Node-RED用レスポンスを返さないTCPサービス用プロキシの雛型

2018/03/20
ソース差し替え。バイナリ再ビルド

2018/03/19
Node-REDのTCP Requestノードを使いたいのに接続先がレスポンスを返さず利用不可になるのでとりあえず解消するために作成。

ダウンロード

tcpaddresponseproxy20180320.zip
- 2018-03-20 公開Node-RED TCP Requestノード利用の支援プロキシーサーバWindows用。ソース差し替えに伴い再ビルド版。

これは何？

Node-REDでTCP Requestノードで実行が止まってしまう場合にその事象をとりあえず回避する事が可能になる(かもしれない)プロキシーサーバです.NET Framework 4.5.1で動作確認しました。4.6でも動くでしょう。

TCP接続でリクエストを受け付けてもレスポンス(のデータ)を返さないデバイスがあります。
このデバイスをNode-REDから扱う時、データを送るだけならTCPノード(TCP OUTノード)でも問題はないのですが、データの送信順序を維持した送信ができず、そして送信側プラットフォームによってはフロー実行途中で送信ができなくなったりします。

- 順序を保ってリクエストを行うには、データの作成と通信の各ノードを順番に接続していく必要があるのですがTCPノード(TCP OUTノード)は出力の口が無いので、ノードを直列に接続できません。二股に分かれてTCPノードの実行が並列になってしまうため、実行順序が保証できなくなります。
- TCPノード(TCP OUTノード)はフローがデプロイされた時点でフロー内に存在する数だけ接続が確立してしまいます。サブフロー内に定義してもサブフローの参照の数だけ接続が確立されてしまいました。接続数の制限や、接続のためのリソースで問題が出てきそうです。
- 一度使ったTCPノードは再利用ができない場合がありました。つまりループ内でTCPノードを利用できない場合があるという事です。

実際にこのドキュメント製作者は、仕事でかかわっているある環境下で、最初は想定の順番でデータ送信されているのに時間が経つとデータ送信されなくなる事象を確認しています。 フローを無効にしてデプロイし直した途端に、送信されなかったデータがデバイスへ送り出されてきたりします。まるでデータ送信経路が目詰まりを起こしていたかのようです。

この事象は、シンプルなTCP(TCP OUT)ノードを使うフローを作って再現させることができました。仕事で使っている環境はCentOS 7上に構築されています。

必要な時だけ接続を確立し、また処理順序を保ちたい場合は入り口・出口がある(処理順序を明示できる)TCP Requestノードを利用します。

ですがTCP Requestノードはリクエストに対するレスポンスがある事を前提としているようです。今回のようなデバイスを相手にした場合レスポンスが無いので後続ノードへ遷移しません。

TCP Requestノードのプロパティ「戻り値」に“即時 - 応答を待たない”等を指定しても解決しません。

デバイスのプログラムは書き換えできません。
そこで、苦肉の策とはなりますが、通信経路途中でプロキシーを挟みプロキシーにあたかもデバイスからレスポンスがあったかのように振る舞わせることを画策しました。

プロキシー使い方

ダウンロードした tcpAddResponseProxy.exe を以下のオプション付きでコマンドプロンプトから実行してください。

```
tcpAddResponseProxy.exe <待ち受けIPアドレス> <待ち受けポート番号> <サービスIPアドレス> <サービスポート番号>
```

例えばIPアドレス 192.168.1.100:13880 で待ち受けしIPアドレス 192.168.1.20:3880 のサービスへ接続させるなら

```
tcpAddResponseProxy.exe 192.168.1.100 13880 192.168.1.20 3880
```

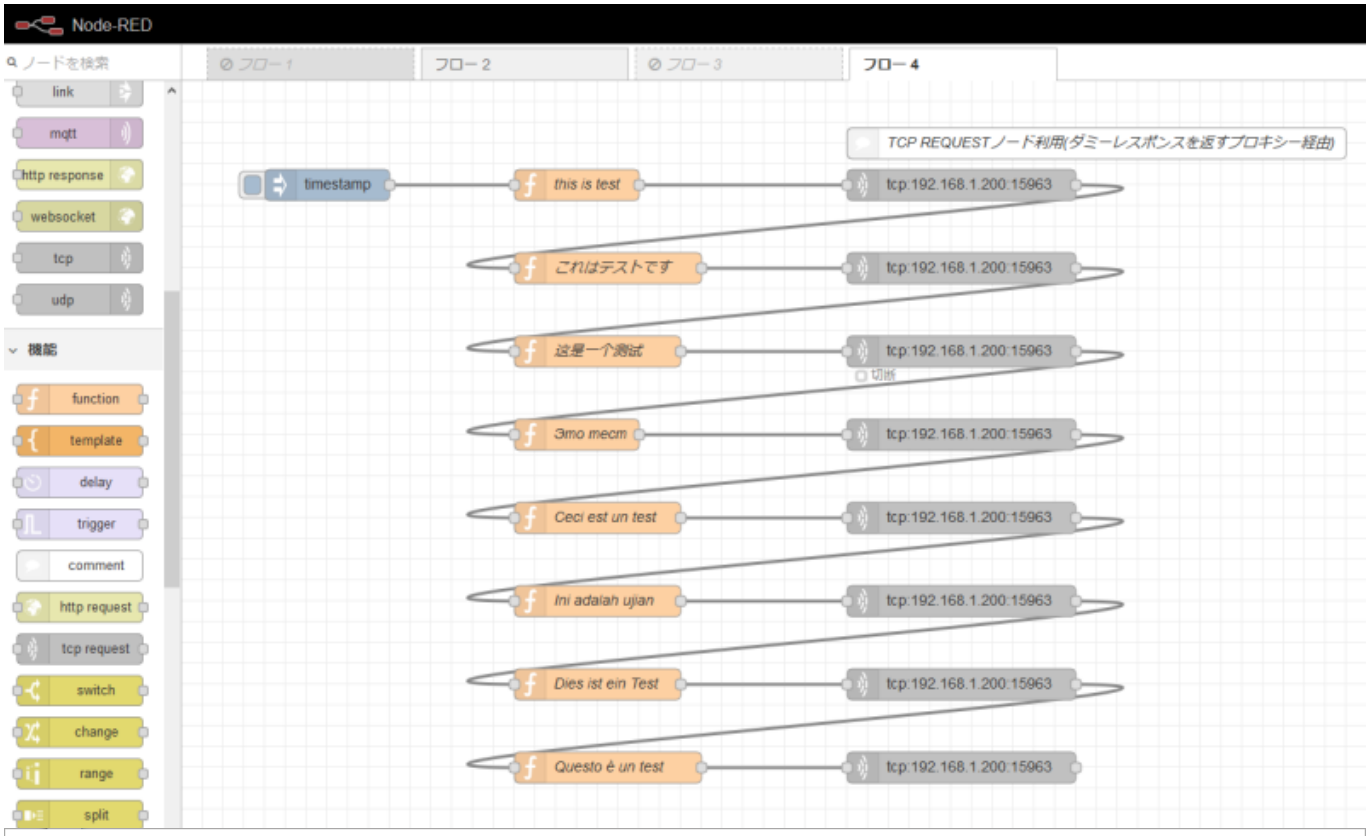
になります。

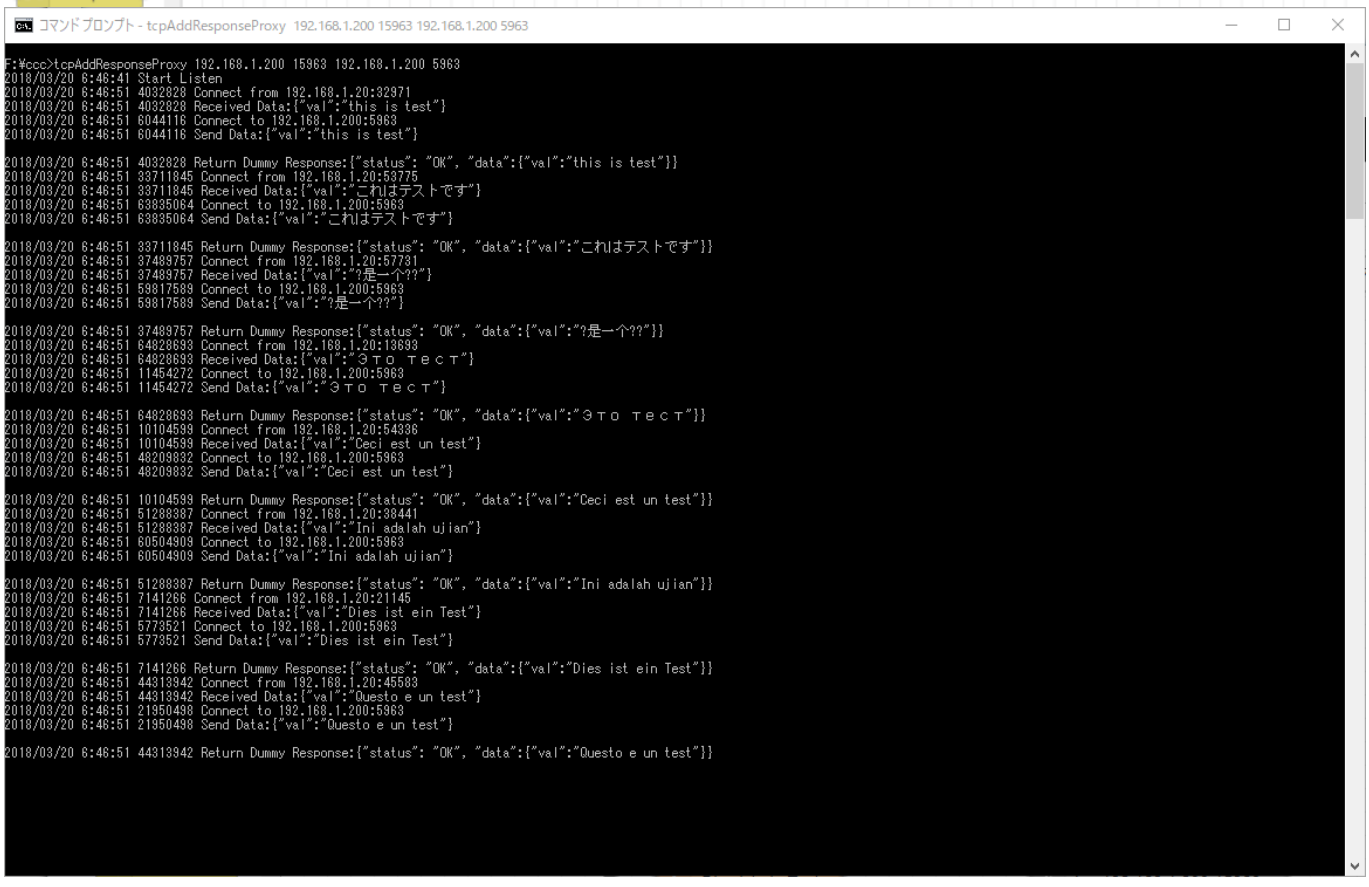
1024より小さいポート番号は指定できないと思います。管理者権限があればできるかもしれませんが試していません。

ダミーのレスポンスはJSONを模した文字列です。

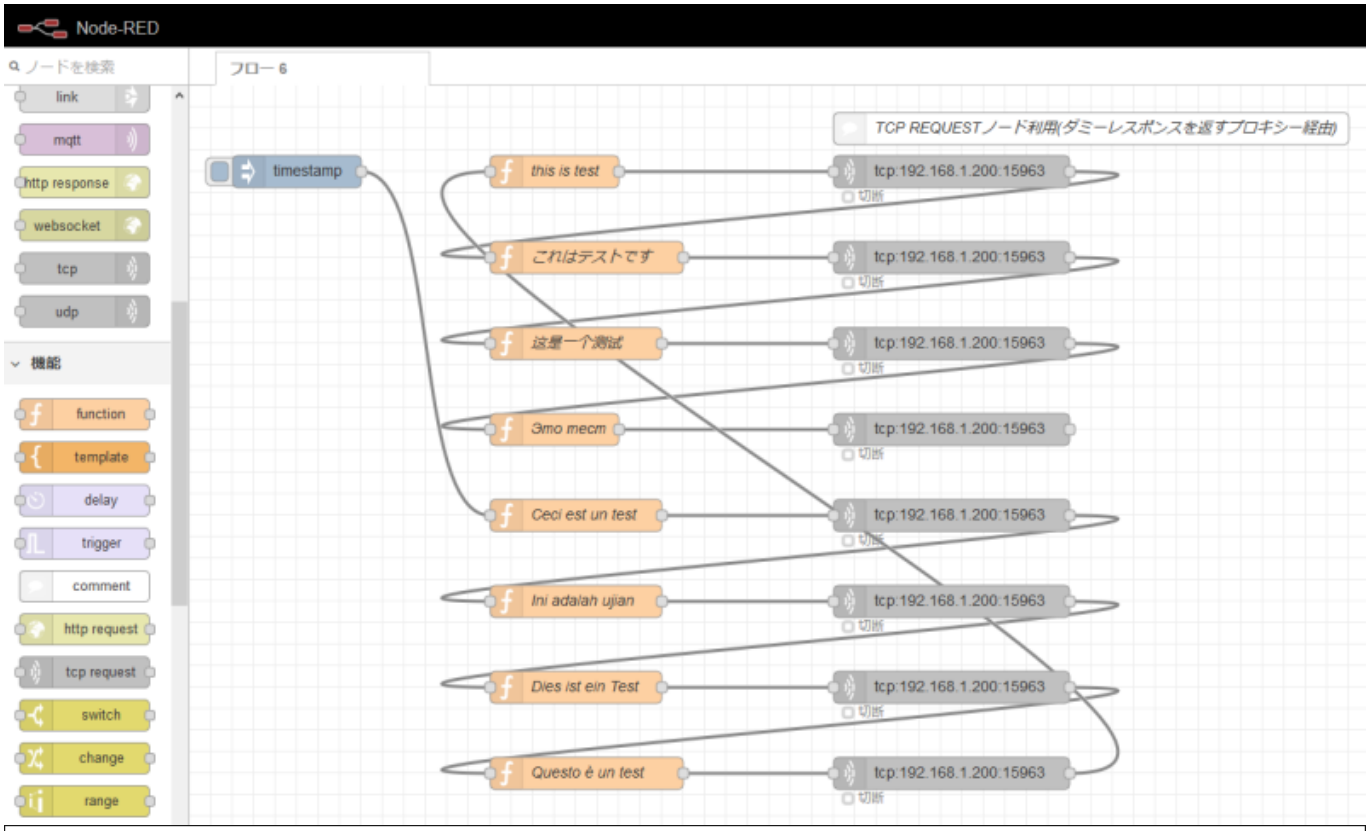
```
{"status":"OK", "data":"プロキシーが受信したデータ"}
```

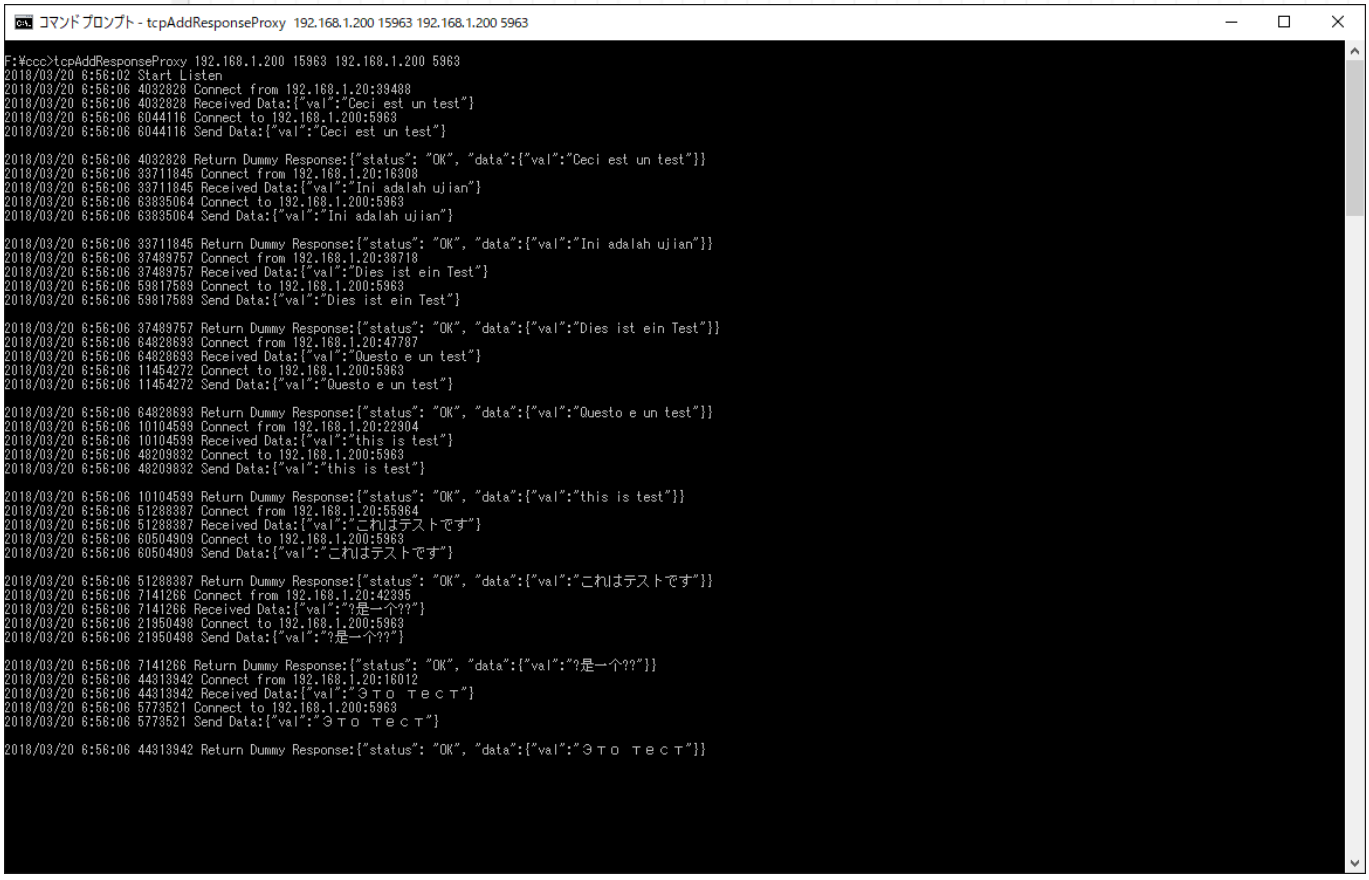
以下は資料の「フロー4」を実行した際の画面キャプチャです。
Node-REDは 192.168.1.20 で動作していて、プロキシーサーバー 192.168.1.200:15963 を通じてデバイスのエミュレーションサービス 192.168.1.200:5963 へ接続を行います。





順番を変えればきちんと通信順序も変わります。





ソース

プロキシーサーバーが受付可能な**リクエストデータのサイズは4096バイト**固定**です。手抜きです。必要ならソースを修正して再コンパイルしてください。**

make.bat

```
C:\Windows\Microsoft.NET\Framework\v4.0.30319\csc.exe /nostdlib+
/define:TRACE /target:exe /utf8output /reference:mscorlib.dll
/reference:System.dll /out:tcpAddResponseProxy.exe
tcpAddResponseProxy.cs
```

run.bat

```
@echo off
tcpAddResponseProxy.exe 192.168.1.200 15963 192.168.1.200 5936
```

tcpAddResponseProxy.cs

```
using System;
using System.Net;
using System.Net.Sockets;
using System.Text;
using System.Threading.Tasks;

namespace tcpAddResponseProxy
{
    class tcpAddResponseProxy
    {
        static void Main(string[] args)
        {
            //待ち受けアドレス、ポート
            string localEndpointStr = args[0];
            string localEndpointPortStr = args[1];

            // リダイレクト先アドレス、ポート
            string remoteEndpointStr = args[2];
            string remoteEndpointPortStr = args[3];

            IPEndPoint localEndpoint = new
IPEndPoint(IPAddress.Parse(localEndpointStr),
int.Parse(localEndpointPortStr));
            IPEndPoint remoteEndpoint = new
IPEndPoint(IPAddress.Parse(remoteEndpointStr),
int.Parse(remoteEndpointPortStr));

            proxyServer proxy = new proxyServer(localEndpoint,
remoteEndpoint);
            Task task = Task.Run(() => proxy.Listen());
            task.Wait();
            proxy.Close();
        }
    }
```

```
    class proxyClient
    {
        IPEndPoint remoteEndpoint;

        public proxyClient(IPEndPoint endpoint)
        {
            remoteEndpoint = endpoint;
        }

        public void Send(string data)
        {
            using (TcpClient client = new TcpClient())
            {
                client.Connect(remoteEndpoint);
                Console.WriteLine("{0} {1} Connect to {2}",
DateTime.Now, client.GetHashCode(), client.Client.RemoteEndPoint);

                using (NetworkStream stream = client.GetStream())
                {
                    byte[] buff = Encoding.UTF8.GetBytes(data);
                    stream.Write(buff, 0, buff.Length);
                    Console.WriteLine("{0} {1} Send Data:{2}",
DateTime.Now, client.GetHashCode(), data);
                }
            }
        }
    }

    class proxyServer
    {
        TcpListener listener;
        IPEndPoint remoteEP;

        public proxyServer(IPEndPoint localEndpoint, IPEndPoint
remoteEndpoint)
        {
            listener = new TcpListener(localEndpoint);
            remoteEP = remoteEndpoint;
        }

        public void Close()
        {
            listener.Stop();
            Console.WriteLine("{0} Stop Listen", DateTime.Now);
        }

        public async Task Listen()
        {
            listener.Start();

            Console.WriteLine("{0} Start Listen", DateTime.Now);
```

```
            while (true)
            {
                TcpClient connectClient = await
listener.AcceptTcpClientAsync();
                Console.WriteLine("{0} {1} Connect from {2}",
DateTime.Now, connectClient.Client.GetHashCode(),
connectClient.Client.RemoteEndPoint);

                Task task = Task.Run(() =>
                {
                    ProcessingData(connectClient);
                    connectClient.Close();
                });
            }
        }

        private void ProcessingData(TcpClient connectClient)
        {
            byte[] buff = new byte[4096];  // データは4096バイトまで

            StringBuilder sb = new StringBuilder();

            sb.Clear();

            using (connectClient)
            {
                using (NetworkStream stream =
connectClient.GetStream())
                {
                    // Node-REDからのリクエストを読み出す
                    int buffIndex = 0;
                    int data;

                    while (buff.Length > buffIndex)
                    {
                        data = stream.ReadByte();

                        if ((-1 == data) || ('\n' == data) || ('\r' ==
data)) break;

                        buff[buffIndex] = Convert.ToByte(data);
                        buffIndex++;
                    }
                    sb.Append(Encoding.UTF8.GetString(buff, 0,
buffIndex));
                    Console.WriteLine("{0} {1} Received Data:{2}",
DateTime.Now, connectClient.Client.GetHashCode(), sb.ToString());


                    // リクエストデータを本来のサービスへ送る
```

```
                    string dummyData;
                    try
                    {
                        proxyClient proClient = new
proxyClient(remoteEP);
                        proClient.Send(sb.ToString() + "\n");
                        dummyData = "{\"status\": \"OK\", \"data\":" +
sb.ToString() + "}";
                    }
                    catch (Exception e)
                    {
                        dummyData = "{\"status\": \"NG\", \"error\":" +
e.Message + "}";
                    }

                    // Node-REDへダミーのレスポンスを返す

                    byte[] sendData =
System.Text.Encoding.UTF8.GetBytes(dummyData + "\n");
                    stream.Write(sendData, 0, sendData.Length);
                    Console.WriteLine("{0} {1} Return Dummy
Response:{2}", DateTime.Now, connectClient.Client.GetHashCode(),
dummyData);
                }
            }
        }

    }
}
```

デバッグで使ったフロー

この資料では、デバイスのエミュレーションサービスを実行し 192.168.1.200:5963 でデバイスのサービスが動いています。
プロキシーサーバーは 192.168.1.200:15963 で待ち受けし、192.168.1.200:5963 へ接続を行います。

JSON部分をコピーしNode-REDのインポートを実行すればフローを取り込めます。

TCP OUTノードで8つのデータを送信。送信されるデータの順番は不定。

フロ□1.json

```
[
    {
        "id": "fad1b7fc.b530b8",
        "type": "inject",
```

```
        "z": "6b1a25a6.77cf5c",
        "name": "",
        "topic": "",
        "payload": "",
        "payloadType": "date",
        "repeat": "",
        "crontab": "",
        "once": false,
        "onceDelay": 0.1,
        "x": 140,
        "y": 100,
        "wires": [
            [
                "f9ecf7c6.9ef4f"
            ]
        ]
    },
    {
        "id": "f9ecf7c6.9ef4f",
        "type": "function",
        "z": "6b1a25a6.77cf5c",
        "name": "this is test",
        "func": "msg.payload =\"{\\\"val\\\":\\\"this is
test\\\"}\"+\"\\n\";\nreturn msg;",
        "outputs": 1,
        "noerr": 0,
        "x": 340,
        "y": 100,
        "wires": [
            [
                "fdc1df57.1274c8",
                "76fad514.4a91fc"
            ]
        ]
    },
    {
        "id": "fdc1df57.1274c8",
        "type": "function",
        "z": "6b1a25a6.77cf5c",
        "name": "これはテストです",
        "func": "msg.payload =\"{\\\"val\\\":\\\"これはテストで
す\\\"}\"+\"\\n\";\nreturn msg;",
        "outputs": 1,
        "noerr": 0,
        "x": 370,
        "y": 180,
        "wires": [
            [
                "1e33eff2.45229",
                "4fec608d.11b1b8"
            ]
```

```
        ]
    },
    {
        "id": "1e33eff2.45229",
        "type": "function",
        "z": "6b1a25a6.77cf5c",
        "name": "这是一个测试",
        "func": "msg.payload =\"{\\\"val\\\":\\\"这是一个测
试\\\"}\"+\"\\n\";\nreturn msg;",
        "outputs": 1,
        "noerr": 0,
        "x": 360,
        "y": 260,
        "wires": [
            [
                "33852f5b.2d1718",
                "33aac842.a223"
            ]
        ]
    },
    {
        "id": "33852f5b.2d1718",
        "type": "function",
        "z": "6b1a25a6.77cf5c",
        "name": "Это тест",
        "func": "msg.payload =\"{\\\"val\\\":\\\"Это
тест\\\"}\"+\"\\n\";\nreturn msg;",
        "outputs": 1,
        "noerr": 0,
        "x": 340,
        "y": 340,
        "wires": [
            [
                "bacb7bf6.5aeed8",
                "96b1d2aa.368a1"
            ]
        ]
    },
    {
        "id": "3d0bd521.075a2a",
        "type": "comment",
        "z": "6b1a25a6.77cf5c",
        "name": "出力のTCPノード(TCP OUT)利用",
        "info": "",
        "x": 720,
        "y": 60,
        "wires": []
    },
    {
        "id": "76fad514.4a91fc",
        "type": "tcp out",
```

```
        "z": "6b1a25a6.77cf5c",
        "host": "192.168.1.200",
        "port": "5963",
        "beserver": "client",
        "base64": false,
        "end": true,
        "name": "",
        "x": 710,
        "y": 100,
        "wires": []
    },
    {
        "id": "4fec608d.11b1b8",
        "type": "tcp out",
        "z": "6b1a25a6.77cf5c",
        "host": "192.168.1.200",
        "port": "5963",
        "beserver": "client",
        "base64": false,
        "end": true,
        "name": "",
        "x": 710,
        "y": 180,
        "wires": []
    },
    {
        "id": "33aac842.a223",
        "type": "tcp out",
        "z": "6b1a25a6.77cf5c",
        "host": "192.168.1.200",
        "port": "5963",
        "beserver": "client",
        "base64": false,
        "end": true,
        "name": "",
        "x": 710,
        "y": 260,
        "wires": []
    },
    {
        "id": "bacb7bf6.5aeed8",
        "type": "tcp out",
        "z": "6b1a25a6.77cf5c",
        "host": "192.168.1.200",
        "port": "5963",
        "beserver": "client",
        "base64": false,
        "end": true,
        "name": "",
        "x": 710,
        "y": 340,
```

```
        "wires": []
    },
    {
        "id": "96b1d2aa.368a1",
        "type": "function",
        "z": "6b1a25a6.77cf5c",
        "name": "Ceci est un test",
        "func": "msg.payload =\"{\\\"val\\\":\\\"Ceci est un
test\\\"}\"+\"\\n\";\nreturn msg;",
        "outputs": 1,
        "noerr": 0,
        "x": 360,
        "y": 420,
        "wires": [
            [
                "2025858c.64806a",
                "38794e6b.18f90a"
            ]
        ]
    },
    {
        "id": "2025858c.64806a",
        "type": "function",
        "z": "6b1a25a6.77cf5c",
        "name": "Ini adalah ujian",
        "func": "msg.payload =\"{\\\"val\\\":\\\"Ini adalah
ujian\\\"}\"+\"\\n\";\nreturn msg;",
        "outputs": 1,
        "noerr": 0,
        "x": 360,
        "y": 500,
        "wires": [
            [
                "f1cf965d.3880d8",
                "fee91adc.979fe8"
            ]
        ]
    },
    {
        "id": "f1cf965d.3880d8",
        "type": "function",
        "z": "6b1a25a6.77cf5c",
        "name": "Dies ist ein Test",
        "func": "msg.payload =\"{\\\"val\\\":\\\"Dies ist ein
Test\\\"}\"+\"\\n\";\nreturn msg;",
        "outputs": 1,
        "noerr": 0,
        "x": 360,
        "y": 580,
        "wires": [
            [
```

```
                "1d156c55.5c37ac",
                "8f5138c7.3a71a"
            ]
        ]
    },
    {
        "id": "1d156c55.5c37ac",
        "type": "function",
        "z": "6b1a25a6.77cf5c",
        "name": "Questo è un test",
        "func": "msg.payload =\"{\\\"val\\\":\\\"Questo è un
test\\\"}\"+\"\\n\";\nreturn msg;",
        "outputs": 1,
        "noerr": 0,
        "x": 370,
        "y": 660,
        "wires": [
            [
                "fe3a5a6c.0b7c78"
            ]
        ]
    },
    {
        "id": "38794e6b.18f90a",
        "type": "tcp out",
        "z": "6b1a25a6.77cf5c",
        "host": "192.168.1.200",
        "port": "5963",
        "beserver": "client",
        "base64": false,
        "end": true,
        "name": "",
        "x": 710,
        "y": 420,
        "wires": []
    },
    {
        "id": "fee91adc.979fe8",
        "type": "tcp out",
        "z": "6b1a25a6.77cf5c",
        "host": "192.168.1.200",
        "port": "5963",
        "beserver": "client",
        "base64": false,
        "end": true,
        "name": "",
        "x": 710,
        "y": 500,
        "wires": []
    },
    {
```

```
        "id": "8f5138c7.3a71a",
        "type": "tcp out",
        "z": "6b1a25a6.77cf5c",
        "host": "192.168.1.200",
        "port": "5963",
        "beserver": "client",
        "base64": false,
        "end": true,
        "name": "",
        "x": 710,
        "y": 580,
        "wires": []
    },
    {
        "id": "fe3a5a6c.0b7c78",
        "type": "tcp out",
        "z": "6b1a25a6.77cf5c",
        "host": "192.168.1.200",
        "port": "5963",
        "beserver": "client",
        "base64": false,
        "end": true,
        "name": "",
        "x": 710,
        "y": 660,
        "wires": []
    }
]
```


TCP Requestノードで8つのデータを送信。送信されるデータの順番は不定。

TCP OUTノードを置き換えただけ。

フロ□2.json


```
[
    {
        "id": "62f9652c.9212a4",
        "type": "comment",
        "z": "8ab7019d.c84638",
        "name": "TCP REQUESTノード利用",
        "info": "",
        "x": 730,
        "y": 80,
        "wires": []
    },
    {
        "id": "edf27bb4.1b91a8",
```

```
        "type": "inject",
        "z": "8ab7019d.c84638",
        "name": "",
        "topic": "",
        "payload": "",
        "payloadType": "date",
        "repeat": "",
        "crontab": "",
        "once": false,
        "onceDelay": 0.1,
        "x": 160,
        "y": 120,
        "wires": [
            [
                "7065fac5.f6107c"
            ]
        ]
    },
    {
        "id": "7065fac5.f6107c",
        "type": "function",
        "z": "8ab7019d.c84638",
        "name": "this is test",
        "func": "msg.payload =\"{\\\"val\\\":\\\"this is
test\\\"}\"+\"\\n\";\nreturn msg;",
        "outputs": 1,
        "noerr": 0,
        "x": 360,
        "y": 120,
        "wires": [
            [
                "886055bc.db3d5",
                "e3b15c0e.873448"
            ]
        ]
    },
    {
        "id": "886055bc.db3d5",
        "type": "function",
        "z": "8ab7019d.c84638",
        "name": "これはテストです",
        "func": "msg.payload =\"{\\\"val\\\":\\\"これはテストで
す\\\"}\"+\"\\n\";\nreturn msg;",
        "outputs": 1,
        "noerr": 0,
        "x": 390,
        "y": 200,
        "wires": [
            [
                "79b06b93.0b8344",
                "c3303edc.d64f88"
```

```
            ]
        ]
    },
    {
        "id": "79b06b93.0b8344",
        "type": "function",
        "z": "8ab7019d.c84638",
        "name": "这是一个测试",
        "func": "msg.payload =\"{\\\"val\\\":\\\"这是一个测
试\\\"}\"+\"\\n\";\nreturn msg;",
        "outputs": 1,
        "noerr": 0,
        "x": 380,
        "y": 280,
        "wires": [
            [
                "580aa94.83bfc58",
                "91e8110e.90a87"
            ]
        ]
    },
    {
        "id": "580aa94.83bfc58",
        "type": "function",
        "z": "8ab7019d.c84638",
        "name": "Это тест",
        "func": "msg.payload =\"{\\\"val\\\":\\\"Это
тест\\\"}\"+\"\\n\";\nreturn msg;",
        "outputs": 1,
        "noerr": 0,
        "x": 360,
        "y": 360,
        "wires": [
            [
                "144cdcaf.b0921b",
                "e5b0bc8b.e6971"
            ]
        ]
    },
    {
        "id": "e5b0bc8b.e6971",
        "type": "function",
        "z": "8ab7019d.c84638",
        "name": "Ceci est un test",
        "func": "msg.payload =\"{\\\"val\\\":\\\"Ceci est un
test\\\"}\"+\"\\n\";\nreturn msg;",
        "outputs": 1,
        "noerr": 0,
        "x": 380,
        "y": 440,
        "wires": [
```

```
            [
                "7867ed7e.a63324",
                "c23b1a7.45ab968"
            ]
        ]
    },
    {
        "id": "7867ed7e.a63324",
        "type": "function",
        "z": "8ab7019d.c84638",
        "name": "Ini adalah ujian",
        "func": "msg.payload =\"{\\\"val\\\":\\\"Ini adalah
ujian\\\"}\"+\"\\n\";\nreturn msg;",
        "outputs": 1,
        "noerr": 0,
        "x": 380,
        "y": 520,
        "wires": [
            [
                "4ddd353c.6aa1f4",
                "a2b10368.ce42e8"
            ]
        ]
    },
    {
        "id": "4ddd353c.6aa1f4",
        "type": "function",
        "z": "8ab7019d.c84638",
        "name": "Dies ist ein Test",
        "func": "msg.payload =\"{\\\"val\\\":\\\"Dies ist ein
Test\\\"}\"+\"\\n\";\nreturn msg;",
        "outputs": 1,
        "noerr": 0,
        "x": 380,
        "y": 600,
        "wires": [
            [
                "971773dd.ca7ff8",
                "7f3e8337.75ad1c"
            ]
        ]
    },
    {
        "id": "971773dd.ca7ff8",
        "type": "function",
        "z": "8ab7019d.c84638",
        "name": "Questo è un test",
        "func": "msg.payload =\"{\\\"val\\\":\\\"Questo è un
test\\\"}\"+\"\\n\";\nreturn msg;",
        "outputs": 1,
        "noerr": 0,
```

```
        "x": 390,
        "y": 680,
        "wires": [
            [
                "295af871.e6eaa8"
            ]
        ]
    },
    {
        "id": "e3b15c0e.873448",
        "type": "tcp request",
        "z": "8ab7019d.c84638",
        "server": "192.168.1.200",
        "port": "5963",
        "out": "immed",
        "splitc": " ",
        "name": "",
        "x": 730,
        "y": 120,
        "wires": [
            []
        ]
    },
    {
        "id": "c3303edc.d64f88",
        "type": "tcp request",
        "z": "8ab7019d.c84638",
        "server": "192.168.1.200",
        "port": "5963",
        "out": "immed",
        "splitc": " ",
        "name": "",
        "x": 730,
        "y": 200,
        "wires": [
            []
        ]
    },
    {
        "id": "91e8110e.90a87",
        "type": "tcp request",
        "z": "8ab7019d.c84638",
        "server": "192.168.1.200",
        "port": "5963",
        "out": "immed",
        "splitc": " ",
        "name": "",
        "x": 730,
        "y": 280,
        "wires": [
            []
```

```
        ]
    },
    {
        "id": "144cdcaf.b0921b",
        "type": "tcp request",
        "z": "8ab7019d.c84638",
        "server": "192.168.1.200",
        "port": "5963",
        "out": "immed",
        "splitc": " ",
        "name": "",
        "x": 730,
        "y": 360,
        "wires": [
            []
        ]
    },
    {
        "id": "c23b1a7.45ab968",
        "type": "tcp request",
        "z": "8ab7019d.c84638",
        "server": "192.168.1.200",
        "port": "5963",
        "out": "immed",
        "splitc": " ",
        "name": "",
        "x": 730,
        "y": 440,
        "wires": [
            []
        ]
    },
    {
        "id": "a2b10368.ce42e8",
        "type": "tcp request",
        "z": "8ab7019d.c84638",
        "server": "192.168.1.200",
        "port": "5963",
        "out": "immed",
        "splitc": " ",
        "name": "",
        "x": 730,
        "y": 520,
        "wires": [
            []
        ]
    },
    {
        "id": "7f3e8337.75ad1c",
        "type": "tcp request",
        "z": "8ab7019d.c84638",
```

```
        "server": "192.168.1.200",
        "port": "5963",
        "out": "immed",
        "splitc": " ",
        "name": "",
        "x": 730,
        "y": 600,
        "wires": [
            []
        ]
    },
    {
        "id": "295af871.e6eaa8",
        "type": "tcp request",
        "z": "8ab7019d.c84638",
        "server": "192.168.1.200",
        "port": "5963",
        "out": "immed",
        "splitc": " ",
        "name": "",
        "x": 730,
        "y": 680,
        "wires": [
            []
        ]
    }
]
```


TCP Requestノードで8つのデータを順番に送信。でも1つ目で止まる。

サービスがRequestに対してのResponseを返さないため、待ち状態。

フロ□3.json


```
[
    {
        "id": "bffbf4c1.f617e8",
        "type": "comment",
        "z": "e9f07343.baab8",
        "name": "TCP REQUESTノード利用(直列接続)",
        "info": "",
        "x": 850,
        "y": 60,
        "wires": []
    },
    {
        "id": "8ac57920.f34d8",
        "type": "inject",
```

```
        "z": "e9f07343.baab8",
        "name": "",
        "topic": "",
        "payload": "",
        "payloadType": "date",
        "repeat": "",
        "crontab": "",
        "once": false,
        "onceDelay": 0.1,
        "x": 200,
        "y": 100,
        "wires": [
            [
                "4df22afa.c7f714"
            ]
        ]
    },
    {
        "id": "4df22afa.c7f714",
        "type": "function",
        "z": "e9f07343.baab8",
        "name": "this is test",
        "func": "msg.payload =\"{\\\"val\\\":\\\"this is
test\\\"}\"+\"\\n\";\nreturn msg;",
        "outputs": 1,
        "noerr": 0,
        "x": 440,
        "y": 100,
        "wires": [
            [
                "921a5477.fcd"
            ]
        ]
    },
    {
        "id": "6e925dcc.e38c7c",
        "type": "function",
        "z": "e9f07343.baab8",
        "name": "これはテストです",
        "func": "msg.payload =\"{\\\"val\\\":\\\"これはテストで
す\\\"}\"+\"\\n\";\nreturn msg;",
        "outputs": 1,
        "noerr": 0,
        "x": 470,
        "y": 180,
        "wires": [
            [
                "3e87f255.e0b5f6"
            ]
        ]
    },
```

```
    {
        "id": "9e5ded52.901b3",
        "type": "function",
        "z": "e9f07343.baab8",
        "name": "这是一个测试",
        "func": "msg.payload =\"{\\\"val\\\":\\\"这是一个测
试\\\"}\"+\"\\n\";\nreturn msg;",
        "outputs": 1,
        "noerr": 0,
        "x": 460,
        "y": 260,
        "wires": [
            [
                "f8b1405a.99ed1"
            ]
        ]
    },
    {
        "id": "7adab55.b54e24c",
        "type": "function",
        "z": "e9f07343.baab8",
        "name": "Это тест",
        "func": "msg.payload =\"{\\\"val\\\":\\\"Это
тест\\\"}\"+\"\\n\";\nreturn msg;",
        "outputs": 1,
        "noerr": 0,
        "x": 440,
        "y": 340,
        "wires": [
            [
                "a06ab5d7.b53d98"
            ]
        ]
    },
    {
        "id": "d088927a.5bf738",
        "type": "function",
        "z": "e9f07343.baab8",
        "name": "Ceci est un test",
        "func": "msg.payload =\"{\\\"val\\\":\\\"Ceci est un
test\\\"}\"+\"\\n\";\nreturn msg;",
        "outputs": 1,
        "noerr": 0,
        "x": 460,
        "y": 420,
        "wires": [
            [
                "38d1ce8f.fdac82"
            ]
        ]
    },
```

```
    {
        "id": "9f041814.05269",
        "type": "function",
        "z": "e9f07343.baab8",
        "name": "Ini adalah ujian",
        "func": "msg.payload =\"{\\\"val\\\":\\\"Ini adalah
ujian\\\"}\"+\"\\n\";\nreturn msg;",
        "outputs": 1,
        "noerr": 0,
        "x": 460,
        "y": 500,
        "wires": [
            [
                "ac6608c1.cff498"
            ]
        ]
    },
    {
        "id": "9314bfb5.3d616",
        "type": "function",
        "z": "e9f07343.baab8",
        "name": "Dies ist ein Test",
        "func": "msg.payload =\"{\\\"val\\\":\\\"Dies ist ein
Test\\\"}\"+\"\\n\";\nreturn msg;",
        "outputs": 1,
        "noerr": 0,
        "x": 460,
        "y": 580,
        "wires": [
            [
                "bcffe6c7.f6ab1"
            ]
        ]
    },
    {
        "id": "c8a4252.1db6b58",
        "type": "function",
        "z": "e9f07343.baab8",
        "name": "Questo è un test",
        "func": "msg.payload =\"{\\\"val\\\":\\\"Questo è un
test\\\"}\"+\"\\n\";\nreturn msg;",
        "outputs": 1,
        "noerr": 0,
        "x": 470,
        "y": 660,
        "wires": [
            [
                "30d3ae5b.785892"
            ]
        ]
    },
```

```
    {
        "id": "921a5477.fcd",
        "type": "tcp request",
        "z": "e9f07343.baab8",
        "server": "192.168.1.200",
        "port": "5963",
        "out": "immed",
        "splitc": " ",
        "name": "",
        "x": 810,
        "y": 100,
        "wires": [
            [
                "6e925dcc.e38c7c"
            ]
        ]
    },
    {
        "id": "3e87f255.e0b5f6",
        "type": "tcp request",
        "z": "e9f07343.baab8",
        "server": "192.168.1.200",
        "port": "5963",
        "out": "immed",
        "splitc": " ",
        "name": "",
        "x": 810,
        "y": 180,
        "wires": [
            [
                "9e5ded52.901b3"
            ]
        ]
    },
    {
        "id": "f8b1405a.99ed1",
        "type": "tcp request",
        "z": "e9f07343.baab8",
        "server": "192.168.1.200",
        "port": "5963",
        "out": "immed",
        "splitc": " ",
        "name": "",
        "x": 810,
        "y": 260,
        "wires": [
            [
                "7adab55.b54e24c"
            ]
        ]
    },
```

```
    {
        "id": "a06ab5d7.b53d98",
        "type": "tcp request",
        "z": "e9f07343.baab8",
        "server": "192.168.1.200",
        "port": "5963",
        "out": "immed",
        "splitc": " ",
        "name": "",
        "x": 810,
        "y": 340,
        "wires": [
            [
                "d088927a.5bf738"
            ]
        ]
    },
    {
        "id": "38d1ce8f.fdac82",
        "type": "tcp request",
        "z": "e9f07343.baab8",
        "server": "192.168.1.200",
        "port": "5963",
        "out": "immed",
        "splitc": " ",
        "name": "",
        "x": 810,
        "y": 420,
        "wires": [
            [
                "9f041814.05269"
            ]
        ]
    },
    {
        "id": "ac6608c1.cff498",
        "type": "tcp request",
        "z": "e9f07343.baab8",
        "server": "192.168.1.200",
        "port": "5963",
        "out": "immed",
        "splitc": " ",
        "name": "",
        "x": 810,
        "y": 500,
        "wires": [
            [
                "9314bfb5.3d616"
            ]
        ]
    },
```

```
    {
        "id": "bcffe6c7.f6ab1",
        "type": "tcp request",
        "z": "e9f07343.baab8",
        "server": "192.168.1.200",
        "port": "5963",
        "out": "immed",
        "splitc": " ",
        "name": "",
        "x": 810,
        "y": 580,
        "wires": [
            [
                "c8a4252.1db6b58"
            ]
        ]
    },
    {
        "id": "30d3ae5b.785892",
        "type": "tcp request",
        "z": "e9f07343.baab8",
        "server": "192.168.1.200",
        "port": "5963",
        "out": "immed",
        "splitc": " ",
        "name": "",
        "x": 810,
        "y": 660,
        "wires": [
            []
        ]
    }
]
```


TCP Requestノードでプロキシーサーバーを経由して8つのデータを送信。送信されるデータの順番はフロー通り。

フロ□4.json


```
[
    {
        "id": "a7b7b3ad.eb1b4",
        "type": "comment",
        "z": "8a0e0867.0732c8",
        "name": "TCP REQUESTノード利用(ダミーレスポンスを返すプロキシー経由)",
        "info": "",
        "x": 880,
        "y": 60,
```

```
        "wires": []
    },
    {
        "id": "580cfe92.e80148",
        "type": "inject",
        "z": "8a0e0867.0732c8",
        "name": "",
        "topic": "",
        "payload": "",
        "payloadType": "date",
        "repeat": "",
        "crontab": "",
        "once": false,
        "onceDelay": 0.1,
        "x": 140,
        "y": 100,
        "wires": [
            [
                "834390a.a6d8cf"
            ]
        ]
    },
    {
        "id": "834390a.a6d8cf",
        "type": "function",
        "z": "8a0e0867.0732c8",
        "name": "this is test",
        "func": "msg.payload =\"{\\\"val\\\":\\\"this is
test\\\"}\"+\"\\n\";\nreturn msg;",
        "outputs": 1,
        "noerr": 0,
        "x": 380,
        "y": 100,
        "wires": [
            [
                "655bfbc8.0a2c4c"
            ]
        ]
    },
    {
        "id": "38db1dcf.ab805a",
        "type": "function",
        "z": "8a0e0867.0732c8",
        "name": "これはテストです",
        "func": "msg.payload =\"{\\\"val\\\":\\\"これはテストで
す\\\"}\"+\"\\n\";\nreturn msg;",
        "outputs": 1,
        "noerr": 0,
        "x": 410,
        "y": 180,
        "wires": [
```

```
            [
                "1457ceb5.a21b31"
            ]
        ]
    },
    {
        "id": "9d1b5c21.7e30e",
        "type": "function",
        "z": "8a0e0867.0732c8",
        "name": "这是一个测试",
        "func": "msg.payload =\"{\\\"val\\\":\\\"这是一个测
试\\\"}\"+\"\\n\";\nreturn msg;",
        "outputs": 1,
        "noerr": 0,
        "x": 400,
        "y": 260,
        "wires": [
            [
                "22a9a283.66860e"
            ]
        ]
    },
    {
        "id": "6b10276e.d73088",
        "type": "function",
        "z": "8a0e0867.0732c8",
        "name": "Это тест",
        "func": "msg.payload =\"{\\\"val\\\":\\\"Это
тест\\\"}\"+\"\\n\";\nreturn msg;",
        "outputs": 1,
        "noerr": 0,
        "x": 380,
        "y": 340,
        "wires": [
            [
                "f6293756.b6d7a"
            ]
        ]
    },
    {
        "id": "5e8bcb1f.8be8d4",
        "type": "function",
        "z": "8a0e0867.0732c8",
        "name": "Ceci est un test",
        "func": "msg.payload =\"{\\\"val\\\":\\\"Ceci est un
test\\\"}\"+\"\\n\";\nreturn msg;",
        "outputs": 1,
        "noerr": 0,
        "x": 400,
        "y": 420,
        "wires": [
```

```
            [
                "1606538d.eeeb0c"
            ]
        ]
    },
    {
        "id": "3ecc1de8.b27e0a",
        "type": "function",
        "z": "8a0e0867.0732c8",
        "name": "Ini adalah ujian",
        "func": "msg.payload =\"{\\\"val\\\":\\\"Ini adalah
ujian\\\"}\"+\"\\n\";\nreturn msg;",
        "outputs": 1,
        "noerr": 0,
        "x": 400,
        "y": 500,
        "wires": [
            [
                "8ebe5621.24ff6"
            ]
        ]
    },
    {
        "id": "54230cb3.3d45d4",
        "type": "function",
        "z": "8a0e0867.0732c8",
        "name": "Dies ist ein Test",
        "func": "msg.payload =\"{\\\"val\\\":\\\"Dies ist ein
Test\\\"}\"+\"\\n\";\nreturn msg;",
        "outputs": 1,
        "noerr": 0,
        "x": 400,
        "y": 580,
        "wires": [
            [
                "d237480c.b8cc4"
            ]
        ]
    },
    {
        "id": "177f4f75.265211",
        "type": "function",
        "z": "8a0e0867.0732c8",
        "name": "Questo è un test",
        "func": "msg.payload =\"{\\\"val\\\":\\\"Questo è un
test\\\"}\"+\"\\n\";\nreturn msg;",
        "outputs": 1,
        "noerr": 0,
        "x": 410,
        "y": 660,
        "wires": [
```

```
            [
                "55f49c18.d1e024"
            ]
        ]
    },
    {
        "id": "655bfbc8.0a2c4c",
        "type": "tcp request",
        "z": "8a0e0867.0732c8",
        "server": "192.168.1.200",
        "port": "15963",
        "out": "time",
        "splitc": "0",
        "name": "",
        "x": 750,
        "y": 100,
        "wires": [
            [
                "38db1dcf.ab805a"
            ]
        ]
    },
    {
        "id": "1457ceb5.a21b31",
        "type": "tcp request",
        "z": "8a0e0867.0732c8",
        "server": "192.168.1.200",
        "port": "15963",
        "out": "time",
        "splitc": "0",
        "name": "",
        "x": 750,
        "y": 180,
        "wires": [
            [
                "9d1b5c21.7e30e"
            ]
        ]
    },
    {
        "id": "22a9a283.66860e",
        "type": "tcp request",
        "z": "8a0e0867.0732c8",
        "server": "192.168.1.200",
        "port": "15963",
        "out": "time",
        "splitc": "0",
        "name": "",
        "x": 750,
        "y": 260,
        "wires": [
```

```
            [
                "6b10276e.d73088"
            ]
        ]
    },
    {
        "id": "f6293756.b6d7a",
        "type": "tcp request",
        "z": "8a0e0867.0732c8",
        "server": "192.168.1.200",
        "port": "15963",
        "out": "time",
        "splitc": "0",
        "name": "",
        "x": 750,
        "y": 340,
        "wires": [
            [
                "5e8bcb1f.8be8d4"
            ]
        ]
    },
    {
        "id": "1606538d.eeeb0c",
        "type": "tcp request",
        "z": "8a0e0867.0732c8",
        "server": "192.168.1.200",
        "port": "15963",
        "out": "time",
        "splitc": "0",
        "name": "",
        "x": 750,
        "y": 420,
        "wires": [
            [
                "3ecc1de8.b27e0a"
            ]
        ]
    },
    {
        "id": "8ebe5621.24ff6",
        "type": "tcp request",
        "z": "8a0e0867.0732c8",
        "server": "192.168.1.200",
        "port": "15963",
        "out": "time",
        "splitc": "0",
        "name": "",
        "x": 750,
        "y": 500,
        "wires": [
```

```
            [
                "54230cb3.3d45d4"
            ]
        ]
    },
    {
        "id": "d237480c.b8cc4",
        "type": "tcp request",
        "z": "8a0e0867.0732c8",
        "server": "192.168.1.200",
        "port": "15963",
        "out": "time",
        "splitc": "0",
        "name": "",
        "x": 750,
        "y": 580,
        "wires": [
            [
                "177f4f75.265211"
            ]
        ]
    },
    {
        "id": "55f49c18.d1e024",
        "type": "tcp request",
        "z": "8a0e0867.0732c8",
        "server": "192.168.1.200",
        "port": "15963",
        "out": "time",
        "splitc": "0",
        "name": "",
        "x": 750,
        "y": 660,
        "wires": [
            []
        ]
    }
]
```


Windows, .NET, Csharp, Node-RED