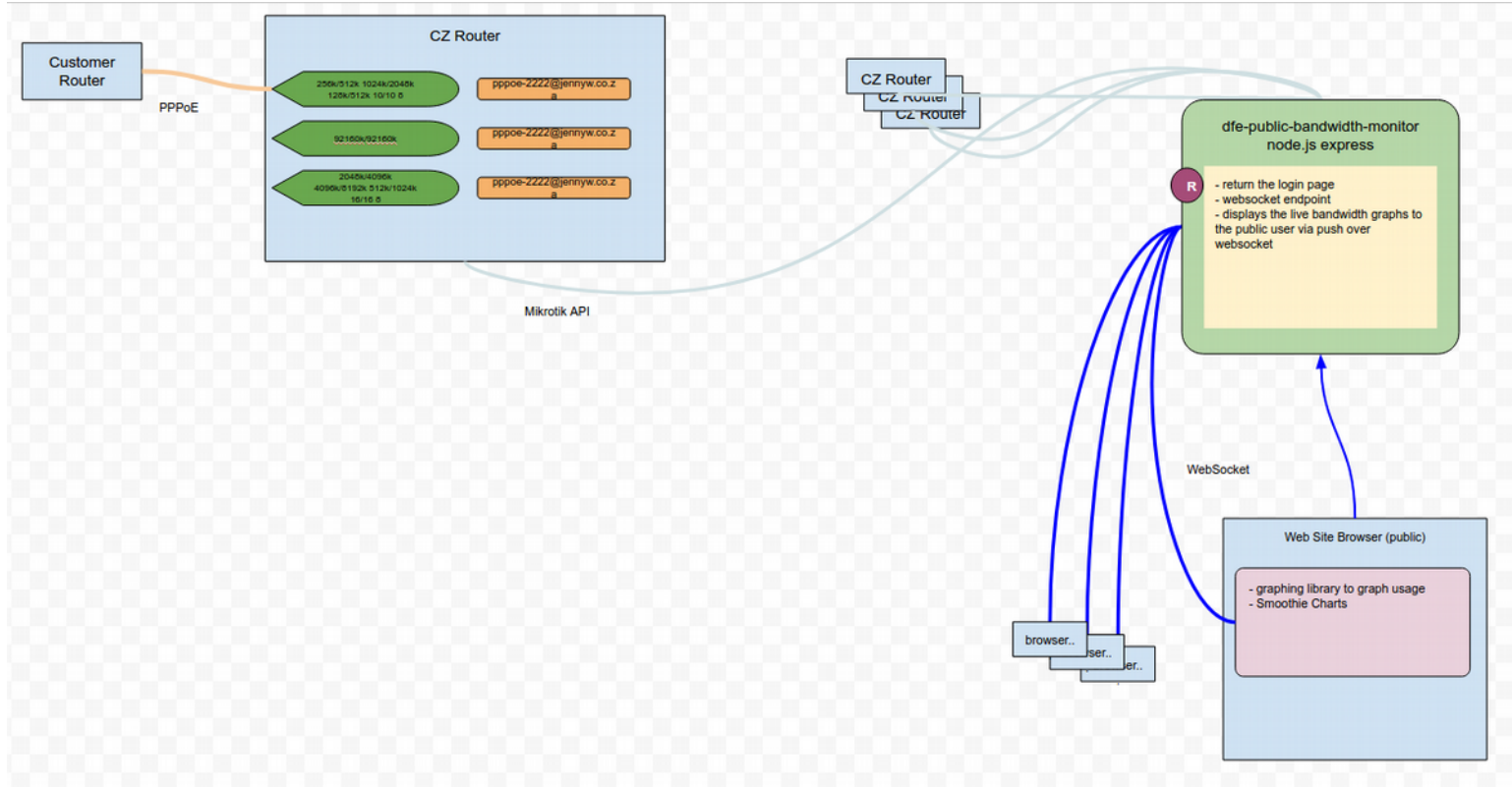


Live Bandwidth Monitoring using the Mikrotik API

Context

- Users don't know what is running in the network
- Support center load increased with the rise of video
- Customers don't have power / tools to understand their bandwidth
- Speedtest services have a major flaw, only useful on single device networks

Technical Solution Overview

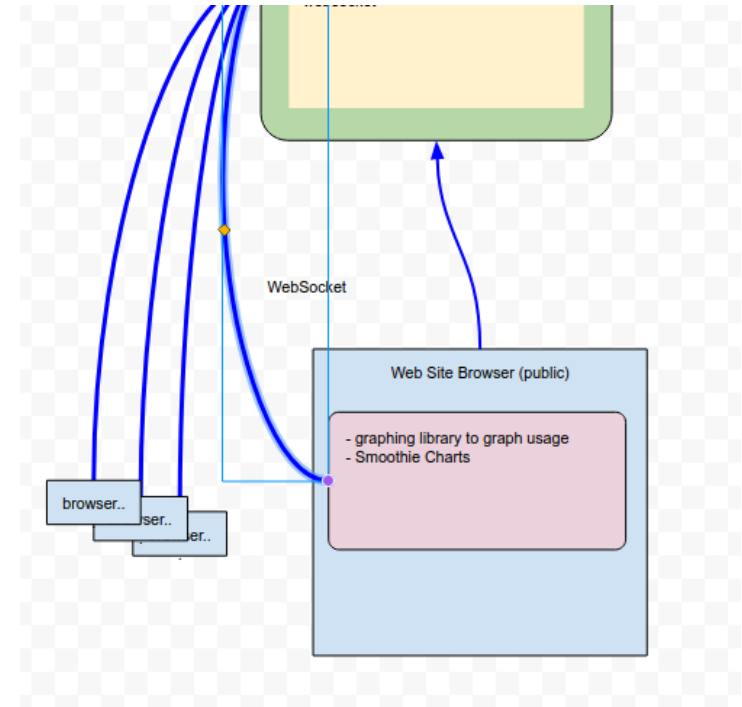


Technical Components Used

- Node.js mikronode-ng library
- Socket.io
- Smoothie.js Charts
- Mikrotik API
- Node.js web framework : express

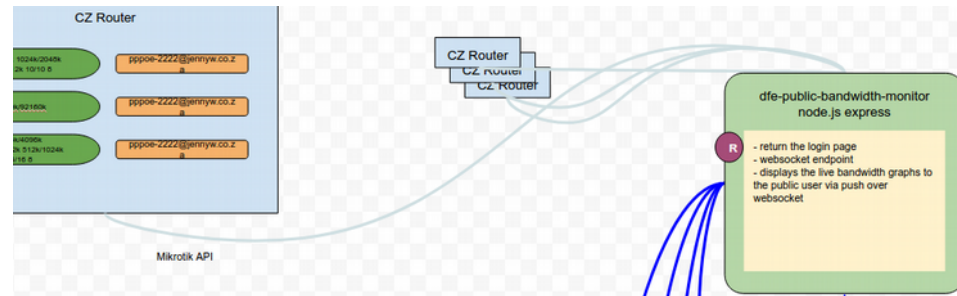
Basic Control Flow - Client Side

- User navigates to the web page with their pppoe username as a parameter
- Page loads the Smoothie.js graphing library
- Page loads the socket.io client library
- Page initiates a websocket connection to the Node.js server
- Page joins a socket.io room for the pppoe username



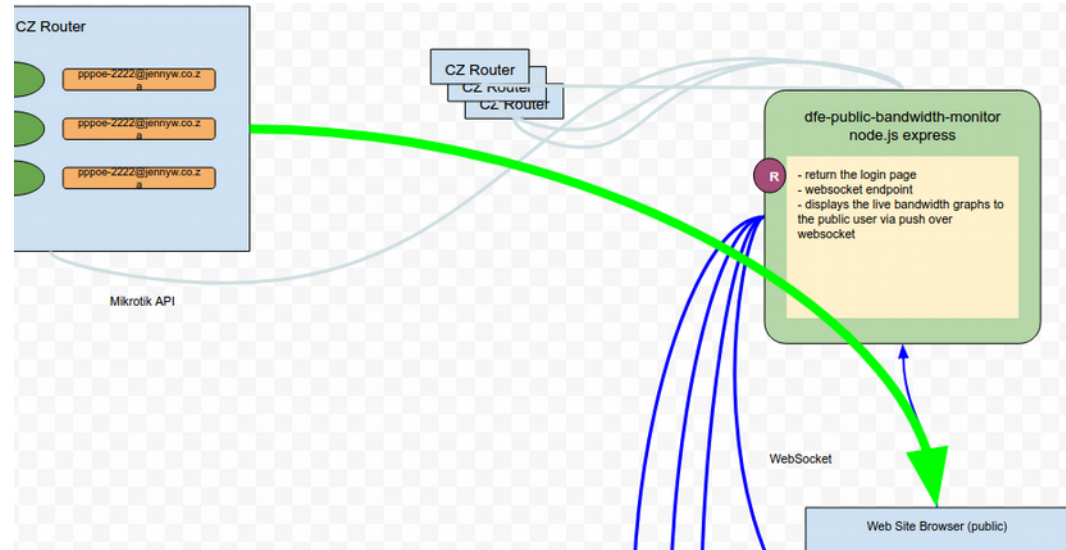
Basic Control Flow - Node.js

- Socket.io listens for a join room message from clients
- Process connects Mikrotik API on the customer router on which the pppoe interface resides
- Process issues a “/interface/monitor-traffic” with the pppoe interface as a parameter
- Process starts receiving events from Mikrotik API for bandwidth



Basic Control Flow - Updates to the client

- For every event received from the Mikrotik, emit a message on the socket.io room for the pppoe user
- The client web page receives a websocket frame and uses this as input for the Smoothie charts javascript graphing library



Final Implementation



Bandwidth Monitor

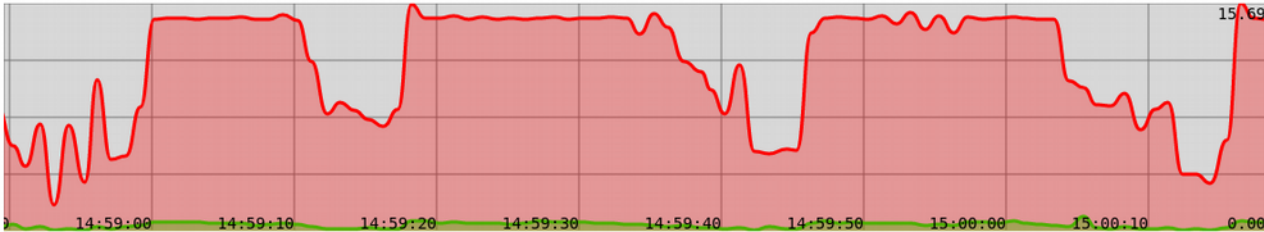
Customer : **Stucky Motors Pty Ltd**

Package : **Piet Retief Fibre Direct Internet**

Username : **4446@jennyw.co.za**

Download : **14.78** Mbps

Upload : **0.62** Mbps



Running

Refresh

TIP : You can bookmark this page for future use.

Jenny Bandwidth Monitor

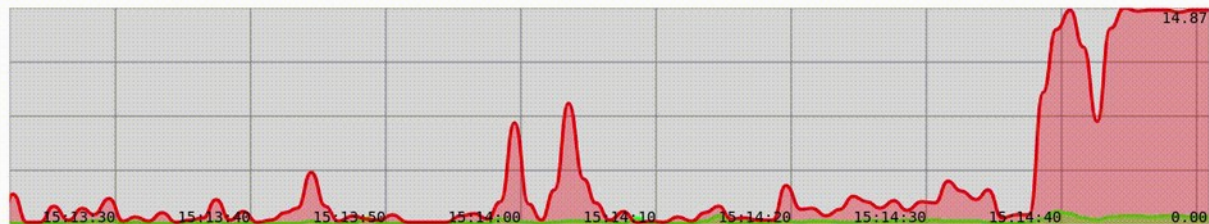
Customer : Stucky Motors Pty Ltd

Package : Plet Relief Fibre Direct Internet

Username : 4446@jennyw.co.za

Download : 14.57 Mbps

Upload : 0.58 Mbps



Running

Refresh

TIP : You can bookmark this page for future use.

Websocket frames on the browser

Data	Length	Time
42[{"bandwidth_update",{"rx":0.095062255859375,"tx":1.7361068725585938}}]	71	15:01:19.943
42[{"bandwidth_update",{"rx":0.095062255859375,"tx":1.7361068725585938}}]	71	15:01:20.236
42[{"bandwidth_update",{"rx":0.13872528076171875,"tx":4.118766784667969}}]	72	15:01:20.967
42[{"bandwidth_update",{"rx":0.13872528076171875,"tx":4.118766784667969}}]	72	15:01:21.216
42[{"bandwidth_update",{"rx":0.09008026123046875,"tx":0.9879608154296875}}]	73	15:01:21.965

▼ 42[{"bandwidth_update", {"rx": 0.1658172607421875, tx: 2.2760696411132812}}]
0: "bandwidth_update"
▼ 1: {rx: 0.1658172607421875, tx: 2.2760696411132812}
rx: 0.1658172607421875
tx: 2.2760696411132812

- no polling
- can scale to thousands of users
- multiple spectators don't reconnect to the backend

Mikrotik API Commands Code

```
//connection to router has been established
channels.chan_pppoe = conn.openChannel();
channels.chan_pppoe.closeOnDone = true;

channels.chan_pppoe.write(['/interface/pppoe-server/getall'],function(c1){
c1.on('done',function(data){
    var pppoeusers = mikronode.parseItems(data);
    var minterfaces = [];
    _.each(pppoeusers,function(v){
        minterfaces.push(v.name);
    });
    minterfaces = _.uniq(minterfaces);
    var interface_str = '=interface=' + minterfaces.join(',');

    channels.chan_monitor.write(['/interface/monitor-traffic',interface_str], function(c) {
        c.addListener('read', function(data){
            var parsed = mikronode.parseItems(data);
            update_emitter(parsed); //function that sends update on the websocket room
        });
    });
})
}
```

Frontend Code

```
var smoothie = new SmoothieChart({minValue:0, millisPerPixel:80...//init smoothie});

smoothie.streamTo(document.getElementById("mycanvas"), 1000);

// Data
var line1 = new TimeSeries();
var line2 = new TimeSeries();

// Add to SmoothieChart
smoothie.addTimeSeries(line1,{ strokeStyle:'rgb(0, 255, 0)', fillStyle:'rgba(0, 255, 0, 0.4)',
lineWidth:3 });
smoothie.addTimeSeries(line2,{ strokeStyle:'rgb(255, 0, 0)', fillStyle:'rgba(255, 0, 0, 0.3)',
lineWidth:3 });

socket.on('bandwidth_update', function(obj){
    line1.append(new Date().getTime(),line1val);
    line2.append(new Date().getTime(),line2val);
});
```

Resources

- SmoothieCharts : <https://github.com/joewalnes/smoothie/>
- Mikronode Library : <https://github.com/f5eng/mikronode-ng>
- Socket.io : <https://socket.io/>