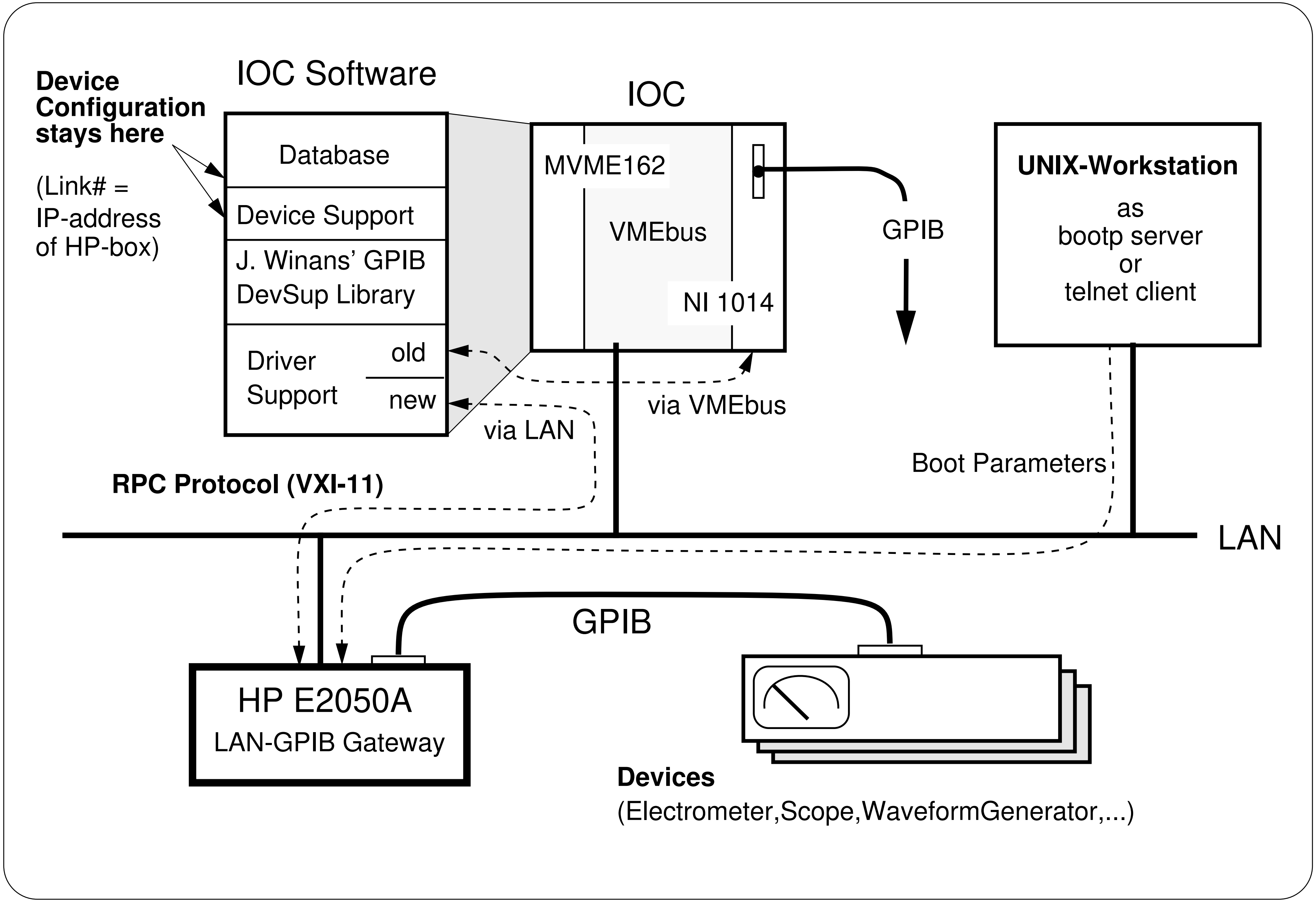
Overview

- interface device: HP E2050A LAN-GPIB gateway
- access is through an RPC protocol (standardized as VXI-11)
- EPICS driver support based on / compatible with GPIB support in base
  => old GPIB device supports can be reused without changes
- advantages over the classical solutions:
  - lack of “lovely features” known from other interface devices (NI1014...)
  - devices and IOC locally independent
  - one IOC can control arbitrarily many GPIB segments

Experience

- used at BESSY II site for master clock, scope readings, keithley electrometers, motor controls (PTB)
- also used by other sites: LANL, Saclay, SLS, LSB, LANL

LAN-GPIB Support - Standard Setup Compared to VME Card Solution



LAN-GPIB Support - New Developments

Driver

- dynamic initialization: new devices and even new gateways can be plugged in at run-time (developed by Till Straumann, PTB); call to initHooks now obsolete
- I/O Interrupt scanning by SRQs is now fully supported, mysterious SRQ problems seem to be solved
- detection of dead communication (gateway shut down or disconnected from LAN) => I/O task for this interface device will stop dispatching requests
- reinitialization of an interface is possible without reboot

Device Support Library

- complete revision greatly reduced the amount of code, interface remains compatible
- support for mbb[io]Direct records (long overdue)
- full SRQ support, generic SRQ handler

General

- lots of bug fixes in both areas
- current version is 2.1

The Big Problem

- five different types of GPIB interface devices: HP E2050A, Sony-Tektronix ET488, NI1014, a Bitbus module and IP carriers
- first one is handled by my driver, second by Noboru Yamamotu's version and last three by Noboru's and the by the old version in base
- no two of the existing drivers can be used on the same IOC

Proposed Solution (according to M. Kraimer)

- all GPIB support will be unbundled from base (& put on the APS server!!)
- the GPIB driver will be stripped of *all* device specific things (Generic Driver)
- gets called by device supports without knowledge of the interface device
- I/O for the specific interface devices is handled by seperate low level modules
- these are called by the generic GPIB support through the same interface
- link numbers are assigned to an interface device type by an init function of the device specific driver module that is called from the startup file
- this function also registers this link with the generic GPIB support so the latter knows which module to call