Our Little Multiuser Dungeon (OLMUD)

Table of Contents

1 SrrTrains is Dead, Long Live SrrTrains!!!

Dear Reader,

Well, I had invested a lot of sweat into the SrrTrains project (between the years 2008 and 2019), in the end it has been frozen somehow.

Nevertheless, I can still remember a lot of the findings that I was granted during that project, so now it makes totally sense, when I try to write down my thoughts about John's projects.

John is currently doing his JSONverse, which is – as far as I understood so far – a WebGL based, X3D based experimental implementation of some Network Sensor Node (NSN) and its application within one (or more than one) exemplary multiuser scene(s).

You can find the JSONverse at following link: https://github.com/coderextreme/JSONverse.

2 Multiplexing

Before we dive a little bit into the needed structure of such a multiuser scene (what I called SMS, some time ago), we need to talk about the technical provisions that are provided by the Internet.

Often, we refer to the Internet as a "Network of Computers", but actually it is a "Network of Processes", as I will elaborate shortly.

The Internet addresses, the @IP, are sometimes referred to as revealing your identity.

Well, that's not completely true. Actually, an @IP reveals the identity of a machine. Only, if it is somehow proven that you are using (have been using) this machine, then the @IP reveals your identity, too.

Second, a machine, sometimes referred to as a host, hosts one ot more processes. Now, it's the processes that actually run the software of your computer. We can think of the processes as of little busy dwarves that are inhabiting the machine.

Now, if one of these dwarves, who inhabits one machine, wants to send a letter to a dwarf, who inhabits another machine, he needs to ADDRESS the dwarf.

Like in real mail, where the address consists of NAME + ADDRESS of a person, when sending an Internet DATAGRAM to another dwarf, the address consists of PORT + @IP.

Therefore, the first dwarf, let's call him "Dwarf A" creates a connection to "Dwarf B", and then the datagram is sent via the connection.

Hence, such a connection is identified by a quadruple:

Network Connection is identified by:

- @IP A
- PORT A
- @IP B
- PORT B

as depicted in following Abbildung 2.1.

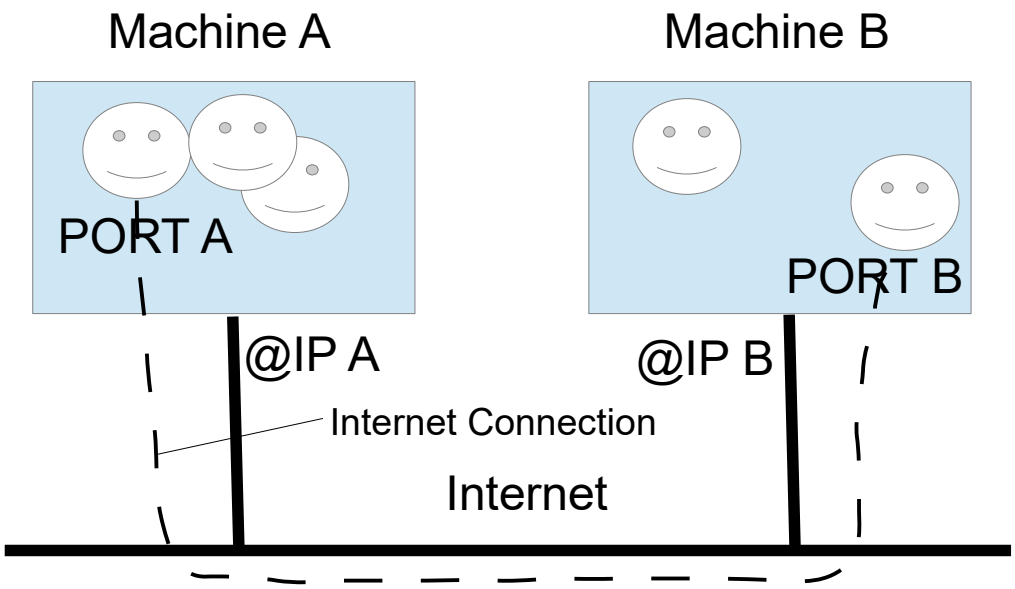

Abbildung 2.1: Basic Multiplexing in the Internet

So, this figure depicts the communication among processes in the Internet, where always two processes are connected by an Internet Connection.

3 Additional Multiplexing (Sub-Multiplexing)

However, when we talk about X3D scenes, then the processes – i.e. what I called the "scene instances" – do not provide sufficient granularity for the network communication.

When an X3D scene instance communicates with an X3D Collaboration Server – or even directly with another X3D scene instance –, then we have to consider **the fact that the scenes will consist of what-i-call "objects"** (in the SrrTrains project, I called such objects the "facilities").

So, although there is only one Internet Connection for each scene instance (i.e. for each human user) between each X3D Scene Instance and the X3D Collaboration Server, **we have to multiplex traffic for several (or many) objects over each of those connections**.

Additionally, **we have some session level control messages (e.g. for chat)**.

In John's NSN implementation, the multiplexing is realized by what is called "namespaces" of the socket.io software. There we use

- The default namespace for session level control: /
- One namespace for each object of the scene: /obj1, /obj2, /obj3, ...

This is depicted in the following figure.

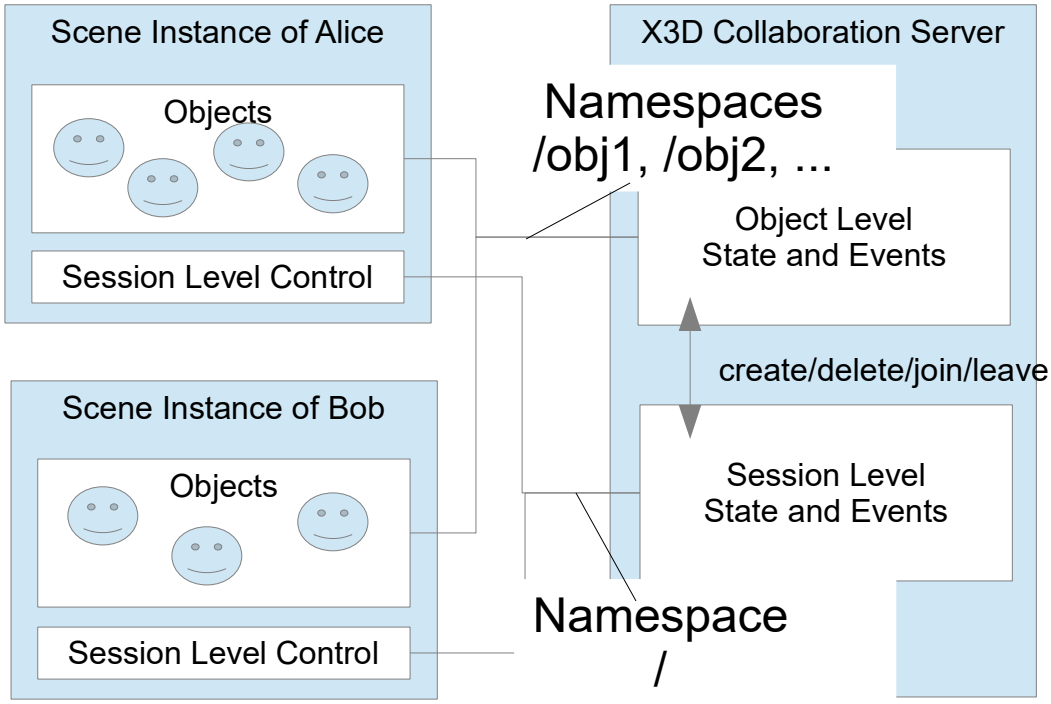

Abbildung 3.1: sub-multiplexing per namespace for objects and session level

Now, the socket.io library needs a global identifier, additionally the server can handle more than one session and each session is provided by a user of the server.

Hence the namespaces look actually like the following:

- **Session Level Namespace: /socket.io/<user>/<sessionid>/**
- **Object Level Namespace: /socket.io/<user>/<sessionid>/<objId>**

4 The Objects of a Multiuser Scene

Tbd.

5 Roles

Tbd.

5.1 Scene Author

Tbd.

5.2 Model Author

Tbd.

5.3 Session Operator

Tbd.

5.4 Gamer

Tbd.

6 Runtime Procedures

Tbd.

6.1 Bootstrapping the Scene / Scene Description

Tbd.

6.2 Login and Avatar Selection

Tbd.

6.3 Event Distribution

Tbd.

6.4 Shared State

Tbd.

6.5 Server Side Calculations

Tbd.

6.6 Flexible Client Side Calculations by the Author

Tbd.