EPICS @ DESY/ XFEL
An Overview

EPICS Collaboration Meeting

San Francisco, October 5th, 2013

Matthias Clausen, DESY

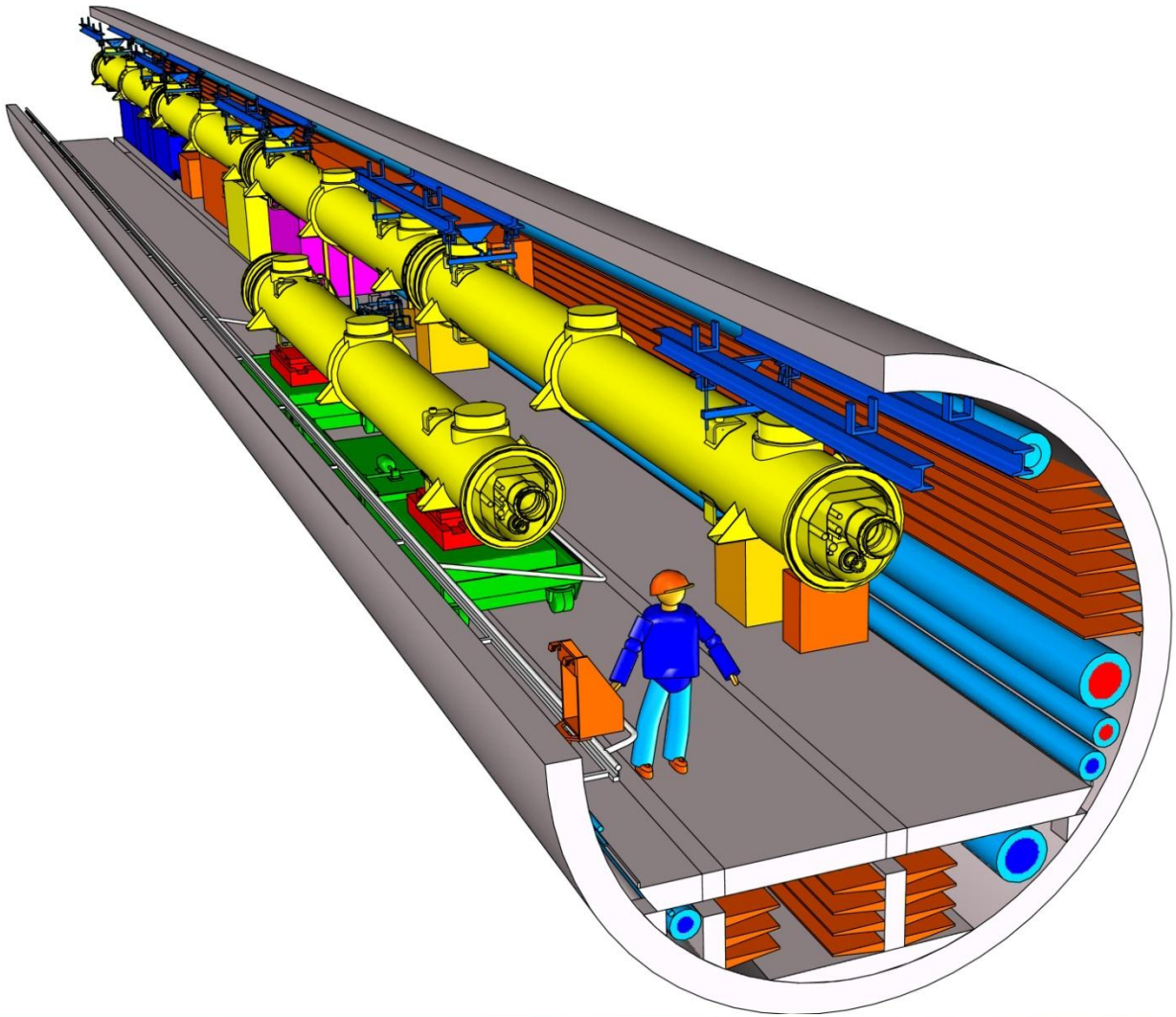
Cryogenic Control Group

Overview

- **Control Systems for the XFEL**
- **News from Cryogenic Controls**
- **CSS Developments**
  - Trent Plotter
  - Synoptic Display Studio (SDS)
  - CAJ – deadlocks
  - DAL -> DAL-II
- **Alarm System**
- **Archiver**
- **IOC Redundancy**

XFEL: One Machine/ Three control systems

The XFEL Tunnel



The real XFEL Tunnel

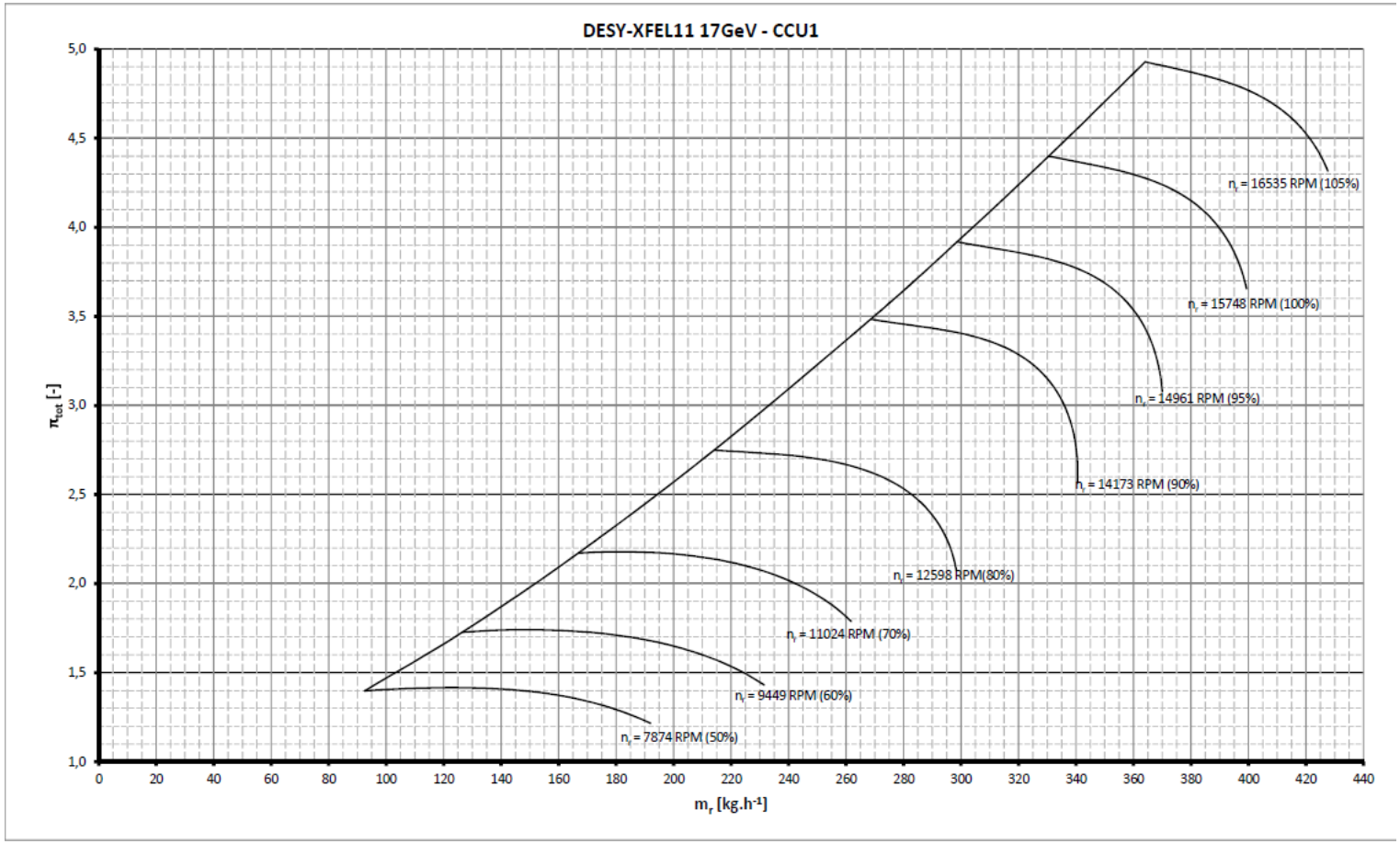
CSS Developments: Trend Plotter

Adding new features to Trend Plotter

- Displaying archive deadband for each channel (Monitor Deadband for online/ live channel)
- Interface to MySql Database

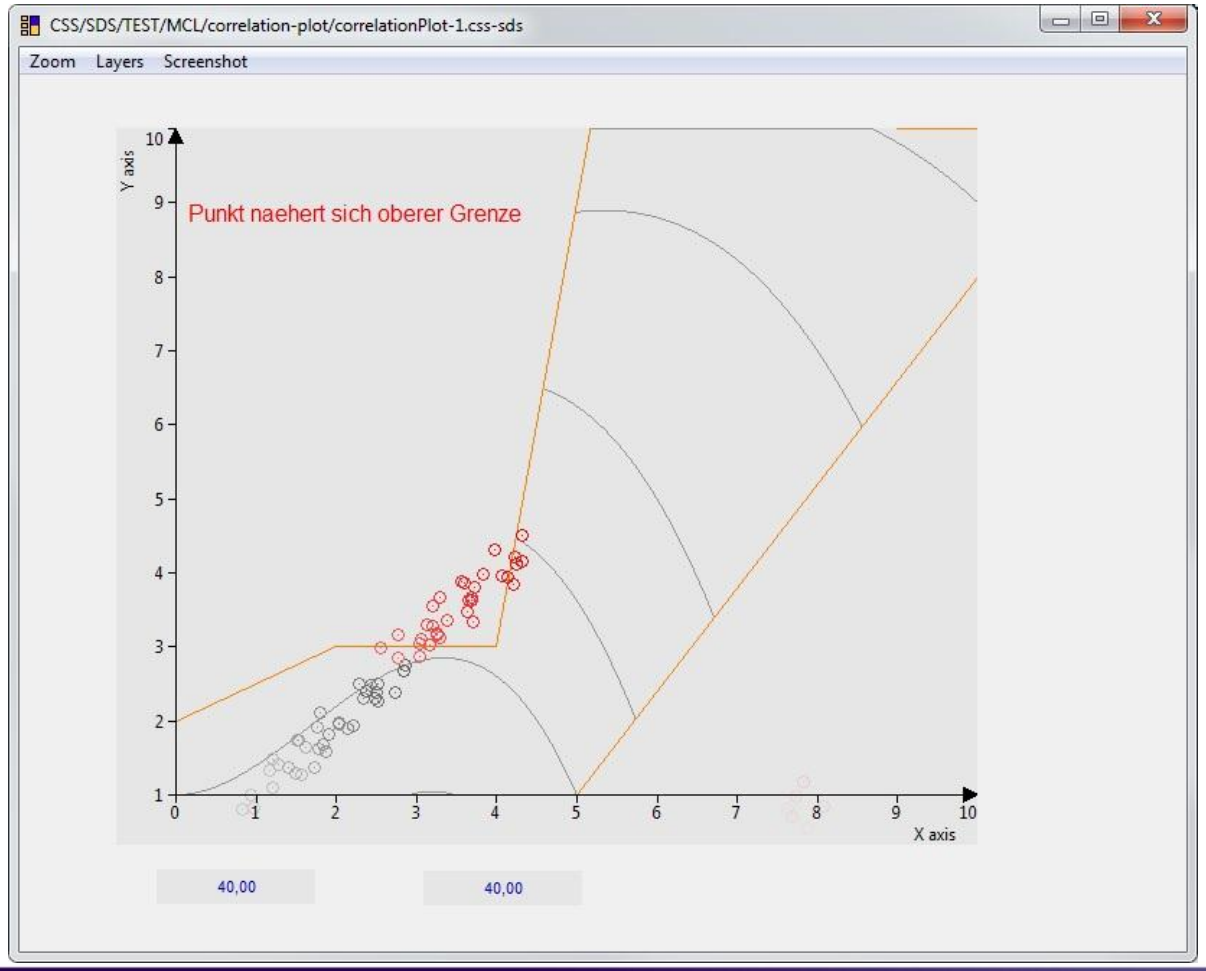
CSS Developments: New SDS Widget: X/ Y Plot

Exyample of Cold Compressor Characteristic Diagramm



CSS Developments: New SDS Widget: X/ Y Plot

Prototype of Cold Compressor Characteristic Diagramm



CSS Lessions Learned: CAJ - deadlocks

CSS is running in the control room as primary operator interface

Over the last two years CAJ deadlocks were causing a lot of trouble

Communication with Matej improved:

- Prepared fixes make it into the code
- New bug fixes are distributed faster

CSS Lessions Learned: DAL -> DAL-II

Lessions learned from DAL:

- **DAL is necessary to interface to several control system protocols**
- **Support for metadata/ characteristics for several protocols requires a common interface**
- **DAL-I was overloaded with too much functionality**
- **Redefining requirements to the basic needs**

A complete rewrite of the DAL layer is on the way

Alarm System

Everything is a message

(alarm-; systemLog-; putLog-;snlLog;log4JLog…)

- **Messages are all transported as JMS messages**
- **Messages can be displayed in tables (and trees)**
- **Messages get written to Oracle (jms2Ora)**

Messages become Alarms when processed in the Alarm Management System (AMS)

Alarm System

Developments:

- **DAL2JMS – (now using DAL-II)**
  - Channel-Lists get monitored and written to JMS (as Message)
  - Configuration similar to alh
- **Synchronized Alarm-Table/ Alarm-Tree**
  - Alarm Tree similar to alh Alarm Tree
- **AMS – Continuous improvement driven by control room**
  - Time based message filterns
    Generate alarm when state is active for more then ?seconds
  - Attach alarm details to message and forward on different topic

Archiver

Running in full production mode since March 2013

Basic requirements

1. **Data retrieval for any interval takes no more than 3 sec**
2. **Display archive deadband for every channel**

Implementation:

1. **MySql Database on Sun Cluster**
2. **In addition to the ‚raw' data, we store mean values on a hourly and minute basis**
3. **Define ‚related' channels which archive the deadband of the main channel**
4. **Replace pvManager by DAL-II (1Q 2014)**

Redundancy

First mentioned during the EPICS Meeting at Argonne 2006:

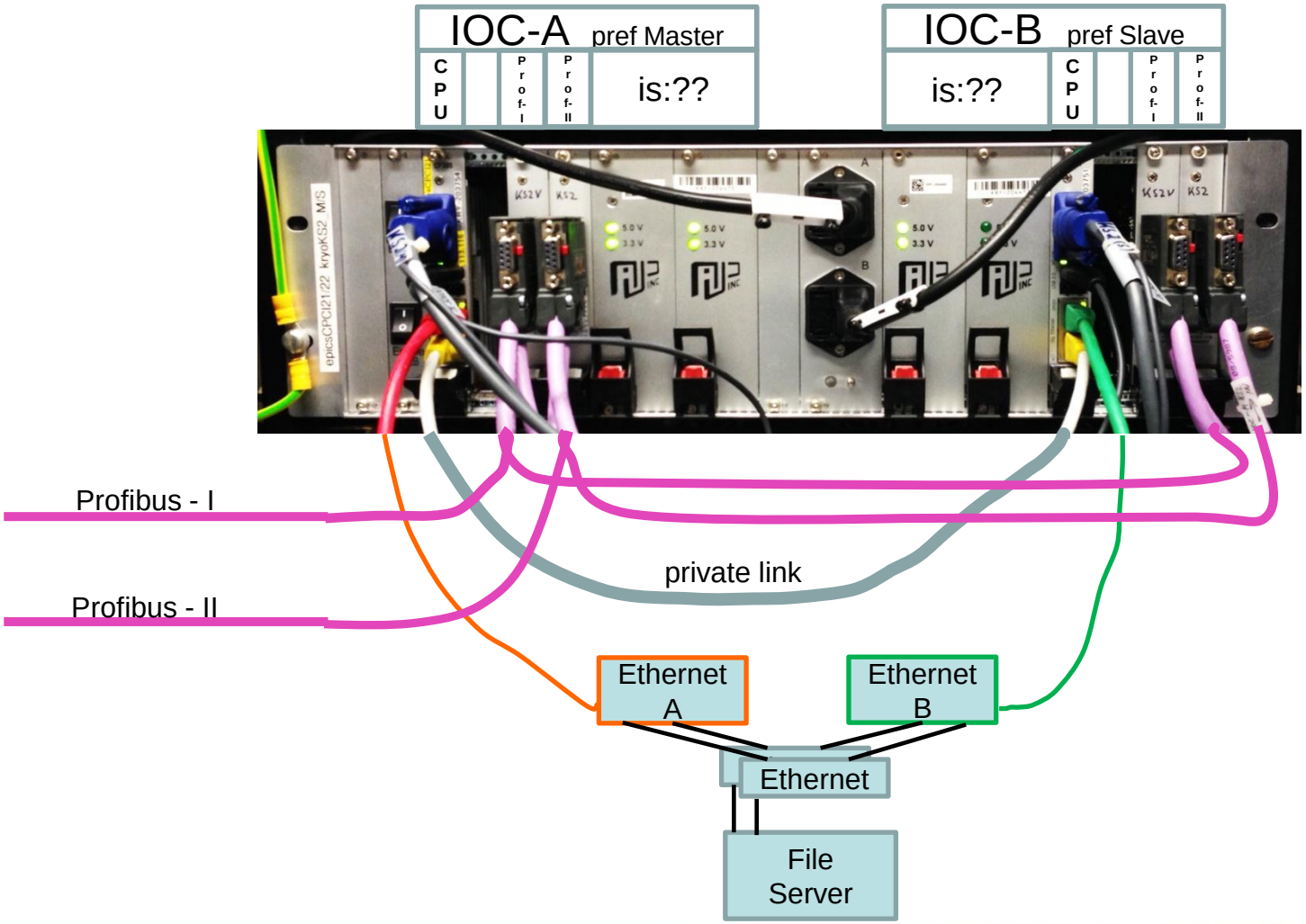
2006 – Presentation:

From the preamble of the design specification:

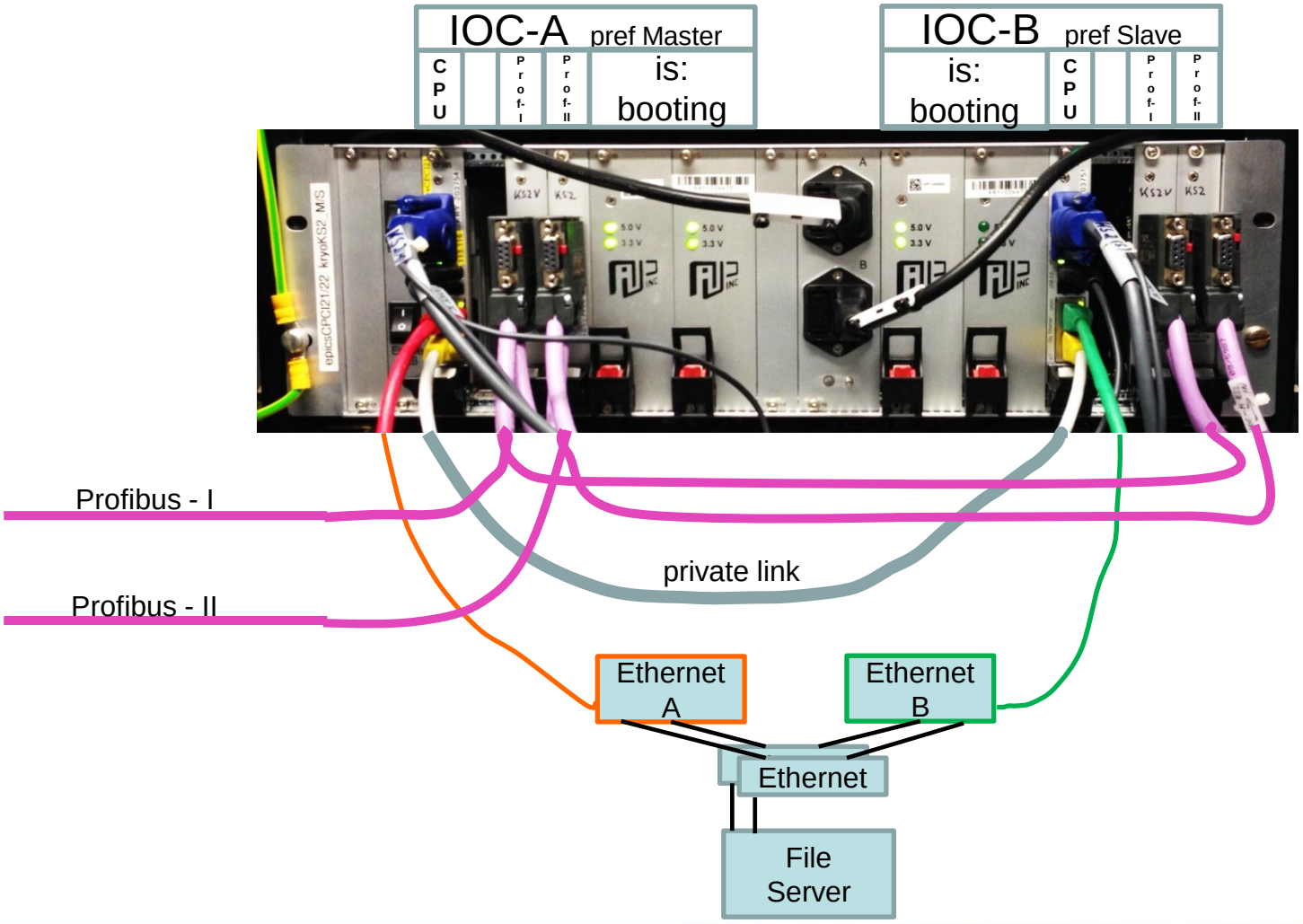
… last and most importantly one major design goal must be matched:

Any redundant implementation must make the system more reliable than the non redundant one. Precaution must be taken especially for the detection of errors which shall initiate the failover. This operation should only be activated if there is no doubt that keeping the actual mastership will definitely cause more damage to the controlled system than an automatic failover.

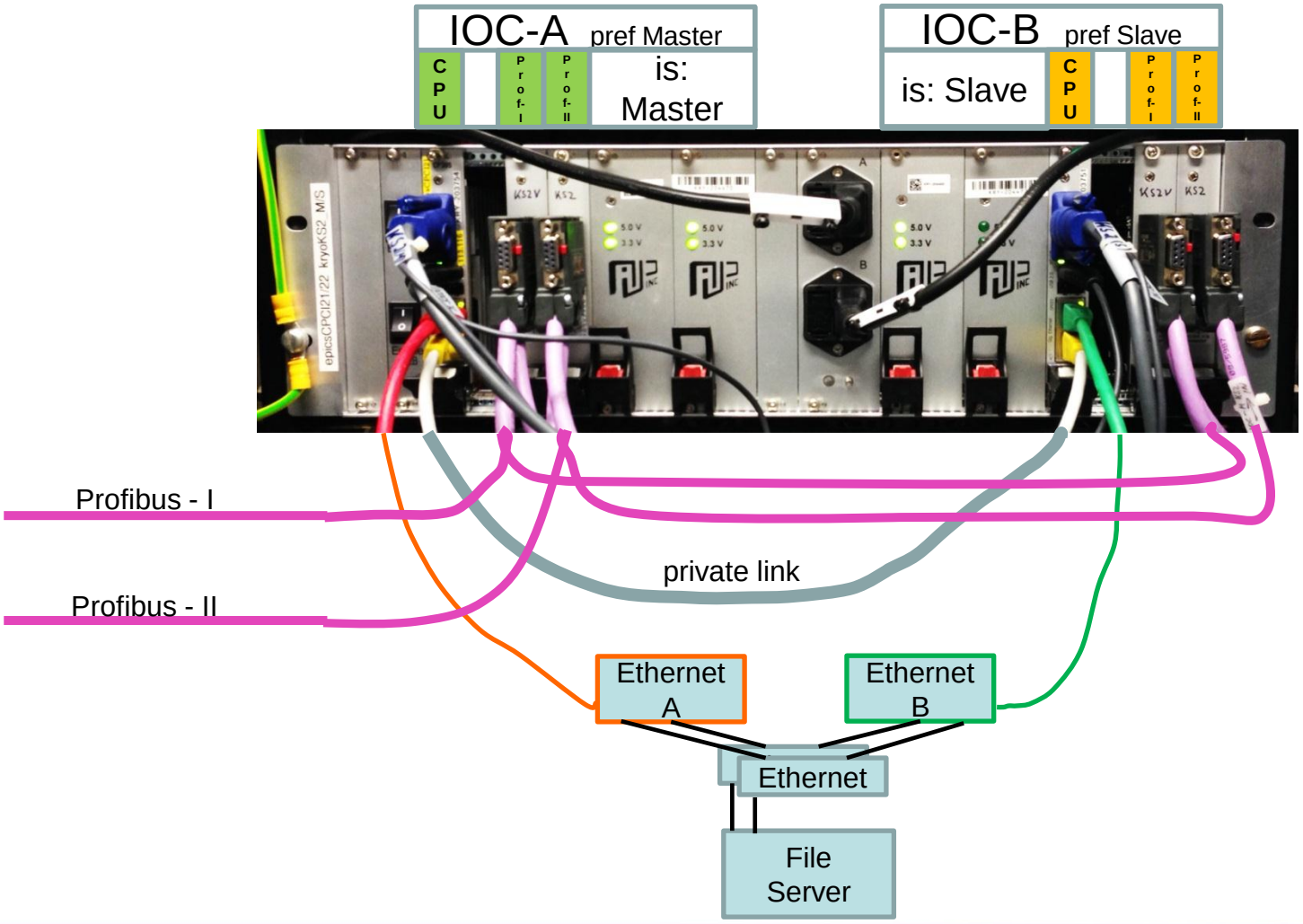
Redundancy: initial states



Redundancy: Startup

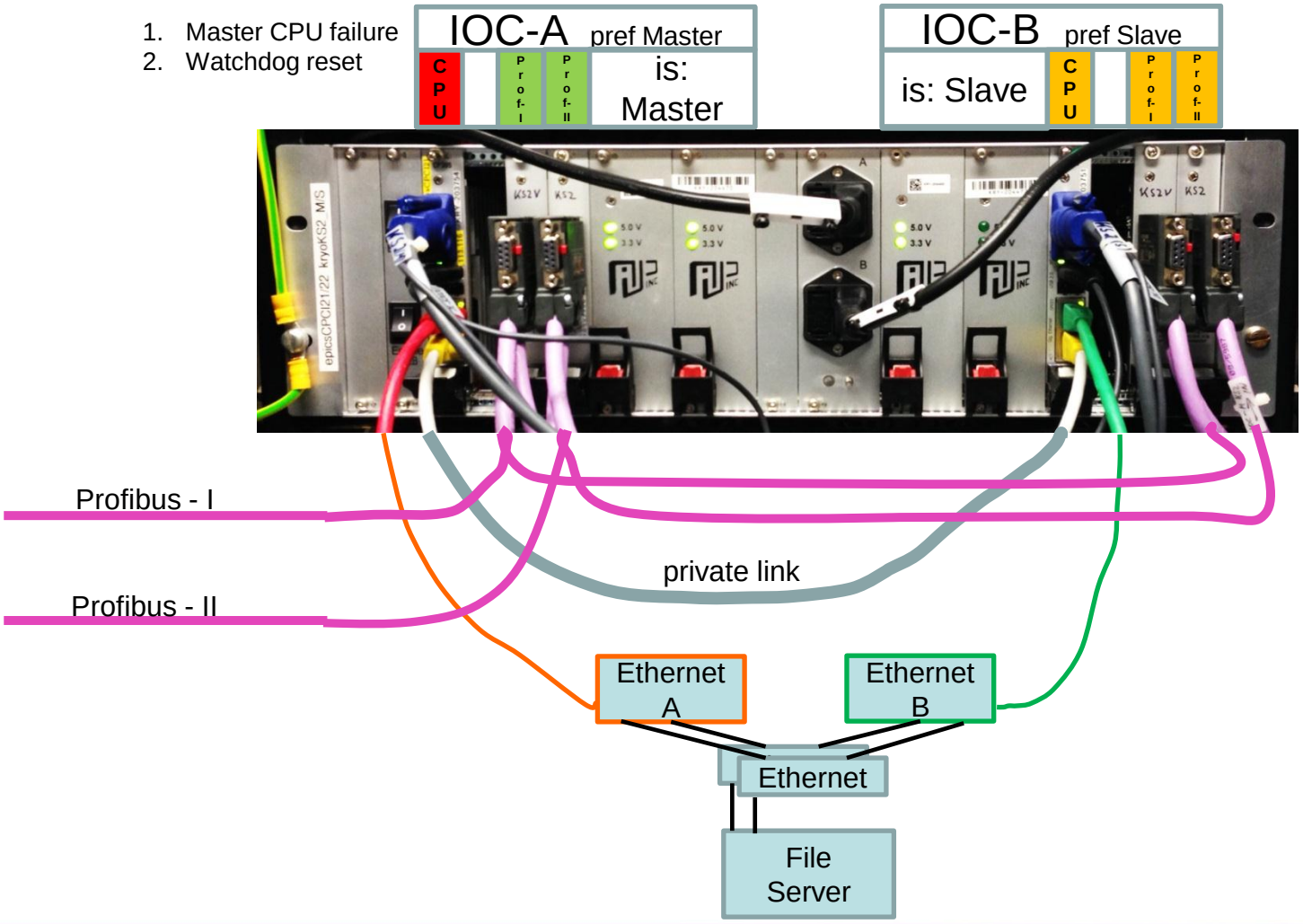


Redundancy: Running



Redundancy: Master CPU Failure

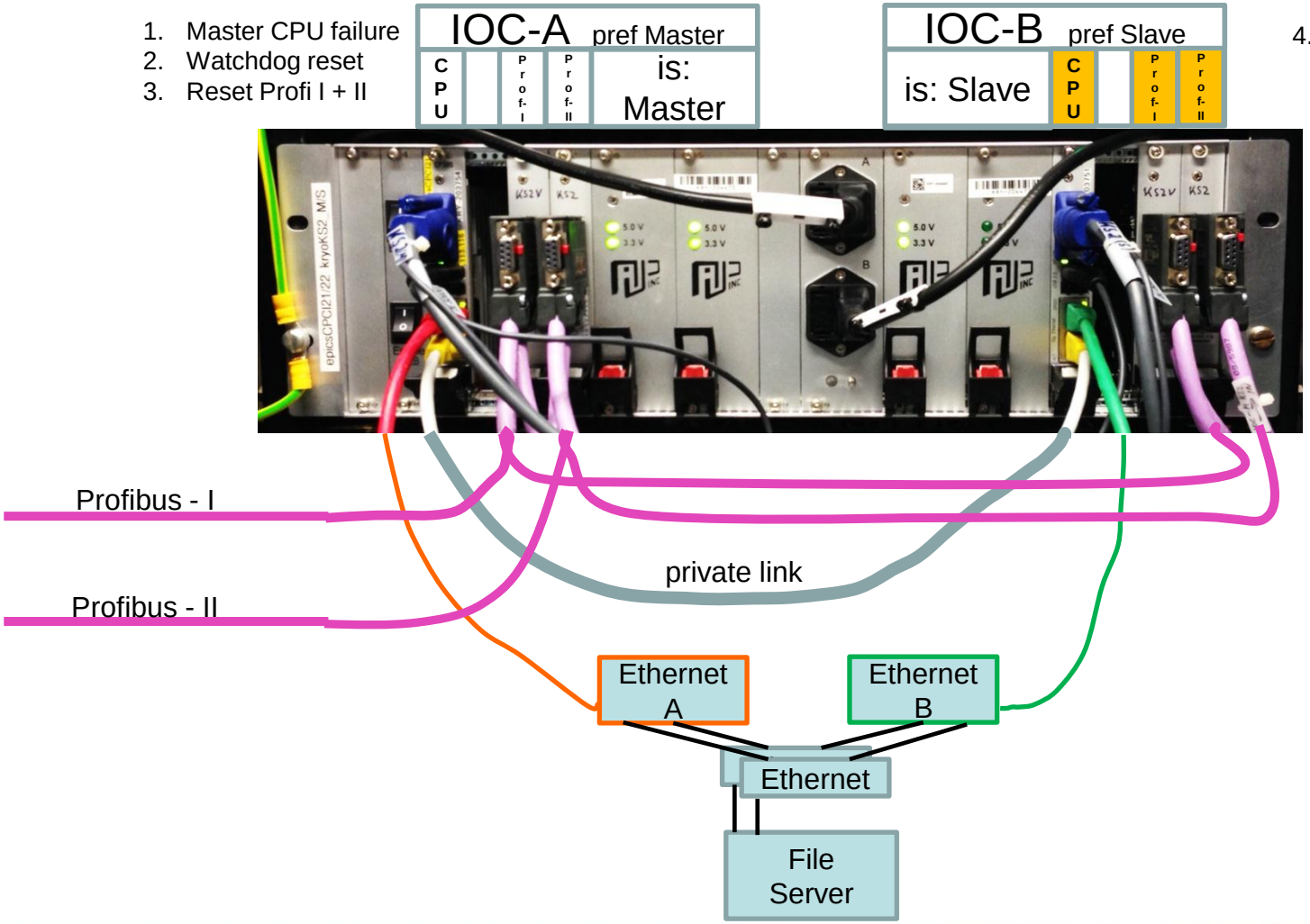
1. Master CPU failure
2. Watchdog reset



Redundancy: Master CPU Failure – Slave detects failure

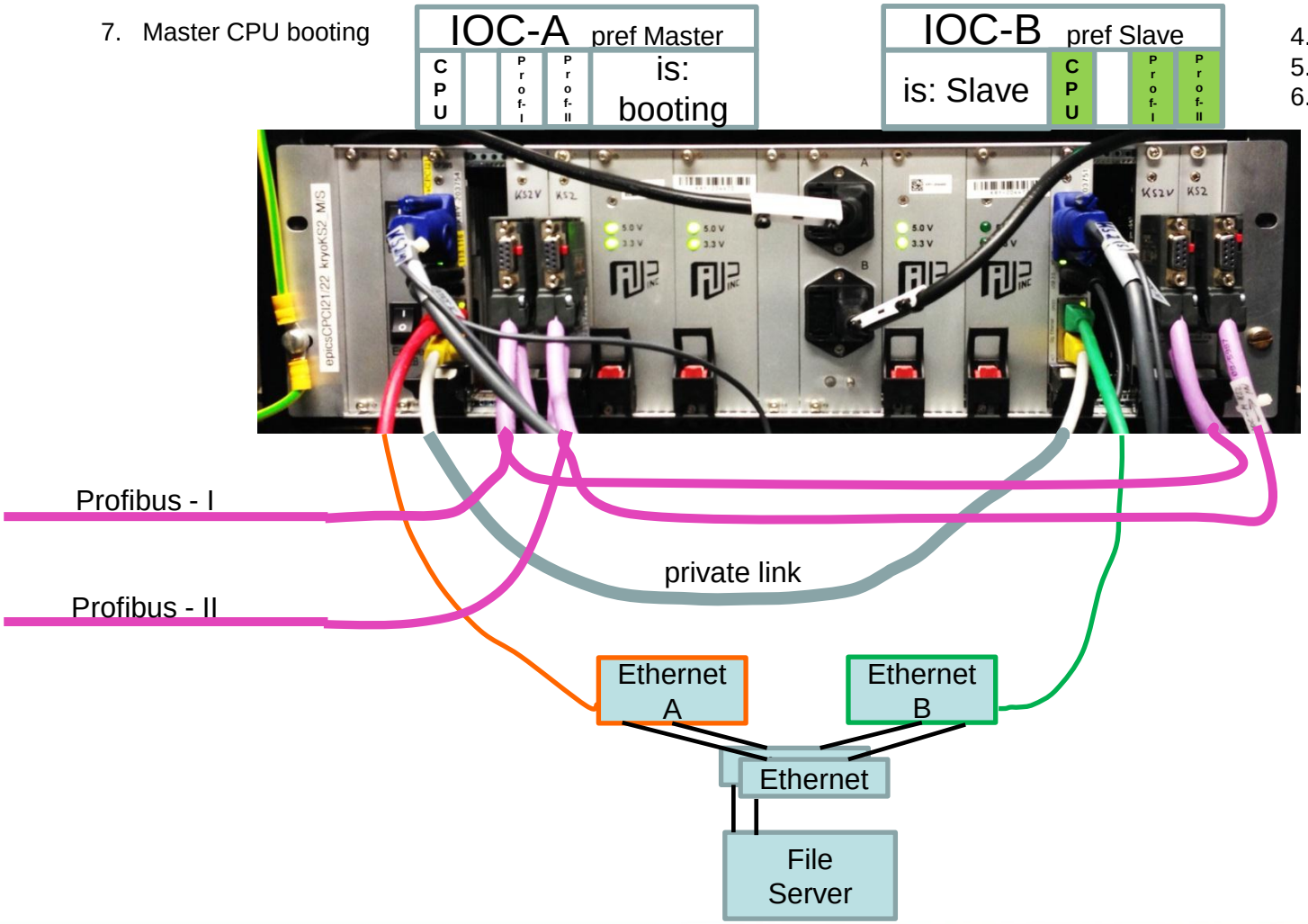
1. Master CPU failure
2. Watchdog reset
3. Reset Profi I + II

4. Slave detect master failure



Redundancy: Master CPU Failure – Slave taking mastership

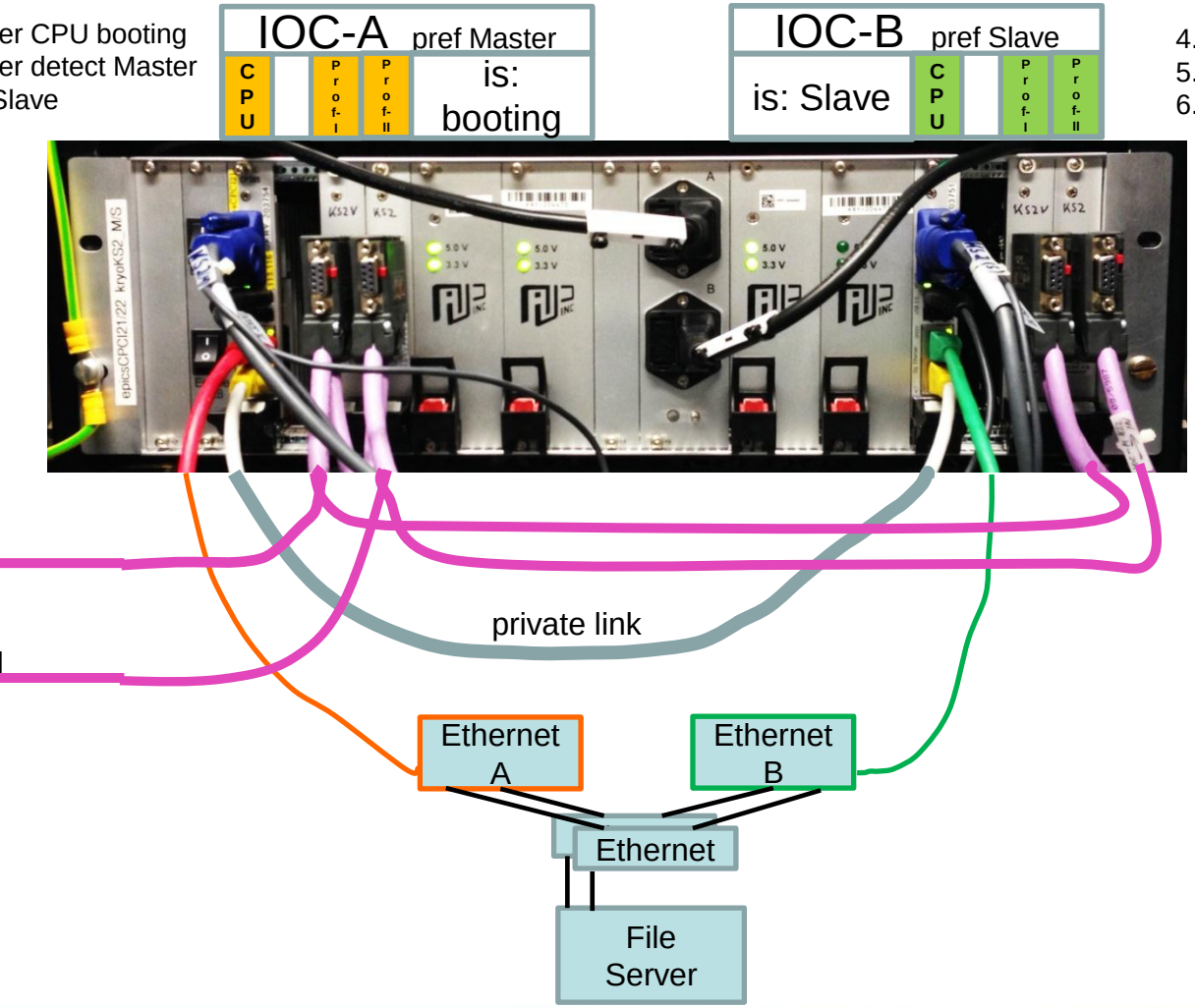
7. Master CPU booting



4. Slave detect master failure
5. Start Master
6. Start Profibus I+II

Redundancy: Master CPU Failure – Slave active/ pref master passive

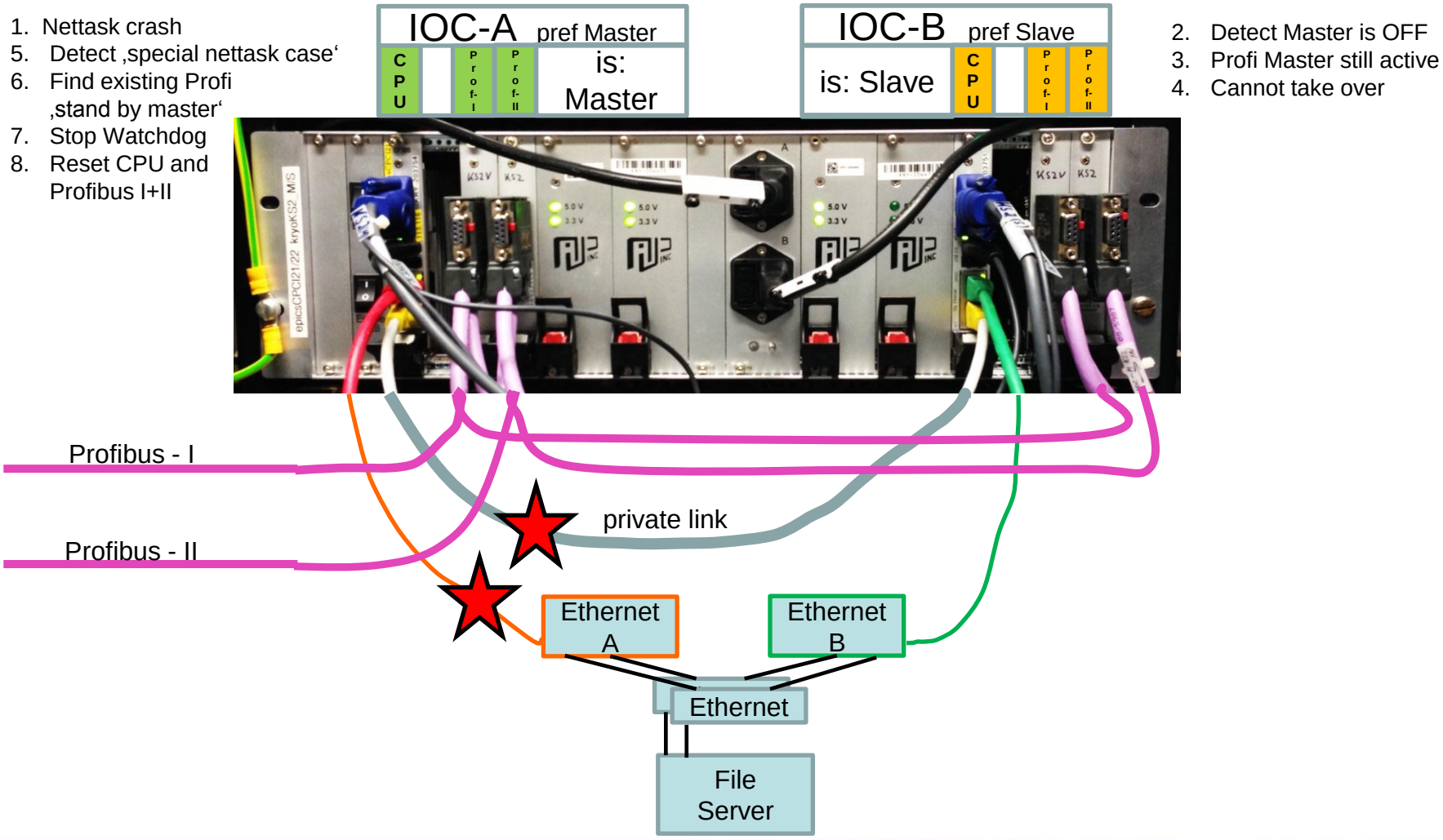
7. Pref Master CPU booting
8. Pref Master detect Master
9. Become Slave



4. Slave detect master failure
5. Start Master
6. Start Profibus I+II

Redundancy: Master CPU Nettask Crash

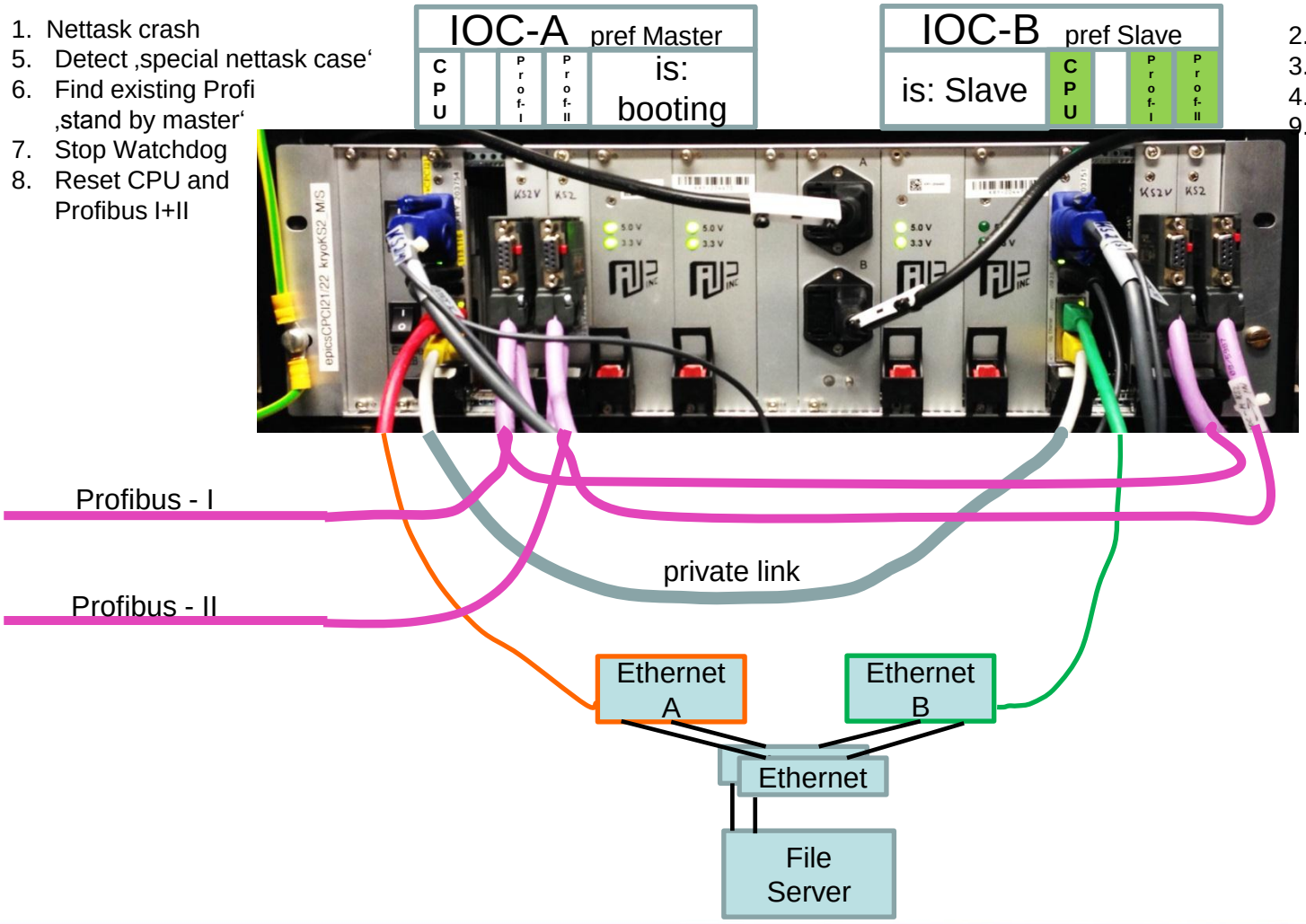
1. Nettask crash
5. Detect ‚special nettask case‘
6. Find existing Profi ‚stand by master‘
7. Stop Watchdog
8. Reset CPU and Profibus I+II



2. Detect Master is OFF
3. Profi Master still active
4. Cannot take over

Redundancy: Master CPU Nettask Crash

1. Nettask crash
5. Detect ‚special nettask case‘
6. Find existing Profi ‚stand by master‘
7. Stop Watchdog
8. Reset CPU and Profibus I+II



2. Detect Master is OFF
3. Profi Master still active
4. Cannot take over
9. Take over

Redundancy

The current implementation is running since three years in three IOCs in production.

Production:

- **6kW@4K Helium plant**
- **3MW primary compressor power**
- **All (PID) control loops running in the IOC**
- **SNL programs controlling sequences and supervisory control loops**
- **PLCs only used for hardware interlock**

➔helped us to survive 6 nettask crashes