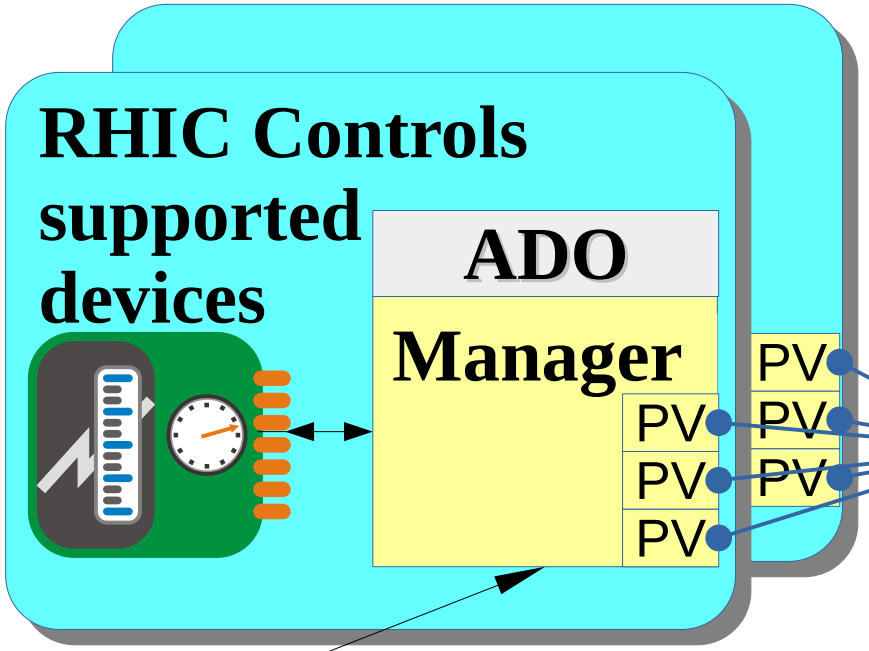
Benchmarking of ADO->EPICS Bridge

- Bridge flowchart.
- Benchmarking control flow.
- Same-host bridge crossing times.
- Summary.

Bridge flowchart

Create PVs:
- Translate EPICS properties to ADO,
- Supply writable parameters with **setters**,
  - Handle legalValues,etc.

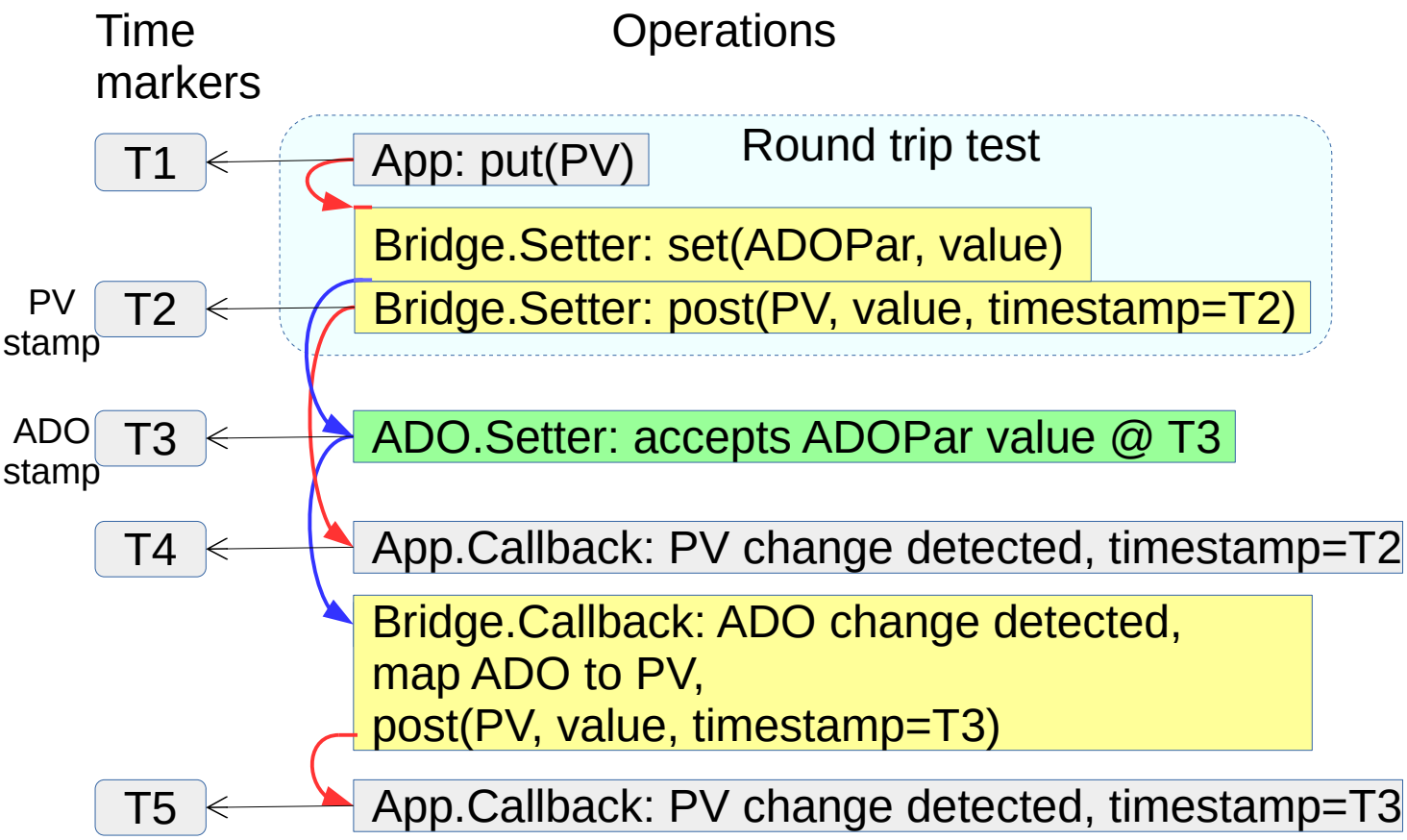


Subscription:

```
for par in ADOpars:
  subscribe(callback,par)
```

```
def callback(args):
   for ADOpar,value in args:
      pv(ADOpar).post(value)
```

Benchmarking control flow



The App is a benchmarking application.
It subscribes to target PV.

Bridge crossing time:
cb_arrival: T5 – T3
Round trip time:
round_trip: T5 – T2

Color coding:
Application operations
Bridge operations
ADO manager operations
EPICS events flow
ADO events flow

Same-host bridge crossing times

PV name	Number of Items	Size (bytes)	cb_arrival (ms)
cscompile01: am_perf: intArray **from Python ADO**	125000	1000040	31.69
	12500	100040	3.57
	1250	10040	0.67
	125	1040	0.47
	12	136	0.34
cscompile01: simple.test: varArrayS **from C++ ADO**	250000	1000040	38.43
	25000	100040	3.79
	2500	10040	0.77
	250	1040	0.36
cscompile01: am_perf: byteArray **from Python ADO**	1000000	1000057	2.91
	100000	100057	0.46
	10000	10057	0.36
	1000	1057	0.36

Slow per-element XDR packing **30 MByte/s**

Same crossing time from Python and C++ ADOs

Fast transfer of byteArrays **330 MByte/s**

ADO drawback: If data size > 1K bytes, then it need to be packed as byte array

Or numpy arrays

Setup time 0.36 ms

Summary

- Setup time of the bridge crossing is ~0.35 ms.
- Data transfer of arrays is ~30 MB/s. Most likely it is limited on ADO side due to slow per-element packing in RPC XDR.
- Data transfer of byte arrays is ~300 MB/s.
- Python and C++ ADOs have similar crossing times.
- If data size is larger than 1 kByte then it make sense to pack it inside the ADO manager as byte array or numpy array.
- It is not obvious that the ADO_Srv approach would be significantly faster than the adoEpics bridge.