Titan Tricks

sACN Unicast

The idea for this is from a post in the Facebook group:

Has anyone successfully controlled the Mission Ballroom disco ball with an Avo? Need to unicast sACN to the nodes but can’t find a way to do that.

sACN - or Streaming ACN - is usually used as multicast protocol: the console sends it to a network switch with a special header so that the switch can define groups, one per universe. Nodes and fixtures in turn announce to the switch which universe(s) they need – they **subscribe** to the respective group. The switch then makes sure that each node/fixture receives the data they want, and are not receiving data they do not want. This reduces network traffic and computing load. This mechanism is called **Multicast**.

There is a fallback mechanism in place if some details are missing. In that case the switch defaults to forwarding all sACN data to all nodes/fixtures which requires much more bandwith and strains the network interfaces but makes sure that all participants have a chance to get their data. This is called **Broadcast**.

However it looks like there are fixtures in the market which do not support this mechanism but require **Unicast**: they only work with data explicitely sent to them.

(Please excuse this very simplistic explanation. I greatly encourage you to read more on this, and there are plenty of websites dedicated to introducing you to network computing. But for the purpose of this article this might suffice.)

Avolites Titan at the moment (Titan v15, 2022) sends sACN only as Multicast (automatically falling back to Broadcast if no IGMP capable switch is present). While it has been requested to implement sACN unicast into Titan a solution is required to circumvent this for the time being.

MIDIMonster

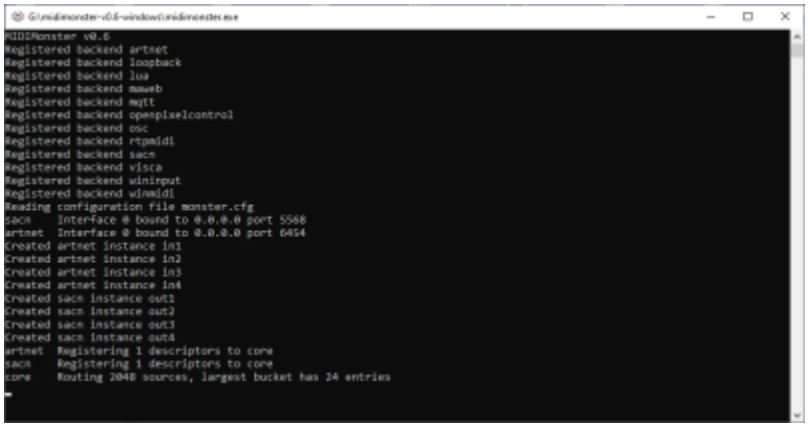
… and along comes MIDIMonster. Available at https://midimonster.net this is something like the Swiss Army Knife for lighting protocols. It is a lightweight console application, available for Windows, Linux and OSX, where you simply write a configuration as a text file and then start the program. This comes at the extra benefit that you don't need drivers, nothing being installed, and this can even be run from a USB thumbdrive.

If you download MIDIMonster from the above link, start it, and get this message

```
Failed to load plugin backends\lua.dll, check that all supporting libraries are present…
```

then you find the solution here, download LUA binaries here, unzip it, and copy the file `lua53.dll` into MIDIMonster's main folder.

In order to run it you simply doubliclick `midimonster.exe` in which case it loads its settings from the file `monster.cfg`. Subsequently you can either edit this directly with a text editor, or you write your own file with a different name and drop it onto midimonster.exe in order to start it with your configuration.

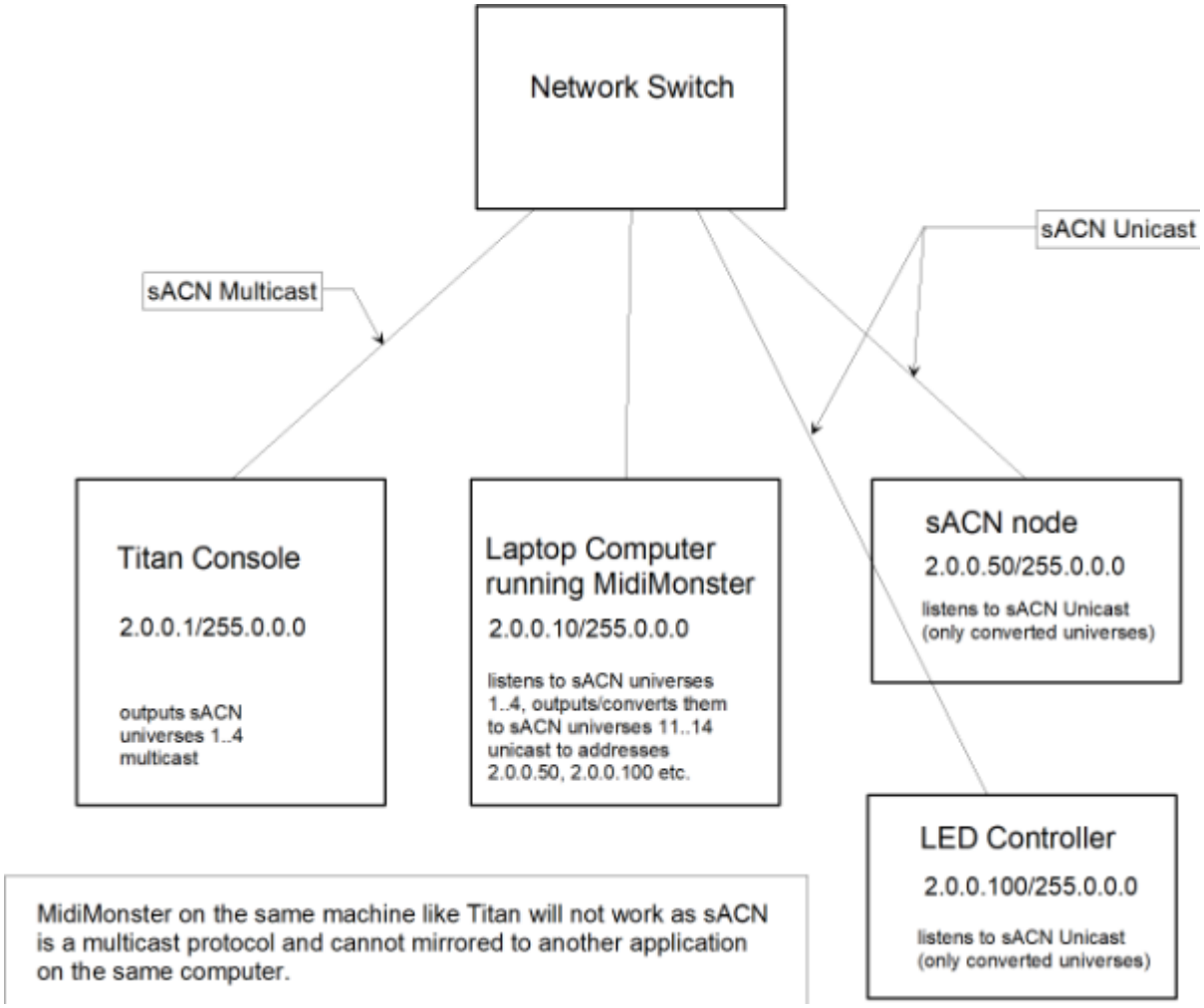


Depending on the situation I successfully tested two scenarios: converting multicast sACN to unicast sACN (sending the data out on different universes) which requires a separate computer, and converting broadcast Art-Net to unicast sACN which may as well run directly on a console. I tried both scenarios with a bunch of computers, checking sACN with sACNView and - to be sure - verifying the sent data with Wireshark.

---

sACN to sACN, multicast to unicast

Network sctructure:



This was the original intention as everything was already prepared on sACN. All it needs is another computer hooked up to the same network, MIDIMonster, and a suitable configuration like this:

monster.cfg

```
; This configuration maps sACN Multicast universe 1 to sACN Unicast universe 11
;   for the receiver (node) IP address 2.0.0.100.

[backend sacn]
bind = 0.0.0.0 5568
detect = verbose

[sacn in1]
universe = 1

[sacn out1]
universe = 11
priority = 100
destination = 2.0.0.100
unicast = 1

[map]
```

```
in1.{1..512} > out1.{1..512}
```

An example file for mapping 4 universes

is attached here

.

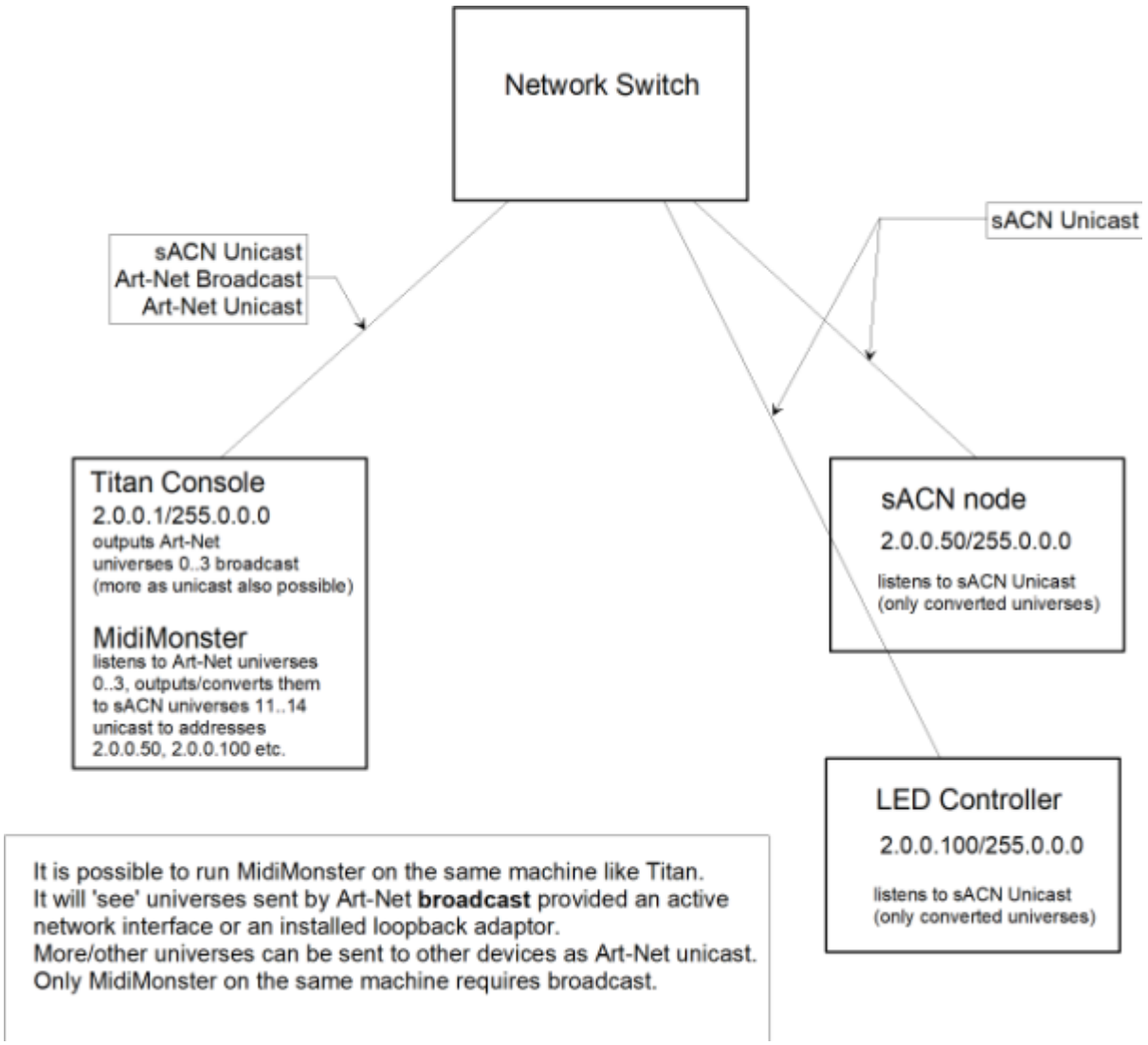
All you need to do is adjust this to your needs (in and out universes, target network addresses, maybe more mappings etc.), save this as monster.cfg (or drop this on midimonster.exe, or call it like this: `midimonster <path/to/configfile.cfg>` (personally I prefer saving this as monster.cfg as I can simply doubleclick midimonster.exe).

Hints:

- midimonster takes very low processing power (at least in my tests with 4 universes converted)
- at least in my tests it was not possible to make this work with MIDIMonster directly on the console: it does run but doesn't output as required - the nodes see the universes but there is no data. I believe it is not possible to have two applications on the same machine which both attempt to send to the same port, and one also listens to the other.
- make sure that all network addresses are correct, and that firewalls are off or have the required rules set.

Art-Net broadcast to sACN unicast

Network structure:



The question arose if the additional computer could be spared and if MIDIMonster could be run on the same machine Titan runs on. As explained in the section above this does **not** work with sACN - the external nodes see that MIDIMonster sends something but the values are always 0. I suspect this is due to some fancy network routing problem: two programs attempting to send on the same port and on the same machine, AND one program trying to listen to the other.

The solution to this dilemma is using Art-Net: we route Titan's DMX-lines to Art-Net broadcast which is sent to everyone in the network, even to another program on the same computer. Of course the MIDIMonster config needs to be slightly altered as now we have to listen to Art-Net. N.B. in Titan you can adjust per universe whether Art-Net should be sent as unicast (= distinct individual IP address) or boradcast (= IP address 255.255.255.255). There is no problem to send unicast to other nodes as well - only the lines for MIDIMonster need to be broadcast.

MIDIMonster config:

monster.cfg

```
; This configuration maps Art-Net Broadcast universe 0 to sACN Unicast
universe 1

[backend sacn]
bind = 0.0.0.0 5568
```

```
detect = verbose

[sacn in1]
universe = 1
[sacn in2]
universe = 2
[sacn in3]
universe = 3
[sacn in4]
universe = 4

[sacn out1]
universe = 11
priority = 100
destination = 2.0.0.100
unicast = 1
[sacn out2]
universe = 12
priority = 100
destination = 2.0.0.100
unicast = 1
[sacn out3]
universe = 13
priority = 100
destination = 2.0.0.100
unicast = 1
[sacn out4]
universe = 14
priority = 100
destination = 2.0.0.100
unicast = 1

[map]

in1.{1..512} > out1.{1..512}
in2.{1..512} > out2.{1..512}
in3.{1..512} > out3.{1..512}
in4.{1..512} > out4.{1..512}
```