助Ace文本编辑器实现语法高亮。定义字段的标识符默认使用mustache的双括号，用户也可以自定定义。图3是配置Markdown模板的一个例子，其中text、image\_url、list\_a、list\_b为有效指定的字段。

模板渲染节点通过正则匹配识别模板中的字段，记录每个字段的状态，当有输入时根据输入中指定的键值更新字段状态，并替换模板中对应内容并输出。



图3 模板节点应用示例

* + 1. 窗口节点

在绘制图表的场景中，用户往往并不是一次性生成所有数据，而是在时间上积累了若干数据，再进行绘制，尤其是折线图的应用，因此为用户提供一个方便的、能够在时间上缓存数据的功能节点是比较重要的。我们参考了Flink流处理框架中的time window的设计机制，设计实现了tumbling window、sliding window和session window。

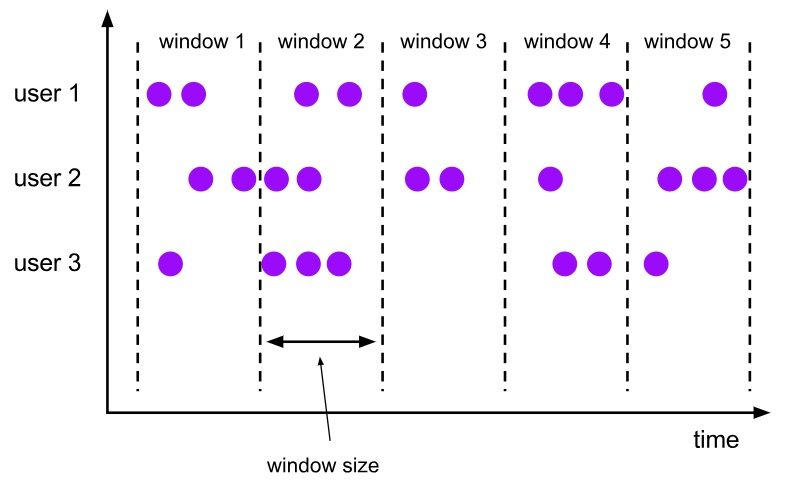


图4 tumbling window示意图

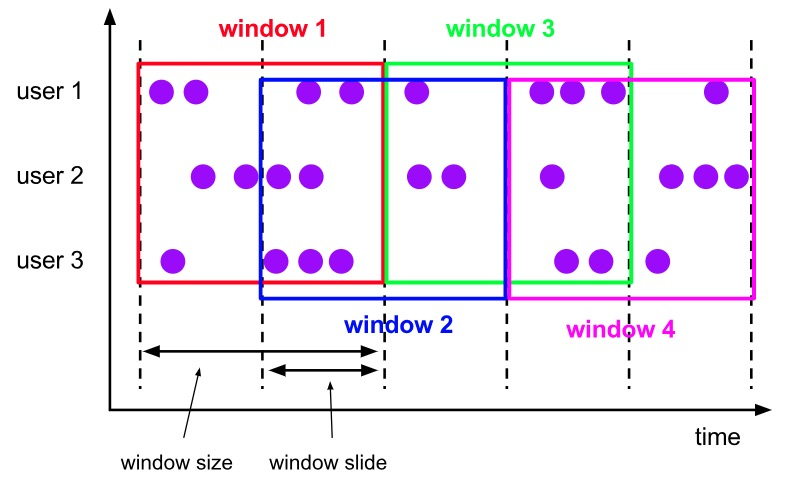


图5 sliding window示意图

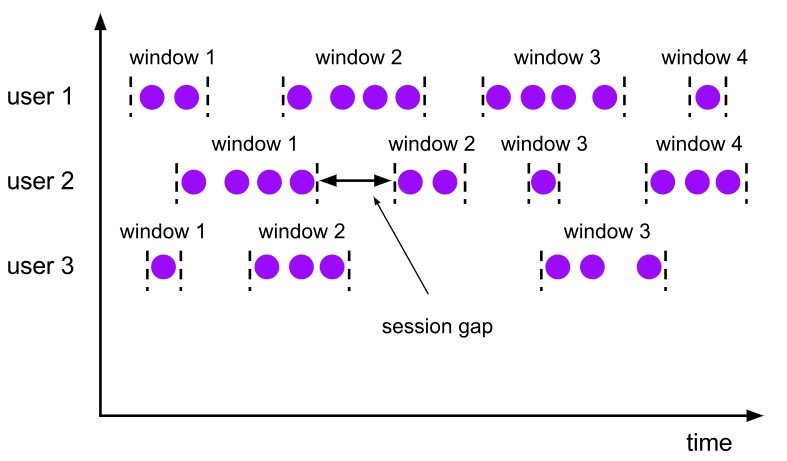


图6 session window示意图

Tumbling window中下一个窗口的起始位置是上一个窗口的结束位置，根据配置的窗口大小在时序数据中不重叠地切分数据，如图4所示。Sliding window中下一个窗口的起始位置相对上一个窗口的起始位置的距离由滑动大小指定，因此窗口像是在数据流上“滑动”切分数据，窗口之间可以存在重叠部分，如图5所示。Session window则是根据数据到来的频率，当超过一定的时间间隔时没有新数据到来时，当前窗口缓存的数据被交付处理，并清空缓存，如图6所示。出于配置灵活性的考虑，tumbling window和sliding window可以以时间或者数据的数量为标准，而session window由于其本身机制只能指定时间作为标准。每个窗口的输出为一个数组，其中每个元素都包含了收到数据的Unix时间戳和其数据，如图7的payload字段所展示。

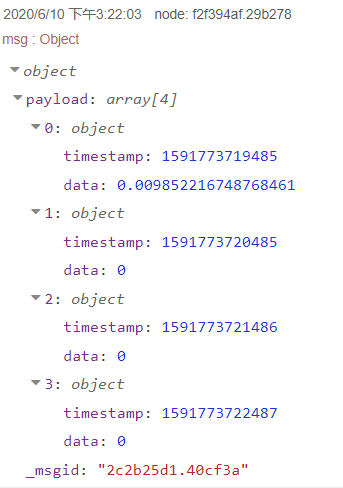


图7 窗口节点输出示意图

# 参考资料

1. Node-RED.Node-RED website[EB/OL].https://nodered.org/， 2020.
2. Node-RED.Node-RED Documentation[EB/OL].https://nodered.org/docs/， 2020.
3. OpenJS Foundation.nodejs website[EB/OL].https://nodejs.org/en/， 2020.
4. 钉钉开放平台.钉钉开发文档[EB/OL].https://ding-doc.dingtalk.com/doc#/serverapi3/rg1occ， 2019.
5. 飞书开放平台.开放平台概述 - 飞书开放平台[EB/OL].https://open.feishu.cn/document， 2020.
6. Apache Flink Community China. A Deeper Look Into Apache Flink Time And Window[EB/OL].https://www.alibabacloud.com/blog/a-deeper-look-into-apache-flink-time-and-window\_595186, 2019.
7. Roger S.Pressman. 软件工程——实践者的研究方法[M]. 机械工业出版社， 2011.
8. Leszek A.Maciaszek. 需求分析与系统设计[M].机械工业出版社， 2009.
9. Nicholas C.Zakas. JavaScript高级程序设计[M].人民邮电出版社， 2012.
10. Mike Cantelon等. Node.js实战[M]. 人民邮电出版社， 2014.