

# THE TINE CONTROL SYSTEM PROTOCOL

HOW TO ACHIEVE  
HIGH SCALABILITY  
AND PERFORMANCE



# Data Flow and Scalability

- Most Scalability issues in a distributed system are concerned with *data flow*.
- Most of the data flow in a distributed control system is concerned with *readback* data.
  - Display
  - Middle Layer logic (automation decisions)
  - Archiving
- The *Control System Protocol* is concerned with data flow.
  - Server i/o -> clients
  - Load on server
  - Load on network
  - API

# Control System Protocols

- Initial Driving Force :

- KISS: 

- e.g. transaction polling is easy to program!

- Initial Consequence:

- 

- Fewer options often lead to more and bigger problems
    - Solutions sometimes artificial
    - Corollary: And later on with more tools available:
      - Still use the hammer ('cause that's what you know)!

## Data Flow Memes : 0<sup>th</sup> Order

- Transaction-based Client-Server
  - Client asks, server responds
  - KISS (no management tables) !
  - Suitable for *small* systems
  - Server Load:

$$L_S \sim N_C \times N_T \times L_D \times U_T$$

Threads ?

Multi-Core ?

$L_S$  = Load on Server/sec (CPU cycles spent)

$N_C$  = Ave. num clients

$N_T$  = Ave num transactions / client

$L_D$  = Ave Load handling a dispatch

$U_T$  = Ave update rate

## Data Flow Memes : 0<sup>th</sup> Order

- Network Load:

Threads ?

Multi-Core ?

$$L_N \sim N_C \times N_T \times P_T \times U_T \times 2$$

$L_N$  = Load on Network (bytes/sec)

$N_C$  = Ave. num clients

$N_T$  = Ave num transactions / client

$P_T$  = Ave Transaction Payload

$U_T$  = Ave update rate

2 = outgoing + incoming payloads ~equal

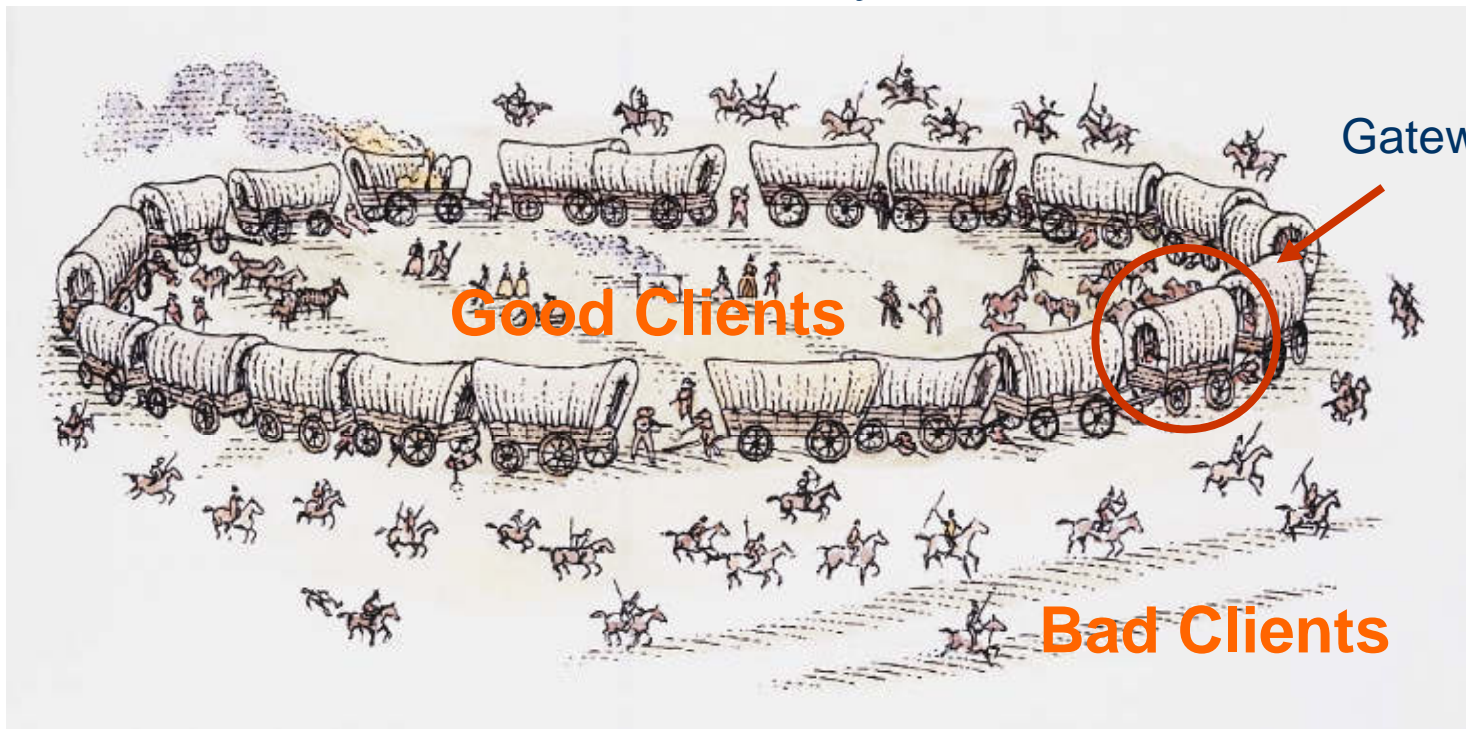
- Increase Scalability => Reduce the Load

Reduce any of these factors !

Transaction-based Client-Server

# Data Flow Memes : 0<sup>th</sup> Order

Possible solutions to Scalability Problems :



## Data Flow Memes : 1<sup>st</sup> Order

- Contract-based Publish-Subscribe
  - Kiss **KISS** goodbye !
    - Contract and connection management
    - Transaction => managed contract + table of clients
    - Larger systems
    - Server Load :

$N_C$  (Number of Clients) no longer a factor !

$$L_S \sim N_T \times L_D \times U_T$$

$L_S$  = Load on Server/sec

$N_T$  = Ave num transactions / client

$L_D$  = Ave Load handling a dispatch

$U_T$  = Ave update rate

## Data Flow Memes : 1<sup>st</sup> Order

- Network Load:
  - Similar, but:
    - Factor 2 gone!
    - Can use '*Send on Change*' to reduce  $U_T$
    - Can use *multicast* to reduce  $N_C$
- Great Benefit to Scalability!
  - BUT: API can still allow inefficiency !
  - AND: Who is doing the programming ?

$$L_N \sim N_C \times N_T \times P_T \times U_T$$

$L_N$  = Load on Network (bytes/sec)

$N_C$  = Ave. num clients

$N_T$  = Ave num transactions / client

$P_T$  = Ave Transaction Payload

$U_T$  = Ave update rate



# APIs and 'Joe, the Programmer'

- Some APIs still allow or even 'only allow' synchronous polling !
  - Easy for Joe to understand
    - no callbacks or events ...
  - $N_C$  once again a factor in the server load!
- Some APIs guide Joe to use individual transactions for all readback values
  - Each of the 300 BPM positions, status, etc.
  - Each of the 600 Ion Pump pressures, status, etc.
  - Each of the 1000 PSC currents, setpoints, status, etc.
  - $N_T$  can be very large !
- But: **These client applications are driving the data flow !**

## Data Flow Memes : 2<sup>nd</sup> Order

- Publish-Subscribe with Contract-Coercion
  - Focus on ways to reduce  $N_C$  and  $N_T$  in server load and network load
    - Police action ?
      - Find Joe and tell him to do it differently.
    - Push the data ?
      - multicast everything!
      - Network load could increase dramatically
      - Client load could increase dramatically
  - Goal: Keep the APIs and let Joe do what he wants.

## Data Flow Memes : 2<sup>nd</sup> Order

- Analyze the transaction request
  - Map to an existing contract if possible
  - Anticipate future requests and renegotiate the contract with the client
    - e.g. “if he’s asking for BPM#1, then he’ll probably want BPM#2 as well”
  - Guide synchronous and asynchronous acquisitions
    - Don’t monitor ‘static data’
    - Don’t synchronously poll monitorable data.
    - Trap ‘foolish’ update intervals
  - KISS is a distant memory
- Briefly review 3 Control System Architecture Models ...

# Control System Models (a review)

- Model I: Database Model
  - **EPICS**, **VISTA** (i.e. VSystem not the OS)
  - Control system data are *elements in a database*.
  - Transfer *Process Variables*
    - pvData have names
    - Actions are 'get', 'set', 'monitor'
  - BUT: Some things aren't variables at all !
    - e.g. command and calls

# Control System Models (a review)

- Model II: Device Server Model

- TANGO, DOOCS, ACS, STARS\*, TINE\*
- Elements are controllable objects managed by a device server.
- Instance of such an object is a *device*, with a hierarchical name.
- Actions pertaining to a device given by its *properties* !
  - i.e. *get, set, monitor, call* some *property* OR *command*
- BUT: some things aren't devices !
  - e.g. "\*" is NOT a device.
- AND: some services are *Property-orientated* !

# Control System Models (a review)

## ● Model III: Property Server Model

- STARS\*, TINE\* (maybe ACS?)
- Elements are *services* with *properties* (or methods)
- Same basic hierarchy as Device Server Model
- Properties have Keywords
  - (instead of Devices having Properties)
- e.g. Middle layer services
  - Name Server
  - Central Alarm Server
  - Central Archive Server
  - CDI Server
  - etc.
- BUT: Not everything divides cleanly into *Device Server* or *Property Server*!

# Transaction Coercion

- Steps to reduce  $N_C$ ,  $N_T$ ,  $L_D$ , and  $U_T$
- Refer to the *Property* in the above models
  - the ‘get’ and ‘monitor’ operations produce most of the data flow in a distributed control system.
- Some Examples

# Transaction Coercion

- Multi-Channel Arrays (**MCA**)
  - An array of all values of all devices for a given *Property* with a well-known order.
    - Same units, settings, etc.
  - Registered Property can declare itself to be an **MCA**.
  - Strict OO Device Servers can declare device *groups* with **MCA** characteristics.
  - e.g. BPMs, BLMs, IonPumps, Temps, PSCs, etc.
  - But Joe's Client Panel will have a widget for a each thing individually !
  - What to do?

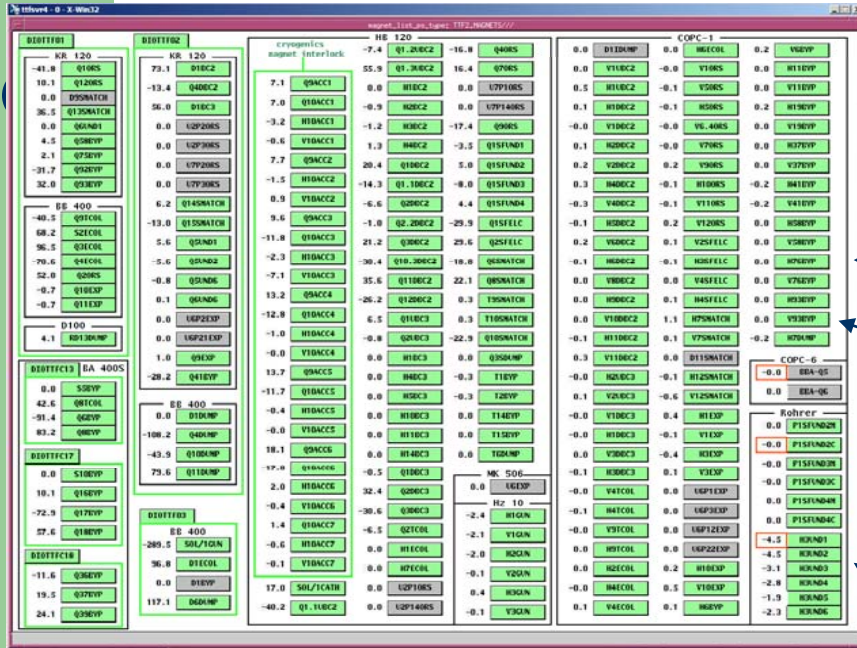


# Transaction Coercion

- Contract-renegotiation
  - delivers entire **MCA**
  - informs client that widget #18 needs element number #43 of the array !
- $N_T$  is decimated !

# FLASH DDD PS Panel with TINE Multichannel Polling

10 connections/property (was ~260)  
50 contracts (was > 1000)



TTMAG-AB/GROUP  
(PC104, 41 elements)

TTMAG-CD/GROUP  
(PC104, 41 elements)

- STEERER/SHGROUP (57)
- STEERER/SVGROUP (48)
- QUAD/QDGROUP (40)
- DIPOLE/DIGROUP (19)
- UNDULATOR/UNGROUP (7)
- MAIN/MNGROUP (6)
- SEXT/SXGROUP (3)
- SOL/1GUN (1)
- SUN (multi-server)

# PETRA BPMs

227 Libera modules

~ 15 Client

~ 35 Contracts

1 run-of-the-mill PC  
with Ubuntu Linux

~ 7% CPU load !

```
1:mstbs02 - default - SSH Secure Shell
File Edit View Window Help
Quick Connect Profiles

get contracts
> CONTRACT
> [0] LBREQM FLAGMASK <BPM_SWR_13> (227 elements) 1000 msec PETRACON
ACCPPER4B.1
>
> [1] LBREQM DATAINFOR <#0> (9040 elements) 1000 msec PETRACON
> [2] LBREQM DD_MC_TIME <BPM_SWR_13> (227 elements) 1000 msec KAROL
PETRACON
> [4] LBREQM DDTRIGSTATUS <BPM_SOR_67> (2 elements) 1000 msec PETRACON
> [5] LBREQM bpmStatus <#0> (227 elements) 1000 msec PETRACON
PETRACON
> [6] LBREQM envgain <BPM_SWR_13> (227 elements) 1000 msec SCOPEDATA.2
> [7] LBREQM NALARMS <*> (6 elements) 500 msec PECASFEC
> [8] LBREQM PM_TRIGGER <BPM_SWR_13> (227 elements) 1000 msec PETRACON
> [10] LBREQM PMBUSY <BPM_SWR_13> (227 elements) 1000 msec PETRACON
> [11] LBREQM ORBIT <BPM_SWR_13> (6072 elements) 100 msec ACCPPER4B.1
> [12] LBREQM ORBIT <#0> (6072 elements) 1000 msec PETRACON
> [13] LBREQM envilk <#0> (1816 elements) 3000 msec PETRACON
> [14] LBREQM ACTIVITY <#0> (68 elements) 30000 msec PESPYFEC
> [15] LBREQM DD_READTIME <BPM_SWR_13> (227 elements) 2000 msec TUNEMEASURE.7
> [16] LBREQM DD_SIZE <BPM_SWR_13> (227 elements) 2000 msec TUNEMEASURE.7
> [17] LBREQM DD_RAW <BPM_SWR_13> (227 elements) 2000 msec TUNEMEASURE.7
> [18] LBREQM DD_DECI <BPM_SWR_13> (227 elements) 2000 msec TUNEMEASURE.7
> [19] LBREQM DD_OFFSET <BPM_SWR_13> (227 elements) 2000 msec TUNEMEASURE.7
> [20] LBREQM DD_TMINTERVAL <BPM_SWR_13> (227 elements) 2000 msec TUNEMEASURE.7
> [21] LBREQM DD_X <BPM_SWR_13> (8196 elements) 2000 msec TUNEMEASURE.7
> [22] LBREQM DD_Y <BPM_SWR_13> (8196 elements) 2000 msec TUNEMEASURE.7
> [23] LBREQM SA_X <BPM_SWR_13> (227 elements) 100 msec MATLAB
MATLAB
MATLAB
MATLAB
> [24] LBREQM SA_Y <BPM_SWR_13> (227 elements) 100 msec MATLAB
MATLAB
MATLAB
MATLAB
> [25] LBREQM SA_Sum <BPM_SWR_13> (227 elements) 1000 msec SCOPEDATA.2
> [26] LBREQM ADC_CNT <BPM_SWR_13> (227 elements) 3000 msec PETRACON
> [27] LBREQM SRVSTATS <> (10 elements) 5000 msec PEFCSTATS
> [28] LBREQM DDTRIGSTATUS <BPM_MOR_101> (2 elements) 1000 msec PETRACON
> [29] LBREQM SRVLSTACCESS <#0> (1 elements) 30000 msec PESPYFEC
> [31] LBREQM FLAGDISA <BPM_SWR_13> (227 elements) 1000 msec PETRACON
> [32] LBREQM active <BPM_SWR_13> (227 elements) 100 msec PETRACON
> [33] LBREQM DDTRIGSTATUS <BPM_SWR_13> (2 elements) 1000 msec PETRACON
> [34] LBREQM ADCTRIGSTATUS <BPM_MOR_101> (2 elements) 1000 msec PETRACON
> [35] LBREQM MODESETTING <BPM_SWR_13> (227 elements) 1000 msec PETRACON
> [36] LBREQM ADCTRIGSTATUS <BPM_SOR_67> (2 elements) 1000 msec PETRACON
>
Connected to mstbs02 SSH2 - blowfish-cbc - hmac-md5 - none 82x47
```

# PETRA Liberas ...

CPU: 9 %

```
acclxpecspi - default - SSH Secure Shell
File Edit View Window Help
Quick Connect Profiles

fecadmin@acclxpecspi:~$ attachfec LBRES
Remote session established
get clients

```

	CLIENT	ADDRESS	PROTOCOL	Contracts
>	(0) TUNEMEASURE.7	131.169.151.223	UDP	6
>	(1) GLOBALS.6	131.169.151.223	UDP	3
>	(2) REFORBIT	131.169.151.41	UDP	1
>	(3) PEFECSTATS	131.169.119.64	UDP	1
>	(4) PESPYFEC	131.169.119.64	UDP	2
>	(5) PETRACON	131.169.121.190	UDP	6
>	(6) PETRACON	131.169.121.186	UDP	6
>	(7) FSPFH	131.169.216.46	UDP	8
>	(8) FSPFH	131.169.216.46	UDP	2
>	(9) PEMARCH	131.169.119.74	UDP	2
>	(10) PETRACON	131.169.121.187	UDP	6
>	(11) MATLAB	131.169.119.71	UDP	435
>	(12) PETRACON	131.169.121.188	UDP	6
>	(13) PETRACON	131.169.121.190	UDP	1
>	(14) MATLAB	131.169.119.71	UDP	75
>	(15) DUVAL	131.169.9.107	UDP	2

```

Connected to acclxpecspi SSH2 - blowfish-cbc - hmac-md5 - none 79x20
```

```
acclxpecspi - default - SSH Secure Shell
File Edit View Window Help
Quick Connect Profiles

top - 15:37:45 up 315 days, 6:46, 4 users, load average: 2.83, 3.87, 2
Tasks: 95 total, 1 running, 94 sleeping, 0 stopped, 0 zombie
Cpu(s): 1.8%us, 1.7%sy, 0.0%ni, 95.2%id, 0.0%wa, 0.0%hi, 1.3%si, 0
Mem: 2075380k total, 1334152k used, 741228k free, 506520k buffers
Swap: 1855468k total, 0k used, 1855468k free, 422748k cached


```

PID	USER	PR	NI	VRT	RES	SHR	S	%CPU	*MEM	TIME+	COMMAND
29391	fecadmin	15	0	2456m	133m	1452	S	9	6.6	133:37.95	rpcserv
8150	fecadmin	15	0	2068	1068	816	R	0	0.1	0:00.14	top
30836	fecadmin	15	0	2392	936	704	S	0	0.0	0:11.88	autoproc
16304	fecadmin	25	0	8188	1956	1252	S	0	0.1	0:01.03	sshd
16305	fecadmin	15	0	6724	3276	1376	S	0	0.2	0:00.29	bash
28549	fecadmin	15	0	4268	1348	944	S	0	0.1	0:00.12	sftp-server
22781	fecadmin	25	0	6732	3284	1376	S	0	0.2	0:00.25	bash
26231	fecadmin	15	0	6756	3304	1372	S	0	0.2	0:00.14	bash
7193	fecadmin	15	0	7884	1740	1232	S	0	0.1	0:00.00	sshd
7203	fecadmin	15	0	6756	3284	1352	S	0	0.2	0:00.12	bash

```

Connected to acclxpecspi SSH2 - blowfish-cbc - hmac-md5 - none 73x20
```

```
acclxpecspi - default - SSH Secure Shell
File Edit View Window Help
Quick Connect Profiles

>[137] LBREQM DD_X <BPM_NWL_75> (10 elements) 1000 msec MATLAB
>[139] LBREQM ADC_C <BPMF_NOR_52> (1024 elements) 1000 msec MATLAB
>[140] LBREQM DD_Y <BPM_NL_126> (10 elements) 1000 msec MATLAB
1] LBREQM DD_X <BPM_NWL_61> (10 elements) 1000 msec MATLAB
2] LBREQM ADC_B <BPMF_NOR_70> (1024 elements) 1000 msec MATLAB
3] LBREQM ADC_D <BPMF_NOR_52> (1024 elements) 1000 msec MATLAB
4] LBREQM DD_X <BPM_NWL_46> (10 elements) 1000 msec MATLAB
7] LBREQM DD_X <BPM_NWL_31> (10 elements) 1000 msec MATLAB
8] LBREQM DD_Y <BPM_NL_111> (10 elements) 1000 msec MATLAB
0] LBREQM ADC_A <BPMF_NOR_55> (1024 elements) 1000 msec MATLAB
1] LBREQM DD_X <BPM_NWL_13> (10 elements) 1000 msec MATLAB
3] LBREQM DD_X <BPM_NWL_1> (10 elements) 1000 msec MATLAB
4] LBREQM ADC_B <BPMF_NOR_55> (1024 elements) 1000 msec MATLAB
5] LBREQM ADC_A <BPMF_NOR_73> (1024 elements) 1000 msec MATLAB
6] LBREQM DD_Y <BPM_NL_97> (10 elements) 1000 msec MATLAB
7] LBREQM DD_X <BPM_NWL_13> (10 elements) 1000 msec MATLAB
8] LBREQM ADC_C <BPMF_NOR_55> (1024 elements) 1000 msec MATLAB
0] LBREQM DD_X <BPM_NWL_31> (10 elements) 1000 msec MATLAB
1] LBREQM ADC_D <BPMF_NOR_55> (1024 elements) 1000 msec MATLAB
2] LBREQM ADC_B <BPMF_NOR_78> (1024 elements) 1000 msec MATLAB
3] LBREQM DD_X <BPM_NWL_46> (10 elements) 1000 msec MATLAB
4] LBREQM DD_Y <BPM_NL_82> (10 elements) 1000 msec MATLAB
5] LBREQM ADC_D <BPMF_NOR_58> (1024 elements) 1000 msec MATLAB
7] LBREQM DD_X <BPM_NWL_61> (10 elements) 1000 msec MATLAB
9] LBREQM DD_X <BPM_NWL_75> (10 elements) 1000 msec MATLAB
0] LBREQM ADC_A <BPMF_NOR_63> (1024 elements) 1000 msec MATLAB
1] LBREQM DD_Y <BPM_NL_68> (10 elements) 1000 msec MATLAB
3] LBREQM DD_X <BPM_NWL_90> (10 elements) 1000 msec MATLAB
4] LBREQM ADC_B <BPMF_NOR_63> (1024 elements) 1000 msec MATLAB
6] LBREQM DD_X <BPM_NWL_104> (10 elements) 1000 msec MATLAB
7] LBREQM ADC_C <BPMF_NOR_63> (1024 elements) 1000 msec MATLAB
9] LBREQM DD_X <BPM_NWL_118> (10 elements) 1000 msec MATLAB
>[180] LBREQM DD_Y <BPM_NL_53> (10 elements) 1000 msec MATLAB
>[181] LBREQM ADC_D <BPMF_NOR_63> (1024 elements) 1000 msec MATLAB
>[183] LBREQM DD_X <BPM_NWL_133> (10 elements) 1000 msec MATLAB
>[184] LBREQM FLAGDISA <BPM_SWR_13> (227 elements) 100 msec PETRACON
>[185] LBREQM ADC_B <BPMF_NOR_67> (1024 elements) 1000 msec MATLAB
>[187] LBREQM DD_X <BPM_NL_140> (10 elements) 1000 msec MATLAB
>[188] LBREQM DD_Y <BPM_NL_36> (10 elements) 1000 msec MATLAB
>[189] LBREQM ADC_C <BPMF_NOR_67> (1024 elements) 1000 msec MATLAB
>[191] LBREQM DD_X <BPM_NL_126> (10 elements) 1000 msec MATLAB
>[192] LBREQM ADC_D <BPMF_NOR_67> (1024 elements) 1000 msec MATLAB
>[193] LBREQM DD_X <BPM_NL_111> (10 elements) 1000 msec MATLAB
>[194] LBREQM ADC_C <BPMF_NOR_70> (1024 elements) 1000 msec MATLAB
>[196] LBREQM DD_Y <BPM_NL_30> (10 elements) 1000 msec MATLAB
>[197] LBREQM DD_X <BPM_NL_97> (10 elements) 1000 msec MATLAB
>[198] LBREQM ADC_D <BPMF_NOR_70> (1024 elements) 1000 msec MATLAB
>[199] LBREQM ADC_B <BPMF_NOR_73> (1024 elements) 1000 msec MATLAB
>[200] LBREQM DD_X <BPM_NL_82> (10 elements) 1000 msec MATLAB
>[202] LBREQM ADC_C <BPMF_NOR_73> (1024 elements) 1000 msec MATLAB

```

Feb. 18: 466 contracts, 15 clients ...  
Just plain old 'Publish Subscribe' ...

# Transaction Coercion

- User-defined Structures
  - Requests for individual fields deliver the entire structure !
  - Reduce  $N_T$  !

Instant Client

Print... Options... Debug Tools... Monitor... Show Globals! Input Panel!

Device Context: PETRA Device Subsystem: ALL Show Stock Properties

Device Server: BunchScope Device Name: Bunch-1 Device Property: Trace.INFO

Data Size: 1 Data Type: STRUCT Description: [TraceHS] Trace Header Info Timeout: 1000

/PETRA/BunchScope/Bunch-1 Trace.INFO @ Sep 29 09:06:21.033

```
[ 0 -> DeviceName] Bunch-1
[ 0 -> DeviceDesc] Bunch/Trace Window
[ 0 -> DataFormat] 517
[ 0 -> ArraySize] 79
[ 0 -> preTrigger] 0
[ 0 -> ScaleX] 1E-10
[ 0 -> OffsetX] -4E-09
[ 0 -> UnitsX] seconds
[ 0 -> PlotMaxY] 2.099923E-02
[ 0 -> PlotMinY] -1.899923E-02
```

READ POLL draw mode Text dun transport udp (default) Autoscale Log Scale

Instant Client (Instance 1)

Print... Options... Debug Tools... Monitor... Input Panel!

Device Context: PETRA Device Subsystem: ALL Show Stock Properties

Device Server: BunchScope Device Name: Bunch-1 Device Property: Trace.INFO.ArraySize

Data Size: 1 Data Type: INT32 Description: [TraceHS] Trace Header Info Timeout: 1000

BunchScope/Bunch-1 Trace.INFO.ArraySize @ Sep 29 09:07:45.065

```
( 0 ) 79
```

Read POLL draw mode Text dun transport udp (default) Autoscale Log Scale

# Transaction Coercion

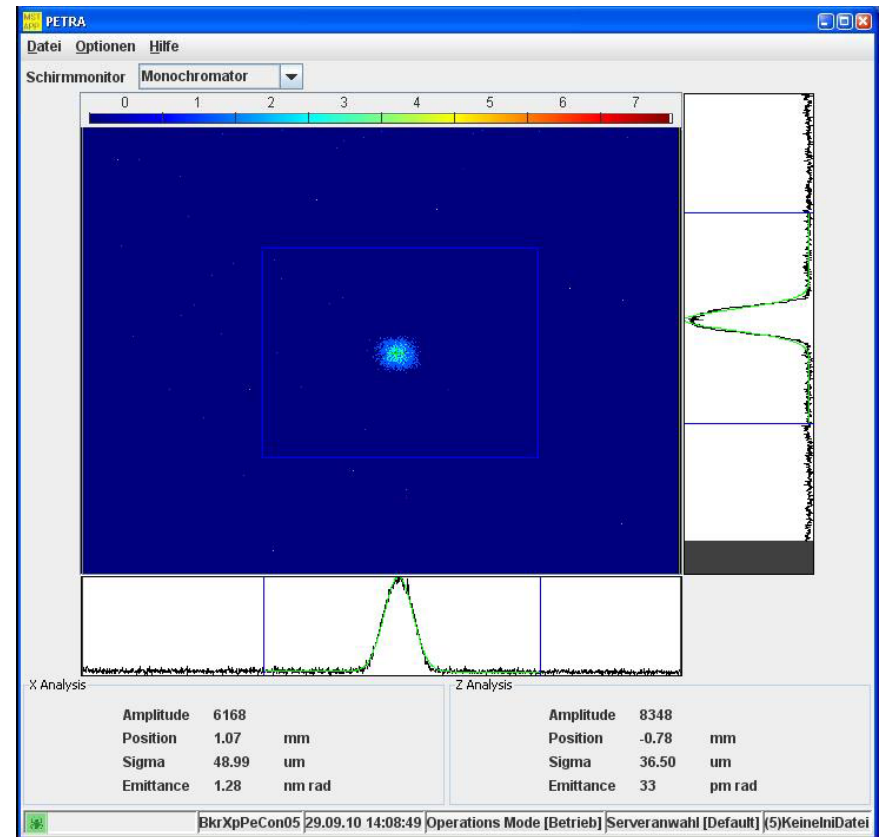
- Polling Intervals and Scheduling
  - Eager Clients:
    - Need to know something '*as soon as it happens*'
    - Sometimes use ridiculously high update rates.
  - Server can enforce a '*minimum polling interval*'!
    - Client is informed and makes an adjustment
  - Server '*schedules*' the data when it changes.
    - e.g. DESY2
      - cycles @ 6.25 Hz
      - clients attempt to update @ 10 or 20 Hz
      - Server establishes minimum polling rate @ 1 Hz but schedules requested data @ the 6.25 Hz hardware trigger rate
      - clients now really do get the data '*as soon as it happens*' !
  - Reduce  $U_T$  !

# Transaction Coercion

- Steering the Acquisition Mode
  - Properties can reject synchronous calls
    - Client can react and insert an asynchronous listener under synchronous looking API calls.
  - Properties can reject monitors
    - Client is informed that monitored data are static
  - Properties with large Payload can enforce multicast acquisition !
    - Monitors coerced into listening for multicasts.
    - Synchronous calls rejected.
  - Reduce  $N_T$  and  $N_C$  !

# Transaction Coercion

- PETRA3 Video Example:
  - Mixed Gbit/100-Mbit Net
  - Large images (large  $P_T$ )
- Tune the transport Parameters !
  - Enforce multicast
  - Minimum polling interval
  - Redirect non-Scheduled Property to Scheduled Property
- $N_T = 1, N_C = 1$





# A Server takes control of its Clients

Example: doing 1 thing for 1 effective client instead of 600 things for 10

## Client

- Give me property "Pressure" for pump "OL146.2"
- Ok then, monitor "Pressure" for pump "OL146.2"
- Ok then, monitor "Pressure" for all pumps
- Ok then, I'll listen for the multicast

## Server

- No! You'll have to monitor this !
- No! You'll have to monitor the entire MCA
- Ok, but I'm going to multicast it!

# Conclusion

- New Data Flow meme: Contract-Coercion
  - Can provide high scalability (without API restrictions)
- Yes, you can ...
  - Run epics2tine
  - Run tango2tine
  - Use doocs (which has TINE embedded)
  - Use stars which has a TINE bridge
- <http://tine.desy.de>
- Email: [tine@desy.de](mailto:tine@desy.de)

