LHC Beam-Based Feedback Architecture

Ralph J. Steinhagen,
CERN, Beam Instrumentation Group

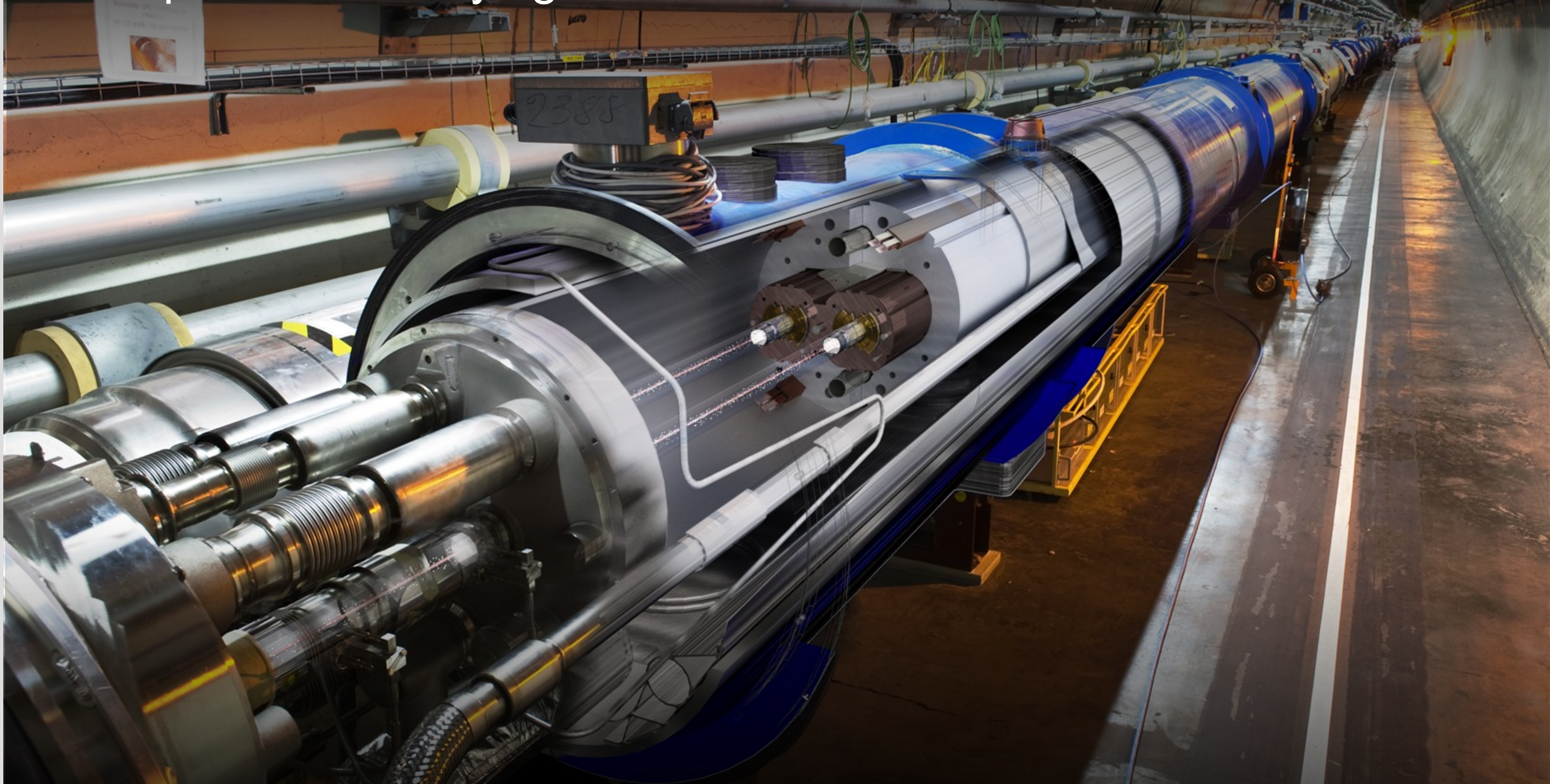
People involved, that contributed and shaped the LHC FBs

- J. Andersson (ex-CO), M. Anderson, A. Boccardi, T. Bohl, A. Butterworth, E. Calvo Giraldo, P. Cameron (BNL), R. Denz, K. Fuchsberger, M. Gasior, S. Jackson, L. Jensen, R. Jones, M. Jonker, J.M. Jouanigot, K. Kasinski (ex-BI), Q. King, K. Kostro, M. Lamont, T. LeFevre, S. Page, L. Ponce, V. Ranjbar (FNAL), G. Sivatskiy, R. Steinhagen, C.-Y. Tan (FNAL), E. Tedesco, J. Tückmantel, A. Verweij, J. Wenninger, W. Venturini, T. Wijnands, M. Zuin (ex-IT), and many more

- Special thanks to our colleagues in the synchrotron-light-source-world colleagues: M. Böge, G. Decker, G. Rehm, T. Schilcher et al.

Outline

- Requirements: 'What was specified' vs. 'What was/is needed' → impact on underlying feedback architecture



- Specific Feedback Design and Constraints → second talk
- Required improvements with respect to Initial Design → third talk

Context and Legacy of Earlier FB Reviews

- 2003: Initial Orbit-FB Prototype tests at SPS – main outcome:
  - Feasible for LHC established (tested up to f_s = 100 Hz) → to be deployed 2007
  - criticality of real-time latencies on the network and host operating system
  - Need for handling input & output errors (measurement data quality)
- 2003: Orbit Feedback Workshop → LTC: established architecture
- 2004: Stabilisation workshop in Grindlewald:
  LHC Orbit-FB more similar to those in SL-Sources
- 2005: Formalised Orbit-FB Specification (LHC OP Meeting #40)
- 2006: Chamonix XV (Spring): Architecture extended by Tune-FB
  & FBs on the roadmap for LHC commissioning
- 2006: LHC Commissioning WG: Review on FB Architecture
  - "[..] Biggest problem so far for LHC feedbacks: Human resources to implement the FB controller, service unit, GUIs, … [..]"
- 2006: Tune-FB Final Design Review (Autumn, CERN & US-LARP), OFSU
- 2007: LHC Commissioning WG: Status Update & Commissioning Plans
- 2007-10: LHC-CWG: Reviewed detection of LHC BPM errors and faults
- 2007-12: Ditanet WS on Q/Q' Diagnostics: … yet another review

Context and Legacy of Earlier FB Reviews – Cont.

- 2008-03: LTC Summary & Review: LHC Q/Q' Diagnostics & FBs
- 2008-09: AB Seminar on LHC Feedbacks
  - for those who never heard of FBs (repeated in 2009)
- 2009-10: BI-Technical Board on LHC Feedbacks
- 2010-10: LHC First Tune-FB Ramps results
- 2010-06: MPS Review: Impact of FBs on Machine Protection
  - Identified previously not-handled issues (timing/energy telegrams, rogue packets, measurement quality, QPS cross-talk → solve non-FB specific issues at source)
- 2011-12: Internal BI review on OFC/OFSU software architecture
- 2012-03: LMC: Update on Orbit- & Tune-FB modifications
- 2013: MP Review: Experiences with FBs and foreseen Improvements for LS1

Some references:

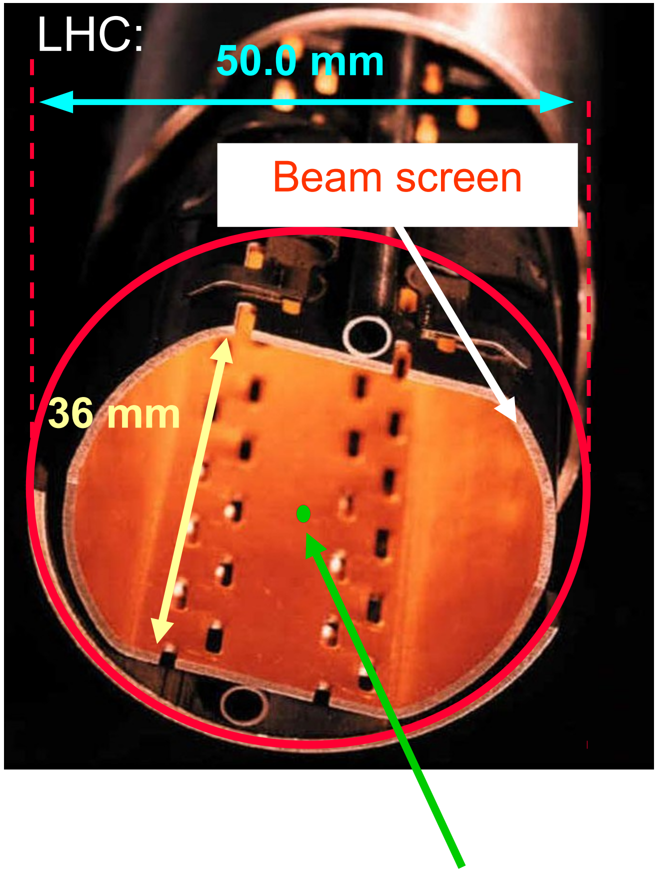
- http://cern.ch/AB-seminar/talks/AB.Seminar.rst.pdf (CERN-AB-2007-049)
- http://lhccwg.web.cern.ch/lhccwg/Meetings/2007/2007.10.23/2007-10-23_LHCCWG-FAULTY_BPM.pdf
- http://accelconf.web.cern.ch/AccelConf/PAC2011/talks/weobn2_talk.pdf &
- http://accelconf.web.cern.ch/AccelConf/PAC2011/papers/weobn2.pdf
- LHC-BPM-ES-0004 rev. 2.0, EDMS #327557, 2002,
- svn+ssh://svn.cern.ch/reps/acco-co/trunk/lhc/lhc-feedbacks

Beam Parameter Stability in Hadron Machines

- Traditional requirements on beam stability...

... to keep the beam in the pipe!

- LHC's increased stored intensity and energy
→ much tighter requirements on beam stability:
  1. Capability to control particle losses
     - Machine protection (MP) & Collimation
     - Quench prevention
  2. Commissioning and operational efficiency



- FBs became a requirement for safe and reliable nominal LHC operation
  – implications on controller reliability, availability and system integration

Expected Dynamic Perturbations vs. Requirements – or: Design Assumption vs. Operational Reality

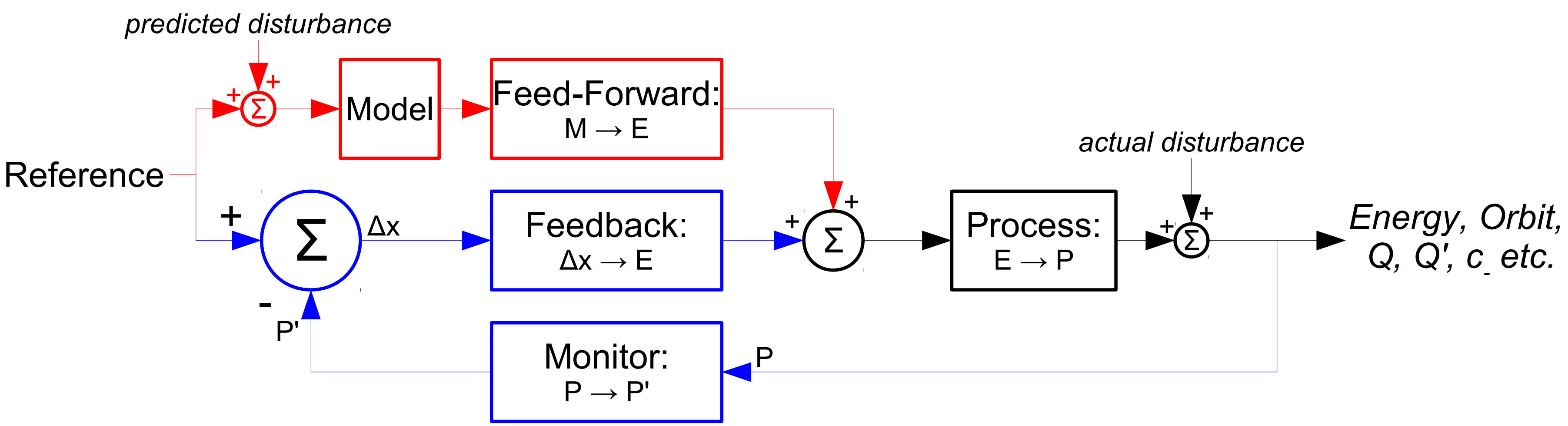
- From Decay/Snap-back expected dynamic perturbations

	Orbit [σ]	Tune [0.5·f_{rev}]	Chroma. [units]	Energy [Δp/p]	Coupling [c_]
Exp. Perturbations ('06):	~ 0.5	0.014	~ 70	± 1.5e-4	~0.01
Nom. Requirements:	± 0.15	±0.001	2 ± 1	± 1e-4	« 0.01
Achieved Stability ('13):	~ 0.1	~ 0.001	± 2 (7)	~1e-5	< 0.003

- Initial assumptions and plans (2006-2009):
  - Chromaticity considered as most critical parameter
  - FB Priority list: Chromaticity → Coupling/Tune → Orbit → Energy
- What turned out to be needed operationally
  - 2009 → 2011: **Tune** → Orbit & Energy/Radial-Loop → Q'(t) →…→ C^-
    - impressive Q'(t), C^- and beta-beat stability/reproducibility
  - In 2012: **Orbit** & Tune (snap-back, instabilities)
    - Higher energy & smaller-β* → much tighter collimator settings
      → convert smallest orbit deviations into losses/dumps

Control Paradigms I/III
Parameter control, either through...

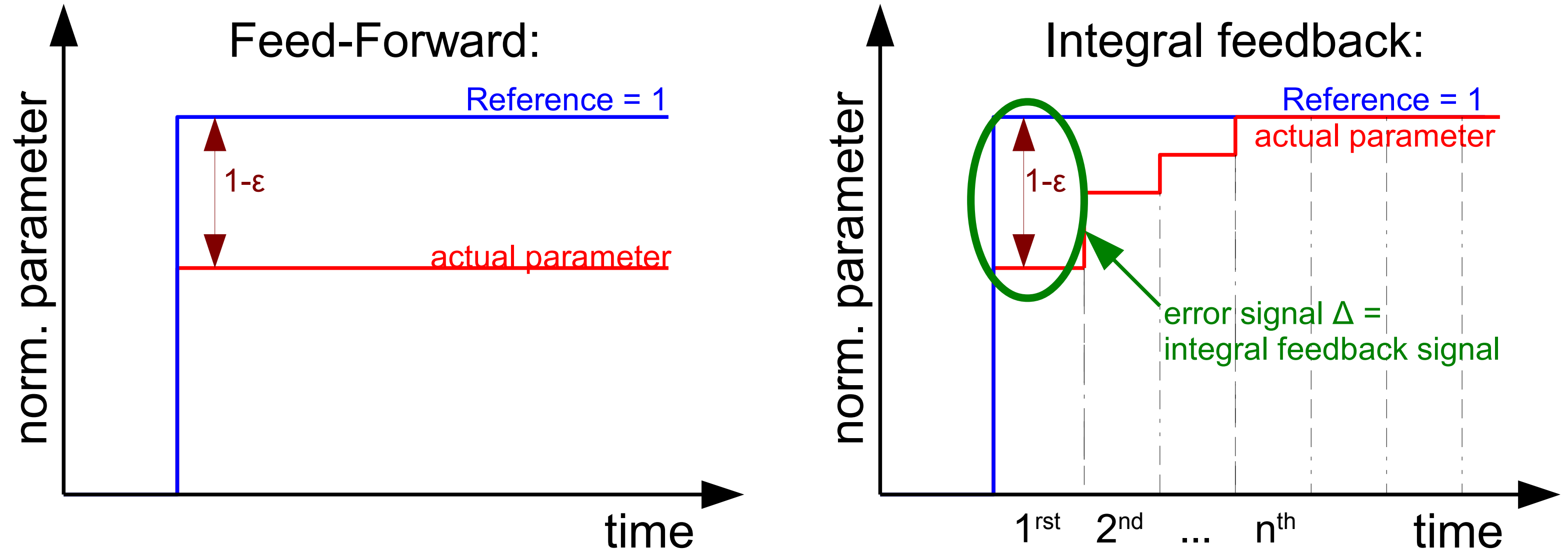
- Feed-Forward: (FF)
  - Steer parameter using precise process model and disturbance prediction
- Feedback: (FB)
  - Steering using <u>rough</u> process model and measurement of parameter
  - Two types: within-cycle (repetition Δt<<10 hours) or cycle-to-cycle (Δt>10 hours)



Control Paradigms II/III

- Machine imperfections cause steady-state offset ε_{ss} and scale error ε_{scale}:

$$\Delta x(s) = R_i(s)\cdot\delta_i \;\rightarrow\; \Delta x(s) = R_i(s)\cdot(\epsilon_{ss} + (1+\epsilon_{scale})\cdot\delta_i)$$



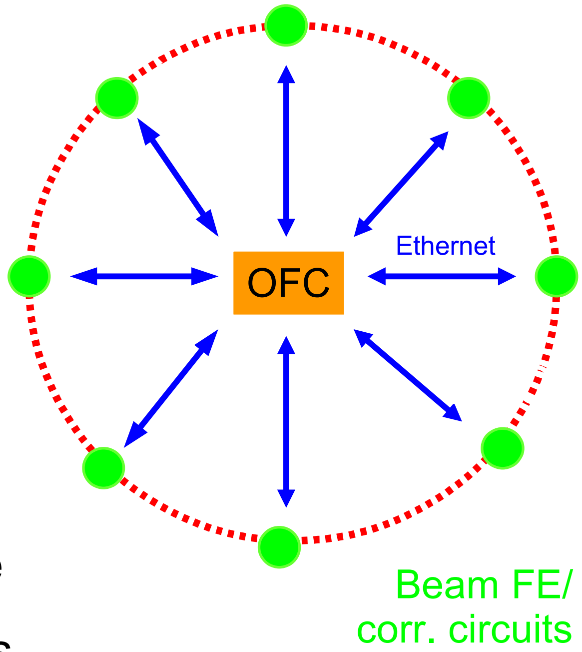
- Uncertainties and scale error of beam response function affects convergence speed (= feedback bandwidth) rather than achievable stability

Orbit-Feedback as Prototype for all LHC Beam-Based Feedback Systems

- Orbit-Feedback is the largest and most complex LHC feedback:
  - 1088 BPMs → 2176+ readings @ 25 Hz from 68 front-end computers
  - 530 correction dipole magnets/plane, distributed over ~50 front-end computers
  - Total >3500 devices involved

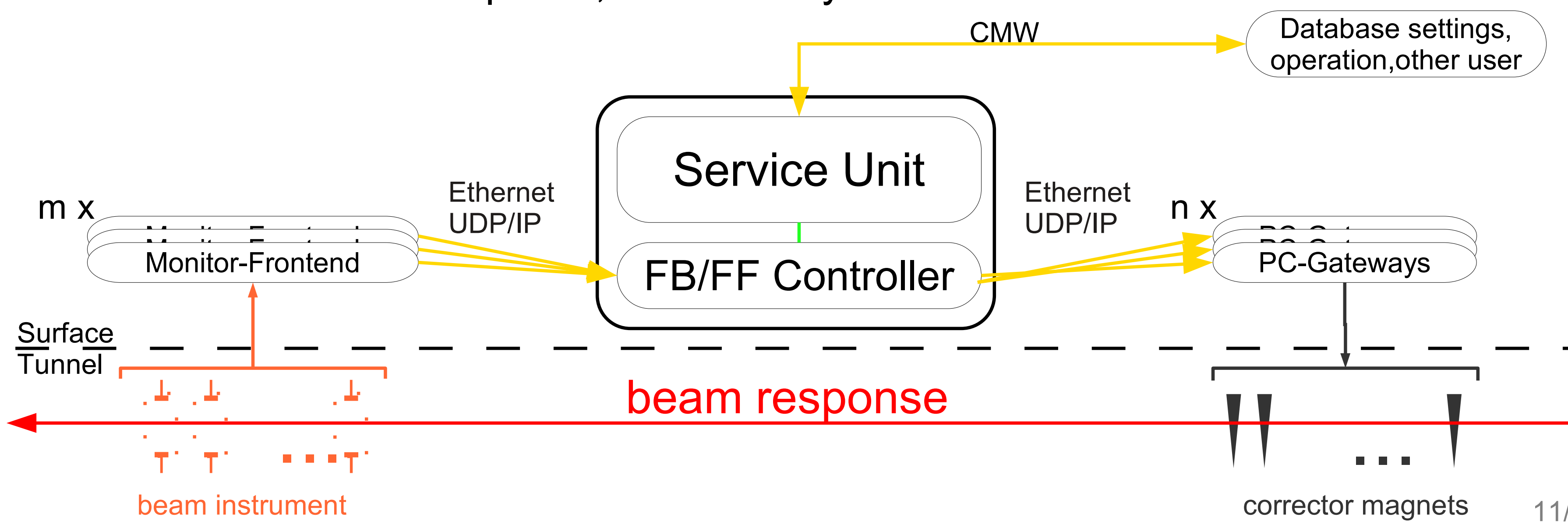
- Specific requirements fairly distributed
  → opted for central global feedback system

- One central controller (OFC + hot spare):
  - higher numerical load
  - higher network load (↔ ~120 front-ends)
  - dependence of machine operation on single device
  - easier synchronisation between front-ends and FBs
  - flexible correction scheme changes and gain-scheduling
  - most efficient to handle cross-talk and (de-)coupling between FBs



- Feedback Controller (OFC) performing actual feedback controller logic
  - Simple streaming task (10% of total load)
  - Beam data quality checks and real-time filtering (80% of total load)
  - Server running Real-Time Linux OS with periodic constant load
    - multi-core, highly redundant – MTBF > 22 yrs (spec, 120 yrs meas.)
  - Technical Network as robust communication backbone

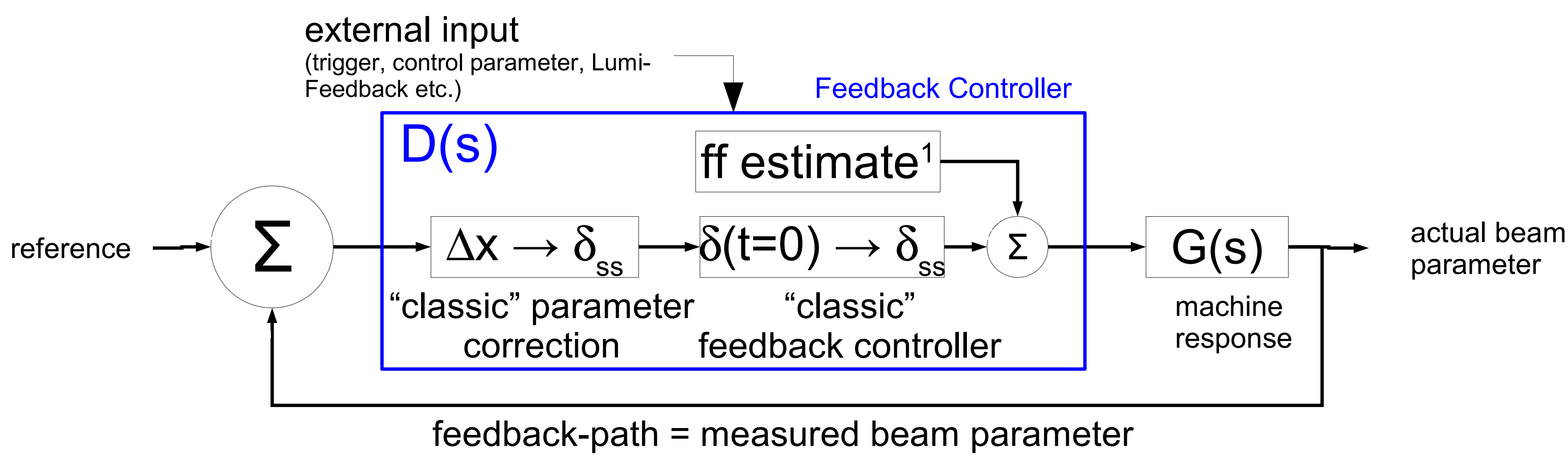
- Service Unit: Interface to high-level software control and interlock systems
  - Proxies user requests, handles asynchronous non-RT tasks



- 'Divide and Conquer' feedback controller design approach:
  1. Compute steady-state corrector settings $\vec{\delta}_{ss} = (\delta_1, \ldots, \delta_n)$ based on measured parameter shift $\Delta x = (x_1, \ldots, x_n)$ that will move the beam to its reference position for $t \rightarrow \infty$.
  2. Compute a $\vec{\delta}(t)$ that will enhance the transition $\vec{\delta}(t=0) \rightarrow \vec{\delta}_{ss}$
  3. Feed-forward: anticipate and add deflections $\vec{\delta}_{ff}$ to compensate changes of well known and properly described sources

space domain (SVD) — items 1

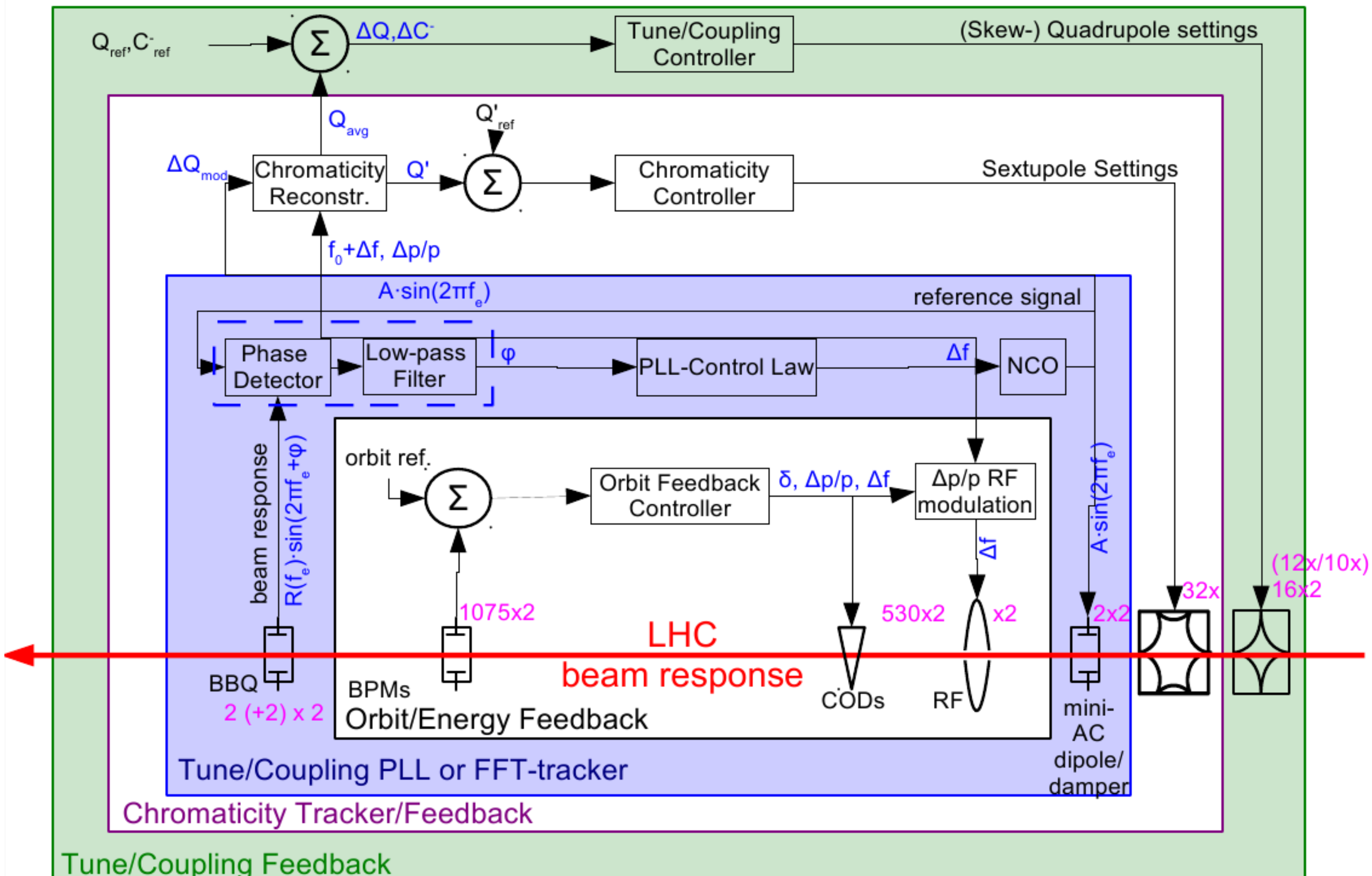
time domain — items 2–3



- (N.B. here G(s) contains the process and monitor response function)

To avoid inherent Cross-Talk between FBs…
… Cascading between individual Feedbacks

- Main strategy: derive measurement from FB control variable
  - Q'-tracker using '$Q_{raw} = Q_{meas} - Q_{trim}$'
  - Sub. Δp/p-mod. from Radial-Loop & Orbit-FB reference



further: $f_{BW}(PLL) \gg f_{BW}(Q') \geq f_{BW}(Q, C^-)$

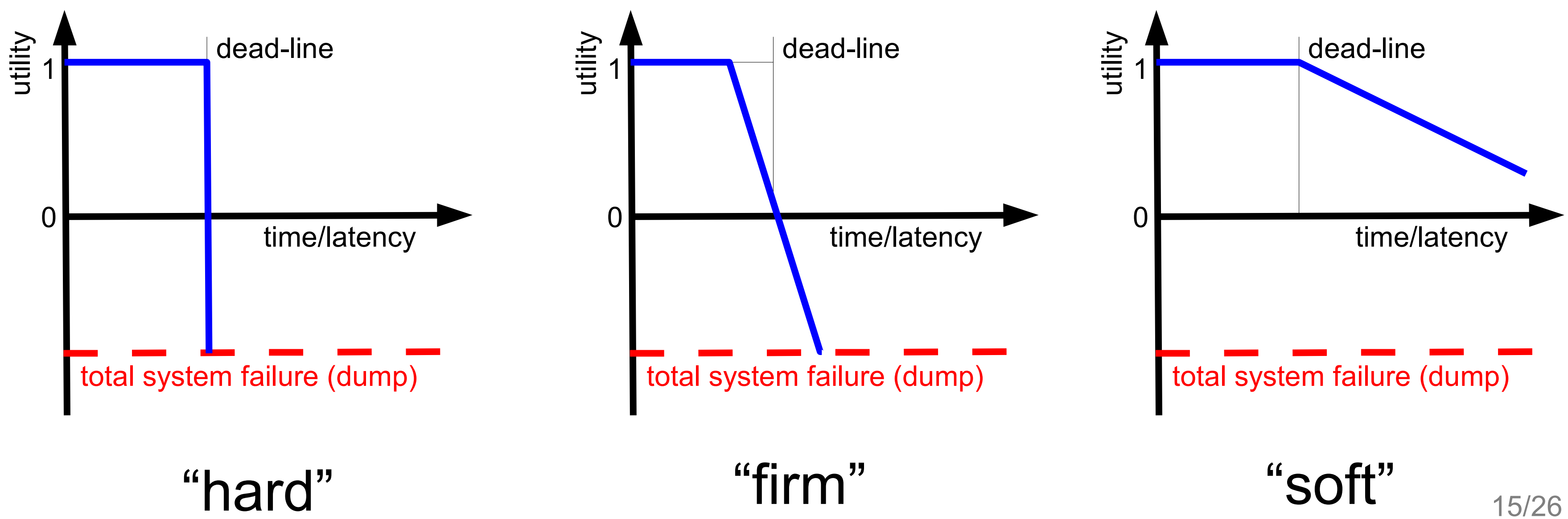
Why the notion/split between 'space' and 'time' domain?

- Separates specific accelerator physics from specific control theory
  - can test the two domains independently
- Multiple-Input-Multiple-Output (MIMO) in space-domain
  - Can modify correction algorithm without having to worry about whether overall loop remains stable
  - Maintains physical meaning of the individual control variables
  - Basically relying on inversion of response matrices → SVD
- Quasi-Single-Input-Single Output (SISO) in time-domain
  - Similar control problem/laws as e.g. for power converters
  - Time-domain controller identical for orbit, energy, Q/Q' vs. integrated/more complex 'Kalman' or 'Youla-Kucera-Klein'-based method
- Most[1] analog control loops are succeeded by digital controller:
  - Implies specific design to mimic the (non-)linear analog behaviour
  - Strong requirement for real-time control system!

[1]exception of high-speed and high-dynamic range

Digital Control System & 'Real-Time'

- … "A system is said to be real-time if the total correctness of an operation depends not only upon its logical correctness, but also upon the time in which it is performed. [..] are classified by the consequence of missing a deadline:
  - Hard – Missing a deadline is a total system failure.
  - Firm – Infrequent deadline misses are tolerable, but may degrade the system's quality of service. The usefulness of a result is zero after its deadline.
  - Soft – The usefulness of a result degrades after its deadline, thereby degrading the system's quality of service."



Some common Misconceptions/Fallacies about 'Real-Time' (RT)[1]

1. *“There is no science in real-time-system design”*
2. *“Advances in supercomputer hardware will take care of RT requirements.”*
3. *“[..] is equivalent to fast computing.”*
4. *“[..] research is performance engineering.”*
5. *“[..] systems function in a static environment.”*
6. *“[..] is assembly coding, priority IRQ programming, and device driver writing.”*
7. *“[..] all been solved in other areas of computer science or operations research.”*
8. *“It is not meaningful to talk about guaranteeing RT performance, because we cannot guarantee that the hardware will not fail and the software is bug free or that the actual operating conditions will not violate the specific design limits.”*

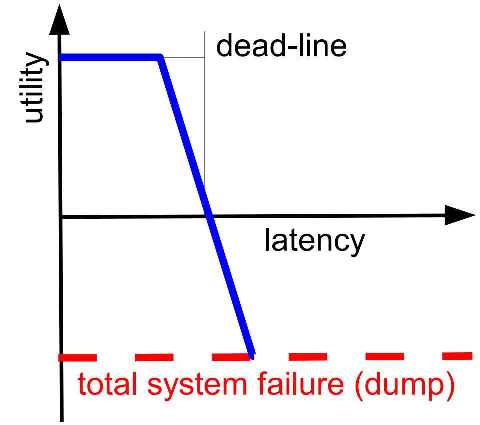
- Obviously, the above is wrong but seems to be sometimes forgotten when discussing the specific technical implications.

[1]John A. Stankovic, “Misconceptions about real-time computing: a serious problem for next-generation systems”, IEEE Computer, Vol. 21 #10, 1988

Control Paradigms III/III
Digital Control System & 'Real-Time'

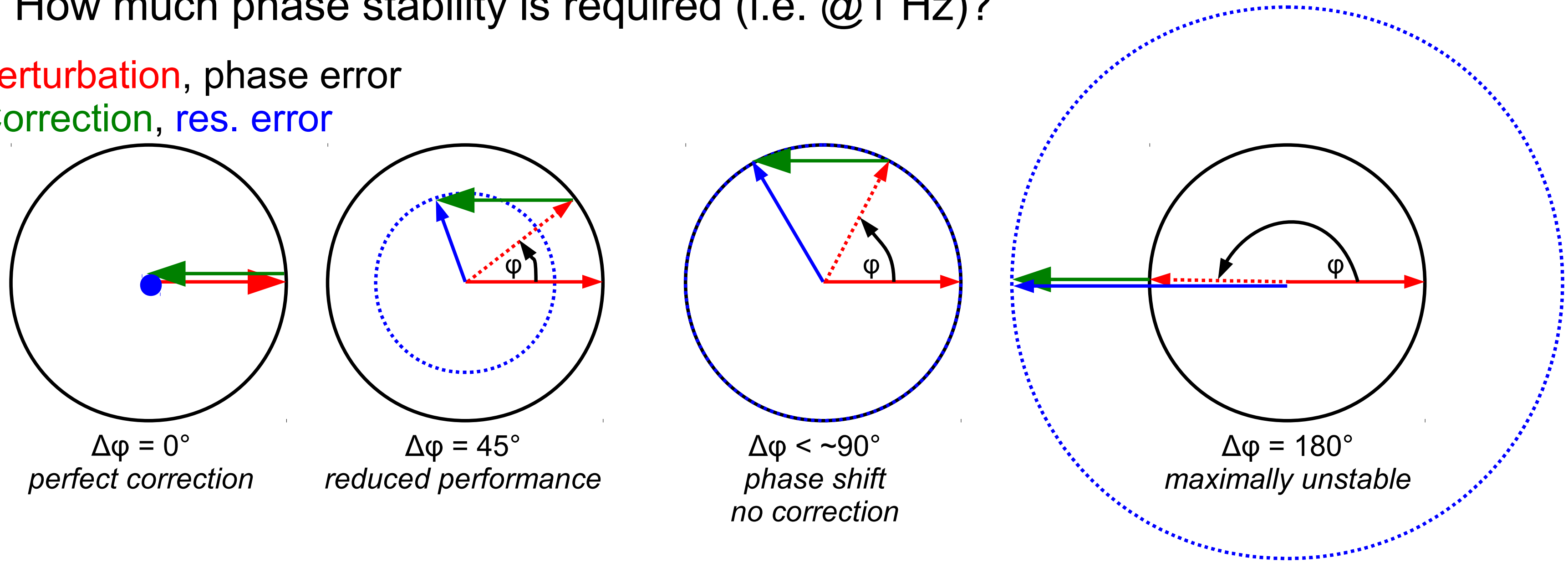
- LHC feedbacks are 'firm real-time systems'
  - some (limited) margin on occasional missing data
  - additional latencies are critical for loop stability, e.g. missing packet reduces phase margin by ~15°@1Hz (0° < stable < 90°< unstable < 180° – max. instability)

$$\Delta\varphi = 2\pi f_{bw} \cdot \Delta t_{delay}$$



- "How much phase stability is required (i.e. @1 Hz)?"

perturbation, phase error
Correction, res. error



Δφ = 0°
perfect correction

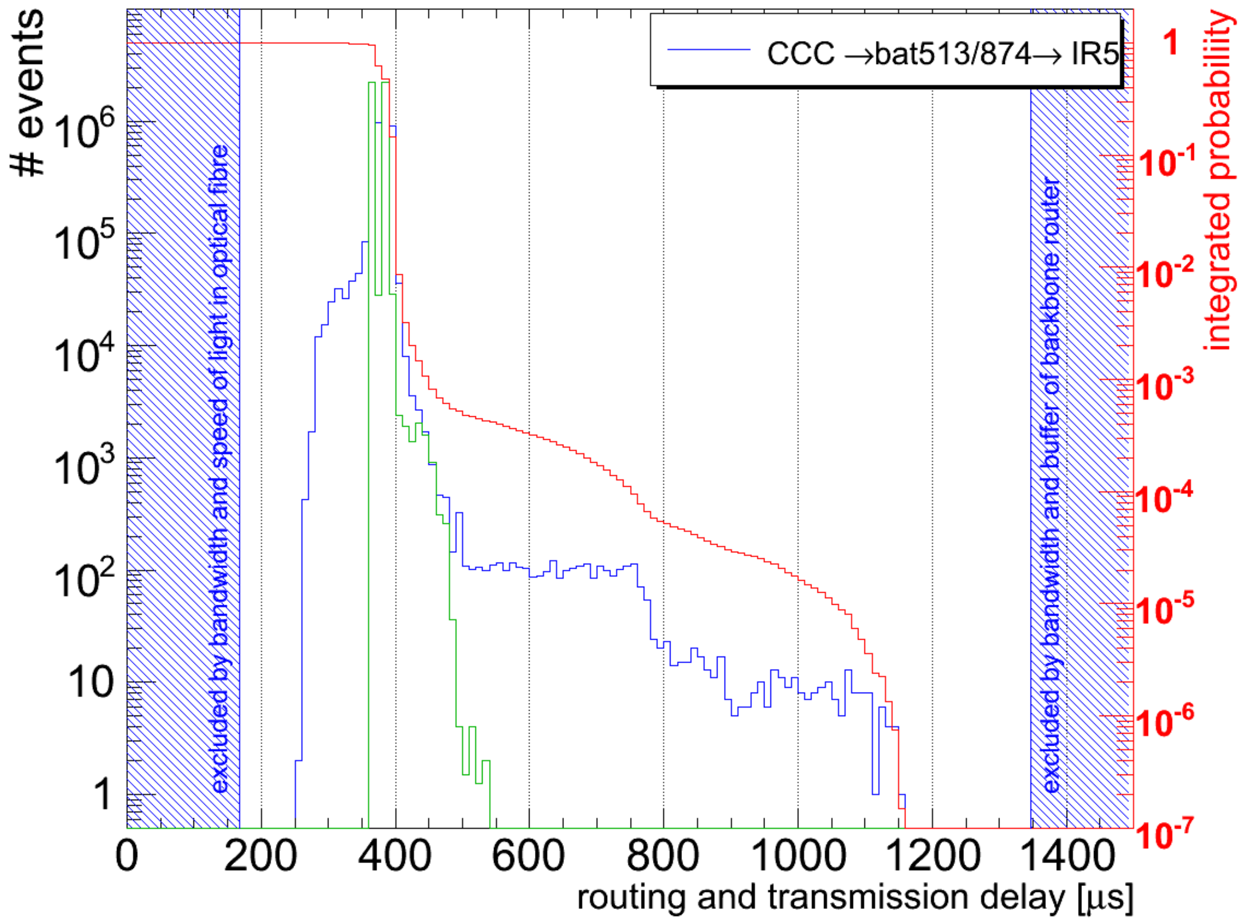
Δφ = 45°
reduced performance

Δφ < ~90°
phase shift
no correction

Δφ = 180°
maximally unstable

- For comparison: a missing/late data packet causes Δφ≈15° margin loss

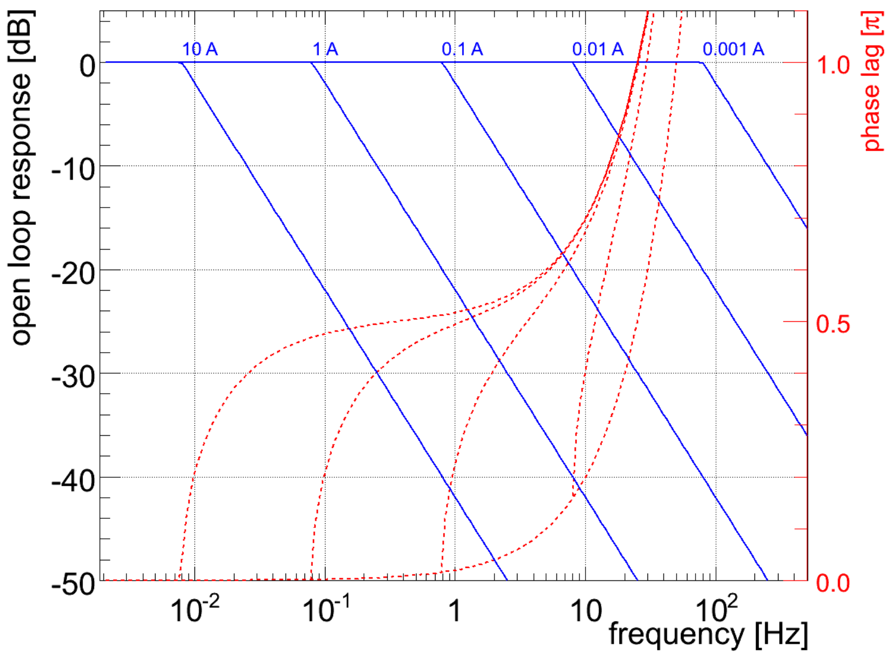
- TechNet round-trip tests: difference between standard and RT-Linux Kernel
  - Important: measure probability & upper-bound (worst-case) latency



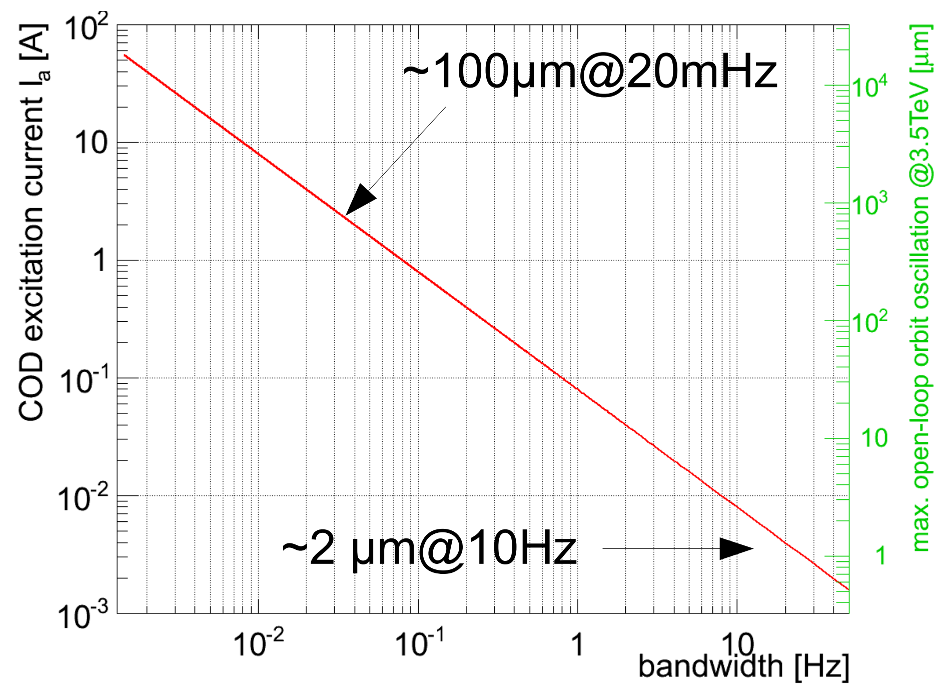
- Many similar test performed and basically excluded technologies such as: Java, non-RT Linux, FESA, CMW, TCP/IP, intermediate concentrators, ...

Orbit Feedback Bandwidth vs. Actual Orbit Perturbations

- Closed-loop bandwidth depends on the excitation amplitude
  - + non-linear phase once rate-limiter kicks in (rapid loss of phase margin!)



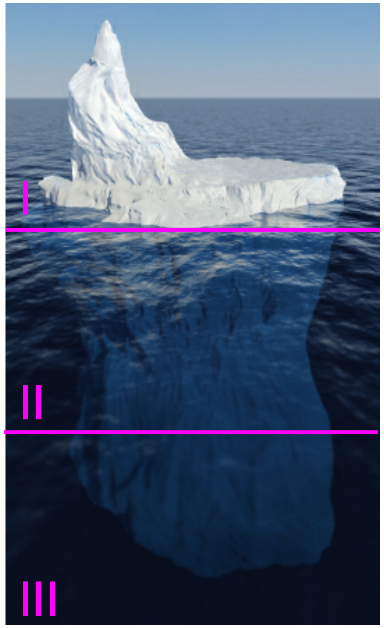
ΔI=0.1A ↔ Δx≈32 μm@β=180m



- Consider ~35μm@1Hz as effective bandwidth @4TeV (assuming 3C bump)

Architecture and Technology Choices

Aimed at keeping BE-CO standards and methodology as much as possible, but only if not at the expense of RT constraints and primary FB operation. Most choices driven by available technology in '04/05 and available BE resources (control theory/RT design/programing not as fashionable as FPGA/Java SW developer)



The good choices:

- Global- vs- local-FB control scheme
- Technical-Network vs. dedicated RT-network infrastructure
- UDP-based vs. CMW(/TCP)-based data transmission
- OFC data concentration vs. 'FEC → interm. conc. → OFC'
- CO deployment infrastructure (common-build, Java & C/C++, GUI frameworks, LSA, Japc, JDataviewer, SDDS, acc-co CVS/SVN)

The bad, less-good, or debatable choices:

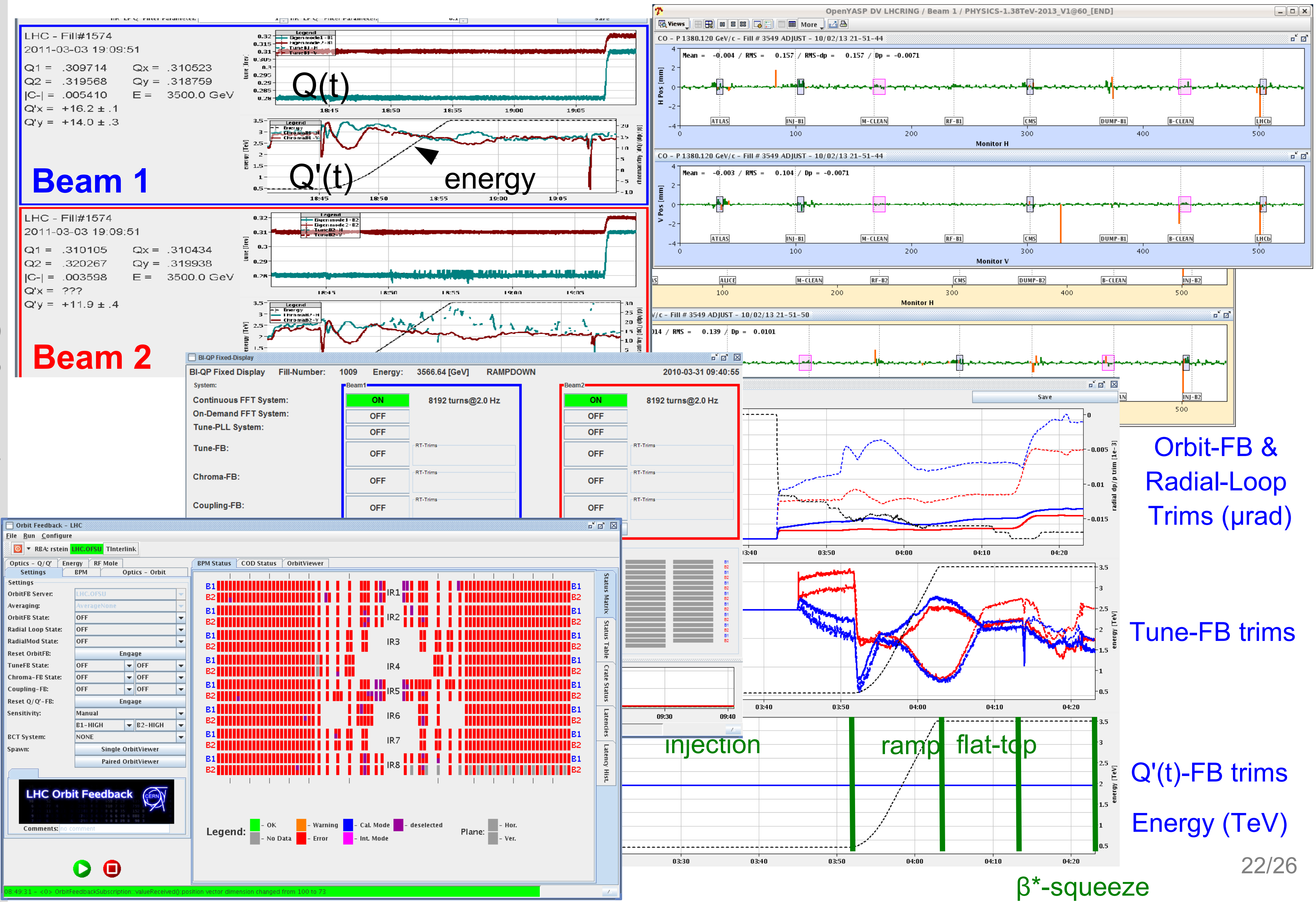
- SW-based (firm-RT) vs. HW(/FPGA)-based (hard-RT,) controller implem.
  - Real-Time- vs. standard Linux kernel
  - CO-IN Proliant Server (CPU, 2 NICs) vs. CO-FE front-end computer
  - OFC-FESA-free ↔ OFSU-FESA (thready safety, RT-latencies, CMW vs. RT)
  - ROOT (I/O streamer, C/C++ coding standard, math, routines)
- Using 'Mix-&Match-' vs. CO-consistent standard (maintainability)

Feedback Sub-Projects: What they do and where to find them...

- In svn+ssh://svn.cern.ch/reps/acc-co/lhc/lhc-feedbacks/
  - lhc-app-orbit-feedback-controller – *the actual feedback controller (aka. OFC)*
  - lhc-lib-feedback-commonalities – glue between various OFC parts and OFSU
    - initially separate feedback controller planned → turned out that this is not possible/recommendable but kept stuff in library to minimise profilling and debugging overhead (rarely changes)
  - lhc-lib-twissoptics – physics/optics related code, not FB dependence per se
  - lhc-lib-twissoptics-examples – examples, documentation and unit-type tests
  - lhc-orbitfeedback – *the OFC/OFSU graphical expert user interface*
  - lhc-app-[orbit/tune]-feedback-serviceunit -- *an orphan FESA class*
  - lhc-orbitfeedback-datamanager -- *reference orbit/sequencer (Kajetan)*
  - lhc-orbitfeedback-services -- *reference orbit/sequencer (Kajetan)*
  - optics-server – *LSA-OFSU link to transfer machine optics data (MAD-X style)*

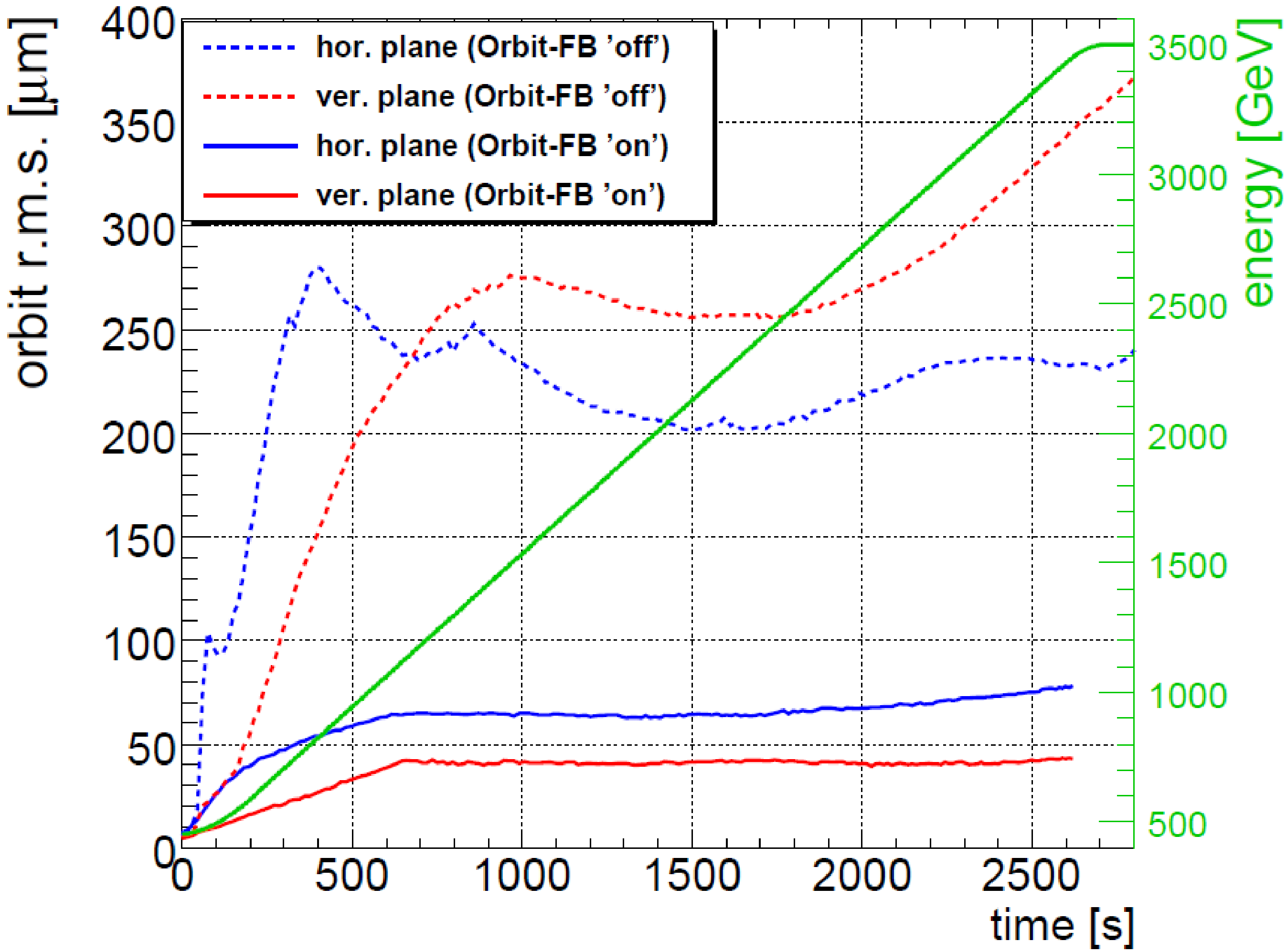
- noteworthy exceptions – *Orbit, Q/Q' related GUIs*:
  - svn+ssh://svn.cern.ch/reps/acc-co/accsoft/steering/
  - svn+ssh://svn.cern.ch/reps/acc-co/lhc/lhc-biqp-fixdisplay/
  - svn+ssh://svn.cern.ch/reps/acc-co/accsoft/tuneviewer

LHC Feedback Operation – Example I/IV
… one of the more visible systems in the control room



LHC Feedback Operation – Example II/IV

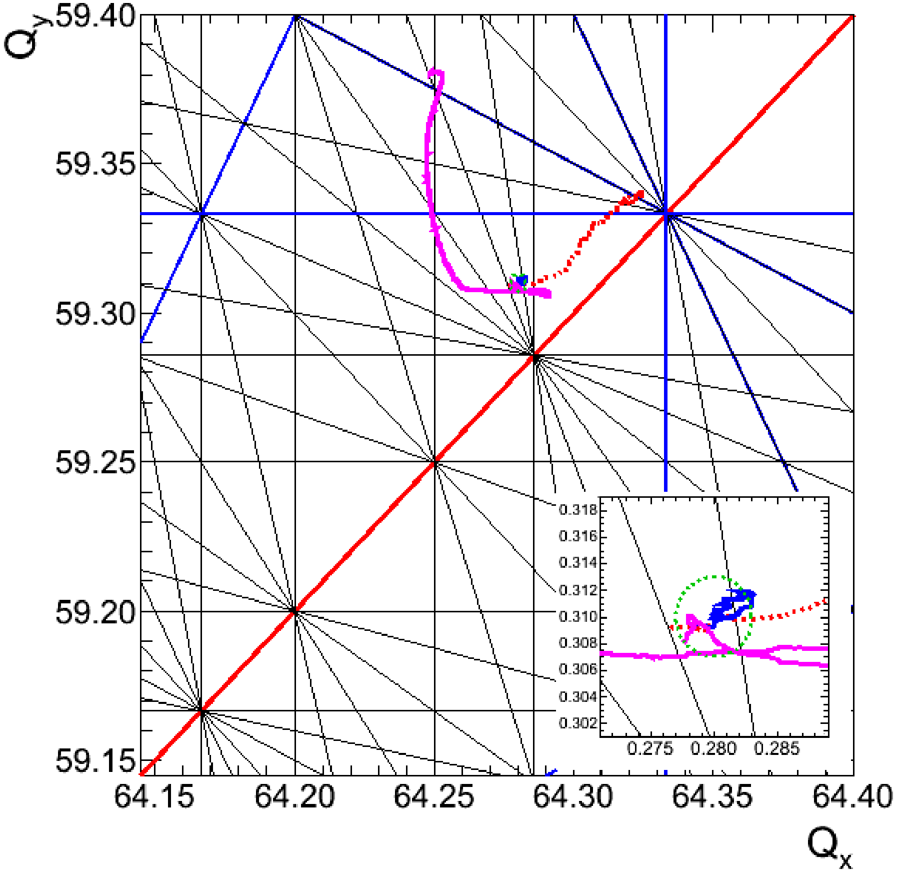
- Orbit feedback used routinely and mandatory for nominal beam

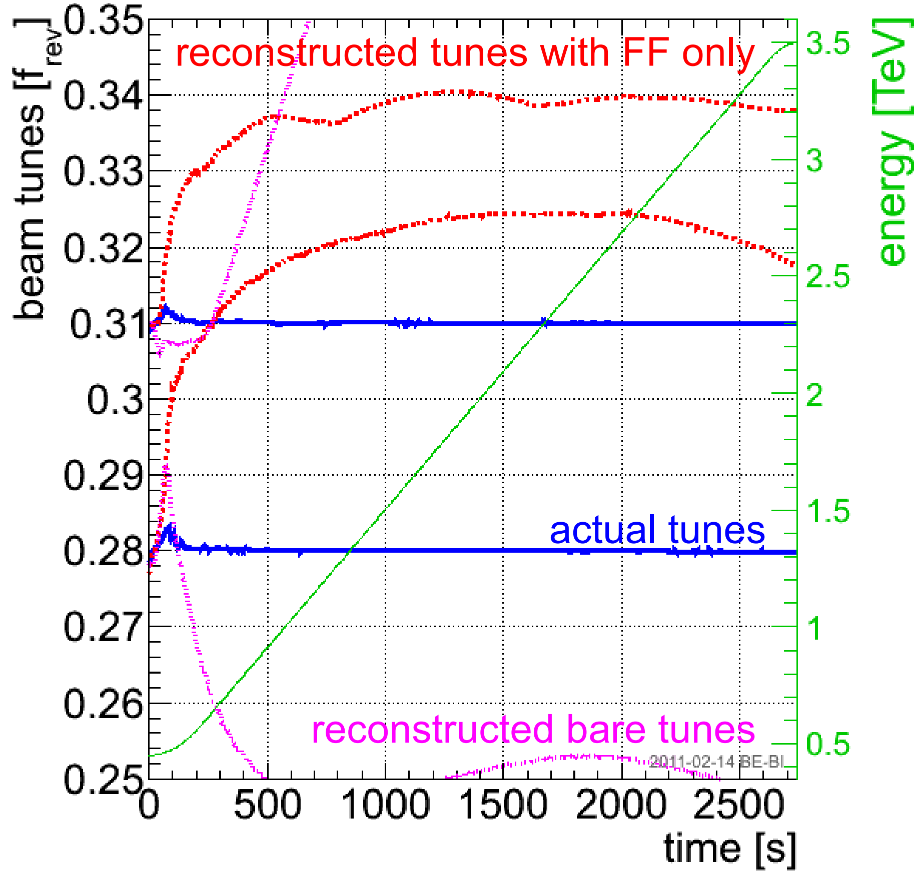


- Typical stability: 80 (20) µm rms. globally (arcs)
- Most perturbations due to Orbit-FB reference changes around experiments

LHC Feedback Operation – Example III/IV

- Tune-FB driving and accelerating early commissioning in 2009-2011
  - Tunes kept stable to better than 10^{-3} for most part of the ramp and squeeze

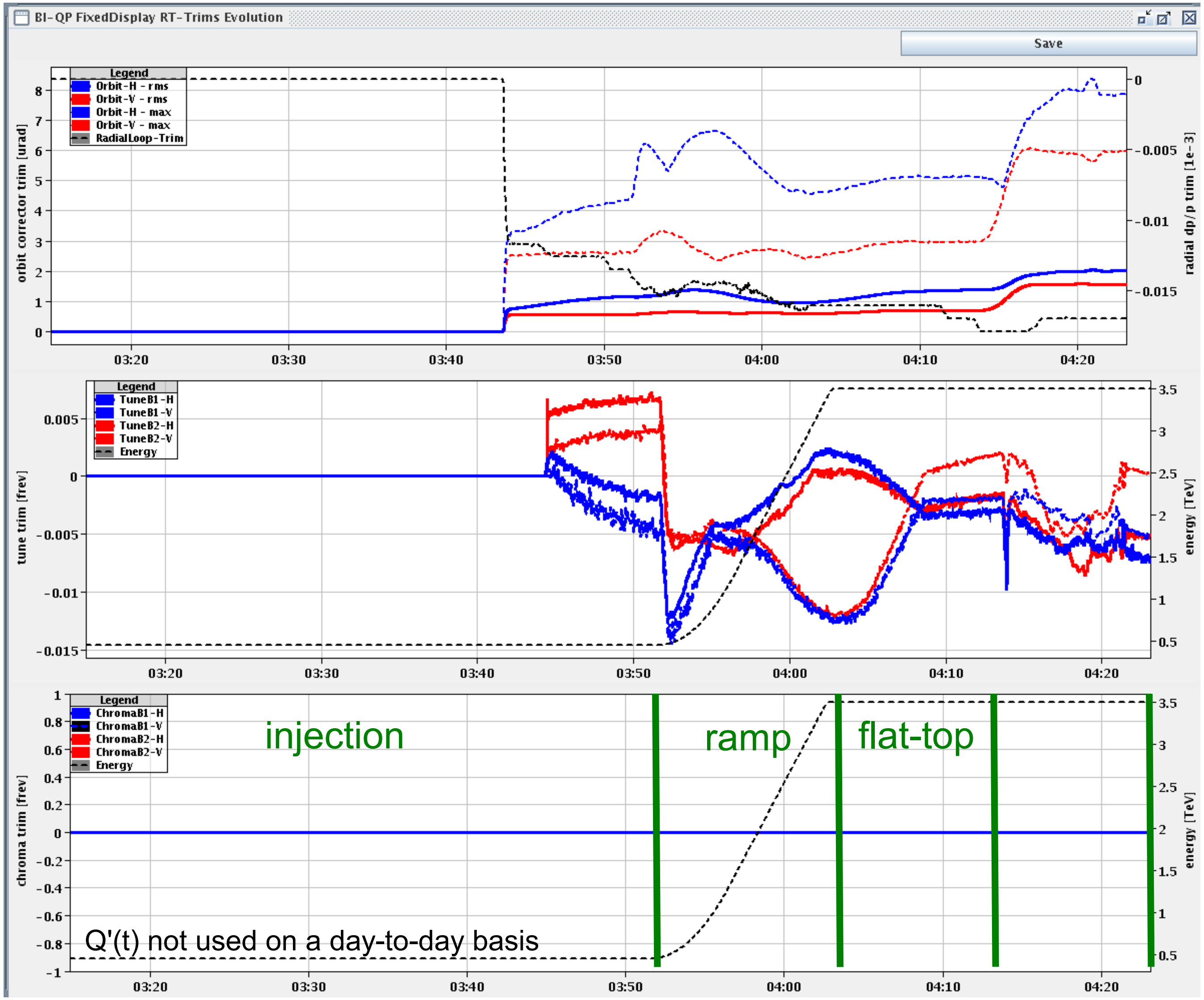




- Tune-FBs most useful and needed during the early (re-)commissioning

LHC Feedback Operation – Example IV/IV

- Trims became de-facto standard to assess the FB and machine performance



β*-squeeze

Orbit-FB & Radial-Loop Trims (μrad)

Tune-FB trims

Q'(t)-FB trims

Energy (TeV)

Architecture Summary

- Generally, feedback performed their designed job. Pushing LHC machine parameter envelope also implied increased performance constraints on Feedback operation (notably orbit stability during squeeze)
  → Need to improve FB sub-systems to keep up with LHC progress post-LS1

- Present architecture and design is based on on …
  - evolution and series of tests, reviews and iterations (2013: 10+ years)
  - working experience and knowledge derived from synchrotron-light-source community (orbit, energy), and Hera, RHIC & Tevatron (Q/Q')

- Main paradigms:
  - Central simple input-processing-output feedback controller (OFC)
  - Managed by service unit (OFSU, settings management, data proxy)
  - LHC Technical-Network as communication backbone
  - 'Firm real-time' constraints using Real-Time capable Linux

→ Second talk tackles specific feedback implementation

Appendix

Real-Time System Design: Specification & Verification

As for documentation, need to consider RT constraints during design phase:

- Numerical complexity and it's variability:
  - fixed numerical complexity, i.e. no loop dependencies on conditional variables (non-RT examples: χ^2-fitting, MICADO, any N-length queue, …)
    - 80% of the LHC feedbacks is about what can go wrong (filters, data integrity checks, exception handling etc.)
  - worst-case latencies of a library function (particularly after a performance update) → reduce dependency on unknown/less-controlled systems
- Measure how well given dead-lines are kept
  - Real-time scheduling & operating systems
    - What is executed in parallel (driver, other services)
    - CPU-shielding: fix threads to given CPU (avoids context switches)
      - reserve one core for dynamic and/or non-RT tasks
  - quantitative upper-bound execution times for all external conditions (i.e. load conditions, 'if-else' sub-branches, failure scenarios)

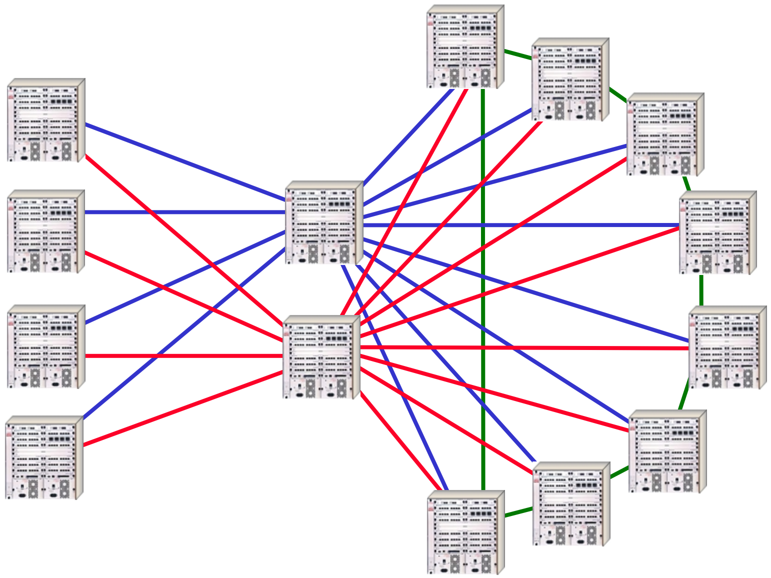
Estimated Closed Loop Latencies

- Estimated average delays:

	Estimated '04	Achieved '12
	[ms]	[ms]
BPMs	5-10	20-40
Network/inbound	1	1
UDP reception	30	10
Correction	10-30	<10
Service tasks (OFSU, optics re-calculation)		10
Network / outbound	1	1
FGC gateway control	30-40	20
Total	**80-120**	

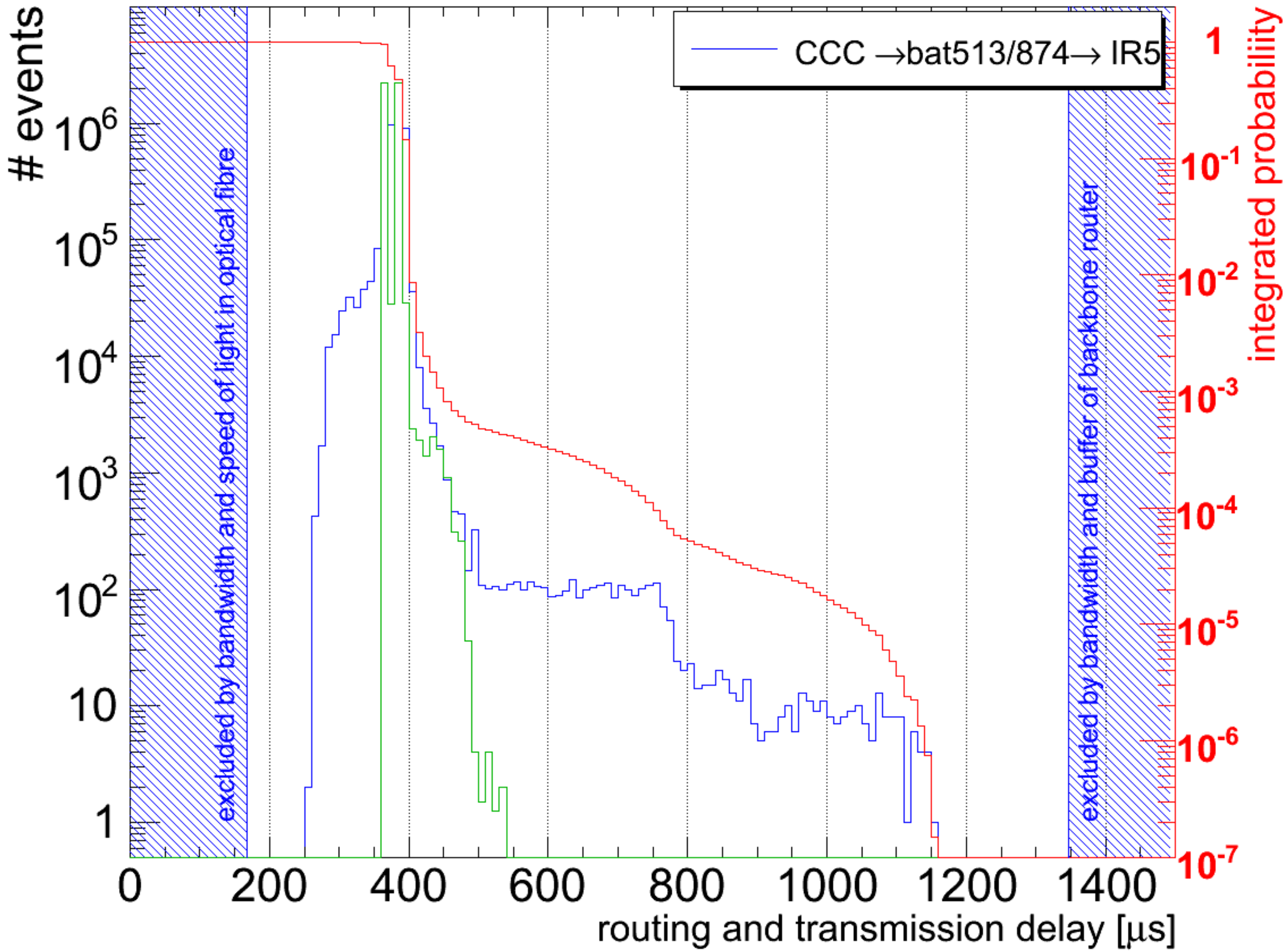
- Just acceptable if you consider the PC limits of 1 Hz.
- For a 25 Hz sampling rate, this is already > 1 period!
- Most issues related to non-real-time FE behaviour!

Technical Network and Data Communication I/II



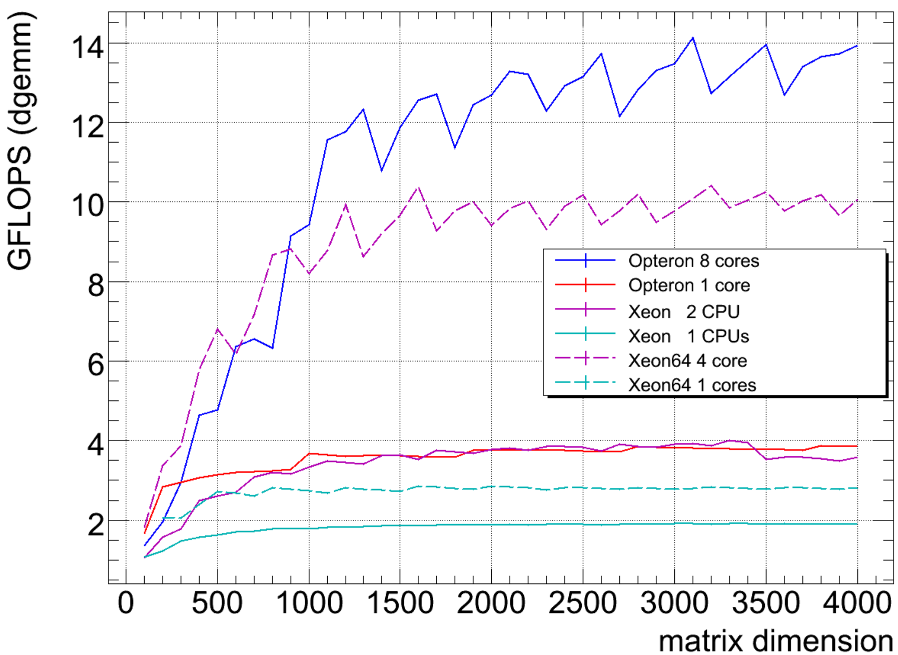
- CERN's Technical Network as backbone
  - Store & Forward switched network
    - no data collisions/data loss
  - double (triple) redundancy
- Core: “Enterasys X-Pedition 8600 Routers”
  - 32 Gbits/s non-blocking, $3 \cdot 10^7$ packets/s
  - 400 000 h MTBF
  - hardware QoS
    - One queue dedicated to real-time feedback
    - ~ private network for the orbit feedback
      - Initially skipped (gain experience/see whether it's really necessary) → now: plan to deploy post-LS1
- Routing delay ~ 13 µs
- longest transmission delay (exp. verified) ~ 320 µs
  (500 bytes, IP5 -> Control room ~5 km)
  - 80% due to traveling speed of light inside the optic fibre
- worst case max network jitter « targeted feedback sampling (25 Hz)!

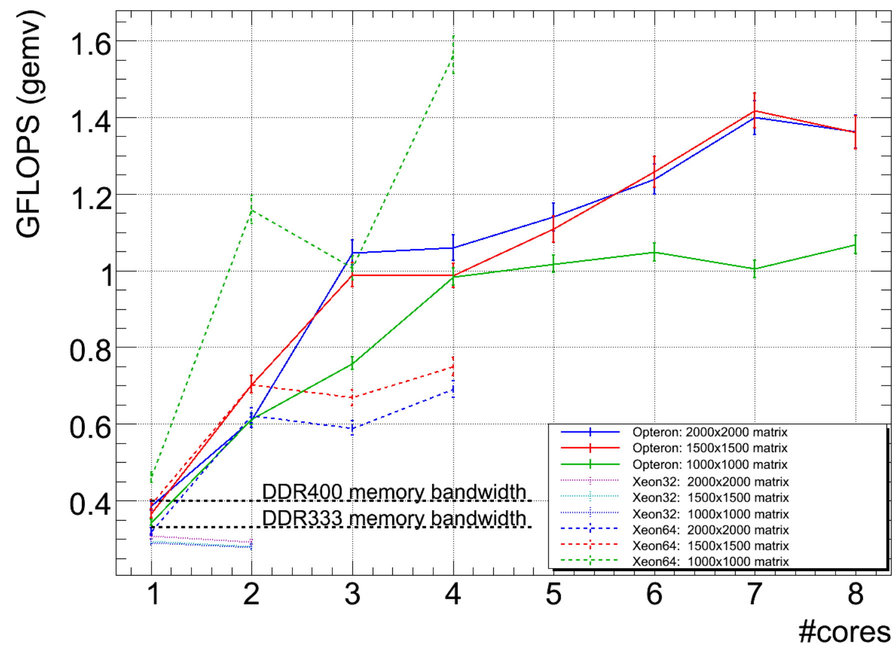
- TechNet round-trip Tests: Difference between standard and RT-Linux Kernel:



Technology Choices – CPU Platform

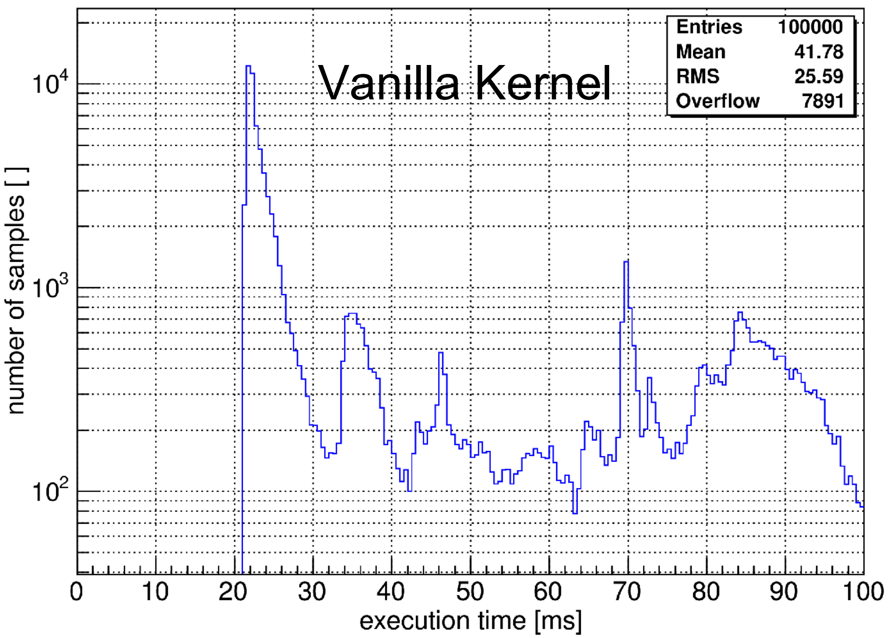
- Choice of using HP Proliant Server was driven by required
  - CPU performance, memory bandwidth & network performance (2 NICs)
  - add. CPUs/cores allowed for 'shielding' of RT- from non-RT tasks
  - available HW in 2006 (FPGAs too small at that time)
- Need information from all BPMs → central controller a logical choice → reliability must not become a single point of failure (Proliant's MTBFs of 120yr)

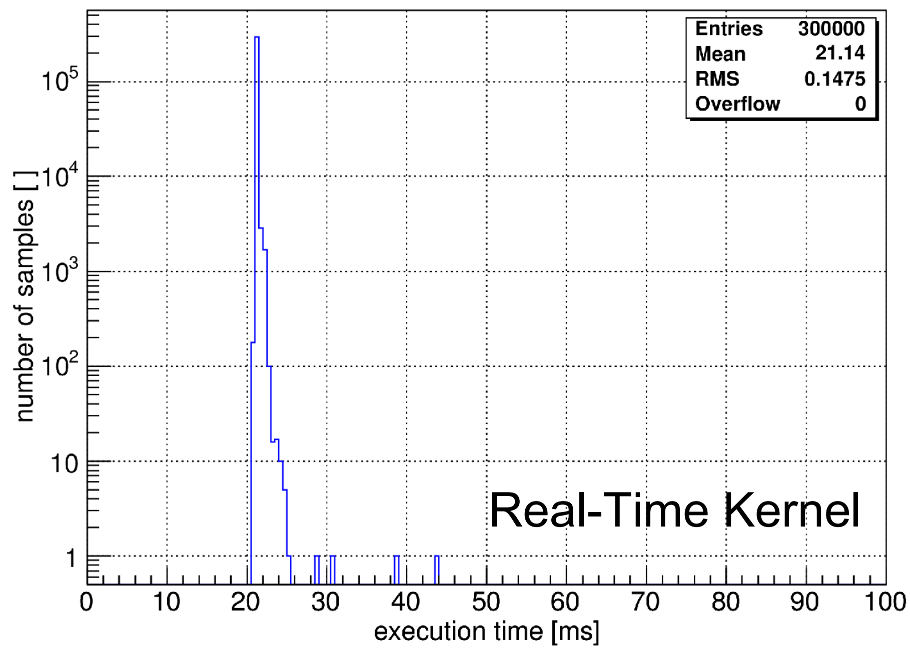




Technology Choices – Real-time vs. Standard (Vanilla) Linux Kernel

- Strong requirements on Constant closed-loop delays
  - Needed to deviated from the supported standard BE Linux installation
- OFC loop stress tests under IO, CPU and network load:





  - Which one would you chose from an RT perspective?

Technology Choices – CMW (TCP) vs. RT (UDP) latencies @32 Hz (RTC) & Java

- Strong requirements on Constant closed-loop delays
  - Needed to deviated from the supported Common-Middle-Ware
  - TCP or any derivatives can block → violated RT constraints

- Early CMW test: no-load condition (loaded: jitter up to few tens of seconds)