For consideration …
Interoperability between CMW & Real-Time Communication

A) Communication prioritisation (linked to CMW review items C2, SR1, SR11&12)

- Exploit TechNet Quality-of-Service (QoS) queues to differentiate between operational and non-operational CMW communication
- CPU shielding – affinity settings of CMW threads

B) Non-blocking front-end-to-client communication
(linked to CMW review items G5, SR2)

- TCP an inherently blocking protocol, UDP would provide non-blocking FE operation, and failure-tolerant/off-load communication problem of 'unresponsive clients'

Communication Prioritisation
(CMW review items C2, SR1, SR11 & SR12)

- Has to be deployed on every OSI layers to be effective to mitigate latency/performance bottlenecks:

1) Network → decided to deploy TechNet QoS option for FB's RT communication (based on TN switch overload experience)
   - Mission critical to minimise cross-talk between RT and other TN data communication
   - IT suggested/is able to support four 'differentiated service' queues (controlled via UDP/TCP DS field (6 bits) – RFC3260 standard):
     - IT-internal – needed for TN administration
     - FB Real-Time: BPM/BBQ → OFC → FGC-Gateways (UDP only)
     - Op-critical: CMW communication of devices/clients that could impact/cause LHC down-time (i.e. Cryo, SIS, FGCs, OFSU, …)
     - Others: default queue, and explicitly set for non-op accounts, GUIs etc.
   - Deployed for FGCs/OFC → main item tbd.: similar/consistent implementation in CMW?
2) Application Layer → CPU shielding (N.B. massively used in OFC)
   - CMW server/client thread affinity kept on different CPU than RT-critical tasks/communication

Non-blocking FE-to-Client Communication I/II
(CMW review items G2 & SR5)

- TCP is an inherently blocking protocol (due to 'handshake')
  - right choice for guaranteed data delivery (i.e. settings transmission)
  - can fine-tune performance but ultimately cannot prevent locking of server/network/client bottlenecks
- Many of BI/FB data streams have high update rates & real-time requirements:
  - "... total correctness of an operation depends not only upon its logical correctness, but also upon the time in which it is performed." (see appendix)
  - i.e. better drop a late data packet in favour of the latest update (i.e. no FIFO queuing)

  → reason why LHC FBs deploy UDP-based data transmission
- UDP-solution works, but present implementation is too application specific to be used generically in other use-cases that would benefit from this
  - Technology was driven by necessity in view of the FB application but is generic enough to be covered by the CO middle-ware infrastructure
  - For consideration: if this functionality could be incorporated into the new CMW, it would improve the performance and ensure common coding practises, better deployment/maintainability, etc.

Non-blocking FE-to-Client Communication II/II
(CMW review items G2 & SR5)

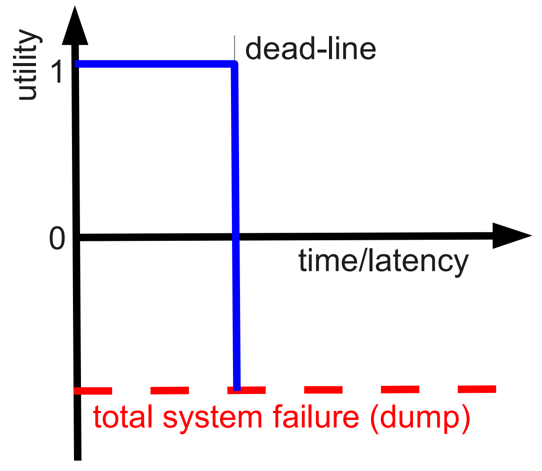
- Most demanding but exemplary use-case: BBQ and BPM Data transmission (BQBBQLHC, OFSU, others use-cases are fairly similar)
  - Receiving/Publishing frequency: (1), 2.5, 10, 25 Hz
  - Avg/Max message latency: firm- (if-possible, C/C++ only) to soft- real-time
  - Avg/Max message size: 200-250 kByte/iteration
  - Required data types inside a message: standard CMW supported fields (+ raw data?)
  - Any other constraints, limits, etc.: C++/Java implementation
- One possible implementation:
  - Use standard CMW-TCP link to maintain session details (ie. priority, alive-status, UDP destination)
  - Server: use CMW-streamer to serialise data, encapsulate into UDP jumbo packet and send via (prioritised) I/O socket
  - Client: standardised UDP reception with (ideally) same abstraction as present CMW/JAPC

CERN

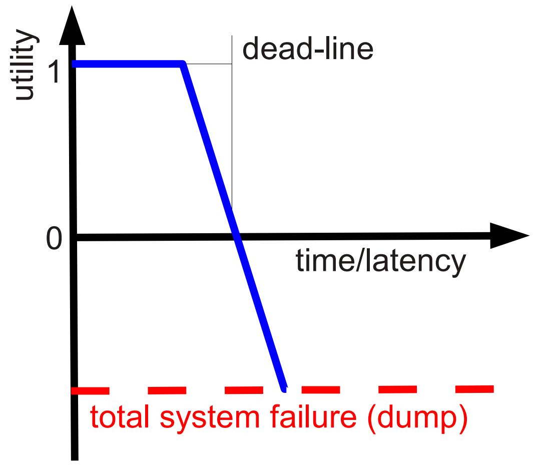
www.cern.ch

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

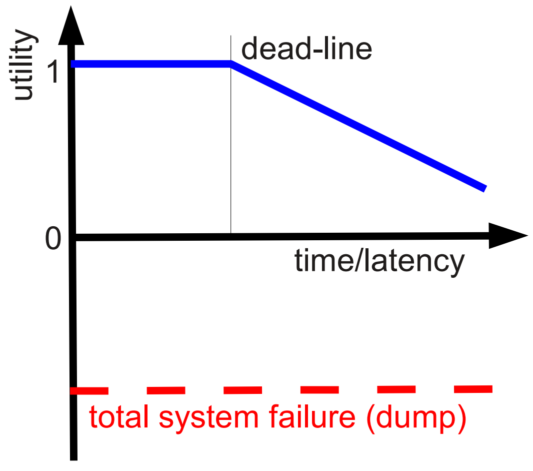
- … "A system is said to be real-time if the total correctness of an operation depends not only upon its logical correctness, but also upon the time in which it is performed. [..] are classified by the consequence of missing a deadline:
  - Hard – Missing a deadline is a total system failure.
  - Firm – 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."



"hard"



"firm"



"soft"

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

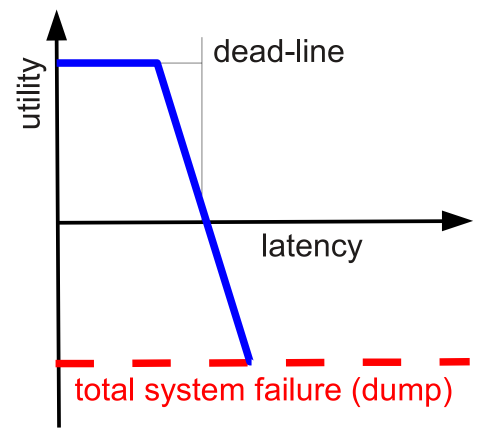
1. *“There is no science in real-time-system design”*
2. *“Advances in supercomputer hardware will take care of RT requirements.”*
3. *“[..] is equivalent to fast computing.”*
4. *“[..] research is performance engineering.”*
5. *“[..] systems function in a static environment.”*
6. *“[..] is assembly 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.

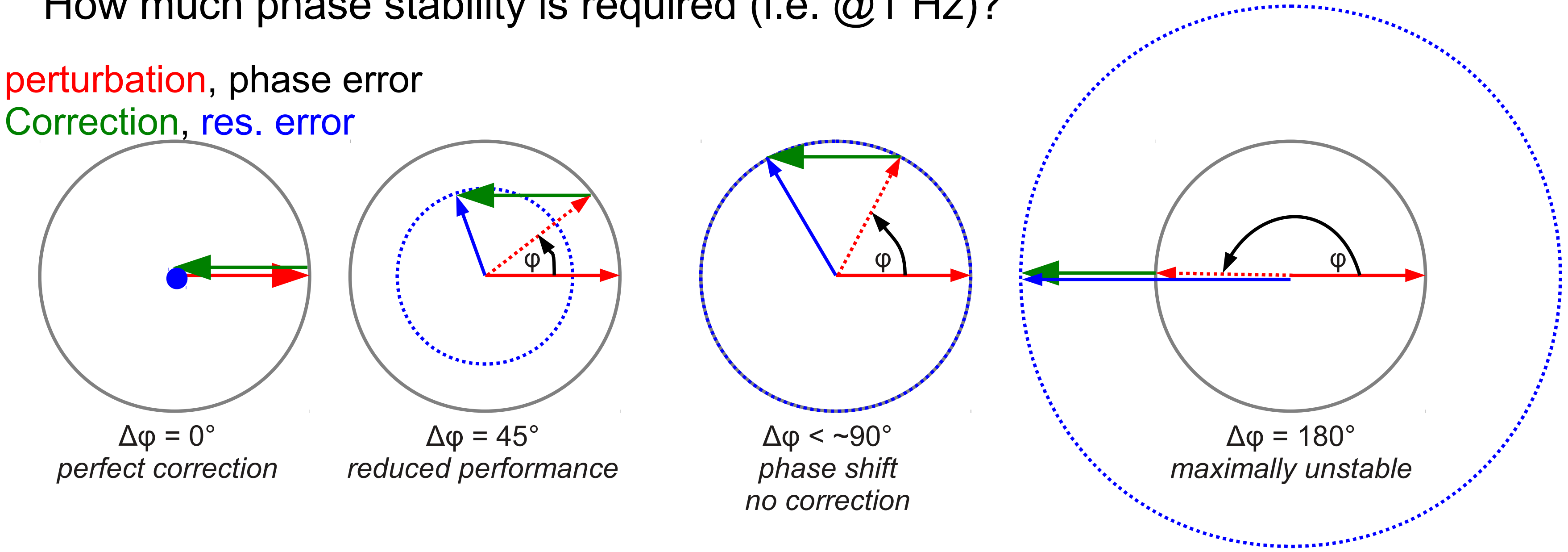
[1]John A. Stankovic, “Misconceptions about real-time computing: a serious problem for next-generation systems”, IEEE Computer, Vol. 21 #10, 1988

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)



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



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