A personal view on the future of lighting control

By Kurt Sterckx
28 September 2008

This article is a very personal view on how lighting control should work in the future. It is almost 18 years ago, when I was 18, that I discussed many of this with a friend of my, during the school play times. A lot has happened since then in the lighting industry. The RS485 protocol we where dreaming of has come, in the form of DMX, but with totally different properties then we had in mind. We where already spook about RDM features. For use it was very important that we could get information back from the fixtures so we could diagnose errors but also discover there functions. We realized very quickly that a RS485 link would be to slow for the features we wanted. We needed a faster network system, but where we lived nobody known the term Ethernet. We where still in the home computer era. This didn't prevent use from designing possible hardware and writing protocol documents. If we would have continued, maybe, we would already had a bidirectional lighting control standard for years, but my friend was more interested in sound and I became a High school student with a very limited hardware platform, called 'PC'.

What where the features we where after? For use it was obvious that we needed to control more then just the intensity. Without a connection to the world wide web we didn't known that there where already moving light fixtures, so we invented them. This meant that we needed to be able to control intensity, beam shape, color, position, and all the things we now see in the specifications. So each fixture type had a set of parameters, that could be different for each of them. Each fixture would have its own computer that controlled the motors and dimmers. All fixtures where connected, in a daisy chain topology (just like DMX). This was very important for use. It would mean that we could use the shorted cable. DMX has proven that this is a very good approach. This is the biggest disadvantage of baseT or baseF Ethernet. You always need a cable from the fixture to a hub (and a hub). This means a lot more cables that are longer. I find this one of most important problems for the future that looks Ethernet based.

If we don't find a high speed, bi-directional, easy to assemble and daisy chain connectable network system, we are undermining the future of lighting control. The new fixtures, like the ShowPix of High End Systems (needs 451 channels in extended protocol mode), have a lot of parameters that must be controllable. When DMX is used to control the ShowPix we end up with one DMX universe for each fixture, this means that we must connect each of them to a DMX to Ethernet gateway, giving us the exact same type of point-to-point cabling as baseT Ethernet. There are already a number of possible solutions for this. There is a system called EtherCAT, developed by BeckHoff, that can be used with almost any bus topology. Each device has at least two RJ45 connectors that are used to connect the device, but some of them have three. I would make it a must that each fixture has three connectors

(using the Neutrik Ethercon style connectors), so that you could connect then anyway that suite the needs. When you have a truss structure that is 50 meter longs and that contains a fixture each 5 meter, you could connected them one after the other in a daisy chain topology. If you now had a T, in the middle of the 50 meter, with 5 meter truss and a fixture on the end, you just use the third connector of one of the already connected fixtures to bring the control signal to the T fixture. Just the though that this is possible gives me the trills. It is proven technology, that allows for 100 meter cable between any two connectors, up to 65535 fixtures, 100BASE-TX physics and deterministic real time behavior. Before you think that I work for a company that sells this product, I want to tell you that I just found this on the internet. I have no experience with it. It looks however the Holy Grail where the lighting control world is waiting for. There are however at least two other possibilities. The first is wireless. This would make our life even simpler: no more data cables. Each fixture has a high speed radio link with the console. This could be a high speed unidirectional link from the console to the fixtures and a slower speed unidirectional link back from the fixtures. There are already systems available, but most them have only DMX/RDM in or out connections. There is no real Ethernet alike connection. You could however use a regular, very cheap, Wireless gateway. They are however not so rugged and tend to life only a short period. If technology allowed use to create a reliable wireless system, we only needed to connect the power cables. It would save use a lot of work and costs. This brings me to a third option, data transmission over the power lines. This would mean that we just needed to plug everything in the mains and we had a wired power and communication channel. There are already Ethernet products that use this technology. The Avolite mDMX product uses this principle to transfer DMX on the power line. In the home market there are power line Ethernet routers. I personally think that this system is only useful for slow speed communication with a limited number of fixtures. The very messy main cabling used in our industry will introduce a lot of high frequency problems, that are very difficult to detect and solve. The mains must also transfer the regular power, with high currents, that will influence the communication. However if it is possible to make it work reliable it would solve a lot of cabling problems. I think it could also be the cheapest and most rugged solution. The conclusion is that, even when new protocols are created that allow for the transport of a lot of data in a bidirectional way, if we don't find a easy to use and rugged physical transport medium, all the efforts are useless. It would have been better to first create a standard for this transport medium, then to create the protocol. I personally think that something like EtherCat has the best papers to be upgradeable to higher speed, be reliable and easy to find/solve problems. With the right tools you can also make your own cables when needed. This is also very important in environments where Murphy's Law is proven everyday.

Suppose that we find and agree on a transport medium that we use to transport a protocol. Than we must find ways to use this communication. A lot of companies already have created there own protocols and ways of using them. For me the product that makes the best use of the possibilities is the grandMA. MA Lighting has created a product range that uses bidirectional, Ethernet based, communication to synchronize different devices. When you add a fixture into the patch, it will show up in the grandMA3D and you can control it immediately. It also works the other way. When you have multiple console that are used to control the same show, they all will adjusted there patch to include the new fixture. You can extend the number of channels by using so called NSP devices. This are a kind of black box consoles that control up to 4096 parameters. You can use any product with console functionality, even the grandMA onPC, to control the NSP devices. The really fun part however starts when you use grandMA Video software/hardware. From your console you can browse the files, selected images or video's, setup parameters and perform any function to

control the video output. The grandMA also uses time synchronization so that all DMX frames are transmitted one the same moment. This is very important for LED walls. The grandMA range is a really integrated system. The only thing that is missing are fixtures with the NSP and visualizer data build in.

Now we get to how I see the future of lighting:

1. Each fixture should contain a computer that controls the complete fixture. This is already the case for many automatic lights (some have external dimmers)
2. Each fixture should have meaningful parameters, like tilt angle in degrees, smoke output in m^3, moving speed in m/s, that can be set/read by the console.
3. The fixtures should be discoverable. Once the fixture is hang and plugged in, the console would be notice and add the fixture to it “patch” (patch is no longer the good word, I call it “Real fixture list”).
4. The fixtures parameters should also be discoverable. When a fixture is found the console immediately gets a list of the parameters. As a operator you can immediately read the current values and change them: Plug and Control.
5. Any visualization data needed to represent the parameters can also be supplied by the fixtures, this can be gobo bitmaps, color information, video clips.
6. When there are special controls needed to adjust the parameters value, this could be supplied by the fixture, as scripts that run on the console. This is a bit like applets used in web browsers. For example for the ShowPIX fixture of High End Systems this would allow you to see the LEDs matrix, that has a very specific layout, and adjust each LED individually.
7. The adjusted parameters should be stored in the console so it could reply them, or in the fixture itself(for the more expensive once). In the latter case the console just send a message to the fixture to indicate that it must reply a cue. This also means the times must be synchronized.
8. All the fixture data must be available without having the real fixtures. This is needed so that the fixture can be simulated. This is very important because it must be possible to preparing the light show before the rig is present. This also means it must be very easy to exchange virtual fixtures for real fixtures (for example when you connect the console with the virtual prepared show to the real lighting rig, it will add each fixture it find to the real fixture list. With the ‘locate features’ functions of the console(that uses values supplied by the fixture) you can see which fixture correspond with a item in the list and then you could drag that item onto the virtual fixture. This would replace the virtual one with the real one. From the moment the rig is back disconnected, the virtual fixtures are re-used !!!)
9. It must be possible to control any equipment, not just fixtures. This means that some devices will not have a intensity parameter. Almost all consoles currently available use the assumption that there is a intensity parameter. I advise you to try to control a smoke machine with the parameters smoke output and smoke density on a console. With any equipment I also means sound, video, hoists, some robot that runs over the stage, you name it.
10. It must be possible to use multiple console to program and run the same show. A operator on stage could control the running robot, a operator in the front of house the

stage lights and another operator at the same location the follow spots and audience lights.

The main shift that must be made in the heads of the people working with lighting control is the fact that there will no longer be a DMX patch. Almost all lighting console currently available use the DMX patch system. This is used to select fixtures from a library and connect there attributes with a DMX address in a DMX universe. Most lighting protocols, like RDM, ArtNet, CIPT, ESP, (I also think grandMA's protocol but there is no information public available), ACN BSR1.31 are based on the DMX principles. There is only one lighting protocol I known of that uses other principles: ACN.

It is also important to notice that all information needed by the console is coming from the fixture itself. Currently when you patch a fixture you select it from a library that is created by the console manufacture. When you add a fixture in a visualization program, then the 3D model is created by the software house that developed the program. This is a little bit strange. The people that create fixtures have all the knowledge of them. They have bitmaps of the gobo's, they have measurements of the optics, the movement speeds, they known, or they should known, there fixture inside out. It is however somebody else, that in most cases has only access to a limited documentation set, that must create a 3D model or a fixture library item. Why can't the manufacture of fixtures add all this information to there fixtures. There are enough open formats, even for dynamic 3D models, that can be used to store this information. Then you only need a protocol, like TFTP, to get the information out of the fixture. The cost of memory that can store this information decreases each day. Most information will be fixed but some information must be easy changeable. For example when you replace a gobo with a custom one, you could get the bitmap from the gobo supplier. You place this on a USB stick. When you physically change the gobo in the fixture, you also insert the stick and change it virtually. If we could find a memory technology that can handle the large temperatures inside a fixture, we could even add this memory to the gobo itself and automatic read it out (or maybe just take a picture of the gobo using some build in camera). All this means that more software must be written and tested and a more powerful computer must be present. However once the software is there it can be used for all the different fixture types, with only minor changes. The price of computing power decrease also each day, so in the long term the spots won't be a lot more expensive. The time needed to setup a rig would however decrease a lot. No more need to setup DMX addresses, no need to read the manual of the fixture trying to decode the DMX address space, no need for a Ethernet connection(or disk) to update the library. You just rent a fixture, connected it and you can control it.

The ACN protocol already addresses most of these principles. However this doesn't mean ACN is perfect. I will go over the10 points again with the ACN standard next to me, and try to find the features that I want.

1. ACN does not tell anything about the location of the computers used to control the fixture. The computer can be inside the fixture or there it can be a external computer that controls more then one fixture. The external computer can communicate with the fixtures through DMX/RDM or any other protocol. This can be seen as a protocol converter. Because of the large installed base of DMX equipment the converters will likely be the first ACN products that come to the marked. I personally also see this as a transition step. However I would like the gateways to be customizable so that they look to the controller as a real fixture. The gateway behaves as multiple fixtures.

2. ACN describes parameters in term of behaviors. This gives the data format used by the parameter (integer number, floating point number, string, …), the possible values (lower limit, upper limits, …) and the unit (m^3, m/s, …). In other word you can describe a parameter as the smoke output between 0 and 10m^3. The parameter set and behaviors are stored in a DDL file that can be loaded from the fixture using TFTP.
3. ACN uses a combination of various protocols to make a fixture discoverable. The main protocol used for this is SLP. Each discoverable item (called component) has a unique number. This makes sure you never have to set a DMX address. Each fixture also has some fixed and user changeable strings making it easier to identify the specific fixture (for example “third fixture from left on the fifth stage truss”).
4. Once the fixture is located the DDL file can be retrieved to discover the parameters. Depending on the powers of the console, it can then represent the parameters as just numeric input fields with the behaviors displayed as strings. The user must interpret them. A more powerful console can also create specific input controls, like a color picker, for behaviors it understand.
5. ACN has the possibility to add read only parameters that can represent visualization data, like gobo bitmaps. However currently the maximum size of a bitmap is very limited because a complete ACN packet can’t have more then 1500 bytes. There is no way to get only parts of a bitmap (so you can get it using multiple reads). It would however be more efficient to use a file transfer protocol like TFTP to get the images. This protocol must already be implemented to get the DDL file. The parameter can then just be a string with a file path that can be used to retrieve the bitmap. This could even be a fixed path stored in the DDL file (for example /gobo wheel1/gobo1.bmp will always return the bitmap of the first gobo, even when the user has placed a custom one). Most of these data needs to be read only once. This would also be more efficient for video clips and 3D models.
6. ACN doesn’t describe a scripting language that could be used to create applets on the console to set/show parameters. There are already many languages that could be used for this purpose, most of them with open source implementations. I would select a scripting language that can be compiled into virtual machine code format. This can then be executed by a virtual machine. This can protected the console from unstable scripts. It also means that the real software code can’t be easily read or changed. The console reads the compiled scripts and executes it in a virtual machine.
7. In ACN there can be parameters present to replay complete cues (for example by changing the value of the cue number). ACN however has no commands. You can make them by using numeric (when parameter is 0, this means stop, 1 is play, 2 is pause) or string(“STOP”, “PLAY”, “PAUSE”) parameters. ACN has currently no specification about standard parameters. This can lead to incompatibilities when each manufacture selects its own sets. ACN also has no real time synchronization. You can implement SNTP or NTP to synchronizes the time of devices. There are however no ways to tell “execute cue 4 on 14:53”. It is possible to do this with parameters, one for the cue number and one for the time, but ACN has no standard for this. This feature is however very important for time critical systems and the distribution of processing power.
8. All the data, like DDL, gobo images…, present in a fixture must also be available when the fixture can’t be connected. This means that the information must be downloadable from a storage medium or the internet. Some of these data must also be editable, like the gobo images files. This way the visualization software can simulate

the fixture. The tools to download and edit the files should be present in the visualization software and the console. With this you can preprogram a show without the need for the real fixtures. The ACN norm doesn't speak about this. It is even very unclear how a visualization program should get its data. The software that is currently available spy in some form on the DMX data and use that information to simulate the fixtures. In ACN however the controller can direct its output directly to a device, making it very difficult to spy on it. There is a way to get events from a fixture that holds the value of changed parameters. This means that the controller first send changes to the device and that the devices send them back to the visualization program. It must also be possible for the console to send the same data directly to the program, when it creates a visual ghost device. The best solution should be that the visualization programs announces it devices as ghost devices of the real fixtures. The console could then connected this virtual fixture to the real fixture and send them both the same parameter values. When the real fixture is disconnected, the virtual will still react, and also the other way round.

9. ACN can be used to control any type of equipment. It doesn't use the "has intensity" way of thinking.
10. ACN can be used to control fixtures using multiple consoles. The consoles will just connect to the device and send there parameter values. There is no priority, nor security system (or I could not find it). With any console you can control any fixture. There are also no features to synchronize the consoles show files.

There are continuously adding extra features to ACN but currently it can't handle all the features I want and maybe it never will.

To get ACN started the first thing that must be created are consoles and protocol converters so we can use it to control the current available equipment. However when I first read the RDM and ACN standard documents I was amazed by the fact that the two standards don't speak about each other. This is rather bizarre. It seems that two separated groups have been working on them. If you read the names of the people that worked on them, you will find however that many of them worked on both. I also could not find any clues about how the two standard should interact with each other. Currently there are drafts in revision that addresses these issues. There is the BSR E1.32 Lightweight streaming protocol for transport of DMX512 using ACN. It has not really something to do with ACN. It is just a ArtNet alike way to transfer DMX data. We already have ArtNet, so there is no need for this. I am personal not happy with this standard because it means that a console must implement both the old DMX way of controlling parameters as the new ACN way. It would make everything even more complex. I think that some manufactures just want to stick a ACN sticker on there equipment for marketing reasons. A better solution could have been based on BSR E1.30-4-200x EPI 26 Device Description Language (DDL) Extensions for DMX and E1.31 devices. (I would add ArtNet devices). It describes additions to the DDL standard that are used to convert real ACN parameters to DMX. The document also adds some form of scripting to ACN. By using the DDL addition it is possible to describe conversions between DMX addresses and ACN parameters. This is the data needed by a protocol converter. It is however not obvious how the DDL must be created, maybe with a kind of patching software package and how it would get into the protocol converters. Each time you add/remove a fixture from the DMX output of the converter you must update the DLL file. You could create one DDL file for each fixture so that the converter represents many fixtures. This would be the most easy to manage. I find this

solution much better. You can have a real ACN only console and let the protocol converters do all the conversions.

I think the BSR E1.30-4-200x EPI26 is the missing piece to get ACN started. It has taken years before the ACN standard was finally released. However nearly two years after the release it seems that almost nobody has implemented it. This is because ACN needs a complete other approach to console behavior and manipulation (where is the patch ???). It also means that all the DMX/RDM based equipment must be connected through protocol converters (this can be part of the console, so there is still a DMX output and a patch). This adds extract costs and ACN converters will be a lot more expensive then ArtNet convertors, because the software is a lot more complex and there need to be a lot more processing power and memory. I currently see no real ACN implementation on the market (ETC claims that there ETCNet3 is based on ACN, but I don't see any features of ACN in the manuals).

I end this rather large personal view with some imaginary case studies. After reading them you may ask your self, where can I buy that stuff, just like I do ? The answer can be: "somewhere in the future", but I hope it will not be "never".

In the first case study you are a company that rent out lights and persons to control them, for parties and small concerts. You don't have time to do pre-production. You just sketch some lighting rig on the smallest paper you can find, load the flight cases and hit the road. Arrived at the location you unload the fixtures, see that you can't make what you had planned, so drop the paper in the bin and just start placing the fixtures. You just don't care about DMX addresses , there are none. You just connect each fixture to each other using Ethercon cables and make sure every fixture is also connected to the mains power supply. You have loaded a brand new ACN console, still in a "breakable" taped cardboard box. You unpack it and put it in the front of house and plug in the Ethercon. You apply power to the console and fixtures. After a minute the console has discovered all fixtures and shows a list. It has found 4 fixtures of type A and 6 of type B. You can se this based on the manufacture name and fixture types displayed by the console. You select one of the discovered fixture of type A and press the lamp on and locate button. One fixture starts turning and after a few seconds a light beam becomes visible. You drag the fixture out of the list onto the light plan window. The fixture is draw in a symbolic way just like on a lighting plan. You press the intensity zero button. Now you select the second fixture of type A and follow the same procedure. After 5 minutes you have a complete light plan with all 10 fixtures. And they are all working. You can now start programming some preset. You select all fixtures and use the controls to tilt them all 50 degree and pan them 30 degree. They all will throw a beam on the same angles. You can store it in the first position preset. After adding a few more position preset, you start programming some color presets. You use the color picker to select a color and store it in the preset. After programming the color presets, you use the gobo picker, that shows the real gobo pictures, read from the fixture, to select a gobo. The same gobo is present at both spots but at different locations. However because each gobo as a unique number (like a UID) the console can match them and setup the parameters correctly. You store this in a gobo preset. The second gobo that you want to use is only present in one fixture type, so the gobo picker will show the gobo only in the column for that fixture type. You store this in the second gobo preset. You continue until you created all preset. Now you start programming cues, some using the effects engine. You can also use the time line editor to draw curves for changes in parameter values. Very use full to create some complex effects. You can even go to lunch because you saved a lot of time setting up the rig. It is very important to notice that you didn't really patch, you didn't opened a fixture library. You could even program the presets without creating the light

plan. You still need to program the presets, cues and effects, but that is a creative process. The various pickers and wizards make that process easy.

In the second case study you grow so you now may handle large concerts. This means however that there are a lot more then 10 fixtures. The new lighting designer wants to use hundreds of them. The concert hall is however only available one day before the performance and that day there must be a rehearsal at 7pm. This means you must pre produce the show. You have the new ACN controllable visualization software. The lighting designer uses this to create his rig. The designer wants to use the new X fixture. You have ordered it but it is not arrived. By using the internet he can download all DDL files, 3D models, gobo bitmaps, … from the fixture manufacture website. He also ordered a special gobo with the logo of the group. It is not yet arrived but attach to the confirmation email of the gobo supplier is a bitmap and unique number(UID) file for the gobo. All this files are loaded in the visualization software. In this software the designer drags the new gobo to a fixture, gets a gobo wheel, drags the gobo to the wanted position and it will replace the previous gobo. He can also select multiple fixtures, right click and select gobo wheels from the menu. This will display the same window with the gobo wheel. Now the designer can drag the gobo to the window to replace the old gobo. Using the ACN parameters Set functions of the software the designer can also control the fixtures and check of the gobo is used. During the drawing of the rig the designer has connected the console. Each time he added a fixture, the console has updated its fixture list. When he removed a fixture, the fixture became red, but it is not deleted. He can now use the "Remove unconnected fixtures" button to deleted the removed fixtures. He starts programming the show using all the pickers and wizards that are available. The software was used to print light plans and equipment lists. The equipment was loaded and at 5am on the rehearsal day you started to build the rig. You didn't have pre-numbered the fixtures. Each time you hang a fixture you use the build in menu system, that runs on battery, to give the fixture the number present on the plan. Because the fixture uses a touch screen with digital ink, you can write the number with our hands and the software will recognize it. When the plan tells that the fixture needs the special gobo, you place the gobo in the fixture, go to the menu to select the gobo position, insert the USB stick with the bitmap and UID file, and press the update button. You connect all fixtures using the Ethercon cables. The three connectors make it very easy to perform this. For a small stage at the back of the concert hall you use fixtures with build in wireless receivers. You place the transmitter on the normal rig, above the heads of the people and in a direct line of sight. The transmitter ethernet connection is connected to some fixture.

At 13pm the designer arrives and finds the rig complete. He connects the console, loads the pre programmed show (he did some extra work on his laptop). Then he starts the discovery process. The console finds all the new fixtures and place them in the real fixture list. He presses the "Connect multiple fixtures mode" button. Then he selects the "Match fixtures numbers" button. The console connect each real fixture to the already present virtual fixture(but for the console it is also a real fixture, because the visualization software simulates them). The console uses the number written on the display of the fixture and the number of the fixture given by the designer during programming to match the fixtures. At a given moment he gets a error message telling that the gobo's don't match. He check this and find that the special gobo is not at the correct position. However if he selects the correct position, and looks at the real light output, the gobo is there. You updated the wrong gobo position using the menu. Using the console, the virtual gobo wheel of the fixture can be adjusted and the gobo can be virtual placed at the correct position and the old gobo at the wrong position can be loaded from the fixture internal memory. The designer now press the "Lamps on group

by group" button to strike the lamps of each power group he has created. When all the lamps are one, the designer loads the first cue. This cue will project the special gobo. The designer can now start to update the position present and at 18pm he is ready for the rehearsal.

In the next case study the show was so good that the group decided to make a tour. Now you can prepare the fixtures before loading. You can give them there number. When you hit the display number button, the display will show the number. There is no need to write it on a tape. The battery used to power the display is charged when mains is applied. You put the special gobo's in the correct fixtures and check this, also with the console, you don't want to make the same mistake you made during the first rehearsal day. Everything is now checked and ready to hit the road. All goes well the first shows and then a fixture gets brooking. You replace it with a spare one. You just write the number on the touch screen, run the discover process, and select "Replace fixture" button. When you replace the fixture with one of another type, the console will complain. It will then represent the parameters of each type and gives suggestions about witch parameter must be used to replace the old one. You can manually make adjustments (a bit like a patch). The console will now each time a parameter value is changed, during a cue or effect, perform the conversion to the new parameters. There is no change to the cue data. You can however add cue data for some specific fixture parameters. When you replace the fixture back with the correct type, the fixture is also replace in the console and the show will run as normal. Any extra data is ignored.

This article contains a lot of new principles. However most of them don't need a lot of new technology. They only need agreement, a bit of extension to the ACN protocol, and manufactures that are willing to implement it. Console manufactures can add special features or working flows, so they can distinct them self from competitors. Specialized companies can create the 3D models or ACN firmware. For fixture manufactures it will be easier to introduce new fixture features because any console will immediately be able to control them, as long as they supply all information needed for this.

This is my very personal view and maybe I must wait another 18 years for this to happen… but I hope not.