SDN详细设计文档

# 技术调研

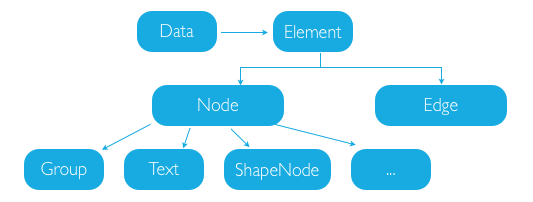
需求分析（技术难点）：

1. 节点能设置不同图片；
2. 能显示箭头动态运行的效果；
3. 连线布局，但又不是传统的布局方式（如力学或树形布局）。

|  |  |  |  |
| --- | --- | --- | --- |
| 控件 | 技术 | 优点 | 缺点 |
| Qunee | CANVAS | 国人研发，简洁、高效、用户体验好  例子丰富，文档中文  有类似的demo | 收费  bug较多 |
| D3 | SVG | 非常强大的一款绘图控件  开源免费  API完整，论坛活跃，例子丰富 | 没有类似的demo，实验了很久也没成功，后放弃 |
| Raphaël | SVG | 是跨浏览器的矢量图形库  开源免费  据说功能非常强大 | 例子简陋 |
| yFiles for HTML | CANVAS,SVG | 德国人搞的，很严谨，布局算法很厉害 | 收费 |

对比SVG和Canvas实现方式的优劣

Qunee for HTML5是一套基于HTML5的图形组建，使用Canvas技术。



可见Qunee中分为节点图元，连线图元，分组图元，与开发需求非常类似。官方例子中有箭头动态运行的例子。

经过实验和对比，最终决定采用jquery+qunee实现。

# 配置

config.xml为SDN图的配置文件，以graph为根节点，

## 节点

节点包括路由器，防火墙，服务器等，是连线的交点。通过在<nodes></nodes>中配置多个<node></node>实现。

<nodes>

<node id=*"controller"*>

<name>Controller</name>

<x>300</x>

<y>-350</y>

<img>assets/images/controller.png</img>

<visible>false</visible>

</node>

</nodes>

id：可选，节点ID，通过该ID可引用到该节点。

name：必填，节点名称，显示在节点下方的文本，可为空。

x：必填，节点的X坐标。

y：必填，节点的Y坐标。

img：必填，节点显示的图片。

type：必填，节点类型，可选值为controller，server，internet，router，firewall。

visible：可选，节点初始化后是否可见，如果为false，则初始化节点后隐藏该节点。

## 连线

连线为节点间的连线，HTTP请求的路径。通过在<edges></ edges >中配置多个<edge></ edge >实现。

<edges>

<edge id=*"edge7"*>

<start>controller</start>

<end>router7</end>

<color>#FF6B6B</color>

<arrow>true</arrow>

<dash>true</dash>

</edge>

</edges>

id：可选，连线ID，通过该ID可引用到该连线。如果设置该ID，则以显示箭头在连线上动态移动。

start：必填，起始节点ID。

end：必填，结束节点ID。

color：必填，连线颜色。

arrow：必填，连线是否带箭头。

dash：必填，连线是否为虚线。

## 分组

分组是采用云将相同类型的节点包围起来。通过在<groups></groups>中配置多个<group></ group>实现。

<groups>

<group id=*"webGroup"*>

server6,server7,server8

</group>

</groups>

id：可选，分组ID，通过该ID可引用到该分组。

属性：以逗号分隔该组中包含的节点ID。

## 文本

文本是显示在图上的文本。通过在<texts></ texts >中配置多个<text></ text>实现。

<texts>

<text>

<name>CloudWAF Based on SDN</name>

<x>-500</x>

<y>-350</y>

<fontSize>30</fontSize>

<color>#000</color>

</text>

</texts>

name：必填，文本内容。

x：必填，文本的X坐标。

y：必填，文本的Y坐标。

fontSize：必填，字体大小。

color：必填，字体颜色。

## 标签

标签是依附于节点或分组的文本，动态闪动显示。在通过在<labels></labels>中配置多个<label></label>实现。

<labels>

<label>

<data>Controller send Flow Table to OpenFlow Switchs</data>

<color>#D9EDF7</color>

<fontSize>20</fontSize>

<interval>4000</interval>

<belong>cloudGroup</belong>

</label>

</labels>

data：必填，标签显示内容。

color：必填，标签边框颜色。

fontSize：必填，字体大小。

interval：必填，标签闪动频率。

belong：必填，标签依附于哪个节点或分组。

## 路径

路径是HTTP请求运行的路径。在通过在<paths></ paths >中配置多个<path></ path>实现。

<paths>

<path id=*"path1"*>

<route>internet1,router1,server1,router1,router2</route>

<start>200</start>

<step>40</step>

<router>router5</router>

<server>server7</server>

<threshold>16</threshold>

</path>

</paths>

id：可选，路径ID，通过该ID可引用到该路径。

route：必填，以逗号分隔，HTTP请求运行所经过的节点ID。

start：必填，HTTP请求运行开始时间，单位毫秒。

step：必填，HTTP请求运行速度。

router：必填，该路径在进入server之前进入的router。

server：必填，该路径最后进入到哪个server中。

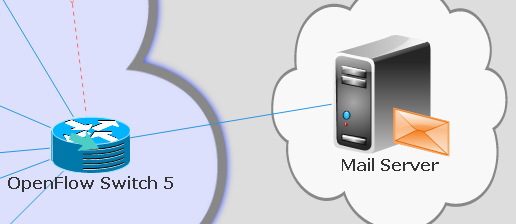
threshold：可选，控制HTTP请求是否进入router的阈值，如果没有设置，默认为step的一半。

router，threshold，server共同控制该server的显示和隐藏。

# 实现原理

一开始server=Mail Server根据节点属性visible=false被设置为隐藏。

当HTTP 请求进入router=OpenFlow Switch 5时，显示server=Mail Server。



当HTTP请求离开router=OpenFlow Switch 5时，隐藏server=Mail Server。



根据HTTP请求的x坐标与router的x坐标的距离小于threshold=step/2，判断HTTP请求是进入还是离开router。