Very Lightweight Process Variable Server

Paper ID: WEPHA151

Andrei Sukhanov, James Jamilkowski
Brookhaven National Laboratory, Upton, NY, USA

Pure Python

ABSTRACT

Modern instruments are often supplied with rich proprietary software tools, which makes it difficult to integrate them to an existing control system. The liteServer is very lightweight, low latency, cross-platform network protocol for signal monitoring and control. It provides very basic functionality of popular channel access protocols like CA or pvAccess of EPICS. It supports request-reply patterns: 'info', 'get' and 'set' requests and publish-subscribe pattern: 'monitor' request.

MOTIVATION

- Provide control for devices connected to non-linux machines.
- Provide control for devices using proprietary software (through DLLs or shared libraries).
- Provide minimal latency of the ethernet transactions.
- **Allow for implementation in FPGA without CPU core**.
- Easy integration to existing Control Architecture (RHIC ADO, EPICS)

liteServer is a possible alternative to EPICS Channel Access for small systems

CLIENT-SERVER MODEL

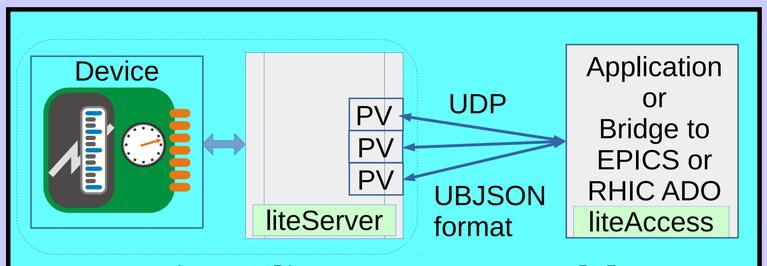


Fig.1 Client – Server Model

Minimal Requirements

- UDP stack.
- UBJSON
- Python.

UBJSON Advantage

- Complete compatibility with the JSON specification – there is a 1:1 mapping between standard JSON and UBJSON.
- Ease of implementation.
- Easy of use.
- Speed and efficiency – UBJSON uses data representations that are (roughly) 30% smaller than their compacted JSON counterparts and are optimized for fast parsing. Streamed serialization is supported, meaning that the transfer of UBJSON over a network connection can start sending data before the final size of the data is known.

FEATURES

Supported requests

- 'info': returns a list of devices, parameters or features
- 'get': get values or features
- 'set': set values or features
- 'monitor': defines a callback which will be called each time the value changes (not fully implemented yet).

Returned data

'get' returns a dictionary, keys are 'device:parameter', values are lists of objects:
{'dev1:counters': {'value': [-10, 10, ...,
'numpy': [[120, 160, 3], 'uint8']}}

Multi-dimentional arrays (numpy)

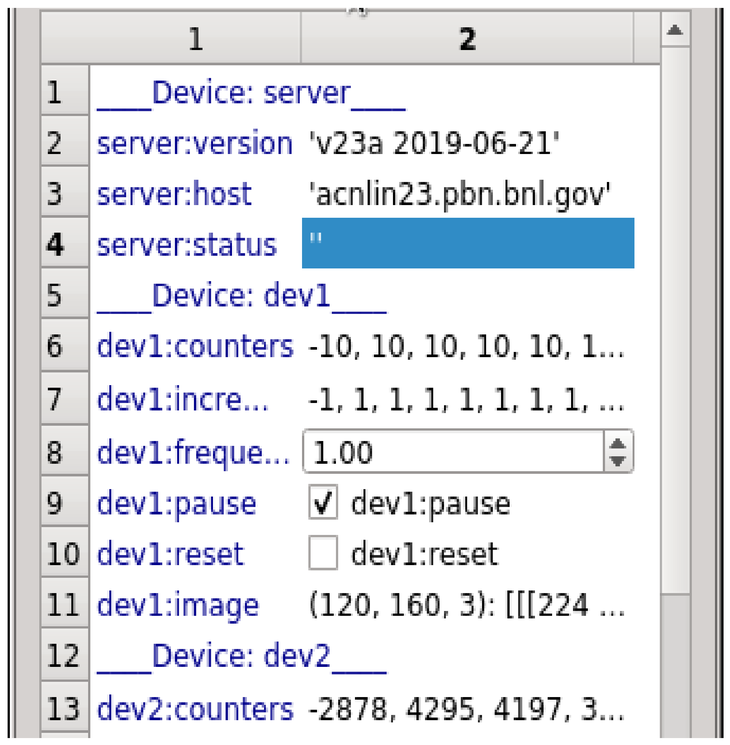
```
{'dev1:image': {'value': array([[[224,   1,   2],
       [  3,   4,   5],
       [  6,   7,   8],
       ...,
       [215, 216, 217],
       [218, 219, 220],
       [221, 222, 223]],
      [[2...t8), 'numpy': [[120, 160, 3], 'uint8']}}
```

Access-control-ready

Username and program ID information is supplied in the client request, it can be used on the server side to protect access to critical PVs.

Spreadsheet GUI, (PyQt5)

- Automatic conversion of parameters to GUI elements: Buttons, TextEdits, SpinBoxes, CheckBoxes.
- User-configurable



liteAccess: access to PVs at liteServer

Steps

- Create access object to PV (or multiple PVs)

```
manyPVs = PV(('host;port',('dev1','dev2',...),
('par1','par2',...)))
```

e.g. to access all parameters of dev1:

```
allPVs = PV(('host;port',('dev1'))
```

- Use PV.info() to get information about features

```
manyPVs.info()
```

- PV.value is python @property, to get values:

```
values = manyPVs.value
```

- To set values

```
manyPVs.value = values
```

Example

Access to a single PV:

```
from liteAccess import PV
aPV = PV(('localhost;9700',('dev1'),
                           ('frequency')))
aPV.info()
{'dev1:frequency': {'value': '?',
  'count': [1],
  'features': 'RW',
  'desc': 'Update frequency of all counters',
  'opLimits': [0, 10]}}
aPV.value
{'dev1:frequency': {'value': [1.0]}}
```

Access to multiple PVs:

```
manyPVs = PV(('localhost;9700',('dev1','dev2'),
                           ('frequency','pause')))
manyPVs.value
{'dev1:frequency': {'value': [1.0]},
 'dev1:pause': {'value': [True]},
 'dev2:frequency': {'value': [1.0]},
 'dev2:pause': {'value': [False]}}
```

STATUS

Several devices at RHIC are served by the liteServer, hosted on Windows and Raspberry Pi platforms:

- Magnetometers
- Laser interferometers
- Infrared cameras.

Devices are integrated into the RHIC Control Architecture.
Transfer of several-megabyte data samples at 50 MB/s has been demonstrated.
Program sizes: 350 lines both liteServer and liteAccess.

TODO List

- **Implement liteServer in FPGA**.
- Support the 'monitor' request.
- Add optional TCP protocol (for large transfers).
- Improve multi-client performance.
- Plotting (Imaging is supported by an imageViewer)

REFERENCES

https://github.com/ASukhanov/liteServer

ACKNOWLEDGMENTS

Work supported by Brookhaven Science Associates, LLC under Contract No. DE-SC0012704 with the U.S. Department of Energy