NIFS　EPICSセミナー

2024/09/09(月) ～ 09/13(金)

はじめに： 目的

KEKにようこそ!

- 今回はEPICSセミナーという形でNIFSのみなさんと交流できることはとても喜ばしいです
- お互いに「良かったな」と思える機会になると期待しています（少しでも）

※ 今回知り合えたことで、それだけで十分な成果!?

- 次はNIFSに伺って現場を見ながら、色々なお話が出来ればと考えています

- 過去の資料を含め、色々と準備はしていますが間に合っていない部分も多くあります。ご容赦ください
- 記録を取りながら進めることで、今回の資料がそのまま次の講習会テキストになると考えています
- 疑問点などあれば、どんどんフィードバックしたいです
- NIFSではどのようにやっているのかにも興味があります

- このような講習会のとき、いつも問題になるのは個人の経験が大きく異なることです
  - 例：Linux経験者 or not
  - 今回は人数も少ないため、相談しながら進めたいと考えています

主な担当者

- KEK:
  - PF: 濁川、帯名、路川徹也、塩澤真未
  - KEKB: 佐々木信也、杉村仁志
  - J-PARC: 山本昇、山田秀衛
  - Linac: 佐藤政則

- NIFS:
  - 前野、鷹見
  - 渋谷、佐藤

超高温プラズマのミクロ集団現象と核融合科学
HLD,CHS

Timetable案

進行状況・要望に応じて変えていきます。順番を変える可能性もあり。

Date		from	to	Item	担当	コメント・目標
9月9日	月	13:30	14:00	概要説明・準備	濁川、帯名	セットアップ、vnc接続など
		14:00	15:00	EPICS概要、コマンドライ	帯名	細かいことはおいておいて、触ってみる
		15:00	15:30	OPI(CSS)	山本	コマンドでやったことをOPIで(RPiのCSSを使って)
		15:30	17:00	database concept	山本	IOC, データベースの考え方 ※翌日分との干渉あり
9月10日	火	9:00	10:00	CSS OPI	山田	参加者PCにCSSインストール。AAはサバ太郎。
		10:00	11:30	database基礎	佐々木	softIoc
		11:30	12:00	makeBaseApp	佐々木	makeBaseApp
		12:00	13:00	昼食		
		13:00	14:00	RPi Hardware I/O 1	帯名	LED/GPIO
		14:00	15:00	NIFS制御の現状	NIFS	
		15:00	15:30	KEKB見学	佐々木	
		15:30	17:00	CSS+AA連携	山田	時間が余れば(?)AA構築をここで。参加者の技量に依存す
9月11日	水	9:00	12:00	Sequencer + α	佐々木／山田／路川	Sqeuencerのほか、AA構築（参加者持ち込みPC）をここ
		12:00	13:00	昼食		
		13:00	14:30	RPi Hardware I/O 2	帯名	割り込み
		14:30	17:00	Stream Device	路川	virtual device 使用
9月12日	木	9:00	17:00	東海見学		
				iBNCT見学		
				中央制御棟見学		
				MRトンネル見学		
				MR電源棟見学		
9月13日	金	9:00	10:30	RPi実習		
		10:30	12:00	質疑・その他トピックス		

→ 次ページでコメント

金曜日午前中

- 希望に応じた実習のほか、ディスカッション時間を取りたいと考えています
- 実習内容は要望に応じて対応可能です
- 実習アイテム候補

CA Gateway
Autosave
Sequencer
CA Security
Python IOC（PCAS; Portable Channel Access Server）
Script Interface: pyepics
Alarm
Machine Learning
Jupyter Notebook
Camera画像の表示/CSS
waveform表示/CSS
OPI builder (Phoebus CSS)
Grafana GUI
LabView連携
横河PLCの例
I2Cセンサー接続
Linux Basics

私の話

1. PF制御室見学
2. 加速器制御システムの目指すものとは
3. 制御フレームワークEPICSの紹介
4. 実習：EPICSクライアントとしての使用

その後、山本さんよりEPICS database

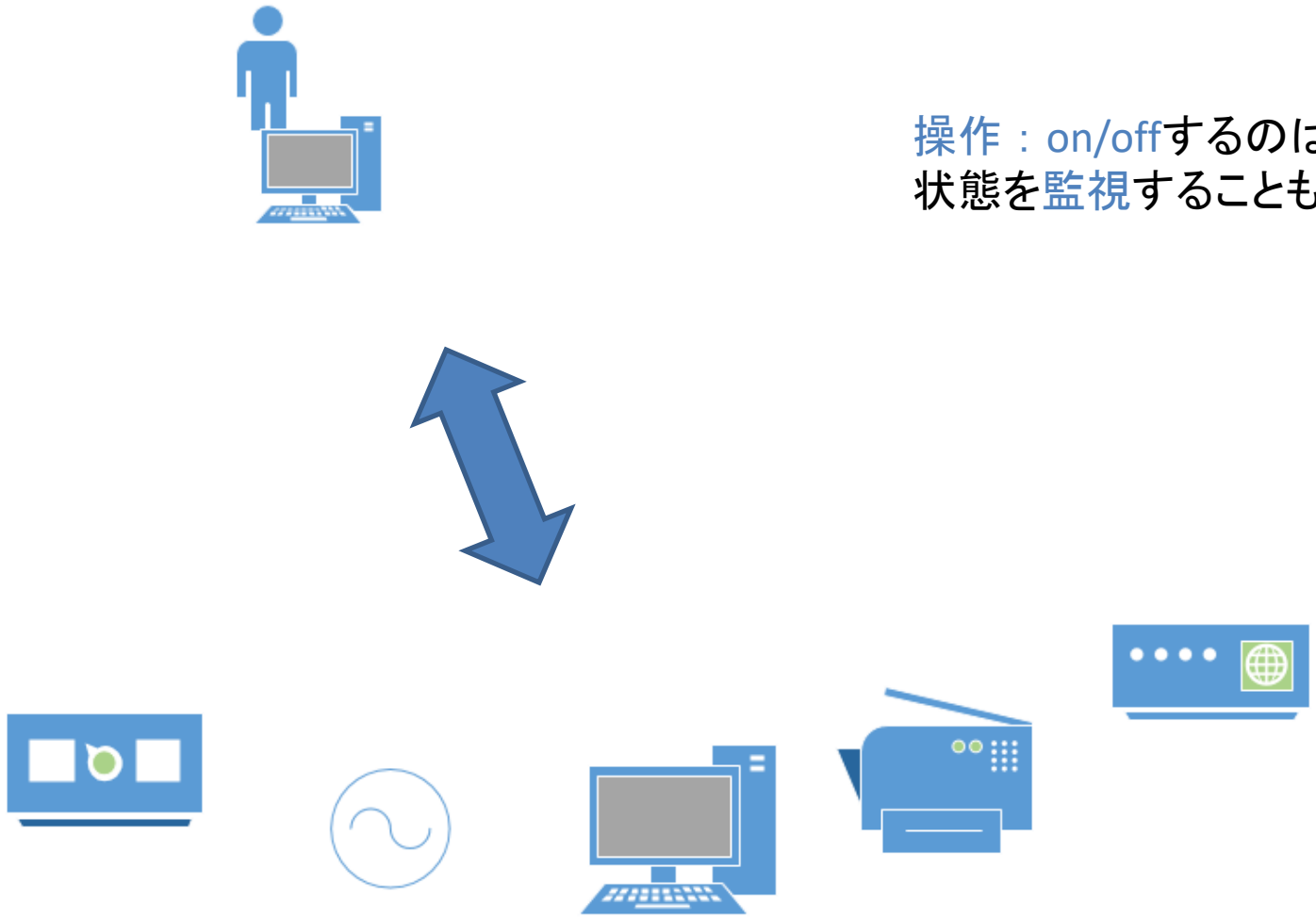
Linuxが「はじめて」という場合はRaspberryPiを使って簡単な実習をおこないます。

加速器制御概論／EPICS 入門

KEK, High Energy Accelerator Research Organization
帯名 崇
(takashi.obina@kek.jp)

加速器制御システムの目指すものとは？

加速器に限らず機器制御に必要なことはたくさんの装置を遠隔から操作すること



操作：on/offするのはもちろんのこと
状態を監視することも含む

再掲：ネットワーク

各種機器をつなぐためのネットワーク：最近はTCP/IP(Ethernet)が簡単

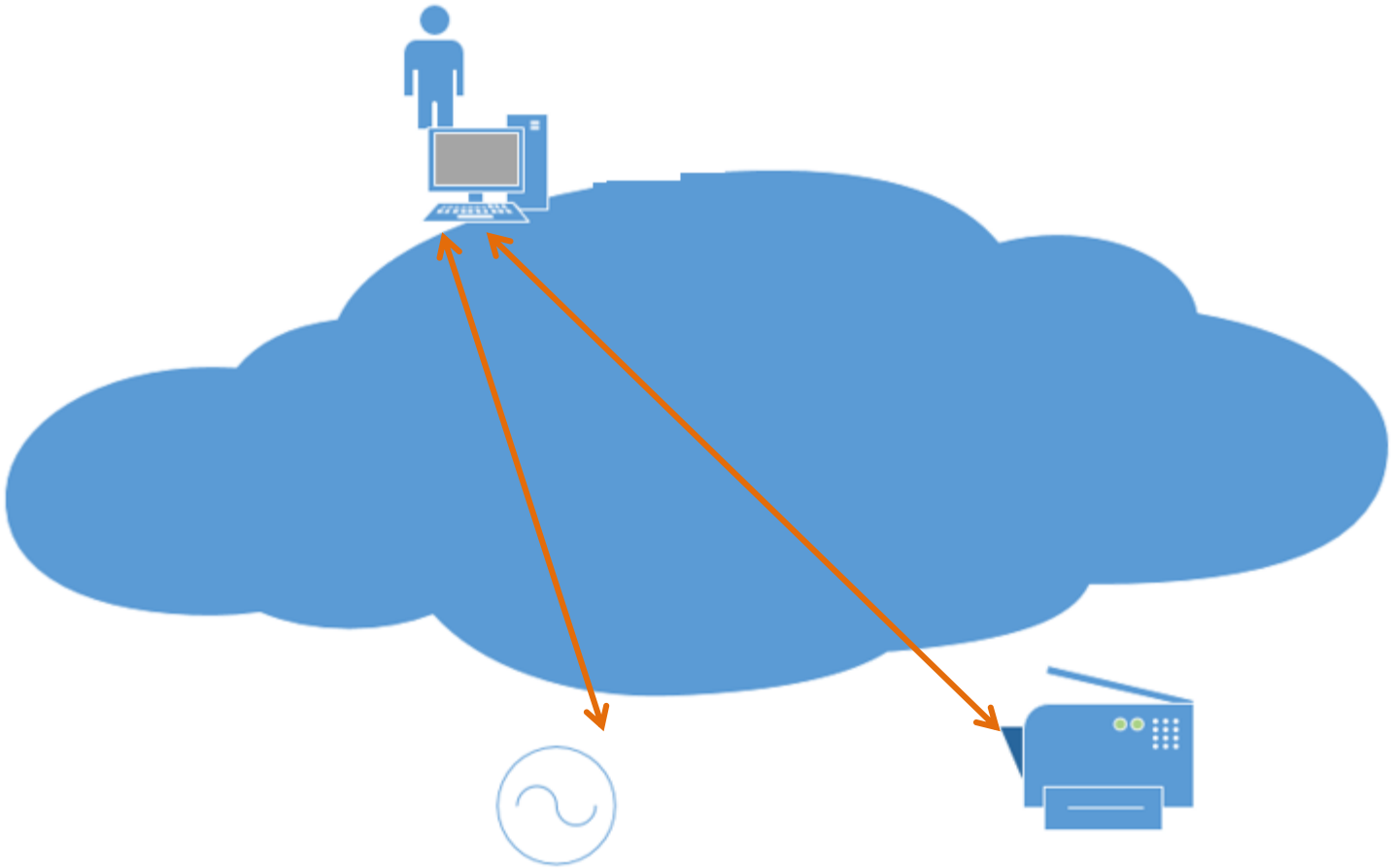
- 他にも色々なネットワークが存在します。目的に応じて適切なモノを選定することが必要

詳細は「OSI参照モデル」とか「産業用イーサネット」「フィールドバス」等のキーワードで調べてください

小規模なシステム（実験室など）

対象機器が少ない場合は....正直言って、何でも良い

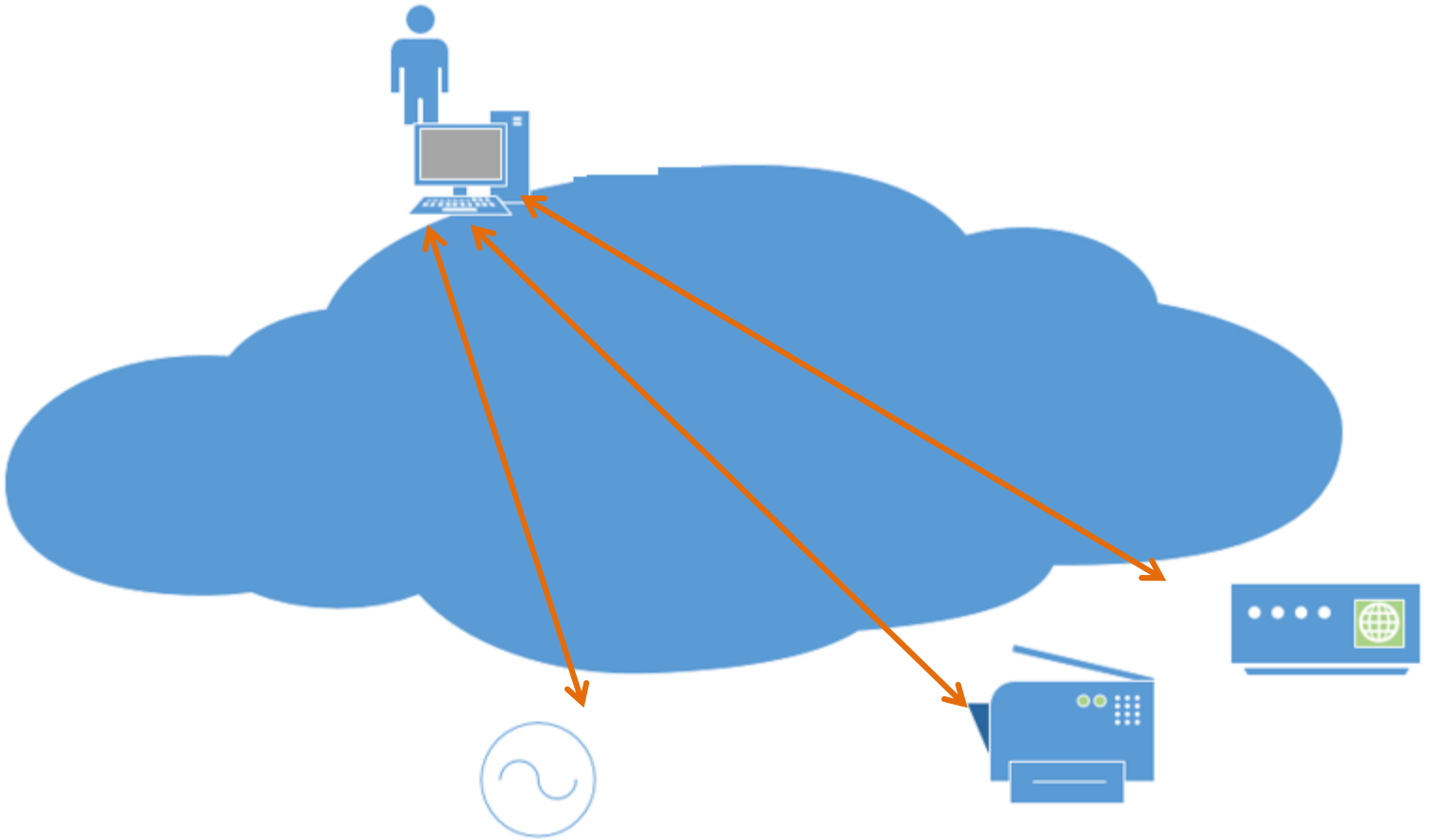
開発者＋使用者が「完全に１人」ならば、好きにやればよい



たとえ1人であっても

測定器が1個増えた

…　…まぁ、プログラム追加すれば良いだけでしょう

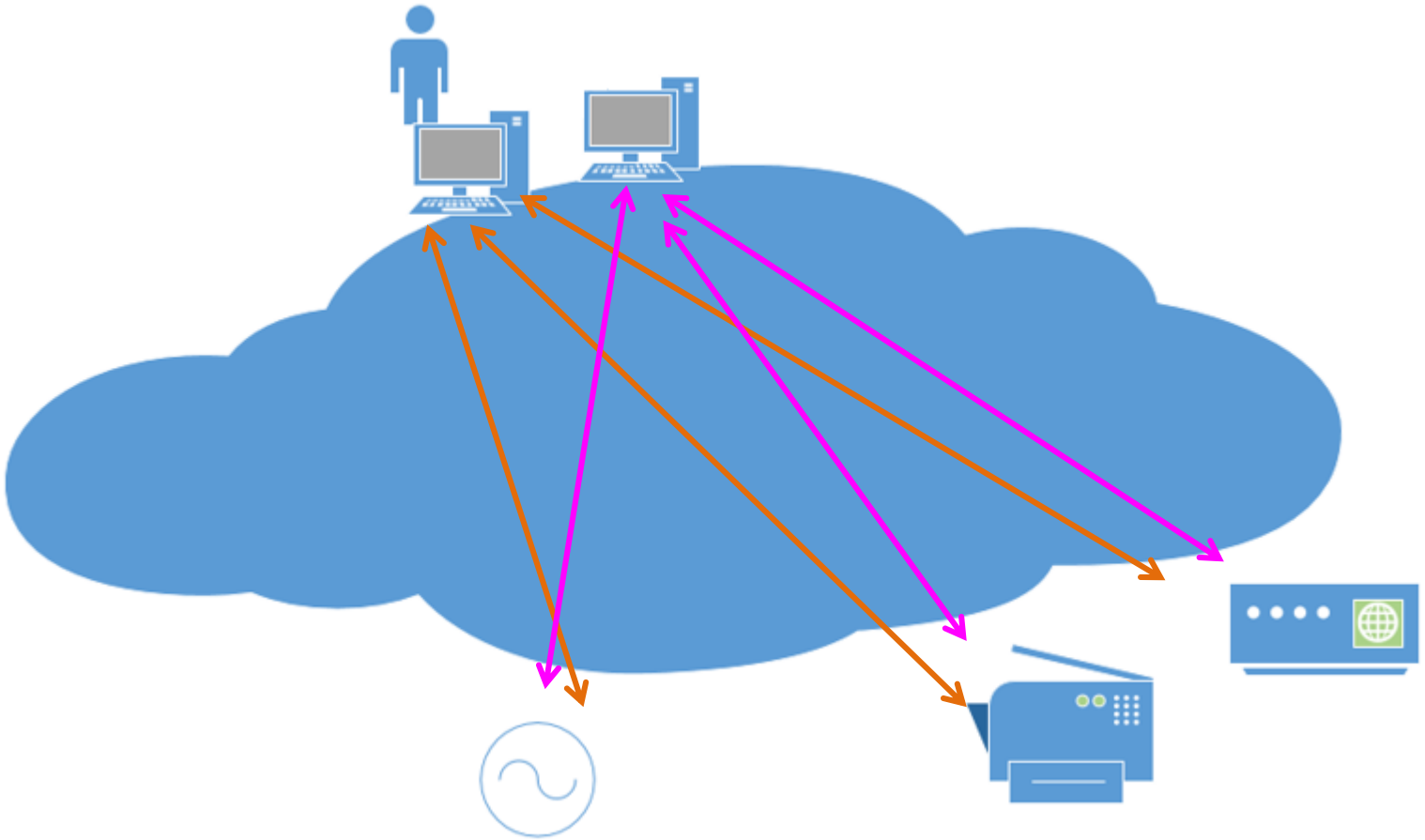


新しい制御用PCを導入した

完全に置き換えるなら簡単だけど、両方から操作したい。

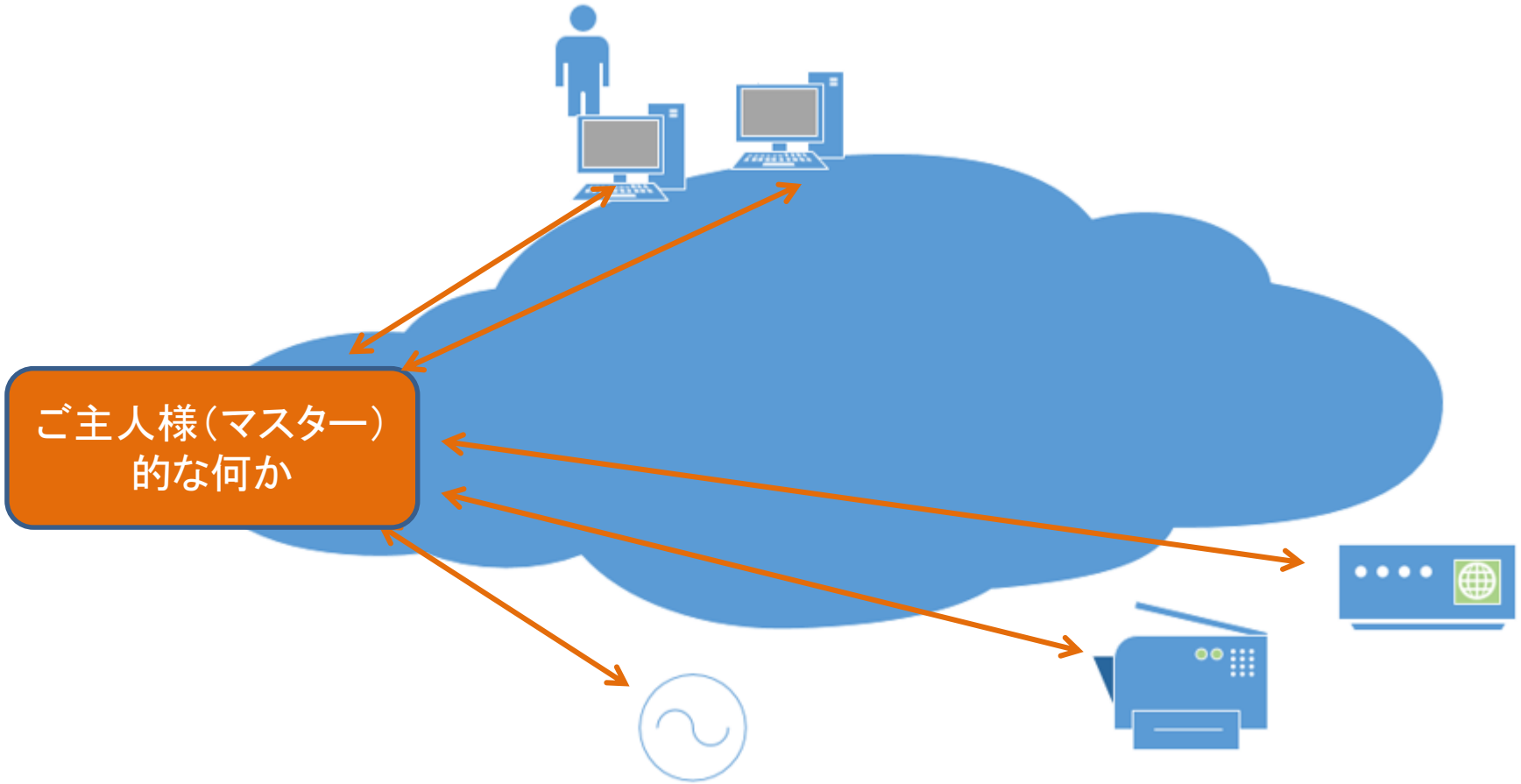
排他制御はどうする？

データファイルはどっちに保存した？



すべてを中継する何かが居れば良いのでは？

実はそれなりに実在するシステム。しかしマスターが死んだら全滅するのはちょっと怖い。
（single point of failure の存在）

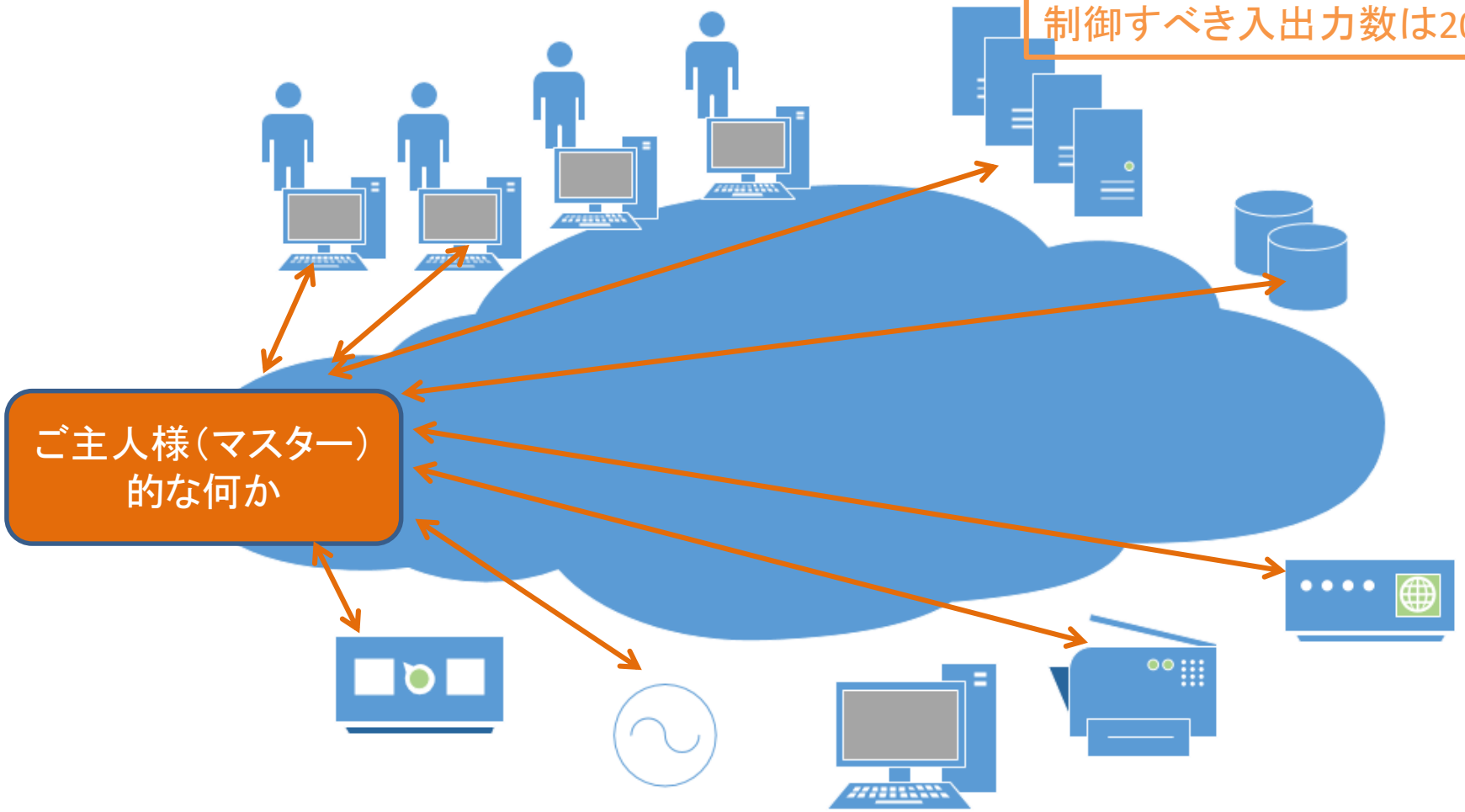


もっと機器や人が増えると？

数10～数100個とかのスケールならなんとかなる？

もっと増えて10,000個なら大丈夫なのだろうか？

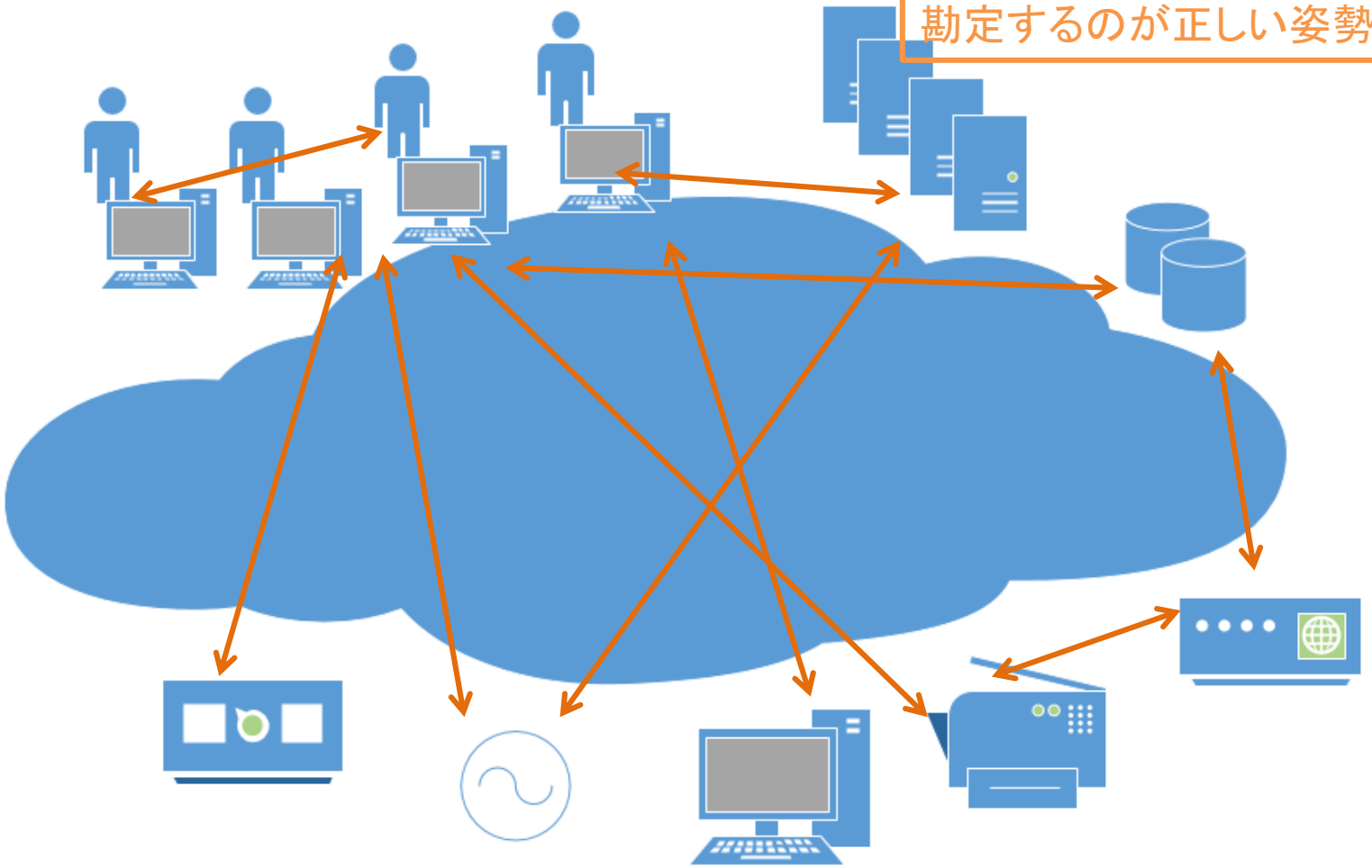
例えばSuperKEKBの場合、制御機器は1万台
制御すべき入出力数は20万個以上に達する



むしろ分散制御した方が良さそう

機器がたくさん(3～4個以上)ある/関係者が複数(2名以上)/制御プログラムが複数など場合は各要素が有機的な連携することが望ましい。

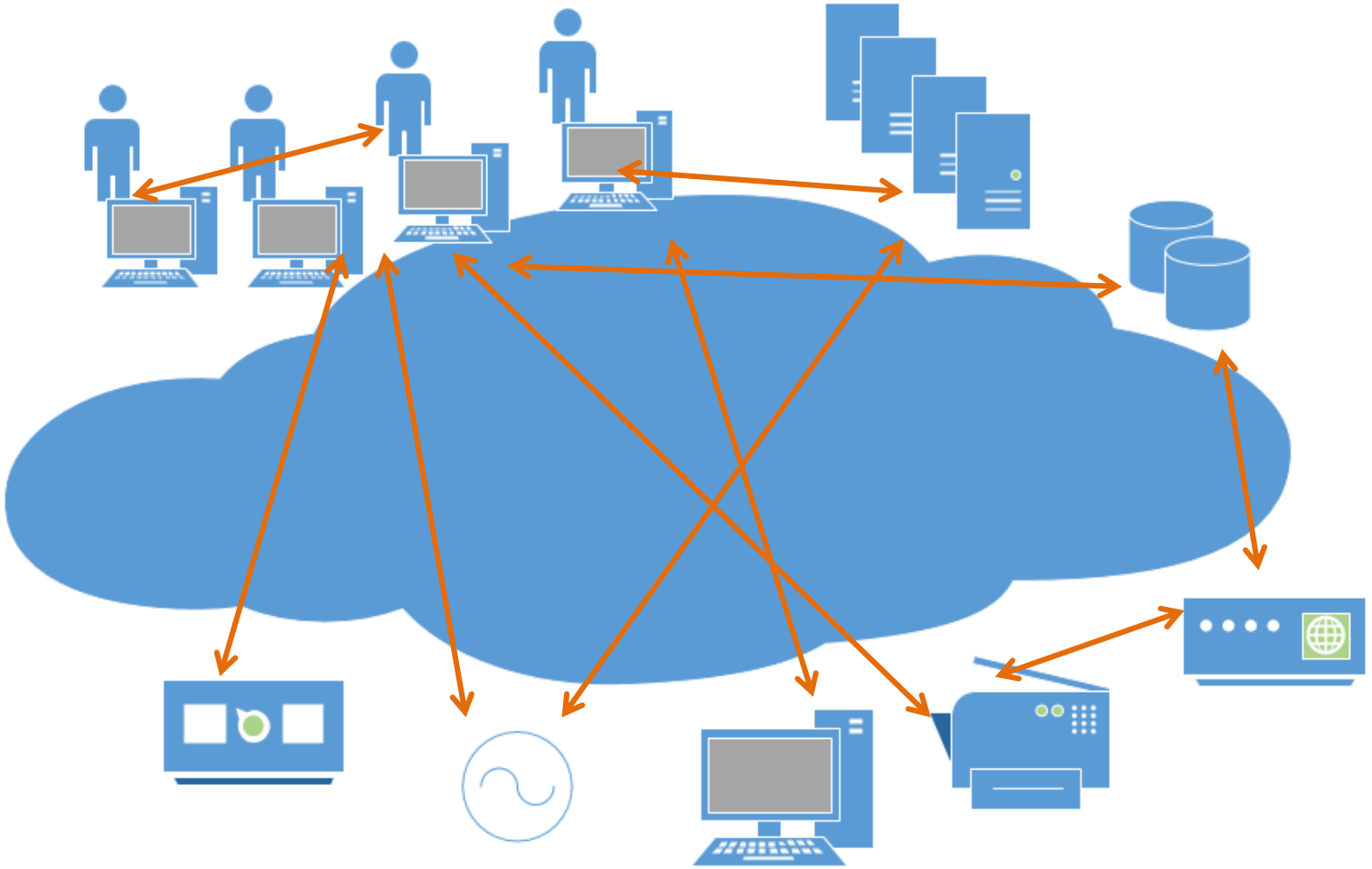
余談になるが、1か月後の自分は他人として勘定するのが正しい姿勢である



制御フレームワーク

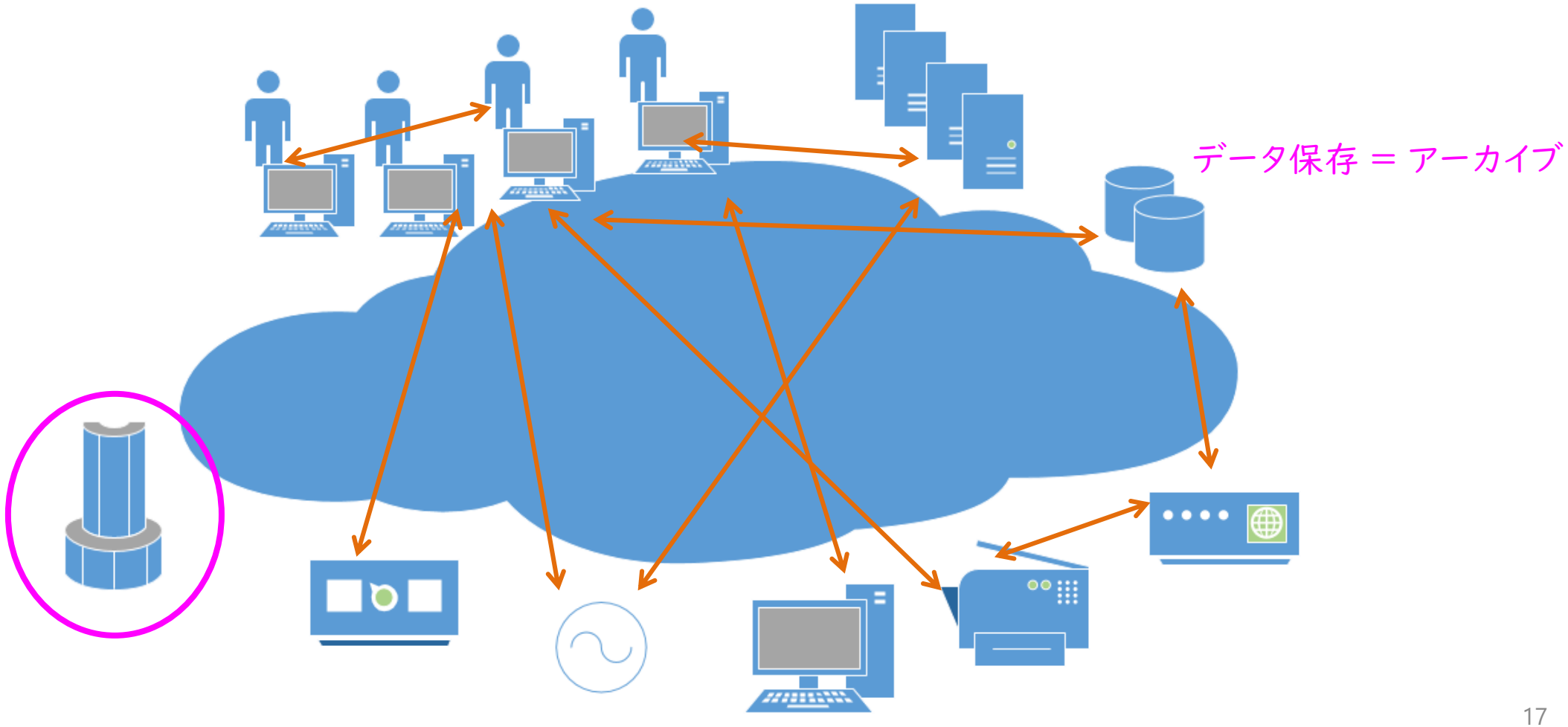
そのとき共通の「言葉」「お約束」でお話することが必要

➡ EPICSはそのような環境を構築するフレームワークです



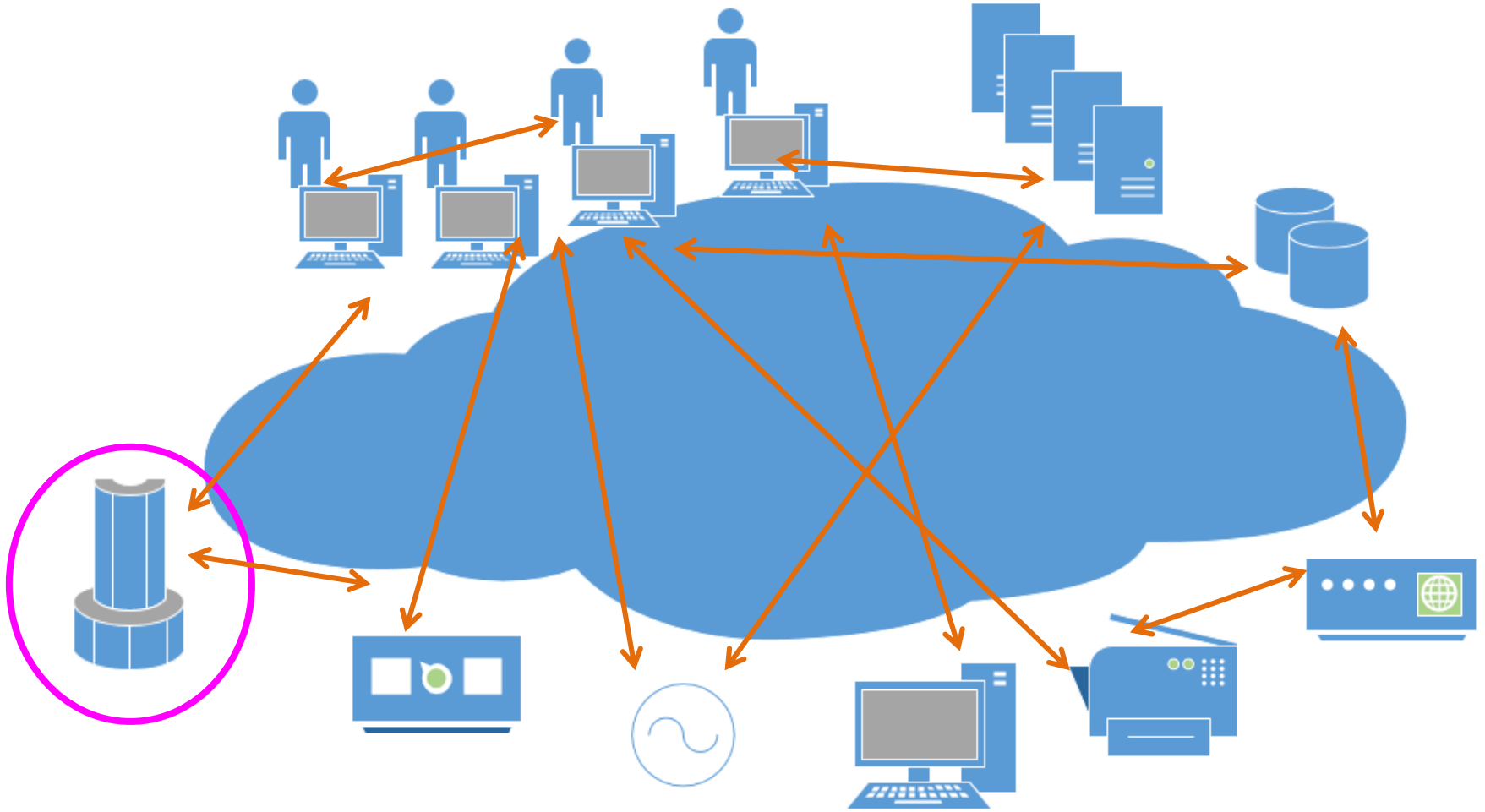
新しい装置を導入したき

もしその機器が独自通信しかできなければ、既存の装置と連携することができない。機器の状態もわからなければ共通のアーカイブにデータを保存することもできなくなってしまう。



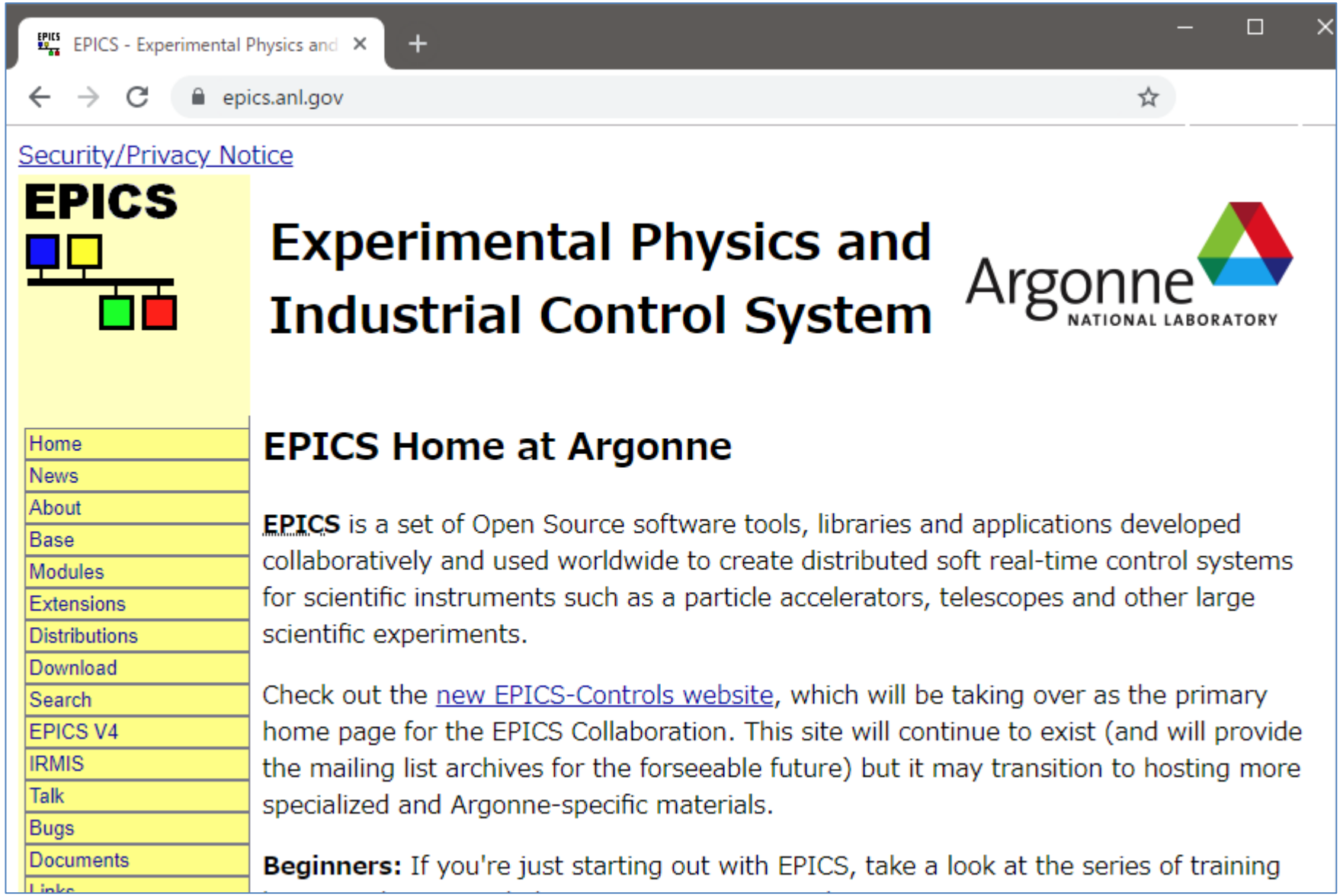
そんなときには

新しい装置が共通の言葉を理解でき、話せるようにすることが必要です



そのような仕組みの１つがEPICS です

https://epics.anl.gov/



EPICSとは何か

Experimental Physics and Industrial Control System
https://epics.anl.gov/
https://epics-controls.org/

特徴としては
- Open Source：国際協力研究開発
- ネットワーク分散環境
- 制御アプリケーション開発フレームワーク

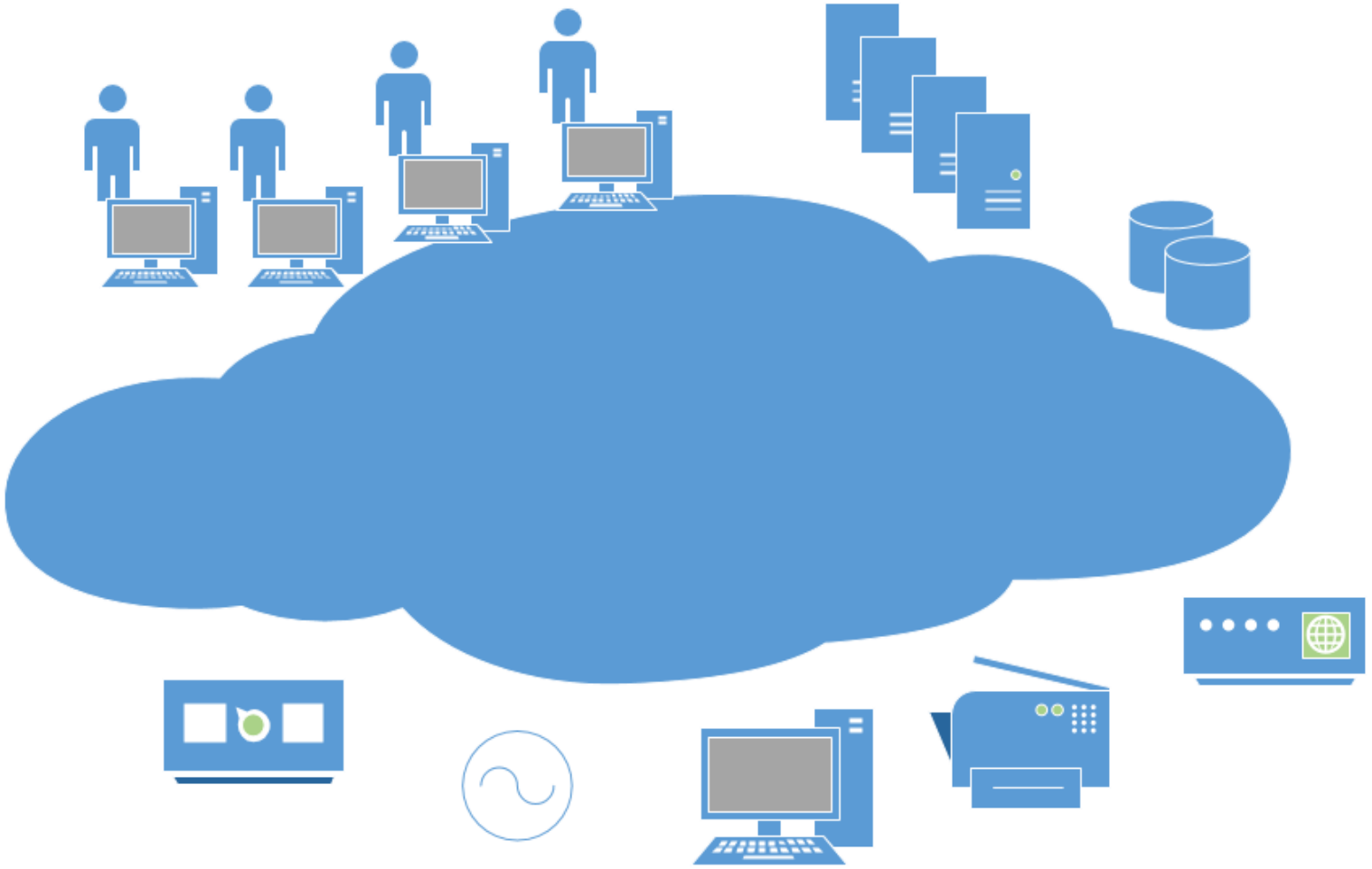
世界中の多くの加速器で使われています。加速器以外でも
- LIGO 重力波
- 天文台
- ITER 核融合炉（建設中）

その他、たくさんの施設で使われています

他にも加速器制御フレームワークは存在します。それぞれに特徴あり。
- TANGO, DOOCS, Tine(DESY), PVSS(CERN), MADOCA(SPing8) など
- PFでのビームライン制御にはSTARSなども使われています

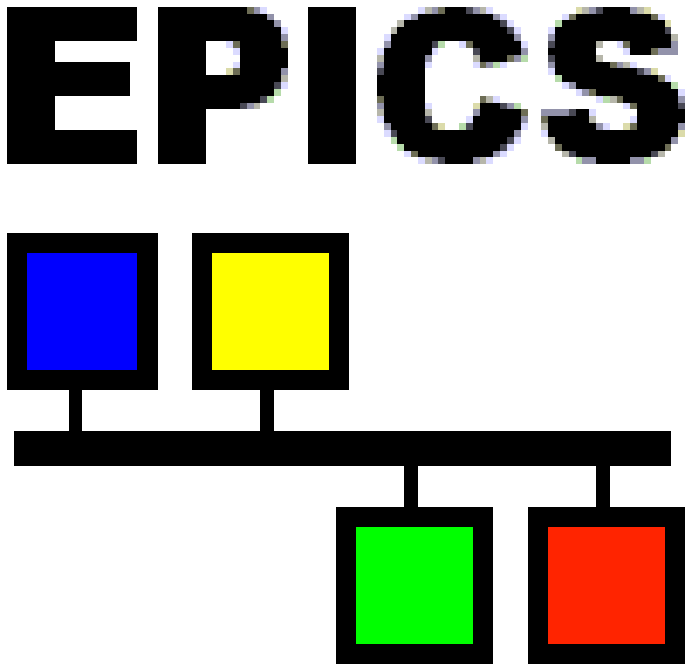
次に

EPICSとはどのようなものか、もう少し詳しく説明していきます。
最初に説明した絵を少しだけ思い出してください



EPICS のロゴがあらわしているもの

制御系を「階層化」して表しています。
用語として「下位」層ほどハードウェア寄りになり、「上位」と言ったら人間寄りの層を指します



写真（小型電磁石電源の例）

- 軌道＝電磁石＝電磁石電源

制御室

計算機、Archiver

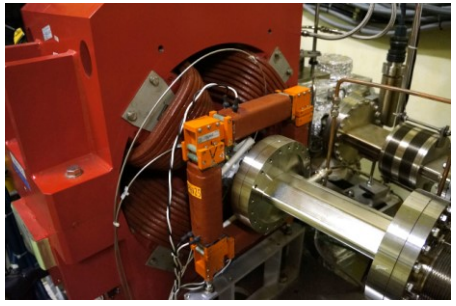
Network

大型磁石(B,Q,S, etc)

小型電源

VME筐体

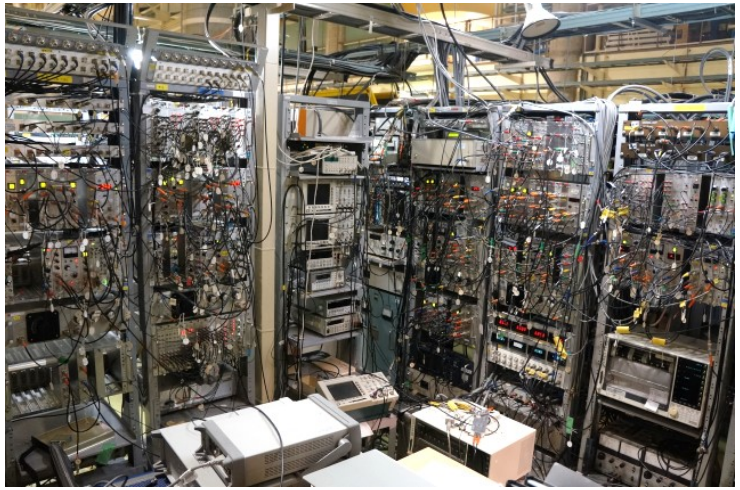
小型磁石
(補正用)

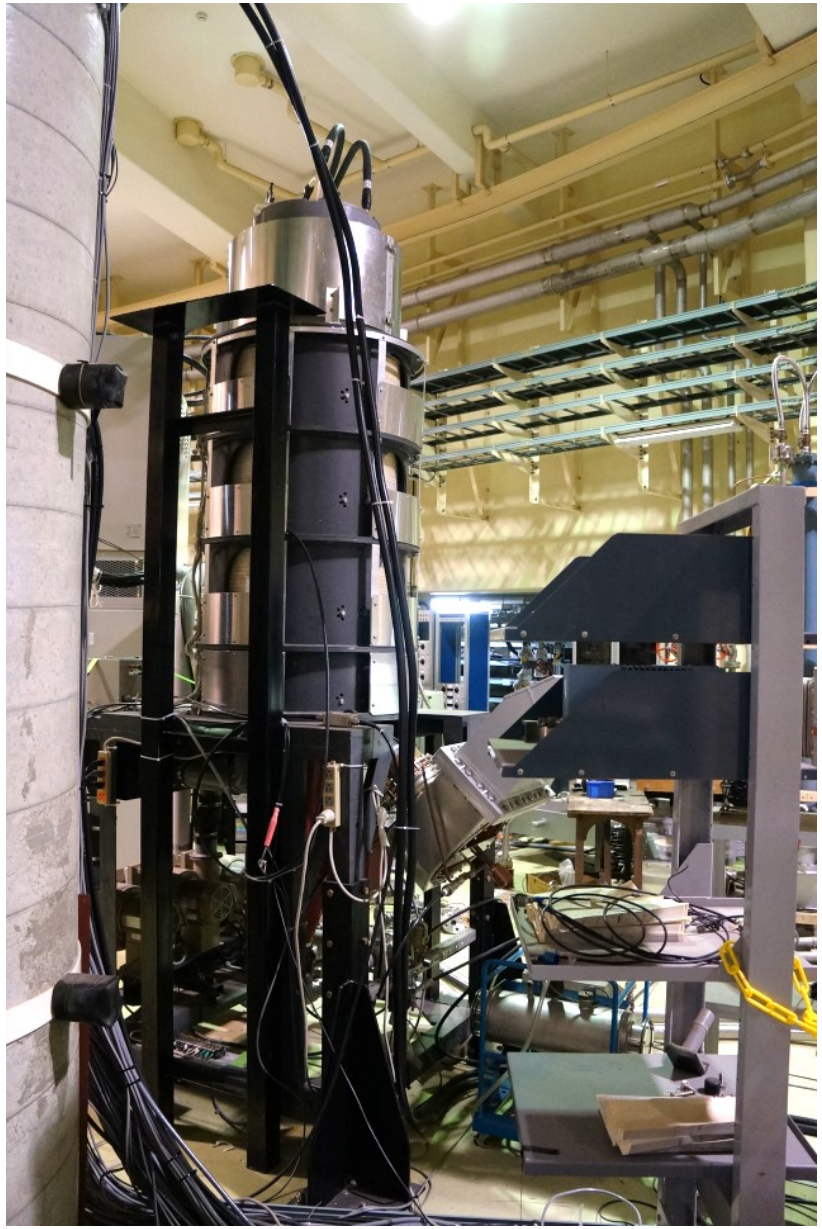


写真：RF関係

RF（高周波）

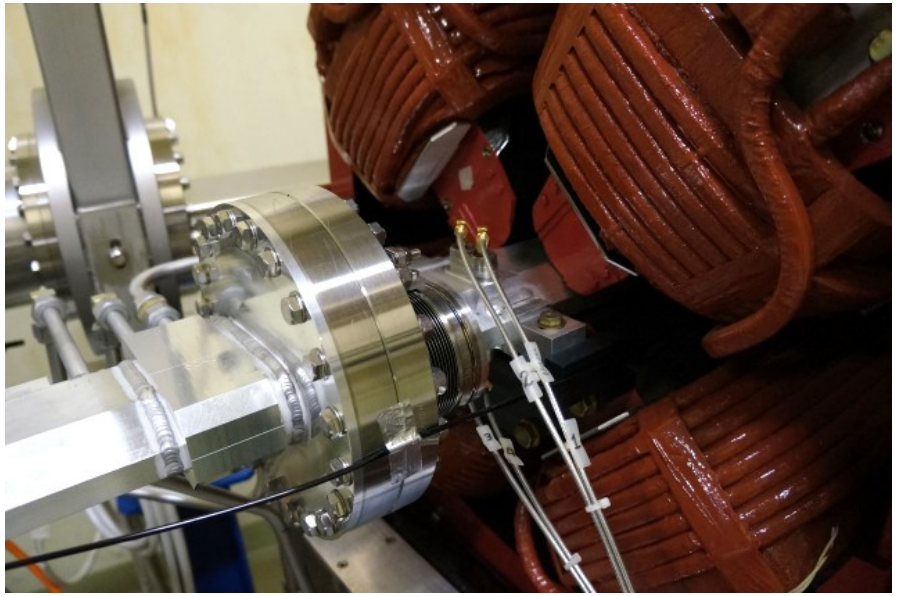
- LowLevel回路
- 高出力クライストロン
- 立体回路（伝送系）
- 加速空洞

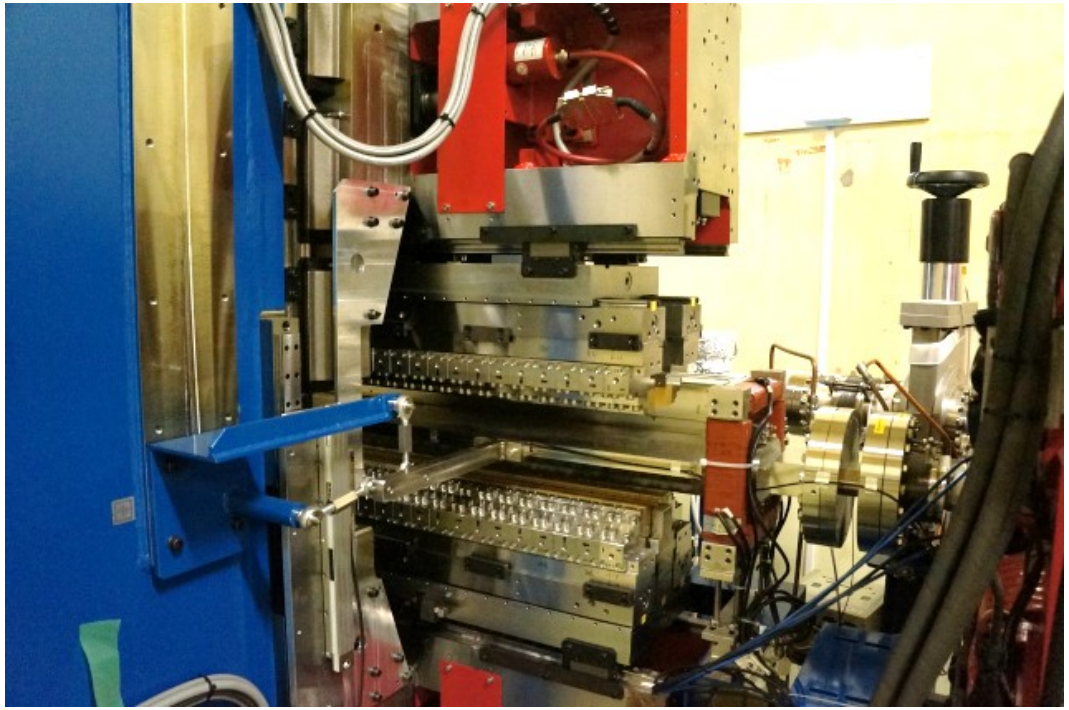




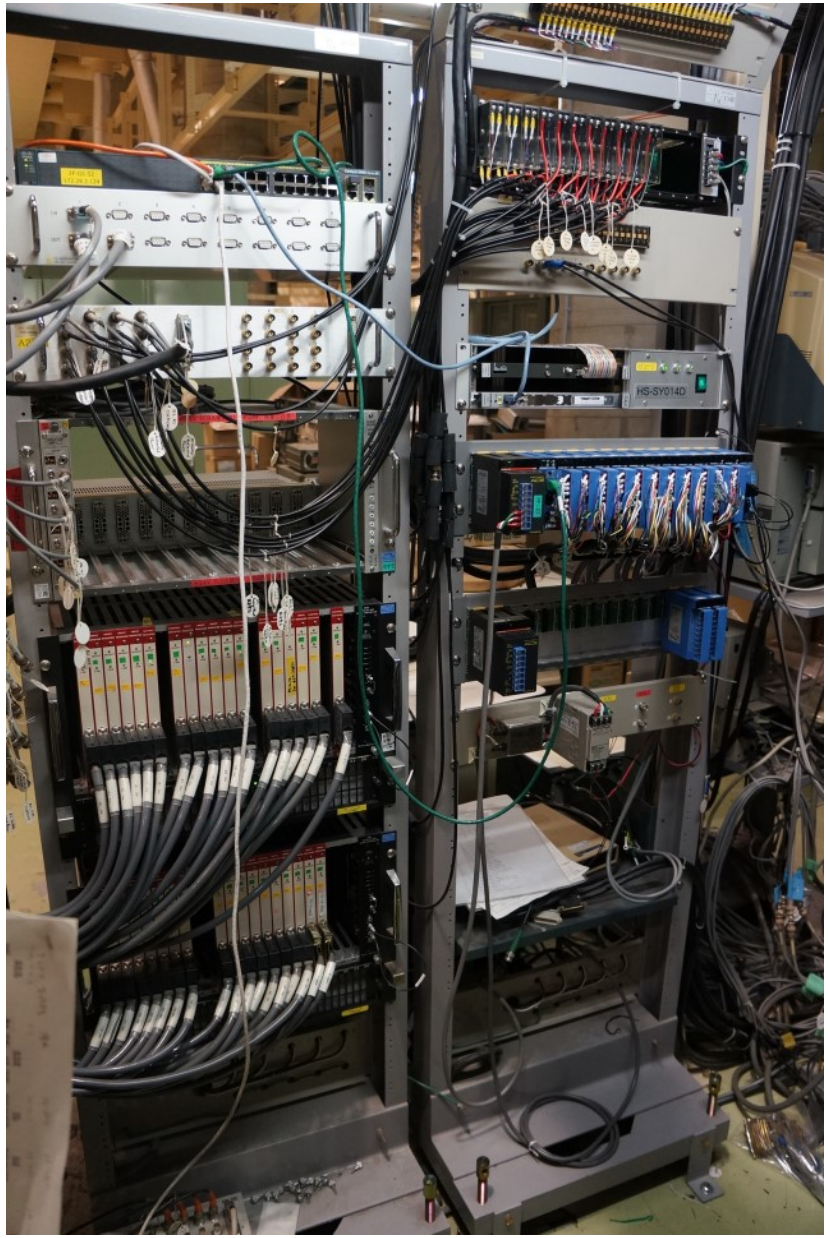
写真：モニター、真空、挿入光源ほか

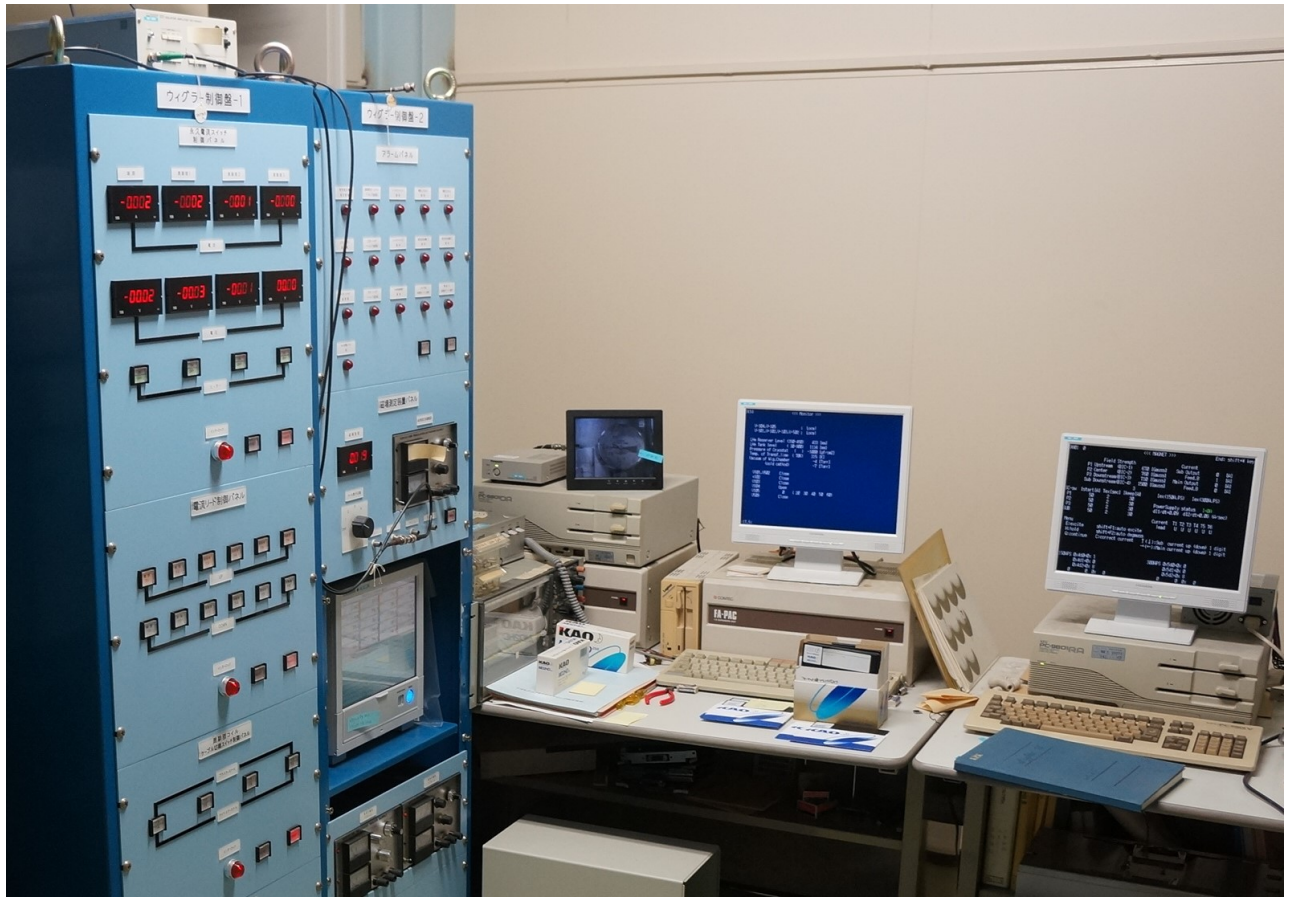
- Monitor
- 真空
- 挿入光源





古いモノも使いつつ、更新している（RF制御、ウィグラー制御など）

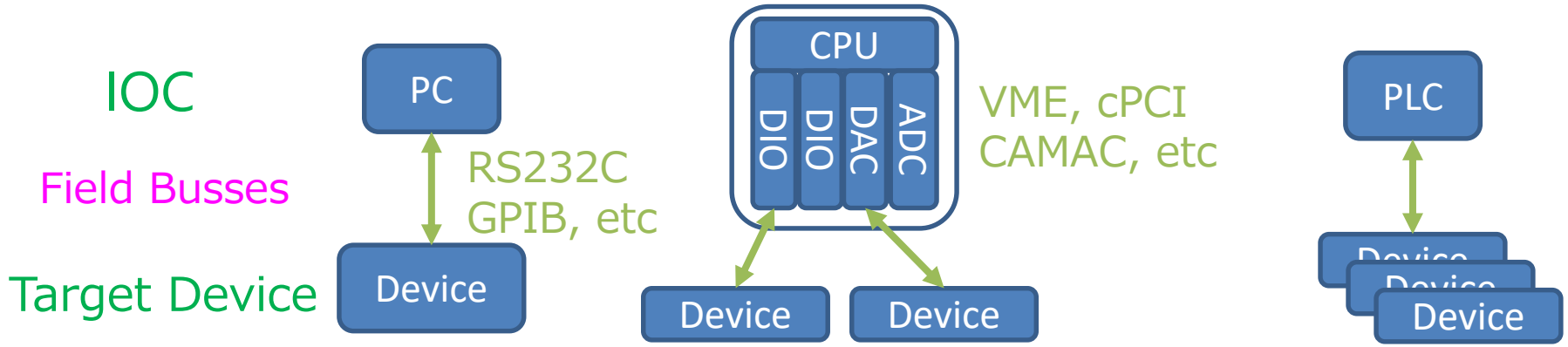




デバイス層 ～ 入出力コントローラー層

IOC (Input/Output Controller) がハードウェアを制御

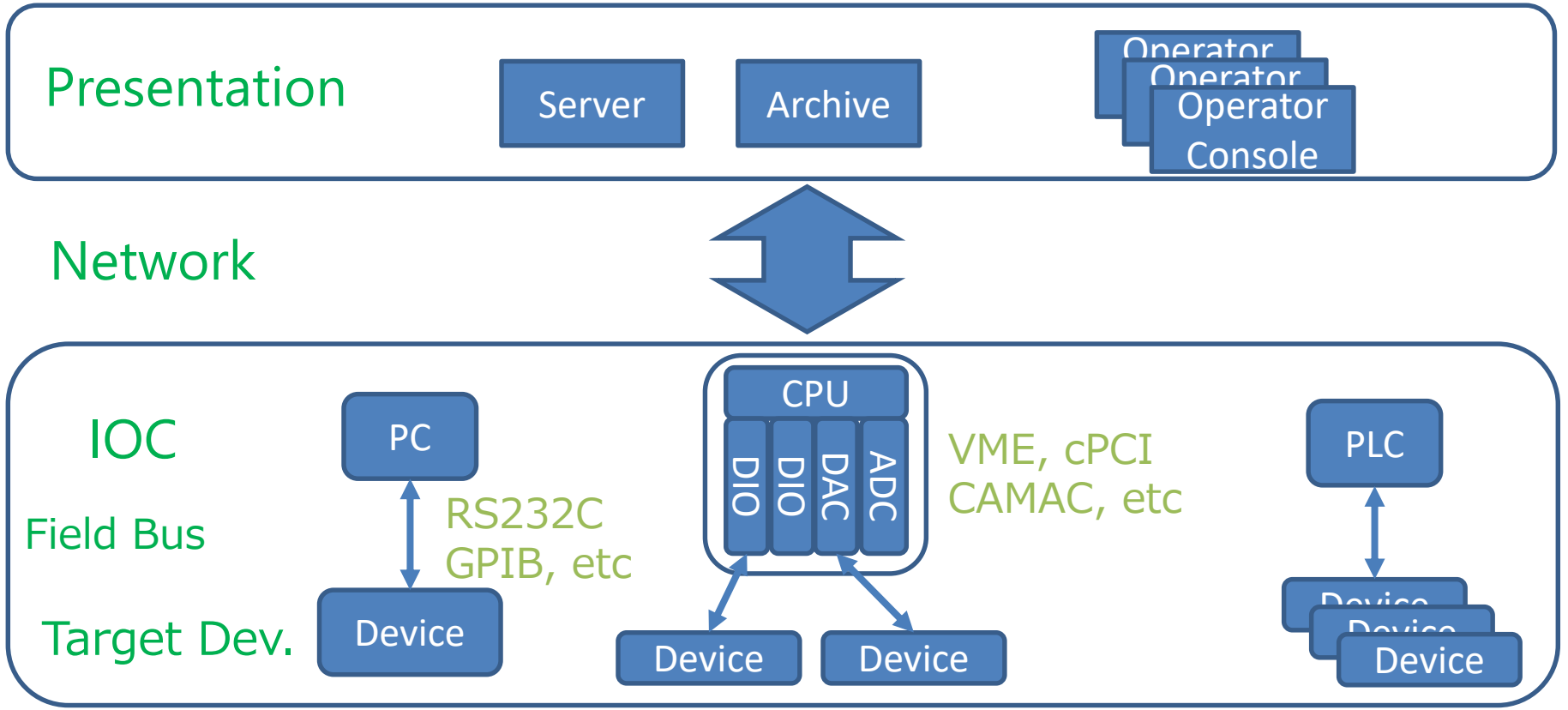
- 加速器を動かすには様々なデバイスが必要：電源、センサー類、高周波回路、etc
- デバイスに応じた制御プログラムが必要
- IOCは機器を制御するCPUをもった計算機で構成することが多い： VME, PLC, CAMAC, PC(パソコン), Raspberry Pi などなど…
- 目的に応じて、適切なOSやハードウェアを選択し、制御したいデバイスと接続する（接続の仕方は色々ある）



Communication Protocol

IOC と上位層との通信に標準的なプロトコルが 必要

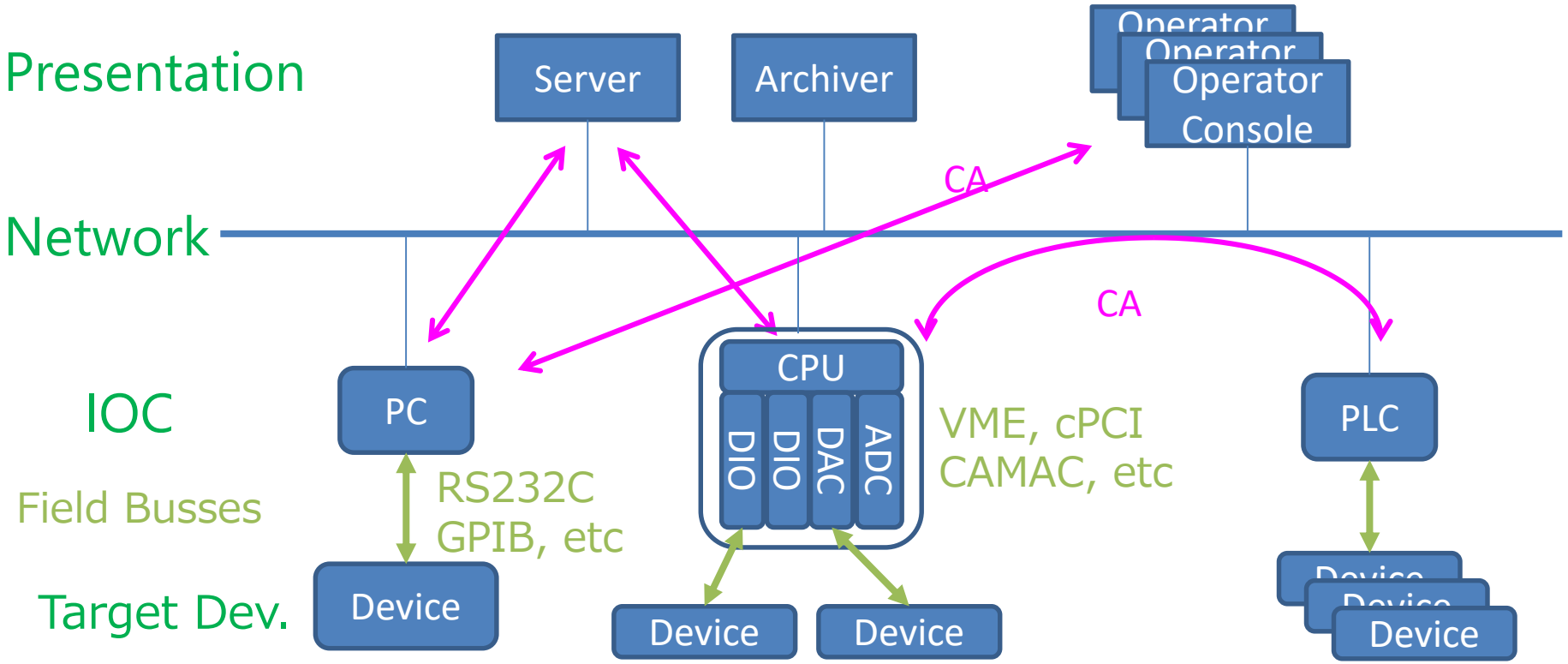
- Presentation Layer : Server process, Archiver, GUI, Alarm, etc
- 各デバイス制御とは独立にしたい。



通信プロトコル

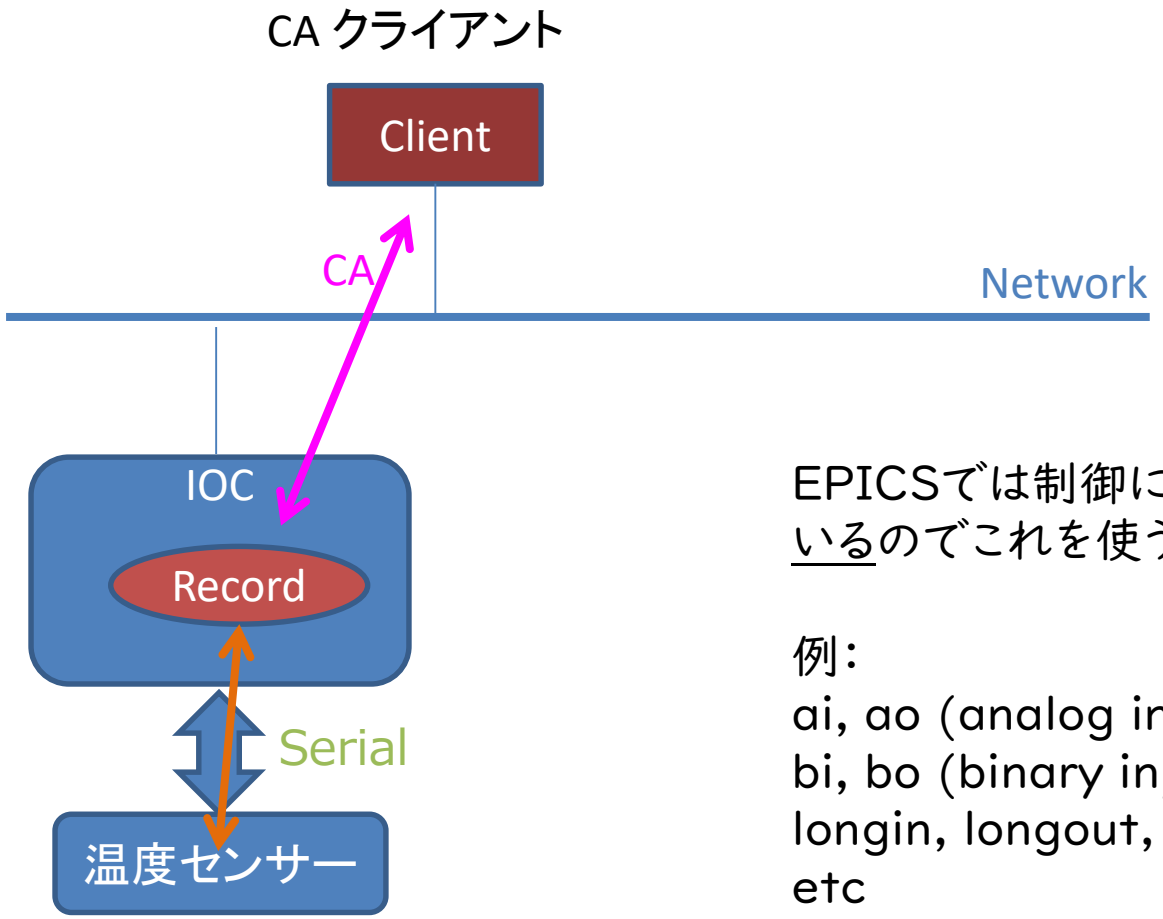
Channel Access protocol で通信

- Network transparent. Distributed system.
- 上位層との通信だけではなく、IOC間の通信でも使用する
- IOCの中にある状態変数(Process Variable, PV)を外から制御



レコード

IOCの中には「レコード」があって、別のホストから（CAプロトコルを使って）読んだり書いたりできる。「レコード」はハードウェアにつながっていたり、純粋なソフトウェアだったり、様々。

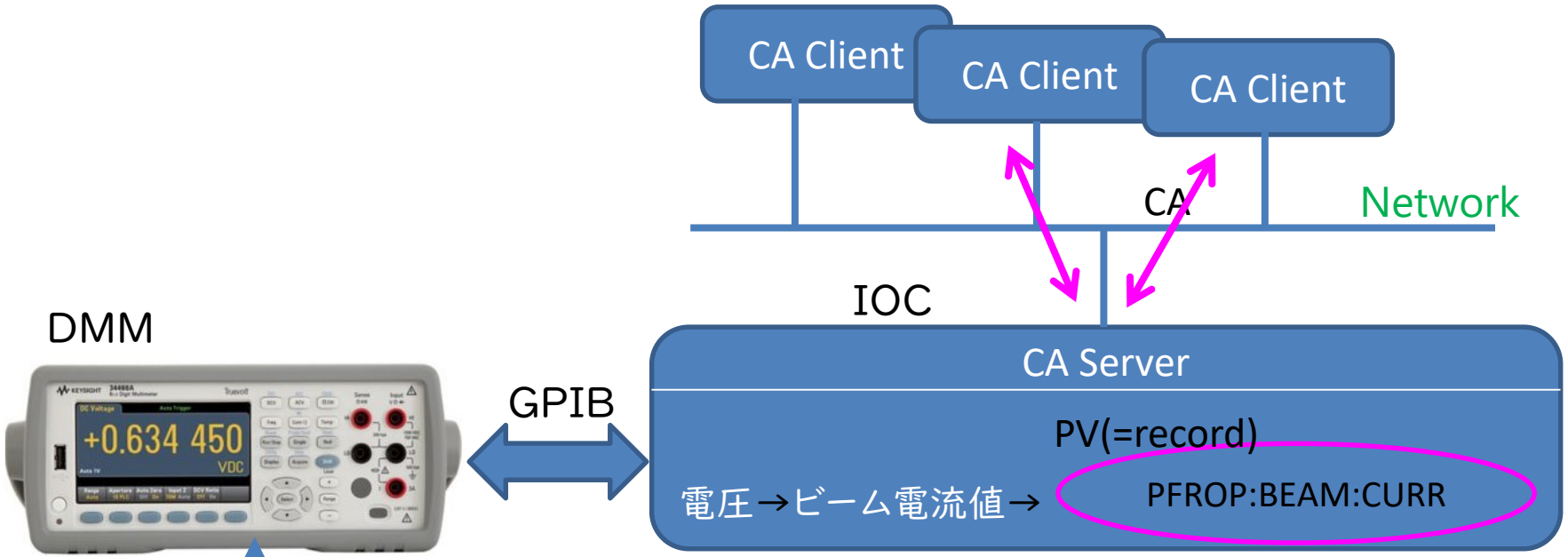


EPICSでは制御に使うレコード型が用意されているのでこれを使うと大部分はカバー出来る

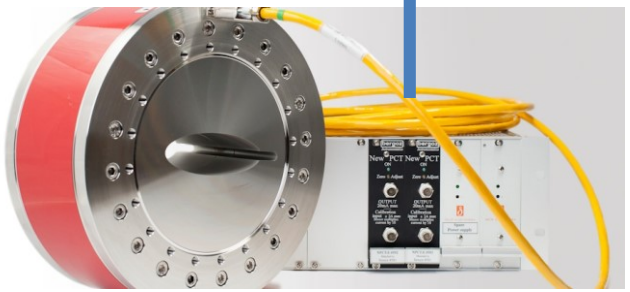
例：
ai, ao (analog in/out)
bi, bo (binary in/out)
longin, longout, waveform, calc, fanout, etc

具体例

下は加速器内のビーム電流 PFROP:BEAM:CURR というレコード例。
EPICSではこの名前(=レコード名)さえ知っていればネットワークを意識せず(透過で)通信できる。



電圧〇〇[V]が△△[mA]に相当する、という変換係数や単位情報を持つ



加速器内のDCCT(直流電流トランス)

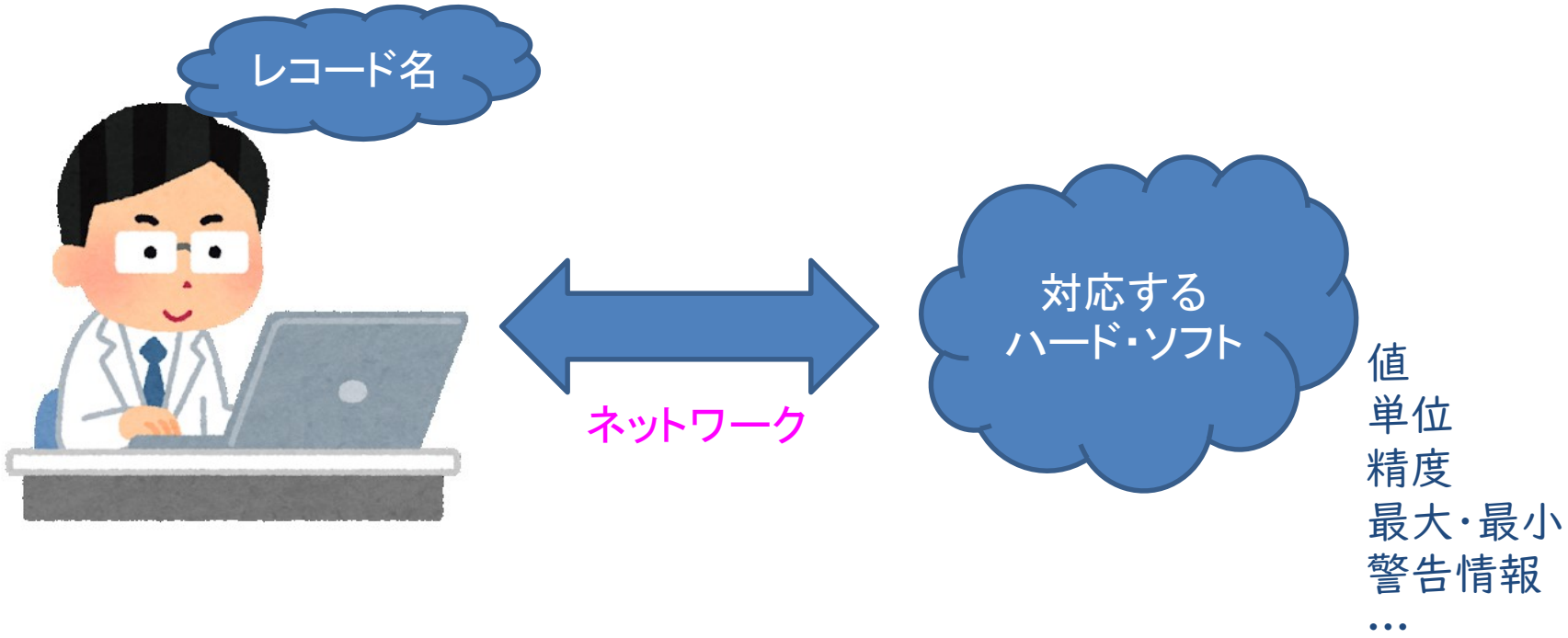
いまは細かいことは置いておいて

イメージとして

EPICSでは「レコード名」だけわかっていれば

- ネットワークや、ハードウェアを意識せずに制御できる
- レコードは、値だけではなく他にいろいろな情報（単位、精度、etc）を持っている

という2点を意識しておいてください



次は実習（デモ）です。まずはネットワーク越しに値をとってくることに挑戦

講習で準備してあるレコード名

今回の実習で用意しているレコード名一覧

ハードウェアにつながっているレコード

ソフトウェアのみ

```
ET_DEMO:USBTHERM1:MON     # USB温度センサーの値 (anglog, read-only)

ET_DEMO:LED_RED:ON        # 赤色LED binary output (on/off 設定可)
ET_DEMO:LED_BLU:ON        # 青色LED binary output (on/off 設定可)
ET_DEMO:LED_GRN:ON        # 緑色LED binary output (on/off 設定可)
ET_DEMO:LED_WHT:ON        # 白色LED binary output (on/off 設定可)
ET_DEMO:LED_YEL:ON        # 黄色LED binary output (on/off 設定可)
ET_DEMO:TOGGLESW:STAT     # トグルスイッチ (on/off, read-only)

ET_DEMO:aiExample         # 0 - 9 のカウント値で1秒ごとに増加 (Soft)
ET_DEMO:jane              # random number    (0.1秒ごとに更新, Soft)
ET_DEMO:fred              # random number    (2秒ごとに更新, Soft)
ET_DEMO:alan              # random waveform (2秒ごとに更新, Soft)

ET_DEMO:SOFTAO:CH01       # software record (-50 ～ +50 の値を設定可能)
ET_DEMO:SOFTAO:CH02
   ....           |      #
ET_DEMO:SOFTAO:CH30
```

レコード名の付け方は自由です（実際の運用ではルールがある方が良い）。ETは「EPICS Training」の略。

実習：EPICS Cient

1. RPiで「ターミナル」を開く or ssh（ターミナルエミュレータ）で RPi にログイン

2. サンプルレコードのうち、どれでも良いので caget してみる
   - レコードの値は読めた？

3. caget で温度センサーの値を読んでみる
   - 値は変化するか？

4. camonitor で温度センサーの値を読む
   - 温度は読めた？
   - センサーに触ってみる
   - camonitor を止めるには CTRL-C を入力

実習の続き

5. 値を設定する
   - caput で LED の on/off

6. お好きなLEDのレコードに caput してみてください

7. caget のオプションを確認
   - cagetにはいろいろなオプションがあります
   - caget -h でヘルプ表示
   - トグルスイッチやLEDの on/off 状態を読んでください。このとき caget で取得したり、caget -n で取得してみてください。

8. camonitor で複数のレコードをモニターしてみてください
   - 例えば、jane と fred を同時にモニター

※ camonitor はとても便利（かつ強力）なツール。値が変わったかどうかをクライアントから問い合わせる必要が無く、値が変わったときに通知が来て表示している。ネットワーク負荷の低減に有効。

時間の余っている人向けのトピックス

9. caget -a オプションをつけるとどうなるか
10. レコードの属性（フィールド）をcagetしてみる
    - EGU, HIGH, HIHI, DRVH
11. caput でLEDのON/OFFを制御するとき
    - ON/OFF でputする
    - 数値（1/0）でputする
12. waveform record (alan)をcagetする。
13. soft ao レコード（ET_DEMO:SOFTAO:CHxx など）にcaput で数値を設定する。このとき、DRVH 以上,DRVL 以下の値が設定できないことを確認。合わせてDRVH, DRVL の値を確認。その後、DRVH をcaput で設定して同様に動作確認をしてみる
14. camonitor で微少な変化には応答しないようにしたい：MDEL

クライアント GUI

ここまではコマンドラインでのデータ取得＆設定を行った。
やはり Graphical User Interface (GUI) も必要

EPICS Collaboration によって、多くの GUI Toolkit がある。その中でもよくつかわれるものとしては

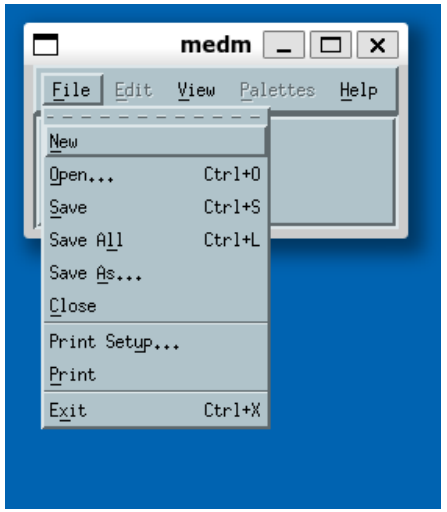
- MEDM (Motif Editor and Display Manager)
- CSS (Control System Studio) : GUIも含めた統合開発環境
- スクリプト言語+Tk や Qt など
  - Python/Tk, Qt, Wx, etc
  - SAD/Tk
- Windows ならば Visual Studio も使用可能

※ 実習：MEDM or CSSで簡単なGUIを作成する

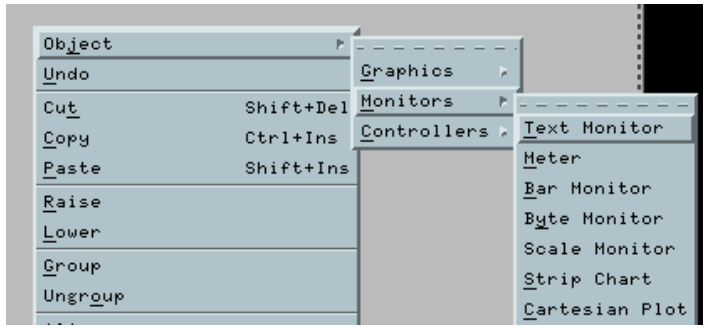
medmの簡単な使い方

1. 起動は”medm”
2. フォントが少し汚いが、今回はちょっと我慢。(きちんと設定すれば scalable font が使える)
3. File → New で空のウィンドウが出る
4. ウィンドウで右クリックすると、メニューが出るので配置したいモノを選ぶ。
   - Object → Monitors → Text Monitor を選択
5. カーソルが「+」になるので、左クリックしてドラッグすると「サイズ」が決まる
6. Readback Channel にレコード名を書いて「Enter」押下
7. 「Execute」を押すと実行状態になり、値のモニターが始まる

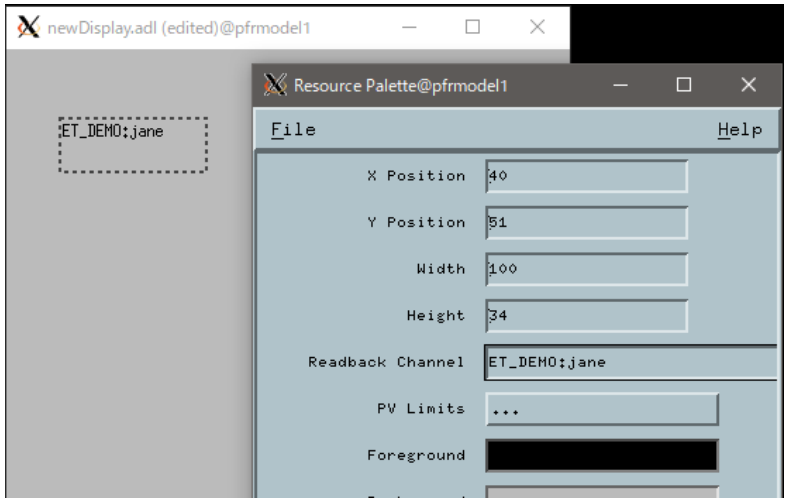
3



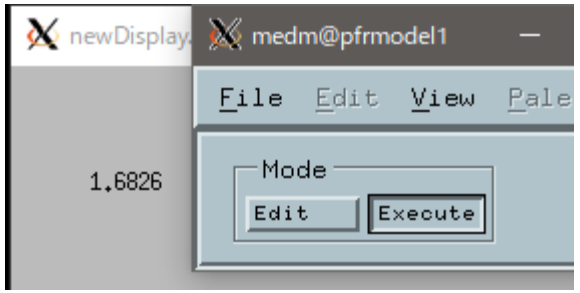
4



5 6

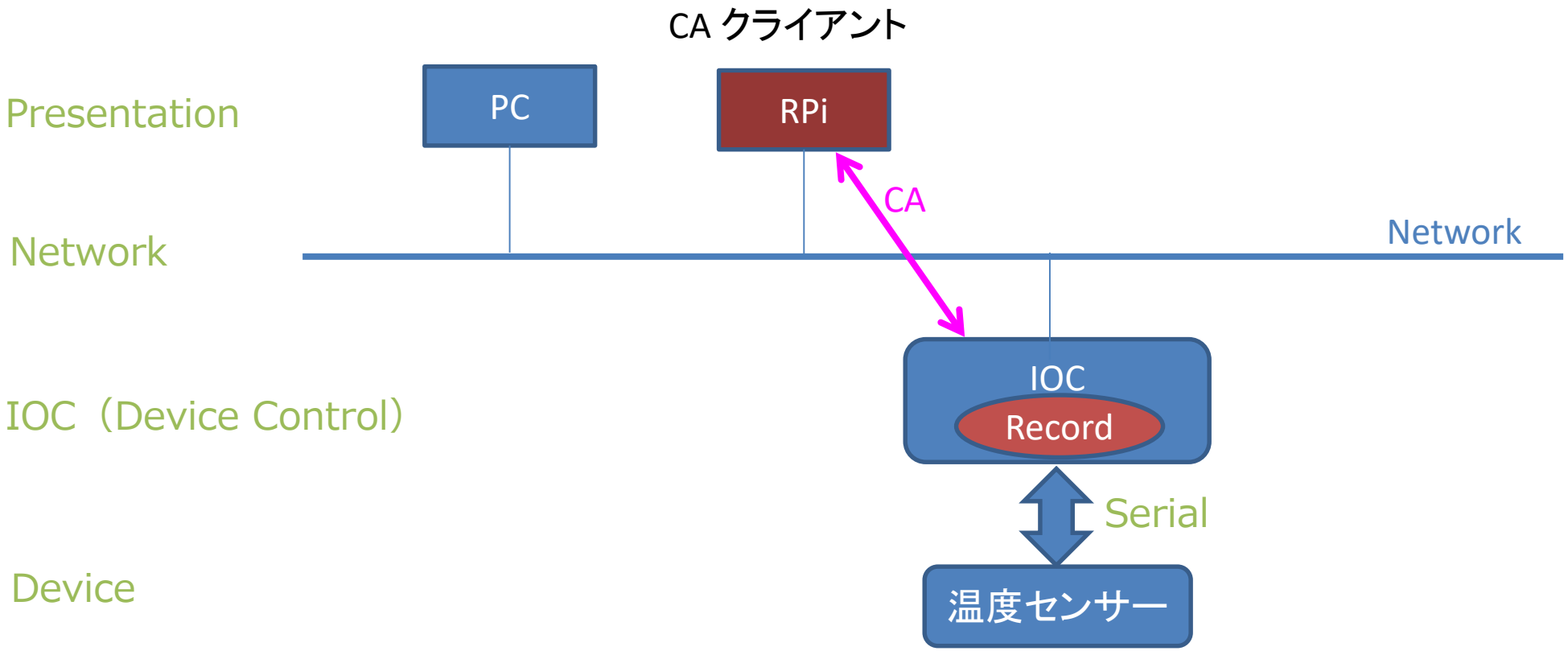


7



この段階でやったこと

Channel Access protocol で通信して、値を取ってきた
Raspberry Pi が CAクライアント

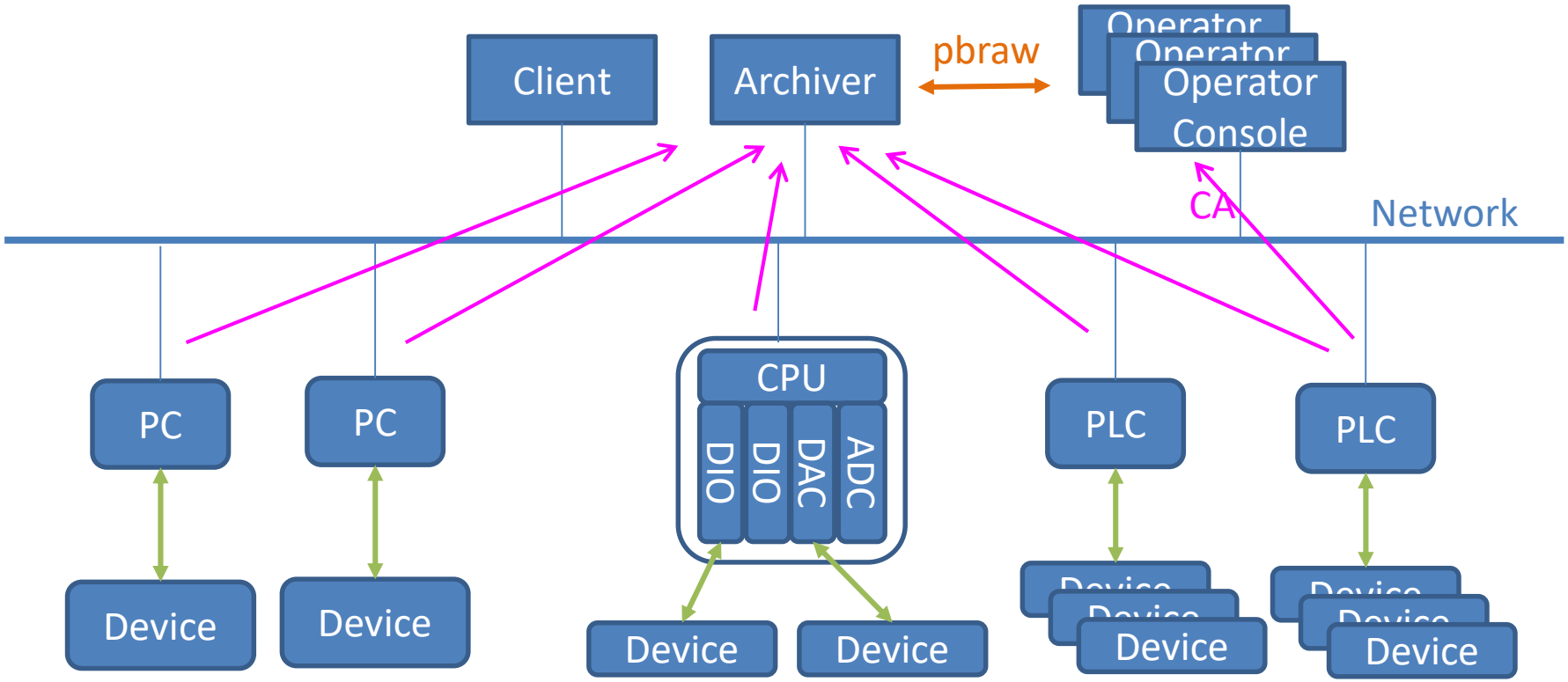


データ保存について

Archiverを設置することで、多くのIOCのデータをまとめて保存できる

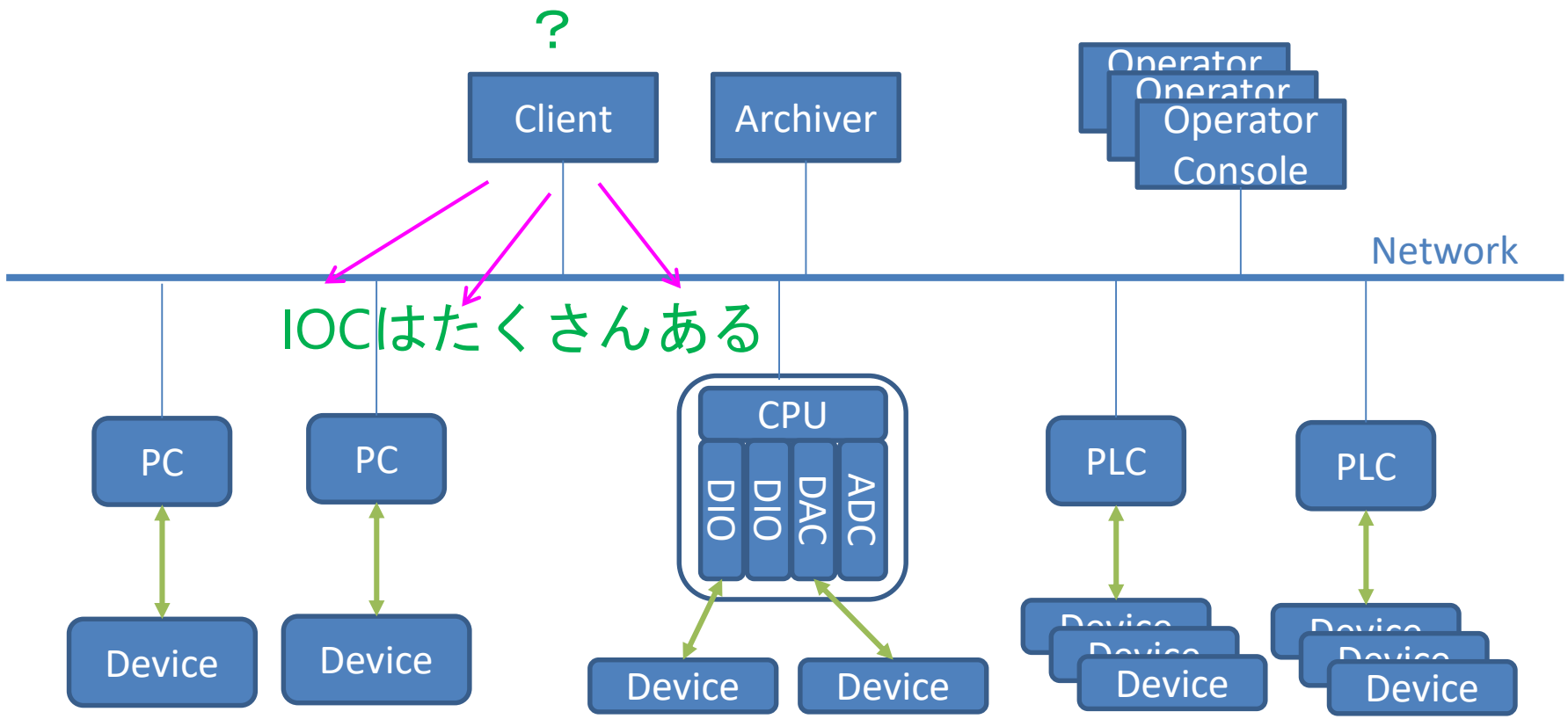
何種類かの実装が存在している。いわゆるデータベース(MySQL, PostgreSQL等)を使う場合や、時系列データに特化した高速・大容量保存形式など。現在はArchiver Applianceが主流 (ORNL開発)

CSS OPIでは現在の値とつなげて表示もできる。



今回は説明スキップ（時間があれば簡単に解説）

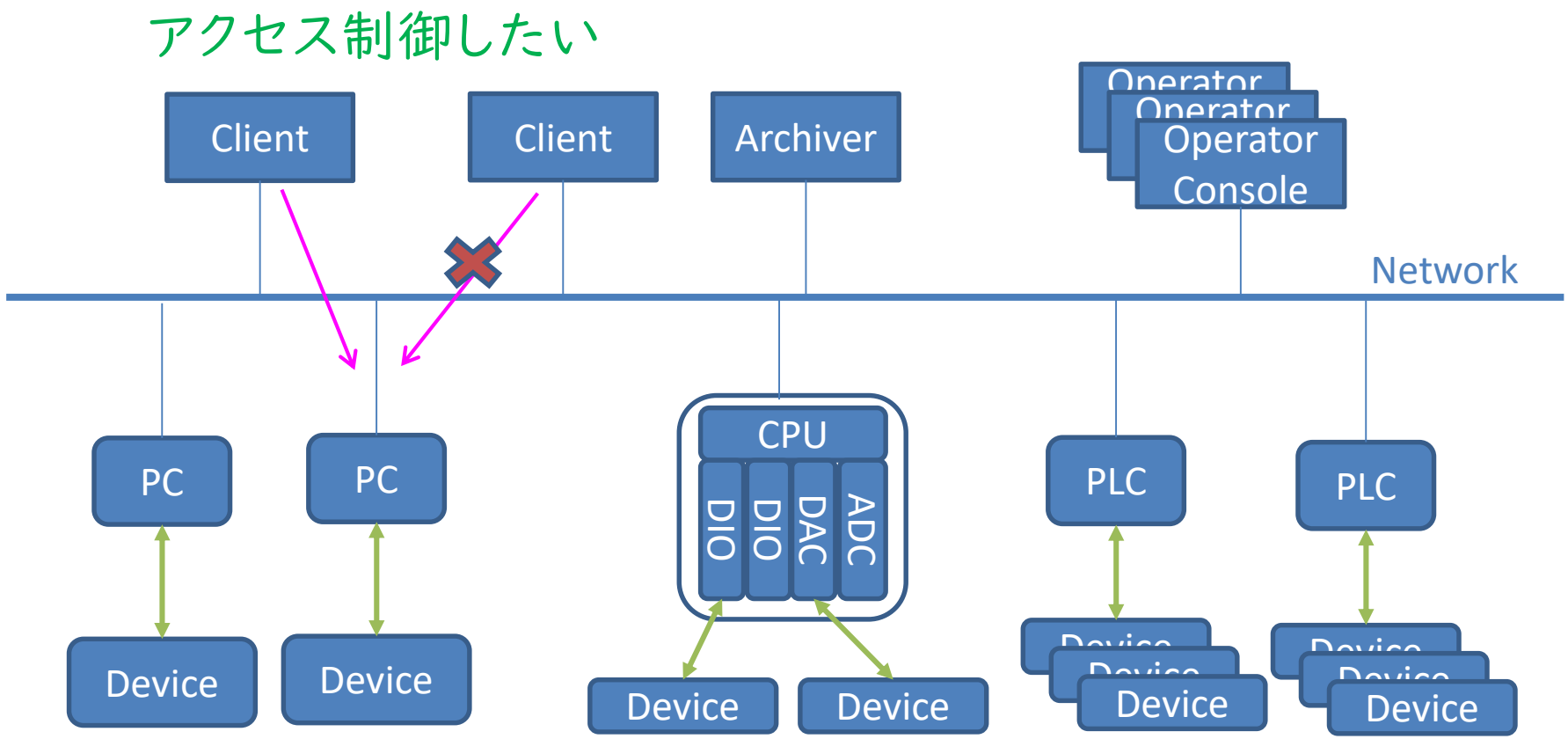
EPICSでは、どのようにして「レコード名だけ」で対象となるIOCを見つけてデータをとってくるのか？ネットワーク内にIOCはたくさん存在しており、どれが何のレコードを持っているかは分からない。



今回は説明スキップ

セキュリティはどうなる？

勝手に制御されては困る場合もある… EPICSでは Access Control の仕組みが用意されています



ほかにも色々な疑問があるかもしれません

- LEDを個別にon/offできることは分かったが、全部まとめてon/offしたい
  - いろいろな実装方法がありますが、EPICSではレコードをリンクして連続して動かすことができます
  - ソフトウェアを使ったsequencerを作ることもできます
  - 時間があれば実装してみましょう

- ネットワーク越しに制御するなら、単なる socket 通信で十分なのでは？
  - 接続管理をEPICS側でやってくれるのはとても楽

- もっと多くの装置をつなぎたい？
- KEKBとPF、Linacなど別の加速器が連携することはあるのか？どのようにつなぐのか？あるいは、ある程度の範囲で分けるべきなのか？
- どれくらいの時間精度で制御できるのか？

EPICSとは xxx ではない

EPICSは制御システムではない

- EPICSは制御システムを構築するためのフレームワーク（ツールキット）
- EPICSを導入しただけで、装置の制御ができるわけではない。制御対象に合わせた作業が必要。

EPICSは商用のソフトではない

- 自ら手を動かす必要がある
- サードパーティからサービスを購入することは不可能ではないが
- EPICS ReadyなHWを販売しているケースもある

EPICSは「DAQ」システムでは無い。

- 素粒子・原子核実験のようなタイプの実験データの取り扱いにBestとはいえない

by N. Yamamoto (What _is_epics.pdf)
EPICS Users JP 講習会資料よりダウンロード可能

EPICS：10のイカシタところ

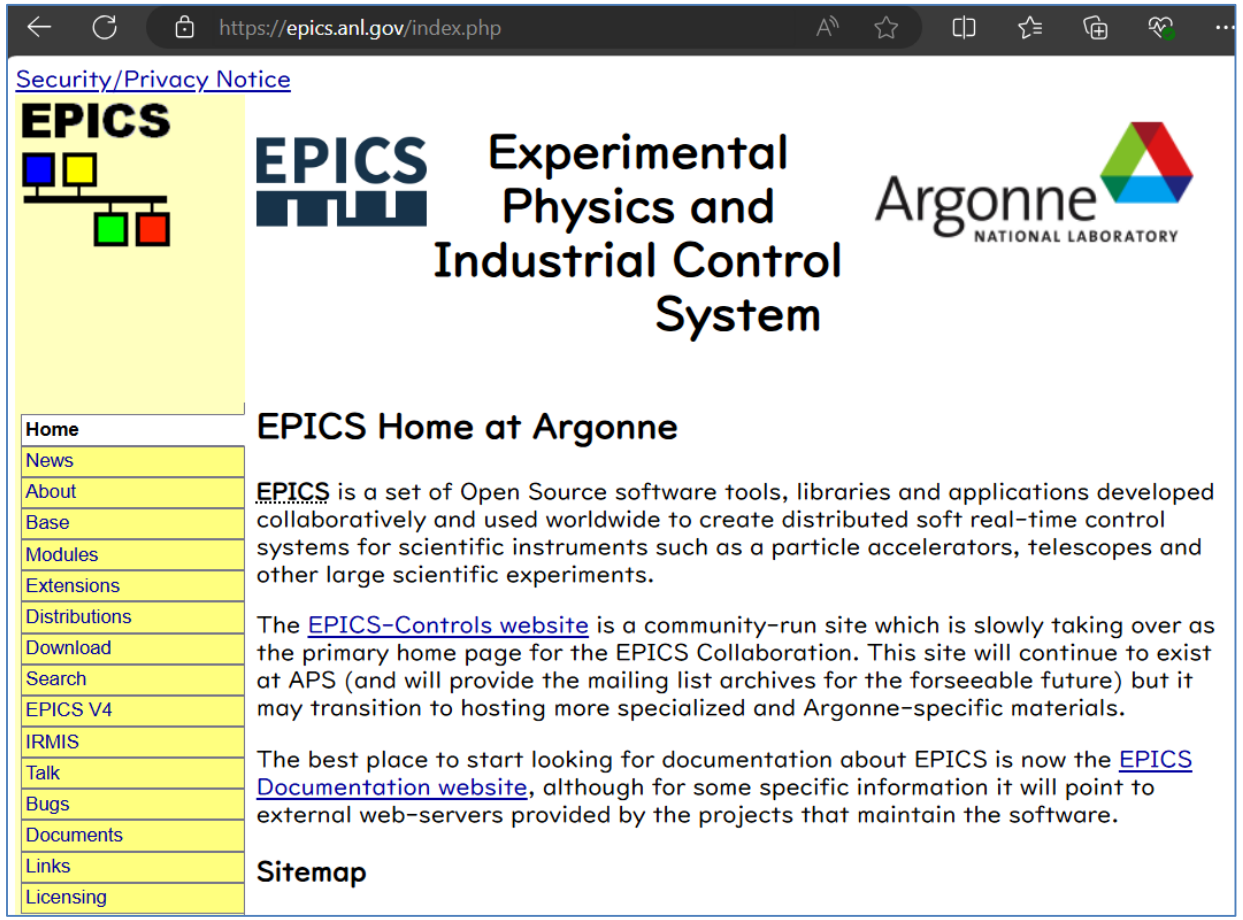
1. 無料である。将来のアップグレードについても課金は発生しない。
2. オープンソースのソフトウェアである。ソースコードはWebから(無料で)ダウンロードできる。改変、改良も自由(EPICS ライセンス)
3. たくさんのユーザがいる。:安定した運用の実績がある。
4. 制御ポイントにつけられた名前(CA name)を知っていれば良い。
5. たくさんのソフトウェアから好きなものを選べば良い….
6. … さもなければ、自分で作り上げることもできる。
7. 退屈な部分はすでに実装されている。
8. すぐ近くに多くの専門家がいる。: 色々相談もできる。
9. 良いコントリビューションは国際的に受け入れられる。:国際的な活躍の場がある。
10. 制御点数が10個でも、100万個でも対応できる。: Scalability

Ten Really Neat Things About EPICS
https://epics.anl.gov/neat.php

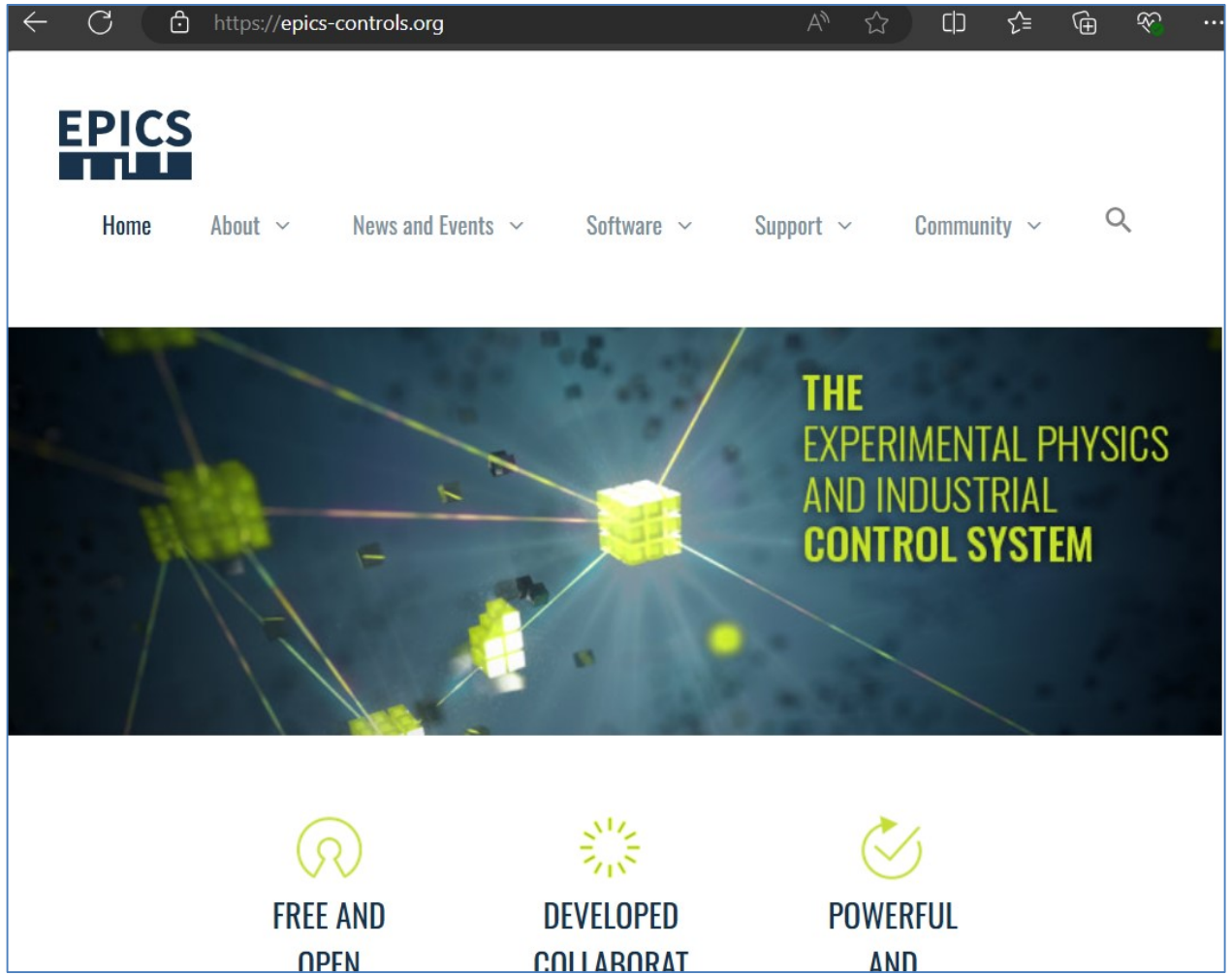
日本語訳 N. Yamamoto (What _is_epics.pdf)

EPICS Collaboration

EPICS Collaboration Meetings (anl.gov) 最新情報はこちらで



https://epics.anl.gov/



https://epics-controls.org/

backup slides

EPICSの基本コマンド

- EPICSの基本コマンド
  - caget
  - caput
  - camonitor

実習：EPICS Server側

- RasPiでIOC作成

IOCで使うOSの選定基準は？

リアルタイム性が必要かどうか、というところが最重要

EPICSで良く使われているのは

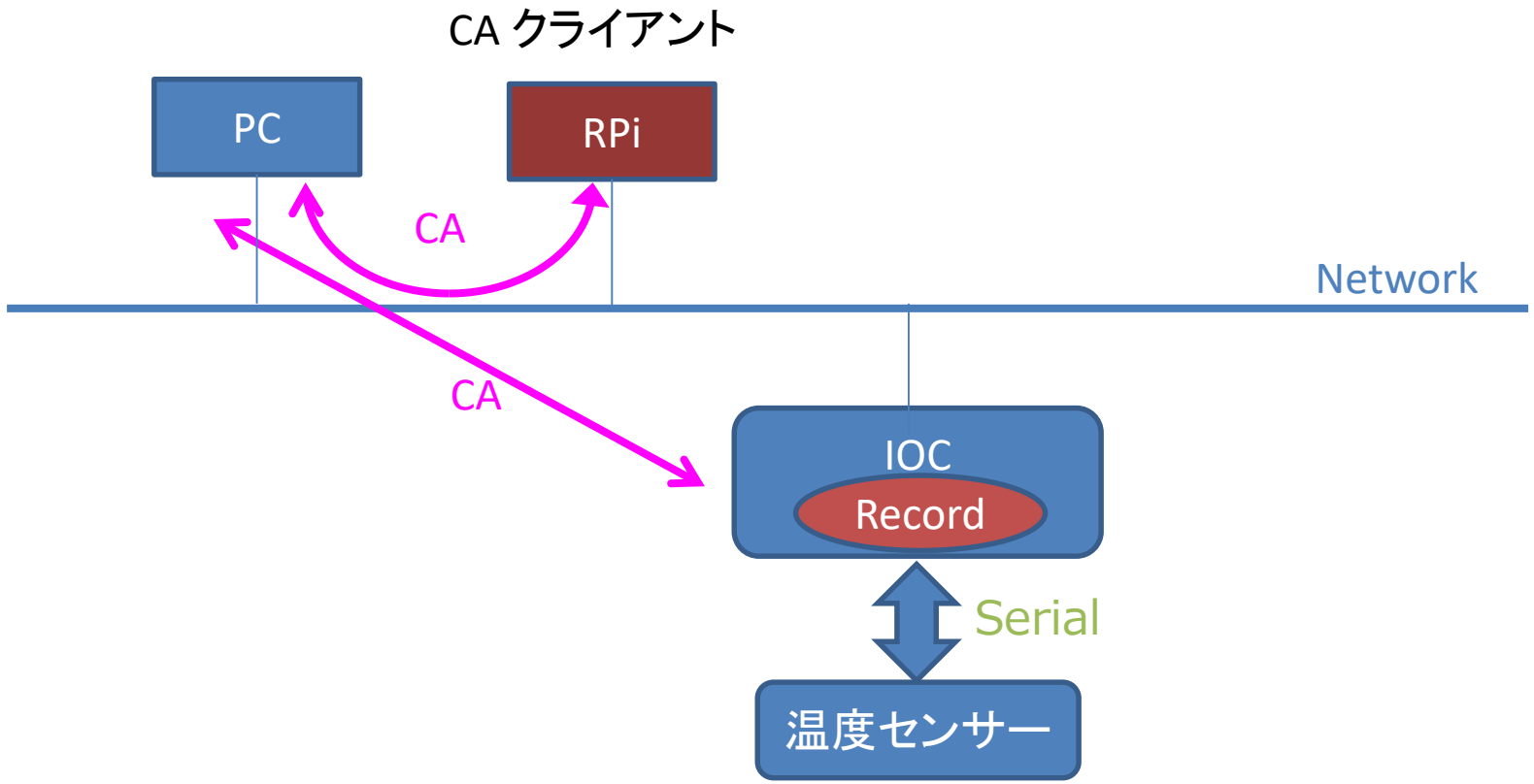
- Linux (linux-x86, x86_64, arm, etc)
- VxWorks
- RTEMS
- Windows (オシロスコープ内蔵タイプなど)

その他、多くのOSでサポートされている

※ IOCを作れば、上位部分はOSに依存しない
※ IOCのソフトウェアもOS非依存の部分が多くある

次のセッションでやること

自分のPCから直接 Channel Access でデータを取ってくる



そもそも、なぜ LabView を使わないの？

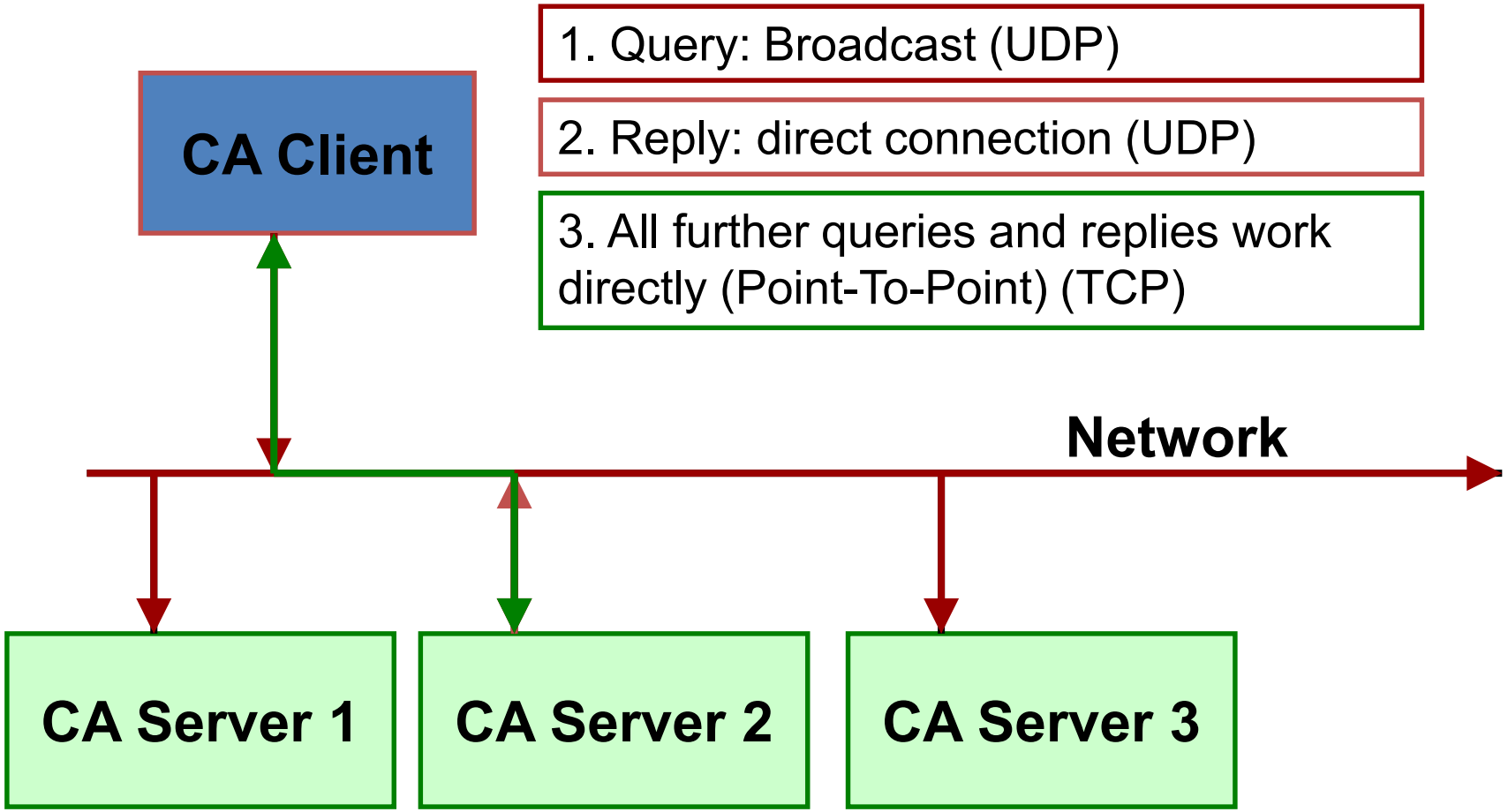
実験室レベルの、Localな測定にはLabViewは便利です。しかし…

- １人以上の関係者
- アーカイブ
- GUI
- 柔軟なクライアント構成：上位の物理プログラムとの連携

LabView以外にも、様々なソフトウェアは実在します。
〇〇を動かしてみた系のおはなし
ちょっと動かして、WebやらiPhoneやらで制御とか….ありますよね

あと、自前のソフトウェアでやっているところも多くあります。
EPICSの場合、最初の少し手間はかかりますが先人の知恵によって様々なトラブルに対しての対処が考えられています

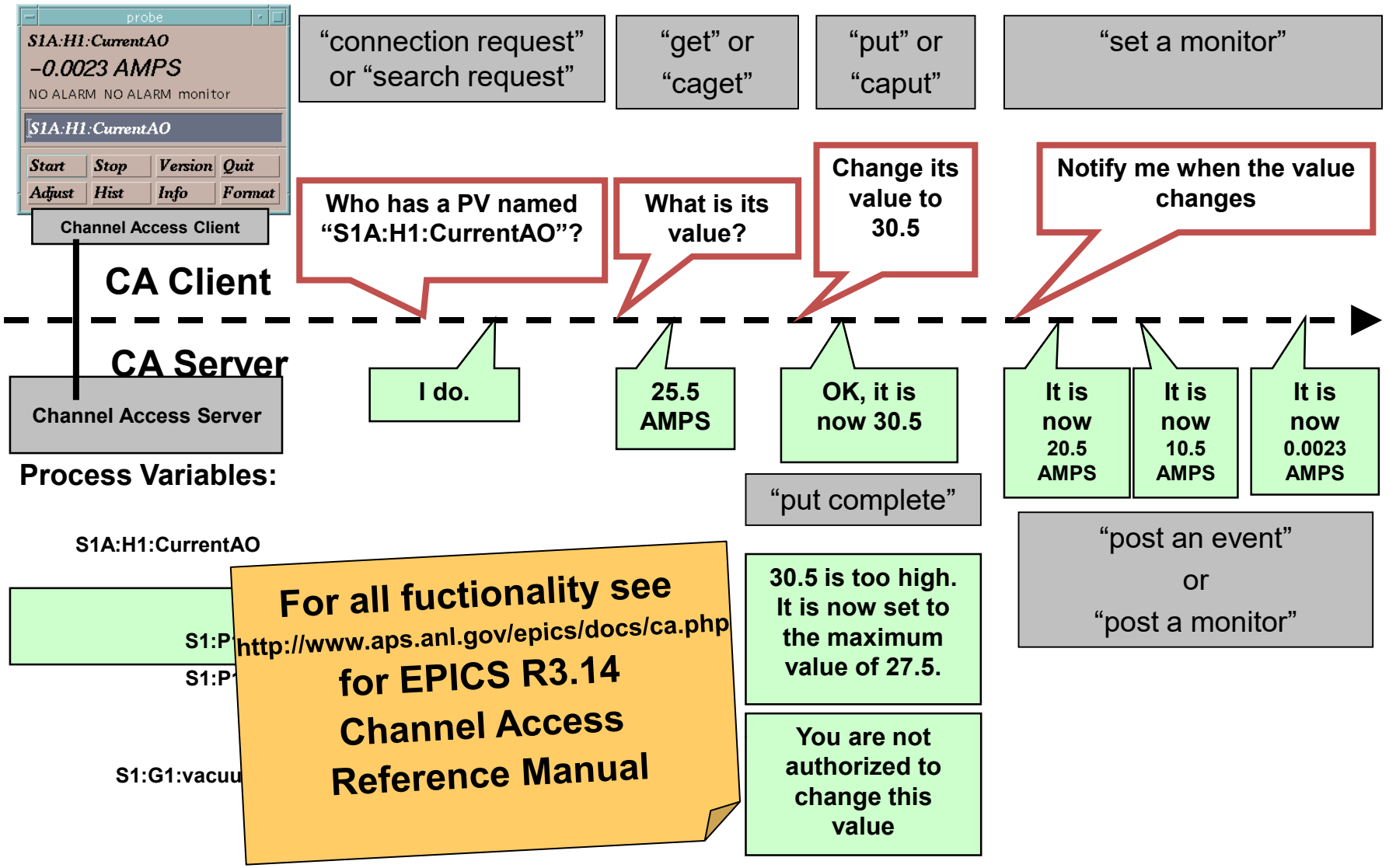
Some Details of Channel Access



Default UDP ports: 5064 and 5065
Default TCP ports: 5064 and 5065

Defined in environment variables
Need to be free in firewall!

Channel Access Commands



Are There Alternatives to EPICS?

PAUL SCHERRER INSTITUT PSI

System Name:

- **EPICS**
- **TANGO**

Collaborations:

Used at more than one Lab

Pro:
Bugs are already found

Contra:
Complicated to adapt to your problems

- **DOOCS**
- **Tine**
- ACS

Single Site Systems:

Developed and used in one Lab

Pro:
Your problems solved perfectly

Contra:
You are on your own (no one can help)

- **PVSS (Cern)**

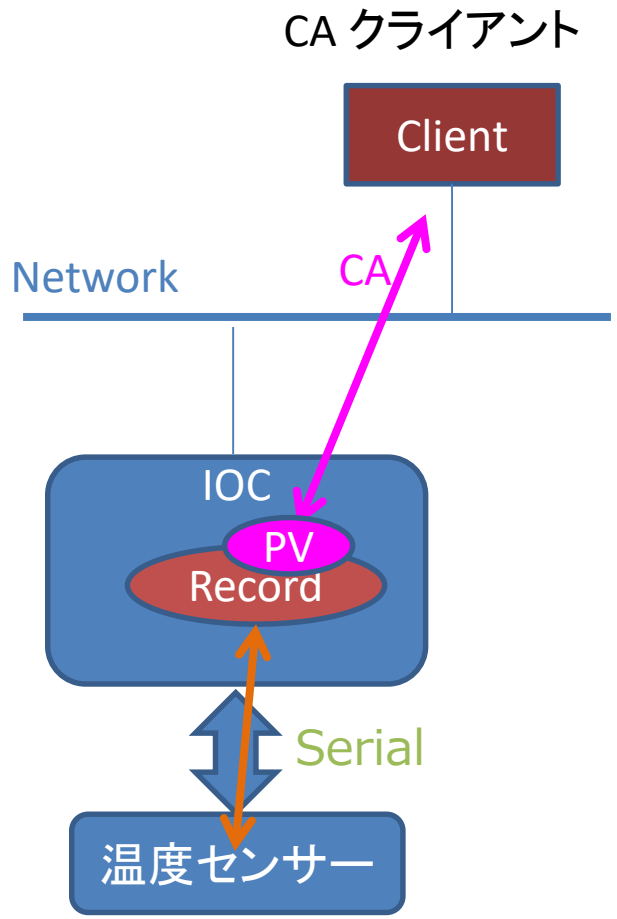
Commercial System

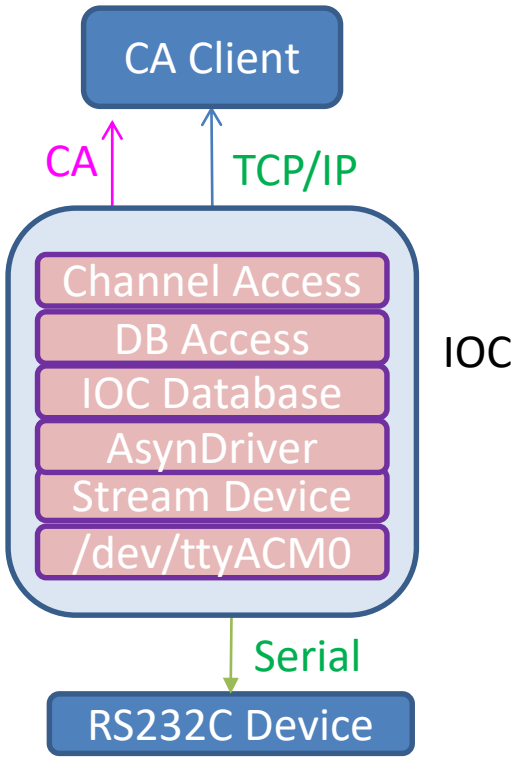
Pro:
Outsource your problems

Contra:
Expensive

Record, PV, Channel

クライアントのプログラムから見ると、CA という仮想回路を張って、その先にいるのがPV。PVにはフィールド（or Attribute）があって、それぞれがレコードのフィールドに対応している。あとはレコードが動いたりする。





ここまできて、いかがでしょうか？

どのような感想をお持ちになったでしょうか

- 難しい？
- あるいは簡単すぎ？

詳細はこの2日間で色々と説明していきますので、現段階ではおおまかな雰囲気を理解して頂ければ幸いです。IOC, Channel Access (CA), レコード（PV）、などのイメージがつかめれば十分です。

EPICSの Learning Curve は最初が急峻すぎる、との話もあります。
最初はとっつきにくくても、すこし経験すれば簡単に感じるでしょう。

はじめに

- わからないことがあれば、その都度質問してください
- いろいろな用語が出てきます。可能な限り英語の語源を考える（調べる）ように心がけましょう
- 今日は正確さよりも「ざっくりとした」イメージをつかんでもらうことを目指します

クライアント・サーバーシステム

- たとえばWikipediaによると: クライアントサーバモデル（英: client-server model）は、機能や情報（サービス）を提供するサーバと、それを利用するクライアントとを分離し、ネットワーク通信によって接続する、コンピュータネットワークのソフトウェアモデル（ソフトウェアアーキテクチャ）
- 例：PCで使っているChromeやEdge, Safari等のソフトは、Webサーバにアクセスして情報をとってくるクライアント。サーバは複数のクライアントからの要求を処理することが多い

- 語源
  - Server：サーブする人、サービスを提供する人
  - Client: 顧客、サービスを受ける人

- 似た用語に「マスター・スレーブ方式」がある
  - CPUをマスターとして、周辺機器がスレーブ
  - ご主人様と奴隷モデル
  - 用語が論争になることがあるので「ペアレント・ワーカー/ヘルパー」と表現することもある

フィールドバス

- そもそも「バス」形式とは？

- real-timeが必要なのかどうかが大きな分かれ目
- いわゆる普通の「イーサーネット」はリアルタイム性は（あまり）無い。

Fieldbus

ref: https://en.wikipedia.org/wiki/Fieldbus

From Wikipedia, the free encyclopedia

Fieldbus is the name of a family of industrial computer networks[1] used for real-time distributed control. Fieldbus profiles are standardized by the International Electrotechnical Commission (IEC) as IEC 61784/61158.

A complex automated industrial system is typically structured in hierarchical levels as a distributed control system (DCS). In this hierarchy the upper levels for production managements are linked to the direct control level of programmable logic controllers (PLC) via a non-time-critical communications system (e.g. Ethernet). The fieldbus[2] links the PLCs of the direct control level to the components in the plant of the field level such as sensors, actuators, electric motors, console lights, switches, valves and contactors and replaces the direct connections via current loops or digital I/O signals. The requirement for a fieldbus are therefore time-critical and cost sensitive. Since the new millennium a number of fieldbuses based on Real-time Ethernet have been established. These have the potential to replace traditional fieldbuses in the long term.

Contents [hide]