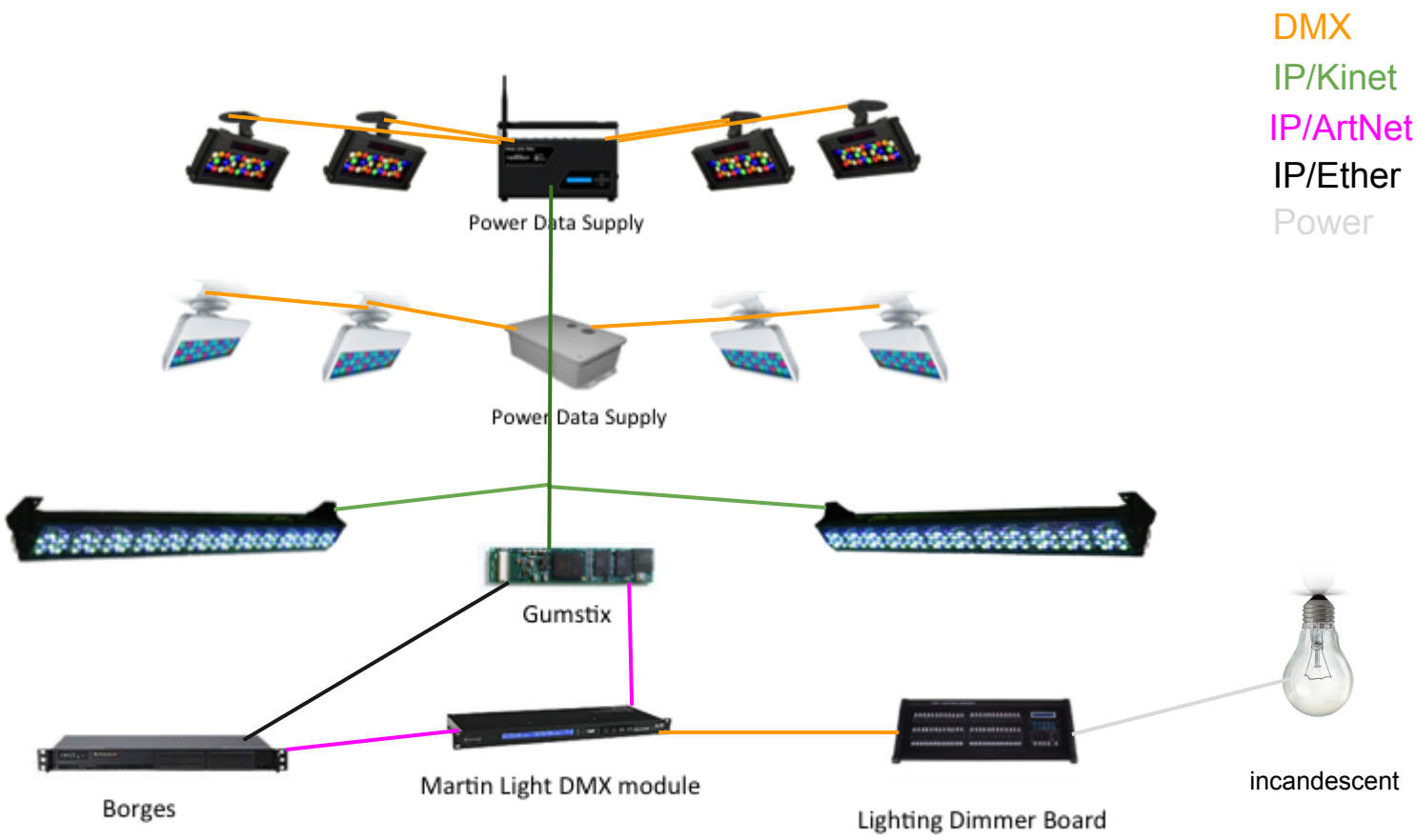
NDN LIGHTING CONTROL

system status report

1.4.201

Jeff Burke, Alexander Horn, Alessandro Marianantoni
Derek Kulinski (PyCCN) and Paolo Gasti (NameCrypto/UCI)



System Diagram (lights, gumstix, borges)

Then: (5/23)

- c implementation
- 5 lights
- used default ccnd key
- signed interest
- abrupt control patterns
- applications:
  - sequencer
  - controller (embedded)

Now: (11/04)

- python implementation
- 11 lights
- multiple keys (1 pair per app)
- signed interest (UCI namecrypto)
  - asymmetric and symmetric keys
  - 'state' hash/counter
- applications:
  - configuration manager
  - sequencer
  - mixing board fader
  - web control

Features

Signed Interests, PyCCN, NameCrypto

ccnx:/ndn/ucla.edu/apps/lighting/TV1/living-room-left/setRGB/4e0000

.../TV1/living-room-left/setRGB/4e0000/[KeyLocator]/[NameCrypto]

['ndn', 'ucla.edu', 'apps', 'lighting', 'TV1', 'living-room-left', 'setRGB', '540000',

"\x01\xe2\x01\xda\n\x950\x81\x9f0\r\x06\t*\x86H\x86\xf7\r\x01\x01\x01\x05\x00\x03\x81\x8d\x000\x81\x89
\x02\x81\x81\x00\xd8\x8e^\x9f'b\xdf4\xfc\xa9M\xf0\xd3c\x97\xf9 \xd8\xd7\x8e\xc2\xcc\xf2yp\xa5
[\xff\xf8X_!2yiu\x8e\xb91\x0c\xf9W\xd3S\x9c\x9e\n\xc6|}
\xe2O\xae\xf32\x15E\xbcc\xe1\xb3\xc3x\x85\xdd\x8e\x0b7\xab\xd6\xdc\r\xad5\x7f
(v\\E\x81\x98\xbc\xed<Sm\xfd\xf7[_\x0b\xb3\rac\xdc\xb2\xaa\xe4?
\xd3\t\xact\xabp\x9cg9\x83Q@sv\xc2\xac:\x88\x9fs0\xfd\xe2z$\x18?+\x02\x03\x01\x00\x01\x00\x00",

'@\x96\x1cQ\x00\x0bTV1Rainbow\x85\xb3\xb2N^\x9e\r\x00\x01\x00\x00\x00\x00\x00\x00\x00\x99lL\x13_
\xdc\xff,(\xff\xb3\xdb\x17\xfb\x1dSo\xc7P+\xd6\x0f\xef<\xab I\xc1ymPX'

]

Signed Interest Implementation

Keylocator for Signed Interest - example code

C:

```
struct ccn_charbuf *empty_name = ccn_charbuf_create();
struct ccn_charbuf *sigContentObj = ccn_charbuf_create();
ccn_name_init(empty_name);
ccn_charbuf_append(sigContentObj, NS_SIGNATURE, NS_SIGNATURE_LEN);
replace_name(sigContentObj, tempContentObj->buf, tempContentObj->length, empty_name);
ccn_charbuf_append_charbuf(name_signed, name);
ccn_name_append(name_signed, sigContentObj->buf, sigContentObj->length);
```

PyCCN:

```
keyLoc = Key.KeyLocator(self.key)
keyLocStr = _pyccn.dump_charbuf(keyLoc.ccn_data)
nameAndKeyLoc = Name.Name(str(fullURI))
nameAndKeyLoc += keyLocStr
```

Python Implementation

C:

compute_app_secret_key.c
Run by the configuration manager (CM).
The resulting application key must be
sent to the application

send_command_symm.c
Run on the application.
Generates an authenticated interest using the
application secret key

receive_commnand_symm.c
Run on the fixture. Authenticates an interest
received from an application.

not yet implimented:
firstauth_cm.c
Run by the configuration maneger (CM).
The resulting interest is sent to the fixture
to perform the first initialization

firstauth_fixture.c
Run on the fixture. Authenticates an interest
received from the configuration
maneger (CM) and extracts the deployment
information, such as a symmetric
(encrypted) and a pubblic key and some
additional (encrypted) information.

/name/<KeyLocator>/<state>/<Signature>

Python:

Sequencer app:

new_state()

generate_application_key
(fixtureKey,appName)

authenticate_command
(state, nameAndKeyLoc, appName, symmKey)

authenticate_command_sig
(state, nameAndKeyLoc,appName, key)

Controller app (fixture):

new_state()

verify_command
(self.state, n, 10000, fixture_key=fixtureKey,
pub_key=keyLoc2.key)

NameCrypto Implementation

Applications

CM, Sequencer, Controller, Fader, Web Control

appName = "controller"
appPrefix = "ccnx:/ndn/ucla.edu/apps/lighting/TV1/"
appDescription = "lighting controller"
keyFile = "controller.pem"

```
deviceList = (
'00:1c:42:00:00:00', '192.168.3.52', 'phillips/ColorBlast', 50009,
'00:1c:42:00:00:02', '192.168.3.53', 'phillips/ColorBlastTRX', 50011,
'00:1c:42:00:00:04', '192.168.3.51', 'phillips/ColorBlaze', 50012,
'00:1c:42:00:00:08', '169.192.0.50', 'phillips/ColorBlaze', 50013,
'00:1c:42:00:00:10', '131.179.141.17', 'ArtNet', 50010
)
```

```
names=[
{'name':'living-room-left' , 'DMX':1,'TYPE':"ColorBlazeL", 'UDP':50013},
{'name':'living-room-right', 'DMX':1,'TYPE':"ColorBlazeR", 'UDP':50012},
{'name':'window-right'     , 'DMX':1,'TYPE':"ColorBlast", 'UDP':50009},
{'name':'entrance-door'    , 'DMX':2,'TYPE':"ColorBlast", 'UDP':50009},
{'name':'stairs'           , 'DMX':3,'TYPE':"ColorBlast", 'UDP':50009},
{'name':'bedroom'          , 'DMX':4,'TYPE':"ColorBlast", 'UDP':50009},
...
]
```

Configuration File Per App

not final / totally minimal... but close

/ndn/ucla.edu/apps/lighting/[appName]

configuration:

python cm.py <app_cfg>

```
appName = "Sequencer"
keyFile = "sequencer.pem"
appPrefix = "ccnx:/ndn/ucla.edu/apps/lighting/TV1/"
controlNameSpace = {
"ccnx:/ndn/ucla.edu/apps/lighting/TV1/living-room-right/setRGB",
"ccnx:/ndn/ucla.edu/apps/lighting/TV1/window-left/setRGB",
"ccnx:/ndn/ucla.edu/apps/lighting/TV1/living-room-front/setRGB",
```

co.content=appKey

co.sign(self.key)

keyMgr.putKey(**ccnx://ndn/ucla.edu/apps/lighting/TV1/Sequencer/CM/TV1/living-room-right/setRGB/key**")

(puts in repo for persistant storage)

runtime RPC:

ccnx:/ndn/ucla.edu/apps/lighting/TV1/**living-room-right/setRGB/FA0022/[KeyLoc]**/[NameCrypto]

keyLoc points to TV1Sequencer public key at .../TV1/Sequencer/key

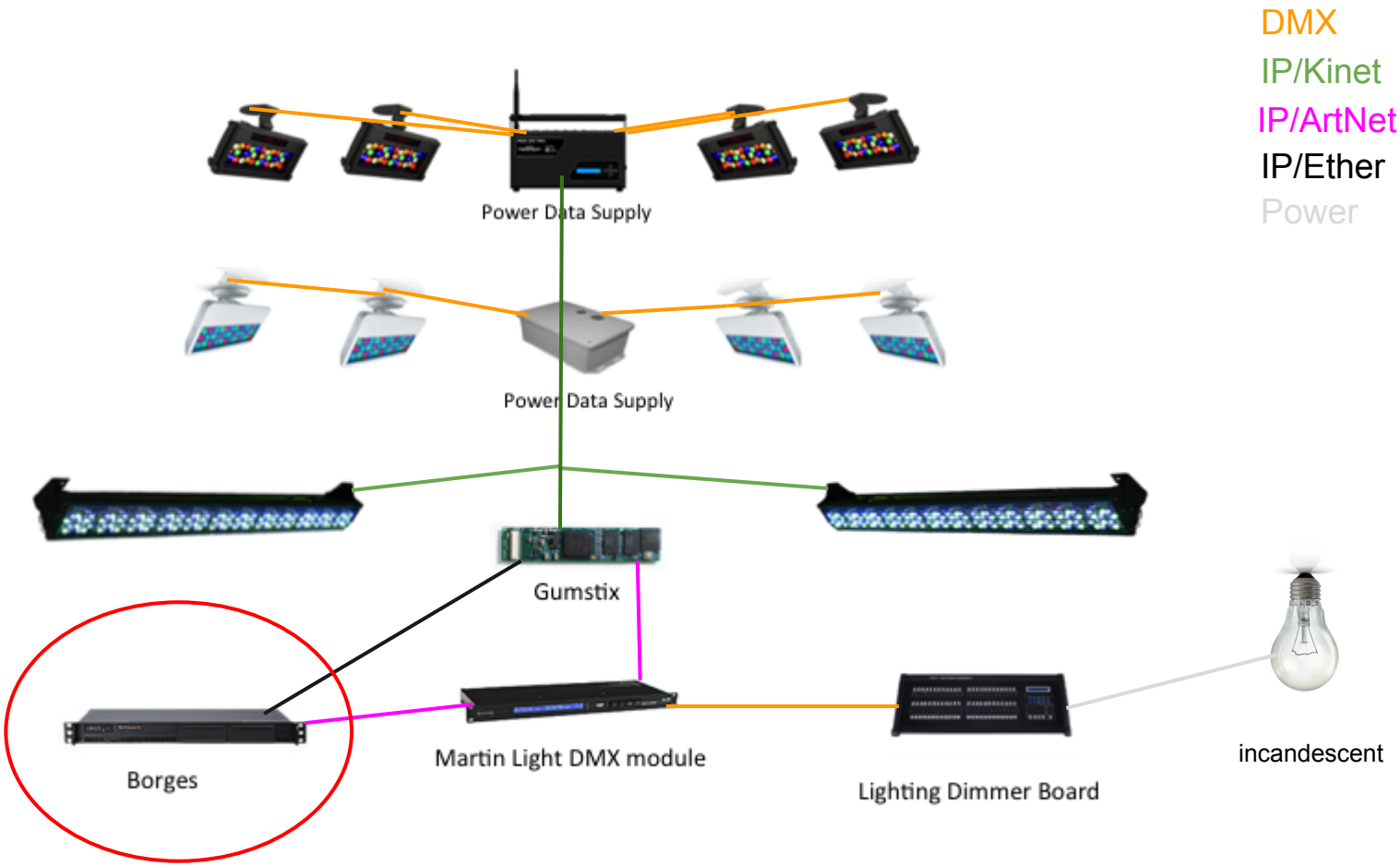
controller verifies that the RPC is verified by reconstructing above *cm signed* key location

for each name, on first interest during controller instance (or at interval N), fixture ensures commanding Key is signed by trusted (CM) key

(trusted CM key/s are inserted during configuration / first auth of fixture)

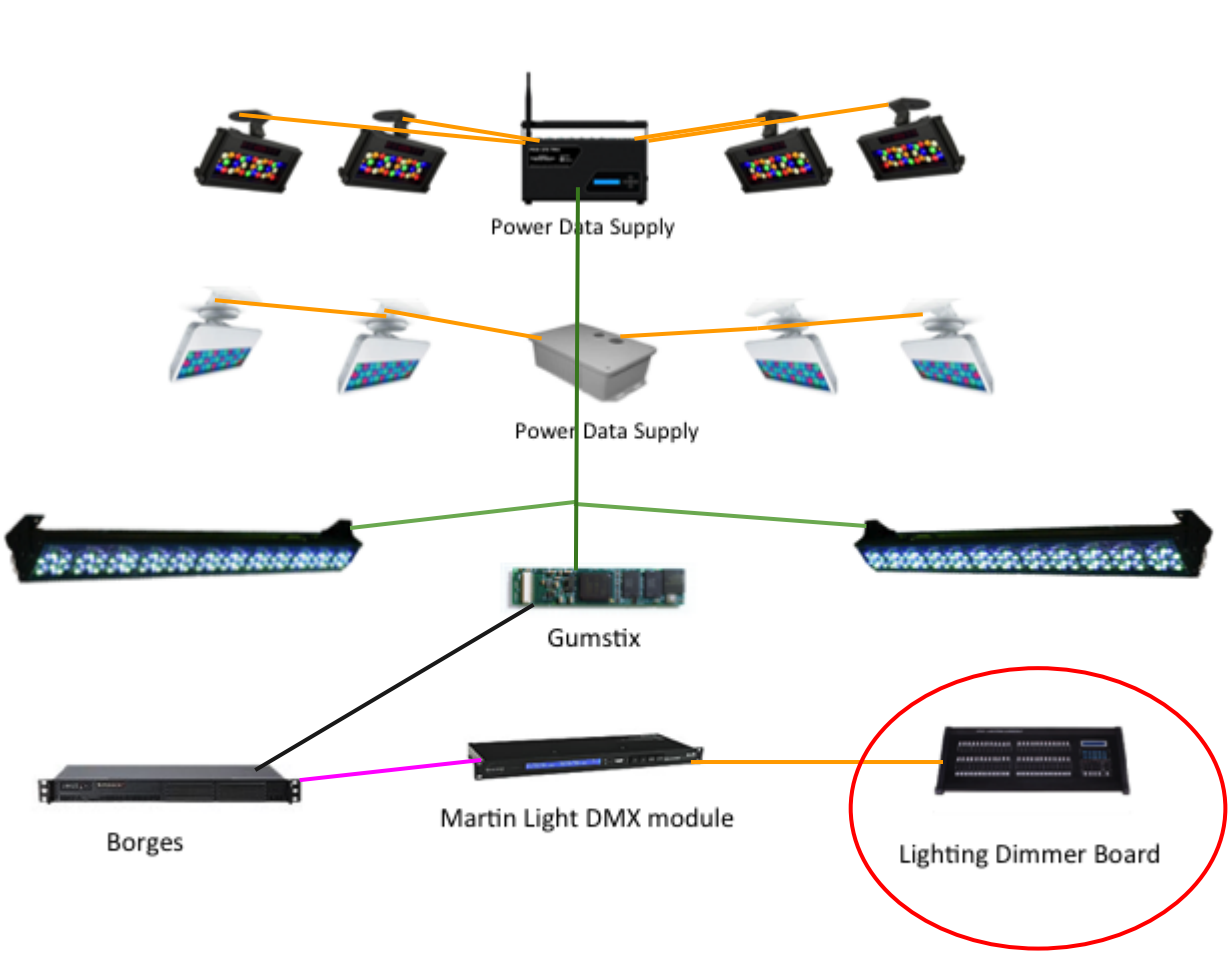
Configuration Manager

/ndn/ucla.edu/apps/lighting/CM



Sequencer

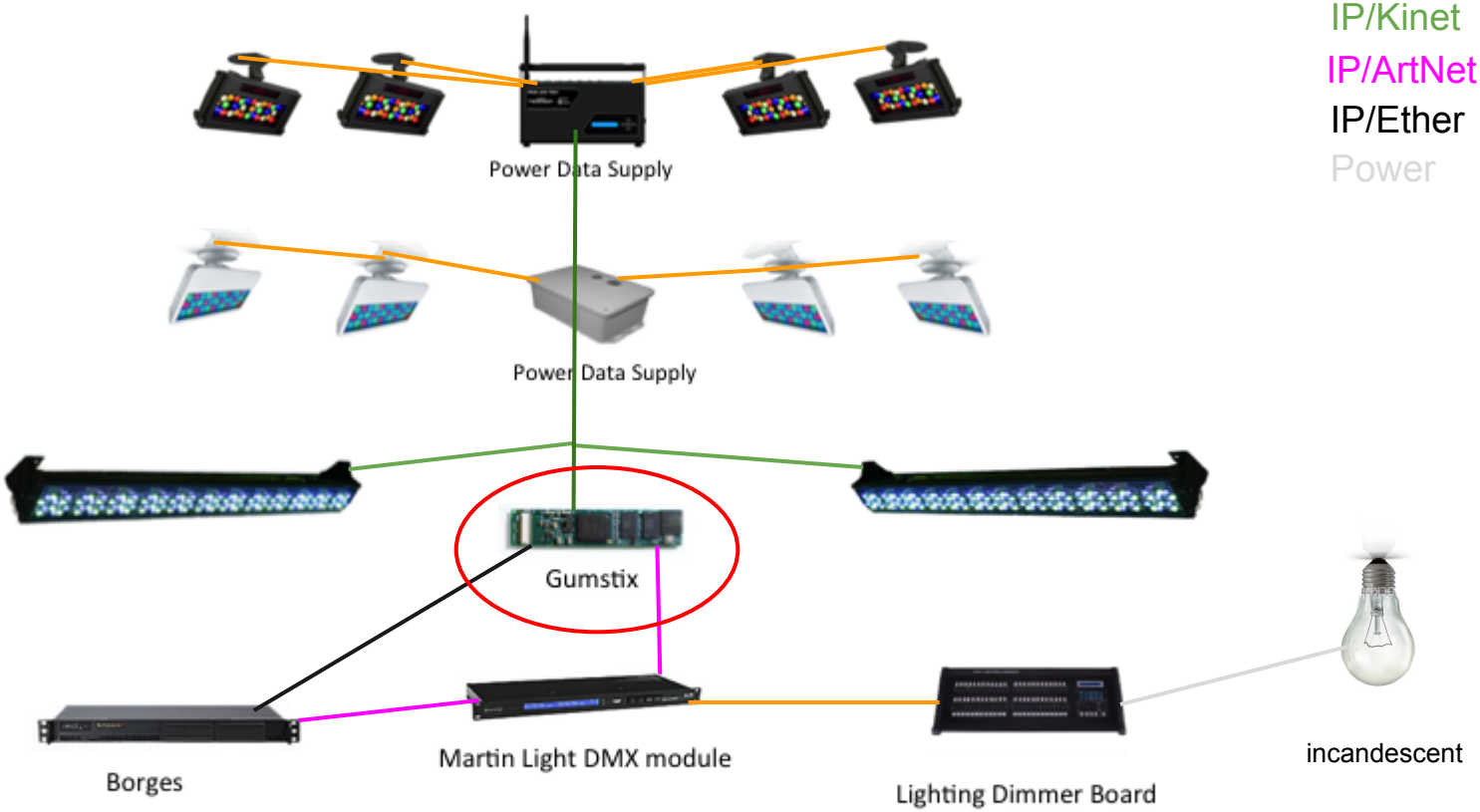
/ndn/ucla.edu/apps/lighting/TV1/Sequencer



3 sliders control RGB of all 10 LED lights

Fader (physical, from board)

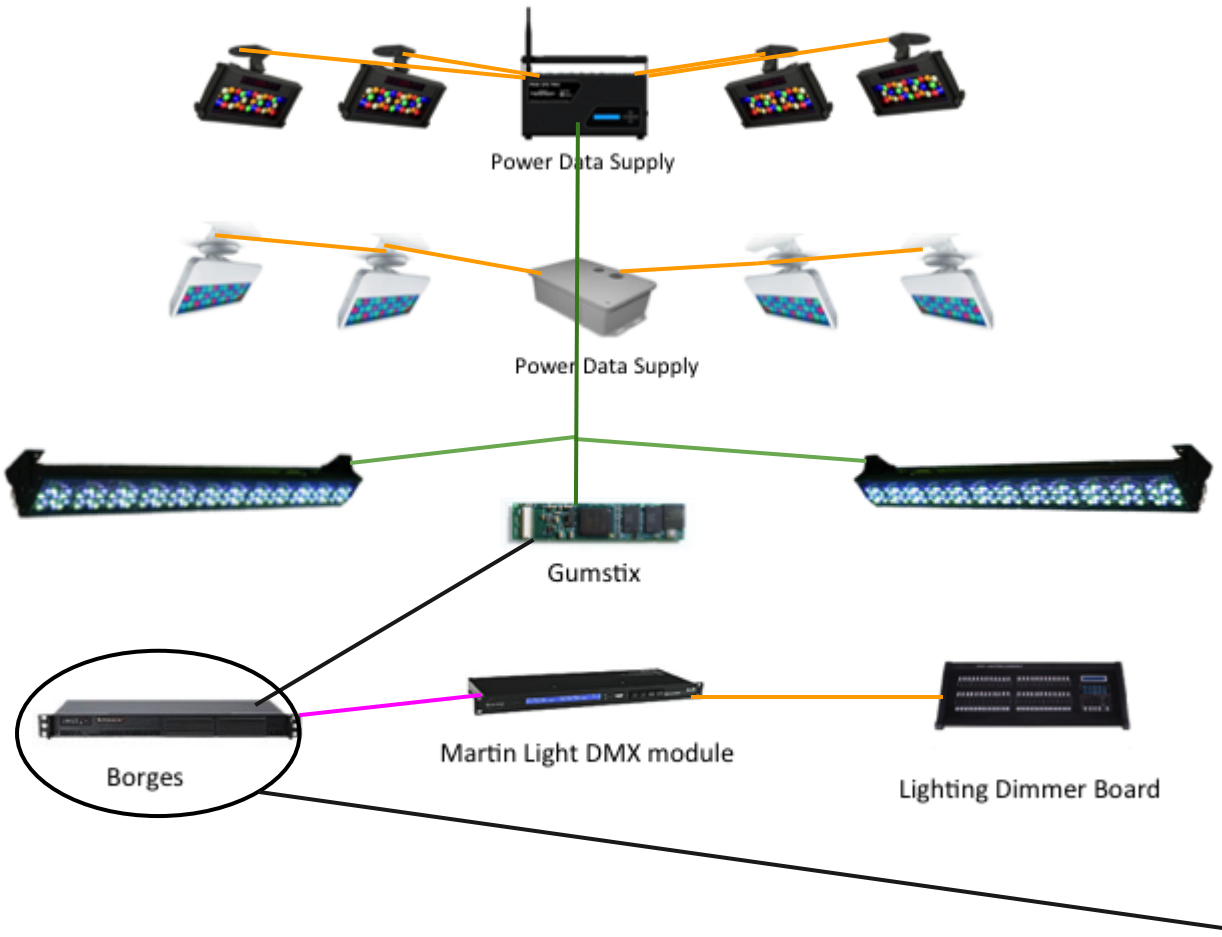
/ndn/ucla.edu/apps/lighting/TV1/Fader

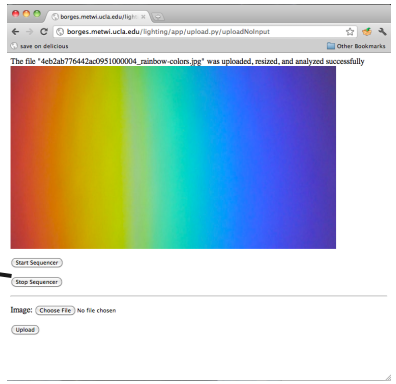


Controller (embedded NDN->IP lighting)

/ndn/ucla.edu/apps/lighting/TV1/Controller

DMX
IP/Kinet
IP/ArtNet
IP/Ether





Web Fader

/ndn/ucla.edu/apps/lighting/TV1/WebControl

Performance Results

Timing, profiling

```
ncalls  tottime  percall  cumtime  percall filename:lineno(function)
2611    1.809    0.001  148.306    0.057 controller.py:99(upcall)
2612    2.310    0.001  111.535    0.043 controller.py:80(makeDefaultContent)
2612    0.926    0.000   93.887    0.036 ContentObject.py:43(sign)
2612   87.034    0.033   87.173    0.033 {pyccn._pyccn.encode_ContentObject}
2611    0.978    0.000   21.567    0.008 controller.py:167(parseAndSendToLight)
2610    0.410    0.000   19.488    0.007 controller.py:224(sendData)
```

generally around 60 ms rt - *w/ CO*

```
ncalls  tottime  percall  cumtime  percall filename:lineno(function)
2611    3.333    0.001   39.638    0.015 controller.py:99(upcall)
2611    2.844    0.001   15.859    0.006 controller.py:167(parseAndSendToLight)
2611    6.823    0.003   11.205    0.004 {pyccn._pyccn.KeyLocator_obj_from_ccn}
2610    0.836    0.000   10.075    0.004 controller.py:224(sendData)
2610    9.238    0.004    9.238    0.004 {method 'sendto' of '_socket.socket' objects}
2611    2.305    0.001    8.970    0.003 Interest.py:21(__init__)
```

generally around 18 ms rt - *w/o CO*

Timing, profiling

UCLA NDN Lighting

NDN findings

python faster to work with esp while prototyping

a way around asym CO ?

what's next...

- override
- bootstrapping / secure hot-swapping
- 'naming designer UI' / UI for app cfg
- more embedded controllers
- upgrade ip drivers:
  - to 16 bit (from 8bit)
  - to use white & amber
- INFOCOM Paper

•Application keys are published under prefixes that define their capabilities:

/<root>/lighting/<capability>/<app_name>/key

e.g., light board interface:

/ndn/ucla.edu/apps/lighting/control/light_board/key

•Apps issue signed interests to control the fixtures, which use commands and signatures concatentated onto any name by which the fixture can be addressed.

/<path_to_fixture>/<capability>/<param_pattern>/<parameters>/<signature>

/ndn/ucla.edu/melnitz/1471/lights/west_wall/wash_down/
control/rgb-8bit-hex/F0FF39/[sigbits]

Trust Namespace spec

earlier design spec not presented in meeting

•Trust delegation: If a fixture does not have the signing key for a command cached, it checks to see if this name returns a copy of that key signed by a key it trusts:

/<path_to_key>/authority/<name_used_to_access_fixture>/<capability>
/ndn/ucla.edu/apps/lighting/control/light_board/key/authority/
ndn/ucla.edu/melnitz/1471/lights/west_wall/wash_down/control

Trust Namespace spec

earlier design spec not presented in meeting