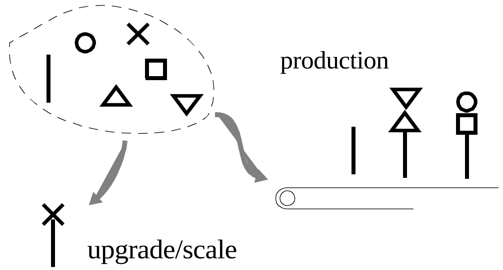
An overview of Quality of Control

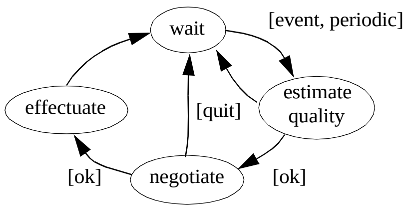
Martin Sanfridson
Mechatronics Lab
Machine Design, KTH
email: mis@md.kth.se

Why Quality of Control?



- Benefit: flexibility for the customer.
- Issues: **1)** activation interval, **2)** mix of applications, and **3)** example architecture.

Activation of QoS management

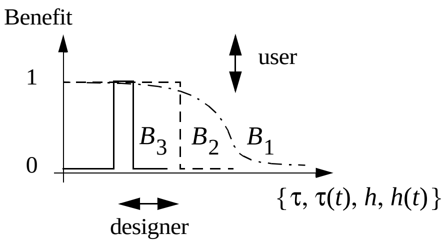


- The computational cost of estimation and negotiation should be considered.
- Too frequent modifications might introduce a new mode.

Activation and purpose

Activation	Purpose	Information needs
One shot, at **configuration** time	Assembly of modules Upgrade and repair	Can be manual (designer) or automatic (user), always pre-runtime, at power on. The future real-time behaviour and quality must be predictable. Test runs can be performed by engineer.
Event triggered, during normal operation	Scalability Mode changes	The future real-time behaviour and quality must be predictable. Relies primarily on first knowledge, not measurements. A possibility to resort to default worst-case values in case of malfunction.
Periodic, during normal operation	Stabilizing	The real-time behaviour is measurable and quality is possible to gauge. Statistical data can be used to capture the current status and feedback can be based on heuristics.

Quality measure, mix of applications



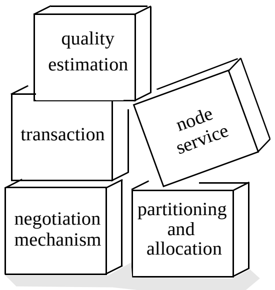
$$B_{tot} = \sum_k w_k B_k$$

- Applications: **controller**, MMI, logging, emergency stop, production management, support tools, etc.
- Benefit function. Hard deadlines are not always necessary.
- Optimization or **feasibility** problem.

Case study: middleware for OSE Epsilon

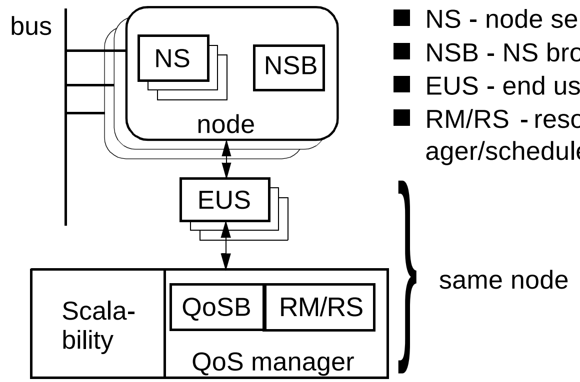
- The target system is a **robot arm** with three links. Each link has a node with a microprocessor, encoder and drive unit.
- OSE and the CAN bus use **fixed priorities**.
- The computer system is treated as **soft** and "unscheduled". Task chains are time stamped.
- The OSE **link handler** makes the partitioning transparent in a distributed system.
- The QoS manager is build on top of a **scalability layer**.

Building blocks



- Foundations for an architecture aimed at scalability and negotiation of resources.

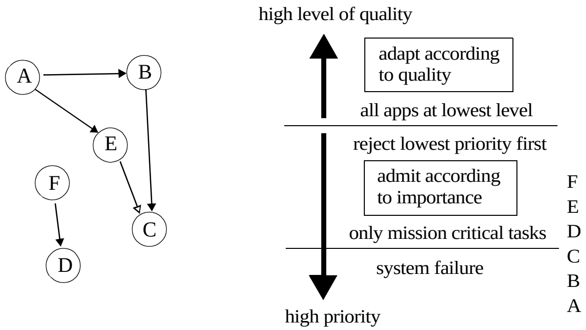
An overview of the distributed system



- NS - node service
- NSB - NS broker
- EUS - end user service
- RM/RS - resource manager/scheduler

- Link handler with CAN protocol.

Admission control



- Importance vs. timeliness.

Other issues

- Model based estimation of weighted state **covariance** to capture the timely behaviour between sampling instants.
- **Open** systems e.g. QoS for multimedia over the internet, vs. **embedded** systems where admission is controlled.
- Scalability vs. QoC. Flexibility vs. **safety critical systems**.
- **Best-effort** and the means to control the quality after negotiation.
- Soft vs. hard real-time. Guaranteed to achieve specified quality level or **guaranteed** to meet every specified deadline?
- Choice of **scheduling strategy**.
- Distributed negotiation, w/o a central node.

References

- Stankovic et al., "Strategic directions in real-time and embedded systems", ACM Computing Surveys, Vol. 28, No. 4, December 1996.
- Abdelzaher et al., "QoS negotiation in real-time systems and its application to automated flight control", Conference RTAS, 1997.
- Sanfridson, "Problem formulation for QoS management in automatic control", Technical report, TRITA-MMK 2000:3, KTH, March 2000.
- Österman, "Scalability for embedded distributed control systems", Master thesis, KTH, 2001.
- Sjunghamn, "QoS for embedded distributed control systems", Master thesis, Mitthögskolan, 2003.